**E-COMMERCE WEBSITE**

**ABSTRACT**

The e-commerce website project is a simplified platform designed to simulate the basic functionalities of an online shopping system. It focuses on providing a user-friendly interface for browsing products, managing a shopping cart and placing orders, etc. The website includes essential features such as user registration and login, Product listing by categories, and a wish list for the users to add items which are not available at the moment.

The database is designed with 9 tables to manage critical aspects of the system, including user data, product details, categories, orders, payments, inventory, cart, reviews and wish list for customers. For example, the products table stores information about available items, while the users table keeps track of customer login credentials. The orders table records placed orders, and the inventory table ensures accurate stock updates.

This project highlights the integration of a basic web interface with a structured database, offering a foundation for understanding e-commerce systems. It is intended as a learning project to explore database design, web development basics, and the flow of data in an online store.

**ER DIAGRAM**

**Entities and Attributes: -**

1)User: - { user\_id, ph\_no, address, name, date of birth, age, password}

2)Products: - {Product\_id, Product name, Price, Quantity}

3) Cart: - {Product\_id, Product name, cart total}

4)Orders: - {Order\_id, Price, quantity, product name, order date}

5)Supplier: - {Supplier\_id, product\_id, product name, Contact info, Supplier Address}

6)Category: - {Category\_id, Category name, description}

7) Wish List: - { Product\_id, Product name, Quantity}

8)Reviews: - {review\_id, product\_id, user\_id, rating}

9)Payments: - {Payment\_id, Amount, Payment methods, order\_id}

A diagram of a diagram

AI-generated content may be incorrect.

1. User: -

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| User\_id | First name | Middle name | Last name | Date of birth | Age | Ph no | Address | Password |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

1. Products: -

|  |  |  |  |
| --- | --- | --- | --- |
| Product\_id | Product name | quantity | price |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. Cart: -

|  |  |  |
| --- | --- | --- |
| Product\_id | Product name | Cart total |
|  |  |  |
|  |  |  |
|  |  |  |

1. Orders: -

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Order\_id | Price | quantity | Product name | Order date |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

1. Supplier: -

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Supplier\_id | Product\_id | Product name | Contact info | Supplier address |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

1. Category: -

|  |  |  |
| --- | --- | --- |
| Category\_id | Category name | Discription |
|  |  |  |
|  |  |  |
|  |  |  |

1. Wish List: -

|  |  |  |
| --- | --- | --- |
| Product\_id | Product name | quantity |
|  |  |  |
|  |  |  |
|  |  |  |

1. Review: -

|  |  |  |  |
| --- | --- | --- | --- |
| Review\_id | User\_id | Product name | Rating |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. Payments: -

|  |  |  |  |
| --- | --- | --- | --- |
| Payment\_id | Payment methods | Order\_id | Amount |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Relational Algebra: -**

**Table 1: User (User\_ID, Name, Phone\_No, Address, DOB, Age, Password)**

**Retrieve all users who live in "New York"**

σAddress=′NewYork′​(User)

**Find all users and suppliers who are from "New York."**

πUser\_ID,Name,Address​(σAddress=′NewYork′​(User)) ∪ πSupplier\_ID,Name,Address​(σAddress=′NewYork′​(Supplier))

**Find users who have the same age**

σAge>30​(User)

**List all user IDs and their names**

πUser\_ID,Name​(User)

**Find users who have the same age**

ρU1​(User)⋈U1.Age=U2.Age​ρU2​(User)

**Find all users who have added a product to their cart and also ordered the same product**

πUser\_ID,Product\_ID,Product\_Name​(Orders) ∩ πUser\_ID,Product\_ID,Product\_Name​(Cart)

**Table 2: Products (Product\_ID, Product\_Name, Price, Quantity)**

**Retrieve all products that cost more than $1000**

σPrice>1000​(Products)

**Find products with the lowest price**

σPrice=MIN(Price)​(Products)

**Find the average price of all products**

gAVG(Price)​(Products)

**Find all products that are either in the cart or have been ordered.**

πProduct\_ID,Product\_Name​(Cart) ∪ πProduct\_ID,Product\_Name​(Orders)

**Table 3: Cart (Product\_ID, Product\_Name, Cart\_Total)**

**Retrieve all products in the cart where total value exceeds $5000**

σCart\_Total>5000​(Cart)

**List all product names from the cart**

πProduct\_Name​(Cart)

**Find total cart value for all users**

gSUM(Cart\_Total)​(Cart)

**Table 4: Orders (Order\_ID, Price, Quantity, Product\_Name, Order\_Date)**

**Retrieve all orders placed after '2024-01-01'**

σOrder\_Date>′2024−01−01′​(Orders)

**Find total revenue from orders**

gSUM(Price×Quantity)​(Orders)

**Find users who have not placed any orders**

πUser\_ID​(User)−πUser\_ID​(Orders)

**Table 5: Supplier (Supplier\_ID, Product\_ID, Product\_Name, Contact\_Info, Supplier\_Address)**

**Retrieve all suppliers located in 'California'**

σSupplier\_Address=′California′​(Supplier)

**Find suppliers that supply more than 5 products**

gSupplier\_ID,COUNT(Product\_ID)>5​(Supplier)

**Table 6: Category (Category\_ID, Category\_Name, Description)**

**Retrieve all categories that contain 'Electronics’**

σCategory\_=′Electronics′​(Category)

**List all category names**

πCategory\_Name​(Category)

**Table 7: Wish List (User\_ID, Product\_ID, Product\_Name, Quantity)**

**Retrieve all products in the wishlist of user ID = 101**

σRating>4​(Reviews)

**List all unique product names in wishlists**

πProduct\_Name​(WishList)

**Table 8: Reviews (Review\_ID, Product\_ID, User\_ID, Rating)**

**Retrieve all reviews where the rating is greater than 4**

σRating>4​(Reviews)

**Find the highest-rated product**

σRating=MAX(Rating)​(Reviews)

**Table 9: Payments (Payment\_ID, Amount, Payment\_Methods, Order\_ID)**

**Retrieve all payments made using "Credit Card"**

σPayment\_Methods=′CreditCard′​(Payments)

**Find the total amount paid for all orders**

gSUM(Amount)​(Payments)

**Tuple Relational Calculus**

**Table 1: User**

**Retrieve all users who live in "New York"**

{U∣U∈User∧U.Address=′NewYork′}

**Find users older than 30 years**

{U∣U∈User∧U.Age>30}

**Table 2: Products**

**Retrieve all products that cost more than $1000**

{P∣P∈Products∧P.Price>1000}

**Find products that are out of stock**

{P∣P∈Products∧P.Quantity=0}

**Table 3: Cart**

**Retrieve all products in the cart where total value exceeds $5000**

σCart\_Total>5000​(Cart)

**Table 4: Orders**

**Retrieve all orders placed after '2024-01-01'**

{O∣O∈Orders∧O.Order\_Date>′2024−01−01′}

**Find users who have placed at least one order**

{U∣U∈User∧∃O∈Orders(U.User\_ID=O.User\_ID)}

**Table 5: Supplier**

**Retrieve all suppliers located in 'California'**

{S∣S∈Supplier∧S.Supplier\_Address=′California′}

**Table 6: Category**

**Retrieve all categories that contain 'Electronics'**

{C∣C∈Category∧C.Category\_Name=′Electronics′}

**Table 7: Wish List**

**Retrieve all wishlist items for user ID 101**

{W∣W∈WishList∧W.User\_ID=101}

**Table 8: Reviews**

**Retrieve all reviews where the rating is greater than 4**

{R∣R∈Reviews∧R.Rating>4}

**Table 9: Payments**

**Retrieve all payments made using "Credit Card"**

{P∣P∈Payments∧P.Payment\_Methods=′CreditCard′}

**Domain Relational Calculus**

**Table 1: User (User\_ID, Name, Phone\_No, Address, DOB, Age, Password)**

**Retrieve all users who live in "New York"**

{⟨UI​D,N,P,A,D,Ag,Pw⟩∣∃U(U∈User∧A=′NewYork′)}

**Find users who are older than every other user**

{⟨UI​D,N⟩∣UI​D∈User∧∀U2​∈User(UI​D!=U2​⇒Age>U2​.Age)}

**Table 2: Products (Product\_ID, Product\_Name, Price, Quantity)**

**Retrieve all products that cost more than $1000**

{⟨PI​D,PN​ame,Pr,Q⟩∣PI​D∈Products∧Pr>1000}

**Find products that are out of stock**

{⟨PI​D,PN​ame⟩∣PI​D∈Products∧Q=0}

**Table 3: Cart (Product\_ID, Product\_Name, Cart\_Total)**

**Retrieve all cart items where total value exceeds $5000**

{⟨PI​D,PN​ame,CT​⟩∣PI​D∈Cart∧CT​>5000}

**Find users who have at least one item in the cart but have never ordered that item**

{⟨UI​D⟩∣∃C(C∈Cart∧C.UI​D=UI​D)∧¬∃O(O∈Orders∧O.PI​D=C.PI​D)}

**Table 4: Orders (Order\_ID, Price, Quantity, Product\_Name, Order\_Date)**

**Retrieve all orders placed after '2024-01-01'**

{⟨OI​D,Pr,Q,PN​ame,OD​ate⟩∣OI​D∈Orders∧OD​ate>′2024−01−01′}

**Find users who have placed at least one order**

{⟨UI​D⟩∣∃O(O∈Orders∧O.UI​D=UI​D)}

**Table 5: Supplier (Supplier\_ID, Product\_ID, Product\_Name, Contact\_Info, Supplier\_Address)**

**Retrieve all suppliers located in 'California'**

{⟨SI​D,PI​D,PN​ame,CI​nfo,SA​ddr⟩∣SI​D∈Supplier∧SA​ddr=′California′}

**Find suppliers who supply at least one product costing more than $500**

{⟨SI​D⟩∣∃P(P∈Products∧P.PI​D=SI​D∧P.Price>500)}

**Table 6: Category (Category\_ID, Category\_Name, Description)**

**Retrieve all categories that contain 'Electronics'**

{⟨CI​D,CN​ame,Desc⟩∣CI​D∈Category∧CN​ame=′Electronics′}

**Find categories that have no products assigned**

{⟨CI​D⟩∣CI​D∈Category∧¬∃P(P∈Products∧P.CI​D=CI​D)}

**Table 7: Wish List (User\_ID, Product\_ID, Product\_Name, Quantity)**

**Retrieve all wishlist items for user ID 101**

{⟨PI​D,PN​ame,Q⟩∣PI​D∈WishList∧UI​D=101}

**Find users who have the same wishlist items**

{⟨U1​,U2​⟩∣∃W1​,W2​(W1​∈WishList∧W2​∈WishList∧W1​.PI​D=W2​.PI​D∧W1​.UI​D!=W2​.UI​D)}

**Table 8: Reviews (Review\_ID, Product\_ID, User\_ID, Rating)**

**Retrieve all reviews where the rating is greater than 4**

{⟨RI​D,PI​D,UI​D,R⟩∣RI​D∈Reviews∧R>4}

**Find products that have only been rated 5 stars**

{⟨PI​D⟩∣∀R(R∈Reviews∧R.PI​D=PI​D⇒R.Rating=5)}

**Table 9: Payments (Payment\_ID, Amount, Payment\_Methods, Order\_ID)**

**Retrieve all payments made using "Credit Card"**

{⟨PI​D,A,PM,OI​D⟩∣PI​D∈Payments∧PM=′CreditCard′}

**Find orders that have not yet been paid for**

{⟨OI​D⟩∣OI​D∈Orders∧¬∃P(P∈Payments∧P.OI​D=OI​D)}

**DDL (Data Definition Language) Queries:**

1. Create commands

CREATE TABLE user (

userid INT PRIMARY KEY AUTO\_INCREMENT,

first\_name VARCHAR(50),

middle\_name VARCHAR(50),

last\_name VARCHAR(50),

phoneno VARCHAR(15), -- Assuming it's stored as a string

date\_of\_birth DATE,

age INT, -- This will be manually calculated

address VARCHAR(255)

);



CREATE TABLE category (

category\_id INT PRIMARY KEY AUTO\_INCREMENT,

category\_name VARCHAR(100),

description TEXT

);

A black screen with white text

AI-generated content may be incorrect.

