Tech ABC Corp - HR Database

[QUANG HUY CHU & 2023-02-19]

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 able to 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.

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.

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**

  Since the customer company introduced the new AI-powered video game, their employee number has been increase significantly. Hence the old-designed Excel spreadsheet which use to manage HR data is obsoleted -> **A new way to store and handle data properly (Database) is necessary**

- **Data to be stored**

Column Name	Meaning
EMP_ID	Employee ID
EMP_NM	Employee name
EMAIL	Employee Email
HIRE_DT	Employee’s hired date
JOB_TITLE	Employee’s job title
SALARY	Employee’s Salary
DEPARTMENT	Employee’s Department
MANAGER	Employee’s Manager (Except President)
START_DT	Employee’s start working date
END_DT	Employee’s end working date (Can be NULL)
LOCATION	Employee’s working location
ADDRESS	Location’s address
CITY	Location’s city
STATE	Location’s state (Abbreviation)
EDUCATION LEVEL	Employee’s education level

Data Architect Business Requirement

- **Who will own/manage data**

  Data owner: HR

  Data managing entities: HR, Management level employees

- **Who will have access to database**

  Any employee with a domain login and with authorized access.

Data Architect Business Requirement

- **Estimated size of database**

  With the current HR Data Excel file, there are 205 rows in this Excel file. This can be consider a small size and even with a 20% projecting growth for the next 5 years, **a standard 1GB database can be applied to this migration case.**

- **Estimated annual growth**

  The customer estimate for **20% growth for the next 5 years**

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

  Any **employee with log in domain have read access** to this database, but **can not see their salary information.**

  Only **HR and Management level employee can see the salary and have read and write access to the database**

- **Data retention and backup requirements**

  By the federal regulation, data retention is **7 years**

  Also, since the customer mentioned that this is **critical data**, the database should be **backup 1x per week, incremental backup daily**

Data Architect Technical Requirement

- **Justification for the new database**

  This database should have a **mapping table**, and since this database is about the HR management, this should be the idea for mapping table

  Salary should be in a separate table with access retricted by log in users.

- **Database objects**

  With the requirement, I will list the necessary tables to form the database:

Table	Meaning
Employee	Employee information
EducationLVL	Employee’s education level
Job	Employee’s job information
Location	Employee’s working location
Department	Employee’s department
Salary	Employee’s salary
Boarding	Employee’s boarding, start and end date

Data Architect Technical Requirement

- **Data ingestion**

  Since we import the data to the dabase from Excel file, the **ETL method** should be considered for this case.

Data Architect Technical Requirement

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

  Ownership: HR will be the owner and manage the data

  User Access: only user with domain login and authorized access to the database

- **Scalability & Flexibility considerations**

  Currently, there are only 205 users (employees) at the moment, considering all of these users have the right to access the database, and 90% of them have read-only access → 185 users use this database to read. We should consider **making a replicated database** for read-only, this should make the database for read-and-write users easier when writing or updating the database even when the user increase.

  Although the customer is not concerned about the reporting at the moment, we should prepare for this scenario can be happen when the data getting larger. Hence, denormalization to OLAP database should also be considered in the future when the customer needed

Data Architect Technical Requirement

- **Storage & retention**

  Storage (disk or in-memory): In this case, storage in disk is the optimized option since the cusomter is not concern about the analytics or any machine learning side at the moment.
  Retention: 7 years by the federal regulation

- **Backup**

  A Critical backup plan: once per week with incremental backup daily is required

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.

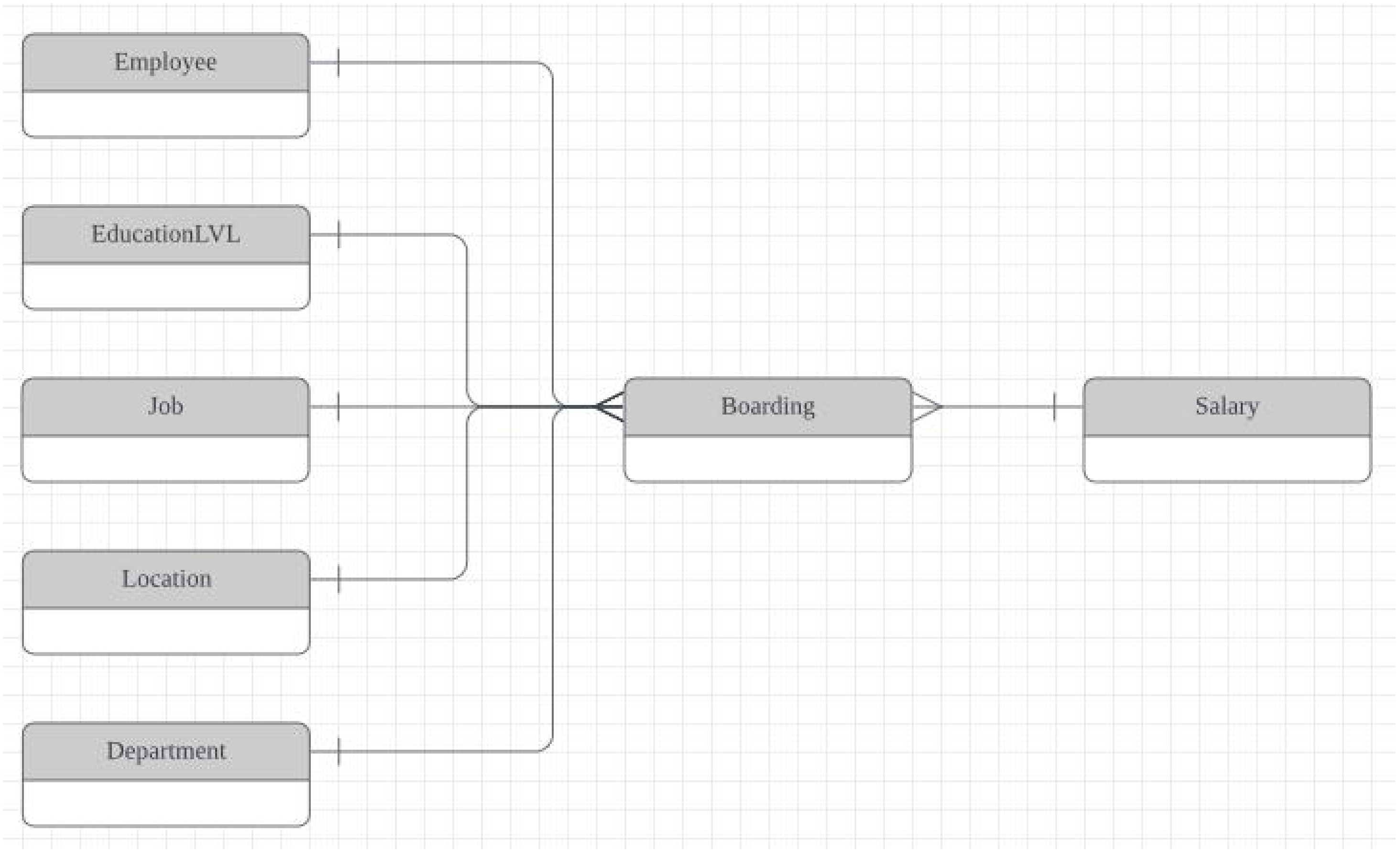
You will submit a screenshot for each of the 3 ERDs you create. You will find detailed instructions for developing each of the ERDs over the next several pages.

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.

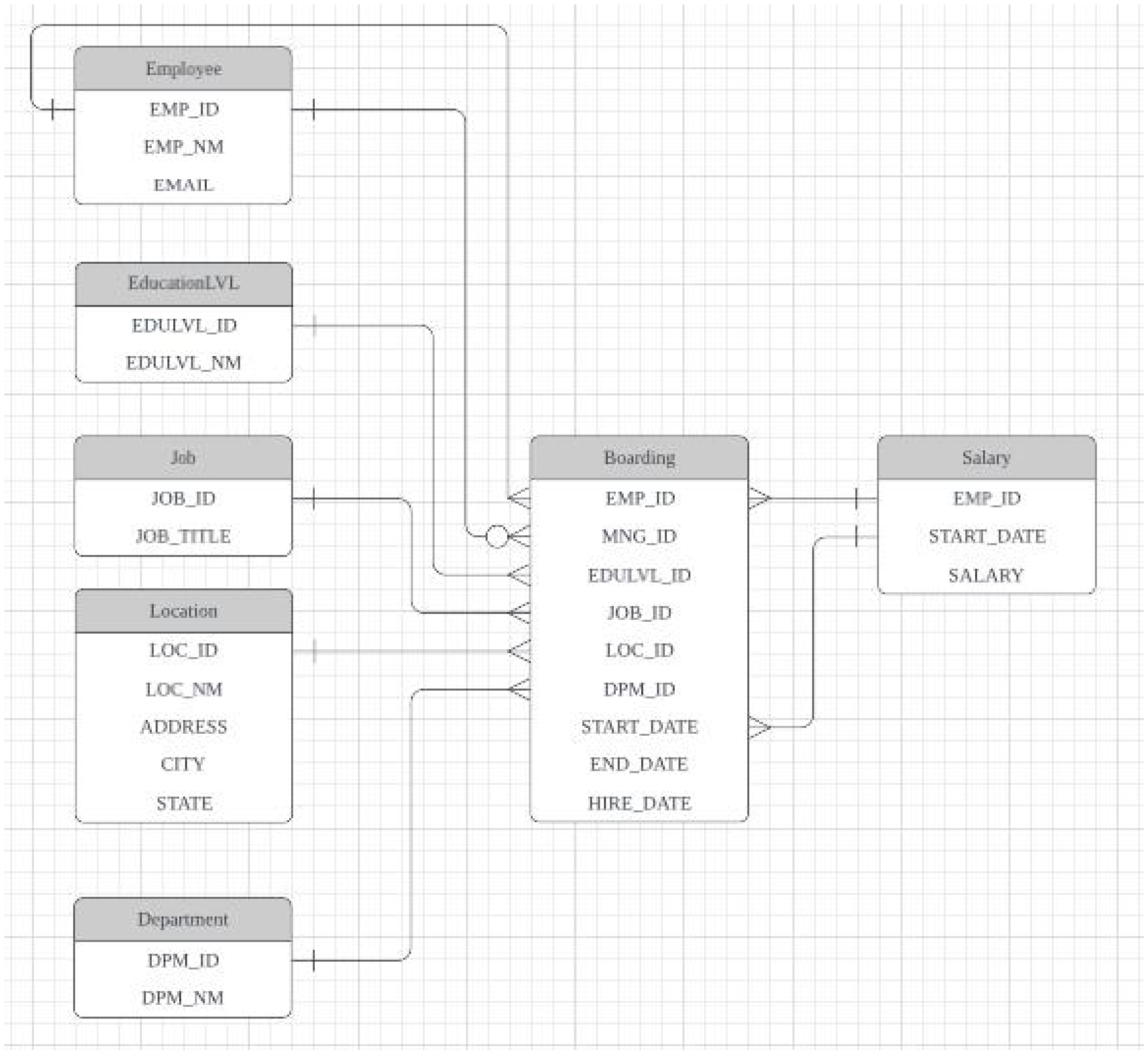


ERD

- ## 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.

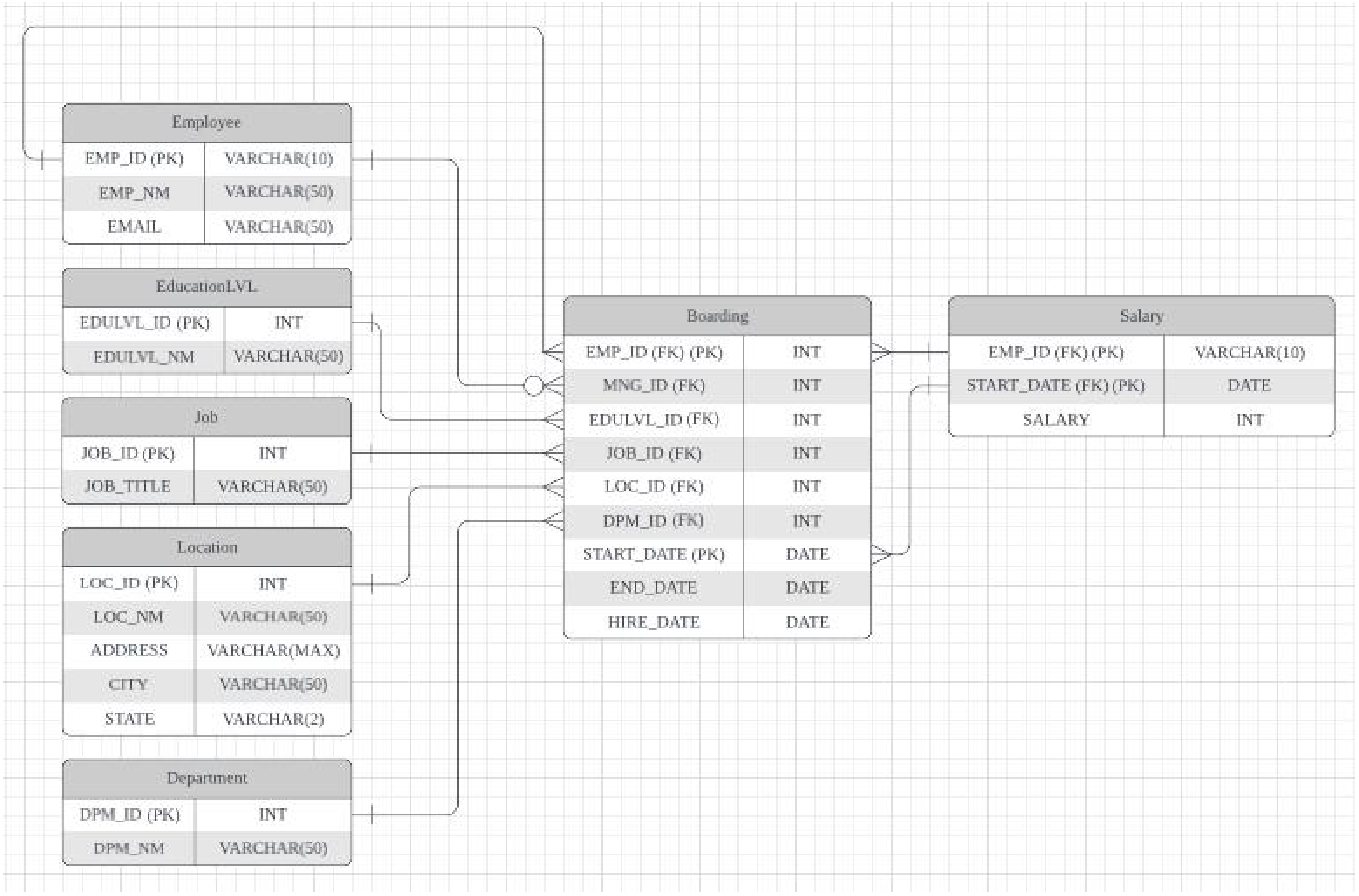
Use Lucidchart’s built-in template for DBMS ER Diagram UML.



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.

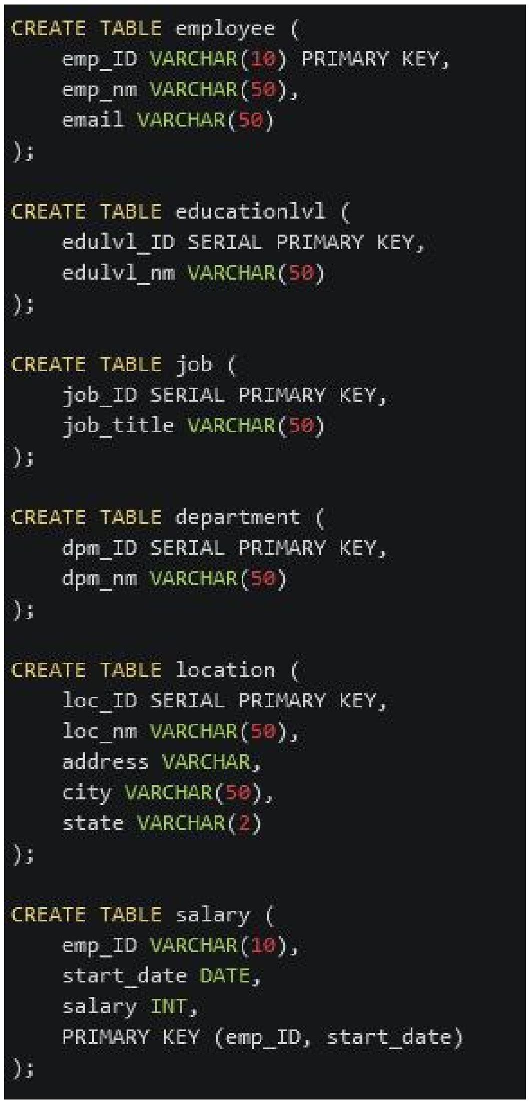
After running CRUD commands like update, insert, or delete, run a `SELECT*` command on the affected table, so the reviewer can see the results of the command.

