

CoogNet

College of Technology: CIS 3365



Jorge Sanchez · Kavon Sabet · Ayoub Fares · Tyler Nullmeier · Daniel Howard · Aleena Khan
Aidahta Natama · Trent Jones · Eduardo Tostado

TABLE OF CONTENTS

<i>College of Technology: CIS 3365</i>	1
EXECUTIVE SUMMARY	3
CLIENT INFORMATION	4
BENEFITS AND COST	5
<i>Project Approach</i>	6
SOLUTION	7
TESTING PROCESS	9
PROJECT IMPROVEMENTS	11
PROJECT DATABASE MAINTENANCE ISSUES	12
LESSONS LEARNED	13
PROJECT SUMMARY	14
REFERENCES	15
<i>PowerPoint Slides</i>	16
APPENDIX A – ERD WITH PRIMARY KEYS AND RELATIONSHIPS ONLY	23
APPENDIX B – ERD WITH ALL ATTRIBUTES AND RELATIONSHIPS	24
APPENDIX C – DATA DICTIONARY	25
APPENDIX D – COPIES OF ALL QUERIES, FORMS, AND REPORTS	27
APPENDIX E – COPIES OF INSERT, UPDATE, & DELETE SCRIPTS	46
APPENDIX F – UPDATED PROBLEMS AND REQUIREMENTS LIST	99
APPENDIX G - Status Reports	101

EXECUTIVE SUMMARY

As a continuation of a two-semester long project, CoogNet decided to work with Jose Sanchez of TKS Studios to build a functional and cost-efficient application that would be capable of storing their client information. The scope of this project is based on the method in which TKS stores project information and generates reports for their clients. CoogNet has analyzed the current system and determined it is inefficient and lacking in functionality.

TKS Studios' current business problems include not having an information system in place. Currently, TKS Studios is using several off-the-shelf applications to complete their tasks including Quickbooks and Buildertrend. Buildertrend, however, does not facilitate the task of transferring data to Quickbooks in order to create invoices. Additionally, Buildertrend does not have a very smartphone-friendly interface which makes it difficult for TKS Studio to add or edit project information while on the construction site. On average, this takes an additional hour per project and if notes of on-site assessment are corrupted it could lead to errors.

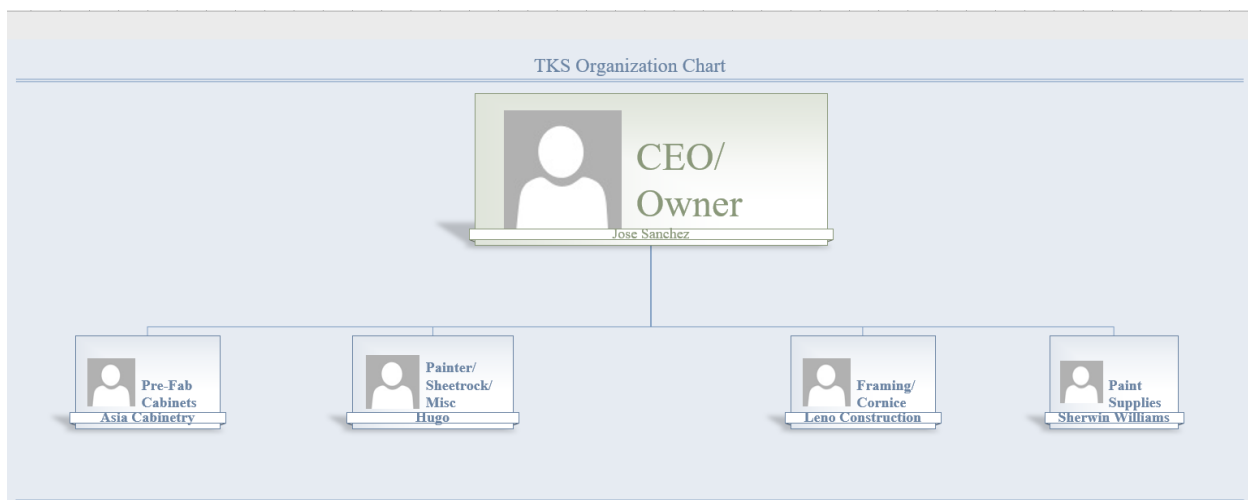
To resolve these obstacles, CoogNet's solution was to construct a reliable system that will be able to handle the capacity of their work. This project will be carried out by completing weekly deliverables to guarantee the success of the application.

The new application will allow the company to work on a variety of devices and can accomplish all the tasks required in TKS Studios' daily schedule. Once our clients enter a customer's information it will lead them to pre-defined automated forms that will record project details. This will allow TKS to further save time while entering information. Additionally, our solution will allow the generation of invoices directly, without Quickbooks as an intermediary.

CLIENT INFORMATION

TKS Studios was founded in July 2018, though, Jose Sanchez, the organization's founder and CEO, has been in the building and contracting industry for more than six years. Mr. Sanchez is skilled in residential construction, Computer-Aided Design (CAD), Project Management, and Budgeting. He is also a strong business development professional with an Architecture degree, a minor in Construction Management, and a certificate in entrepreneurship. At TKS Studios, they take great pride in their experience and expertise, and in the quality of their work as well as the customer service that they provide.

TKS Studios' work is concentrated on the importance of the small details that are required to produce a cohesive and purposeful design that is deserving of being built. Since their work does not stop at design, they can shepherd projects to completion, making sure that along the way nothing is overlooked, and details are improved. To understand the needs and expectations of their customers, they take great care to work and communicate with every customer in a personal and professional manner. They provide their customers with the tools, knowledge, and confidence to make the design/build process simple and enjoyable.



Required Economic Feasibility

The proposed system has the potential to reduce monthly expenditures by approximately 90%. Currently, Buildertrend costs our client \$90 per month or \$1080 annually. The proposed system, however, may cost closer to \$8 per month. Additionally, our client informed us that he tends to wait until he arrives back at the office to input information gathered in the field. We believe that this habit could be a costly misappropriation of time that we can curb by implementing an improved interface in the proposed system. The estimated costs for developing the new system are as follows:

- **Develop database with friendly user interface:** The current estimation of cost for our team's labor is approximately 130 hours to develop the database, application program and user interface for our client. At a rate of \$40/hr and 130 hours, the initial labor costs come to \$5,200. Once our student free-labor discount is applied, labor costs will be reduced to \$0.
- **Hosting database and web server online:** The cost for housing the system online will vary depending on the service provider; however, our research yielded an average of \$8 per month which totals \$96 annually.
- **Total:** Labor (\$5,200 * 9) + monthly fees (\$96) = \$46,896.00
- **Total after discount:** Only the monthly server fees, which are \$96 for the first year or \$8 per month.

Project Approach

From the start of the development of the application and database design, CoogNet's most important guidelines are the business rules, along with the problems and requirements listed by TKS Studios. Through these two guidelines, and by analyzing the current information systems of TKS Studios, we were able to determine the major entities and relationship data models. After identifying the major entities and relationships, we map out the primary and foreign keys based on the interactions that could exist between each entity. Once all the necessary information was gathered, we were able to develop the entity relationship diagram (ERD) which allowed us to visualize our data model and acted as a framework for a database. With the creation of the framework, we could further develop the ERD in detail by adding all the attributes required by the entities and relationships. Developing a solid framework of entities and relationships was our number one priority. We figured that if we could successfully identify all the entities and relationships, we could always add more attributes and functionality later. Once our ERD was finished, we began to select the software that would power our project's graphical user interface.

SOLUTION

The initial problem that our client was currently facing was that, our client was using a combination of applications in order to accomplish the tasks of invoicing and storing project and client information while offering additional unnecessary functions. Our current application includes the task listed above and adds additional functionalities requested by TKS Studios. The next request is to allow the application to be able to store, export, edit, update and delete client, contractor, invoicing and client information. Sample update query for FirstName of Contractor by Contractor ID:

```
UPDATE Contractor
SET Cont_FirstName = 'Joe'
WHERE Cont_ID = 1;
```

Before Query:

Imported Keys	Indexes	Privileges	Column Privileges	Row IDs	Version
Info	Content	Row Count	Columns	Primary Key	
Cont_ID	Cont_FirstName	Cont_LastName	Cont_Salary		
1	Greg	Gumpert	50000.0000		
2	Tony	Lambo	60000.0000		
3	Torstein	Muller	70000.0000		

After Query:

Imported Keys	Indexes	Privileges	Column Privileges	Row IDs	Ver
Info	Content	Row Count	Columns	Primary Key	
Cont_ID	Cont_FirstName	Cont_LastName	Cont_Salary		
1	Joe	Gumpert	50000.0000		
2	Tony	Lambo	60000.0000		
3	Torstein	Muller	70000.0000		

All key decisions from the original group design that lead to our current design had to do with the business rules, requirements, and also the table requirement of 35 or more tables for our

database. For example, one of the business rules is that TKS Studios would like to follow up on all project proposals that could potentially lead to a job. Project proposals are the initial phase of gaining contact information from a potential client and is represented as Job Proposal on the ERD. The Job Proposal has a many-to-one relationship with a Customer on the ERD which allows the formation of a relationship between customer and job proposal. For additional information required by a project lead, Job Proposal has a one-to-many relationship with Site Survey where the project managers upload notes for reviewing the potential project site. TKS Studios also requested that pictures be stored for future use, in order to accommodate this a one-to-many relationship between Site Survey and Survey Pictures was developed. After reviewing all this content and presenting a draft invoice the job lead could become a project site, which is represented in a one-to-one relationship in the ERD through Job Proposal and Job. Allowing our database design to cover all aspects of a project lead, and requirements of the business rules.

TESTING PROCESS

Purpose:

The purpose of the application building and testing phase is to analyze the efficiency, functionality, and accuracy of application functionalities. Using the project goals and the company's requirements as a baseline to ensure that our application satisfies the companies goals. The additional benefit of testing allows is to identify any current problems and potential problems with the continuous iterations of our application. Another purpose of our application testing is to ensure the integrity and availability of our database.

Objectives:

The objectives to be accomplished by testing the database, application, and interface are:

- 1) Ensure that interface, application, and database can successfully interact.
- 2) Ensure that the user can request and update data through the interface from the database
- 3) Ensure that the data entered in the interface is correctly inserted and stored in the database.
- 4) Ensure that the interface is simple and user-friendly.
- 5) Ensure that all company goals are successfully interpreted in our application.

Design:

The design part of this test is a two-part process. The first part dealt with the database side of our application and the other side concerned our user interface. We wanted to develop a simple interface with limited options for it to be easier to understand and learn. It is necessary that this interface be compatible and capable of executing tasks, finding tables quickly and living up to what TKS expects.

Importing/Exporting:

It is extremely important that we can import and export csv files into and out of the application. This was the first step we tested before progressing in the project. We will learn what is needed to achieve this and any difficulties we can overcome now versus during the development phase of the application. CSV files will need to be able to link to TKS Studios system software.

Database:

The database itself is the most important part of the entire project compared to the interface portion and needs to be tested very thoroughly. The information within the database must be easily manipulated and sorted in a logical order. The information within the database must be created, imported, updated, and deleted.

PROJECT IMPROVEMENTS

List of project improvements:

1. Increase granularity of customer invoices by adding a breakdown of the costs involved in each task.
2. Add more useful attributes to all tables.
3. Would add material count to job material count to Job_Material table so that it is clear on the expected use of each material that is required for a job.
4. Keeping track of vehicles that are not in accounts payable.
5. Add entities to keep track of the materials inventory.

After analyzing the current database design, many faults can be found within our database design and have listed the improvements that would be made by priority with one being the highest. For example, one of TKS Studios' goals is to be able to expand to a four-man company and own company vehicles. In order to allow this, the database includes accounts payable to be recorded through the application. Upon testing this feature, we have come to realize that after a vehicle is paid in full it will become an orphan in the database instead of being archived in a different location and being tagged as paid off.

PROJECT DATABASE MAINTENANCE ISSUES

List of ongoing maintenance procedures that should be conducted on an ongoing basis:

1. Empty rows that are created and insufficient data should be removed throughout the database.
2. Use bulk copy program utility in a batch file in order to automatically create a full backup of the database.
3. Use Robocopy script which allows for incremental backups.

TKS Studio is a home remodeling company that will receive many project proposals but not all will lead to a job. This will lead to having an increasingly growing amount of useless data and rows that will fill up the database. In order to prevent this, a data clean up method will be required. This can be accomplished by allowing a script to compare both the job proposals and job tables and delete unwanted data. Scripts, batch files, and utilities will allow TKS Studio to clean up tables, backup data, archive data, maintain the efficiency of lookup tables, recover data in case of potential disasters, and shrink the amount of data stored in the database.

LESSONS LEARNED

Throughout this project, our team learned so many important things that are helpful for future projects such as how to identify the problem by collecting and gathering data. This allows us to recognize the problem and find a solution. We have also learned how to organize tables and create SQL scripts. Additionally, we gained an understanding of how to model our data with entity relationship diagrams (ERD) and data dictionaries. Overall, however, the greatest lesson that we learned is that a well-designed database takes an immense amount of time, effort, and organization to create. Database design is an iterative process. While we are proud of the work we did on this project, we only managed to get to iterate through the database design process eight times. Given more time, without a doubt, our team could greatly expand upon the data model that we created.

PROJECT SUMMARY

After analyzing the business rules and requirements list our team was able to identify the major entities and relationships in order to have an efficient database design. With the basic data information established, our team began to construct the entity-relationship data model and began normalizing entity attributes. Having established our ERD, after normalization the primary and foreign keys became increasingly more obvious. This allowed our team to create a database design that would be efficient and fulfil all the required information needed for the future and current size of TKS Studios. Having created our design all that was left was to create our SQL scripts and develop an interface that would allow TKS Studio to easily update, delete, read and write any information into the database. Through testing and feedback from our client, our team was able to develop a custom and easy to use and efficient information system that can expand alongside the business it serves.

REFERENCES

- *Coronel, C. & Morris, S. (2019) Database Systems: Design, Implementation, & Management*
- *Sanchez, J. (2019, Feb 8). Personal Interview*
- *Azure SQL Database pricing. (n.d.). Retrieved from https://azure.microsoft.com/en-us/pricing/details/sql-database/managed/?&OCID=AID719825_SEM_WXBiicq6&lnkd=Google_Azure_Brand&gclid=CjwKCAjwjIHeBRAnEiwAhYT2h-d6Fpe7TjWqVpVu3EJ-8jtRzaAVdizGEoETXMmUCOTH9gGCNbmOEhoC7pkQAvD_BwE&dclid=CKXII Lvngd4CFQeTswodc6MN6w*
- *CREATE LOGIN (Transact-SQL). (2019). Retrieved from <https://docs.microsoft.com/en-us/sql/t-sql/statements/create-login-transact-sql?view=sql-server-ver15>*
- *CREATE PROCEDURE (Transact-SQL). (2019). Retrieved from <https://docs.microsoft.com/en-us/sql/t-sql/statements/create-procedure-transact-sql?view=sql-server-ver15>*
- *ALTER TABLE (Transact-SQL). (2019). Retrieved from <https://docs.microsoft.com/en-us/sql/t-sql/statements/alter-table-transact-sql?view=sql-server-ver15>*
- *Generate or Export the data dictionary of SQL Server database Schema using SQL Query. (2018). Retrieved from <https://sqlarts.blogspot.com/2017/12/generate-data-dictionary-for-sql.html>*



CoogNet

Jorge Sanchez · Kavon Sabet · Ayoub Fares · Tyler Nullmeier · Daniel Howard · Aleena Khan ·
Aidahta Natama · Trent Jones · Eduardo Tostado
CIS 3365 – fall '19

Outline



- Our Client
- background
- Client organization objective list
- Current system problem description
- System Proposal
- Required Feasibility Analysis
- ERD Demonstration
- Demonstration of project
- ERD design decision
- Questions



OUR CLIENT: TKS STUDIOS

Background

- TKS is a design build company
- TKS is known for Kitchen, Bathroom construction
- Founded in august 2018 as TKS STUDIOS
- By 2019, TKS had 30 projects so far
- Became Accredited by the BBB, GHBA, TAB, NAHB and The HDP



Client Organization Objective List

Over the next five years, our client wants to:

1. Increase productivity (\$100,000 to \$500,000 projects)
2. Add employees
3. Receive more referrals
4. Increase profit by \$50,000 each year



Current System Problem Description

- Unable to export data from Buildertrend application to create an invoice.
- No existing database to be able to store all information locally. It is all stored on the Buildertrend application.
- Insufficient tracking and record keeping of data.
- Paper-based and human-memory format are stifling organizational growth.
- New employee has difficulty accessing records and data due to paper system.

Systems Proposal

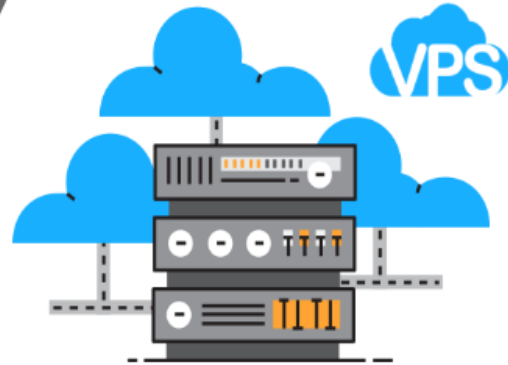
Option 1: Virtual Private Server Hosting

•Pros:

- No upfront hardware costs
- Headache-free expansion options
- Cheapest option

• Cons:

- Must be handled by a dedicated IT employee
- Less control over data



7

Required Feasibility Analysis

• Technical

- SQL server
- Virtual Private Server host
- CoogNet does not foresee any technical issues.

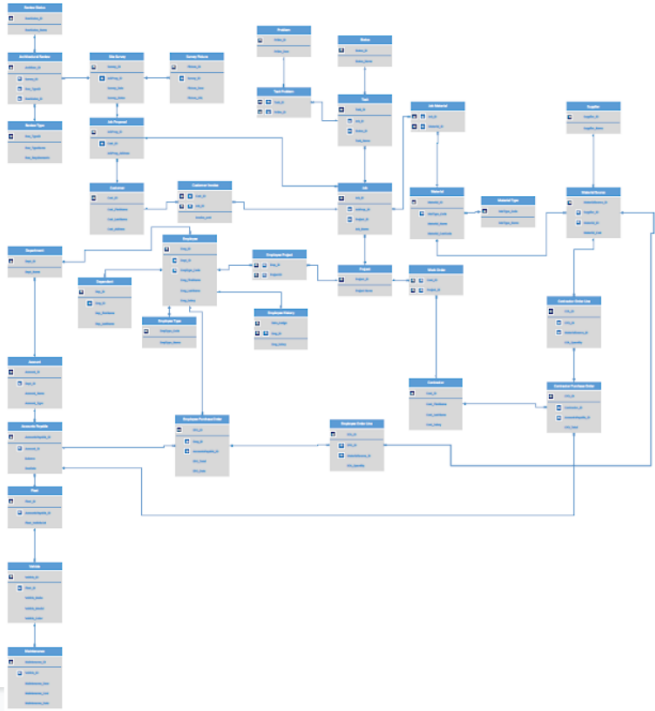
• Operational

- Mobile accessibility with user friendly application
- Easy to learn
- TKS will have the responsibility of encouraging the use of the system



8

ERD DEMONSTRATION



DEMONSTRATION OF PROJECT





ERD DESIGN DECISIONS

QUESTIONS?

