Tech ABC Corp - HR Database

James Kenney

6/12/2021

Business Scenario

Business requirement

Tech ABC Corp saw explosive growth with a sudden appearance onto the gaming scene with their new AI-powered video game console. As a result, they have gone from a small 10 person operation to 200 employees and 5 locations in under a year. HR is having trouble keeping up with the growth, since they are still maintaining employee information in a spreadsheet. While that worked for ten employees, it has becoming increasingly cumbersome to manage as the company expands.

As such, the HR department has tasked you, as the new data architect, to design and build a database capable of managing their employee information.

Dataset

The HR dataset you will be working with is an Excel workbook which consists of 206 records, with eleven columns. The data is in human readable format, and has not been normalized at all. The data lists the names of employees at Tech ABC Corp as well as information such as job title, department, manager's name, hire date, start date, end date, work location, and salary.

IT Department Best Practices

The IT Department has certain Best Practices policies for databases you should follow, as detailed in the Best Practices document.

Step 1

Data Architecture Foundations

Step 1: Data Architecture Foundations

Hi,

Welcome to Tech ABC Corp. We are excited to have some new talent onboard. As you may already know, Tech ABC Corp has recently experienced a lot of growth. Our AI powered video game console WOPR has been hugely successful and as a result, our company has grown from 10 employees to 200 in only 6 months (and we are projecting a 20% growth a year for the next 5 years). We have also grown from our Dallas, Texas office, to 4 other locations nationwide: New York City, NY, San Francisco, CA, Minneapolis, MN, and Nashville, TN.

While this growth is great, it is really starting to put a strain on our record keeping in HR. We currently maintain all employee information on a shared spreadsheet. When HR consisted of only myself, managing everyone on an Excel spreadsheet was simple, but now that it is a shared document I am having serious reservations about data integrity and data security. If the wrong person got their hands on the HR file, they would see the salaries of every employee in the company, all the way up to the president.

After speaking with Jacob Lauber, the manager of IT, he suggested I put in a request to have my HR Excel file converted into a database. He suggested I reach out to you as I am told you have experience in designing and building databases. When you are building this, please keep in mind that I want any employee with a domain login to be have read only access the database. I just don't want them having access to salary information. That needs to be restricted to HR and management level employees only. Management and HR employees should also be the only ones with write access. By our current estimates, 90% of users will be read only.

I also want to make sure you know that am looking to turn my spreadsheet into a live database, one I can input and edit information into. I am not really concerned with reporting capabilities at the moment. Since we are working with employee data we are required by federal regulations to maintain this data for at least 7 years; additionally, since this is considered business critical data, we need to make sure it gets backed up properly.

As a final consideration. We would like to be able to connect with the payroll department's system in the future. They maintain employee attendance and paid time off information. It would be nice if the two systems could interface in the future

I am looking forward to working with you and seeing what kind of database you design for us.

Thanks,
Sarah Collins
Head of HR

Data Architect Business Requirement

- **Purpose of the new database:**

  The purpose is for more efficient record keeping by HR as the business scales due to rapid growth and projected growth over the next five years. Additionally, a database can provide enhanced security and data integrity.

- **Describe current data management solution:**

  The current method is an excel file that stores employee information.

- **Describe current data available:**

  Employee data: contained in excel that consists of two hundred employees and all pertinent employee information.

- **Additional data requests:**

  Security: restrict salary table to Management and HR. Write access as well for managers and HR. Additionally, database should be backed up and a possible future 1 to 1 connection for payroll.

- **Who will own/manage data**

  The data in this databse will be owned and managed by HR.

- **Who will have access to database**

  All employees with a domain login. HR and Management will have full access while employees willl have read only access.

Data Architect Business Requirement

- **Estimated size of database**

  There will be 206 rows with 11 columns.

- **Estimated annual growth**

  Growth is estimated to be 20% annually over the next five years.

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

  The salary table will be considered sensitive and restricted to only HR and Management.

Data Architect Technical Requirement

- **Justification for the new database**

    1. Data integrity/security-both are a risk with a spreadsheet.

    2. Scalibiity-20% annually over the next 5 years.

- **Database objects**

    The database will consist of the following tables and reporting views:

    Tables:

    1. Employee (emp id, name, email, hire date)

    2. Job Title (job title id, job title)

    3. Department (department id, department name)

    4. Location (location id, location, state, city, address)

    5. Education (education id, education level)

    6. Salary (salary id, salary)

    7. Employee Map (table foreign key references for joining tables in addition to start & end date for job history)

- **Data ingestion**

    Data will be ingested via ETL process given a flat fle is used to capture info intially and then pulled into the tables.

Data Architect Technical Requirement

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

  Ownership: HR will own the database.

  User Access: All employees with a domain login will have acess though only HR and management will have write access in addition to salary table access.

- **Scalability**

  Given this is a dedicated database for HR, replication is the best solution as having a copy of the database ensures that when new data is added, it will be copied from the master database and the organization scales.

- **Flexibility**

  Entities are designed in a 3NF model to anticipate growth and sure data integrity. This will allow for easier integrations in the future if the need arises.

- **Storage & retention**

  Storage (disk or in-memory): Disk storage would be the best option given there is no need for any computations. This method will reduce costs and meet the current and anticipated needs of the organization.

  Retention: At least 7 years due to federal regulation.

- **Backup:** Critical back up is the best option as it will ensure a full back up once a week and a daily back up once a day. As the organization grows, it is critical to back up the data in a timely fashion.

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.

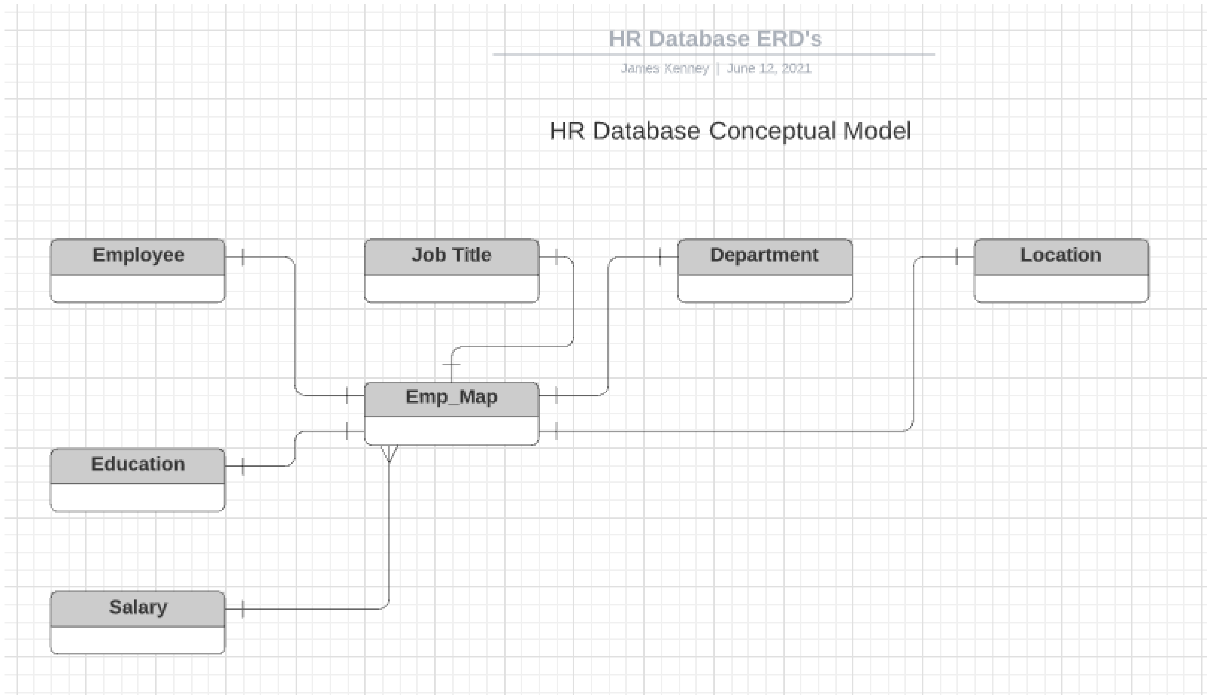
ERD

- **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 Model



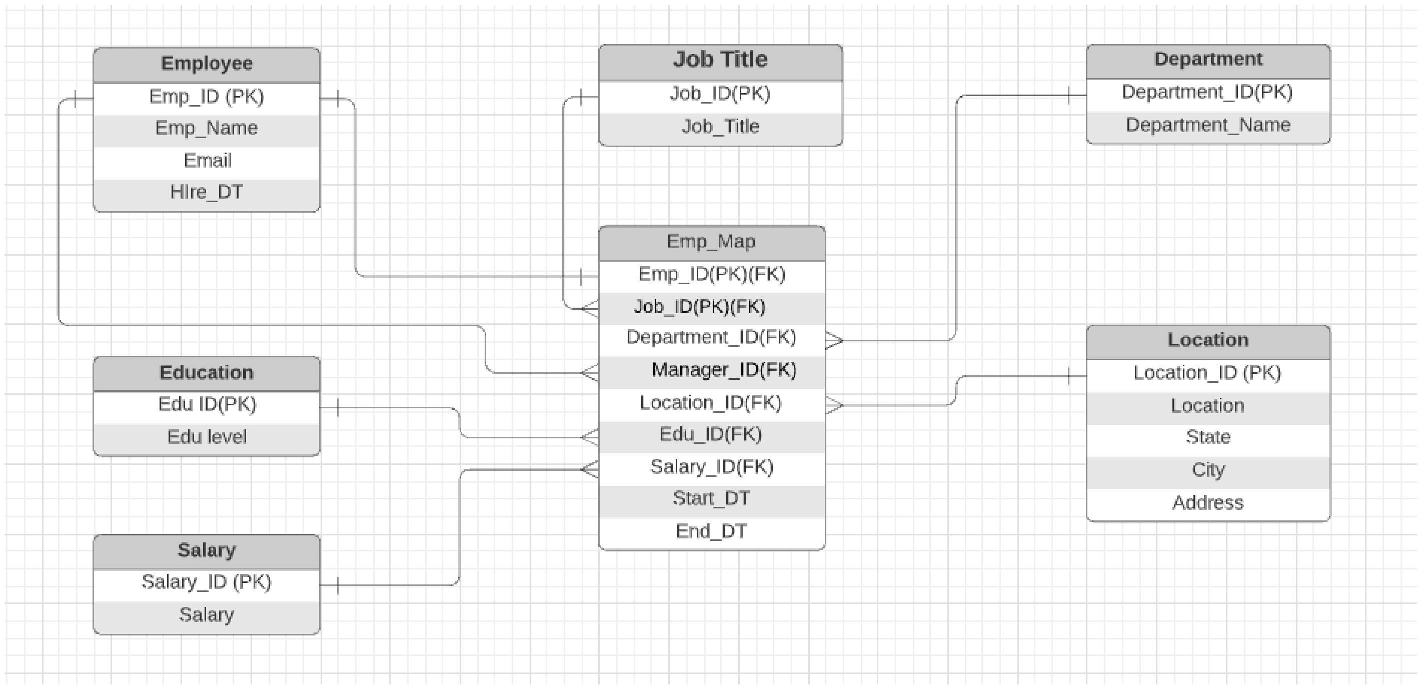
ERD

- **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.

Logical Model

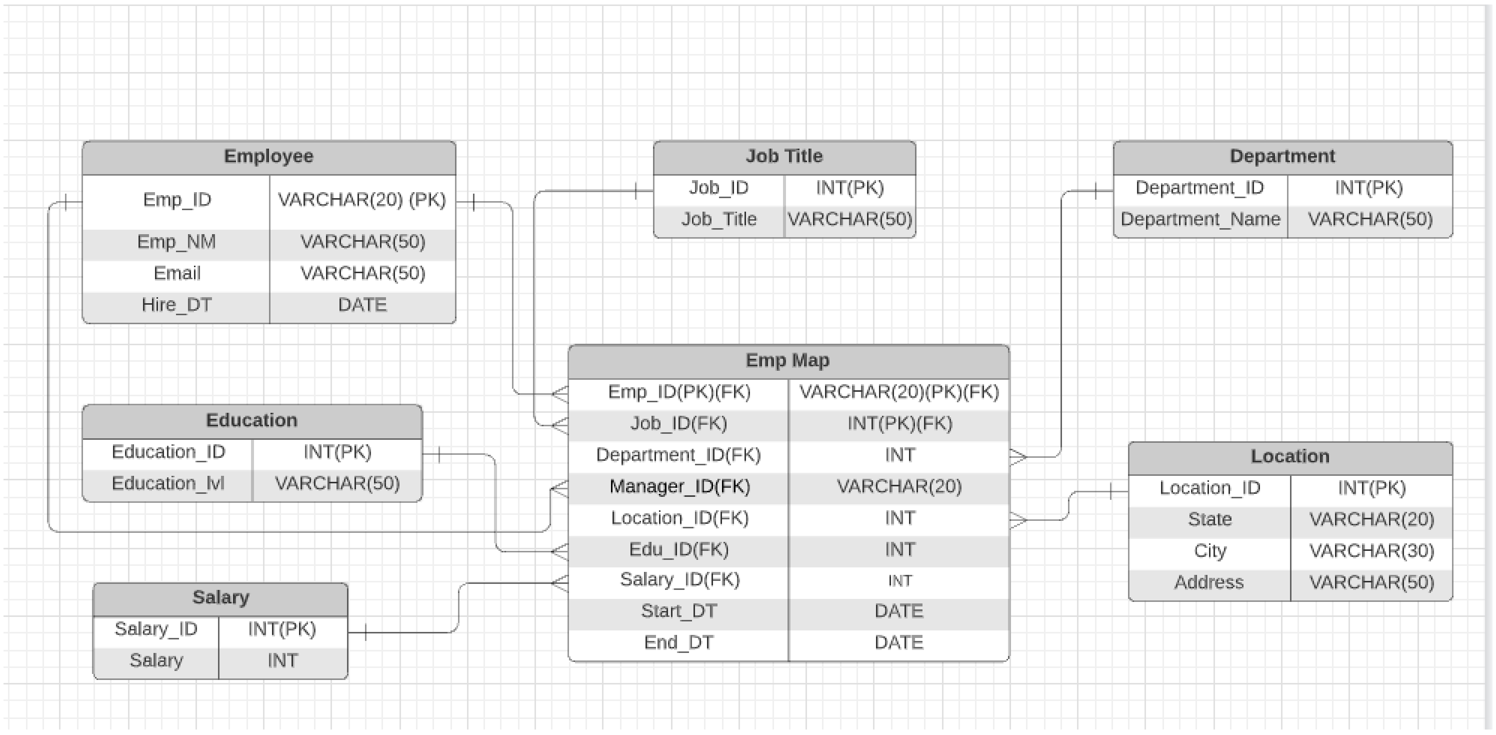


ERD

- ## 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.

Physical Model



Employee	
Emp_ID	VARCHAR(20) (PK)
Emp_NM	VARCHAR(50)
Email	VARCHAR(50)
Hire_DT	DATE

Job Title	
Job_ID	INT(PK)
Job_Title	VARCHAR(50)

Department	
Department_ID	INT(PK)
Department_Name	VARCHAR(50)

Education	
Education_ID	INT(PK)
Education_lvl	VARCHAR(50)

Emp Map	
Emp_ID(PK)(FK)	VARCHAR(20)(PK)(FK)
Job_ID(FK)	INT(PK)(FK)
Department_ID(FK)	INT
Manager_ID(FK)	VARCHAR(20)
Location_ID(FK)	INT
Edu_ID(FK)	INT
Salary_ID(FK)	INT
Start_DT	DATE
End_DT	DATE

Location	
Location_ID	INT(PK)
State	VARCHAR(20)
City	VARCHAR(30)
Address	VARCHAR(50)

Salary	
Salary_ID	INT(PK)
Salary	INT

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

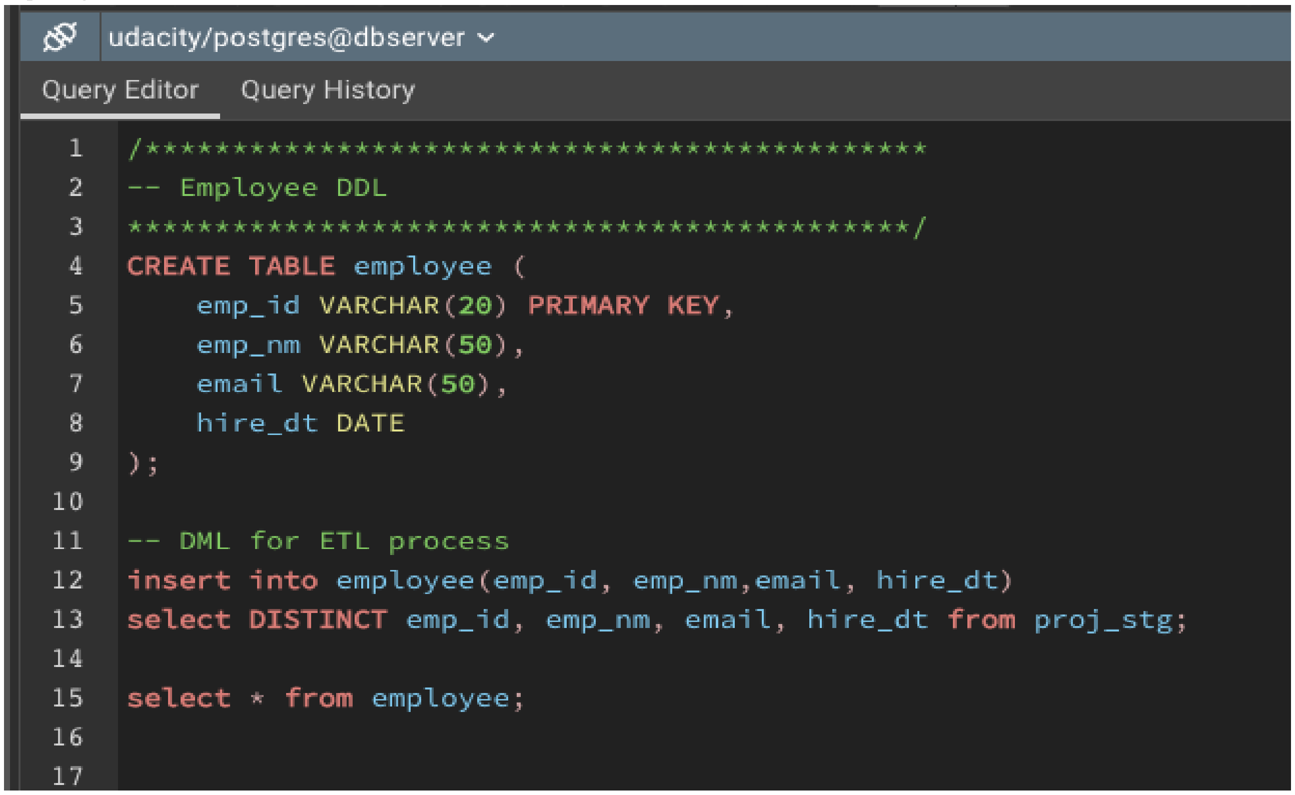
Create a DDL SQL script capable of building the database you designed in Step 2

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.

