BIG DATA FOR MANAGERS AND ANALYTICS

ENTITY-RELATION DIAGRAM FOR FLIPKART ECOMMERCE PLATFORM USING TOP-DOWN APPROACH

SUBMITTED TO:

PROF. ASHOK HARNAL

SUBMITTED BY:

MEGHA GARG (045030)
SHOBHIT GUPTA (045041)
SANSKRITI BAHL (045050)

INTRODUCTION

This report presents a comprehensive analysis of the Entity-Relationship Diagram (ERD) created for Flipkart, a leading e-commerce platform in India. The ERD was developed using a top-down approach to model the core entities and relationships within Flipkart's database system.

E-commerce platforms like Flipkart require robust and scalable database designs to handle vast amounts of data, from user information and product catalogs to order processing and inventory management. The ERD serves as a crucial blueprint for designing and implementing such a database system, ensuring efficient data storage, retrieval, and manipulation.

In this project, we focused on creating a streamlined yet comprehensive ERD that captures the essential components of Flipkart's operations. This design aims to support key functionalities such as user management, product categorization, order processing, payment handling and order details

In developing an optimized database system for an e-commerce platform like Flipkart, a top-down approach is employed to ensure a robust and scalable architecture. The process begins by identifying high-level requirements, focusing on the core entities such as customers, products, orders, and payments, along with their key relationships and functionalities necessary to support seamless operations. This initial phase is followed by the conceptual design, where an Entity-Relationship (ER) diagram is created to map out these entities and their interactions, abstracting away the complexities of data storage. The model is then normalized, decomposing it into smaller entities to eliminate redundancy and avoid data anomalies while maintaining logical consistency. Next, the conceptual model is translated into a logical design, where it is represented as a set of normalized tables and relationships, independent of any specific database management system (DBMS). In the final phase, the logical design is transformed into a physical design tailored to the chosen DBMS, involving the definition of indexes, storage parameters, and other implementation-specific details. This structured approach ensures that the database system is not only aligned with Flipkart's operational needs but is also flexible enough to support future growth and technological advancements.

OBJECTIVES

The primary objectives of this project were to:

- Design a comprehensive ERD representing Flipkart's e-commerce ecosystem
- Identify key entities and their relationships using top down approach in database system.
- Establish a foundation for database implementation for Flipkart
- Provide insights into the platform's data structure

IDENTIFYING HIGH-LEVEL REQUIREMENTS

A comprehensive analysis was undertaken to identify the high-level requirements of Flipkart's e-commerce platform involving understanding the core functionalities that the database system must support, including **user management, product cataloging, order processing, payment handling, and order details** The focus is on recognizing the primary entities—such as Users, Products, Orders, Payments, Categories, Order Detail—and their interrelationships to ensure that the database design aligns with the operational needs and business objectives of the platform.

Determined attributes for each entity: For each entity, we defined relevant attributes. For example, Users have attributes like **user ID, name, email, password, address, and phone number.**

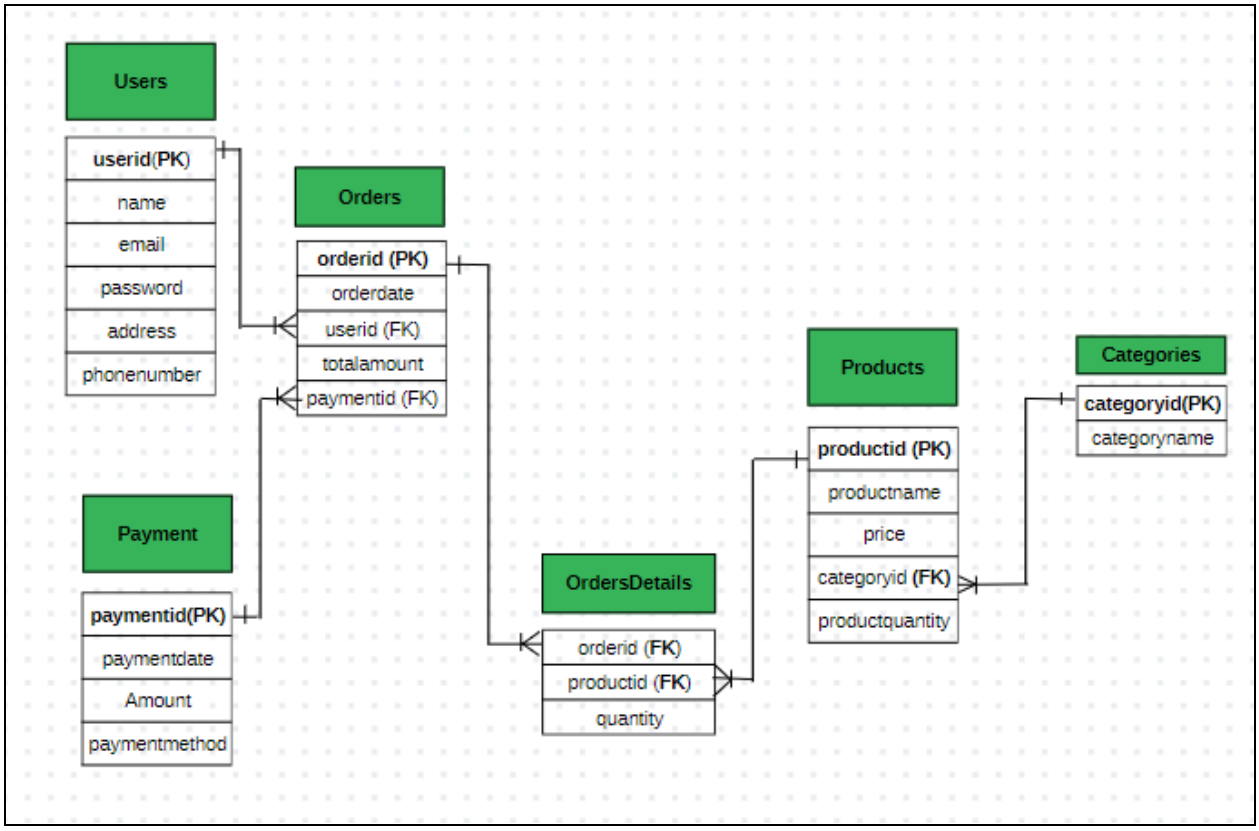
Established relationships between entities and analyzing how these entities interact within the system. For instance, we established that Users place Orders, Products belong to Categories, and Orders are linked to Payments.

