Tech ABC Corp - HR Database

[Student Name & Date]

How to use this Template

- Make a copy of this Google Slide deck.
- We have provided these slides as a guide to ensure that you submit all the required components to successfully complete your project.
- When presenting your project, please only think of this as a guide. We encourage you to use creative freedom when making changes, as long as the required information is present.
- **Remember to delete this and all** of the other example slides before you submit your project.
- **Remember to add your name and the date** to the cover slide

Reference slide remove before you submit

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 requested database would be a scalable HR system to store the company's employee data. This would replace the existing Excel spreadsheet currently in use, which is prone to human error, and insecure.

- **Describe current data management solution:**

The current storage method is an Excel spreadsheet.

- **Describe current data available:**

The shared spreadsheet contains existing data

- **Additional data requests:**

The expect the business to grow 20% over the next 5 years and requires a direct data feed to Payroll in future

- **Who will own/manage data**

HR department of the business

- **Who will have access to database**

Employee - They must have read-only privilege and <u>not</u> be able to read Salary information

HR and Management - Read and Write privilege. Are able to access salary information

Data Architect Business Requirement

- **Estimated size of database**

  200 rows currently, growing by 20% per year for 5 years covering aproximately 14 columns.

Estimated annual growth

The expected growth of the data is 20% per year for 5 years

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

  Yes. Salary information is sensitive and should be restricted to non-Manager and non-HR employees

Data Architect Technical Requirement

- **Justification for the new database**

  1. the data would be more secure, current implementation allows anyone access the Salary information of all employees.
  2. Integrity can be ensured, current implementation in Excel is prone to human error as it is manually maintained leading to duplication, inconsistent information.
  3. A data base would be scalable, allowing for the anticipated growth and retention period require as well as allowing every user access

- **Database objects**

  Tables: STAGING_HR_RAW, EMPLOYMENT_HISTORY, JOB, LOCATION, DEPARTMENT, SALARY, ADDRESS, EMPLOYEE, EDUCATION,

  VIEW: excel_extract

  PROCEDURE: f_RETURN_EMPLOYEE_HIST

- **Data ingestion**

  Since the data is currently stored on flat file, ETL is the preferred method if data ingestion stipulated by Tech ABC Groups IT Best Practice.

Data Architect Technical Requirement

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

  Ownership: HR department

  User Access:

Employee - accounting for approx 90% of users. Restricted to non-sensitive information that includes salary

HR and Management- Read and write access, including access sensitive information

- **Scalability** Keeping in mind volumes anticipated and usage anticipated , sharding would not be required

- **Flexibility** Payroll data required in future would use direct data feed so this must be accounted for via user management and privileges considered

- **Storage & retention**

  Storage (disk or in-memory): Per the best practices, disk storage is sufficient as no higher level computational tasks are required.
  Retention: 7years due to federal regulations

- **Backup**

  Since the data is considered critical, a schedule of full back 1x per week, 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.

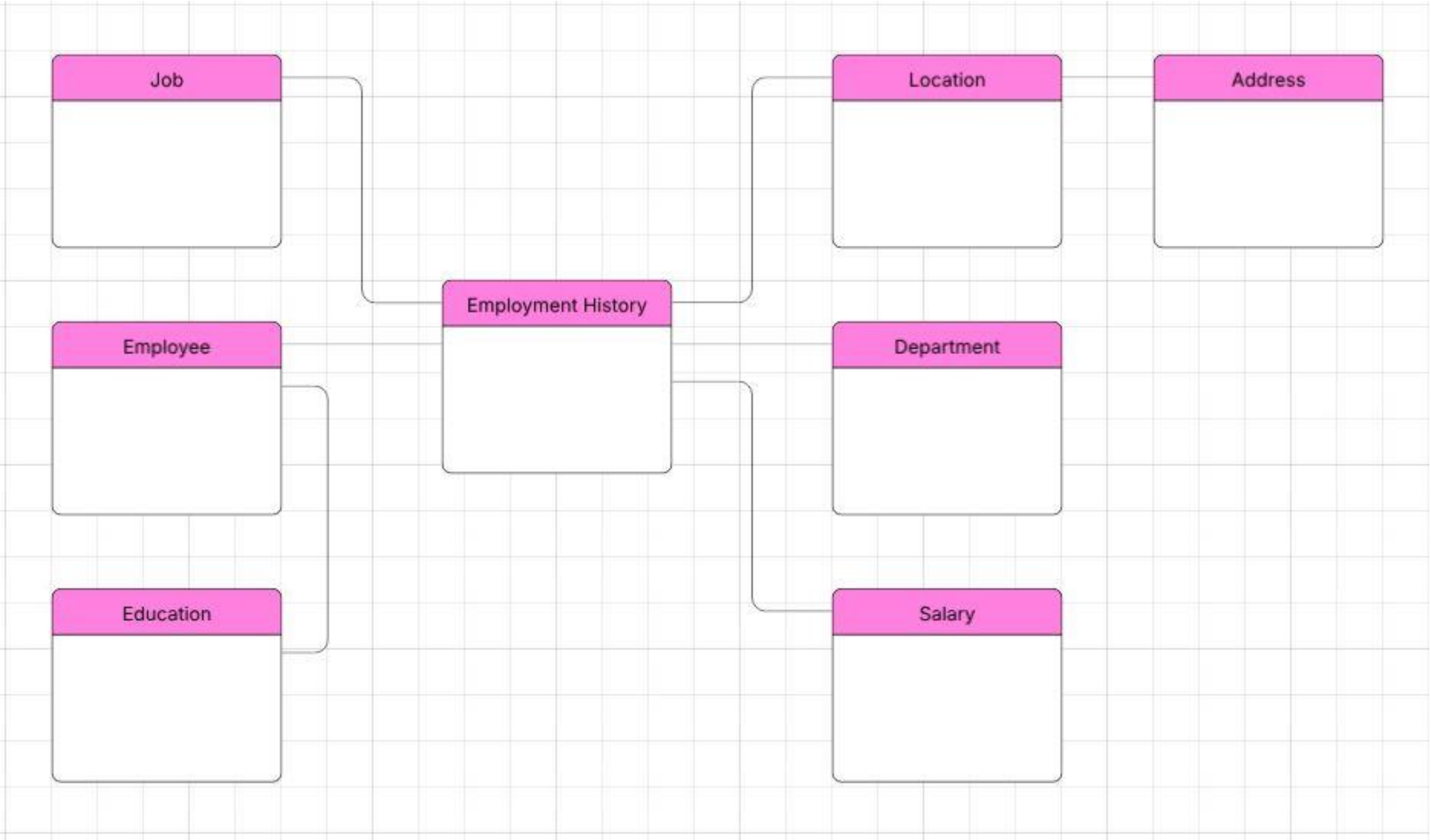
You will submit a screenshot for each of the 3 ERDs you create. You will find detailed instructions for developing each of the ERDs over the next several pages.

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.

