Contents:-

[SQL Practice 1: Select all columns and all rows of the table. 2](#_Toc534716689)

[SQL Practice 2: Select only needed columns from table. 3](#_Toc534716690)

[SQL Practice 3: Select using a numeric criterion. 3](#_Toc534716691)

[SQL Practice 4: Select using a string criterion. 4](#_Toc534716692)

[SQL Practice 5: Select using AND and OR. 4](#_Toc534716693)

[SQL Practice 6: Sort data using ascending and descending. 5](#_Toc534716694)

[SQL Practice 7: Provide user friendly ALIAS for column names. 5](#_Toc534716695)

[SQL Practice 8: Display unique records from a table. 6](#_Toc534716696)

[SQL Practice 9: Searching using pattern and wildcards. 7](#_Toc534716697)

[SQL Practice 10: Create runtime calculated columns. 7](#_Toc534716698)

[SQL Practice 11: CASE statements with SQL. 8](#_Toc534716699)

[SQL Practice 12: Join data from two SELECTS using UNION and UNION ALL. 8](#_Toc534716700)

[SQL Practice 13: Show matching data from two tables. (Inner Join) 9](#_Toc534716701)

[SQL Practice 14: Show all records from one table and only matching record from other table. (Left and Right Join) 9](#_Toc534716702)

[SQL Practice 15: Show all records from matching or unmatching. (Full Outer Join) 10](#_Toc534716703)

[SQL Practice 16: Show Cartesian of two tables. (Cross join) 10](#_Toc534716704)

[SQL Practice 17: Writing a complex SQL inner join statements. 11](#_Toc534716705)

[SQL Practice 18: Display aggregate values from a table. (Group BY) 12](#_Toc534716706)

[SQL Practice 19: Filter on Aggregate values. (HAVING CLAUSE) 12](#_Toc534716707)

[SQL Practice 20: Self Join. 12](#_Toc534716708)

[SQL Practice 21: ISNULL. 13](#_Toc534716709)

[SQL Practice 22: Sub Queries. 14](#_Toc534716710)

[SQL Practice 23: Co-related Queries. 14](#_Toc534716711)

[SQL Practice 24: Find Max, Min and Average. 14](#_Toc534716712)

[SQL Practice 25: Find the between numeric values. 15](#_Toc534716713)

[SQL Practice 26: Dump table data in to new table. (SELECT INTO) 15](#_Toc534716714)

[SQL Practice 27: Insert data in to table. 16](#_Toc534716715)

[SQL Practice 28: Insert bulk data in existing table. 17](#_Toc534716716)

[SQL Practice 29: Update data in to table. 18](#_Toc534716717)

[SQL Practice 30: Delete data from a table. 19](#_Toc534716718)

[Data Definition Language 20](#_Toc534716719)

[SQL Practice 31: Create Database Query. 20](#_Toc534716720)

[SQL Practice 32: Create Database with FileName, Size, MaxSize and Growth. 21](#_Toc534716721)

[SQL Practice 33: Backup Database. 23](#_Toc534716722)

[SQL Practice 34: Restore Database. 23](#_Toc534716723)

[SQL Practice 35: Create table. 28](#_Toc534716724)

[SQL Practice 36: Alter a table structure drop a existing column. 28](#_Toc534716725)

[SQL Practice 37: Alter a table structure change an existing column. 29](#_Toc534716726)

[SQL Practice 38: Adding validation constraints NOT NULL and UNIQUE. 30](#_Toc534716727)

[SQL Practice 39: Check Constraints for data value validation. 31](#_Toc534716728)

[SQL Practice 40: Provide Default values for a table. 32](#_Toc534716729)

[SQL Practice 41: Creating Primary and Foreign key constraints. 33](#_Toc534716730)

[SQL Practice 42: Get Customer who purchases more than one product. 34](#_Toc534716731)

[SQL Practice 43: Get Customer and how much product they have purchased. 34](#_Toc534716732)

[SQL Practice 44: Get Customer who has not purchased any product. 35](#_Toc534716733)

[SQL Practice 45: Get month wise order between the financial year. 35](#_Toc534716734)

[SQL Practice 46: Get year wise order between the financial year. 36](#_Toc534716735)

[SQL Practice 47: Get month wise order between the ‘2018-01-01’ to ‘2018-12-31’ but Row-wise Month. 36](#_Toc534716736)

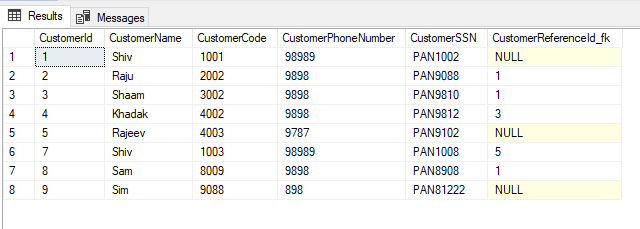
[SQL Practice 48: Get CustomerName, ProductName, ProductCost, Quantity from InvoiceItems whose ProductCost must be greater than 1000 and Quantity between 10 to 50. 37](#_Toc534716737)

[SQL Practice 49: Get Names of customer who have purchase pants and T-shirts both. 38](#_Toc534716738)

SQL Practice 50:

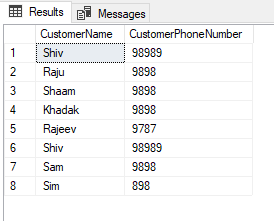
## SQL Practice 1: Select all columns and all rows of the table.

Select \* from tblCustomer



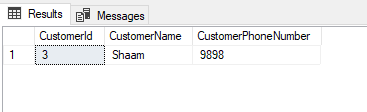
## SQL Practice 2: Select only needed columns from table.

Select CustomerName,CustomerPhoneNumber from tblCustomer



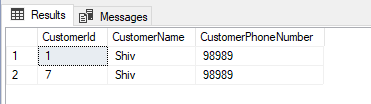
## SQL Practice 3: Select using a numeric criterion.

Select CustomerId,CustomerName,CustomerPhoneNumber from tblCustomer where CustomerId =3



## SQL Practice 4: Select using a string criterion.

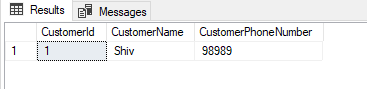
Select CustomerId,CustomerName,CustomerPhoneNumber from tblCustomer where CustomerName ='Shiv'



## SQL Practice 5: Select using AND and OR.

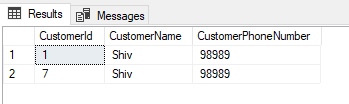
Using AND:

Select CustomerId,CustomerName,CustomerPhoneNumber from tblCustomer where CustomerName ='Shiv' and CustomerId=1



Using OR:

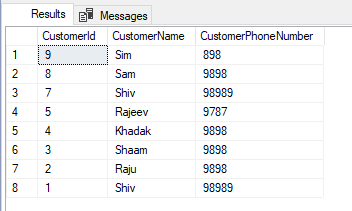
Select CustomerId,CustomerName,CustomerPhoneNumber from tblCustomer where CustomerName ='Shiv' or CustomerId=7



## SQL Practice 6: Sort data using ascending and descending.

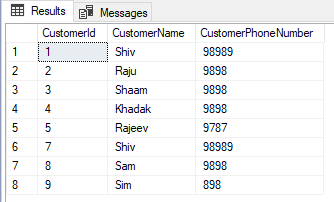
Using descending order:

Select CustomerId,CustomerName,CustomerPhoneNumber from tblCustomer order by CustomerId desc



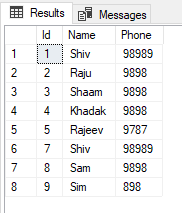
Using ascending order:

Select CustomerId,CustomerName,CustomerPhoneNumber from tblCustomer order by CustomerId,CustomerName asc



## SQL Practice 7: Provide user friendly ALIAS for column names.

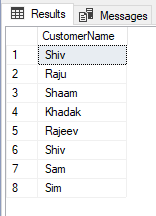
Select CustomerId as Id,CustomerName as Name,CustomerPhoneNumber as Phone from tblCustomer order by CustomerId,CustomerName asc



## SQL Practice 8: Display unique records from a table.

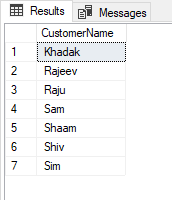
Before:

Select CustomerName from tblCustomer



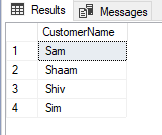
After:

Select distinct CustomerName from tblCustomer

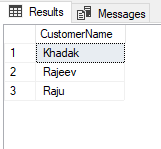


## SQL Practice 9: Searching using pattern and wildcards.

Select distinct CustomerName from tblCustomer where CustomerName like 's%'

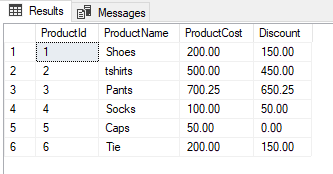


Select distinct CustomerName from tblCustomer where CustomerName like '[^s]%'



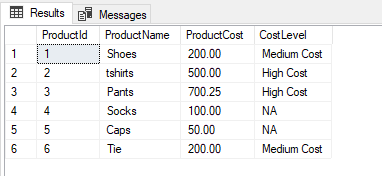
## SQL Practice 10: Create runtime calculated columns.

Select ProductId,ProductName,ProductCost,(ProductCost - 50) as Discount from tblProduct



## SQL Practice 11: CASE statements with SQL.

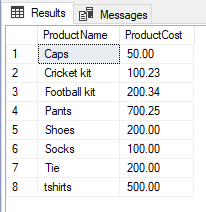
Select ProductId,ProductName,ProductCost, (case when ProductCost >100 and ProductCost <=200 then 'Medium Cost' when ProductCost >201 then 'High Cost' else 'NA' end) as CostLevel from tblProduct



## SQL Practice 12: Join data from two SELECTS using UNION and UNION ALL.

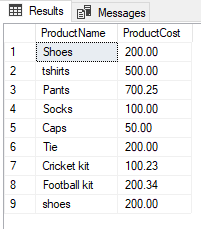
Using UNION:

select ProductName,ProductCost from tblProduct union select AncillaryName,AncillaryCost from tblAncillary



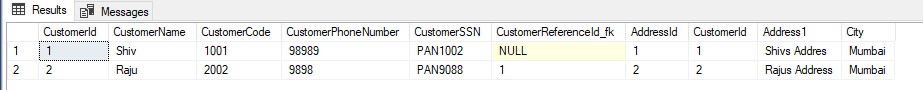
Using UNION ALL:

select ProductName,ProductCost from tblProduct union all select AncillaryName,AncillaryCost from tblAncillary



## SQL Practice 13: Show matching data from two tables. (Inner Join)

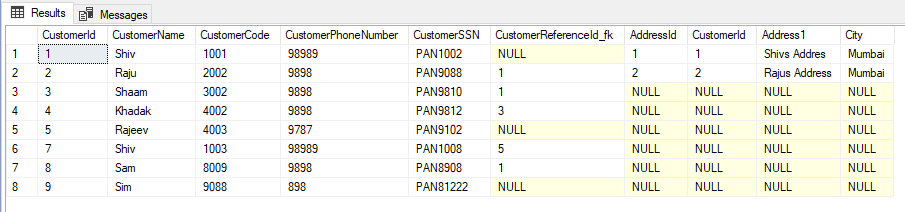
select \* from tblCustomer inner join tblAddress on tblCustomer.CustomerId = tblAddress.CustomerId



## SQL Practice 14: Show all records from one table and only matching record from other table. (Left and Right Join)

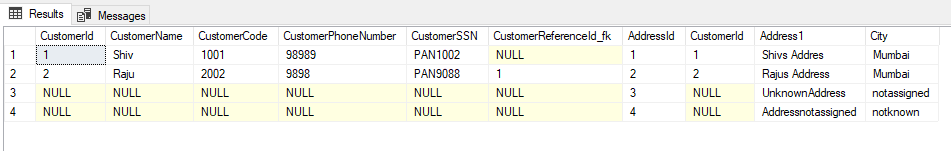
Left join:

select \* from tblCustomer left join tblAddress on tblCustomer.CustomerId = tblAddress.CustomerId



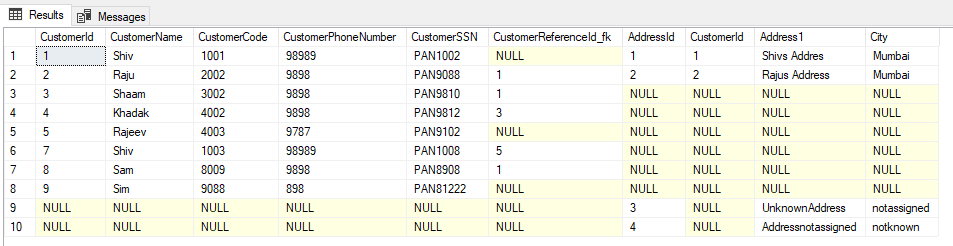
Right join:

select \* from tblCustomer Right join tblAddress on tblCustomer.CustomerId = tblAddress.CustomerId



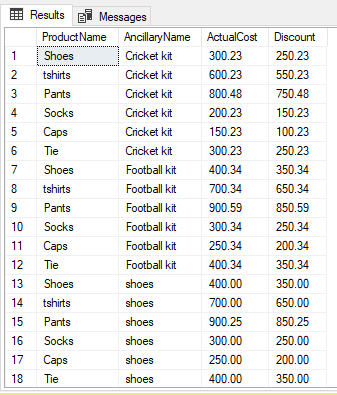
## SQL Practice 15: Show all records from matching or unmatching. (Full Outer Join)

select \* from tblCustomer full outer join tblAddress on tblCustomer.CustomerId = tblAddress.CustomerId



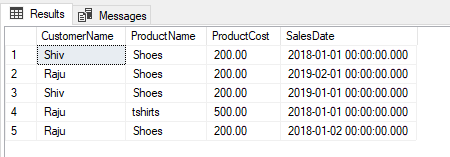
## SQL Practice 16: Show Cartesian of two tables. (Cross join)

select tblProduct.ProductName,tblAncillary.AncillaryName ,(tblProduct.ProductCost + tblAncillary.AncillaryCost)as ActualCost, (tblProduct.ProductCost + tblAncillary.AncillaryCost) - 50 as Discount from tblProduct cross join tblAncillary



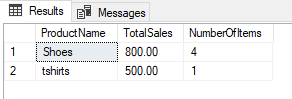
## SQL Practice 17: Writing a complex SQL inner join statements.

select c.CustomerName,p.ProductName,p.ProductCost,pc.SalesDate from tblProductCustomer as pc inner join tblCustomer as c on pc.CustomerId\_fk =c.CustomerId inner join tblProduct as p on pc.Productid\_fk =p.ProductId



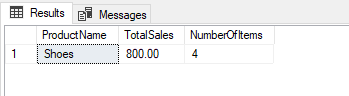
## SQL Practice 18: Display aggregate values from a table. (Group BY)

select p.ProductName,sum(p.ProductCost)as TotalSales, count(\*) as NumberOfItems from tblProductCustomer as pc inner join tblProduct as p on pc.Productid\_fk =p.ProductId group by p.ProductName



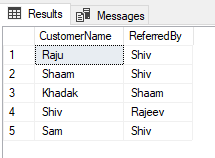
## SQL Practice 19: Filter on Aggregate values. (HAVING CLAUSE)

select p.ProductName,sum(p.ProductCost)as TotalSales, count(\*) as NumberOfItems from tblProductCustomer as pc inner join tblProduct as p on pc.Productid\_fk =p.ProductId group by p.ProductName having count(\*) > 1



## SQL Practice 20: Self Join.

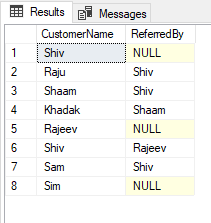
select t1.CustomerName as CustomerName,t2.CustomerName as ReferredBy from tblCustomer t1 inner join tblCustomer t2 on t1.CustomerReferenceId\_fk = t2.CustomerId



## SQL Practice 21: ISNULL.

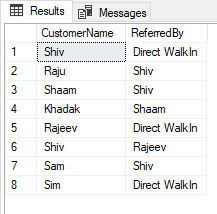
Before ISNULL:

select t1.CustomerName as CustomerName,t2.CustomerName as ReferredBy from tblCustomer t1 left join tblCustomer t2 on t1.CustomerReferenceId\_fk = t2.CustomerId



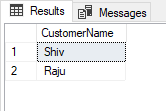
After ISNULL:

select t1.CustomerName as CustomerName,isnull(t2.CustomerName,'Direct WalkIn') as ReferredBy from tblCustomer t1 left join tblCustomer t2 on t1.CustomerReferenceId\_fk = t2.CustomerId



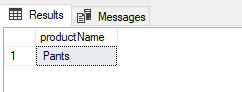
## SQL Practice 22: Sub Queries.

select tblCustomer.CustomerName from tblCustomer where tblCustomer.CustomerId in (select tblProductCustomer.CustomerId\_fk from tblProductCustomer )



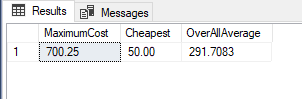
## SQL Practice 23: Co-related Queries.

select t1.productName from tblProduct as t1 where 1 = (select count(\*) from tblProduct t2 where t2.ProductCost >= t1.ProductCost )



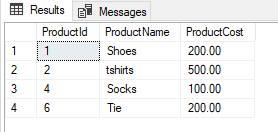
## SQL Practice 24: Find Max, Min and Average.

select max(ProductCost) as MaximumCost, min(ProductCost) as Cheapest, avg(ProductCost) as OverAllAverage from tblProduct



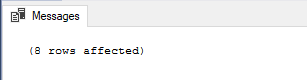
## SQL Practice 25: Find the between numeric values.

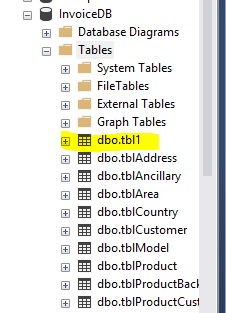
select \* from tblProduct where ProductCost between 100 and 500



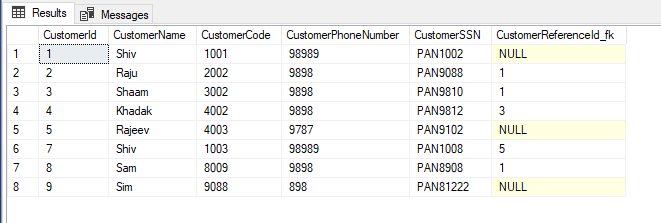
## SQL Practice 26: Dump table data in to new table. (SELECT INTO)

select \* into tbl1 from tblCustomer



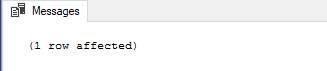


SELECT \* FROM tbl1

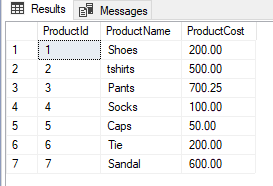


## SQL Practice 27: Insert data in to table.

insert into tblProduct (ProductId ,ProductName,ProductCost) values (7,'Sandal',600)



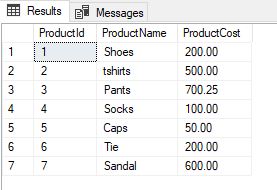
select \* FROM tblProduct



## SQL Practice 28: Insert bulk data in existing table.

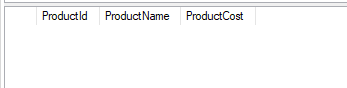
Before Insert data:

select \* FROM tblProduct

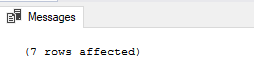


After Insert data:

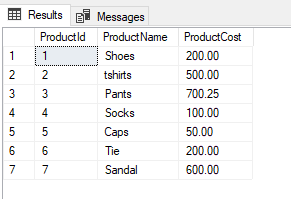
select \* from tblProductBackup



insert into tblProductBackup select \* from tblProduct



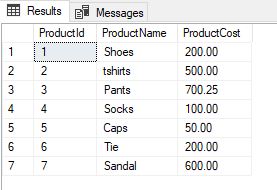
select \* from tblProductBackup



## SQL Practice 29: Update data in to table.

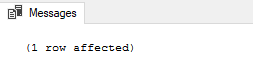
Before Updating:

select \* FROM tblProduct

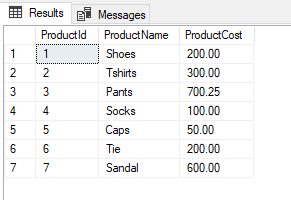


After Updating:

update tblProduct set ProductName ='Tshirts', ProductCost=300 where ProductId =2



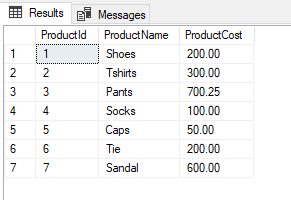
select \* FROM tblProduct



## SQL Practice 30: Delete data from a table.

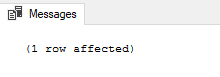
Before Delete:

select \* FROM tblProduct

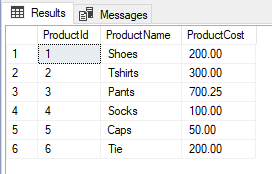


After Delete:

delete from tblProduct where ProductId =7



select \* FROM tblProduct

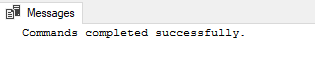


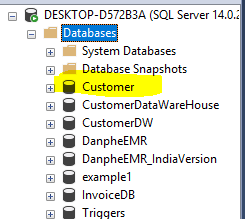
# Data Definition Language

## SQL Practice 31: Create Database Query.

Create Database:

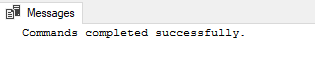
create database Customer

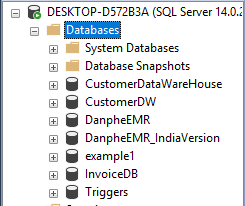




Drop Database:

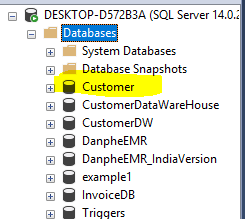
drop database Customer

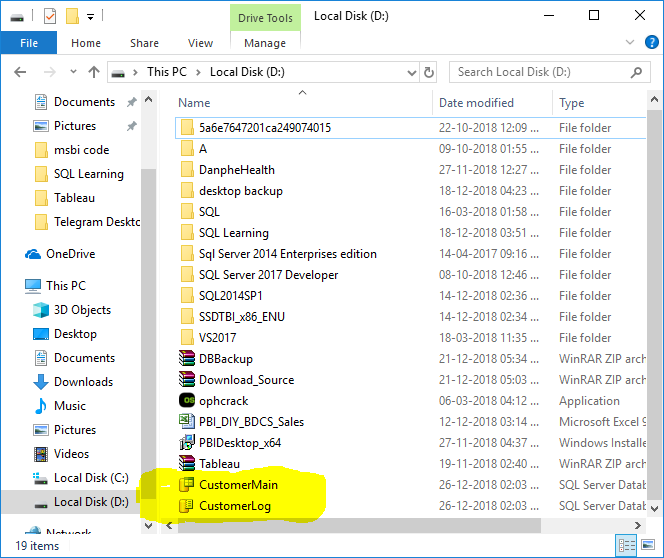




## SQL Practice 32: Create Database with FileName, Size, MaxSize and Growth.

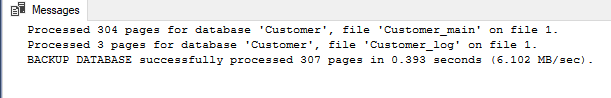
create database Customer on ( name= Customer\_main, filename='D:\CustomerMain.mdf', size=10, maxsize=50, filegrowth=5) log on ( name= Customer\_log, filename='D:\CustomerLog.ldf', size=5, maxsize=25, filegrowth=5)

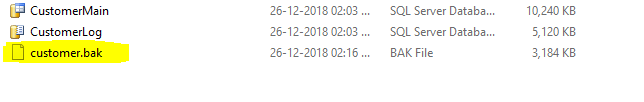




## SQL Practice 33: Backup Database.

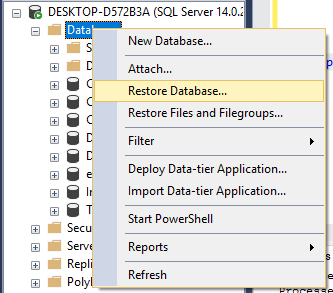
backup database Customer to disk= 'D:\customer.bak'



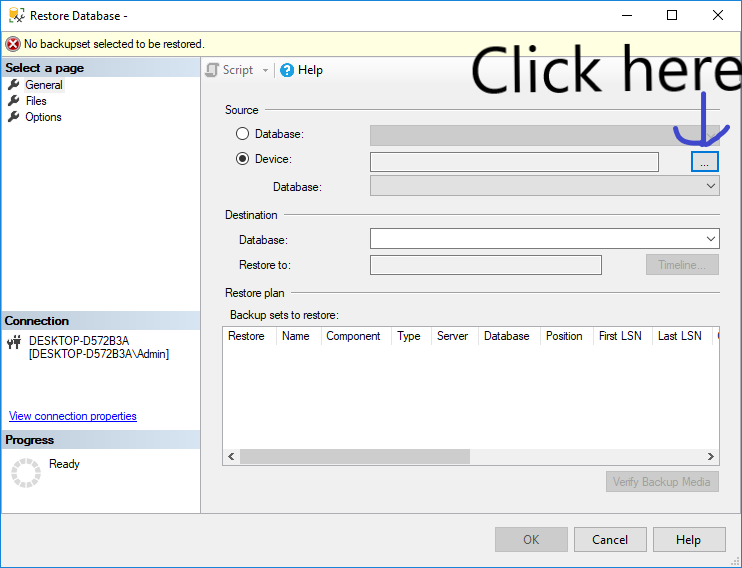


## SQL Practice 34: Restore Database.

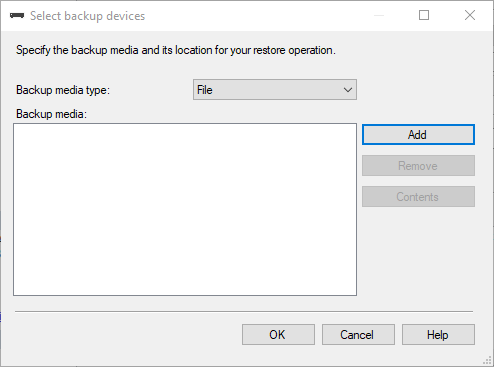
Step 1: Right click on databases folder and select restore database option.

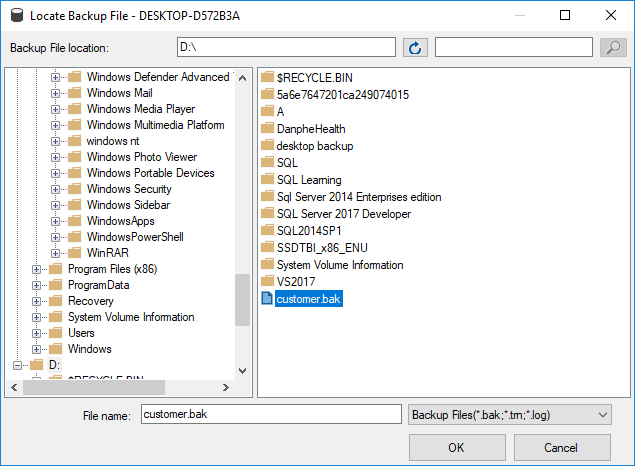


Step 2: Select Device option and click on … button.

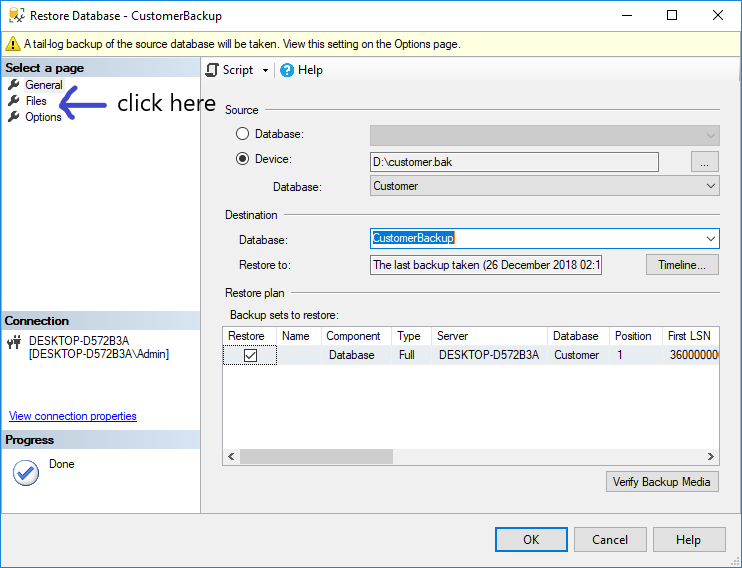


Step 3: Click on ADD button and Select the path of backup file.

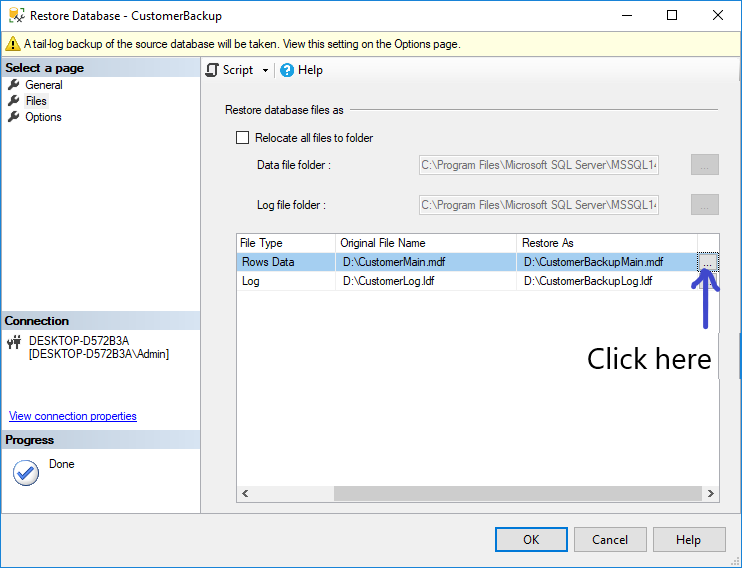


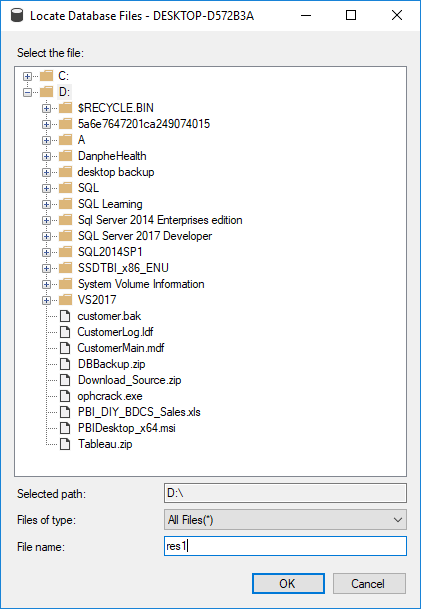


Step 4: change the name of destination database and select file option.

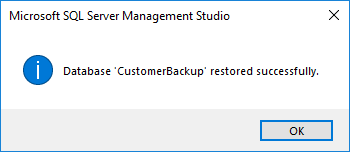


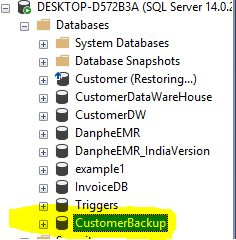
Step 5: Click on … button select location and set the file name and click on ok.





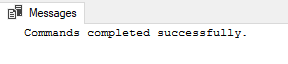
Step 6: click on ok.

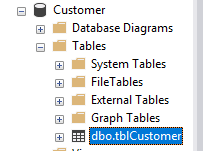




## SQL Practice 35: Create table.

use Customer go create table tblCustomer (id int, Name nvarchar(50) )

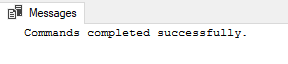


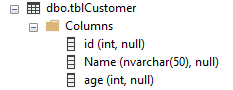


## SQL Practice 36: Alter a table structure drop a existing column.

Add Column:

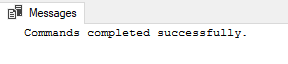
alter table tblCustomer add age int

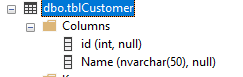




Drop Column:

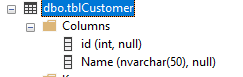
alter table tblCustomer drop column age





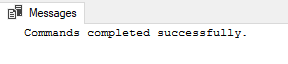
## SQL Practice 37: Alter a table structure change an existing column.

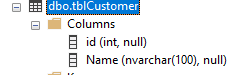
Before change:



After change:

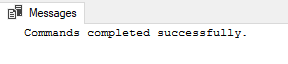
alter table tblCustomer alter column name nvarchar(100)

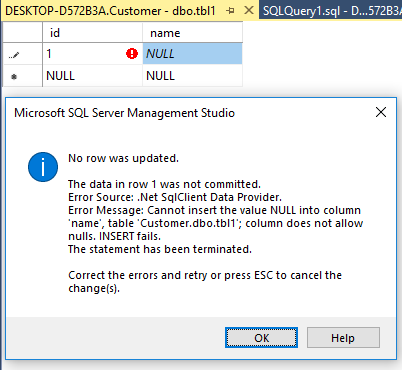


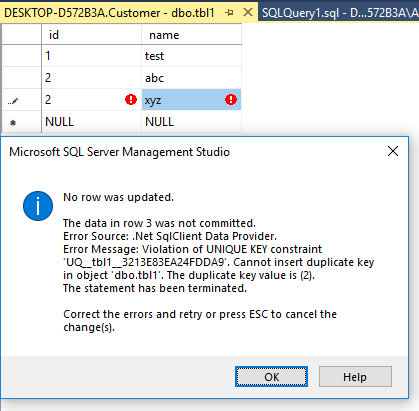


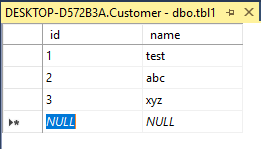
## SQL Practice 38: Adding validation constraints NOT NULL and UNIQUE.

use Customer go create table tbl1 (id int unique, name nvarchar(50) not null)



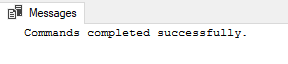


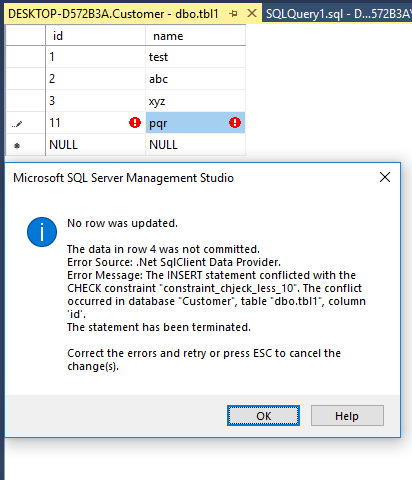


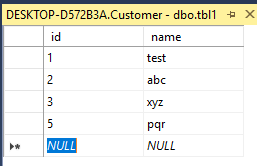


## SQL Practice 39: Check Constraints for data value validation.

alter table tbl1 add constraint constraint\_chjeck\_less\_10 check (id < 10 )

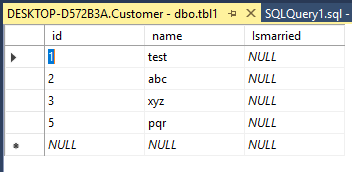






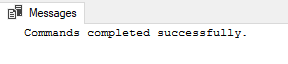
## SQL Practice 40: Provide Default values for a table.

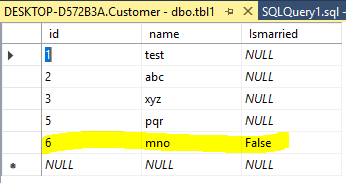
Before:



After:

alter table tbl1 add constraint ismarried\_default default 0 for Ismarried

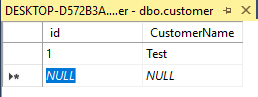




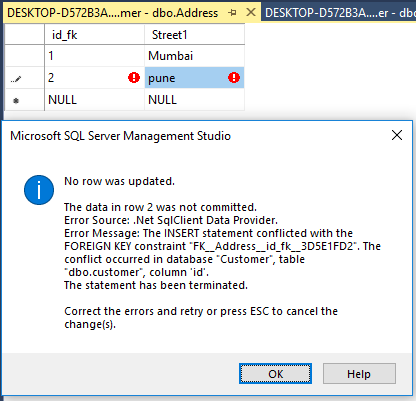
## SQL Practice 41: Creating Primary and Foreign key constraints.

create table customer (id int primary key, CustomerName nvarchar(20) not null) create table Address (id\_fk int foreign key references customer(id), Street1 nvarchar(50) )

Customer table:

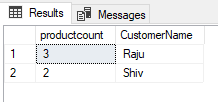


Address table:



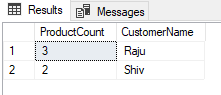
## SQL Practice 42: Get Customer who purchases more than one product.

select Count(P.ProductId) as productcount , C.CustomerName from tblProductCustomer PC inner join tblProduct P on PC.Productid\_fk = p.ProductId inner join tblCustomer C on PC.CustomerId\_fk = C.CustomerId group by C.CustomerName having Count(P.ProductId) > 1



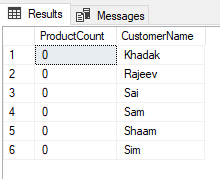
## SQL Practice 43: Get Customer and how much product they have purchased.

select Count(C.CustomerId) as ProductCount, C.CustomerName from tblCustomer C left join tblProductCustomer PC on C.CustomerId = pc.CustomerId\_fk inner join tblProduct P on PC.Productid\_fk = p.ProductId group by C.CustomerName



