Bike Store Database Analysis

Overview

The purpose of this project is to analyze the **Bike Store** database to gain insights into sales performance, customer behavior, and inventory management. This analysis helps in understanding key business metrics such as total sales, top-selling products, store performance, and customer demographics. By utilizing SQL queries, we explore the dataset to derive actionable insights that can aid in decision-making and business growth.

Dataset Description:

- **Tables Used:** Brands, Categories, Customers, Order_Items, Orders, Products, Staffs, Stocks, Stores.
- **Source:** Kaggle Dataset
- **Link:** https://www.kaggle.com/datasets/dillonmyrick/bike-store-sample-database

Key objectives of this project include:

- Understanding overall sales performance and trends.
- Identifying top-selling products, brands, and categories.
- Analyzing store-wise and location-based sales performance.
- Evaluating customer purchase patterns and top customers.
- Assessing employee sales contributions and store staff efficiency.
- Monitoring stock levels to ensure effective inventory management.

By leveraging SQL for data retrieval and analysis, this project provides a structured approach to understanding business performance and optimizing operations.

1. Database Creation & Initial Exploration

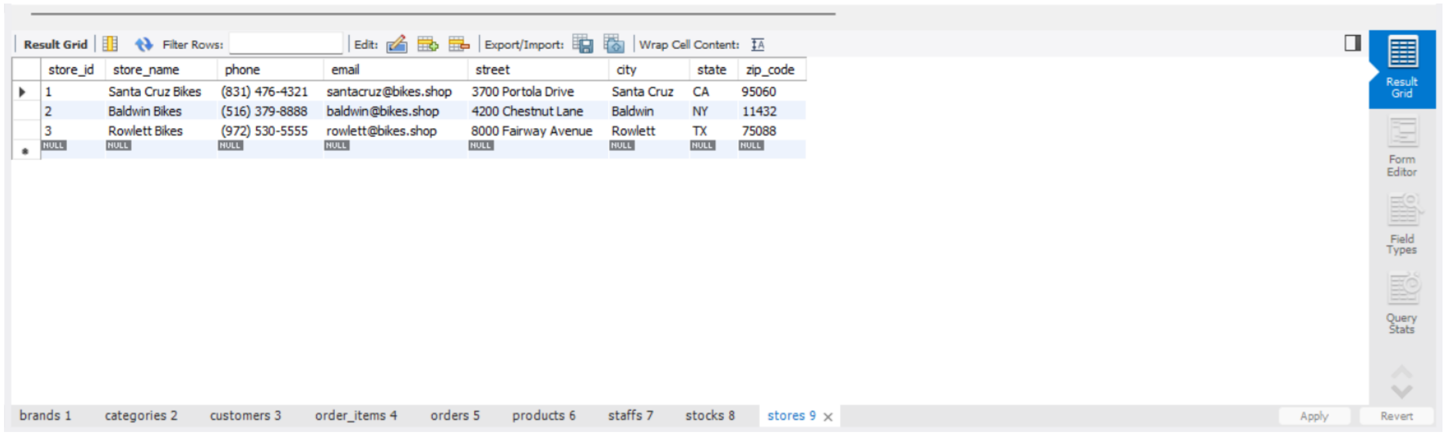
Database Creation

```
CREATE DATABASE Bike_Store;
USE Bike_Store;
```

Viewing All Tables and Their Data

```
SELECT * FROM brands;
SELECT * FROM categories;
SELECT * FROM customers;
SELECT * FROM order_items;
SELECT * FROM orders;
SELECT * FROM products;
SELECT * FROM staffs;
SELECT * FROM stocks;
SELECT * FROM stores;
```

OUTPUT



store_id	store_name	phone	email	street	city	state	zip_code
1	Santa Cruz Bikes	(831) 476-4321	santacruz@bikes.shop	3700 Portola Drive	Santa Cruz	CA	95060
2	Baldwin Bikes	(516) 379-8888	baldwin@bikes.shop	4200 Chestnut Lane	Baldwin	NY	11432
3	Rowlett Bikes	(972) 530-5555	rowlett@bikes.shop	8000 Fairway Avenue	Rowlett	TX	75088
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

2. Data Cleaning & Data Type Correction

Modifying Date Columns

```
ALTER TABLE orders
MODIFY COLUMN order_date DATE,
MODIFY COLUMN shipped_date DATE,
MODIFY COLUMN required_date DATE;
```

3. Assigning Primary Keys

```
ALTER TABLE brands ADD PRIMARY KEY (brand_id);
ALTER TABLE categories ADD PRIMARY KEY (category_id);
ALTER TABLE customers ADD PRIMARY KEY (customer_id);
ALTER TABLE orders ADD PRIMARY KEY (order_id);
ALTER TABLE products ADD PRIMARY KEY (product_id);
ALTER TABLE staffs ADD PRIMARY KEY (staff_id);
ALTER TABLE stores ADD PRIMARY KEY (store_id);
```

4. Establishing Foreign Key Relationships

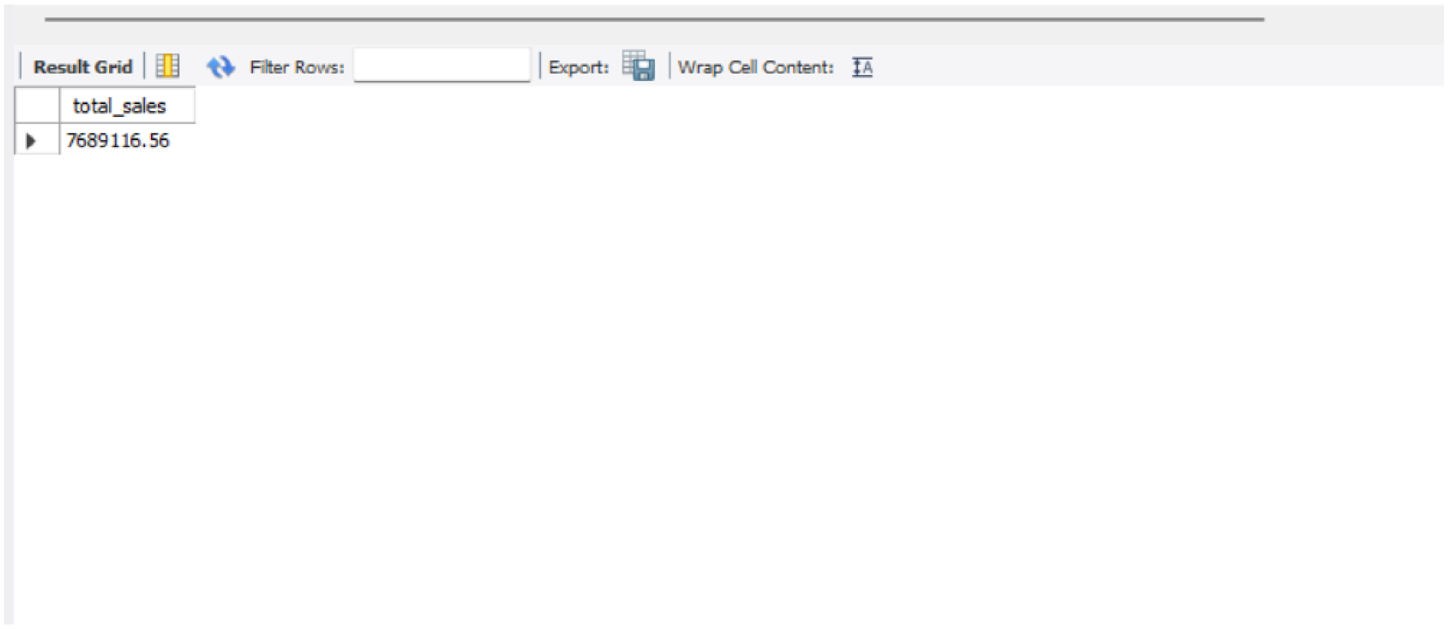
```
ALTER TABLE order_items
ADD CONSTRAINT fk_order FOREIGN KEY (order_id) REFERENCES orders(order_id),
ADD CONSTRAINT fk_product FOREIGN KEY (product_id) REFERENCES products(product_id);

ALTER TABLE orders
ADD CONSTRAINT fk_customer FOREIGN KEY (customer_id) REFERENCES customers(customer_id),
ADD CONSTRAINT fk_stores FOREIGN KEY (store_id) REFERENCES stores(store_id),
ADD CONSTRAINT fk_staff FOREIGN KEY (staff_id) REFERENCES staffs(staff_id);

ALTER TABLE products
ADD CONSTRAINT fk_brand FOREIGN KEY (brand_id) REFERENCES brands(brand_id),
ADD CONSTRAINT fk_category FOREIGN KEY (category_id) REFERENCES categories(category_id);

ALTER TABLE staffs
ADD FOREIGN KEY (store_id) REFERENCES stores(store_id);

ALTER TABLE stocks
ADD FOREIGN KEY fK_store (store_id) REFERENCES stores(store_id),
ADD FOREIGN KEY fK_product (product_id) REFERENCES products(product_id);
```

5. Business Questions & Solutions

1. Total Sales Amount

```
SELECT ROUND(SUM(quantity * list_price * (1 - discount)),2) AS total_sales FROM order_items;
```

OUTPUT

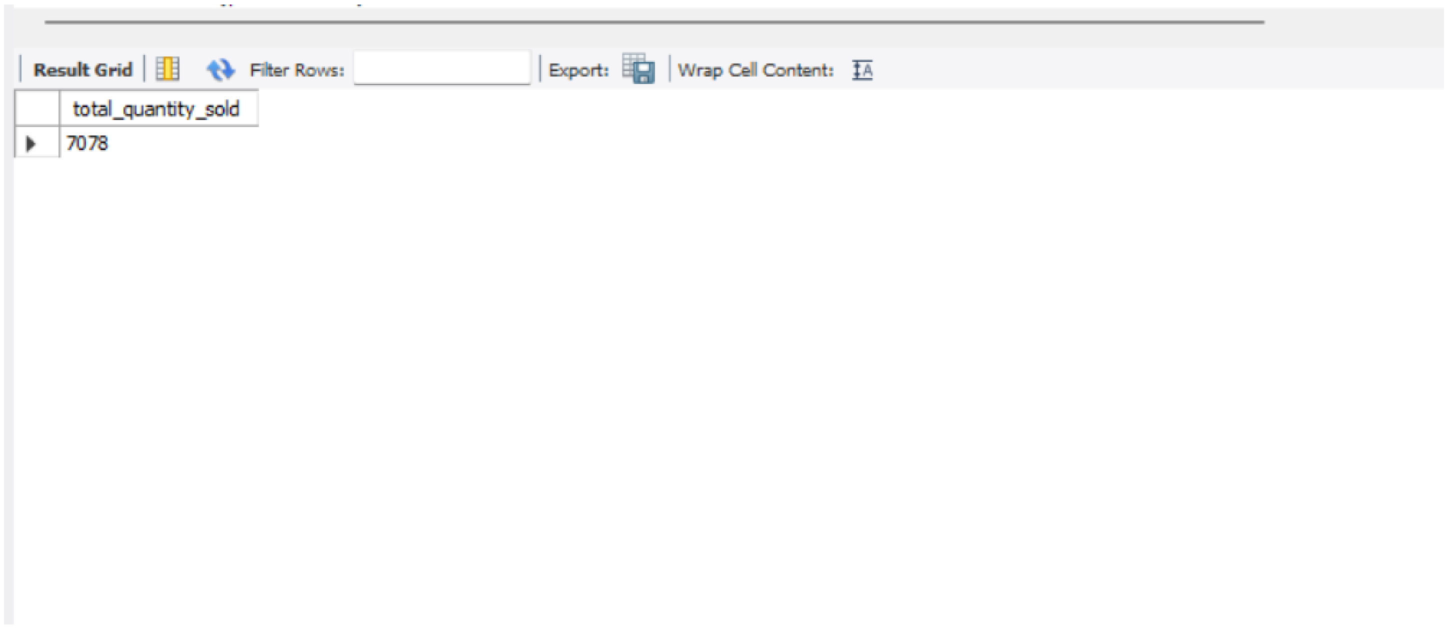


```
total_sales
7689116.56
```

2. Total Quantity Sold

```
SELECT SUM(quantity) AS total_quantity_sold FROM order_items;
```

OUTPUT

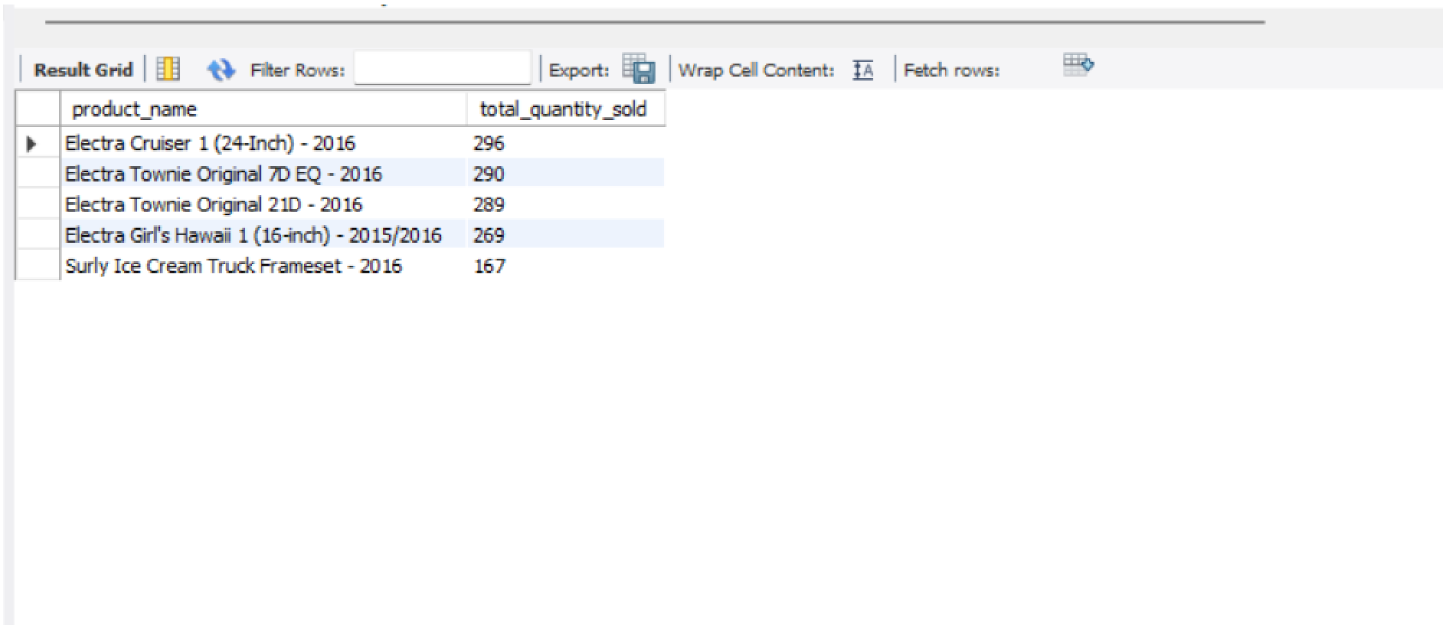


```
total_quantity_sold
7078
```

3. Top 5 Most Popular Products by Quantity Sold

```
SELECT p.product_name, SUM(oi.quantity) AS total_quantity_sold
FROM order_items oi
JOIN products p USING (product_id)
GROUP BY p.product_name
ORDER BY total_quantity_sold DESC
LIMIT 5;
```

OUTPUT

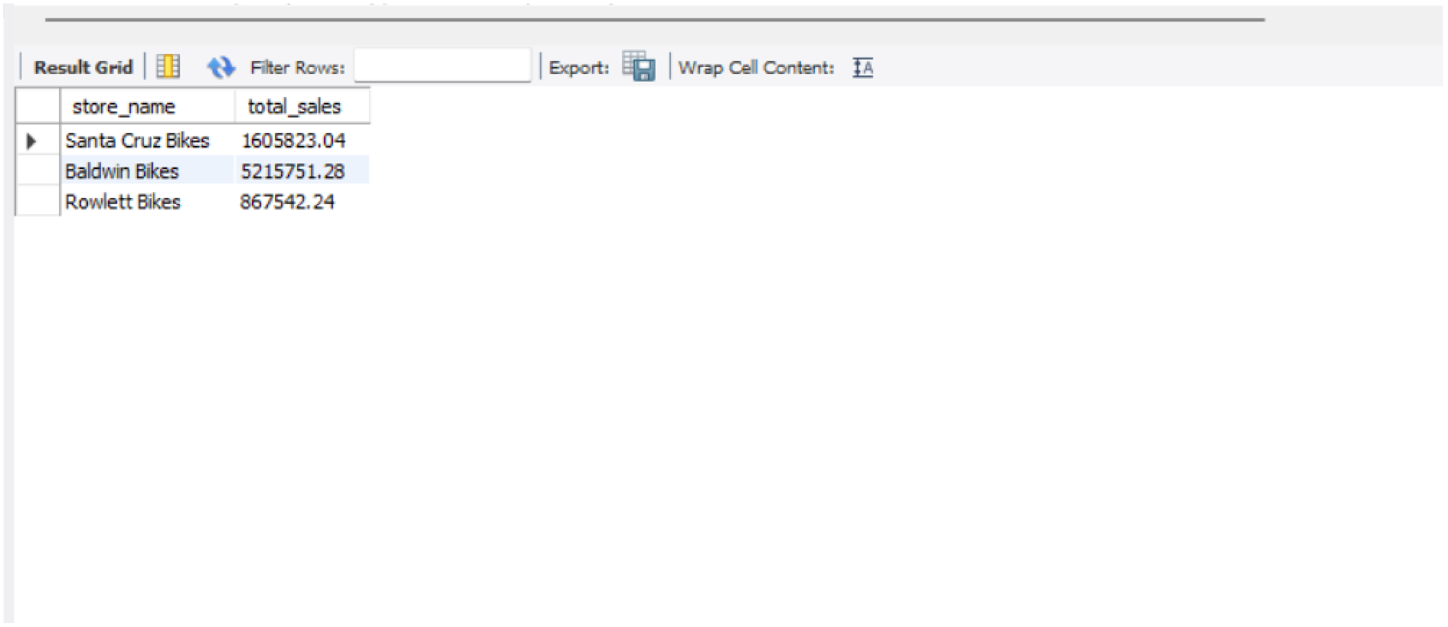


product_name	total_quantity_sold
Electra Cruiser 1 (24-Inch) - 2016	296
Electra Townie Original 7D EQ - 2016	290
Electra Townie Original 21D - 2016	289
Electra Girl's Hawaii 1 (16-inch) - 2015/2016	269
Surly Ice Cream Truck Frameset - 2016	167

