Database for E-commerce firm

Objective:

The goal is to build a strong and effective platform specifically for Boat Lifestyle firm's e-commerce activities which are used to optimize and streamline several operational factors that are essential for the effective online retail administration. It is designed to smoothly combine different business components to improve user experience, streamline sales procedures, provide thorough cost analysis, and support well-informed decision-making.

Scope:

The core functionality includes managing customer information such as names, contact details, and addresses, facilitating seamless order fulfillment and maintaining a robust product catalog that includes details like availability, categories, and pricing. The system also integrates with payment processing for smooth transaction management and tracks order statuses to ensure timely deliveries.

Additionally, the database incorporates sophisticated cost analysis features tied to different countries and products, enabling the firm to make informed decisions regarding pricing strategies and market competitiveness. It also includes stock management functionalities to monitor inventory levels, track purchase and sale dates, and manage product availability across different regions.

Overall, the database project is geared towards enhancing user experience, optimizing sales operations, enabling data-driven decision-making through comprehensive analytics, and ensuring seamless coordination of various aspects of the Boat Lifestyle firm's e-commerce activities. It will be a collection storage system designed to meet all company needs while concentrating on the complexities of the industry

inserting products* | Table created 1* | SQL File 28* | Query 1*

Limit to 2000 rows

```sql
create database cost_comparison_DATBASE;
use cost_comparison_DATBASE;


CREATE TABLE Country (
    Country_id INT AUTO_INCREMENT PRIMARY KEY,
    Country_name VARCHAR(255) UNIQUE NOT NULL
);

select * from country;

CREATE TABLE Customers (
    Customer_id INT AUTO_INCREMENT PRIMARY KEY,
    Customer_Fname VARCHAR(100),
    Customer_Lname VARCHAR(100),
    Phone_Number VARCHAR(15),
    Address VARCHAR(250),  -- Changed the data type to VARCHAR(250)
    Pincode VARCHAR(10),
    Country_id INT,
    CONSTRAINT fk_country FOREIGN KEY (Country_id) REFERENCES Country(Count
);

select * from customers
order by Country_id;
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	16:33:51	create database cost_comparison_...	1 row(s) affected	0.000 sec
2	16:33:57	use cost_comparison_DATABASE	Error Code: 1049. Unknown databa...	0.000 sec
3	16:34:07	use cost_comparison_DATBASE	0 row(s) affected	0.000 sec
4	16:34:46	CREATE TABLE Country (Count...	0 row(s) affected	0.094 sec

inserting products* | Table created 1* | SQL File 28* | Query 1*

Limit to 2000 rows

```sql
    Country_name VARCHAR(255) UNIQUE NOT NULL
);

select * from country;

CREATE TABLE Customers (
    Customer_id INT AUTO_INCREMENT PRIMARY KEY,
    Customer_Fname VARCHAR(100),
    Customer_Lname VARCHAR(100),
    Phone_Number VARCHAR(15),
    Address VARCHAR(250),  -- Changed the data type to VARCHAR(250)
    Pincode VARCHAR(10),
    Country_id INT,
    CONSTRAINT fk_country FOREIGN KEY (Country_id) REFERENCES Country(Count
);

select * from customers
order by Country_id;

CREATE INDEX idx_address ON Customers(Address);
CREATE INDEX idx_pincode ON Customers(Pincode);
CREATE INDEX idx_customer_id ON Customers(Customer_id);

-- Categories Entity
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
4	16:34:46	CREATE TABLE Country (Count...	0 row(s) affected	0.094 sec
5	16:34:51	CREATE TABLE Customers (Cu...	0 row(s) affected	0.109 sec
6	16:35:00	CREATE INDEX idx_address ON C...	0 row(s) affected Records: 0 Dupli...	0.047 sec
7	16:35:00	CREATE INDEX idx_pincode ON C...	0 row(s) affected Records: 0 Dupli...	0.046 sec

inserting products* | Table created 1* | SQL File 28* | Query 1*

Limit to 2000 rows

```sql
CREATE TABLE Categories (
    Category_id INT AUTO_INCREMENT PRIMARY KEY,
    Category_Name VARCHAR(255) NOT NULL
);

-- Products Entity
CREATE TABLE Products (
    Product_id INT AUTO_INCREMENT PRIMARY KEY,
    Category_id INT,
    product_name varchar(100),
    Product_Details TEXT,
    FOREIGN KEY (Category_id) REFERENCES Categories(Category_id)
);

select * from products;

-- Adding Inventory Entity
CREATE TABLE Inventory (
    Inventory_id INT AUTO_INCREMENT PRIMARY KEY,
    Product_id INT,
    Quantity INT,
    country_id int,
    Item_cost DECIMAL(10, 2),
    FOREIGN KEY (Product_id) REFERENCES Products(Product_id),
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
8	16:35:00	CREATE INDEX idx_customer_id ...	0 row(s) affected Records: 0 Dupli...	0.032 sec
9	16:35:07	CREATE TABLE Categories (Ca...	0 row(s) affected	0.047 sec
10	16:36:00	CREATE TABLE Products (Prod...	0 row(s) affected	0.062 sec
11	16:37:05	CREATE TABLE Inventory (Inv...	0 row(s) affected	0.078 sec

inserting products* | Table created 1* | SQL File 28* | Query 1*

Limit to 2000 rows

```sql

-- Orders Entity
CREATE TABLE Orders(
    Order_id INT AUTO_INCREMENT PRIMARY KEY,
    product_id INT,
    Quantity int,
    foreign key (product_id) references products(product_id),
    foreign key (quantity) references inventory(quantity)
);
alter table orders add column item_cost Decimal(10,2);

-- Payments Entity
CREATE TABLE Payments (
    Payment_id INT AUTO_INCREMENT PRIMARY KEY,
    Order_id INT,
    Payment_Method VARCHAR(50),
    Payment_Amount DECIMAL(10, 2),
    FOREIGN KEY (Order_id) REFERENCES Orders(Order_id)
);
alter table payments
drop column payment_method;

-- Adding Invoice Entity
CREATE TABLE Invoices (
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
17	16:44:55	CREATE INDEX idx_quantity ON I...	0 row(s) affected Records: 0 Dupli...	0.047 sec
18	16:45:01	CREATE TABLE Orders(Order_i...	0 row(s) affected	0.125 sec
19	16:45:12	CREATE TABLE Order_Details (...	Error Code: 3780. Referencing colu...	0.015 sec
20	16:47:07	CREATE TABLE Order_Details (...	Error Code: 3780. Referencing colu...	0.016 sec

```
);
alter table orders add column item_cost Decimal(10,2);

-- Payments Entity
CREATE TABLE Payments (
    Payment_id INT AUTO_INCREMENT PRIMARY KEY,
    Order_id INT,
    Payment_Method VARCHAR(50),
    Payment_Amount DECIMAL(10, 2),
    FOREIGN KEY (Order_id) REFERENCES Orders(Order_id)
);
alter table payments
drop column payment_method;

-- Adding Invoice Entity
CREATE TABLE Invoices (
    Invoice_id INT AUTO_INCREMENT PRIMARY KEY,
    Order_id INT,
    Total_cost DECIMAL(10, 2),
    Purchased_Date DATE,
    FOREIGN KEY (Order_id) REFERENCES Orders(Order_id)
);
show tables;
select * from inventory;
```

Output — Action Output

#	Time	Action	Message	Duration / Fetch
21	16:47:19	CREATE TABLE Order_Details (...	0 row(s) affected	0.094 sec
22	16:47:41	CREATE TABLE Payments (Pay...	0 row(s) affected	0.063 sec
23	16:47:46	CREATE TABLE Invoices (Invoi...	0 row(s) affected	0.078 sec
24	16:47:50	show tables	10 row(s) returned	0.000 sec / 0.000 sec

```
    Order_id INT,
    Payment_Method VARCHAR(50),
    Payment_Amount DECIMAL(10, 2),
    FOREIGN KEY (Order_id) REFERENCES Orders(Order_id)
);
alter table payments
drop column payment_method;

-- Adding Invoice Entity
CREATE TABLE Invoices (
    Invoice_id INT AUTO_INCREMENT PRIMARY KEY,
    Order_id INT,
    Total_cost DECIMAL(10, 2),
    Purchased_Date DATE,
    FOREIGN KEY (Order_id) REFERENCES Orders(Order_id)
);
show tables;
select * from inventory;
```

Result Grid

Tables_in_cost_comparison_datbase
categories
country
customers
inventory
invoices
orders
payments
products

```
INSERT INTO Country (Country_name) VALUES ('India'), ('China'), ('Europe'), ('Australia'), ('USA');

INSERT INTO Customers (Customer_Fname, Customer_Lname, Phone_Number, Address, Pincode, Country_id) VALUE
('John', 'Doe', '1234567890', '123 Main St', '00001', (SELECT Country_id FROM Country WHERE Country_name
('Jane', 'Doe', '1234567891', '124 Main St', '00002', (SELECT Country_id FROM Country WHERE Country_name
('Raj', 'Kumar', '1234567892', '125 Main St', '00003', (SELECT Country_id FROM Country WHERE Country_nam
('Amit', 'Shah', '1234567893', '126 Main St', '00004', (SELECT Country_id FROM Country WHERE Country_nam
('Li', 'Wang', '1234567894', '127 Main St', '00005', (SELECT Country_id FROM Country WHERE Country_name
('Xiu', 'Chen', '1234567895', '128 Main St', '00006', (SELECT Country_id FROM Country WHERE Country_name
('Lucas', 'Müller', '1234567896', '129 Main St', '00007', (SELECT Country_id FROM Country WHERE Country_
('Emma', 'Schmidt', '1234567897', '130 Main St', '00008', (SELECT Country_id FROM Country WHERE Country_
('Ethan', 'Brown', '1234567898', '131 Main St', '00009', (SELECT Country_id FROM Country WHERE Country_r
('Olivia', 'Smith', '1234567899', '132 Main St', '00010', (SELECT Country_id FROM Country WHERE Country_

INSERT INTO Categories (category_name) VALUES ('Electronics');

-- Inserting electronic products
INSERT INTO Products (category_id, product_name, product_details) VALUES
((SELECT category_id FROM categories WHERE category_name = 'Electronics'), 'iPhone 12', '128GB, 6.1" Sup
((SELECT category_id FROM categories WHERE category_name = 'Electronics'), 'Samsung Galaxy S21', '256GB,
((SELECT category_id FROM categories WHERE category_name = 'Electronics'), 'Sony PlayStation 5', '825GB
((SELECT category_id FROM categories WHERE category_name = 'Electronics'), 'Apple MacBook Pro', '13" Ret
((SELECT category_id FROM categories WHERE category_name = 'Electronics'), 'DJI Mavic Air 2', '48MP came
((SELECT category_id FROM categories WHERE category_name = 'Electronics'), 'Canon EOS R5', '45MP full-fr
```

