In [2]:

```
"""AnomalyDetection-FlowScope Model with Evaluation Metrics"""

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.metrics import precision_score, recall_score, accuracy_score

import warnings
warnings.filterwarnings("ignore")


#Load and prepare data
df = pd.read_csv("~/Downloads/global_fund_transactions_dataset.csv")
df['transaction_date'] = pd.to_datetime(df['transaction_date'])
df['week'] = df['transaction_date'].dt.to_period('W').apply(lambda r: r.start_ti

#Aggregate transaction behavior per entity per week
#Feature engineering input are applied manually to identify the target
weekly_behavior = df.groupby(['company_number', 'week']).agg(
    transaction_count=('transaction_id', 'count'),
    total_amount=('amount_usd', 'sum'),
    avg_amount=('amount_usd', 'mean'),
    unique_transaction_types=('transaction_type', 'nunique')
).reset_index()
```

```python
#Behavioral Profiling (rolling stats & peer comparison)
#rolling z-score preferred over global z-score to understand the consequence of
def compute_rolling_features(df, features, window):
    for col in features:
        df = df.sort_values(['company_number', 'week'])
        df[f'{col}_mean'] = df.groupby('company_number')[col].transform(lambda x
        df[f'{col}_std'] = df.groupby('company_number')[col].transform(lambda x:
        df[f'{col}_z'] = (df[col] - df[f'{col}_mean']) / df[f'{col}_std']
    return df

rolling_cols = ['transaction_count', 'total_amount', 'avg_amount', 'unique_trans
window_size = 4 #weeks considered
weekly_behavior = compute_rolling_features(weekly_behavior, rolling_cols, window

#Peer-group scoring
for col in rolling_cols:
    weekly_behavior[f'{col}_peer_z'] = weekly_behavior.groupby('week')[col].tran

#Composite anomaly score
zscore_columns = [f'{col}_z' for col in rolling_cols] + [f'{col}_peer_z' for col
weekly_behavior['flowscope_score'] = weekly_behavior[zscore_columns].abs().mean(
#i.e., Fewer anomalies were detected for each company

#Flag anomalies.
threshold = 0.5 #Improved RECALL metric.
weekly_behavior['is_anomaly'] = weekly_behavior['flowscope_score'] > threshold
```

```python
#Merge with company metadata
company_meta = df[['company_number', 'company_name', 'jurisdiction']].drop_dupli
results = weekly_behavior.merge(company_meta, on='company_number', how='left')

#Split legal and illegal (anomalous) companies
company_anomaly_flags = results.groupby('company_number')['is_anomaly'].apply(la
legal_companies_set = set(company_anomaly_flags[company_anomaly_flags].index)
anomalous_companies_set = set(company_anomaly_flags[~company_anomaly_flags].inde

n_illegal = min(10, len(anomalous_companies_set))
n_legal = min(10, len(legal_companies_set))

sample_illegal_ids = pd.Series(list(anomalous_companies_set)).sample(n=n_illegal
sample_legal_ids = pd.Series(list(legal_companies_set)).sample(n=n_legal, random

sample_illegal = results[results['company_number'].isin(sample_illegal_ids)]
sample_legal = results[results['company_number'].isin(sample_legal_ids)]

#print("\n Legal Companies Transactions Sample:")
#print(sample_legal.groupby('company_number').first()[['company_name', 'week', '

#print("\n Anomalous Companies Transactions Sample:")
#print(sample_illegal.groupby('company_number').first()[['company_name', 'week',

#Risk Scoring Model per Company (with Confidence Score)
risk_scores = results.groupby(['company_number', 'company_name', 'jurisdiction']
```

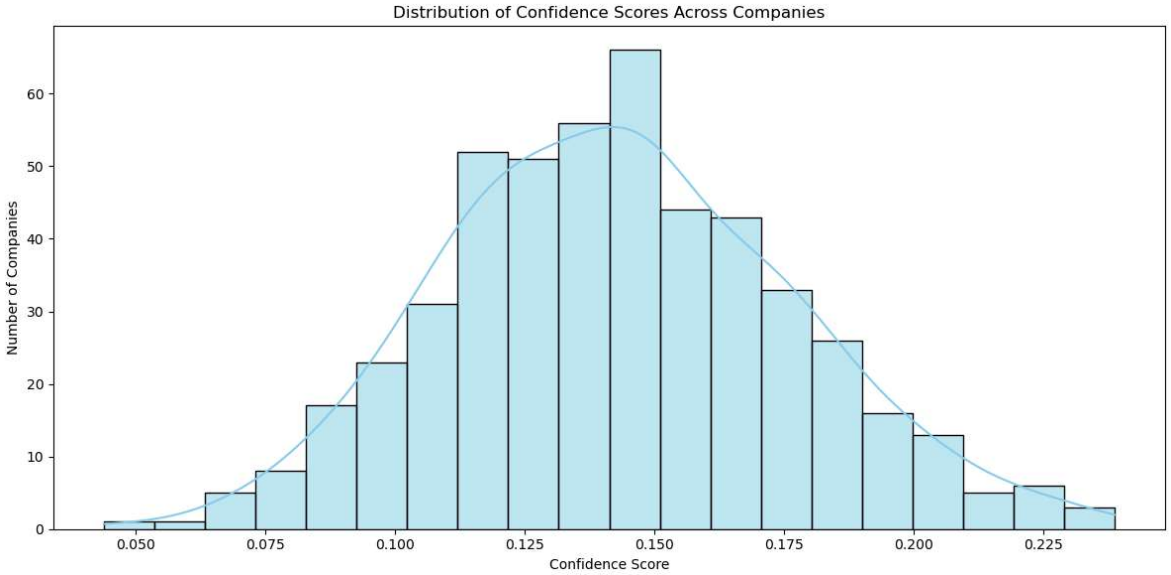
```
    avg_flowscope_score=('flowscope_score', 'mean'),
    max_flowscope_score=('flowscope_score', 'max'),
    anomaly_weeks=('is_anomaly', 'sum'),
    total_weeks=('week', 'count')
).reset_index()

max_score_global = risk_scores['max_flowscope_score'].max()
risk_scores['normalized_flowscope'] = risk_scores['max_flowscope_score'] / (max_
anomaly_severity = results[results['is_anomaly']].groupby('company_number')['flo
risk_scores['anomaly_severity'] = risk_scores['company_number'].map(anomaly_seve

risk_scores['risk_score'] = (
    0.2 * risk_scores['normalized_flowscope'] +  # Measure of spike intensity
    0.4 * (risk_scores['anomaly_weeks'] / risk_scores['total_weeks']) + #frequen
    0.4 * (risk_scores['anomaly_severity'] / (risk_scores['anomaly_severity'].ma
)

risk_scores['confidence_score'] = 1 - (risk_scores['anomaly_weeks'] / risk_score
risk_scores['risk_category'] = pd.cut(
    risk_scores['risk_score'],
    bins=[-np.inf, 0.33, 0.66, np.inf], #Random values chosen for the classifica
    labels=['Low', 'Medium', 'High']
)

risk_scores = risk_scores.sort_values(by='risk_score', ascending=False)

print("Computed Risk scores of Companies:")
```

```
print(risk_scores[['company_name', 'jurisdiction', 'risk_score', 'confidence_sco

#(optional) Visualize Confidence Scores
plt.figure(figsize=(12, 6))
sns.histplot(risk_scores['confidence_score'], bins=20, kde=True, color='skyblue'
plt.title('Distribution of Confidence Scores Across Companies')
plt.xlabel('Confidence Score')
plt.ylabel('Number of Companies')
plt.tight_layout()
plt.show()

#Export Risk Scores - Rather printing a long list on console
risk_scores.to_csv("companies_risk_scores_report.csv", index=False)
print("Risk scores report saved as 'companies_risk_scores_report.csv'")

#(Optional)Visualize Anomalies for Sampled Companies
def plot_multiple_companies(unique_company_ids, title):
    for company_id in unique_company_ids:
        subset = results[results['company_number'] == company_id].sort_values('w
        plt.figure(figsize=(10, 4))
        plt.plot(subset['week'], subset['total_amount'], label='Total Amount')
        plt.scatter(subset[subset['is_anomaly']]['week'],
                    subset[subset['is_anomaly']]['total_amount'],
                    color='red', label='Anomaly')
        plt.title(f"{title} - {subset['company_name'].iloc[0]} ({company_id})")
        plt.xlabel("Week")
        plt.ylabel("Total Amount (USD)")
```

```
        plt.xticks(rotation=45)
        plt.legend()
        plt.tight_layout()
        plt.show()

#plot_multiple_companies(sample_legal_ids, "Legal Company")
#plot_multiple_companies(sample_illegal_ids, "Anomalous Company")

#Evaluation Metrics - Computing metrics PRECISION and RECALL
# Convert ground truth to integer (1 = anomaly, 0 = normal) from the synthetic d
df['label'] = df['is_sanctioned'].astype(int)

# Merge labels into weekly_behavior using company_number and week
label_df = df[['company_number', 'week', 'label']].drop_duplicates()
eval_df = weekly_behavior.merge(label_df, on=['company_number', 'week'], how='le

# Drop any rows without labels
eval_df = eval_df.dropna(subset=['label'])

# Ground truth and predicted values
y_true = eval_df['label']
y_pred = eval_df['is_anomaly'].astype(int)

# Compute metrics
precision = precision_score(y_true, y_pred)
recall = recall_score(y_true, y_pred)
accuracy = accuracy_score(y_true, y_pred)
```

```
print("\n Evaluation Metrics: \n")
print(f"Precision: {precision:.2f}")
print(f"Recall:    {recall:.2f}")
print(f"Accuracy:  {accuracy:.2f}")
```

```
Computed Risk scores of Companies:
                   company_name    jurisdiction  risk_score  confidence_score  \
198                  Wong Group         Somalia    0.939535          0.151163
103                  Buck-Blake   United States    0.918317          0.202247
86   Lawrence, Jones and Cooper         Liberia    0.896847          0.139535
135     Solis, Craig and Parker  United Kingdom    0.864628          0.116279
317    Sparks, Morgan and Wyatt     North Korea    0.864164          0.168539

    risk_category
198          High
103          High
86           High
135          High
317          High
```




```
Risk scores report saved as 'companies_risk_scores_report.csv'

 Evaluation Metrics:

Precision: 0.40
Recall:    0.86
Accuracy:  0.43
```

In [ ]: