Assignment-2

Exercise-1

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

Query **- Select count(BusinessEntityID) as Number\_of\_Records**

**from Sales.SalesPerson**

Graphical user interface, text, application, Word

Description automatically generated

2.Select both the FirstName and Last Name of records from the Person table where the FirstName begins with the letter ‘B’.

Query - **Select FirstName,LastName**

**from Person.Person**

**where FirstName LIKE 'B%';**

Graphical user interface, application, Word

Description automatically generated

3.Select a list of FirstName and LastName for employees where Title is one of Design Engineer, Tool Designer or Marketing Assistant.

Query – **SELECT Person.Person.FirstName , Person.Person.LastName**

**from Person.Person Inner Join HumanResources.Employee**

**on Person.Person.BusinessEntityID=HumanResources.Employee.BusinessEntityID**

**where HumanResources.Employee.JobTitle='Design Engineer' or**

**HumanResources.Employee.JobTitle='Tool Designer' or**

**HumanResources.Employee.JobTitle='Marketing Assistant';**

Graphical user interface, text, application

Description automatically generated

4.Display the Name and Color of the Product with the maximum weight.

Query – **SELECT Name,Color**

**from Production.Product**

**where Weight= (Select MAX(Weight) from Production.Product);**

Graphical user interface, text, application

Description automatically generated

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.

Query – **SELECT Description,Coalesce(MaxQty,0.00) as MaxQty**

**from Sales.SpecialOffer**

A screenshot of a computer

Description automatically generated

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.'

Query – **SELECT AVG(AverageRate) as Average\_exchange\_rate\_for\_the\_day**

**from [Sales].[CurrencyRate]**

**where FromCurrencyCode = 'USD' and ToCurrencyCode = 'GBP' and year(CurrencyRateDate)=2005;**

Graphical user interface, text, application

Description automatically generated

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.

Query –

**Select Row\_Number() over (order by FirstName) as Sequence\_No,FirstName,LastName**

**from Person.Person**

**where FirstName LIKE '%ss%';**

Graphical user interface, text, application

Description automatically generated

8.Sales people receive various commission rates that belong to 1 of 4 bands .

Display the [SalesPersonID] with an additional column entitled ‘Commission Band’ indicating the appropriate band as above.

Query –

**Select BusinessEntityID as Sales\_Person\_Id,**

**CASE**

**when CommissionPct=0.00 then 'BAND\_0'**

**when CommissionPct>0.00 and CommissionPct<=0.01 then 'BAND\_1'**

**when CommissionPct>0.01 and CommissionPct<=0.015 then 'BAND\_2'**

**when CommissionPct>0.015 then 'Band\_3'**

**END AS 'Commission\_Band'**

**from Sales.SalesPerson**

**order by Commission\_Band**

A screenshot of a computer

Description automatically generated

9. Display the managerial hierarchy from Ruth Ellerbrock (person type – EM) up to CEO Ken Sanchez.

Query –

**Declare @BusinessEntityId int**

**select @BusinessEntityId = BusinessEntityID from Person.Person**

**where FirstName = 'Ruth' and LastName = 'Ellerbrock' and PersonType ='EM'**

**Exec uspGetEmployeeManagers @BusinessEntityId**

A screenshot of a computer

Description automatically generated

10.Display the ProductId of the product with the largest stock level.

Query –

**Select ProductID**

**from Production.Product**

**where SafetyStockLevel=(Select MAX(SafetyStockLevel) from Production.Product);**

A screenshot of a computer

Description automatically generated

Exercise-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.

Query –

**Using Join**

**Select sc.CustomerID**

**from Sales.Customer as sc**

**left join Sales.SalesOrderHeader as sso on sc.CustomerID=sso.CustomerID**

**where sso.SalesOrderID is Null**

Graphical user interface, text, application, email

Description automatically generated

**Using Subquery-**

**Select CustomerID**

**from Sales.Customer**

**WHERE CustomerID not in(Select CustomerID from Sales.SalesOrderHeader)**

Graphical user interface, text, application

Description automatically generated

**Using CTE**

**WITH NoOrderCustomers(CustomerID)**

**AS(**

**SELECT sc.CustomerID**

**FROM Sales.Customer as sc**

**LEFT JOIN Sales.SalesOrderHeader as sso ON sc.CustomerID = sso.CustomerID**

**WHERE sso.SalesOrderID IS NULL**

**)**

**SELECT CustomerID FROM NoOrderCustomers**

A screenshot of a computer

Description automatically generated

**Using EXISTS**

**Select sc.CustomerID**

**from Sales.Customer sc**

**where not exists(select sso.CustomerID from Sales.SalesOrderHeader as sso**

**where sc.CustomerID=sso.CustomerID)**

Graphical user interface, text, application

Description automatically generated

Exercise-3

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

Query **– SELECT TOP 5 SalesOrderID AS 'Order ID',**

**OrderDate AS 'Date Of Order',**

**AccountNumber AS 'Account Number',**

**SUM(TotalDue) AS 'Amount Spent'**

**FROM Sales.SalesOrderHeader**

**GROUP BY AccountNumber,**

**OrderDate,**

**SalesOrderID**

**HAVING SUM(TotalDue) > 70000**

**ORDER BY OrderDate DESC;**

A screenshot of a computer

Description automatically generated

Exercise-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.

Query –

**Create function GetConvertedPrice (@CurrencyCode nchar(3), @Date date, @UnitPrice money)**

**returns money**

**as**

**begin**

**declare @Price money**

**declare @DayRate money**

**Select @DayRate = EndOfDayRate from Sales.CurrencyRate where ToCurrencyCode = @CurrencyCode and ModifiedDate = @Date**

**Set @Price = @UnitPrice \* @DayRate**

**return @Price**

**end**

**--Main Function**

**Create Function GetSalesOrderDetail (@SalesOrderID int , @CurrencyCode nchar(3), @Date datetime)**

**Returns Table**

**as**

**Return (Select OrderQty, ProductID, UnitPrice, dbo.GetConvertedPrice(@CurrencyCode, @Date, UnitPrice)**

**as 'Converted Price'from Sales.SalesOrderDetail**

**where SalesOrderID=@SalesOrderID and ModifiedDate=@Date)**

**--Calling Function**

**Select \* from dbo.GetSalesOrderDetail(43659, 'AUD', '2005-07-01 00:00:00')**

Graphical user interface, text, application

Description automatically generated

Exercise-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.

Query-

**Create Procedure spGetName**

**@Firstname nvarchar(50)**

**as**

**begin**

**Select FirstName +' '+ isNull(MiddleName+' ', ' ')+LastName as FullName from Person.Person where FirstName = @FirstName**

**end**

**Alter procedure spGetName**

**@FirstName nvarchar(50) = Null**

**as**

**begin**

**Select FirstName +' '+ isNull(MiddleName+' ', ' ')+LastName as FullName from Person.Person where FirstName = isNull(@FirstName,'Ken')**

**end**

**--Calling Stored Procedure**

**Exec spGetName**

Graphical user interface, text, application

Description automatically generated

Exercise-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

Query-

**CREATE OR ALTER TRIGGER [Production].UpdateTrigger**

**ON Production.Product**

**INSTEAD OF UPDATE**

**AS**

**SET NOCOUNT ON**

**BEGIN**

**IF UPDATE(ListPrice) -- Modification**

**DECLARE @OldListPrice money**

**DECLARE @InsertedListPrice money**

**DECLARE @ID int**

**SELECT @OldListPrice = p.ListPrice,**

**@InsertedListPrice=inserted.ListPrice,**

**@ID = inserted.ProductID**

**FROM Production.Product p, inserted**

**WHERE p.ProductID = inserted.ProductID;**

**IF( @InsertedListPrice > ( @OldListPrice + (0.15\*@OldListPrice) ) )**

**BEGIN**

**RAISERROR('LIST PRICE MORE THAN 15 PERCENT, TRANSACTION FAILED',16,0)**

**return**

**END**

**ELSE**

**BEGIN**

**Update Production.Product SET ListPrice=@InsertedListPrice**

**WHERE Production.Product.ProductID = @ID;**

**END**

**END;**

**UPDATE PRODUCTION.Product**

**SET ListPrice=10**

**WHERE Product.ProductID=4;**

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application, Word

Description automatically generated