Tech ABC Corp - HR Database

[Binay Shah]

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:**

  Tech ABC Corp saw explosive growth with a sudden appearance onto the gaming scene with its 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 become increasingly cumbersome to manage as the company expands.

- **Describe current data management solution:**

  Based on the request, current data is stored on the excel sheet

- **Describe current data available:**

The current is available in an excel workbook consisting 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.

Data Architect Business Requirement

- **Additional data requests:**

  Connect with the payroll department's system in the future.

- **Who will own/manage data**

  The HR department the owner of the data

- **Who will have access to database**
  Any employee with a domain login wil have read only access to the database but restricted to salary information. That need will be restricted to HR and management level employees only. HR and management level employees should also be the only ones to with write access.

- **Estimated size of database**

  Currently the data has 205 rows.

- **Estimated annual growth**

  The company has grown from 10 employees to 200 in only 6 months and has projected 20% growth a year for next 5 years.

Data Architect Business Requirement

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

  Since only HR and management level employees will have access to salary, salary is restricted.

Data Architect Technical Requirement

- **Justification for the new database**

  1.Explosive growth of the company. It has become increasingly cumbersome to manage as company expands

  2.Data integrity and security issues in shared excel sheet.

- **Database objects**

  - Emp
  - Dept
  - Education_level
  - Job
  - Location
  - State
  - City
  - Address
  - Dept_mgr
  - Job_history
  - HR_dateset
  - Salary

- **Data ingestion**

  ETL. Future data input method would need an API.

Data Architect Technical Requirement

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

  Ownership: HR

  User Access: Any employee with domain to have read only access but will not have access to salary information. HR and managemnt level employees will have read and write access

- **Scalability**

  By our current estimates, 90% of users will be read only. So, replication should be used to ensure scalability.

- **Flexibility**

  In order to connect with the payroll department's system in the future, i'll use json columns to store extra information as needed.

- **Storage & retention**

  Storage (disk or in-memory): Databases are stored on spinning disk by default. In-memory storage is available, but only for data that

  requires higher level computations (advanced analytics, machine learning applications).

  Retention: Data should be retained for at least 7 year

Data Architect Technical Requirement

- **Backup**

This data is business critical, so we should go for critical backup plan. Its backup schedule is full backup 1x per week, incremental backup daily.

Step 2

Relational Database Design

Step 2: Relational Database Design

This step is where you will go through the process of designing a new database for Tech ABC Corp's HR department. Using the dataset provided, along with the requirements gathered in step one, you are going to develop a relational database set to the 3NF.

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

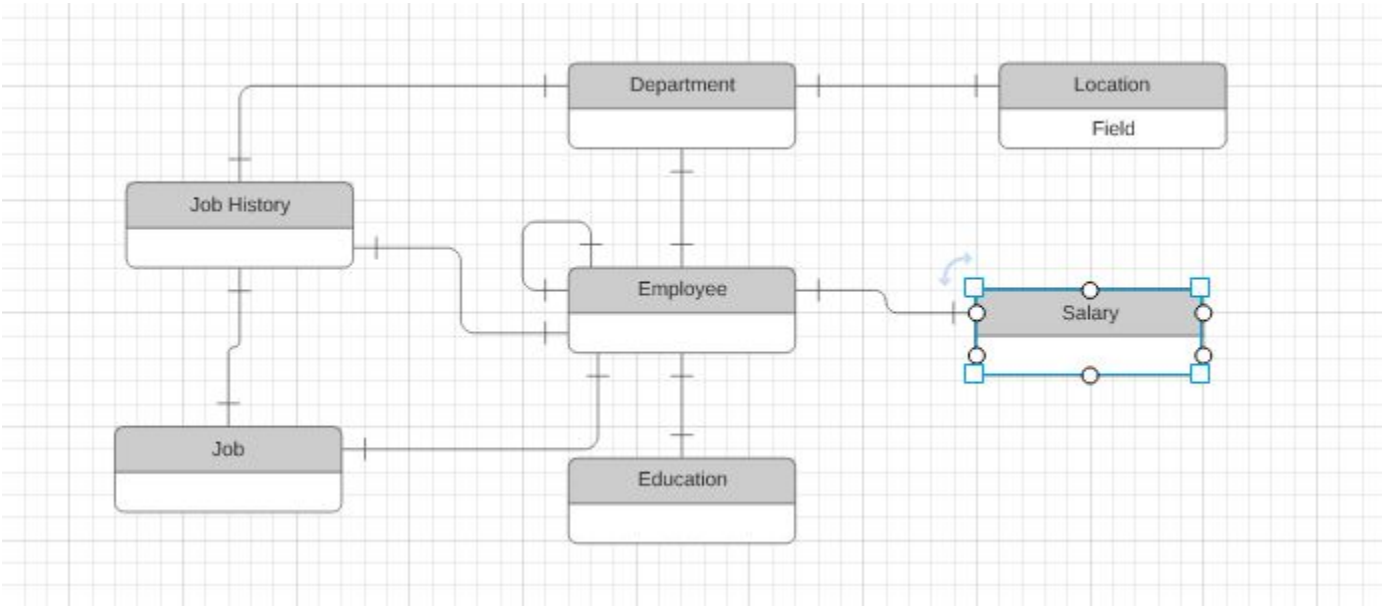
You will submit a screenshot for each of the 3 ERDs you create. You will find detailed instructions for developing each of the ERDs over the next several pages.

