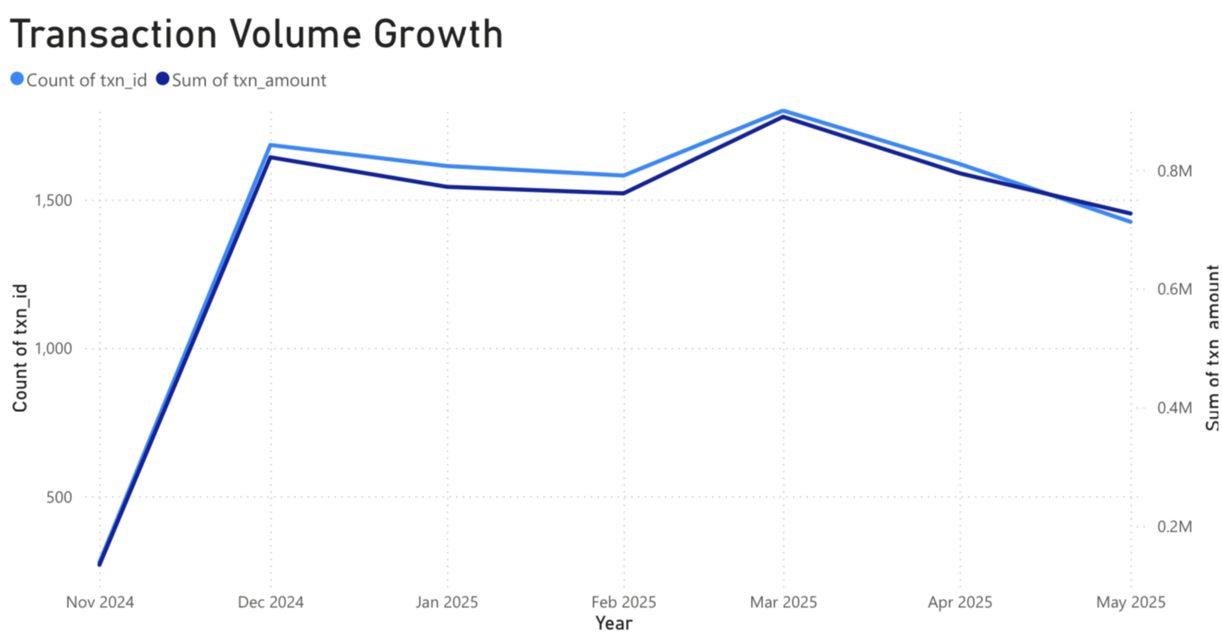
Assignment - Junior Data Analyst Role
Sharon Benjamin

Task 2: Exploratory Analysis

1. **Analysing overall transaction volume (count & value) grown over the last 6 months:**

The following graph shows the overall **transaction volume grown** over the last 6 months:



From **Nov 2024 to Mar 2025**, both **transaction count and value grew significantly**, showing **strong upward momentum**.

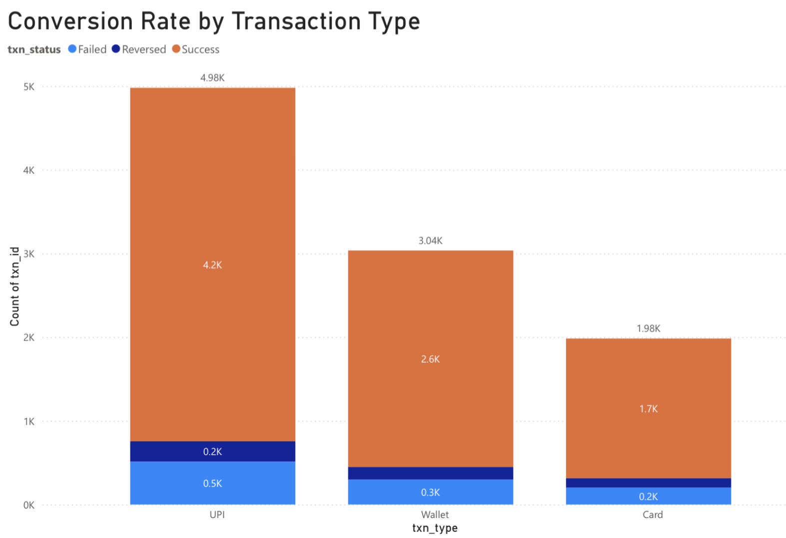
March 2025 marked the **peak** in both metrics.

