Double-click (or enter) to edit

```
# POC Project for Claims Automation Processing
#Install dependencies
!pip install pandas requests faker scikit-learn

import pandas as pd
import numpy as np
import requests
from faker import Faker
import random
from collections import defaultdict
from datetime import datetime, timedelta
from sklearn.ensemble import IsolationForest

np.random.seed(42)
random.seed(42)

# Get dog breeds
resp = requests.get("https://dog.ceo/api/breeds/list/all")
breeds = [f"{b}-{sb}" if sb else b
          for b, subs in resp.json()["message"].items()
          for sb in (subs or [None])]

# Procedures & cost ranges
procedures = [
    'Routine visit', 'Vaccines', 'Spay/neuter', 'Dental cleaning', 'X-ray/ultrasound',
    'Emergency surgery (ACL)', 'Emergency surgery (foreign body)', 'Emergency (bloat)',
    'Emergency surgery (hip replacement)', 'Surgery (Tumor Removal)'
]
cost_ranges = {
    'Routine visit': (50, 150), 'Vaccines': (20, 100), 'Spay/neuter': (200, 600),
    'Dental cleaning': (50, 200), 'X-ray/ultrasound': (300, 600),
    'Emergency surgery (ACL)': (400, 1000), 'Emergency surgery (foreign body)': (400, 1000),
    'Emergency (bloat)': (600, 1200), 'Emergency surgery (hip replacement)': (700, 2000),
    'Surgery (Tumor Removal)': (500, 1500)
}

fake = Faker()
provider_names = [fake.company() for _ in range(50)] + ["Unknown Vet", "Budget Clinic"]
comments = ['urgent', 'routine check', 'follow-up', 'client requested', 'multiple issues', 'repeat procedure']

num_customers = 500
customer_ids = [f"CUST{str(i).zfill(4)}" for i in range(num_customers)]
customer_names = [fake.name() for _ in range(num_customers)]

pet_ids_by_customer = defaultdict(list)
pet_counter = 1
for cust in customer_ids:
    for _ in range(random.randint(1, 3)):
        pet_ids_by_customer[cust].append(f"PET{str(pet_counter).zfill(5)}")
        pet_counter += 1

num_records = 10000
records = []
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (2.2.2)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (2.32.3)
Collecting faker
  Downloading faker-37.4.0-py3-none-any.whl.metadata (15 kB)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (1.6.1)
Requirement already satisfied: numpy>=1.23.2 in /usr/local/lib/python3.11/dist-packages (from pandas) (2.0.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests) (3.4.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests) (2.4.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests) (2025.6.15)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.15.3)
Requirement already satisf                                                          om scikit-learn) (1.5.1)
Requirement already satisf                                                          ges (from scikit-learn) (3.6.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas) (1.
Downloading faker-37.4.0-py3-none-any.whl (1.9 MB)
```

What can I help you build?

```
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.9/1.9 MB 61.9 MB/s eta 0:00:00
Installing collected packages: faker
Successfully installed faker-37.4.0
```

```
# Generate claims with random dates in last year
start_date = datetime.now() - timedelta(days=365)

for _ in range(num_records):
    cust_index = random.randint(0, num_customers - 1)
    cust_id = customer_ids[cust_index]
    cust_name = customer_names[cust_index]
    pet_id = random.choice(pet_ids_by_customer[cust_id])
    breed = random.choice(breeds)
    age_months = random.randint(2, 240)
    proc = random.choice(procedures)
    comment = random.choice(comments)
    provider = random.choice(provider_names)

    cost_low, cost_high = cost_ranges[proc]
    if random.random() < 0.05:
        cost = random.randint(2000, 5000)  # anomalously high
    elif random.random() < 0.02:
        cost = random.randint(1, 30)  # anomalously low
    else:
        cost = random.randint(cost_low, cost_high)

    # Random claim date within last year
    claim_date = start_date + timedelta(days=random.randint(0, 365))

    records.append({
        'customer_id': cust_id,
        'customer_name': cust_name,
        'pet_id': pet_id,
        'breed': breed,
        'age_in_months': age_months,
        'procedure': proc,
        'cost': cost,
        'provider': provider,
        'claim_comment': comment,
        'claim_date': claim_date
    })

df = pd.DataFrame(records)
df['age_in_years'] = df['age_in_months'] / 12
df['claim_month'] = df['claim_date'].dt.to_period('M')

# Rule 1: Cost outliers per procedure (mean ± 3 std dev)
cost_stats = df.groupby('procedure')['cost'].agg(['mean', 'std']).reset_index()
df = df.merge(cost_stats, on='procedure', how='left')

df['cost_outlier'] = ~df['cost'].between(df['mean'] - 3*df['std'], df['mean'] + 3*df['std'])

# Rule 2: Frequency anomaly - >5 claims per pet per month
claims_per_pet_month = df.groupby(['pet_id', 'claim_month']).size().reset_index(name='claims_count')
freq_anomaly = claims_per_pet_month[claims_per_pet_month['claims_count'] > 5]
freq_anomaly_set = set(zip(freq_anomaly['pet_id'], freq_anomaly['claim_month']))

df['freq_anomaly'] = df.apply(lambda r: (r['pet_id'], r['claim_month']) in freq_anomaly_set, axis=1)

# Compose anomaly reason column
def anomaly_reason(row):
    reasons = []
    if row['cost_outlier']:
        reasons.append('Cost outlier')
    if row['freq_anomaly']:
        reasons.append('High frequency claims')
    return '; '.join(reasons) if reasons else None

df['flag_reason'] = df.apply(anomaly_reason, axis=1)

# Optional: ML anomaly detection using IsolationForest on numeric features
from sklearn.preprocessing import LabelEncoder

enc_proc = LabelEncoder()
df['proc_enc'] = enc_proc.fit_transform(df['procedure'])

features = df[['cost', 'proc_enc', 'age_in_years']]
```

```
iso_forest = IsolationForest(contamination=0.05, random_state=42)
df['ml_anomaly'] = iso_forest.fit_predict(features)
# ml_anomaly == -1 means anomaly detected
df['ml_flag'] = df['ml_anomaly'] == -1

print(df.head(10))
```

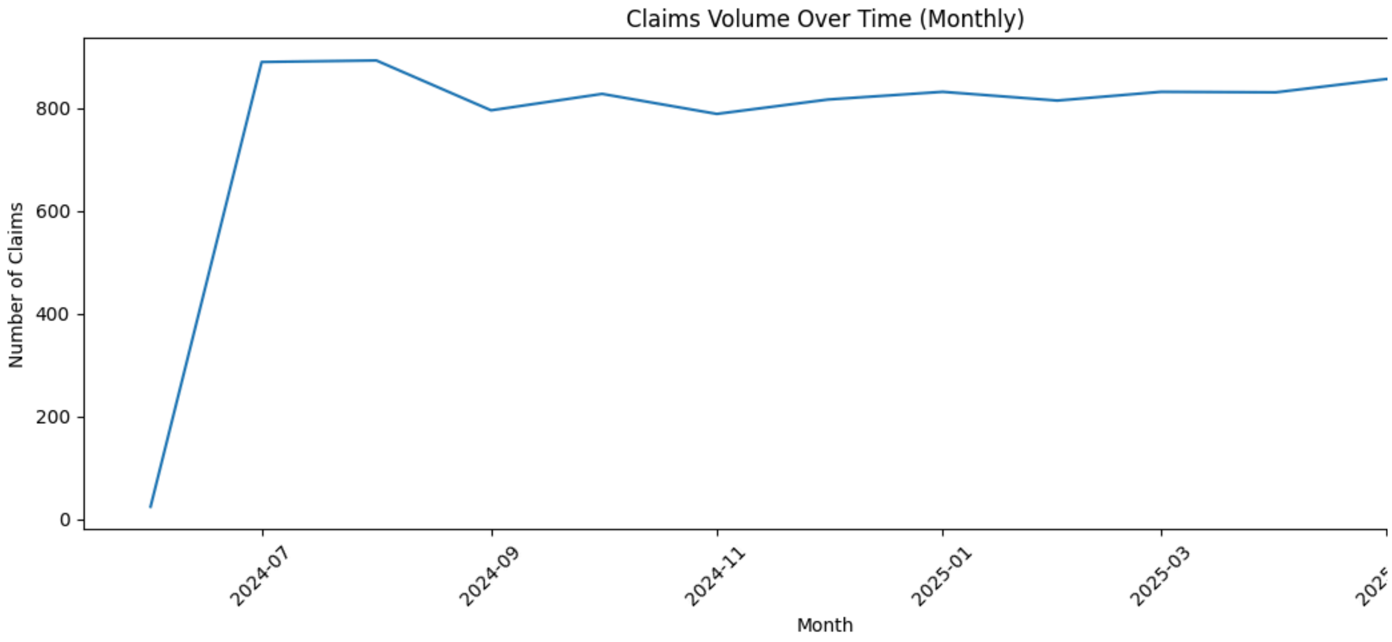
```
0    CUST0055  Brandon Fitzgerald  PET00107                beagle
1    CUST0114          Amy Wilson  PET00223           groenendael
2    CUST0347      Terri Mckinney  PET00685         pariah-indian
3    CUST0305    Richard Williams  PET00602               pitbull
4    CUST0130     Samuel Williams  PET00258   schnauzer-miniature
5    CUST0220        Peter Kelley  PET00440         pariah-indian
6    CUST0209        Erika Bryant  PET00418  retriever-flatcoated
7    CUST0383      Michael Taylor  PET00756                vizsla
8    CUST0402       Sheila Chavez  PET00792     sheepdog-shetland
9    CUST0043         Troy Benson  PET00085      springer-english

   age_in_months                            procedure  cost  \
0            221                    Emergency (bloat)   912
1            104              Emergency surgery (ACL)   921
2              9                             Vaccines    24
3            188              Emergency surgery (ACL)   990
4              2  Emergency surgery (hip replacement)  1103
5            161              Emergency surgery (ACL)   716
6            180                    X-ray/ultrasound   494
7            147                    X-ray/ultrasound   407
8            115                    Emergency (bloat)   773
9            171              Surgery (Tumor Removal)  1188

                   provider     claim_comment                claim_date  \
0     Rios, Banks and Calderon     routine check 2024-10-24 05:09:36.507464
1             Moreno-Mitchell         follow-up 2025-01-20 05:09:36.507464
2                Benjamin LLC         follow-up 2024-08-24 05:09:36.507464
3    Miller, Rivers and Suarez  client requested 2024-10-05 05:09:36.507464
4              Cannon-Rosales   multiple issues 2025-01-02 05:09:36.507464
5  Howell, Graham and Martinez  repeat procedure 2025-06-06 05:09:36.507464
6           Sanchez-Mcdonald   multiple issues 2025-06-11 05:09:36.507464
7                Peterson LLC  client requested 2025-02-05 05:09:36.507464
8   Brown, Cooper and Gonzalez  repeat procedure 2025-06-02 05:09:36.507464
9                   Davis Ltd         follow-up 2024-12-05 05:09:36.507464

   age_in_years claim_month         mean         std  cost_outlier  \
0     18.416667     2024-10   984.958418  560.229224         False
1      8.666667     2025-01   831.870314  669.802040         False
2      0.750000     2024-08   264.846380  843.588815         False
3     15.666667     2024-10   831.870314  669.802040         False
4      0.166667     2025-01  1423.970209  672.725056         False
5     13.416667     2025-06   831.870314  669.802040         False
6     15.000000     2025-06   602.179386  692.197668         False
7     12.250000     2025-02   602.179386  692.197668         False
8      9.583333     2025-06   984.958418  560.229224         False
9     14.250000     2024-12  1061.143281  572.107412         False

   freq_anomaly flag_reason  proc_enc  ml_anomaly  ml_flag
0         False        None         1           1    False
1         False        None         2           1    False
2         False        None         8           1    False
3         False        None         2           1    False
4         False        None         4           1    False
