You can find this on my github: <https://github.com/princesscorn/assignments-antra.git>

# Write queries for following scenarios

All scenarios are based on Database NORTHWND

## 1.      List all cities that have both Employees and Customers.

Solution:

SELECT DISTINCT city FROM Customers WHERE city IN (SELECT city FROM Employees)

## 2.      List all cities that have Customers but no Employee.

a.      Use sub-query

b.      Do not use sub-query

Solutions:

a. SELECT DISTINCT City FROM Customers

WHERE city NOT IN (SELECT DISTINCT City FROM Employees)

b. SELECT DISTINCT City FROM Customers

EXCEPT

SELECT DISTINCT City FROM Employees

## 3.      List all products and their total order quantities throughout all orders.

Solution:

SELECT DISTINCT ProductID, SUM(Quantity) AS TotalOrderQty

FROM [Order Details]

GROUP BY ProductID

## 4.      List all Customer Cities and total products ordered by that city.

Solution

(Both LEFT JOIN or INNER JOIN are fine for here)

SELECT c.City, SUM(od.Quantity) AS TotalProducts

FROM Customers c LEFT JOIN Orders o ON c.CustomerID = o.CustomerID

INNER JOIN [Order Details] od ON o.OrderID = od.OrderID

GROUP BY c.City

## 5.      List all Customer Cities that have at least two customers.

a.      Use union

b.      Use sub-query and no union

Solutions:

a. SELECT City from Customers

EXCEPT

(

SELECT City FROM customers GROUP BY City HAVING COUNT(\*)=1

UNION

SELECT City FROM customers GROUP BY City HAVING COUNT(\*)=0

)

b. SELECT c.City FROM Customers c

GROUP BY c.City

HAVING COUNT(c.CustomerID) >= 2

6.      List all Customer Cities that have ordered at least two different kinds of products.

Solution:

SELECT DISTINCT City

FROM Orders o JOIN [Order Details] od ON o.OrderID =od.OrderID

JOIN Customers c ON c.Customerid=o.CustomerID

GROUP BY City

HAVING COUNT(\*)>=2

ORDER BY City

## 7.      List all Customers who have ordered products, but have the ‘ship city’ on the order different from their own customer cities.

Solution:

SELECT DISTINCT c.CustomerID, c.ContactName, c.City, o.ShipCity

FROM Customers c JOIN Orders o ON c.CustomerID = o.CustomerID

WHERE c.City <> o.ShipCity

## 8.      List 5 most popular products, their average price, and the customer city that ordered most quantity of it.

Solution

SELECT TOP 5 ProductID, AVG(UnitPrice) AS AvgPrice,

(

SELECT TOP 1 City

FROM Customers c JOIN Orders o ON o.CustomerID=c.CustomerID

JOIN [Order Details] od2 ON od2.OrderID=o.OrderID

WHERE od2.ProductID = od1.ProductID

GROUP BY City ORDER BY SUM(Quantity) DESC

) AS City

FROM [Order Details] od1

GROUP BY ProductID

ORDER BY SUM(Quantity) DESC

## 9.      List all cities that have never ordered anything, but we have employees there.

a.      Use sub-query

b.      Do not use sub-query

Solutions:

a.

SELECT e.City FROM Employees e WHERE City NOT IN

(SELECT DISTINCT c.CITY

FROM Customers c LEFT JOIN Orders o

ON c.CustomerID = o.CustomerID)

b.

SELECT e.City

FROM Employees e

EXCEPT

SELECT DISTINCT c.City FROM Customers c LEFT JOIN Orders o ON c.CustomerID = o.CustomerID

## 10.  List one city, if exists, that is the city from where the employee sold most orders (not the product quantity) is, and also the city of most total quantity of products ordered from. (tip: join  sub-query)

Solution:

--The city of Employee who sell most orders

SELECT TOP 1 e.EmployeeID, e.City AS EmployeeCity, COUNT(o.OrderID) as NumOfSoldOrders

FROM Employees e INNER JOIN Orders o ON e.EmployeeID = o.EmployeeID

GROUP BY e.EmployeeID, e.City

ORDER BY COUNT(o.OrderID) DESC

INTERSECT

--The city of Customer who order most quantity product

SELECT TOP 1 c.City AS CustomerCity, SUM(od.Quantity) AS NumOfOrders

FROM Customers c INNER JOIN Orders o ON c.CustomerID = o.CustomerID

INNER JOIN [Order Details] od ON o.OrderID = od.OrderID

GROUP BY c.CustomerID, c.ContactName, c.City

ORDER BY SUM(od.Quantity)

## 11. How do you remove the duplicates record of a table?

Solution:

[Remove duplicate rows from a table in SQL Server - SQL Server | Microsoft Docs](https://docs.microsoft.com/en-us/troubleshoot/sql/database-design/remove-duplicate-rows-sql-server-tab)

1) Using Group By and having clause

SELECT DISTINCT \*

INTO duplicate\_table

FROM original\_table

GROUP BY key\_value

HAVING COUNT(key\_value) > 1

DELETE original\_table

WHERE key\_value

IN (SELECT key\_value

FROM duplicate\_table)

INSERT original\_table

SELECT \*

FROM duplicate\_table

DROP TABLE duplicate\_table

2) Using ROW\_NUMBER function

DELETE T

FROM

(

SELECT \*, DupRank = ROW\_NUMBER() OVER (

PARTITION BY key\_value

ORDER BY (SELECT NULL)

)

FROM original\_table

) AS T

WHERE DupRank > 1

3) Using Common Table Expressions (CTE)

4) Use SSIS package to SQL delete duplicate rows

# Test

## 1. What is CTE and when to use it?

Solution:

CTE stands for Common Table Expression, which is a temporary named result set that you can reference within SELECT, INSERT, UPDATE, and DELETE. CTE can also be used in a View.

CTE allows you to create and use a query just before your main query.

## 2. What are Table Variables? What is their scope and where are they created in SQL Server?

Solution:

Table Variables is a data type that can be used in T-SQL batch, stored procedure, or function.

The scope is the current T-SQL batch, stored procedure, or function.

Table Variables are stored in Tempdb.

## 3. What's the difference between RANK() and DENSE\_RANK()?

Solution:

RANK() and DENSE\_RANK() are similar to ROW\_NUMBER(), but when there are ties, they will give the same value to the tied values, then

RANK() will keep the ranking, so the number may go like 1, 2, 2, 4

DENSE\_RANK() will never give any gaps, so the number may go like 1, 2, 2, 3

## 4. What is the difference between Union and Union All?

Solution:

1) Union will automatically remove the duplicate records, Union All will not.

2) Union will sort the result by the order of the first column by default, Union All will not.

3) Union All’s performance is better than Union.

4) Union All can be used in CTEs, Union cannot.

## 5. What is View? What are the benefits of using views?

Solution:

View is a virtual table, whose contents are defined by a query.

View is not allowed on temporary tables.

Benefits:

1) Simplify the data manipulation.

2) Allows for the backward-compatible interface.

3) Customize data.

## 6. What's the difference between Local Temp table and Global Temp table?

Solution:

1) Local Temp table name is stared with single hash ("#") sign.

Global Temp table name is stared with double hashes (“##”) sign.

2) Local Temp tables are visible only in the current session.

Global Temp tables are visible to all sessions.

## 7. What's the difference between table variables and temp tables?

Solution:

1) Storage: Both Table variables(@tableVar) and Temp tables(#tmpTable) are stored in tempdb.

2) Scope: Table Variables scoped for current batch,

Temp Tables scoped local/global.

3) Scenario: Table variables for small data (like rows < 1000)

Temp tables for big data.

4) Usage: Table variables can be passed as a parameter to functions and stored procedures,

while the same cannot be done with Temp tables.

5) Structure: Temp tables are allowed CREATE INDEX except FOREIGN KEY.

Table variables can have index by using Primary Key/Unique Constraint.

6) Performance: Table variables are faster than temp tables.

## 8. What is SQL Injection and how can we prevent it?

Solution:

SQL Injection is a web vulnerability caused by mistakes made by programmers. It allows an attacker to send commands to the database that the website or web application communicates with. This, in turn, lets the attacker get data from the database or even modify it.

The only sure way to prevent SQL Injection attacks is input validation and parametrized queries including prepared statements. The application code should never use the input directly. The developer must sanitize all input, not only web form inputs such as login forms.