**ONLINE MOBILE SHOPPING**

**DATABASE MANAGEMENT SYSTEMS**

**INTRODUCTION:**

Online Mobile Shopping Database gives a well-structured framework for managing an online mobile retail shop. The importance of database in e-commerce has made it crucial for handling tasks like user registrations, mobile product listings, order placements, payments management, and inventory control. The project intends to provide a highly safe and scalable database with data integrity, high retrieval, and very smooth transaction processing.

The data base is critical in online shopping because it stores and manages large amounts of customer data as well as product data. The system works as much as multiple users simultaneously while providing an assured security performance. This project also lays a good foundation for systematic service of customer interaction, tracking order, and payment by employing principles of relational database.

The project strongly deals with e-business operation by minimizing redundancy and optimizing query performance. This report details the conceptualization, design, and development of the database, emphasizing its importance in effectively managing an online mobile shopping platform.

**BACKGROUND:**

The e-commerce sector has been growing at a great speed and is changing people's buying behaviour for all commodities including mobile phones. Traditional retail outlets now turn part of their business to the online world in an effort to reach more customers as well as offer them convenience. However, such activities can only be carried out effectively with the aid of a well-structured database that can manage large volumes of data efficiently and securely.

No proper database means losses due to issues like duplication of records, latency in the retrieval of the data, and poor tracing of transactions. Manual record-keeping is prone to errors and does not scale well with the increase in customer demands. An efficiently designed relational database system solves the problem by creating table definitions and maintaining any relationships among various entities in the database.

The project aims to set up a database for managing mobile phone listings, customer details, processing orders, and managing payment transactions efficiently. The system will deliver a properly maintained record with reduced redundancy while increasing the overall efficiency of operations. With normalization techniques applied for optimizing the queries, the database guarantees consistency within the data and quick access to relevant information, making it a vitally important component of an online mobile shopping platform.

**ENTITY-RELATIONSHIP DIAGRAM**:

The ER model gives a conceptual framework as far as the structure of the database is concerned; it shows the relationship between the various entities. The important entities in this database are as described below:

**Users**: It refers to the customers who register on the platform. The attributes are UserID, Name, Email, PhoneNumber, and Address.

**Mobiles**: It refers to the products that store the details of the mobile phones such as MobileID, Brand, Model, Price, Stock, and Specifications.

**Orders**: Used to track the customer orders with attributes: OrderID, UserID, OrderDate, TotalAmount, and Status.

**Payments**: For transactions that deal with payment, the entity will maintain fields such as PaymentID, OrderID, Payment Method, Payment Status, and TransactionID.

**OrderDetails**: It consists of details about mobile phones being ordered, as well as its attributes OrderDetailID, OrderID, MobileID, Quantity, and Subtotal.

The ER diagram provides a visual specification for the interaction of these entities to ensure that the database design is systematic and efficient.

**DESCRIPTION OF ENTITY RELATIONSHIP DIAGRAM:**

The relationship in between Users and their Orders represents a one to many, as one user might give rise to many orders. Every order is linked to a payment that settles the amount for that order through OrderID. The product details are found in the Mobiles entity and directly connected to OrderDetails, which contains information on the quantity and subtotal for every mobile phone it has bought. It serves as a bridge in-between Orders and Mobiles and ensures that each purchase has a well-detailed breakdown.

This design allows a smooth transaction monitor, maintains data integrity, and optimizes query performance by way of reducing redundancy and improving data retrieval efficiency.



**CONVERSION OF ENTITY RELATIONSHIP DIAGRAM INTO TABLES:**

The ER diagram is converted to relational tables with primary and foreign key constraints for relationship preservation and data consistency. The tables are as follows:

**Users:** Holds customer details with unique constraints on Email and PhoneNumber.

**Mobiles:** Holds specifications, prices, and stock levels of mobile phones.

**Orders:** Manages user purchases, including total cost, and order status.

**Payments:** Holds transaction details associated with each order.

**OrderDetails:** Maintains mobile phone quantities and subtotals for each order.

Together, these tables enable the retrieval and management of data in an efficient way.

**CREATION OF TABLES:**

All The tables are created using SQL commands, defining their structures with appropriate data types, constraints, and relationships. The **Users** table has an auto-incrementing UserID as the primary key for unique identification. The **Mobiles** table includes attributes like Brand, Model, Price, and Stock for managing product details.

The **Orders** table records purchases, linking them to users via a foreign key. The **Payments** table separately handles transactions, ensuring clarity in payment tracking. The **OrderDetails** table links orders and mobiles, detailing purchase quantities and subtotals.

A screenshot of a computer

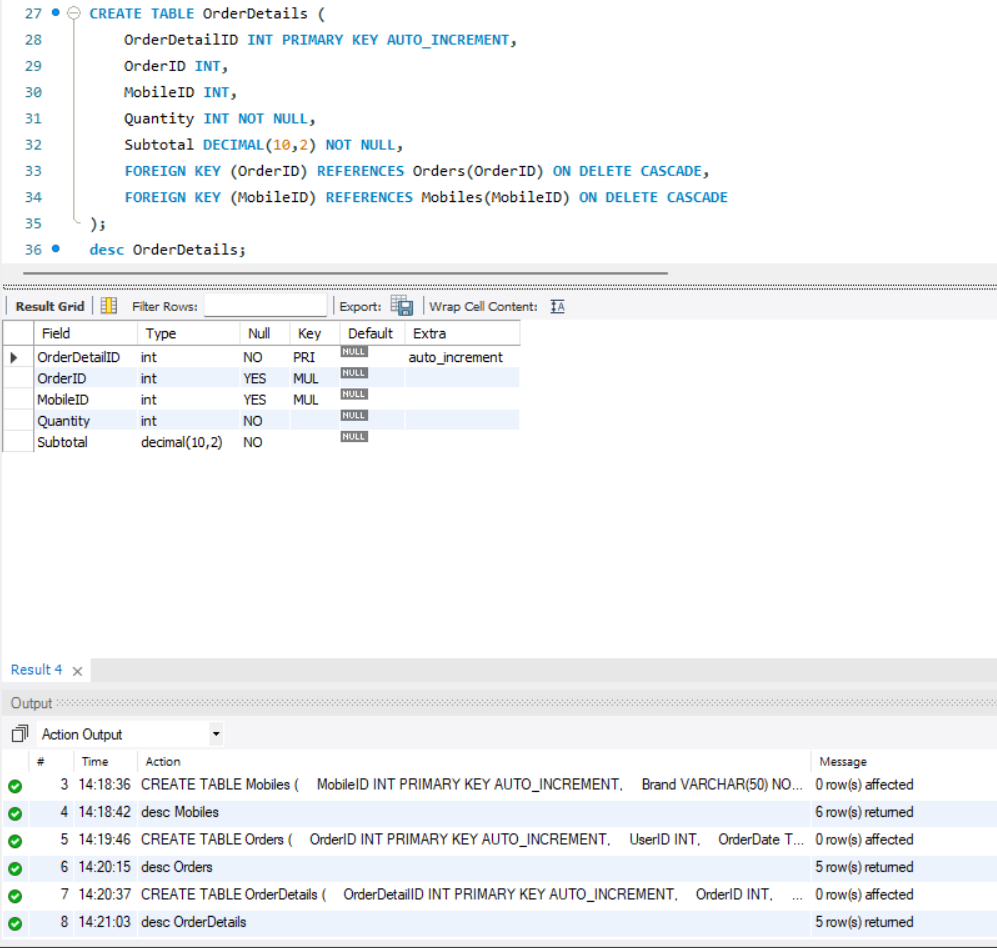
AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.



A screenshot of a computer

AI-generated content may be incorrect.

**TABLE CREATION**

**USER TABLE:**

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Key Type** |
| UserID | INT | Primary Key (PK), Auto Increment |
| Name | VARCHAR(100) | Not Null |
| Email | VARCHAR(100) | Unique, Not Null |
| PhoneNumber | VARCHAR(15) | Unique, Not Null |
| Address | TEXT | — |

**MOBILES:**

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Key Type** |
| MobileID | INT | Primary Key (PK), Auto Increment |
| Brand | VARCHAR(50) | Not Null |
| Model | VARCHAR(100) | Not Null |
| Price | DECIMAL(10,2) | Not Null |
| Stock | INT | Not Null |
| Specifications | TEXT | — |

**ORDERS:**

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Key Type** |
| OrderID | INT | Primary Key (PK), Auto Increment |
| UserID | INT | Foreign Key (FK) → Users(UserID) |
| OrderDate | TIMESTAMP | Default: CURRENT\_TIMESTAMP |
| TotalAmount | DECIMAL(10,2) | Not Null |
| Status | ENUM('Pending', 'Shipped', 'Delivered', 'Cancelled') | Default: 'Pending' |

**PAYMENTS:**

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Key Type** |
| PaymentID | INT | Primary Key (PK), Auto Increment |
| OrderID | INT | Foreign Key (FK) → Orders(OrderID) |
| PaymentMethod | ENUM('Credit Card', 'Debit Card', 'UPI', 'Net Banking', 'Cash on Delivery') | Not Null |
| PaymentStatus | ENUM('Success', 'Failed', 'Pending') | Default: 'Pending' |
| TransactionID | VARCHAR(50) | Unique |

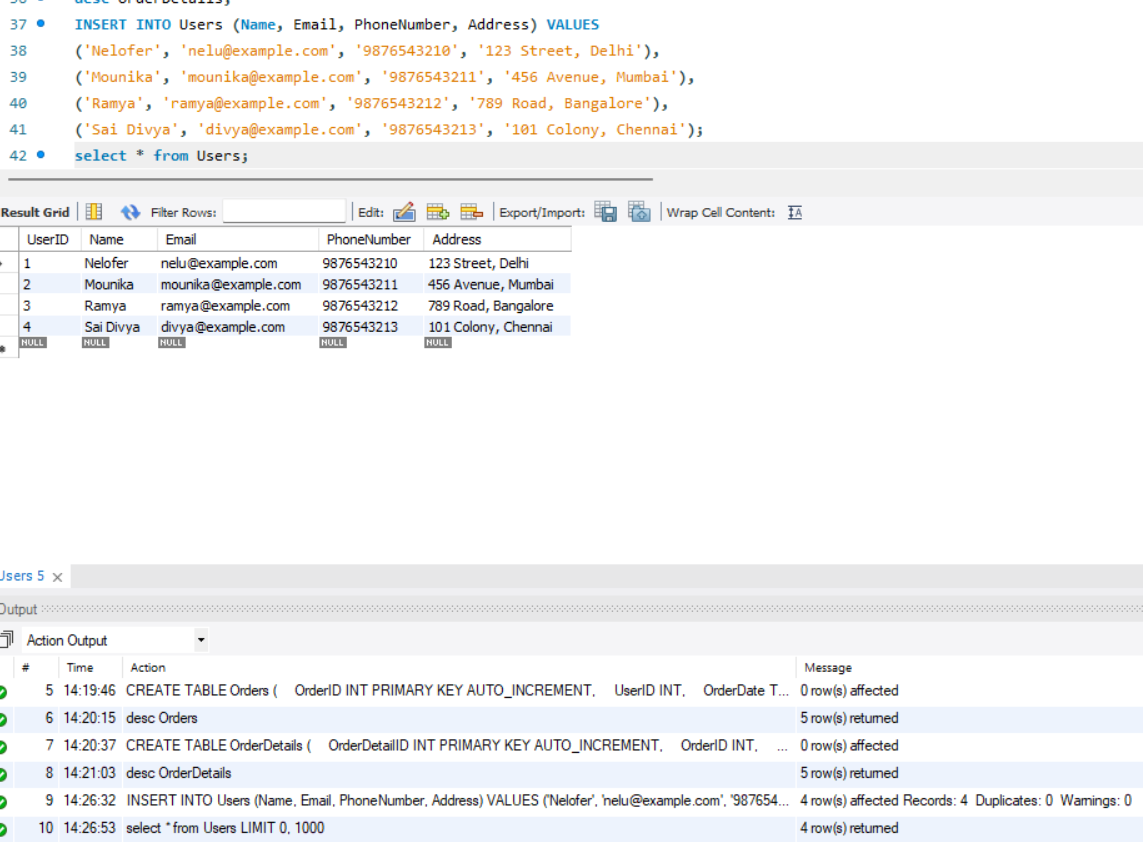
**ORDERDETAIL:**

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Key Type** |
| OrderDetailID | INT | Primary Key (PK), Auto Increment |
| OrderID | INT | Foreign Key (FK) → Orders(OrderID) |
| MobileID | INT | Foreign Key (FK) → Mobiles(MobileID) |
| Quantity | INT | Not Null |
| Subtotal | DECIMAL(10,2) | Not Null |

**INSERTION OF TABLES:**

Sample data is inserted into tables to validate functionality, including:

* User registrations.
* Mobile phone details with pricing and specifications.
* Orders with status tracking.
* Payments associated with orders.
* Order details specifying purchased quantities and costs.



A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**SQL QUERIES:**

**1.Inner join orders, customers, and mobile phones.**

**Obtain cumulative sales by orders joined with payments and users.**

A screenshot of a computer

AI-generated content may be incorrect.

**2.Compute all revenue generated from mobile telephones.**

**Count units sold by mobile brand.**

A screenshot of a computer

AI-generated content may be incorrect.

**3. Find users who have more than one order.**

**Select the most expensive order placed by a user.**

A screenshot of a computer

AI-generated content may be incorrect.

**4.Which users have placed at least one order?**

A screenshot of a computer

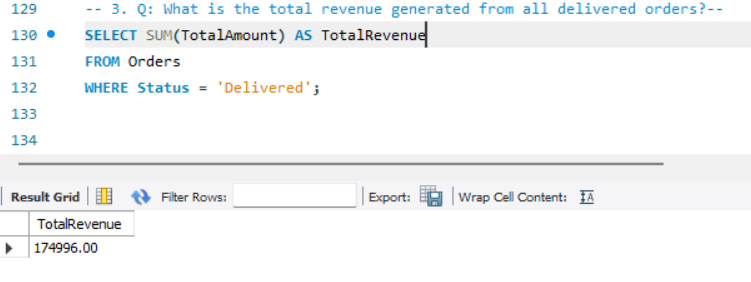
AI-generated content may be incorrect.

**5. What are the details of all orders that are currently 'Pending'?**

A screenshot of a computer

AI-generated content may be incorrect.

**6.What is the total revenue generated from all delivered orders?**



**7. What is the average price of mobiles available in the store?**

A screenshot of a computer

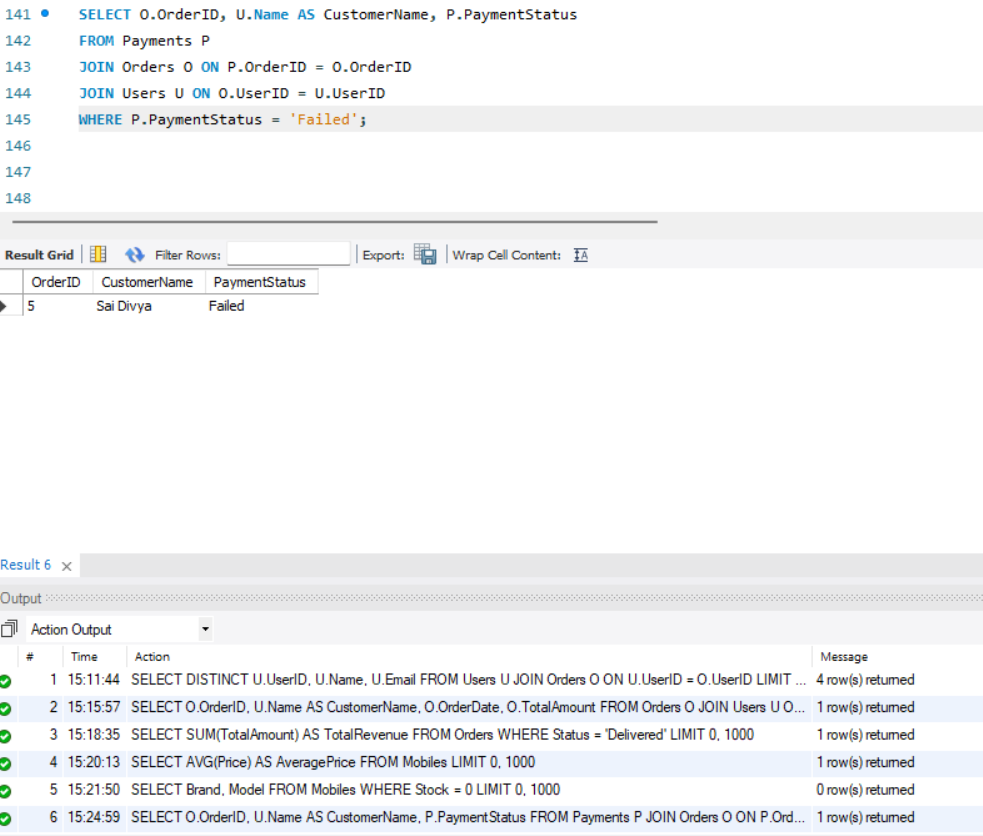
AI-generated content may be incorrect.

**8. Which mobile brands are out of stock?**

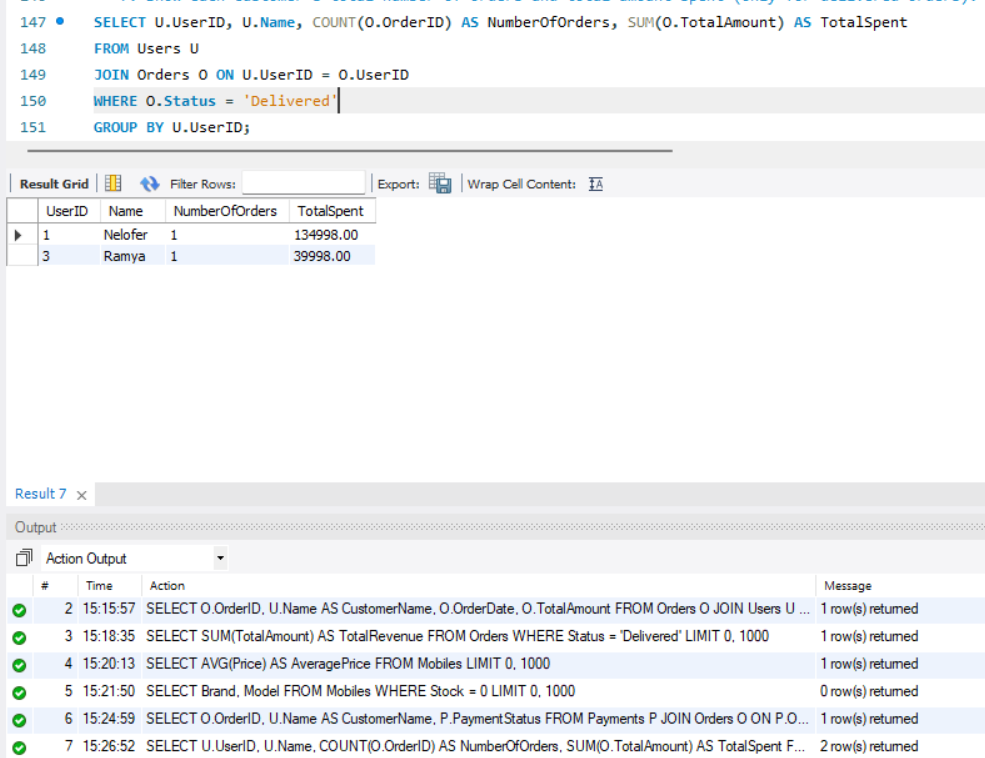
A screenshot of a computer

AI-generated content may be incorrect.