CONCEPTUAL DESIGN
(ENTITY- RELATIONSHIP DIAGRAM)

- **Representation of Entities as Rectangles**: Each identified entity was visually represented as a rectangle in the ERD. This provided a clear and organized depiction of the core components of the database, such as Users, Products, Orders, Payments, Categories, and Order Details.

- **Incorporation of Attributes within Entities**: For each entity, the relevant attributes were listed within the corresponding rectangle. Primary keys were distinctly underlined to signify their role in uniquely identifying records within the entity. This step ensured that all critical data points were systematically organized within their respective entities.
- **Establishment of Relationships Between Entities**: To illustrate the connections between related entities, relationship lines were drawn. These lines depicted the logical associations between entities, such as the connection between Users and Orders, ensuring that the relationships were clearly understood and accurately represented.
- **Specification of Cardinality and Participation Constraints**: Cardinality and participation constraints were explicitly noted on the relationship lines to define the nature of the relationships between entities. Notations were used to indicate the relationship types, such as one-to-many or many-to-one, and to specify whether the participation of an entity in a relationship was mandatory or optional. This step was crucial in ensuring that the database design accurately reflected the business rules and operational requirements of Flipkart.

ENTITY RELATION DIAGRAM



NORMALIZATION PROCESS

First Normal Form (1NF) Implementation

The initial step in the normalization process involves structuring each table to have atomic, indivisible values in its columns, thus eliminating any repeating groups or arrays within the table. This ensures that all attributes, such as

ProductName, Description, and Price, in the Products table were atomic, guaranteeing each column contained only a single piece of information per record.

Second Normal Form (2NF) Implementation

Following 1NF, the database design was refined to meet the requirements of the Second Normal Form (2NF) focused on removing partial dependencies by ensuring that all non-key attributes were fully dependent on the entire primary key. In the case of the attributes like TotalAmount in the Orders table were dependent solely on OrderID and introduced the OrderDetails table to avoid partial dependencies and organize data without redundancy.

