Sales Management System[DB]

Step 1: Project Idea Selection

a database system that manages [customers, products, categories, orders, and sales] for a retail environment. The system provides functionalities for tracking customers, organizing products into categories, processing orders, and recording sales.

Step 2: Analyzing

Customer Needs:

- A platform to store and retrieve customer data.
- A way to categorize and manage products.
- Tracking and processing customer orders.
- Recording sales transactions and generating summaries.
- Efficient searching, sorting, and querying of data.

Entities Identified:

- **Customers**: To store client information.
- **Categories**: To classify products into types (e.g., electronics, clothing).
- **Products**: Contains product data like name, price, quantity, etc.
- **Orders**: Represents order requests made by customers.
- **Sales**: Records completed sales transactions.

Users of the System:

- **Admin**: Can manage all tables and perform insert, update, delete operations.
- **Cashier/Salesperson**: Can only view products, insert new orders and record sales.

Step 3: ERD Diagram.

Product and Category (One-to-Many)

- **One category has many products, but each product belongs to exactly one category.**

[product : total , category: partial]

Order and Customer (Many-to-One)

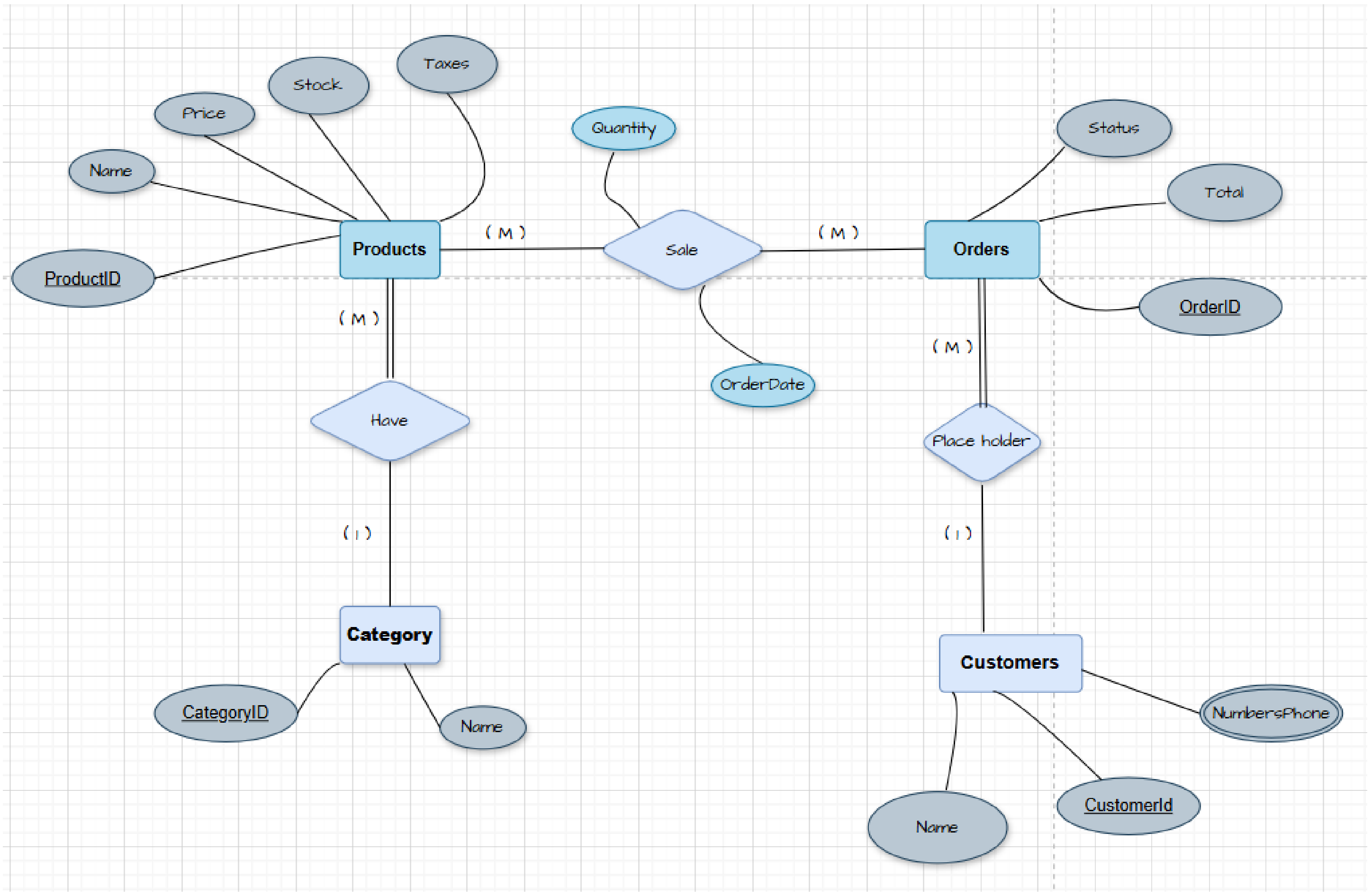
- **Many orders belong to one customer, but each order is made by exactly one customer.**

[order: total , category: partial]

Order and Product (Many-to-Many)

- **One order can have many products, and one product can be in many orders.**

[order: partial , product: partial]



How Mapping Was Done by Men3m?

Each Entity represent One Table

- **Attributes become columns.**
- **Primary Key is added.**

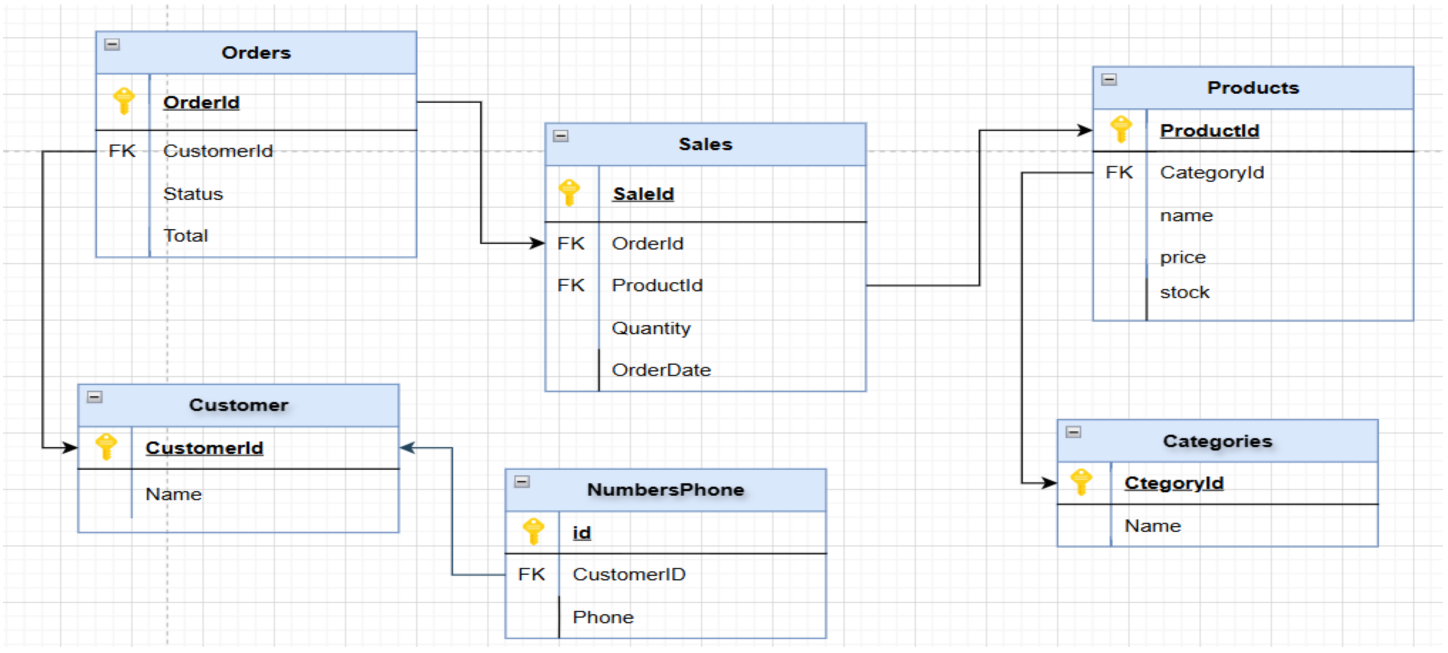
One-to-Many

- **The “partial Entity” ID becomes a foreign key in the "Total" side.**
- **Example: category_id into the Product table.**

Many-to-Many

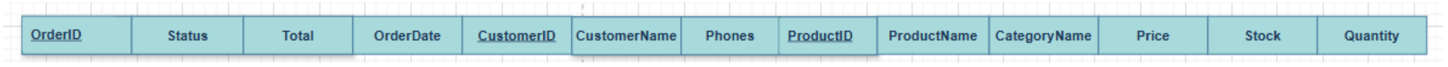
- **Make a bridge table with two foreign keys.**
- **Add extra data if needed (like quantity, date).**

The composite attribute Phone Number was mapped to a separate table to allow storing multiple phone numbers per customer.



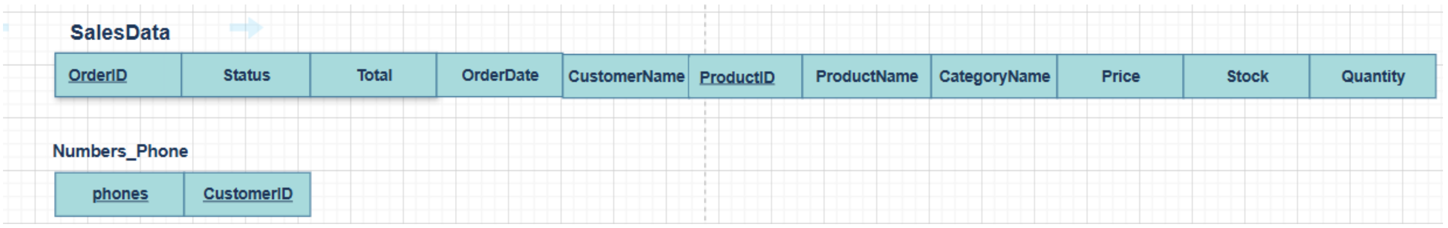
Normalization

0f befory AnyThing



OrderID	Status	Total	OrderDate	CustomerID	CustomerName	Phones	ProductID	ProductName	CategoryName	Price	Stock	Quantity

1F



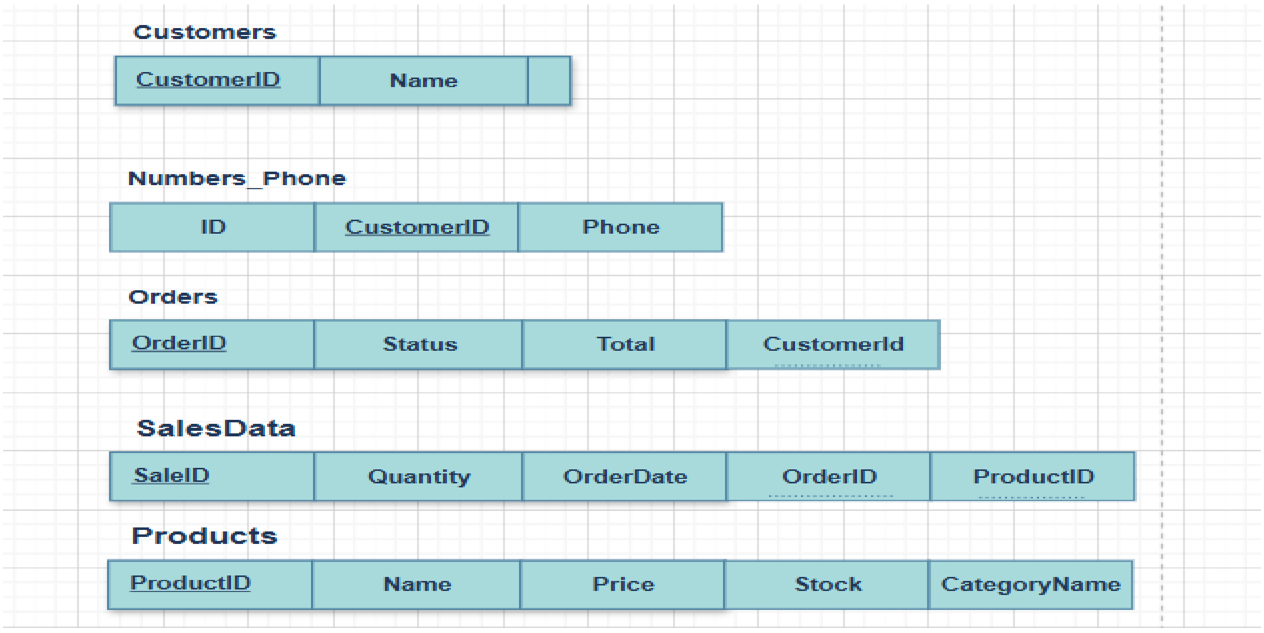
SalesData

OrderID	Status	Total	OrderDate	CustomerName	ProductID	ProductName	CategoryName	Price	Stock	Quantity

Numbers_Phone

phones	CustomerID

2F



Customers

CustomerID	Name	

Numbers_Phone

ID	CustomerID	Phone

Orders

OrderID	Status	Total	CustomerId

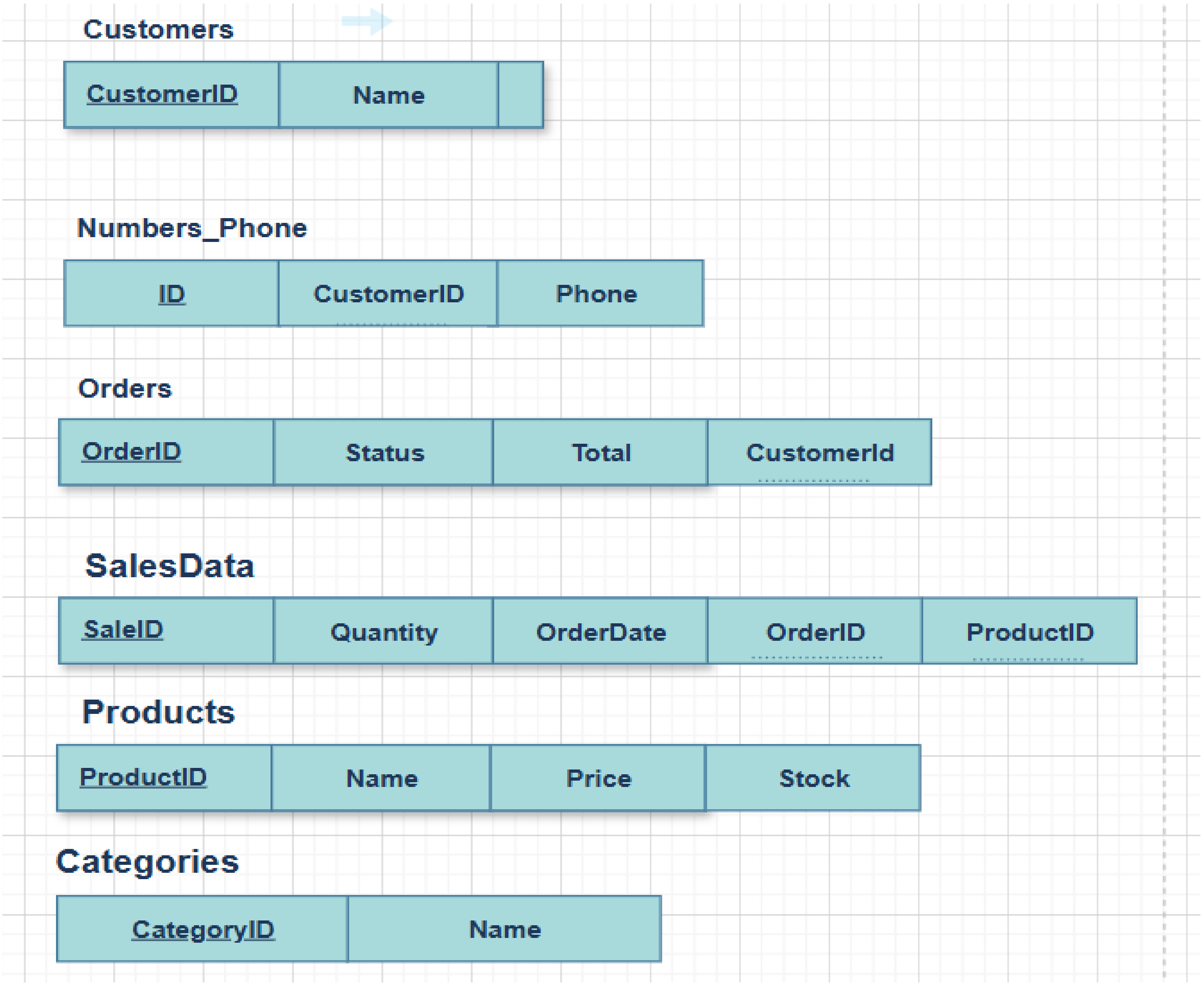
SalesData

SaleID	Quantity	OrderDate	OrderID	ProductID

Products

ProductID	Name	Price	Stock	CategoryName

