Tech ABC Corp - HR Database

[Nhan V. Nguyen 27/04/2024]

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:**

  To maintain data integrity and enhance data security, implementing row-level security measures is essential. This ensures that only authorized personnel, such as management and HR employees, have access to sensitive data like employee salaries, thereby safeguarding the integrity and confidentiality of the data.

- **Describe current data management solution:**

  All the information has been stored within an Excel file.

- **Describe current data available:**

  Employee id, employee' s, name, email, hire date, job title, salary, department, manager name, start date, end date, location, address, city, state, education level

- **Additional data requests:**

  They request that this data be maintained for at least 7 years. Additionally, they express a desire to establish connectivity with the payroll department's system in the future.

- **Who will own/manage data**

  The management and the HR employees

- **Who will have access to database**

  Every employee with a domain login is granted read access to the database; however, they are restricted from accessing salary information. Conversely, management and HR employees are provided with both read and write access, along with access to

Data Architect Business Requirement

- **Estimated size of database**

  The dataset comprises 206 rows and 15 columns.

- **Estimated annual growth**

  The growth rate is projected to be 20% annually for the next 5 years.

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

  Restricted access is limited to employees who are not managers or HR personnel.

Data Architect Technical Requirement

- **Justification for the new database**

  Data integrity and security are paramount.

- **Database objects**

  Table: education_level, employee, employment, manager, location, department, job, salary
  View: 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: Every employee has access, but access to salary information is restricted to management and HR employees.

- **Scalability**

  Replication

- **Flexibility**

  A direct feed could prove invaluable in the future for seamlessly connecting the current database with the payroll system.

- **Storage & retention**

  Storage (disk or in-memory): disk

  Retention: 7 years

- **Backup**

  A full backup will be performed weekly, supplemented by daily interval backups.

Step 2

Relational Database Design

Step 2: Relational Database Design

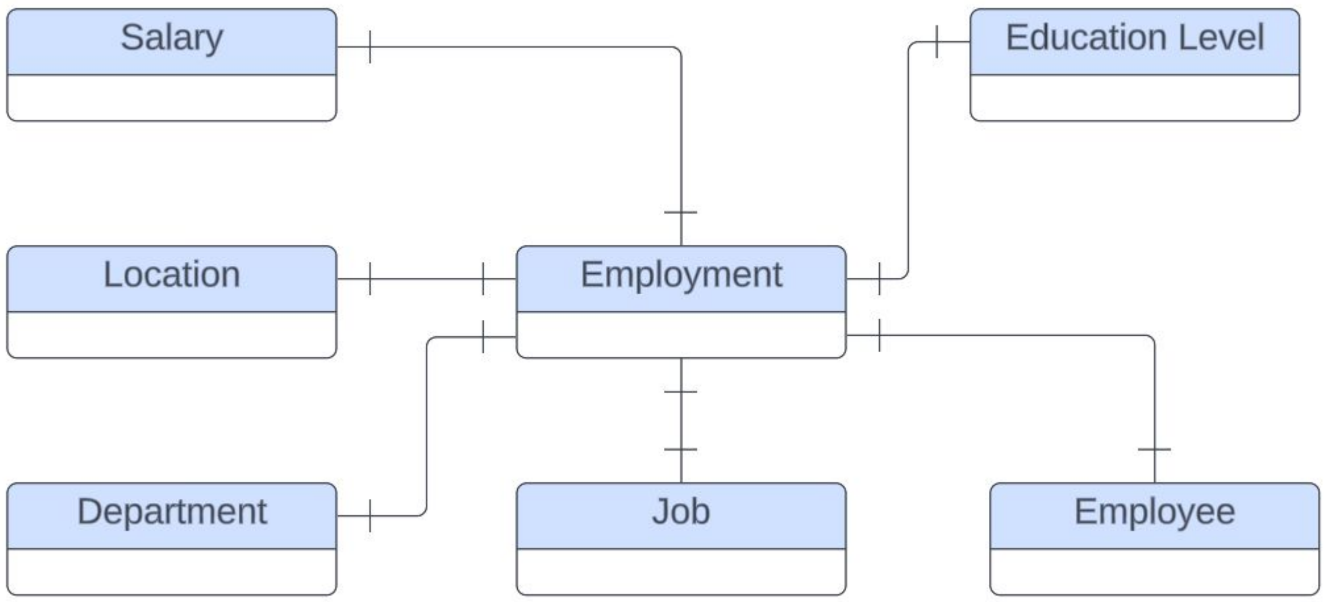
This step is where you will go through the process of designing a new database for Tech ABC Corp's HR department. Using the dataset provided, along with the requirements gathered in step one, you are going to develop a relational database set to the 3NF.

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.

