```
CREATE TABLE customers (
  customer_id INTEGER PRIMARY KEY,
  name TEXT,
  email TEXT
);

CREATE TABLE products (
  product_id INTEGER PRIMARY KEY,
  name TEXT,
  category TEXT,
  price REAL
);

CREATE TABLE orders (
  order_id INTEGER PRIMARY KEY,
  customer_id INTEGER,
  order_date TEXT,
  FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
);

CREATE TABLE order_items (
  order_item_id INTEGER PRIMARY KEY,
  order_id INTEGER,
  product_id INTEGER,
  quantity INTEGER,
  FOREIGN KEY (order_id) REFERENCES orders(order_id),
  FOREIGN KEY (product_id) REFERENCES products(product_id)
);

##Insert Sample Data

INSERT INTO customers (name, email) VALUES ('Alice', 'alice@example.com');
INSERT INTO customers (name, email) VALUES ('Bob', 'bob@example.com');

INSERT INTO products (name, category, price) VALUES ('Laptop', 'Electronics', 1000);
INSERT INTO products (name, category, price) VALUES ('Mouse', 'Electronics', 25);
INSERT INTO products (name, category, price) VALUES ('Book', 'Stationery', 15);

INSERT INTO orders (customer_id, order_date) VALUES (1, '2024-05-01');
INSERT INTO orders (customer_id, order_date) VALUES (2, '2024-05-02');

INSERT INTO order_items (order_id, product_id, quantity) VALUES (1, 1, 1);
INSERT INTO order_items (order_id, product_id, quantity) VALUES (1, 2, 2);
INSERT INTO order_items (order_id, product_id, quantity) VALUES (2, 3, 3);


#Query 1: Products with price > 20
SELECT * FROM products WHERE price > 20;

# Query 2: Inner Join to show order details
SELECT o.order_id, c.name, p.name AS product, oi.quantity
FROM orders o
JOIN customers c ON o.customer_id = c.customer_id
JOIN order_items oi ON o.order_id = oi.order_id
JOIN products p ON oi.product_id = p.product_id;

# Query 3: Total amount spent by each customer
SELECT c.name, SUM(p.price * oi.quantity) AS total_spent
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
```

```
JOIN order_items oi ON o.order_id = oi.order_id
JOIN products p ON oi.product_id = p.product_id
GROUP BY c.customer_id;

#Query 4: Subquery to list products priced above average
SELECT name FROM products
WHERE price > (SELECT AVG(price) FROM products);

#Query 5: Create View
CREATE VIEW customer_totals AS
SELECT c.name, SUM(p.price * oi.quantity) AS total_spent
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
JOIN order_items oi ON o.order_id = oi.order_id
JOIN products p ON oi.product_id = p.product_id
GROUP BY c.customer_id;

#Query 6: Use the View
SELECT * FROM customer_totals ORDER BY total_spent DESC;
```

```
sqlite> -- Customers table
sqlite> CREATE TABLE customers (
(x1...>   customer_id INTEGER PRIMARY KEY,
(x1...>   name TEXT,
(x1...>   email TEXT
(x1...> );
sqlite>
sqlite> -- Products table
sqlite> CREATE TABLE products (
(x1...>   product_id INTEGER PRIMARY KEY,
(x1...>   name TEXT,
(x1...>   category TEXT,
(x1...>   price REAL
(x1...> );
sqlite>
sqlite> -- Orders table
sqlite> CREATE TABLE orders (
(x1...>   order_id INTEGER PRIMARY KEY,
(x1...>   customer_id INTEGER,
(x1...>   order_date TEXT,
(x1...>   FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
(x1...> );
sqlite>
sqlite> -- Order Items table
sqlite> CREATE TABLE order_items (
(x1...>   order_item_id INTEGER PRIMARY KEY,
(x1...>   order_id INTEGER,
(x1...>   product_id INTEGER,
(x1...>   quantity INTEGER,
(x1...>   FOREIGN KEY (order_id) REFERENCES orders(order_id),
(x1...>   FOREIGN KEY (product_id) REFERENCES products(product_id)
(x1...> );
sqlite> -- Customers
sqlite> INSERT INTO customers (name, email) VALUES ('Alice', 'alice@example.com');
sqlite> INSERT INTO customers (name, email) VALUES ('Bob', 'bob@example.com');
sqlite>
sqlite> -- Products
sqlite> INSERT INTO products (name, category, price) VALUES ('Laptop', 'Electronics', 1000);
sqlite> INSERT INTO products (name, category, price) VALUES ('Mouse', 'Electronics', 25);
sqlite> INSERT INTO products (name, category, price) VALUES ('Book', 'Stationery', 15);
sqlite>
sqlite> -- Orders
sqlite> INSERT INTO orders (customer_id, order_date) VALUES (1, '2024-05-01');
sqlite> INSERT INTO orders (customer_id, order_date) VALUES (2, '2024-05-02');
sqlite>
sqlite> -- Order Items
sqlite> INSERT INTO order_items (order_id, product_id, quantity) VALUES (1, 1, 1);
sqlite> INSERT INTO order_items (order_id, product_id, quantity) VALUES (1, 2, 2);
sqlite> INSERT INTO order_items (order_id, product_id, quantity) VALUES (2, 3, 3);
sqlite> SELECT * FROM products WHERE price > 20;
1|Laptop|Electronics|1000.0
2|Mouse|Electronics|25.0
sqlite> SELECT o.order_id, c.name, p.name AS product, oi.quantity
   ...> FROM orders o
   ...> JOIN customers c ON o.customer_id = c.customer_id
   ...> JOIN order_items oi ON o.order_id = oi.order_id
   ...> JOIN products p ON oi.product_id = p.product_id;
1|Alice|Laptop|1
1|Alice|Mouse|2
```

```
sqlite> CREATE TABLE order_items (
(x1...>   order_item_id INTEGER PRIMARY KEY,
(x1...>   order_id INTEGER,
(x1...>   product_id INTEGER,
(x1...>   quantity INTEGER,
(x1...>   FOREIGN KEY (order_id) REFERENCES orders(order_id),
(x1...>   FOREIGN KEY (product_id) REFERENCES products(product_id)
(x1...> );
sqlite> -- Customers
sqlite> INSERT INTO customers (name, email) VALUES ('Alice', 'alice@example.com');
sqlite> INSERT INTO customers (name, email) VALUES ('Bob', 'bob@example.com');
sqlite>
sqlite> -- Products
sqlite> INSERT INTO products (name, category, price) VALUES ('Laptop', 'Electronics', 1000);
sqlite> INSERT INTO products (name, category, price) VALUES ('Mouse', 'Electronics', 25);
sqlite> INSERT INTO products (name, category, price) VALUES ('Book', 'Stationery', 15);
sqlite>
sqlite> -- Orders
sqlite> INSERT INTO orders (customer_id, order_date) VALUES (1, '2024-05-01');
sqlite> INSERT INTO orders (customer_id, order_date) VALUES (2, '2024-05-02');
sqlite>
sqlite> -- Order Items
sqlite> INSERT INTO order_items (order_id, product_id, quantity) VALUES (1, 1, 1);
sqlite> INSERT INTO order_items (order_id, product_id, quantity) VALUES (1, 2, 2);
sqlite> INSERT INTO order_items (order_id, product_id, quantity) VALUES (2, 3, 3);
sqlite> SELECT * FROM products WHERE price > 20;
1|Laptop|Electronics|1000.0
2|Mouse|Electronics|25.0
sqlite> SELECT o.order_id, c.name, p.name AS product, oi.quantity
   ...> FROM orders o
   ...> JOIN customers c ON o.customer_id = c.customer_id
   ...> JOIN order_items oi ON o.order_id = oi.order_id
   ...> JOIN products p ON oi.product_id = p.product_id;
1|Alice|Laptop|1
1|Alice|Mouse|2
2|Bob|Book|3
sqlite> SELECT c.name, SUM(p.price * oi.quantity) AS total_spent
   ...> FROM customers c
   ...> JOIN orders o ON c.customer_id = o.customer_id
   ...> JOIN order_items oi ON o.order_id = oi.order_id
   ...> JOIN products p ON oi.product_id = p.product_id
   ...> GROUP BY c.customer_id;
Alice|1050.0
Bob|45.0
sqlite> SELECT name FROM products
   ...> WHERE price > (SELECT AVG(price) FROM products);
Laptop
sqlite> CREATE VIEW customer_totals AS
   ...> SELECT c.name, SUM(p.price * oi.quantity) AS total_spent
   ...> FROM customers c
   ...> JOIN orders o ON c.customer_id = o.customer_id
   ...> JOIN order_items oi ON o.order_id = oi.order_id
   ...> JOIN products p ON oi.product_id = p.product_id
   ...> GROUP BY c.customer_id;
sqlite> SELECT * FROM customer_totals ORDER BY total_spent DESC;
Alice|1050.0
Bob|45.0
sqlite> sqlite3 ecommerce.db < queries.sql > output.txt
   ...>
```