Name:Kaushik Kotian Roll no.:30 Div:D15B Batch:B

Experiment No.:-1

Case Study on Finance

Background
An online retail company has broadened its offerings to include a Digital Transaction feature, enabling customers to store credit and debit transaction details for purchases made on the platform. This Digital Wallet supports various transaction types, including deposits, withdrawals, and purchases, across different account types such as Savings, Checking, and Credit.

Objective
The Finance Department aims to deepen its understanding of Digital Transaction utilization, analyze financial activities, and furnish users with detailed transaction reports. To accomplish this, a sophisticated system was developed to monitor and report on transactions, budgets, and spending categories.

Design Data Warehouse

1. Select the Business Process

- The primary business process focuses on Digital transactions, including both credits (e.g., deposits) and debits (e.g., purchases, withdrawals).

2. Declare the Grain

- The grain of the data warehouse is the individual eWallet transaction. This means that every record in the fact table represents a single transaction (credit or debit) that occurs in an eWallet.

3. Identify the Dimensions

- Customer Dimension: Contains details about the eWallet owners, such as customer ID, name, and contact information.
- Account Dimension: Information about each account, including account ID, account type (Savings, Checking, Credit), and account name.
- Time Dimension: Date of transaction, with attributes for day, month, year, and possibly time for more detailed analysis.

- Category Dimension: Types of transactions, categorized (e.g., Groceries, Utilities, Entertainment).

4. Identify the Facts

- Transaction Fact Table: Holds the key metrics of interest, such as transaction ID, customer ID (linked to the Customer Dimension), account ID, transaction date (linked to the Time Dimension), amount, and transaction type (credit/debit). It can also include foreign keys to dimensions such as Category.

Implementation of Data Warehouse

- Database Name: eWalletDB
- Dimension Tables:
  Customer Dimension Table: CustomerID, Name, ContactInfo, etc.
  Account Dimension Table: AccountID, AccountType, AccountName, linked to Customer Dimension.
  Time Dimension Table: DateKey (YYYYMMDD), Day, Month, Year, Quarter, Weekday, etc.
  Category Dimension Table: CategoryID, CategoryName.
- Fact Table:
  Transactions Fact Table: TransactionID, CustomerID, AccountID, DateKey, Amount, TransactionType, CategoryID.

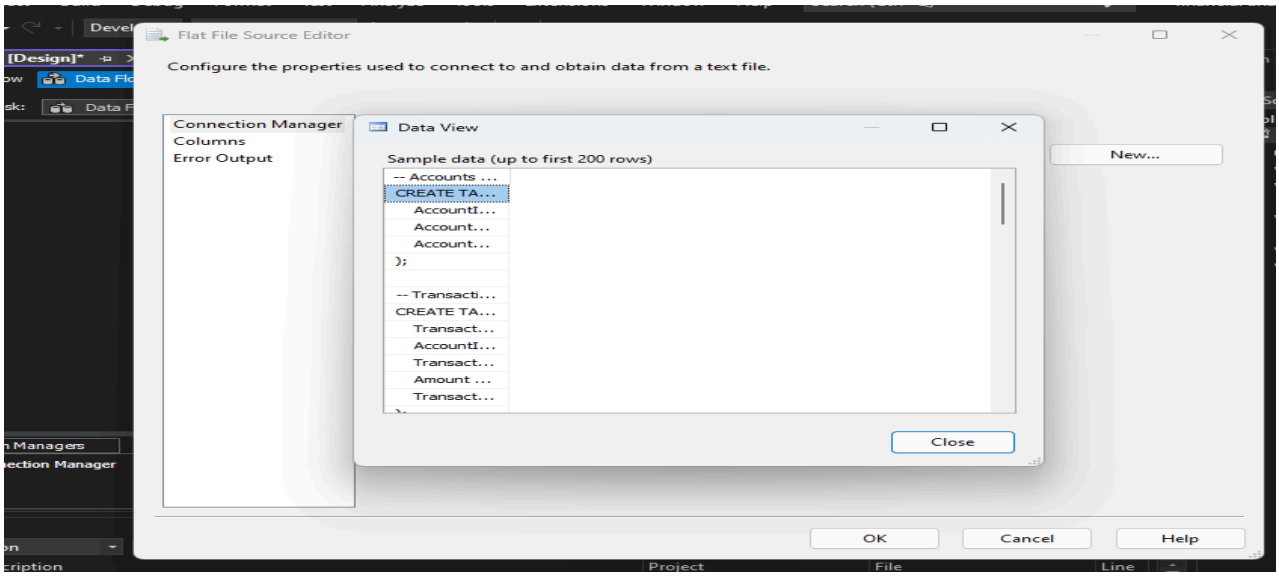
This design aims to facilitate complex queries and reports that can analyze transactions by various dimensions, providing the company with insights into eWallet usage and helping consumers track their finances more effectively.

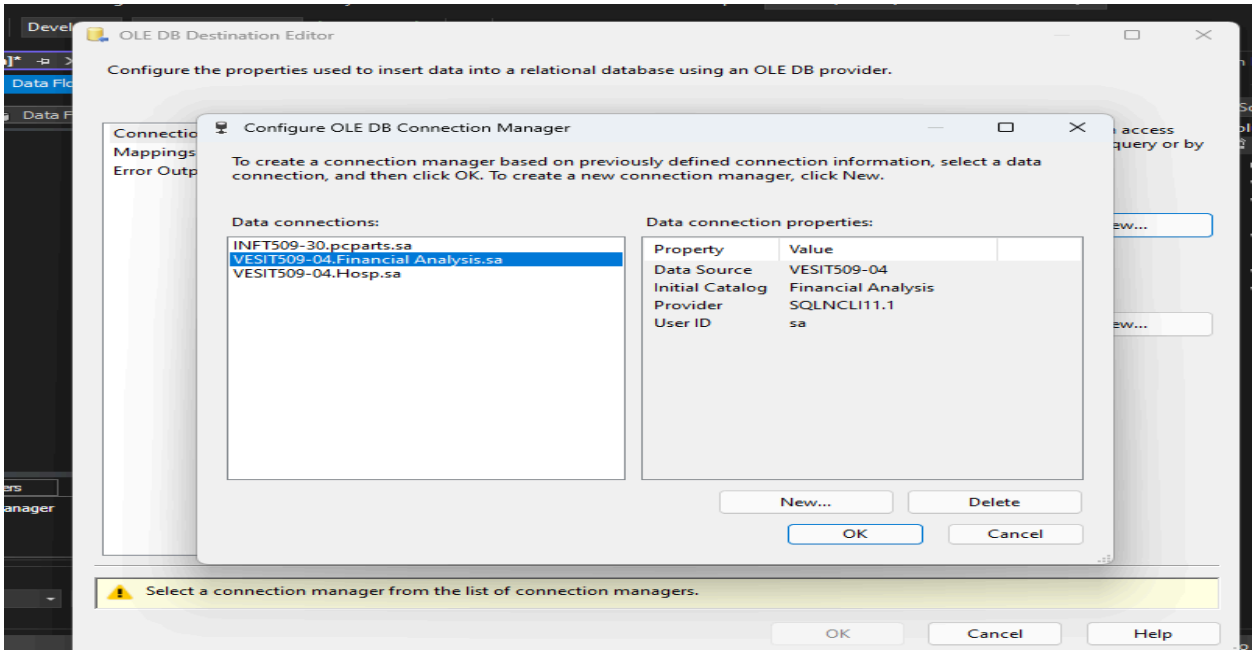
```
-- Drop existing tables if they exist to avoid errors
-- Note: Order of dropping tables is important due to foreign key constraints
IF OBJECT_ID('TransactionCategories', 'U') IS NOT NULL DROP TABLE
TransactionCategories;
IF OBJECT_ID('Budgets', 'U') IS NOT NULL DROP TABLE Budgets;
IF OBJECT_ID('Transactions', 'U') IS NOT NULL DROP TABLE Transactions;
IF OBJECT_ID('Categories', 'U') IS NOT NULL DROP TABLE Categories;
IF OBJECT_ID('Accounts', 'U') IS NOT NULL DROP TABLE Accounts;

-- Create Accounts table
CREATE TABLE Accounts (
    AccountID INT PRIMARY KEY,
    AccountName VARCHAR(100),
```

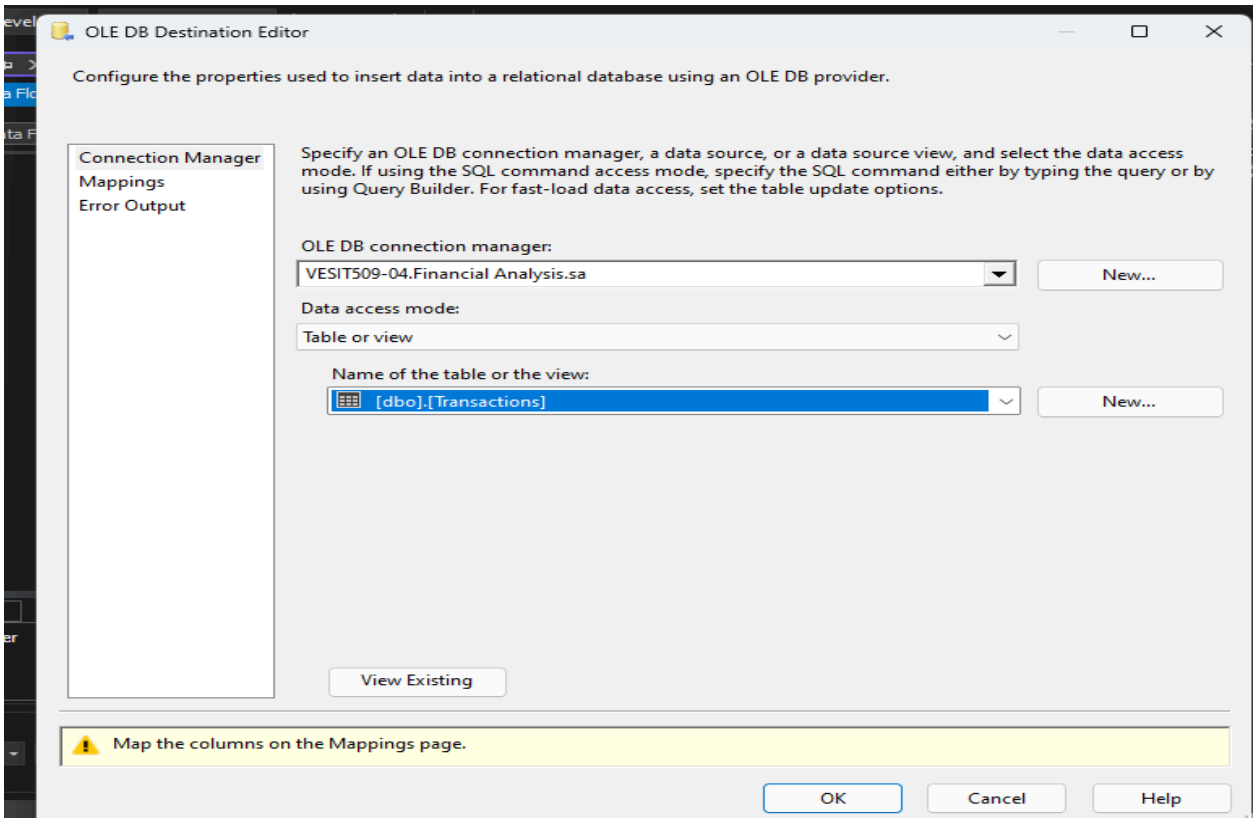
```
    AccountType VARCHAR(50)
);
-- Create Transactions table
CREATE TABLE Transactions (
    TransactionID INT PRIMARY KEY,
    AccountID INT,
    TransactionDate DATE,
    Amount DECIMAL(10, 2),
    TransactionType VARCHAR(50),
    FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID)
);

-- Create Budgets table
CREATE TABLE Budgets (
    BudgetID INT PRIMARY KEY,
    AccountID INT,
    BudgetYear INT,
    Amount DECIMAL(10, 2),
    FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID)
);

-- Create Categories table
CREATE TABLE Categories (
    CategoryID INT PRIMARY KEY,
    CategoryName VARCHAR(100)
);

-- Create TransactionCategories table
CREATE TABLE TransactionCategories (
    TransactionID INT,
    CategoryID INT,
    PRIMARY KEY (TransactionID, CategoryID),
    FOREIGN KEY (TransactionID) REFERENCES Transactions(TransactionID),
    FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID)
);

-- Insert data into Accounts
INSERT INTO Accounts (AccountID, AccountName, AccountType) VALUES
(1, 'Savings Account', 'Savings'),
(2, 'Checking Account', 'Checking'),
(3, 'Credit Card', 'Credit');
```