DDL

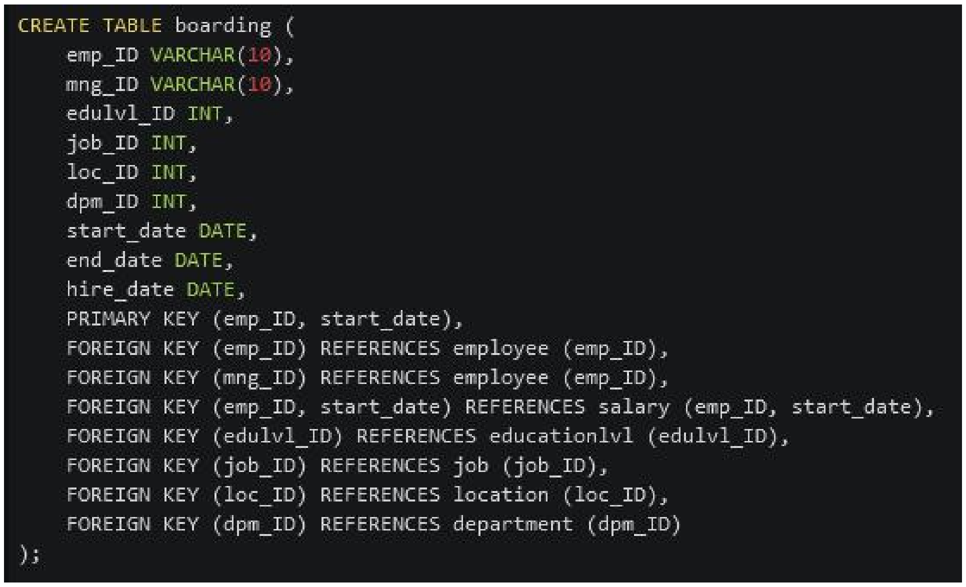
Create a DDL SQL script capable of building the database you designed in Step 2

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.



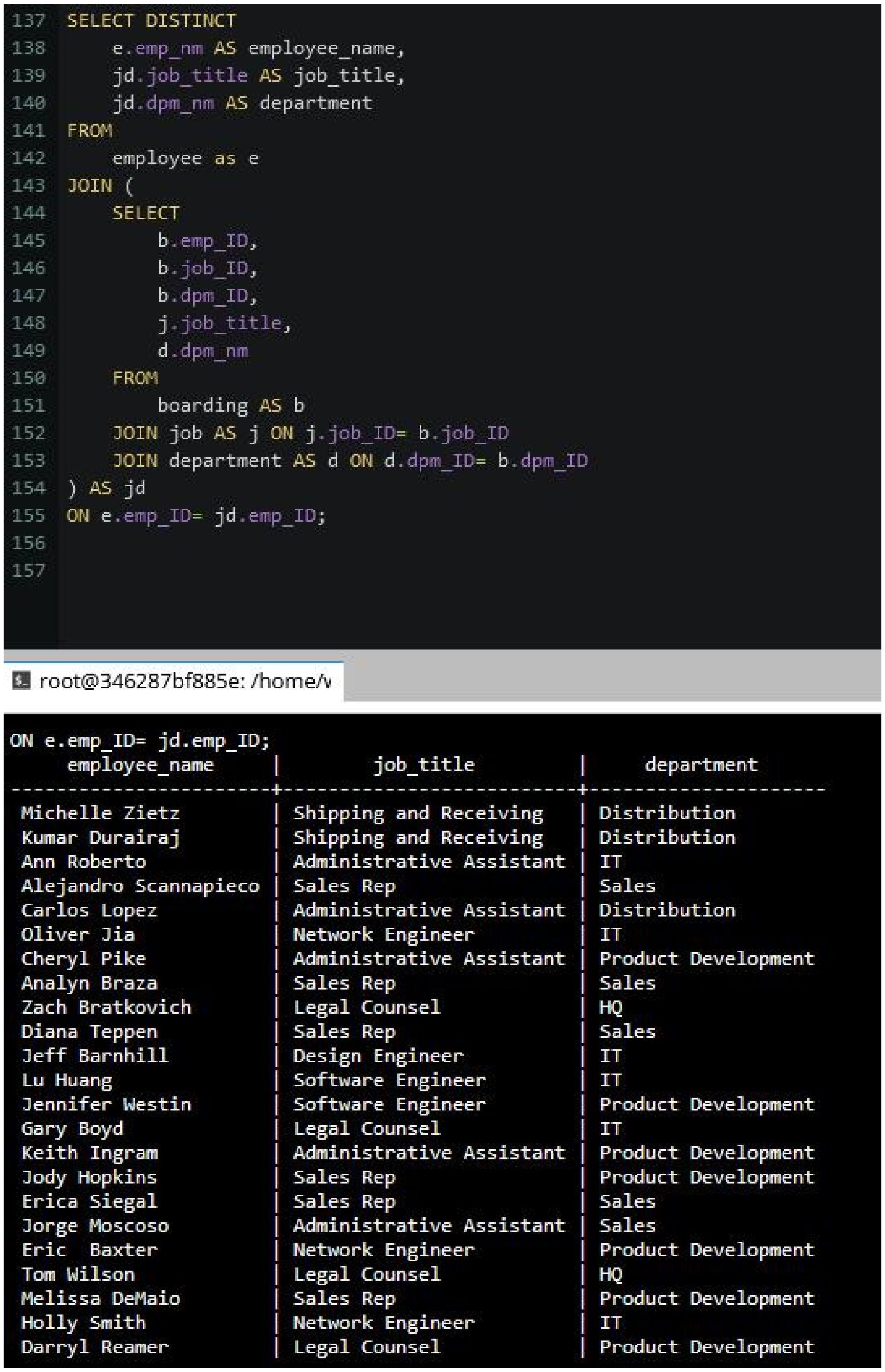
```sql
CREATE TABLE employee (
    emp_ID VARCHAR(10) PRIMARY KEY,
    emp_nm VARCHAR(50),
    email VARCHAR(50)
);

CREATE TABLE educationlvl (
    edulvl_ID SERIAL PRIMARY KEY,
    edulvl_nm VARCHAR(50)
);

CREATE TABLE job (
    job_ID SERIAL PRIMARY KEY,
    job_title VARCHAR(50)
);

CREATE TABLE department (
    dpm_ID SERIAL PRIMARY KEY,
    dpm_nm VARCHAR(50)
);

CREATE TABLE location (
    loc_ID SERIAL PRIMARY KEY,
    loc_nm VARCHAR(50),
    address VARCHAR,
    city VARCHAR(50),
    state VARCHAR(2)
);

CREATE TABLE salary (
    emp_ID VARCHAR(10),
    start_date DATE,
    salary INT,
    PRIMARY KEY (emp_ID, start_date)
);
```



```sql
CREATE TABLE boarding (
    emp_ID VARCHAR(10),
    mng_ID VARCHAR(10),
    edulvl_ID INT,
    job_ID INT,
    loc_ID INT,
    dpm_ID INT,
    start_date DATE,
    end_date DATE,
    hire_date DATE,
    PRIMARY KEY (emp_ID, start_date),
    FOREIGN KEY (emp_ID) REFERENCES employee (emp_ID),
    FOREIGN KEY (mng_ID) REFERENCES employee (emp_ID),
    FOREIGN KEY (emp_ID, start_date) REFERENCES salary (emp_ID, start_date),
    FOREIGN KEY (edulvl_ID) REFERENCES educationlvl (edulvl_ID),
    FOREIGN KEY (job_ID) REFERENCES job (job_ID),
    FOREIGN KEY (loc_ID) REFERENCES location (loc_ID),
    FOREIGN KEY (dpm_ID) REFERENCES department (dpm_ID)
);
```

CRUD

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



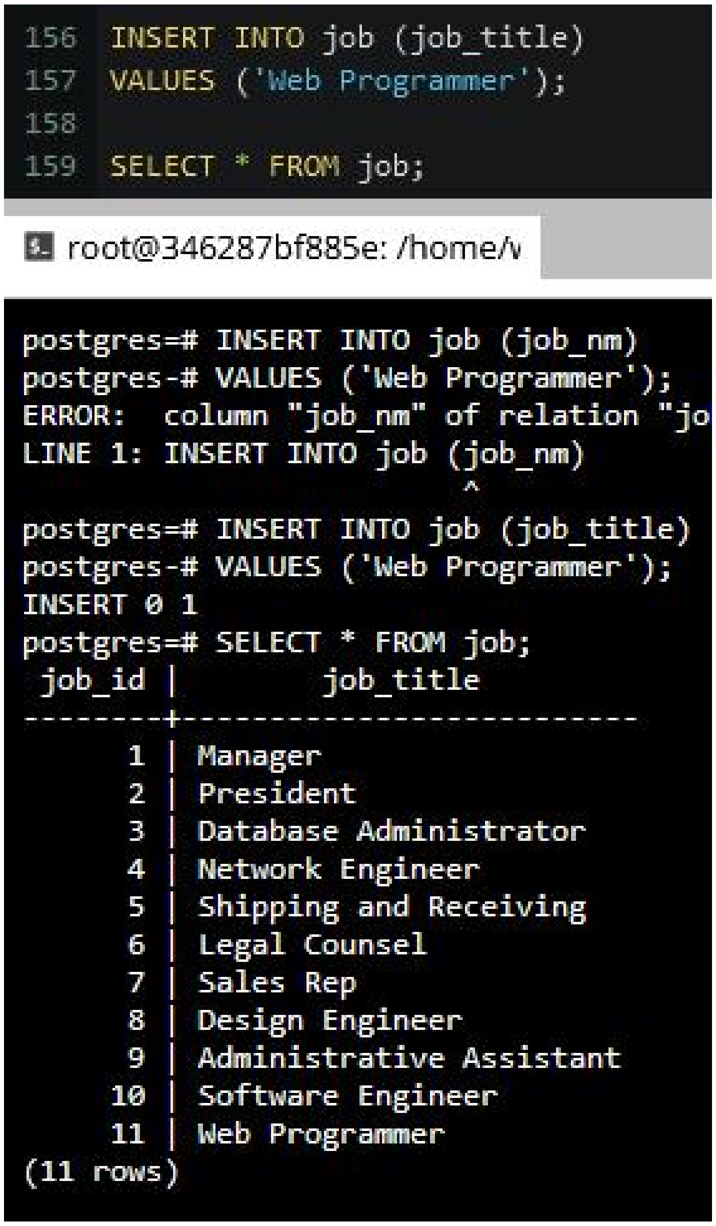
```
SELECT DISTINCT
    e.emp_nm AS employee_name,
    jd.job_title AS job_title,
    jd.dpm_nm AS department
FROM
    employee as e
JOIN (
    SELECT
        b.emp_ID,
        b.job_ID,
        b.dpm_ID,
        j.job_title,
        d.dpm_nm
    FROM
        boarding AS b
    JOIN job AS j ON j.job_ID= b.job_ID
    JOIN department AS d ON d.dpm_ID= b.dpm_ID
) AS jd
ON e.emp_ID= jd.emp_ID;
```

root@346287bf885e: /home/v

```
ON e.emp_ID= jd.emp_ID;
     employee_name      |         job_title          |      department
------------------------+----------------------------+---------------------
 Michelle Zietz         | Shipping and Receiving     | Distribution
 Kumar Durairaj         | Shipping and Receiving     | Distribution
 Ann Roberto            | Administrative Assistant   | IT
 Alejandro Scannapieco  | Sales Rep                  | Sales
 Carlos Lopez           | Administrative Assistant   | Distribution
 Oliver Jia             | Network Engineer           | IT
 Cheryl Pike            | Administrative Assistant   | Product Development
 Analyn Braza           | Sales Rep                  | Sales
 Zach Bratkovich        | Legal Counsel              | HQ
 Diana Teppen           | Sales Rep                  | Sales
 Jeff Barnhill          | Design Engineer            | IT
 Lu Huang               | Software Engineer          | IT
 Jennifer Westin        | Software Engineer          | Product Development
 Gary Boyd              | Legal Counsel              | IT
 Keith Ingram           | Administrative Assistant   | Product Development
 Jody Hopkins           | Sales Rep                  | Product Development
 Erica Siegal           | Sales Rep                  | Sales
 Jorge Moscoso          | Administrative Assistant   | Sales
 Eric  Baxter           | Network Engineer           | Product Development
 Tom Wilson             | Legal Counsel              | HQ
 Melissa DeMaio         | Sales Rep                  | Product Development
 Holly Smith            | Network Engineer           | IT
 Darryl Reamer          | Legal Counsel              | Product Development
```

CRUD

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



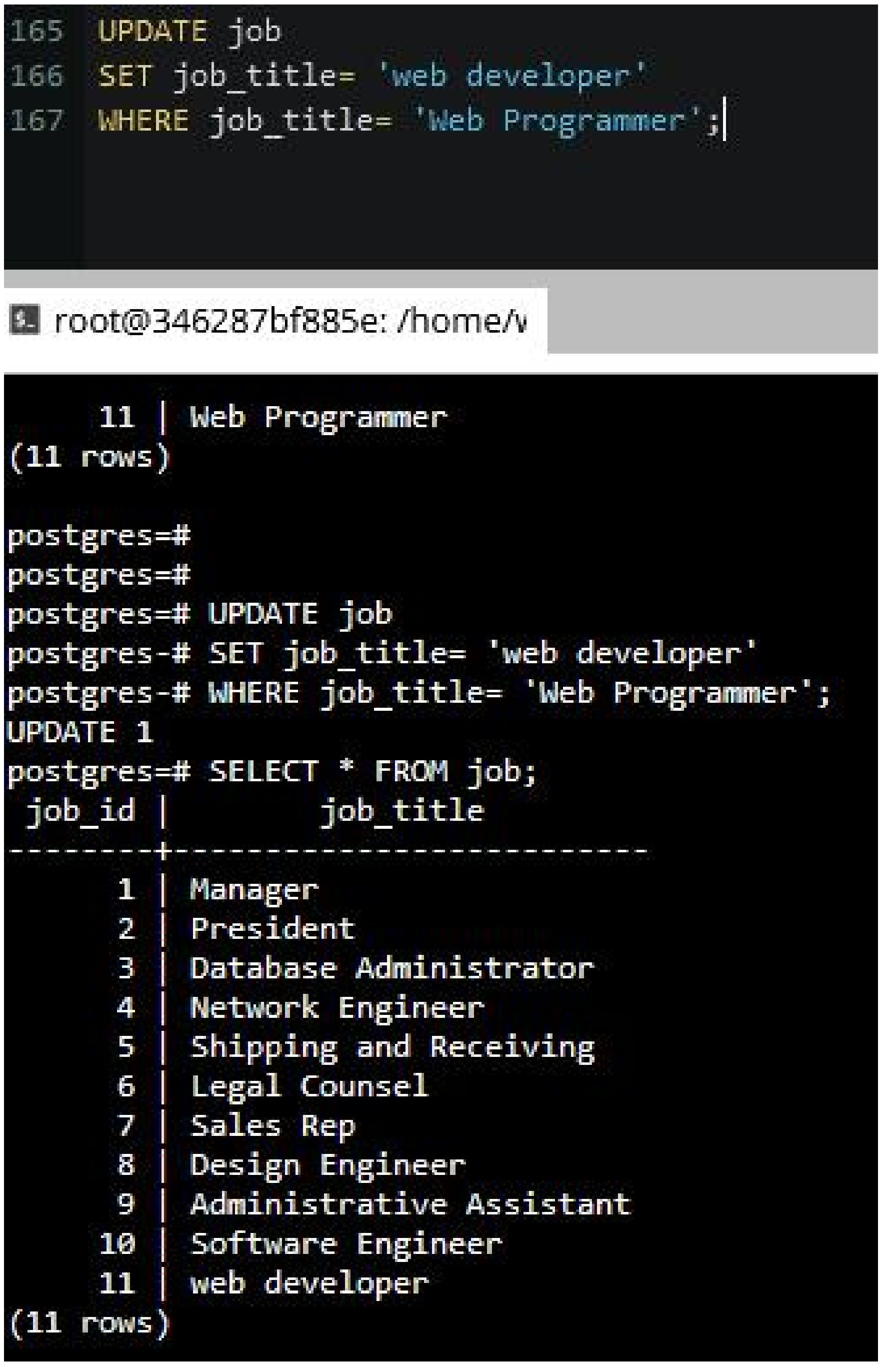
```
INSERT INTO job (job_title)
VALUES ('Web Programmer');

SELECT * FROM job;
```

root@346287bf885e: /home/v

```
postgres=# INSERT INTO job (job_nm)
postgres-# VALUES ('Web Programmer');
ERROR:  column "job_nm" of relation "jo
LINE 1: INSERT INTO job (job_nm)
                         ^
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)
```

