Day wise Assignment Solution – By Ninad S. Mandavkar

Note: -

1. The tables that are mentioned in the questions for the references are available in the classic model database.

2. In the questions, if they specifically mention to create the tables, then you need to create the tables as per given specifications.

Day 1

No questions

Day 2

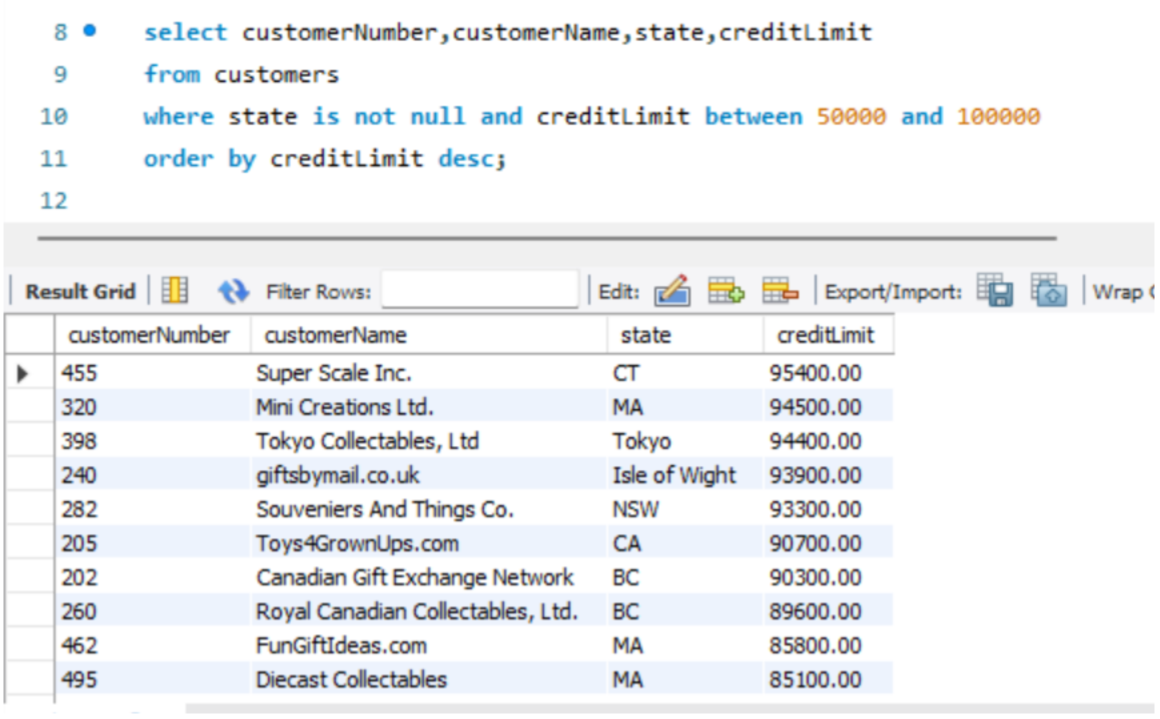
No questions

Day 3

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

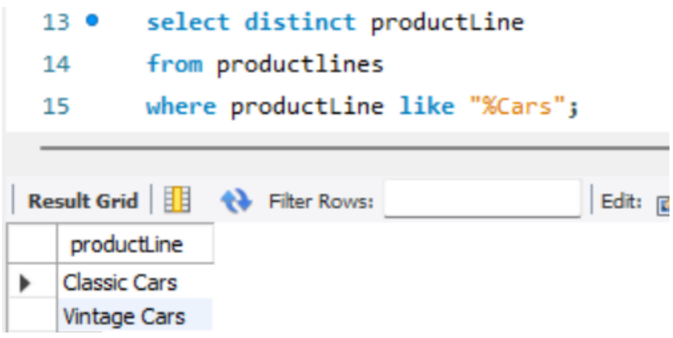
Solution:



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

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

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



```
select distinct productLine
from productlines
where productLine like "%Cars";
```

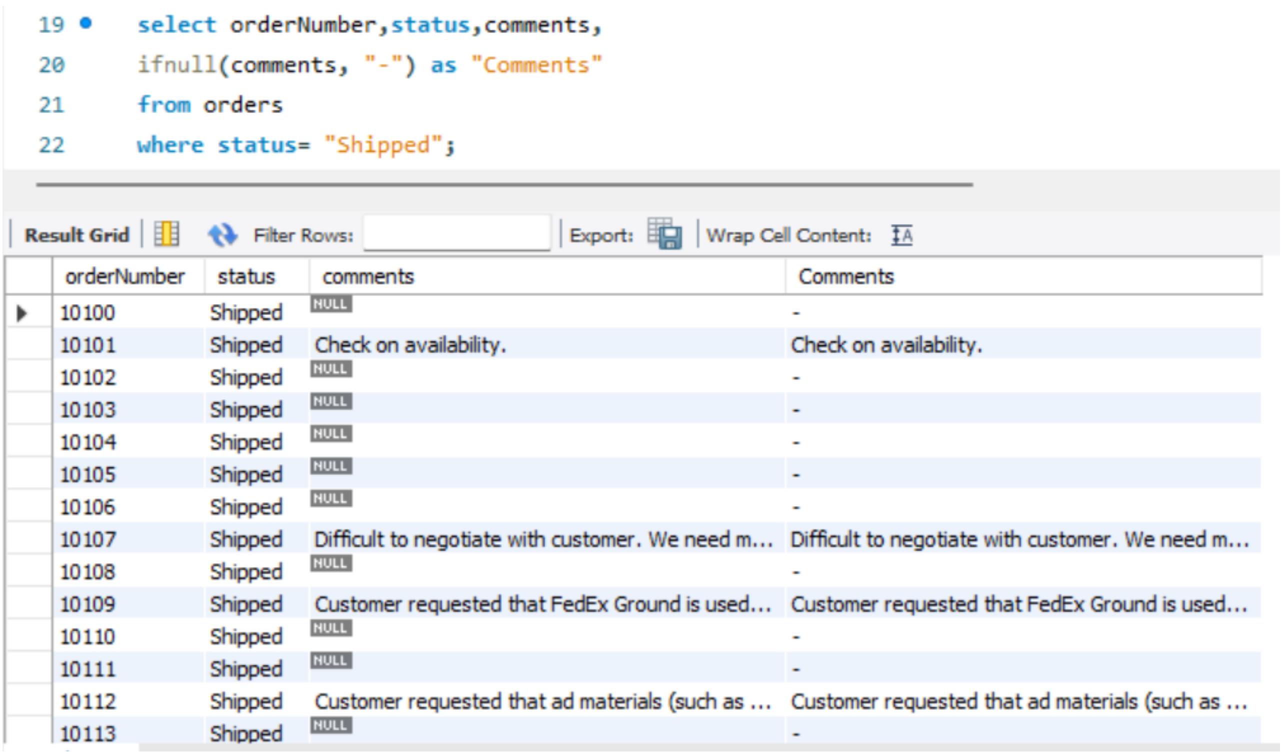
Result Grid | Filter Rows: | Edit:

productLine
Classic Cars
Vintage Cars

Day 4

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:



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

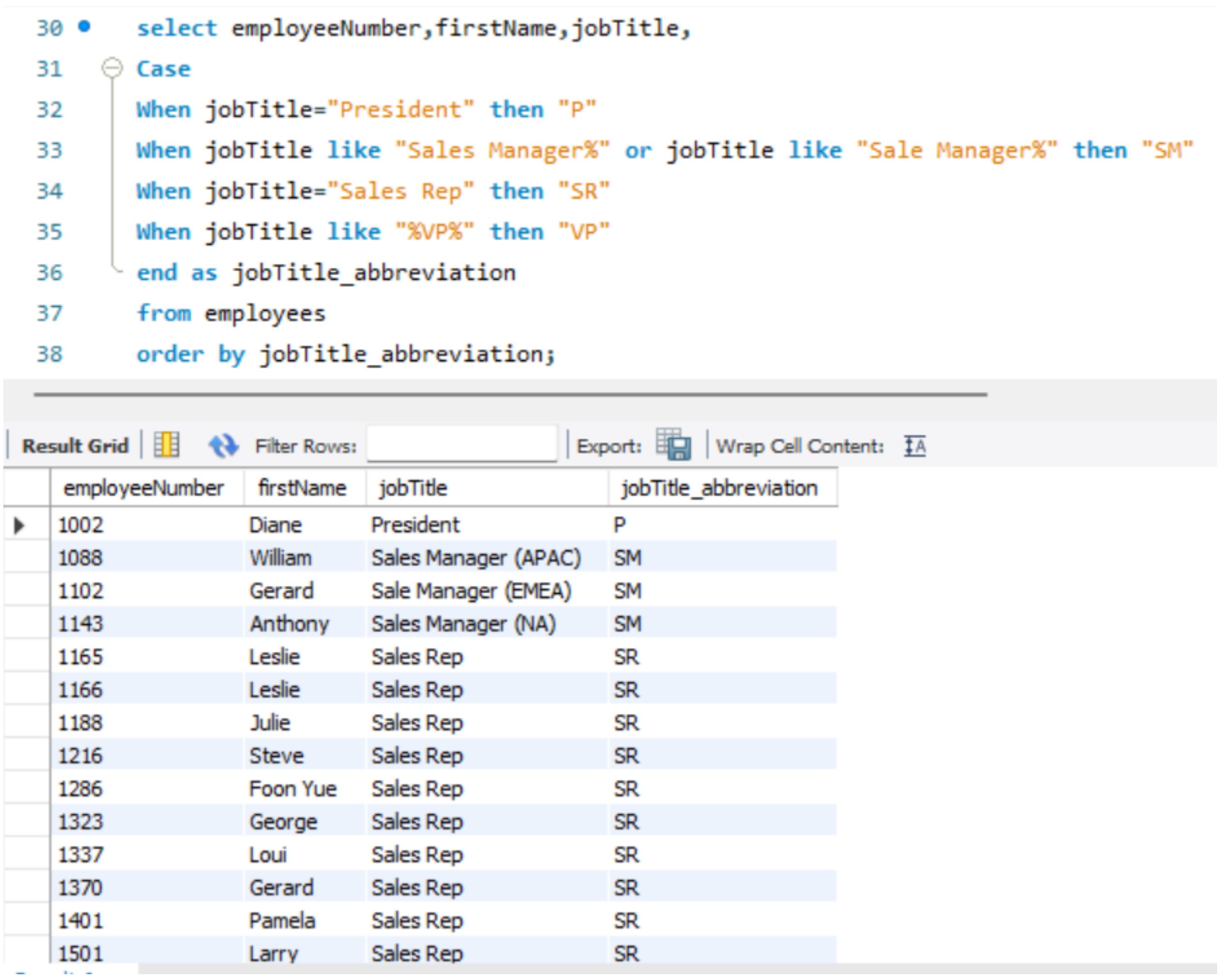
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

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

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”

Solution:



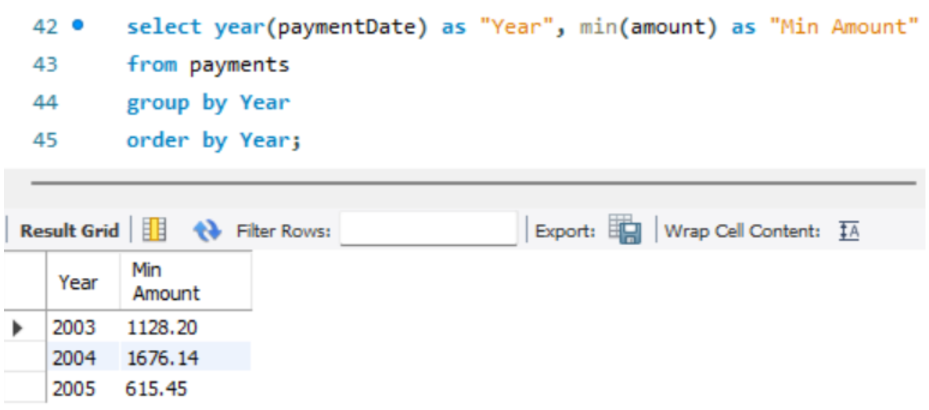
```
select employeeNumber,firstName,jobTitle,
Case
When jobTitle="President" then "P"
When jobTitle like "Sales Manager%" or jobTitle like "Sale Manager%" then "SM"
When jobTitle="Sales Rep" then "SR"
When jobTitle like "%VP%" then "VP"
end as jobTitle_abbreviation
from employees
order by jobTitle_abbreviation;
```

employeeNumber	firstName	jobTitle	jobTitle_abbreviation
1002	Diane	President	P
1088	William	Sales Manager (APAC)	SM
1102	Gerard	Sale Manager (EMEA)	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

Day 5:

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

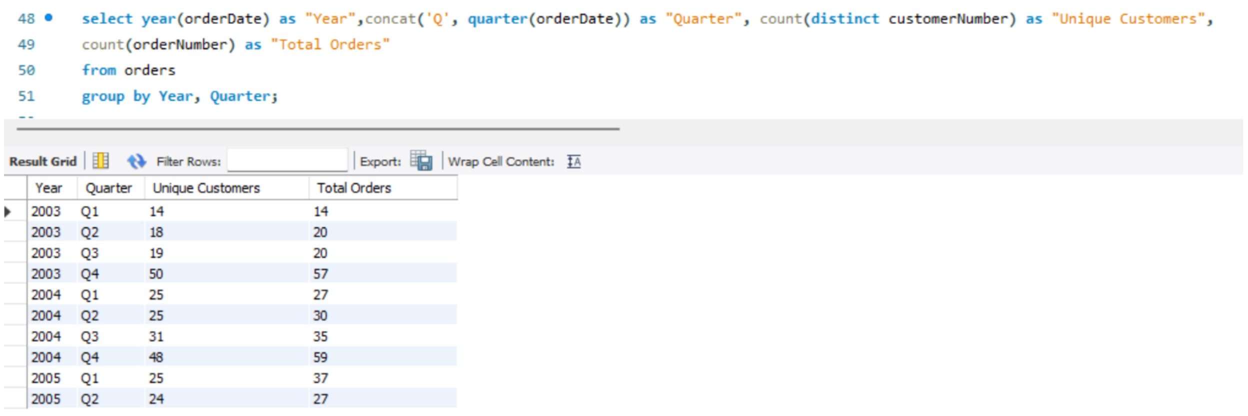
Solution:



```
select year(paymentDate) as "Year", min(amount) as "Min Amount"
from payments
group by Year
order by Year;
```

Year	Min 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.

