Tech ABC Corp - HR Database

[Student Name & Date]

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:

Tech ABC Corp Partner has recently experienced subdenly growth. After has been hugely successfull with the IA powered video game console WORD. Their employees has grown suddenly from only 10 staff to 200 in only 6 months. Beside that from an office location at Dalas Texas has increased to 4 other locations nationwide. At Present, HR department is going in significant trouble to keep their bussiness up to firm expands by a shared Excell Sheet dataset.

To keep up to the current sistuation. They have the request to design and build up a HR database system to manage their employee information with the seamless integrity and security.

- Describe current data management solution:

The dataset currently storaged on a shared Excel Sheet that could be updated with any HR-employees.

- Describe current data available:

Excell dataset include 205 records with 15 columns that keep the employee information such as: Employee personal Information, Hire|Start|End date, working location and salary ...

- Additional data requests:-

1. Be able to connect ro payroll department's system in the future for intergration.

2. Maintain the employee attendance and paid time off information.

- Who will own/manage data

- HR employees.

- Who will have access to database

- All of Tech ABC Corp employess with domain could login with read only to access to database and restricted to the salary information HR and management level employees only.

- Only HR and management level employees should have write permission.

User	Database Access Right
General employees	Read only acess Restricted to salary infomation
HR and Managment level employees	Read and Write acess

Data Architect Business Requirement

- Estimated size of database
  - 205 rows and 15 columns.
- Estimated annual growth
  - 20% growth a year for the next 5 years.
- Is any of the data sensitive/restricted

  Resticting access to salary information with the employees that not HR or Managment employee level.

Data Architect Technical Requirement

- Justification for the new database
  - Maintain data integrity and data security
  - Be able to integate with payroll's system in the futures
  - Maintance employee attendance and paid time off information.

- Database objects (tables, views, special procedures)

No	Table
1	EMPLOYEES
2	EDUCATIONS
3	DEPARTMENTS
4	STATES
5	CITIES
6	LOCATIONS
7	JOBS
8	EMP_ASSIGNENTS

No	View
1	EMP_ASSIGNMENTS_W

- Data ingestion

  ETL would be suitable to ingest flat file to database.

Data Architect Technical Requirement

- Data governance (Ownership and User access)

  Ownership: HR

  User Access:

User	Database Access Right
General employees	Read only acess Restricted to salary infomation
HR and Managment level employees	Read and Write acess

- Scalability

- Once the compay have a fast growth. The increasing number of employee lead to be overload on reading when a huge of request to server. To keep the performance not degrade we should have replication deployment with read on slave and write on master.

- Flexibility

- The integraion to payroll's system could be solve by direct feed through dblink connetion.

- Storage & retention

  Storage (disk or in-memory): check IT best practices document
  - The current required was not involve to higher level computations so that databases are stored on disk will be approriately with acceptable performance and costing.

  Retention: how long does the data have to be kept for?
  - Following the federal regulations the data should be retain for at least 7 years.

- Backup

  IT Best Practices document lists Backup schedule requirements
  - Base on requirement, the HR data considered business critical so that must to apply critical Backup plan with schedule is full backup 1x per week, incremental backup daily.

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.

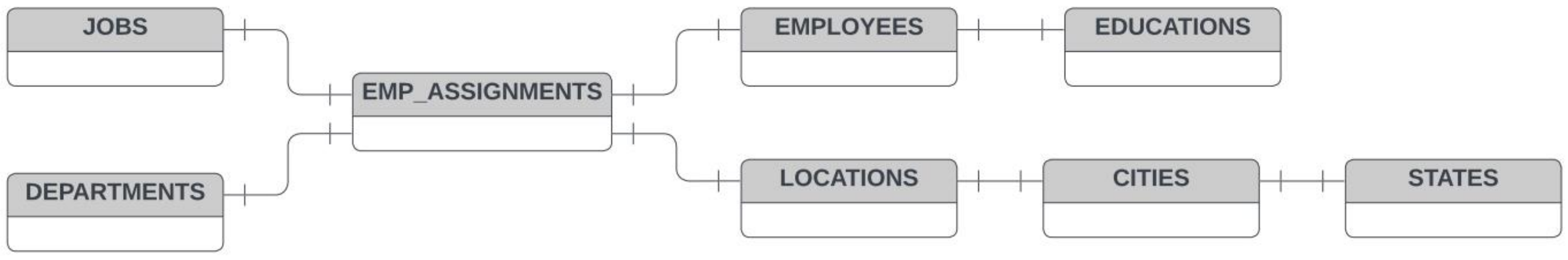
You will submit a screenshot for each of the 3 ERDs you create. You will find detailed instructions for developing each of the ERDs over the next several pages.

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.

HR database - Conceptual ERD

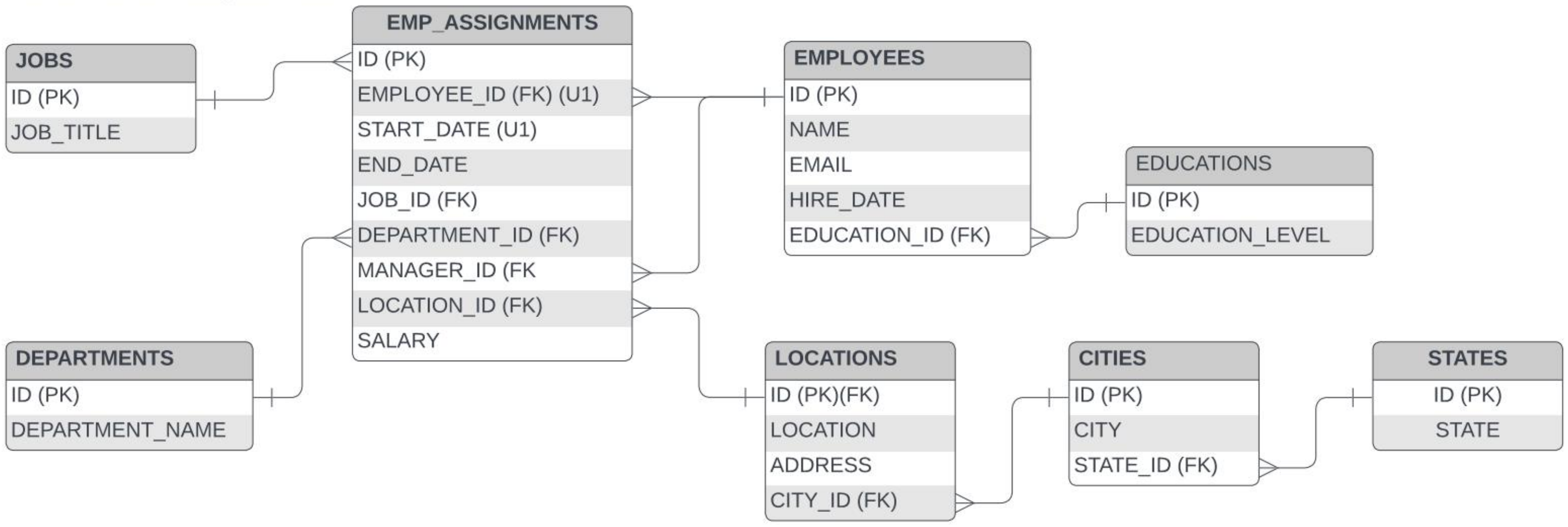


ERD

- 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.

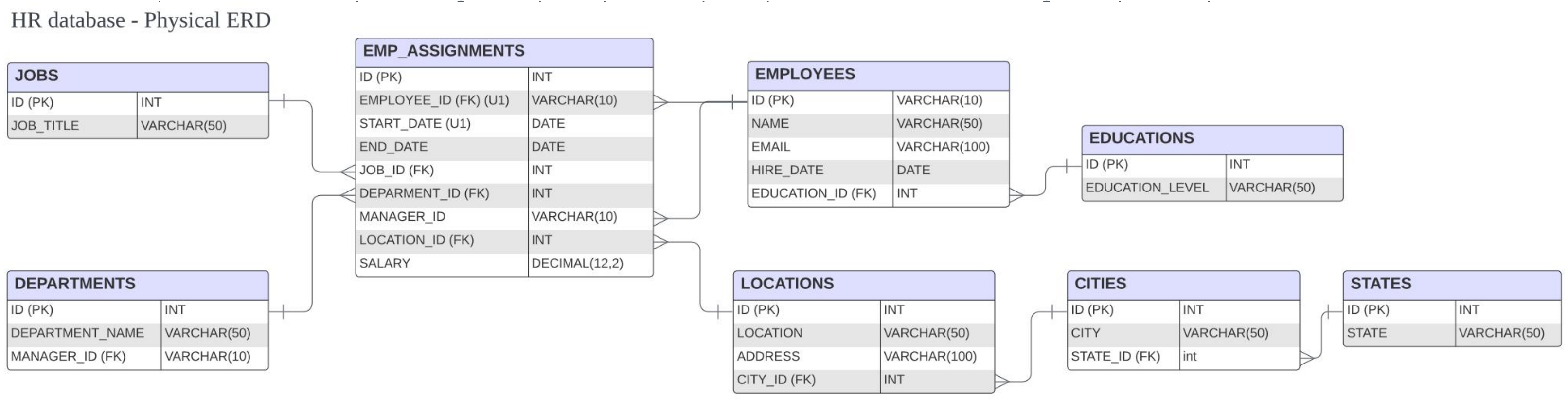
HR database - Logical ERD



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



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.

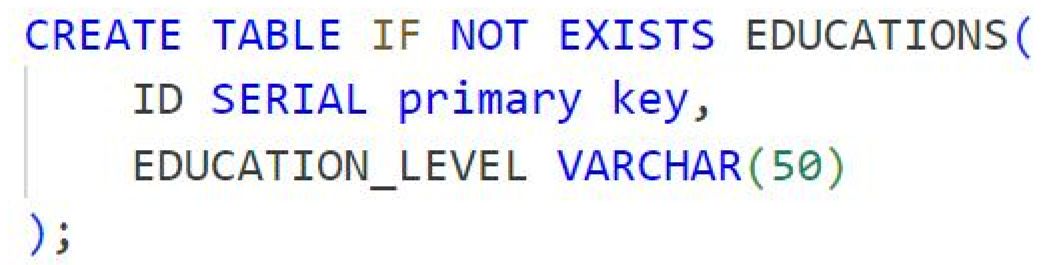
DDL

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.

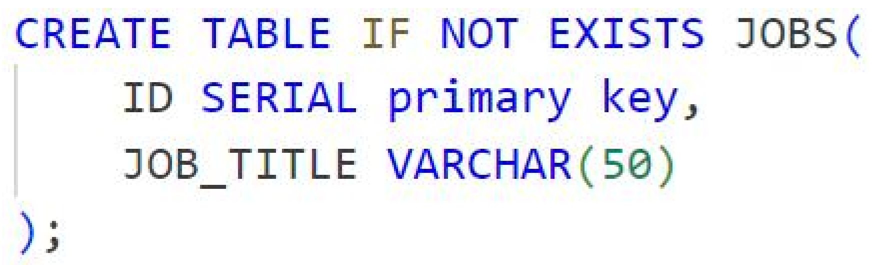
Foreign keys cannot be created on tables that do not exist yet, so it may be easier to create all tables in the database, then to go back and run modify statements on the tables to create foreign key constraints.

1. EDUCATIONS Table:



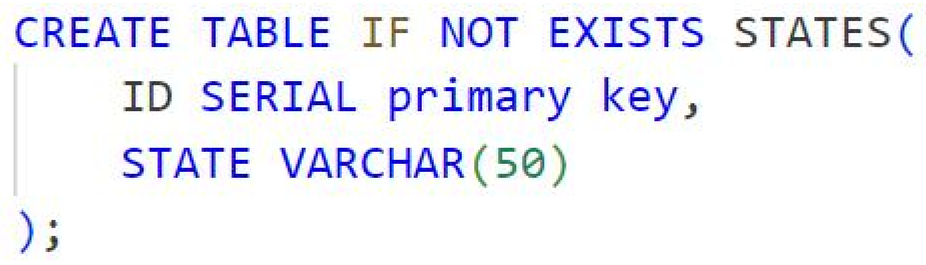
```sql
CREATE TABLE IF NOT EXISTS EDUCATIONS(
    ID SERIAL primary key,
    EDUCATION_LEVEL VARCHAR(50)
);
```

2. JOBS Table:



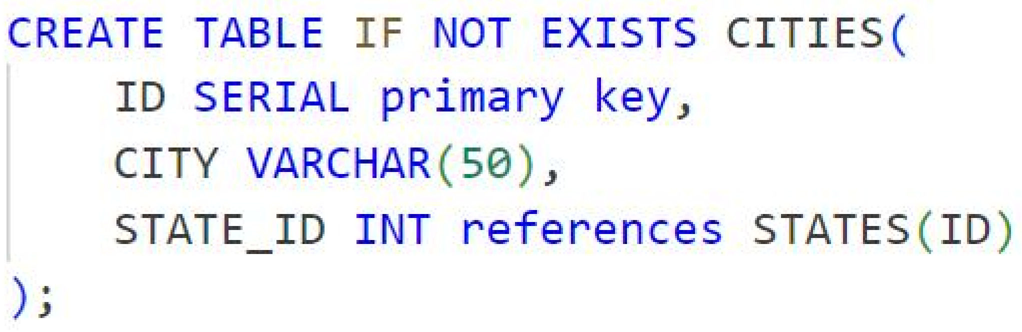
```sql
CREATE TABLE IF NOT EXISTS JOBS(
    ID SERIAL primary key,
    JOB_TITLE VARCHAR(50)
);
```

3. STATES Table:



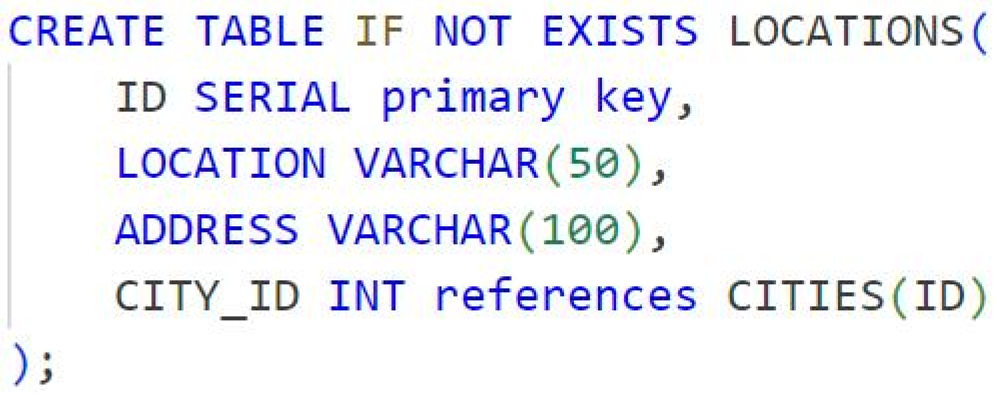
```sql
CREATE TABLE IF NOT EXISTS STATES(
    ID SERIAL primary key,
    STATE VARCHAR(50)
);
```

4. CITIES Table:



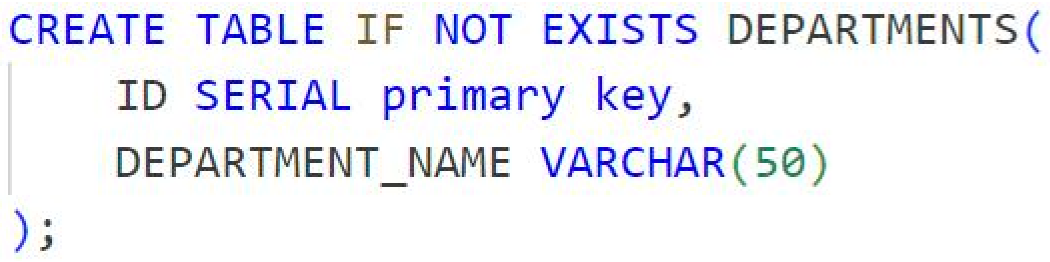
```sql
CREATE TABLE IF NOT EXISTS CITIES(
    ID SERIAL primary key,
    CITY VARCHAR(50),
    STATE_ID INT references STATES(ID)
);
```

5. LOCATIONS Table:



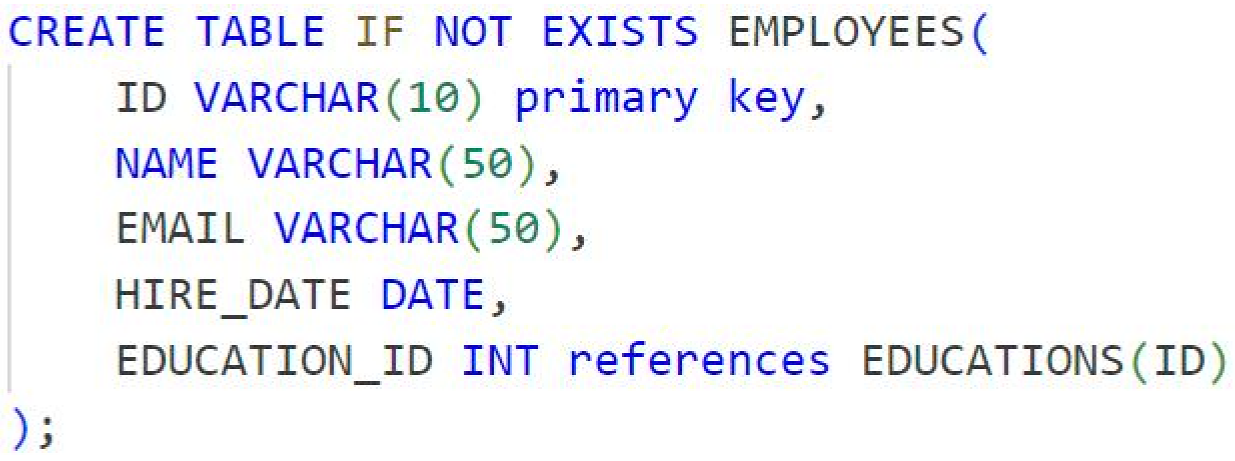
```sql
CREATE TABLE IF NOT EXISTS LOCATIONS(
    ID SERIAL primary key,
    LOCATION VARCHAR(50),
    ADDRESS VARCHAR(100),
    CITY_ID INT references CITIES(ID)
);
```

6. DEPARTMENTS Table:



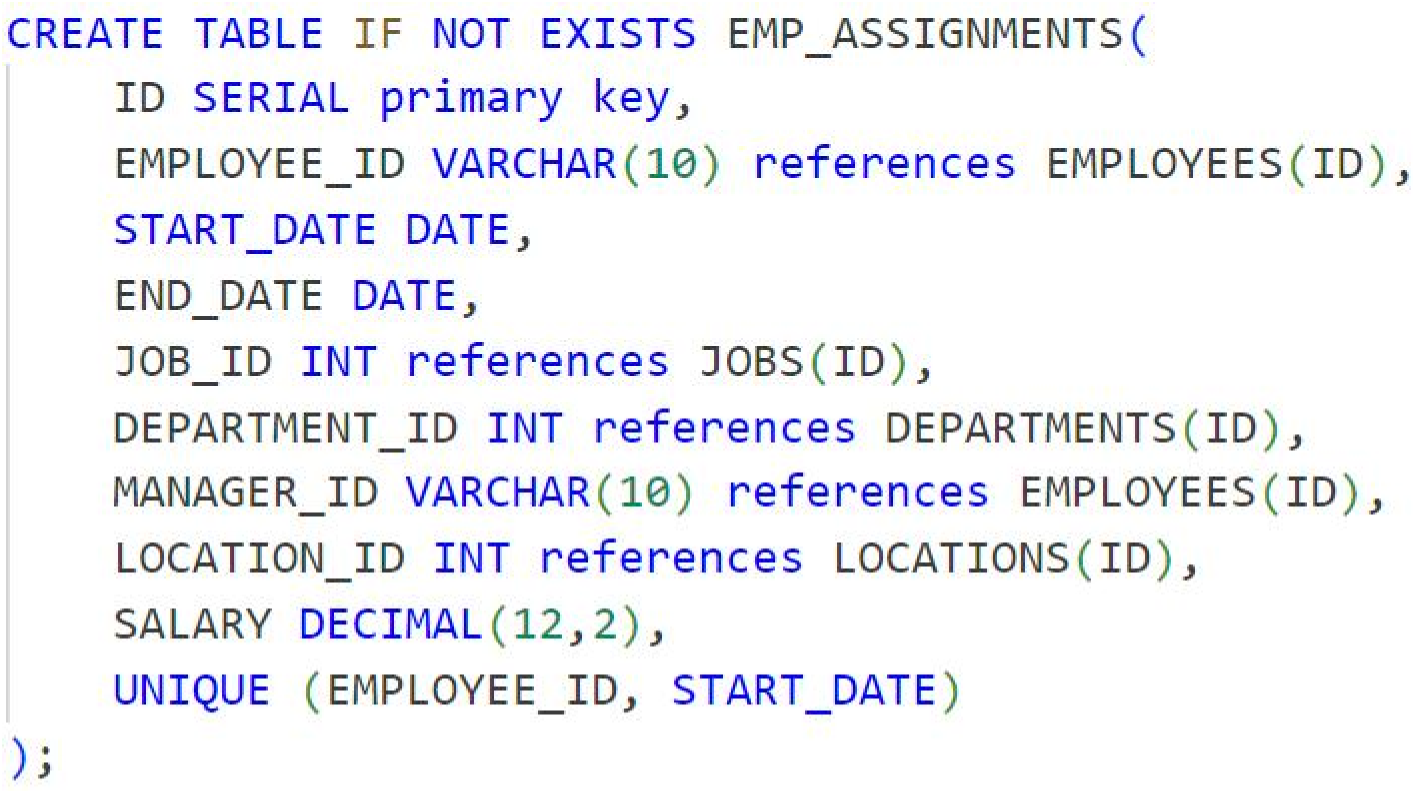
```sql
CREATE TABLE IF NOT EXISTS DEPARTMENTS(
    ID SERIAL primary key,
    DEPARTMENT_NAME VARCHAR(50)
);
```

7. EMPLOYEES Table:



```sql
CREATE TABLE IF NOT EXISTS EMPLOYEES(
    ID VARCHAR(10) primary key,
    NAME VARCHAR(50),
    EMAIL VARCHAR(50),
    HIRE_DATE DATE,
    EDUCATION_ID INT references EDUCATIONS(ID)
);
```

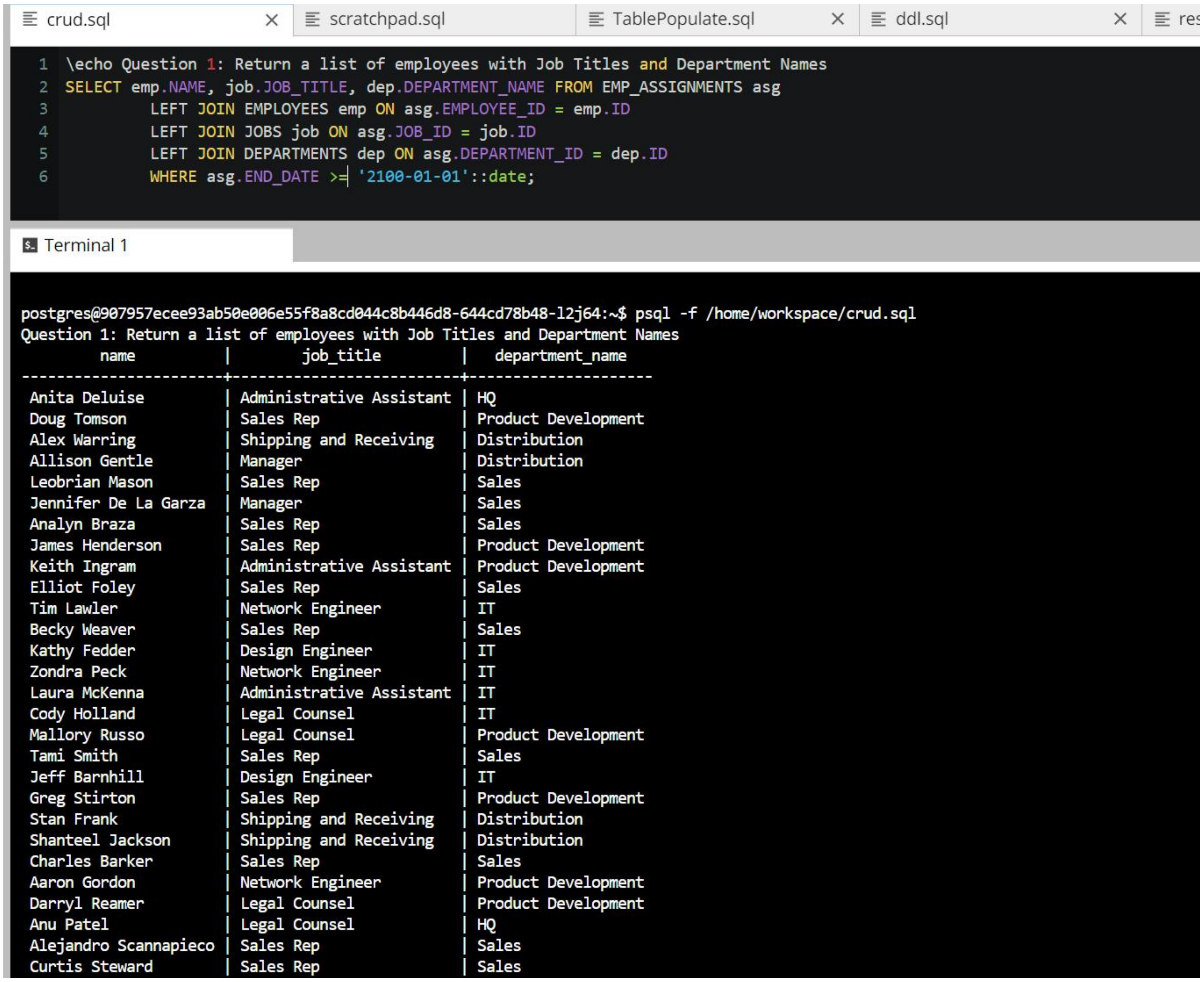
8. EMP_ASSIGNMENTS



```sql
CREATE TABLE IF NOT EXISTS EMP_ASSIGNMENTS(
    ID SERIAL primary key,
    EMPLOYEE_ID VARCHAR(10) references EMPLOYEES(ID),
    START_DATE DATE,
    END_DATE DATE,
    JOB_ID INT references JOBS(ID),
    DEPARTMENT_ID INT references DEPARTMENTS(ID),
    MANAGER_ID VARCHAR(10) references EMPLOYEES(ID),
    LOCATION_ID INT references LOCATIONS(ID),
    SALARY DECIMAL(12,2),
    UNIQUE (EMPLOYEE_ID, START_DATE)
);
```

