PCA & iForest

This notebook applies two unsupervised methods to the feature-engineered credit card transactions: PCA & Isolation Forest.

- PCA (Principal Component Analysis) projects data into directions of maximum variance and flags points with large reconstruction error or extreme component scores as outliers.
- Isolation Forest isolates anomalies quickly via random splits; rare, "easy-to-separate" points receive higher anomaly scores.

Dataset: feature-engineered credit card transactions
Model setup: numeric features standardized; contamination set to 5% for both methods.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from pyod.models.pca import PCA as PyODPCA
from pyod.models.iforest import IForest
from IPython.display import display

pd.set_option("display.max_columns", None)
plt.rcParams["figure.figsize"] = (8, 5)
```

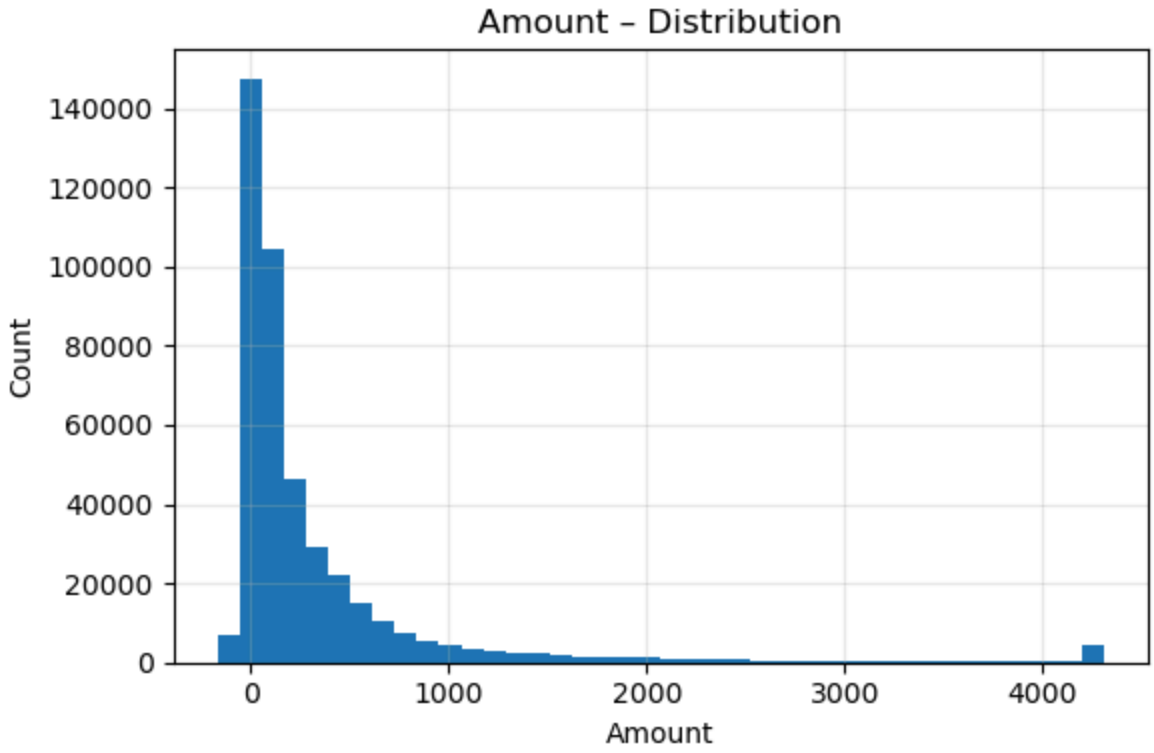
1. Prepare the Data

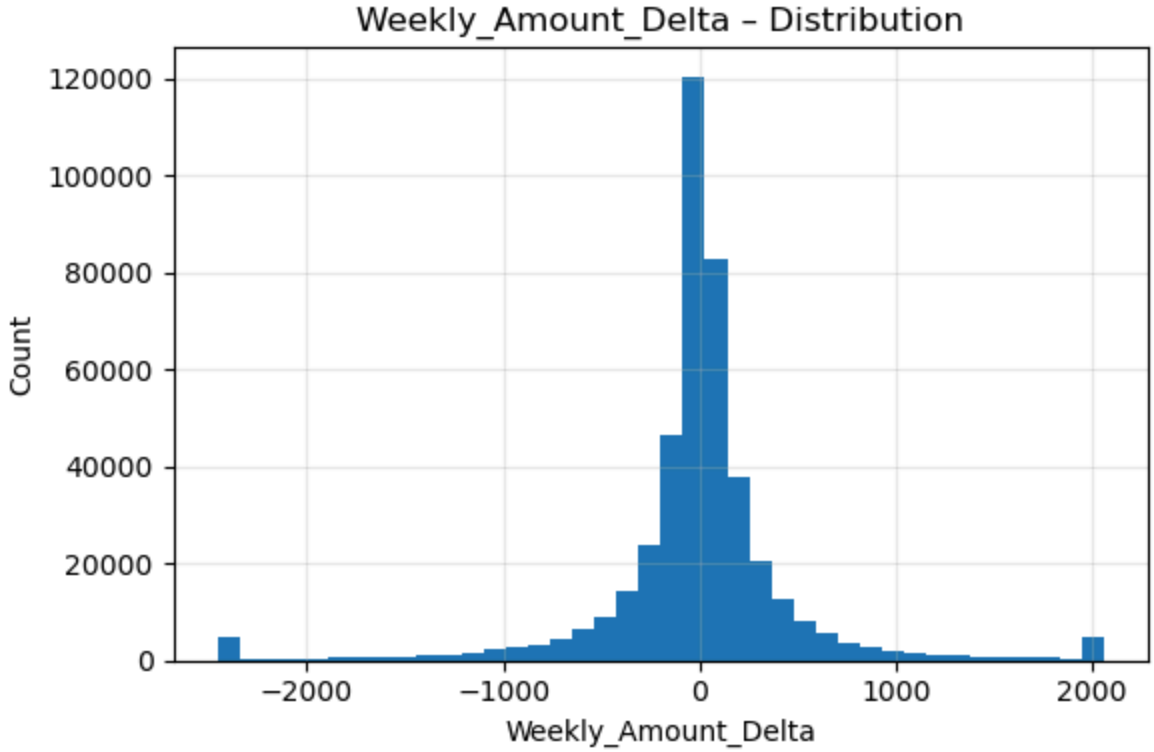
In [2]:
```
PATH = "/users/tiffanytruong/Documents/APAN5420/df_model.csv"

df = pd.read_csv(PATH)

# Keep untouched copy for joining model outputs later
df_full = df.copy()

# Drop Descriptive/text/ID columns before unsupervised modeling
drop_text_cols = [
    'Unnamed: 0', 'Year-Month', 'Agency Number', 'Agency Name',
    'Cardholder Last Name', 'Cardholder First Initial', 'cardholder_full_nam
    'Description', 'Vendor', 'Transaction Date', 'Posted Date',
    'Merchant Category Code (MCC)', 'Weekend', 'Day of Week',
    'Quarter', 'Week_Number', 'Year'
]

df_model = df.drop(columns=[c for c in drop_text_cols if c in df.columns], e

# Convert any booleans to integers
for c in df_model.columns:
    if df_model[c].dtype == bool:
```

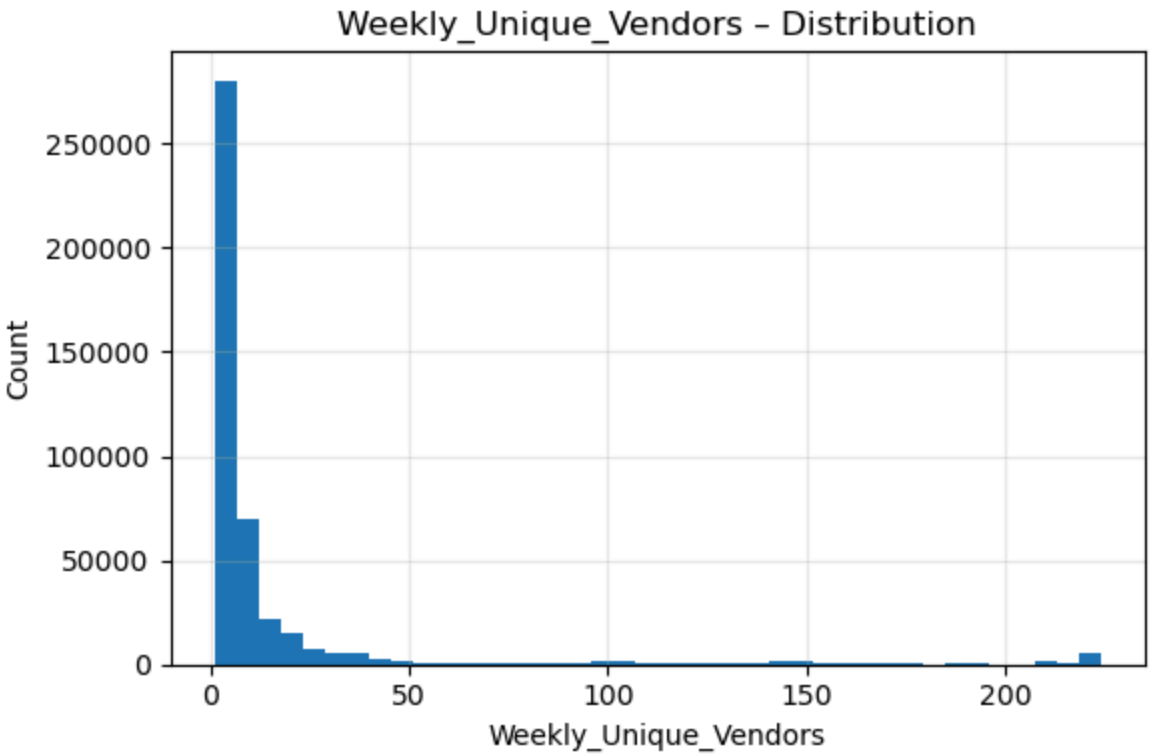
```
        df_model[c] = df_model[c].astype(int)

