SQL – SOLVED – QUERIES

- Nagendra V Kini

1) Show customer number, customer name, state and credit limit from customers table for below conditions. Sort the results by highest to lowest values of creditLimit.

   - State should not contain null values
   - credit limit should be between 50000 and 100000

Expected output:

customerNumber	customerName	state	creditLim
455	Super Scale Inc.	CT	95400.00
320	Mini Creations Ltd.	MA	94500.00
398	Tokyo Collectables, Ltd	Tokyo	94400.00
240	giftsbymail.co.uk	Isle of Wight	93900.00
282	Souveniers And Things Co.	NSW	93300.00
205	Toys4GrownUps.com	CA	90700.00
202	Canadian Gift Exchange Netw...	BC	90300.00
260	Royal Canadian Collectables, ...	BC	89600.00
462	FunGiftIdeas.com	MA	85800.00
495	Diecast Collectables	MA	85100.00
161	Technics Stores Inc.	CA	84600.00
175	Gift Depot Inc.	CT	84300.00
177	Osaka Souveniers Co.	Osaka	81200.00
339	Classic Gift Ideas, Inc	PA	81100.00

Query :-

```
select customerNumber,customerName,state,creditLimit from customers
where state is not null and
creditLimit between 50000 and 100000
order by creditLimit desc;
```

Output :-

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

customerNumber	customerName	state	creditLimit
455	Super Scale Inc.	CT	95400.00
320	Mini Creations Ltd.	MA	94500.00
398	Tokyo Collectables, Ltd	Tokyo	94400.00
240	giftsbymail.co.uk	Isle of Wight	93900.00
282	Souveniers And Things Co.	NSW	93300.00
205	Toys4GrownUps.com	CA	90700.00
202	Canadian Gift Exchange Network	BC	90300.00
260	Royal Canadian Collectables, Ltd.	BC	89600.00
462	FunGiftIdeas.com	MA	85800.00
495	Diecast Collectables	MA	85100.00
161	Technics Stores Inc.	CA	84600.00
175	Gift Depot Inc.	CT	84300.00
177	Osaka Souveniers Co.	Osaka	81200.00
339	Classic Gift Ideas, Inc	PA	81100.00
450	The Sharp Gifts Warehouse	CA	77600.00
181	Vitachrome Inc.	NY	76400.00
486	Motor Mint Distributors Inc.	PA	72600.00

2) Show the unique productline values containing the word cars at the end from products table.

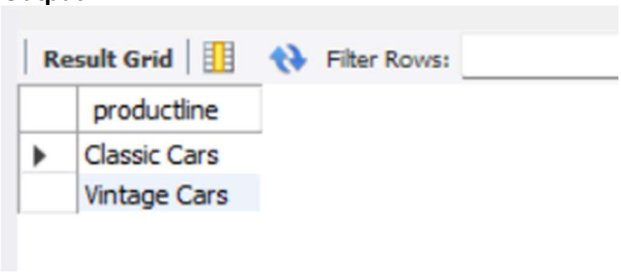
Expected output:

productLine
Classic Cars
Vintage Cars

Query :-

```
SELECT distinct productline FROM products where productLine like '%cars';
```

Output :-



1) Show the orderNumber, status and comments from orders table for shipped status only. If some comments are having null values then show them as "-".

Expected output:

orderNumber	status	Comments
10100	Shipped	-
10101	Shipped	Check on availability.
10102	Shipped	-
10103	Shipped	-
10104	Shipped	-
10105	Shipped	-
10106	Shipped	-
10107	Shipped	Difficult to negotiate with customer. We need more marketing materials
10108	Shipped	-
10109	Shipped	Customer requested that FedEx Ground is used for this shipping
10110	Shipped	-
10111	Shipped	-
10112	Shipped	Customer requested that ad materials (such as posters, pamphlets) be included in the shippment
10113	Shipped	-

Query :-

```
select orderNumber,status,coalesce(comments,"-")as comments from orders
where status="Shipped";
```

Output :-

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

orderNumber	status	comments
10100	Shipped	-
10101	Shipped	Check on availability.
10102	Shipped	-
10103	Shipped	-
10104	Shipped	-
10105	Shipped	-
10106	Shipped	-
10107	Shipped	Difficult to negotiate with customer. We need m...
10108	Shipped	-
10109	Shipped	Customer requested that FedEx Ground is used...
10110	Shipped	-
10111	Shipped	-
10112	Shipped	Customer requested that ad materials (such as ...
10113	Shipped	-
10114	Shipped	-
10115	Shipped	-
10116	Shipped	-

2) Select employee number, first name, job title and job title abbreviation from employees table based on following conditions.
If job title is one among the below conditions, then job title abbreviation column should show below forms.
   - President then “P”
   - Sales Manager / Sale Manager then “SM”
   - Sales Rep then “SR”
   - Containing VP word then “VP”

Expected output:

employeeNumber	firstName	jobTitle	jobTitle_abbr
1002	Diane	President	P
1102	Gerard	Sale Manager (EMEA)	SM
1088	William	Sales Manager (APAC)	SM
1143	Anthony	Sales Manager (NA)	SM
1165	Leslie	Sales Rep	SR
1166	Leslie	Sales Rep	SR
1188	Julie	Sales Rep	SR
1216	Steve	Sales Rep	SR
1286	Foon Yue	Sales Rep	SR
1323	George	Sales Rep	SR
1337	Loui	Sales Rep	SR
1370	Gerard	Sales Rep	SR
1401	Pamela	Sales Rep	SR
1501	Larry	Sales Rep	SR

Query :-

```
select employeeNumber,firstName,jobTitle,
case
when jobTitle="president" then 'P'
when (jobTitle like "sales Manager%" or jobTitle like 'sale Manager%')then 'SM'
when jobTitle='Sales Rep'then 'SR'
when jobTitle like '%VP%' then 'VP'
else " "
end as JobTitleAbbreviation
from employees
order by jobTitle;
```

Output :-

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

employeeNumber	firstName	jobTitle	JobTitleAbbreviation
1002	Diane	President	P
1102	Gerard	Sale Manager (EMEA)	SM
1088	William	Sales Manager (APAC)	SM
1143	Anthony	Sales Manager (NA)	SM
1165	Leslie	Sales Rep	SR
1166	Leslie	Sales Rep	SR
1188	Julie	Sales Rep	SR
1216	Steve	Sales Rep	SR
1286	Foon Yue	Sales Rep	SR
1323	George	Sales Rep	SR
1337	Loui	Sales Rep	SR
1370	Gerard	Sales Rep	SR
1401	Pamela	Sales Rep	SR
1501	Larry	Sales Rep	SR
1504	Barry	Sales Rep	SR
1611	Andy	Sales Rep	SR
1612	Peter	Sales Rep	SR
1619	Tom	Sales Rep	SR
1621	Mami	Sales Rep	SR
1625	Yoshimi	Sales Rep	SR
1702	Martin	Sales Rep	SR
1076	Jeff	VP Marketing	VP
1056	Mary	VP Sales	VP

1) For every year, find the minimum amount value from payments table.

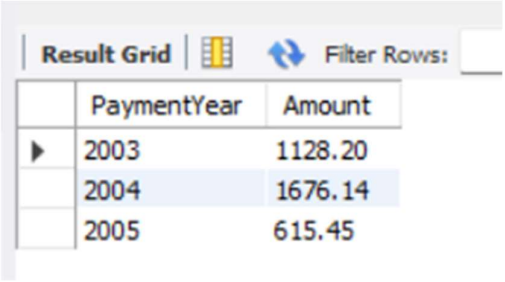
Expected output:

Year	Min Amount
2003	1128.20
2004	1676.14
2005	615.45

Query :-

```
select year(paymentdate) as PaymentYear,min(amount) as Amount from payments
group by paymentyear
order by paymentyear;
```

Output :-



	PaymentYear	Amount
▶	2003	1128.20
	2004	1676.14
	2005	615.45

2) For every year and every quarter, find the unique customers and total orders from orders table. Make sure to show the quarter as Q1,Q2 etc.