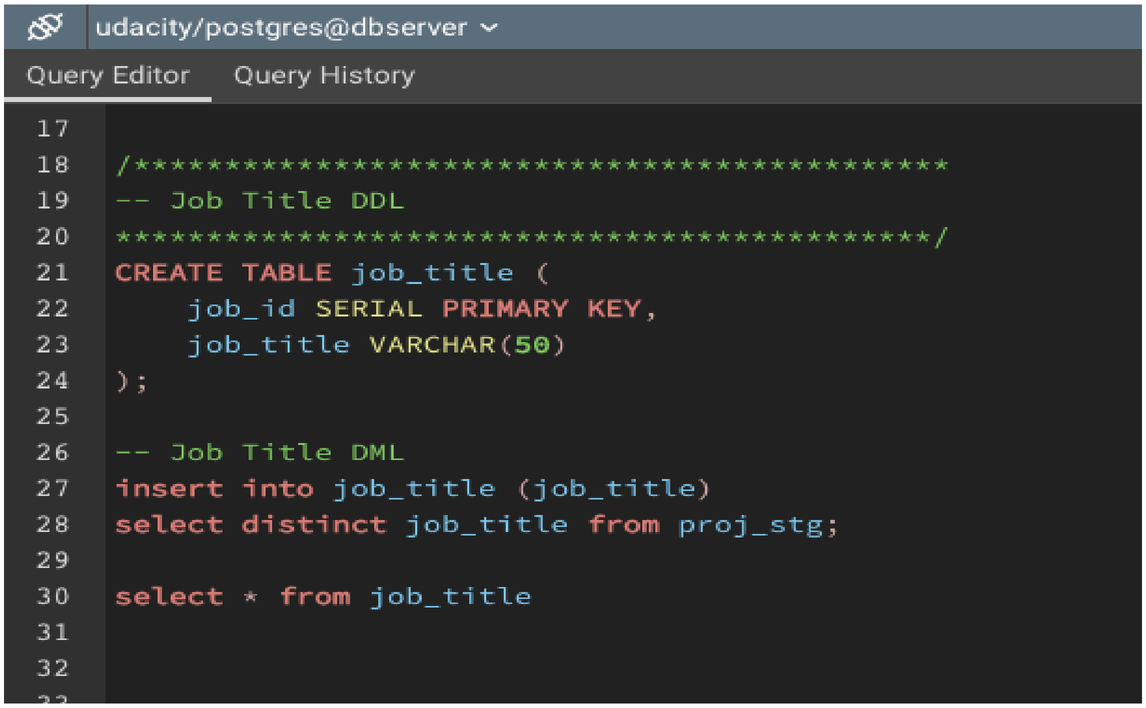
DDL Tables: * all queries and DB were created in PostgresSQL

Employee



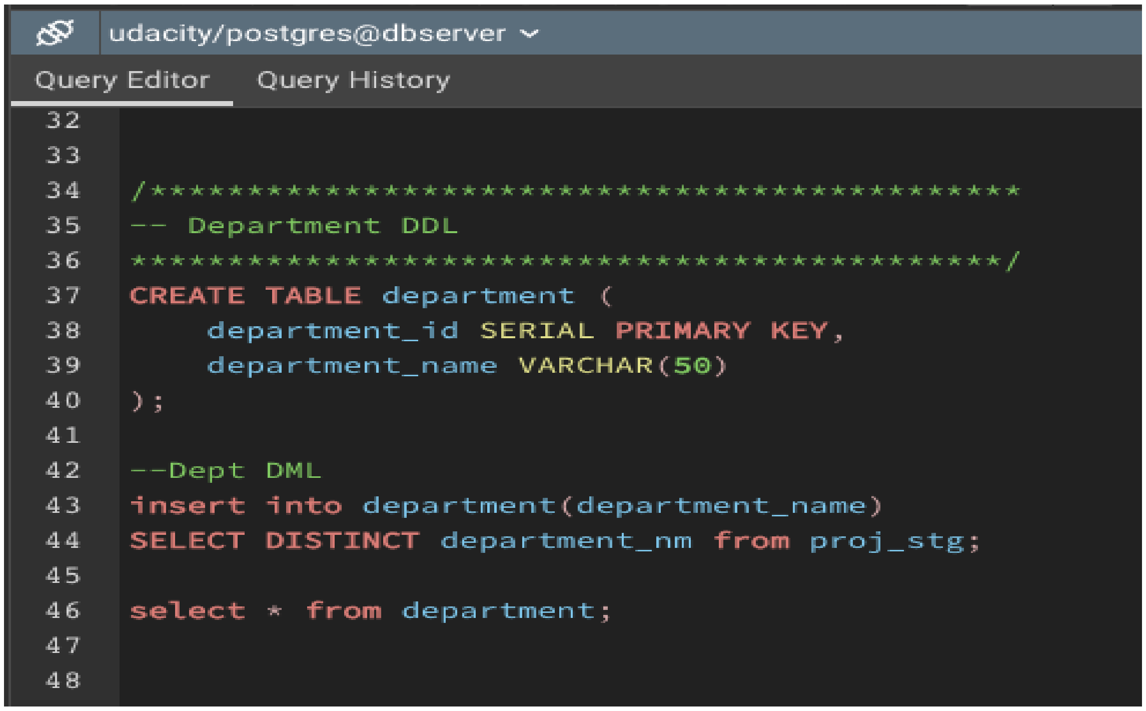
```sql
/*******************************************
-- Employee DDL
*******************************************/
CREATE TABLE employee (
    emp_id VARCHAR(20) PRIMARY KEY,
    emp_nm VARCHAR(50),
    email VARCHAR(50),
    hire_dt DATE
);

-- DML for ETL process
insert into employee(emp_id, emp_nm,email, hire_dt)
select DISTINCT emp_id, emp_nm, email, hire_dt from proj_stg;

select * from employee;
```

Job Title



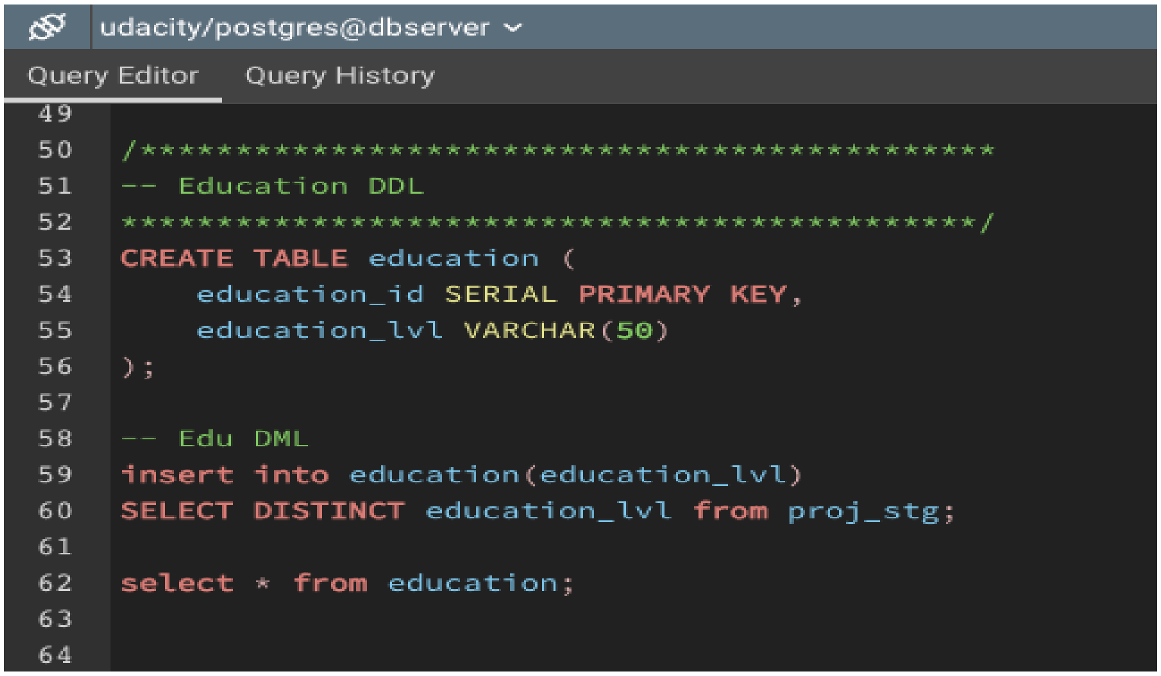
```
/*********************************************
-- Job Title DDL
*********************************************/
CREATE TABLE job_title (
    job_id SERIAL PRIMARY KEY,
    job_title VARCHAR(50)
);

-- Job Title DML
insert into job_title (job_title)
select distinct job_title from proj_stg;

select * from job_title
```

