Tech ABC Corp - HR Database

[Osama Alsubaie, 18/11/2023]

Business Scenario

Business requirement

Tech ABC Corp saw explosive growth with a sudden appearance onto the gaming scene with their new AI-powered video game console. As a result, they have gone from a small 10 person operation to 200 employees and 5 locations in under a year. HR is having trouble keeping up with the growth, since they are still maintaining employee information in a spreadsheet. While that worked for ten employees, it has becoming increasingly cumbersome to manage as the company expands.

As such, the HR department has tasked you, as the new data architect, to design and build a database capable of managing their employee information.

Dataset

The HR dataset you will be working with is an Excel workbook which consists of 206 records, with eleven columns. The data is in human readable format, and has not been normalized at all. The data lists the names of employees at Tech ABC Corp as well as information such as job title, department, manager's name, hire date, start date, end date, work location, and salary.

IT Department Best Practices

The IT Department has certain Best Practices policies for databases you should follow, as detailed in the Best Practices document.

Step 1

Data Architecture Foundations

Step 1: Data Architecture Foundations

Hi,

Welcome to Tech ABC Corp. We are excited to have some new talent onboard. As you may already know, Tech ABC Corp has recently experienced a lot of growth. Our AI powered video game console WOPR has been hugely successful and as a result, our company has grown from 10 employees to 200 in only 6 months (and we are projecting a 20% growth a year for the next 5 years). We have also grown from our Dallas, Texas office, to 4 other locations nationwide: New York City, NY, San Francisco, CA, Minneapolis, MN, and Nashville, TN.

While this growth is great, it is really starting to put a strain on our record keeping in HR. We currently maintain all employee information on a shared spreadsheet. When HR consisted of only myself, managing everyone on an Excel spreadsheet was simple, but now that it is a shared document I am having serious reservations about data integrity and data security. If the wrong person got their hands on the HR file, they would see the salaries of every employee in the company, all the way up to the president.

After speaking with Jacob Lauber, the manager of IT, he suggested I put in a request to have my HR Excel file converted into a database. He suggested I reach out to you as I am told you have experience in designing and building databases. When you are building this, please keep in mind that I want any employee with a domain login to be have read only access the database. I just don't want them having access to salary information. That needs to be restricted to HR and management level employees only. Management and HR employees should also be the only ones with write access. By our current estimates, 90% of users will be read only.

I also want to make sure you know that am looking to turn my spreadsheet into a live database, one I can input and edit information into. I am not really concerned with reporting capabilities at the moment. Since we are working with employee data we are required by federal regulations to maintain this data for at least 7 years; additionally, since this is considered business critical data, we need to make sure it gets backed up properly.

As a final consideration. We would like to be able to connect with the payroll department's system in the future. They maintain employee attendance and paid time off information. It would be nice if the two systems could interface in the future

I am looking forward to working with you and seeing what kind of database you design for us.

Thanks,
Sarah Collins
Head of HR

Data Architect Business Requirement

- ## Purpose of the new database:

  The current Excel spreadsheet lacks scalability and security, resulting in data integrity issues across multiple regions and inadequate protection of sensitive information like salaries. To address these concerns, a new OLTP database is proposed, aiming to offer enhanced scalability, data integrity, and security measures. This solution accommodates data growth while specifically catering to the business partner's needs.

- ## Describe current data management solution:

  The company's HR data is currently managed in a single Excel spreadsheet comprising 15 columns. This spreadsheet, containing sensitive information, is shared company-wide, posing potential security risks.

- ## Describe current data available:

  The current data available is a single spreadsheet with 15 columns and 206 records.
  The columns are -> [employee id, employee name, e-mail, hire date, job title, salary, department, manager, start date, end date, location, address, city, state, education level].

- ## Additional data requests:

  1- Scaling the database, reflecting an increase in records driven by the company's annual hiring.
  2- Integration with the payroll system, facilitating access to paid time off and employee attendance details.

- ## Who will own/manage data

  The owner of the data will be HR and management level employees.

- ## Who will have access to database

  1- Regular employees - Read access.
  The access will be through user and password login and sensitive data, such as salary, will not be accessible for those users.