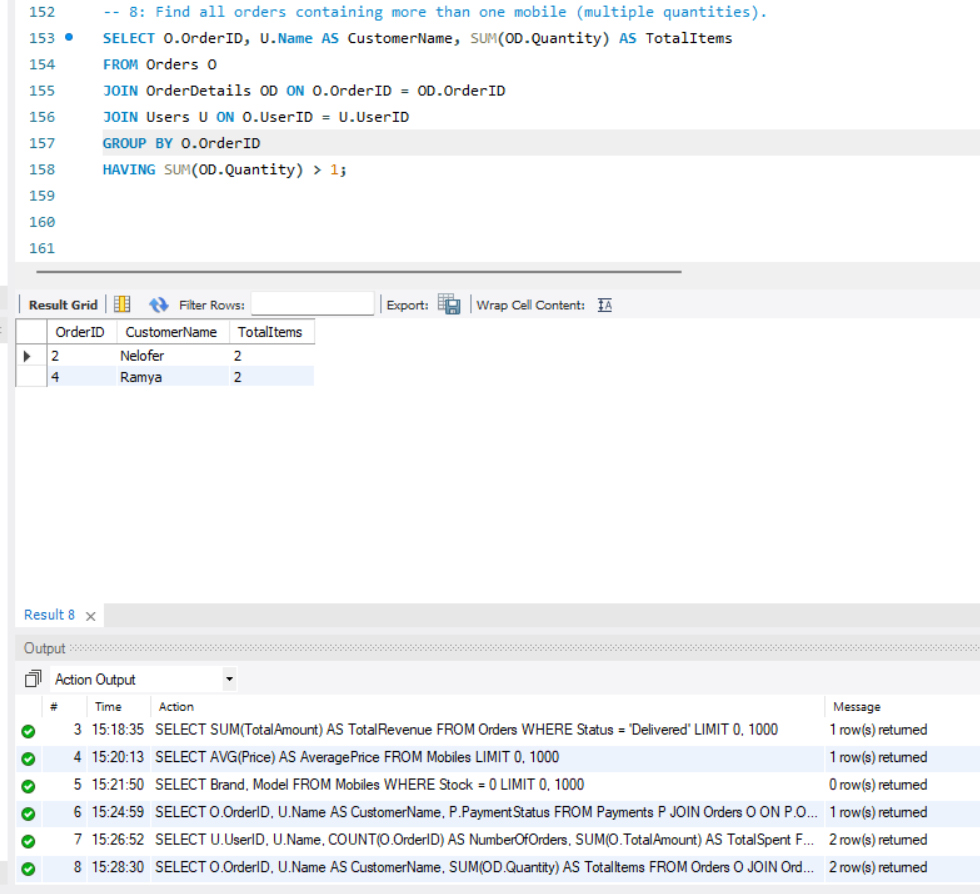
**9.Which orders have failed payments?**



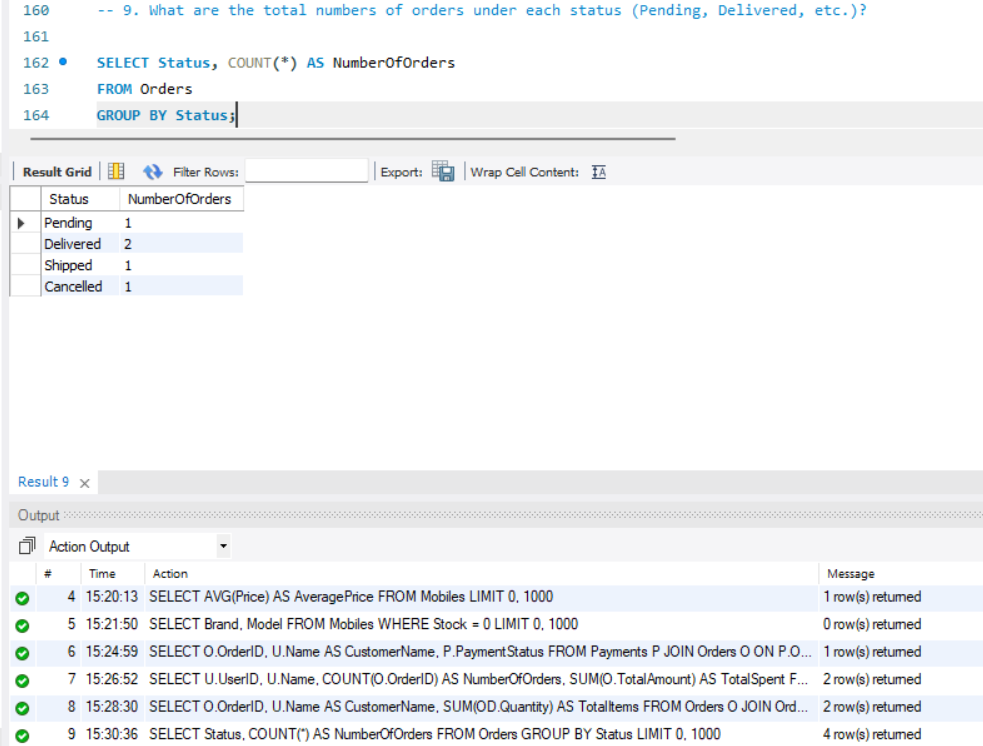
**10. Show each customer's total number of orders and total amount spent (only for delivered orders).**



