Tech ABC Corp - HR Database

[Dr. Christopher O’Hara, 11/04/2025]

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 requests a secure, normalized operational HR database to replace the shared spreadsheet and provide broad read access while protecting salary data.
- **Describe current data management solution:**
  Employee records are stored in a shared Excel workbook used by multiple staff, creating data integrity and security risks.
- **Describe current data available:**
  Employee records are stored in a shared Excel workbook used by multiple staff, creating data integrity and security risks.
- **Additional data requests:**
  HR seeks future integration with payroll for attendance and PTO, and company-wide read-only visibility excluding salary.
- **Who will own/manage data**
  The HR department will own and manage the database and its governance.
- **Who will have access to database**
  All domain-authenticated employees will have read-only access excluding salary, while HR and management will have read and write access including salary.

Data Architect Business Requirement

- **Estimated size of database**
  About 600 to 800 total rows across normalized tables at go-live, assuming roughly 200 employees with one current assignment and one current salary each plus small lookups.

- **Estimated annual growth**
  Expect roughly 25 to 35 percent row growth per year driven by 20 percent headcount growth and accumulating assignment and salary history.

- **Is any of the data sensitive/restricted**
  Salary and compensation fields are restricted to HR and management, and any future personally identifiable information would be restricted as well.

Data Architect Technical Requirement

- **Justification for the new database**

1) Build a normalized HR database to enforce data integrity,
2) Role-based security, and to scale operational history while enabling future integration with payroll.

- **Database objects**

  Tables employee, organization_unit, job_title, work_location, employment_assignment, salary; views v_employee_current and v_employee_flat; stored procedure sp_get_employee_jobs(name).

- **Data ingestion**

  ETL from the Excel workbook into staging then core tables now, with a future direct feed or API to payroll after IT security review.

Data Architect Technical Requirement

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

  Ownership: The HR department will own and maintain the data, with IT administering the platform and enforcing security controls.

  User Access: All domain-authenticated employees have read-only access to non-salary data, while HR and management have read and write access including salary, and non-management roles are explicitly denied salary access.

- **Scalability**
  Replication and sharding are not required at current scale and growth, a single instance with proper indexing and periodic housekeeping is sufficient.

- **Flexibility**
  Surrogate keys, effective-dated assignments and salaries, and an integration boundary keyed by employee_id and assignment_id enable future payroll feeds or APIs without schema changes.

- **Storage & retention**

  Storage (disk or in-memory): Use spinning disk storage with the standard 1 GB partition per IT guidance, and avoid in-memory storage since advanced analytics are out of scope.

  Retention: Retain HR records for at least 7 years to meet regulatory requirements.

- **Backup**
  Classify the database as Critical and run weekly full backups with daily incremental backups per IT Best Practices.

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.

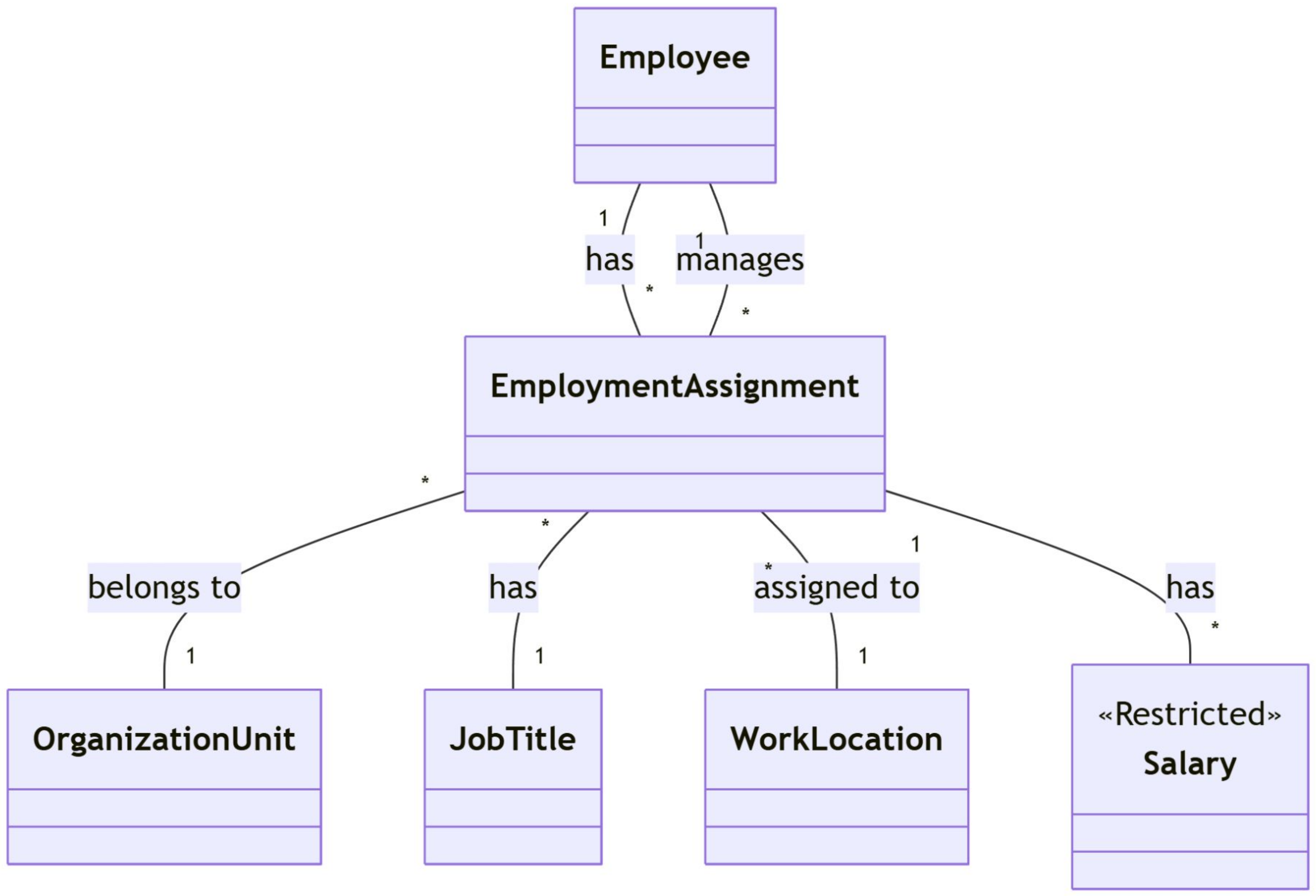
Using Lucidchart, you will create 3 entity relationship diagrams (ERDs) to show how you developed the final design for your data.