CRUD

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



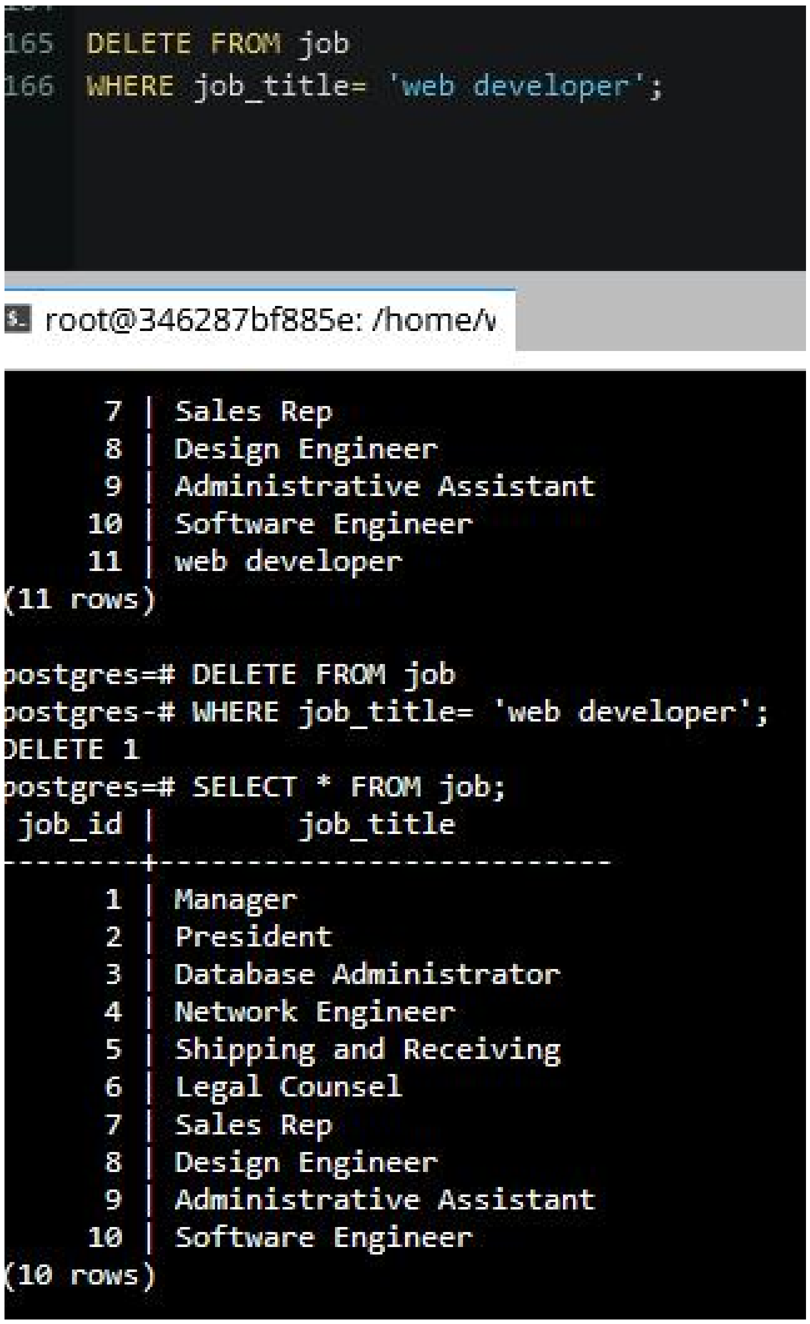
```
UPDATE job
SET job_title= 'web developer'
WHERE job_title= 'Web Programmer';
```

root@346287bf885e: /home/v

```
      11 | Web Programmer
(11 rows)

postgres=#
postgres=#
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)
```

CRUD

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



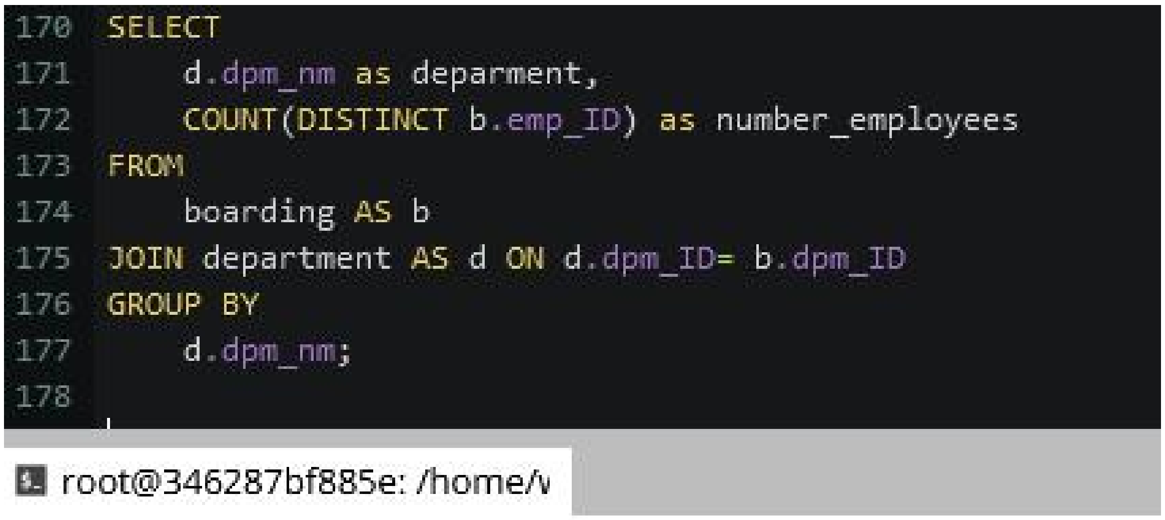
```
DELETE FROM job
WHERE job_title= 'web developer';
```

root@346287bf885e: /home/v

```
       7 | Sales Rep
       8 | Design Engineer
       9 | Administrative Assistant
      10 | Software Engineer
      11 | web developer
(11 rows)

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)
```

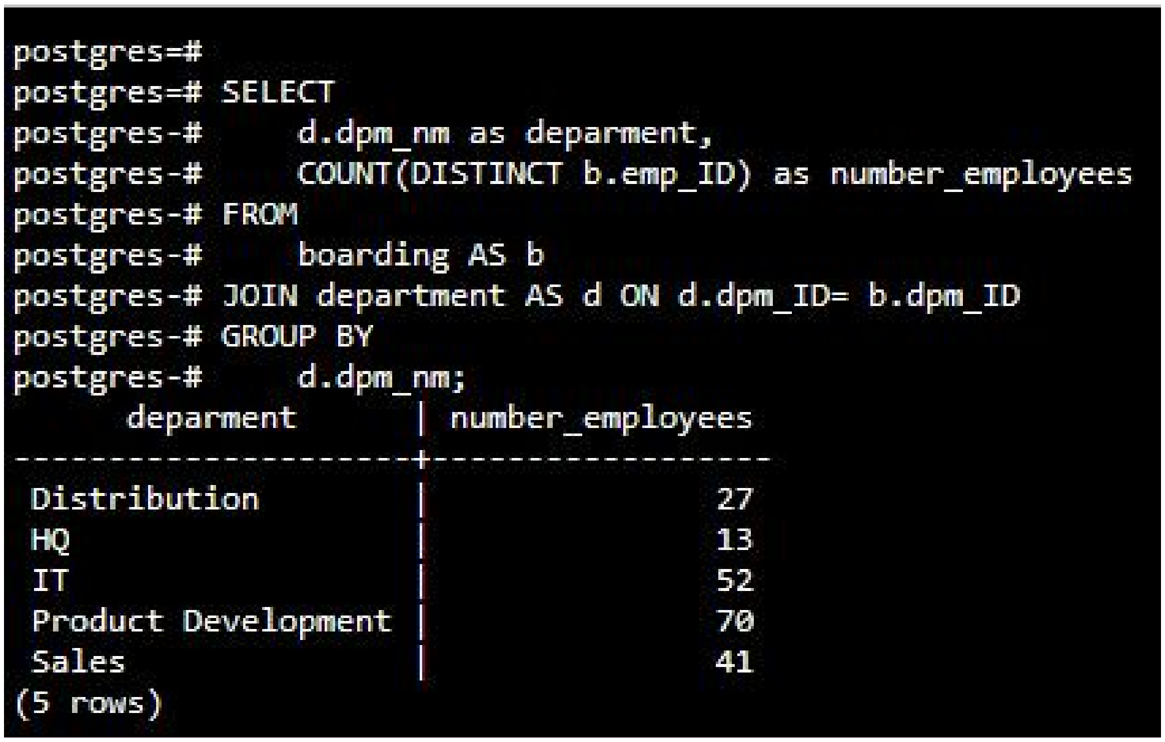
CRUD

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



```sql
SELECT
    d.dpm_nm as deparment,
    COUNT(DISTINCT b.emp_ID) as number_employees
FROM
    boarding AS b
JOIN department AS d ON d.dpm_ID= b.dpm_ID
GROUP BY
    d.dpm_nm;
```

