SVKM's NMIMS
School of Technology Management & Engineering
A.Y. 2023 - 24
Course: Database Management Systems

Project Report

Program	BTech CE	
Semester	IV	
Name of the Project:	E Commerce Shopping Cart System	
Details of Project Members		
Batch	Roll No.	Name
1	T009	Arohi Agrawal
1	T013	Arushi Dubey
1	T022	Divyendra Singh
Date of Submission: 27/03/24		

Contribution of each project Members:

Roll No.	Name:	Contribution
T009	Arohi Agrawal	ER Diagram
T013	Arushi Dubey	Relational Model
T022	Divyendra Singh	Normalization, SQL Queries

Github link of your project:

Note:

https://github.com/AviSingh1112/DBMS-Assignment-on-Ecommerce-Cart-System

Project Report

Selected Topic

E Commerce Shopping Cart System

by

Arohi Agrawal, T009

Arushi Dubey, T013

Divyendra Singh, T022

Course: DBMS

AY: 2023-24

Table of Contents

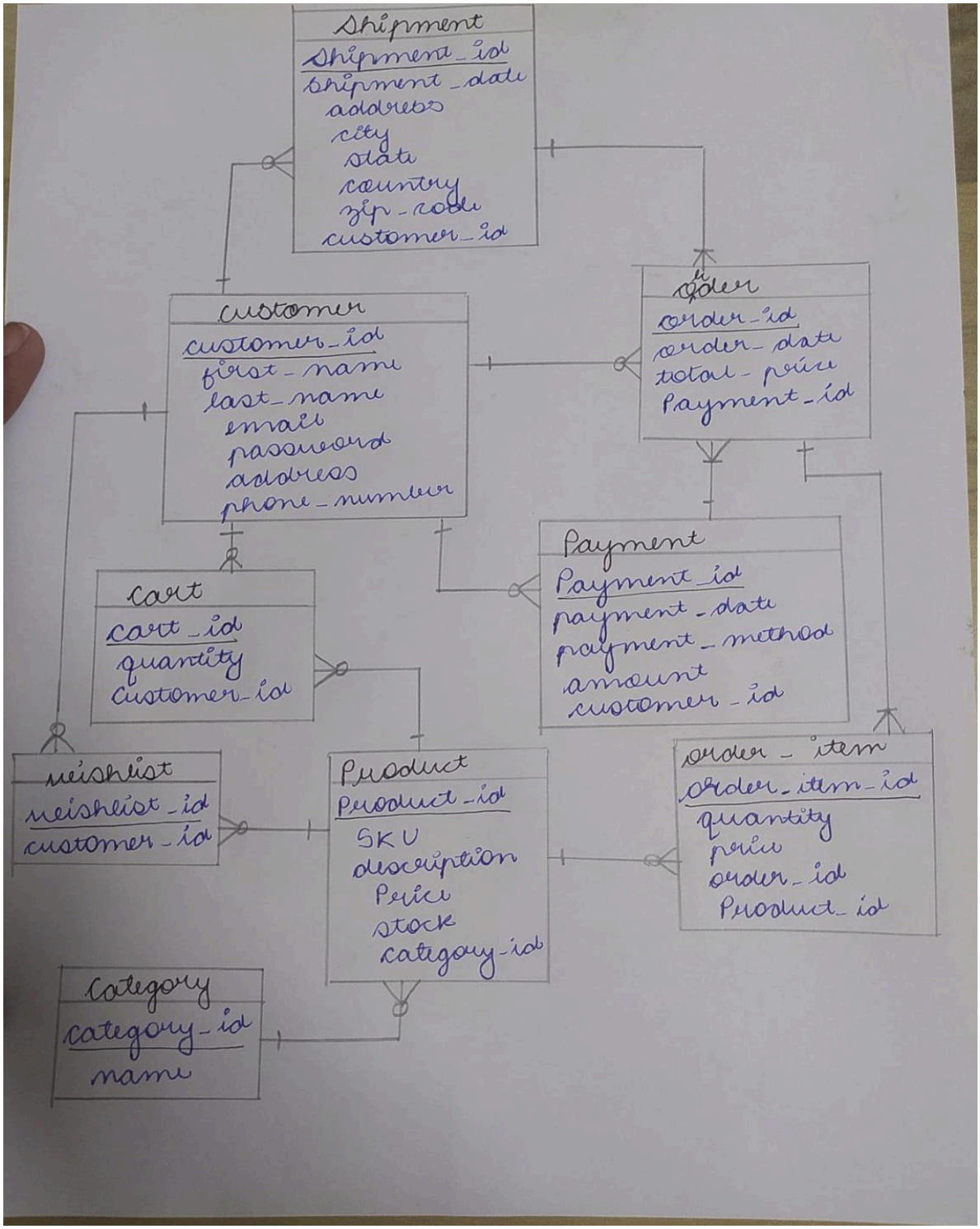
I. Storyline

Maya, a skilled potter known for her whimsical creations, pours her heart into each piece but struggles to sell beyond local craft fairs. Frustrated, she stumbles upon our e-commerce system. Learning its ropes, she hesitantly uploads her work. Soon, stunning photos and detailed descriptions captivate online customers. Orders trickle in, then flow, transforming Maya's tiny studio into a bustling hub. Packaging delicate mugs and shipping worldwide, she marvels at the reach she never dreamed of. Maya's success story becomes a beacon, inspiring fellow artisans to embrace e-commerce, proving that even the most unique creations can find their audience in the vast digital marketplace.

II. Components of Database Design

Entity	Attributes	Primary key	Foreign Key
Shipment	Shipment_id,Shipment_data, address, city, state, country, zip_code, Customer_id	Shipment_id	Customer_id
Customer	Customer_id, first_name, last_name, email, password, address, phone_number	Customer_id	NA
Order	Order_id, Order_date, total_price, Payment_id	Order_id	Payment_id
Cart	Cart_id, quantity,Customer_id	Cart_id	Customer_id
Payment	Payment_id, Payment_date, Payment_method, amount, Customer_id	Payment_id	Customer_id
Order_item	Order_item_id, quantity, price, Order_id, Product_id	Order_item_id	Order_id Product_id
Wishlist	Wishlist_id, Customer_id	Wishlist_id	Customer_id
Product	Product_id, SKU, description, price, stock, Category_id	Product_id	Category_id
Category	Category_id, name	Category_id	NA

III. Entity Relationship Diagram



