**SVKM’s NMIMS**

**School of Technology Management & Engineering (Indore Campus)**

**Computer Engineering Department (B Tech CE Sem IV)**

**Database Management System**

**Project Report**

|  |  |  |
| --- | --- | --- |
| Program: | BTech CE Section - B | |
| Semester | IV | |
| Name of the Project: | Zyra Clothing Brand | |
|  | | |
| Details of Project Members |  |  |
| Batch | Roll No. | Name |
| 01 | D102 | Jatin Dhawad |
| 01 | D045 | Lakshya Sitlani |
| 01 | D072 | Riddhi Hirani |
| Date of Submission: | | |

**Contribution of each project Members:**

|  |  |  |
| --- | --- | --- |
| Roll No. | Name: | Contribution |
| D045 | Lakshya Sitlani | Backend,Queries |
| D072 | Riddhi Hirani | Backend,Queries |
| D102 | Jatin Dhawad | Frontend,Backend,Queries |

**Github link of your project:** [**Github: Zyra**](https://github.com/JatinDhawad/Zyra)

**Note:**

1. Create a readme file if you have multiple files
2. All files must be properly named (Example:R004\_DBMSProject)
3. Submit all relevant files of your work ( Report, all SQL files, Any other files)
4. **Plagiarism is highly discouraged (Your report will be checked for plagiarism)**

**Rubrics for the Project evaluation:**

|  |  |
| --- | --- |
| First phase of evaluation:  Innovative Ideas (5 Marks)  Design and Partial implementation (5 Marks) | 10 marks |
| Final phase of evaluation  Implementation, presentation and viva, Self-Learning and Learning Beyond classroom | 10 marks |

**Project Report**

**Selected Topic: Zyra – Clothing Brand**

**By**

**Jatin Dhawad, Roll number: D102**

**Riddhi Hirani, Roll number: D072**

**Lakshya Sitlani, Roll number: D045**

**Course: DBMS**

**AY: 2024-25**

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Sr no.** | **Topic** | **Page no.** |
| **1** | Storyline |  |
| **2** | Components of Database Design |  |
| **3** | Entity Relationship Diagram |  |
| **4** | Relational Model |  |
| **5** | Normalization |  |
| **6** | SQL Queries |  |
| **7** | Learning from the Project |  |
| **8** | Project Demonstration |  |
| **9** | Self-learning beyond classroom |  |
| **10** | Learning from the project |  |
| **8** | Challenges faced |  |
| **9** | Conclusion |  |

**I. Storyline**

In today's fast-paced world, style, comfort, and sustainability are at the forefront of fashion. Our brand, **ZYRA**, is dedicated to empowering individuals to express their unique style while making a positive impact on the planet. We aim to revolutionize the clothing industry by offering high-quality, sustainable clothing options that combine fashion-forward designs with eco-friendly materials.

Our clothing brand provides a wide range of stylish apparel designed to fit every body type, occasion, and lifestyle. Whether you’re looking for casual wear, office attire, or something more formal, EcoChic has you covered.

### Key Features of ZYRA:

1. **Personalized Styling:**
   * **Style Profile**: Users create a style profile to receive personalized fashion recommendations based on their preferences, body shape, and lifestyle needs.
   * **Virtual Try-On**: A cutting-edge virtual try-on tool allows users to see how clothes will look on them before making a purchase.
2. **Sustainable Collections:**
   * **Eco-Friendly Materials**: We offer a wide variety of clothing made from organic cotton, recycled fabrics, and other sustainable materials.
   * **Transparent Sourcing**: Our brand is committed to transparency. Users can track the sourcing and production process of each item, ensuring ethical practices are followed.
   * **Low-Waste Manufacturing**: Our production processes minimize waste by using advanced technology and sustainable methods.
3. **Product Customization:**
   * **Custom Designs**: Users can personalize certain products by choosing colors, patterns, and even adding initials or unique designs to make their garments one-of-a-kind.
   * **Limited-Edition Collections**: Exclusive, limited-edition releases that provide customers with unique, high-quality items that cannot be found anywhere else.
4. **Inclusive Sizing:**
   * **Size Variety**: We believe everyone should have access to fashionable, comfortable clothing. Our sizing ranges from petite to plus size, ensuring that all individuals can find their perfect fit.
   * **Fit Guarantee**: If the fit isn’t right, we offer easy returns and exchanges to ensure customer satisfaction.
5. **Lifestyle Integration:**
   * **Outfit Builder**: A feature that allows users to mix and match pieces to create complete outfits based on their wardrobe needs and personal style.
   * **Seasonal Wardrobe Guide**: Personalized wardrobe suggestions for every season, helping users curate their perfect seasonal collection.
6. **Efficient Shopping Experience:**
   * **Easy Browsing & Filters**: Customers can browse by category, style, size, and fabric preference, making it simple to find exactly what they’re looking for.
   * **Secure Checkout & Fast Delivery**: A secure payment gateway and fast, reliable delivery options ensure a smooth shopping experience.
7. **Sustainable Packaging:**
   * **Eco-Friendly Packaging**: All products are packaged using recyclable and biodegradable materials, aligning with our mission to reduce waste and support sustainability.
8. **Community & Impact:**
   * **Giving Back Program**: For every purchase made, a portion of the proceeds is donated to environmental organizations and community initiatives supporting sustainability.
   * **Sustainability Challenges**: Engage our community in sustainability challenges, like recycling old clothes or reducing carbon footprints, rewarding participants with discounts and eco-friendly rewards.

**II. Components of Database Design**

Our database design consists of the following entities:

#### ****Entities and Attributes:****

#### ****1.**** Users (Stores customer details)

• user\_id (PK, INT)

• first\_name (VARCHAR)

• last\_name (VARCHAR)

• email (VARCHAR, UNIQUE)

• password\_hash (VARCHAR)

• phone\_number (VARCHAR)

• address (TEXT)

• created\_at (DATETIME)

2. Products (Stores clothing items)

• product\_id (PK, INT)

• name (VARCHAR)

• description (TEXT)

• price (DECIMAL)

• category\_id (FK -> Categories)

• brand\_id (FK -> Brands)

• stock\_quantity (INT)

• created\_at (DATETIME)

3. Categories (Clothing types like shirts, jeans, etc.)

• category\_id (PK, INT)

• category\_name (VARCHAR)

4. Brands (Stores clothing brands)

• brand\_id (PK, INT)

• brand\_name (VARCHAR)

5. Orders (Tracks customer purchases)

• order\_id (PK, INT)

• user\_id (FK -> Users)

• order\_date (DATETIME)

• total\_amount (DECIMAL)

• status (ENUM: ‘Pending’, ‘Shipped’, ‘Delivered’, ‘Cancelled’)

6. Order\_Items (Tracks items in each order)

• order\_item\_id (PK, INT)

• order\_id (FK -> Orders)

• product\_id (FK -> Products)

• quantity (INT)

• price\_at\_purchase (DECIMAL)

7. Payments (Stores payment transactions)

• payment\_id (PK, INT)

• order\_id (FK -> Orders)

• user\_id (FK -> Users)

• amount (DECIMAL)

• payment\_method (ENUM: ‘Credit Card’, ‘PayPal’, ‘Bank Transfer’)

• status (ENUM: ‘Pending’, ‘Completed’, ‘Failed’)

• transaction\_date (DATETIME)

8. Reviews (Stores customer reviews on products)

• review\_id (PK, Int)

• user\_id (FK -> Users)

• product\_id (FK -> Products)

• rating (INT CHECK 1-5)

• review\_text (TEXT)

• created\_at (DATETIME)

9. Inventory (Manages stock updates)

• inventory\_id (PK, INT)

• product\_id (FK -> Products)

• stock\_added (INT)

• stock\_removed (INT)

• updated\_at (DATETIME)

10. Wishlist (Stores user wishlists)

• wishlist\_id (PK, INT)

• user\_id (FK -> Users)

• product\_id (FK -> Products)

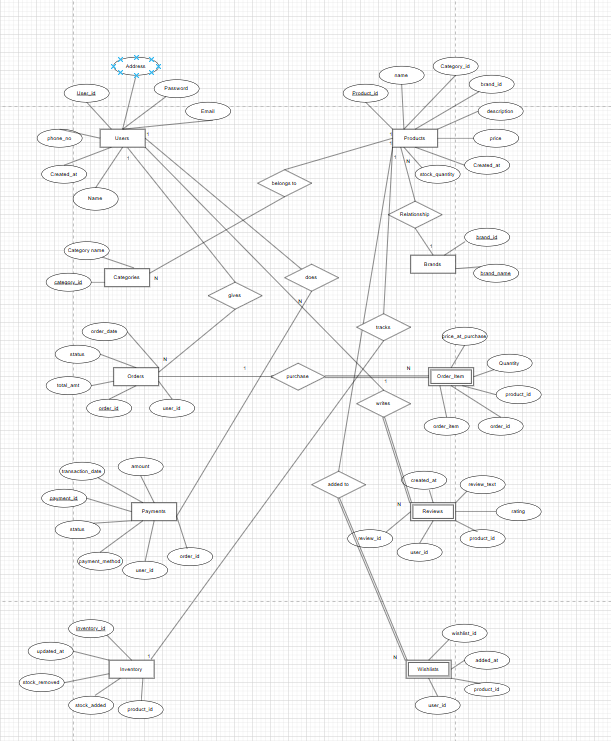
• added\_at (DATETIME)

**III. Entity Relationship Diagram**

The software we used to draw ER model is Drawio

ER diagram is:-

Link to ER Diagram: [ER Diagram](https://tinyurl.com/ycymkfm8)



**IV. Relational Model**

Various tables obtained are:-

1. User

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| UserID(Primary key) | Name | Email | Password | Address | Created\_at | Phone\_number |
|  |  |  |  |  |  |  |

