Financial Fraud Detection: Credit Card Transaction Analysis for SecureGuard

SecureGuard Financial Solutions specialises in delivering innovative, real-time solutions to detect and prevent fraud in the financial sector. With the rise of digital payments, rapid fraud detection is vital to reduce risk, maintain customer trust, and protect valuable assets. This report presents a comprehensive workflow for analyzing credit card transaction data using Python, Excel, SQL, and Tableau, supporting SecureGuard's mission to identify anomalous spending and fraudulent activity.

1. Data Pre-processing in Python

1.1. Package Installation

```
pip install pandas
```

1.2. Filtering, Selecting, and Stratified Sampling

```
import pandas as pd

# Load the dataset
df = pd.read_csv('your_file.csv')

# Keep necessary columns
cols_needed = ['amt', 'city_pop', 'is_fraud', 'gender', 'category',
'state', 'job']
df = df[cols_needed]

# Clean data
df = df[df['amt'] > 0]
df = df[df['gender'].notnull()]
df = df.drop_duplicates()

# Stratified sample: 5% from each 'category', retaining small groups
optimum_frac = 0.05
stratified_sample = df.groupby('category', group_keys=False).apply(
    lambda x: x.sample(frac=optimum_frac, random_state=42) if len(x) >
20 else x
).reset_index(drop=True)

# Export for Excel analysis
stratified_sample.to_csv('stratified_sample.csv', index=False)
```

2. Data Exploration & Analysis in Excel Online

2.1. Import Cleaned Data

- Upload `stratified_sample.csv` to Excel Online for exploration.

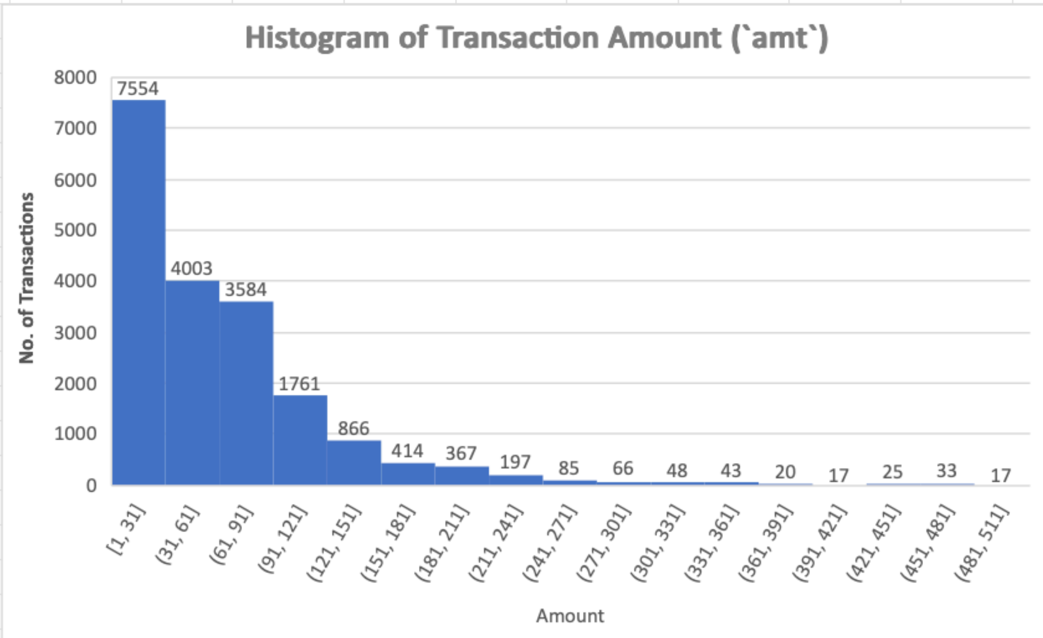
2.2. Statistical Overview

- Calculate min, max, average, median, and stdev for `amt` and `city_pop`.
- Add formulae such as `=MAX(A2:A1000)` as needed.

Statistical Summary		
	amt	city_pop
range	a2:a19343	b2:b19343
min	1	23
max	6853.9	2906700
average	70.43245528	88977.26119
median	47.845	2456
stdev	142.8941042	298286.5915
Correl.	-0.000896042	

2.3. Visual Exploration

- Plot histogram for `amt`.



- Use pivot tables for:
  - Fraud count by gender and category

Cross-tab (Number of Frauds by Gender and Category)		
gender	**category**	**Sum of is_fraud**
⊟F	entertainment	4
	food_dining	0
	gas_transport	3
	grocery_net	2
	grocery_pos	10
	health_fitness	1
	home	1
	kids_pets	3
	misc_net	7
	misc_pos	6
	personal_care	4
	shopping_net	9
	shopping_pos	9
	travel	0
F Total		**59**
⊟M	entertainment	0
	food_dining	0
	gas_transport	6
	grocery_net	0
	grocery_pos	11
	health_fitness	0
	home	0
	kids_pets	1
	misc_net	14
	misc_pos	3
	personal_care	0
	shopping_net	17
	shopping_pos	8
	travel	2
M Total		**62**
Grand Total		**121**

Screenshot 2025-08-18 at 11.09.00.png

- Top 3 states by transaction count

state	Count of amt
TX	1394
NY	1238
PA	1209
CA	816
OH	754
MI	689
AL	631

Screenshot 2025-08-24 at 11.16.59.png

- Average transaction amount by job

Job Role	Avg. amt.
Office on the web Frame	35.44
Air Traffic Controller	771.49
IT Consultant	282.49
Minerals Surveyor	262.77
Medical Physicist	191.99

Screenshot 2025-08-24 at 11.17.26.png

- Fraud count by category

category	Sum of is_fraud
entertainment	4
food_dining	0
gas_transport	9
grocery_net	2
grocery_pos	21
health_fitness	1
home	1
kids_pets	4
misc_net	21
misc_pos	9
personal_care	4
shopping_net	26
shopping_pos	17
travel	2
Grand Total	**121**

Screenshot 2025-08-24 at 11.17.12.png

2.4. Insights

- Most transactions are low-value with few outliers.
- Fraud most common in shopping and grocery categories.
- Certain professions have higher average amounts.
- Texas, New York, Pennsylvania lead in transaction count.

3. Data Analysis with SQL

3.1. Schema and Loading

```
CREATE SCHEMA finance;
USE finance;
-- Create cc_data and location_data tables matching CSVs
```