IV. Relational Model

Convert the ER diagram to the relational model using the concepts learned in the class. List the various tables obtained.

- **Customer**
  - customer_id (primary key)
  - first_name
  - last_name
  - address
  - phone_number
- **Order**
  - order_id (primary key)
  - customer_id (foreign key references Customer.customer_id)
  - order_date
  - payment_id (foreign key references Payment.payment_id)
- **Payment**
  - payment_id (primary key)
  - payment_date
  - payment_method
  - amount
  - customer_id (foreign key references Customer.customer_id)
- **Product**
  - product_id (primary key)
  - SKU
  - description
  - price
  - stock
  - category_id (foreign key references Category.category_id)
- **Category**
  - category_id (primary key)
  - name
- **Cart**
  - cart_id (primary key)
  - customer_id (foreign key references Customer.customer_id)
- **Wishlist**
  - wishlist_id (primary key)
  - customer_id (foreign key references Customer.customer_id)
- **Order_Item**
  - order_item_id (primary key)
  - order_id (foreign key references Order.order_id)
  - product_id (foreign key references Product.product_id)
  - quantity
  - price

V. Normalization

Normalization is a fundamental database design principle aimed at reducing redundancy and ensuring data integrity. The following is an analysis of the provided entities in a relational database schema, evaluated for adherence to the principles of normalization through the Boyce-Codd Normal Form (BCNF).

Shipment Entity
- First Normal Form (1NF): Confirmed. The Shipment entity contains atomic values, and each row is uniquely identifiable by `Shipment_id`.
- Second Normal Form (2NF): Confirmed. All attributes are fully dependent on the primary key.
- Third Normal Form (3NF): Confirmed. No attributes are transitively dependent on the primary key.
- Boyce-Codd Normal Form (BCNF): Confirmed. Since the primary key is the sole determinant, the Shipment entity satisfies BCNF.

Customer Entity
-1NF: Confirmed. The Customer entity exhibits atomicity and has unique `Customer_id` identifiers for each row.
- 2NF: Confirmed. It meets all criteria for 2NF as there are no composite primary keys to induce partial dependencies.
- 3NF: Confirmed. There are no transitive dependencies within the entity.
- BCNF: Confirmed. The Customer entity adheres to BCNF with all determinants being candidate keys.

