**1,2)**

CREATE TABLE IF NOT EXISTS supplier(

SUPP\_ID INT PRIMARY KEY AUTO\_INCREMENT,

SUPP\_Name VARCHAR(50) NOT NULL,

SUPP\_CITY VARCHAR(50),

SUPP\_PHONE VARCHAR(10) NOT NULL

);

CREATE TABLE if NOT EXISTS customer(

CUS\_ID INT AUTO\_INCREMENT ,

CUS\_NAME VARCHAR(20) NOT NULL,

CUS\_PHONE VARCHAR(10) NULL,

CUS\_CITY VARCHAR(30) NOT NULL,

CUS\_GENDER CHAR,

PRIMARY KEY(CUS\_ID)

);

CREATE TABLE if NOT EXISTS category(

CAT\_ID INT AUTO\_INCREMENT ,

CAT\_NAME VARCHAR(20) NOT NULL,

PRIMARY KEY (CAT\_ID)

);

CREATE TABLE if NOT EXISTS product(

PRO\_ID INT AUTO\_INCREMENT,

PRO\_NAME VARCHAR(20) NOT NULL DEFAULT('Dummy'),

PRO\_DESC VARCHAR(60),

CAT\_ID INT,

PRIMARY KEY (PRO\_ID),

FOREIGN KEY(CAT\_ID) REFERENCES category(CAT\_ID)

);

CREATE TABLE if NOT EXISTS supplier\_pricing(

PRICING\_ID INT AUTO\_INCREMENT,

PRO\_ID INT,

SUPP\_ID INT,

SUPP\_PRICE INT DEFAULT 0,

PRIMARY KEY(PRICING\_ID),

FOREIGN KEY (PRO\_ID) REFERENCES product(PRO\_ID),

FOREIGN KEY (SUPP\_ID) REFERENCES supplier(SUPP\_ID)

);

CREATE TABLE if NOT EXISTS `order`(

ORD\_ID INT AUTO\_INCREMENT ,

ORD\_AMOUNT INT NOT NULL,

ORD\_DATE DATE NOT NULL ,

CUS\_ID INT ,

PRICING\_ID INT ,

PRIMARY KEY(ORD\_ID),

FOREIGN KEY(CUS\_ID) REFERENCES customer(CUS\_ID),

FOREIGN KEY(PRICING\_ID) REFERENCES supplier\_pricing(PRICING\_ID)

);

CREATE TABLE if NOT EXISTS rating(

RAT\_ID INT NOT NULL AUTO\_INCREMENT ,

ORD\_ID INT NOT NULL,

RAT\_RATSTARS INT NOT NULL ,

PRIMARY KEY(RAT\_ID),

FOREIGN KEY(ORD\_ID) REFERENCES `order` (ORD\_ID)

);

**3)**

INSERT INTO supplier(SUPP\_NAME,SUPP\_CITY,SUPP\_PHONE)

VALUES

("Rajesh Retails","Delhi",'1234567890'),

("Appario Ltd.","Mumbai",'2589631470'),

("Knome products","Banglore",'9785462315'),

("Bansal Retails","Kochi",'8975463285'),

("Mittal Ltd.","Lucknow",'7898456532');

INSERT INTO customer(CUS\_NAME,CUS\_PHONE,CUS\_CITY,CUS\_GENDER) VALUES

("AAKASH",'9999999999',"DELHI",'M'),

("AMAN",'9785463215',"NOIDA",'M'),

("NEHA",'9999999999',"MUMBAI",'F'),

("MEGHA",'9994562399',"KOLKATA",'F'),

("PULKIT",'7895999999',"LUCKNOW",'M');

INSERT INTO category(CAT\_NAME) VALUES("BOOKS");

INSERT INTO category(CAT\_NAME) VALUES("GAMES");

INSERT INTO category(CAT\_NAME) VALUES("GROCERIES");

INSERT INTO category(CAT\_NAME) VALUES ("ELECTRONICS");

INSERT INTO category(CAT\_NAME) VALUES("CLOTHES");

INSERT INTO PRODUCT VALUES(1,"GTA V","Windows 7 and above with i5 processor and 8GB RAM",2);