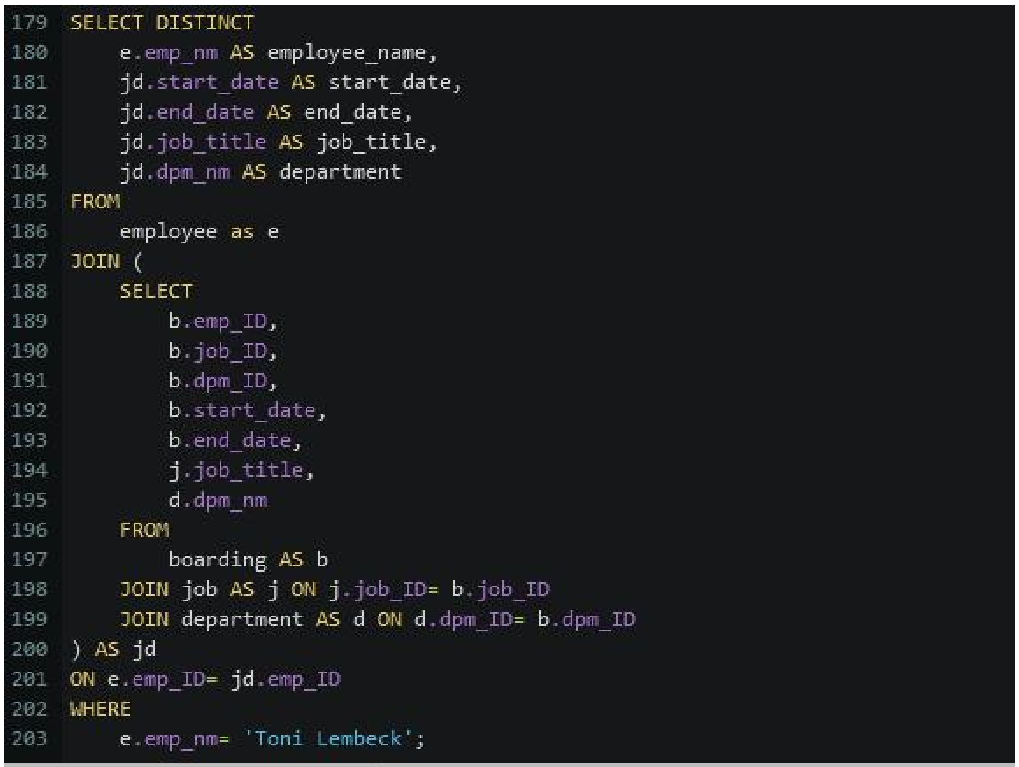
root@346287bf885e: /home/v



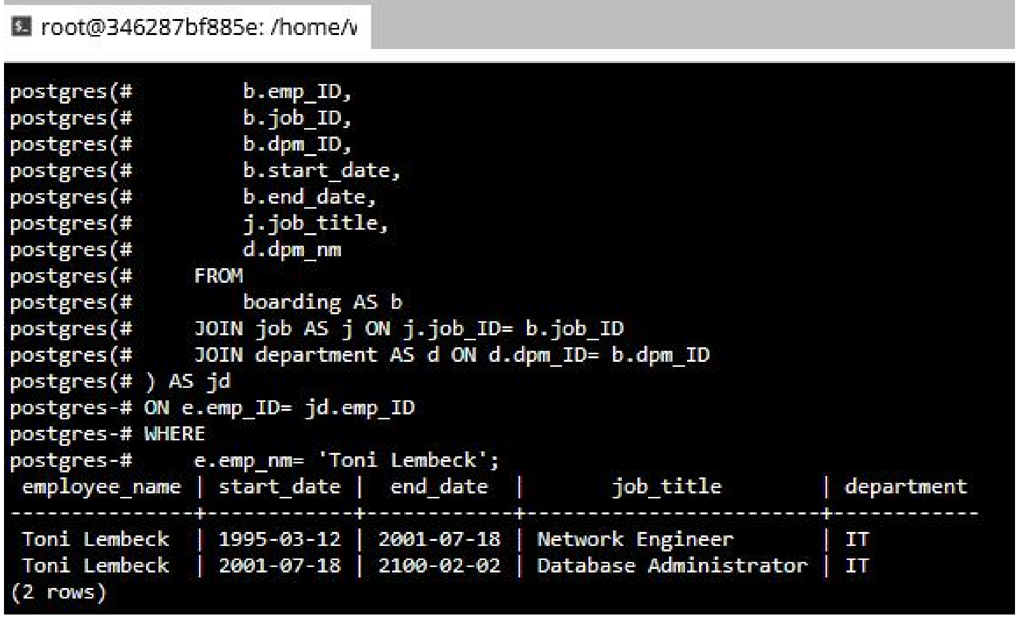
```
postgres=#
postgres=# SELECT
postgres-#     d.dpm_nm as deparment,
postgres-#     COUNT(DISTINCT b.emp_ID) as number_employees
postgres-# FROM
postgres-#     boarding AS b
postgres-# JOIN department AS d ON d.dpm_ID= b.dpm_ID
postgres-# GROUP BY
postgres-#     d.dpm_nm;
      deparment       | number_employees
----------------------+------------------
 Distribution         |               27
 HQ                   |               13
 IT                   |               52
 Product Development  |               70
 Sales                |               41
(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 DISTINCT
    e.emp_nm AS employee_name,
    jd.start_date AS start_date,
    jd.end_date AS end_date,
    jd.job_title AS job_title,
    jd.dpm_nm AS department
FROM
    employee as e
JOIN (
    SELECT
        b.emp_ID,
        b.job_ID,
        b.dpm_ID,
        b.start_date,
        b.end_date,
        j.job_title,
        d.dpm_nm
    FROM
        boarding AS b
    JOIN job AS j ON j.job_ID= b.job_ID
    JOIN department AS d ON d.dpm_ID= b.dpm_ID
) AS jd
ON e.emp_ID= jd.emp_ID
WHERE
    e.emp_nm= 'Toni Lembeck';
```



root@346287bf885e: /home/\

```
postgres(#         b.emp_ID,
postgres(#         b.job_ID,
postgres(#         b.dpm_ID,
postgres(#         b.start_date,
postgres(#         b.end_date,
postgres(#         j.job_title,
postgres(#         d.dpm_nm
postgres(#     FROM
postgres(#         boarding AS b
postgres(#     JOIN job AS j ON j.job_ID= b.job_ID
postgres(#     JOIN department AS d ON d.dpm_ID= b.dpm_ID
postgres(# ) AS jd
postgres-# ON e.emp_ID= jd.emp_ID
postgres-# WHERE
postgres-#     e.emp_nm= 'Toni Lembeck';
 employee_name | start_date |  end_date  |       job_title        | department
---------------+------------+------------+------------------------+------------
 Toni Lembeck  | 1995-03-12 | 2001-07-18 | Network Engineer       | IT
 Toni Lembeck  | 2001-07-18 | 2100-02-02 | Database Administrator | IT
(2 rows)
```

CRUD

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

First, grant access to all users.

Then revoke access to salary table for users who have employee **role**, using the REVOKE command

Step 4
Above and Beyond
(optional)

