

# CASE STUDY

## ONLINE RETAIL DATABASE

## Objective:

The objective of this project is to write optimized SQL queries to perform data extraction, transformation, and analysis on transactional sales data, and generate meaningful business insights, identify trends in sales and customer behaviour, and document the entire workflow, including data cleaning, calculation of key metrics, and query optimization, to support informed business decisions.

## Dataset:

Online Retail Dataset

## Data Structure:

- InvoiceNo - Invoice number for each transaction
- StockCode - Code assigned to each product item
- Description - Description of the product
- Quantity - Number of products purchased
- InvoiceDate - Date and time when the invoice was issued
- UnitPrice - Price per unit
- CustomerID - ID for each customer
- Country - Country of the customer

## Tools Required:

- SQL Database: SQL Server
- Documentation: Google Docs / Word

## Execution:

### Importing data:

- Open SSMS and connect to your SQL Server
- Right click on the Database folder and select ‘New Database’
- Name the database ‘Retail\_fp’ and select Close

## Phase 1: Data Acquisition & Cleaning

- Handle missing values
- Finding the number of rows with no customerID: Removing the customerID will eliminate missing or incomplete information, making the dataset more accurate for providing insights.

```
SELECT COUNT(*) - COUNT([Customer_ID]) AS null_ID  
FROM [dbo].[online_retail];
```

The screenshot shows the SQL Server Management Studio interface. The Object Explorer on the left shows the database structure, including the 'Retail\_fp' database and its tables. The central pane displays a query window titled 'SQLQuery2.s...0ACER (66)\*'. The query is:

```
--Counting number of rows with no customer_ID  
SELECT COUNT(*) - COUNT([Customer_ID]) AS null_CustomerID  
FROM [dbo].[online_retail];
```

The results pane shows a single row with the value 236682. A status bar at the bottom indicates 'Query executed successfully.'

- Cleaning the table by removing the rows with no customerID

```
Delete from [dbo].[online_retail]  
where [Customer_ID] IS NULL;
```

The screenshot shows the SQL Server Management Studio interface. The Object Explorer on the left shows the database structure, including the 'Retail\_fp' database and its tables. The central pane displays a query window titled 'SQLQuery2.s...0ACER (66)\*'. The query is:

```
--Deleting the rows with NULL customer_ID  
Delete from [dbo].[online_retail]  
where [Customer_ID] IS NULL;
```

The results pane shows a message '(236682 rows affected)'. A status bar at the bottom indicates 'Query executed successfully.'

### 3. Finding rows with negative Price or Quantity value

```
select * from [dbo].[online_retail]
where [Quantity]<=0 or [Price]<=0;
```

The screenshot shows the SQL Server Management Studio interface. The Object Explorer on the left shows the database structure, including the 'Retail\_fp' database and its tables. The central pane displays the results of the following query:

```
--Finding negative Quantity or Price values
select * from [dbo].[online_retail]
where [Quantity]<=0 or [Price]<=0;
```

The results grid shows 18 rows of data from the 'online\_retail' table, filtered by negative values for either 'Quantity' or 'Price'. The columns include Invoice, StockCode, Description, Quantity, InvoiceDate, Price, Customer\_ID, Country, and Total\_Sale\_Amount. The data includes various items like 'PAPER BUNTING WHITE LACE', 'CREAM FELT EASTER EGG BASKET', and 'POTTING SHED SOW N' GROW SET'.

### 4. Deleting the rows with negative values

```
Delete from [dbo].[online_retail]
where [Quantity]<=0 or [Price]<=0;
```

The screenshot shows the SQL Server Management Studio interface. The Object Explorer on the left shows the database structure, including the 'Retail\_fp' database and its tables. The central pane displays the results of the following delete query:

```
--Deleting negative Quantity or Price values
Delete from [dbo].[online_retail]
where [Quantity]<=0 or [Price]<=0;
```

The results grid shows the message '(18230 rows affected)' indicating the number of rows deleted. The status bar at the bottom right shows '0 rows'.

- Remove duplicates

1. Finding duplicate data: To find duplicate data row\_number() window function is used which displays rows that occur more than once.

```
select *,count(*) as Row_count
from [dbo].[online_retail]
group by
[Invoice],[StockCode],[Description],[Quantity],[InvoiceDate],[Price],[Customer_ID],[Country]
having count(*)>1;
```

The screenshot shows the SQL Server Management Studio interface. The Object Explorer on the left shows the database structure, including the 'Retail\_fp' database and its tables. The 'SQLQuery2.x - [ACER (66)]' window contains the following T-SQL code:

```
--Finding duplicate data
select *,count(*) as Row_count
from [dbo].[online_retail]
group by
[Invoice],[StockCode],[Description],[Quantity],[InvoiceDate],[Price],[Customer_ID],[Country]
having count(*)>1
```

The results grid shows 24,887 rows of data from the 'online\_retail' table, grouped by various columns. The 'Row\_count' column indicates the number of occurrences for each group. The results show several groups with a count greater than 1, indicating duplicates.

## 2. Deleting duplicate data

```
WITH CTE_TEMP as (select *,row_number() over (
Partition by [Invoice], [StockCode], [Description], [Quantity],
[InvoiceDate], [Price], [Customer_ID], [Country] Order by [Invoice])as Row_num
from [dbo].[online_retail])
delete from CTE_TEMP
where Row_num>1;
```

The screenshot shows the SQL Server Management Studio interface. The Object Explorer on the left shows the database structure, including the 'Retail\_fp' database and its tables. The 'SQLQuery2.x - [ACER (66)]' window contains the following T-SQL code:

```
--Deleting duplicate data
WITH CTE_TEMP as (select *,row_number() over (
Partition by [Invoice], [StockCode], [Description], [Quantity],
[InvoiceDate], [Price], [Customer_ID], [Country] Order by [Invoice])as Row_num
from [dbo].[online_retail])
delete from CTE_TEMP
where Row_num>1;
```

The results grid shows the execution message: '(40234 rows affected)'. The completion time is listed as '2018-11-11T14:11:58.9999041+08:00'. The status bar at the bottom indicates '0 rows'.

- Ensuring proper datatypes:

- Price datatype from varchar(50) to float

```
ALTER TABLE [dbo].[online_retail]
ALTER COLUMN [Price] float;
```

The screenshot shows the SSMS interface with a query window titled "SQLQuery2.s...0ACER (56)\*". The query is:

```
1 --Ensuring proper datatypes
2 --Price datatype from varchar to float
3 ALTER TABLE [dbo].[online_retail]
4 ALTER COLUMN [Price] float;
```

The "Messages" pane at the bottom displays the results:

```
Commands completed successfully.
Completion time: 2025-11-14T16:30:38.3197615-05:00
```

- Quantity datatype from nvarchar(50) to int

```
ALTER TABLE [dbo].[online_retail]
ALTER COLUMN [Quantity] int;
```

The screenshot shows the SSMS interface with a query window titled "SQLQuery2.s...0ACER (54)\*". The query is:

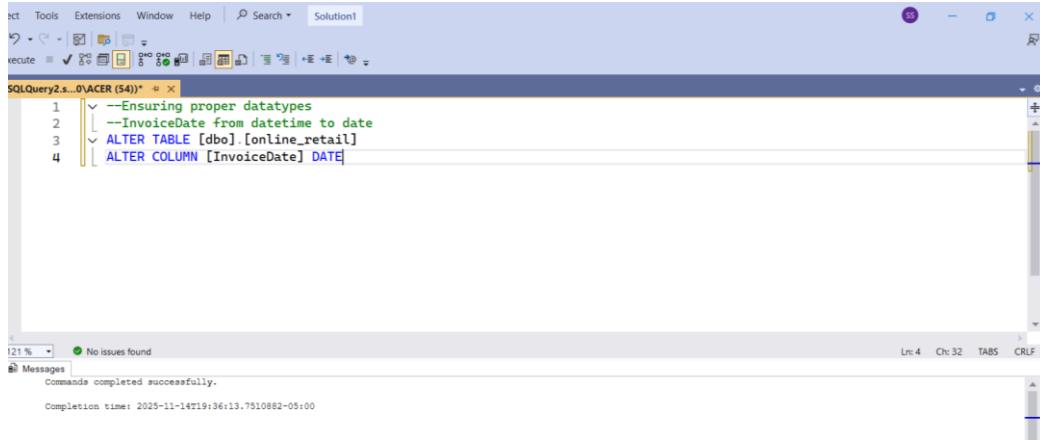
```
1 --Ensuring proper datatypes
2 --Quantity datatype from nvarchar(50) to int
3 ALTER TABLE [dbo].[online_retail]
4 ALTER COLUMN [Quantity] int;
```

The "Messages" pane at the bottom displays the results:

```
Commands completed successfully.
Completion time: 2025-11-14T19:36:13.7510882-05:00
```

### 3. InvoiceDate from datetime to date

```
ALTER TABLE [dbo].[online_retail]
ALTER COLUMN [InvoiceDate] DATE;
```



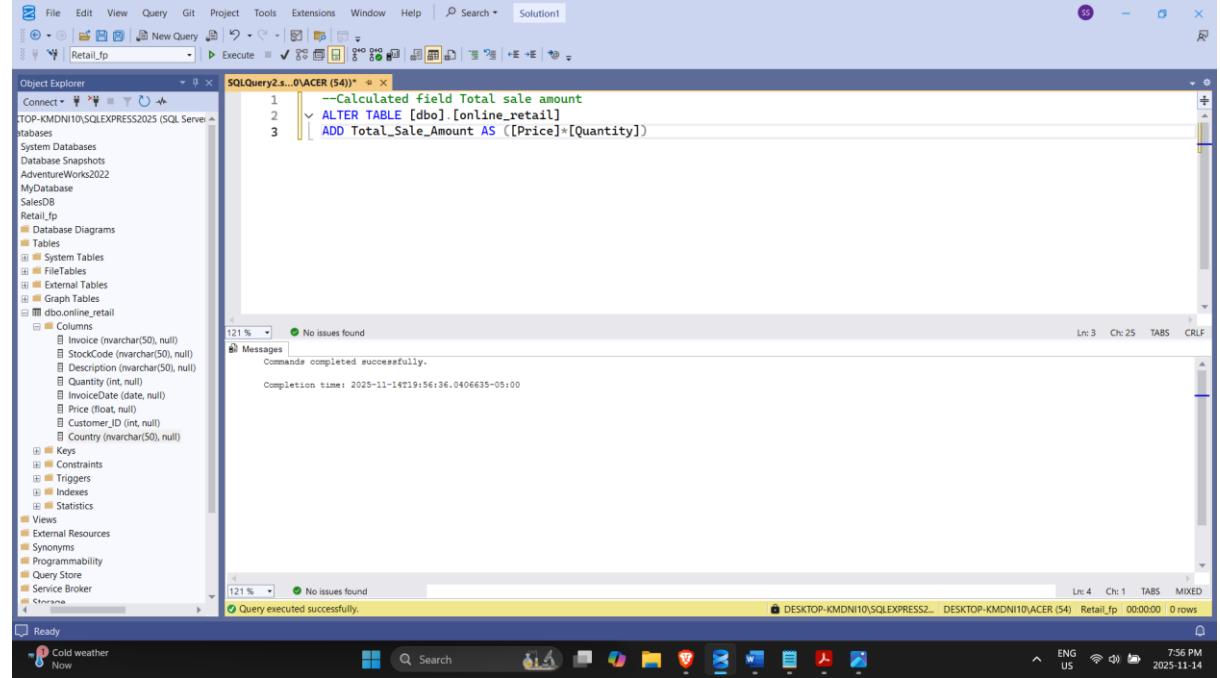
The screenshot shows the SQL Server Management Studio interface. In the Object Explorer on the left, a database named 'Retail\_fp' is selected. In the center, the 'SQLQuery2... (54)' window displays the following T-SQL code:

```
1 --Ensuring proper datatypes
2   --InvoiceDate from datetime to date
3   ALTER TABLE [dbo].[online_retail]
4     ALTER COLUMN [InvoiceDate] DATE
```

The code is highlighted with yellow boxes around the comments and the ALTER TABLE statement. Below the code, the 'Messages' pane shows the output: "Commands completed successfully." and "Completion time: 2025-11-14T19:36:13.7510882-05:00".

- Create calculated field Total sales Amount

```
ALTER TABLE [dbo].[online_retail]
ADD Total_Sales_Amount AS ([Price]*[Quantity])
```



The screenshot shows the SQL Server Management Studio interface. In the Object Explorer on the left, a database named 'Retail\_fp' is selected. In the center, the 'SQLQuery2... (54)' window displays the following T-SQL code:

```
1 --Calculated field Total sale amount
2   ALTER TABLE [dbo].[online_retail]
3     ADD Total_Sales_Amount AS ([Price]*[Quantity])
```

The code is highlighted with yellow boxes around the comments and the ALTER TABLE statement. Below the code, the 'Messages' pane shows the output: "Commands completed successfully." and "Completion time: 2025-11-14T19:56:36.0406635-05:00".

At the bottom of the screen, the status bar indicates "Query executed successfully.", the computer name "DESKTOP-KMDN110", the user name "DESKTOP-KMDN110\ACER (S4)", the session ID "Retail\_fp", the time "00:00:00", and the number of rows affected "0 rows".

## Phase 2: SQL Query Development

- Key metrics for sales analysis

### 1. Regional Sales

```
Select [Country],round(sum([Total_Sale_Amount]),2) as Total_sales  
from [dbo].[online_retail]  
group by [Country]  
order by Total_sales desc;
```

The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. The top menu bar includes File, Edit, View, Query, Git, Project, Tools, Extensions, Window, Help, and a search bar. Below the menu is a toolbar with various icons. The left pane is the Object Explorer, showing a tree structure of databases, tables, and other objects under 'DESKTOP-KMDN110\SQLEXPRESS2025'. The central pane contains a query window titled 'SQLQuery1.s...0ACER (59)\*' with the following T-SQL code:

```
--Total sales by Country/Region  
Select [Country],round(sum([Total_Sale_Amount]),2) as Total_sales  
from [dbo].[online_retail]  
group by [Country]  
order by Total_sales desc;
```

Below the code is a results grid titled 'Results' showing the output of the query. The columns are 'Country' and 'Total\_sales'. The data is as follows:

Country	Total_sales
United Kingdom	14009957.79
EIRE	610268.57
Netherlands	542310.07
Germany	417798.41
France	343984.68
Australia	169233.46
Spain	100162.28
Switzerland	93000.44
Nigeria	91271.83
Denmark	69411.79
Belgium	64280.41
Portugal	54183.52
Norway	53536.8
Channel Islands	44424.93
Japan	43023.91
Italy	31907.72
Finland	29925.54
Singapore	25317.06

At the bottom of the results grid, a message says 'Query executed successfully.' The status bar at the bottom right shows the computer name 'DESKTOP-KMDN110\SQLEXPRESS2...', the session ID 'DESKTOP-KMDN110\ACER (59)', the database 'Retail\_fp', the time '00:00:00', and the number of rows '41 rows'. The system tray at the bottom shows the date and time '10/27/2025 11:15 PM'.

### Analysis:

The Country/Region sales data shows a high concentration of revenue in certain countries, with the United Kingdom contributing 14,009,957.79 which is the largest value. Following UK are EIRE, Netherlands, Germany, France that also impact revenue significantly. This suggests that Western Europe is the retailer's primary market.

Moderate sales were observed in countries such as Australia, Switzerland, Spain, and Sweden, while regions like Nigeria, Saudi Arabia, and West Indies had minimal sales, suggesting limited market reach or presence.

For moderate sales countries, targeted ads, localized campaigns, or partnerships with local distributors could help increase market penetration. Low performing countries present long term growth opportunities which can be done by exploring regional demand, strategic pricing.

## 2. Time based trends: Monthly Sales Trends

```
--Monthly sales trends
SELECT YEAR([InvoiceDate]) AS SALES_YEAR,
       MONTH([InvoiceDate]) AS SALES_MONTH ,
       round(SUM([Total_Sale_Amount]),2) as MONTHLY_SALES
FROM [dbo].[online_retail]
GROUP BY YEAR([InvoiceDate]), MONTH([InvoiceDate])
ORDER BY YEAR([InvoiceDate]) DESC, MONTH([InvoiceDate]) ASC;
```

The screenshot shows the SQL Server Management Studio (SSMS) interface. The top menu bar includes File, Edit, View, Query, Git, Project, Tools, Extensions, Window, Help, and Search. Below the menu is a toolbar with various icons. The Object Explorer on the left shows the database structure for 'DESKTOP-KMDN10\SQLEXPRESS2022'. The main window contains a query editor titled 'SQLQuery1... - ACER (59)\*' with the previously provided T-SQL code. To the right of the code is a results grid showing the output of the query. The results grid has three columns: 'SALES\_YEAR', 'SALES\_MONTH', and 'MONTHLY\_SALES'. The data consists of 21 rows, representing monthly sales for the year 2011. The bottom status bar indicates the query was executed successfully at 10:30 PM on 2025-11-15.

SALES_YEAR	SALES_MONTH	MONTHLY_SALES
1	1	568101.31
2	2	446034.92
3	3	594081.76
4	4	488374.33
5	5	677355.15
6	6	660046.05
7	7	598952.9
8	8	644051.04
9	9	950690.2
10	10	1035642.45
11	11	1156205.41
12	12	99305.12
13	10	55802.67
14	2	50455.96
15	3	696972.47
16	4	591982
17	5	597833.38
18	6	636711.13
19	7	580775.17
20	8	602244
21	9	629013.95

### Analysis:

The monthly sales analysis for 2011 reveals clear seasonal trends in customer purchasing behavior. Sales began at 568,101.31 in January, with peaks in May (677,355.15), June (660,046.05), September (950,690.20), October (1,035,642.45), and November (1,156,205.61), and the lowest sales was recorded in December. A similar trend is noticed in the months of the previous year with sales hikes in September, October, November indicating peak demand. To increase sales, the business could implement targeted marketing campaigns during the low-performing months, and introduce seasonal promotions.

### 3. Time based trends: Quarterly Sales Trends

```
SELECT YEAR([InvoiceDate]) AS SALES_YEAR,
       DATEPART(QUARTER,[InvoiceDate]) AS SALES_QUARTER,
       round(SUM([Total_Sale_Amount]),2) AS QUARTER_SALES
  FROM [dbo].[online_retail]
 GROUP BY YEAR([InvoiceDate]), DATEPART(QUARTER,[InvoiceDate])
 ORDER BY YEAR([InvoiceDate]) DESC,DATEPART(QUARTER,[InvoiceDate]) ASC;
```

The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. The Object Explorer on the left shows a database named 'Retail\_fp' with various objects like tables, stored procedures, and triggers. The central pane displays a query window titled 'SQLQuery1.s...0ACER (59)\*'. The query itself is:

```
--Time based trends
--Quarterly sales trends
SELECT YEAR([InvoiceDate]) AS SALES_YEAR,
       DATEPART(QUARTER,[InvoiceDate]) AS SALES_QUARTER,
       round(SUM([Total_Sale_Amount]),2) AS QUARTER_SALES
  FROM [dbo].[online_retail]
 GROUP BY YEAR([InvoiceDate]), DATEPART(QUARTER,[InvoiceDate])
 ORDER BY YEAR([InvoiceDate]) DESC,DATEPART(QUARTER,[InvoiceDate]) ASC;
```

The results pane below shows the output of the query:

SALES_YEAR	SALES_QUARTER	QUARTER_SALES
2011	1	1608267.99
2011	2	1803775.53
2011	3	2135715.14
2011	4	2291153.18
2010	1	1757340.1
2010	2	1626186.61
2010	3	2020974.72
2010	4	276994.76
2009	4	683504.01

A status bar at the bottom indicates 'Query executed successfully.'

#### Analysis:

The result shows a clear growth in revenue from the start of the year in 2011, growing steadily and reaching Q4 with 2,291,153.18 in revenue, indicating demand. A similar trend in the previous year 2010, with revenue peaking in Q4 at 2,769,994.76, but is slightly higher than 2011 Q4. These trends suggest that the business can maximize revenue by focusing more on seasonal peaks, aligning marketing campaigns. The lower-performing quarters can be an opportunity for targeted promotions, and product launches to help keep sales up.

#### 4. Product Performance: Identify top-selling products based on revenue

```
select top 20 [Description] as Products,
       round(sum([Total_Sale_Amount]),2) as Revenue,
       rank() over (order by round(sum([Total_Sale_Amount]),2) desc) as Product_Rank
  from [dbo].[online_retail]
 group by [Description]
 order by Revenue desc;
```

--Top 20 selling products based on Revenue

Products	Revenue	Product_Rank
REGENCY CAKESTAND 3 TIER	273994.42	1
WHITE HANGING HEART T-LIGHT HOLDER	245480.77	2
Manuals	151350.21	3
JUMBO BAG RED RETROSPOT	133467	4
ASSORTED COLOUR BIRD ORNAMENT	123330.16	5
POSTAGE	122320	6
PARTY BUNTING	103030.93	7
MEDIUM CERAMIC TOP STORAGE JAR	91243.22	8
PAPER CHAIN KIT 50'S CHRISTMAS	79913.76	9
CHILLI LIGHTS	67555.98	10
JUMBO BAG STRAWBERRY	64048.73	11
BLACK RECORD COVER FRAME	58334.07	12
ROTATING SILVER ANGELS T-LIGHT HOLDER	54482.24	13
VINTAGE UNION JACK BUNTING	54288.49	14
EDITION 1000 PIECE JIGSAW PUZZLE	53900.00	15
WOOD BLACK BOARD ANT WHITE FINISH	52748.16	16
JUMBO BAG BAROQUE BLACK WHITE	51914.44	17
HEART OF WICKER LARICE	49894.52	18
RED HANGING HEART T-LIGHT HOLDER	48793.32	19

#### Analysis:

The product analysis shows that the REGENCY CAKESTAND 3 TIER and the WHITE HANGING HEART T-LIGHT HOLDER are the top ranked in terms of revenue, followed by Manuals, JUMBO BAG RED RETROSPOT, and ASSORTED COLOUR BIRD ORNAMENT. These results indicate a high demand for home decor and seasonal home accessories. To increase sales, the business can focus on keeping these popular products in stock, expanding similar product ranges and strengthening marketing around these products.

## 5. Customer Behavior Analysis: Segment customers by purchase frequency

```
Select [Customer_ID], count(distinct[Invoice]) as Number_of_Purchases,
CASE
    when count(distinct[Invoice])<=10 then 'Low Frequency'
    when count(distinct[Invoice])<=50 then 'Medium frequency'
    when count(distinct[Invoice])>50 then 'High frequency'
END as Purchase_Frequency
from [dbo].[online_retail]
group by Customer_ID
order by count(distinct[Invoice]) desc;
```

The screenshot shows the SSMS interface with the following details:

- Object Explorer:** Shows the database structure, including the `MyDatabase` and `Retail_fp` databases.
- SQLQuery1.s\_0ACER (59)\***: The current query window contains the T-SQL code for segmenting customers by purchase frequency.
- Results:** The results grid displays 5,860 rows of data. The columns are:
 

Customer_ID	Number_of_Purchases	Purchase_Frequency
40	56	High frequency
41	55	High frequency
42	55	High frequency
43	55	High frequency
44	54	High frequency
45	54	High frequency
46	54	High frequency
47	53	High frequency
48	53	High frequency
49	52	High frequency
50	52	High frequency
51	51	High frequency
52	51	High frequency
53	51	High frequency
54	49	Medium frequency
55	47	Medium frequency
56	47	Medium frequency
57	47	Medium frequency
58	46	Medium frequency
- Status Bar:** Shows the message "Query executed successfully." and the system information: DESKTOP-KMDN110\SQLEXPRESS2\_... DESKTOP-KMDN110\ACER (59) Retail\_fp 00:00:00 5,860 rows.

### Analysis:

Customers are segmented by purchase frequency into High, Medium, and Low frequency groups. The analysis shows a large group of high frequency buyers that play a significant role in the company's sales revenue. The medium-frequency segment is also substantial, suggesting an opportunity for growth through targeted marketing or personalized offers. The data also shows a large group of low frequency buyers who make very few purchases. Overall growth can be strengthened by engaging and focusing on high frequency buyers.

Business recommendations:

- Increase Sales: Focus marketing and promotional efforts on high-value regions and top-selling products. Offer discounts or bundle slow-moving items to clear inventory.
- Customer Engagement: Implement targeted campaigns for medium- and low-frequency customers to encourage repeat purchases.
- Growth Opportunities: Expand outreach in emerging markets, leverage seasonal sales trends for timely promotions, and use insights from purchase frequency segmentation to improve personalized marketing.

Conclusion:

This project showcased how SQL can be used to transform raw transactional data into meaningful business insights. Through data cleaning, handling missing values, removing duplicates, and creating calculated fields, the dataset was prepared for accurate analysis. Key findings, including regional sales patterns, time-based trends, top-selling products, and customer purchase behaviors, provided actionable insights to boost sales, optimize inventory, and enhance customer engagement.