Tech ABC Corp - HR Database

[Student Name & Date]

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

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:**

  uphold data integrity and increase data security

- **Describe current data management solution:**

  They created an excel file containing all the data.

- **Describe current data available:** employee ID, name, email, employment date, department, manager name, start date, finish date, location, address, city, state, and degree of education

- **Additional data requests:**They request that these data be kept for at least 7 years. In the future, they would like to be able to interface with the system of the payroll department.

- **Who will own/manage data:**the management and the HR

  employees

- **Who will have access to database:**Each employee with a domain login is permitted read-only access to the database, but they are not permitted access to the wage data. Instead, management and HR staff members can view the salary information and have read-only and write access to it.

Data Architect Business Requirement

- **Estimated size of database**

  206 rows and 15 columns

- **Estimated annual growth**

  20% annual growth for the following five years

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

  salary data are restricted for employees who are not manager or HR employees

Data Architect Technical Requirement

- **Justification for the new database**

  integrity of data and security.

- **Database objects**
  - **Table** → education_level, employee, employment, manager, location, department, job, salary
  - **View Table** → manager (created in order to support the creation of the the table employment)

- **Data ingestion**

  ETL

Data Architect Technical Requirement

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

  Ownership: HR Employees

  User Access: **E**very worker. But only management and HR staff are have access to wage information.

- **Scalability**

  expansion

- **Flexibility:**In the future, a direct feed could be quite helpful to link the payroll system with the actual database.

- **Storage & retention**

  Storage (disk or in-memory): disk

  Retention: 7 years

- **Backup**

  weekly complete backup and daily interval backups

Step 2

Relational Database Design

Step 2: Relational Database Design

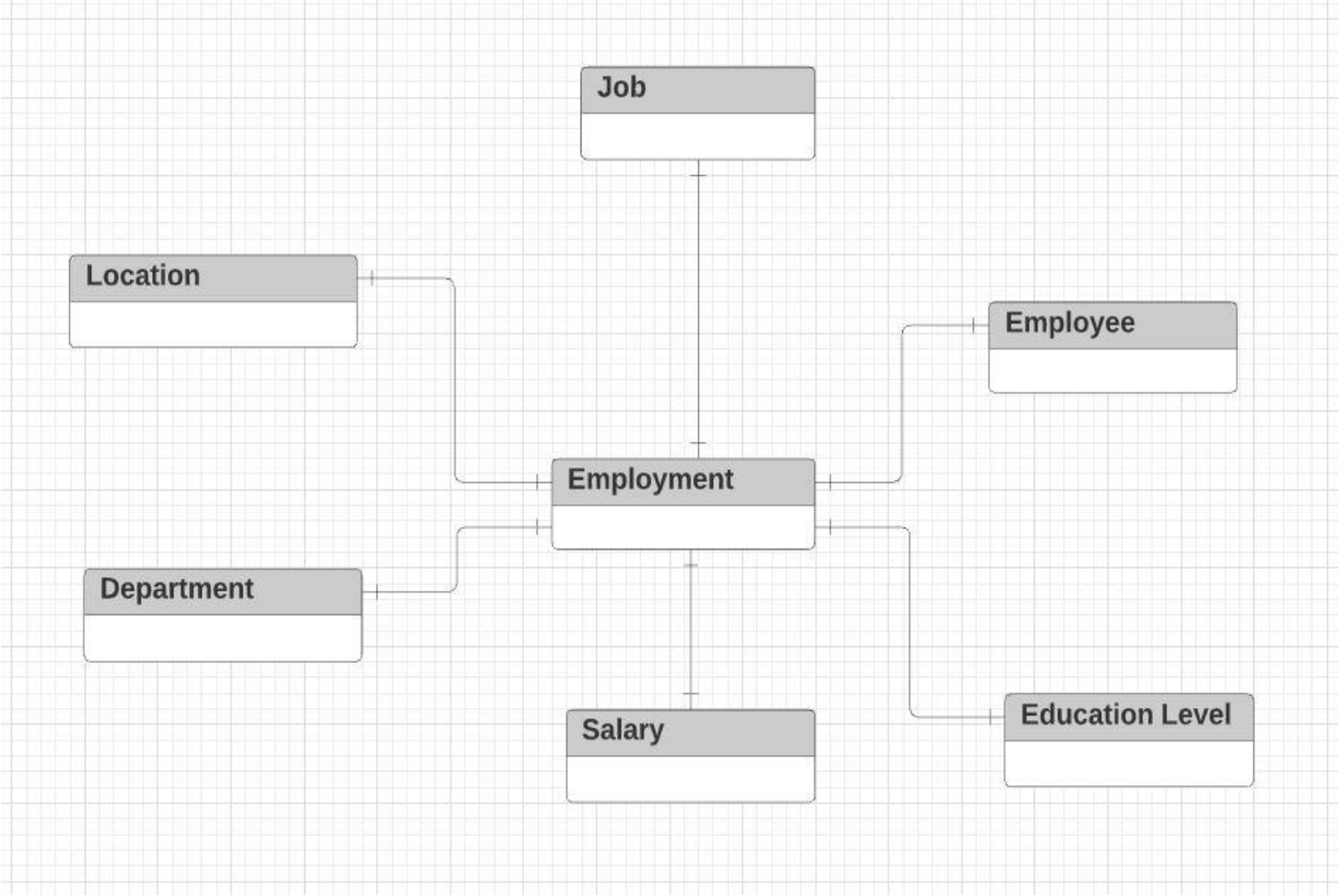
This step is where you will go through the process of designing a new database for Tech ABC Corp's HR department. Using the dataset provided, along with the requirements gathered in step one, you are going to develop a relational database set to the 3NF.

