

# Assignment 1

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

Select \*

FROM employees

WHERE LEFT(first\_name, 1) IN ('A', 'E', 'I', 'O', 'U')

AND RIGHT(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

department\_id,

employee\_id, salary,

SUM(salary) OVER (PARTITION BY department\_id) AS total\_salary\_expenditure,

AVG(salary) OVER (PARTITION BY department\_id) AS avg\_salary,

MAX(salary) OVER (PARTITION BY department\_id) AS max\_salary

FROM employees;

3. Write a query that fetches the following:

All employees, their department name, their manager's name and their salary.

You will need to:

Join employees with their department.

Perform a self-join to fetch the manager's name.

SELECT

```

e.employee_id,
e.first_name || ' ' || e.last_name AS employee_name,
d.department_name,
e.salary,
m.first_name || ' ' || m.last_name AS manager_name
FROM employees e
JOIN departments d ON e.department_id = d.department_id
LEFT JOIN employees m ON e.manager_id = m.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 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.

```
SELECT employee_id, first_name, last_name, department_id, salary  
  
FROM employees  
  
WHERE salary > 50000;
```

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,
    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
        s1.product_id,
        s1.salesperson_id,
        SUM(s1.amount) AS total_sales_by_salesperson
    FROM sales s1
    GROUP BY s1.product_id, s1.salesperson_id
    HAVING SUM(s1.amount) = (
        SELECT MAX(SUM(s2.amount))
        FROM sales s2
        WHERE s2.product_id = s1.product_id
        GROUP BY s2.salesperson_id
    )
) top_sp ON p.product_id = top_sp.product_id JOIN salespeople sp ON top_sp.salesperson_id =
sp.salesperson_id GROUP BY p.product_id, p.product_name, top_sp.salesperson_id, sp.first_name,
sp.last_name;
SELECT * FROM temp_sales_report;

```