

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*

```
1 • create database cost_comparison_DATABASE;
2 • use cost_comparison_DATABASE;
3
4
5 • CREATE TABLE Country (
6     Country_id INT AUTO_INCREMENT PRIMARY KEY,
7     Country_name VARCHAR(255) UNIQUE NOT NULL
8 );
9
10 • select * from country;
11
12 • CREATE TABLE Customers (
13     Customer_id INT AUTO_INCREMENT PRIMARY KEY,
14     Customer_Fname VARCHAR(100),
15     Customer_Lname VARCHAR(100),
16     Phone_Number VARCHAR(15),
17     Address VARCHAR(250), -- Changed the data type to VARCHAR(250)
18     Pincode VARCHAR(10),
19     Country_id INT,
20     CONSTRAINT fk_country FOREIGN KEY (Country_id) REFERENCES Country(Country_id)
21 );
22
23 • select * from customers
24     order by Country_id;
```

Output

#	Time	Action	Message	Duration / Fetch
1	16:33:51	create database cost_comparison_DATABASE	1 row(s) affected	0.000 sec
2	16:33:57	use cost_comparison_DATABASE	Error Code: 1049. Unknown database 'cost_comparison_DATABASE'.	0.000 sec
3	16:34:07	use cost_comparison_DATABASE	0 row(s) affected	0.000 sec
4	16:34:46	CREATE TABLE Country (Country_id INT AUTO_INCREMENT PRIMARY KEY, Country_name VARCHAR(255) UNIQUE NOT NULL)	0 row(s) affected	0.094 sec

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

```
7 Country_name VARCHAR(255) UNIQUE NOT NULL
8 );
9
10 • select * from country;
11
12 • CREATE TABLE Customers (
13     Customer_id INT AUTO_INCREMENT PRIMARY KEY,
14     Customer_Fname VARCHAR(100),
15     Customer_Lname VARCHAR(100),
16     Phone_Number VARCHAR(15),
17     Address VARCHAR(250), -- Changed the data type to VARCHAR(250)
18     Pincode VARCHAR(10),
19     Country_id INT,
20     CONSTRAINT fk_country FOREIGN KEY (Country_id) REFERENCES Country(Country_id)
21 );
22
23 • select * from customers
24     order by Country_id;
25
26 • CREATE INDEX idx_address ON Customers(Address);
27 • CREATE INDEX idx_pincode ON Customers(Pincode);
28 • CREATE INDEX idx_customer_id ON Customers(Customer_id);
29
30 -- Categories Entity
```

Output

#	Time	Action	Message	Duration / Fetch
4	16:34:46	CREATE TABLE Country (Country_id INT AUTO_INCREMENT PRIMARY KEY, Country_name VARCHAR(255) UNIQUE NOT NULL)	0 row(s) affected	0.094 sec
5	16:34:51	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), Pincode VARCHAR(10), Country_id INT, CONSTRAINT fk_country FOREIGN KEY (Country_id) REFERENCES Country(Country_id))	0 row(s) affected	0.109 sec
6	16:35:00	CREATE INDEX idx_address ON Customers(Address)	0 row(s) affected	0.047 sec
7	16:35:00	CREATE INDEX idx_pincode ON Customers(Pincode)	0 row(s) affected	0.046 sec

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

```
31 • CREATE TABLE Categories (
32     Category_id INT AUTO_INCREMENT PRIMARY KEY,
33     Category_Name VARCHAR(255) NOT NULL
34 );
35
36 -- Products Entity
37 • CREATE TABLE Products (
38     Product_id INT AUTO_INCREMENT PRIMARY KEY,
39     Category_id INT,
40     product_name varchar(100),
41     Product_Details TEXT,
42     FOREIGN KEY (Category_id) REFERENCES Categories(Category_id)
43 );
44
45 • select * from products;
46
47 -- Adding Inventory Entity
48 • CREATE TABLE Inventory (
49     Inventory_id INT AUTO_INCREMENT PRIMARY KEY,
50     Product_id INT,
51     Quantity INT,
52     country_id int,
53     Item_cost DECIMAL(10, 2),
54     FOREIGN KEY (Product_id) REFERENCES Products(Product_id),
55 );
```