Third Normal Form (3NF) Implementation

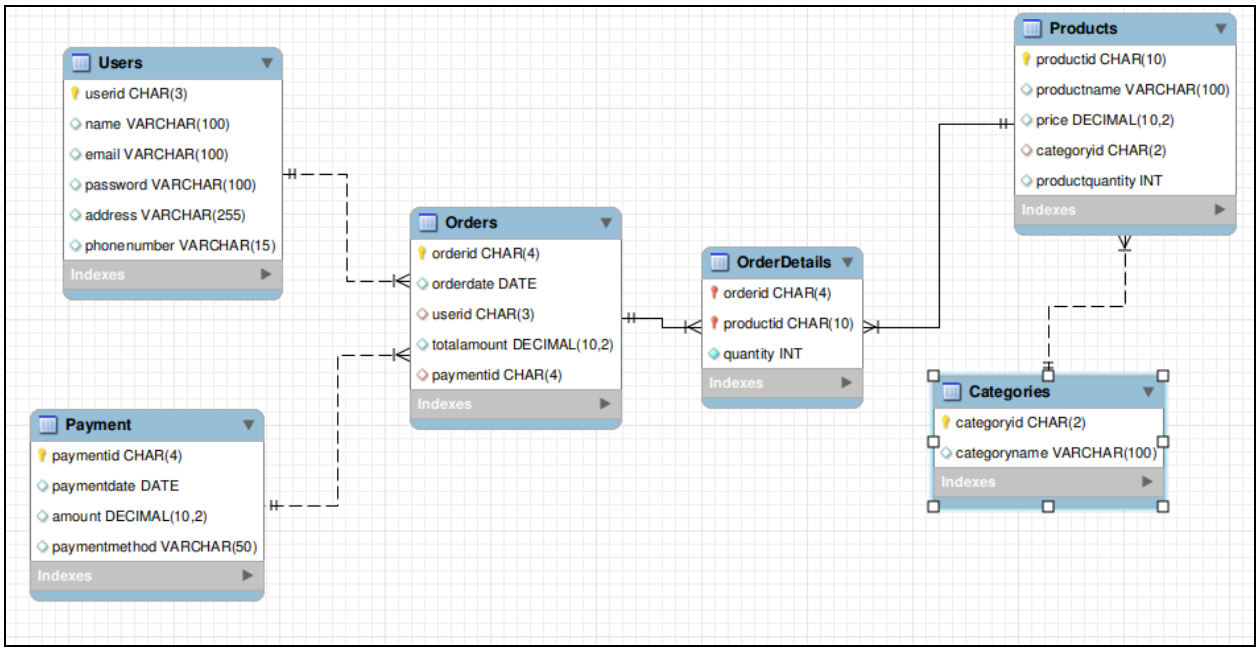
The final phase of normalization involved bringing the database into compliance with the Third Normal Form (3NF). This step aimed to eliminate transitive dependencies, ensuring that non-key attributes depended only on the primary key and not on any other non-key attributes. In this step attributes like PhoneNumber in the Users table depended directly on UserID, ensuring the data structure was free from unnecessary dependencies, thereby enhancing database integrity and performance.

SQL IMPLEMENTATION

- **Translation of ERD into SQL Code**: The visual representation of the ERD was meticulously converted into SQL statements, ensuring an accurate translation of the conceptual design into a tangible database structure. This step laid the groundwork for the database by defining the entities and their relationships in SQL.

- **Creation of Tables for Each Entity**: SQL CREATE TABLE statements were crafted for each identified entity, including Users, Products, Categories, Orders, Payments, and Order Detail. These statements were designed to establish the necessary tables with their respective attributes, as outlined in the ERD.
- **Definition of Primary and Foreign Keys**: Primary keys were specified for each table to uniquely identify records within the database. Additionally, foreign keys were defined where relationships existed between tables, linking related data and establishing the foundational structure of the database.
- **Establishment of Relationships Through Foreign Key Constraints**: Foreign key constraints were implemented to maintain referential integrity between related tables. This step was crucial in ensuring that relationships, such as those between Users and Orders or Products and Categories, were properly enforced within the database.
- **Verification of Relationships in MySQL Workbench**: Each relationship was confirmed using MySQL Workbench to ensure that the foreign key constraints were correctly established and functioning as intended. This verification process was essential in validating the integrity and accuracy of the database design.

ENTITY RELATIONSHIP DIAGRAM ON MYSQL WORKBENCH



DATA ACCESS LANGUAGE

Data Access Language (DAL) is a set of commands or languages used to interact with databases, enabling users to retrieve, insert, update, and delete data. It serves as the interface for managing and accessing the data stored within a database system, ensuring that information can be efficiently manipulated and retrieved as needed.

1. **User Management**

- **Entities Accessed**: Users
- **Data Accessibility:** Full access to user data including name, email, password (hashed/encrypted), address, and phone number.
- Managing user profiles, authentication, and customer support.
- **User:** *user_mgmt*
- **Password:** *user_mgmt_password*
- **Code:**

CREATE USER 'user_mgmt'@'localhost' IDENTIFIED BY 'user_mgmt_password';

GRANT SELECT, INSERT, UPDATE, DELETE ON flipkart.Users TO 'user_mgmt'@'localhost';