ERD

- **Conceptual**

This is the most general level of data modeling. At the conceptual level, you should be thinking about creating entities that represent business objects for the database. Think broadly here. Attributes (or column names) are not required at this point, but relationship lines are required (although Crow's foot notation is not needed at this level). Create at least three entities for this model; thinking about the 3NF will aid you in deciding the type of entities to create.

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

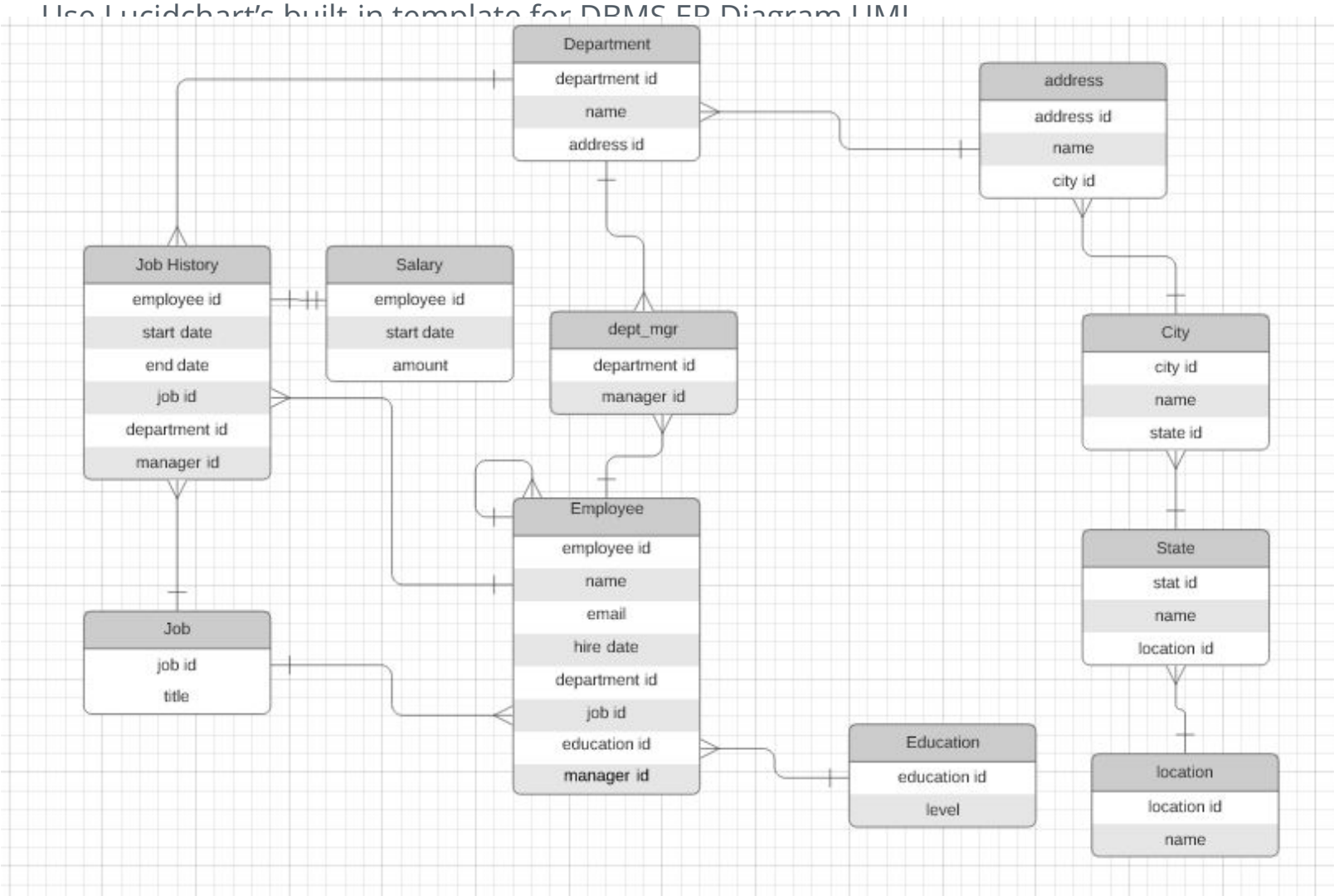


ERD

- **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.

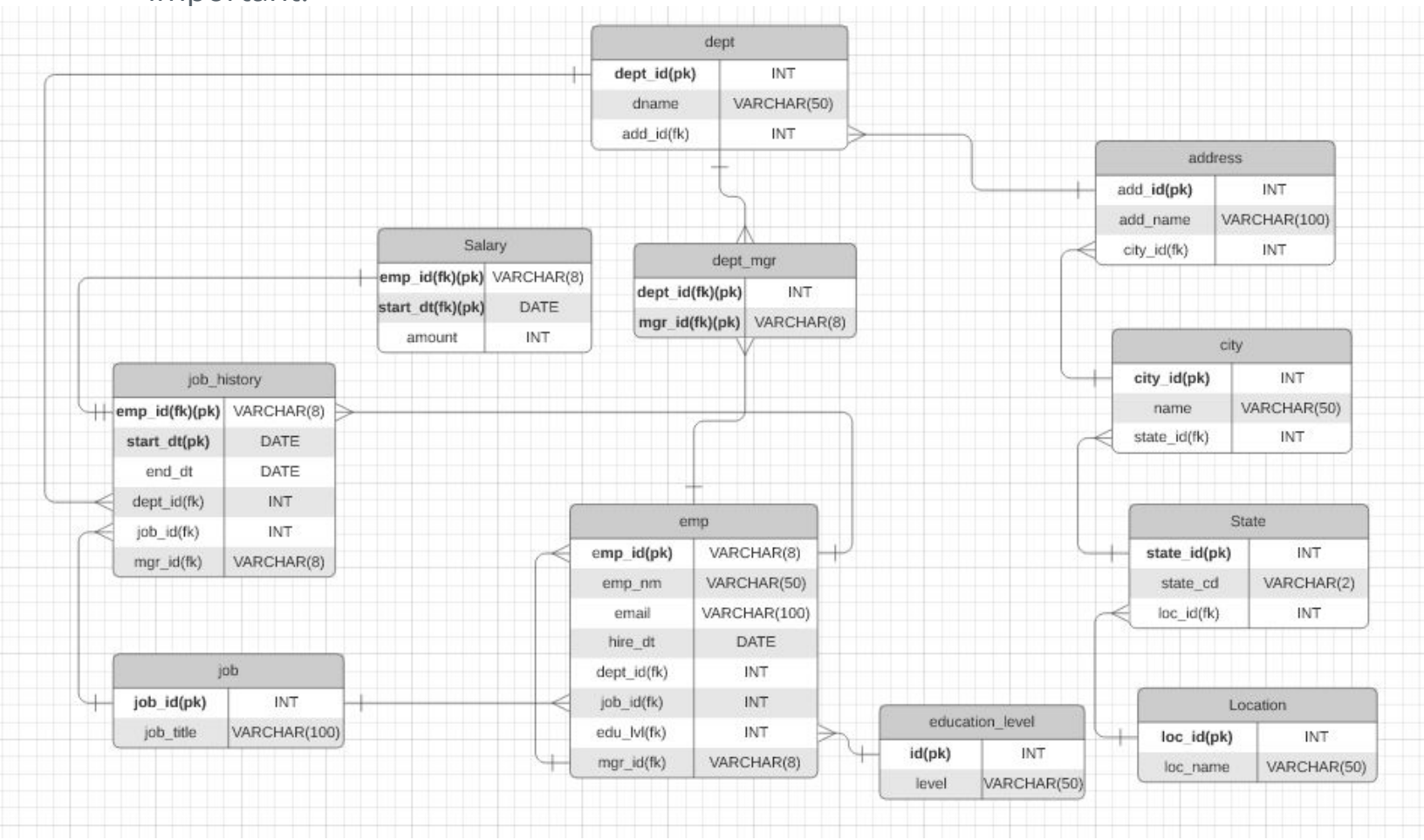
Use Lucidchart's built in template for DBMS ER Diagram UML



