Tech ABC Corp - HR Database

[Thu Huynh - Aug 8th, 2022]

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:**

  Maintain all employee information that ensures data integrity and data security when the company has more and more employees

- **Describe current data management solution:**

  All employee information is saved in a shared spreadsheet

- **Describe current data available:**

A spreadsheet that contains data of more than 200 employees in all levels, from employee contact, job information to the sensitive data of salary

- **Additional data requests:**

  Following federal regulations, employee data needs to be maintained for at least 7 years and gets backed up properly. Finally, database needs to be able to connect with the payroll department's system in the future

- **Who will own/manage data**

  Management and HR employees will own/manage data

- **Who will have access to database**

  All employees with domain login will have **read-only** access with only one restriction: **no access to salary information**

Data Architect Business Requirement

- **Estimated size of database**

  As current size of company, size of the database in terms of numbers of rows: from 200 to 500 rows

- **Estimated annual growth**

  Expected growth to the data: currently the company has more than 200 employees with the expected 20% growth a year for the next 5 years, this will also lead to the expected growth in data

- **Is any of the data sensitive/restricted**
  - All employees with domain login: will have **read-only** access with only one restriction: **no access to salary information**
  - Management and HR employees will be the only ones with full access

Data Architect Technical Requirement

- **Justification for the new database**

  Ensures data integrity and data security

- **Database objects**
  - **Table:** human resources (hr), employee, job, education, department, location
  - **View**: hr_view (for non management and HR users, restricted from **salary** column

- **Data ingestion**

  ETL

Data Architect Technical Requirement

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

  Ownership: HR Employees

  User Access:
  All employees with domain login: will have read-only access with only one restriction: **no access to salary information**
  Management and HR employees will be the only ones with full access

- **Scalability**

  Replication

- **Flexibility**

  Direct feed will be more optimal in the future for connecting with the real database and payroll system

- **Storage & retention**

  Storage : disk
  Retention: 7 years following federal regulations for employees data

- **Backup**
  Full back-up daily and weekly

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.

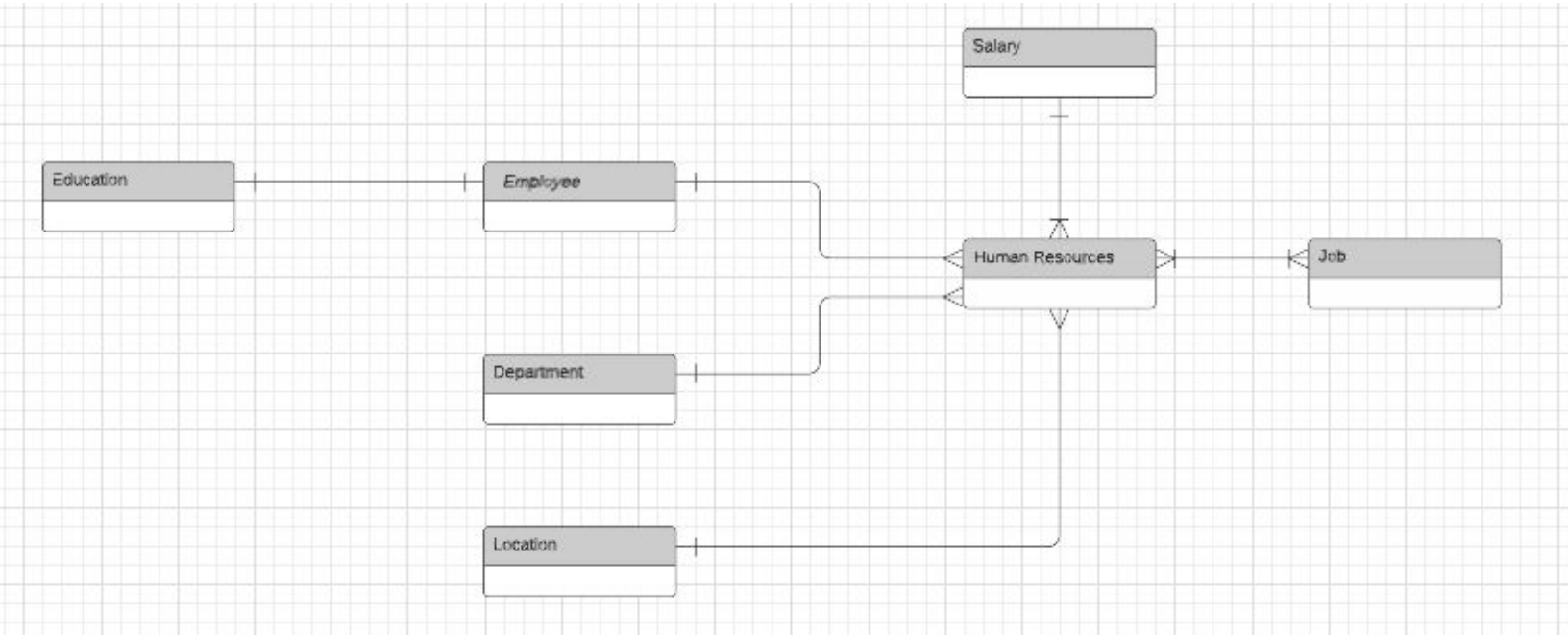
You will submit a screenshot for each of the 3 ERDs you create. You will find detailed instructions for developing each of the ERDs over the next several pages.