Order Entity
- 1NF: Confirmed. Each attribute in the Order entity is stored as atomic values, with `Order_id` serving as a unique identifier.
- 2NF: Confirmed. It satisfies the criteria of 2NF.
- 3NF: Confirmed. There are no transitive dependencies and the table is already in 2NF.
- BCNF: Confirmed. The Order entity adheres to BCNF with all determinants being candidate keys.

Cart Entity
- 1NF: Confirmed. The Cart entity contains atomic, indivisible data with `Cart_id` as the primary key.
- 2NF: Confirmed. Attributes are fully dependent on the primary key with no partial dependencies.
- 3NF: Confirmed. Non-key attributes do not have transitive dependencies.
- BCNF: Confirmed. `Cart_id` is a superkey, and there are no other determinants, thus fulfilling the conditions for BCNF.

Payment Entity
- 1NF: Confirmed. All attributes in the Payment entity have atomic data entries.
- 2NF: Confirmed. The Payment entity's attributes are all fully dependent on the primary key `Payment_id`.
- 3NF: Confirmed. There are no transitive dependencies present.
- BCNF: Confirmed. The primary key is the only determinant, ensuring compliance with BCNF.

Order_item Entity
- 1NF: Confirmed. The entity displays atomicity in its attributes and has `Order_item_id` as a primary key.

- 2NF: Confirmed. Full functional dependency on the primary key is observed.
- 3NF: Confirmed. There are no transitive dependencies present.
- BCNF: Confirmed. The primary key is the only determinant, ensuring compliance with BCNF.

Wishlist Entity
- 1NF: Confirmed. The Wishlist entity maintains atomicity and uniqueness through `Wishlist_id`.
- 2NF: Confirmed. The entity adheres to 2NF as all attributes are dependent on the primary key.
- 3NF: Confirmed. No transitive dependencies are found within the Wishlist entity.
- BCNF: Confirmed. All functional dependencies are on the primary key, satisfying BCNF.

Product Entity
- 1NF: Confirmed. The Product entity possesses atomic values and unique product identifiers.
- 2NF: Confirmed. There are no partial dependencies; all attributes are fully dependent on `Product_id`.
- 3NF: Confirmed. Each non-key attribute is only dependent on the primary key.
- BCNF: Confirmed. `Product_id` serves as the only determinant, which aligns with the BCNF criteria.

Category Entity
- 1NF: Confirmed. The Category entity contains atomic values with a unique identifier for each row.
- 2NF: Confirmed. Since there is only a single primary key, there are no issues with partial dependency.
- 3NF: Confirmed. There are no indications of transitive dependencies within the entity.
- BCNF: Confirmed. There is no evidence of non-primary key determinants, thus the entity complies with BCNF.

Conclusion
The evaluation of the entities against normalization standards indicates that most entities are well-structured and adhere to the principles of normalization up to BCNF, with a few concerns noted for the Order and Payment entities. For the Order entity, the presence of `Payment_id` may suggest a need for an associative entity to handle the many-to-many relationship between orders and payments. The Payment entity's inclusion of `Customer_id` should be reviewed to ensure no transitive dependencies exist that would violate BCNF. Addressing these concerns will result in a robust database schema that maintains data integrity, reduces redundancy, and supports efficient data management practices.

VI. SQL Queries

1. **Sort by price(lowest to highest)**

```
250 •   SELECT * FROM Product ORDER BY price ASC;
251
252
```

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

Product_id	SKU	description	price	stock	Category_id
10	SKU107	Bluetooth Speaker	1499.00	80	1
6	SKU103	Wireless Charger	1999.00	100	1
15	SKU112	Router	2499.00	130	1
3	SKU789	Bluetooth Headphones	2999.00	200	1
12	SKU109	Video Game	3499.00	200	2
13	SKU110	Fitness Band	3499.00	110	1
14	SKU111	External Hard Drive	5999.00	120	1
9	SKU106	E-reader	7999.00	150	2
8	SKU105	Printer	8999.00	60	1
4	SKU101	Smartwatch	9999.00	75	1
1	SKU123	Smartphone	19999.00	100	1
5	SKU102	Tablet	24999.00	50	1
7	SKU104	DSLR Camera	29999.00	40	1
11	SKU108	Gaming Console	29999.00	90	1
2	SKU456	Laptop	49999.00	150	1

2. Find products within a specific price range (e.g., 15000 to 20000):

```
252 •   SELECT * FROM Product WHERE price BETWEEN 15000 AND 20000;
```

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

Product_id	SKU	description	price	stock	Category_id
1	SKU123	Smartphone	19999.00	100	1

3.Get products by category name (e.g., "Electronics"):

```
254 •   SELECT p.*
255     FROM Product p
256     JOIN Category c ON p.Category_id = c.Category_id
257     WHERE c.name = 'Electronics';
```

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

Product_id	SKU	description	price	stock	Category_id
1	SKU123	Smartphone	19999.00	100	1
2	SKU456	Laptop	49999.00	150	1
3	SKU789	Bluetooth Headphones	2999.00	200	1
4	SKU101	Smartwatch	9999.00	75	1
5	SKU102	Tablet	24999.00	50	1
6	SKU103	Wireless Charger	1999.00	100	1
7	SKU104	DSLR Camera	29999.00	40	1
8	SKU105	Printer	8999.00	60	1
10	SKU107	Bluetooth Speaker	1499.00	80	1
11	SKU108	Gaming Console	29999.00	90	1
13	SKU110	Fitness Band	3499.00	110	1
14	SKU111	External Hard Drive	5999.00	120	1
15	SKU112	Router	2499.00	130	1

4. Find products with low stocks:

```
260 •   SELECT * FROM Product WHERE stock < 10;
```

Result Grid | Filter Rows: | Edit: | Export/Im

Product_id	SKU	description	price	stock	Category_id
1	SKU123	Smartphone	19999.00	9	1
3	SKU789	Bluetooth Headphones	2999.00	8	1
NULL	NULL	NULL	NULL	NULL	NULL

5. Search for products with the keyword “Smartphone” in their description:

```
262 •   SELECT * FROM Product WHERE description LIKE '%smartphone%';
```

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

Product_id	SKU	description	price	stock	Category_id
1	SKU123	Smartphone	19999.00	9	1

6.Find products that have never been purchased:

```
264 •   SELECT p.*
265     FROM Product p
266     LEFT JOIN Order_item oi ON p.Product_id = oi.Product_id
267     WHERE oi.Product_id IS NULL;
```

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

Product_id	SKU	description	price	stock	Category_id
101	SKU101	New Smartphone X	20000.00	50	1
102	SKU102	New Headphones Y	3000.00	50	1

7.Sort products within a specific category by popularity:

```
269 •   SELECT p.*, SUM(oi.quantity) as TotalSold
270     FROM Product p
271     JOIN Order_item oi ON p.Product_id = oi.Product_id
272     JOIN Category c ON p.Category_id = c.Category_id
273     WHERE c.name = 'Electronics'
274     GROUP BY p.Product_id
275     ORDER BY TotalSold DESC;
276
```

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

Product_id	SKU	description	price	stock	Category_id	TotalSold
3	SKU789	Bluetooth Headphones	2999.00	8	1	2
10	SKU107	Bluetooth Speaker	1499.00	80	1	2
1	SKU123	Smartphone	19999.00	9	1	1
2	SKU456	Laptop	49999.00	150	1	1
4	SKU101	Smartwatch	9999.00	75	1	1
5	SKU102	Tablet	24999.00	50	1	1
6	SKU103	Wireless Charger	1999.00	100	1	1
7	SKU104	DSLR Camera	29999.00	40	1	1
8	SKU105	Printer	8999.00	60	1	1
11	SKU108	Gaming Console	29999.00	90	1	1
13	SKU110	Fitness Band	3499.00	110	1	1
14	SKU111	External Hard Drive	5999.00	120	1	1
15	SKU112	Router	2499.00	130	1	1

8. Get details of customers along with the count of items in their cart:

```
SELECT c.Customer_id, c.first_name, c.last_name, COUNT(cart.Cart_id) as CartItemCount
FROM Customer c
LEFT JOIN Cart cart ON c.Customer_id = cart.Customer_id
GROUP BY c.Customer_id;
```

Customer_id	first_name	last_name	CartItemCount
1	Arnav	Chaturvedi	1
2	Riya	Shagun	1
3	Ishika	Rathore	1
4	Anika	Sethia	1
5	Yashwardhan	Sant	1
6	Nakul	Sethi	1
7	Arjav	Kasliwal	1
8	Pranav	Joshi	1
9	Oshin	Ahuja	1
10	Lakshya	Dubey	1
11	Shanali	Singh	1
12	Suhani	Khandelwal	1
13	Kashish	Wadhwani	1
14	Prakhar	Agrawal	1
15	Ananya	Pandey	1
16	Kabir	Singh	0
17	Saanvi	Kumar	0
18	Harshit	Chawla	0
19	Khushi	Upadhyay	0
20	Chhavi	Ambor	0

