PROJECT SUBMISSION 4

Relational Schema:

```
CREATE TABLE user(
      user_id INTEGER PRIMARY KEY auto_increment,
    user_name VARCHAR(50) NOT NULL,
    user_age INTEGER NOT NULL,
    user_phone VARCHAR(50),
    user_email VARCHAR(50) NOT NULL UNIQUE,
    user_password VARCHAR(50) NOT NULL UNIQUE
    );

INSERT INTO user VALUES(NULL,"Ankit", 18, "9800000234", "ankit@gmail.com",
"ankit1234");
INSERT INTO user VALUES(NULL,"Ankita", 19, "9811000234",
"ankita121@gmail.com", "1234a");
INSERT INTO user VALUES(NULL,"Anu", 17, "9810400223", "anu@gmail.com",
"anuuu234");
INSERT INTO user VALUES(NULL,"Shiv", 19, "9845239234", "shiv@gmail.com",
"s34");
INSERT INTO user VALUES(NULL,"Shivam", 21, "9222229234",
"shivam@hotmail.com", "srrrr");
INSERT INTO user VALUES(NULL,"Om", 16, "9845255555", "om@gmail.com",
"swdhiw");
INSERT INTO user VALUES(NULL,"Gauri", 22, "980012124", "gauri@gmail.com",
"g34");
INSERT INTO user VALUES(NULL,"Vinayak", 19, "9080897234",
"vinayak@gmail.com", "sv4");
INSERT INTO user VALUES(NULL,"Kartikeya", 18, "9808639234",
"kartikeya@gmail.com", "k34");
INSERT INTO user VALUES(NULL,"Renu", 21, "91010239234", "renu@gmail.com",
"rrr34");
SELECT * FROM user;

CREATE TABLE product(
      product_id INTEGER PRIMARY KEY auto_increment,
    product_name VARCHAR(50) NOT NULL,
    brand_name VARCHAR(50) NOT NULL,
    product_specifications VARCHAR(150),
    product_price INTEGER NOT NULL,
    product_qty INTEGER NOT NULL,
    product_discount INTEGER ,
     category_id INTEGER NOT NULL REFERENCES category(category_id),
     vendor_id INTEGER NOT NULL REFERENCES vendor(vendor_id)
    );

INSERT INTO product VALUES(NULL,"Pen",  "Cello", "smooth flow",60,1,5,
11,2);
INSERT INTO product VALUES(NULL,"Highlighter",  "Clearpoint",
"Fluorescent",65,1,0, 11,2);
INSERT INTO product VALUES(NULL,"Pencil",  "Faber-Castell", "HB",20,1,0,
11,4);
INSERT INTO product VALUES(NULL,"Notebook", "Moleskine", "Plain pages",
150, 1, 10, 11,5);
INSERT INTO product VALUES(NULL,"Eraser", "Pentel", "Soft", 30, 1, 2,
11,1);
INSERT INTO product VALUES(NULL,"Ruler", "Westcott", "12-inch", 40, 1, 3,
11,9);
INSERT INTO product VALUES(NULL,"Binder", "Avery", "3-inch", 200, 1, 8,
11,8);
INSERT INTO product VALUES(NULL,"Stapler", "Swingline", "Standard", 100,1,
5, 11,6);
INSERT INTO product VALUES(NULL,"Scissors", "Fiskars", "8-inch", 80, 1, 4,
11,1);
INSERT INTO product VALUES(NULL,"Marker", "Sharpie", "Fine tip", 70, 1, 0,
18,2);
INSERT INTO product VALUES(NULL,"Glue", "Elmer's", "Clear", 50, 1, 2,
12,3);
INSERT INTO product VALUES(NULL,"Calculator", "Texas Instruments",
"Scientific", 300, 1, 10,14,3);
INSERT INTO product VALUES(NULL,"Tape", "Scotch", "Transparent", 25, 1, 1,
11,5);
INSERT INTO product VALUES(NULL,"Folder", "Smead", "Letter size", 60, 1,3,
15,10);
INSERT INTO product VALUES(NULL,"Desk Organizer", "Rolodex", "Mesh", 120,
1, 6, 15,7);
INSERT INTO product VALUES(NULL,"Hole Punch", "Bostitch", "3-hole", 90, 1,
