Class Assignment 1

Submitted by Nakul Dev

1. Find all employees whose first names start with a vowel and whose last names end with a consonant

```
SELECT *
FROM employees
WHERE first_name REGEXP '^[aeiouAEIOU]'
AND last_name REGEXP '[^aeiouAEIOU]$';
```

2. For each department, display the total salary expenditure, the average salary, and the highest salary. Use window functions to calculate the total, average, and max salary, but show each result for all employees in that department.

```
SELECT department_id, employee_id, salary,
    SUM(salary) OVER (PARTITION BY department_id) AS total_salary_expdr,
    AVG(salary) OVER (PARTITION BY department_id) AS avg_salary,
    MAX(salary) OVER (PARTITION BY department_id) AS max_salary
FROM employees e
JOIN salaries s ON e.employee_id = s.employee_id;
```

3. Write a query that fetches the following:
All employees, their department name, their manager's name (if they have one), and their salary.
You will need to:
Join employees with their department.
Perform a self-join to fetch the manager's name.

```
SELECT CONCAT(e.first_name,' ', e.last_name) AS employee,
       d.department_name,
       CONCAT(m.first_name,' ', m.last_name) AS manager,
       s.salary
From employees e
LEFT JOIN departments d ON d.department_id = s.department_id
LEFT JOIN employees m ON e.manager_id = m.employee_id
LEFT JOIN salaries s ON e.employee_id = s.employee_id;
```

4. Create a query using a recursive CTE to list all employees and their respective reporting chains (i.e., list the manager's manager and so on).

```
WITH RECURSIVE reporting_chain AS (
       SELECT employee_id, first_name, manager_id
       FROM employees
       WHERE manager_id IS NULL

UNION ALL

       SELECT e.employee_id, e.first_name, e.manager_id
       FROM employees e
       INNER JOIN employee_hierarchy eh ON e.manager_id = eh.employee_id
)
SELECT * FROM reporting_chain;
```

5. Write a query to fetch the details of employees earning above a certain salary threshold. Investigate the performance of this query and suggest improvements, including the use of indexes

```
SELECT e.first_name, e.last_name, s.salary
FROM employee e
JOIN salaries s ON e.employee_id = s.employee_id
SELECT s.salary > 6000;
```

6. You need to create a detailed sales report. First, create a temporary table to store interim sales data for each product, including total sales, average sales per customer, and the top salesperson for each product.
Hint:
Use temporary tables and insert data from subqueries.

```
CREATE TEMPORARY TABLE temp_sales_report (
    product_id INT,
    product_name VARCHAR(255),
    total_sales DECIMAL(10, 2),
    average_sales_per_customer DECIMAL(10, 2),
    top_salesperson_id INT,
    top_salesperson_name VARCHAR(255)
);
```

```
INSERT INTO temp_sales_report (product_id, product_name, total_sales,
average_sales_per_customer,
top_salesperson_id, top_salesperson_name)
SELECT
    p.product_id,
    p.product_name,
    SUM(s.amount) AS total_sales,
    AVG(s.amount) AS average_sales_per_customer,
    sp.salesperson_id AS top_salesperson_id,
    CONCAT(sp.first_name, ' ', sp.last_name) AS top_salesperson_name
FROM products p
JOIN sales s ON p.product_id = s.product_id
JOIN (
    SELECT
        product_id,
        salesperson_id,
        SUM(amount) AS total_sales_by_salesperson
    FROM sales
    GROUP BY product_id, salesperson_id
) AS total_sales_by_sp ON s.product_id = total_sales_by_sp.product_id AND
s.salesperson_id = total_sales_by_sp.salesperson_id
JOIN salespeople sp ON total_sales_by_sp.salesperson_id = sp.salesperson_id
WHERE (total_sales_by_sp.product_id,
total_sales_by_sp.total_sales_by_salesperson) IN (
    SELECT product_id, MAX(total_sales_by_salesperson)
    FROM (
        SELECT
            product_id,
            salesperson_id,
            SUM(amount) AS total_sales_by_salesperson
```

```
        FROM sales
        GROUP BY product_id, salesperson_id
    ) AS inner_query
    GROUP BY product_id
)
GROUP BY p.product_id, p.product_name;

SELECT * FROM temp_sales_report;
```