The **last two months** (April and May 2025) show a **slight cooling off**, but the volumes remain **consistently higher than in November 2024**, indicating **overall positive growth** across the 6-month period.

2. Conversion rate across transaction types

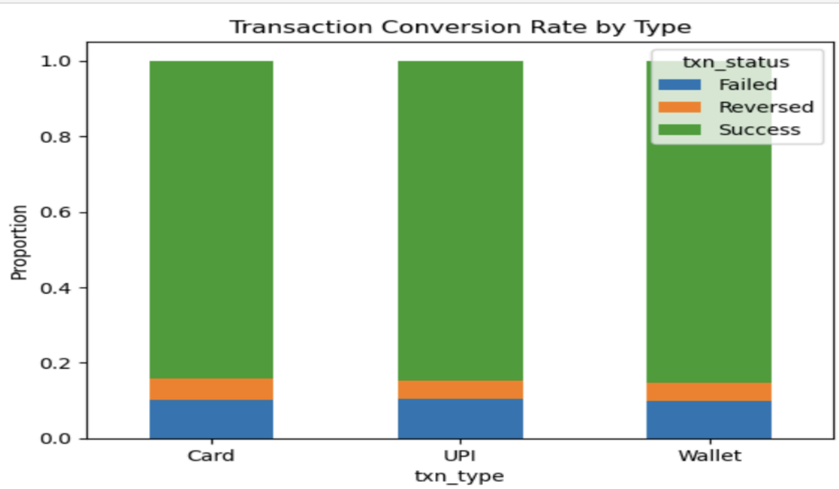
The Transaction types being used are - Cards, wallets and cards. The data has been analysed and stacked column charts have been generated using Powerbi as well as Python Pandas.

Plot Generated using Powerbi:



Plot generated using Pandas:

```python
conversion = df.groupby('txn_type')['txn_status'].value_counts(normalize=True).unstack()
conversion.plot(kind='bar', stacked=True)
plt.title('Transaction Conversion Rate by Type')
plt.ylabel('Proportion')
plt.xticks(rotation=0)
plt.show()
```



These are the data insights:

UPI

- Success: 4.2K
- Failed: 0.5K
- Total: 4.2K + 0.5K + 0.2K (Reversed) = 4.9K
- Conversion Rate = 4.2K / 4.9K ≈ 85.7%

Wallet

- Success: 2.6K
- Failed: 0.3K
- Total: 2.6K + 0.3K + 0.1K (Reversed) = 3.0K
- Conversion Rate = 2.6K / 3.0K ≈ 86.7%

Card

- Success: 1.7K
- Failed: 0.2K
- Total: 1.7K + 0.2K + 0.08K (Reversed) = ~1.98K
- Conversion Rate = 1.7K / 1.98K ≈ 85.9%

Summary:

All three transaction types show strong conversion rates:

- Wallet: 86.7%
- Card: 85.9%
- UPI: 85.7%

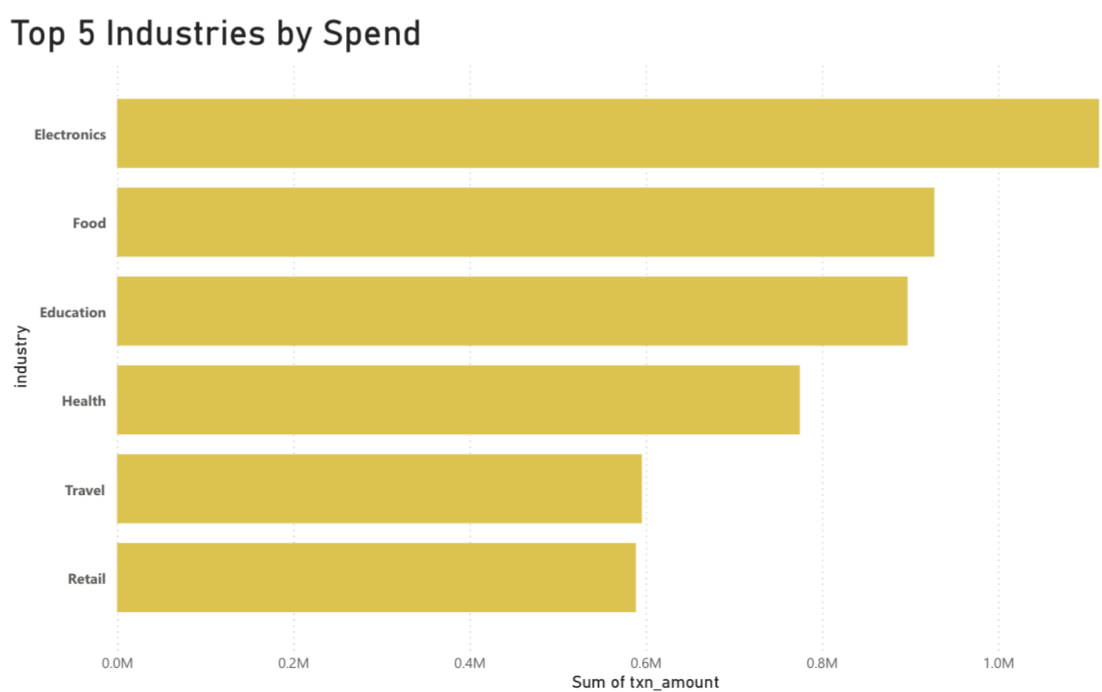
Wallets lead slightly in terms of success rate.

3. **What are the top 5 industries where users spend the most?**

The Retail, Health, Education, Food and Education are present in the dataset. These industries have the highest total transaction values, with **Electronics** being the highest among them.

Visualisation:

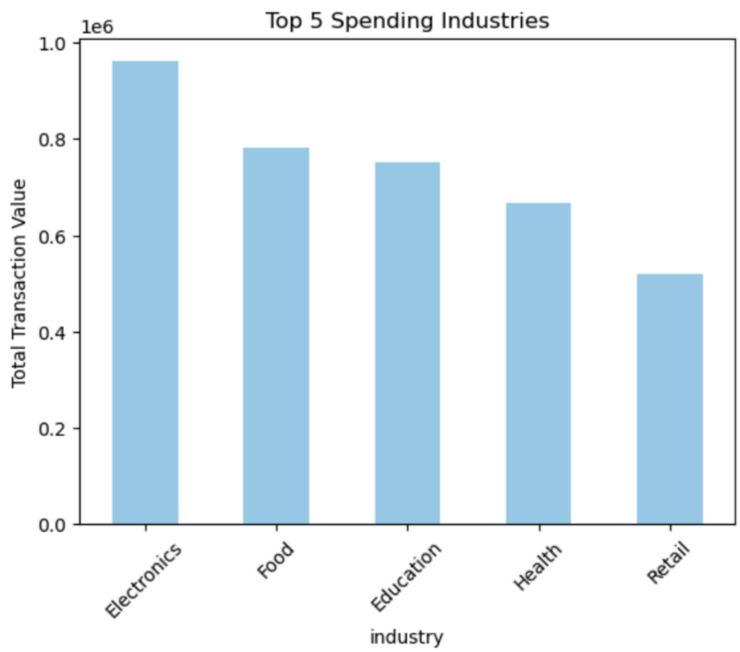
Plot generated using Powerbi:



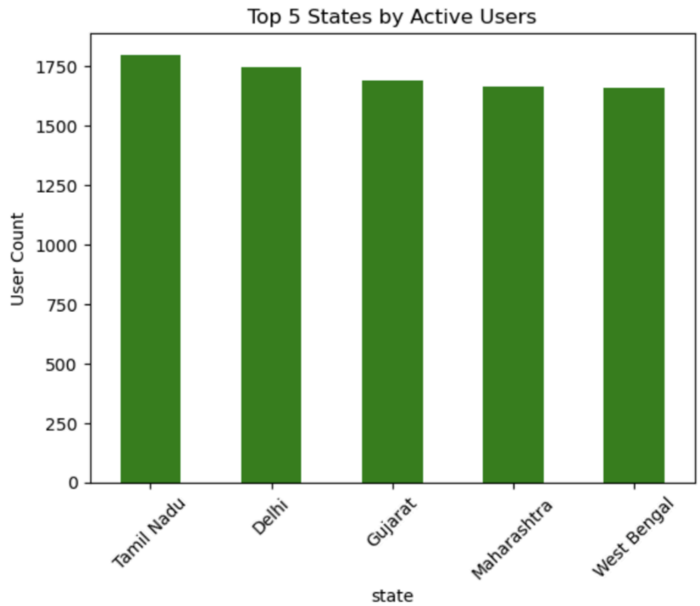
Plot Generated using Pandas in python:

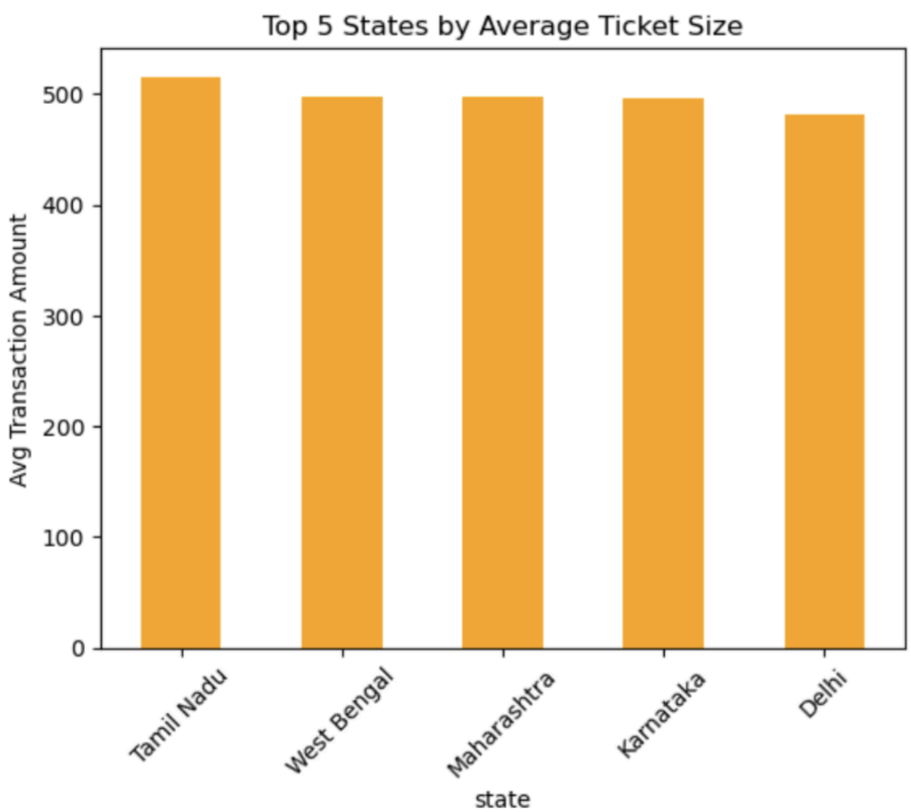
```
[12]: industry_spend = df[df['txn_status'] == 'Success'].groupby('industry')['txn_amount'].sum()
      top_5_industries = industry_spend.nlargest(5)

      top_5_industries.plot(kind='bar', color='skyblue')
      plt.title('Top 5 Spending Industries')
      plt.ylabel('Total Transaction Value')
      plt.xticks(rotation=45)
      plt.show()
```



4. Which states have the highest **active users** and **highest average ticket size**?





If a user has at least one transaction, they're considered active.

So, to find active users by state, we:

- Group by state
- Count the number of distinct user_ids

To find the Average Ticket Size = Total transaction amount / Number of transactions

We calculate this per state, and then sort to get the top 5.

Conclusions: State with the Highest Number of Active Users:

- **Tamil Nadu** has the highest number of active users.
- State with the Highest Average Ticket Size (Avg Transaction Amount):
- **Tamil Nadu** also has the highest average ticket size.

So, **Tamil Nadu** leads in both **user activity** and **average spending per transaction**.

Task 3: Fraud Detection Heuristics

Rule based analysis:

1. **High-value transactions by users with unverified KYC**

   Logic: If a user's KYC status is not verified or "pending" , and the transaction amount exceeds a certain threshold (e.g., Rs.4,000), flag it as suspicious.

   SQL Pseudo Code:

   SELECT *,

   CASE

   WHEN kyc_verified = 0 AND transaction_amount > 3000 THEN 'suspicious_high_value_unverified'

   ELSE 'normal'

   END AS fraud_flag

   FROM transactions;

   Pandas code snippet and results:

```
[37]: high_value_unverified = df[(df['kyc_status'] != 'Verified') & (df['txn_amount'] > 3000)].copy()
high_value_unverified['fraud_flag'] = 'High-Value Unverified KYC'

# printing results
print("High-Value Transactions by Unverified KYC:")
print(high_value_unverified[['txn_id', 'user_id', 'txn_amount', 'kyc_status', 'timestamp']])
```

```
High-Value Transactions by Unverified KYC:
         txn_id user_id  txn_amount kyc_status            timestamp
291   TXN000291  U00758     3020.92    Pending 2025-05-26 05:29:13
1859  TXN001859  U00771     3680.89    Pending 2025-03-16 04:20:07
2116  TXN002116  U00892     3657.95    Pending 2025-04-14 02:09:58
4050  TXN004050  U00774     3631.84    Pending 2025-03-24 10:56:54
4435  TXN004435  U00270     3425.48    Pending 2025-02-12 15:00:30
5009  TXN005009  U00869     3416.56    Pending 2025-01-03 06:23:37
5388  TXN005388  U00091     3249.48    Pending 2025-01-10 14:05:34
8297  TXN008297  U00431     3039.18    Pending 2025-01-28 10:31:09
9267  TXN009267  U00383     3116.50    Pending 2024-12-25 15:44:34
```

2. **Sudden spikes in transaction frequency for a user or merchant**

   Logic: If a user or merchant makes 5 or more transactions within 10 minutes, it could be a bot or suspicious transaction.

   SQL Pseudo Code:

```
SELECT

t1.user_id,

t1.txn_id,

t1.timestamp,

'User Transaction Spike' AS fraud_flag

FROM transactions t1

JOIN (

SELECT

user_id,

COUNT(*) AS txn_count,

MIN(timestamp) AS start_time,

MAX(timestamp) AS end_time

FROM transactions

GROUP BY user_id, DATE_TRUNC('minute', timestamp), merchant_id

HAVING COUNT(*) >= 5

AND MAX(timestamp) - MIN(timestamp) <= INTERVAL '10 minutes'

) suspicious_users

ON t1.user_id = suspicious_users.user_id

AND t1.timestamp BETWEEN suspicious_users.start_time AND
suspicious_users.end_time;
```

Pandas code and results snippet:

```
[38]: df_sorted = df.sort_values(['user_id', 'timestamp'])
spike_user_ids = set()

for user, group in df_sorted.groupby('user_id'):
    times = group['timestamp'].tolist()
    for i in range(len(times) - 4):
        if (times[i+4] - times[i]) <= timedelta(minutes=10):
            spike_user_ids.add(user)
            break

spike_user_txns = df[df['user_id'].isin(spike_user_ids)].copy()
spike_user_txns['fraud_flag'] = 'User Transaction Spike'

print("\nUsers with Transaction Spikes:")
print(spike_user_txns[['txn_id', 'user_id', 'timestamp']].head())
```

```
Users with Transaction Spikes:
Empty DataFrame
Columns: [txn_id, user_id, timestamp]
Index: []
```

***Returns Empty DataFrame because no transactions that fit the rule.**

3. **Repeated failed transaction within 5 minutes.**

   Logic: If a user fails 3 or more transactions within 5 minutes, flag for suspicious retry patterns.

