SQL_Assignemnt

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

Query:

```
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. Display total, average, and highest salary for each department using window functions

Query:

```
Select

    department_id,

    department_name,

    employee_id,

    first_name,

    last_name,

    salary,

    Sum(salary) Over (PARTITION BY department_id) AS total_salary,

    Avg(salary) Over (PARTITION BY department_id) AS avg_salary,
```

Max(salary) Over (PARTITION BY department_id) AS max_salary

FROM employees e

JOIN departments d ON e.department_id = d.department_id;

3. Fetch all employees, their department, their manager's name, and their salary

Query:

```
Select
    e.employee_id,
    e.first_name AS employee_first_name,
    e.last_name AS employee_last_name,
    d.department_name,
    m.first_name AS manager_first_name,
    m.last_name AS manager_last_name,
    e.salary
From employees e
Left Join employees m On e.manager_id = m.employee_id
Join departments d On e.department_id = d.department_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).

Query:

```
WITH RECURSIVE EmployeeHierarchy AS (
    SELECT
        employee_id,
        first_name,
        last_name,
        manager_id,
        0 AS level
    FROM employees
    WHERE manager_id IS NULL

    UNION ALL

    SELECT
        e.employee_id,
        e.first_name,
        e.last_name,
        e.manager_id,
        eh.level + 1
    FROM employees e
    JOIN EmployeeHierarchy eh ON e.manager_id = eh.employee_id
```

)

SELECT * FROM EmployeeHierarchy;

5. Fetch details of employees earning above a salary threshold and suggest improvements

Query:

```
SELECT
    employee_id,
    first_name,
    last_name,
    department_id,
    salary
FROM employees
WHERE salary > 40000;
```

6. Create a temporary table for interim sales data and populate it

Query:

```
CREATE TEMPORARY TABLE product_sales_report (
    product_id INT,
    total_sales DECIMAL(10, 2),
    avg_sales_per_customer DECIMAL(10, 2),
    top_salesperson_id INT
);
```

```
INSERT INTO product_sales_report (product_id, total_sales, avg_sales_per_customer, top_salesperson_id)
SELECT
    p.product_id,
    SUM(s.amount) AS total_sales,
    AVG(s.amount / s.customer_count) AS avg_sales_per_customer,
    (
        SELECT salesperson_id
        FROM sales s2
        WHERE s2.product_id = s.product_id
        ORDER BY s2.amount DESC
        LIMIT 1
    ) AS top_salesperson_id
FROM sales s
JOIN products p ON s.product_id = p.product_id
GROUP BY p.product_id;
```