Department



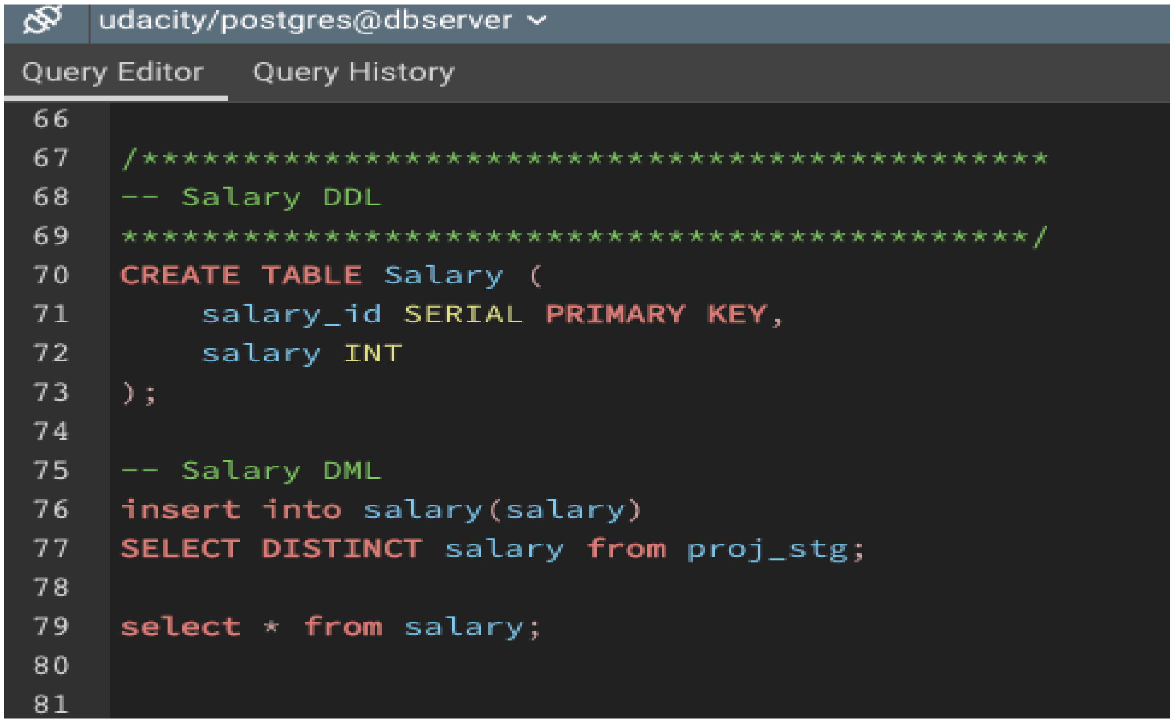
```
/***********************************************
-- Department DDL
***********************************************/
CREATE TABLE department (
    department_id SERIAL PRIMARY KEY,
    department_name VARCHAR(50)
);

--Dept DML
insert into department(department_name)
SELECT DISTINCT department_nm from proj_stg;

select * from department;
```

Education



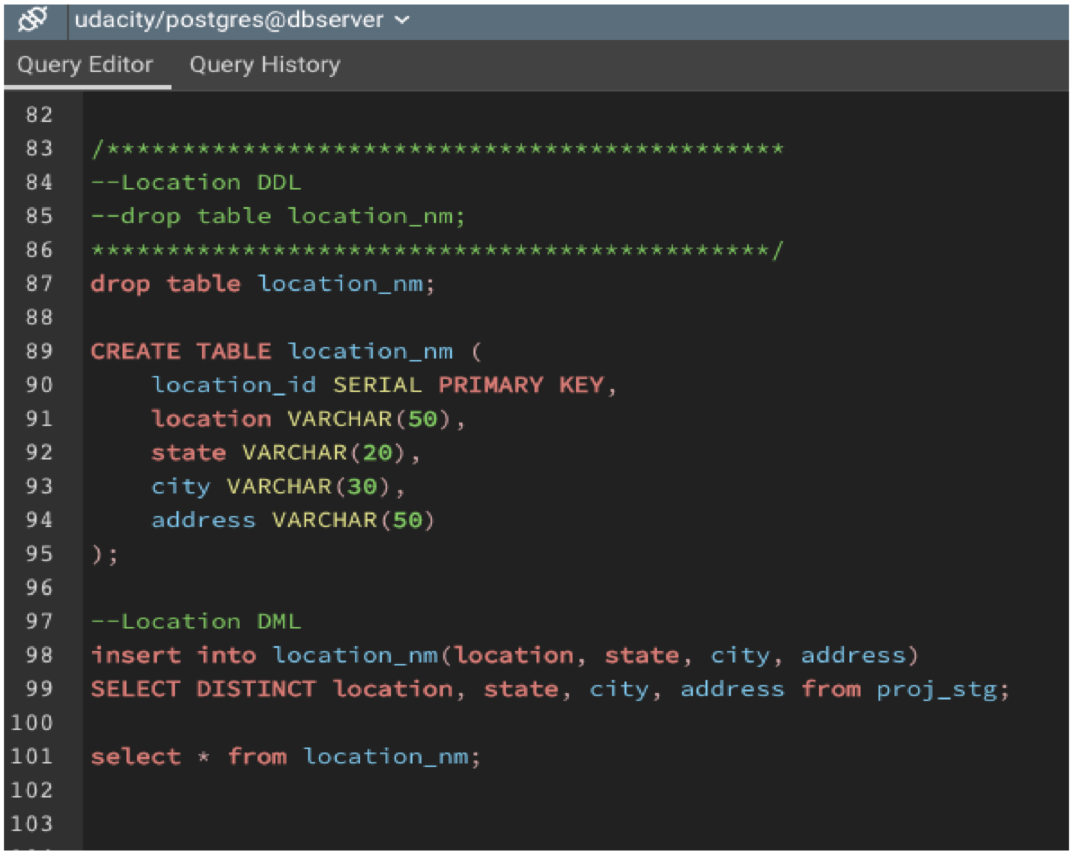
```
/*********************************************
-- Education DDL
*********************************************/
CREATE TABLE education (
    education_id SERIAL PRIMARY KEY,
    education_lvl VARCHAR(50)
);

-- Edu DML
insert into education(education_lvl)
SELECT DISTINCT education_lvl from proj_stg;

select * from education;
```

Salary



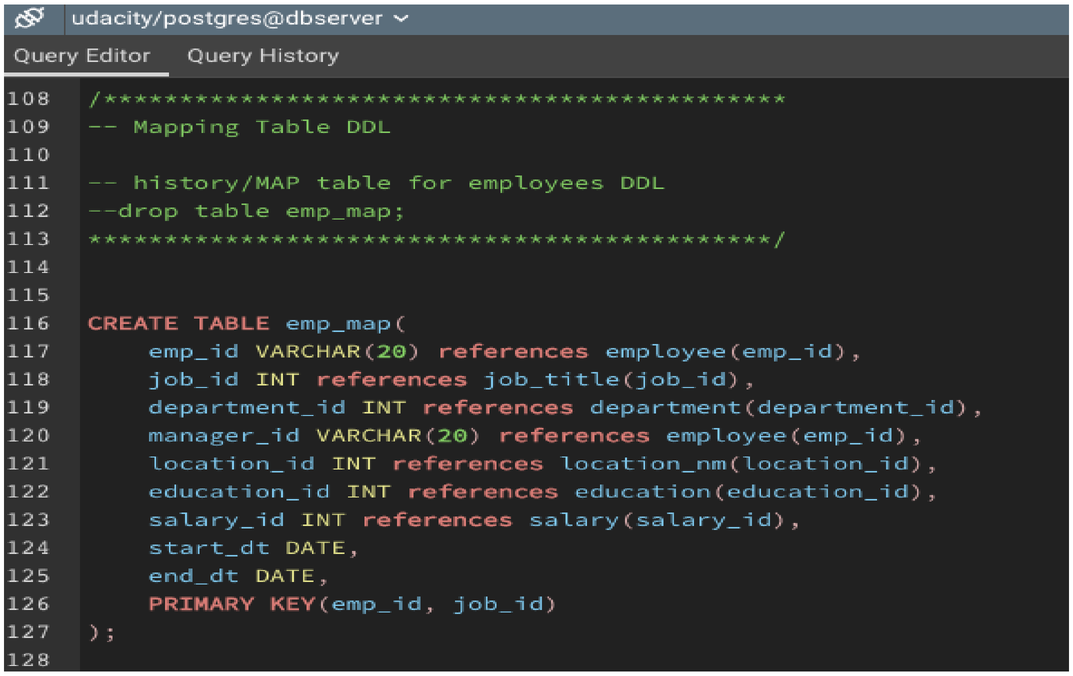
```
/*********************************************
-- Salary DDL
*********************************************/
CREATE TABLE Salary (
    salary_id SERIAL PRIMARY KEY,
    salary INT
);

-- Salary DML
insert into salary(salary)
SELECT DISTINCT salary from proj_stg;

select * from salary;
```