INSERT INTO PRODUCT VALUES(2,"TSHIRT","SIZE-L with Black, Blue and White variations",5);

INSERT INTO PRODUCT VALUES(3,"ROG LAPTOP","Windows 10 with 15inch screen, i7 processor, 1TB SSD",4);

INSERT INTO PRODUCT VALUES(4,"OATS","Highly Nutritious from Nestle",3);

INSERT INTO PRODUCT VALUES(5,"HARRY POTTER","Best Collection of all time by J.K Rowling",1);

INSERT INTO PRODUCT VALUES(6,"MILK","1L Toned MIlk",3);

INSERT INTO PRODUCT VALUES(7,"Boat EarPhones","1.5Meter long Dolby Atmos",4);

INSERT INTO PRODUCT VALUES(8,"Jeans","Stretchable Denim Jeans with various sizes and color",5);

INSERT INTO PRODUCT VALUES(9,"Project IGI","compatible with windows 7 and above",2);

INSERT INTO PRODUCT VALUES(10,"Hoodie","Black GUCCI for 13 yrs and above",5);

INSERT INTO PRODUCT VALUES(11,"Rich Dad Poor Dad","Written by RObert Kiyosaki",1);

INSERT INTO PRODUCT VALUES(12,"Train Your Brain","By Shireen Stephen",1);

INSERT INTO supplier\_pricing(PRO\_ID,SUPP\_ID,SUPP\_PRICE)

VALUES((SELECT PRO\_ID FROM PRODUCT WHERE PRO\_NAME='ROG LAPTOP'),(SELECT SUPP\_ID FROM supplier WHERE SUPP\_PHONE='7898456532'),30000),

((SELECT PRO\_ID FROM PRODUCT WHERE PRO\_NAME='GTA V'),2,1500),

(3,5,30000),

(5,1,3000),

(2,3,2500),

(4,1,1000),

(12,2,780),

(12,4,789),

(3,1,31000),

(1,5,1450),

(4,2,999),

(7,3,549),

(7,4,529),

(6,2,105),

(6,1,99),

(2,5,2999),

(5,2,2999);

INSERT INTO `ORDER`(ORD\_ID,ORD\_AMOUNT,ORD\_DATE, CUS\_ID, PRICING\_ID)

VALUES

(101,1500,"2021-10-06",(SELECT CUS\_ID FROM CUSTOMER WHERE CUS\_PHONE='9785463215'),1),

(102,1000,"2021-10-12",3,5),

(103,30000,"2021-09-16",5,2),

(104,1500,"2021-10-05",1,1),

(105,3000,"2021-08-16",4,3),

(106,1450,"2021-08-18",1,9),

(107,789,"2021-09-01",3,7),

(108,780,"2021-09-07",5,6),

(109,3000,"2021-09-10",5,3),

(110,2500,"2021-09-10",2,4),

(111,1000,"2021-09-15",4,5),

(112,789,"2021-09-16",4,7),

(113,31000,"2021-09-16",1,8),

(114,1000,"2021-09-16",3,5),

(115,3000,"2021-09-16",5,3),

(116,99,"2021-09-17",2,14);

INSERT INTO rating(ORD\_ID,RAT\_RATSTARS)

VALUES

(101,4),

(102,3),

(103,1),

(104,2),

(105,4),

(106,3),

(107,4),

(108,4),

(109,3),

(110,5),

(111,3),

(112,4),

(113,2),

(114,1),

(115,1),

(116,0);

**4) Display the total number of customers based on gender who have placed orders of worth at least Rs.3000.**

#Identify tables [order]

SELECT \* FROM `ORDER`;

#Identify tables [order,CUSTOMER] AND APPLY JOIN

SELECT \* FROM `ORDER` AS O

INNER JOIN CUSTOMER AS C

ON O.CUS\_ID = C.CUS\_ID;

# THOSE CUSTOMERS who have placed orders of worth at least Rs.3000

SELECT \* FROM `ORDER` AS O

INNER JOIN CUSTOMER AS C

ON O.CUS\_ID = C.CUS\_ID

HAVING O.ORD\_AMOUNT>=3000;

