XYZ Retail, Inc

Retail Data Optimization

Technical Design Document

Case Study for Retail Data Optimization of XYZ, Inc for efficient data management and analysis

NEOSTATS

Versioning of the document

S.N	Date	Version	Owner
1	2024-08-17	Ver 1	Nidhi Chaurasia
2	2024-08-18	Ver 2	Nidhi Chaurasia

Table of Contents

Overview on XYZ Retail Inc

XYZ Retail Inc is a multinational electronic devices retail store having online (Web and mobile app) and physical presence in more than 10 countries. Their operation mainly consists of sales of electronic devices such as Smartphones, tablets and laptops. They have huge customer base in the countries they operate. All the sales data is stored in different source systems according to the store size. Their sales data typically consists of product, quantity of product sold and detail information about customers. They also store credit card information of customers in order to manage the payment information.

Problem Statement of case study:

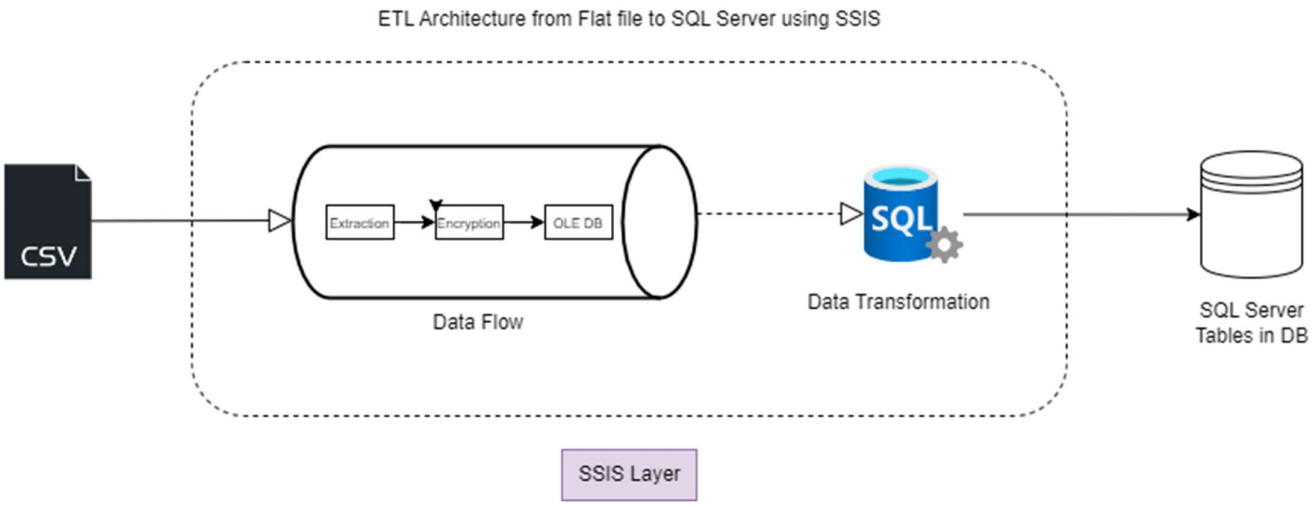
XYZ Retail Inc., a prominent player in the retail industry, faces the challenge of efficiently managing and analyzing its sales data. Since they operate through different channel, their source systems are classified into different types which makes it difficult to manage, clean and analyze the data coming from those channels. The company operates through multiple channels—online platforms, physical stores, and mobile apps. To enhance decision-making in different levels, XYZ Retail aims to create an end-to-end data pipeline that ingests, processes, and visualizes sales data.

Solution Overview:

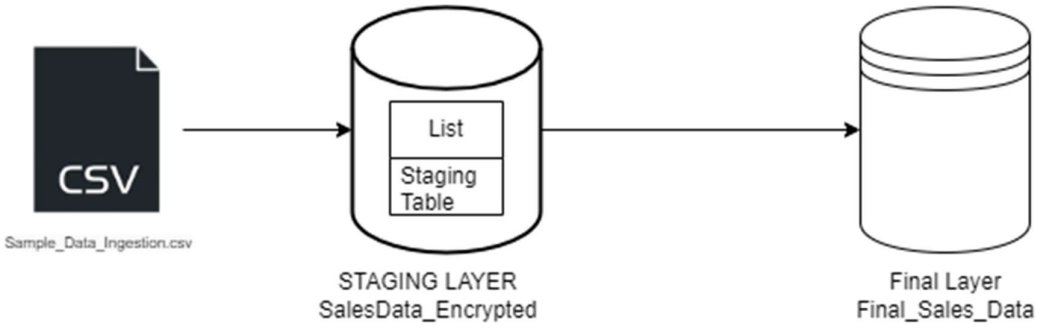
Since the data source provided to us is in form of flat file which is a structured data, and assuming Neostats would be receiving the data in batch processing, we will be using RDBMS and the concept of ETL and ELT process to efficiently cleanse and load the data into the data warehouse. Once the cleaner data is loaded to data warehouse using SSIS, SQL Server and Azure SQL database, we will be connecting the database with PowerBI to properly analyse the sales performance to gather deeper insights into their business and help XYZ Retail to make their decision systems efficient.

