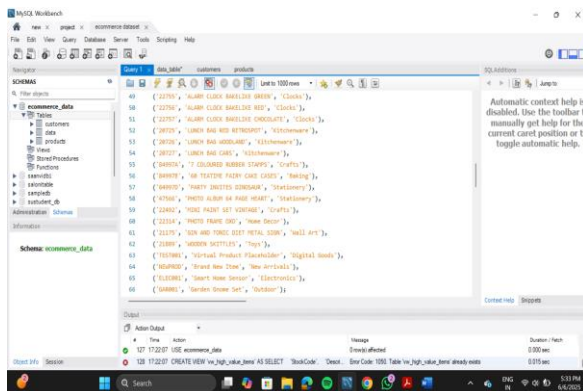
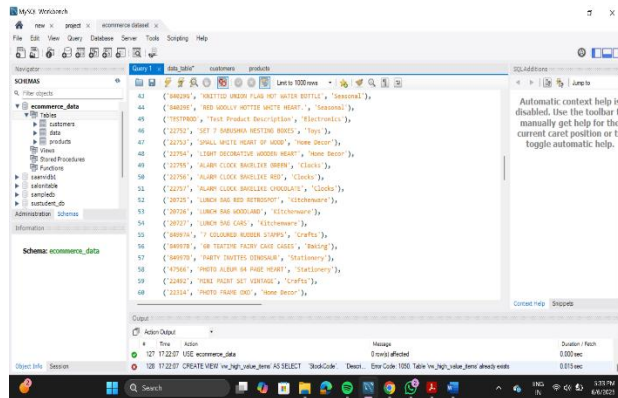
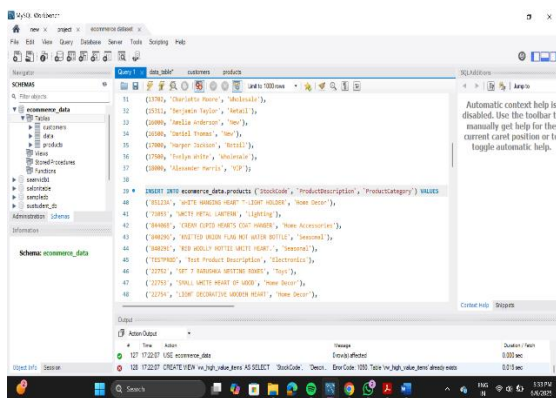
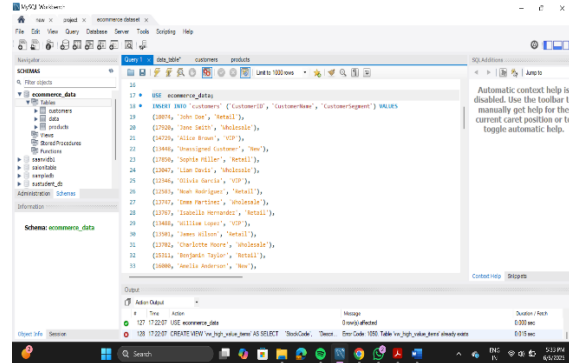
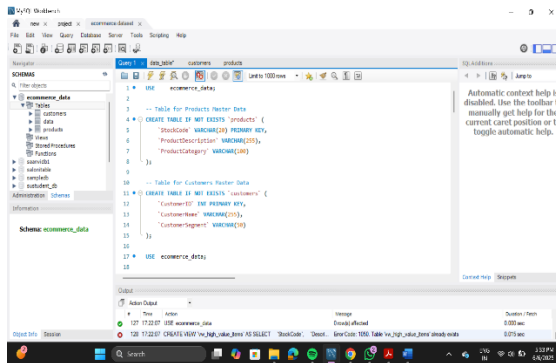


# SQL QUERIES ON ECOMMECE DATASET



-- Top 10 Products by Total Sales Revenue  
SELECT

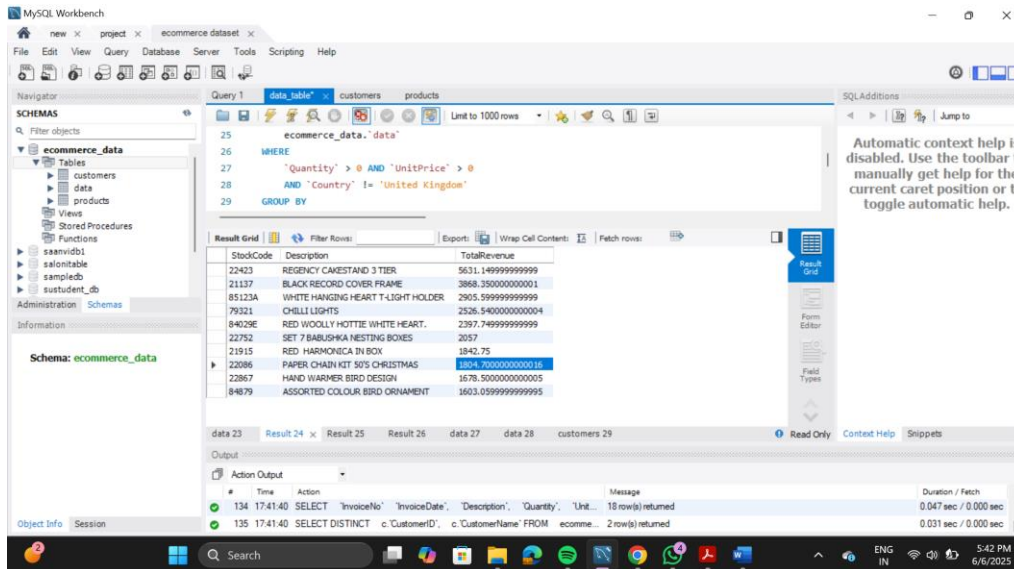
'StockCode',

'Description',

```

SUM(`Quantity` * `UnitPrice`) AS TotalRevenue
FROM
ecommerce_data.`data`
WHERE
Quantity > 0 AND UnitPrice > 0
GROUP BY
`StockCode`,
`Description`
ORDER BY
TotalRevenue DESC
LIMIT 10;

```



-- Monthly Sales Performance and Number of Unique Orders by Country (Excluding UK)

```

SELECT
`Country`,
SUM(`Quantity` * `UnitPrice`) AS MonthlyRevenue,
COUNT(DISTINCT `InvoiceNo`) AS NumberOfOrders

```

FROM

ecommerce\_data.`data`

WHERE

`Quantity` > 0 AND `UnitPrice` > 0

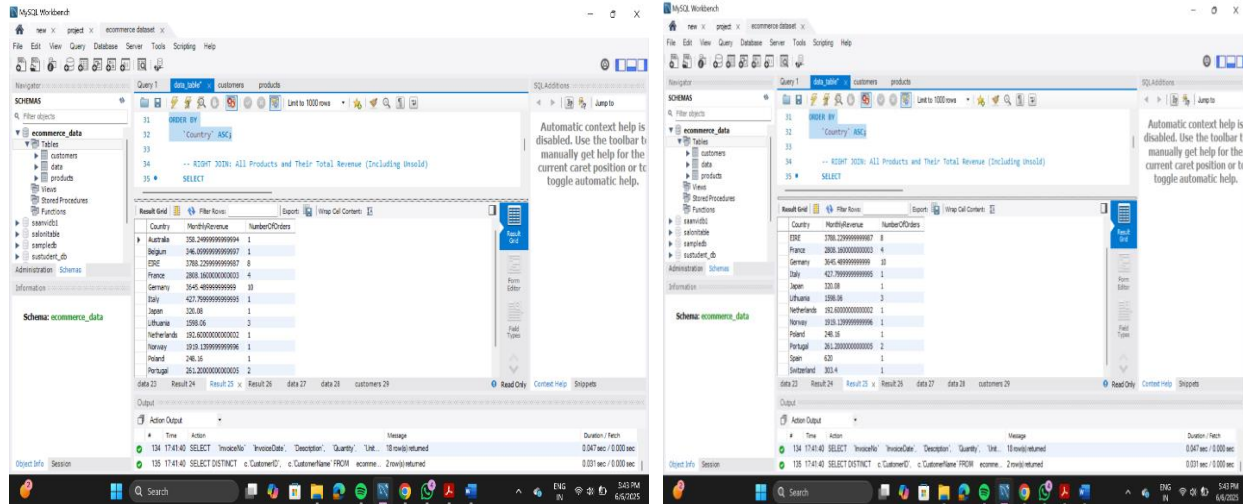
AND `Country` != 'United Kingdom'

GROUP BY

`Country`

ORDER BY

`Country` ASC;



SELECT \* FROM ecommerce\_data.`products`;

