Tech ABC Corp - HR Database

[Lawrence Mugwena & 2023-07-16]

Business Scenario

Business requirement

Tech ABC Corp saw explosive growth with a sudden appearance onto the gaming scene with their new AI-powered video game console. As a result, they have gone from a small 10 person operation to 200 employees and 5 locations in under a year. HR is having trouble keeping up with the growth, since they are still maintaining employee information in a spreadsheet. While that worked for ten employees, it has becoming increasingly cumbersome to manage as the company expands.

As such, the HR department has tasked you, as the new data architect, to design and build a database capable of managing their employee information.

Dataset

The HR dataset you will be working with is an Excel workbook which consists of 206 records, with eleven columns. The data is in human readable format, and has not been normalized at all. The data lists the names of employees at Tech ABC Corp as well as information such as job title, department, manager's name, hire date, start date, end date, work location, and salary.

IT Department Best Practices

The IT Department has certain Best Practices policies for databases you should follow, as detailed in the Best Practices document.

Step 1: Data Architecture Foundations

Hi,

Welcome to Tech ABC Corp. We are excited to have some new talent onboard. As you may already know, Tech ABC Corp has recently experienced a lot of growth. Our AI powered video game console WOPR has been hugely successful and as a result, our company has grown from 10 employees to 200 in only 6 months (and we are projecting a 20% growth a year for the next 5 years). We have also grown from our Dallas, Texas office, to 4 other locations nationwide: New York City, NY, San Francisco, CA, Minneapolis, MN, and Nashville, TN.

While this growth is great, it is really starting to put a strain on our record keeping in HR. We currently maintain all employee information on a shared spreadsheet. When HR consisted of only myself, managing everyone on an Excel spreadsheet was simple, but now that it is a shared document I am having serious reservations about data integrity and data security. If the wrong person got their hands on the HR file, they would see the salaries of every employee in the company, all the way up to the president.

After speaking with Jacob Lauber, the manager of IT, he suggested I put in a request to have my HR Excel file converted into a database. He suggested I reach out to you as I am told you have experience in designing and building databases. When you are building this, please keep in mind that I want any employee with a domain login to be have read only access the database. I just don't want them having access to salary information. That needs to be restricted to HR and management level employees only. Management and HR employees should also be the only ones with write access. By our current estimates, 90% of users will be read only.

I also want to make sure you know that am looking to turn my spreadsheet into a live database, one I can input and edit information into. I am not really concerned with reporting capabilities at the moment. Since we are working with employee data we are required by federal regulations to maintain this data for at least 7 years; additionally, since this is considered business critical data, we need to make sure it gets backed up properly.

As a final consideration. We would like to be able to connect with the payroll department's system in the future. They maintain employee attendance and paid time off information. It would be nice if the two systems could interface in the future

I am looking forward to working with you and seeing what kind of database you design for us.

Thanks,
Sarah Collins
Head of HR

Data Architect Business Requirement

- ## Purpose of the new database:

  What is the business partner requesting

  Answer: Our business partner experienced rapid growth with their AI-powered video game console, expanding their team from 10 to 200 employees in just one year and establishing operations in four new locations. However, this growth has posed challenges for the HR department in managing the increasing number of employees. To address this, they have requested the design and construction of a robust employee information management database. This database will centralize employee data, facilitate secure access and updates, and include modules for managing benefits, payroll, performance evaluations, and training records. The implementation of this comprehensive database will enable smoother operations, support scalability, and improve HR efficiency.

  ## Describe current data management solution:

  What is the current method data storage/management

  Answer: Currently all the employee information is maintained on a shared spreadsheet.

- ## Describe current data available:

  What data does the business currently have available

  Answer: The business currently manages its data through an Excel workbook consisting of 206 records and 11 columns. However, the data is in denormalized form, making it challenging to efficiently manage and analyze. To address this, it is recommended to design and implement a normalized database system. This will involve creating separate tables for entities such as employees, job titles, departments, and locations, and establishing relationships between them. The transition to a database system will improve data organization, facilitate easier management and analysis, and enable advanced querying and reporting capabilities. Furthermore, it will support scalability and integration with other systems.

Data Architect Business Requirement

- **Additional data requests:**

  Does the user have future data requests
  Answer: In the future, the business plans to integrate employee attendance and paid time off information with the existing database by incorporating the payroll department's system. This integration aims to streamline payroll processes and ensure accurate tracking of attendance and time off. By synchronizing data between the systems, HR and payroll personnel will have access to up-to-date information, enabling efficient payroll processing and leave management. Data security and access controls will be implemented, and thorough testing will be conducted to ensure a smooth and reliable integrated system. Overall, the integration will enhance data accuracy, reduce administrative burdens, and improve HR and payroll operations.

- **Who will own/manage data**

  What department will own / manage the data in the database
  Answer: HR department with some management level employees

- **Who will have access to database**

  List user types that will have access; also list any restrictions to access.

  Answer: To ensure data security, employees will be granted read-only access to the database, while the HR team will have full access. This measure prevents employees from accessing sensitive salary information. User authentication and access control mechanisms will be implemented to enforce these restrictions. Regular monitoring and audit trails will be put in place to maintain data integrity and detect any unauthorized access attempts. This approach ensures a balance between data security and operational needs within the organization.

Data Architect Business Requirement

- **Estimated size of database**

  List the size of the database in terms of numbers of rows. Business users often understand row or column size instead of GBs or MBs

  Answer: Number of rows 206 and number of columns 11

- **Estimated annual growth**

  List any expected growth to the data

  Answer: 20% of expected growth each year for next 5 years

- **Is any of the data sensitive/restricted**

  List any data that may be sensitive or restricted from particular users

  Answer: Yes, salary data is vey much sensitive and it is to be restricted from employees otherwise the would see the salaries of every employee in the company, all the way up to the president

Data Architect Technical Requirement

- ## Justification for the new database

Provide at least two justifications for building a database

Answer: To accommodate the projected growth of 20% per year for the next five years, it is essential to migrate data from spreadsheets to a more manageable database. This transition will provide scalability, flexibility, and better organization, enabling efficient data management and analysis as the company expands. Additionally, moving to a database will address issues related to data integrity and security. By implementing proper data normalization, validation rules, and security measures such as user authentication and access controls, the company can maintain data integrity, enhance data security, and protect sensitive information. Overall, the shift to a database system will support the company's growth objectives and ensure reliable and secure data management.

- ## Database objects

List the database objects (tables, views, special procedures) that will be created for the database. Hint - you may want to circle back to this answer after completing the logical ERD in step 2.

Answer:
1) Education Table
2) Employee Table
3) Job Table
4) Department Table
5) Location Table
6) State Table
7) City Table
8) Address Table
9) Salary Table
10) Employee Detail

- ## Data ingestion

Select a data ingestion method (ETL, Direct feed, API) based on the information provided

Answer: ETL is the appropriate ingestion technique here as we are provided with flat files (excel file). Through this, we can create an automated ETL process which will ingest data into the database from the flat files.

Data Architect Technical Requirement

- **Data governance (Ownership and User access)**

  Ownership: who will own and maintain the data
  Answer: HR

  User Access: who will and will not have access to the data

  Answer: Employees will have read only access to the database, no access to the salary table and HR while have access full to salary table.
- **Scalability**

  Should replication or sharding be used to ensure scalability based on user needs

  Answer: Replication should be used for our case. As the company grows, the number of employees joining the company will increase, and hence the amount of employees accessing the database will increase. This will increase the load on database and will degrade the performance. So we need to replicate the database in order to distribute the load on different databases

- **Storage & retention**

  Storage (disk or in-memory): check IT best practices document

  Retention: how long does the data have to be kept for?

- **Backup**

Data Architect Technical Requirement

- **Flexibility**

  Describe measures taken to ensure future data integration if needed

  Answer: 1) Presence of some similar entities to ensure the fully functional and meaningful integration. 2) Maintain similar frameworks for the DBMS used in different departments. This will ensure hassle free integration if needed.

- **Storage & retention**

  Storage (disk or in-memory): check IT best practices document

  Answer: Disk-based storage is recommended as it fulfills our requirements without the need for higher-level computations. In-memory storage types, while providing faster performance for advanced analytics, are generally expensive. Thus, for our purposes, utilizing disk-based storage is a cost-effective choice.

  Retention: how long does the data have to be kept for?
  Answer: As per the federal regulations, it is to be kept for 7 years.