3NF



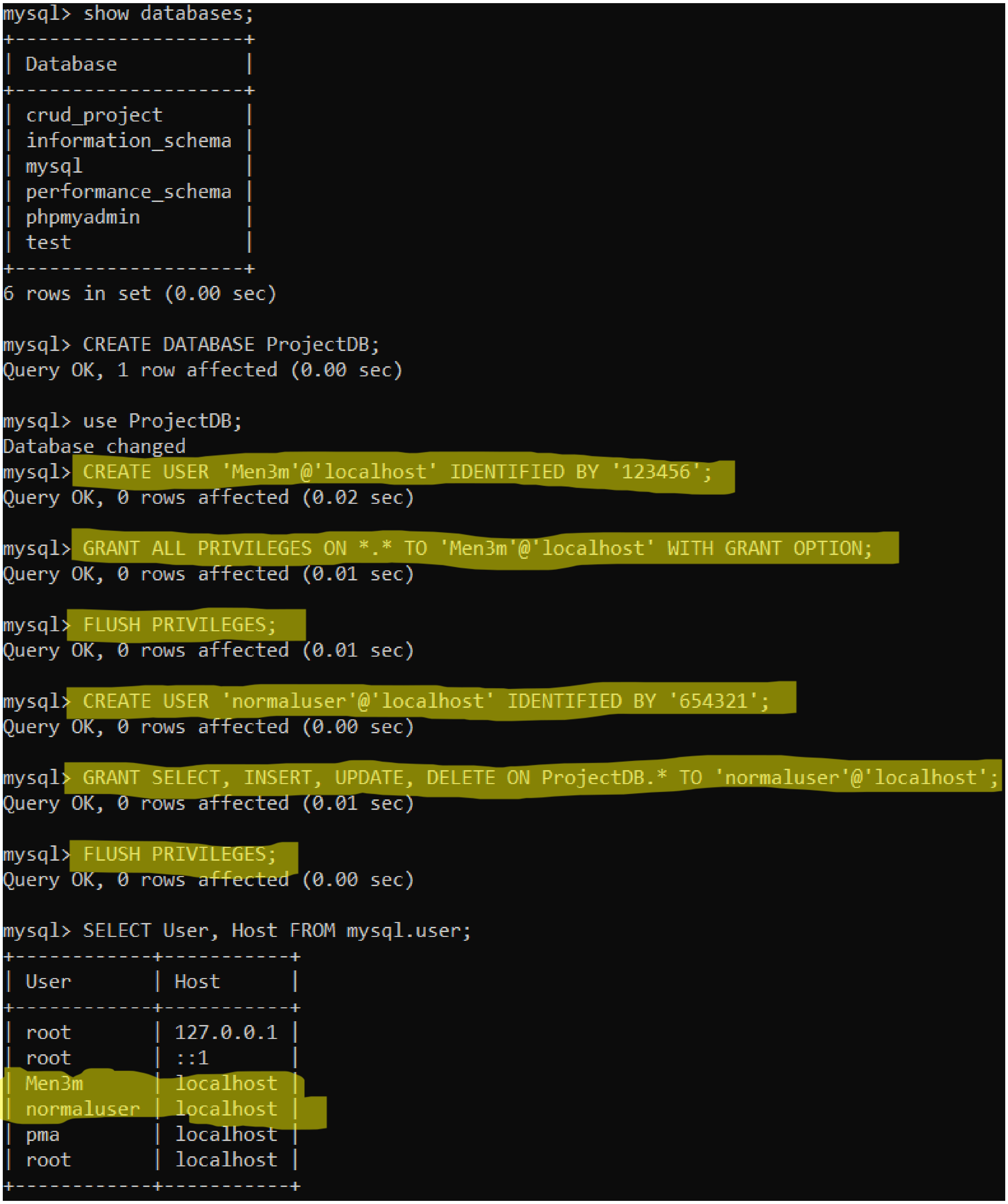
Define user

Men3m (Admin):

Full access to everything. Can manage data, databases, and users.

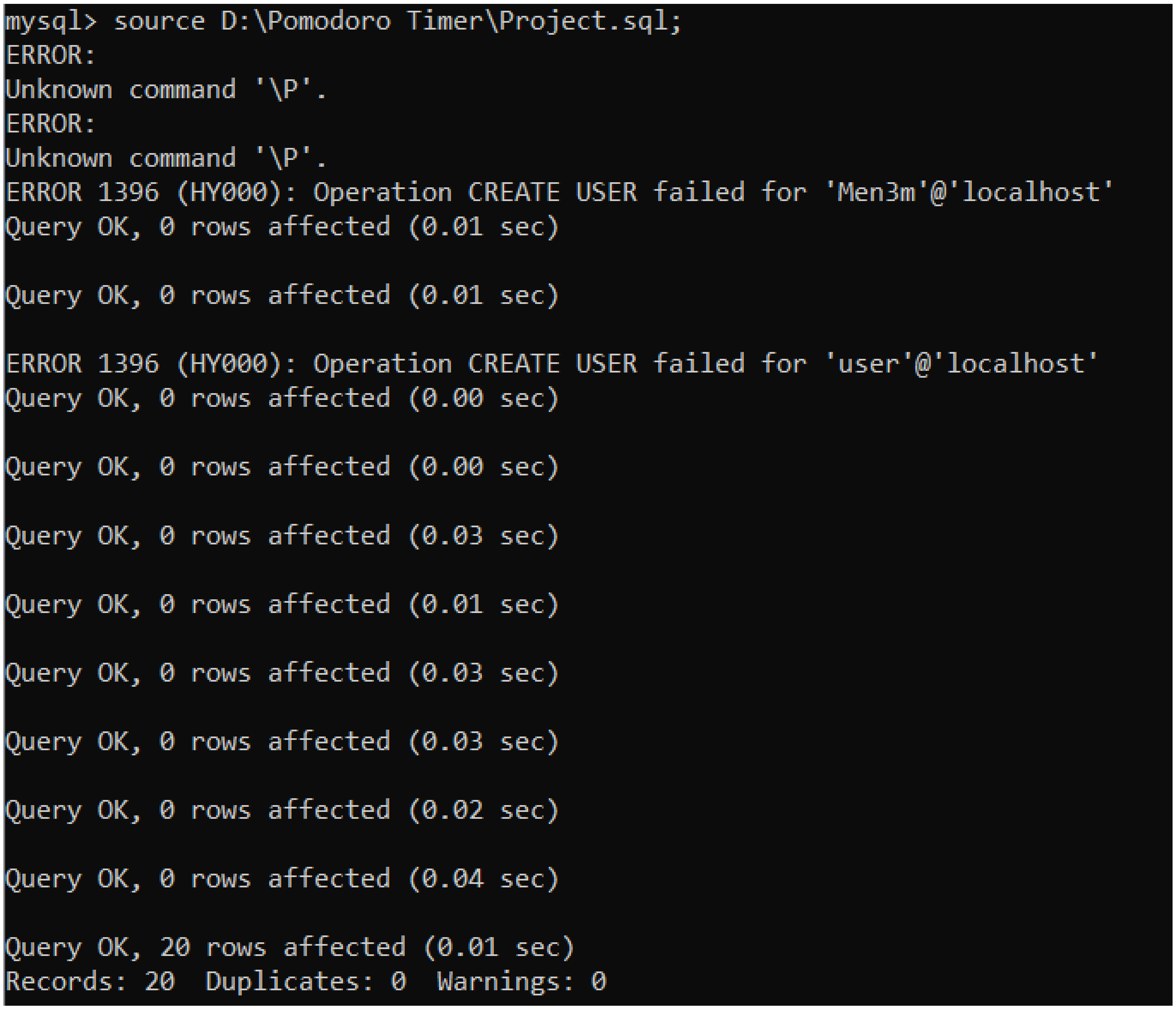
normaluser

Can only read, add, edit, and delete data in ProjectDB.



