E-Commerce Overview

The topic in focus for this assignment is the e-commerce database management system. E-commerce has revolutionized how customers shop for products from the comfort of their own homes. Since the pandemic, e-commerce has skyrocketed as customers and businesses adapted to a new way of life with social distancing and quarantine in effect. It has created a global marketplace that anyone from anywhere in the world can access. There have been many businesses that solely focus on connecting brands and companies with customers through online shopping. This complex ecosystem which provides an exceptional online shopping experience depends on a variety of functions and technology to ensure the efficiency as well as security of online transactions.

Key Functions

Many functions go on in the backend to ensure a customer's experience with e-commerce goes as efficiently as possible. One key function of online shopping is the catalog itself, this is where the business can display its products and where the customer can view the product and its details such as the description of the product, price and if it is available. This particular function allows for a smooth and efficient browsing experience for the customer. The next key function for e-commerce is managing the orders placed, which would track the order placed by the customer from when the order is placed to when it reaches the customer. This function is responsible for payment confirmation, shipping details, billing information etc. Having efficient order management lets the business ensure customer satisfaction. The most important key function would be the payment gateways for online businesses where transactions are made. The gateways ensure that the transactions are safe and accurate to the price. It would also be optimal for the gateways to have multiple secure payment methods such as credit cards, digital wallets and services like PayPal. These gateways are important in ensuring the transactions are safe and reducing the risk of fraud.

Information Flow

With many key functions that are crucial for online shopping to be conducted smoothly and safely, these systems must work together to give the customer a positive experience shopping online. The DBMS helps create, manage and control databases which ensure the smooth operation of online shopping. They are expected to synchronize the various types of data effectively for different key functions from the moment the customer visits the website to the customer receiving the product. The process starts with product catalogue management which allows the customer to view descriptions of a product and if interested, buy the product, this is then the responsibility of the order management system as it collects customer data such as

their shipping address, payment method and the status of their order until it is delivered to the customer. The most sensitive and vital DBMS is the payment gateway. After the customer data is collected, the customer has to use the payment gateway to pay for the product. This is where the financial aspect of the process is stored, the customer is asked to choose their method of payment and enter the required credentials to ensure the method of payment is valid and the transaction is completely secure. Using such steps allows the business to ensure they reduce the risk of fraud and protect the customer's sensitive information.

Entities

1. Admin
- Attributes:
  - ❖ Admin_ID (Primary Key)
  - ❖ Admin_name
  - ❖ Admin_role

2. E-commerce Website
- Attributes:
  - ❖ Website_name (Primary Key)
  - ❖ Website_URL

3. Category
- Attributes:
  - ❖ Category_ID (Primary Key)
  - ❖ Category_name

4. Product
- Attributes:
  - ❖ Product_ID (Primary Key)
  - ❖ Product_name
  - ❖ Quantity
  - ❖ Price

5. Cart
- Attributes:
  - ❖ Cart_id (Primary Key)
  - ❖ Product_id (Related to product entity)
  - ❖ Quantity

6. Supplier
- Attributes:
  - ❖ Supplier_ID (Primary Key)
  - ❖ Supplier_name
  - ❖ Supplier_address
    - ➔ Country
    - ➔ State

➔ City
➔ Street
➔ Unit
➔ Postal Code

7. Inventory
● Attributes:
❖ Batch_ID (Primary Key)
❖ Product_ID (Related to Product entity)
❖ Price
❖ Quantity

8. Customer
● Attributes:
❖ Customer_ID (Primary Key)
❖ Email
❖ Password
❖ Contact_number
❖ F_name
❖ L_name
❖ Address
➔ Country
➔ State
➔ City
➔ Street
➔ Unit
➔ Postal Code