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.

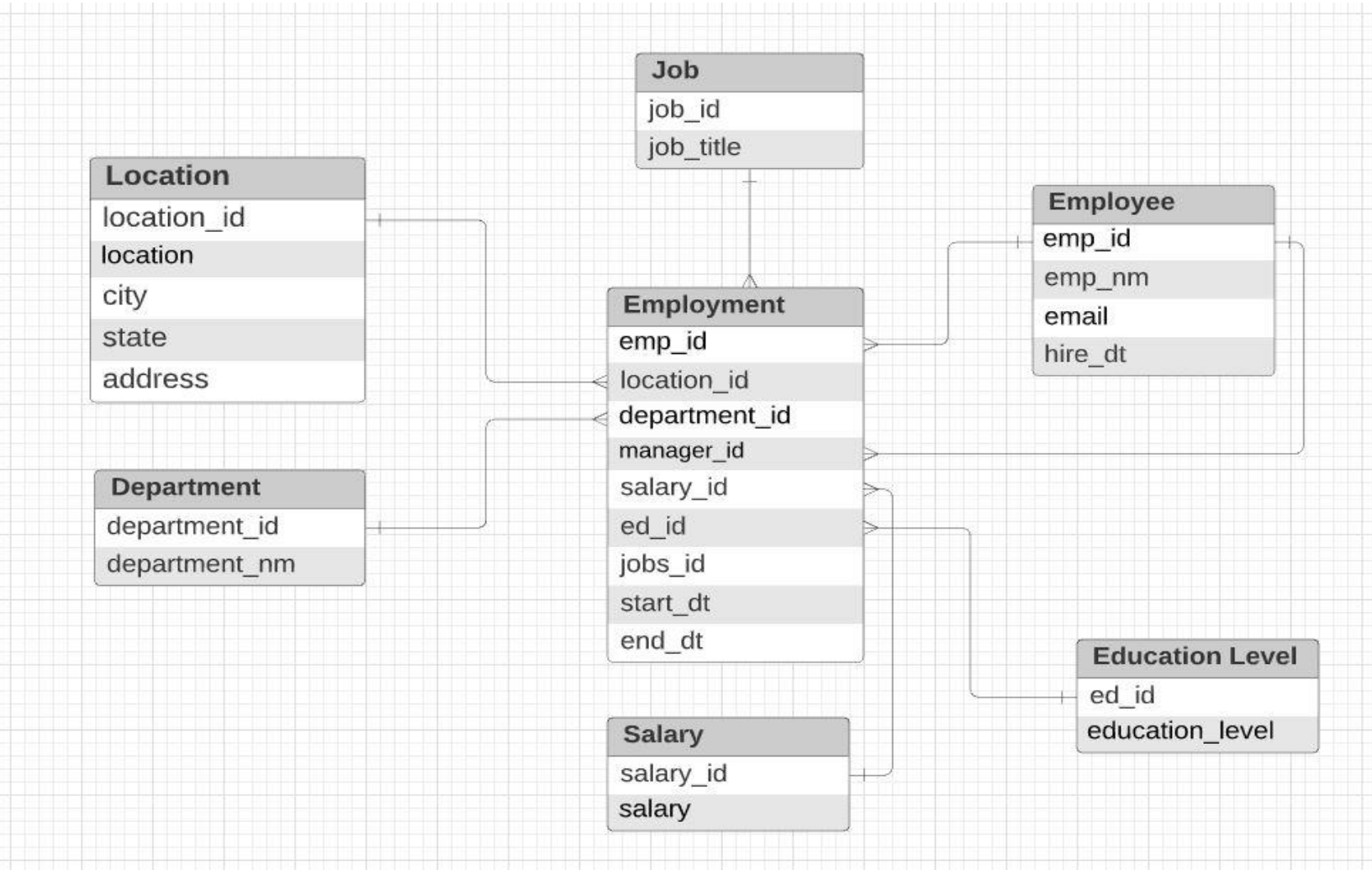
ERD

- **Conceptual**



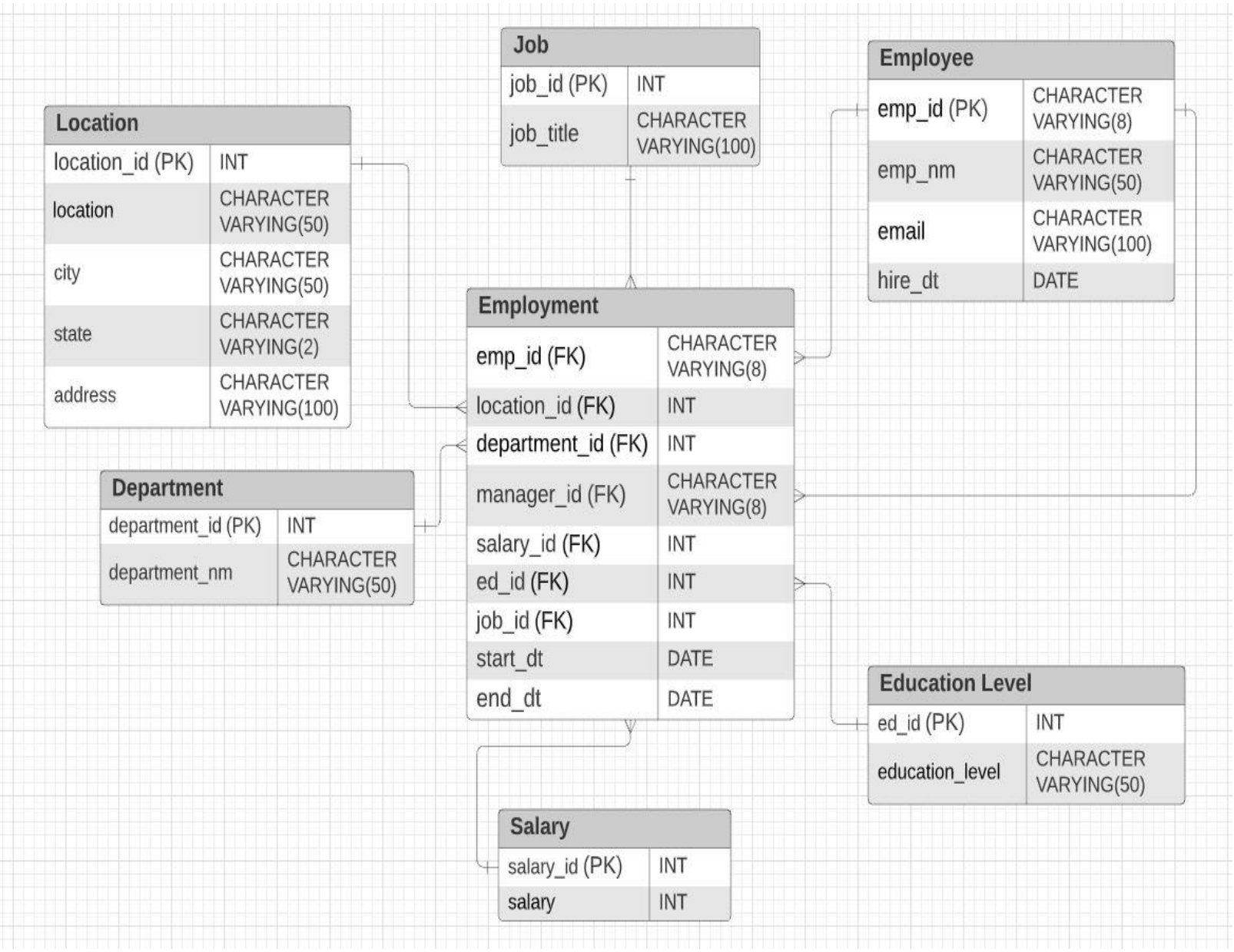
ERD

- **Logical**


Job
job_id
job_title
Location
location_id
location
city
state
address
Employee
emp_id
emp_nm
email
hire_dt
Employment
emp_id
location_id
department_id
manager_id
salary_id
ed_id
jobs_id
start_dt
end_dt
Department
department_id
department_nm
Education Level
ed_id
education_level
Salary
salary_id
salary

ERD

- **Physical**



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

```
CREATE TABLE Employee (
     emp_id CHARACTER VARYING(8) PRIMARY KEY,
     emp_nm CHARACTER VARYING(50),
     email CHARACTER VARYING(100),
     hire_dt DATE);


CREATE TABLE Job (
     job_id SERIAL PRIMARY KEY,
     job_title CHARACTER VARYING(100));


CREATE TABLE Department (
     department_id SERIAL PRIMARY KEY,
     department_nm CHARACTER VARYING(50));


CREATE TABLE Salary (
     salary_id SERIAL PRIMARY KEY,
     salary INTEGER);


CREATE TABLE Location (
     location_id SERIAL PRIMARY KEY,
     location CHARACTER VARYING(50),
     state CHARACTER VARYING(2),
     city CHARACTER VARYING(50),
     address CHARACTER VARYING(100));


CREATE TABLE education_level (
     ed_id SERIAL PRIMARY KEY,
     education_level CHARACTER VARYING(50));
```

DDL

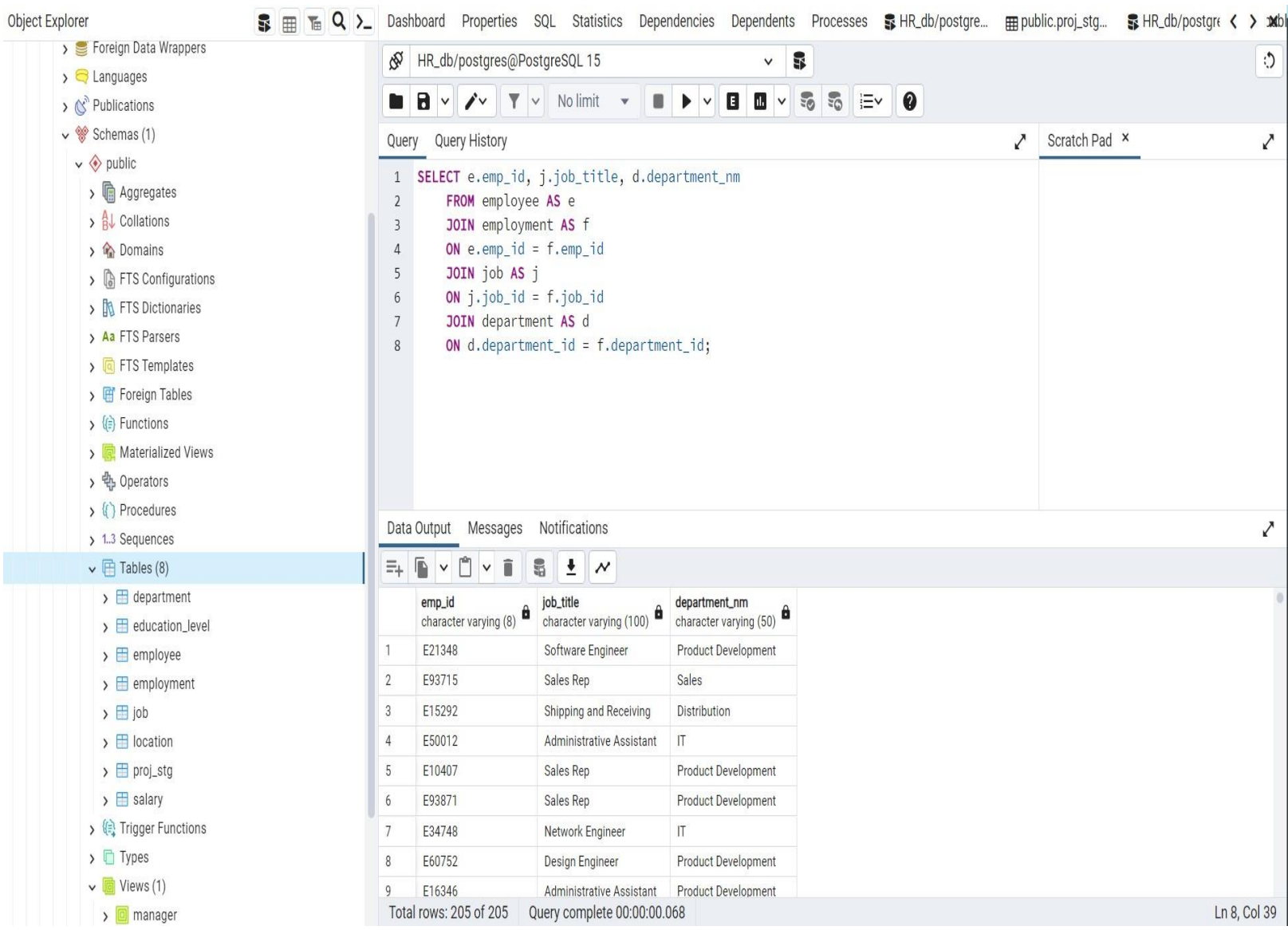
Create a DDL SQL script capable of building the database you designed in Step 2

```
CREATE TABLE Employment (
    emp_id CHARACTER VARYING(8),
    location_id INTEGER,
    department_id INTEGER,
    salary_id INTEGER,
    ed_id INTEGER,
    job_id INTEGER,
    manager_id CHARACTER VARYING(8),
    start_dt DATE,
    end_dt DATE);
```

```
CREATE VIEW manager
AS SELECT s.emp_id AS manager_id,
p.manager AS manager_name
FROM proj_stg AS p
FULL JOIN (SELECT DISTINCT emp_id, emp_nm FROM proj_stg
WHERE emp_nm IN (SELECT DISTINCT manager FROM proj_stg)) AS s
ON p.manager=s.emp_nm;
```

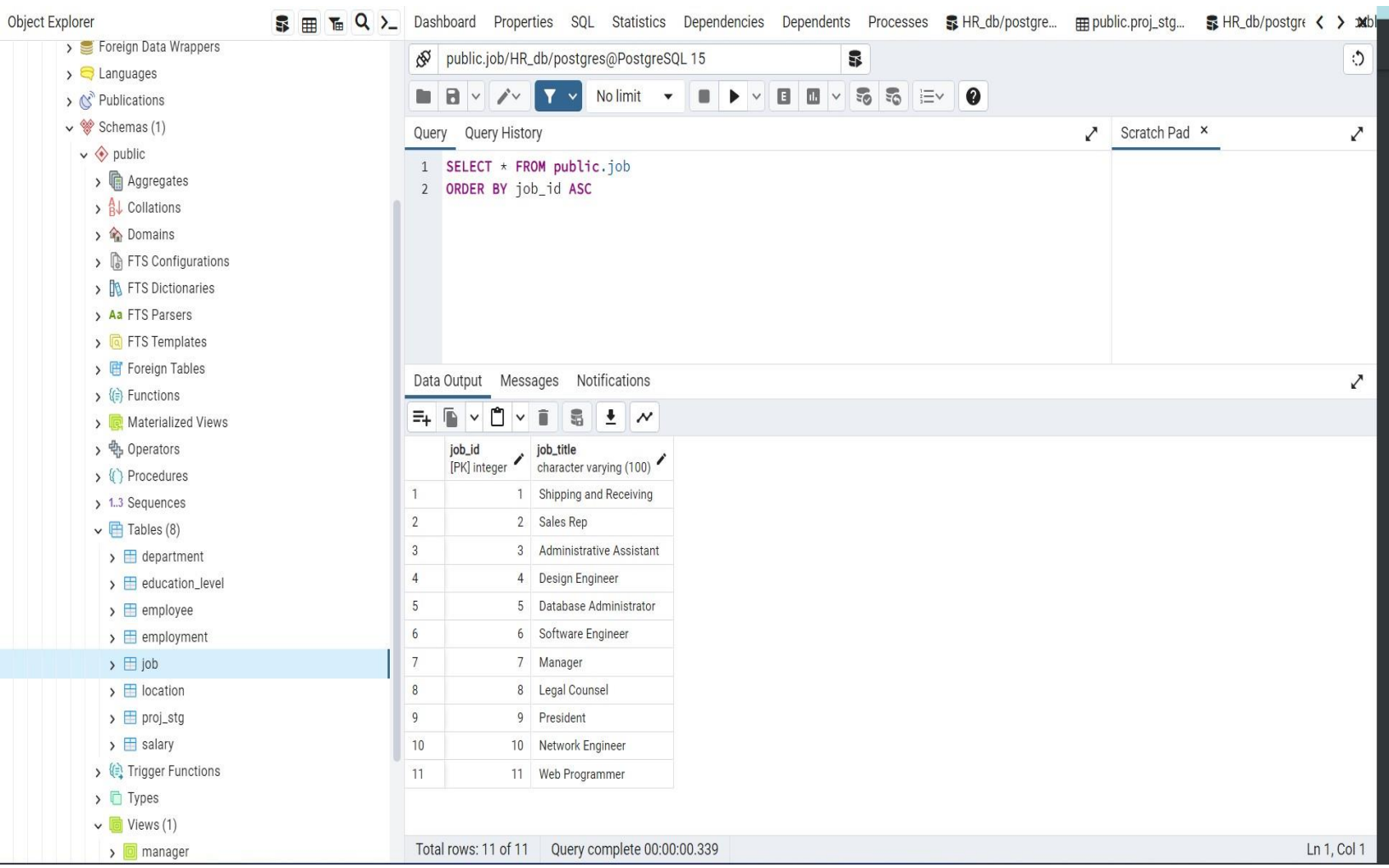
CRUD

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



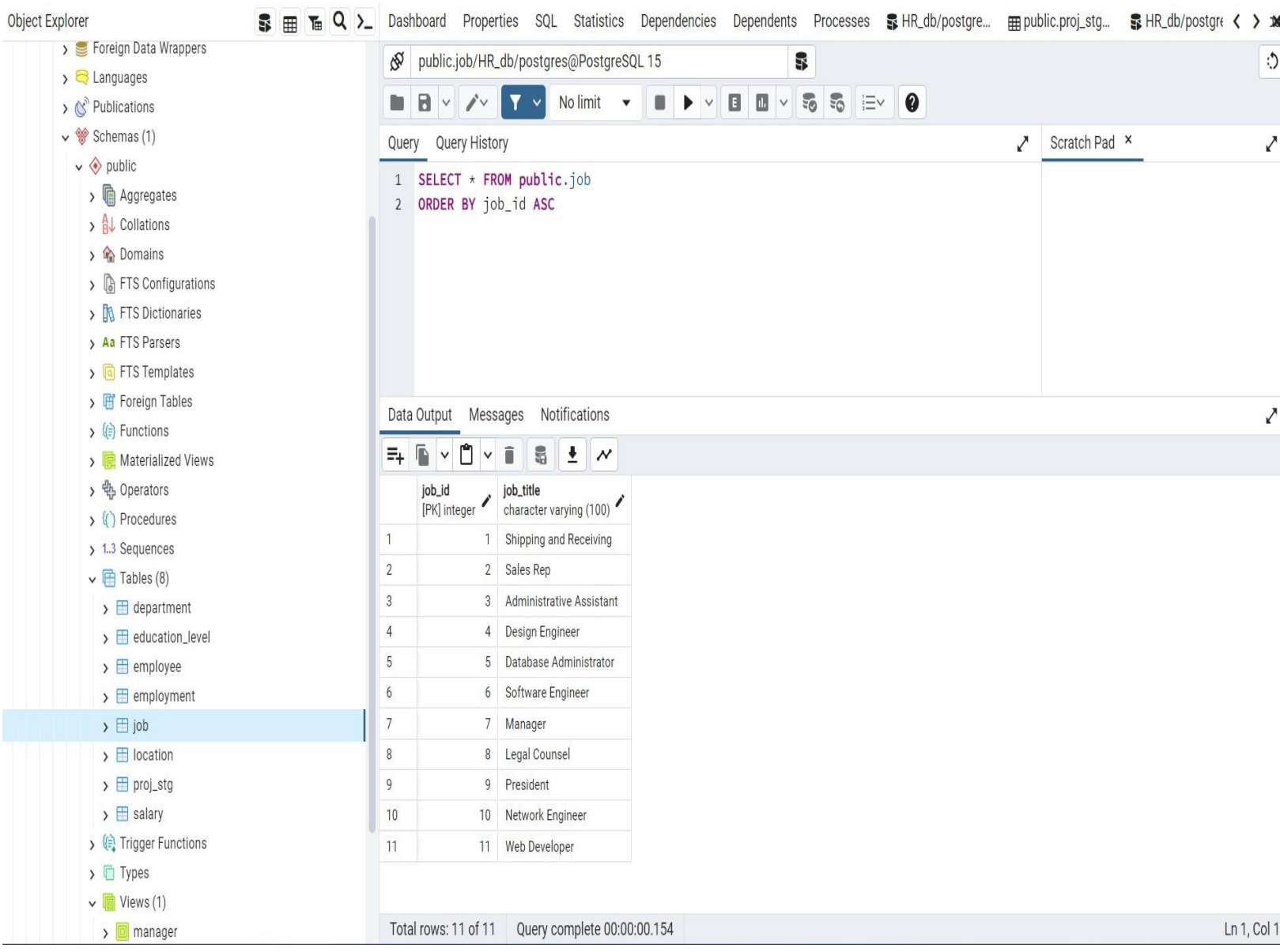
CRUD

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