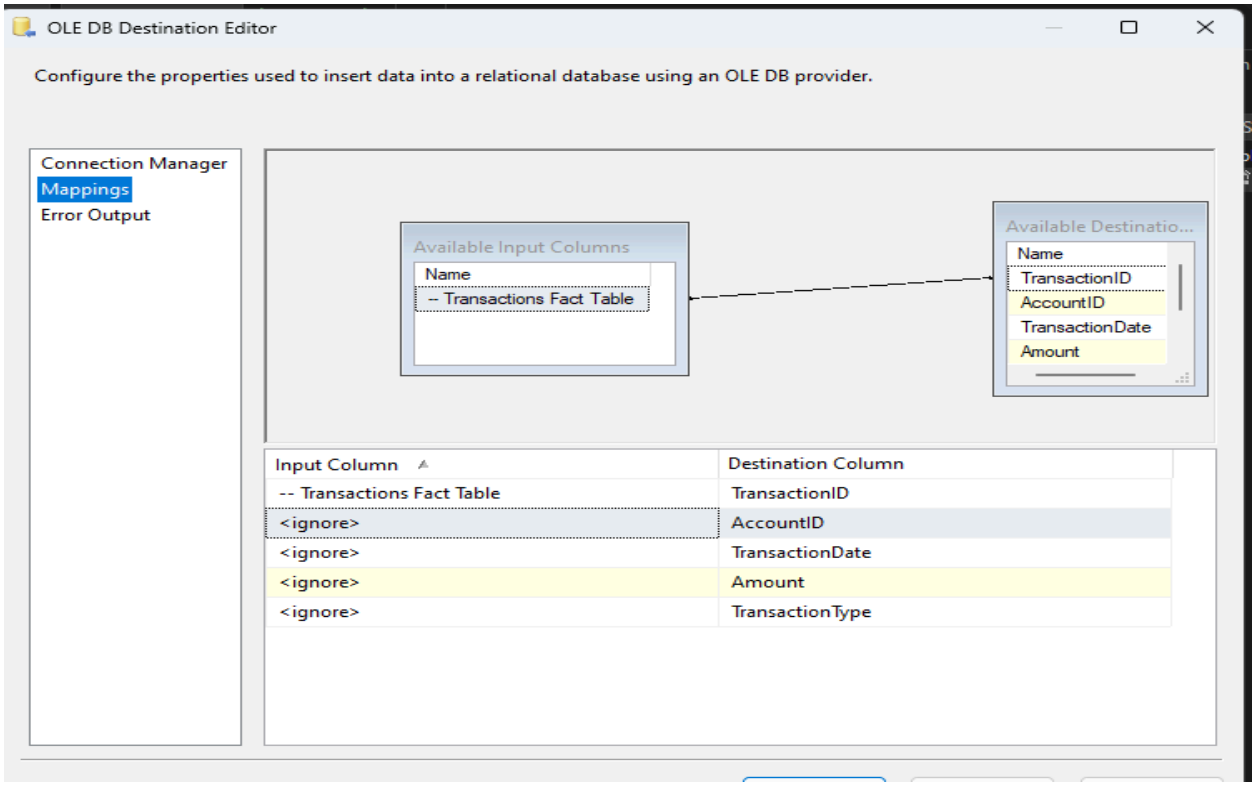
```
-- Insert data into Transactions
INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount,
TransactionType) VALUES
(101, 1, '2024-02-08', 500.00, 'Deposit'),
(102, 2, '2024-02-09', -100.00, 'Withdrawal'),
(103, 3, '2024-02-10', 50.00, 'Purchase');

-- Insert data into Budgets
INSERT INTO Budgets (BudgetID, AccountID, BudgetYear, Amount) VALUES
(201, 1, 2024, 1000.00),
(202, 2, 2024, 800.00),
(203, 3, 2024, 500.00);

-- Insert data into Categories
INSERT INTO Categories (CategoryID, CategoryName) VALUES
(301, 'Groceries'),
(302, 'Utilities'),
(303, 'Entertainment');

-- Insert data into TransactionCategories
INSERT INTO TransactionCategories (TransactionID, CategoryID) VALUES
(101, 301),
(103, 302),
(102, 303);
```

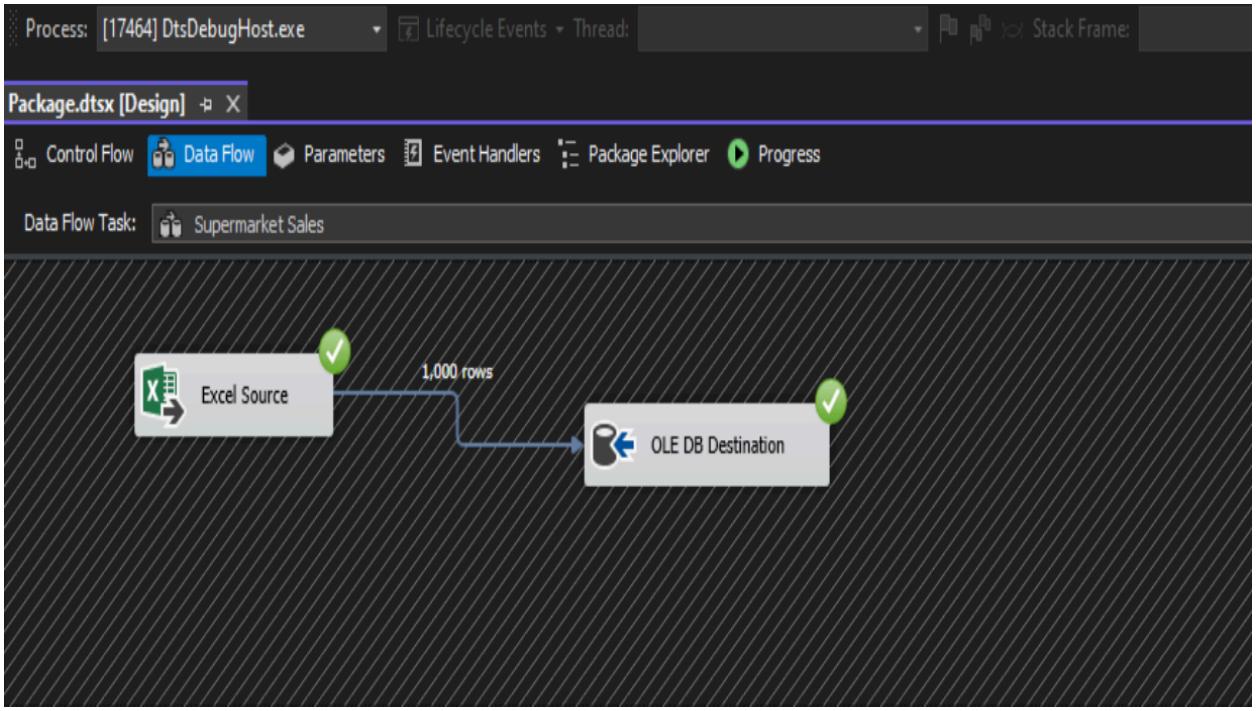
ETL Operation:

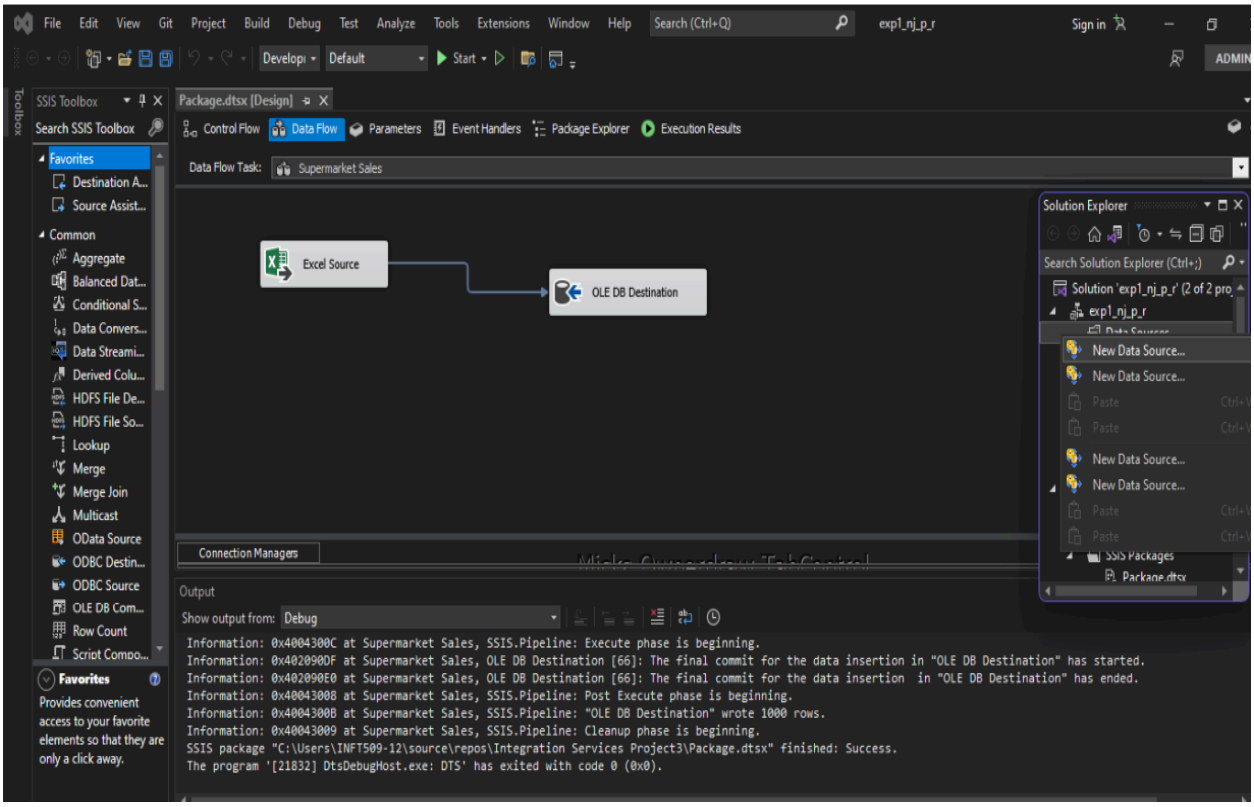



OLE DB Destination Editor
Configure the properties used to insert data into a relational database using an OLE DB provider.
Configure OLE DB Connection Manager
To create a connection manager based on previously defined connection information, select a data connection, and then click OK. To create a new connection manager, click New.
Data connections:
INFT509-30.pcparts.sa
VESIT509-04.Financial Analysis.sa
VESIT509-04.Hosp.sa
Data connection properties:
Property
Value
Data Source
VESIT509-04
Initial Catalog
Financial Analysis
Provider
SQLNCLI11.1
User ID
sa
New...
Delete
OK
Cancel
Select a connection manager from the list of connection managers.
OK
Cancel
Help


OLE DB Destination Editor
Configure the properties used to insert data into a relational database using an OLE DB provider.
Connection Manager
Mappings
Error Output
Specify an OLE DB connection manager, a data source, or a data source view, and select the data access mode. If using the SQL command access mode, specify the SQL command either by typing the query or by using Query Builder. For fast-load data access, set the table update options.
OLE DB connection manager:
VESIT509-04.Financial Analysis.sa
New...
Data access mode:
Table or view
Name of the table or the view:
[dbo].[Transactions]
New...
View Existing
Map the columns on the Mappings page.
OK
Cancel
Help


OLE DB Destination Editor
Configure the properties used to insert data into a relational database using an OLE DB provider.
Connection Manager
Mappings
Error Output
Available Input Columns
Name
-- Transactions Fact Table
Available Destinatio...
Name
TransactionID
AccountID
TransactionDate
Amount
Input Column
Destination Column
-- Transactions Fact Table
TransactionID
<ignore>
AccountID
<ignore>
TransactionDate
<ignore>
Amount
<ignore>
TransactionType

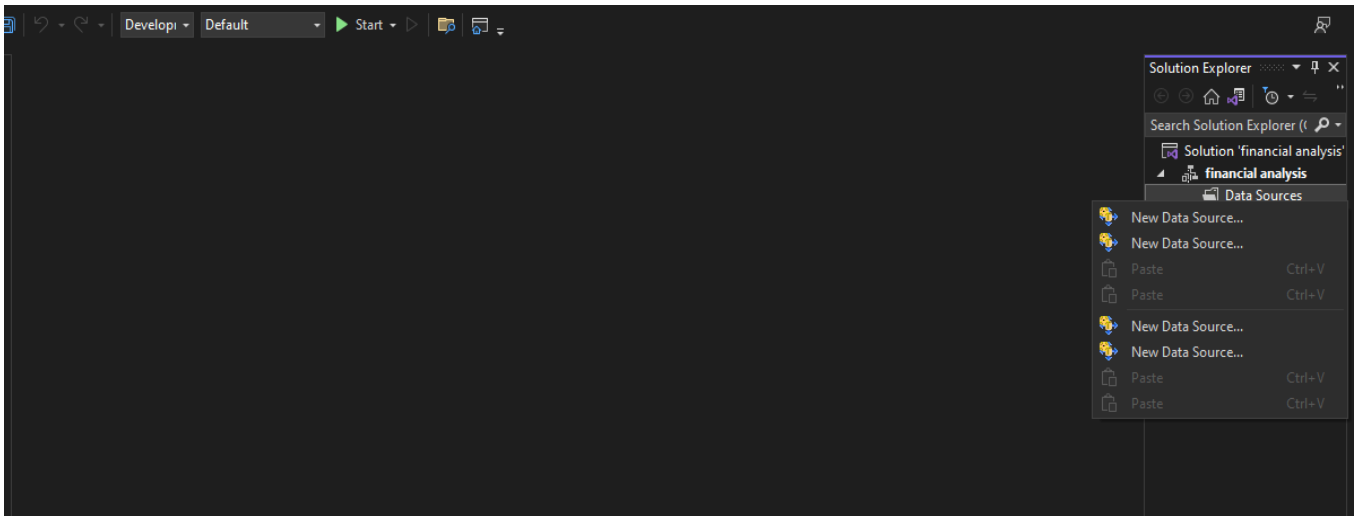

Process: [17464] DtsDebugHost.exe
Lifecycle Events
Thread:
Stack Frame:
Package.dtsx [Design]
Control Flow
Data Flow
Parameters
Event Handlers
Package Explorer
Progress
Data Flow Task:
Supermarket Sales
Excel Source
1,000 rows
OLE DB Destination

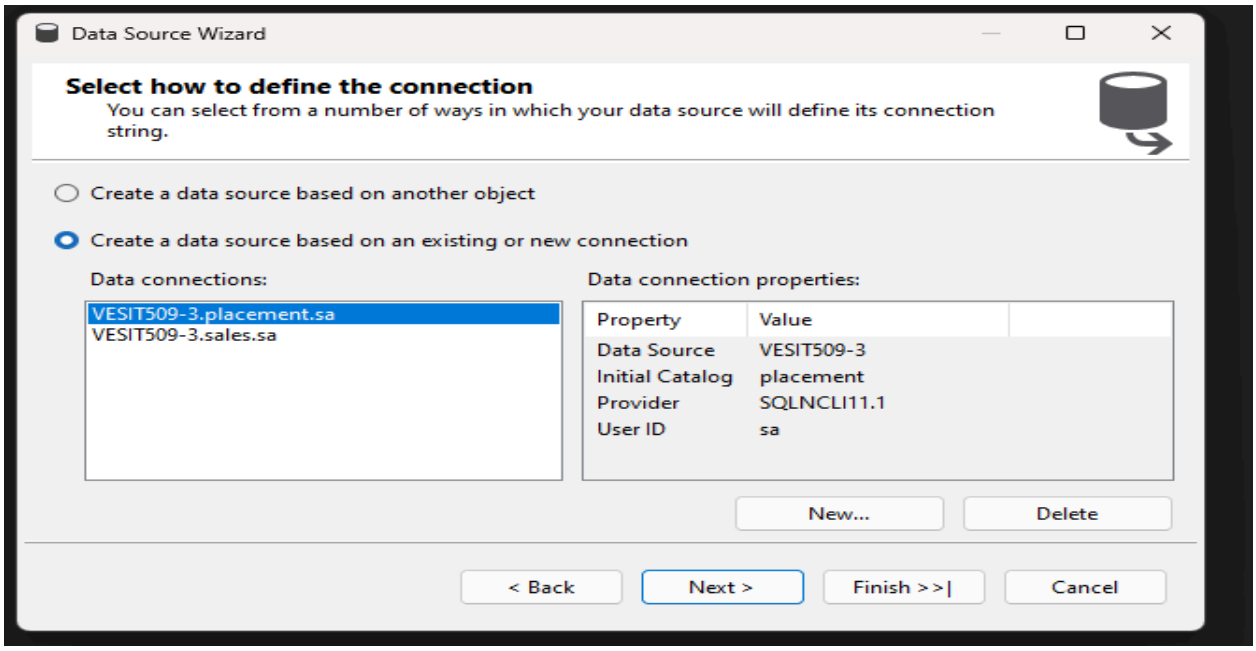


Cube Setup:

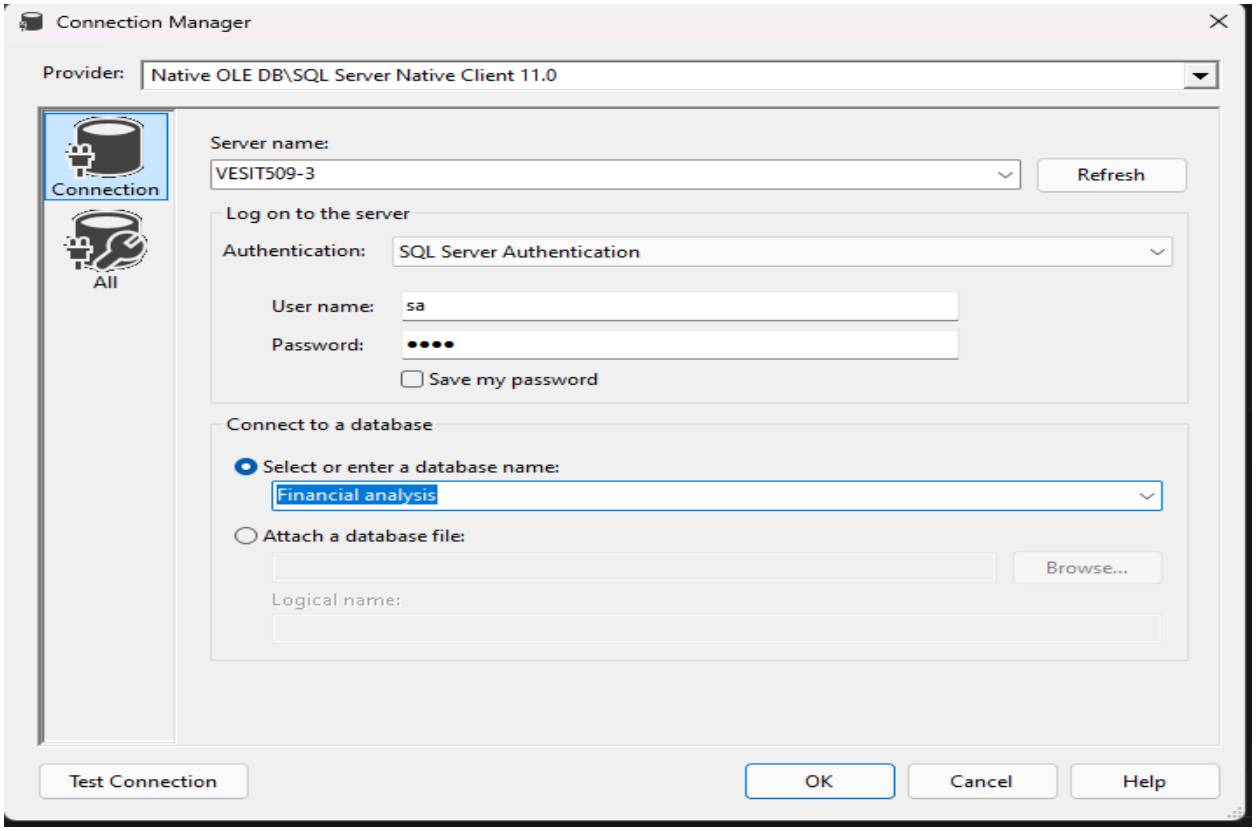
1. Create a new "Analysis Services Multidimensional and Data Mining Project" with the name "Financial Analysis" on Visual Studio.

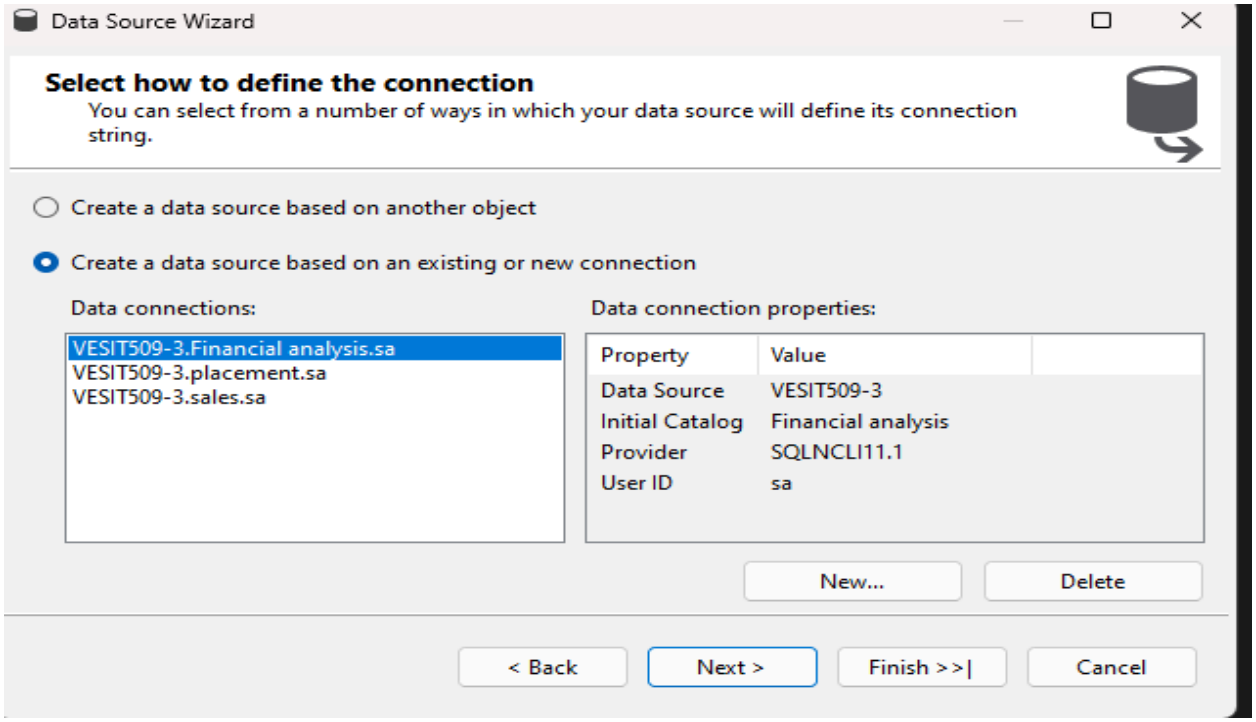
2. In Solution Explorer, Right click on Data Source -> Click New Data Source



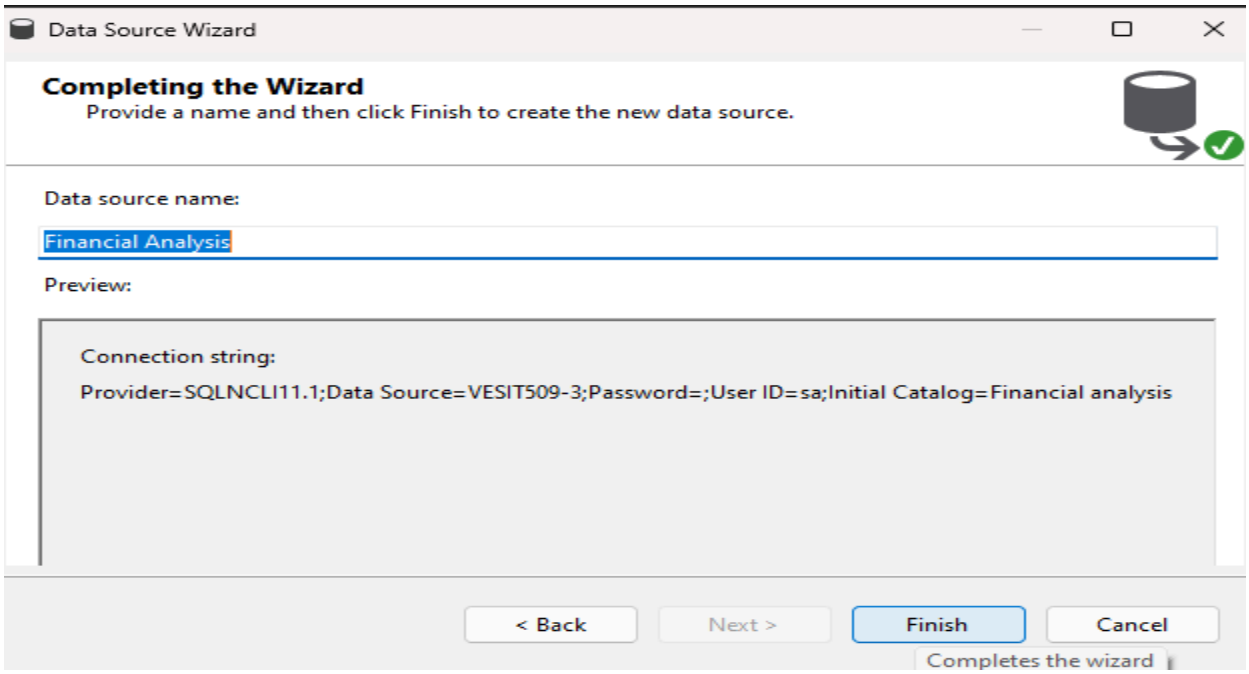


3. Click on Next. Click on New Button and create a new SQL Server Connection.



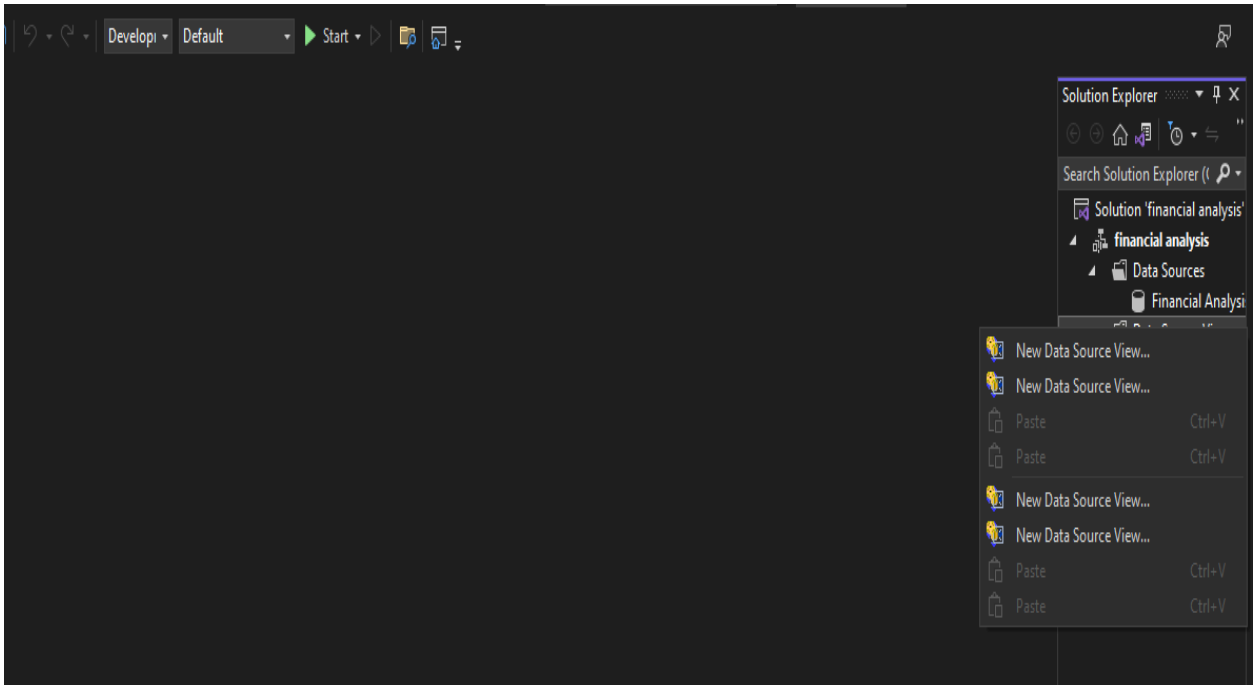


4. Select Connection created in Data Connections-> Click Next. Select Option Inherit. Assign Data Source Name -> Click Finish



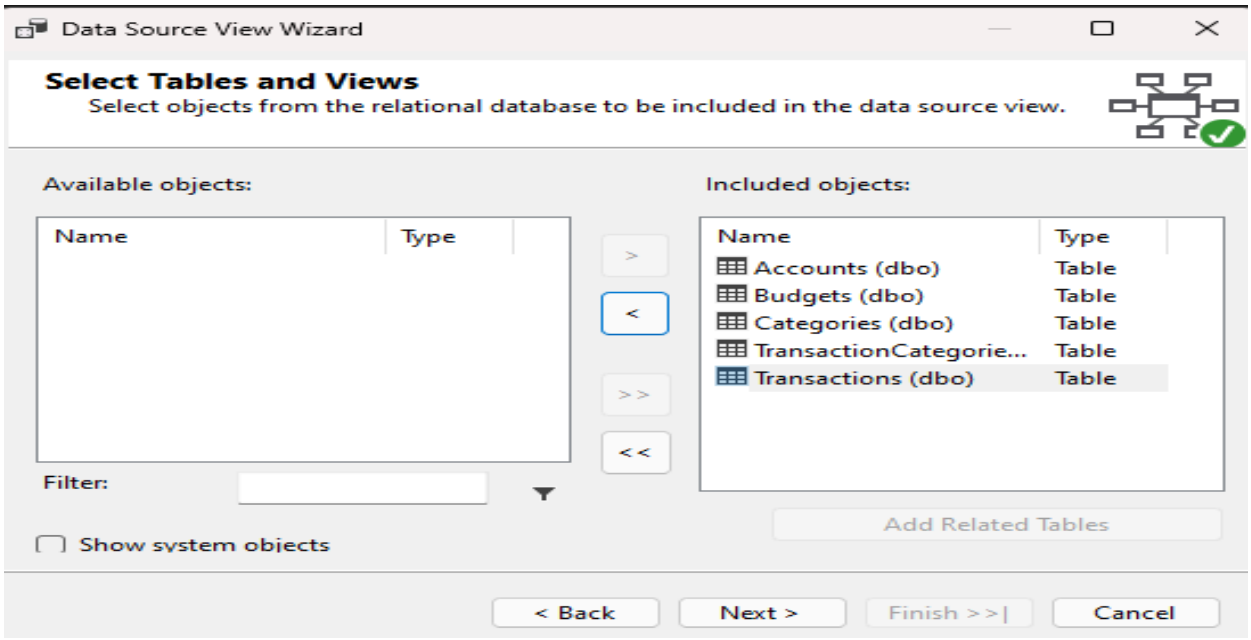
5. In the Solution Explorer, Right Click on Data Source View -> Click on New Data Source View.

6. Click Next. Select Relational Data Source we have created previously-> Click Next

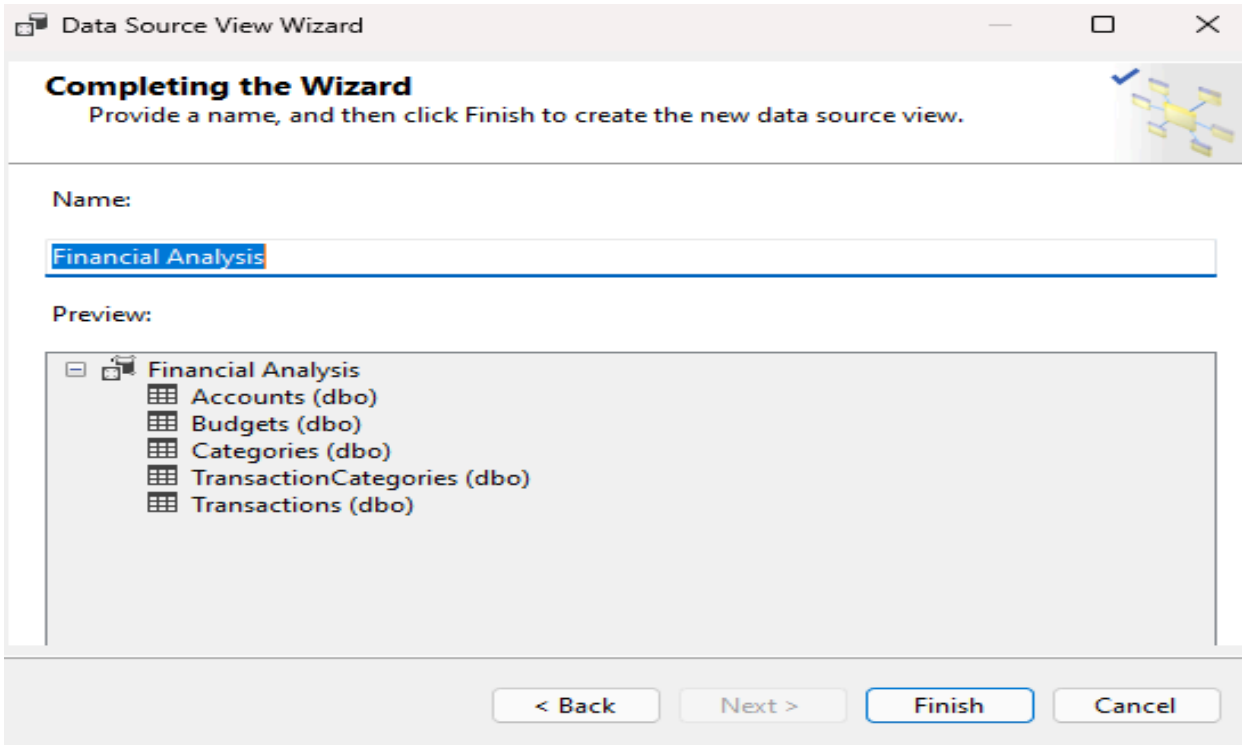


7. First move your Fact Table to the right side to include in the objects list. Select Transaction table -> Click on Arrow Button to move the selected object to the Right

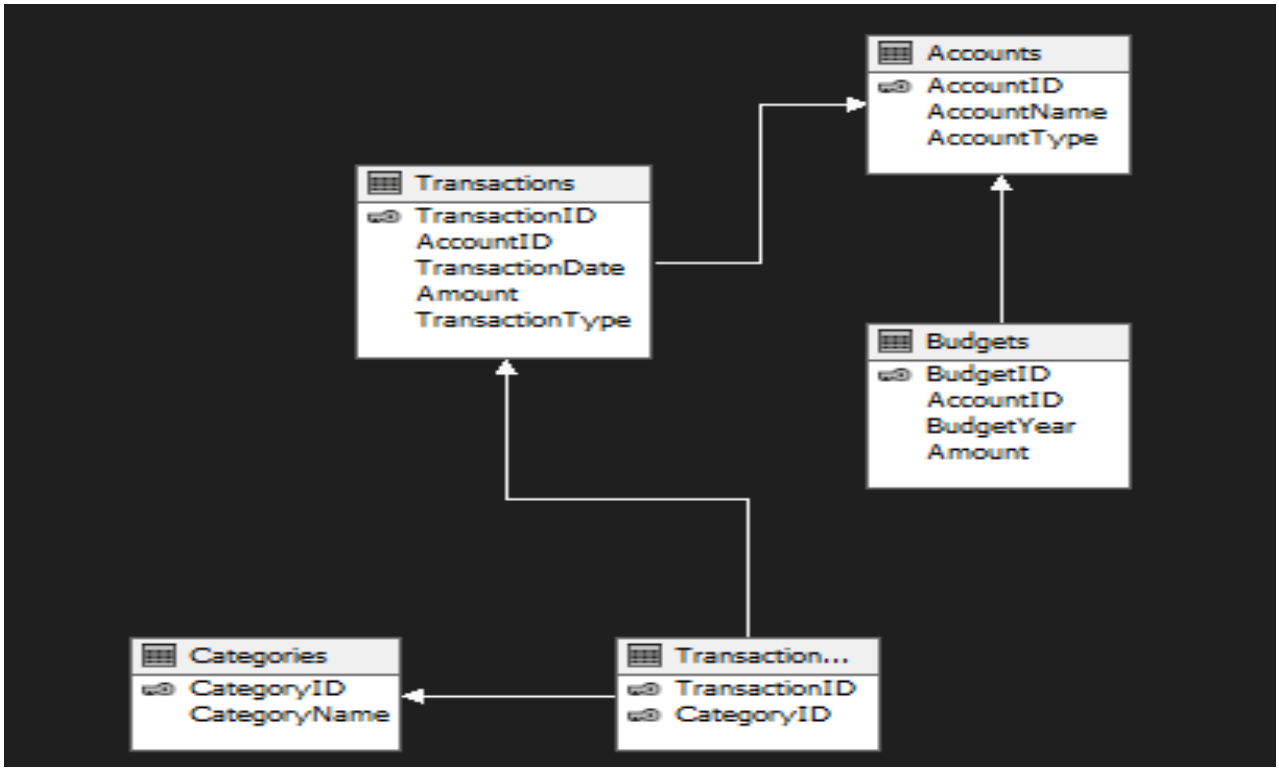
8. Now to add dimensions which are related to your Fact Table. Select Fact Table in Right Pane -> Click On Add Related Tables



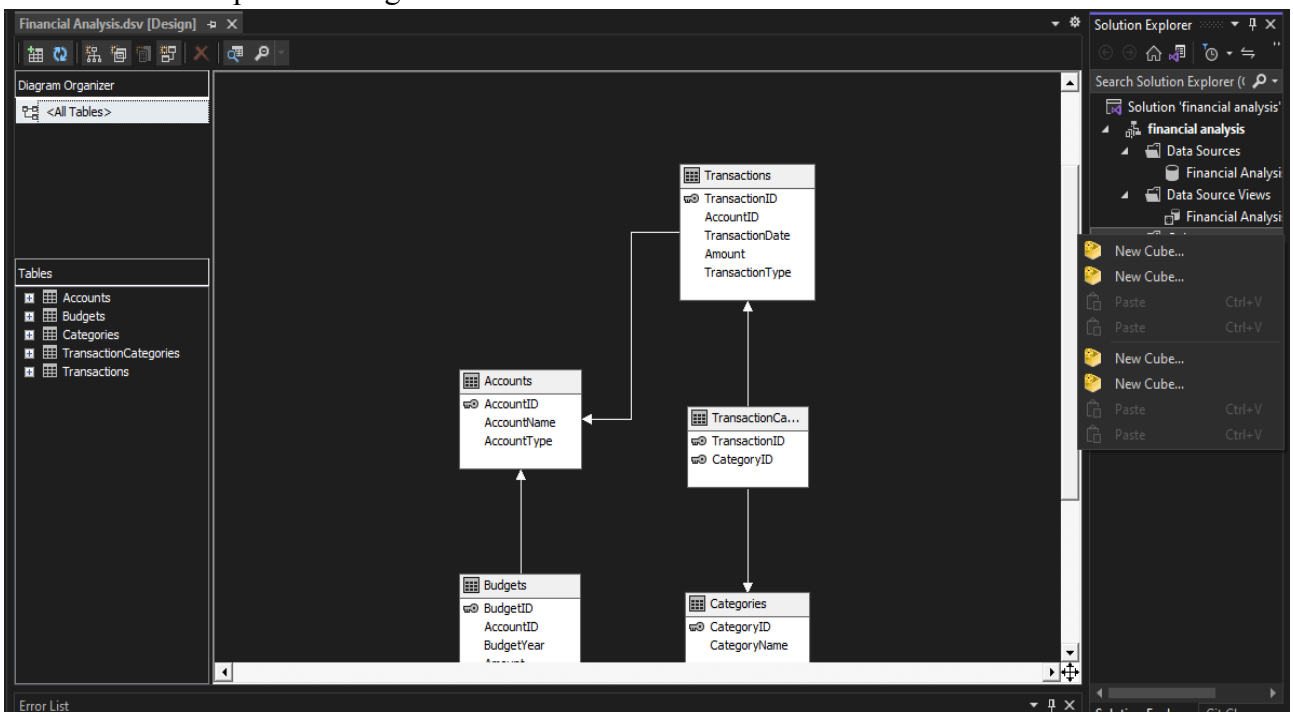
9. Assign appropriate name (Financial Analusis SourceView) -> Click Finish



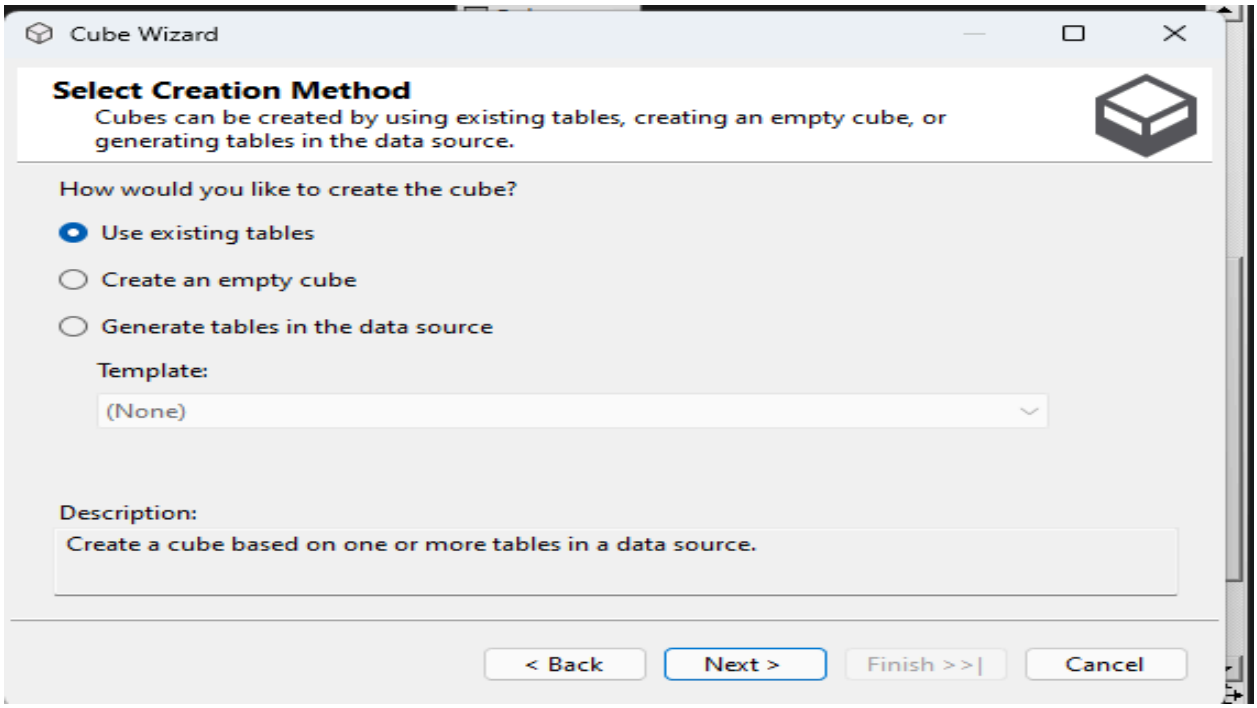
10. Double click on the data source view we created.



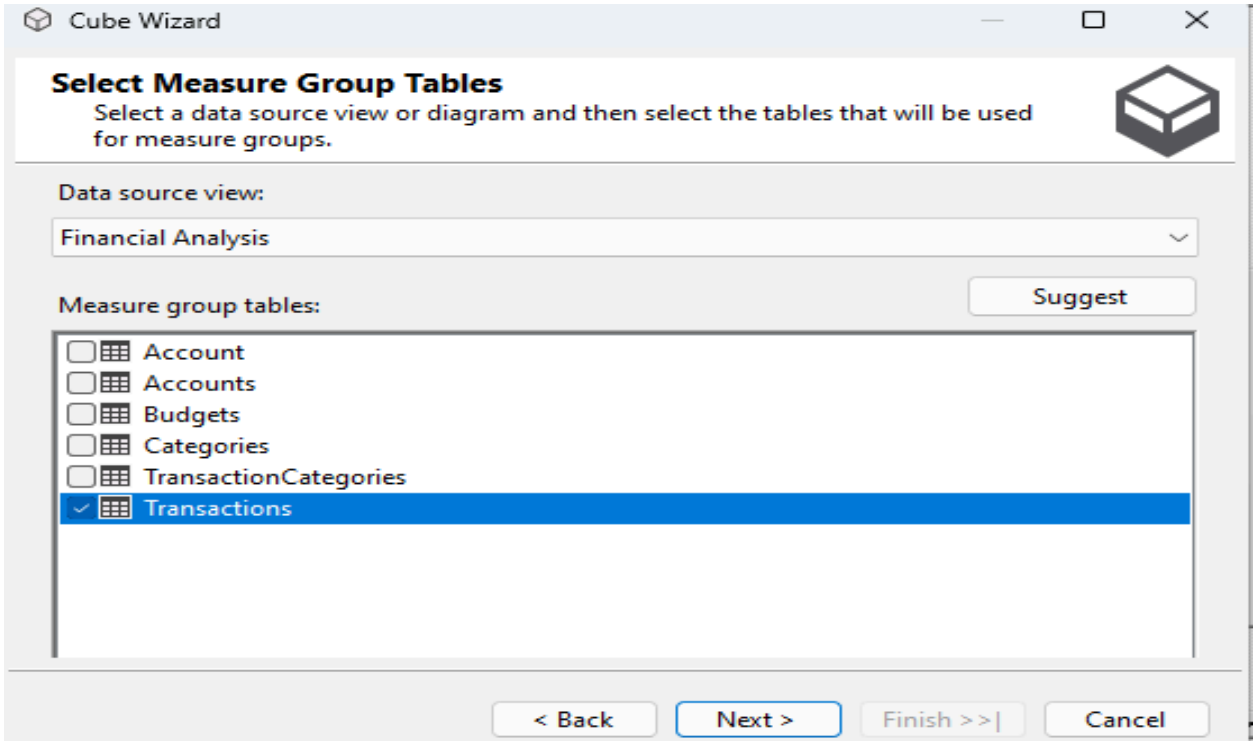
11. In Solution Explorer -> Right Click on Cube-> Click New Cube



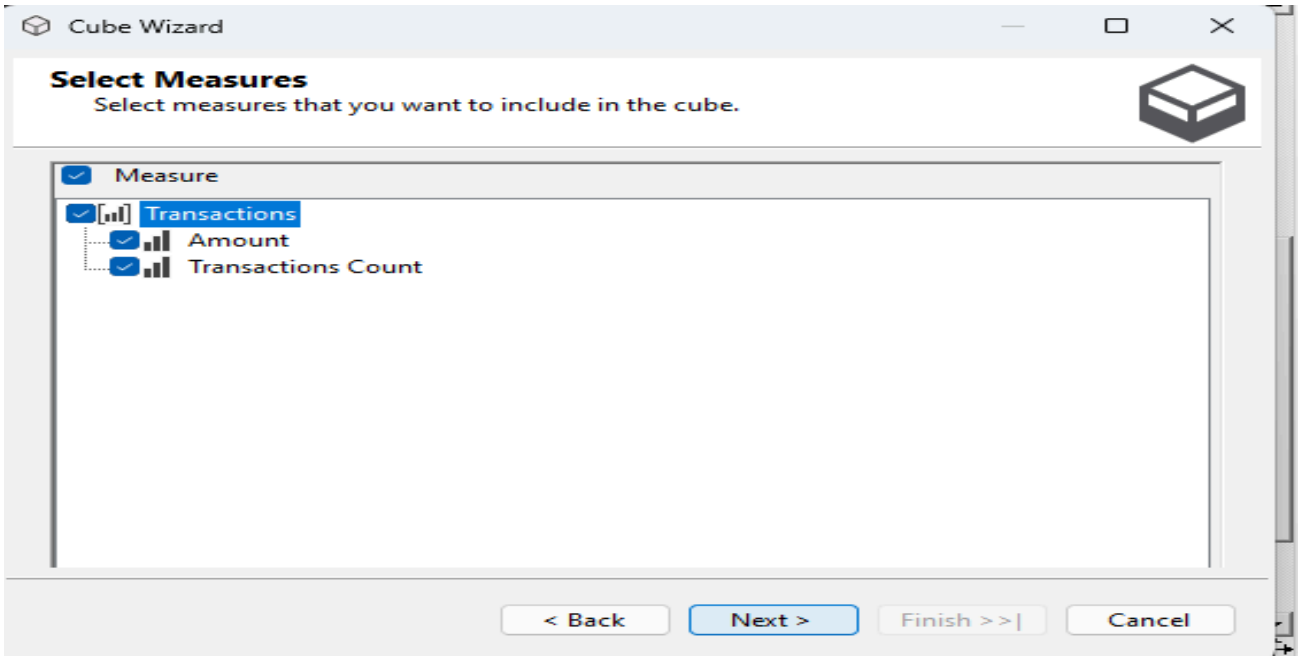
12. Click Next. Select Option Use existing Tables -> Click Next.



