**SQL Assignment**

**Q.1**

**1**. Display the number of records in the [SalesPerson] table. (Schema(s) involved: Sales)

Ans- select \* from Sales.SalesPerson;

select COUNT(\*) from Sales.SalesPerson;

**2.** Select both the FirstName and LastName of records from the Person table where the FirstName begins with the letter ‘B’. (Schema(s) involved: Person)

Ans- select FirstName, LastName from Person.Person where FirstName like 'b%';

**3.** Select a list of FirstName and LastName for employees where Title is one of Design Engineer, Tool Designer or Marketing Assistant. (Schema(s) involved: HumanResources, Person)

Ans- select Person.FirstName, Person.LastName from Person.Person full outer join HumanResources.Employee on Person.BusinessEntityID=Employee.BusinessEntityID

where JobTitle='Design Engineer' or JobTitle='Tool Designer' or JobTitle='Marketing Assistant';

**4.** Display the Name and Color of the Product with the maximum weight. (Schema(s) involved: Production)

Ans- select Name, Color, Weight from Production.Product where Weight=(select max(Weight) from Production.Product);

**5.** Display Description and MaxQty fields from the SpecialOffer table. Some of the MaxQty values are NULL, in this case display the value 0.00 instead. (Schema(s) involved: Sales)

Ans- select Description, isnull(MaxQty, 0.0) as MaxQty from Sales.SpecialOffer;

**6.** Display the overall Average of the [CurrencyRate].[AverageRate] values for the exchange rate ‘USD’ to ‘GBP’ for the year 2005 i.e. FromCurrencyCode = ‘USD’ and ToCurrencyCode = ‘GBP’. Note: The field [CurrencyRate].[AverageRate] is defined as 'Average exchange rate for the day.' (Schema(s) involved: Sales)

Ans- SELECT AVG(AverageRate) AS 'Average Exchange Rate'

FROM Sales.CurrencyRate

WHERE FromCurrencyCode = 'USD'

AND ToCurrencyCode = 'GBP'

AND YEAR(CurrencyRateDate) = 2012;

**7.** Display the FirstName and LastName of records from the Person table where FirstName contains the letters ‘ss’. Display an additional column with sequential numbers for each row returned beginning at integer 1. (Schema(s) involved: Person)

Ans- SELECT ROW\_NUMBER() OVER(ORDER BY FirstName, LastName) AS 'Sr. No.', FirstName, LastName

FROM Person.Person

WHERE FirstName LIKE '%ss%'

**8.** Sales people receive various commission rates that belong to 1 of 4 bands. (Schema(s) involved: Sales)

Ans- select BusinessEntityID as 'SalesPersonID',

case

when CommissionPct = 0.0 then 'Band0'

when CommissionPct > 0.0 and CommissionPct < 0.01 then 'Band1'

when CommissionPct > 0.01 and CommissionPct < 0.015 then 'Band2'

when CommissionPct > 0.015 then 'Band3'

end as 'CommissionPct'

from Sales.SalesPerson;

**9.** Display the managerial hierarchy from Ruth Ellerbrock (person type – EM) up to CEO Ken Sanchez. **Hint**: use [uspGetEmployeeManagers] (*Schema(s) involved: [Person], [HumanResources]*)

Ans-

DECLARE @ID int;

SELECT @ID = BusinessEntityID

FROM Person.Person

WHERE FirstName = 'Ruth'

AND LastName = 'Ellerbrock'

AND PersonType = 'EM';

EXEC dbo.uspGetEmployeeManagers @BusinessEntityID = @ID;

**10.** Display the ProductId of the product with the largest stock level. **Hint**: Use the Scalar-valued function [dbo]. [UfnGetStock]. (*Schema(s) involved: Production*)

Ans-

SELECT MAX(dbo.ufnGetStock(ProductID)) FROM Production.Product;

**Q.2** Write separate queries using a join, a subquery, a CTE, and then an EXISTS to list all AdventureWorks customers who have not placed an order.

**Ans-**

select Customer.CustomerID from Sales.SalesOrderHeader right join Sales.Customer on Customer.CustomerID=SalesOrderHeader.CustomerID where SalesOrderID is null;

* select CustomerID from Sales.Customer

where CustomerID not in (select CustomerID from Sales.SalesOrderHeader);

* with Cte

as

(

select Customer.CustomerID from Sales.SalesOrderHeader

right join Sales.Customer on SalesOrderHeader.CustomerID=Customer.CustomerID

where SalesOrderID IS NULL

)

select \* from cte;

* Exists

**Q.3** Show the most recent five orders that were purchased from account numbers that have spent more than $70,000 with AdventureWorks.

**Ans-**

select top 5 SalesOrderID,OrderDate,AccountNumber from Sales.SalesOrderHeader

where AccountNumber in

(select AccountNumber from Sales.SalesOrderHeader group by AccountNumber having sum(SubTotal)>70000)

order by OrderDate desc;

**Q.4** Create a function that takes as inputs a SalesOrderID, a Currency Code, and a date, and returns a table of all the SalesOrderDetail rows for that Sales Order including Quantity, ProductID, UnitPrice, and the unit price converted to the target currency based on the end of day rate for the date provided. Exchange rates can be found in the Sales.CurrencyRate table. (Use AdventureWorks)

Ans-

GO

CREATE FUNCTION Sales.getProducts(@SalesOrderID int, @CurrencyCode nchar(3) ,@CurrencyRateDate datetime)

RETURNS TABLE

AS

RETURN

WITH Products

AS (

SELECT \*

FROM Sales.SalesOrderDetail AS sod

WHERE sod.SalesOrderID = @SalesOrderID

)

SELECT p.ProductID, p.OrderQty, p.UnitPrice, p.UnitPrice\*scr.EndOfDayRate AS 'Converted Price'

FROM Products AS p, Sales.CurrencyRate AS scr

WHERE scr.ToCurrencyCode = @CurrencyCode

AND scr.CurrencyRateDate = @CurrencyRateDate

GO

**Q.5** Write a Procedure supplying name information from the Person.Person table and accepting a filter for the first name. Alter the above Store Procedure to supply Default Values if user does not enter any value. ( Use AdventureWorks)

Ans-

GO

CREATE PROCEDURE Person.up\_getNamesByType

@Type nchar(2) = 'EM'

AS

SELECT FirstName FROM Person.Person WHERE PersonType = @Type

GO

**Q.6** Write a trigger for the Product table to ensure the list price can never be raised more than 15 Percent in a single change. Modify the above trigger to execute its check code only if the ListPrice column is updated (Use AdventureWorks Database).

Ans-

GO

CREATE TRIGGER CheckPriceRise

ON Production.Product

AFTER UPDATE

AS

BEGIN

SET NOCOUNT ON;

IF UPDATE(ListPrice)

BEGIN

UPDATE Production.Product

SET ListPrice = IIF((i.ListPrice - d.ListPrice) > d.ListPrice\*0.15, d.ListPrice, i.ListPrice)

FROM deleted AS d, Production.Product AS p

INNER JOIN inserted AS i ON i.ProductID = p.ProductID

END

END

GO