Tech ABC Corp - HR Database

[Mohit Bansal & 12 Nov 2022]

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 205 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:** Employee data is keep growing so its hard to maintain in Excel sheet and maintain to data integrity and data security data should be store in database system.
- **Describe current data management solution:** Currently they are storing data in Excel workbook with eleven columns. The data is in human-readable format and has not been normalized at all.
- **Describe current data available:** Currently excel sheet consist 205 records with eleven columns. The data lists the names of employees at Tech ABC Corp, as well as information such as email, hire date, job title, salary, department, manager name, start date, end date, location, address, city, state, education level.
- **Additional data requests:** Maintain data at least 7 year for federal regulation and they want to make sure it gets backed up properly. In future they want to connect this database with payroll department's system.
- **Who will own/manage data**
  Management and HR employees
- **Who will have access to database**
  - Any employee with a domain login to be have read only access the database.
  - No employee will have access to salary information
  - Management and HR employees will have only read and write access on complete database.

Data Architect Business Requirement

- **Estimated size of database**

  Number of Rows =205
  Number of columns=11

- **Estimated annual growth**

  20% growth a year for the next 5 years

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

  Salary information are restricted for employee who are not in management and HR team.

- **Justification for the new database**
  - Expected growth of 20% a year for the next 5 years, therefore, it is a necessity to move the data from spreadsheet to more manageable database.
  - Maintaining data integrity and data security

Data Architect Technical Requirement

- **Database objects**
  - **The denormalised data** which is provide by business contains: (Employee ID, Employee Name, Email, Hire Date, Job Title, Salary, Department, Manager, Start Date, End Date, Location, Address, City, State, Education Level)
  - **3NF Normalised form tables**
    - **Table** → Education Table, Employee Table, Location Table , Department Table, Job Table, Salary Table, Employment Table
  - **Data Element**

    1. Education Table - (Education_id, Education Level)
    2. Employee Table - (Employee ID, Employee Name, Email, Hire Date, Education ID)
    3. Location Table - (Location ID , Location, City, State, Address)
    4. Department Table – (Department ID, Department Name)
    5. Job Table – (Job ID, Job Name)
    6. Salary Table – (Salary ID, Salary)
    7. Employment Table - (Employee ID, Job ID, Department ID, Manager ID, Start Date, End Date, Location ID, Salary ID)

- **Data ingestion**

  ETL is the appropriate ingestion technique here as we are provided with flat files (excel file)

Data Architect Technical Requirement

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

  Ownership: HR and Management

  User Access:
  1. Employee - Read only access to database No access to salary table
  2. HR and Management – All access (read and write)
- **Scalability**

  Replication
- should be used. As the company grows, the number of employees joining the company will increase and 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.

- **Flexibility**

  A direct feed could be very useful in the future in order to connect the actual DB with the payroll system.

- **Storage & retention**

  Storage (disk or in-memory): Disk based storage should be used
  Retention: As per the federal regulations, it is to be kept for 7 years.

- **Backup:** We have to opt for Critical Backup option. We need to schedule full backup 1x per week, incremental backup daily as the Head of HR stated that she needs a live database where she can often input and edit 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

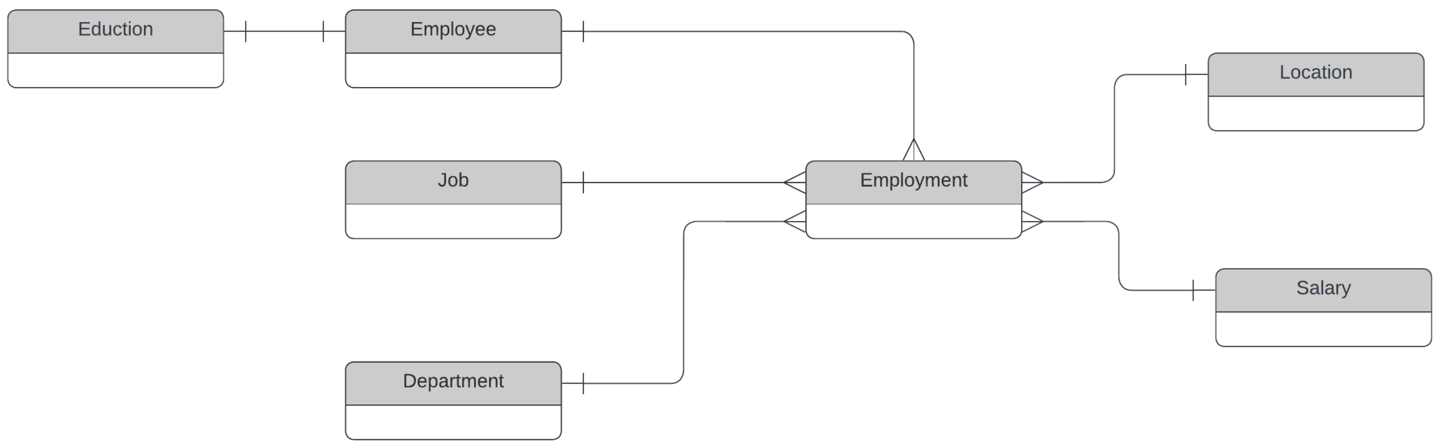
- ## Conceptual

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.

Conceptual ERD For HR DB

Mohit Bansal | November 12, 2022

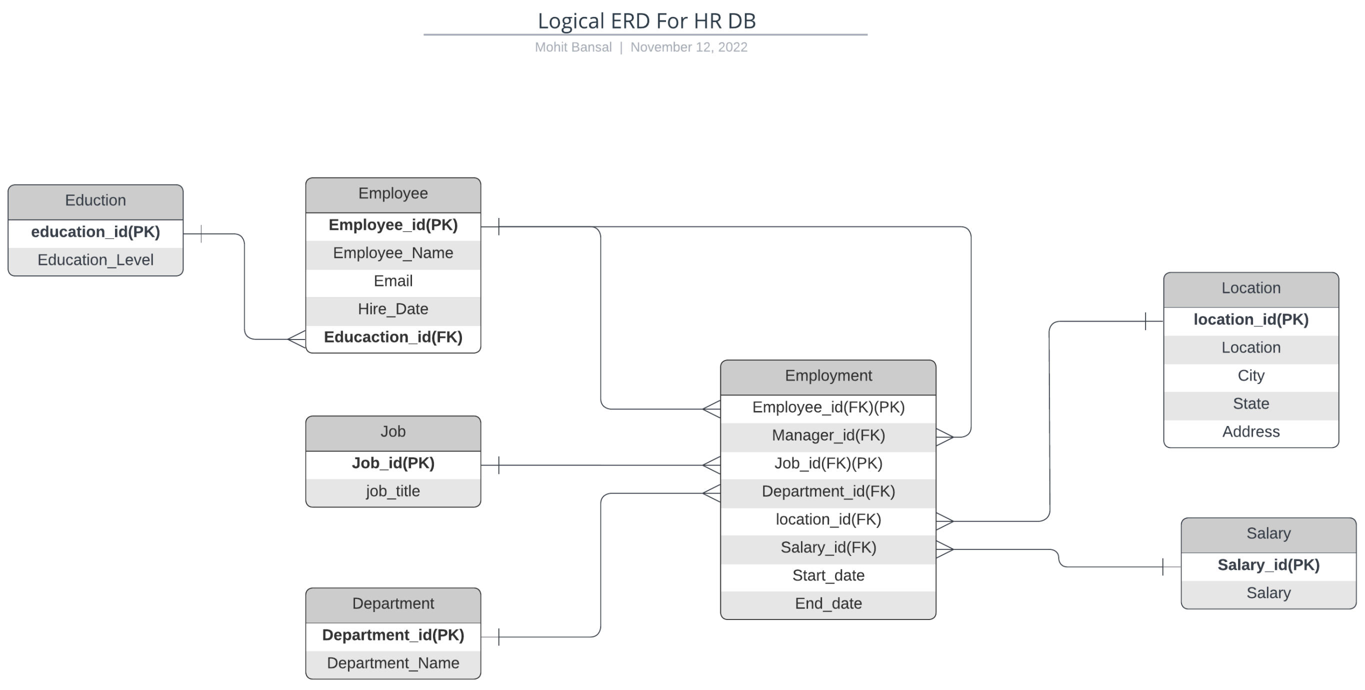


ERD

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.

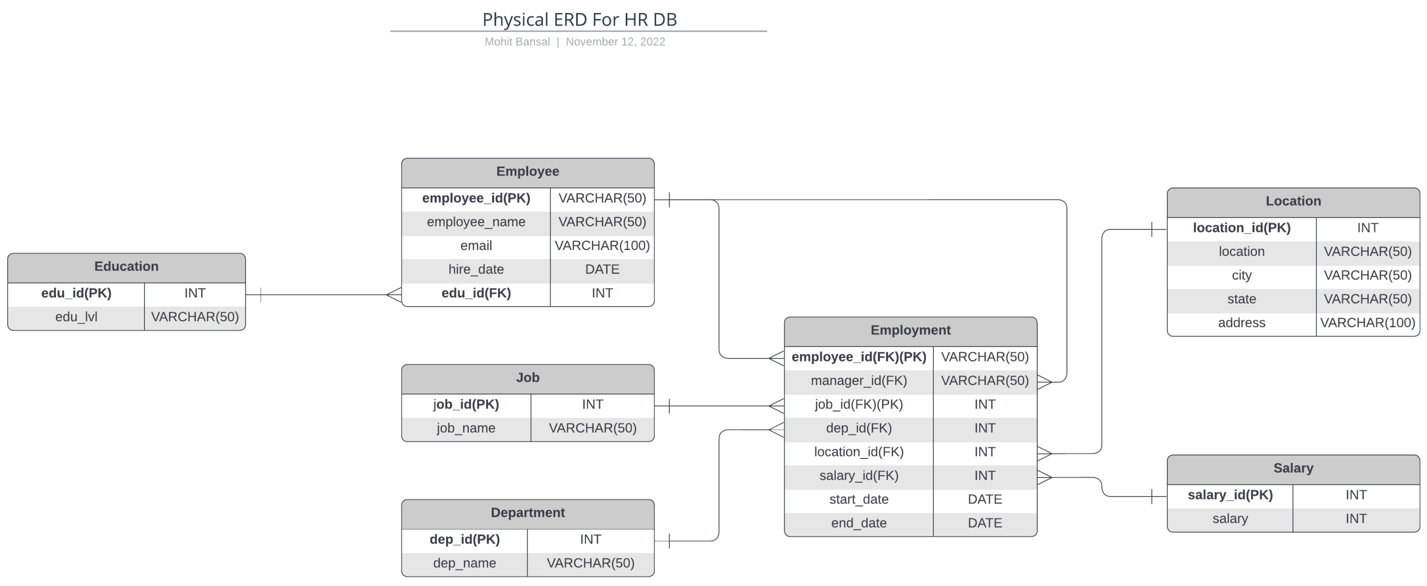
Use Lucidchart's built-in template for DBMS ER Diagram UML.



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.

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.

- Education Table

```
/* Creating Education table (primary key - edu_id) */

CREATE table Education(
    edu_id SERIAL primary key,
    edu_level varchar(50)
);
```

- Employee Table

```
/* Creating Employee table (primary key - employee_id and Forigen_key - edu_id) */

CREATE table Employee(
    employee_id varchar(8) primary key,
    employee_name varchar(50),
    email varchar(50),
    hire_date date,
    edu_id int references Education(edu_id)
);
```

DDL

- Salary Table

```
/* Creating Salary table (primary key - salary id) */

CREATE table Salary(
    salary_id serial primary key,
    salary int
);
```

- Location Table

```
/* Creating Location table (primary key - location_id) */

CREATE table Location(
    location_id serial primary key,
    location varchar(50),
    city VARCHAR(50),
    state VARCHAR(50),
    address VARCHAR(100)
);
```

- Department Table

```
/* Creating Department table (primary key - dep_id) */

CREATE table Department(
    dep_id serial primary key,
    dep_name varchar(50)
);
```

DDL

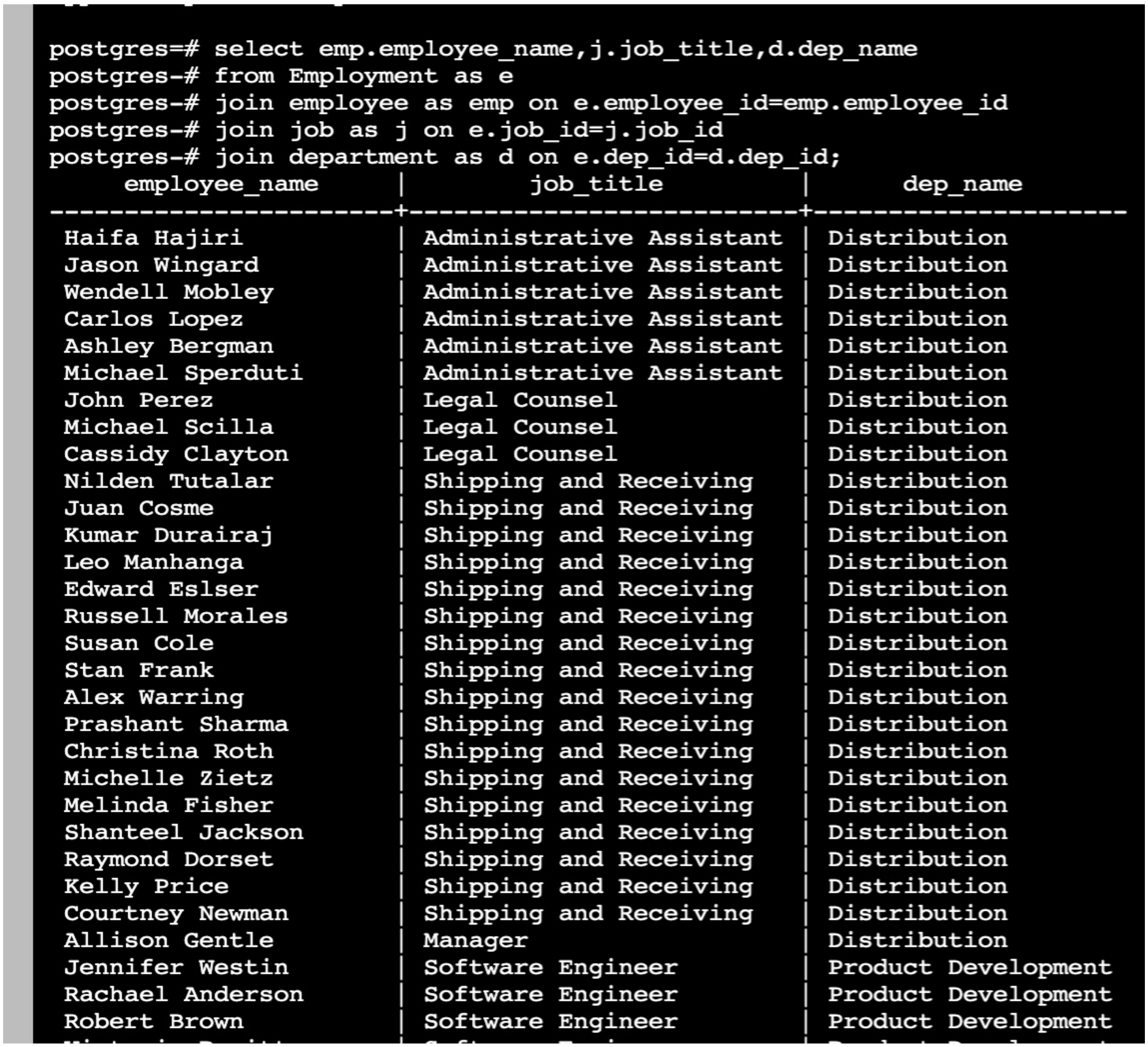
- Employment Table

```
/* Creating Employment table as there is
   many to many relationship across emp_id and
   job_id. So this table acts as pivot point */

CREATE table Employment(
    employee_id varchar(8) references Employee(employee_id),
    job_id int references Job(job_id),
    dep_id int references Department(dep_id),
    manager_id varchar(8) references Employee(employee_id),
    location_id int references Location(location_id),
    start_date date,
    end_date date,
    salary_id int references Salary(salary_id),
    primary key(employee_id, job_id)
);
```

CRUD

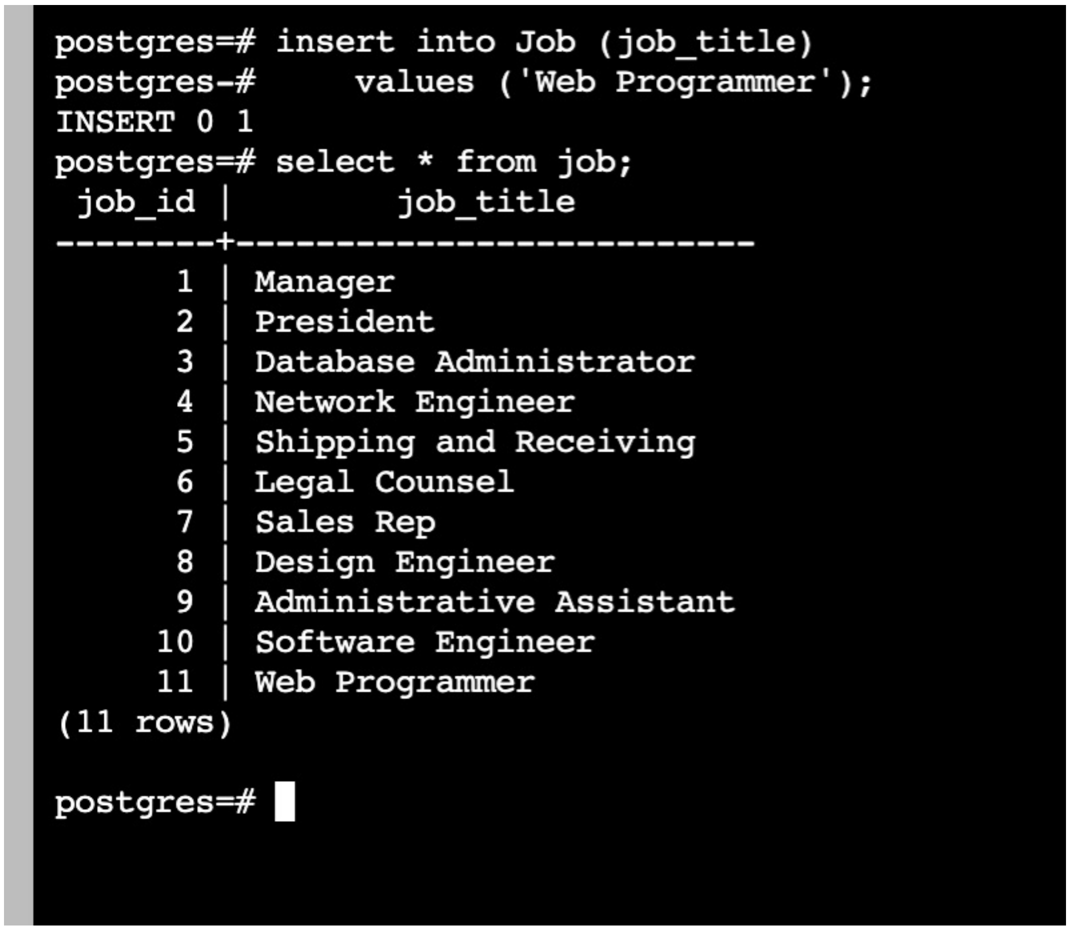
- **Question 1: Return a list of employees with Job Titles and Department Names**



```
postgres=# select emp.employee_name,j.job_title,d.dep_name
postgres-# from Employment as e
postgres-# join employee as emp on e.employee_id=emp.employee_id
postgres-# join job as j on e.job_id=j.job_id
postgres-# join department as d on e.dep_id=d.dep_id;
     employee_name       |        job_title          |       dep_name
-------------------------+---------------------------+---------------------
 Haifa Hajiri            | Administrative Assistant | Distribution
 Jason Wingard           | Administrative Assistant | Distribution
 Wendell Mobley          | Administrative Assistant | Distribution
 Carlos Lopez            | Administrative Assistant | Distribution
 Ashley Bergman          | Administrative Assistant | Distribution
 Michael Sperduti        | Administrative Assistant | Distribution
 John Perez              | Legal Counsel            | Distribution
 Michael Scilla          | Legal Counsel            | Distribution
 Cassidy Clayton         | Legal Counsel            | Distribution
 Nilden Tutalar          | Shipping and Receiving   | Distribution
 Juan Cosme              | Shipping and Receiving   | Distribution
 Kumar Durairaj          | Shipping and Receiving   | Distribution
 Leo Manhanga            | Shipping and Receiving   | Distribution
 Edward Eslser           | Shipping and Receiving   | Distribution
 Russell Morales         | Shipping and Receiving   | Distribution
 Susan Cole              | Shipping and Receiving   | Distribution
 Stan Frank              | Shipping and Receiving   | Distribution
 Alex Warring            | Shipping and Receiving   | Distribution
 Prashant Sharma         | Shipping and Receiving   | Distribution
 Christina Roth          | Shipping and Receiving   | Distribution
 Michelle Zietz          | Shipping and Receiving   | Distribution
 Melinda Fisher          | Shipping and Receiving   | Distribution
 Shanteel Jackson        | Shipping and Receiving   | Distribution
 Raymond Dorset          | Shipping and Receiving   | Distribution
 Kelly Price             | Shipping and Receiving   | Distribution
 Courtney Newman         | Shipping and Receiving   | Distribution
 Allison Gentle          | Manager                  | Distribution
 Jennifer Westin         | Software Engineer        | Product Development
 Rachael Anderson        | Software Engineer        | Product Development
 Robert Brown            | Software Engineer        | Product Development
```
]

CRUD

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

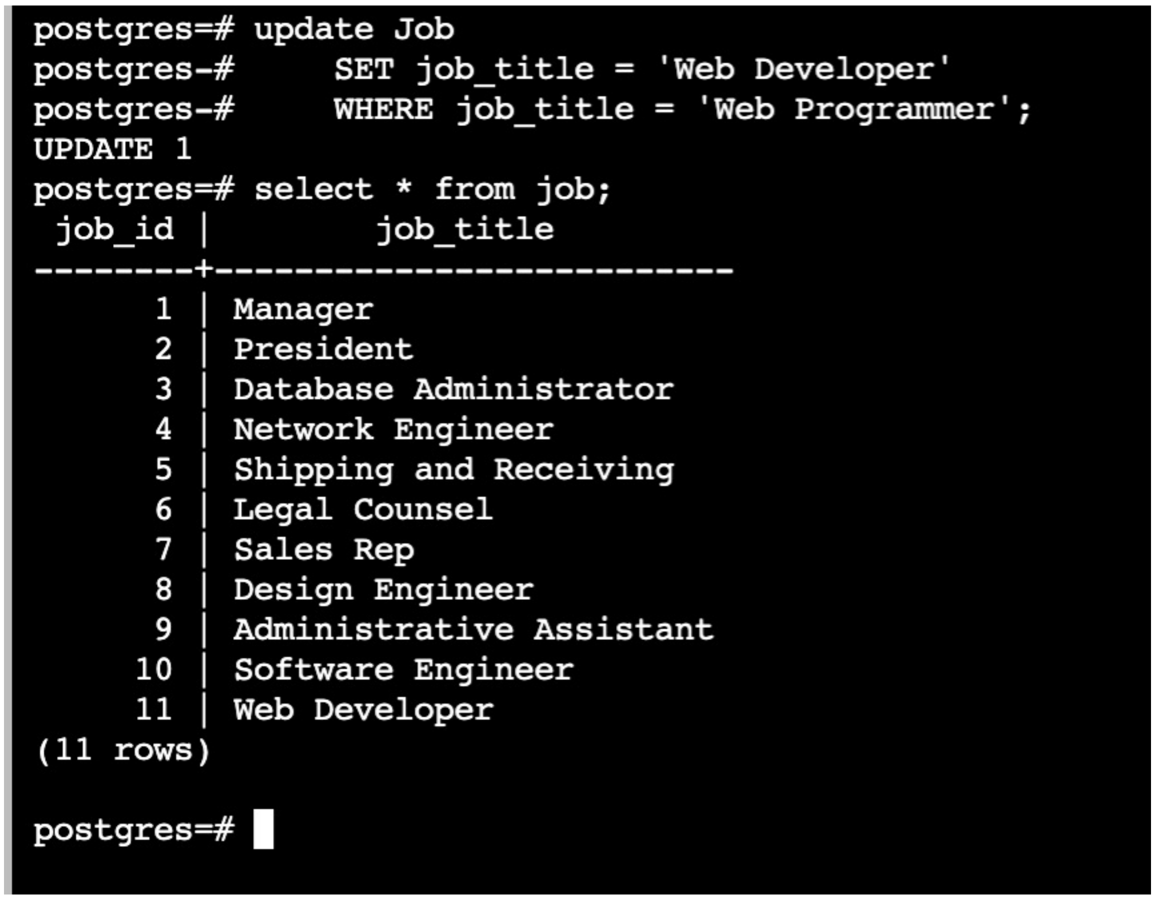


```
postgres=# insert into Job (job_title)
postgres-#     values ('Web Programmer');
INSERT 0 1
postgres=# select * from job;
 job_id |         job_title
--------+--------------------------
      1 | Manager
      2 | President
      3 | Database Administrator
      4 | Network Engineer
      5 | Shipping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

postgres=# 
```

CRUD

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

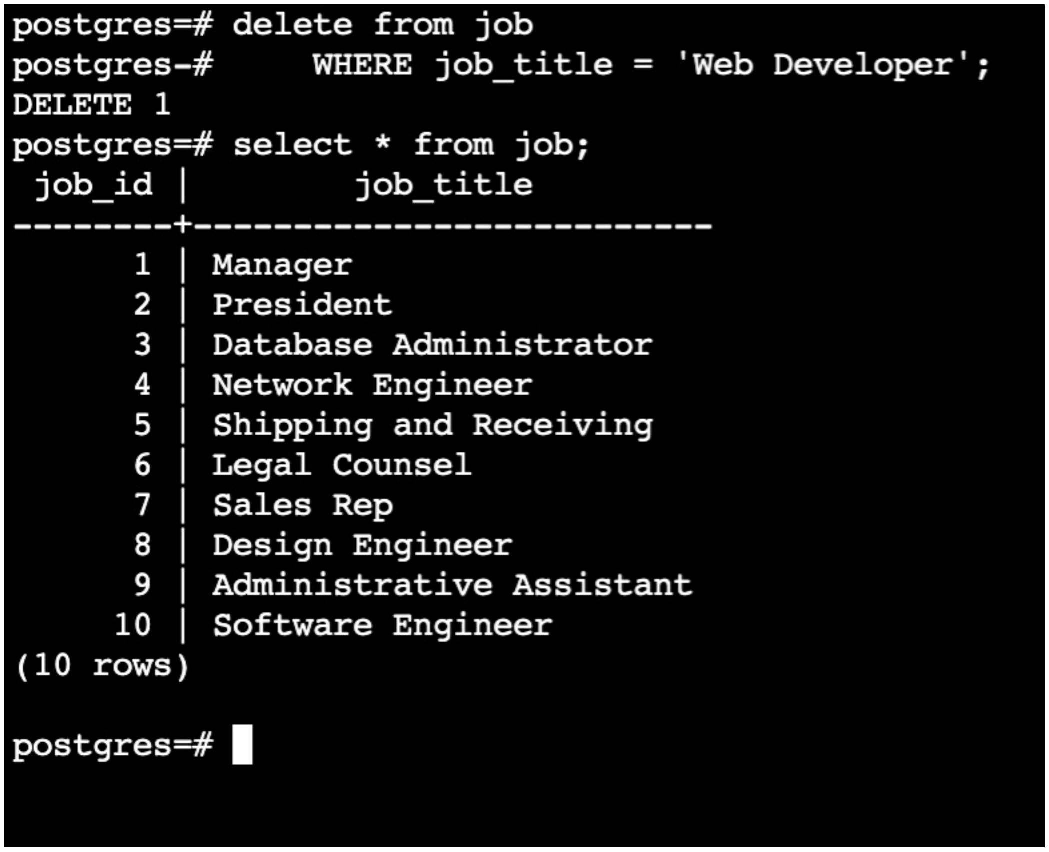


```
postgres=# update Job
postgres-#     SET job_title = 'Web Developer'
postgres-#     WHERE job_title = 'Web Programmer';
UPDATE 1
postgres=# select * from job;
 job_id |        job_title
--------+--------------------------
      1 | Manager
      2 | President
      3 | Database Administrator
   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

postgres=# 
```

CRUD

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

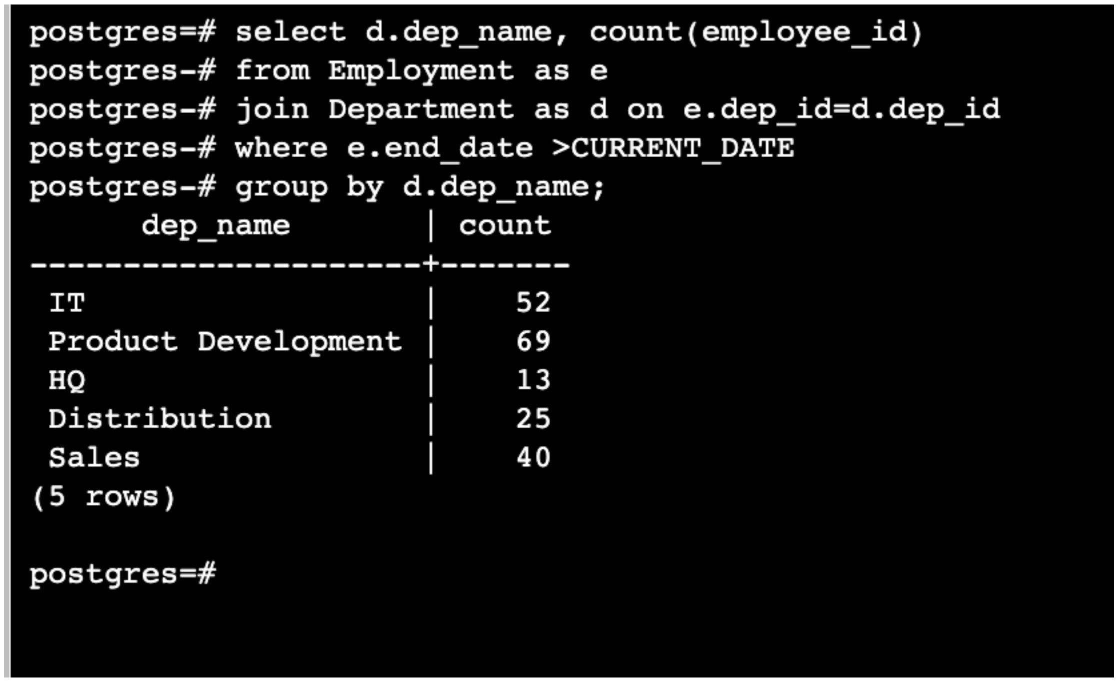


```
postgres=# delete from job
postgres-#     WHERE job_title = 'Web Developer';
DELETE 1
postgres=# select * from job;
 job_id |         job_title
--------+-------------------------
      1 | Manager
      2 | President
      3 | Database Administrator
      4 | Network Engineer
      5 | Shipping and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

postgres=# 
```

CRUD

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

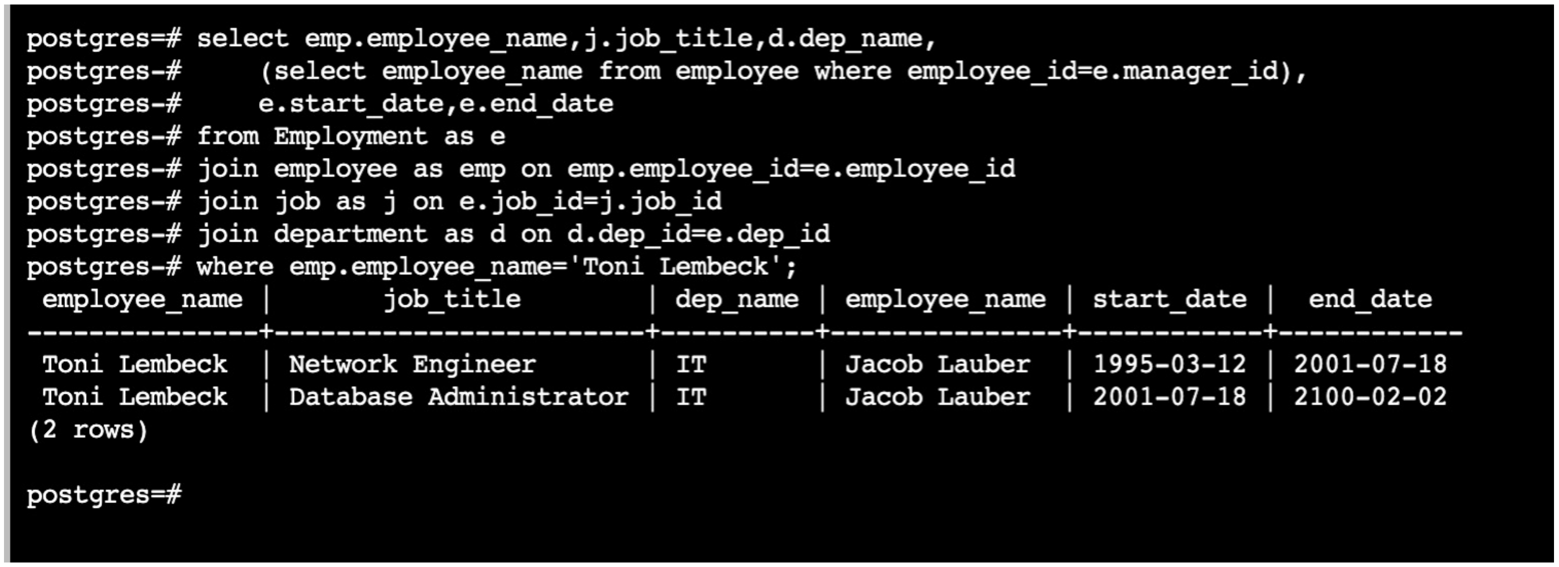


```
postgres=# select d.dep_name, count(employee_id)
postgres-# from Employment as e
postgres-# join Department as d on e.dep_id=d.dep_id
postgres-# where e.end_date >CURRENT_DATE
postgres-# group by d.dep_name;
      dep_name       | count
---------------------+-------
 IT                  |    52
 Product Development |    69
 HQ                  |    13
 Distribution        |    25
 Sales               |    40
(5 rows)

postgres=#
```

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 emp.employee_name,j.job_title,d.dep_name,
postgres-#     (select employee_name from employee where employee_id=e.manager_id),
postgres-#     e.start_date,e.end_date
postgres-# from Employment as e
postgres-# join employee as emp on emp.employee_id=e.employee_id
postgres-# join job as j on e.job_id=j.job_id
postgres-# join department as d on d.dep_id=e.dep_id
postgres-# where emp.employee_name='Toni Lembeck';
 employee_name |        job_title        | dep_name | employee_name | start_date |  end_date
---------------+-------------------------+----------+---------------+------------+------------
 Toni Lembeck  | Network Engineer        | IT       | Jacob Lauber  | 1995-03-12 | 2001-07-18
 Toni Lembeck  | Database Administrator  | IT       | Jacob Lauber  | 2001-07-18 | 2100-02-02
(2 rows)

postgres=#
```

CRUD

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

**** answer in a short paragraph, how you would apply table security to restrict access to employee salaries**

  - We can restrict employees from accessing the employee salaries by revoking employees access to the Salary table and that can be done at user level.
  - Create view on any table with some constraint like excluding salary column and give access to employee.

Step 4

Above and Beyond

(optional)

Step 4: Above and Beyond

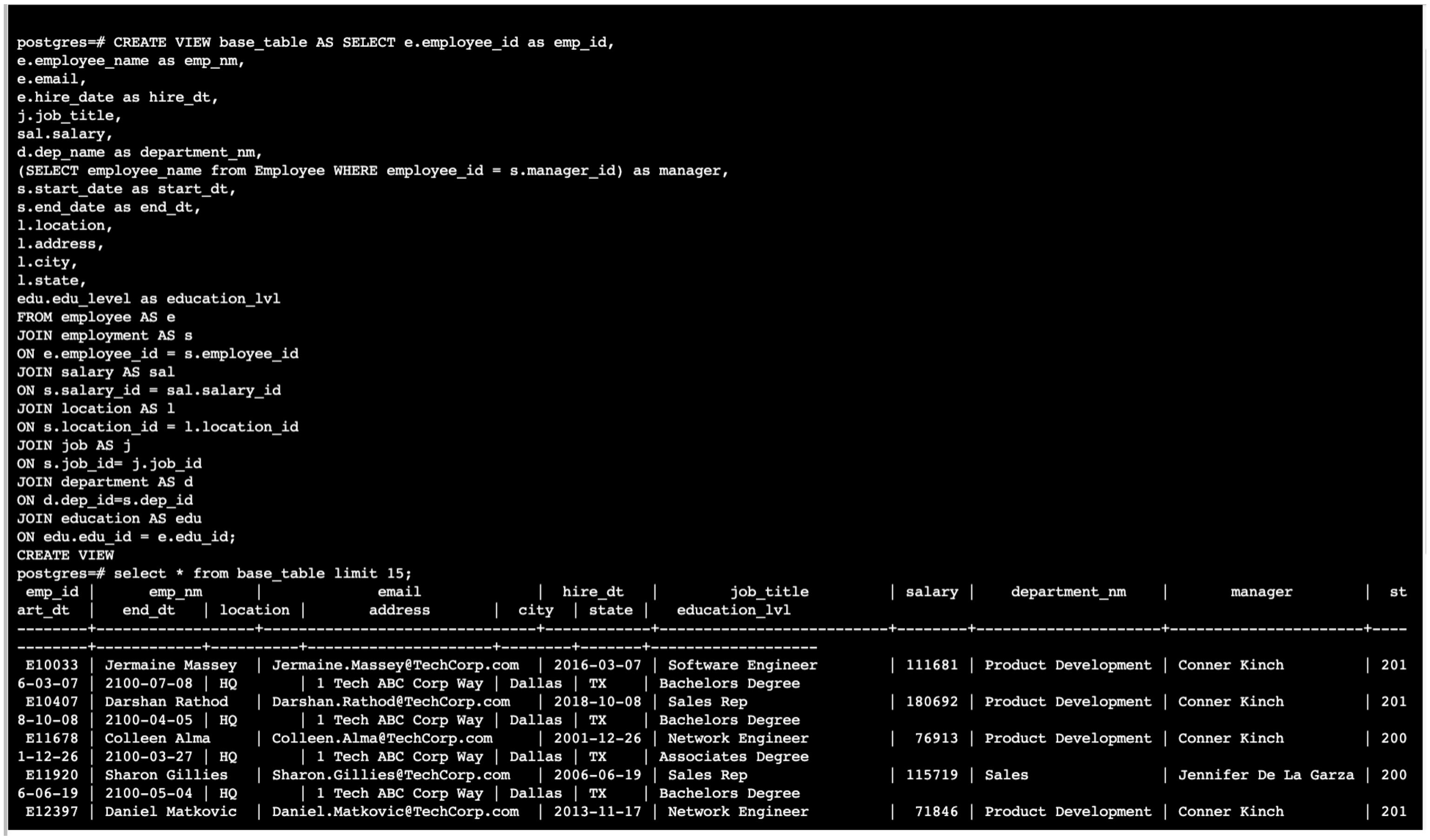
This last step is called Above and Beyond. In this step, I have proposed 3 challenges for you to complete, which are above and beyond the scope of the project. This is a chance to flex your coding muscles and show everyone how good you really are.

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

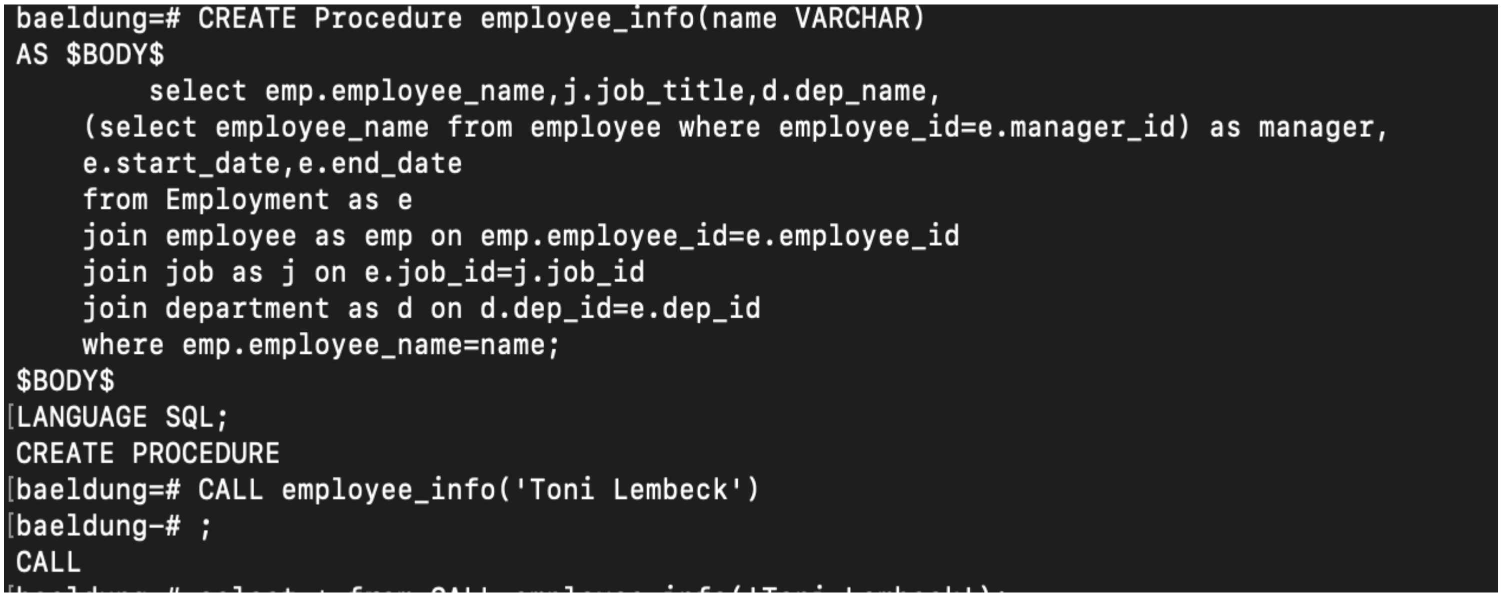
Create a view that returns all employee attributes; results should resemble initial Excel file



```
postgres=# CREATE VIEW base_table AS SELECT e.employee_id as emp_id,
e.employee_name as emp_nm,
e.email,
e.hire_date as hire_dt,
j.job_title,
sal.salary,
d.dep_name as department_nm,
(SELECT employee_name from Employee WHERE employee_id = s.manager_id) as manager,
s.start_date as start_dt,
s.end_date as end_dt,
l.location,
l.address,
l.city,
l.state,
edu.edu_level as education_lvl
FROM employee AS e
JOIN employment AS s
ON e.employee_id = s.employee_id
JOIN salary AS sal
ON s.salary_id = sal.salary_id
JOIN location AS l
ON s.location_id = l.location_id
JOIN job AS j
ON s.job_id= j.job_id
JOIN department AS d
ON d.dep_id=s.dep_id
JOIN education AS edu
ON edu.edu_id = e.edu_id;
CREATE VIEW
postgres=# select * from base_table limit 15;
 emp_id |      emp_nm      |             email              |  hire_dt   |         job_title          | salary |    department_nm     |       manager        | st
art_dt |   end_dt   | location |        address        |  city  | state |   education_lvl
--------+------------------+--------------------------------+------------+----------------------------+--------+----------------------+----------------------+----
--------+------------+----------+----------------------+--------+-------+-------------------
 E10033 | Jermaine Massey  | Jermaine.Massey@TechCorp.com   | 2016-03-07 | Software Engineer          | 111681 | Product Development | Conner Kinch         | 201
6-03-07 | 2100-07-08 | HQ       | 1 Tech ABC Corp Way | Dallas | TX    | Bachelors Degree
 E10407 | Darshan Rathod   | Darshan.Rathod@TechCorp.com    | 2018-10-08 | Sales Rep                  | 180692 | Product Development | Conner Kinch         | 201
8-10-08 | 2100-04-05 | HQ       | 1 Tech ABC Corp Way | Dallas | TX    | Bachelors Degree
 E11678 | Colleen Alma     | Colleen.Alma@TechCorp.com      | 2001-12-26 | Network Engineer           |  76913 | Product Development | Conner Kinch         | 200
1-12-26 | 2100-03-27 | HQ       | 1 Tech ABC Corp Way | Dallas | TX    | Associates Degree
 E11920 | Sharon Gillies   | Sharon.Gillies@TechCorp.com    | 2006-06-19 | Sales Rep                  | 115719 | Sales                | Jennifer De La Garza | 200
6-06-19 | 2100-05-04 | HQ       | 1 Tech ABC Corp Way | Dallas | TX    | Bachelors Degree
 E12397 | Daniel Matkovic  | Daniel.Matkovic@TechCorp.com   | 2013-11-17 | Network Engineer           |  71846 | Product Development | Conner Kinch         | 201
```

Standout Suggestion 2

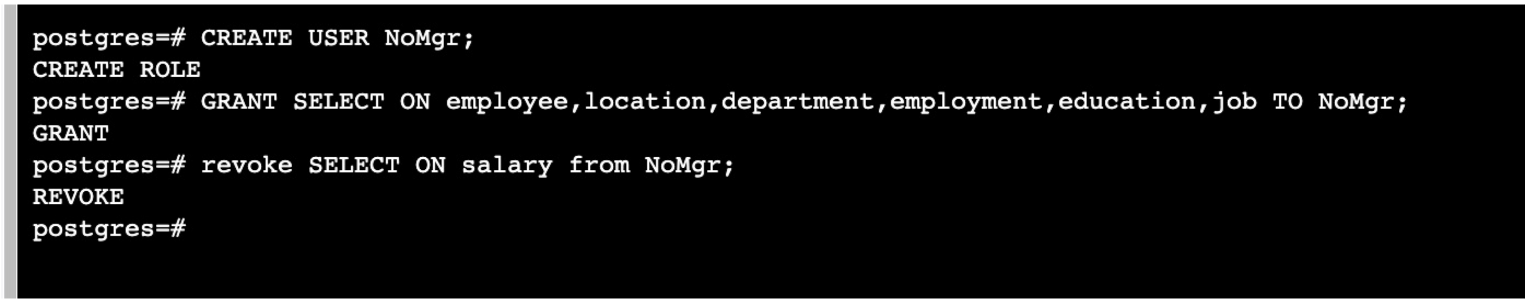
Create a stored procedure with parameters that returns current and past jobs (include employee name, job title, department, manager name, start and end date for position) when given an employee name.



```
baeldung=# CREATE Procedure employee_info(name VARCHAR)
AS $BODY$
        select emp.employee_name,j.job_title,d.dep_name,
    (select employee_name from employee where employee_id=e.manager_id) as manager,
    e.start_date,e.end_date
    from Employment as e
    join employee as emp on emp.employee_id=e.employee_id
    join job as j on e.job_id=j.job_id
    join department as d on d.dep_id=e.dep_id
    where emp.employee_name=name;
$BODY$
[LANGUAGE SQL;
CREATE PROCEDURE
[baeldung=# CALL employee_info('Toni Lembeck')
[baeldung-# ;
CALL
```

Standout Suggestion 3

Implement user security on the restricted salary attribute.



```
postgres=# CREATE USER NoMgr;
CREATE ROLE
postgres=# GRANT SELECT ON employee,location,department,employment,education,job TO NoMgr;
GRANT
postgres=# revoke SELECT ON salary from NoMgr;
REVOKE
postgres=#
```

Appendix

Additional Info

- Link to the Lucidchart:
  - [https://lucid.app/lucidchart/8ed5dd2c-9d5b-4ad7-9edc-3cdd93e30036/edit?view_items=kxR8VAtS9T33&invitationId=inv_3c168e59-931c-4660-a388-bedc15f92438]