Roll No. 22DEC02	**Experiment No.** 02	**Marks :**
BATCH - C		**Sign :**

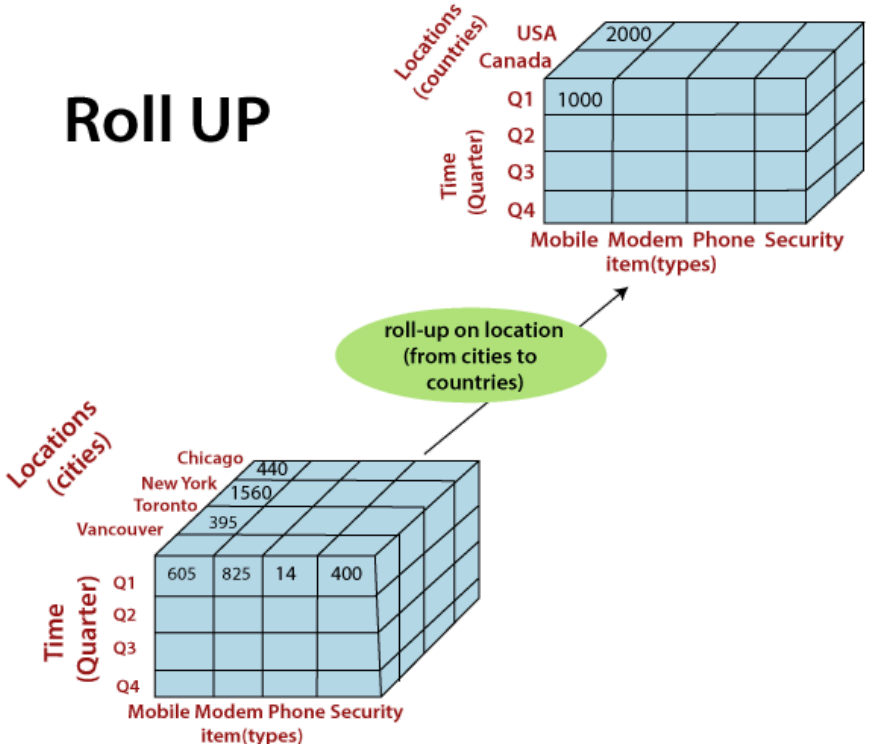
Aim: Implementation of OLAP operations:
Slice, Dice, Rollup, Drilldown and Pivot for the above problem statement

Apparatus: XAMPP Server

Theory:

Roll-Up

The roll-up operation (also known as drill-up or aggregation operation) performs aggregation on a data cube, by climbing down concept hierarchies, i.e., dimension reduction. Roll-up is like zooming-out on the data cubes. Figure shows the result of roll-up operations performed on the dimension location. The hierarchy for the location is defined as the Order Street, city, province, or state, country. The roll-up operation aggregates the data by ascending the location hierarchy from the level of the city to the level of the country.