FLUSH PRIVILEGES;

```
mysql> select * from Users;
+--------+--------------+--------------------------+------------+----------------------------+---------
----+
| userid | name         | email                    | password   | address                    | phonenum
ber |
+--------+--------------+--------------------------+------------+----------------------------+---------
----+
| 001    | John Doe     | john.doe@example.com     | password123 | 123 Elm Street, Springfield | 555-1234
    |
| 002    | Jane Smith   | jane.smith@example.com   | mypassword  | 456 Oak Avenue, Metropolis  | 555-5678
    |
| 003    | Alice Johnson | alice.johnson@example.com | alicepass   | 789 Pine Road, Gotham       | 555-8765
    |
+--------+--------------+--------------------------+------------+----------------------------+---------
----+
```

```
mysql> INSERT INTO Users (userid, name, email, password, address, phonenumber)
    -> VALUES ('004', 'Bob Brown', 'bob.brown@example.com', 'securepass', '321 Birch Lane, Star City', '555
-4321');
Query OK, 1 row affected (0.08 sec)

mysql> select * from Users;
+--------+--------------+--------------------------+------------+----------------------------+---------
----+
| userid | name         | email                    | password   | address                    | phonenum
ber |
+--------+--------------+--------------------------+------------+----------------------------+---------
----+
| 001    | John Doe     | john.doe@example.com     | password123 | 123 Elm Street, Springfield | 555-1234
    |
| 002    | Jane Smith   | jane.smith@example.com   | mypassword  | 456 Oak Avenue, Metropolis  | 555-5678
    |
| 003    | Alice Johnson | alice.johnson@example.com | alicepass   | 789 Pine Road, Gotham       | 555-8765
    |
| 004    | Bob Brown    | bob.brown@example.com    | securepass  | 321 Birch Lane, Star City   | 555-4321
    |
+--------+--------------+--------------------------+------------+----------------------------+---------
----+
```

```
mysql> -- Update user information
mysql> UPDATE Users
    -> SET name = 'Robert Brown', email = 'robert.brown@example.com', phonenumber = '555-1235'
    -> WHERE userid = '004';
Query OK, 1 row affected (0.08 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from Users;
+--------+--------------+--------------------------+------------+----------------------------+---------
----+
| userid | name         | email                    | password   | address                    | phonenum
ber |
+--------+--------------+--------------------------+------------+----------------------------+---------
----+
| 001    | John Doe     | john.doe@example.com     | password123 | 123 Elm Street, Springfield | 555-1234
    |
| 002    | Jane Smith   | jane.smith@example.com   | mypassword  | 456 Oak Avenue, Metropolis  | 555-5678
    |
| 003    | Alice Johnson | alice.johnson@example.com | alicepass   | 789 Pine Road, Gotham       | 555-8765
    |
| 004    | Robert Brown  | robert.brown@example.com  | securepass  | 321 Birch Lane, Star City   | 555-1235
    |
+--------+--------------+--------------------------+------------+----------------------------+---------
----+
```

```
mysql> -- Delete a user record
mysql> DELETE FROM Users
    -> WHERE userid = '004';
Query OK, 1 row affected (0.04 sec)

mysql> select * from Users;
+--------+--------------+--------------------------+------------+----------------------------+---------
----+
| userid | name         | email                    | password   | address                    | phonenum
ber |
+--------+--------------+--------------------------+------------+----------------------------+---------
----+
| 001    | John Doe     | john.doe@example.com     | password123 | 123 Elm Street, Springfield | 555-1234
    |
| 002    | Jane Smith   | jane.smith@example.com   | mypassword  | 456 Oak Avenue, Metropolis  | 555-5678
    |
| 003    | Alice Johnson | alice.johnson@example.com | alicepass   | 789 Pine Road, Gotham       | 555-8765
    |
+--------+--------------+--------------------------+------------+----------------------------+---------
----+
```

2. **Product Management User**

- Entities Accessed: Categories and Products
- Data Accessibility: Full access to categories and product data including product names, prices, quantities, and associated categories.
- Managing product listings, categorization and inventory management.
- **User:** *product_mgmt*

- **Password:** *product_mgmt_password*
- **Code:**

CREATE USER 'product_mgmt'@'localhost' IDENTIFIED BY 'product_mgmt_password';
GRANT SELECT, INSERT, UPDATE, DELETE ON flipkart.Categories TO 'product_mgmt'@'localhost';

GRANT SELECT, INSERT, UPDATE, DELETE ON flipkart.Products TO 'product_mgmt'@'localhost';

FLUSH PRIVILEGES;

```
mysql> -- Retrieve all category details
mysql> SELECT *
    -> FROM Categories;
+------------+----------------+
| categoryid | categoryname   |
+------------+----------------+
| C1         | Electronics    |
| C2         | Clothing       |
| C3         | Home & Kitchen |
+------------+----------------+
```

```
mysql> -- Fetch product details with their corresponding categories
mysql> SELECT p.productid, p.productname, p.price, p.productquantity, c.categoryname
    -> FROM Products p
    -> JOIN Categories c ON p.categoryid = c.categoryid;
+-----------+-------------+--------+-----------------+----------------+
| productid | productname | price  | productquantity | categoryname   |
+-----------+-------------+--------+-----------------+----------------+
| PROD1     | Smartphone  | 299.99 |              50 | Electronics    |
| PROD2     | Laptop      | 999.99 |              20 | Electronics    |
| PROD3     | T-Shirt     |  19.99 |             100 | Clothing       |
| PROD4     | Blender     |  49.99 |              30 | Home & Kitchen |
+-----------+-------------+--------+-----------------+----------------+
```

```
mysql> -- Insert a new category
mysql> INSERT INTO Categories (categoryid, categoryname)
    -> VALUES ('C4', 'Sports & Outdoors');
Query OK, 1 row affected (0.02 sec)

mysql> select * from Categories;
+------------+--------------------+
| categoryid | categoryname       |
+------------+--------------------+
| C1         | Electronics        |
| C2         | Clothing           |
| C3         | Home & Kitchen     |
| C4         | Sports & Outdoors  |
+------------+--------------------+
```

```
mysql> -- Add a new product under a specific category
mysql> INSERT INTO Products (productid, productname, price, categoryid, productquantity)
    -> VALUES ('PROD5', 'Running Shoes', 79.99, 'C4', 150);
Query OK, 1 row affected (0.02 sec)

mysql> select * from Products;
+-----------+---------------+--------+------------+-----------------+
| productid | productname   | price  | categoryid | productquantity |
+-----------+---------------+--------+------------+-----------------+
| PROD1     | Smartphone    | 299.99 | C1         |              50 |
| PROD2     | Laptop        | 999.99 | C1         |              20 |
| PROD3     | T-Shirt       |  19.99 | C2         |             100 |
| PROD4     | Blender       |  49.99 | C3         |              30 |
| PROD5     | Running Shoes |  79.99 | C4         |             150 |
+-----------+---------------+--------+------------+-----------------+
```

```
mysql> -- Update product price and quantity
mysql> UPDATE Products
    -> SET price = 89.99, productquantity = 140
    -> WHERE productid = 'PROD5';
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from Products;
+-----------+---------------+--------+------------+-----------------+
| productid | productname   | price  | categoryid | productquantity |
+-----------+---------------+--------+------------+-----------------+
| PROD1     | Smartphone    | 299.99 | C1         |              50 |
| PROD2     | Laptop        | 999.99 | C1         |              20 |
| PROD3     | T-Shirt       |  19.99 | C2         |             100 |
| PROD4     | Blender       |  49.99 | C3         |              30 |
| PROD5     | Running Shoes |  89.99 | C4         |             140 |
+-----------+---------------+--------+------------+-----------------+
```

```
mysql> -- Remove a product from the inventory
mysql> DELETE FROM Products
    -> WHERE productid = 'PROD5';
Query OK, 1 row affected (0.05 sec)

mysql> select * from Products;
+-----------+-------------+--------+------------+-----------------+
| productid | productname | price  | categoryid | productquantity |
+-----------+-------------+--------+------------+-----------------+
| PROD1     | Smartphone  | 299.99 | C1         |              50 |
| PROD2     | Laptop      | 999.99 | C1         |              20 |
| PROD3     | T-Shirt     |  19.99 | C2         |             100 |
| PROD4     | Blender     |  49.99 | C3         |              30 |
+-----------+-------------+--------+------------+-----------------+
```

```
mysql> -- Change the name of a category
mysql> UPDATE Categories
    -> SET categoryname = 'Outdoor & Sports Equipment'
    -> WHERE categoryid = 'C4';
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from Categories;
+------------+-----------------------------+
| categoryid | categoryname                |
+------------+-----------------------------+
| C1         | Electronics                 |
| C2         | Clothing                    |
| C3         | Home & Kitchen              |
| C4         | Outdoor & Sports Equipment  |
+------------+-----------------------------+
```

```
mysql> -- Remove a category (and ensure products under this category are handled separately)
mysql> DELETE FROM Categories
    -> WHERE categoryid = 'C4';
Query OK, 1 row affected (0.03 sec)

mysql> select * from Categories;
+------------+---------------+
| categoryid | categoryname  |
+------------+---------------+
| C1         | Electronics   |
| C2         | Clothing      |
| C3         | Home & Kitchen |
+------------+---------------+
```

3. **Sales and Order Management User**

- Entities Accessed: Orders and OrderDetails
- Data Accessibility: Full access to order details including order ID, order date, user ID, total amount, and associated products in each order.
- Limited access to user data: Only user IDs to track orders.
- **User:** *sales_mgmt*
- **Password:** *sales_mgmt_password*

- **Code:**

CREATE USER 'sales_mgmt'@'localhost' IDENTIFIED BY 'sales_mgmt_password';

GRANT SELECT, INSERT, UPDATE, DELETE ON flipkart.Orders TO 'sales_mgmt'@'localhost';

GRANT SELECT, INSERT, UPDATE, DELETE ON flipkart.OrderDetails TO 'sales_mgmt'@'localhost';

GRANT SELECT ON flipkart.Users(userid) TO 'sales_mgmt'@'localhost';