# Keep only numeric cols since PyOD expects numeric features
num_cols = df_model.select_dtypes(include=[np.number]).columns.tolist()
X = df_model[num_cols].copy()

# Handle inf/nan
X = X.replace([np.inf, -np.inf], np.nan)
X = X.fillna(0)

interpretation_cols = [
    'Amount', 'Amount Weekly Mean', 'Transaction Count Weekly',
    'Amount_Above_90th', 'Amount_Above_95th',
    'Frequency_Above_90th', 'Frequency_Above_95th',
    'Transaction_to_Merchant_Mean_Ratio',
    'Weekly_Unique_Vendors', 'Vendor_Diversity_Ratio',
    'Weekly_Amount_Delta', 'Weekly_Frequency_Delta',
    'Weekly_Weekend_Spend_Proportion', 'Is_Round_Amount'
]
interpretation_cols = [c for c in interpretation_cols if c in X.columns]

# Standardize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

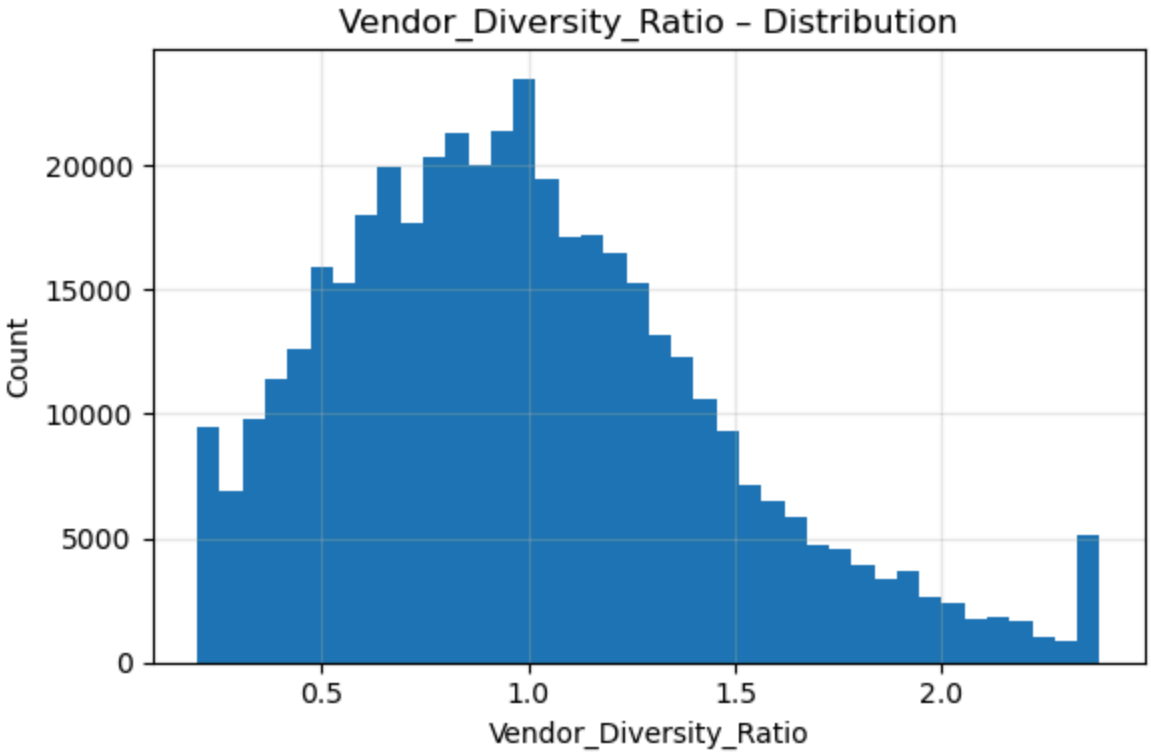
In [3]:

```
# Visualize a few features
features = [
    "Amount",
    "Weekly_Amount_Delta",
    "Weekly_Unique_Vendors",
    "Vendor_Diversity_Ratio",
]

# clean for plotting
plot_df = df[features].replace([np.inf, -np.inf], np.nan).dropna()

# Histograms
for col in features:
    plt.figure(figsize=(6,4))
    # Use bins robust to outliers
    series = plot_df[col].clip(lower=plot_df[col].quantile(0.01),
                               upper=plot_df[col].quantile(0.99))
    plt.hist(series, bins=40)
    plt.title(f"{col} — Distribution")
    plt.xlabel(col)
    plt.ylabel("Count")
    plt.grid(True, alpha=0.3)
    plt.tight_layout()
```


Amount – Distribution
Count
140000
120000
100000
80000
60000
40000
20000
0
0
1000
2000
3000
4000
Amount


Weekly_Amount_Delta – Distribution
Count
120000
100000
80000
60000
40000
20000
0
−2000
−1000
0
1000
2000
Weekly_Amount_Delta





2. PCA (PyOD)

Principal Component Analysis (PCA) projects high-dimensional data onto a lower-dimensional space that captures the directions of maximum variance. In PyOD's PCA for anomaly detection, points with large reconstruction errors or extreme projection distances are flagged as outliers.

In [4]:
```
# Fit PCA (PyOD)
pca = PyODPCA(
    contamination=0.05,
    n_components=None,
    random_state=42,
    weighted=True
)
pca.fit(X_scaled)

# Scores & labels
pca_scores = pca.decision_function(X_scaled)
pca_labels = pca.predict(X_scaled)
pca_threshold = pca.threshold_

# Attach to full df for summaries
df_full['pca_score'] = pca_scores
df_full['pca_flag'] = pca_labels
```

3. Isolation Forest (PyOD)

Isolation Forest isolates anomalies by randomly partitioning the data; outliers are easier to separate and hence require fewer splits. The model builds many random trees and assigns an anomaly score based on how quickly a point becomes isolated across trees.

In [5]:
```
iforest = IForest(
    contamination=0.05,
    n_estimators=300,
    max_samples='auto',
    random_state=42,
    behaviour='new' if 'behaviour' in IForest.__init__.__code__.co_varnames
)
iforest.fit(X_scaled)

if_scores = iforest.decision_function(X_scaled)
if_labels = iforest.predict(X_scaled)
if_threshold = iforest.threshold_

df_full['iforest_score'] = if_scores
df_full['iforest_flag'] = if_labels
```

4. Summary Statistics & Anomalous Case Sizes

In [6]:
```
def summarize_flags(df_in, flag_col, score_col, method_name):
    out = {}
    n = len(df_in)
    n_out = int(df_in[flag_col].sum())
    pct_out = 100 * n_out / n if n > 0 else 0

    out['method'] = method_name
    out['n'] = n
    out['n_anomalies'] = n_out
    out['pct_anomalies'] = round(pct_out, 2)
    out['threshold'] = df_in[score_col].quantile(0.95) if 'pca' in method_na
```

```
    # Group means on interpretable features
    cols_to_summarize = interpretation_cols.copy()
    if score_col not in cols_to_summarize:
        cols_to_summarize.append(score_col)

    available = [c for c in cols_to_summarize if c in df_in.columns]
    group_means = (
        df_in.assign(Group=np.where(df_in[flag_col] == 1, 'Outlier', 'Normal
            .groupby('Group')[available]
            .mean()
            .round(2)
    )

    out['group_means'] = group_means
    return out

pca_summary = summarize_flags(df_full, 'pca_flag', 'pca_score', 'PCA')
if_summary  = summarize_flags(df_full, 'iforest_flag', 'iforest_score', 'Isc

print("Size of Anomalous Cases:")
print(f"PCA -> anomalies: {pca_summary['n_anomalies']} of {pca_summary['n']}
      f"({pca_summary['pct_anomalies']}%)")
print(f"Isolation Forest -> anomalies: {if_summary['n_anomalies']} of {if_su
      f"({if_summary['pct_anomalies']}%)\n")

print("PCA Group Means (selected features):")
display(pca_summary['group_means'])
print("Isolation Forest Group Means (selected features):")
display(if_summary['group_means'])
```

```
Size of Anomalous Cases:
PCA -> anomalies: 22123 of 442458 (5.0%)
Isolation Forest -> anomalies: 22123 of 442458 (5.0%)

PCA Group Means (selected features):
```