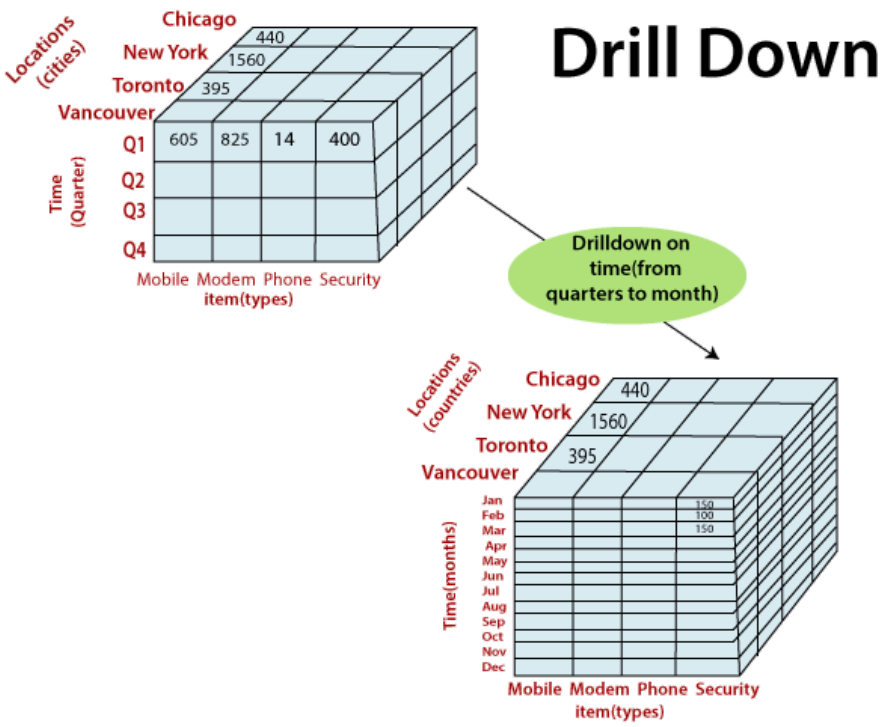


When a roll-up is performed by dimensions reduction, one or more dimensions are removed from the cube. For example, consider a sales data cube having two dimensions, location and time. Roll-up may be performed by removing, the time dimensions, appearing in an aggregation of the total sales by location, relatively than by location and by time.

Drill-Down

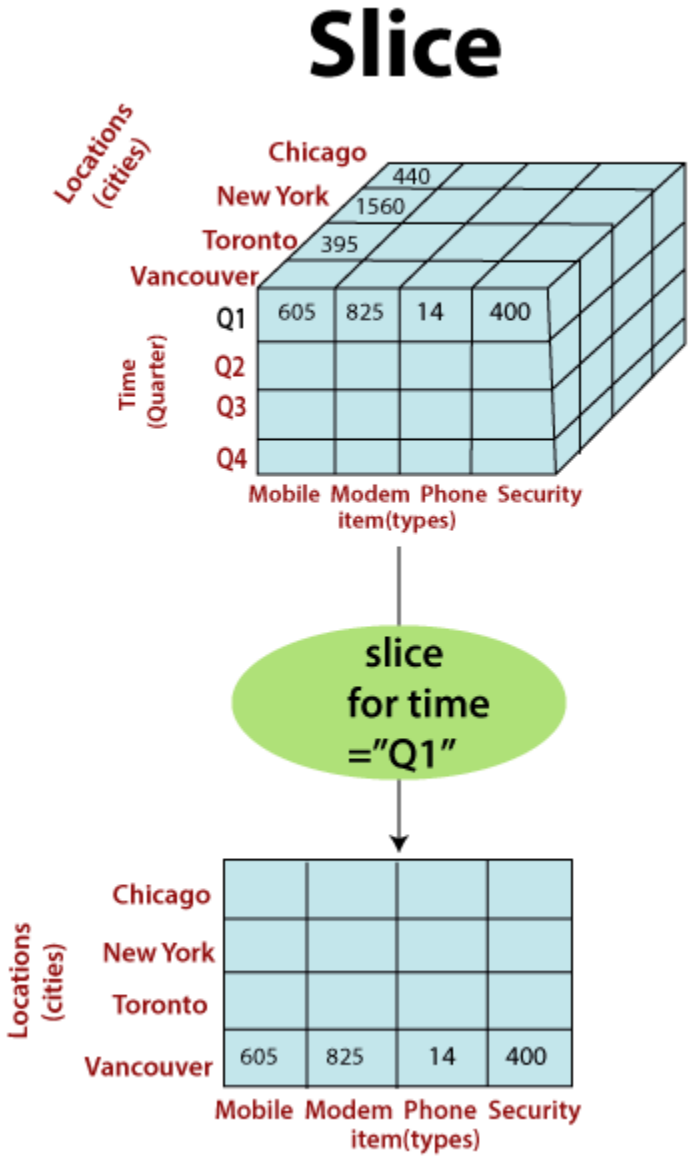
The drill-down operation (also called roll-down) is the reverse operation of roll-up. Drill-down is like zooming-in on the data cube. It navigates from less detailed records to more detailed data. Drill-down can be performed by either stepping down a concept hierarchy for a dimension or adding additional dimensions.

Figure shows a drill-down operation performed on the dimension time by stepping down a concept hierarchy which is defined as day, month, quarter, and year. Drill-down appears by descending the time hierarchy from the level of the quarter to a more detailed level of the month. Because a drill-down adds more details to the given data, it can also be performed by adding a new dimension to a cube. For example, a drill-down on the central cubes of the figure can occur by introducing an additional dimension, such as a customer group.



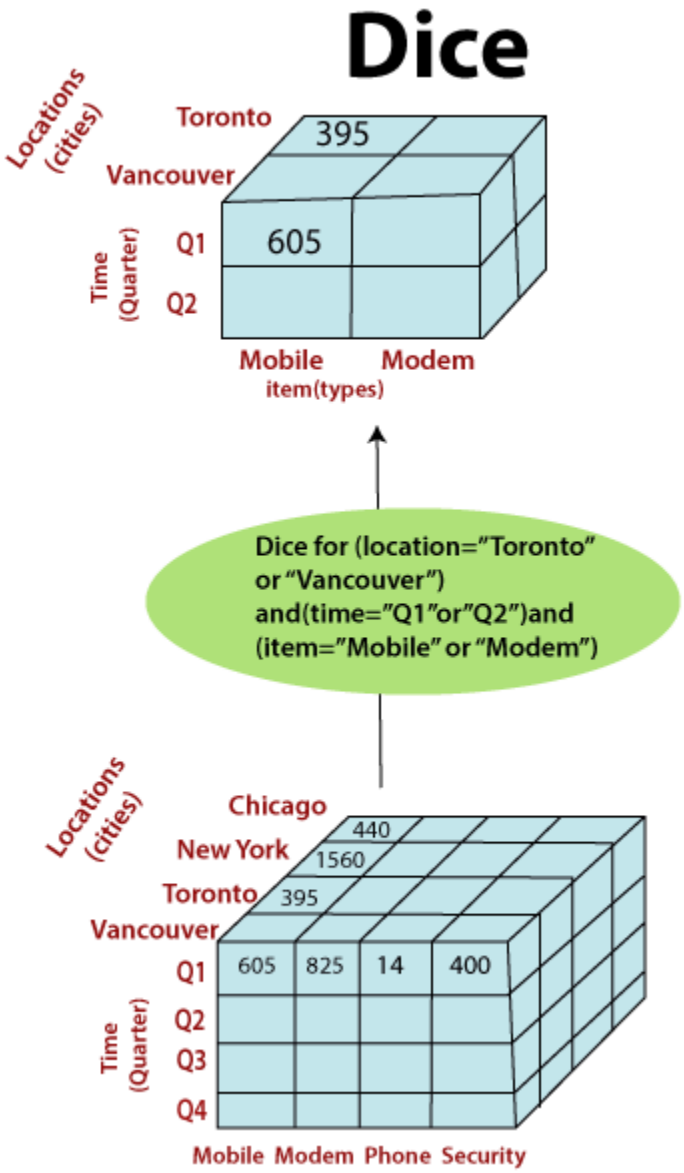
Slice

A slice is a subset of the cubes corresponding to a single value for one or more members of the dimension. For example, a slice operation is executed when the customer wants a selection on one dimension of a three-dimensional cube resulting in a two-dimensional site. So, the Slice operations perform a selection on one dimension of the given cube, thus resulting in a subcube.