Output — Action Output

#	Time	Action	Message	Duration / Fetch
27	16:49:02	INSERT INTO Categories (category_name) VALUES ('El...	1 row(s) affected	0.016 sec
28	16:49:02	INSERT INTO Products (category_id, product_name, pro...	25 row(s) affected Records: 25 Duplicates: 0 Warnings: 0	0.015 sec
29	16:50:17	INSERT INTO Inventory (Product_id, Country_id, Quantit...	25 row(s) affected Records: 25 Duplicates: 0 Warnings: 0	0.032 sec
30	16:50:17	select * from inventory LIMIT 0, 2000	25 row(s) returned	0.000 sec / 0.000 sec

```
INSERT INTO Inventory (Product_id, Country_id, Quantity, Item_cost)
VALUES
    -- iPhone 12
    (1, 1, 100, 800.00),
    -- Samsung Galaxy S21
    (2, 1, 150, 700.00),
    -- Sony PlayStation 5
    (3, 1, 80, 1000.00),
    -- Apple MacBook Pro
    (4, 1, 120, 1500.00),
    -- DJI Mavic Air 2
    (5, 1, 50, 900.00),
    -- Canon EOS R5
    (6, 1, 70, 2500.00),
    -- Bose QuietComfort 35 II
    (7, 1, 90, 300.00),
    -- LG CX OLED TV
    (8, 1, 40, 2000.00),
    -- Nintendo Switch
    (9, 1, 200, 400.00),
    -- Google Pixel 5
    (10, 1, 100, 700.00),
    -- Fitbit Charge 4
    (11, 1, 150, 150.00),
    -- Amazon Echo Dot (4th Gen)
    (12, 1, 180, 50.00),
    -- Microsoft Surface Laptop 4
    (13, 1, 60, 1200.00),
    -- GoPro HERO9 Black
    (14, 1, 30, 450.00),
    -- Xbox Series X
    (15, 1, 40, 600.00),
    -- NVIDIA GeForce RTX 3080
    (16, 1, 20, 800.00),
    -- Roku Ultra
    (17, 1, 80, 100.00),
    -- Apple AirPods Pro
    (18, 1, 150, 200.00),
    -- Samsung Odyssey G7
    (19, 1, 50, 700.00),
    -- Garmin Fenix 6 Pro
    (20, 1, 70, 400.00),
    -- Sony WH-1000XM4
    (21, 1, 90, 300.00),
    -- Canon EOS 90D
    (22, 1, 40, 1200.00),
    -- HP Spectre x360
    (23, 1, 60, 1500.00),
    -- Ring Video Doorbell 3
    (24, 1, 100, 150.00),
    -- Sony WH-CH510
    (25, 1, 200, 50.00)
```

Output — Action Output

#	Time	Action	Duration / Fetch
36	13:34:34	INSERT INTO Inve	0.016 sec
37	13:34:57	INSERT INTO Inve	0.016 sec
38	13:36:16	INSERT INTO Inve	0.016 sec
39	13:36:54	INSERT INTO Inve	0.016 sec
40	13:37:28	INSERT INTO Inve	0.016 sec
41	13:38:22	select * from invent	0.000 sec / 0.000 sec

```
-- Inserting data into data table

INSERT INTO Orders (item_cost, product_id, Quantity)
SELECT item_cost, product_id, FLOOR(1 + (RAND() * 5)) AS Quantity
FROM inventory
ORDER BY RAND()
LIMIT 25;

-- Inserting data into payment table
INSERT INTO Payments (order_id, payment_amount)
SELECT o.Order_id, o.Quantity * o.item_cost AS payment_amount
FROM Orders o;

select * from invoices;

-- Inserting data into invoice table
INSERT INTO Invoices (order_id, Total_cost, Purchased_Date)
SELECT
    o.order_id,
    (o.Quantity * o.item_cost) AS Total_cost,
    DATE_ADD('2024-01-01', INTERVAL FLOOR(RAND() * 365) DAY) AS Purchased_Date
```