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.



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.

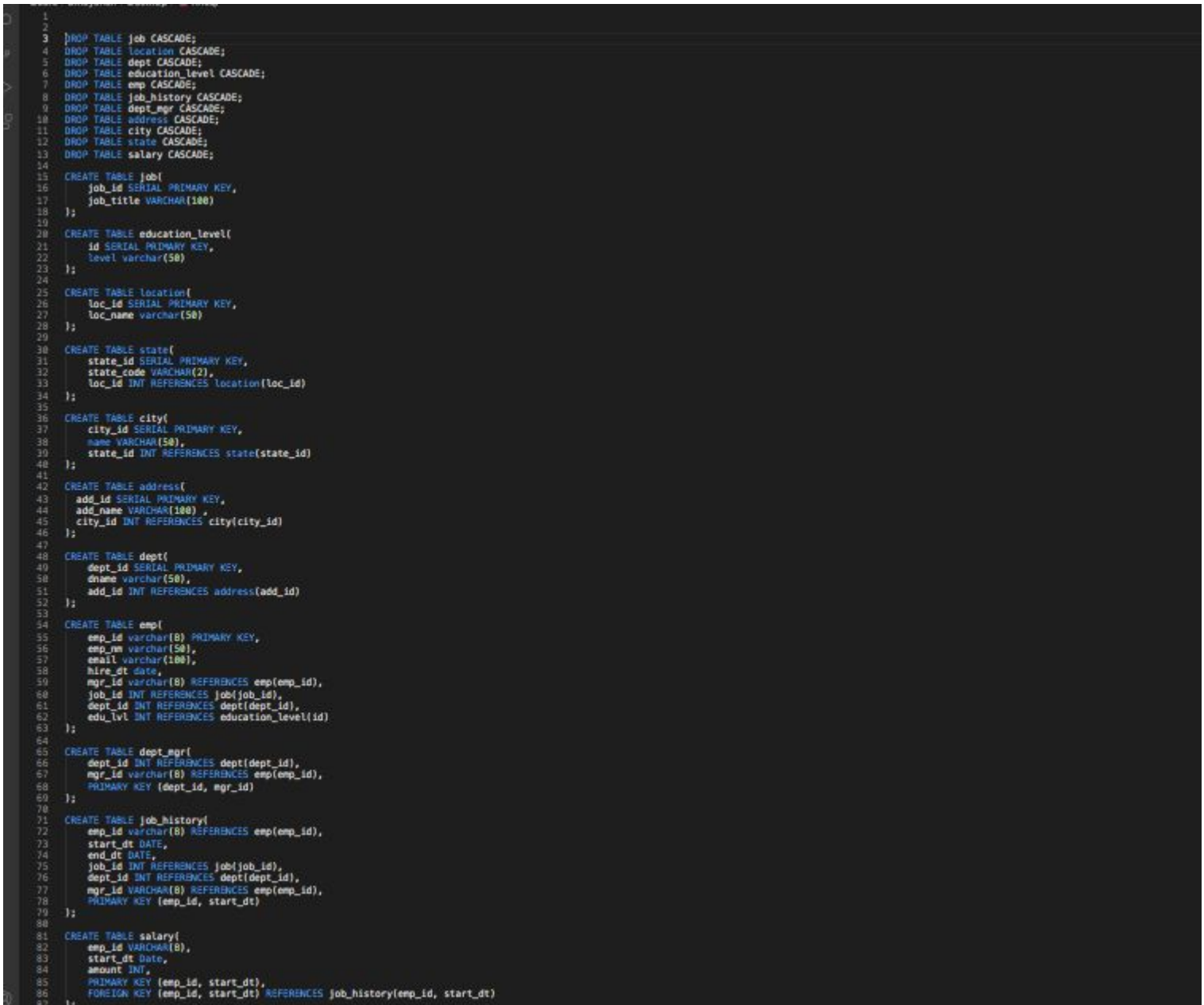
Hints
Your DDL script will be graded by running the code you submit. Please ensure your SQL code runs properly!

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

DROP TABLE job CASCADE;
DROP TABLE location CASCADE;
DROP TABLE dept CASCADE;
DROP TABLE education_level CASCADE;
DROP TABLE emp CASCADE;
DROP TABLE job_history CASCADE;
DROP TABLE dept_mgr CASCADE;
DROP TABLE address CASCADE;
DROP TABLE city CASCADE;
DROP TABLE state CASCADE;
DROP TABLE salary CASCADE;

CREATE TABLE job(
    job_id SERIAL PRIMARY KEY,
    job_title VARCHAR(100)
);

CREATE TABLE education_level(
    id SERIAL PRIMARY KEY,
    level varchar(50)
);

CREATE TABLE location(
    loc_id SERIAL PRIMARY KEY,
    loc_name varchar(50)
);

CREATE TABLE state(
    state_id SERIAL PRIMARY KEY,
    state_code VARCHAR(2),
    loc_id INT REFERENCES location(loc_id)
);

CREATE TABLE city(
    city_id SERIAL PRIMARY KEY,
    name VARCHAR(50),
    state_id INT REFERENCES state(state_id)
);

CREATE TABLE address(
  add_id SERIAL PRIMARY KEY,
  add_name VARCHAR(100) ,
  city_id INT REFERENCES city(city_id)
);

CREATE TABLE dept(
    dept_id SERIAL PRIMARY KEY,
    dname varchar(50),
    add_id INT REFERENCES address(add_id)
);

CREATE TABLE emp(
    emp_id varchar(8) PRIMARY KEY,
    emp_nm varchar(50),
    email varchar(100),
    hire_dt date,
    mgr_id varchar(8) REFERENCES emp(emp_id),
    job_id INT REFERENCES job(job_id),
    dept_id INT REFERENCES dept(dept_id),
    edu_lvl INT REFERENCES education_level(id)
);

CREATE TABLE dept_mgr(
    dept_id INT REFERENCES dept(dept_id),
    mgr_id varchar(8) REFERENCES emp(emp_id),
    PRIMARY KEY (dept_id, mgr_id)
);

CREATE TABLE job_history(
    emp_id varchar(8) REFERENCES emp(emp_id),
    start_dt DATE,
    end_dt DATE,
    job_id INT REFERENCES job(job_id),
    dept_id INT REFERENCES dept(dept_id),
    mgr_id VARCHAR(8) REFERENCES emp(emp_id),
    PRIMARY KEY (emp_id, start_dt)
);

