Tech ABC Corp - HR Database

[Daniel Freitas - 09/09/2022]

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 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:

The business partner is requesting a new physical database to replace the current solution of an Excel spreadsheet. As of today, the Excel spreadsheet does not meet the needs of scalability: multiple regions manipulating the same data leading to lack of data integrity; and security: not restricting sensitive information such as salary that the HR data requires.

With this new database all the concerns listed above will be addressed along with the need of having a platform that can accomodate data growth.

The DB will be OLTP because it fits the business partner purpose with no reporting usage.

- ## Describe current data management solution:

The current data management solution is a single Excel spreadsheet with 15 columns that contains all the HR data of the company. The file is shared across the company and it has sensitive information in it.

- ## Describe current data available:

The current data available is a single spreadsheet with 15 columns and 206 records. The columns are: employee id, employee name, e-mail, hire date, job title, salary, department, manager, start date, end date, location, address, city, state, education level.

- ## Additional data requests:

The additional data can be interpreted on two ways:

1. The growth of the database in terms of records due to the number of hires of the company every year.
2. The intent of connecting to the payroll system which has paid time off and employee attendance information.

- ## Who will own/manage data

The owner of the data will be HR and management level employees, which consist of 10% of the users of the database.

- ## Who will have access to database

The database will have two types of users:

1. **Regular employees** - Read access. The access will be through user and password login and sensitive data, such as salary, will not be accessible for those users.
2. **HR and management level employees** - Read and Write access. Those users will have unlimited access to the database.

Data Architect Business Requirement

- ## Estimated size of database

The database will consist of 7 tables.

Estimation for each table:

- Employee: ~200 rows, 7 columns
- Job: ~200 rows, 7 columns
- Geography: 6 rows, 5 columns
- Education_level: 10 rows, 2 columns
- Job_title: 10 rows, 2 columns
- Department: 20 rows, 2 columns
- Salary: ~200 rows, 2 columns

Given that, the estimated size is > ~1000 records for this year.

- ## Estimated annual growth

The annual growth expected by the business partner is 20% a year.

Given the current data is 200 records for the tables that vary the most with new entries, in one year the data will increase 40 records on each table job, employee and salary. Total of 120 records.

- ## Is any of the data sensitive/restricted

The business partner states that the **salary** information is sensitive and the **regular employees** must **not** have access to that information.

Data Architect Technical Requirement

- ## Justification for the new database

1. **Data integrity:** the new database will provide data integrity since it relies on ACID principles and this is needed given that multiple regions are manipulating the same data at the same time.
2. **Data access management (security):** the DBMS will provide security aspects that the spreadsheet does not offer and this is also needed because there is sensitive information in the spreadsheet that cannot be visible for all the users.

- ## Database objects

Tables of the database:

- Employee - employee personal information
- Salary - sensitive information about salary and the access will be restricted
- Job_title - job titles of the employees in the company
- Job - historical data of all current and past jobs
- Geography - location of the workplace
- Department - department names within the company

- ## Data ingestion

Since the current storage solution is an Excel spreadsheet, in other words a flat file, based on the IT Best Practices standards, the ETL approach should be chosen.

Data Architect Technical Requirement

- ## Data governance (Ownership and User access)

  Ownership: HR and management level employees

  User Access:
  Full access - HR and management level employees,
  Restricted access (salary not visible) - All regular employees

- ## Scalability

  Shard is not needed for this database as the input data process will not be massive.

  Replication is needed to ensure scalability given that 90% of the usage is read access and the users are spread on different locations.

- ## Flexibility

  The future integration with the payroll department system is envisioned. Therefore, we need to build our solution with **standards** and **structure** that make this integration smooth in the future. For example, we might share the same employee id on both systems to be able to join them together on a simple way.

- ## Storage & retention

  Storage (disk or in-memory): Standard partition of 1GB as it will not increase 10k rows in the next year. The data should be stored in disk as no high level computation will be performed.

  Retention: The data has to be kept for at least 7 years required by federal regulation.

- ## Backup

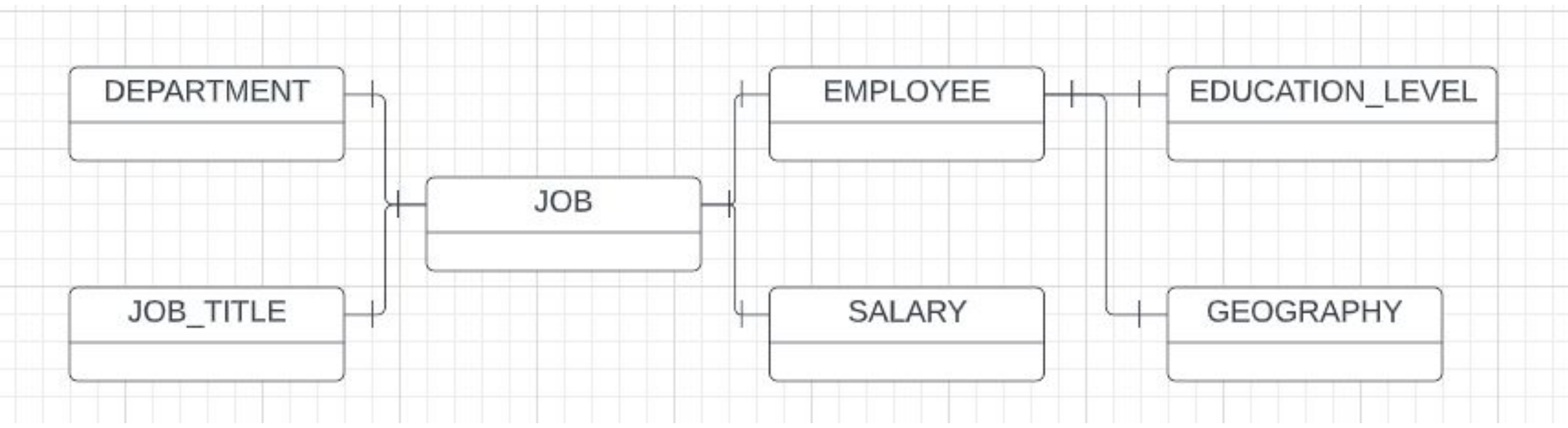
  Based on the IT Best Practices, for Critical Business Data, the backup schedule is full backup 1x per week, incremental backup daily.

Step 2

Relational Database Design

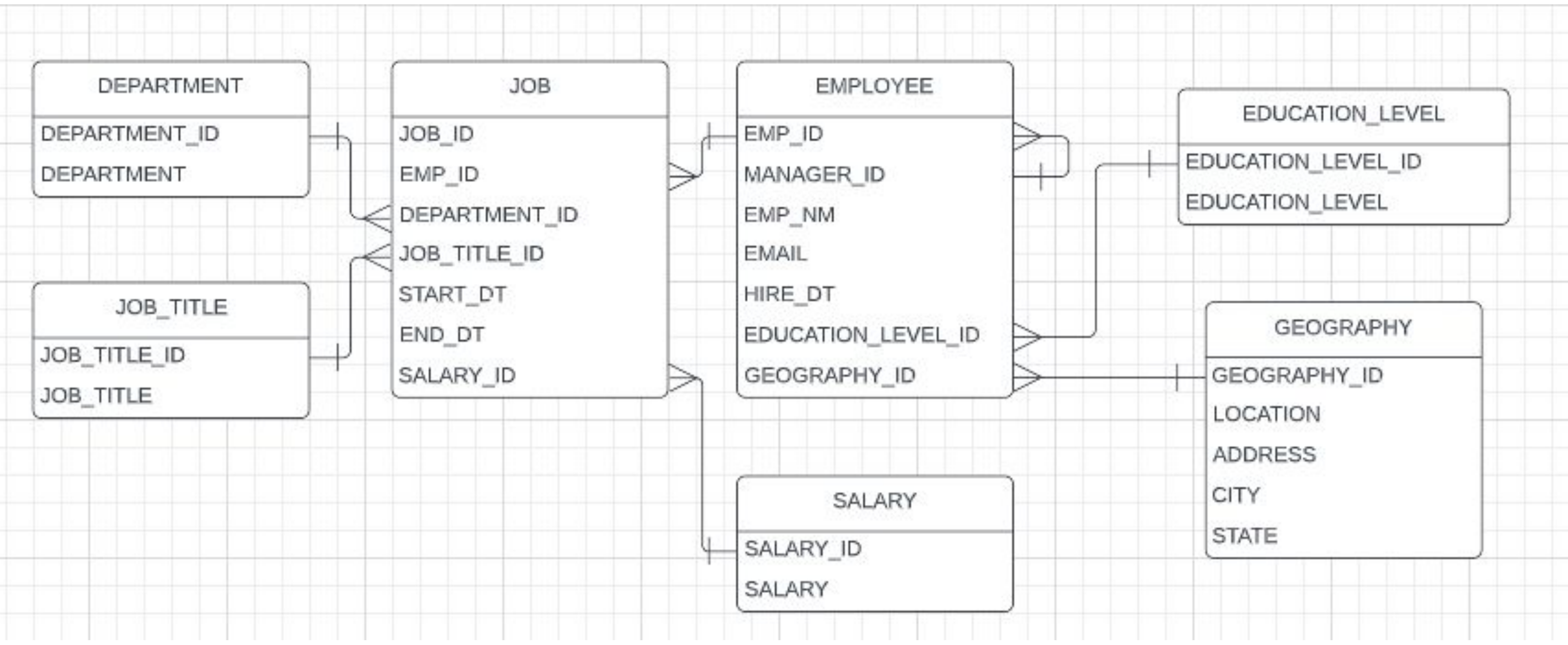
ERD

- **Conceptual**



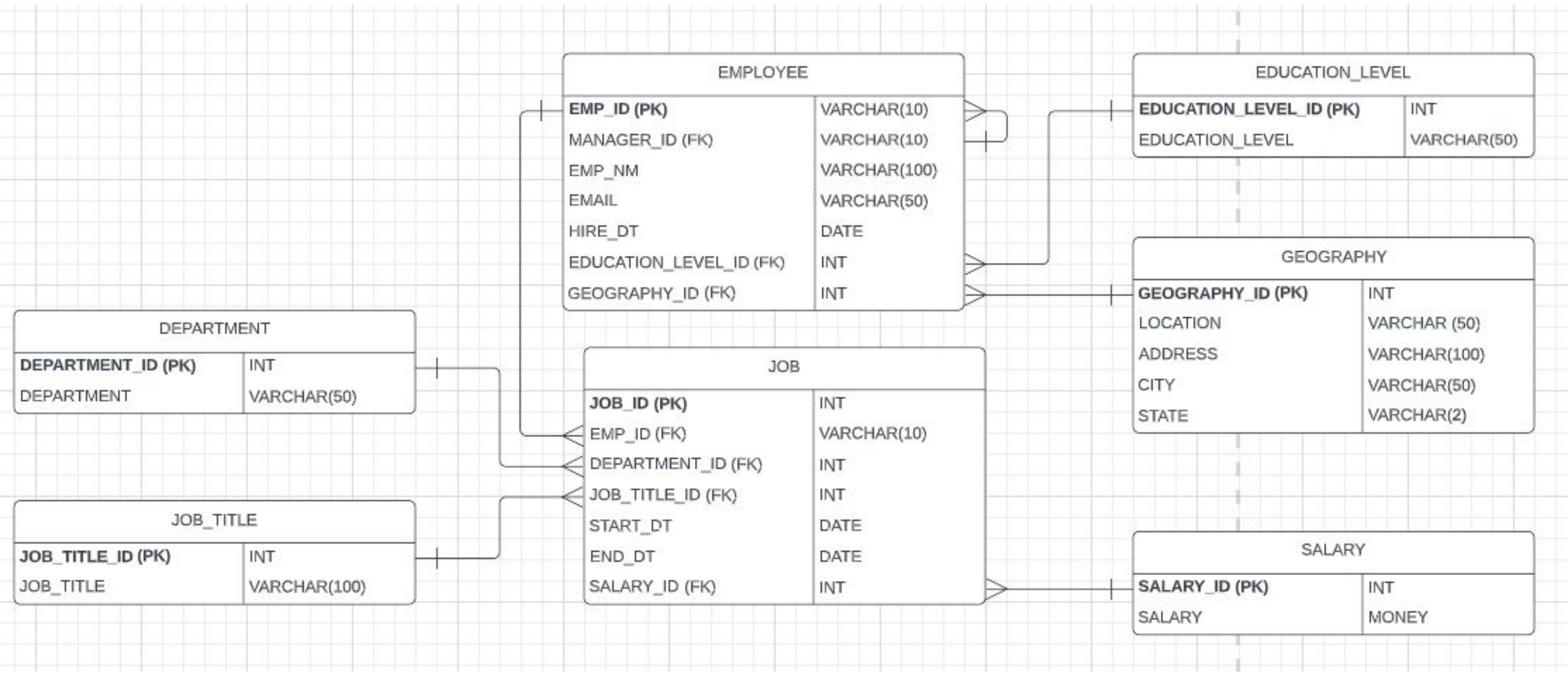
ERD

- **Logical**


DEPARTMENT
DEPARTMENT_ID
DEPARTMENT
JOB_TITLE
JOB_TITLE_ID
JOB_TITLE
JOB
JOB_ID
EMP_ID
DEPARTMENT_ID
JOB_TITLE_ID
START_DT
END_DT
SALARY_ID
EMPLOYEE
EMP_ID
MANAGER_ID
EMP_NM
EMAIL
HIRE_DT
EDUCATION_LEVEL_ID
GEOGRAPHY_ID
SALARY
SALARY_ID
SALARY
EDUCATION_LEVEL
EDUCATION_LEVEL_ID
EDUCATION_LEVEL
GEOGRAPHY
GEOGRAPHY_ID
LOCATION
ADDRESS
CITY
STATE

ERD

- **Physical**



EMPLOYEE	
EMP_ID (PK)	VARCHAR(10)
MANAGER_ID (FK)	VARCHAR(10)
EMP_NM	VARCHAR(100)
EMAIL	VARCHAR(50)
HIRE_DT	DATE
EDUCATION_LEVEL_ID (FK)	INT
GEOGRAPHY_ID (FK)	INT

EDUCATION_LEVEL	
EDUCATION_LEVEL_ID (PK)	INT
EDUCATION_LEVEL	VARCHAR(50)

GEOGRAPHY	
GEOGRAPHY_ID (PK)	INT
LOCATION	VARCHAR (50)
ADDRESS	VARCHAR(100)
CITY	VARCHAR(50)
STATE	VARCHAR(2)

DEPARTMENT	
DEPARTMENT_ID (PK)	INT
DEPARTMENT	VARCHAR(50)

JOB	
JOB_ID (PK)	INT
EMP_ID (FK)	VARCHAR(10)
DEPARTMENT_ID (FK)	INT
JOB_TITLE_ID (FK)	INT
START_DT	DATE
END_DT	DATE
SALARY_ID (FK)	INT

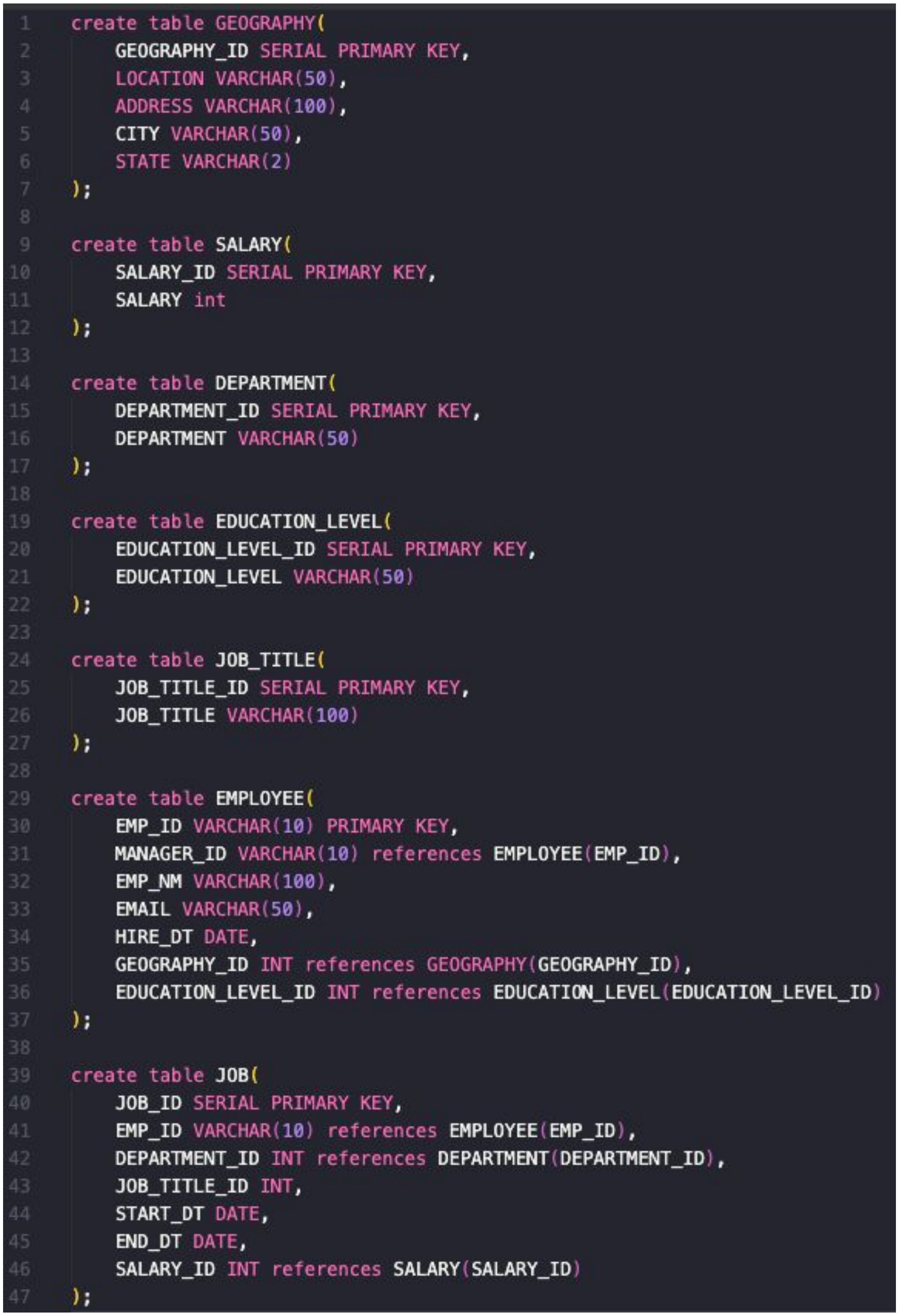
JOB_TITLE	
JOB_TITLE_ID (PK)	INT
JOB_TITLE	VARCHAR(100)

SALARY	
SALARY_ID (PK)	INT
SALARY	MONEY

Step 3

Create A Physical Database

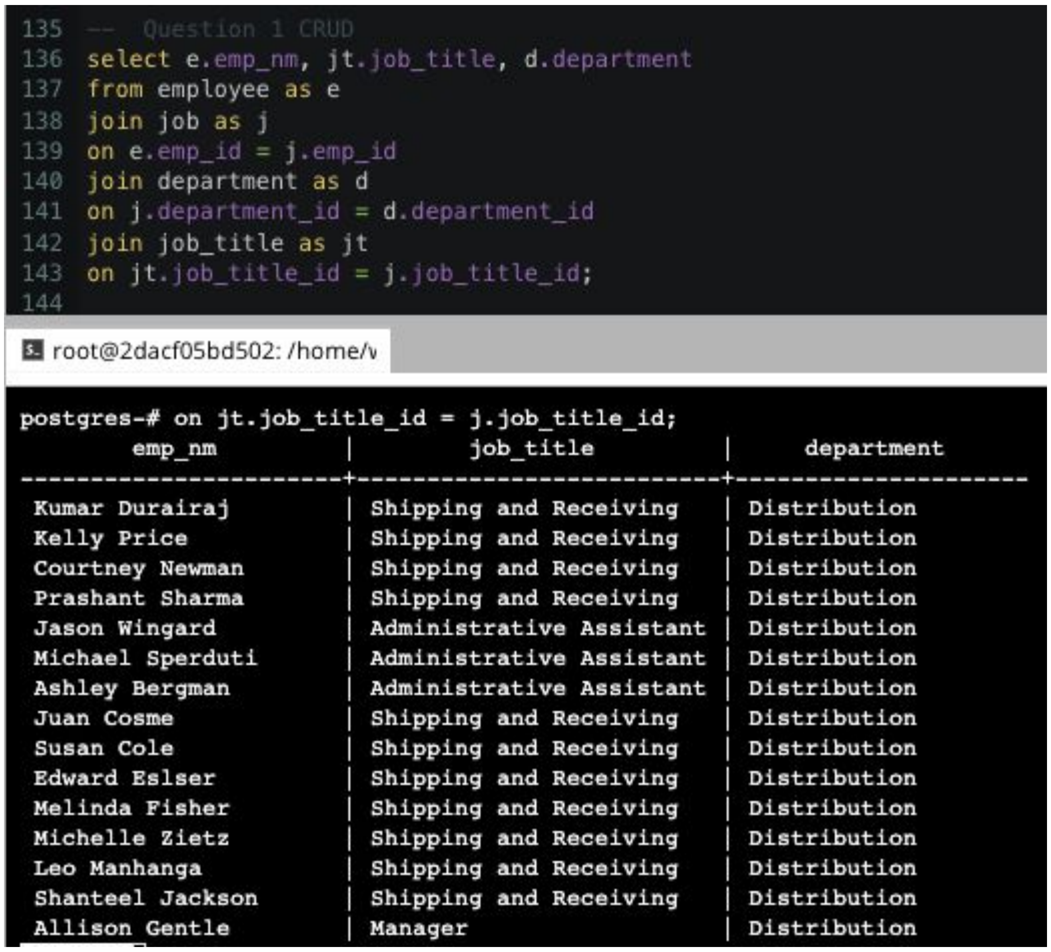
DDL



```
create table GEOGRAPHY(
    GEOGRAPHY_ID SERIAL PRIMARY KEY,
    LOCATION VARCHAR(50),
    ADDRESS VARCHAR(100),
    CITY VARCHAR(50),
    STATE VARCHAR(2)
);

create table SALARY(
    SALARY_ID SERIAL PRIMARY KEY,
    SALARY int
);

create table DEPARTMENT(
    DEPARTMENT_ID SERIAL PRIMARY KEY,
    DEPARTMENT VARCHAR(50)
);

create table EDUCATION_LEVEL(
    EDUCATION_LEVEL_ID SERIAL PRIMARY KEY,
    EDUCATION_LEVEL VARCHAR(50)
);

create table JOB_TITLE(
    JOB_TITLE_ID SERIAL PRIMARY KEY,
    JOB_TITLE VARCHAR(100)
);

create table EMPLOYEE(
    EMP_ID VARCHAR(10) PRIMARY KEY,
    MANAGER_ID VARCHAR(10) references EMPLOYEE(EMP_ID),
    EMP_NM VARCHAR(100),
    EMAIL VARCHAR(50),
    HIRE_DT DATE,
    GEOGRAPHY_ID INT references GEOGRAPHY(GEOGRAPHY_ID),
    EDUCATION_LEVEL_ID INT references EDUCATION_LEVEL(EDUCATION_LEVEL_ID)
);

create table JOB(
    JOB_ID SERIAL PRIMARY KEY,
    EMP_ID VARCHAR(10) references EMPLOYEE(EMP_ID),
    DEPARTMENT_ID INT references DEPARTMENT(DEPARTMENT_ID),
    JOB_TITLE_ID INT,
    START_DT DATE,
    END_DT DATE,
    SALARY_ID INT references SALARY(SALARY_ID)
);
```