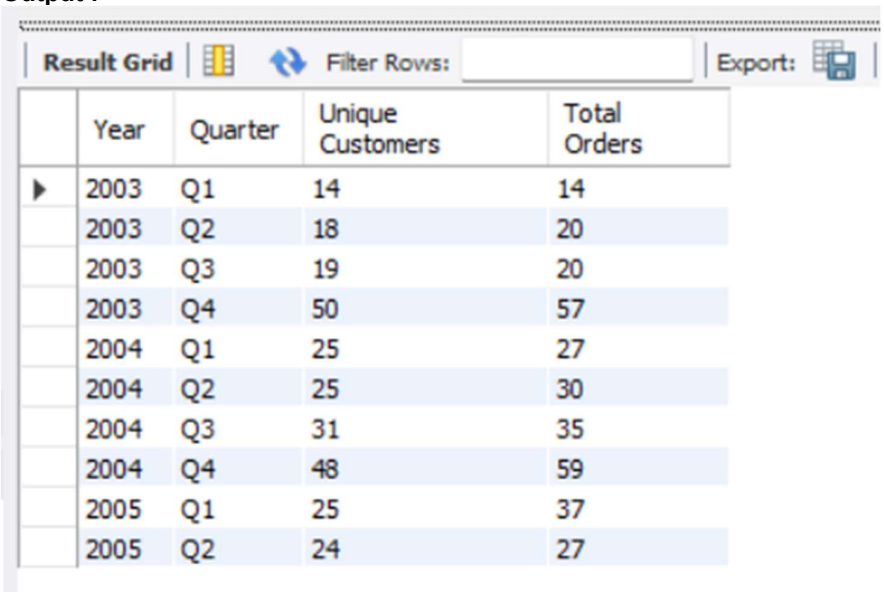
Expected output:

Year	Quarter	Unique Customers	Total Orders
2003	Q1	14	14
2003	Q2	18	20
2003	Q3	19	20
2003	Q4	50	57
2004	Q1	25	27
2004	Q2	25	30
2004	Q3	31	35
2004	Q4	48	59
2005	Q1	25	37
2005	Q2	24	27

Query :-

```
select
year(orderdate)as Year,
concat("Q",Quarter(orderdate)) as Quarter,
count(distinct customerNumber)as "Unique Customers",
count(orderNumber)as "Total Orders"
from orders
group by Year,Quarter
order by Year,Quarter;
```

Output :-



	Year	Quarter	Unique Customers	Total Orders
▶	2003	Q1	14	14
	2003	Q2	18	20
	2003	Q3	19	20
	2003	Q4	50	57
	2004	Q1	25	27
	2004	Q2	25	30
	2004	Q3	31	35
	2004	Q4	48	59
	2005	Q1	25	37
	2005	Q2	24	27

3) Show the formatted amount in thousands unit (e.g. 500K, 465K etc.) for every month (e.g. Jan, Feb etc.) with filter on total amount as 500000 to 1000000. Sort the output by total amount in descending mode. [Refer. Payments Table]

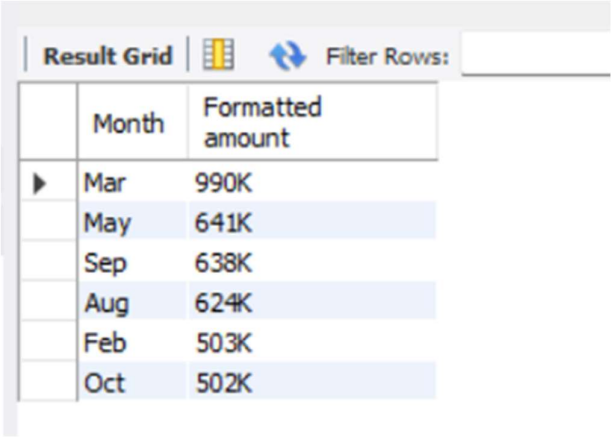
Expected output:

Month	formatted amount
Mar	990K
May	641K
Sep	638K
Aug	624K
Feb	503K
Oct	502K

Query :-

```
select
date_format(paymentDate,'%b') as  Month,
concat(format(sum(amount)/1000,0),"K") as "Formatted amount"
from payments
group by month
having sum(amount) between 500000 and 1000000
order by sum(amount) desc;
```

Output :-



Month	Formatted amount
Mar	990K
May	641K
Sep	638K
Aug	624K
Feb	503K
Oct	502K

1) Create a journey table with following fields and constraints.

- Bus_ID (No null values)
- Bus_Name (No null values)
- Source_Station (No null values)
- Destination (No null values)
- Email (must not contain any duplicates)

Query :-

```
create table journey(
        Bus_ID int not null,
        Bus_Name varchar(100) not null,
        Source_Station varchar(100) not null,
        Destination varchar(100) not null,
        Email varchar(100)unique
);
```

Output :-

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
Bus_ID	int	NO		NULL	
Bus_Name	varchar(100)	NO		NULL	
Source_Station	varchar(100)	NO		NULL	
Destination	varchar(100)	NO		NULL	
Email	varchar(100)	YES	UNI	NULL	

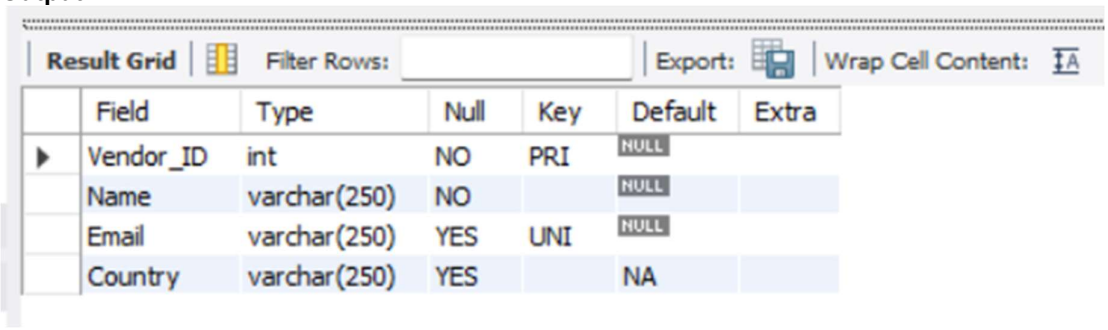
2) Create vendor table with following fields and constraints.

- Vendor_ID (Should not contain any duplicates and should not be null)
- Name (No null values)
- Email (must not contain any duplicates)
- Country (If no data is available then it should be shown as “N/A”)

Query :-

```
create table vendor(
   Vendor_ID int primary key,
   Name varchar(250)not null,
   Email varchar(250)unique,
   Country varchar(250)default "NA"
);
```

Output :-



Field	Type	Null	Key	Default	Extra
Vendor_ID	int	NO	PRI	NULL	
Name	varchar(250)	NO		NULL	
Email	varchar(250)	YES	UNI	NULL	
Country	varchar(250)	YES		NA	

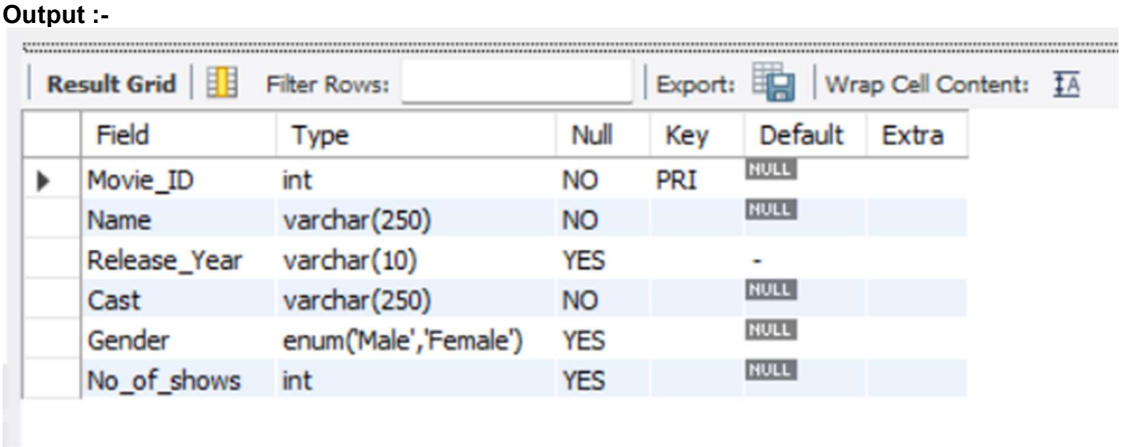
3) Create movies table with following fields and constraints.

- Movie_ID (Should not contain any duplicates and should not be null)
- Name (No null values)
- Release_Year (If no data is available then it should be shown as “-”)
- Cast (No null values)
- Gender (Either Male/Female)
- No_of_shows (Must be a positive number)

Query :-

```
create table movies(
   Movie_ID int primary key,
   Name varchar(250)not null,
   Release_Year varchar(10) default '-',
   Cast varchar(250)not null,
   Gender enum('Male','Female'),
   No_of_shows int check( No_of_shows >=0)
);
```

Output :-



Field	Type	Null	Key	Default	Extra
Movie_ID	int	NO	PRI	NULL	
Name	varchar(250)	NO		NULL	
Release_Year	varchar(10)	YES		-	
Cast	varchar(250)	NO		NULL	
Gender	enum('Male','Female')	YES		NULL	
No_of_shows	int	YES		NULL	

4) Create the following tables. Use auto increment wherever applicable

a. Product
- ✔ product_id - primary key
- ✔ product_name - cannot be null and only unique values are allowed
- ✔ description
- ✔ supplier_id - foreign key of supplier table

b. Suppliers
- ✔ supplier_id - primary key
- ✔ supplier_name
- ✔ location

c. Stock
- ✔ id - primary key
- ✔ product_id - foreign key of product table
- ✔ balance_stock

Query :-

```
create table Suppliers(
    supplier_id int auto_increment primary key,
    supplier_name varchar(250),
    location varchar(250)
);

create table Product(
    product_id int auto_increment primary key,
    product_name varchar(250) not null unique,
    description text,
    supplier_id int,
    foreign key (supplier_id) references Suppliers(supplier_id)
);

create table Stock(
    id int auto_increment primary key,
    product_id int,
    balance_stock int,
    foreign key(product_id) references Product(product_id)
);
```

Output :-

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
supplier_id	int	NO	PRI	NULL	auto_increment
supplier_name	varchar(250)	YES		NULL	
location	varchar(250)	YES		NULL	

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
product_id	int	NO	PRI	NULL	auto_increment
product_name	varchar(250)	NO	UNI	NULL	
description	text	YES		NULL	
supplier_id	int	YES	MUL	NULL	

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
product_id	int	YES	MUL	NULL	
balance_stock	int	YES		NULL	

1) Show employee number, Sales Person (combination of first and last names of employees), unique customers for each employee number and sort the data by highest to lowest unique customers.

   Tables: Employees, Customers

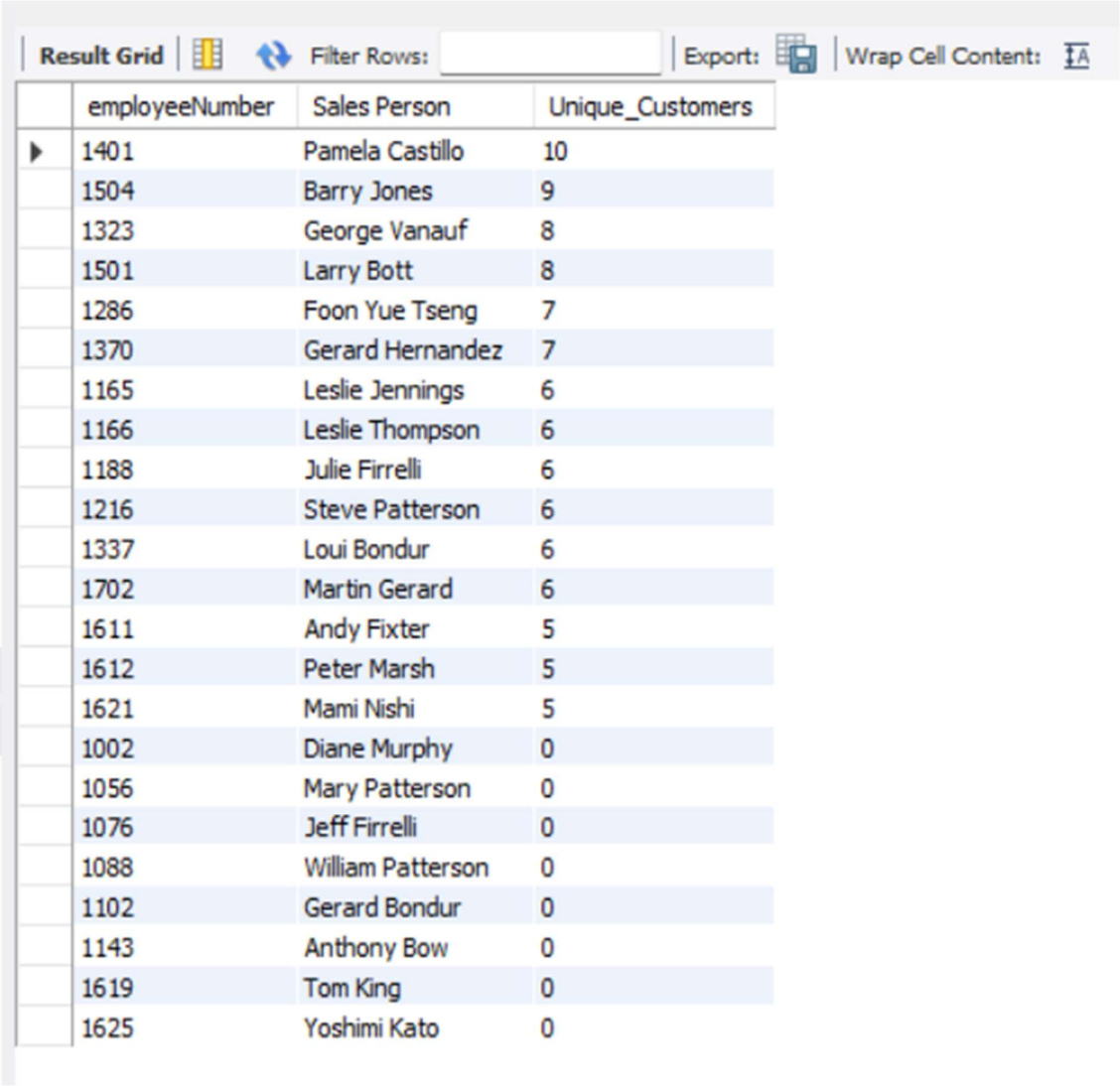
Expected output:

employeeNumber	Sales Person	Unique Customers
1401	Pamela Castillo	10
1504	Barry Jones	9
1323	George Vanauf	8
1501	Larry Bott	8
1286	Foon Yue Tseng	7
1370	Gerard Hernandez	7
1165	Leslie Jennings	6
1166	Leslie Thompson	6
1188	Julie Firrelli	6
1216	Steve Patterson	6
1337	Loui Bondur	6
1702	Martin Gerard	6
1611	Andy Fixter	5
1612	Peter Marsh	5

Query :-

```
select
employeeNumber,
concat(firstName," ",lastname)as "Sales Person",
count(distinct customerNumber)as Unique_Customers
from employees left join customers on
employeeNumber=salesRepEmployeeNumber
group by employeeNumber,"Sales Person"
order by Unique_Customers desc;
```

Output :-



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

employeeNumber	Sales Person	Unique_Customers
1401	Pamela Castillo	10
1504	Barry Jones	9
1323	George Vanauf	8
1501	Larry Bott	8
1286	Foon Yue Tseng	7
1370	Gerard Hernandez	7
1165	Leslie Jennings	6
1166	Leslie Thompson	6
1188	Julie Firrelli	6
1216	Steve Patterson	6
1337	Loui Bondur	6
1702	Martin Gerard	6
1611	Andy Fixter	5
1612	Peter Marsh	5
1621	Mami Nishi	5
1002	Diane Murphy	0
1056	Mary Patterson	0
1076	Jeff Firrelli	0
1088	William Patterson	0
1102	Gerard Bondur	0
1143	Anthony Bow	0
1619	Tom King	0
1625	Yoshimi Kato	0

2) Show total quantities, total quantities in stock, left over quantities for each product and each customer. Sort the data by customer number.

Tables: Customers, Orders, Orderdetails, Products

Expected output:

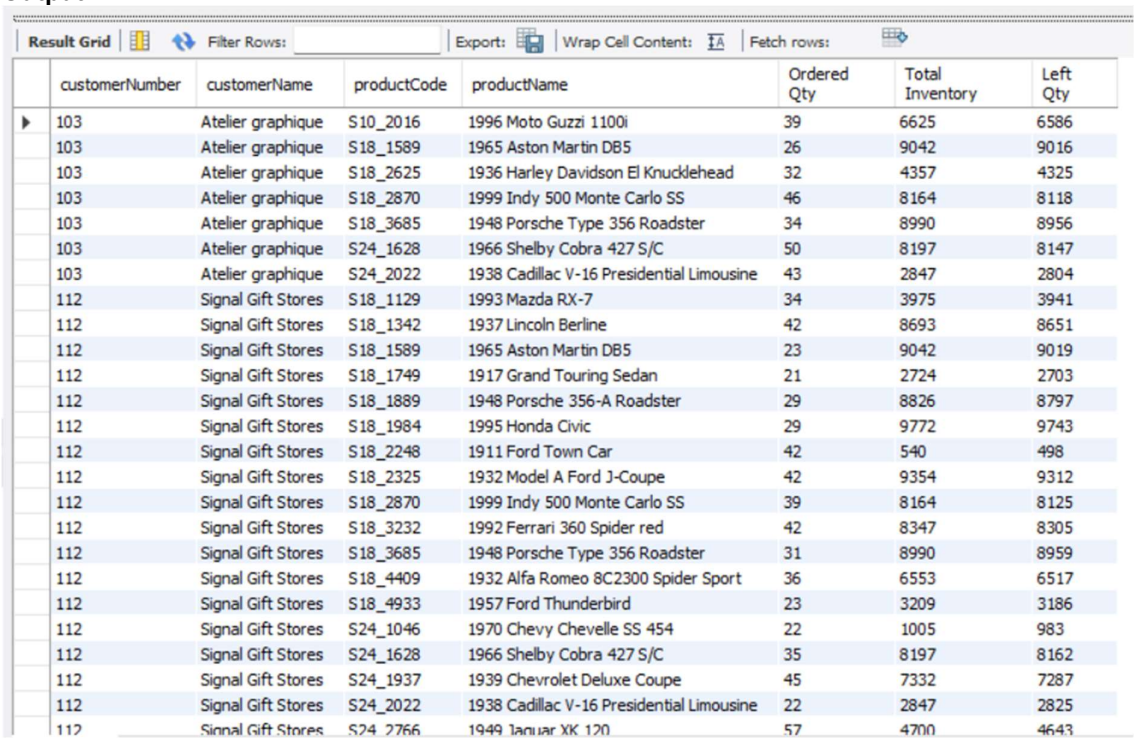
customerNumber	customerName	productCode	productName	Ordered Qty	Total Inventory	Left Qty
103	Atelier graphique	S10_2016	1996 Moto Guzzi 1100i	39	6625	6586
103	Atelier graphique	S18_1589	1965 Aston Martin DB5	26	9042	9016
103	Atelier graphique	S18_2625	1936 Harley Davidson El Knucklehead	32	4357	4325
103	Atelier graphique	S18_2870	1999 Indy 500 Monte Carlo SS	46	8164	8118
103	Atelier graphique	S18_3685	1948 Porsche Type 356 Roadster	34	8990	8956
103	Atelier graphique	S24_1628	1966 Shelby Cobra 427 S/C	50	8197	8147
103	Atelier graphique	S24_2022	1938 Cadillac V-16 Presidential Limousine	43	2847	2804
112	Signal Gift Stores	S18_1129	1993 Mazda RX-7	34	3975	3941
112	Signal Gift Stores	S18_1342	1937 Lincoln Berline	42	8693	8651
112	Signal Gift Stores	S18_1589	1965 Aston Martin DB5	23	9042	9019
112	Signal Gift Stores	S18_1749	1917 Grand Touring Sedan	21	2724	2703
112	Signal Gift Stores	S18_1889	1948 Porsche 356-A Roadster	29	8826	8797
112	Signal Gift Stores	S18_1984	1995 Honda Civic	29	9772	9743
112	Signal Gift Stores	S18_2248	1911 Ford Town Car	42	540	498

Query :-

```
select
    customers.customerNumber,
    customers.customerName,
    products.productCode,
    products.productName,
    sum(orderdetails.quantityOrdered) as "Ordered Qty",
    sum(products.quantityInStock)as "Total Inventory",
    sum(products.quantityInStock - orderdetails.quantityOrdered)as "Left Qty"
from
        customers
inner join
        orders
on
        customers.customerNumber = orders.customerNumber
inner join
        orderdetails
on
        orders.orderNumber = orderdetails.orderNumber
inner join
        products
```

```
on
        orderdetails.productCode = products.productCode
group by
        customers.customerNumber,
        products.productCode
order by
        customers.customerNumber asc;
```

Output :-



Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

customerNumber	customerName	productCode	productName	Ordered Qty	Total Inventory	Left Qty
103	Atelier graphique	S10_2016	1996 Moto Guzzi 1100i	39	6625	6586
103	Atelier graphique	S18_1589	1965 Aston Martin DB5	26	9042	9016
103	Atelier graphique	S18_2625	1936 Harley Davidson El Knucklehead	32	4357	4325
103	Atelier graphique	S18_2870	1999 Indy 500 Monte Carlo SS	46	8164	8118
103	Atelier graphique	S18_3685	1948 Porsche Type 356 Roadster	34	8990	8956
103	Atelier graphique	S24_1628	1966 Shelby Cobra 427 S/C	50	8197	8147
103	Atelier graphique	S24_2022	1938 Cadillac V-16 Presidential Limousine	43	2847	2804
112	Signal Gift Stores	S18_1129	1993 Mazda RX-7	34	3975	3941
112	Signal Gift Stores	S18_1342	1937 Lincoln Berline	42	8693	8651
112	Signal Gift Stores	S18_1589	1965 Aston Martin DB5	23	9042	9019
112	Signal Gift Stores	S18_1749	1917 Grand Touring Sedan	21	2724	2703
112	Signal Gift Stores	S18_1889	1948 Porsche 356-A Roadster	29	8826	8797
112	Signal Gift Stores	S18_1984	1995 Honda Civic	29	9772	9743
112	Signal Gift Stores	S18_2248	1911 Ford Town Car	42	540	498
112	Signal Gift Stores	S18_2325	1932 Model A Ford J-Coupe	42	9354	9312
112	Signal Gift Stores	S18_2870	1999 Indy 500 Monte Carlo SS	39	8164	8125
112	Signal Gift Stores	S18_3232	1992 Ferrari 360 Spider red	42	8347	8305
112	Signal Gift Stores	S18_3685	1948 Porsche Type 356 Roadster	31	8990	8959
112	Signal Gift Stores	S18_4409	1932 Alfa Romeo 8C2300 Spider Sport	36	6553	6517
112	Signal Gift Stores	S18_4933	1957 Ford Thunderbird	23	3209	3186
112	Signal Gift Stores	S24_1046	1970 Chevy Chevelle SS 454	22	1005	983
112	Signal Gift Stores	S24_1628	1966 Shelby Cobra 427 S/C	35	8197	8162
112	Signal Gift Stores	S24_1937	1939 Chevrolet Deluxe Coupe	45	7332	7287
112	Signal Gift Stores	S24_2022	1938 Cadillac V-16 Presidential Limousine	22	2847	2825
112	Signal Gift Stores	S24_2766	1949 Jaguar XK 120	57	4700	4643

3) Create below tables and fields. (You can add the data as per your wish)

- Laptop: (Laptop_Name)
- Colours: (Colour_Name)

Perform cross join between the two tables and find number of rows.

Expected output:

Laptop_Name	Colour_Name
Dell	White
Dell	Silver
Dell	Black
HP	White
HP	Silver
HP	Black

Query :-

```
create table Laptop(Laptop_Name varchar(250));
insert into Laptop(Laptop_Name) values
("Dell"),
("HP"),
("Lenovo");
select*from Laptop;

create table Colours(Colour_Name varchar(250));
insert into Colours(Colour_Name) values
("Black"),
("Silver"),
("White");
select*from Colours;

select Laptop.Laptop_Name, Colours.colour_Name
from Laptop
cross join Colours
order by Laptop_Name;
```

Output :-

Result Grid | Filter Rows:

	Laptop_Name	colour_Name
▶	Dell	Black
	Dell	Silver
	Dell	White
	HP	Black
	HP	Silver
	HP	White
	Lenovo	Black
	Lenovo	Silver
	Lenovo	White

4) Create table project with below fields.

- EmployeeID
- FullName
- Gender
- ManagerID

Add below data into it.

```
INSERT INTO Project VALUES(1, 'Pranaya', 'Male', 3);
INSERT INTO Project VALUES(2, 'Priyanka', 'Female', 1);
INSERT INTO Project VALUES(3, 'Preety', 'Female', NULL);
INSERT INTO Project VALUES(4, 'Anurag', 'Male', 1);
INSERT INTO Project VALUES(5, 'Sambit', 'Male', 1);
INSERT INTO Project VALUES(6, 'Rajesh', 'Male', 3);
INSERT INTO Project VALUES(7, 'Hina', 'Female', 3);
```

Find out the names of employees and their related managers.

Expected output:

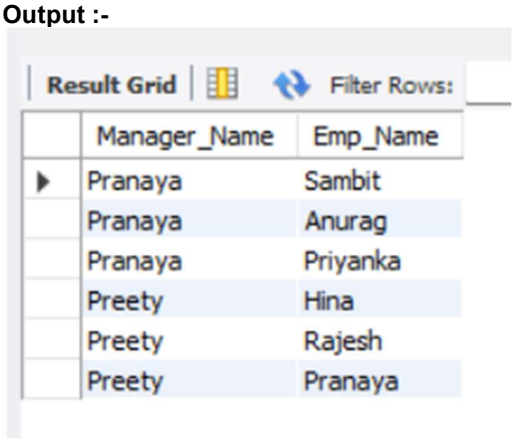
Manager Name	Emp Name
Pranaya	Priyanka
Pranaya	Anurag
Pranaya	Sambit
Preety	Pranaya
Preety	Rajesh
Preety	Hina

Query :-

```
create table project(
        EmployeeID int primary key,
        FullName varchar(250),
        Gender varchar(25),
        ManagerID int
);

INSERT INTO Project VALUES(1, 'Pranaya', 'Male', 3);
INSERT INTO Project VALUES(2, 'Priyanka', 'Female', 1);
```

```
INSERT INTO Project VALUES(3, 'Preety', 'Female', NULL);
INSERT INTO Project VALUES(4, 'Anurag', 'Male', 1);
INSERT INTO Project VALUES(5, 'Sambit', 'Male', 1);
INSERT INTO Project VALUES(6, 'Rajesh', 'Male', 3);
INSERT INTO Project VALUES(7, 'Hina', 'Female', 3);

select*from project;

select mgr.FullName as Manager_Name, emp.FullName as Emp_Name
from project mgr
inner join project emp
on
mgr.EmployeeID=emp.ManagerID
order by Manager_Name;
```

Output :-



Manager_Name	Emp_Name
Pranaya	Sambit
Pranaya	Anurag
Pranaya	Priyanka
Preety	Hina
Preety	Rajesh
Preety	Pranaya

1) Create table facility. Add the below fields into it.

- Facility_ID
- Name
- State
- Country

i) Alter the table by adding the primary key and auto increment to Facility_ID column.

ii) Add a new column city after name with data type as varchar which should not accept any null values.

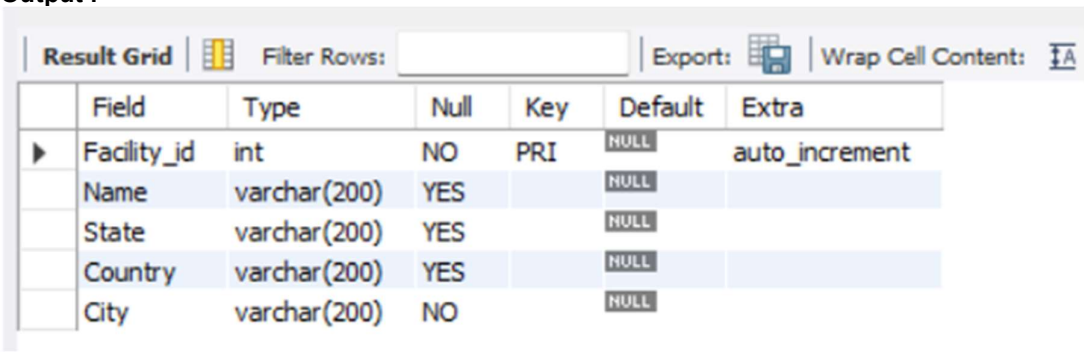
Expected output:

Field	Type	Null	Key	Default	Extra
Facility_ID	int	NO	PRI	NULL	auto_increment
Name	varchar(100)	YES		NULL	
City	varchar(100)	NO		NULL	
State	varchar(100)	YES		NULL	
Country	varchar(100)	YES		NULL	

Query :-

```
create table facility(
        Facility_ID int primary key,
        Name varchar(200),
        State varchar(200),
        Country varchar(200)
);

select* from facility;

alter table facility Modify Facility_id int auto_increment;

alter table facility add City varchar(200) not null;

desc facility;
```

Output :-



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	Field	Type	Null	Key	Default	Extra
▶	Facility_id	int	NO	PRI	NULL	auto_increment
	Name	varchar(200)	YES		NULL	
	State	varchar(200)	YES		NULL	
	Country	varchar(200)	YES		NULL	
	City	varchar(200)	NO		NULL	

1) Create table university with below fields.

- ID
- Name

Add the below data into it as it is.

INSERT INTO University

VALUES (1, " Pune University "),

(2, " Mumbai University "),

(3, " Delhi University "),

(4, "Madras University"),

(5, "Nagpur University");

Remove the spaces from everywhere and update the column like Pune University etc.

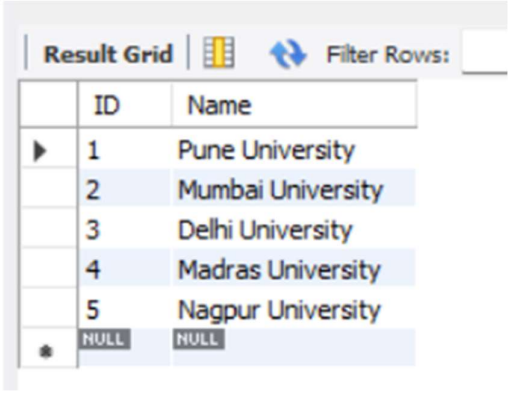
Expected output:

ID	Name
1	Pune University
2	Mumbai University
3	Delhi University
4	Madras University
5	Nagpur University
NULL	NULL

Query :-

```
create table university(
        ID int primary key,
        Name varchar(200)
);

INSERT INTO University
VALUES  (1, "    Pune        University    "),
                (2, "  Mumbai        University    "),
                (3, "    Delhi   University    "),
                (4, "Madras University"),
                (5, "Nagpur University");

select*from university;

SET SQL_SAFE_UPDATES=0;
update university set name=trim(both " " from REGEXP_REPLACE(name, ' {1,}',' '))
where id is not null;
```

Output :-



1) Create the view products status. Show year wise total products sold. Also find the percentage of total value for each year. The output should look as shown in below figure.

Expected output:

Year	Value
2004	1421 (47%)
2003	1052 (35%)
2005	523 (17%)

Query :-

```
create view products_status as

select

year(o.orderdate)  as Year,

concat(

round(count(od.quantityordered*od.priceEach)),

'(',

round((sum(od.quantityordered*od.priceEach)/ sum(sum(od.quantityordered*od.priceEach)) over())
*100),

'%)'

)as "Value"

from orders o

join

orderdetails od

on o.ordernumber=od.ordernumber

group by year(o.orderdate);


select * from products_status;
```

Output :-

Result Grid | Filter Rows:

	Year	Value
▶	2003	1052(35%)
	2004	1421(47%)
	2005	523(18%)

1) Create a stored procedure GetCustomerLevel which takes input as customer number and gives the output as either Platinum, Gold or Silver as per below criteria.

   Table: Customers

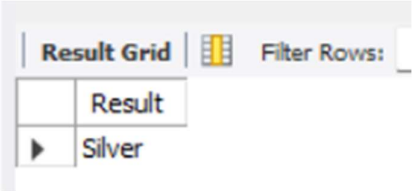
   - Platinum: creditLimit > 100000
   - Gold: creditLimit is between 25000 to 100000
   - Silver: creditLimit < 25000

Query :-

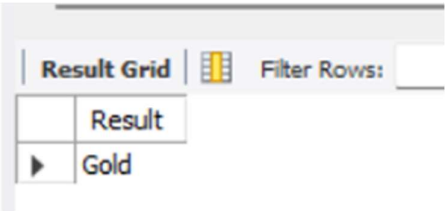
```
DELIMITER //
create procedure `GetCustomerLevel` (in CUST int)
begin
	declare lct_creditlimit integer;
	select creditLimit into lct_creditlimit from customers where customerNumber=CUST;
	case
		when lct_creditlimit > 100000 then
			select "Platinum" as Result;
		when lct_creditlimit between 25000 and 100000 then
			select "Gold" as Result;
		when lct_creditlimit < 25000 then
			select "Silver" as Result;
		else
			select "Out of range" as Result;
	end case;
end //
DELIMITER ;

select*from customers;
```

Output :-

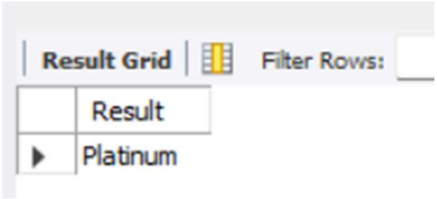
call assignment.GetCustomerLevel(103);



call assignment.GetCustomerLevel(450);



call assignment.GetCustomerLevel(496);



2) Create a stored procedure Get_country_payments which takes in year and country as inputs and gives year wise, country wise total amount as an output. Format the total amount to nearest thousand unit (K)

Tables: Customers, Payments

Expected output:

Year	country	Total Amount
2003	France	283K

Query :-

```
DELIMITER //
create procedure `Get_country_payments` (in input_year int,input_country varchar(250))
begin
	select
		year(paymentDate)as Year,
		country as Country,
		concat(format(sum(amount)/1000,0), 'K')as 'Total Amount'
	from payments
	inner join customers on payments.customerNumber=customers.customerNumber
	where year(paymentDate)=input_year and country=input_country
	group by Year,Country;
end //
DELIMITER ;

call assignment.Get_country_payments(2003,'france');
```

Output :-

Result Grid | Filter Rows:

	Year	Country	Total Amount
▶	2003	France	283K

1) Calculate year wise, month name wise count of orders and year over year (YoY) percentage change. Format the YoY values in no decimals and show in % sign.

Table: Orders

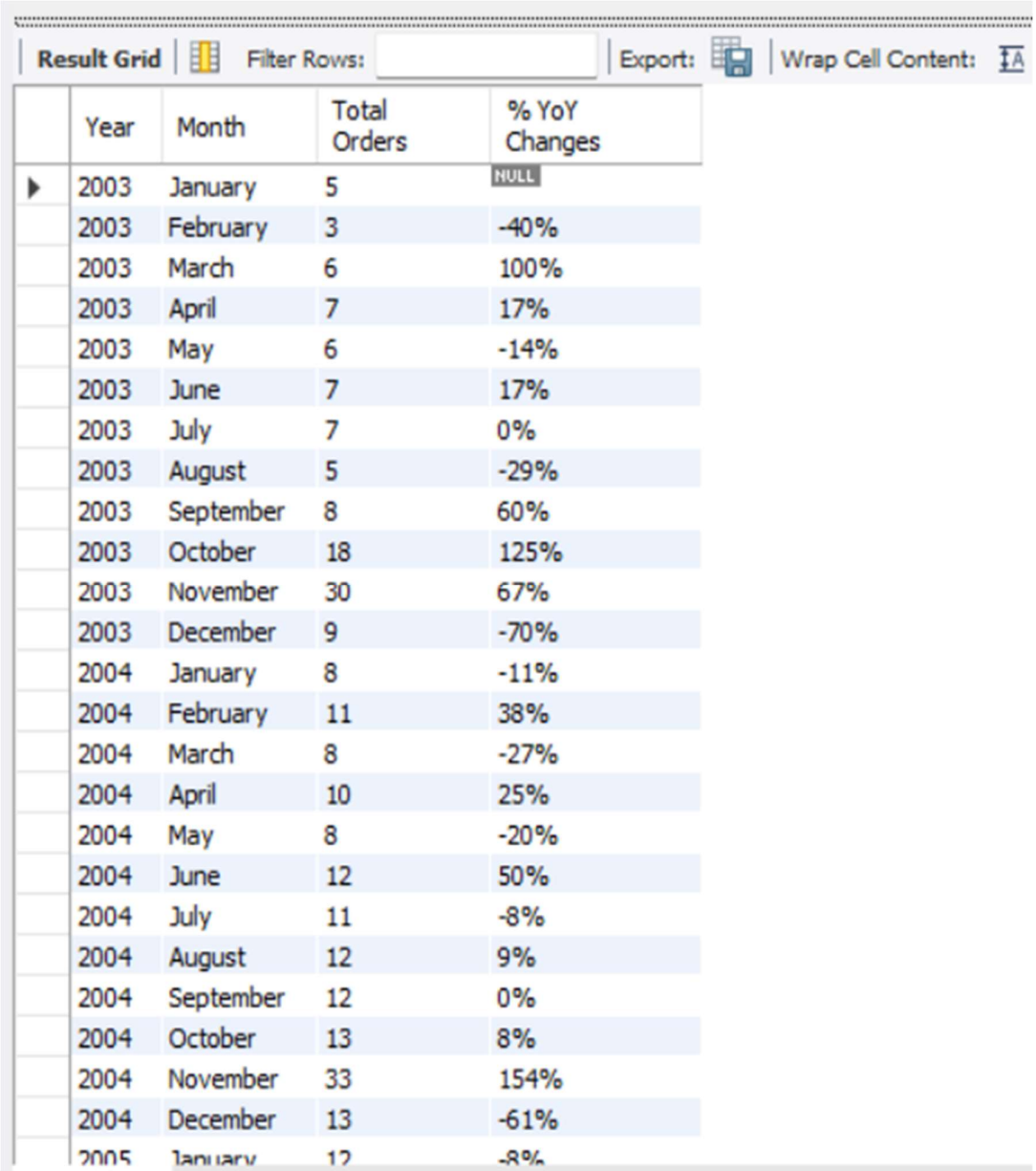
Expected output:

Year	Month	Total Orders	% YoY Change
2003	January	5	NULL
2003	February	3	-40%
2003	March	6	100%
2003	April	7	17%
2003	May	6	-14%
2003	June	7	17%
2003	July	7	0%
2003	August	5	-29%
2003	September	8	60%
2003	October	18	125%
2003	November	30	67%
2003	December	9	-70%
2004	January	8	-11%
2004	February	11	38%

Query :-

```
with X as (
select
        year(orderdate) as Year,
        Monthname(orderdate) as Month,
        count(orderdate)as Total_Orders
        from
                orders
                group by
                        Year,Month
)
select
        Year,
        Month,
        Total_Orders as 'Total Orders',
        concat(round(100*(( Total_Orders - LAG(Total_orders) over (order by year)) /
        LAG(Total_orders) over (order by year)),0),'%') as "% YoY Changes" from X;
```

Output :-



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Year	Month	Total Orders	% YoY Changes
2003	January	5	NULL
2003	February	3	-40%
2003	March	6	100%
2003	April	7	17%
2003	May	6	-14%
2003	June	7	17%
2003	July	7	0%
2003	August	5	-29%
2003	September	8	60%
2003	October	18	125%
2003	November	30	67%
2003	December	9	-70%
2004	January	8	-11%
2004	February	11	38%
2004	March	8	-27%
2004	April	10	25%
2004	May	8	-20%
2004	June	12	50%
2004	July	11	-8%
2004	August	12	9%
2004	September	12	0%
2004	October	13	8%
2004	November	33	154%
2004	December	13	-61%
2005	January	12	-8%

2) Create the table emp_udf with below fields.

- Emp_ID
- Name
- DOB

Add the data as shown in below query.

INSERT INTO Emp_UDF(Name, DOB)

VALUES ("Piyush", "1990-03-30"), ("Aman", "1992-08-15"), ("Meena", "1998-07-28"), ("Ketan", "2000-11-21"), ("Sanjay", "1995-05-21");

Create a user defined function calculate_age which returns the age in years and months (e.g. 30 years 5 months) by accepting DOB column as a parameter.

Expected output:

Emp_ID	Name	DOB	Age
1	Piyush	1990-03-30	33 years 0 months
2	Aman	1992-08-15	30 years 8 months
3	Meena	1998-07-28	24 years 8 months
4	Ketan	2000-11-21	22 years 5 months
5	Sanjay	1995-05-21	27 years 11 months

Query :-

```
create table emp_udf (
        Emp_ID int auto_increment primary key,
        Name varchar(250),
        DOB date
);


INSERT INTO Emp_UDF(Name, DOB)
VALUES ("Piyush", "1990-03-30"), ("Aman", "1992-08-15"), ("Meena", "1998-07-28"), ("Ketan", "2000-11-21"), ("Sanjay", "1995-05-21");


select*from Emp_UDF;
```

```
DELIMITER //
create function calculate_age(date_of_birth date)returns varchar(50)
deterministic
begin
        declare years int;
    declare months int;
    declare age varchar(50);

    set years=timestampdiff(year,date_of_birth,curdate());
    set months=timestampdiff(month,date_of_birth,curdate())%12;

    set age=concat(years,' years',months, 'months');

    return age;
end //

DELIMITER ;

select
emp_id,
name,
dob,
calculate_age(dob) as age
from emp_udf;
```

Output :-

Result Grid | Filter Rows: | Export:

emp_id	name	dob	age
1	Piyush	1990-03-30	34 years0months
2	Aman	1992-08-15	31 years7months
3	Meena	1998-07-28	25 years8months
4	Ketan	2000-11-21	23 years4months
5	Sanjay	1995-05-21	28 years10months

1) Display the customer numbers and customer names from customers table who have not placed any orders using subquery

Table: Customers, Orders

Expected output:

customerNumber	customerName
125	Havel & Zbyszek Co
168	American Souvenirs Inc
169	Porto Imports Co.
206	Asian Shopping Network, Co
223	Natürlich Autos
237	ANG Resellers
247	Messner Shopping Network
273	Franken Gifts, Co
293	BG&E Collectables
303	Schuyler Imports
307	Der Hund Imports
335	Cramer Spezialitäten, Ltd
348	Asian Treasures, Inc.
356	SAR Distributors, Co

Query :-

```
select customerNumber,customerName
from customers
where customerName Not in (select customerNumber from orders);
```

Output :-

Result Grid | Filter Rows:

customerNumber	customerName
103	Atelier graphique
112	Signal Gift Stores
114	Australian Collectors, Co.
119	La Rochelle Gifts
121	Baane Mini Imports
124	Mini Gifts Distributors Ltd.
125	Havel & Zbyszek Co
128	Blauer See Auto, Co.
129	Mini Wheels Co.
131	Land of Toys Inc.
141	Euro+ Shopping Channel
144	Volvo Model Replicas, Co
145	Danish Wholesale Imports
146	Saveley & Henriot, Co.
148	Dragon Souveniers, Ltd.
151	Muscle Machine Inc
157	Diecast Classics Inc.
161	Technics Stores Inc.
166	Handji Gifts& Co
167	Herkku Gifts
168	American Souvenirs Inc
169	Porto Imports Co.
171	Daedalus Designs Imports
172	La Corne D'abondance, Co.
173	Cambridge Collectables Co.

2) Write a full outer join between customers and orders using union and get the customer number, customer name, count of orders for every customer.

Table: Customers, Orders

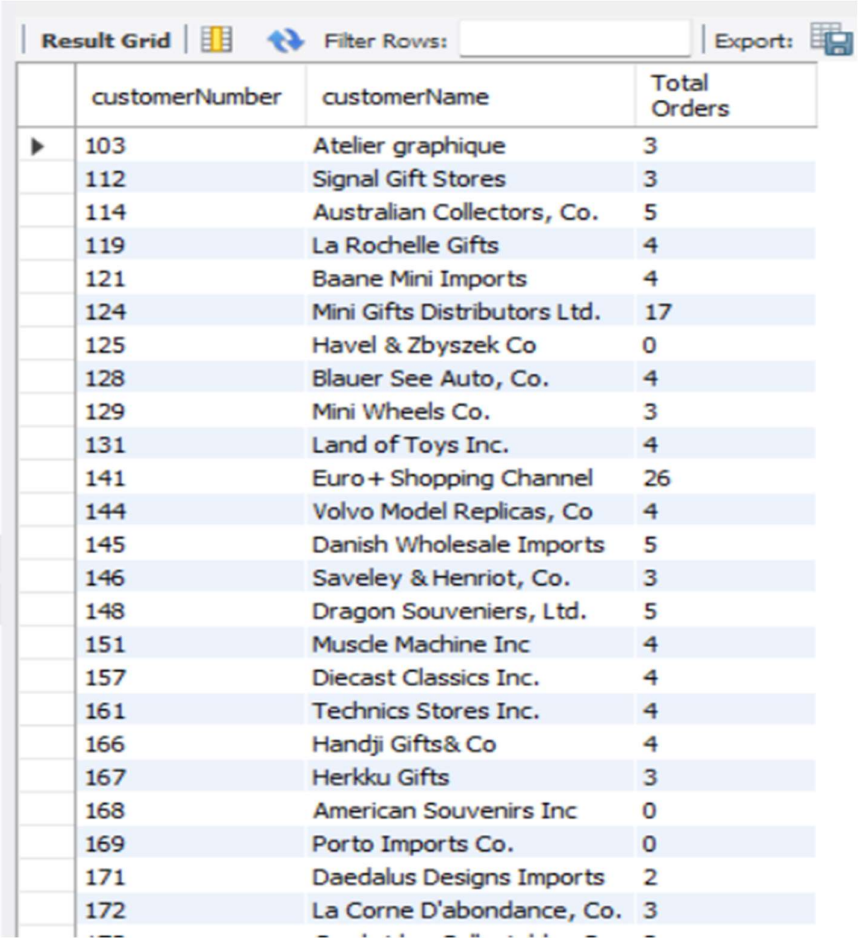
Expected output:

customerNumber	customerName	Total Orders
103	Atelier graphique	3
112	Signal Gift Stores	3
114	Australian Collectors, Co.	5
119	La Rochelle Gifts	4
121	Baane Mini Imports	4
124	Mini Gifts Distributors Ltd.	17
125	Havel & Zbyszek Co	0
128	Blauer See Auto, Co.	4
129	Mini Wheels Co.	3
131	Land of Toys Inc.	4
141	Euro+ Shopping Channel	26
144	Volvo Model Replicas, Co	4
145	Danish Wholesale Imports	5
146	Saveley & Henriot, Co.	3

Query :-

```
select c.customerNumber,c.customerName,count(o.ordernumber) as 'Total Orders'
from customers c
left join orders o on c.customerNumber=o.customerNumber
group by c.customerNumber,c.customerName
union
select c.customerNumber,c.customerName,0 as 'Total Orders'
from customers c
where c.customerNumber not in (select distinct customerNumber from orders)
union
select o.customerNumber,null as customerName,count(o.ordernumber) as 'Total Orders'
from orders o
where o.customerNumber not in (select distinct customerNumber from customers)
group by o.customerNumber;
```

Output :-



customerNumber	customerName	Total Orders
103	Atelier graphique	3
112	Signal Gift Stores	3
114	Australian Collectors, Co.	5
119	La Rochelle Gifts	4
121	Baane Mini Imports	4
124	Mini Gifts Distributors Ltd.	17
125	Havel & Zbyszek Co	0
128	Blauer See Auto, Co.	4
129	Mini Wheels Co.	3
131	Land of Toys Inc.	4
141	Euro+ Shopping Channel	26
144	Volvo Model Replicas, Co	4
145	Danish Wholesale Imports	5
146	Saveley & Henriot, Co.	3
148	Dragon Souveniers, Ltd.	5
151	Muscle Machine Inc	4
157	Diecast Classics Inc.	4
161	Technics Stores Inc.	4
166	Handji Gifts& Co	4
167	Herkku Gifts	3
168	American Souvenirs Inc	0
169	Porto Imports Co.	0
171	Daedalus Designs Imports	2
172	La Corne D'abondance, Co.	3

3) Show the second highest quantity ordered value for each order number.

Table: Orderdetails

Expected output:

orderNumber	quantityOrdered
10100	49
10101	45
10102	39
10103	45
10104	44
10105	44
10106	49
10107	38
10108	44
10109	46
10110	46
10111	43
10112	23
10113	49

Query :-

```
select ordernumber,max(quantityordered) as QunatityOrders
from orderdetails o
where quantityordered < (
        select max(quantityordered)
        from orderdetails od
        where od.orderNumber=o.orderNumber
)
group by orderNumber;
```

Output :-

Result Grid | Filter Rows:

ordernumber	QunatityOrders
10100	49
10101	45
10102	39
10103	45
10104	44
10105	44
10106	49
10107	38
10108	44
10109	46
10110	46
10111	43
10112	23
10113	49
10114	45
10115	46
10117	45
10119	43
10120	47
10121	44
10122	42
10123	46
10124	46
10125	32
10126	46

4) For each order number count the number of products and then find the min and max of the values among count of orders.

Table: Orderdetails

Expected output:

MAX(Total)	MIN(Total)
18	1

Query :-

```
select
        ordernumber,
        count(ordernumber) as TotalProduct
from orderdetails
group by ordernumber
having count(orderNumber) > 0;
select
        max(TotalProduct) as 'Max(Total)',
        min(TotalProduct) as 'Min(Total)'
        from(
            select
                ordernumber,
                count(ordernumber) as TotalProduct
            from orderdetails
            group by ordernumber
            having count(orderNumber) > 0
        )as ProductCounts;
```

Output :-

Result Grid | Filter Rows:

	Max(Total)	Min(Total)
▶	18	1

5) Find out how many product lines are there for which the buy price value is greater than the average of buy price value. Show the output as product line and its count.

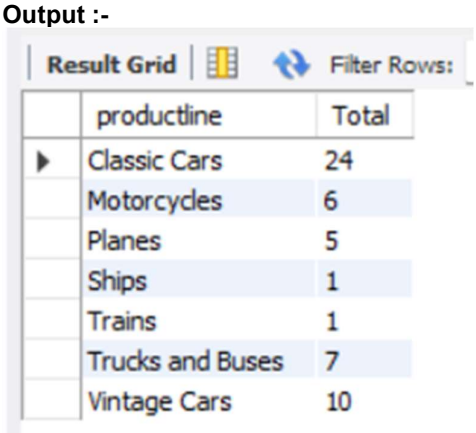
Expected output:

productLine	Total
Classic Cars	25
Vintage Cars	8
Trucks and Buses	7
Motorcycles	5
Planes	3
Ships	1
Trains	1

Query :-

```
select productline,count(*) as Total
from products
where buyprice > (
        select avg(buyprice)
        from products
)
group by productline;
```

Output :-



productline	Total
Classic Cars	24
Motorcycles	6
Planes	5
Ships	1
Trains	1
Trucks and Buses	7
Vintage Cars	10

1) Create the table Emp_EH. Below are its fields.

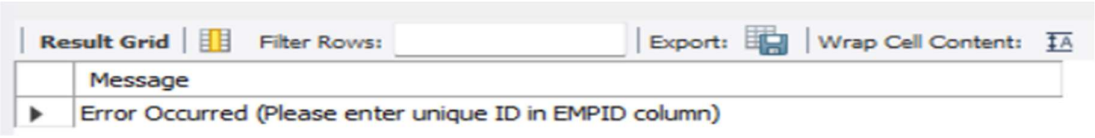
- EmpID (Primary Key)
- EmpName
- EmailAddress

Create a procedure to accept the values for the columns in Emp_EH. Handle the error using exception handling concept. Show the message as “Error occurred” in case of anything wrong.

Query :-

```
create table Emp_EH(
        EmpID int primary key,
        EmpName varchar(250),
        EmailAddress varchar(250)
);


DELIMITER //
CREATE PROCEDURE `InsertTableRecord`(in ID int, EmpName varchar(250), EmailAddress varchar(250))
BEGIN
Declare DuplicateEntry condition for 1062;


declare exit handler for DuplicateEntry
select 'Error Occurred (Please enter unique ID in EMPID column)' Message;


insert into Emp_EH(EmpID,EmpName,EmailAddress) values (ID,EmpName,EmailAddress);
select*from EMP_EH;
END //
DELIMITER ;

select*from EMP_EH;
call assignment.InsertTableRecord(106, 'ram', 'ram@gmail.com');
```

Output :-



1) Create the table Emp_BIT. Add below fields in it.

- Name
- Occupation
- Working_date
- Working_hours

Insert the data as shown in below query.

```
INSERT INTO Emp_BIT VALUES
('Robin', 'Scientist', '2020-10-04', 12),
('Warner', 'Engineer', '2020-10-04', 10),
('Peter', 'Actor', '2020-10-04', 13),
('Marco', 'Doctor', '2020-10-04', 14),
('Brayden', 'Teacher', '2020-10-04', 12),
('Antonio', 'Business', '2020-10-04', 11);
```

Create before insert trigger to make sure any new value of Working_hours, if it is negative, then it should be inserted as positive.

Query :-

```
create table Emp_BIT(
        Name varchar(250),
        Occupation varchar(250),
        Working_date varchar(250),
        Working_hours int
);

INSERT INTO Emp_BIT VALUES
('Robin', 'Scientist', '2020-10-04', 12),
('Warner', 'Engineer', '2020-10-04', 10),
('Peter', 'Actor', '2020-10-04', 13),
('Marco', 'Doctor', '2020-10-04', 14),
('Brayden', 'Teacher', '2020-10-04', 12),
('Antonio', 'Business', '2020-10-04', 11);
```

INSERT INTO Emp_BIT VALUES

('Lokesh', 'Scientist', '2020-10-04', -12),

('Mahesh', 'Teacher', '2020-10-04', -10);

select*from Emp_BIT;

Output :-

INSERT INTO Emp_BIT VALUES

('Lokesh', 'Scientist', '2020-10-04', -12),

('Mahesh', 'Teacher', '2020-10-04', -10);

select*from Emp_BIT;

Result Grid | Filter Rows: | Export:

Name	Occupation	Working_date	Working_hours
Robin	Scientist	2020-10-04	12
Warner	Engineer	2020-10-04	10
Peter	Actor	2020-10-04	13
Marco	Doctor	2020-10-04	14
Brayden	Teacher	2020-10-04	12
Antonio	Business	2020-10-04	11
Lokesh	Scientist	2020-10-04	12
Mahesh	Teacher	2020-10-04	10

THANK YOU