3.2. Key SQL Queries

```
-- 1. Total transactions
SELECT COUNT(*) FROM cc_data;

-- 2. Top merchants
SELECT merchant, COUNT(*) AS txn_count FROM cc_data GROUP BY merchant
ORDER BY txn_count DESC LIMIT 10;

-- 3. Avg. transaction by category
SELECT category, AVG(amt) FROM cc_data GROUP BY category;

-- 4. Count & % fraud
SELECT COUNT(*) AS total, SUM(is_fraud), ROUND(100.0 * SUM(is_fraud) /
COUNT(*), 2) FROM cc_data;
```

merchant	txn_count
fraud_Kilback LLC	1305
fraud_Cormier LLC	1120
fraud_Kuhn LLC	1058
fraud_Schumm PLC	1024
fraud_Dickinson Ltd	997
fraud_Boyer PLC	990
fraud_Jenkins, Hauck and Friesen	856
fraud_Rodriguez Group	839
fraud_Eichmann, Bogan and Rodriguez	829
fraud_Christiansen, Goyette and Schamberger	828

total_transactions
389002

category	avg_transaction_amt
misc_net	82.03714838844692
grocery_pos	117.5270554633738
shopping_net	88.52703420495718
personal_care	48.12700690461437
gas_transport	63.39943986072221
entertainment	64.09267228504379
home	58.292295579587204
food_dining	51.19593018101227
kids_pets	57.602997316188
travel	114.35104240137805
misc_pos	61.842209331337294
shopping_pos	79.33264623557683
health_fitness	54.22613982589778
grocery_net	53.7010638451539

total_txns	total_frauds	fraud_pct
389002	2252	0.58

- Join `cc_data` & `location_data` to get Geo-coordinates for mapping.

```
SELECT
  cc.trans_num,
  cc.cc_num,
  cc.city,
  cc.state,
  loc.lat,
  loc.long AS long_
FROM cc_data cc
LEFT JOIN location_data loc
  ON cc.cc_num = loc.cc_num
WHERE loc.lat IS NOT NULL AND loc.long IS NOT NULL;
```

trans_num	cc_num	city	state	lat	long_
9ffdd433bd2838e9945f0fee1934d185	60416207185	Fort Washakie	WY	43.0048	-108.8964
6a41999fce77e2c185d5a577d4de992d	60422928733	North Augusta	SC	33.6028	-81.9748
d78866e6c80ce4e1af1b7388bcbc5d46	60416207185	Fort Washakie	WY	43.0048	-108.8964
bfb251df5cb4f4319d36123b80588ab4	60422928733	North Augusta	SC	33.6028	-81.9748
b0349e51f9b6427b86b298cdce2b215d	60427851591	Burns Flat	OK	35.3492	-99.188
16aa0f8cc0d1de40ca15ff25103cc83d	60422928733	North Augusta	SC	33.6028	-81.9748
4b64fb301328ae2b32386b12ff5431ee	60495593109	Dallas	TX	32.7699	-96.743
7d94c48ce8a59bde5cdeace1c68aef97	60423098130	Amorita	OK	36.9412	-98.2458
00615d54ac379310bcd9cc0be7e56fbe	60490596305	Haynes	AR	34.8838	-90.7666
8356383cb344d67ed420685f8ff65c3d	60416207185	Fort Washakie	WY	43.0048	-108.8964
93ec62362833289f9c4e85febe479d1c	60487002085	Jackson	MS	32.3739	-90.1293
3d0cd73ea68e3bd9211ece71e0f97409	60422928733	North Augusta	SC	33.6028	-81.9748
b81527995c158ffbdd1925c3964b00c6	60490596305	Haynes	AR	34.8838	-90.7666
9149089bff985d21a4e3a77f3899b62b	60422928733	North Augusta	SC	33.6028	-81.9748
f196dedd61bdd55f4a387d06e94e091d	60422928733	North Augusta	SC	33.6028	-81.9748
5c89f9d4e8a2ccdaa9b10b33739327bd	60427851591	Burns Flat	OK	35.3492	-99.188
9f8033bb78e68184a5733901f689f563	60416207185	Fort Washakie	WY	43.0048	-108.8964
63d3d75d8d057124df01cdcb1907b259	60495593109	Dallas	TX	32.7699	-96.743
3acc16cf35e3c5b5a4f29b62c894a1bd	60495593109	Dallas	TX	32.7699	-96.743

Screenshot 2025-08-19 at 11.50.20.png

- Find city with highest population and transaction activity.

```
SELECT city, state, city_pop
FROM cc_data
ORDER BY city_pop DESC
LIMIT 1;

SELECT city, state, COUNT(*) AS txn_count
FROM cc_data
GROUP BY city, state
ORDER BY txn_count DESC
LIMIT 1;
```

city	state	city_pop
Houston	TX	2906700

city	state	txn_count
San Antonio	TX	1542

- Extract earliest/latest transaction dates.

```
SELECT
  MIN(trans_date_trans_time) AS earliest_txn,
  MAX(trans_date_trans_time) AS latest_txn
FROM cc_data;
```

earliest_txn	latest_txn
01-01-2019 00:06	31-12-2019 23:59

Screenshot 2025-08-19 at 11.54.17.png

- Summarise transaction total, category counts, and average amount by gender or day of week.

```
SELECT SUM(amt) AS total_spent FROM cc_data;
```

total_spent
27402136.520000726

Screenshot 2025-08-19 at 11.54.35.png

```
SELECT category, COUNT(*) AS txn_count
FROM cc_data
GROUP BY category;
```

category	txn_count
misc_net	19112
grocery_pos	36763
shopping_net	29294
personal_care	27373
gas_transport	39633
entertainment	28122
home	36897
food_dining	27070
kids_pets	33907
travel	12193
misc_pos	24048
shopping_pos	35012
health_fitness	25732
grocery_net	13846

Screenshot 2025-08-19 at 11.54.47.png

```
SELECT gender, AVG(amt) AS avg_amt
FROM cc_data
GROUP BY gender;
```

gender	avg_amt
F	69.85478898471565
M	71.15468427669961

Screenshot 2025-08-19 at 11.55.01.png

```
SELECT
  DAYNAME(STR_TO_DATE(trans_date_trans_time, '%d-%m-%Y %H:%i')) AS
day_of_week,
  AVG(amt) AS avg_amt
FROM cc_data
GROUP BY day_of_week
ORDER BY avg_amt DESC
LIMIT 1;
```

day_of_week	avg_amt
Friday	71.56510343775214

Screenshot 2025-08-19 at 11.57.16.png

4. Exploratory Data Analysis (Python / Jupyter)

4.1. Dataset Dimensions

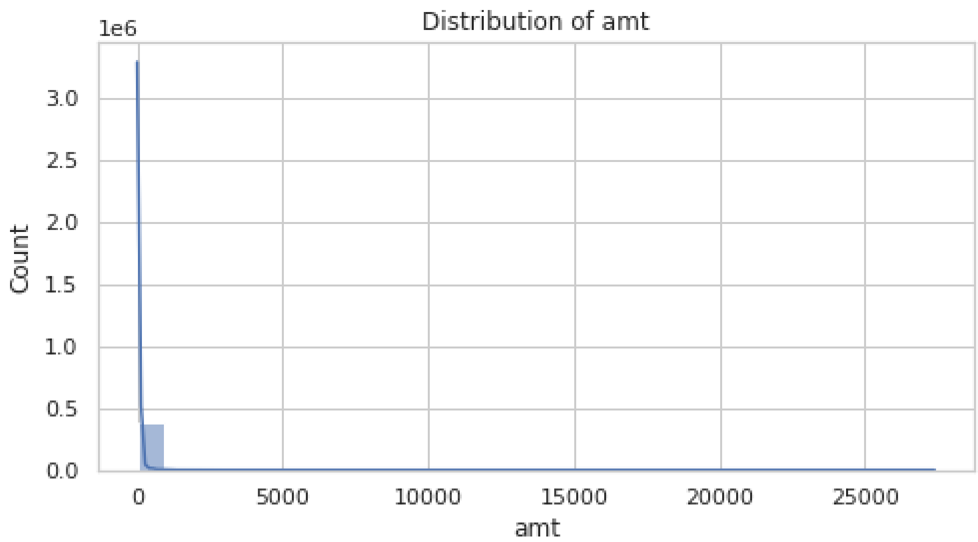
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os

# Create directory for saving plots
```

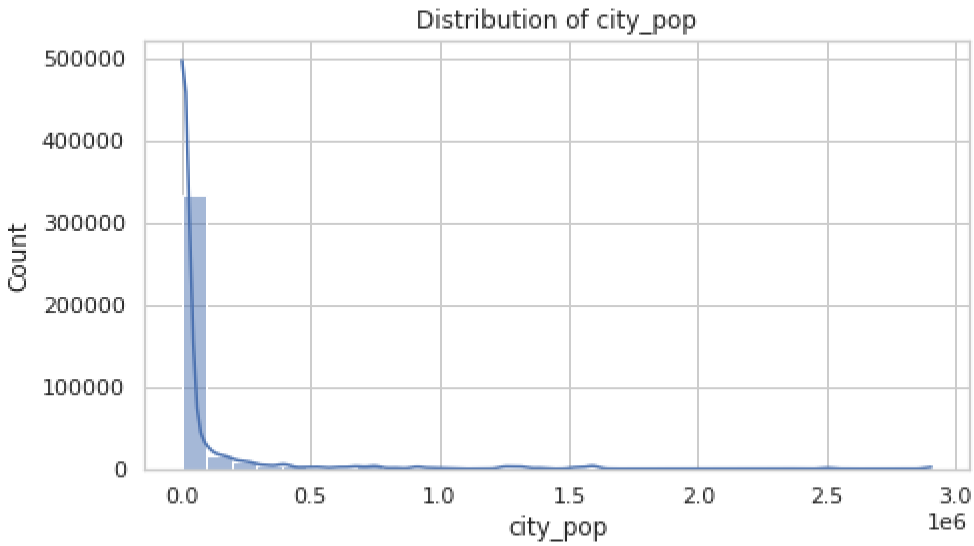
```
os.makedirs('plots', exist_ok=True)

# Visualization style
sns.set(style="whitegrid")

# Load dataset
df = pd.read_csv('cc_data.csv')

print('Rows, columns:', df.shape)
```

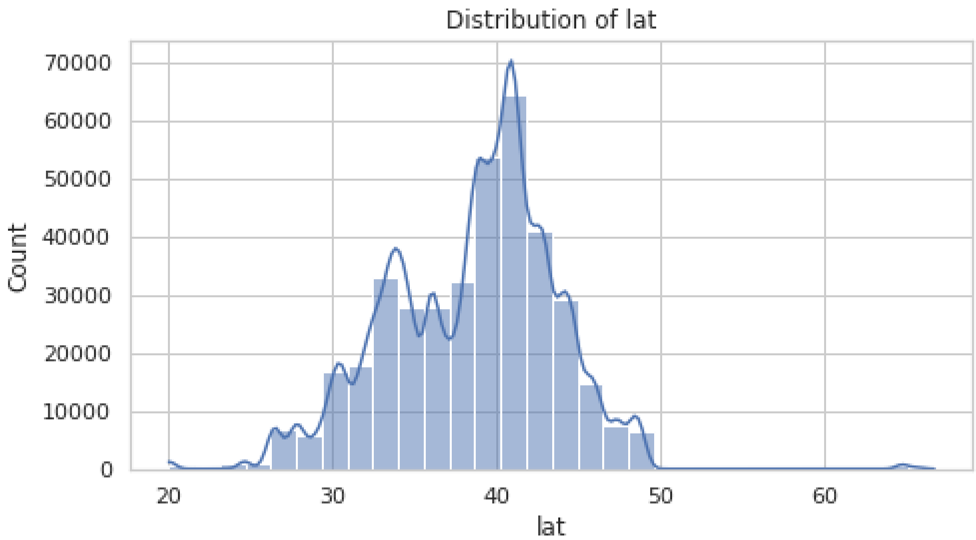
4.2. Unique Categorical Values & Distribution Plots

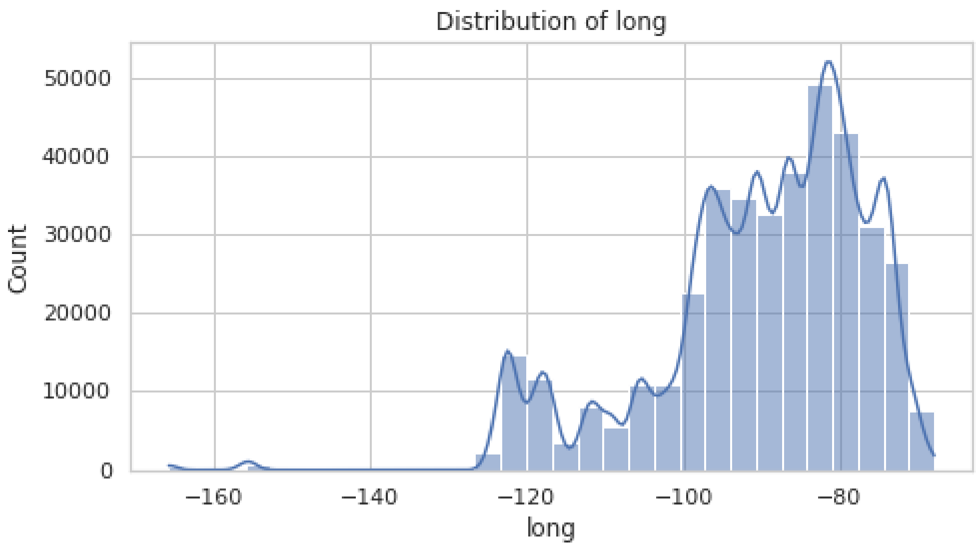
```
for col in df.select_dtypes(include=['object', 'category']):
    print(f"{col}: {df[col].nunique()} unique")
# Histograms, boxplots, KDE for numerical columns
num_cols = df.select_dtypes(include=[np.number]).columns
df[num_cols].hist(bins=30, figsize=(15, 10))
plt.suptitle('Numerical Feature Distributions')
plt.tight_layout()
plt.show()

# Focused distributions with KDE and save plots
num_plot_cols = ['amt', 'city_pop', 'lat', 'long']
for col in num_plot_cols:
    plt.figure(figsize=(7,4))
    sns.histplot(df[col], bins=30, kde=True)
    plt.title(f'Distribution of {col}')
    plt.xlabel(col)
    plt.ylabel("Count")
    plt.tight_layout()
    plt.savefig(f'plots/{col}_distribution.png')
    plt.show()
```

trans_date_trans_time: 293627 unique values merchant: 693 unique values category: 14 unique values first: 352 unique values last: 481 unique values gender: 2 unique values street: 979 unique values city: 890 unique values state: 51 unique values job: 492 unique values dob: 964 unique values trans_num: 389002 unique values.









- Visualize and check for missing values, outliers, and skew-ness.

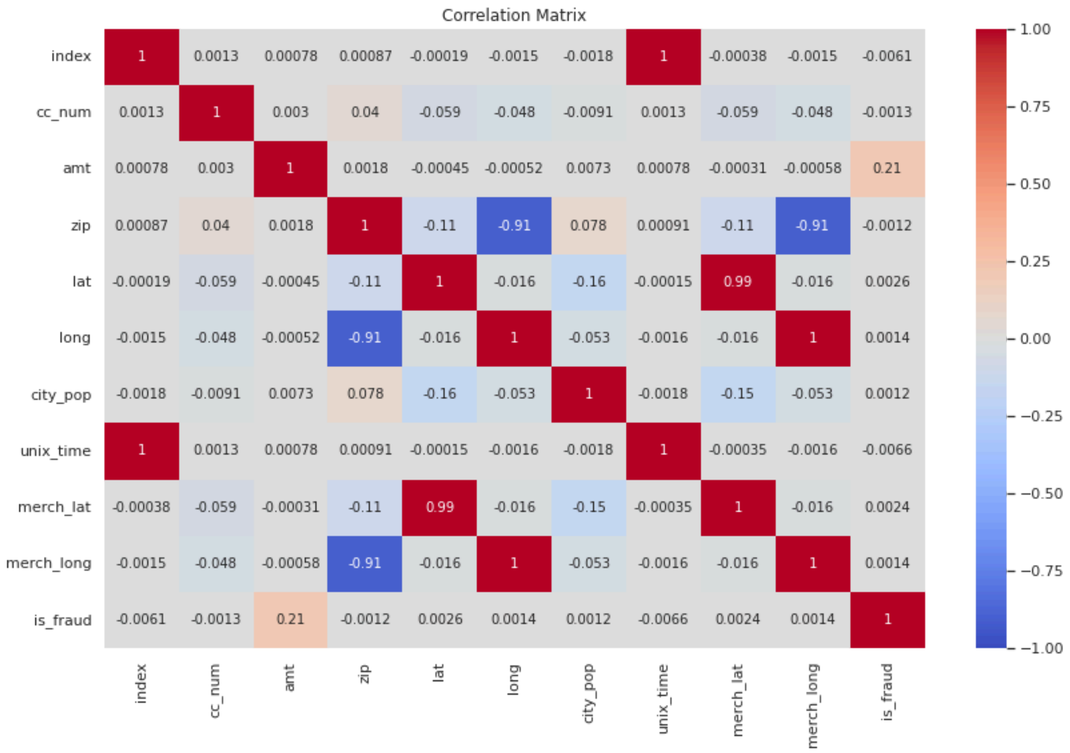
```
print(df.isnull().sum())
# No missing values detected. If needed, we could impute or drop
        missing data here.
```

index 0 trans_date_trans_time 0 cc_num 0 merchant 0 category 0 amt 0 first 0 last 0 gender 0 street 0 city 0 state 0 zip 0 lat 0 long 0 city_pop 0 job 0 dob 0 trans_num 0 unix_time 0 merch_lat 0 merch_long 0 is_fraud 0 dtype: int64.

- Compute summary stats and correlation matrix.

```
display(df.describe().T)

plt.figure(figsize=(12,8))
sns.heatmap(df[num_cols].corr(numeric_only=True), annot=True,
        cmap='coolwarm', vmin=-1, vmax=1)
plt.title("Correlation Matrix")
plt.tight_layout()
plt.savefig('plots/correlation_matrix.png')
plt.show()
# Correlation values closer to +1 or -1 indicate strong relationships.
```

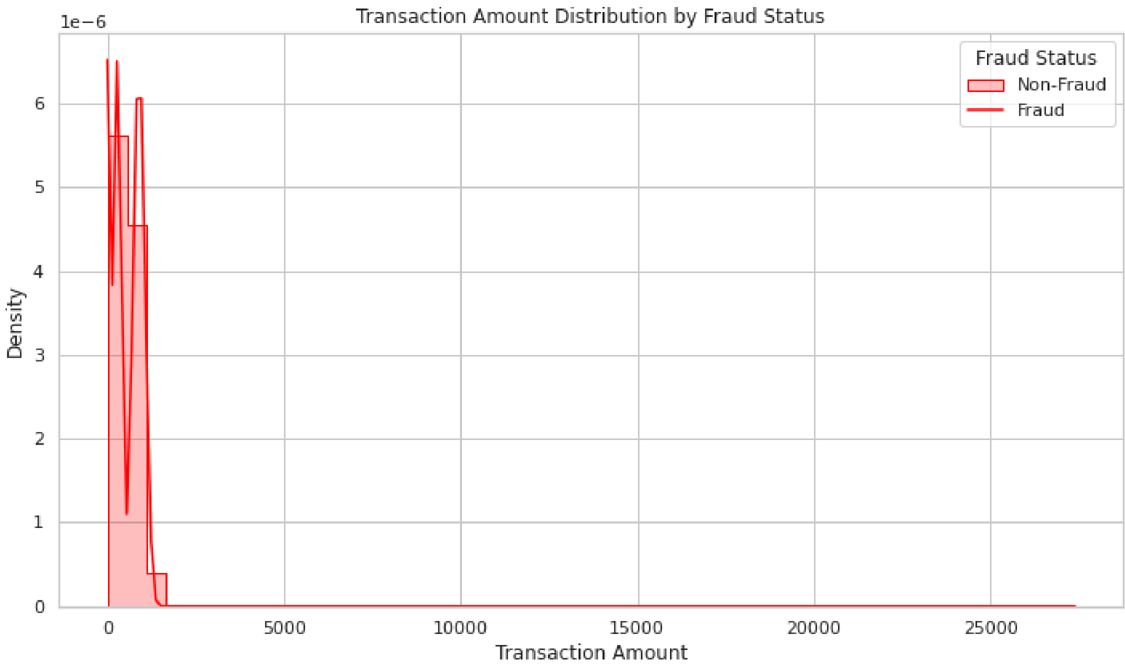
	count	mean	std	min	25%	50%	75%	max
index	389002	648520.5142	374574.3902	11	324184.25	648648.5	973503.25	1296674
cc_num	389002	4.19151E+17	1.31158E+18	60416207185	1.80043E+14	3.52142E+15	4.64226E+15	4.99235E+18
amt	389002	70.44214816	162.2039152	1	9.66	47.57	83.0775	27390.12
zip	389002	48818.0643	26879.38322	1257	26237	48174	72011	99783
lat	389002	38.53312141	5.074595765	20.0271	34.6205	39.3543	41.9404	66.6933
long	389002	-90.23766409	13.7458552	-165.6723	-96.798	-87.4769	-80.158	-67.9503
city_pop	389002	88680.84286	301210.1017	23	743	2456	20328	2906700
unix_time	389002	1349250579	12850848.63	1325376413	1338750920	1349266968	1359459514	1371816817
merch_lat	389002	38.53168276	5.109399849	19.029798	34.7193935	39.3610655	41.956012	67.064277
merch_long	389002	-90.2366743	13.75731118	-166.669638	-96.90544475	-87.4468425	-80.25383075	-66.95654
is_fraud	389002	0.005789173	0.075866156	0	0	0	0	1
hour	389002	12.80283649	6.822402724	0	7	14	19	23



- Split/compare distributions by `is_fraud`, gender, or category.

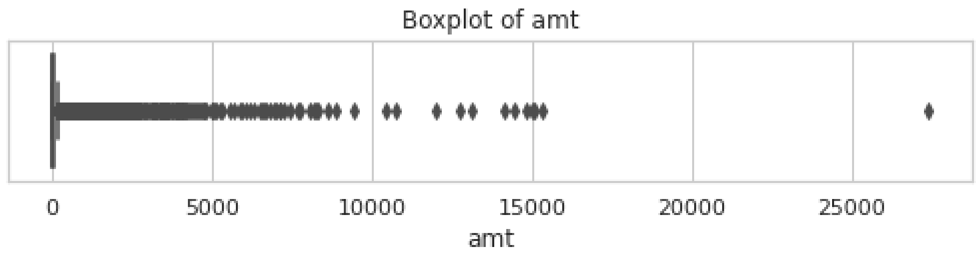
```
plt.figure(figsize=(10,6))
sns.histplot(
    data=df,
    x='amt',
    hue='is_fraud',
    hue_order=[1],
    bins=50,
    kde=True,
```

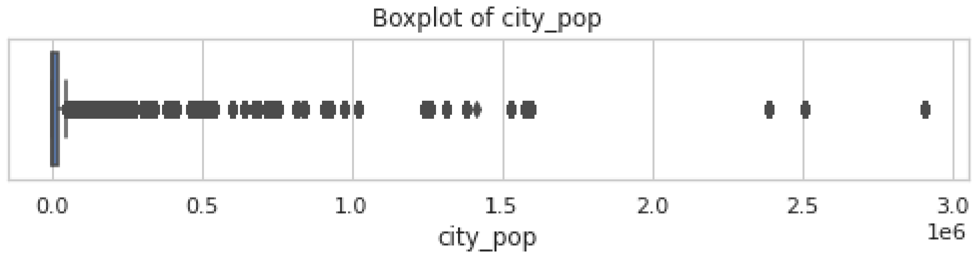
```
    element='step',
    stat='density',
    palette={0: 'blue', 1: 'red'}
)
plt.title('Transaction Amount Distribution by Fraud Status')
plt.xlabel("Transaction Amount")
plt.ylabel("Density")
plt.legend(title="Fraud Status", labels=["Non-Fraud", "Fraud"])
plt.tight_layout()
plt.savefig('plots/amt_fraud_hist.png')
plt.show()
```



- Identify and count outliers.

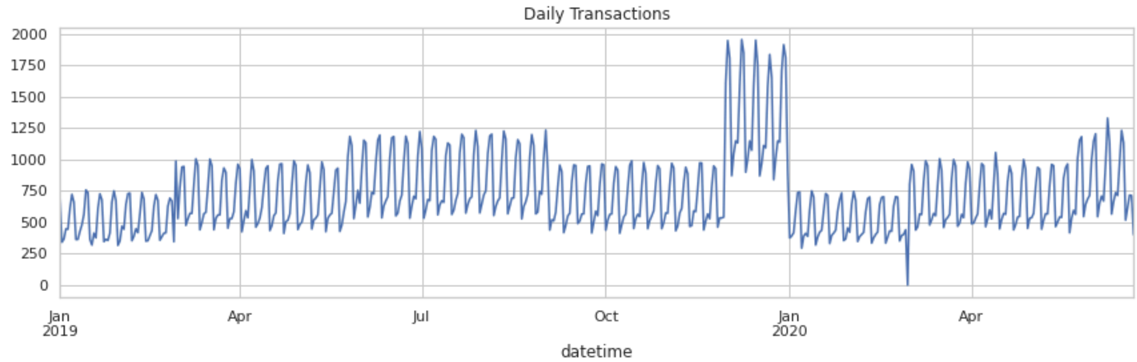
```
for col in ['amt', 'city_pop']:
    plt.figure(figsize=(7,2))
    sns.boxplot(x=df[col])
    plt.title(f'Boxplot of {col}')
    plt.tight_layout()
    plt.savefig(f'plots/{col}_boxplot.png')
    plt.show()

Q1 = df['amt'].quantile(0.25)
Q3 = df['amt'].quantile(0.75)
IQR = Q3 - Q1
outliers = df[(df['amt'] < (Q1 - 1.5*IQR)) | (df['amt'] > (Q3 +
        1.5*IQR))]
print(f"{len(outliers)} outlier(s) detected in amt")
```

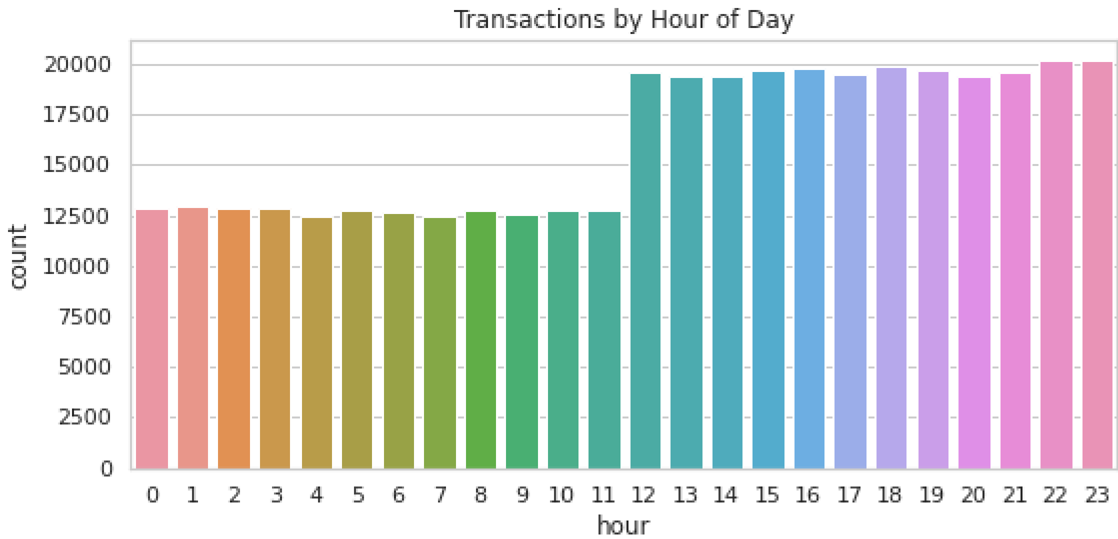




-Analyse trends over time (daily/hourly/monthly).

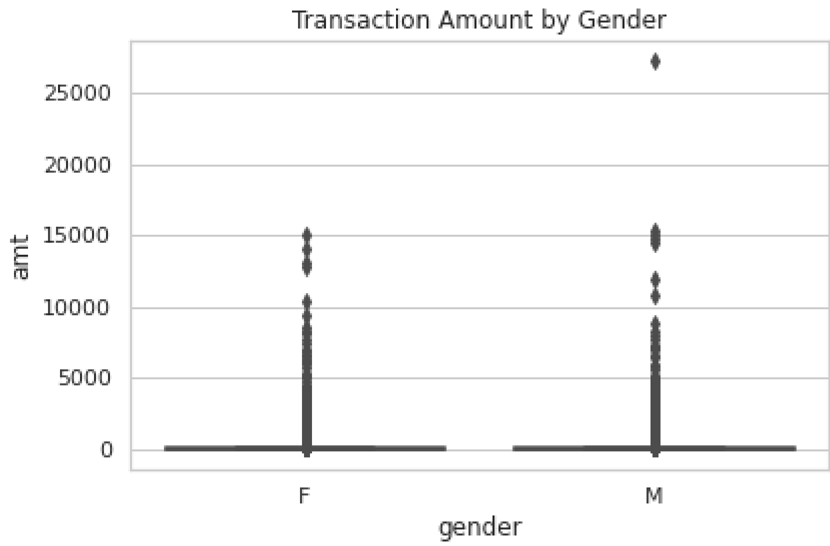
```
# Convert to datetime
df['datetime'] = pd.to_datetime(df['trans_date_trans_time'],
        format='%d-%m-%Y %H:%M')

# Daily transaction counts
plt.figure(figsize=(12,4))
df.set_index('datetime').resample('D').size().plot(title="Daily
        Transactions")
plt.tight_layout()
plt.savefig('plots/daily_transactions.png')
plt.show()

# Extract hour and plot hourly transaction counts
df['hour'] = df['datetime'].dt.hour
plt.figure(figsize=(8,4))
sns.countplot(x='hour', data=df)
plt.title("Transactions by Hour of Day")
plt.tight_layout()
plt.savefig('plots/hour_transactions.png')
plt.show()
```

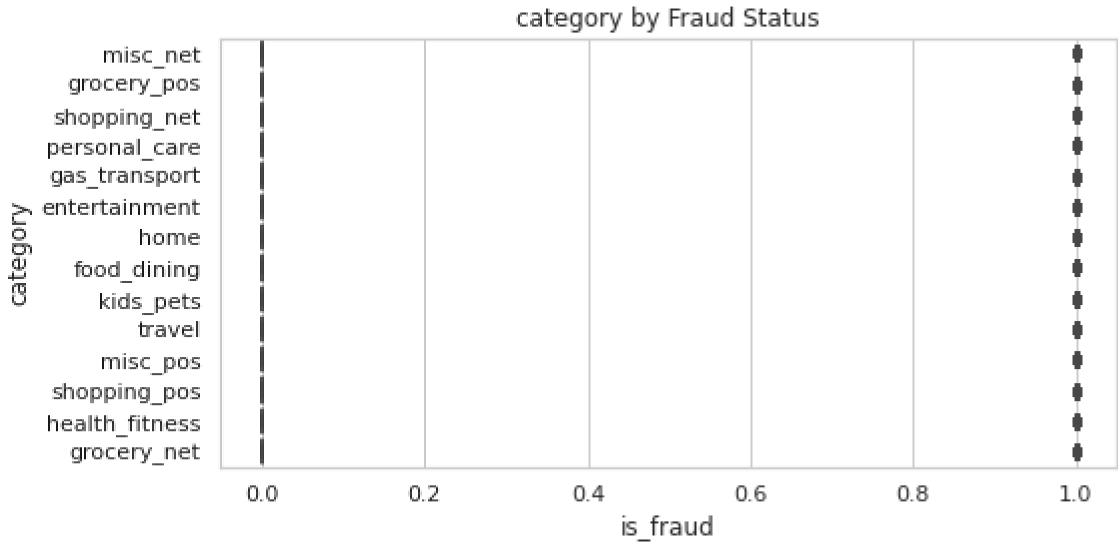




- Segment-wise (e.g. by job or location) comparisons.

```
# Transaction amount by gender
plt.figure(figsize=(6,4))
sns.boxplot(x='gender', y='amt', data=df)
plt.title("Transaction Amount by Gender")
plt.tight_layout()
plt.savefig('plots/gender_amt_boxplot.png')
plt.show()

# Fraud rate by category
plt.figure(figsize=(10,4))
fraud_rates = df.groupby('category')
        ['is_fraud'].mean().sort_values(ascending=False)
fraud_rates.plot(kind='bar', color='crimson', title='Fraud Rate by
        Transaction Category')
plt.ylabel("Fraud Rate")
plt.tight_layout()
plt.savefig('plots/category_fraud_rate.png')
plt.show()
```





4.3. Key EDA Insights

- Transaction values are highly skewed; a few large outliers.
- Fraud concentrated in specific categories.
- Geographic, time, and group-based analysis reveal patterns useful for fraud detection.

---

5. Major Visual Data Insights & Interactive Reporting (Tableau)

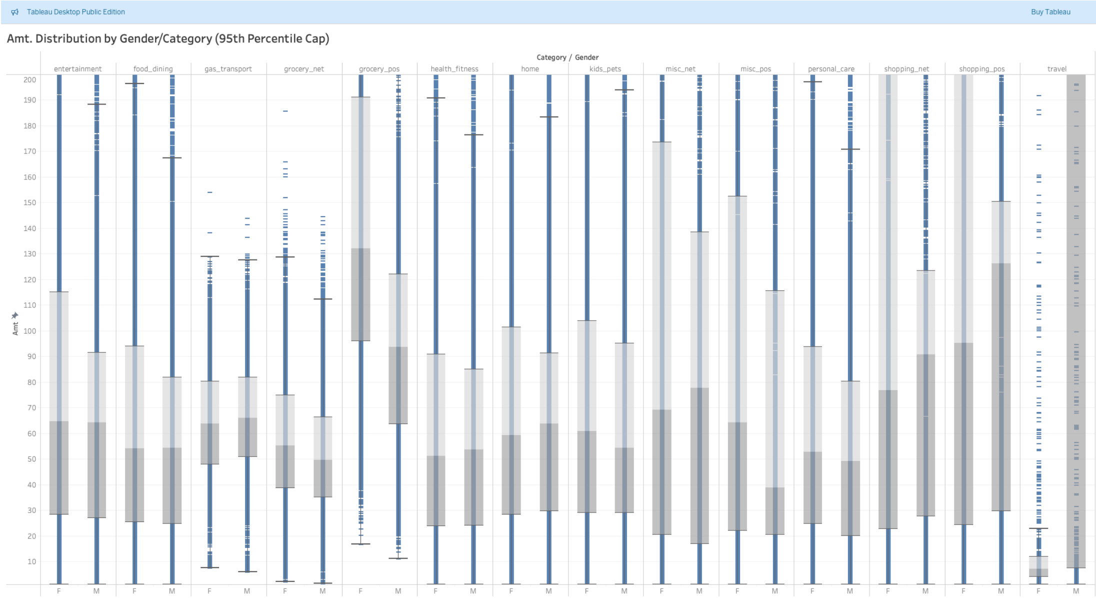
5.1 Workflow Overview

Interactive Tableau dashboards are created for in-depth fraud analysis and transparency, enabling dynamic data slicing and stakeholder exploration.

5.2 Step-by-Step Tableau Implementation

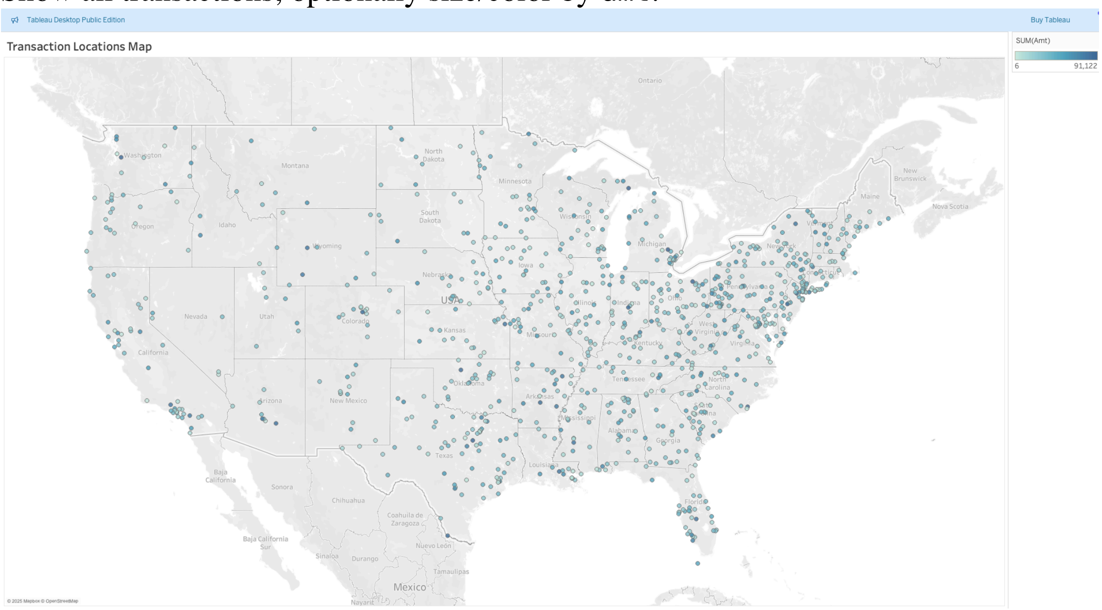
A. Box & Whisker Plot: Transaction Amount by Gender & Category

1. New worksheet: drag `category` (Columns), `amt` (Rows), and `gender` (Color/Columns).
2. Select "Box-and-Whisker Plot" in Show Me.
3. Edit tooltips, labels, and titles for clarity.



B. Map Visualization: All Transaction Locations

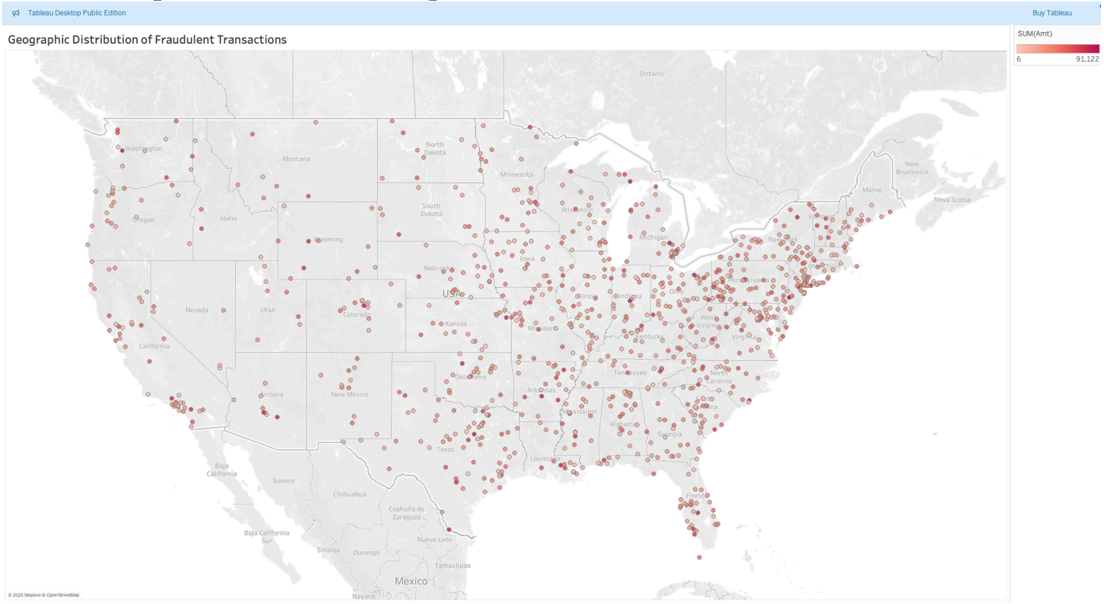
1. Ensure `lat` and `long` are geographic.
2. Drag `lat` (Rows) and `long` (Columns) for map.
3. Show all transactions; optionally size/color by `amt`.



C. Fraud Map

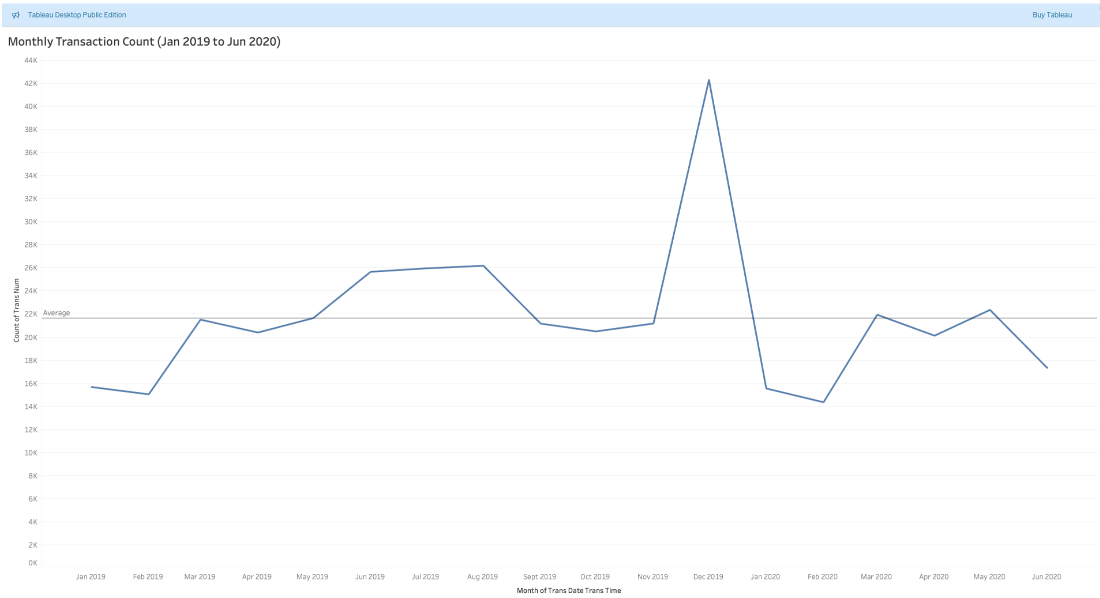
1. Duplicate location map.
2. Filter by `is_fraud` = 1.

3. Use red for points; enhance tooltips.



D. Monthly Trend Chart

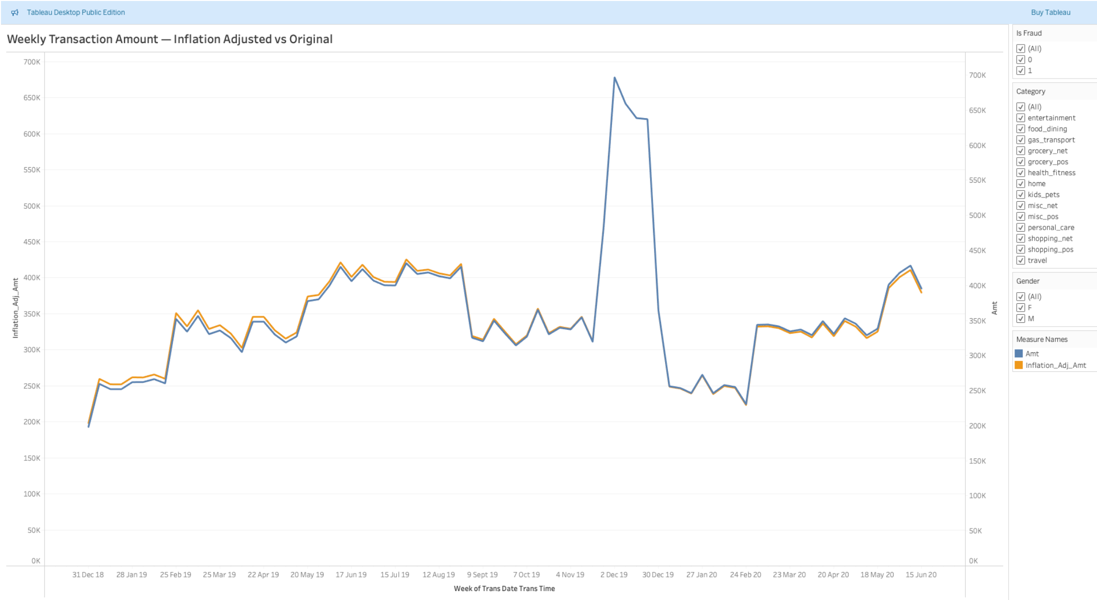
1. Drag `trans_date_trans_time` (Columns), set to Month.
2. Drag `trans_num` (Rows, as COUNT).
3. Use Line chart, label axes.



E. Inflation Adjustment Analysis

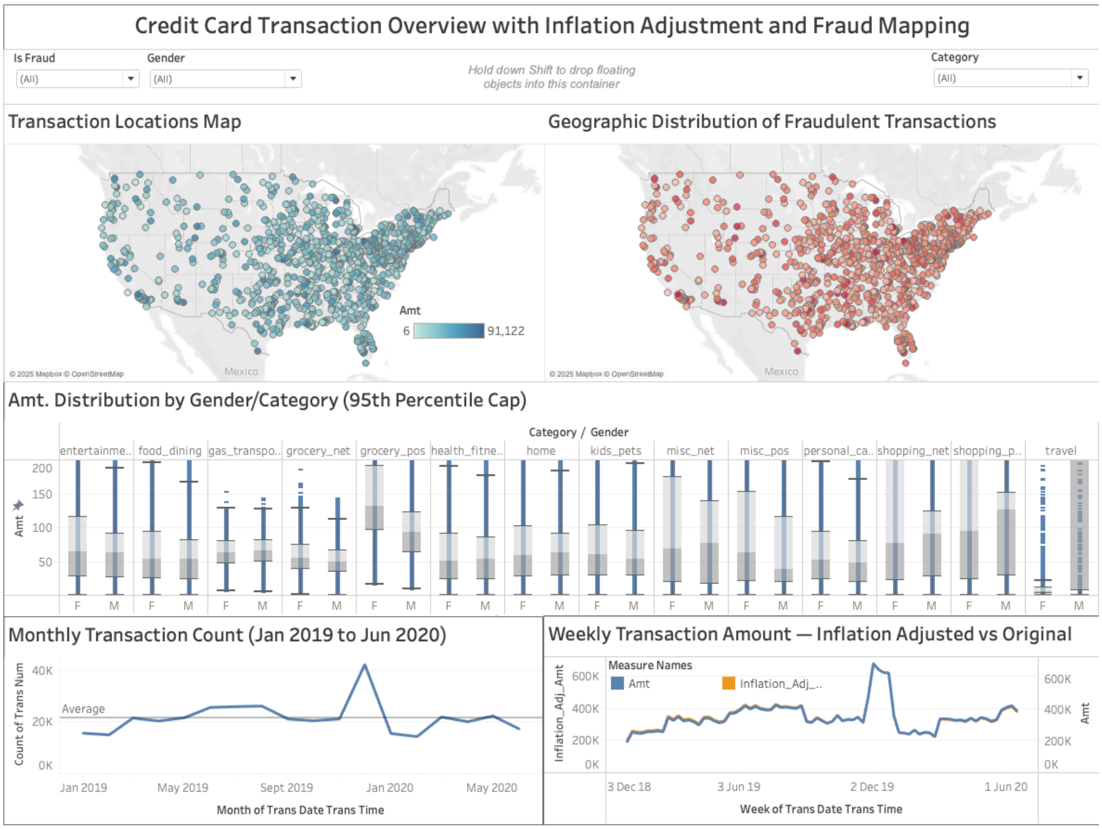
1. Create `Inflation_Adj_Amt` field: `[Amt] / POWER(1 + 0.03, DATEDIFF('month', DATE("2019-01-01"), DATETRUNC('month', [Trans Date Trans Time])) / 12)`

2. Chart by week: use line(s) for SUM(adjusted) and SUM(original) amounts.



F. Interactive Dashboard Assembly

1. Create new Dashboard.
2. Drag and arrange all key worksheets.
3. Add slicers for Gender, Category, Is_Fraud:
   - Show filter for each field.
   - Use "Apply to all worksheets using this data source."
4. Add titles, descriptions, and adjust for clarity.
5. Format tooltips; add project/author info.
6. Publish via "Save to Tableau Public As…"



6. Summary and Recommendations

The exploratory data analysis and visualisation efforts have yielded a comprehensive understanding of the credit card transaction dataset and the patterns indicative of fraudulent behavior. Below is a detailed summary of key findings and actionable recommendations aimed at strengthening SecureGuard's fraud detection capabilities.

Key Findings

- **Skewed Transaction Amounts:** Transaction values are heavily right-skewed, with most transactions being low-value and a small number of high-value outliers. These extreme values have significant implications for modeling and anomaly detection.
- **Fraud Concentration:** Fraudulent transactions constitute less than 1% of all transactions, predominantly occurring in categories such as online shopping (`shopping_net`), grocery POS, and miscellaneous networks. The gender distribution among fraud cases is nearly even, but crime patterns vary subtly by category and gender.
- **Geographical Clustering:** Spatial analysis revealed clusters of fraudulent activity, especially in high-transaction volume states such as Texas, New York, and Pennsylvania. Geographic insights can inform region-specific monitoring and intervention strategies.
- **Temporal Patterns:** Volume fluctuations across time demonstrate seasonality and trend effects. Inflation adjustments reveal genuine changes in transaction values beyond economic inflation, enhancing temporal models.
- **Predictive Feature Identification:** Correlations and distributional differences highlight features such as transaction amount, transaction timestamp (hour), geographic coordinates, and category as valuable inputs for fraud predictive models.

Recommendations

1. **Outlier Treatment:** Develop strategies—such as transformations or capping—to mitigate the influence of extreme transaction and city population values in predictive modeling and reporting.
2. **Focus on High-Risk Categories:** Allocate analytical and monitoring resources disproportionately to categories with elevated fraud risk (e.g., online shopping), employing adaptive rules and machine learning models attuned to these segments.
3. **Leverage Geographic Insights:** Integrate spatial fraud patterns into real-time monitoring systems, enabling SecureGuard to deploy localised alerts, investigations, and possibly enhanced verification in fraud hot-spots.
4. **Incorporate Inflation and Time Trends:** Adjust for inflation and seasonal effects within fraud detection frameworks to ensure historical comparisons and trends reflect true transactional risk changes.
5. **Deploy Interactive Dashboards:** Utilise the Tableau dashboards designed in this pipeline to empower analysts with dynamic filters for gender, category, and fraud status, improving anomaly investigation turnaround and transparency.
6. **Further Enhancements:**

- Expand data sources to include merchant risk profiles, customer demographics, and behavioural signals.
- Explore and validate advanced machine learning models leveraging the identified predictive features.
- Establish ongoing feedback loops from fraud investigations to continuously refine detection rules and model accuracy.