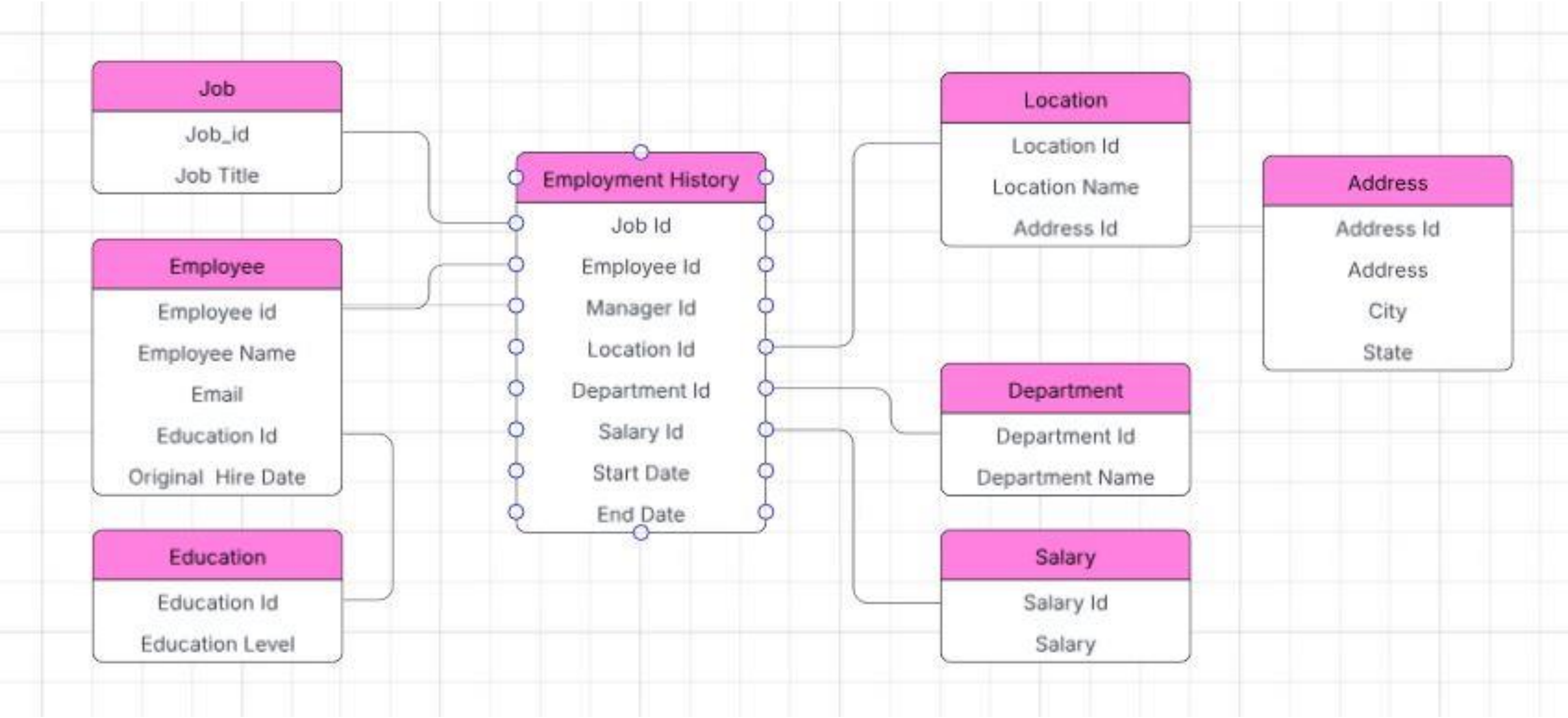


ERD

- ## 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.

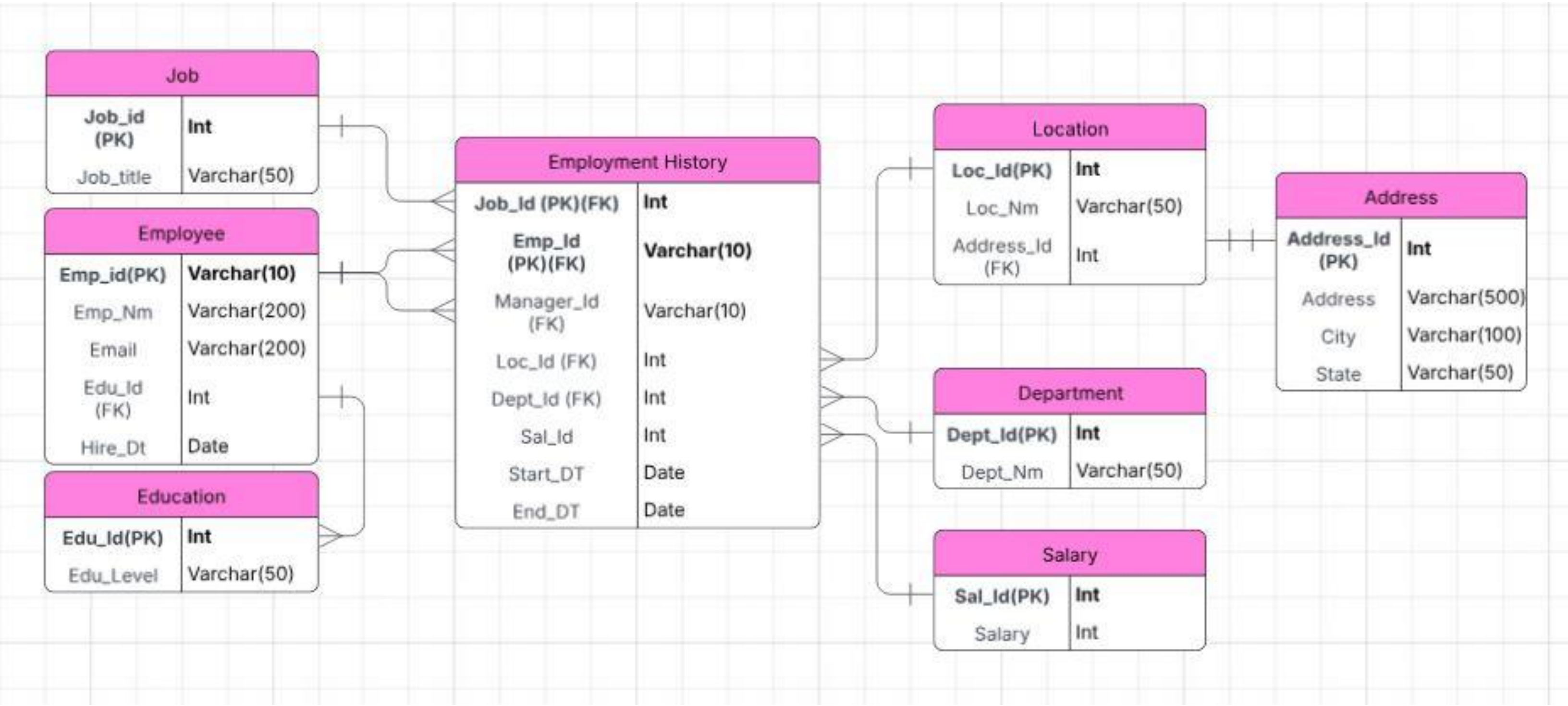
Use Lucidchart's built-in template for DBMS ER Diagram UML.



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.



Job	
Job_id (PK)	**Int**
Job_title	Varchar(50)

Employee	
Emp_id(PK)	**Varchar(10)**
Emp_Nm	Varchar(200)
Email	Varchar(200)
Edu_Id (FK)	Int
Hire_Dt	Date

Education	
Edu_Id(PK)	**Int**
Edu_Level	Varchar(50)

Employment History	
Job_Id (PK)(FK)	**Int**
Emp_Id (PK)(FK)	**Varchar(10)**
Manager_Id (FK)	Varchar(10)
Loc_Id (FK)	Int
Dept_Id (FK)	Int
Sal_Id	Int
Start_DT	Date
End_DT	Date

Location	
Loc_Id(PK)	**Int**
Loc_Nm	Varchar(50)
Address_Id (FK)	Int

Address	
Address_Id (PK)	**Int**
Address	Varchar(500)
City	Varchar(100)
State	Varchar(50)

Department	
Dept_Id(PK)	**Int**
Dept_Nm	Varchar(50)

Salary	
Sal_Id(PK)	**Int**
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.

