**# SQL Homework - Employee Database: A Mystery in Two Parts**

**## Background**

It is a beautiful spring day, and it is two weeks since you have been hired as a new data engineer at Pewlett Hackard. Your first major task is a research project on employees of the corporation from the 1980s and 1990s. All that remain of the database of employees from that period are six CSV files.

In this assignment, you will design the tables to hold data in the CSVs, import the CSVs into a SQL database, and answer questions about the data. In other words, you will perform:

1. Data Modeling

2. Data Engineering

3. Data Analysis

**### Before You Begin**

🗹 1. Create a new repository for this project called `sql-challenge`. \*\*Do not add this homework to an existing repository\*\*.

🗹 2. Clone the new repository to your computer.

🗹 3. Inside your local git repository, create a directory for the SQL challenge. Use a folder name to correspond to the challenge: \*\*EmployeeSQL\*\*.

🗹 4. Add your files to this folder.

🗹 5. Push the above changes to GitHub.

**## Instructions**

**#### Data Modeling**

Inspect the CSVs and sketch out an ERD of the tables. Feel free to use a tool like [http://www.quickdatabasediagrams.com] (http://www.quickdatabasediagrams.com).

**#### Data Engineering**

🗹 Use the information you have to create a table schema for each of the six CSV files. Remember to specify data types, primary keys, foreign keys, and other constraints.

🗹 Import each CSV file into the corresponding SQL table.

**#### Data Analysis**

**/\*\* schemata? Try gestaltzerfall! \*\*/**

*/\*\* seriously. Look it up! It's the best symptom I've seen in months. \*\*/*

**CREATE TABLE** departments (dept\_no VARCHAR **PRIMARY KEY**, dept\_name VARCHAR );

**CREATE TABLE** employees (emp\_no INTEGER **PRIMARY KEY**, birth\_date DATE, first\_name VARCHAR, last\_name VARCHAR,

gender VARCHAR, hire\_date DATE);

**CREATE TABLE** dept\_emp (emp\_no INTEGER, dept\_no VARCHAR, from\_date DATE, to\_date DATE,

**FOREIGN KEY** (emp\_no) **REFERENCES** employees (emp\_no),

**FOREIGN KEY** (dept\_no) **REFERENCES** departments (dept\_no));

**CREATE TABLE** dept\_manager (dept\_no VARCHAR, emp\_no INTEGER, from\_date DATE, to\_date DATE,

**FOREIGN KEY** (dept\_no) **REFERENCES** departments (dept\_no));

**CREATE TABLE** salaries (emp\_no INTEGER, salary INTEGER, from\_date DATE, to\_date DATE,

**FOREIGN KEY** (emp\_no) **REFERENCES** employees (emp\_no));

**CREATE TABLE** titles (emp\_no INTEGER, title VARCHAR, from\_date DATE, to\_date DATE,

**FOREIGN KEY** (emp\_no) **REFERENCES** employees (emp\_no));

/\*\* Obviously, I know how to produce output as a list. To me that's bad form. I come from a world where we have data reporting cycles, a cleaning period, and use static/audited datasets for analysis. There is a query that is out to table, the table is manually renamed, put on another machine, and privileges are locked. Our cleaning goes into a delta file: we don't ever touch the original dB because of audits. …. I was a 'quant jock' and I'm taking this course to get back to that. I've never in my life worked where we did updated as part of analysis. It seems unnatural to me. My syntax is all going to be query-output-table because that's how they do things. \*\*/

Once you have a complete database, do the following:

1. List the following details of each employee: employee number, last name, first name, gender, and salary.

*SELECT emp\_no, birth\_date, first\_name, last\_name, gender, hire\_date FROM employees*

**SELECT** emp\_no,

birth\_date,

first\_name,

last\_name,

gender

**FROM** employees;

1. List employees who were hired in 1986.

*SELECT emp\_no, first\_name, last\_name, hire\_date FROM employees WHERE date\_part('year', hire\_date) = 1986;*

**SELECT** emp\_no,

first\_name,

last\_name,

hire\_date

**FROM** employees

**WHERE** date\_part('year', hire\_date) = 1986;

1. List the manager of each department with the following information: department number, department name, the manager's employee number, last name, first name, and start and end employment dates.

