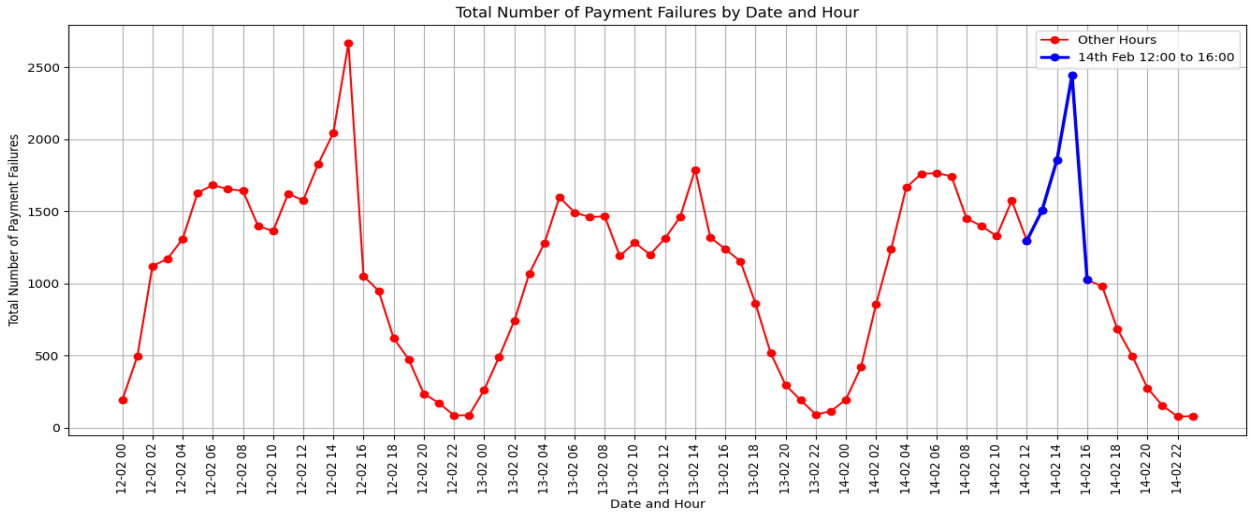
My solution to the Payments Failure Case-Study

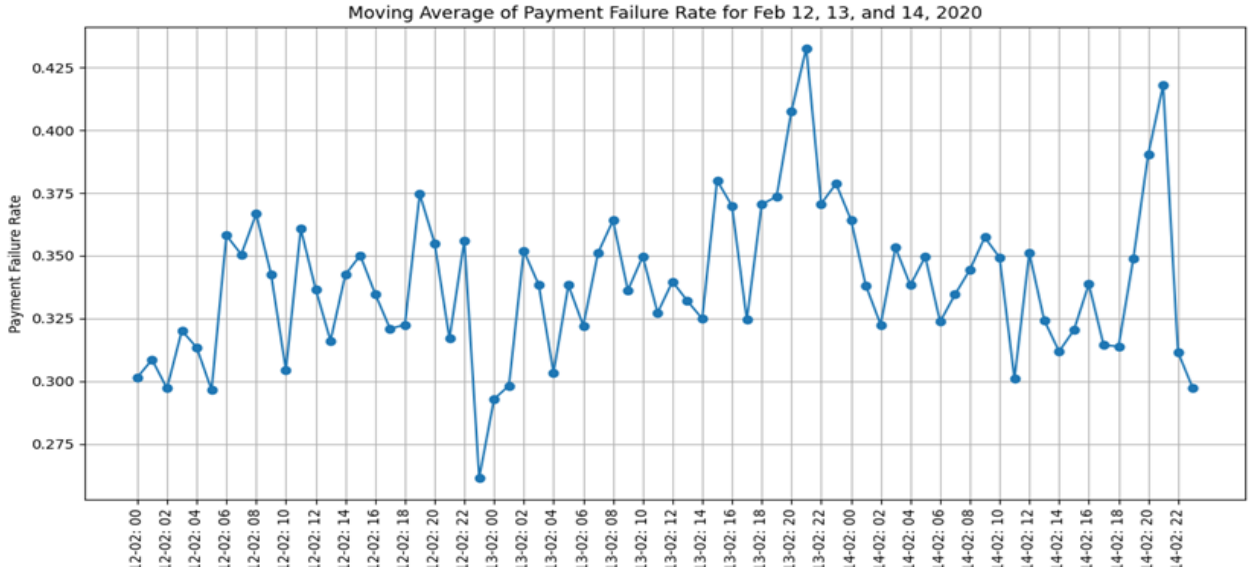
Q 1.1.1: Which dimension combination caused the issue?
Explore the data and visualization to understand when the issue (possibly a significant number of failures in transactions) happened and `which combination of dimension` (pmt, pg, bank and sub_type) has the impact.

