**Easy Propositions**

**Problem 1:**

Write a query in TSQL to return list of Marketing manager who have a region.

**Solution:**

Select where contact tittle is marketing manager and region is null

**Diagram:**

A screenshot of a cell phone

Description automatically generated

**Columns in Select Clause:**

|  |  |
| --- | --- |
| **Table** | **Column(s)** |
| [Sales].[Customers] | 1. custid 2. Contactname 3. Contacttitle 4. Region |

**Problem Solving Query:**

use TSQLV4

select custid, contactname, contacttitle, region

from [Sales].[Customers]

where contacttitle = 'marketing manager' and region is not null

**Sample Output with total number of rows returned (4)**

**A screenshot of a cell phone

Description automatically generated**

**Sample JSON Output with total number of rows returned (10)**

use TSQLV4

select custid, contactname, contacttitle, region

from [Sales].[Customers]

where contacttitle = 'marketing manager' and region is not null

for json path, root('RegionList'), include\_null\_values;

**A screenshot of a cell phone

Description automatically generated**

**Problem 2:**

Returning the three ship countries with the highest freight which placed on the last day of the month

**Solution:**

Get all the freight where order is placed last day of the month

**Diagram:**

**A screenshot of a cell phone

Description automatically generated**

**Columns in Select Clause:**

|  |  |
| --- | --- |
| **Table** | **Column(s)** |
| [Sales].[Orders] | 1. shipcountry 2. max(freight) as maxfreight |

**Problem Solving Query:**

use TSQLV4

select top(3) shipcountry, max(freight) maxfreight

from [Sales].[Orders]

where orderdate = EOMONTH(orderdate)

group by shipcountry

order by maxfreight desc

**Sample Output with total number of rows returned (3)**

A screenshot of a cell phone

Description automatically generated

**Sample JSON Output with total number of rows returned (3)**

use TSQLV4

select top(3) shipcountry, max(freight) as maxfreight

from [Sales].[Orders]

where orderdate = EOMONTH(orderdate)

group by shipcountry

order by maxfreight desc

for json path, root('Top 3 countries with highest flight'), include\_null\_values;

A screenshot of a social media post

Description automatically generated

**Problem 3:**

Return customers with orders placed on 2015 along with their orders id

**Solution:**

We have to do a Inner join between customer and order where the order is placed on 2015

**Diagram:**

A screenshot of a computer

Description automatically generated

**Columns in Select Clause:**

|  |  |
| --- | --- |
| **Table** | **Column(s)** |
| Sales.Order | 1. orderid 2. orderdate |
| Sales.Customers | 1. custid 2. companyname |

**Problem Solving Query:**

use TSQLV4

go

SELECT Customers.custid, Customers.companyname, orders.orderid, Orders.orderdate

FROM Sales.Customers AS Customers

INNER JOIN Sales.Orders AS orders

ON orders.custid = Customers.custid

WHERE orders.orderdate >= '20150101' and orders.orderdate < '20160101'

**Sample Output with total number of rows returned(408):**

A screenshot of a computer

Description automatically generated

**Sample JSON Output with total number of rows returned (408)**

use TSQLV4

go

SELECT Customers.custid, Customers.companyname, orders.orderid, Orders.orderdate

FROM Sales.Customers AS Customers

INNER JOIN Sales.Orders AS orders

ON orders.custid = Customers.custid

WHERE orders.orderdate >= '20150101' and orders.orderdate < '20160101'

for json path, root('CustomerList')

A screenshot of a cell phone

Description automatically generated

**Problem 4:**

Write a query that returns all customers in the output, but matchesthem with their respective orders only if they were placed between February 12, 2016 and March 12, 2016

**Solution:**

Creating a left outer join between customer and sales order on their customer id and checking the date between February 12, 2016 and March 12, 2016

**Diagram:**

**A screenshot of a social media post

Description automatically generated**

**Columns in Select Clause:**

|  |  |
| --- | --- |
| **Table** | **Column(s)** |
| [Sales].[Customer] | 1. CustomerId 2. CustomerCompanyName |
| [Sales].[Order] | 1. Orderid 2. Orderdate |

**Problem Solving Query:**

use Northwinds2019TSQLV5

SELECT C.CustomerId, C.CustomerCompanyName, O.orderid, O.orderdate

FROM [Sales].[Customer] AS C

LEFT OUTER JOIN [Sales].[Order] AS O

ON O.CustomerId = C.CustomerId

AND O.orderdate between '20160212' and '20160312'

**Sample Output with total number of rows returned (108)**

**A screenshot of a cell phone

Description automatically generated**

**Sample JSON Output with total number of rows returned (108)**

use Northwinds2019TSQLV5

SELECT C.CustomerId, C.CustomerCompanyName, O.orderid, O.orderdate

FROM [Sales].[Customer] AS C

LEFT OUTER JOIN [Sales].[Order] AS O

ON O.CustomerId = C.CustomerId

AND O.orderdate between '20160212' and '20160312'

for json path, root('CustomersList'), include\_null\_values;

**A screenshot of a social media post

Description automatically generated**

**Problem 5:**

joins the Customers and Orders tables, based on a match between the customer’s customer ID and the order’s customer ID

**Solution:**

Create a left outer join between customer and sales order on customer id and returning the custid, companyname and orderid

**Diagram A screenshot of a cell phone

Description automatically generated**

**Columns in Select Clause:**

|  |  |
| --- | --- |
| **Table** | **Column(s)** |
| [Sales].[Order] | 1. orderid |
| [Sales].[Customer] | 1. custid 2. companyname |