5         False        None         2           1    False
6         False        None         9           1    False
7         False        None         9           1    False
8         False        None         1           1    False
9         False        None         7           1    False
```

```
# Final decision logic:
def final_decision(row):
    if row['flag_reason']:
        return 'flagged_for_review'
    if row['ml_flag']:
        return 'flagged_for_review'
    return 'approved'

df['status'] = df.apply(final_decision, axis=1)

# Simulated manual review queue
manual_review_queue = df[df['status'] == 'flagged_for_review']

# Output summary
print("Claims Processing Summary:")
print(f"Total claims: {len(df)}")
print(f"Approved: {(df['status'] == 'approved').sum()}")
print(f"Flagged for review: {len(manual_review_queue)}")

print("\nSample flagged claims:")
display(manual_review_queue[['customer_id','pet_id','procedure','cost','provider','flag_reason','ml_flag','claim_comment','claim_
```

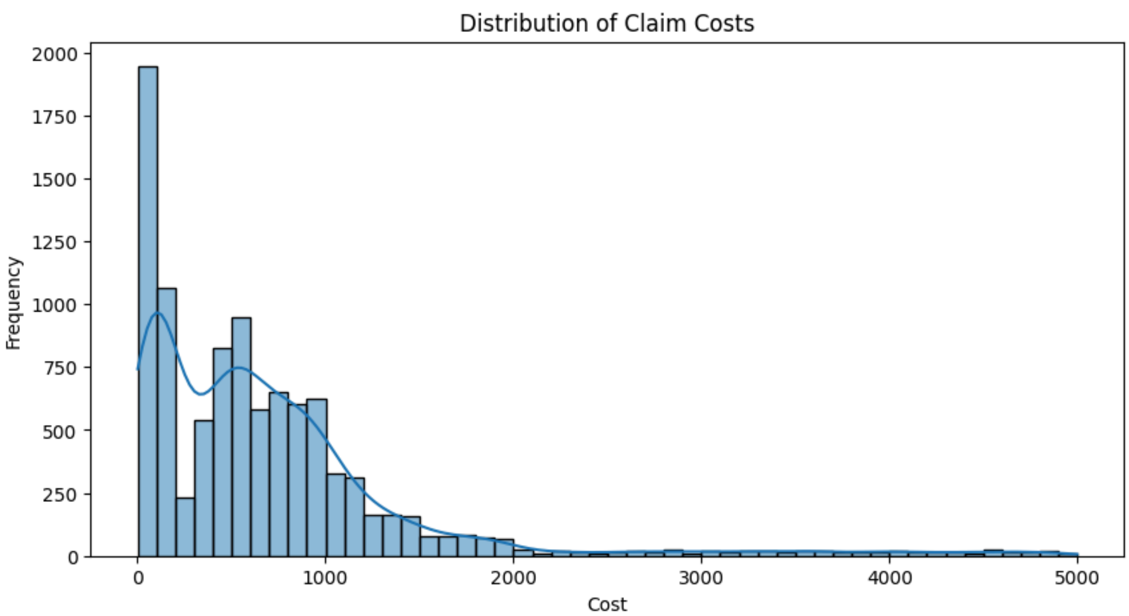
```
Claims Processing Summary:
Total claims: 10000
Approved: 9408
Flagged for review: 592

Sample flagged claims:
```

	customer_id	pet_id	procedure	cost	provider	flag_reason	ml_flag	claim_comment	claim_date
27	CUST0177	PET00357	Emergency surgery (hip replacement)	3720	Christensen, Johnston and Pineda	Cost outlier	True	repeat procedure	2025-03-07 05:09:36.507464
47	CUST0041	PET00082	Emergency surgery (ACL)	2937	Patel PLC	Cost outlier	True	multiple issues	2025-06-10 05:09:36.507464
100	CUST0341	PET00672	Emergency surgery (foreign body)	3234	Fowler Inc	Cost outlier	True	multiple issues	2024-10-16 05:09:36.507464
127	CUST0014	PET00029	Emergency surgery (ACL)	3935	Sanchez-Mcdonald	Cost outlier	True	routine check	2025-06-21 05:09:36.507464
128	CUST0425	PET00840	Surgery (Tumor Removal)	2883	Davis Ltd	Cost outlier	True	urgent	2024-09-14 05:09:36.507464
130	CUST0214	PET00428	Emergency (bloat)	4110	Davis, Taylor and Jones	Cost outlier	True	multiple issues	2025-04-20 05:09:36.507464
					Rodriguez, Wolfe	High frequency			2024-12-24

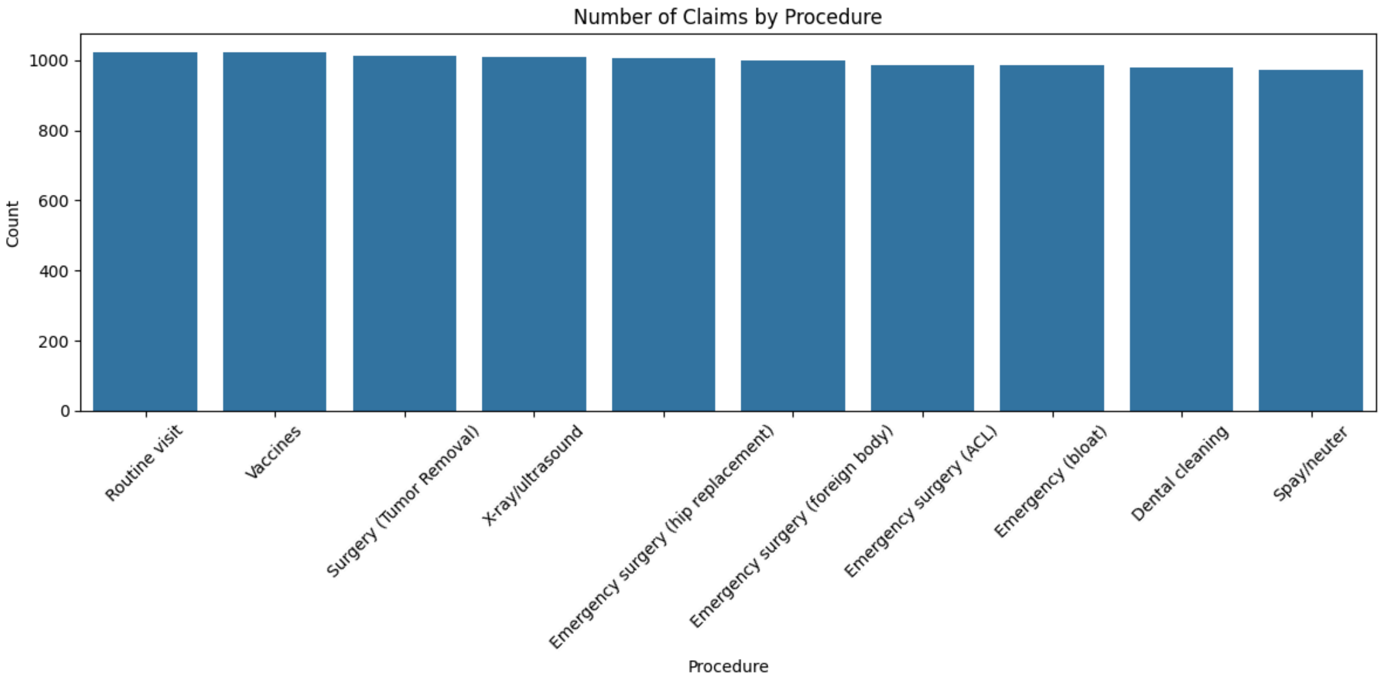
```
import matplotlib.pyplot as plt
import seaborn as sns

df['claim_month'] = df['claim_date'].dt.to_period('M').dt.to_timestamp()

monthly_counts = df.groupby('claim_month').size().reset_index(name='num_claims')

plt.figure(figsize=(12,5))
sns.lineplot(data=monthly_counts, x='claim_month', y='num_claims')
plt.title("Claims Volume Over Time (Monthly)")
plt.xlabel("Month")
plt.ylabel("Number of Claims")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



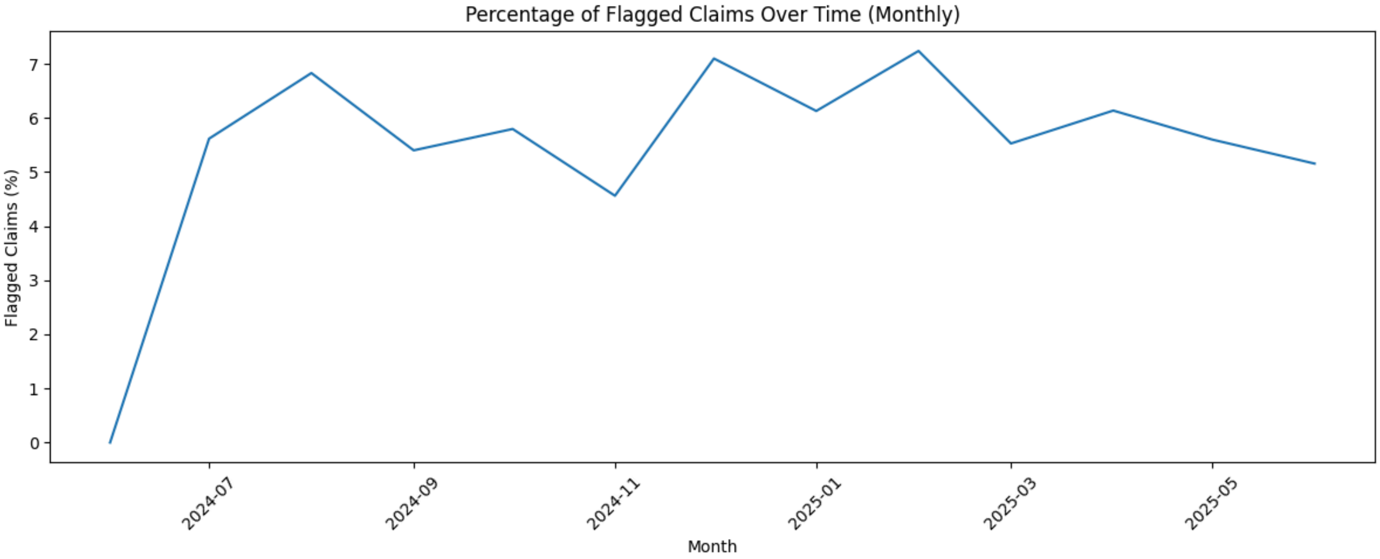
```
plt.figure(figsize=(10,5))
sns.histplot(df['cost'], bins=50, kde=True)
plt.title("Distribution of Claim Costs")
plt.xlabel("Cost")
plt.ylabel("Frequency")
plt.show()
```



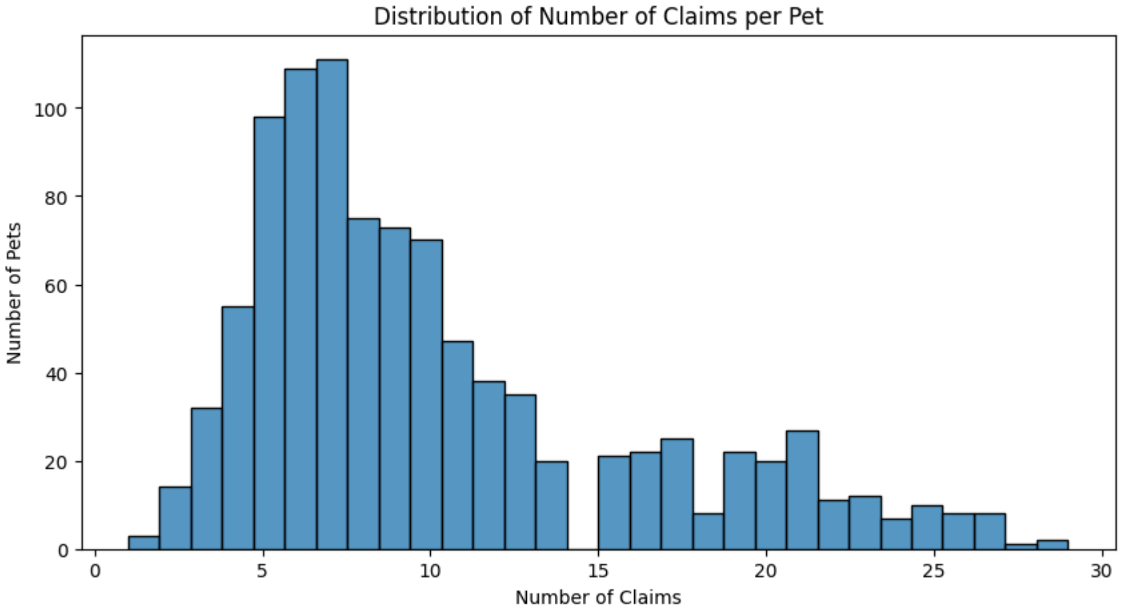
```
proc_counts = df['procedure'].value_counts()

plt.figure(figsize=(12,6))
sns.barplot(x=proc_counts.index, y=proc_counts.values)
plt.title("Number of Claims by Procedure")
plt.xlabel("Procedure")
plt.ylabel("Count")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



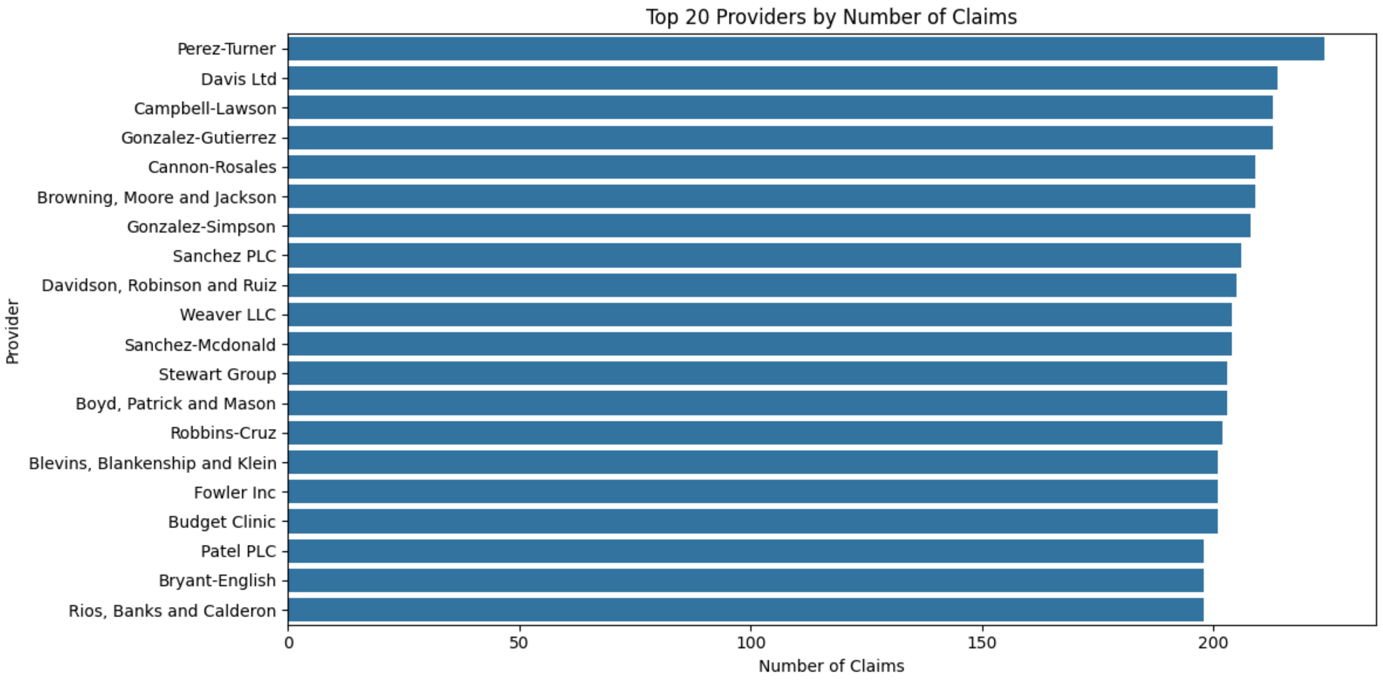
```
anomaly_over_time = df[df['status'] == 'flagged_for_review'].groupby('claim_month').size().reset_index(name='flagged_count')
total_over_time = df.groupby('claim_month').size().reset_index(name='total_count')

merged = anomaly_over_time.merge(total_over_time, on='claim_month', how='right').fillna(0)
merged['flagged_pct'] = merged['flagged_count'] / merged['total_count'] * 100

plt.figure(figsize=(12,5))
sns.lineplot(data=merged, x='claim_month', y='flagged_pct')
plt.title("Percentage of Flagged Claims Over Time (Monthly)")
