Sql task

e-commerce

Create a database named ecommerce.

```
-- Create the database
    CREATE DATABASE ecommerce;
    USE ecommerce;
```

Create three tables: customers, orders, and products.

```
-- Create the customers table
    CREATE TABLE customers (
        id INT AUTO_INCREMENT PRIMARY KEY,
        name VARCHAR(255) NOT NULL,
        email VARCHAR(255) UNIQUE NOT NULL,
        address TEXT NOT NULL
    );
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
name	varchar(255)	NO		NULL	
email	varchar(255)	NO	UNI	NULL	
address	text	NO		NULL	

```
-- Create the orders table
CREATE TABLE orders (
```

```
    id INT AUTO_INCREMENT PRIMARY KEY,
    customer_id INT NOT NULL,
    order_date DATE NOT NULL,
    total_amount DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (customer_id) REFERENCES customers(id)
);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
customer_id	int	NO	MUL	NULL	
order_date	date	NO		NULL	
total_amount	decimal(10,2)	NO		NULL	

```
-- Create the products table
CREATE TABLE products (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    price DECIMAL(10, 2) NOT NULL,
    description TEXT
);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
name	varchar(255)	NO		NULL	
price	decimal(10,2)	NO		NULL	
description	text	YES		NULL	
discount	decimal(5,2)	YES		0.00	

Result 5

Table Structure:

-- Insert sample data into customers table

INSERT INTO customers (name, email, address) VALUES

('Alice Johnson', 'alice@example.com', '123 Maple Street'),

('Bob Smith', 'bob@example.com', '456 Oak Avenue'),

('Catherine Lee', 'catherine@example.com', '789 Pine Lane');

Result Grid | Filter Rows: | Edit: | Export/Import:

id	name	email	address
1	Alice Johnson	alice@example.com	123 Maple Street
2	Bob Smith	bob@example.com	456 Oak Avenue
3	Catherine Lee	catherine@example.com	789 Pine Lane
NULL	NULL	NULL	NULL

customers 6

-- Insert sample data into products table

INSERT INTO products (name, price, description) VALUES

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

('Product B', 30.00, 'Description of Product B'),

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

Result Grid | Filter Rows: | Edit: | Export/Impo

id	name	price	description	discount
1	Product A	25.00	Description of Product A	0.00
2	Product B	30.00	Description of Product B	0.00
3	Product C	45.00	Description of Product C	0.00
NULL	NULL	NULL	NULL	NULL

products 7

-- Insert sample data into orders table

INSERT INTO orders (customer_id, order_date, total_amount) VALUES

(1, CURDATE(), 55.00),

(2, CURDATE() - INTERVAL 10 DAY, 30.00),

(1, CURDATE() - INTERVAL 20 DAY, 100.00);

Result Grid | Filter Rows: | Edit:

id	customer_id	order_date	total_amount
1	1	2024-12-02	55.00
2	2	2024-11-22	30.00
3	1	2024-11-12	100.00
NULL	NULL	NULL	NULL

orders 8

Queries

-- Query 1: Retrieve all customers who have placed an order in the last 30 days

SELECT DISTINCT c.*

FROM customers c

```
JOIN orders o ON c.id = o.customer_id
WHERE o.order_date >= CURDATE() - INTERVAL 30 DAY;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	name	email	address
1	Alice Johnson	alice@example.com	alice@example.com
2	Bob Smith	bob@example.com	456 Oak Avenue

Result 9

```
-- Query 2: Get the total amount of all orders placed by each customer
SELECT c.name, SUM(o.total_amount) AS total_spent
FROM customers c
JOIN orders o ON c.id = o.customer_id
GROUP BY c.id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name	total_spent
Alice Johnson	155.00
Bob Smith	30.00

Result 10

```
-- Query 3: Update the price of Product C to 45.00
UPDATE products
SET price = 45.00
WHERE name = 'Product C';
```

* id int	* name varchar(255)	* price decimal(10,2)	description text	discount decimal(5,2)
Filter	Filter	Filter	Filter	Filter
3	Product C	45.00	Description of Product C	0.00
2	Product B	30.00	Description of Product B	0.00
1	Product A	25.00	Description of Product A	0.00

-- Query 4: Add a new column discount to the products table

ALTER TABLE products

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

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	name	price	description	discount
3	Product C	45.00	Description of Product C	0.00
2	Product B	30.00	Description of Product B	0.00
1	Product A	25.00	Description of Product A	0.00
NULL	NULL	NULL	NULL	NULL

products 16 ×

-- Query 5: Retrieve the top 3 products with the highest price

SELECT *

FROM products

ORDER BY price DESC

LIMIT 3;

Result Grid | Filter Rows: | Edit: | Export/Import:

id	name	price	description	discount
3	Product C	45.00	Description of Product C	0.00
2	Product B	30.00	Description of Product B	0.00
1	Product A	25.00	Description of Product A	0.00
NULL	NULL	NULL	NULL	NULL

products 11 ×

-- Query 6: Get the 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';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name

Result 12

Output

Action Output

#	Time	Action	Message
18	01:10:26	SELECT * FROM products ORDER BY price DESC LIMIT 3	3 row(s) returned
19	01:11:32	SELECT DISTINCT c.name FROM customers c JOIN orders o ON c.id = o.custome...	0 row(s) returned

-- Query 7: Join the orders and customers tables to retrieve the customer's name and order date for each order

```
SELECT c.name, o.order_date
FROM customers c
JOIN orders o ON c.id = o.customer_id;
```

Result Grid | Filter Rows: | Export:

name	order_date
Alice Johnson	2024-12-02
Bob Smith	2024-11-22
Alice Johnson	2024-11-12

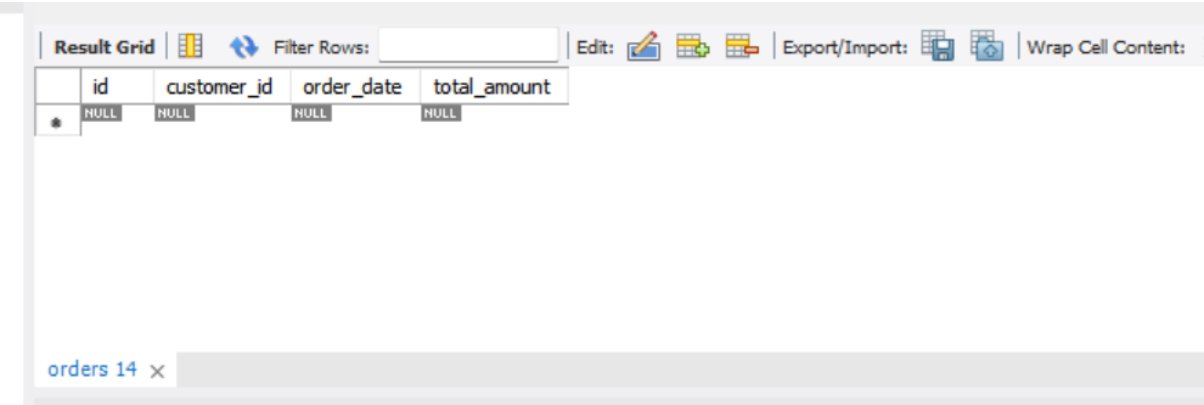
Result 13

-- Query 8: Retrieve the orders with a total amount greater than 150.00

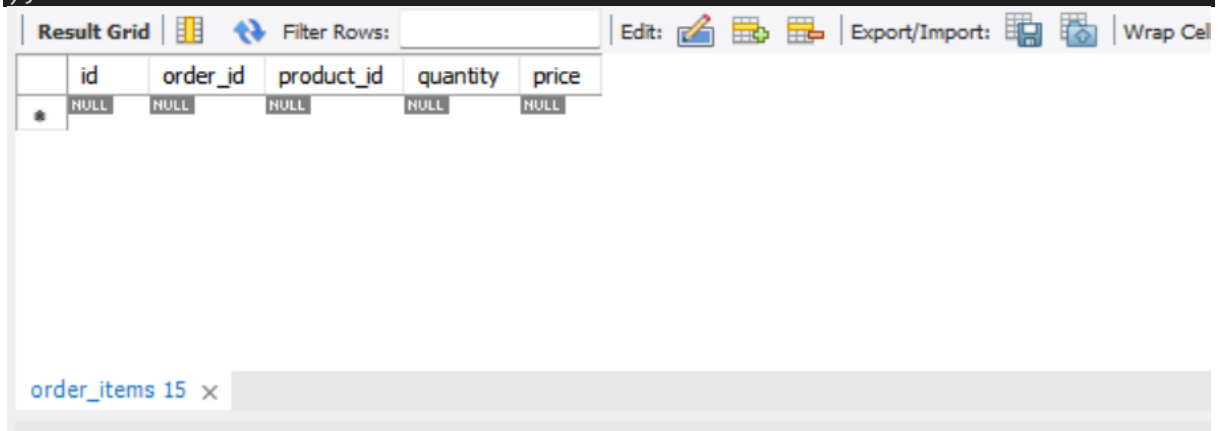
```
SELECT *
```

FROM orders

WHERE total_amount > 150.00;




```
-- Query 9: Create order_items table and update orders table
CREATE TABLE order_items (
    id INT AUTO_INCREMENT PRIMARY KEY,
    order_id INT NOT NULL,
    product_id INT NOT NULL,
    quantity INT NOT NULL,
    price DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (order_id) REFERENCES orders(id),
    FOREIGN KEY (product_id) REFERENCES products(id)
);
```




```
-- Query 10: Retrieve the average total of all orders
SELECT AVG(total_amount) AS average_order_total
FROM (
    SELECT SUM(oi.quantity * oi.price) AS total_amount
    FROM order_items oi
    GROUP BY oi.order_id
) AS order_totals;
```

average_order_total decimal
Filter
(NULL)