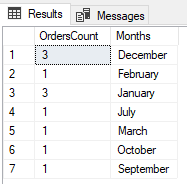
## SQL Practice 44: Get Customer who has not purchased any product.

select Count(P.ProductId) as ProductCount, C.CustomerName from tblCustomer C left join tblProductCustomer PC on C.CustomerId = pc.CustomerId\_fk left join tblProduct P on PC.Productid\_fk = p.ProductId group by C.CustomerName having Count(P.ProductId) <=0



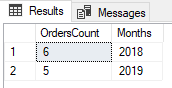
## SQL Practice 45: Get month wise order between the financial year.

select Count(O.OrderId) OrdersCount , datename(MONTH,O.OrderDate) as Months from Orders O where Orderdate between '2018-04-01' and '2019-03-31' group by datename(MONTH,O.OrderDate)



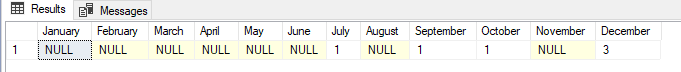
## SQL Practice 46: Get year wise order between the financial year.

select Count(O.OrderId) OrdersCount , datename(YEAR,O.OrderDate) as Months from Orders O where Orderdate between '2018-04-01' and '2019-03-31' group by datename(YEAR,O.OrderDate)



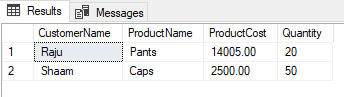
## SQL Practice 47: Get month wise order between the ‘2018-01-01’ to ‘2018-12-31’ but Row-wise Month.

select \* from ( select Count(O.OrderId) OrdersCount , datename(MONTH,O.OrderDate) as Months from Orders O where Orderdate between '2018-01-01' and '2018-12-31' group by datename(MONTH,O.OrderDate) ) src pivot ( sum(OrdersCount) for Months in (January, February,March,April,May, June, July,August,September,October,November,December) ) piv;



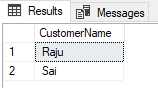
## SQL Practice 48: Get Customer Name, Product Name, Product Cost, Quantity from Invoice Items whose Product Cost must be greater than 1000 and Quantity between 10 to 50.

select c.CustomerName ,P.ProductName , IV.ProductCost , IV.Quantity from Orders O right join OrderItems OI on O.OrderId =OI.OrderId right join Invoices V on O.OrderId = V.OrderId right join InvoiceItems IV on V.InvoiceId = IV.InvoiceId right join tblProduct P on IV.ProductId = P.ProductId right join tblCustomer C on O.CustomerId = c.CustomerId where IV.Quantity between 10 and 50 and IV.ProductCost > 1000 group by c.CustomerName ,P.ProductName ,IV.ProductCost , IV.Quantity

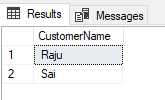


## SQL Practice 49: Get Names of customer who have purchase pants and T-shirts both.

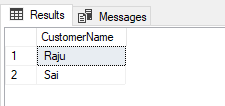
SELECT Pants.CustomerName FROM ( select C.CustomerName from [dbo].[InvoiceItems] IV inner join Invoices I on IV.InvoiceId = I.InvoiceId inner join Orders O on I.OrderId = O.OrderId Inner join tblProduct p on IV.ProductId = P.ProductId Inner join tblCustomer C on O.CustomerId = C.CustomerId where IV.ProductId = 3 Group by C.CustomerName )AS Pants INNER JOIN ( select C.CustomerName from [dbo].[InvoiceItems] IV inner join Invoices I on IV.InvoiceId = I.InvoiceId inner join Orders O on I.OrderId = O.OrderId Inner join tblProduct p on IV.ProductId = P.ProductId Inner join tblCustomer C on O.CustomerId = C.CustomerId where IV.ProductId = 2 Group by C.CustomerName ) as Tshirts on Tshirts.CustomerName = Pants.CustomerName



select a.CustomerName from tblCustomer as a inner join ( select CustomerId,ProductId from [dbo].[InvoiceItems] IV inner join Invoices I on IV.InvoiceId = I.InvoiceId inner join Orders O on I.OrderId = O.OrderId where ProductId in(2,3) group by CustomerId,ProductId ) as b on a.CustomerId=b.CustomerId group by a.CustomerName having Count(1)>1

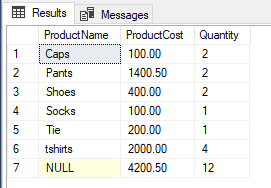


select C.CustomerName from [dbo].[InvoiceItems] IV inner join Invoices I on IV.InvoiceId = I.InvoiceId inner join Orders O on I.OrderId = O.OrderId Inner join tblProduct p on IV.ProductId = P.ProductId Inner join tblCustomer C on O.CustomerId = C.CustomerId where IV.ProductId = 2 INTERSECT select C.CustomerName from [dbo].[InvoiceItems] IV inner join Invoices I on IV.InvoiceId = I.InvoiceId inner join Orders O on I.OrderId = O.OrderId Inner join tblProduct p on IV.ProductId = P.ProductId Inner join tblCustomer C on O.CustomerId = C.CustomerId where IV.ProductId = 3



## SQL Practice 50: Get All Product Ordered Between '2018-01-01' and '2018-12-31' and display Product Name, Product Cost and Quantity along with Total at end.

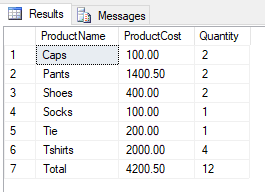
select P.ProductName as [ProductName] ,SUM(P.ProductCost) ProductCost, count(OI.Quantity) as Quantity from Orders O inner join OrderItems OI on O.OrderId = OI.OrderId inner join tblProduct P on OI.ProductId = P.ProductId where Orderdate between '2018-04-01' and '2019-03-31' group by rollup (P.ProductName)



## SQL Practice 51: Get All Product Ordered Between '2018-01-01' and '2018-12-31' and display Product Name, Product Cost and Quantity along with Total at end which will be sum of Product Cost and Quantity.

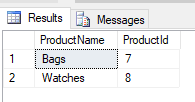
## Using "WITH common\_table\_expression".

;WITH Temp AS ( select P.ProductName as ProductName ,SUM(P.ProductCost) ProductCost, count(OI.Quantity) as Quantity from Orders O inner join OrderItems OI on O.OrderId = OI.OrderId inner join tblProduct P on OI.ProductId = P.ProductId where Orderdate between '2018-04-01' and '2019-03-31' group by (P.ProductName) ) SELECT ProductName, ProductCost ,Quantity FROM Temp UNION ALL SELECT 'Total' as ProductName, SUM(ProductCost) as ProductCost ,sum(Quantity) as Quantity FROM Temp



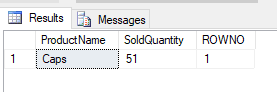
## SQL Practice 52: Get Products which are not purchased by ANY Customer.

select P.ProductName ,P.ProductId from tblProduct P left join OrderItems OI ON P.ProductId = OI.ProductId where OI.OrderId IS NULL



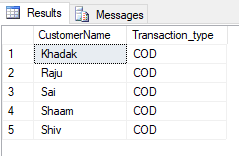
## SQL Practice 53: Get Highest Sold Products, Product Name and its Quantity

SELECT \* FROM ( select T.ProductName,T.SoldQuantity, ROW\_NUMBER()OVER(ORDER BY T.SoldQuantity DESC) AS ROWNO from ( select Sum(OI.Quantity) as SoldQuantity ,OI.ProductId, P.ProductName from OrderItems OI inner join Orders O on OI.OrderId = O.OrderId inner join tblProduct P on OI.ProductId = P.ProductId where Orderdate between '2018-04-01' and '2019-03-31' group by OI.ProductId , P.ProductName) T ) R where R.ROWNO = 1



## SQL Practice 54: Get the customers who had purchased products by Cash on Delivery.

select C.CustomerName , T.Transaction\_type from tblCustomer C inner join orders O on C.CustomerId = O.CustomerId inner join orderitems OI on O.OrderId = OI.OrderId inner join Invoices I on OI.OrderId = I.orderid inner join Transaction\_type T on I.Transaction\_type\_code = T.Transaction\_type\_code where T.Transaction\_type\_code = 1 group by C.CustomerName , T.Transaction\_type



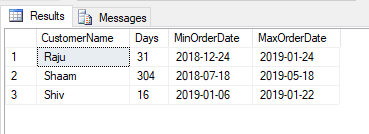
## SQL Practice 55: Get Weekend Sold Products between current financial year.

select \* from ( select T.ProductName,T.SoldQuantity, Case when DATEPART(WEEKDAY, T.OrderDate) = 1 then '1' when DATEPART(WEEKDAY, T.OrderDate) = 7 then '1' else '0' end as Weekend,DATENAME(dw,T.OrderDate) DayofWeek from ( select Sum(OI.Quantity) as SoldQuantity ,OI.ProductId, P.ProductName, O.OrderDate from OrderItems OI inner join Orders O on OI.OrderId = O.OrderId inner join tblProduct P on OI.ProductId = P.ProductId where Orderdate between '2018-04-01' and '2019-03-31' group by OI.ProductId , P.ProductName, O.OrderDate ) T ) W where W.Weekend = 1

## 

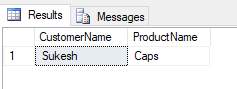
## SQL Practice 56: Get day’s difference between 2 Orders by customer.

SELECT CustomerName, DATEDIFF(DAY, MIN(o.OrderDate) , MAX(o.OrderDate)) as Days, MIN(o.OrderDate) as MinOrderDate, MAX(o.OrderDate) as MaxOrderDate FROM Orders O inner join tblCustomer C on O.CustomerId = C.CustomerId GROUP BY CustomerName HAVING COUNT(\*) = 2;



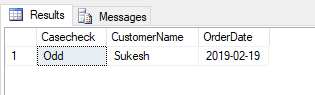
## SQL Practice 57: Get Customers whose Invoice is Not Generated but Order is placed.

select c.CustomerName ,P.ProductName from Orders O right join OrderItems OI on O.OrderId =OI.OrderId right join tblProduct P on OI.ProductId = P.ProductId right join tblCustomer C on O.CustomerId = C.CustomerId where O.OrderId not in ( select OrderId from Invoices)



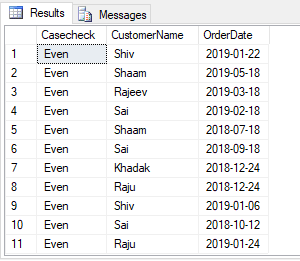
## SQL Practice 58: Get Customer Who has ordered products on Odd Dates.

select temp.Casecheck ,temp.CustomerName, temp.OrderDate from ( SELECT C.CustomerName , CASE WHEN DAY(O.OrderDate) % 2 =0 THEN 'Even' ELSE 'Odd' END Casecheck ,O.OrderDate from Orders O inner join tblCustomer C on O.CustomerId = C.CustomerId ) temp where temp.Casecheck = 'Odd'



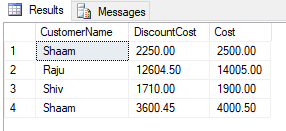
## SQL Practice 59: Get Customer Who has ordered products on Even Dates.

select temp.Casecheck ,temp.CustomerName, temp.OrderDate from ( SELECT C.CustomerName , CASE WHEN DAY(O.OrderDate) % 2 =0 THEN 'Even' ELSE 'Odd' END Casecheck ,O.OrderDate from Orders O inner join tblCustomer C on O.CustomerId = C.CustomerId ) temp where temp.Casecheck = 'Even'

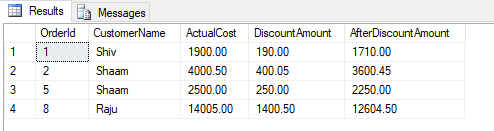


## SQL Practice 60: Give 10 % Discount to Customer who has purchased product who's Product Cost more than 1500.

select temp.CustomerName,cast (temp.Cost -(temp.Cost\*0.10)as decimal(10,2)) as DiscountCost ,temp.Cost from ( select C.CustomerName , sum (IV.ProductCost) as Cost from [dbo].[InvoiceItems] IV inner join Invoices I on IV.InvoiceId = I.InvoiceId inner join Orders O on I.OrderId = O.OrderId Inner join tblProduct p on IV.ProductId = P.ProductId Inner join tblCustomer C on O.CustomerId = C.CustomerId group by C.CustomerName , I.InvoiceDate ) temp where temp.Cost > 1500



select O.OrderId,c.CustomerName, ActualCost ,[DiscountAmount],AfterDiscountAmount from ( select InvoiceId,sum(ProductCost) ActualCost ,sum(ProductCost) \* 10 /100 as [DiscountAmount], sum(ProductCost) - (sum(ProductCost) \* 10 /100) as AfterDiscountAmount from InvoiceItems group by InvoiceId having sum(ProductCost) > 1500) as a inner join Invoices as b on a.InvoiceId = b.InvoiceId inner join Orders as O on b.OrderId = o.OrderId inner join tblCustomer as c on O.CustomerId = c.CustomerId



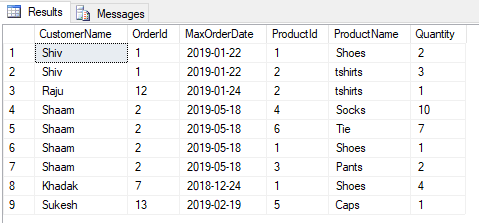
## SQL Practice 61: List Customer and a count of the products they purchased using Invoices and Invoice Items Tables.

SELECT Cust.CustomerName , ( SELECT count(IV.ProductId) FROM InvoiceItems IV inner join Invoices I on IV.InvoiceId = I.InvoiceId inner join Orders O on I.OrderId = O.OrderId Inner join tblProduct p on IV.ProductId = P.ProductId Inner join tblCustomer C on O.CustomerId = C.CustomerId WHERE O.CustomerId = Cust.CustomerId ) AS ProductCount FROM tblcustomer Cust

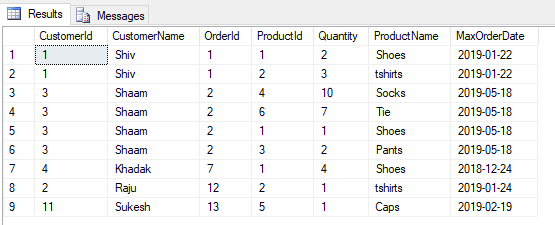
## 

## SQL Practice 62: Get List customers and all the details from their last order.

SELECT C.CustomerName,O.OrderId,O.OrderDate as MaxOrderDate, OI.ProductId, P.ProductName, OI.Quantity FROM tblCustomer as C INNER JOIN Orders O ON C.CustomerID = O.CustomerId INNER JOIN Orderitems OI ON O.OrderId = OI.OrderId INNER JOIN tblProduct P ON P.ProductId = OI.ProductId WHERE O.OrderDate = ( SELECT MAX(OrderDate) FROM Orders AS O2 WHERE O2.CustomerID = C.CustomerID )



SELECT a.CustomerId , b.CustomerName , o.OrderId , oi.ProductId , oi.Quantity , p.ProductName , a.MaxOrderDate FROM ( SELECT CustomerId ,max(OrderDate) MaxOrderDate FROM Orders GROUP BY CustomerId ) AS a INNER JOIN tblCustomer AS b ON a.CustomerId = b.CustomerId INNER JOIN Orders AS o ON a.CustomerId = o.CustomerId AND a.MaxOrderDate = o.OrderDate INNER JOIN OrderItems AS OI ON o.OrderId = OI.OrderId INNER JOIN tblProduct AS P ON oi.ProductId = p.ProductId



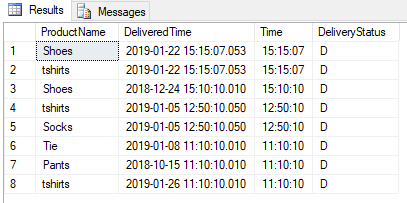
## SQL Practice 63: Get List of Customers who purchased Product Using Credit or Debit and had purchased cost is greater than 2000.

select C.CustomerName , T.Transaction\_Description from tblCustomer C inner join orders O on C.CustomerId = O.CustomerId inner join orderitems OI on O.OrderId = OI.OrderId inner join Invoices I on OI.OrderId = I.orderid inner join InvoiceItems IV on I.InvoiceId = Iv.InvoiceId inner join Transaction\_type T on I.Transaction\_type\_code = T.Transaction\_type\_code where T.Transaction\_type\_code = 3 and IV.ProductCost > 2000 group by C.CustomerName , T.Transaction\_Description

## 

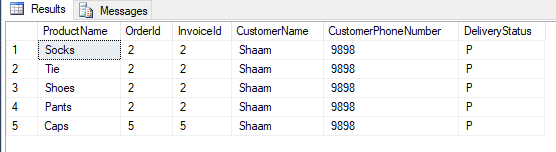
## SQL Practice 64: Get the list of Products which are delivered between 10 am to 4 pm.

select p.ProductName , D.DeliveredTime ,CONVERT(VARCHAR(8),D.DeliveredTime,108) as Time , D.DeliveryStatus from Invoices I inner Join InvoiceItems IV on I.InvoiceId = IV.InvoiceId inner join tblProduct P on IV.ProductId = P.ProductId inner join Delivery D on I.InvoiceId = D.InvoiceId where CONVERT(VARCHAR(8),D.DeliveredTime,108) between '10:00:00' and '16:00:00' and DeliveryStatus = 'D'



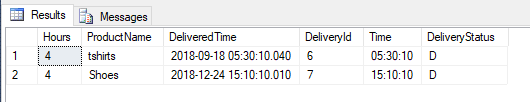
## SQL Practice 65: Get Order no and Customer Name and Phone No of Orders which is Not delivered.

select p.ProductName,O.OrderId , I.InvoiceId, C.CustomerName ,C.CustomerPhoneNumber , D.DeliveryStatus from Invoices I inner Join InvoiceItems IV on I.InvoiceId = IV.InvoiceId inner join tblProduct P on IV.ProductId = P.ProductId inner join Delivery D on I.InvoiceId = D.InvoiceId inner join Orders O on I.OrderId = O.OrderId inner join tblCustomer C on O.CustomerId = C.CustomerId where DeliveryStatus = 'P'



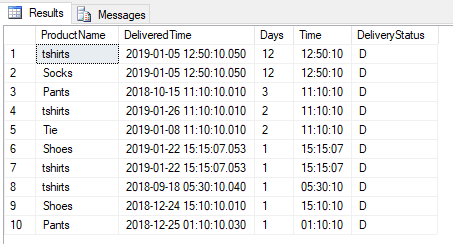
## SQL Practice 66: Get Product which is delivered in 4 hour.

select DATEDIFF(HH, D.PickupTime, D.DeliveredTime) as Hours, p.ProductName , D.DeliveredTime , D.DeliveryId ,CONVERT(VARCHAR(8),D.DeliveredTime,108) as Time , D.DeliveryStatus from Invoices I inner Join InvoiceItems IV on I.InvoiceId = IV.InvoiceId inner join tblProduct P on IV.ProductId = P.ProductId inner join Delivery D on I.InvoiceId = D.InvoiceId where DATEDIFF(HH, D.PickupTime, D.DeliveredTime) =4 and DeliveryStatus = 'D'



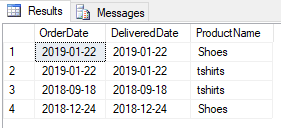
## SQL Practice 67: Get Each Product and its Delivery time and Days in descending order.

select p.ProductName , D.DeliveredTime , Case when DATEDIFF(day, D.PickupTime, D.DeliveredTime) = 0 then 1 else DATEDIFF(day, D.PickupTime, D.DeliveredTime)end as Days , CONVERT(VARCHAR(8),D.DeliveredTime,108) as Time , D.DeliveryStatus from Invoices I left Join InvoiceItems IV on I.InvoiceId = IV.InvoiceId inner join tblProduct P on IV.ProductId = P.ProductId inner join Delivery D on I.InvoiceId = D.InvoiceId where DeliveryStatus = 'D' order by Days desc



## SQL Practice 68: Get List of Orders which are ordered and Delivered on Same Day along Order Date and Delivered Date with Product Name.

select O.OrderDate , CONVERT(date,D.DeliveredTime,108) as DeliveredDate , p.ProductName from Orders O Inner join OrderItems OI on o.OrderId = oi.OrderId Inner join Invoices I on OI.OrderId = I.OrderId inner join Delivery D on I.InvoiceId = D.InvoiceId inner join tblProduct p on OI.ProductId = p.ProductId where O.OrderDate = CONVERT(date,D.DeliveredTime,108)



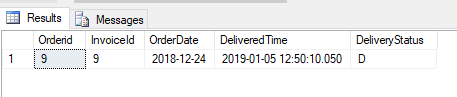
## SQL Practice 69: Get Products list along with Product Name, Delivered Time, Time and Delivery Status which are delivered within a week.

select p.ProductName , D.DeliveredTime ,CONVERT(VARCHAR(8),D.DeliveredTime,108) as Time , D.DeliveryStatus from Invoices I inner join Orders o on I.OrderId = o.OrderId inner Join InvoiceItems IV on I.InvoiceId = IV.InvoiceId inner join tblProduct P on IV.ProductId = P.ProductId inner join Delivery D on I.InvoiceId = D.InvoiceId where DATEDIFF(day, D.PickupTime, D.DeliveredTime) <= 7 and DeliveryStatus = 'D'

## 

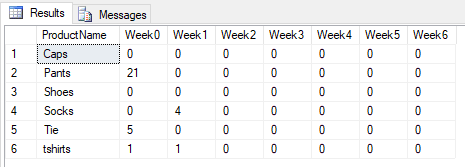
## SQL Practice 70: Get List of Deliveries which have only Delivered Product 'Socks'.

select O.Orderid,I.InvoiceId,O.OrderDate , D.DeliveredTime, D.DeliveryStatus from Invoices I inner join Orders o on I.OrderId = o.OrderId inner Join InvoiceItems IV on I.InvoiceId = IV.InvoiceId inner join tblProduct P on IV.ProductId = P.ProductId inner join Delivery D on I.InvoiceId = D.InvoiceId where P.ProductName = 'Socks'and DeliveryStatus = 'D'



## SQL Practice 71: Select ordered product quantities based on delivery date sorted by week.

select p.ProductName , sum(case when DATEDIFF(DAY,CONVERT(date,O.OrderDate,108),D.DeliveredTime)<=7 and DATEDIFF(DAY,CONVERT(date,O.OrderDate,108),D.DeliveredTime)>0 then oi.Quantity else 0 END) as Week0, sum(case when DATEDIFF(DAY,CONVERT(date,O.OrderDate,108),D.DeliveredTime)<=14 and DATEDIFF(DAY,CONVERT(date,O.OrderDate,108),D.DeliveredTime)>7 then oi.Quantity else 0 END) as Week1, sum(case when DATEDIFF(DAY,CONVERT(date,O.OrderDate,108),D.DeliveredTime)<=21 and DATEDIFF(DAY,CONVERT(date,O.OrderDate,108),D.DeliveredTime)>14 then oi.Quantity else 0 END) as Week2, sum(case when DATEDIFF(DAY,CONVERT(date,O.OrderDate,108),D.DeliveredTime)<=28 and DATEDIFF(DAY,CONVERT(date,O.OrderDate,108),D.DeliveredTime)>21 then oi.Quantity else 0 END) as Week3, sum(case when DATEDIFF(DAY,CONVERT(date,O.OrderDate,108),D.DeliveredTime)<=35 and DATEDIFF(DAY,CONVERT(date,O.OrderDate,108),D.DeliveredTime)>28 then oi.Quantity else 0 END) as Week4, sum(case when DATEDIFF(DAY,CONVERT(date,O.OrderDate,108),D.DeliveredTime)<=42 and DATEDIFF(DAY,CONVERT(date,O.OrderDate,108),D.DeliveredTime)>35 then oi.Quantity else 0 END) as Week5, sum(case when DATEDIFF(DAY,CONVERT(date,O.OrderDate,108),D.DeliveredTime)>42 and DATEDIFF(DAY,CONVERT(date,O.OrderDate,108),D.DeliveredTime)>0 then oi.Quantity else 0 END ) as Week6 from Orders O Inner join OrderItems OI on o.OrderId = oi.OrderId Inner join Invoices I on OI.OrderId = I.OrderId inner join Delivery D on I.InvoiceId = D.InvoiceId inner join tblProduct p on OI.ProductId = p.ProductId group by p.ProductName



# Date and Time Function

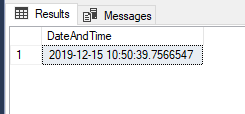
## **High Precision Function:**

## SQL Practice 72: Get the system date and time.

**SYSDATETIME:**

Returns the date and time of the machine the SQL Server is running on.

SELECT SYSDATETIME() AS 'DateAndTime'

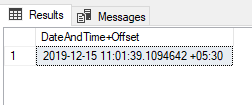


## SQL Practice 73: Get the system date and time with offset.

**SYSDATETIMEOFFSET:**

Returns the date and time of the machine the SQL Server is running on plus the offset from UTC.

SELECT SYSDATETIMEOFFSET() AS 'DateAndTime+Offset'

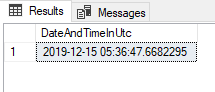


## SQL Practice 74: Get the system date and time on UTC.

**SYSUTCDATETIME :**

Returns the date and time of the machine the SQL Server is running on as UTC.

SELECT SYSUTCDATETIME() AS 'DateAndTimeInUtc'



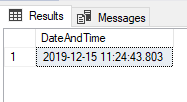
## **Less Precision Function:**

## SQL Practice 75: Get the system date and time.

**CURRENT\_TIMESTAMP:**

Returns the date and time of the machine the SQL Server is running on.

SELECT CURRENT\_TIMESTAMP AS 'DateAndTime'

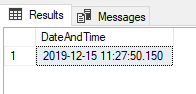


## SQL Practice 76: Get the system date and time.

**GETDATE() :**

Returns the date and time of the machine the SQL Server is running on.

SELECT GETDATE() AS 'DateAndTime'

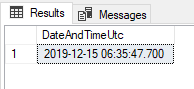


## SQL Practice 77: Get the system date and time on UTC.

**GETUTCDATE():**

Returns the date and time of the machine the SQL Server is running on as UTC.

SELECT GETUTCDATE() AS 'DateAndTimeUtc'



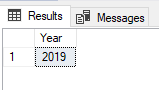
## **Date Name Function:**

## SQL Practice 78: Get the Year from date.

**DATENAME:**

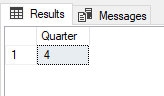
Returns a string corresponding to the date part specified.

SELECT DATENAME(YEAR, GETDATE()) AS 'Year'



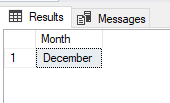
## SQL Practice 79: Get the Quarter from date.

SELECT DATENAME(QUARTER, GETDATE()) AS 'Quarter'



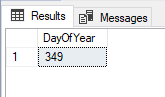
## SQL Practice 80: Get the Month from date.

SELECT DATENAME(MONTH, GETDATE()) AS 'Month'



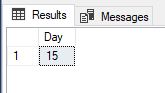
## SQL Practice 81: Get the Day of Year from date.

SELECT DATENAME(DAYOFYEAR, GETDATE()) AS 'DayOfYear'



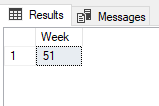
## SQL Practice 82: Get the Day from date.

SELECT DATENAME(DAY, GETDATE()) AS 'Day'



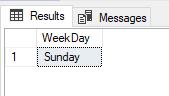
## SQL Practice 83: Get the Week from date.

SELECT DATENAME(WEEK, GETDATE()) AS 'Week'



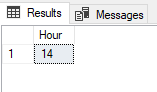
## SQL Practice 84: Get the Week Day from date.

SELECT DATENAME(WEEKDAY, GETDATE()) AS 'WeekDay'



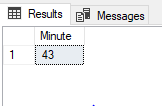
## SQL Practice 85: Get the Hour from date.

SELECT DATENAME(HOUR, GETDATE()) AS 'Hour'



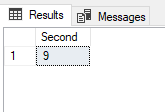
## SQL Practice 86: Get the Minute from date.

SELECT DATENAME(MINUTE, GETDATE()) AS 'Minute'



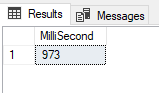
## SQL Practice 87: Get the Second from date.

SELECT DATENAME(SECOND, GETDATE()) AS 'Second'



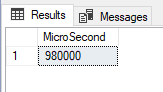
## SQL Practice 88: Get the Millisecond from date.

