ST1508 Practical AI CA1

By:
Luong Onn Kah Jovan - P2342898
Ng Qing Yang - P2308870
Rejey Ezekiel - P2348935
Li Yongjie - P2342377

SP BUYS

November 2024

CRISP-DM

Buisness Understanding

Data Understanding

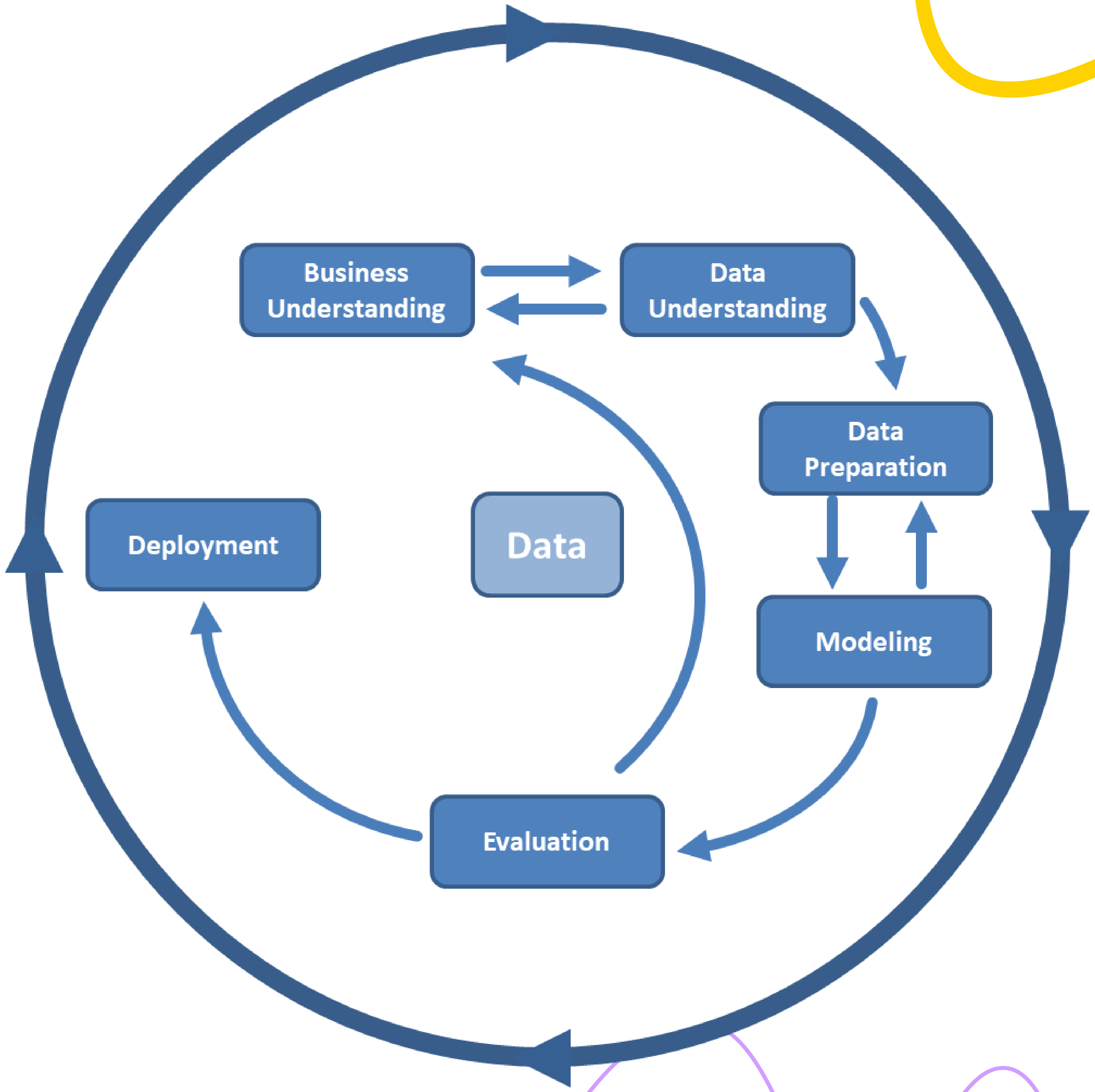
- EDA
- SQL

Data Preparation

- Feature Engineering

Tableau Dashboard

Additional



EDA (Explaratory Data Analysis)

Customer

- **Shape**: 2,195,916 rows, 9 columns
- Unique Values of Categorical Data
- **country_code**: ['PH' 'MY' 'BD' 'PK' 'TH'], 5 unique values
- **customer_id**: 1287588 unique values e.g.['phjr7fpu']
- **Missing Values**: 1240 na values in first_order_datetime
- **Duplicates**:
  - Exact duplicated rows: 34
  - No. of rows w duplicated composite keys customer_id & country_code: 908294 (excludes exact duplicates)
- **Inconsistencies**:
  - 427 Customer ID not in Label dataset.

```
## Use the describe method to get a summary of th
customer_df.describe().round()
```

Type to search

	country_code	customer_id
count	2195916	2195916
unique	5	1287588
top	MY	my2nvlmz
freq	839248	134

```
country_code                        0
customer_id                         0
mobile_verified                     0
num_orders_last_50days              0
num_cancelled_orders_last_50days    0
num_refund_orders_last_50days       0
total_payment_last_50days           0
num_associated_customers            0
first_order_datetime             1240
dtype: int64
```

```
## Check distrubution of length of customer_id
customer_df['customer_id'].str.len().value_counts()
```

```
customer_id
8     2192007
9        3888
12         17
13          4
Name: count, dtype: int64
```

EDA (Explaratory Data Analysis)

Order

- **Shape** : 2,270,509 rows, 8 columns
- **Unique Values of Categorical Data**:
    - country_code: ['PH', 'MY', 'BD', 'PK', 'TH'], 5 unique values
    - order_id: ['d8b8-ni51'] 2,269,398 unique values
    - collect_type: ['delivery', 'pickup'], 2 unique values
    - payment_method: ['credit card', 'payment on delivery', 'antfinancial gcash', 'generic creditcard', ...], 18 unique values
- **Duplicates**:
    - Exact duplicated rows: 1,110

```
country_code: ['PH' 'BD' 'MY' 'PK' 'TH'], 5 unique values

order_id: ['d8b8-ni51' 'q4zf-tpxz' 'r2mt-m6u8' ... 'hkih-qla6' 't7sj-53m5'
 'i4wl-im1w'], 2269398 unique values

collect_type: ['delivery' 'pickup'], 2 unique values

payment_method: ['credit card' 'payment on delivery' 'antfinancial gcash' 'balance'
 'generic creditcard' 'invoice' 'no payment' 'paypal' 'xendit directdebit'
 'antfinancial bkash' 'cybersource creditcard' 'antfinancial tng'
 'razer online banking' 'adyen hpp boost' 'adyen hpp molpay'
 'cybersource applepay' 'jazzcash wallet' 'antfinancial truemoney'], 18 unique values
```

```
Number of exactly duplicate rows: 1110
Number of duplicate composite key of 'order_id & country_code' 0
```

	country_code	order_id	collect_type	payment_method
count	2270509	2270509	2270509	2270509
unique	5	2269398	2	18
top	MY	o333-nddg	delivery	payment on delivery
freq	860790	3	2241675	747537

EDA (Explaratory Data Analysis)

Label

- **Shape** : 2,414,179 rows, 4 columns
- **Unique Values of Categorical Data**:
  - country_code: ['PH', 'MY', 'BD', 'PK', 'TH'], 5 unique values
  - order_id: ['d8b8-ni51'] 2,269,398 unique values
  - customer_id: 1287588 unique values e.g.['phjr7fpu']
  - is_fraud: [’0’,’1’] 2 unique values
- **Duplicates**:
  - Exact duplicated rows: 143,361
- **Data Integrity**:
  - **Customer IDs**: **label** has 1,291,985 unique customer IDs, while **customer** has 1,287,588, resulting in 4,816 unmatched entries in **label_df**.
  - **Order IDs**: **label** contains 2,270,563 unique order IDs, compared to 2,269,398 in **order**, indicating 1,165 unidentified orders in **label**.

```
## Check for duplicates
print("Number of exactly duplicate rows:", label_df.duplicated().sum())

## Check for number of duplicated composite key
print("Number of duplicate composite key of 'order_id & country_code'",
```

```
Number of exactly duplicate rows: 143361
Number of duplicate composite key of 'order id & country code' 254
```

	country_code	order_id	customer_id	is_fraud
count	2414179	2414179	2414179	2414179
unique	5	2270563	1291985	2
top	MY	n1vp-nn6p	my2nvlmz	0
freq	916691	12	375	2163141

```
## Check for missing values
label_df.isna().sum()
```

```
country_code    0
order_id        0
customer_id     0
is_fraud        0
dtype: int64
```

EDA (Explaratory Data Analysis)

Data Type Issues:

- All columns in dataset are objects.
- Convert them to their respective datatype. (int ,float ,uint ,datetime ,category ,bool)

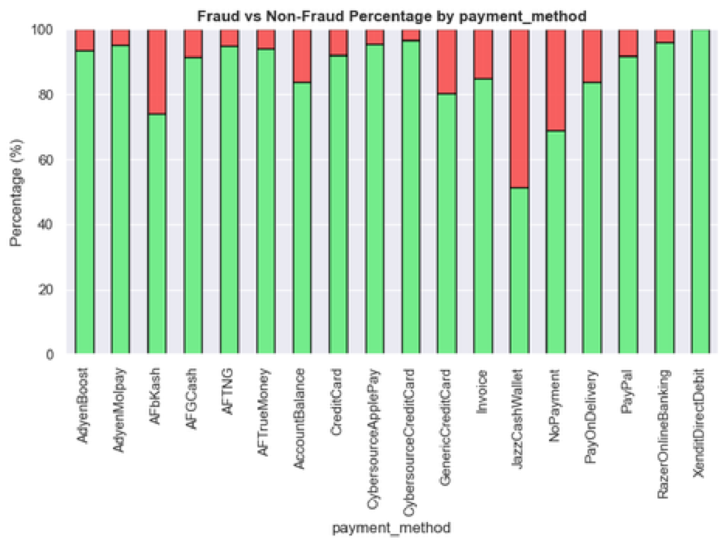
```
label_df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2414179 entries, 0 to 2414178
Data columns (total 4 columns):
 #   Column        Dtype
---  ------        -----
 0   country_code  object
 1   order_id      object
 2   customer_id   object
 3   is_fraud      object
dtypes: object(4)
memory usage: 73.7+ MB
```

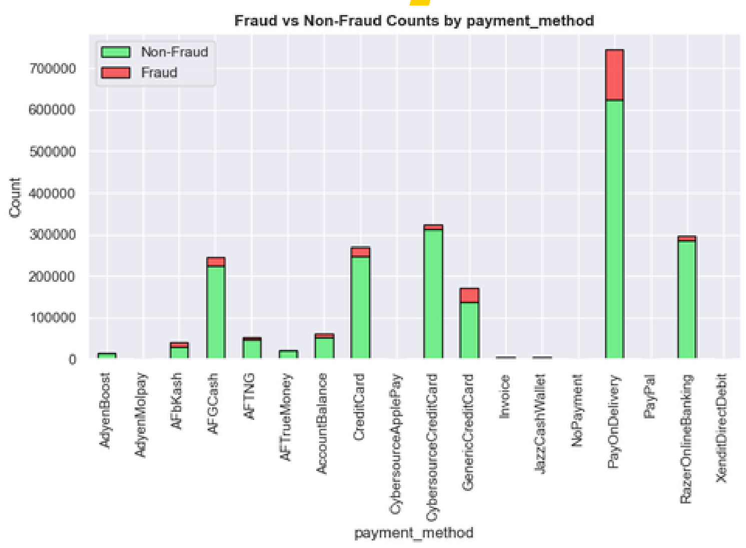
```
order_df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2270509 entries, 0 to 2270508
Data columns (total 8 columns):
 #   Column            Dtype
---  ------            -----
 0   country_code      object
 1   order_id          object
 2   collect_type      object
 3   payment_method    object
 4   order_value       object
 5   num_items_ordered object
 6   refund_value      object
 7   order_date        object
dtypes: object(8)
memory usage: 138.6+ MB
```

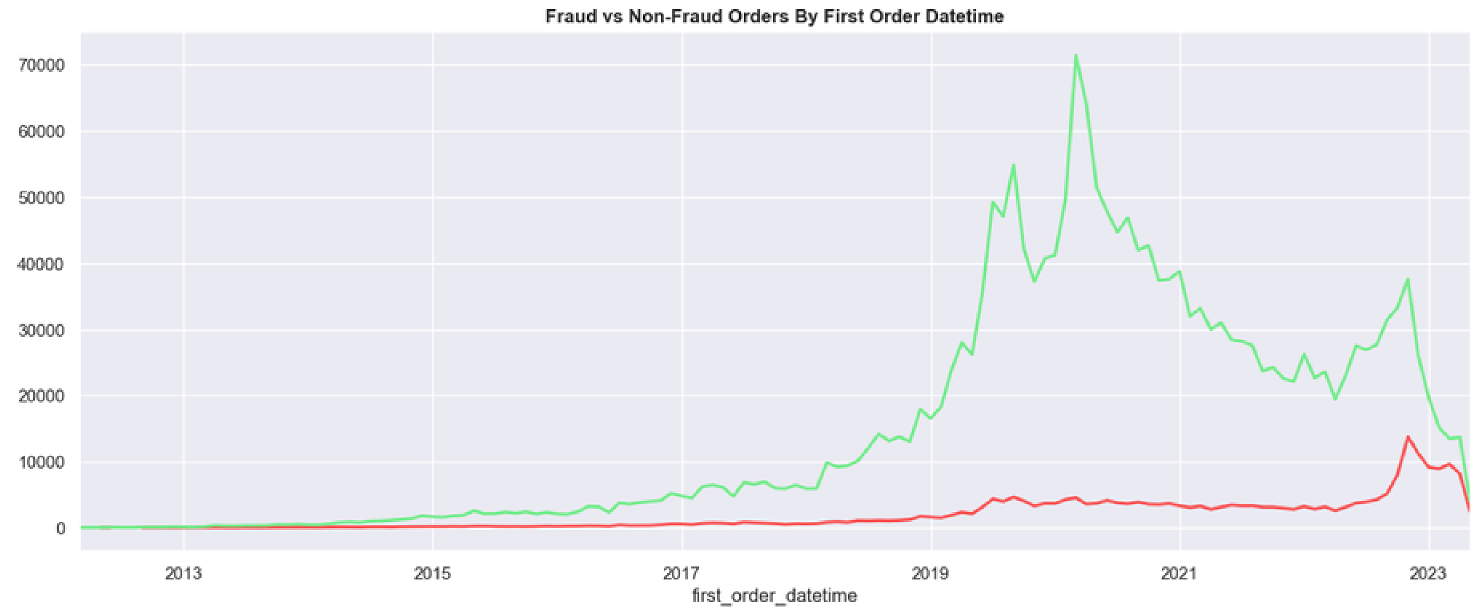
```
customer_df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2195916 entries, 0 to 2195915
Data columns (total 10 columns):
 #   Column                          Dtype
---  ------                          -----
 0   country_code                    object
 1   customer_id                     object
 2   mobile_verified                 object
 3   num_orders_last_50days          object
 4   num_cancelled_orders_last_50days object
 5   num_refund_orders_last_50days   object
 6   total_payment_last_50days       object
 7   num_associated_customers        object
 8   first_order_datetime            object
 9   idx                             int64
dtypes: int64(1), object(9)
memory usage: 167.5+ MB
```

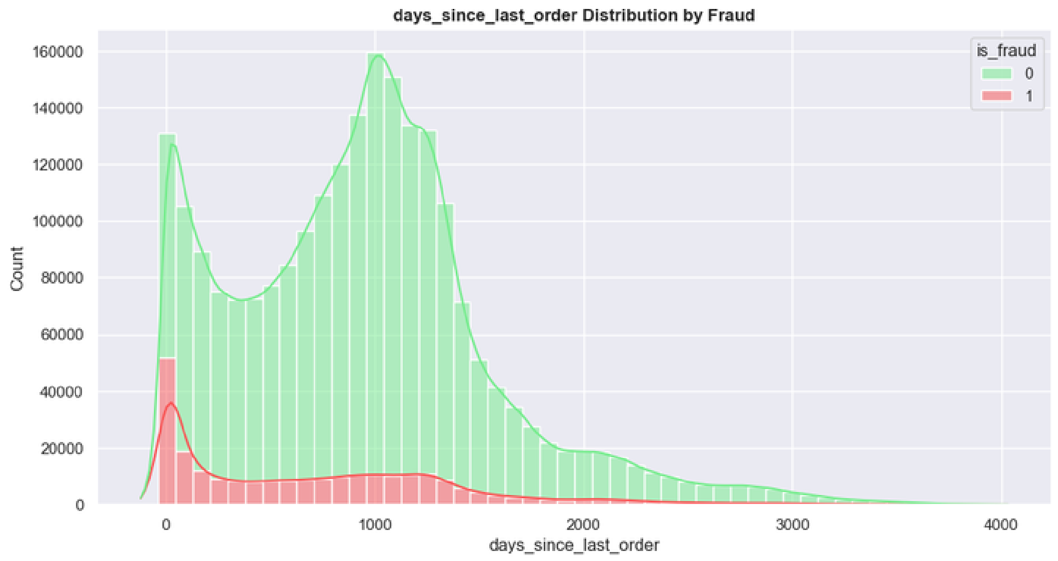
Further EDA

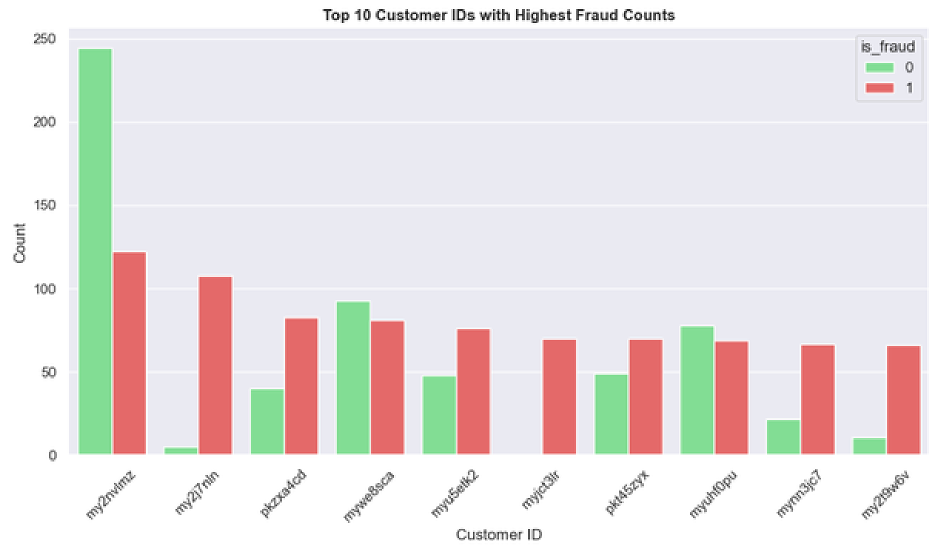
The Python graphs below helped us gain extra insights towards building our dashboards. By showing important trends and patterns, these graphs allow us to spot areas where fraud might be higher and identify any unusual activity. This way, we can create utilise the findings and put it in our dashboards

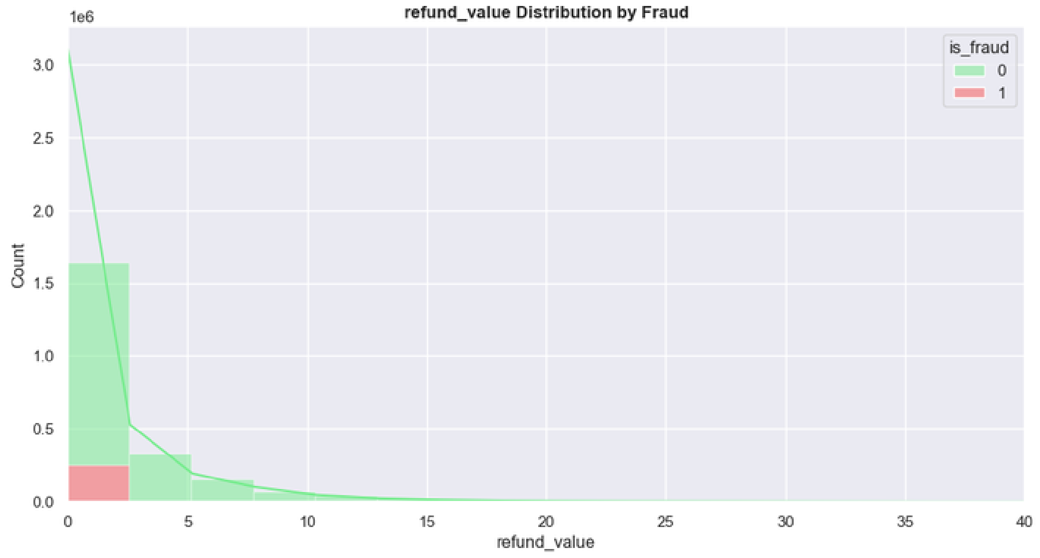












SQL Complex Query 1

- About **11%** of orders are fraudulent.
- Average order value is higher for fraudulent orders.
- Fraudulent transactions have more items on average than legitimate ones.
- Newer customers primarily place fraudulent orders.
- This suggests newer customers may be a key source of fraud.

```sql
WITH CustomerDetails AS (
    SELECT
        customer_id,
        DATEDIFF(DAY, first_order_datetime, GETDATE()) AS customer_time_on_platform,
        mobile_verified
    FROM Customer
),

OrderDetails AS (
    SELECT
        o.order_id,
        f.customer_id,
        o.order_value,
        o.num_items_ordered,
        f.is_fraud
    FROM "Order" o
    JOIN Fraud f ON o.order_id = f.order_id AND o.country_code = f.country_code
),

AggregatedData AS (
    SELECT
        is_fraud,
        AVG(CAST(order_value AS FLOAT)) AS avg_order_value,
        AVG(CAST(num_items_ordered AS FLOAT)) AS avg_num_items_ordered,
        AVG(CAST(customer_time_on_platform AS FLOAT)) AS avg_customer_time_on_platform,
        COUNT(*) AS order_count
    FROM OrderDetails od
    JOIN CustomerDetails cd ON od.customer_id = cd.customer_id
    GROUP BY is_fraud
)

SELECT
    CASE WHEN is_fraud = '1' THEN 'Fraud' ELSE 'Non-Fraud' END AS [Order Status],
    ROUND(avg_order_value, 2) AS [Average Order Value],
    ROUND(avg_num_items_ordered, 2) AS [Average Number of Items Ordered],
    ROUND(avg_customer_time_on_platform, 2) AS [Average Time On Platform],
    order_count
FROM AggregatedData
```

	Order Status	Average Order Value	Average Number of Items Ordered	Average Time On Platform	order_count
1	Fraud	7.78	5.26	1371.33	249348
2	Non-Fraud	6.83	3.47	1626.16	2014332

SQL Complex Query 2

- Pickup collection orders are more prone to fraud.
- Payment methods like JazzCashWallet and AFbKash have fraud rates exceeding 50%.
- These high fraud rates represent a significant portion of total orders.

	Payment Method	Collection Method	Total Orders	Fraudulent Orders	fraud_rate	avg_order_value	risk_classification
1	JazzCashWallet	pickup	163	113	69.33	2.52	High Fraud Risk Channel
2	AFbKash	pickup	4066	2320	57.06	3.44	High Fraud Risk Channel
3	PayOnDelivery	pickup	472	234	49.58	11.62	High Fraud Risk Channel
4	JazzCashWallet	delivery	5675	2720	47.93	3.82	High Fraud Risk Channel
5	GenericCreditCard	pickup	4143	1528	36.88	3.94	High Fraud Risk Channel
6	CreditCard	pickup	6617	2185	33.02	5.08	High Fraud Risk Channel
7	NoPayment	delivery	1393	437	31.37	6.76	High Fraud Risk Channel

```sql
WITH OrderStats AS (
    SELECT
        o.payment_method,
        o.collect_type,
        COUNT(*) as total_orders,
        SUM(CASE WHEN t.is_fraud = 1 THEN 1 ELSE 0 END) as fraud_orders,
        ROUND(AVG(o.order_value), 2) as avg_order_value,
        ROUND(AVG(o.refund_value), 2) as avg_refund_value
    FROM "Order" o
    JOIN Fraud t ON o.order_id = t.order_id
    GROUP BY
        o.payment_method,
        o.collect_type
)
SELECT
    payment_method AS [Payment Method],
    collect_type As [Collection Method],
    total_orders As [Total Orders],
    fraud_orders AS [Fraudulent Orders],
    ROUND(CAST(fraud_orders AS FLOAT) / total_orders * 100, 2) as fraud_rate,
    avg_order_value,
    CASE
        WHEN fraud_orders > 10 AND (CAST(fraud_orders AS FLOAT) / total_orders * 100) > 5
        THEN 'High Fraud Risk Channel'
        ELSE 'Normal Channel'
    END as risk_classification
FROM OrderStats
ORDER BY fraud_rate DESC;
```

SQL Complex Query 3

- Bangladesh and Pakistan have higher fraud rates, around 30% and 25%, respectively.
- This indicates a greater risk of fraudulent orders in these two countries compared to others.

	Country Code	Country	Total Orders	Fraud Orders	Fraud Rate	risk_segment
1	BD	Bangladesh	142398	41873	29.41	Higher Risk
2	PK	Pakistan	440078	108217	24.59	Higher Risk
3	PH	Philippines	670952	56867	8.48	Lower Risk
4	TH	Thailand	151871	9066	5.97	Lower Risk
5	MY	Malaysia	858381	33325	3.88	Lower Risk

```sql
WITH FraudCounts AS (
    SELECT
        f.country_code,
        COUNT(*) AS total_orders,
        SUM(CASE WHEN f.is_fraud = '1' THEN 1 ELSE 0 END) AS fraud_cases
    FROM Fraud f
    JOIN "Order" o ON f.order_id = o.order_id AND f.country_code = o.country_code
    GROUP BY f.country_code
),

FraudPercentages AS (
    SELECT
        country_code,
        CASE country_code
            WHEN 'BD' THEN 'Bangladesh'
            WHEN 'PK' THEN 'Pakistan'
            WHEN 'PH' THEN 'Philippines'
            WHEN 'TH' THEN 'Thailand'
            WHEN 'MY' THEN 'Malaysia'
            ELSE 'Unknown'
        END AS country_name,
        total_orders,
        fraud_cases,
        (CAST(fraud_cases AS FLOAT) / NULLIF(total_orders, 0)) * 100 AS fraud_percentage
    FROM FraudCounts
)

SELECT
    country_code AS [Country Code],
    country_name AS [Country],
    total_orders AS [Total Orders],
    fraud_cases AS [Fraud Orders],
    ROUND(fraud_percentage, 2) AS [Fraud Rate],
    CASE
        WHEN fraud_percentage >= 10 THEN 'Higher Risk'
        ELSE 'Lower Risk'
    END AS risk_segment
FROM FraudPercentages
ORDER BY [Fraud Rate] DESC;
```

ETL Pipeline

Extract

- Data is sourced from a SQL Server database (PAI_CA1) using SQLAlchemy's create_engine.
- Three tables are queried:
  - training-data-customer-features_v1.0 (customer data)
  - training-data-order-features_v1.0 (order data)
  - training-labels_v1.0 (label data)
- Data from these tables is loaded into Pandas DataFrames for further processing.
- Dask is later utilized, and the ETL pipeline is modified to enhance processing speed, as discussed in the additional section.

```python
## SQL Server Name and Database Name
# server = 'yj\SQLEXPRESS'
# server = 'REJEY-LAPTOP\SQLEXPRESS'
server = 'XIAOYANG23\SQLEXPRESS'
database = 'PAI_CA1'

## Create a connection to the SQL Server
engine = create_engine('mssql+pyodbc://{}/{}?driver=ODBC Driver 17 for SQL Server'.format(server, database))

customer_df = pd.read_sql('SELECT * FROM [dbo].[training-data-customer-features_v1.0]', engine)
order_df = pd.read_sql('SELECT * FROM [dbo].[training-data-order-features_v1.0]', engine)
label_df = pd.read_sql('SELECT * FROM [dbo].[training-labels_v1.0]', engine)
```

ETL Pipeline

Transform - Customer

Key Actions:

1. Converted data types for memory efficiency
2. Dropped **1,240** NA values
3. Removed **34** exact duplicates
4. Eliminated **908,294** duplicate composite keys
5. Filtered **35,223** inconsistent date rows
6. Dropped **417** customer IDs not in label dataset

Result:

From **2,195,916** to **1,286,357** rows

Improved data quality for fraud analysis

ETL Pipeline

Transform - Order

Key Actions:

- Converted data types for efficiency
- Dropped 3 NA values in num_items_ordered
- Removed 1,110 exact duplicates
- Renamed values for readability

Result:

- From 2,270,509 to 2,269,396 rows
- Improved data quality and clarit

ETL Pipeline

Transform - Label

Key Actions:

- Converted data types to bool and category
- Removed 4,816 customer IDs not in customer dataset
- Removed 1,165 order IDs not in order dataset
- Dropped 143,615 duplicate composite keys

Result:

- From 2,414,179 to 2,263,680 rows
- Enhanced data consistency for fraud analysis

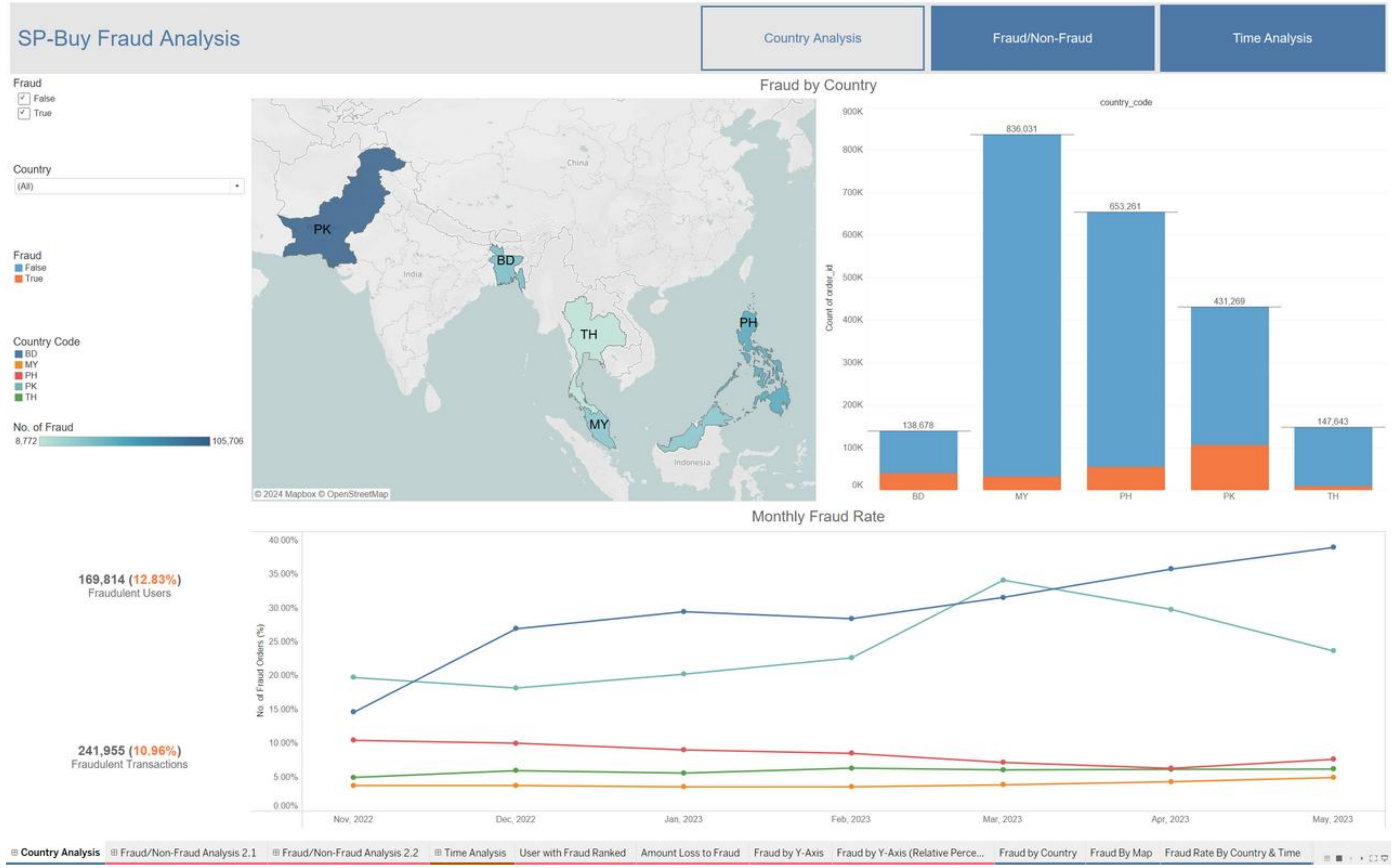
ETL Pipeline

Load

The cleaned and transformed DataFrames (customer_df, order_df, label_df) are ready for further analysis or integration into the next steps of the workflow. Therefore, we loaded the cleaned datasets back into SQL for further analysis using complex SQL Queries & Tableau.

```
# load back to sql
customer_df.to_sql('clean-data-customer_v1.0', con=engine, index=True, if_exists='replace')
order_df.to_sql('clean-data-order_v1.0', con=engine, index=True, if_exists='replace')
label_df.to_sql('clean-data-label_v1.0', con=engine, index=True, if_exists='replace')
```

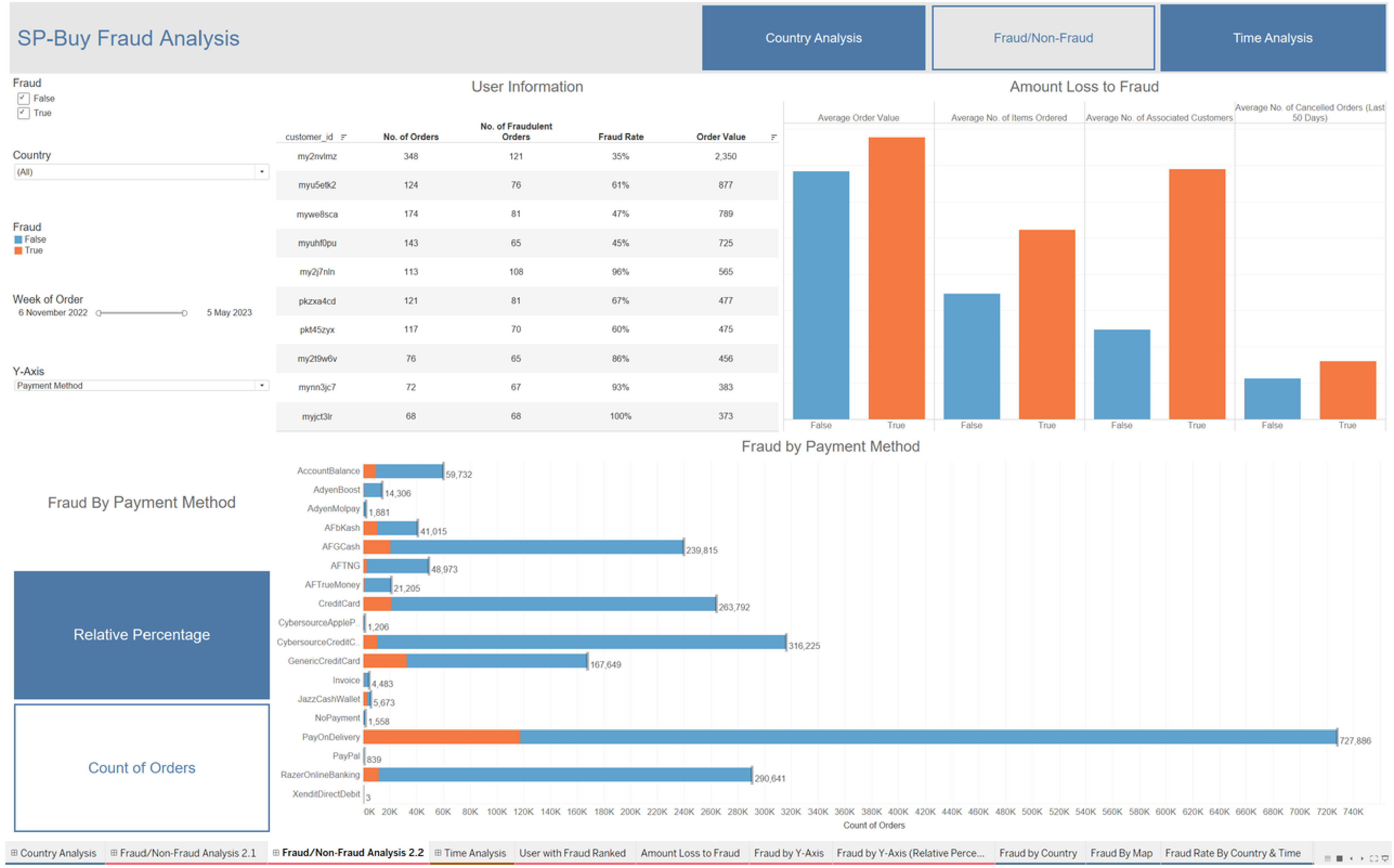
Data Visualization - 1



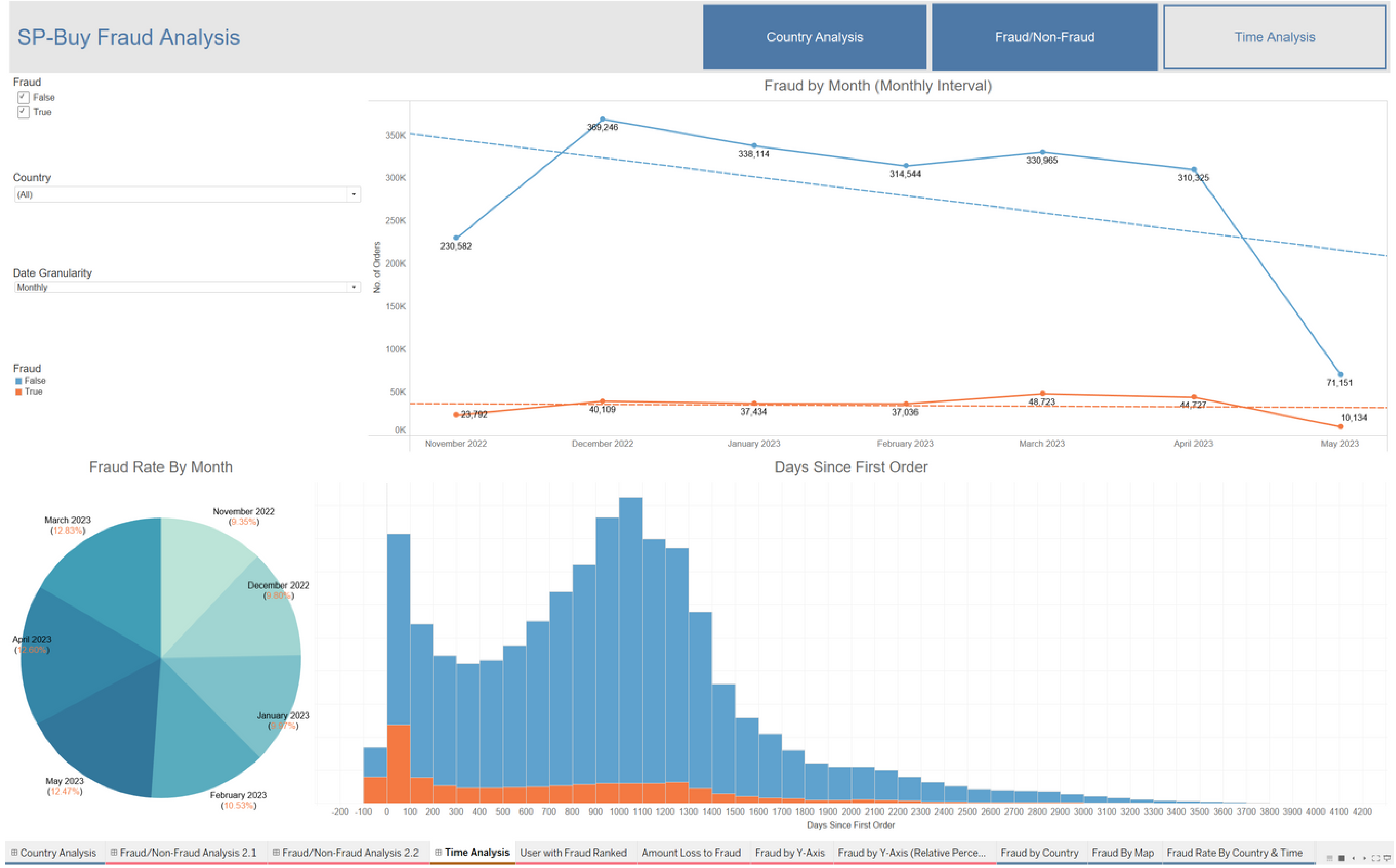
- The Fraud by Country map and bar chart indicates that Malaysia, Philippines, and Pakistan have a significantly higher number of orders.
- A notable portion of orders from Pakistan consists of fraudulent transactions.
- Fraudulent users account for 12.83% of the total user base.
- Fraudulent transactions represent 10.96% of all transactions, highlighting the severity of fraud in the company.
- The Monthly Fraud Rate line chart shows trends over time:
  - Countries like Pakistan and Bangladesh experience consistent increases in fraud rates.
  - Pakistan’s fraud rates have dropped since March 2023.
  - Other countries, such as Thailand (TH), remain relatively stable in their fraud rates.

Data Visualization - 2

- Fraudulent transactions have:
- Higher average order values
- A greater number of items ordered
- More associated customers
- A higher number of canceled orders compared to non-fraudulent transactions
- Payment methods with significantly higher fraud incidences include:
  - "CreditCard"
  - "GenericCreditCard"
  - "AFGCash"
- The "PayOnDelivery" payment method has the highest fraud incidence and significantly more orders.



Data Visualization - 3



- Total orders peaked in December 2022 with **369,246** orders
- Fraudulent orders were highest in March 2023, accounting for **12.83%** of total orders that month
- A steady decline in both total and fraudulent orders is observed
- A significant drop in orders occurred in May 2023
- Distribution of days since the first order:
- A higher proportion of fraud occurs earlier in a customer's lifecycle
- This highlights potential risks associated with new customers
- Insights and recommendations:
- Enhanced fraud detection strategies are needed for new users
- Especially important during peak order months

Additional

- **Dusk**
  - Utilized Dask for its efficient parallel computing to merge datasets, significantly outperforming Pandas in speed during the merging process.
  - Had difficulties in implementing it and takes longer after each run.
- **Airflow**
  - Tried to setup automatic schedules to run the ETL Pipeline.
  - Once the docker enviroment was setup we would place the ETL notebook into the DAG whilst using the BashOperator to run the file.
- **Tableau Dashboard**
  - We have created a flow that will send the merged file to Tableau Server where the Dashboard will extract the database from the ever updating database, thus the dashboard will be very responsive even as data scales.

Conclusion

We established a robust data analysis system for SP-buy, addressing the critical issue of fraudulent activities within its e-commerce platform. By implementing a SQL database, an efficient ETL pipeline connecting SQL and Python, and conducting thorough data analysis, we uncovered significant fraud patterns and provided actionable insights through interactive dashboards.

Showing that Pakistan and Bangladesh has the most number of fraudulent cases, newer customers are most likely to commit fraud, we can also see that the most fraudulent transactions methods are "CreditCard","GenericCreditCard" and "AFGCash" thus our dashboard and analysis will help SP-buy find fradulent purchase trends and insights to deal with the issue at hand