Output

#	Time	Action	Message	Duration / Fetch
8	16:35:00	CREATE INDEX idx_customer_id ON Customers(Customer_id)	0 row(s) affected	0.032 sec
9	16:35:07	CREATE TABLE Categories (Category_id INT AUTO_INCREMENT PRIMARY KEY, Category_Name VARCHAR(255) NOT NULL)	0 row(s) affected	0.047 sec
10	16:36:00	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))	0 row(s) affected	0.062 sec
11	16:37:05	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),)	0 row(s) affected	0.078 sec

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

```
61
62 -- Orders Entity
63 • CREATE TABLE Orders(
64     Order_id INT AUTO_INCREMENT PRIMARY KEY,
65     product_id INT,
66     Quantity int,
67     foreign key (product_id) references products(product_id),
68     foreign key (quantity) references inventory(quantity)
69 );
70 • alter table orders add column item_cost Decimal(10,2);
71
72 -- Payments Entity
73 • CREATE TABLE Payments (
74     Payment_id INT AUTO_INCREMENT PRIMARY KEY,
75     Order_id INT,
76     Payment_Method VARCHAR(50),
77     Payment_Amount DECIMAL(10, 2),
78     FOREIGN KEY (Order_id) REFERENCES Orders(Order_id)
79 );
80 • alter table payments
81     drop column payment_method;
82
83 -- Adding Invoice Entity
84 • CREATE TABLE Invoices (
```

Output

#	Time	Action	Message	Duration / Fetch
17	16:44:55	CREATE INDEX idx_quantity ON Inventory(Quantity)	0 row(s) affected	0.047 sec
18	16:45:01	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))	0 row(s) affected	0.125 sec
19	16:45:12	CREATE TABLE Order_Details (Order_id INT, product_id INT, Quantity int, Item_cost DECIMAL(10,2), FOREIGN KEY (Order_id) REFERENCES Orders(Order_id), FOREIGN KEY (product_id) REFERENCES Products(Product_id),)	Error Code: 3780. Referencing column quantity in foreign key clause does not match column name in referenced table.	0.015 sec
20	16:47:07	CREATE TABLE Order_Details (Order_id INT, product_id INT, Quantity int, Item_cost DECIMAL(10,2), FOREIGN KEY (Order_id) REFERENCES Orders(Order_id), FOREIGN KEY (product_id) REFERENCES Products(Product_id),)	Error Code: 3780. Referencing column quantity in foreign key clause does not match column name in referenced table.	0.016 sec

SQL File 28* Query 1*

```
69 )
70 alter table orders add column item_cost Decimal(10,2);
71
72 -- Payments Entity
73 CREATE TABLE Payments (
74     Payment_id INT AUTO_INCREMENT PRIMARY KEY,
75     Order_id INT,
76     Payment_Method VARCHAR(50),
77     Payment_Amount DECIMAL(10, 2),
78     FOREIGN KEY (Order_id) REFERENCES Orders(Order_id)
79 );
80 alter table payments
81 drop column payment_method;
82
83 -- Adding Invoice Entity
84 CREATE TABLE Invoices (
85     Invoice_id INT AUTO_INCREMENT PRIMARY KEY,
86     Order_id INT,
87     Total_cost DECIMAL(10, 2),
88     Purchased_Date DATE,
89     FOREIGN KEY (Order_id) REFERENCES Orders(Order_id)
90 );
91 show tables;
92 select * from inventory;
```

Result Grid

Tables in cost_comparison_database
categories
country
customers
inventory
invoices
orders
payments
products

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 (0 row(s) affected	0.063 sec
23	16:47:46	CREATE TABLE Invoices (0 row(s) affected	0.078 sec
24	16:47:50	show tables	10 row(s) returned	0.000 sec / 0.000 sec

Query 1*

```
1 INSERT INTO Country (Country_name) VALUES ('India'), ('China'), ('Europe'), ('Australia'), ('USA');
2
3 INSERT INTO Customers (Customer_Fname, Customer_Lname, Phone_Number, Address, Pincode, Country_id) VALUE
4 ('John', 'Doe', '1234567890', '123 Main St', '00001', (SELECT Country_id FROM Country WHERE Country_name
5 ('Jane', 'Doe', '1234567891', '124 Main St', '00002', (SELECT Country_id FROM Country WHERE Country_name
6 ('Raj', 'Kumar', '1234567892', '125 Main St', '00003', (SELECT Country_id FROM Country WHERE Country_nam
7 ('Amit', 'Shah', '1234567893', '126 Main St', '00004', (SELECT Country_id FROM Country WHERE Country_nam
8 ('Li', 'Wang', '1234567894', '127 Main St', '00005', (SELECT Country_id FROM Country WHERE Country_name
9 ('Xiu', 'Chen', '1234567895', '128 Main St', '00006', (SELECT Country_id FROM Country WHERE Country_name
10 ('Lucas', 'Müller', '1234567896', '129 Main St', '00007', (SELECT Country_id FROM Country WHERE Country_
11 ('Emma', 'Schmidt', '1234567897', '130 Main St', '00008', (SELECT Country_id FROM Country WHERE Country_
12 ('Ethan', 'Brown', '1234567898', '131 Main St', '00009', (SELECT Country_id FROM Country WHERE Country_r
13 ('Olivia', 'Smith', '1234567899', '132 Main St', '00010', (SELECT Country_id FROM Country WHERE Country_
14
15 INSERT INTO Categories (category_name) VALUES ('Electronics');
16
17 -- Inserting electronic products
18 INSERT INTO Products (category_id, product_name, product_details) VALUES
19 ((SELECT category_id FROM categories WHERE category_name = 'Electronics'), 'iPhone 12', '128GB, 6.1" Sup
20 ((SELECT category_id FROM categories WHERE category_name = 'Electronics'), 'Samsung Galaxy S21', '256GB,
21 ((SELECT category_id FROM categories WHERE category_name = 'Electronics'), 'Sony PlayStation 5', '825GB
22 ((SELECT category_id FROM categories WHERE category_name = 'Electronics'), 'Apple MacBook Pro', '13" Ret
23 ((SELECT category_id FROM categories WHERE category_name = 'Electronics'), 'DJI Mavic Air 2', '48MP came
24 ((SELECT category_id FROM categories WHERE category_name = 'Electronics'), 'Canon EOS R5', '45MP full-fr
```

Action Output

#	Time	Action	Message	Duration / Fetch
27	16:49:02	INSERT INTO Categories (category_name) VALUES ('E...	1 row(s) affected	0.016 sec
28	16:49:02	INSERT INTO Products (category_id, product_name, pr...	25 row(s) affected Records: 25 Duplicates: 0 Warnings: 0	0.015 sec
29	16:50:17	INSERT INTO Inventory (Product_id, Country_id, Quanti...	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

details

product table

Inventory

1

-- Insert products

2

INSERT INTO Inventory

3

VALUES

4

-- iPhone 12

5

(1, 1, 100, 800

6

-- Samsung Gala

7

(2, 1, 150, 700

8

-- Sony PlaySta

9

(3, 1, 80, 1000

10

-- Apple MacBoo

11

(4, 1, 120, 150

12

-- DJI Mavic Ai

13

(5, 1, 50, 900.

14

-- Canon EOS R5

15

(6, 1, 70, 2500

16

-- Bose QuietCo

17

(7, 1, 90, 300.

18

-- LG CX OLED T

19

(8, 1, 40, 2000

20

-- Nintendo Swi

21

(9, 1, 200, 400

22

-- Google Pixel

Output

Action Output

#

Time

Action

36

13:34:34

INSERT INTO Inve

37

13:34:57

INSERT INTO Inve

38

13:36:16

INSERT INTO Inve

39

13:36:54

INSERT INTO Inve

40

13:37:28

INSERT INTO Inve

41

13:38:22

select * from invent

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)

inserting products

Duration / Fetch

0 Wa... 0.016 sec

0 Wa... 0.016 sec

0 Wa... 0.016 sec

0 Wa... 0.016 sec

0 Wa... 0.016 sec

0.000 sec / 0.000 sec

inserting products*

Table created 1*

Inserting customer details

inserting orders.payments, invo...