CRUD

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



crud.sql | scratchpad.sql | TablePopulate.sql | ddl.sql | res

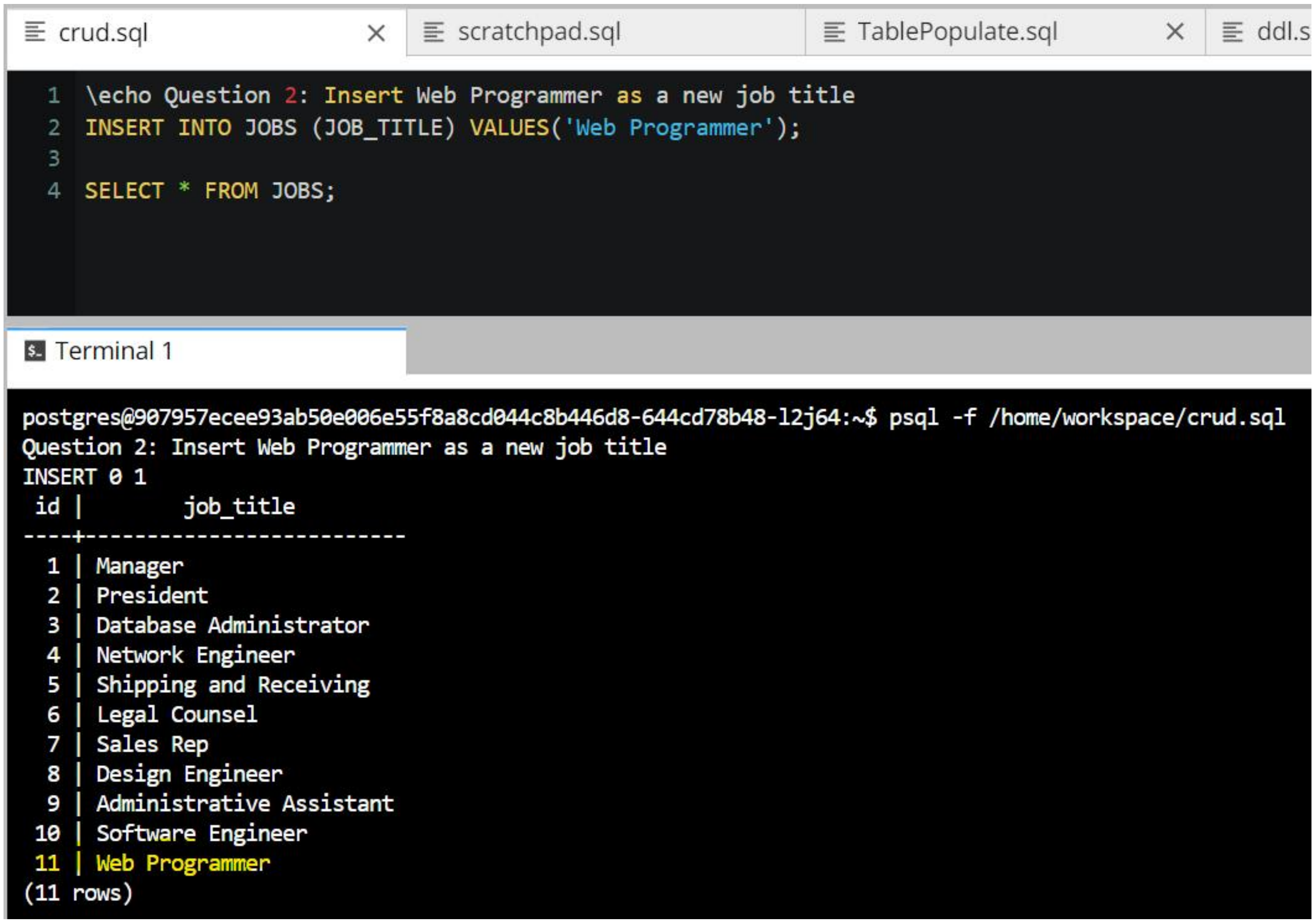
```sql
\echo Question 1: Return a list of employees with Job Titles and Department Names
SELECT emp.NAME, job.JOB_TITLE, dep.DEPARTMENT_NAME FROM EMP_ASSIGNMENTS asg
         LEFT JOIN EMPLOYEES emp ON asg.EMPLOYEE_ID = emp.ID
         LEFT JOIN JOBS job ON asg.JOB_ID = job.ID
         LEFT JOIN DEPARTMENTS dep ON asg.DEPARTMENT_ID = dep.ID
         WHERE asg.END_DATE >= '2100-01-01'::date;
```

Terminal 1

```
postgres@907957ecee93ab50e006e55f8a8cd044c8b446d8-644cd78b48-l2j64:~$ psql -f /home/workspace/crud.sql
Question 1: Return a list of employees with Job Titles and Department Names
         name          |         job_title          |   department_name
-----------------------+----------------------------+---------------------
 Anita Deluise         | Administrative Assistant | HQ
 Doug Tomson           | Sales Rep                | Product Development
 Alex Warring          | Shipping and Receiving   | Distribution
 Allison Gentle        | Manager                  | Distribution
 Leobrian Mason        | Sales Rep                | Sales
 Jennifer De La Garza  | Manager                  | Sales
 Analyn Braza          | Sales Rep                | Sales
 James Henderson       | Sales Rep                | Product Development
 Keith Ingram          | Administrative Assistant | Product Development
 Elliot Foley          | Sales Rep                | Sales
 Tim Lawler            | Network Engineer         | IT
 Becky Weaver          | Sales Rep                | Sales
 Kathy Fedder          | Design Engineer          | IT
 Zondra Peck           | Network Engineer         | IT
 Laura McKenna         | Administrative Assistant | IT
 Cody Holland          | Legal Counsel            | IT
 Mallory Russo         | Legal Counsel            | Product Development
 Tami Smith            | Sales Rep                | Sales
 Jeff Barnhill         | Design Engineer          | IT
 Greg Stirton          | Sales Rep                | Product Development
 Stan Frank            | Shipping and Receiving   | Distribution
 Shanteel Jackson      | Shipping and Receiving   | Distribution
 Charles Barker        | Sales Rep                | Sales
 Aaron Gordon          | Network Engineer         | Product Development
 Darryl Reamer         | Legal Counsel            | Product Development
 Anu Patel             | Legal Counsel            | HQ
 Alejandro Scannapieco | Sales Rep                | Sales
 Curtis Steward        | Sales Rep                | Sales
```

