Tech ABC Corp - HR Database

Eric Lok   March 6, 2021

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:**

  Business partner experienced a growth in company size from 10 employees to 200 in 6 months. The spreadsheet used to manage the 10 employees does not scale well to accommodate 200 employees. Also, new functionality is required such as providing any employee with read access except salary information. This type of functionality is best suited for a database.

- **Describe current data management solution:**

  Current method of data storage and management is an excel spreadsheet maintained by one person.

- **Describe current data available:**

  The business data currently has 11 columns and 206 rows. The data columns include title, department, manager's name, hire date, start date, end date, work location, and salary. The data is in denormalized form.

- **Additional data requests:**

  In the future, the user wants to integrate with the payroll department's system to track attendance and paid time off.

- **Who will own/manage data**

  HR and management level employees will manage the data.

Data Architect Business Requirement

- **Who will have access to database**

  All employees will have read only access to the database except to the salary information. HR has read and write access with no restrictions.

- **Estimated size of database**

  There are 206 rows and 11 columns.

- **Estimated annual growth**

  There is an expected 20% growth each year for the next 5 years

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

  Salary data is extremely sensitive and must be restricted to HR access only.

Data Architect Technical Requirement

- **Justification for the new database**
  1. Expected growth of 20% a year over the next 5 years.
  2. Data access, security, integrity, and scale are all new necessary features
- **Database objects**

  The following database tables will be created for this project:

  - Employee
  - Education
  - JobMapping
  - Job
  - Salary
  - Department
  - OfficeLocation
- **Data ingestion**

  Based on the information provided, an ETL process would be best fit to ingest the data into the database. The source data is in excel/flat file form and fits nicely into an ETL workflow.

Data Architect Technical Requirement

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

  Ownership: HR will own and maintain the data

  User Access: Employees will have read only access to the database except to salary data. HR and management level employees have read and write access.

- **Scalability**

  Sharding is unnecessary as there are no plans for any reporting capabilities at the moment. However, replication and backup is necessary to comply with federal regulations to maintain data for at least 7 years.

- **Flexibility**

  Tables that represent real life entities such as employees and jobs are separated into different tables so they can be integrated to future entities as necessary.

- **Storage & retention**
  Storage (disk or in-memory): We will use disk storage. In memory storage is fast but expensive and not necessary for this project.

  Retention: Data has to be kept 7 years per regulation.

Data Architect Technical Requirement

- **Backup**

  IT Best Practices document lists Backup schedule requirements

  Due to the sensitivity of the data and the request for a live database in which HR can input and edit information, the critical backup schedule is chosen. Critical back schedule includes a full backup per week and incremental backup daily.

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.

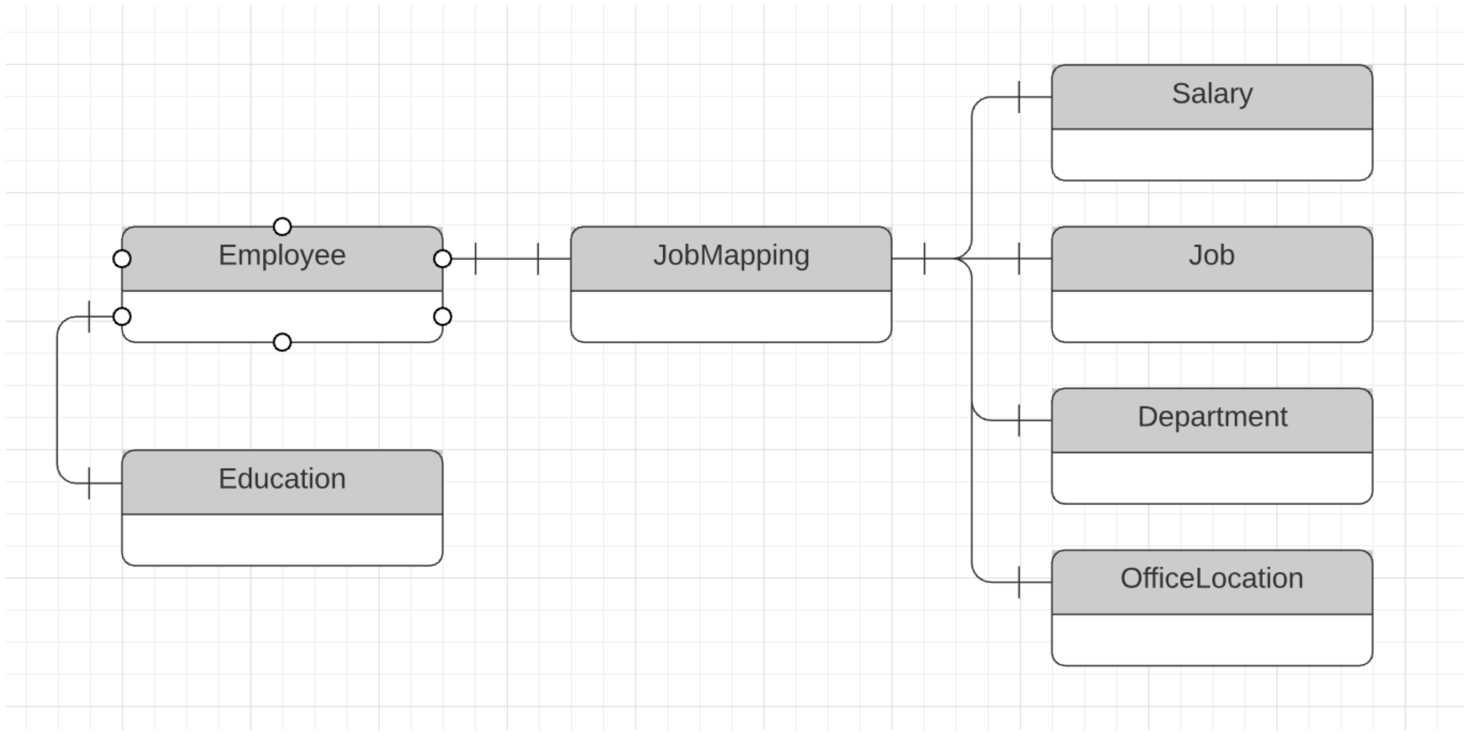
You will submit a screenshot for each of the 3 ERDs you create. You will find detailed instructions for developing each of the ERDs over the next several pages.

