AdventureWork - 2019 Database

Get the entity relationship diagram: Click here

1. CREATE Table called DemoProduct including 7 columns (ProductID, Name, color, StandardCost, ListPrice, Size, Weight).

```
DROP TABLE IF EXISTS [dbo].[demoProduct]
GO
CREATE TABLE [dbo].[demoProduct](
 [ProductID] [INT] NOT NULL PRIMARY KEY,
 [Name] [NVARCHAR](100) NOT NULL,
 [Color] [NVARCHAR](15) NULL,
 [StandardCost] [MONEY] NOT NULL,
 [ListPrice] [MONEY] NOT NULL,
 [Size] [NVARCHAR](5) NULL,
 [Weight] [DECIMAL](8, 2) NULL,
);
```

2. CREATE Table called DemoSalesOrderHeader including (SalesOrderID, SalesID, OrderDate, CustomerID, Subtotal, TaxAmt, Freight, DateEntered, TotalDue, Rowersion).

```
DROP TABLE IF EXISTS [dbo].[demoSalesOrderHeader]
GO
CREATE TABLE [dbo].[demoSalesOrderHeader](
 [SalesOrderID] [INT] NOT NULL PRIMARY KEY,
 [SalesID] [INT] NOT NULL IDENTITY,
 [OrderDate] [DATETIME] NOT NULL,
 [CustomerID] [INT] NOT NULL,
 [SubTotal] [MONEY] NOT NULL,
 [TaxAmt] [MONEY] NOT NULL,
 [Freight] [MONEY] NOT NULL,
 [DateEntered] [DATETIME],
 [TotalDue] AS (ISNULL(([SubTotal]+[TaxAmt])+[Freight],(0))),
 [RV] ROWVERSION NOT NULL);
GO
```

3. CREATE table DemoAddress including (AddressID, AdressLine1, AdressLine2, city, PostalCode).

```
DROP TABLE IF EXISTS [dbo].[demoAddress]
GO
CREATE TABLE [dbo].[demoAddress](
 [AddressID] [INT] NOT NULL IDENTITY PRIMARY KEY,
 [AddressLine1] [NVARCHAR](60) NOT NULL,
 [AddressLine2] [NVARCHAR](60) NULL,
 [City] [NVARCHAR](30) NOT NULL,
 [PostalCode] [NVARCHAR](15) NOT NULL
);
```

4. Write a SELECT statement to retrieve data from the Production.Product table. Use these values to insert five rows into the dbo.demoProduct table using literal values. Write five individual INSERT statements. The rows you choose to insert will vary.

```
INSERT INTO dbo.demoProduct
     VALUES (1, 'Adjustable Race', NULL, 0.00, 0.00, NULL, NULL)
INSERT INTO dbo.demoProduct
     VALUES (2, 'Bearing Ball', NULL, 0.00, 0.00, NULL, NULL)
INSERT INTO dbo.demoProduct
     VALUES (3, 'BB Ball Bearing', NULL, 0.00, 0.00, NULL, NULL)
INSERT INTO dbo.demoProduct
     VALUES (4, 'Headset Ball Bearings', NULL, 0.00, 0.00, NULL, NULL)
INSERT INTO dbo.demoProduct
     VALUES (5, 'Blade', NULL, 0.00, 0.00, NULL, NULL)
```

5. Insert five more rows using literal values into the dbo.demoProduct table. This time, write one INSERT statement. The rows you choose to insert will vary.

```
INSERT INTO dbo.demoProduct
     VALUES (317, 'LL Crankarm', 'Black', 0.00, 0.00, NULL, NULL),
            (318, 'ML Crakarm', 'Black', 0.00, 0.00, NULL, NULL),
            (319, 'HL Crankarm', 'Black', 0.00, 0.00, NULL, NULL),
            (320, 'LL Chainring Bolts', 'Silver', 0.00, 0.00, NULL, NULL),
            (321, 'Chainring Nut', 'Silver', 0.00, 0.00, NULL, NULL);
```

6. Write an INSERT statement that inserts all the rows into the dbo.demoSalesOrderHeader table from the Sales.SalesOrderHeader table.

```
INSERT INTO dbo.demoSalesOrderHeader (SalesOrderID, OrderDate, CustomerID, SubTotal,
TaxAmt, Freight)
SELECT SalesOrderID, OrderDate, CustomerID, Subtotalm, TaxAmt, Freight
  FROM Sales.SalesOrderHeader;
```

7. Write a SELECT INTO statement that creates a table, dbo. tempCustomerSales, inserting every CustomerID from the Sales. Customer along with a count of the orders placed and the total amount due for each customer.

```
SELECT c.CustomerID, COUNT(s.SalesOrderID) AS TotalOrder, SUM(s.TotalDue) AS
TotalPrice
FROM sales.Customer AS c
LEFT JOIN dbo.demoSalesOrderHeader s ON s.CustomerID = c.CustomerID
GROUP BY c.CustomerID
ORDER BY 1;
```

8. Write an INSERT statement that inserts all the products into the dbo. demoProduct table from the Production.Product table that have not already been inserted.

```
INSERT INTO dbo.demoProduct
SELECT ProductID, Name, Color, StandardCost, ListPrice, Weight
FROM Production.Product
WHERE ProductID NOT IN (1,2,3,4,316,317,318,319,320,321);
```

9. Write an INSERT statement that inserts all the addresses into the dbo.demoAddress table from the Person.Address table joined to the Person.StateProvince table. Before running the INSERT statement, type in and run the following command so that you can insert values into the AddressID column: SET IDENTITY_INSERT dbo.demoAddress ON;

```
SET IDENTITY_INSERT dbo.demoAddress ON;
INSERT INTO dbo.demoAddress (AddressID, AddressLine1, AddressLine2, City, PostalCode)
SELECT p.AddressID, p.AddressLine1, p.AddressLine2, p.City, p.PostalCode
FROM Person.Address AS p
JOIN Person.StateProvince AS s ON s.StateProvinceID = p.StateProvinceID
```

10. RECREATE copied table called dbo.DemoProduct from Product table.

```
DROP TABLE IF EXISTS [dbo].[demoProduct];
GO
SELECT * INTO dbo.demoProduct FROM Production.Product;
```

11. RECREATE table called demoCustomer include all columns from Sales.Customer table and LastName & FirstName from Person.Person table.

```
DROP TABLE IF EXISTS [dbo].[demoCustomer];
GO
SELECT C.*, LastName, FirstName
INTO dbo.demoCustomer
FROM Sales.Customer AS C
JOIN Person.Person AS P ON C.CustomerID = P.BusinessEntityID;
```

12. RECREATE copied table called demoSalesOrderHeader including all columns from Sales.SalesOrderHeader.

```
DROP TABLE IF EXISTS [dbo].[demoSalesOrderHeader];
GO
SELECT * INTO dbo.demoSalesOrderHeader FROM Sales.SalesOrderHeader;
DROP TABLE IF EXISTS [dbo].[demoSalesOrderDetail];
```

13. RECREATE copied table called demoSalesOrderDetail including all columns from Sales.SalesOrderDetail

```
DROP TABLE IF EXISTS [dbo].[demoSalesOrderDetail];
 GO
SELECT * INTO dbo.demoSalesOrderDetail FROM Sales.SalesOrderDetail;
```

14. Write a query that deletes the rows from the dbo.demoCustomer table where the LastName values begin with the letter S.

```
DELETE FROM dbo.demoCustomer
WHERE LastName LIKE 'S%';
```

15. Delete the rows from the dbo.demoCustomer table if the sum of the TotalDue from the dbo.demoSalesOrderHeader table for the customer is less than $1000. Hint: Use a subquery.

```
DELETE FROM dbo.demoCustomer
WHERE CustomerID IN (SELECT c.CustomerID
                       FROM dbo.democustomer c
                       JOIN dbo.demoSalesOrderHeader s ON s.CustomerID = c.CustomerID
                      GROUP BY c.CustomerID
                      HAVING SUM(s.TotalDue) < 1000.0);
```

16. Write an UPDATE statement that changes all NULLs of the AddressLine2 column in the dbo.demoAddress table to N/A.

```
UPDATE dbo.demoAddress
SET AddressLine2 = ISNULL(AddressLine2, 'N/A')
```

17.Write an UPDATE statement that increases the ListPrice of every product in the dbo.demoProduct table by 10%.

```
UPDATE dbo.demoProduct
SET ListPrice = ListPrice * 1.1;
```

18.Write an UPDATE statement that corrects the UnitPrice with the ListPrice of each row of the dbo.demoSalesOrderDetail table by joining the table on the dbo.demoProduct table.

```
UPDATE dbo.demoSalesOrderDetail
SET UnitPrice = p.ListPrice
FROM dbo.demoSalesOrderDetail o
JOIN dbo.demoProduct p ON p.ProductID = o.ProductID;
```