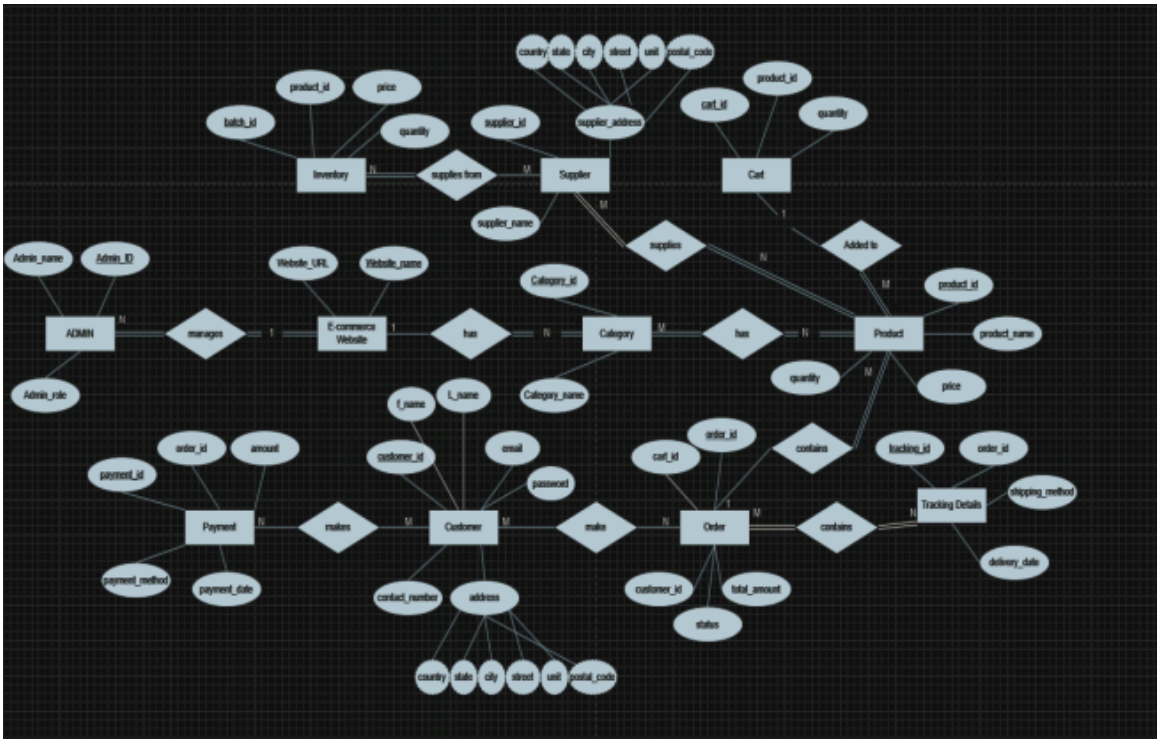
9. Order
● Attributes:
❖ Order_ID (Primary Key)
❖ Cart_ID (Related to Cart Entity)
❖ Customer_ID (Related to Customer Entity)
❖ Status
❖ Total Amount

10.Tracking Details
● Attributes:
❖ Tracking_ID (Primary key)
❖ Order_ID (Related to Order Entity)
❖ Shipping_Method
❖ Delivery_date

11.Payment

● Attributes:
❖ Payment_ID (Primary Key)
❖ Order_ID (Related to Order Entity)
❖ Payment_Method
❖ Payment_date
❖ Amount

ER for E-Commerce Website



SQL Queries

Filling the data
-- 1. Insert into Admin

INSERT INTO Admin (Admin_ID, Admin_name, Admin_role)
VALUES (1, 'John Doe', 'Manager');
INSERT INTO Admin (Admin_ID, Admin_name, Admin_role)
VALUES (2, 'Jane Smith', 'Assistant');

