Graded Lab 4 – DBMS Solution

1. You are required to create tables for supplier, customer, category, product, supplier\_pricing, order, rating to store the data for the E-commerce.
2. You are required to develop SQL based programs (Queries) to facilitate the Admin team of the E-Commerce company to retrieve the data in summarized format - The Data Retrieval needs are described below.



CREATE TABLE supplier (

SUPP\_ID INT PRIMARY KEY,

SUPP\_NAME VARCHAR(50) NOT NULL,

SUPP\_CITY VARCHAR(50) NOT NULL,

SUPP\_PHONE VARCHAR(50) NOT NULL

);

CREATE TABLE customer (

CUS\_ID INT PRIMARY KEY,

CUS\_NAME VARCHAR(20) NOT NULL,

CUS\_PHONE VARCHAR(10) NOT NULL,

CUS\_CITY VARCHAR(30) NOT NULL,

CUS\_GENDER CHAR

);

CREATE TABLE category (

CAT\_ID INT PRIMARY KEY,

CAT\_NAME VARCHAR(20) NOT NULL

);

CREATE TABLE product (

PRO\_ID INT PRIMARY KEY,

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

PRO\_DESC VARCHAR(60),

CAT\_ID INT,

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

);

CREATE TABLE supplier\_pricing (

PRICING\_ID INT PRIMARY KEY,

PRO\_ID INT,

SUPP\_ID INT,

SUPP\_PRICE INT DEFAULT 0,

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

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

);

CREATE TABLE `order` (

ORD\_ID INT PRIMARY KEY,

ORD\_AMOUNT INT NOT NULL,

ORD\_DATE DATE NOT NULL,

CUS\_ID INT,

PRICING\_ID INT,

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

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

);

CREATE TABLE rating (

RAT\_ID INT PRIMARY KEY,

ORD\_ID INT,

RAT\_RATSTARS INT NOT NULL,

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

);

1. Insert the data



-- Supplier Data

INSERT INTO supplier VALUES

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

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

(3, 'Knome products', 'Bangalore', '9785462315'),

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

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

-- Customer Data

INSERT INTO customer VALUES

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

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

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

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

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

-- Category Data

INSERT INTO category VALUES

(1, 'BOOKS'),

(2, 'GAMES'),

(3, 'GROCERIES'),

(4, 'ELECTRONICS'),

(5, 'CLOTHES');

-- Product Data

INSERT INTO product VALUES

(1, 'GTA V', 'Windows 7 and above with i5 processor and 8GB RAM', 2),

(2, 'TSHIRT', 'SIZE-L with Black, Blue and White variations', 5),

(3, 'ROG LAPTOP', 'Windows 10 with 15inch screen, i7 processor, 1TB SSD', 4),

(4, 'OATS', 'Highly Nutritious from Nestle', 3),

(5, 'HARRY POTTER', 'Best Collection of all time by J.K Rowling', 1),

(6, 'MILK', '1L Toned Milk', 3),

(7, 'Boat Earphones', '1.5Meter long Dolby Atmos', 4),

(8, 'Jeans', 'Stretchable Denim Jeans with various sizes and color', 5),

(9, 'Project IGI', 'Compatible with windows 7 and above', 2),

(10, 'Hoodie', 'Black GUCCI for 13 yrs and above', 5),

(11, 'Rich Dad Poor Dad', 'Written by Robert Kiyosaki', 1),

(12, 'Train Your Brain', 'By Shireen Stephen', 1);

-- Supplier Pricing Data

INSERT INTO supplier\_pricing VALUES

(1, 1, 2, 1500),

(2, 3, 5, 30000),

(3, 5, 1, 3000),

(4, 2, 3, 2500),

(5, 4, 1, 1000),

(6, 12, 2, 780),

(7, 12, 4, 789),

(8, 3, 1, 31000),

(9, 1, 5, 1450),

(10, 4, 2, 999),

(11, 7, 3, 549),

(12, 7, 4, 529),

(13, 6, 2, 105),

(14, 6, 1, 99),

(15, 2, 5, 2999),

(16, 5, 2, 2999);

