Tech ABC Corp - HR Database

Step 1

Data Architecture Foundations

Data Architect Business Requirement

The new HR database will be used to maintain all employee information ensuring appropriate data integrity and security. The database will replace the current solution, which is a shared Excel spreadsheet.

The complete currently available data covering 205 rows and 15 columns will be migrated from the spreadsheet to the database. Hence, the information of 199 unique employees will be stored in the database and those employees will be able to access the data using their domain account. Only HR department and management employees will be able to create, read, update and delete records without any restrictions. Other employees will have read-only access without having access to sensitive salary information. The HR department will be the owner of the database.

The annual growth of data is expected to be 20% for the next 5 years and 90% of the users are estimated to have read-only access. That's why the database technology and design chosen will be optimized for online analytical processing (OLAP). It's more important to ensure a stable and fast read performance than to focus on real-time writing. This means that the database will not be optimized for constant updates and inputs of data, but it provides the flexibility to extend it in future for analytic purposes, e.g. dashboards.

Data Architect Business Requirement

To establish an interface to the payroll department’s system in future, the database design needs to be extended, e.g. add further access rights.

The data will be retained for at least 7 years as required by federal regulations. As backup schedule a weekly full backup and an incremental daily backup will be ensured.

Data Architect Technical Requirement

Justification
A PostgreSQL database will be setup in order to fullfill the business requirements. It is a reasonable choice of technology for ensuring data security as stated in the IT department best practices guide. Additionally, the data integrity will be ensured considering the project annual growth by leveraging database transactions of a DBMS compared to the current shared Excel spreadsheet solution.

Database Objects and Data Governance
The following database objects and their access rights will be created:

Name	Type	User Access	Owner
Salary	Table	• Full access (CRUD) to all HR department and management employees • Any access by other employee will be denied	HR Depart-ment
Department	Table	• Full access (CRUD) to all HR department and management employees • Any other employee with a domain account will have **read-only access**	
Location	Table		
Education_Lvl	Table		
Job_Title	Table		
Employee_History			
Employee	Table		

Data Architect Technical Requirement

Data Ingestion
The data will be completely migrated from the spreadsheet flat file export provided to the database using ETL.

Scalability
The use of replications will be appropriate to ensure scalability, because a normalized (3NF) transactional database (OLTP) will be provided. There are likely more reading than writing interactions. No analytical use cases, e.g. dashboards are planned to be implemented. There is no need to have all copies of the original database (replicas) updated in realtime. Currently, 5 office locations with roughly 200 employees will access the database. 90% of the users are estimated to have read-only access. An annual growth of employees by 20% for the next 5 years is projected. This means approx. 500 users will have access to the database in 5 years from now.

Flexibility
The DDL SQL for normalized (3NF) PostgreSQL database creation and DML SQL for ETL leverages basic psql commands. No PostgreSQL specific features will be used that would limit the flexibility to migrate the schema to another RDBMS. The EMP_ID column in the spreadsheet, will be kept as unique reference, because we assume other systems, e.g. from the payroll department are using the same ID.

Data Architect Technical Requirement

Storage & Retention
The standard partition of 1 GB per server group stored on spinning HDDs will be suitable for the database considering the projected growth of number of employees inclusive fluctuation for the next 5 years. The data needs to be retained for at least 7 years as required by federal regulations.

