

# MySQL Task

## Create Database and Tables

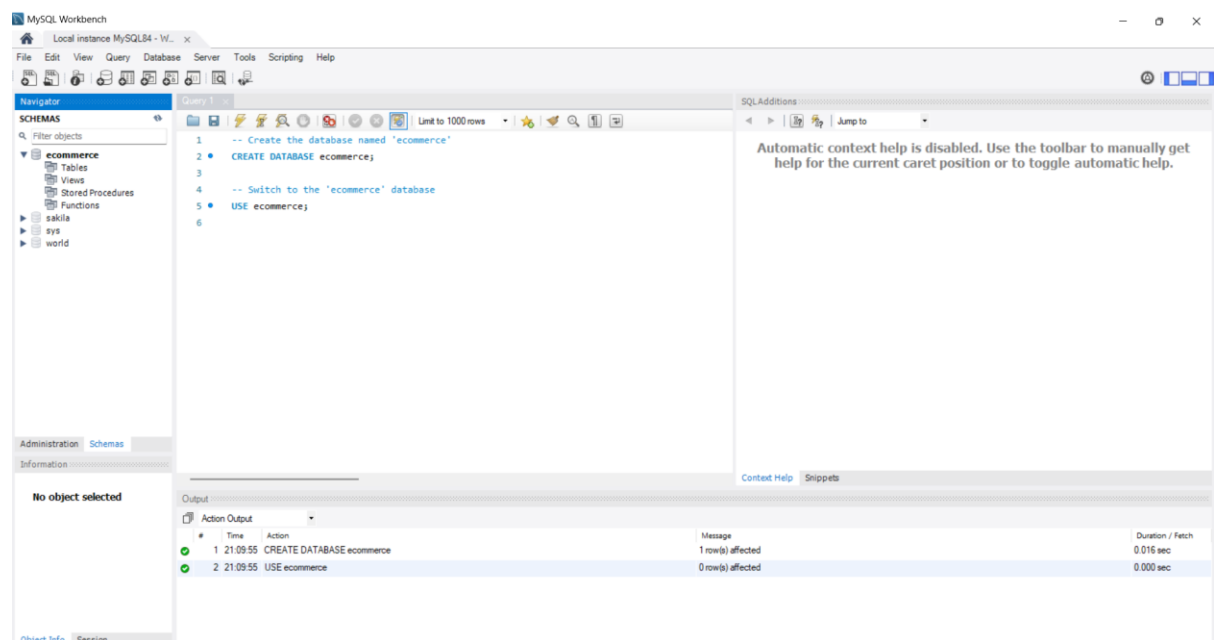
-- Create the database named 'ecommerce'

```
CREATE DATABASE ecommerce;
```

-- Switch to the 'ecommerce' database

```
USE ecommerce;
```

## Output:



-- Create the 'customers' table to store customer details

```
CREATE TABLE customers (
```

```
    id INT AUTO_INCREMENT PRIMARY KEY, -- Unique identifier for each customer
```

```
    name VARCHAR(255) NOT NULL, -- Name of the customer
```

```
    email VARCHAR(255) NOT NULL UNIQUE, -- Email address of the customer
```

```
    address TEXT NOT NULL -- Address of the customer
```

```
);
```

### -- Create the 'products' table to store product details

```
CREATE TABLE products (  
    id INT AUTO_INCREMENT PRIMARY KEY, -- Unique identifier for each product  
    name VARCHAR(255) NOT NULL, -- Name of the product  
    price DECIMAL(10, 2) NOT NULL, -- Price of the product  
    description TEXT NOT NULL -- Description of the product  
);
```

### -- Create the 'orders' table to store order details

```
CREATE TABLE orders (  
    id INT AUTO_INCREMENT PRIMARY KEY, -- Unique identifier for each order  
    customer_id INT NOT NULL, -- ID of the customer who placed the order  
    order_date DATE NOT NULL, -- Date the order was placed  
    total_amount DECIMAL(10, 2) NOT NULL, -- Total amount for the order  
    FOREIGN KEY (customer_id) REFERENCES customers(id) -- Link to the 'customers' table  
);
```

