Tech ABC Corp - HR Database

[Sohaib Zafar & 04.09.2022]

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:**

  To have some data integrity and data security of employee data.

- **Describe current data management solution:**

  Currently the data management is done using Excel spreadsheet.

- **Describe current data available:**

  All the employee related information like, name, email adress, education level, job title, salary, etc.

- **Additional data requests:**

  They want the data to be maintained for atleast 7 years. In future, they would also like to combine it with Payroll department's system.

- **Who will own/manage data**

  The management and the HR employees.

- **Who will have access to database**

  Any employee with domain login will have read access to the database but they will not have access to view salary information. The management and HR employee will have read and write access and will also be able to view salary information.

Data Architect Business Requirement

- **Estimated size of database**

  205 rows and 15 columns.

- **Estimated annual growth**

  20% annual growth per year for the next 5 years.

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

  Salary data is restricted for all employees who are not managers or from HR.

Data Architect Technical Requirement

- **Justification for the new database**

  1. Data integrity, 2. Data security

- **Database objects**

  List the database objects (tables, views, special procedures) that will be created for the database.

  - Tables: Education, Employee, Job, Employment, Department, Location, Salary
  - Views: original_view
  - Function: employee_job_history()

- **Data ingestion**

  ETL

Data Architect Technical Requirement

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

  Ownership: HR employees.
  User Access: every emplyee with domain logic will have read access to data except for the salary (which is restricted to management and HR).

- **Scalability**

  Replication.

- **Flexibility**

  A direct feed could be useful in order to integrate the employee database with the payroll department‘s system

- **Storage & retention**

  Storage (disk or in-memory): disk

  Retention: 7 years.

- **Backup**

  For critical data, it is recommended to have a full back-up once a week, and 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.

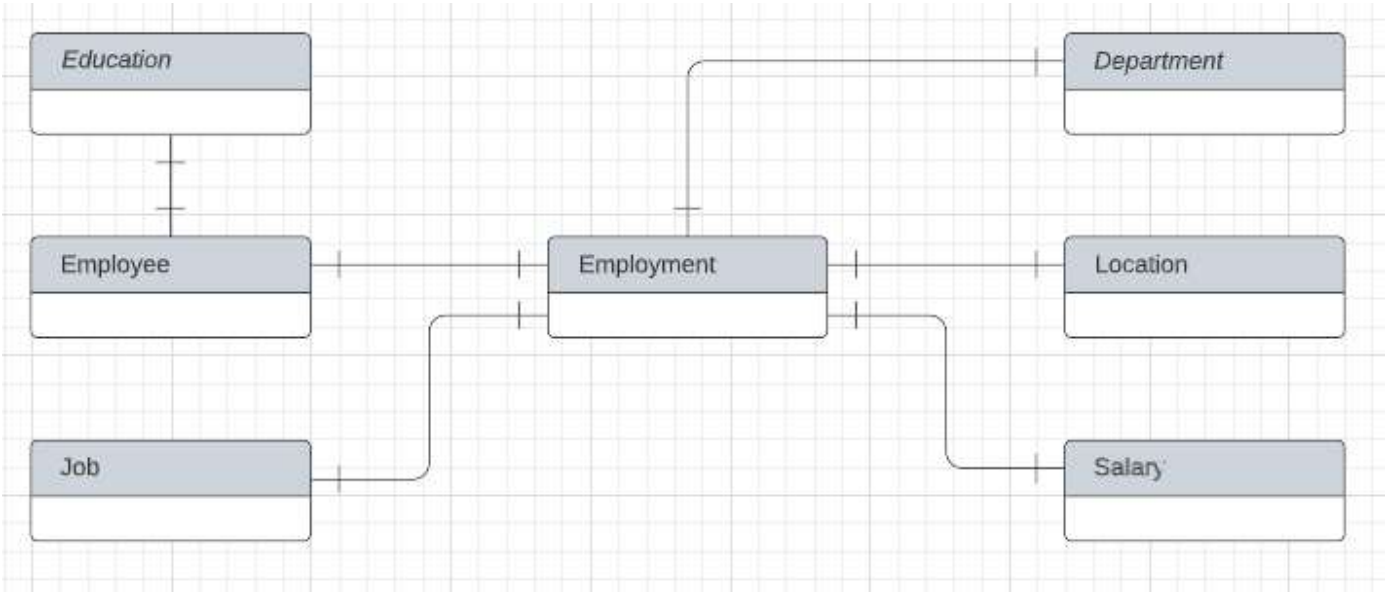
Using Lucidchart, you will create 3 entity relationship diagrams (ERDs) to show how you developed the final design for your data.

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

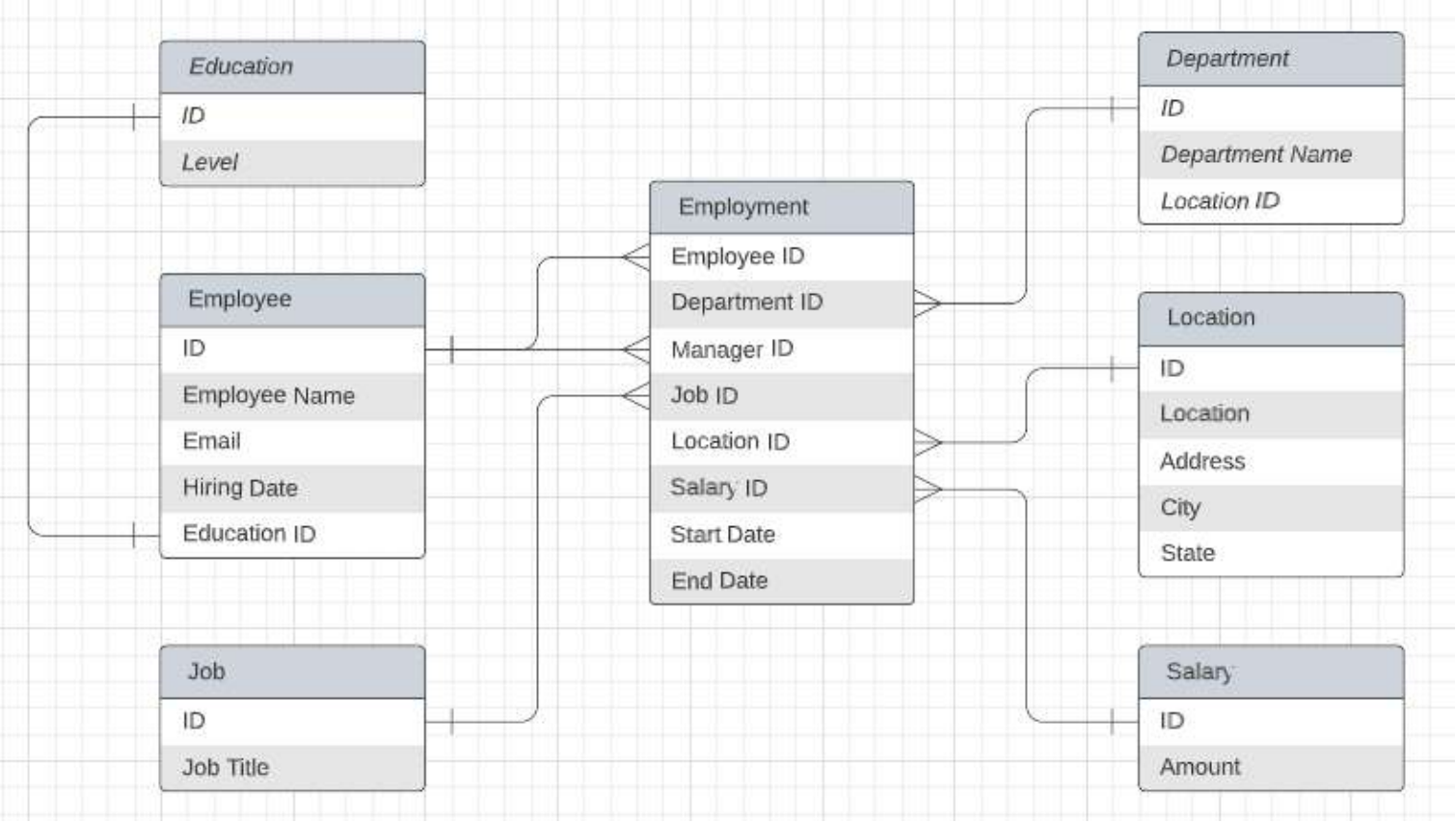
- **Conceptual**

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.



ERD

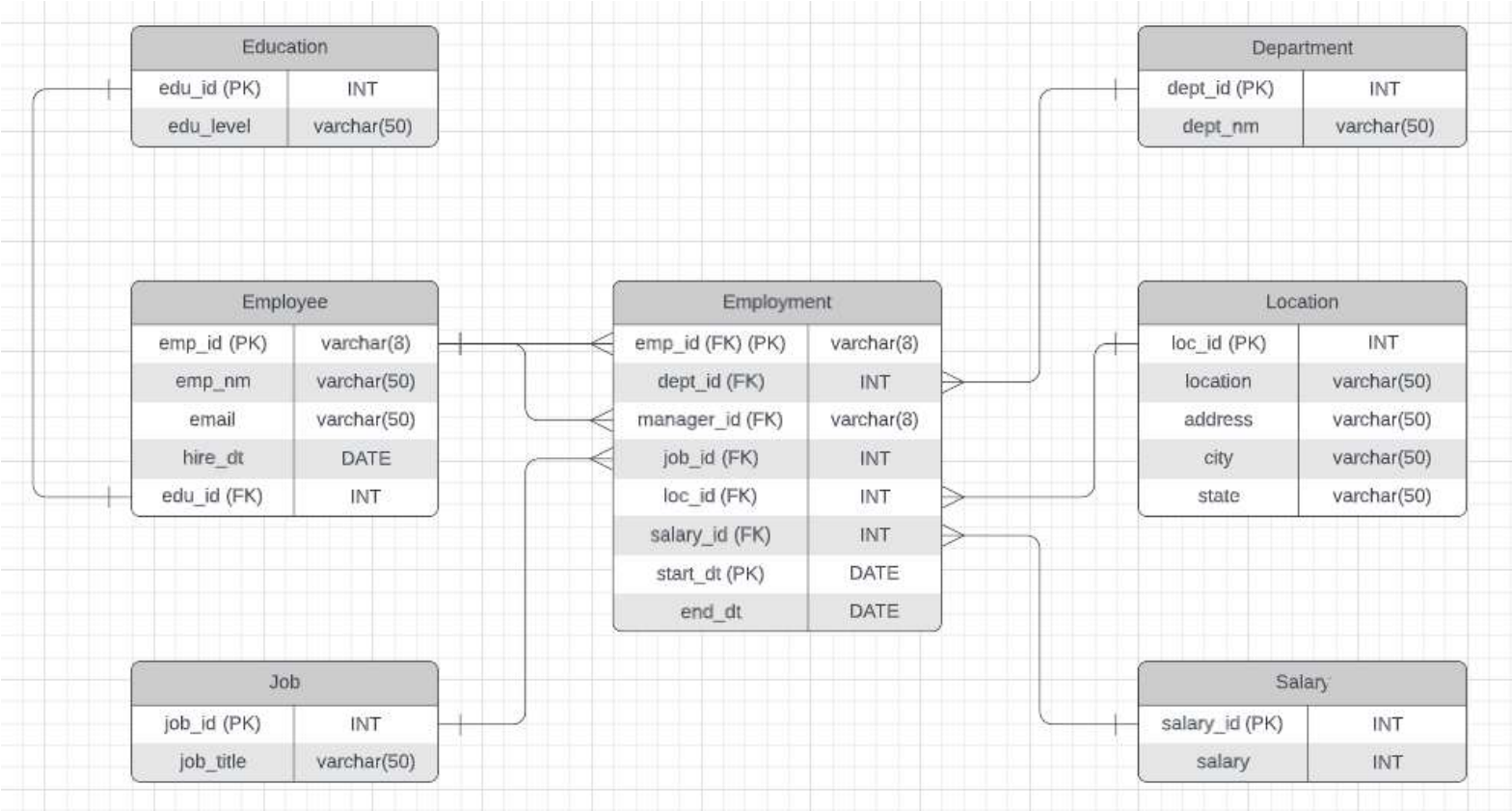
- **Logical**


Education
ID
Level
Department
ID
Department Name
Location ID
Employment
Employee ID
Department ID
Manager ID
Job ID
Location ID
Salary ID
Start Date
End Date
Employee
ID
Employee Name
Email
Hiring Date
Education ID
Location
ID
Location
Address
City
State
Job
ID
Job Title
Salary
ID
Amount

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.



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

```
CREATE TABLE Education(
   edu_id SERIAL PRIMARY KEY,
   edu_level VARCHAR(50)
);

CREATE TABLE Employee(
   emp_id VARCHAR(8) PRIMARY KEY,
   emp_nm VARCHAR(50),
   email VARCHAR(50),
   hire_dt DATE,
   edu_id INT REFERENCES Education(edu_id)
);

CREATE TABLE Job(
   job_id SERIAL PRIMARY KEY,
   job_title VARCHAR(50)
);

CREATE TABLE Location(
   loc_id SERIAL PRIMARY KEY,
   location VARCHAR(50),
   address VARCHAR(50),
   city VARCHAR(50),
   state VARCHAR(50)
);
```

DDL

```
CREATE TABLE Location(
   loc_id SERIAL PRIMARY KEY,
   location VARCHAR(50),
   address VARCHAR(50),
   city VARCHAR(50),
   state VARCHAR(50)
);

CREATE TABLE Department(
   dept_id SERIAL PRIMARY KEY,
   dept_nm VARCHAR(50),
   loc_id INT REFERENCES Location(loc_id)
);

CREATE TABLE Salary(
   salary_id SERIAL PRIMARY KEY,
   salary INT
);

CREATE TABLE Employment(
   emp_id VARCHAR(8) REFERENCES Employee(emp_id),
   dept_id INT REFERENCES Department(dept_id),
   manager_id VARCHAR(8) REFERENCES Employee(emp_id),
   job_id INT REFERENCES Job(job_id),
   salary_id INT REFERENCES Salary(salary_id),
   start_dt DATE,
   end_dt DATE,
   PRIMARY KEY (emp_id, start_dt)
);
```

DML

Queries to insert the data in the database from staging table

```
INSERT INTO Education(edu_level)
Select distinct(education_lvl) from proj_stg;


INSERT INTO Employee(emp_id, emp_nm, email, hire_dt, edu_id)
select distinct ps.emp_id, ps.emp_nm, ps.email, ps.hire_dt, ed.edu_id from proj_stg as ps
inner join Education ed on ed.edu_level=ps.education_lvl;


INSERT INTO Job(job_title)
select distinct job_title from proj_stg;


INSERT INTO Location(location, address, city, state)
select distinct location, address, city, state from proj_stg;


INSERT INTO Department(dept_nm)
select distinct department_nm from proj_stg;


INSERT INTO Salary(salary)
Select distinct(salary) from proj_stg;


INSERT INTO Employment(emp_id, dept_id, manager_id, job_id, loc_id, salary_id, start_dt, end_dt)
select distinct ps.emp_id, de.dept_id, mn.emp_id, job.job_id, loc.loc_id, sa.salary_id, ps.start_dt, ps.end_dt
from proj_stg as ps
full outer join Employee mn on mn.emp_nm=ps.manager
full outer join Employee em on em.emp_id=ps.emp_id
full outer join Department de on de.dept_nm=ps.department_nm
full outer join Job job on job.job_title=ps.job_title
full outer join Location loc on loc.location=ps.location
inner join Salary sa on sa.salary=ps.salary;
```

CRUD

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

```
postgres=# select em.emp_nm, dp.dept_nm, job.job_title from employment emp
postgres-# inner join Employee em on em.emp_id=emp.emp_id
postgres-# inner join Department dp on dp.dept_id=emp.dept_id
postgres-# inner join Job job on job.job_id=emp.job_id;
        emp_nm         |       dept_nm       |         job_title
-----------------------+---------------------+---------------------------
 Jermaine Massey       | Product Development | Software Engineer
 Darshan Rathod        | Product Development | Sales Rep
 Colleen Alma          | Product Development | Network Engineer
 Sharon Gillies        | Sales               | Sales Rep
 Daniel Matkovic       | Product Development | Network Engineer
 Keith Ingram          | Product Development | Administrative Assistant
 Robert Brown          | Product Development | Software Engineer
 Susan Cole            | Distribution        | Shipping and Receiving
 Eric  Baxter          | Product Development | Network Engineer
 Eric  Baxter          | IT                  | Database Administrator
 Kenneth Dewitt        | Product Development | Sales Rep
 Juan Cosme            | Distribution        | Shipping and Receiving
 Aaron Gordon          | Product Development | Network Engineer
 Shanteel Jackson      | Distribution        | Shipping and Receiving
 Melinda Fisher        | Distribution        | Shipping and Receiving
 Melinda Fisher        | IT                  | Software Engineer
 Analyn Braza          | Sales               | Sales Rep
 Jill Fram             | Product Development | Administrative Assistant
 Abby Lockhart         | IT                  | Database Administrator
 Abby Lockhart         | IT                  | Network Engineer
--More--
```

CRUD

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

```
postgres=# INSERT INTO Job(job_title) VALUES('Web Programmer');
INSERT 0 1
postgres=# select * from Job;
 job_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**

```
postgres=# UPDATE Job SET job_title='Web Developer' where LOWER(job_title)='web programmer'
postgres-# ;
UPDATE 1
postgres=# select * from Job;
 job_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**

```
postgres=# DELETE FROM JOB where job_title='Web Developer';
DELETE 1
postgres=# select * from Job;
 job_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?**

```
postgres=# SELECT dp.dept_nm, count(emp.emp_id) from Employment emp
postgres-# inner join Department dp on dp.dept_id=emp.dept_id
postgres-# GROUP BY dp.dept_nm;
       dept_nm       | count
---------------------+-------
 IT                  |    54
 Product Development |    70
 HQ                  |    13
 Distribution        |    27
 Sales               |    41
(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=# select distinct em.emp_nm as Employee, j.job_title,
postgres-#                 d.dept_nm,  mn.emp_nm as Manager,
postgres-#                 emp.start_dt, emp.end_dt from Employment as emp
postgres-# join Employee mn on mn.emp_id=emp.emp_id
postgres-# join Employee em on em.emp_id=emp.emp_id
postgres-# join Department d on d.dept_id=emp.dept_id
postgres-# join Job j on j.job_id=emp.job_id
postgres-# where em.emp_nm='Toni Lembeck';
   employee   |       job_title        | dept_nm |   manager    |  start_dt  |   end_dt
--------------+------------------------+---------+--------------+------------+------------
 Toni Lembeck | Database Administrator | IT      | Toni Lembeck | 2001-07-18 | 2100-02-02
 Toni Lembeck | Network Engineer       | IT      | Toni Lembeck | 1995-03-12 | 2001-07-18
(2 rows)
```

CRUD

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

  I would create user groups based on their role and grant permission to the “Salary” table only to the users who belong to management or HR role.

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.

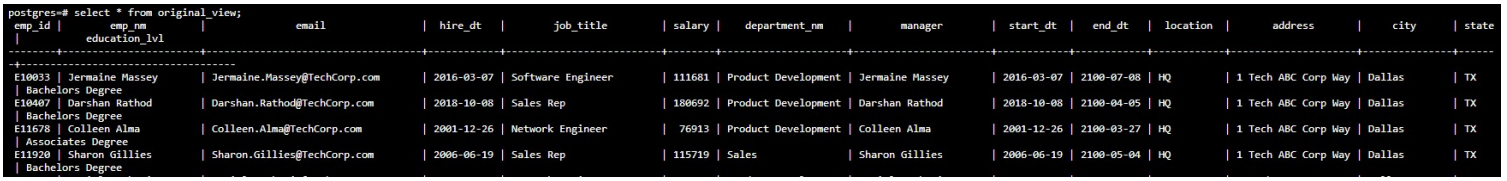
These challenge steps will bring your project even more in line with a real-world project, as these are the kind of "finishing touches" that will make your database more usable. Imagine building a car without air conditioning or turn signals. Sure, it will work, but who would want to drive it.

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

```
postgres=# CREATE VIEW original_view AS
postgres-# SELECT em.Emp_ID, em.Emp_NM, em.email, em.hire_dt,
postgres-#     j.job_title, sa.salary, d.dept_nm as department_nm, mn.emp_nm as manager,
postgres-#     emp.start_dt, emp.end_dt,
postgres-#     loc.location, loc.address, loc.city, loc.state, ed.edu_level as education_lvl
postgres-#     FROM Employment emp
postgres-# JOIN Employee mn on mn.emp_id=emp.emp_id
postgres-# JOIN Employee em on em.emp_id=emp.emp_id
postgres-# JOIN Department d on d.dept_id=emp.dept_id
postgres-# JOIN Job j on j.job_id=emp.job_id
postgres-# JOIN Location loc on loc.loc_id=emp.loc_id
postgres-# JOIN Salary sa on sa.salary_id=emp.salary_id
postgres-# JOIN Education ed on ed.edu_id=em.edu_id;
CREATE VIEW
```



```
postgres=# select * from original_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 | Jermaine Massey     | 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 | Darshan Rathod      | 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 | Colleen Alma        | 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               | Sharon Gillies      | 2006-06-19 | 2100-05-04 | HQ       | 1 Tech ABC Corp Way | Dallas       | TX
 | Bachelors Degree
```

Standout Suggestion 2

Create a stored procedure/function 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.

```
postgres=# CREATE OR REPLACE FUNCTION employee_job_history(Employee_name text)
postgres-# RETURNS TABLE (emp_nm text, job_title text, department_nm text, manager text,
postgres(#                start_dt date, end_dt date)
postgres-# AS
postgres-# $$
postgres$#     SELECT emp_nm, job_title, department_nm, manager, start_dt, end_dt FROM original_view
postgres$#     WHERE emp_nm = Employee_name;
postgres$# $$
postgres-# LANGUAGE SQL;
CREATE FUNCTION
```

```
postgres=# select * from employee_job_history('Toni Lembeck');
    emp_nm    |       job_title        | department_nm |   manager    |  start_dt  |   end_dt
--------------+------------------------+---------------+--------------+------------+------------
 Toni Lembeck | Database Administrator | IT            | Toni Lembeck | 2001-07-18 | 2100-02-02
 Toni Lembeck | Network Engineer       | IT            | Toni Lembeck | 1995-03-12 | 2001-07-18
(2 rows)
```

Standout Suggestion 3

Implement user security on the restricted salary attribute.

```
postgres=# CREATE USER NoMgr;
CREATE ROLE
```

```
postgres=# GRANT SELECT ON Employee TO NoMgr;
GRANT
postgres=# GRANT SELECT ON Education TO NoMgr;
GRANT
postgres=# GRANT SELECT ON Job TO NoMgr;
GRANT
postgres=# GRANT SELECT ON Department TO NoMgr;
GRANT
postgres=# GRANT SELECT ON Location TO NoMgr;
GRANT
postgres=# GRANT SELECT ON Employment TO NoMgr;
GRANT
postgres=#
postgres=# REVOKE SELECT ON original_view FROM NoMgr;
REVOKE
postgres=# REVOKE SELECT ON Salary FROM NoMgr;
REVOKE
```

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.