9. List all shipments that are either 'In Transit' or 'Pending':

```
282 •   SELECT * FROM Shipment WHERE Shipment_data IN ('In Transit', 'Pending');
```

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

Shipment_id	Shipment_data	address	city	state	country	zip_code	Customer_id
2	In Transit	Kandivali, Mumbai	Mumbai	Maharashtra	India	400067	2
4	Pending	Gomasthanagar, Indore	Indore	Madhya Pradesh	India	452009	4
6	In Transit	bhojpuria, Patna	Patna	Bihar	India	800026	6
8	Pending	1514 Malleshwaram, Bangalore	Bangalore	Karnataka	India	560003	8
10	In Transit	Khajrana, Indore	Indore	Madhya Pradesh	India	452016	10
12	Pending	Nala Supara, Mumbai	Mumbai	Maharashtra	India	401209	12
14	In Transit	Dhawari, Satna	Satna	Madhya Pradesh	India	485005	14
16	Pending	2323 Gandhinagar, Delhi	New Delhi	Delhi	India	110031	16
18	In Transit	2525 C Scheme, Jaipur	Jaipur	Rajasthan	India	302001	18
20	Pending	2727 Hazratganj, Lucknow	Lucknow	Uttar Pradesh	India	226001	20

10. Find the top 3 most ordered products in each category:

```
284 •  SELECT p.Category_id, p.Product_id, p.description, COUNT(oi.Order_item_id) AS order_count
285    FROM Product p
286    JOIN Order_item oi ON p.Product_id = oi.Product_id
287    GROUP BY p.Category_id, p.Product_id
288    ORDER BY p.Category_id, order_count DESC
289    LIMIT 3;
```

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

Category_id	Product_id	description	order_count
1	1	Smartphone	1
1	2	Laptop	1
1	3	Bluetooth Headphones	1

11. Apply a 10% discount on items priced above 200:

```
UPDATE Orders
SET total_price = total_price * 0.9
WHERE total_price > 200;
```

Order_id	Order_date	total_price	Payment_id	Customer_id
3	2023-01-12	89.99	NULL	3
4	2023-01-13	59.99	NULL	4
5	2023-01-14	99.99	NULL	5
6	2023-01-15	29.99	NULL	6
7	2023-01-16	199.99	NULL	7
8	2023-01-17	269.99	NULL	8
9	2023-01-18	359.99	NULL	9
10	2023-01-19	129.99	NULL	10
11	2023-01-20	159.99	NULL	11
12	2023-01-21	224.99	NULL	12
13	2023-01-22	251.99	NULL	13
14	2023-01-23	179.99	NULL	14
15	2023-01-24	199.99	NULL	15

12.Find the latest order for each customer

```
SELECT
    o.Customer_id,
    c.first_name,
    c.last_name,
    MAX(o.Order_date) AS LatestOrderDate
FROM
    Orders o
INNER JOIN
    Customer c ON o.Customer_id = c.Customer_id
GROUP BY
    o.Customer_id;
```

Customer_id	first_name	last_name	LatestOrderDate
1	Arnav	Chaturvedi	2023-01-10
2	Riya	Shagun	2023-01-11
3	Ishika	Rathore	2023-01-12
4	Anika	Sethia	2023-01-13
5	Yashwardhan	Sant	2023-01-14
6	Nakul	Sethi	2023-01-15
7	Arjav	Kasliwal	2023-01-16
8	Pranav	Joshi	2023-01-17
9	Oshin	Ahuja	2023-01-18
10	Lakshya	Dubey	2023-01-19
11	Shanali	Singh	2023-01-20
12	Suhani	Khandelwal	2023-01-21
13	Kashish	Wadhwani	2023-01-22
14	Prakhar	Agrawal	2023-01-23
15	Ananya	Pandey	2023-01-24

13. Aggregates total revenue generated per product category

```
314 •   SELECT
315       cat.name AS CategoryName,
316       SUM(oi.price * oi.quantity) AS TotalRevenue
317     FROM Category cat
318     JOIN Product p ON cat.Category_id = p.Category_id
319     JOIN Order_item oi ON p.Product_id = oi.Product_id
320     GROUP BY cat.Category_id
321     ORDER BY TotalRevenue DESC;
```