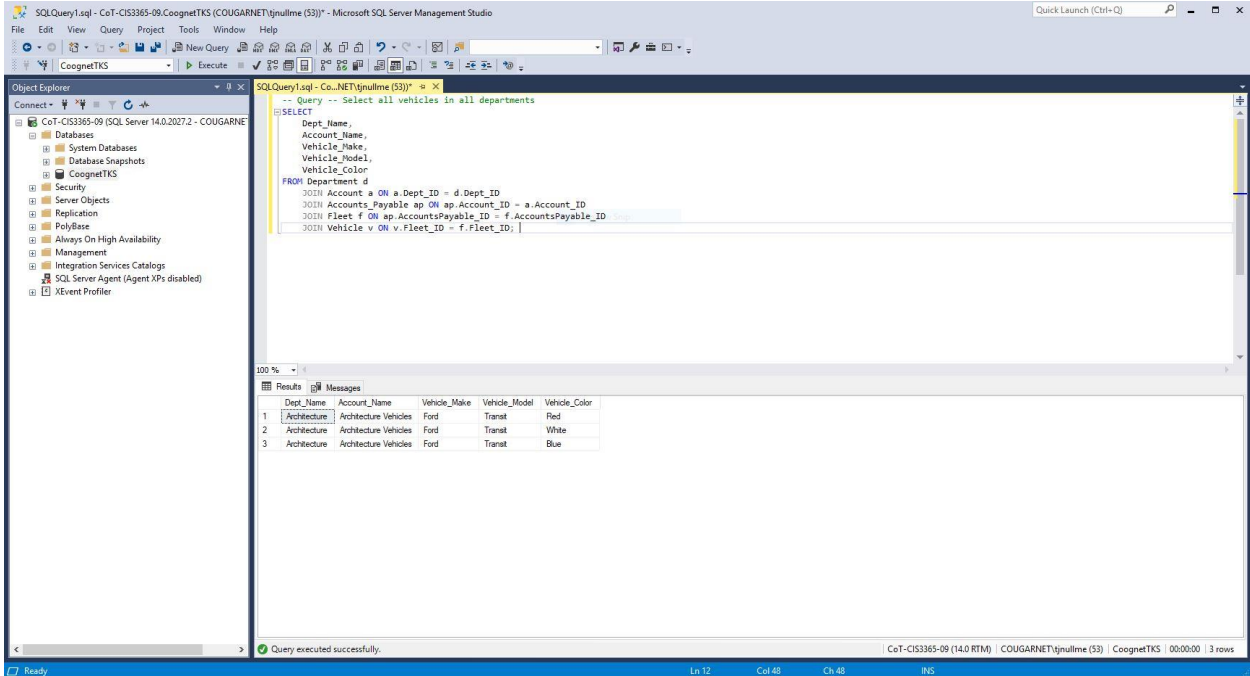


APPENDIX C – DATA DICTIONARY

Database Name	Schema	Table Name	Attribute Name	Data Type	Length	Precision	Scale	Is Nullable	Is PK	Is Indexed	Is Foreign Key	Parent Table	
CoognetTKS	dbo	Account	Account_ID	INT	4		10	0	FALSE	TRUE	TRUE	FALSE	NULL
CoognetTKS	dbo	Account	Dept_ID	INT	4		10	0	FALSE	FALSE	TRUE	TRUE	Department
CoognetTKS	dbo	Account	Account_Name	NVARCHAR	200		0	0	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Account	Account_Type	NVARCHAR	200		0	0	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Accounts_Payable	AccountsPayable_ID	INT	4		10	0	FALSE	TRUE	TRUE	FALSE	NULL
CoognetTKS	dbo	Accounts_Payable	Account_ID	INT	4		10	0	FALSE	FALSE	TRUE	TRUE	Account
CoognetTKS	dbo	Accounts_Payable	Balance	MONEY	8		19	4	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Accounts_Payable	DueDate	DATE	3		10	0	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Architectural_Review	ArchRvw_ID	INT	4		10	0	FALSE	TRUE	TRUE	FALSE	NULL
CoognetTKS	dbo	Architectural_Review	Survey_ID	INT	4		10	0	FALSE	FALSE	TRUE	TRUE	Site_Survey
CoognetTKS	dbo	Architectural_Review	Rvw_TypeID	INT	4		10	0	FALSE	FALSE	TRUE	TRUE	Review_Type
CoognetTKS	dbo	Architectural_Review	RvwStatus_ID	INT	4		10	0	FALSE	FALSE	TRUE	TRUE	Review_Status
CoognetTKS	dbo	Contractor	Cont_ID	INT	4		10	0	FALSE	TRUE	TRUE	FALSE	NULL
CoognetTKS	dbo	Contractor	Cont_FirstName	NVARCHAR	200		0	0	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Contractor	Cont_LastName	NVARCHAR	200		0	0	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Contractor	Cont_Salary	MONEY	8		19	4	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Contractor_Order_Line	COL_ID	INT	4		10	0	FALSE	TRUE	TRUE	FALSE	NULL
CoognetTKS	dbo	Contractor_Order_Line	CPO_ID	INT	4		10	0	FALSE	FALSE	TRUE	TRUE	Contractor_Purchase_Order
CoognetTKS	dbo	Contractor_Order_Line	MaterialSource_ID	INT	4		10	0	FALSE	FALSE	TRUE	TRUE	Material_Source
CoognetTKS	dbo	Contractor_Order_Line	COL_Quantity	INT	4		10	0	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Contractor_Purchase_Order	CPO_ID	INT	4		10	0	FALSE	TRUE	TRUE	FALSE	NULL
CoognetTKS	dbo	Contractor_Purchase_Order	Cont_ID	INT	4		10	0	FALSE	FALSE	TRUE	TRUE	Contractor
CoognetTKS	dbo	Contractor_Purchase_Order	AccountsPayable_ID	INT	4		10	0	FALSE	FALSE	TRUE	TRUE	Accounts_Payable
CoognetTKS	dbo	Contractor_Purchase_Order	CPO_Total	MONEY	8		19	4	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Contractor_Purchase_Order	CPO_Date	DATE	3		10	0	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Customer	Cust_ID	INT	4		10	0	FALSE	TRUE	TRUE	FALSE	NULL
CoognetTKS	dbo	Customer	Cust_FirstName	NVARCHAR	200		0	0	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Customer	Cust_LastName	NVARCHAR	200		0	0	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Customer	Cust_Address	NVARCHAR	200		0	0	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Customer_Invoice	Cust_ID	INT	4		10	0	FALSE	TRUE	TRUE	TRUE	Customer
CoognetTKS	dbo	Customer_Invoice	Job_ID	INT	4		10	0	FALSE	TRUE	TRUE	TRUE	Job
CoognetTKS	dbo	Customer_Invoice	Invoice_amt	MONEY	8		19	4	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Department	Dept_ID	INT	4		10	0	FALSE	TRUE	TRUE	FALSE	NULL
CoognetTKS	dbo	Department	Dept_Name	NVARCHAR	200		0	0	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Dependent	Dep_ID	INT	4		10	0	FALSE	TRUE	TRUE	FALSE	NULL
CoognetTKS	dbo	Dependent	Emp_ID	INT	4		10	0	FALSE	FALSE	TRUE	TRUE	Employee
CoognetTKS	dbo	Dependent	Dep_FirstName	NVARCHAR	200		0	0	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Dependent	Dep_LastName	NVARCHAR	200		0	0	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Employee	Emp_ID	INT	4		10	0	FALSE	TRUE	TRUE	FALSE	NULL
CoognetTKS	dbo	Employee	Dept_ID	INT	4		10	0	FALSE	FALSE	TRUE	TRUE	Department
CoognetTKS	dbo	Employee	EmpType_Code	INT	4		10	0	FALSE	FALSE	FALSE	TRUE	Employee_Type
CoognetTKS	dbo	Employee	Emp_FirstName	NVARCHAR	200		0	0	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Employee	Emp_LastName	NVARCHAR	200		0	0	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Employee	Emp_Salary	MONEY	8		19	4	TRUE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Employee_History	Date_Assign	DATE	3		10	0	FALSE	TRUE	TRUE	FALSE	NULL
CoognetTKS	dbo	Employee_History	Emp_Id	INT	4		10	0	FALSE	TRUE	TRUE	TRUE	Employee
CoognetTKS	dbo	Employee_History	Emp_Salary	MONEY	8		19	4	TRUE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Employee_Order_Line	EOL_ID	INT	4		10	0	FALSE	TRUE	TRUE	FALSE	NULL
CoognetTKS	dbo	Employee_Order_Line	EPO_ID	INT	4		10	0	FALSE	FALSE	TRUE	TRUE	Employee_Purchase_Order
CoognetTKS	dbo	Employee_Order_Line	MaterialSource_ID	INT	4		10	0	FALSE	FALSE	TRUE	TRUE	Material_Source
CoognetTKS	dbo	Employee_Order_Line	EOL_Quantity	INT	4		10	0	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Employee_Project	Emp_ID	INT	4		10	0	FALSE	TRUE	TRUE	TRUE	Employee
CoognetTKS	dbo	Employee_Project	Project_ID	INT	4		10	0	FALSE	TRUE	TRUE	TRUE	Project

CoognetTKS	dbo	Employee_Purchase_Order	EPO_ID	INT	4	10	0	FALSE	TRUE	TRUE	FALSE	NULL
CoognetTKS	dbo	Employee_Purchase_Order	Emp_ID	INT	4	10	0	FALSE	FALSE	TRUE	TRUE	Employee
CoognetTKS	dbo	Employee_Purchase_Order	AccountsPayable_ID	INT	4	10	0	FALSE	FALSE	TRUE	TRUE	Accounts_Payable
CoognetTKS	dbo	Employee_Purchase_Order	EPO_Total	MONEY	8	19	4	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Employee_Purchase_Order	EPO_Date	DATE	3	10	0	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Employee_Type	EmpType_Code	INT	4	10	0	FALSE	TRUE	TRUE	FALSE	NULL
CoognetTKS	dbo	Employee_Type	EmpType_Name	NVARCHAR	200	0	0	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Fleet	Fleet_ID	INT	4	10	0	FALSE	TRUE	TRUE	FALSE	NULL
CoognetTKS	dbo	Fleet	AccountsPayable_ID	INT	4	10	0	FALSE	FALSE	TRUE	TRUE	Accounts_Payable
CoognetTKS	dbo	Fleet	Fleet_VehicleCnt	INT	4	10	0	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Job	Job_ID	INT	4	10	0	FALSE	TRUE	TRUE	FALSE	NULL
CoognetTKS	dbo	Job	JobProp_ID	INT	4	10	0	FALSE	FALSE	TRUE	TRUE	Job_Proposal
CoognetTKS	dbo	Job	Project_ID	INT	4	10	0	FALSE	FALSE	TRUE	TRUE	Project
CoognetTKS	dbo	Job	Job_Name	NVARCHAR	200	0	0	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Job_Material	Job_ID	INT	4	10	0	FALSE	TRUE	TRUE	TRUE	Job
CoognetTKS	dbo	Job_Material	Material_ID	INT	4	10	0	FALSE	TRUE	TRUE	TRUE	Material
CoognetTKS	dbo	Job_Proposal	JobProp_ID	INT	4	10	0	FALSE	TRUE	TRUE	FALSE	NULL
CoognetTKS	dbo	Job_Proposal	Cust_ID	INT	4	10	0	FALSE	FALSE	TRUE	TRUE	Customer
CoognetTKS	dbo	Job_Proposal	JobProp_Address	NVARCHAR	200	0	0	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Maintenance	Maintenance_ID	INT	4	10	0	FALSE	TRUE	TRUE	FALSE	NULL
CoognetTKS	dbo	Maintenance	Vehicle_ID	INT	4	10	0	FALSE	FALSE	TRUE	TRUE	Vehicle
CoognetTKS	dbo	Maintenance	Maintenance_Desc	NVARCHAR	2000	0	0	TRUE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Maintenance	Maintenance_Cost	MONEY	8	19	4	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Maintenance	Maintenance_Date	DATE	3	10	0	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Material	Material_ID	INT	4	10	0	FALSE	TRUE	TRUE	FALSE	NULL
CoognetTKS	dbo	Material	MatType_Code	INT	4	10	0	FALSE	FALSE	FALSE	TRUE	Material_Type
CoognetTKS	dbo	Material	Material_Name	NVARCHAR	200	0	0	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Material	Material_CostCode	CHAR	5	0	0	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Material_Source	MaterialSource_ID	INT	4	10	0	FALSE	TRUE	TRUE	FALSE	NULL
CoognetTKS	dbo	Material_Source	Supplier_ID	INT	4	10	0	FALSE	FALSE	TRUE	TRUE	Supplier
CoognetTKS	dbo	Material_Source	Material_ID	INT	4	10	0	FALSE	FALSE	TRUE	TRUE	Material
CoognetTKS	dbo	Material_Source	Material_Cost	MONEY	8	19	4	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Material_Type	MatType_Code	INT	4	10	0	FALSE	TRUE	TRUE	FALSE	NULL
CoognetTKS	dbo	Material_Type	MatType_Name	NVARCHAR	200	0	0	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Problem	Prblm_ID	INT	4	10	0	FALSE	TRUE	TRUE	FALSE	NULL
CoognetTKS	dbo	Problem	Prblm_Desc	NVARCHAR	200	0	0	FALSE	FALSE	TRUE	FALSE	NULL
CoognetTKS	dbo	Project	Project_ID	INT	4	10	0	FALSE	TRUE	TRUE	FALSE	NULL
CoognetTKS	dbo	Project	Project_Name	NVARCHAR	200	0	0	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Review_Status	RvwStatus_ID	INT	4	10	0	FALSE	TRUE	TRUE	FALSE	NULL
CoognetTKS	dbo	Review_Status	RvwStatus_Name	NVARCHAR	300	0	0	FALSE	FALSE	TRUE	FALSE	NULL
CoognetTKS	dbo	Review_Type	Rvw_TypeID	INT	4	10	0	FALSE	TRUE	TRUE	FALSE	NULL
CoognetTKS	dbo	Review_Type	Rvw_TypeName	NVARCHAR	300	0	0	FALSE	FALSE	TRUE	FALSE	NULL
CoognetTKS	dbo	Review_Type	Rvw_Requirements	NVARCHAR	2000	0	0	TRUE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Site_Survey	Survey_ID	INT	4	10	0	FALSE	TRUE	TRUE	FALSE	NULL
CoognetTKS	dbo	Site_Survey	JobProp_ID	INT	4	10	0	FALSE	FALSE	TRUE	TRUE	Job_Proposal
CoognetTKS	dbo	Site_Survey	Survey_Date	DATE	3	10	0	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Site_Survey	Survey_Notes	NVARCHAR	2000	0	0	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Status	Status_ID	INT	4	10	0	FALSE	TRUE	TRUE	FALSE	NULL
CoognetTKS	dbo	Status	Status_Name	NVARCHAR	200	0	0	FALSE	FALSE	TRUE	FALSE	NULL
CoognetTKS	dbo	Supplier	Supplier_ID	INT	4	10	0	FALSE	TRUE	TRUE	FALSE	NULL
CoognetTKS	dbo	Supplier	Supplier_Name	NVARCHAR	200	0	0	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Survey_Picture	Picture_ID	INT	4	10	0	FALSE	TRUE	TRUE	FALSE	NULL
CoognetTKS	dbo	Survey_Picture	Survey_ID	INT	4	10	0	FALSE	FALSE	TRUE	TRUE	Site_Survey
CoognetTKS	dbo	Survey_Picture	Picture_Desc	NVARCHAR	2000	0	0	TRUE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Survey_Picture	Picture_URL	NVARCHAR	510	0	0	TRUE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Task	Task_ID	INT	4	10	0	FALSE	TRUE	TRUE	FALSE	NULL
CoognetTKS	dbo	Task	Job_ID	INT	4	10	0	FALSE	FALSE	TRUE	TRUE	Job
CoognetTKS	dbo	Task	Status_ID	INT	4	10	0	FALSE	FALSE	FALSE	TRUE	Status
CoognetTKS	dbo	Task	Task_Name	NVARCHAR	200	0	0	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Task_Problem	Task_ID	INT	4	10	0	FALSE	TRUE	TRUE	TRUE	Task
CoognetTKS	dbo	Task_Problem	Prblm_ID	INT	4	10	0	FALSE	TRUE	TRUE	TRUE	Problem
CoognetTKS	dbo	Vehicle	Vehicle_ID	INT	4	10	0	FALSE	TRUE	TRUE	FALSE	NULL
CoognetTKS	dbo	Vehicle	Fleet_ID	INT	4	10	0	FALSE	FALSE	TRUE	TRUE	Fleet
CoognetTKS	dbo	Vehicle	Vehicle_Make	NVARCHAR	200	0	0	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Vehicle	Vehicle_Model	NVARCHAR	200	0	0	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Vehicle	Vehicle_Color	NVARCHAR	200	0	0	FALSE	FALSE	FALSE	FALSE	NULL
CoognetTKS	dbo	Work_Order	Cont_ID	INT	4	10	0	FALSE	TRUE	TRUE	TRUE	Contractor
CoognetTKS	dbo	Work_Order	Project_ID	INT	4	10	0	FALSE	TRUE	TRUE	TRUE	Project

APPENDIX D – COPIES OF ALL QUERIES, FORMS, AND REPORTS



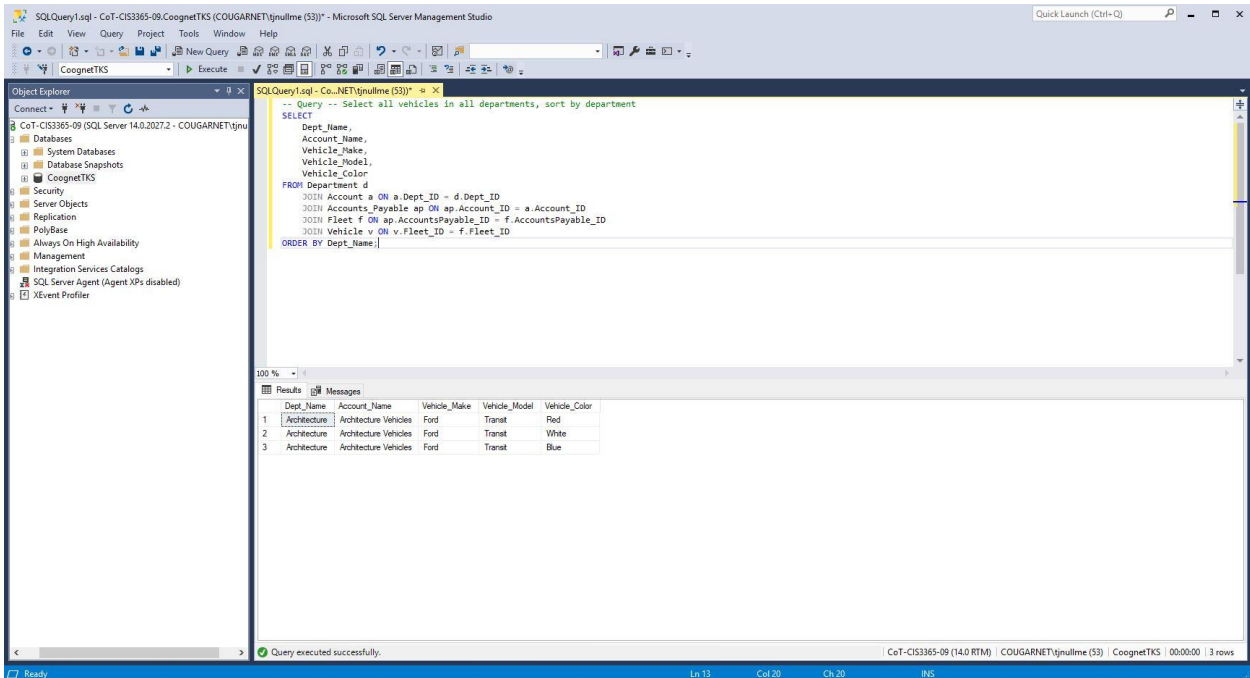
SQLQuery1.sql - CoT-CIS3365-09.CoognetTKS (COUGARNET\jnullme (53)) - Microsoft SQL Server Management Studio

```
-- Query -- Select all vehicles in all departments
SELECT
    Dept_Name,
    Account_Name,
    Vehicle_Make,
    Vehicle_Model,
    Vehicle_Color
FROM Department d
JOIN Account a ON a.Dept_ID = d.Dept_ID
JOIN Accounts_Payable ap ON ap.Account_ID = a.Account_ID
JOIN Fleet f ON ap.AccountsPayable_ID = f.AccountsPayable_ID
JOIN Vehicle v ON v.Fleet_ID = f.Fleet_ID;
```

Dept_Name	Account_Name	Vehicle_Make	Vehicle_Model	Vehicle_Color
Architecture	Architecture Vehicles	Ford	Transat	Red
Architecture	Architecture Vehicles	Ford	Transat	White
Architecture	Architecture Vehicles	Ford	Transat	Blue

Query executed successfully. CoT-CIS3365-09 (14.0 RTM) COUGARNET\jnullme (53) CoognetTKS 00:00:00 | 3 rows

Query 1- Select all Vehicles in all departments



SQLQuery1.sql - CoT-CIS3365-09.CoognetTKS (COUGARNET\jnullme (53)) - Microsoft SQL Server Management Studio

```
-- Query -- Select all vehicles in all departments, sort by department
SELECT
    Dept_Name,
    Account_Name,
    Vehicle_Make,
    Vehicle_Model,
    Vehicle_Color
FROM Department d
JOIN Account a ON a.Dept_ID = d.Dept_ID
JOIN Accounts_Payable ap ON ap.Account_ID = a.Account_ID
JOIN Fleet f ON ap.AccountsPayable_ID = f.AccountsPayable_ID
JOIN Vehicle v ON v.Fleet_ID = f.Fleet_ID
ORDER BY Dept_Name;
```

Dept_Name	Account_Name	Vehicle_Make	Vehicle_Model	Vehicle_Color
Architecture	Architecture Vehicles	Ford	Transat	Red
Architecture	Architecture Vehicles	Ford	Transat	White
Architecture	Architecture Vehicles	Ford	Transat	Blue

Query executed successfully. CoT-CIS3365-09 (14.0 RTM) COUGARNET\jnullme (53) CoognetTKS 00:00:00 | 3 rows

Query 2-

SQLQuery1.sql - CoT-CIS3365-09.CoognetTKS (COUGARNET\gnulme (53)) - Microsoft SQL Server Management Studio

```

-- Query -- Select all vehicles in one department
SELECT
    Dept_Name,
    Account_Name,
    Vehicle_Make,
    Vehicle_Model,
    Vehicle_Color
FROM Department d
    JOIN Account a ON a.Dept_ID = d.Dept_ID
    JOIN Accounts_Payable ap ON ap.Account_ID = a.Account_ID
    JOIN Fleet f ON ap.AccountsPayable_ID = f.AccountsPayable_ID
    JOIN Vehicle v ON v.Fleet_ID = f.Fleet_ID
WHERE Dept_Name = 'Architecture';

```

Dept_Name	Account_Name	Vehicle_Make	Vehicle_Model	Vehicle_Color
Architecture	Architecture Vehicles	Ford	Transit	Red
Architecture	Architecture Vehicles	Ford	Transit	White
Architecture	Architecture Vehicles	Ford	Transit	Blue

Query executed successfully.

Query 4-

SQLQuery1.sql - CoT-CIS3365-09.CoognetTKS (COUGARNET\gnulme (53)) - Microsoft SQL Server Management Studio

```

-- Query -- Find out which departments have a specific type of vehicle
SELECT DISTINCT(Dept_Name)
FROM Department d
    JOIN Account a ON a.Dept_ID = d.Dept_ID
    JOIN Accounts_Payable ap ON ap.Account_ID = a.Account_ID
    JOIN Fleet f ON ap.AccountsPayable_ID = f.AccountsPayable_ID
    JOIN Vehicle v ON v.Fleet_ID = f.Fleet_ID
WHERE Vehicle_Make = 'Ford' AND Vehicle_Model = 'Transit';

```

Dept_Name
Architecture

Query executed successfully.

Query 5

The screenshot displays the Microsoft SQL Server Management Studio interface. The main window shows a SQL query designed to find vehicles in accounts payable that have been under maintenance in the last x years. The query uses a series of JOINs to connect the Department, Account, Fleet, and Vehicle tables, and filters for maintenance records within a specific date range.

```
-- Query -- Find all vehicles in accounts payable that have been under maintenance in the last x years
GO
SELECT
    Dept_Name,
    v.Vehicle_ID,
    Vehicle_Make,
    Vehicle_Model,
    Maintenance_Desc,
    Maintenance_Cost,
    Maintenance_Date
FROM Department d
JOIN Account a ON a.Dept_ID = d.Dept_ID
JOIN Accounts_Payable ap ON ap.Account_ID = a.Account_ID
JOIN Fleet f ON ap.Accounts_Payable_ID = f.Accounts_Payable_ID
JOIN Vehicle v ON v.Fleet_ID = f.Fleet_ID
JOIN Maintenance m ON m.Vehicle_ID = v.Vehicle_ID
WHERE Maintenance_Date > DATEADD(year, -1, GETDATE());
```

The results pane shows a single row of data:

Dept_Name	Vehicle_ID	Vehicle_Make	Vehicle_Model	Maintenance_Desc	Maintenance_Cost	Maintenance_Date
Architecture	1	Ford	Transit	Oil Change	20.34	2019-11-06

The status bar at the bottom indicates that the query was executed successfully and returned 1 row.

Query 6

SQLQuery2.sql - CoT-CIS3365-09.CoognetTKS (COUGARNET\jnullme (52)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Object Explorer

CoT-CIS3365-09 (SQL Server 14.0.2027.2 - COUGARNET\jnu

SQLQuery2.sql - Co-NET\jnullme (52) * SQLQuery1.sql - Co-NET\jnullme (53) *

```
-- Query 13 -- Select a specific department and information about the materials they have bought in the last x years
SELECT
    Dept_Name,
    Emp_FirstName + ' ' + Emp_LastName AS Emp_Name,
    EmpType_Name,
    EOL_Quantity,
    Supplier_Name,
    MatType_Name,
    Material_Name
FROM Department d
JOIN Employee e ON e.Dept_ID = d.Dept_ID
JOIN Employee_Type et ON et.EmpType_Code = e.EmpType_Code
JOIN Employee_Purchase_Order epo ON epo.Emp_ID = e.Emp_ID
JOIN Employee_Order_Line eol ON eol.EPO_ID = epo.EPO_ID
JOIN Material_Source ms ON eol.MaterialSource_ID = ms.MaterialSource_ID
JOIN Supplier s ON s.Supplier_ID = ms.Supplier_ID
JOIN Material m ON ms.Material_ID = m.Material_ID
JOIN Material_Type mt ON mt.MatType_Code = m.MatType_Code
WHERE Dept_Name = 'Architecture' AND EPO_Date > DATETIME(year, -5, GETDATE());
```

Results Messages

Dept_Name	Emp_Name	EmpType_Name	EOL_Quantity	Supplier_Name	MatType_Name	Material_Name
Architecture	Albert Bandura Carpenter		4	Lowes	Porcelain Tile	12x 24 inch

Query executed successfully. CoT-CIS3365-09 (14.0 RTM) COUGARNET\jnullme (52) CoognetTKS 00:00:00 | 1 rows

SQLQuery2.sql - CoT-CIS3365-09.CoognetTKS (COUGARNET\jnullme (52)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Object Explorer

CoT-CIS3365-09 (SQL Server 14.0.2027.2 - COUGARNET\jnu

SQLQuery2.sql - Co-NET\jnullme (52) * SQLQuery1.sql - Co-NET\jnullme (53) *

```
-- Query 12 -- Select all employees and information about the materials they have bought in the last x years
SELECT
    Dept_Name,
    Emp_FirstName + ' ' + Emp_LastName AS Emp_Name,
    EmpType_Name,
    EOL_Quantity,
    Supplier_Name,
    MatType_Name,
    Material_Name
FROM Department d
JOIN Employee e ON e.Dept_ID = d.Dept_ID
JOIN Employee_Type et ON et.EmpType_Code = e.EmpType_Code
JOIN Employee_Purchase_Order epo ON epo.Emp_ID = e.Emp_ID
JOIN Employee_Order_Line eol ON eol.EPO_ID = epo.EPO_ID
JOIN Material_Source ms ON eol.MaterialSource_ID = ms.MaterialSource_ID
JOIN Supplier s ON s.Supplier_ID = ms.Supplier_ID
JOIN Material m ON ms.Material_ID = m.Material_ID
JOIN Material_Type mt ON mt.MatType_Code = m.MatType_Code
WHERE EPO_Date > DATETIME(year, -10, GETDATE());
```

Results Messages

Dept_Name	Emp_Name	EmpType_Name	EOL_Quantity	Supplier_Name	MatType_Name	Material_Name
Information Technology	Robert Powell IT Specialist		5	Home Depot	Porcelain Tile	12 x 24 inch
Information Technology	Robert Powell IT Specialist		12	Lowes	PVC Pipe	3 1/4 inch
Architecture	Albert Bandura Carpenter		4	Lowes	Porcelain Tile	12 x 24 inch

Query executed successfully. CoT-CIS3365-09 (14.0 RTM) COUGARNET\jnullme (52) CoognetTKS 00:00:00 | 3 rows

SQLQuery1.sql - CoT-CIS3365-09.CoognetTKS (COUGARNET\jnullme (53)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

CoognetTKS

Object Explorer

CoT-CIS3365-09 (SQL Server 14.0.20272 - COUGARNET\jnullme)

Databases

System Databases

Database Snapshots

CoognetTKS

Security

Server Objects

Replication

PolyBase

Always On High Availability

Management

Integration Services Catalogs

SQL Server Agent (Agent XPs disabled)

XEvent Profiler

```

-- Query -- Select all employees and their salary assignments in the last x years
SELECT
    Dept_Name,
    Emp_FirstName + ' ' + Emp_LastName AS Emp_Name,
    EmpType_Name,
    Date_Assign,
    eh.Emp_Salary
FROM Department d
    JOIN Employee e ON e.Dept_ID = d.Dept_ID
    JOIN Employee_Type et ON et.EmpType_Code = e.EmpType_Code
    JOIN Employee_History eh ON eh.Emp_ID = e.Emp_ID
WHERE Date_Assign > DATEADD(year, -5, GETDATE());

```

Results Messages

Dept_Name	Emp_Name	EmpType_Name	Date_Assign	Emp_Salary
Information Technology	Robert Powell	IT Specialist	2016-05-10	75000.00
Information Technology	Robert Powell	IT Specialist	2018-05-10	80000.00
Architecture	Abert Bandura	Carpenter	2016-05-10	75000.00
Architecture	Jones Anable	Electrician	2016-05-10	80000.00

Query executed successfully.

CoT-CIS3365-09 (14.0 RTM) COUGARNET\jnullme (53) CoognetTKS | 00:00:00 | 4 rows

SQLQuery1.sql - CoT-CIS3365-09.CoognetTKS (COUGARNET\jnullme (53)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

CoognetTKS

Object Explorer

CoT-CIS3365-09 (SQL Server 14.0.20272 - COUGARNET\jnullme)

Databases

System Databases

Database Snapshots

CoognetTKS

Security

Server Objects

Replication

PolyBase

Always On High Availability

Management

Integration Services Catalogs

SQL Server Agent (Agent XPs disabled)

XEvent Profiler

```

-- Query -- Select one employee and their dependents
SELECT
    Dept_Name,
    Emp_FirstName + ' ' + Emp_LastName AS Emp_Name,
    EmpType_Name,
    Dep_FirstName + ' ' + Dep_LastName AS Dep_Name
FROM Department d
    JOIN Employee e ON e.Dept_ID = d.Dept_ID
    JOIN Employee_Type et ON et.EmpType_Code = e.EmpType_Code
    JOIN Dependent dep ON dep.Emp_ID = e.Emp_ID
WHERE Emp_FirstName = 'Abert' AND Emp_LastName = 'Bandura';

```

Results Messages

Dept_Name	Emp_Name	EmpType_Name	Dep_Name
Architecture	Abert Bandura	Carpenter	Pichy Bandura

Query executed successfully.

CoT-CIS3365-09 (14.0 RTM) COUGARNET\jnullme (53) CoognetTKS | 00:00:00 | 1 rows

SQLQuery1.sql - CoT-CIS3365-09.CoognetTKS (COUGARNET\jnullme (53)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

CoognetTKS

Object Explorer

- CoT-CIS3365-09 (SQL Server 14.0.2027.2 - COUGARNET\jnullme)
 - Databases
 - System Databases
 - Database Snapshots
 - CoognetTKS
 - Security
 - Server Objects
 - Replication
 - PolyBase
 - Always On High Availability
 - Management
 - Integration Services Catalogs
 - SQL Server Agent (Agent XPs disabled)
 - XEvent Profiler

```

-- Query -- Select all employees of a specific type and their dependents
SELECT
  Dept_Name,
  Emp_FirstName + ' ' + Emp_LastName AS Emp_Name,
  EmpType_Name,
  Dep_FirstName + ' ' + Dep_LastName AS Dep_Name
FROM Department d
JOIN Employee e ON e.Dept_ID = d.Dept_ID
JOIN Employee_Type et ON et.EmpType_Code = e.EmpType_Code
JOIN Dependent dep ON dep.Emp_ID = e.Emp_ID
WHERE EmpType_Name = 'Carpenter';

```

Results

Dept_Name	Emp_Name	EmpType_Name	Dep_Name
Architecture	Abet Bandura	Carpenter	Poty Bandura

Query executed successfully.

CoT-CIS3365-09 (14.0 RTM) COUGARNET\jnullme (53) CoognetTKS | 00:00:00 | 1 rows

SQLQuery1.sql - CoT-CIS3365-09.CoognetTKS (COUGARNET\jnullme (53)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

CoognetTKS

Object Explorer

- CoT-CIS3365-09 (SQL Server 14.0.2027.2 - COUGARNET\jnullme)
 - Databases
 - System Databases
 - Database Snapshots
 - CoognetTKS
 - Security
 - Server Objects
 - Replication
 - PolyBase
 - Always On High Availability
 - Management
 - Integration Services Catalogs
 - SQL Server Agent (Agent XPs disabled)
 - XEvent Profiler

```

-- Query -- Select all employees and their dependents in one department
SELECT
  Dept_Name,
  Emp_FirstName + ' ' + Emp_LastName AS Emp_Name,
  EmpType_Name,
  Dep_FirstName + ' ' + Dep_LastName AS Dep_Name
FROM Department d
JOIN Employee e ON e.Dept_ID = d.Dept_ID
JOIN Employee_Type et ON et.EmpType_Code = e.EmpType_Code
JOIN Dependent dep ON dep.Emp_ID = e.Emp_ID
WHERE Dept_Name = 'Information Technology';

```

Results

Dept_Name	Emp_Name	EmpType_Name	Dep_Name
Information Technology	Robert Powell	IT Specialist	Timmy Powell
Information Technology	Robert Powell	IT Specialist	Abby Powell

Query executed successfully.

CoT-CIS3365-09 (14.0 RTM) COUGARNET\jnullme (53) CoognetTKS | 00:00:00 | 2 rows

SQLQuery1.sql - CoT-CIS3365-09.CoognetTKS (COUGARNET\jnullme (53)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

CoognetTKS

Object Explorer

- CoT-CIS3365-09 (SQL Server 14.0.2027.2 - COUGARNET\jnullme)
- Databases
 - System Databases
 - Database Snapshots
 - CoognetTKS
 - Security
 - Server Objects
 - Replication
 - PolyBase
 - Always On High Availability
 - Management
 - Integration Services Catalogs
 - SQL Server Agent (Agent XPs disabled)
 - XEvent Profiler

```
-- Query -- Select all employees and their dependents per department
GO
SELECT
    Dept_Name,
    Emp_FirstName + ' ' + Emp_LastName AS Emp_Name,
    EmpType_Name,
    Dep_FirstName + ' ' + Dep_LastName AS Dep_Name
FROM Department d
JOIN Employee e ON e.Dept_ID = d.Dept_ID
JOIN Employee_Type et ON et.EmpType_Code = e.EmpType_Code
JOIN Dependent dep ON dep.Emp_ID = e.Emp_ID;
```

100 %

Dept_Name	Emp_Name	EmpType_Name	Dep_Name
Information Technology	Robert Powell	IT Specialist	Tommy Powell
Information Technology	Robert Powell	IT Specialist	Abby Powell
Architecture	Abnet Bandura	Carpenter	Richy Bandura

Query executed successfully.

CoT-CIS3365-09 (14.0 RTM) | COUGARNET\jnullme (53) | CoognetTKS | 00:00:00 | 3 rows

Ready | Ln:10 | Col:41 | Ch:41 | INS

SQLQuery2.sql - CoT-CIS3365-09.CoognetTKS (COUGARNET\jnullme (S2)) - Microsoft SQL Server Management Studio

Object Explorer: CoT-CIS3365-09 (SQL Server 14.0.2027.2 - COUGARNET\jnu)

```

-- Query 14 -- Select a specific employee and information about the materials they have bought in the last x years
SELECT
    Dept_Name,
    Emp_FirstName + ' ' + Emp_LastName AS Emp_Name,
    EmpType_Name,
    EOL_Quantity,
    Supplier_Name,
    MatType_Name,
    Material_Name
FROM Department d
JOIN Employee e ON e.Dept_ID = d.Dept_ID
JOIN Employee_Type et ON et.EmpType_Code = e.EmpType_Code
JOIN Employee_Purchase_Order epo ON epo.Emp_ID = e.Emp_ID
JOIN Employee_Order_Line eol ON eol.EPO_ID = epo.EPO_ID
JOIN Material_Source ms ON eol.MaterialSource_ID = ms.MaterialSource_ID
JOIN Supplier s ON s.Supplier_ID = ms.Supplier_ID
JOIN Material m ON ms.Material_ID = m.Material_ID
JOIN Material_Type et ON et.MatType_Code = m.MatType_Code
WHERE Emp_FirstName = 'Albert' AND Emp_LastName = 'Bandura' AND EPO_Date > DATEADD(year, -3, GETDATE());

```

Results

Dept_Name	Emp_Name	EmpType_Name	EOL_Quantity	Supplier_Name	MatType_Name	Material_Name
Architecture	Abet Bandura	Carpenter	4	Loves	Porcelain Tile	12x24 inch

Query executed successfully.

SQLQuery1.sql - CoT-CIS3365-09.CoognetTKS (COUGARNET\jnullme (S3)) - Microsoft SQL Server Management Studio

Object Explorer: CoT-CIS3365-09 (SQL Server 14.0.2027.2 - COUGARNET\jnu)

```

-- Query -- Find all vehicles in accounts payable that have been under maintenance in the last x years
SELECT
    Dept_Name,
    v.Vehicle_ID,
    Vehicle_Make,
    Vehicle_Model,
    Maintenance_Desc,
    Maintenance_Cost,
    Maintenance_Date
FROM Department d
JOIN Account a ON a.Dept_ID = d.Dept_ID
JOIN Accounts_Payable ap ON ap.Account_ID = a.Account_ID
JOIN Fleet f ON ap.AccountsPayable_ID = f.AccountsPayable_ID
JOIN Vehicle v ON v.Fleet_ID = f.Fleet_ID
JOIN Maintenance m ON m.Vehicle_ID = v.Vehicle_ID
WHERE Maintenance_Date > DATEADD(year, -1, GETDATE());

```

Results

Dept_Name	Vehicle_ID	Vehicle_Make	Vehicle_Model	Maintenance_Desc	Maintenance_Cost	Maintenance_Date
Architecture	1	Ford	Transit	Oil Change	20.34	2019-11-06

Query executed successfully.

SQLQuery2.sql - CoT-CIS3365-09.CoognetTKS (COUGARNET\jnullme (52)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

CoognetTKS

Object Explorer

Connect - CoT-CIS3365-09 (SQL Server 14.0.20272 - COUGARNET\jnullme (52))

Databases

- System Databases
- Database Snapshots
- CoognetTKS
- Security
- Server Objects
- Replication
- PolyBase
- Always On High Availability
- Management
- Integration Services Catalogs
- SQL Server Agent (Agent XPs disabled)
- XEvent Profiler

```

-- Query 36 -- Find out how many vehicles a specific department has in accounts payable
SELECT
    Dept_Name,
    COUNT (Vehicle_ID) AS Vehicle_Count
FROM Department d
    JOIN Account a ON a.Dept_ID = d.Dept_ID
    JOIN Accounts_Payable ap ON ap.Account_ID = a.Account_ID
    JOIN Fleet f ON ap.AccountsPayable_ID = f.AccountsPayable_ID
    JOIN Vehicle v ON v.Fleet_ID = f.Fleet_ID
WHERE d.Dept_ID = 1
GROUP BY Dept_Name;

```

100 %

Results Messages

Dept_Name	Vehicle_Count
-----------	---------------

Query executed successfully.

CoT-CIS3365-09 (14.0 RTM) | COUGARNET\jnullme (52) | CoognetTKS | 00:00:00 | 0 rows

SQLQuery2.sql - CoT-CIS3365-09.CoognetTKS (COUGARNET\jnullme (52)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

CoognetTKS

Object Explorer

Connect - CoT-CIS3365-09 (SQL Server 14.0.20272 - COUGARNET\jnullme (52))

Databases

- System Databases
- Database Snapshots
- CoognetTKS
- Security
- Server Objects
- Replication
- PolyBase
- Always On High Availability
- Management
- Integration Services Catalogs
- SQL Server Agent (Agent XPs disabled)
- XEvent Profiler

```

-- Query 35 -- Find out how many vehicles each department has in accounts payable
SELECT
    Dept_Name,
    COUNT (Vehicle_ID) AS Vehicle_Count
FROM Department d
    JOIN Account a ON a.Dept_ID = d.Dept_ID
    JOIN Accounts_Payable ap ON ap.Account_ID = a.Account_ID
    JOIN Fleet f ON ap.AccountsPayable_ID = f.AccountsPayable_ID
    JOIN Vehicle v ON v.Fleet_ID = f.Fleet_ID
GROUP BY d.Dept_ID, Dept_Name;

```

100 %

Results Messages

Dept_Name	Vehicle_Count
Architecture	3

Query executed successfully.

CoT-CIS3365-09 (14.0 RTM) | COUGARNET\jnullme (52) | CoognetTKS | 00:00:00 | 1 rows

SQLQuery2.sql - CoT-CIS3365-09.CoognetTKS (COUGARNET\jnullme (52)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

CoognetTKS

Object Explorer

Connect - CoT-CIS3365-09 (SQL Server 14.0.20272 - COUGARNET\jnullme)

Databases

- System Databases
- Database Snapshots
- CoognetTKS
- Security
- Server Objects
- Replication
- PolyBase
- Always On High Availability
- Management
- Integration Services Catalogs
- SQL Server Agent (Agent XPs disabled)
- XEvent Profiler

SQLQuery2.sql - Co-NET\jnullme (52) * SQLQuery1.sql - Co-NET\jnullme (53) *

```
-- Query 34 -- Find out which departments are on a specific job
SELECT
FROM Department d
JOIN Employee e ON e.Dept_ID = d.Dept_ID
JOIN Employee_Project ep ON ep.Emp_ID = e.Emp_ID
JOIN Project p ON p.Project_ID = ep.Project_ID
JOIN Job j ON j.Project_ID = p.Project_ID
WHERE Job_Name = 'Smith 4465 Bambert'
GROUP BY d.Dept_ID, Dept_Name, Job_Name
```

Results Messages

Dept_Name	Job_Name
Architecture	Smith 4465 Bambert

Query executed successfully.

CoT-CIS3365-09 (14.0 RTM) COUGARNET\jnullme (52) CoognetTKS 00:00:00 | 1 rows

SQLQuery2.sql - CoT-CIS3365-09.CoognetTKS (COUGARNET\jnullme (52)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

CoognetTKS

Object Explorer

Connect - CoT-CIS3365-09 (SQL Server 14.0.20272 - COUGARNET\jnullme)

Databases

- System Databases
- Database Snapshots
- CoognetTKS
- Security
- Server Objects
- Replication
- PolyBase
- Always On High Availability
- Management
- Integration Services Catalogs
- SQL Server Agent (Agent XPs disabled)
- XEvent Profiler

SQLQuery2.sql - Co-NET\jnullme (52) * SQLQuery1.sql - Co-NET\jnullme (53) *

```
-- Query 33 -- Connect customer to job to customer Invoice
SELECT
Cust_FirstName + ' ' + Cust_LastName AS Cust_Name,
Job_Name,
JobProp_Address,
Invoice_amt
FROM Customer c
JOIN Job_Proposal jp ON jp.Cust_ID = c.Cust_ID
JOIN Job j ON j.JobProp_ID = jp.JobProp_ID
JOIN Customer_Invoice ci ON ci.Job_ID = j.Job_ID
WHERE Cust_FirstName = 'Anthony' AND Cust_LastName = 'Smith' AND Cust_Address = '4465 Bambert St, Wild Horse, LA 12345'
```

Results Messages

Cust_Name	Job_Name	JobProp_Address	Invoice_amt
Anthony Smith	Smith 4465 Bambert	4465 Bambert St, Wild Horse, LA 12345	3512.25
Anthony Smith	Smith 5565 Bambert	5565 Bambert St, Wild Horse, LA 12345	5501.89

Query executed successfully.

CoT-CIS3365-09 (14.0 RTM) COUGARNET\jnullme (52) CoognetTKS 00:00:00 | 2 rows

SQLQuery2.sql - CoT-CIS3365-09.CoognetTKS (COUGARNET\jnullme (52)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Object Explorer

CoT-CIS3365-09 (SQL Server 14.0.2027.2 - COUGARNET\jnullme (52))

SQLQuery2.sql - Co.NET\jnullme (52)*

```
-- Query 32 -- Find what jobs a contractor is assigned to
SELECT
    Cont_FirstName + ' ' + Cont_LastName AS Cont_Name,
    Project_Name,
    Job_Name
FROM Contractor c
JOIN Work_Order wo ON wo.Cont_ID = c.Cont_ID
JOIN Project p ON p.Project_ID = wo.Project_ID
JOIN Job j ON j.Project_ID = p.Project_ID
WHERE c.Cont_ID = 1;
```

Results

Cont_Name	Project_Name	Job_Name
1 Greg Gumpert	Garvel Housing	Smith 4465 Bambert
2 Greg Gumpert	Garvel Housing	Smith 5565 Bambert

Query executed successfully.

CoT-CIS3365-09 (14.0 RTM) COUGARNET\jnullme (52) CoognetTKS | 00:00:00 | 2 rows

SQLQuery2.sql - CoT-CIS3365-09.CoognetTKS (COUGARNET\jnullme (52)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Object Explorer

CoT-CIS3365-09 (SQL Server 14.0.2027.2 - COUGARNET\jnullme (52))

SQLQuery2.sql - Co.NET\jnullme (53)*

```
-- Query 31 -- Select a specific contractor to find information about what material they have spent the most on in the last year
SELECT
    Cont_FirstName + ' ' + Cont_LastName AS Cont_Name,
    MatType_Name,
    Material_Name,
    SUM(Material_Cost * COL_Quantity) AS Total_Spent
FROM Contractor c
JOIN Contractor_Purchase_Order cpo ON cpo.Cont_ID = c.Cont_ID
JOIN Contractor_Order_Line col ON col.CPO_ID = cpo.CPO_ID
JOIN Material_Source ms ON col.MaterialSource_ID = ms.MaterialSource_ID
JOIN Supplier s ON s.Supplier_ID = ms.Supplier_ID
JOIN Material m ON m.Material_ID = col.Material_ID
JOIN Material_Type mt ON mt.MatType_Code = m.MatType_Code
WHERE c.Cont_ID = 1 AND CPO_Date > DATEADD(year, -5, GETDATE())
GROUP BY Cont_FirstName, Cont_LastName, MatType_Name, Material_Name
ORDER BY SUM(Material_Cost * COL_Quantity) DESC;
```

Results

Cont_Name	MatType_Name	Material_Name	Total_Spent
1 Greg Gumpert	PVC Pipe	3 1/4 inch	37.80
2 Greg Gumpert	Porcelain Tile	12 x 24 inch	9.95

Query executed successfully.

CoT-CIS3365-09 (14.0 RTM) COUGARNET\jnullme (52) CoognetTKS | 00:00:00 | 2 rows

SQLQuery2.sql - CoT-CIS3365-09.CoognetTKS (COUGARNET\jnullme (52)) - Microsoft SQL Server Management Studio

Object Explorer: CoT-CIS3365-09 (SQL Server 14.0.20272 - COUGARNET\jnu)

```

-- Query 30 -- Select a specific contractor and information about the amount they have spent last x years
SELECT
    Cont_FirstName + ' ' + Cont_LastName AS Cont_Name,
    SUM(Material_Cost * COL_Quantity) AS Total_Spent
FROM Contractor c
JOIN Contractor_Purchase_Order cpo ON cpo.Cont_ID = c.Cont_ID
JOIN Contractor_Order_Line col ON col.CPO_ID = cpo.CPO_ID
JOIN Material_Source ms ON col.MaterialSource_ID = ms.MaterialSource_ID
JOIN Supplier s ON s.Supplier_ID = ms.Supplier_ID
JOIN Material m ON m.Material_ID = ms.Material_ID
WHERE c.Cont_ID = 1 AND CPO_Date > DATEADD(year, -5, GETDATE())
GROUP BY Cont_FirstName, Cont_LastName;

```

Results

Cont_Name	Total_Spent
1 Greg Gumpert	47.75

Query executed successfully.

CoT-CIS3365-09 (14.0 RTM) | COUGARNET\jnullme (52) | CoognetTKS | 00:00:00 | 1 rows

SQLQuery2.sql - CoT-CIS3365-09.CoognetTKS (COUGARNET\jnullme (52)) - Microsoft SQL Server Management Studio

Object Explorer: CoT-CIS3365-09 (SQL Server 14.0.20272 - COUGARNET\jnu)

```

-- Query 29 -- Select a specific contractor and information about the materials they have bought in the last x years
SELECT
    Cont_FirstName + ' ' + Cont_LastName AS Cont_Name,
    COL_Quantity,
    Supplier_Name,
    Material_Name
FROM Contractor c
JOIN Contractor_Purchase_Order cpo ON cpo.Cont_ID = c.Cont_ID
JOIN Contractor_Order_Line col ON col.CPO_ID = cpo.CPO_ID
JOIN Material_Source ms ON col.MaterialSource_ID = ms.MaterialSource_ID
JOIN Supplier s ON s.Supplier_ID = ms.Supplier_ID
JOIN Material m ON m.Material_ID = ms.Material_ID
WHERE c.Cont_ID = 1 AND CPO_Date > DATEADD(year, -1, GETDATE());

```

Results

Cont_Name	COL_Quantity	Supplier_Name	Material_Name
1 Greg Gumpert	5	Home Depot	12 x 24 inch
2 Greg Gumpert	12	Lowe's	3 1/4 inch

Query executed successfully.

CoT-CIS3365-09 (14.0 RTM) | COUGARNET\jnullme (52) | CoognetTKS | 00:00:00 | 2 rows

SQLQuery2.sql - CoT-CIS3365-09.CoognetTKS (COUGARNET\jnullme (52)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

CoognetTKS

Object Explorer

CoT-CIS3365-09 (SQL Server 14.0.20272 - COUGARNET\jnu

SQLQuery2.sql - Co-.NET\jnullme (52) SQLQuery1.sql - Co-.NET\jnullme (53)

```
-- Query 28 -- Find a job proposal and show its survey pictures
SELECT
    Cust_FirstName + ' ' + Cust_LastName AS Cust_Name,
    JobProp_Address,
    Survey_Date,
    Survey_Notes,
    Picture_Desc,
    Picture_URL
FROM Job_Proposal jp
JOIN Customer c ON c.Cust_ID = jp.Cust_ID
JOIN Site_Survey ss ON ss.JobProp_ID = jp.JobProp_ID
JOIN Survey_Picture sp ON sp.Survey_ID = ss.Survey_ID
WHERE jp.JobProp_ID = 1
```

Results

Cust_Name	JobProp_Address	Survey_Date	Survey_Notes	Picture_Desc	Picture_URL
-----------	-----------------	-------------	--------------	--------------	-------------

Query executed successfully.

CoT-CIS3365-09 (14.0 RTM) COUGARNET\jnullme (52) CoognetTKS | 00:00:00 | 0 rows

SQLQuery2.sql - CoT-CIS3365-09.CoognetTKS (COUGARNET\jnullme (52)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

CoognetTKS

Object Explorer

CoT-CIS3365-09 (SQL Server 14.0.20272 - COUGARNET\jnu

SQLQuery2.sql - Co-.NET\jnullme (52) SQLQuery1.sql - Co-.NET\jnullme (53)

```
-- Query 27 -- Find all job proposals where architectural review is of a specific type
SELECT
    Cust_FirstName + ' ' + Cust_LastName AS Cust_Name,
    JobProp_Address,
    Survey_Date,
    Survey_Notes,
    Rvw_TypeName,
    Rvw_Requirements,
    RvwStatus_Name
FROM Job_Proposal jp
JOIN Customer c ON c.Cust_ID = jp.Cust_ID
JOIN Site_Survey ss ON ss.JobProp_ID = jp.JobProp_ID
JOIN Architectural_Review ar ON ar.Survey_ID = ss.Survey_ID
JOIN Review_Type rt ON rt.Rvw_TypeID = ar.Rvw_TypeID
JOIN Review_Status rs ON rs.RvwStatus_ID = ar.RvwStatus_ID
WHERE Rvw_TypeName = 'Home Inspection'
```

Results

Cust_Name	JobProp_Address	Survey_Date	Survey_Notes	Rvw_TypeName	Rvw_Requirements	RvwStatus_Name
Anthony Smith	5565 Bambet St. Wild Horse, LA 12345	2019-08-12	Need to check home for water damage.	Home Inspection	Home is searched for mold, rot, termites, and ot...	Pending

Query executed successfully.

CoT-CIS3365-09 (14.0 RTM) COUGARNET\jnullme (52) CoognetTKS | 00:00:00 | 1 rows

SQLQuery2.sql - CoT-CIS3365-09.CoognetTKS (COUGARNET\jnu) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

CoognetTKS

Object Explorer

CoT-CIS3365-09 (SQL Server 14.0.2027.2 - COUGARNET\jnu)

SQLQuery2.sql - Co-.NET\jnu\lme (52) * SQLQuery1.sql - Co-.NET\jnu\lme (53) *

```
-- Query 26 -- Find all job proposals where architectural review is of a specific status
SELECT
    Cust_FirstName + ' ' + Cust_LastName AS Cust_Name,
    JobProp_Address,
    Survey_Date,
    Survey_Notes,
    Rvw_TypeName,
    Rvw_Requirements,
    RvwStatus_Name
FROM Job_Proposal jp
JOIN Customer c ON c.Cust_ID = jp.Cust_ID
JOIN Site_Survey ss ON ss.JobProp_ID = jp.JobProp_ID
JOIN Architectural_Review ar ON ar.Survey_ID = ss.Survey_ID
JOIN Review_Type rt ON rt.Rvw_TypeID = ar.Rvw_TypeID
JOIN Review_Status rs ON rs.RvwStatus_ID = ar.RvwStatus_ID
WHERE RvwStatus_Name = 'Complete'
```

Results

	Cust_Name	JobProp_Address	Survey_Date	Survey_Notes	Rvw_TypeName	Rvw_Requirements	RvwStatus_Name
1	Althor Ambler	3202 Woodlawn Ave, Harbur, TX 77777	2019-11-06	Might need to pay additional attention to the ki...	Preliminary Inspection	Home is inspected to ensure that job proposal is...	Complete
2	Anthony Smith	4465 Bambert St, Wild Horse, LA 12345	2019-10-23	Kitchen tiles need to be replaced.	Preliminary Inspection	Home is inspected to ensure that job proposal is...	Complete

Query executed successfully.

CoT-CIS3365-09 (14.0 RTM) COUGARNET\jnu\lme (52) CoognetTKS | 00:00:00 | 2 rows

SQLQuery2.sql - CoT-CIS3365-09.CoognetTKS (COUGARNET\jnu) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

CoognetTKS

Object Explorer

CoT-CIS3365-09 (SQL Server 14.0.2027.2 - COUGARNET\jnu)

SQLQuery2.sql - Co-.NET\jnu\lme (52) * SQLQuery1.sql - Co-.NET\jnu\lme (53) *

```
-- Query 25 -- Find all job proposals from a specific customer and show its survey pictures
SELECT
    Cust_FirstName + ' ' + Cust_LastName AS Cust_Name,
    JobProp_Address,
    Survey_Date,
    Survey_Notes,
    Picture_Desc,
    Picture_URL
FROM Job_Proposal jp
JOIN Customer c ON c.Cust_ID = jp.Cust_ID
JOIN Site_Survey ss ON ss.JobProp_ID = jp.JobProp_ID
JOIN Survey_Picture sp ON sp.Survey_ID = ss.Survey_ID
WHERE Cust_FirstName = 'Anthony' AND Cust_LastName = 'Smith' AND Cust_Address = '4465 Bambert St, Wild Horse, LA 12345';
```

Results

	Cust_Name	JobProp_Address	Survey_Date	Survey_Notes	Picture_Desc	Picture_URL
--	-----------	-----------------	-------------	--------------	--------------	-------------

Query executed successfully.

CoT-CIS3365-09 (14.0 RTM) COUGARNET\jnu\lme (52) CoognetTKS | 00:00:00 | 0 rows

SQLQuery2.sql - CoT-CIS3365-09.CoognetTKS (COUGARNET\jnullme (52)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

CoognetTKS

Object Explorer

CoT-CIS3365-09 (SQL Server 14.0.20272 - COUGARNET\jnu

SQLQuery2.sql - Co-.NET\jnullme (52) SQLQuery1.sql - Co-.NET\jnullme (53)

```
-- Query 24 -- Find all job proposals from a specific customer and show its architectural review requirements and status
SELECT
    Cust_FirstName + ' ' + Cust_LastName AS Cust_Name,
    JobProp_Address,
    Survey_Date,
    Survey_Notes,
    Rvw_TypeName,
    Rvw_Requirements,
    RvwStatus_Name
FROM Job_Proposal jp
JOIN Customer c ON c.Cust_ID = jp.Cust_ID
JOIN Site_Survey ss ON ss.JobProp_ID = jp.JobProp_ID
JOIN Architectural_Review ar ON ar.Survey_ID = ss.Survey_ID
JOIN Review_Type rt ON rt.Rvw_TypeID = ar.Rvw_TypeID
JOIN Review_Status rs ON rs.RvwStatus_ID = ar.RvwStatus_ID
WHERE Cust_FirstName = 'Anthony' AND Cust_LastName = 'Smith';
```

Results

	Cust_Name	JobProp_Address	Survey_Date	Survey_Notes	Rvw_TypeName	Rvw_Requirements	RvwStatus_Name
1	[Anthony Smith]	4465 Bambet St. Wild Horse, LA 12345	2019-10-23	Kitchen tiles need to be replaced.	Preliminary Inspection	Home is inspected to ensure that job proposal is...	Complete
2	Anthony Smith	5565 Bambet St. Wild Horse, LA 12345	2019-08-12	Need to check home for water damage.	Home Inspection	Home is searched for mold, rot, termites, and of...	Pending

Query executed successfully.

CoT-CIS3365-09 (14.0 RTM) COUGARNET\jnullme (52) CoognetTKS | 00:00:00 | 2 rows

SQLQuery2.sql - CoT-CIS3365-09.CoognetTKS (COUGARNET\jnullme (52)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

CoognetTKS

Object Explorer

CoT-CIS3365-09 (SQL Server 14.0.20272 - COUGARNET\jnu

SQLQuery2.sql - Co-.NET\jnullme (52) SQLQuery1.sql - Co-.NET\jnullme (53)

```
-- Query 23 -- Find a job proposal and show its architectural review requirements and status
SELECT
    Cust_FirstName + ' ' + Cust_LastName AS Cust_Name,
    JobProp_Address,
    Survey_Date,
    Survey_Notes,
    Rvw_TypeName,
    Rvw_Requirements,
    RvwStatus_Name
FROM Job_Proposal jp
JOIN Customer c ON c.Cust_ID = jp.Cust_ID
JOIN Site_Survey ss ON ss.JobProp_ID = jp.JobProp_ID
JOIN Architectural_Review ar ON ar.Survey_ID = ss.Survey_ID
JOIN Review_Type rt ON rt.Rvw_TypeID = ar.Rvw_TypeID
JOIN Review_Status rs ON rs.RvwStatus_ID = ar.RvwStatus_ID
WHERE jp.JobProp_ID = 3;
```

Results

	Cust_Name	JobProp_Address	Survey_Date	Survey_Notes	Rvw_TypeName	Rvw_Requirements	RvwStatus_Name
1	[Anthony Smith]	5565 Bambet St. Wild Horse, LA 12345	2019-08-12	Need to check home for water damage.	Home Inspection	Home is searched for mold, rot, termites, and of...	Pending

Query executed successfully.

CoT-CIS3365-09 (14.0 RTM) COUGARNET\jnullme (52) CoognetTKS | 00:00:00 | 1 rows

SQLQuery2.sql - CoT-CIS3365-09.CoognetTKS (COUGARNET\jnullme (52)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

CoognetTKS

Object Explorer

- CoT-CIS3365-09 (SQL Server 14.0.2027.2 - COUGARNET\jnu)
 - Databases
 - System Databases
 - Database Snapshots
 - CoognetTKS
 - Security
 - Server Objects
 - Replication
 - PolyBase
 - Always On High Availability
 - Management
 - Integration Services Catalogs
 - SQL Server Agent (Agent XPs disabled)
 - XEvent Profiler

SQLQuery2.sql - Co-.NET\jnullme (52) * SQLQuery1.sql - Co-.NET\jnullme (53) *

```
-- Query 22 -- Select all jobs and sort by number of problems
SELECT
  Job_Name,
  COUNT(p.Prblm_ID) AS Prblm_Count
FROM Job j
JOIN Job_Material jm ON jm.Job_ID = j.Job_ID
JOIN Task t ON t.Job_ID = j.Job_ID
JOIN Task_Problem tp ON tp.Task_ID = t.Task_ID
JOIN Problem p ON p.Prblm_ID = tp.Prblm_ID
GROUP BY j.Job_ID, j.Job_Name
ORDER BY Prblm_Count;
```

Results Messages

Job_Name	Prblm_Count
Smeth 4465 Bambert	1

Query executed successfully.

CoT-CIS3365-09 (14.0 RTM) | COUGARNET\jnullme (52) | CoognetTKS | 00:00:00 | 1 rows

SQLQuery2.sql - CoT-CIS3365-09.CoognetTKS (COUGARNET\jnullme (52)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

CoognetTKS

Object Explorer

- CoT-CIS3365-09 (SQL Server 14.0.2027.2 - COUGARNET\jnu)
 - Databases
 - System Databases
 - Database Snapshots
 - CoognetTKS
 - Security
 - Server Objects
 - Replication
 - PolyBase
 - Always On High Availability
 - Management
 - Integration Services Catalogs
 - SQL Server Agent (Agent XPs disabled)
 - XEvent Profiler

SQLQuery2.sql - Co-.NET\jnullme (52) * SQLQuery1.sql - Co-.NET\jnullme (53) *

```
-- Query 21 -- Select all problems for one job
SELECT
  Job_Name,
  Task_Name,
  Prblm_Desc
FROM Job j
JOIN Job_Material jm ON jm.Job_ID = j.Job_ID
JOIN Task t ON t.Job_ID = j.Job_ID
JOIN Task_Problem tp ON tp.Task_ID = t.Task_ID
JOIN Problem p ON p.Prblm_ID = tp.Prblm_ID
WHERE Job_Name = 'Smeth 4465 Bambert';
```

Results Messages

Job_Name	Task_Name	Prblm_Desc
Smeth 4465 Bambert	Place new tea	Required materials are absent

Query executed successfully.

CoT-CIS3365-09 (14.0 RTM) | COUGARNET\jnullme (52) | CoognetTKS | 00:00:00 | 1 rows

SQLQuery2.sql - CoT-CIS3365-09.CoognetTKS (COUGARNET\jnu) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

CoognetTKS

Object Explorer

CoT-CIS3365-09 (SQL Server 14.0.20272 - COUGARNET\jnu)

Databases

- System Databases
- Database Snapshots
- CoognetTKS
- Security
- Server Objects
- Replication
- PolyBase
- Always On High Availability
- Management
- Integration Services Catalogs
- SQL Server Agent (Agent XPs disabled)
- XEvent Profiler

```

-- Query 20 -- Select all materials, their costs, and suppliers, for one job
SELECT
    Job_Name,
    MatType_Name,
    Material_Name,
    Material_Cost,
    Supplier_Name
FROM Job j
    JOIN Job_Material jm ON jm.Job_ID = j.Job_ID
    JOIN Material m ON m.Material_ID = jm.Material_ID
    JOIN Material_Type mt ON mt.MatType_Code = m.MatType_Code
    JOIN Material_Source ms ON ms.Material_ID = m.Material_ID
    JOIN Supplier s ON s.Supplier_ID = ms.Supplier_ID
WHERE Job_Name = 'Smith 4465 Bambart';

```

100 %

Job_Name	MatType_Name	Material_Name	Material_Cost	Supplier_Name
1 Smith 4465 Bambart	Porcelain Tile	12x 24 inch	1.99	Home Depot
2 Smith 4465 Bambart	Porcelain Tile	12x 24 inch	2.19	Lowes
3 Smith 5565 Bambart	PVC Pipe	3/4 inch	3.15	Lowes
4 Smith 5565 Bambart	PVC Pipe	3/4 inch	3.00	Home Depot

Query executed successfully.

CoT-CIS3365-09 (14.0 RTM) | COUGARNET\jnu (S2) | CoognetTKS | 00:00:00 | 4 rows

SQLQuery2.sql - CoT-CIS3365-09.CoognetTKS (COUGARNET\jnu) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

CoognetTKS

Object Explorer

CoT-CIS3365-09 (SQL Server 14.0.20272 - COUGARNET\jnu)

Databases

- System Databases
- Database Snapshots
- CoognetTKS
- Security
- Server Objects
- Replication
- PolyBase
- Always On High Availability
- Management
- Integration Services Catalogs
- SQL Server Agent (Agent XPs disabled)
- XEvent Profiler

```

-- Query 19 -- Select all jobs and show their progress
SELECT
    JobProp_Address,
    Job_Name,
    CAST(CAST(
        SELECT
            COUNT(1)
        FROM Task t
            JOIN Status s ON s.Status_ID = t.Status_ID
        WHERE
            t.Job_ID = j.Job_ID
            AND Status_Name = 'Complete'
        ) / 1.0 / (
        SELECT
            COUNT(1)
        FROM Task t
        WHERE t.Job_ID = j.Job_ID
        ) * 100 AS INT) AS VARCHAR(3)) + '%' AS Job_Progress
FROM Job j
    JOIN Job_Proposal jp ON j.JobProp_ID = jp.JobProp_ID
    JOIN Customer c ON jp.Cust_ID = c.Cust_ID
    JOIN Task t1 ON t1.Job_ID = j.Job_ID
GROUP BY j.Job_ID, JobProp_Address, Job_Name HAVING COUNT(t1.Task_ID) > 0;

```

100 %

JobProp_Address	Job_Name	Job_Progress
1 4465 Bambart St, Wild Horse, LA 712345	Smith 4465 Bambart	40%
2 3202 Woodane Ave, Hartbur, TX 77777	Amble 3202 Woodane	0%

Query executed successfully.

CoT-CIS3365-09 (14.0 RTM) | COUGARNET\jnu (S2) | CoognetTKS | 00:00:00 | 2 rows

SQLQuery2.sql - CoT-CIS3365-09.CoognetTKS (COUGARNET\jnullme (52)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

CoognetTKS

Object Explorer

Connect - CoT-CIS3365-09 (SQL Server 14.0.2027.2 - COUGARNET\jnullme (52))

SQLQuery2.sql - Co.NET\jnullme (52) SQLQuery1.sql - Co.NET\jnullme (53)

```
-- Query 18 -- Select all job proposals that have become jobs and view the tasks involved
SELECT
    Cust_FirstName + ' ' + Cust_LastName AS Cust_Name,
    JobProp_Address,
    Job_Name,
    Task_Name,
    Status_Name
FROM Job j
    JOIN Job_Proposal jp ON j.JobProp_ID = jp.JobProp_ID
    JOIN Customer c ON jp.Cust_ID = c.Cust_ID
    JOIN Task t ON t.Job_ID = j.Job_ID
    JOIN Status s ON s.Status_ID = t.Status_ID
ORDER BY Job_Name, Task_ID;
```

Results

	Cust_Name	JobProp_Address	Job_Name	Task_Name	Status_Name
1	Althor Ambles	3202 Woodlane Ave, Harbur, TX 77777	Ambles 3202 Woodlane	Search outside roof for imperfections	Pending
2	Althor Ambles	3202 Woodlane Ave, Harbur, TX 77777	Ambles 3202 Woodlane	Search inside roof for imperfections	Pending
3	Althor Ambles	3202 Woodlane Ave, Harbur, TX 77777	Ambles 3202 Woodlane	Search upstairs for signs of water damage	Pending
4	Althor Ambles	3202 Woodlane Ave, Harbur, TX 77777	Ambles 3202 Woodlane	Search downstairs for signs of water damage	Pending
5	Anthony Smith	4465 Bambert St, Wild Horse, LA 12345	Smith 4465 Bambert	Remove old tiles from kitchen	Complete
6	Anthony Smith	4465 Bambert St, Wild Horse, LA 12345	Smith 4465 Bambert	Inspect surface and ensure it is level	Complete
7	Anthony Smith	4465 Bambert St, Wild Horse, LA 12345	Smith 4465 Bambert	Place new tiles	In Progress
8	Anthony Smith	4465 Bambert St, Wild Horse, LA 12345	Smith 4465 Bambert	Grout tiles	Pending
9	Anthony Smith	4465 Bambert St, Wild Horse, LA 12345	Smith 4465 Bambert	Clean tiles and inspect surface	Pending

Query executed successfully.

CoT-CIS3365-09 (14.0 RTM) COUGARNET\jnullme (52) CoognetTKS | 00:00:00 | 9 rows

SQLQuery2.sql - CoT-CIS3365-09.CoognetTKS (COUGARNET\jnullme (52)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

CoognetTKS

Object Explorer

Connect - CoT-CIS3365-09 (SQL Server 14.0.2027.2 - COUGARNET\jnullme (52))

SQLQuery2.sql - Co.NET\jnullme (52) SQLQuery1.sql - Co.NET\jnullme (53)

```
-- Query 17 --- Find what jobs an employee is assigned to
SELECT
    Emp_FirstName + ' ' + Emp_LastName AS Emp_Name,
    Project_Name,
    Job_Name
FROM Employee e
    JOIN Employee_Project ep ON ep.Emp_ID = e.Emp_ID
    JOIN Project p ON p.Project_ID = ep.Project_ID
    JOIN Job j ON j.Project_ID = p.Project_ID;
```

Results

	Emp_Name	Project_Name	Job_Name
1	Robert Powell	Ambles Woodane Home Kitchen Renovation	Ambles 3202 Woodane
2	Albert Bandura	Ambles Woodane Home Kitchen Renovation	Ambles 3202 Woodane
3	Albert Bandura	Garwel Housing	Smith 4465 Bambert
4	Albert Bandura	Garwel Housing	Smith 5565 Bambert
5	Jones Anable	Garwel Housing	Smith 4465 Bambert
6	Jones Anable	Garwel Housing	Smith 5565 Bambert

Query executed successfully.

CoT-CIS3365-09 (14.0 RTM) COUGARNET\jnullme (52) CoognetTKS | 00:00:00 | 6 rows

SQLQuery2.sql - CoT-CIS3365-09.CogonetTKS (COUGARNET\jnullme (52)) - Microsoft SQL Server Management Studio

```

-- Query 16 -- Select the departments that have spent more than x on materials in the last y years
SELECT
    Dept_Name,
    EOL_Quantity,
    Supplier_Name,
    MatType_Name,
    Material_Name,
    SUM(Material_Cost * EOL_Quantity) AS Total_Spent
FROM Department d
JOIN Employee e ON e.Dept_ID = d.Dept_ID
JOIN Employee_Purchase_Order epo ON epo.Emp_ID = e.Emp_ID
JOIN Employee_Order_Line eol ON eol.EPO_ID = epo.EPO_ID
JOIN Material_Source ms ON eol.MaterialSource_ID = ms.MaterialSource_ID
JOIN Supplier s ON s.Supplier_ID = ms.Supplier_ID
JOIN Material m ON ms.Material_ID = m.Material_ID
JOIN Material_Type mt ON mt.MatType_Code = m.MatType_Code
WHERE EPO_Date > DATEADD(year, -3, GETDATE())
GROUP BY d.Dept_ID, Dept_Name, EOL_Quantity, Supplier_Name, MatType_Name, Material_Name HAVING SUM(Material_Cost * EOL_Quantity) > 2.0

```

Dept_Name	EOL_Quantity	Supplier_Name	MatType_Name	Material_Name	Total_Spent
Information Technology	5	Home Depot	Porcelain Tile	12 x 24 inch	9.95
Information Technology	12	Lowe's	PVC Pipe	3 1/4 inch	37.80
Architecture	4	Lowe's	Porcelain Tile	12 x 24 inch	8.76

Query executed successfully.

SQLQuery2.sql - CoT-CIS3365-09.CogonetTKS (COUGARNET\jnullme (52)) - Microsoft SQL Server Management Studio

```

-- Query 15 -- Select the departments that have bought material x in the last y years
SELECT
    Dept_Name,
    EOL_Quantity,
    Supplier_Name,
    MatType_Name,
    Material_Name
FROM Department d
JOIN Employee e ON e.Dept_ID = d.Dept_ID
JOIN Employee_Purchase_Order epo ON epo.Emp_ID = e.Emp_ID
JOIN Employee_Order_Line eol ON eol.EPO_ID = epo.EPO_ID
JOIN Material_Source ms ON eol.MaterialSource_ID = ms.MaterialSource_ID
JOIN Supplier s ON s.Supplier_ID = ms.Supplier_ID
JOIN Material m ON ms.Material_ID = m.Material_ID
JOIN Material_Type mt ON mt.MatType_Code = m.MatType_Code
WHERE MatType_Name = 'Porcelain Tile' AND Material_Name = '12 x 24 inch' AND EPO_Date > DATEADD(year, -3, GETDATE())

```

Dept_Name	EOL_Quantity	Supplier_Name	MatType_Name	Material_Name
Information Technology	5	Home Depot	Porcelain Tile	12 x 24 inch
Architecture	4	Lowe's	Porcelain Tile	12 x 24 inch

Query executed successfully.

APPENDIX E – COPIES OF INSERT, UPDATE, & DELETE SCRIPTS

INSERT SCRIPTS:

Insert Query 1

This query inserts a new row into the Customer Table:

```
INSERT INTO Customer
VALUES (custfirstname, custlastname, custaddress);
```

Example:

Pure SQL query:

```
INSERT INTO Customer
VALUES ('Joe', 'Schmoe', '1234 Test St, Austin, TX 12345');
```

Before Query:

Info	Content	Row Count	Columns	Primary Key	Exported Keys	Imported Keys	Indexes	Privileges	Colu
			Cust_ID	Cust_FirstName	Cust_LastName	Cust_Address			
1	Abithor					3202 Woodlane Ave, Hartbur, TX 77777			
2	Anthony					4465 Bambert St, Wild Horse, LA 12345			
3	Pam					56731 Joseph St, Olga TX 54321			

After Query:

Imported Keys	Indexes	Privileges	Column Privileges	Row IDs	Versions	
Info	Content	Row Count	Columns	Primary Key	Exported Keys	
			Cust_ID	Cust_FirstName	Cust_LastName	Cust_Address
1	Abithor					3202 Woodlane Ave, Hartbur, TX 77777
2	Anthony					4465 Bambert St, Wild Horse, LA 12345
3	Pam					56731 Joseph St, Olga TX 54321
5	Joe					1234 Test St, Austin, TX 12345

Insert Query 2

This query inserts a new row into the Review_Type table:

```
INSERT INTO Review_Type  
VALUES (rvwtypename, revwreq);
```

Pure SQL Query:

```
INSERT INTO Review_Type  
VALUES ('Fire Code Inspection', 'The structure is searched for any and all fire hazards');
```

Before Query:

Info	Content	Row Count	Columns	Primary Key	Exported Keys	Imported Keys	Indexes
Rvw_Typ...	Rvw_TypeName						
1	Structural Review						
2	Home Inspection						
3	Building Inspection						
4	Preliminary Inspection						

After Query:

Info	Content	Row Count	Columns	Primary Key	Exported Keys	Imported Keys	Ir
Rvw_Typ...	Rvw_TypeName						
1	Structural Review						
2	Home Inspection						
3	Building Inspection						
4	Preliminary Inspection						
7	Fire Code Inspection						

Insert Query 3

This query inserts a new row into the ContractorTable:

```
INSERT INTO Contractor  
VALUES (contfistname, contlastname, contsalary);
```

Example:

Pure SQL Query:

```
INSERT INTO Contractor  
VALUES ('Joe', 'Schmoe', 40000);
```

Before Query:

Imported Keys	Indexes	Privileges	Column Privileges	Row IDs
Info	Content	Row Count	Columns	Primary Key
Cont_ID	Cont_FirstName	Cont_LastName	Cont_Salary	
1	Greg	Gumpert	50000.0000	
2	Tony	Lambo	60000.0000	
3	Torstein	Muller	70000.0000	

After Query:

Imported Keys	Indexes	Privileges	Column Privileges	Row IDs	Version
Info	Content	Row Count	Columns	Primary Key	
Cont_ID	Cont_FirstName	Cont_LastName	Cont_Salary		
1	Smith	Gumpert	50000.0000		
2	Tony	Lambo	60000.0000		
3	Torstein	Muller	70000.0000		
5	Joe	Schmoe	40000.0000		

Insert Query 4

This query inserts a row into the Supplier table:

```
INSERT INTO Supplier  
VALUES (suppliername);
```

Example:

Pure SQL Query:

```
INSERT INTO Supplier  
VALUES ('Wholesale Construction');
```

Before Query:

Supplier_ID	Supplier_Name
1	Lowe's
2	Home Depot
3	Harbor Freight

After Query:

Info	Content	Row Count	Columns	Primary Key	Exported Ke
	Supplier_ID		Supplier_Name		
	1		Lowe's		
	2		Home Depot		
	3		Harbor Freight		
	5		Wholesale Construction		

Insert Query 5

This query inserts a row into the Task Table

```
INSERT INTO TASK  
VALUES (jobid, statusid, taskname);
```

Example:

Pure SQL Query:

```
INSERT INTO TASK  
VALUES (2,3, 'Replace lighting on front porch');
```

Before Query:

Info	Content	Row Count	Columns	Primary Key	Exported Keys	Imported Keys	It
Task_ID	Job_ID	Status_ID	Task_Name				
1	1	5	Remove old tiles from kitchen				
2	1	5	Inspect surface and ensure it is level				
3	1	2	Place new tiles				
4	1	1	Grout tiles				
5	1	1	Clean tiles and inspect surface				
6	3	1	Search outside roof for imperfections				
7	3	1	Search inside roof for imperfections				
8	3	1	Search upstairs for signs of water damage				
9	3	1	Search downstairs for signs of water damage				

After Query:

Info	Content	Row Count	Columns	Primary Key	Exported Keys	Imported Keys
Task_ID	Job_ID	Status_ID	Task_Name			
1	1	5	Remove old tiles from kitchen			
2	1	5	Inspect surface and ensure it is level			
3	1	2	Place new tiles			
4	1	1	Grout tiles			
5	1	1	Clean tiles and inspect surface			
6	3	1	Search outside roof for imperfections			
7	3	1	Search inside roof for imperfections			
8	3	1	Search upstairs for signs of water damage			
9	3	1	Search downstairs for signs of water damage			
11	2	3	Replace lighting on front porch			

Insert Query 6

This query inserts a row into the Task_Problem Table:

```
INSERT INTO TASK_Problem  
VALUES (taskid, prblmid);
```

Example:

Pure SQL Query:

```
INSERT INTO TASK_Problem  
VALUES (1,3);
```

Before Query:

Info	Content	Row C
Task_ID	Prblm_ID	
2	1	
2	2	

After Query:

Info	Content	Row Co
Task_ID	Prblm_ID	
2	1	
2	2	
1	3	

Insert Query 7

This query inserts a new row into the Vehicle Table:

```
INSERT INTO Vehicle  
VALUES (fleetid, vehiclemake, vehiclemodel, vehiclecolor);
```

Example:

Pure SQL Query:

```
INSERT INTO Vehicle  
VALUES (1,'Ford', 'F-150', 'Blue');
```

Before Query:

Info	Content	Row Count	Columns	Primary Key	Exported Keys
↑ Vehicl...	Fleet...	Vehicle_Make	Vehicle_Model	Vehicle_Color	
1	1	Ford	Transit	Red	
2	1	Ford	Transit	White	
3	1	Ford	Transit	Blue	

After Query:

Info	Content	Row Count	Columns	Primary Key	Exported Keys	Imported
Vehicle_ID	Fleet_ID	Vehicle_Make	Vehicle_Model	Vehicle_Color		
1	1	Ford	Transit	Red		
2	1	Ford	Transit	White		
3	1	Ford	Transit	Blue		
5	1	Ford	F-150	Blue		

Insert Query 8

This query inserts a row into Work_Order:

```
INSERT INTO Work_Order  
VALUES (contrid, projid);
```

Example:

Pure SQL Query:

```
INSERT INTO Work_Order  
VALUES (3,2);
```

Before Query:

Cont_ID	Project_ID
1	2
2	1
2	2
3	1

After Query:

Cont_ID	Project_ID
1	2
2	1
2	2
3	1
3	2

Insert Query 9

This Inserts a row into the Department Table:

```
INSERT INTO Department  
SET Dept_Name = deptname  
WHERE Dept_ID = selection;
```

Example:

Pure SQL Query:

```
INSERT INTO Department  
VALUES ('Accounting');
```

Before Query:

Info	Content	Row Count	Columns	Primary Key	Exported Ke
	Dept_ID	Dept_Name			
1	Management				
2	Information Technology				
3	Architecture				
4	Contract Work				
5	Human Resources				

After Query:

Imported Keys	Indexes	Privileges	Column Privileges	Row ID:	
Info	Content	Row Count	Columns	Prima	
	Dept_ID	Dept_Name			
1	Management				
2	Information Technology				
3	Architecture				
4	Contract Work				
5	Human Resources				
7	Accounting				

Insert Query 10

This query inserts a row into the Problem table:

```
INSERT INTO Problem  
SET Prblm_Desc = prblmdesc  
WHERE Prblm_ID = selection;
```

Example:

Pure SQL Query:

```
INSERT INTO Problem  
VALUES ('Needs evalaluation');
```

Before Query:

Info	Content	Row Count	Columns	Primary Key	Exported I
↑ Prblm...	Prblm_Desc				
1	Spatial constraints hinder movement of materials				
2	Required materials are absent				
3	Expert assistance required				

After Query:

Imported Keys	Indexes	Privileges	Column Privileges	Row ID
Info	Content	Row Count	Columns	Prim
↑ Prblm...	Prblm_Desc			
1	Spatial constraints hunder movement of materials			
2	Required materials are absent			
3	Expert assistance required.			
6	Needs evalaluation			

Insert Query 11

This query Inserts a row into the Review Status table:

```
INSERT INTO Review_Status  
SET RvwStatus_Name = prblmdesc  
WHERE RvwStatus_ID = selection;
```

Example:

Pure SQL Query:

```
INSERT INTO Review_Status  
VALUES ('Incomplete');
```

Before Query:

Info	Content	Row Count	Columns	Primary
↑ RvwSta...	RvwStatus_Name			
1	Pending			
2	In Progress			
3	Attention Required			
4	Complete			

After Query:

↑ RvwSta...	RvwStatus_Name			
1	Partial Completion			
2	In Progress			
3	Attention Required			
4	Complete			
6	Incomplete			

Insert Query 12

This query inserts a row into the Employee History Table:

```
INSERT INTO Employee_History  
VALUES (emphistdate, empid, emphistsalary);
```

Example:

Pure SQL Query:

```
INSERT INTO Employee_History  
VALUES (GETDATE(), 2, 50000);
```

Before Query:

Info	Content	Row Count	Columns	Prim
Date_Assi...	Emp_Id	Emp_Salary		
2012-05-13	1	65000.0000		
2016-05-10	1	75000.0000		
2018-05-10	1	80000.0000		
2019-05-10	1	190000.0000		
2016-05-10	2	75000.0000		
2016-05-10	3	80000.0000		

After Query:

Info	Content	Row Count	Columns	Prim
↑ Date_...	Emp_Id	Emp_Salary		
2012-05-13	1	65000.0000		
2016-05-10	1	75000.0000		
2016-05-10	2	75000.0000		
2016-05-10	3	80000.0000		
2018-05-10	1	80000.0000		
2019-05-10	1	190000.0000		
2019-11-11	2	50000.0000		

Insert Query 13

This query assigns an employee a new project by adding a row into the Employee_Project Table

SQL

Example:

Pure SQL Query:

```
INSERT INTO Employee_Project  
VALUES (3, 1);
```

Before Query:

Emp_ID	Project_ID
1	1
1	2
2	1
2	2
3	2
3	3

After Query:

Emp_ID	Project_ID
1	1
1	2
2	1
2	2
3	1
3	2
3	3

Insert Query 14

This query Inserts a row into the Job_Material Table :

```
INSERT INTO Job_Material  
VALUES (jobid, materialid);
```

Example:

Pure SQL Query:

```
INSERT INTO Job_Material  
VALUES (3, 2);
```

Before Query:

Job_ID	Material_ID
1	2
2	2
3	3

After Query:

Job_ID	Material_ID
1	2
2	2
3	2
3	3

Insert Query 15

This query inserts a row into the Maintenance table:

```
INSERT INTO Maintenance  
VALUES (vehicleid, maintenancedesc, maintenencecost, maintinencedate);
```

Example:

Pure SQL Query:

```
INSERT INTO Maintenance  
VALUES (2,'Tire rotation', 40.34, GETDATE());
```

Before Query:

Info	Content	Row Count	Columns	Primary Key	Exported Keys	Imported Keys	Indexes	Pr
	Maintenance_ID	Vehicle_ID	Maintenance_Desc	Maintenance_Cost	Maintenance_Date			
1		1	Oil Change	201.3400	2019-11-06			

After Query:

Info	Content	Row Count	Columns	Primary Key	Exported Keys	Imported Keys	Indexes	Pr
	Maintenance...	Vehicle_ID	Maintenance_Desc	Maintenance_Cost	Maintenance_D...			
1		1	Oil Change	201.3400	2019-11-06			
3		2	Tire rotation	40.3400	2019-11-11			

Insert Query 16

This query inserts a row into the Material Table :

```
INSERT INTO Material  
VALUES (mattypecode, matname, matcostcode);
```

Example:

Pure SQL Query:

```
INSERT INTO Material  
VALUES (6, '10x12 feet','PT32');
```

Before Query:

Info	Content	Row Count	Columns	Primary Key	Exported Keys	Imported Keys	Indexes	F
	Material_ID	MatType_Code	Material_CostCode	Material_Name				
1	1	IN32	Fire Proof					
2	2	PI53	3 1/4 inch					
3	3	WP24	2 x 4 inch					
4	4	PT12	12 x 24 inch					

After Query:

Info	Content	Row Count	Columns	Primary Key	Exported Keys	Imported Keys	Indexes
	Material_ID	MatType_C...	Material_Name				Material_CostCode
1	1	Fire Proof				IN32	
2	2	3 1/4 inch				PI53	
3	3	2 x 4 inch				WP24	
4	4	12 x 24 inch				PT12	
7	6	10x12 feet				PT32	

Insert Query 17

This query inserts a row into the Material_Type Table

```
INSERT INTO Material_Type  
VALUES (materialtype);
```

Example:

Pure SQL Query:

```
INSERT INTO Material_Type  
VALUES ('Plywood Sheet');
```

Before Query:

Info	Content	Row Count	Columns	Primary Key	Exported K
MatType_C...	MatType_Name				
1	Insulation				
2	PVC Pipe				
3	Wooden Plank				
4	Porcelain Tile				

After Query:

Info	Content	Row Count	Columns	Primary Key	Exported K
MatType_C...	MatType_Name				
1	Insulation				
2	PVC Pipe				
3	Wooden Plank				
4	Porcelain Tile				
6	Plywood Sheet				

Insert Query 18

This query inserts a row into the Project Table :

```
INSERT INTO Project  
VALUES (projectname);
```

Example:

Pure SQL Query:

```
INSERT INTO Project  
VALUES ('Holmes Developers Job');
```

Before Query:

Info	Content	Row Count	Columns	Primary Key	Exported Ke
	Project_ID	Project_Name			
1	Ambles Woodlane Home Kitchen Renovation				
2	Garwel Housing				
3	Olga Insulation Deal				

After Query:

Info	Content	Row Count	Columns	Primary Key	Exported Ke
	Project_ID	Project_Name			
1	Ambles Woodlane Home Kitchen Renovation				
2	Garwel Housing				
3	Olga Insulation Deal				
5	Holmes Developers Job				

UPDATE SCRIPTS:

Update Query 1

This query updates an entry in the 'Customer' table by selection of Customer_ID:

```
UPDATE Customer  
SET Cust_FirstName = custfirstname , Cust_LastName = custlastname, Cust_Address =  
custaddress  
WHERE Cust_ID = selection;
```

Example:

Pure SQL query:

```
UPDATE Customer
```

SET Cust_FirstName = 'Tony' , Cust_LastName = 'Rambone', Cust_Address = '4800 Calhoun Rd, Houston, TX 77004'
 WHERE Cust_ID = 2;

Before Query:

Info	Content	Row Count	Columns	Primary Key	Exported Keys	Imported Keys	Indexes	Privileges	Colu
	Cust_ID	Cust_FirstName	Cust_LastName	Cust_Address					
1	Abithor	Ambles	3202 Woodlane Ave, Hartbur, TX 77777						
2	Anthony	Smith	4465 Bambert St, Wild Horse, LA 12345						
3	Pam	Miller	56731 Joseph St, Olga TX 54321						

After Query:

Imported Keys	Indexes	Privileges	Column Privileges	Row IDs	Versions	
Info	Content	Row Count	Columns	Primary Key	Exported Keys	
	Cust_ID	Cust_FirstName	Cust_LastName	Cust_Address		
1	Abithor	Ambles	3202 Woodlane Ave, Hartbur, TX 77777			
2	Tony	Rambone	4800 Calhoun Rd, Houston, TX 77004			
3	Pam	Miller	56731 Joseph St, Olga TX 54321			

Update Query 2

This query updates only an address for a customer by selected Customer ID:

```
UPDATE Customer
SET Cust_Address = custaddress
WHERE Cust_ID = selection;
```

jdbc:sqlserver://CoT-CIS3365-09;databaseName=CoognetTKSEExample:

Pure SQL Query:

```
UPDATE Customer
SET Cust_Address = '4800 Calhoun Rd, Houston, TX 77004'
WHERE Cust_ID = 2;
```

Before Query:

Imported Keys		Indexes	Privileges	Column Privileges	Row IDs	Versions	
Info	Content	Row Count	Columns	Primary Key	Exported Keys		
Cust_ID	Cust_FirstName	Cust_LastName	Cust_Address				
1	Abithor	Ambles	3202 Woodlane Ave, Hartbur, TX 77777				
2	Anthony	Smith	4465 Bambert St, Wild Horse, LA 12345				
3	Pam	Miller	56731 Joseph St, Olga TX 54321				

After Query:

Imported Keys		Indexes	Privileges	Column Privileges	Row IDs	Versions	
Info	Content	Row Count	Columns	Primary Key	Exported Keys		
↑ Cust_ID	Cust_FirstName	Cust_LastName	Cust_Address				
1	Abithor	Ambles	3202 Woodlane Ave, Hartbur, TX 77777				
2	Anthony	Smith	4800 Calhoun Rd, Houston, TX 77004				
3	Pam	Miller	56731 Joseph St, Olga TX 54321				

Update Query 3

This Query updates a Contractor row entry that is selected by the Contractor ID:

```
UPDATE Contractor
SET Cont_FirstName = contfirstname, Cont_LastName = contlastname, Cont_Salary =
contsalary
WHERE Cont_ID = selection;
```

Example:

Pure SQL Query:

```
UPDATE Contractor
SET Cont_FirstName = 'Shaun', Cont_LastName = 'Humphrey', Cont_Salary = 100000
WHERE Cont_ID = 1;
```

Before Query:

Imported Keys	Indexes	Privileges	Column Privileges	Row IDs
Info	Content	Row Count	Columns	Primary K
Cont_ID	Cont_FirstName	Cont_LastName	Cont_Salary	
1	Greg	Gumpert	50000.0000	
2	Tony	Lambo	60000.0000	
3	Torstein	Muller	70000.0000	

After Query:

Imported Keys	Indexes	Privileges	Column Privileges	Row IDs	Ve
Info	Content	Row Count	Columns	Primary Key	
↑ Cont_ID	Cont_FirstName	Cont_LastName	Cont_Salary		
1	Shaun	Humphrey	100000.0000		
2	Tony	Lambo	60000.0000		
3	Torstein	Muller	70000.0000		

UPDATE QUERY 4

This query updates a Contractors Salary by the Contractor ID:

```
UPDATE Contractor  
SET Cont_Salary = contsalary  
WHERE Cont_ID = selection;
```

Example:

Pure SQL Query:

```
UPDATE Contractor  
SET Cont_Salary = 151000  
WHERE Cont_ID = 1;
```

Before Query:

Imported Keys	Indexes	Privileges	Column Privileges	Row IDs
Info	Content	Row Count	Columns	Primary K
Cont_ID	Cont_FirstName	Cont_LastN...	Cont_Salary	
1	Greg	Gumpert	50000.0000	
2	Tony	Lambo	60000.0000	
3	Torstein	Muller	70000.0000	

After Query:

Imported Keys	Indexes	Privileges	Column Privileges	Row IDs	Version
Info	Content	Row Count	Columns	Primary Key	
Cont_ID	Cont_FirstName	Cont_LastName	Cont_Salary		
1	Greg	Gumpert	151000.0000		
2	Tony	Lambo	60000.0000		
3	Torstein	Muller	70000.0000		

UPDATE QUERY 5

This query changes the FirstName of a Contractor by Contractor ID:

```
UPDATE Contractor  
SET Cont_FirstName = contfirstname  
WHERE Cont_ID = selection;
```

Example:

Pure SQL Query:

```
UPDATE Contractor  
SET Cont_FirstName = 'Joe'  
WHERE Cont_ID = 1;
```

Before Query:

Imported Keys	Indexes	Privileges	Column Privileges	Row IDs	Version
Info	Content	Row Count	Columns	Primary Key	
Cont_ID	Cont_FirstName	Cont_LastName	Cont_Salary		
1	Greg	Gumpert	50000.0000		
2	Tony	Lambo	60000.0000		
3	Torstein	Muller	70000.0000		

After Query:

Imported Keys	Indexes	Privileges	Column Privileges	Row IDs	Ver
Info	Content	Row Count	Columns	Primary Key	
Cont_ID	Cont_FirstName	Cont_LastName	Cont_Salary		
1	Joe	Gumpert	50000.0000		
2	Tony	Lambo	60000.0000		
3	Torstein	Muller	70000.0000		

UPDATE QUERY 6

This query changes the LastName of a Contractor by Contractor ID:

```
UPDATE Contractor  
SET Cont_LastName = contlastname  
WHERE Cont_ID = selection;
```

Example:

Pure SQL Query:

```
UPDATE Contractor  
SET Cont_FirstName = 'Smith'  
WHERE Cont_ID = 1;
```

Before Query:

Imported Keys	Indexes	Privileges	Column Privileges	Row IDs	Ver
Info	Content	Row Count	Columns	Primary Key	
Cont_ID	Cont_FirstName	Cont_LastName	Cont_Salary		
1	Greg	Gumpert	50000.0000		
2	Tony	Lambo	60000.0000		
3	Torstein	Muller	70000.0000		

After Query:

Imported Keys	Indexes	Privileges	Column Privileges	Row IDs	Ver
Info	Content	Row Count	Columns	Primary Key	
Cont_ID	Cont_FirstName	Cont_LastName	Cont_Salary		
1	Smith	Gumpert	50000.0000		
2	Tony	Lambo	60000.0000		
3	Torstein	Muller	70000.0000		

UPDATE QUERY 7

This query updates a customer FirstName by the selected Customer ID:

```
UPDATE Customer
SET Cust_FirstName = custfirstname
WHERE Cust_ID = selection;
```

Example:

Pure SQL Query:

```
UPDATE Customer
SET Cust_FirstName = 'Bret'
WHERE Cust_ID = 1;
```

Before Query:

Info	Content	Row Count	Columns	Primary Key	Exported Keys	Imported Keys	Indexes	F
	Cust_ID	Cust_FirstName	Cust_LastName	Cust_Address				
1	Abithor	Ambles	3202 Woodlane Ave, Hartbur, TX 77777					
2	Anthony	Smith	4465 Bambert St, Wild Horse, LA 12345					
3	Pam	Miller	56731 Joseph St, Olqa TX 54321					

After Query:

Info	Content	Row Count	Columns	Primary Key	Exported Keys	Imported Keys	Indexes	F
	Cust_ID	Cust_FirstName	Cust_LastName	Cust_Address				
1	Bret	Ambles	3202 Woodlane Ave, Hartbur, TX 77777					
2	Anthony	Smith	4465 Bambert St, Wild Horse, LA 12345					
3	Pam	Miller	56731 Joseph St, Olqa TX 54321					

UPDATE QUERY 8

This query updates a customer LastName by the selected Customer ID:

```
UPDATE Customer
SET Cust_LastName = custlastname
WHERE Cust_ID = selection;
```

Example:

Pure SQL Query:

```
UPDATE Customer
SET Cust_LastName = 'Hill'
WHERE Cust_ID = 1;
```

Before Query:

Info	Content	Row Count	Columns	Primary Key	Exported Keys	Imported Keys	Indexes
	Cust_ID	Cust_FirstN...	Cust_LastName	Cust_Address			
1	Abithor	Ambles	3202 Woodlane Ave, Hartbur, TX 77777				
2	Anthony	Smith	4465 Bambert St, Wild Horse, LA 12345				
3	Pam	Miller	56731 Joseph St, Olqa TX 54321				

After Query:

Info	Content	Row Count	Columns	Primary Key	Exported Keys	Imported Keys	Indexes	Pri
	Cust_ID	Cust_FirstName	Cust_LastName	Cust_Address				
1	Abithor	Hill	3202 Woodlane Ave, Hartbur, TX 77777					
2	Anthony	Smith	4465 Bambert St, Wild Horse, LA 12345					
3	Pam	Miller	56731 Joseph St, Olqa TX 54321					

UPDATE QUERY 9

This query updates the Department name by selection of Department ID:

```
UPDATE Department
SET Dept_Name = deptname
WHERE Dept_ID = selection;
```

Example:

Pure SQL Query:

```
UPDATE Department
SET Dept_Name = 'Accounting'
WHERE Dept_ID = 1;
```

Before Query:

Info	Content	Row Count	Columns	Primary Key	Exported Ke
	Dept_ID	Dept_Name			
1	Management				
2	Information Technology				
3	Architecture				
4	Contract Work				
5	Human Resources				

After Query:

Info	Content	Row Count	Columns	Primary Key	Exported
	Dept_ID	Dept_Name			
1	Accounting				
2	Information Technology				
3	Architecture				
4	Contract Work				
5	Human Resources				

UPDATE QUERY 10

This query updates the problem description by Problem ID:

```
UPDATE Problem  
SET Prblm_Desc = prblmdesc  
WHERE Prblm_ID = selection;
```

Example:

Pure SQL Query:

```
UPDATE Problem  
SET Prblm_Desc = 'moving material problems'  
WHERE Prblm_ID = 1;
```

Before Query:

Info	Content	Row Count	Columns	Primary Key	Exported
↑ Prblm...	Prblm_Desc				
1	Spatial constraints hinder movement of materials				
2	Required materials are absent				
3	Expert assistance required				

After Query:

Info	Content	Row Count	Columns	Primary Key	Exported
↑ Prblm...	Prblm_Desc				
1	moving material problems				
2	Required materials are absent				
3	Expert assistance required				

UPDATE QUERY 11

This query updates the Review status name where the review status ID matches the selection:

```
UPDATE Review_Status  
SET RvwStatus_Name = prblmdesc  
WHERE RvwStatus_ID = selection;
```

Example:

Pure SQL Query:

```
UPDATE Review_Status  
SET RvwStatus_Name = 'Partial Completion'  
WHERE RvwStatus_ID = 1;
```

Before Query:

Info	Content	Row Count	Columns	Primary
↑ RvwSta...	RvwStatus_Name			
1	Pending			
2	In Progress			
3	Attention Required			
4	Complete			

After Query:

↑ RvwSta...	RvwStatus_Name			
1	Partial Completion			
2	In Progress			
3	Attention Required			
4	Complete			

Update Query 12

This query updates an Employee_History entry by the given Employee salary:

```
UPDATE Employee_History  
SET Emp_ID = empid  
WHERE Emp_Salary = empsalary;
```

Example:

Pure SQL Query:

```
UPDATE Employee_History  
SET Emp_ID = 2  
WHERE Emp_Salary = 65000;
```

Before Query:

Info	Content	Row Count	Columns	Primary
Date_Assi...	Emp_Id	Emp_Salary		
2012-05-13	1	65000.0000		
2016-05-10	1	75000.0000		
2018-05-10	1	80000.0000		
2019-05-10	1	190000.0000		
2016-05-10	2	75000.0000		
2016-05-10	3	80000.0000		

After Query:

↑ Date_...	Emp_Id	Emp_Salary		
2012-05-13	2	65000.0000		
2016-05-10	1	75000.0000		
2016-05-10	2	75000.0000		
2016-05-10	3	80000.0000		
2018-05-10	1	80000.0000		
2019-05-10	1	190000.0000		

Update Query 13

This query changes an employees assigned project:

```
UPDATE Employee_Project  
SET Project_ID = selectedempid  
WHERE Emp_ID=empid AND Project_ID = projid;
```

Example:

Pure SQL Query:

```
UPDATE Employee_Project  
SET Project_ID = 1  
WHERE Emp_ID=3 AND Project_ID = 3;
```

Before Query:

Emp_ID	Project_ID
1	1
1	2
2	1
2	2
3	2
3	3

After Query:

Emp_ID	Project_ID
1	1
1	2
2	1
2	2
3	1
3	2

Update Query 14

This query updates a row into the Job_Material Table :

```
UPDATE Job_Material
SET Material_ID = matid
WHERE Job_ID = jobid;
```

Example:

Pure SQL Query:

```
UPDATE Job_Material
SET Material_ID = 1
WHERE Job_ID = 2;
```

Before Query:

Job_ID	Material_ID
1	2
2	2
3	3

After Query:

Job_ID	Material_ID
1	2
2	1
3	3

Update Query 15

This query updates the vehicle ID in the maintenance table:

```
UPDATE Maintenance
SET Vehicle_ID = vehicleid
WHERE Maintenance_ID = selectedid;
```

Example:

Pure SQL Query:

```
UPDATE Maintenance
SET Vehicle_ID = 2
WHERE Maintenance_ID = 1;
```

Before Query:

Info	Content	Row Count	Columns	Primary Key	Exported Keys	Imported Keys	Indexes	Pr
	Maintenance_ID	Vehicle_ID	Maintenance_Desc	Maintenance_Cost	Maintenance_Date			
1		1	Oil Change	201.3400	2019-11-06			

After Query:

Info	Content	Row Count	Columns	Primary Key	Exported Keys	Imported Keys	Ir	
	Maintenance...	Vehicle_ID	Maintenance_Desc	Maintenance_Cost	Maintenance_D...			
1		2	Oil Change	201.3400	2019-11-06			

Update Query 16

This query updates a row into the Material Table :

```
UPDATE Material
SET Material_Name = matname
WHERE Material_ID = selectionid;
```

Example:

Pure SQL Query:

```
UPDATE Material
SET Material_Name = 'Water Proof'
WHERE Material_ID = 1;
```

Before Query:

Info	Content	Row Count	Columns	Primary Key	Exported Keys	Imported Keys	Indexes	F
	Material_ID	MatType_Code	Material_CostCode	Material_Name				
1	1	IN32	Fire Proof					
2	2	PI53	3 1/4 inch					
3	3	WP24	2 x 4 inch					
4	4	PT12	12 x 24 inch					

After Query:

Info	Content	Row Count	Columns	Primary Key	Exported
↑ Materi...	MatType_C...		Material_Name	Material_CostCode	
1	1		Water Proof	IN32	
2	2		3 1/4 inch	PI53	
3	3		2 x 4 inch	WP24	
4	4		12 x 24 inch	PT12	

Update Query 17

This query updates a row into the Material_Type Table

```
UPDATE Material_Type  
SET MatType_Name = mattypename  
WHERE MatType_Code = selectionid;
```

Example:

Pure SQL Query:

```
UPDATE Material_Type  
SET MatType_Name = 'Fiberglass Insulation'  
WHERE MatType_Code = 1;
```

Before Query:

MatType_C...	MatType_Name
1	Insulation
2	PVC Pipe
3	Wooden Plank
4	Porcelain Tile

After Query:

MatType_C...	MatType_Name
1	Fiberglass Insulation
2	PVC Pipe
3	Wooden Plank
4	Porcelain Tile

Update Query 18

This query updates a row into the Project Table :

```
UPDATE Project
SET Project_Name = projname
WHERE Project_ID = selectionid;
```

Example:

Pure SQL Query:

```
UPDATE Project
SET Project_Name = 'Kitchen Renovation Woodlane'
WHERE Project_ID = 1;
```

Before Query:

Info	Content	Row Count	Columns	Primary Key	Exported Keys
	Project_ID	Project_Name			
1		Ambles Woodlane Home Kitchen Renovation			
2		Garwel Housing			
3		Olga Insulation Deal			

After Query:

Info	Content	Row Count	Columns	Primary Key	Exported Keys
	Project_ID	Project_Name			
1		Kitchen Renovation Woodlane			
2		Garwel Housing			
3		Olga Insulation Deal			

DELETE SCRIPTS:

Delete Query 1

This query Deletes a row into the Customer Table:

```
DELETE FROM Customer
WHERE Cust_ID = custid;
```

Example:

Pure SQL query:

```
DELETE FROM Customer
```

WHERE Cust_ID = 5;

Before Query:

Imported Keys	Indexes	Privileges	Column Privileges	Row IDs	Versions	
Info	Content	Row Count	Columns	Primary Key	Exported Keys	
Cust_ID	Cust_FirstName	Cust_LastName	Cust_Address			
1	Abithor	Ambles	3202 Woodlane Ave, Hartbur, TX 77777			
2	Anthony	Smith	4465 Bambert St, Wild Horse, LA 12345			
3	Pam	Miller	56731 Joseph St, Olga TX 54321			
5	Joe	Schmoe	1234 Test St, Austin, TX 12345			

After Query:

Info	Content	Row Count	Columns	Primary Key	Exported Keys	Imported Keys	Indexes	Privileges	Colu	
Cust_ID	Cust_FirstName	Cust_LastName	Cust_Address							
1	Abithor	Ambles	3202 Woodlane Ave, Hartbur, TX 77777							
2	Anthony	Smith	4465 Bambert St, Wild Horse, LA 12345							
3	Pam	Miller	56731 Joseph St, Olga TX 54321							

Delete Query 2

This query Deletes a row into the Review_Type table:

```
DELETE FROM Review_Type  
WHERE Rvw_TypeID = rvwtypeid;
```

Pure SQL Query:

```
DELETE FROM Review_Type  
WHERE Rvw_TypeID = 7;
```

Before Query:

Info	Content	Row Count	Columns	Primary Key	Exported Keys	Imported Keys	Ir
Rvw_Typ...	Rvw_TypeName	Rvw_Requirements					
1	Structural Review	Structure must meet applicable building code require...					
2	Home Inspection	Home is searched for mold, rot, termites, and other ail...					
3	Building Inspection	Home is searched for mold, rot, termites, and other ail...					
4	Preliminary Inspection	Home is inspected to ensure that job proposal is applic...					
7	Fire Code Inspection	The structure is searched for any and all fire hazards					

After Query:

Info	Content	Row Count	Columns	Primary Key	Exported Keys	Imported Keys	Indexes
Rvw_Typ...	Rvw_TypeName	Rvw_Requirements					
1	Structural Review	Structure must meet applicable building code require...					
2	Home Inspection	Home is searched for mold, rot, termites, and other ail...					
3	Building Inspection	Home is searched for mold, rot, termites, and other ail...					
4	Preliminary Inspection	Home is inspected to ensure that job proposal is applic...					

Delete Query 3

This query Deletes a row into the ContractorTable:

```
DELETE FROM Contractor  
WHERE Cont_ID = contid;
```

Example:

Pure SQL Query:

```
DELETE FROM Contractor  
WHERE Cont_ID = 5;
```

Before Query:

Imported Keys	Indexes	Privileges	Column Privileges	Row IDs	Versions
Info	Content	Row Count	Columns	Primary Key	
Cont_ID	Cont_FirstName	Cont_LastName	Cont_Salary		
1	Smith	Gumpert	50000.0000		
2	Tony	Lambo	60000.0000		
3	Torstein	Muller	70000.0000		
5	Joe	Schmoe	40000.0000		

After Query:

Imported Keys	Indexes	Privileges	Column Privileges	Row IDs	Versions
Info	Content	Row Count	Columns	Primary Key	
Cont_ID	Cont_FirstName	Cont_LastName	Cont_Salary		
1	Greg	Gumpert	50000.0000		
2	Tony	Lambo	60000.0000		
3	Torstein	Muller	70000.0000		

Delete Query 4

This query Deletes a row into the Supplier table:

```
DELETE FROM Supplier  
WHERE (suppliername);
```

Example:

Pure SQL Query:

```
DELETE FROM Supplier  
WHERE ('Wholesale Construction');
```

Before Query:

Info	Content	Row Count	Columns	Primary Key	Exported Ke
	Supplier_ID	Supplier_Name			
1	Lowes				
2	Home Depot				
3	Harbor Freight				
5	Wholesale Construction				

After Query:

	Supplier_ID	Supplier_Name			
1	Lowes				
2	Home Depot				
3	Harbor Freight				

Delete Query 5

This query Deletes a row into the Task Table

```
DELETE FROM Task  
WHERE Task_ID = taskid;
```

Example:

Pure SQL Query:

```
DELETE FROM Task  
WHERE Task_ID = 11;
```

Before Query:

Task_ID	Job_ID	Status_ID	Task_Name
1	1	5	Remove old tiles from kitchen
2	1	5	Inspect surface and ensure it is level
3	1	2	Place new tiles
4	1	1	Grout tiles
5	1	1	Clean tiles and inspect surface
6	3	1	Search outside roof for imperfections
7	3	1	Search inside roof for imperfections
8	3	1	Search upstairs for signs of water damage
9	3	1	Search downstairs for signs of water damage
11	2	3	Replace lighting on front porch

After Query:

Task_ID	Job_ID	Status_ID	Task_Name
1	1	5	Remove old tiles from kitchen
2	1	5	Inspect surface and ensure it is level
3	1	2	Place new tiles
4	1	1	Grout tiles
5	1	1	Clean tiles and inspect surface
6	3	1	Search outside roof for imperfections
7	3	1	Search inside roof for imperfections
8	3	1	Search upstairs for signs of water damage
9	3	1	Search downstairs for signs of water damage

Delete Query 6

This query Deletes a row into the Task_Problem Table:

```
DELETE FROM TASK_Problem  
WHERE (taskid, prblmid);
```

Example:

Pure SQL Query:

```
DELETE FROM Task_Problem  
WHERE Task_ID = 1 AND Prblm_ID = 3;
```

Before Query:

Task_ID	Prblm_ID
2	1
2	2
1	3

After Query:

Task_ID	Prblm_ID
2	1
2	2

Delete Query 7

This query Deletes a row into the Vehicle Table:

```
DELETE FROM Vehicle  
WHERE Vehicle_ID = vehicleid;
```

Example:

Pure SQL Query:

```
DELETE FROM Vehicle  
WHERE Vehicle_ID = 5;
```

Before Query:

Info	Content	Row Count	Columns	Primary Key	Exported Keys	Imported
Vehicle_ID	Fleet_ID	Vehicle_Make	Vehicle_Model	Vehicle_Color		
1	1	Ford	Transit	Red		
2	1	Ford	Transit	White		
3	1	Ford	Transit	Blue		
5	1	Ford	F-150	Blue		

After Query:

Info	Content	Row Count	Columns	Primary Key	Exported Keys
↑ Vehicl...	Fleet...	Vehicle_Make	Vehicle_Model	Vehicle_Color	
1	1	Ford	Transit	Red	
2	1	Ford	Transit	White	
3	1	Ford	Transit	Blue	

Delete Query 8

This query Deletes a row into Work_Order:

```
DELETE FROM Work_Order  
WHERE Cont_ID = contid AND Project_ID = projid;
```

Example:

Pure SQL Query:

```
DELETE FROM Work_Order  
WHERE Cont_ID = 3 AND Project_ID = 2;
```

Before Query:

Cont_ID	Project_ID
1	2
2	1
2	2
3	1
3	2

After Query:

Cont_ID	Project_ID
1	2
2	1
2	2
3	1

Delete Query 9

This Deletes a row into the Department Table:

```
DELETE FROM Department  
WHERE Dept_ID = deptid;
```

Example:

Pure SQL Query:

```
DELETE FROM Department  
WHERE Dept_ID = 7;
```

Before Query:

Imported Keys	Indexes	Privileges	Column Privileges	Row ID:
Info	Content	Row Count	Columns	Primary
Dept_ID	Dept_Name			
1	Management			
2	Information Technology			
3	Architecture			
4	Contract Work			
5	Human Resources			
7	Accounting			

After Query:

Info	Content	Row Count	Columns	Primary Key	Exported Ke
Dept_ID	Dept_Name				
1	Management				
2	Information Technology				
3	Architecture				
4	Contract Work				
5	Human Resources				

Delete Query 10

This query Deletes a row into the Problem table:

```
DELETE FROM Problem  
WHERE Prblm_ID = prblmid;
```

Example:

Pure SQL Query:

```
DELETE FROM Problem  
WHERE Prblm_ID = 6;
```

Before Query:

Imported Keys	Indexes	Privileges	Column Privileges	Row ID
Info	Content	Row Count	Columns	Prim
↑ Prblm...	Prblm_Desc			
1	Spatial constraints hinder movement of materials			
2	Required materials are absent			
3	Expert assistance required.			
6	Needs evaluation			

After Query:

Info	Content	Row Count	Columns	Primary Key	Exported I
↑ Prblm...	Prblm_Desc				
1	Spatial constraints hinder movement of materials				
2	Required materials are absent				
3	Expert assistance required				

Delete Query 11

This query Deletes a row into the Review Status table:

```
DELETE FROM Review_Status  
WHERE RvwStatus_ID = rwstatusid;
```

Example:

Pure SQL Query:

```
DELETE FROM Review_Status  
WHERE RvwStatus_ID = 6;
```

Before Query:

↑ RvwSta...	RvwStatus_Name
1	Partial Completion
2	In Progress
3	Attention Required
4	Complete
6	Incomplete

After Query:

Info	Content	Row Count	Columns	Primary
↑ RvwSta...	RvwStatus_Name			
1	Pending			
2	In Progress			
3	Attention Required			
4	Complete			

Delete Query 12

This query Deletes a row into the Employee History Table:

```
DELETE FROM Employee_History  
WHERE Emp_Salary=empsalary;
```

Example:

Pure SQL Query:

```
DELETE FROM Employee_History  
WHERE Emp_Salary=50000;
```

Before Query:

Info	Content	Row Count	Columns	Prim
↑	Date_...	Emp_Id	Emp_Salary	
	2012-05-13	1	65000.0000	
	2016-05-10	1	75000.0000	
	2016-05-10	2	75000.0000	
	2016-05-10	3	80000.0000	
	2018-05-10	1	80000.0000	
	2019-05-10	1	190000.0000	
	2019-11-11	2	50000.0000	

After Query:

Info	Content	Row Count	Columns	Prim
	Date_Assi...	Emp_Id	Emp_Salary	
	2012-05-13	1	65000.0000	
	2016-05-10	1	75000.0000	
	2018-05-10	1	80000.0000	
	2019-05-10	1	190000.0000	
	2016-05-10	2	75000.0000	
	2016-05-10	3	80000.0000	

Delete Query 13

This query assigns deletes an entry from the Employee_Project Table:

```
DELETE FROM Employee_Project  
WHERE Emp_ID=empid AND Project_ID = projid;
```

Example:

Pure SQL Query:

```
DELETE FROM Employee_Project  
WHERE Emp_ID=3 AND Project_ID = 1;
```

Before Query:

Emp_ID	Project_ID
1	1
1	2
2	1
2	2
3	1
3	2
3	3

After Query:

Emp_ID	Project_ID
1	1
1	2
2	1
2	2
3	2
3	3

Delete Query 14

This query Deletes a row into the Job_Material Table :

```
DELETE FROM Job_Material  
WHERE (jobid, materialid);
```

Example:

Pure SQL Query:

```
DELETE FROM Job_Material  
WHERE Job_ID = 3 AND Material_ID = 2;
```

Before Query:

Job_ID	Material_ID
1	2
2	2
3	2
3	3

After Query:

Job_ID	Material_ID
1	2
2	2
3	3

Delete Query 15

This query Deletes a row into the Maintenance table:

```
DELETE FROM Maintenance  
WHERE Maintenance_ID = maintID;
```

Example:

Pure SQL Query:

```
DELETE FROM Maintenance  
WHERE Maintenance_ID = 3;
```

Before Query:

Info	Content	Row Count	Columns	Primary Key	Exported Keys	Imported Keys
Maintenance...	Vehicle_ID	Maintenance_Desc	Maintenance_Cost	Maintenance_D...		
1	1	Oil Change	201.3400	2019-11-06		
3	2	Tire rotation	40.3400	2019-11-11		

After Query:

Info	Content	Row Count	Columns	Primary Key	Exported Keys	Imported Keys	Indexes	Pr
Maintenance_ID	Vehicle_ID	Maintenance_Desc	Maintenance_Cost	Maintenance_Date				
1	1	Oil Change	201.3400	2019-11-06				

Delete Query 16

This query Deletes a row into the Material Table :

```
DELETE FROM Material
WHERE Material_ID=materialid;
```

Example:

Pure SQL Query:

```
DELETE FROM Material
WHERE Material_ID=7;
```

Before Query:

Info	Content	Row Count	Columns	Primary Key	Exported Keys	Imported Keys	Indexes
	Material_ID	MatType_C...	Material_Name			Material_CostCode	
1	1	Fire Proof	IN32				
2	2	3 1/4 inch	PI53				
3	3	2 x 4 inch	WP24				
4	4	12 x 24 inch	PT12				
7	6	10x12 feet	PT32				

After Query:

Info	Content	Row Count	Columns	Primary Key	Exported Keys	Imported Keys	Indexes	F
	Material_ID	MatType_Code	Material_CostCode	Material_Name				
1	1	IN32	Fire Proof					
2	2	PI53	3 1/4 inch					
3	3	WP24	2 x 4 inch					
4	4	PT12	12 x 24 inch					

Delete Query 17

This query Deletes a row into the Material_Type Table

```
DELETE FROM Material_Type  
WHERE MatType_Code = mattypecode;
```

Example:

Pure SQL Query:

```
DELETE FROM Material_Type  
WHERE MatType_Code = 6;
```

Before Query:

Info	Content	Row Count	Columns	Primary Key	Exported K
	MatType_C...		MatType_Name		
1			Insulation		
2			PVC Pipe		
3			Wooden Plank		
4			Porcelain Tile		
6			Plywood Sheet		

After Query:

Info	Content	Row Count	Columns	Primary Key	Exported K
	MatType_C...		MatType_Name		
1			Insulation		
2			PVC Pipe		
3			Wooden Plank		
4			Porcelain Tile		

Delete Query 18

This query Deletes a row into the Project Table :

```
DELETE FROM Project  
WHERE (projectname);
```

Example:

Pure SQL Query:

```
DELETE FROM Project  
WHERE ('Holmes Developers Job');
```

Before Query:

Info	Content	Row Count	Columns	Primary Key	Exported Ke
	Project_ID	Project_Name			
1	Ambles Woodlane Home Kitchen Renovation				
2	Garwel Housing				
3	Olga Insulation Deal				
5	Holmes Developers Job				

After Query:

Info	Content	Row Count	Columns	Primary Key	Exported Ke
	Project_ID	Project_Name			
1	Ambles Woodlane Home Kitchen Renovation				
2	Garwel Housing				
3	Olga Insulation Deal				

Problems List:

- Unable to export data from Buildertrend application to create an invoice.
- No existing database to be able to store all information locally. It is all stored on the Buildertrend application.

Requirement List:

- User interface should be friendly and easy to use and learn.
- Ability to store all client information.
- Able to display tables of current project, customers, and contractor work.
- Allow to create an invoice on the same application.
- Permit the storage of additional notes and pictures that will be matched to project.

Team Status Report

Group Name: CoogNet

Project Manager: Kavon Sabet/ Daniel Howard

Date: August 23, 2019

Our first team status report. This week the CoogNet team has been working on the following deliverables listed below. We have not put much work into our project yet, we are now getting acquainted with what is desired and needs to be done. The following is a breakdown for what each member is expected to do and has done so far.

CoogNet	Deliverable Description	Start Date/Time	Hours Worked	Est. Hours Remaining
Kavon Sabet				
	Group Meeting	8/21/2019 19:00	1	N/A
	Normalization	8/21/2019 8:00	0	4
	CSV Files	8/21/2019 8:00	0	2
	SQL General Programming	8/21/2019 8:00	0	6
			1	
Daniel Howard				
	Group Meeting	8/21/2019 8:00	0	N/A
	Java Forms	8/21/2019 8:00	0	40
	ERD Tables	8/21/2019 8:00	2	2
	SQL General Programming	8/21/2019 8:00	0	6
			2	
Jorge Sanchez				
	Group Meeting	8/21/2019 19:00	1	N/A
	Java General Programming	8/21/2019 8:00	0	20
	Normalization	8/21/2019 8:00	0	4
	SQL General Programming	8/21/2019 8:00	0	6
			1	
Ayoub Fares				

SQL General Programming	8/21/2019 8:00	0	6
Java General Programming	8/21/2019 8:00	0	20
Java forms	8/21/2019 8:00	0	20
Group Meeting	8/21/2019 19:00	1	N/A
		1	

Tyler Nullmeier

SQL General Programming	8/21/2019 8:00	0	6
Java General Programming	8/21/2019 8:00	0	20
ERD Tables	8/21/2019 8:00	2	2
CSV Files	8/21/2019 8:00	0	2
		2	

Trent Jones

Group Meeting	8/21/2019 19:00	1	N/A
SQL General Programming	8/21/2019 8:00	0	6
Normalization	8/21/2019 8:00	0	4
CSV Files	8/21/2019 8:00	0	2
		1	

Aleena Khan

Group Meeting	8/21/2019 19:00	1	N/A
SQL General Programming	8/21/2019 8:00	0	6
Normalization	8/21/2019 8:00	0	4
CSV Files	8/21/2019 8:00	0	2
		1	

Aidahta Natama

Group Meeting	8/21/2019 19:00	1	N/A
SQL General Programming	8/21/2019 8:00	0	6
Normalization	8/21/2019 8:00	0	4
CSV Files	8/21/2019 8:00	0	2
		1	

Eduardo Tostado

Group Meeting	8/21/2019 19:00	1	N/A
Java General Programming	8/21/2019 8:00	0	6
Java Forms	8/21/2019 8:00	0	4
CSV Files	8/21/2019 8:00	0	2
		1	

Team Status Report

Group Name: CoogNet

Project Manager: Kavon Sabet/ Daniel Howard

Date: August 30, 2019

Week two status report. This week CoogNet has been working on the following deliverables listed below. This past week we focused on our ERD and data dictionary in an attempt to get 30 tables. The following is a breakdown per person of what each member is expected to do and has done so far.

CoogNet	Deliverable Description	Start Date/Time	Hours Worked	Est. Hours Remaining
Kavon Sabet				
	Group Meeting	8/26/2019 19:00	2	N/A
	SQL Scripts	8/26/2019 8:00	0	4
	CSV Files	8/26/2019 8:00	0	2
	SQL General Programming	8/26/2019 8:00	0	6
			2	
Daniel Howard				
	Group Meeting	8/26/2019 8:00	2	N/A
	Java Forms	8/26/2019 8:00	0	40
	ERD Tables	8/26/2019 8:00	2	2
	SQL General Programming	8/26/2019 8:00	0	6
			4	
Jorge Sanchez				
	Group Meeting	8/26/2019 19:00	2	N/A
	Java General Programming	8/26/2019 8:00	0	20
	Normalization	8/26/2019 8:00	0	4
	SQL General Programming	8/26/2019 8:00	0	6
			2	
Ayoub Fares				
	SQL General Programming	8/26/2019 8:00	0	6

	Java General Programming	8/26/2019 8:00	0	20
	Java forms	8/26/2019 8:00	0	20
	Group Meeting	8/26/2019 19:00	2	N/A
			2	
Tyler Nullmeier	SQL General Programming	8/26/2019 8:00	0	6
	Java General Programming	8/26/2019 8:00	0	20
	ERD Tables	8/26/2019 8:00	2	2
	CSV Files	8/26/2019 8:00	0	2
			2	
Trent Jones	Group Meeting	8/26/2019 19:00	2	N/A
	SQL Scripts	8/26/2019 8:00	0	6
	Normalization	8/26/2019 8:00	0	4
	CSV Files	8/26/2019 8:00	0	2
			2	
Aleena Khan	Group Meeting	8/26/2019 19:00	2	N/A
	SQL Scripts	8/26/2019 8:00	0	6
	Normalization	8/26/2019 8:00	0	4
	CSV Files	8/26/2019 8:00	0	2
			2	
Aidahta Natama	Group Meeting	8/26/2019 19:00	2	N/A
	SQL Scripts	8/26/2019 8:00	0	6
	Normalization	8/26/2019 8:00	0	4
	CSV Files	8/26/2019 8:00	0	2
			2	
Eduardo Tostado	Group Meeting	8/21/2019 19:00	2	N/A
	Java General Programming	8/21/2019 8:00	0	6
	Java Forms	8/21/2019 8:00	0	4
	CSV Files	8/21/2019 8:00	0	2
			2	

Please copy and paste the latest version/screenshot of your projects Entity Relational Diagram (ERD):

Team Status Report

Group Name: CoogNet
 Project Manager: Kavon Sabet/ Daniel Howard
 2019

Date: October 21,

Week eight status report. This week CoogNet has been working on the following deliverables listed below. The following is a breakdown of what each member is expected to do and has done so far.

CoogNet	Deliverable Description	Start Date/Time	Hours Worked	Est. Hours Remaining
Kavon Sabet				
		10/15/2019		
	Group Meeting	19:00	2	N/A
	SQL Scripts	10/15/2019 8:00	6	0
	CSV Files	10/15/2019 8:00	2	0
	SQL General Programming	10/15/2019 8:00	6	0
			16	
Daniel Howard				
	Group Meeting	10/15/2019 8:00	2	N/A
	Normalization	10/15/2019 8:00	4	0
	ERD Tables	10/15/2019 8:00	4	0
	SQL General Programming	10/15/2019 8:00	6	0
			16	
Jorge Sanchez				
		10/15/2019		
	Group Meeting	19:00	2	N/A
	CSV Files	10/15/2019 8:00	2	0
	Normalization	10/15/2019 8:00	4	0
	SQL General Programming	10/15/2019 8:00	6	0
			14	
Ayoub Fares				
	CSV Files	10/15/2019 8:00	2	0
	Java General Programming	10/15/2019 8:00	16	4

	Java forms	10/15/2019 8:00	16	24
		10/15/2019		
	Group Meeting	19:00	2	N/A
			36	
Tyler Nullmeier	SQL General Programming	10/15/2019 8:00	6	0
	Java General Programming	10/15/2019 8:00	16	4
	ERD Tables	10/15/2019 8:00	4	0
	CSV Files	10/15/2019 8:00	2	0
			28	
Eduardo Tostado	SQL General Programming	10/15/2019 8:00	6	0
	Java General Programming	10/15/2019 8:00	16	4
	ERD Tables	10/15/2019 8:00	4	0
	CSV Files	10/15/2019 8:00	2	0
			28	
Trent Jones	Group Meeting	10/15/2019 8:00	2	N/A
	SQL General Programming	10/15/2019 8:00	14	6
	Normalization	10/15/2019 8:00	4	0
	CSV Files	10/15/2019 8:00	1	1
			19	
Aleena Khan		10/15/2019		
	Group Meeting	19:00	2	N/A
	SQL Scripts	10/15/2019 8:00	6	0
	Normalization	10/15/2019 8:00	4	0
	CSV Files	10/15/2019 8:00	2	0
			14	
Aidahta Natama		10/15/2019		
	Group Meeting	19:00	2	N/A
	SQL Scripts	10/15/2019 8:00	6	0
	Normalization	10/15/2019 8:00	4	0
	CSV Files	10/15/2019 8:00	2	0
			14	

Team Status Report

Group Name: CoogNet
 Project Manager: Kavon Sabet/ Daniel Howard
 2019

Date: October 27,

Week nine status report. This week CoogNet has been working on the following deliverables listed below. The following is a breakdown of what each member is expected to do and has done so far.

CoogNet	Deliverable Description	Start Date/Time	Hours Worked	Est. Hours Remaining
Kavon Sabet				
		10/22/2019		
	Group Meeting	19:00	2	N/A
	SQL Scripts	10/22/2019 8:00	6	0
	CSV Files	10/22/2019 8:00	2	0
	SQL General Programming	10/22/2019 8:00	6	0
			16	
Daniel Howard				
	Group Meeting	10/22/2019 8:00	2	N/A
	Normalization	10/22/2019 8:00	4	0
	ERD Tables	10/22/2019 8:00	4	0
	SQL General Programming	10/22/2019 8:00	6	0
			16	
Jorge Sanchez				
		10/22/2019		
	Group Meeting	19:00	2	N/A
	CSV Files	10/22/2019 8:00	2	0
	Normalization	10/22/2019 8:00	4	0
	SQL General Programming	10/22/2019 8:00	6	0
			14	
Ayoub Fares				
	CSV Files	10/22/2019 8:00	2	0
	Java General Programming	10/22/2019 8:00	20	0

	Java forms	10/22/2019 8:00	20	20
		10/22/2019		
	Group Meeting	19:00	2	N/A
			44	
Tyler Nullmeier				
	SQL General Programming	10/22/2019 8:00	6	0
	Java General Programming	10/22/2019 8:00	20	0
	ERD Tables	10/22/2019 8:00	4	0
	CSV Files	10/22/2019 8:00	2	0
			32	
Eduardo Tostado				
	SQL General Programming	10/22/2019 8:00	6	0
	Java General Programming	10/22/2019 8:00	20	0
	ERD Tables	10/22/2019 8:00	4	0
	CSV Files	10/22/2019 8:00	2	0
			32	
Trent Jones				
	Group Meeting	10/22/2019 8:00	2	N/A
	SQL General Programming	10/22/2019 8:00	17	3
	Normalization	10/22/2019 8:00	4	0
	CSV Files	10/22/2019 8:00	1	1
			22	
Aleena Khan				
		10/22/2019		
	Group Meeting	19:00	2	N/A
	SQL Scripts	10/22/2019 8:00	6	0
	Normalization	10/22/2019 8:00	4	0
	CSV Files	10/22/2019 8:00	2	0
			14	
Aidahta Natama				
		10/22/2019		
	Group Meeting	19:00	2	N/A
	SQL Scripts	10/22/2019 8:00	6	0
	Normalization	10/22/2019 8:00	4	0
	CSV Files	10/22/2019 8:00	2	0
			14	

Team Status Report

Group Name: CoogNet
 Project Manager: Kavon Sabet/ Daniel Howard
 2019

Date: November 3,

Week ten status report. This week CoogNet has been working on the following deliverables listed below. We're finishing up what's left of things to do, only a few forms left to complete. The following is a breakdown of what each member is expected to do and has done so far.

CoogNet	Deliverable Description	Start Date/Time	Hours Worked	Est. Hours Remaining
Kavon Sabet				
		10/29/2019		
	Group Meeting	19:00	2	N/A
	SQL Scripts	10/29/2019 8:00	6	0
	CSV Files	10/29/2019 8:00	2	0
	SQL General Programming	10/29/2019 8:00	6	0
			16	
Daniel Howard				
	Group Meeting	10/29/2019 8:00	2	N/A
	Normalization	10/29/2019 8:00	4	0
	ERD Tables	10/29/2019 8:00	4	0
	SQL General Programming	10/29/2019 8:00	6	0
			16	
Jorge Sanchez				
		10/29/2019		
	Group Meeting	19:00	2	N/A
	CSV Files	10/29/2019 8:00	2	0
	Normalization	10/29/2019 8:00	4	0
	SQL General Programming	10/29/2019 8:00	6	0
			14	
Ayoub Fares				
	CSV Files	10/29/2019 8:00	2	0

	Java General Programming	10/29/2019 8:00	20	0
	Java forms	10/29/2019 8:00	32	8
		10/29/2019		
	Group Meeting	19:00	2	N/A
			56	
Tyler Nullmeier	SQL General Programming	10/29/2019 8:00	6	0
	Java General Programming	10/29/2019 8:00	20	0
	ERD Tables	10/29/2019 8:00	4	0
	CSV Files	10/29/2019 8:00	2	0
			32	
Eduardo Tostado	SQL General Programming	10/29/2019 8:00	6	0
	Java General Programming	10/29/2019 8:00	20	0
	ERD Tables	10/29/2019 8:00	4	0
	CSV Files	10/29/2019 8:00	2	0
			32	
Trent Jones	Group Meeting	10/29/2019 8:00	2	N/A
	SQL General Programming	10/29/2019 8:00	19	1
	Normalization	10/29/2019 8:00	4	0
	CSV Files	10/29/2019 8:00	1	1
			24	
Aleena Khan		10/29/2019		
	Group Meeting	19:00	2	N/A
	SQL Scripts	10/29/2019 8:00	6	0
	Normalization	10/29/2019 8:00	4	0
	CSV Files	10/29/2019 8:00	2	0
			14	
Aidahta Natama		10/29/2019		
	Group Meeting	19:00	2	N/A
	SQL Scripts	10/29/2019 8:00	6	0
	Normalization	10/29/2019 8:00	4	0
	CSV Files	10/29/2019 8:00	2	0
			14	

Team Status Report

Group Name: CoogNet

Project Manager: Kavon Sabet/ Daniel Howard

Date: November 10, 2019

Final week of the project before the presentation. At this point we're all done with our parts, we just need to plan on our presentation.

CoogNet	Deliverable Description	Start Date/Time	Hours Worked	Est. Hours Remaining
Kavon Sabet				
		11/10/2019		
	Group Meeting	19:00	2	N/A
	SQL Scripts	11/10/2019 8:00	6	0
	CSV Files	11/10/2019 8:00	2	0
	SQL General Programming	11/10/2019 8:00	6	0
			16	
Daniel Howard				
	Group Meeting	11/10/2019 8:00	2	N/A
	Normalization	11/10/2019 8:00	4	0
	ERD Tables	11/10/2019 8:00	4	0
	SQL General Programming	11/10/2019 8:00	6	0
			16	
Jorge Sanchez				
		11/10/2019		
	Group Meeting	19:00	2	N/A
	CSV Files	11/10/2019 8:00	2	0
	Normalization	11/10/2019 8:00	4	0
	SQL General Programming	11/10/2019 8:00	6	0
			14	
Ayoub Fares				
	CSV Files	11/10/2019 8:00	2	0
	Java General Programming	11/10/2019 8:00	20	0
	Java forms	11/10/2019 8:00	40	0

		11/10/2019		
	Group Meeting	19:00	2	N/A
			64	
Tyler Nullmeier	SQL General Programming	11/10/2019 8:00	6	0
	Java General Programming	11/10/2019 8:00	20	0
	ERD Tables	11/10/2019 8:00	4	0
	CSV Files	11/10/2019 8:00	2	0
			32	
Eduardo Tostado	SQL General Programming	11/10/2019 8:00	6	0
	Java General Programming	11/10/2019 8:00	20	0
	ERD Tables	11/10/2019 8:00	4	0
	CSV Files	11/10/2019 8:00	2	0
			32	
Trent Jones	Group Meeting	11/10/2019 8:00	2	N/A
	SQL General Programming	11/10/2019 8:00	20	0
	Normalization	11/10/2019 8:00	4	0
	CSV Files	11/10/2019 8:00	2	0
			28	
Aleena Khan		11/10/2019		
	Group Meeting	19:00	2	N/A
	SQL Scripts	11/10/2019 8:00	6	0
	Normalization	11/10/2019 8:00	4	0
	CSV Files	11/10/2019 8:00	2	0
			14	
Aidahta Natama		11/10/2019		
	Group Meeting	19:00	2	N/A
	SQL Scripts	11/10/2019 8:00	6	0
	Normalization	11/10/2019 8:00	4	0
	CSV Files	11/10/2019 8:00	2	0
			14	

Please copy and paste the latest version/screenshot of your projects Entity Relational Diagram (ERD):