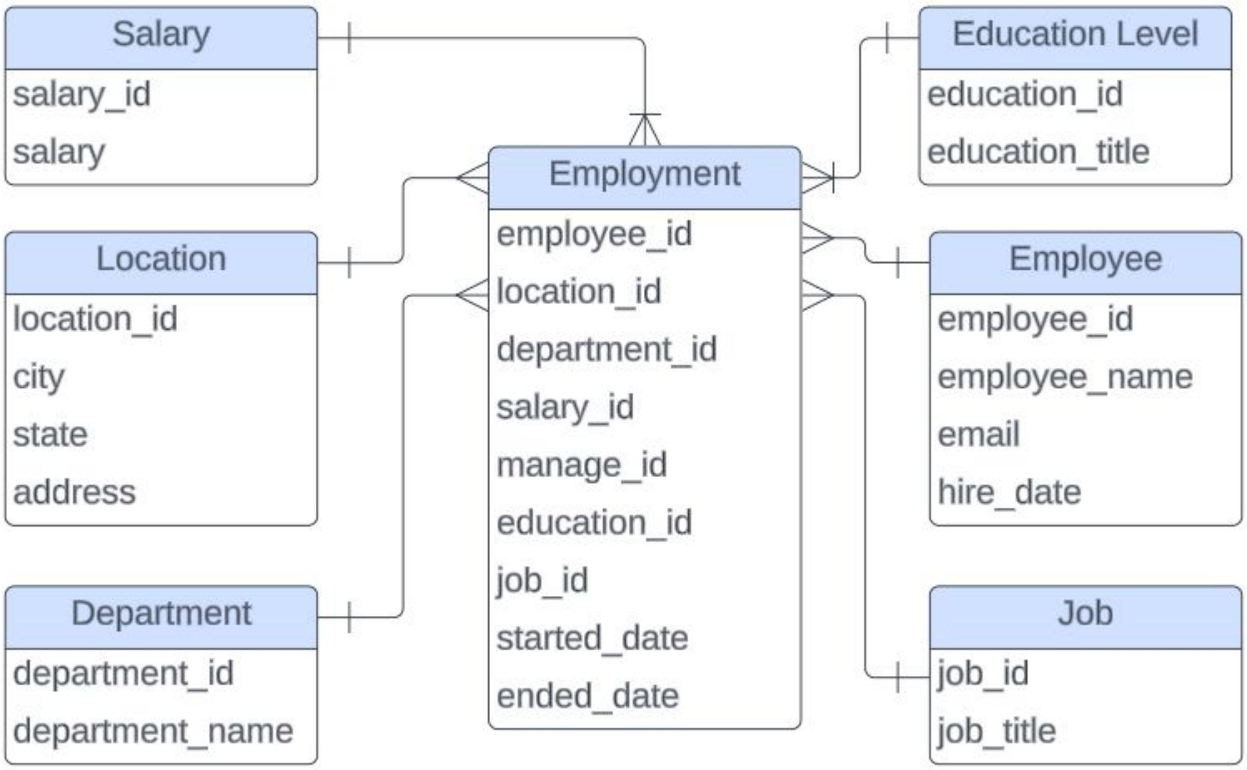
ERD

- **Conceptual**



ERD

- **Logical**



Salary
salary_id
salary

Location
location_id
city
state
address

Department
department_id
department_name

Employment
employee_id
location_id
department_id
salary_id
manage_id
education_id
job_id
started_date
ended_date

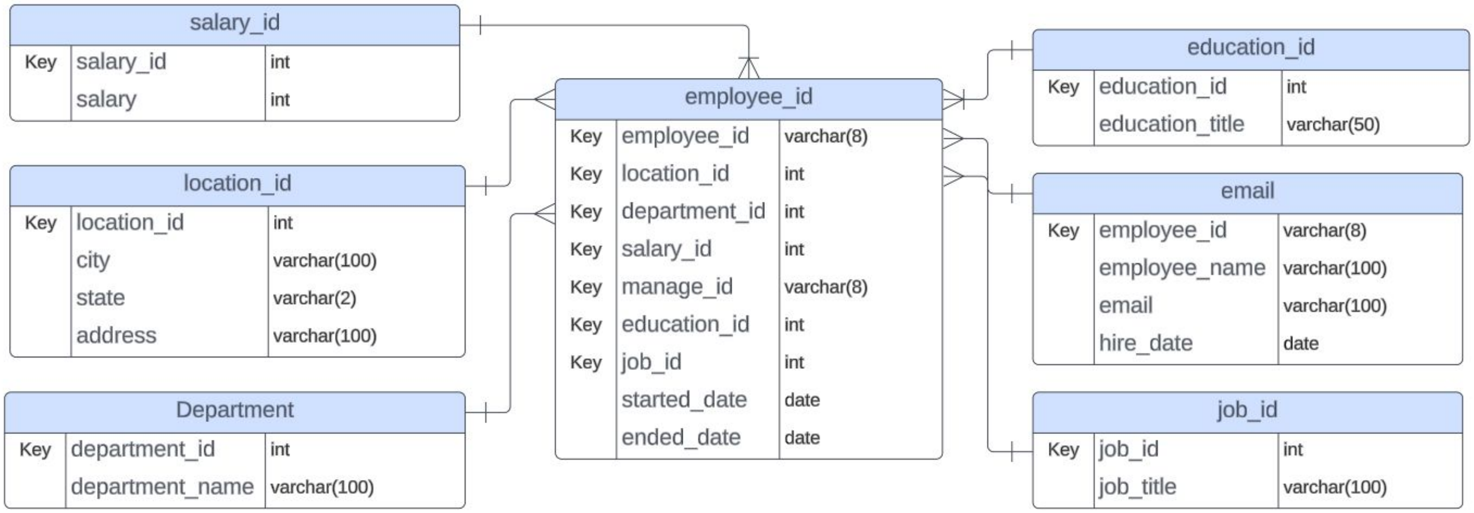
Education Level
education_id
education_title