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.

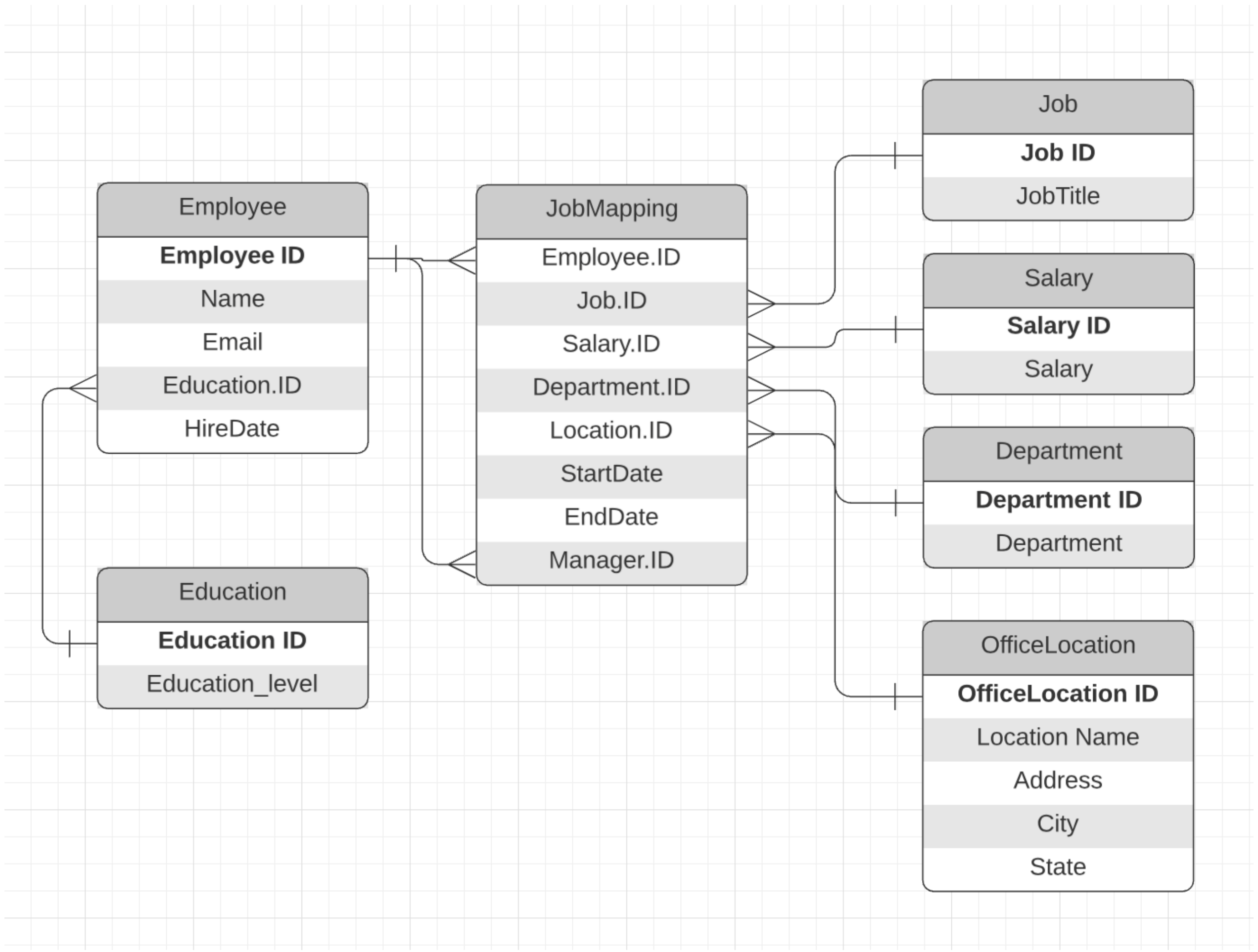


ERD

- **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.