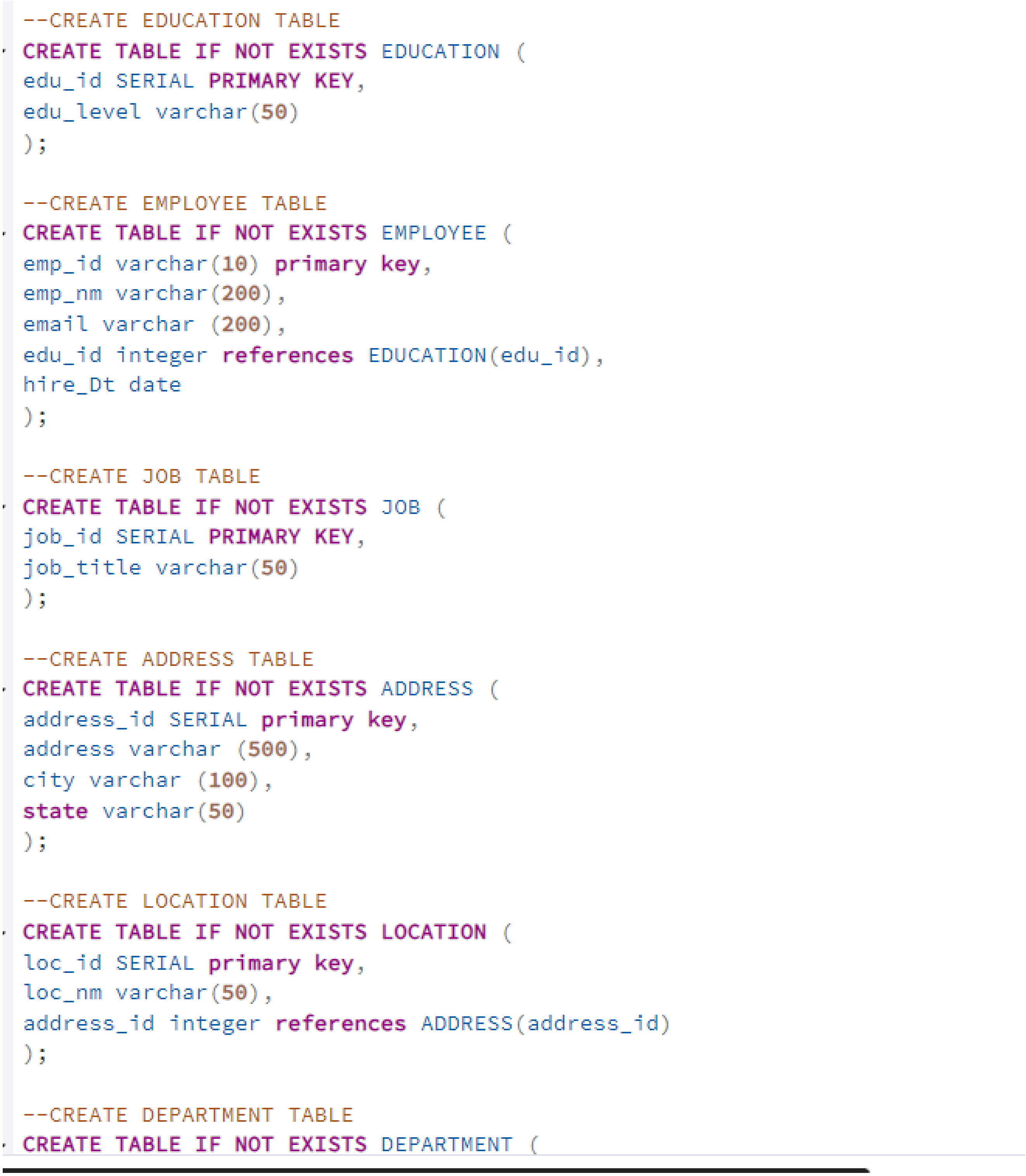
Hints
Your DDL script will be graded by running the code you submit. Please ensure your SQL code runs properly!

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

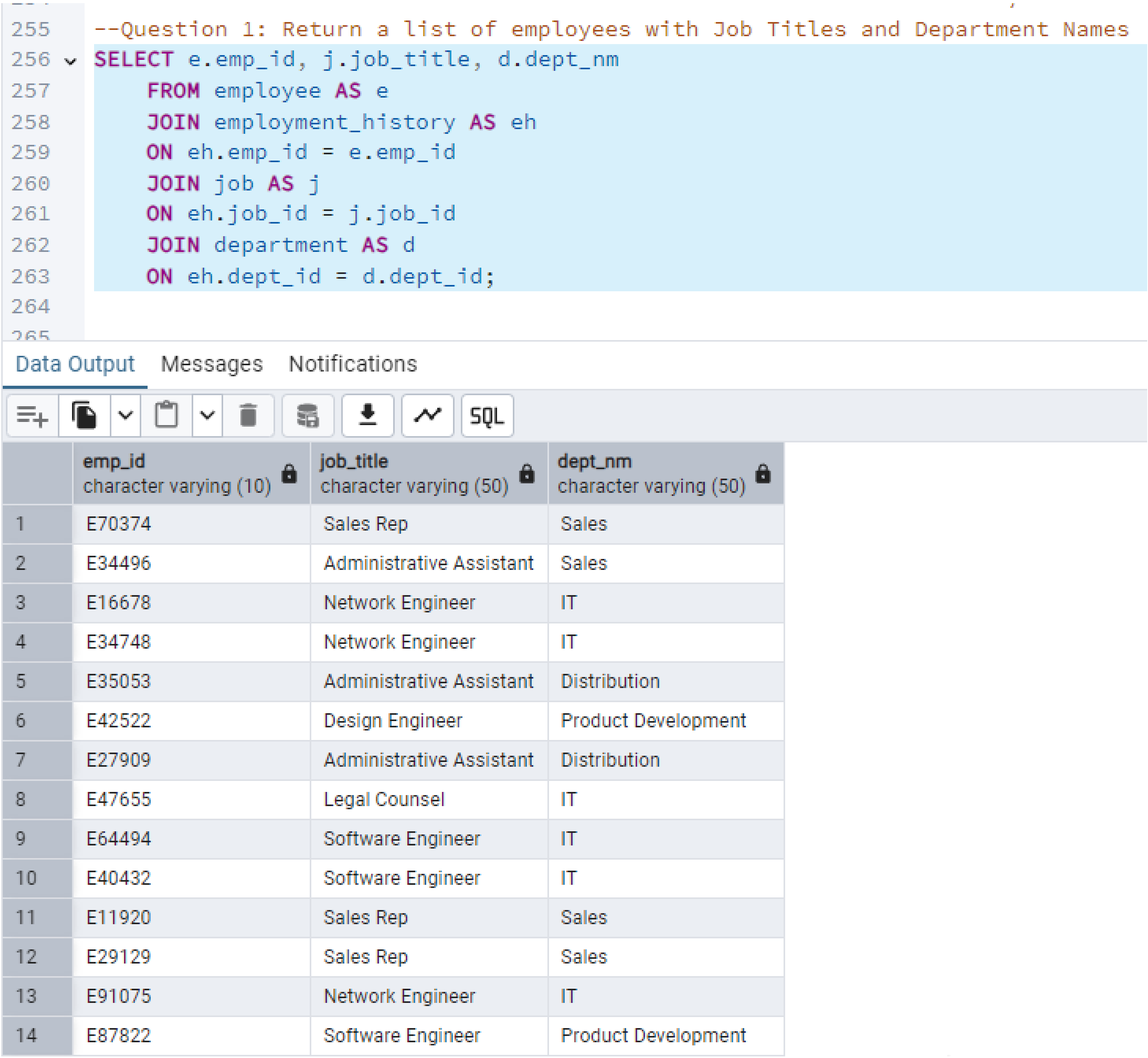
Create a DDL SQL script capable of building the database you designed in Step 2



```sql
--CREATE EDUCATION TABLE
CREATE TABLE IF NOT EXISTS EDUCATION (
edu_id SERIAL PRIMARY KEY,
edu_level varchar(50)
);

--CREATE EMPLOYEE TABLE
CREATE TABLE IF NOT EXISTS EMPLOYEE (
emp_id varchar(10) primary key,
emp_nm varchar(200),
email varchar (200),
edu_id integer references EDUCATION(edu_id),
hire_Dt date
);

--CREATE JOB TABLE
CREATE TABLE IF NOT EXISTS JOB (
job_id SERIAL PRIMARY KEY,
job_title varchar(50)
);

--CREATE ADDRESS TABLE
CREATE TABLE IF NOT EXISTS ADDRESS (
address_id SERIAL primary key,
address varchar (500),
city varchar (100),
state varchar(50)
);

--CREATE LOCATION TABLE
CREATE TABLE IF NOT EXISTS LOCATION (
loc_id SERIAL primary key,
loc_nm varchar(50),
address_id integer references ADDRESS(address_id)
);

--CREATE DEPARTMENT TABLE
CREATE TABLE IF NOT EXISTS DEPARTMENT (
```