Output — Action Output

#	Time	Action	Message	Duration / Fetch
77	17:43:37	INSERT INTO Orders (item_cost, product_id, Quantity) S...	25 row(s) affected Records: 25 Duplicates: 0 Warnings: 0	0.016 sec
78	17:43:37	select *, country_id from orders o join inventory i where i....	250 row(s) returned	0.000 sec / 0.000 sec
79	17:44:17	select o.order_id, i.country_id from orders o join inventory ...	250 row(s) returned	0.000 sec / 0.000 sec
80	17:44:40	select * from orders LIMIT 0, 2000	50 row(s) returned	0.000 sec / 0.000 sec
81	17:54:58	INSERT INTO Payment (order_id, payment_amount) SEL...	Error Code: 1146. Table 'cost_comparison_database.paym...	0.015 sec
82	17:55:29	INSERT INTO Payments (order_id, payment_amount) SE...	50 row(s) affected Records: 50 Duplicates: 0 Warnings: 0	0.016 sec
83	17:55:54	select * from payments LIMIT 0, 2000	50 row(s) returned	0.000 sec / 0.000 sec

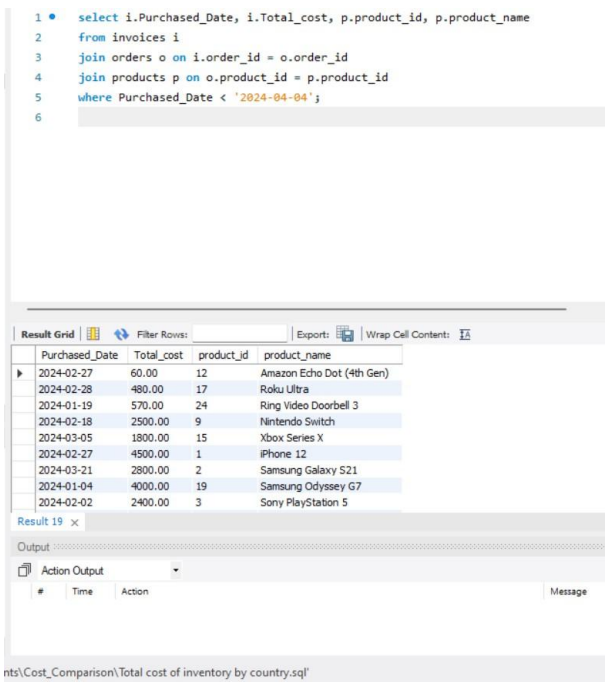
Data Retrieval and Simple Reports:

1. **Query 1**

```
select i.Purchased_Date, i.Total_cost, p.product_id, p.product_name
    from invoices i
    join orders o on i.order_id = o.order_id
    join products p on o.product_id = p.product_id
        where Purchased_Date < '2024-04-04';
```

Explanation: The SQL query retrieves the requested data from various tables using predefined parameters, such as invoices, orders, and products. It matches similar information to determine the purchase date, total cost, and product specifications. The query uses a filter to include only purchases made before April 4, 2024.

Result:

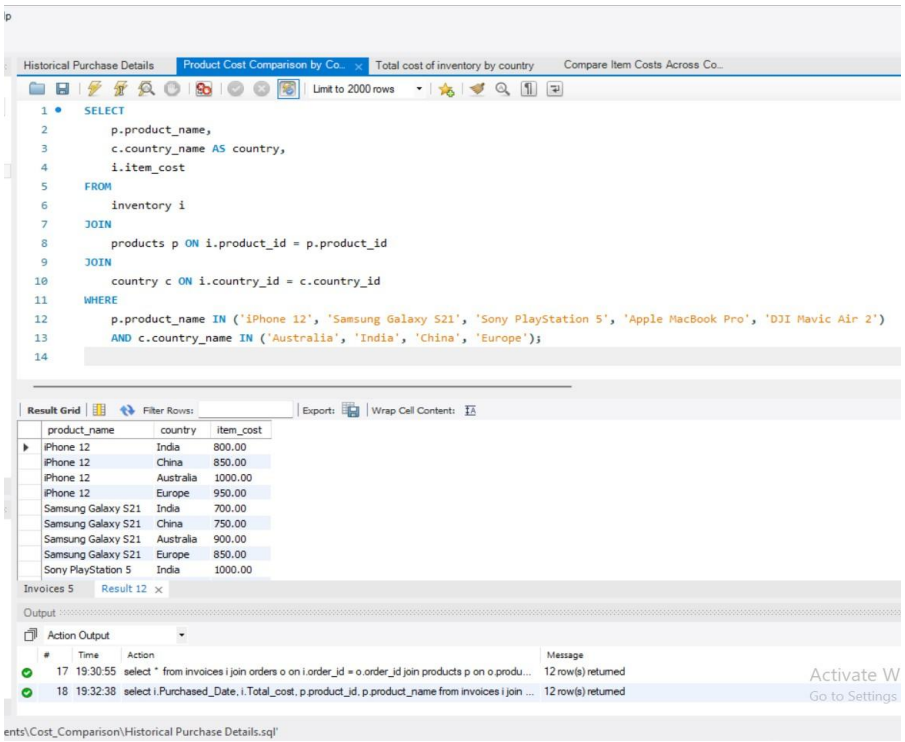


2. Query 2

```
SELECT
    p.product_name,
    c.country_name AS country,
    i.item_cost
FROM
    inventory i
JOIN
    products p ON i.product_id = p.product_id
JOIN
    country c ON i.country_id = c.country_id
WHERE
 p.product_name IN ('iPhone 12', 'Samsung Galaxy S21', 'Sony PlayStation 5', 'Apple MacBook Pro', 'DJI
    Mavic Air 2')
    AND c.country_name IN ('Australia', 'India', 'China', 'Europe');
```

Explanation: This SQL query retrieves the product name, country name, and item price from the "inventory," "products," and "country" tables. It connects the tables based on the product and country IDs. The results are then filtered to only contain records where the product name is one of the defined products (iPhone 12, Samsung Galaxy S21, Sony PlayStation 5, Apple MacBook Pro, DJI Mavic Air 2) and the nation name is one of the chosen regions (Australia, India, China, Europe).

Result:

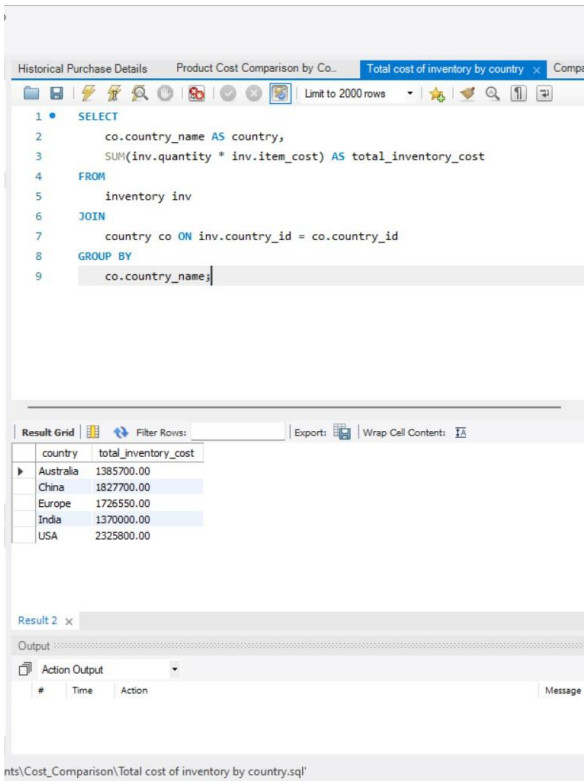


3. Query 3

```
SELECT
    co.country_name AS country,
    SUM(inv.quantity * inv.item_cost) AS total_inventory_cost
FROM
    inventory inv
JOIN
    country co ON inv.country_id = co.country_id
GROUP BY
    co.country_name;
```

Explanation: This SQL query shows the total inventory cost for each country. It selects the nation name from the "country" database and calculates the overall inventory cost by multiplying the number of items in inventory by their individual costs and adding them together. It connects the "inventory" and "country" tables depending on the country ID. The findings are then grouped by country name to calculate the overall inventory cost for each individual country.