# REFINE COLUMNS ONLY RELEVANT

SELECT C.CUS\_ID,CUS\_NAME,CUS\_GENDER,O.ORD\_ID,O.ORD\_AMOUNT FROM `ORDER` AS O

INNER JOIN CUSTOMER AS C

ON O.CUS\_ID = C.CUS\_ID

HAVING O.ORD\_AMOUNT>=3000;

# SORT GENDER COLUMN WE HAVE ONE F AND 3 MALE SO COUNT CUS\_GENDER

SELECT COUNT(TEST.CUS\_GENDER) AS NoOfCustomer, TEST.CUS\_GENDER FROM

(

SELECT C.CUS\_ID,CUS\_NAME,CUS\_GENDER,O.ORD\_ID,O.ORD\_AMOUNT FROM `ORDER` AS O

INNER JOIN CUSTOMER AS C

ON O.CUS\_ID = C.CUS\_ID

HAVING O.ORD\_AMOUNT>=3000

) AS TEST

GROUP BY TEST.CUS\_GENDER;

SELECT COUNT(t2.cus\_gender) AS NoOfCustomers, t2.cus\_gender FROM

(

SELECT t1.cus\_id, t1.cus\_gender, t1.ord\_amount, t1.cus\_name FROM

(

SELECT `order`.\*, customer.cus\_gender, customer.cus\_name FROM `order`

INNER JOIN customer

WHERE `order`.cus\_id = customer.cus\_id

HAVING `order`.ord\_amount >= 3000

) AS t1

) AS t2 GROUP BY t2.cus\_gender;

**5) Display all the orders along with product name ordered by a customer having Customer\_Id=2**

# Identfiy tables [Order, supplier\_pricing]

SELECT O.ORD\_ID,O.ORD\_DATE,O.ORD\_AMOUNT,O.CUS\_ID, SP.PRO\_ID FROM `order` AS O

INNER JOIN supplier\_pricing AS SP

ON O.PRICING\_ID=SP.PRICING\_ID;

# Identfiy tables [Order, supplier\_pricing,PRODUCT]

# APPLY JOIN BETWEEN PRODUCT AND ABOVE RESULT

SELECT \* FROM PRODUCT AS P

INNER JOIN

(

SELECT O.ORD\_ID,O.ORD\_DATE,O.ORD\_AMOUNT,O.CUS\_ID, SP.PRO\_ID FROM `order` AS O

INNER JOIN supplier\_pricing AS SP

ON O.PRICING\_ID=SP.PRICING\_ID

) AS T1 ON T1.PRO\_ID = P.PRO\_ID;

# REFINE RELEVENT COLUMN ONLY

SELECT T1.ORD\_ID,T1.ORD\_DATE,T1.ORD\_AMOUNT,T1.CUS\_ID, P.PRO\_ID FROM PRODUCT AS P

INNER JOIN

(

SELECT O.ORD\_ID,O.ORD\_DATE,O.ORD\_AMOUNT,O.CUS\_ID, SP.PRO\_ID FROM `order` AS O

INNER JOIN supplier\_pricing AS SP

ON O.PRICING\_ID=SP.PRICING\_ID

) AS T1 ON T1.PRO\_ID = P.PRO\_ID;

# APPLY JOIN WITH CUSTOMER TABLE

SELECT C.CUS\_ID,C.CUS\_NAME, T2.ORD\_ID,T2.ORD\_DATE,T2.ORD\_AMOUNT, T2.PRO\_NAME FROM CUSTOMER AS C

INNER JOIN

(

SELECT T1.ORD\_ID,T1.ORD\_DATE,T1.ORD\_AMOUNT,T1.CUS\_ID, P.PRO\_ID , P.PRO\_NAME FROM PRODUCT AS P

INNER JOIN

(

SELECT O.ORD\_ID,O.ORD\_DATE,O.ORD\_AMOUNT,O.CUS\_ID, SP.PRO\_ID FROM `order` AS O

INNER JOIN supplier\_pricing AS SP

ON O.PRICING\_ID=SP.PRICING\_ID

) AS T1 ON T1.PRO\_ID = P.PRO\_ID

) AS T2 ON T2.CUS\_ID = C.CUS\_ID;

# DISPLAY ONLY THAT CUSTOMER WHOSE ID =2

SELECT C.CUS\_ID,C.CUS\_NAME, T2.ORD\_ID,T2.ORD\_DATE,T2.ORD\_AMOUNT, T2.PRO\_NAME FROM CUSTOMER AS C

INNER JOIN

(

SELECT T1.ORD\_ID,T1.ORD\_DATE,T1.ORD\_AMOUNT,T1.CUS\_ID, P.PRO\_ID , P.PRO\_NAME FROM PRODUCT AS P

INNER JOIN

(

SELECT O.ORD\_ID,O.ORD\_DATE,O.ORD\_AMOUNT,O.CUS\_ID, SP.PRO\_ID FROM `order` AS O

INNER JOIN supplier\_pricing AS SP

ON O.PRICING\_ID=SP.PRICING\_ID

) AS T1 ON T1.PRO\_ID = P.PRO\_ID

) AS T2 ON T2.CUS\_ID = C.CUS\_ID AND C.CUS\_ID=2;

**6) Display the Supplier details of who is supplying more than one product.**

# Identfiy tables supplier\_pricing

select \* from supplier\_pricing;

# sort the supp\_id column we found that each supplier having more than one pro\_id so we have

SELECT supp\_id FROM supplier\_pricing GROUP BY supp\_id HAVING COUNT(supp\_id) > 1;

# Now Join above result with supplier table

SELECT supplier.\* FROM supplier

WHERE supplier.supp\_id IN (SELECT supp\_id FROM supplier\_pricing GROUP BY supp\_id HAVING COUNT(supp\_id) > 1)

GROUP BY supplier.supp\_id;

**7) Find the least expensive product from each category and print the table with category id, name, product and price of the product**

# Identify tables[supplier\_pricing]

SELECT \* FROM supplier\_pricing;

# sort the PRO\_ID column, we found that for each pro\_id we have different supplier with different price so find least expensive

select PRO\_ID, min(SUPP\_PRICE) AS Min\_Price from supplier\_pricing GROUP BY PRO\_ID;

#APPLY JOIN BETWEEN PRODUCT AND ABOVE RESULT

SELECT \* FROM PRODUCT

INNER JOIN

(

select PRO\_ID, min(SUPP\_PRICE) AS Min\_Price from supplier\_pricing GROUP BY PRO\_ID

) AS T1 ON PRODUCT.PRO\_ID = T1.PRO\_ID;

#APPLY JOIN BETWEEN CATEGORY AND ABOVE RESULT

SELECT \* FROM CATEGORY

INNER JOIN

(

SELECT \* FROM PRODUCT

INNER JOIN

(

select PRO\_ID, min(SUPP\_PRICE) AS Min\_Price from supplier\_pricing GROUP BY PRO\_ID

) AS T1 ON PRODUCT.PRO\_ID = T1.PRO\_ID

) AS T2 ON T2.CAT\_ID= CATEGORY.CAT\_ID;

# IT GIVES DUPLICATE COLUMN ERRO PRO\_ID THERFORE WE HAVE TO REFINE T1 RESULT

SELECT \* FROM CATEGORY

INNER JOIN

(

SELECT PRODUCT.PRO\_ID, PRODUCT.PRO\_NAME,PRODUCT.CAT\_ID,T1.Min\_Price FROM PRODUCT

INNER JOIN

(

select PRO\_ID, min(SUPP\_PRICE) AS Min\_Price from supplier\_pricing GROUP BY PRO\_ID

) AS T1 ON PRODUCT.PRO\_ID = T1.PRO\_ID

) AS T2 ON T2.CAT\_ID= CATEGORY.CAT\_ID;

#WHEN WE SORT WE FOUND MORE THAN ONE CAT\_ID WITH DIFFRENT PRICE SO WE HAVE TO select MINIMUM PRICE FROM

# ABOVE RESULT AND JOIN WITH CATEGORY TABLE

SELECT category.cat\_id, category.cat\_name, MIN(t3.min\_price) AS Min\_Price FROM category

INNER JOIN

(

SELECT product.cat\_id, product.pro\_name, t2.\* FROM product

INNER JOIN

(

SELECT pro\_id, MIN(supp\_price) AS Min\_Price FROM supplier\_pricing GROUP BY pro\_id

) AS t2 WHERE t2.pro\_id = product.pro\_id

) AS t3 WHERE t3.cat\_id = category.cat\_id

GROUP BY t3.cat\_id;

SELECT category.cat\_id, category.cat\_name, MIN(t3.min\_price) AS Min\_Price FROM category

INNER JOIN

(

SELECT product.cat\_id, product.pro\_name, t2.\* FROM product

INNER JOIN

(

SELECT pro\_id, MIN(supp\_price) AS Min\_Price FROM supplier\_pricing GROUP BY pro\_id

) AS t2 WHERE t2.pro\_id = product.pro\_id

) AS t3 WHERE t3.cat\_id = category.cat\_id

GROUP BY t3.cat\_id;

**8) Display the Id and Name of the Product ordered after “2021-10-05”.**

# Identify the TABLES [order,supplier\_pricing]

SELECT \* FROM `order`;

# from above result we can see that pricing\_id is common between order table and supplier\_pricing table so apply join

SELECT \* FROM `order` AS o

INNER JOIN supplier\_pricing AS sp

ON o.pricing\_id = sp.pricing\_id;

# Now Apply join between Product table and test table

SELECT test.pro\_id, p.pro\_name FROM product as p

INNER JOIN (

SELECT \* FROM `order` AS o

INNER JOIN supplier\_pricing AS sp

ON o.pricing\_id = sp.pricing\_id

) as test

on test.pro\_id= p.pro\_id;

# it will give name duplicate column name pricing\_id

SELECT test.pro\_id, p.pro\_name FROM product as p

INNER JOIN (

SELECT o.ord\_id, o.ord\_date,sp.pro\_id FROM `order` AS o

INNER JOIN supplier\_pricing AS sp

ON o.pricing\_id = sp.pricing\_id

) as test

on test.pro\_id= p.pro\_id;

SELECT product.pro\_id, product.pro\_name FROM `order`

INNER JOIN supplier\_pricing

ON supplier\_pricing.pricing\_id = `order`.pricing\_id

INNER JOIN product

ON product.pro\_id = supplier\_pricing.pro\_id

WHERE `order`.ord\_date > '2021-10-05';

**9) Display customer name and gender whose names start or end with character 'A'.**

SELECT \* FROM customer AS c;

# Exact Match

SELECT \* FROM customer AS c WHERE c.CUS\_NAME = 'MEGHA';

# Partial Match

# Names starting with A or names ending with c.

SELECT c.CUS\_NAME,c.CUS\_GENDER FROM customer AS c WHERE (c.CUS\_NAME LIKE 'A%' OR c.CUS\_NAME LIKE '%A');

# Name starting with A and ending with N

SELECT \* FROM customer AS c WHERE c.CUS\_NAME LIKE 'A%N';

**10) Create a stored procedure to display supplier id, name, Rating(Average rating of all the products sold by every customer) and Type\_of\_Service. For Type\_of\_Service, If rating =5, print “Excellent Service”,If rating >4 print “Good Service”, If rating >2 print “Average Service” else print “Poor Service”. Note that there should be one rating per supplier.**

# IDENTIFIED the TABLE [ rating, `ORDER`] and apply join

select \* from `order` as o

inner join rating as r

on o.ORD\_ID = r.ORD\_ID;

# fetch only relevant COLUMNS

select o.PRICING\_ID,r.ORD\_ID,r.RAT\_RATSTARS from `order` as o

inner join

rating as r

on o.ORD\_ID = r.ORD\_ID;

# Join between supplier\_pricing and test TABLE

select \* from supplier\_pricing as sp

inner join (

select o.PRICING\_ID,r.ORD\_ID,r.RAT\_RATSTARS from `order` as o

inner join

rating as r

on o.ORD\_ID = r.ORD\_ID) as test

on test.PRICING\_ID = sp.PRICING\_ID;

# taking relevant COLUMNS only

select sp.SUPP\_ID,test.ord\_id,test.rat\_ratstars from supplier\_pricing as sp

inner join (

select o.PRICING\_ID,r.ORD\_ID,r.RAT\_RATSTARS from `order` as o

inner join

rating as r

on o.ORD\_ID = r.ORD\_ID) as test

on test.PRICING\_ID = sp.PRICING\_ID;

# TAKING Average RAT\_RATSTARS AS Average

select test2.supp\_id, sum(test2.rat\_ratstars)/count(test2.rat\_ratstars) as Average from

(select sp.SUPP\_ID,test.ord\_id,test.rat\_ratstars from supplier\_pricing as sp

inner join (

select o.PRICING\_ID,r.ORD\_ID,r.RAT\_RATSTARS from `order` as o

inner join

rating as r

on o.ORD\_ID = r.ORD\_ID) as test

on test.PRICING\_ID = sp.PRICING\_ID) AS test2

group by test2.supp\_id;

# Apply Join between final and supplier TABLE

select final.supp\_id, supplier.SUPP\_Name, final.Average from

(select test2.supp\_id, sum(test2.rat\_ratstars)/count(test2.rat\_ratstars) as Average from

(select sp.SUPP\_ID,test.ord\_id,test.rat\_ratstars from supplier\_pricing as sp

inner join (

select o.PRICING\_ID,r.ORD\_ID,r.RAT\_RATSTARS from `order` as o

inner join

rating as r

on o.ORD\_ID = r.ORD\_ID) as test

on test.PRICING\_ID = sp.PRICING\_ID) AS test2

group by test2.supp\_id) as final

inner join supplier

on supplier.SUPP\_ID = final.supp\_id;

#Apply CASE

SELECT report.supp\_id, report.supp\_name, report.Average,

CASE

WHEN report.Average = 5 THEN 'Excellent Service'

WHEN report.Average > 4 THEN 'Good Service'

WHEN report.Average > 2 THEN 'Average Service'

ELSE 'Poor Service'

END AS Type\_of\_Service FROM

(

SELECT final.supp\_id, supplier.supp\_name, final.Average FROM

(

SELECT test2.supp\_id,SUM(test2.rat\_ratstars) / COUNT(test2.rat\_ratstars) AS Average FROM

(

SELECT supplier\_pricing.supp\_id, test.ORD\_ID, test.RAT\_RATSTARS FROM supplier\_pricing

INNER JOIN

(

SELECT `order`.pricing\_id, rating.ORD\_ID, rating.RAT\_RATSTARS FROM `order`

INNER JOIN rating

ON rating.`ord\_id` = `order`.ord\_id

) AS test

ON test.pricing\_id = supplier\_pricing.pricing\_id

) AS test2 GROUP BY supplier\_pricing.supp\_id

) AS final

INNER JOIN supplier

ON final.supp\_id = supplier.supp\_id

) AS report;

# Create Stored PROCEDURE

CREATE DEFINER=`root`@`localhost` PROCEDURE `SUP\_RATTINGS`()

BEGIN

SELECT report.supp\_id, report.supp\_name, report.Average,

CASE

WHEN report.Average = 5 THEN 'Excellent Service'

WHEN report.Average > 4 THEN 'Good Service'

WHEN report.Average > 2 THEN 'Average Service'

ELSE 'Poor Service'

END AS Type\_of\_Service FROM

(

SELECT final.supp\_id, supplier.supp\_name, final.Average FROM

(

SELECT test2.supp\_id,SUM(test2.rat\_ratstars) / COUNT(test2.rat\_ratstars) AS Average FROM

(

SELECT supplier\_pricing.supp\_id, test.ORD\_ID, test.RAT\_RATSTARS FROM supplier\_pricing

INNER JOIN

(

SELECT `order`.pricing\_id, rating.ORD\_ID, rating.RAT\_RATSTARS FROM `order`

INNER JOIN rating

ON rating.`ord\_id` = `order`.ord\_id

) AS test

ON test.pricing\_id = supplier\_pricing.pricing\_id

) AS test2 GROUP BY supplier\_pricing.supp\_id

) AS final

INNER JOIN supplier

ON final.supp\_id = supplier.supp\_id

) AS report;

END

# calling Stored PROCEDURE

call ecom\_db.SUP\_RATTINGS();