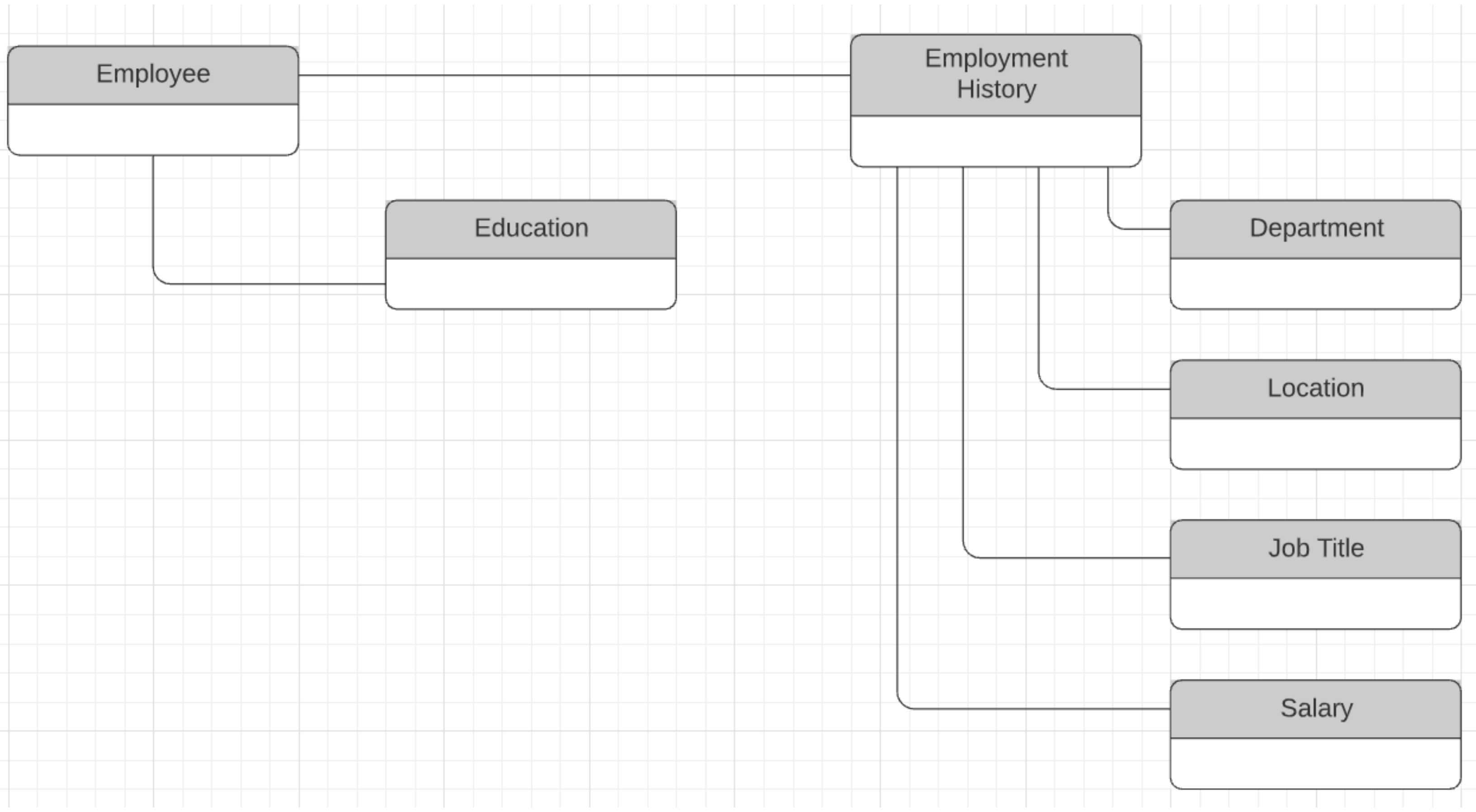
Backup
The data is considered as business critical. A backup schedule based on priority *Critical* is required (full backup 1x per week, incremental backup daily).

Step 2

Relational Database Design

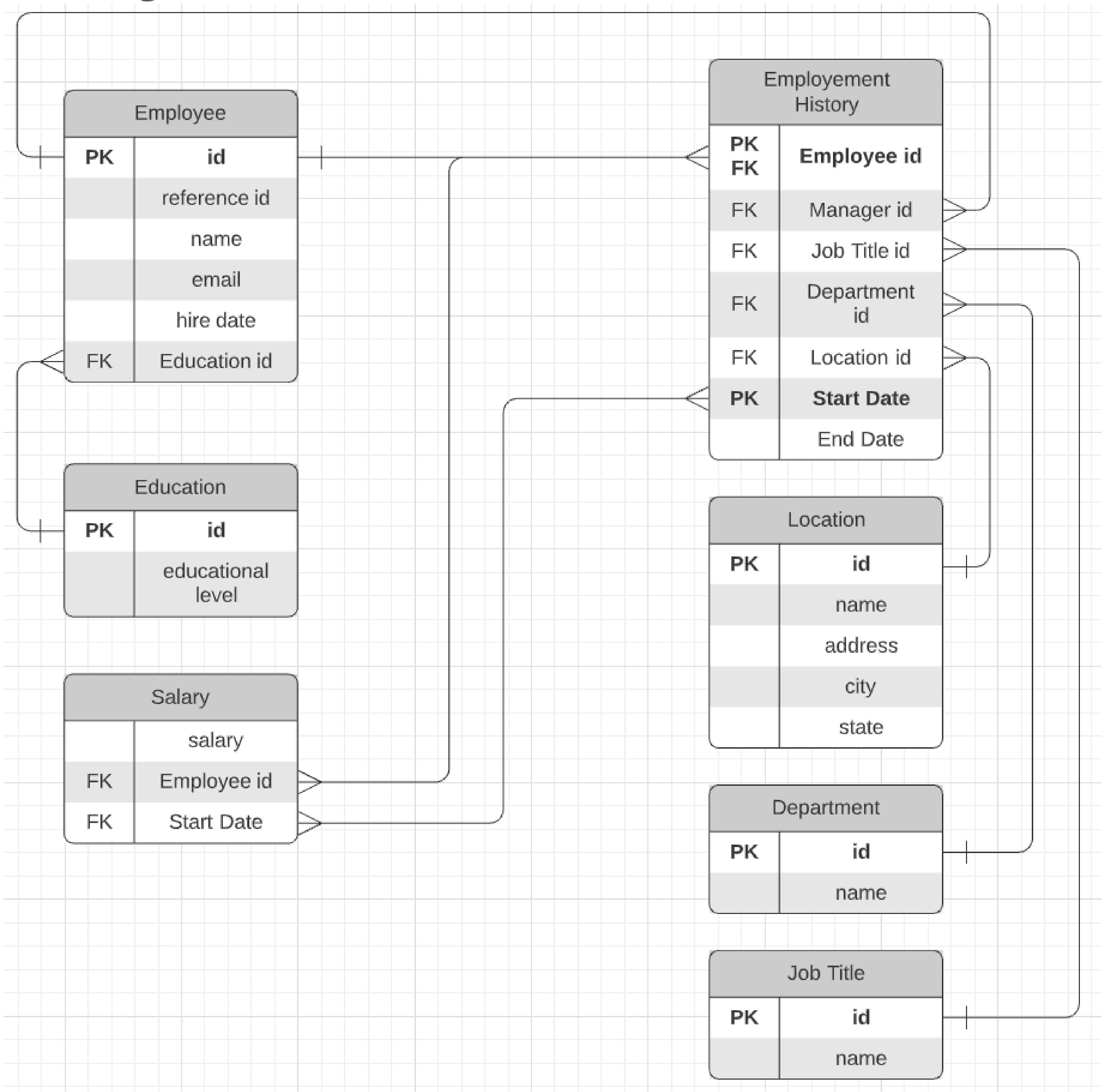
ERD

- **Conceptual**



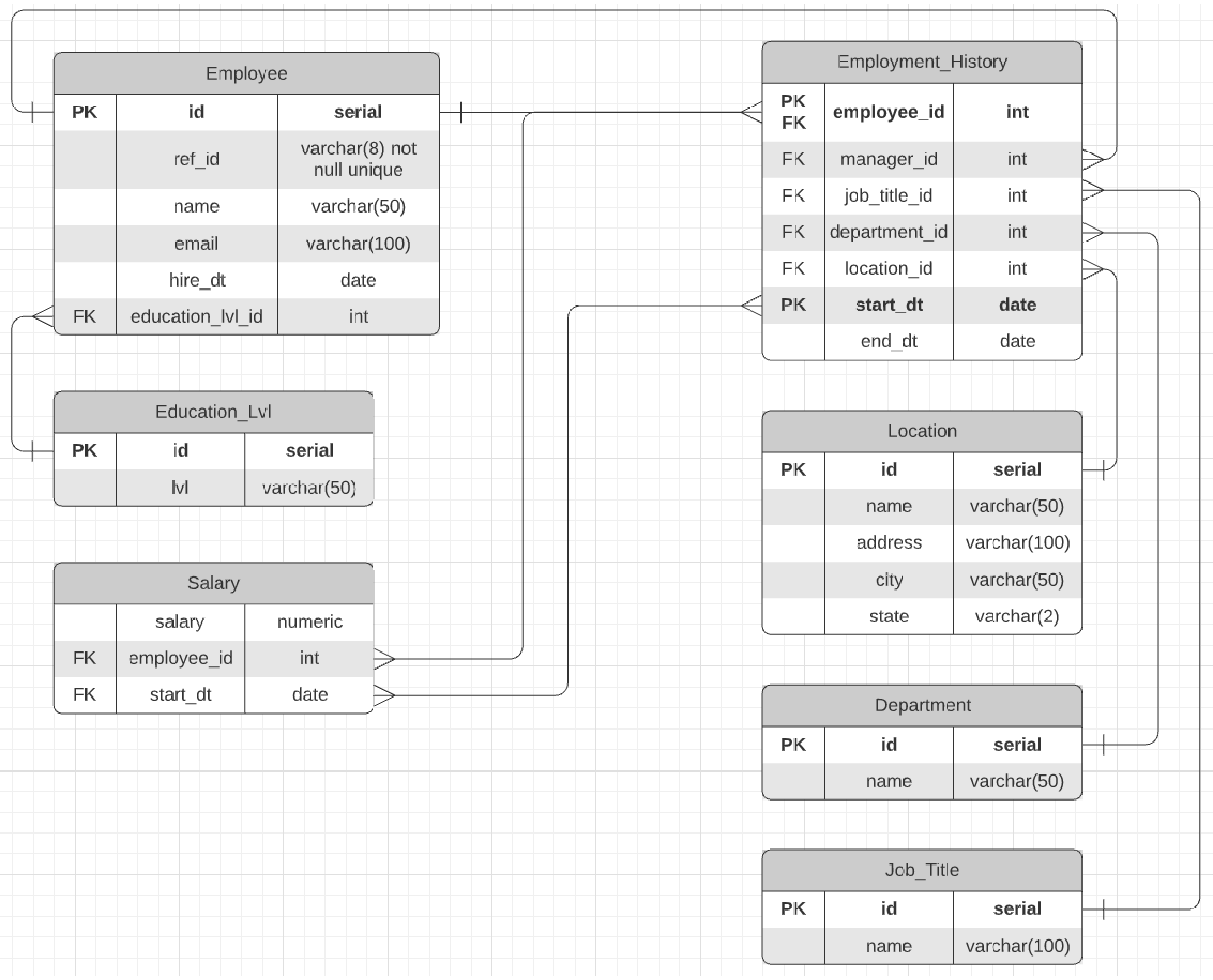
ERD

- **Logical**


Employee
PK id
reference id
name
email
hire date
FK Education id
Education
PK id
educational level
Salary
salary
FK Employee id
FK Start Date
Employement History
PK FK Employee id
FK Manager id
FK Job Title id
FK Department id
FK Location id
PK Start Date
End Date
Location
PK id
name
address
city
state
Department
PK id
name
Job Title
PK id
name

ERD

- **Physical**



