Gonzalez, Adrian

Abstract

This will explain the process and overview of implementing a database for business needs.

FINAL ITERATION 5

Table of Contents

**Project Direction Overview2**

**Project Use Cases and Fields** **2-7**

**Structural Database Rules7-8**

**Conceptual Entity-Relationship Diagram9**

**DBMS Physical ERD10**

**Index Identification and Creations** **11-12**

**Store Procedure and Execution and Explanations13-16**

**Trigger Creation and Use17-19**

**Question Identification and Explanations** **20**

**Query Executions and Explanations20-23**

**Summary and Reflection23**

Project Direction Overview

For this project, a database will be designed and implemented for my employer Tesla Motors. We are currently building a Gigafactory in Austin, TX and it would be important to have a database which can manage assets, manage departments, manage employees, and managing transactions or logging. For asset management, information such as the assetID, assetLocation, systemOwner, and assetInformation will be useful for helping manage assets at the company. The goal for managing this part of the database would be to easily gain access to the name of the systemOwner, assetLocation, and assetInformation by simply passing the ID through a procedure. For instance,

EXECUTE GETASSETINFO 1029389238;

Or if we need change the system owner, then it could be changed from a procedure. For example,

EXECUTE CHANGEASSETOWNER 1029389238, 203948093;

Other types of management, such as departmental and employee management is important, so those will also require procedures for their very specific operations.

This will facilitate the ease of use when managing all the assets, departments, and employees at the facility. However, the main goal of this database management system is to really emphasize the ease of management and use. At the Tesla Giga Factory, managing this information is difficult since the interfaces they use have a high learning curve. Therefore, with this database management system, I want to ensure that it is as simple as calling a procedure or command, then yielding the desired results.

Database Use Case

One of the most important parts of use of this database is through a program that will be developed which will make good use of the procedures and database queries from the database.

*Tesla Database Management Software/Non-DB Admin Use Case*

1. A non-administrator needs to modify access records from the Tesla Giga Factory Database.
2. The application asks the user to login with their Tesla ID.
3. The application will call a procedure which will verify whether the employee (who is a non-admin) is allowed to access data from the database.
4. If yes, a trigger will be activated which will log who logged into the DBMS in the action log. If no, they are rejected; if they are rejected, the step STOPS here.
5. The user types in the field which they are looking for and hits *request*
6. The application will call a procedure which will return the desired fields and data back to the user.
7. The data request will be logged on the action log (which database admins are only allowed to see) through a trigger.

*Tesla Database Management Software/DB Admin Use Case*

1. An administrator needs to modify access records from the Tesla Giga Factory Database.
2. The application asks the user to login with their Tesla ID.
3. The application will call a procedure which will verify whether the user is allowed to access data from the database.
4. If yes, the application will call a procedure which will log who logged into the DBMS in the action log. If no, they are rejected; if they are rejected, the step STOPS here.
5. The user types in the field which they are looking for and hits *request*.
6. The application will call a procedure which will return the desired fields and data back to the user.
7. The admin calls a query or procedure from the built-in command line (which is only allowed for database administrators).
8. The database trigger logs the query or procedure into the action log (which database admins are only allowed to see).
9. The software returns the data from the database back to the DB admin.

One of the most important tables is the System Owner/System Engineer table, because it contains data regarding who manages or owns what assets, if they can access the database for management purposes, and personal data.

|  |  |  |
| --- | --- | --- |
| **Field** | **What it Stores** | **Why it’s Needed** |
| OwnerID | This contains the ID of the System Owner. | This is needed because the ID is very important for referencing this row, or set, of data. |
| EmployeeID | This contains the employee ID of the System Owner. | This is important because personal information such as their full name, email, phone number, and department. |
| DBPermissions | This is a code which determines what permissions they have when trying to access the database. | This is important because some system owners are only allowed to do specific things with the database. Some may have higher privileges, whilst some may have lower privileges. |

Another important table in the database will be the employee table because it contains personal information regarding the employee.