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

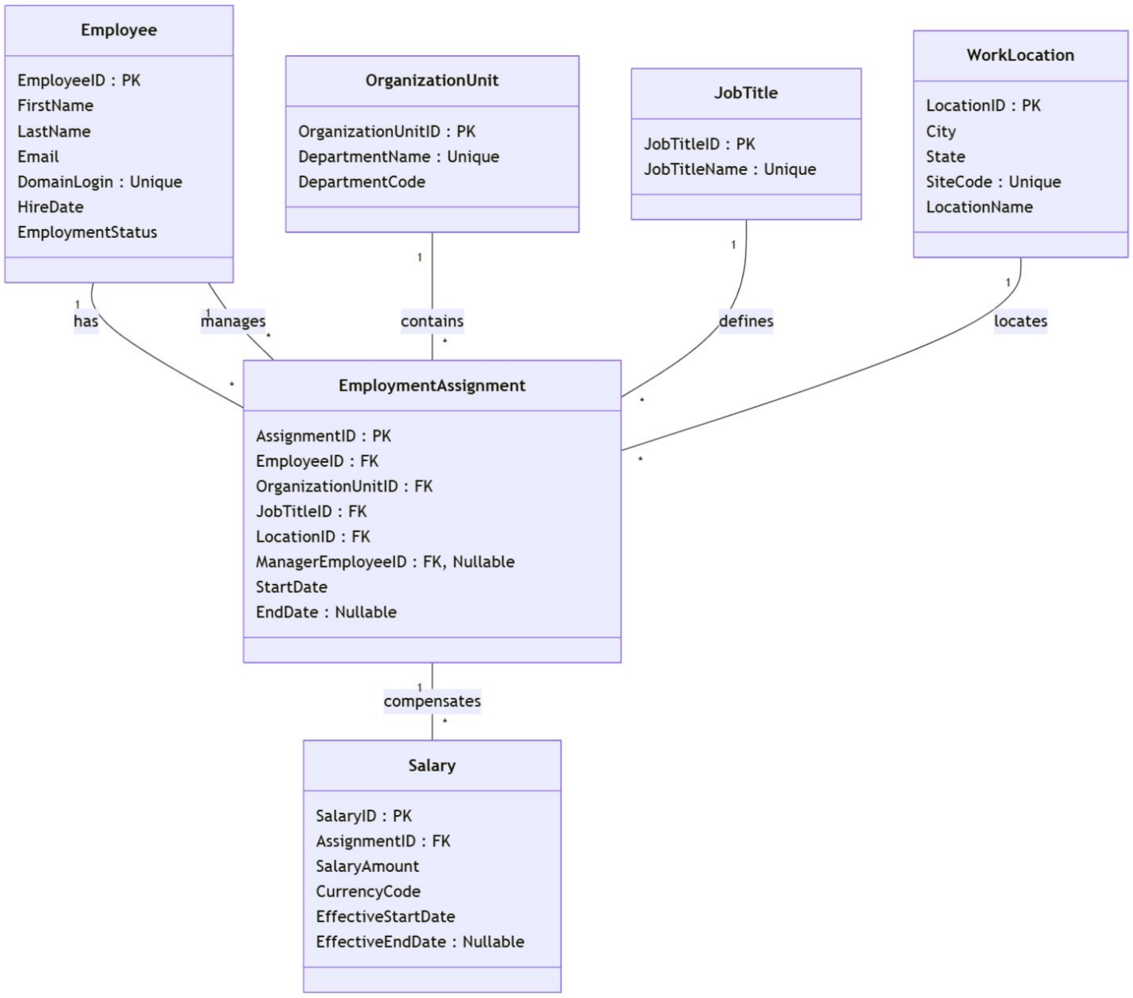
This is the most general level of data modeling. At the conceptual level, you should be thinking about creating entities that represent business objects for the database. Think broadly here. Attributes (or column names) are not required at this point, but relationship lines are required (although Crow's 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.



ERD

- **Logical**

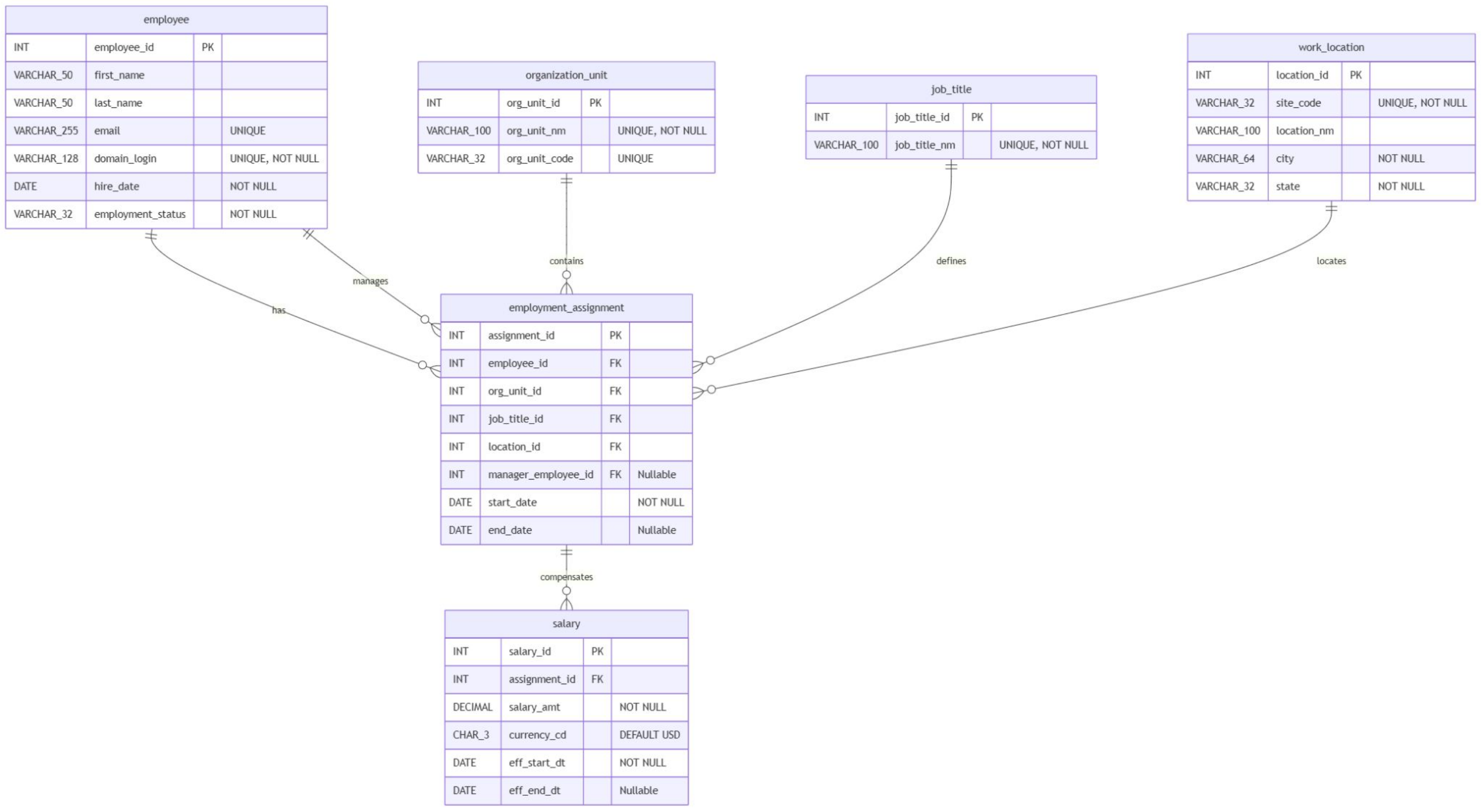
The logical model is the next level of refinement from the conceptual ERD. At this point, you should have normalized the data to the 3NF. Attributes should also be listed now in the ERD. You can still use human-friendly entity and attribute names in the logical model, and while relationship lines are 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.



employee

INT	employee_id	PK	
VARCHAR_50	first_name		
VARCHAR_50	last_name		
VARCHAR_255	email		UNIQUE
VARCHAR_128	domain_login		UNIQUE, NOT NULL
DATE	hire_date		NOT NULL
VARCHAR_32	employment_status		NOT NULL

organization_unit

INT	org_unit_id	PK	
VARCHAR_100	org_unit_nm		UNIQUE, NOT NULL
VARCHAR_32	org_unit_code		UNIQUE

job_title

INT	job_title_id	PK	
VARCHAR_100	job_title_nm		UNIQUE, NOT NULL

work_location

INT	location_id	PK	
VARCHAR_32	site_code		UNIQUE, NOT NULL
VARCHAR_100	location_nm		
VARCHAR_64	city		NOT NULL
VARCHAR_32	state		NOT NULL

employment_assignment

INT	assignment_id	PK	
INT	employee_id	FK	
INT	org_unit_id	FK	
INT	job_title_id	FK	
INT	location_id	FK	
INT	manager_employee_id	FK	Nullable
DATE	start_date		NOT NULL
DATE	end_date		Nullable

salary

INT	salary_id	PK	
INT	assignment_id	FK	
DECIMAL	salary_amt		NOT NULL
CHAR_3	currency_cd		DEFAULT USD
DATE	eff_start_dt		NOT NULL
DATE	eff_end_dt		Nullable

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.

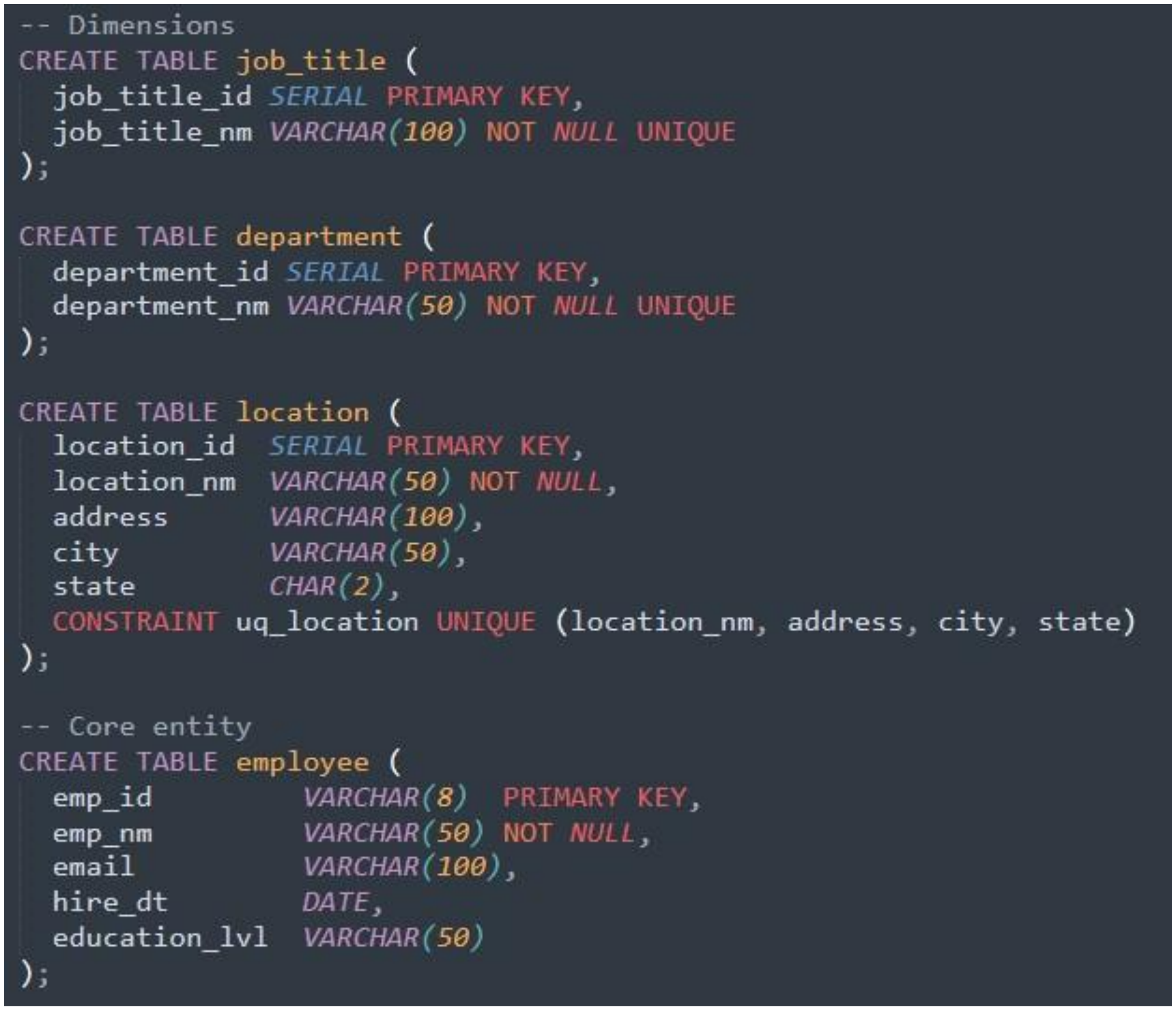
After running CRUD commands like update, insert, or delete, run a `SELECT*` command on the affected table, so the reviewer can see the results of the command.

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.

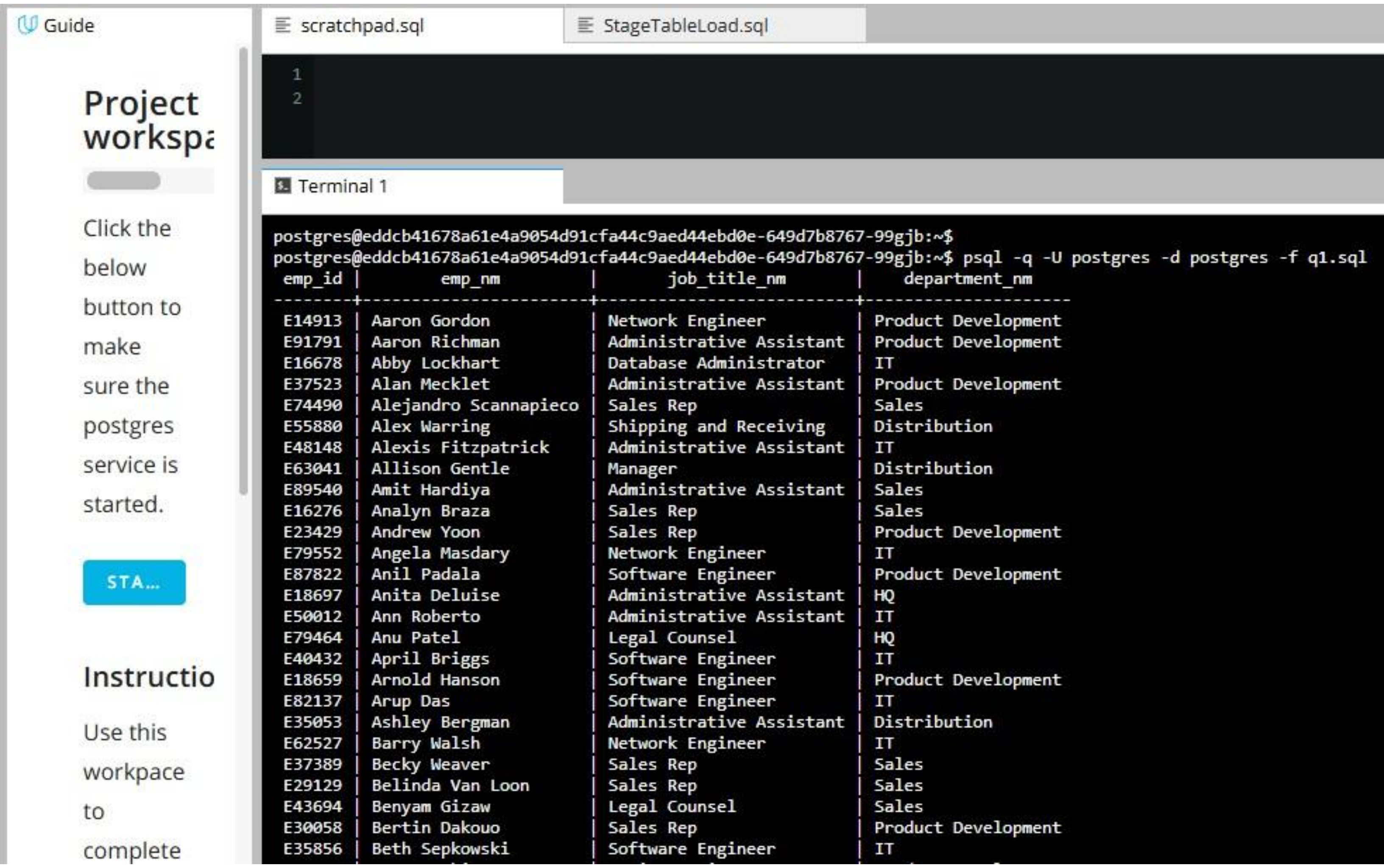
Foreign keys cannot be created on tables that do not exist yet, so it may be easier to create all tables in the database, then to go back and run modify statements on the tables to create foreign key constraints.



```
-- Dimensions
CREATE TABLE job_title (
  job_title_id SERIAL PRIMARY KEY,
  job_title_nm VARCHAR(100) NOT NULL UNIQUE
);

CREATE TABLE department (
  department_id SERIAL PRIMARY KEY,
  department_nm VARCHAR(50) NOT NULL UNIQUE
);

CREATE TABLE location (
  location_id  SERIAL PRIMARY KEY,
  location_nm  VARCHAR(50) NOT NULL,
  address      VARCHAR(100),
  city         VARCHAR(50),
  state        CHAR(2),
  CONSTRAINT uq_location UNIQUE (location_nm, address, city, state)
);

-- Core entity
CREATE TABLE employee (
  emp_id         VARCHAR(8)  PRIMARY KEY,
  emp_nm         VARCHAR(50) NOT NULL,
  email          VARCHAR(100),
  hire_dt        DATE,
  education_lvl  VARCHAR(50)
);
```

CRUD

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



Project workspa

Click the below button to make sure the postgres service is started.

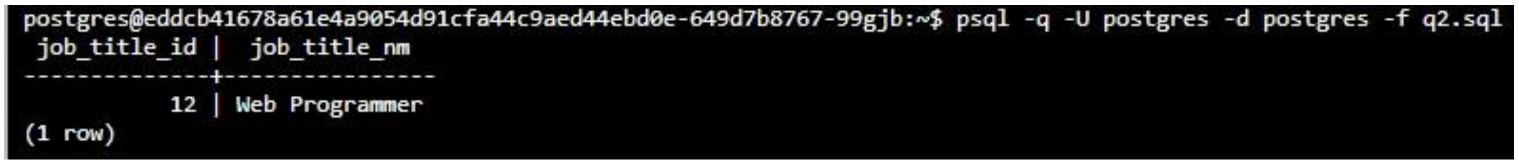
Instructio

Use this workpace to complete

```
postgres@eddcb41678a61e4a9054d91cfa44c9aed44ebd0e-649d7b8767-99gjb:~$
postgres@eddcb41678a61e4a9054d91cfa44c9aed44ebd0e-649d7b8767-99gjb:~$ psql -q -U postgres -d postgres -f q1.sql
 emp_id |        emp_nm          |       job_title_nm        |    department_nm
--------+------------------------+---------------------------+---------------------
 E14913 | Aaron Gordon           | Network Engineer          | Product Development
 E91791 | Aaron Richman          | Administrative Assistant  | Product Development
 E16678 | Abby Lockhart          | Database Administrator    | IT
 E37523 | Alan Mecklet           | Administrative Assistant  | Product Development
 E74490 | Alejandro Scannapieco  | Sales Rep                 | Sales
 E55880 | Alex Warring           | Shipping and Receiving    | Distribution
 E48148 | Alexis Fitzpatrick     | Administrative Assistant  | IT
 E63041 | Allison Gentle         | Manager                   | Distribution
 E89540 | Amit Hardiya           | Administrative Assistant  | Sales
 E16276 | Analyn Braza           | Sales Rep                 | Sales
 E23429 | Andrew Yoon            | Sales Rep                 | Product Development
 E79552 | Angela Masdary         | Network Engineer          | IT
 E87822 | Anil Padala            | Software Engineer         | Product Development
 E18697 | Anita Deluise          | Administrative Assistant  | HQ
 E50012 | Ann Roberto            | Administrative Assistant  | IT
 E79464 | Anu Patel              | Legal Counsel             | HQ
 E40432 | April Briggs           | Software Engineer         | IT
 E18659 | Arnold Hanson          | Software Engineer         | Product Development
 E82137 | Arup Das               | Software Engineer         | IT
 E35053 | Ashley Bergman         | Administrative Assistant  | Distribution
 E62527 | Barry Walsh            | Network Engineer          | IT
 E37389 | Becky Weaver           | Sales Rep                 | Sales
 E29129 | Belinda Van Loon       | Sales Rep                 | Sales
 E43694 | Benyam Gizaw           | Legal Counsel             | Sales
 E30058 | Bertin Dakouo          | Sales Rep                 | Product Development
 E35856 | Beth Sepkowski         | Software Engineer         | IT
```

CRUD

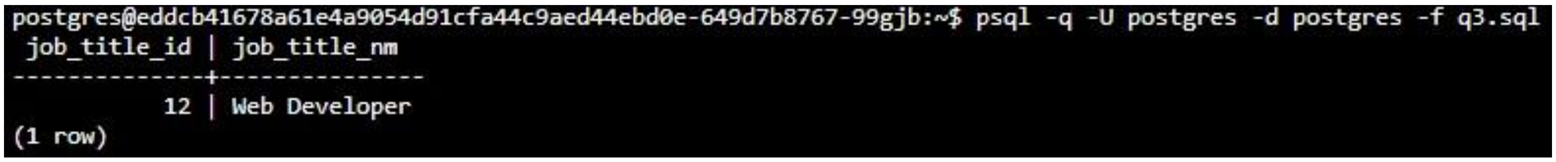
- **Question 2: Insert Web Programmer as a new job title**



```
postgres@eddcb41678a61e4a9054d91cfa44c9aed44ebd0e-649d7b8767-99gjb:~$ psql -q -U postgres -d postgres -f q2.sql
 job_title_id |  job_title_nm
--------------+----------------
           12 | Web Programmer
(1 row)
```