Step 4: Above and Beyond

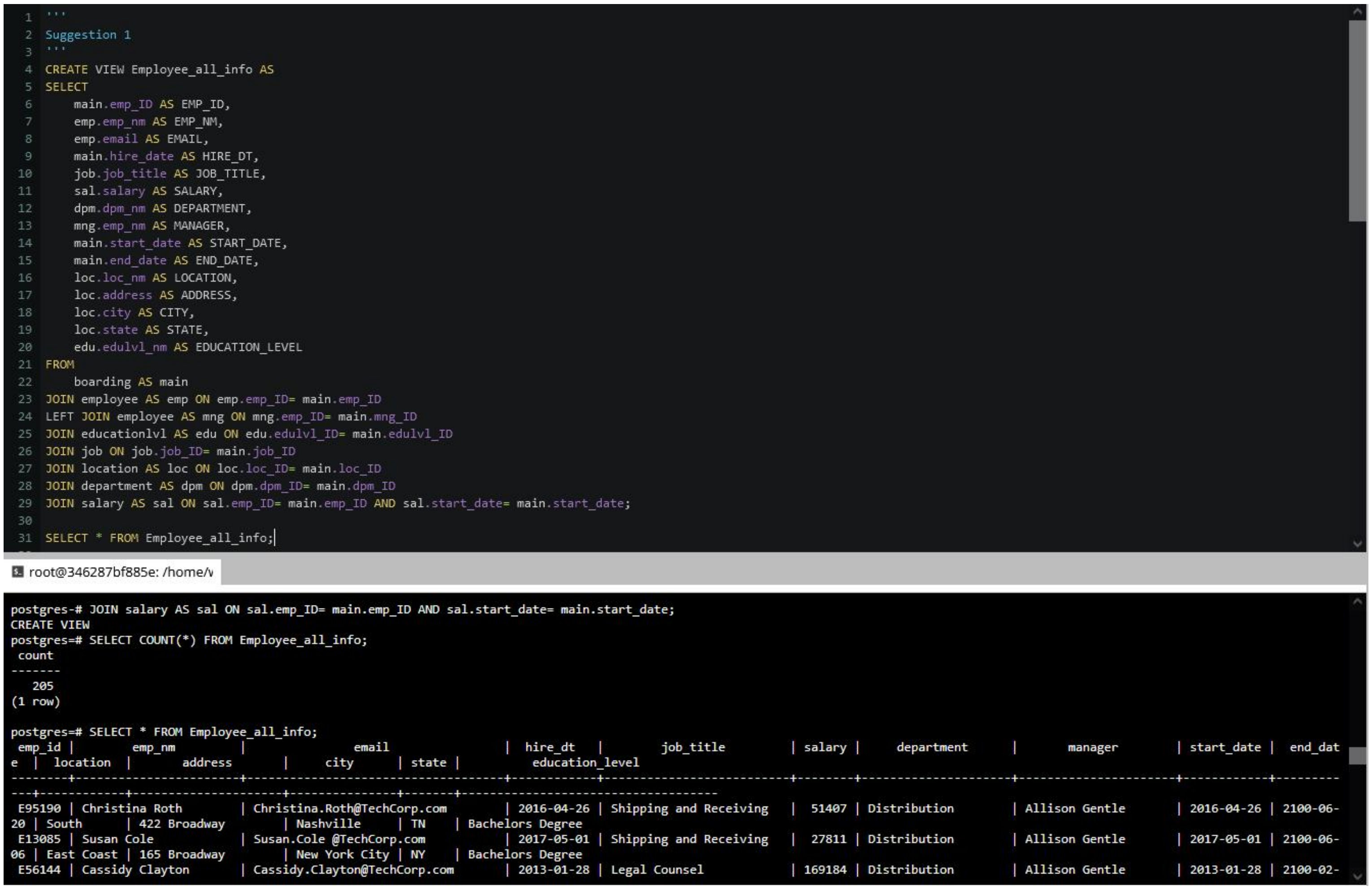
This last step is called Above and Beyond. In this step, I have proposed 3 challenges for you to complete, which are above and beyond the scope of the project. This is a chance to flex your coding muscles and show everyone how good you really are.

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



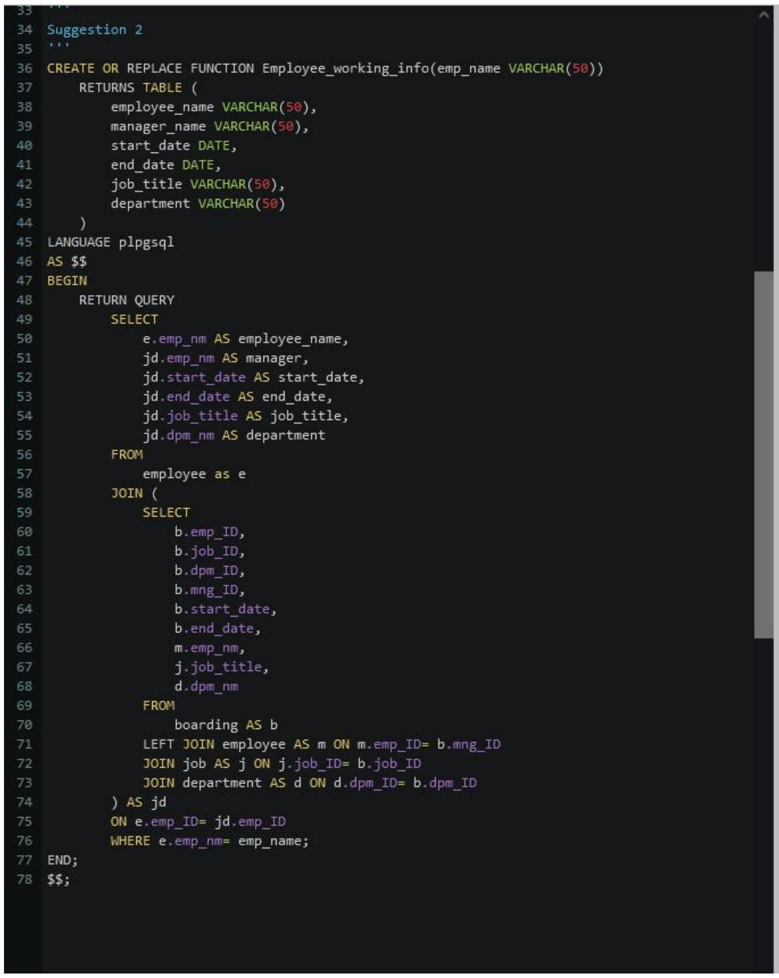
```sql
'''
Suggestion 1
'''
CREATE VIEW Employee_all_info AS
SELECT
    main.emp_ID AS EMP_ID,
    emp.emp_nm AS EMP_NM,
    emp.email AS EMAIL,
    main.hire_date AS HIRE_DT,
    job.job_title AS JOB_TITLE,
    sal.salary AS SALARY,
    dpm.dpm_nm AS DEPARTMENT,
    mng.emp_nm AS MANAGER,
    main.start_date AS START_DATE,
    main.end_date AS END_DATE,
    loc.loc_nm AS LOCATION,
    loc.address AS ADDRESS,
    loc.city AS CITY,
    loc.state AS STATE,
    edu.edulvl_nm AS EDUCATION_LEVEL
FROM
    boarding AS main
JOIN employee AS emp ON emp.emp_ID= main.emp_ID
LEFT JOIN employee AS mng ON mng.emp_ID= main.mng_ID
JOIN educationlvl AS edu ON edu.edulvl_ID= main.edulvl_ID
JOIN job ON job.job_ID= main.job_ID
JOIN location AS loc ON loc.loc_ID= main.loc_ID
JOIN department AS dpm ON dpm.dpm_ID= main.dpm_ID
JOIN salary AS sal ON sal.emp_ID= main.emp_ID AND sal.start_date= main.start_date;

SELECT * FROM Employee_all_info;
```

