Tech ABC Corp - HR Database

Oscar Mendoza – 02/10/2021

DATE IN DD/MM/YYYY

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:**

  HR is asking for a database to store Employee's information. Since the company is growing steadily, some issues regarding data security and management have been raised, and thus, the former Excel-spreadsheet should be now a PostgreSQL database.

- **Current situation and management**

  The current data is been stored in a shared Excel spreadsheet, being the HR team in charge of its manual updating and managing.

- **Current data available**

  The current data is a human-readable file, storing personal data from all employees in the company, including the salaries.

- **Further requests**

  The data should be stored for 7 years, according to regulations and it is considered critical.

- **Requested Data Management**

  The data should be managed by HR, although Management will also have Access to write in the DB.

- **Requested Data Security**
  Employees – Read only Access, salary information not displayed
  HR – Total access to the data
  Management level employees – total access to data.

Data Architect Business Requirement

- **Estimated size of database**

  205 rows of employee data.

- **Estimated annual growth**

  20% per year for the following 5 years. (Approximately, 40 new employees per year)

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

  Salary data is restricted.

Data Architect Technical Requirement

- **Justification for the new database**

  - By migrating the data from the spreadsheet to a database, there is more control on security and access to critical and sensible data.

  - By having all data concentrated in a database, the possible issues with data integrity are reduced and thus the data is more reliable.

- **Database objects**

  Tables: employee, education, jobhist, salary, job, department, address, location, city, state

  Views: Report

- **Data ingestion**

  For feeding the database, the selected method of data ingestion is ETL. Probably, an EXCEL file will be provided with new information, so the Extraction, Transformation and Loading will be all automated.

Data Architect Technical Requirement

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

  Ownership: HR will own and maintain the data

  User Access: Employees with domain login will have read access, employees from Management Depts will have write and read access.

- **Scalability**

  Replication is better for scalability, since 90% of the users will have reading access.

- **Flexibility**

  In order to connect with the payroll system, proper documentation will be needed regarding the data dictionary and Reference Data to integrate both data systems for reporting.

- **Storage & retention**

  Storage (disk or in-memory): disk storage, 1 GB partition is enough.

  Retention: 7 years.

- **Backup**

  Although the rate of increment for the data is low, it is still considered as critical. For this reason, the Backup will be schedule as follows, according to IT Best Practices Guide:

  - Full backu 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.

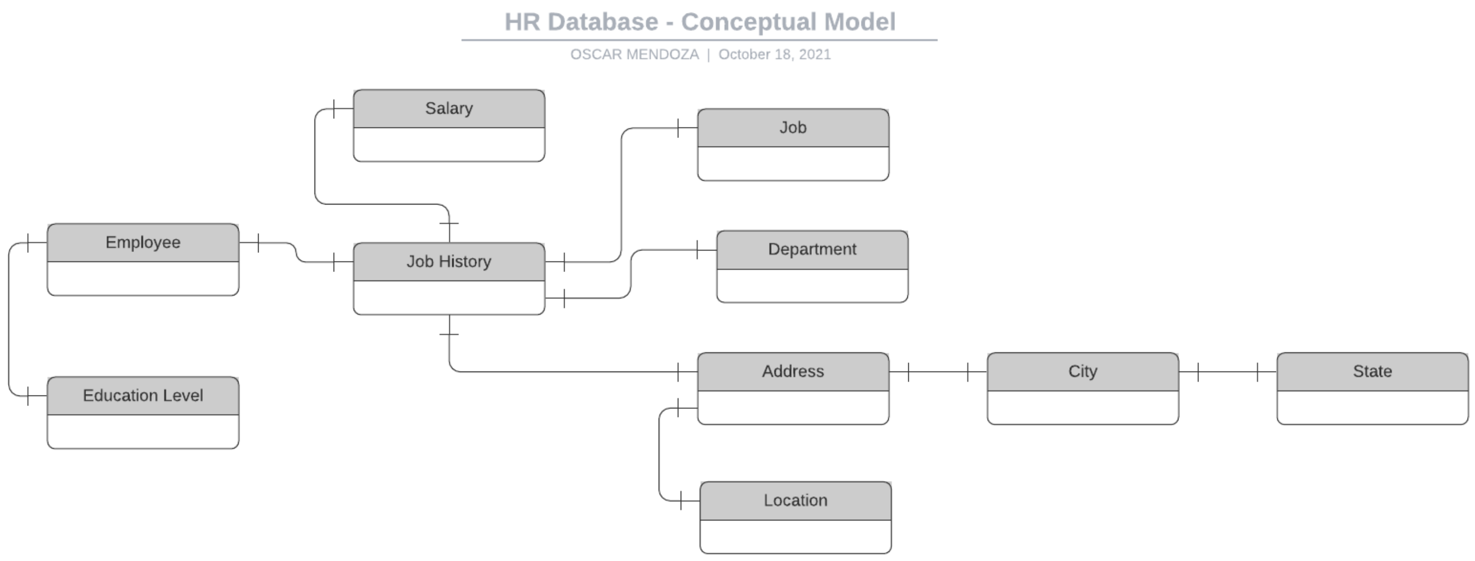
You will submit a screenshot for each of the 3 ERDs you create. You will find detailed instructions for developing each of the ERDs over the next several pages.

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.

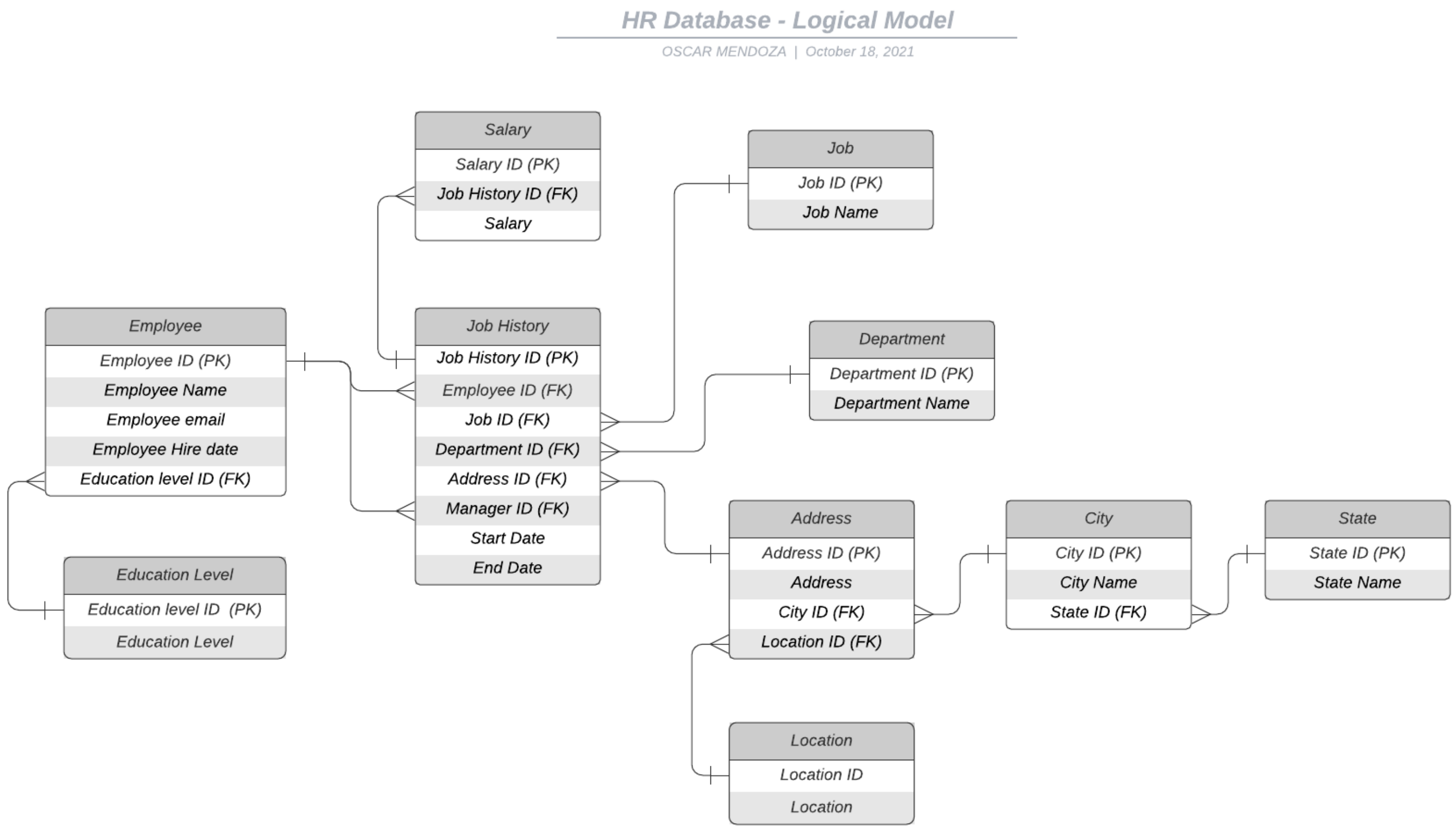
Use Lucidchart's built-in template for DBMS ER Diagram UML.



ERD

- **Logical**

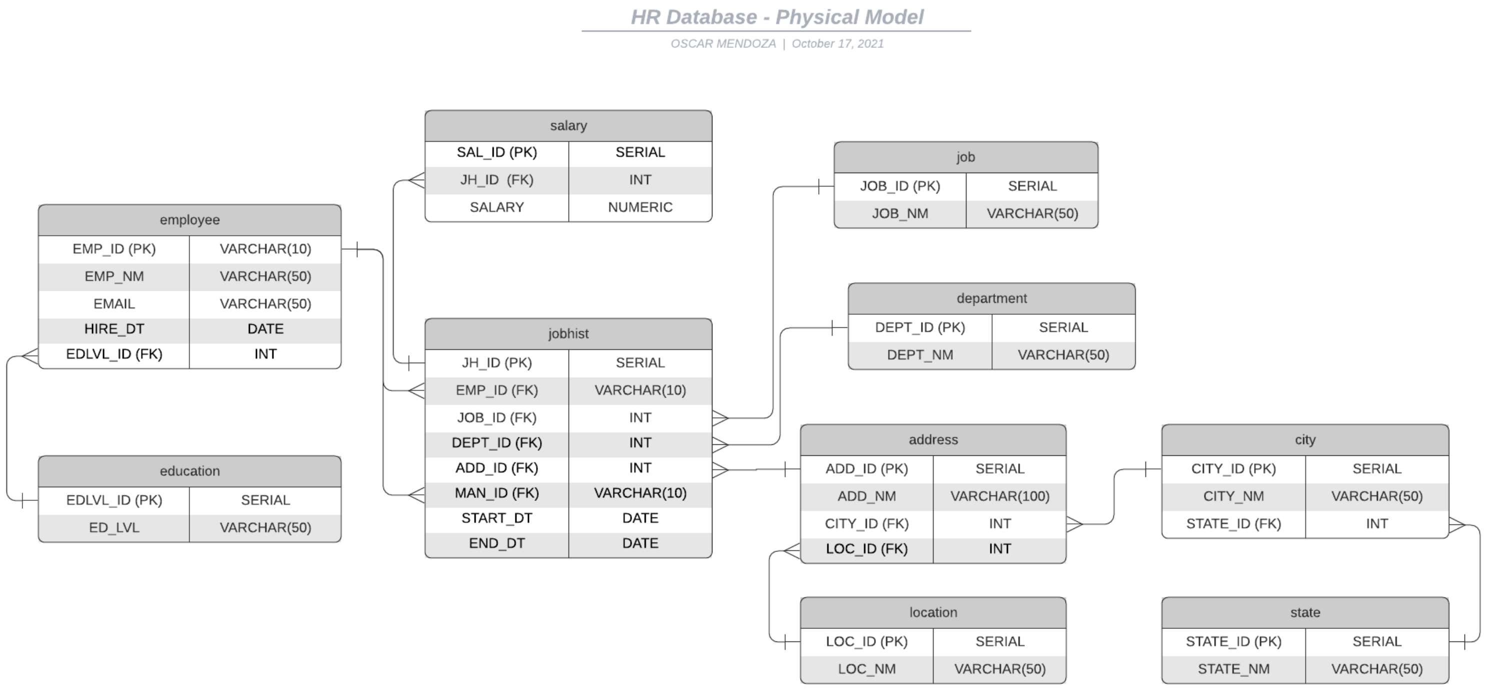
The logical model is the next level of refinement from the conceptual ERD. At this point, you should have normalized the data to the 3NF. Attributes should also be listed now in the ERD. You can still use human-friendly entity and attribute names in the logical model, and while relationship lines are required, Crow's foot notation is still not needed at this point.



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

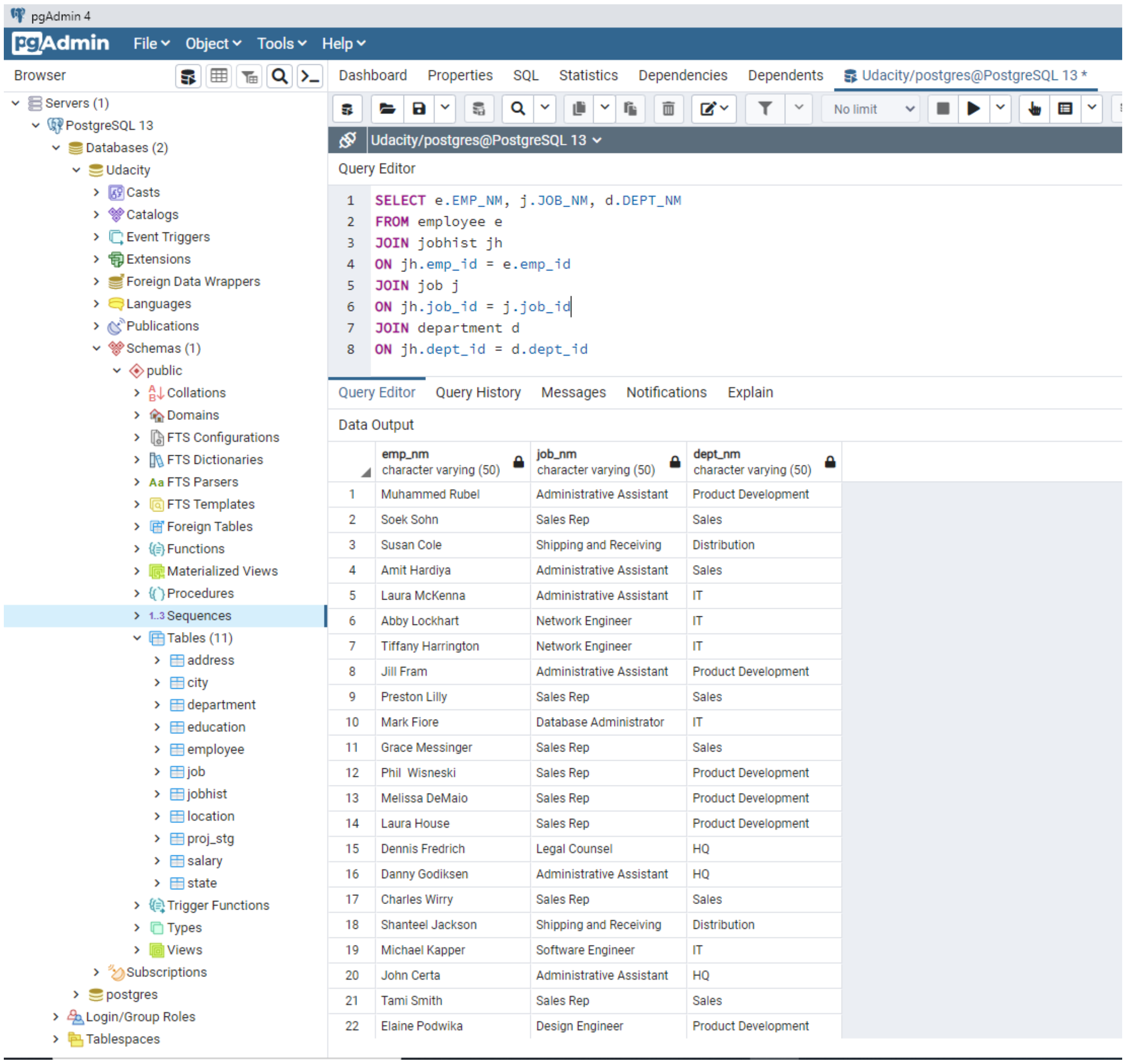
```sql
/*Tables creation following the physical model*/

CREATE TABLE education(
    EDLVL_ID SERIAL PRIMARY KEY,
    ED_LVL VARCHAR(50)
);

CREATE TABLE employee(
    EMP_ID VARCHAR(10) PRIMARY KEY,
    EMP_NM VARCHAR(50),
    EMAIL VARCHAR(50),
    HIRE_DT DATE,
    EDLVL_ID INT REFERENCES education(EDLVL_ID)
);

CREATE TABLE department(
    DEPT_ID SERIAL PRIMARY KEY,
    DEPT_NM VARCHAR(50)
);

CREATE TABLE job(
    JOB_ID SERIAL PRIMARY KEY,
    JOB_NM VARCHAR(50)
);

CREATE TABLE state(
    STATE_ID SERIAL PRIMARY KEY,
    STATE_NM VARCHAR(50)
);

CREATE TABLE city(
    CITY_ID SERIAL PRIMARY KEY,
    CITY_NM VARCHAR(50),
    STATE_ID INT REFERENCES state(STATE_ID)
);

CREATE TABLE location(
    LOC_ID SERIAL PRIMARY KEY,
    LOC_NM VARCHAR(50)
);

CREATE TABLE address(
    ADD_ID SERIAL PRIMARY KEY,
    ADD_NM VARCHAR(100),
    CITY_ID INT REFERENCES city(CITY_ID),
    LOC_ID INT REFERENCES location(LOC_ID)
);

CREATE TABLE jobhist(
    JH_ID SERIAL PRIMARY KEY,
    EMP_ID VARCHAR(10) REFERENCES employee(EMP_ID),
    JOB_ID INT REFERENCES job(JOB_ID),
    ADD_ID INT REFERENCES address(ADD_ID),
    MAN_ID VARCHAR(10) REFERENCES employee(EMP_ID),
    START_DT DATE,
    END_DT DATE
);

CREATE TABLE salary(
    SAL_ID SERIAL PRIMARY KEY,
    JH_ID INT REFERENCES jobhist(JH_ID),
    SALARY NUMERIC
);
```

DDL - ETL

```
INSERT INTO education(ED_LVL)
SELECT DISTINCT education_lvl
FROM proj_stg;

INSERT INTO employee
SELECT DISTINCT p.emp_id, p.emp_nm, p.email, p.hire_dt, e.edlvl_id
FROM proj_stg p
JOIN education e
ON p.education_lvl = e.ED_LVL;

INSERT INTO department(DEPT_NM)
SELECT DISTINCT department_nm
FROM proj_stg;

INSERT INTO job(JOB_NM)
SELECT DISTINCT job_title
FROM proj_stg;

INSERT INTO state(STATE_NM)
SELECT DISTINCT state
FROM proj_stg;

INSERT INTO city(CITY_NM, STATE_ID)
SELECT DISTINCT p.city, s.STATE_ID
FROM proj_stg p
JOIN state s
ON s.STATE_NM = P.state;

INSERT INTO location(LOC_NM)
SELECT DISTINCT location
FROM proj_stg;

INSERT INTO address(ADD_NM, CITY_ID, LOC_ID)
SELECT DISTINCT p.address, c.CITY_ID, l.LOC_ID
FROM proj_stg p
JOIN city c
ON p.city = c.CITY_NM
JOIN location l
ON p.location = l.LOC_NM;

INSERT INTO jobhist(EMP_ID, JOB_ID, DEPT_ID, ADD_ID, MAN_ID, START_DT, END_DT)
SELECT DISTINCT p.EMP_ID, j.JOB_ID, d.DEPT_ID, a.ADD_ID, e.EMP_ID AS MAN_ID, START_DT, END_DT
FROM proj_stg p
JOIN job j
ON p.job_title = j.JOB_NM
JOIN department d
ON p.department_nm = d.DEPT_NM
JOIN address a
ON p.address = a.ADD_NM
LEFT JOIN employee e
ON p.manager = e.EMP_NM;

INSERT INTO salary(JH_ID, SALARY)
SELECT jh.JH_ID, p.salary
FROM proj_stg p
JOIN jobhist jh
ON p.emp_id = jh.emp_id AND p.start_dt = jh.start_dt AND p.end_dt = jh.end_dt;
```

CRUD

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



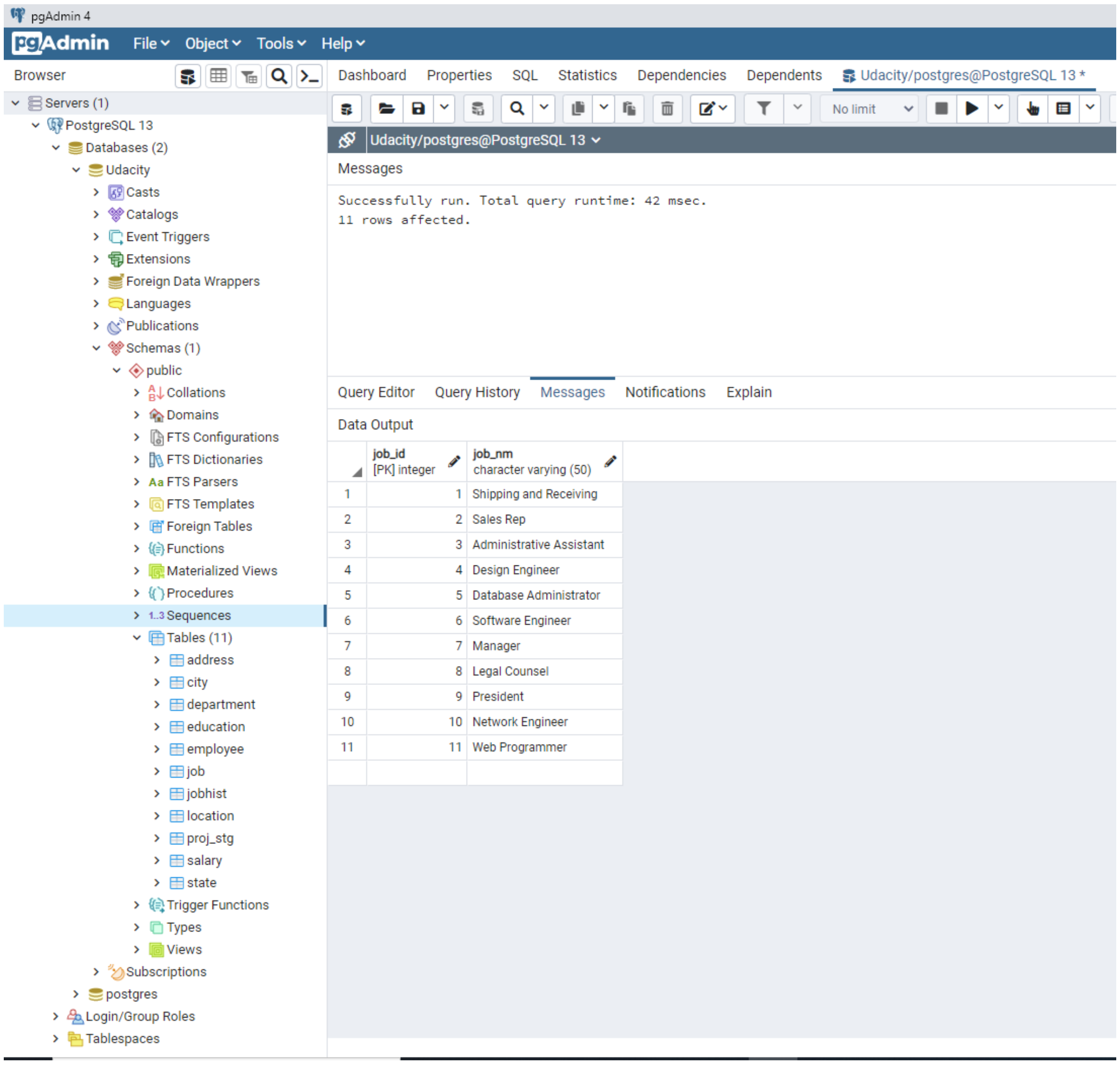
```sql
SELECT e.EMP_NM, j.JOB_NM, d.DEPT_NM
FROM employee e
JOIN jobhist jh
ON jh.emp_id = e.emp_id
JOIN job j
ON jh.job_id = j.job_id
JOIN department d
ON jh.dept_id = d.dept_id
```

Data Output

	emp_nm character varying (50)	job_nm character varying (50)	dept_nm character varying (50)
1	Muhammed Rubel	Administrative Assistant	Product Development
2	Soek Sohn	Sales Rep	Sales
3	Susan Cole	Shipping and Receiving	Distribution
4	Amit Hardiya	Administrative Assistant	Sales
5	Laura McKenna	Administrative Assistant	IT
6	Abby Lockhart	Network Engineer	IT
7	Tiffany Harrington	Network Engineer	IT
8	Jill Fram	Administrative Assistant	Product Development
9	Preston Lilly	Sales Rep	Sales
10	Mark Fiore	Database Administrator	IT
11	Grace Messinger	Sales Rep	Sales
12	Phil Wisneski	Sales Rep	Product Development
13	Melissa DeMaio	Sales Rep	Product Development
14	Laura House	Sales Rep	Product Development
15	Dennis Fredrich	Legal Counsel	HQ
16	Danny Godiksen	Administrative Assistant	HQ
17	Charles Wirry	Sales Rep	Sales
18	Shanteel Jackson	Shipping and Receiving	Distribution
19	Michael Kapper	Software Engineer	IT
20	John Certa	Administrative Assistant	HQ
21	Tami Smith	Sales Rep	Sales
22	Elaine Podwika	Design Engineer	Product Development

CRUD

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



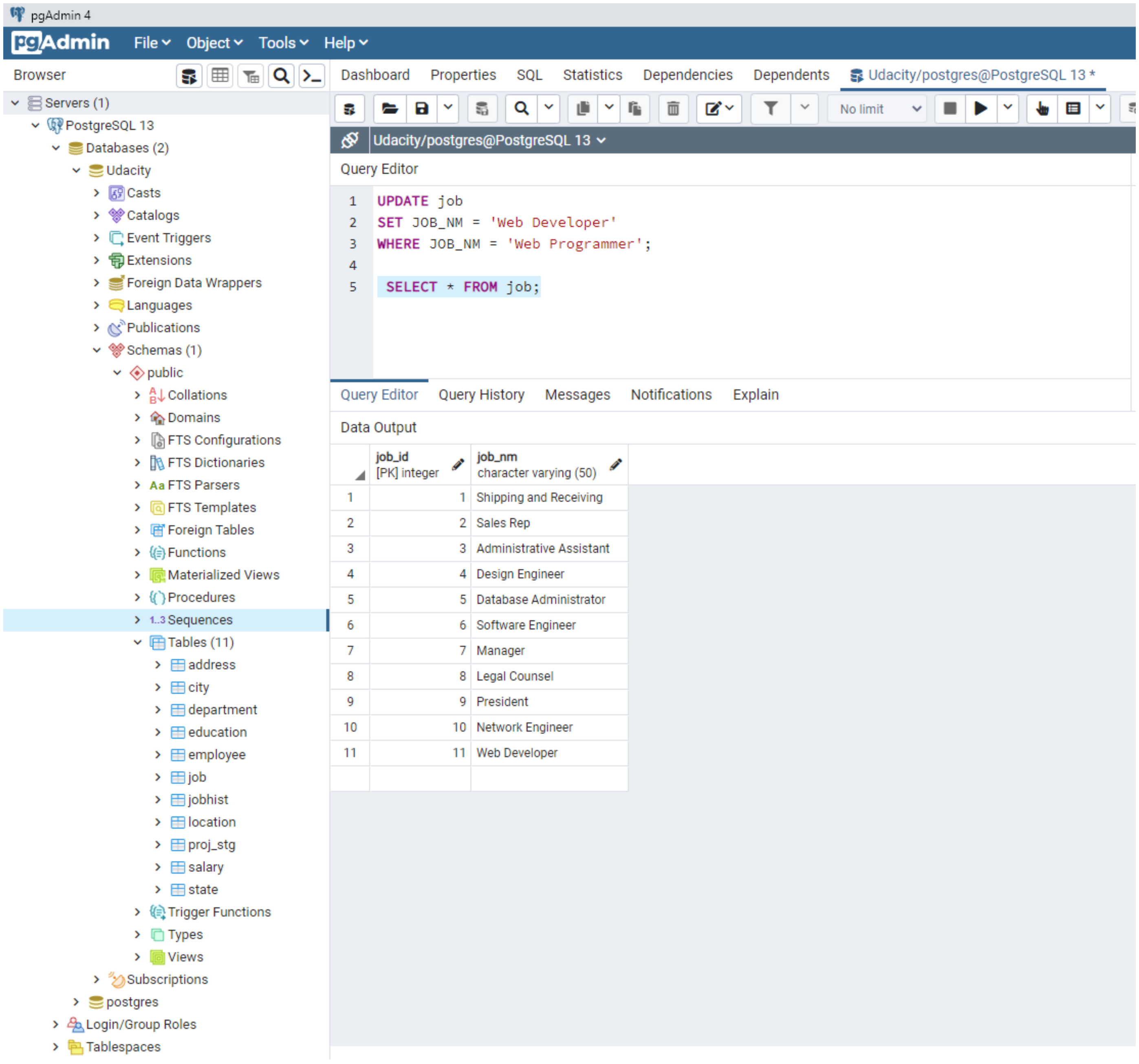
Messages

```
Successfully run. Total query runtime: 42 msec.
11 rows affected.
```

	job_id [PK] integer	job_nm 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	11	Web Programmer

CRUD

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

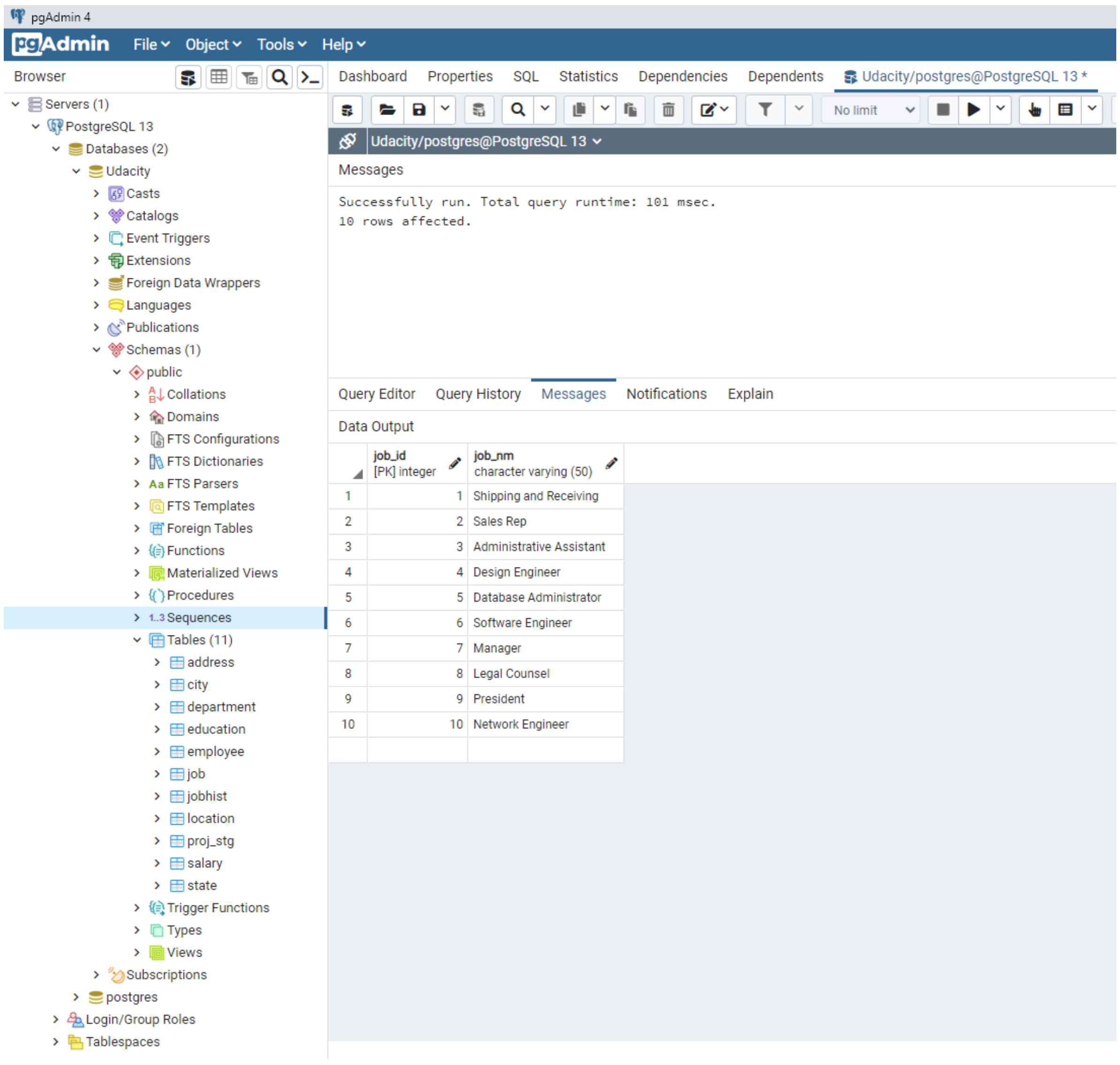


```
UPDATE job
SET JOB_NM = 'Web Developer'
WHERE JOB_NM = 'Web Programmer';

SELECT * FROM job;
```

	job_id [PK] integer	job_nm 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	11	Web Developer

CRUD

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



Messages

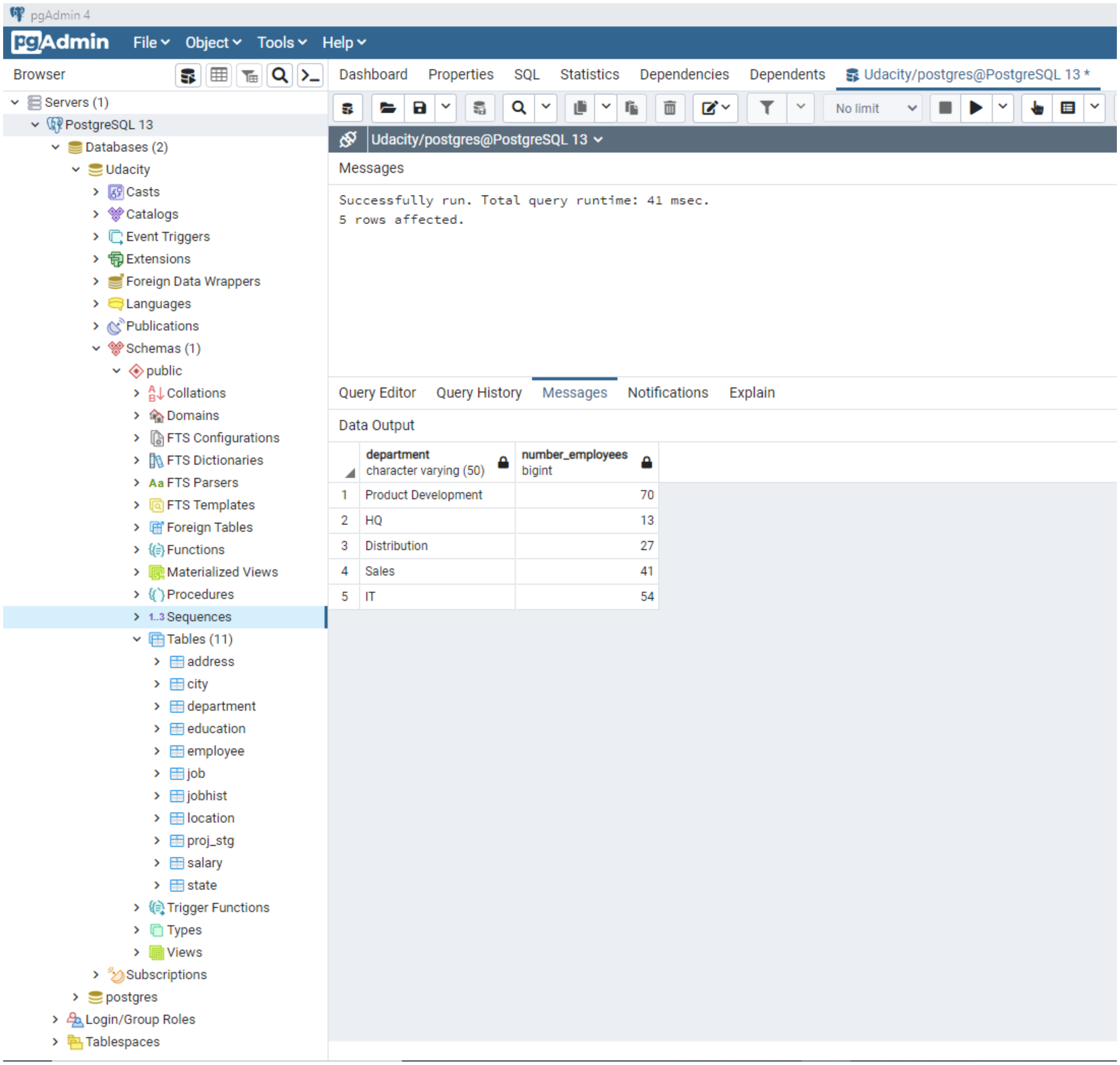
```
Successfully run. Total query runtime: 101 msec.
10 rows affected.
```

Data Output

	job_id [PK] integer	job_nm 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?**



