LAB # 05

Task 01 :

Write a query to order employee first name in Descending Order.

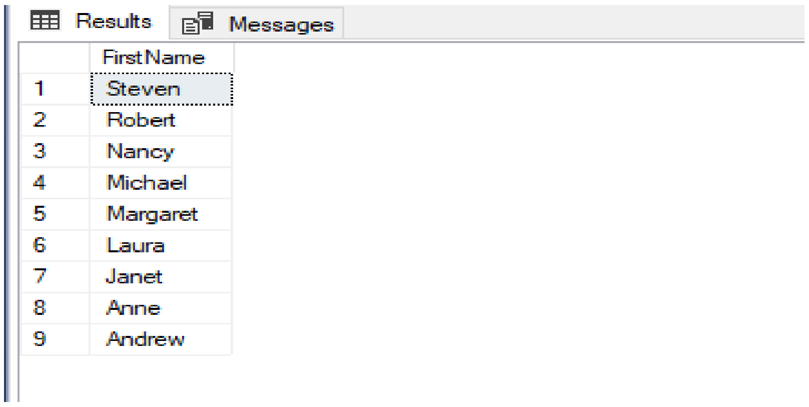
(Description)

The ORDER BY keyword is used to sort the result-set in ascending or descending order.The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.

(Query Text)

```
select FirstName from Employees  order by FirstName  desc
```

(Query Output)



	FirstName
1	Steven
2	Robert
3	Nancy
4	Michael
5	Margaret
6	Laura
7	Janet
8	Anne
9	Andrew

××××-------- ××××--------××××--------××××

Task 02 :

Display the highest, lowest, sum and average UnitPrice of each Category. Label column as CategoryId, Maximum, Minimum, Sum and Average, respectively. Round your results to the nearest whole number. (Table: Products)

(Description)

The MIN() function returns the smallest value of the selected column.

The MAX() function returns the largest value of the selected column.

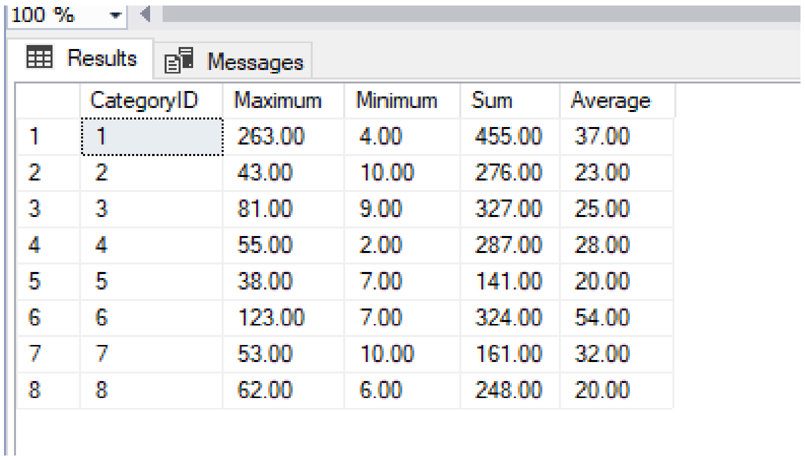
The AVG() function returns the average value of a numeric column.

The SUM() function returns the total sum of a numeric column.

(Query text)

```
select CategoryID,floor(Max(UnitPrice)) as 'Maximum' , floor(Min(UnitPrice)) as
'Minimum',floor(Sum(UnitPrice)) as Sum ,floor(AVG(UnitPrice)) as Average from Products
Group By CategoryID
```

(Query Output)



	CategoryID	Maximum	Minimum	Sum	Average
1	1	263.00	4.00	455.00	37.00
2	2	43.00	10.00	276.00	23.00
3	3	81.00	9.00	327.00	25.00
4	4	55.00	2.00	287.00	28.00
5	5	38.00	7.00	141.00	20.00
6	6	123.00	7.00	324.00	54.00
7	7	53.00	10.00	161.00	32.00
8	8	62.00	6.00	248.00	20.00

××××-------- ××××--------××××--------××××

Task 03 :

Display the highest, lowest, sum and average UnitPrice of each Category, where highest UnitPrice lies in the range of 50$ to 100$. Label column as CategoryId, Maximum, Minimum, Sum and Average, respectively. (Table: Products)

(Description)

The MIN() function returns the smallest value of the selected column.

The MAX() function returns the largest value of the selected column.

The AVG() function returns the average value of a numeric column.

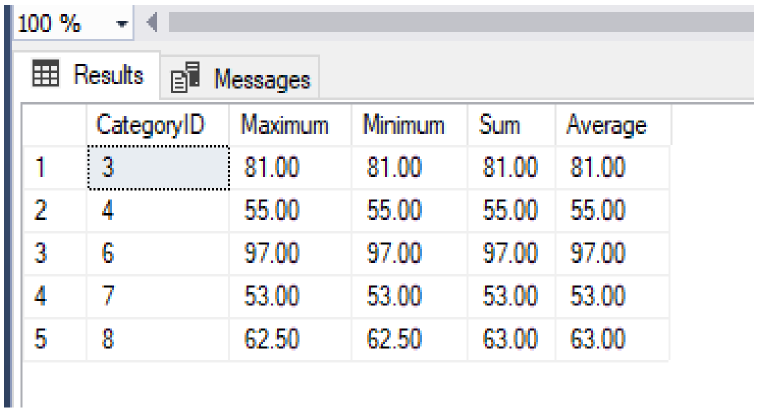
The SUM() function returns the total sum of a numeric column.

The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates.The AND operator displays a record if all the conditions separated by AND are TRUE.

(Query Text)

```sql
select CategoryID,Max(UnitPrice) as 'Maximum' , Min(UnitPrice) as 'Minimum',
CEILING(Sum(UnitPrice)) as Sum , CEILING(AVG(UnitPrice)) as Average from Products where
UnitPrice between 50 and 100 Group By CategoryID
```

(Query Output)



	CategoryID	Maximum	Minimum	Sum	Average
1	3	81.00	81.00	81.00	81.00
2	4	55.00	55.00	55.00	55.00
3	6	97.00	97.00	97.00	97.00
4	7	53.00	53.00	53.00	53.00
5	8	62.50	62.50	63.00	63.00

××××-------- ××××--------××××--------××××

Task 04 :

From customers table, Count all customers is each region where region is not null. (Table: Customers)

(Description)

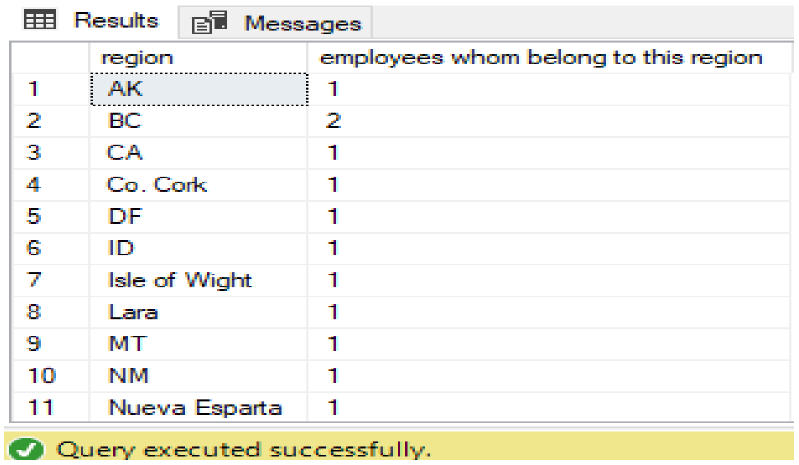
The COUNT() function returns the number of rows that matches a specified criterion.

A NOT NULL constraint in SQL is used to prevent inserting NULL values into the specified column, considering it as a not accepted value for that column

(Query Text)

```sql
select region,count(Region) as [employees whom belong to this region] from Customers
where Region is not null Group By Region
```

(Query Output)



Results | Messages

	region	employees whom belong to this region
1	AK	1
2	BC	2
3	CA	1
4	Co. Cork	1
5	DF	1
6	ID	1
7	Isle of Wight	1
8	Lara	1
9	MT	1
10	NM	1
11	Nueva Esparta	1

Query executed successfully.