Tip: Identify the method to detect an anomaly in a metric across 4+ dimensions and apply that method to find the above.

A 1.1.1:

First, let's start by visualizing the payment failures by the hour across the 3 dates.





Analyzing spikes in Moving Averages/Total payment failures through visualization:

In the chart that shows **Total Number of Payment Failures** by Date & Hour, we observe a seasonal trend across the 3 days. A peak (spike) in payment failures is observed each day**, >2500 failures on 12th Feb @ 15:00, ~1700 failures on 13th Feb @ 14:00, and ~2500 failures on 14th Feb @ 15:00.**

In the **Moving Average of Payment Failure Rate chart**, we see that Payment Failure Rate has peaked at **43% *Average* Payment Failure Rate** around **13th February 21:00**. This increase started around 13th February 17:00 and kept increasing till it peaked. Then there was a huge upswing again starting at **42% *Average* Payment Failure Rate** around **14th February 18:00**, continuing increasing till **14th February 21:00**.

Anomaly analysis to identify features having most impact on Payment Failures:

I used Isolation Forest ML model to understand which combination of features among Payment Gateway (pg), Payment Method Type (pmt), Payment Subtype (sub_type), Bank (bank).

1. **Observing Payment Parameters** (pg, pmt, sub_type, bank) **associated Overall across all 3 days that were associated with the highest failure rates**:

i. Observing all 4 payment parameters combined:

a. PhonePe, Wallet, REDIRECT_WALLET_DEBIT, PhonePe with 98.10% Failure Rate.
b. PayPal, Wallet, REDIRECT_WALLET_DEBIT, PayPal with 96.04% Failure Rate.
c. PayTM_V2, UPI, UPI_COLLECT, UPI with 92.75% Failure Rate.

ii. Observing each of the 4 payment parameters separately:

a. PAYTM_UPI, PhonePe, PayPal Payment Gateways with 100%, 98%, 96% failure rates respectively.
b. NB Payment Type with 72.73% failure rate.
c. UPI_PAY with 98.14% failure rate.
d. PhonePe, PayPal, NB_BOB with 98.12%, 96.04%, 95.27% failure rates respectively.

2. **Observing Payment Parameters** (pg, pmt, sub_type, bank) **associated in the spike between 14th February 12:00 to 15:00 PM**

i. Observing all 4 payment parameters combined:

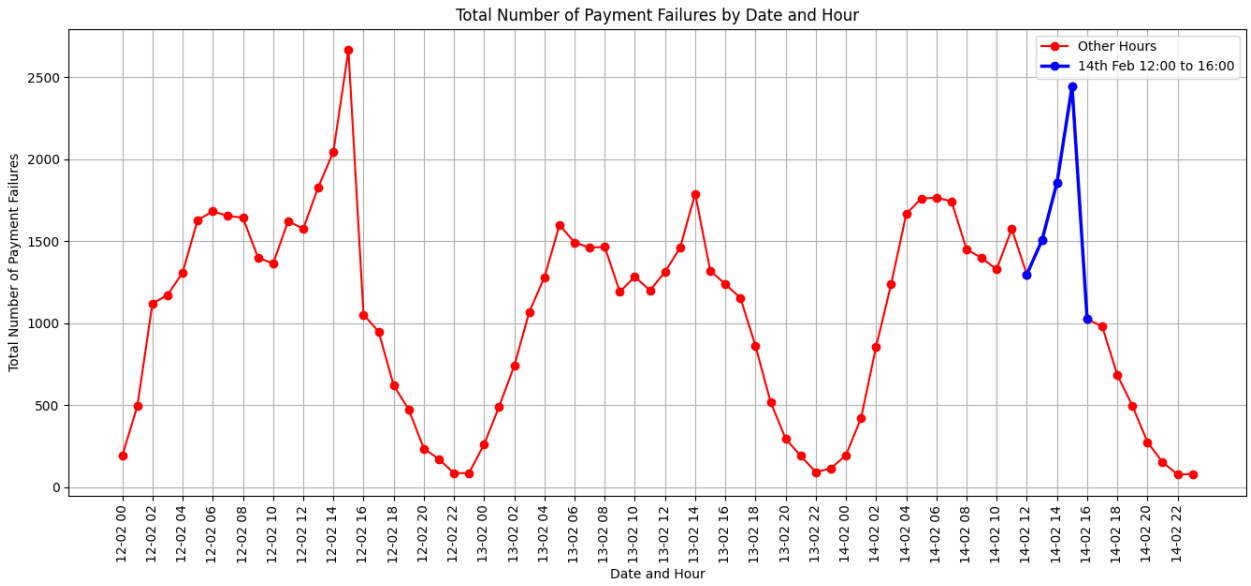
a. Freecharge, Wallet, REDIRECT_WALLET_DEBIT, Freecharge at 85.7% Failure Rate.
b. AmazonPay, Wallet, REDIRECT_WALLET_DEBIT, AmazonPay at 83% Failure Rate.
c. Mobikwik, Wallet, REDIRECT_WALLET_DEBIT, Mobikwik with 77.77% Failure Rate.

ii. Observing each of the 4 payment parameters separately:

a. AmazonPay, Freecharge, Mobikwik Payment Gateways with 88.88%, 85.71%, 78.57% failure rates respectively.
b. NB with 71.32% failure rate.
c. REDIRECT_DEBIT_WALLET sub_type with 89.4% failure rate.
d. AMEX, NB_SBI, AmazonPay banks with 91.66%, 90%, 88.88% failure rates.

Q 1.1.2: When did the issue happen?
What is the `starting hour` and `ending hour` of the reported issue? Illustrate your answer with a plot.

A 1.1.2: For this question, I am focusing on the spike in payment failures specifically on February 14th, as the merchant complaint calls were reported on that date.