  2- HR and management level employees - Read and Write access.
  Those users will have unlimited access to the database.

Data Architect Business Requirement

- ## Estimated size of database

  The database will consist of 7 tables, Estimation for each table:

  Employee: ~200 rows, 7 columns

  Job: ~200 rows, 7 columns

  Geography: 6 rows, 5 columns

  Education Level: 10 rows, 2 columns

  Job Title: 10 rows, 2 columns

  Department: 20 rows, 2 columns

  Salary: ~200 rows, 2 columns

  The total estimated size is ~1000 records for this year.

- ## Estimated annual growth

  Anticipating a 20% annual growth, the business partner foresees an increase of 40 records in the tables for jobs, employees, and salaries each, resulting in a total of 120 additional records within a year from the current 200 records.

- ## Is any of the data sensitive/restricted

  The business partner states that the salaries information are sensitive, and the regular employees must not have access to that information.

Data Architect Technical Requirement

- ## Justification for the new database

1. Data integrity: the new database will provide data integrity since it relies on ACID principles .

2. Data access management: DBMS addresses security concerns absent in spreadsheets, particularly safeguarding sensitive information from unauthorized access by restricting visibility to designated users.

- ## Database objects

Tables of the database:
Employee - employee personal information.
Salary - sensitive information about salary and the access will be restricted.
Job - job titles of the employees in the company.
Education – education information.
Department - department names within the company.
Location - location and the address of the workplace.
City – list of the location cities.
State – list of the city's states.

- ## Data ingestion

Since the current storage solution is an Excel spreadsheet,, the ETL approach should be chosen.

Data Architect Technical Requirement

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

  Ownership: HR and Management level employees

  User Access: **Full access:** HR and management level employees.
  Restricted access with salary not visible: All regular employees

- **Scalability**

  Sharding isn't necessary for this database since the input data processing won't entail massive volumes.
  However, replication is vital for scalability, especially with 90% of usage dedicated to read access across diverse locations. This approach ensures efficient access and availability of data across different user locations.

- **Flexibility**

  Our solution's design aims for seamless integration with the payroll department system in the future. We prioritize standards and structures conducive to smooth integration, such as adopting shared employee IDs across both systems. This alignment simplifies future processes, allowing for straightforward data linkage between the systems.

- **Storage & retention**

  Storage (disk or in-memory):

  - Standard partition of 1GB since it will not increase 10k rows in the next year.
  - The data should be stored in disk as no high-level computation will be performed.

  Retention:

  - Data has to be kept for at least 7 years required by federal regulation.

- **Backup**

  For Critical Business Data, the backup schedule is full backup 1x per week, with an incremental backup daily.

Step 2

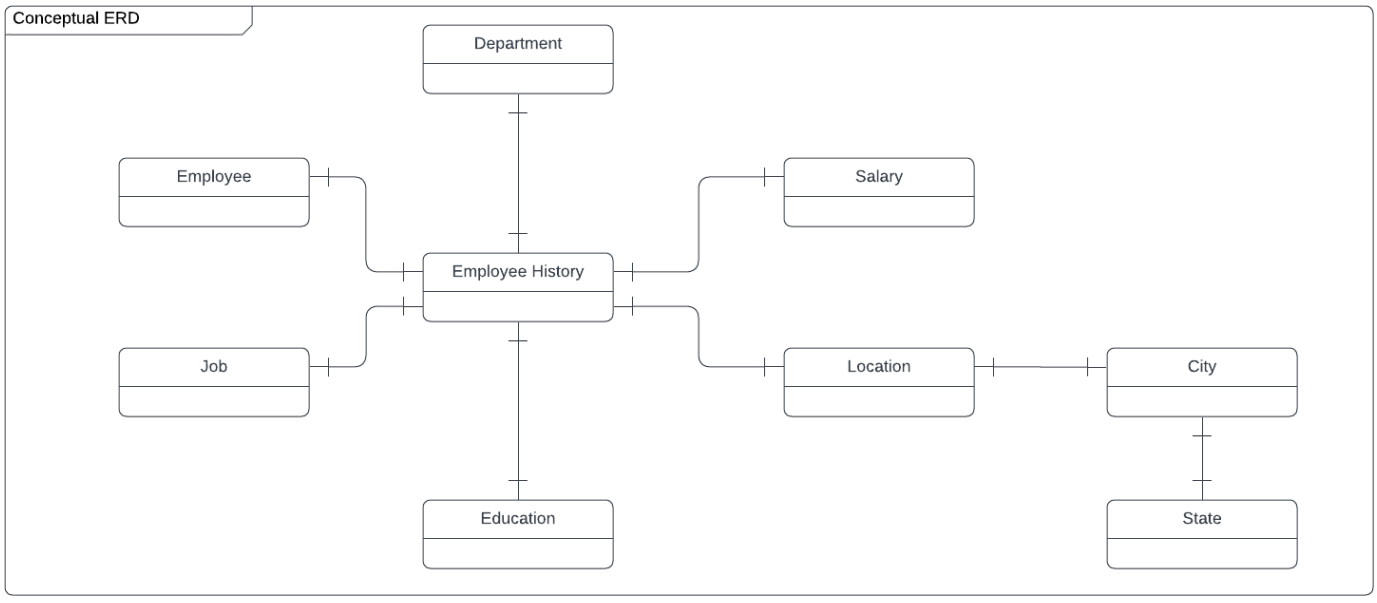
Relational Database Design

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.