-- INNER JOIN: Total Sales Revenue by Product Category

SELECT

p.`ProductCategory`,

SUM(o.`Quantity` \* o.`UnitPrice`) AS TotalCategoryRevenue

FROM

ecommerce\_data.`data` AS o

INNER JOIN

```
`products` AS p ON o.`StockCode` = p.`StockCode`
```

WHERE

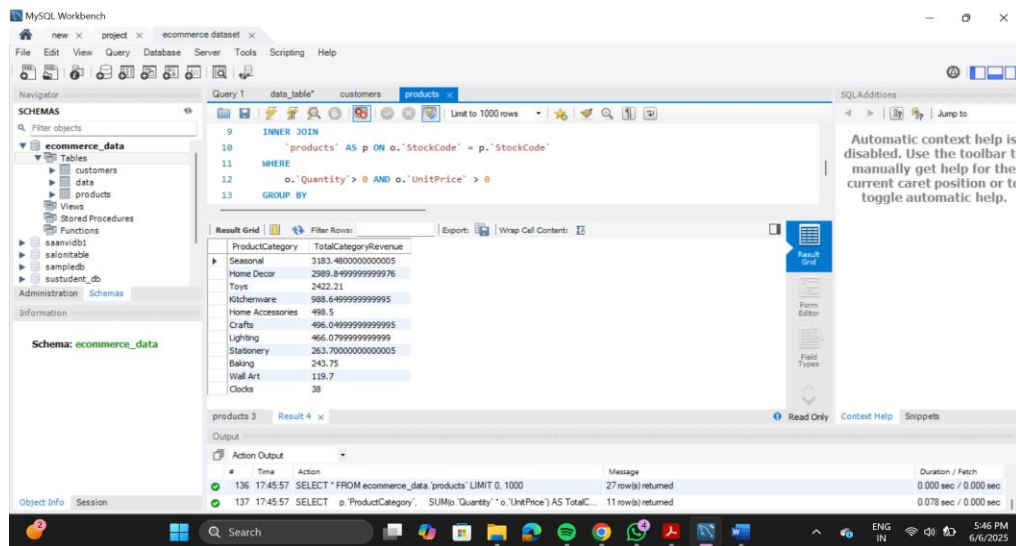
```
o.`Quantity` > 0 AND o.`UnitPrice` > 0
```

GROUP BY

```
p.`ProductCategory`
```

ORDER BY

```
TotalCategoryRevenue DESC;
```



```
SELECT * FROM ecommerce_data.`customers`;
```

-- LEFT JOIN: All Customers and Their Total Spending

SELECT

```
c.`CustomerID`,
```

```
c.`CustomerName`,
```

```
c.`CustomerSegment`,
```

```
SUM(CASE WHEN o.`Quantity` > 0 AND o.`UnitPrice` > 0 THEN o.`Quantity` * o.`UnitPrice`  
ELSE 0 END) AS TotalSpending
```

FROM

```
ecommerce_data.`customers` AS c
```

LEFT JOIN

ecommerce\_data.`data` AS o ON c.`CustomerID` = o.`CustomerID`

GROUP BY

c.`CustomerID`,

c.`CustomerName`,

c.`CustomerSegment`

ORDER BY

TotalSpending DESC;

The image displays two screenshots of the Microsoft SQL Server Enterprise Manager interface. Both screenshots show a query window with a SQL query and its results. The left screenshot shows a query for the 'data' table, and the right screenshot shows a query for the 'products' table. Both queries are ordered by 'TotalSpending' in descending order. The results are displayed in a table format with columns for CustomerID, CustomerName, CustomerSegment, and TotalSpending. The left screenshot shows results for the 'data' table, and the right screenshot shows results for the 'products' table.

CustomerID	CustomerName	CustomerSegment	TotalSpending
1363	Alpha Miller	Retail	591.220000000000
1297	David Hernandez	Retail	368.599999999999
1551	Benjamin Taylor	Retail	1.05500000000000
1263	Heath Rodriguez	Retail	855.860000000000
1763	Jane Smith	Wholesale	515.400000000000
1874	John Doe	Retail	489.800000000000
1248	Unassigned Customer	New	445.899999999999
1347	Liam Davis	Wholesale	366.63
1473	Alisa Brown	VP	315.400000000000
1297	Bruce Taylor	Wholesale	79.6
1296	Olivia Garcia	VP	0
1248	William Lopez	VP	0
1281	Jane Wilson	Retail	0
1272	Charlotte Moore	Wholesale	0
1880	Amanda Anderson	New	0
1880	David Thomas	New	0
1700	Harper Jackson	Retail	0
1700	Emily White	Wholesale	0
1880	Alexander Harris	VP	0