4, 15,6);
INSERT INTO product VALUES(NULL,"Whiteboard", "Quartet", "Magnetic", 250,
1, 15, 15,4);
INSERT INTO product VALUES(NULL,"Paper Clips", "Acco", "Jumbo", 35, 1, 1,
15,3);

SELECT * FROM product;

CREATE TABLE reviews(
      review_id INTEGER PRIMARY KEY auto_increment,
    product_id INTEGER REFERENCES product(product_id),
    user_id INTEGER REFERENCES user(userid),
    product_review VARCHAR(50) ,
    product_ratings VARCHAR(50),
    review_date DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP
    );



INSERT INTO reviews (product_id,  user_id, product_review,product_ratings)
VALUES (1, 2,  'OK', '5 star'),
      (2,  2,  'BAD', '1 star'),
    (3 ,  2,  'OK', '2 star'),
    (3,  3,  'low quality', '1 star'),
    (4, 4,  'nooo', '1 star'),
    (5,  5,  'bad', '2 star'),
    (6 , 6,  'poor quality', '3 star'),
    (7, 7,  'damaged product', '1 star'),
    (1, 8, 'dissatisfied', '1 star'),
    (1,  2, 'bad experience', '1 star')
    ;




SELECT * FROM reviews;

CREATE TABLE vendor(
      vendor_id INTEGER PRIMARY KEY auto_increment,
    vendor_username VARCHAR(50) NOT NULL,
    vendor_phone VARCHAR(50),
    vendor_email VARCHAR(50) NOT NULL UNIQUE,
      vendor_password VARCHAR(50) NOT NULL UNIQUE,
    vendor_address VARCHAR(50) NOT NULL
    );

INSERT INTO vendor VALUES
(NULL, 'ram', '123-456-7890', 'ram@gmail.com', 'password123', 'kolkata'),
(NULL, 'jaya', NULL, 'jaya@gmail.com', 'securepass', 'shimla'),
(NULL, 'shyam', '555-123-4567', 'shyam@gmail.com', 'strongpassword',
'connaught place'),
(NULL, 'lisa_wong', '321-654-0987', 'lisa@gmail.com', 'letmein123', '101
Pine St'),
(NULL, 'mike_jones', NULL, 'mike@gmail.com', 'password456', '202 MapleSt'),
(NULL, 'sara', '999-888-7777', 'sara@gmail.com', 'safepassword', 'goa'),
(NULL, 'adani', '777-666-5555', 'adani@gmail.com', '12345678', 'kerala'),
(NULL, 'ambani', '444-333-2222', 'ambani@gmail.com', 'password789',
'bihar'),
(NULL, 'kartik', '111-222-3333', 'kartik@gmail.com', 'passwordabc',
'chennai'),
(NULL, 'jaman', '666-777-8888', 'jaman@gmail.com', 'securepassword123',
'mumbai');

SELECT * FROM vendor;

CREATE TABLE payment(
      payment_id INTEGER PRIMARY KEY auto_increment,
    user_id INTEGER NOT NULL,
      payment_type_id INTEGER NOT NULL REFERENCES
PaymentType(payment_type_id),
      account_number INTEGER NOT NULL,
    CONSTRAINT payment_fk FOREIGN KEY(user_id) REFERENCES user(user_id)

    );

INSERT INTO payment VALUES
      (NULL, 1, 101, 23221),
    (NULL, 1, 103, 23222),
    (NULL, 3, 104, 23223),
    (NULL, 4, 101, 23224),
    (NULL, 2, 104, 23225),
    (NULL, 6, 102, 23226),
    (NULL, 7, 105, 23227),
    (NULL, 4, 104, 23228),
    (NULL, 9, 101, 23229),
    (NULL, 10, 105, 23220);


SELECT * FROM payment;

CREATE TABLE shop_order(
      order_id INTEGER PRIMARY KEY auto_increment,
    user_id INTEGER NOT NULL,