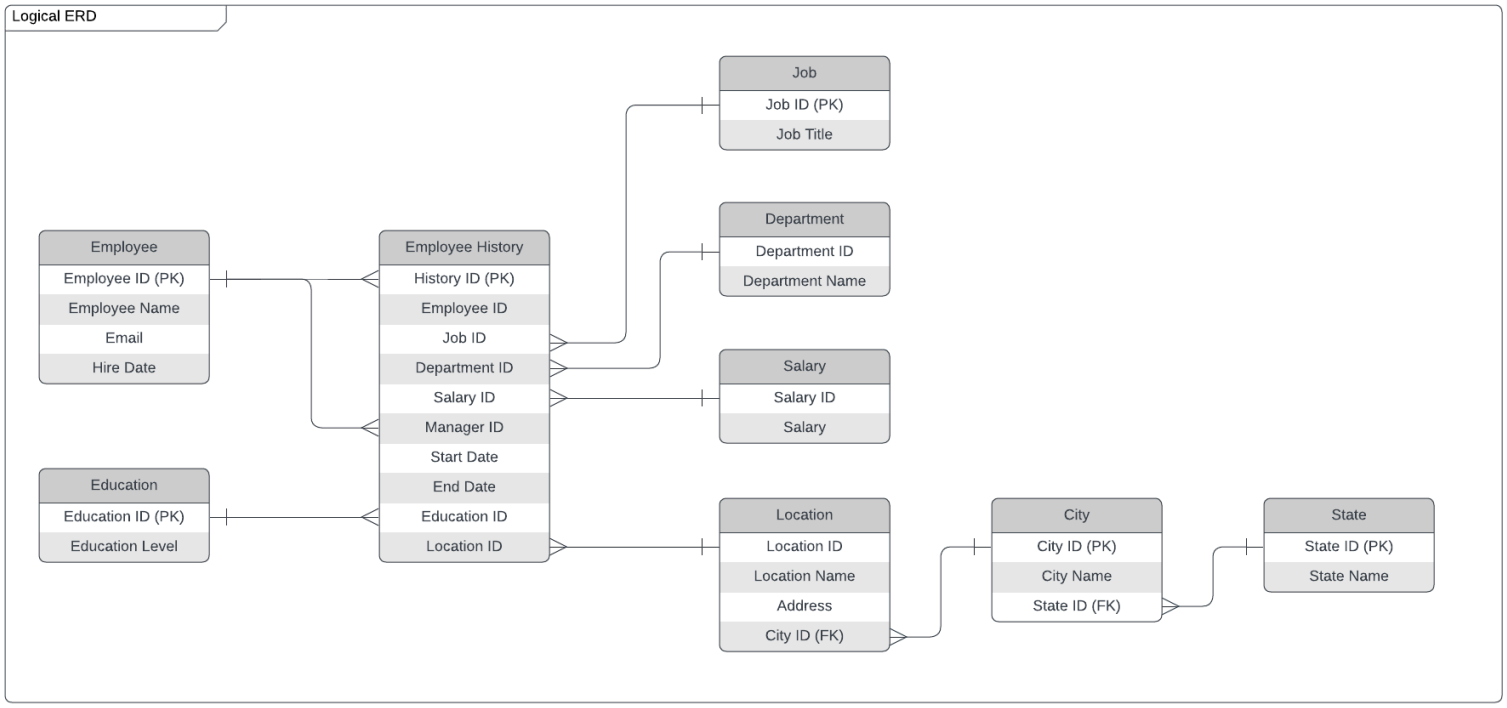


ERD

- **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.

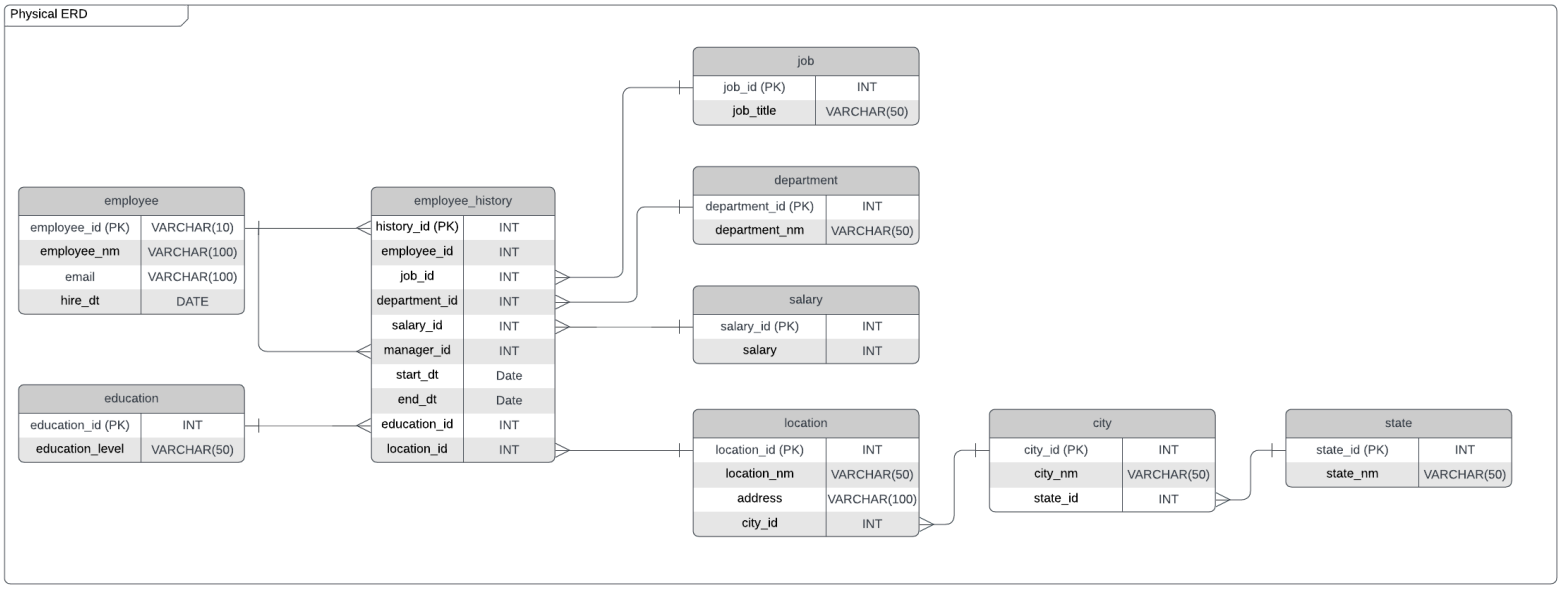
Use Lucidchart’s built-in template for DBMS ER Diagram UML.



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

Query | Query History

```sql
CREATE TABLE IF NOT EXISTS employee (
                                employee_id VARCHAR(10) PRIMARY KEY,
                                employee_nm VARCHAR(100),
                                email       VARCHAR(100),
                                hire_dt     DATE
                                );

CREATE TABLE IF NOT EXISTS education (
                                education_id    SERIAL PRIMARY KEY,
                                education_level VARCHAR(50)
                                );

CREATE TABLE IF NOT EXISTS job (
                                job_id      SERIAL PRIMARY KEY,
                                job_title   VARCHAR(50)
                                );

CREATE TABLE IF NOT EXISTS department (
                                department_id SERIAL PRIMARY KEY,
                                department_nm VARCHAR(50)
                                );

CREATE TABLE IF NOT EXISTS salary (
                                salary_id   SERIAL PRIMARY KEY,
                                salary      INT
                                );

CREATE TABLE IF NOT EXISTS state (
                                state_id SERIAL PRIMARY KEY,
                                state_nm VARCHAR(50)
                                );

CREATE TABLE IF NOT EXISTS city (
                                city_id     SERIAL PRIMARY KEY,
                                city_nm     VARCHAR(50),
                                state_id    INT
                                );

CREATE TABLE IF NOT EXISTS location (
                                location_id SERIAL PRIMARY KEY,
                                location_nm VARCHAR(50),
                                address     VARCHAR(100),
                                city_id     INT
                                );

CREATE TABLE IF NOT EXISTS employee_history (
                                history_id      SERIAL      PRIMARY KEY,
                                employee_id     VARCHAR(10) REFERENCES employee (employee_id),
                                job_id          INT         REFERENCES job (job_id),
                                department_id   INT         REFERENCES department (department_id),
                                salary_id       INT         REFERENCES salary (salary_id),
                                manager_id      VARCHAR(10) REFERENCES employee (employee_id),
                                start_dt        DATE,
                                end_dt          DATE,
                                education_id    INT         REFERENCES education (education_id),
                                location_id     INT         REFERENCES location (location_id)
                                );
```

Data Output | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 37 msec.
```

Total rows: 0 of 0 | Query complete 00:00:00.037

CRUD

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

Query | Query History

```
1  --Question #1: Return a list of employees with Job Titles and Department Names
2
3  SELECT eh.employee_id, e.employee_nm, j.job_title, d.department_nm
4  FROM employee_history eh
5  JOIN employee e     ON eh.employee_id = e.employee_id
6  JOIN job j          ON eh.job_id = j.job_id
7  JOIN department d   ON eh.department_id = d.department_id
8
9
10
```

Data Output | Messages | Notifications

	employee_id character varying (10)	employee_nm character varying (100)	job_title character varying (50)	department_nm character varying (50)
1	E17469	Haifa Hajiri	Administrative Assistant	Distribution
2	E27621	Wendell Mobley	Administrative Assistant	Distribution
3	E27909	Michael Sperduti	Administrative Assistant	Distribution
4	E35053	Ashley Bergman	Administrative Assistant	Distribution
5	E51723	Carlos Lopez	Administrative Assistant	Distribution
6	E59688	Jason Wingard	Administrative Assistant	Distribution
7	E18697	Anita Deluise	Administrative Assistant	HQ
8	E25640	Joseph Donohue	Administrative Assistant	HQ
9	E35075	John Certa	Administrative Assistant	HQ
10	E56459	Raven Landis	Administrative Assistant	HQ
11	E78732	Randy Myers	Administrative Assistant	HQ
12	E90407	Danny Godiksen	Administrative Assistant	HQ
13	E45824	Raj Prudvi	Administrative Assistant	IT
14	E48148	Alexis Fitzpatrick	Administrative Assistant	IT
15	E50012	Ann Roberto	Administrative Assistant	IT
16	E81502	Danny Laxton	Administrative Assistant	IT
17	E83558	Laura McKenna	Administrative Assistant	IT
18	E99949	William Graf	Administrative Assistant	IT

CRUD

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

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

INSERT INTO job (job_title)
VALUES          ('Web Programmer');

SELECT * FROM job
```

Data Output | Messages | Notifications

	job_id [PK] integer	job_title 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**