plt.xlabel("Month")
plt.ylabel("Flagged Claims (%)")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



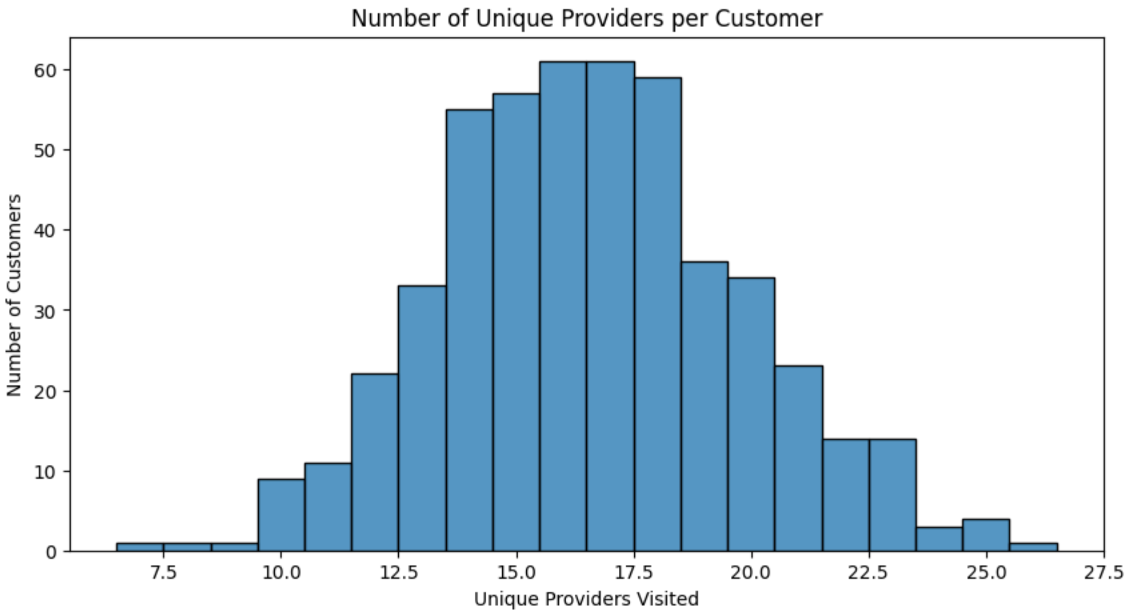
```
claims_per_pet = df.groupby('pet_id').size()

plt.figure(figsize=(10,5))
sns.histplot(claims_per_pet, bins=30, kde=False)
plt.title("Distribution of Number of Claims per Pet")
plt.xlabel("Number of Claims")
plt.ylabel("Number of Pets")
plt.show()
```



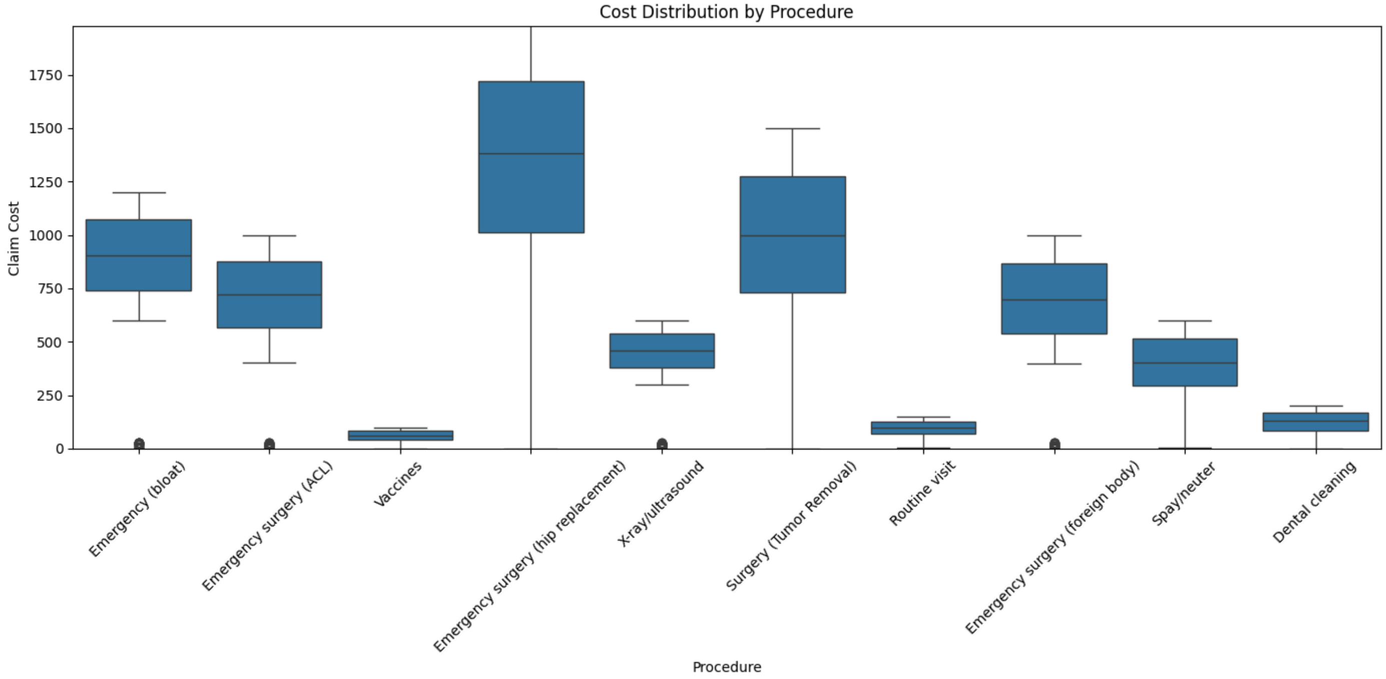
```
provider_counts = df['provider'].value_counts().head(20)

plt.figure(figsize=(12,6))
sns.barplot(x=provider_counts.values, y=provider_counts.index)
plt.title("Top 20 Providers by Number of Claims")
plt.xlabel("Number of Claims")
plt.ylabel("Provider")
plt.tight_layout()
plt.show()
```



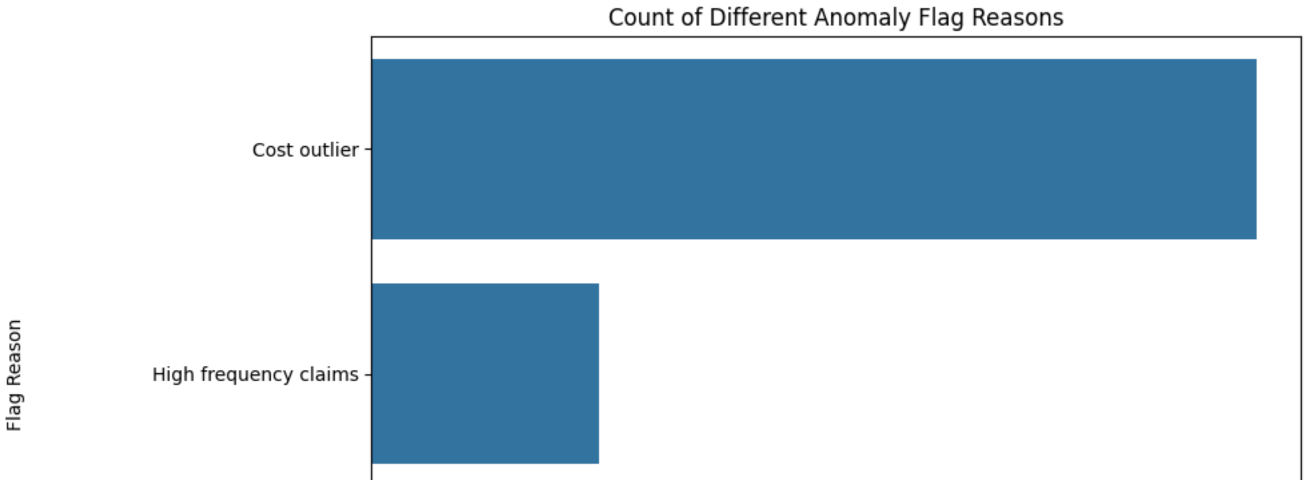
```
providers_per_customer = df.groupby('customer_id')['provider'].nunique()

plt.figure(figsize=(10,5))
sns.histplot(providers_per_customer, bins=range(1, providers_per_customer.max()+2), discrete=True)
plt.title("Number of Unique Providers per Customer")
plt.xlabel("Unique Providers Visited")
plt.ylabel("Number of Customers")
plt.show()
```



```
plt.figure(figsize=(14,7))
sns.boxplot(x='procedure', y='cost', data=df)
plt.title("Cost Distribution by Procedure")
plt.xlabel("Procedure")
plt.ylabel("Claim Cost")
plt.xticks(rotation=45)
plt.ylim(0, df['cost'].quantile(0.95))  # Focus on 95% quantile to limit outlier stretch
plt.tight_layout()
plt.show()
```



```
flag_reason_counts = df['flag_reason'].value_counts()

plt.figure(figsize=(10,6))
sns.barplot(x=flag_reason_counts.values, y=flag_reason_counts.index)
plt.title("Count of Different Anomaly Flag Reasons")
plt.xlabel("Count")
plt.ylabel("Flag Reason")
plt.tight_layout()
plt.show()
```


Count of Different Anomaly Flag Reasons
Cost outlier
High frequency claims
Flag Reason