

HR Database



Business Scenario

Business requirement

The video game company saw explosive growth with a sudden appearance onto the gaming scene with its new AI-powered video game console. As a result, they went from a small 10 person operation to 200 employees and 5 locations in under a year. HR was having trouble keeping up with the growth since they were still maintaining employee information in a spreadsheet. While that worked for ten employees, it became increasingly cumbersome to manage as the company expands. As such, the HR department requested a database capable of managing their employee information.

Dataset

The [HR dataset](#) is an Excel workbook consisting of 206 records, with eleven columns. The data is in human-readable format and was not normalized at all. The data lists the names of employees, 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, as detailed in the [Best Practices document](#).

Step 1

Business and Technical
Requirement

Database Request

Hi,

As you may already know, we have 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 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,
Head of HR

Business Requirement

- **Purpose of the new database:**

HR department wants to keep up with the growth of the company and ensure data integrity and security of the employee records.

- **Describe current data management solution:**

Currently, all employee records are maintained on a shared Excel spreadsheet.

- **Describe current data available:**

The current data consists of 206 records, with eleven columns, including the names of employees, job title, department, manager's name, hire date, start date, end date, work location, and salary.

- **Who will own/manage data**

HR department will own and maintain the data.

- **Who will have access to database**

Any employee with a domain login has read only access to the database. Management and HR employees also have write access.

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

Salary information is restricted to management and HR employees.

Business Requirement

- **Estimated size of database**

About 200 rows.

- **Estimated annual growth**

For the next five years, HR projected a 20% annual growth. There are about 200 employees now. The projected growth will lead to the size of 500 employees at the end of year five.

- **Additional data requests:**

There is a request to connect the HR database with the payroll database that maintains employee attendance and paid time off information.

Technical Requirement

- **Justification for the new database**

The shared spreadsheet is not sufficient to keep up with the growth of the company. It also imposes data integrity and data security risks to the company. The new database will alleviate the strain of record keeping and improve the data integrity and data security.

- **Database objects**

There are eight tables in the HR database, including Employee, Education, Job, Department, Location, Address, Salary, and Employee History.

- **Data ingestion**

The data ingestion method will be ETL.

Technical Requirement

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

Ownership: HR will own and maintain the data.

User Access: Any employee with a domain login has read only access to the database. Management and HR employees also have write access. Salary information is restricted to management and HR employees.

- **Scalability**

Replication can be implemented if reporting or other reading needs arise. Sharding can be implemented to satisfy increased writing needs.

- **Flexibility**

Define and document integration priorities and frequency.

Monitor the integration process and put various check points

- **Storage & retention**

Storage: 1GB partition on spinning disk

Retention: at least 7 years required by federal regulations

- **Backup**

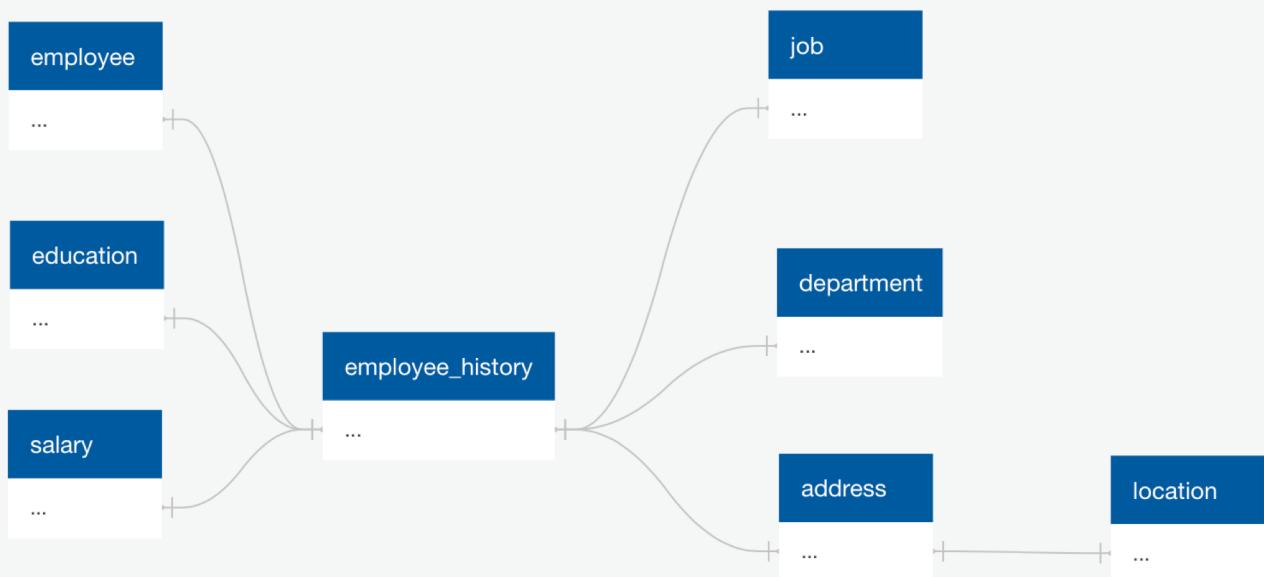
The data is considered business critical. Backup schedule is full backup once per week and incremental backup daily.

Step 2

Relational Database
Design

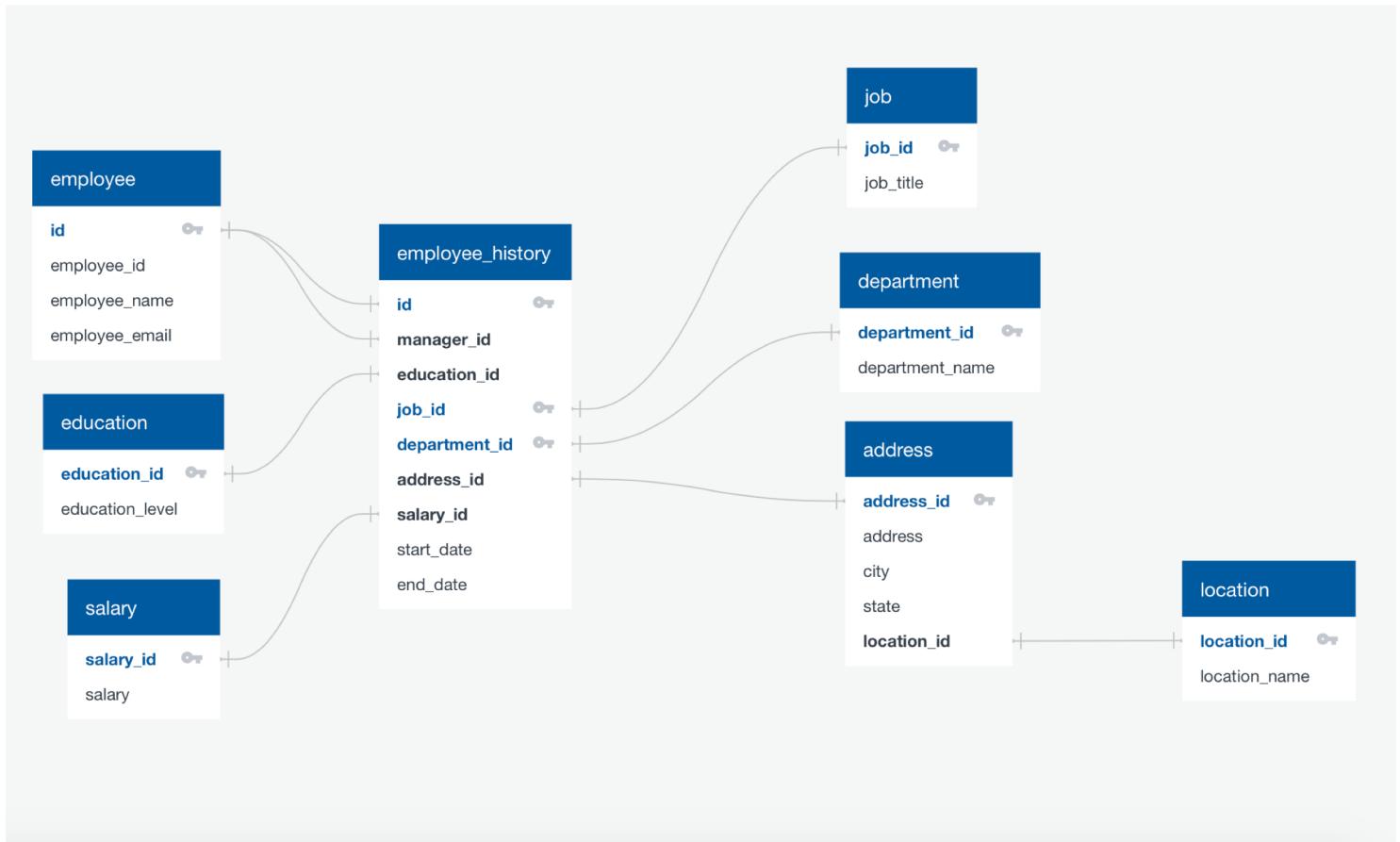
ERD

- **Conceptual**



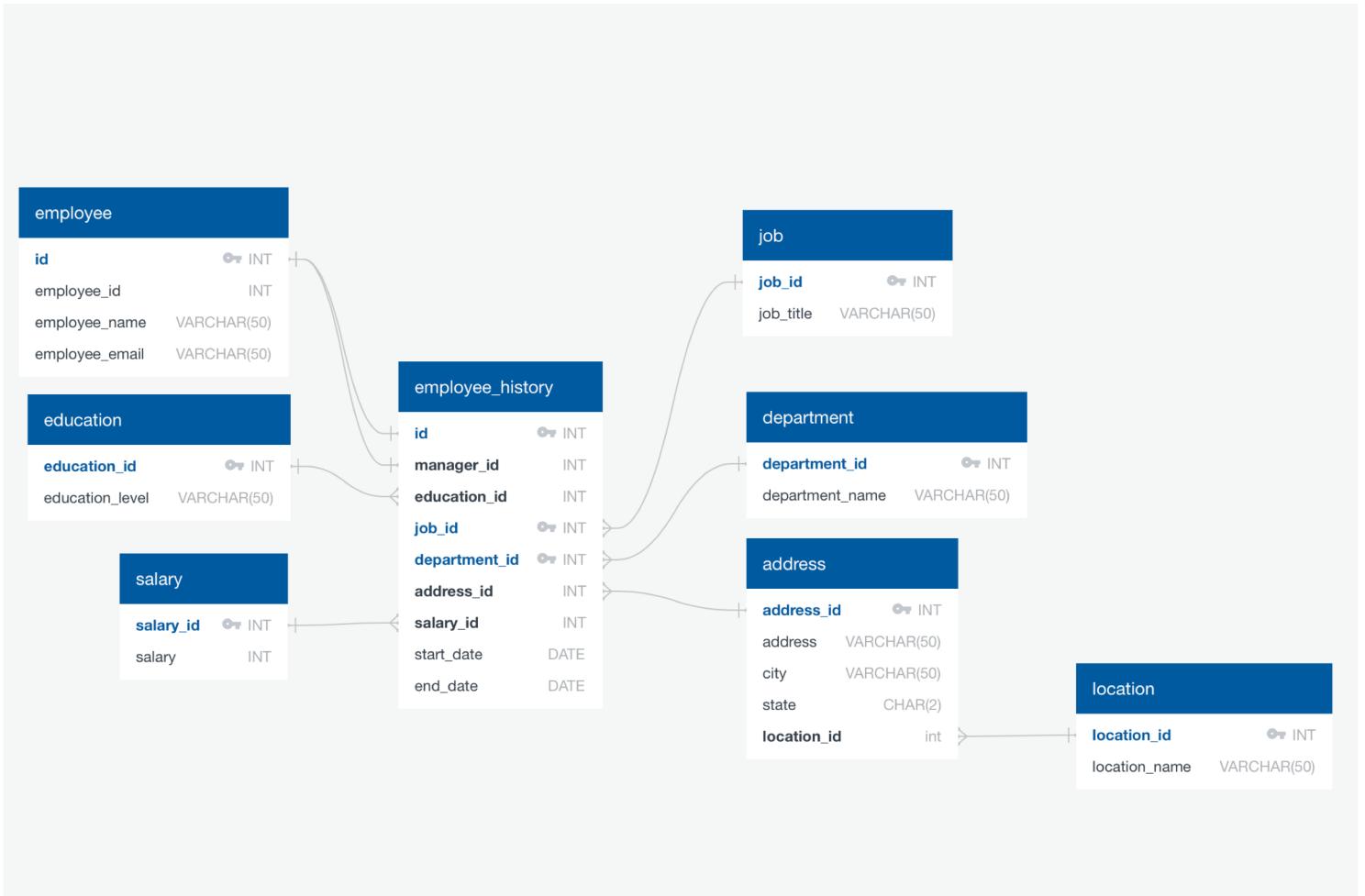
ERD

- Logical



ERD

- Physical

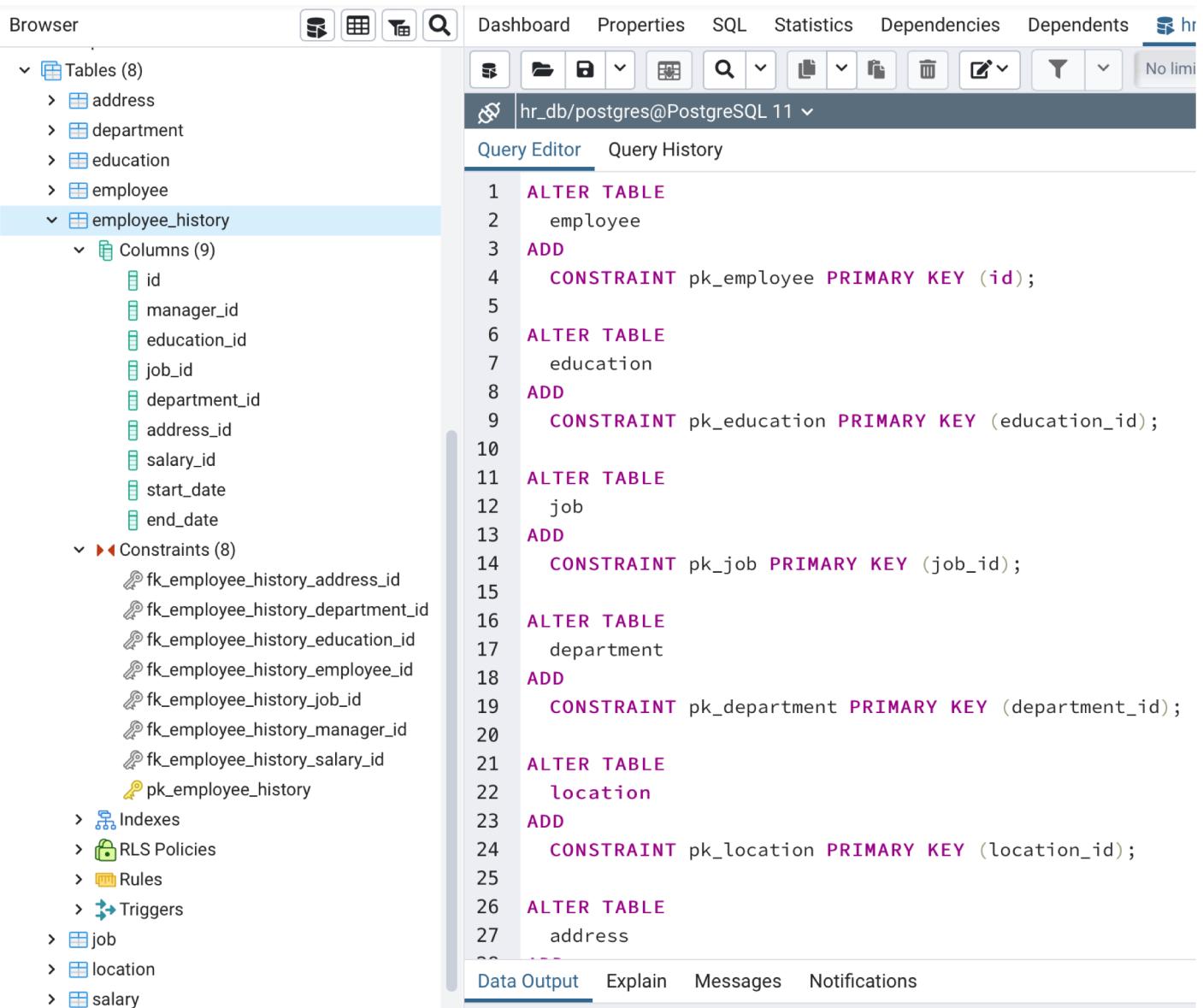


Step 3

Create A Physical
Database

DDL

I used [python](#) for the ETL process and [SQL](#) script to add primary key and secondary key constraints.



The screenshot shows the pgAdmin 4 interface. The left pane is the 'Browser' pane, which displays a tree view of database objects. The 'Tables' node is expanded, showing eight tables: address, department, education, employee, and employee_history (which is selected). Under employee_history, there are nine columns: id, manager_id, education_id, job_id, department_id, address_id, salary_id, start_date, and end_date. Below these are eight foreign key constraints and one primary key constraint (pk_employee_history). The right pane is the 'Query Editor' pane, showing a SQL script with numbered lines. The script adds primary keys to five tables: employee, education, job, department, and location, and a secondary key constraint to the address table.

```
1 ALTER TABLE
2   employee
3 ADD
4   CONSTRAINT pk_employee PRIMARY KEY (id);
5
6 ALTER TABLE
7   education
8 ADD
9   CONSTRAINT pk_education PRIMARY KEY (education_id);
10
11 ALTER TABLE
12   job
13 ADD
14   CONSTRAINT pk_job PRIMARY KEY (job_id);
15
16 ALTER TABLE
17   department
18 ADD
19   CONSTRAINT pk_department PRIMARY KEY (department_id);
20
21 ALTER TABLE
22   location
23 ADD
24   CONSTRAINT pk_location PRIMARY KEY (location_id);
25
26 ALTER TABLE
27   address
```