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, ETL & CRUD

DDL
The DDL script is represented by the file *001-create_hr_db.sql*, which needs to be initially executed in order to create the empty noarmlized (3NF) database.

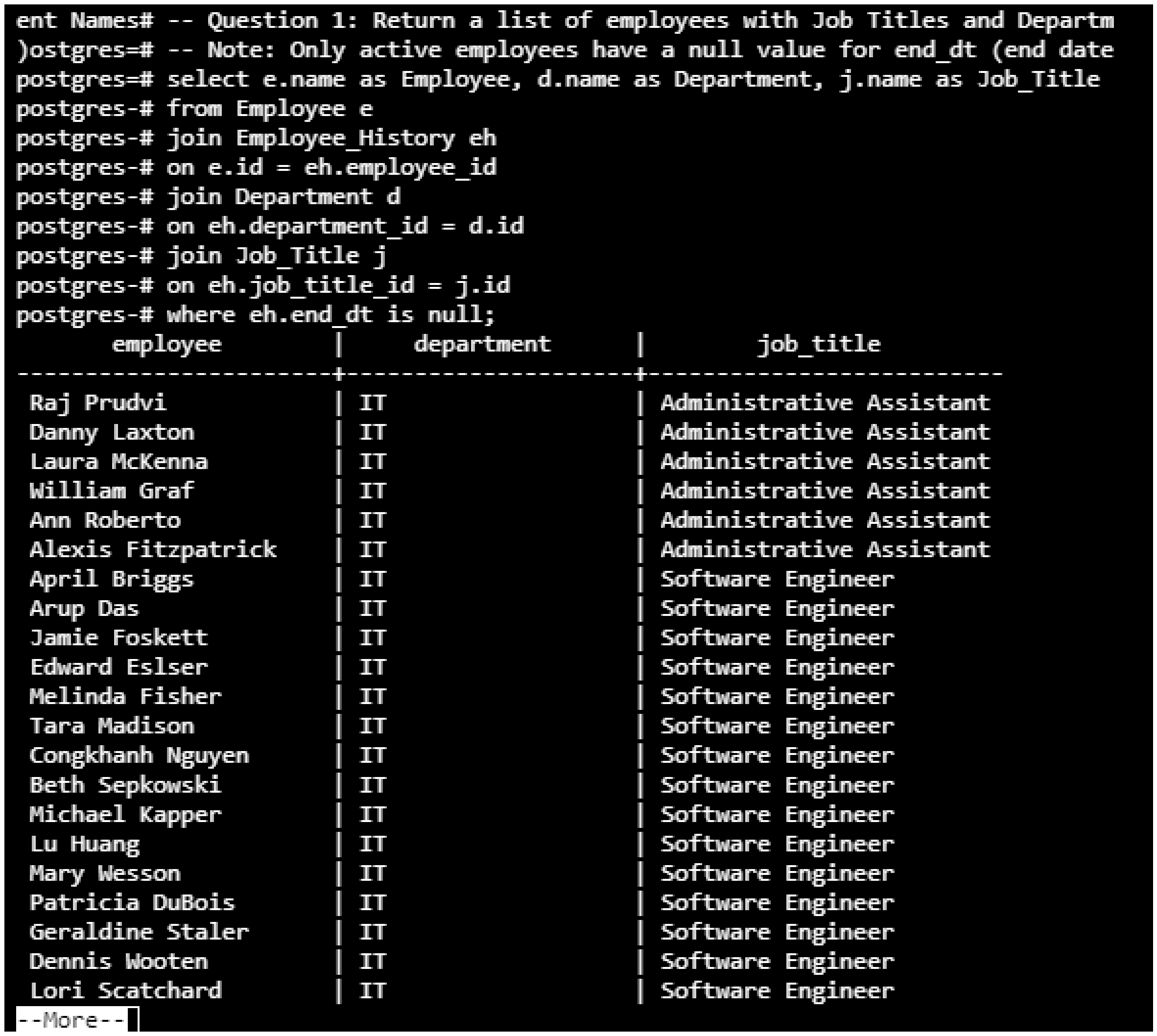
ETL
The data has been provided as Excel file (xlsx-Extension), which was converted to a csv-flat file *hr-dataset.csv*. The script *002-load_stage_table_from_csv.sql* creates a temporary staging table name *stg_table* and loads the data from the csv-flat file.