CRUD

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



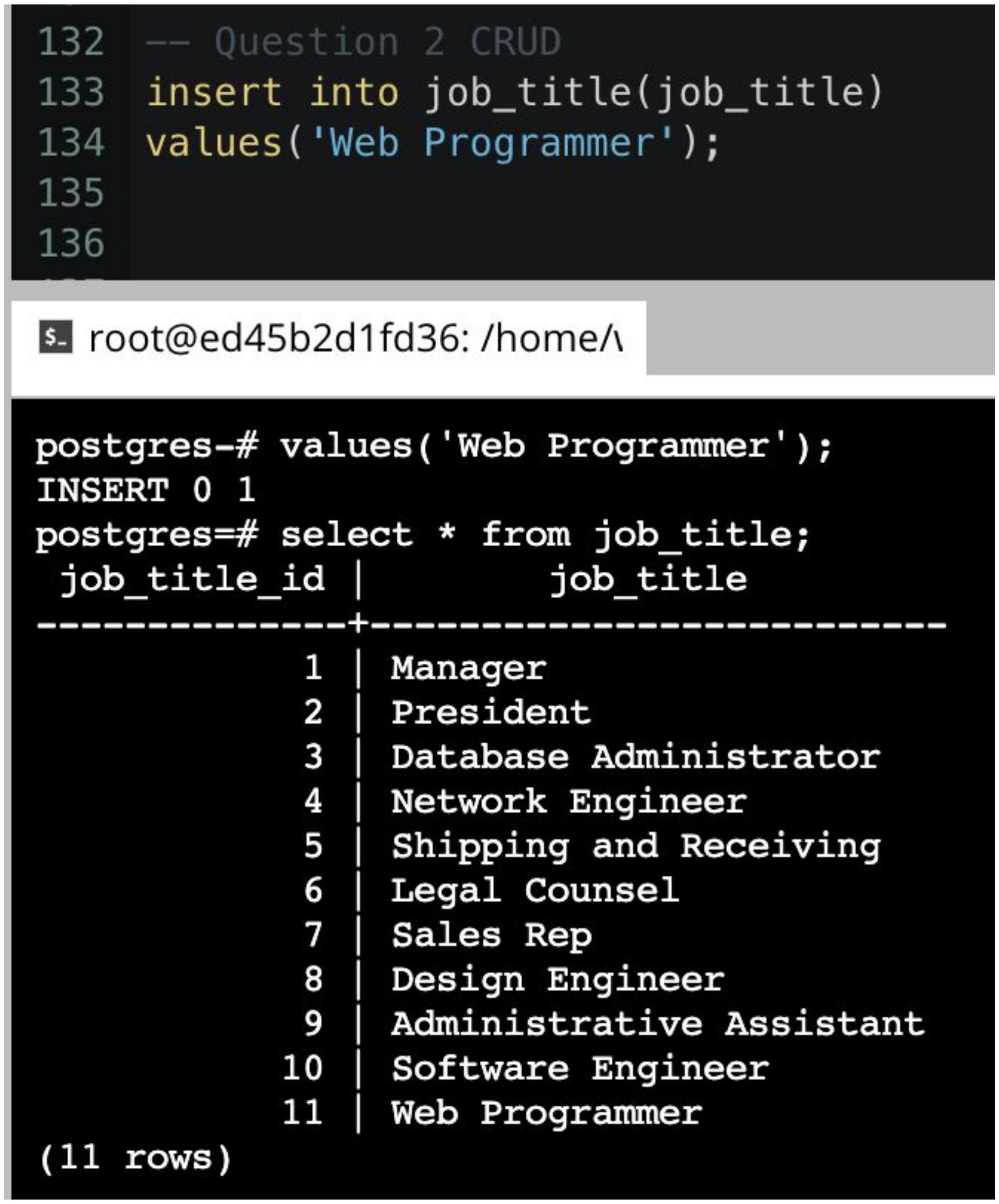
```sql
--  Question 1 CRUD
select e.emp_nm, jt.job_title, d.department
from employee as e
join job as j
on e.emp_id = j.emp_id
join department as d
on j.department_id = d.department_id
join job_title as jt
on jt.job_title_id = j.job_title_id;
```

root@2dacf05bd502: /home/v

```
postgres-# on jt.job_title_id = j.job_title_id;
        emp_nm          |        job_title         |     department
------------------------+--------------------------+---------------------
 Kumar Durairaj         | Shipping and Receiving   | Distribution
 Kelly Price            | Shipping and Receiving   | Distribution
 Courtney Newman        | Shipping and Receiving   | Distribution
 Prashant Sharma        | Shipping and Receiving   | Distribution
 Jason Wingard          | Administrative Assistant | Distribution
 Michael Sperduti       | Administrative Assistant | Distribution
 Ashley Bergman         | Administrative Assistant | Distribution
 Juan Cosme             | Shipping and Receiving   | Distribution
 Susan Cole             | Shipping and Receiving   | Distribution
 Edward Eslser          | Shipping and Receiving   | Distribution
 Melinda Fisher         | Shipping and Receiving   | Distribution
 Michelle Zietz         | Shipping and Receiving   | Distribution
 Leo Manhanga           | Shipping and Receiving   | Distribution
 Shanteel Jackson       | Shipping and Receiving   | Distribution
 Allison Gentle         | Manager                  | Distribution
```

CRUD

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



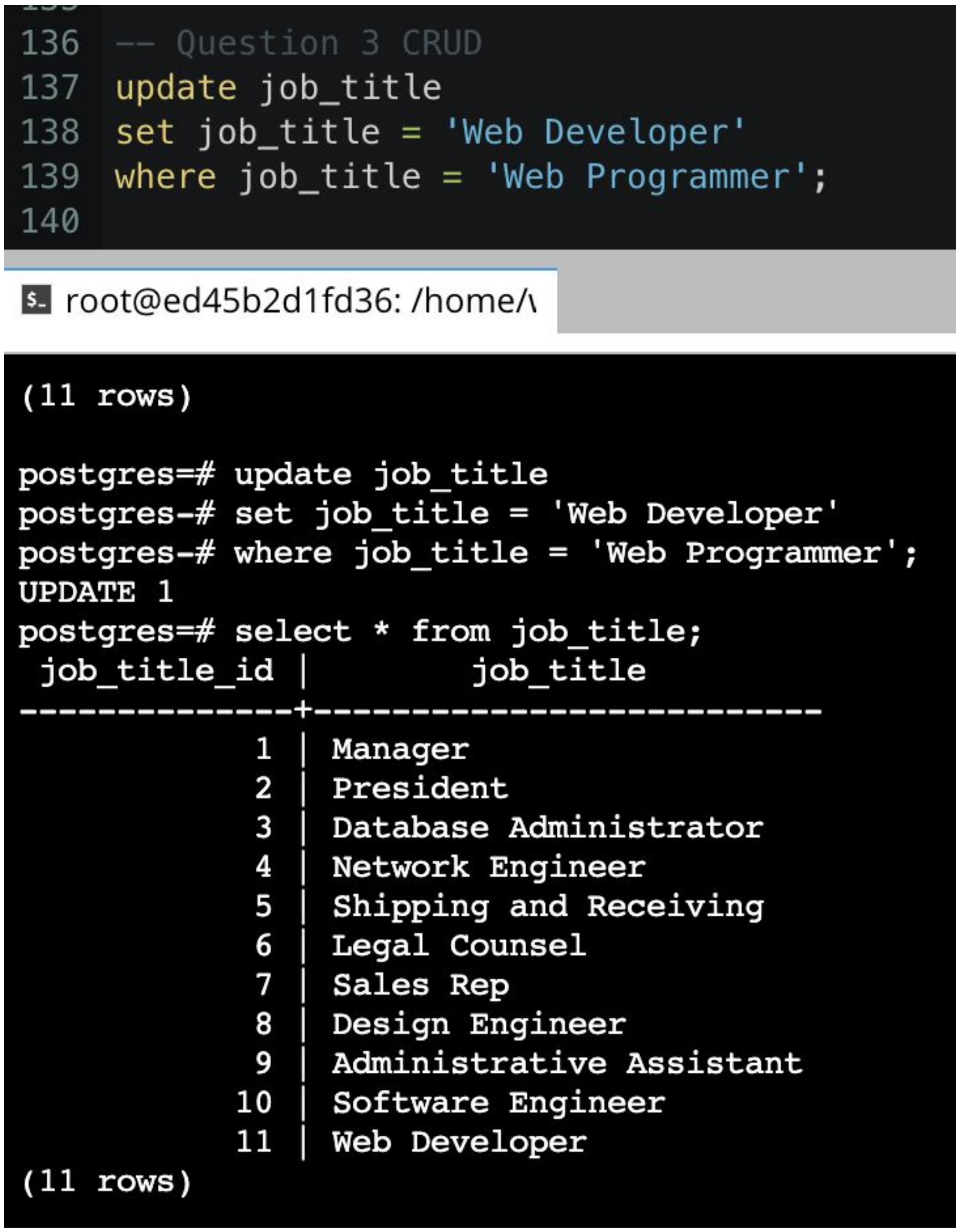
```
-- Question 2 CRUD
insert into job_title(job_title)
values('Web Programmer');
```

root@ed45b2d1fd36: /home/\

```
postgres-# values('Web Programmer');
INSERT 0 1
postgres=# select * from job_title;
 job_title_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)
```

CRUD

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



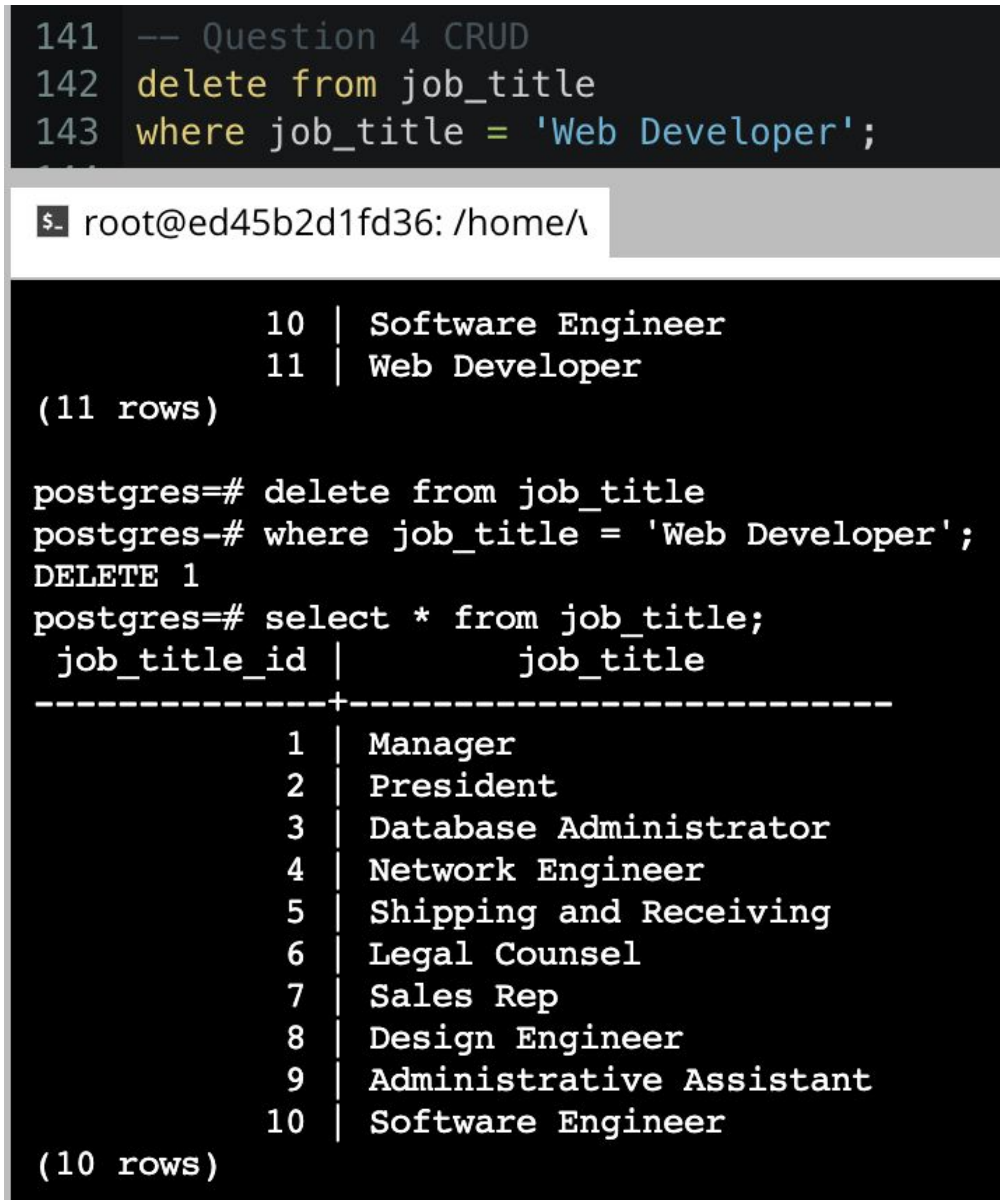
```
-- Question 3 CRUD
update job_title
set job_title = 'Web Developer'
where job_title = 'Web Programmer';
```

root@ed45b2d1fd36: /home/\

```
(11 rows)

postgres=# update job_title
postgres-# set job_title = 'Web Developer'
postgres-# where job_title = 'Web Programmer';
UPDATE 1
postgres=# select * from job_title;
 job_title_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)
```

CRUD

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



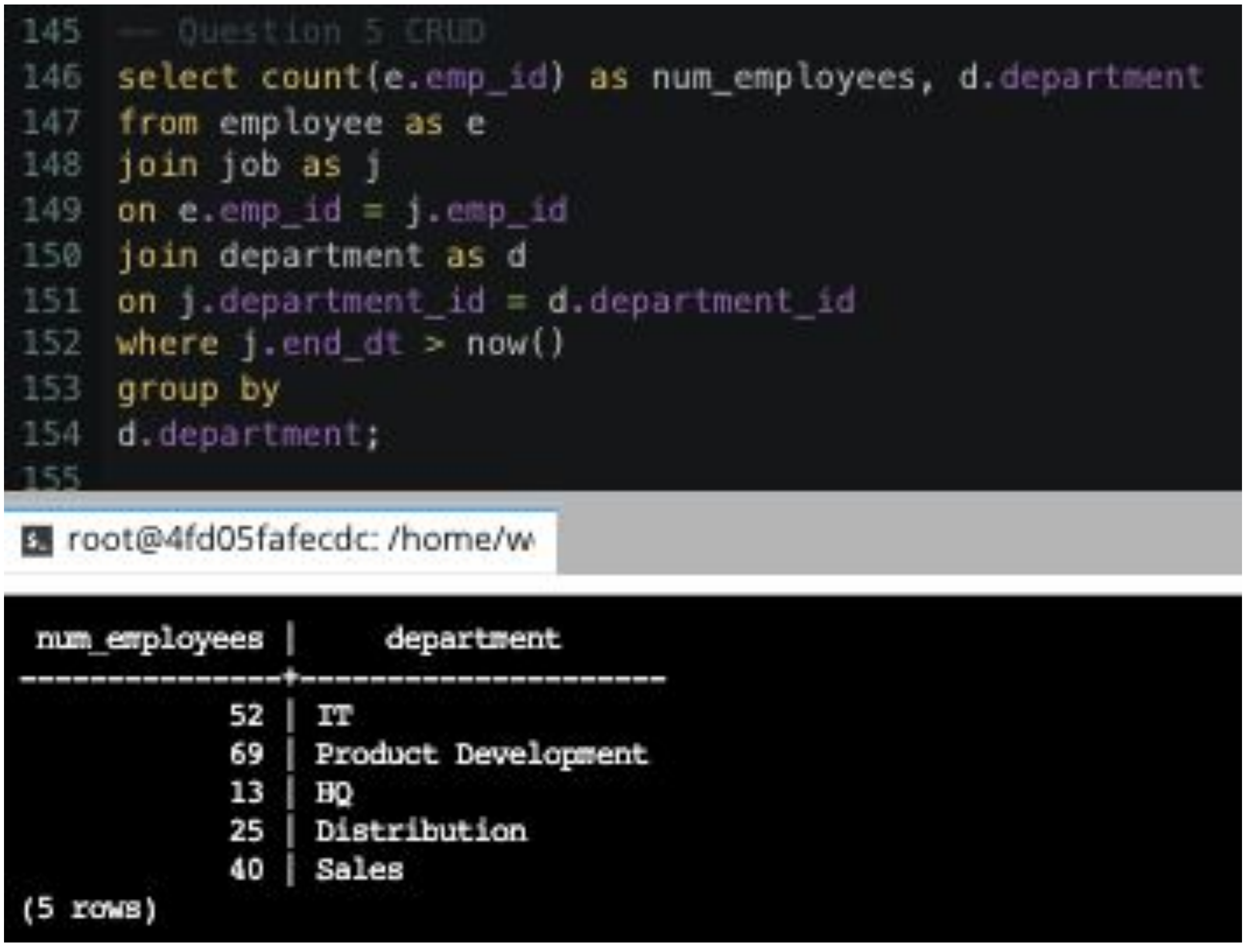
```
-- Question 4 CRUD
delete from job_title
where job_title = 'Web Developer';
```

root@ed45b2d1fd36: /home/\

```
             10 | Software Engineer
             11 | Web Developer
(11 rows)

postgres=# delete from job_title
postgres-# where job_title = 'Web Developer';
DELETE 1
postgres=# select * from job_title;
 job_title_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)
```