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

UPDATE job
SET     job_title = 'Web Developer'
WHERE   job_title = 'Web Programmer';

SELECT * FROM job
```

Data Output | Messages | Notifications

	job_id [PK] integer	job_title 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**

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

DELETE FROM job
WHERE job_title = 'Web Developer';

SELECT * FROM job
```

Data Output | Messages | Notifications

	job_id [PK] integer	job_title 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?**

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

SELECT COUNT(eh.*), d.department_nm
FROM employee_history eh
JOIN department d
ON eh.department_id = d.department_id
GROUP BY department_nm
```

Data Output | Messages | Notifications

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

CRUD

- **Question 6: Write a query that returns current and past jobs (include employee name, job title, department, manager name, start and end date for position) for employee Toni Lembeck.**

```
--Question #6: Write a query that returns current and past jobs (include employee name, job title, department, manager name, start and end date for position) for employee Toni Lembeck.

SELECT  e.employee_nm   AS "employee name",
        j.job_title     AS "job title",
        d.department_nm AS "department name",
        (SELECT employee_nm FROM employee e WHERE eh.manager_id = e.employee_id) AS "manager name",
        eh.start_dt     AS "start date",
        eh.end_dt       AS "end date"
FROM employee_history eh
JOIN employee e     ON eh.employee_id = e.employee_id
JOIN job j          ON eh.job_id = j.job_id
JOIN department d   ON eh.department_id = d.department_id
WHERE employee_nm = 'Toni Lembeck'
```

Data Output | Messages | Notifications

	employee name character varying (100)	job title character varying (50)	department name character varying (50)	manager name character varying (100)	start date date	end date 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.**

  To restrict access to employee salaries in SQL Server:

  Column-Level Permissions: Grant SELECT permission to specific columns excluding salaries.

  Views: Create views without salary details for access control.

  Row-Level Security: Implement policies to restrict row access based on conditions.

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

Query | Query History

```sql
-- Standout Suggestion #1: Create a view that returns all employee attributes; results should resemble initial Excel file

CREATE VIEW HR AS
SELECT eh.employee_id               AS "EMP_ID",
       e.employee_nm                AS "EMP_NM",
       e.email                      AS "EMAIL",
       e.hire_dt                    AS "HIRE_DT",
       j.job_title                  AS "JOB_TITLE",
       s.salary                     AS "SALARY",
       d.department_nm              AS "DEPARTMENT",
       (SELECT employee_nm FROM employee e WHERE eh.manager_id = e.employee_id) AS "MANAGER",
       eh.start_dt                  AS "START_DT",
       eh.end_dt                    AS "END_DT",
       l.location_nm                AS "LOCATION",
       l.address                    AS "ADDRESS CITY",
       st.state_nm                  AS "STATE",
       edu.education_level          AS "EDUCATION LEVEL"
FROM employee_history eh
JOIN employee e      ON eh.employee_id = e.employee_id
JOIN job j           ON eh.job_id = j.job_id
JOIN department d    ON eh.department_id = d.department_id
JOIN education edu   ON eh.education_id = edu.education_id
JOIN salary s        ON eh.salary_id = s.salary_id
JOIN location l      ON eh.location_id = l.location_id
JOIN city c          ON l.city_id = c.city_id
JOIN state st        ON c.state_id = st.state_id