* Schema:User(UserID,Name Email,Pasword,Address,Created\_at,Contact\_number)

1. Orders

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Order\_ID(Primary Key) | UserID(Foreign key) | OrderDate | TotalAmount | Status |
|  |  |  |  |  |

* Schema: Orders(Order\_ID,UserID,OrderDate,TotalAmount,Status)

1. Order\_Items

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ProductID(Foreign key) | Quantity | Order\_ID | Price\_at\_Purchase | |
|  |  |  |  |

* Schema: order\_details(OrderID,Product\_ID,Quantity,Price\_at\_Purchase,Order\_Item)

1. Products

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ProductID(Primary Key) | Name | Stock  Quantity | Price | Description | Created\_at | Brand\_ID(Foreign Key) | CategoryID(Foreign key) |
|  |  |  |  |  |  |  |  |

* Schema: Products(ProductID,ProductName,Stock\_Quantity,Price,Description,CategoryID,Created\_at)

1. category

|  |  |
| --- | --- |
| CategoryID(Primary Key) | CategoryName |
|  |  |

* Schema: Categories(CategoryID,CategoryName)

1. Brand

|  |  |
| --- | --- |
| Brand\_ID(Primary key) | Brand\_Name |
|  |  |

* Schema:Brand(Brand\_ID,Brand\_Name)

7)Payments

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Payment\_Id(Primary Key) | Order\_ID(Foreign Key) | User\_ID(Foreign Key) | Amount | Payment\_Method | Status | Transaction\_Data |
|  |  |  |  |  |  |  |

8)Reviews

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Review\_ID(Primary Key) | User\_ID(Foreign Key) | Product\_ID(Foreign key) | Rating | Review\_Text | Created\_at |
|  |  |  |  |  |  |

9)Inventory

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Inventory\_ID(Primary Key) | Product\_ID(Foreign Key) | Stock\_Added | Stock\_Removed | Updated\_at |
|  |  |  |  |  |

10)Wishlist

|  |  |  |  |
| --- | --- | --- | --- |
| Wishlist\_ID(Primary Key) | User\_ID(Foreign Key) | Product\_ID(Foreign Key) | Added\_at |
|  |  |  |  |

V.Normalization

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

### Requirements:

* Each table has a **primary key**
* All attributes contain **atomic values**
* No **repeating groups** or arrays

### Evaluation of Current Schema:

All tables in the e-commerce schema satisfy 1NF:

1. Every table has a well-defined **primary key** (e.g., user\_id, product\_id, etc.)
2. All fields contain **atomic values** (e.g., email, rating, review\_text)
3. No repeating columns or grouped values (e.g., no multi-product fields in Orders)

### ****Example – Products Table in 1NF****

| **product\_id (PK)** | **name** | **description** | **price** | **category\_id** | **brand\_id** | **stock\_quantity** | **created\_at** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 101 | T-shirt | 100% Cotton | 19.99 | 1 | 1 | 50 | 2023-01-15 10:00:00 |
| 102 | Jeans | Slim fit jeans | 49.99 | 2 | 2 | 30 | 2023-01-16 11:00:00 |

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

### Requirements:

* Must be in **1NF**
* All **non-key attributes** are fully dependent on the **entire primary key**
* No **partial dependencies**

### Evaluation:

Most tables use **single-column primary keys** → automatically satisfy 2NF  
Tables with **composite keys** (like Order\_Items) are also properly normalized:

* Each non-key field is fully dependent on the full composite key (order\_id, product\_id or order\_item\_id)

### ****Example – Order\_Items Table in 2NF****

| **order\_item\_id (PK)** | **order\_id (FK)** | **product\_id (FK)** | **quantity** | **price\_at\_purchase** |
| --- | --- | --- | --- | --- |
| 1 | 1001 | 101 | 2 | 19.99 |
| 2 | 1001 | 102 | 1 | 49.99 |

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

### Requirements:

* Must be in **2NF**
* **No transitive dependencies** (i.e., non-key fields depending on other non-key fields)

### Evaluation:

The schema eliminates transitive dependencies by:

* Storing category and brand info in separate reference tables
* Using foreign keys like user\_id, brand\_id, category\_id
* Removing calculated or derived fields (e.g., total cost per item is not stored in the Products table)

### ****Example – Users Table in 3NF****

| **user\_id (PK)** | **first\_name** | **last\_name** | **email** | **password\_hash** | **phone\_number** | **address** | **created\_at** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | John | Doe | [john@example.com](mailto:john@example.com) | hashed\_pw\_1 | 9876543210 | 123 Main St | 2023-01-10 09:00:00 |
| 2 | Alice | Smith | [alice@example.com](mailto:alice@example.com) | hashed\_pw\_2 | 9123456789 | 456 Elm St | 2023-01-11 10:30:00 |

## ****4. Boyce-Codd Normal Form (BCNF)****

### Requirements:

* Must be in **3NF**
* Every **determinant is a superkey** Evaluation:
* All tables are free from **partial** and **transitive dependencies**
* Foreign keys reference **normalized tables**
* All non-key attributes depend only on **primary keys**

### ****Example – Payments Table in BCNF****

| **payment\_id (PK)** | **order\_id (FK)** | **user\_id (FK)** | **amount** | **payment\_method** | **status** | **transaction\_date** |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 1001 | 1 | 89.97 | Credit Card | Completed | 2023-04-01 12:00:00 |
| 2 | 1002 | 2 | 49.99 | PayPal | Pending | 2023-04-02 17:00:00 |

## Normalization Summary Table

| **Table** | **1NF** | **2NF** | **3NF** | **BCNF** | **Notes** |
| --- | --- | --- | --- | --- | --- |
| Users | ✓ | ✓ | ✓ | ✓ | All fields atomic; email is unique; no derived attributes |
| Products | ✓ | ✓ | ✓ | ✓ | Separated category and brand; no redundant data |
| Categories | ✓ | ✓ | ✓ | ✓ | Simple lookup table |
| Brands | ✓ | ✓ | ✓ | ✓ | Simple lookup table |
| Orders | ✓ | ✓ | ✓ | ✓ | All attributes tied to order\_id; no redundant user/product info |
| Order\_Items | ✓ | ✓ | ✓ | ✓ | Fully dependent on composite key; price stored to preserve history |
| Payments | ✓ | ✓ | ✓ | ✓ | Linked by order\_id; no redundant customer info |
| Reviews | ✓ | ✓ | ✓ | ✓ | All attributes depend on review\_id; user/product referenced by FK |
| Inventory | ✓ | ✓ | ✓ | ✓ | Tracks changes without duplication |
| Wishlist | ✓ | ✓ | ✓ | ✓ | No transitive or derived data; clean foreign key relationships |

SQL Queries

create DATABASE ZYRA1;

USE ZYRA1;



### 1. Users Table

CREATE TABLE Users (

user\_id INT PRIMARY KEY AUTO\_INCREMENT,

first\_name VARCHAR(100),

last\_name VARCHAR(100),

email VARCHAR(150) UNIQUE,

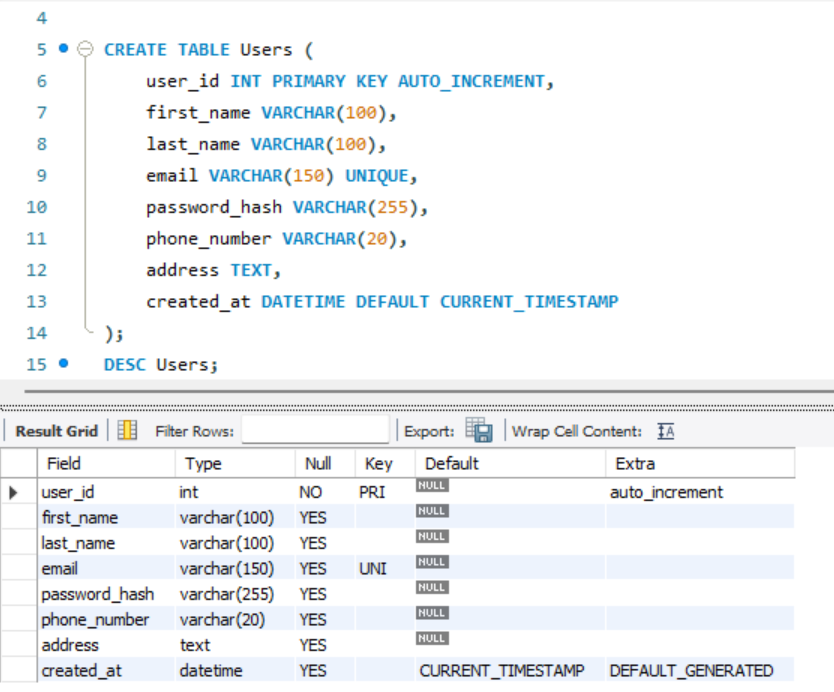
password\_hash VARCHAR(255),

phone\_number VARCHAR(20),

address TEXT,

created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP

);



