**DAY 8 Assessment**

CREATE TABLE if NOT EXISTS Employees (

employee\_id INT PRIMARY KEY,

name VARCHAR(100),

job\_title VARCHAR(50),

department\_id INT,

salary DECIMAL(10, 2),

manager\_id INT,

hire\_date DATE

);

INSERT INTO Employees (employee\_id, name, job\_title, department\_id, salary, manager\_id, hire\_date) VALUES

(1, 'John Doe', 'Manager', 1, 90000, NULL, '2015-06-15'),

(2, 'Jane Smith', 'Developer', 1, 75000, 1, '2018-04-10'),

(3, 'Emily Davis', 'Developer', 2, 80000, 1, '2019-07-22'),

(4, 'Michael Brown', 'Salesperson', 3, 60000, 1, '2020-01-05'),

(5, 'David Wilson', 'Developer', 2, 95000, 1, '2021-03-15'),

(6, 'Sarah Miller', 'Salesperson', 3, 65000, 4, '2021-06-20'),

(7, 'Robert Moore', 'HR', 4, 50000, NULL, '2017-11-25'),

(8, 'Laura Taylor', 'Developer', 1, 70000, 1, '2022-09-10');

CREATE TABLE if NOT EXISTS Departments (

department\_id INT PRIMARY KEY,

department\_name VARCHAR(50)

);

INSERT INTO Departments (department\_id, department\_name) VALUES

(1, 'Engineering'),

(2, 'Marketing'),

(3, 'Sales'),

(4, 'Human Resources');

CREATE TABLE if NOT EXISTS Sales (

sale\_id INT PRIMARY KEY,

product\_id INT,

sale\_date DATE,

quantity INT,

price DECIMAL(10, 2),

salesperson\_id INT

);

INSERT INTO if NOT EXISTS Sales (sale\_id, product\_id, sale\_date, quantity, price, salesperson\_id) VALUES

(1, 101, '2024-05-01', 10, 100, 4),

(2, 102, '2024-05-10', 5, 200, 4),

(3, 101, '2024-06-01', 3, 100, 6),

(4, 103, '2024-06-10', 7, 150, 6),

(5, 104, '2024-07-05', 2, 300, 4),

(6, 101, '2024-08-15', 12, 100, 6),

(7, 105, '2024-08-20', 6, 120, 4);

CREATE TABLE if NOT EXISTS Products (

product\_id INT PRIMARY KEY,

product\_name VARCHAR(100),

category\_id INT,

cost DECIMAL(10, 2)

);

INSERT INTO Products (product\_id, product\_name, category\_id, cost) VALUES

(101, 'Product A', 1, 50),

(102, 'Product B', 2, 120),

(103, 'Product C', 3, 75),

(104, 'Product D', 2, 200),

(105, 'Product E', 1, 80);

CREATE TABLE if NOT EXISTS Customers (

customer\_id INT PRIMARY KEY,

customer\_name VARCHAR(100),

email VARCHAR(100)

);

INSERT INTO Customers (customer\_id, customer\_name, email) VALUES

(1, 'Alice Johnson', 'alice.johnson@email.com'),

(2, 'Bob Anderson', 'bob.anderson@email.com'),

(3, 'Charlie Davis', 'charlie.davis@email.com'),

(4, 'Diana Lee', 'diana.lee@email.com'),

(5, 'Ethan Moore', 'ethan.moore@email.com');

CREATE TABLE if NOT EXISTS Orders (

order\_id INT PRIMARY KEY,

customer\_id INT,

order\_date DATE,

order\_value DECIMAL(10, 2)

);

INSERT INTO Orders (order\_id, customer\_id, order\_date, order\_value) VALUES

(1, 1, '2024-01-15', 500),

(2, 2, '2024-02-20', 300),

(3, 3, '2024-03-05', 700),

(4, 4, '2024-04-10', 450),

(5, 5, '2024-05-25', 600),

(6, 1, '2024-06-10', 400),

(7, 2, '2024-07-10', 550),

(8, 5, '2024-08-15', 800);

CREATE TABLE if NOT EXISTS EmployeeVacations (

employee\_id INT,

vacation\_date DATE,

PRIMARY KEY (employee\_id, vacation\_date)

);

INSERT INTO EmployeeVacations (employee\_id, vacation\_date) VALUES

(1, '2024-01-01'),

(2, '2024-03-01'),

(3, '2024-04-15'),

(5, '2024-06-05'),

(7, '2024-07-10');

CREATE TABLE if NOT EXISTS Promotions (

employee\_id INT,

promotion\_date DATE,

PRIMARY KEY (employee\_id, promotion\_date)

);

INSERT INTO Promotions (employee\_id, promotion\_date) VALUES

(2, '2019-07-10'),

(4, '2020-01-15'),

(5, '2021-05-20'),

(6, '2023-02-10'),

(8, '2023-09-15')

1. Write a query to find the second highest salary from the employees table.

**SELECT MAX(salary) AS second\_highest\_salary**

**FROM Employees**

**WHERE salary < (SELECT MAX(salary) FROM Employees);**

1. Write a SQL query to retrieve the names of employees who have the same job title as the employee with the ID of 100.

**SELECT name**

**FROM Employees**

**WHERE job\_title = (SELECT job\_title FROM Employees WHERE employee\_id = 3);**

1. Write a query to calculate the total sales for each product in the sales table, grouped by product ID.

**SELECT product\_id, SUM(quantity \* price) AS total\_sales**

**FROM Sales**

**GROUP BY product\_id;**

1. Write a SQL query to find all employees who joined in the last 30 days.

**SELECT name**

**FROM Employees**

**WHERE hire\_date > CURDATE() - INTERVAL 30 DAY;**

1. Write a query to return the top 5 products with the highest sales revenue from the sales table.

**SELECT product\_id, SUM(quantity \* price) AS total\_sales**

**FROM Sales**

**GROUP BY product\_id**

**ORDER BY total\_sales DESC**

**LIMIT 5;**

1. Write a SQL query to find the average salary of employees in each department, but only include departments with more than 5 employees.

**SELECT department\_id, AVG(salary) AS average\_salary**

**FROM Employees**

**GROUP BY department\_id**

**HAVING COUNT(employee\_id) > 5;**

1. Write a query to list all customers who have not placed any orders in the last year.

**SELECT customer\_name**

**FROM Customers**

**WHERE customer\_id NOT IN (**

**SELECT DISTINCT customer\_id**

**FROM Orders**

**WHERE order\_date > CURDATE() - INTERVAL 1 YEAR**

**);**

1. Write a SQL query to find the total number of orders placed by each customer, including customers with zero orders.

**SELECT c.customer\_name, COUNT(o.order\_id) AS total\_orders**

**FROM Customers c**

**LEFT JOIN Orders o ON c.customer\_id = o.customer\_id**

**GROUP BY c.customer\_id;**

1. Write a query to find the maximum and minimum salary in each department from the employees table.

**SELECT department\_id, MAX(salary) AS max\_salary, MIN(salary) AS min\_salary**

**FROM Employees**

**GROUP BY department\_id;**

1. Write a SQL query to retrieve the names of employees who earn more than the average salary of their department.

**SELECT name**

**FROM Employees e**

**WHERE e.salary > (**

**SELECT AVG(salary)**

**FROM Employees**

**WHERE department\_id = e.department\_id**

**);**

1. Write a query to find the employees who have the same manager as the employee with the name 'John Doe'.

**SELECT name**

**FROM Employees**

**WHERE manager\_id = (SELECT manager\_id FROM Employees WHERE name = 'John Doe');**

1. Write a SQL query to calculate the cumulative sales for each month from the sales table.

**SELECT YEAR(sale\_date) AS year, MONTH(sale\_date) AS month,**

**SUM(quantity \* price) AS cumulative\_sales**

**FROM Sales**

**GROUP BY YEAR(sale\_date), MONTH(sale\_date)**

**ORDER BY year, month;**

1. Write a query to find the products that have never been sold from the products table.

**SELECT product\_id, product\_name**

**FROM Products**

**WHERE product\_id NOT IN (SELECT DISTINCT product\_id FROM Sales);**

1. Write a SQL query to retrieve the top 3 customers based on total order value from the orders table.

**SELECT customer\_id, SUM(order\_value) AS total\_order\_value**

**FROM Orders**

**GROUP BY customer\_id**

**ORDER BY total\_order\_value DESC**

**LIMIT 3;**

1. Write a query to find the total number of unique products sold in each quarter of the year.

**SELECT QUARTER(sale\_date) AS quarter, COUNT(DISTINCT product\_id) AS unique\_products\_sold**

**FROM Sales**

**GROUP BY QUARTER(sale\_date);**

1. Write a SQL query to identify duplicate records in the customers table based on the email column.

**SELECT email, COUNT(\*) AS count**

**FROM Customers**

**GROUP BY email**

**HAVING COUNT(\*) > 1;**

1. Write a query to find the department with the highest total salary expenditure.

**SELECT department\_id, SUM(salary) AS total\_salary**

**FROM Employees**

**GROUP BY department\_id**

**ORDER BY total\_salary DESC**

**LIMIT 1;**

1. Write a SQL query to retrieve the last 5 orders placed by each customer.

**SELECT o.customer\_id, o.order\_id, o.order\_date**

**FROM Orders o**

**WHERE o.order\_id IN (**

**SELECT order\_id**

**FROM Orders**

**WHERE customer\_id = o.customer\_id**

**ORDER BY order\_date DESC**

**LIMIT 5**

**)**

**ORDER BY o.customer\_id, o.order\_date DESC;**

1. Write a query to find the average time taken to fulfill orders from the order stable.

**SELECT AVG(DATEDIFF(order\_date, order\_received\_date)) AS avg\_fulfillment\_time**

**FROM Orders;**

1. Write a SQL query to return the names of employees who have not taken any vacation days in the last year.

**SELECT name**

**FROM Employees e**

**WHERE e.employee\_id NOT IN (**

**SELECT employee\_id**

**FROM EmployeeVacations**

**WHERE vacation\_date > CURDATE() - INTERVAL 1 YEAR**

**);**

1. Write a query to find the top 10 products with the highest profit margin from the products table.

**SELECT p.product\_name, (p.price - p.cost) AS profit\_margin**

**FROM Products p**

**ORDER BY profit\_margin DESC**

**LIMIT 10;**

1. Write a SQL query to calculate the percentage of total sales for each product category in the sales table.

**SELECT p.category\_id,**

**SUM(s.quantity \* s.price) / (SELECT SUM(quantity \* price) FROM Sales) \* 100 AS sales\_percentage**

**FROM Sales s**

**JOIN Products p ON s.product\_id = p.product\_id**

**GROUP BY p.category\_id;**

1. Write a query to find the employees who have received more than one promotion in the last 5 years.

**SELECT e.name**

**FROM Employees e**

**JOIN Promotions p ON e.employee\_id = p.employee\_id**

**WHERE p.promotion\_date > CURDATE() - INTERVAL 5 YEAR**

**GROUP BY e.employee\_id**

**HAVING COUNT(p.promotion\_date) > 1;**

1. Write a SQL query to retrieve the names of customers who have placed orders in every month of the last year.

**SELECT customer\_id**

**FROM Orders**

**WHERE order\_date > CURDATE() - INTERVAL 1 YEAR**

**GROUP BY customer\_id**

**HAVING COUNT(DISTINCT MONTH(order\_date)) = 12;**

1. Write a query to find the average order value for each customer in the orders table.

**SELECT customer\_id, AVG(order\_value) AS avg\_order\_value**

**FROM Orders**

**GROUP BY customer\_id;**

1. Write a SQL query to retrieve the names of employees who have the same salary as the highest-paid employee in the company.

**SELECT name**

**FROM Employees**

**WHERE salary = (SELECT MAX(salary) FROM Employees);**

1. Write a query to find the total number of products sold by each salesperson in the sales table.

**SELECT salesperson\_id, COUNT(\*) AS total\_products\_sold**

**FROM Sales**

**GROUP BY salesperson\_id;**

1. Write a SQL query to retrieve the names of employees who have worked in more than one department.

**SELECT name**

**FROM Employees**

**GROUP BY name**

**HAVING COUNT(DISTINCT department\_id) > 1;**

1. Write a query to find the total sales for each product category, including categories with zero sales.

**SELECT p.category\_id,**

**COALESCE(SUM(s.quantity \* s.price), 0) AS total\_sales**

**FROM Products p**

**LEFT JOIN Sales s ON p.product\_id = s.product\_id**

**GROUP BY p.category\_id;**

1. Write a SQL query to calculate the year-over-year growth in sales for each product in the sales table.

**SELECT product\_id,**

**YEAR(sale\_date) AS year,**

**SUM(quantity \* price) AS total\_sales,**

**LAG(SUM(quantity \* price), 1) OVER (PARTITION BY product\_id ORDER BY YEAR(sale\_date)) AS previous\_year\_sales,**

**(SUM(quantity \* price) - LAG(SUM(quantity \* price), 1) OVER (PARTITION BY product\_id ORDER BY YEAR(sale\_date)))**

**/ LAG(SUM(quantity \* price), 1) OVER (PARTITION BY product\_id ORDER BY YEAR(sale\_date)) \* 100 AS yoy\_growth**

**FROM Sales**

**GROUP BY product\_id, YEAR(sale\_date);**