Employee
employee_id
employee_name
email
hire_date

Job
job_id
job_title

ERD

- **Physical**



salary_id		
Key	salary_id	int
	salary	int

location_id		
Key	location_id	int
	city	varchar(100)
	state	varchar(2)
	address	varchar(100)

Department		
Key	department_id	int
	department_name	varchar(100)

employee_id		
Key	employee_id	varchar(8)
Key	location_id	int
Key	department_id	int
Key	salary_id	int
Key	manage_id	varchar(8)
Key	education_id	int
Key	job_id	int
	started_date	date
	ended_date	date

education_id		
Key	education_id	int
	education_title	varchar(50)

email		
Key	employee_id	varchar(8)
	employee_name	varchar(100)
	email	varchar(100)
	hire_date	date

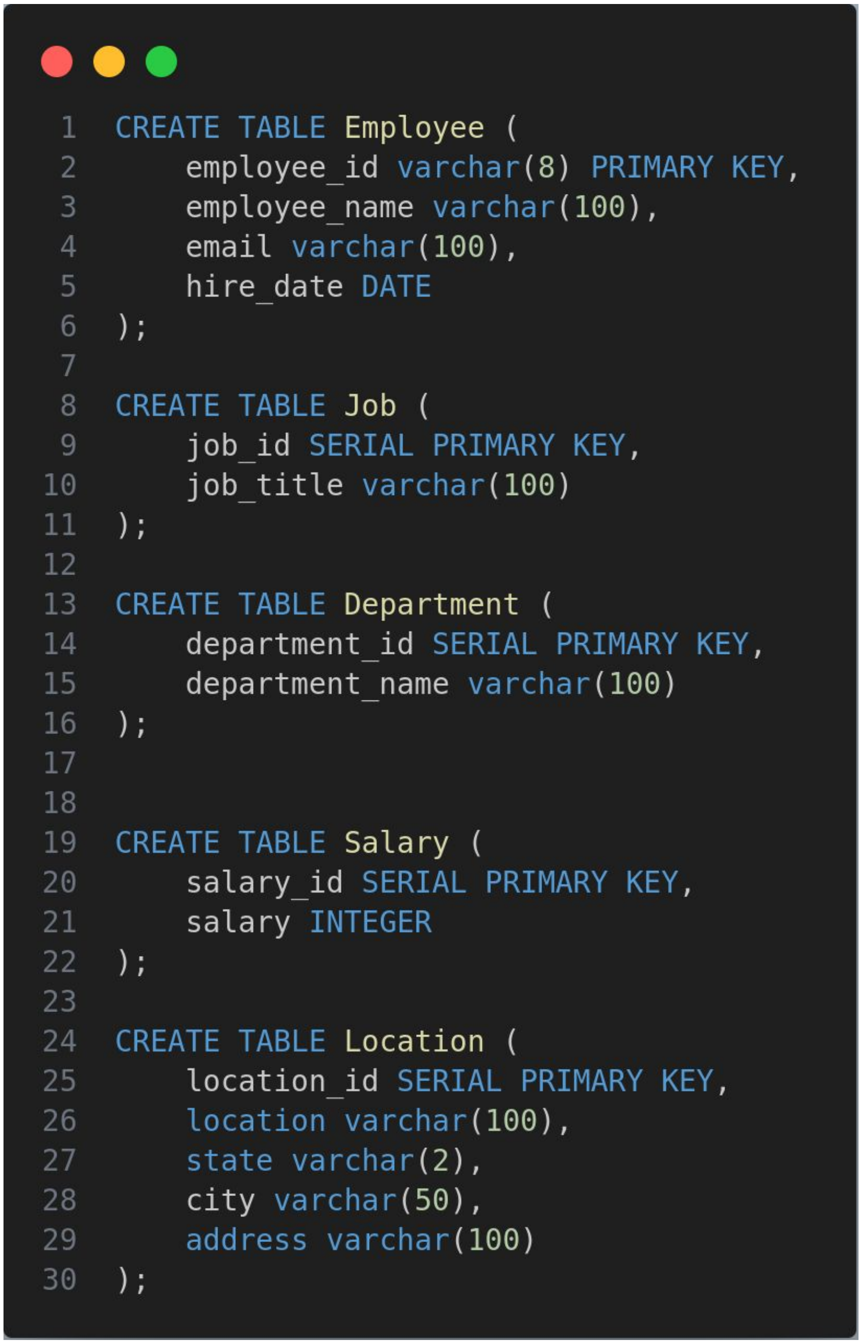
job_id		
Key	job_id	int
	job_title	varchar(100)