SELECT DATENAME(MILLISECOND, GETDATE()) AS 'MilliSecond'



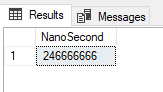
## SQL Practice 89: Get the Microsecond from date.

SELECT DATENAME(MICROSECOND, GETDATE()) AS 'MicroSecond'



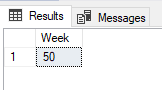
## SQL Practice 90: Get the Nanosecond from date.

SELECT DATENAME(NANOSECOND, GETDATE()) AS 'NanoSecond'



## SQL Practice 91: Get the Week from date.

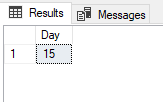
SELECT DATENAME(ISO\_WEEK, GETDATE()) AS 'Week'



**The DatePart function is similar to DateName function but it returns only integer values.**

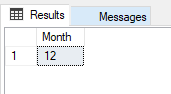
## SQL Practice 92: Get the Day from date.

SELECT DAY(GETDATE()) AS 'Day'



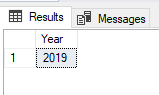
## SQL Practice 93: Get the Month from date.

SELECT MONTH(GETDATE()) AS 'Month'



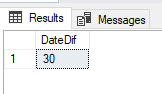
## SQL Practice 94: Get the Year from date.

SELECT YEAR(GETDATE()) AS 'Year'



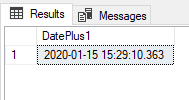
## SQL Practice 95: Get the Difference between two dates.

SELECT DATEDIFF(DAY, 2019-31-12, 2019-01-12) AS 'DateDif'



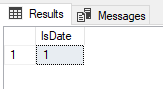
## SQL Practice 96: Add the intervals in date.

SELECT DATEADD(Month,1,GETDATE()) AS 'DatePlus1'



## SQL Practice 97: Check Date and Time is valid or not.

SELECT ISDATE(GETDATE()) AS 'IsDate'



**Returns 1 if a valid datetime type and 0 if not.**

# String Function

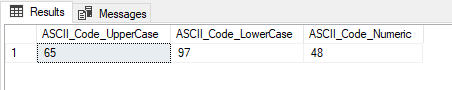
## SQL Practice 98: Get the ASCII code.

ASCII (): Returns the ASCII code of the given character expression.

select ASCII('A') as ASCII\_Code\_UpperCase,

ASCII('a') as ASCII\_Code\_LowerCase,

ASCII('0') as ASCII\_Code\_Numeric



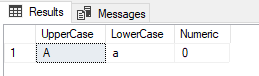
## SQL Practice 99: Get the Character from given ASCII code.

Char(): Converts an integer ASCII to a Character and Range is 0 to 255.

select Char(65) as UpperCase,

Char(97) as LowerCase,

Char(48) as Numeric



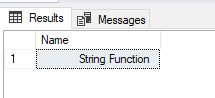
## SQL Practice 100: Remove spaces from given expression.

LTrim(): Remove blanks on the left hand side of the given expression.

RTrim(): Remove blanks on the right hand side of the given expression.

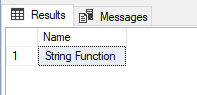
Before Removing Space:

select ' String Function'as Name



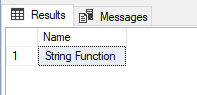
After Removing Space:

select LTRIM(' String Function')as Name



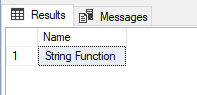
Before Removing Space:

select ('String Function ')as Name



After Removing Space:

select RTRIM('String Function ')as Name



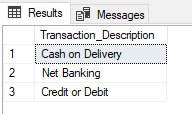
Note: In the Rtrim() function no any difference shows in before and after removing space because the right hand side of the expression spaces are not consider by default.

## SQL Practice 101: Converts all character in upper case from given expression.

Upper(): Converts all characters in upper case

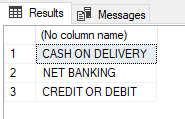
Before Upper Case:

Select Transaction\_Description from Transaction\_type



After Upper Case:

Select UPPER(Transaction\_Description) from Transaction\_type

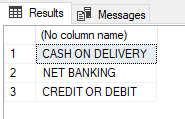


## SQL Practice 102: Converts all character in Lower case from given expression.

Lower(): Converts all characters in lower case.

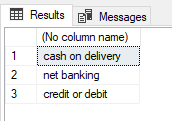
Before Lower Case:

Select UPPER(Transaction\_Description) from Transaction\_type



After Lower Case:

Select LOWER(Transaction\_Description) from Transaction\_type

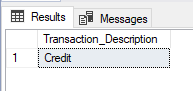


## SQL Practice 103: Converts all character in Reverse form.

Reverse(): Reverse all the characters from expression.

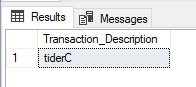
Before Reverse String:

Select substring(Transaction\_Description,1,6)as Transaction\_Description from Transaction\_type where Transaction\_type\_code=3



After Reverse String:

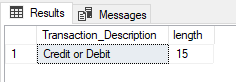
Select reverse(substring(Transaction\_Description,1,6))as Transaction\_Description from Transaction\_type where Transaction\_type\_code=3



## SQL Practice 104: How to calculate length of given Character expression.

Length(): Calculate length of given Characters.

Select Transaction\_Description,len(Transaction\_Description) as length from Transaction\_type where Transaction\_type\_code=3

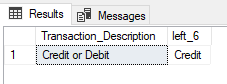


Note: In len() function also count blank spaces in length.

## SQL Practice 105: Get the left hand side 6 Character from given Character expression.

Left(): Returns the specified number of characters from left side from given character expression.

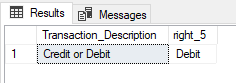
Select Transaction\_Description,LEFT(Transaction\_Description,6) as left\_6 from Transaction\_type where Transaction\_type\_code=3



## SQL Practice 106: Get the right hand side 5 Character from given Character expression.

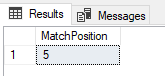
Right(): Returns the specified numbers of characters from right side from given character expression.

Select Transaction\_Description,RIGHT(Transaction\_Description,5) as right\_5 from Transaction\_type where Transaction\_type\_code=3



## SQL Practice 107: Get the specified characters Position from the given Character expression.

SELECT CHARINDEX('OM', 'Customer') AS MatchPosition

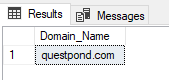


## SQL Practice 108: Get the Domain Name from the given Character expression.

SubString(): Returns part of the string from given expression.

Email Id- [questpond@questpond.com](mailto:questpond@questpond.com) Domain Name=questpond.com

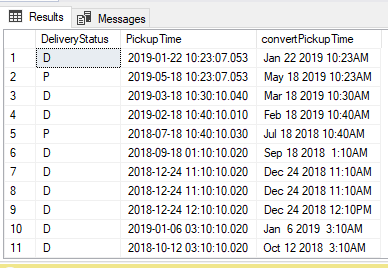
select Substring('questpond@questpond.com',CHARINDEX('@','questpond@questpond.com')+1,13) as Domain\_Name



# CAST and CONVERT:

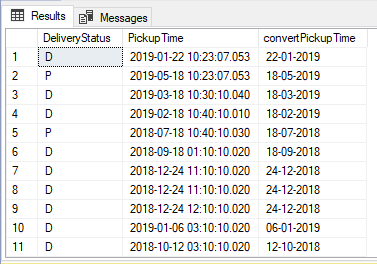
## SQL Practice 109: How to convert one data type to another using CAST.

select DeliveryStatus,PickupTime,cast(PickupTime as nvarchar) as convertPickupTime from Delivery



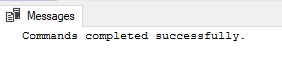
## SQL Practice 110: How to convert one data type to another using CONVERT with style (dd-mm-yy).

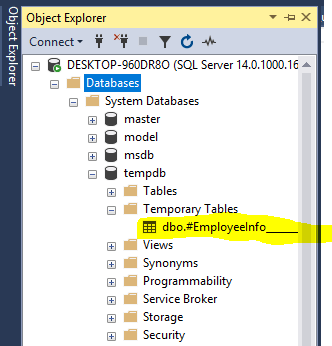
select DeliveryStatus,PickupTime,convert(nvarchar,PickupTime,105) as convertPickupTime from Delivery



## SQL Practice 110: How to create the temporary table.

create table #EmployeeInfo (Id int,Name nvarchar(25))



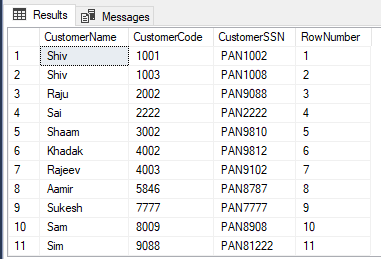


## SQL Practice 111: Get the row numbers of records using Row Number function.

select CustomerName,CustomerCode,CustomerSSN,

row\_number() over (order by CustomerCode) as RowNumber

from tblCustomer

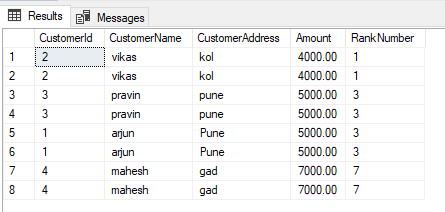


## SQL Practice 112: Get the row numbers of records using Rank function.

select CustomerId,CustomerName,CustomerAddress,Amount,

Rank() over (order by Amount) as RankNumber

from CustomerSales



## SQL Practice 113: Get the row numbers of records using Dense Rank function.

select CustomerId,CustomerName,CustomerAddress,Amount,

DENSE\_RANK() over (order by Amount) as DenseNumber

from CustomerSales