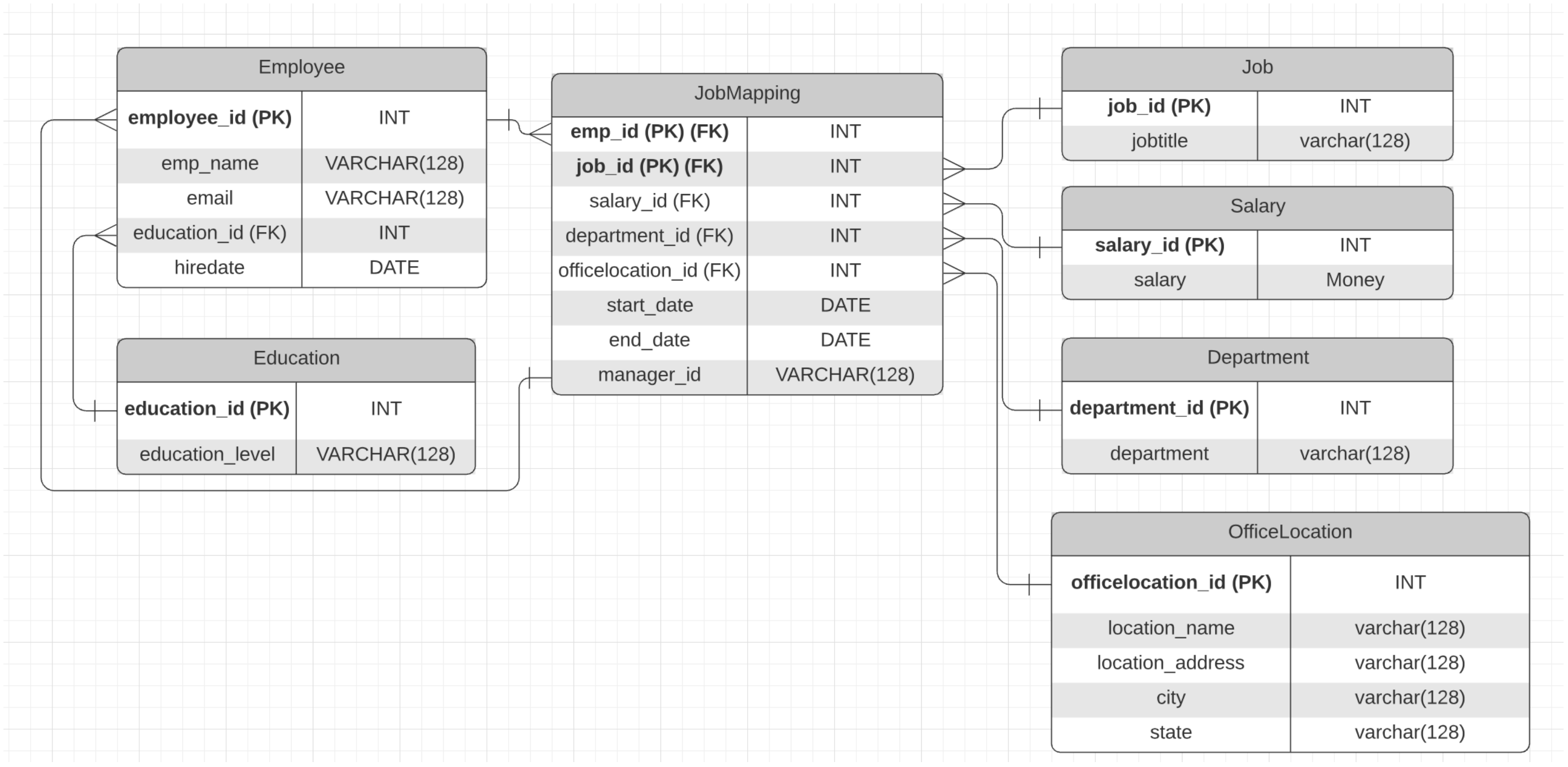
Use Lucidchart's built-in template for DBMS ER Diagram UML.



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.

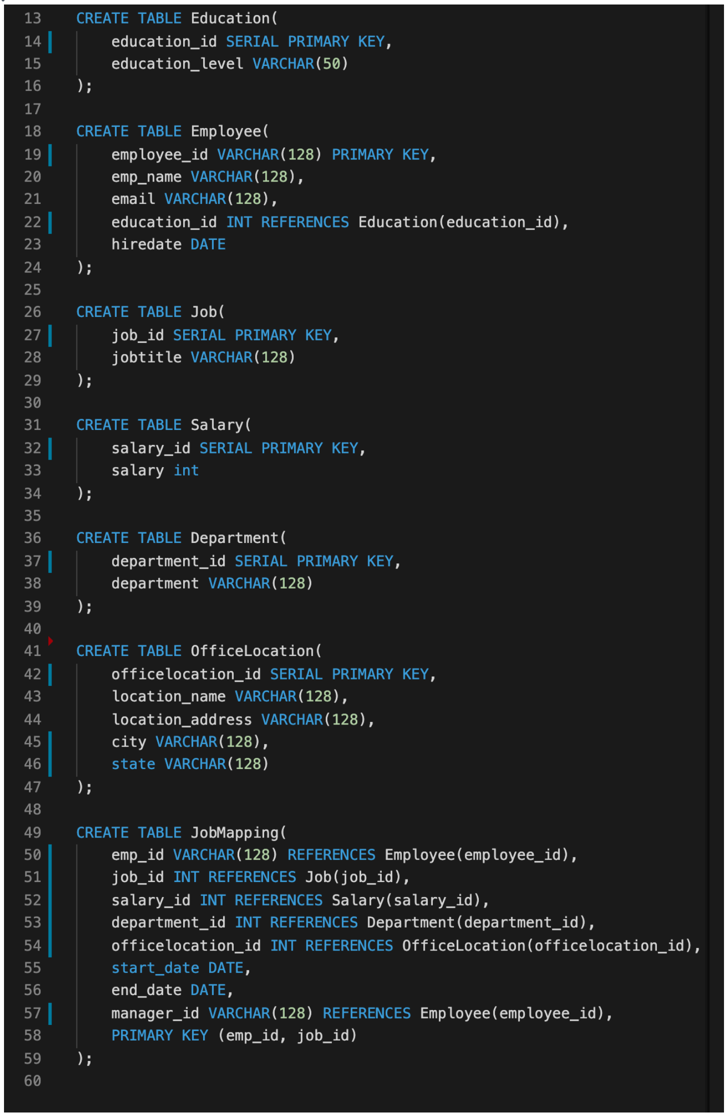
After running CRUD commands like update, insert, or delete, run a `SELECT*` command on the affected table, so the reviewer can see the results of the command.

DDL

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

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

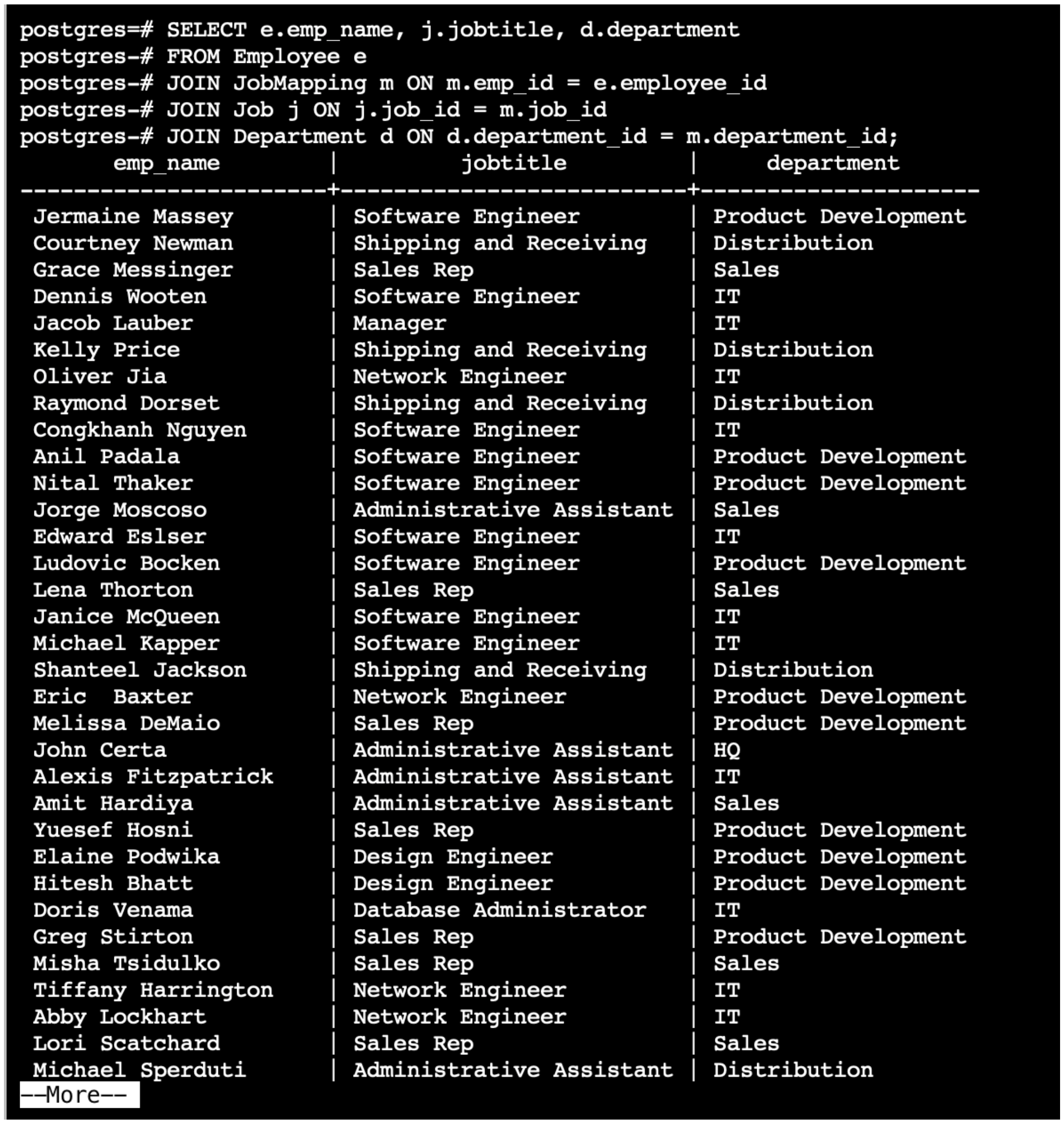
Foreign keys cannot be created on tables that do not exist yet, so it may be easier to create all tables in the database, then to go back and run modify statements on the tables to create foreign key constraints.



```sql
CREATE TABLE Education(
    education_id SERIAL PRIMARY KEY,
    education_level VARCHAR(50)
);

CREATE TABLE Employee(
    employee_id VARCHAR(128) PRIMARY KEY,
    emp_name VARCHAR(128),
    email VARCHAR(128),
    education_id INT REFERENCES Education(education_id),
    hiredate DATE
);

CREATE TABLE Job(
    job_id SERIAL PRIMARY KEY,
    jobtitle VARCHAR(128)
);

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

CREATE TABLE Department(
    department_id SERIAL PRIMARY KEY,
    department VARCHAR(128)
);

CREATE TABLE OfficeLocation(
    officelocation_id SERIAL PRIMARY KEY,
    location_name VARCHAR(128),
    location_address VARCHAR(128),
    city VARCHAR(128),
    state VARCHAR(128)
);

CREATE TABLE JobMapping(
    emp_id VARCHAR(128) REFERENCES Employee(employee_id),
    job_id INT REFERENCES Job(job_id),
    salary_id INT REFERENCES Salary(salary_id),
    department_id INT REFERENCES Department(department_id),
    officelocation_id INT REFERENCES OfficeLocation(officelocation_id),
    start_date DATE,
    end_date DATE,
    manager_id VARCHAR(128) REFERENCES Employee(employee_id),
    PRIMARY KEY (emp_id, job_id)
);
```

CRUD

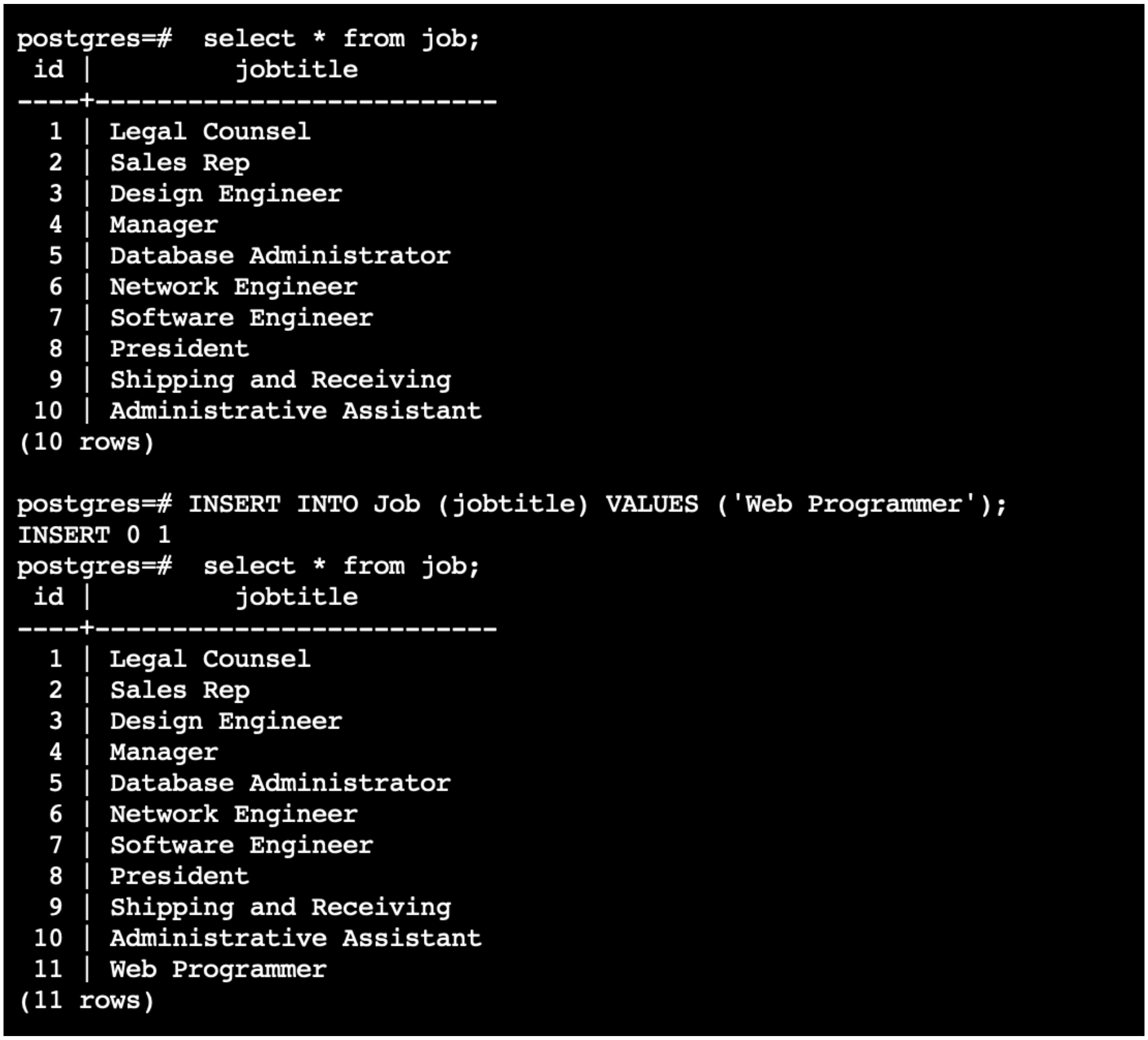
- **Question 1: Return a list of employees with Job Titles and Department Names**



```
postgres=# SELECT e.emp_name, j.jobtitle, d.department
postgres-# FROM Employee e
postgres-# JOIN JobMapping m ON m.emp_id = e.employee_id
postgres-# JOIN Job j ON j.job_id = m.job_id
postgres-# JOIN Department d ON d.department_id = m.department_id;
       emp_name        |         jobtitle          |     department
-----------------------+---------------------------+---------------------
 Jermaine Massey       | Software Engineer         | Product Development
 Courtney Newman       | Shipping and Receiving    | Distribution
 Grace Messinger       | Sales Rep                 | Sales
 Dennis Wooten         | Software Engineer         | IT
 Jacob Lauber          | Manager                   | IT
 Kelly Price           | Shipping and Receiving    | Distribution
 Oliver Jia            | Network Engineer          | IT
 Raymond Dorset        | Shipping and Receiving    | Distribution
 Congkhanh Nguyen      | Software Engineer         | IT
 Anil Padala           | Software Engineer         | Product Development
 Nital Thaker          | Software Engineer         | Product Development
 Jorge Moscoso         | Administrative Assistant  | Sales
 Edward Eslser         | Software Engineer         | IT
 Ludovic Bocken        | Software Engineer         | Product Development
 Lena Thorton          | Sales Rep                 | Sales
 Janice McQueen        | Software Engineer         | IT
 Michael Kapper        | Software Engineer         | IT
 Shanteel Jackson      | Shipping and Receiving    | Distribution
 Eric  Baxter          | Network Engineer          | Product Development
 Melissa DeMaio        | Sales Rep                 | Product Development
 John Certa            | Administrative Assistant  | HQ
 Alexis Fitzpatrick    | Administrative Assistant  | IT
 Amit Hardiya          | Administrative Assistant  | Sales
 Yuesef Hosni          | Sales Rep                 | Product Development
 Elaine Podwika        | Design Engineer           | Product Development
 Hitesh Bhatt          | Design Engineer           | Product Development
 Doris Venama          | Database Administrator    | IT
 Greg Stirton          | Sales Rep                 | Product Development
 Misha Tsidulko        | Sales Rep                 | Sales
 Tiffany Harrington    | Network Engineer          | IT
 Abby Lockhart         | Network Engineer          | IT
 Lori Scatchard        | Sales Rep                 | Sales
 Michael Sperduti      | Administrative Assistant  | Distribution
--More--
```

