AdventureWorks Database – 2019

Get the entity relationship diagram: Click here

1. The Sales.SalesOrderHeader table contains foreign keys to the Sales.CurrencyRate and Purchasing.ShipMethod tables. Write a query joining all three tables, and make sure it contains all rows from Sales.SalesOrderHeader. Include the CurrencyRateID, AverageRate, SalesOrderID, and ShipBase columns.

```
SELECT SOH.CurrencyRateID, CR.AverageRate, SOH.SalesOrderID, SM.ShipBase
  FROM Sales.SalesOrderHeader AS SOH
  LEFT JOIN Sales.CurrencyRate AS CR ON SOH.CurrencyRateID = CR.CurrencyRateID
  LEFT JOIN Purchasing.ShipMethod AS SM ON SOH.ShipMethodID = SM.ShipMethodID
```

2. Using a derived table, join the Sales.SalesOrderHeader table to the Sales.SalesOrderDetail table. Display the SalesOrderID, OrderDate, and ProductID columns in the results. The Sales. SalesOrderDetail table should be inside the derived table.

```
SELECT H.SalesOrderID, H.OrderDate, D.ProductID
  FROM Sales.SalesOrderHeader AS H
  JOIN (SELECT SalesOrderID, ProductID FROM Sales.SalesOrderDetail) AS D ON
D.SalesOrderID = H.SalesOrderID
```

3. Write a query that returns a list of the products where the subcategory or the product name contains the word Mountain. Display only those rows where the color is silver. Return the subcategory and product names, the product ID, and the color information.

```
SELECT p.Name, p.ProductID, p.Color
  FROM Production.Product p
  JOIN Production.ProductSubcategory s ON s.ProductSubCategoryID =
p.ProductSubcategoryID
 WHERE (p.Name LIKE '%Mountain%' OR s.Name = '%Mountain%')
   AND p.Color = 'Silver'
```

4. Write a query joining the Person.Person, Sales.Customer, and Sales.SalesOrderHeader tables to return a list of the customer names along with a count of the orders placed .

```
SELECT CONCAT(P.FirstName, ' ', P.LastName) AS Customername,
       COUNT(SOH.SalesOrderID) AS orders_placed
  FROM Sales.SalesOrderHeader SOH
  JOIN Sales.Customer C ON SOH.CustomerID = C.CustomerID
  JOIN Person.Person P ON P.BusinessEntityID = C.CustomerID
 GROUP BY P.BusinessEntityID, P.FirstName, P.LastName
 ORDER BY 2 DESC;
```

5. Write a query using the Sales.SalesOrderHeader, Sales. SalesOrderDetail, and Production.Product tables to display the total sum of products by Name and OrderDate .

```
SELECT P.Name, SOH.OrderDate, SUM(SOD.OrderQty) AS 'Total Quantity'
  FROM Sales.SalesOrderHeader SOH
  JOIN Sales.SalesOrderDetail SOD ON SOH.SalesOrderID = SOD.SalesOrderID
  JOIN Production.Product P ON P.ProductID = SOD.ProductID
 GROUP BY P.ProductID, P.Name, SOH.OrderDate
```

6. Using a subquery that includes the Sales.SalesOrderDetail table, display the product names and product ID numbers from the Production.Product table that have been ordered.

```
SELECT Name, ProductID
  FROM Production.Product
 WHERE ProductID IN (SELECT DISTINCT ProductID FROM Sales.SalesOrderDetail)
```

7. Change the query written in Question 1 to display the products that have not been ordered.

```
SELECT Name, ProductID
  FROM Production.Product
 WHERE ProductID NOT IN (SELECT DISTINCT ProductID FROM Sales.SalesOrderDetail)
```

8. Can we use EXCEPT/MINUS for Q7.

```
SELECT Name, ProductID
FROM Production.Product
EXCEPT
SELECT Name, ProductID
FROM Production.Product
WHERE ProductID IN (SELECT DISTINCT ProductID FROM Sales.SalesOrderDetail)
```

9. Write a query using a subquery that returns the rows from the Production.ProductColor table that are not being used in the Production.Product table.

```
SELECT color
  FROM Production.ProductColor
 WHERE color NOT IN (SELECT DISTINCT Color
                       FROM Production.Product
                      WHERE color IS NOT NULL);
```

10. Can we use EXCEPT/MINUS for Q9.

```
SELECT color
FROM Production.ProductColor
EXCEPT
SELECT DISTINCT Color
FROM Production.Product
```

```
WHERE color IS NOT NULL
```

11. Write a query that displays the colors used in the Production.Product table that are not listed in the Production.ProductColor table using a subquery. Use the keyword DISTINCT before the column name to return each color only once. Use the NOT EXISTS method in this query.

```
SELECT color
FROM Production.Product AS P
WHERE NOT EXISTS (SELECT DISTINCT PC.color
                    FROM Production.ProductColor AS PC
                   WHERE PC.Color = P.Color)
```

12. Write a query that combines the ModifiedDate from Person.Person and the HireDate from HumanResources.Employee with no duplicates in the results.

```
SELECT ModifiedDate
  FROM Person.Person
 UNION
SELECT HireDate
  FROM HumanResources.Employee
```

13. Get all BusinessEntityID which exist in both table HumanResources.Employee, Person.Person;

```
SELECT BusinessEntityID
FROM HumanResources.Employee
INTERSECT
SELECT BusinessEntityID
FROM Person.Person
```

14. Get all BusinessEntityID which exist in table HumanResources.Employee or Person.Person:

```
SELECT BusinessEntityID
FROM HumanResources.Employee
UNION ALL
SELECT BusinessEntityID
FROM Person.Person
```

15. Get all First name of Employee and Customer without duplicate (hint: Using Employee, Customer and Person tables)

```
SELECT P.FirstName
FROM Sales.Customer C
JOIN Person.Person P ON C.CustomerID = P.BusinessEntityID
UNION
SELECT P.FirstName
FROM HumanResources.Employee E
```

```sql
JOIN Person.Person P ON E.BusinessEntityID = P.BusinessEntityID
```

16. Using a derived table, join the Sales.SalesOrderHeader table to the Sales.SalesOrderDetail table. Display the SalesOrderID, OrderDate, and ProductID columns in the results. The Sales.SalesOrderDetail table should be inside the derived table query.

```sql
SELECT H.SalesOrderID, H.OrderDate, D.ProductID
FROM Sales.SalesOrderHeader AS H
JOIN (SELECT SalesOrderID, ProductID
FROM Sales.SalesOrderDetail) AS D ON D.SalesOrderID = H.SalesOrderID
```

17. Rewrite the query in Question 16 with a common table expression.

```sql
WITH SalesOrderDetailCTE AS (
SELECT ProductID, SalesOrderID
FROM Sales.SalesOrderDetail
)
SELECT SOH.SalesOrderID, SOH.OrderDate, CTE.ProductID
FROM Sales.SalesOrderHeader SOH
JOIN SalesOrderDetailCTE CTE ON CTE.SalesOrderID = SOH.SalesOrderID
```

18. Write a query that displays all customers along with the orders placed in 2011. Use a common table expression to write the query and include the CustomerID, SalesOrderID, and OrderDate columns in the results.

```sql
WITH CustomerOrderCTE AS (
SELECT C.CustomerID, SOH.SalesOrderID, SOH.OrderDate
FROM Sales.SalesOrderHeader SOH
JOIN sales.Customer C ON C.CustomerID = SOH.CustomerID
WHERE YEAR(OrderDate) = 2011
)
SELECT C.CustomerID, CTE.SalesOrderID, CTE.OrderDate
FROM Sales.Customer AS C
JOIN CustomerOrderCTE CTE ON C.CustomerID = CTE.CustomerID;
```