-- 3. Insert into Category

```
INSERT INTO Category (Category_id, Category_name)
VALUES (1, 'Electronics');
INSERT INTO Category (Category_id, Category_name)
VALUES (2, 'Clothing');
INSERT INTO Category (Category_id, Category_name)
VALUES (3, 'Books');

-- 4. Insert into Product
INSERT INTO Product (Product_id, Product_name, Price, Category_id)
VALUES (1, 'Laptop', 1200.00, 1);
INSERT INTO Product (Product_id, Product_name, Price, Category_id)
VALUES (2, 'T-shirt', 25.00, 2);
INSERT INTO Product (Product_id, Product_name, Price, Category_id)
VALUES (3, 'Novel', 15.00, 3);

-- 5. Insert into Customer
INSERT INTO Customer (Customer_id, F_name, L_name, Email, Password, Contact_number,
Address)
VALUES (1, 'Alice', 'Johnson', 'alice@example.com', 'password123', '123-456-7890', '123 Main
St');
INSERT INTO Customer (Customer_id, F_name, L_name, Email, Password, Contact_number,
Address)
VALUES (2, 'Bob', 'Williams', 'bob@example.com', 'password456', '987-654-3210', '456 Park
Ave');

-- 6. Insert into "Order"
INSERT INTO "Order" (Order_id, Customer_id, Total_amount, Status)
VALUES (1, 1, 1225.00, 'Shipped');
INSERT INTO "Order" (Order_id, Customer_id, Total_amount, Status)
VALUES (2, 2, 40.00, 'Pending');

-- 7. Insert into Cart
INSERT INTO Cart (Cart_id, Product_id, Quantity)
VALUES (1, 1, 1);
INSERT INTO Cart (Cart_id, Product_id, Quantity)
VALUES (2, 2, 3);
INSERT INTO Cart (Cart_id, Product_id, Quantity)
VALUES (3, 3, 2);

-- 8. Insert into Supplier
INSERT INTO Supplier (Supplier_id, Supplier_name, Supplier_address)
VALUES (1, 'TechCorp', '789 Industrial Rd');
INSERT INTO Supplier (Supplier_id, Supplier_name, Supplier_address)
VALUES (2, 'FashionHub', '101 Fashion St');
```

```
-- 9. Insert into Inventory (without Supplier_id)
INSERT INTO Inventory (Batch_id, Product_id, Quantity, Price)
VALUES (1, 1, 50, 1100.00);
INSERT INTO Inventory (Batch_id, Product_id, Quantity, Price)
VALUES (2, 2, 200, 20.00);
INSERT INTO Inventory (Batch_id, Product_id, Quantity, Price)
VALUES (3, 3, 150, 12.00);

-- 10. Insert into Payment
INSERT INTO Payment (Payment_id, Order_id, Amount, Payment_method, Payment_date)
VALUES (1, 1, 1225.00, 'Credit Card', TO_DATE('2024-09-20', 'YYYY-MM-DD'));
INSERT INTO Payment (Payment_id, Order_id, Amount, Payment_method, Payment_date)
VALUES (2, 2, 40.00, 'PayPal', TO_DATE('2024-09-22', 'YYYY-MM-DD'));

-- 11. Insert into TrackingDetails
INSERT INTO TrackingDetails (Tracking_id, Order_id, Shipping_method, Delivery_date)
VALUES (1, 1, 'Standard Shipping', TO_DATE('2024-09-25', 'YYYY-MM-DD'));
INSERT INTO TrackingDetails (Tracking_id, Order_id, Shipping_method, Delivery_date)
VALUES (2, 2, 'Express Shipping', TO_DATE('2024-09-27', 'YYYY-MM-DD'));
```

```
--  1.  Select All Admins: Retrieves all records from the Admin table.
SELECT * FROM Admin;

--  2.  Count Number of Customers: Counts the total number of entries in the Customer table.
SELECT COUNT(*) AS Total_Customers FROM Customer;

--  3.  List All Products with their Categories: Joins Product and Category tables to display each
product along with its category name.
SELECT P.Product_id, P.Product_name, C.Category_name
FROM Product P
JOIN Category C ON P.Category_id = C.Category_id;

--  4.  Select All Orders and Corresponding Customer Names: Joins the Order and Customer
tables to list orders with customer names and total amounts.
SELECT O.Order_id, C.F_name, C.L_name, O.Total_amount, O.Status
FROM "Order" O
JOIN Customer C ON O.Customer_id = C.Customer_id;

--  5.  Get Inventory Details for a Specific Product: Fetches details about a specific product in
the Inventory table. You can change the Product_id for different products.
SELECT I.Batch_id, I.Quantity, I.Price
```

```
FROM Inventory I
WHERE I.Product_id = 1;  -- Change Product_id as needed

--    6.    Total Sales Amount by Customer: Groups by customer to sum their total sales.
SELECT C.Customer_id, C.F_name, C.L_name, SUM(O.Total_amount) AS Total_Sales
FROM Customer C
JOIN "Order" O ON C.Customer_id = O.Customer_id
GROUP BY C.Customer_id, C.F_name, C.L_name;

--    7.    Get All Payments and their Corresponding Orders: Joins the Payment and Order tables
to show each payment associated with an order.
SELECT P.Payment_id, P.Amount, P.Payment_method, O.Order_id
FROM Payment P
JOIN "Order" O ON P.Order_id = O.Order_id;
```

```
-- Query for Admin Table: List all Admins in the system
SELECT Admin_ID, Admin_name AS "Admin Name", Admin_role AS "Admin Role"
FROM Admin
ORDER BY Admin_name ASC;

-- Query for Customer Table: Total Customers and Distinct Emails
SELECT COUNT(*) AS "Total Customers", COUNT(DISTINCT Email) AS "Distinct Emails"
FROM Customer;

-- Query for Product Table: Products with their Categories
SELECT P.Product_id AS "Product ID", P.Product_name AS "Product Name",
C.Category_name AS "Category"
FROM Product P
JOIN Category C ON P.Category_id = C.Category_id
ORDER BY P.Product_name ASC;

-- Query for Order Table: List of Orders with Customer Details
SELECT O.Order_id AS "Order ID", C.F_name AS "First Name", C.L_name AS "Last Name",
      O.Total_amount AS "Total Amount", O.Status AS "Order Status"
FROM "Order" O
JOIN Customer C ON O.Customer_id = C.Customer_id
ORDER BY O.Order_id DESC;

-- Query for Cart Table: Count Items in Each Cart
SELECT Cart_id AS "Cart ID", COUNT(DISTINCT Product_id) AS "Number of Unique Items"
FROM Cart
GROUP BY Cart_id
ORDER BY "Number of Unique Items" DESC;
```

```
-- Query for Inventory Table: Available Inventory for Products
SELECT Batch_id AS "Batch ID", Product_id AS "Product ID", Quantity, Price
FROM Inventory
WHERE Quantity > 0
ORDER BY Quantity DESC;

-- Query for Supplier Table: List all Suppliers
SELECT Supplier_id AS "Supplier ID", Supplier_name AS "Supplier Name", Supplier_address
AS "Supplier Address"
FROM Supplier
ORDER BY Supplier_name ASC;

-- Query for Payment Table: Payments Made by Customers
SELECT P.Payment_id AS "Payment ID", P.Amount AS "Amount", P.Payment_method AS
"Payment Method",
      O.Order_id AS "Order ID"
FROM Payment P
JOIN "Order" O ON P.Order_id = O.Order_id
ORDER BY P.Payment_date DESC;

-- Query for Tracking Details Table: Orders and their Tracking Information
SELECT TD.Tracking_id AS "Tracking ID", O.Order_id AS "Order ID", TD.Shipping_method AS
"Shipping Method",
      TD.Delivery_date AS "Delivery Date"
FROM TrackingDetails TD
JOIN "Order" O ON TD.Order_id = O.Order_id
ORDER BY TD.Delivery_date ASC;
```

```
-- Advanced Join Query 1: Retrieve all orders, products, and customers
-- Select specific fields for display, aliasing them for clarity
SELECT O.Order_id AS "Order ID",         -- Display the unique Order ID
      C.F_name AS "Customer First Name",-- Display the first name of the customer who placed
the order
      C.L_name AS "Customer Last Name", -- Display the last name of the customer
      P.Product_name AS "Product Name", -- Display the name of the product(s) ordered
      O.Total_amount AS "Order Total"   -- Display the total amount of the order

-- Specify the main table (Order) and join it with related tables
FROM "Order" O
-- Join the Order table with the Customer table to get customer information
JOIN Customer C ON O.Customer_id = C.Customer_id
```

```
-- Join the Order table with the Cart table to link orders with products (assuming each order has associated cart entries)
JOIN Cart CT ON O.Order_id = CT.Cart_id
-- Join the Cart table with the Product table to retrieve product details for each cart entry
JOIN Product P ON CT.Product_id = P.Product_id
-- Order the results by Order ID in descending order, so the most recent orders appear first
ORDER BY O.Order_id DESC;


-- Advanced Join Query 2: Total Sales Amount by Customer and Product Category

-- Select specific fields to display, aliasing them for clarity
SELECT C.Customer_id AS "Customer ID",         -- Display the unique Customer ID
       C.F_name AS "Customer First Name",      -- Display the customer's first name
       C.L_name AS "Customer Last Name",       -- Display the customer's last name
       SUM(O.Total_amount) AS "Total Sales",   -- Calculate and display the total sales amount for each customer
       CAT.Category_name AS "Product Category" -- Display the name of the product category purchased by the customer

-- Specify the main table (Customer) and join it with related tables
FROM Customer C
-- Join the Customer table with the Order table to link customers with their orders
JOIN "Order" O ON C.Customer_id = O.Customer_id
-- Join the Order table with the Cart table to link orders with products (assuming the Cart stores product details for each order)
JOIN Cart CT ON O.Order_id = CT.Cart_id
-- Join the Cart table with the Product table to retrieve product details from each cart entry
JOIN Product P ON CT.Product_id = P.Product_id
-- Join the Product table with the Category table to retrieve the category of each product
JOIN Category CAT ON P.Category_id = CAT.Category_id
-- Group the results by customer details and product category to aggregate total sales for each combination
GROUP BY C.Customer_id, C.F_name, C.L_name, CAT.Category_name
-- Order the results by the total sales amount in descending order, so the customers with the highest total sales appear first
ORDER BY "Total Sales" DESC;
```

```
-- View 1 - Customer Order History for each customer

-- Create a view that shows the order history for each customer
CREATE VIEW CustomerOrderHistory AS
SELECT C.Customer_id AS "Customer ID",         -- Display the unique Customer ID
       C.F_name AS "First Name",               -- Display the first name of the customer
```

```
    C.L_name AS "Last Name",              -- Display the last name of the customer
    O.Order_id AS "Order ID",             -- Display the unique Order ID for each order
    O.Total_amount AS "Order Total"       -- Display the total amount for the corresponding
order

-- Specify the main table (Customer) and join it with the Order table
FROM Customer C
JOIN "Order" O ON C.Customer_id = O.Customer_id; -- Link customers to their orders using the
Customer_id

-- View 2 Checks inventory and Available Products

-- Create a view that shows all available products in stock
CREATE VIEW AvailableProducts AS
SELECT P.Product_id AS "Product ID",          -- Display the unique Product ID
    P.Product_name AS "Product Name",       -- Display the name of the product
    I.Quantity AS "Stock Available"         -- Display the available quantity of the product in stock

-- Specify the main table (Product) and join it with the Inventory table
FROM Product P
JOIN Inventory I ON P.Product_id = I.Product_id -- Link products to their inventory details using
Product_id
WHERE I.Quantity > 0;                       -- Only include products where the stock quantity is
greater than zero

-- View 3 Payment Details for tracking and analysis

-- Create a view that shows payment details along with the associated orders and customers
CREATE VIEW PaymentDetails AS
SELECT P.Payment_id AS "Payment ID",          -- Display the unique Payment ID
    P.Amount AS "Payment Amount",           -- Display the payment amount made by the
customer
    P.Payment_method AS "Payment Method",   -- Display the payment method (e.g., credit
card, cash)
    O.Order_id AS "Order ID",               -- Display the unique Order ID associated with the
payment
    C.F_name AS "Customer First Name",      -- Display the first name of the customer who
made the payment
    C.L_name AS "Customer Last Name"        -- Display the last name of the customer who
made the payment

-- Specify the main table (Payment) and join it with related tables
FROM Payment P
```

```
JOIN "Order" O ON P.Order_id = O.Order_id      -- Link payments to their respective orders using Order_id
JOIN Customer C ON O.Customer_id = C.Customer_id; -- Link orders to customers to retrieve customer details
```

Inserting Data

```
-- Insert into Admin
INSERT INTO Admin (Admin_ID, Admin_name, Admin_role) VALUES (1, 'John Doe', 'Manager');
INSERT INTO Admin (Admin_ID, Admin_name, Admin_role) VALUES (2, 'Jane Smith', 'Assistant');

-- Insert into Category
INSERT INTO Category (Category_id, Category_name) VALUES (1, 'Electronics');
INSERT INTO Category (Category_id, Category_name) VALUES (2, 'Clothing');
INSERT INTO Category (Category_id, Category_name) VALUES (3, 'Books');

-- Insert into Product
INSERT INTO Product (Product_id, Product_name, Price, Category_id) VALUES (1, 'Laptop', 1200.00, 1);
INSERT INTO Product (Product_id, Product_name, Price, Category_id) VALUES (2, 'T-shirt', 25.00, 2);
INSERT INTO Product (Product_id, Product_name, Price, Category_id) VALUES (3, 'Novel', 15.00, 3);

-- Insert into Customer
INSERT INTO Customer (Customer_id, F_name, L_name, Email, Password, Contact_number, Address)
VALUES (1, 'Alice', 'Johnson', 'alice@example.com', 'password123', '123-456-7890', '123 Main St');
INSERT INTO Customer (Customer_id, F_name, L_name, Email, Password, Contact_number, Address)
VALUES (2, 'Bob', 'Williams', 'bob@example.com', 'password456', '987-654-3210', '456 Park Ave');

-- Insert into "Order"
INSERT INTO "Order" (Order_id, Customer_id, Total_amount, Status)
VALUES (1, 1, 1225.00, 'Shipped');
INSERT INTO "Order" (Order_id, Customer_id, Total_amount, Status)
VALUES (2, 2, 40.00, 'Pending');
```

```
-- Insert into Cart
INSERT INTO Cart (Cart_id, Product_id, Quantity)
VALUES (1, 1, 1);
INSERT INTO Cart (Cart_id, Product_id, Quantity)
VALUES (2, 2, 3);
INSERT INTO Cart (Cart_id, Product_id, Quantity)
VALUES (3, 3, 2);

-- Insert into Supplier
INSERT INTO Supplier (Supplier_id, Supplier_name, Supplier_address)
VALUES (1, 'TechCorp', '789 Industrial Rd');
INSERT INTO Supplier (Supplier_id, Supplier_name, Supplier_address)
VALUES (2, 'FashionHub', '101 Fashion St');

-- Insert into Inventory
INSERT INTO Inventory (Batch_id, Product_id, Quantity, Price)
VALUES (1, 1, 50, 1100.00);
INSERT INTO Inventory (Batch_id, Product_id, Quantity, Price)
VALUES (2, 2, 200, 20.00);
INSERT INTO Inventory (Batch_id, Product_id, Quantity, Price)
VALUES (3, 3, 150, 12.00);

-- Insert into Payment
INSERT INTO Payment (Payment_id, Order_id, Amount, Payment_method, Payment_date)
VALUES (1, 1, 1225.00, 'Credit Card', TO_DATE('2024-09-20', 'YYYY-MM-DD'));
INSERT INTO Payment (Payment_id, Order_id, Amount, Payment_method, Payment_date)
VALUES (2, 2, 40.00, 'PayPal', TO_DATE('2024-09-22', 'YYYY-MM-DD'));

-- Insert into TrackingDetails
INSERT INTO TrackingDetails (Tracking_id, Order_id, Shipping_method, Delivery_date)
VALUES (1, 1, 'Standard Shipping', TO_DATE('2024-09-25', 'YYYY-MM-DD'));
INSERT INTO TrackingDetails (Tracking_id, Order_id, Shipping_method, Delivery_date)
VALUES (2, 2, 'Express Shipping', TO_DATE('2024-09-27', 'YYYY-MM-DD'));
```

Data Retrieval Queries

```
-- Retrieve All Admins
SELECT Admin_ID, Admin_name AS "Admin Name", Admin_role AS "Admin Role"
FROM Admin
ORDER BY Admin_name ASC;

-- Count Total Customers and Distinct Emails
SELECT COUNT(*) AS "Total Customers", COUNT(DISTINCT Email) AS "Distinct Emails"
```

```
FROM Customer;

-- List Products and their Categories
SELECT P.Product_id AS "Product ID", P.Product_name AS "Product Name",
C.Category_name AS "Category"
FROM Product P
JOIN Category C ON P.Category_id = C.Category_id
ORDER BY P.Product_name ASC;

-- List Orders and Corresponding Customer Names
SELECT O.Order_id AS "Order ID", C.F_name AS "First Name", C.L_name AS "Last Name",
        O.Total_amount AS "Total Amount", O.Status AS "Order Status"
FROM "Order" O
JOIN Customer C ON O.Customer_id = C.Customer_id
ORDER BY O.Order_id DESC;

-- Count Items in Each Cart
SELECT Cart_id AS "Cart ID", COUNT(DISTINCT Product_id) AS "Number of Unique Items"
FROM Cart
GROUP BY Cart_id
ORDER BY "Number of Unique Items" DESC;

-- Show Inventory for Products
SELECT Batch_id AS "Batch ID", Product_id AS "Product ID", Quantity, Price
FROM Inventory
WHERE Quantity > 0
ORDER BY Quantity DESC;

-- List All Suppliers
SELECT Supplier_id AS "Supplier ID", Supplier_name AS "Supplier Name", Supplier_address
AS "Supplier Address"
FROM Supplier
ORDER BY Supplier_name ASC;

-- Payments Made by Customers
SELECT P.Payment_id AS "Payment ID", P.Amount AS "Amount", P.Payment_method AS
"Payment Method",
        O.Order_id AS "Order ID"
FROM Payment P
JOIN "Order" O ON P.Order_id = O.Order_id
ORDER BY P.Payment_date DESC;

-- Tracking Details for Orders
```

```
SELECT TD.Tracking_id AS "Tracking ID", O.Order_id AS "Order ID", TD.Shipping_method AS
"Shipping Method",
        TD.Delivery_date AS "Delivery Date"
FROM TrackingDetails TD
JOIN "Order" O ON TD.Order_id = O.Order_id
ORDER BY TD.Delivery_date ASC;
```