-- Order Data

INSERT INTO `order` VALUES

(101, 1500, '2021-10-06', 2, 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, '2021-10-10', 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);

-- Rating Data

INSERT INTO rating VALUES

(1, 101, 4),

(2, 102, 3),

(3, 103, 1),

(4, 104, 2),

(5, 105, 4),

(6, 106, 3),

(7, 107, 4),

(8, 108, 4),

(9, 109, 3),

(10, 110, 5),

(11, 111, 3),

(12, 112, 4),

(13, 113, 2),

(14, 114, 1),

(15, 115, 1),

(16, 116, 0);

1. Display the total number of customers based on gender who have placed individual orders of worth at least Rs.3000.



SELECT

CUS\_GENDER, COUNT(DISTINCT c.CUS\_ID) AS Customer\_Count

FROM

`order` o

JOIN customer c ON o.CUS\_ID = c.CUS\_ID

WHERE

ORD\_AMOUNT >= 3000

GROUP BY

CUS\_GENDER;

1. Display all the orders along with product name ordered by a customer having Customer\_Id=2



SELECT

o.ORD\_ID AS Order\_ID,

o.ORD\_DATE AS Order\_Date,

p.PRO\_NAME AS Product\_Name

FROM

`order` o

JOIN supplier\_pricing sp ON o.PRICING\_ID = sp.PRICING\_ID

JOIN product p ON sp.PRO\_ID = p.PRO\_ID

WHERE

o.CUS\_ID = 2;

1. Display the Supplier details who can supply more than one product.



SELECT

s.SUPP\_ID, s.SUPP\_NAME, COUNT(DISTINCT sp.PRO\_ID) AS Product\_Count

FROM

supplier s

JOIN supplier\_pricing sp ON s.SUPP\_ID = sp.SUPP\_ID

GROUP BY

s.SUPP\_ID, s.SUPP\_NAME

HAVING

COUNT(DISTINCT sp.PRO\_ID) > 1;

1. Find the least expensive product from each category and print the table with category id, name, product name and price of the product



SELECT

c.CAT\_ID, c.CAT\_NAME, p.PRO\_NAME, MIN(sp.SUPP\_PRICE) AS Lowest\_Price

FROM

category c

JOIN product p ON c.CAT\_ID = p.CAT\_ID

JOIN supplier\_pricing sp ON p.PRO\_ID = sp.PRO\_ID

GROUP BY

c.CAT\_ID, c.CAT\_NAME, p.PRO\_NAME

ORDER BY

Lowest\_Price ASC;

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



SELECT

DISTINCT p.PRO\_ID,

p.PRO\_NAME

FROM

`order` o

JOIN supplier\_pricing sp ON o.PRICING\_ID = sp.PRICING\_ID

JOIN product p ON sp.PRO\_ID = p.PRO\_ID

WHERE

o.ORD\_DATE > '2021-10-05';

1. Display customer name and gender whose names start or end with character 'A'.



SELECT

CUS\_NAME,

CUS\_GENDER

FROM

customer

WHERE

CUS\_NAME LIKE 'A%' OR CUS\_NAME LIKE '%A';

1. 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.



DELIMITER //

CREATE PROCEDURE GetSupplierRatings()

BEGIN

SELECT

s.SUPP\_ID,

s.SUPP\_NAME,

AVG(r.RAT\_RATSTARS) AS Avg\_Rating,

CASE

WHEN AVG(r.RAT\_RATSTARS) = 5 THEN 'Excellent Service'

WHEN AVG(r.RAT\_RATSTARS) > 4 THEN 'Good Service'

WHEN AVG(r.RAT\_RATSTARS) > 2 THEN 'Average Service'

ELSE 'Poor Service'

END AS Type\_of\_Service

FROM

supplier s

JOIN supplier\_pricing sp ON s.SUPP\_ID = sp.SUPP\_ID

JOIN `order` o ON sp.PRICING\_ID = o.PRICING\_ID

JOIN rating r ON o.ORD\_ID = r.ORD\_ID

GROUP BY

s.SUPP\_ID, s.SUPP\_NAME;

END //

DELIMITER ;

CALL GetSupplierRatings();