

# 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:

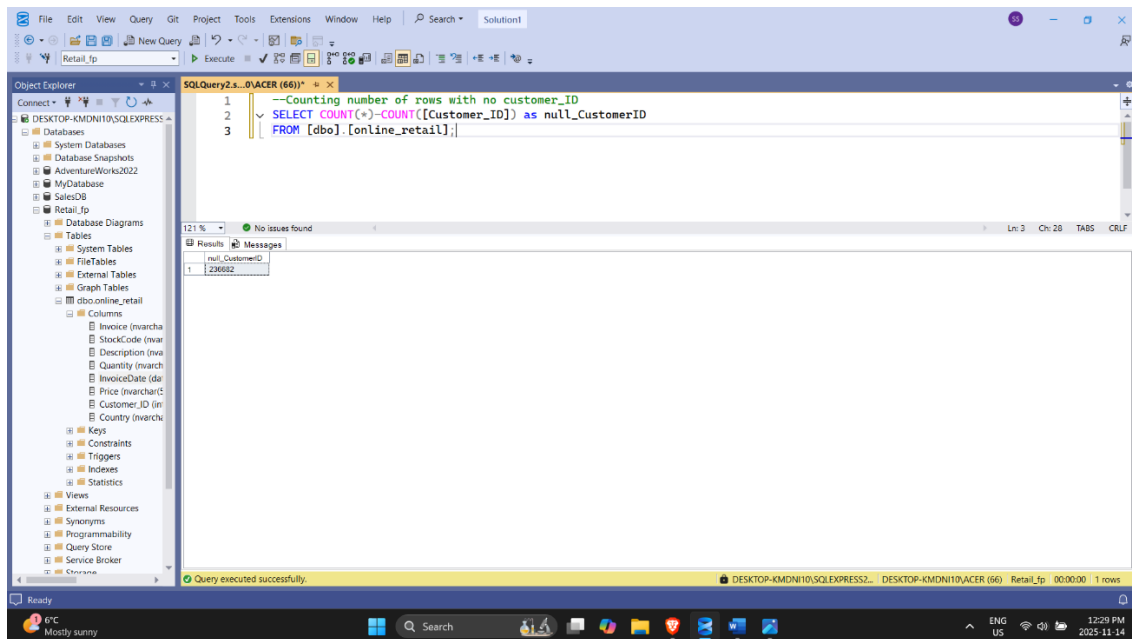
### Importing data:

- 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

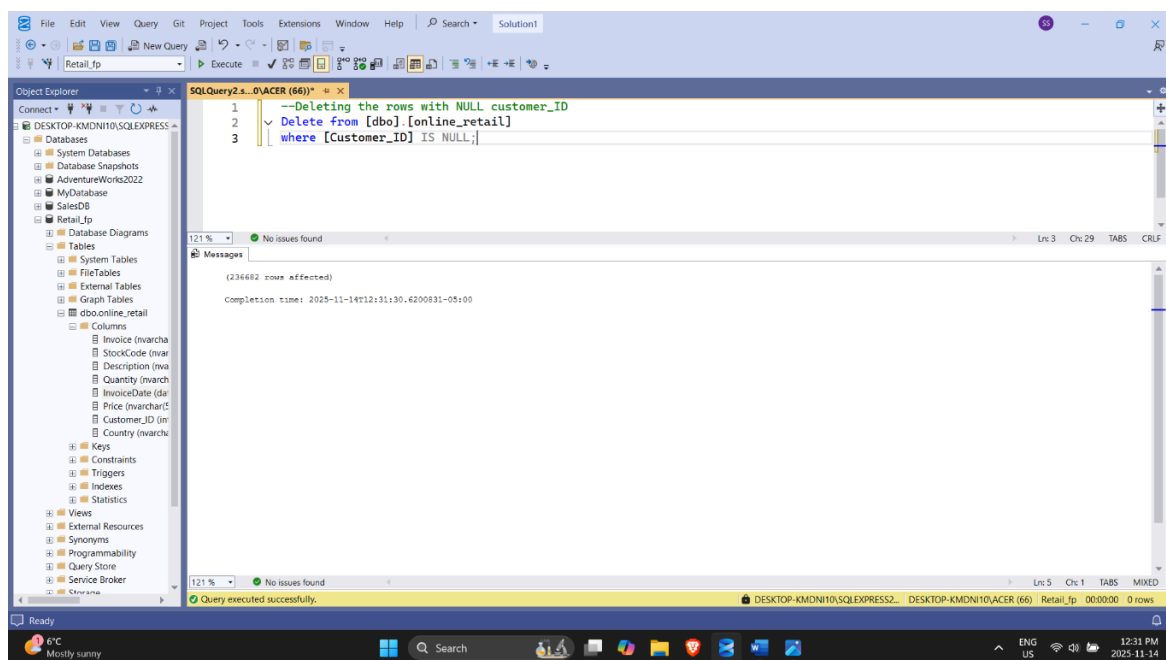
- Handle missing values
  1. Finding the number of rows with no customerID: Removing the customerID will eliminate missing or incomplete information, making the dataset more accurate for providing insights.

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



2. Cleaning the table by removing the rows with no customerID

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



### 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 Enterprise Manager interface. The left pane displays the Object Explorer with the database structure of 'DESKTOP-KMDN10\SQLEXPRESS2025 (S)'. The right pane shows a query window titled 'SQLQuery1.s...0\ACER (59)' with the following SQL code:

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

The query results are displayed in a table with the following columns: Invoice, StockCode, Description, Quantity, InvoiceDate, Price, Customer\_ID, Country, and Total\_Sale\_Amount. The results show 18 rows of data, including items like 'PAPER BUNTING WHITE LACE', 'CREAM FELT EASTER EGGS BASKET', and 'POTTING SHED SOW N' GROW SET'. The status bar at the bottom indicates 'Query executed successfully' and '18,230 rows'.

### 4. Deleting the rows with negative values

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

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the Object Explorer with the database structure of 'DESKTOP-KMDN10\SQLEXPRESS2025 (S)'. The right pane shows a query window titled 'SQLQuery1.s...0\ACER (59)' with the following SQL code:

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

The query results are displayed in a table with the following columns: Invoice, StockCode, Description, Quantity, InvoiceDate, Price, Customer\_ID, Country, and Total\_Sale\_Amount. The results show 18 rows of data, including items like 'PAPER BUNTING WHITE LACE', 'CREAM FELT EASTER EGGS BASKET', and 'POTTING SHED SOW N' GROW SET'. The status bar at the bottom indicates 'Query executed successfully' and '18,230 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;
```

Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer_ID	Country	Row_count
489517	21821	RET OF THREE VINTAGE GFT WRAPS	1	2009-12-01 11:34:00.0000000	1.98	18329	United Kingdom	2
489517	21821	GLITTER STAR GARLAND WITH BELLS	1	2009-12-01 11:34:00.0000000	3.75	18329	United Kingdom	2
489517	21912	VINTAGE ENIGMAS & LOCKERS	1	2009-12-01 11:34:00.0000000	3.75	18329	United Kingdom	3
489517	21913	VINTAGE SEASIDE JIGSAW PUZZLES	1	2009-12-01 11:34:00.0000000	3.75	18329	United Kingdom	2
489517	22130	PARTY CONE CHRISTMAS DECORATION	6	2009-12-01 11:34:00.0000000	0.85	18329	United Kingdom	2
489517	22319	HAIKILIPS FORTIES FABRIC ASSORTED	12	2009-12-01 11:34:00.0000000	0.65	18329	United Kingdom	2
489517	54914	SEA PISTACHIO COVERED COASTERS	1	2009-12-01 11:34:00.0000000	2.95	18329	United Kingdom	2
489529	22028	PENNY FARTINGS BIRTHDAY CARD	12	2009-12-01 11:51:00.0000000	0.42	17864	United Kingdom	2
489529	22036	DINGDOLPH BIRTHDAY CARD	12	2009-12-01 11:51:00.0000000	0.42	17864	United Kingdom	2
489529	48129	DOOR MAT TOWNERY	1	2009-12-01 11:51:00.0000000	6.75	17864	United Kingdom	2
489531	22073	RETRO SPOT STORAGE JAR	1	2009-12-01 12:02:00.0000000	3.75	18011	United Kingdom	2
489531	84446	ANTIQUE SILVER TEA GLASS ETCHED	6	2009-12-01 12:02:00.0000000	1.25	18011	United Kingdom	2
489536	171840	ARI COIL SMALL BANG FROG FWEIGHT	1	2009-12-01 12:13:00.0000000	0.42	16393	United Kingdom	2
489536	21034	REX CASH+CARRY JUMBO SHOPPER	1	2009-12-01 12:13:00.0000000	0.95	16393	United Kingdom	2
489536	21231	BRIGHTHEART CERAMIC PRINCE ROSE	1	2009-12-01 12:13:00.0000000	1.25	16393	United Kingdom	2
489536	21759	LOLITA DESIGN COTTON TOTE BAG	1	2009-12-01 12:13:00.0000000	2.25	16393	United Kingdom	3
489536	21950	ASSORTED TUTTI FRUTTI BRACELET	1	2009-12-01 12:13:00.0000000	0.65	16393	United Kingdom	2
489536	21786	RAIN-HAT WITH RED SPOTS	1	2009-12-01 12:13:00.0000000	0.42	16393	United Kingdom	2
489536	21791	CHRISTMAS HANGING TREE WITH BELL	1	2009-12-01 12:13:00.0000000	1.25	16393	United Kingdom	2
489536	21809	CHRISTMAS HANGING TREE WITH BELL	1	2009-12-01 12:13:00.0000000	1.25	16393	United Kingdom	2
489536	22107	PIZZA PLAY W BOX	1	2009-12-01 12:13:00.0000000	3.75	16393	United Kingdom	2
489536	22142	CHRISTMAS CRAFT WHITE FAIRY	1	2009-12-01 12:13:00.0000000	1.45	16393	United Kingdom	3
489536	22185	SLATE TILE NATURAL HANDS	2	2009-12-01 12:13:00.0000000	1.65	16393	United Kingdom	2
489536	22353	LUNCHBOX WITH CULINARY FAIRY CAKES	1	2009-12-01 12:13:00.0000000	2.95	16393	United Kingdom	2
489536	850330	SETH SILVER REINDEER T-LIGHTS	1	2009-12-01 12:13:00.0000000	1.95	16393	United Kingdom	2
489555	20977	36 PENCILS TUBE WOODLAND	2	2009-12-01 12:47:00.0000000	1.25	18719	United Kingdom	2

2. Deleting duplicate data

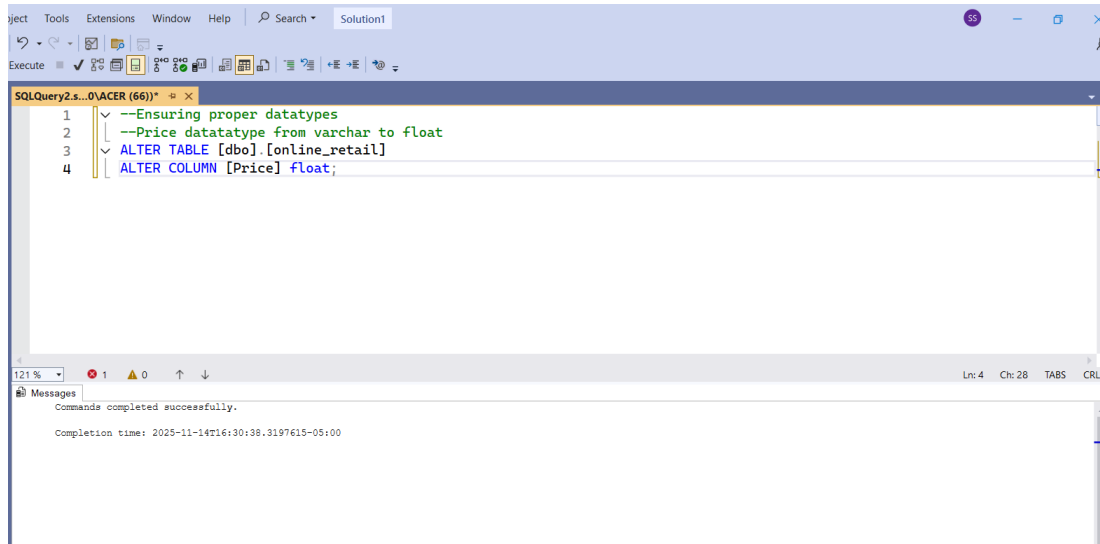
```
WITH CTE_TEMP as (select *,row_number() over (
Partition by [Invoice], [StockCode], [Description], [Quantity],
[InvoiceDate], [Price], [Customer_ID], [Country] Order by [Invoice])as
Row_num
from [dbo].[online_retail])
delete from CTE_TEMP
where Row_num>1;
```

Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer_ID	Country	Row_num
---------	-----------	-------------	----------	-------------	-------	-------------	---------	---------

- Ensuring proper datatypes:

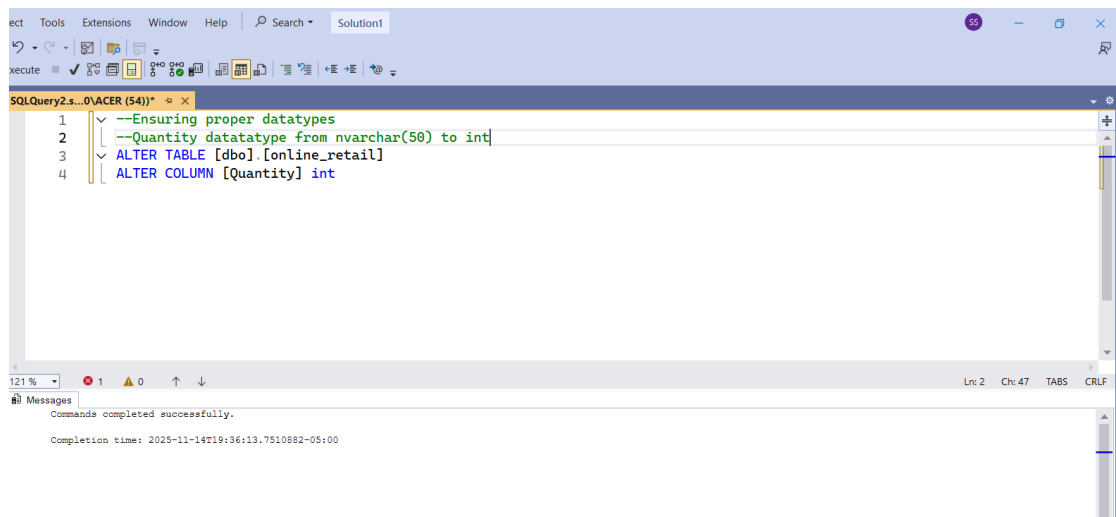
### 1. Price datatype from varchar(50) to float

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



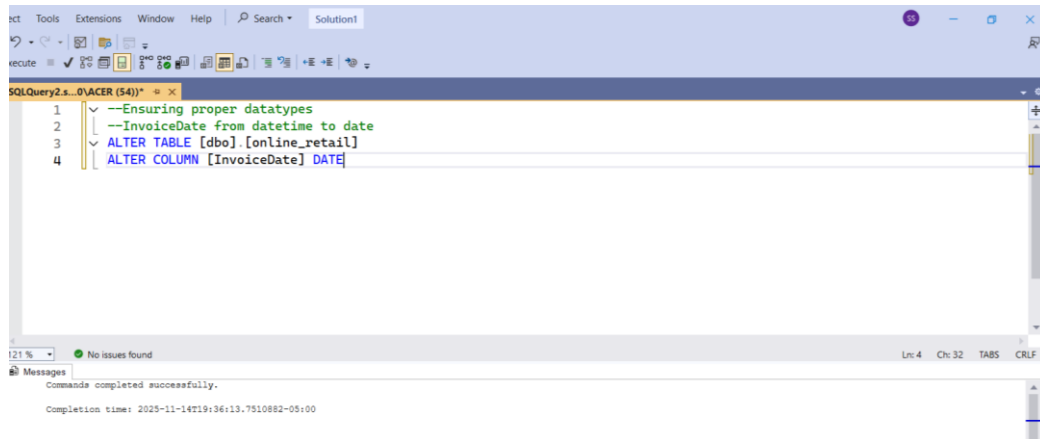
### 2. Quantity datatype from nvarchar(50) to int

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



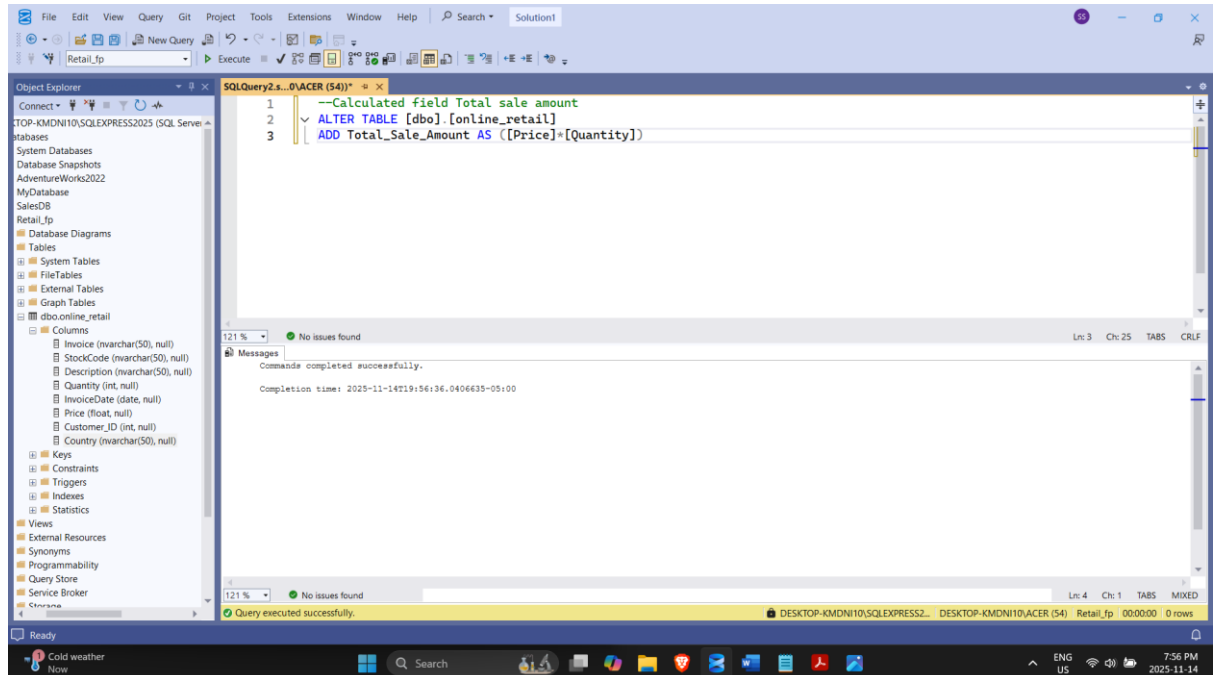
### 3. InvoiceDate from datetime to date

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



- Create calculated field Total sales Amount

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

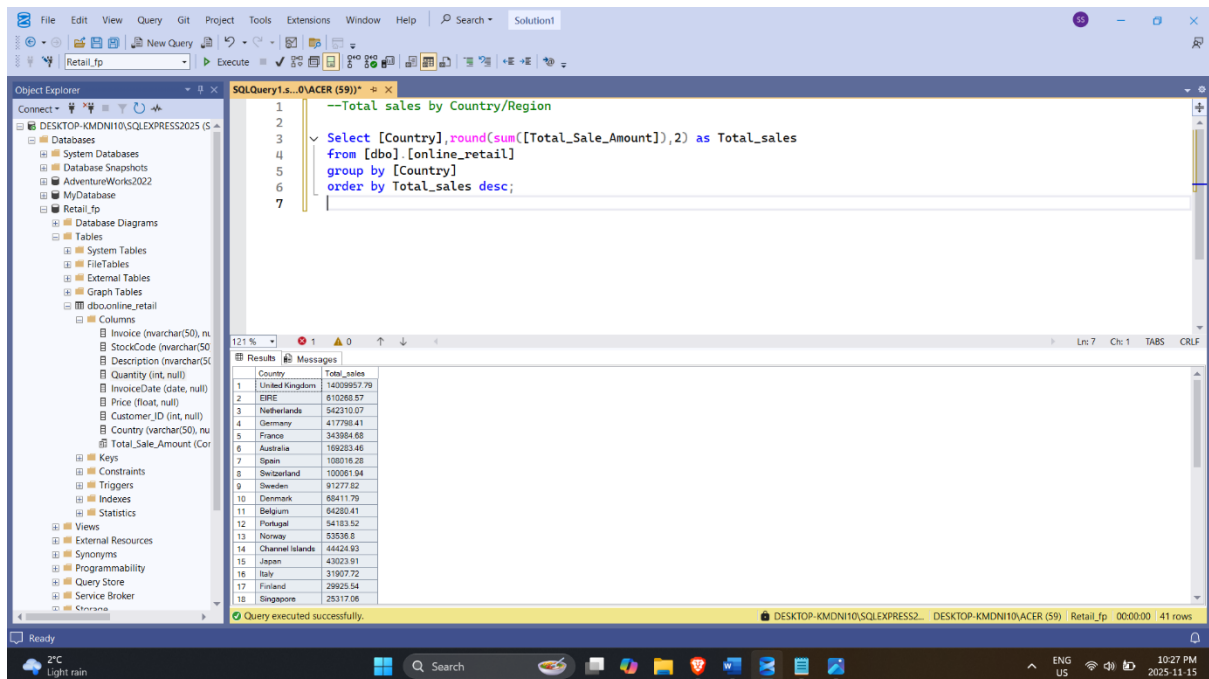


## Phase 2: SQL Query Development

- 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 displays the SQL Server Enterprise Edition interface. The left pane shows the Object Explorer with the 'Retail\_fp' database selected. The central pane shows the SQL query editor with the following query:

```
--Total sales by Country/Region
1
2
3
4 Select [Country],round(sum([Total_Sale_Amount]),2) as Total_sales
5 from [dbo].[online_retail]
6 group by [Country]
7 order by Total_sales desc;
```

The bottom pane shows the Results tab with the following data:

Country	Total_sales
United Kingdom	14009957.79
EIRE	610268.87
Netherlands	542310.07
Germany	417796.41
France	343984.68
Australia	169283.46
Spain	108016.28
Switzerland	100061.94
Sweden	91277.62
Denmark	68411.79
Belgium	64280.41
Portugal	54183.52
Norway	53536.6
Channel Islands	44624.93
Japan	43023.91
Italy	31907.72
Finland	29925.54
Singapore	25317.06

### 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.

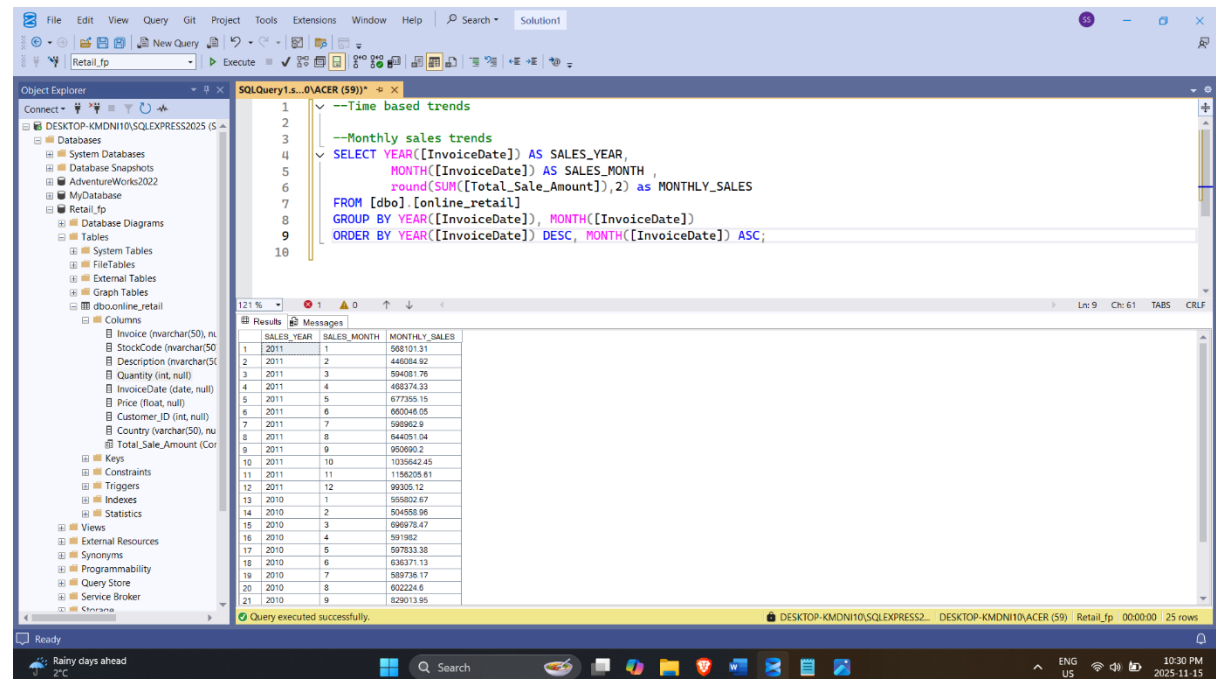
Moderate sales were observed in countries such as Australia, Switzerland, Spain, and Sweden, while regions like Nigeria, Saudi Arabia, and West Indies had minimal sales, suggesting limited market reach or presence.

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 Enterprise Manager interface. The left pane displays the 'Object Explorer' with the 'Retail.fp' database selected. The right pane shows the 'SQLQuery1.s...0\ACER (59)' window with the following query:

```
--Time based 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 'Results' pane shows the output of the query, displaying a table with 21 rows and 3 columns: SALES\_YEAR, SALES\_MONTH, and MONTHLY\_SALES. The data shows monthly sales for the years 2011 and 2010, ordered by year descending and then by month ascending.

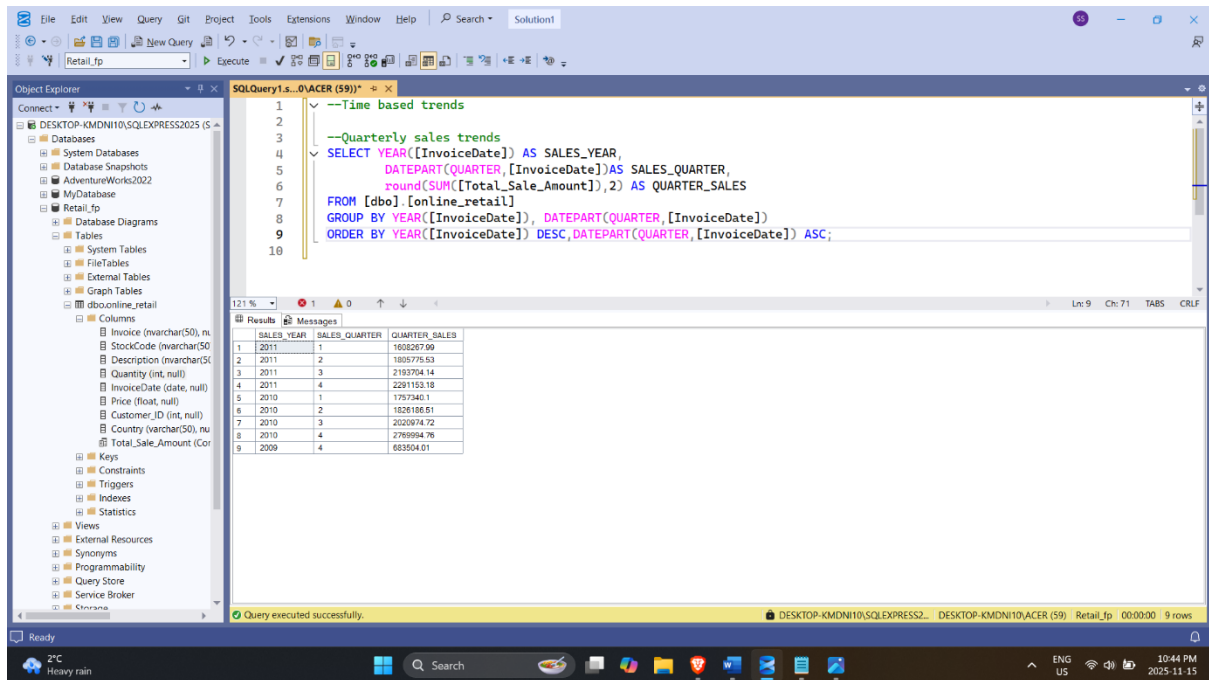
SALES_YEAR	SALES_MONTH	MONTHLY_SALES
2011	1	568101.31
2011	2	446084.62
2011	3	594081.76
2011	4	469274.23
2011	5	677355.15
2011	6	660046.05
2011	7	598962.9
2011	8	644051.04
2011	9	950690.2
2011	10	1035642.45
2011	11	1156205.61
2011	12	99305.12
2010	1	555802.67
2010	2	504558.96
2010	3	696978.47
2010	4	591982
2010	5	597933.38
2010	6	636271.13
2010	7	589738.17
2010	8	602246.6
2010	9	829013.95

### 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 displays the SQL Server Enterprise Manager interface. The left pane shows the 'Object Explorer' with the 'Retail\_fp' database selected. The right pane shows the 'SQLQuery1.sql' file with the following query:

```
--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 shows the output of the query, which is a table with 9 rows and 3 columns: SALES\_YEAR, SALES\_QUARTER, and QUARTER\_SALES. The data is as follows:

SALES_YEAR	SALES_QUARTER	QUARTER_SALES
2011	1	160267.09
2011	2	180575.53
2011	3	2193704.14
2011	4	2291153.18
2010	1	1707340.1
2010	2	1826186.61
2010	3	2020974.72
2010	4	2769994.76
2009	4	683504.01

The status bar at the bottom indicates 'Query executed successfully.' and '9 rows'.

#### 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;
```

The screenshot displays the Microsoft SQL Server Enterprise Manager interface. The left pane shows the 'Object Explorer' with the 'Retail\_Tp' database selected. The central pane shows a SQL query titled 'SQLQuery1.s...0\ACER (59)' with the following code:

```
--Top 20 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;
```

The bottom pane shows the 'Results' tab with a table of 20 rows. The table has three columns: 'Products', 'Revenue', and 'Product\_Rank'. The top three products are 'REGENCY CAKESTAND 3 TIER', 'WHITE HANGING HEART T-LIGHT HOLDER', and 'Manual'.

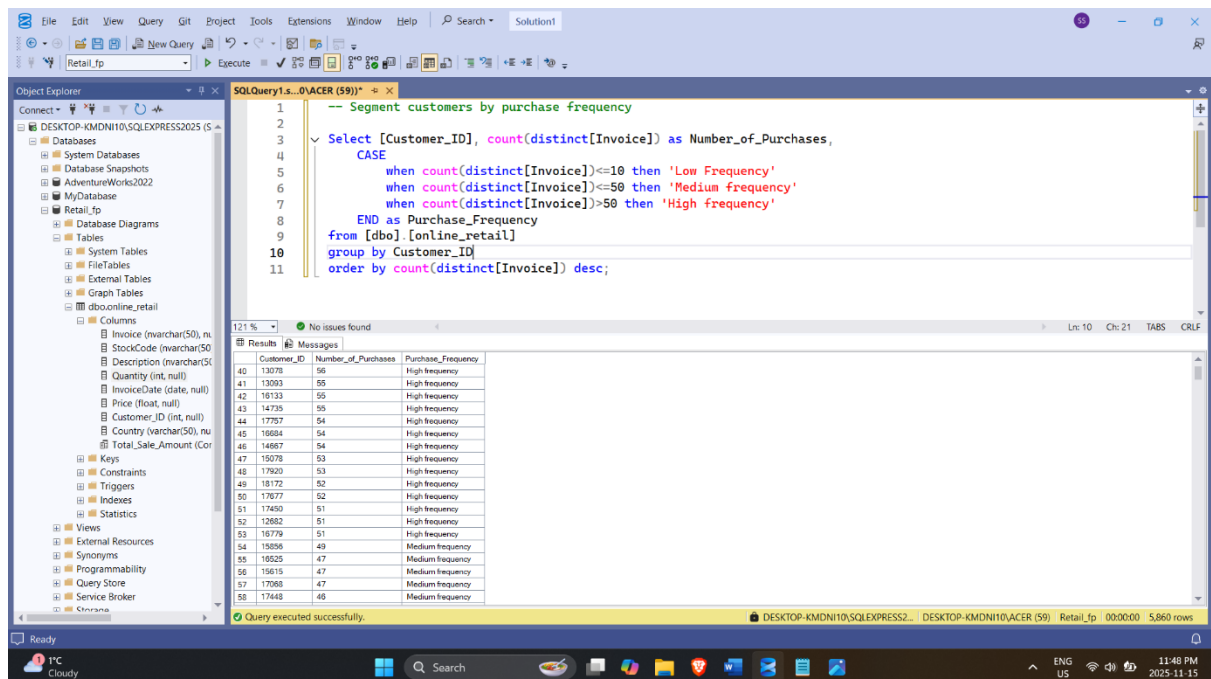
Products	Revenue	Product_Rank
1 REGENCY CAKESTAND 3 TIER	272694.42	1
2 WHITE HANGING HEART T-LIGHT HOLDER	245488.77	2
3 Manual	191380.21	3
4 JUMBO BAG RED RETROSPOT	133467	4
5 ASSORTED COLOUR BIRD ORNAMENT	123330.16	5
6 POSTAGE	122827.04	6
7 PARTY BUNTING	103030.93	7
8 MEDIUM CERAMIC TOP STORAGE JAR	81243.22	8
9 PAPER CHAIN KIT 50'S CHRISTMAS	75913.76	9
10 CHILLI LIGHTS	67855.98	10
11 JUMBO BAG STRAWBERRY	64048.73	11
12 BLACK RECORD COVER FRAME	58334.07	12
13 ROTATING SILVER ANGELS T-LIGHT HLD	54402.24	13
14 VINTAGE UNION JACK BUNTING	54288.49	14
15 EDWARDIAN PARASOL NATURAL	53945.86	15
16 WOOD BLACK BOARD ANT WHITE FINISH	52748.16	16
17 JUMBO BAG BAROQUE BLACK WHITE	51814.44	17
18 HEART OF WICKER LARGE	49884.52	18
19 WHITE HANGING HEART T-LIGHT HOLDER	48763.33	19
20		

#### 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 displays the SQL Server Enterprise Manager interface. The left pane shows the Object Explorer with the 'Retail\_fp' database selected. The central pane shows a SQL query titled 'SQLQuery1.s...0\ACER (59)' with the following code:

```
-- Segment customers by purchase frequency
1
2
3
4 Select [Customer_ID], count(distinct[Invoice]) as Number_of_Purchases,
5 CASE
6     when count(distinct[Invoice])<=10 then 'Low Frequency'
7     when count(distinct[Invoice])<=50 then 'Medium frequency'
8     when count(distinct[Invoice])>50 then 'High frequency'
9 END as Purchase_Frequency
10 from [dbo].[online_retail]
11 group by Customer_ID
12 order by count(distinct[Invoice]) desc;
```

The bottom pane shows the Results tab with a table containing 58 rows. The table has three columns: Customer\_ID, Number\_of\_Purchases, and Purchase\_Frequency. The data is sorted by Number\_of\_Purchases in descending order.

Customer_ID	Number_of_Purchases	Purchase_Frequency
40	13078	High frequency
41	13093	High frequency
42	16133	High frequency
43	14735	High frequency
44	17757	High frequency
45	16664	High frequency
46	14667	High frequency
47	15078	High frequency
48	17920	High frequency
49	18172	High frequency
50	17677	High frequency
51	17450	High frequency
52	12882	High frequency
53	16779	High frequency
54	15856	Medium frequency
55	16525	Medium frequency
56	15615	Medium frequency
57	17068	Medium frequency
58	17448	Medium frequency

The status bar at the bottom indicates 'Query executed successfully.' and shows the file path 'DESKTOP-KMDN110\SQLSERVER2025\Retail\_fp' with 00:00:00 execution time and 5,860 rows returned.

### 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.