|  |  |  |
| --- | --- | --- |
| **Field** | **What it Stores** | **Why it’s Needed** |
| EmployeeID | This contains the ID of the employee. | This is important because one can gain personal information regarding an employee. |
| FirstName | This contains the first name of the employee. | It’s important to know the first name of the employee. |
| LastName | This contains the last name of the employee | It’s important to know the last name of the employee. |
| PhoneNumber | This contains the phone number of the employee’s phone. | It’s important to know the phone number of an employee, so that they could be contacted. |
| EmailID | This contains the email of the employee. | It’s important to know the email of an employee so there is another way of contacting them. |
| DepartmentID | This contains the departmentID which references the department the employee works under. | It’s important to know which department an employee works under. |
| JobID | This contains the jobID which references the job title the employee has. | It’s essential to know what job the employee is doing. Especially if one is trying to manage their employees. |

Since the employee table relies on an email ID, there is a need for an email table.

|  |  |  |
| --- | --- | --- |
| **Field** | **What it Stores** | **Why it’s Needed** |
| EmailID | This is the ID of an email address. | This is important because this will reference the email address and its information. |
| EmailAddress | This is the email address. | Every email needs an email address. |
| Encrypted | This determines the type of encryption the email will have. | Some employees will have less important information than others. All system owners will always have highly encrypted emails. |

Another important table in the database will be the jobs table because it contains the type of jobs that are available or valid.

|  |  |  |
| --- | --- | --- |
| **Field** | **What it Stores** | **Why it’s Needed** |
| JobID | This is the ID of a job. | This is important because all information regarding this job can be accessed from this ID. |
| JobName | This is the name of the job. | It’s important to know the name of the position because every job needs a title. |
| JobDescription | This is the description of the job. | It’s important to know the job responsibilities and duties. |

Another significant table in the database is the departments table because it contains the types of departments that are available at the Tesla Giga Factory.

|  |  |  |
| --- | --- | --- |
| **Field** | **What it Stores** | **Why it’s Needed** |
| DepartmentID | This is the ID of a department. | This is important because all information regarding a department will be referenced from this ID. |
| DepartmentName | This is the name of a department. | It’s important to know the department name because it suggests the responsibilities and duties for the department. |
| LocationID | This is the ID of the location which the department likes to work at. | It’s important to know the location where the department works at in the factory. |

Since the department relies on location, there is the need for a location table since it contains the position and site name for the factory.

|  |  |  |
| --- | --- | --- |
| **Field** | **What it Stores** | **Why it’s Needed** |
| LocationID | This is the ID of the location. | This is important because all information regarding the location at the Tesla factory. |
| LocationName | This is the name of the location at the Tesla Factory. | It’s important to know the name of the location in order to gain confirmation if traveling there. |
| Row | This is the row relative to the factory. | At the Tesla factory, locations are determined by rows and columns, kind of like a data base, relative to the factory. It is important to know the row relative to the factory. |
| Column | This is the column relative to the factory. | At the Tesla factory, locations are determined by rows and columns, kind of like a data base, relative to the factory. It is important to know the column relative to the factory. |

The last table, which is of great important since it manages the assets of the Tesla Giga Factory, is the assets table, which will contain information of assets at the Tesla Giga Factory.

|  |  |  |
| --- | --- | --- |
| **Field** | **What it Stores** | **Why it’s Needed** |
| AssetID | This is the ID of the asset. | This is important because all information regarding the asset will be referenced from the ID. |
| AssetName | This is the name of the asset. | It’s important to know the name of the asset in order to identify which asset it is. |
| OwnerID | This is containing the ID of the system owner. | This is important because these assets will have system owners which may need to be referenced for management purposes. |
| LocationID | This is the location of the asset at the Tesla Factory. | At the Tesla factory, assets may change locations, so it is important to keep track of the location at the Tesla Giga Factory. |

Structural Database Rules

*General Database Rules*

1. There are three types of users.
   1. There are users that can modify and view all records.
   2. There are users who cannot modify but view some records.
   3. There are users who cannot modify, nor view any records.
2. All actions in the database must be recorded in logged in the action log. This includes:
   1. Requesting data from the database
   2. Adding data into the database
   3. Removing data from the database
   4. Verifying the permissions of a user
3. The database must meet physical computer requirements for the system to work at optimized and efficient speeds.