Below is the detailed solution architecture along with the SSIS Data pipeline explanation. In the Architecture, we used SQL Server Integration services (SSIS) since our sales data is a structured data and SSIS provides a robust way of ingesting structured flat files to RDBMS which can be cost efficient, easy to setup and efficient for further reporting and analysis.

ETL Architecture



Data Flow Architecture from File to SQL Server Database:



Flat file to Database Tables Architecture and data flow

Sales Data Flat file Description:

The csv flat file given to us is a structured file, having rows and columns. The columns are separated using a delimiter –"," (comma) and row delimiter as {CR}{LF}. Below are the detailed property of the columns, and it's data types:

Column_name	Type	Length	Prec	Nullable
OrderID	varchar	100		NO
CustomerName	varchar	100		yes
PhoneNumber	varchar	20		yes
Location	varchar	100		yes
Country	varchar	50		yes
StoreCode	varchar	50		yes
Product	varchar	100		yes
Quantity	float	8	53	yes
Price	float	8	53	yes
Date	datetime	8		yes
EncryptedCreditCardNumber	varbinary	256		yes
ExpiryDate	Datetime	10		yes

Steps performed for ETL/ELT:

Sales Data Ingestion

The primary objective is to optimize retail operations by leveraging data-driven insights. Here are the key components of the problem statement:

A) Data Ingestion:

- Ingest sales data from a CSV file into Azure SQL Database.

Sample_Data_Ingestion.csv

- Ensure data quality and handle any data anomalies (data duplicates, data type anomalies, data governance and data security) during ingestion.

Procedures:

1. **Extract/Source Selection**: Data Extraction is the first step of ETL Process. Since the sales data is in flat file format (csv), we used traditional data warehouse concept to extract data using SSIS Data Flow task. The **decision for choosing SSIS** over any other ETL tool like Azure Data factory, Airflow, etc for end-to-end ETL processes can be advantageous for the following reasons:

   a) **Structured Data Management**: Ideal for managing structured, relational data with defined schemas and relationships. The csv format is a structured data, hence RDBMS is the best choice to load the data, which can be achieved using SSIS.

b) **Advanced Querying**: RDBMS and SSIS Supports complex transformations like data cleaning, security or governance and queries using SQL Server, enhancing data manipulation capabilities without complex architecture. Data transformations can be placed even before loading the data to data warehouse.

c) **Performance Optimization**: SSIS supports the extraction of different sources of files namely csv, txt, excel, xml, etc without complex architecture and also efficiently managing memory. The data extraction can also be processed parallelly which optimizes the performance significantly.

d) **Security and Compliance**: Offers robust security features Eg: Encryption services, key and certificate generation and compliance with industry standards, crucial for sensitive data, even before loading the data to data warewhouse which gives ability to secure data at runtime.

e) **Integration with RDBMS**: SSIS easily integrated with RDBMS (Azure SQL Database or SQL Server Management Studio) giving opportunity to load, analyse data in tabular format.

f) **Cost Efficiency**: SSIS, being an open source tool and it's community edition being free for use makes it cost efficient. Since the data received is a structured flat file, we can skip the use of cloud data lake and use traditional data warehouse technologies to reduce the cost in the project. Microsoft built SSIS boasts huge open source community, which makes it popular and easy to handle or solve errors, if any.

The tools/IDE installed and configured with the local host as server to initiate the procedure are listed below

Tools:

- SQL Server 2022
- Azure Data Studio
- Visual Studio 2022
- SQL Server Integration Services

B) Data Transformation:

a) Data Security: XYZ Retail Inc. sends credit card details as a sensitive data. During Data Ingestion, It is very important to handle these sensitive information. We have handled it using Symmetric Encryption Key and Certificate function in SQL Server while loading the data to database. We

applied OLE DB Command Destination Task in SSIS to achieve this functionality ensuring robust data security mechanism without exposing sensitive data.

We have considered the column credit card number as sensitive information, so we are using the method to Encrypt this particular column data. To perform encryption technique, We need to follow the steps listed below:

- Create a master key and secure it with a password as 'password'.
- After creating master key, we have created a certificate as CreditCardCert with a subject which in our case is 'Credit Card Encryption Certificate'.
- The third step is to create a Symmetric Key and encrypted it wit the certificate CreditCardCert.
- The updated encrypted table (dbo.SalesData_Encrypted) is created as a last step and the column CreditCard Number is updated by VARBINARY datatype values in the renamed column EncrytedCreditCardNumber.

```
--step 1: create master key encryption using master key function
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'Password';

CREATE CERTIFICATE CreditCardCert
WITH SUBJECT = 'Credit Card Encryption Certificate';

CREATE SYMMETRIC KEY CreditCardKey
WITH ALGORITHM = AES_256
ENCRYPTION BY CERTIFICATE CreditCardCert;

--step2: create table with encryption
CREATE TABLE dbo.SalesData_Encrypted (
  --col1
  --col2...
   EncryptedCreditCardNumber VARBINARY(256),

);
```