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.

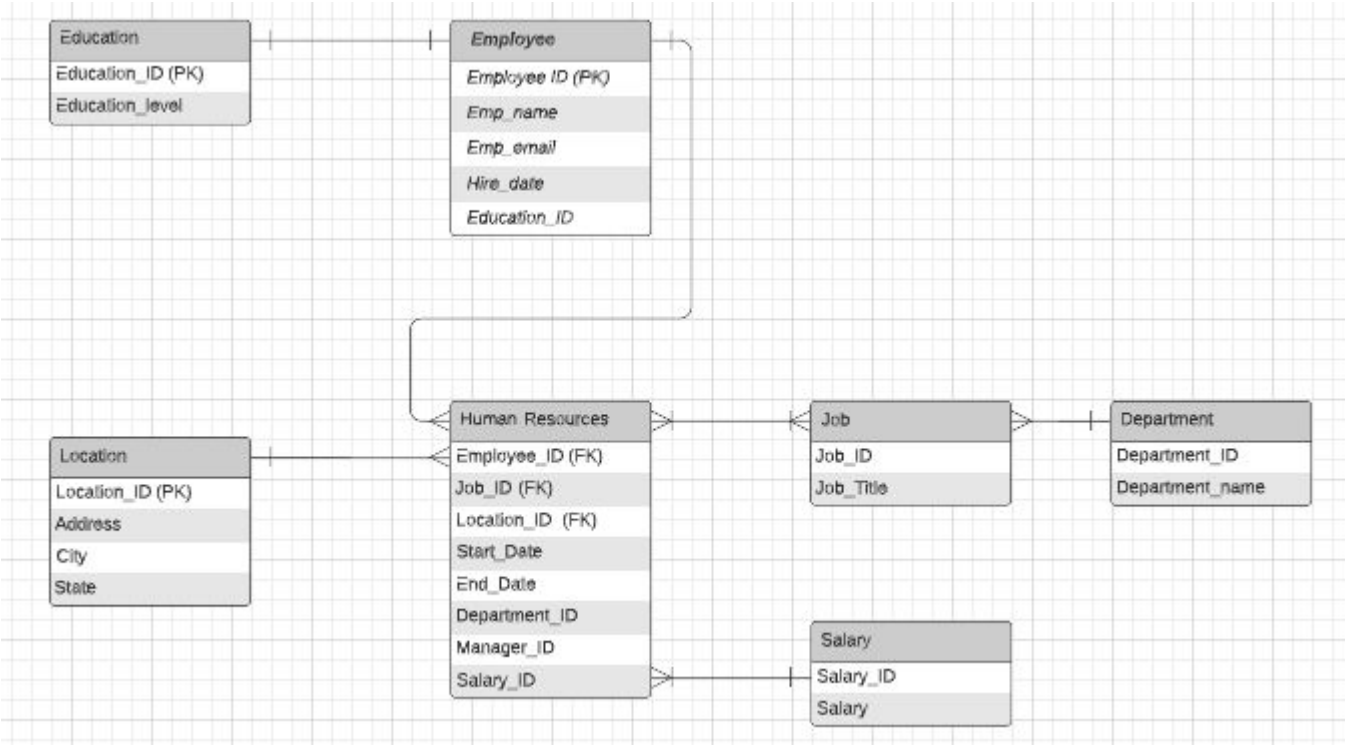


ERD

- **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.

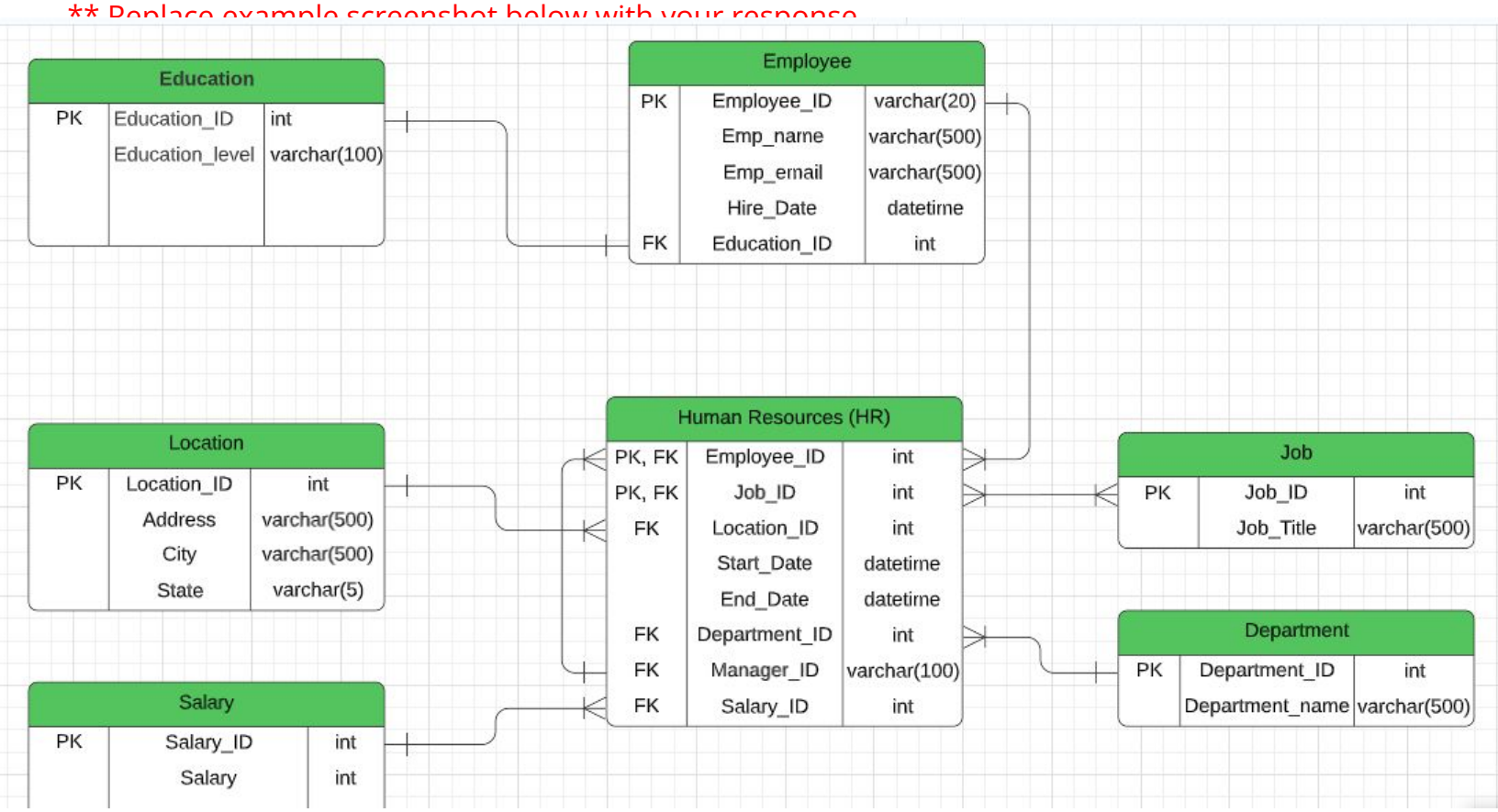


ERD

- **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.

** Replace example screenshot below with your response



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. *(Attached file name: DDL.sql)*

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.

DDL.sql

```sql
DROP TABLE IF EXISTS education CASCADE;
CREATE TABLE education (
    education_id serial,
    education_level VARCHAR(100),
    PRIMARY KEY (education_id)
);

DROP TABLE IF EXISTS location CASCADE;
CREATE TABLE location (
    location_id serial,
    location VARCHAR(500),
    address VARCHAR(500),
    city VARCHAR(500),
    state VARCHAR(5),
    PRIMARY KEY (location_id)
);

DROP TABLE IF EXISTS employee CASCADE;
CREATE TABLE employee (
    employee_id VARCHAR(100),
    emp_name VARCHAR(500),
    emp_email VARCHAR(500),
    hire_date date,
    education_id int,
    PRIMARY KEY (employee_id)
);
```

DDL

```sql
DROP TABLE IF EXISTS department CASCADE;
CREATE TABLE department (
    department_id serial,
    department_name VARCHAR(500),
    PRIMARY KEY (department_id)
);

DROP TABLE IF EXISTS job CASCADE;
CREATE TABLE job (
    job_id serial,
    job_title VARCHAR(500),
    PRIMARY KEY (job_id)