## Output:

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'Schemas' tree with 'ecommerce' selected. The main editor shows the SQL script for creating the 'orders' table. The bottom 'Output' tab shows the execution results of the commands.

| # | Time     | Action  | Message   | Duration / Fetch |
|---|----------|---|---|------------------|
| 1 | 21:09:55 | CREATE DATABASE ecommerce   | 1 row(s) affected   | 0.016 sec        |
| 2 | 21:09:55 | USE ecommerce   | 0 row(s) affected   | 0.000 sec        |
| 3 | 21:10:58 | CREATE TABLE customers ( id INT AUTO_INCREMENT PRIMARY KEY, -- Unique identifier for each customer...     | 0 row(s) affected   | 0.047 sec        |
| 4 | 21:11:17 | CREATE TABLE products ( id INT AUTO_INCREMENT PRIMARY KEY, -- Unique identifier for each product ...      | 0 row(s) affected   | 0.031 sec        |
| 5 | 21:11:34 | - Create the 'orders' table to store order details CREATE TABLE orders ( id INT AUTO_INCREMENT PRIMARY... | Error Code: 1064 You have an error in your SQL syntax; check the manual that corresponds to your MySQL serve... | 0.000 sec        |
| 6 | 21:11:57 | CREATE TABLE orders ( id INT AUTO_INCREMENT PRIMARY KEY, -- Unique identifier for each order cust...      | 0 row(s) affected   | 0.047 sec        |

## 2. Insert Sample Data

### -- Insert sample data into the 'customers' table

```
INSERT INTO customers (name, email, address)
```

```
VALUES
```

```
('John Doe', 'john.doe@example.com', '123 Elm Street'),
```

```
('Jane Smith', 'jane.smith@example.com', '456 Oak Avenue'),
```

```
('Alice Brown', 'alice.brown@example.com', '789 Maple Drive');
```

### -- Insert sample data into the 'products' table

```
INSERT INTO products (name, price, description)
```

```
VALUES
```

```
('Product A', 25.00, 'Description of Product A'),
```

```
('Product B', 35.00, 'Description of Product B'),
```

```
('Product C', 40.00, 'Description of Product C');
```

### -- Insert sample data into the 'orders' table

```
INSERT INTO orders (customer_id, order_date, total_amount)
```

```
VALUES
```

```
(1, CURDATE(), 60.00), -- Today's date
```

```
(2, CURDATE() - INTERVAL 20 DAY, 100.00), -- 20 days ago
```

```
(3, CURDATE() - INTERVAL 45 DAY, 75.00); -- 45 days ago
```

## Output:

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'ecommerce' selected. The main editor shows a SQL query with three INSERT statements: one for the 'customers' table, one for the 'products' table, and one for the 'orders' table. The 'orders' table insert uses CURDATE() and INTERVAL to calculate dates. The bottom output pane shows the execution results, including a message about a syntax error (Error Code: 1064) and a table of execution statistics.