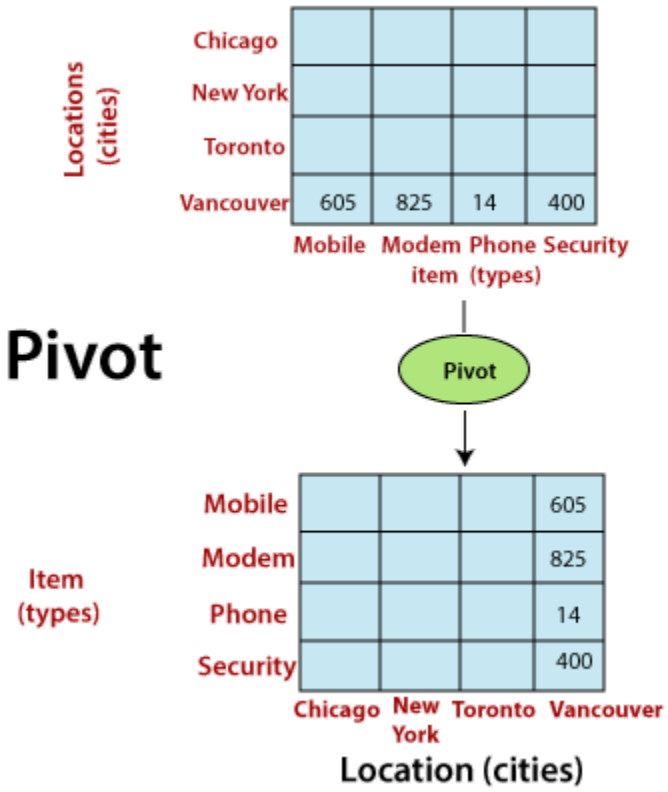
Dice

The dice operation describes a subcube by operating a selection on two or more dimension**.**



Pivot

The pivot operation is also called a rotation. Pivot is a visualization operations which rotates the data axes in view to provide an alternative presentation of the data. It may contain swapping the rows and columns or moving one of the row-dimensions into the column dimensions.



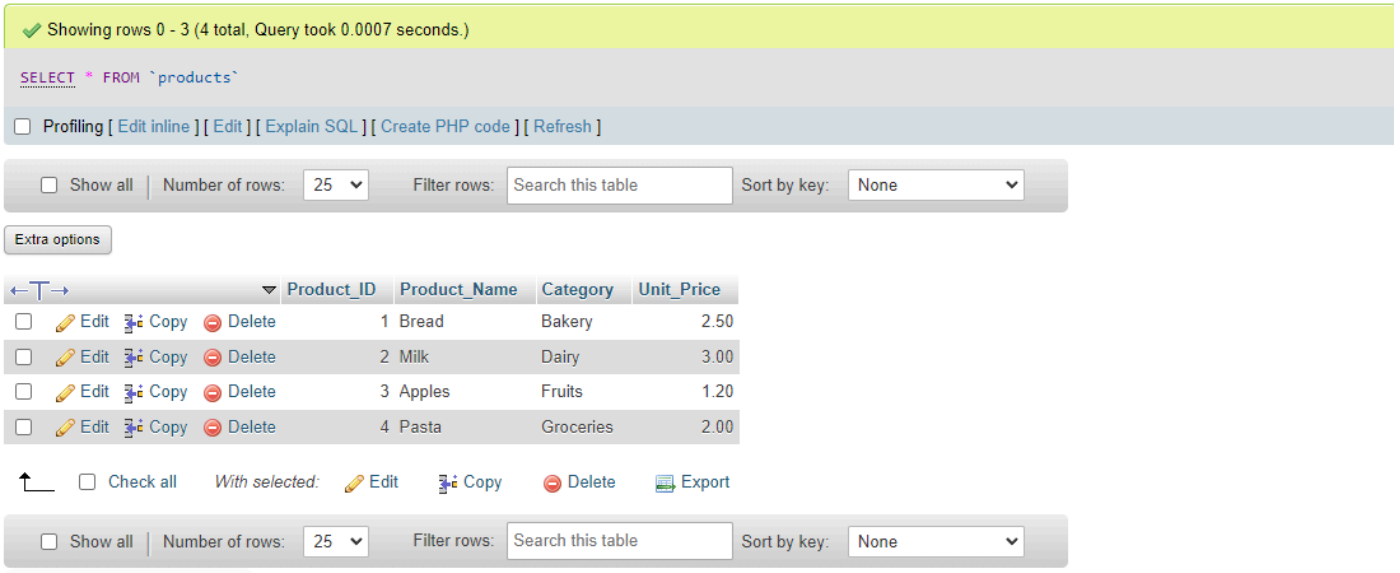
Conclusion:

In conclusion, the experiment illustrates the effectiveness of OLAP operations in analyzing and exploring multidimensional data. By employing operations such as Slice, Dice, Rollup, Drilldown, and Pivot, users can gain valuable insights into their data from various perspectives. These operations enable dynamic analysis and facilitate deeper understanding of data relationships, leading to informed decision-making and strategic planning.

SQL Query

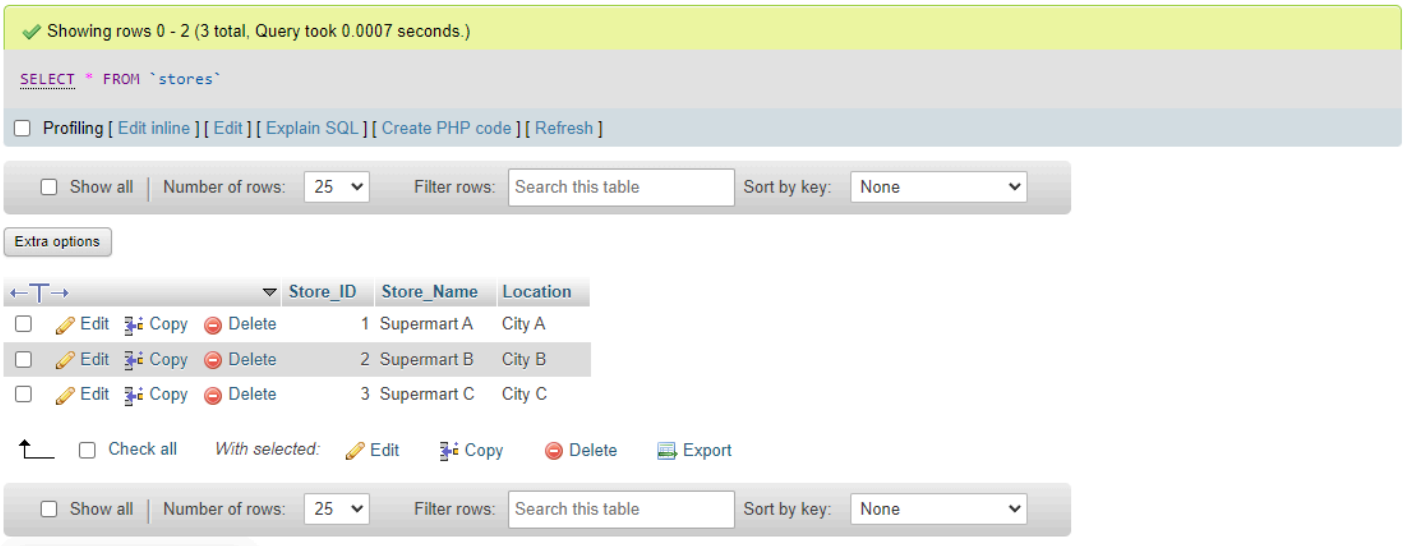
Products Table:

```
CREATE TABLE Products (
    Product_ID INT PRIMARY KEY,
    Product_Name VARCHAR(255),
    Category VARCHAR(255),
    Unit_Price DECIMAL(10, 2)
);

INSERT INTO Products (Product_ID, Product_Name, Category, Unit_Price)
VALUES
    (1, 'Bread', 'Bakery', 2.50),
    (2, 'Milk', 'Dairy', 3.00),
    (3, 'Apples', 'Fruits', 1.20),
    (4, 'Pasta', 'Groceries', 2.00);
```