CREATE TABLE salary(
    emp_id VARCHAR(8),
    start_dt Date,
    amount INT,
    PRIMARY KEY (emp_id, start_dt),
    FOREIGN KEY (emp_id, start_dt) REFERENCES job_history(emp_id, start_dt)
```

CRUD

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

```
121
122 SELECT e.emp_id, e.emp_nm, e.email, e.hire_dt, e.mgr_id, j.job_title, d.dname
123 FROM emp e JOIN job j
124 ON e.job_id = j.job_id
125 JOIN dept d
126 ON e.dept_id = d.dept_id;
127
```

root@fd6efdc036d9: /home/w

emp_id	emp_nm	email	hire_dt	job_title	dname
E30317	Siena Spitzer	Siena.Spitzer@TechCorp.com	2019-01-03	Network Engineer	Product Development
E37389	Becky Weaver	Becky.Weaver@TechCorp.com	1996-12-08	Sales Rep	Sales
E77884	Conner Kinch	Conner.Kinch@TechCorp.com	2002-12-09	Manager	Product Development
E87230	Sara Erwin	Sara.Erwin@TechCorp.com	2014-11-10	Sales Rep	Sales
E35860	Vivian Kovach	Vivian.Kovach@TechCorp.com	1999-08-18	Sales Rep	Product Development
E56144	Cassidy Clayton	Cassidy.Clayton@TechCorp.com	2013-01-28	Legal Counsel	Distribution
E78732	Randy Myers	Randy.Myers@TechCorp.com	2020-01-24	Administrative Assistant	HQ
E91791	Aaron Richman	Aaron.Richman@TechCorp.com	2017-06-02	Administrative Assistant	Product Development
E87822	Anil Padala	Anil.Padala@TechCorp.com	2014-04-02	Software Engineer	Product Development
E52461	Lena Thorton	Lena.Thorton@TechCorp.com	2014-11-11	Sales Rep	Sales
E32359	Jen Frangias	Jen .Frangias@TechCorp.com	2019-03-24	Sales Rep	Sales
E32058	Curtis Gibson	Curtis.Gibson@TechCorp.com	2009-01-19	Sales Rep	Product Development
E49025	Mark Fiore	Mark.Fiore@TechCorp.com	2005-06-09	Database Administrator	IT
E21348	Nital Thaker	Nital.Thaker@TechCorp.com	2016-09-28	Software Engineer	Product Development
E13085	Susan Cole	Susan.Cole @TechCorp.com	2017-05-01	Shipping and Receiving	Distribution
E80744	Ginger Logan	Ginger.Logan@TechCorp.com	2020-01-09	Network Engineer	Product Development
E16276	Analyn Braza	Analyn.Braza@TechCorp.com	1996-03-07	Sales Rep	Sales
E20848	Edward Eslser	Edward.Eslser@TechCorp.com	2006-07-26	Software Engineer	IT
E17372	Greg Pratt	Greg.Pratt@TechCorp.com	2009-06-08	Sales Rep	Sales
E86828	Philip Barnett	Philip.Barnett@TechCorp.com	2011-09-29	Database Administrator	IT
E93715	Charles Barker	Charles.Barker@TechCorp.com	1998-04-29	Sales Rep	Sales
E55855	Parker Williams	Parker.Williams@TechCorp.com	2018-05-20	Sales Rep	Sales

--More--

CRUD

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

```
131  INSERT INTO job(job_title)
132  VALUES('Web programmer');
133
134  SELECT * FROM job;
```

root@fd6efdc036d9: /home/w

```
postgres=# INSERT INTO job(job_title)
postgres-# 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)

postgres=#
```

CRUD

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

```
138 UPDATE job set job_title = 'Web developer' WHERE job_id='11';
139
140 SELECT * FROM job;
141
```

root@fd6efdc036d9: /home/w