FLUSH PRIVILEGES;

```
mysql> -- Retrieve all orders with user IDs and total amounts
mysql> SELECT orderid, orderdate, userid, totalamount
    -> FROM Orders;
+---------+------------+--------+-------------+
| orderid | orderdate  | userid | totalamount |
+---------+------------+--------+-------------+
| ORD1    | 2024-08-01 | 001    |      299.99 |
| ORD2    | 2024-08-02 | 002    |      999.99 |
| ORD3    | 2024-08-03 | 003    |       19.99 |
+---------+------------+--------+-------------+
```

```
mysql> -- Fetch details of a specific order including product information
mysql> SELECT o.orderid, o.orderdate, o.userid, o.totalamount, od.productid, p.productname, od.quantity
    -> FROM Orders o
    -> JOIN OrderDetails od ON o.orderid = od.orderid
    -> JOIN Products p ON od.productid = p.productid
    -> WHERE o.orderid = 'ORD1';
+---------+------------+--------+-------------+-----------+-------------+----------+
| orderid | orderdate  | userid | totalamount | productid | productname | quantity |
+---------+------------+--------+-------------+-----------+-------------+----------+
| ORD1    | 2024-08-01 | 001    |      299.99 | PROD4     | Blender     |       11 |
+---------+------------+--------+-------------+-----------+-------------+----------+
```

```
mysql> INSERT INTO Payment (paymentid, paymentdate, amount, paymentmethod)
    -> VALUES ('PAY4', '2024-08-04', 129.99, 'Credit Card');
Query OK, 1 row affected (0.02 sec)

mysql> INSERT INTO Orders (orderid, orderdate, userid, totalamount, paymentid)
    -> VALUES ('ORD4', '2024-08-04', '003', 129.99, 'PAY4');
Query OK, 1 row affected (0.03 sec)
```

```
mysql> select * from Payment;
+-----------+-------------+--------+---------------+
| paymentid | paymentdate | amount | paymentmethod |
+-----------+-------------+--------+---------------+
| PAY1      | 2024-08-01  | 299.99 | Credit Card   |
| PAY2      | 2024-08-02  | 999.99 | PayPal        |
| PAY3      | 2024-08-03  |  19.99 | Debit Card    |
| PAY4      | 2024-08-04  | 129.99 | Credit Card   |
+-----------+-------------+--------+---------------+
4 rows in set (0.00 sec)

mysql> select * from Orders;
+---------+-------------+--------+-------------+-----------+
| orderid | orderdate   | userid | totalamount | paymentid |
+---------+-------------+--------+-------------+-----------+
| ORD1    | 2024-08-01 | 001    |      299.99 | PAY1      |
| ORD2    | 2024-08-02 | 002    |      999.99 | PAY2      |
| ORD3    | 2024-08-03 | 003    |       19.99 | PAY3      |
| ORD4    | 2024-08-04 | 003    |      129.99 | PAY4      |
+---------+-------------+--------+-------------+-----------+
```

```
mysql> INSERT INTO OrderDetails (orderid, productid, quantity) VALUES ('ORD4', 'PROD2', 2);
Query OK, 1 row affected (0.02 sec)

mysql> select * from OrderDetails;
+---------+-----------+----------+
| orderid | productid | quantity |
+---------+-----------+----------+
| ORD1    | PROD4     |       11 |
| ORD2    | PROD2     |        5 |
| ORD2    | PROD3     |        1 |
| ORD3    | PROD1     |        7 |
| ORD4    | PROD2     |        2 |
+---------+-----------+----------+
```

```
mysql> -- Update the total amount of an existing order
mysql> UPDATE Orders
    -> SET totalamount = 149.99
    -> WHERE orderid = 'ORD4';
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from Orders;
+---------+------------+--------+-------------+-----------+
| orderid | orderdate  | userid | totalamount | paymentid |
+---------+------------+--------+-------------+-----------+
| ORD1    | 2024-08-01 | 001    |      299.99 | PAY1      |
| ORD2    | 2024-08-02 | 002    |      999.99 | PAY2      |
| ORD3    | 2024-08-03 | 003    |       19.99 | PAY3      |
| ORD4    | 2024-08-04 | 003    |      149.99 | PAY4      |
+---------+------------+--------+-------------+-----------+
```

```
mysql> -- Remove an order and its associated details
mysql> DELETE FROM OrderDetails
    -> WHERE orderid = 'ORD4';
Query OK, 1 row affected (0.03 sec)

mysql> select * from OrderDetails;
+---------+-----------+----------+
| orderid | productid | quantity |
+---------+-----------+----------+
| ORD1    | PROD4     |       11 |
| ORD2    | PROD2     |        5 |
| ORD2    | PROD3     |        1 |
| ORD3    | PROD1     |        7 |
+---------+-----------+----------+
```

```
mysql> DELETE FROM Orders
    -> WHERE orderid = 'ORD4';
Query OK, 1 row affected (0.03 sec)

mysql> select * from Orders;
+---------+------------+--------+-------------+-----------+
| orderid | orderdate  | userid | totalamount | paymentid |
+---------+------------+--------+-------------+-----------+
| ORD1    | 2024-08-01 | 001    |      299.99 | PAY1      |
| ORD2    | 2024-08-02 | 002    |      999.99 | PAY2      |
| ORD3    | 2024-08-03 | 003    |       19.99 | PAY3      |
+---------+------------+--------+-------------+-----------+
```

```
mysql> -- Retrieve order details and associated products for all orders
mysql> SELECT o.orderid, o.orderdate, o.userid, od.productid, p.productname, od.quantity
    -> FROM Orders o
    -> JOIN OrderDetails od ON o.orderid = od.orderid
    -> JOIN Products p ON od.productid = p.productid;
+---------+------------+--------+-----------+-------------+----------+
| orderid | orderdate  | userid | productid | productname | quantity |
+---------+------------+--------+-----------+-------------+----------+
| ORD1    | 2024-08-01 | 001    | PROD4     | Blender     |       11 |
| ORD2    | 2024-08-02 | 002    | PROD2     | Laptop      |        5 |
| ORD2    | 2024-08-02 | 002    | PROD3     | T-Shirt     |        1 |
| ORD3    | 2024-08-03 | 003    | PROD1     | Smartphone  |        7 |
+---------+------------+--------+-----------+-------------+----------+
```

4. **Logistics and Inventory User**

- Entities Accessed: Products and OrderDetails
- Data Accessibility: Access to product data (quantities and IDs) to manage inventory and coordinate shipments. Access to order details to ensure correct items are shipped and delivered.
- Managing inventory levels, ensuring timely shipments, and handling returns.
- **User:** *logistics_mgmt*
- **Password:** *logistics_mgmt_password*
- **Code:**

CREATE USER 'logistics_mgmt'@'localhost' IDENTIFIED BY 'logistics_mgmt_password';

GRANT SELECT, UPDATE ON flipkart.Products TO 'logistics_mgmt'@'localhost';

GRANT SELECT ON flipkart.OrderDetails TO 'logistics_mgmt'@'localhost';

FLUSH PRIVILEGES;

```
mysql> select * from Products;
+-----------+-------------+--------+------------+-----------------+
| productid | productname | price  | categoryid | productquantity |
+-----------+-------------+--------+------------+-----------------+
| PROD1     | Smartphone  | 299.99 | C1         |              50 |
| PROD2     | Laptop      | 999.99 | C1         |              20 |
| PROD3     | T-Shirt     |  19.99 | C2         |             100 |
| PROD4     | Blender     |  49.99 | C3         |              30 |
+-----------+-------------+--------+------------+-----------------+
```

```
mysql> -- Fetch details of products in a specific order to manage shipment
mysql> SELECT od.orderid, od.productid, p.productname, od.quantity
    -> FROM OrderDetails od
    -> JOIN Products p ON od.productid = p.productid
    -> WHERE od.orderid = 'ORD2';
+---------+-----------+-------------+----------+
| orderid | productid | productname | quantity |
+---------+-----------+-------------+----------+
| ORD2    | PROD2     | Laptop      |        5 |
| ORD2    | PROD3     | T-Shirt     |        1 |
+---------+-----------+-------------+----------+
```

```
mysql> -- Update product quantity after shipment
mysql> UPDATE Products
    -> SET productquantity = productquantity - 5
    -> WHERE productid = 'PROD1';
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from Products;
+-----------+-------------+--------+------------+-----------------+
| productid | productname | price  | categoryid | productquantity |
+-----------+-------------+--------+------------+-----------------+
| PROD1     | Smartphone  | 299.99 | C1         |              45 |
| PROD2     | Laptop      | 999.99 | C1         |              20 |
| PROD3     | T-Shirt     |  19.99 | C2         |             100 |
| PROD4     | Blender     |  49.99 | C3         |              30 |
+-----------+-------------+--------+------------+-----------------+
```

```
mysql> -- Retrieve total quantity of each product in all orders for inventory management
mysql> SELECT p.productid, p.productname, SUM(od.quantity) AS total_ordered_quantity
    -> FROM OrderDetails od
    -> JOIN Products p ON od.productid = p.productid
    -> GROUP BY p.productid, p.productname;
+-----------+-------------+-----------------------+
| productid | productname | total_ordered_quantity |
+-----------+-------------+-----------------------+
| PROD4     | Blender     |                    11 |
| PROD2     | Laptop      |                     5 |
| PROD3     | T-Shirt     |                     1 |
| PROD1     | Smartphone  |                     7 |
+-----------+-------------+-----------------------+
```

5. **Analytics and Reporting User**

- Entities Accessed: All Entities (Aggregated Data)
- Data Accessibility: Aggregated and anonymized data from all tables to generate insights, trends, and performance metrics.
- Analyzing sales, user behavior, product performance, and generating reports for decision-making.
- **User:** *analyst*
- **Password:** *analyst_password*
- **Code:**

CREATE USER 'analyst'@'localhost' IDENTIFIED BY 'analyst_password';

GRANT SELECT ON flipkart. TO 'analyst'@'localhost';*

FLUSH PRIVILEGES;

```
mysql> -- Retrieve total sales amount by month
mysql> SELECT DATE_FORMAT(orderdate, '%Y-%m') AS month, SUM(totalamount) AS total_sales
    -> FROM Orders
    -> GROUP BY month
    -> ORDER BY month;
+---------+-------------+
| month   | total_sales |
+---------+-------------+
| 2024-08 |     1319.97 |
+---------+-------------+
```

```
mysql> -- Analyze average product price and total quantity sold by category
mysql> SELECT c.categoryname, AVG(p.price) AS average_price, SUM(od.quantity) AS total_quantity_sold
    -> FROM Products p
    -> JOIN Categories c ON p.categoryid = c.categoryid
    -> JOIN OrderDetails od ON p.productid = od.productid
    -> GROUP BY c.categoryname;
+---------------+---------------+---------------------+
| categoryname  | average_price | total_quantity_sold |
+---------------+---------------+---------------------+
| Electronics   |    649.990000 |                  12 |
| Clothing      |     19.990000 |                   1 |
| Home & Kitchen |     49.990000 |                  11 |
+---------------+---------------+---------------------+
```

```
mysql> -- Get the total number of orders and average order value per user
mysql> SELECT u.userid, COUNT(o.orderid) AS total_orders, AVG(o.totalamount) AS average_order_value
    -> FROM Users u
    -> JOIN Orders o ON u.userid = o.userid
    -> GROUP BY u.userid;
+--------+--------------+---------------------+
| userid | total_orders | average_order_value |
+--------+--------------+---------------------+
| 001    |            1 |          299.990000 |
| 002    |            1 |          999.990000 |
| 003    |            1 |           19.990000 |
+--------+--------------+---------------------+
```

```
mysql> -- Fetch the top 5 products with the highest total sales
mysql> SELECT p.productname, SUM(od.quantity * p.price) AS total_sales
    -> FROM Products p
    -> JOIN OrderDetails od ON p.productid = od.productid
    -> GROUP BY p.productname
    -> ORDER BY total_sales DESC
    -> LIMIT 5;
+-------------+-------------+
| productname | total_sales |
+-------------+-------------+
| Laptop      |     4999.95 |
| Smartphone  |     2099.93 |
| Blender     |      549.89 |
| T-Shirt     |       19.99 |
+-------------+-------------+
```

```
mysql> -- Generate a report of payment methods used and their total amounts
mysql> SELECT paymentmethod, SUM(amount) AS total_amount
    -> FROM Payment
    -> GROUP BY paymentmethod;
+---------------+--------------+
| paymentmethod | total_amount |
+---------------+--------------+
| Credit Card   |       429.98 |
| PayPal        |       999.99 |
| Debit Card    |        19.99 |
| Bank Transfer |       159.99 |
+---------------+--------------+
```

INSIGHTS

1.1 Core Entities

The ERD reveals six core entities: Users, Categories, Products, Payment, Orders, and Order Detail. Each entity plays a crucial role in the e-commerce ecosystem:

- Users: Represents customer information
- Categories: Allows for logical grouping of products
- Products: Contains details of items available for purchase
- Payment: Tracks transaction details
- Orders: Records customer purchases
- Order Details: Junction Table for Product and Order

1.2 Relationships

- Users can place multiple Orders (one-to-many): This allows a single user to have a history of multiple orders over time.
- Products belong to Categories (many-to-one): Enables efficient categorization and browsing of products.
- Orders are associated with one Payment (one-to-one): Ensures each order has a corresponding payment record.
- Products are tracked in Categories (one-to-many): Facilitates real-time updates for highest selling categories in turn use for stock management.

1.3 Data Integrity

Foreign key constraints ensure referential integrity between related tables. For example, an order cannot exist without a corresponding user, maintaining data consistency.

1.4 Scalability

The design allows for easy addition of new products, categories, and users. This is crucial for an e-commerce platform that continuously expands its user base and product offerings.

1.5 Payment Tracking

The separate Payment table enables detailed transaction recording and supports various payment methods. This design choice allows for easy integration of new payment options in the future.

1.6 Inventory Management

The Category table and quantity attribute facilitates stock tracking and can be used for reordering processes. This is essential for maintaining accurate product availability and managing supply chain operations.

CONCLUSION

The Entity-Relationship Diagram created for Flipkart's e-commerce platform provides a solid foundation for a robust and scalable database system. By capturing the essential entities (Users, Products, Categories, Orders, Payments, and Order Detail) and their interrelationships, the ERD effectively models the core functionalities of an e-commerce ecosystem.

The design demonstrates several strengths:

1. Flexibility: It can accommodate growth in users, products, and transactions.

2. Data Integrity: Relationships and constraints ensure consistent and reliable data.

3. Operational Support: The structure facilitates key e-commerce operations from user management to inventory control.

The SQL implementation of this ERD provides a practical starting point for database creation. However, as Flipkart continues to evolve, this model may need to be expanded. Future enhancements could include:

- Integration of customer reviews and ratings
- Support for wish lists and saved items
- Incorporation of seller information for a marketplace model
- Advanced inventory management features

In conclusion, this ERD serves as a valuable tool for understanding and implementing Flipkart's data architecture. It provides a clear visualization of the system's structure, guiding database design and application development. As Flipkart grows and adapts to changing market demands, this foundational design can be iteratively refined to support new features and optimizations, ensuring the platform's continued success in the competitive e-commerce landscape.