CRUD

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



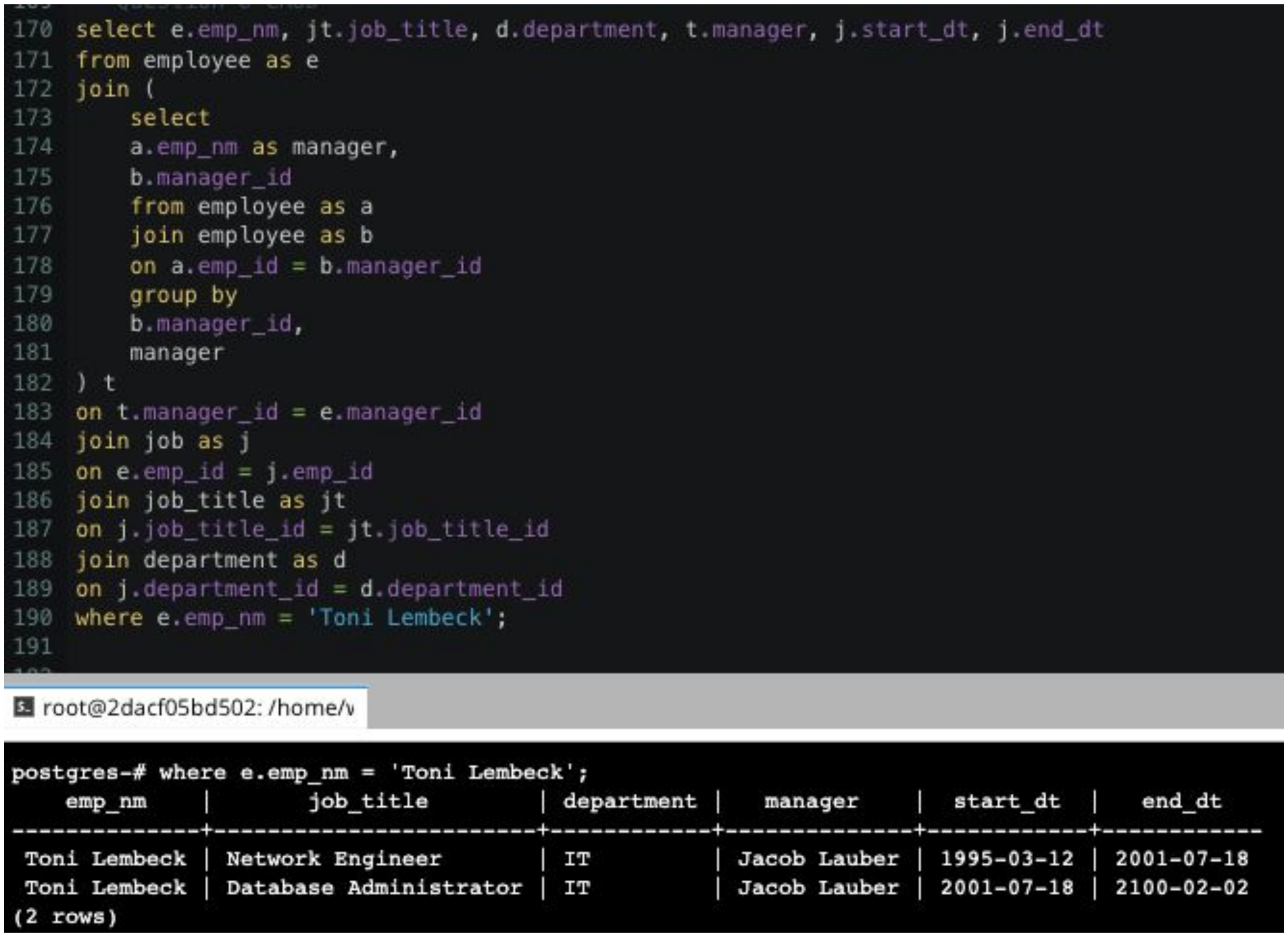
```
-- Question 5 CRUD
select count(e.emp_id) as num_employees, d.department
from employee as e
join job as j
on e.emp_id = j.emp_id
join department as d
on j.department_id = d.department_id
where j.end_dt > now()
group by
d.department;
```

root@4fd05fafecdc: /home/w

```
 num_employees |      department
---------------+---------------------
            52 | IT
            69 | Product Development
            13 | HQ
            25 | Distribution
            40 | Sales
(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.**



```
select e.emp_nm, jt.job_title, d.department, t.manager, j.start_dt, j.end_dt
from employee as e
join (
    select
    a.emp_nm as manager,
    b.manager_id
    from employee as a
    join employee as b
    on a.emp_id = b.manager_id
    group by
    b.manager_id,
    manager
) t
on t.manager_id = e.manager_id
join job as j
on e.emp_id = j.emp_id
join job_title as jt
on j.job_title_id = jt.job_title_id
join department as d
on j.department_id = d.department_id
where e.emp_nm = 'Toni Lembeck';
```

root@2dacf05bd502: /home/v

```
postgres-# where e.emp_nm = 'Toni Lembeck';
    emp_nm    |        job_title        | department |    manager    |  start_dt  |   end_dt
--------------+-------------------------+------------+---------------+------------+------------
 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)
```

CRUD

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

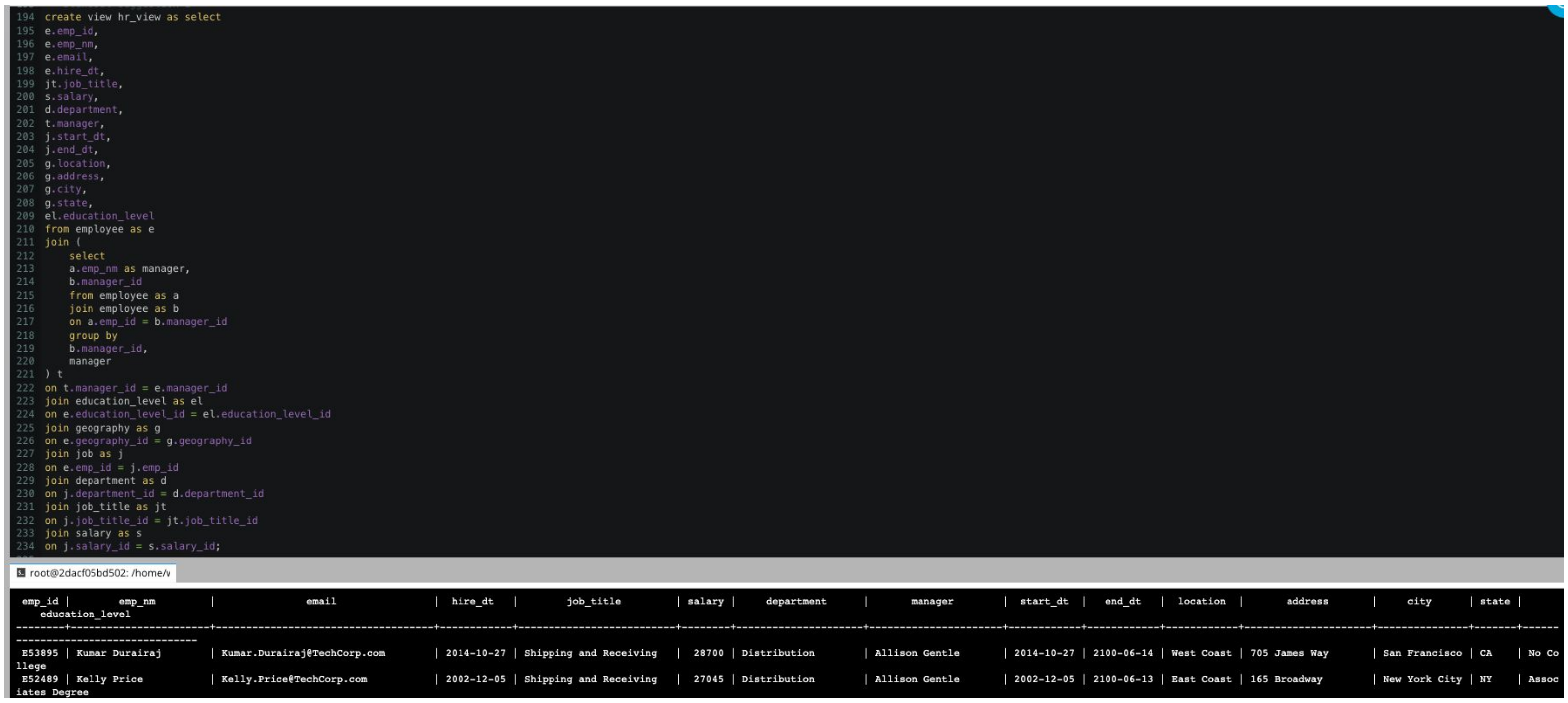
Based on the IT Best Practices document, the access restriction would be given by *Specific data in a database*, where the access is given to the entire database but revoked for the Salary table, which contains the mapping of the salary.

Step 4

Above and Beyond

(optional)

Standout Suggestion 1



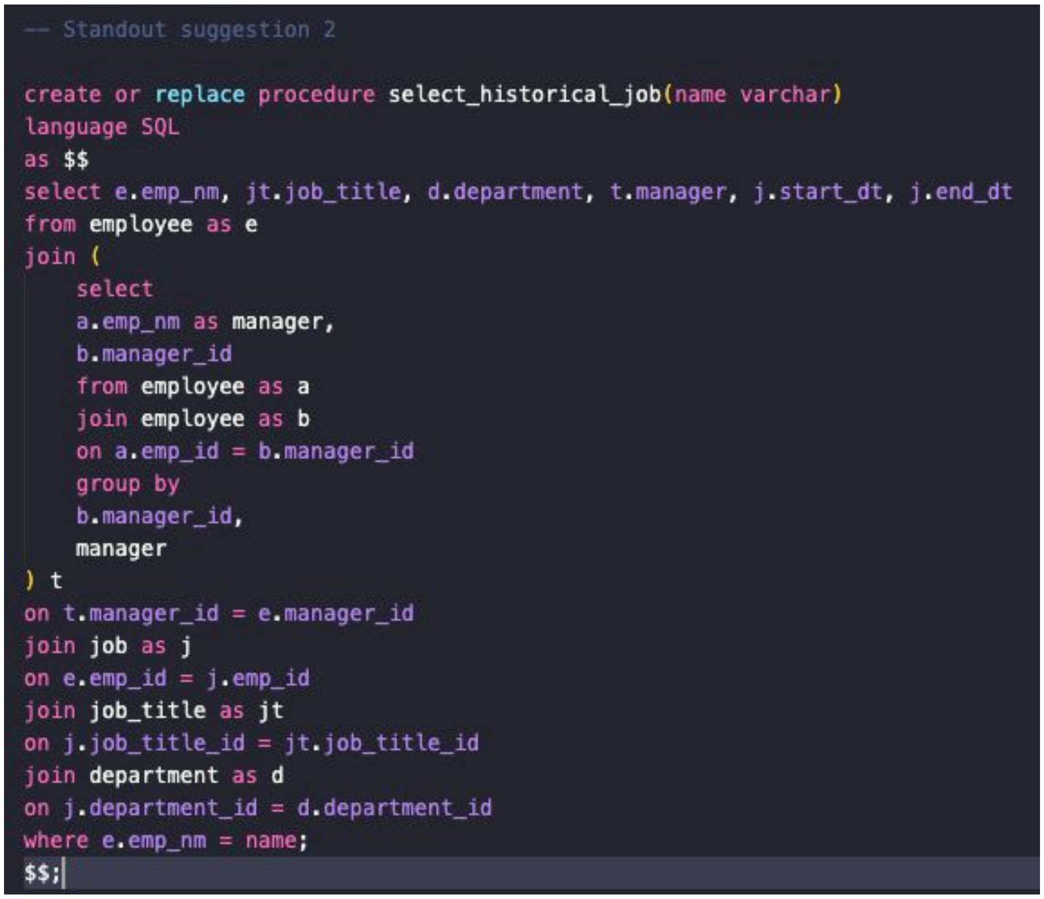
```sql
create view hr_view as select
e.emp_id,
e.emp_nm,
e.email,
e.hire_dt,
jt.job_title,
s.salary,
d.department,
t.manager,
j.start_dt,
j.end_dt,
g.location,
g.address,
g.city,
g.state,
el.education_level
from employee as e
join (
    select
    a.emp_nm as manager,
    b.manager_id
    from employee as a
    join employee as b
    on a.emp_id = b.manager_id
    group by
    b.manager_id,
    manager
) t
on t.manager_id = e.manager_id
join education_level as el
on e.education_level_id = el.education_level_id
join geography as g
on e.geography_id = g.geography_id
join job as j
on e.emp_id = j.emp_id
join department as d
on j.department_id = d.department_id
join job_title as jt
on j.job_title_id = jt.job_title_id
join salary as s
on j.salary_id = s.salary_id;
```

root@2dacf05bd502: /home/v