4. Total Sales by Store

```
SELECT s.store_name, ROUND(SUM(oi.quantity * oi.list_price * (1 - oi.discount)),2) AS total_sales
FROM order_items oi
JOIN orders o USING (order_id)
JOIN stores s USING (store_id)
GROUP BY s.store_name;
```

OUTPUT

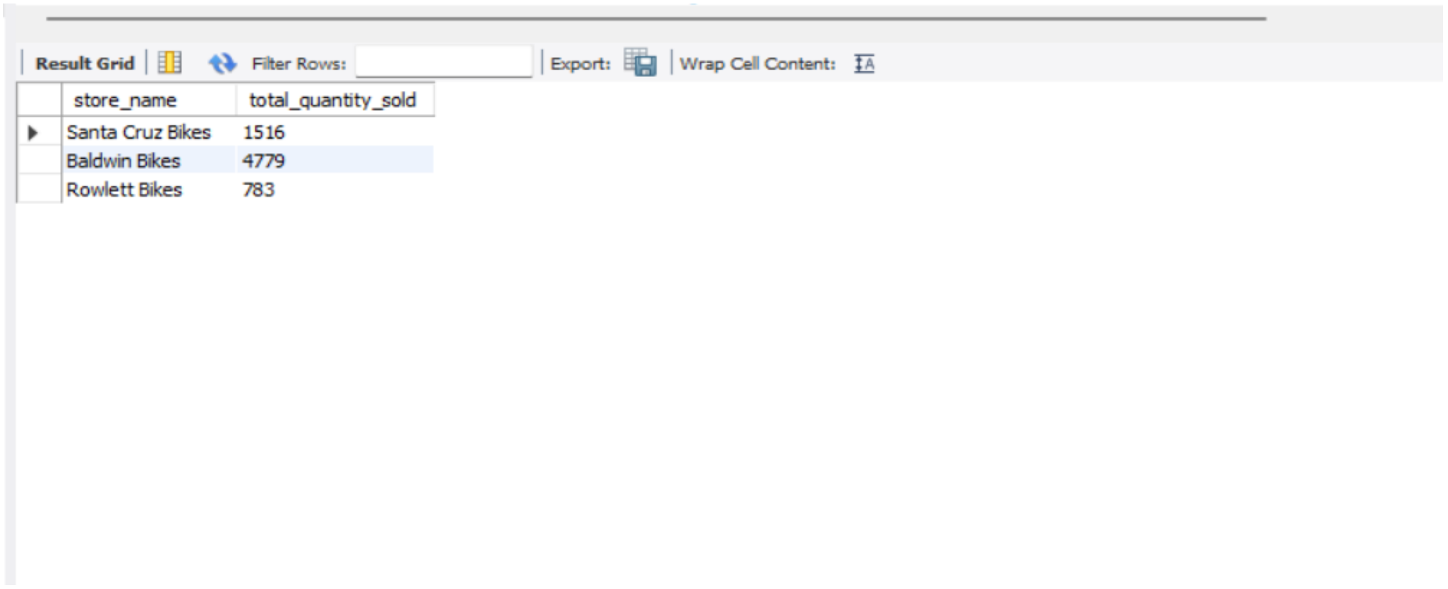


store_name	total_sales
Santa Cruz Bikes	1605823.04
Baldwin Bikes	5215751.28
Rowlett Bikes	867542.24

5. Total Quantity Sold by Store

```
SELECT s.store_name, SUM(oi.quantity) AS total_quantity_sold
FROM order_items oi
JOIN orders o USING (order_id)
JOIN stores s USING (store_id)
GROUP BY s.store_name;
```

OUTPUT

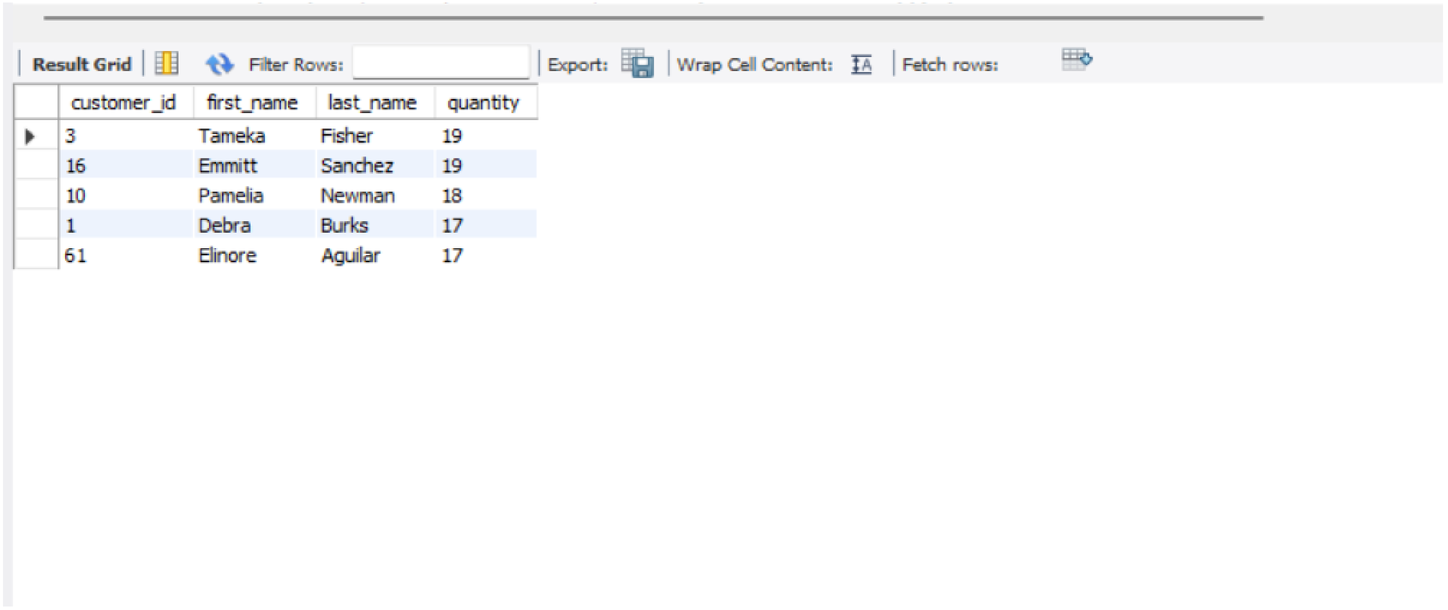


store_name	total_quantity_sold
Santa Cruz Bikes	1516
Baldwin Bikes	4779
Rowlett Bikes	783

6. Top 5 Customers by Quantity Purchased

```
SELECT c.customer_id, c.first_name, c.last_name, SUM(oi.quantity) AS quantity
FROM order_items oi
JOIN orders o ON oi.order_id = o.order_id
JOIN customers c ON o.customer_id = c.customer_id
GROUP BY c.customer_id, c.first_name, c.last_name
ORDER BY quantity DESC
LIMIT 5;
```

OUTPUT

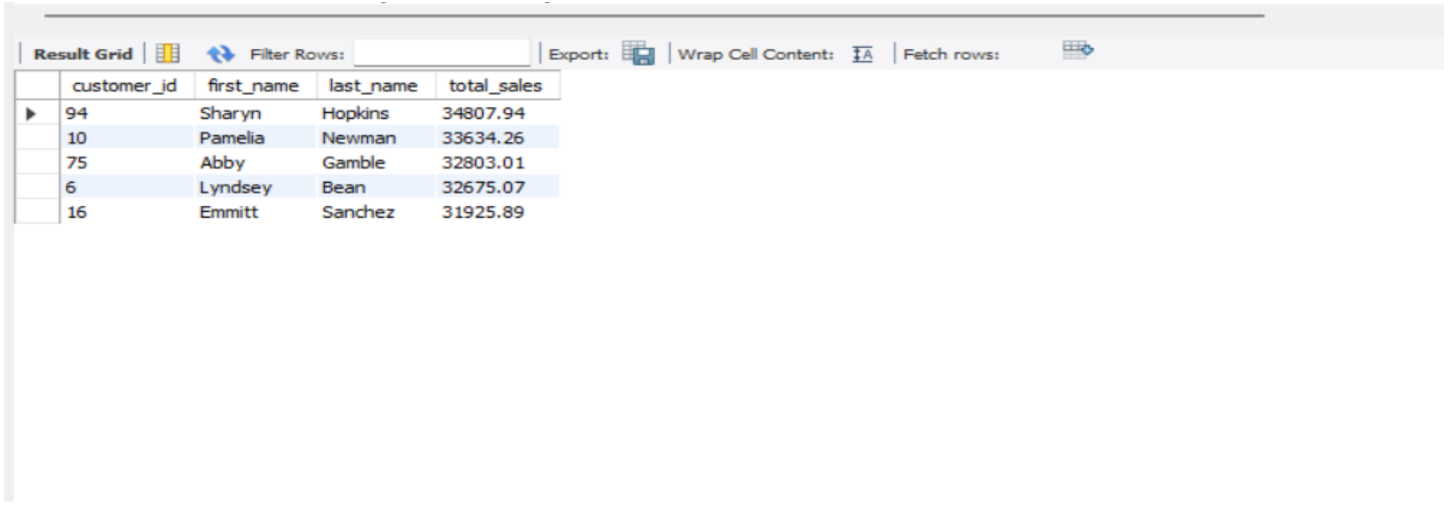


customer_id	first_name	last_name	quantity
3	Tameka	Fisher	19
16	Emmitt	Sanchez	19
10	Pamelia	Newman	18
1	Debra	Burks	17
61	Elinore	Aguilar	17

7. Top 5 Customers by Total Sales

```
SELECT c.customer_id, c.first_name, c.last_name, ROUND(SUM(oi.quantity * oi.list_price * (1 - oi.discount)),2) AS
total_sales
FROM order_items oi
JOIN orders o ON oi.order_id = o.order_id
JOIN customers c ON o.customer_id = c.customer_id
GROUP BY c.customer_id, c.first_name, c.last_name
ORDER BY total_sales DESC
LIMIT 5;
```

OUTPUT

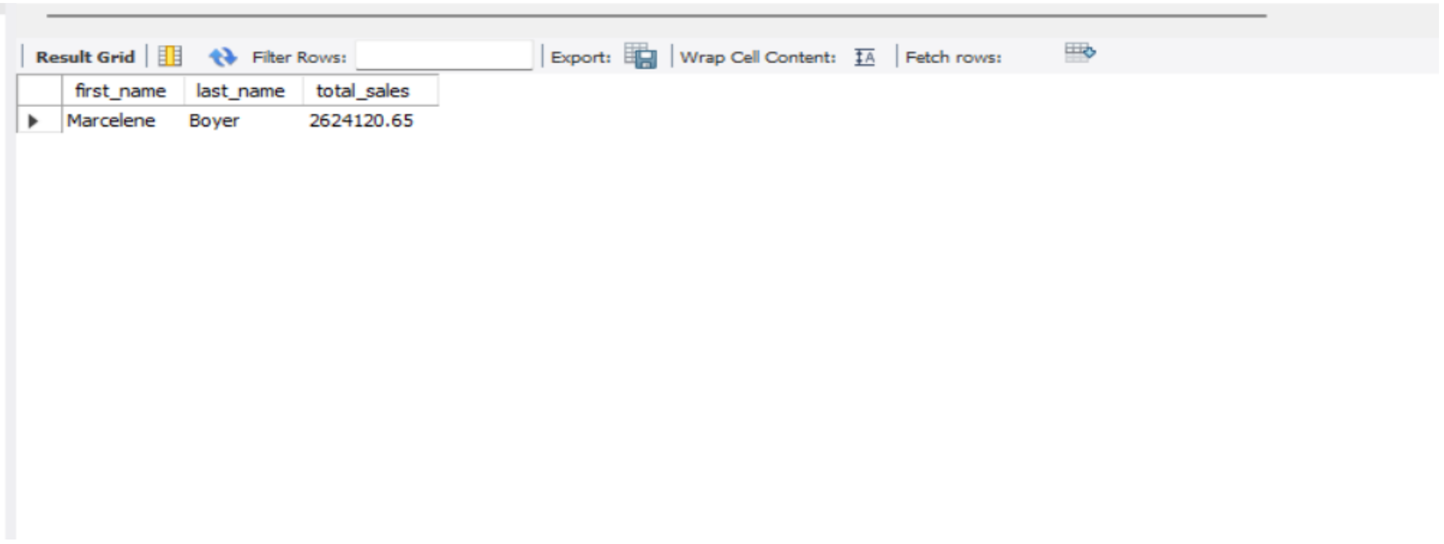


customer_id	first_name	last_name	total_sales
94	Sharyn	Hopkins	34807.94
10	Pamelia	Newman	33634.26
75	Abby	Gamble	32803.01
6	Lyndsey	Bean	32675.07
16	Emmitt	Sanchez	31925.89

8. Top Salesperson by Total Sales

```
SELECT s.first_name, s.last_name, ROUND(SUM(oi.quantity * oi.list_price * (1 - oi.discount)),2) AS total_sales
FROM order_items oi
JOIN orders o ON oi.order_id = o.order_id
JOIN staffs s ON o.staff_id = s.staff_id
GROUP BY s.first_name, s.last_name
ORDER BY total_sales DESC
LIMIT 1;
```

OUTPUT

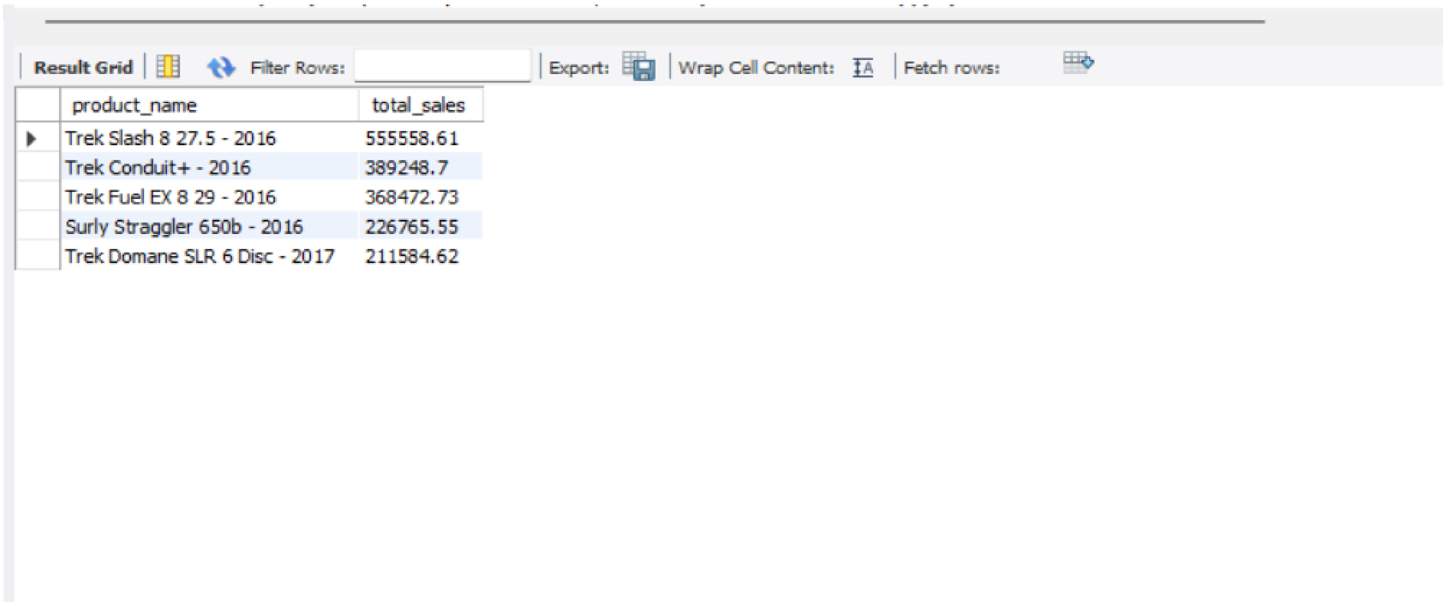


first_name	last_name	total_sales
Marcelene	Boyer	2624120.65

9. Top 5 Products by Total Sales

```
SELECT p.product_name, ROUND(SUM(oi.quantity * oi.list_price * (1 - oi.discount)),2) AS total_sales
FROM order_items oi
JOIN products p USING (product_id)
GROUP BY p.product_name
ORDER BY total_sales DESC
LIMIT 5;
```

OUTPUT

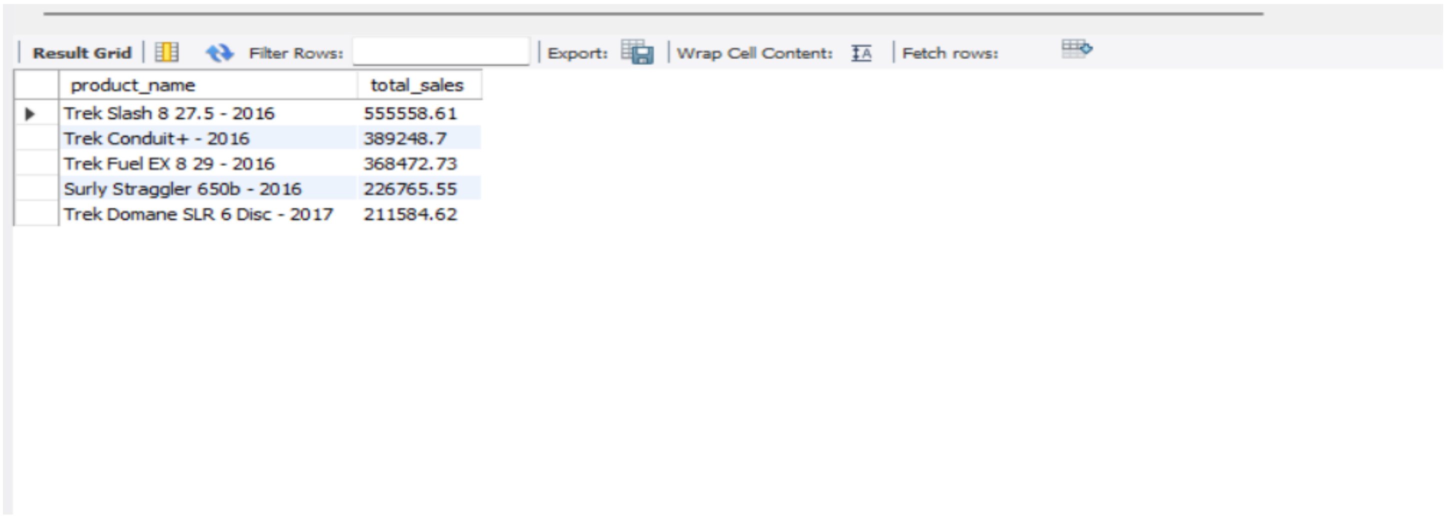


product_name	total_sales
Trek Slash 8 27.5 - 2016	555558.61
Trek Conduit+ - 2016	389248.7
Trek Fuel EX 8 29 - 2016	368472.73
Surly Straggler 650b - 2016	226765.55
Trek Domane SLR 6 Disc - 2017	211584.62

10. Top 5 Most Sold Product by Brand

```
SELECT b.brand_name, p.product_name, ROUND(SUM(oi.quantity * oi.list_price * (1 - oi.discount)),2) AS
total_sales
FROM order_items oi
JOIN products p ON oi.product_id = p.product_id
JOIN brands b ON p.brand_id = b.brand_id
GROUP BY p.product_name, b.brand_name
ORDER BY total_sales DESC
LIMIT 5;
```

OUTPUT

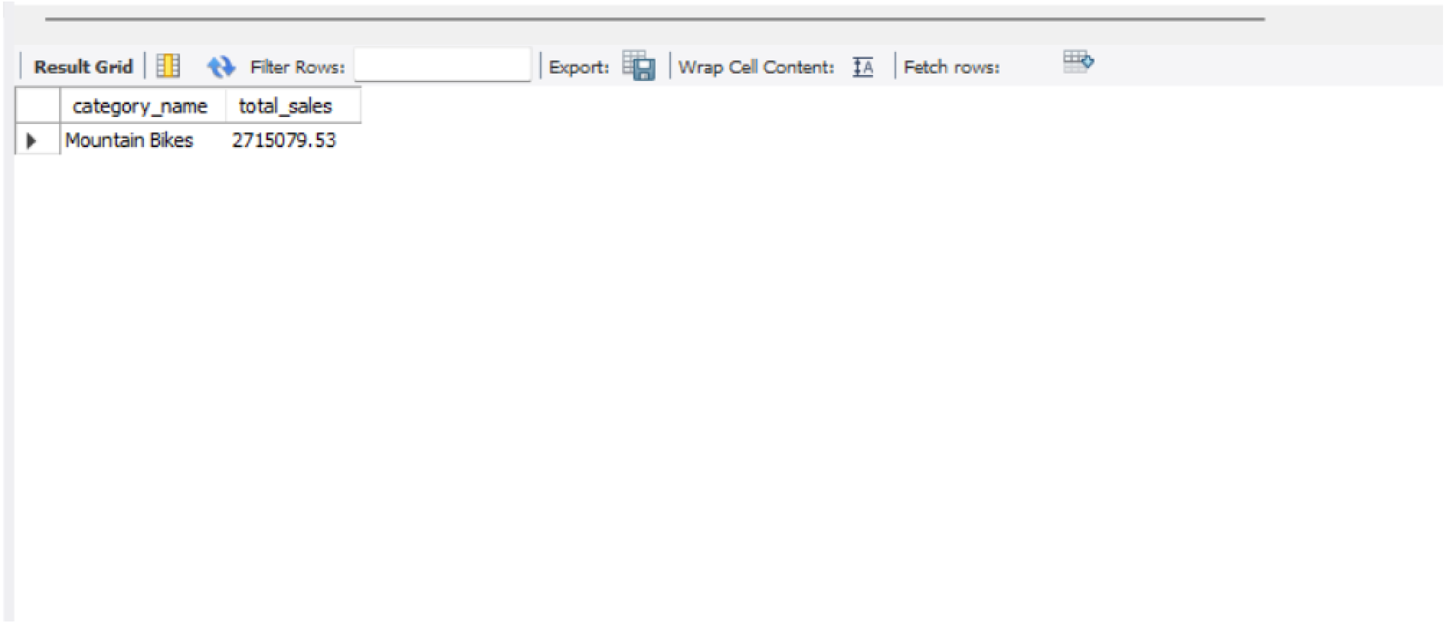


product_name	total_sales
Trek Slash 8 27.5 - 2016	555558.61
Trek Conduit+ - 2016	389248.7
Trek Fuel EX 8 29 - 2016	368472.73
Surly Straggler 650b - 2016	226765.55
Trek Domane SLR 6 Disc - 2017	211584.62

11. Most Sold Product Category

```
SELECT c.category_name, ROUND(SUM(oi.quantity * oi.list_price * (1 - oi.discount)),2) AS total_sales
FROM order_items oi
JOIN products p ON oi.product_id = p.product_id
JOIN categories c ON p.category_id = c.category_id
GROUP BY c.category_name
ORDER BY total_sales DESC
LIMIT 1;
```

OUTPUT

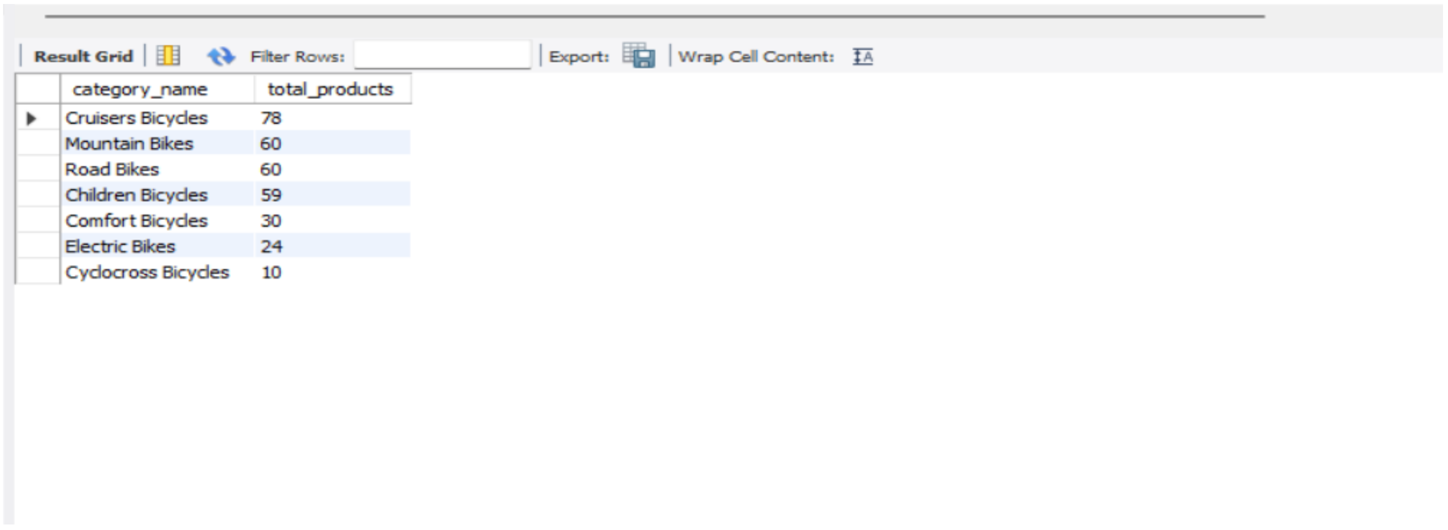


category_name	total_sales
Mountain Bikes	2715079.53

12. Number of Products in Each Category

```
SELECT c.category_name, COUNT(DISTINCT p.product_name) AS total_products
FROM products p
JOIN categories c ON p.category_id = c.category_id
GROUP BY c.category_name
ORDER BY total_products DESC;
```

OUTPUT

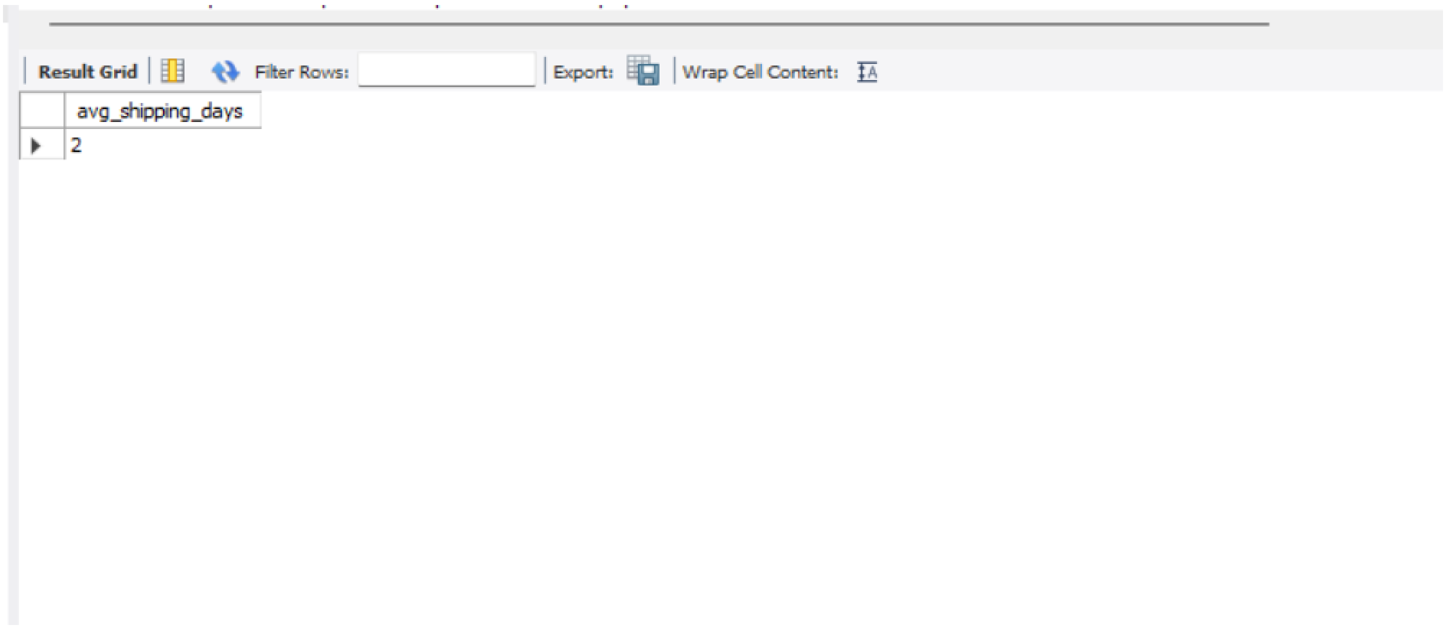


category_name	total_products
Cruisers Bicycles	78
Mountain Bikes	60
Road Bikes	60
Children Bicycles	59
Comfort Bicycles	30
Electric Bikes	24
Cyclocross Bicycles	10

13. Average Shipping Time

```
SELECT ROUND(AVG(DATEDIFF(shipped_date, order_date))) AS avg_shipping_days
FROM orders
WHERE shipped_date IS NOT NULL;
```

OUTPUT

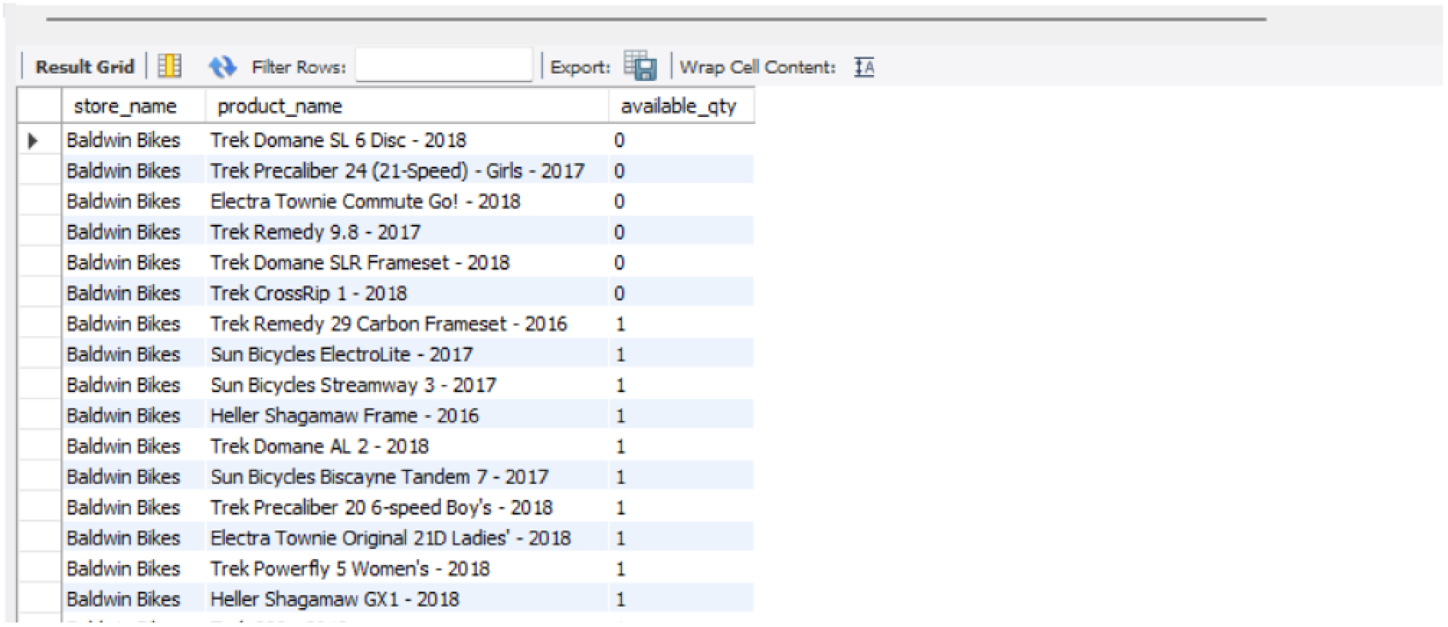


avg_shipping_days
2

14. Available Stock Quantity per Store and Product

```
SELECT str.store_name, p.product_name, SUM(quantity) AS available_qty
FROM stocks stk
JOIN stores str ON stk.store_id = str.store_id
JOIN products p ON stk.product_id = p.product_id
GROUP BY str.store_name, p.product_name
ORDER BY str.store_name, available_qty;
```

OUTPUT

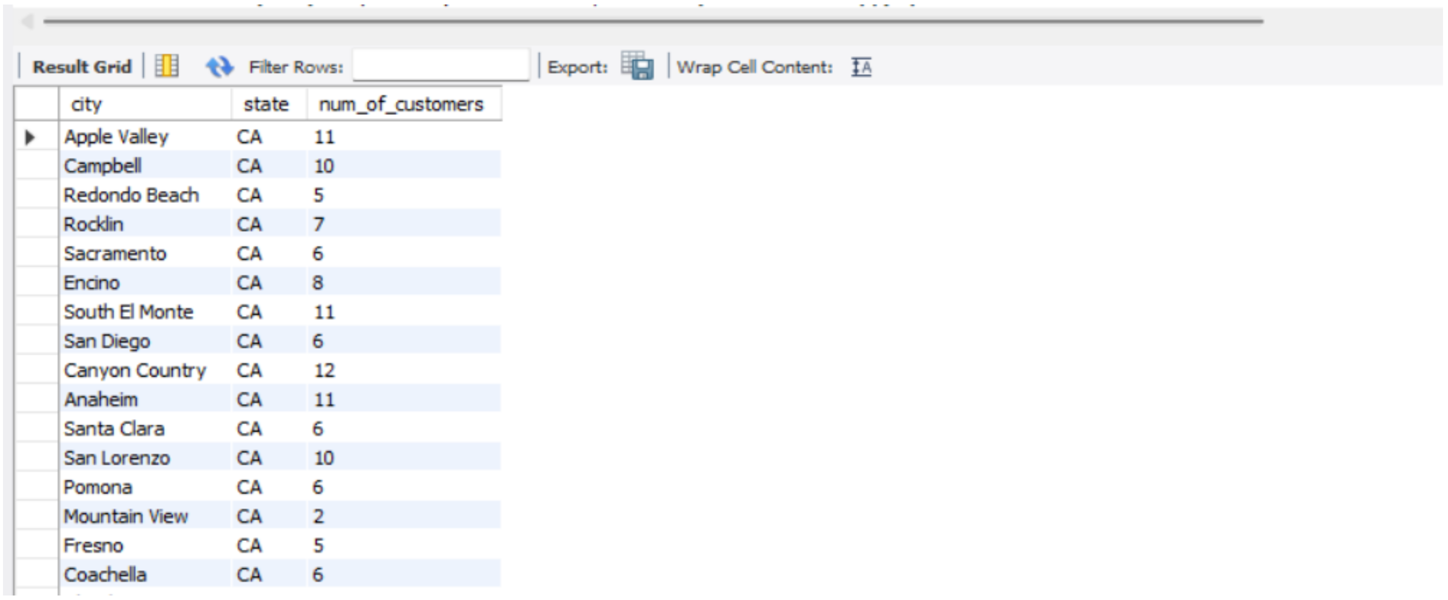


store_name	product_name	available_qty
Baldwin Bikes	Trek Domane SL 6 Disc - 2018	0
Baldwin Bikes	Trek Precaliber 24 (21-Speed) - Girls - 2017	0
Baldwin Bikes	Electra Townie Commute Go! - 2018	0
Baldwin Bikes	Trek Remedy 9.8 - 2017	0
Baldwin Bikes	Trek Domane SLR Frameset - 2018	0
Baldwin Bikes	Trek CrossRip 1 - 2018	0
Baldwin Bikes	Trek Remedy 29 Carbon Frameset - 2016	1
Baldwin Bikes	Sun Bicycles ElectroLite - 2017	1
Baldwin Bikes	Sun Bicycles Streamway 3 - 2017	1
Baldwin Bikes	Heller Shagamaw Frame - 2016	1
Baldwin Bikes	Trek Domane AL 2 - 2018	1
Baldwin Bikes	Sun Bicycles Biscayne Tandem 7 - 2017	1
Baldwin Bikes	Trek Precaliber 20 6-speed Boy's - 2018	1
Baldwin Bikes	Electra Townie Original 21D Ladies' - 2018	1
Baldwin Bikes	Trek Powerfly 5 Women's - 2018	1
Baldwin Bikes	Heller Shagamaw GX1 - 2018	1

15. Customer Count per City & State

```
SELECT c.city, c.state, COUNT(customer_id) AS num_of_customers
FROM customers c
GROUP BY c.city, c.state
ORDER BY c.state;
```

OUTPUT

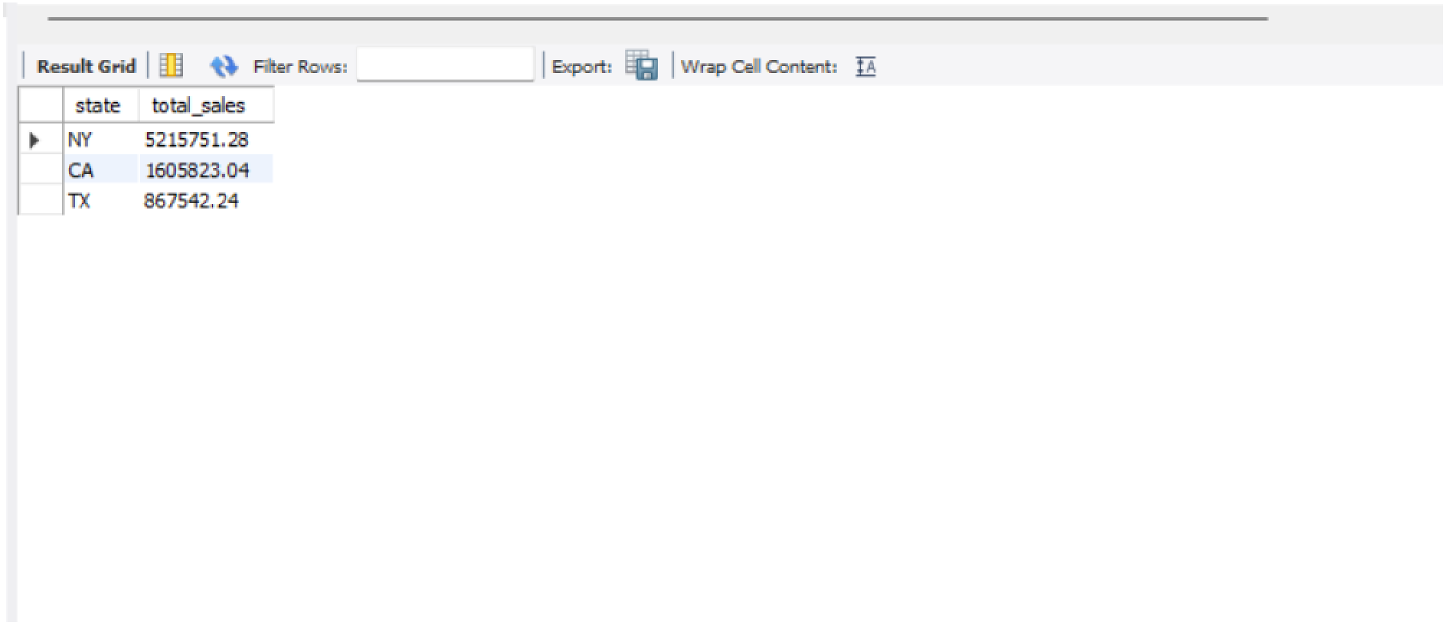


city	state	num_of_customers
Apple Valley	CA	11
Campbell	CA	10
Redondo Beach	CA	5
Rocklin	CA	7
Sacramento	CA	6
Encino	CA	8
South El Monte	CA	11
San Diego	CA	6
Canyon Country	CA	12
Anaheim	CA	11
Santa Clara	CA	6
San Lorenzo	CA	10
Pomona	CA	6
Mountain View	CA	2
Fresno	CA	5
Coachella	CA	6

16. Total Sales in Each State

```
SELECT c.state,
        ROUND(SUM(oi.quantity * oi.list_price * (1 - discount)),2) total_sales
FROM order_items oi
JOIN orders o ON oi.order_id = o.order_id
JOIN customers c ON c.customer_id = o.customer_id
GROUP BY c.state
ORDER BY total_sales DESC;
```

OUTPUT

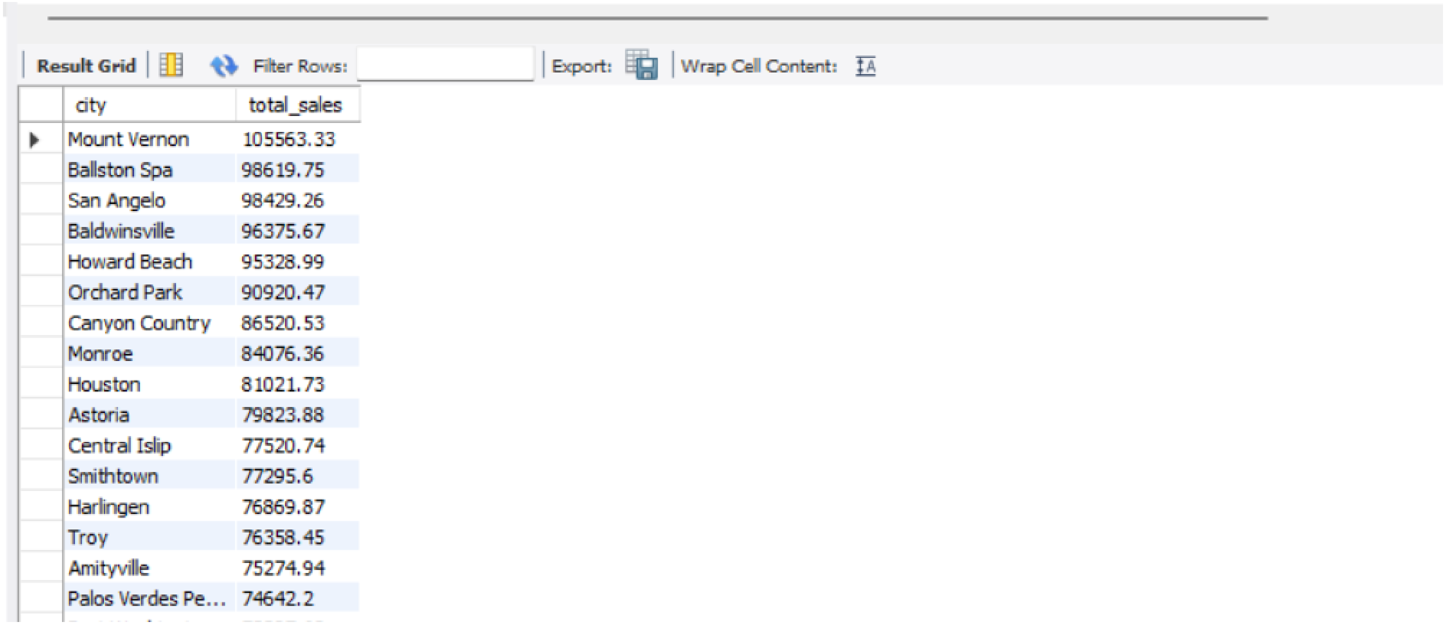


state	total_sales
NY	5215751.28
CA	1605823.04
TX	867542.24

17. Total Sales in Each City

```
SELECT c.city,
        ROUND(SUM(oi.quantity * oi.list_price * (1 - discount)),2) total_sales
FROM order_items oi
JOIN orders o ON oi.order_id = o.order_id
JOIN customers c ON c.customer_id = o.customer_id
GROUP BY c.city
ORDER BY total_sales DESC;
```

OUTPUT



city	total_sales
Mount Vernon	105563.33
Ballston Spa	98619.75
San Angelo	98429.26
Baldwinsville	96375.67
Howard Beach	95328.99
Orchard Park	90920.47
Canyon Country	86520.53
Monroe	84076.36
Houston	81021.73
Astoria	79823.88
Central Islip	77520.74
Smithtown	77295.6
Harlingen	76869.87
Troy	76358.45
Amityville	75274.94
Palos Verdes Pe...	74642.2

This document provides a structured SQL analysis of the **Bike Store** dataset, addressing key business insights on sales, customers, products, and store performance.