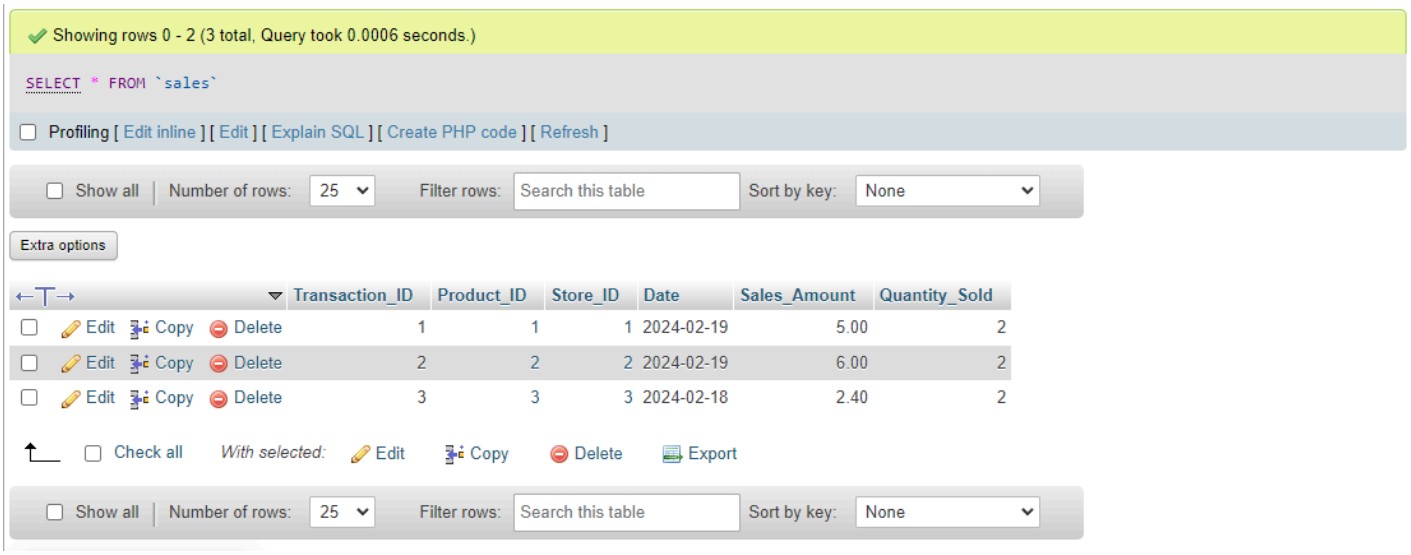
Stores Table:

```
CREATE TABLE Stores (
    Store_ID INT PRIMARY KEY,
    Store_Name VARCHAR(255),
    Location VARCHAR(255)
);

INSERT INTO Stores (Store_ID, Store_Name, Location)
VALUES
    (1, 'Supermart A', 'City A'),
    (2, 'Supermart B', 'City B'),
    (3, 'Supermart C', 'City C');
```



Sales Table:

```
CREATE TABLE Sales (
    Transaction_ID INT PRIMARY KEY,
    Product_ID INT,
    Store_ID INT,
    Date DATE,
    Sales_Amount DECIMAL(10, 2),
    Quantity_Sold INT,
    FOREIGN KEY (Product_ID) REFERENCES Products(Product_ID),
    FOREIGN KEY (Store_ID) REFERENCES Stores(Store_ID)
);

INSERT INTO Sales (Transaction_ID, Product_ID, Store_ID, Date, Sales_Amount, Quantity_Sold)
VALUES
    (1, 1, 1, '2024-02-19', 5.00, 2),
    (2, 2, 2, '2024-02-19', 6.00, 2),
    (3, 3, 3, '2024-02-18', 2.40, 2);
```



OUTPUT

Roll Up:

Run SQL query/queries on database supermarket:

```
1 SELECT p.Category, SUM(s.Sales_Amount) AS Total_Sales
2 FROM Sales s
3 JOIN Products p ON s.Product_ID = p.Product_ID
4 GROUP BY p.Category;
```

Showing rows 0 - 2 (3 total, Query took 0.0012 seconds.)

```
SELECT p.Category, SUM(s.Sales_Amount) AS Total_Sales FROM Sales s JOIN Products p ON s.Product_ID = p.Product_ID GROUP BY p.Category;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

Category	Total_Sales
Bakery	5.00
Dairy	6.00
Fruits	2.40

Slice:

Run SQL query/queries on database supermarket:

```
1 SELECT s.Date, p.Product_Name, s.Sales_Amount, s.Quantity_Sold
2 FROM Sales s
3 JOIN Products p ON s.Product_ID = p.Product_ID
4 WHERE s.Date = '2024-02-19';
5
```

Showing rows 0 - 1 (2 total, Query took 0.0009 seconds.)