Location

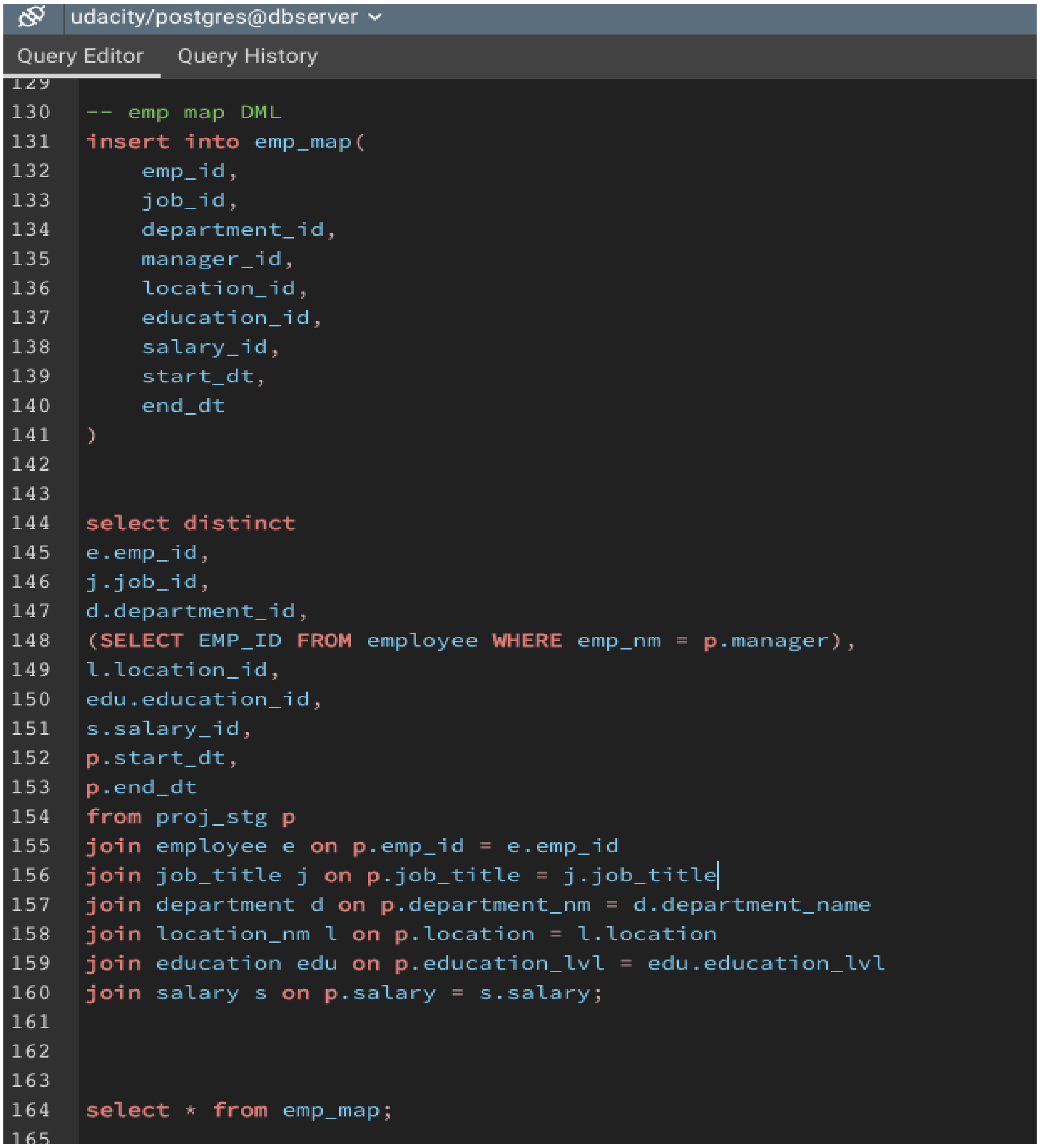


```

/*********************************************
--Location DDL
--drop table location_nm;
*********************************************/
drop table location_nm;

CREATE TABLE location_nm (
    location_id SERIAL PRIMARY KEY,
    location VARCHAR(50),
    state VARCHAR(20),
    city VARCHAR(30),
    address VARCHAR(50)
);

--Location DML
insert into location_nm(location, state, city, address)
SELECT DISTINCT location, state, city, address from proj_stg;

select * from location_nm;


```

Emp Map



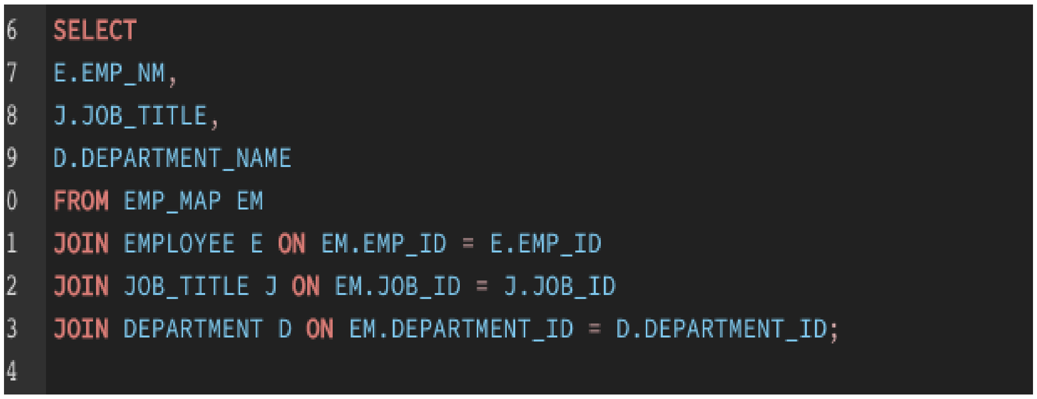
```
/*********************************************
-- Mapping Table DDL

-- history/MAP table for employees DDL
--drop table emp_map;
*********************************************/


CREATE TABLE emp_map(
    emp_id VARCHAR(20) references employee(emp_id),
    job_id INT references job_title(job_id),
    department_id INT references department(department_id),
    manager_id VARCHAR(20) references employee(emp_id),
    location_id INT references location_nm(location_id),
    education_id INT references education(education_id),
    salary_id INT references salary(salary_id),
    start_dt DATE,
    end_dt DATE,
    PRIMARY KEY(emp_id, job_id)
);

```



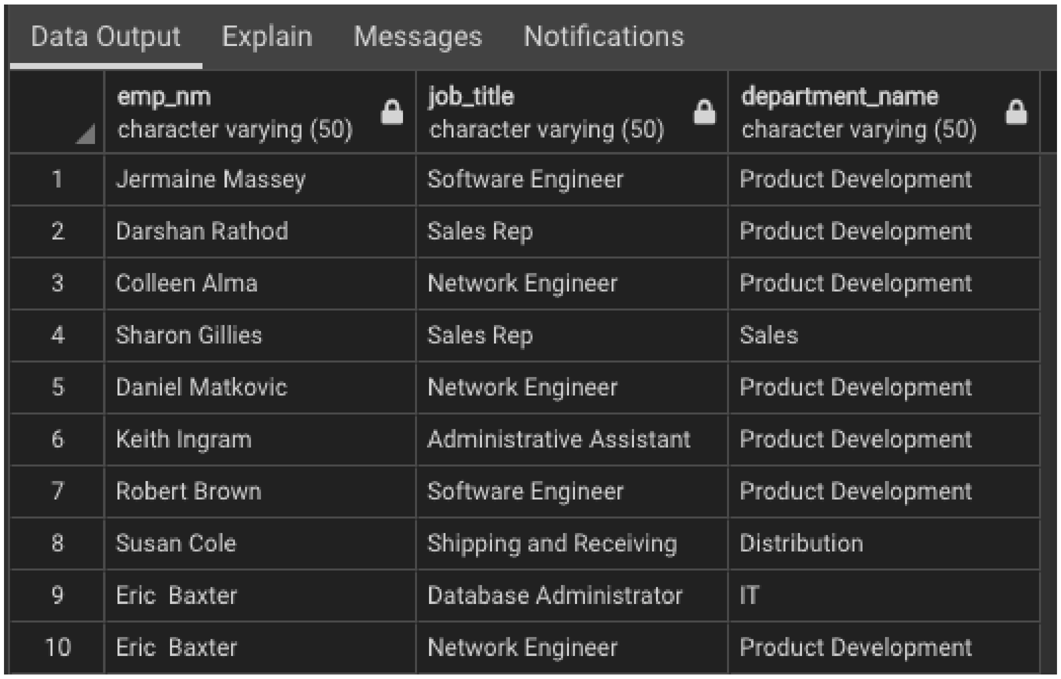
```sql
-- emp map DML
insert into emp_map(
    emp_id,
    job_id,
    department_id,
    manager_id,
    location_id,
    education_id,
    salary_id,
    start_dt,
    end_dt
)


select distinct
e.emp_id,
j.job_id,
d.department_id,
(SELECT EMP_ID FROM employee WHERE emp_nm = p.manager),
l.location_id,
edu.education_id,
s.salary_id,
p.start_dt,
p.end_dt
from proj_stg p
join employee e on p.emp_id = e.emp_id
join job_title j on p.job_title = j.job_title
join department d on p.department_nm = d.department_name
join location_nm l on p.location = l.location
join education edu on p.education_lvl = edu.education_lvl
join salary s on p.salary = s.salary;



select * from emp_map;
```

CRUD

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



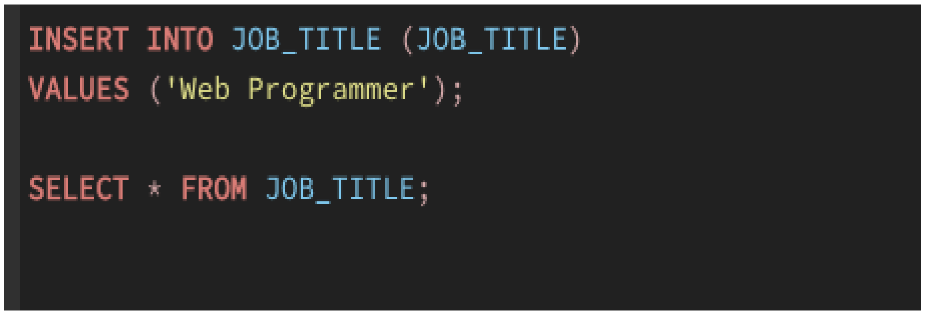
```sql
SELECT
E.EMP_NM,
J.JOB_TITLE,
D.DEPARTMENT_NAME
FROM EMP_MAP EM
JOIN EMPLOYEE E ON EM.EMP_ID = E.EMP_ID
JOIN JOB_TITLE J ON EM.JOB_ID = J.JOB_ID
JOIN DEPARTMENT D ON EM.DEPARTMENT_ID = D.DEPARTMENT_ID;
```



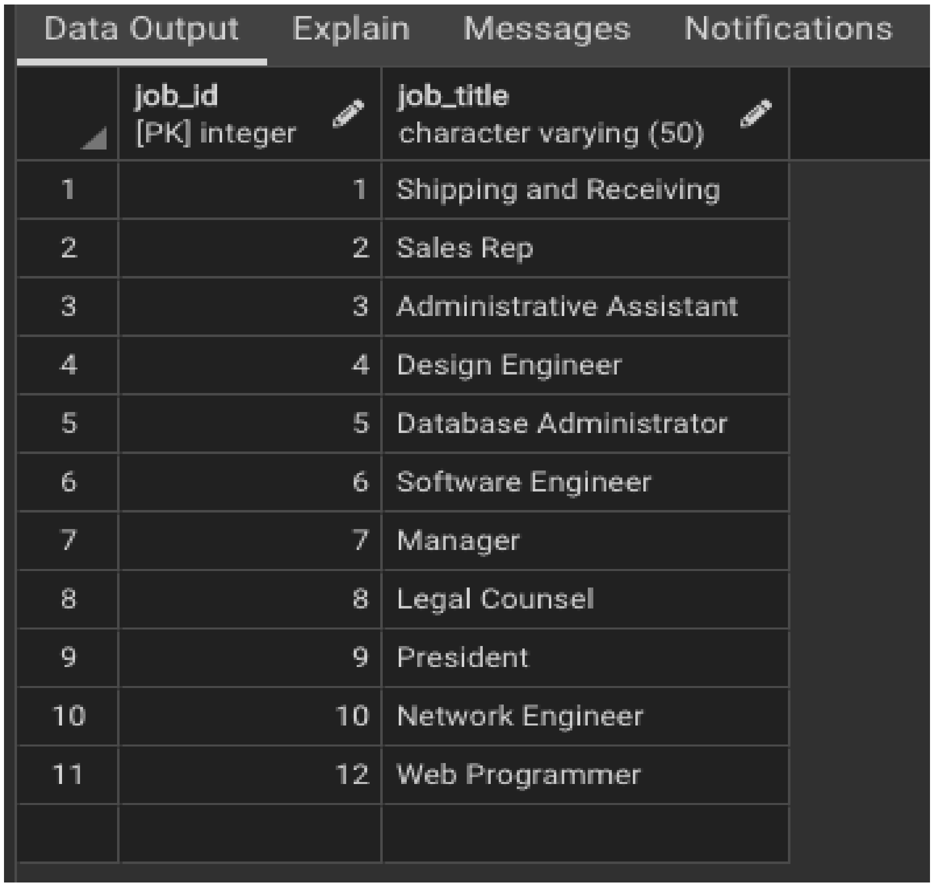
	emp_nm character varying (50)	job_title character varying (50)	department_name character varying (50)
1	Jermaine Massey	Software Engineer	Product Development
2	Darshan Rathod	Sales Rep	Product Development
3	Colleen Alma	Network Engineer	Product Development
4	Sharon Gillies	Sales Rep	Sales
5	Daniel Matkovic	Network Engineer	Product Development
6	Keith Ingram	Administrative Assistant	Product Development
7	Robert Brown	Software Engineer	Product Development
8	Susan Cole	Shipping and Receiving	Distribution
9	Eric Baxter	Database Administrator	IT
10	Eric Baxter	Network Engineer	Product Development

CRUD

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



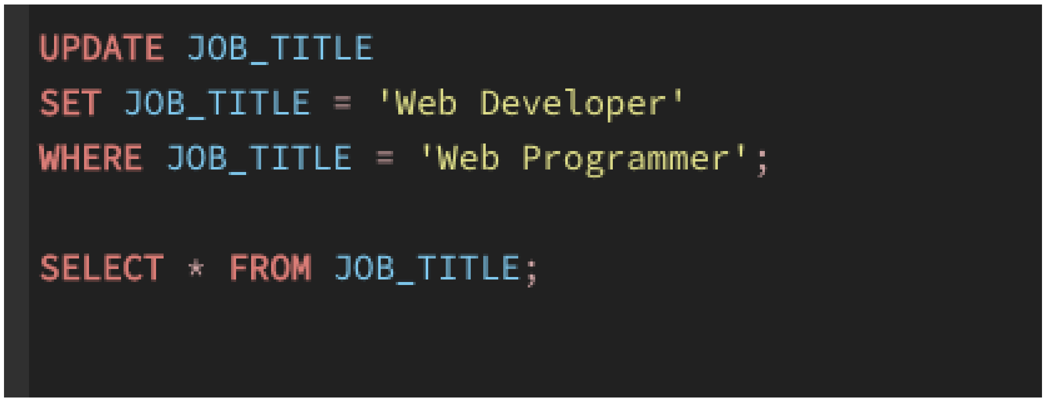
```
INSERT INTO JOB_TITLE (JOB_TITLE)
VALUES ('Web Programmer');

SELECT * FROM JOB_TITLE;
```



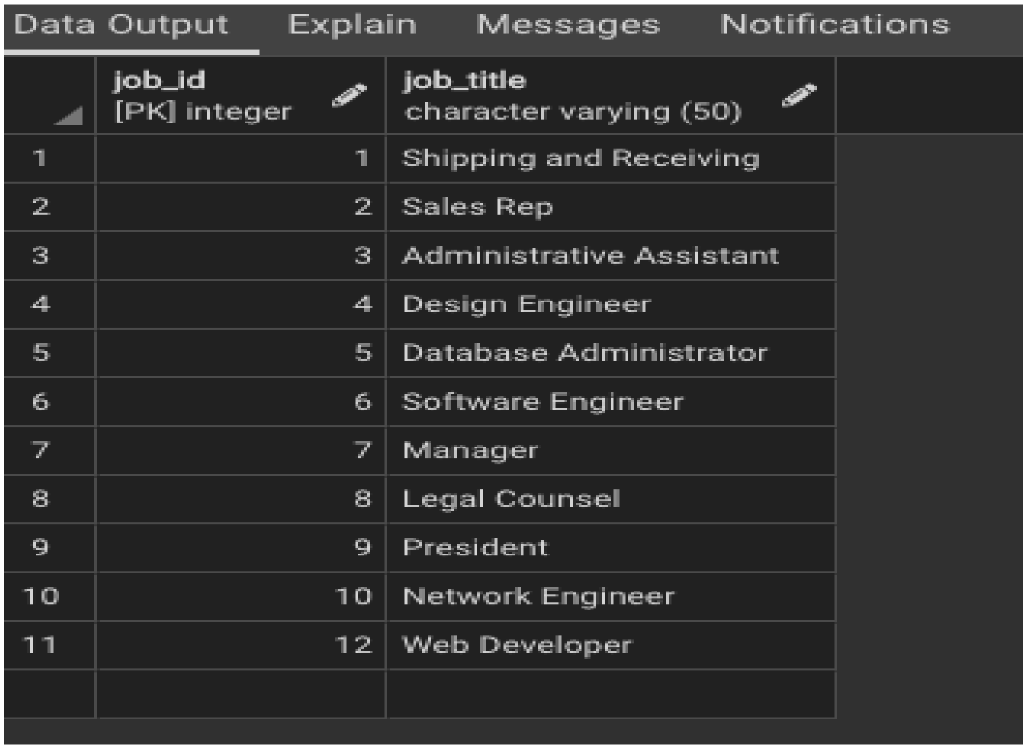
	job_id [PK] integer	job_title character varying (50)
1	1	Shipping and Receiving
2	2	Sales Rep
3	3	Administrative Assistant
4	4	Design Engineer
5	5	Database Administrator
6	6	Software Engineer
7	7	Manager
8	8	Legal Counsel
9	9	President
10	10	Network Engineer
11	12	Web Programmer