CREATE TABLE stage3a as

SELECT d.dept\_name, d.dept\_no, dm.emp\_no,

dm.from\_date as mgr\_from\_date,

dm.to\_date as mgr\_to\_date

FROM dept\_manager dm inner join departments d

ON dm.dept\_no = d.dept\_no

WHERE date\_part('year', dm.to\_date) = 9999;

CREATE TABLE stage3 as

SELECT s1.dept\_no, s1.dept\_name, s1.emp\_no,

e.last\_name, e.first\_name, e.hire\_date , s1.mgr\_to\_date

FROM stage1 s1 inner join employees e

ON s1.emp\_no = e.emp\_no;

DROP TABLE stage3a;

*/\*\* to just list them –* ***produce display with no saved output*** *– do this:*

CREATE TABLE stage3a as

SELECT d.dept\_name, d.dept\_no, dm.emp\_no,

dm.from\_date as mgr\_from\_date,

dm.to\_date as mgr\_to\_date

FROM dept\_manager dm inner join departments d

ON dm.dept\_no = d.dept\_no

WHERE date\_part('year', dm.to\_date) = 9999;

SELECT s1.dept\_no, s1.dept\_name, s1.emp\_no,

e.last\_name, e.first\_name, e.hire\_date , s1.mgr\_to\_date

FROM stage1 s1 inner join employees e

ON s1.emp\_no = e.emp\_no;

\*\*/

**NOTE:** The question asks for “the manager” – interpreted to mean the current manager. If you want previous managers – plural – then drop the WHERE statement on stage3a and include a stage3b table that draws data from titles, selecting all rows. That gives you j copies of all managers where j is instances of title for individual managers (max j = 2) . To get to one copy, end query with WHERE title = ‘manager’

1. List the department of each employee with the following information: employee number, last name, first name, and department name.

CREATE TABLE stage4a as

SELECT e.emp\_no, e.last\_name, e.first\_name,

de.dept\_no, de.from\_date, de.to\_date

FROM employees e INNER JOIN dept\_emp de

ON e.emp\_no = de.emp\_no

ORDER BY e.emp\_no ASC,

de.to\_date DESC;

CREATE TABLE stage4 as

SELECT DISTINCT ON (s4.emp\_no)

s4.emp\_no, s4.last\_name, s4.first\_name,

d.dept\_name

FROM stage4\_1 s4 INNER JOIN departments d

ON s4.dept\_no = d.dept\_no;

DROP TABLE stage4a;

*/\*\* to just list them –* ***produce display with no saved output*** *– do this:*

CREATE TABLE stage4a as

SELECT e.emp\_no, e.last\_name, e.first\_name,

de.dept\_no, de.from\_date, de.to\_date

FROM employees e INNER JOIN dept\_emp de

ON e.emp\_no = de.emp\_no

ORDER BY e.emp\_no ASC,

de.to\_date DESC;

SELECT DISTINCT ON (s4.emp\_no)

s4.emp\_no, s4.last\_name, s4.first\_name,

d.dept\_name

FROM stage4\_1 s4 INNER JOIN departments d

ON s4.dept\_no = d.dept\_no;

Look, obviously I know how to produce output as a list. To me that's bad form. I come from a world where we have data reporting cycles, a cleaning period, and use static/audited datasets for analysis. \*\*/

1. List all employees whose first name is "Hercules" and last names begin with "B."

Select \* from employees where first\_name = 'Hercules' AND LEFT(last\_name, 1) = 'B';

1. List all employees in the Sales department, including their employee number, last name, first name, and department name.

Select

CREATE TABLE stage6a as

SELECT e.emp\_no, e.last\_name, e.first\_name,

de.dept\_no, de.to\_date

FROM employees e INNER JOIN dept\_emp de

ON e.emp\_no = de.emp\_no

WHERE date\_part('year', de.to\_date) = 9999 AND de.dept\_no = 'd007';

CREATE TABLE stage6 as

SELECT s6a.emp\_no, s6a.last\_name, s6a.first\_name,

d.dept\_name

FROM stage6a s6a INNER JOIN departments d

ON s6a.dept\_no = d.dept\_no;

DROP TABLE stage6a;

*/\*\* to just list them –* ***produce display with no saved output*** *– do this:*

CREATE TABLE stage6a as

SELECT e.emp\_no, e.last\_name, e.first\_name,

de.dept\_no, de.to\_date

FROM employees e INNER JOIN dept\_emp de

ON e.emp\_no = de.emp\_no

WHERE date\_part('year', de.to\_date) = 9999 AND de.dept\_no = 'd007';

SELECT s6a.emp\_no, s6a.last\_name, s6a.first\_name,

d.dept\_name

FROM stage6a s6a INNER JOIN departments d

ON s6a.dept\_no = d.dept\_no; \*\*/

\*\*/

1. List all employees in the Sales and Development departments, including their employee number, last name, first name, and department name.

CREATE TABLE stage7a as

SELECT e.emp\_no, e.last\_name, e.first\_name,

de.dept\_no, de.to\_date

FROM employees e INNER JOIN dept\_emp de

ON e.emp\_no = de.emp\_no

WHERE date\_part('year', de.to\_date) = 9999 AND de.dept\_no = 'd007' OR de.dept\_no ='d005' ;

CREATE TABLE stage7 as

SELECT s7a.emp\_no, s7a.last\_name, s7a.first\_name,

d.dept\_name

FROM stage7a s7a INNER JOIN departments d

ON s7a.dept\_no = d.dept\_no;

DROP TABLE stage7a;

*/\*\* to just* ***list*** *them with no saved output, do this:*

*CREATE TABLE stage7a as*

*SELECT e.emp\_no, e.last\_name, e.first\_name,*

*de.dept\_no, de.to\_date*

*FROM employees e INNER JOIN dept\_emp de*

*ON e.emp\_no = de.emp\_no*

*WHERE date\_part('year', de.to\_date) = 9999 AND de.dept\_no = 'd007' OR de.dept\_no ='d005' ;*

*SELECT s7a.emp\_no, s7a.last\_name, s7a.first\_name,*

*d.dept\_name*

*FROM stage7a s7a INNER JOIN departments d*

## ON s7a.dept\_no = d.dept\_no;

## \*\*/

1. In descending order, list the frequency count of employee last names, i.e., how many employees share each last name.

SELECT last\_name, COUNT (last\_name)

FROM employees

GROUP BY last\_name

ORDER BY count DESC;

/\*\*\* REVERSAL to save the output

*CREATE TABLE stage8 as*

*SELECT last\_name, COUNT (last\_name)*

*FROM employees*

*GROUP BY last\_name*

*ORDER BY count DESC;* \*\*/

**## Bonus (Optional)**

As you examine the data, you are overcome with a creeping suspicion that the dataset is fake. You surmise that your boss handed you spurious data in order to test the data engineering skills of a new employee. To confirm your hunch, you decide to take the following steps to generate a visualization of the data, with which you will confront your boss:

1. Import the SQL database into Pandas. (Yes, you could read the CSVs directly in Pandas, but you are, after all, trying to prove your technical mettle.) This step may require some research. Feel free to use the code below to get started. Be sure to make any necessary modifications for your username, password, host, port, and database name:

```sql

from sqlalchemy import create\_engine

engine = create\_engine('postgresql://localhost:5432/<your\_db\_name>')

connection = engine.connect()

```

\* Consult [SQLAlchemy documentation] (https://docs.sqlalchemy.org/en/latest/core/engines.html#postgresql) for more information.

\* If using a password, do not upload your password to your GitHub repository. See [https://www.youtube.com/watch?v=2uaTPmNvH0I](https://www.youtube.com/watch?v=2uaTPmNvH0I) and [https://martin-thoma.com/configuration-files-in-python/](https://martin-thoma.com/configuration-files-in-python/) for more information.

2. Create a bar chart of average salary by title.

3. You may also include a technical report in markdown format, in which you outline the data engineering steps taken in the homework assignment.

**## Epilogue**

Evidence in hand, you march into your boss's office and present the visualization. With a sly grin, your boss thanks you for your work. On your way out of the office, you hear the words, "Search your ID number." You look down at your badge to see that your employee ID number is 499942.

**## Submission**

🗹 \* Create an image file of your ERD.

\* Create a `.sql` file of your table schemata.

\* Create a `.sql` file of your queries.

\* (Optional) Create a Jupyter Notebook of the bonus analysis.

\* Create and upload a repository with the above files to GitHub and post a link on BootCamp Spot.

1. One

SELECT

emp\_no,

birth\_date,

first\_name,

last\_name,

gender

FROM

Employees ;