To populate the database from the staging table *stg_table* use the script *003-etl_stage_to_hr_db.sql*. It will print the data of all tables at the end.

CRUD
Finally, feel free to execute the script *004-crud.sql*, which was created to answer the 6 questions on the following slides.

CRUD

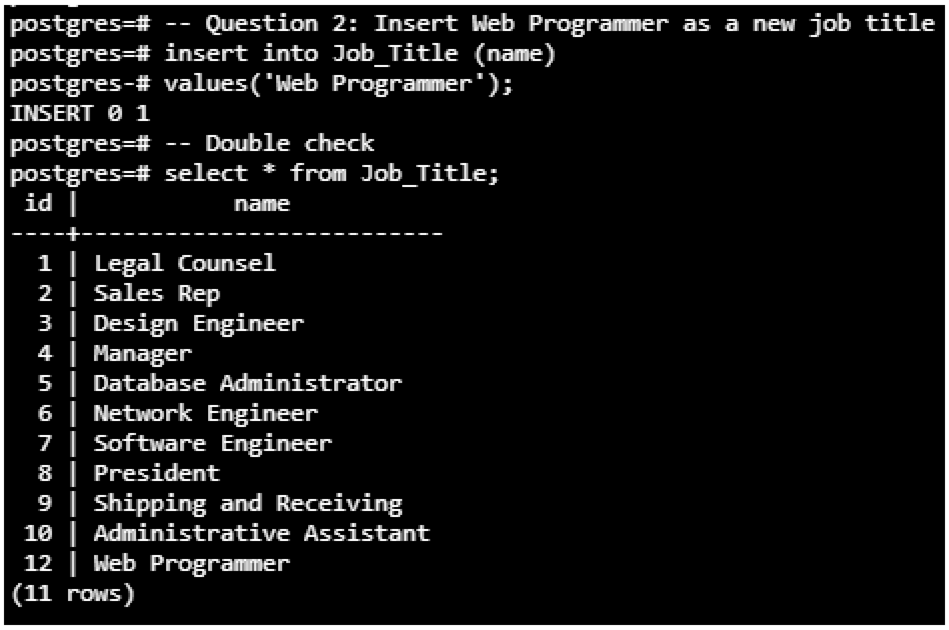
- **Question 1: Return a list of employees with Job Titles and Department Names**



```
ent Names# -- Question 1: Return a list of employees with Job Titles and Departm
)ostgres=# -- Note: Only active employees have a null value for end_dt (end date
postgres=# select e.name as Employee, d.name as Department, j.name as Job_Title
postgres-# from Employee e
postgres-# join Employee_History eh
postgres-# on e.id = eh.employee_id
postgres-# join Department d
postgres-# on eh.department_id = d.id
postgres-# join Job_Title j
postgres-# on eh.job_title_id = j.id
postgres-# where eh.end_dt is null;
       employee         |     department      |        job_title
------------------------+---------------------+--------------------------
 Raj Prudvi             | IT                  | Administrative Assistant
 Danny Laxton           | IT                  | Administrative Assistant
 Laura McKenna          | IT                  | Administrative Assistant
 William Graf           | IT                  | Administrative Assistant
 Ann Roberto            | IT                  | Administrative Assistant
 Alexis Fitzpatrick     | IT                  | Administrative Assistant
 April Briggs           | IT                  | Software Engineer
 Arup Das               | IT                  | Software Engineer
 Jamie Foskett          | IT                  | Software Engineer
 Edward Eslser          | IT                  | Software Engineer
 Melinda Fisher         | IT                  | Software Engineer
 Tara Madison           | IT                  | Software Engineer
 Congkhanh Nguyen       | IT                  | Software Engineer
 Beth Sepkowski         | IT                  | Software Engineer
 Michael Kapper         | IT                  | Software Engineer
 Lu Huang               | IT                  | Software Engineer
 Mary Wesson            | IT                  | Software Engineer
 Patricia DuBois        | IT                  | Software Engineer
 Geraldine Staler       | IT                  | Software Engineer
 Dennis Wooten          | IT                  | Software Engineer
 Lori Scatchard         | IT                  | Software Engineer
--More--
```