);

DROP TABLE IF EXISTS salary CASCADE;
CREATE TABLE salary (
    salary_id SERIAL,
    salary int,
    PRIMARY KEY (salary_id)
);

DROP TABLE IF EXISTS hr CASCADE;
CREATE TABLE hr (
    employee_id VARCHAR(100),
    job_id int,
    location_id int,
    start_date date,
    end_date date,
    department_id int,
    manager_id varchar(100),
    salary_id int,
    PRIMARY KEY (employee_id, job_id)
);
```

```sql
--ADD FOREIGN KEY CONSTRAINT
ALTER TABLE employee
ADD CONSTRAINT education_fk FOREIGN KEY (education_id)
REFERENCES education (education_id)
ON DELETE CASCADE;

ALTER TABLE hr
ADD CONSTRAINT employee_fk FOREIGN KEY (employee_id)
REFERENCES employee (employee_id)
ON DELETE CASCADE,
ADD CONSTRAINT job_fk FOREIGN KEY (job_id)
REFERENCES job (job_id)
ON DELETE CASCADE,
ADD CONSTRAINT manager_fk FOREIGN KEY (manager_id)
REFERENCES employee (employee_id)
ON DELETE CASCADE,
ADD CONSTRAINT department_fk FOREIGN KEY (department_id)
REFERENCES department (department_id)
ON DELETE CASCADE,
ADD CONSTRAINT salary_fk FOREIGN KEY (salary_id)
REFERENCES salary (salary_id)
ON DELETE CASCADE,
ADD CONSTRAINT location_fk FOREIGN KEY (location_id)
REFERENCES location (location_id)
ON DELETE CASCADE;
```

SELECT *

```
1  SELECT * FROM job;
```

Data output | Messages | Notifications

	job_id [PK] integer	job_title character varying (500)
1	11	Shipping and Receiving
2	12	Sales Rep
3	13	Administrative Assistant
4	14	Design Engineer
5	15	Database Administrator
6	16	Software Engineer
7	17	Manager
8	18	Legal Counsel
9	19	President
10	20	Network Engineer

```
1  SELECT * FROM education;
```

Data output | Messages | Notifications

	education_id [PK] integer	education_level character varying (100)
1	8	Associates Degree
2	9	Masters of Business A...
3	10	Masters Degree
4	11	Bachelors Degree
5	12	Doctorate
6	13	No College
7	14	Some College

```
1  SELECT * FROM salary;
```

Data output | Messages | Notificatio

	salary_id [PK] integer	salary integer
1	206	28700
2	207	96416
3	208	52924
4	209	103370
5	210	49786
6	211	117958
7	212	167887

```
1  SELECT * FROM employee;
```

Data output | Messages | Notifications

	employee_id [PK] character varying (100)	emp_name character varying (500)	emp_email character varying (500)	hire_date date	education_id integer
1	E90439	Janice McQueen	Janice.McQueen@Tec...	2000-07-23	11
2	E87822	Anil Padala	Anil.Padala@TechCorp...	2014-04-02	11
3	E70603	Soek Sohn	Soek.Sohn@TechCorp....	2006-07-06	11
4	E56459	Raven Landis	Raven.Landis@TechCo...	2016-04-21	8
5	E69297	Nilden Tutalar	Nilden.Tutalar@TechC...	2013-03-21	8
6	E65052	Leobrian Mason	Leobrian.Mason@Tech...	1995-09-15	11
7	E13085	Susan Cole	Susan.Cole @TechCor...	2017-05-01	11
8	E95199	Carlos Fernandes	Carlos.Fernandes@Tec...	2017-01-09	10
9	E31241	Laura House	Laura.House@TechCor...	2014-05-10	8
10	E72436	Eileen Chuss	Eileen.Chuss@TechCor...	2013-05-11	11
11	E27909	Michael Sperduti	Michael.Sperduti@Tec...	2014-06-20	8
12	E16276	Analyn Braza	Analyn.Braza@TechCo...	1996-03-07	11
13	E36346	Tyrone Curtis	Tyrone.Curtis@TechCo...	1999-06-22	11

SELECT *

```
1  SELECT * FROM location;
```

Data output | Messages | Notifications

	location_id [PK] integer	location character varying (500)	address character varying (500)	city character varying (500)	state character varying (5)
1	6	East Coast	165 Broadway	New York City	NY
2	7	HQ	1 Tech ABC Corp Way	Dallas	TX
3	8	West Coast	705 James Way	San Francisco	CA
4	9	South	422 Broadway	Nashville	TN
5	10	Midwest	1300 Nicollet Mall	Minnapolis	MN

```
1  SELECT * FROM department;
```

Data output | Messages | Notifications

	department_id [PK] integer	department_name character varying (500)
1	6	Distribution
2	7	HQ
3	8	Product De
4	9	Sales
5	10	IT

```
1  SELECT * FROM hr;
```

Data output | Messages | Notifications

	employee_id [PK] character varying (100)	job_id [PK] integer	location_id integer	start_date date	end_date date	department_id integer	manager_id character varying (100)	salary_id integer
1	E95199	12	7	2017-01-09	[null]	8	E77884	309
2	E22197	20	7	2013-09-22	[null]	10	E44426	260
3	E12397	20	7	2013-11-17	[null]	8	E77884	324
4	E72436	12	9	2013-05-11	[null]	9	E88667	310
5	E25640	13	7	2007-12-27	[null]	7	E17054	381
6	E12562	13	7	1996-04-14	[null]	8	E77884	299
7	E14913	20	7	1998-07-15	[null]	8	E77884	351
8	E85640	16	8	2001-02-03	[null]	10	E44426	246
9	E86828	15	8	2011-09-29	[null]	10	E44426	410
10	E83512	20	6	1996-10-22	[null]	10	E44426	316
11	E60901	12	6	2010-01-10	[null]	9	E88667	292
12	E87370	12	8	2013-11-18	[null]	9	E88667	385
13	E13596	12	7	2012-04-09	[null]	8	E77884	390
14	E57502	12	8	2016-07-21	[null]	8	E77884	344
15	E16276	12	7	1996-03-07	[null]	9	E88667	305

CRUD

(Attached file name: CRUD.sql)

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

Query | Query History

```
1  select e.emp_name, j.job_title, d.department_name
2  from hr
3  left join employee e
4  on hr.employee_id = e.employee_id
5  left join job j
6  on hr.job_id = j.job_id
7  left join department d
8  on hr.department_id = d.department_id
```

Data output | Messages | Notifications

	emp_name character varying (500)	job_title character varying (500)	department_name character varying (500)
1	Jeff Barnhill	Design Engineer	IT
2	John Certa	Administrative Assistant	HQ
3	Tim Lawler	Network Engineer	IT
4	Curtis Steward	Sales Rep	Sales
5	Aaron Gordon	Network Engineer	Product Development
6	Stan Frank	Shipping and Receiving	Distribution
7	Mary Wesson	Software Engineer	IT
8	Bertin Dakouo	Sales Rep	Product Development
9	Darryl Reamer	Legal Counsel	Product Development
10	William Graf	Administrative Assistant	IT
11	Roseann Martineeti	Sales Rep	Product Development

CRUD

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

```
1  INSERT INTO job (job_title)
2  VALUES ('Web Programmer');
3
4  SELECT * FROM job;
```

Data output | Messages | Notifications

	job_id [PK] integer	job_title character varying (500)
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**

```
1  UPDATE job
2  SET job_title = 'Web Developer'
3  WHERE job_title = 'Web Programmer';
4
5  SELECT * FROM job;
```

Data output | Messages | Notifications

	job_id [PK] integer	job_title character varying (500)
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**

```
1  DELETE FROM job
2  WHERE job_title = 'Web Developer';
3
4  SELECT * FROM job;
```

Data output | Messages | Notifications

	job_id [PK] integer	job_title character varying (500)
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?**

```sql
SELECT d.department_name, count(*) as Total_Employees
FROM hr
LEFT JOIN department d
ON hr.department_id = d.department_id
WHERE hr.end_date IS NULL   --only count current active employee

GROUP BY d.department_name;
```

Data output | Messages | Notifications

	department_name character varying (500)	total_employees bigint
1	Product Development	69
2	HQ	13
3	Distribution	25
4	Sales	40
5	IT	52

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.**

```
1  SELECT e.emp_name, j.job_title, d.department_name, m.emp_name, start_date, end_date
2  FROM hr
3  INNER JOIN employee e
4  ON  hr.employee_id = e.employee_id
5  AND e.emp_name = 'Toni Lembeck'
6  LEFT JOIN job j
7  ON hr.job_id = j.job_id
8  LEFT JOIN department d
9  ON hr.department_id = d.department_id
10 LEFT JOIN employee m
11 ON hr.manager_id = m.employee_id;
```

Data output | Messages | Notifications

	emp_name character varying (500)	job_title character varying (500)	department_name character varying (500)	emp_name character varying (500)	start_date date	end_date date
1	Toni Lembeck	Network Engineer	IT	Jacob Lauber	1995-03-12	2001-07-17
2	Toni Lembeck	Database Administrator	IT	Jacob Lauber	2001-07-18	[null]

CRUD

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

- Revoke all users' permission on table **salary** (except for management & HR users)

Appendix