**Problem Solving Query:**

use TSQLV4

SELECT C.custid, C.companyname, O.orderid

FROM Sales.Customers AS C

LEFT OUTER JOIN Sales.Orders AS O

ON C.custid = O.custid

**Sample Output with total number of rows returned (832):**

**A screenshot of text

Description automatically generated**

**Sample JSON Output with total number of rows returned (832)**

use TSQLV4

SELECT C.custid, C.companyname, O.orderid

FROM Sales.Customers AS C

LEFT OUTER JOIN Sales.Orders AS O

ON C.custid = O.custid

for json path, root('match between customers and orderers'), include\_null\_values;

**A screenshot of a cell phone

Description automatically generated**

**Medium Propositions**

**Problem 6:**

Returning a customer who at least placed an order

**Solution:**

We are making a inner join between sales order and customer and order by customer company name and order date

**Diagram:**

**A screenshot of a social media post

Description automatically generated**

**Columns in Select Clause:**

|  |  |
| --- | --- |
| **Table** | **Column(s)** |
| [Sales].[Customer] | 1. CustomerId 2. CustomerCompanyName |
| [Sales].[Order] | 1. OrderId |

**Problem Solving Query:**

use Northwinds2019TSQLV5

go

SELECT C.CustomerId, c.CustomerCompanyName, o.OrderId, CONVERT(VARCHAR(10), CAST(o.OrderDate AS DATE), 101) as Date

FROM [Sales].[Customer] AS C

INNER JOIN [Sales].[Order] AS O

ON C.CustomerId= O.CustomerId

Order By C.CustomerCompanyName, o.OrderDate

**Sample Output with total number of rows returned (830)**

**A screenshot of a computer

Description automatically generated**

**Sample JSON Output with total number of rows returned (830)**

use Northwinds2019TSQLV5

go

SELECT C.CustomerId, c.CustomerCompanyName, o.OrderId, CONVERT(VARCHAR(10), CAST(o.OrderDate AS DATE), 101) as Date

FROM [Sales].[Customer] AS C

INNER JOIN [Sales].[Order] AS O

ON C.CustomerId= O.CustomerId

Order By C.CustomerCompanyName, o.OrderDate

for json path, root('OrderList'), include\_null\_values;

**A screenshot of a cell phone

Description automatically generated**

**Problem 7:**

Write a query joining three tables, making sure it contains top 100 rows from SalesOrderHeader

**Solution:**

Creating a left outer join between salesOrderHeader, currencyRate and ShipMethjod where we are checking the average rate is not null than we are order by the currency rate and sales order id

**Diagram:**

**A screenshot of a social media post

Description automatically generated**

**Columns in Select Clause:**

|  |  |
| --- | --- |
| **Table** | **Column(s)** |
| Sales.SalesOrderHeader | 1. SalesOrderID |
| Sales.CurrencyRate | 1. CurrencyRateID 2. AverageRate |
| Purchasing.ShipMethod | 1. ShipBase |

**Problem Solving Query:**

use AdventureWorks2014

SELECT top (100) C.CurrencyRateID, C.AverageRate, S.ShipBase, SalesOrderID

FROM Sales.SalesOrderHeader AS OH

LEFT OUTER JOIN Sales.CurrencyRate AS C

ON OH.CurrencyRateID = C.CurrencyRateID

LEFT OUTER JOIN Purchasing.ShipMethod AS S

ON OH.ShipMethodID = S.ShipMethodID

where C.AverageRate is not null

order by c.CurrencyRateID, SalesOrderID desc

**Sample Output with total number of rows returned (100):**

**A picture containing screenshot

Description automatically generated**

**Sample JSON Output with total number of rows returned (100)**

use AdventureWorks2014

SELECT top (100) C.CurrencyRateID, C.AverageRate, S.ShipBase, SalesOrderID

FROM Sales.SalesOrderHeader AS OH

LEFT OUTER JOIN Sales.CurrencyRate AS C

ON OH.CurrencyRateID = C.CurrencyRateID

LEFT OUTER JOIN Purchasing.ShipMethod AS S

ON OH.ShipMethodID = S.ShipMethodID

where C.AverageRate is not null

order by c.CurrencyRateID, SalesOrderID desc

for json path, root('top 100 rows'), include\_null\_values;

**A screenshot of a computer

Description automatically generated**

**Problem 8:**

Return all the different product name and the customer for all Customers who ordered ProductModel 'Racing Socks, M'

**Solution:**

To solve this, we have to create a join between ProductModel, Product, customer, salesOrderDetails and salesPrderHeader where name is equal to 'Racing Socks, M'

**Diagram:**

**A screenshot of a cell phone

Description automatically generated**

**Columns in Select Clause:**

|  |  |
| --- | --- |
| **Table** | **Column(s)** |
| [Production].[ProductModel | 1. ProductModelId |
| [Production].[Product] | 1. name |
| [Sales].[SalesOrderDetail] | 1. ProductId |
| [Sales].[SalesOrderHeader] | 1. salesOrderId |
| [Sales].[Customer] | 1. CustomerId |