Group	Amount	Amount_Above_95th	Transaction_to_Merchant_Mean_Ratio	Weekly_U
Normal	339.72	0.04	-1.570690e+11	
Outlier	2045.06	0.28	2.984296e+12	

```
Isolation Forest Group Means (selected features):
```

Group	Amount	Amount_Above_95th	Transaction_to_Merchant_Mean_Ratio	Weekly_U
Normal	288.37	0.04	-7.514289e+10	
Outlier	3020.69	0.37	1.427708e+12	

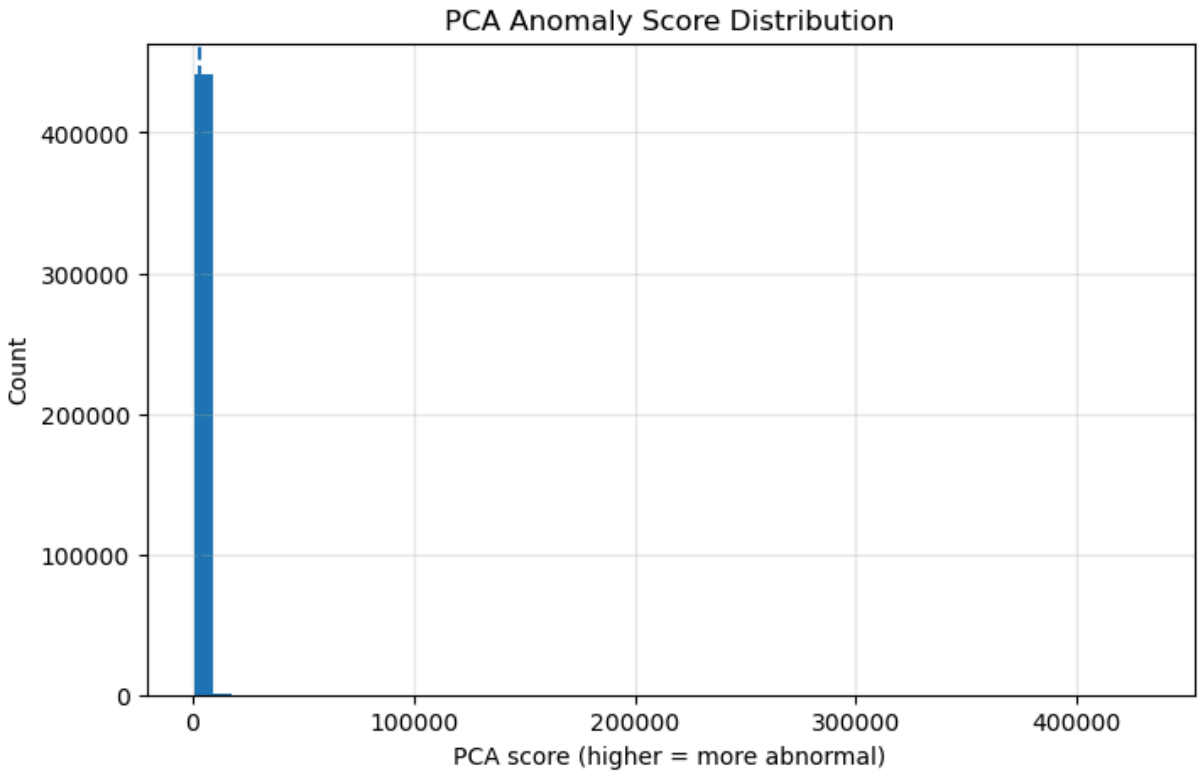
5. Overlap Analysis (Potential Fraudulent Transactions)

In [7]:
```
df_full['both_flag'] = ((df_full['pca_flag'] == 1) & (df_full['iforest_flag'
n_both = int(df_full['both_flag'].sum())
pct_of_pca = round(100 * n_both / max(1, int(df_full['pca_flag'].sum())), 2)
pct_of_if  = round(100 * n_both / max(1, int(df_full['iforest_flag'].sum()))

print("\n=== Overlap (flagged by BOTH PCA & Isolation Forest) ===")
print(f"Count: {n_both}")
print(f"As % of PCA anomalies: {pct_of_pca}%")
print(f"As % of iForest anomalies: {pct_of_if}%")
```

```
=== Overlap (flagged by BOTH PCA & Isolation Forest) ===
Count: 13974
As % of PCA anomalies: 63.17%
As % of iForest anomalies: 63.17%
```

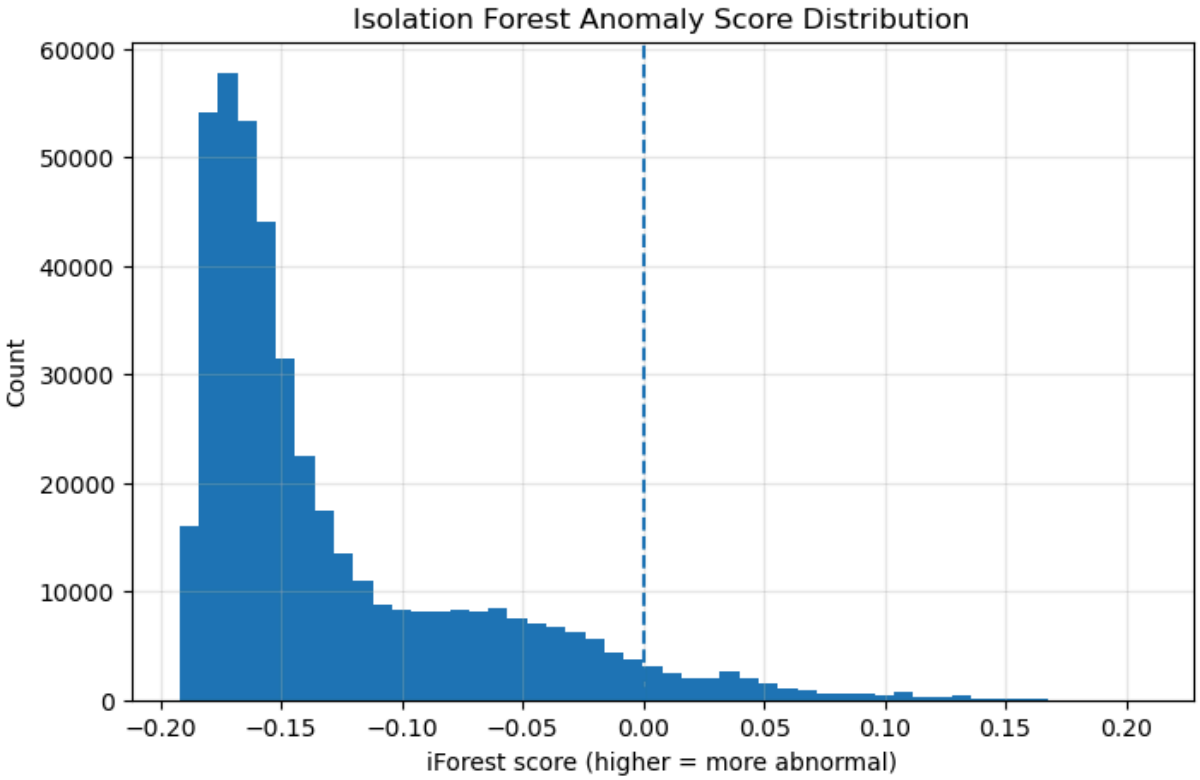
6. Diagnostics Plots (score distributions)

In [8]:
```
# PCA score histogram
plt.figure()
plt.hist(df_full['pca_score'], bins=50)
plt.axvline(pca_threshold, linestyle='--')
plt.title("PCA Anomaly Score Distribution")
plt.xlabel("PCA score (higher = more abnormal)")
plt.ylabel("Count")
plt.grid(True, alpha=0.3)
plt.show()
```

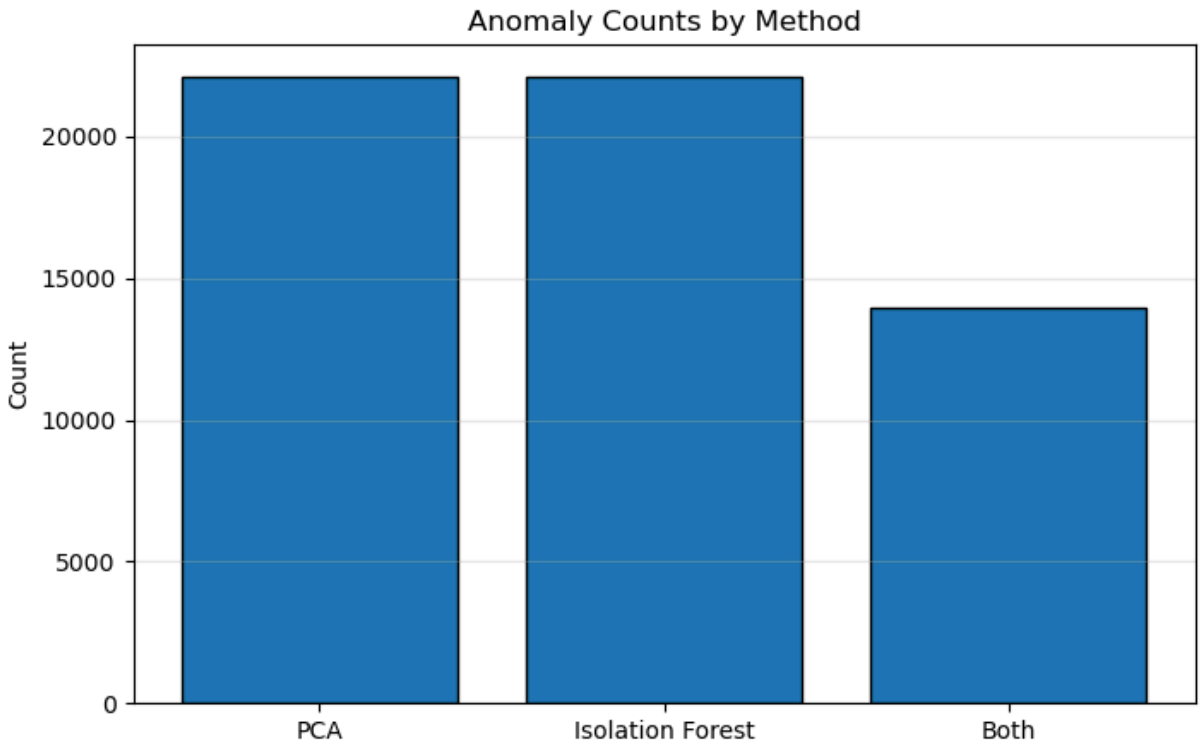


In [9]:
```
# Isolation Forest score histogram
plt.figure()
```

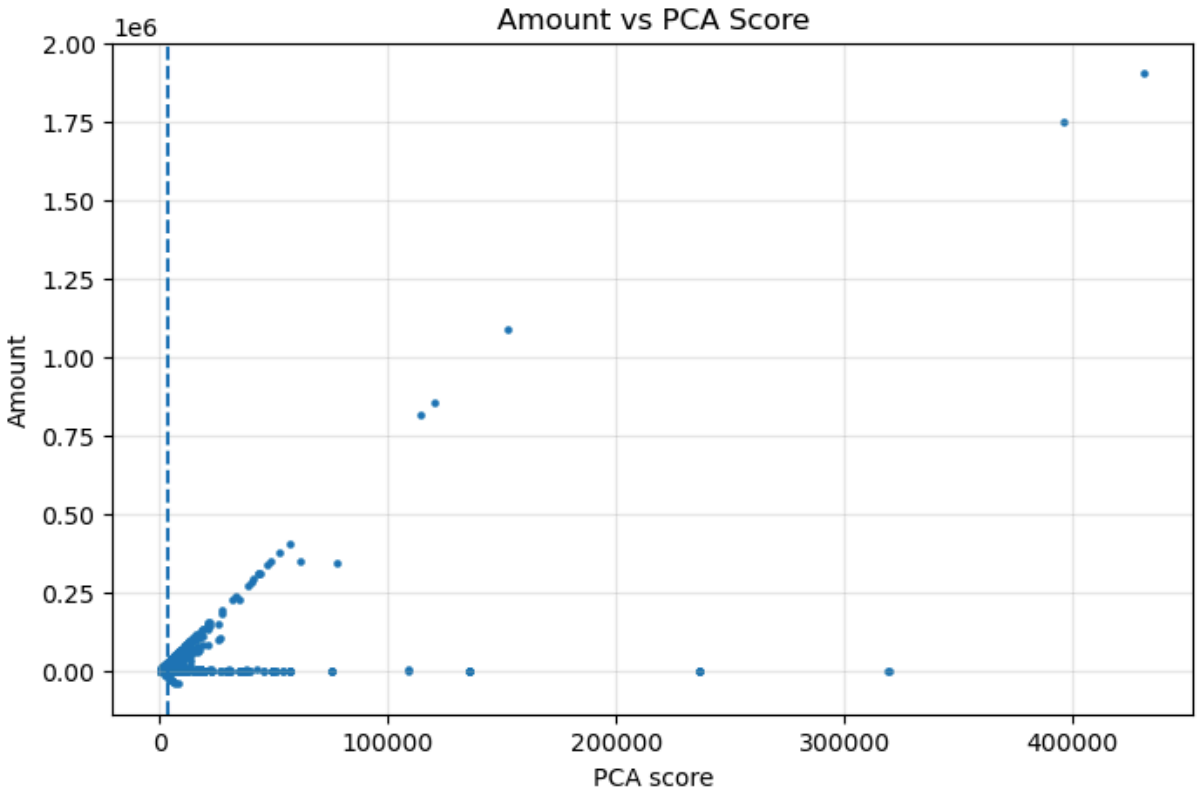
```
plt.hist(df_full['iforest_score'], bins=50)
plt.axvline(if_threshold, linestyle='--')
plt.title("Isolation Forest Anomaly Score Distribution")
plt.xlabel("iForest score (higher = more abnormal)")
plt.ylabel("Count")
plt.grid(True, alpha=0.3)
plt.show()
```