b) Data Quality Check: To ensure data quality, We performed Null and Datatype checks on columns Quantity, Price, OrderID, CustomerName, Country, Location. If any Null value found in table, We updated validflag(additional field) to -1. This data cleaning process validates the column, its data maintaining data quality

```
-- Null check/ Data Quality Check
UPDATE STG SET ValidFlag = -1
from [dbo].[SalesData_Encrypted] AS STG where OrderID IS NULL OR PhoneNumber IS NULL
```

```
UPDATE STG SET ValidFlag = -1
from [dbo].[SalesData_Encrypted] AS STG where CustomerName IS NULL OR StoreCode IS NULL

UPDATE STG SET ValidFlag = -1
from [dbo].[SalesData_Encrypted] AS STG where [Location] IS NULL OR [Country] IS NULL

UPDATE STG SET ValidFlag = -1
from [dbo].[SalesData_Encrypted] AS STG where Product IS NULL

update  STG SET Quantity = NULL
from [dbo].[SalesData_Encrypted] AS STG where Quantity = 0

update  STG SET Price = NULL
from [dbo].[SalesData_Encrypted] AS STG where Price = 0

update STG set ValidFlag = -1 from [dbo].[SalesData_Encrypted] as stg
where [Quantity] is not null and [Price] is null

update STG set ValidFlag = -1 from [dbo].[SalesData_Encrypted] as stg
where [Price] is not null and [Quantity] is null

update STG set ValidFlag = -1 from [dbo].[SalesData_Encrypted] as stg
where [Quantity] is  null and [Price] is null
```

Above scripts are used for data quality checks in our transformation step. We are checking whether key column like OrderID and other fields which are important for analysis are NULL. If those columns are NULL, the code modifies validFlag value to -1. Similarly, we are also performing column data type checks in order to maintain data quality.

```
-- data type check

UPDATE STG SET ValidFlag = -1 from [dbo].[SalesData_Encrypted] AS STG WHERE ISNUMERIC(Price)
= 0

UPDATE STG SET ValidFlag = -1 from [dbo].[SalesData_Encrypted] AS STG WHERE ISDATE(Date) = 0
```

c) **Data Governance/Integrity:** In sales data provided to us, following data anomalies/discrepancies were observed:

- **Country → Location mismatch:** To solve this mismatch, We assumed that we have a master Look-up Table having valid country → Location data. We used this lookup table to check valid and update valid country according to the locations provided in the dataset. This approach made sure that we avoided any hard coding in our implementation.

TableName	Column_name	Type	Length
country_lookup	Country	varchar	100
country_lookup	Location	varchar	100

Table 1:Lookup Table Format

In raw data, there are incorrect Country → Location pair. Example given below:

Results Messages

	Country	Location	OrderID
10	Australia	New York	JZSJXC
11	Australia	New York	LSMOVN
12	Australia	New York	QFEQWC
13	Canada	Chicago	BVRTFM
14	Canada	Chicago	PILEFL
15	Canada	Houston	WWVNGW
16	Canada	Los Angeles	CRQPDQ
17	Canada	Los Angeles	MXLPTM
18	Canada	New York	COHLOL
19	India	Mumbai	DRTYF
20	UK	Chicago	KCFBLY

According to the raw data example above, we used the query provided below to correct this data anomaly. The below code is used to update the country wherever we have incorrect country information in raw data. Lookup table country_lookup is used to JOIN with the dataset on Location and then correct is updated based on correct data provided in the lookup table.

```
update STG set STG.Country = LKP.Country
from [dbo].[SalesData_Encrypted] STG
JOIN [dbo].[country_lookup] LKP
ON STG.[Location] = LKP.[Location]
```

- **Invalid Country code in Phone Number:** There are incorrect country codes in the phone numbers provided to us in the raw dataset. To solve this invalid data, we updated the country code in each mobile numbers of customers using the values provided in country field.

  Example of Raw data:

Results | Messages

	Country	PhoneNumber
21	Australia	+1 919-905-3000
22	Australia	+1 931-426-6301
23	Canada	+1 222-849-4865
24	Canada	+1 289-951-6678
25	Canada	+1 453-576-7373
26	Canada	+1 482-131-6898
27	Canada	+1 637-159-1129
28	Canada	+1 650-277-1089
29	India	+1 657-683-1988
30	India	+1 657-683-1989
31	India	+1 657-683-1990

To solve the above data anomaly, we will be using SUBSTRING function in the SQL, to calculate the index of country code. We, then will update the country code according to the enriched country data from the dataset.