   SQL Pseudo Code:
   SELECT

   t1.user_id,

   t1.txn_id,

   t1.timestamp,

   'Repeated Fails in 5 mins' AS fraud_flag

   FROM transactions t1

   JOIN (

SELECT

user_id,

COUNT(*) AS fail_count,

MIN(timestamp) AS start_time,

MAX(timestamp) AS end_time

FROM transactions

WHERE txn_status = 'Failed'

GROUP BY user_id, DATE_TRUNC('minute', timestamp)

HAVING COUNT(*) >= 3

AND MAX(timestamp) - MIN(timestamp) <= INTERVAL '5 minutes'

) suspicious_fails

ON t1.user_id = suspicious_fails.user_id

AND t1.timestamp BETWEEN suspicious_fails.start_time AND suspicious_fails.end_time

WHERE t1.txn_status = 'Failed';

Pandas Code and results snippet:

```
[44]:
failed_txns = df[df['txn_status'] == 'Failed'].copy()
failed_txns = failed_txns.sort_values(['user_id', 'timestamp'])

flagged_failed_users = set()

for user, group in failed_txns.groupby('user_id'):
    times = group['timestamp'].tolist()
    for i in range(len(times) - 2):
        if (times[i+2] - times[i]) <= timedelta(minutes=5):
            flagged_failed_users.add(user)
            break

repeated_failed_txns = df[(df['user_id'].isin(flagged_failed_users)) & (df['txn_status'] == 'Failed')].copy()
repeated_failed_txns['fraud_flag'] = 'Repeated Fails in 5 mins'

# printing
print("\nRepeated Failed Transactions (Same User):")
print(repeated_failed_txns[['txn_id', 'user_id', 'timestamp']].head())
```

```
Repeated Failed Transactions (Same User):
Empty DataFrame
Columns: [txn_id, user_id, timestamp]
Index: []
```

Task 4: Business Recommendations

- **Improving User Retention:**

  Data Insight: States with the most **active users** (distinct users with at least one successful transaction):

  Delhi (176), **Tamil Nadu (174)**, **Gujarat (171)**, **Maharashtra (167)**, **West Bengal (163)**

  Recommendation:

  Launch **state-specific loyalty and cashback campaigns** targeting these high-engagement states. This leverages already active markets and encourages repeat usage.

  Increase marketing and ad-campaigns in states with lesser traction as well to improve brand awareness.

- **Promote Wallet & UPI Over Cards**

  Data Insight: Success rates by transaction type:

  Wallet: 85.2%

  UPI: 84.8%

  Card: 84.1%

  Recommendation:

  - Offer cashbacks or rewards on wallets and UPI that are competitive with ones that cards offer.
  - Implement a reward points system for UPI transactions that can be redeemed with airlines, restaurants, online stores etc.
  - Promote safe and secure transactions through wallets.
- **Target High-Spend Industries for Marketing Partnerships**

Data Insight: Highest spending trends seen in the **Electronics Industry (Sum of transaction amount: Rs.11,14,391)**

Recommendation:

- Have tie-ups with organisations from the Electronics Industry
- Lookout for sponsorship opportunities to increase brand awareness and gain traction
- Have companies using our products to promote.

Data Insight: Encourage KYC Verification with Instant Benefits

- **28.8% of users** still have **Pending KYC**

Recommendation:

- Offer instant wallet credit (₹50-100) or exclusive features (like higher transaction limits) for completing KYC. This improves trust and reduces fraud exposure.
- Introduce KYC-based tiered transaction caps, e.g.: Unverified: ₹2,000 max per txn, Verified: ₹50,000 max. Also trigger manual or automated reviews for unusual behavior from unverified users.