CRUD

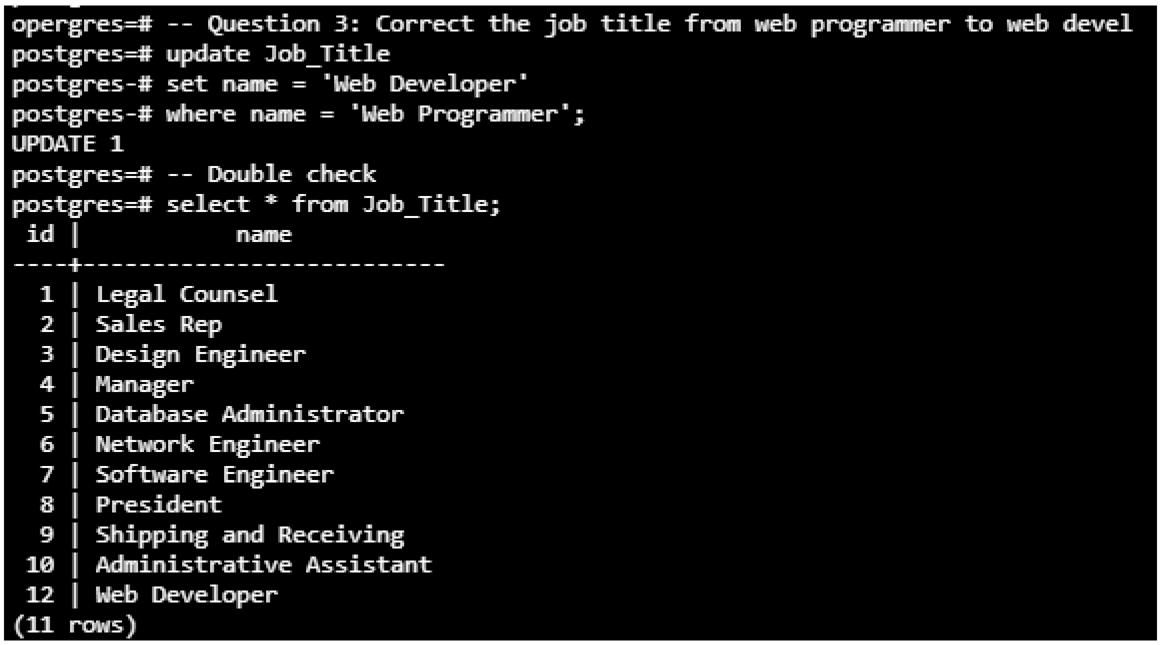
- **Question 2: Insert Web Programmer as a new job title**



```
postgres=# -- Question 2: Insert Web Programmer as a new job title
postgres=# insert into Job_Title (name)
postgres-# values('Web Programmer');
INSERT 0 1
postgres=# -- Double check
postgres=# select * from Job_Title;
 id |           name
----+--------------------------
  1 | Legal Counsel
  2 | Sales Rep
  3 | Design Engineer
  4 | Manager
  5 | Database Administrator
  6 | Network Engineer
  7 | Software Engineer
  8 | President
  9 | Shipping and Receiving
 10 | Administrative Assistant
 12 | Web Programmer
(11 rows)
```

CRUD

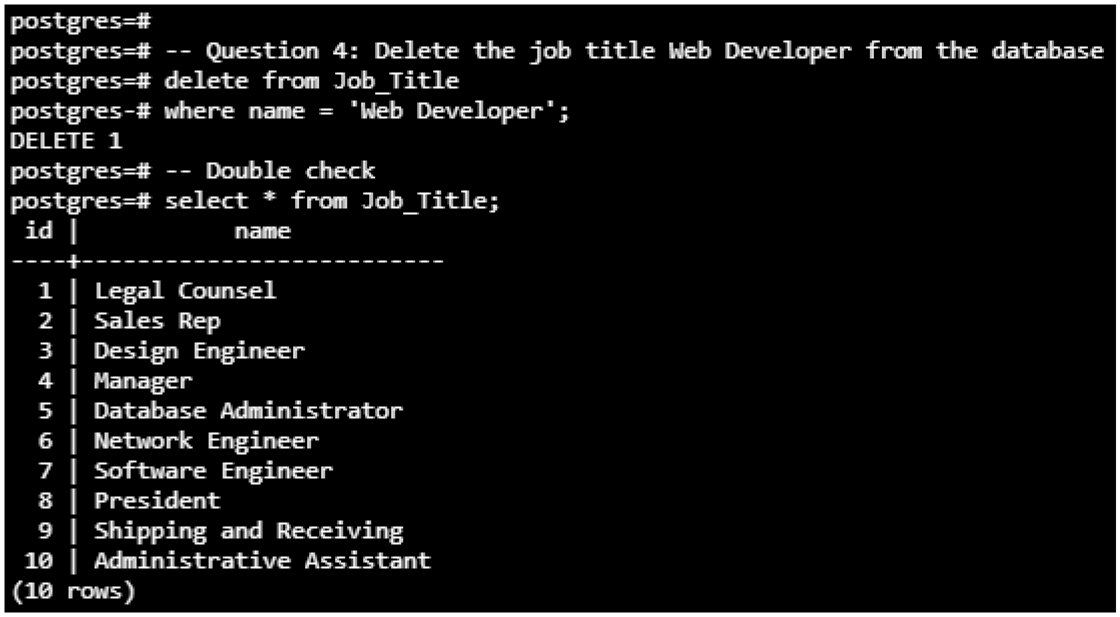
- **Question 3: Correct the job title from web programmer to web developer**



```
opergres=# -- Question 3: Correct the job title from web programmer to web devel
postgres=# update Job_Title
postgres-# set name = 'Web Developer'
postgres-# where name = 'Web Programmer';
UPDATE 1
postgres=# -- Double check
postgres=# select * from Job_Title;
 id |           name
----+-------------------------
  1 | Legal Counsel
  2 | Sales Rep
  3 | Design Engineer
  4 | Manager
  5 | Database Administrator
  6 | Network Engineer
  7 | Software Engineer
  8 | President
  9 | Shipping and Receiving
 10 | Administrative Assistant
 12 | Web Developer
(11 rows)
```

CRUD

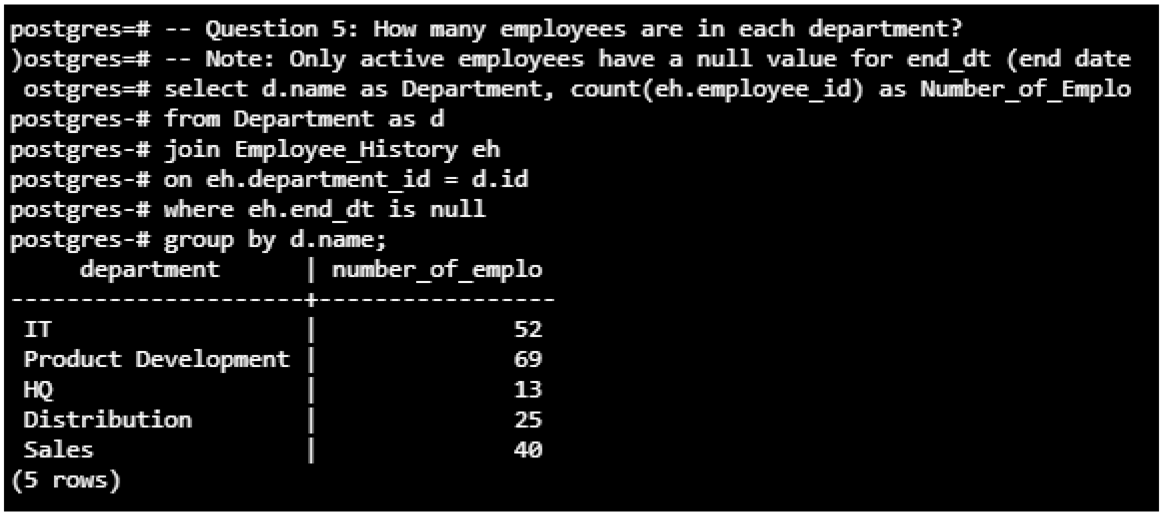
- **Question 4: Delete the job title Web Developer from the database**



```
postgres=#
postgres=# -- Question 4: Delete the job title Web Developer from the database
postgres=# delete from Job_Title
postgres-# where name = 'Web Developer';
DELETE 1
postgres=# -- Double check
postgres=# select * from Job_Title;
 id |           name
----+--------------------------
  1 | Legal Counsel
  2 | Sales Rep
  3 | Design Engineer
  4 | Manager
  5 | Database Administrator
  6 | Network Engineer
  7 | Software Engineer
  8 | President
  9 | Shipping and Receiving
 10 | Administrative Assistant
(10 rows)
```