CRUD

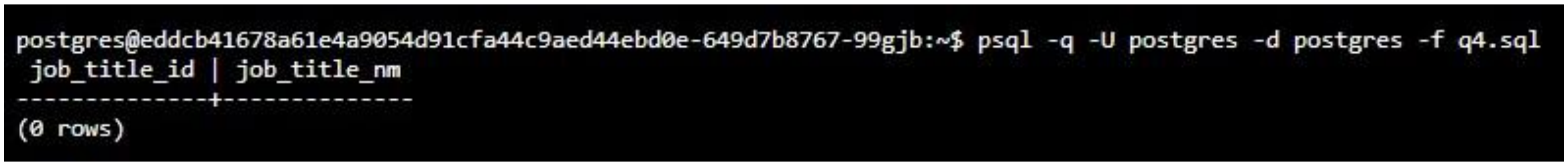
- **Question 3: Correct the job title from web programmer to web developer**



```
postgres@eddcb41678a61e4a9054d91cfa44c9aed44ebd0e-649d7b8767-99gjb:~$ psql -q -U postgres -d postgres -f q3.sql
 job_title_id | job_title_nm
--------------+---------------
           12 | Web Developer
(1 row)
```

CRUD

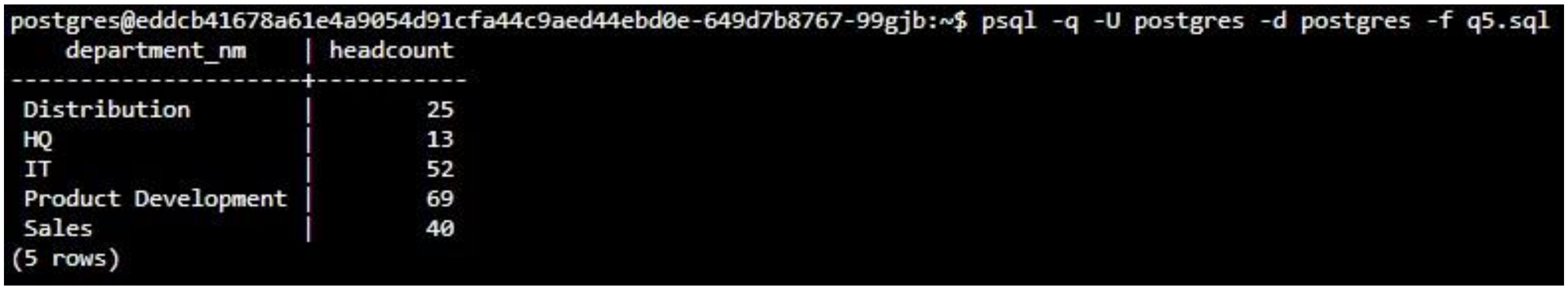
- **Question 4: Delete the job title Web Developer from the database**



```
postgres@eddcb41678a61e4a9054d91cfa44c9aed44ebd0e-649d7b8767-99gjb:~$ psql -q -U postgres -d postgres -f q4.sql
 job_title_id | job_title_nm
--------------+--------------
(0 rows)
```

CRUD

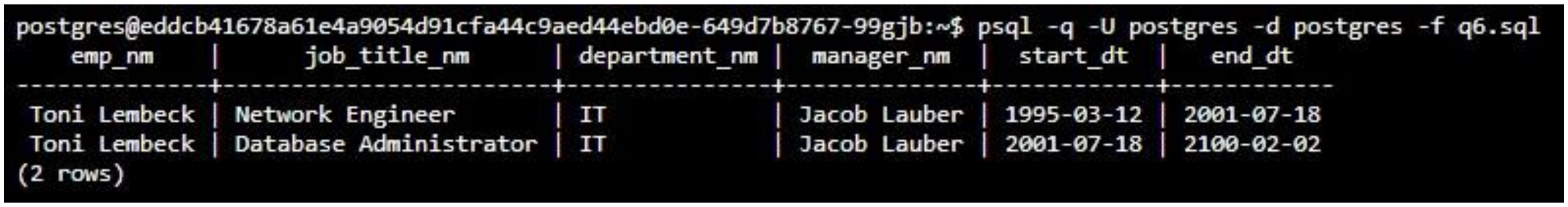
- **Question 5: How many employees are in each department?**



```
postgres@eddcb41678a61e4a9054d91cfa44c9aed44ebd0e-649d7b8767-99gjb:~$ psql -q -U postgres -d postgres -f q5.sql
    department_nm     | headcount
----------------------+-----------
 Distribution         |        25
 HQ                   |        13
 IT                   |        52
 Product Development  |        69
 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.**

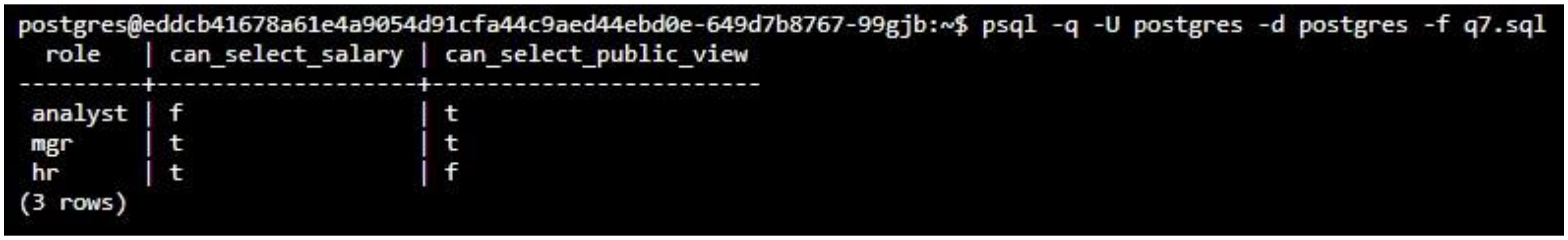


```
postgres@eddcb41678a61e4a9054d91cfa44c9aed44ebd0e-649d7b8767-99gjb:~$ psql -q -U postgres -d postgres -f q6.sql
    emp_nm    |      job_title_nm      | department_nm |  manager_nm  |  start_dt  |   end_dt
--------------+------------------------+---------------+--------------+------------+------------
 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.**

