**-- SQL Question Set: Part 1**

**-- Question 1: Combine Two Tables**

SELECT p.firstName, p.lastName, a.city, a.state

FROM Person p

LEFT JOIN Address a

ON p.personId = a.personId;

CREATE TABLE Person (

personId INT PRIMARY KEY,

firstName VARCHAR(50),

lastName VARCHAR(50)

);

CREATE TABLE Address (

addressId INT PRIMARY KEY,

personId INT,

city VARCHAR(50),

state VARCHAR(50),

FOREIGN KEY (personId) REFERENCES Person(personId)

);

INSERT INTO Person VALUES (1, 'John', 'Doe'), (2, 'Jane', 'Smith');

INSERT INTO Address VALUES (1, 1, 'New York', 'NY'), (2, 2, 'Los Angeles', 'CA');

**-- Question 2: Second Highest Salary**

SELECT MAX(salary) AS SecondHighestSalary

FROM Employee

WHERE salary < (SELECT MAX(salary) FROM Employee);

CREATE TABLE Employee (

employeeId INT PRIMARY KEY,

name VARCHAR(50),

salary DECIMAL(10, 2)

);

INSERT INTO Employee VALUES (1, 'Alice', 50000), (2, 'Bob', 70000), (3, 'Charlie', 60000);

**-- Question 3: Rank Scores Without Gaps**

SELECT score, DENSE\_RANK () OVER (ORDER BY score DESC) AS rank

FROM Scores;

CREATE TABLE Scores (

id INT PRIMARY KEY,

score INT

);

INSERT INTO Scores VALUES (1, 100), (2, 200), (3, 100), (4, 300);

**-- Question 4: Numbers Appearing at Least Three Times Consecutively**

SELECT DISTINCT num

FROM (

SELECT num,

LEAD(num, 1) OVER (ORDER BY id) AS next\_num,

LEAD(num, 2) OVER (ORDER BY id) AS next\_num2

FROM Logs

) t

WHERE num = next\_num AND num = next\_num2;

CREATE TABLE Logs (

id INT PRIMARY KEY,

num INT

);

INSERT INTO Logs VALUES (1, 1), (2, 1), (3, 1), (4, 2), (5, 2), (6, 1);

**-- Question 5: Employees Who Earn More Than Their Managers**

SELECT e.name AS Employee, m.name AS Manager

FROM Employee e

JOIN Employee m ON e.managerId = m.employeeId

WHERE e.salary > m.salary;

CREATE TABLE Employee (

employeeId INT PRIMARY KEY,

name VARCHAR(50),

salary DECIMAL(10, 2),

managerId INT,

FOREIGN KEY (managerId) REFERENCES Employee(employeeId)

);

INSERT INTO Employee VALUES

(1, 'Alice', 50000, NULL),

(2, 'Bob', 70000, 1),

(3, 'Charlie', 60000, 1),

(4, 'David', 55000, 2);

**-- Question 6: Department with the Highest Average Salary**

SELECT d.name AS Department, AVG(e.salary) AS AverageSalary

FROM Department d

JOIN Employee e ON d.departmentId = e.departmentId

GROUP BY d.name

ORDER BY AverageSalary DESC

LIMIT 1;

CREATE TABLE Department (

departmentId INT PRIMARY KEY,

name VARCHAR (50)

);

ALTER TABLE Employee ADD departmentId INT;

ALTER TABLE Employee ADD FOREIGN KEY (departmentId) REFERENCES Department(departmentId);

INSERT INTO Department VALUES (1, 'HR'), (2, 'Engineering');

INSERT INTO Employee (employeeId, name, salary, managerId, departmentId) VALUES

(1, 'Alice', 50000, NULL, 1),

(2, 'Bob', 70000, 1, 2),

(3, 'Charlie', 60000, 1, 2),

(4, 'David', 55000, 2, 2);

**-- Question 7: Find Employees Hired in the Last Year**

SELECT name

FROM Employee

WHERE hireDate >= DATEADD (YEAR, -1, GETDATE ());

ALTER TABLE Employee ADD hireDate DATE;

UPDATE Employee SET hireDate = '2023-01-01' WHERE employeeId = 1;

UPDATE Employee SET hireDate = '2024-03-15' WHERE employeeId = 2;

UPDATE Employee SET hireDate = '2024-07-20' WHERE employeeId = 3;

UPDATE Employee SET hireDate = '2022-12-10' WHERE employeeId = 4;