CRUD

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



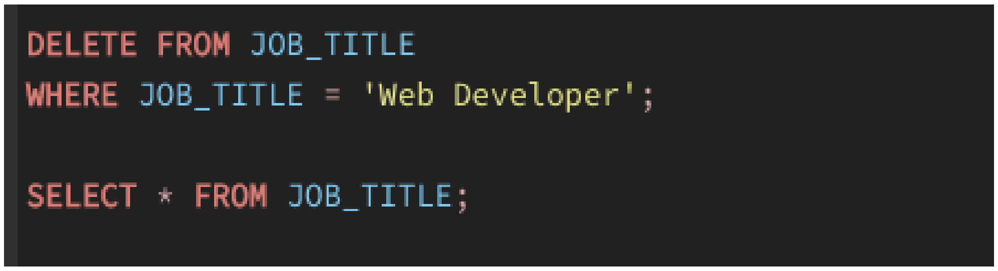
```
UPDATE JOB_TITLE
SET JOB_TITLE = 'Web Developer'
WHERE JOB_TITLE = 'Web Programmer';

SELECT * FROM JOB_TITLE;
```



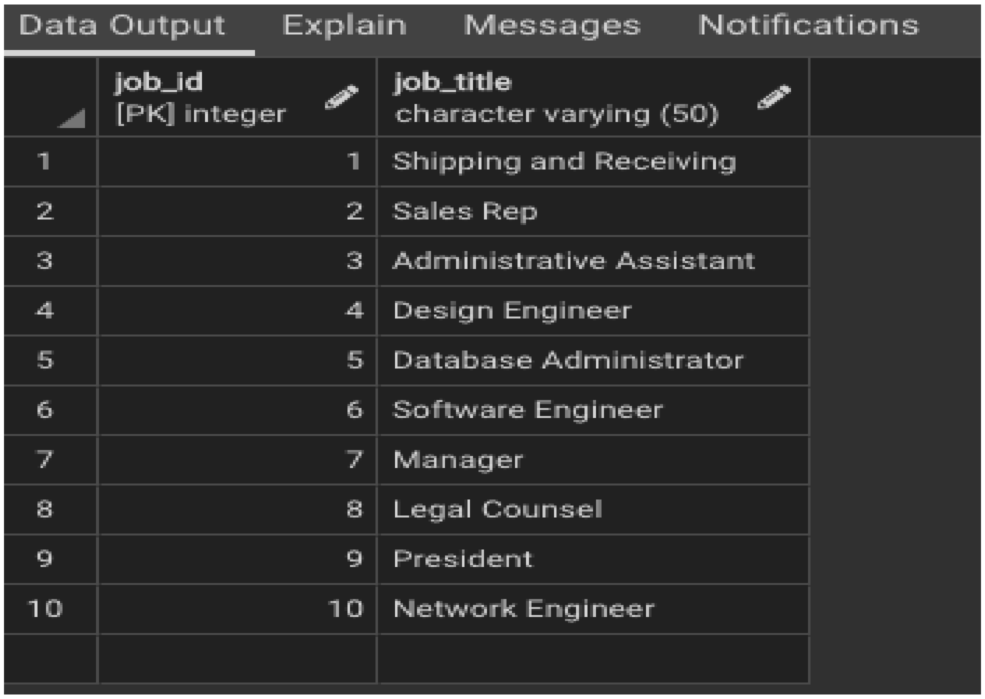
	job_id [PK] integer	job_title character varying (50)
1	1	Shipping and Receiving
2	2	Sales Rep
3	3	Administrative Assistant
4	4	Design Engineer
5	5	Database Administrator
6	6	Software Engineer
7	7	Manager
8	8	Legal Counsel
9	9	President
10	10	Network Engineer
11	12	Web Developer

CRUD

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



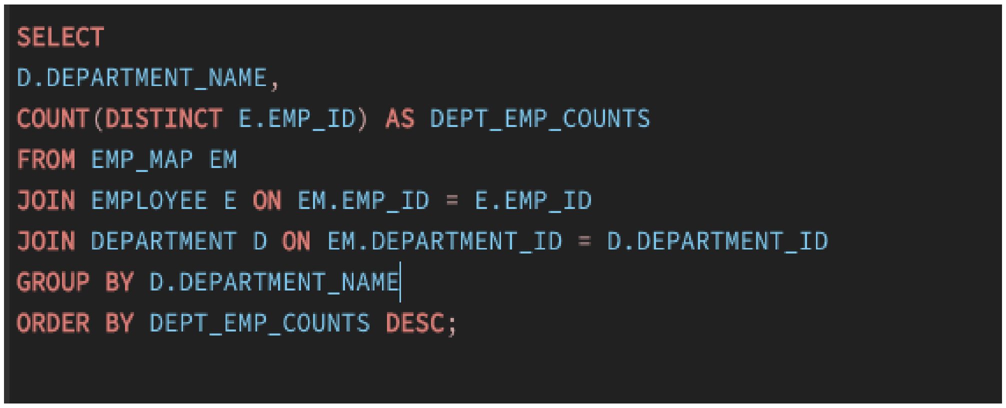
```
DELETE FROM JOB_TITLE
WHERE JOB_TITLE = 'Web Developer';

SELECT * FROM JOB_TITLE;
```



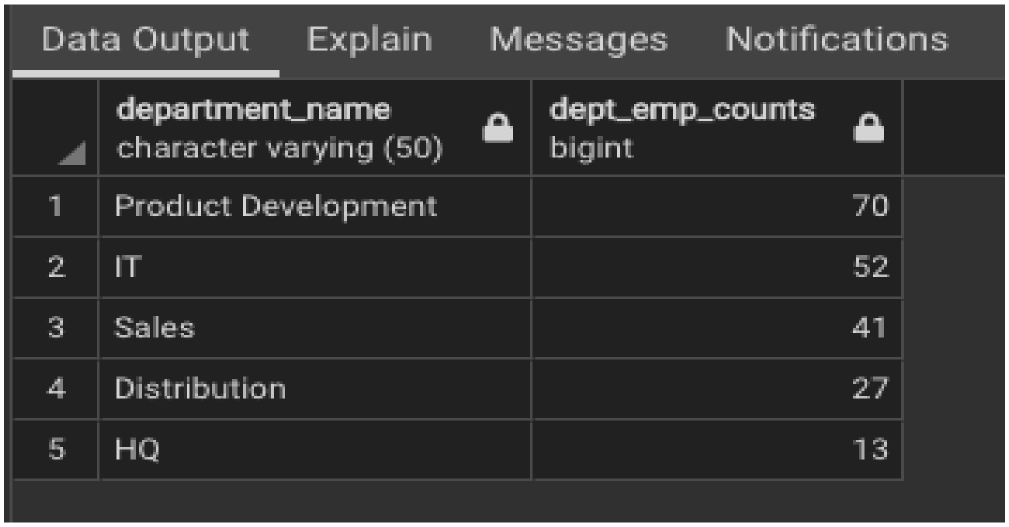
	job_id [PK] integer	job_title character varying (50)
1	1	Shipping and Receiving
2	2	Sales Rep
3	3	Administrative Assistant
4	4	Design Engineer
5	5	Database Administrator
6	6	Software Engineer
7	7	Manager
8	8	Legal Counsel
9	9	President
10	10	Network Engineer

CRUD

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



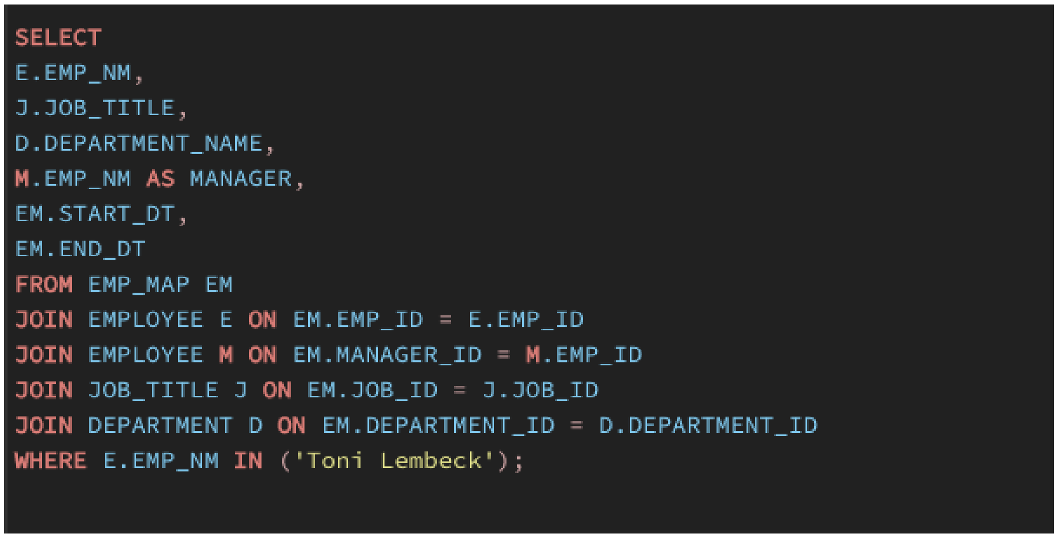
```
SELECT
D.DEPARTMENT_NAME,
COUNT(DISTINCT E.EMP_ID) AS DEPT_EMP_COUNTS
FROM EMP_MAP EM
JOIN EMPLOYEE E ON EM.EMP_ID = E.EMP_ID
JOIN DEPARTMENT D ON EM.DEPARTMENT_ID = D.DEPARTMENT_ID
GROUP BY D.DEPARTMENT_NAME
ORDER BY DEPT_EMP_COUNTS DESC;
```