    order_date DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    payment_id INTEGER NOT NULL REFERENCES payment(payment_id),
    shipping_addresss VARCHAR(50) NOT NULL,
    order_total INTEGER NOT NULL,
    order_status VARCHAR(50) NOT NULL,
    delivered_by INTEGER REFERENCES delivery_employee(employee_id),

    CONSTRAINT shop_order_fk FOREIGN KEY(user_id) REFERENCES user(user_id)
    );
```

```
INSERT INTO shop_order (user_id, payment_id, shipping_addresss,
order_total, order_status, delivered_by)
VALUES
(1, 1, 4, 50, 'Pending', NULL),
(2, 5, 3, 75, 'Processing' , NULL),
(3, 3, 5, 100, 'Shipped', NULL),
(1, 2 , 2, 120, 'Delivered', 6),
(6, 6, 5, 80, 'Cancelled',NULL),
(4, 8, 3, 90, 'Pending', NULL),
(10 , 10, 13, 110, 'Processing', NULL),
(7, 7, 11, 70, 'Delivered', 2),
(4, 4, 3, 95, 'Delivered', 4),
(9, 9, 2, 85, 'Cancelled', NULL);

SELECT * FROM shop_order;



CREATE TABLE shipping_details(
      shipping_id INTEGER PRIMARY KEY auto_increment,
    unit_number INTEGER NOT NULL,
    street_number VARCHAR(50) NOT NULL,
    region VARCHAR(100) NOT NULL,
    city VARCHAR(50) NOT NULL,
    postal_code INT NOT NULL,
    country VARCHAR(100) NOT NULL

    );

INSERT INTO shipping_details VALUES
(NULL,1,"68","Delhi","New Delhi",110001,"India"),
(NULL,2,"70","Tamil Nadu","Chennai",600001,"India"),
(NULL,3,"53","Andhra Pradesh","Visakhapatnam",530001,"India"),
(NULL,4,"33","Jharkhand","Ranchi",834001,"India"),
(NULL,5,"26","Gujarat","Ahmedabad",380001,"India"),
(NULL,6,"12","Rajasthan","Jaipur",302001,"India"),
(NULL,7,"27","Maharashtra","Mumbai",400001,"India"),
(NULL,8,"54","Karnataka","Bengaluru",560001,"India"),
(NULL,9,"35","West Bengal","Kolkata",700001,"India"),
(NULL,10,"17","Telangana","Hyderabad",500001,"India"),
(NULL,11,"28","Uttar Pradesh","Lucknow",226001,"India"),
(NULL,12,"7","Madhya Pradesh","Bhopal",462001,"India");

SELECT * FROM shipping_details;
```

```
CREATE TABLE PaymentType(
      payment_type_id INTEGER PRIMARY KEY ,
      Type VARCHAR(50) NOT NULL
    );

INSERT INTO PaymentType (payment_type_id, Type)
VALUES
(101, 'Credit Card'),
(102, 'Debit Card'),
(103, 'PayPal'),
(104, 'Cash'),
(105, 'Check');

SELECT * FROM paymenttype;

CREATE TABLE category(
      category_id INTEGER PRIMARY KEY ,
    category_name VARCHAR(50) NOT NULL UNIQUE

    );

INSERT INTO category VALUES
(11,"Stationary"),
(12,"Craft Supplies"),
(13,"Dairy Products"),
(14,"Electronics"),
(15,"Office supplies"),
(16,"Clothing"),
(17,"Furniture"),
(18,"Art Supplies"),
(19,"Books"),
(20,"Medical Supplies");

SELECT * FROM category;



CREATE TABLE shopping_cart(
      id INTEGER PRIMARY KEY auto_increment,
      user_id INTEGER NOT NULL REFERENCES user(user_id)
    );



INSERT INTO shopping_cart VALUES(NULL,
      10),
    (NULL, 2),
    (NULL, 3),
    (NULL, 5),
    (NULL, 4),
    (NULL, 7),
    (NULL, 6),
    (NULL, 1),
    (NULL, 8),
    (NULL, 9);

SELECT * FROM shopping_cart;



CREATE TABLE wishlist(
      id INTEGER PRIMARY KEY auto_increment,