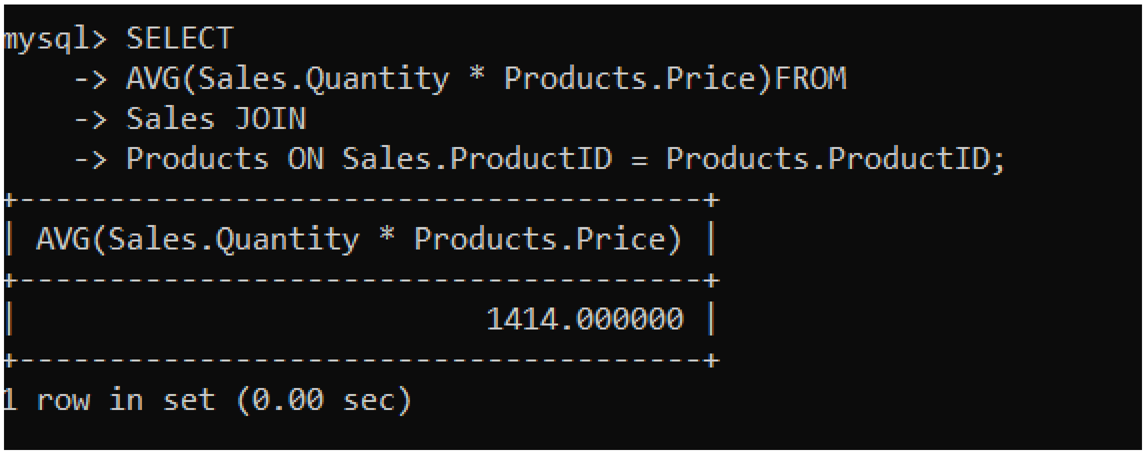
```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| crud_project       |
| information_schema |
| mysql              |
| performance_schema |
| phpmyadmin         |
| test               |
+--------------------+
6 rows in set (0.00 sec)

mysql> CREATE DATABASE ProjectDB;
Query OK, 1 row affected (0.00 sec)

mysql> use ProjectDB;
Database changed
mysql> CREATE USER 'Men3m'@'localhost' IDENTIFIED BY '123456';
Query OK, 0 rows affected (0.02 sec)

mysql> GRANT ALL PRIVILEGES ON *.* TO 'Men3m'@'localhost' WITH GRANT OPTION;
Query OK, 0 rows affected (0.01 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE USER 'normaluser'@'localhost' IDENTIFIED BY '654321';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT SELECT, INSERT, UPDATE, DELETE ON ProjectDB.* TO 'normaluser'@'localhost';
Query OK, 0 rows affected (0.01 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT User, Host FROM mysql.user;
+------------+-----------+
| User       | Host      |
+------------+-----------+
| root       | 127.0.0.1 |
| root       | ::1       |
| Men3m      | localhost |
| normaluser | localhost |
| pma        | localhost |
| root       | localhost |
+------------+-----------+
```

1.Import DB From File project.sql



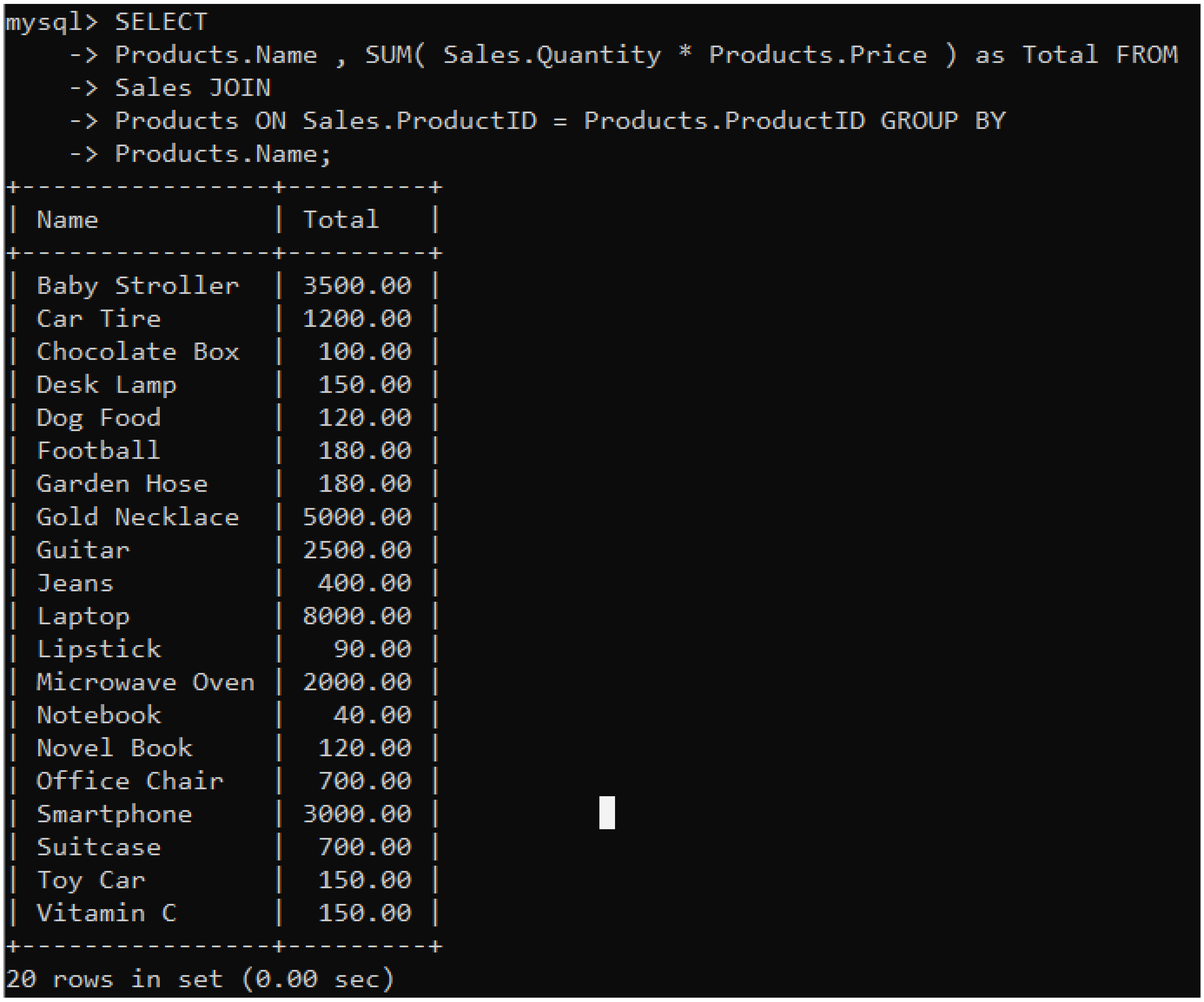
```
mysql> source D:\Pomodoro Timer\Project.sql;
ERROR:
Unknown command '\P'.
ERROR:
Unknown command '\P'.
ERROR 1396 (HY000): Operation CREATE USER failed for 'Men3m'@'localhost'
Query OK, 0 rows affected (0.01 sec)

Query OK, 0 rows affected (0.01 sec)

ERROR 1396 (HY000): Operation CREATE USER failed for 'user'@'localhost'
Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.03 sec)

Query OK, 0 rows affected (0.01 sec)

Query OK, 0 rows affected (0.03 sec)

Query OK, 0 rows affected (0.03 sec)

Query OK, 0 rows affected (0.02 sec)

Query OK, 0 rows affected (0.04 sec)

Query OK, 20 rows affected (0.01 sec)
Records: 20  Duplicates: 0  Warnings: 0
```

2.Calculate Avg total



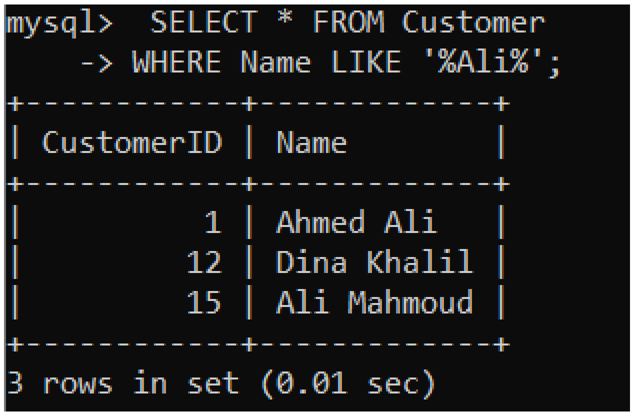
```
mysql> SELECT
    -> AVG(Sales.Quantity * Products.Price)FROM
    -> Sales JOIN
    -> Products ON Sales.ProductID = Products.ProductID;
+--------------------------------------+
| AVG(Sales.Quantity * Products.Price) |
+--------------------------------------+
|                          1414.000000 |
+--------------------------------------+
1 row in set (0.00 sec)
```

3.Find Sum Total each Customer



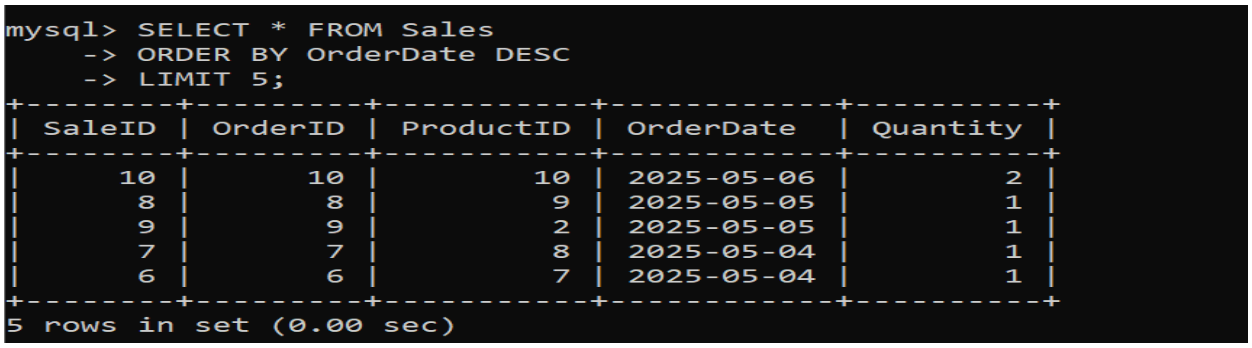
```
mysql> SELECT
    -> Products.Name , SUM( Sales.Quantity * Products.Price ) as Total FROM
    -> Sales JOIN
    -> Products ON Sales.ProductID = Products.ProductID GROUP BY
    -> Products.Name;
+----------------+---------+
| Name           | Total   |
+----------------+---------+
| Baby Stroller  | 3500.00 |
| Car Tire       | 1200.00 |
| Chocolate Box  |  100.00 |
| Desk Lamp      |  150.00 |
| Dog Food       |  120.00 |
| Football       |  180.00 |
| Garden Hose    |  180.00 |
| Gold Necklace  | 5000.00 |
| Guitar         | 2500.00 |
| Jeans          |  400.00 |
| Laptop         | 8000.00 |
| Lipstick       |   90.00 |
| Microwave Oven | 2000.00 |
| Notebook       |   40.00 |
| Novel Book     |  120.00 |
| Office Chair   |  700.00 |
| Smartphone     | 3000.00 |
| Suitcase       |  700.00 |
| Toy Car        |  150.00 |
| Vitamin C      |  150.00 |
+----------------+---------+
20 rows in set (0.00 sec)
```

4.Find Any customer subName = 'Ali'



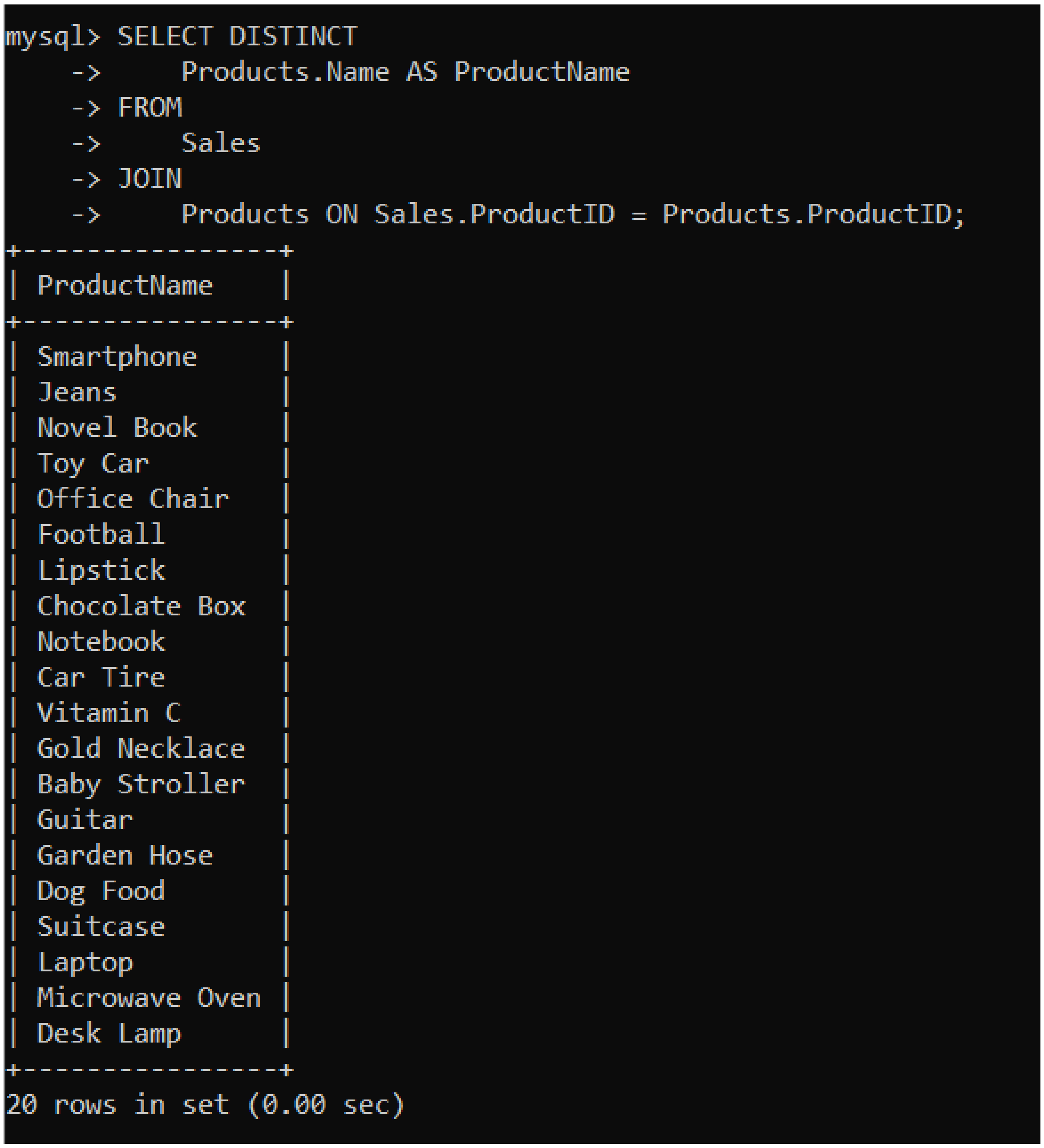
```
mysql>  SELECT * FROM Customer
    -> WHERE Name LIKE '%Ali%';
+------------+-------------+
| CustomerID | Name        |
+------------+-------------+
|          1 | Ahmed Ali   |
|         12 | Dina Khalil |
|         15 | Ali Mahmoud |
+------------+-------------+
3 rows in set (0.01 sec)
```

5.Find first sales depend on order Date Desc



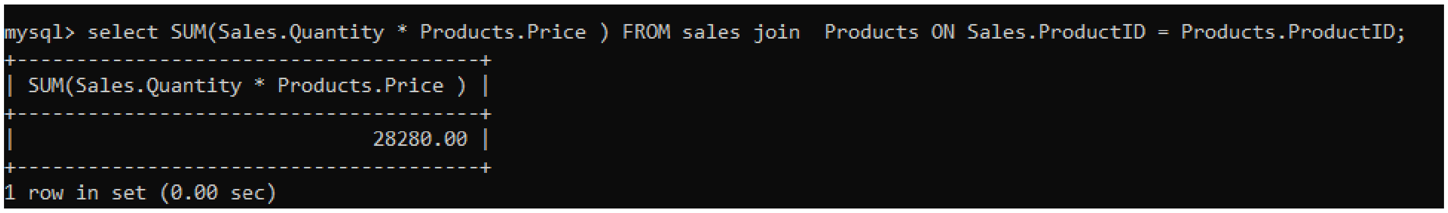
```
mysql> SELECT * FROM Sales
    -> ORDER BY OrderDate DESC
    -> LIMIT 5;
+--------+---------+-----------+------------+----------+
| SaleID | OrderID | ProductID | OrderDate  | Quantity |
+--------+---------+-----------+------------+----------+
|     10 |      10 |        10 | 2025-05-06 |        2 |
|      8 |       8 |         9 | 2025-05-05 |        1 |
|      9 |       9 |         2 | 2025-05-05 |        1 |
|      7 |       7 |         8 | 2025-05-04 |        1 |
|      6 |       6 |         7 | 2025-05-04 |        1 |
+--------+---------+-----------+------------+----------+
5 rows in set (0.00 sec)
```

6.Find unique names of products that have been sold, by joining the sales and products tables



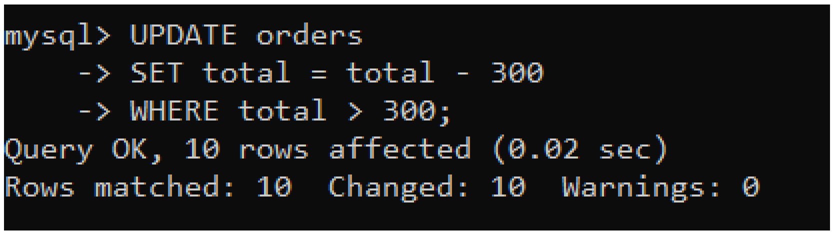
```
mysql> SELECT DISTINCT
    ->     Products.Name AS ProductName
    -> FROM
    ->     Sales
    -> JOIN
    ->     Products ON Sales.ProductID = Products.ProductID;
+----------------+
| ProductName    |
+----------------+
| Smartphone     |
| Jeans          |
| Novel Book     |
| Toy Car        |
| Office Chair   |
| Football       |
| Lipstick       |
| Chocolate Box  |
| Notebook       |
| Car Tire       |
| Vitamin C      |
| Gold Necklace  |
| Baby Stroller  |
| Guitar         |
| Garden Hose    |
| Dog Food       |
| Suitcase       |
| Laptop         |
| Microwave Oven |
| Desk Lamp      |
+----------------+
20 rows in set (0.00 sec)
```