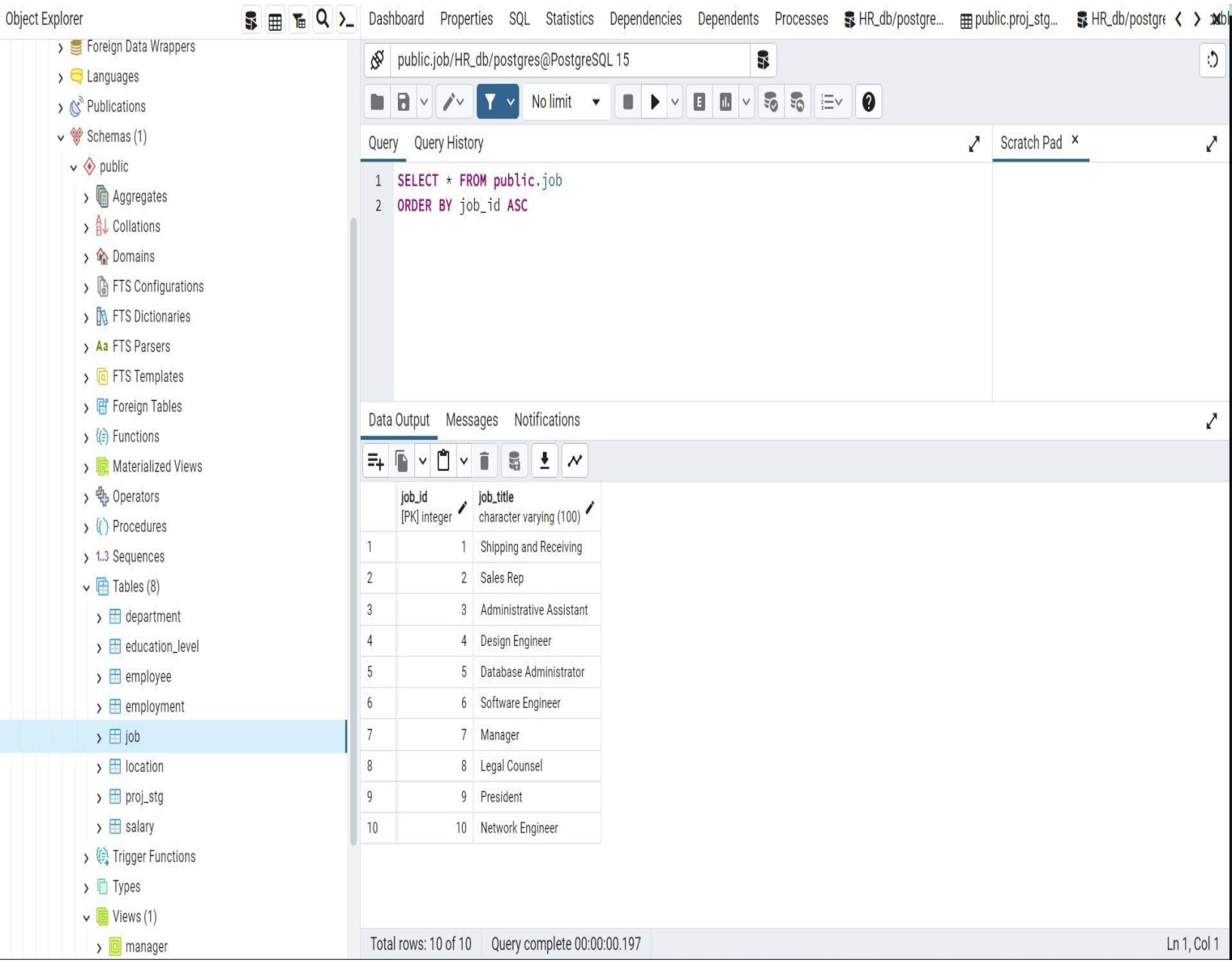
CRUD

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



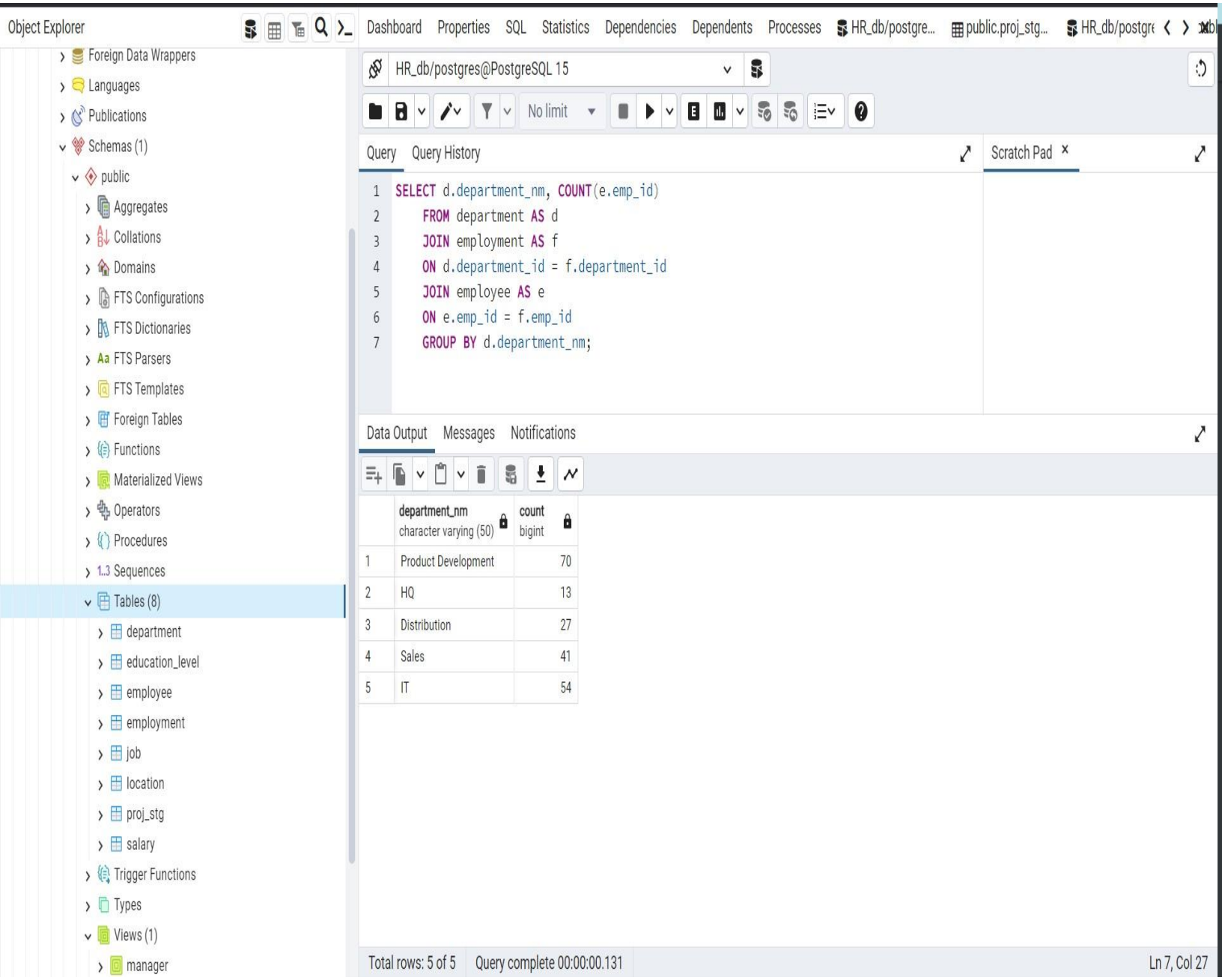
CRUD

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



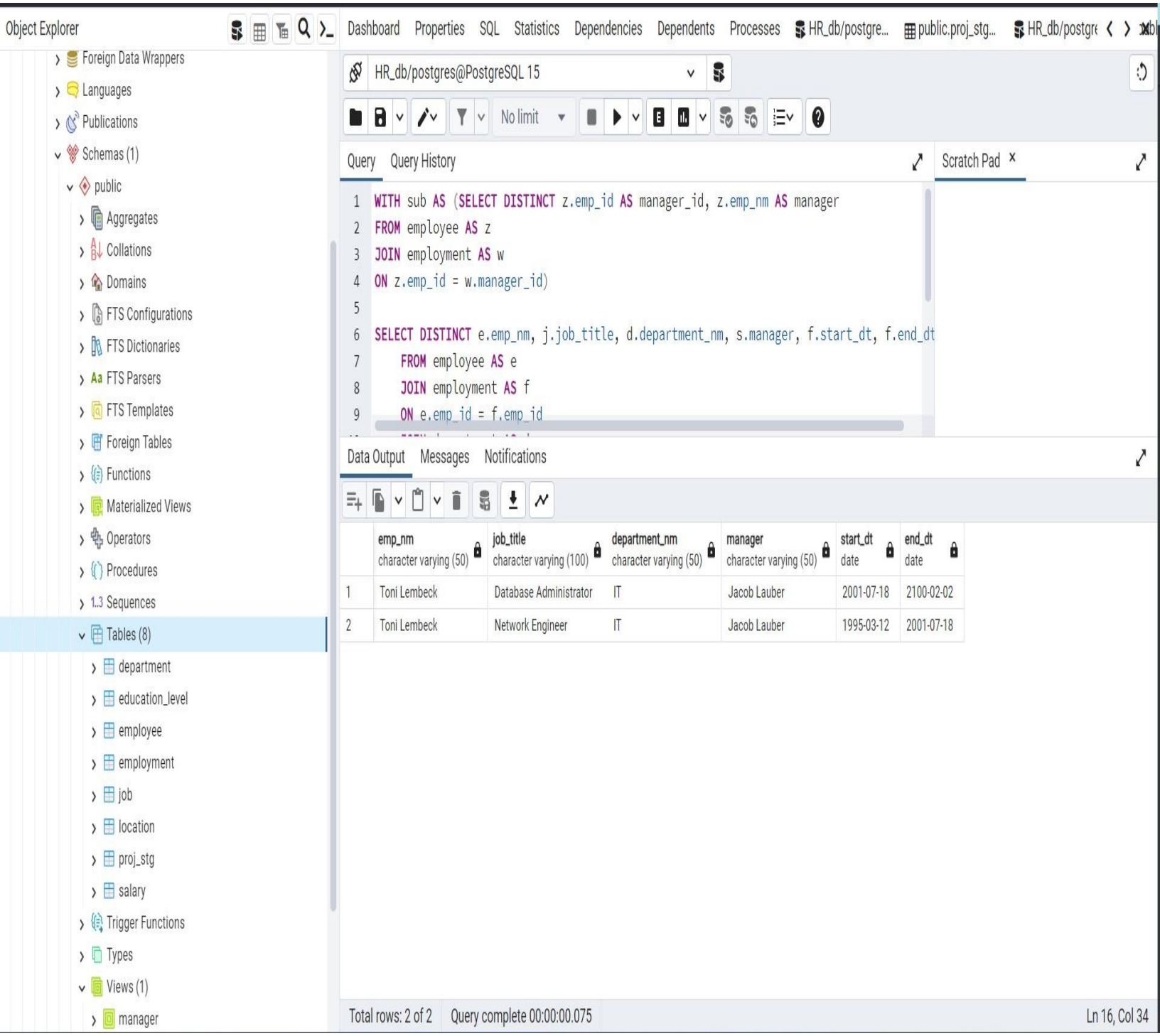
CRUD

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



CRUD

- **Question 6: Write a query that returns current and**



CRUD

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

I believe the best option to limit access to employee pay is to implement row-level security, which enables you to restrict access to the salary table to only management and HR staff.

Step 4

Above and Beyond

(optional)

Appendix