SQL SUBQUERIES PROJECT

Question, SQL Queries, & Code Walkthroughs

https://github.com/camit001

1: Retrieve data from all tables in the 'employees' database.

SQL Query:

```
SELECT * FROM employees;
SELECT * FROM departments;
SELECT * FROM dept_emp;
SELECT * FROM dept_manager;
SELECT * FROM salaries;
SELECT * FROM customers;
```

```
SELECT * FROM sales;
```

Code Walkthrough:

Each of these queries uses the basic SELECT * statement to fetch all the rows and columns from different tables in the database.

2.1: Retrieve a list of all employees that are not managers.

SQL Query:

```
SELECT * FROM employees
WHERE emp_no NOT IN (SELECT emp_no FROM dept_manager);
```

Code Walkthrough:

This query retrieves all employees who are not listed as managers in the dept_manager table by using a subquery to exclude their employee numbers (emp_no).

2.2: ´Retrieve all columns in the sales table for customers above 60 years old.

SQL Query:

```
SELECT s.*
FROM sales s
JOIN customers c ON s.customer_id = c.customer_id
WHERE (c.age) > 60;
```

Code Walkthrough:

Retrieves all sales records for customers older than 60 by joining `sales` with `customers`.

2.3: Retrieve a list of all manager’s employees number, first and last names returns all the data from the dept_manager table.

SQL Query:

```
SELECT e.emp_no, first_name, last_name
FROM employees e
JOIN  dept_manager dm ON e.emp_no = dm.emp_no;
```

Code Walkthrough:

Lists each manager's employee number and name by joining `employees` with `dept_manager`.

2.4 Retrieve a list of all managers that were employed between 1st January, 1990 and 1st January, 1995.

SQL Query:

```
SELECT * FROM dept_manager
WHERE emp_no IN (SELECT emp_no FROM employees
WHERE hire_date BETWEEN '1990-01-01' AND '1995-01-01');
```

Code Walkthrough:

Finds managers hired between January 1, 1990 and January 1, 1995 using a subquery on `employees`.

3.1: Retrieve a list of all customers living in the southern region.

SQL Query:

```
SELECT *
FROM customers
WHERE Region = 'South';
```

Code Walkthrough:

Selects all customers whose `Region` is "South."

3.2: Retrieve a list of managers and their department names.

SQL Query:

```
SELECT
    dm.emp_no,
    e.first_name,
    e.last_name,
    d.dept_name
FROM dept_manager dm
JOIN employees e ON dm.emp_no = e.emp_no
JOIN departments d ON dm.dept_no = d.dept_no;
```

Code Walkthrough:

Joins `dept_manager`, `employees`, and `departments` to list managers with their department names.

3.3: Retrieve a list of managers, their first, last, and their department names returns data from the employees table.

SQL Query:

```
SELECT dm.*,  e.first_name, e.last_name, d.dept_name
FROM dept_manager dm, employees e, (SELECT dept_no, dept_name
FROM departments) d
WHERE dm.dept_no = d.dept_no AND e.emp_no = dm.emp_no;
```

Code Walkthrough:

Uses an inline subquery for `departments` and implicit joins to fetch manager details and department names.

4.1: Retrieve the first name, last name and average salary of all employees.

SQL Query:

```
SELECT
```

```
    e.first_name,
    e.last_name,
    AVG(s.salary) AS average_salary
FROM employees e
JOIN salaries s ON e.emp_no = s.emp_no
GROUP BY e.emp_no, e.first_name, e.last_name;
```

Code Walkthrough:

Computes each employee's average salary by grouping `salaries` joined to `employees`.

4.2: Retrieve a list of customer_id, product_id, order_line and the name of the customer returns data from the sales and customers tables.

SQL Query:

```
SELECT customer_id, product_id, order_line, (SELECT customer_name
FROM customers c
WHERE s.customer_id = c.customer_id)
FROM sales s
ORDER BY customer_id;
```

Code Walkthrough:

Fetches sales details plus customer names using a scalar subquery in the `SELECT` clause.

5.1: Return a list of all employees who are in Customer Service department returns data from the dept_emp and departments tables.

SQL Query:

```
SELECT * FROM dept_emp
WHERE dept_no IN (SELECT dept_no FROM departments
```

```
WHERE dept_name = 'Customer Service');
```

Code Walkthrough:

Retrieves all rows from `dept_emp` for the “Customer Service” department via a subquery on `departments`.

5.2: Include the employee number, first and last names.

SQL Query:

```
SELECT a.emp_no, b.dept_no, a.first_name, a.last_name
FROM employees a
JOIN (SELECT * FROM dept_emp
WHERE dept_no IN (SELECT dept_no FROM departments
WHERE dept_name = 'Customer Service')) b
ON a.emp_no = b.emp_no
ORDER BY emp_no;
```

Code Walkthrough:

Joins filtered `dept_emp` rows for Customer Service with `employees` to include employee names.

5.3: Retrieve a list of all managers who became managers after the 1st of January, 1985 and are in the Finance or HR department.

SQL Query:

```
SELECT * FROM dept_manager
WHERE from_date > '1985-01-01'
AND dept_no IN (SELECT dept_no FROM departments
WHERE dept_name = 'Finance' OR dept_name = 'Human Resources');
```

Code Walkthrough:

Lists managers who started after January 1, 1985 in Finance or HR by filtering on both `from_date` and dept subquery.

5.4: Retrieve a list of all employees that earn above 120,000 and are in the Finance or HR departments.

SQL Query:

```
SELECT emp_no, salary FROM salaries
WHERE salary > 120000
AND emp_no IN (SELECT emp_no FROM dept_emp
WHERE dept_no IN ('d002','d003'));
```

Code Walkthrough:

Finds employees in Finance (d002) or HR (d003) earning above 120,000 by combining salary and department filters.

5.5: Retrieve the average salary of these employees.

SQL Query:

```
SELECT emp_no, ROUND(AVG(salary), 2) AS avg_salary
FROM salaries
WHERE salary > 120000
AND emp_no IN (SELECT emp_no FROM dept_emp
WHERE dept_no = 'd002' OR dept_no = 'd003')
GROUP BY emp_no
ORDER BY avg_salary DESC;
```

Code Walkthrough:

Calculates and ranks the average salaries of high-earning Finance/HR employees using aggregation after filtering.

6.1: Return a list of all employees number, first and last name. Also, return the average salary of all the employees and average salary of each employee.

SQL Query:

```
SELECT e.emp_no, e.first_name, e.last_name, a.emp_avg_salary,
(SELECT ROUND(AVG(salary), 2) avg_salary FROM salaries)
FROM employees e
JOIN (SELECT s.emp_no, ROUND(AVG(salary), 2) AS emp_avg_salary
FROM salaries s
GROUP BY s.emp_no
ORDER BY s.emp_no) a
ON e.emp_no = a.emp_no
ORDER BY emp_no;
```

Code Walkthrough:

Joins each employee to their personal average salary and also retrieves the overall average via a scalar subquery.

6.2: Find the difference between an employee's average salary and the average salary of all employees.

SQL Query:

```
SELECT e.emp_no, e.first_name, e.last_name, a.emp_avg_salary,
(SELECT ROUND(AVG(salary), 2) avg_salary FROM salaries),
a.emp_avg_salary - (SELECT ROUND(AVG(salary), 2) avg_salary FROM
salaries) AS salary_diff
FROM employees e
JOIN (SELECT s.emp_no, ROUND(AVG(salary), 2) AS emp_avg_salary
FROM salaries s
GROUP BY s.emp_no
ORDER BY s.emp_no) a
ON e.emp_no = a.emp_no
ORDER BY emp_no;
```

Code Walkthrough:

Calculates each employee’s salary difference from the overall average by combining joins and scalar subqueries.

6.3: Find the difference between the maximum salary of employees in the Finance or HR department and the maximum salary of all employees.

SQL Query:

```
SELECT e.emp_no, e.first_name, e.last_name, a.emp_max_salary,
(SELECT MAX(salary) max_salary FROM salaries),
(SELECT MAX(salary) max_salary FROM salaries) - a.emp_max_salary
salary_diff
FROM employees e
JOIN (SELECT s.emp_no, MAX(salary) AS emp_max_salary
FROM salaries s
GROUP BY s.emp_no
ORDER BY s.emp_no) a
ON e.emp_no = a.emp_no
WHERE e.emp_no IN (SELECT emp_no FROM dept_emp WHERE
dept_no IN ('d002', 'd003'))
ORDER BY emp_no;
```

Code Walkthrough:

Determines how far each Finance/HR manager's top salary falls short of the company's maximum salary using nested subqueries.

7.1: Retrieve the salary that occurred the most.

SQL Query:

```
SELECT a.salary
FROM (
SELECT salary, COUNT(*)
FROM salaries
GROUP BY salary
ORDER BY COUNT(*) DESC, salary DESC
```

```
LIMIT 1) a;
```

Code Walkthrough:

Identifies the most frequent salary value by counting occurrences, ordering by count (and salary), and limiting to one.

7.2: Find the average salary excluding the highest and the lowest salaries.

SQL Query:

```
SELECT ROUND(AVG(salary), 2) avg_salary
FROM salaries
WHERE salary NOT IN (
(SELECT MIN(salary) FROM salaries),
(SELECT MAX(salary) FROM salaries)
);
```

Code Walkthrough:

Calculates the average salary excluding the minimum and maximum salaries via a NOT IN filter with two scalar subqueries.

7.3: Retrieve a list of customers id, name that has bought the most from the store.

SQL Query:

```
SELECT c.customer_id, c.customer_name, a.cust_count
FROM customers c,
(SELECT customer_id, COUNT(*) AS cust_count
FROM sales
 GROUP BY customer_id
ORDER BY cust_count DESC) AS a
  WHERE c.customer_id = a.customer_id
ORDER BY a.cust_count DESC;
```

Code Walkthrough:

Joins customers to a subquery that ranks them by purchase count to list the highest-buying customers.

7.4: Retrieve a list of the customer name and segment of those customers that bought the most from the store and had the highest total sales.

SQL Query:

```
SELECT c.customer_id, c.customer_name, c.segment, a.cust_count,
a.total_sales
FROM customers c,
(SELECT customer_id, COUNT(*) AS cust_count, SUM(sales) total_sales
FROM sales
GROUP BY customer_id
ORDER BY total_sales DESC, cust_count DESC) AS a
WHERE c.customer_id = a.customer_id
ORDER BY a.total_sales DESC, a.cust_count DESC;
```

Code Walkthrough:

Lists customers by total sales and purchase count by joining customer details to aggregated sales data sorted by revenue and count.

THANK YOU
FOR YOUR ATTENTION

- Want to explore the project further?
- Visit our GitHub repository for the full code and documentation.
- Access SQL queries, datasets, and project details.
- We welcome your contributions and feedback!

https://github.com/camit001