CREATE TABLE product (

product\_id INT PRIMARY KEY AUTO\_INCREMENT,

product\_name VARCHAR(100),

quantity INT,

price DECIMAL(10, 2),

category\_id INT,

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

);

A computer screen with white text

AI-generated content may be incorrect.

CREATE TABLE orders (

order\_id INT PRIMARY KEY AUTO\_INCREMENT,

price DECIMAL(10, 2),

order\_date DATE,

product\_name VARCHAR(100),

quantity INT,

user\_id INT,

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

);

A screen shot of a computer code

AI-generated content may be incorrect.

CREATE TABLE payments (

payment\_id INT PRIMARY KEY AUTO\_INCREMENT,

payment\_method VARCHAR(50),

amount DECIMAL(10, 2),

order\_id INT,

FOREIGN KEY (order\_id) REFERENCES orders(order\_id)

);

A black screen with white text

AI-generated content may be incorrect.

CREATE TABLE reviews (

review\_id INT PRIMARY KEY AUTO\_INCREMENT,

product\_name VARCHAR(100),

user\_id INT,

rating INT CHECK (rating BETWEEN 1 AND 5),

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

);

A black screen with white text

AI-generated content may be incorrect.

CREATE TABLE cart (

cart\_id INT PRIMARY KEY AUTO\_INCREMENT,

cart\_total DECIMAL(10, 2),

product\_name VARCHAR(100),

product\_id INT,

user\_id INT,

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

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

);

A screen shot of a computer code

AI-generated content may be incorrect.

CREATE TABLE supplier (

supplier\_id INT PRIMARY KEY AUTO\_INCREMENT,

contact\_info VARCHAR(100),

product\_id INT,

product\_name VARCHAR(100),

supplier\_address VARCHAR(255),

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

);

A screen shot of a computer program

AI-generated content may be incorrect.

CREATE TABLE wishlist (

wishlist\_id INT PRIMARY KEY AUTO\_INCREMENT,

product\_id INT,

product\_name VARCHAR(100),

quantity\_needed INT,

user\_id INT,

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

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

);

A screen shot of a computer program

AI-generated content may be incorrect.

1. INSERT INTO commands:

INSERT INTO user (first\_name, middle\_name, last\_name, phoneno, date\_of\_birth, age, address)

VALUES

('John', 'Doe', 'Smith', '1234567890', '1995-05-15', 30, '123 Main St, City, Country'),

('Alice', 'Marie', 'Johnson', '2345678901', '1987-07-22', 38, '456 Oak St, City, Country'),

('Bob', 'L', 'Williams', '3456789012', '2000-02-11', 25, '789 Pine St, City, Country'),

('Eve', 'S', 'Brown', '4567890123', '1990-11-09', 35, '101 Maple St, City, Country'),

('Charlie', 'B', 'Davis', '5678901234', '1992-03-29', 33, '202 Birch St, City, Country');

A computer screen with numbers and numbers

AI-generated content may be incorrect.

INSERT INTO category (category\_name, description)

VALUES

('Electronics', 'Devices and gadgets like phones, laptops, etc.'),

('Clothing', 'Apparel for men, women, and children'),

('Home Appliances', 'Electrical appliances for home use'),

('Sports', 'Sports equipment and accessories'),

('Books', 'Physical and e-books for all genres');

A computer screen with white text

AI-generated content may be incorrect.

INSERT INTO product (product\_name, quantity, price, category\_id)

VALUES

('Smartphone', 100, 499.99, 1),

('T-shirt', 200, 19.99, 2),

('Washing Machine', 50, 299.99, 3),

('Soccer Ball', 150, 25.99, 4),

('Java Programming Book', 300, 29.99, 5);

A screen shot of a computer

AI-generated content may be incorrect.

INSERT INTO orders (price, order\_date, product\_name, quantity, user\_id)

VALUES

(499.99, '2025-03-01', 'Smartphone', 1, 1),

(19.99, '2025-03-02', 'T-shirt', 2, 2),

(299.99, '2025-03-03', 'Washing Machine', 1, 3),

(25.99, '2025-03-04', 'Soccer Ball', 3, 4),

(29.99, '2025-03-05', 'Java Programming Book', 5, 5);

A computer screen with white text

AI-generated content may be incorrect.

INSERT INTO payments (payment\_method, amount, order\_id)

VALUES

('Credit Card', 499.99, 1),

('PayPal', 39.98, 2),

('Bank Transfer', 299.99, 3),

('Debit Card', 77.97, 4),

('Credit Card', 149.95, 5);

A screenshot of a computer screen

AI-generated content may be incorrect.

INSERT INTO reviews (product\_name, user\_id, rating)

VALUES

('Smartphone', 1, 5),

('T-shirt', 2, 4),

('Washing Machine', 3, 5),

('Soccer Ball', 4, 3),

('Java Programming Book', 5, 5);

A screenshot of a computer screen

AI-generated content may be incorrect.

INSERT INTO cart (cart\_total, product\_name, product\_id, user\_id)

VALUES

(499.99, 'Smartphone', 1, 1),

(39.98, 'T-shirt', 2, 2),

(299.99, 'Washing Machine', 3, 3),

(77.97, 'Soccer Ball', 4, 4),

(149.95, 'Java Programming Book', 5, 5);

A screen shot of a computer

AI-generated content may be incorrect.

INSERT INTO supplier (contact\_info, product\_id, product\_name, supplier\_address)

VALUES

('supplier1@electronics.com', 1, 'Smartphone', '123 Electronics St'),

('supplier2@clothing.com', 2, 'T-shirt', '456 Apparel Ave'),

('supplier3@appliances.com', 3, 'Washing Machine', '789 Appliance Rd'),

('supplier4@sports.com', 4, 'Soccer Ball', '101 Sports Blvd'),

('supplier5@books.com', 5, 'Java Programming Book', '202 Book Lane');

A screen shot of a computer

AI-generated content may be incorrect.

INSERT INTO wishlist (product\_id, product\_name, quantity\_needed, user\_id)

VALUES

(1, 'Smartphone', 1, 1),

(2, 'T-shirt', 2, 2),

(3, 'Washing Machine', 1, 3),

(4, 'Soccer Ball', 3, 4),

(5, 'Java Programming Book', 5, 5);

A screen shot of a computer

AI-generated content may be incorrect.

1. DELETE command

DELETE FROM wishlist WHERE user\_id = 1 AND product\_id = 1;



DELETE FROM cart WHERE user\_id = 2 AND product\_id = 2;



DELETE FROM reviews WHERE review\_id = 1;



DELETE FROM payments WHERE payment\_id = 2;



DELETE FROM reviews WHERE review\_id = 2;



DELETE FROM wishlist WHERE user\_id = 1;



DELETE FROM cart WHERE user\_id = 1;



DELETE FROM reviews WHERE user\_id = 1;



DELETE FROM orders WHERE user\_id = 1;



DELETE FROM payments WHERE order\_id IN (SELECT order\_id FROM orders WHERE user\_id = 1);



1. DROP command

CREATE TABLE customer\_feedback (

feedback\_id INT PRIMARY KEY AUTO\_INCREMENT,

product\_id INT,

user\_id INT,

rating INT,

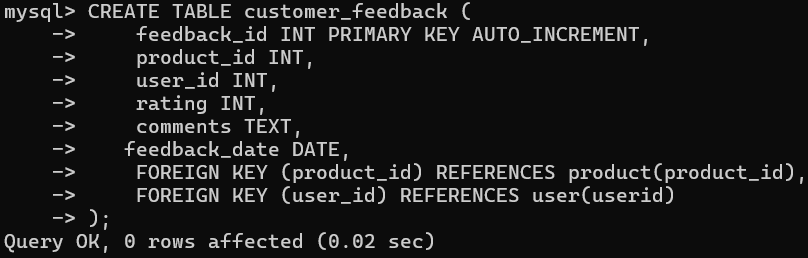
comments TEXT,

feedback\_date DATE,

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

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

);

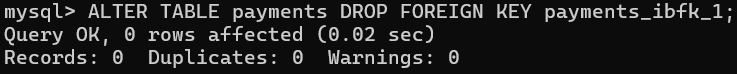


DROP TABLE customer\_feedback;

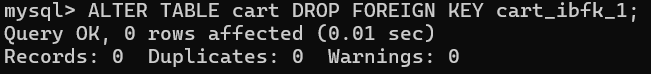


1. ALTER TABLE DROP

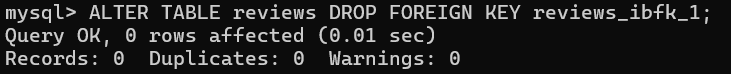
ALTER TABLE payments DROP FOREIGN KEY payments\_ibfk\_1;



ALTER TABLE cart DROP FOREIGN KEY cart\_ibfk\_1;

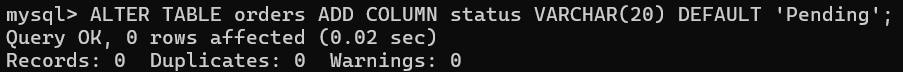


ALTER TABLE reviews DROP FOREIGN KEY reviews\_ibfk\_1;

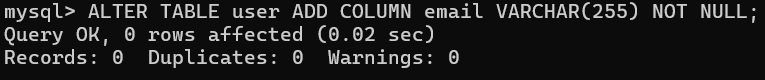


1. ALTER TABLE ADD:

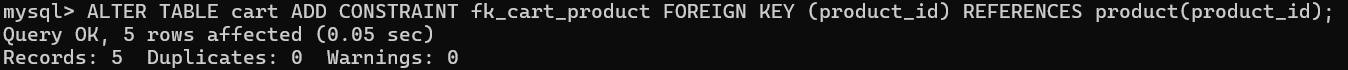
ALTER TABLE orders ADD COLUMN status VARCHAR(20) DEFAULT 'Pending';



ALTER TABLE user ADD COLUMN email VARCHAR(255) NOT NULL;

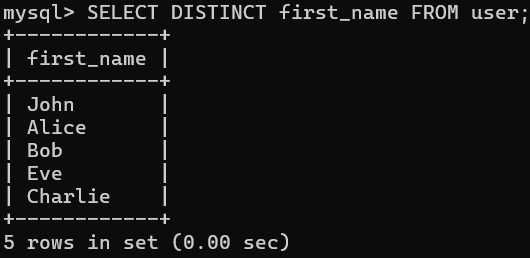


ALTER TABLE cart ADD CONSTRAINT fk\_cart\_product FOREIGN KEY (product\_id) REFERENCES product(product\_id);



1. Select Query for Single Table (including DISTINCT)

SELECT DISTINCT first\_name FROM user;

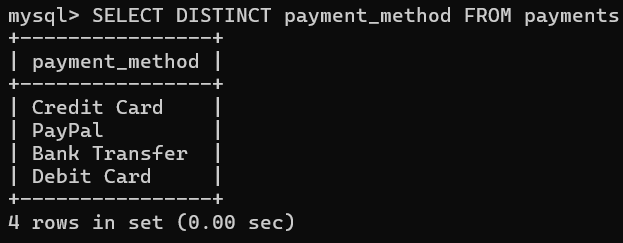


SELECT DISTINCT product\_name FROM product;

A screen shot of a computer

AI-generated content may be incorrect.

SELECT DISTINCT payment\_method FROM payments;



SELECT DISTINCT order\_id FROM orders;

A black screen with white text

AI-generated content may be incorrect.

SELECT DISTINCT review\_id FROM reviews;

A black screen with white text

AI-generated content may be incorrect.

SELECT DISTINCT cart\_total FROM cart;

A black screen with white text

AI-generated content may be incorrect.

SELECT DISTINCT category\_name FROM category;

A screen shot of a computer

AI-generated content may be incorrect.

SELECT DISTINCT supplier\_id FROM supplier;

A black screen with white text

AI-generated content may be incorrect.

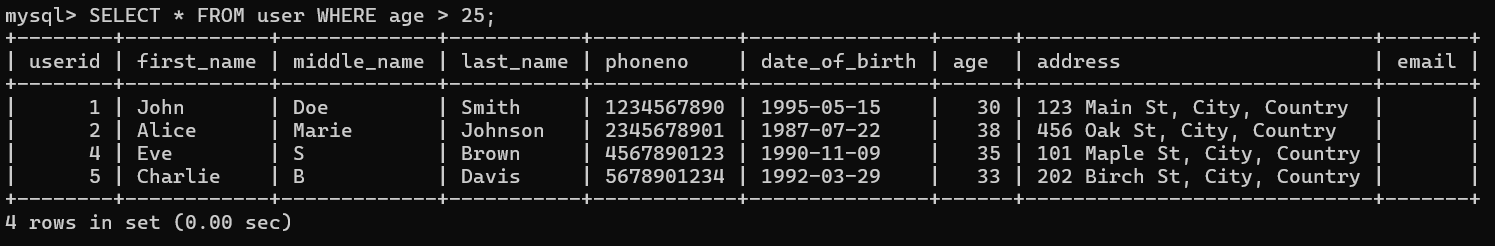
SELECT DISTINCT wishlist\_id FROM wishlist;

A black screen with white text

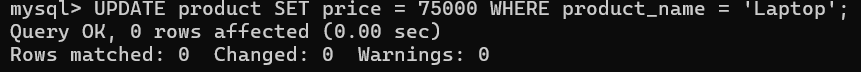
AI-generated content may be incorrect.

1. WHERE claus:

SELECT \* FROM user WHERE age > 25;



UPDATE product SET price = 75000 WHERE product\_name = 'Laptop';



SELECT \* FROM product WHERE category\_id = (SELECT category\_id FROM category WHERE category\_name = 'Electronics');

A screenshot of a computer

AI-generated content may be incorrect.

SELECT \* FROM orders WHERE price > 250;

A screen shot of a computer

AI-generated content may be incorrect.

SELECT \* FROM reviews WHERE rating >= 4;

A black screen with white text

AI-generated content may be incorrect.

SELECT \* FROM cart WHERE cart\_total > 149;

A screen shot of a computer

AI-generated content may be incorrect.

SELECT \* FROM supplier WHERE supplier\_id = 2;

A screen shot of a computer

AI-generated content may be incorrect.

SELECT \* FROM wishlist WHERE quantity\_needed > 1;

A screen shot of a computer program

AI-generated content may be incorrect.

1. Queries on Multiple Relations

SELECT orders.order\_id, orders.product\_name, orders.price, user.first\_name, user.middle\_name, user.last\_name, user.email

FROM orders

JOIN user ON orders.user\_id = user.userid;

A screenshot of a computer screen

AI-generated content may be incorrect.

SELECT product.product\_id, product.product\_name, product.price, category.category\_name

FROM product

JOIN category ON product.category\_id = category.category\_id;

A screen shot of a computer

AI-generated content may be incorrect.

SELECT user.userid, user.first\_name, user.last\_name, user.email

FROM user

WHERE user.userid IN (SELECT reviews.user\_id FROM reviews);

A screenshot of a computer program

AI-generated content may be incorrect.

SELECT orders.order\_id, orders.product\_name, orders.price, users.first\_name, users.middle\_name, users.last\_name, users.email

FROM orders

JOIN users ON orders.user\_id = users.userid;

A screen shot of a computer

AI-generated content may be incorrect.

SELECT products.product\_id, products.product\_name, products.price, category.category\_name

FROM products

JOIN category ON products.category\_id = category.category\_id;

A screen shot of a computer

AI-generated content may be incorrect.

SELECT users.userid, users.first\_name, users.last\_name, users.email

FROM users

WHERE users.userid IN (SELECT reviews.user\_id FROM reviews);

A screenshot of a computer

AI-generated content may be incorrect.

SELECT cart.cart\_id, cart.cart\_total, products.product\_name, products.price

FROM cart

JOIN products ON cart.product\_id = products.product\_id;

A screen shot of a computer

AI-generated content may be incorrect.

SELECT wishlist.wishlist\_id, wishlist.product\_name, products.product\_name, wishlist.quantity\_needed

FROM wishlist

JOIN products ON wishlist.product\_id = products.product\_id;

A screen shot of a computer program

AI-generated content may be incorrect.

SELECT supplier.supplier\_id, supplier.contact\_info, products.product\_name, products.price

FROM supplier

JOIN products ON supplier.product\_id = products.product\_id;

A screen shot of a computer

AI-generated content may be incorrect.

SELECT payments.payment\_id, payments.payment\_method, payments.amount, orders.order\_id

FROM payments

JOIN orders ON payments.order\_id = orders.order\_id;

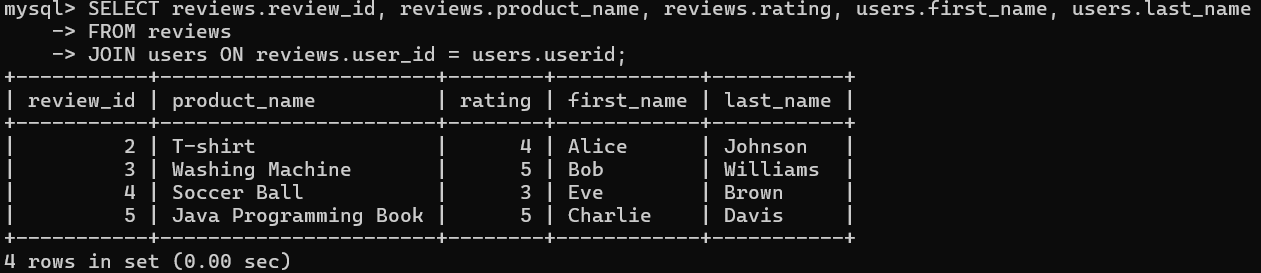
A screen shot of a computer

AI-generated content may be incorrect.

SELECT reviews.review\_id, reviews.product\_name, reviews.rating, users.first\_name, users.last\_name

FROM reviews

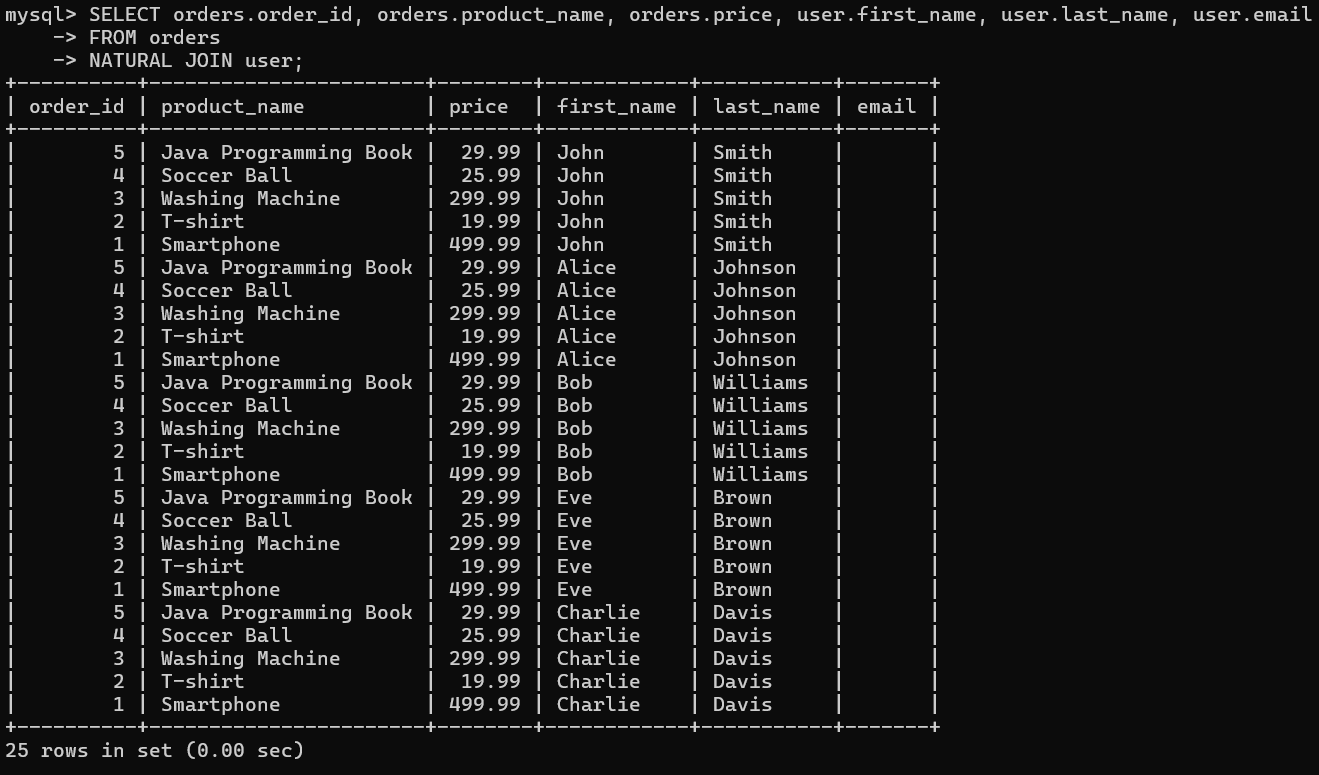
JOIN users ON reviews.user\_id = users.userid;



1. Natural Join

SELECT orders.order\_id, orders.product\_name, orders.price, user.first\_name, user.last\_name, user.email

FROM orders

NATURAL JOIN user;

SELECT product.product\_id, product.product\_name, product.price, category.category\_name

FROM product

NATURAL JOIN category;

A screenshot of a computer screen

AI-generated content may be incorrect.

SELECT reviews.review\_id, reviews.product\_name, reviews.rating, user.first\_name, user.last\_name

FROM reviews

NATURAL JOIN user;

A screenshot of a computer

AI-generated content may be incorrect.

SELECT cart.cart\_id, cart.cart\_total, products.product\_name, products.price

FROM cart

NATURAL JOIN products;

A screen shot of a computer

AI-generated content may be incorrect.

