ASSESSMENT 1

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

```
SELECT *
FROM employees
WHERE LOWER(SUBSTRING(first_name, 1, 1)) IN ('a', 'e', 'i', 'o', 'u')
  AND LOWER(SUBSTRING(last_name, -1)) NOT IN ('a', 'e', 'i', 'o', 'u');
```

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
    d.department_name,
    SUM(s.salary) OVER (PARTITION BY e.department_id) AS total_salary_expenditure,
    AVG(s.salary) OVER (PARTITION BY e.department_id) AS average_salary,
    MAX(s.salary) OVER (PARTITION BY e.department_id) AS highest_salary
FROM employees e
JOIN salaries s ON e.employee_id = s.employee_id
JOIN departments d ON e.department_id = d.department_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
    e.first_name,
    e.last_name,
```

```
    d.department_name,
    CONCAT(m.first_name, ' ', m.last_name) AS manager_name,
    s.salary
FROM employees e
LEFT JOIN employees m ON e.employee_id = m.employee_id
JOIN departments d ON e.department_id = d.department_id
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 .

```
WITH RECURSIVE ReportingChain AS (
    SELECT
        employee_id,
        first_name || ' ' || last_name AS employee_name,
        manager_id,
        CAST(first_name || ' ' || last_name AS VARCHAR(255)) AS reporting_chain
    FROM employees
    WHERE manager_id IS NULL

    UNION ALL

    SELECT
        e.employee_id,
        e.first_name || ' ' || e.last_name AS employee_name,
        e.manager_id,
        rc.reporting_chain || ' -> ' || e.first_name || ' ' || e.last_name AS reporting_chain
    FROM employees e
```

```
    JOIN ReportingChain rc ON e.manager_id = rc.employee_id  )
SELECT
    employee_id,
    employee_name,
    reporting_chain
FROM ReportingChain;
```

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 *
FROM employees e
JOIN salaries s ON e.employee_id = s.employee_id
WHERE s.salary > 70000;
```

```
CREATE INDEX idx_salary ON salaries(salary);
```

6//ou 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.

has context menu

```
CREATE TEMPORARY TABLE temp_sales_repo AS
SELECT
    product_id,
    SUM(sales_amount) AS total_sales,
    AVG(sales_amount) AS average_sales_per_customer,
    (SELECT salesperson
```

```
    FROM sales_data s2
    WHERE s2.product_id = s1.product_id
    GROUP BY salesperson
    ORDER BY SUM(sales_amount) DESC
    LIMIT 1) AS top_salesperson
FROM sales_data s1
GROUP BY product_id;


SELECT * FROM temp_sales_repo;


DROP TABLE temp_sales_repo;
```