CRUD

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

hr_db/postgres@PostgreSQL 11 ▾

Query Editor Query History

```
1 select
2   employee_name,
3   job,
4   department
5 from
6   employee_history h
7   join department d on h.department_id = d.department_id
8   join job j on h.job_id = j.job_id
9   join employee e on h.id = e.id
10 order by
11   employee_name
```

Data Output Explain Messages Notifications

	employee_name text	job text	department text	
1	Aaron Gordon	Network E...	Product Developm...	
2	Aaron Richman	Administr...	Product Developm...	
3	Abby Lockhart	Database ...	IT	
4	Abby Lockhart	Network E...	IT	
5	Alan Mecklet	Administr...	Product Developm...	
6	Alejandro Scannapieco	Sales Rep	Sales	
7	Alex Warring	Shipping ...	Distribution	
8	Alexis Fitzpatrick	Administr...	IT	
9	Allison Gentle	Manager	Distribution	
10	Amit Hardiya	Administr...	Sales	
11	Analyn Braza	Sales Rep	Sales	
12	Andrew Yoon	Sales Rep	Product Developm...	
13	Angela Masdary	Network E...	IT	
14	Anil Padala	Software ...	Product Developm...	
15	Anita Deluise	Administr...	HQ	

CRUD

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

The screenshot shows a PostgreSQL query editor interface. At the top, it displays the connection information: **hr_db/postgres@PostgreSQL 11**. Below the connection bar, there are two tabs: **Query Editor** (which is currently selected) and **Query History**. The main area contains the following SQL code:

```
1 INSERT INTO job (job_id, job)
2 VALUES
3     (10, 'Web Programmer');
4
5 SELECT
6     *
7 FROM
8     job;
9
```

Below the code, there are four tabs: **Data Output** (selected), **Explain**, **Messages**, and **Notifications**. The **Data Output** tab displays a table with the following data:

	job_id [PK] bigint	job text
1	0	Administrative Assi...
2	1	Database Administ...
3	2	Design Engineer
4	3	Legal Counsel
5	4	Manager
6	5	Network Engineer
7	6	President
8	7	Sales Rep
9	8	Shipping and Recei...
10	9	Software Engineer
11	10	Web Programmer

CRUD

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

The screenshot shows a PostgreSQL query editor interface. The top bar indicates the connection is to 'hr_db/postgres@PostgreSQL 11'. The main area displays a numbered SQL script:

```
1 UPDATE
2   job
3 SET
4   job = 'Web Developer'
5 WHERE
6   job_id = 10;
7
8 SELECT
9   *
10 FROM
11   job;
```

Below the script, there are tabs for 'Data Output', 'Explain', 'Messages', and 'Notifications'. The 'Data Output' tab is selected, showing a table with two columns: 'job_id' and 'job'. The data rows are:

	job_id	job
1	0	Administrative Assista...
2	1	Database Administrator
3	2	Design Engineer
4	3	Legal Counsel
5	4	Manager
6	5	Network Engineer
7	6	President
8	7	Sales Rep
9	8	Shipping and Receiving
10	9	Software Engineer
11	10	Web Developer

CRUD

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

The screenshot shows a PostgreSQL query editor interface. At the top, it displays the connection information: **hr_db/postgres@PostgreSQL 11**. Below the connection bar, there are two tabs: **Query Editor** (which is currently selected) and **Query History**. The main area contains the following SQL code:

```
1  DELETE FROM
2    job
3 WHERE
4   job_id = 10;
5 SELECT
6   *
7 FROM
8   job;
```

Below the code, there are four tabs: **Data Output** (selected), **Explain**, **Messages**, and **Notifications**. The **Data Output** tab displays a table with the following data:

	job_id [PK] bigint	job text
1	0	Administrative Assistant
2	1	Database Administrator
3	2	Design Engineer
4	3	Legal Counsel
5	4	Manager
6	5	Network Engineer
7	6	President
8	7	Sales Rep
9	8	Shipping and Receiving
10	9	Software Engineer

CRUD

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

The screenshot shows a PostgreSQL query editor interface. The top bar displays the connection information: **hr_db/postgres@PostgreSQL 11**. Below the bar, there are two tabs: **Query Editor** (which is selected) and **Query History**. The main area contains the SQL code for the query:

```
1 SELECT
2     department,
3     COUNT(DISTINCT id) as employee_count
4 FROM
5     employee_history h
6     JOIN department d ON h.department_id = d.department_id
7 WHERE
8     end_date IS NULL
9 GROUP BY
10    department
11 ORDER BY
12    employee_count DESC
```

Below the code, there are four navigation tabs: **Data Output** (selected), **Explain**, **Messages**, and **Notifications**. The **Data Output** tab displays the results of the query in a table:

	department	employee_count
1	Product Development	69
2	IT	52
3	Sales	40
4	Distribution	25
5	HQ	13

CRUD

- **Query 6: Returns current and past jobs (include employee name, job title, department, manager name, start and end date for position) for employee Toni Lembeck.**

The screenshot shows a PostgreSQL query editor interface. The top bar displays the connection information: hr_db/postgres@PostgreSQL 11. Below the bar, there are tabs for 'Query Editor' (which is active) and 'Query History'. The main area contains a numbered SQL query:

```
1  SELECT
2      e1.employee_name,
3      j.job,
4      d.department,
5      e2.employee_name as manager_name,
6      h.start_date,
7      h.end_date
8  FROM
9      employee_history h
10     JOIN department d ON h.department_id = d.department_id
11     JOIN job j ON h.job_id = j.job_id
12     JOIN employee e1 ON h.id = e1.id
13     JOIN employee e2 ON h.manager_id = e2.id
14 WHERE
15     e1.employee_name = 'Toni Lembeck'
```

Below the query, there are tabs for 'Data Output', 'Explain', 'Messages', and 'Notifications'. The 'Data Output' tab is active, showing a table with the results of the query:

	employee_name	job	department	manager_name	start_date	end_date
1	Toni Lembeck	Databa...	IT	Jacob Lauber	2001-07-18	[null]
2	Toni Lembeck	Networ...	IT	Jacob Lauber	1995-03-12	2001-07-17

CRUD

- **Describe how to apply table security to restrict access to employee salaries using an SQL server.**

Grant employees with a domain login read-only access to all tables in the HR Database and revoke access to the salary table for the employees who are not management or HR.

Step 4

Additional Features

Additional Feature 1

Create a view that returns all employee attributes.

The screenshot shows the pgAdmin interface for PostgreSQL 11. On the left, the object browser lists various database objects: FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Procedures, Sequences, Tables (8), Trigger Functions, Types, Views (2), address.location, and all_attributes. The all_attributes view is selected, revealing its 13 columns: employee_id, employee_name, start_date, end_date, education, job, department, manager_name, salary, address, city, state, and location. The main area is the Query Editor, which contains the SQL code for creating the view:

```
13 CREATE VIEW all_attributes AS
14 SELECT
15   e1.employee_id,
16   e1.employee_name,
17   h.start_date,
18   h.end_date,
19   ed.education,
20   j.job,
21   d.department,
22   e2.employee_name AS manager_name,
23   s.salary,
24   a.address,
25   a.city,
26   a.state,
27   a.location
28 FROM
29   employee_history h
30   JOIN department d ON h.department_id = d.department_id
31   JOIN job j ON h.job_id = j.job_id
32   JOIN employee e1 ON h.id = e1.id
33   JOIN employee e2 ON h.manager_id = e2.id
34   JOIN education ed ON h.education_id = ed.education_id
35   JOIN address_location a ON h.address_id = a.address_id
36   JOIN salary s ON h.salary_id = s.salary_id;
```

Below the query editor are tabs for Data Output, Explain, Messages, and Notifications. The Data Output tab is active, displaying the results of the query:

	employee_id	employee_name	start_date	end_date	education	job	department	manager_name	salary	address	city
1	E17469	Haifa Hajiri	2003-12-17	[null]	No College	Admini...	Distribution	Allison Gentle	47418	705 James ...	San Fra
2	E53895	Kumar Durairaj	2014-10-27	[null]	No College	Shippin...	Distribution	Allison Gentle	28700	705 James ...	San Fra
3	E57987	Nicole Lee	2016-05-15	[null]	No College	Admini...	Product Develop...	Conner Kinch	49786	705 James ...	San Fra

Appendix