CRUD

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

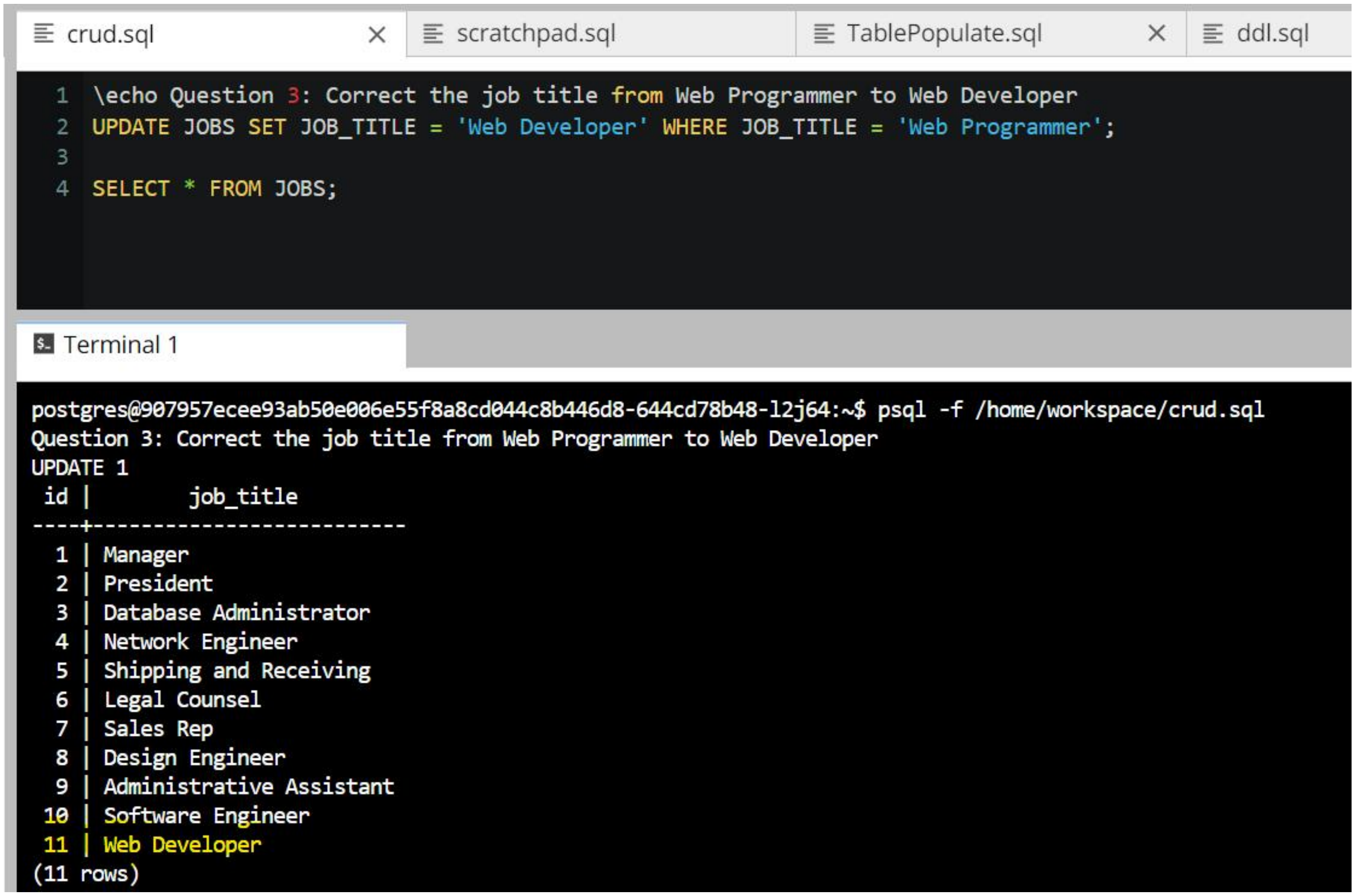


```
\echo Question 2: Insert Web Programmer as a new job title
INSERT INTO JOBS (JOB_TITLE) VALUES('Web Programmer');

SELECT * FROM JOBS;
```

```
postgres@907957ecee93ab50e006e55f8a8cd044c8b446d8-644cd78b48-l2j64:~$ psql -f /home/workspace/crud.sql
Question 2: Insert Web Programmer as a new job title
INSERT 0 1
 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



crud.sql | scratchpad.sql | TablePopulate.sql | ddl.sql

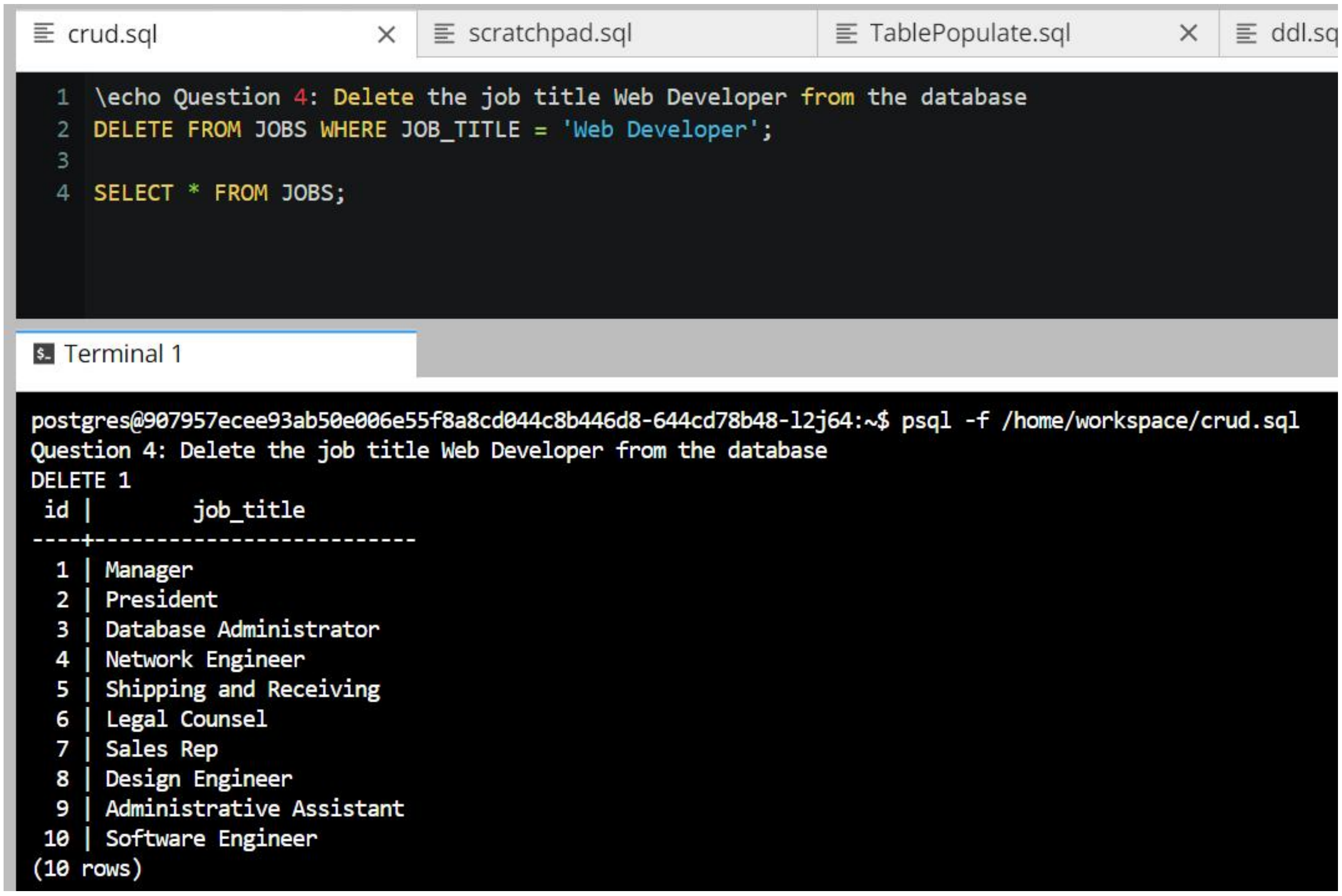
```sql
\echo Question 3: Correct the job title from Web Programmer to Web Developer
UPDATE JOBS SET JOB_TITLE = 'Web Developer' WHERE JOB_TITLE = 'Web Programmer';

SELECT * FROM JOBS;
```

Terminal 1

```
postgres@907957ecee93ab50e006e55f8a8cd044c8b446d8-644cd78b48-l2j64:~$ psql -f /home/workspace/crud.sql
Question 3: Correct the job title from Web Programmer to Web Developer
UPDATE 1
 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

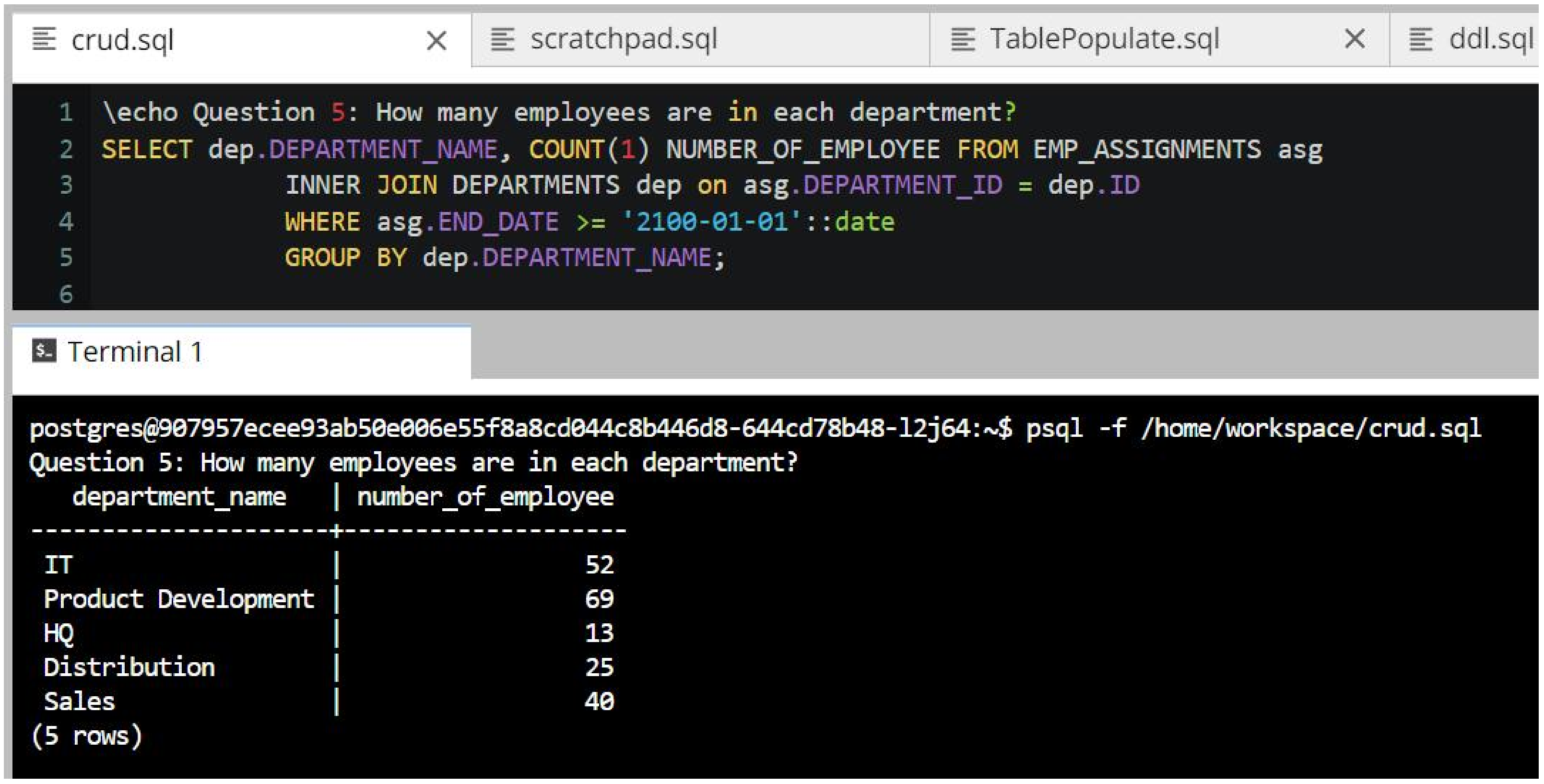


```sql
\echo Question 4: Delete the job title Web Developer from the database
DELETE FROM JOBS WHERE JOB_TITLE = 'Web Developer';

SELECT * FROM JOBS;
```

```
postgres@907957ecee93ab50e006e55f8a8cd044c8b446d8-644cd78b48-l2j64:~$ psql -f /home/workspace/crud.sql
Question 4: Delete the job title Web Developer from the database
DELETE 1
 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?


crud.sql | scratchpad.sql | TablePopulate.sql | ddl.sql

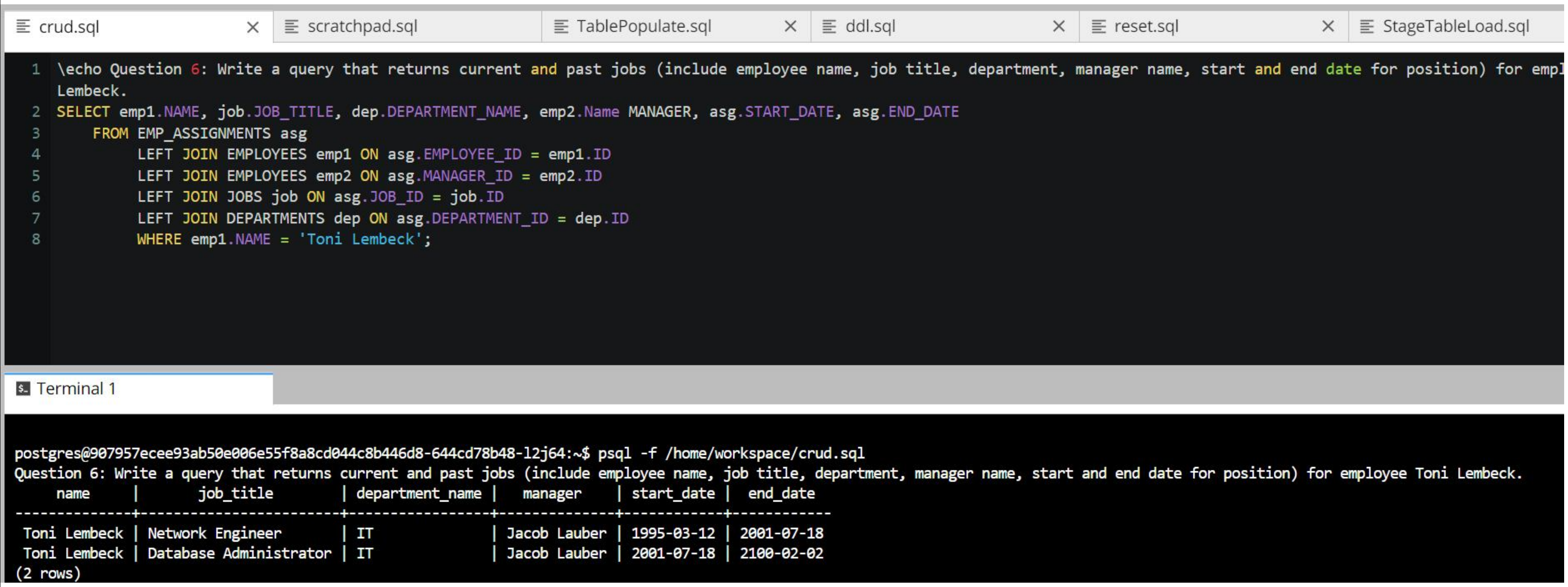
```sql
\echo Question 5: How many employees are in each department?
SELECT dep.DEPARTMENT_NAME, COUNT(1) NUMBER_OF_EMPLOYEE FROM EMP_ASSIGNMENTS asg
            INNER JOIN DEPARTMENTS dep on asg.DEPARTMENT_ID = dep.ID
            WHERE asg.END_DATE >= '2100-01-01'::date
            GROUP BY dep.DEPARTMENT_NAME;

```

Terminal 1

```
postgres@907957ecee93ab50e006e55f8a8cd044c8b446d8-644cd78b48-12j64:~$ psql -f /home/workspace/crud.sql
Question 5: How many employees are in each department?
   department_name    | number_of_employee
---------------------+--------------------
 IT                  |                 52
 Product Development |                 69
 HQ                  |                 13
 Distribution        |                 25
 Sales               |                 40
(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.



```sql
\echo 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 empl
Lembeck.
SELECT emp1.NAME, job.JOB_TITLE, dep.DEPARTMENT_NAME, emp2.Name MANAGER, asg.START_DATE, asg.END_DATE
    FROM EMP_ASSIGNMENTS asg
         LEFT JOIN EMPLOYEES emp1 ON asg.EMPLOYEE_ID = emp1.ID
         LEFT JOIN EMPLOYEES emp2 ON asg.MANAGER_ID = emp2.ID
         LEFT JOIN JOBS job ON asg.JOB_ID = job.ID
         LEFT JOIN DEPARTMENTS dep ON asg.DEPARTMENT_ID = dep.ID
         WHERE emp1.NAME = 'Toni Lembeck';
```

```
postgres@907957ecee93ab50e006e55f8a8cd044c8b446d8-644cd78b48-l2j64:~$ psql -f /home/workspace/crud.sql
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.
     name     |       job_title        | department_name |   manager    | start_date |  end_date
--------------+------------------------+-----------------+--------------+------------+------------
 Toni Lembeck | Network Engineer       | IT              | Jacob Lauber | 1995-03-12 | 2001-07-18
 Toni Lembeck | Database Administrator | IT              | Jacob Lauber | 2001-07-18 | 2100-02-02
(2 rows)
```

CRUD

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

Each employee in the company has a domain authenticated username that they will use to access any database they have been authorized username access to. We have two step.

1. Grant user access to all tables in the database, then revoke access to EMP_ASSIGMENTS that contain salary information.

2. Create the specific Views that not contain Salary information to restricted employees.

Step 4

Above and Beyond

(optional)

Step 4: Above and Beyond

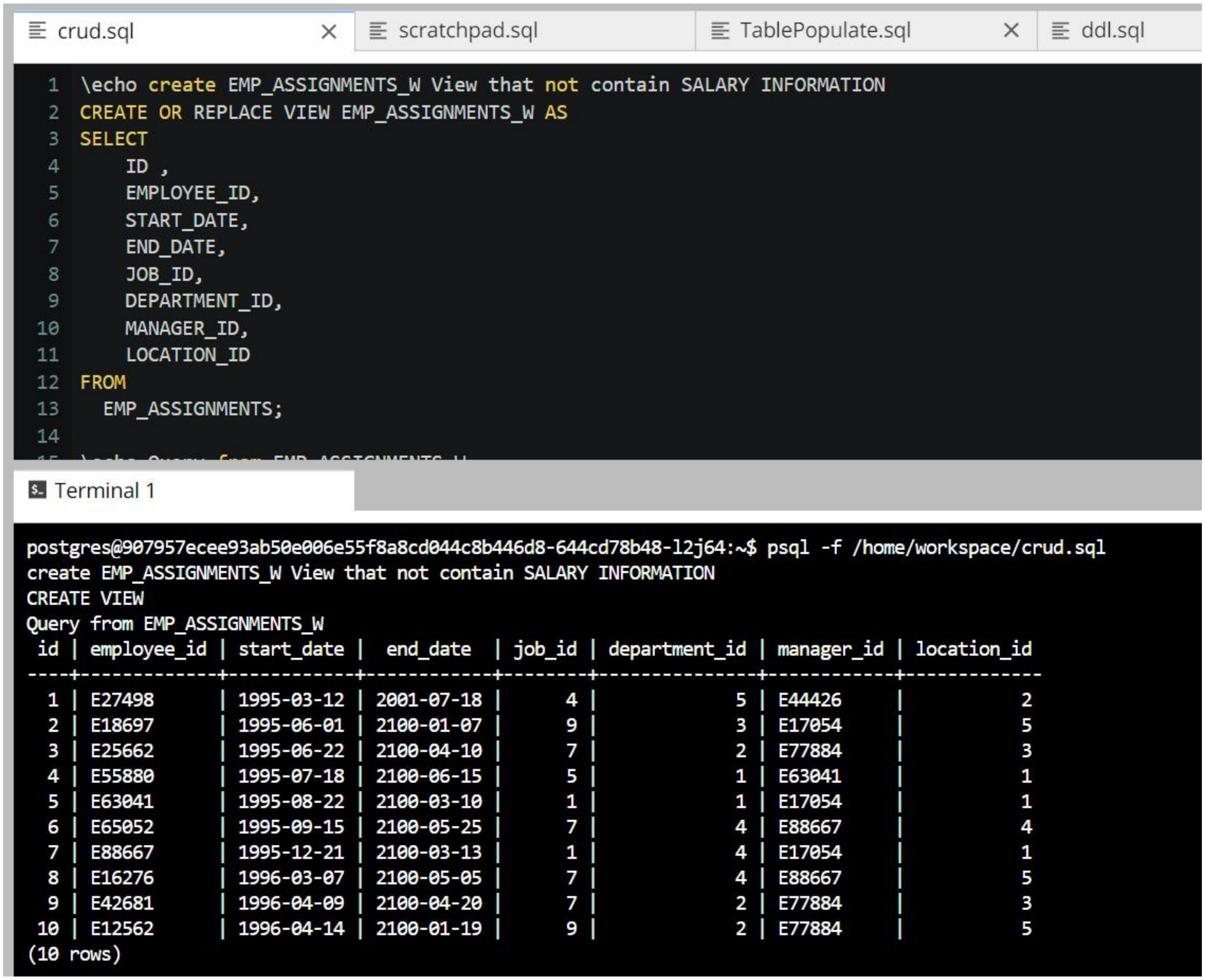
This last step is called Above and Beyond. In this step, I have proposed 3 challenges for you to complete, which are above and beyond the scope of the project. This is a chance to flex your coding muscles and show everyone how good you really are.

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

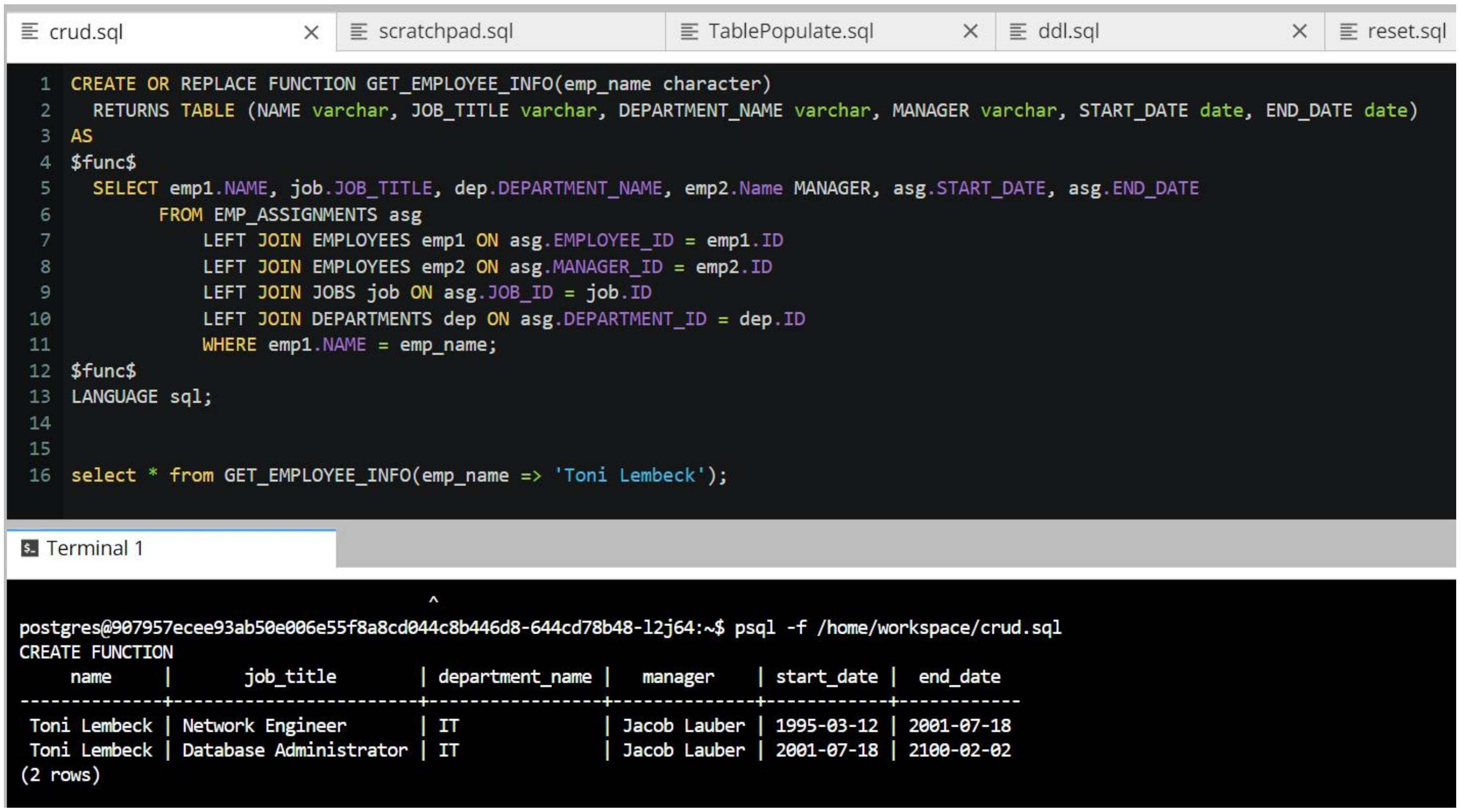


```
\echo create EMP_ASSIGNMENTS_W View that not contain SALARY INFORMATION
CREATE OR REPLACE VIEW EMP_ASSIGNMENTS_W AS
SELECT
    ID ,
    EMPLOYEE_ID,
    START_DATE,
    END_DATE,
    JOB_ID,
    DEPARTMENT_ID,
    MANAGER_ID,
    LOCATION_ID
FROM
  EMP_ASSIGNMENTS;