7.Find Sum total



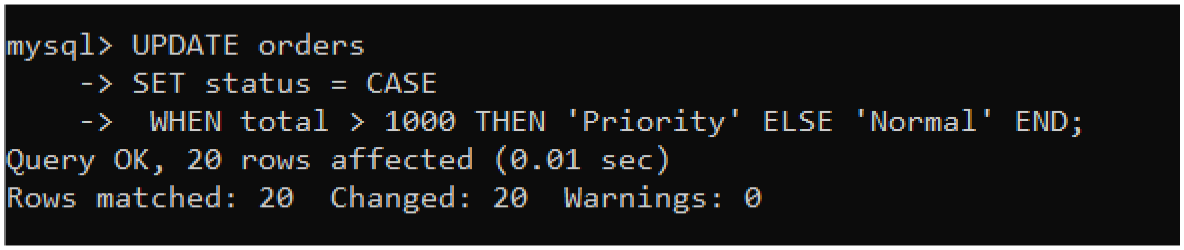
```
mysql> select SUM(Sales.Quantity * Products.Price ) FROM sales join  Products ON Sales.ProductID = Products.ProductID;
+---------------------------------------+
| SUM(Sales.Quantity * Products.Price ) |
+---------------------------------------+
|                              28280.00 |
+---------------------------------------+
1 row in set (0.00 sec)
```

8.Any order has total greater than 300 get offer 300.



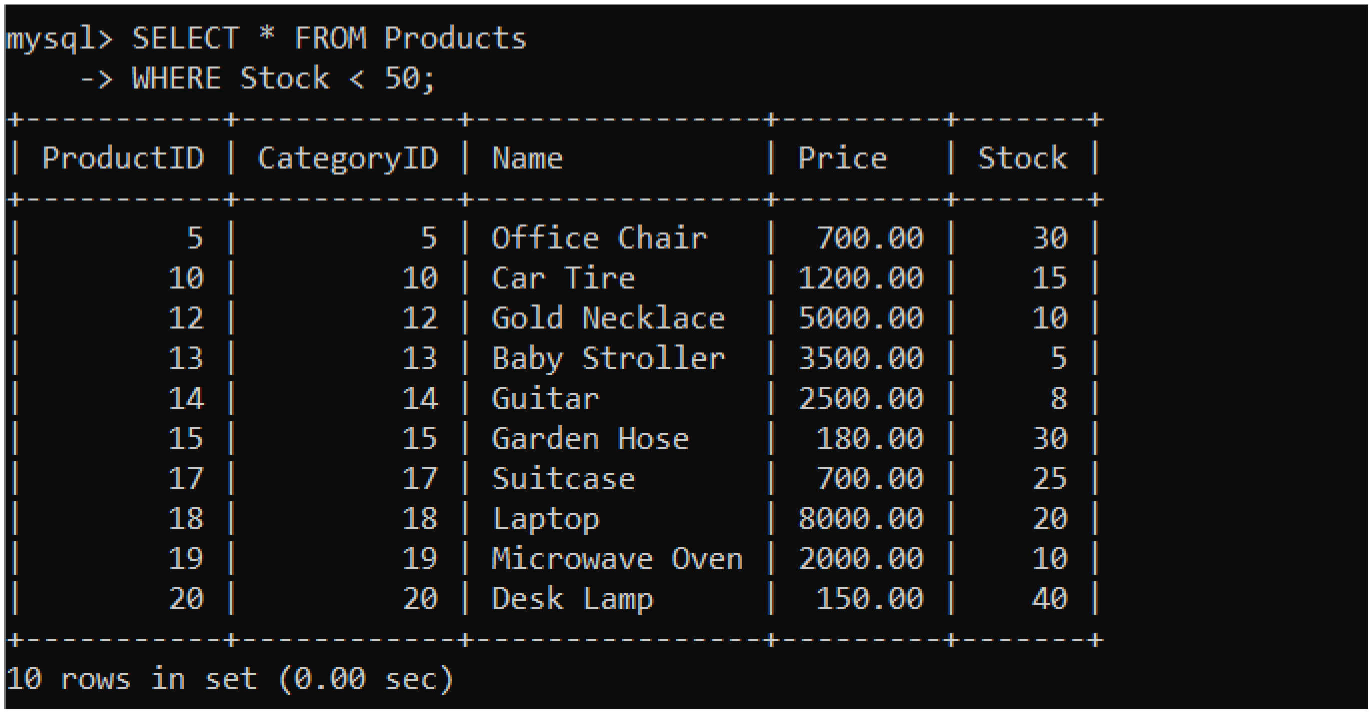
```
mysql> UPDATE orders
    -> SET total = total - 300
    -> WHERE total > 300;
Query OK, 10 rows affected (0.02 sec)
Rows matched: 10  Changed: 10  Warnings: 0
```

9.Any order total > 1000 Status = ‘priority’ else ‘Normal’



```
mysql> UPDATE orders
    -> SET status = CASE
    ->  WHEN total > 1000 THEN 'Priority' ELSE 'Normal' END;
Query OK, 20 rows affected (0.01 sec)
Rows matched: 20  Changed: 20  Warnings: 0
```

10.find any product has Stock < 50



```
mysql> SELECT * FROM Products
    -> WHERE Stock < 50;
+-----------+------------+----------------+---------+-------+
| ProductID | CategoryID | Name           | Price   | Stock |
+-----------+------------+----------------+---------+-------+
|         5 |          5 | Office Chair   |  700.00 |    30 |
|        10 |         10 | Car Tire       | 1200.00 |    15 |
|        12 |         12 | Gold Necklace  | 5000.00 |    10 |
|        13 |         13 | Baby Stroller  | 3500.00 |     5 |
|        14 |         14 | Guitar         | 2500.00 |     8 |
|        15 |         15 | Garden Hose    |  180.00 |    30 |
|        17 |         17 | Suitcase       |  700.00 |    25 |
|        18 |         18 | Laptop         | 8000.00 |    20 |
|        19 |         19 | Microwave Oven | 2000.00 |    10 |
|        20 |         20 | Desk Lamp      |  150.00 |    40 |
+-----------+------------+----------------+---------+-------+
10 rows in set (0.00 sec)
```