CRUD

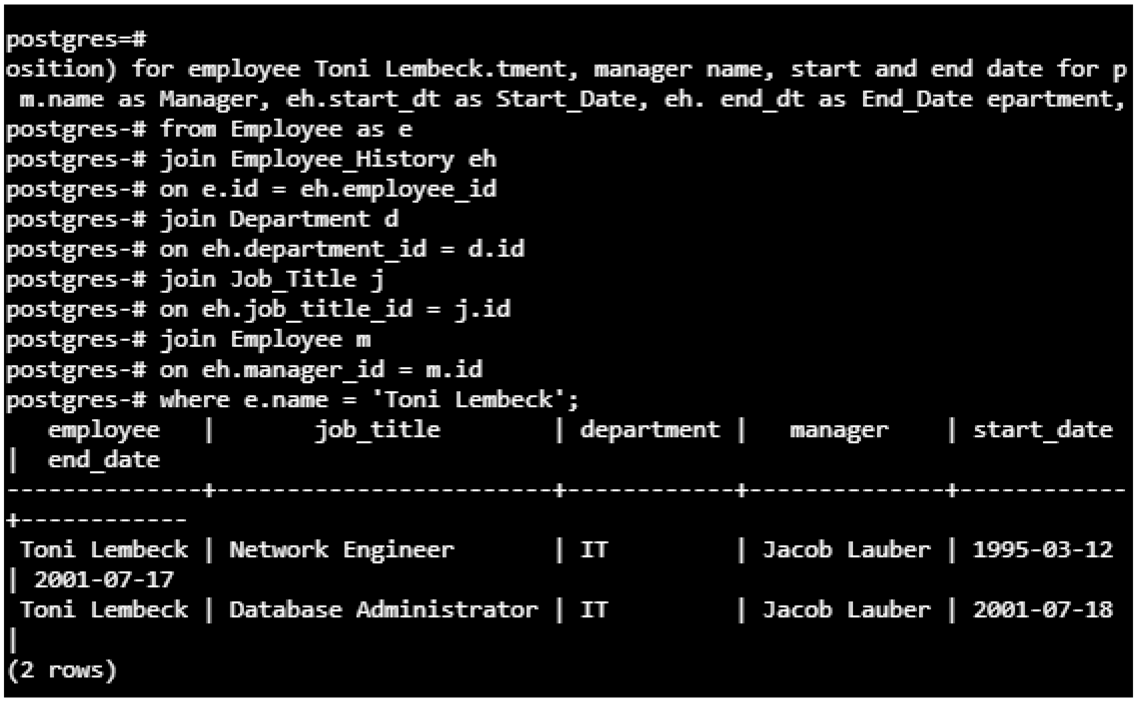
- **Question 5: How many employees are in each department?**



```
postgres=# -- Question 5: How many employees are in each department?
)ostgres=# -- Note: Only active employees have a null value for end_dt (end date
 ostgres=# select d.name as Department, count(eh.employee_id) as Number_of_Emplo
postgres-# from Department as d
postgres-# join Employee_History eh
postgres-# on eh.department_id = d.id
postgres-# where eh.end_dt is null
postgres-# group by d.name;
     department      | number_of_emplo
---------------------+-----------------
 IT                  |              52
 Product Development |              69
 HQ                  |              13
 Distribution        |              25
 Sales               |              40
(5 rows)
```

CRUD

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



```
postgres=#
osition) for employee Toni Lembeck.tment, manager name, start and end date for p
 m.name as Manager, eh.start_dt as Start_Date, eh. end_dt as End_Date epartment,
postgres-# from Employee as e
postgres-# join Employee_History eh
postgres-# on e.id = eh.employee_id
postgres-# join Department d
postgres-# on eh.department_id = d.id
postgres-# join Job_Title j
postgres-# on eh.job_title_id = j.id
postgres-# join Employee m
postgres-# on eh.manager_id = m.id
postgres-# where e.name = 'Toni Lembeck';
   employee   |       job_title        | department |   manager    | start_date
|  end_date
--------------+------------------------+------------+--------------+------------
+------------
 Toni Lembeck | Network Engineer       | IT         | Jacob Lauber | 1995-03-12
| 2001-07-17
 Toni Lembeck | Database Administrator | IT         | Jacob Lauber | 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.**

Based on the Tech ABC Corp IT Best Practices Guide, the database security is based at the user level. Each employee in the company has a domain authenticated username that they will use to access any database they have been authorized access to.

When creating the empty database schema, initially all users will be granted access to all tables. Before loading the data to the tables, the access for all non HR Department and Management employee users will be revoked for the table *Salary*.