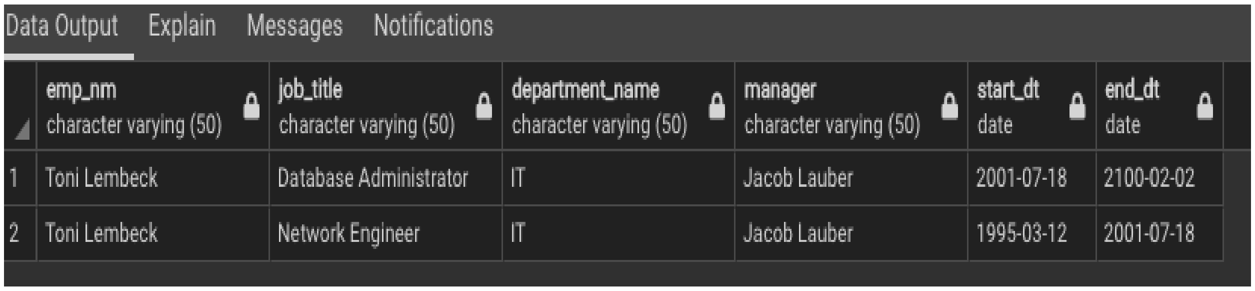
	department_name character varying (50)	dept_emp_counts bigint
1	Product Development	70
2	IT	52
3	Sales	41
4	Distribution	27
5	HQ	13

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,
J.JOB_TITLE,
D.DEPARTMENT_NAME,
M.EMP_NM AS MANAGER,
EM.START_DT,
EM.END_DT
FROM EMP_MAP EM
JOIN EMPLOYEE E ON EM.EMP_ID = E.EMP_ID
JOIN EMPLOYEE M ON EM.MANAGER_ID = M.EMP_ID
JOIN JOB_TITLE J ON EM.JOB_ID = J.JOB_ID
JOIN DEPARTMENT D ON EM.DEPARTMENT_ID = D.DEPARTMENT_ID
WHERE E.EMP_NM IN ('Toni Lembeck');
```



	emp_nm character varying (50)	job_title character varying (50)	department_name character varying (50)	manager character varying (50)	start_dt date	end_dt date
1	Toni Lembeck	Database Administrator	IT	Jacob Lauber	2001-07-18	2100-02-02
2	Toni Lembeck	Network Engineer	IT	Jacob Lauber	1995-03-12	2001-07-18

CRUD

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

Table level security can be achieved by revoking access to read only users and granting access to specific users. Typically we can implement Database level, table level, row level and column level security but given we want to restrict the entire table from employees, it would be best to use the user role and restrict for read only.

This is achieved by creating the role within the domain and applying the table level security constraints. To use this method, the users would be restrained by their user id within the domain by using GRANT and REVOKE.

Step 4

Above and Beyond

(optional)

Step 4: Above and Beyond

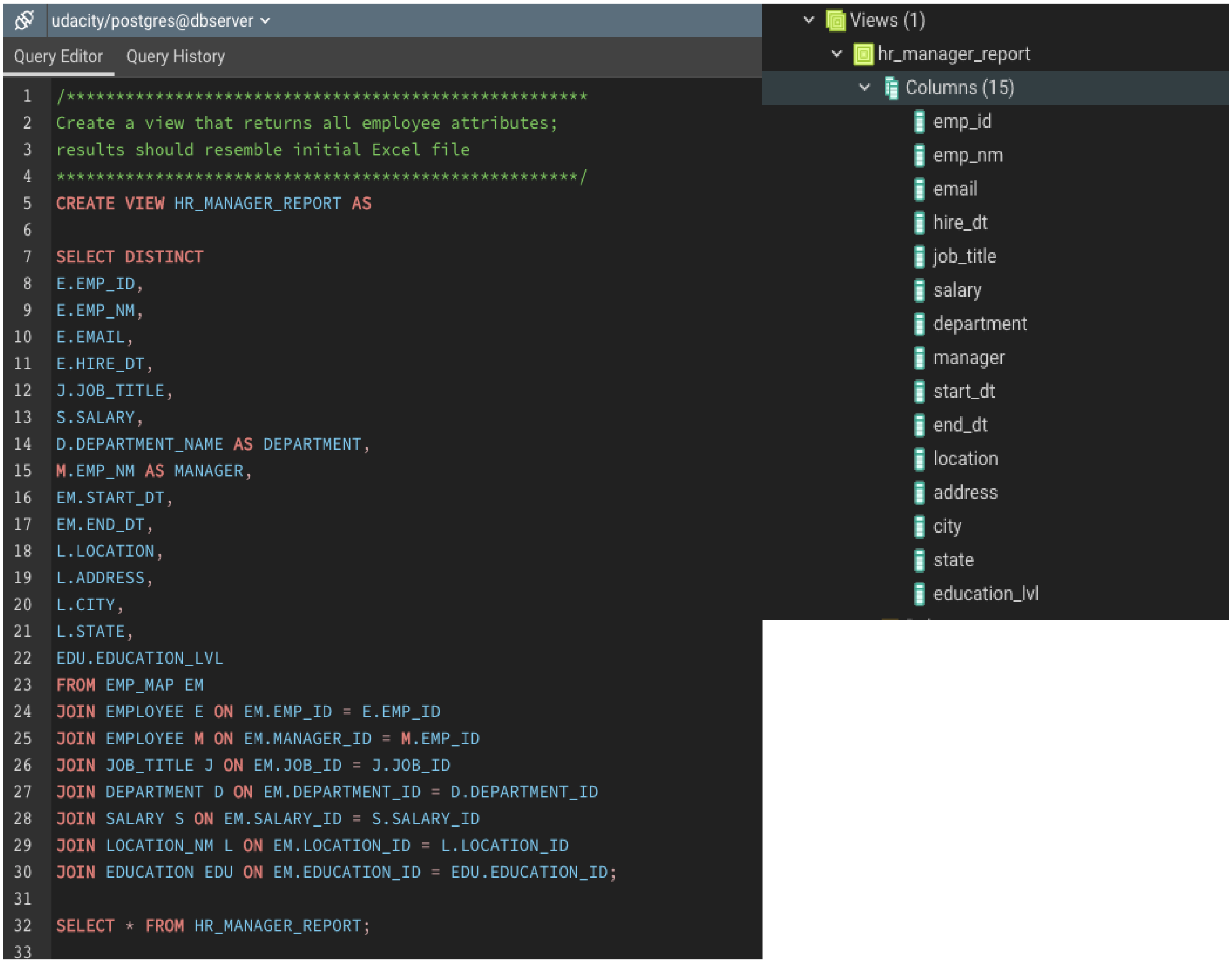
This last step is called Above and Beyond. In this step, I have proposed 3 challenges for you to complete, which are above and beyond the scope of the project. This is a chance to flex your coding muscles and show everyone how good you really are.

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


udacity/postgres@dbserver

Query Editor Query History

```
/*******************************************************
Create a view that returns all employee attributes;
results should resemble initial Excel file
*******************************************************/
CREATE VIEW HR_MANAGER_REPORT AS

SELECT DISTINCT
E.EMP_ID,
E.EMP_NM,
E.EMAIL,
E.HIRE_DT,
J.JOB_TITLE,
S.SALARY,
D.DEPARTMENT_NAME AS DEPARTMENT,
M.EMP_NM AS MANAGER,
EM.START_DT,
EM.END_DT,
L.LOCATION,
L.ADDRESS,
L.CITY,
L.STATE,
EDU.EDUCATION_LVL
FROM EMP_MAP EM
JOIN EMPLOYEE E ON EM.EMP_ID = E.EMP_ID
JOIN EMPLOYEE M ON EM.MANAGER_ID = M.EMP_ID
JOIN JOB_TITLE J ON EM.JOB_ID = J.JOB_ID
JOIN DEPARTMENT D ON EM.DEPARTMENT_ID = D.DEPARTMENT_ID
JOIN SALARY S ON EM.SALARY_ID = S.SALARY_ID
JOIN LOCATION_NM L ON EM.LOCATION_ID = L.LOCATION_ID
JOIN EDUCATION EDU ON EM.EDUCATION_ID = EDU.EDUCATION_ID;

SELECT * FROM HR_MANAGER_REPORT;
```

```
Views (1)
  hr_manager_report
    Columns (15)
      emp_id
      emp_nm
      email
      hire_dt
      job_title
      salary
      department
      manager
      start_dt
      end_dt
      location
      address
      city
      state
      education_lvl
```

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.

Standout Suggestion 3

Implement user security on the restricted salary attribute.

Appendix

Additional Info

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