```
SELECT s.Date, p.Product_Name, s.Sales_Amount, s.Quantity_Sold FROM Sales s JOIN Products p ON s.Product_ID = p.Product_ID WHERE s.Date = '2024-02-19';
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

Date	Product_Name	Sales_Amount	Quantity_Sold
2024-02-19	Bread	5.00	2
2024-02-19	Milk	6.00	2

Dice:

Run SQL query/queries on database supermarket:

```
SELECT s.Date, p.Product_Name, s.Sales_Amount, s.Quantity_Sold
FROM Sales s
JOIN Products p ON s.Product_ID = p.Product_ID
JOIN Stores st ON s.Store_ID = st.Store_ID
WHERE p.Category = 'Dairy' AND st.Store_Name = 'Supermart B';
```

Showing rows 0 - 0 (1 total, Query took 0.0008 seconds.)

```
SELECT s.Date, p.Product_Name, s.Sales_Amount, s.Quantity_Sold FROM Sales s JOIN Products p ON s.Product_ID = p.Product_ID JOIN Stores st ON s.Store_ID = st.Store_ID WHERE p.Category = 'Dairy' AND st.Store_Name = 'Supermart B';
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

Date	Product_Name	Sales_Amount	Quantity_Sold
2024-02-19	Milk	6.00	2

Drilldown:

Run SQL query/queries on database supermarket:

```
SELECT s.Date, p.Product_Name, s.Sales_Amount, s.Quantity_Sold
FROM Sales s
JOIN Products p ON s.Product_ID = p.Product_ID;
```

Showing rows 0 - 2 (3 total, Query took 0.0012 seconds.)

```
SELECT s.Date, p.Product_Name, s.Sales_Amount, s.Quantity_Sold FROM Sales s JOIN Products p ON s.Product_ID = p.Product_ID;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

Date	Product_Name	Sales_Amount	Quantity_Sold
2024-02-19	Bread	5.00	2
2024-02-19	Milk	6.00	2
2024-02-18	Apples	2.40	2

Pivot:

Run SQL query/queries on database supermarket:

```
SELECT
    Date,
    SUM(CASE WHEN Category = 'Bakery' THEN Sales_Amount ELSE 0 END) AS Bakery_Sales,
    SUM(CASE WHEN Category = 'Dairy' THEN Sales_Amount ELSE 0 END) AS Dairy_Sales,
    SUM(CASE WHEN Category = 'Fruits' THEN Sales_Amount ELSE 0 END) AS Fruits_Sales,
    SUM(CASE WHEN Category = 'Groceries' THEN Sales_Amount ELSE 0 END) AS Groceries_Sales
FROM Sales
JOIN Products ON Sales.Product_ID = Products.Product_ID
GROUP BY Date;
```

Showing rows 0 - 1 (2 total, Query took 0.0009 seconds.)

```
SELECT Date, SUM(CASE WHEN Category = 'Bakery' THEN Sales_Amount ELSE 0 END) AS Bakery_Sales, SUM(CASE WHEN Category = 'Dairy' THEN Sales_Amount ELSE 0 END) AS Dairy_Sales, SUM(CASE WHEN Category = 'Fruits' THEN Sales_Amount ELSE 0 END) AS Fruits_Sales, SUM(CASE WHEN Category = 'Groceries' THEN Sales_Amount ELSE 0 END) AS Groceries_Sales FROM Sales JOIN Products ON Sales.Product_ID = Products.Product_ID GROUP BY Date;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

Date	Bakery_Sales	Dairy_Sales	Fruits_Sales	Groceries_Sales
2024-02-18	0.00	0.00	2.40	0.00
2024-02-19	5.00	6.00	0.00	0.00