××××-------- ××××--------××××--------××××

Task 05 :

Write a query to display the number of ContactName with same ContactTitle. Sort contact title in descending order. (Table: Customers)

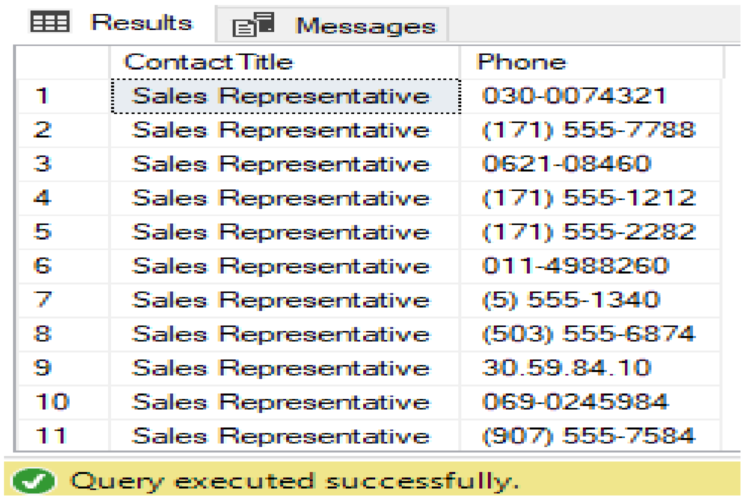
(Description)

The ORDER BY keyword is used to sort the result-set in ascending or descending order.The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.

(Query Text)

```
select ContactTitle,Phone from Customers order by ContactTitle desc
```

(Query Output)



Results | Messages

	ContactTitle	Phone
1	Sales Representative	030-0074321
2	Sales Representative	(171) 555-7788
3	Sales Representative	0621-08460
4	Sales Representative	(171) 555-1212
5	Sales Representative	(171) 555-2282
6	Sales Representative	011-4988260
7	Sales Representative	(5) 555-1340
8	Sales Representative	(503) 555-6874
9	Sales Representative	30.59.84.10
10	Sales Representative	069-0245984
11	Sales Representative	(907) 555-7584

Query executed successfully.

××××-------- ××××--------××××--------××××

Task 06 :

Write a query that count all orders against each product id. No of orders should be greater than 50. (Table: [Order Details])

(Description)

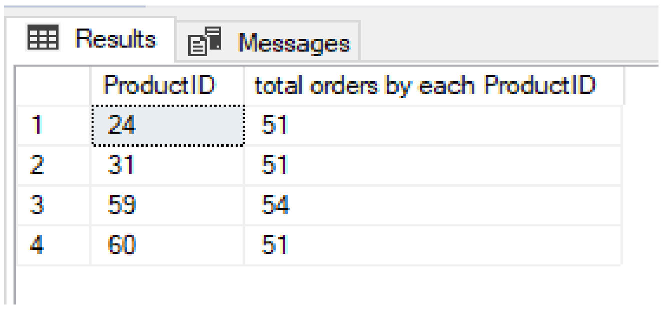
The COUNT() function returns the number of rows that matches a specified criterion.

The SQL Greater Than comparison operator (>) is used to compare two values. It returns TRUE if the first value is greater than the second. If the second is greater, it returns FALSE.

(Query Text)

```
select ProductID,count(Quantity) as [total orders by each ProductID] from [Order Details]
where  ProductID In (select ProductID  from [Order Details] group by ProductID having
count(Quantity) > 50)  group by ProductID
```

(Query Output)



	ProductID	total orders by each ProductID
1	24	51
2	31	51
3	59	54
4	60	51

××××-------- ××××--------××××--------××××

Task 07 :

How many people are in each unique city in the employee table that have more than one person in the city? Select the city and display the number of how many people are in each if it's greater than 1.(Table: Employees)

(Description)

The WHERE clause is used to filter records.

It is used to extract only those records that fulfill a specified condition.

The COUNT() function returns the number of rows that matches a specified criterion.

The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country".

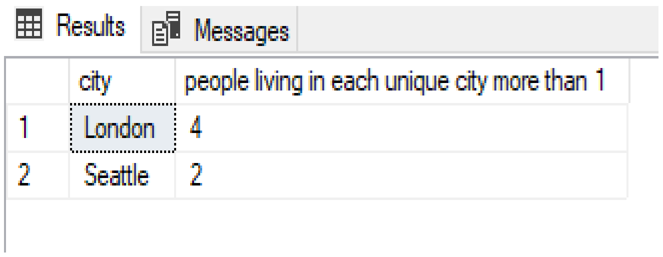
The GROUP BY statement is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns

The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions

(Query Text)

```
SELECT city, COUNT(City) as [people living in each unique city more than 1] FROM
Employees WHERE City IN (SELECT City FROM employees GROUP BY city HAVING COUNT(City)>1)
GROUP BY city
```

(Query Output)



	city	people living in each unique city more than 1
1	London	4
2	Seattle	2

××××-------- ××××--------××××--------××××

Task 08 :

List only those cities in which more than or equals to 2 employees are living.

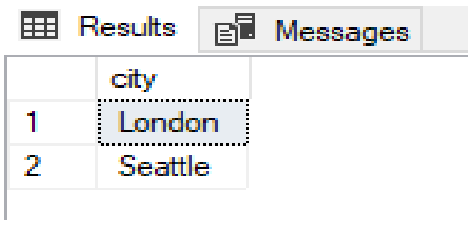
(Description)

The SQL Greater Than or Equal To comparison operator (>=) is used to compare two values. It returns TRUE if the first value is greater than or equal to the second. If the second is greater, it returns FALSE.

(Query Text)

```
select  distinct city from Employees  where city IN (SELECT City FROM employees GROUP BY
city HAVING COUNT(City)>=2)
```

(Query Output)


Results Messages

	city
1	London
2	Seattle

××××-------- ××××--------××××--------××××

Task 09 :

From the [Order Details] table, select the Product's id , maximum price and minimum price for each specific product in the table, sort the list by product id in ascending order.

(Description)

The MIN() function returns the smallest value of the selected column.

The MAX() function returns the largest value of the selected column.

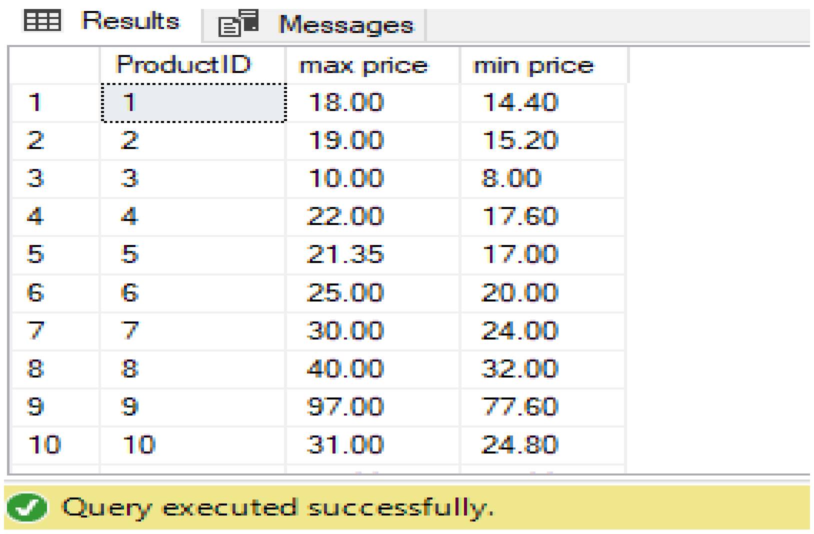
The ORDER BY keyword is used to sort the result-set in ascending or descending order.The ORDER BY keyword sorts the records in ascending order by default.

The GROUP BY statement is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns

(Query Text)

```
select ProductID,max(UnitPrice) as [max price] ,min(UnitPrice) as [min price] from [Order
Details] group by ProductID order by ProductID asc
```

(Query Output)


Results Messages

	ProductID	max price	min price
1	1	18.00	14.40
2	2	19.00	15.20
3	3	10.00	8.00
4	4	22.00	17.60
5	5	21.35	17.00
6	6	25.00	20.00
7	7	30.00	24.00
8	8	40.00	32.00
9	9	97.00	77.60
10	10	31.00	24.80

Query executed successfully.

××××-------- ××××--------××××--------××××

Task 10 :

Retrieve the number of employees in each city in which there are at least 2 employees.

(Description)

The less than equal to operator is used to test whether an expression (or number) is either less than or equal to another one.

The GROUP BY statement is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns

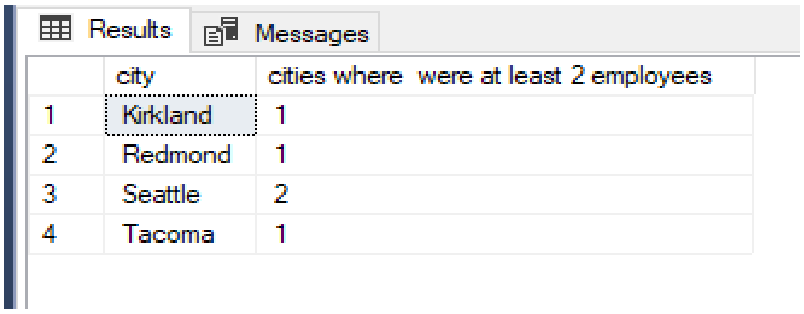
The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions

The COUNT() function returns the number of rows that matches a specified criterion.

(Query Text)

```
select city,count(city) as [cities where  were at least 2 employees  ]  from Employees
where city IN (SELECT City FROM employees GROUP BY city HAVING COUNT(City)<=2) GROUP BY
city
```

(Query Output)



	city	cities where were at least 2 employees
1	Kirkland	1
2	Redmond	1
3	Seattle	2
4	Tacoma	1

××××-------- ××××--------××××--------××××

Task 11:

Find the product name, maximum price and minimum price of each product having maximum price greater than 20.00 $. Order by maximum price.

(Description)

The MIN() function returns the smallest value of the selected column.

The MAX() function returns the largest value of the selected column.

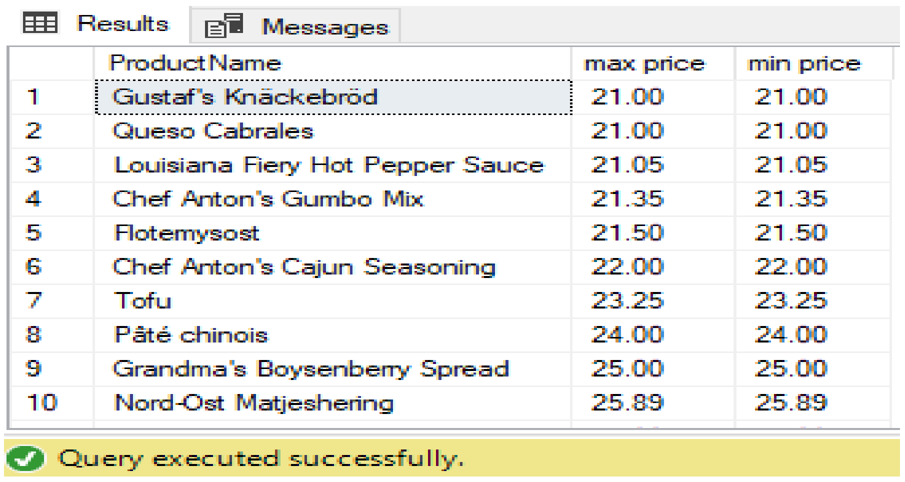
The ORDER BY keyword is used to sort the result-set in ascending or descending order.The ORDER BY keyword sorts the records in ascending order by default.

The SQL Greater Than or Equal To comparison operator (>) is used to compare two values. It returns TRUE if the first value is greater than to the second. If the second is greater, it returns FALSE.