Udacity/postgres@PostgreSQL 13

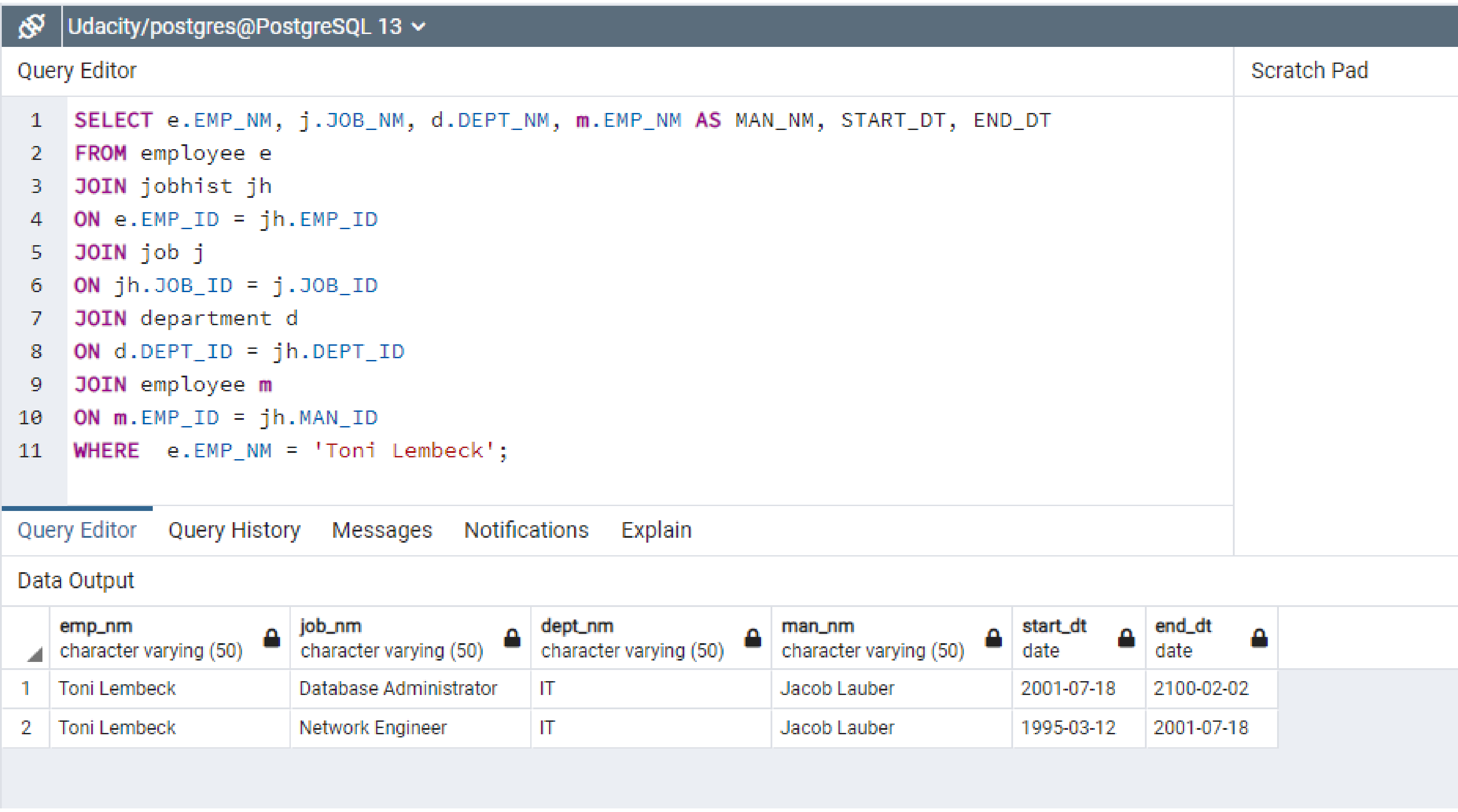
Messages

```
Successfully run. Total query runtime: 41 msec.
5 rows affected.
```

	department character varying (50)	number_employees bigint
1	Product Development	70
2	HQ	13
3	Distribution	27
4	Sales	41
5	IT	54

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.**



Udacity/postgres@PostgreSQL 13

Query Editor

Scratch Pad

```sql
SELECT e.EMP_NM, j.JOB_NM, d.DEPT_NM, m.EMP_NM AS MAN_NM, START_DT, END_DT
FROM employee e
JOIN jobhist jh
ON e.EMP_ID = jh.EMP_ID
JOIN job j
ON jh.JOB_ID = j.JOB_ID
JOIN department d
ON d.DEPT_ID = jh.DEPT_ID
JOIN employee m
ON m.EMP_ID = jh.MAN_ID
WHERE  e.EMP_NM = 'Toni Lembeck';
```

Query Editor | Query History | Messages | Notifications | Explain

Data Output

	emp_nm character varying (50)	job_nm character varying (50)	dept_nm character varying (50)	man_nm 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.**

Since salary data is stored in a separated table, access to this data would be done through table-level security. All data users with access permission (both on read and read/write roles) are identified and given permission using company’s login credentials.

Step 4

Above and Beyond

(optional)

Step 4: Above and Beyond

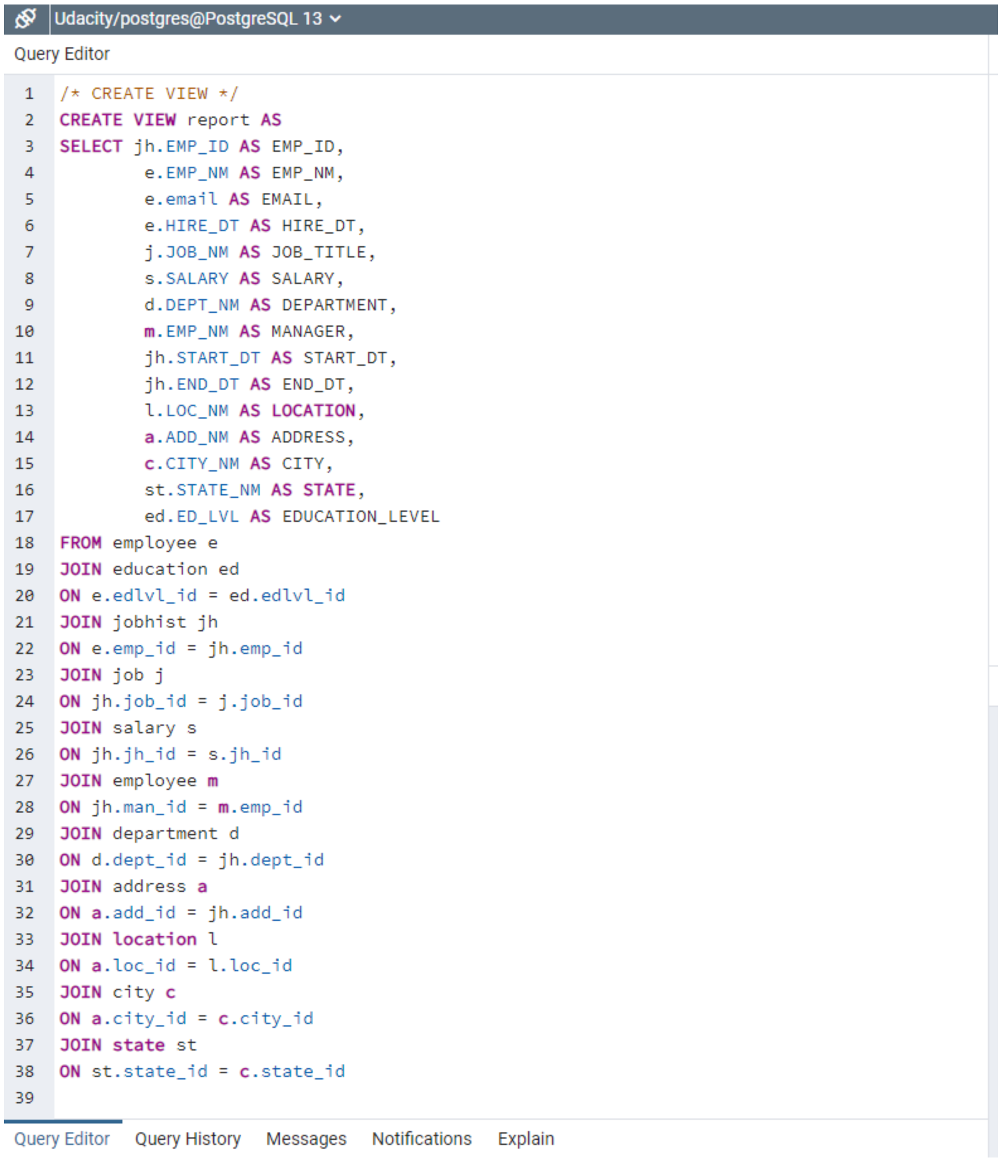
This last step is called Above and Beyond. In this step, I have proposed 3 challenges for you to complete, which are above and beyond the scope of the project. This is a chance to flex your coding muscles and show everyone how good you really are.

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

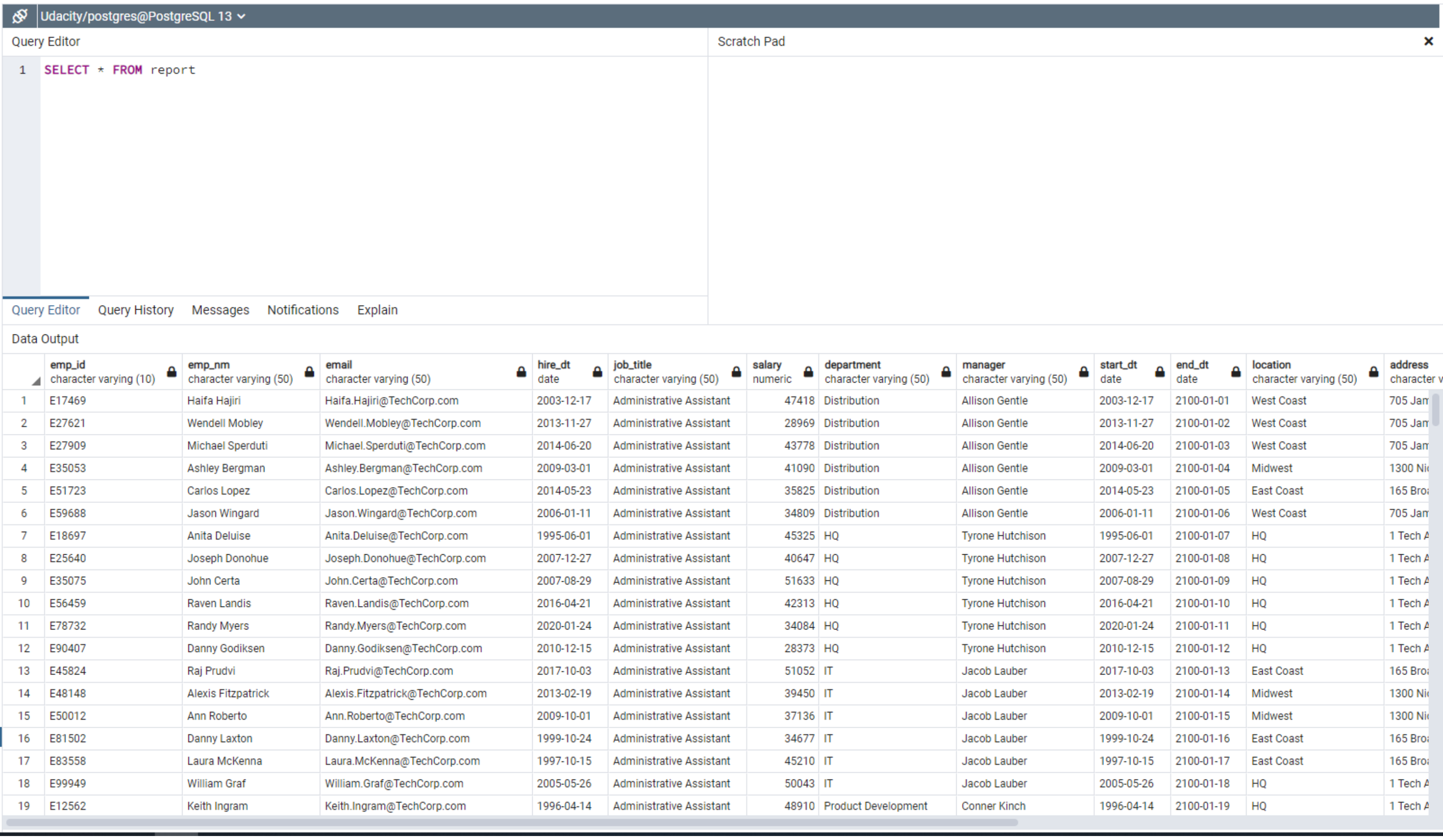
Create a view that returns all employee attributes; results should resemble initial Excel file



```sql
/* CREATE VIEW */
CREATE VIEW report AS
SELECT jh.EMP_ID AS EMP_ID,
        e.EMP_NM AS EMP_NM,
        e.email AS EMAIL,
        e.HIRE_DT AS HIRE_DT,
        j.JOB_NM AS JOB_TITLE,
        s.SALARY AS SALARY,
        d.DEPT_NM AS DEPARTMENT,
        m.EMP_NM AS MANAGER,
        jh.START_DT AS START_DT,
        jh.END_DT AS END_DT,
        l.LOC_NM AS LOCATION,
        a.ADD_NM AS ADDRESS,
        c.CITY_NM AS CITY,
        st.STATE_NM AS STATE,
        ed.ED_LVL AS EDUCATION_LEVEL
FROM employee e
JOIN education ed
ON e.edlvl_id = ed.edlvl_id
JOIN jobhist jh
ON e.emp_id = jh.emp_id
JOIN job j
ON jh.job_id = j.job_id
JOIN salary s
ON jh.jh_id = s.jh_id
JOIN employee m
ON jh.man_id = m.emp_id
JOIN department d
ON d.dept_id = jh.dept_id
JOIN address a
ON a.add_id = jh.add_id
JOIN location l
ON a.loc_id = l.loc_id
JOIN city c
ON a.city_id = c.city_id
JOIN state st
ON st.state_id = c.state_id
```

Standout Suggestion 1

Create a view that returns all employee attributes; results should resemble initial Excel file



Udacity/postgres@PostgreSQL 13

Query Editor

```
SELECT * FROM report
```

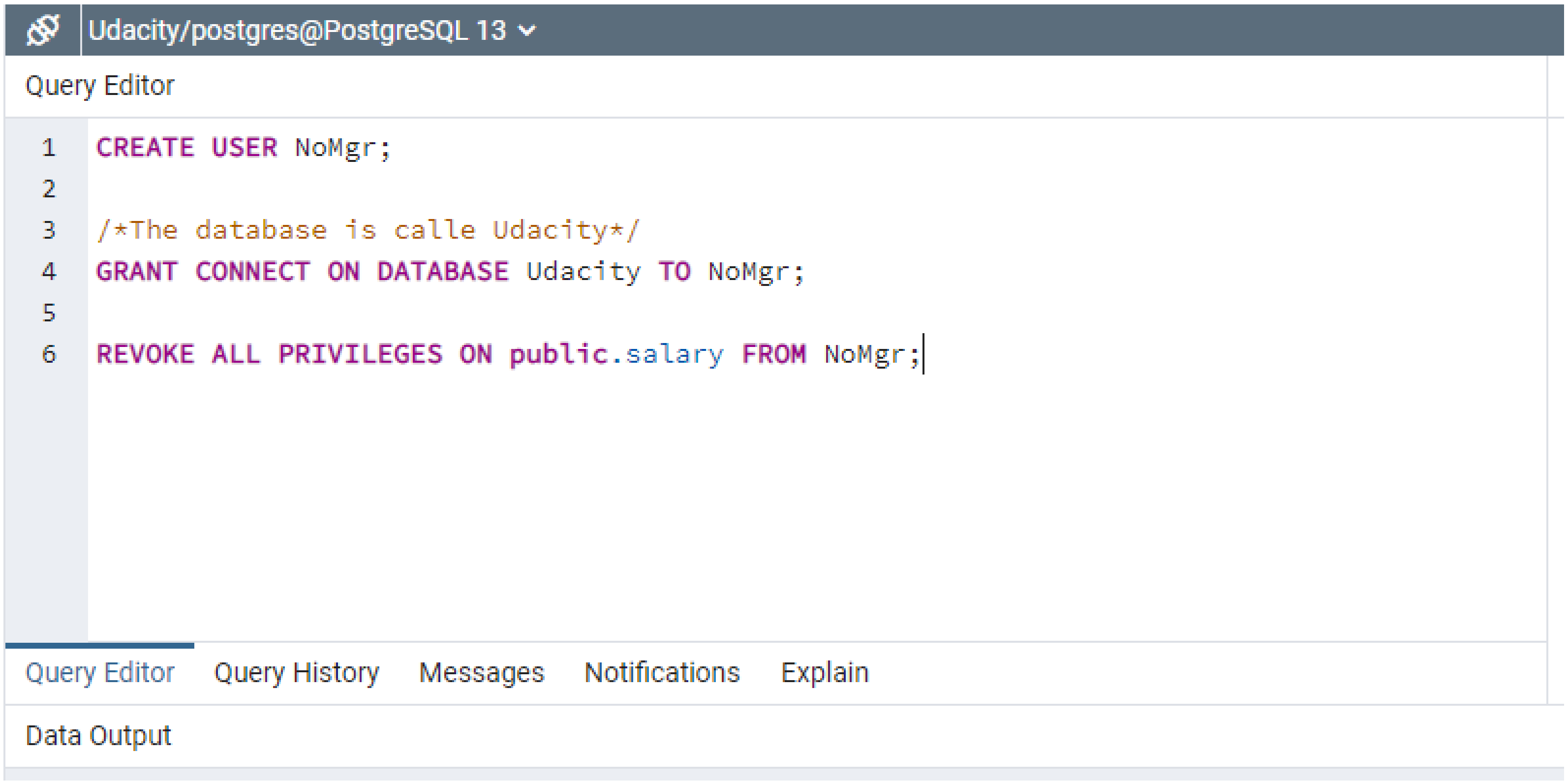
Data Output

	emp_id character varying (10)	emp_nm character varying (50)	email character varying (50)	hire_dt date	job_title character varying (50)	salary numeric	department character varying (50)	manager character varying (50)	start_dt date	end_dt date	location character varying (50)	address character v
1	E17469	Haifa Hajiri	Haifa.Hajiri@TechCorp.com	2003-12-17	Administrative Assistant	47418	Distribution	Allison Gentle	2003-12-17	2100-01-01	West Coast	705 Jam
2	E27621	Wendell Mobley	Wendell.Mobley@TechCorp.com	2013-11-27	Administrative Assistant	28969	Distribution	Allison Gentle	2013-11-27	2100-01-02	West Coast	705 Jam
3	E27909	Michael Sperduti	Michael.Sperduti@TechCorp.com	2014-06-20	Administrative Assistant	43778	Distribution	Allison Gentle	2014-06-20	2100-01-03	West Coast	705 Jam
4	E35053	Ashley Bergman	Ashley.Bergman@TechCorp.com	2009-03-01	Administrative Assistant	41090	Distribution	Allison Gentle	2009-03-01	2100-01-04	Midwest	1300 Ni
5	E51723	Carlos Lopez	Carlos.Lopez@TechCorp.com	2014-05-23	Administrative Assistant	35825	Distribution	Allison Gentle	2014-05-23	2100-01-05	East Coast	165 Bro
6	E59688	Jason Wingard	Jason.Wingard@TechCorp.com	2006-01-11	Administrative Assistant	34809	Distribution	Allison Gentle	2006-01-11	2100-01-06	West Coast	705 Jam
7	E18697	Anita Deluise	Anita.Deluise@TechCorp.com	1995-06-01	Administrative Assistant	45325	HQ	Tyrone Hutchison	1995-06-01	2100-01-07	HQ	1 Tech A
8	E25640	Joseph Donohue	Joseph.Donohue@TechCorp.com	2007-12-27	Administrative Assistant	40647	HQ	Tyrone Hutchison	2007-12-27	2100-01-08	HQ	1 Tech A
9	E35075	John Certa	John.Certa@TechCorp.com	2007-08-29	Administrative Assistant	51633	HQ	Tyrone Hutchison	2007-08-29	2100-01-09	HQ	1 Tech A
10	E56459	Raven Landis	Raven.Landis@TechCorp.com	2016-04-21	Administrative Assistant	42313	HQ	Tyrone Hutchison	2016-04-21	2100-01-10	HQ	1 Tech A
11	E78732	Randy Myers	Randy.Myers@TechCorp.com	2020-01-24	Administrative Assistant	34084	HQ	Tyrone Hutchison	2020-01-24	2100-01-11	HQ	1 Tech A
12	E90407	Danny Godiksen	Danny.Godiksen@TechCorp.com	2010-12-15	Administrative Assistant	28373	HQ	Tyrone Hutchison	2010-12-15	2100-01-12	HQ	1 Tech A
13	E45824	Raj Prudvi	Raj.Prudvi@TechCorp.com	2017-10-03	Administrative Assistant	51052	IT	Jacob Lauber	2017-10-03	2100-01-13	East Coast	165 Bro
14	E48148	Alexis Fitzpatrick	Alexis.Fitzpatrick@TechCorp.com	2013-02-19	Administrative Assistant	39450	IT	Jacob Lauber	2013-02-19	2100-01-14	Midwest	1300 Ni
15	E50012	Ann Roberto	Ann.Roberto@TechCorp.com	2009-10-01	Administrative Assistant	37136	IT	Jacob Lauber	2009-10-01	2100-01-15	Midwest	1300 Ni
16	E81502	Danny Laxton	Danny.Laxton@TechCorp.com	1999-10-24	Administrative Assistant	34677	IT	Jacob Lauber	1999-10-24	2100-01-16	East Coast	165 Bro
17	E83558	Laura McKenna	Laura.McKenna@TechCorp.com	1997-10-15	Administrative Assistant	45210	IT	Jacob Lauber	1997-10-15	2100-01-17	East Coast	165 Bro
18	E99949	William Graf	William.Graf@TechCorp.com	2005-05-26	Administrative Assistant	50043	IT	Jacob Lauber	2005-05-26	2100-01-18	HQ	1 Tech A
19	E12562	Keith Ingram	Keith.Ingram@TechCorp.com	1996-04-14	Administrative Assistant	48910	Product Development	Conner Kinch	1996-04-14	2100-01-19	HQ	1 Tech A

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.



```
CREATE USER NoMgr;

/*The database is calle Udacity*/
GRANT CONNECT ON DATABASE Udacity TO NoMgr;

REVOKE ALL PRIVILEGES ON public.salary FROM NoMgr;
```