Step 3

Create A Physical Database

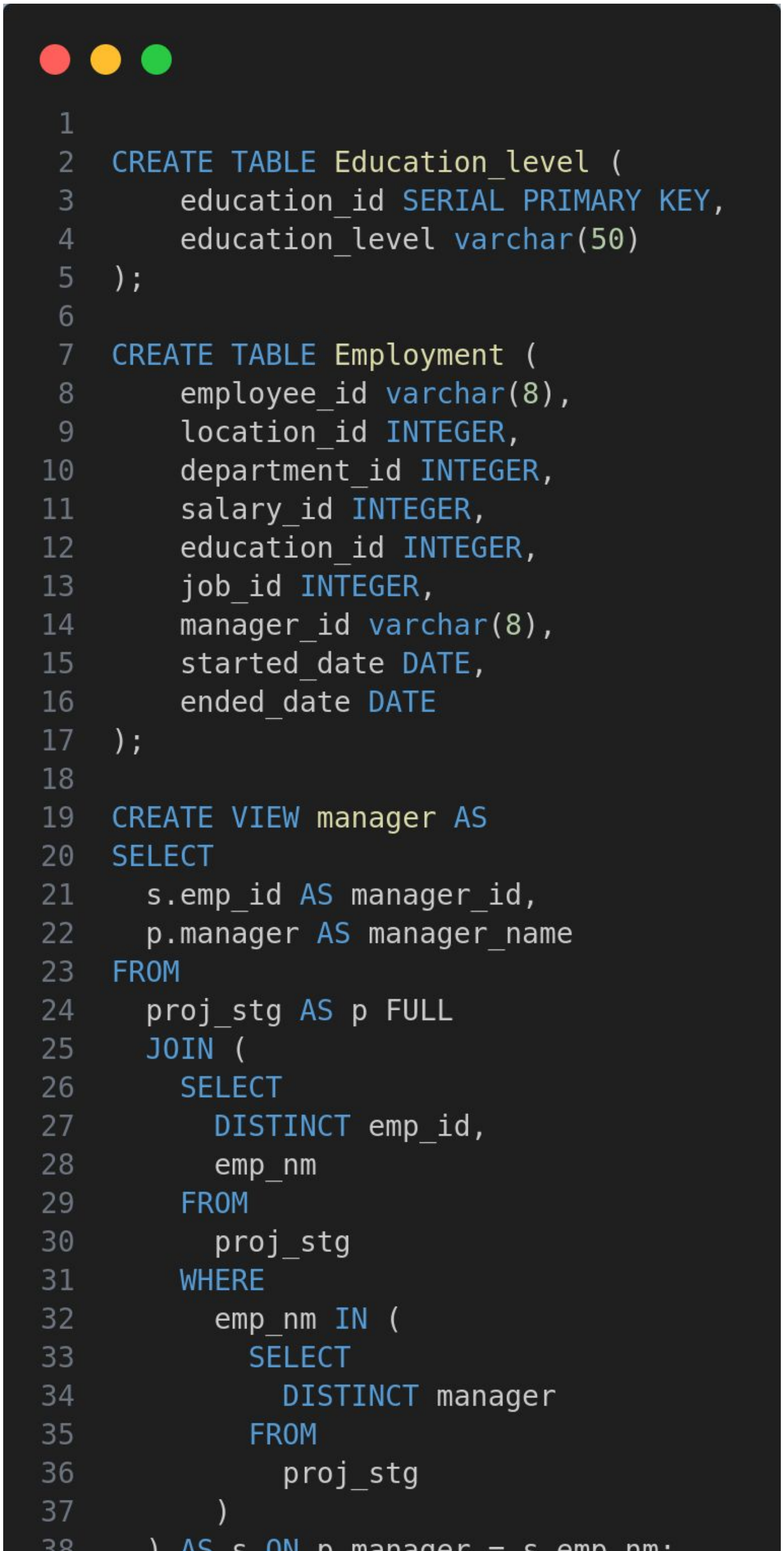
DDL

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



```sql
CREATE TABLE Employee (
    employee_id varchar(8) PRIMARY KEY,
    employee_name varchar(100),
    email varchar(100),
    hire_date DATE
);

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

CREATE TABLE Department (
    department_id SERIAL PRIMARY KEY,
    department_name varchar(100)
);


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

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

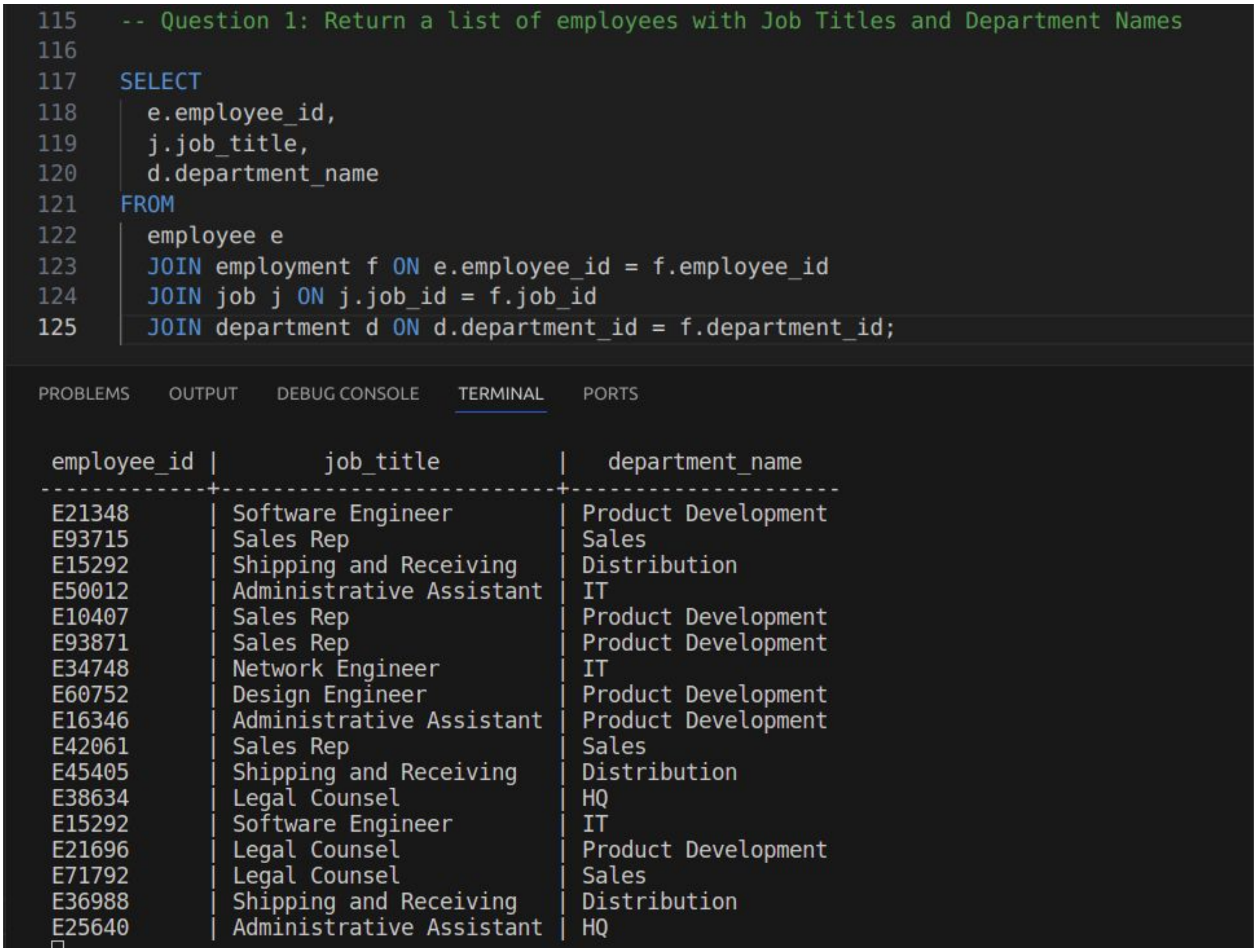
DDL



```sql
CREATE TABLE Education_level (
    education_id SERIAL PRIMARY KEY,
    education_level varchar(50)
);

CREATE TABLE Employment (
    employee_id varchar(8),
    location_id INTEGER,
    department_id INTEGER,
    salary_id INTEGER,
    education_id INTEGER,
    job_id INTEGER,
    manager_id varchar(8),
    started_date DATE,
    ended_date 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**