-- RIGHT JOIN: All Products and Their Total Revenue (Including Unsold)

SELECT

p.`StockCode`,

p.`ProductDescription`,

p.`ProductCategory`,

SUM(CASE WHEN o.`Quantity` > 0 AND o.`UnitPrice` > 0 THEN o.`Quantity` \* o.`UnitPrice`  
ELSE 0 END) AS TotalProductRevenue

FROM

ecommerce\_data.`data` AS o

RIGHT JOIN

```
ecommerce_data.`products` AS p ON o.`StockCode` = p.`StockCode`
```

GROUP BY

```
p.`StockCode`,
```

```
p.`ProductDescription`,
```

```
p.`ProductCategory`
```

ORDER BY

```
TotalProductRevenue DESC;
```

The screenshot shows the MySQL Workbench interface. The query window contains the following SQL query:

```
SELECT o.`InvoiceNo`,
       o.`Quantity`,
       p.`ProductDescription`,
       p.`ProductCategory`,
       o.`TotalProductRevenue`
FROM ecommerce_data.`orders` AS o
JOIN ecommerce_data.`products` AS p ON o.`StockCode` = p.`StockCode`
ORDER BY o.`TotalProductRevenue` DESC;
```

The results grid displays the following data:

InvoiceNo	Quantity	ProductDescription	ProductCategory	TotalProductRevenue
1000000000000000000	1	WHITE KITCHEN KAT FLOUT FLOUT	Home Decor	290.0
1000000000000000000	1	RED KITCHEN KAT FLOUT FLOUT	Home Decor	290.0
1000000000000000000	1	WHITE KITCHEN KAT FLOUT FLOUT	Home Decor	290.0
1000000000000000000	1	WHITE KITCHEN KAT FLOUT FLOUT	Home Decor	290.0
1000000000000000000	1	WHITE KITCHEN KAT FLOUT FLOUT	Home Decor	290.0
1000000000000000000	1	WHITE KITCHEN KAT FLOUT FLOUT	Home Decor	290.0
1000000000000000000	1	WHITE KITCHEN KAT FLOUT FLOUT	Home Decor	290.0
1000000000000000000	1	WHITE KITCHEN KAT FLOUT FLOUT	Home Decor	290.0
1000000000000000000	1	WHITE KITCHEN KAT FLOUT FLOUT	Home Decor	290.0
1000000000000000000	1	WHITE KITCHEN KAT FLOUT FLOUT	Home Decor	290.0

The screenshot shows the MySQL Workbench interface. The query window contains the following SQL query:

```
SELECT o.`InvoiceNo`,
       o.`Quantity`,
       p.`ProductDescription`,
       p.`ProductCategory`,
       o.`TotalProductRevenue`
FROM ecommerce_data.`orders` AS o
JOIN ecommerce_data.`products` AS p ON o.`StockCode` = p.`StockCode`
ORDER BY o.`TotalProductRevenue` DESC;
```

The results grid displays the following data:

InvoiceNo	Quantity	ProductDescription	ProductCategory	TotalProductRevenue
1000000000000000000	1	WHITE KITCHEN KAT FLOUT FLOUT	Home Decor	290.0
1000000000000000000	1	RED KITCHEN KAT FLOUT FLOUT	Home Decor	290.0
1000000000000000000	1	WHITE KITCHEN KAT FLOUT FLOUT	Home Decor	290.0
1000000000000000000	1	WHITE KITCHEN KAT FLOUT FLOUT	Home Decor	290.0
1000000000000000000	1	WHITE KITCHEN KAT FLOUT FLOUT	Home Decor	290.0
1000000000000000000	1	WHITE KITCHEN KAT FLOUT FLOUT	Home Decor	290.0
1000000000000000000	1	WHITE KITCHEN KAT FLOUT FLOUT	Home Decor	290.0
1000000000000000000	1	WHITE KITCHEN KAT FLOUT FLOUT	Home Decor	290.0
1000000000000000000	1	WHITE KITCHEN KAT FLOUT FLOUT	Home Decor	290.0
1000000000000000000	1	WHITE KITCHEN KAT FLOUT FLOUT	Home Decor	290.0

-- Scalar Subquery

```
SELECT
```

```
`InvoiceNo`,
```

```
`Description`,
```

```
`Quantity`
```

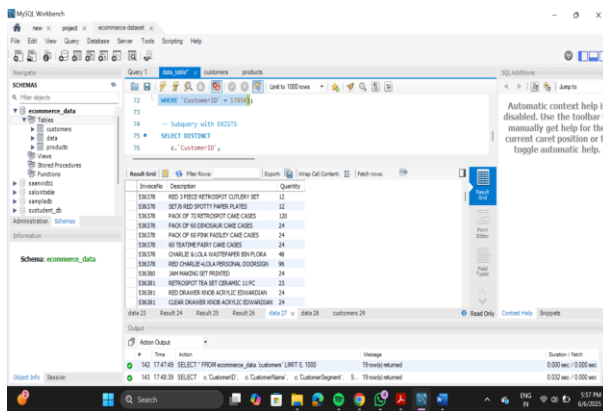
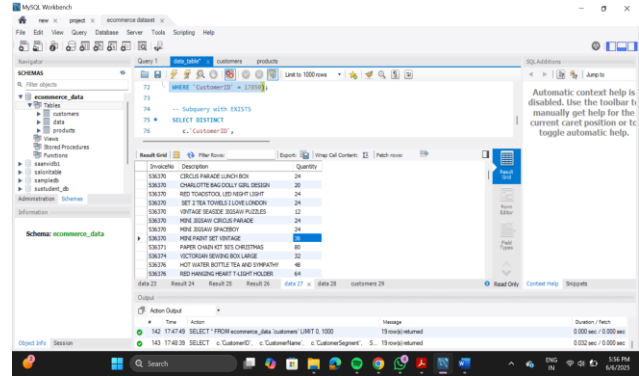
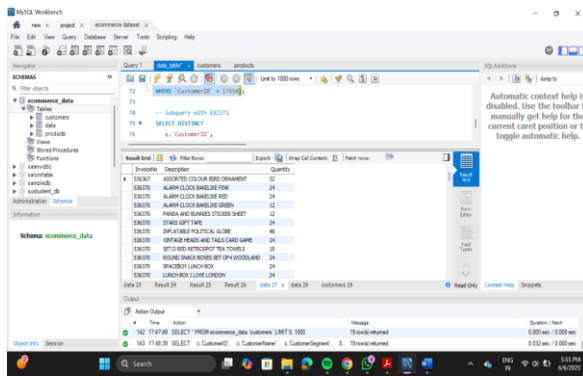
FROM

```
ecommerce_data.`data`
```

WHERE

```
`Quantity` > (SELECT AVG(`Quantity`) FROM ecommerce_data.`data`
```

```
WHERE `Quantity` > 0);
```



-- Row Subquery

SELECT

`InvoiceNo`

, `InvoiceDate`,

`Description`,

`Quantity`,

`UnitPrice`

FROM

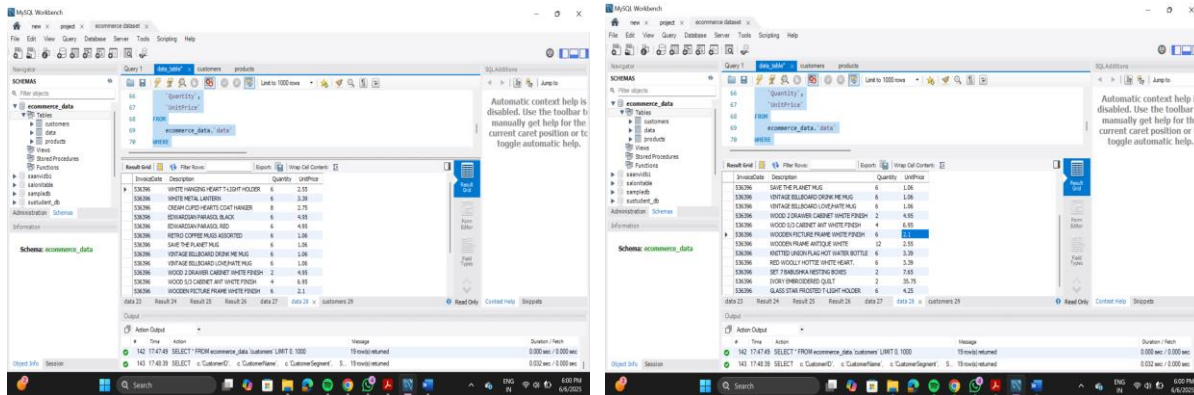
ecommerce\_data.`data`

WHERE

(`CustomerID`, `InvoiceDate`) = (SELECT `CustomerID`, MIN(`InvoiceDate`) FROM ecommerce\_data.`data`

WHERE `CustomerID` = 17850);





-- Subquery with EXISTS

SELECT DISTINCT

c.`CustomerID`,

c.`CustomerName`

FROM

ecommerce\_data.`customers` AS c

WHERE

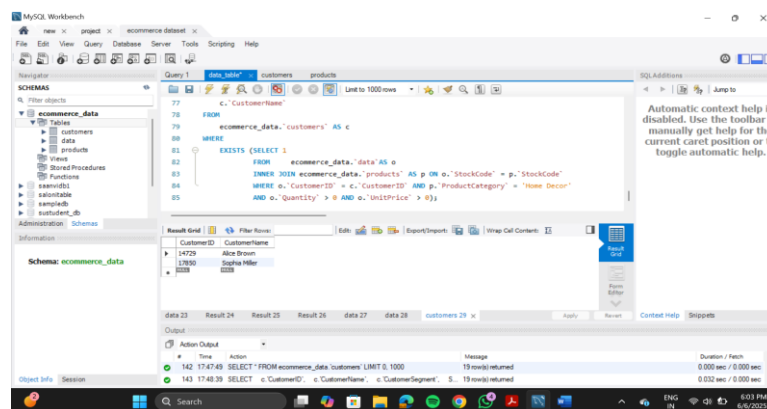
EXISTS (SELECT 1

FROM ecommerce\_data.`data` AS o

INNER JOIN ecommerce\_data.`products` AS p ON o.`StockCode` = p.`StockCode`

WHERE o.`CustomerID` = c.`CustomerID` AND p.`ProductCategory` = 'Home Decor'

AND o.`Quantity` > 0 AND o.`UnitPrice` > 0);





VIEWS:

1. USE ecommerce\_data;

```
CREATE VIEW `vw_daily_sales_summary` AS
SELECT
    DATE(`InvoiceDate`) AS SaleDate,
    SUM(`Quantity` * `UnitPrice`) AS DailyRevenue,
    COUNT(DISTINCT `InvoiceNo`) AS NumberOfOrders
FROM
    ecommerce_data.`data`
WHERE
    `Quantity` > 0 AND `UnitPrice` > 0
GROUP BY
    DATE(`InvoiceDate`)
ORDER BY
    SaleDate ASC;
```

2. USE ecommerce\_data;

```
CREATE VIEW `vw_customer_spending_summary` AS
SELECT
    c.`CustomerID`,
    c.`CustomerName`,
    c.`CustomerSegment`,
```

```

SUM(CASE WHEN o.`Quantity` > 0 AND o.`UnitPrice` > 0 THEN o.`Quantity` * o.`UnitPrice`
ELSE 0 END) AS TotalSpending,

COUNT(DISTINCT o.`InvoiceNo`) AS NumberOfOrders,

COUNT(DISTINCT o.`StockCode`) AS NumberOfUniqueProducts

FROM

ecommerce_data.`customers` AS c

LEFT JOIN

ecommerce_data.`data` AS o ON c.`CustomerID` = o.`CustomerID`

GROUP BY

c.`CustomerID`,

c.`CustomerName`,

c.`CustomerSegment`;

```

3. USE ecommerce\_data;

```

CREATE VIEW `vw_product_category_performance` AS

SELECT

p.`ProductCategory`,

SUM(o.`Quantity` * o.`UnitPrice`) AS TotalCategoryRevenue,

COUNT(DISTINCT o.`StockCode`) AS NumberOfUniqueProductsSold

FROM ecommerce_data.`data` AS o

INNER JOIN

ecommerce_data.`products` AS p ON o.`StockCode` = p.`StockCode`

WHERE

o.`Quantity` > 0 AND o.`UnitPrice` > 0

GROUP BY

```

p.`ProductCategory`

ORDER BY

TotalCategoryRevenue DESC;

4. USE ecommerce\_data;

CREATE VIEW `vw\_high\_value\_items` AS

SELECT

`StockCode`,

`Description`,

AVG(`UnitPrice`) AS `AverageUnitPrice`,

COUNT(\*) AS TotalTransactions

FROM

ecommerce\_data.`data`

WHERE

`UnitPrice` > 100

AND `Quantity` > 0

GROUP BY

`StockCode`,

`Description`

HAVING

COUNT(\*) > 5

ORDER BY

AverageUnitPrice DESC;