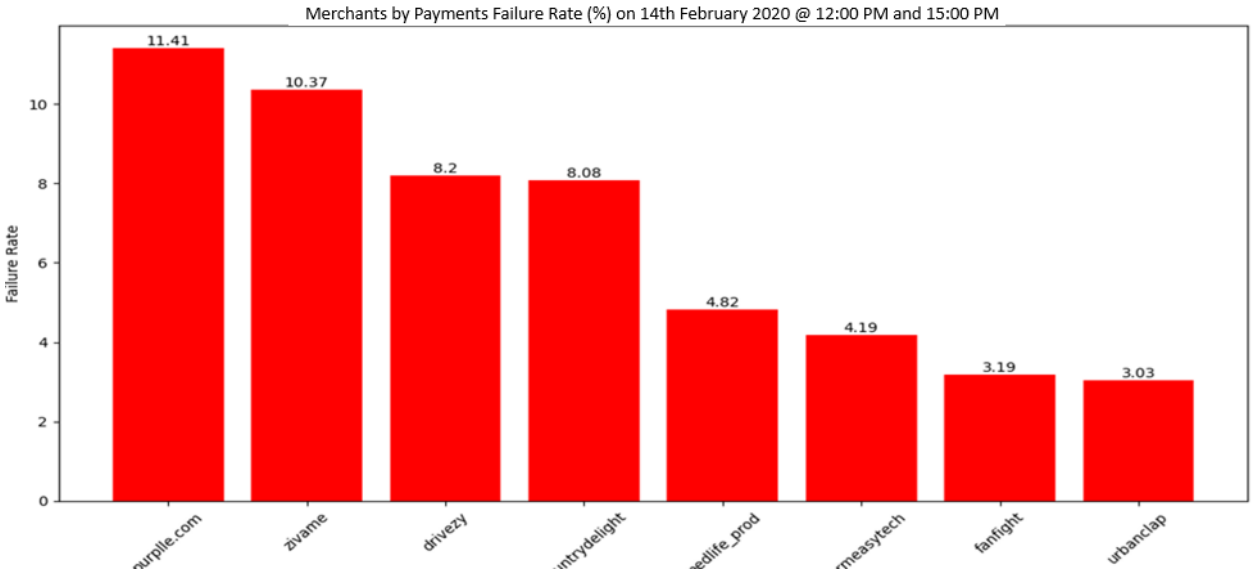
On 14th Feb, there was a huge spike in Total Number of Payment Failures observed which started at 12:00 PM, peaked at 15:00 PM, and dropped at 16:00 PM.

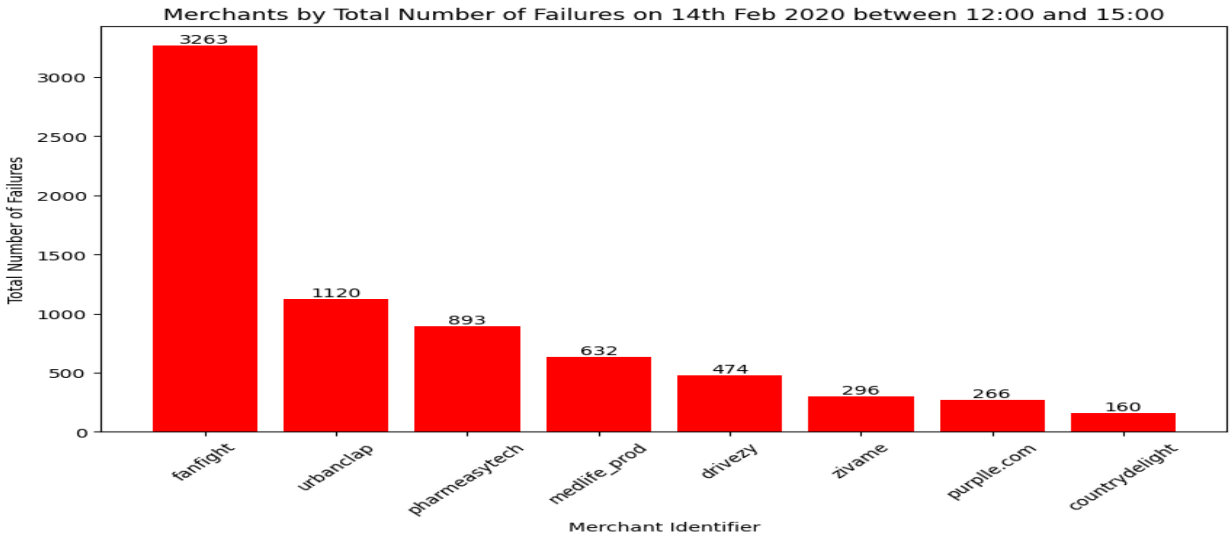
Starting Time of Issue on 14th Feb: **12:00 PM**

Ending Time of Issue on 14th Feb: **16:00 PM.**

Q 1.1.3: For whom did the issue happen? Which all `merchants` were impacted because of the issue and which all merchants were not? Illustrate your answer with a plot and a table.

A 1.1.3:





Here, we observe that all the 8 merchants that made payments in the spike on 14th February 2020 between 12:00 PM and 15:00 PM, but were affected with different impacts.

When considering the Payment Failure Rate (%) in this time period, purplle.com, zivame, drivezy, countrydelight were affected with 11.41%, 10.37%, 8.2%, 8.08% respectively.

When considering the Total Number of Failed Payments in this time period, fanfight was disproportionately affected with 3263 failures.

Merchant	**Failure Rate (%)**	**Total Number of Failures**
fanfight	3.19	3263
urbanclap	3.03	1120
pharmeasytech	4.19	893
medlife_prod	4.82	632
drivezy	8.2	474
zivame	10.37	296
purplle.com	11.41	266
countrydelight	8.08	160

Q 1.2: How could we have detected this issue before the merchant reported? Almost all the merchants expect us to detect the issue before they detect or their customers detect. What are your thoughts on how this can be accomplished? Explain with code/pseudo-code

A 1.2:

In this spike on 14th Feb 2020, between 12:00 and 15:00, these were the payment configurations that resulted in the highest payment failure rates:
a. Freecharge, Wallet, REDIRECT_WALLET_DEBIT, Freecharge at 85.7% Failure Rate.
b. AmazonPay, Wallet, REDIRECT_WALLET_DEBIT, AmazonPay at 83% Failure Rate.
c. Mobikwik, Wallet, REDIRECT_WALLET_DEBIT, Mobikwik with 77.77% Failure Rate.

Steps to detect this issue before the merchant reported:

i. **Real-Time Monitoring**: Setup data pipelines to make the incoming Payments data as Stream data instead of bulk extracts. This will allow us to continuously monitor payment transaction data in near-real time.

ii. **Anomaly Detection**: Use machine learning models such as Isolation Forest used here to detect anomalies in key metrics such as the payment failure rate or total number of failed payments per merchant.

iii. **Threshold-Based Alerts**: Set thresholds for acceptable performance metrics (like payment failure rates or total # of failed payments per merchant) and trigger alerts if the metrics exceed these thresholds.

iv. **Automated Response**: Once an anomaly is detected, automatically notify relevant stakeholders (including merchants if action is required from their side) and provide diagnostic information for quick resolution.

Sample code to detect issues before merchants report it:

```
# Function to preprocess data
def preprocess_data(data):
  data['hr'] = pd.to_datetime(data['hr'])
  data['date_hour'] = data['hr'].dt.strftime('%Y-%m-%d %H')
  data['failure_rate'] = (1 - data['success'] / data['t'])
  return data

# Function to detect anomalies using Isolation Forest
def detect_anomalies(data):
  data_grouped = data.groupby('date_hour')['failure_rate'].mean().reset_index()
  iso_forest = IsolationForest(contamination='auto', random_state=42)
  data_grouped['anomaly_score'] =
iso_forest.fit_predict(data_grouped[['failure_rate']])
  anomalies = data_grouped[data_grouped['anomaly_score'] == -1]
  return anomalies

# Function to set thresholds and send alerts
def send_alerts(anomalies, threshold=0.5):
  # Check if any failure rate exceeds a certain threshold
  if not anomalies.empty and anomalies['failure_rate'].max() > threshold:
    print("ALERT: Anomalous payment failure rate detected!")
    # Send alerts to relevant stakeholders

# Main function for continuous monitoring
def continuous_monitoring():
  while True:
```

```
        # Fetch the latest transaction data
        transactions_data = fetch_transactions_data()

        # Preprocess the data
        processed_data = preprocess_data(transactions_data)

        # Detect anomalies
        anomalies = detect_anomalies(processed_data)

        # Send alerts if anomalies are detected
        send_alerts(anomalies)

        # Sleep for a specified interval before checking again (e.g., 5 minutes)
        time.sleep(300)

# Start the monitoring process
continuous_monitoring()
```