CRUD

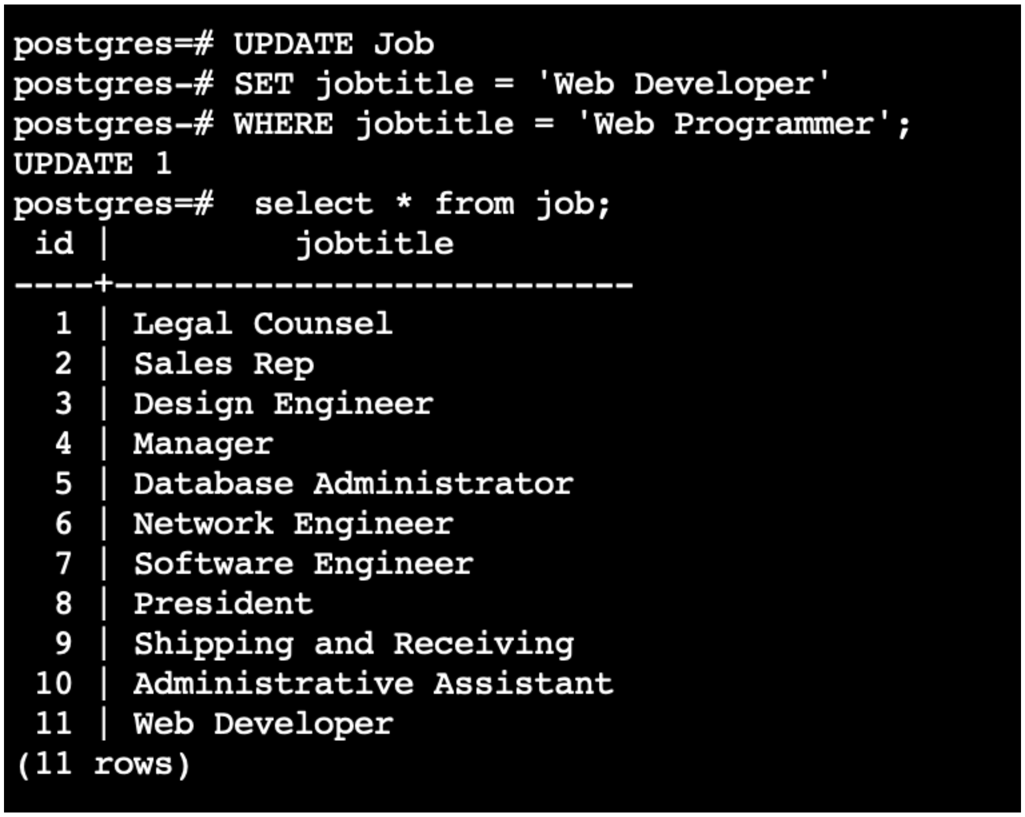
- **Question 2: Insert Web Programmer as a new job title**



```
postgres=#  select * from job;
 id |         jobtitle
----+--------------------------
  1 | Legal Counsel
  2 | Sales Rep
  3 | Design Engineer
  4 | Manager
  5 | Database Administrator
  6 | Network Engineer
  7 | Software Engineer
  8 | President
  9 | Shipping and Receiving
 10 | Administrative Assistant
(10 rows)

postgres=# INSERT INTO Job (jobtitle) VALUES ('Web Programmer');
INSERT 0 1
postgres=#  select * from job;
 id |         jobtitle
----+--------------------------
  1 | Legal Counsel
  2 | Sales Rep
  3 | Design Engineer
  4 | Manager
  5 | Database Administrator
  6 | Network Engineer
  7 | Software Engineer
  8 | President
  9 | Shipping and Receiving
 10 | Administrative Assistant
 11 | Web Programmer
(11 rows)
```

CRUD

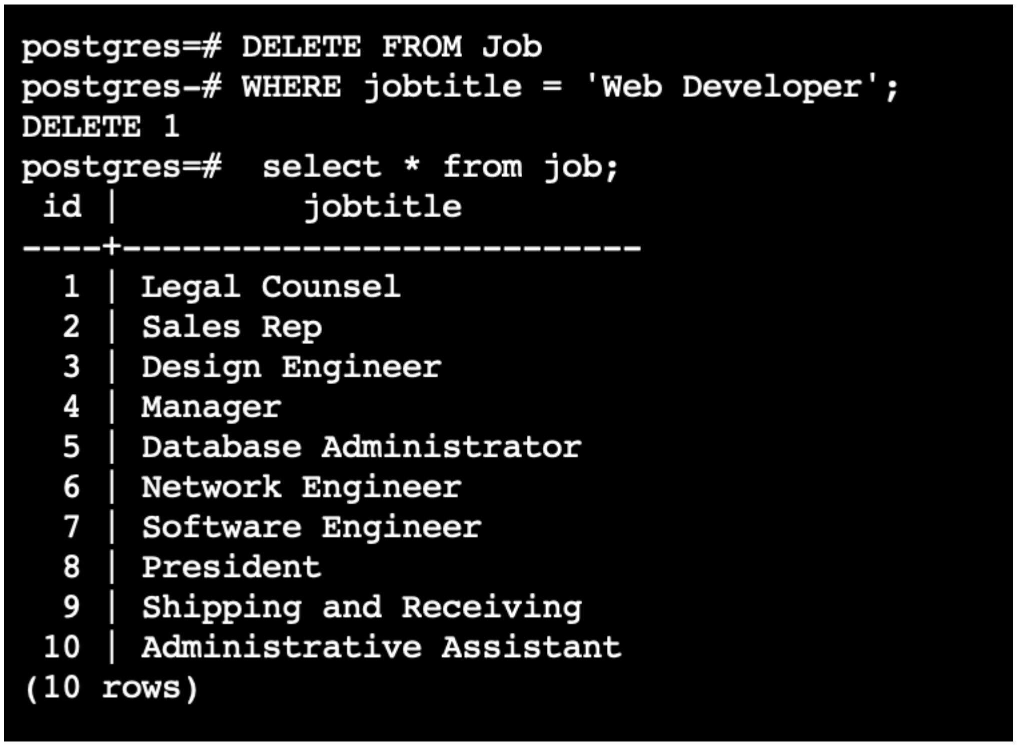
- **Question 3: Correct the job title from web programmer to web developer**



```
postgres=# UPDATE Job
postgres-# SET jobtitle = 'Web Developer'
postgres-# WHERE jobtitle = 'Web Programmer';
UPDATE 1
postgres=#  select * from job;
 id |         jobtitle
----+-------------------------
  1 | Legal Counsel
  2 | Sales Rep
  3 | Design Engineer
  4 | Manager
  5 | Database Administrator
  6 | Network Engineer
  7 | Software Engineer
  8 | President
  9 | Shipping and Receiving
 10 | Administrative Assistant
 11 | Web Developer
(11 rows)
```

CRUD

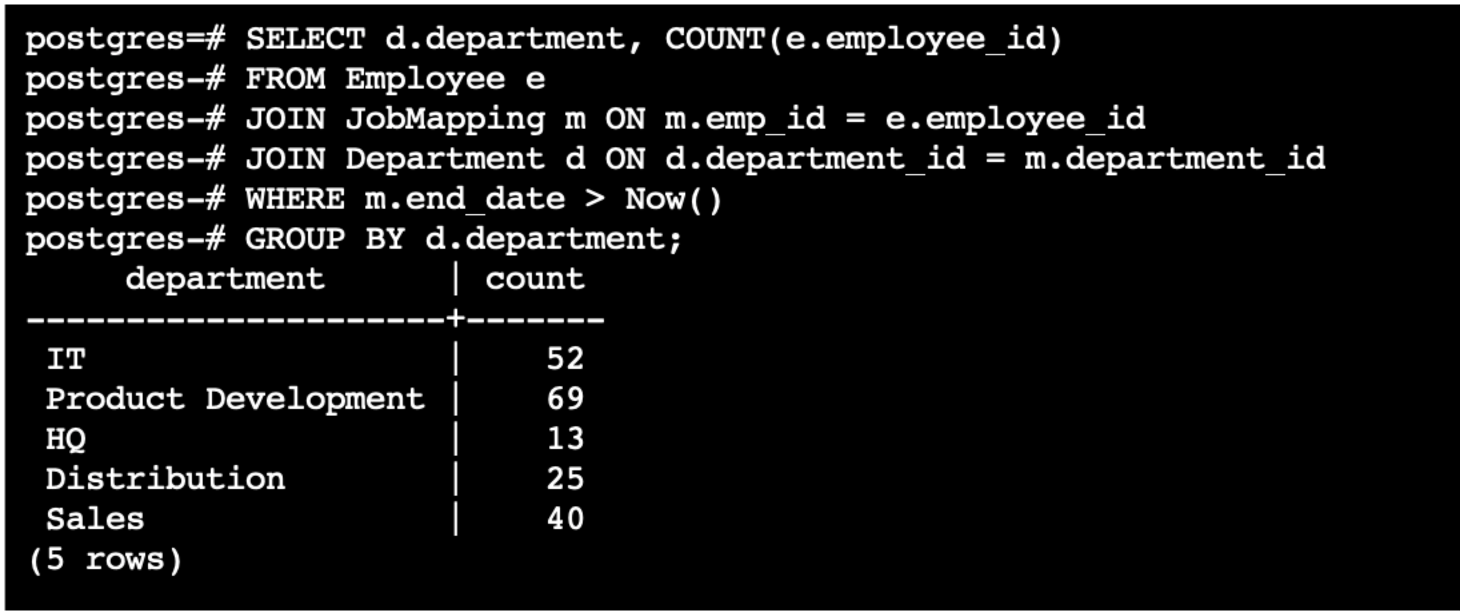
- **Question 4: Delete the job title Web Developer from the database**



```
postgres=# DELETE FROM Job
postgres-# WHERE jobtitle = 'Web Developer';
DELETE 1
postgres=#  select * from job;
 id |          jobtitle
----+--------------------------
  1 | Legal Counsel
  2 | Sales Rep
  3 | Design Engineer
  4 | Manager
  5 | Database Administrator
  6 | Network Engineer
  7 | Software Engineer
  8 | President
  9 | Shipping and Receiving
 10 | Administrative Assistant
(10 rows)
```

CRUD

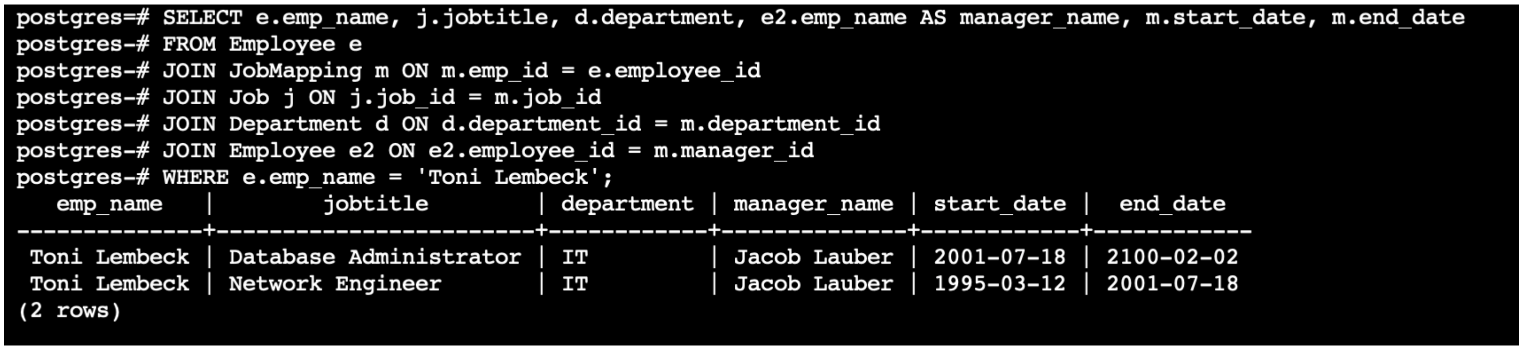
- **Question 5: How many employees are in each department?**



```
postgres=# SELECT d.department, COUNT(e.employee_id)
postgres-# FROM Employee e
postgres-# JOIN JobMapping m ON m.emp_id = e.employee_id
postgres-# JOIN Department d ON d.department_id = m.department_id
postgres-# WHERE m.end_date > Now()
postgres-# GROUP BY d.department;
     department       | count
----------------------+-------
 IT                   |    52
 Product Development  |    69
 HQ                   |    13
 Distribution         |    25
 Sales                |    40
(5 rows)
```

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.**



```
postgres=# SELECT e.emp_name, j.jobtitle, d.department, e2.emp_name AS manager_name, m.start_date, m.end_date
postgres-# FROM Employee e
postgres-# JOIN JobMapping m ON m.emp_id = e.employee_id
postgres-# JOIN Job j ON j.job_id = m.job_id
postgres-# JOIN Department d ON d.department_id = m.department_id
postgres-# JOIN Employee e2 ON e2.employee_id = m.manager_id
postgres-# WHERE e.emp_name = 'Toni Lembeck';
   emp_name   |        jobtitle         | department | manager_name | start_date |  end_date
--------------+-------------------------+------------+--------------+------------+------------
 Toni Lembeck | Database Administrator  | IT         | Jacob Lauber | 2001-07-18 | 2100-02-02
 Toni Lembeck | Network Engineer        | IT         | Jacob Lauber | 1995-03-12 | 2001-07-18
(2 rows)
```

CRUD

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

  The best way to assign permissions are through groups. There will be two groups. The first group contains HR and high level managers. This group has read and write access to all tables including the salary table. The second group contains everyone else in the company. This group has read access only to all tables except the salary table.

  If the first group is “hr_manager_group” and the second is “employee_group” then the commands below would be used to implement this security restriction:

  revoke SELECT on salary from employee_group ;

  grant SELECT on salary TO hr_manager_group;

Step 4

Above and Beyond

(optional)

Step 4: Above and Beyond

This last step is called Above and Beyond. In this step, I have proposed 3 challenges for you to complete, which are above and beyond the scope of the project. This is a chance to flex your coding muscles and show everyone how good you really are.

These challenge steps will bring your project even more in line with a real-world project, as these are the kind of "finishing touches" that will make your database more usable. Imagine building a car without air conditioning or turn signals. Sure, it will work, but who would want to drive it.

I encourage you to take on these challenges in this course and any future courses you take. I designed these challenges to be a challenge to your current abilities, but I ensured they are not an unattainable challenge. Remember, these challenges are completely optional - you can pass the project by doing none of them, or just some of them, but I encourage you to at least attempt them!

Standout Suggestion 1

Create a view that returns all employee attributes; results should resemble initial Excel file

** return a screenshot of the view create code, along with the results of a select all on the view

Standout Suggestion 2

Create a stored procedure with parameters that returns current and past jobs (include employee name, job title, department, manager name, start and end date for position) when given an employee name.

** submit screenshot of stored procedure creation code, along with a screenshot of the stored procedure executed using Toni Lembeck as the parameter value

Standout Suggestion 3

Implement user security on the restricted salary attribute.

Create a non-management user named `NoMgr`. Show the code of how your would grant access to the database, but revoke access to the salary data.

Submit screenshot of code

Appendix

Additional Info

You can include supporting or additional information that supports your previous slides, but isn’t necessary for every person to see that looks at your slides.