```
postgres=# UPDATE job set job_title = 'Web developer' WHERE job_id='11';
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)

postgres=#
```

CRUD

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

```
144  DELETE FROM job where job_id = 11;
145
146  SELECT * FROM job;
```

root@fd6efdc036d9: /home/w

```
postgres=# DELETE FROM job where job_id = 11;
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)

postgres=#
```

CRUD

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

```
postgres=# SELECT d.dname, count(*) num_employee
postgres-# FROM job_history e JOIN dept d
postgres-# ON e.dept_id = d.dept_id
postgres-# group by d.dname;
        dname         | num_employee
----------------------+--------------
 IT                   |           54
 Product Development  |           70
 HQ                   |           13
 Distribution         |           27
 Sales                |           41
(5 rows)

postgres=#
```

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.**

```
159 SELECT e.emp_nm, j.job_title, d.dname, h.start_dt, h.end_dt, mgr.emp_nm as manager
160 FROM JOB_HISTORY h JOIN emp e
161 ON h.emp_id = e.emp_id
162 JOIN job j
163 ON h.job_id = j.job_id
164 JOIN dept d
165 ON h.dept_id = d.dept_id
166 JOIN emp mgr
167 ON h.mgr_id = mgr.emp_id
168 where e.emp_nm = 'Toni Lembeck';
169
170
```

root@fd6efdc036d9: /home/w

```
postgres=# SELECT e.emp_nm, j.job_title, d.dname, h.start_dt, h.end_dt, mgr.emp_nm as manager
postgres-# FROM JOB_HISTORY h JOIN emp e
postgres-# ON h.emp_id = e.emp_id
postgres-# JOIN job j
postgres-# ON h.job_id = j.job_id
postgres-# JOIN dept d
postgres-# ON h.dept_id = d.dept_id
postgres-# JOIN emp mgr
postgres-# ON h.mgr_id = mgr.emp_id
postgres-# where e.emp_nm = 'Toni Lembeck';
    emp_nm    |       job_title        | dname |  start_dt  |   end_dt   |   manager
--------------+------------------------+-------+------------+------------+--------------
 Toni Lembeck | Network Engineer       | IT    | 1995-03-12 | 2001-07-18 | Jacob Lauber
 Toni Lembeck | Database Administrator | IT    | 2001-07-18 | 2100-02-02 | Jacob Lauber
(2 rows)
```

CRUD

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

I would the salary data to a separate table and then map it to the rest of the tables. You can then limit user access to the table with salary data.

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 hr_dataset as
postgres-# select h.emp_id emp_id, e.emp_nm, e.email, e.hire_dt, j.job_title, sl.amount, d.dname as d
epartment, mgr.emp_nm as manager, h.start_dt, h.end_dt, l.loc_name as location, ad.add_name as addres
s, ct.name as city, s.state_code as state, el.level as "education level"
postgres-# FROM job_history h LEFT JOIN emp e
postgres-# ON h.emp_id = e.emp_id
postgres-# LEFT JOIN job j
postgres-# ON h.job_id = j.job_id
postgres-# LEFT JOIN dept d
postgres-# ON h.dept_id = d.dept_id
postgres-# LEFT JOIN emp mgr
postgres-# ON h.mgr_id = mgr.emp_id
postgres-# LEFT JOIN address ad
postgres-# ON  d.add_id = ad.add_id
postgres-# LEFT JOIN city ct
postgres-# ON ad.city_id = ct.city_id
postgres-# LEFT JOIN state s
postgres-# ON ct.state_id = s.state_id
postgres-# LEFT JOIN location l
postgres-# ON l.loc_id = s.loc_id
postgres-# JOIN education_level el
postgres-# ON e.edu_lvl = el.id
postgres-# LEFT JOIN salary sl
postgres-# ON h.emp_id = sl.emp_id and h.start_dt = sl.start_dt;
CREATE VIEW
postgres=# select count(*) from hr_dataset;
 count
-------
   205
(1 row)
```

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.

** submit screenshot of stored procedure creation code, along with a screenshot of the stored procedure executed using Toni Lembeck as the parameter value

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.

Submit screenshot of code

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.