CRUD

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

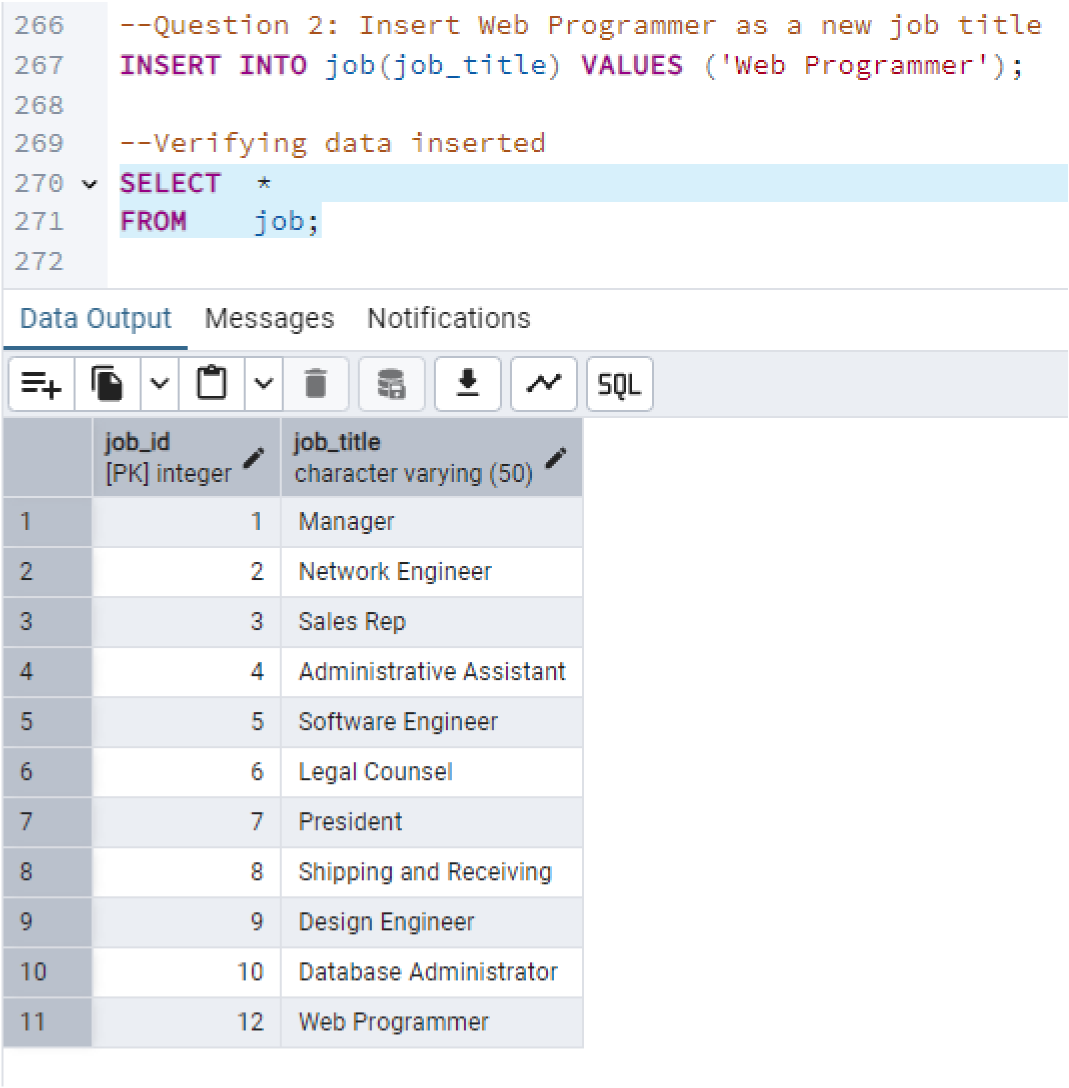


```
--Question 1: Return a list of employees with Job Titles and Department Names
SELECT e.emp_id, j.job_title, d.dept_nm
    FROM employee AS e
    JOIN employment_history AS eh
    ON eh.emp_id = e.emp_id
    JOIN job AS j
    ON eh.job_id = j.job_id
    JOIN department AS d
    ON eh.dept_id = d.dept_id;
```

	emp_id character varying (10)	job_title character varying (50)	dept_nm character varying (50)
1	E70374	Sales Rep	Sales
2	E34496	Administrative Assistant	Sales
3	E16678	Network Engineer	IT
4	E34748	Network Engineer	IT
5	E35053	Administrative Assistant	Distribution
6	E42522	Design Engineer	Product Development
7	E27909	Administrative Assistant	Distribution
8	E47655	Legal Counsel	IT
9	E64494	Software Engineer	IT
10	E40432	Software Engineer	IT
11	E11920	Sales Rep	Sales
12	E29129	Sales Rep	Sales
13	E91075	Network Engineer	IT
14	E87822	Software Engineer	Product Development

CRUD

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

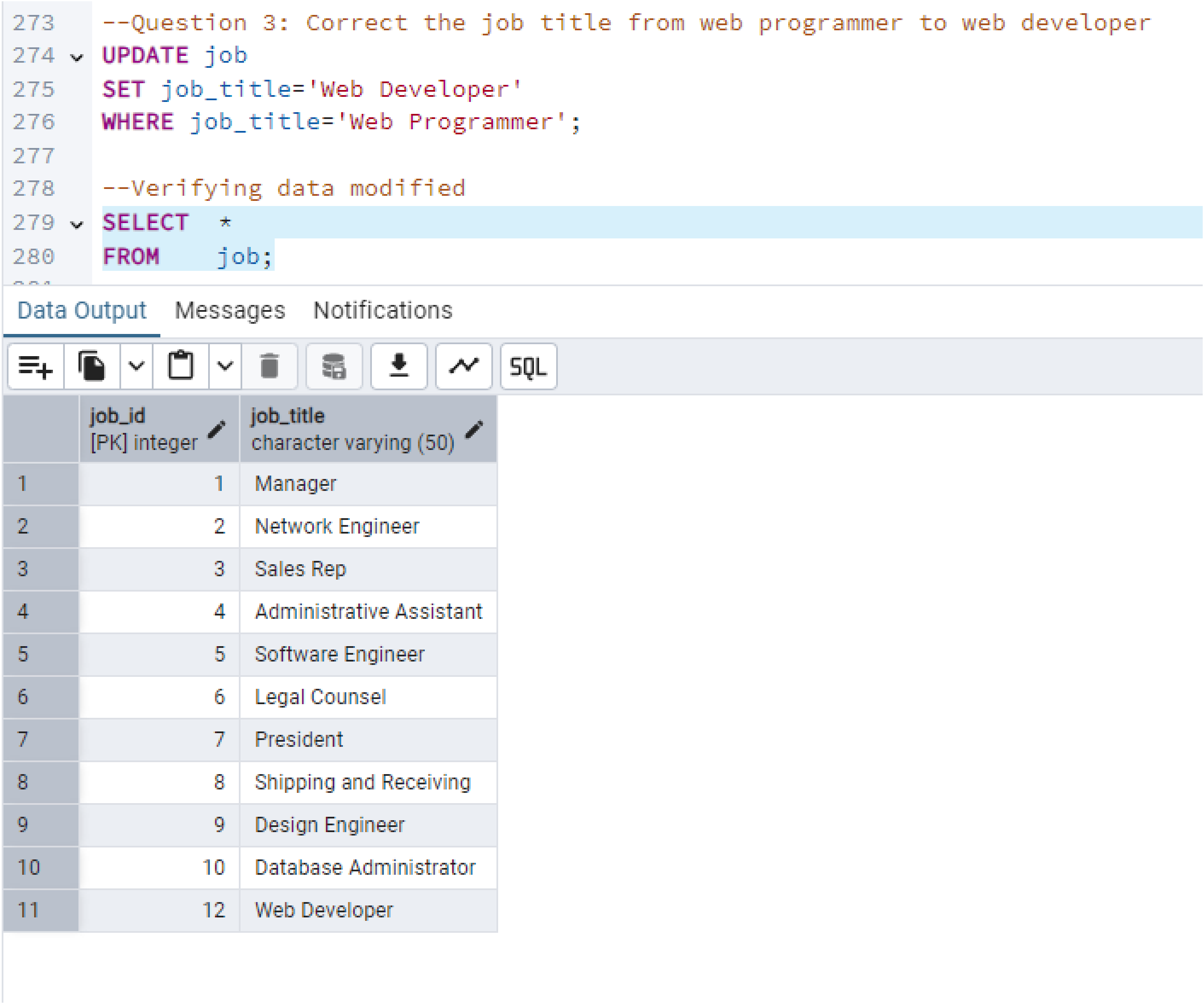


```
--Question 2: Insert Web Programmer as a new job title
INSERT INTO job(job_title) VALUES ('Web Programmer');

--Verifying data inserted
SELECT  *
FROM    job;
```

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

CRUD

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

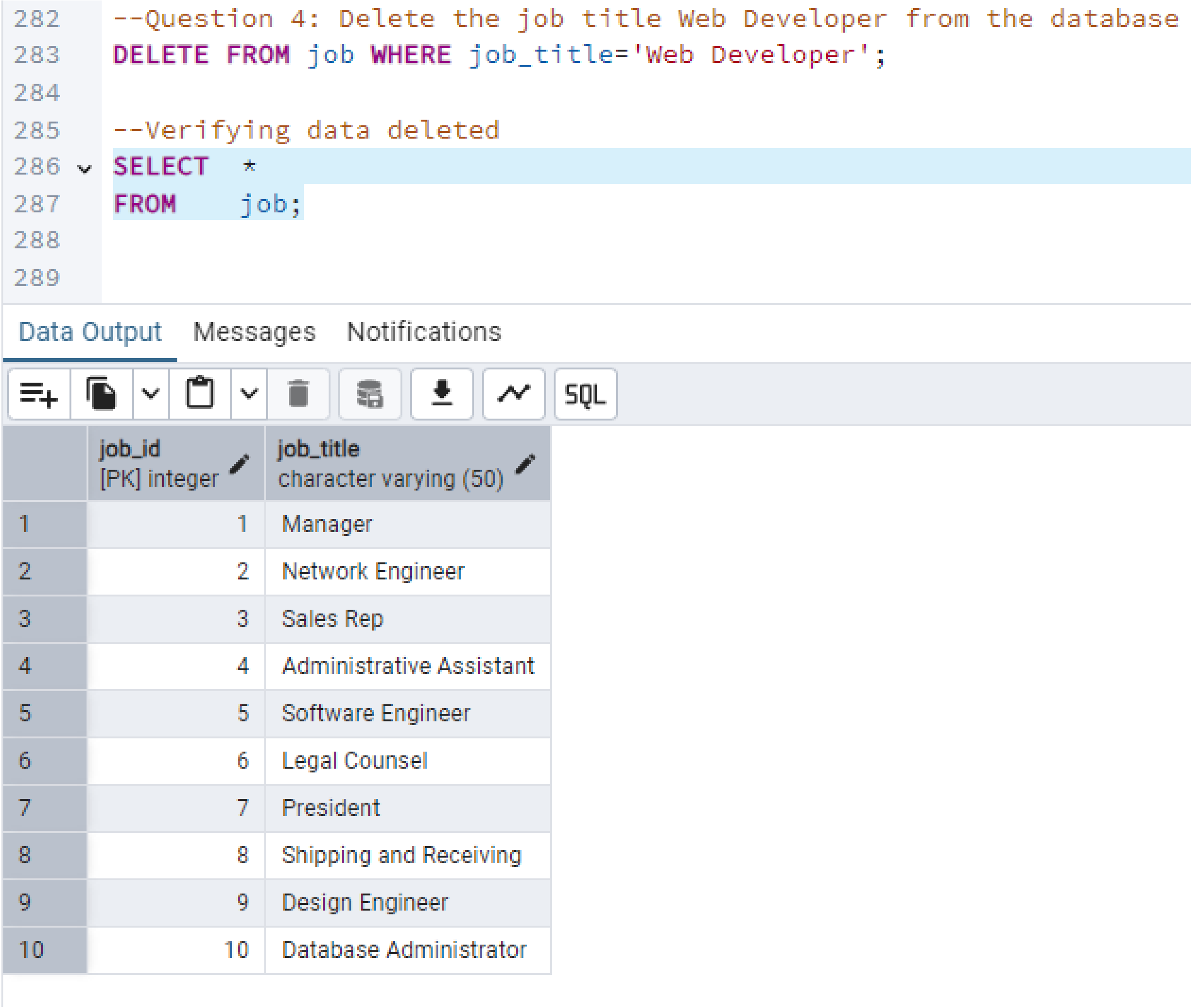


```sql
--Question 3: Correct the job title from web programmer to web developer
UPDATE job
SET job_title='Web Developer'
WHERE job_title='Web Programmer';

--Verifying data modified
SELECT  *
FROM    job;
```

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

CRUD

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

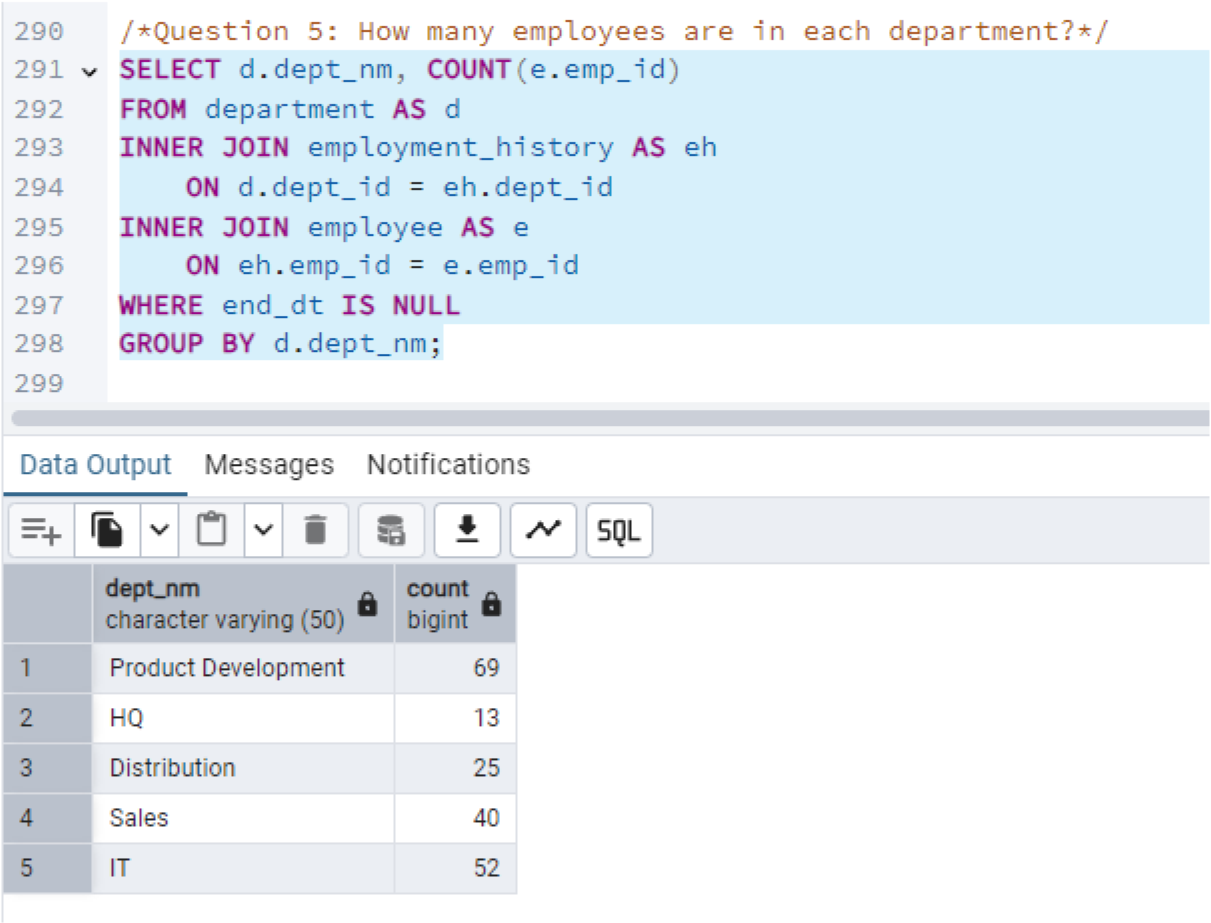


```
--Question 4: Delete the job title Web Developer from the database
DELETE FROM job WHERE job_title='Web Developer';

--Verifying data deleted
SELECT  *
FROM    job;


```

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

CRUD

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

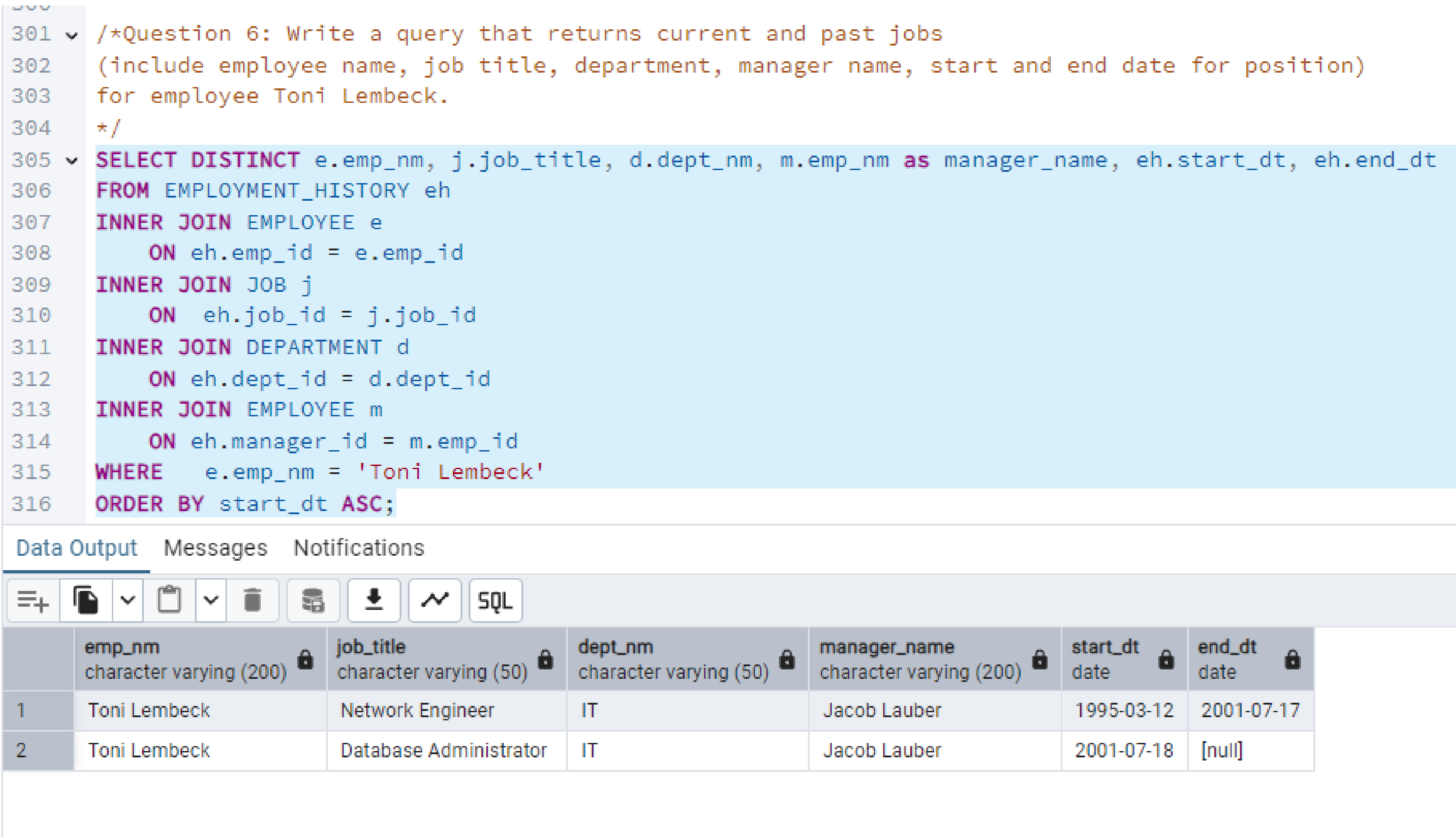


```sql
/*Question 5: How many employees are in each department?*/
SELECT d.dept_nm, COUNT(e.emp_id)
FROM department AS d
INNER JOIN employment_history AS eh
    ON d.dept_id = eh.dept_id
INNER JOIN employee AS e
    ON eh.emp_id = e.emp_id
WHERE end_dt IS NULL
GROUP BY d.dept_nm;
```

	dept_nm character varying (50)	count bigint
1	Product Development	69
2	HQ	13
3	Distribution	25
4	Sales	40
5	IT	52

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.**



```
/*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, j.job_title, d.dept_nm, m.emp_nm as manager_name, eh.start_dt, eh.end_dt
FROM EMPLOYMENT_HISTORY eh
INNER JOIN EMPLOYEE e
    ON eh.emp_id = e.emp_id
INNER JOIN JOB j
    ON  eh.job_id = j.job_id
INNER JOIN DEPARTMENT d
    ON eh.dept_id = d.dept_id
INNER JOIN EMPLOYEE m
    ON eh.manager_id = m.emp_id
WHERE   e.emp_nm = 'Toni Lembeck'
ORDER BY start_dt ASC;
```

	emp_nm character varying (200)	job_title character varying (50)	dept_nm character varying (50)	manager_name character varying (200)	start_dt date	end_dt date
1	Toni Lembeck	Network Engineer	IT	Jacob Lauber	1995-03-12	2001-07-17
2	Toni Lembeck	Database Administrator	IT	Jacob Lauber	2001-07-18	[null]

CRUD

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

We can utilise separate ROLES and object GRANTS so that there is a ROLE with 'elevated access' that has GRANTS on all tables,

and a separate role for employees that has access revoked to a seperate SALARY table.

Existing users should be given the appropriate ROLE to their job role.

When a user is onboarded, they should be given the appropriate role by the ADMIN

Step 4

Above and Beyond

(optional)

Step 4: Above and Beyond

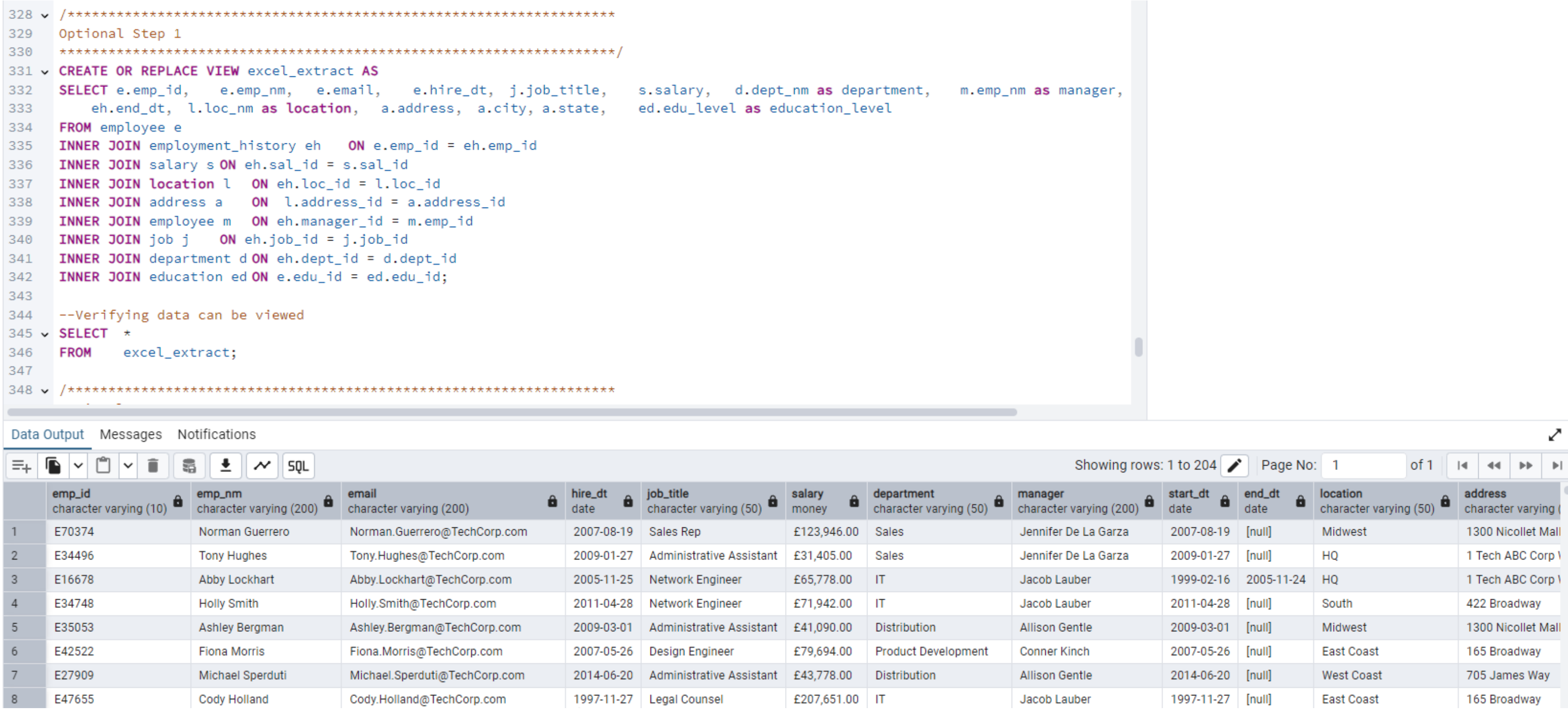
This last step is called Above and Beyond. In this step, I have proposed 3 challenges for you to complete, which are above and beyond the scope of the project. This is a chance to flex your coding muscles and show everyone how good you really are.

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

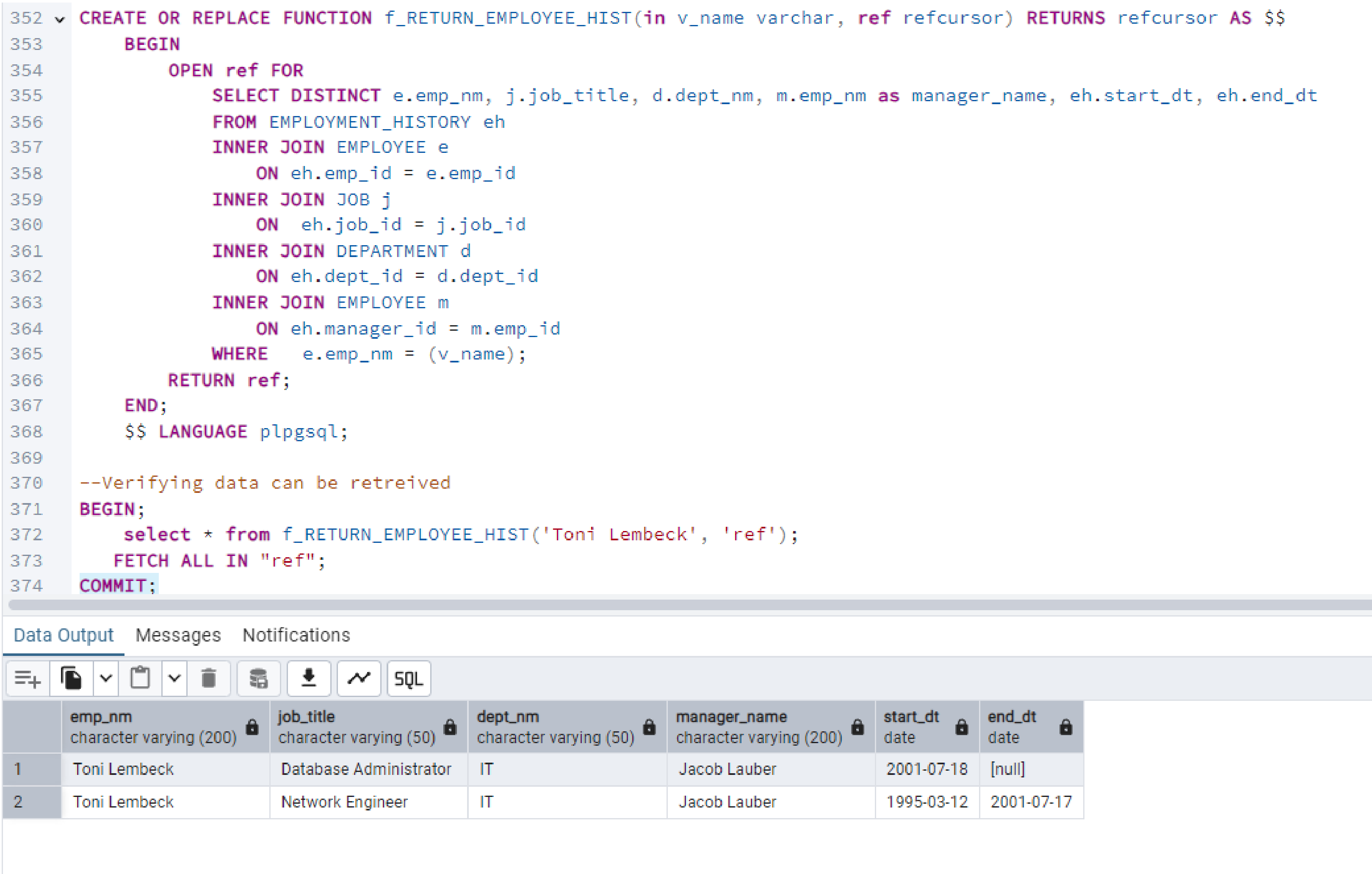


```
/**********************************************************************
Optional Step 1
**********************************************************************/
CREATE OR REPLACE VIEW excel_extract AS
SELECT e.emp_id,    e.emp_nm,   e.email,    e.hire_dt,  j.job_title,    s.salary,   d.dept_nm as department,    m.emp_nm as manager,
    eh.end_dt,  l.loc_nm as location,   a.address,  a.city, a.state,    ed.edu_level as education_level
FROM employee e
INNER JOIN employment_history eh    ON e.emp_id = eh.emp_id
INNER JOIN salary s ON eh.sal_id = s.sal_id
INNER JOIN location l   ON eh.loc_id = l.loc_id
INNER JOIN address a    ON  l.address_id = a.address_id
INNER JOIN employee m   ON eh.manager_id = m.emp_id
INNER JOIN job j    ON eh.job_id = j.job_id
INNER JOIN department d ON eh.dept_id = d.dept_id
INNER JOIN education ed ON e.edu_id = ed.edu_id;

--Verifying data can be viewed
SELECT  *
FROM    excel_extract;

/**********************************************************************
```

	emp_id character varying (10)	emp_nm character varying (200)	email character varying (200)	hire_dt date	job_title character varying (50)	salary money	department character varying (50)	manager character varying (200)	start_dt date	end_dt date	location character varying (50)	address character varying (
1	E70374	Norman Guerrero	Norman.Guerrero@TechCorp.com	2007-08-19	Sales Rep	£123,946.00	Sales	Jennifer De La Garza	2007-08-19	[null]	Midwest	1300 Nicollet Mall
2	E34496	Tony Hughes	Tony.Hughes@TechCorp.com	2009-01-27	Administrative Assistant	£31,405.00	Sales	Jennifer De La Garza	2009-01-27	[null]	HQ	1 Tech ABC Corp
3	E16678	Abby Lockhart	Abby.Lockhart@TechCorp.com	2005-11-25	Network Engineer	£65,778.00	IT	Jacob Lauber	1999-02-16	2005-11-24	HQ	1 Tech ABC Corp
4	E34748	Holly Smith	Holly.Smith@TechCorp.com	2011-04-28	Network Engineer	£71,942.00	IT	Jacob Lauber	2011-04-28	[null]	South	422 Broadway
5	E35053	Ashley Bergman	Ashley.Bergman@TechCorp.com	2009-03-01	Administrative Assistant	£41,090.00	Distribution	Allison Gentle	2009-03-01	[null]	Midwest	1300 Nicollet Mall
6	E42522	Fiona Morris	Fiona.Morris@TechCorp.com	2007-05-26	Design Engineer	£79,694.00	Product Development	Conner Kinch	2007-05-26	[null]	East Coast	165 Broadway
7	E27909	Michael Sperduti	Michael.Sperduti@TechCorp.com	2014-06-20	Administrative Assistant	£43,778.00	Distribution	Allison Gentle	2014-06-20	[null]	West Coast	705 James Way
8	E47655	Cody Holland	Cody.Holland@TechCorp.com	1997-11-27	Legal Counsel	£207,651.00	IT	Jacob Lauber	1997-11-27	[null]	East Coast	165 Broadway

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.

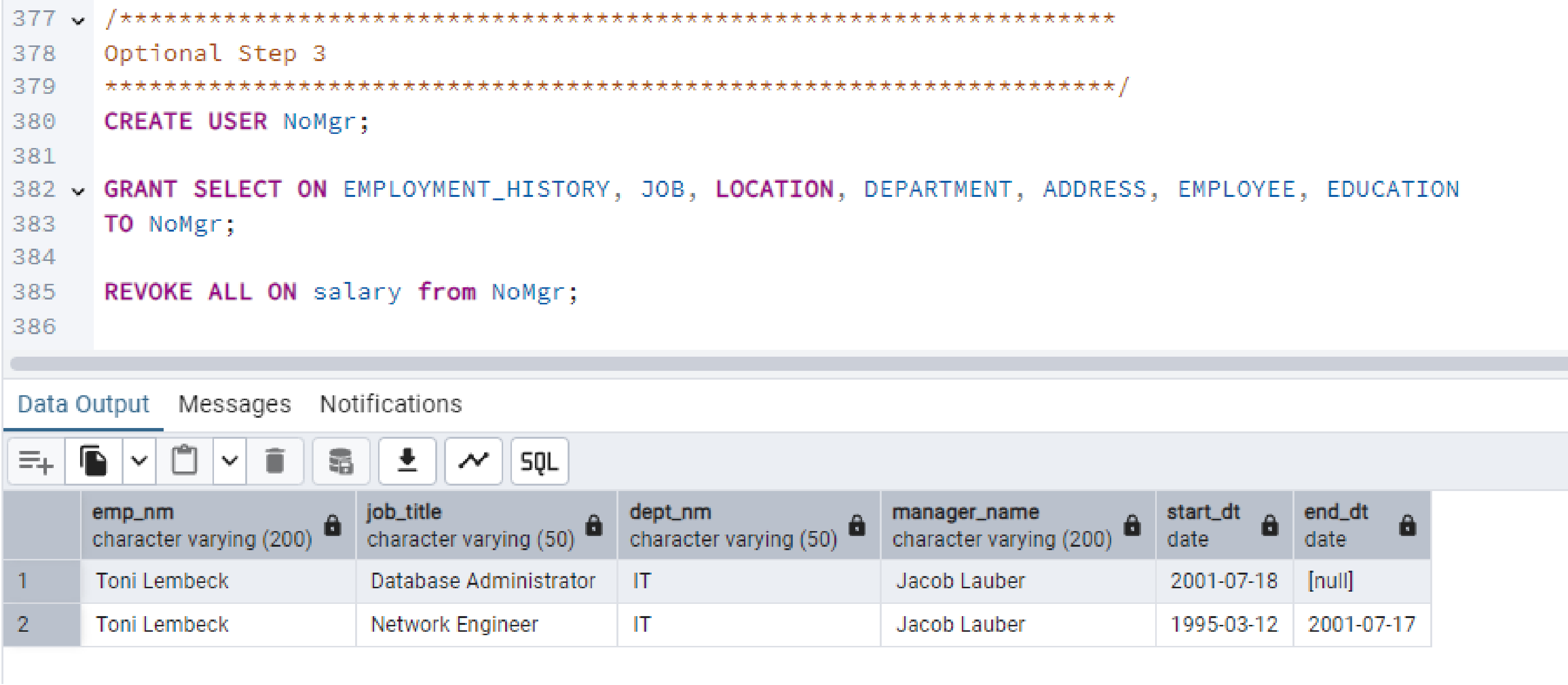


```
CREATE OR REPLACE FUNCTION f_RETURN_EMPLOYEE_HIST(in v_name varchar, ref refcursor) RETURNS refcursor AS $$
    BEGIN
        OPEN ref FOR
            SELECT DISTINCT e.emp_nm, j.job_title, d.dept_nm, m.emp_nm as manager_name, eh.start_dt, eh.end_dt
            FROM EMPLOYMENT_HISTORY eh
            INNER JOIN EMPLOYEE e
                ON eh.emp_id = e.emp_id
            INNER JOIN JOB j
                ON  eh.job_id = j.job_id
            INNER JOIN DEPARTMENT d
                ON eh.dept_id = d.dept_id
            INNER JOIN EMPLOYEE m
                ON eh.manager_id = m.emp_id
            WHERE   e.emp_nm = (v_name);
        RETURN ref;
    END;
    $$ LANGUAGE plpgsql;

--Verifying data can be retreived
BEGIN;
    select * from f_RETURN_EMPLOYEE_HIST('Toni Lembeck', 'ref');
   FETCH ALL IN "ref";
COMMIT;
```

	emp_nm character varying (200)	job_title character varying (50)	dept_nm character varying (50)	manager_name character varying (200)	start_dt date	end_dt date
1	Toni Lembeck	Database Administrator	IT	Jacob Lauber	2001-07-18	[null]
2	Toni Lembeck	Network Engineer	IT	Jacob Lauber	1995-03-12	2001-07-17

Standout Suggestion 3

Implement user security on the restricted salary attribute.



```
/********************************************************************
Optional Step 3
********************************************************************/
CREATE USER NoMgr;

GRANT SELECT ON EMPLOYMENT_HISTORY, JOB, LOCATION, DEPARTMENT, ADDRESS, EMPLOYEE, EDUCATION
TO NoMgr;

REVOKE ALL ON salary from NoMgr;

```

	emp_nm character varying (200)	job_title character varying (50)	dept_nm character varying (50)	manager_name character varying (200)	start_dt date	end_dt date
1	Toni Lembeck	Database Administrator	IT	Jacob Lauber	2001-07-18	[null]
2	Toni Lembeck	Network Engineer	IT	Jacob Lauber	1995-03-12	2001-07-17

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.