Query 1*

Inserting Data ccd-1

1

-- Inserting data into data table

2

3

INSERT INTO Orders (item_cost, product_id, Quantity)

4

SELECT item_cost, product_id, FLOOR(1 + (RAND() * 5)) AS Quantity

5

FROM inventory

6

ORDER BY RAND()

7

LIMIT 25;

8

9

-- Inserting data into payment table

10

INSERT INTO Payments (order_id, payment_amount)

11

SELECT o.Order_id, o.Quantity * o.item_cost AS payment_amount

12

FROM Orders o;

13

14

select * from invoices;

15

16

-- Inserting data into invoice table

17

INSERT INTO Invoices (order_id, Total_cost, Purchased_Date)

18

SELECT

19

o.order_id,

20

(o.Quantity * o.item_cost) AS Total_cost,

21

DATE_ADD('2024-01-01', INTERVAL FLOOR(RAND() * 365) DAY) AS Purchased_Date

22

--

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 i 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:

```
1 • select i.Purchased_Date, i.Total_cost, p.product_id, p.product_name
2   from invoices i
3   join orders o on i.order_id = o.order_id
4   join products p on o.product_id = p.product_id
5   where Purchased_Date < '2024-04-04';
6
```

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

Purchased_Date	Total_cost	product_id	product_name
2024-02-27	60.00	12	Amazon Echo Dot (4th Gen)
2024-02-28	480.00	17	Roku Ultra
2024-01-19	570.00	24	Ring Video Doorbell 3
2024-02-18	2500.00	9	Nintendo Switch
2024-03-05	1800.00	15	Xbox Series X
2024-02-27	4500.00	1	iPhone 12
2024-03-21	2800.00	2	Samsung Galaxy S21
2024-01-04	4000.00	19	Samsung Odyssey G7
2024-02-02	2400.00	3	Sony PlayStation 5

Result 19 x

Output

Action Output

#	Time	Action	Message
---	------	--------	---------

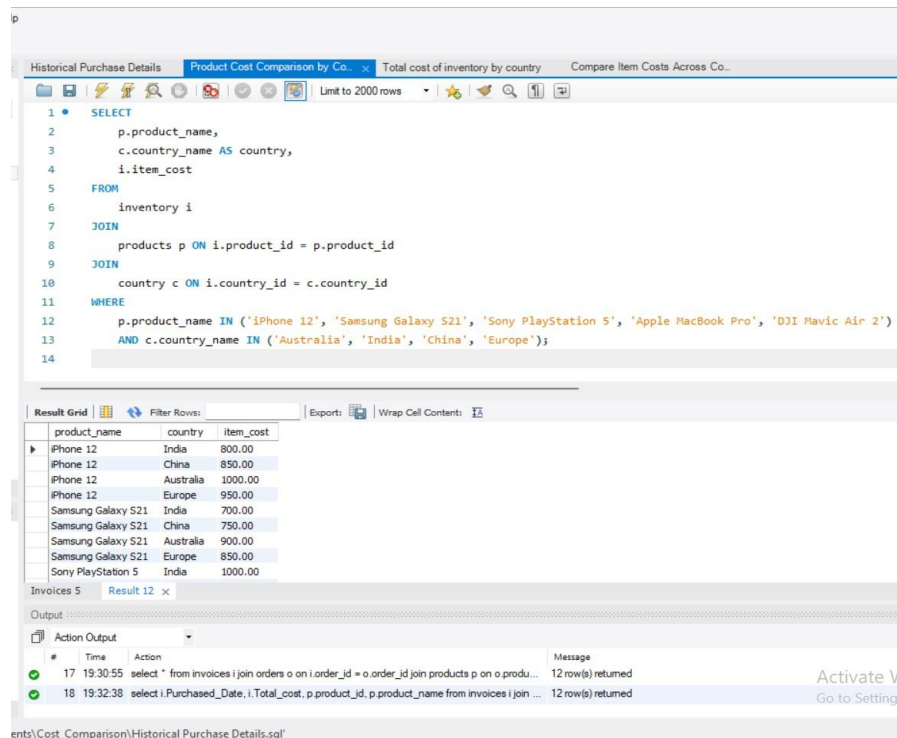
nts\Cost_Comparison\Total cost of inventory by country.sql'

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:



The screenshot shows a SQL query editor with the following query:

```
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');
```

The results are displayed in a table with the following data:

product_name	country	item_cost
iPhone 12	India	800.00
iPhone 12	China	850.00
iPhone 12	Australia	1000.00
iPhone 12	Europe	950.00
Samsung Galaxy S21	India	700.00
Samsung Galaxy S21	China	750.00
Samsung Galaxy S21	Australia	900.00
Samsung Galaxy S21	Europe	850.00
Sony PlayStation 5	India	1000.00

The bottom of the screenshot shows the "Output" section with the following message:

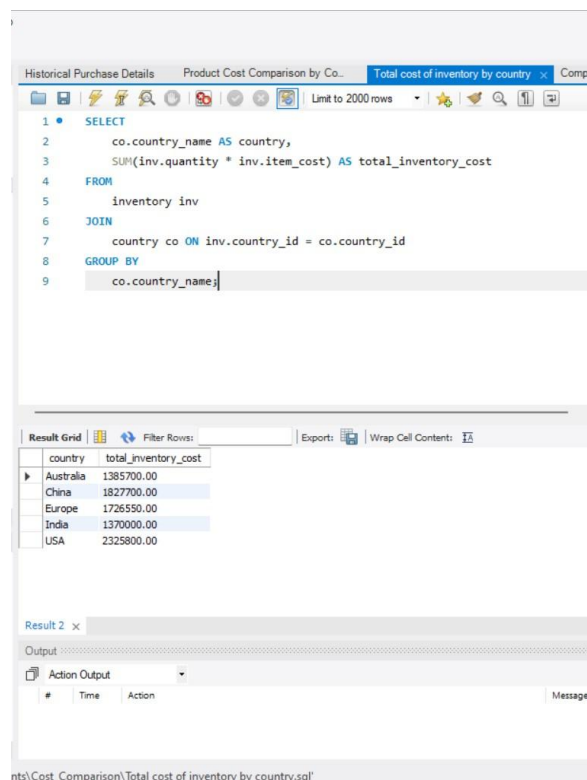
```
select * from invoices i join orders o on i.order_id = o.order_id join products p on o.produ... 12 row(s) returned
```

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:



The screenshot shows a SQL query editor with the following query:

```
1 SELECT
2     co.country_name AS country,
3     SUM(inv.quantity * inv.item_cost) AS total_inventory_cost
4 FROM
5     inventory inv
6 JOIN
7     country co ON inv.country_id = co.country_id
8 GROUP BY
9     co.country_name;
```

Below the query, the results are displayed in a table with the following data:

country	total_inventory_cost
Australia	1385700.00
China	1827700.00
Europe	1726550.00
India	1370000.00
USA	2325800.00

The screenshot also shows a "Result 2" tab with an "Output" section and an "Action Output" table.

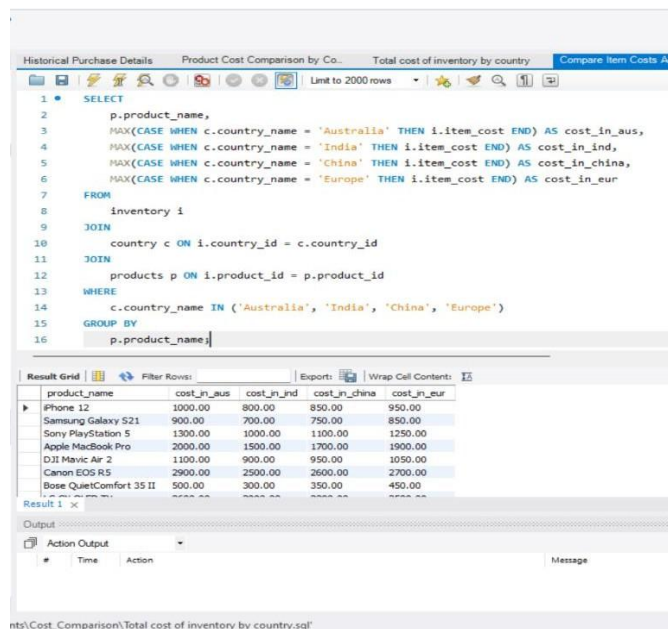
4. Query 4

```
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:



The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, a search icon, and a 'Limit to 2000 rows' dropdown. The SQL editor contains the query from the previous block. Below the editor, the 'Result Grid' tab is active, displaying the query results. The results are organized into columns: product_name, cost_in_aus, cost_in_ind, cost_in_china, and cost_in_eur. The data is sorted by product_name. Below the result grid, there is an 'Output' section with a dropdown menu set to 'Action Output' and a 'Message' column.

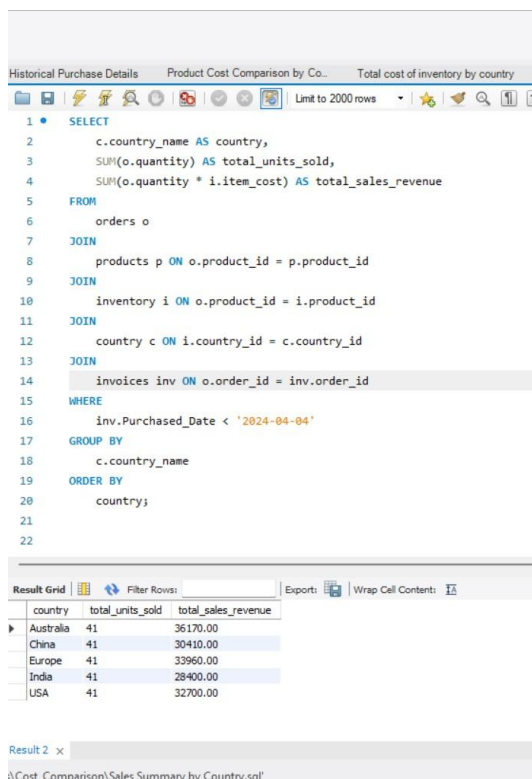
product_name	cost_in_aus	cost_in_ind	cost_in_china	cost_in_eur
iPhone 12	1000.00	800.00	850.00	950.00
Samsung Galaxy S21	900.00	700.00	750.00	850.00
Sony PlayStation 5	1300.00	1000.00	1100.00	1250.00
Apple MacBook Pro	2000.00	1500.00	1700.00	1900.00
DJI Mavic Air 2	1100.00	900.00	950.00	1050.00
Canon EOS R5	2900.00	2500.00	2600.00	2700.00
Bose QuietComfort 35 II	500.00	300.00	350.00	450.00

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:



The screenshot shows a SQL IDE interface. At the top, there are three tabs: "Historical Purchase Details", "Product Cost Comparison by Co...", and "Total cost of inventory by country". The "Product Cost Comparison by Co..." tab is active. Below the tabs is a toolbar with various icons and a "Limit to 2000 rows" dropdown. The main area displays the SQL query from the previous block. Below the query editor, there is a "Result Grid" section. It includes a "Filter Rows:" input field, an "Exports:" button, and a "Wrap Cell Contents:" checkbox. The result grid shows a table with three columns: "country", "total_units_sold", and "total_sales_revenue". The data is as follows:

country	total_units_sold	total_sales_revenue
Australia	41	36170.00
China	41	30410.00
Europe	41	33960.00
India	41	28400.00
USA	41	32700.00

At the bottom of the screenshot, there is a "Result 2" tab with a close button (X). Below it, the file path is shown: "A:\Cost_Comparison\Sales Summary by Country.sql".