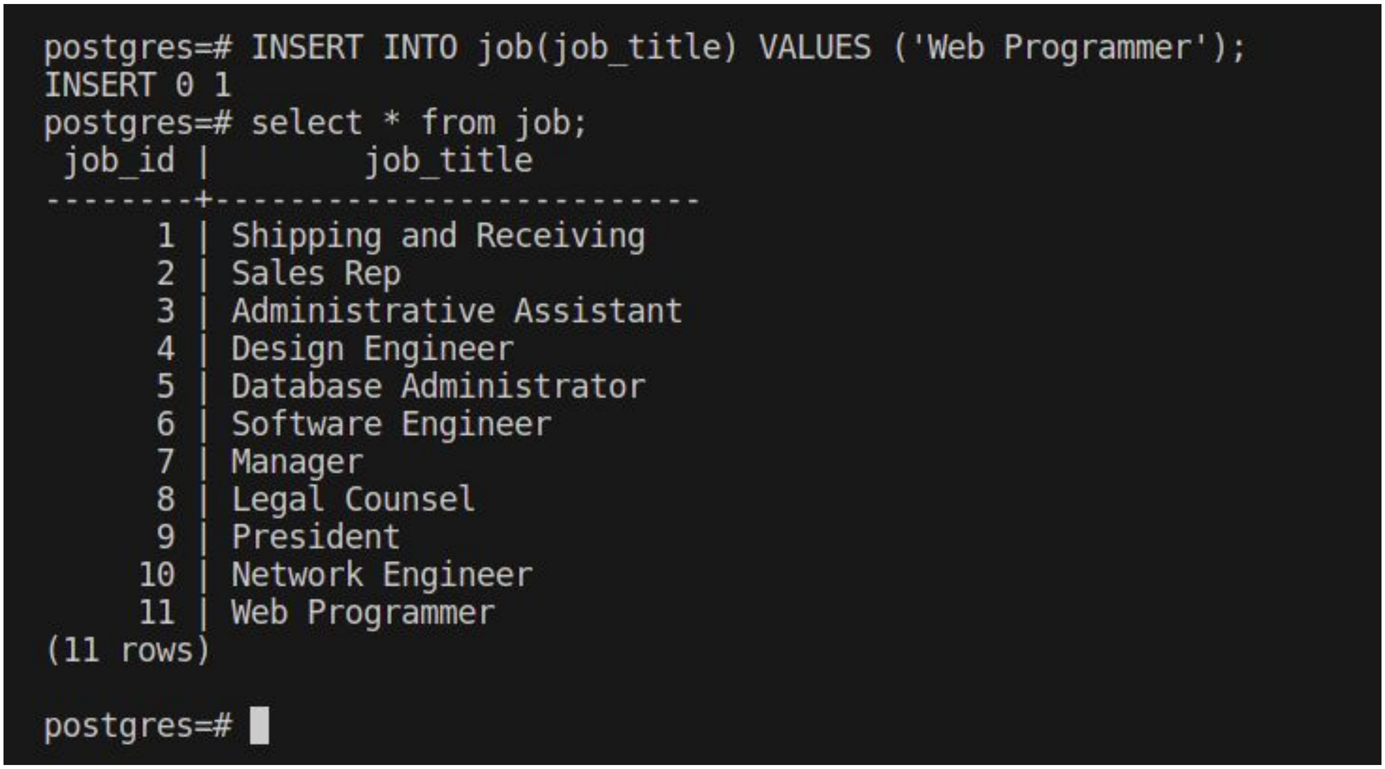


```sql
-- Question 1: Return a list of employees with Job Titles and Department Names

SELECT
  e.employee_id,
  j.job_title,
  d.department_name
FROM
  employee e
  JOIN employment f ON e.employee_id = f.employee_id
  JOIN job j ON j.job_id = f.job_id
  JOIN department d ON d.department_id = f.department_id;
```

```
 employee_id |        job_title         |   department_name
-------------+--------------------------+---------------------
 E21348      | Software Engineer        | Product Development
 E93715      | Sales Rep                | Sales
 E15292      | Shipping and Receiving   | Distribution
 E50012      | Administrative Assistant | IT
 E10407      | Sales Rep                | Product Development
 E93871      | Sales Rep                | Product Development
 E34748      | Network Engineer         | IT
 E60752      | Design Engineer          | Product Development
 E16346      | Administrative Assistant | Product Development
 E42061      | Sales Rep                | Sales
 E45405      | Shipping and Receiving   | Distribution
 E38634      | Legal Counsel            | HQ
 E15292      | Software Engineer        | IT
 E21696      | Legal Counsel            | Product Development
 E71792      | Legal Counsel            | Sales
 E36988      | Shipping and Receiving   | Distribution
 E25640      | Administrative Assistant | HQ
```

CRUD

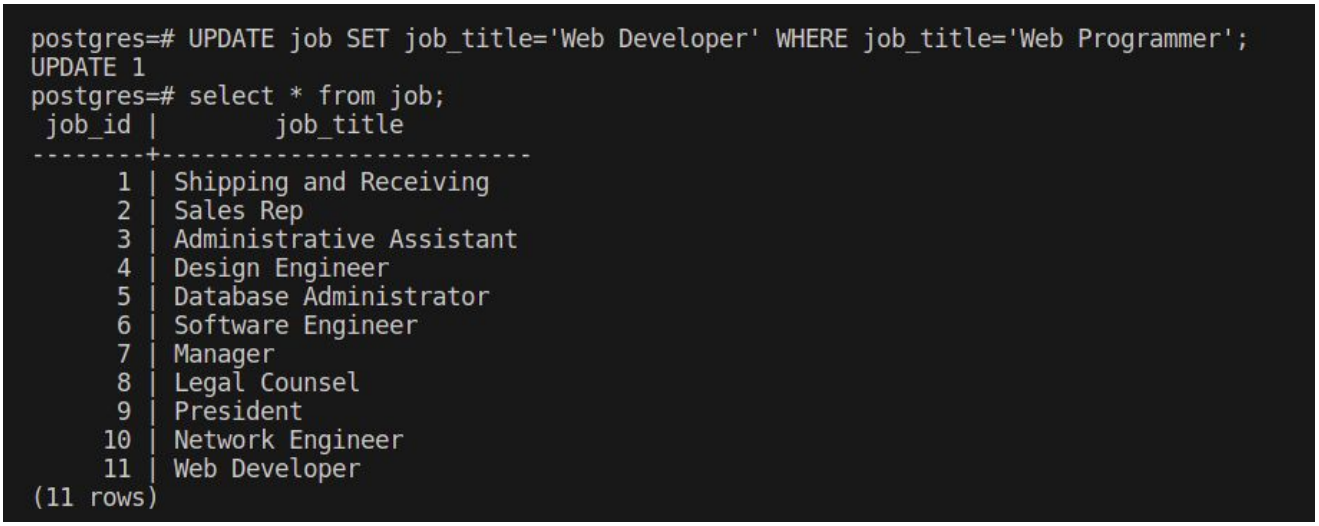
- **Question 2: Insert Web Programmer as a new job title**



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

postgres=# █
```

CRUD

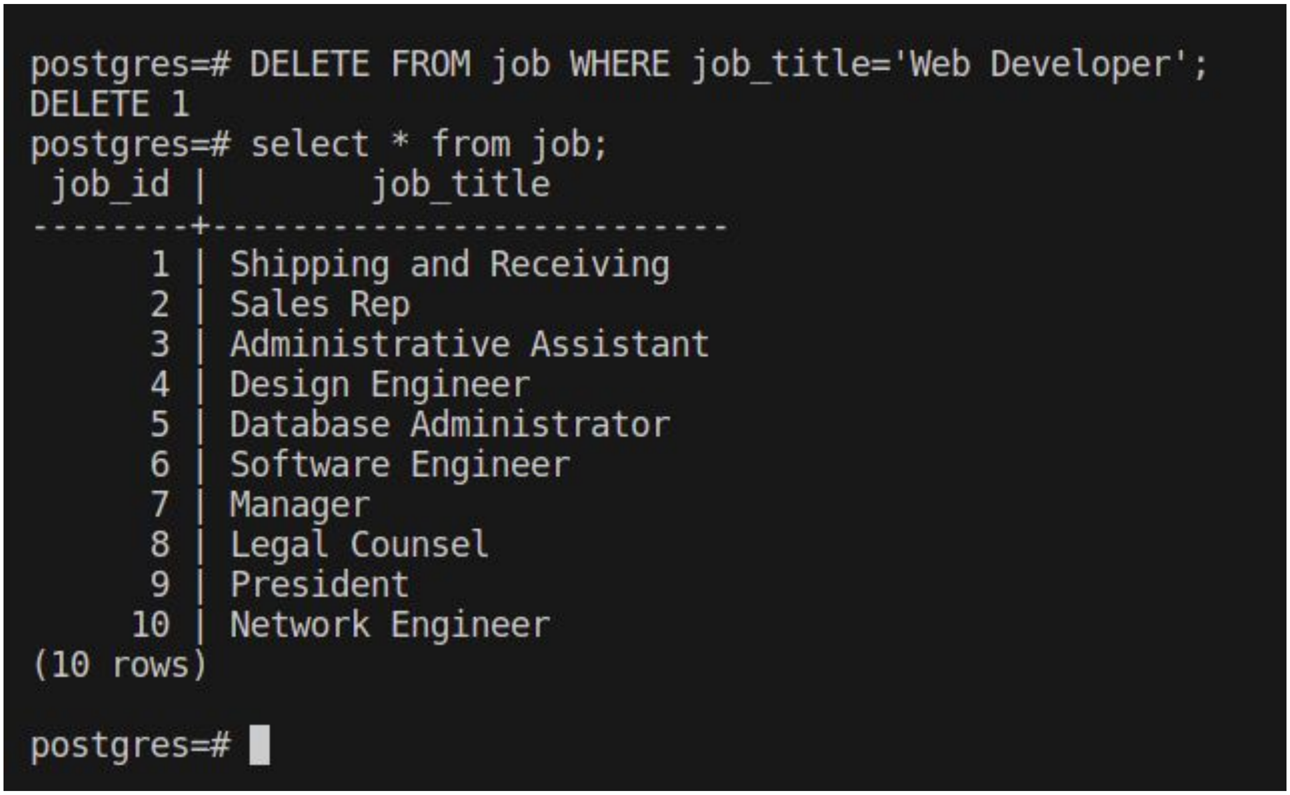
- **Question 3: Correct the job title from web programmer to web developer**



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

CRUD

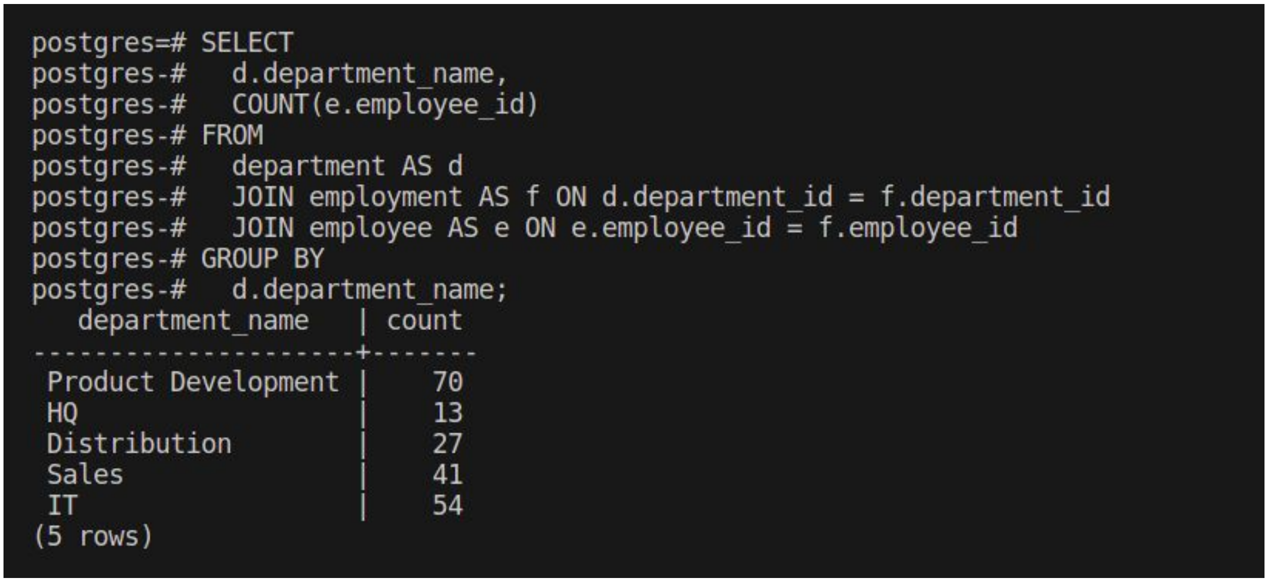
- **Question 4: Delete the job title Web Developer from the database**



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

postgres=# █
```

