Graduation Project Report

Anti-Money Laundering (AML) Detection System

Prepared By

Ahmed Osama

Nahla Mohamed

Mona Osama

Rowan Amr

Adham Medhat

Mahmoud Magdy

Supervised By

Eng. Ahmed Essam Azab

Overview

Our project focuses on developing a predictive analytics system to identify potential money laundering activities and suspects in global financial transactions. By analyzing transaction patterns and behavioral data using some tools such as SQL server for Data warehousing and Power BI for visualization.

ETL Process for AML Data Warehouse

Purpose of Data Warehousing for the AML Project

The purpose of data warehousing in this anti-money laundering project is to create a centralized repository for storing, managing, and analyzing large volumes of transaction data from various sources. This enables:

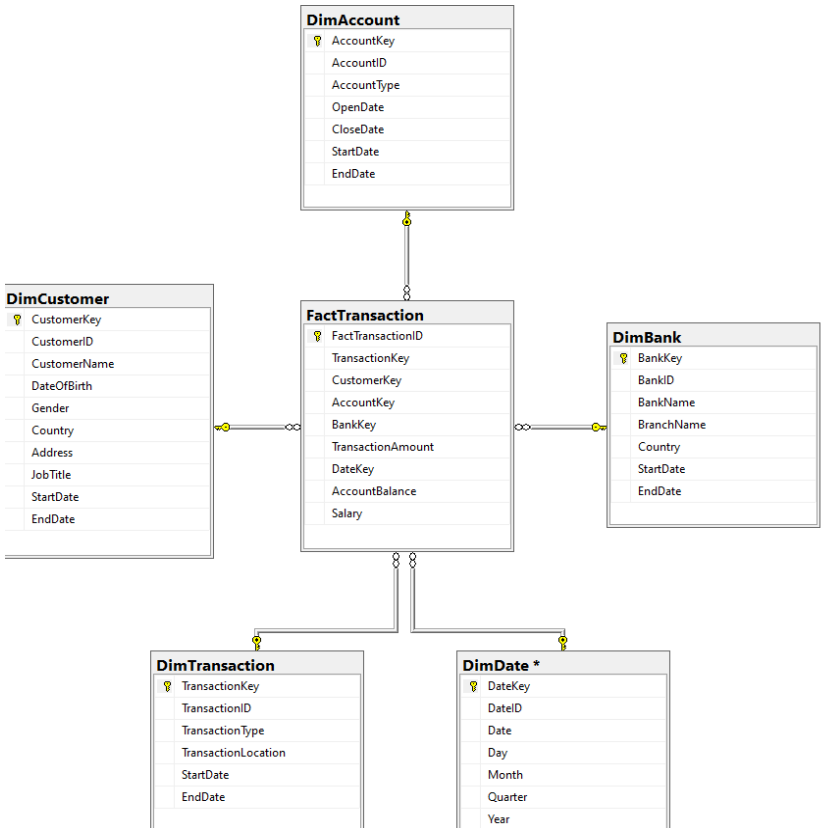
1. **Enhanced Data Integration:** Consolidating data from multiple banks and financial institutions allows for a comprehensive view of client transactions.
2. **Improved Query Performance:** The structured star schema design facilitates faster and more efficient querying, enabling analysts to quickly identify suspicious patterns and trends.
3. **Historical Analysis:** Storing historical transaction data supports longitudinal studies and trend analysis, essential for detecting evolving money laundering tactics.
4. **Data Quality and Consistency:** A data warehouse ensures high-quality, consistent data through rigorous preprocessing, making it reliable for compliance and reporting.
5. **Support for Advanced Analytics:** The warehouse serves as a foundation for implementing machine learning models and advanced analytics, helping to predict and identify potential money laundering activities effectively.

Overall, the data warehouse enhances the project's ability to detect and mitigate financial crime risks while ensuring regulatory compliance.

Star Schema

In a star schema designed for an Anti-Money Laundering (AML) project, the central fact table is connected to several dimension tables, facilitating efficient data analysis. Here's a breakdown of the potential structure:

- Fact Transactions (FactTransactionsID , CustomerID, TransactionAmount, ..)
- Dimensions_Tables (DimCustomer, DimTransaction, DimBank, ..) as shown



This star schema structure allows for efficient querying and reporting on transaction data, providing insights into potential money laundering activities. Analysts can easily join fact and dimension tables to explore relationships, identify suspicious patterns, and perform comprehensive analyses across different dimensions, such as time, client demographics, and transaction types.

Create AML Warehouse with Fact and Dimensions tables:

- **Customer Dimension**

```
CREATE TABLE DimCustomer (
    CustomerKey INT IDENTITY(1,1) PRIMARY KEY,
    CustomerID INT,
    CustomerName NVARCHAR(100),
    DateOfBirth DATE,
    Gender NVARCHAR(10),
    Country NVARCHAR(50),
    Address NVARCHAR(255),
    JobTitle NVARCHAR(50),
    StartDate DATETIME DEFAULT GetDate() NOT NULL,
    EndDate DATETIME DEFAULT NULL
);
```

- **Transaction Dimension**

```
CREATE TABLE DimTransaction (
    TransactionKey INT IDENTITY(1,1) PRIMARY KEY,
    TransactionID INT,
    TransactionType NVARCHAR(50),
    TransactionLocation NVARCHAR(100),
    StartDate DATETIME DEFAULT GetDate() NOT NULL,
    EndDate DATETIME DEFAULT NULL
);
```

- **Account Dimension**

```
CREATE TABLE DimAccount (
    AccountKey INT IDENTITY(1,1) PRIMARY KEY,
    AccountID INT,
    AccountType NVARCHAR(50),
    OpenDate DATE,
    CloseDate DATE,
    StartDate DATETIME DEFAULT GetDate() NOT NULL,
    EndDate DATETIME DEFAULT NULL
);
```

- **Bank Dimension**

```
CREATE TABLE DimBank (
    BankKey INT IDENTITY(1,1) PRIMARY KEY,
    BankID INT,
    BankName NVARCHAR(100),
    BranchName NVARCHAR(100),
    Country NVARCHAR(50),
    StartDate DATETIME DEFAULT GetDate() NOT NULL,
    EndDate DATETIME DEFAULT NULL
);
```

- **Date Dimension**

```sql
CREATE TABLE DimDate (
    DateKey INT IDENTITY(1,1) PRIMARY KEY,
    DateID INT,
    Date DATE,
    Day INT,
    Month INT,
    Quarter INT,
    Year INT
);
```

- **Fact Table**

```sql
CREATE TABLE FactTransaction (
    FactTransactionID INT IDENTITY(1,1) PRIMARY KEY,
    TransactionKey INT FOREIGN KEY REFERENCES DimTransaction(TransactionKey),
    CustomerKey INT FOREIGN KEY REFERENCES DimCustomer(CustomerKey),
    AccountKey INT FOREIGN KEY REFERENCES DimAccount(AccountKey),
    BankKey INT FOREIGN KEY REFERENCES DimBank(BankKey),
    TransactionAmount DECIMAL(18, 2),
    DateKey INT FOREIGN KEY REFERENCES DimDate(DateKey),
    AccountBalance DECIMAL(18, 2),
    Salary DECIMAL(18, 2)
);
```

ETL Process Overview for the AML Project:

The ETL (Extract, Transform, Load) process is crucial for preparing data for analysis in the Anti-Money Laundering (AML) project. Here's an overview of each step:

1. **Extract**

- **Data Generated:**
  - **Bank Transactions:** Comprehensive records of all transactions.
  - **Transaction Amount:** The value associated with each transaction.
  - **Account ID:** Unique identifier for each bank account involved in the transactions.
  - **Country of Transaction:** Geographic location where the transaction occurred.
  - **Address:** Specific address associated with the transaction (if available).
  - **Date of Transaction:** Timestamp indicating when each transaction took place.
  - **Transaction Type:** Classification of the transaction (e.g., Deposit, Withdrawal, Transfer).

- **Data Generation Process:**

- We used Python to write scripts that generated synthetic data, which mimicked real-world transaction patterns.
- Libraries we used: Pandas for creating and manipulating structured datasets efficiently, Faker for generating realistic data such as (names, addresses, … etc), allowing us to simulate customer profiles and their activities.

- **Sample of Code:**

```python
import pandas as pd
from faker import Faker
from datetime import datetime

fake = Faker()

def generate_customer_data(num_customers):
    customer_data = []
    global countries, sanctioned_countries

    job_titles = ["Engineer", "Doctor", "Teacher", "Software Developer", "Data Scientist",
                  "Banker", "Consultant", "Artist", "Freelancer", "Manager", "Sales Executive"]
    countries = ["USA", "Germany", "France", "Japan", "India", "Brazil", "UK", "Canada"]
    sanctioned_countries = ["Russia", "China", "Iraq", "Lebanon", "North Korea", "Iran",
                            "Syria", "Sudan", "Cuba", "Myanmar", "Venezuela", "Yemen",
                            "Libya", "Somalia", "Pakistan", "Panama", "Afghanistan"]
    customer_account_types = ["Personal", "Business", "VIP", "Basic"]
    account_types = ["Loan", "Investment", "Saving", "Retirement"]

    for i in range(1000, num_customers+1000):
        customer_id = f"CUST{i:06d}"
        account_id = f"ACC{i:06d}"
        customer_name = fake.name()
        dob = fake.date_of_birth(minimum_age=18, maximum_age=60)
        country = random.choice(countries)
        address = fake.address()
        gender = random.choice(["Male", "Female"])
        job_title = random.choice(job_titles)
        AccountType = random.choice(account_types)
        CustomerAccount = random.choice(customer_account_types)

        if job_title in ["Engineer", "Doctor", "Software Developer", "Data Scientist"]:
            salary = round(random.uniform(50000, 200000), 2)
        elif job_title in ["Teacher", "Artist", "Freelancer"]:
            salary = round(random.uniform(30000, 80000), 2)
        else:
            salary = round(random.uniform(40000, 150000), 2)

        customer_data.append([customer_id, account_id, customer_name, dob, country, address, gender, job_title, salary, AccountType, CustomerAccount])
    customer_df = pd.DataFrame(customer_data, columns=[
        "CustomerID", "AccountID", "CustomerName", "DateOfBirth", "Country", "Address", "Gender", "JobTitle", "Salary", "AccountType", "CustomerAccount"
    ])
    return customer_df
```

Sample of Data:

CustomerID	AccountID	CustomerName	DateOfBirth	Country	Address	Gender	JobTitle	Salary	CustomerAccount	AccountType
CUST001015	ACC001015	Vincent Cook	1993-07-02 00:00:00.000	Germany	1396 Castro Springs Ericbury, AK 98650	Male	Engineer	145912.71	Basic	Saving
CUST001016	ACC001016	Charles Stanley	2004-03-22 00:00:00.000	Germany	90414 David Union Suite 375 New Darrenstad, OK 9...	Male	Doctor	169793.43	Personal	Saving
CUST001017	ACC001017	Mr. Matthew Jones	1975-03-05 00:00:00.000	France	70213 Gabriel Course New Jasontown, SD 39457	Male	Consultant	123028.24	Personal	Saving
CUST001018	ACC001018	Tonya Davis	1973-07-23 00:00:00.000	Canada	57156 Norma Islands Apt. 579 Tranport, DE 43516	Female	Sales Executive	122114.57	Personal	Saving
CUST001019	ACC001019	John Berg	1971-03-13 00:00:00.000	Canada	1896 Aaron Isle Apt. 446 Wallberg, MN 91842	Male	Doctor	155743.56	Basic	Investment

TransactionID	CustomerID	AccountID	TransactionDate	TransactionType	TransactionAmount	TransactionLocation	CardType	BankID
TRANS164751	CUST009187	ACC009187	2024-07-27 18:46:58.000	Withdrawal	26840.44	Pakistan	Debit	BANK0058
TRANS164752	CUST009187	ACC009187	2024-08-30 15:59:45.000	Payment	86705.41	Iraq	Debit	BANK0043
TRANS164753	CUST009187	ACC009187	2024-09-28 03:49:29.000	Payment	738931	Pakistan	Prepaid	BANK0047
TRANS164754	CUST009187	ACC009187	2024-06-27 09:16:00.000	Withdrawal	711521.89	Somalia	Debit	BANK0009
TRANS164755	CUST009187	ACC009187	2024-10-07 12:02:07.000	Payment	47895.78	Afghanistan	Prepaid	BANK0005

2. Transform

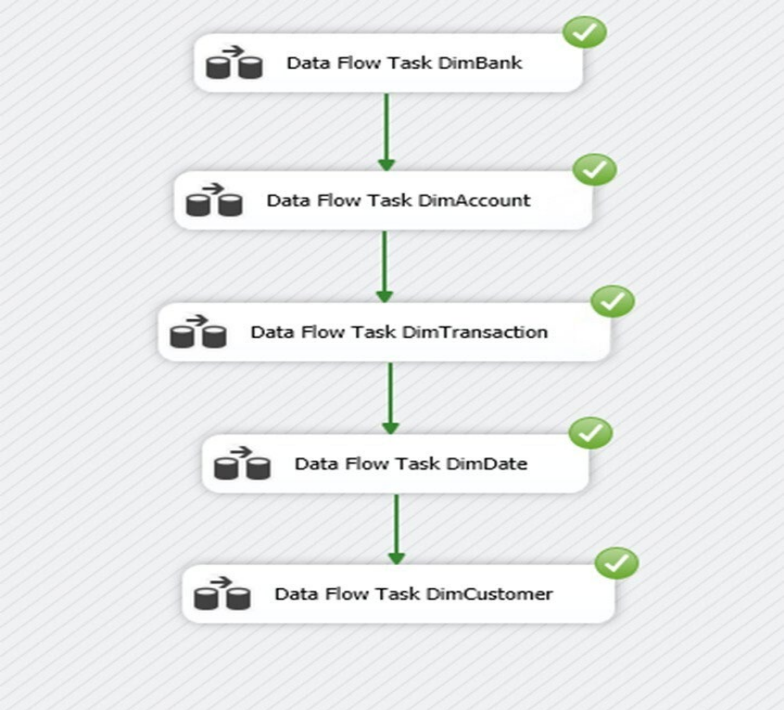
Objective: Cleanse and prepare the extracted data for integration into the data warehouse.

- **Data Preparation Using SSIS:**
  - **Derived Columns:**
    - **ID Transformation:** Modify account IDs by removing prefixes (e.g., change "ACC1102" to "1102") for standardization.
    - **Null Value Removal:** Eliminate rows with null values in critical fields to ensure data integrity.
    - **Date Formatting:** Convert date fields to a standardized format that includes day, month, and year, …etc.
  - **Data Conversion:**
    - Ensure all data types are consistent and appropriate for analysis
  - **Slowly Changing Dimensions (SCD):**
    - Implement SCD to track changes in dimension data over time, ensuring historical accuracy.
  - **Merge:**
    - Combine data from various sources into a unified dataset, linking transaction records to client and account details.
  - **Sort:**
    - Organize the data by key attributes (e.g., transaction date or account ID) to optimize loading and querying in the data warehouse.
- **Utilizing SSIS:**
  - Leverage SQL Server Integration Services (SSIS) to automate and streamline these transformation processes, ensuring efficiency and accuracy.

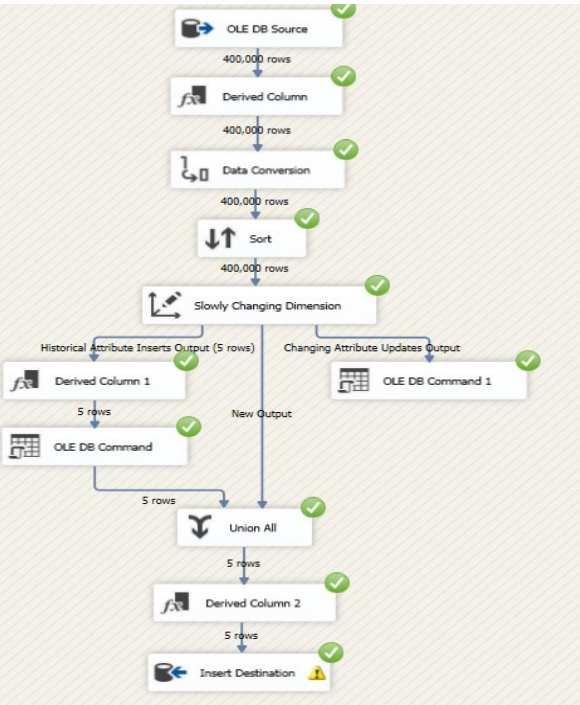
- # Slowly Changing Dimension

A Slowly Changing Dimension (SCD) is a dimension that stores and manages both current and historical data over time in a data warehouse. It is considered and implemented as one of the most critical ETL tasks in tracking the history of dimension records.

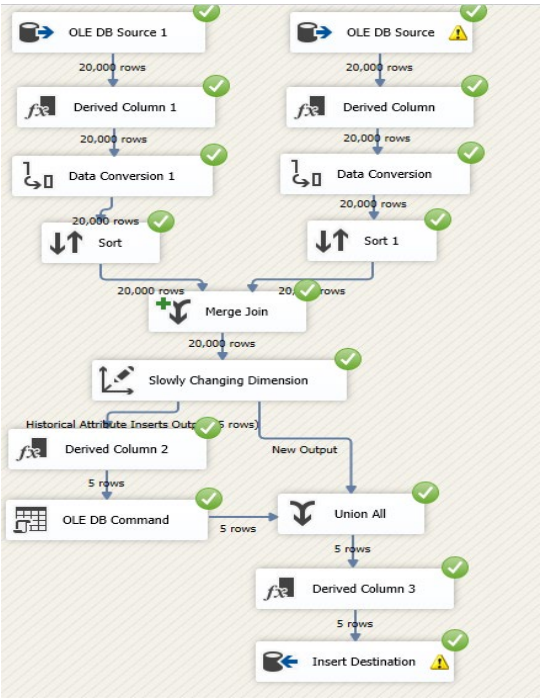
- Create Data Flow Task for Each Dimension:



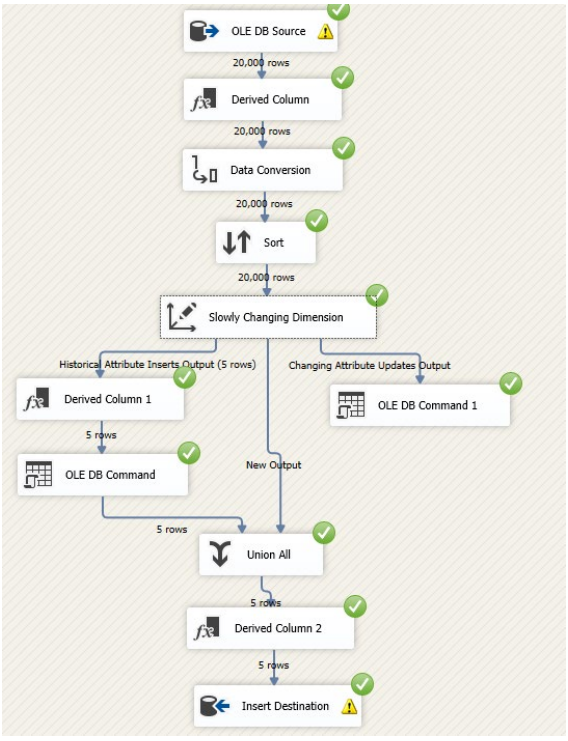
A. Create Data Flow Task for Transaction Dimension



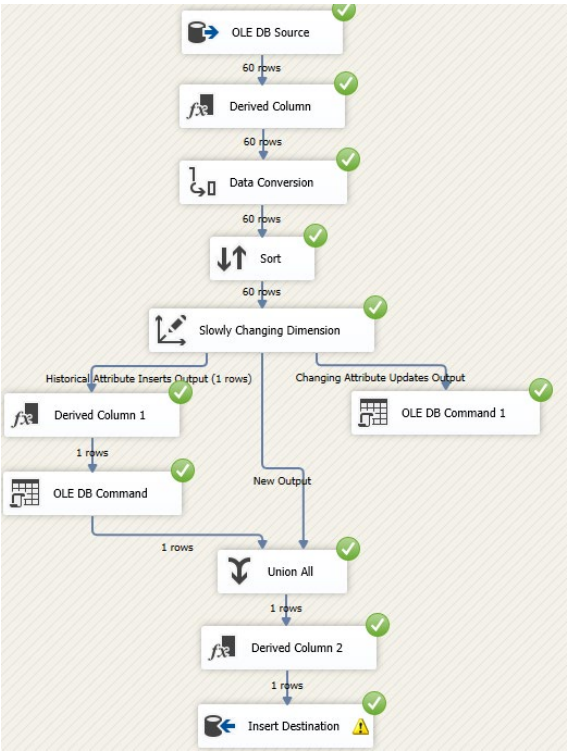
B. Create Data Flow Task for Account Dimension



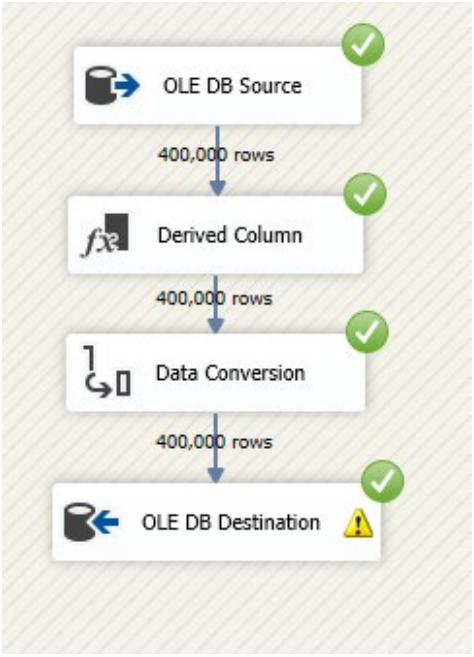
C. Create Data Flow Task for Customer Dimension



D. Create Data Flow Task for Bank Dimension



E. Create Data Flow Task for Date Dimension



Results:

- For Transaction Dimension:

TransactionKey	TransactionID	TransactionType	TransactionLocation	StartDate	EndDate	CardType
843901	164787	Deposit	China	2024-10-20 11:58:26.980	2024-10-20 22:26:50.000	Credit Card
843931	164817	Payment	Russia	2024-10-20 11:58:26.987	2024-10-20 22:26:50.000	Prepaid
843985	164871	Withdrawal	Cuba	2024-10-20 11:58:26.993	2024-10-20 22:26:50.000	Prepaid
843994	164880	Deposit	Iran	2024-10-20 11:58:26.997	2024-10-20 22:26:50.000	Credit Card
844043	164929	Deposit	China	2024-10-20 11:58:27.003	2024-10-20 22:26:50.000	Prepaid
1080114	164787	Deposit	Russia	2024-10-21 00:55:05.200	NULL	Credit Card
1080115	164817	Payment	Panama	2024-10-21 00:55:05.200	NULL	Prepaid
1080116	164871	Withdrawal	Venezuela	2024-10-21 00:55:05.203	NULL	Prepaid
1080117	164880	Deposit	Cuba	2024-10-21 00:55:05.203	NULL	Credit Card
1080118	164929	Deposit	North Korea	2024-10-21 00:55:05.203	NULL	Prepaid

- For Customer Dimension:

CustomerKey	CustomerID	CustomerName	Gender	Country	Address	JobTitle	StartDate	EndDate
45345	6343	Elizabeth Rivera	Male	USA	02262 Taylor Branch Apt. 733 North Rachelbury, VA...	Banker	2024-10-20 11:41:53.000	2024-10-20 22:22:45.000
45360	6358	Jason Smith	Male	Japan	20136 Veronica Wall Suite 593 South Drew, HI 20336	Software Developer	2024-10-20 11:41:53.000	2024-10-20 22:22:45.000
45376	6374	George Stanley	Female	France	129 Moore Centers Suite 735 Jonesstad, ID 13057	Sales Executive	2024-10-20 11:41:53.000	2024-10-20 22:22:45.000
45389	6387	Lindsay Buch...	Female	France	583 Bates Key Apt. 922 Christinahaven, AR 25195	Freelancer	2024-10-20 11:41:53.000	2024-10-20 22:22:45.000
45415	6413	Nicholas Rob...	Male	Germ...	05761 Evans Light Apt. 206 Port Jessica, MO 09705	Software Developer	2024-10-20 11:41:53.000	2024-10-20 22:22:45.000
60002	6343	Elizabeth Rivera	Male	USA	24416 Diana Mission Apt. 073 Port Christystad, KS 8...	Banker	2024-10-20 22:22:45.000	NULL
60003	6358	Jason Smith	Male	Japan	881 Walker Green Suite 842 East Rachel, GU 58266	Software Developer	2024-10-20 22:22:45.000	NULL
60004	6374	George Stanley	Female	France	583 Weber Ramp West Sarah, IL 88268	Sales Executive	2024-10-20 22:22:45.000	NULL
60005	6387	Lindsay Buch...	Female	France	3522 Joseph Fort Maloneburgh, NH 67270	Freelancer	2024-10-20 22:22:45.000	NULL
60006	6413	Nicholas Rob...	Male	Germ...	97925 Sanchez Ranch North Davidfurt, MS 60029	Software Developer	2024-10-20 22:22:45.000	NULL

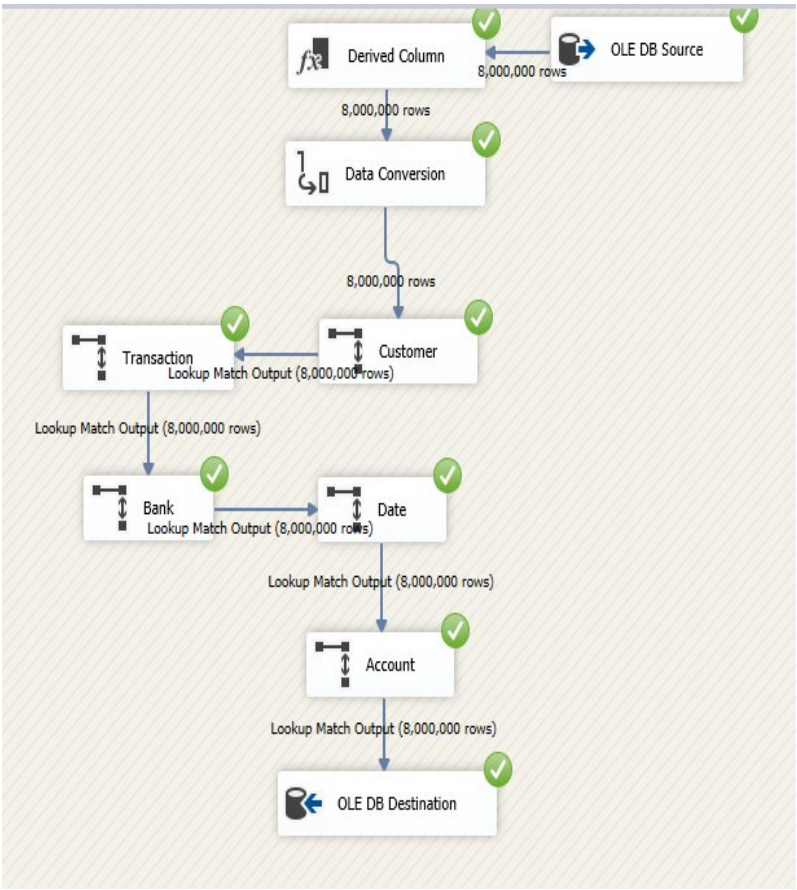
- For Bank Dimension:

BankKey	BankID	BankName	BranchName	Country	StartDate	EndDate
77	16	Townsend-Good	South Andresstad	Russia	2024-10-20 11:40:55.000	2024-10-20 22:21:48.000
79	18	Perry, Vaughn and Smith	Danielberg	Sweden	2024-10-20 11:40:55.000	2024-10-20 22:21:48.000
88	27	Mcneil PLC	Cherylfort	China	2024-10-20 11:40:55.000	2024-10-20 22:21:48.000
93	32	Ortiz LLC	Davidmouth	Russia	2024-10-20 11:40:55.000	2024-10-20 22:21:48.000
117	56	Mason-Hanson	Jilltown	Russia	2024-10-20 11:40:55.000	2024-10-20 22:21:48.000
122	16	Townsend-Good	East Gregory	Russia	2024-10-20 22:21:48.000	NULL
123	18	Perry, Vaughn and Smith	South Andresstad	Sweden	2024-10-20 22:21:48.000	NULL
124	27	Mcneil PLC	Lewisport	China	2024-10-20 22:21:48.000	NULL
125	32	Ortiz LLC	Cherylfort	Russia	2024-10-20 22:21:48.000	NULL
126	56	Mason-Hanson	Beanberg	Russia	2024-10-20 22:21:48.000	NULL

- For Account Dimension:

AccountKey	AccountID	AccountType	StartDate	EndDate
100128	1121	Retirement	2024-10-20 11:44:06.607	2024-10-20 22:24:07.000
100164	1157	Loan	2024-10-20 11:44:06.613	2024-10-20 22:24:07.000
100176	1169	Retirement	2024-10-20 11:44:06.613	2024-10-20 22:24:07.000
100184	1177	Investment	2024-10-20 11:44:06.617	2024-10-20 22:24:07.000
100197	1190	Saving	2024-10-20 11:44:06.617	2024-10-20 22:24:07.000
120007	1121	Saving	2024-10-20 22:25:26.167	NULL
120008	1157	Investment	2024-10-20 22:25:26.167	NULL
120009	1169	Loan	2024-10-20 22:25:26.167	NULL
120010	1177	Saving	2024-10-20 22:25:26.167	NULL
120011	1190	Investment	2024-10-20 22:25:26.167	NULL

- Create Data Flow Task for Fact Transaction:



Results:

FactTransactionID	TransactionKey	CustomerKey	AccountKey	BankKey	TransactionAmount	DateKey	AccountBalance	Salary
343659	1058442	58918	118923	64	8666.45	688299	32071.05	50094.67
343660	1058442	58918	118923	64	15967.10	688299	55915.08	50094.67
343661	1058455	58919	118924	64	31673.14	688312	92123.12	198337.49
343662	1058455	58919	118924	64	40915.49	688312	312823.20	198337.49
343663	1058484	58920	118925	64	26603.96	688341	-284700.84	68917.48
343664	1058614	58927	118932	64	37114.19	688471	-321252.04	78780.44
343665	1058614	58927	118932	64	12419.08	688471	-74405.42	78780.44
343666	1058648	58928	118933	64	40701.70	688505	247310.28	40956.89
343667	1058648	58928	118933	64	41646.35	688505	288011.98	40956.89
343668	1058705	58931	118936	64	30216.97	688562	-477408.72	165469.56
343669	1058709	58931	118936	64	41453.04	688566	-98886.67	165469.56
343670	1058709	58931	118936	64	33906.00	688566	-57433.63	165469.56
343671	1058721	58932	118937	64	14886.88	688578	-15304.17	178106.99
343672	1058721	58932	118937	64	21236.51	688578	-417.29	178106.99
343673	1058721	58932	118937	64	24326.07	688578	263795.69	178106.99
343674	1058742	58933	118938	64	14272.40	688599	-218590.63	81148.74
343675	1058778	58935	118940	64	40277.35	688635	-232661.07	130977.86
343676	1058778	58935	118940	64	24208.01	688635	135401.99	130977.86
343677	1058799	58936	118941	64	41142.28	688656	264086.03	39702.43
343678	1058820	58937	118942	64	47210.52	688677	-141159.39	49436.08
343679	1058820	58937	118942	64	21280.87	688677	-47548.81	49436.08
343680	1058820	58937	118942	64	20749.07	688677	-26267.94	49436.08

This transformation phase ensures the data is clean, standardized, and ready for effective analysis in the data warehouse.

3. Load

Objective: Store the transformed data in the data warehouse.

- Load data into the structured star schema designed for the AML project, populating fact and dimension tables.

Summary:

The ETL process ensures that the data used in the AML project is accurate, consistent, and ready for analysis. By systematically extracting, transforming, and loading data, the project can leverage advanced analytics to identify and mitigate money laundering risks effectively.

SQL Scripts and Queries:

This query retrieves account transactions that have an amount greater than 30,000, and have occurred within the 7 days:

```
SELECT f.AccountKey, d.Date, f.TransactionAmount
FROM FactTransaction f
Join DimDate d
on d.DateKey = f.DateKey
WHERE TransactionAmount > 30000.00
AND DATEDIFF(day, d.Date, GETDATE()) <= 7
ORDER BY f.TransactionAmount DESC;
```

Result of Query:

AccountKey	Date	TransactionAmount
118256	2024-10-16 01:41:24.000	49999.68
118256	2024-10-15 01:41:24.000	49999.68
118256	2024-10-16 03:41:24.000	49999.68
118731	2024-10-15 03:05:33.000	49998.34
118731	2024-10-16 03:05:33.000	49998.34
115117	2024-10-15 09:47:02.000	49989.66
115117	2024-10-15 04:47:02.000	49989.66
115161	2024-10-15 02:18:15.000	49974.50
112479	2024-10-16 08:55:40.000	49974.01
112479	2024-10-16 14:55:40.000	49974.01
112479	2024-10-15 08:55:40.000	49974.01
114140	2024-10-18 20:52:38.000	49969.76
114140	2024-10-17 10:52:38.000	49969.76
114140	2024-10-16 01:52:38.000	49969.76
114140	2024-10-17 13:52:38.000	49969.76

This query retrieves the total sum of transaction amounts greater than 40,000, and the count of those transactions. It groups the results by **AccountKey** and only includes accounts that have more than 3 such transactions.

```
SELECT AccountKey, SUM(TransactionAmount) AS TotalAmount, COUNT(*) AS TransactionCount
FROM FactTransaction
WHERE TransactionAmount > 40000.00
GROUP BY AccountKey
HAVING COUNT(*) > 3;
```

Results of Query:

AccountKey	TotalAmount	TransactionCount
108639	5282575.00	120
112132	8031620.80	180
112139	4208192.60	100
109357	3722984.00	80
108904	2761737.00	60
109097	7217782.60	160
109290	6133835.60	140
110874	5440098.60	120
110062	2638841.20	60
113176	5377757.00	120
111087	4551851.00	100
112411	4496769.40	100
113909	1838148.00	40
110092	3657450.80	80
113535	4537098.80	100

This query retrieves the AccountKey, the total amount of deposits, and the total amount of withdrawals for each account by summing the TransactionAmount based on the transaction type (either "Deposit" or "Withdrawal"). It filters to include only accounts where the total amount of withdrawals exceeds 40,000.

```
SELECT f.AccountKey,
    SUM(CASE WHEN t.TransactionType = 'Deposit' THEN f.TransactionAmount ELSE 0 END) AS TotalDeposits,
    SUM(CASE WHEN t.TransactionType = 'Withdrawal' THEN f.TransactionAmount ELSE 0 END) AS TotalWithdrawals
FROM FactTransaction f
Join DimTransaction t
on f.TransactionKey = t.TransactionKey
GROUP BY f.AccountKey
HAVING SUM(CASE WHEN t.TransactionType = 'Withdrawal' THEN f.TransactionAmount ELSE 0 END) > 40000.00;
```

Results of Query:

AccountKey	TotalDeposits	TotalWithdrawals
101725	2105415.75	2947582.05
106263	1628495.16	2714158.60
104266	1715129.04	1715129.04
106442	1670345.10	3897471.90
104631	1712564.97	2854274.95
102276	2666977.75	2133582.20
103799	1802872.96	1352154.72
106714	2391897.60	2391897.60
103892	3604531.63	1544799.27
103348	2520987.78	840329.26
105150	1570349.76	1046899.84
103501	2542009.25	3558812.95
107853	3034875.06	1011625.02
107879	2597318.40	4155709.44
105898	2172470.08	1086235.04

This SQL query identifies suspicious transactions by grouping transactions based on AccountKey, summing the TransactionAmount, and counting the number of transactions for each account. It filters for transactions above 40,000 and shows only accounts where the total sum of transactions exceeds 500,000.

```
SELECT AccountKey, SUM(TransactionAmount) AS SuspiciousAmount, COUNT(TransactionKey) AS NumberOfTransactions
FROM FactTransaction
WHERE TransactionAmount > 40000.00
GROUP BY AccountKey
HAVING SUM(TransactionAmount) > 500000.00;
```

Results of Query:

AccountKey	SuspiciousAmount	NumberOfTransactions
117649	3603370.20	80
117622	5277821.60	120
116815	4390162.20	100
118226	2823466.00	60
117750	4581952.40	100
116904	3672038.80	80
118315	6363184.00	140
118414	5772626.20	120
117971	4588668.80	100
118196	4594924.80	100
117857	5434760.00	120
118805	3614741.00	80
119233	4510883.20	100
118669	2632286.00	60
118107	865952.60	20

This query retrieves transaction details. It selects TransactionAmount for transactions that occurred between 2 days to track the accounts that applied suspicious transactions.

```
SELECT f.AccountKey, d.Date, f.TransactionAmount
FROM FactTransaction f
Join DimDate d
ON d.DateKey = f.DateKey
WHERE d.Date BETWEEN '2024-03-11 21:54:13.000' AND '2024-03-12 21:54:13.000'
ORDER BY f.AccountKey, d.Date;
```

Results of Query:

AccountKey	Date	TransactionAmount
103250	2024-03-12 17:56:44.000	34042.63
103250	2024-03-12 17:56:44.000	12063.40
103250	2024-03-12 17:56:44.000	44760.55
103250	2024-03-12 17:56:44.000	889.02
103250	2024-03-12 17:56:44.000	27767.19
103250	2024-03-12 17:56:44.000	36642.93
103250	2024-03-12 17:56:44.000	6886.39
103250	2024-03-12 17:56:44.000	519.66
103279	2024-03-12 17:51:15.000	25587.89
103279	2024-03-12 17:51:15.000	13515.32
103279	2024-03-12 17:51:15.000	12076.60
103279	2024-03-12 17:51:15.000	38159.89
103279	2024-03-12 17:51:15.000	28902.49
103279	2024-03-12 17:51:15.000	35644.77
103279	2024-03-12 17:51:15.000	32853.59

This query identifies repeated transactions for each account within a 24-hour period.

```
WITH RepeatedTransactions AS (
    SELECT f.AccountKey, d.Date, f.TransactionAmount,
           LEAD(d.Date) OVER (PARTITION BY f.AccountKey ORDER BY d.Date) AS NextTransactionDate
    FROM FactTransaction f
    Join DimDate d
    on d.DateKey = f.DateKey
)
SELECT AccountKey, Date, NextTransactionDate, DATEDIFF(hour, Date, NextTransactionDate) AS HoursBetweenTransactions
FROM RepeatedTransactions
WHERE DATEDIFF(hour, Date, NextTransactionDate) < 24;
```

Result of Query:

AccountKey	Date	NextTransactionDate	HoursBetweenTransactions
108210	2024-07-23 19:33:03.000	2024-07-23 22:33:03.000	3
108210	2024-07-23 22:33:03.000	2024-07-23 22:33:03.000	0

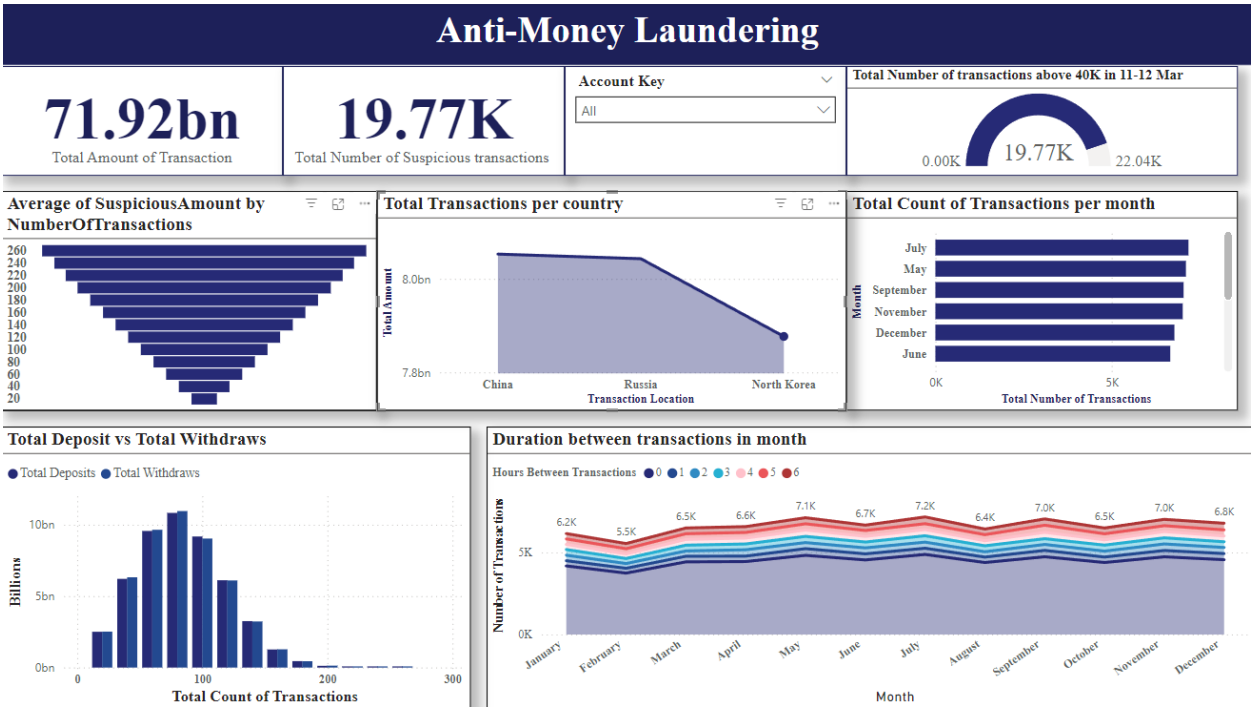
This query retrieves customer transactions from specific high-risk locations ('Russia', 'North Korea', 'China'). It counts the number of transactions and sums the total transaction amount for each customer in those locations, only including customers who have more than one transaction in these locations.

```
--Suspecious Countries
SELECT
    f.AccountKey,
    t.TransactionLocation,
    COUNT(f.TransactionKey) AS TransactionCount,
    SUM(f.TransactionAmount) AS TotalAmount
FROM
    FactTransaction f
JOIN DimTransaction t
ON t.TransactionKey = f.TransactionKey
WHERE
    t.TransactionLocation in ('Russia', 'North Korea', 'China')
GROUP BY
    f.AccountKey, t.TransactionLocation
having
    COUNT(f.TransactionKey) > 1
```

Results of Query:

AccountKey	TransactionLocation	TransactionCount	TotalAmount
112112	North Korea	20	517219.07
110726	Russia	60	1870487.37
102405	China	20	420438.79
108516	North Korea	20	552765.57
108537	Russia	20	525760.08
104069	China	40	958771.56
115213	China	20	415089.96
113050	Russia	20	437764.69
119143	Russia	60	1456794.99
105499	Russia	80	2194522.24
103879	China	40	994376.18
111024	North Korea	20	460858.70
109841	North Korea	80	2239098.60
105126	Russia	20	512118.44
104231	Russia	40	967089.10
105350	Russia	60	1326328.47
118863	North Korea	20	487951.90
113947	Russia	20	483536.46

Visualization



Key Insights:

The demonstrated dashboard presents key insights from our case study on the Anti-Money Laundering (AML) system. The data selected through our queries highlights the most suspicious transactions:

-The total value of transactions above 40,000 amounts to 71.92 billion.

-China has the highest value and number of transactions, exceeding 8 billion.

-North Korea, while one of the most suspicious countries, ranks lowest with over 7 billion in transactions.

-In terms of transaction durations, July has the longest average time between transactions, while February has the shortest.

-When comparing transaction types by number, withdrawals have a higher total value than deposits.

Conclusion

This project successfully implemented a framework for detecting potential money laundering activities using synthetic transaction data. By simulating various transactions, including those linked to sanctioned countries and unusual amounts, we analyzed patterns that indicate suspicious activities. The structured data and SQL queries provided valuable insights for identifying potential risks. This approach offers a foundation for improving AML detection and can be expanded with machine learning and real-time monitoring for enhanced accuracy in real-world applications.