| #  | Time     | Action   | Message  | Duration / Fetch |
|----|----------|--|--|------------------|
| 5  | 21:11:34 | Create the 'orders' table to store order details   | CREATE TABLE orders ( id INT AUTO_INCREMENT PRIMAR...  | 0.000 sec        |
| 6  | 21:11:57 | CREATE TABLE orders ( id INT AUTO_INCREMENT PRIMARY KEY; -- Unique identifier for each order                     | cu... 0 rows) affected   | 0.047 sec        |
| 7  | 21:14:23 | INSERT INTO orders (customer_id, order_date, total_amount) VALUES (1, CURDATE(), 60.00); -- Today's date         | Error Code: 1452: Cannot add or update a child row: a foreign key constraint fails ('ecommerce'.'orders', CONST... | 0.000 sec        |
| 8  | 21:14:53 | INSERT INTO customers (name, email, address) VALUES ('John Doe', 'john.doe@example.com', '123 Elm Stree...       | 3 rows) affected Records: 3 Duplicates: 0 Warnings: 0  | 0.016 sec        |
| 9  | 21:14:53 | INSERT INTO products (name, price, description) VALUES ('Product A', 25.00, 'Description of Product A'); (Pro... | 3 rows) affected Records: 3 Duplicates: 0 Warnings: 0  | 0.016 sec        |
| 10 | 21:14:53 | INSERT INTO orders (customer_id, order_date, total_amount) VALUES (1, CURDATE(), 60.00); -- Today's date         | 3 rows) affected Records: 3 Duplicates: 0 Warnings: 0  | 0.000 sec        |

## 3. Queries

### a. Retrieve all customers who have placed an order in the last 30 days

-- Select customers who have placed an order within the past 30 days

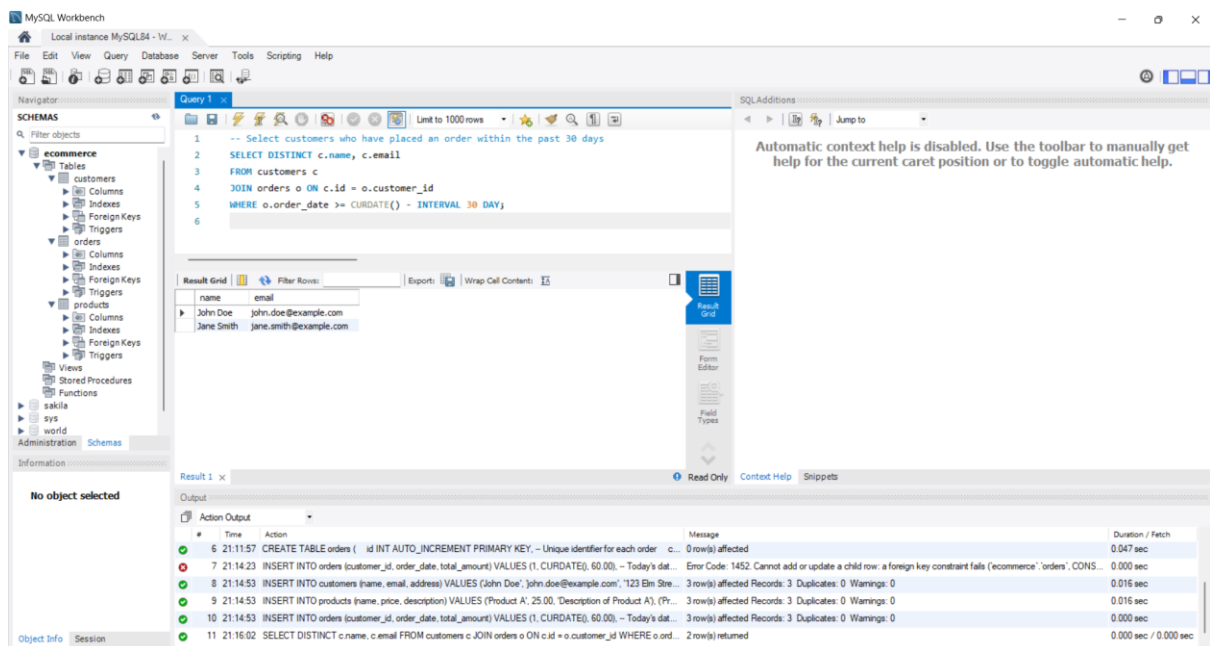
```
SELECT DISTINCT c.name, c.email
```

```
FROM customers c
```

```
JOIN orders o ON c.id = o.customer_id
```

```
WHERE o.order_date >= CURDATE() - INTERVAL 30 DAY;
```