SELECT wishlist.wishlist\_id, wishlist.product\_name, products.product\_name, wishlist.quantity\_needed

FROM wishlist

NATURAL JOIN products;

A screen shot of a computer

AI-generated content may be incorrect.

SELECT supplier.supplier\_id, supplier.contact\_info, products.product\_name, products.price

FROM supplier

NATURAL JOIN products;

A screen shot of a computer

AI-generated content may be incorrect.

SELECT payments.payment\_id, payments.payment\_method, payments.amount, orders.order\_id

FROM payments

NATURAL JOIN orders;

A screen shot of a computer

AI-generated content may be incorrect.

SELECT reviews.review\_id, reviews.product\_name, reviews.rating, users.first\_name, users.last\_name

FROM reviews

NATURAL JOIN users;

A screenshot of a computer

AI-generated content may be incorrect.

1. Rename Operation

RENAME TABLE user TO users;



RENAME TABLE orders TO order\_details;



1. String Operation

SELECT CONCAT(first\_name, ' ', last\_name) AS full\_name

FROM users;

A computer screen shot of a black screen

AI-generated content may be incorrect.

SELECT UPPER(product\_name) AS uppercase\_product\_name

FROM product;

A screen shot of a computer

AI-generated content may be incorrect.

SELECT CONCAT(first\_name, ' ', middle\_name, ' ', last\_name) AS full\_name

FROM users;

A black screen with white text

AI-generated content may be incorrect.

SELECT CONCAT(product\_name, ' - ', category.category\_name) AS product\_category

FROM products

JOIN category ON products.category\_id = category.category\_id;

A screen shot of a computer program

AI-generated content may be incorrect.

SELECT LOWER(payment\_method) AS payment\_method\_lower

FROM payments;

A screen shot of a computer program

AI-generated content may be incorrect.

SELECT LENGTH(product\_name) AS product\_name\_length

FROM products;

A screen shot of a computer

AI-generated content may be incorrect.

SELECT SUBSTRING(product\_name, 1, 5) AS product\_name\_substring

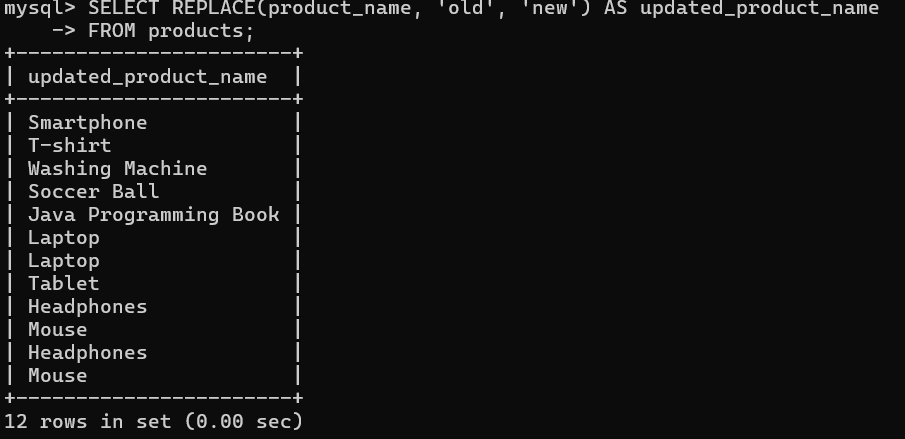
FROM products;

A screen shot of a computer

AI-generated content may be incorrect.

SELECT REPLACE(product\_name, 'old', 'new') AS updated\_product\_name

FROM products;

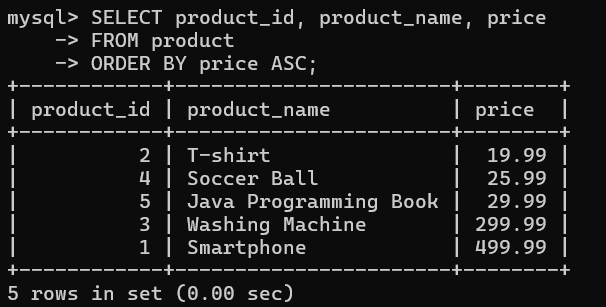


1. Order By Ascending

SELECT product\_id, product\_name, price

FROM product

ORDER BY price ASC;



SELECT userid, first\_name, last\_name, email

FROM users

ORDER BY last\_name ASC;

A screen shot of a computer screen

AI-generated content may be incorrect.

SELECT order\_id, order\_date, product\_name, price

FROM order\_details

ORDER BY order\_date ASC;

A screen shot of a computer

AI-generated content may be incorrect.

SELECT payment\_id, payment\_method, amount

FROM payments

ORDER BY amount ASC;

A screen shot of a computer

AI-generated content may be incorrect.

SELECT order\_id, product\_name, quantity, price

FROM orders

ORDER BY price ASC;

A screen shot of a computer program

AI-generated content may be incorrect.

SELECT review\_id, product\_name, rating

FROM reviews

ORDER BY rating ASC;

A screen shot of a computer program

AI-generated content may be incorrect.

SELECT cart\_id, cart\_total, product\_name

FROM cart

ORDER BY cart\_total ASC;

A screen shot of a computer program

AI-generated content may be incorrect.

SELECT category\_id, category\_name, description

FROM category

ORDER BY category\_name ASC;

A screen shot of a computer

AI-generated content may be incorrect.

SELECT supplier\_id, product\_name, supplier\_address

FROM supplier

ORDER BY supplier\_id ASC;

A screen shot of a computer

AI-generated content may be incorrect.

SELECT product\_id, product\_name, price, quantity

FROM products

ORDER BY price ASC;

A screenshot of a computer

AI-generated content may be incorrect.

SELECT wishlist\_id, product\_name, quantity\_needed

FROM wishlist

ORDER BY quantity\_needed ASC;

A screen shot of a computer program

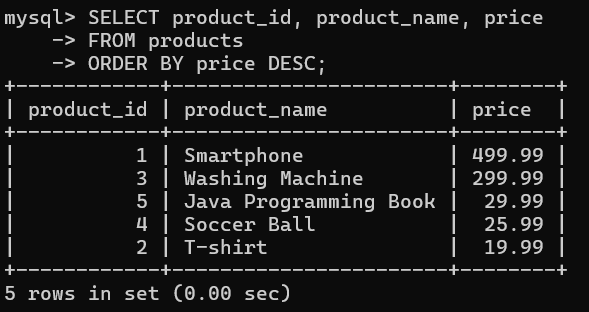
AI-generated content may be incorrect.

1. Order By Descending

SELECT product\_id, product\_name, price

FROM products

ORDER BY price DESC;



SELECT userid, first\_name, last\_name, email

FROM users

ORDER BY last\_name DESC;A screen shot of a computer

AI-generated content may be incorrect.

SELECT order\_id, order\_date, product\_name, price

FROM order\_details

ORDER BY order\_date DESC;

A screen shot of a computer program

AI-generated content may be incorrect.

SELECT order\_id, price, order\_date, product\_name, quantity, user\_id, status

FROM orders

ORDER BY price DESC;

A screen shot of a computer

AI-generated content may be incorrect.

SELECT payment\_id, payment\_method, amount, order\_id

FROM payments

ORDER BY amount DESC;

A screen shot of a computer

AI-generated content may be incorrect.

SELECT review\_id, product\_name, user\_id, rating

FROM reviews

ORDER BY rating DESC;

A screen shot of a computer program

AI-generated content may be incorrect.

SELECT cart\_id, cart\_total, product\_name, product\_id, user\_id

FROM cart

ORDER BY cart\_total DESC;

A screen shot of a computer

AI-generated content may be incorrect.

SELECT category\_id, category\_name, description

FROM category

ORDER BY category\_name DESC;

A screen shot of a computer

AI-generated content may be incorrect.

SELECT supplier\_id, contact\_info, product\_id, product\_name, supplier\_address

FROM supplier

ORDER BY supplier\_id DESC;

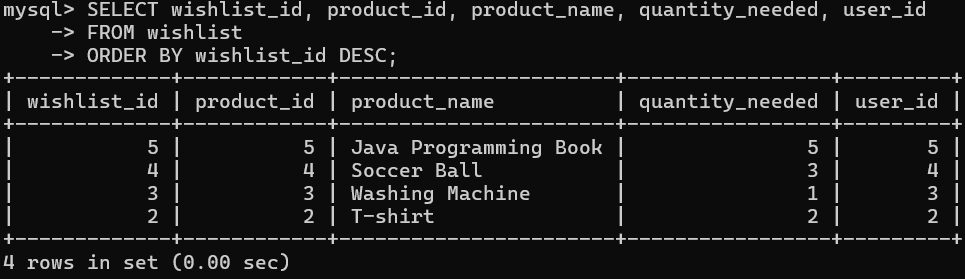
A screen shot of a computer program

AI-generated content may be incorrect.

SELECT wishlist\_id, product\_id, product\_name, quantity\_needed, user\_id

FROM wishlist

ORDER BY wishlist\_id DESC;

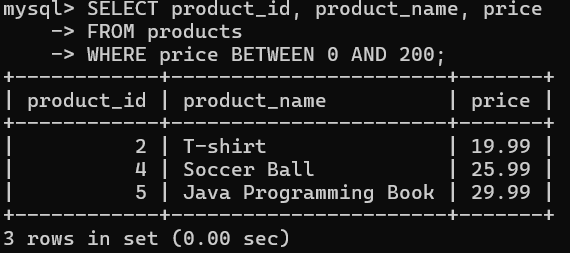


1. Where Clause Between

SELECT product\_id, product\_name, price

FROM products

WHERE price BETWEEN 0 AND 200;



SELECT order\_id, order\_date, product\_name, price

FROM order\_details

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

A black background with white text

AI-generated content may be incorrect.

SELECT userid, first\_name, last\_name, age

FROM users

WHERE age BETWEEN 20 AND 40;

A screen shot of a computer

AI-generated content may be incorrect.

SELECT order\_id, price, order\_date, product\_name, quantity, user\_id, status

FROM orders

WHERE price BETWEEN 100 AND 500;

A screen shot of a computer

AI-generated content may be incorrect.

SELECT payment\_id, payment\_method, amount, order\_id

FROM payments

WHERE amount BETWEEN 50 AND 500;

A screen shot of a computer screen

AI-generated content may be incorrect.

SELECT review\_id, product\_name, user\_id, rating

FROM reviews

WHERE rating BETWEEN 3 AND 5;

A screen shot of a computer program

AI-generated content may be incorrect.

SELECT cart\_id, cart\_total, product\_name, product\_id, user\_id

FROM cart

WHERE cart\_total BETWEEN 0 AND 500;

A screen shot of a computer program

AI-generated content may be incorrect.

SELECT category\_id, category\_name, description

FROM category

WHERE category\_id BETWEEN 1 AND 5;

A computer screen with white text

AI-generated content may be incorrect.

SELECT supplier\_id, contact\_info, product\_id, product\_name, supplier\_address

FROM supplier

WHERE supplier\_id BETWEEN 1 AND 10;

A screen shot of a computer

AI-generated content may be incorrect.

SELECT wishlist\_id, product\_id, product\_name, quantity\_needed, user\_id

FROM wishlist

WHERE quantity\_needed BETWEEN 4 AND 10;

A screen shot of a computer

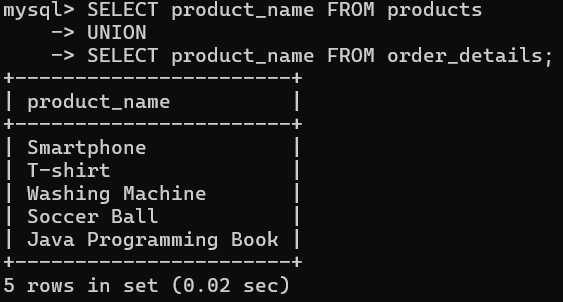
AI-generated content may be incorrect.

1. Set Operations (UNION, INTERSECT, EXCEPT)

SELECT product\_name FROM products

UNION

SELECT product\_name FROM order\_details;



SELECT product\_name FROM products

INTERSECT

SELECT product\_name FROM order\_details;

A screen shot of a computer

AI-generated content may be incorrect.

SELECT product\_name FROM products

EXCEPT

SELECT product\_name FROM order\_details;

A black background with white text

AI-generated content may be incorrect.

SELECT product\_name FROM products

UNION

SELECT product\_name FROM orders;

A screen shot of a computer

AI-generated content may be incorrect.

SELECT userid FROM users

INTERSECT

SELECT user\_id FROM reviews;

A screenshot of a computer program

AI-generated content may be incorrect.

SELECT product\_name FROM products

EXCEPT

SELECT product\_name FROM orders;

A screen shot of a computer

AI-generated content may be incorrect.

SELECT product\_name FROM cart

UNION

SELECT product\_name FROM wishlist;

A screen shot of a computer

AI-generated content may be incorrect.

SELECT category\_id FROM category

INTERSECT

SELECT category\_id FROM products;

A screenshot of a computer program

AI-generated content may be incorrect.

SELECT product\_name FROM supplier

EXCEPT

SELECT product\_name FROM products;

A black background with white text

AI-generated content may be incorrect.

SELECT order\_id FROM orders

UNION

SELECT order\_id FROM payments;

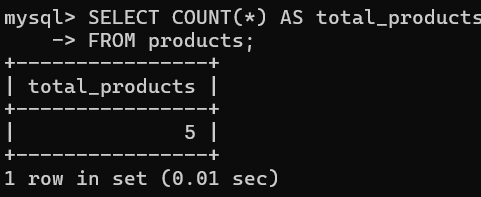
A screenshot of a computer program

AI-generated content may be incorrect.

1. Aggregate Function

SELECT COUNT(\*) AS total\_products

FROM products;



SELECT SUM(price) AS total\_price

FROM products;

A black screen with white text

AI-generated content may be incorrect.

SELECT AVG(price) AS average\_price

FROM products;

A black screen with white text

AI-generated content may be incorrect.

SELECT COUNT(\*) AS total\_orders

FROM orders;

A black screen with white text

AI-generated content may be incorrect.

SELECT SUM(price) AS total\_order\_value

FROM orders;

A screen shot of a computer code

AI-generated content may be incorrect.

SELECT AVG(price) AS average\_order\_price

FROM orders;

A screen shot of a computer code

AI-generated content may be incorrect.

SELECT COUNT(\*) AS total\_users

FROM users;

A black screen with white text

AI-generated content may be incorrect.

SELECT AVG(rating) AS average\_rating

FROM reviews;

A screen shot of a computer

AI-generated content may be incorrect.

SELECT COUNT(\*) AS total\_wishlist\_items

FROM wishlist;

A screen shot of a computer program

AI-generated content may be incorrect.

SELECT COUNT(\*) AS total\_suppliers

FROM supplier;

A black screen with white text

AI-generated content may be incorrect.

1. Aggregate with Grouping
   1. COUNT with GROUP BY

SELECT product\_name, COUNT(\*) AS total\_orders

FROM order\_details

GROUP BY product\_name;

A screen shot of a computer

AI-generated content may be incorrect.

* 1. SUM with GROUP BY

SELECT product\_name, SUM(price) AS total\_price

FROM orders

GROUP BY product\_name;

A screen shot of a computer

AI-generated content may be incorrect.

* 1. AVG with GROUP BY

SELECT product\_name, COUNT(\*) AS total\_orders

FROM orders

GROUP BY product\_name;

A screen shot of a computer

AI-generated content may be incorrect.

SELECT category\_id, COUNT(\*) AS total\_products

FROM products

GROUP BY category\_id;

A screen shot of a computer program

AI-generated content may be incorrect.

SELECT age, COUNT(\*) AS total\_users

FROM users

GROUP BY age;

A screenshot of a computer program

AI-generated content may be incorrect.

SELECT user\_id, COUNT(\*) AS total\_orders

FROM orders

GROUP BY user\_id;

A screen shot of a computer

AI-generated content may be incorrect.

SELECT product\_name, COUNT(\*) AS total\_wishlist\_entries

FROM wishlist

GROUP BY product\_name;

A computer screen with white text

AI-generated content may be incorrect.

SELECT product\_name, COUNT(\*) AS total\_suppliers

FROM supplier

GROUP BY product\_name;

A screen shot of a computer

AI-generated content may be incorrect.

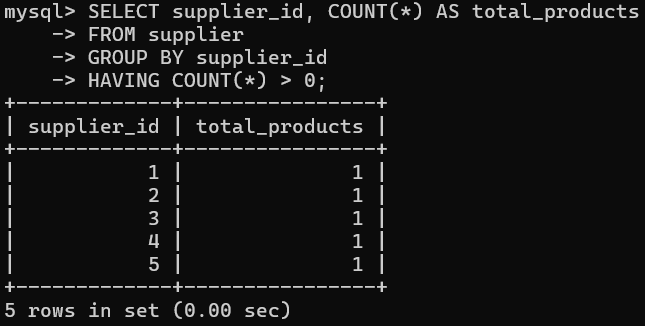
1. Having Clause

SELECT supplier\_id, COUNT(\*) AS total\_products

FROM supplier

GROUP BY supplier\_id

HAVING COUNT(\*) > 0;



SELECT product\_name, SUM(price) AS total\_price

FROM order\_details

GROUP BY product\_name

HAVING SUM(price) > 100;

A screen shot of a computer

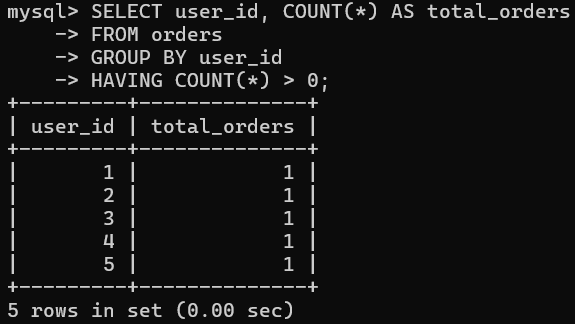
AI-generated content may be incorrect.

SELECT user\_id, COUNT(\*) AS total\_orders

FROM orders

GROUP BY user\_id

HAVING COUNT(\*) > 0;



SELECT category\_id, COUNT(\*) AS total\_products

FROM products

GROUP BY category\_id

HAVING COUNT(\*) > 5;

A screen shot of a computer

AI-generated content may be incorrect.

SELECT age, COUNT(\*) AS total\_users

FROM users

GROUP BY age

HAVING COUNT(\*) > 0;

A screen shot of a computer program

AI-generated content may be incorrect.

SELECT product\_name, AVG(rating) AS average\_rating

FROM reviews

GROUP BY product\_name

HAVING AVG(rating) > 4;

A screen shot of a computer

AI-generated content may be incorrect.

SELECT product\_name, COUNT(\*) AS total\_wishlist\_entries

FROM wishlist

GROUP BY product\_name

HAVING COUNT(\*) > 0;

A screen shot of a computer program

AI-generated content may be incorrect.

SELECT product\_name, COUNT(\*) AS total\_orders

FROM orders

GROUP BY product\_name

HAVING COUNT(\*) > 1;

A screen shot of a computer

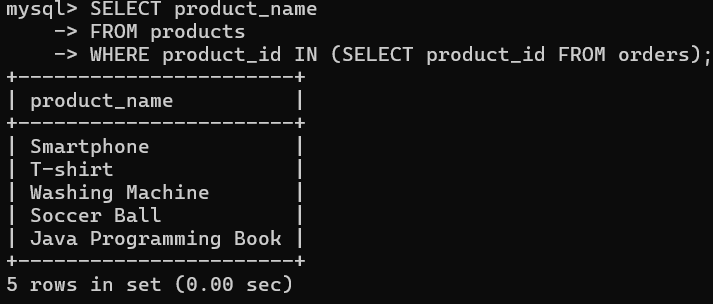
AI-generated content may be incorrect.

1. Nested Queries

SELECT product\_name

FROM products

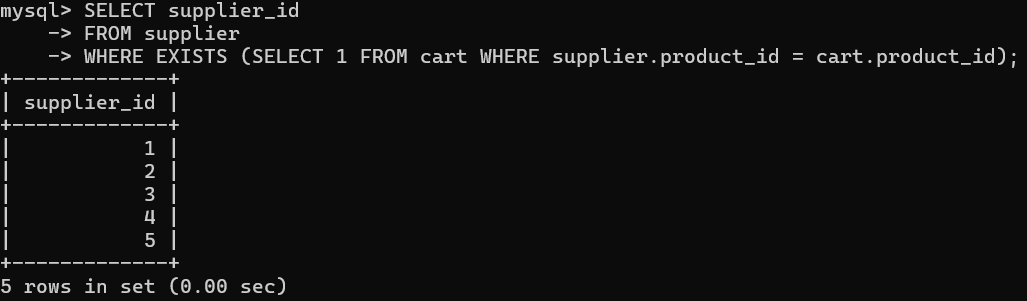
WHERE product\_id IN (SELECT product\_id FROM orders);



SELECT supplier\_id

FROM supplier

WHERE EXISTS (SELECT 1 FROM cart WHERE supplier.product\_id = cart.product\_id);



SELECT order\_id

FROM orders

WHERE user\_id IN (SELECT user\_id FROM users WHERE age > 25);

A screenshot of a computer

AI-generated content may be incorrect.

SELECT userid

FROM users

WHERE EXISTS (SELECT 1 FROM orders WHERE users.userid = orders.user\_id);

A black screen with white text

AI-generated content may be incorrect.

SELECT product\_name

FROM reviews

WHERE product\_name IN (SELECT product\_name FROM reviews WHERE rating > 3);

A screen shot of a computer

AI-generated content may be incorrect.

SELECT cart\_id

FROM cart

WHERE EXISTS (SELECT 1 FROM orders WHERE cart.product\_id = orders.order\_id);

A black screen with white text

AI-generated content may be incorrect.

SELECT product\_name

FROM wishlist

WHERE product\_id IN (SELECT product\_id FROM orders);

A screen shot of a computer

AI-generated content may be incorrect.

SELECT supplier\_id

FROM supplier

WHERE EXISTS (SELECT 1 FROM cart WHERE supplier.product\_id = cart.product\_id);

A screen shot of a computer

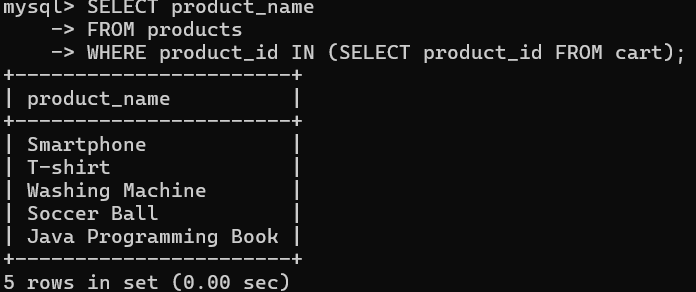
AI-generated content may be incorrect.

1. All Set Comparison (e.g., IN, ANY, ALL)
   1. IN

SELECT product\_name

FROM products

WHERE product\_id IN (SELECT product\_id FROM cart);



* 1. ANY

SELECT product\_name

FROM products

WHERE price > ANY (SELECT price FROM products WHERE product\_id IN (SELECT product\_id FROM cart));

A black screen with white text

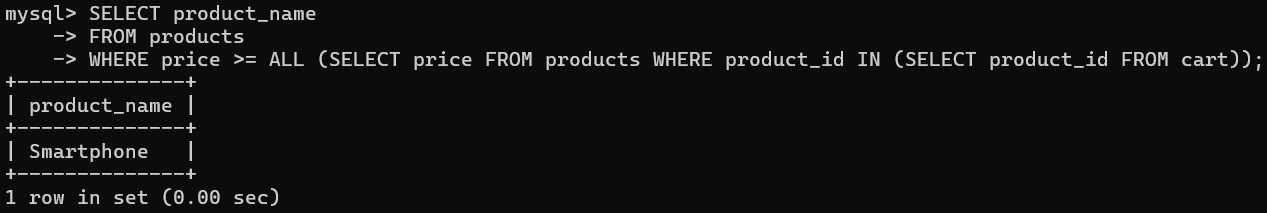
AI-generated content may be incorrect.

* 1. ALL

SELECT product\_name

FROM products

WHERE price >= ALL (SELECT price FROM products WHERE product\_id IN (SELECT product\_id FROM cart));



SELECT product\_name

FROM orders

WHERE user\_id IN (SELECT user\_id FROM users WHERE age > 25);

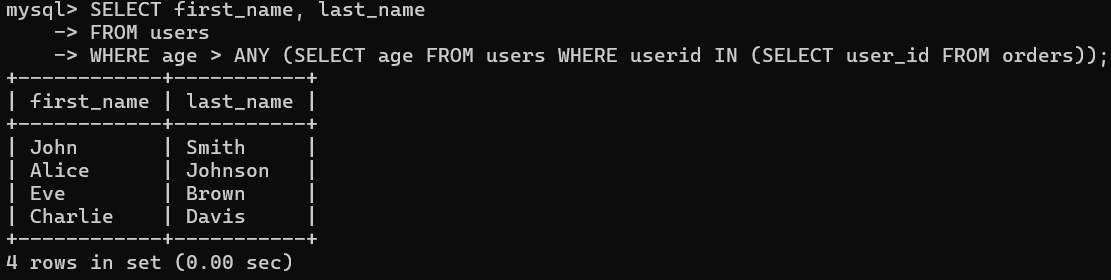
A screenshot of a computer

AI-generated content may be incorrect.

SELECT first\_name, last\_name

FROM users

WHERE age > ANY (SELECT age FROM users WHERE userid IN (SELECT user\_id FROM orders));



SELECT product\_name

FROM reviews

WHERE rating >= ALL (SELECT rating FROM reviews WHERE user\_id IN (SELECT user\_id FROM orders));

A black screen with white text

AI-generated content may be incorrect.

1. With Clause

WITH OrderTotal AS (

SELECT o.order\_id, p.product\_name, o.quantity, (p.price \* o.quantity) AS total\_price

FROM orders o

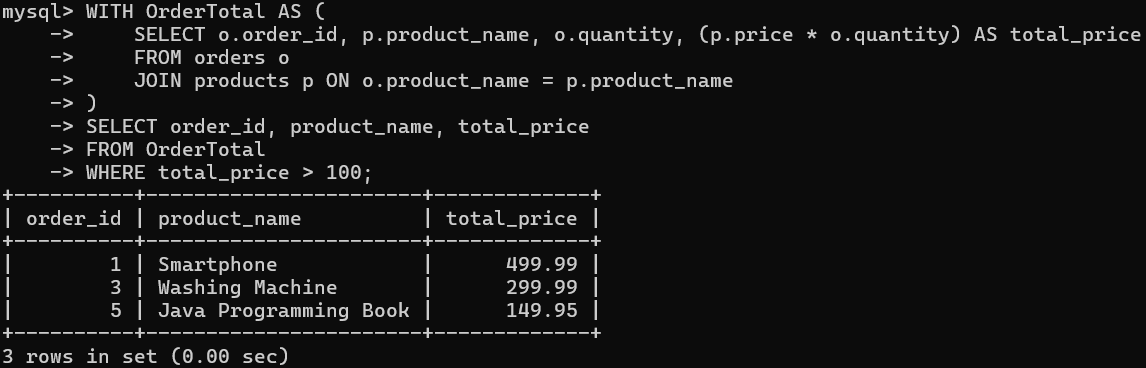
JOIN products p ON o.product\_name = p.product\_name

)

SELECT order\_id, product\_name, total\_price

FROM OrderTotal

WHERE total\_price > 100;



WITH ProductQuantities AS (

SELECT o.order\_id, p.product\_name, SUM(o.quantity) AS total\_quantity

FROM orders o

JOIN products p ON o.product\_name = p.product\_name

GROUP BY o.order\_id, p.product\_name

)

SELECT order\_id, product\_name, total\_quantity

FROM ProductQuantities

WHERE total\_quantity > 2;

A computer screen with white text

AI-generated content may be incorrect.

WITH UserOrders AS (

SELECT o.user\_id, COUNT(\*) AS total\_orders

FROM orders o

GROUP BY o.user\_id

)

SELECT u.userid, u.first\_name, u.last\_name, u.age, u.email, u.phoneno, u.address, u.date\_of\_birth, UserOrders.total\_orders

FROM users u

JOIN UserOrders ON u.userid = UserOrders.user\_id

WHERE UserOrders.total\_orders > 0;

A screenshot of a computer screen

AI-generated content may be incorrect.

WITH ProductRatings AS (

SELECT product\_name, AVG(rating) AS average\_rating

FROM reviews

GROUP BY product\_name

)

SELECT product\_name, average\_rating

FROM ProductRatings

WHERE average\_rating > 0;

A screenshot of a computer

AI-generated content may be incorrect.

WITH UserCart AS (

SELECT user\_id, SUM(cart\_total) AS total\_cart\_value

FROM cart

GROUP BY user\_id

)

SELECT u.userid, u.first\_name, u.last\_name, UserCart.total\_cart\_value

FROM users u

JOIN UserCart ON u.userid = UserCart.user\_id

WHERE UserCart.total\_cart\_value > 150;

A screenshot of a computer program

AI-generated content may be incorrect.

WITH UserWishlist AS (

SELECT user\_id, COUNT(\*) AS total\_wishlist\_items

FROM wishlist

GROUP BY user\_id

)

SELECT u.userid, u.first\_name, u.last\_name, UserWishlist.total\_wishlist\_items

FROM users u

JOIN UserWishlist ON u.userid = UserWishlist.user\_id

WHERE UserWishlist.total\_wishlist\_items > 0;

A computer screen with white text

AI-generated content may be incorrect.

WITH SupplierProducts AS (

SELECT supplier\_id, COUNT(\*) AS total\_products

FROM supplier

GROUP BY supplier\_id

)

SELECT s.supplier\_id, s.contact\_info, s.supplier\_address, SupplierProducts.total\_products

FROM supplier s

JOIN SupplierProducts ON s.supplier\_id = SupplierProducts.supplier\_id

WHERE SupplierProducts.total\_products > 0;

A screenshot of a computer

AI-generated content may be incorrect.

1. Scalar Queries

SELECT COUNT(\*)

FROM orders;

A black screen with white text

AI-generated content may be incorrect.

SELECT AVG(price)

FROM products;

A black screen with white text

AI-generated content may be incorrect.

SELECT MAX(quantity)

FROM orders;

A black screen with white text

AI-generated content may be incorrect.

1. Update

UPDATE products

SET price = 99.99

WHERE product\_name = 'T-shirt';

A black background with white text

AI-generated content may be incorrect.

UPDATE orders

SET quantity = 5

WHERE order\_id = 2;

A black background with white text

AI-generated content may be incorrect.

UPDATE cart

SET cart\_total = 150.75

WHERE cart\_id = 3;

A black background with white text

AI-generated content may be incorrect.

1. Types of Join (INNER, LEFT, RIGHT, FULL)
2. INNER JOIN

SELECT o.order\_id, p.product\_name, o.quantity

FROM orders o

INNER JOIN products p ON o.product\_name = p.product\_name;

A screen shot of a computer program

AI-generated content may be incorrect.

1. LEFT JOIN

SELECT o.order\_id, p.product\_name, o.quantity

FROM orders o

LEFT JOIN products p ON o.product\_name = p.product\_name;

A screen shot of a computer program

AI-generated content may be incorrect.

1. RIGHT JOIN

SELECT o.order\_id, p.product\_name, o.quantity

FROM orders o

RIGHT JOIN products p ON o.product\_name = p.product\_name;

A screen shot of a computer

AI-generated content may be incorrect.

1. FULL JOIN

SELECT o.order\_id, p.product\_name, o.quantity

FROM orders o

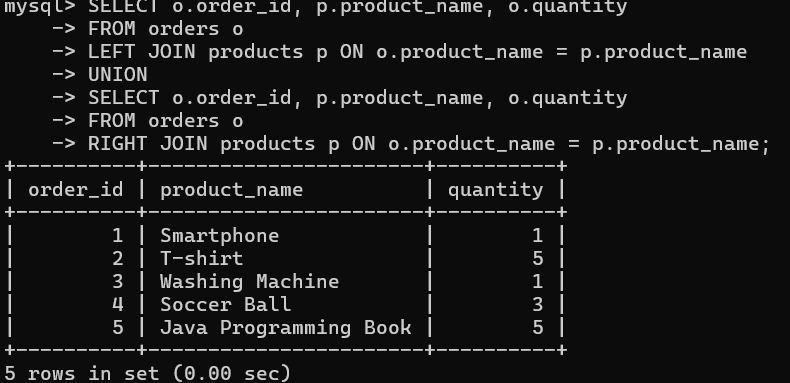
LEFT JOIN products p ON o.product\_name = p.product\_name

UNION

SELECT o.order\_id, p.product\_name, o.quantity

FROM orders o

RIGHT JOIN products p ON o.product\_name = p.product\_name;



1. COMMIT

-- Start a new transaction

START TRANSACTION;

-- Insert a new record into the products table

INSERT INTO products (product\_name, price)

VALUES ('Laptop', 999.99);

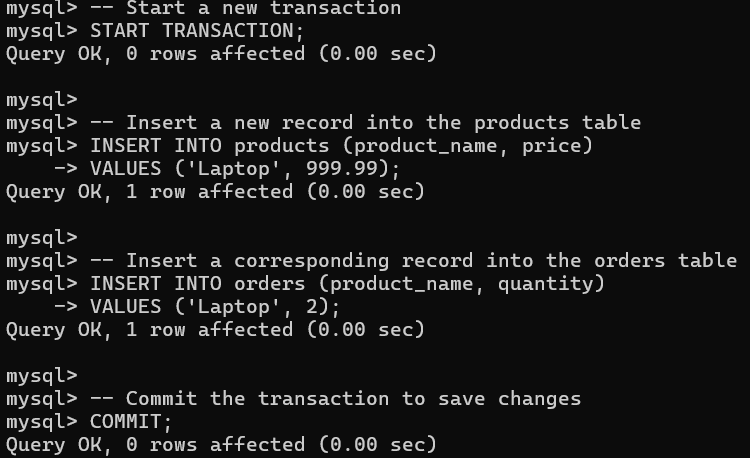
-- Insert a corresponding record into the orders table

INSERT INTO orders (product\_name, quantity)

VALUES ('Laptop', 2);

-- Commit the transaction to save changes

COMMIT;



-- Start a new transaction

START TRANSACTION;

-- Update the product price for a specific product

UPDATE products

SET price = 89.99

WHERE product\_name = 'T-shirt';

-- Insert a new order into the orders table

INSERT INTO orders (product\_name, quantity)

VALUES ('Shoes', 3);

-- Commit the transaction to save changes

COMMIT;

A screen shot of a computer program

AI-generated content may be incorrect.

1. ROLLBACK

-- Start a new transaction

START TRANSACTION;

-- Update the price of a product

UPDATE products

SET price = 79.99

WHERE product\_name = 'T-shirt';

-- Simulate a mistake (you can decide to rollback)

-- Undo all changes made in this transaction

ROLLBACK;

A screen shot of a computer

AI-generated content may be incorrect.

-- Start a new transaction

START TRANSACTION;

-- Insert a new order into the orders table

INSERT INTO orders (product\_name, quantity)

VALUES ('Shoes', 5);

-- You realize there's an error in the inserted data

-- Rollback to undo the insertion

ROLLBACK;

A computer screen shot of white text

AI-generated content may be incorrect.

START TRANSACTION;

INSERT INTO products (product\_name, price)

VALUES ('Smartphone', 499.99);

UPDATE products

SET price = 199.99

WHERE product\_name = 'Laptop';

ROLLBACK;

A screenshot of a computer program

AI-generated content may be incorrect.

1. SAVEPOINT

START TRANSACTION;

INSERT INTO products (product\_name, price)

VALUES ('Tablet', 299.99);

SAVEPOINT after\_insert;

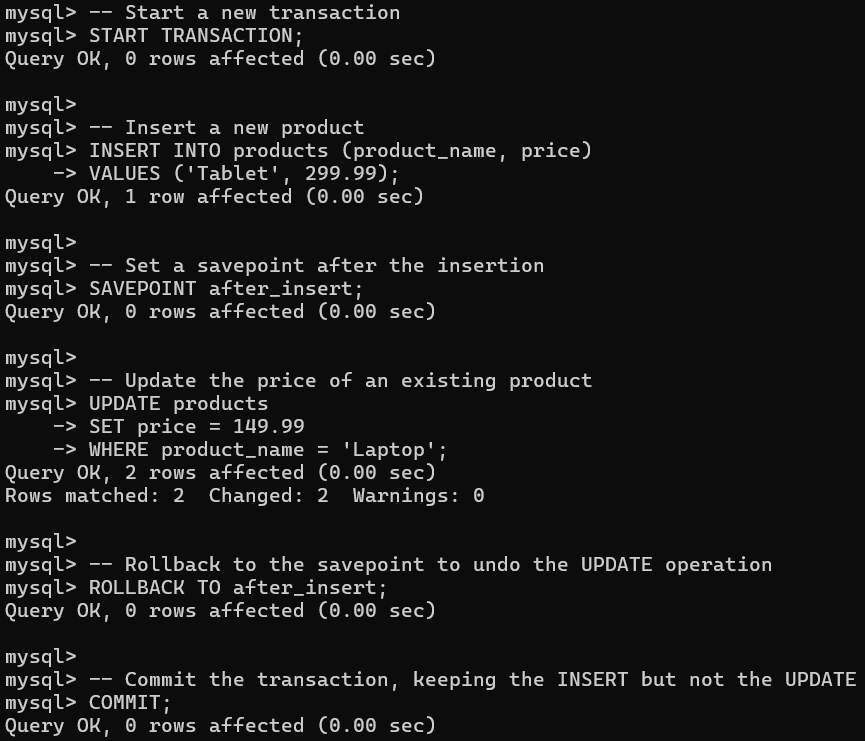
UPDATE products

SET price = 149.99

WHERE product\_name = 'Laptop';

ROLLBACK TO after\_insert;

COMMIT;



START TRANSACTION;

INSERT INTO products (product\_name, price)

VALUES ('Headphones', 79.99);

SAVEPOINT after\_first\_insert;

INSERT INTO products (product\_name, price)

VALUES ('Mouse', 25.99);

SAVEPOINT after\_second\_insert;

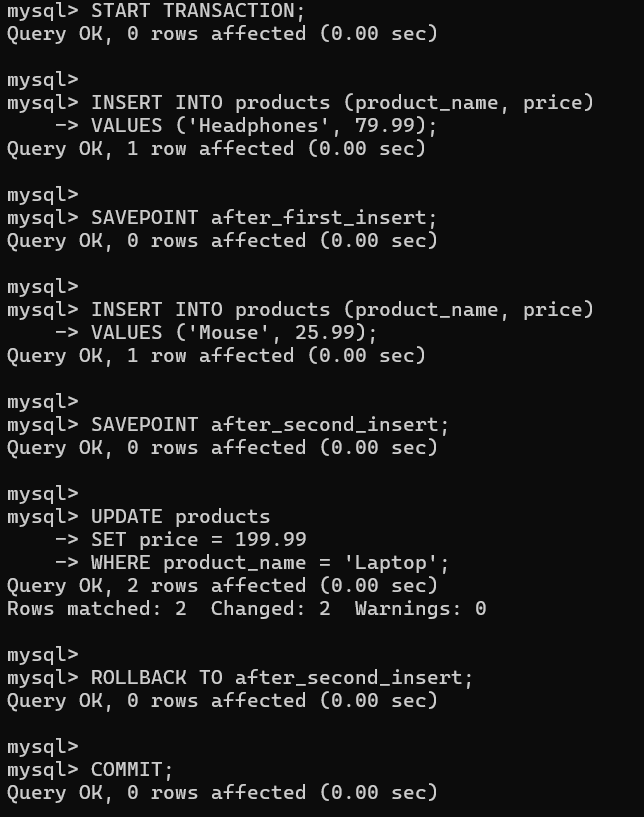
UPDATE products

SET price = 199.99

WHERE product\_name = 'Laptop';

ROLLBACK TO after\_second\_insert;

COMMIT;



1. PL/SQL (Procedural Language in SQL)

DELIMITER $$

CREATE PROCEDURE GetUserDetails(IN u\_id INT)

BEGIN

SELECT userid, first\_name, middle\_name, last\_name, phoneno, date\_of\_birth, age, address

FROM users

WHERE userid = u\_id;

END $$

DELIMITER ;

CALL GetUserDetails(1);

A screenshot of a computer

AI-generated content may be incorrect.

DELIMITER $$

CREATE FUNCTION GetCartTotal(u\_id INT) RETURNS DECIMAL(10,2)

DETERMINISTIC

BEGIN

DECLARE total DECIMAL(10,2);

SELECT SUM(cart\_total) INTO total

FROM cart

WHERE user\_id = u\_id;

RETURN total;

END $$

DELIMITER ;

SELECT GetCartTotal(1);

A screenshot of a computer program

AI-generated content may be incorrect.

1. Cursor( Implicit)

DELIMITER $$

CREATE PROCEDURE GetProductDetails()

BEGIN

DECLARE v\_product\_name VARCHAR(100);

DECLARE v\_product\_price DECIMAL(10, 2);

DECLARE product\_cursor CURSOR FOR

SELECT product\_name, price

FROM products

WHERE product\_id = 1;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET @done = TRUE;

OPEN product\_cursor;

read\_loop: LOOP

FETCH product\_cursor INTO v\_product\_name, v\_product\_price;

IF @done THEN

LEAVE read\_loop;

END IF;

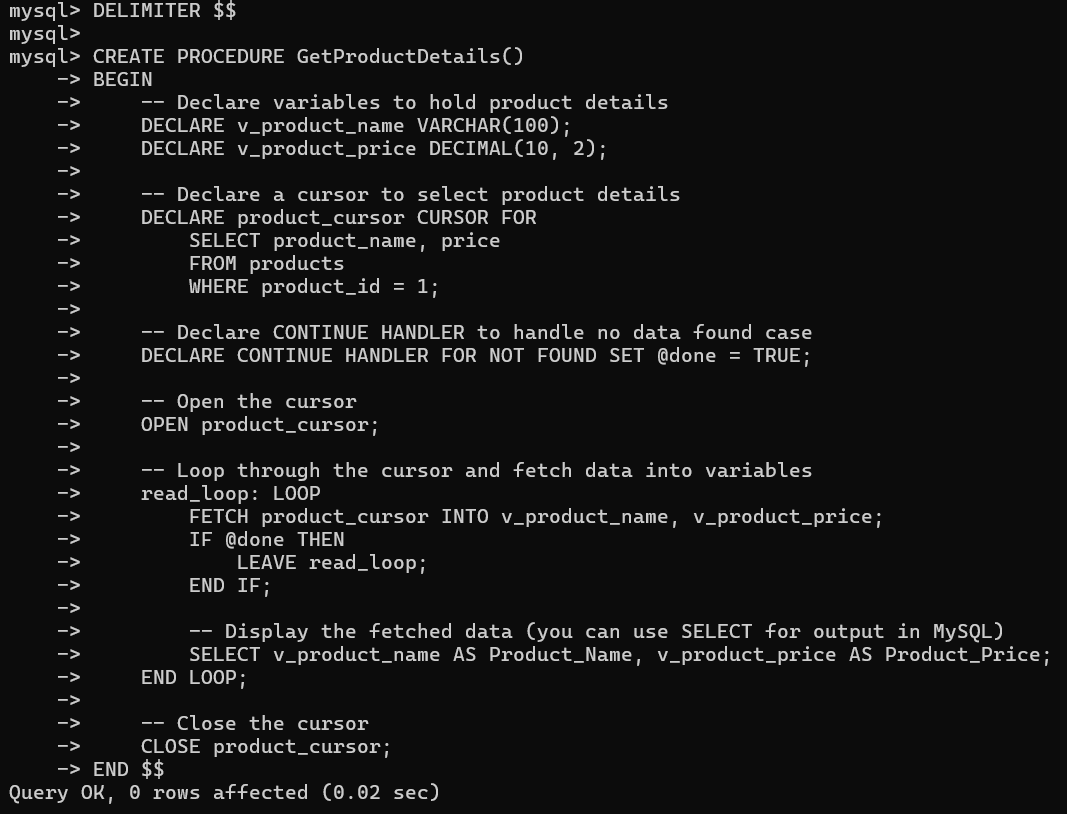
SELECT v\_product\_name AS Product\_Name, v\_product\_price AS Product\_Price;

END LOOP;

CLOSE product\_cursor;

END $$

DELIMITER ;



1. Cursor ( Explicit)

DELIMITER $$

CREATE PROCEDURE GetExplicitProductDetails()

BEGIN

DECLARE v\_product\_name VARCHAR(100);

DECLARE v\_product\_price DECIMAL(10, 2);

DECLARE product\_cursor CURSOR FOR

SELECT product\_name, price

FROM products

WHERE price > 50; -- You can change the condition as needed

DECLARE CONTINUE HANDLER FOR NOT FOUND SET @done = TRUE;

OPEN product\_cursor;

read\_loop: LOOP

FETCH product\_cursor INTO v\_product\_name, v\_product\_price;

IF @done THEN

LEAVE read\_loop;

END IF;

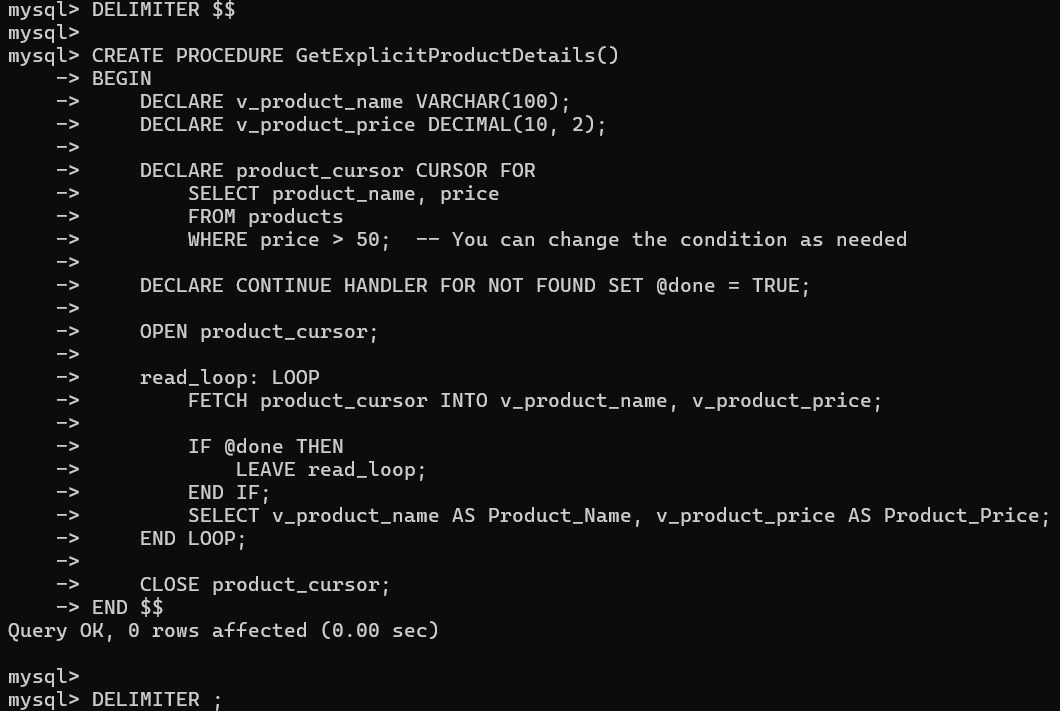
SELECT v\_product\_name AS Product\_Name, v\_product\_price AS Product\_Price;

END LOOP;

CLOSE product\_cursor;

END $$

DELIMITER ;



1. Triggers

DELIMITER $$

CREATE TRIGGER before\_insert\_product

BEFORE INSERT ON products

FOR EACH ROW

BEGIN

-- Check if the price is negative

IF NEW.price < 0 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Price cannot be negative';

END IF;

END $$

DELIMITER ;

A computer screen with white text

AI-generated content may be incorrect.

DELIMITER $$

CREATE TRIGGER after\_insert\_product

AFTER INSERT ON products

FOR EACH ROW

BEGIN

INSERT INTO product\_audit (product\_id, product\_name, action)

VALUES (NEW.product\_id, NEW.product\_name, 'INSERT');

END $$

DELIMITER ;

A screen shot of a computer program

AI-generated content may be incorrect.

DELIMITER $$

CREATE TRIGGER before\_update\_product

BEFORE UPDATE ON products

FOR EACH ROW

BEGIN

-- Check if the new product name length is less than 5 characters

IF LENGTH(NEW.product\_name) < 5 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Product name must be at least 5 characters long';

END IF;

END $$

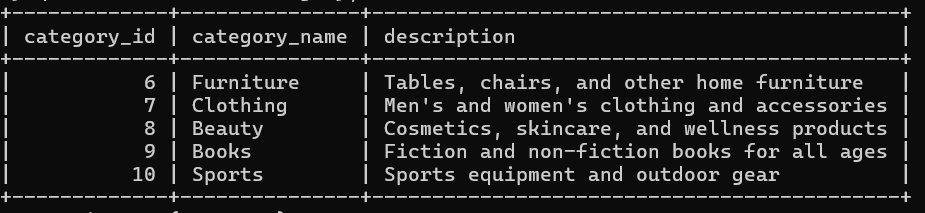
DELIMITER ;

A screen shot of a computer

AI-generated content may be incorrect.

**Normalization**

1. Category: -



|  |  |  |
| --- | --- | --- |
| Normal Form | Status | Reason |
| 1NF | YES | All attributes are atomic |
| 2NF | YES | Fully functionally dependent on the primary key |
| 3NF | YES | No transitive dependencies |
| 4NF | YES | No multi-valued dependencies |
| 5NF | YES | No join dependencies |

Changes: - None — already normalized.

1. User : -

A screen shot of a computer

AI-generated content may be incorrect.

|  |  |  |
| --- | --- | --- |
| Normal Form | Status | Reason |
| 1NF | No | Contains multi-valued attribute: phone numbers |
| 2NF | No | Partial dependency exists (phone numbers are dependent on user\_id, not on the primary key). |
| 3NF | No | Transitive dependency exists (phone numbers depend on user\_id, but are not fully dependent on the primary key). |
| 4NF | No | Multi-valued dependency exists (phones) |
| 5NF | No | Data is not fully decomposed into the smallest possible parts. Data is not fully decomposed into the smallest possible parts. |

Changes: - We need to remove the phone1 and phone2 columns from the User table and store them separately in a new User\_Phone table to resolve the multivalued column issue and move towards normalization.

After Normalization:

A black screen with white text

AI-generated content may be incorrect.

A screenshot of a phone number

AI-generated content may be incorrect.

|  |  |  |
| --- | --- | --- |
| Normal Form | Status | Reason |
| 1NF | YES | no multivalued columns |
| 2NF | YES | No partial dependency (phone numbers moved to separate table). |
| 3NF | YES | No transitive dependency (phone numbers stored separately). |
| 4NF | YES | No multivalued dependency (phone numbers stored separately). |
| 5NF | YES | Table is decomposed into smallest parts (phone numbers in separate rows). |

1. Cart :-

A screen shot of a computer

AI-generated content may be incorrect.

|  |  |  |
| --- | --- | --- |
| Normal Form | Status | Reason |
| 1NF | YES | Atomic attributes |
| 2NF | YES | Fully functionally dependent on primary key |
| 3NF | YES | No transitive dependencies |
| 4NF | YES | No multi-valued dependencies |
| 5NF | YES | No join dependencies |

Changes: -No changes required

1. Orders:

A screen shot of a computer screen

AI-generated content may be incorrect.

|  |  |  |
| --- | --- | --- |
| Normal Form | Status | Reason |
| 1NF | YES | All values are atomic |
| 2NF | NO | product\_name is partially dependent on order\_id and user\_id |
| 3NF | NO | product\_name → product\_id (transitive) |
| 4NF | YES | No multi-valued dependencies |
| 5NF | YES | Order data mixes product and user dependencies |

Changes: -

* Remove product\_name and keep only product\_id
* Ensure product details are fetched from Products table

A screen shot of a computer code

AI-generated content may be incorrect.

A screen shot of a black screen

AI-generated content may be incorrect.

|  |  |  |
| --- | --- | --- |
| Normal Form | Status | Reason |
| 1NF | YES | All values atomic |
| 2NF | YES | Full functional dependency on primary key |
| 3NF | YES | No transitive dependencies |
| 4NF | YES | No multi-valued dependencies |
| 5NF | YES | No join dependencies |

1. Supplier Table:

A screen shot of a computer

AI-generated content may be incorrect.

|  |  |  |
| --- | --- | --- |
| Normal Form | Status | Reason |
| 1NF | YES | All attributes are atomic |
| 2NF | NO | Contains product information that doesn't depend solely on supplier\_id |
| 3NF | NO | Transitive dependency: product\_name → product\_id |
| 4NF | NO | One supplier can supply multiple products (multi-valued) |
| 5NF | NO | Product info should be in separate relation |

Changes: -

* + Remove product\_name
  + Create a new table SupplierProduct to link suppliers and products

A black background with white text

AI-generated content may be incorrect.

A screen shot of a computer code

AI-generated content may be incorrect.

A screen shot of a black screen

AI-generated content may be incorrect.

|  |  |  |
| --- | --- | --- |
| Normal From | Status | Reason |
| 1NF | YES | All atomic |
| 2NF | YES | Each attribute depends on full primary key |
| 3NF | YES | No transitive dependencies |
| 4NF | YES | No multi-valued dependencies |
| 5NF | YES | No join dependencies |

1. Payments:

A screen shot of a black screen

AI-generated content may be incorrect.

|  |  |  |
| --- | --- | --- |
| Normal Form | Status | Reason |
| 1NF | YES | All attributes atomic |
| 2NF | YES | Fully functionally dependent on the primary key |
| 3NF | YES | No transitive dependencies |
| 4NF | YES | No multi-valued dependencies |
| 5NF | YES | No join dependencies |

Changes: - No changes required

1. Products:

A screen shot of a black screen

AI-generated content may be incorrect.

|  |  |  |
| --- | --- | --- |
| Normal Form | Status | Reason |
| 1NF | YES | All values are atomic |
| 2NF | YES | Fully dependent on primary key |
| 3NF | NO | price and quantity might be updated together |
| 4NF | YES | No multi-valued dependency |
| 5NF | NO | Product details may need split if price varies by supplier |

Changes: -

* Move price and quantity to a table where stock can be tracked per supplier

A screen shot of a computer

AI-generated content may be incorrect.

A screen shot of a black screen

AI-generated content may be incorrect.

|  |  |  |
| --- | --- | --- |
| Normal Form | Status | Reason |
| 1NF | YES | All atomic |
| 2NF | YES | All attributes depend on full primary key |
| 3NF | YES | No transitive dependencies |
| 4NF | YES | No multi-valued dependencies |
| 5NF | YES | No join dependencies |

1. Reviews:

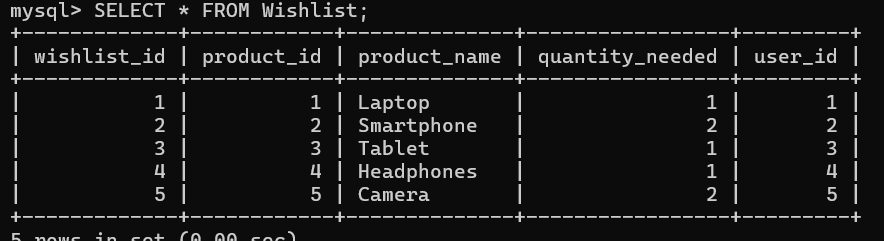
A screenshot of a computer program

AI-generated content may be incorrect.

|  |  |  |
| --- | --- | --- |
| Normal Form | Status | Reason |
| 1NF | YES | Atomic fields |
| 2NF | YES | Fully dependent on primary key |
| 3NF | YES | No transitive dependency |
| 4NF | YES | No multi-valued fields |
| 5NF | YES | No join dependency |

Changes: - No changes required.

1. WishList : -



|  |  |  |
| --- | --- | --- |
| Normal Form | Status | Reason |
| 1NF | YES | Atomic attributes |
| 2NF | YES | Fully functionally dependent on primary key |
| 3NF | YES | No transitive dependencies |
| 4NF | YES | No multi-valued dependencies |
| 5NF | YES | No join dependencies |

Changes: - No changes required.