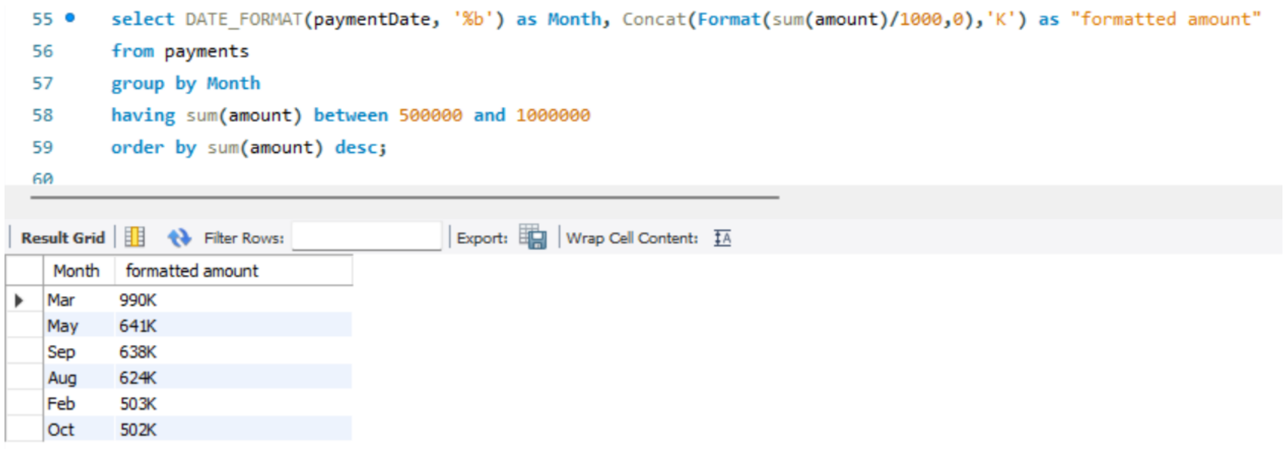


```sql
select year(orderDate) as "Year",concat('Q', quarter(orderDate)) as "Quarter", count(distinct customerNumber) as "Unique Customers",
count(orderNumber) as "Total Orders"
from orders
group by Year, Quarter;
```

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:



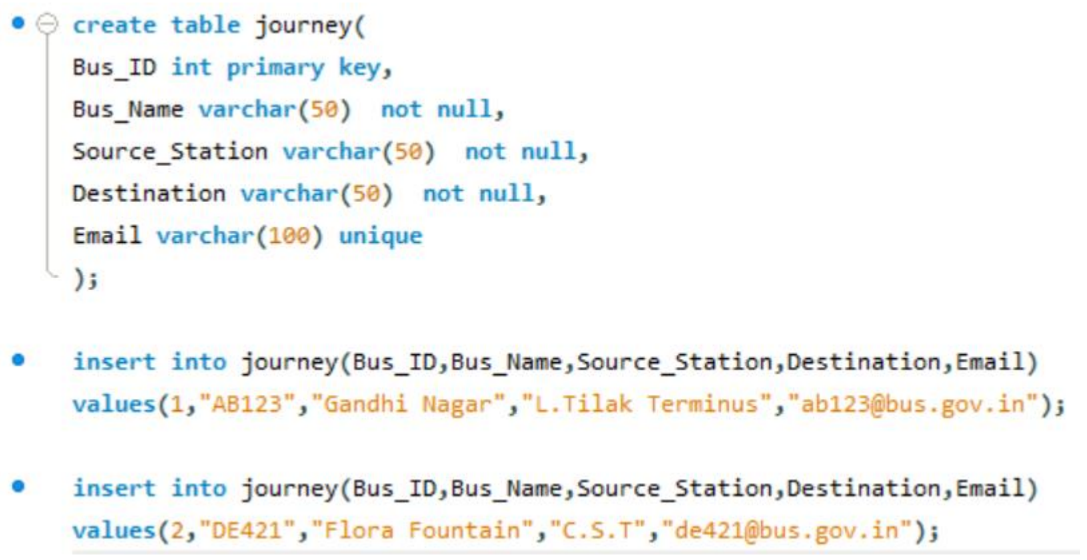
```sql
select DATE_FORMAT(paymentDate, '%b') as Month, Concat(Format(sum(amount)/1000,0),'K') as "formatted amount"
from payments
group by Month
having sum(amount) between 500000 and 1000000
order by sum(amount) desc;
```

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

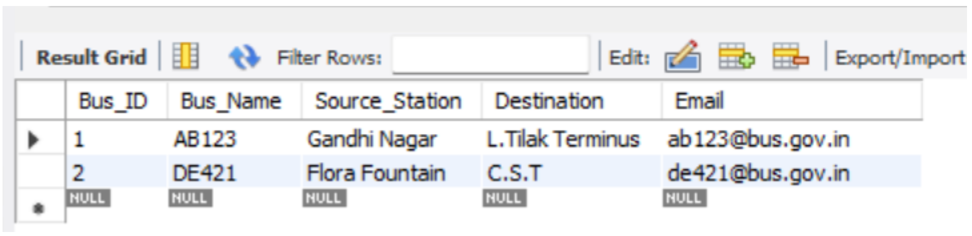
Day 6:

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)



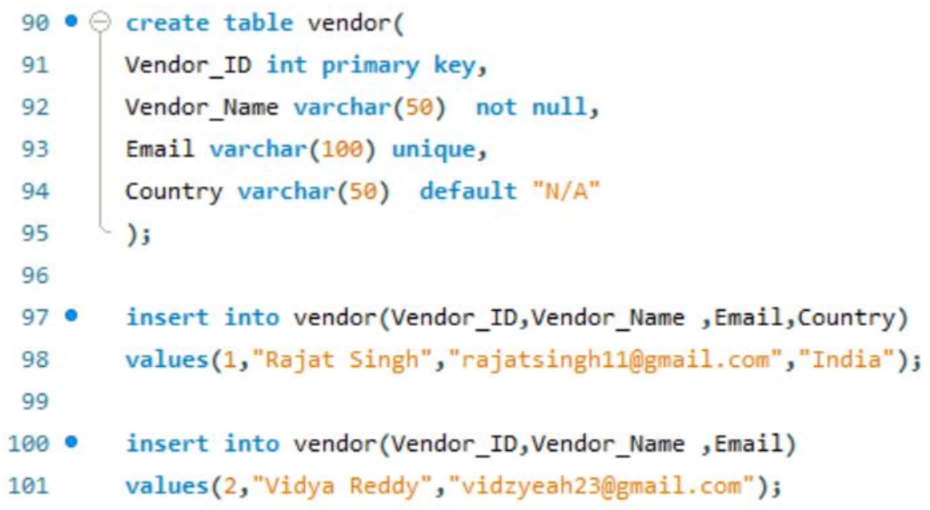
```
create table journey(
Bus_ID int primary key,
Bus_Name varchar(50)  not null,
Source_Station varchar(50)  not null,
Destination varchar(50)  not null,
Email varchar(100) unique
);

insert into journey(Bus_ID,Bus_Name,Source_Station,Destination,Email)
values(1,"AB123","Gandhi Nagar","L.Tilak Terminus","ab123@bus.gov.in");

insert into journey(Bus_ID,Bus_Name,Source_Station,Destination,Email)
values(2,"DE421","Flora Fountain","C.S.T","de421@bus.gov.in");
```



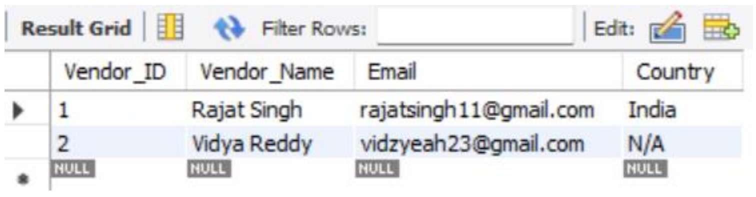
Bus_ID	Bus_Name	Source_Station	Destination	Email
1	AB123	Gandhi Nagar	L.Tilak Terminus	ab123@bus.gov.in
2	DE421	Flora Fountain	C.S.T	de421@bus.gov.in
NULL	NULL	NULL	NULL	NULL

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”)



```
create table vendor(
Vendor_ID int primary key,
Vendor_Name varchar(50)  not null,
Email varchar(100) unique,
Country varchar(50)  default "N/A"
);

insert into vendor(Vendor_ID,Vendor_Name ,Email,Country)
values(1,"Rajat Singh","rajatsingh11@gmail.com","India");

insert into vendor(Vendor_ID,Vendor_Name ,Email)
values(2,"Vidya Reddy","vidzyeah23@gmail.com");
```

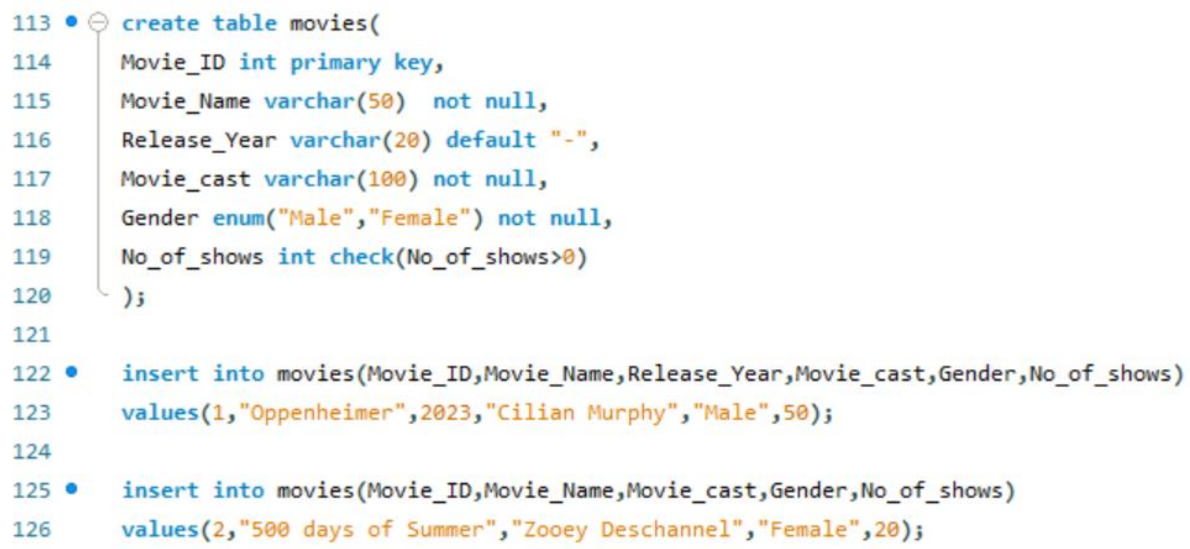


Result Grid | Filter Rows: | Edit:

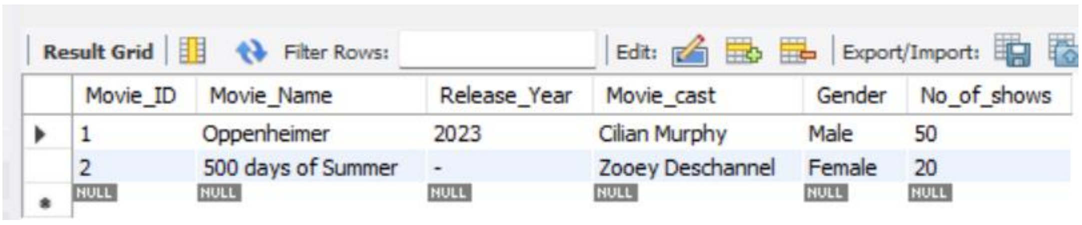
	Vendor_ID	Vendor_Name	Email	Country
▶	1	Rajat Singh	rajatsingh11@gmail.com	India
	2	Vidya Reddy	vidzyeah23@gmail.com	N/A
*	NULL	NULL	NULL	NULL

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)



```
create table movies(
Movie_ID int primary key,
Movie_Name varchar(50)  not null,
Release_Year varchar(20) default "-",
Movie_cast varchar(100) not null,
Gender enum("Male","Female") not null,
No_of_shows int check(No_of_shows>0)
);

insert into movies(Movie_ID,Movie_Name,Release_Year,Movie_cast,Gender,No_of_shows)
values(1,"Oppenheimer",2023,"Cilian Murphy","Male",50);

insert into movies(Movie_ID,Movie_Name,Movie_cast,Gender,No_of_shows)
values(2,"500 days of Summer","Zooey Deschannel","Female",20);
```



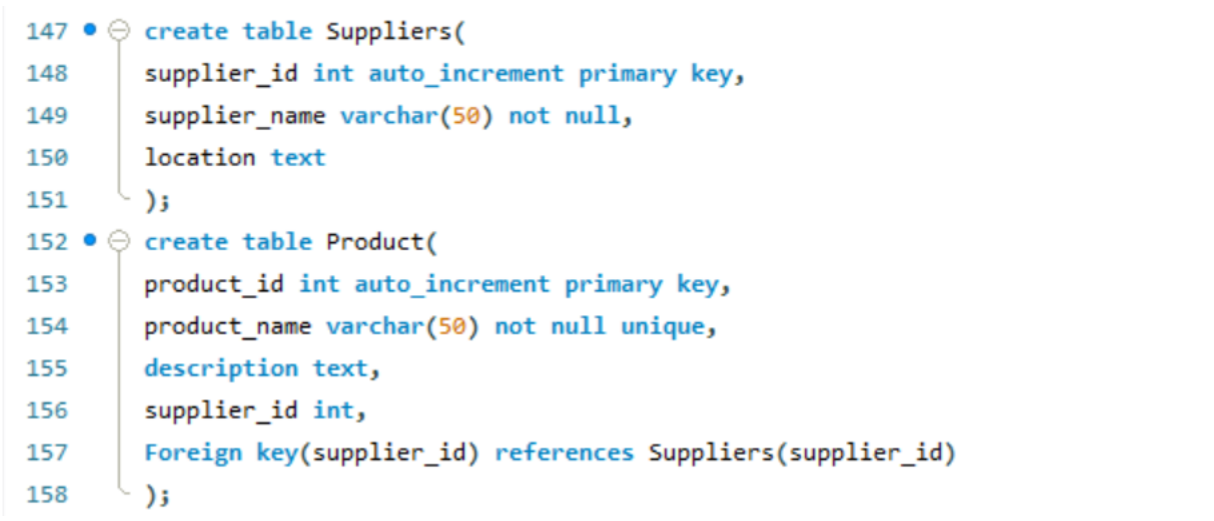
Movie_ID	Movie_Name	Release_Year	Movie_cast	Gender	No_of_shows
1	Oppenheimer	2023	Cilian Murphy	Male	50
2	500 days of Summer	-	Zooey Deschannel	Female	20
NULL	NULL	NULL	NULL	NULL	NULL

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

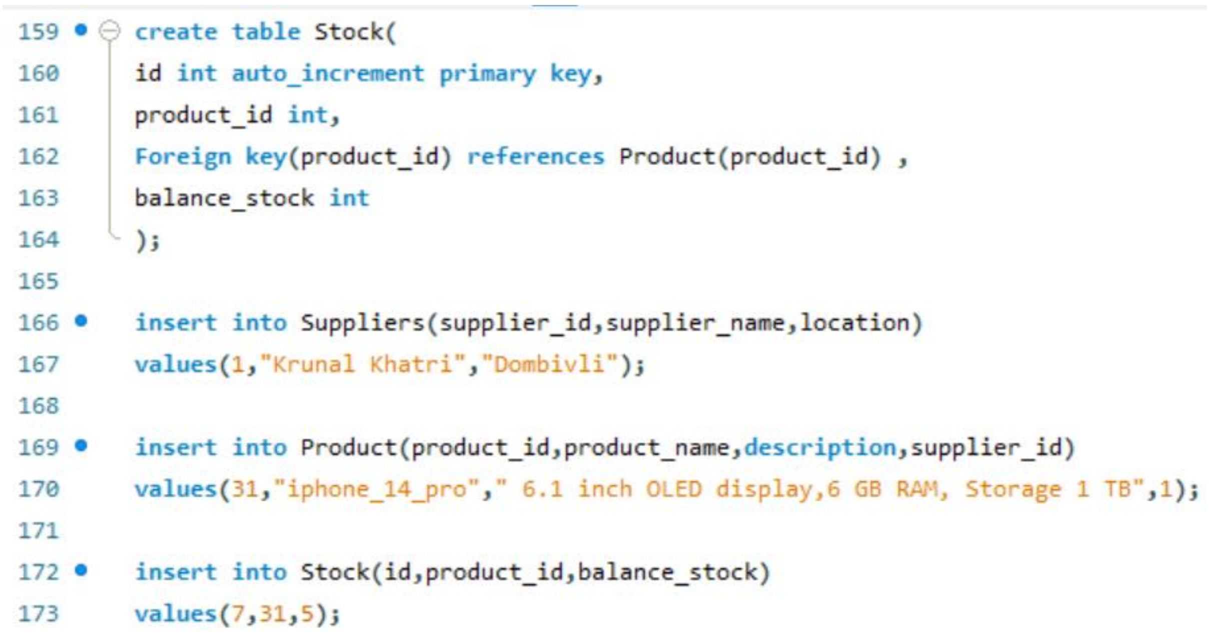
a. Product
- ✓ product_id - primary key
- ✓ product_name - cannot be null and only unique values are allowed
- ✓ description
- ✓ supplier_id - foreign key of supplier table

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

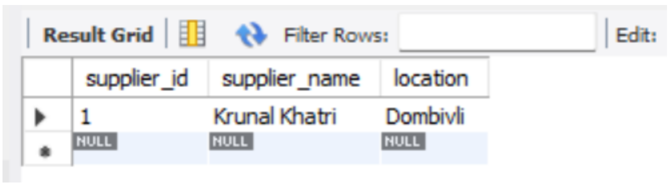
c. Stock
- ✓ id - primary key
- ✓ product_id - foreign key of product table
- ✓ balance_stock



```sql
create table Suppliers(
supplier_id int auto_increment primary key,
supplier_name varchar(50) not null,
location text
);
create table Product(
product_id int auto_increment primary key,
product_name varchar(50) not null unique,
description text,
supplier_id int,
Foreign key(supplier_id) references Suppliers(supplier_id)
);
```

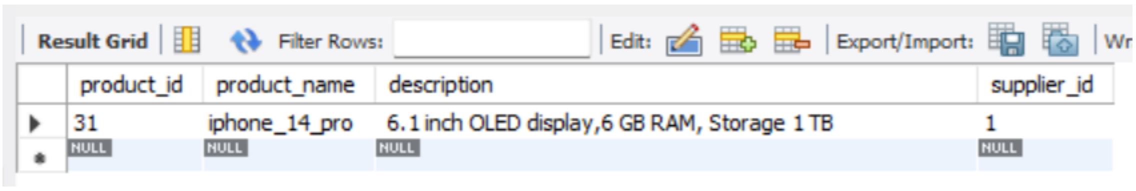


```sql
create table Stock(
id int auto_increment primary key,
product_id int,
Foreign key(product_id) references Product(product_id) ,
balance_stock int
);

insert into Suppliers(supplier_id,supplier_name,location)
values(1,"Krunal Khatri","Dombivli");

insert into Product(product_id,product_name,description,supplier_id)
values(31,"iphone_14_pro"," 6.1 inch OLED display,6 GB RAM, Storage 1 TB",1);

insert into Stock(id,product_id,balance_stock)
values(7,31,5);
```



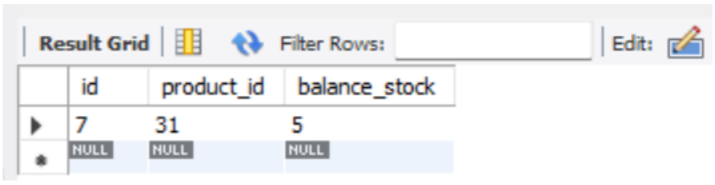
Result Grid | Filter Rows: | Edit:

	supplier_id	supplier_name	location
▶	1	Krunal Khatri	Dombivli
*	NULL	NULL	NULL



Result Grid | Filter Rows: | Edit: | Export/Import: | Wr

	product_id	product_name	description	supplier_id
▶	31	iphone_14_pro	6.1 inch OLED display,6 GB RAM, Storage 1 TB	1
*	NULL	NULL	NULL	NULL



Result Grid | Filter Rows: | Edit:

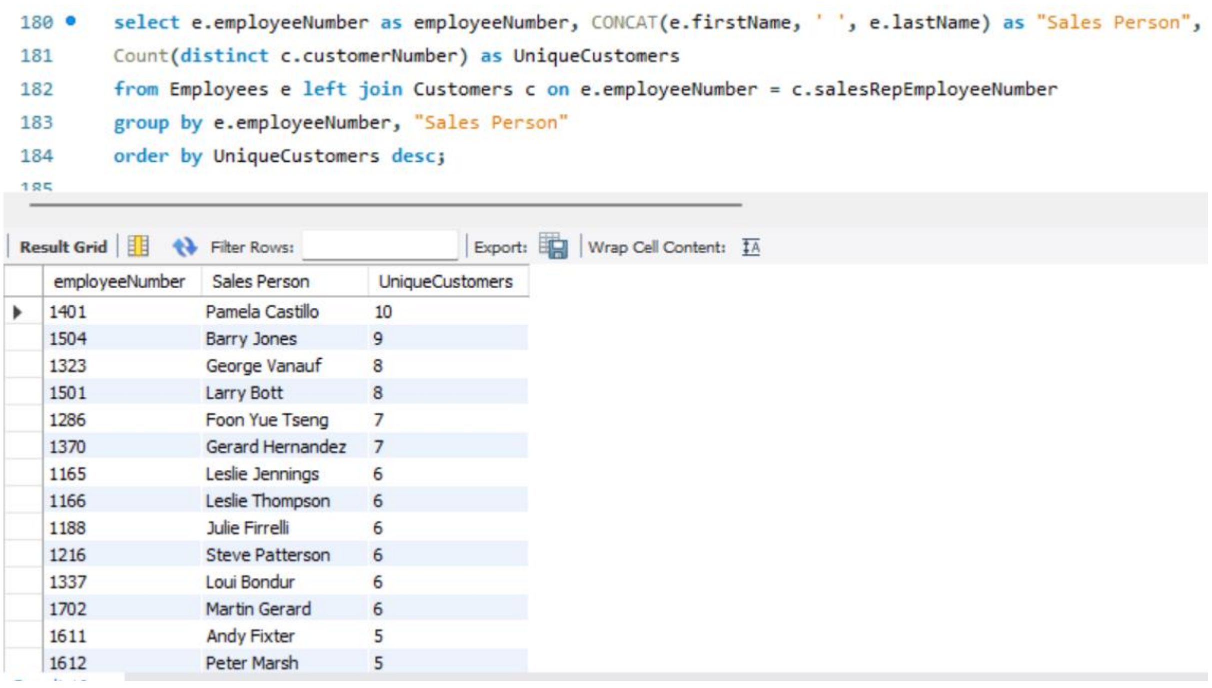
	id	product_id	balance_stock
▶	7	31	5
*	NULL	NULL	NULL

Day 7

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:



```
select e.employeeNumber as employeeNumber, CONCAT(e.firstName, ' ', e.lastName) as "Sales Person",
Count(distinct c.customerNumber) as UniqueCustomers
from Employees e left join Customers c on e.employeeNumber = c.salesRepEmployeeNumber
group by e.employeeNumber, "Sales Person"
order by UniqueCustomers desc;
```

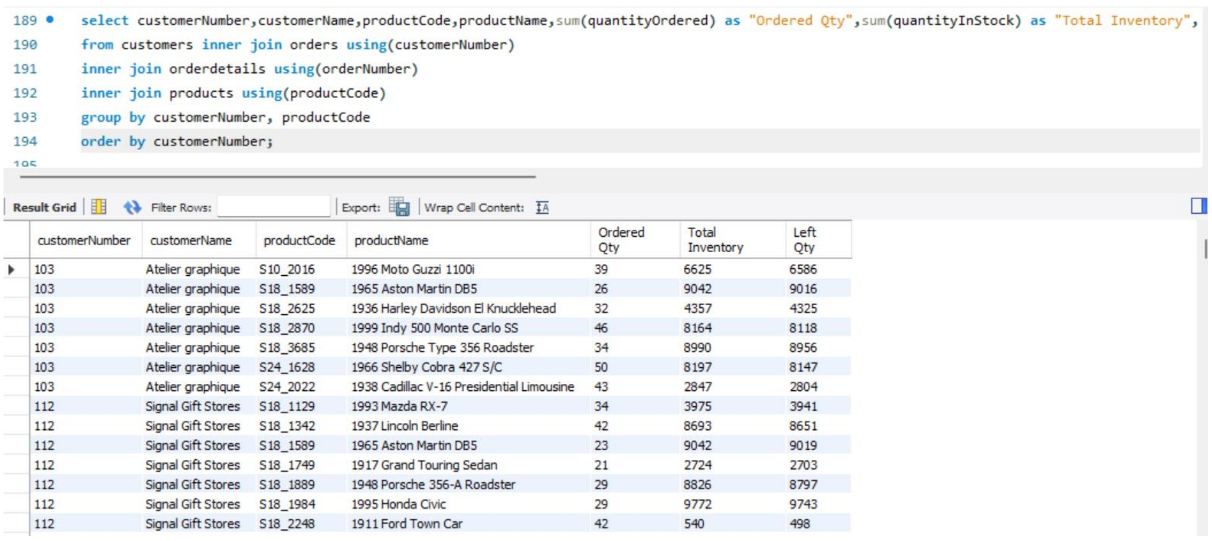
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	employeeNumber	Sales Person	UniqueCustomers
▶	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

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:



```
select customerNumber,customerName,productCode,productName,sum(quantityOrdered) as "Ordered Qty",sum(quantityInStock) as "Total Inventory",
from customers inner join orders using(customerNumber)
inner join orderdetails using(orderNumber)
inner join products using(productCode)
group by customerNumber, productCode
order by customerNumber;
```

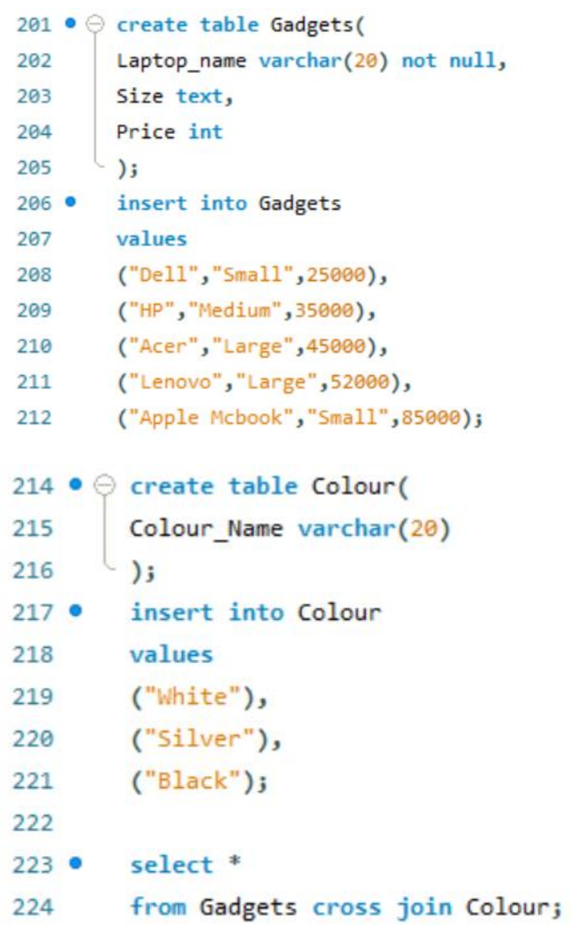
customerNumber	customerName	productCode	productName	Ordered Qty	Total Inventory	Left Qty
103	Atelier graphique	S10_2016	1996 Moto Guzzi 1100i	39	6625	6586
103	Atelier graphique	S18_1589	1965 Aston Martin DB5	26	9042	9016
103	Atelier graphique	S18_2625	1936 Harley Davidson El Knucklehead	32	4357	4325
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

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

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

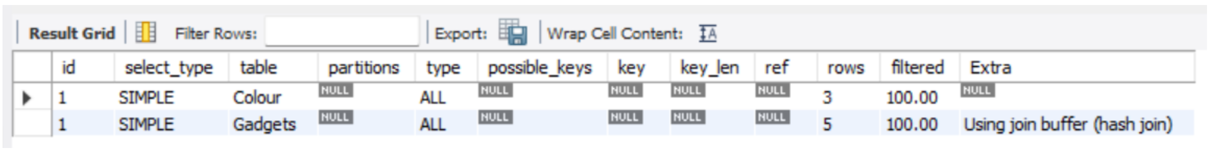
Perform cross join between the two tables and find number of rows.

Expected output:



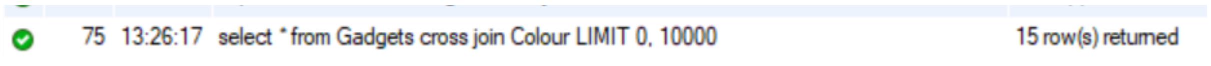
```
create table Gadgets(
Laptop_name varchar(20) not null,
Size text,
Price int
);
insert into Gadgets
values
("Dell","Small",25000),
("HP","Medium",35000),
("Acer","Large",45000),
("Lenovo","Large",52000),
("Apple Mcbook","Small",85000);

create table Colour(
Colour_Name varchar(20)
);
insert into Colour
values
("White"),
("Silver"),
("Black");

select *
from Gadgets cross join Colour;
```

Individual rows returned before execution of the query. (using explain before the query)

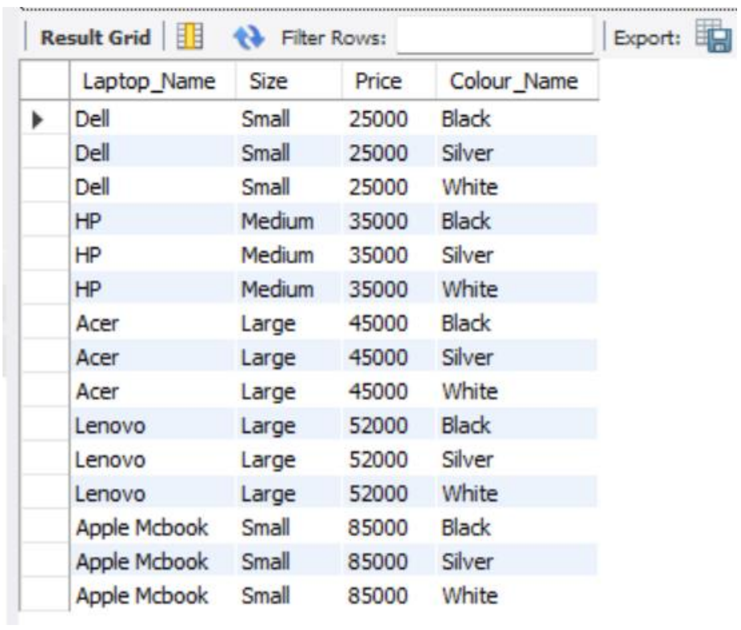


id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	Colour	NULL	ALL	NULL	NULL	NULL	NULL	3	100.00	NULL
1	SIMPLE	Gadgets	NULL	ALL	NULL	NULL	NULL	NULL	5	100.00	Using join buffer (hash join)

No of rows returned after execution of the query.



```
75  13:26:17  select * from Gadgets cross join Colour LIMIT 0, 10000        15 row(s) returned
```



Laptop_Name	Size	Price	Colour_Name
Dell	Small	25000	Black
Dell	Small	25000	Silver
Dell	Small	25000	White
HP	Medium	35000	Black
HP	Medium	35000	Silver
HP	Medium	35000	White
Acer	Large	45000	Black
Acer	Large	45000	Silver
Acer	Large	45000	White
Lenovo	Large	52000	Black
Lenovo	Large	52000	Silver
Lenovo	Large	52000	White
Apple Mcbook	Small	85000	Black
Apple Mcbook	Small	85000	Silver
Apple Mcbook	Small	85000	White

4) Create table project with below fields.

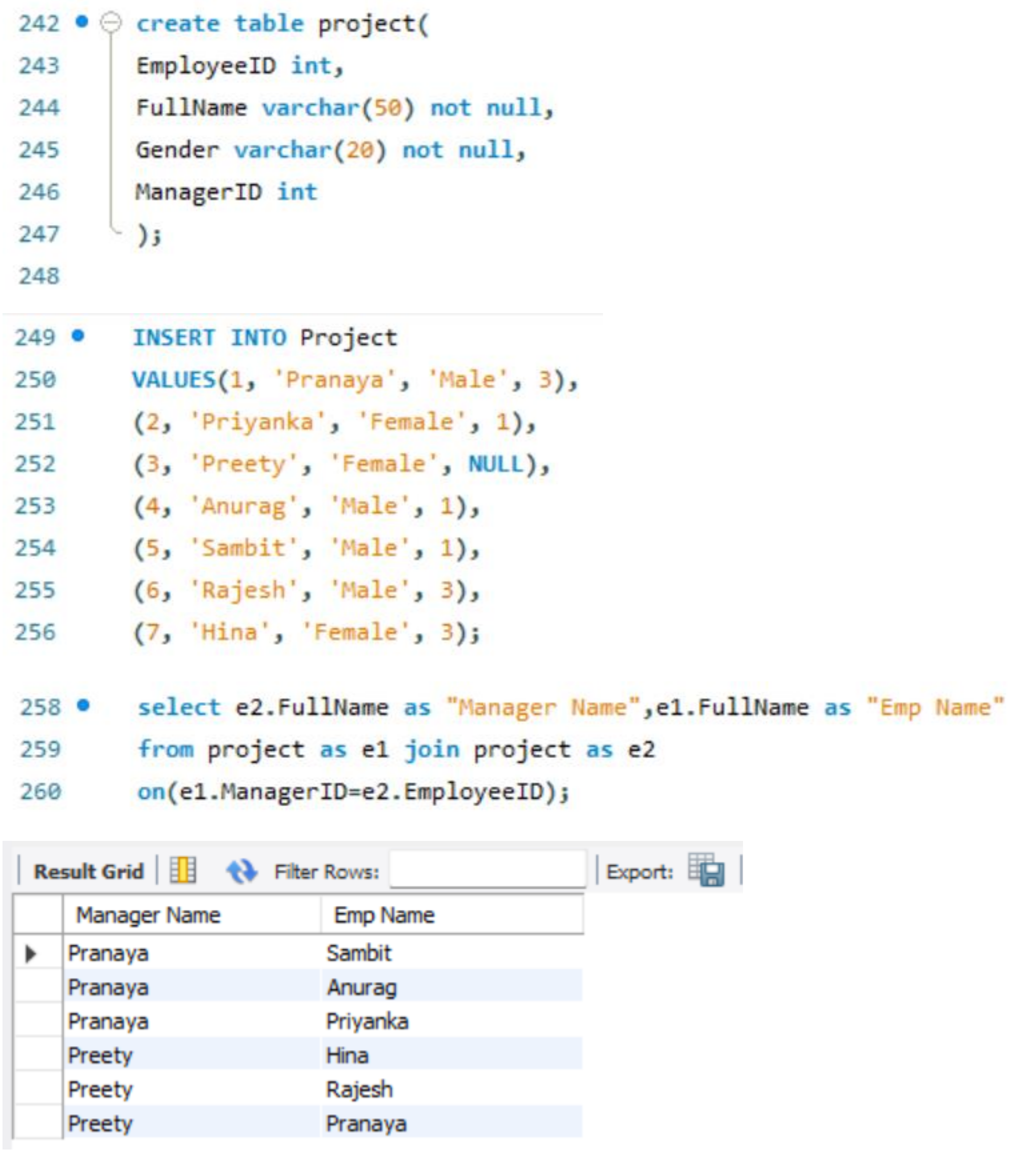
- EmployeeID
- FullName
- Gender
- ManagerID

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:



```
create table project(
EmployeeID int,
FullName varchar(50) not null,
Gender varchar(20) not null,
ManagerID int
);

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

select e2.FullName as "Manager Name",e1.FullName as "Emp Name"
from project as e1 join project as e2
on(e1.ManagerID=e2.EmployeeID);
```

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

Day 8

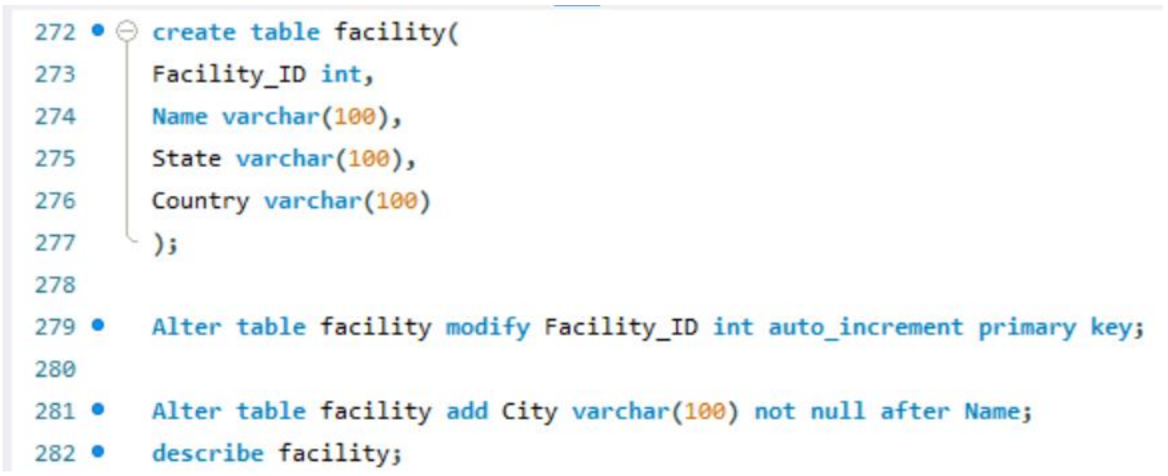
Create table facility. Add the below fields into it.

- Facility_ID
- Name
- State
- Country

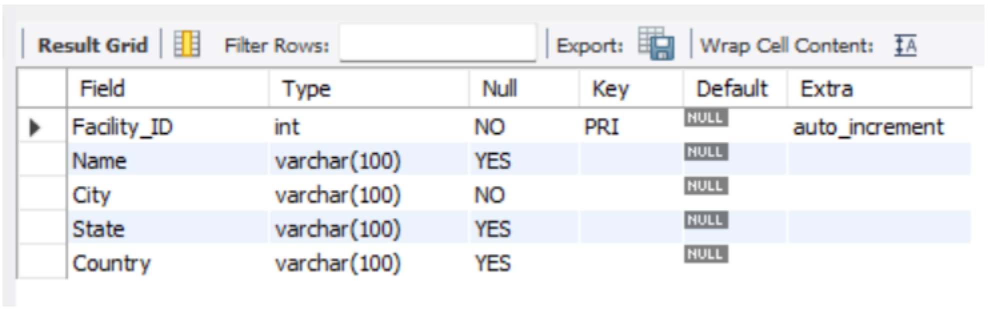
i) Alter the table by adding the primary key and auto increment to Facility_ID column.