(Query Text)

```
select ProductName, max(UnitPrice) as [max price] ,min(UnitPrice) as [min price] from
Products where UnitPrice > 20 group by ProductName order by max(UnitPrice)
```

(Query Output)



	ProductName	max price	min price
1	Gustaf's Knäckebröd	21.00	21.00
2	Queso Cabrales	21.00	21.00
3	Louisiana Fiery Hot Pepper Sauce	21.05	21.05
4	Chef Anton's Gumbo Mix	21.35	21.35
5	Flotemysost	21.50	21.50
6	Chef Anton's Cajun Seasoning	22.00	22.00
7	Tofu	23.25	23.25
8	Pâté chinois	24.00	24.00
9	Grandma's Boysenberry Spread	25.00	25.00
10	Nord-Ost Matjeshering	25.89	25.89

Query executed successfully.

xxxx-------- xxxx--------xxxx--------xxxx

Task 12:

Find the number of sales representatives in each city that contains at least 2 sales representatives. Order by the number of employees.

(Description)

The less than equal to operator(<=) is used to test whether an expression (or number) is either less than or equal to another one.

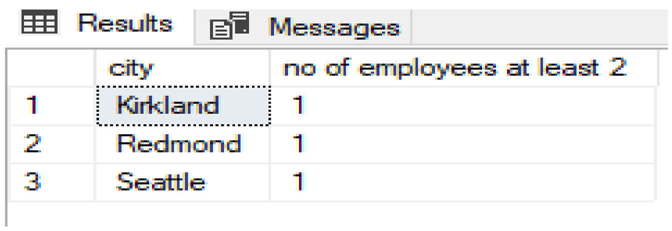
The COUNT() function returns the number of rows that matches a specified criterion.

The ORDER BY keyword is used to sort the result-set in ascending or descending order.The ORDER BY keyword sorts the records in ascending order by default.

(Query Text)

```
select city,count(title) as [no of employees at least 2] from Employees where title =
'Sales Representative' And city in(SELECT City FROM employees GROUP BY city HAVING
COUNT(City)<=2)  group by City  order by count(EmployeeID)
```

(Query Output)


Results | Messages

	city	no of employees at least 2
1	Kirkland	1
2	Redmond	1
3	Seattle	1

××××-------- ××××--------××××--------××××

Task 13:

From customers table, Count all customers in each region whose contactname contains manager and region is not null. (Table: Customers)

(Description)

The COUNT() function returns the number of rows that matches a specified criterion.

The GROUP BY statement is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns

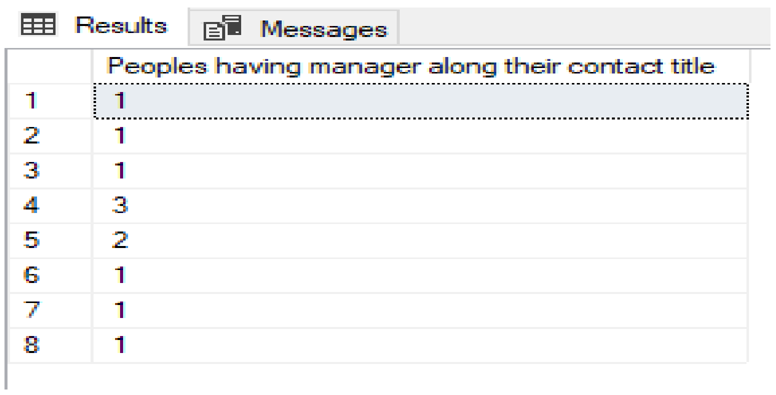
The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate

A NOT NULL constraint in SQL is used to prevent inserting NULL values into the specified column, considering it as a not accepted value for that column

(Query Text)

```
select count(CustomerID) as [Peoples having manager along their contact title] from
Customers  where ContactTitle like '%Manager' And region is not null  group by Region
```

(Query Output)


Results Messages

	Peoples having manager along their contact title
1	1
2	1
3	1
4	3
5	2
6	1
7	1
8	1

××××-------- ××××--------××××--------××××