ASSIGNMENT-3 (SQL)

/\* Exercise1

quest:1.1 Display the number of records in the [SalesPerson] table. (Schema(s) involved: Sales)\*/

Select count(BusinessEntityID) AS Total\_Records from Sales.SalesPerson

/\* 1.2 . Select both the FirstName and LastName of records from the Person table where the FirstName begins with the letter ‘B’.

(Schema(s) involved: Person)\*/

Select FirstName , LastName from Person.person where FirstName Like 'B%'

/\* 1.3 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)\*/

Select pers.FirstName , pers.LastName from Person.person as pers inner join HumanResources.Employee Hum\_res ON pers.BusinessEntityID=Hum\_res.BusinessEntityID

where Hum\_res.JobTitle='Design Engineer' OR Hum\_res.JobTitle='Tool Engineer' OR Hum\_res.JobTitle='Marketing Assistence';

/\* 1.4 Display the Name and Color of the Product with the maximum weight. (Schema(s) involved: Production)\*/

select Name , Color from Production.Product

where Weight=(select MAX(Weight) from Production.Product);

/\* 1.5 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)\*/

select Description, ISNULL(MaxQty,0.00) as Max\_quantity from Sales.SpecialOffer;

/\*1.6 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)\*/

Select AVG(AverageRate) as Avg\_exch\_Rate from Sales.CurrencyRate

where FromCurrencyCode='USD' AND ToCurrencyCode='GBP';

/\*1.7 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)\*/

Select ROW\_NUMBER() OVER(Order by FirstName) as Sequence\_number , FirstName , LastName

from Person.Person where FirstName like '%ss%';

/\*1.8 Sales people receive various commission rates that belong to 1 of 4 bands. (Schema(s) involved: Sales)

CommissionPct Commission Band

0.00 Band 0

Up To 1% Band 1

Up To 1.5% Band 2

Greater 1.5% Band 3

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

select BusinessEntityID as SalesPersonID ,

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

/\*1.9 Display the managerial hierarchy from Ruth Ellerbrock (person type – EM) up to CEO Ken Sanchez. Hint: use

[uspGetEmployeeManagers] (Schema(s) involved: [Person], [HumanResources])\*

declare @id int select @id=BusinessEntityID from Person.person where FirstName='Ruth' ANd LastName='Ellerbrock' AND PersonType='EM' Exec uspGetEmployeeManagers @BusinessEntityID=@id

/\* 1.10 Display the ProductId of the product with the largest stock level. Hint: Use the Scalar-valued function [dbo]. [UfnGetStock].

(Schema(s) involved: Production)\*/

select ProductID from Production.Product

where SafetyStockLevel=(select max(SafetyStockLevel) from Production.Product)

/\* Exercise.2. Write separate queries to list all AdventureWorks customers who

have not placed an order.\*/

-- 2.1:: By Using JOIN Statement

SELECT Pers.FirstName + Pers.LastName AS Customer\_Name

FROM Person.Person Pers INNER JOIN

Sales.Customer SC ON

Pers.BusinessEntityID = SC.CustomerID LEFT JOIN

Sales.SalesOrderHeader Sh ON

SC.CustomerID = Sh.CustomerID

WHERE Sh.SalesOrderID IS NULL;

-- 2.2:: By Using SubQuery

SELECT FirstName + LastName AS Customer\_Name

FROM Person.Person

Where BusinessEntityID IN (SELECT CustomerID

FROM Sales.Customer

WHERE CustomerID NOT IN (SELECT CustomerID

FROM Sales.SalesOrderHeader));

/\*

Exercise3. Show the most recent five orders that were purchased from account numbers that have spent more than $70,000 with AdventureWorks. \*/

Select Top 5 SalesOrderID, OrderDate, AccountNumber, Sum(TotalDue) as Spent\_money From Sales.SalesOrderHeader

Group By AccountNumber, OrderDate, SalesOrderID

Having Sum(TotalDue) > 70000

ORDER BY OrderDate DESC;

/\*

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. ( Use AdventureWorks) \*

Create function New2Function(@Sales\_OrderId int, @Currency\_Code nvarchar(3), @Date datetime) Returns table AS Return (Select sd.ProductID, sd.OrderQty , sd.UnitPrice , sd.UnitPrice\*sr.EndOfDayRate AS 'Target Price'

From Sales.SalesOrderDetail As sd,

Sales.CurrencyRate As sr

Where sr.ToCurrencyCode = @Currency\_Code AND

sr.ModifiedDate = @Date AND

sd.SalesOrderID = @Sales\_OrderID)

Go

Select \* from New2Function(43665,'ARS','2005-07-01');

drop Function New2Function

/\*

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.( Use AdventureWorks).

\*/

GO

Create procedure SupplyingNameInfFromPerson

@FirstName nvarchar(20) = 'Sam' As Begin

Select BusinessEntityID ,

CONCAT(FirstName,LastName) As 'NAME', PersonType

From Person.Person

Where FirstName = @FirstName

End

GO

Execute SupplyingNameInfFromPerson

Execute SupplyingNameInfFromPerson @FirstName = 'Zoe'

Drop Procedure SupplyingNameInfFromPerson

/\*

Exercise.6. Write a trigger for the Product table to ensure the list 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).

\*/

GO

CREATE or ALTER TRIGGER [Production].[trgLimitPriceChanges]

ON [Production].[Product]

FOR UPDATE AS IF EXISTS (SELECT \* FROM inserted i JOIN deleted d ON i.ProductID = d.ProductID

WHERE i.ListPrice > (d.ListPrice \* 1.15))

BEGIN

RAISERROR ( 'Price increase may not be greater than 15 percent . Transaction Failed.',16,1)

ROLLBACK TRAN

END

GO

Select \* from Production.product where ProductID=706

update Production.product set ListPrice=2395.0475 where ProductID=706s