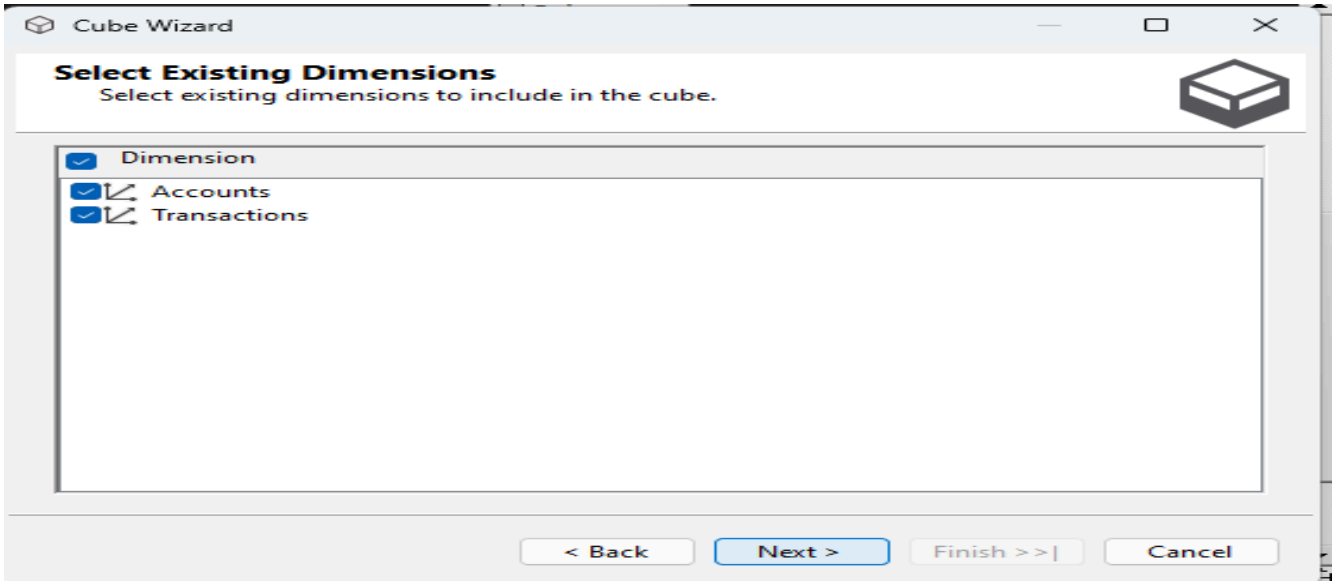
13. Select Fact Table Name from Measure Group Tables -> Click



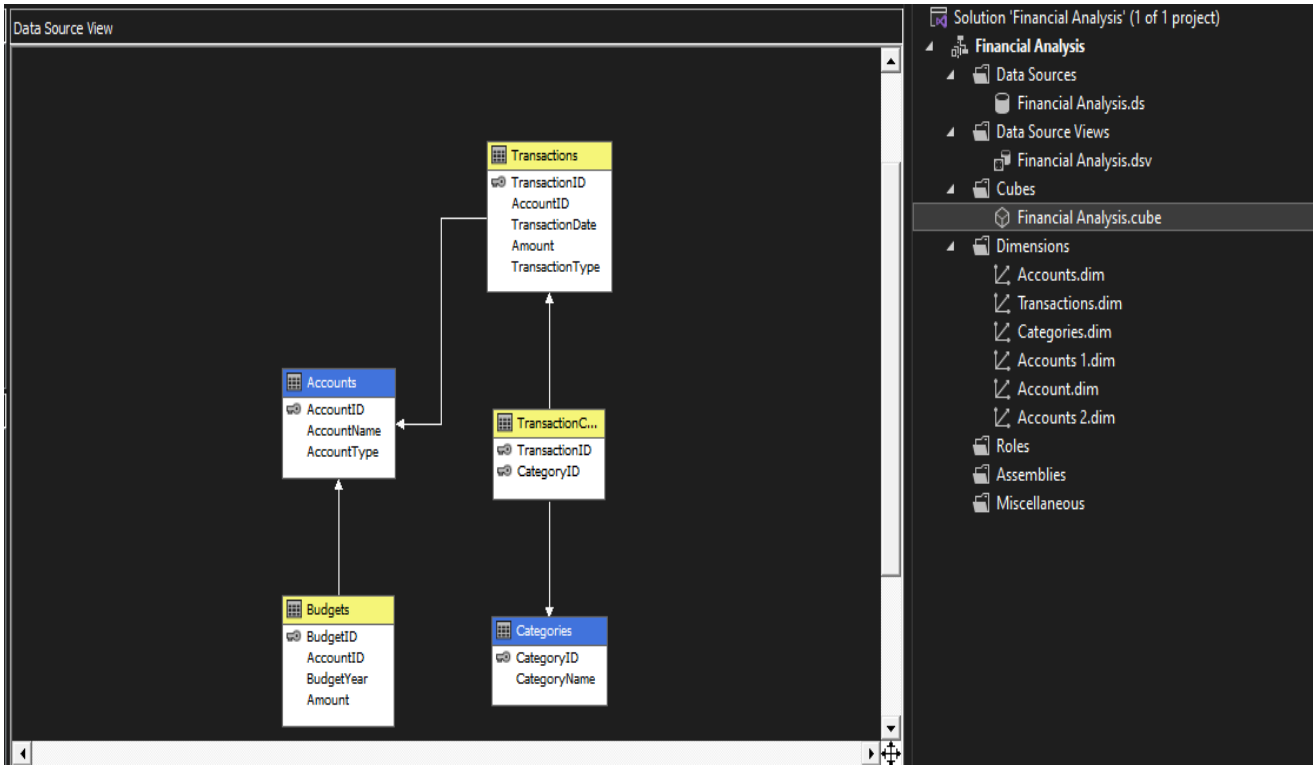
14. Choose Measures from the List which you want to place in your Cube --> Click



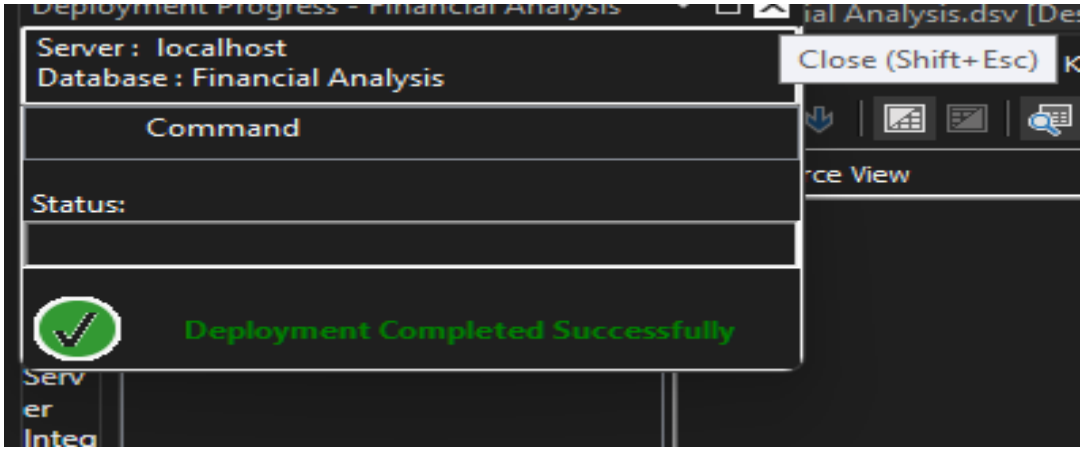
15. Select All Dimensions here which are associated with your Fact Table-> Click



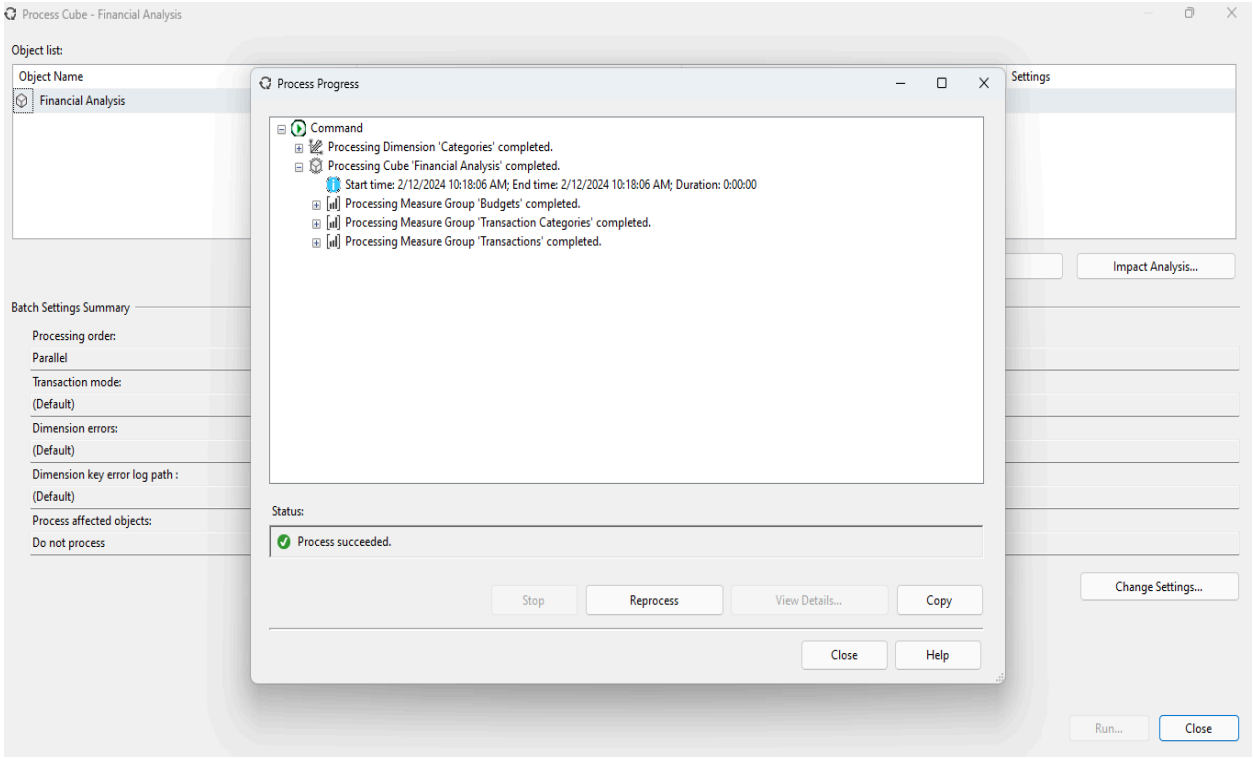
16. Now your Cube is ready, you can see the newly created cube and dimensions added in your solution explorer.



17. In Solution Explorer, right click on Project Name) -- > Click Deploy. Once Deployment will finish, you can see the message Deployment Completed in deployment Properties window.

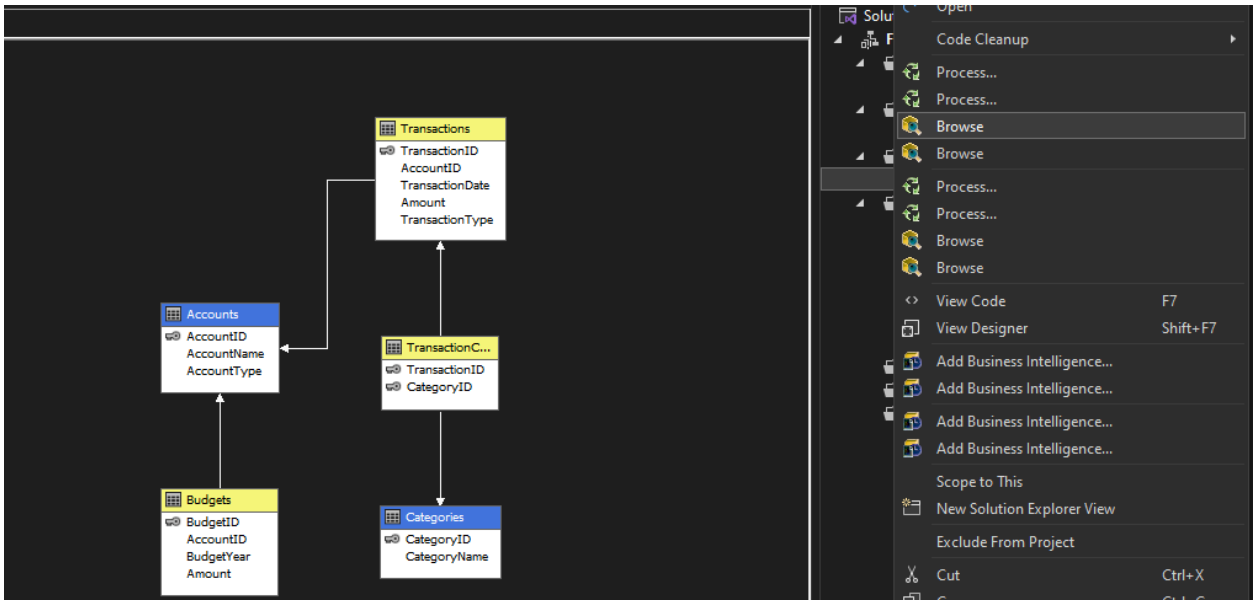


18. In Solution Explorer, right click on Project Name -- > Click Process. Click on Run button to process the Cube.
19.Once processing is complete, you can see Status as Process Succeeded -->Click Close to close both the open windows for processing one after the other.



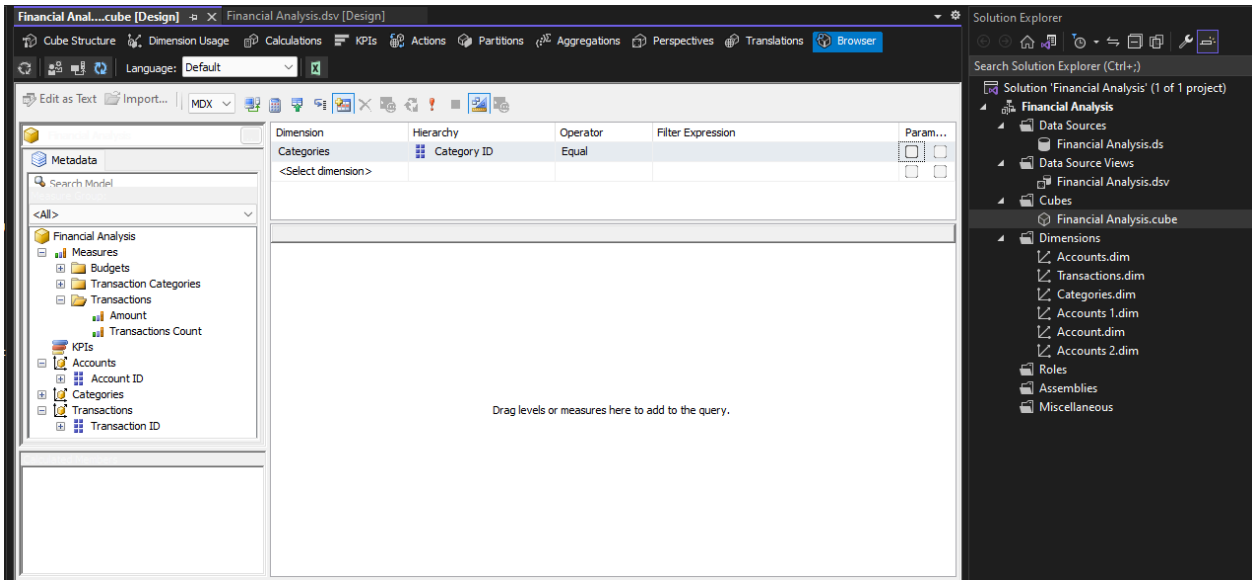
20. Browse the Cube for Analysis:

In Solution Explorer, right click on Cube Name (eWalletAnalyticalCube) -- > Click Browse



21. OLAP Querying using Cube:

Drag and drop measures into Detail fields, & Drag and Drop Dimension Attributes in Row Field or Column fields. Then click on "Execute query…". This generates the following MDX Query and displays the output:

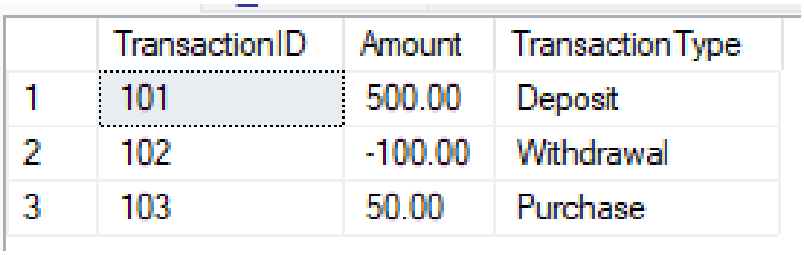


1. Regular Query:

Retrieve all transactions with their amounts and types.

SELECT TransactionID, Amount, TransactionType

FROM Transactions;

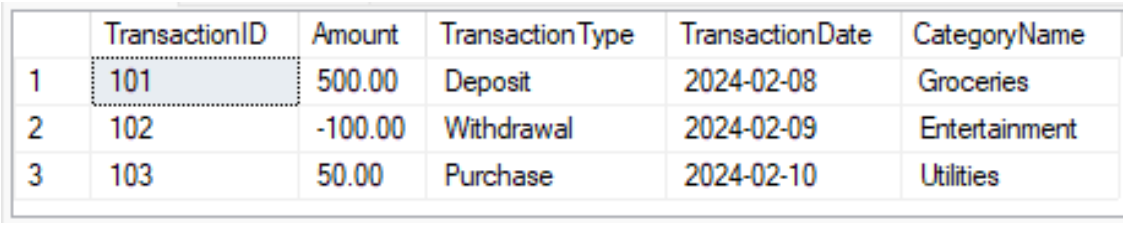


	TransactionID	Amount	TransactionType
1	101	500.00	Deposit
2	102	-100.00	Withdrawal
3	103	50.00	Purchase

2. Drill Down

Drilling down adds more detail to the analysis. Here, we drill down into transactions by including transaction date and category names.

```
SELECT T.TransactionID, T.Amount, T.TransactionType, T.TransactionDate,
C.CategoryName
FROM Transactions T
JOIN TransactionCategories TC ON T.TransactionID = TC.TransactionID
JOIN Categories C ON TC.CategoryID = C.CategoryID;
```



	TransactionID	Amount	TransactionType	TransactionDate	CategoryName
1	101	500.00	Deposit	2024-02-08	Groceries
2	102	-100.00	Withdrawal	2024-02-09	Entertainment
3	103	50.00	Purchase	2024-02-10	Utilities

3. Roll Up

Rolling up aggregates data to a higher level of hierarchy. Here, we aggregate transactions by month and account type.

```
SELECT YEAR(T.TransactionDate) AS Year, MONTH(T.TransactionDate) AS Month,
A.AccountType, SUM(T.Amount) AS TotalAmount
FROM Transactions T
JOIN Accounts A ON T.AccountID = A.AccountID
GROUP BY YEAR(T.TransactionDate), MONTH(T.TransactionDate), A.AccountType;
```

	Year	Month	AccountType	TotalAmount
1	2024	2	Checking	-100.00
2	2024	2	Credit	50.00
3	2024	2	Savings	500.00

4. Pivot

A pivot operation changes the way aggregation is viewed. Here, we pivot transaction amounts by transaction type for each account.

```
SELECT AccountID,
    SUM(CASE WHEN TransactionType = 'Deposit' THEN Amount ELSE 0 END) AS TotalDeposits,
    SUM(CASE WHEN TransactionType = 'Withdrawal' THEN Amount ELSE 0 END) AS TotalWithdrawals,
    SUM(CASE WHEN TransactionType = 'Purchase' THEN Amount ELSE 0 END) AS TotalPurchases
FROM Transactions
GROUP BY AccountID;
```

	AccountID	TotalDeposits	TotalWithdrawals	TotalPurchases
1	1	500.00	0.00	0.00
2	2	0.00	-100.00	0.00
3	3	0.00	0.00	50.00

5. Slice

Slicing filters data along one dimension. Here, we slice the data for transactions made in February 2024.

```
SELECT TransactionID, AccountID, Amount, TransactionType
FROM Transactions
WHERE TransactionDate BETWEEN '2024-02-01' AND '2024-02-28';
```

	TransactionID	AccountID	Amount	TransactionType
1	101	1	500.00	Deposit
2	102	2	-100.00	Withdrawal
3	103	3	50.00	Purchase

6. Dice

Dicing filters data on two or more dimensions. Here, we dice the data for transactions of type 'Deposit' in the year 2024, with amounts greater than $100.

```
SELECT TransactionID, AccountID, TransactionDate, Amount
FROM Transactions
WHERE TransactionType = 'Deposit' AND YEAR(TransactionDate) = 2024 AND Amount > 100;
```

	TransactionID	AccountID	TransactionDate	Amount
1	101	1	2024-02-08	500.00

Conclusion: Thus we have learnt about basics of data warehousing, and design schemas. We also learnt about.ETL and OLAP operations and applied them on our data warehouse as a case study on Finance.