*Access Rules*

1. Database Administrators have the following permissions
   1. They can modify and view all records
   2. They can create procedures and triggers
   3. They can query data
2. Non-administrators have the following permissions
   1. They cannot modify records, unless given permission
   2. They can only view very limited data allowed to them.
   3. They cannot create procedures and triggers
   4. They can query certain data they are allowed to view.

*Location Table Rules*

1. The location must have an ID
2. The location must have a name
3. The location must have a row and column
4. The row cannot be greater than 1000, nor less than 1
5. The column cannot be greater than 10, nor less than 1

*System Owner Table Rules*

1. The system owner must have an ID
2. The system owner must be an employee
3. The system owner must have level 2 access to the database
4. There must be an asset under the System Owner’s name
5. A system owner can own many assets.

*Employee Table Rules*

1. All employees must have a first name and a last name
2. All employees must have an email and a phone number
3. All employees must have an ID
4. All employees must have a JobID

*Asset Table Rules*

1. Assets can only have one system owner.
2. Only the system owner can query information regarding the asset (including db admins).
3. There can only be one asset location.
4. There can only be one ID per asset.

*Email Table Rules*

1. All emails must have an ID
2. All emails must have an address.
3. If encryption is left empty, the automatic encryption level is defined as 1.

*Job Table Rules*

1. All jobs must have an ID
2. The job must have a name
3. It’s ok to leave the job description empty

*Department Table Rules*

1. All departments must have an ID
2. All departments must have a name
3. All departments must have a location

Conceptual Entity Relationship Diagram

Diagram

Description automatically generated

*Design Reasoning*

The System Owner table has a generalization relationship with Employee since the system owner is an employee who has extra permissions such as database access, as well as owning an asset.

Since the System Owner is a single employee. It is an only one relationship.

The email table has an only one relationship with employee because there can only be one email per employee.

The job table has an only one relationship with employee because each employee can only have one job.

The System Owner has a one-to-many relationship with assets because the System Owner can have many assets, but the asset can only have one system owner.

The asset table an only one relationship with location because the asset can only be at one location.

DBMS Physical ERD

A picture containing text, receipt, screenshot

Description automatically generated*Description*

The Employee table consists of a primary key which is EmployeeID (DECIMAL(12) NOT NULL), two foreign keys which are the EmailID(DECIMAL(12) NOT NULL) and DepartmentID(DECIMAL(12) NOT NULL), and the rest being FirstName(VARCHAR(50) N/NULL), LastName (VARCHAR(50) N/NULL), and PhoneNumber (BIGINT NOT NULL).

The Email table consists of a primary key which is EmailID (DECIMAL(12) NOT NULL) and the rest being Jobname (VARCHAR(50) NOT NULL) and JobDescription (VARCHAR(500) NOT NULL).

The Job table consists of a primary key which is JobID (DECIMAL(12) NOT NULL) and the rest being JobName(VARCHAR(50) NOT NULL), and JobDescription(VARCHAR(500) NOT NULL).

The Department table consists of a primary key which is DepartmentID (DECIMAL(12) NOT NULL), a column DepartmentName(VARCHAR(50) NOT NULL), and foreign key LocationID(DECIMAL(12) NOT NULL).

The SystemOwner table consists of primary key OwnerID (DECIMAL(12) NOT NULL), a foreign key EmployeeID (DECIMAL(12) NOT NULL), and a column JobDescription (VARCHAR(500) NOT NULL).

The Asset table consists of a primary key AssetID (DECIMAL(12) NOT NULL), two foreign keys OwnerID(DECIMAL(12) NOT NULL) and LocationID(DECIMAL(12) NOT NULL), and a column AssetName (VARCHAR(50) NOT NULL).

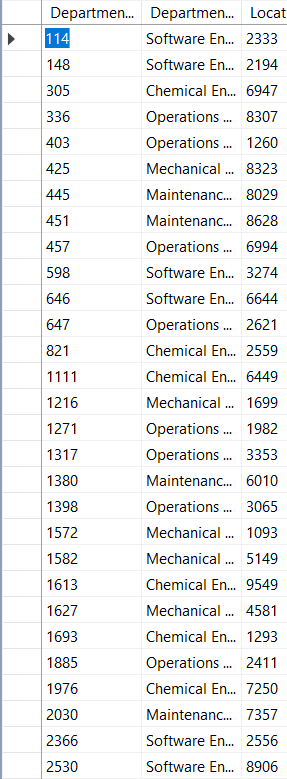
Finally, the table Location consists of a primary key LocationID(DECIMAL(12) NOT NULL), and rest being LocationName (VARCHAR(50) NOT NULL), Row(BIGINT NOT NULL), and Column (INT NOT NULL).

Index Identifications and Creation

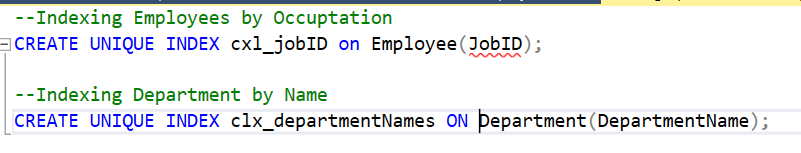
Table

Description automatically generated with medium confidence

To increase the speeds of querying from the table, the data should be indexed by the JobID to have it organized since jobs don’t have unique IDs for the same positions.



And for the department’s table, the data should be index by the DepartmentName, so that for instance, each department type is with their own groups. Therefore, the following indexes are created for the tables so that queries can be optimized and run faster.



Store Procedure Execution and Explanations

Graphical user interface, text, application, email

Description automatically generated

The *create\_email* procedure has two parameters, the email address which is a VARCHAR with a size limit of 500, and the EncryptionLevel which is an integer. The emailID will be randomly generated and the values are inserted in accordance to the parameters and declared variables. This is important because as soon as an employee is created or added to the database, an email will be generated through the procedure.

Text, email

Description automatically generated

The *create\_employee* email procedure has five parameters FirstName, which is a varchar with a maximum of 50 characters, LastName, which is a varchar with a maximum 50 characters, EmailAddress with a maximum of 500 characters, PhoneNumber, which is a BIGINT, and JobID which is a decimal(12). The procedure will generate a random employee ID. An email address is generate for the employee. A query is used to get the emailID from the email address. Then the employee is inserted into the email table.

Graphical user interface, text, application, email

Description automatically generated

The *create\_location* with three parameters LocationName, which is a varchar of a maximum of 50, Row, which is a BIGINT, and Column, which is an INT. A random location ID is generated. Then the location is added to the location table.

Graphical user interface, text, application

Description automatically generated

The *make\_department* has two parameters DepartmentName, which is a varchar with a maximum of 50 characters, and LocationID which is a decimal(12). A department ID is randomly generated and the department is inserted into the Department Table.

Graphical user interface, text, application, email

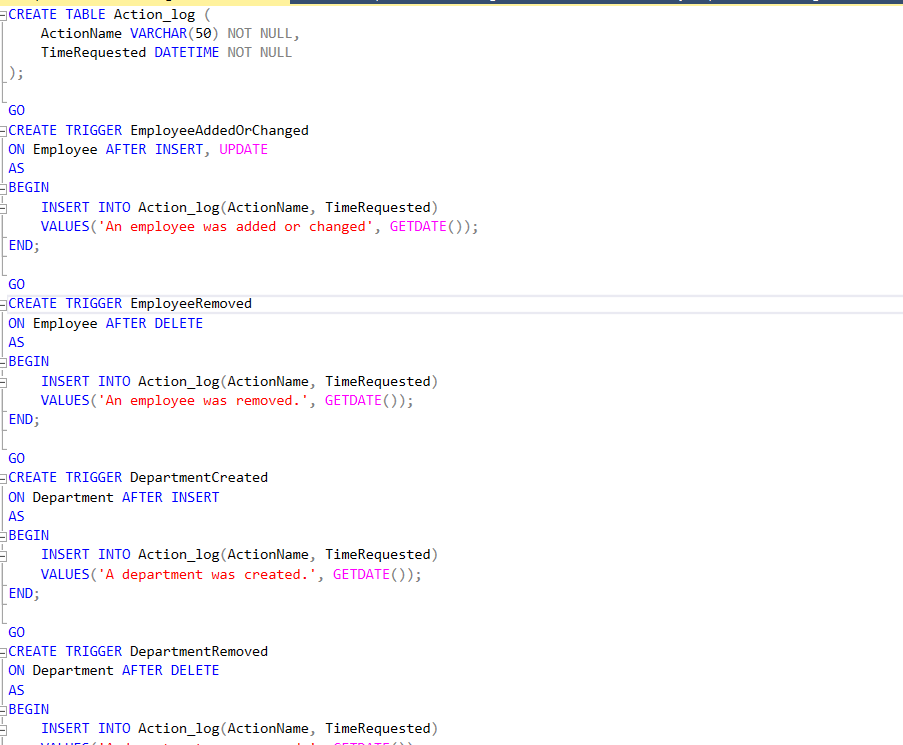
Description automatically generated

The *make\_into\_owner* has only one parameter which is an EmployeeID, which is a DECIMAL(12), and the sole purpose of this procedure is to make an employee into a system owner. A random system owner ID is generated and the employee is made into a system owner by adding their information to the System Owner tableGraphical user interface, text, application, email

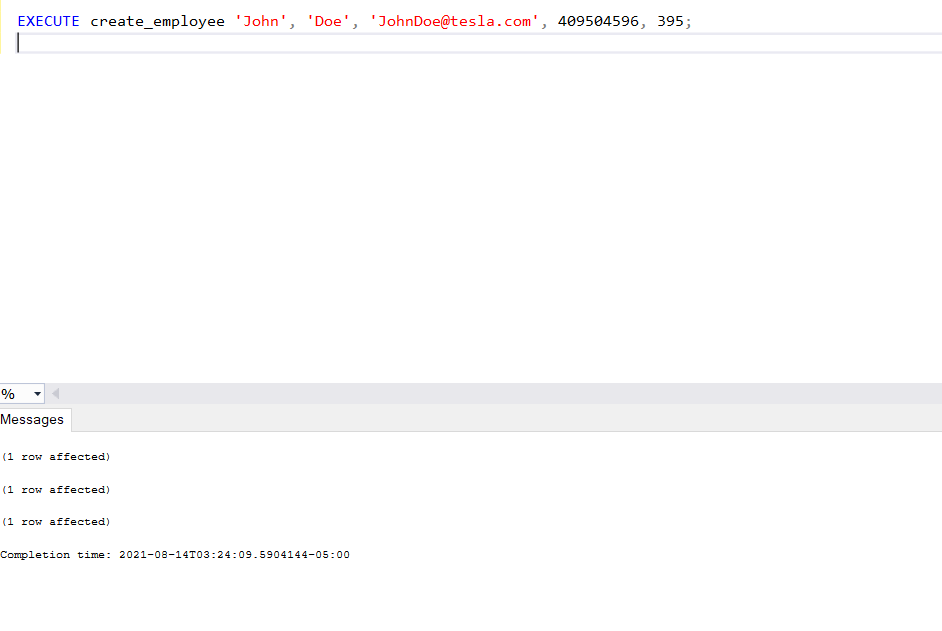
Description automatically generated

The *add\_asset­* table has three parameters, which is AssetName, which is a VARCHAR with a maximum of 50 characters, OwnerID and LocationID, which are both DECIMAL(12)’s. This table will generate a random asset ID and insert the asset into the table. This is important because there needs to be assets for the Tesla Giga Factory to manage.

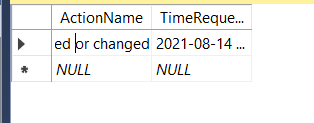
Trigger Creation and Use

o 

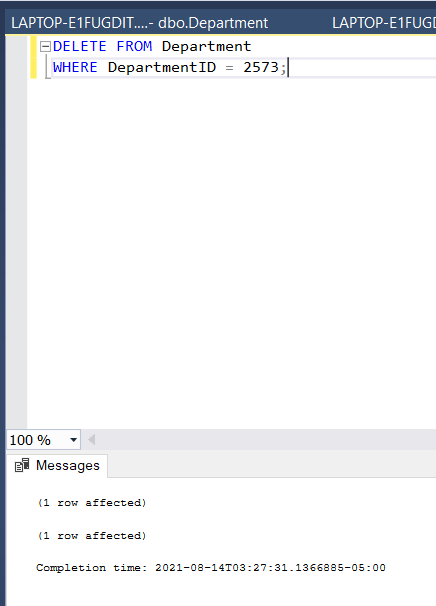
The triggers which are important for managing the activity in the action log are now established for very important information. Anything that is not too important can be disregarded from the logs. To see the full code, please look at triggers.sql. Hence to test the code, let’s add/remove some data.



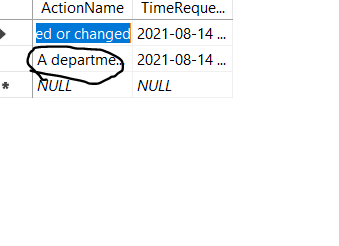
The first table was affected was of course, the email being created for the employee, then the employee being added to the employee table. But the third one? Well, it was added to the action log since that’s what the code was designed to do; to keep track of what is being added, removed, or modified.



Let’s say that it is time that a department has to go; that it is no longer needed. We will drop the department,



We can now see that it was removed at 8-14-2021 from the action log.



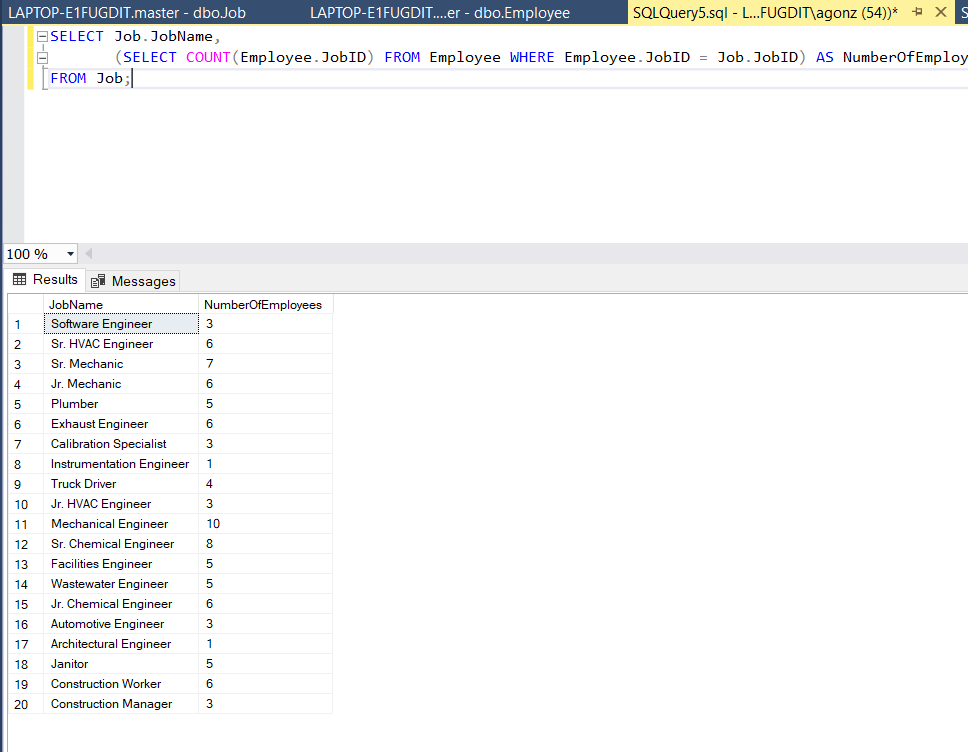
Question Identification and Explanations

There are many important questions that the Tesla Giga Factory must ask in order to operate at its full potential. The Tesla Giga Factory would like to obtain the statistics of career types at the Tesla Giga Factory to determine what kind of people they are missing and what kind of people they may not need anymore. Sometimes there may be too many plumbers than needed, or too many chemical engineers. The question that Tesla is trying to find out is, what kind of people do we need?

Another problem that Tesla may also be trying to solve is determine what kind of email encryption employees should have. Of course, regular employees will receive default encryption. However, important employees that are system owners needed to have maximum encryption since they hold very confidential data. Therefore, the question that Tesla has is, what kind of employees need to have their email encryption changed to maximum? Who are the System owners and how can we change their email encryptions to maximum encryption?

Question Executions and Explanations

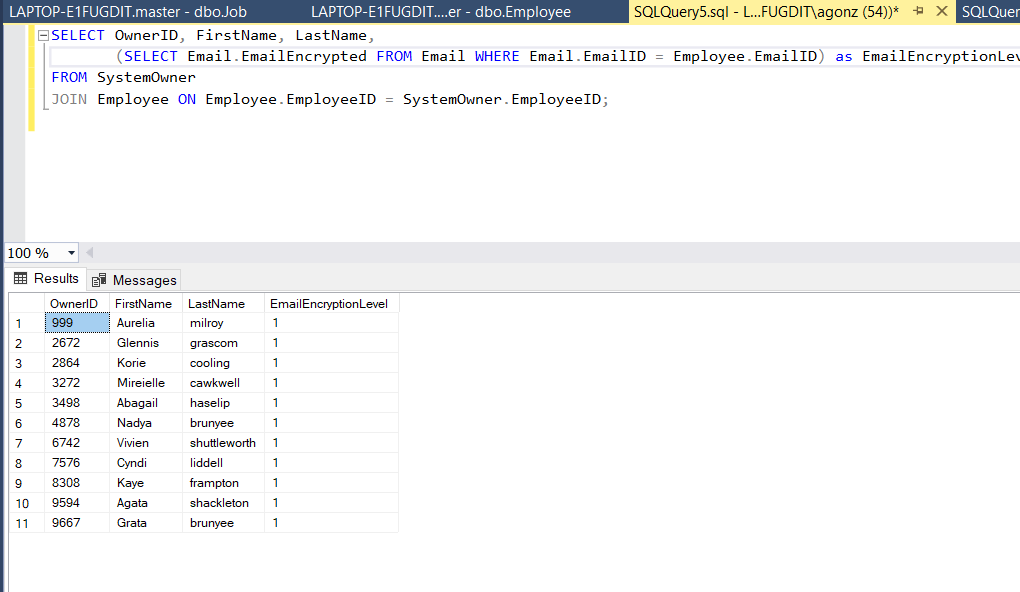
The first question that Tesla is trying to solve is what kind of employees do we need? What kind of employees are we missing? So, to solve this question, we execute the following query,



To help visualize the data, let’s transform this data into a chart.

From this chart, we can see that the plant lacks Instrumentation Engineers, Architectural Engineers, Construction managers, and Jr. HVAC Engineers. There is a prevalence of Mechanical Engineers and Sr. Chemical Engineers, therefore it would be best to hire more of the lacking fields mentioned thereof.

The next problem that Tesla is trying to solve is who are the system administrators who need their emails to become encrypted? But also check to see if they already have encrypted emails. We can solve this with the following query.



Therefore, these are the following System Owners, who need their email encryption level to be changed to 2 (which means highly encrypted).

Summary and Reflection

For this project, a database was designed and implemented for my employer Tesla Motors. We are currently building a Gigafactory in Austin, TX and as mentioned, it would be important to have a database which can manage assets, manage departments, manage employees, and managing transactions or logging. The Conceptual Entity Diagram was created to help visualize how the database will work. In addition, many revisions had to made for it to make sense, especially since this is used for my employer at Tesla Motors. Once the design was finalized, a physical DBMS was created, then converted into SQL code. Now that the database tables were created, it was time to fill the tables with data. Of course, it was a bit tedious modifying the code since the IDs were generated randomly and creating procedures which do very similar functions but for different tables and data sets. However, the database is much easier to track and use. Although many improvements could be made in order to facilitate the easy of use of the data base. Queries are very important for this kind of data because it can help solve real world problems at the Tesla Factory as mentioned above in the questions earlier. All the scripts will be contained in the scripts folder which the assignment will be turned in with. And photos will be in the main source folder.