**Problem Solving Query:**

use AdventureWorks2014

SELECT Distinct P.name, C.CustomerID

FROM

[Production].[ProductModel] as PM

JOIN

[Production].[Product] as P

ON PM.ProductModelID = P.ProductModelID

JOIN

[Sales].[SalesOrderDetail] as OD

ON OD.ProductID = P.ProductID

JOIN

[Sales].[SalesOrderHeader] as OH

ON OD.SalesOrderID = OH.SalesOrderID

JOIN

[Sales].[Customer] as C

ON OH.CustomerID = C.CustomerID

WHERE

P.Name = 'Racing Socks, M' and c.CustomerID % 2 = 0

order by C.CustomerID, P.name

**Sample Output with total number of rows returned (186):**

A screenshot of a cell phone

Description automatically generated

**Sample JSON Output with total number of rows returned (186)**

use AdventureWorks2014

SELECT Distinct P.name, C.CustomerID

FROM

[Production].[ProductModel] as PM

JOIN

[Production].[Product] as P

ON PM.ProductModelID = P.ProductModelID

JOIN

[Sales].[SalesOrderDetail] as OD

ON OD.ProductID = P.ProductID

JOIN

[Sales].[SalesOrderHeader] as OH

ON OD.SalesOrderID = OH.SalesOrderID

JOIN

[Sales].[Customer] as C

ON OH.CustomerID = C.CustomerID

WHERE

P.Name = 'Racing Socks, M' and c.CustomerID % 2 = 0

order by C.CustomerID, P.name

for json path, root('product and customers list'), include\_null\_values;

**A screenshot of a computer

Description automatically generated**

**Problem 9:**

write a query where all the customers who atleast had one order

**Solution:**

We need to make an inner join between customer and sales order on customerId and Group by the query with CustomerCompanyName

**Diagram:**

**A screenshot of a social media post

Description automatically generated**

**Columns in Select Clause:**

|  |  |
| --- | --- |
| **Table** | **Column(s)** |
| [Sales].[Customer] | 1. CustomerCompanyName 2. CustomerId |
| [Sales].[Order] | 1. OrderCount 2. OrderId |

**Problem Solving Query:**

use Northwinds2019TSQLV5

Select C.CustomerCompanyName, count(O.orderId) as OrderCount

FROM [Sales].[Customer] as C

INNER JOIN [Sales].[Order]as O

On C.CustomerId = O.CustomerId

GROUP BY C.CustomerCompanyName

ORDER BY count(O.orderId) asc

**Sample Output with total number of rows returned (89)**

**A screenshot of text

Description automatically generated**

Sample JSON Output with total number of rows returned (89)

use Northwinds2019TSQLV5

Select C.CustomerCompanyName, count(O.orderId) as OrderCount

FROM [Sales].[Customer] as C

INNER JOIN [Sales].[Order]as O

On C.CustomerId = O.CustomerId

GROUP BY C.CustomerCompanyName

ORDER BY count(O.orderId) asc

for json path, root('Customer whon ordered'), include\_null\_values;

**A screenshot of a cell phone

Description automatically generated**

**Problem 10:**

write a query which returns all the companies and their order

**Solution:**

We need to make a left outer join between customer and sales order on customerId and Group by the query with CustomerCompanyName

**Diagram:**

**Columns in Select Clause:**

|  |  |
| --- | --- |
| **Table** | **Column(s)** |
| [Sales].[Customer] | 1. CustomerCompanyName 2. CustomerId |
| [Sales].[Order] | 1. OrderCount 2. OrderId |

**Problem Solving Query:**

use Northwinds2019TSQLV5

Select C.CustomerCompanyName, count(O.orderId) as OrderCount

FROM [Sales].[Customer] as C

left outer join [Sales].[Order]as O

On C.CustomerId = O.CustomerId

GROUP BY C.CustomerCompanyName

ORDER BY count(O.orderId) asc

**Sample Output with total number of rows returned (91)**

A screenshot of a cell phone

Description automatically generated

Sample JSON Output with total number of rows returned (91)

use Northwinds2019TSQLV5

Select C.CustomerCompanyName, count(O.orderId) as OrderCount

FROM [Sales].[Customer] as C

left outer join [Sales].[Order]as O

On C.CustomerId = O.CustomerId

GROUP BY C.CustomerCompanyName

ORDER BY count(O.orderId) asc

for json path, root('List of companies'), include\_null\_values;

Sample JSON Output with total number of rows returned (91)

**A screenshot of a social media post

Description automatically generated**

**Problem 11:**

Return customers and their orders including customers who placed no orders

**Solution:** Creating a inner join between customer, sales order and orderdetails where the country is USA

**Diagram:**

A screenshot of a cell phone

Description automatically generated

**Columns in Select Clause:**

|  |  |
| --- | --- |
| **Table** | **Column(s)** |
| Sales.Customers | 1. Custid 2. country |
| Sales.Orders | 1. orderid 2. numorders |
| Sales.OrderDetails | 1. Orderid |

**Problem Solving Query:**

use TSQLV4

SELECT C.custid, COUNT( DISTINCT O.orderid) AS numorders, SUM(OD.qty) AS totalqty

FROM Sales.Customers AS C

INNER JOIN Sales.Orders AS O

ON O.custid = C.custid

INNER JOIN Sales.OrderDetails AS OD

ON OD.orderid = O.orderid

WHERE C.country = N'USA'

GROUP BY C.custid

**Sample Output with total number of rows returned (13)**

**A screenshot of a cell phone

Description automatically generated**

**Sample JSON Output with total number of rows returned (13)**

use TSQLV4

SELECT C.custid, COUNT( DISTINCT O.orderid) AS numorders, SUM(OD.qty) AS totalqty

FROM Sales.Customers AS C

INNER JOIN Sales.Orders AS O

ON O.custid = C.custid

INNER JOIN Sales.OrderDetails AS OD

ON OD.orderid = O.orderid

WHERE C.country = N'USA'

GROUP BY C.custid

for json path, root('USA Customers'), include\_null\_values;

A screenshot of a cell phone

Description automatically generated

**Problem 12:** Write a query that is similar to the above query but where country is from japan

**Solution:** Creating a inner join between customer, sales order and orderdetails where the country is japan

**Diagram:**

A screenshot of a cell phone

Description automatically generated

**Columns in Select Clause:**

|  |  |
| --- | --- |
| **Table** | **Column(s)** |
| Sales.Customers | 1. Custid 2. country |
| Sales.Orders | 1. orderid 2. numorders |
| Sales.OrderDetails | 1. Orderid |

**Problem Solving Query:**

use TSQLV4

SELECT C.custid, COUNT( DISTINCT O.orderid) AS numorders, SUM(OD.qty) AS totalqty

FROM Sales.Customers AS C

INNER JOIN Sales.Orders AS O

ON O.custid = C.custid

INNER JOIN Sales.OrderDetails AS OD

ON OD.orderid = O.orderid

WHERE C.country = N'Japan'

GROUP BY C.custid

**Sample Output with total number of rows returned (0)**

**A picture containing screenshot

Description automatically generated**

Sample JSON Output with total number of rows returned (0)

use TSQLV4

SELECT C.custid, COUNT( DISTINCT O.orderid) AS numorders, SUM(OD.qty) AS totalqty

FROM Sales.Customers AS C

INNER JOIN Sales.Orders AS O

ON O.custid = C.custid

INNER JOIN Sales.OrderDetails AS OD

ON OD.orderid = O.orderid

WHERE C.country = N'Japan'

GROUP BY C.custid

for json path, root('JPN Customers'), include\_null\_values;

**Note:** There is no output for the japan. So, we don’ get any JSON output for this Query

**Problem 13:**

Return customers and their all orders

**Solution:** Creating a inner join between customer, sales order and orderdetails and group by custid

**Diagram:**

**A screenshot of a cell phone

Description automatically generated**

**Columns in Select Clause:**

|  |  |
| --- | --- |
| **Table** | **Column(s)** |
| Sales.Customers | 1. Custid 2. country |
| Sales.Orders | 1. orderid 2. numorders |
| Sales.OrderDetails | 1. Orderid |

**Problem Solving Query:**

use TSQLV4

SELECT C.custid, COUNT( DISTINCT O.orderid) AS numorders, SUM(OD.qty) AS totalqty

FROM Sales.Customers AS C

INNER JOIN Sales.Orders AS O

ON O.custid = C.custid

INNER JOIN Sales.OrderDetails AS OD

ON OD.orderid = O.orderid

GROUP BY C.custid

**Sample Output with total number of rows returned (89)**

**A screenshot of a cell phone

Description automatically generated**

**Sample JSON Output with total number of rows returned (89)**

use TSQLV4

SELECT C.custid, COUNT( DISTINCT O.orderid) AS numorders, SUM(OD.qty) AS totalqty

FROM Sales.Customers AS C

INNER JOIN Sales.Orders AS O

ON O.custid = C.custid

INNER JOIN Sales.OrderDetails AS OD

ON OD.orderid = O.orderid

GROUP BY C.custid

for json path, root('Customer and order list'), include\_null\_values;

**A screenshot of a cell phone

Description automatically generated**

**Complex Propositions**

**Problem 14:**Creating a query that’s going to return customer id in an ascending order

**Solution:** We need to create an inner join between GetCustOrders, sales order and orderdetails. After we are creating a left outer join between OrderDetailsAudit than order by custid

**Diagram:**

**A screenshot of a social media post

Description automatically generated**

**Columns in Select Clause:**

|  |  |
| --- | --- |
| **Table** | **Column(s)** |
| dbo.GetCustOrders | 1. **custid** |
| Sales.Orders | 1. **orderid** |
| Sales.OrderDetails | 1. **numorder** 2. **totalqty** |
| [Sales].[OrderDetailsAudit] | 1. **orderid** |

**Problem Solving Query:**

use TSQLV4

drop function if exists dbo.GetCustOrders;

go

create function dbo.GetCustOrders

(@cid AS INT) RETURNS TABLE

AS

RETURN

SELECT orderid, custid, empid, orderdate, requireddate, shipregion, shippostalcode, shipcountry

FROM Sales.Orders

WHERE custid = @cid

GO

use TSQLV4

SELECT C.custid, COUNT( DISTINCT ODA.orderid) AS numorders, SUM(OD.qty) AS totalqty

FROM dbo.GetCustOrders(5) AS C

INNER JOIN Sales.Orders AS O

ON O.custid = C.custid

INNER JOIN Sales.OrderDetails AS OD

ON OD.orderid = O.orderid

LEFT OUTER JOIN [Sales].[OrderDetailsAudit] AS ODA

ON O.orderid = ODA.orderid

GROUP BY C.custid;

**Sample Output with total number of rows returned (1)**

A screenshot of a cell phone

Description automatically generated

Sample JSON Output with total number of rows returned (1)

use TSQLV4

drop function if exists dbo.GetCustOrders;

go

create function dbo.GetCustOrders

(@cid AS INT) RETURNS TABLE

AS

RETURN

SELECT orderid, custid, empid, orderdate, requireddate, shipregion, shippostalcode, shipcountry

FROM Sales.Orders

WHERE custid = @cid

GO

use TSQLV4

SELECT C.custid, COUNT( DISTINCT ODA.orderid) AS numorders, SUM(OD.qty) AS totalqty

FROM dbo.GetCustOrders(5) AS C

INNER JOIN Sales.Orders AS O

ON O.custid = C.custid

INNER JOIN Sales.OrderDetails AS OD

ON OD.orderid = O.orderid

LEFT OUTER JOIN [Sales].[OrderDetailsAudit] AS ODA

ON O.orderid = ODA.orderid

GROUP BY C.custid

for json path, root('list of custid'), include\_null\_values;

A screenshot of a cell phone

Description automatically generated

**Problem 15:**

Create a function that returns the number of college supplies

each student orders per semester

**Solution:**

We are creating an inner join between factInternetSales and DimCustomer where lastname has twice e. After we are using GetSeasonByOrder function to get date.

**Diagram:**

**A screenshot of a social media post

Description automatically generated**

**Columns in Select Clause:**

|  |  |
| --- | --- |
| **Table** | **Column(s)** |
| [dbo].[DimCustomer] | 1. FirstName 2. LastName |
| [dbo].[FactInternetSales] | 1. CustomerKey |
| GetSeasonByOrder | 1. OrderDate |

**Problem Solving Query:**

USE AdventureWorksDW2016;

GO

IF OBJECT\_ID(N'GetSeasonByOrder', N'SO') IS NOT NULL

DROP FUNCTION GetSeasonByOrder;

GO

Alter FUNCTION dbo.GetSeasonByOrder(

@OrderDate DATETIME

)

RETURNS VARCHAR(20)

AS

BEGIN

DECLARE @Answer VARCHAR(20)

IF(MONTH(@OrderDate) >= 1 AND MONTH(@OrderDate) <6)

BEGIN

SET @Answer = 'Spring'

END

ELSE IF(MONTH(@OrderDate) >= 6 AND MONTH(@OrderDate) < 9)

BEGIN

SET @Answer = 'Summer'

END

ELSE IF(MONTH(@OrderDate) >= 9 AND MONTH(@OrderDate) <=12)

BEGIN

SET @Answer = 'Fall'

END

ELSE IF(MONTH(@OrderDate) = 12 OR MONTH(@OrderDate) = 1)

BEGIN

SET @Answer = 'Winter'

END

RETURN @Answer

END

GO

USE AdventureWorksDW2016;

SELECT

S.CustomerKey, C.FirstName, C.LastName, COUNT(S.CustomerKey) AS NumberOfSales, [dbo].[GetSeasonByOrder](S.OrderDate) AS OrderSeason

FROM

[dbo].[FactInternetSales] AS S

INNER JOIN

[dbo].[DimCustomer] AS C

ON S.CustomerKey = C.CustomerKey

WHERE c.LastName LIKE '%e%e%'

GROUP BY

S.CustomerKey, C.FirstName, C.LastName, [dbo].[GetSeasonByOrder](S.OrderDate)

ORDER BY

S.CustomerKey ASC

**Sample Output with total number of rows returned (1847):**

**A screenshot of a cell phone

Description automatically generated**

**Sample JSON Output with total number of rows returned (1847)**

USE AdventureWorksDW2016;

GO

IF OBJECT\_ID(N'GetSeasonByOrder', N'SO') IS NOT NULL

DROP FUNCTION GetSeasonByOrder;

GO

Alter FUNCTION dbo.GetSeasonByOrder(

@OrderDate DATETIME

)

RETURNS VARCHAR(20)

AS

BEGIN

DECLARE @Answer VARCHAR(20)

IF(MONTH(@OrderDate) >= 1 AND MONTH(@OrderDate) <6)

BEGIN

SET @Answer = 'Spring'

END

ELSE IF(MONTH(@OrderDate) >= 6 AND MONTH(@OrderDate) < 9)

BEGIN

SET @Answer = 'Summer'

END

ELSE IF(MONTH(@OrderDate) >= 9 AND MONTH(@OrderDate) <=12)

BEGIN

SET @Answer = 'Fall'

END

ELSE IF(MONTH(@OrderDate) = 12 OR MONTH(@OrderDate) = 1)

BEGIN

SET @Answer = 'Winter'

END

RETURN @Answer

END

GO

USE AdventureWorksDW2016;

SELECT

S.CustomerKey, C.FirstName, C.LastName, COUNT(S.CustomerKey) AS NumberOfSales, [dbo].[GetSeasonByOrder](S.OrderDate) AS OrderSeason

FROM

[dbo].[FactInternetSales] AS S

INNER JOIN

[dbo].[DimCustomer] AS C

ON S.CustomerKey = C.CustomerKey

WHERE c.LastName LIKE '%e%e%'

GROUP BY

S.CustomerKey, C.FirstName, C.LastName, [dbo].[GetSeasonByOrder](S.OrderDate)

ORDER BY

S.CustomerKey ASC

for json path, root('list of orders'), include\_null\_values;

A screenshot of a computer

Description automatically generated

**Problem 16:** create a query that returning employees fullname who didn't place an order

**Solution:** Creating full outer join between HumanResources , sales order and sales customer where customer name is null

**Diagram:**

**A screenshot of a cell phone

Description automatically generated**

**Columns in Select Clause:**

|  |  |
| --- | --- |
| **Table** | **Column(s)** |
| [HumanResources].[Employee] | 1. EmployeeId 2. FirstName 3. LastName |
| [Sales].[Order] | 1. orderid 2. employeeid |
| [sales].[customer] | 1. CustomerCompanyName 2. CustomerContactName 3. CustomerContactTitle 4. CustomerCity 5. CustomerCountry |

**Problem Solving Query:**

Scalar Function:

use Northwinds2019TSQLV5

IF OBJECT\_ID(N'getFullName', N'FN') IS NOT NULL

DROP FUNCTION getFullName;

GO

Create function getFullName

(

@FirstName nvarchar(100),

@LastName nvarchar(100)

)

RETURNS VARCHAR(100)

AS

BEGIN

DECLARE @FullName VARCHAR(200)

set @fullname = concat(@FirstName,', ', @LastName )

return @fullname

END;

use Northwinds2019TSQLV5

go

select E.EmployeeId, [dbo].[getFullName](E. EmployeeFirstName, E.EmployeeLastName) as [Full Name],

O.OrderId, C.CustomerCompanyName, C.CustomerContactName, c.CustomerContactTitle,

concat(C.CustomerCity, ', ', C.CustomerCountry) as [Location]

from [HumanResources].[Employee] as E

full outer join [Sales].[Order] as O

on E.EmployeeID = O.EmployeeId

full outer join [sales].[customer] as C

on O.customerId = C.customerId

where O.OrderId is not null

group by E.EmployeeId, [dbo].getFullName(E. EmployeeFirstName, E.EmployeeLastName),

O.orderid, C.CustomerCompanyName, C.CustomerContactName,C.CustomerContactTitle,

concat(C.CustomerCity, ', ', C.CustomerCountry)

order by E.EmployeeId asc

**Sample Output with total number of rows returned (08)**

A screenshot of a cell phone

Description automatically generated

**Sample JSON Output with total number of rows returned (08)**

use Northwinds2019TSQLV5

IF OBJECT\_ID(N'getFullName', N'FN') IS NOT NULL

DROP FUNCTION getFullName;

GO

Create function getFullName

(

@FirstName nvarchar(100),

@LastName nvarchar(100)

)

RETURNS VARCHAR(100)

AS

BEGIN

DECLARE @FullName VARCHAR(200)

set @fullname = concat(@FirstName,', ', @LastName )

return @fullname

END;

use Northwinds2019TSQLV5

go

select E.EmployeeId, [dbo].[getFullName](E. EmployeeFirstName, E.EmployeeLastName) as [Full Name],

O.OrderId, C.CustomerCompanyName, C.CustomerContactName, c.CustomerContactTitle,

concat(C.CustomerCity, ', ', C.CustomerCountry) as [Location]

from [HumanResources].[Employee] as E

full outer join [Sales].[Order] as O

on E.EmployeeID = O.EmployeeId

full outer join [sales].[customer] as C

on O.customerId = C.customerId

where O.OrderId is not null

group by E.EmployeeId, [dbo].getFullName(E. EmployeeFirstName, E.EmployeeLastName),

O.orderid, C.CustomerCompanyName, C.CustomerContactName,C.CustomerContactTitle,

concat(C.CustomerCity, ', ', C.CustomerCountry)

order by E.EmployeeId asc

for json path,root('CustomerWithNoOrder')

A screenshot of a social media post

Description automatically generated

**Problem 17:**

Create a query that return the sum of All Sales per Employee in their history of sales. Then assign a category to each employee based on their Sales they’ve made

**Solution:**

Create an inner join between fact FactResellerSale and DinEmployee on EmployeeKey.

**Diagram:**

**A screenshot of a cell phone

Description automatically generated**

**Columns in Select Clause:**

|  |  |
| --- | --- |
| **Table** | **Column(s)** |
| [dbo].[EmployeeCatagory] | 1. EmployeeKey |
| [dbo].[FactResellerSales] | 1. EmployeeKey 2. SalesAmount |
| [dbo].[DimEmployee] | 1. FirstName 2. LastName |

**Problem Solving Query:**

USE AdventureWorksDW2016;

GO

IF OBJECT\_ID(N'EmployeeCatagory','C') IS NOT NULL

DROP FUNCTION EmployeeCatagory;

GO

Alter function [dbo].[EmployeeCatagory](

@TotalSales INT

)

RETURNS VARCHAR(20)

AS

BEGIN

DECLARE @Answer VARCHAR(20)

IF(@TotalSales > 10000000)

BEGIN

SET @Answer = 'Exceptional'

END

ELSE IF(@TotalSales > 8000000)

BEGIN

SET @Answer = 'Exceeds Expectations'

END

ELSE IF(@TotalSales > 5000000)

BEGIN

SET @Answer = 'Fully Meets Exceptions'

END

ELSE IF(@TotalSales > 30000000)

BEGIN

SET @Answer = 'Needs Development'

END

ELSE

BEGIN

SET @Answer = 'Unsatisfactory'

END

RETURN @Answer

END;

USE AdventureWorksDW2016

go

SELECT

FRS.EmployeeKey, E.FirstName, E.LastName, SUM(FRS.SalesAmount) AS TotalSales,

[dbo].[EmployeeCatagory](SUM(FRS.SalesAmount)) AS EmployeeCategory

FROM

[dbo].[FactResellerSales] AS FRS

INNER JOIN

[dbo].[DimEmployee] AS E

ON FRS.EmployeeKey = E.EmployeeKey

GROUP BY

FRS.EmployeeKey, E.FirstName, E.LastName

ORDER BY

TotalSales DESC

**Sample Output with total number of rows returned (798)**

**A screenshot of a computer

Description automatically generated**

**Sample JSON Output with total number of rows returned (17)**

USE AdventureWorksDW2016;

GO

IF OBJECT\_ID(N'EmployeeCatagory','C') IS NOT NULL

DROP FUNCTION EmployeeCatagory;

GO

Alter function [dbo].[EmployeeCatagory](

@TotalSales INT

)

RETURNS VARCHAR(20)

AS

BEGIN

DECLARE @Answer VARCHAR(20)

IF(@TotalSales > 10000000)

BEGIN

SET @Answer = 'Exceptional'

END

ELSE IF(@TotalSales > 8000000)

BEGIN

SET @Answer = 'Exceeds Expectations'

END

ELSE IF(@TotalSales > 5000000)

BEGIN

SET @Answer = 'Fully Meets Exceptions'

END

ELSE IF(@TotalSales > 30000000)

BEGIN

SET @Answer = 'Needs Development'

END

ELSE

BEGIN

SET @Answer = 'Unsatisfactory'

END

RETURN @Answer

END;

USE AdventureWorksDW2016

go

SELECT

FRS.EmployeeKey, E.FirstName, E.LastName, SUM(FRS.SalesAmount) AS TotalSales,

[dbo].[EmployeeCatagory](SUM(FRS.SalesAmount)) AS EmployeeCategory

FROM

[dbo].[FactResellerSales] AS FRS

INNER JOIN

[dbo].[DimEmployee] AS E

ON FRS.EmployeeKey = E.EmployeeKey

GROUP BY

FRS.EmployeeKey, E.FirstName, E.LastName

ORDER BY

TotalSales DESC

for json path, root('list of Employee Category'), include\_null\_values;

**A screenshot of a cell phone

Description automatically generated**

**Problem 18:**   
 Creat a quary that return all the employees from UK and customers from USA with their full address

**Solution:** Create an inner join between Employee, order and customer

**Diagram:**

**A screenshot of a cell phone

Description automatically generated**

**Columns in Select Clause:**

|  |  |
| --- | --- |
| **Table** | **Column(s)** |
| [Sales].[Customer] | 1. CustomerId 2. CustomerCompanyName 3. CustomerAddress 4. CustomerCity 5. CustomerRegion 6. CustomerPostalCode 7. CustomerCountry |
| [Sales].order | 1. Orderid 2. Orderdate |
| [HumanResources].[Employee] | 1. FirstName 2. LastName 3. EmployeeAddress 4. EmployeeCity 5. EmployeeRegion 6. EmployeePostalCode 7. EmployeeCountry |

**Problem Solving Query:**

use Northwinds2019TSQLV5;

go

create function getLocation(

@add varchar(100),

@city varchar(100),

@region varchar(100),

@zipCode varchar(100),

@Country varchar(100))

returns nvarchar(250)

as

begin

declare @result nvarchar(250)

set @result = concat(@add,', ', @city,', ',@region,', ', @zipcode,', ',@country)

return @result

end;

use Northwinds2019TSQLV5;

go

select concat(E.EmployeeFirstName,' ', E.EmployeeLastName) as Fullname,

[dbo].[getLocation](E.EmployeeAddress,E.EmployeeCity,E.EmployeeRegion,E.EmployeePostalCode,E.EmployeeCountry)

as [Employee Address],

O.OrderId, O.OrderDate,

C.CustomerCompanyName,

[dbo].getLocation(C.CustomerAddress,C.CustomerCity,C.CustomerRegion, C.CustomerPostalCode,C.CustomerCountry)

as [Customer Address]

from [HumanResources].[Employee] as E

inner join [sales].[order] as O

on E.EmployeeId = O.EmployeeId

inner join [sales].[Customer] as C

on O.CustomerId = C.CustomerId

where E.EmployeeCountry <> N'UK' and C.CustomerCountry like N'USA'

group by e.EmployeeFirstName, e.EmployeeLastName,

[dbo].getLocation(E.EmployeeAddress,E.EmployeeCity,E.EmployeeRegion,E.EmployeePostalCode,E.EmployeeCountry),

O.OrderId, O.OrderDate, C.CustomerCompanyName,

[dbo].getLocation(C.CustomerAddress,C.CustomerCity,C.CustomerRegion, C.CustomerPostalCode,C.CustomerCountry)

order by fullname

**Sample Output with total number of rows returned (92)**

**A picture containing window, standing, large, holding

Description automatically generated**

**Sample JSON Output with total number of rows returned (92)**

use Northwinds2019TSQLV5;

go

create function getLocation(

@add varchar(100),

@city varchar(100),

@region varchar(100),

@zipCode varchar(100),

@Country varchar(100))

returns nvarchar(250)

as

begin

declare @result nvarchar(250)

set @result = concat(@add,', ', @city,', ',@region,', ', @zipcode,', ',@country)

return @result

end;

use Northwinds2019TSQLV5;

go

select concat(E.EmployeeFirstName,' ', E.EmployeeLastName) as Fullname,

[dbo].[getLocation](E.EmployeeAddress,E.EmployeeCity,E.EmployeeRegion,E.EmployeePostalCode,E.EmployeeCountry)

as [Employee Address],

O.OrderId, O.OrderDate,

C.CustomerCompanyName,

[dbo].getLocation(C.CustomerAddress,C.CustomerCity,C.CustomerRegion, C.CustomerPostalCode,C.CustomerCountry)

as [Customer Address]

from [HumanResources].[Employee] as E

inner join [sales].[order] as O

on E.EmployeeId = O.EmployeeId

inner join [sales].[Customer] as C

on O.CustomerId = C.CustomerId

where E.EmployeeCountry <> N'UK' and C.CustomerCountry like N'USA'

group by e.EmployeeFirstName, e.EmployeeLastName,

[dbo].getLocation(E.EmployeeAddress,E.EmployeeCity,E.EmployeeRegion,E.EmployeePostalCode,E.EmployeeCountry),

O.OrderId, O.OrderDate, C.CustomerCompanyName,

[dbo].getLocation(C.CustomerAddress,C.CustomerCity,C.CustomerRegion, C.CustomerPostalCode,C.CustomerCountry)

order by fullname

for json path, root('list of Employees and Customers'), include\_null\_values;

**A screenshot of a social media post

Description automatically generated**

**Problem 19:**Creating a query that’s going to return customer id in an ascending order

**Solution:** We need to create an inner join between GetCustOrders, sales order and orderdetails. After we are creating a left outer join between OrderDetailsAudit than order by custid

**Diagram:**

**A screenshot of a social media post

Description automatically generated**

**Columns in Select Clause:**

|  |  |
| --- | --- |
| **Table** | **Column(s)** |
| dbo.GetCustOrders | 1. **custid** |
| Sales.Orders | 1. **orderid** |
| Sales.OrderDetails | 1. **numorder** 2. **totalqty** |
| [Sales].[OrderDetailsAudit] | 1. **orderid** |