Restrict salary access by role. First revoke all default privileges from PUBLIC on the base tables, then create least-privilege roles (for example, hr_read, hr_admin, analyst). Store compensation in a separate salary table keyed by emp_id, or keep it as a column and use column-level privileges. Grant SELECT/UPDATE on the salary data only to hr_* roles and deny it to everyone else. Expose a v_employee_public view that omits salary for general users, and optionally add row-level security so managers can see only their reports. Route any changes through SECURITY DEFINER procedures and audit grants to ensure only HR can read or modify salaries.



```
postgres@eddcb41678a61e4a9054d91cfa44c9aed44ebd0e-649d7b8767-99gjb:~$ psql -q -U postgres -d postgres -f q7.sql
  role   | can_select_salary | can_select_public_view
---------+-------------------+-----------------------
 analyst | f                 | t
 mgr     | t                 | t
 hr      | t                 | f
(3 rows)
```

Step 4

Above and Beyond

(optional)

Step 4: Above and Beyond

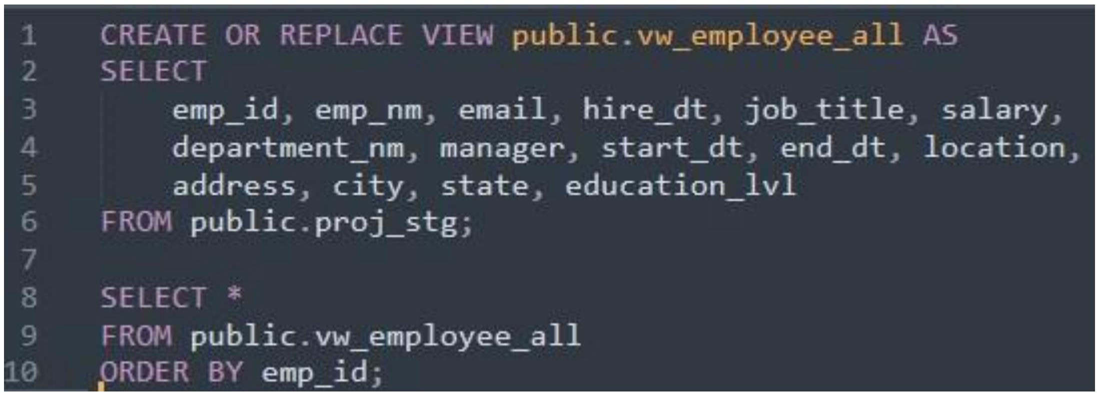
This last step is called Above and Beyond. In this step, I have proposed 3 challenges for you to complete, which are above and beyond the scope of the project. This is a chance to flex your coding muscles and show everyone how good you really are.

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.

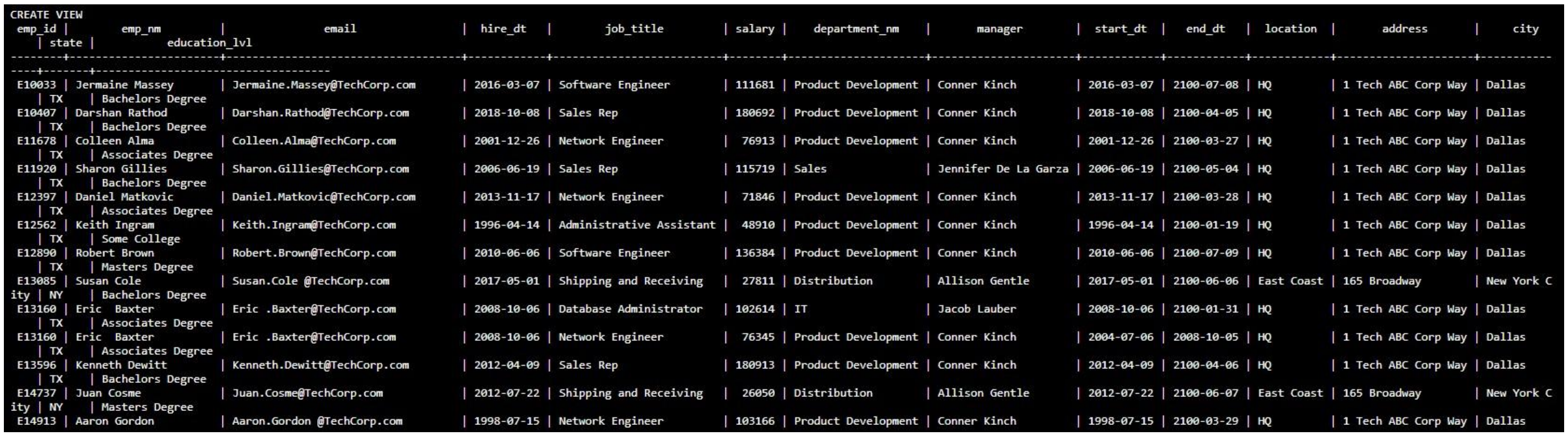
I encourage you to take on these challenges in this course and any future courses you take. I designed these challenges to be a challenge to your current abilities, but I ensured they are not an unattainable challenge. Remember, these challenges are completely optional - you can pass the project by doing 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



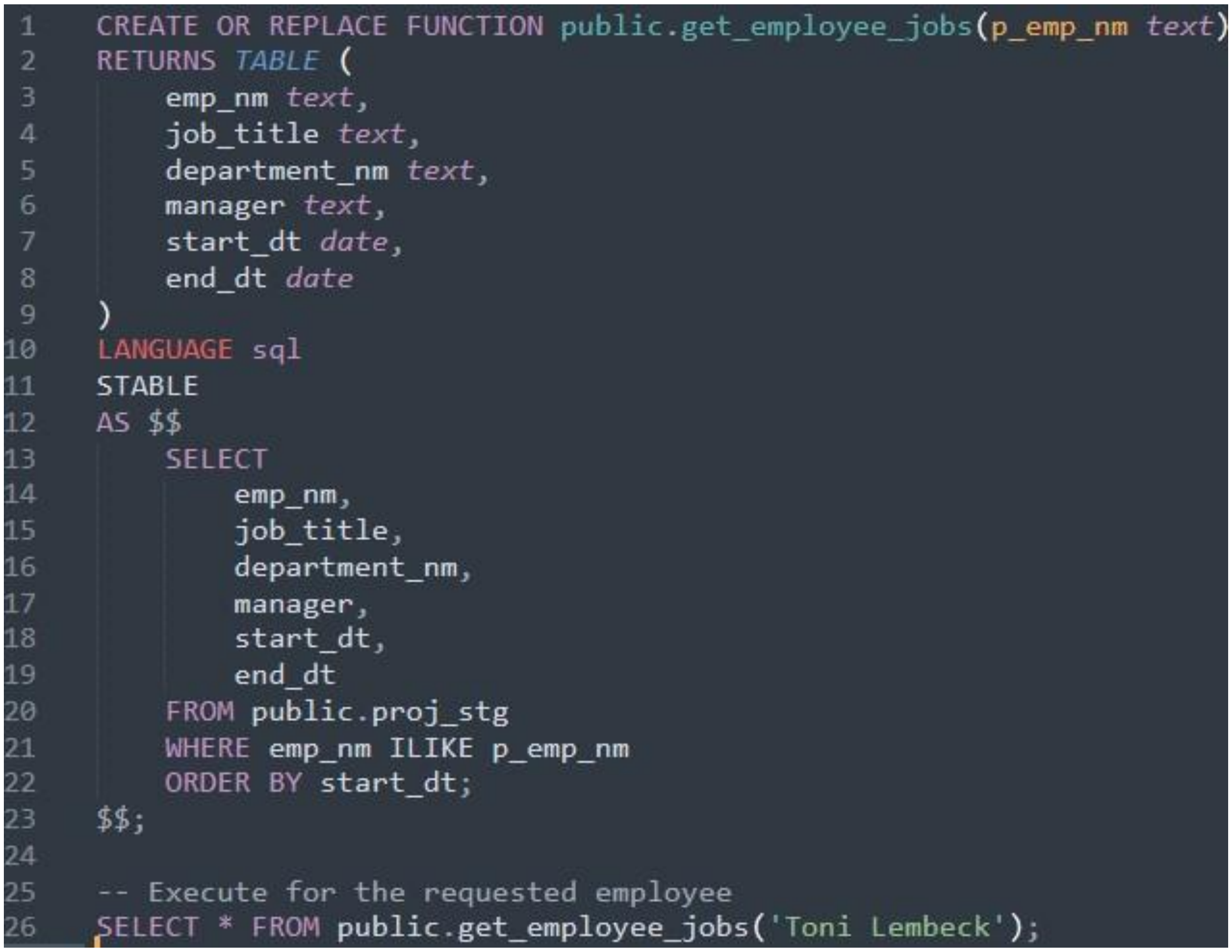
```sql
CREATE OR REPLACE VIEW public.vw_employee_all AS
SELECT
    emp_id, emp_nm, email, hire_dt, job_title, salary,
    department_nm, manager, start_dt, end_dt, location,
    address, city, state, education_lvl
FROM public.proj_stg;

SELECT *
FROM public.vw_employee_all
ORDER BY emp_id;
```



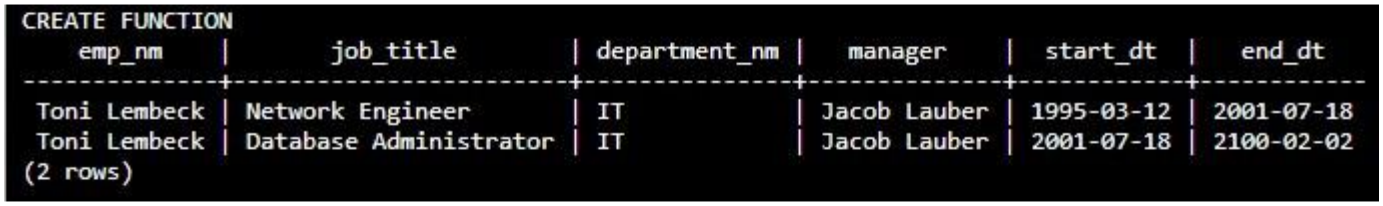
```
CREATE VIEW
 emp_id |       emp_nm         |               email                 |  hire_dt   |        job_title         | salary |    department_nm    |       manager        |  start_dt  |   end_dt   | location   |       address       |    city
    | state |          education_lvl
--------+----------------------+-------------------------------------+------------+--------------------------+--------+---------------------+----------------------+------------+------------+------------+---------------------+----------
----+-------+-----------------------------------
 E10033 | Jermaine Massey      | Jermaine.Massey@TechCorp.com        | 2016-03-07 | Software Engineer        | 111681 | Product Development | Conner Kinch         | 2016-03-07 | 2100-07-08 | HQ         | 1 Tech ABC Corp Way | Dallas
    | TX    | Bachelors Degree
 E10407 | Darshan Rathod       | Darshan.Rathod@TechCorp.com         | 2018-10-08 | Sales Rep                | 180692 | Product Development | Conner Kinch         | 2018-10-08 | 2100-04-05 | HQ         | 1 Tech ABC Corp Way | Dallas
    | TX    | Bachelors Degree
 E11678 | Colleen Alma         | Colleen.Alma@TechCorp.com           | 2001-12-26 | Network Engineer         |  76913 | Product Development | Conner Kinch         | 2001-12-26 | 2100-03-27 | HQ         | 1 Tech ABC Corp Way | Dallas
    | TX    | Associates Degree
 E11920 | Sharon Gillies       | Sharon.Gillies@TechCorp.com         | 2006-06-19 | Sales Rep                | 115719 | Sales               | Jennifer De La Garza | 2006-06-19 | 2100-05-04 | HQ         | 1 Tech ABC Corp Way | Dallas
    | TX    | Bachelors Degree
 E12397 | Daniel Matkovic      | Daniel.Matkovic@TechCorp.com        | 2013-11-17 | Network Engineer         |  71846 | Product Development | Conner Kinch         | 2013-11-17 | 2100-03-28 | HQ         | 1 Tech ABC Corp Way | Dallas
    | TX    | Associates Degree
 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 ABC Corp Way | Dallas
    | TX    | Some College
 E12890 | Robert Brown         | Robert.Brown@TechCorp.com           | 2010-06-06 | Software Engineer        | 136384 | Product Development | Conner Kinch         | 2010-06-06 | 2100-07-09 | HQ         | 1 Tech ABC Corp Way | Dallas
    | TX    | Masters Degree
 E13085 | Susan Cole           | Susan.Cole @TechCorp.com            | 2017-05-01 | Shipping and Receiving   |  27811 | Distribution        | Allison Gentle       | 2017-05-01 | 2100-06-06 | East Coast | 165 Broadway        | New York C
ity | NY    | Bachelors Degree
 E13160 | Eric  Baxter         | Eric .Baxter@TechCorp.com           | 2008-10-06 | Database Administrator   | 102614 | IT                  | Jacob Lauber         | 2008-10-06 | 2100-01-31 | HQ         | 1 Tech ABC Corp Way | Dallas
    | TX    | Associates Degree
 E13160 | Eric  Baxter         | Eric .Baxter@TechCorp.com           | 2008-10-06 | Network Engineer         |  76345 | Product Development | Conner Kinch         | 2004-07-06 | 2008-10-05 | HQ         | 1 Tech ABC Corp Way | Dallas
    | TX    | Associates Degree
 E13596 | Kenneth Dewitt       | Kenneth.Dewitt@TechCorp.com         | 2012-04-09 | Sales Rep                | 180913 | Product Development | Conner Kinch         | 2012-04-09 | 2100-04-06 | HQ         | 1 Tech ABC Corp Way | Dallas
    | TX    | Bachelors Degree
 E14737 | Juan Cosme           | Juan.Cosme@TechCorp.com             | 2012-07-22 | Shipping and Receiving   |  26050 | Distribution        | Allison Gentle       | 2012-07-22 | 2100-06-07 | East Coast | 165 Broadway        | New York C
ity | NY    | Masters Degree
 E14913 | Aaron Gordon         | Aaron.Gordon @TechCorp.com          | 1998-07-15 | Network Engineer         | 103166 | Product Development | Conner Kinch         | 1998-07-15 | 2100-03-29 | HQ         | 1 Tech ABC Corp Way | Dallas
```

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.



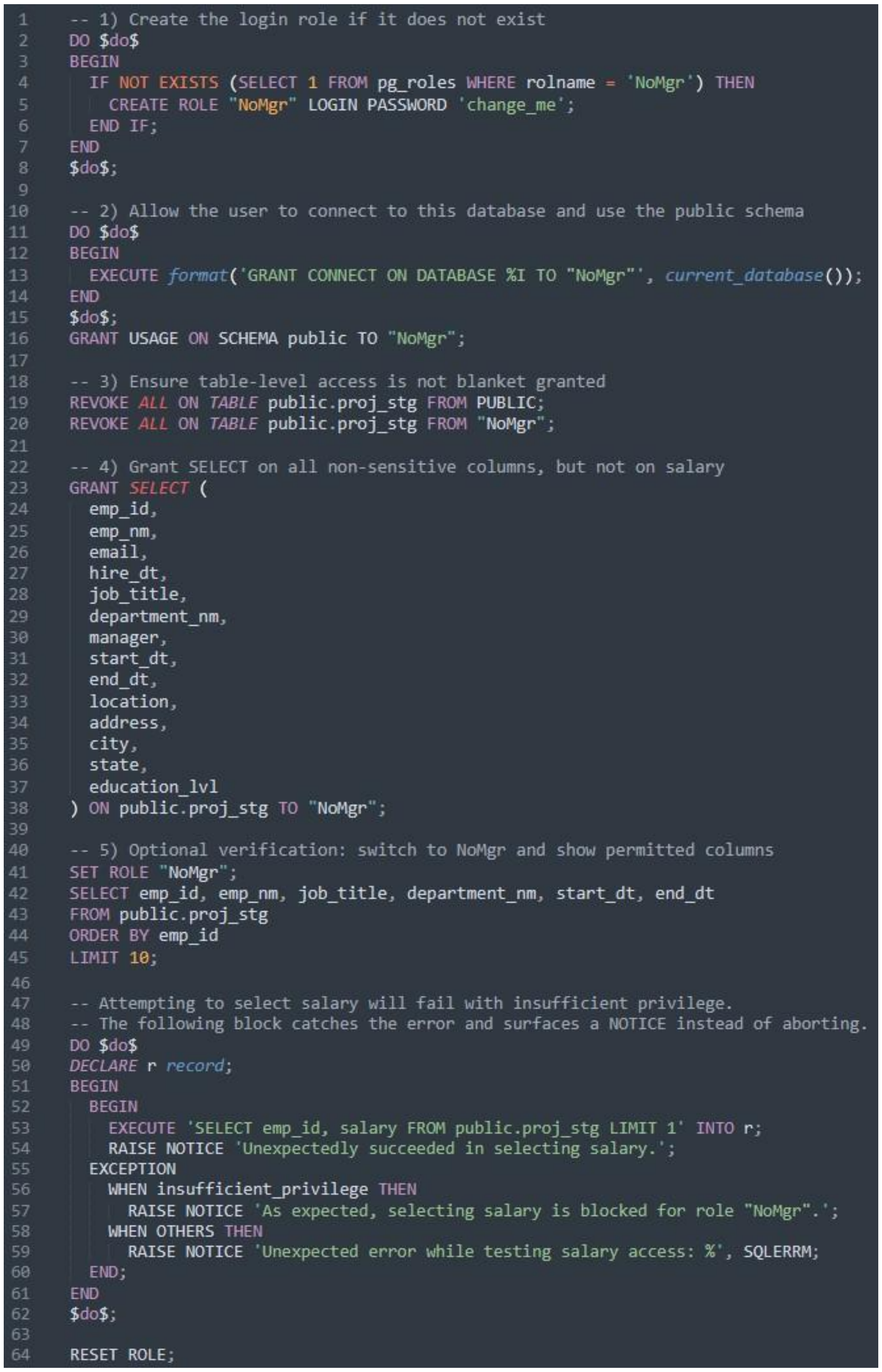
```sql
CREATE OR REPLACE FUNCTION public.get_employee_jobs(p_emp_nm text)
RETURNS TABLE (
    emp_nm text,
    job_title text,
    department_nm text,
    manager text,
    start_dt date,
    end_dt date
)
LANGUAGE sql
STABLE
AS $$
    SELECT
        emp_nm,
        job_title,
        department_nm,
        manager,
        start_dt,
        end_dt
    FROM public.proj_stg
    WHERE emp_nm ILIKE p_emp_nm
    ORDER BY start_dt;
$$;

-- Execute for the requested employee
SELECT * FROM public.get_employee_jobs('Toni Lembeck');
```



```
CREATE FUNCTION
    emp_nm    |        job_title        | department_nm |    manager    |  start_dt  |   end_dt
--------------+-------------------------+---------------+---------------+------------+------------
 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
-- 1) Create the login role if it does not exist
DO $do$
BEGIN
  IF NOT EXISTS (SELECT 1 FROM pg_roles WHERE rolname = 'NoMgr') THEN
    CREATE ROLE "NoMgr" LOGIN PASSWORD 'change_me';
  END IF;
END
$do$;

-- 2) Allow the user to connect to this database and use the public schema
DO $do$
BEGIN
  EXECUTE format('GRANT CONNECT ON DATABASE %I TO "NoMgr"', current_database());
END
$do$;
GRANT USAGE ON SCHEMA public TO "NoMgr";

-- 3) Ensure table-level access is not blanket granted
REVOKE ALL ON TABLE public.proj_stg FROM PUBLIC;
REVOKE ALL ON TABLE public.proj_stg FROM "NoMgr";

-- 4) Grant SELECT on all non-sensitive columns, but not on salary
GRANT SELECT (
  emp_id,
  emp_nm,
  email,
  hire_dt,
  job_title,
  department_nm,
  manager,
  start_dt,
  end_dt,
  location,
  address,
  city,
  state,
  education_lvl
) ON public.proj_stg TO "NoMgr";

-- 5) Optional verification: switch to NoMgr and show permitted columns
SET ROLE "NoMgr";
SELECT emp_id, emp_nm, job_title, department_nm, start_dt, end_dt
FROM public.proj_stg
ORDER BY emp_id
LIMIT 10;

-- Attempting to select salary will fail with insufficient privilege.
-- The following block catches the error and surfaces a NOTICE instead of aborting.
DO $do$
DECLARE r record;
BEGIN
  BEGIN
    EXECUTE 'SELECT emp_id, salary FROM public.proj_stg LIMIT 1' INTO r;
    RAISE NOTICE 'Unexpectedly succeeded in selecting salary.';
  EXCEPTION
    WHEN insufficient_privilege THEN
      RAISE NOTICE 'As expected, selecting salary is blocked for role "NoMgr".';
    WHEN OTHERS THEN
      RAISE NOTICE 'Unexpected error while testing salary access: %', SQLERRM;
  END;
END
$do$;

RESET ROLE;
```

Appendix

V&V cmds

```
psql -U postgres -d postgres -f ddl.sql

psql -q -U postgres -d postgres -f q1.sql
psql -q -U postgres -d postgres -f q2.sql
psql -q -U postgres -d postgres -f q3.sql
psql -q -U postgres -d postgres -f q4.sql
psql -q -U postgres -d postgres -f q5.sql
psql -q -U postgres -d postgres -f q6.sql
psql -q -U postgres -d postgres -f q7.sql

psql -q -U postgres -d postgres -f b1.sql
psql -q -U postgres -d postgres -f b2.sql
psql -q -U postgres -d postgres -f b3.sql
```