Views for Easier Future Quering

```
-- View: Customer Order History
CREATE VIEW CustomerOrderHistory AS
SELECT C.Customer_id AS "Customer ID", C.F_name AS "First Name", C.L_name AS "Last
Name",
        O.Order_id AS "Order ID", O.Total_amount AS "Order Total"
FROM Customer C
JOIN "Order" O ON C.Customer_id = O.Customer_id;

-- View: Available Products (in stock)
CREATE VIEW AvailableProducts AS
SELECT P.Product_id AS "Product ID", P.Product_name AS "Product Name", I.Quantity AS
"Stock Available"
FROM Product P
JOIN Inventory I ON P.Product_id = I.Product_id
WHERE I.Quantity > 0;

-- View: Payment Details (for tracking)
CREATE VIEW PaymentDetails AS
SELECT P.Payment_id AS "Payment ID", P.Amount AS "Payment Amount", P.Payment_method
AS "Payment Method",
        O.Order_id AS "Order ID", C.F_name AS "Customer First Name", C.L_name AS
"Customer Last Name"
FROM Payment P
JOIN "Order" O ON P.Order_id = O.Order_id
JOIN Customer C ON O.Customer_id = C.Customer_id;
```

Primary Keys and FDs

1. Admin Table

Primary Key: Admin_ID
Functional Dependencies:
Admin_ID → Admin_name, Admin_role

---

2. Category Table
Primary Key: Category_id
Functional Dependencies:
Category_id → Category_name

---

3. Product Table
Primary Key: Product_id
Functional Dependencies:
Product_id → Product_name, Price, Category_id

---

4. Customer Table
Primary Key: Customer_id
Functional Dependencies:
Customer_id → F_name, L_name, Email, Password, Contact_number, Address
Email → Customer_id, F_name, L_name, Password, Contact_number, Address

---

5. Order Table
Primary Key: Order_id
Functional Dependencies:
Order_id → Customer_id, Total_amount, Status

---

6. Cart Table
Primary Key: Cart_id
Functional Dependencies:
Cart_id → Product_id, Quantity

---

7. Supplier Table
Primary Key: Supplier_id

Functional Dependencies:
Supplier_id → Supplier_name, Supplier_address

---

8. Inventory Table
Primary Key: Batch_id
Functional Dependencies:
Batch_id → Product_id, Quantity, Price

---

9. Payment Table
Primary Key: Payment_id
Functional Dependencies:
Payment_id → Order_id, Amount, Payment_method, Payment_date

---

10. TrackingDetails Table
Primary Key: Tracking_id
Functional Dependencies:
Tracking_id → Order_id, Shipping_method, Delivery_date

Table Decomposition

Example: Non-normalized table of Order records

Initial Table (Non-Normalized)

OrderID	CustomerName	CustomerAddress	Product	ProductCategory	Price
1	Alice Smith	123 Elm St.	Laptop	Electronics	1200
2	Bob Jones	456 Oak St.	T-shirt	Clothing	25
3	Alice Smith	123 Elm St.	Novel	Books	15
4	Alice Smith	123 Elm St.	T-shirt	Clothing	25

First Normal Form (1NF)

In 1NF, each column must contain atomic values with no repeating groups.

First Normal Form (1NF)

OrderID	CustomerName	CustomerAddress	Product	ProductCategory	Price
1	Alice Smith	123 Elm St.	Laptop	Electronics	1200
2	Bob Jones	456 Oak St.	T-shirt	Clothing	25
3	Alice Smith	123 Elm St.	Novel	Books	15
4	Alice Smith	123 Elm St.	T-shirt	Clothing	25

Second Normal Form (2NF)

Order Table

Order Table

OrderID	Product	ProductCategory	Price	Quantity	TotalAmount
1	Laptop	Electronics	1200	1	1200
2	T-shirt	Clothing	25	2	50
3	Novel	Books	15	3	45
4	T-shirt	Clothing	25	1	25

Customer Table

CustomerID	CustomerName	CustomerAddress
1	Alice Smith	123 Elm St.
2	Bob Jones	456 Oak St.

Order-Customer Table

OrderID	CustomerID
1	1
2	2
3	1
4	1

Product Table

ProductID	ProductName	ProductCategory	Price
1	Laptop	Electronics	1200
2	T-shirt	Clothing	25
3	Novel	Books	15

Revised Order Table

OrderID	ProductID	Quantity	TotalAmount
1	1	1	1200
2	2	2	50
3	3	3	45
4	2	1	25

Customer Table

CustomerID	CustomerName	CustomerAddress
1	Alice Smith	123 Elm St.
2	Bob Jones	456 Oak St.

Order-Customer Table

OrderID	CustomerID
1	1
2	2
3	1
4	1

Review Tables for BCNF Compliance

1. Admin Table
• Primary Key: Admin_ID
• Attributes: Admin_name, Admin_role
• Dependency Analysis: All attributes depend solely on Admin_ID, the primary key. There are no partial or transitive dependencies.
• **Conclusion: The table is in BCNF.**

2. Category Table
• Primary Key: Category_id
• Attributes: Category_name
• Dependency Analysis: Category_name depends directly on Category_id. No other dependencies exist.
• **Conclusion: The table is in BCNF.**

3. Product Table
• Primary Key: Product_id
• Attributes: Product_name, Price, Category_id
• Dependency Analysis: Each attribute depends directly on Product_id. Category_id is a foreign key, preserving referential integrity.
• **Conclusion: The table is in BCNF.**

4. Customer Table
• Primary Key: Customer_id
• Attributes: F_name, L_name, Email, Password, Contact_number, Address
• Dependency Analysis: All attributes depend on Customer_id, and there are no partial or transitive dependencies.
• **Conclusion: The table is in BCNF.**

5. Order Table
• Primary Key: Order_id
• Attributes: Customer_id, Total_amount, Status
• Dependency Analysis: Each attribute depends directly on Order_id. Customer_id is a foreign key, ensuring referential integrity.
• **Conclusion: The table is in BCNF.**

6. Cart Table
• Composite Primary Key: (Cart_id, Product_id)
• Attributes: Quantity
• Dependency Analysis: Quantity depends on the composite key (Cart_id, Product_id), and there are no transitive dependencies.
• **Conclusion**: The table is in BCNF.

7. Supplier Table
• Primary Key: Supplier_id
• Attributes: Supplier_name, Supplier_address
• Dependency Analysis: All attributes depend on Supplier_id, and there are no additional dependencies.
• **Conclusion:** The table is in BCNF.

8. Inventory Table
• Primary Key: Batch_id
• Attributes: Product_id, Quantity, Price, Supplier_id
• Dependency Analysis: Each attribute depends on Batch_id, and Supplier_id and Product_id are foreign keys.
• Conclusion: The table is in BCNF.
9. Payment Table
• Primary Key: Payment_id
• Attributes: Order_id, Amount, Payment_method, Payment_date
• Dependency Analysis: All attributes are fully dependent on Payment_id, with Order_id as a foreign key.
• Conclusion: The table is in BCNF.
10. TrackingDetails Table
• Primary Key: Tracking_id
• Attributes: Order_id, Shipping_method, Delivery_date
• Dependency Analysis: All attributes depend solely on Tracking_id, with Order_id as a foreign key to maintain integrity.
• **Conclusion:** The table is in BCNF.

Partial dependencies and Functional Dependencies and Algorithms

Step 1: Identify and List Functional Dependencies (FDs)
For each table, identify primary keys, candidate keys, and functional dependencies, noting any transitive or partial dependencies.
1. Admin Table
Primary Key: Admin_ID
FDs: Admin_ID → Admin_name, Admin_role
Analysis: Admin_ID is a superkey, and all other attributes depend on it. This table is already in BCNF.
2. Category Table
Primary Key: Category_id
FDs: Category_id → Category_name
Analysis: Category_id is a superkey, so this table is already in BCNF.
3. Product Table
Primary Key: Product_id
FDs: Product_id → Product_name, Price, Category_id

Partial Dependency: Category_id → Category_name
Problem: The dependency Category_id → Category_name creates a partial dependency since Category_id is not a superkey of the Product table.
Solution: Decompose using the Bernstein algorithm to remove partial dependency and reach BCNF
(explained in Step 2).
4. Customer Table
Primary Key: Customer_id
FDs: Customer_id → F_name, L_name, Email, Password, Contact_number, Address
Analysis: Customer_id is a superkey, so the table meets BCNF.
5. Order Table
Primary Key: Order_id
FDs: Order_id → Customer_id, Total_amount, Status
Analysis: Order_id is a superkey, meaning it satisfies BCNF.
6. Cart Table
Primary Key: Composite key of Cart_id, Product_id
FDs: Cart_id, Product_id → Quantity
Analysis: The composite key (Cart_id, Product_id) is the superkey, so the table is in BCNF.
7. Supplier Table
Primary Key: Supplier_id
FDs: Supplier_id → Supplier_name, Supplier_address
Analysis: Supplier_id is a superkey, meaning the table meets BCNF.
8. Inventory Table
Primary Key: Batch_id
FDs: Batch_id → Product_id, Quantity, Price
Partial Dependency: Product_id → Product_name, Category_id
Problem: Product_id determines other attributes (Product_name and Category_id) and is not a superkey in the Inventory table.
Solution: Decompose using the Bernstein algorithm to reach BCNF (explained in Step 2).

9. Payment Table
Primary Key: Payment_id
FDs: Payment_id → Order_id, Amount, Payment_method, Payment_date
Analysis: Payment_id is a superkey, making this table BCNF-compliant.
10. TrackingDetails Table
Primary Key: Tracking_id
FDs: Tracking_id → Order_id, Shipping_method, Delivery_date
Analysis: Tracking_id is a superkey, satisfying BCNF
Step 2: Applying the Bernstein Algorithm for BCNF Decomposition
For tables with partial and transitive dependencies, use the Bernstein algorithm to reach BCNF. Here's the process applied to the Product and Inventory tables:
Decomposition Example 1: Product Table
Initial FDs:
• Product_id → Product_name, Price, Category_id

• Category_id → Category_name (partial dependency)
Steps:
1. Identify that Category_id → Category_name is a partial dependency.
2. Decompose Product into two tables:g
• Product: Product_id → Product_name, Price, Category_id
• Category: Category_id → Category_name
3. Result: Each table is now in BCNF.
Decomposition Example 2: Inventory Table
Initial FDs:
• Batch_id → Product_id, Quantity, Price
• Product_id → Product_name, Category_id (partial dependency)
Steps:
1. Identify that Product_id → Product_name, Category_id creates a partial dependency.
2. Decompose Inventory as follows:
• Inventory: Batch_id → Quantity, Price
• Inventory_Product: Batch_id, Product_id → Product_name, Category_id
3. Result: Each table now satisfies BCNF.
Step 3: Explain the Normalization Process and Algorithm Use
For the oral evaluation, be prepared to explain the following points:
1. BCNF Definition: Explain that a table is in BCNF if, for every FD, the LHS is a superkey.
2. Why Decomposition Was Needed:
Product Table: Category_id → Category_name meant the table was not in BCNF because Category_id was not a superkey. Decomposing into separate Product and Category tables ensured
BCNF compliance.
Inventory Table: The FD Product_id → Product_name, Category_id indicated a partial dependency
on Product_id. Decomposition allowed the main inventory details to be separate from product-specific
data, meeting BCNF.
3. Bernstein Algorithm Process:
Step-by-step, the Bernstein algorithm identifies dependencies, isolates partial dependencies, and
decomposes tables until only superkey FDs remain. By applying the algorithm, we ensure each table
satisfies BCNF.

Menu with UI

```
Menu:
1. Create Tables
2. Drop Tables
3. Populate Tables
4. Query Data
5. Exit
Choose an option: 1
All tables created successfully!
```

```
Menu:
1. Create Tables
2. Drop Tables
3. Populate Tables
4. Query Data
5. Exit
Choose an option: 2
Table dropped: TrackingDetails
Table dropped: Payment
Table dropped: Inventory
Table dropped: Supplier
Table dropped: Cart
Table dropped: "Order"
Table dropped: Customer
Table dropped: Product
Table dropped: Category
Table dropped: Admin
```

```
Menu:
1. Create Tables
2. Drop Tables
3. Populate Tables
4. Query Data
5. Exit
Choose an option: 4
Admin_ID: 1, Name: John Doe, Role: Manager
Admin_ID: 2, Name: Jane Smith, Role: Assistant
```