```

```
postgres@907957ecee93ab50e006e55f8a8cd044c8b446d8-644cd78b48-l2j64:~$ psql -f /home/workspace/crud.sql
create EMP_ASSIGNMENTS_W View that not contain SALARY INFORMATION
CREATE VIEW
Query from EMP_ASSIGNMENTS_W
 id | employee_id | start_date |  end_date  | job_id | department_id | manager_id | location_id
----+-------------+------------+------------+--------+---------------+------------+-------------
  1 | E27498      | 1995-03-12 | 2001-07-18 |      4 |             5 | E44426     |           2
  2 | E18697      | 1995-06-01 | 2100-01-07 |      9 |             3 | E17054     |           5
  3 | E25662      | 1995-06-22 | 2100-04-10 |      7 |             2 | E77884     |           3
  4 | E55880      | 1995-07-18 | 2100-06-15 |      5 |             1 | E63041     |           1
  5 | E63041      | 1995-08-22 | 2100-03-10 |      1 |             1 | E17054     |           1
  6 | E65052      | 1995-09-15 | 2100-05-25 |      7 |             4 | E88667     |           4
  7 | E88667      | 1995-12-21 | 2100-03-13 |      1 |             4 | E17054     |           1
  8 | E16276      | 1996-03-07 | 2100-05-05 |      7 |             4 | E88667     |           5
  9 | E42681      | 1996-04-09 | 2100-04-20 |      7 |             2 | E77884     |           3
 10 | E12562      | 1996-04-14 | 2100-01-19 |      9 |             2 | E77884     |           5
(10 rows)
```

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.

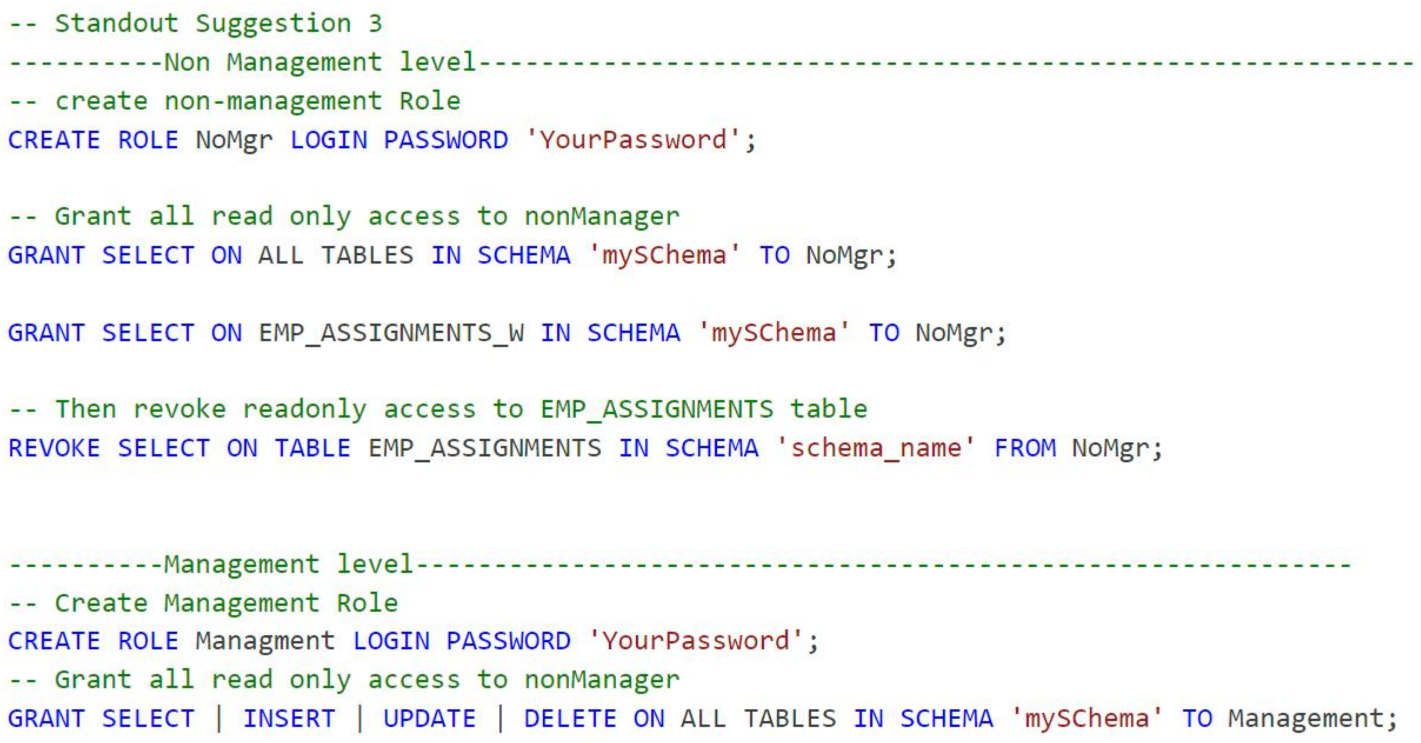


```sql
CREATE OR REPLACE FUNCTION GET_EMPLOYEE_INFO(emp_name character)
  RETURNS TABLE (NAME varchar, JOB_TITLE varchar, DEPARTMENT_NAME varchar, MANAGER varchar, START_DATE date, END_DATE date)
AS
$func$
  SELECT emp1.NAME, job.JOB_TITLE, dep.DEPARTMENT_NAME, emp2.Name MANAGER, asg.START_DATE, asg.END_DATE
        FROM EMP_ASSIGNMENTS asg
            LEFT JOIN EMPLOYEES emp1 ON asg.EMPLOYEE_ID = emp1.ID
            LEFT JOIN EMPLOYEES emp2 ON asg.MANAGER_ID = emp2.ID
            LEFT JOIN JOBS job ON asg.JOB_ID = job.ID
            LEFT JOIN DEPARTMENTS dep ON asg.DEPARTMENT_ID = dep.ID
            WHERE emp1.NAME = emp_name;
$func$
LANGUAGE sql;


select * from GET_EMPLOYEE_INFO(emp_name => 'Toni Lembeck');
```

```
                                       ^
postgres@907957ecee93ab50e006e55f8a8cd044c8b446d8-644cd78b48-l2j64:~$ psql -f /home/workspace/crud.sql
CREATE FUNCTION
     name     |       job_title        | department_name |   manager    | start_date |  end_date
--------------+------------------------+-----------------+--------------+------------+------------
 Toni Lembeck | Network Engineer       | IT              | Jacob Lauber | 1995-03-12 | 2001-07-18
 Toni Lembeck | Database Administrator | IT              | Jacob Lauber | 2001-07-18 | 2100-02-02
(2 rows)
```

Standout Suggestion 3

Implement user security on the restricted salary attribute.



```sql
-- Standout Suggestion 3
----------Non Management level-------------------------------------------------------------
-- create non-management Role
CREATE ROLE NoMgr LOGIN PASSWORD 'YourPassword';

-- Grant all read only access to nonManager
GRANT SELECT ON ALL TABLES IN SCHEMA 'mySChema' TO NoMgr;

GRANT SELECT ON EMP_ASSIGNMENTS_W IN SCHEMA 'mySChema' TO NoMgr;

-- Then revoke readonly access to EMP_ASSIGNMENTS table
REVOKE SELECT ON TABLE EMP_ASSIGNMENTS IN SCHEMA 'schema_name' FROM NoMgr;


----------Management level-------------------------------------------------------------
-- Create Management Role
CREATE ROLE Managment LOGIN PASSWORD 'YourPassword';
-- Grant all read only access to nonManager
GRANT SELECT | INSERT | UPDATE | DELETE ON ALL TABLES IN SCHEMA 'mySChema' TO Management;
```

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.