CRUD

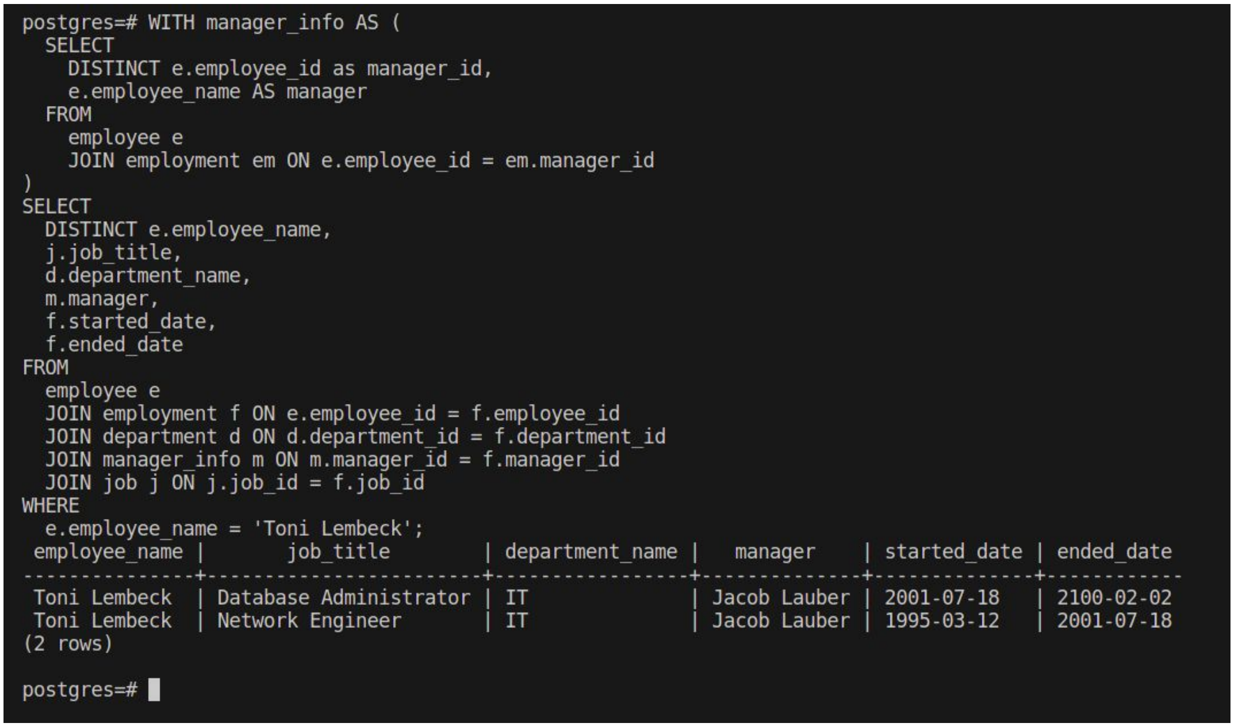
- **Question 5: How many employees are in each department?**



```
postgres=# SELECT
postgres-#   d.department_name,
postgres-#   COUNT(e.employee_id)
postgres-# FROM
postgres-#   department AS d
postgres-#   JOIN employment AS f ON d.department_id = f.department_id
postgres-#   JOIN employee AS e ON e.employee_id = f.employee_id
postgres-# GROUP BY
postgres-#   d.department_name;
   department_name   | count
---------------------+-------
 Product Development |    70
 HQ                  |    13
 Distribution        |    27
 Sales               |    41
 IT                  |    54
(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=# WITH manager_info AS (
  SELECT
    DISTINCT e.employee_id as manager_id,
    e.employee_name AS manager
  FROM
    employee e
    JOIN employment em ON e.employee_id = em.manager_id
)
SELECT
  DISTINCT e.employee_name,
  j.job_title,
  d.department_name,
  m.manager,
  f.started_date,
  f.ended_date
FROM
  employee e
  JOIN employment f ON e.employee_id = f.employee_id
  JOIN department d ON d.department_id = f.department_id
  JOIN manager_info m ON m.manager_id = f.manager_id
  JOIN job j ON j.job_id = f.job_id
WHERE
  e.employee_name = 'Toni Lembeck';
 employee_name |       job_title        | department_name |   manager    | started_date | ended_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)

postgres=# 
```

CRUD

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

->To restrict access to employee salaries, I believe the most effective approach is to implement row-level security. This method allows granting access to the salary table exclusively to management and HR personnel.

Appendix