```
 emp_id |        emp_nm        |                email                |  hire_dt   |        job_title         | salary |     department      |       manager        |  start_dt  |   end_dt   |  location   |       address        |     city     | state |
    education_level
--------+----------------------+-------------------------------------+------------+--------------------------+--------+---------------------+----------------------+------------+------------+-------------+----------------------+--------------+-------+------
-----------------------------
 E53895 | Kumar Durairaj       | Kumar.Durairaj@TechCorp.com         | 2014-10-27 | Shipping and Receiving   |  28700 | Distribution        | Allison Gentle       | 2014-10-27 | 2100-06-14 | West Coast  | 705 James Way        | San Francisco | CA    | No Co
llege
 E52489 | Kelly Price          | Kelly.Price@TechCorp.com            | 2002-12-05 | Shipping and Receiving   |  27045 | Distribution        | Allison Gentle       | 2002-12-05 | 2100-06-13 | East Coast  | 165 Broadway         | New York City | NY    | Assoc
iates Degree
```

Standout Suggestion 2

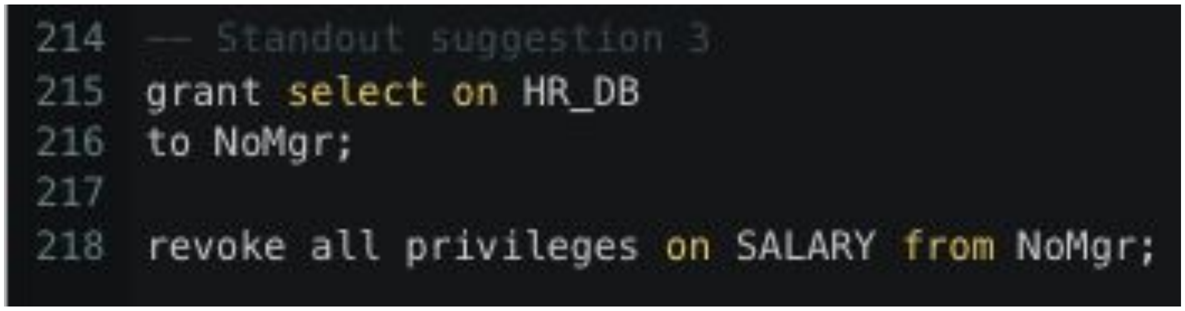
Note: The project workspace provided by Udacity uses PostgreSQL on version 9 which does not have stored procedure. This way, I was not able to test it. However, this is how I believe the stored procedure would look like:



```sql
-- Standout suggestion 2

create or replace procedure select_historical_job(name varchar)
language SQL
as $$
select e.emp_nm, jt.job_title, d.department, t.manager, j.start_dt, j.end_dt
from employee as e
join (
    select
    a.emp_nm as manager,
    b.manager_id
    from employee as a
    join employee as b
    on a.emp_id = b.manager_id
    group by
    b.manager_id,
    manager
) t
on t.manager_id = e.manager_id
join job as j
on e.emp_id = j.emp_id
join job_title as jt
on j.job_title_id = jt.job_title_id
join department as d
on j.department_id = d.department_id
where e.emp_nm = name;
$$;
```

Standout Suggestion 3

Implement user security on the restricted salary attribute.



```
-- Standout suggestion 3
grant select on HR_DB
to NoMgr;

revoke all privileges on SALARY from NoMgr;
```