      product_id INT NOT NULL REFERENCES product(product_id) ,
    user_id INT NOT NULL REFERENCES user(user_id),
    add_to_cart BOOLEAN NOT NULL

    );

INSERT INTO wishlist VALUES(NULL,5,1,1),
(NULL,6,9,0),
(NULL,7,1,1),
(NULL, 8, 4, 0),
(NULL, 9, 5, 1),
(NULL, 10, 2, 0),
(NULL, 11,3, 1),
(NULL, 12, 2, 0),
(NULL, 13, 9, 1),
(NULL, 14, 7, 0);

SELECT * FROM wishlist;

CREATE TABLE delivery_employee(
      employee_id INTEGER PRIMARY KEY auto_increment,
    employee_username VARCHAR(50) NOT NULL,
    employee_phone VARCHAR(50),
    employee_email VARCHAR(50) NOT NULL UNIQUE,
      employee_password VARCHAR(50) NOT NULL UNIQUE

    );
```

```
INSERT INTO delivery_employee VALUES
      (NULL, "Ram",9888888841, "ram@gmail.com", "rrrr"),
    (NULL, "Shyam",9888888851, "shyam@gmail.com", "rr888r"),
    (NULL, "Mohan",9888888871, "mohan@gmail.com", "rrrr9999"),
    (NULL, "Mahesh",9888888891, "mahesh@gmail.com", "rrr1111r"),
    (NULL, "Shankar",9888888821, "shankar@gmail.com", "rrr23456r"),
    (NULL, "Prakash",9888888831, "pr@gmail.com", "rr55678rr"),
    (NULL, "Renuka",9888888862, "rm@gmail.com", "rrr665432r"),
    (NULL, "Rama",9888818841, "rama@gmail.com", "r6r"),
    (NULL, "Ramu",9828888841, "ramu@gmail.com", "rsdfr"),
    (NULL, "Govind",9888388841, "govind@gmail.com", "rrrg356ur");

SELECT * FROM delivery_employee;

CREATE TABLE shopping_cart_item(
      id INTEGER PRIMARY KEY auto_increment,
      cart_id INT NOT NULL,
      CONSTRAINT ShoppingCartItem FOREIGN KEY(cart_id) REFERENCES
shopping_cart(id),
      product_id INTEGER REFERENCES product(product_id),
      product_quantity INTEGER,
      cart_total_cost INTEGER,
      cart_total_items INTEGER
    );

INSERT INTO shopping_cart_item VALUES
(NULL,1,80,2,2000,3),
(NULL,2,92,4,1000,4),
(NULL,3,62,3,500,2),
(NULL,4,75,1,800,2),
(NULL,1,58,3,1900,4),
(NULL,2,95,2,1200,1),
(NULL,3,80,4,750,3),
(NULL,4,65,1,600,2),
(NULL,1,98,3,1500,1),
(NULL,2,82,2,1100,3);

SELECT * FROM shopping_cart_item;

CREATE TABLE user_address(
      user_id INTEGER REFERENCES user(userid),
    shipping_id INTEGER REFERENCES shipping_details(shipping_id),
    is_default BOOLEAN NOT NULL

 );
```

```
INSERT INTO user_address VALUES
(1,4,0),
(2,3,1),
(3,5,0),
(1, 2, 0),
(2, 1, 1),
(3, 5, 1),
(4, 3, 0),
(8, 4, 1),
(9, 2, 0),
(1, 5, 1);

SELECT * FROM user_address;

CREATE TABLE order_line(
    id INTEGER PRIMARY KEY auto_increment,
    product_item_id INTEGER NOT NULL REFERENCES product(product_id),
    order_id INTEGER NOT NULL,
    qty INTEGER NOT NULL,
    price INTEGER NOT NULL,
    CONSTRAINT shop_order_fk1 FOREIGN KEY(order_id) REFERENCES
shop_order(order_id)
);

INSERT INTO order_line (product_item_id, order_id, qty, price)
VALUES
(2, 1, 1, 65),
(2, 3, 3, 65),
(5, 3, 5, 100),
(1, 2 , 2, 120),
(2, 6, 5, 80),
(1, 1, 3, 90),
(10 , 10, 3, 45),
(5, 3, 1, 70),
(4, 4, 2, 25),
(2, 9, 2, 85);

SELECT * FROM order_line;
```

Queries:

- Query 1: Retrieve the top 3 users who have purchased the most products from the "Stationery" category, along with their total spending

```
SELECT u.user_id, u.user_name, COUNT(ol.product_item_id) AS
total_products_purchased,SUM(p.product_price) AS total_spending
FROM user u
JOIN shop_order o ON u.user_id = o.user_id
JOIN order_line ol ON o.order_id = ol.order_id
JOIN product p ON ol.product_item_id = p.product_id
JOIN category c ON p.category_id = c.category_id
WHERE c.category_name = "Stationary"
GROUP BY u.user_id
ORDER BY total_products_purchased DESC
LIMIT 3;
```

```
π(user_id, user_name, total_products_purchased, total_spending)
    (γ(user_id, user_name, COUNT(product_item_id) AS
total_products_purchased, SUM(product_price) AS total_spending) (
        ρ(u, user) ⋈
        ρ(o, shop_order) ⋈
        ρ(ol, order_line) ⋈
        ρ(p, product) ⋈ρ(c,
        category) (
            user ⋈ (shop_order ⋈ (order_line ⋈ product ⋈ category))
        )
        σ(category_name="Stationery") (
            category
        )
        group by user_id, user_name
    )
)
```

Query 2:- Find users who have not purchased any products from a electronics category:

```
SELECT u.user_id, u.user_name
FROM user u
WHERE NOT EXISTS
    (SELECT *
    FROM shop_order o
    JOIN order_line ol ON o.order_id = ol.order_id
    JOIN product p ON ol.product_item_id = p.product_id
    JOIN category c ON p.category_id = c.category_id
    WHERE c.category_name = 'Electronics'
    AND o.user_id = u.user_id
);
```

```
π(user_id, user_name)
    (ρ(u, user) ÷ (
        ρ(u_electronics, user) ⋈
            (ρ(o, shop_order) ⋈
            ρ(ol, order_line) ⋈ρ(p,
            product) ⋈
            ρ(c, category) (
                shop_order ⋈ order_line ⋈ product ⋈ category
            )
        )
        σ(category_name='Electronics') (
            category
        )
    )
)
```

Query 3:- Retrieve the top 2 most profitable products along with their total revenue and vendor information:

```
SELECT p.product_id, p.product_name, v.vendor_id, v.vendor_username,
      SUM(ol.qty * ol.price) AS total_revenue
FROM product p
JOIN order_line ol ON p.product_id = ol.product_item_id
JOIN vendor v ON p.vendor_id = v.vendor_id
GROUP BY p.product_id, p.product_name, v.vendor_id, v.vendor_username
ORDER BY total_revenue DESC
LIMIT 2;
```

```
π(product_id, product_name, vendor_id, vendor_username, total_revenue) (
    ρ(p,  product)  ⋈
        ( ρ(ol, order_line)
        ⋈ρ(v, vendor) (
            product ⋈ (order_line ⋈ vendor)
        )
    )
    γ(product_id, product_name, vendor_id, vendor_username; total_revenue)
(
        p - p{product_id, product_name, vendor_id, vendor_username}
    )
)
```

Query 4:- Retrieve users who have purchased all products in theirwishlist:

```
SELECT u.user_id, u.user_name
FROM user u
WHERE NOT EXISTS (
    SELECT w.product_id
    FROM wishlist w
    WHERE w.user_id = u.user_id
    AND w.product_id NOT IN (
        SELECT ol.product_item_id
        FROM order_line ol
        JOIN shop_order so ON ol.order_id = so.order_id
        WHERE so.user_id = u.user_id
    )
);

π(user_id, user_name)(
    user
    -
    π(product_id)(
        ρ(w, wishlist) ⋈ user
        -
        ρ(ol, order_line) ⋈ π(order_id)(σ(user_id = user_id)(shop_order))
    )
)
```

Query 5:- Retrieve users who have purchased products from all categories:

```
SELECT u.user_id, u.user_name
FROM user u
JOIN (
    SELECT u.user_id
    FROM user u
    JOIN shop_order so ON u.user_id = so.user_id
    JOIN order_line ol ON so.order_id = ol.order_id
    JOIN product p ON ol.product_item_id = p.product_id
    GROUP BY u.user_id
    HAVING COUNT(DISTINCT p.category_id) = (SELECT COUNT(*) FROM category)
) AS users_all_categories ON u.user_id = users_all_categories.user_id;
```

```
π(user_id, user_name)(
    user ⋈ (
        π(user_id)(
            ρ(u, user) ⋈ shop_order ⋈ order_line ⋈ product
            ÷
            category
        )
    )
)
```

Query 6:- Decrease the price of all products in the "Electronics" category by 20% if they have not been purchased in the last 6 months

```
UPDATE product
SET product_price = product_price * 0.8
WHERE category_id = (SELECT category_id FROM category WHERE category_name
= 'Electronics')
AND product_id NOT IN
    ( SELECT
    ol.product_item_idFROM
    order_line ol
    JOIN shop_order o ON ol.order_id = o.order_id
    WHERE DATEDIFF(CURRENT_DATE, o.order_date) <= 180
);
```

```
product := product * (1 - 0.2) ⋈ π(product_id)(
                ρ(p, product) ⋈ category
                ÷
                ρ(purchased_product_id, π(product_id)(
                    ρ(s, shop_order) ⋈ order_line ⋈ product
                    ⋈ ρ(c, category)
                    ⋈ σ(category_name = 'Electronics' AND order_date >=
CURRENT_DATE - INTERVAL '6' MONTH)(shop_order)
                ))
            )
```

Query 7:- Increase the product quantity by 10 for all products with a total revenue greater than $1000

```
UPDATE product
SET product_qty = product_qty + 10
WHERE product_id IN (
    SELECT p.product_id
    FROM product p
    JOIN order_line ol ON p.product_id = ol.product_item_id
    GROUP BY p.product_id
    HAVING SUM(ol.qty * ol.price) > 1000
);
```

```
product_quantity := product_quantity + 10 (σ(total_revenue > 1000)(product
⋈ (ρ(total_revenue, SUM(qty * price))(order_line) ÷ product)))
```

Query 8:- Remove all users who have not made any purchases and have not reviewed any products:

```
DELETE FROM user
WHERE user_id NOT IN (
    SELECT DISTINCT u.user_id
    FROM user u
    LEFT JOIN shop_order o ON u.user_id = o.user_id
    LEFT JOIN reviews r ON u.user_id = r.user_id
    WHERE o.order_id IS NOT NULL OR r.review_id IS NOT NULL
);
```

```
user := user - π(user_id)(σ(order_id IS NOT NULL OR review_id IS NOT
NULL)((user ⋈ shop_order) U (user ⋈ reviews)))
```

Query 9:- Mark all orders with a total value greater than $500 as expedited orders:

```
UPDATE shop_order
SET order_type = 'Expedited'
WHERE order_id IN (
    SELECT order_id
    FROM (
        SELECT so.order_id, SUM(ol.qty * ol.price) AS total_value
        FROM shop_order so
        JOIN order_line ol ON so.order_id = ol.order_id
        GROUP BY so.order_id
    ) AS order_totals
    WHERE total_value > 500
);
```

```
expedited_orders := σ(total_value > 500)(shop_order ⋈ (ρ(total_value,
SUM(qty * price))(order_line) ÷ shop_order))
```

Query 10:- Retrieve the average amount spent per transaction for each payment type

```
SELECT pt.Type AS payment_type, AVG(s.product_price * ol.qty) AS
avg_amount_per_transaction
FROM payment p
JOIN PaymentType pt ON p.payment_type_id = pt.payment_type_id
JOIN shop_order so ON p.payment_id = so.payment_id
JOIN order_line ol ON so.order_id = ol.order_id
JOIN product s ON ol.product_item_id = s.product_id
GROUP BY pt.Type;
```

```
π(payment_type, avg_amount_per_transaction)
(ρ(pt, PaymentType) ⋈ (
ρ(p, payment) ⋈ (
ρ(so, shop_order) ⋈ (
ρ(ol, order_line) ⋈ (
ρ(s, product) ⋈ pt.payment_type_id = p.payment_type_id ⋈ so.payment_id =
p.payment_id ⋈ ol.order_id = so.order_id ⋈ s.product_id =
ol.product_item_id
))))
γ(pt.Type; AVG(s.product_price * ol.qty) AS avg_amount_per_transaction) ÷
pt.Type
)
```

By:
Gurupriya Vaikundam(2022191)
Jas keerat Singh(2022226)
Mananya Kohli(2022275)
Rituj Upadhyay(2020570)