SELECT * FROM HR
```

Data Output | Messages | Notifications

	EMP_ID character varying (10)	EMP_NM character varying (100)	EMAIL character varying (100)	HIRE_DT date	JOB_TITLE character varying (50)	SALARY integer	DEPARTMENT character varying (50)	MANAGER character varying (100)	START_DT date	END_DT date	LOCATION character varying (50)	ADDRESS CITY character varying (100)	STATE character varying (50)	EDUCATION LEVEL character varying (50)
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 James Way	CA	No College
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 James Way	CA	Some College
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 James Way	CA	Associates Degree
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 Nicollet Mall	MN	No College
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 Broadway	NY	No College
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 James Way	CA	Some College
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 ABC Corp Way	TX	Associates Degree
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 ABC Corp Way	TX	No College
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 ABC Corp Way	TX	Some College
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 ABC Corp Way	TX	Associates Degree
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 ABC Corp Way	TX	Associates Degree
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 ABC Corp Way	TX	Associates Degree
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 Broadway	NY	Associates Degree
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 Nicollet Mall	MN	No College
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 Nicollet Mall	MN	No College
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 Broadway	NY	Some College
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 Broadway	NY	Associates Degree
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 ABC Corp Way	TX	Bachelors Degree
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 ABC Corp Way	TX	Some College
20	E16346	Jill Fram	Jill.Fram@TechCorp.com	2007-12-16	Administrative Assistant	50462	Product Development	Conner Kinch	2007-12-16	2100-01-20	HQ	1 Tech ABC Corp Way	TX	Associates Degree
21	E37523	Alan Mecklet	Alan.Mecklet@TechCorp.com	2017-12-02	Administrative Assistant	35638	Product Development	Conner Kinch	2017-12-02	2100-01-21	East Coast	165 Broadway	NY	Bachelors Degree
22	E57987	Nicole Lee	Nicole.Lee@TechCorp.com	2016-05-15	Administrative Assistant	49786	Product Development	Conner Kinch	2016-05-15	2100-01-22	West Coast	705 James Way	CA	No College
23	E73518	Cheryl Pike	Cheryl.Pike@TechCorp.com	2001-03-24	Administrative Assistant	51207	Product Development	Conner Kinch	2001-03-24	2100-01-23	South	422 Broadway	TN	No College
24	E74206	Krishna Purli	Krishna.Purli@TechCorp.com	2016-12-10	Administrative Assistant	47000	Product Development	Conner Kinch	2016-12-10	2100-01-24	South	422 Broadway	TN	Some College

Total rows: 205 of 205 | Query complete 00:00:00.043 | Ln 1, Col 122

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.

Query | Query History

```sql
-- Standout Suggestion #2: Create a stored procedure with parameters that returns current and past jobs (include employee name, job title, department, manager name, start and end date for position) when given an employee name.

CREATE OR REPLACE FUNCTION get_employee_history(param_employee_name VARCHAR(100))
RETURNS TABLE (
    employee_id     VARCHAR(10),
    employee_nm     VARCHAR(100),
    job_title       VARCHAR(50),
    department_name VARCHAR(50),
    manager_name    VARCHAR(100),
    start_date      DATE,
    end_date        DATE
) AS $$
BEGIN
    RETURN QUERY
    SELECT
        eh.employee_id,
        e.employee_nm,
        j.job_title,
        d.department_nm,
        (SELECT emp.employee_nm FROM employee emp WHERE eh.employee_id = emp.employee_id) AS manager_name,
        eh.start_dt,
        eh.end_dt
    FROM
        employee_history eh
    JOIN
        employee e ON eh.employee_id = e.employee_id
    JOIN
        job j ON eh.job_id = j.job_id
    JOIN
        department d ON eh.department_id = d.department_id
    WHERE
        e.employee_nm = param_employee_name;
END;
$$ LANGUAGE plpgsql;

SELECT * FROM get_employee_history('Toni Lembeck');

```

Data Output | Messages | Notifications

	employee_id character varying	employee_nm character varying	job_title character varying	department_name character varying	manager_name character varying	start_date date	end_date date
1	E27498	Toni Lembeck	Database Administrator	IT	Toni Lembeck	2001-07-18	2100-02-02
2	E27498	Toni Lembeck	Network Engineer	IT	Toni Lembeck	1995-03-12	2001-07-18

Standout Suggestion 3

Implement user security on the restricted salary attribute.

Query | Query History

```
1  -- Standout Suggestion #3: Implement user security on the restricted salary attribute.
2
3  -- Create user
4  CREATE USER NoMgr;
5  -- Grant access to the DB
6  GRANT SELECT ON HR TO NoMgr;
7  -- Revoke access to the salary table for the NoMgr user
8  REVOKE SELECT ON TABLE salary FROM NoMgr;
9
10
11
```

Data Output | Messages | Notifications

```
REVOKE

Query returned successfully in 59 msec.
```