Below is the SQL Code for solving this data anomaly. We added this SQL Code in our stored procedure as well as part of data transformation step in SSIS.

```
--Phone Number Enrichment

UPDATE STG
SET PhoneNumber = case
                      when country = 'USA' THEN
                          concat('+1', SUBSTRING(PhoneNumber,
LEN('+1 ') + 1, LEN(PhoneNumber) - LEN('+1 ')))
                      WHEN Country = 'India' THEN
                          concat('+91', SUBSTRING(PhoneNumber,
LEN('+1 ') + 1, LEN(PhoneNumber) - LEN('+1 ')))
                      WHEN Country = 'UK' THEN
                          concat('+44', SUBSTRING(PhoneNumber,
LEN('+1 ') + 1, LEN(PhoneNumber) - LEN('+1 ')))
                      WHEN Country = 'Australia' THEN
                          concat('+61', SUBSTRING(PhoneNumber,
LEN('+1 ') + 1, LEN(PhoneNumber) - LEN('+1 ')))
                      WHEN Country = 'UK' THEN
                          concat('+44', SUBSTRING(PhoneNumber,
LEN('+1 ') + 1, LEN(PhoneNumber) - LEN('+1 ')))
                      WHEN Country = 'Canada' THEN
                          concat('+1', SUBSTRING(PhoneNumber,
LEN('+1 ') + 1, LEN(PhoneNumber) - LEN('+1 ')))
                    END
from [dbo].[SalesData_Encrypted] STG
```

- **Invalid datatype in CustomerName eg: Test User1001:** To solve this invalid data, We extracted only string values from the CustomerName field using SUBSTRING Function in SQL.

```
--CustomerName enrichment

UPDATE STG Set CustomerName = LEFT(CustomerName, PATINDEX('%[0-9]%', CustomerName) - 1)
from [dbo].[SalesData_Encrypted] STG
where CustomerName LIKE '%[0-9]%'
```

The above SQL Code checks all the numeric data index from the column Customer Name and ignores them while extracting the Customer Name. Hence, standardising the Customer Names in the dataset to Test User instead of Test User1001, etc. We understood from the dataset that for one orderID, there cannot be multiple Customer Names. Thus, enrichment or consistency of Customer Name is an important transformation needed for our dataset.

- **Invalid datatype in quantity eg: 20.1123:** Since Quantity cannot have Float datatype, We converted all float quantities to INT Value using ROUND Function.

```
update STG SET Quantity = ROUND(Quantity,0)
from [dbo].[SalesData_Encrypted] as STG
where Quantity != CAST(Quantity AS INT);
```

d) Data Cleaning: To carry out this process, We followed the above mentioned transformation steps using different SQL Functions and inserted all the cleaned and valid data into a different table (Final_Sales_Data). This cleaned data is further used for Analysis.

```
-- Total Sales per OrderID
update [dbo].[SalesData_Encrypted] set TotalSales = Quantity * Price where ValidFlag IS NULL
```

C) Data Loading:

We applied both ETL and ELT Technique to load data into the Data Warehouse.

- **ETL:** While Ingesting data, We applied Data Encryption Functionality to credit card number ensuring the data security which comes under data transformation steps. Once this transformation was completed, The data from Flat File was loaded into OLE DB Destination using SSIS. This OLE DB Destination is a Staging Table called SalesData_Encrypted. Hence, this process supported

ETL Functionality in our data pipeline.

Column_name	Type	Length	ColumnType
OrderID	varchar	100	File Based
CustomerName	varchar	100	File Based
PhoneNumber	varchar	20	File Based
Location	varchar	100	File Based
Country	varchar	50	File Based
StoreCode	varchar	50	File Based
Product	varchar	100	File Based
Quantity	float	8	File Based
Price	float	8	File Based
Date	datetime	8	File Based
EncryptedCreditCardNumber	varbinary	256	File Based
ExpiryDate	varchar	10	File Based
Ingestion_Date	datetime	8	Audit Column
ValidFlag	varchar	10	Audit Column
TotalSales	float	8	Derived Column

Table 2: SalesData_Encrypted (Staging) Table Format

- **ELT:** Once the data was loaded into the staging table mentioned above, We performed above mentioned data transformation steps such as: Data Quality, Data Integrity Check, Data Cleaning. After Data Transformation, all the valid data was then loaded into a final table called, Final_Sales_Data. This process supported ELT Functionality in our Pipeline.

Column_name	Type	Length
orderID	varchar	100
CustomerName	varchar	100
PhoneNumber	varchar	20
Location	varchar	100
Country	varchar	50
StoreCode	varchar	50
Product	varchar	100
Quantity	float	8
Price	float	8
Date	datetime	8
EncryptedCreditCardNumber	varbinary	256
ExpiryDate	varchar	10
TotalSales	float	8

Table 3: Final_Sales_Data Table (clean data) format

SSIS Data Pipeline Development:

The data pipeline was built in SQL Server Integration Services (SSIS) using different SSIS components like Data Flow task, Execute SQL Task to execute stored procedures in SQL Server. SSIS Pipelines is a robust way of handling ETL or ELT process along with batch scheduling using SSMS SQL Server Agent functionality, to ensure we automate all the ETL Steps in any data engineering project.

We have 2 SSIS Packages for this project:

- **LoadSalesDataToDB.dtsx:** This SSIS Package is used for ETL and ELT process of sales data. It consists of steps – Truncate Staging table, Load the flat file data to SQL Server DB using Data flow task, performing all the data transformation steps using stored procedure. It also consists of a step to secure (encrypt) the credit card number provided, even before loading this information to table. This step ensures we have not exposed sensitive data to users.
- **CreateAllTablesAndSP.dtsx:** This SSIS package is used/ run to create all the Staging, Final and Lookup tables required for this particular project. If stakeholders needs to execute the data pipeline, they will have to run this package in their database as a first step. It will then create tables (SalesData_Encrypted, Final_Sales_Data, country_lookup) in their database which will be used in LoadSalesDataToDB.dtsx package.
- **AutomatedPipeline.dtsx:** This SSIS Package is the automated flow of above 2 steps. Running this package runs the CreateAllTablesAndSP.dtsx package first and then LoadSalesDataToDB.dtsx package.

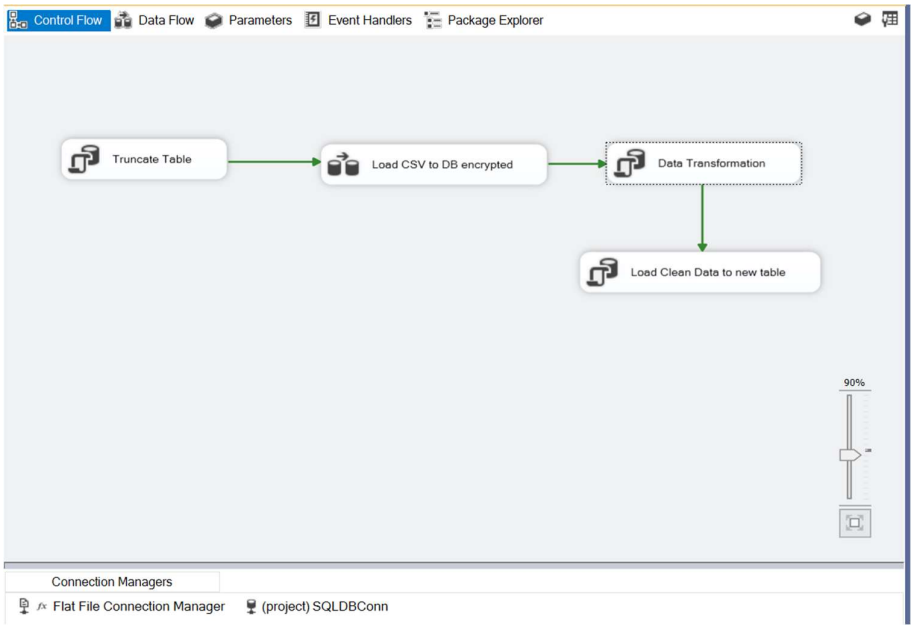


Figure 1 SSIS Data Pipeline for Data Ingestion

There are four steps in this pipeline:

1. Truncate table: This step runs the command to truncate the table for existing data in order to load the data for next run.
2. Load CSV to DB encrypted: This steps extracts the data from flat file, Encrypts the credit card number using Symmetric Key and Certificate and then Loads the data into SQL Server Destination Table using Flat File source and OLE DB Command Component.
   We are using Stored-Procedure called,

   ```
   [dbo].[InsertSalesDataWithEncryptedCreditCard]
     @OrderID = ?,
     @CustomerName = ?,
     @PhoneNumber = ?,
     @Location = ?,
     @Country = ?,
     @StoreCode = ?,
     @Product = ?,
     @Quantity = ?,
     @Price = ?,
     @Date = ?,
     @CreditCardNumber = ?,
     @ExpiryDate = ?
   ```

   All the above fields are columns taken from flat file used as variable to load into staging table

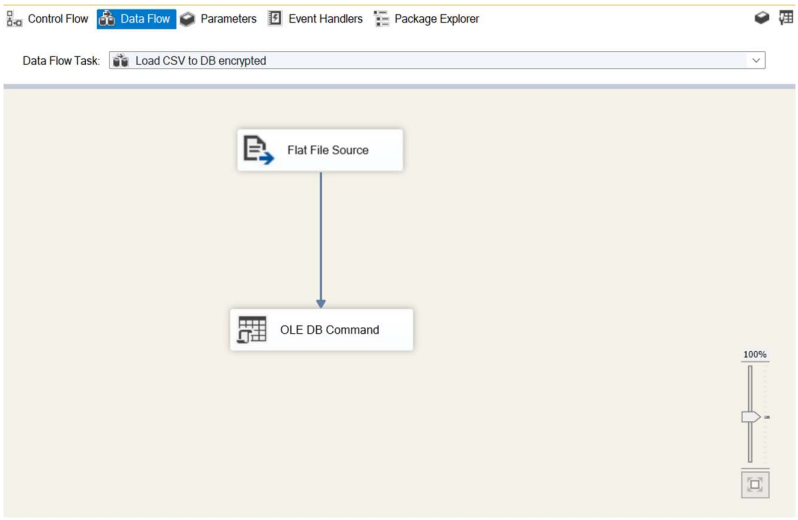


Figure 2 Data Flow Task for Loading data

3. Data Transformation: This step is used for Data Cleaning, Data Integrity Check and Data Validation. This step runs a Stored-Procedure called, [dbo].[salesdata_transformation]. This Stored Procedure consist of T-SQL Code for Data Transformation.

4. Load Clean Data to new table: After Data Transformation, All the cleaned data is further created into new table called: Final_Sales_Data

```
DROP TABLE IF EXISTS dbo.Final_Sales_Data;
--Generating cleaner data
select orderID, CustomerName, PhoneNumber, [Location], Country, StoreCode, Product,
Quantity, Price,
[Date], EncryptedCreditCardNumber, ExpiryDate, TotalSales
INTO dbo.Final_Sales_Data
from [dbo].[SalesData_Encrypted]
```

Sales Data Analysis:

1.) Total Sales Amount by OrderID

Insights gained from this metric: Which customer placed the highest order value based on total sales amount, and we also understood what was the highest quantity of sales for which customer. Hence, providing insight for most valuable customer.

Query for analysis:

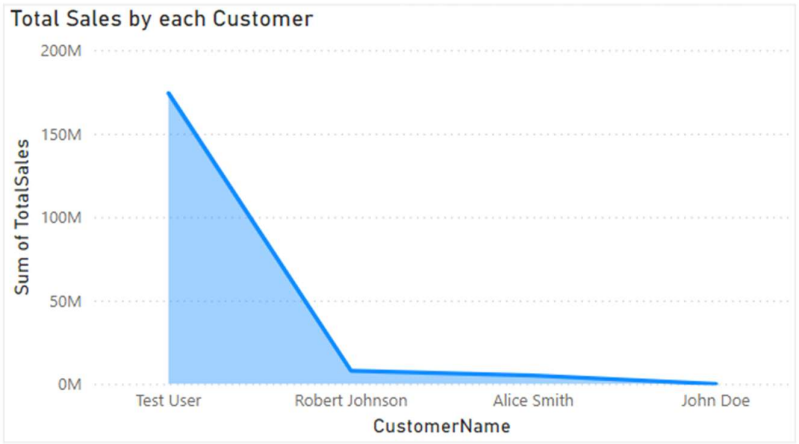
```
select
        orderID,
```

```
        CustomerName,
        [Location],
        sum(Quantity) TotalQuantity,
        sum(TotalSales) TotalSalesbyorderID
from [dbo].[Final_Sales_Data]
group by orderID, CustomerName, [Location]
order by TotalSalesbyorderID desc;
```

Result:

Results | Messages | Chart

	orderID	CustomerName	Location	TotalQuantity	TotalSalesbyorderID
1	DRTYF	Test User	Mumbai	975	174298443.3305664
2	NFCXII	Robert Johnson	Los Angeles	937	2796823.7626953125
3	CPLUKY	Alice Smith	Chicago	904	2672646.7126464844
4	LVZGLX	Robert Johnson	New York	882	2553266.550415039
5	KVZCLD	Robert Johnson	Houston	857	2445840.9936523438
6	ZYLXIT	Alice Smith	Houston	839	2333393.7117004395
7	JZSJXC	Robert Johnson	New York	7	6501.9498291015625
8	QELSGN	John Doe	Los Angeles	10	5724.000244140625
9	SVWHJS	Robert Johnson	Houston	6	5328.06005859375
10	WVHPTU	Robert Johnson	Chicago	8	5102.39990234375
11	HEXHEV	John Doe	New York	5	4899.849853515625
12	LSMOVN	Alice Smith	New York	8	4764.39990234375
13	MSYOAB	Alice Smith	Houston	6	4752.900146484375
14	YIIKFN	Alice Smith	Los Angeles	8	4600.39990234375



Insights Gathered:

- Test User is the most valuable customer with Highest Total sales amount
- Test User is most valuable customer ordering highest quantity of product from XYZ Retail.

- Mumbai is the most valuable location according to highest sales and quantity of product sold.

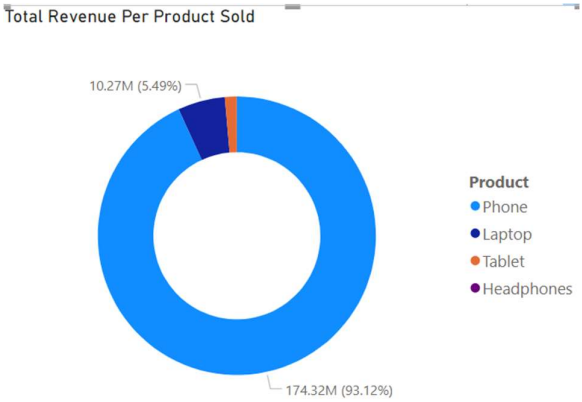
2) Total Sales Amount By Product:

This Metric gives us the best overall performing product, by total quantity of product sold and Total Sale Amount. We have the SQL Query to calculate the best performing product by highest number of quantity of product sold and also, by highest total sale amount grouped by product.

```
-- Total Sales by Product
-- best performing product by quantity of items sold and by totalsales
select
       Product,
       sum(Quantity) TotalQuantity,
       sum(TotalSales) TotalSalesbyProduct
       from [dbo].[Final_Sales_Data]
group by Product
order by TotalSalesbyProduct desc;
```

Results | Messages

	Product	TotalQuantity	TotalSalesbyProduct
1	Phone	1020	174324274.74032593
2	Laptop	3604	10271774.480529785
3	Tablet	936	2581196.5408935547
4	Headphones	56	23029.909957885742



Insights Gained from above analysis:

- Phone is the most sold product if we observe by total sales. The most highest revenue generating product for XYZ Retail is “Phone”
- Customers are more interested in buying laptops, if we observe the best performing product by quantities sold.
- Headphone is the least favourite product among XYZ customers.

3) Average Order Value by OrderID:

This metric gives us the the Average Order Value of each OrderID by calculating the average of Total Sales Sum and the frequency of order as orderOccurance. The query

```
-- Average Order Value by OrderID
select OrderID, CustomerName, round((TotalSalesSum/orderOccurance), 2, 0)
AvgOrderValuebyOrderID  from  (
    select orderID, CustomerName, Count(*) orderOccurance,
    sum(TotalSales) TotalSalesSum
    from [dbo].[Final_Sales_Data]
group by orderID, CustomerName
) orderfreq order by AvgOrderValuebyOrderID desc
```

	OrderID	CustomerName	AvgOrderValuebyOrderID
1	DRTYF	Test User	15845313.03
2	NFCXII	Robert Johnson	254256.71
3	CPLUKY	Alice Smith	242967.88
4	LVZGLX	Robert Johnson	232115.14
5	KVZCLD	Robert Johnson	222349.18
6	ZYLXIT	Alice Smith	212126.7
7	JZSJXC	Robert Johnson	6501.95
8	QELSGN	John Doe	5724
9	SVWHJS	Robert Johnson	5328.06
10	WVHPTU	Robert Johnson	5102.4
11	HEXHEV	John Doe	4899.85
12	LSMOVN	Alice Smith	4764.4
13	MSYOAB	Alice Smith	4752.9
14	YIIKFN	Alice Smith	4600.4
15	DARJZI	John Doe	4571.64

Insight Gained from above analysis:

- Test User with OrderId as “DRTYF” has the highest Average Order Value.
- The most valuable customer according to average order value per orders is Test User who hails from Mumbai, India.

4) Top product by total sales and average order value:

To calculate the best product sold by XYZ Retail Inc., We are considering the factor such as highest number of product sold by calculating the total sales and the average order value for the same. The query to calculate it is given below:

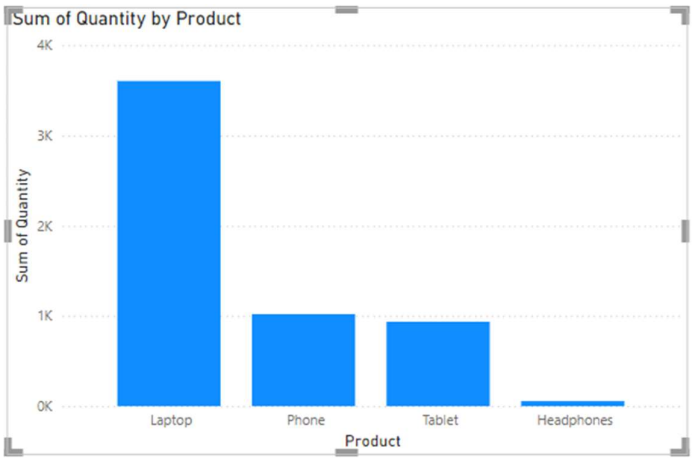
```sql
-- Top performing product by total sales and average order value


select Product, round((TotalSalesSum/orderOccurance), 2, 0)
AvgOrderValuebyProduct from  (
    select Product, Count(*) orderOccurance,
    sum(TotalSales) TotalSalesSum
    from [dbo].[Final_Sales_Data]
    group by Product
) orderfreq order by AvgOrderValuebyProduct desc
```

Results Messages

	Product	AvgOrderVal…
1	Phone	9684681.93
2	Laptop	193807.07
3	Tablet	135852.45
4	Headphones	2558.88

Insights gained:

- Phone is the top performing product by average order value.
- Headphone is the least favourite product of XYZ Retail.



5) Top Performing Product in each year:

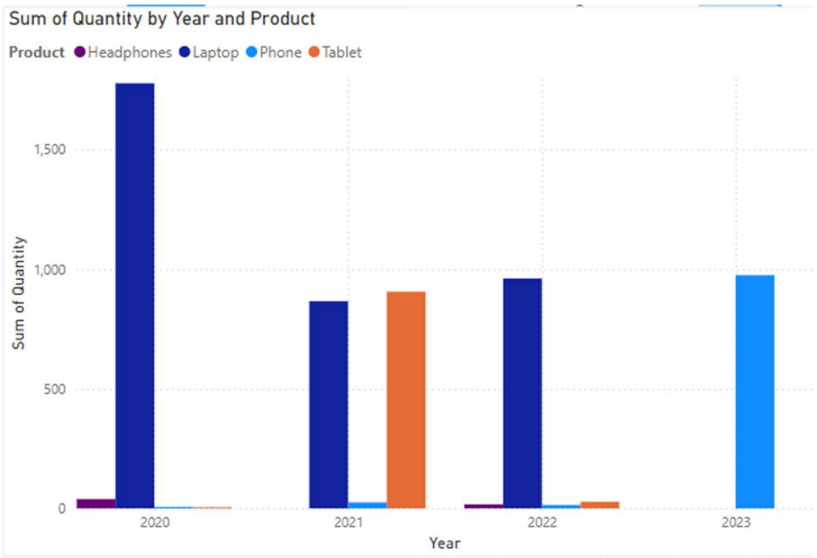
Using this metrics, we are analyzing the most valuable product by total sales amount in each year.

```sql
-- Top performing product in each year by highest totalsalesamount
select * from (
select product, YEAR([Date]) [Year], SUM(TotalSales)
[SumofTotalSalesAmtByProd],
ROW_NUMBER() OVER(PARTITION BY YEAR([Date]) ORDER BY SUM(TotalSales) DESC ) rn
from [dbo].[Final_Sales_Data]
GROUP BY product,YEAR([Date])
```

```
) A where rn =1
```

Results Messages

	product	Year	SumofTotalSalesAmtByProd	rn
1	Laptop	2020	5124488.0263671875	1
2	Tablet	2021	2562444.090698242	1
3	Laptop	2022	2804754.642578125	1
4	Phone	2023	174298443.3305664	1



The Insights Gained are:

- Laptop was top performing product for two alternate years 2020 and 2022.
- In the latest year, according to table in 2023 Phone was the top performing product with the highest total sales amount comparing last three years data.
- In year 2023, XYZ Retail earned the highest revenue ever in last three years by selling phone.

6) Most valuable customer in each year:

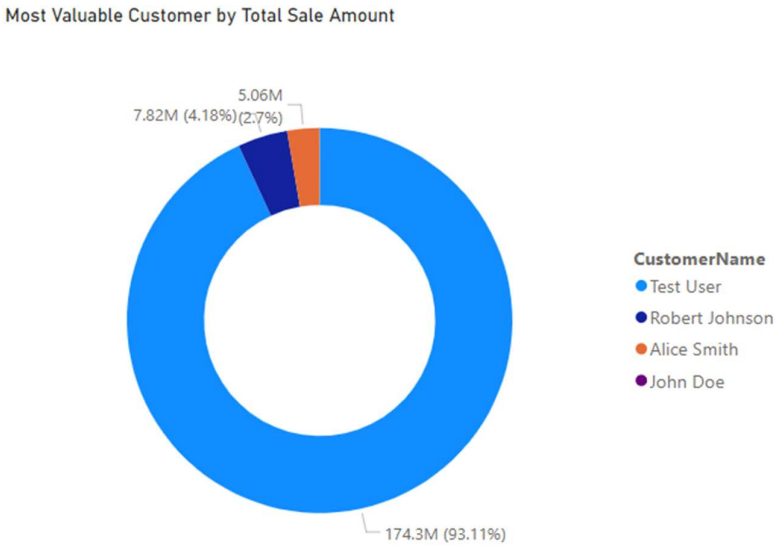
From this metrics, we are analysing the most valuable customer as per the record of sum of total sales per year. The query to partition data by year is given below:

```
select * from (
select OrderID, CustomerName, YEAR([Date]) [Year], sum(TotalSales)
TotalSalesSum,
       ROW_NUMBER() OVER(PARTITION BY YEAR([Date]) ORDER BY sum(TotalSales)
DESC ) rn
       from [dbo].[Final_Sales_Data]
```

```
GROUP BY OrderID, CustomerName, YEAR([Date])) A
where rn =1
```

Results | Messages

	OrderID	CustomerName	Year	TotalSalesSum	rn
1	CPLUKY	Alice Smith	2020	2672646.7126464844	1
2	LVZGLX	Robert Johnson	2021	2553266.550415039	1
3	NFCXII	Robert Johnson	2022	2796823.7626953125	1
4	DRTYF	Test User	2023	174298443.3305664	1



The insights gained are:

- In year 2023, XYZ Retail earned the highest revenue ever from Customer, Test User who contributed to 93% of total revenue of XYZ Retail..
- The second highest revenue was generated in the year 2022 by the Customer, Robert Johnson.

7) Count of Orders by each Customer:

In order to analyse sales data, we have calculated the order count per customer. We can also gather the information of most valuable products and most active customer per year categorized by total count of orders and highest total sales sum.

```
select count(*) countoforders, CustomerName, sum(TotalSales) TotalSalesSum from
[dbo].[Final_Sales_Data]
group by CustomerName
```

Results Messages

	countoforders	CustomerName	TotalSalesSum
1	41	Alice Smith	5055519.84425354
2	8	John Doe	29107.640258789062
3	39	Robert Johnson	7817204.856628418
4	11	Test User	174298443.3305664

The insights that we can gather are:

- Alice Smith is the most active customer in the category of highest count of orders.
- Test User is the most valuable customer in the category of highest revenue earning.