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

Expected output:



```
create table facility(
Facility_ID int,
Name varchar(100),
State varchar(100),
Country varchar(100)
);

Alter table facility modify Facility_ID int auto_increment primary key;

Alter table facility add City varchar(100) not null after Name;
describe facility;
```



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

	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	

Day 9

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   "),

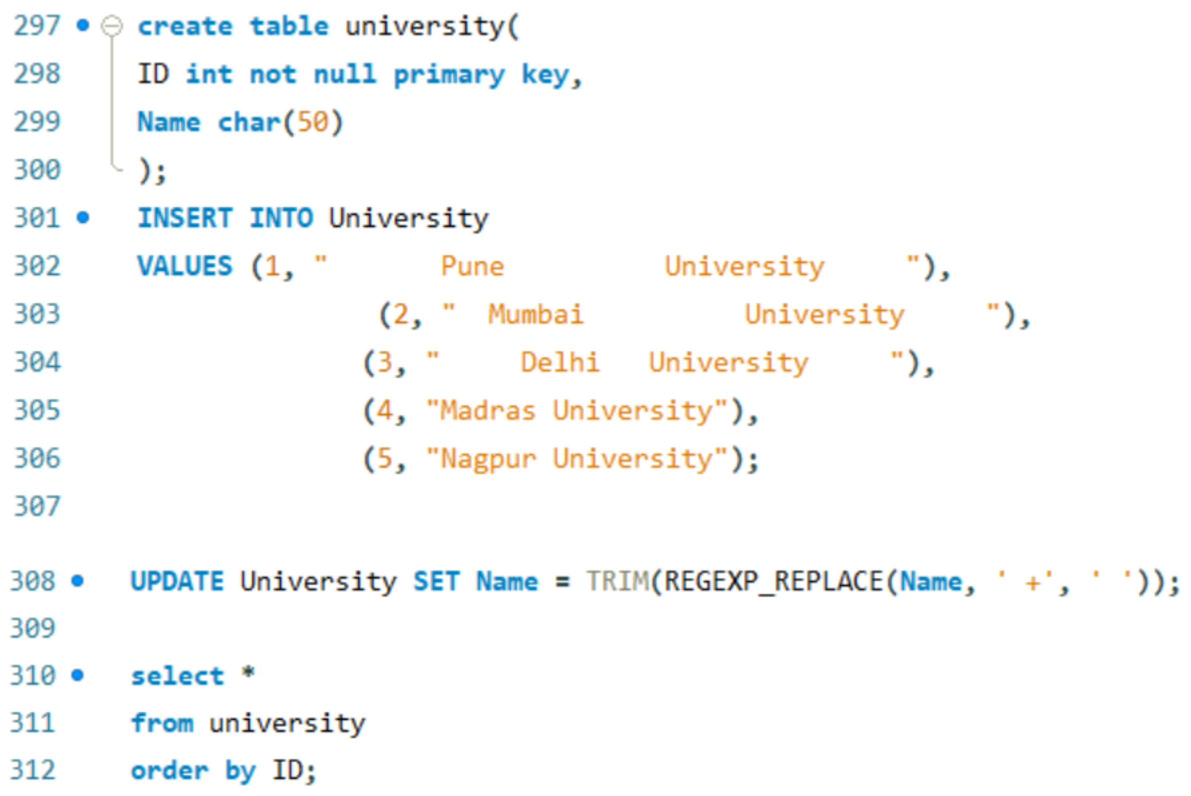
(3, "   Delhi   University   "),

(4, "Madras University"),

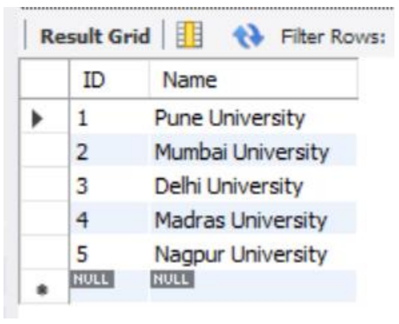
(5, "Nagpur University");

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

Expected output:



```
create table university(
ID int not null primary key,
Name char(50)
);
INSERT INTO University
VALUES (1, "       Pune          University     "),
                 (2, "  Mumbai           University     "),
                (3, "     Delhi    University     "),
                (4, "Madras University"),
                (5, "Nagpur University");

UPDATE University SET Name = TRIM(REGEXP_REPLACE(Name, ' +', ' '));

select *
from university
order by ID;
```



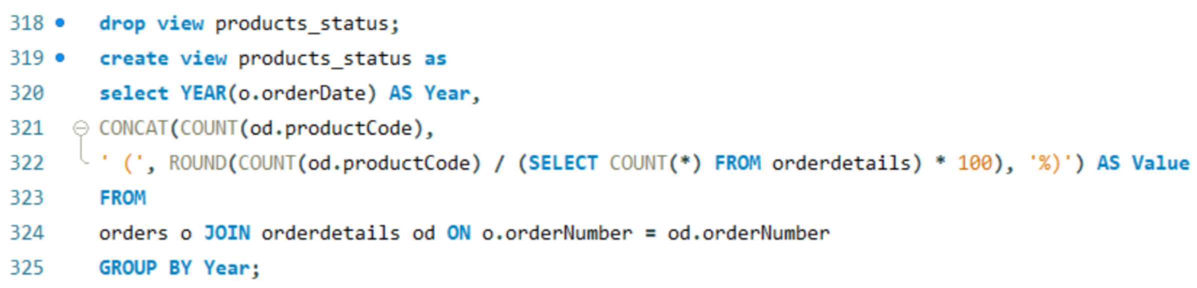
Result Grid | Filter Rows:

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

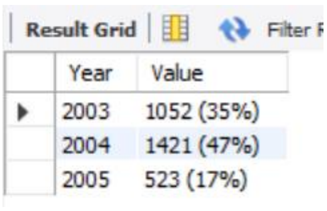
Day 10

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:



```
drop view products_status;
create view products_status as
select YEAR(o.orderDate) AS Year,
CONCAT(COUNT(od.productCode),
' (', ROUND(COUNT(od.productCode) / (SELECT COUNT(*) FROM orderdetails) * 100), '%)') AS Value
FROM
orders o JOIN orderdetails od ON o.orderNumber = od.orderNumber
GROUP BY Year;
```



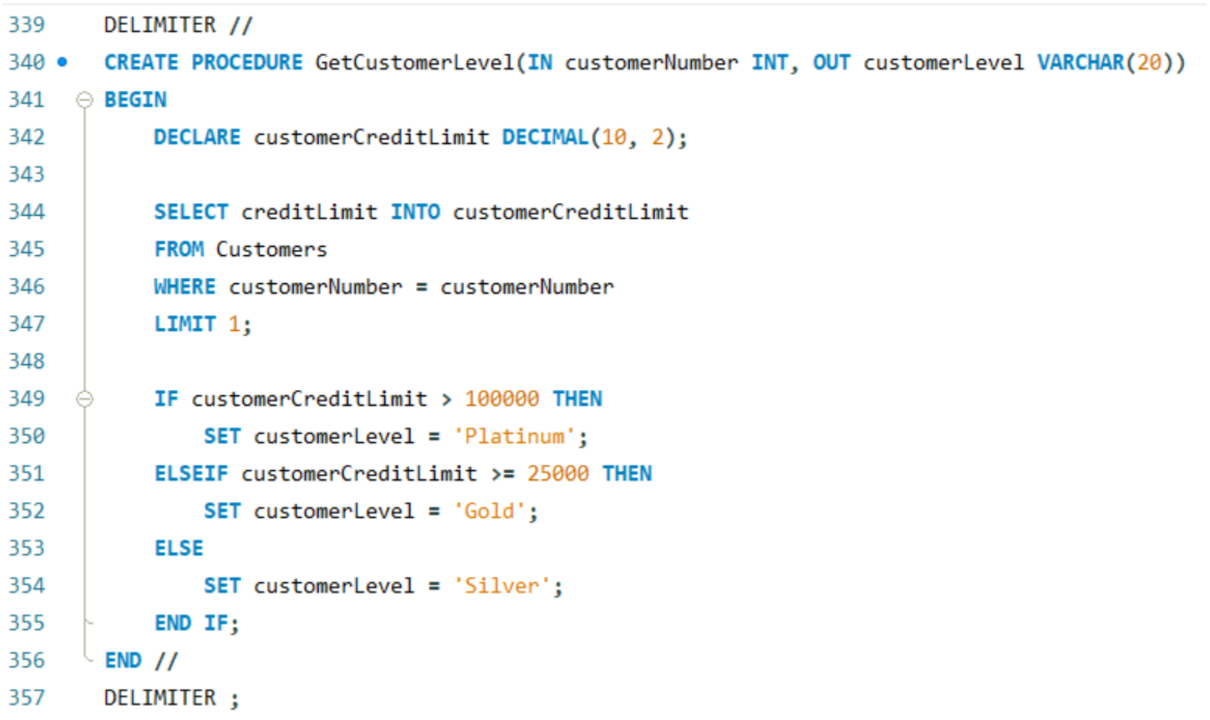
Year	Value
2003	1052 (35%)
2004	1421 (47%)
2005	523 (17%)

Day 11

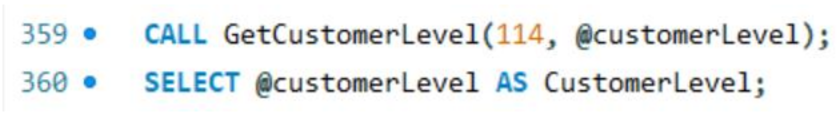
1) Create a stored procedure GetCustomerLevel which takes input as customer number and gives the output as either Platinum, Gold or Silver as per below criteria.

   Table: Customers

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



```
DELIMITER //
CREATE PROCEDURE GetCustomerLevel(IN customerNumber INT, OUT customerLevel VARCHAR(20))
BEGIN
    DECLARE customerCreditLimit DECIMAL(10, 2);

    SELECT creditLimit INTO customerCreditLimit
    FROM Customers
    WHERE customerNumber = customerNumber
    LIMIT 1;

    IF customerCreditLimit > 100000 THEN
        SET customerLevel = 'Platinum';
    ELSEIF customerCreditLimit >= 25000 THEN
        SET customerLevel = 'Gold';
    ELSE
        SET customerLevel = 'Silver';
    END IF;
END //
DELIMITER ;
```

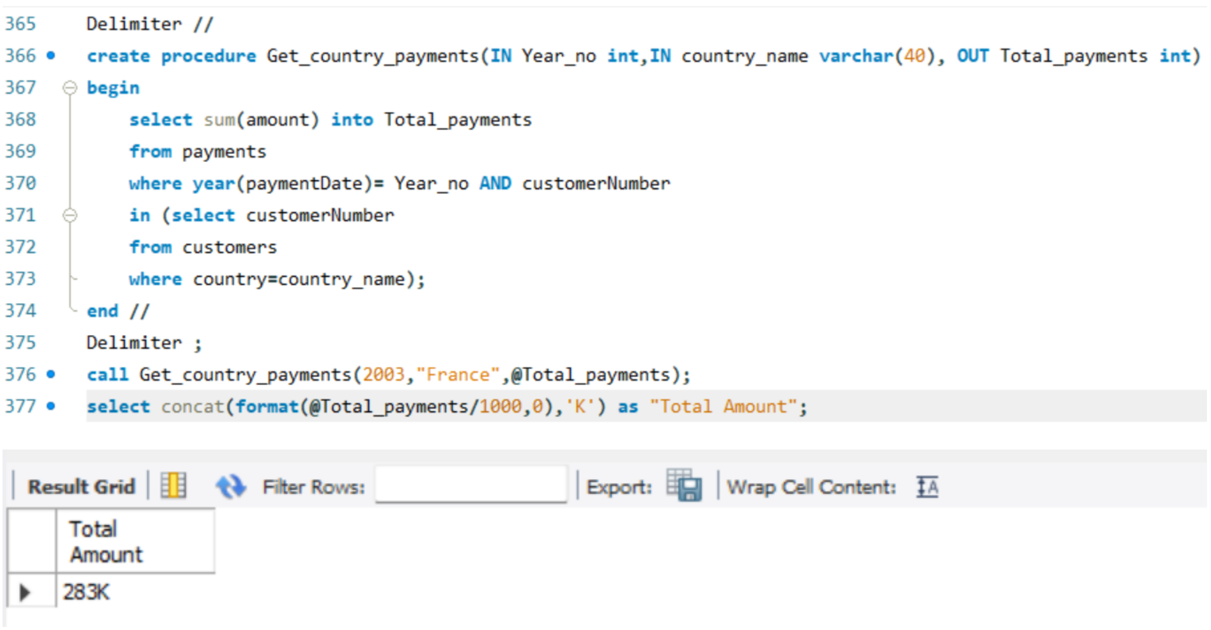


```
CALL GetCustomerLevel(114, @customerLevel);
SELECT @customerLevel AS CustomerLevel;
```

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:



```sql
Delimiter //
create procedure Get_country_payments(IN Year_no int,IN country_name varchar(40), OUT Total_payments int)
begin
    select sum(amount) into Total_payments
    from payments
    where year(paymentDate)= Year_no AND customerNumber
    in (select customerNumber
    from customers
    where country=country_name);
end //
Delimiter ;
call Get_country_payments(2003,"France",@Total_payments);
select concat(format(@Total_payments/1000,0),'K') as "Total Amount";
```

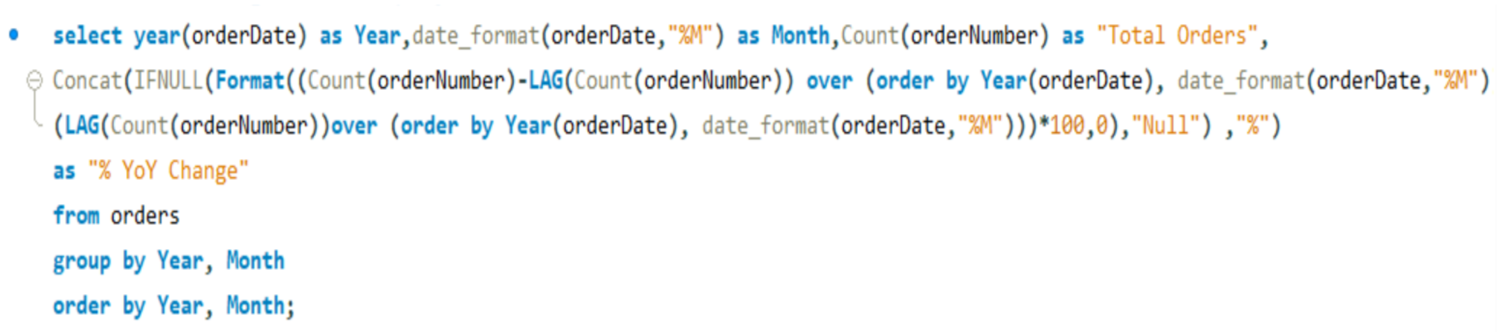
```
Total
Amount
283K
```

Day 12

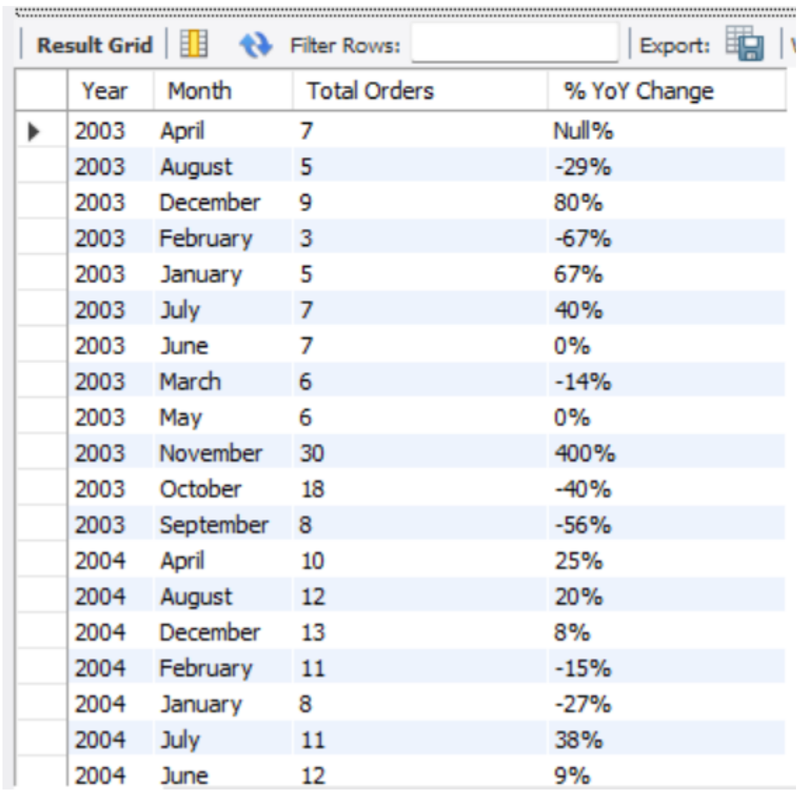
1) Calculate year wise, month name wise count of orders and year over year (YoY) percentage change. Format the YoY values in no decimals and show in % sign.

   Table: Orders

   Expected output:



```sql
select year(orderDate) as Year,date_format(orderDate,"%M") as Month,Count(orderNumber) as "Total Orders",
Concat(IFNULL(Format((Count(orderNumber)-LAG(Count(orderNumber)) over (order by Year(orderDate), date_format(orderDate,"%M")
(LAG(Count(orderNumber))over (order by Year(orderDate), date_format(orderDate,"%M")))*100,0),"Null") ,"%")
as "% YoY Change"
from orders
group by Year, Month
order by Year, Month;
```



	Year	Month	Total Orders	% YoY Change
▶	2003	April	7	Null%
	2003	August	5	-29%
	2003	December	9	80%
	2003	February	3	-67%
	2003	January	5	67%
	2003	July	7	40%
	2003	June	7	0%
	2003	March	6	-14%
	2003	May	6	0%
	2003	November	30	400%
	2003	October	18	-40%
	2003	September	8	-56%
	2004	April	10	25%
	2004	August	12	20%
	2004	December	13	8%
	2004	February	11	-15%
	2004	January	8	-27%
	2004	July	11	38%
	2004	June	12	9%

2) Create the table emp_udf with below fields.

- Emp_ID
- Name
- DOB

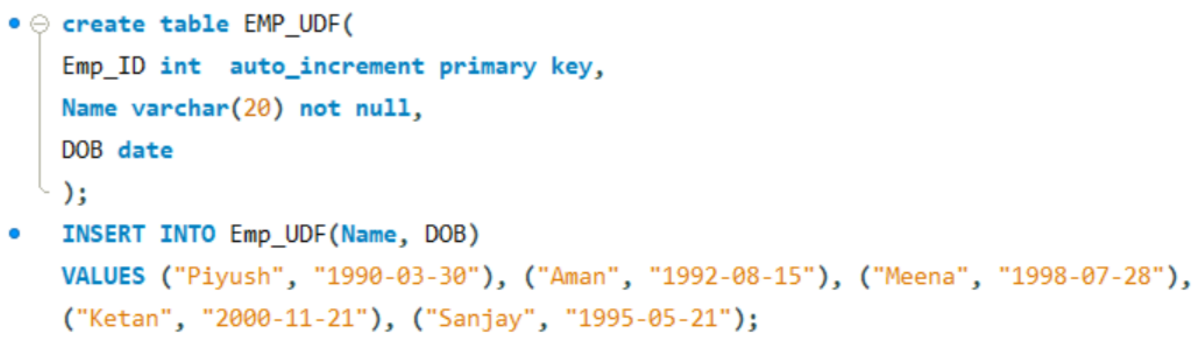
Add the data as shown in below query.

INSERT INTO Emp_UDF(Name, DOB)

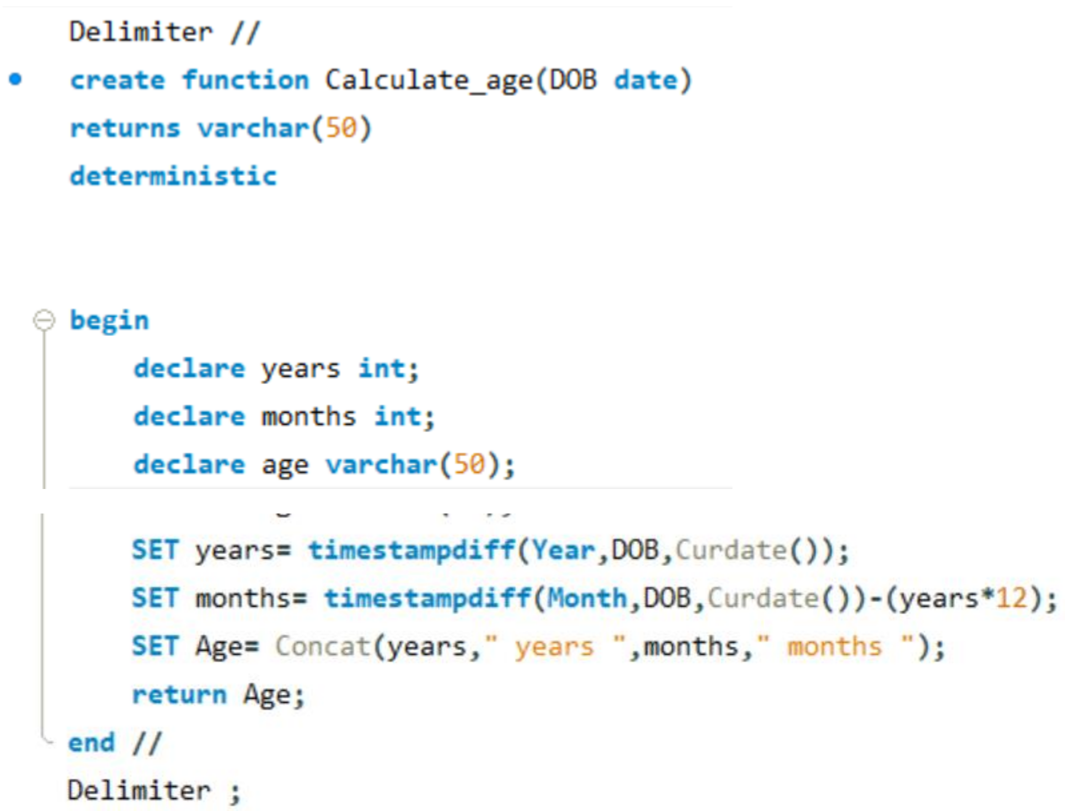
VALUES ("Piyush", "1990-03-30"), ("Aman", "1992-08-15"), ("Meena", "1998-07-28"), ("Ketan", "2000-11-21"), ("Sanjay", "1995-05-21");

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:

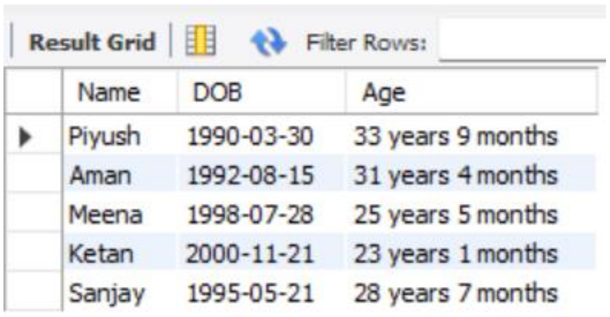


```
create table EMP_UDF(
Emp_ID int  auto_increment primary key,
Name varchar(20) not null,
DOB date
);
INSERT INTO Emp_UDF(Name, DOB)
VALUES ("Piyush", "1990-03-30"), ("Aman", "1992-08-15"), ("Meena", "1998-07-28"),
("Ketan", "2000-11-21"), ("Sanjay", "1995-05-21");
```



```
Delimiter //
create function Calculate_age(DOB date)
returns varchar(50)
deterministic


begin
    declare years int;
    declare months int;
    declare age varchar(50);
    SET years= timestampdiff(Year,DOB,Curdate());
    SET months= timestampdiff(Month,DOB,Curdate())-(years*12);
    SET Age= Concat(years," years ",months," months ");
    return Age;
end //
Delimiter ;
```

```
select Name,DOB, Calculate_age(DOB) as Age
from EMP_UDF;
```



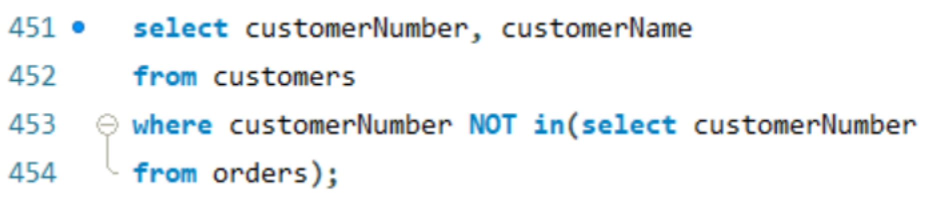
Name	DOB	Age
Piyush	1990-03-30	33 years 9 months
Aman	1992-08-15	31 years 4 months
Meena	1998-07-28	25 years 5 months
Ketan	2000-11-21	23 years 1 months
Sanjay	1995-05-21	28 years 7 months

Day 13

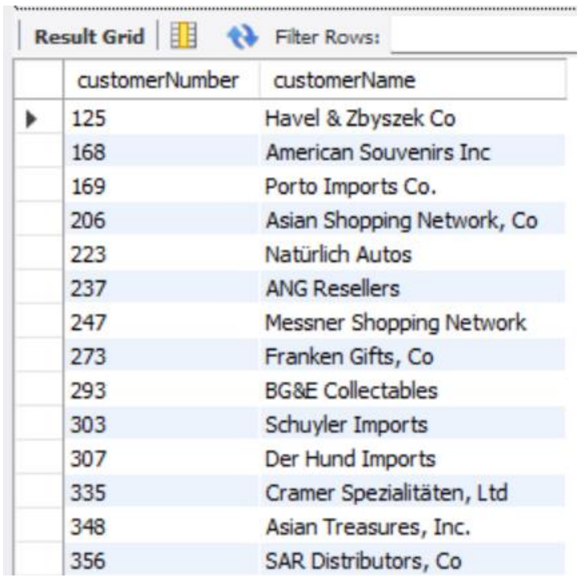
1) Display the customer numbers and customer names from customers table who have not placed any orders using subquery

   Table: Customers, Orders

   Expected output:



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

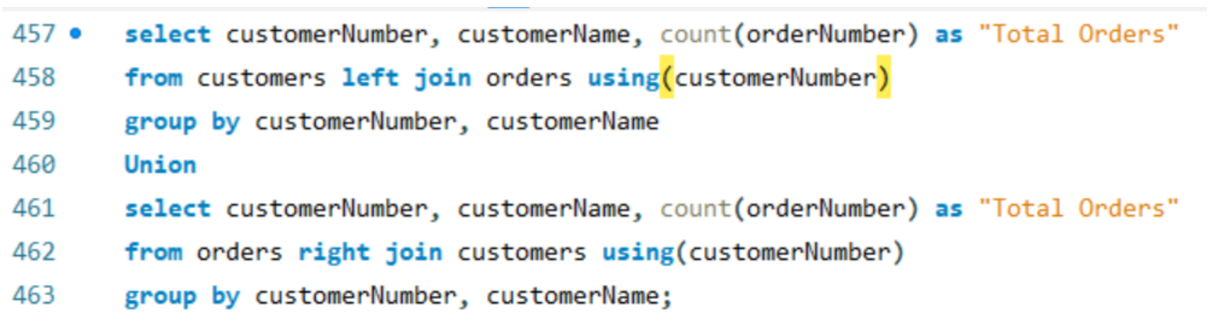


customerNumber	customerName
125	Havel & Zbyszek Co
168	American Souvenirs Inc
169	Porto Imports Co.
206	Asian Shopping Network, Co
223	Natürlich Autos
237	ANG Resellers
247	Messner Shopping Network
273	Franken Gifts, Co
293	BG&E Collectables
303	Schuyler Imports
307	Der Hund Imports
335	Cramer Spezialitäten,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

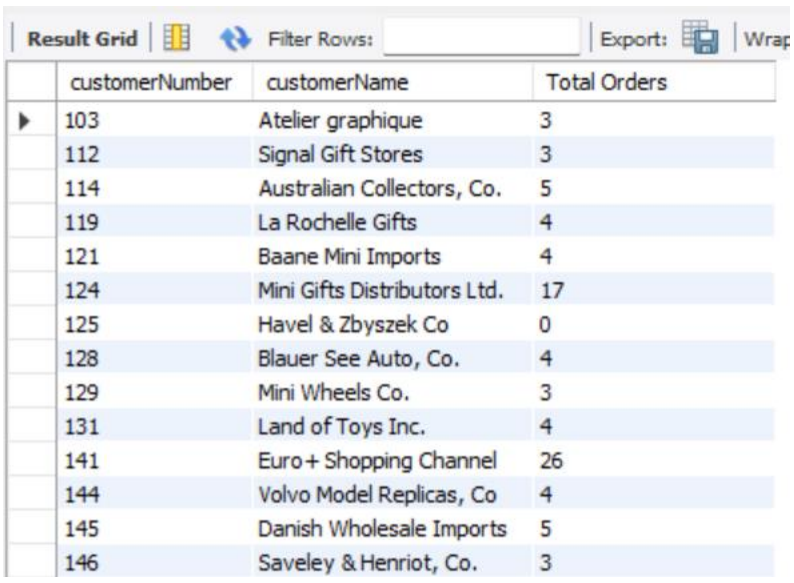
2) Write a full outer join between customers and orders using union and get the customer number, customer name, count of orders for every customer.

   Table: Customers, Orders

   Expected output:



```sql
select customerNumber, customerName, count(orderNumber) as "Total Orders"
from customers left join orders using(customerNumber)
group by customerNumber, customerName
Union
select customerNumber, customerName, count(orderNumber) as "Total Orders"
from orders right join customers using(customerNumber)
group by customerNumber, customerName;
```

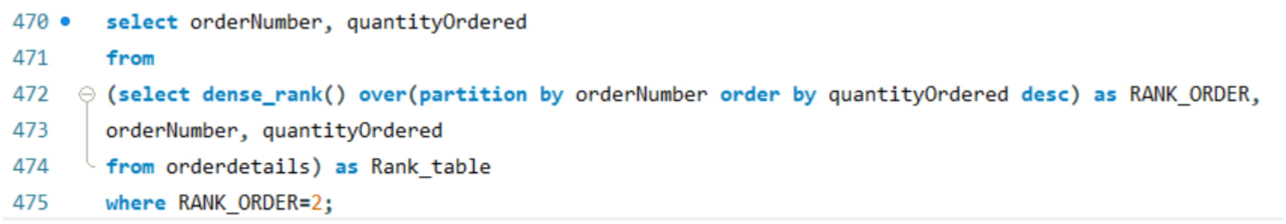


customerNumber	customerName	Total Orders
103	Atelier graphique	3
112	Signal Gift Stores	3
114	Australian Collectors, Co.	5
119	La Rochelle Gifts	4
121	Baane Mini Imports	4
124	Mini Gifts Distributors Ltd.	17
125	Havel & Zbyszek Co	0
128	Blauer See Auto, Co.	4
129	Mini Wheels Co.	3
131	Land of Toys Inc.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

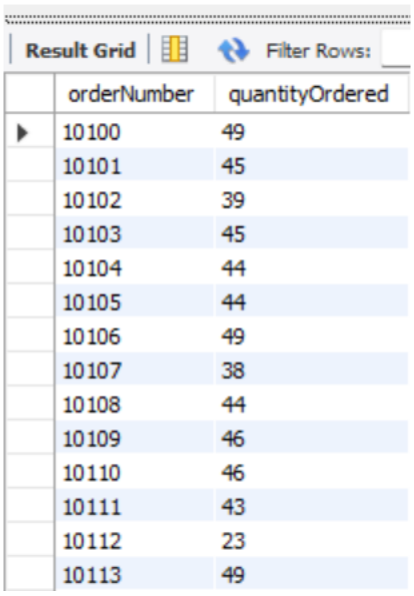
3) Show the second highest quantity ordered value for each order number.

Table: Orderdetails

Expected output:



```
select orderNumber, quantityOrdered
from
(select dense_rank() over(partition by orderNumber order by quantityOrdered desc) as RANK_ORDER,
orderNumber, quantityOrdered
from orderdetails) as Rank_table
where RANK_ORDER=2;
```

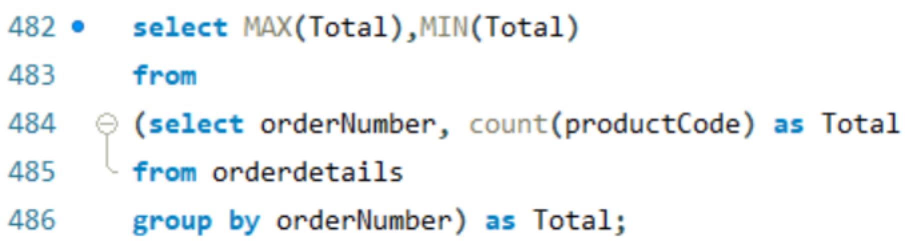


orderNumber	quantityOrdered
10100	49
10101	45
10102	39
10103	45
10104	44
10105	44
10106	49
10107	38
10108	44
10109	46
10110	46
10111	43
10112	23
10113	49

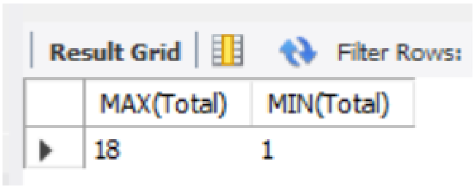
4) For each order number count the number of products and then find the min and max of the values among count of orders.

Table: Orderdetails

Expected output:



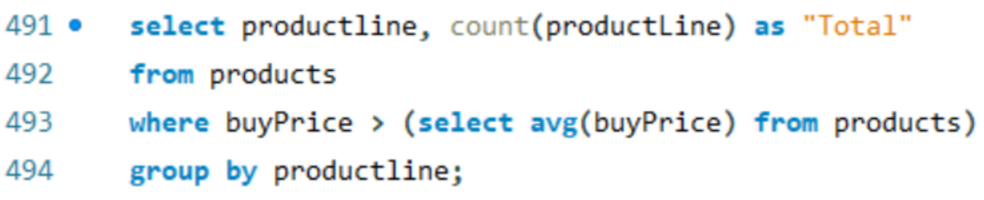
```
select MAX(Total),MIN(Total)
from
(select orderNumber, count(productCode) as Total
from orderdetails
group by orderNumber) as Total;
```



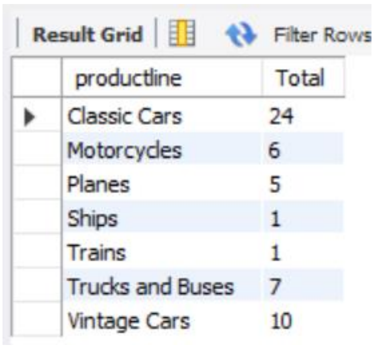
	MAX(Total)	MIN(Total)
▶	18	1

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:



```
select productline, count(productLine) as "Total"
from products
where buyPrice > (select avg(buyPrice) from products)
group by productline;
```



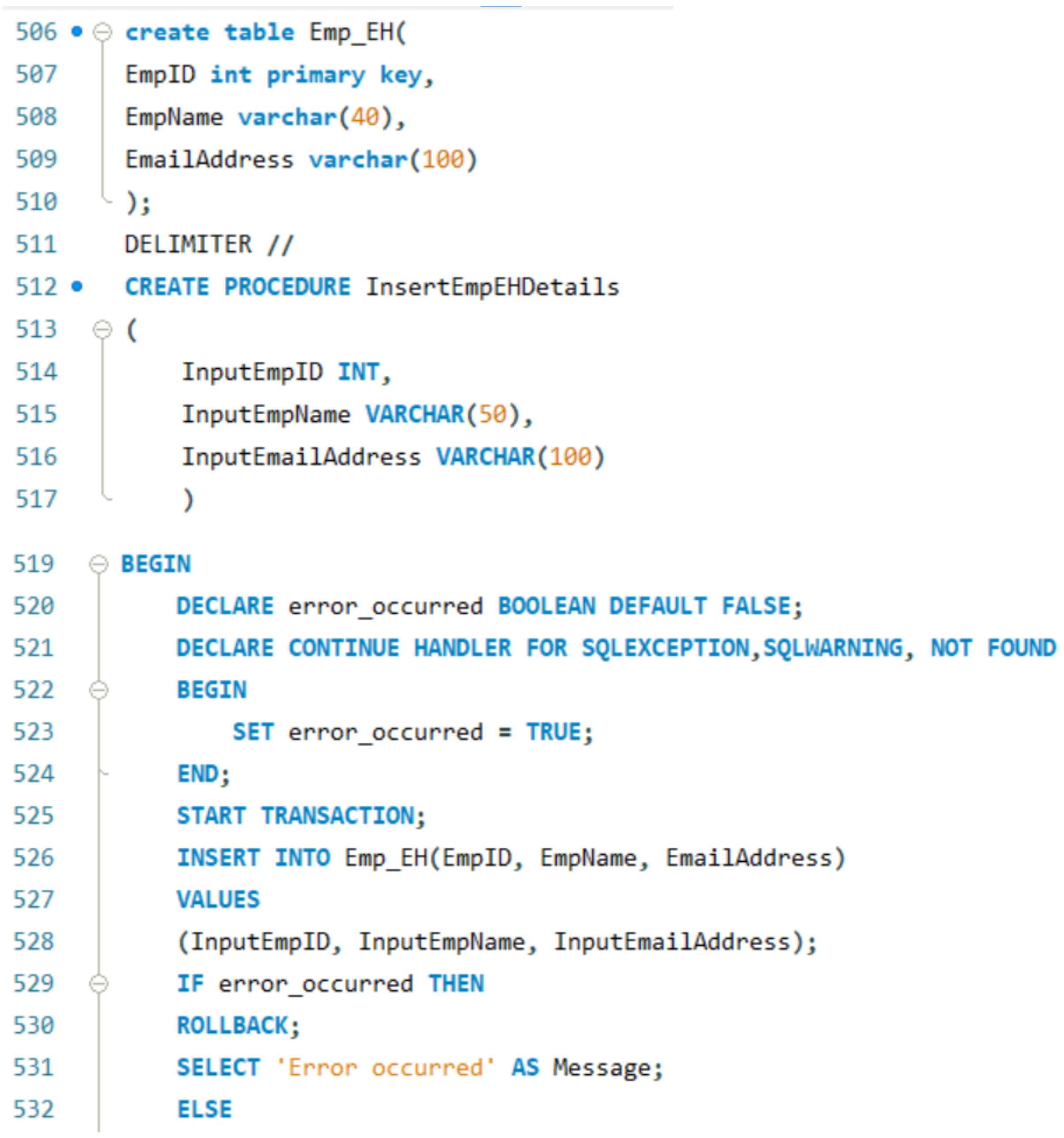
	productline	Total
▶	Classic Cars	24
	Motorcycles	6
	Planes	5
	Ships	1
	Trains	1
	Trucks and Buses	7
	Vintage Cars	10

Day 14

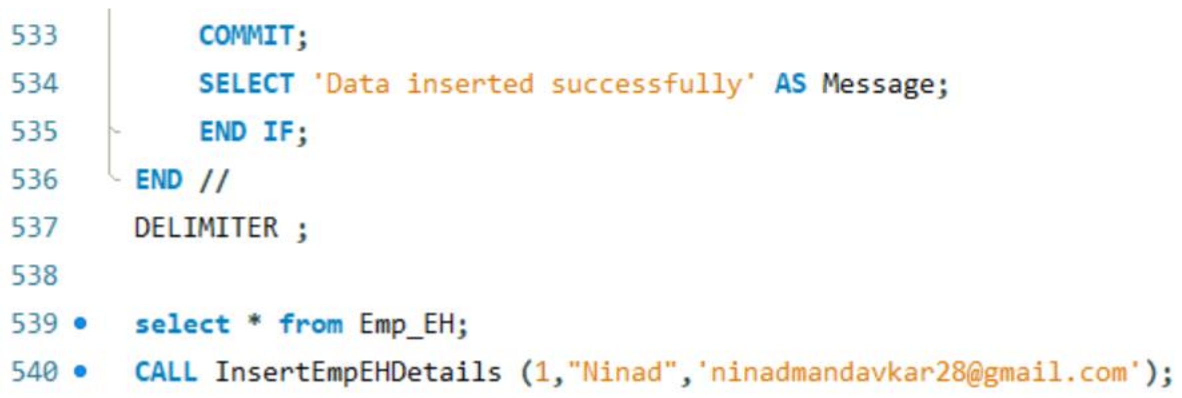
Create the table Emp_EH. Below are its fields.

- EmpID (Primary Key)
- EmpName
- EmailAddress

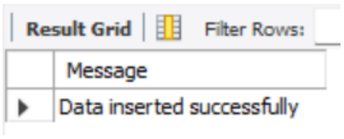
Create a procedure to accept the values for the columns in Emp_EH. Handle the error using exception handling concept. Show the message as “Error occurred” in case of anything wrong.



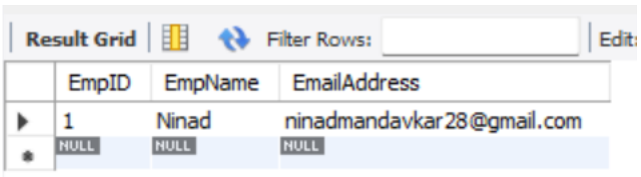
```
create table Emp_EH(
EmpID int primary key,
EmpName varchar(40),
EmailAddress varchar(100)
);
DELIMITER //
CREATE PROCEDURE InsertEmpEHDetails
(
    InputEmpID INT,
    InputEmpName VARCHAR(50),
    InputEmailAddress VARCHAR(100)
    )

BEGIN
    DECLARE error_occurred BOOLEAN DEFAULT FALSE;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION,SQLWARNING, NOT FOUND
    BEGIN
        SET error_occurred = TRUE;
    END;
    START TRANSACTION;
    INSERT INTO Emp_EH(EmpID, EmpName, EmailAddress)
    VALUES
    (InputEmpID, InputEmpName, InputEmailAddress);
    IF error_occurred THEN
    ROLLBACK;
    SELECT 'Error occurred' AS Message;
    ELSE
```



```
        COMMIT;
        SELECT 'Data inserted successfully' AS Message;
        END IF;
    END //
    DELIMITER ;

    select * from Emp_EH;
    CALL InsertEmpEHDetails (1,"Ninad",'ninadmandavkar28@gmail.com');
```

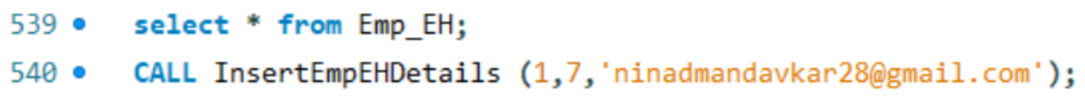


	Message
▶	Data inserted successfully

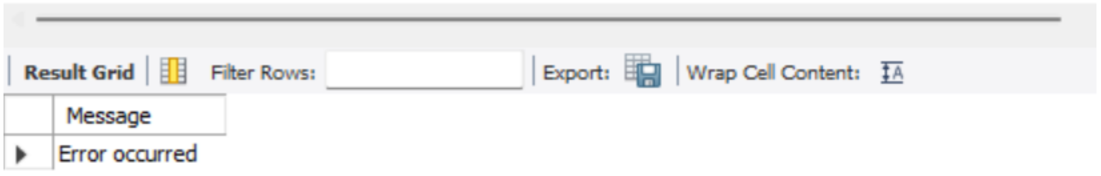


	EmpID	EmpName	EmailAddress
▶	1	Ninad	ninadmandavkar28@gmail.com
*	NULL	NULL	NULL

Now let us replace the Name “Ninad” in the argument by an Integer “7” while calling a procedure and check how it handles the error.

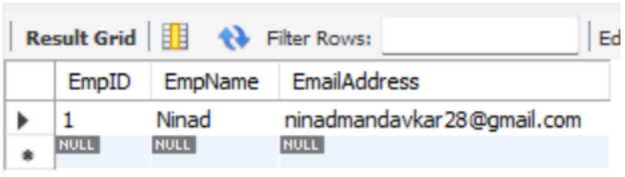


```
select * from Emp_EH;
CALL InsertEmpEHDetails (1,7,'ninadmandavkar28@gmail.com');
```



	Message
▶	Error occurred

Since we used a handler it will not affect the previous record in the table Emp_EH table and since it is Continuous handler it won’t exit the loop.



	EmpID	EmpName	EmailAddress
▶	1	Ninad	ninadmandavkar28@gmail.com
*	NULL	NULL	NULL

Day 15

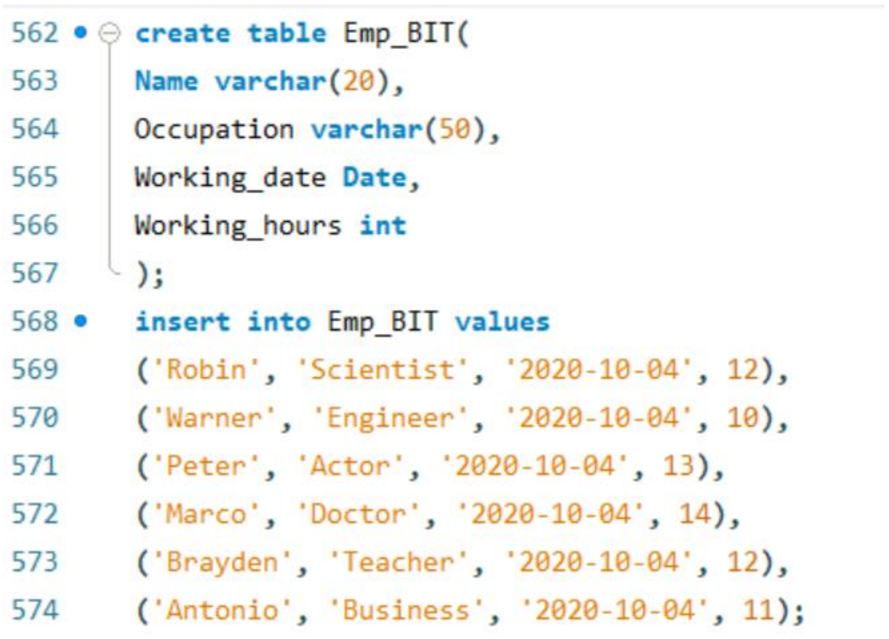
Create the table Emp_BIT. Add below fields in it.

- Name
- Occupation
- Working_date
- Working_hours

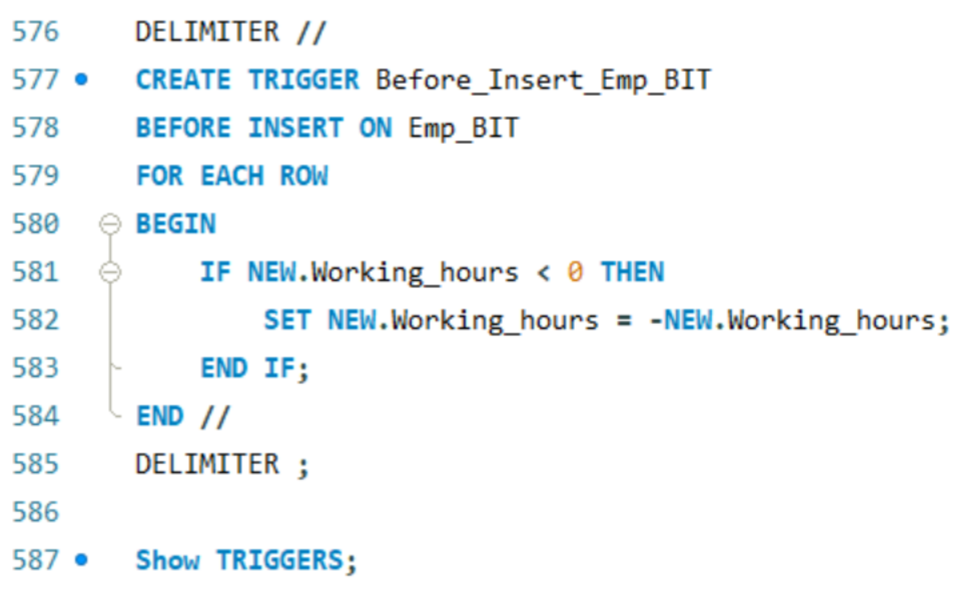
Insert the data as shown in below query.

```
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.

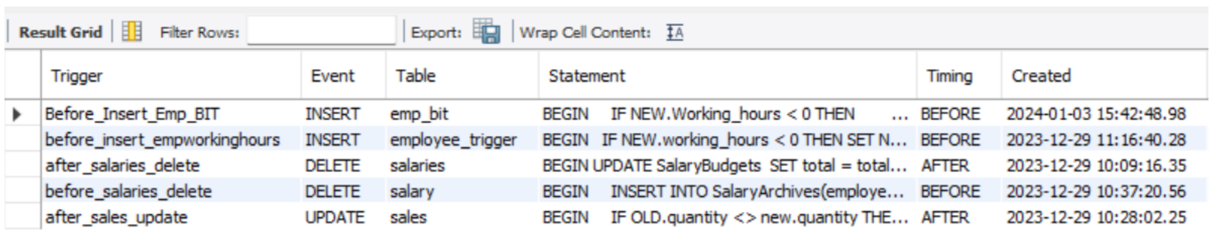


```
create table Emp_BIT(
Name varchar(20),
Occupation varchar(50),
Working_date Date,
Working_hours int
);
insert into Emp_BIT values
('Robin', 'Scientist', '2020-10-04', 12),
('Warner', 'Engineer', '2020-10-04', 10),
('Peter', 'Actor', '2020-10-04', 13),
('Marco', 'Doctor', '2020-10-04', 14),
('Brayden', 'Teacher', '2020-10-04', 12),
('Antonio', 'Business', '2020-10-04', 11);
```



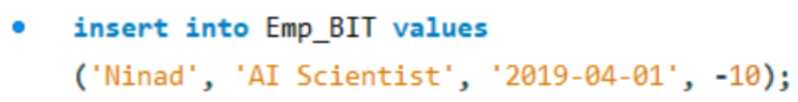
```sql
DELIMITER //
CREATE TRIGGER Before_Insert_Emp_BIT
BEFORE INSERT ON Emp_BIT
FOR EACH ROW
BEGIN
    IF NEW.Working_hours < 0 THEN
        SET NEW.Working_hours = -NEW.Working_hours;
    END IF;
END //
DELIMITER ;

Show TRIGGERS;
```

To view the Trigger information we use the query "Show Triggers" which will display us information of all the Triggers created.

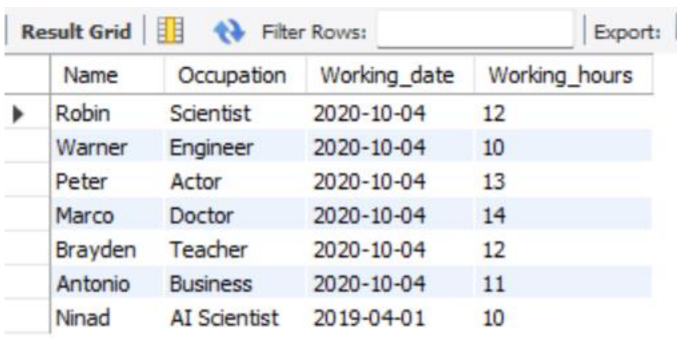


Trigger	Event	Table	Statement	Timing	Created
Before_Insert_Emp_BIT	INSERT	emp_bit	BEGIN IF NEW.Working_hours < 0 THEN ...	BEFORE	2024-01-03 15:42:48.98
before_insert_empworkinghours	INSERT	employee_trigger	BEGIN IF NEW.working_hours < 0 THEN SET N...	BEFORE	2023-12-29 11:16:40.28
after_salaries_delete	DELETE	salaries	BEGIN UPDATE SalaryBudgets SET total = total...	AFTER	2023-12-29 10:09:16.35
before_salaries_delete	DELETE	salary	BEGIN INSERT INTO SalaryArchives(employe...	BEFORE	2023-12-29 10:37:20.56
after_sales_update	UPDATE	sales	BEGIN IF OLD.quantity <> new.quantity THE...	AFTER	2023-12-29 10:28:02.25

Now to check the Trigger, let us insert a negative working hour "-10" for the record of Name=Ninad and check in the EMP_BIT table to see if positive values are inserted or not.



```sql
insert into Emp_BIT values
('Ninad', 'AI Scientist', '2019-04-01', -10);
```



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
Ninad	AI Scientist	2019-04-01	10

So apparently, the negative value of 10 is inserted as a positive one. This indicates Before_Insert Trigger is working properly.