-- ============================================

-- Rishabh\_Srivastava\_CPDA\_Batch 21.sql

-- Northwind SQL Assignment - 15 Questions

-- ============================================

/\* IMPORTANT INSTRUCTIONS FOR LEARNERS

1) DO NOT CHANGE THE ORDER OF COLUMNS.

2) YOUR QUERY SHOULD DISPLAY COLUMNS IN THE SAME ORDER AS MENTIONED IN ALL QUESTIONS.

3) YOU CAN FIND THE ORDER OF COLUMNS IN QUESTION TEMPLATE SECTION OF EACH QUESTION.

4) USE ALIASING AS MENTIONED IN QUESTION TEMPLATE FOR ALL COLUMNS

5) DO NOT CHANGE COLUMN NAMES\*/

-- Question 1 (Marks: 2)

-- Objective: Retrieve data using basic SELECT statements

-- List the names of all customers in the database.

-- Question Template: Display CustomerName Column

# SQL SOLUTION -01

SELECT CustomerName

FROM customers;

-- Simple SELECT fetching customer names.

-- Question 2 (Marks: 2)

-- Objective: Apply filtering using the WHERE clause

-- Retrieve the names and prices of all products that cost less than $15.

-- Question Template: Display ProductName Column

# SQL SOLUTION-02

SELECT ProductName

FROM products

WHERE Price < 15;

-- Filtering with WHERE condition.

-- Question 3 (Marks: 2)

-- Objective: Use SELECT to extract multiple fields

-- Display all employees first and last names.

-- Question Template: Display FirstName, LastName Columns

#SQL SOLUTION -03

SELECT FirstName, LastName

FROM employees;

-- Basic column selection.

-- Question 4 (Marks: 2)

-- Objective: Filter data using a function on date values

-- List all orders placed in the year 1997.

-- Question Template: Display OrderID, OrderDate Columns

#SQL SOLUTION-04

SELECT OrderID, OrderDate

FROM orders

WHERE YEAR(OrderDate) = 1997;

-- Used YEAR() function to extract year from OrderDate.

-- Question 5 (Marks: 2)

-- Objective: Apply numeric filters

-- Retrieve the names of all products that are currently in stock (UnitsInStock > 0).

-- Question Template: Display ProductName, Price Column

#SQL SOLUTION-05

SELECT ProductName

FROM products

WHERE Unit > 0;

-- Condition applied on stock column (Unit).

-- Question 6 (Marks: 3)

-- Objective: Perform multi-table JOIN operations

-- Show the names of customers and the names of the employees who handled their orders.

-- Question Template: Display CustomerName, FirstName, LastName Columns

#SQL SOLUTION-06

SELECT c.CustomerName, e.FirstName, e.LastName

FROM orders o

INNER JOIN customers c ON o.CustomerID = c.CustomerID

INNER JOIN employees e ON o.EmployeeID = e.EmployeeID;

-- Multi-table JOIN query.

-- Question 7 (Marks: 3)

-- Objective: Use GROUP BY for aggregation

-- List each country along with the number of customers from that country.

-- Question Template: Display Country, CustomerCount Columns

# SQL Solution-07

SELECT Country, COUNT(\*) AS NumberOfCustomers

FROM customers

GROUP BY Country;

-- Aggregation with GROUP BY.

-- Question 8 (Marks: 3)

-- Objective: Group data by a foreign key relationship and apply aggregation

-- Find the average price of products grouped by category.

-- Question Template: Display CategoryName, AvgPrice Columns

-- SQL SOLUTION 08

SELECT cat.CategoryName, AVG(p.Price) AS AvgPrice

FROM products p

INNER JOIN categories cat ON p.CategoryID = cat.CategoryID

GROUP BY cat.CategoryName;

-- GROUP BY foreign key relationship.

-- Question 9 (Marks: 3)

-- Objective: Use aggregation to count records per group

-- Show the number of orders handled by each employee.

-- Question Template: Display EmployeeID, OrderCount Columns

#SQL SOLUTION-09

SELECT e.EmployeeID, COUNT(o.OrderID) AS OrderCount

FROM employees e

INNER JOIN orders o ON e.EmployeeID = o.EmployeeID

GROUP BY e.EmployeeID, e.FirstName, e.LastName;

-- Count orders per employee.

-- Question 10 (Marks: 3)

-- Objective: Filter results using values from a joined table

-- List the names of products supplied by "Exotic Liquids".

-- Question Template: Display ProductName Column

#SQL SOLUTION-10

SELECT p.ProductName

FROM products p

INNER JOIN suppliers s ON p.SupplierID = s.SupplierID

WHERE s.SupplierName = 'Exotic Liquid';

-- Filtering using a joined table.

-- Question 11 (Marks: 5)

-- Objective: Rank records using aggregation and sort

-- List the top 3 most ordered products (by quantity).

-- Question Template: Display ProductID, TotalOrdered Columns

#SQL SOLUTION-11

SELECT p.ProductID, SUM(od.Quantity) AS TotalOrdered

FROM orderdetails od

INNER JOIN products p ON od.ProductID = p.ProductID

GROUP BY p.ProductID

ORDER BY TotalOrdered DESC

LIMIT 3;

-- Ranking by aggregation and sorting.

-- Question 12 (Marks: 5)

-- Objective: Use GROUP BY and HAVING to filter on aggregates

-- Find customers who have placed orders worth more than $10,000 in total.

-- Question Template: Display CustomerName, TotalSpent Columns

#SQL SOLUTION-12

SELECT c.CustomerName, SUM(od.Quantity \* p.Price) AS TotalSpent

FROM customers c

INNER JOIN orders o ON c.CustomerID = o.CustomerID

INNER JOIN orderdetails od ON o.OrderID = od.OrderID

INNER JOIN products p ON od.ProductID = p.ProductID

GROUP BY c.CustomerName

HAVING SUM(od.Quantity \* p.Price) > 10000;

-- GROUP BY + HAVING for filtering aggregated values.

-- Question 13 (Marks: 5)

-- Objective: Aggregate and filter at the order level

-- Display order IDs and total order value for orders that exceed $2,000 in value.

-- Question Template: Display OrderID, OrderValue Columns

#SQL SOLUTION-13

SELECT o.OrderID, SUM(od.Quantity \* p.Price) AS OrderValue

FROM orders o

INNER JOIN orderdetails od ON o.OrderID = od.OrderID

INNER JOIN products p ON od.ProductID = p.ProductID

GROUP BY o.OrderID

HAVING SUM(od.Quantity \* p.Price) > 2000;

-- Aggregation at order level with HAVING.

-- Question 14 (Marks: 5)

-- Objective: Use subqueries in HAVING clause

-- Find the name(s) of the customer(s) who placed the largest single order (by value).

-- Question Template: Display CustomerName, OrderID, TotalValue Column

#SQL SOLUTION-14

SELECT c.CustomerName, o.OrderID, SUM(od.Quantity \* p.Price) AS TotalValue

FROM customers c

INNER JOIN orders o ON c.CustomerID = o.CustomerID

INNER JOIN orderdetails od ON o.OrderID = od.OrderID

INNER JOIN products p ON od.ProductID = p.ProductID

GROUP BY c.CustomerName, o.OrderID

HAVING SUM(od.Quantity \* p.Price) = (

-- Subquery to calculate the maximum order value across all orders

SELECT MAX(OrderTotal)

FROM (

SELECT SUM(od2.Quantity \* p2.Price) AS OrderTotal

FROM orderdetails od2

INNER JOIN products p2 ON od2.ProductID = p2.ProductID

GROUP BY od2.OrderID

) AS Subquery

);

-- Question 15 (Marks: 5)

-- Objective: Identify records using NOT IN with subquery

-- Get a list of products that have never been ordered.

-- Question Template: Display ProductName Columns

#SQL SOLUTION-15

SELECT p.ProductName

FROM products p

WHERE p.ProductID NOT IN (

SELECT DISTINCT ProductID

FROM orderdetails

);

-- Subquery with NOT IN to exclude ordered products.

# Thank You