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

CategoryName	TotalRevenue
Electronics	196985.00
Books	22996.00

14. Identify customers who have not placed any orders in the last year

```
323 •   SELECT
324       c.Customer_id,
325       CONCAT(c.first_name, ' ', c.last_name) AS CustomerName
326     FROM Customer c
327     WHERE NOT EXISTS (
328       SELECT 1
329       FROM Orders o
330       WHERE c.Customer_id = o.Customer_id AND o.Order_date >= CURDATE() - INTERVAL 1 YEAR
331     );
```

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

Customer_id	CustomerName
1	Arnav Chaturvedi
2	Riya Shagun
3	Ishika Rathore
4	Anika Sethia
5	Yashwardhan Sant
6	Nakul Sethi
7	Arjav Kasliwal
8	Pranav Joshi

15. Rank customers based on their total spending:

```
333 •   SELECT
334         c.Customer_id,
335         CONCAT(c.first_name, ' ', c.last_name) AS CustomerName,
336         SUM(o.total_price) AS TotalSpending,
337         RANK() OVER (ORDER BY SUM(o.total_price) DESC) AS SpendingRank
338     FROM
339         Customer c
340     JOIN
341         Orders o ON c.Customer_id = o.Customer_id
342     GROUP BY
343         c.Customer_id, c.first_name, c.last_name
344     ORDER BY
345         TotalSpending DESC;
```

Customer_id	CustomerName	TotalSpending	SpendingRank
9	Oshin Ahuja	323.99	1
8	Pranav Joshi	242.99	2
13	Kashish Wadhwani	226.79	3
12	Suhani Khandelwal	202.49	4
7	Arjav Kasliwal	199.99	5
15	Ananya Pandey	199.99	5
14	Prakhar Agrawal	179.99	7
11	Shanali Singh	159.99	8
10	Lakshya Dubey	129.99	9
5	Yashwardhan Sant	99.99	10
3	Ishika Rathore	89.99	11
1	Arnav Chaturvedi	79.99	12
4	Anika Sethia	59.99	13
2	Riya Shagun	49.99	14
6	Nakul Sethi	29.99	15

16. Customer Purchase Frequency Segmentation

```
SELECT
c.Customer_id,
CONCAT(c.first_name, ' ', c.last_name) AS CustomerName,
CASE
    WHEN COUNT(o.Order_id) >= 3 THEN 'Frequent Buyer'
    WHEN COUNT(o.Order_id) BETWEEN 1 AND 2 THEN 'Moderate Buyer'
    ELSE 'Rare Buyer'
END AS CustomerSegment
FROM Customer c
LEFT JOIN Orders o ON c.Customer_id = o.Customer_id
GROUP BY c.Customer_id
ORDER BY COUNT(o.Order_id) DESC;
```

Customer_id	CustomerName	CustomerSegment
1	Arnav Chaturvedi	Moderate Buyer
2	Riya Shagun	Moderate Buyer
3	Ishika Rathore	Moderate Buyer
4	Anika Sethia	Moderate Buyer
5	Yashwardhan Sant	Moderate Buyer
6	Nakul Sethi	Moderate Buyer
7	Arjav Kasliwal	Moderate Buyer
8	Pranav Joshi	Moderate Buyer
9	Oshin Ahuja	Moderate Buyer
10	Lakshya Dubey	Moderate Buyer
11	Shanali Singh	Moderate Buyer
12	Suhani Khandelwal	Moderate Buyer
13	Kashish Wadhwani	Moderate Buyer
14	Prakhar Agrawal	Moderate Buyer
15	Ananya Pandey	Moderate Buyer
16	Kabir Singh	Rare Buyer
17	Saanvi Kumar	Rare Buyer
18	Harshit Chawla	Rare Buyer
19	Khushi Upadhyay	Rare Buyer
20	Chhavi Ambor	Rare Buyer

17. Customers who add items to their cart but do not complete the purchase.

```
SELECT
    c.Customer_id,
    CONCAT(c.first_name, ' ', c.last_name) AS CustomerName,
    COUNT(DISTINCT cart.Cart_id) AS CartsCreated,
    COUNT(DISTINCT o.Order_id) AS OrdersCompleted,
    (COUNT(DISTINCT cart.Cart_id) - COUNT(DISTINCT o.Order_id)) / COUNT(DISTINCT cart.Cart_id) * 100 AS AbandonmentRate
FROM Customer c
LEFT JOIN Cart cart ON c.Customer_id = cart.Customer_id
LEFT JOIN Orders o ON c.Customer_id = o.Customer_id
GROUP BY c.Customer_id
HAVING CartsCreated > 0
ORDER BY AbandonmentRate DESC;
```

Customer_id	CustomerName	CartsCreated	OrdersCompleted	AbandonmentRate
1	Arnav Chaturvedi	1	1	0.0000
2	Riya Shagun	1	1	0.0000
3	Ishika Rathore	1	1	0.0000
4	Anika Sethia	1	1	0.0000
5	Yashwardhan Sant	1	1	0.0000
6	Nakul Sethi	1	1	0.0000
7	Arjav Kasliwal	1	1	0.0000
8	Pranav Joshi	1	1	0.0000
9	Oshin Ahuja	1	1	0.0000
10	Lakshya Dubey	1	1	0.0000
11	Shanali Singh	1	1	0.0000
12	Suhani Khandelwal	1	1	0.0000
13	Kashish Wadhwani	1	1	0.0000
14	Prakhar Agrawal	1	1	0.0000
15	Ananya Pandey	1	1	0.0000

18. Segment customers based on profitability.

```
SELECT
    c.Customer_id,
    CONCAT(c.first_name, ' ', c.last_name) AS CustomerName,
    SUM(o.total_price) AS TotalSpent,
    COUNT(o.Order_id) AS OrderCount,
    SUM(o.total_price) / COUNT(o.Order_id) AS AvgOrderValue,
    CASE
        WHEN SUM(o.total_price) / COUNT(o.Order_id) > 200 THEN 'High Value'
        WHEN SUM(o.total_price) / COUNT(o.Order_id) BETWEEN 200 AND 0 THEN 'Medium Value'
        ELSE 'Low Value'
    END AS CustomerSegment
FROM Orders o
JOIN Customer c ON o.Customer_id = c.Customer_id
GROUP BY c.Customer_id
ORDER BY TotalSpent DESC;
```

Customer_id	CustomerName	TotalSpent	OrderCount	AvgOrderValue	CustomerSegment
9	Oshin Ahuja	323.99	1	323.990000	High Value
8	Pranav Joshi	242.99	1	242.990000	High Value
13	Kashish Wadhwani	226.79	1	226.790000	High Value
12	Suhani Khandelwal	202.49	1	202.490000	High Value
7	Arjav Kasliwal	199.99	1	199.990000	Low Value
15	Ananya Pandey	199.99	1	199.990000	Low Value
14	Prakhar Agrawal	179.99	1	179.990000	Low Value
11	Shanali Singh	159.99	1	159.990000	Low Value
10	Lakshya Dubey	129.99	1	129.990000	Low Value
5	Yashwardhan Sant	99.99	1	99.990000	Low Value
3	Ishika Rathore	89.99	1	89.990000	Low Value
1	Arnav Chaturvedi	79.99	1	79.990000	Low Value
4	Anika Sethia	59.99	1	59.990000	Low Value
2	Riya Shagun	49.99	1	49.990000	Low Value
6	Nakul Sethi	29.99	1	29.990000	Low Value

19. Determine the count of unique customers who placed orders in the 1st month

```
389 •   SELECT
390         DATE_FORMAT(Order_date, '%Y-%m') AS Month,
391         COUNT(DISTINCT Customer_id) AS MonthlyActiveUsers
392     FROM Orders
393     GROUP BY Month
394     ORDER BY Month;
```

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

Month	MonthlyActiveUsers
2023-01	15

20. Identify products that have not been sold in the last month.

```
SELECT
    p.Product_id,
    p.description
FROM Product p
WHERE p.Product_id NOT IN (
    SELECT oi.Product_id
    FROM Order_item oi
    JOIN Orders o ON oi.Order_id = o.Order_id
    WHERE o.Order_date >= CURDATE() - INTERVAL 1 MONTH
)
ORDER BY p.Product_id;
```

Product_id	description
1	Smartphone
2	Laptop
3	Bluetooth Headphones
4	Smartwatch
5	Tablet
6	Wireless Charger

VI. Project demonstration

- This DBMS project showcases a functional e-commerce shopping cart system. We'll navigate the user interface, demonstrating product browsing, adding items to the cart, and product comparisons.
- We'll then process a secure checkout simulating a real-world purchase.
- Finally, we'll explore the database structure, highlighting tables storing product information, customer details, and shopping cart contents, emphasizing how this system efficiently manages online shopping transactions.