**Problem Solving Query:**

use TSQLV4

drop function if exists dbo.GetCustOrders;

go

create function dbo.GetCustOrders

(@cid AS INT) RETURNS TABLE

AS

RETURN

SELECT orderid, custid, empid, orderdate, requireddate, shipregion, shippostalcode, shipcountry

FROM Sales.Orders

WHERE custid = @cid

GO

use TSQLV4

SELECT C.custid, COUNT( DISTINCT ODA.orderid) AS numorders, SUM(OD.qty) AS totalqty

FROM dbo.GetCustOrders(3) AS C

INNER JOIN Sales.Orders AS O

ON O.custid = C.custid

INNER JOIN Sales.OrderDetails AS OD

ON OD.orderid = O.orderid

LEFT OUTER JOIN [Sales].[OrderDetailsAudit] AS ODA

ON O.orderid = ODA.orderid

GROUP BY C.custid;

**Sample Output with total number of rows returned (1)**

A picture containing screenshot

Description automatically generated

**Sample JSON Output with total number of rows returned (1)**

use TSQLV4

drop function if exists dbo.GetCustOrders;

go

create function dbo.GetCustOrders

(@cid AS INT) RETURNS TABLE

AS

RETURN

SELECT orderid, custid, empid, orderdate, requireddate, shipregion, shippostalcode, shipcountry

FROM Sales.Orders

WHERE custid = @cid

GO

use TSQLV4

SELECT C.custid, COUNT( DISTINCT ODA.orderid) AS numorders, SUM(OD.qty) AS totalqty

FROM dbo.GetCustOrders(3) AS C

INNER JOIN Sales.Orders AS O

ON O.custid = C.custid

INNER JOIN Sales.OrderDetails AS OD

ON OD.orderid = O.orderid

LEFT OUTER JOIN [Sales].[OrderDetailsAudit] AS ODA

ON O.orderid = ODA.orderid

GROUP BY C.custid

for json path, root('list of custid'), include\_null\_values;

A screenshot of a cell phone

Description automatically generated

**Problem 20:**

Creat a query that returns average unit price, total unit order and check supply and demond

**Solution:**

We are creating an inner join between product, suppliers and orderdetais table and group by productname

**Diagram:**

**A screenshot of a social media post

Description automatically generated**

**Columns in Select Clause:**

|  |  |
| --- | --- |
| **Table** | **Column(s)** |
| [Production].[Products] | 1. ProductName |
| [Production].[Suppliers] | 1. SupplierId |
| [Sales].[OrderDetails] | 1. ProductId 2. unitPrice 3. qty |

**Problem Solving Query:**use TSQLV4;

IF OBJECT\_ID(N'QuantityCheck', N'QC') IS NOT NULL

DROP FUNCTION QuantityCheck;

GO

Alter function QuantityCheck(

@quantity INT

)

RETURNS VARCHAR(100)

AS

BEGIN

DECLARE @Answer VARCHAR(100)

IF(@quantity > 600)

BEGIN

SET @Answer = 'Surplus'

END

ELSE IF(@quantity >=400) and (@quantity <600)

BEGIN

SET @Answer = ' Above Equlibrium'

END

ELSE IF(@quantity >= 200) and (@quantity <400)

BEGIN

SET @Answer = 'Equlibrium'

END

ELSE IF(@quantity >=100) and (@quantity <200)

BEGIN

SET @Answer = 'Below Equlibrium'

END

ELSE

BEGIN

SET @Answer = 'Shortage'

END

RETURN @Answer

END

GO

use TSQLV4;

select p.ProductName, avg(od.unitprice) AvgUnitPrice, sum(od.qty) TotalUnitsOrdered, [dbo].[QuantityCheck](sum(od.qty)) as SupplyAndDemond

from [Production].[Products] as P

inner join

[Production].[Suppliers] as S

on P.SupplierId = s.SupplierId

inner join

[Sales].[OrderDetails] as OD

on P.ProductId = od.ProductId

group by

P.ProductName

ORDER BY

P.productname ASC

for json path, root('Check supply and demonds'), include\_null\_values;

**Sample Output with total number of rows returned (08)**

A screenshot of text

Description automatically generated

**Sample JSON Output with total number of rows returned (08)**

use TSQLV4;

IF OBJECT\_ID(N'QuantityCheck', N'QC') IS NOT NULL

DROP FUNCTION QuantityCheck;

GO

Alter function QuantityCheck(

@quantity INT

)

RETURNS VARCHAR(100)

AS

BEGIN

DECLARE @Answer VARCHAR(100)

IF(@quantity > 600)

BEGIN

SET @Answer = 'Surplus'

END

ELSE IF(@quantity >=400) and (@quantity <600)

BEGIN

SET @Answer = ' Above Equlibrium'

END

ELSE IF(@quantity >= 200) and (@quantity <400)

BEGIN

SET @Answer = 'Equlibrium'

END

ELSE IF(@quantity >=100) and (@quantity <200)

BEGIN

SET @Answer = 'Below Equlibrium'

END

ELSE

BEGIN

SET @Answer = 'Shortage'

END

RETURN @Answer

END

GO

use TSQLV4;

select p.ProductName, avg(od.unitprice) AvgUnitPrice, sum(od.qty) TotalUnitsOrdered, [dbo].[QuantityCheck](sum(od.qty)) as SupplyAndDemond

from [Production].[Products] as P

inner join

[Production].[Suppliers] as S

on P.SupplierId = s.SupplierId

inner join

[Sales].[OrderDetails] as OD

on P.ProductId = od.ProductId

group by

P.ProductName

ORDER BY

P.productname ASC

for json path, root('Check supply and demonds'), include\_null\_values;

A screenshot of a computer

Description automatically generated