Result:

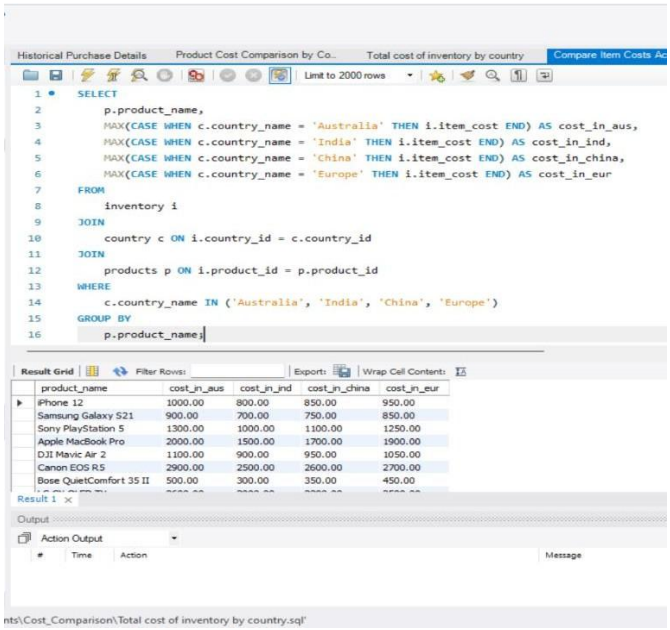


4. Query 4

```
SELECT
    p.product_name,
    MAX(CASE WHEN c.country_name = 'Australia' THEN i.item_cost END) AS cost_in_aus,
    MAX(CASE WHEN c.country_name = 'India' THEN i.item_cost END) AS cost_in_ind,
    MAX(CASE WHEN c.country_name = 'China' THEN i.item_cost END) AS cost_in_china,
    MAX(CASE WHEN c.country_name = 'Europe' THEN i.item_cost END) AS cost_in_eur
FROM
    inventory i
JOIN
    country c ON i.country_id = c.country_id
JOIN
    products p ON i.product_id = p.product_id
WHERE
    c.country_name IN ('Australia', 'India', 'China', 'Europe')
GROUP BY
    p.product_name;
```

Explanation: This SQL query retrieves the highest item price for each product in specified countries (Australia, India, China, and Europe). It chooses the product name from the "products" table and applies conditional aggregation to determine the maximum item cost for each country independently. It joins the "inventory" table with the "country" table using the country ID, followed by the "products" table using the product ID.
It produces the results to only items in which the country name is one of the chosen areas. Finally, it organizes the results by product name and displays the maximum item cost for each product in each country.

Result:



5. **Query 5**

```
SELECT
    c.country_name AS country,
    SUM(o.quantity) AS total_units_sold,
    SUM(o.quantity * i.item_cost) AS total_sales_revenue
FROM
    orders o
JOIN
    products p ON o.product_id = p.product_id
JOIN
    inventory i ON o.product_id = i.product_id
JOIN
    country c ON i.country_id = c.country_id
JOIN
    invoices inv ON o.order_id = inv.order_id
WHERE
    inv.Purchased_Date < '2024-04-04'
GROUP BY
    c.country_name
ORDER BY
    country;
```

Explanation: This SQL query determines total units sold and sales revenue for each country based on orders placed before April 4, 2024. It obtains the nation name from the "country" table and adds the amount of units sold to the associated sales revenue, which is determined by multiplying the quantity of units sold by the item cost. To get the required data, the query links the "orders" table with the "products," "inventory," "country," and "invoices" tables. It then filters the results to include only orders placed before April 4, 2024, and classifies them by country name. Finally, the results are ordered alphabetically by nation name.

Result: