Customer Lifetime Value Prediction

Group Name: Group Sigma

Group Members:

First name	Last Name	Student number
Amrita Kaur	Badhan	C0928455
Jaldhi	Shah	C0924359
Mohamed	Gaafar	C0905127
Mukul	Sharma	C0926138

Submission date: 12 August 2024

Template Prepared by: William Pourmajidi

Updated by Vahid Hadavi Nov 2023

Contents

Abstract

In this project we aim to predict the customer live time value (CLV) for 19,119 customers of a business from 121,317 transaction records. We acquired the business data by querying the Microsoft Adventure Works sample database using SQL, Microsoft SQL Management Server and Azure Data Studio. We performed data pre-processing to handle missing values, data formatting, data integrity, and detecting outliers. Then we processed the data by trimming the outliers, features engineering, and metrics calculation. The CLV metrics we calculated are Frequency, Recency, T, and Monetary Values. To predict the CLV we performed data preparation by splitting our data to training and test datasets and recalculating the metrics and the labels for both datasets. After preparing our data we predicted the CLV with 5 methods, which are: Historical, Probabilistic BG/NBD, Machine Learning Regression using XGBoost, Liner Deep Learning Regression, and Machine Learning Segmentation and Classification. We achieved the least error with Liner Deep Learning scoring an RMSE of 5.33 model with which is very good error margin, followed by ML XGBoost Regression with an RMSE of 294.28, Benchmark Historical CLV with an RMSE of 4406.50, and the BG-NBD Probabilistic with RMSE of 4464.26.

Introduction

When running a business, the customer is the most valuable asset. Modern businesses generate numerous amounts of data about their customers but this data isn't useful until transformed into meaningful information that helps the decision maker to act on and drive knowledge about their business. Business information is usually called business metrics of key performance indicators (KPIs). One of the most important and valuable business metrics is customer lifetime value (CLV). The CLV indicates how useful the individual customer is to the business, it takes into account many factors to calculate the value (mostly the revenue) generated to the business by the customer.

CLV is very helpful and informative for the business when running a marketing campiness, knowing the most valuable customers and working on keeping them alive for the business is a key success step. When knowing the CLV a business can utilize it to make customers person for the most valuable customers and focus on acquiring this segment of customers. On the other hand, the business can manage to classify the less valuable customers to avoid spending less money on acquiring them. Also, the CLV can be very informative when pivoting it with different business elements such as business offers, it can show the effective offers from the less effective ones.

Predicting the CLV makes it more robust for the business to manage its customer's base and enhance its processes. There are many different approaches to calculate and predict the CLV, in this project we encountered 5 methods which are: Historical, Probabilistic BG/NBD, Machine

Learning Regression using XGBoost, Liner Deep Learning Regression, and Machine Learning Segmentation and Classification. For the business to be able to utilize these methods it needs to perform data pre-processing and data preparation to make the data ready for modeling.

In the project we queried the data from a business database, pre-processed it, understand it throw EDA, processed and prepared it, then we modeled and evaluated it.

Methods

The following steps show the workflow to get the CLV prediction from the business row data.

Step 1: Data Accusation:

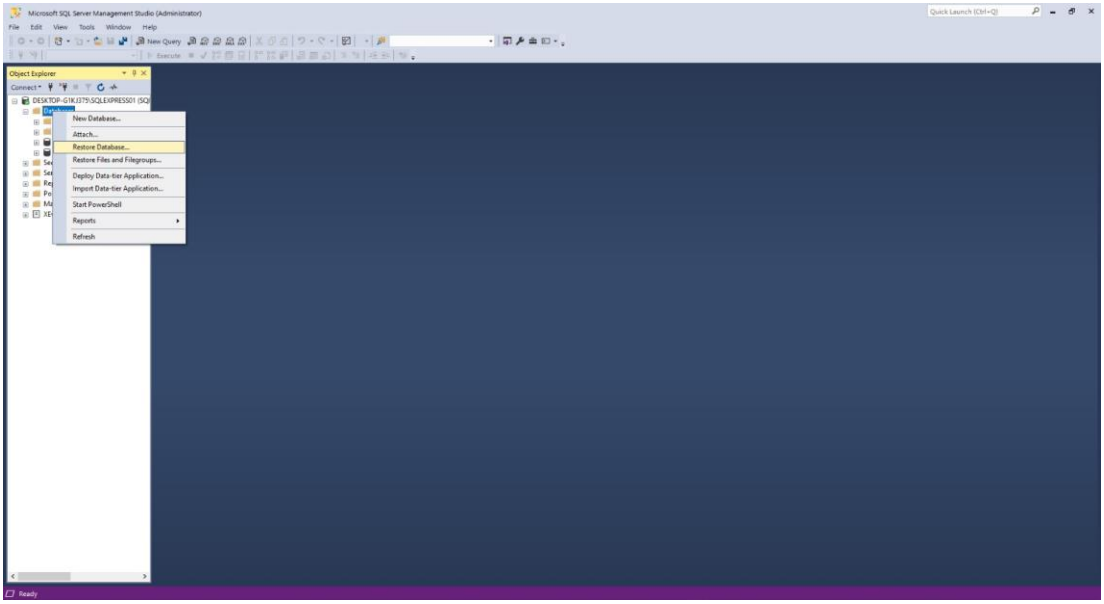
We acquired our data from Microsoft AdventureWorks sample databases. These are sample databases simulating real-world applications. The databases simulate real world scenarios such as Online Transaction Processing (OLTP), and Data Warehousing (DW).

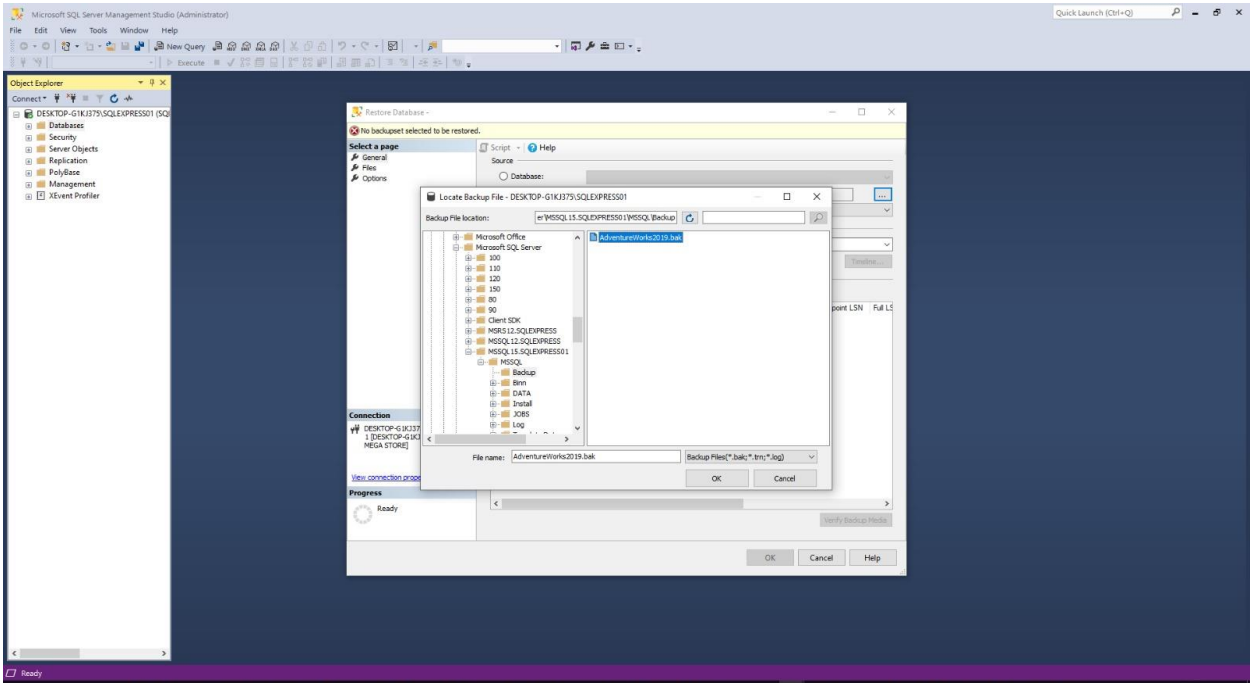
We followed the following steps to acquire and extract the data we are interested in.

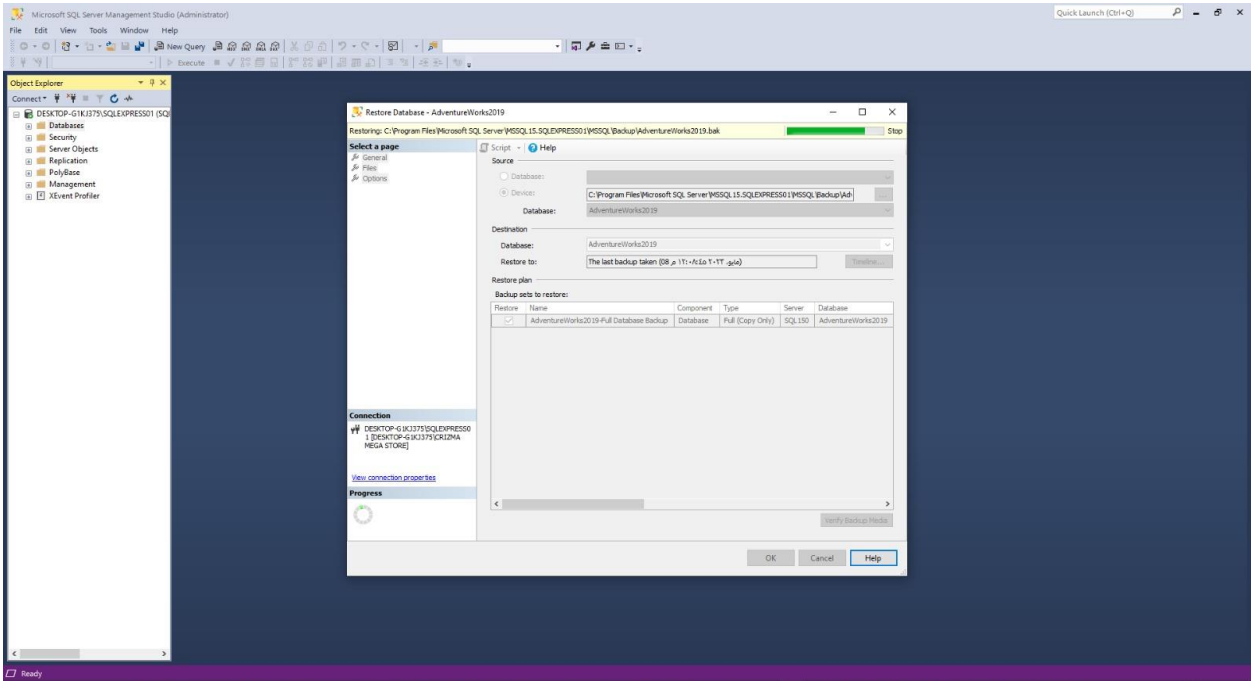
1.1. We downloaded the database backup file from the Microsoft website.

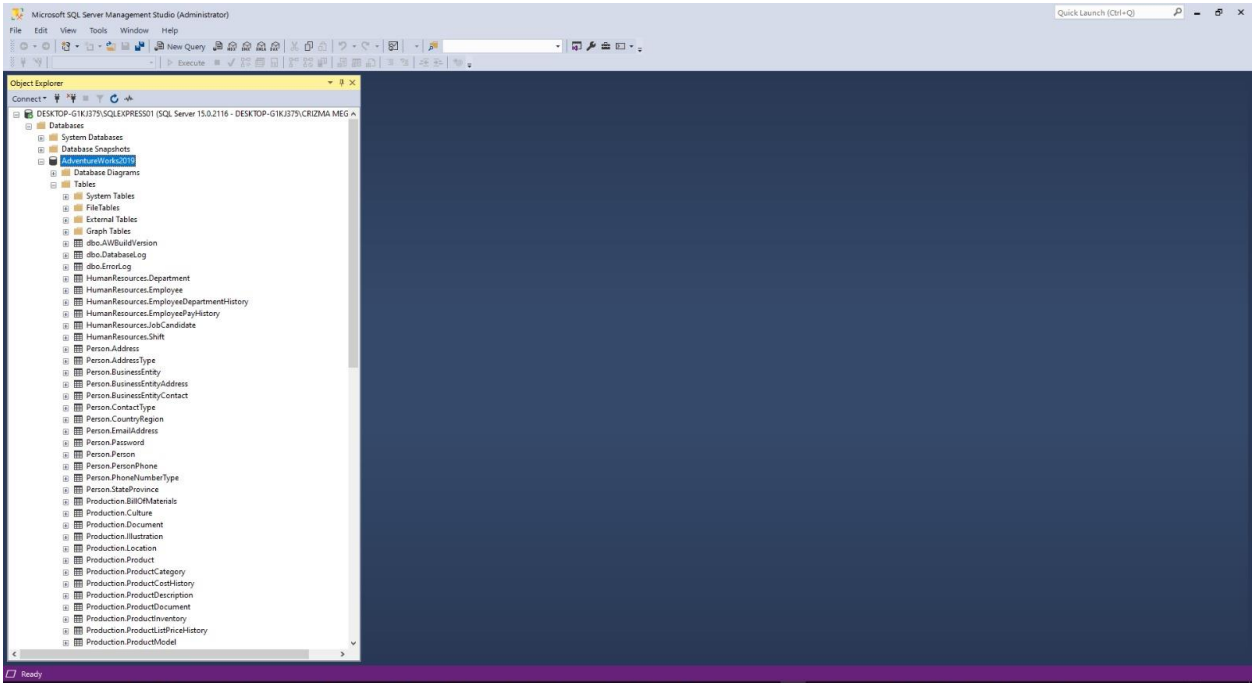
https://learn.microsoft.com/en-us/sql/samples/adventureworks-install-configure?view=sql-server-ver16&tabs=ssms

1.2. Restoring the 'AdventureWorks2019.bak' file using SQL Server Management Studio.




Microsoft SQL Server Management Studio (Administrator)
Object Explorer
Restore Database -
No backupset selected to be restored.
Locate Backup File - DESKTOP-G1KJ375\SQLEXPRESS01
AdventureWorks2019.bak
File name: AdventureWorks2019.bak
OK
Cancel
Help
Ready


Microsoft SQL Server Management Studio (Administrator)
Object Explorer
Restore Database - AdventureWorks2019
Restoring: C:\Program Files\Microsoft SQL Server\MSSQL15.SQLEXPRESS01\MSSQL\Backup\AdventureWorks2019.bak
Select a page
Source
Destination
Restore plan
Backup sets to restore:
OK
Cancel
Help
Ready

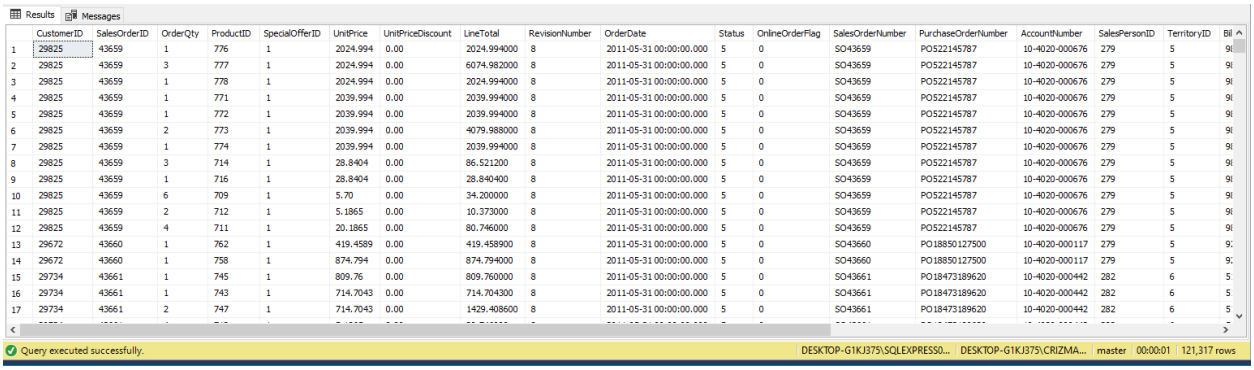


1.3. Query the data using SQL Server Management or Azure Data Studio:

After restoring the database we extracted the customers' transactions data by running an SQL query against the database.

```
SELECT TOP (200000) [AdventureWorks2019].[Sales].[SalesOrderHeader].[CustomerID]
      ,[AdventureWorks2019].[Sales].[SalesOrderHeader].[SalesOrderID]
      ,[AdventureWorks2019].[Sales].[SalesOrderDetail].[OrderQty]
      ,[AdventureWorks2019].[Sales].[SalesOrderDetail].[ProductID]
      ,[AdventureWorks2019].[Sales].[SalesOrderDetail].[SpecialOfferID]
      ,[AdventureWorks2019].[Sales].[SalesOrderDetail].[UnitPrice]
      ,[AdventureWorks2019].[Sales].[SalesOrderDetail].[UnitPriceDiscount]
      ,[AdventureWorks2019].[Sales].[SalesOrderDetail].[LineTotal]
      ,[AdventureWorks2019].[Sales].[SalesOrderHeader].[RevisionNumber]
      ,[AdventureWorks2019].[Sales].[SalesOrderHeader].[OrderDate]
      ,[AdventureWorks2019].[Sales].[SalesOrderHeader].[Status]
      ,[AdventureWorks2019].[Sales].[SalesOrderHeader].[OnlineOrderFlag]
      ,[AdventureWorks2019].[Sales].[SalesOrderHeader].[SalesOrderNumber]
      ,[AdventureWorks2019].[Sales].[SalesOrderHeader].[PurchaseOrderNumber]
      ,[AdventureWorks2019].[Sales].[SalesOrderHeader].[AccountNumber]
      ,[AdventureWorks2019].[Sales].[SalesOrderHeader].[SalesPersonID]
      ,[AdventureWorks2019].[Sales].[SalesOrderHeader].[TerritoryID]
      ,[AdventureWorks2019].[Sales].[SalesOrderHeader].[BillToAddressID]
      ,[AdventureWorks2019].[Sales].[SalesOrderHeader].[ShipToAddressID]
      ,[AdventureWorks2019].[Sales].[SalesOrderHeader].[ShipMethodID]
      ,[AdventureWorks2019].[Sales].[SalesOrderHeader].[CreditCardID]
      ,[AdventureWorks2019].[Sales].[SalesOrderHeader].[CurrencyRateID]
      ,[AdventureWorks2019].[Sales].[SalesOrderHeader].[SubTotal]
      ,[AdventureWorks2019].[Sales].[SalesOrderHeader].[TaxAmt]
      ,[AdventureWorks2019].[Sales].[SalesOrderHeader].[Freight]
      ,[AdventureWorks2019].[Sales].[SalesOrderHeader].[TotalDue]
  FROM [AdventureWorks2019].[Sales].[SalesOrderHeader], [AdventureWorks2019].[Sales].[SalesOrderDetail]
  WHERE [AdventureWorks2019].[Sales].[SalesOrderHeader].[SalesOrderID] = [AdventureWorks2019].[Sales].[SalesOrderDetail].[SalesOrderID]
```

Executing the SQL query extracted 121,317 rows.



Results | Messages

	CustomerID	SalesOrderID	OrderQty	ProductID	SpecialOfferID	UnitPrice	UnitPriceDiscount	LineTotal	RevisionNumber	OrderDate	Status	OnlineOrderFlag	SalesOrderNumber	PurchaseOrderNumber	AccountNumber	SalesPersonID	TerritoryID	Bil
1	29825	43659	1	776	1	2024.994	0.00	2024.994000	8	2011-05-31 00:00:00.000	5	0	SO43659	PO522145787	10-4020-000676	279	5	9(
2	29825	43659	3	777	1	2024.994	0.00	6074.982000	8	2011-05-31 00:00:00.000	5	0	SO43659	PO522145787	10-4020-000676	279	5	9(
3	29825	43659	1	778	1	2024.994	0.00	2024.994000	8	2011-05-31 00:00:00.000	5	0	SO43659	PO522145787	10-4020-000676	279	5	9(
4	29825	43659	1	771	1	2039.994	0.00	2039.994000	8	2011-05-31 00:00:00.000	5	0	SO43659	PO522145787	10-4020-000676	279	5	9(
5	29825	43659	1	772	1	2039.994	0.00	2039.994000	8	2011-05-31 00:00:00.000	5	0	SO43659	PO522145787	10-4020-000676	279	5	9(
6	29825	43659	2	773	1	2039.994	0.00	4079.988000	8	2011-05-31 00:00:00.000	5	0	SO43659	PO522145787	10-4020-000676	279	5	9(
7	29825	43659	1	774	1	2039.994	0.00	2039.994000	8	2011-05-31 00:00:00.000	5	0	SO43659	PO522145787	10-4020-000676	279	5	9(
8	29825	43659	3	714	1	28.8404	0.00	86.521200	8	2011-05-31 00:00:00.000	5	0	SO43659	PO522145787	10-4020-000676	279	5	9(
9	29825	43659	1	716	1	28.8404	0.00	28.840400	8	2011-05-31 00:00:00.000	5	0	SO43659	PO522145787	10-4020-000676	279	5	9(
10	29825	43659	6	709	1	5.70	0.00	34.200000	8	2011-05-31 00:00:00.000	5	0	SO43659	PO522145787	10-4020-000676	279	5	9(
11	29825	43659	2	712	1	5.1865	0.00	10.373000	8	2011-05-31 00:00:00.000	5	0	SO43659	PO522145787	10-4020-000676	279	5	9(
12	29825	43659	4	711	1	20.1865	0.00	80.746000	8	2011-05-31 00:00:00.000	5	0	SO43659	PO522145787	10-4020-000676	279	5	9(
13	29672	43660	1	762	1	419.4589	0.00	419.458900	8	2011-05-31 00:00:00.000	5	0	SO43660	PO18850127500	10-4020-000117	279	5	9:
14	29672	43660	1	758	1	874.794	0.00	874.794000	8	2011-05-31 00:00:00.000	5	0	SO43660	PO18850127500	10-4020-000117	279	5	9:
15	29734	43661	1	745	1	809.76	0.00	809.760000	8	2011-05-31 00:00:00.000	5	0	SO43661	PO18473189620	10-4020-000442	282	6	5
16	29734	43661	1	743	1	714.7043	0.00	714.704300	8	2011-05-31 00:00:00.000	5	0	SO43661	PO18473189620	10-4020-000442	282	6	5
17	29734	43661	2	747	1	714.7043	0.00	1429.408600	8	2011-05-31 00:00:00.000	5	0	SO43661	PO18473189620	10-4020-000442	282	6	5

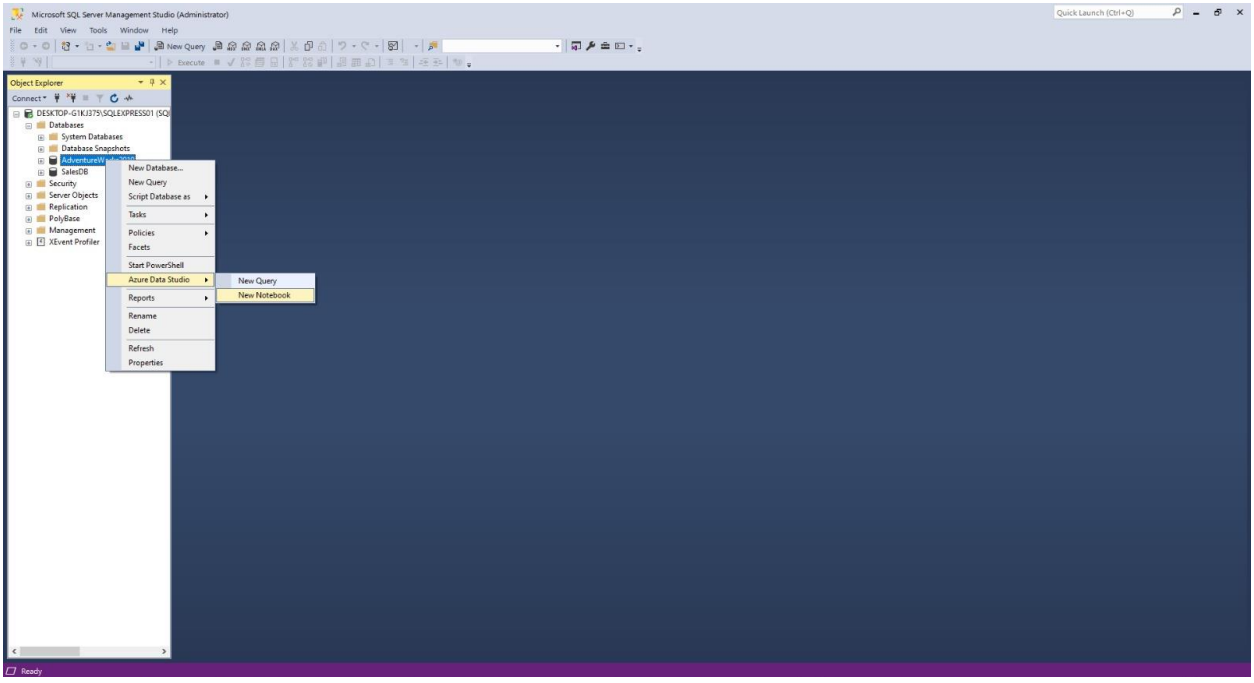
Query executed successfully. | DESKTOP-G1KJ375\SQLEXPRESS0... | DESKTOP-G1KJ375\CRIZMA... | master | 00:00:01 | 121,317 rows

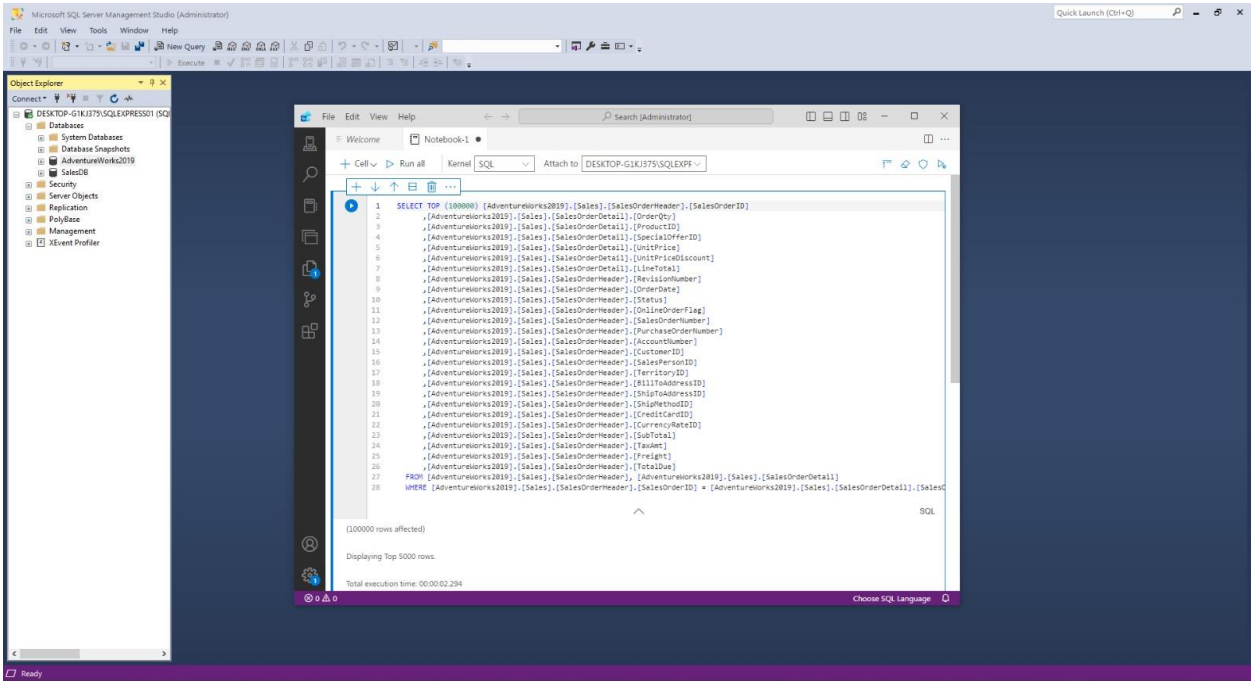
1.4. Exporting the data to a .csv file:

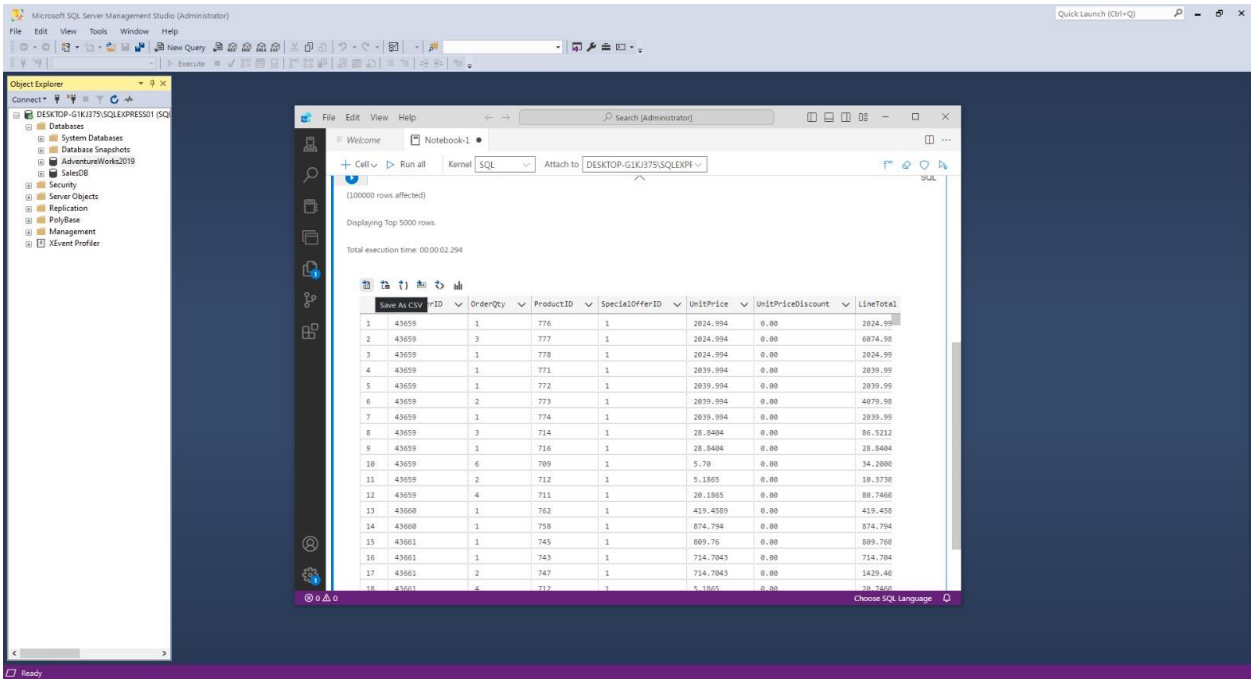
To work with the data and manipulate it in Python we needed to transform it to .CSV (or .XLSX) file.

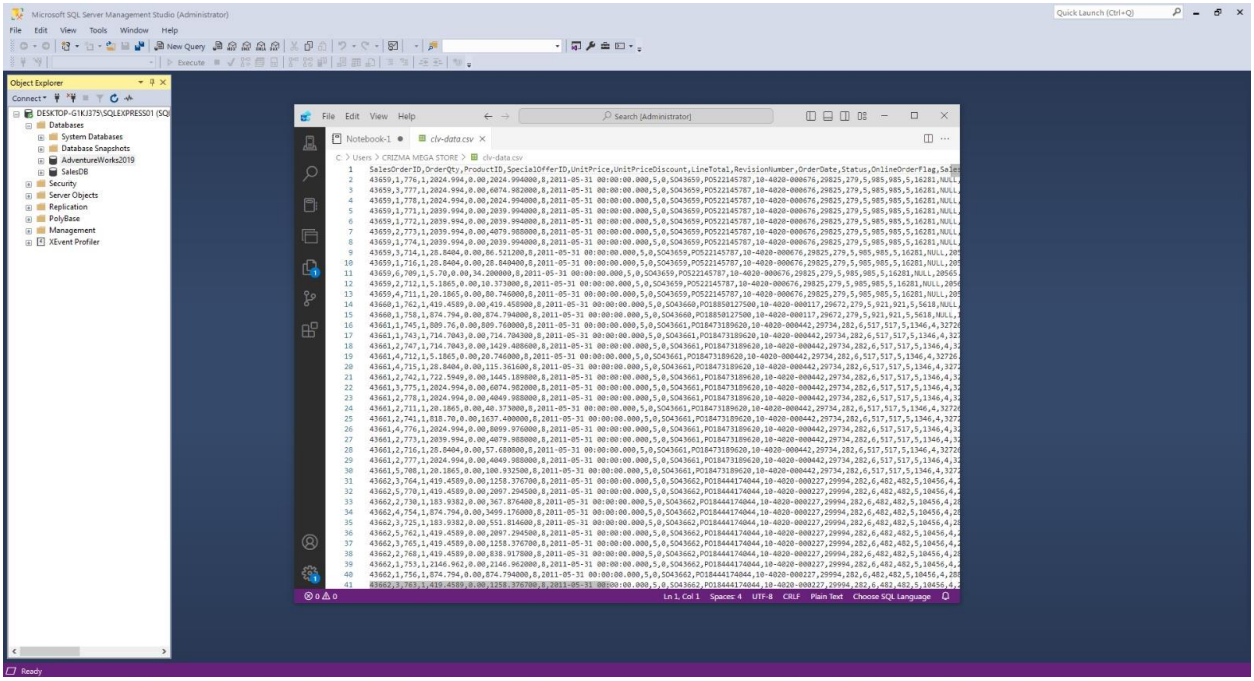
We used two methods to export our data.

1.4.1. Azure Data Studio:



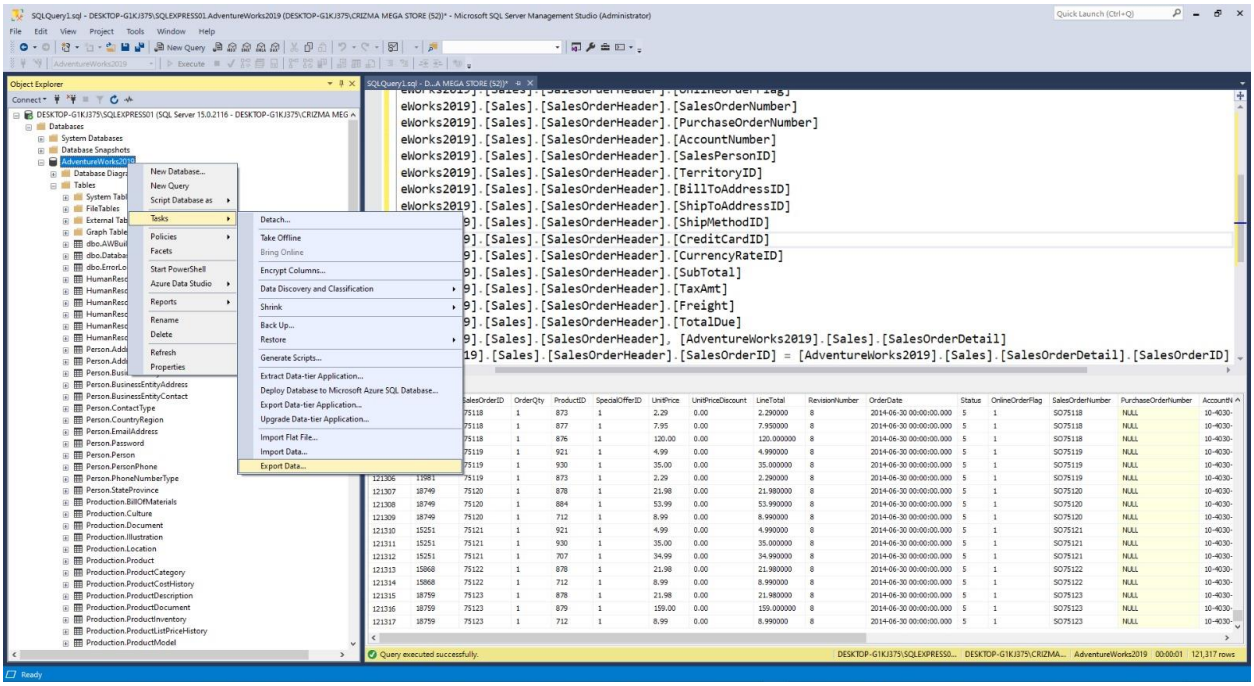

Microsoft SQL Server Management Studio (Administrator)
Object Explorer
Notebook-1
SELECT TOP (100000) [AdventureWorks2019].[Sales].[SalesOrderHeader].[SalesOrderID]
(100000 rows affected)
Displaying Top 5000 rows.
Total execution time: 00:00:02.294
Choose SQL Language
Ready

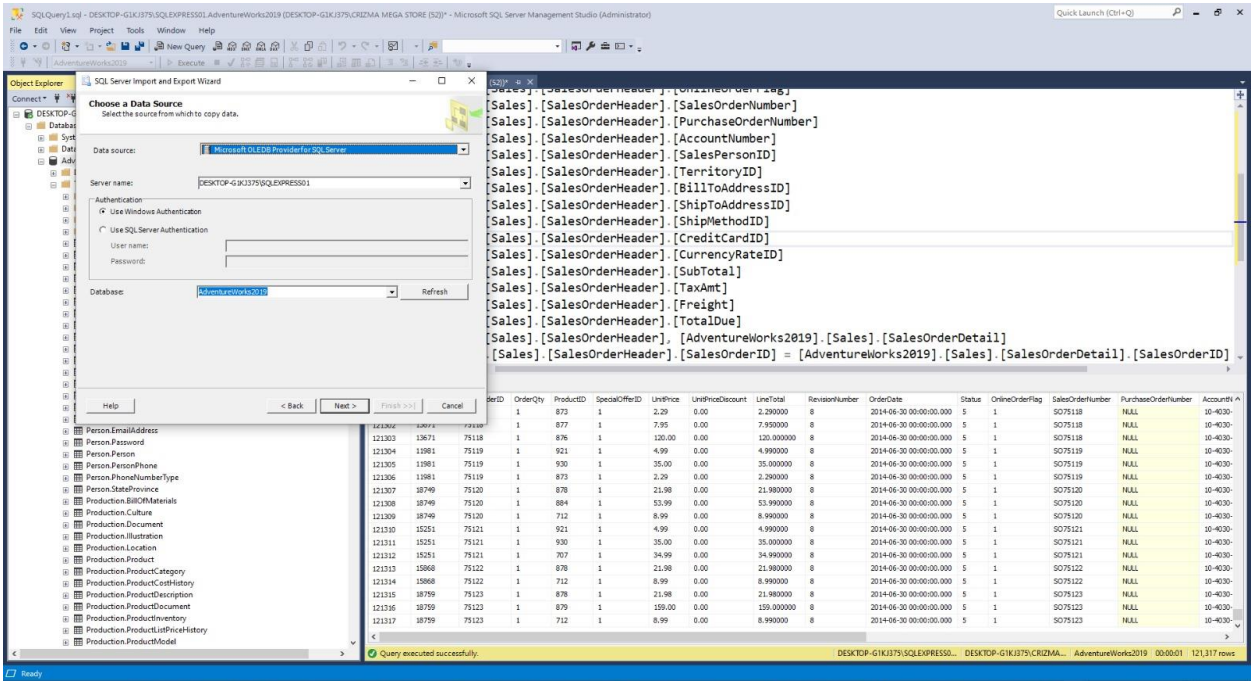

Microsoft SQL Server Management Studio (Administrator)
Object Explorer
Notebook-1
(100000 rows affected)
Displaying Top 5000 rows.
Total execution time: 00:00:02.294
Save As CSV
Choose SQL Language
Ready

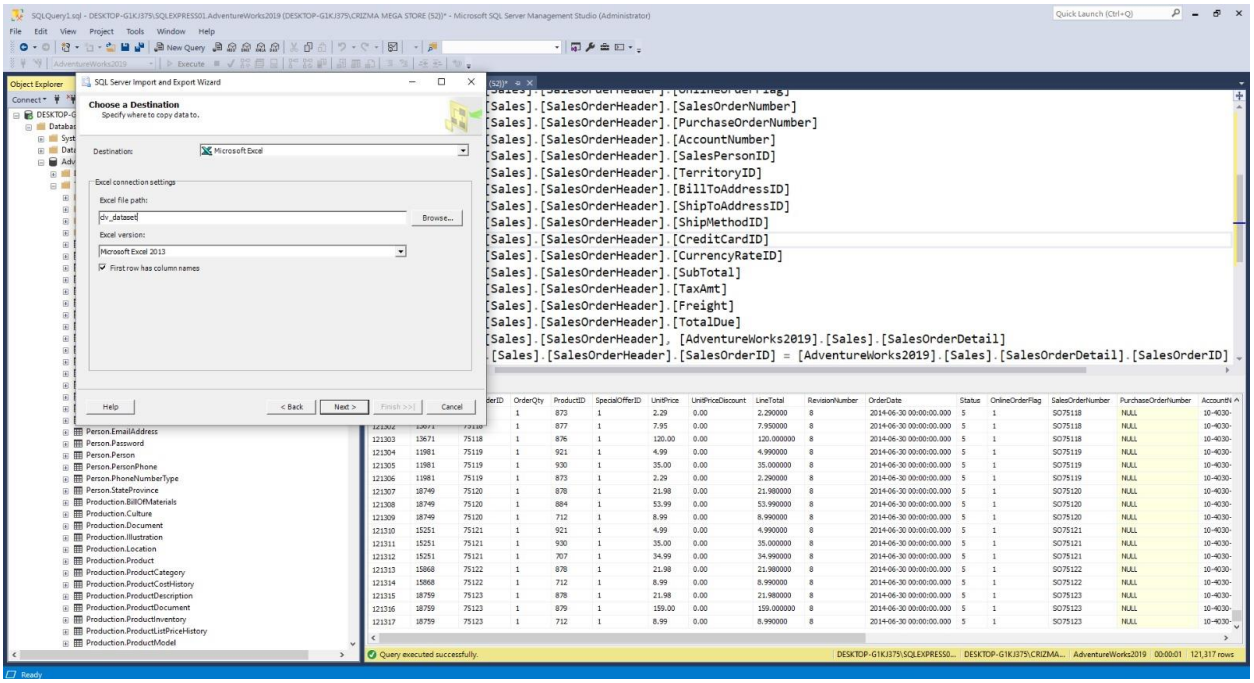


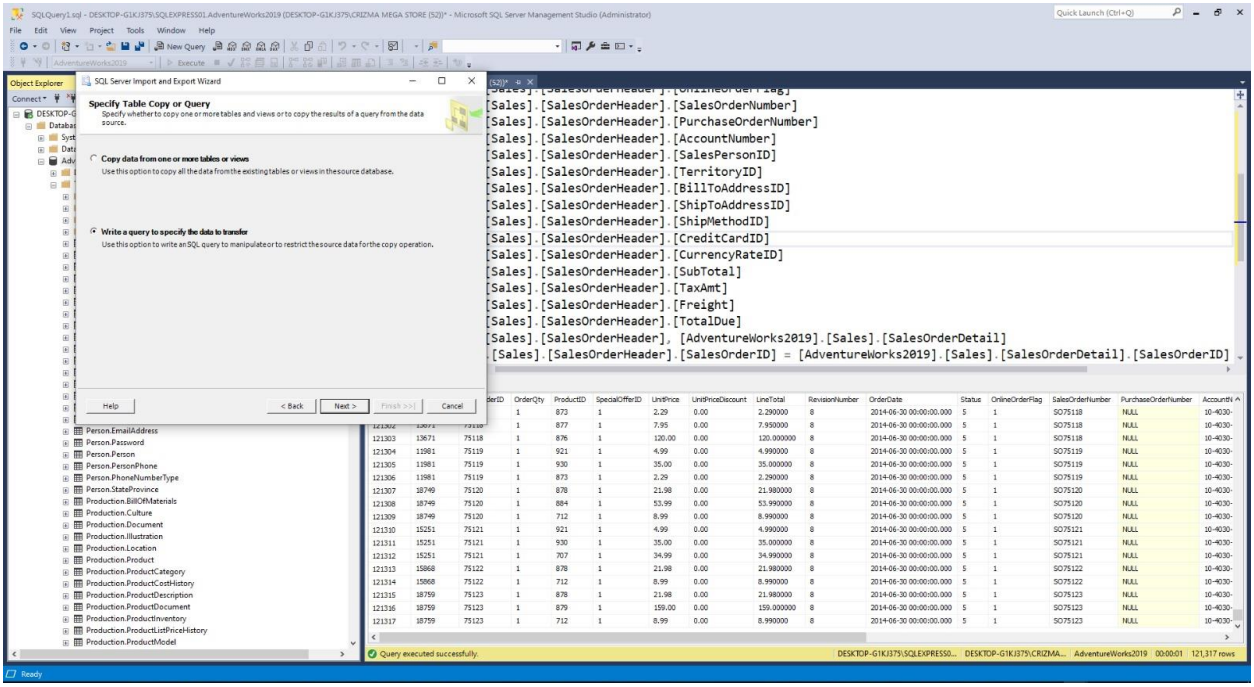
The issue with this method is its exporting limit to just export 5,000 rows of the data. We used different method to extract the full dataset.

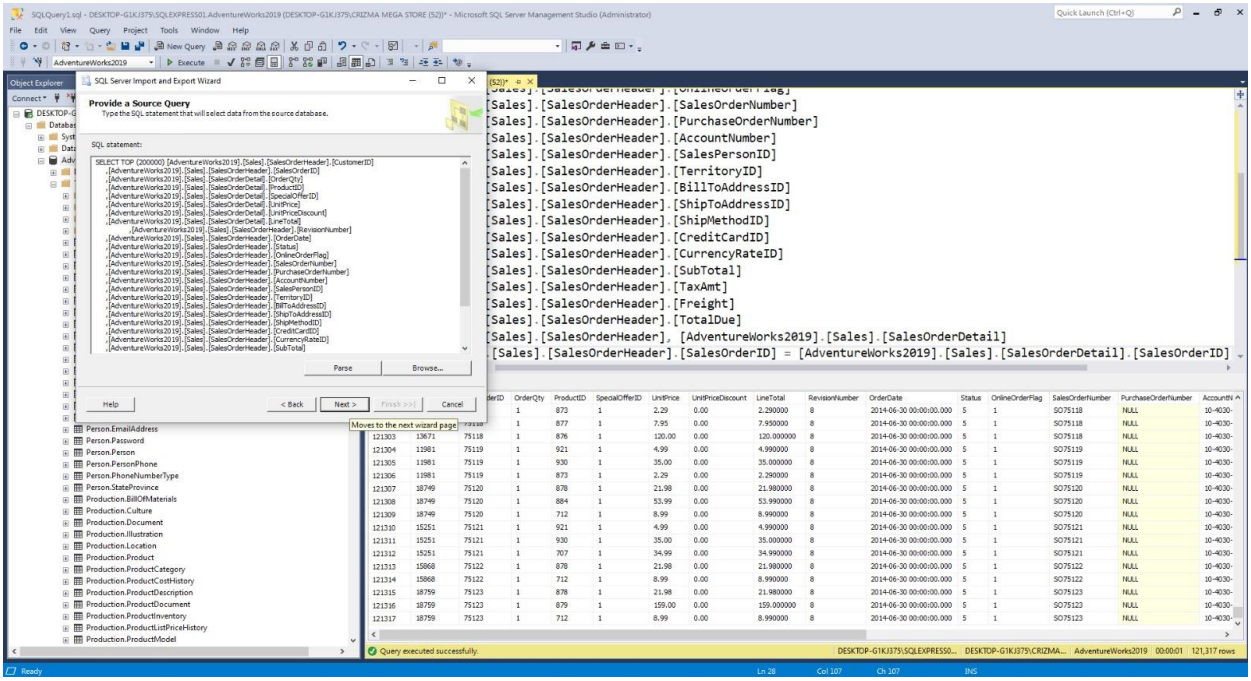
1.4.2. SQL Management Server:

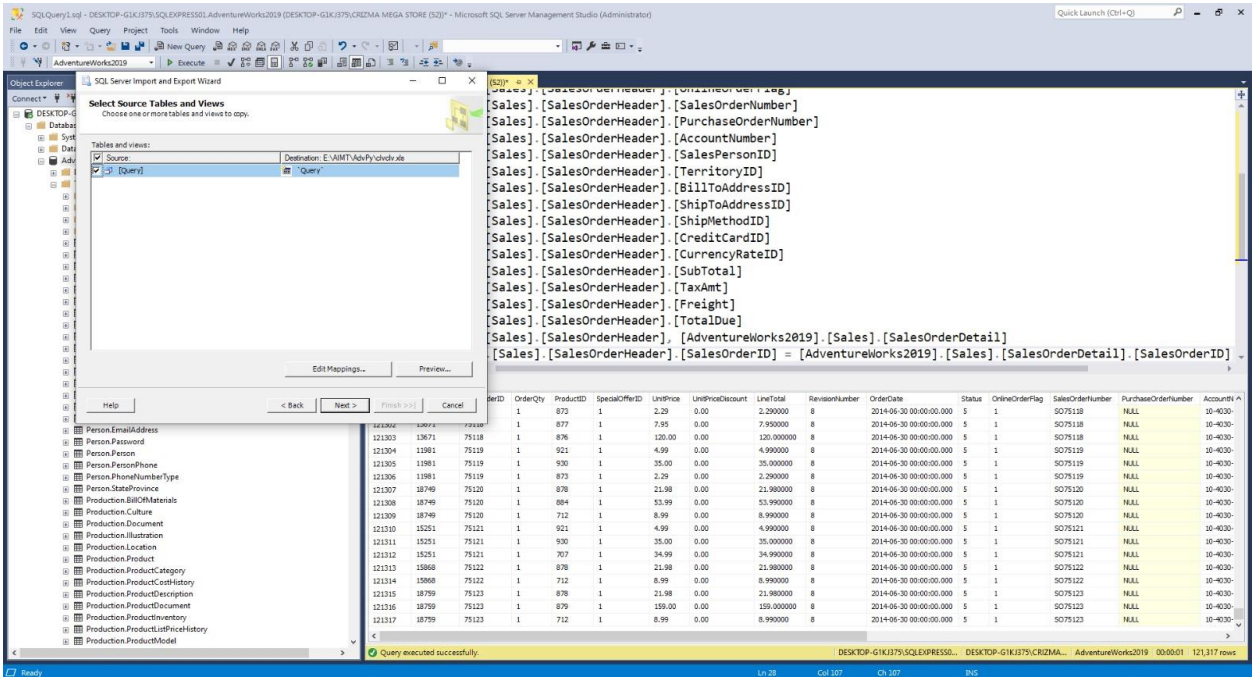


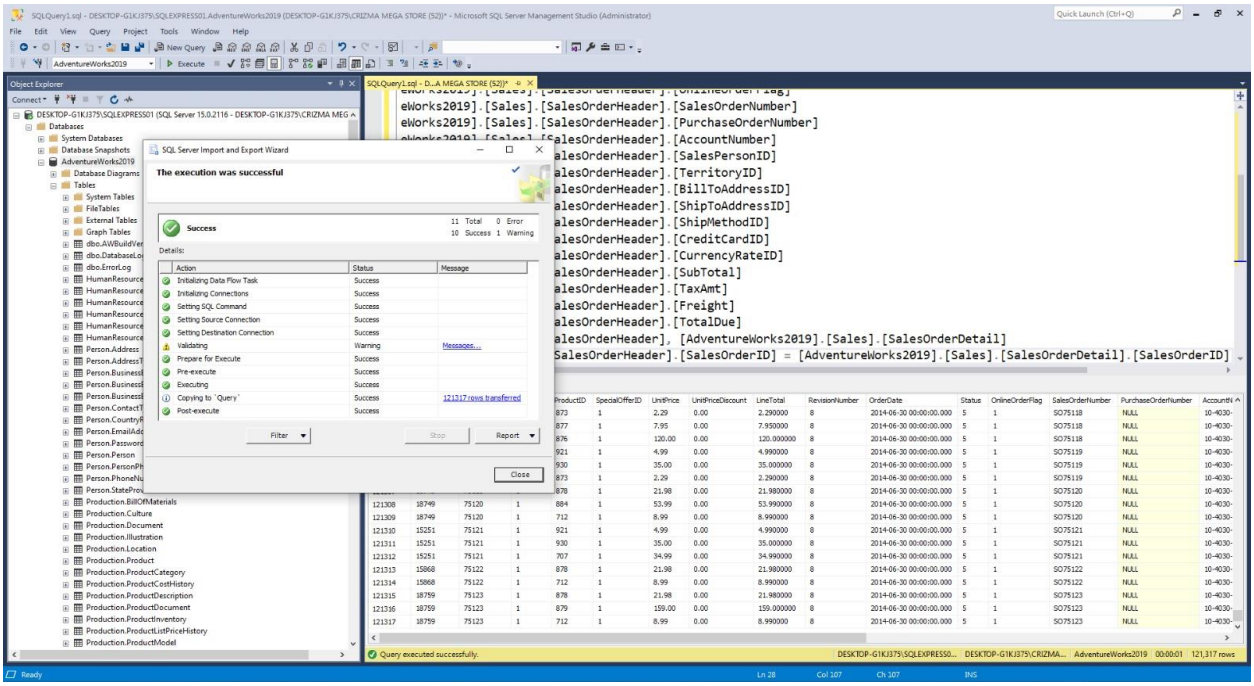

SQL Server Import and Export Wizard
Choose a Data Source
Select the source from which to copy data.
Data source:
Microsoft OLEDB Provider for SQL Server
Server name:
DESKTOP-G1KJ375\SQLEXPRESS01
Authentication
Use Windows Authentication
Use SQL Server Authentication
User name:
Password:
Database:
AdventureWorks2019
Refresh
Help
< Back
Next >
Finish >>|
Cancel
[Sales].[SalesOrderHeader].[SalesOrderNumber]
[Sales].[SalesOrderHeader].[PurchaseOrderNumber]
[Sales].[SalesOrderHeader].[AccountNumber]
[Sales].[SalesOrderHeader].[SalesPersonID]
[Sales].[SalesOrderHeader].[TerritoryID]
[Sales].[SalesOrderHeader].[BillToAddressID]
[Sales].[SalesOrderHeader].[ShipToAddressID]
[Sales].[SalesOrderHeader].[ShipMethodID]
[Sales].[SalesOrderHeader].[CreditCardID]
[Sales].[SalesOrderHeader].[CurrencyRateID]
[Sales].[SalesOrderHeader].[SubTotal]
[Sales].[SalesOrderHeader].[TaxAmt]
[Sales].[SalesOrderHeader].[Freight]
[Sales].[SalesOrderHeader].[TotalDue]
[Sales].[SalesOrderHeader], [AdventureWorks2019].[Sales].[SalesOrderDetail]
[Sales].[SalesOrderHeader].[SalesOrderID] = [AdventureWorks2019].[Sales].[SalesOrderDetail].[SalesOrderID]
Query executed successfully.
Ready


SQL Server Import and Export Wizard
Choose a Destination
Specify where to copy data to.
Destination:
Microsoft Excel
Excel connection settings
Excel file path:
clv_datasets
Browse...
Excel version:
Microsoft Excel 2013
First row has column names
Help
< Back
Next >
Finish >>|
Cancel
[Sales].[SalesOrderHeader].[SalesOrderNumber]
[Sales].[SalesOrderHeader].[PurchaseOrderNumber]
[Sales].[SalesOrderHeader].[AccountNumber]
[Sales].[SalesOrderHeader].[SalesPersonID]
[Sales].[SalesOrderHeader].[TerritoryID]
[Sales].[SalesOrderHeader].[BillToAddressID]
[Sales].[SalesOrderHeader].[ShipToAddressID]
[Sales].[SalesOrderHeader].[ShipMethodID]
[Sales].[SalesOrderHeader].[CreditCardID]
[Sales].[SalesOrderHeader].[CurrencyRateID]
[Sales].[SalesOrderHeader].[SubTotal]
[Sales].[SalesOrderHeader].[TaxAmt]
[Sales].[SalesOrderHeader].[Freight]
[Sales].[SalesOrderHeader].[TotalDue]
[Sales].[SalesOrderHeader], [AdventureWorks2019].[Sales].[SalesOrderDetail]
[Sales].[SalesOrderHeader].[SalesOrderID] = [AdventureWorks2019].[Sales].[SalesOrderDetail].[SalesOrderID]
Query executed successfully.
Ready


SQLQuery1.sql - DESKTOP-G1KJ375\SQLEXPRESS01.AdventureWorks2019 (DESKTOP-G1KJ375\CRIZMA MEGA STORE (52))* - Microsoft SQL Server Management Studio (Administrator)
SQL Server Import and Export Wizard
Specify Table Copy or Query
Specify whether to copy one or more tables and views or to copy the results of a query from the data source.
Copy data from one or more tables or views
Use this option to copy all the data from the existing tables or views in the source database.
Write a query to specify the data to transfer
Use this option to write an SQL query to manipulate or to restrict the source data for the copy operation.
Help
< Back
Next >
Finish >>|
Cancel
[Sales].[SalesOrderHeader].[SalesOrderNumber]
[Sales].[SalesOrderHeader].[PurchaseOrderNumber]
[Sales].[SalesOrderHeader].[AccountNumber]
[Sales].[SalesOrderHeader].[SalesPersonID]
[Sales].[SalesOrderHeader].[TerritoryID]
[Sales].[SalesOrderHeader].[BillToAddressID]
[Sales].[SalesOrderHeader].[ShipToAddressID]
[Sales].[SalesOrderHeader].[ShipMethodID]
[Sales].[SalesOrderHeader].[CreditCardID]
[Sales].[SalesOrderHeader].[CurrencyRateID]
[Sales].[SalesOrderHeader].[SubTotal]
[Sales].[SalesOrderHeader].[TaxAmt]
[Sales].[SalesOrderHeader].[Freight]
[Sales].[SalesOrderHeader].[TotalDue]
[Sales].[SalesOrderHeader], [AdventureWorks2019].[Sales].[SalesOrderDetail]
[Sales].[SalesOrderHeader].[SalesOrderID] = [AdventureWorks2019].[Sales].[SalesOrderDetail].[SalesOrderID]
Query executed successfully.
Ready


SQLQuery1.sql - DESKTOP-G1KJ375\SQLEXPRESS01.AdventureWorks2019 (DESKTOP-G1KJ375\CRIZMA MEGA STORE (52))* - Microsoft SQL Server Management Studio (Administrator)
SQL Server Import and Export Wizard
Provide a Source Query
Type the SQL statement that will select data from the source database.
SQL statement:
SELECT TOP (200000) [AdventureWorks2019].[Sales].[SalesOrderHeader].[CustomerID]
Parse
Browse...
Help
< Back
Next >
Finish >>|
Cancel
Moves to the next wizard page
[Sales].[SalesOrderHeader].[SalesOrderNumber]
[Sales].[SalesOrderHeader].[PurchaseOrderNumber]
[Sales].[SalesOrderHeader].[AccountNumber]
[Sales].[SalesOrderHeader].[SalesPersonID]
[Sales].[SalesOrderHeader].[TerritoryID]
[Sales].[SalesOrderHeader].[BillToAddressID]
[Sales].[SalesOrderHeader].[ShipToAddressID]
[Sales].[SalesOrderHeader].[ShipMethodID]
[Sales].[SalesOrderHeader].[CreditCardID]
[Sales].[SalesOrderHeader].[CurrencyRateID]
[Sales].[SalesOrderHeader].[SubTotal]
[Sales].[SalesOrderHeader].[TaxAmt]
[Sales].[SalesOrderHeader].[Freight]
[Sales].[SalesOrderHeader].[TotalDue]
[Sales].[SalesOrderHeader], [AdventureWorks2019].[Sales].[SalesOrderDetail]
[Sales].[SalesOrderHeader].[SalesOrderID] = [AdventureWorks2019].[Sales].[SalesOrderDetail].[SalesOrderID]
Query executed successfully.
Ready





1.5. Reading the dataset:

We used the Pandas library to read the CSV file and store it in the Padas data frame.

	CustomerID	SalesOrderID	OrderQty	ProductID	SpecialOfferID	UnitPrice	UnitPriceDiscount	LineTotal	RevisionNumber	OrderDate	...	TerritoryID	BillToAddressID	ShipToAddressID	ShipMethodID	CreditCardID	CurrencyRateID	SubT
0	29825	43659	1	776	1	2024.994	0.0	2024.994	8	2011-05-31 00:00:00	...	5	985	985	5	16281.0	NaN	20565.
1	29825	43659	3	777	1	2024.994	0.0	6074.982	8	2011-05-31 00:00:00	...	5	985	985	5	16281.0	NaN	20565.
2	29825	43659	1	778	1	2024.994	0.0	2024.994	8	2011-05-31 00:00:00	...	5	985	985	5	16281.0	NaN	20565.
3	29825	43659	1	771	1	2039.994	0.0	2039.994	8	2011-05-31 00:00:00	...	5	985	985	5	16281.0	NaN	20565.
4	29825	43659	1	772	1	2039.994	0.0	2039.994	8	2011-05-31 00:00:00	...	5	985	985	5	16281.0	NaN	20565.

Our dataset shape is 121,317 rows and 26 columns.

Step 2: Data Pre-Processing:

In this step, we aim to clean our data, format it correctly, ensure its integrity, and detect the outliers.

2.1. Features Selection:

Starting by selecting only the features we are interested in. For predicting the CLV we need our data to have the following columns 'CustomerID', 'SalesOrderID', 'OrderDate', 'OrderQty', 'LineTotal' to calculate the metrics values (Frequency, Recency, T, Monetary Values and Revenue).

For the EDA we need the columns 'ProductID', 'SpecialOfferID' to visualize useful findings and answer interesting questions about the business.

Our selected features data frame snapshot:

	CustomerID	SalesOrderID	ProductID	SpecialOfferID	OrderDate	OrderQty	LineTotal
0	29825	43659	776	1	2011-05-31 00:00:00	1	2024.994
1	29825	43659	777	1	2011-05-31 00:00:00	3	6074.982
2	29825	43659	778	1	2011-05-31 00:00:00	1	2024.994
3	29825	43659	771	1	2011-05-31 00:00:00	1	2039.994
4	29825	43659	772	1	2011-05-31 00:00:00	1	2039.994
...	...	...	...	...	...	...	...
121312	15868	75122	878	1	2014-06-30 00:00:00	1	21.980
121313	15868	75122	712	1	2014-06-30 00:00:00	1	8.990
121314	18759	75123	878	1	2014-06-30 00:00:00	1	21.980
121315	18759	75123	879	1	2014-06-30 00:00:00	1	159.000
121316	18759	75123	712	1	2014-06-30 00:00:00	1	8.990

121317 rows × 7 columns

2.2. Data Cleaning: Missing Values

Next step is to handle missing values in our data.

```
Percentage of missing values in df columns:
CustomerID        0.0
SalesOrderID      0.0
ProductID         0.0
SpecialOfferID    0.0
OrderDate         0.0
OrderQty          0.0
LineTotal         0.0
dtype: float64
```

We found 0 missing values in our dataset.

2.3. Data Formatting and Structuring: Data Types

Ensuring our data are stored in the right format is a crucial pre-processing step.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 121317 entries, 0 to 121316
Data columns (total 7 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   CustomerID      121317 non-null  int64
 1   SalesOrderID    121317 non-null  int64
 2   ProductID       121317 non-null  int64
 3   SpecialOfferID  121317 non-null  int64
 4   OrderDate       121317 non-null  object
 5   OrderQty        121317 non-null  int64
 6   LineTotal       121317 non-null  float64
dtypes: float64(1), int64(5), object(1)
```

We found the Orders Date values stored in the wrong format, so we need to cast it to datetime format instead of object format.

	OrderDate
0	2011-05-31
1	2011-05-31
2	2011-05-31
3	2011-05-31
4	2011-05-31

dtype: datetime64[ns]

2.4. Data Validation: Data Integrity

We investigated the data inside our columns to ensure it integrity.

```
CustomerID
29722    530
29966    482
29614    451
29950    446
30048    441
        ...
26855      1
25476      1
23935      1
18017      1
27205      1
Name: count, Length: 19119, dtype: int64
Number of unique values: 19119
Minimun value: 11000
Maximum value: 30118

SalesOrderID
51721    72
51739    72
53465    71
51160    71
47355    68
         ..
64020     1
46465     1
46464     1
46463     1
46852     1
Name: count, Length: 31465, dtype: int64
Number of unique values: 31465
Minimun value: 43659
Maximum value: 75123
```

```
ProductID
870    4688
712    3382
873    3354
921    3095
711    3090
       ...
927       9
911       6
943       6
942       5
897       2
Name: count, Length: 266, dtype: int64
Number of unique values: 266
Minimun value: 707
Maximum value: 999

SpecialOfferID
1     115884
2       3428
3        606
13       524
14       244
16       169
7        137
8         98
11        84
4         80
9         61
5          2
Name: count, dtype: int64
Number of unique values: 12
Minimun value: 1
Maximum value: 16
```

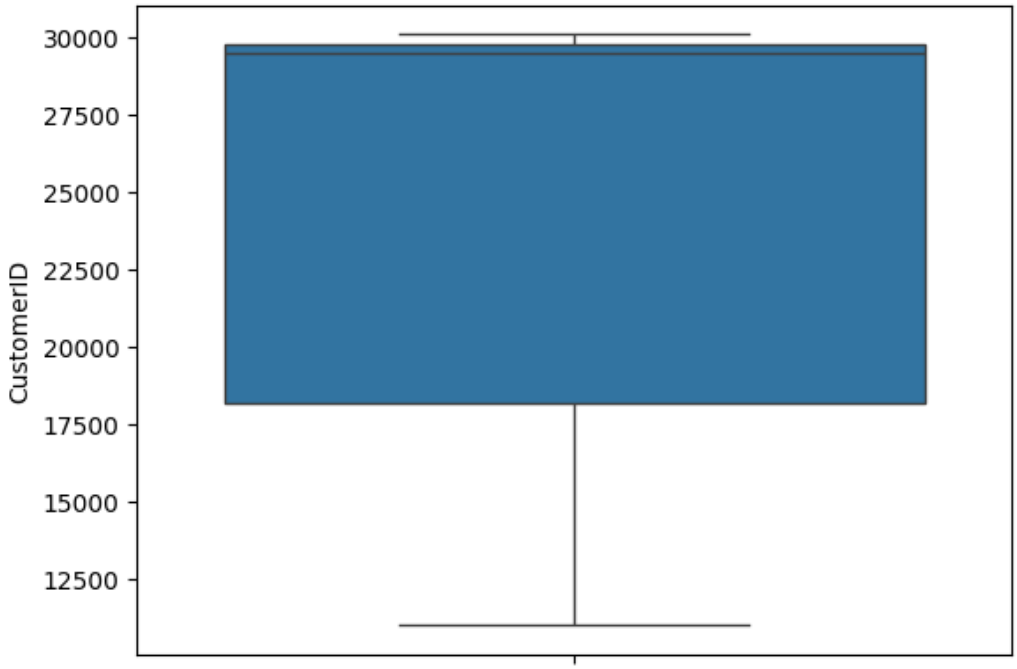
```
OrderDate
2013-06-30    3876
2013-07-31    3852
2013-10-30    3202
2014-05-01    3126
2014-03-31    3065
              ...
2012-01-21       1
2011-09-24       1
2012-10-03       1
2011-08-29       1
2012-04-27       1
Name: count, Length: 1124, dtype: int64
Number of unique values: 1124
Minimun value: 2011-05-31 00:00:00
Maximum value: 2014-06-30 00:00:00


OrderQty
1     74954
2     14200
3     10058
4      7494
5      4386
6      3298
7      1735
8      1521
9       837
10      768
12      466
11      392
14      265
13      230
16      133
15      119
18      101
17       94
19       53
20       46
21       31
23       23
24       19
25       17
26       15
22       12
27        9
32        7
33        6
28        6
31        5
34        3
29        3
35        2
36        2
40        2
39        1
44        1
30        1
38        1
41        1
Name: count, dtype: int64
Number of unique values: 41
Minimun value: 1
Maximum value: 44
```

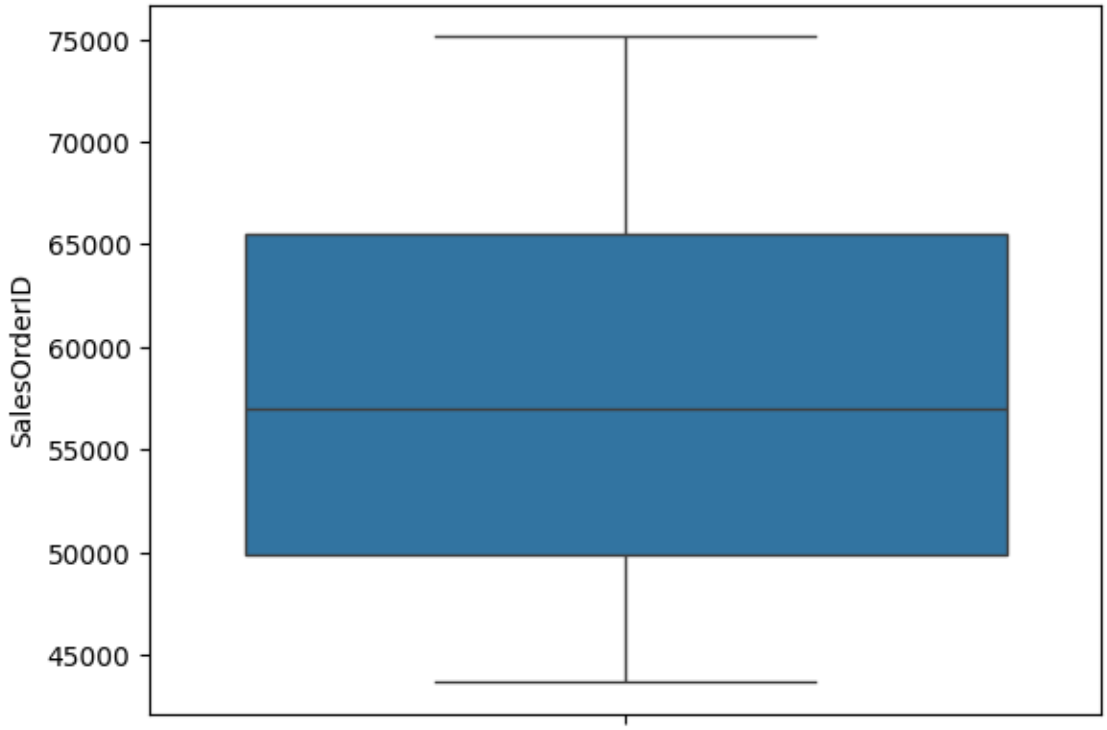
```
LineTotal
4.990000       8827
34.990000      6440
8.990000       4470
2.290000       3191
3.990000       2376
               ...
914.400000        1
7418.091772       1
1376.940000       1
89.437500         1
5078.605950       1
Name: count, Length: 1488, dtype: int64
Number of unique values: 1488
Minimun value: 1.374
Maximum value: 27893.619
```

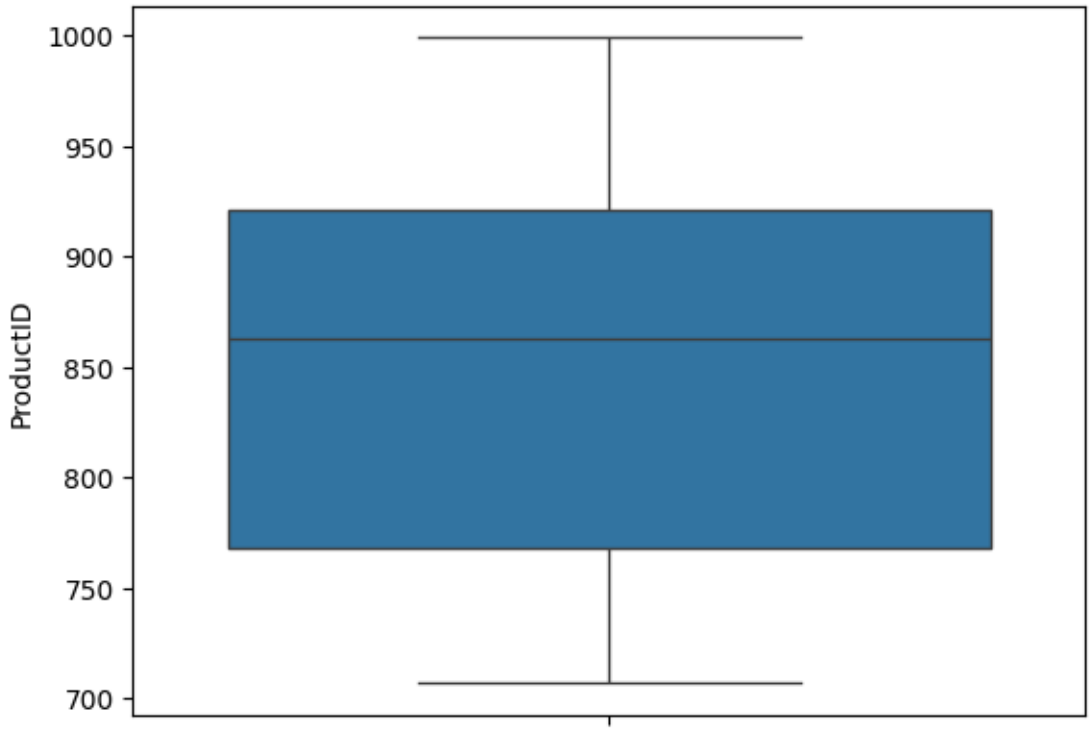
There are no issues with the data stored in our columns. We conclude that our data are healthy.

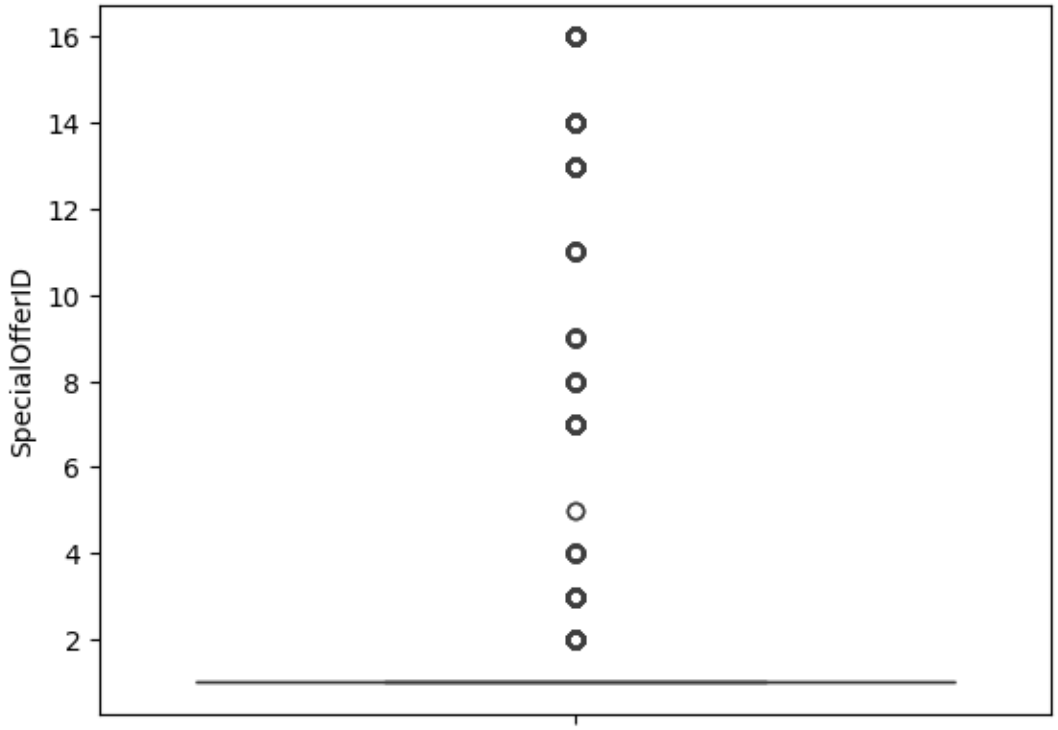
2.5. Outliers Detection:

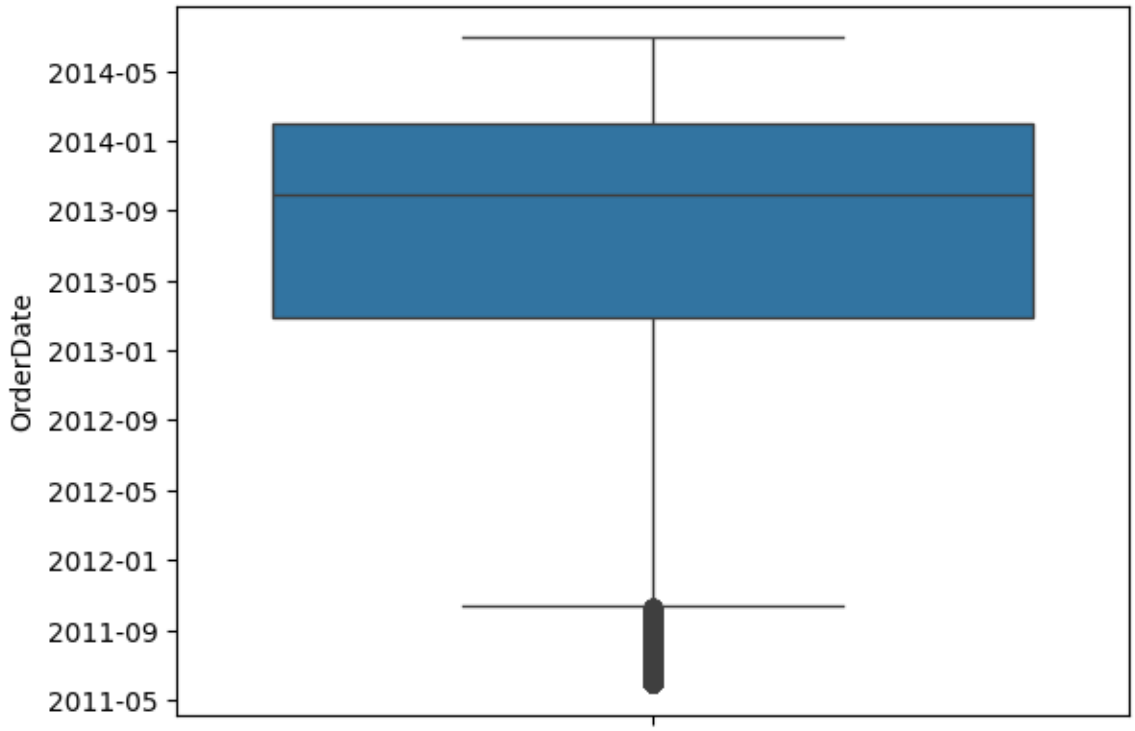
We will end our data pre-processing by defining the outliers in our dataset using boxplot, so we can handle it in the data processing step.

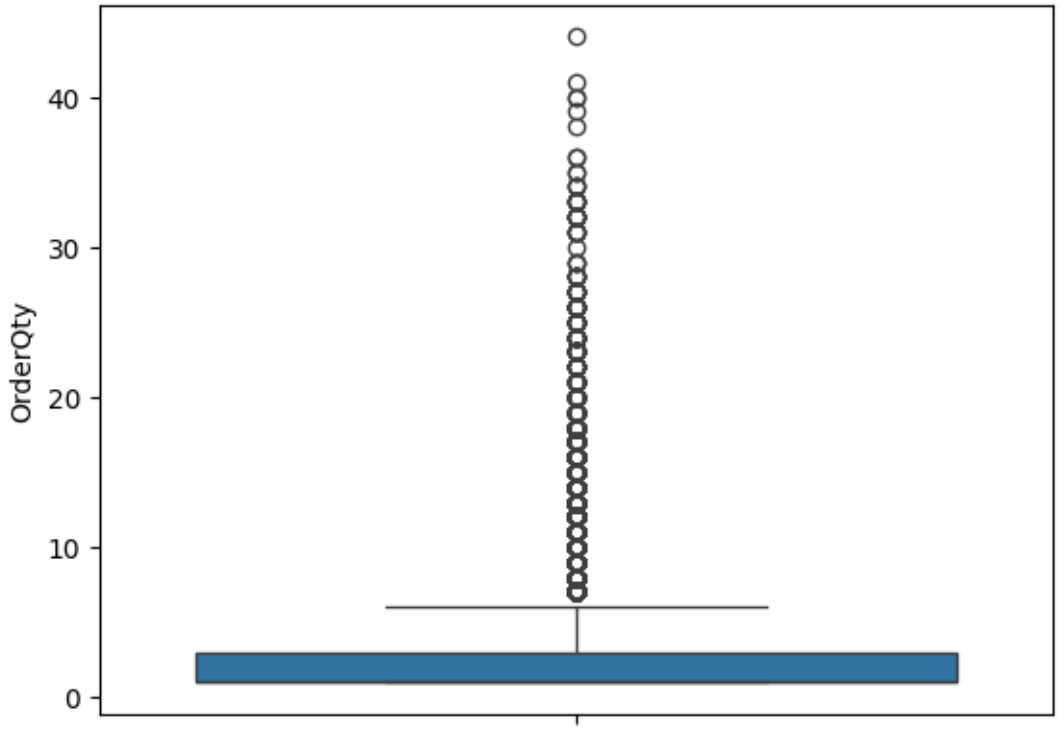


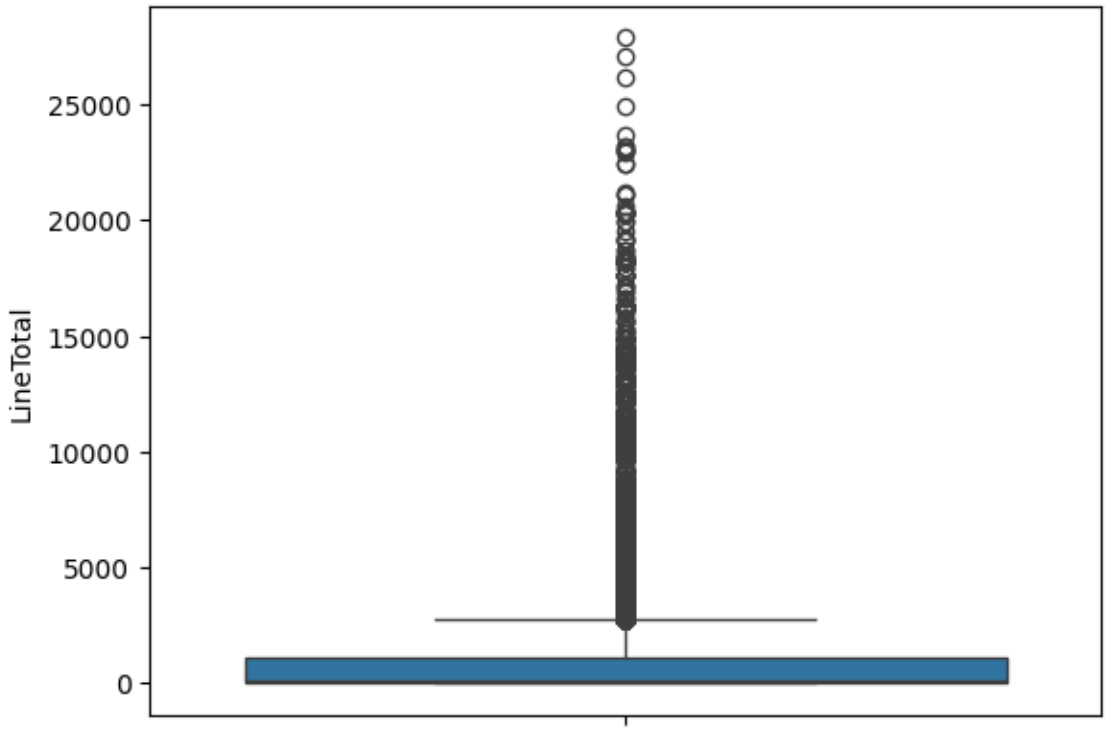

75000
70000
65000
60000
55000
50000
45000
SalesOrderID


1000
950
900
850
800
750
700
ProductID


SpecialOfferID
16
14
12
10
8
6
4
2


OrderDate
2014-05
2014-01
2013-09
2013-05
2013-01
2012-09
2012-05
2012-01
2011-09
2011-05


OrderQty
0
10
20
30
40


LineTotal
0
5000
10000
15000
20000
25000

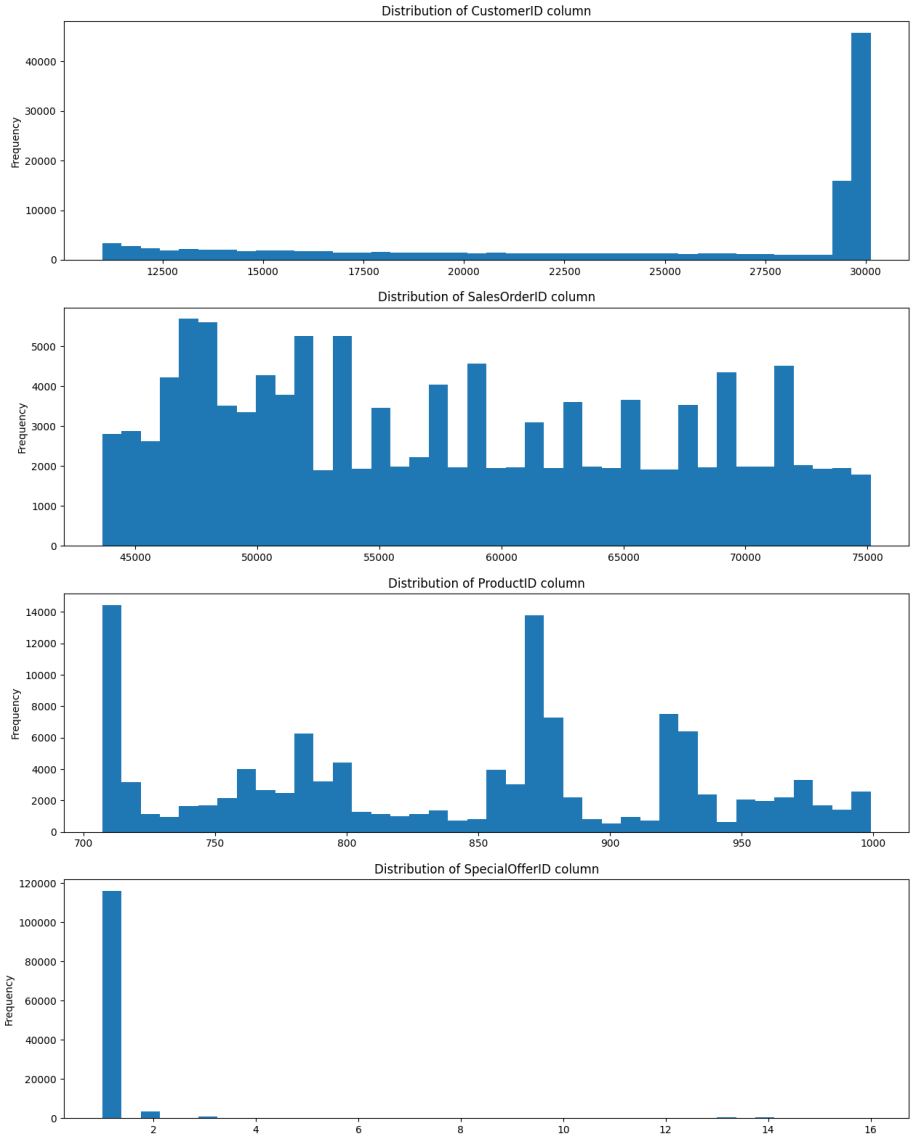
We found outliers in columns 'SpecialOfferID', 'OrderDate', 'OrderQty', 'LineTotal'. We will address it the data processing step.

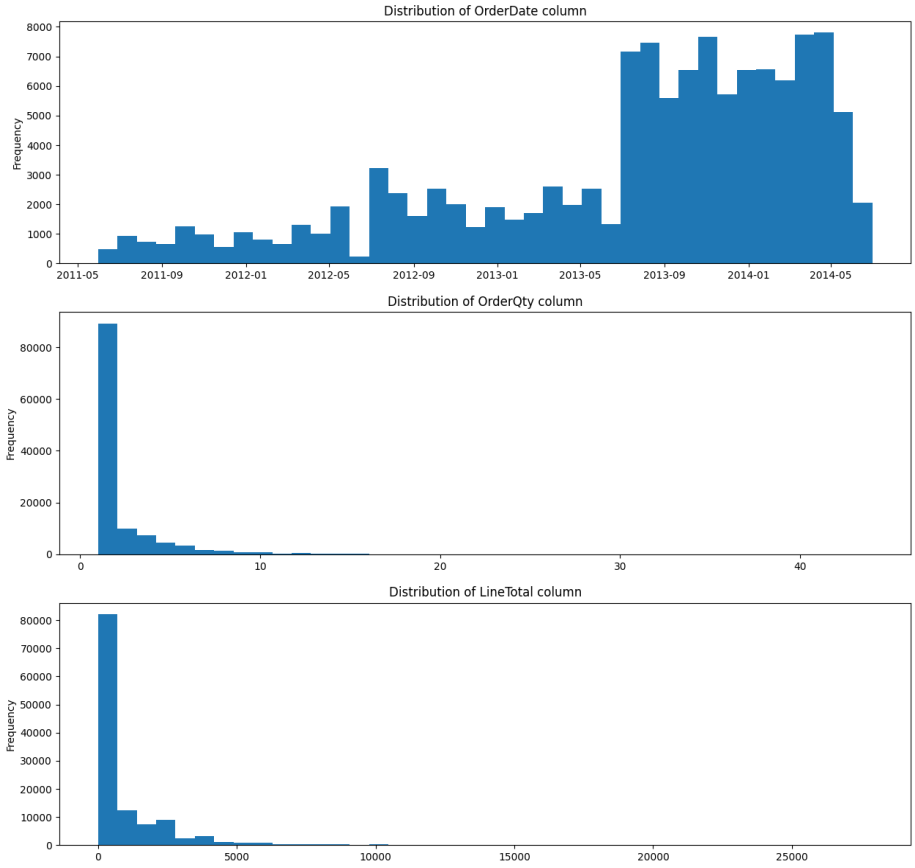
Step 3: Data Analysis:

To understand our dataset better we will perform the following data analysis techniques.

3.1. Data Distribution:

Since all our features data are numerical features we will use the histogram to plot the distribution of our features data.

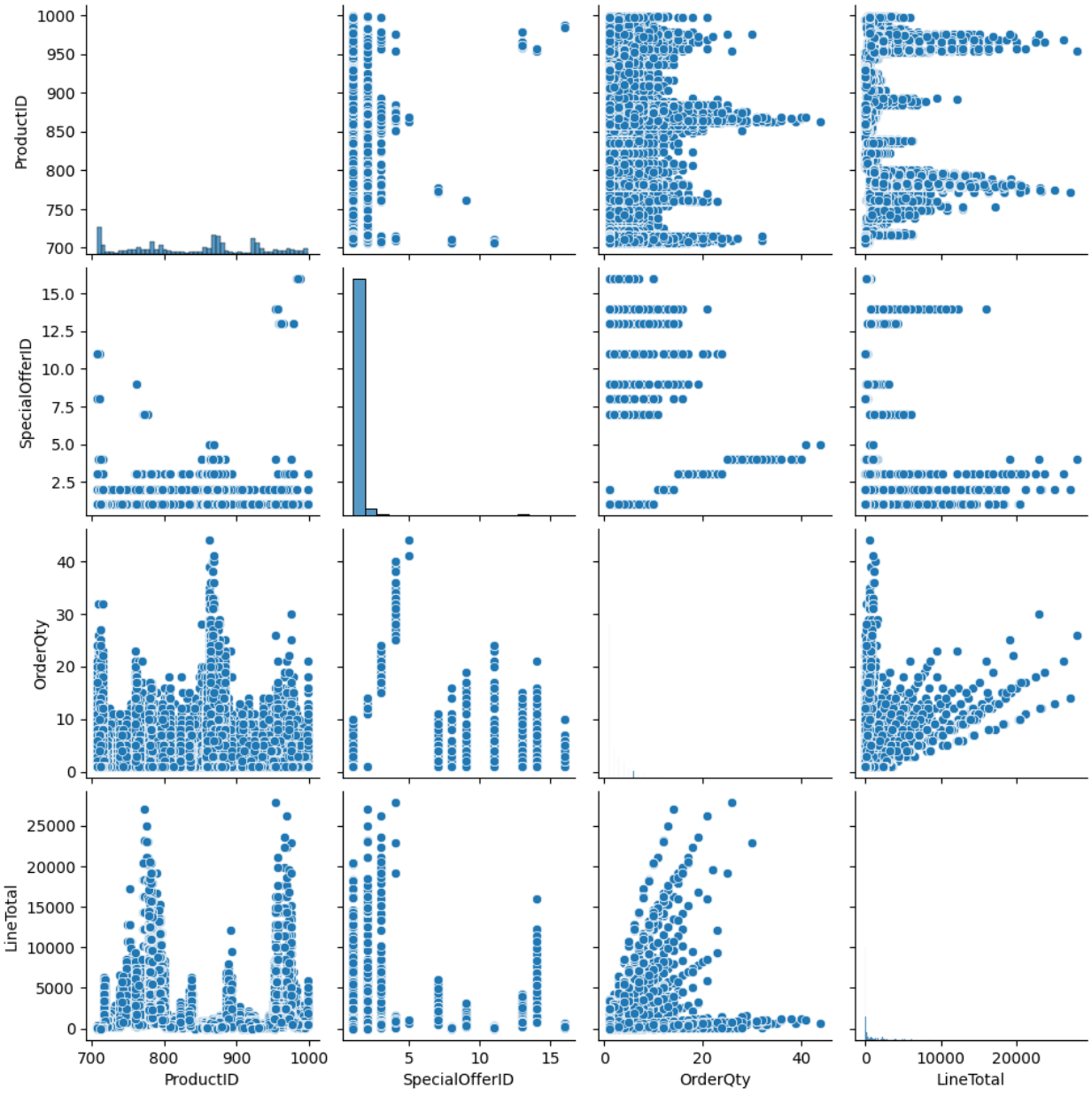




We can interpret that the 'OrderQty' and the 'LineTotal' columns are right-skewed. We can also say that the 'OrderDate' is left skewed, and the rest of the columns don't follow any distribution pattern.

3.2. Correlation Analysis: Pair plot (Scatter plot):

The next step in our analysis is to define the correlation between our columns. We used a scatter plot to visualize the correlation between our columns.



We can see there is no high correlation between our meaningful columns. But there is a sign of positive correlation between OrderQty and LineTotal which make sense.

3.3. Panda Profiling

To summaries our data profiling in one document we used Pandas profiling library.

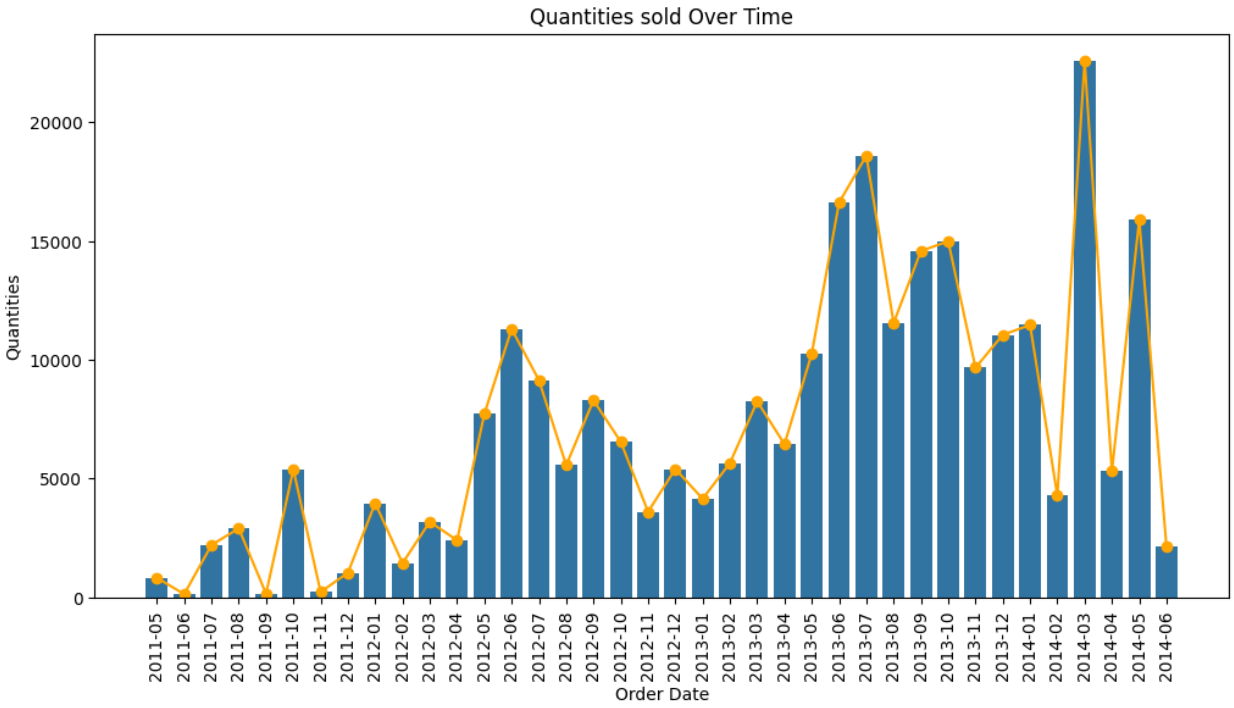
The profiling report shows the same result we already done, with 2 alerts about the correlation between the CustomerID and OrderQty which may be result of the sampling method used to

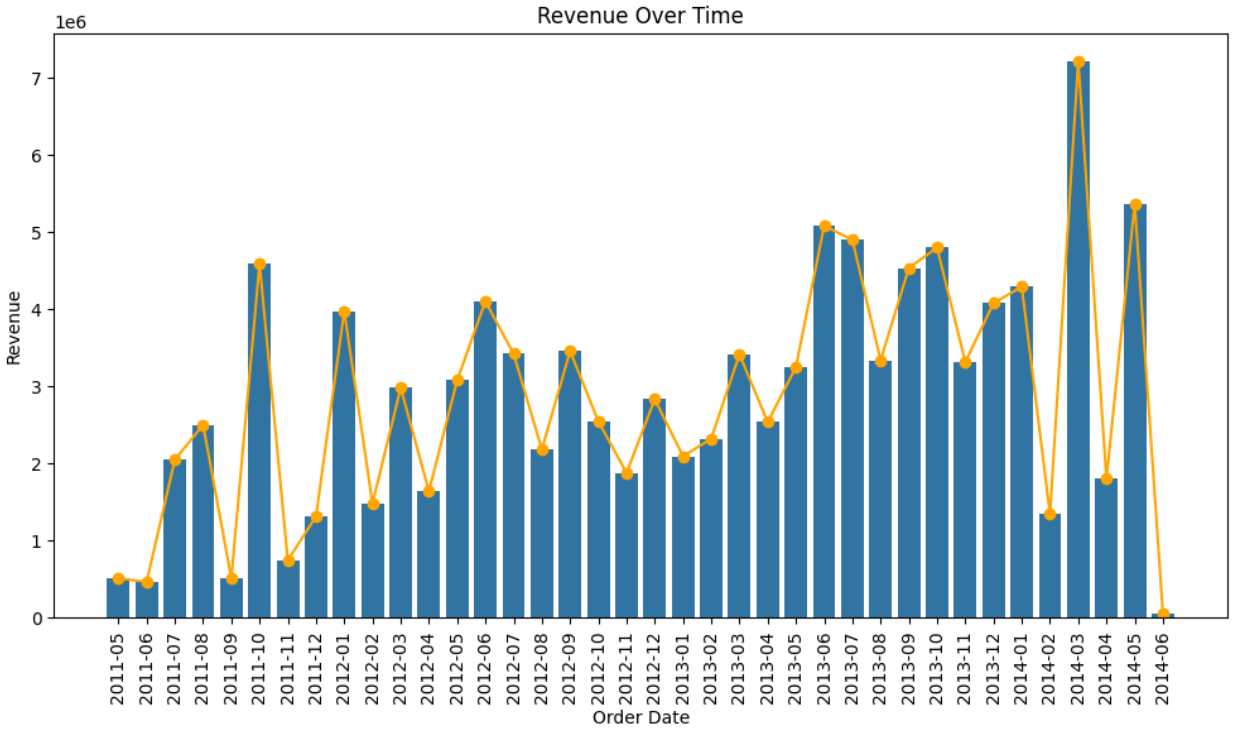
generate the data by Microsoft, and it's not worth worry about since it doesn't indicate any meaningful information.

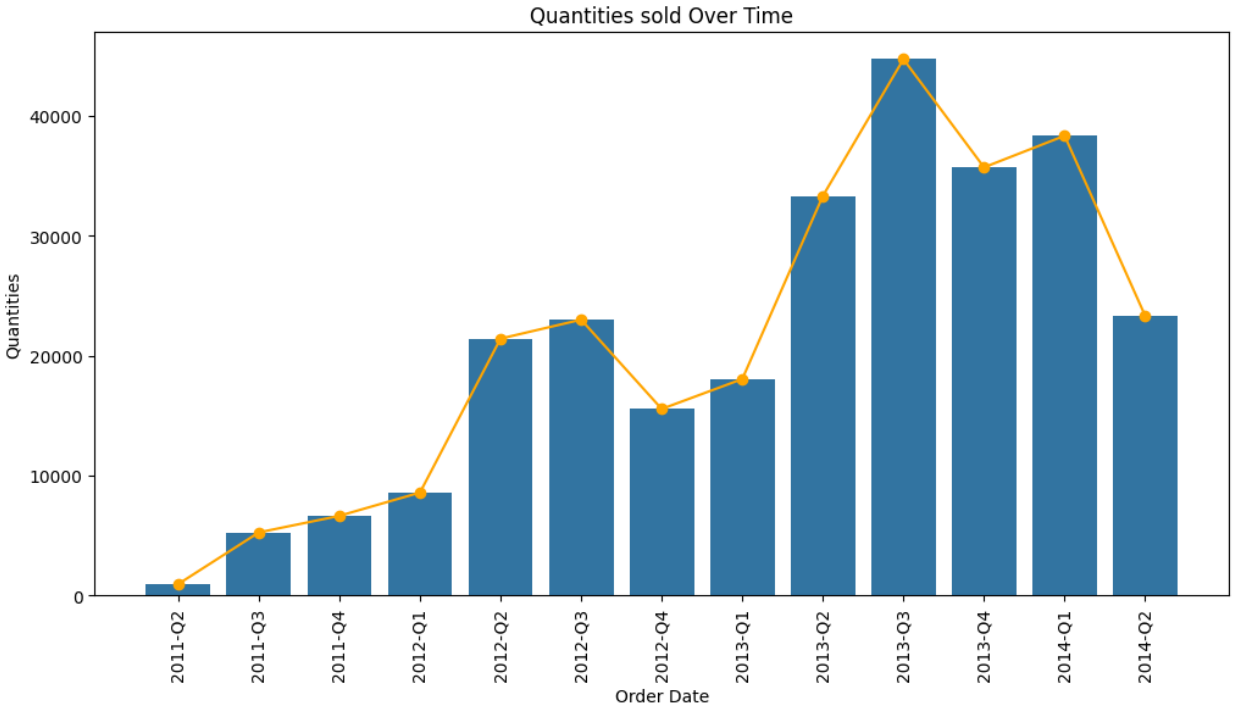
3.4. EDA:

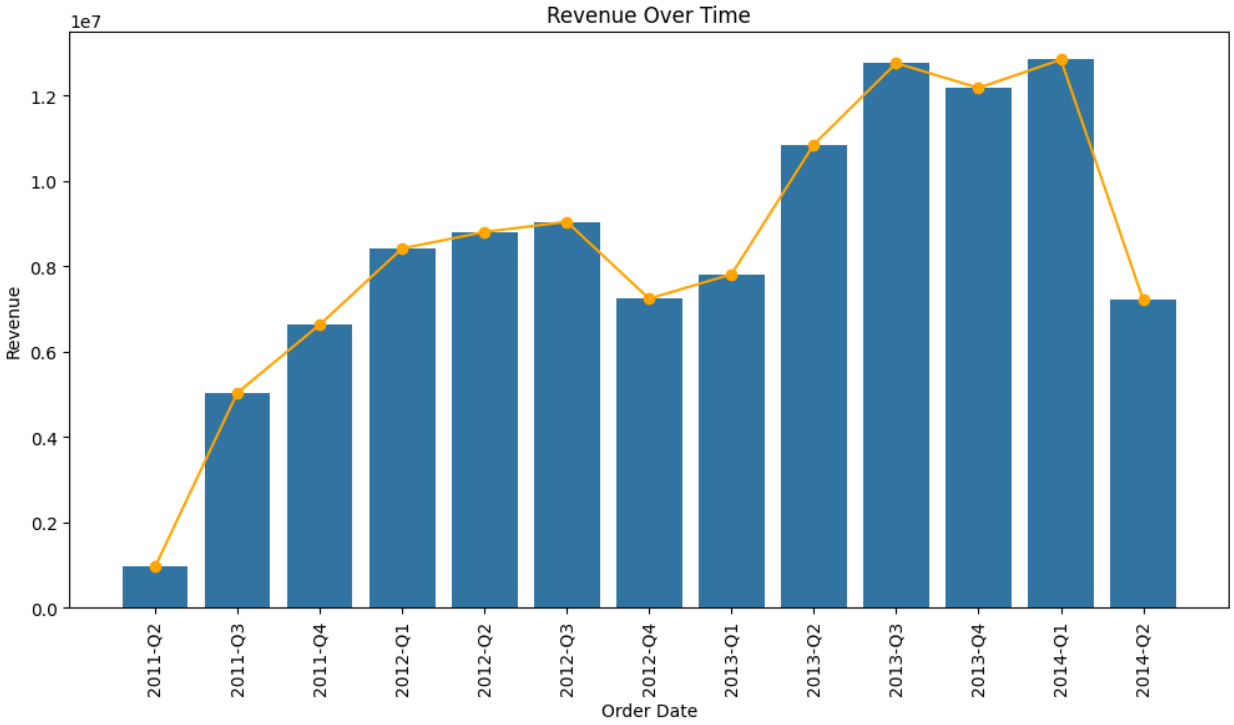
Pursuing with the data analysis we want to understand our dataset better, to do so we will explore our data by answering the following question:

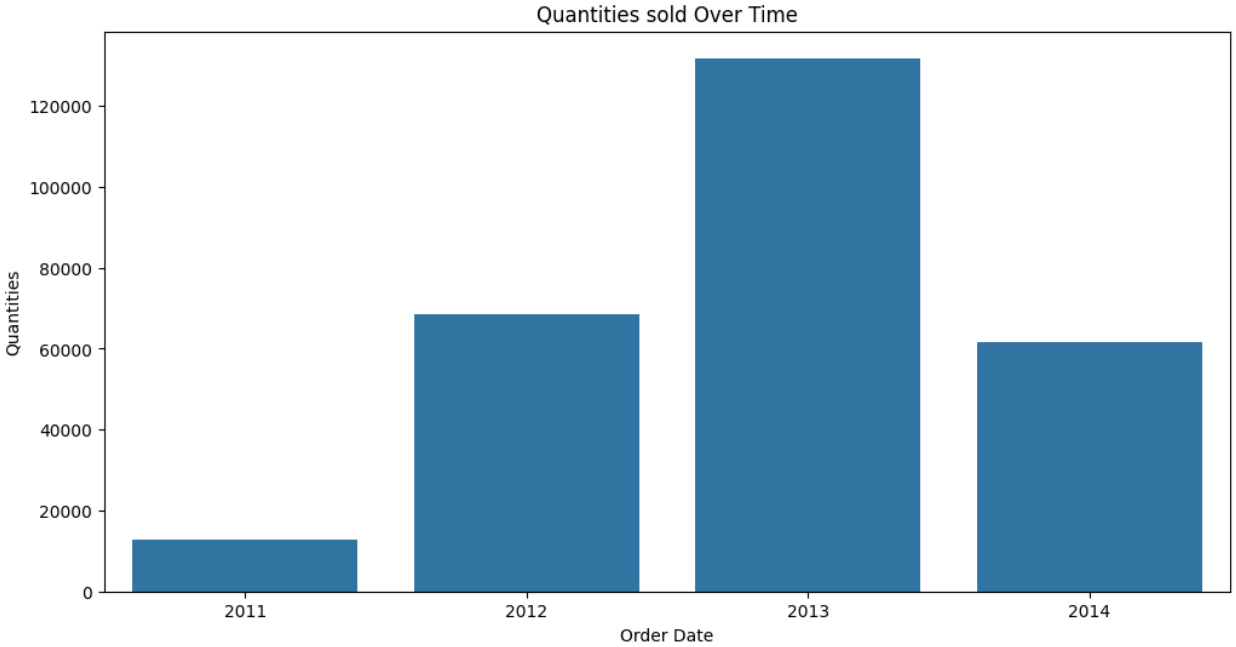
3.4.1. What are the Sales Trends over Time?

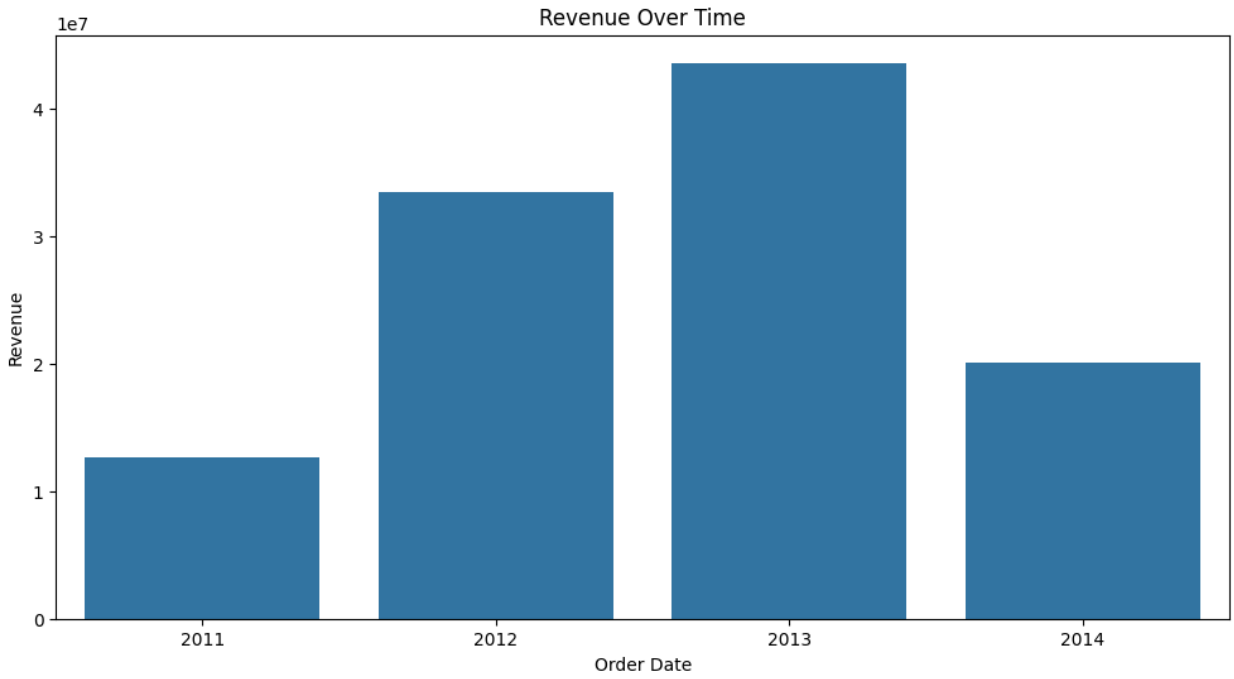



Revenue Over Time
1e6
Revenue
Order Date

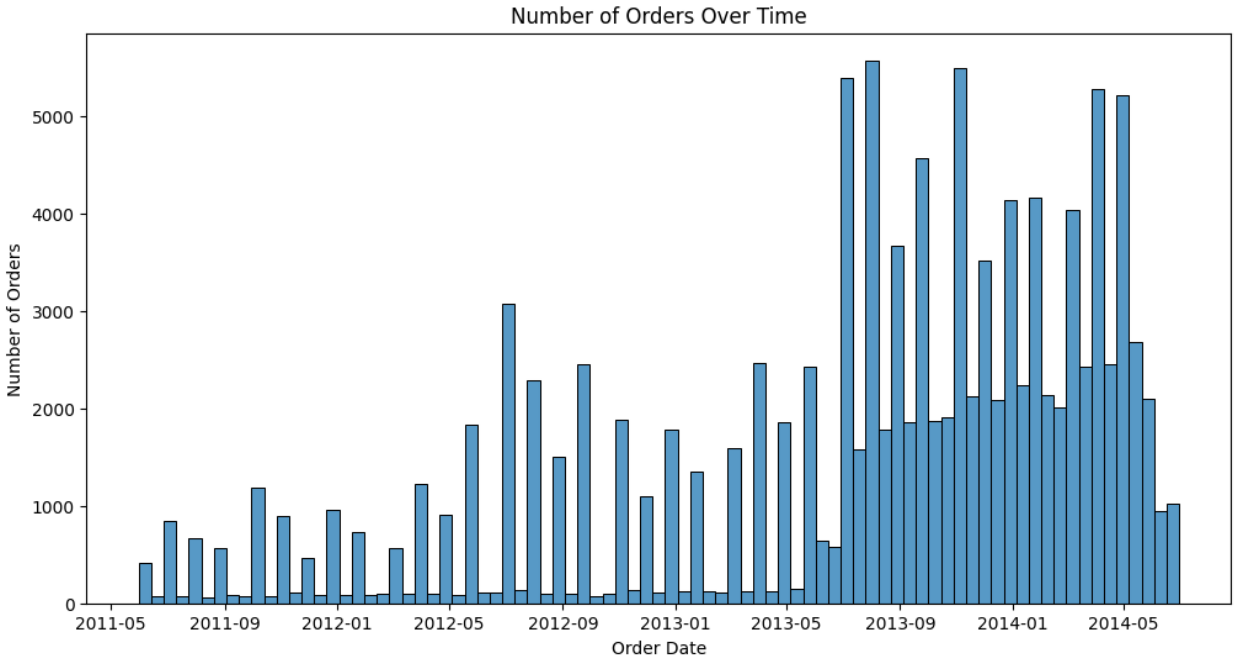

Quantities sold Over Time
Quantities
Order Date


Revenue Over Time
1e7
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Revenue
2011-Q2
2011-Q3
2011-Q4
2012-Q1
2012-Q2
2012-Q3
2012-Q4
2013-Q1
2013-Q2
2013-Q3
2013-Q4
2014-Q1
2014-Q2
Order Date

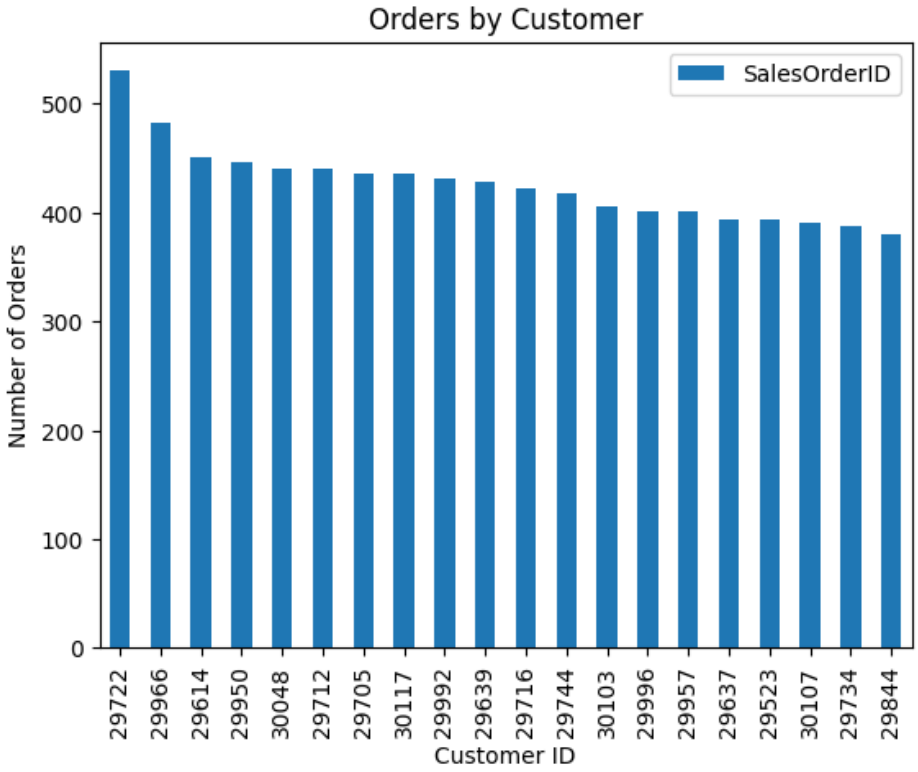

Quantities sold Over Time
120000
100000
80000
60000
40000
20000
0
Quantities
2011
2012
2013
2014
Order Date

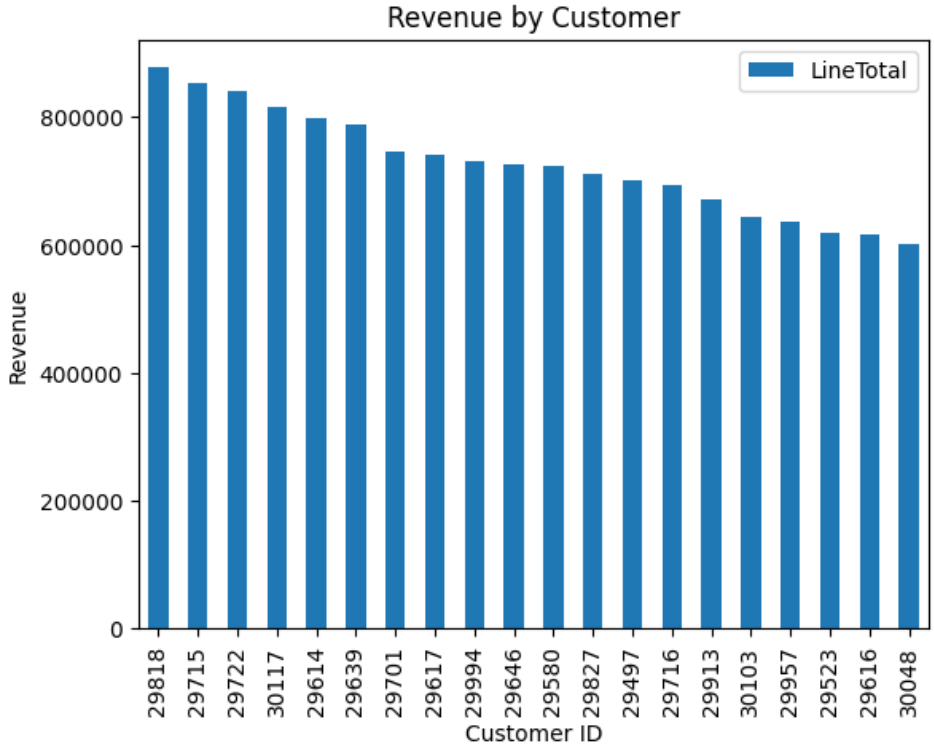


3.4.2. What are the number of order over time?

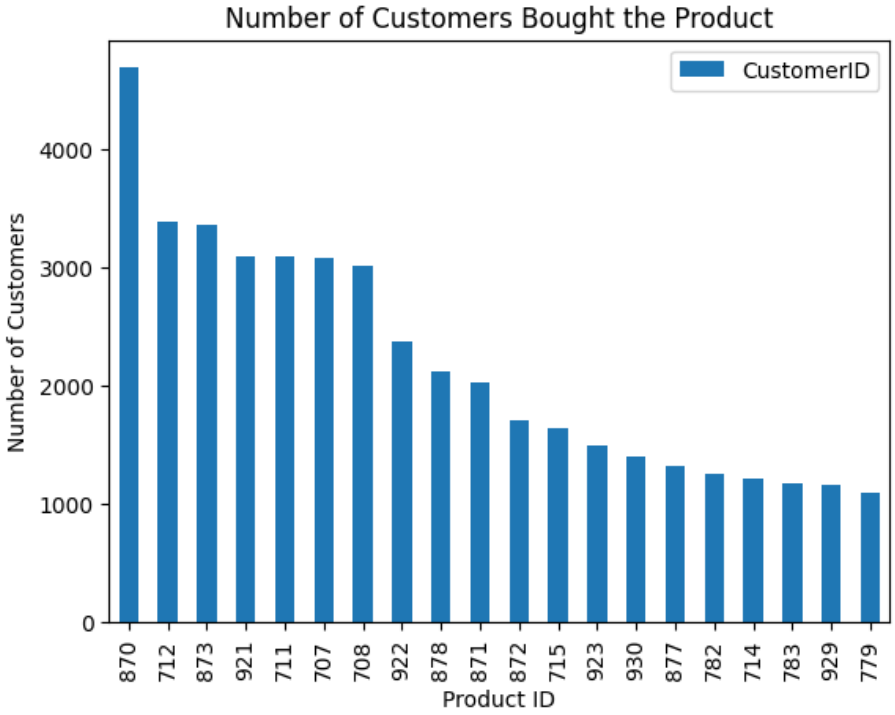


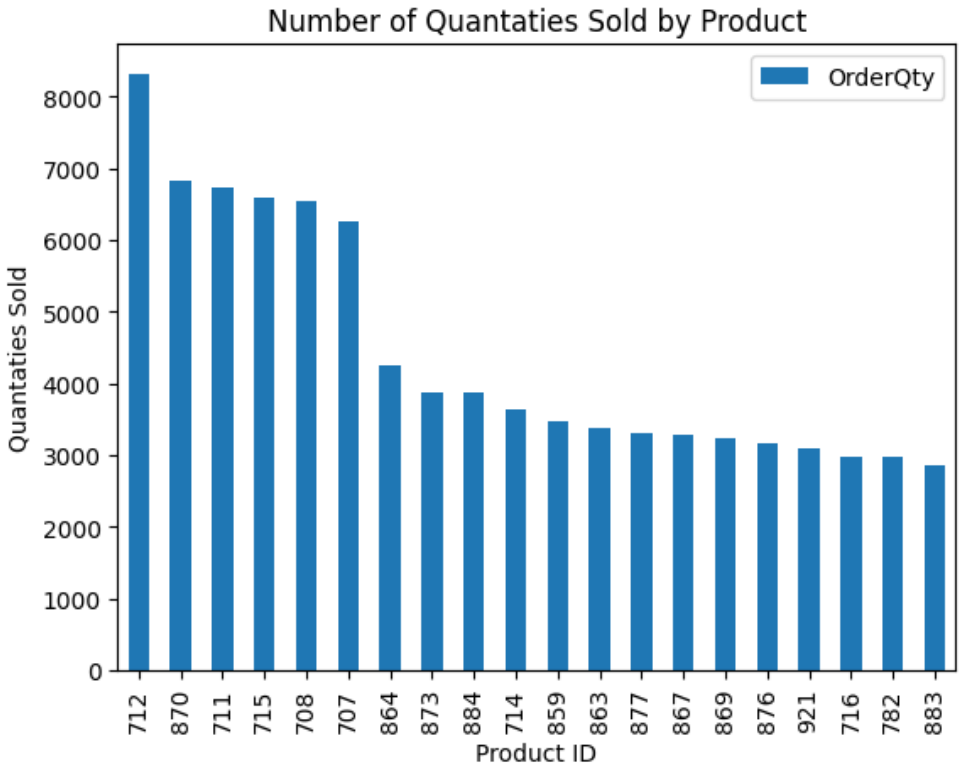
3.4.3. What are the customers purchase behaviors?

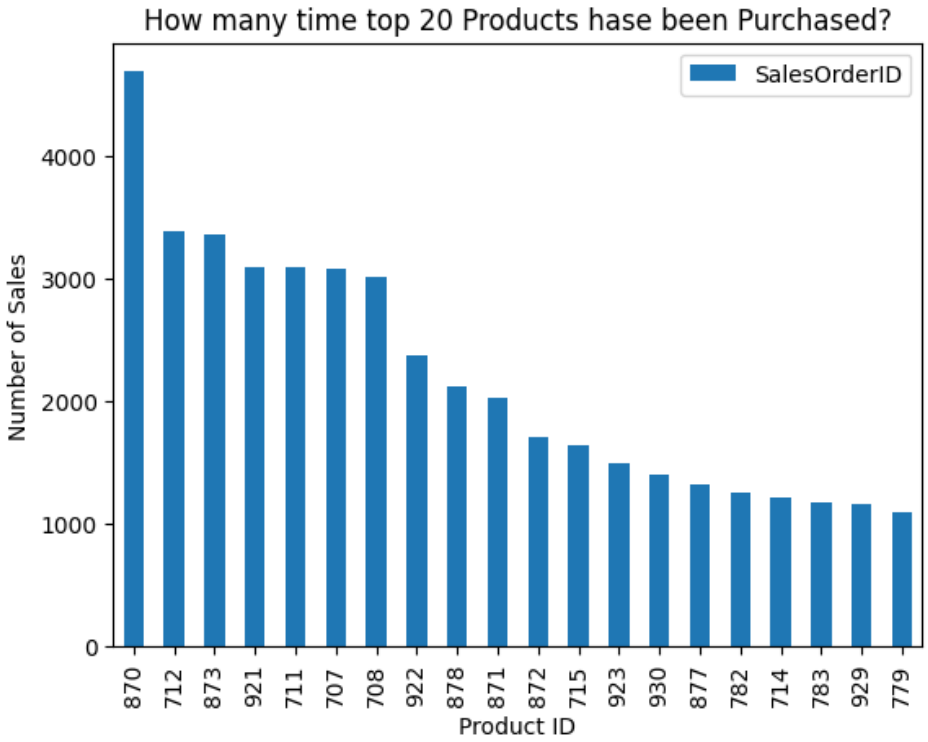


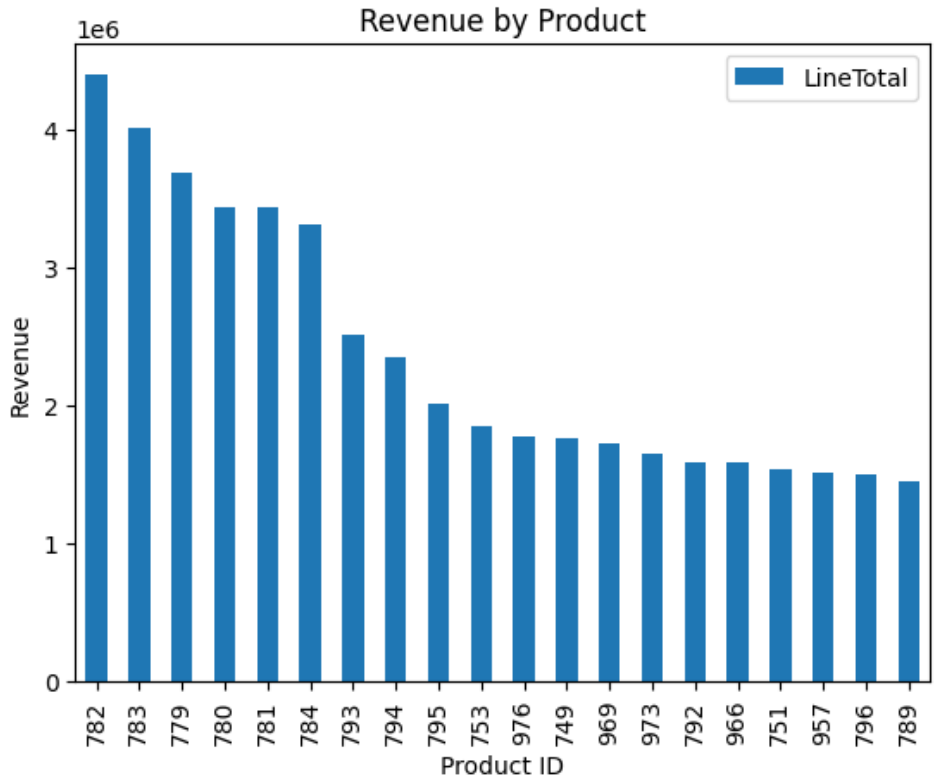


3.4.4. What are the popularity for the products?

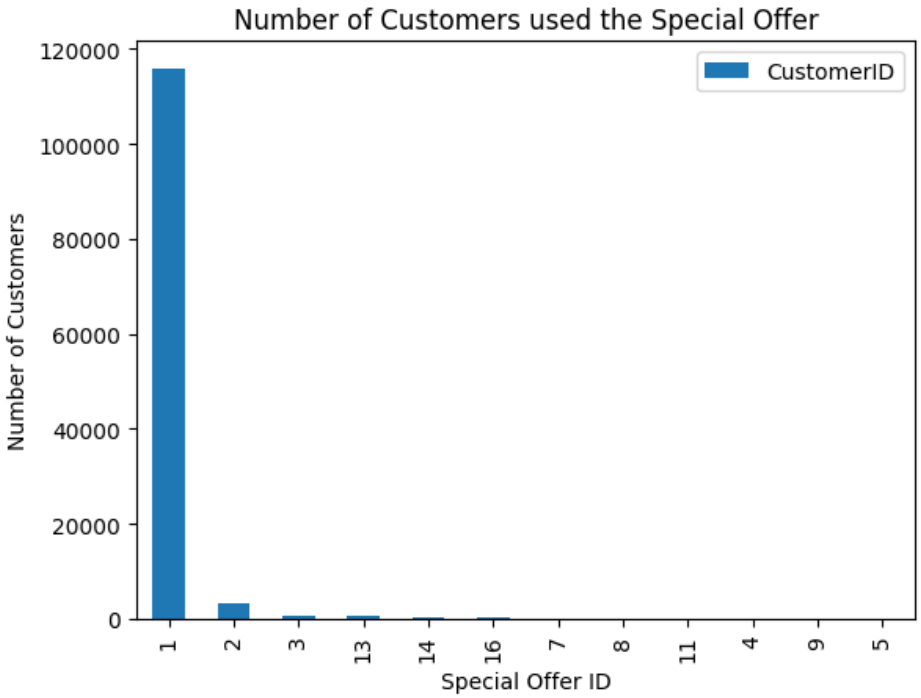


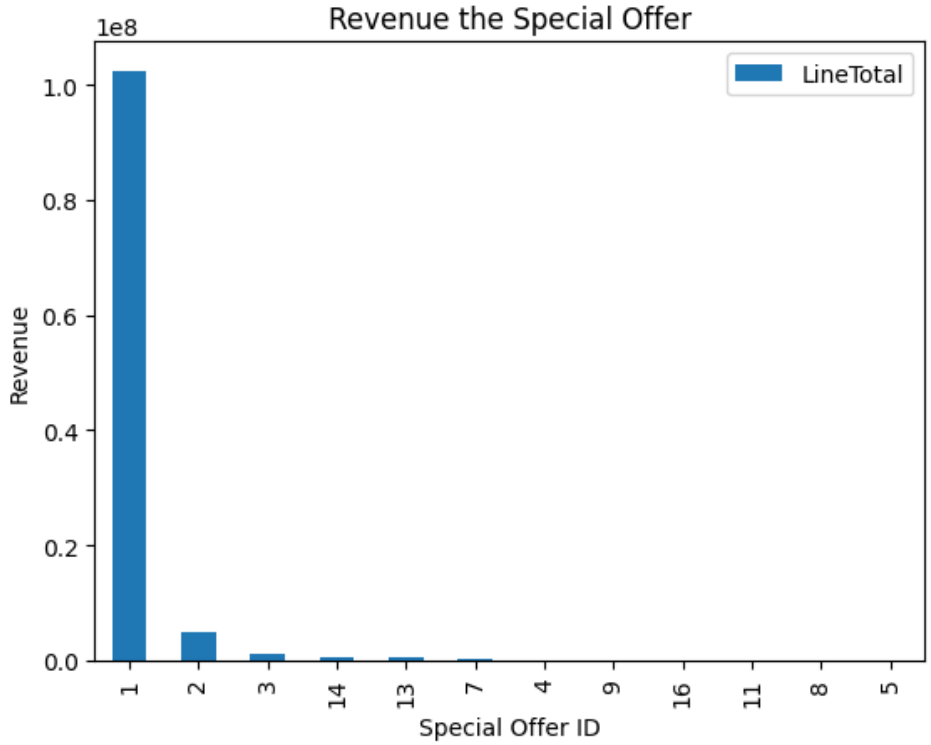



How many time top 20 Products hase been Purchased?
SalesOrderID
Number of Sales
0
1000
2000
3000
4000
870
712
873
921
711
707
708
922
878
871
872
715
923
930
877
782
714
783
929
779
Product ID


Revenue by Product
1e6
LineTotal
Revenue
0
1
2
3
4
782
783
779
780
781
784
793
794
795
753
976
749
969
973
792
966
751
957
796
789
Product ID

3.4.5. What are the most effective offers?





Step 4: Data Processing and Metrics Calculation:

4.1. Outliers Handling: Trimming

We will drop the observations greater than the upper whisker, or less than the lower whisker.

```
Number of records in the column CustomerID is (121317, 7)
Number of outlaiers in the column CustomerID is 0
Number of records in the column CustomerID after trimming is (121317, 7)

Number of records in the column SalesOrderID is (121317, 7)
Number of outlaiers in the column SalesOrderID is 0
Number of records in the column SalesOrderID after trimming is (121317, 7)

Number of records in the column ProductID is (121317, 7)
Number of outlaiers in the column ProductID is 0
Number of records in the column ProductID after trimming is (121317, 7)

Number of records in the column SpecialOfferID is (121317, 7)
Number of outlaiers in the column SpecialOfferID is 5433
Number of records in the column SpecialOfferID after trimming is (115884, 7)

Number of records in the column OrderDate is (115884, 7)
Number of outlaiers in the column OrderDate is 3983
Number of records in the column OrderDate after trimming is (111901, 7)

Number of records in the column OrderQty is (111901, 7)
Number of outlaiers in the column OrderQty is 4648
Number of records in the column OrderQty after trimming is (107253, 7)

Number of records in the column LineTotal is (107253, 7)
Number of outlaiers in the column LineTotal is 7691
Number of records in the column LineTotal after trimming is (99562, 7)
```

The shape of our data after trimming the outliers is 99,562 rows and 7 columns

4.2. Features Engineering

We engineered new feature to represent the days from the last order and the last observed date in our dataset for each customer. This features help us define the inactive customer and it can be used to indicate if the churned customers for further analysis.

	CustomerID	SalesOrderID	OrderDate	OrderQty	LineTotal	OrderedSinceDays
0	11000	51522	2013-06-20	1	21.9800	375
1	11000	57418	2013-10-03	5	2507.0300	270
2	11001	51493	2013-06-18	6	2419.9300	377
3	11001	72773	2014-05-12	4	588.9600	49
4	11002	51238	2013-06-02	1	2294.9900	393
...	...	...	...	...	...	...
28558	30118	50675	2013-04-30	78	14823.0076	426
28559	30118	53480	2013-07-31	105	20631.4680	334
28560	30118	58928	2013-10-30	86	13441.3620	243
28561	30118	65221	2014-01-29	61	10037.7180	152
28562	30118	71803	2014-05-01	98	11139.7260	60

28563 rows × 6 columns

4.3. Metrics Calculation

To calculate and then predict the CLV we need to transform our observation data to a set of metrics data represent a specific feature about the customer instead of dealing with non-informative data forms such as OrderDate, SalesOrderID, and OrderQty in their original format. We will calculate the following metrics.

- Frequency — the number of repeat purchases (more than 1 purchases)
- Recency — the time between the first and the last transaction
- T — the time between the first purchase and the end of the transaction period (last date of the time frame considered for the analysis)
- Monetary Value — it is the mean of a given customers sales value

	CustomerID	Frequency	Recency	T	MonetaryValue	Revenue
0	11000	1	105	375	2507.030000	2529.0100
1	11001	1	328	377	588.960000	3008.8900
2	11002	1	54	393	2419.060000	4714.0500
3	11003	1	125	388	2420.340000	4739.3000
4	11004	1	99	371	2419.060000	4796.0200
...	...	...	...	...	...	...
18741	30114	7	640	761	1339.262386	11652.9911
18742	30115	7	639	730	1175.541814	8917.5594
18743	30116	3	275	396	18417.776000	68579.7420
18744	30117	9	820	911	24191.248544	237366.2232
18745	30118	7	639	699	12582.259929	98766.1883

18746 rows × 6 columns

We calculated the metrics for our 18,746 customers.

Step 5: Data Modeling:

Next step is to use modeling methods to calculate or predict the CLV.

4.1. Data Preparation:

Before we process/train our data with models we need to prepare it.

We need to split our data in time-wise. We will spate the data of the last 6 months observed to be our target data which we will use to evaluate our predictive models.

```
trainning_df
```

	CustomerID	SalesOrderID	OrderDate	OrderQty	LineTotal	OrderedSinceDays
0	11000	51522	2013-06-20	1	21.9800	375
1	11000	57418	2013-10-03	5	2507.0300	270
2	11001	51493	2013-06-18	6	2419.9300	377
4	11002	51238	2013-06-02	1	2294.9900	393
5	11002	53237	2013-07-26	2	2419.0600	339
...	...	...	...	...	...	...
28556	30118	48318	2012-10-30	66	12458.7918	608
28557	30118	49499	2013-01-28	27	5543.7461	518
28558	30118	50675	2013-04-30	78	14823.0076	426
28559	30118	53480	2013-07-31	105	20631.4680	334
28560	30118	58928	2013-10-30	86	13441.3620	243

16880 rows × 6 columns

```
test_df
```

	CustomerID	SalesOrderID	OrderDate	OrderQty	LineTotal	OrderedSinceDays
3	11001	72773	2014-05-12	4	588.960	49
25	11012	68413	2014-03-17	2	6.280	105
27	11013	74908	2014-06-23	3	74.980	7
33	11017	68396	2014-03-16	1	742.350	106
35	11018	69087	2014-03-26	4	791.320	96
...	...	...	...	...	...	...
28540	30115	69469	2014-03-31	1	672.294	91
28544	30116	67279	2014-03-01	66	22114.950	121
28554	30117	69412	2014-03-31	94	19085.682	91
28561	30118	65221	2014-01-29	61	10037.718	152
28562	30118	71803	2014-05-01	98	11139.726	60

11683 rows × 6 columns

Then we will drop the uncommon observations.

```
test_df
```

	CustomerID	SalesOrderID	OrderDate	OrderQty	LineTotal	OrderedSinceDays
3	11001	72773	2014-05-12	4	588.960	49
25	11012	68413	2014-03-17	2	6.280	105
27	11013	74908	2014-06-23	3	74.980	7
33	11017	68396	2014-03-16	1	742.350	106
35	11018	69087	2014-03-26	4	791.320	96
...	...	...	...	...	...	...
28540	30115	69469	2014-03-31	1	672.294	91
28544	30116	67279	2014-03-01	66	22114.950	121
28554	30117	69412	2014-03-31	94	19085.682	91
28561	30118	65221	2014-01-29	61	10037.718	152
28562	30118	71803	2014-05-01	98	11139.726	60

4421 rows × 6 columns

```
trainning_df
```

	CustomerID	SalesOrderID	OrderDate	OrderQty	LineTotal	OrderedSinceDays
2	11001	51493	2013-06-18	6	2419.9300	377
24	11012	54508	2013-08-16	3	74.9800	318
26	11013	56137	2013-09-13	2	38.9800	290
32	11017	51256	2013-06-03	1	21.9800	392
34	11018	51492	2013-06-18	1	21.9800	377
...	...	...	...	...	...	...
28556	30118	48318	2012-10-30	66	12458.7918	608
28557	30118	49499	2013-01-28	27	5543.7461	518
28558	30118	50675	2013-04-30	78	14823.0076	426
28559	30118	53480	2013-07-31	105	20631.4680	334
28560	30118	58928	2013-10-30	86	13441.3620	243

6253 rows × 6 columns

Now we are ready to calculate the metrics for the customers eligible for modeling.

	CustomerID	Frequency	Recency	T	MonetaryValue	Revenue	Next_6_Months_Revenue	Adjusted_Revenue
0	11001	0	0.0	377	0.000000	2419.9300	588.960	206.1360
1	11012	0	0.0	318	0.000000	74.9800	6.280	2.1980
2	11013	0	0.0	290	0.000000	38.9800	74.980	26.2430
3	11017	0	0.0	392	0.000000	21.9800	742.350	259.8225
4	11018	0	0.0	377	0.000000	21.9800	791.320	276.9620
...	...	...	...	...	...	...	...	...
3549	30114	6	549.0	761	1467.610783	11083.8191	569.172	199.2102
3550	30115	6	549.0	730	1259.416450	8245.2654	672.294	235.3029
3551	30116	2	184.0	396	16569.189000	46464.7920	22114.950	7740.2325
3552	30117	8	730.0	911	24829.444362	218280.5412	19085.682	6679.9887
3553	30118	5	456.0	699	13379.675100	77588.7443	21177.444	7412.1054

3554 rows × 8 columns

We ended up with 3554 customers that we have enough data for modeling to predict and validate their CLV and model performance.

Method 1: [Benchmark] Historical CLV

We will use the simple historical method to predict the CLV as a benchmark to compare the models performance with.

To calculate the historical CLV and predict the CLV for the next 6 months based on historical CLV we will follow the following steps:

1. Calculate AVP = Total Revenue/ Number of Purchases
2. Calculate APFR = Number of Purchases/ Number of Unique Periods
3. Calculate CV = APV * APFR
4. Calculate ACL = Last Purchase - First Purchase (in years)
5. Calculate CLV = CV * ACL
6. Calculate Adjusted_CLV = CLV* Average gross margin
7. Calculate Predectied_CLV for next 6 months = CLV * (0.5 / Period in years)
8. Calculate Adjusted_Predicted_CLV = Predectied_CLV* Average gross margin

	CustomerID	Frequency	Recency	T	MonetaryValue	Revenue	Next_6_Months_Revenue	Adjusted_Revenue	APV	APFR	CV	ACL	CLV	Adjusted_CLV	Predicted_CLV	Adjucted_Predicted_CLV
0	11001	0	0.0	377	0.000000	2419.9300	588.960	206.1360	2419.930000	0.450062	1089.117694	0.000000	0.000000	0.000000	0.000000	0.000000
1	11012	0	0.0	318	0.000000	74.9800	6.280	2.1980	74.980000	0.450062	33.745623	0.000000	0.000000	0.000000	0.000000	0.000000
2	11013	0	0.0	290	0.000000	38.9800	74.980	26.2430	38.980000	0.450062	17.543403	0.000000	0.000000	0.000000	0.000000	0.000000
3	11017	0	0.0	392	0.000000	21.9800	742.350	259.8225	21.980000	0.450062	9.892355	0.000000	0.000000	0.000000	0.000000	0.000000
4	11018	0	0.0	377	0.000000	21.9800	791.320	276.9620	21.980000	0.450062	9.892355	0.000000	0.000000	0.000000	0.000000	0.000000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3549	30114	6	549.0	761	1467.610783	11083.8191	569.172	199.2102	1583.402729	3.150432	4988.401938	1.504110	7503.103189	2626.086116	1688.429509	590.950328
3550	30115	6	549.0	730	1259.416450	8245.2654	672.294	235.3029	1177.895057	3.150432	3710.877769	1.504110	5581.566837	1953.548393	1256.024596	439.608609
3551	30116	2	184.0	396	16569.189000	46464.7920	22114.950	7740.2325	15488.264000	1.350185	20912.021060	0.504110	10541.950343	3689.682620	2372.263795	830.292328
3552	30117	8	730.0	911	24829.444362	218280.5412	19085.682	6679.9887	24253.393467	4.050555	98239.701033	2.000000	196479.402067	68767.790723	44213.922167	15474.872758
3553	30118	5	456.0	699	13379.675100	77588.7443	21177.444	7412.1054	12931.457383	2.700370	34919.718458	1.249315	43625.730457	15269.005660	9817.134166	3435.996958

3554 rows × 16 columns

To measure the accuracy of the method results we will compare the predicted CLV to the known CLV for the last 6 months for each customer.

```
Root Mean Squared Error: 4406.50
R-squared: 0.15
```

The RMSE for the benchmark historical method is 4406.50 from the actual value.

Method 2: Probabilistic BG/NBD

The BG/NBD is a probabilistic model uses the customer buying behavior and customer churn rate to predict the number of next purchases the customer will place in specific

amount of time, the customer churn state (if the customer will stay alive customer), and the CLV.

The model uses the combination of Poisson and Gamma distribution which is called negative binomial distribution (NBD) to model the customer buying behavior.

It uses Beta distribution to predict the churn probability and the CLV of the customer.

To perform the model, we will start with predicting the number of purchases for each customer in the next 6 months.

	CustomerID	Frequency	Recency	T	MonetaryValue	Revenue	Next_6_Months_Revenue	Adjusted_Revenue	Predicted_Purchases
145	11331	19	168.0	355	46.242105	892.58	429.86	150.4510	7.089120
132	11287	15	181.0	364	38.070667	667.13	499.94	174.9790	5.570504
144	11330	14	159.0	350	45.107143	646.48	437.59	153.1565	5.367343
87	11200	14	171.0	355	42.028571	782.29	681.09	238.3815	5.314424
74	11176	14	176.0	360	41.008571	645.70	617.45	216.1075	5.262538

Then we will predict the churn probability where 1 indicate the customer won't churn in the specified period.

	CustomerID	Frequency	Recency	T	MonetaryValue	Revenue	Next_6_Months_Revenue	Adjusted_Revenue	Predicted_Purchases	Predicted_Alive
3518	30068	1	182.0	973	419.4589	603.3971	37.254	13.0389	0.315607	1.0
912	13668	1	4.0	356	27.2800	106.2600	7.950	2.7825	0.698835	1.0
825	13318	1	5.0	350	68.9700	116.9100	97.350	34.0725	0.707186	1.0
363	11965	1	9.0	352	27.2800	94.2500	24.490	8.5715	0.704380	1.0
1957	16771	1	6.0	339	21.9800	54.5800	49.970	17.4895	0.723025	1.0

Now, we will utilize the model to predict the CLV.

	CustomerID	Frequency	Recency	T	MonetaryValue	Revenue	Next_6_Months_Revenue	Adjusted_Revenue	Predicted_Purchases	Predicted_Alive	Predicted_CLV
3221	29639	8	730.0	973	26909.275000	240127.6550	36938.352	12928.4232	1.435478	1.0	40023.69
3154	29546	5	456.0	699	26221.993600	164886.2331	30215.394	10575.3879	1.263144	1.0	35079.86
3109	29486	8	730.0	973	22513.267950	207477.8272	35005.728	12252.0048	1.435478	1.0	33485.34
3547	30112	6	549.0	730	21678.521150	171008.2769	15587.298	5455.5543	1.422806	1.0	32348.86
3201	29614	8	730.0	942	20150.633187	178935.7383	19640.214	6874.0749	1.476150	1.0	30820.48

We notice a NaN results for some customers from the model, this could happen due to one or more of the following reasons:

- If Recency and T are very close.
- If the monetary values are too similar.

- If there is only one transaction (Frequency = 1), the model may not be able to generate a meaningful prediction, leading to NaN values.

Finally, we will evaluate our model.

```
Root Mean Squared Error: 4464.26
R-squared: 0.13
```

The BG/NBG model predict the CLV with RMSE of 4464.26 which is higher than the benchmark model meaning either the data is not sufficient for the model or our implementation for the model is not perfect.

Method 3: Machine Learning Regression

Next, we will use a supervised machine learning regression technique to predict the CLV for the next 6 months using XGBoost.

Stating with standardizing our data to avoid bias.

	Frequency	Recency	T	MonetaryValue	Adjusted_Revenue
0	-0.400332	-0.454107	-0.506097	-0.305053	-0.353697
1	-0.400332	-0.454107	-0.798358	-0.305053	-0.475445
2	-0.400332	-0.454107	-0.937058	-0.305053	-0.461090
3	-0.400332	-0.454107	-0.431794	-0.305053	-0.321647
4	-0.400332	-0.454107	-0.506097	-0.305053	-0.311415

Next, we will train our model

```
pred_xgb[0:10]
```

```
array([ 209.39305 , 1744.771   ,   54.27122 , 2218.6467  , 2443.324   ,
        2443.9543  ,   62.500767,   32.487354,   27.51617 ,   27.602818],
      dtype=float32)
```

Then validating our model

```
XGBoost Root Mean Squared Error: 294.28
XGBoost R-squared: 0.15
```

The XGBoost model predicted the CLV with RMSE of 294.28 which represent a high accuracy for the model.

Using this trained model we ca predict the CLV for the next 6 months with expected error margin of 294.28 revenue unit.

Method 4: Deep Learning Linear

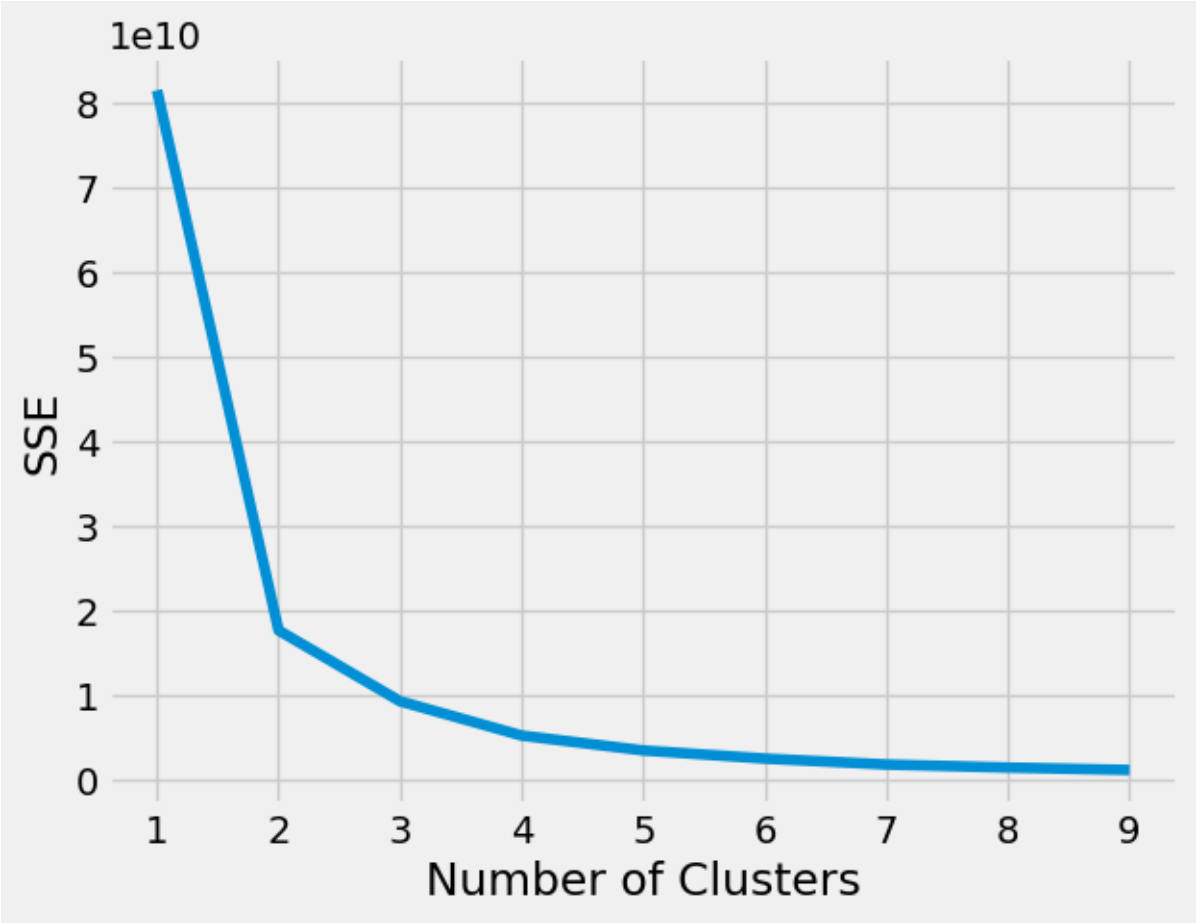
Next, we will model an artificial neural network (ANN) to predict the CLV for the next 6 months using deep learning.

```
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 21.5324 - mean_squared_error: 21.5324
[28.418746948242188, 28.418746948242188]
Loss =  28.418746948242188
Deep Learning RMSE =  5.33
```

The RMSE for the deep learning linear model is 5.33 that beats the XGBoost model and rank as the best predictive model for CLV.

Method 5: Machine Learning Segmentation and Classification

In this step we will try to classify and predict the customer Life Value classes (High LV, Low LV) using both segmentation with KMeans and classification with XGBoost.



The elbow method indicate there are 2 classes for our revenue (LV) column.

Revenue_Class	count
0	3383
1	171

dtype: int64

Most of our customers are on the Low LV class

```
ML_df
```

	CustomerID	Frequency	Recency	T	MonetaryValue	Next_6_Months_Revenue	Adjusted_Revenue	Revenue_Class
0	11001	0	0.0	377	0.000000	588.960	206.1360	0
1	11012	0	0.0	318	0.000000	6.280	2.1980	0
2	11013	0	0.0	290	0.000000	74.980	26.2430	0
3	11017	0	0.0	392	0.000000	742.350	259.8225	0
4	11018	0	0.0	377	0.000000	791.320	276.9620	0
...	...	...	...	...	...	...	...	...
3549	30114	6	549.0	761	1467.610783	569.172	199.2102	0
3550	30115	6	549.0	730	1259.416450	672.294	235.3029	0
3551	30116	2	184.0	396	16569.189000	22114.950	7740.2325	1
3552	30117	8	730.0	911	24829.444362	19085.682	6679.9887	1
3553	30118	5	456.0	699	13379.675100	21177.444	7412.1054	1

3554 rows × 8 columns

Standardization to avoid bias.

	CustomerID	Frequency	Recency	T	MonetaryValue
0	-1.221531	-0.400332	-0.454107	-0.506097	-0.305053
1	-1.219724	-0.400332	-0.454107	-0.798358	-0.305053
2	-1.219560	-0.400332	-0.454107	-0.937058	-0.305053
3	-1.218903	-0.400332	-0.454107	-0.431794	-0.305053
4	-1.218738	-0.400332	-0.454107	-0.506097	-0.305053

```
Accuracy of XGB classifier on training set: 99.78%
Accuracy of XGB classifier on test set: 98.03%
```

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       334
           1       0.89      0.77      0.83        22

    accuracy                           0.98       356
   macro avg       0.94      0.88      0.91       356
weighted avg       0.98      0.98      0.98       356
```

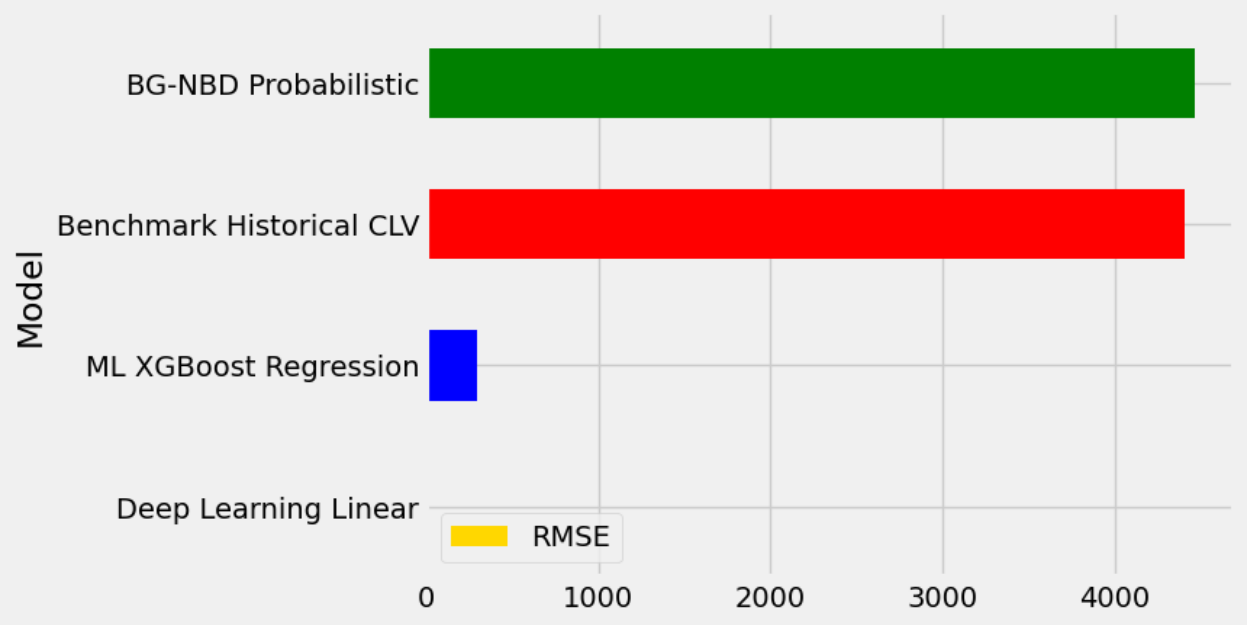
```
from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_test, y_pred)*100
print(f"Model Accuracy: {accuracy:.2f}%")
```

```
Model Accuracy: 98.03%
```

Step 6: Models Evaluations

Summary and rank of the models performance:

	Model	RMSE
3	Deep Learning Linear	5.33
2	ML XGBoost Regression	294.28
0	Benchmark Historical CLV	4406.50
1	BG-NBD Probabilistic	4464.26



The very low error for the linear deep learning model may indicate an overfitting, this can be validated/solved by having more data points.

The ML model error indicates a balanced model this also can be enhanced by providing more data to the model.

The BG-NBD errors indicate either the data is not sufficient for the model makes it under fitted or our implementation of the model is not perfect. Some potential errors with the data:

- If Recency and T are very close.
- If the monetary values are too similar.
- If there is only one transaction (Frequency = 1), the model may not be able to generate a meaningful prediction, leading to NaN values.

The benchmark error represents an average performance for a simple historical model.

The classification model is fitting very well for the dominant class (Low LV) and doing good for the non-rich data class (High LV) this resulted in loss in recall that can be increased with more data points for the class (High LV).

Results

We have successfully predicted the CLV for 3554 customers with 5 different methods, ranked by the RMSE as follows Liner Deep Learning 5.33 RMSE (a potential over fitted model), ML XGBoost Regression 294.28 RMSE (balanced model), Benchmark Historical CLV 4406.50 RMSE (simple model), and the

BG-NBD Probabilistic 4464.26 RMSE (under fitted or wrong implementation), and The classification model is fitting very well for both classes with classification accuracy of 98%.

Through the EDA we found discovered that a single special offer is used 93% of the times the customers used a special offer. The sales trended to increase from the years 1-3 and dropped in the 4th years that may be resulting of only the first 6 months of the 4th year was recorded. The top customer by orders is different than the top customer by revenue with the first had over 500 orders and the second contributed over 800,000 revenue. We identified the most popular product have been ordered over 4000 times and the top selling product sold over 8000 quantities and the most revenue-generating product generated 4 million unit price.

To achieve this, we queried our data from database server with SQL resulting in acquiring 121,317 data records. We performed data pre-processing using Pandas resulting in cleaning the data from missing values, cast the data into the right format, ensured data integrity, and detect outliers. We analyzed the data by visualizing its distribution, and correlation resulting on define skewed features and highly correlated features. We summarized the data profile using Pandas profiling. We performed extensive data preparation, outliers trimming, data processing, modeling and evaluation.

Conclusions and Future Work

Predicting the CLV is valuable to the business due its association with each customer behavior and value contribution directly. By predicting the CLV correctly it opens the door to many advanced application and use cases to the predicted metrics. One of the potential future works is to build an automated pipeline to predict the CLV for the customers periodically and automatically, some of the technology stack that could be used is Airflow, Docker, and AutoML, the future application would increase the models performance by retraining them on larger and recent data, this app could utilize the distributed computing power of apache Hadoop for batch processing.

In other hand this project can be integrated to many CRM systems and business managements systems to add the predicted CLV into account for more advanced business decision.

References

- Scikit-learn Machine Learning in Python documentation: https://scikit-learn.org/stable/
- Pandas documentation: https://pandas.pydata.org/docs/
- Pandas profiling: https://pypi.org/project/pandas-profiling/
- Customer Life Time Value Prediction: https://www.analyticsvidhya.com/blog/2020/10/a-definitive-guide-for-predicting-customer-lifetime-value-clv/