VII. Self -Learning beyond classroom

:

- **Explore Open-Source Shopping Cart Systems:** Dig into open-source platforms like WooCommerce or PrestaShop. These provide real-world examples of how databases are used in e-commerce and can inspire your project design. You can see how they handle product variations, manage stock, and secure user data.
- **Online Tutorials and Courses:** There are fantastic online resources! Look for tutorials on building shopping carts with specific DBMS software like MySQL or PostgreSQL. Many free courses offer a structured approach to learning the technical aspects.
- **Experiment with Sample Databases:** Many DBMS providers offer sample databases you can practice with. Experimenting with these will help you get comfortable with writing queries, managing data types, and understanding how data is organized within the database.
- **Join Online Communities:** There are vibrant online communities for database enthusiasts and e-commerce developers. Forums like Stack Overflow can be a great place to ask specific questions about challenges you encounter in your project and learn from others' experiences.
- **Build a Mock User Interface:** While your project focuses on the database side, don't forget the user experience! Sketch or create a basic user interface (UI) to visualize how users will interact with your shopping cart. This will help you identify data elements you need to store and retrieve effectively.

VIII. Learning from the Project

- **Data Modeling Expertise:** You'll develop practical skills in designing relational database schemas. This includes understanding data relationships, defining tables and columns effectively, and normalizing data to prevent redundancy and inconsistencies.
- **Problem-Solving Prowess:** This project throws real-world challenges at you, like handling product variations (size, color), managing stock levels in real-time, and ensuring data security for customer information. Solving these problems hones your critical thinking and problem-solving skills, valuable assets in any technical field.
- **E-commerce Systems Demystified:** You'll gain valuable insight into the inner workings of e-commerce platforms. From product data management to order processing, you'll understand how databases play a crucial role in powering these online stores.
- **Performance Optimization Importance:** As your 'store' grows virtually, you'll face the challenge of optimizing database queries and data structures. This teaches you the importance of efficient data retrieval and manipulation, a crucial skill for handling large datasets.
- **Integration Fundamentals:** Connecting your shopping cart with other systems, like a payment gateway or product catalog, might be necessary. This teaches you valuable skills in integrating different software components, a common task in modern software development.
- **Testing and Debugging Techniques:** Building a functional shopping cart involves rigorous testing. You'll learn to identify and fix errors in your database queries and logic, a vital skill for any developer or database administrator.
- **Security Awareness:** Protecting sensitive user data like addresses and payment details is paramount. This project emphasizes the importance of data security practices, like encryption and access control mechanisms.
- **Version Control Best Practices:** Using a version control system like Git is crucial for tracking changes, collaborating effectively, and reverting to previous versions if needed. You'll get hands-on experience with these essential practices.
- **Project Management Skills:** Planning, scheduling, and executing different project phases become important. You'll gain valuable experience in managing a software project, even if it's a smaller-scale one.
- **Technical Communication Enhancement:** Documenting your project, including data models, query logic, and design choices becomes important. This hones your technical communication skills, essential for explaining technical concepts to others.

IX. Challenges Faced

1. Data Modeling:
We need to decide how to structure our data in tables (like Products, Customers, Carts). This includes defining columns (like product name, price, quantity in cart) and their relationships (e.g., a cart belongs to a customer and contains products).

2. Handling Product Variations:
How would we handle products with different sizes, colors, or other variations? We need additional tables or clever data storage within a table to represent these variations.

3. Managing Stock Levels:
When a customer adds an item to their cart, we need to ensure the stock level is updated accurately to reflect real-time availability. This can get complex if multiple users try to buy the same item at once.

4. Security Considerations:
We need to make sure customer information like addresses and payment details are stored securely in the database. This might involve encryption and access control mechanisms.

5. Performance Optimization:
As our e-commerce store grows, the database needs to handle a larger number of users and transactions efficiently. We might need to optimize queries and consider database scaling techniques.

X. Conclusion

- Building an e-commerce shopping cart with a DBMS is a challenging but rewarding project.
- It provides practical experience in data modeling, problem-solving, understanding e-commerce systems, and the importance of performance and integration.
- These learned skills are valuable assets for anyone pursuing a career in database management, software development, or any field that involves data manipulation and system design.