## Output:



## b. Get the total amount of all orders placed by each customer

-- Calculate the total order amount grouped by each customer

SELECT c.name, c.email, SUM(o.total\_amount) AS total\_spent

FROM customers c

JOIN orders o ON c.id = o.customer\_id

GROUP BY c.id;

## Output:

The screenshot shows the MySQL Workbench interface. The 'Query' tab is active, displaying a SQL query that calculates the total order amount for each customer. The 'Result Grid' shows the output of the query, which is a table with three columns: name, email, and total\_spent. The results are as follows:

| name        | email                   | total_spent |
|-------------|-------------------------|-------------|
| John Doe    | john.doe@example.com    | 60.00       |
| Jane Smith  | jane.smith@example.com  | 100.00      |
| Alice Brown | alice.brown@example.com | 75.00       |

The 'Output' tab is also visible, showing a list of actions and their results. The actions include inserting data into the 'customers' and 'products' tables, and selecting data from the 'customers' table. The results show the number of rows affected, duplicates, and warnings for each action.

### c. Update the price of Product C to 45.00

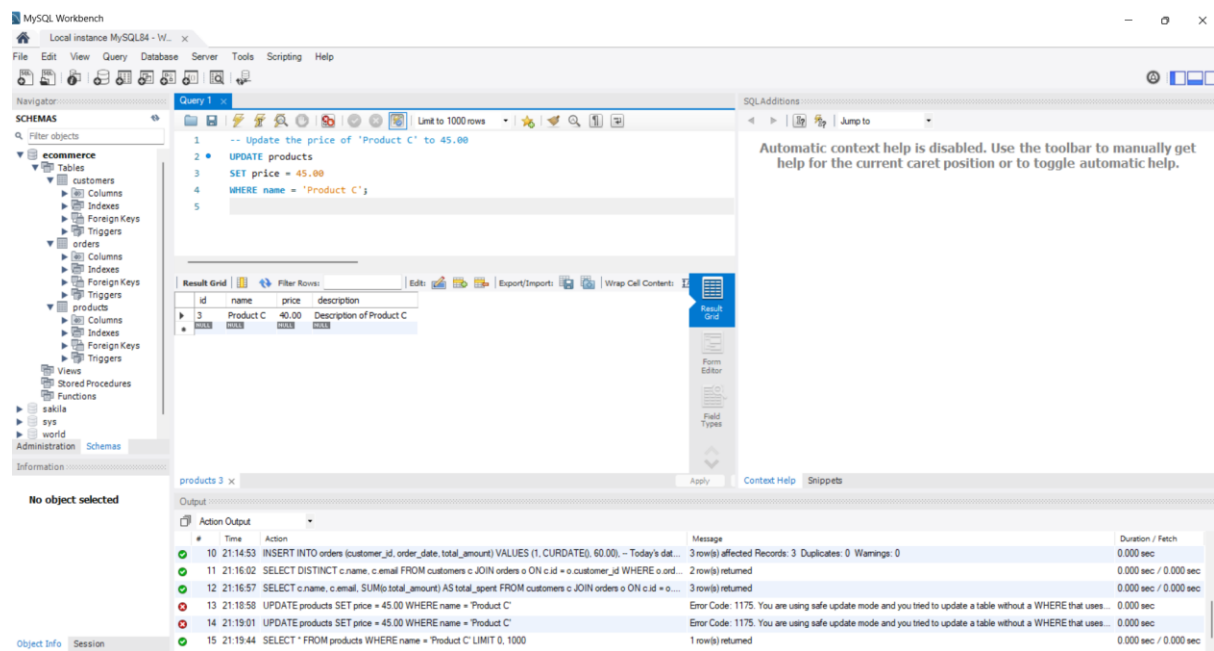
-- Update the price of 'Product C' to 45.00

UPDATE products

SET price = 45.00

WHERE name = 'Product C';

## Output:



## d. Add a new column discount to the products table

-- Add a new column named 'discount' with a default value of 0.00

ALTER TABLE products

ADD COLUMN discount DECIMAL(5, 2) DEFAULT 0.00;

## e. Retrieve the top 3 products with the highest price

-- Fetch the top 3 most expensive products

SELECT name, price

FROM products

ORDER BY price DESC

LIMIT 3;

## Output:

The screenshot shows the MySQL Workbench interface. On the left is the 'Navigator' pane showing a database schema named 'ecommerce' with tables like 'customers', 'orders', 'products', 'sakila', 'sys', and 'world'. The 'products' table is selected. The main editor shows a SQL query: `-- Fetch the top 3 most expensive products`, `SELECT name, price`, `FROM products`, `ORDER BY price DESC`, `LIMIT 3;`. Below the query is the 'Result Grid' showing three rows: Product C (40.00), Product B (35.00), and Product A (25.00). On the right is the 'SQLAdditions' pane with a message: 'Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.' At the bottom is the 'Output' pane showing a table with columns 'Time', 'Action', 'Message', and 'Duration / Fetch'. It contains five rows of log entries for the query execution.

| Time        | Action   | Message           | Duration / Fetch      |
|-------------|--|-------------------|-----------------------|
| 30 21:21:50 | SELECT name, price FROM products ORDER BY price DESC LIMIT 3 | 3 row(s) returned | 0.000 sec / 0.000 sec |
| 31 21:21:51 | SELECT name, price FROM products ORDER BY price DESC LIMIT 3 | 3 row(s) returned | 0.000 sec / 0.000 sec |
| 32 21:21:52 | SELECT name, price FROM products ORDER BY price DESC LIMIT 3 | 3 row(s) returned | 0.000 sec / 0.000 sec |
| 33 21:21:52 | SELECT name, price FROM products ORDER BY price DESC LIMIT 3 | 3 row(s) returned | 0.000 sec / 0.000 sec |
| 34 21:21:52 | SELECT name, price FROM products ORDER BY price DESC LIMIT 3 | 3 row(s) returned | 0.000 sec / 0.000 sec |

## f. Get the names of customers who have ordered Product A

-- Retrieve names of customers who have ordered 'Product A'

SELECT DISTINCT c.name

FROM customers c

JOIN orders o ON c.id = o.customer\_id

JOIN order\_items oi ON o.id = oi.order\_id

JOIN products p ON oi.product\_id = p.id

WHERE p.name = 'Product A';



## Output:

The screenshot shows the MySQL Workbench interface. The 'Query' tab is active, displaying a SQL query that joins the 'customers', 'orders', 'order\_items', and 'products' tables to find the name of the product ordered by customer 'Product A'. The 'Result Grid' shows one row: Product C with a price of 40.00. The 'Output' tab at the bottom shows the execution log, including a warning about a missing table 'ecommerce.order\_items'.

```
2 SELECT DISTINCT c.name
3 FROM customers c
4 JOIN orders o ON c.id = o.customer_id
5 JOIN order_items oi ON o.id = oi.order_id
6 JOIN products p ON oi.product_id = p.id
7 WHERE p.name = 'Product A';
8
```

| id | name      | price | description              |
|----|-----------|-------|--------------------------|
| 3  | Product C | 40.00 | Description of Product C |

| #  | Time     | Action   | Message   | Duration / Fetch      |
|----|----------|--|---|-----------------------|
| 32 | 21:21:52 | SELECT name, price FROM products ORDER BY price DESC LIMIT 3   | 3 row(s) returned   | 0.000 sec / 0.000 sec |
| 33 | 21:21:52 | SELECT name, price FROM products ORDER BY price DESC LIMIT 3   | 3 row(s) returned   | 0.000 sec / 0.000 sec |
| 34 | 21:21:52 | SELECT name, price FROM products ORDER BY price DESC LIMIT 3   | 3 row(s) returned   | 0.000 sec / 0.000 sec |
| 35 | 21:21:53 | SELECT name, price FROM products ORDER BY price DESC LIMIT 3   | 3 row(s) returned   | 0.000 sec / 0.000 sec |
| 36 | 21:23:24 | SELECT DISTINCT c.name FROM customers c JOIN orders o ON c.id = o.customer_id JOIN order_items oi ON o.id = oi.order_id JOIN products p ON oi.product_id = p.id WHERE p.name = 'Product A' | Error Code: 1146. Table 'ecommerce.order_items' doesn't exist | 0.000 sec             |
| 37 | 21:23:40 | SELECT * FROM products WHERE name = 'Product C' LIMIT 0, 1000  | 1 row(s) returned   | 0.000 sec / 0.000 sec |

## g. Join orders and customers tables to retrieve customer name and order date

-- Get the customer's name and the date of each order

SELECT c.name AS customer\_name, o.order\_date

FROM orders o

JOIN customers c ON o.customer\_id = c.id;

## Output:

The screenshot shows the MySQL Workbench interface. The 'Query' tab is active, displaying a SQL query that joins the 'customers' and 'orders' tables to retrieve customer names and order dates. The 'Result Grid' shows three rows of data. The 'Output' tab at the bottom shows the execution log, including a warning about a missing table 'ecommerce.order\_items'.

```
1 -- Get the customer's name and the date of each order
2 SELECT c.name AS customer_name, o.order_date
3 FROM orders o
4 JOIN customers c ON o.customer_id = c.id;
5
```

| customer_name | order_date |
|---------------|------------|
| John Doe      | 2024-11-21 |
| Jane Smith    | 2024-11-01 |
| Alice Brown   | 2024-09-07 |

| #  | Time     | Action   | Message   | Duration / Fetch      |
|----|----------|--|---|-----------------------|
| 33 | 21:21:52 | SELECT name, price FROM products ORDER BY price DESC LIMIT 3   | 3 row(s) returned   | 0.000 sec / 0.000 sec |
| 34 | 21:21:52 | SELECT name, price FROM products ORDER BY price DESC LIMIT 3   | 3 row(s) returned   | 0.000 sec / 0.000 sec |
| 35 | 21:21:53 | SELECT name, price FROM products ORDER BY price DESC LIMIT 3   | 3 row(s) returned   | 0.000 sec / 0.000 sec |
| 36 | 21:23:24 | SELECT DISTINCT c.name FROM customers c JOIN orders o ON c.id = o.customer_id JOIN order_items oi ON o.id = oi.order_id JOIN products p ON oi.product_id = p.id WHERE p.name = 'Product A' | Error Code: 1146. Table 'ecommerce.order_items' doesn't exist | 0.000 sec             |
| 37 | 21:23:40 | SELECT * FROM products WHERE name = 'Product C' LIMIT 0, 1000  | 1 row(s) returned   | 0.000 sec / 0.000 sec |
| 38 | 21:24:54 | SELECT c.name AS customer_name, o.order_date FROM orders o JOIN customers c ON o.customer_id = c.id  | 3 row(s) returned   | 0.000 sec / 0.000 sec |

## h. Retrieve orders with a total amount greater than 150.00

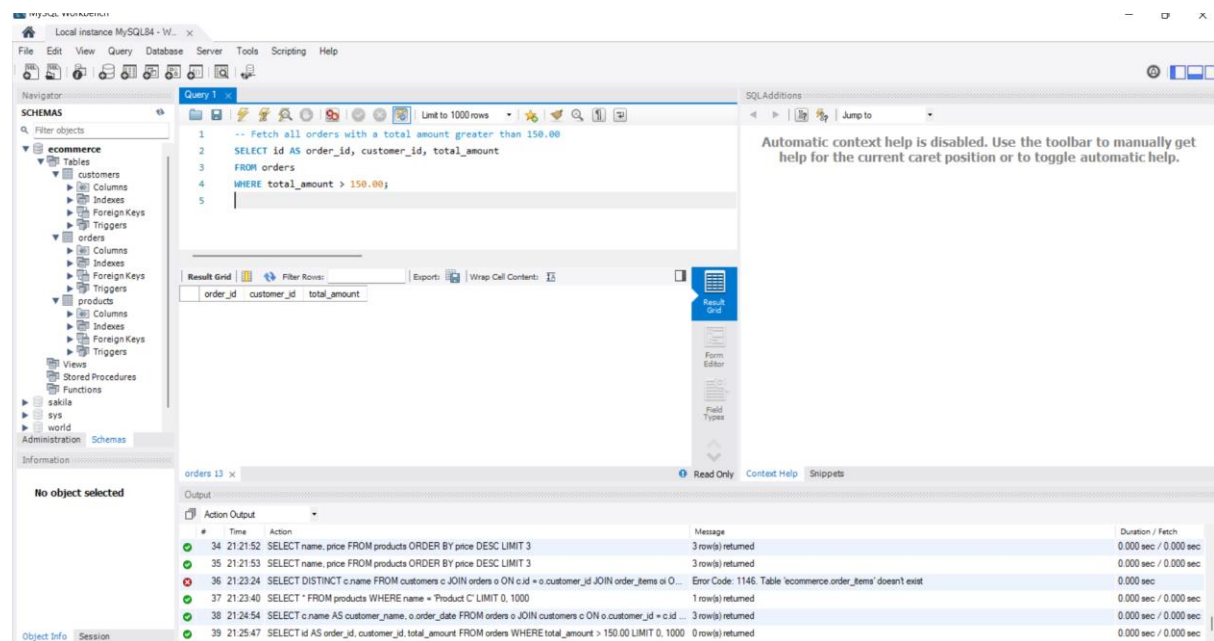
-- Fetch all orders with a total amount greater than 150.00

```
SELECT id AS order_id, customer_id, total_amount
```

```
FROM orders
```

```
WHERE total_amount > 150.00;
```

### Output:



## i. Normalize the database (add order\_items table)

-- Create the 'order\_items' table to store details of individual items in an order

```
CREATE TABLE order_items (
```

```
    id INT AUTO_INCREMENT PRIMARY KEY, -- Unique identifier for each order item
```

```
    order_id INT NOT NULL, -- ID of the order
```

```
    product_id INT NOT NULL, -- ID of the product
```

```
    quantity INT NOT NULL, -- Quantity of the product ordered
```

```
    FOREIGN KEY (order_id) REFERENCES orders(id), -- Link to the 'orders' table
```

```
    FOREIGN KEY (product_id) REFERENCES products(id) -- Link to the 'products' table
```

```
);
```

-- Remove the 'total\_amount' column from the 'orders' table to calculate it dynamically

```
ALTER TABLE orders DROP COLUMN total_amount;
```

-- Insert sample data into the 'order\_items' table

```
INSERT INTO order_items (order_id, product_id, quantity)
```

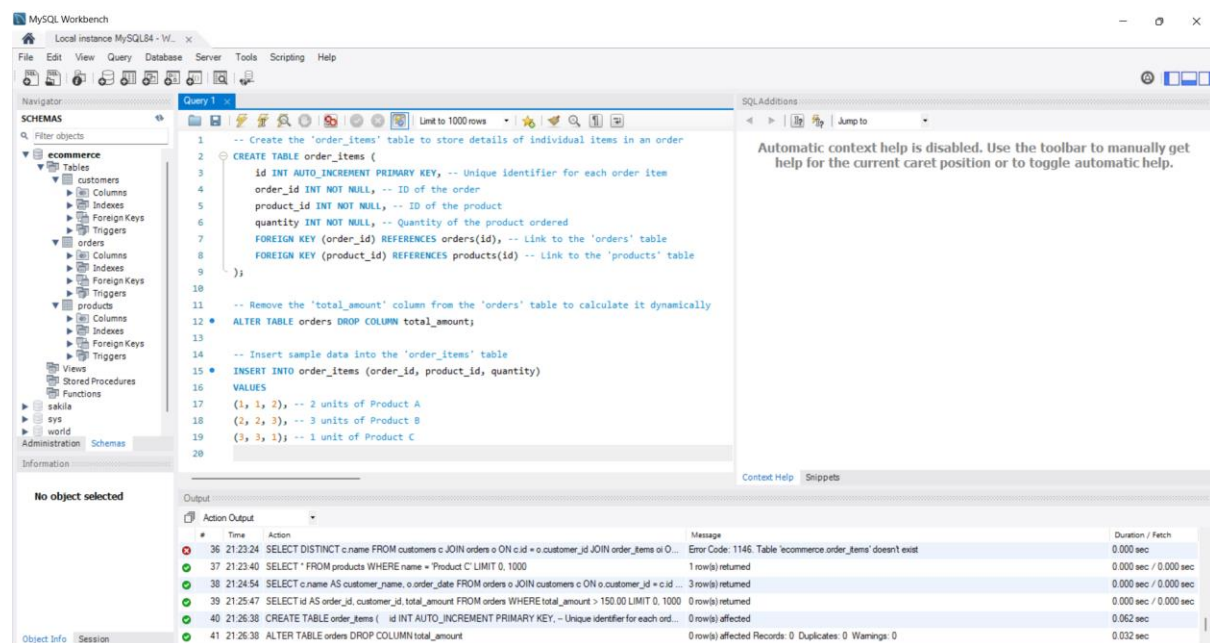
```
VALUES
```

```
(1, 1, 2), -- 2 units of Product A
```

```
(2, 2, 3), -- 3 units of Product B
```

```
(3, 3, 1); -- 1 unit of Product C
```

## Output:



## j. Retrieve the average total of all orders

-- Calculate the average total amount for all orders dynamically

```
SELECT AVG(oi.quantity * p.price) AS average_order_total
```

```
FROM orders o
```

```
JOIN order_items oi ON o.id = oi.order_id
```

```
JOIN products p ON oi.product_id = p.id;
```

## Output:

The screenshot displays the MySQL Workbench interface. On the left, the 'SCHEMAS' pane shows a tree view of the 'ecommerce' database, including tables like 'customers', 'orders', 'products', and 'order\_items'. The main 'Query Editor' window contains the following SQL query:

```
1 -- Calculate the average total amount for all orders dynamically
2 SELECT AVG(oi.quantity * p.price) AS average_order_total
3 FROM orders o
4 JOIN order_items oi ON o.id = oi.order_id
5 JOIN products p ON oi.product_id = p.id;
```

Below the query editor, the 'Result Grid' shows a single column named 'average\_order\_total' with a value of 100.00. To the right, the 'SQLAdditions' pane displays a message: 'Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.'

At the bottom, the 'Output' pane shows the 'Action Output' tab with a log of database actions:

| #  | Time     | Action  | Message  | Duration / Fetch      |
|----|----------|---|--|-----------------------|
| 38 | 21:24:54 | SELECT c.name AS customer_name, o.order_date FROM orders o JOIN customers c ON o.customer_id = c.id ... | 3 row(s) returned                                      | 0.000 sec / 0.000 sec |
| 39 | 21:25:47 | SELECT id AS order_id, customer_id, total_amount FROM orders WHERE total_amount > 150.00 LIMIT 0, 1000  | 0 row(s) returned                                      | 0.000 sec / 0.000 sec |
| 40 | 21:26:38 | CREATE TABLE order_items ( id INT AUTO_INCREMENT PRIMARY KEY, -- Unique identifier for each ord...      | 0 row(s) affected                                      | 0.062 sec             |
| 41 | 21:26:38 | ALTER TABLE orders DROP COLUMN total_amount   | 0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0 | 0.032 sec             |