**11.Find all orders containing more than one mobile (multiple quantities).**



**12. What are the total numbers of orders under each status (Pending, Delivered, etc.)?**



**13.This is the view of a database, OrderSummary, which offers a consolidation of all essential details on orders, payments, and customers for easier reporting and improved data access. Examples of the kind of details included in the view:**

**Name and contacts of the customer.**

**Order details such as purchases and total price.**

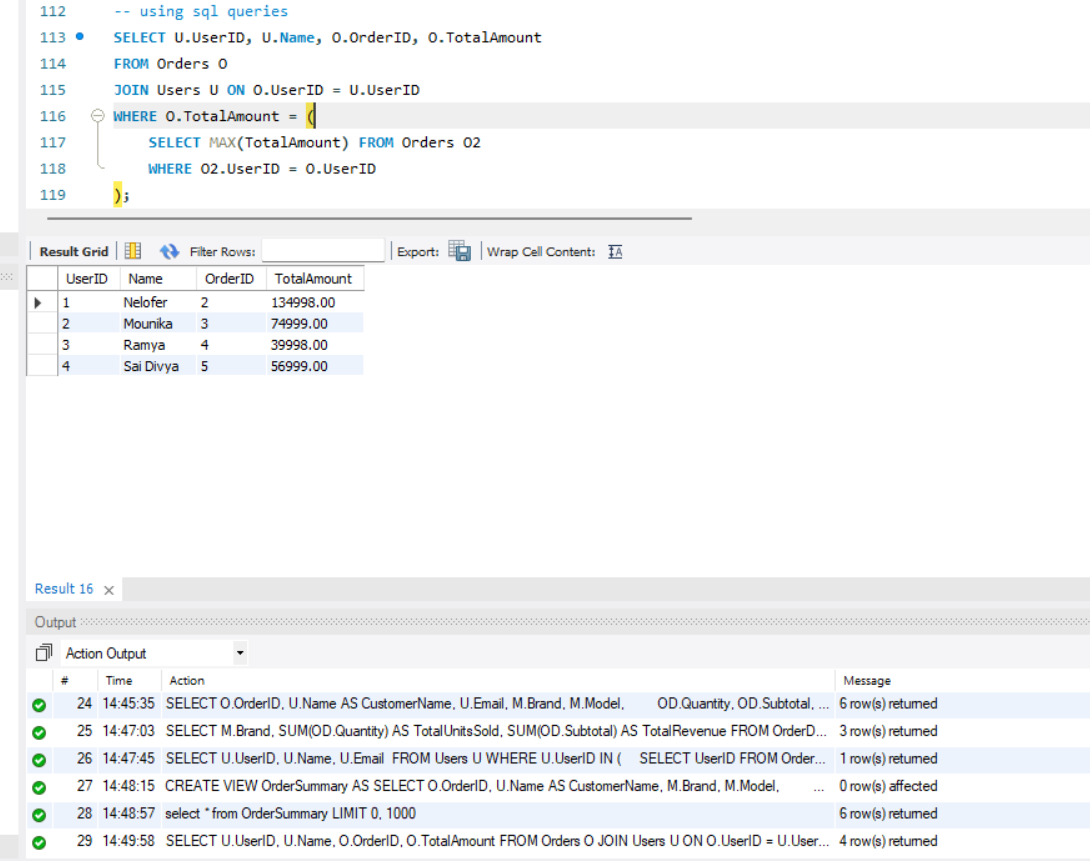
**Payment method and status of transaction.**

**Views enhance data security as only specific views with limited views on some information are made by the surface.**

A screenshot of a computer

AI-generated content may be incorrect.

**14. This query retrieves each user's highest-value order by selecting the maximum TotalAmount per UserID. It joins the Users and Orders tables to display user names alongside their most expensive orders, aiding in customer spending analysis.**



**15. Which users have never placed any order?**

****

**NORMALIZATION**

**NORMALIZATION PROCESS**

**1. First Normal Form (1NF):**

For a table to be in 1NF:

* It must have a primary key.
* Each column should contain atomic values (indivisible, simplest form of data that cannot be further divided.).

**2. Second Normal Form (2NF)**

For a table to be in 2NF:

* It must be in 1NF.
* There should be no partial dependency, i.e., no non-prime attribute should depend on only a part of any composite primary key (prime attributes). Instead, every non-prime attribute must depend on the entire set of prime attributes that form the primary key.

**3. Third Normal Form (3NF)**

For a table to be in 3NF:

* It must be in 2NF.
* It should have no transitive dependencies, i.e., non-prime attributes must not depend on other non-prime attributes.

**NORMALIZATION OF EACH TABLE**

**1.Users Table**

**Attributes**: UserID (PK), Name, Email, PhoneNumber, Address

* **1NF**: Each field contains atomic values. No repeating groups.
* **2NF**: All non-key attributes depend fully on the primary key (UserID).
* **3NF**: No transitive dependencies. Attributes like Email or PhoneNumber do not depend on other non-key attributes.

**2. Mobiles Table**

**Attributes**: MobileID (PK), Brand, Model, Price, Stock, Specifications

* **1NF**: Each mobile record is atomic with distinct fields.
* **2NF**: All attributes are dependent on the entire primary key (MobileID).
* **3NF**: No attribute is dependent on another non-key attribute (e.g., Price does not depend on Brand).

**3. Orders Table**

**Attributes**: OrderID (PK), UserID (FK), OrderDate, TotalAmount, Status

* **1NF**: Each order is a single entry with unique and atomic fields.
* **2NF**: Every field is fully dependent on the primary key (OrderID).
* **3NF**: No transitive dependencies are present. For example, Status depends only on OrderID.

**4. OrderDetails Table**

**Attributes**: OrderDetailID (PK), OrderID (FK), MobileID (FK), Quantity, Subtotal

* **1NF**: Each order detail represents one mobile in one order — no multivalued attributes.
* **2NF**: All non-key attributes are dependent on OrderDetailID.
* **3NF**: There are no non-key attributes depending on other non-key attributes.

**5. Payments Table**

**Attributes**: PaymentID (PK), OrderID (FK), PaymentMethod, PaymentStatus, TransactionID

* **1NF**: Each payment record is atomic and unique.
* **2NF**: All attributes are fully functionally dependent on PaymentID.
* **3NF**: Attributes like PaymentStatus and PaymentMethod do not depend on each other — avoiding transitive dependency.

**Benefits of Normalization in This Database**

* Ensures **data integrity** through use of primary and foreign keys.
* **Eliminates redundancy** by storing repeated information only once.
* Enables **efficient updates** by centralizing frequently changing data.
* Boosts **query performance** with smaller, well-structured tables.
* Improves **data security** by isolating sensitive information.
* Promotes **modular design** by separating entities into logical tables.
* Enhances **scalability** for adding future features or entities.
* **Prevents anomalies** during insert, update, or delete operations.
* Encourages **data reusability** across multiple related transactions.
* Supports **accurate** and efficient reporting through clean data relationships.
* Maintains **data consistency** and integrity (e.g., foreign keys with cascading behavior)

**CONCLUSION:**

The "Online Mobile Shopping Database" provides well-structured, efficient, and scalable solutions to an e-commerce platform mainly dealing with mobile phone sales. This system promotes smooth operations in the handling of user registration, product management, order processing, and payment tracking by enforcing certain principles of relational databases. Data integrity is ensured through the use of foreign keys and other constraints, while redundancy is minimized through the normalization of tables, up to the third normal form (3NF).

With structured tables and optimized queries and well-defined relationships, the database maximizes operational efficiency and ensures that crucial data can be retrieved rapidly. Data joins, aggregate functions, and subqueries are then very useful to analyze data insights, helping businesses understand their customer behavior and sales trends. The views provide a way of accessing the data while securing the information since they abstract sensitive columns.

Therefore, this project has built a firm foundation for a solid and scalable e-commerce system that guarantees effective business operations, enhanced user experience, and secure transaction processing.