RAW CODE :-

```
-- ------------------ DATABASE CREATION & INITIAL EXPLORATION ------------------
-- Create the database
CREATE DATABASE Bike_Store;

-- Select the database for use
USE Bike_Store;

-- View all tables and their data
SELECT * FROM brands;
SELECT * FROM categories;
SELECT * FROM customers;
SELECT * FROM order_items;
SELECT * FROM orders;
SELECT * FROM products;
SELECT * FROM staffs;
SELECT * FROM stocks;
SELECT * FROM stores;


-- ------------------ DATA CLEANING & DATA TYPE CORRECTION ------------------
-- Change the data type of order_date, shipped_date, and required_date to DATE for accurate date operations
ALTER TABLE orders
MODIFY COLUMN order_date DATE,
MODIFY COLUMN shipped_date DATE,
MODIFY COLUMN required_date DATE;


-- ------------------ ASSIGNING PRIMARY KEYS ------------------
-- Ensure each table has a unique identifier
ALTER TABLE brands ADD PRIMARY KEY (brand_id);
ALTER TABLE categories ADD PRIMARY KEY (category_id);
ALTER TABLE customers ADD PRIMARY KEY (customer_id);
ALTER TABLE orders ADD PRIMARY KEY (order_id);
ALTER TABLE products ADD PRIMARY KEY (product_id);
ALTER TABLE staffs ADD PRIMARY KEY (staff_id);
ALTER TABLE stores ADD PRIMARY KEY (store_id);


-- ------------------ ESTABLISHING FOREIGN KEY RELATIONSHIPS ------------------
-- Link order_items table to orders and products tables
ALTER TABLE order_items
ADD CONSTRAINT fk_order FOREIGN KEY (order_id) REFERENCES orders(order_id),
ADD CONSTRAINT fk_product FOREIGN KEY (product_id) REFERENCES products(product_id);


-- Link orders table to customers, stores, and staffs tables
```

```
ALTER TABLE orders
ADD CONSTRAINT fk_customer FOREIGN KEY (customer_id) REFERENCES customers(customer_id),
ADD CONSTRAINT fk_stores FOREIGN KEY (store_id) REFERENCES stores(store_id),
ADD CONSTRAINT fk_staff FOREIGN KEY (staff_id) REFERENCES staffs(staff_id);


-- Link products table to brands and categories tables
ALTER TABLE products
ADD CONSTRAINT fk_brand FOREIGN KEY (brand_id) REFERENCES brands(brand_id),
ADD CONSTRAINT fk_category FOREIGN KEY (category_id) REFERENCES categories(category_id);


-- Link staffs table to stores table
ALTER TABLE staffs
ADD FOREIGN KEY (store_id) REFERENCES stores(store_id);


-- Link stocks table to stores and products tables
ALTER TABLE stocks
ADD FOREIGN KEY fK_store (store_id) REFERENCES stores(store_id),
ADD FOREIGN KEY fK_product (product_id) REFERENCES products(product_id);

-- ====================== PROJECT BUSINESS QUESTIONS =======================
-- BELOW ARE SOME KEY BUSINESS QUESTIONS I IDENTIFIED AND SOLVED IN THIS PROJECT.
-- THEY AIM TO PROVIDE INSIGHTS INTO SALES PERFORMANCE, PRODUCT POPULARITY, AND STORE-LEVEL
ANALYSIS.

use bike_store;

-- 1. WHAT IS THE TOTAL SALES AMOUNT?
SELECT ROUND(SUM(quantity * list_price * (1 - discount)),2) AS total_sales
FROM order_items;

-- 2. WHAT IS THE TOTAL QUANTITY SOLD
SELECT SUM(quantity) AS total_quantity_sold
FROM order_items;

-- 3. WHAT ARE THE TOP 5 MOST POPULAR PRODUCTS BASED ON TOTAL QUANTITY SOLD?
SELECT p.product_name,
	SUM(oi.quantity) AS total_quantity_sold
FROM order_items oi
JOIN products p
USING (product_id)
GROUP BY p.product_name
ORDER BY total_quantity_sold DESC
LIMIT 5;

-- 4. WHAT ARE THE TOTAL SALES BY EACH STORE?
```

```
SELECT s.store_name,
        ROUND(SUM(oi.quantity * oi.list_price * (1 - oi.discount)),2) AS total_sales
FROM order_items oi
JOIN orders o USING (order_id)
JOIN stores s USING (store_id)
GROUP BY s.store_name;

-- 5. WHAT IS THE TOTAL QUANTITY SOLD BY EACH STORE?
SELECT s.store_name,
        SUM(oi.quantity) AS total_quantity_sold
FROM order_items oi
JOIN orders o USING (order_id)
JOIN stores s USING (store_id)
GROUP BY s.store_name;

-- 6. WHO ARE THE TOP 5 CUSTOMERS BASED ON QUANTITY PURCHASED?
SELECT c.customer_id, c.first_name, c.last_name,
   SUM(oi.quantity) AS quantity
FROM order_items oi
JOIN orders o ON oi.order_id = o.order_id
JOIN customers c ON o.customer_id = c.customer_id
GROUP BY c.customer_id, c.first_name, c.last_name
ORDER BY quantity DESC
LIMIT 5;

-- 7. WHO ARE THE TOP 5 CUSTOMERS BASED ON TOTAL SALES?
SELECT c.customer_id, c.first_name, c.last_name,
   ROUND(SUM(oi.quantity * oi.list_price * (1 - oi.discount)),2) AS total_sales
FROM order_items oi
JOIN orders o ON oi.order_id = o.order_id
JOIN customers c ON o.customer_id = c.customer_id
GROUP BY c.customer_id, c.first_name, c.last_name
ORDER BY total_sales DESC
LIMIT 5;

-- 8. WHO IS THE TOP SALESPERSON BASED ON TOTAL SALES?
SELECT s.first_name, s.last_name,
        ROUND(SUM(oi.quantity * oi.list_price * (1 - oi.discount)),2) AS total_sales
FROM order_items oi
JOIN orders o ON oi.order_id = o.order_id
JOIN staffs s ON o.staff_id = s.staff_id
GROUP BY s.first_name, s.last_name
ORDER BY total_sales DESC
LIMIT 1;

-- 9. WHAT ARE THE TOP 5 MOST POPULAR PRODUCTS BASED ON TOTAL SALES?
SELECT p.product_name,
        ROUND(SUM(oi.quantity * oi.list_price * (1 - oi.discount)),2) AS total_sales
```

```
FROM order_items oi
JOIN products p
USING (product_id)
GROUP BY p.product_name
ORDER BY total_sales DESC
LIMIT 5;

-- 10. WHAT IS THE MOST SOLD PRODUCT BY BRAND?
SELECT b.brand_name, p.product_name,
	ROUND(SUM(oi.quantity * oi.list_price * (1 - oi.discount)),2) AS total_sales
FROM order_items oi
JOIN products p ON oi.product_id = p.product_id
JOIN brands b ON p.brand_id = b.brand_id
GROUP BY p.product_name, b.brand_name
ORDER BY total_sales DESC
LIMIT 5;

-- 11. WHAT IS THE MOST SOLD PRODUCT CATEGORY?
SELECT c.category_name,
	ROUND(SUM(oi.quantity * oi.list_price* (1 - oi.discount)),2) AS total_sales
FROM order_items oi
JOIN products p ON oi.product_id = p.product_id
JOIN categories c ON p.category_id = c.category_id
GROUP BY c.category_name
ORDER BY total_sales DESC
LIMIT 1;

-- 12. HOW MANY PRODUCTS ARE THERE IN EACH CATEGORY?
SELECT c.category_name,
	COUNT(DISTINCT p.product_name) AS total_products
FROM products p
JOIN categories c ON p.category_id = c.category_id
GROUP BY c.category_name
ORDER BY total_products DESC;

-- 13. WHAT IS THE AVERAGE SHIPPING TIME IN DAYS?
SELECT ROUND(AVG(DATEDIFF(shipped_date, order_date))) AS avg_shipping_days
FROM orders
WHERE shipped_date IS NOT NULL;

-- 14. WHAT IS THE TOTAL STOCK QUANTITY AVAILABLE FOR EACH PRODUCT IN EACH STORE?
SELECT str.store_name, p.product_name,
	SUM(quantity) AS available_qty
FROM stocks stk
JOIN stores str ON stk.store_id = str.store_id
JOIN products p ON stk.product_id = p.product_id
GROUP BY str.store_name, p.product_name
ORDER BY str.store_name, available_qty;
```

```
-- 15. HOW MANY CUSTOMERS ARE THERE IN EACH CITY & STATE?
SELECT c.city, c.state, COUNT(customer_id) AS num_of_customers
FROM customers c
GROUP BY c.city, c.state
ORDER BY c.state;

-- 16. WHAT ARE TOTAL SALES IN EACH STATE?
SELECT c.state,
        ROUND(SUM(oi.quantity * oi.list_price * (1 - discount)),2) total_sales
FROM order_items oi
JOIN orders o ON oi.order_id = o.order_id
JOIN customers c ON c.customer_id = o.customer_id
GROUP BY c.state
ORDER BY total_sales DESC;

-- 17. WHAT ARE TOTAL SALES IN EACH CITY?
SELECT c.city,
        ROUND(SUM(oi.quantity * oi.list_price * (1 - discount)),2) total_sales
FROM order_items oi
JOIN orders o ON oi.order_id = o.order_id
JOIN customers c ON c.customer_id = o.customer_id
GROUP BY c.city
ORDER BY total_sales DESC;
```