**-- SQL Question Set: Comprehensive**

**-- Question 8: Find Duplicate Emails**

SELECT email, COUNT(\*)

FROM Person

GROUP BY email

HAVING COUNT(\*) > 1;

ALTER TABLE Person ADD email VARCHAR(50);

UPDATE Person SET email = 'john@example.com' WHERE personId = 1;

UPDATE Person SET email = 'jane@example.com' WHERE personId = 2;

INSERT INTO Person VALUES (3, 'Jake', 'Doe', 'john@example.com');

**-- Question 9: Find Median Salary**

SELECT AVG(salary) AS MedianSalary

FROM (

SELECT salary

FROM Employee

ORDER BY salary

LIMIT 2 - (SELECT COUNT(\*) FROM Employee) % 2 OFFSET (SELECT (COUNT(\*) - 1) / 2 FROM Employee)

) AS Median;

**-- Question 10: Find Top N Salaries**

SELECT salary

FROM Employee

ORDER BY salary DESC

LIMIT N;

**-- Question 11: Count Employees in Each Department**

SELECT d.name AS Department, COUNT(e.employeeId) AS EmployeeCount

FROM Department d

LEFT JOIN Employee e ON d.departmentId = e.departmentId

GROUP BY d.name;

**-- Question 12: Self Join to Find Employee Pairs**

SELECT e1.name AS Employee1, e2.name AS Employee2

FROM Employee e1

JOIN Employee e2 ON e1.managerId = e2.employeeId;

**-- Question 13: Find All Managers**

SELECT DISTINCT m.name AS Manager

FROM Employee e

JOIN Employee m ON e.managerId = m.employeeId;

**-- Question 14: Department Salary Sum**

SELECT d.name AS Department, SUM(e.salary) AS TotalSalary

FROM Department d

JOIN Employee e ON d.departmentId = e.departmentId

GROUP BY d.name;

**-- Question 15: Delete Duplicate Rows**

WITH CTE AS (

SELECT id, ROW\_NUMBER () OVER (PARTITION BY name, salary ORDER BY id) AS rn

FROM Employee

)

DELETE FROM Employee

WHERE id IN (SELECT id FROM CTE WHERE rn > 1);

**-- Question 16: Find Consecutive Logins**

SELECT userId, loginDate

FROM (

SELECT userId, loginDate,

LAG(loginDate) OVER (PARTITION BY userId ORDER BY loginDate) AS prevLogin

FROM Logins

) t

WHERE DATEDIFF(day, prevLogin, loginDate) = 1;

CREATE TABLE Logins (

loginId INT PRIMARY KEY,

userId INT,

loginDate DATE

);

INSERT INTO Logins VALUES (1, 101, '2024-01-01'), (2, 101, '2024-01-02'), (3, 102, '2024-01-05');

**-- Question 17: Get the Maximum Difference in Salaries**

SELECT MAX(salary) - MIN(salary) AS MaxSalaryDifference

FROM Employee;

**-- Question 18: Employees with No Manager**

SELECT name

FROM Employee

WHERE managerId IS NULL;

**-- Question 19: Common Customers Between Two Tables**

SELECT c1. customerId

FROM Customers1 c1

INNER JOIN Customers2 c2 ON c1. customerId = c2.customerId;

CREATE TABLE Customers1 (

customerId INT PRIMARY KEY

);

CREATE TABLE Customers2 (

customerId INT PRIMARY KEY

);

INSERT INTO Customers1 VALUES (1), (2), (3);

INSERT INTO Customers2 VALUES (2), (3), (4);

**-- Question 20: Products Without Sales**

SELECT p.productId, p.name

FROM Products p

LEFT JOIN Sales s ON p.productId = s.productId

WHERE s.productId IS NULL;

CREATE TABLE Products (

productId INT PRIMARY KEY,

name VARCHAR (50)

);

CREATE TABLE Sales (

saleId INT PRIMARY KEY,

productId INT,

quantity INT

);

INSERT INTO Products VALUES (1, 'Laptop'), (2, 'Mouse');

INSERT INTO Sales VALUES (1, 1, 10);

**-- Question 21: Find Customers with Total Sales Above Threshold**

SELECT c.name, SUM(s.amount) AS TotalSales

FROM Customers c

JOIN Sales s ON c.customerId = s.customerId

GROUP BY c.name

HAVING SUM(s.amount) > 500;

CREATE TABLE Customers (

customerId INT PRIMARY KEY,

name VARCHAR (50)

);

ALTER TABLE Sales ADD customerId INT;

INSERT INTO Customers VALUES (1, 'Alice'), (2, 'Bob');

INSERT INTO Sales (saleId, productId, quantity, customerId) VALUES (2, 2, 5, 1), (3, 1, 3, 2);

**-- Remaining questions (22-45) will be outlined in the next update for brevity.**

**-- Question 22: Find Top 3 Highest Paid Employees in Each Department**

SELECT e.name, e.salary, d.name AS Department

FROM Employee e

JOIN Department d ON e.departmentId = d.departmentId

QUALIFY ROW\_NUMBER () OVER (PARTITION BY d.name ORDER BY e.salary DESC) <= 3;

**-- Question 23: Average Salary per Job Title**

SELECT jobTitle, AVG (salary) AS AverageSalary

FROM Employee

GROUP BY jobTitle;

ALTER TABLE Employee ADD jobTitle VARCHAR (50);

**-- Question 24: Employees Who Didn’t Receive a Bonus**

SELECT e.name

FROM Employee e

LEFT JOIN Bonuses b ON e.employeeId = b.employeeId

WHERE b.bonusAmount IS NULL;

CREATE TABLE Bonuses (

bonusId INT PRIMARY KEY,

employeeId INT,

bonusAmount DECIMAL (10, 2),

FOREIGN KEY (employeeId) REFERENCES Employee(employeeId)

);

**-- Question 25: Find Employees with Odd Salaries**

SELECT name

FROM Employee

WHERE MOD (salary, 2) = 1;

**-- Question 26: Retrieve Last Updated Records**

SELECT \*

FROM Employee

WHERE updateTime = (SELECT MAX (updateTime) FROM Employee);

ALTER TABLE Employee ADD updateTime DATETIME;

**-- Question 27: Employees Who Have Worked on All Projects**

SELECT e.name

FROM Employee e

WHERE NOT EXISTS (

SELECT p.projectId

FROM Projects p

WHERE NOT EXISTS (

SELECT \*

FROM WorksOn w

WHERE w.projectId = p.projectId AND w.employeeId = e.employeeId

)

);

CREATE TABLE Projects (

projectId INT PRIMARY KEY,

projectName VARCHAR (50)

);

CREATE TABLE WorksOn (

employeeId INT,

projectId INT,

FOREIGN KEY (employeeId) REFERENCES Employee(employeeId),

FOREIGN KEY (projectId) REFERENCES Projects(projectId)

);

**-- Question 28: Total Sales per Product**

SELECT p.name AS ProductName, SUM(s.quantity) AS TotalQuantitySold

FROM Products p

LEFT JOIN Sales s ON p.productId = s.productId

GROUP BY p.name;

**-- Question 29: Employees Working on Maximum Projects**

SELECT e.name

FROM Employee e

JOIN WorksOn w ON e.employeeId = w.employeeId

GROUP BY e.name

HAVING COUNT(w.projectId) = (

SELECT MAX(ProjectCount)

FROM (

SELECT employeeId, COUNT (projectId) AS ProjectCount

FROM WorksOn

GROUP BY employeeId

) AS SubQuery

);

**-- Question 30: Departments with No Employees**

SELECT d.name

FROM Department d

LEFT JOIN Employee e ON d.departmentId = e.departmentId

WHERE e.employeeId IS NULL;

**-- Question 31: Find Most Recent Hire in Each Department**

SELECT e.name, e.hireDate, d.name AS Department

FROM Employee e

JOIN Department d ON e.departmentId = d.departmentId

QUALIFY ROW\_NUMBER () OVER (PARTITION BY d.name ORDER BY e.hireDate DESC) = 1;

**-- Question 32: Find Employees with Above-Average Salary in Their Department**

SELECT e.name, e.salary, d.name AS Department

FROM Employee e

JOIN Department d ON e.departmentId = d.departmentId

WHERE e.salary > (

SELECT AVG (salary)

FROM Employee

WHERE departmentId = d.departmentId

);

**-- Question 33: Find Project Count per Department**

SELECT d.name AS Department, COUNT (DISTINCT w.projectId) AS ProjectCount

FROM Department d

JOIN Employee e ON d.departmentId = e.departmentId

JOIN WorksOn w ON e.employeeId = w.employeeId

GROUP BY d.name;

**-- Question 34: Employees with the Same Salary as Their Manager**

SELECT e.name AS Employee, m.name AS Manager, e.salary

FROM Employee e

JOIN Employee m ON e.managerId = m.employeeId

WHERE e.salary = m.salary;

**-- Question 35: Find Employees Not Assigned to Any Projects**

SELECT e.name

FROM Employee e

LEFT JOIN WorksOn w ON e.employeeId = w.employeeId

WHERE w.projectId IS NULL;

**-- Question 36: Total Sales Revenue Per Product**

SELECT p.name AS ProductName, SUM (s.quantity \* s.price) AS TotalRevenue

FROM Products p

JOIN Sales s ON p.productId = s.productId

GROUP BY p.name;

ALTER TABLE Sales ADD price DECIMAL (10, 2);

UPDATE Sales SET price = 100 WHERE productId = 1;

UPDATE Sales SET price = 50 WHERE productId = 2;

**-- Question 37: Retrieve Departments with All Employees Above a Threshold Salary**

SELECT d.name AS Department

FROM Department d

JOIN Employee e ON d.departmentId = e.departmentId

GROUP BY d.name

HAVING MIN(e.salary) > 50000;

**-- Question 38: Find Consecutive Absences for Employees**

SELECT e.name, a.absenceDate

FROM Absences a

JOIN Employee e ON a.employeeId = e.employeeId

WHERE EXISTS (

SELECT 1

FROM Absences a2

WHERE a2. employeeId = a.employeeId

AND DATEDIFF (day, a.absenceDate, a2.absenceDate) = 1

);

CREATE TABLE Absences (

absenceId INT PRIMARY KEY,

employeeId INT,

absenceDate DATE,

FOREIGN KEY (employeeId) REFERENCES Employee(employeeId)

);

INSERT INTO Absences VALUES (1, 1, '2024-01-01'), (2, 1, '2024-01-02'), (3, 2, '2024-01-05');

**-- Question 39: Find Managers with More than 5 Employees**

SELECT m.name AS Manager, COUNT(e.employeeId) AS EmployeeCount

FROM Employee e

JOIN Employee m ON e.managerId = m.employeeId

GROUP BY m.name

HAVING COUNT(e.employeeId) > 5;

**-- Question 40: Products Sold on the Most Days**

SELECT p.name AS ProductName

FROM Products p

JOIN Sales s ON p.productId = s.productId

GROUP BY p.name

ORDER BY COUNT(DISTINCT s.saleDate) DESC

LIMIT 1;

ALTER TABLE Sales ADD saleDate DATE;

UPDATE Sales SET saleDate = '2024-01-01' WHERE saleId = 1;

UPDATE Sales SET saleDate = '2024-01-02' WHERE saleId = 2;

**-- Question 41: Employees Assigned to All Projects in Their Department**

SELECT e.name

FROM Employee e

JOIN Department d ON e.departmentId = d.departmentId

WHERE NOT EXISTS (

SELECT p.projectId

FROM Projects p

WHERE p.departmentId = d.departmentId

AND NOT EXISTS (

SELECT 1

FROM WorksOn w

WHERE w.employeeId = e.employeeId

AND w.projectId = p.projectId

)

);

ALTER TABLE Projects ADD departmentId INT;

UPDATE Projects SET departmentId = 1 WHERE projectId = 1;

UPDATE Projects SET departmentId = 2 WHERE projectId = 2;

**-- Question 42: Find Products Never Sold**

SELECT p.name AS ProductName

FROM Products p

LEFT JOIN Sales s ON p.productId = s.productId

WHERE s.productId IS NULL;

**-- Question 43: Employees Who Worked on the Same Projects**

SELECT DISTINCT e1.name AS Employee1, e2.name AS Employee2

FROM WorksOn w1

JOIN WorksOn w2 ON w1. projectId = w2. projectId

JOIN Employee e1 ON w1. employeeId = e1. employeeId

JOIN Employee e2 ON w2. employeeId = e2. employeeId

WHERE e1. employeeId < e2. employeeId;

**-- Question 44: Find Highest Spending Customers**

SELECT c.name AS CustomerName, SUM (s.quantity \* s.price) AS TotalSpent

FROM Customers c

JOIN Sales s ON c.customerId = s.customerId

GROUP BY c.name

ORDER BY TotalSpent DESC

LIMIT 1;

**-- Question 45: Departments with No Ongoing Projects**

SELECT d.name AS Department

FROM Department d

LEFT JOIN Projects p ON d.departmentId = p.departmentId

WHERE p.projectId IS NULL;