root@346287bf885e: /home/v

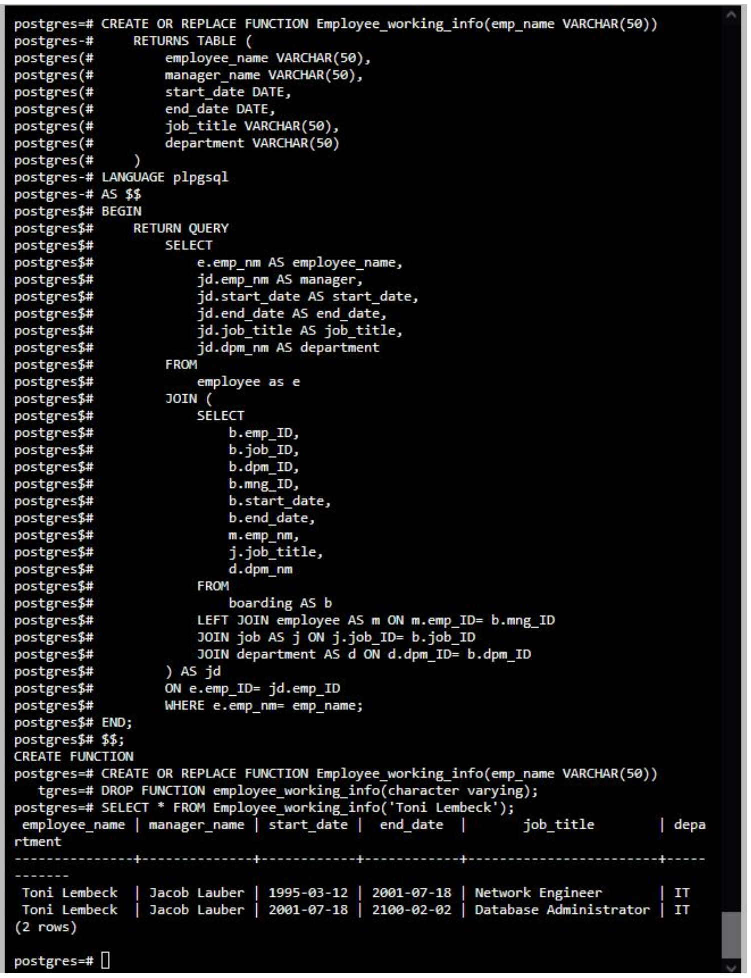
```
postgres-# JOIN salary AS sal ON sal.emp_ID= main.emp_ID AND sal.start_date= main.start_date;
CREATE VIEW
postgres=# SELECT COUNT(*) FROM Employee_all_info;
 count
-------
   205
(1 row)

postgres=# SELECT * FROM Employee_all_info;
 emp_id |        emp_nm        |             email              |  hire_dt   |        job_title        | salary |     department      |       manager        | start_date |  end_dat
e  |  location  |      address       |     city     | state |        education_level
--------+----------------------+--------------------------------+------------+-------------------------+--------+---------------------+----------------------+------------+---------
---+------------+--------------------+--------------+-------+---------------------------------
 E95190 | Christina Roth       | Christina.Roth@TechCorp.com    | 2016-04-26 | Shipping and Receiving  |  51407 | Distribution        | Allison Gentle       | 2016-04-26 | 2100-06-
20 | South      | 422 Broadway       | Nashville    | TN    | Bachelors Degree
 E13085 | Susan Cole           | Susan.Cole @TechCorp.com       | 2017-05-01 | Shipping and Receiving  |  27811 | Distribution        | Allison Gentle       | 2017-05-01 | 2100-06-
06 | East Coast | 165 Broadway       | New York City | NY   | Bachelors Degree
 E56144 | Cassidy Clayton      | Cassidy.Clayton@TechCorp.com   | 2013-01-28 | Legal Counsel           | 169184 | Distribution        | Allison Gentle       | 2013-01-28 | 2100-02-
```

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.



```
Suggestion 2
'''
CREATE OR REPLACE FUNCTION Employee_working_info(emp_name VARCHAR(50))
    RETURNS TABLE (
        employee_name VARCHAR(50),
        manager_name VARCHAR(50),
        start_date DATE,
        end_date DATE,
        job_title VARCHAR(50),
        department VARCHAR(50)
    )
LANGUAGE plpgsql
AS $$
BEGIN
    RETURN QUERY
        SELECT
            e.emp_nm AS employee_name,
            jd.emp_nm AS manager,
            jd.start_date AS start_date,
            jd.end_date AS end_date,
            jd.job_title AS job_title,
            jd.dpm_nm AS department
        FROM
            employee as e
        JOIN (
            SELECT
                b.emp_ID,
                b.job_ID,
                b.dpm_ID,
                b.mng_ID,
                b.start_date,
                b.end_date,
                m.emp_nm,
                j.job_title,
                d.dpm_nm
            FROM
                boarding AS b
            LEFT JOIN employee AS m ON m.emp_ID= b.mng_ID
            JOIN job AS j ON j.job_ID= b.job_ID
            JOIN department AS d ON d.dpm_ID= b.dpm_ID
        ) AS jd
        ON e.emp_ID= jd.emp_ID
        WHERE e.emp_nm= emp_name;
END;
$$;
```



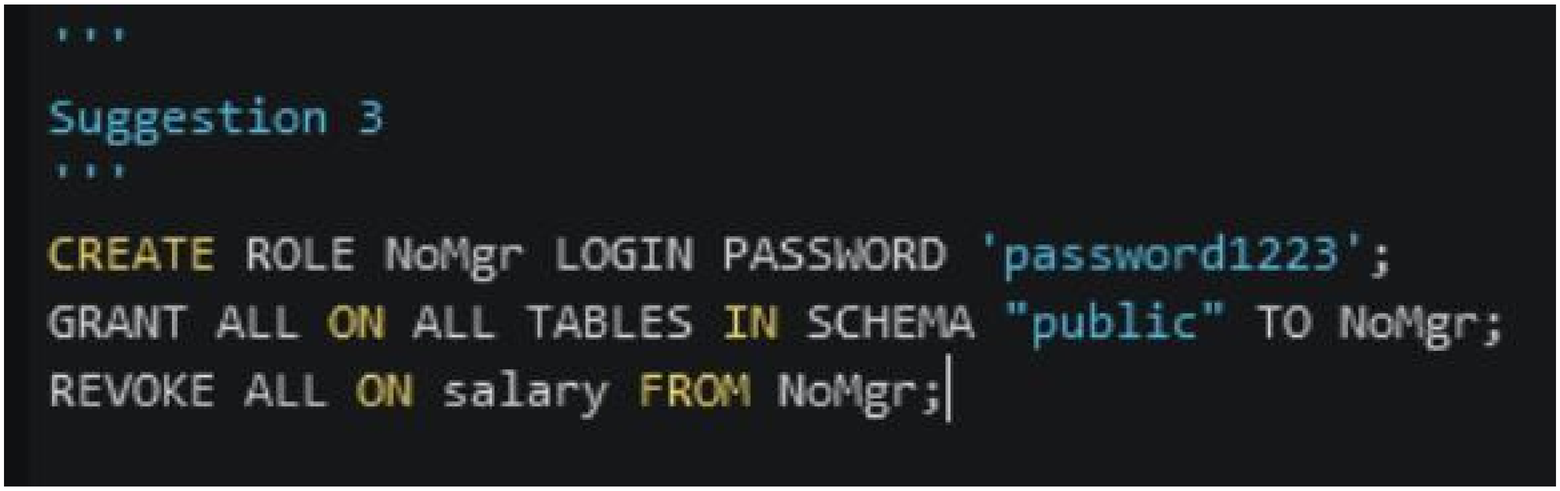
```
postgres=# CREATE OR REPLACE FUNCTION Employee_working_info(emp_name VARCHAR(50))
postgres-#     RETURNS TABLE (
postgres(#         employee_name VARCHAR(50),
postgres(#         manager_name VARCHAR(50),
postgres(#         start_date DATE,
postgres(#         end_date DATE,
postgres(#         job_title VARCHAR(50),
postgres(#         department VARCHAR(50)
postgres(#     )
postgres-# LANGUAGE plpgsql
postgres-# AS $$
postgres$# BEGIN
postgres$#     RETURN QUERY
postgres$#         SELECT
postgres$#             e.emp_nm AS employee_name,
postgres$#             jd.emp_nm AS manager,
postgres$#             jd.start_date AS start_date,
postgres$#             jd.end_date AS end_date,
postgres$#             jd.job_title AS job_title,
postgres$#             jd.dpm_nm AS department
postgres$#         FROM
postgres$#             employee as e
postgres$#         JOIN (
postgres$#             SELECT
postgres$#                 b.emp_ID,
postgres$#                 b.job_ID,
postgres$#                 b.dpm_ID,
postgres$#                 b.mng_ID,
postgres$#                 b.start_date,
postgres$#                 b.end_date,
postgres$#                 m.emp_nm,
postgres$#                 j.job_title,
postgres$#                 d.dpm_nm
postgres$#             FROM
postgres$#                 boarding AS b
postgres$#             LEFT JOIN employee AS m ON m.emp_ID= b.mng_ID
postgres$#             JOIN job AS j ON j.job_ID= b.job_ID
postgres$#             JOIN department AS d ON d.dpm_ID= b.dpm_ID
postgres$#         ) AS jd
postgres$#         ON e.emp_ID= jd.emp_ID
postgres$#         WHERE e.emp_nm= emp_name;
postgres$# END;
postgres$# $$;
CREATE FUNCTION
postgres=# CREATE OR REPLACE FUNCTION Employee_working_info(emp_name VARCHAR(50))
   tgres=# DROP FUNCTION employee_working_info(character varying);
postgres=# SELECT * FROM Employee_working_info('Toni Lembeck');
 employee_name | manager_name | start_date |  end_date  |        job_title        | depa
rtment
---------------+--------------+------------+------------+-------------------------+-----
-------
 Toni Lembeck  | Jacob Lauber | 1995-03-12 | 2001-07-18 | Network Engineer        | IT
 Toni Lembeck  | Jacob Lauber | 2001-07-18 | 2100-02-02 | Database Administrator  | IT
(2 rows)

postgres=# 
```

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



```
'''
Suggestion 3
'''
CREATE ROLE NoMgr LOGIN PASSWORD 'password1223';
GRANT ALL ON ALL TABLES IN SCHEMA "public" TO NoMgr;
REVOKE ALL ON salary FROM NoMgr;
```