In [10]:
```
# Bar chart of anomaly counts
labels = ['PCA', 'Isolation Forest', 'Both']
values = [
    int(df_full['pca_flag'].sum()),
    int(df_full['iforest_flag'].sum()),
    n_both
]
plt.figure()
plt.bar(labels, values, edgecolor='black')
plt.title("Anomaly Counts by Method")
plt.ylabel("Count")
plt.grid(True, axis='y', alpha=0.3)
plt.show()
```



In [11]:
```
# Amount vs PCA Score
plt.figure()
plt.scatter(df_full['pca_score'], df_full['Amount'], s=5)
plt.axvline(pca_threshold, linestyle='--')
plt.title("Amount vs PCA Score")
plt.xlabel("PCA score")
plt.ylabel("Amount")
plt.grid(True, alpha=0.3)
plt.show()
```



7. Business Insight

This section summarizes the main findings from both PCA and Isolation Forest models. Each method reveals a different dimension of abnormal spending behavior, allowing for a more complete picture of potential fraud or policy violations in the dataset.

What I found:
PCA flagged 22,123 transactions (5.00%), while Isolation Forest also flagged 22,123 transactions (5.00%). There is an overlap of 13,974 records that were flagged by both methods (these overlapping transactions represent the strongest signals and should be prioritized for review).

PCA results interpretation:
PCA outliers tend to have unusually large reconstruction errors, meaning they deviate significantly from the normal pattern across multiple spending attributes at once. These transactions often combine a high Amount with an atypical cadence or vendor mix.

- In this dataset, PCA outliers showed a higher mean Amount (2,045.06) compared to normal transactions (339.72), reinforcing the unusual-spend narrative.

Isolation Forest interpretation:
Isolation Forest identifies anomalies by isolating rare combinations of features through random partitions. Transactions that are easy to isolate—those that stand out strongly from the rest—receive higher anomaly scores.

- In this dataset, Isolation Forest outliers showed a higher mean Amount (3,020.69) compared to normals (288.37), which suggests these anomalies are driven by extreme spending spikes, elevated weekly frequency, or large fluctuations in transaction behavior.

What looks suspicious (potential fraud or policy violations):

- High-Amount spikes and large positive Weekly_Amount_Delta, indicating one-off big purchases or possible split transactions.
- Elevated Transaction Count Weekly combined with greater Weekly_Unique_Vendors or Vendor_Diversity_Ratio, suggesting vendor hopping or unusual spending dispersion.
- Transactions flagged by both PCA and Isolation Forest, as they represent structural deviations (from PCA) and statistical rarity (from Isolation Forest).

Business actionability:

- Prioritize manual review for transactions flagged by both methods.
- Establish monitoring rules for Amount spikes, vendor diversity changes, and round-amount patterns that may indicate unusual spending habits.
- Incorporate analyst feedback into future model iterations by refining input features (e.g., merchant risk tiers, time-of-day/week trends) and re-evaluating the contamination rate to improve detection accuracy.