### 2. Products Table

CREATE TABLE Products (

product\_id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100),

description TEXT,

price DECIMAL(10,2),

category\_id INT,

brand\_id INT,

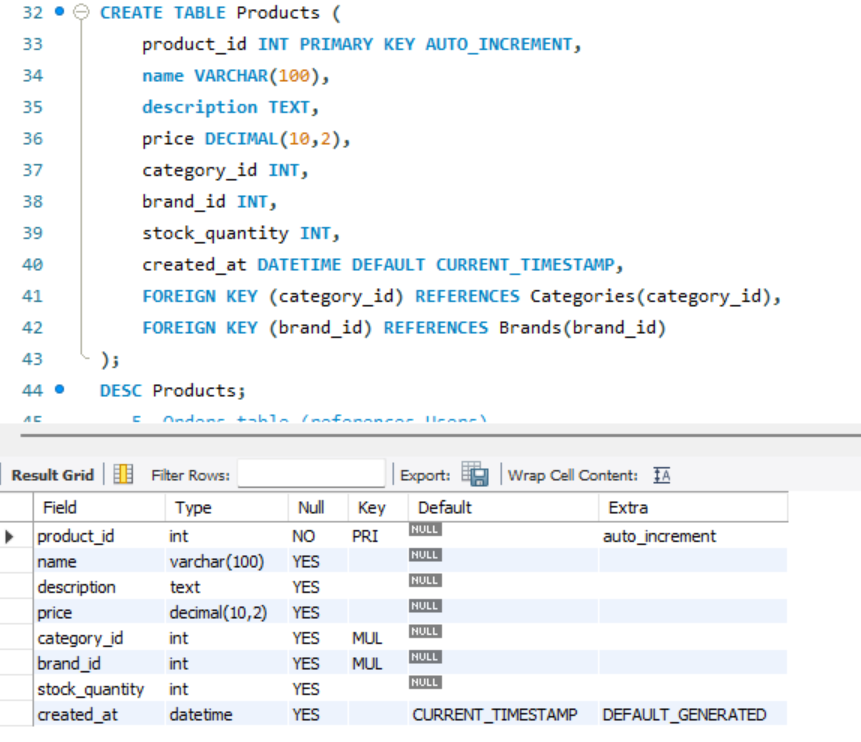
stock\_quantity INT,

created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (category\_id) REFERENCES Categories(category\_id),

FOREIGN KEY (brand\_id) REFERENCES Brands(brand\_id)

);



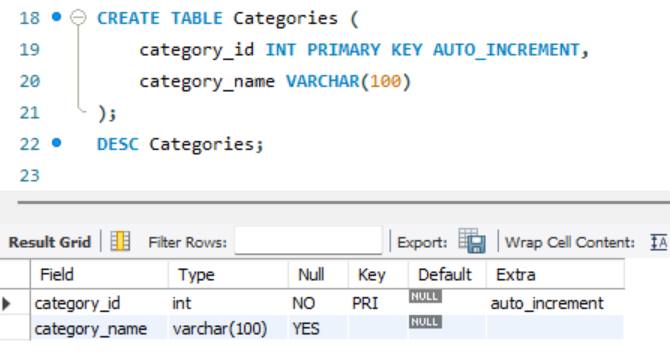
### 3. Categories Table

CREATE TABLE Categories (

category\_id INT PRIMARY KEY AUTO\_INCREMENT,

category\_name VARCHAR(100)

);



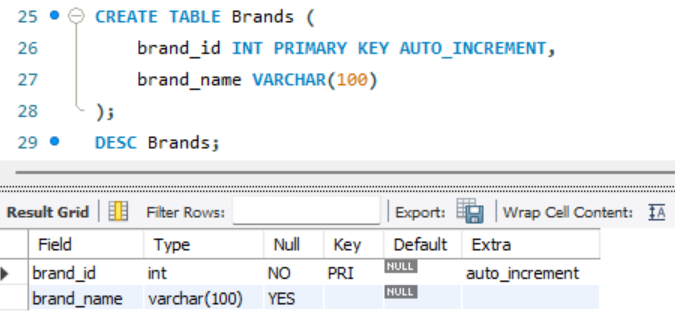
### 4. Brands Table

CREATE TABLE Brands (

brand\_id INT PRIMARY KEY AUTO\_INCREMENT,

brand\_name VARCHAR(100)

);



### 5. Orders Table

CREATE TABLE Orders (

order\_id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT,

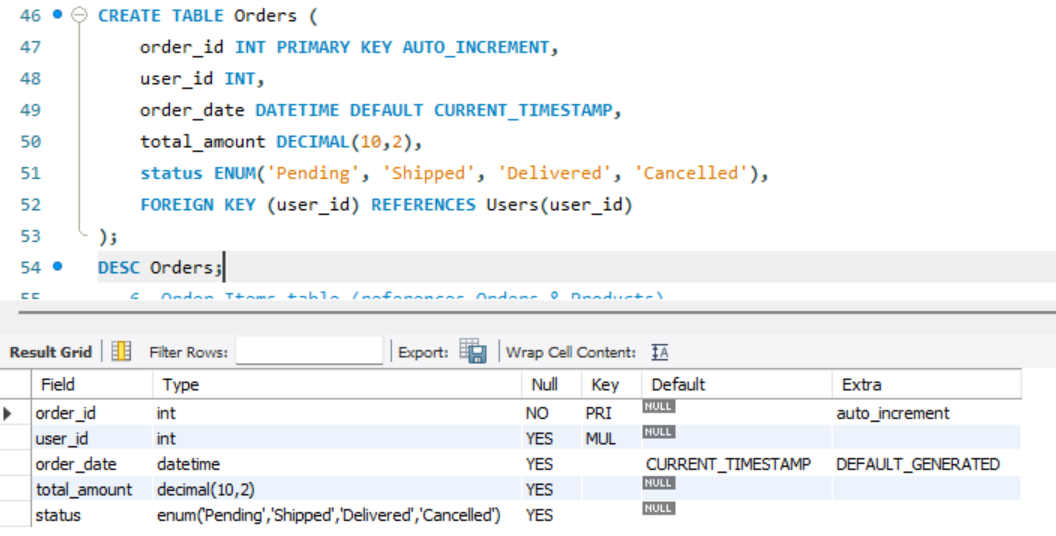
order\_date DATETIME DEFAULT CURRENT\_TIMESTAMP,

total\_amount DECIMAL(10,2),

status ENUM('Pending', 'Shipped', 'Delivered', 'Cancelled'),

FOREIGN KEY (user\_id) REFERENCES Users(user\_id)

);



### 6. Order\_Items Table

CREATE TABLE Order\_Items (

order\_item\_id INT PRIMARY KEY AUTO\_INCREMENT,

order\_id INT,

product\_id INT,

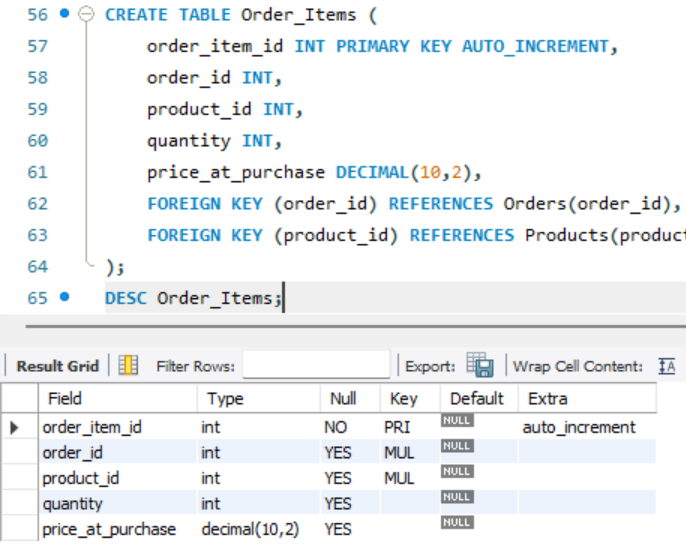
quantity INT,

price\_at\_purchase DECIMAL(10,2),

FOREIGN KEY (order\_id) REFERENCES Orders(order\_id),

FOREIGN KEY (product\_id) REFERENCES Products(product\_id)

);



### 7. Payments Table

CREATE TABLE Payments (

payment\_id INT PRIMARY KEY AUTO\_INCREMENT,

order\_id INT,

user\_id INT,

amount DECIMAL(10,2),

payment\_method ENUM('Credit Card', 'PayPal', 'Bank Transfer'),

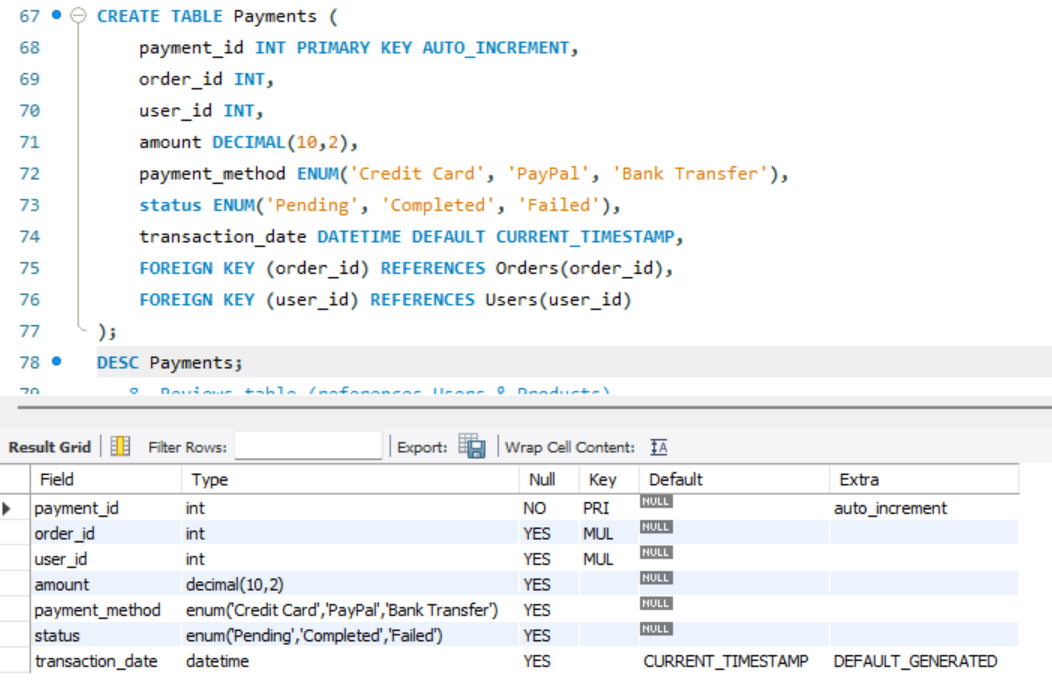
status ENUM('Pending', 'Completed', 'Failed'),

transaction\_date DATETIME DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (order\_id) REFERENCES Orders(order\_id),

FOREIGN KEY (user\_id) REFERENCES Users(user\_id)

);



### 8. Reviews Table

CREATE TABLE Reviews (

review\_id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT,

product\_id INT,

rating INT CHECK (rating >= 1 AND rating <= 5),

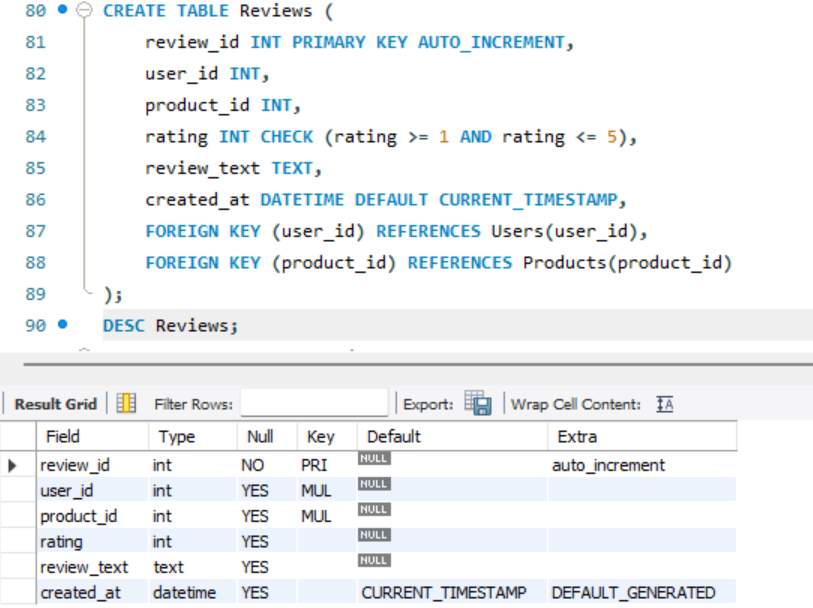
review\_text TEXT,

created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES Users(user\_id),

FOREIGN KEY (product\_id) REFERENCES Products(product\_id)

);



### 9. Inventory Table

CREATE TABLE Inventory (

inventory\_id INT PRIMARY KEY AUTO\_INCREMENT,

product\_id INT,

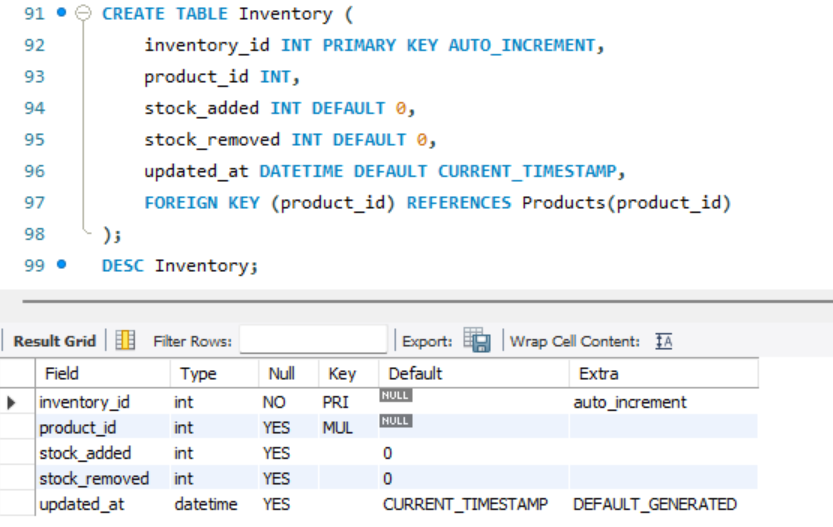
stock\_added INT DEFAULT 0,

stock\_removed INT DEFAULT 0,

updated\_at DATETIME DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (product\_id) REFERENCES Products(product\_id)

);



### 10. Wishlist Table

CREATE TABLE Wishlist (

wishlist\_id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT,

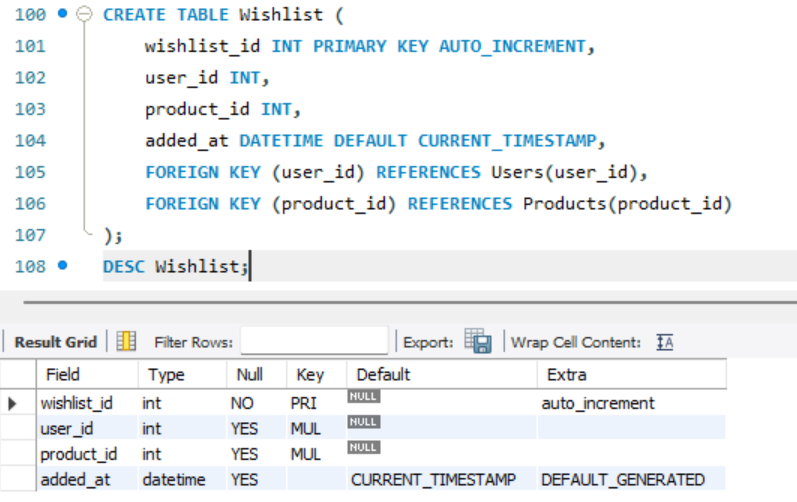
product\_id INT,

added\_at DATETIME DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES Users(user\_id),

FOREIGN KEY (product\_id) REFERENCES Products(product\_id)

);



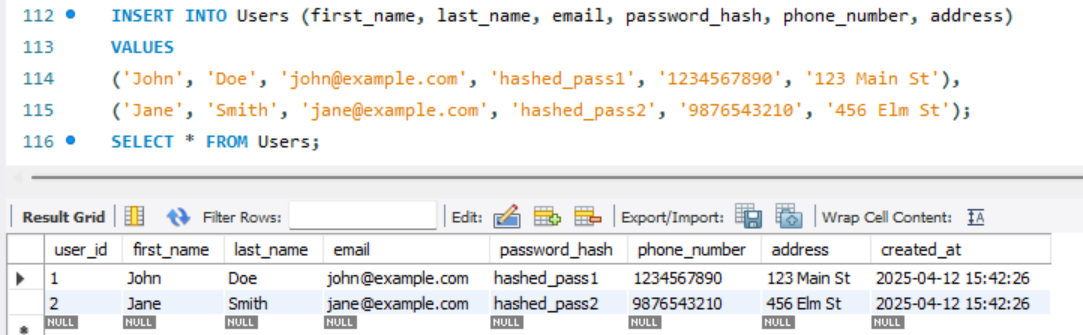
**INSERT QUERIES:**

INSERT INTO Users (first\_name, last\_name, email, password\_hash, phone\_number, address)

VALUES

('John', 'Doe', 'john@example.com', 'hashed\_pass1', '1234567890', '123 Main St'),

('Jane', 'Smith', 'jane@example.com', 'hashed\_pass2', '9876543210', '456 Elm St');



-- CATEGORIES

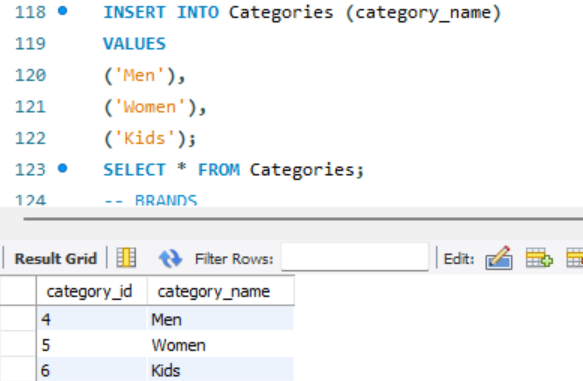
INSERT INTO Categories (category\_name)

VALUES

('Men'),

('Women'),

('Kids');



-- BRANDS

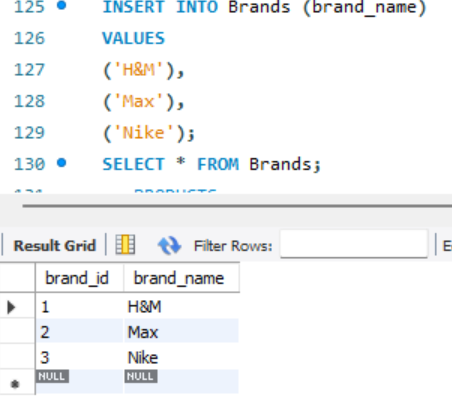
INSERT INTO Brands (brand\_name)

VALUES

('H&M'),

('Max'),

('Nike');



-- PRODUCTS

INSERT INTO Products (name, description, price, category\_id, brand\_id, stock\_quantity)

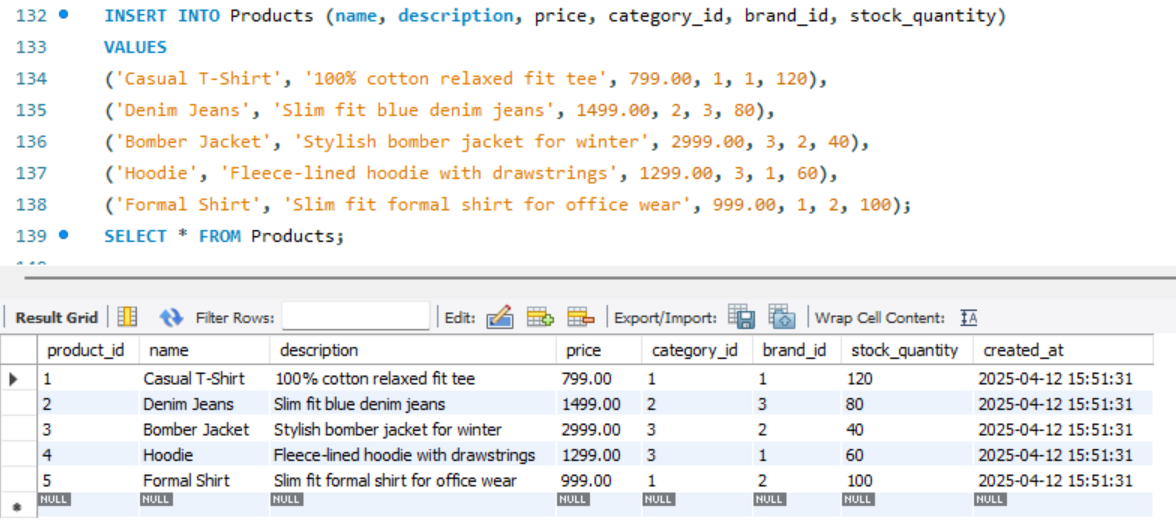
VALUES

('Casual T-Shirt', '100% cotton relaxed fit tee', 799.00, 1, 1, 120),

('Denim Jeans', 'Slim fit blue denim jeans', 1499.00, 2, 3, 80),

('Bomber Jacket', 'Stylish bomber jacket for winter', 2999.00, 3, 2, 40),

('Hoodie', 'Fleece-lined hoodie with drawstrings', 1299.00, 3, 1, 60),

('Formal Shirt', 'Slim fit formal shirt for office wear', 999.00, 1, 2, 100);

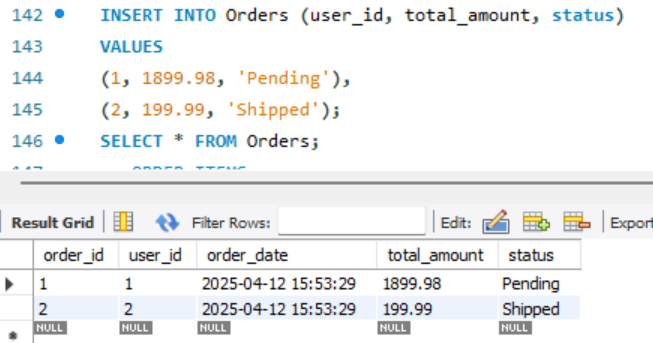
-- ORDERS

INSERT INTO Orders (user\_id, total\_amount, status)

VALUES

(1, 1899.98, 'Pending'),

(2, 199.99, 'Shipped');



-- ORDER\_ITEMS

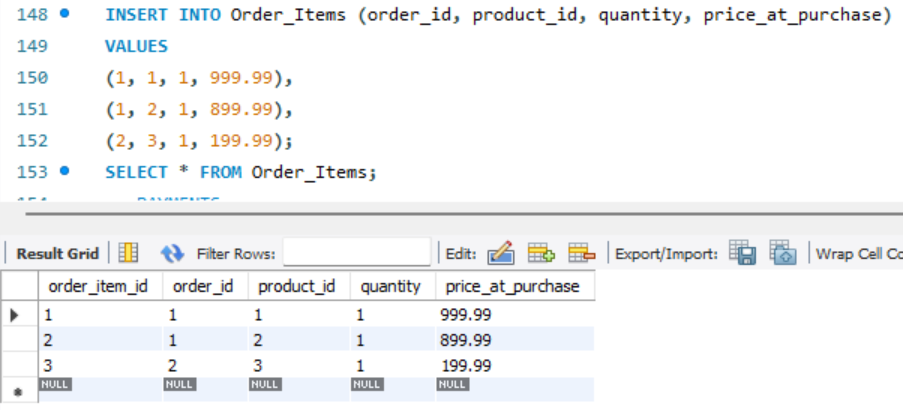
INSERT INTO Order\_Items (order\_id, product\_id, quantity, price\_at\_purchase)

VALUES

(1, 1, 1, 999.99),

(1, 2, 1, 899.99),

(2, 3, 1, 199.99);



-- PAYMENTS

INSERT INTO Payments (order\_id, user\_id, amount, payment\_method, status)

VALUES

(1, 1, 1899.98, 'Credit Card', 'Completed'),

(2, 2, 199.99, 'PayPal', 'Completed');

-- REVIEWS

INSERT INTO Reviews (user\_id, product\_id, rating, review\_text)

VALUES

(1, 1, 5, 'Amazing phone!'),

(2, 3, 4, 'Comfortable and stylish.');

-- INVENTORY

INSERT INTO Inventory (product\_id, stock\_added, stock\_removed)

VALUES

(1, 100, 50),

(2, 60, 30),

(3, 120, 20);

-- WISHLIST

INSERT INTO Wishlist (user\_id, product\_id)

VALUES

(1, 3),

(2, 1);

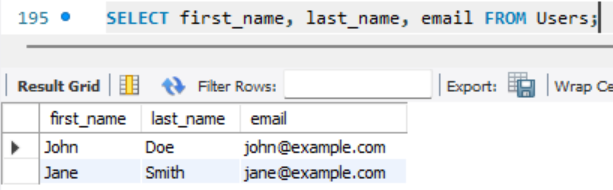
**SELECT QUERIES:**

-- 1. Retrieve all details from Users

SELECT \* FROM Users;

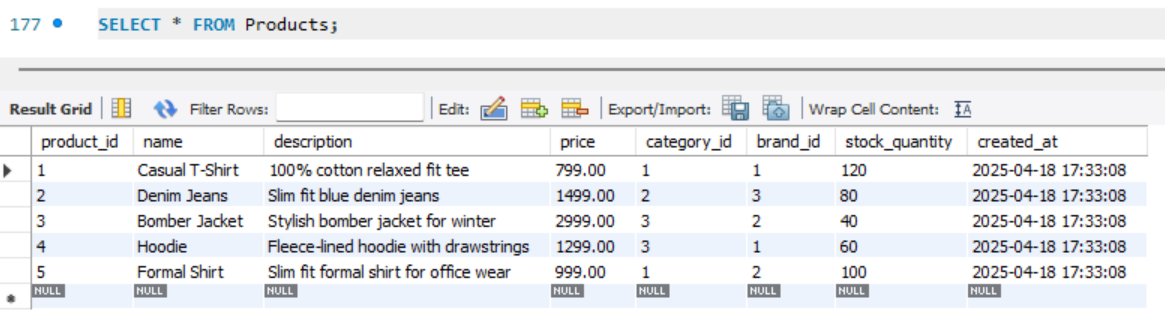
-- 2. Get specific fields (first name, last name, and email) from Users

SELECT first\_name, last\_name, email FROM Users;



-- 3. Retrieve all product details from Products

SELECT \* FROM Products;

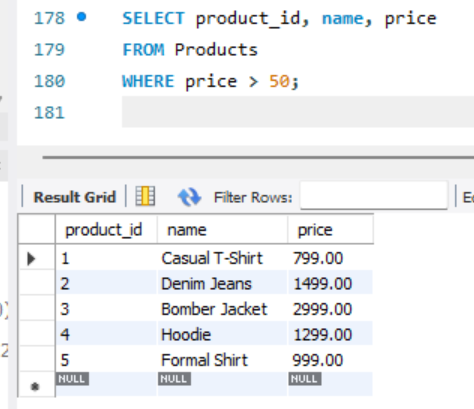


-- 4. Select products with price greater than 50

SELECT product\_id, name, price

FROM Products

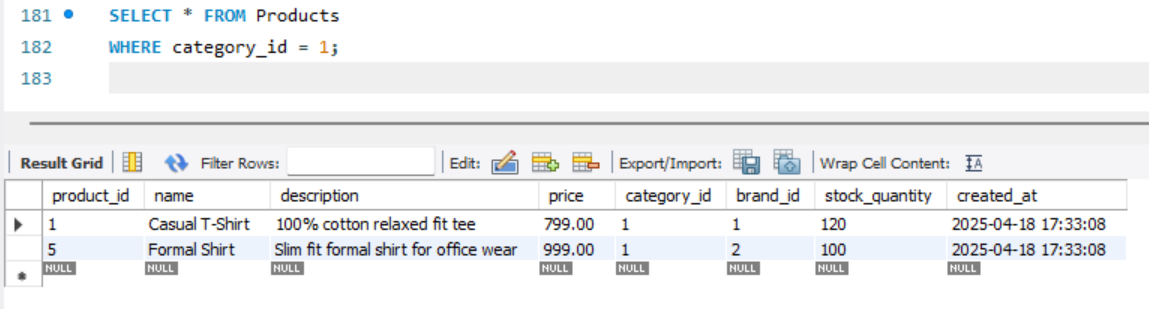
WHERE price > 50;



-- 5. Get products for a specific category (e.g., category\_id = 1)

SELECT \* FROM Products

WHERE category\_id = 1;

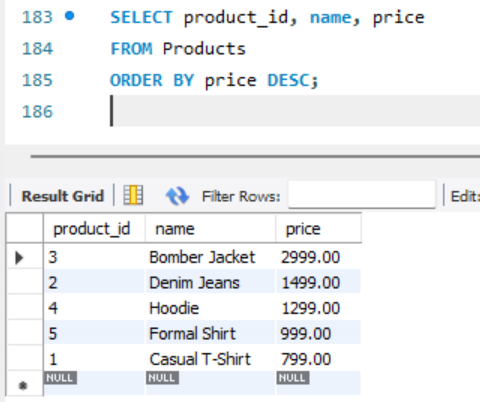


-- 6. Select products sorted by price in descending order

SELECT product\_id, name, price

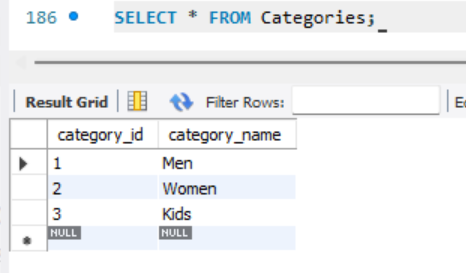
FROM Products

ORDER BY price DESC;



-- 7. Retrieve all categories

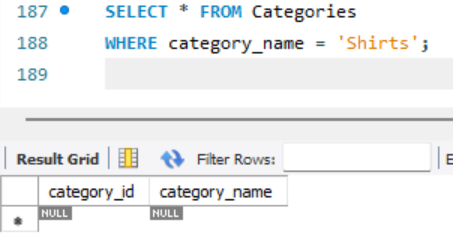
SELECT \* FROM Categories;



-- 8. Select a particular category by its name (e.g., 'Shirts')

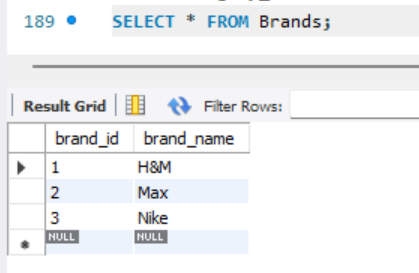
SELECT \* FROM Categories

WHERE category\_name = 'Shirts';



-- 9. Retrieve all brands

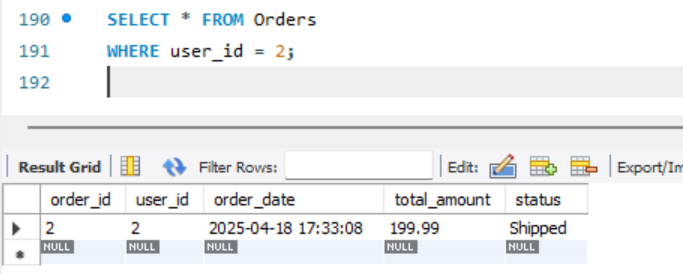
SELECT \* FROM Brands;



-- 10. Select orders placed by a specific user (e.g., user\_id = 2)

SELECT \* FROM Orders

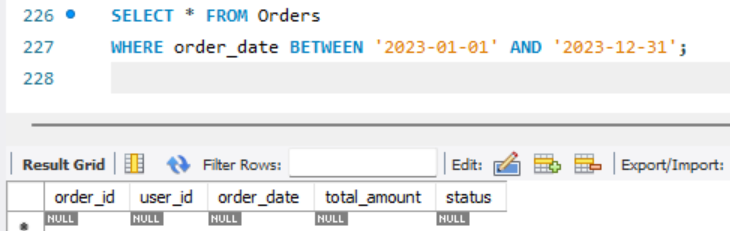
WHERE user\_id = 2;



-- 11. Retrieve orders placed within a specific date range

SELECT \* FROM Orders

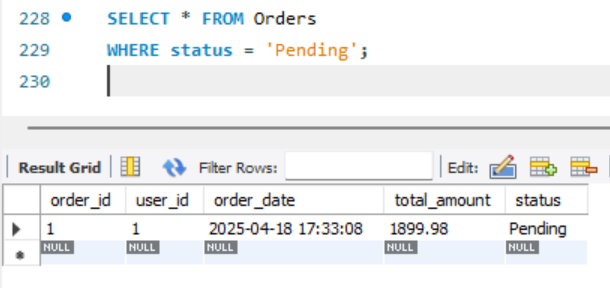
WHERE order\_date BETWEEN '2023-01-01' AND '2023-12-31';



-- 12. Select orders with a specific status, e.g., 'Pending'

SELECT \* FROM Orders

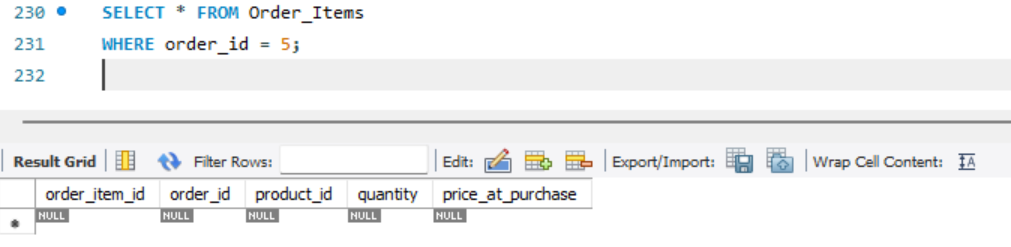
WHERE status = 'Pending';



-- 13. Retrieve all items for a given order (e.g., order\_id = 5)

SELECT \* FROM Order\_Items

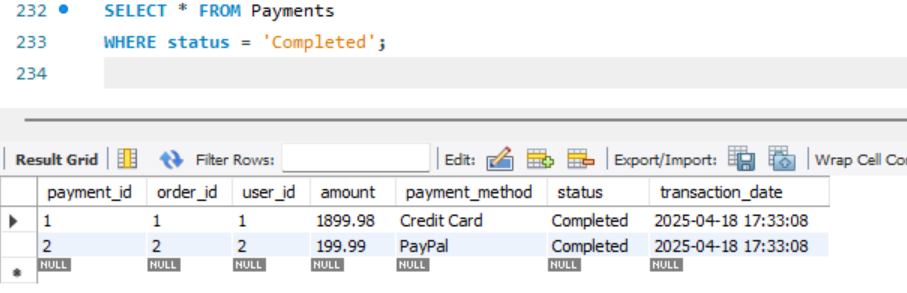
WHERE order\_id = 5;



-- 14. Select payments that have been completed

SELECT \* FROM Payments

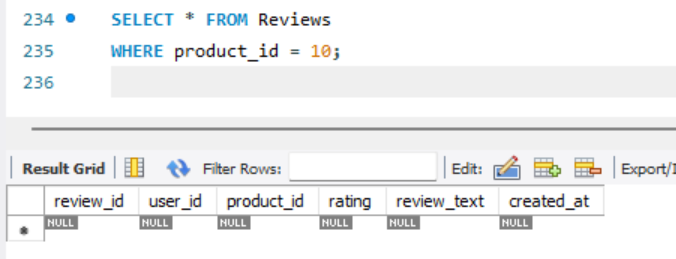
WHERE status = 'Completed';



-- 15. Get all reviews for a specific product (e.g., product\_id = 10)

SELECT \* FROM Reviews

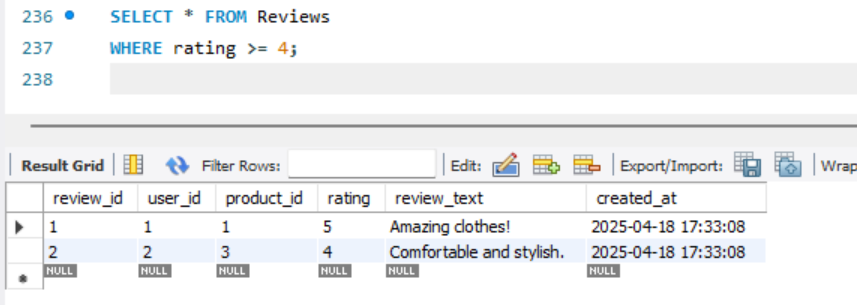
WHERE product\_id = 10;



-- 16. Select reviews with rating 4 or higher

SELECT \* FROM Reviews

WHERE rating >= 4;

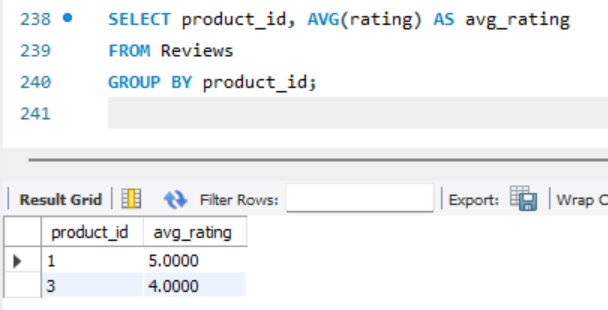


-- 17. Calculate the average rating for each product

SELECT product\_id, AVG(rating) AS avg\_rating

FROM Reviews

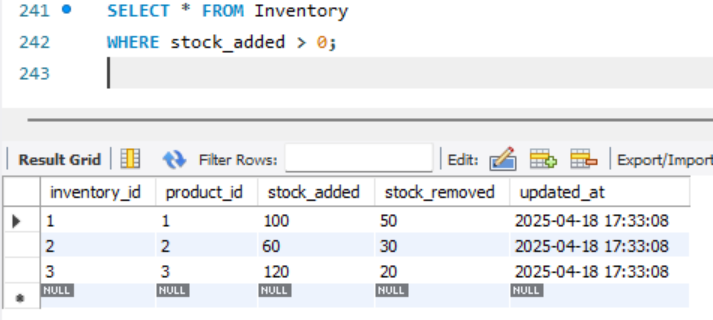
GROUP BY product\_id;



-- 18. Retrieve all inventory update logs where stock was added

SELECT \* FROM Inventory

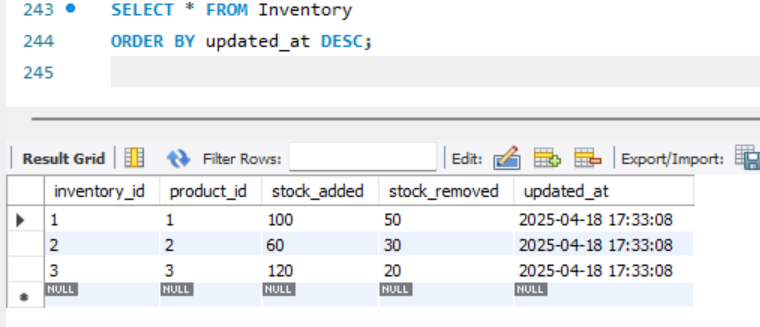
WHERE stock\_added > 0;



-- 19. Select inventory logs sorted by the most recent update

SELECT \* FROM Inventory

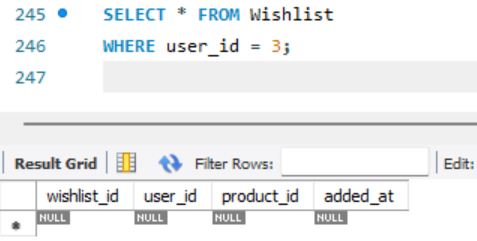
ORDER BY updated\_at DESC;



-- 20. Retrieve wishlist entries for a specific user (e.g., user\_id = 3)

SELECT \* FROM Wishlist

WHERE user\_id = 3;



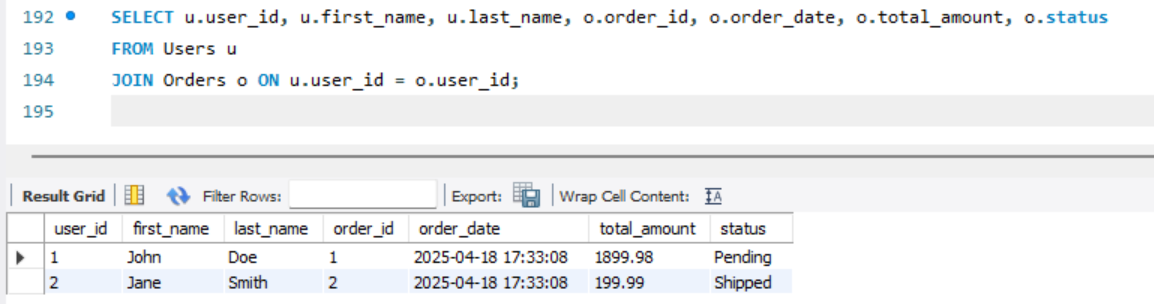
**JOIN QUERIES:**

-- 1. Users and their Orders: Retrieve each user’s orders.

SELECT u.user\_id, u.first\_name, u.last\_name, o.order\_id, o.order\_date, o.total\_amount, o.status

FROM Users u

JOIN Orders o ON u.user\_id = o.user\_id;



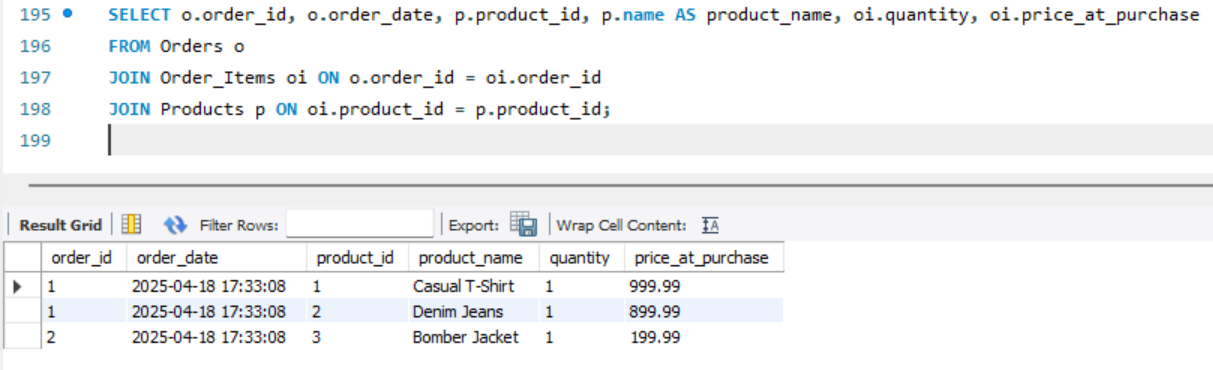
-- 2. Orders, Order\_Items, and Products: List all order items with product details.

SELECT o.order\_id, o.order\_date, p.product\_id, p.name AS product\_name, oi.quantity, oi.price\_at\_purchase

FROM Orders o

JOIN Order\_Items oi ON o.order\_id = oi.order\_id

JOIN Products p ON oi.product\_id = p.product\_id;

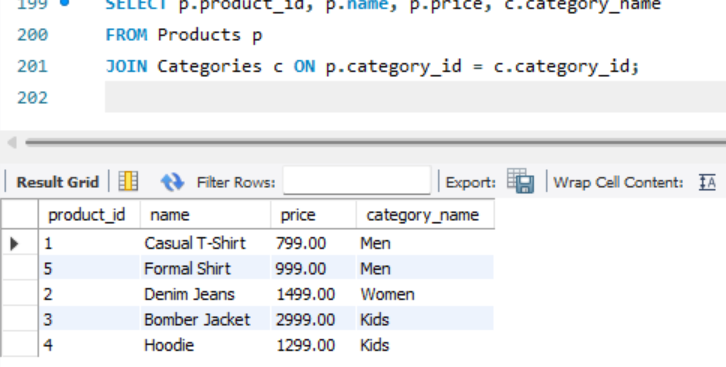


-- 3. Products with Categories: Get product details along with their category names.

SELECT p.product\_id, p.name, p.price, c.category\_name

FROM Products p

JOIN Categories c ON p.category\_id = c.category\_id;

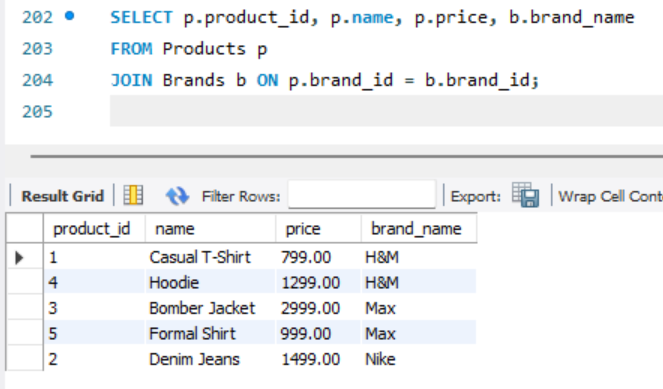


-- 4. Products with Brands: Display product details along with associated brand names.

SELECT p.product\_id, p.name, p.price, b.brand\_name

FROM Products p

JOIN Brands b ON p.brand\_id = b.brand\_id;

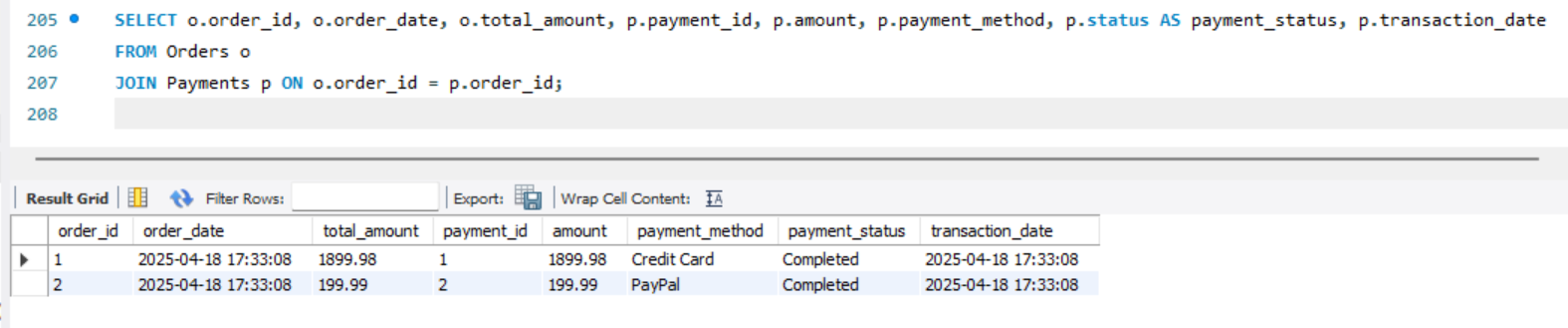


-- 5. Orders with Payments: Show payment details for each order.

SELECT o.order\_id, o.order\_date, o.total\_amount, p.payment\_id, p.amount, p.payment\_method, p.status AS payment\_status, p.transaction\_date

FROM Orders o

JOIN Payments p ON o.order\_id = p.order\_id;

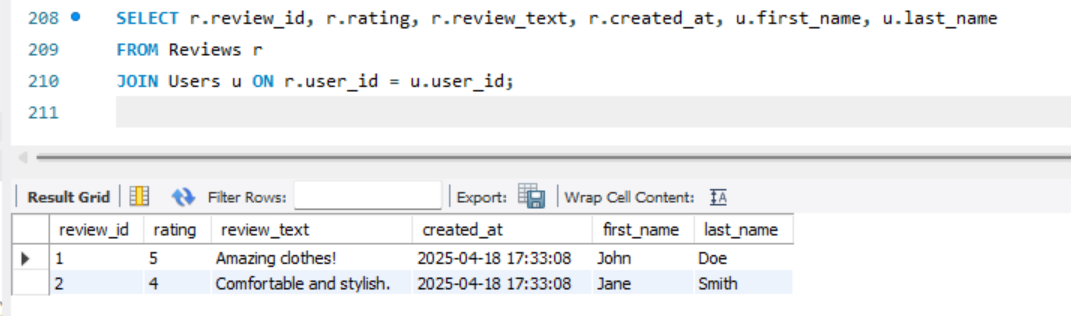


-- 6. Reviews with Users: Retrieve reviews along with reviewer details.

SELECT r.review\_id, r.rating, r.review\_text, r.created\_at, u.first\_name, u.last\_name

FROM Reviews r

JOIN Users u ON r.user\_id = u.user\_id;

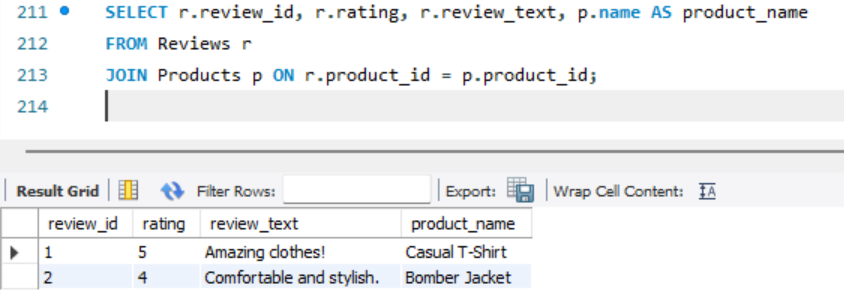


-- 7. Reviews with Products: Get reviews with product names.

SELECT r.review\_id, r.rating, r.review\_text, p.name AS product\_name

FROM Reviews r

JOIN Products p ON r.product\_id = p.product\_id;



-- 8. Orders, Order\_Items, Products, and Users: Combine order, order item, product, and user details.

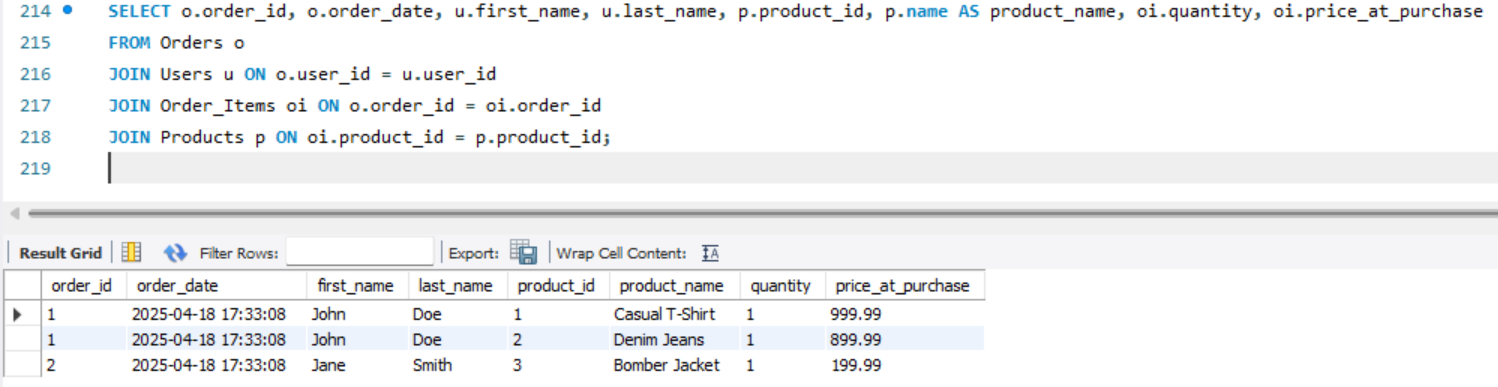
SELECT o.order\_id, o.order\_date, u.first\_name, u.last\_name, p.product\_id, p.name AS product\_name, oi.quantity, oi.price\_at\_purchase

FROM Orders o

JOIN Users u ON o.user\_id = u.user\_id

JOIN Order\_Items oi ON o.order\_id = oi.order\_id

JOIN Products p ON oi.product\_id = p.product\_id;



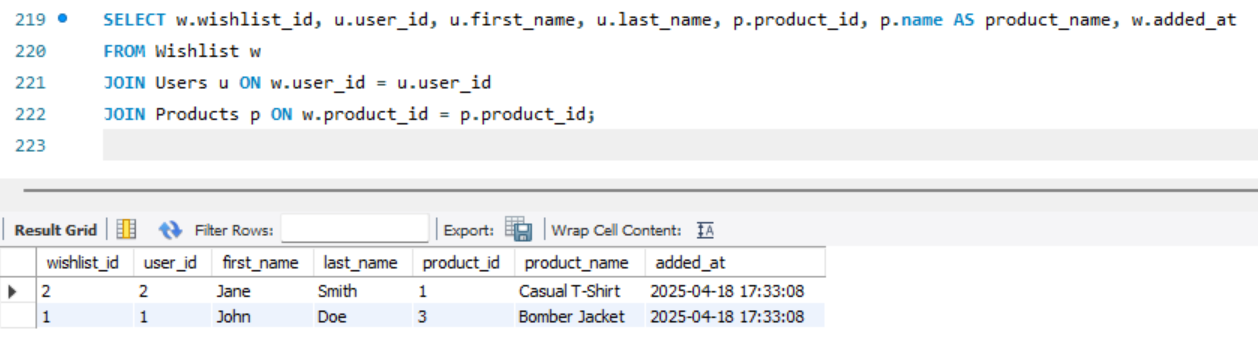
-- 9. Users’ Wishlist with Product Details: List all wishlist items for users with product details.

SELECT w.wishlist\_id, u.user\_id, u.first\_name, u.last\_name, p.product\_id, p.name AS product\_name, w.added\_at

FROM Wishlist w

JOIN Users u ON w.user\_id = u.user\_id

JOIN Products p ON w.product\_id = p.product\_id;

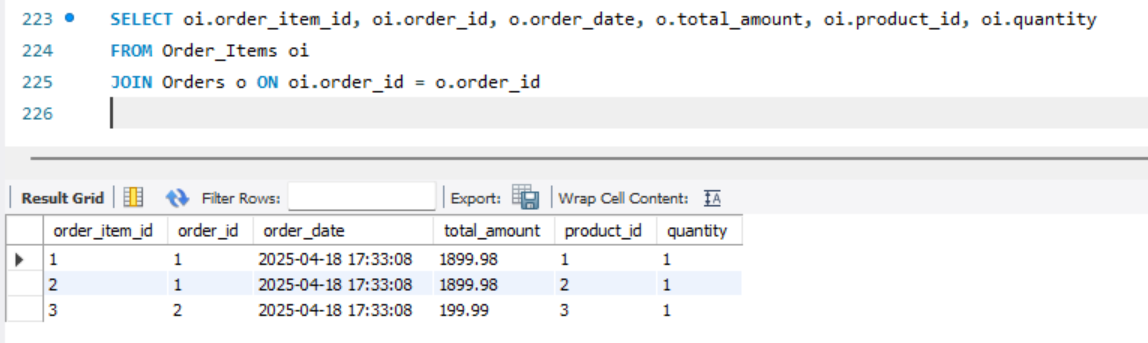


-- 10. Order\_Items with their Orders: Get order items with order date and total amount.

SELECT oi.order\_item\_id, oi.order\_id, o.order\_date, o.total\_amount, oi.product\_id, oi.quantity

FROM Order\_Items oi

JOIN Orders o ON oi.order\_id = o.order\_id



### ****VI. Project Demonstration****

**Tools/Software Used:**

* **MySQL** – Used for designing, creating, normalizing, and managing the relational database for the e-commerce system.
* **Power BI** – Utilized to create dashboards and visualizations for analyzing sales, customer trends, product reviews, and inventory updates.
* **PHP & HTML/CSS/JavaScript** – Used to develop the interactive front-end interface for user registration, product browsing, ordering, and reviewing.
* **XAMPP** – Deployed as a local server to host the web application and connect PHP with MySQL during testing and development.

**Demonstration Description:** The **E-Commerce Clothing Store Database** was developed to manage products, users, orders, payments, and reviews. Key features of the demonstration include:

* **Table Creation:** Designed and normalized tables like Users, Products, Orders, Order\_Items, Payments, Reviews, etc., with appropriate primary and foreign key relationships.
* **Data Insertion:** Inserted realistic sample data to simulate store operations such as product listings, customer purchases, and feedback.
* **Query Execution:** Demonstrated complex SQL queries involving JOIN, aggregate functions, and filters for insights like top-selling products, customer order history, and product stock levels.
* **Front-End Integration:** Integrated a PHP-based front end with the MySQL database for live data entry, display, and updates.
* **Interactive Features:** Implemented dynamic dropdowns for categories and brands, and filter options for products, enhancing user experience during shopping and data input.

### ****VII. Self-Learning Beyond Classroom****

* Gained practical hands-on experience designing a full-stack database-driven system from scratch.
* Learned and applied normalization concepts from 1NF to BCNF for real-world e-commerce data.
* Explored server-side scripting with PHP and database interaction using MySQLi.
* Learned how to manage SQL errors and foreign key issues through debugging during insertion and updates.
* Experimented with UI elements like product cards, responsive layouts, dark mode, and dropdowns for a smoother UX.

### ****VIII. Learning from the Project****

* Developed a strong understanding of **relational database design**, normalization, and indexing for performance.
* Understood the relationship between tables like users–orders–payments and how foreign keys enforce data integrity.
* Improved ability to write optimized SQL queries for business reporting and user interaction.
* Gained insight into front-end to back-end interaction through real-time order placement and product search.
* Implemented login and session handling to differentiate between customer and admin functionality.

### ****IX. Challenges Faced****

* Initial difficulty in **normalizing product and order data** without losing flexibility in category-brand combinations.
* Encountered **foreign key constraint issues** due to insert order and missing references in sample data.
* Struggled with **JOIN queries** when fetching multi-level data such as user orders and product reviews.
* Faced challenges styling the front-end with CSS while ensuring responsiveness and usability.
* Had to refactor several PHP scripts to handle cart logic, inventory management, and payment validation.

### ****X. Conclusion****

* This project provided an in-depth learning experience by combining theoretical knowledge with practical implementation.
* Strengthened my understanding of **MySQL database design, normalization, and SQL querying**.
* Helped me understand how full-stack e-commerce platforms work—from user login to placing an order and giving reviews.
* Built a strong foundation in **web development, database management**, and **data visualization**.
* Inspired confidence in handling larger data-driven projects in future academic or professional work.