Data Architect Technical Requirement

- **Backup**

  IT Best Practices document lists Backup schedule requirements

  Answer: Considering the need for a live database where the Head of HR frequently inputs and edits information, it is essential to opt for the Critical Backup option. To ensure data integrity and minimize potential loss, a schedule should be set for full backups once per week, along with daily incremental backups. This approach allows for frequent capturing of changes made to the database, providing necessary backup coverage and reducing the risk of data loss. By implementing this backup strategy, the company can maintain an up-to-date and reliable database, ensuring the availability and integrity of HR information.

Step 2

Relational Database Design

Step 2: Relational Database Design

This step is where you will go through the process of designing a new database for Tech ABC Corp's HR department. Using the dataset provided, along with the requirements gathered in step one, you are going to develop a relational database set to the 3NF.

Using Lucidchart, you will create 3 entity relationship diagrams (ERDs) to show how you developed the final design for your data.

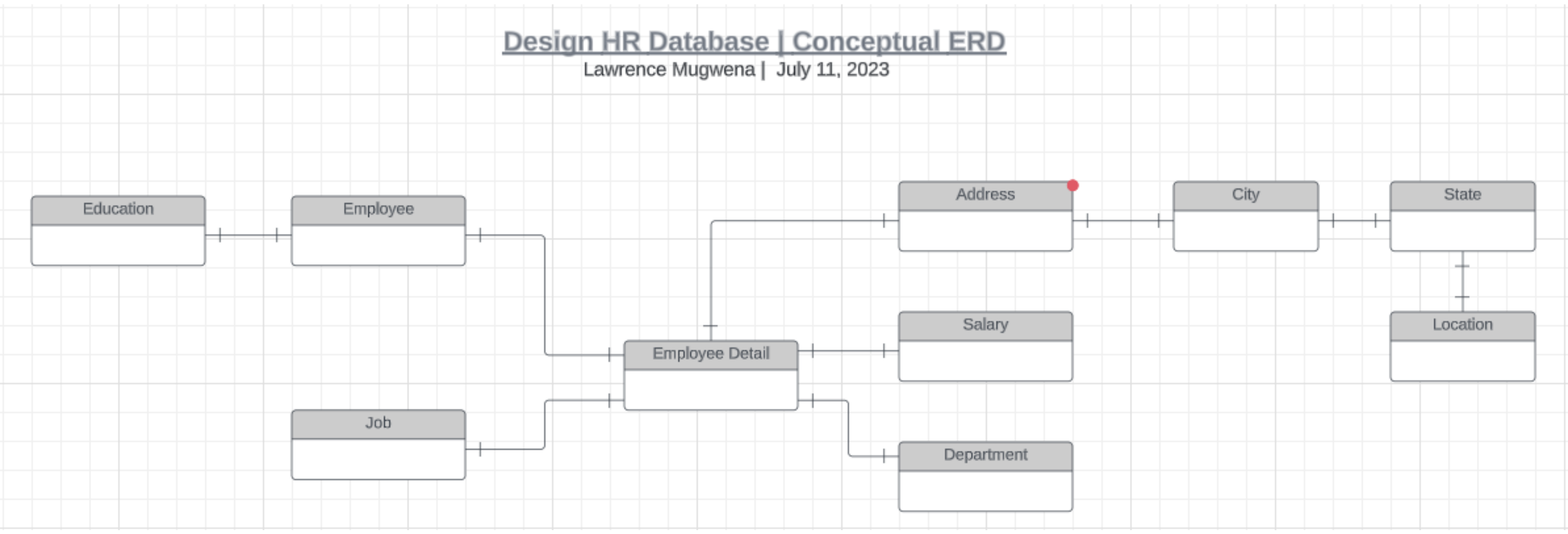
You will submit a screenshot for each of the 3 ERDs you create. You will find detailed instructions for developing each of the ERDs over the next several pages.

ERD

- **Conceptual**

This is the most general level of data modeling. At the conceptual level, you should be thinking about creating entities that represent business objects for the database. Think broadly here. Attributes (or column names) are not required at this point, but relationship lines are required (although Crow's foot notation is not needed at this level). Create at least three entities for this model; thinking about the 3NF will aid you in deciding the type of entities to create.

Use Lucidchart's built-in template for DBMS ER Diagram UML.

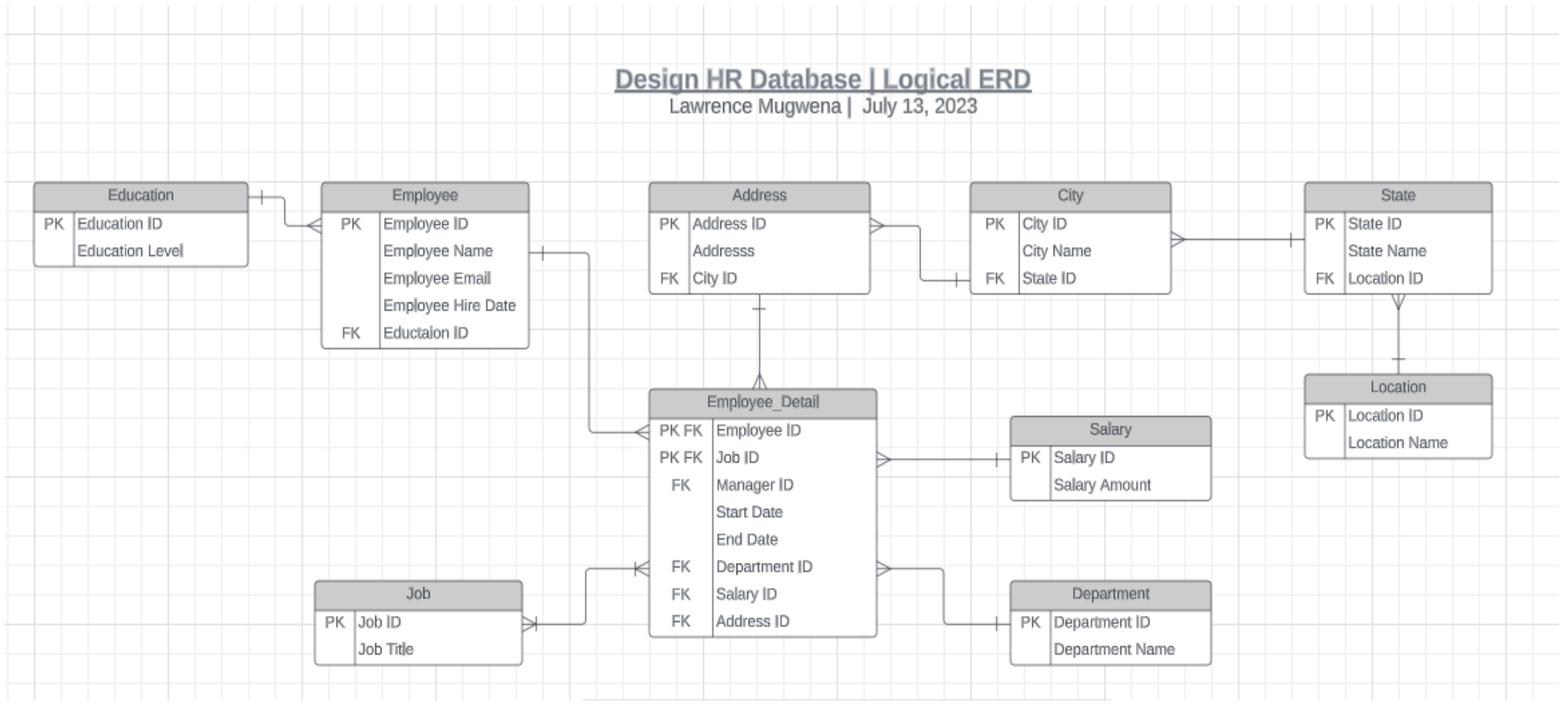


ERD

- **Logical**

The logical model is the next level of refinement from the conceptual ERD. At this point, you should have normalized the data to the 3NF. Attributes should also be listed now in the ERD. You can still use human-friendly entity and attribute names in the logical model, and while relationship lines are required, Crow's foot notation is still not needed at this point.

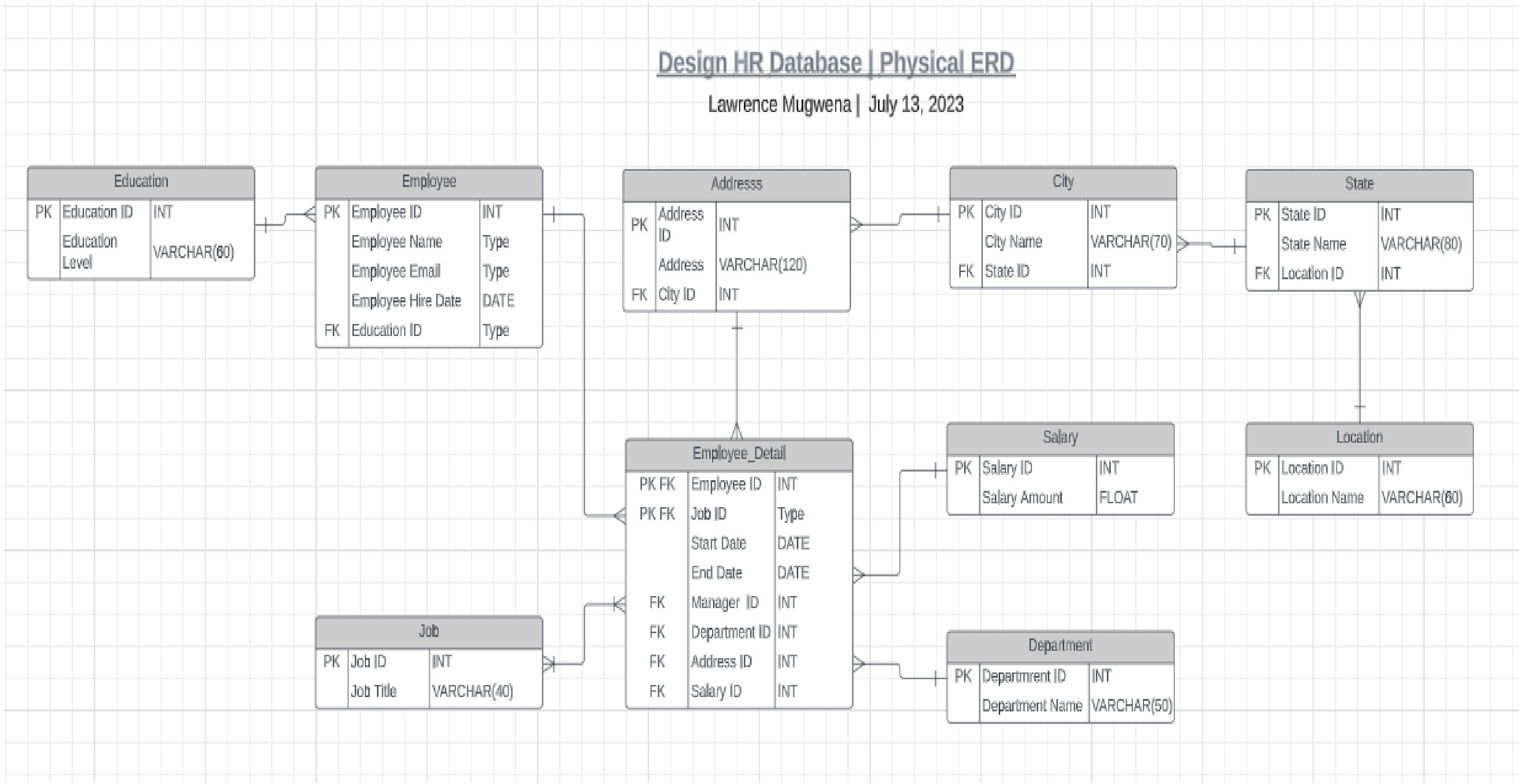
Use Lucidchart’s built-in template for DBMS ER Diagram UML.



ERD

- **Physical**

The physical model is what will be built in the database. Each entity should represent a database table, complete with column names and data types. Primary keys and foreign keys should also be represented here. Primary keys should be in bold type with the (PK) designation following the field name. Foreign keys should be in normal type face, but have the designation (FK) after the column name. Finally, in the physical model, Crow's foot notation is important.



Step 3

Create A Physical Database

Step 3: Create A Physical Database

In this step, you will be turning your database model into a physical database.

You will:

- Create the database using SQL DDL commands
- Load the data into your database, utilizing flat file ETL
- Answer a series of questions using CRUD SQL commands to demonstrate your database was created and populated correctly

Submission
For this step, you will need to submit SQL files containing all DDL SQL scripts used to create the database.

You will also have to submit screenshots showing CRUD commands, along with results for each of the questions found in the starter template.

Hints
Your DDL script will be graded by running the code you submit. Please ensure your SQL code runs properly!

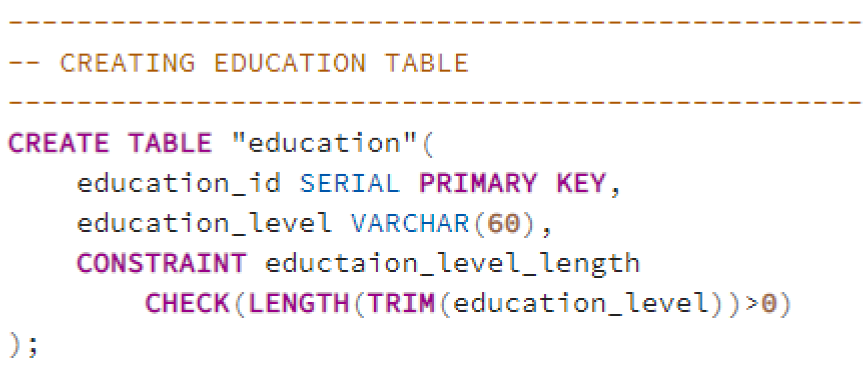
Foreign keys cannot be created on tables that do not exist yet, so it may be easier to create all tables in the database, then to go back and run modify statements on the tables to create foreign key constraints.

After running CRUD commands like update, insert, or delete, run a `SELECT*` command on the affected table, so the reviewer can see the results of the command.

DDL

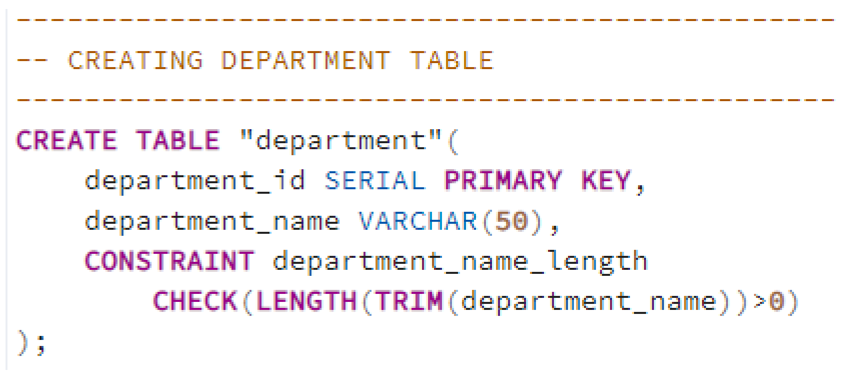
Create a DDL SQL script capable of building the database you designed in Step 2

Education Table



```sql
--------------------------------------------------
-- CREATING EDUCATION TABLE
--------------------------------------------------
CREATE TABLE "education"(
    education_id SERIAL PRIMARY KEY,
    education_level VARCHAR(60),
    CONSTRAINT eductaion_level_length
        CHECK(LENGTH(TRIM(education_level))>0)
);
```

Department Table

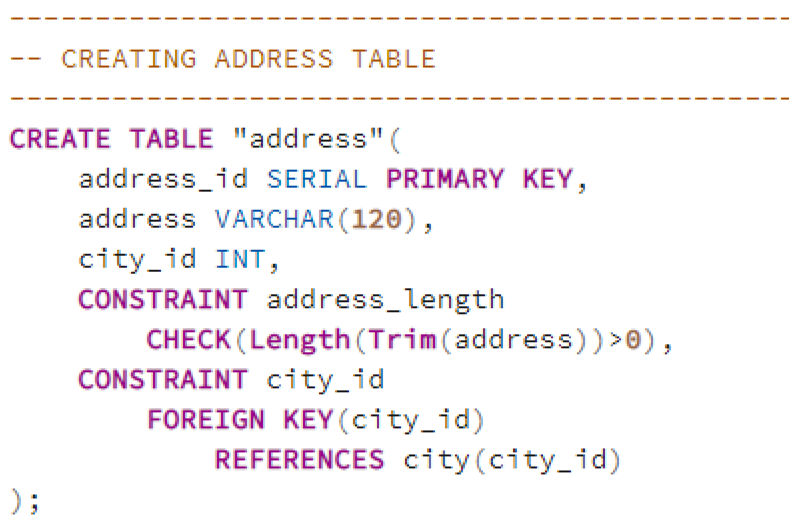


```sql
------------------------------------------------
-- CREATING DEPARTMENT TABLE
------------------------------------------------
CREATE TABLE "department"(
    department_id SERIAL PRIMARY KEY,
    department_name VARCHAR(50),
    CONSTRAINT department_name_length
        CHECK(LENGTH(TRIM(department_name))>0)
);
```

DDL

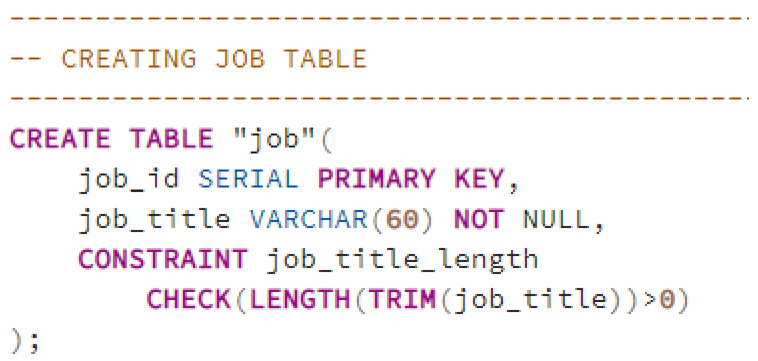
Create a DDL SQL script capable of building the database you designed in Step 2

Address Table



```sql
---------------------------------------------
-- CREATING ADDRESS TABLE
---------------------------------------------
CREATE TABLE "address"(
    address_id SERIAL PRIMARY KEY,
    address VARCHAR(120),
    city_id INT,
    CONSTRAINT address_length
        CHECK(Length(Trim(address))>0),
    CONSTRAINT city_id
        FOREIGN KEY(city_id)
            REFERENCES city(city_id)
);
```

Job Table

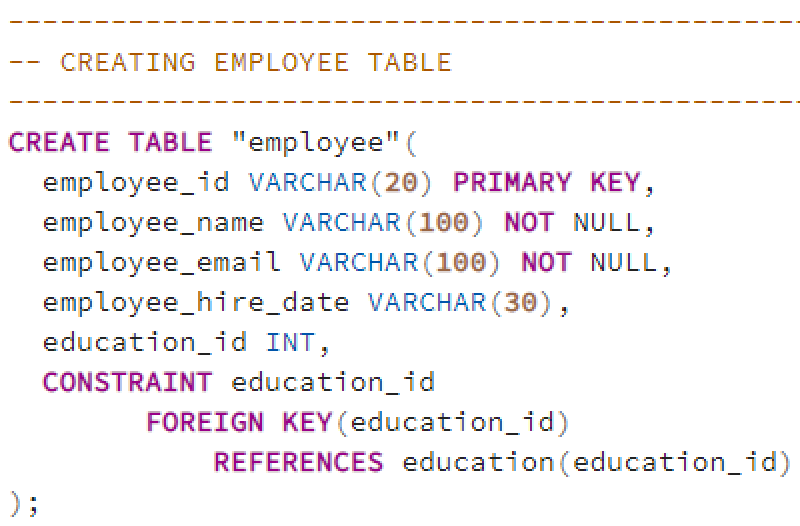


```sql
-------------------------------------------
-- CREATING JOB TABLE
-------------------------------------------
CREATE TABLE "job"(
    job_id SERIAL PRIMARY KEY,
    job_title VARCHAR(60) NOT NULL,
    CONSTRAINT job_title_length
        CHECK(LENGTH(TRIM(job_title))>0)
);
```

DDL

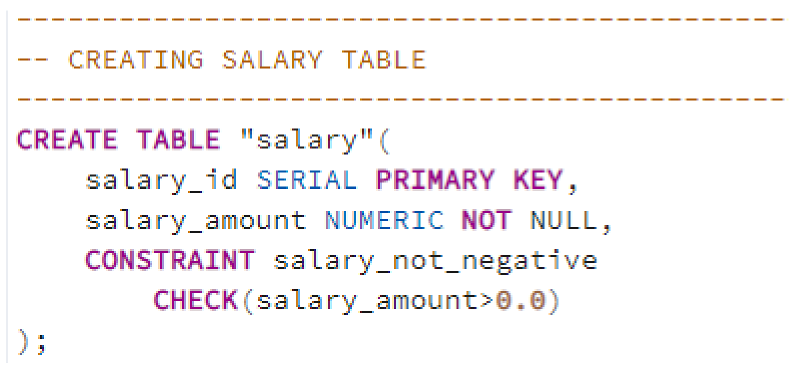
Create a DDL SQL script capable of building the database you designed in Step 2

Employee Table



```sql
---------------------------------------------------
-- CREATING EMPLOYEE TABLE
---------------------------------------------------
CREATE TABLE "employee"(
  employee_id VARCHAR(20) PRIMARY KEY,
  employee_name VARCHAR(100) NOT NULL,
  employee_email VARCHAR(100) NOT NULL,
  employee_hire_date VARCHAR(30),
  education_id INT,
  CONSTRAINT education_id
        FOREIGN KEY(education_id)
            REFERENCES education(education_id)
);
```

Salary Table

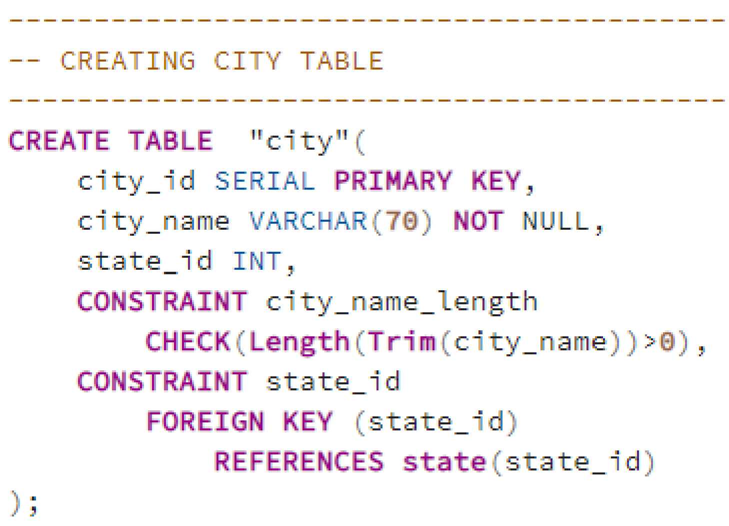


```sql
---------------------------------------------
-- CREATING SALARY TABLE
---------------------------------------------
CREATE TABLE "salary"(
    salary_id SERIAL PRIMARY KEY,
    salary_amount NUMERIC NOT NULL,
    CONSTRAINT salary_not_negative
        CHECK(salary_amount>0.0)
);
```

DDL

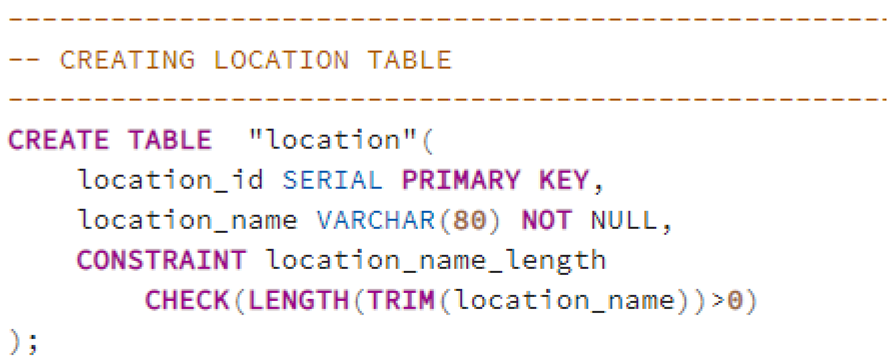
Create a DDL SQL script capable of building the database you designed in Step 2

City Table



```
----------------------------------------------
-- CREATING CITY TABLE
----------------------------------------------
CREATE TABLE  "city"(
    city_id SERIAL PRIMARY KEY,
    city_name VARCHAR(70) NOT NULL,
    state_id INT,
    CONSTRAINT city_name_length
        CHECK(Length(Trim(city_name))>0),
    CONSTRAINT state_id
        FOREIGN KEY (state_id)
            REFERENCES state(state_id)
);
```

Location Table

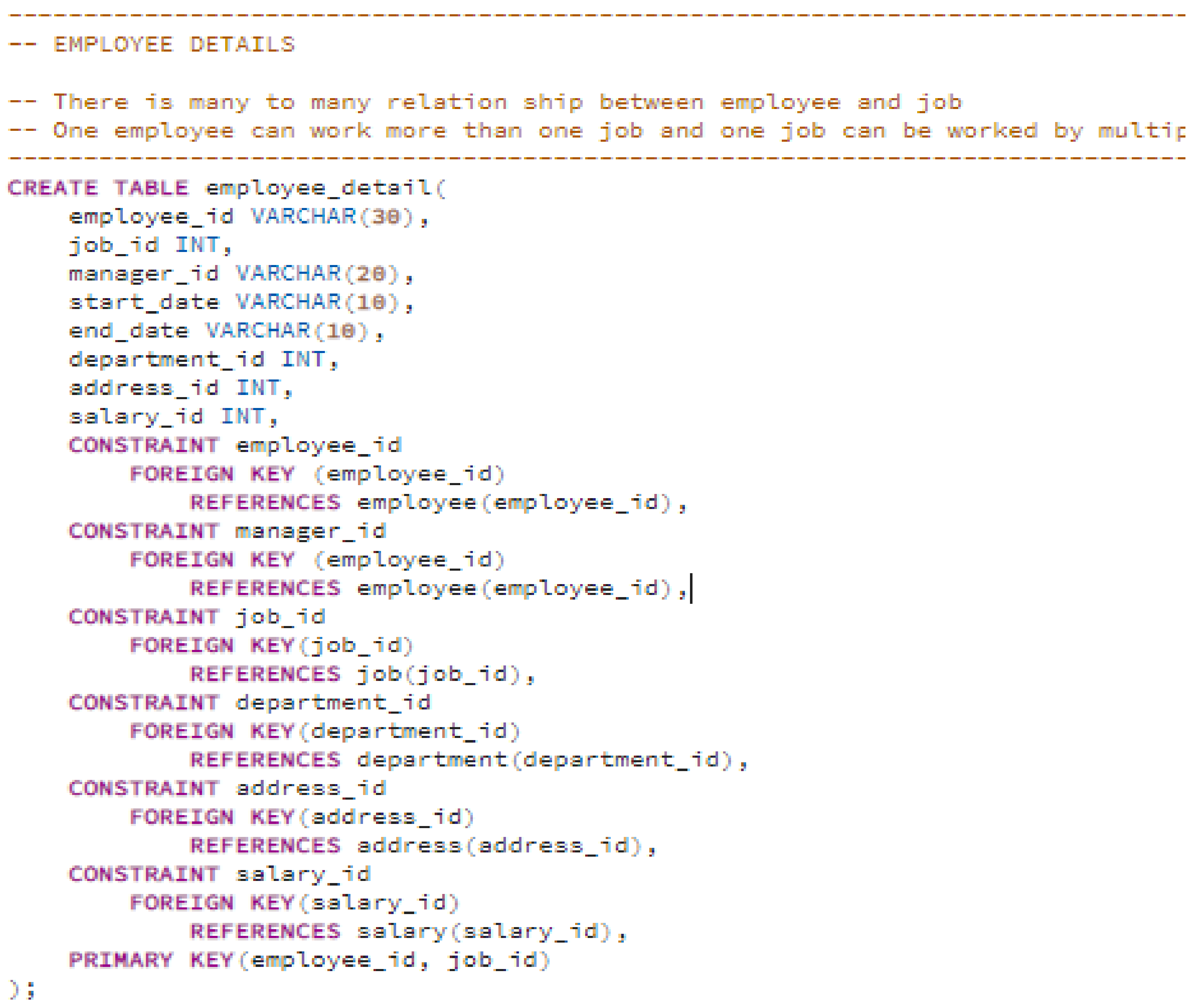


```
----------------------------------------------------
-- CREATING LOCATION TABLE
----------------------------------------------------
CREATE TABLE  "location"(
    location_id SERIAL PRIMARY KEY,
    location_name VARCHAR(80) NOT NULL,
    CONSTRAINT location_name_length
        CHECK(LENGTH(TRIM(location_name))>0)
);
```

DDL

Create a DDL SQL script capable of building the database you designed in Step 2

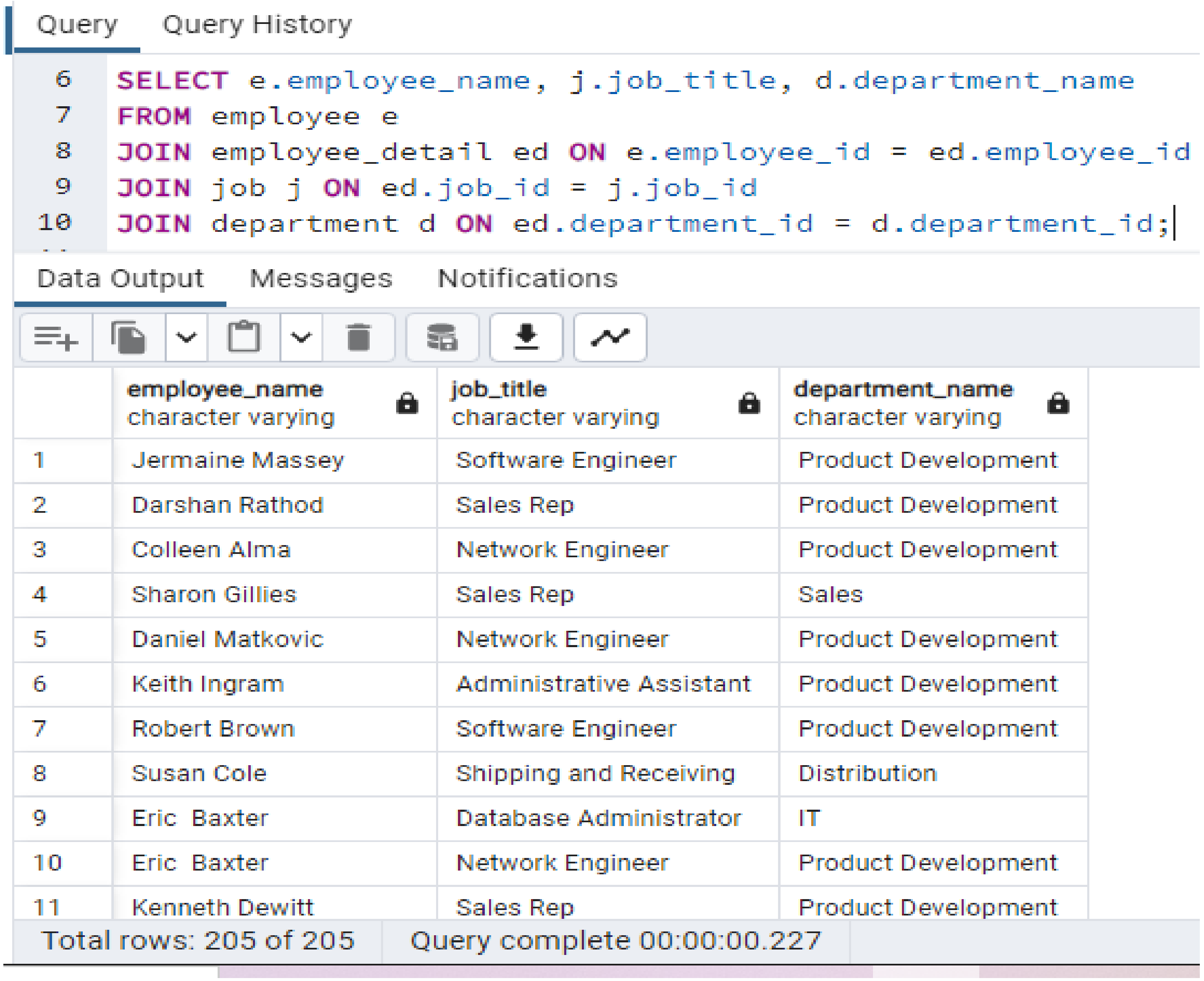
Employee Detail Table



```
------------------------------------------------------------------------------
-- EMPLOYEE DETAILS

-- There is many to many relation ship between employee and job
-- One employee can work more than one job and one job can be worked by multip
------------------------------------------------------------------------------
CREATE TABLE employee_detail(
    employee_id VARCHAR(30),
    job_id INT,
    manager_id VARCHAR(20),
    start_date VARCHAR(10),
    end_date VARCHAR(10),
    department_id INT,
    address_id INT,
    salary_id INT,
    CONSTRAINT employee_id
        FOREIGN KEY (employee_id)
            REFERENCES employee(employee_id),
    CONSTRAINT manager_id
        FOREIGN KEY (employee_id)
            REFERENCES employee(employee_id),
    CONSTRAINT job_id
        FOREIGN KEY(job_id)
            REFERENCES job(job_id),
    CONSTRAINT department_id
        FOREIGN KEY(department_id)
            REFERENCES department(department_id),
    CONSTRAINT address_id
        FOREIGN KEY(address_id)
            REFERENCES address(address_id),
    CONSTRAINT salary_id
        FOREIGN KEY(salary_id)
            REFERENCES salary(salary_id),
    PRIMARY KEY(employee_id, job_id)
);
```

CRUD

- **Question 1: Return a list of employees with Job Titles and Department Names**



Query | Query History

```sql
SELECT e.employee_name, j.job_title, d.department_name
FROM employee e
JOIN employee_detail ed ON e.employee_id = ed.employee_id
JOIN job j ON ed.job_id = j.job_id
JOIN department d ON ed.department_id = d.department_id;
```

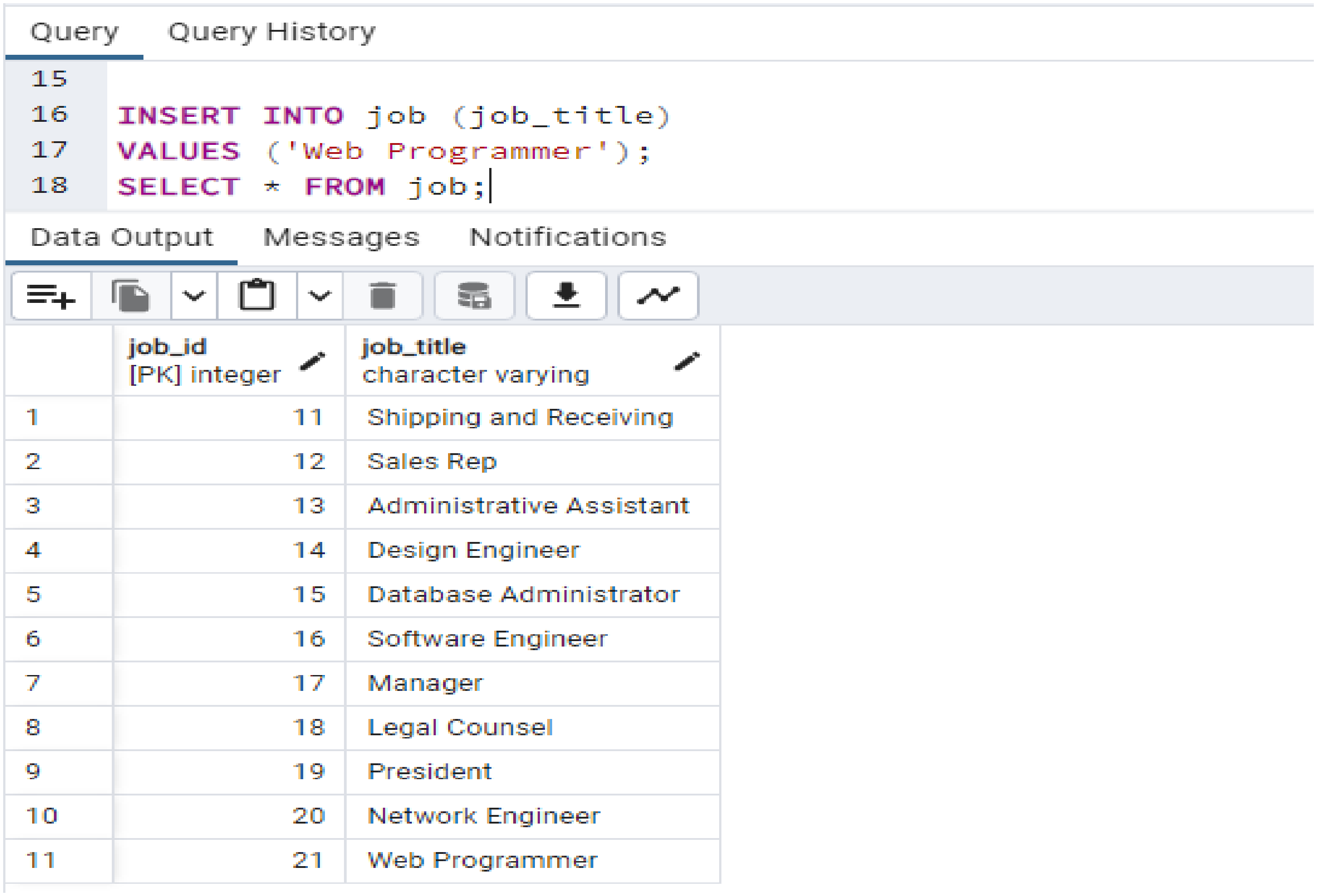
Data Output | Messages | Notifications

	employee_name character varying	job_title character varying	department_name character varying
1	Jermaine Massey	Software Engineer	Product Development
2	Darshan Rathod	Sales Rep	Product Development
3	Colleen Alma	Network Engineer	Product Development
4	Sharon Gillies	Sales Rep	Sales
5	Daniel Matkovic	Network Engineer	Product Development
6	Keith Ingram	Administrative Assistant	Product Development
7	Robert Brown	Software Engineer	Product Development
8	Susan Cole	Shipping and Receiving	Distribution
9	Eric Baxter	Database Administrator	IT
10	Eric Baxter	Network Engineer	Product Development
11	Kenneth Dewitt	Sales Rep	Product Development

Total rows: 205 of 205   Query complete 00:00:00.227

CRUD

- **Question 2: Insert Web Programmer as a new job title**

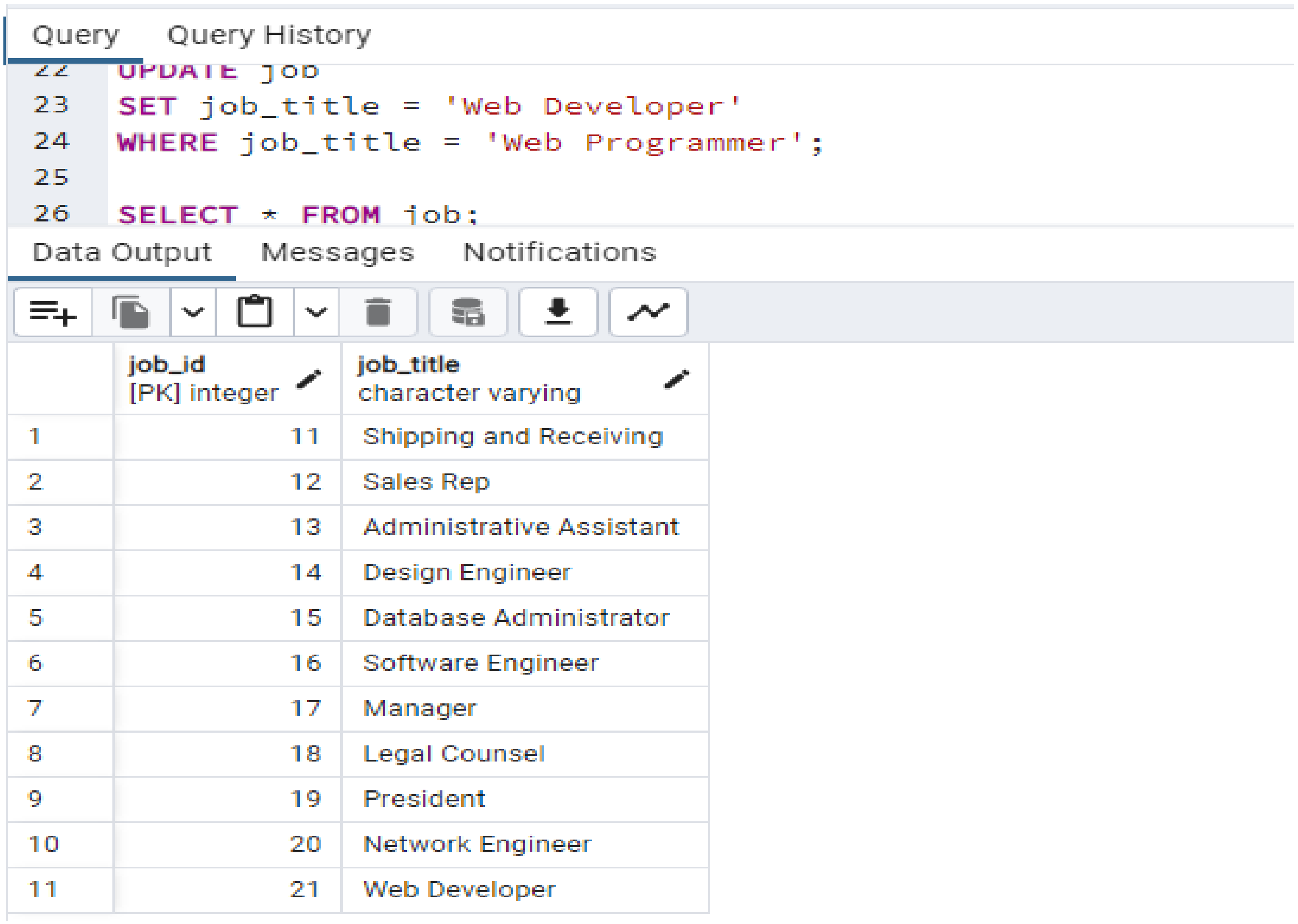


```sql

INSERT INTO job (job_title)
VALUES ('Web Programmer');
SELECT * FROM job;
```

	job_id [PK] integer	job_title character varying
1	11	Shipping and Receiving
2	12	Sales Rep
3	13	Administrative Assistant
4	14	Design Engineer
5	15	Database Administrator
6	16	Software Engineer
7	17	Manager
8	18	Legal Counsel
9	19	President
10	20	Network Engineer
11	21	Web Programmer

CRUD

- **Question 3: Correct the job title from web programmer to web developer**



Query | Query History

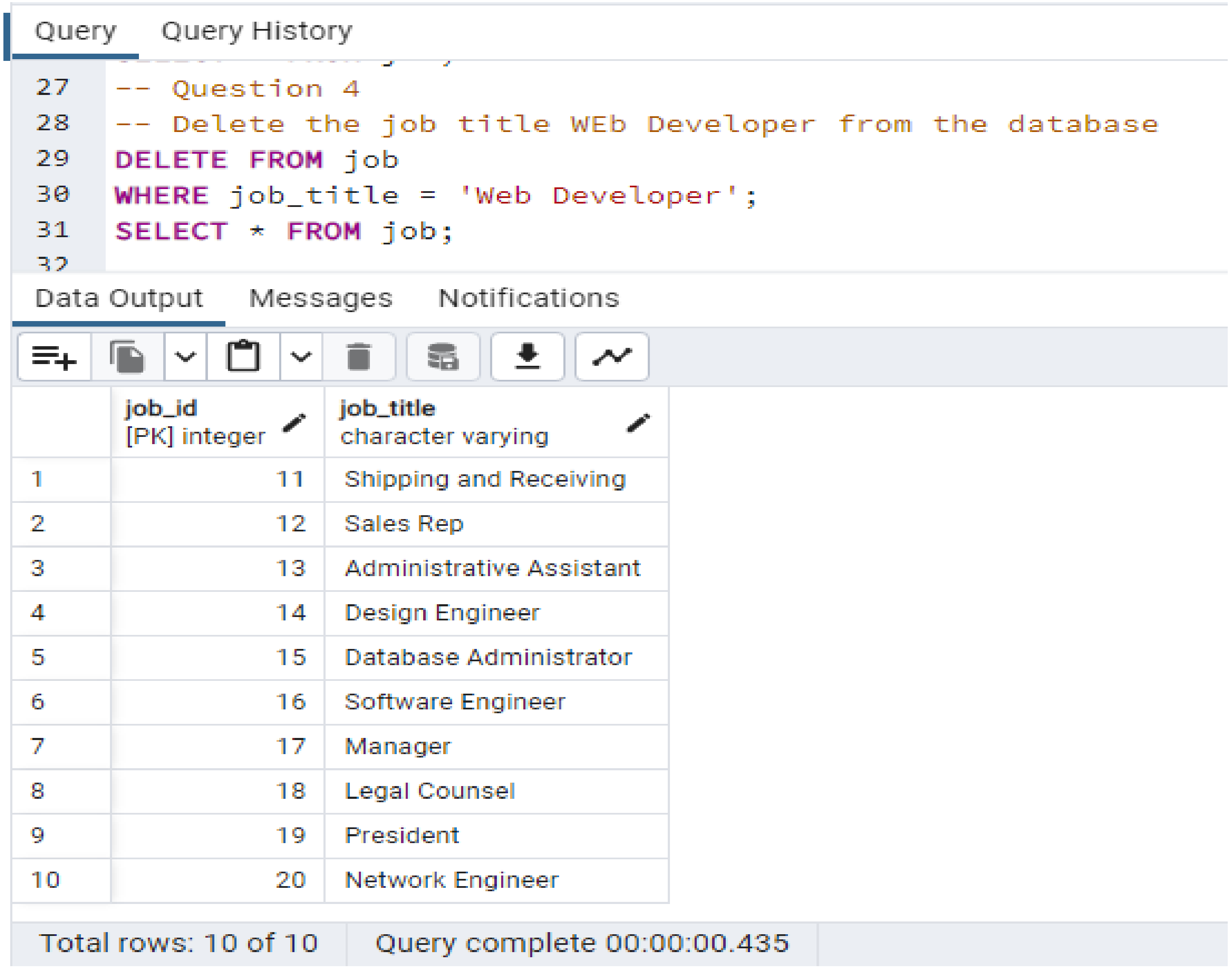
```
UPDATE job
SET job_title = 'Web Developer'
WHERE job_title = 'Web Programmer';

SELECT * FROM job;
```

Data Output | Messages | Notifications

	job_id [PK] integer	job_title character varying
1	11	Shipping and Receiving
2	12	Sales Rep
3	13	Administrative Assistant
4	14	Design Engineer
5	15	Database Administrator
6	16	Software Engineer
7	17	Manager
8	18	Legal Counsel
9	19	President
10	20	Network Engineer
11	21	Web Developer

CRUD

- **Question 4: Delete the job title Web Developer from the database**



Query | Query History

```
-- Question 4
-- Delete the job title WEb Developer from the database
DELETE FROM job
WHERE job_title = 'Web Developer';
SELECT * FROM job;
```

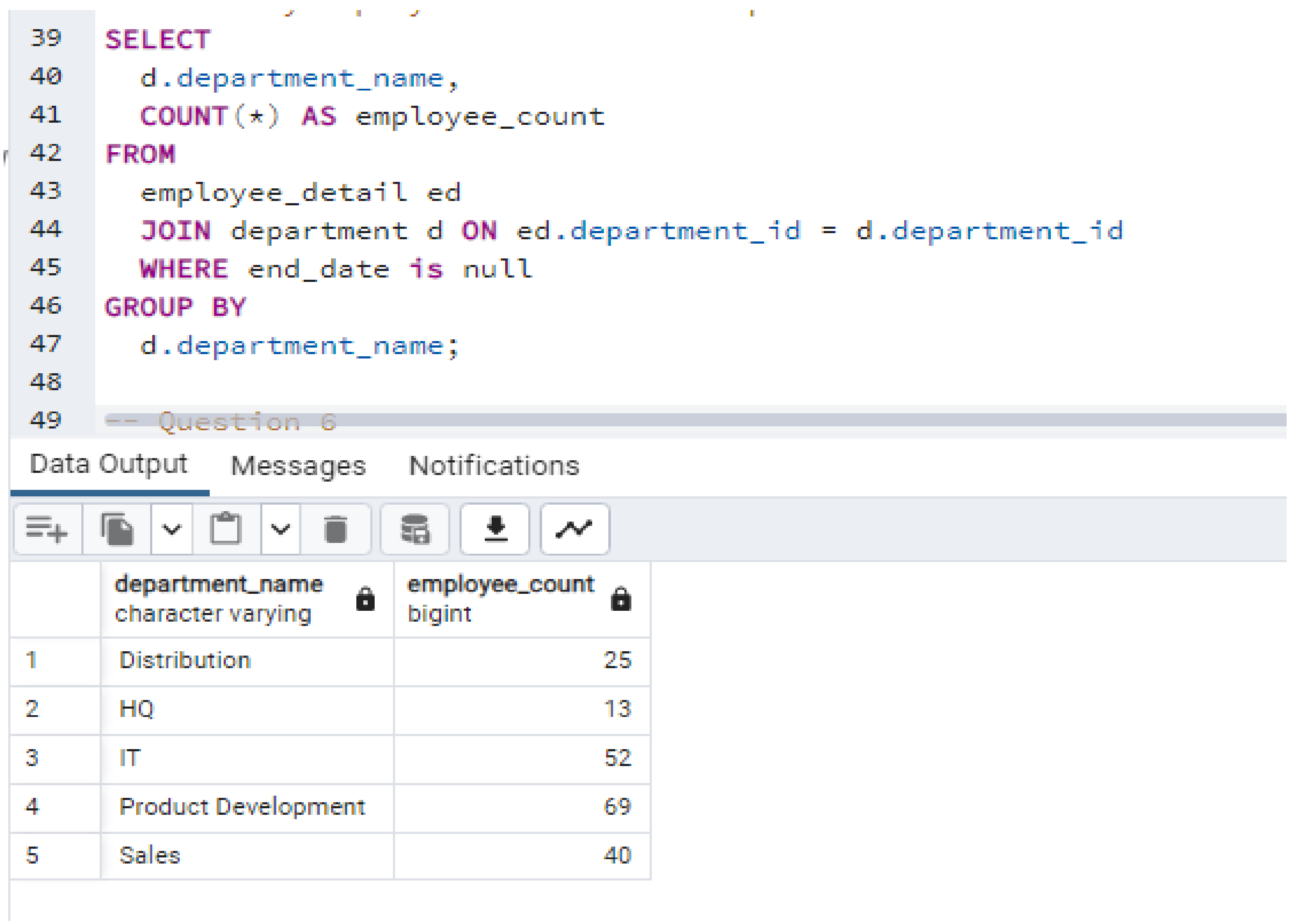
Data Output | Messages | Notifications

	job_id [PK] integer	job_title character varying
1	11	Shipping and Receiving
2	12	Sales Rep
3	13	Administrative Assistant
4	14	Design Engineer
5	15	Database Administrator
6	16	Software Engineer
7	17	Manager
8	18	Legal Counsel
9	19	President
10	20	Network Engineer

Total rows: 10 of 10 Query complete 00:00:00.435

CRUD

- **Question 5: How many employees are in each department?**



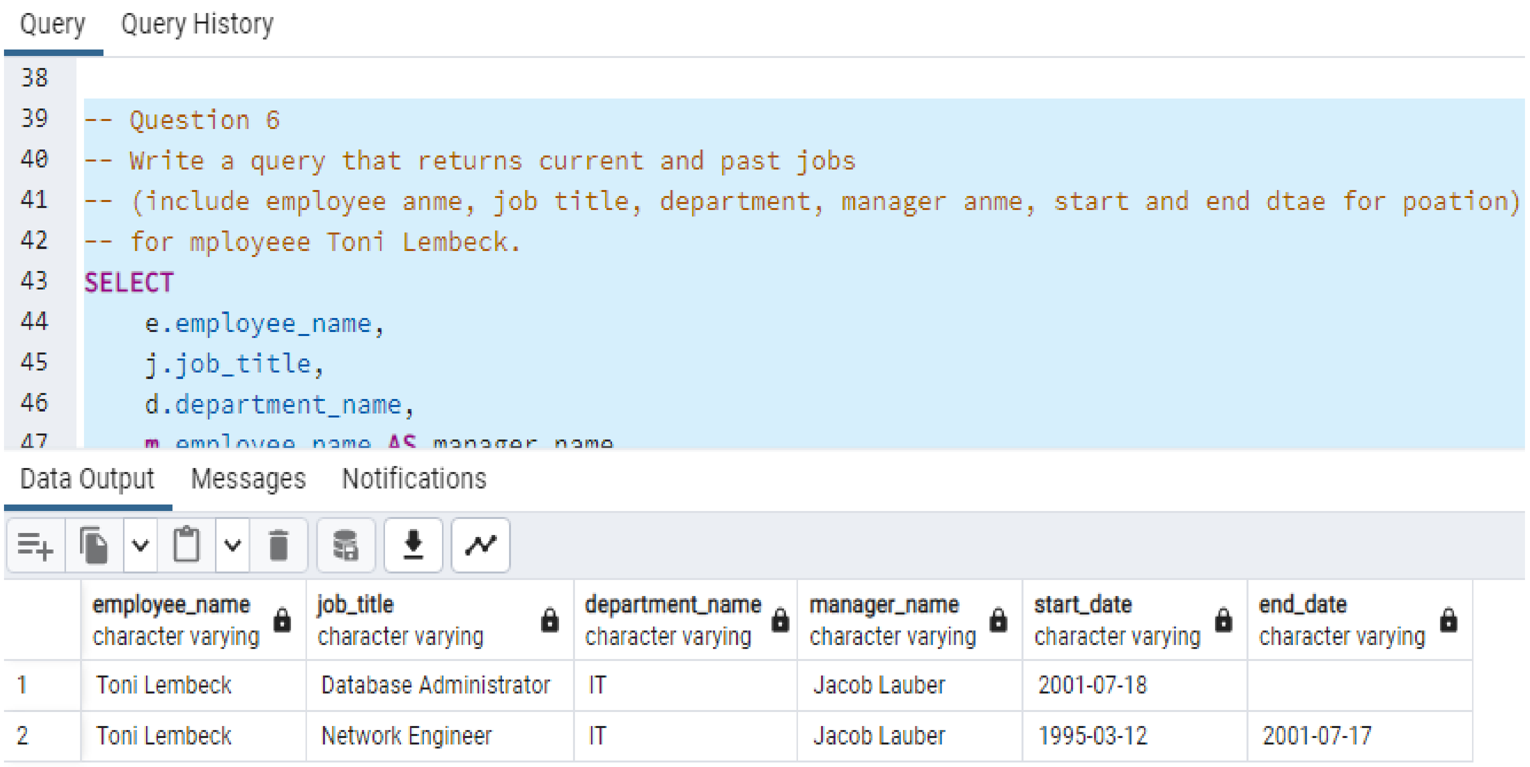
```
SELECT
  d.department_name,
  COUNT(*) AS employee_count
FROM
  employee_detail ed
  JOIN department d ON ed.department_id = d.department_id
  WHERE end_date is null
GROUP BY
  d.department_name;

-- Question 6
```

Data Output   Messages   Notifications

	department_name character varying	employee_count bigint
1	Distribution	25
2	HQ	13
3	IT	52
4	Product Development	69
5	Sales	40

CRUD

- **Question 6: Write a query that returns current and past jobs (include employee name, job title, department, manager name, start and end date for position) for employee Toni Lembeck.**



```

-- Question 6
-- Write a query that returns current and past jobs
-- (include employee anme, job title, department, manager anme, start and end dtae for poation)
-- for mployeee Toni Lembeck.
SELECT
    e.employee_name,
    j.job_title,
    d.department_name,
    m.employee_name AS manager_name
```

	employee_name character varying	job_title character varying	department_name character varying	manager_name character varying	start_date character varying	end_date character varying
1	Toni Lembeck	Database Administrator	IT	Jacob Lauber	2001-07-18	
2	Toni Lembeck	Network Engineer	IT	Jacob Lauber	1995-03-12	2001-07-17

CRUD

- **Question 7: Describe how you would apply table security to restrict access to employee salaries using an SQL server.**

  Answer:

  1) To ensure data security, we enforce restrictions on employee access to sensitive information such as employee salaries. By revoking access to the Salary table at the user level, we can effectively prevent employees from accessing this confidential data.

  2) Additionally, we can enhance data privacy by creating customized views of the Employee Details table. These views act as an abstraction layer, allowing us to display only non-sensitive data elements to users when they access the database. By hiding the actual table names and critical data elements, we can protect sensitive information from being exposed on the front-end.