```
# Crypto_Solana_Meme_Coins_Analysis.ipynb

# ------------------------------
# 1️⃣ Import Libraries
# ------------------------------
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats

sns.set(style="whitegrid")

# ------------------------------
# 2️⃣ Load Dataset
# ------------------------------
df = pd.read_csv('crypto_solana_meme_coins.csv')
print("Initial dataset shape:", df.shape)
df.head()

# ------------------------------
# 3️⃣ Basic Cleaning
# ------------------------------

# Strip whitespace and normalize strings
df['Coin Name'] = df['Coin Name'].str.upper().str.strip()
```

```
df['Symbol'] = df['Symbol'].str.upper().str.strip()

# Convert Launch Date
df['Launch Date'] = pd.to_datetime(df['Launch Date'], errors='coerce')

# Convert numeric columns
numeric_cols = ['Current Price (USD)', 'Market Cap (USD)', 'Trading Volume (24h)',
        'Circulating Supply', 'Total Supply', 'Price Change (24h)',
        'All-Time High (Price)', 'All-Time Low (Price)',
        'Number of Holders', 'Transactions Count', 'Token Liquidity (USD)', 'Returns (%)']

for col in numeric_cols:
   df[col] = pd.to_numeric(df[col], errors='coerce')

# Drop rows with essential missing info
df = df.dropna(subset=['Coin Name', 'Symbol', 'Token Address', 'Current Price (USD)'])

# Fill missing numeric values with median for robust analysis
fill_median_cols = ['Market Cap (USD)', 'Trading Volume (24h)', 'Token Liquidity (USD)']
for col in fill_median_cols:
   df[col] = df[col].fillna(df[col].median())

# Remove duplicates based on Token Address
df = df.drop_duplicates(subset=['Token Address'])

# Remove negative or zero values where impossible
```

```
df = df[df['Current Price (USD)'] > 0]

df = df[df['Market Cap (USD)'] >= 0]

df = df[df['Circulating Supply'] >= 0]

df = df[df['Total Supply'] >= 0]

df = df[df['Number of Holders'] >= 0]

df = df[df['Transactions Count'] >= 0]


# Ensure logical consistency for price

df = df[df['All-Time High (Price)'] >= df['All-Time Low (Price)']]

df = df[df['Current Price (USD)'].between(df['All-Time Low (Price)'], df['All-Time High (Price)'])]


# Reset index

df.reset_index(drop=True, inplace=True)


# ------------------------------

# 4️⃣ Feature Engineering

# ------------------------------


# Market Cap per Holder

df['Market Cap per Holder'] = df['Market Cap (USD)'] / df['Number of Holders'].replace(0, np.nan)


# Liquidity ratio

df['Liquidity Ratio'] = df['Token Liquidity (USD)'] / df['Market Cap (USD)']
```

```
# Price range ratio (volatility)
df['Price Range Ratio'] = (df['All-Time High (Price)'] - df['All-Time Low (Price)']) / df['All-Time Low (Price)']

# Coin Age in days
df['Coin Age (days)'] = (pd.Timestamp.today() - df['Launch Date']).dt.days

# Active trading flag
df['Active'] = df['Trading Volume (24h)'] > 0

# ------------------------------
# 5️⃣ Outlier Detection & Removal
# ------------------------------

outlier_cols = ['Current Price (USD)', 'Market Cap (USD)', 'Trading Volume (24h)', 'Token Liquidity (USD)']
z_scores = np.abs(stats.zscore(df[outlier_cols].fillna(0)))
df = df[(z_scores < 3).all(axis=1)]

# ------------------------------
# 6️⃣ Basic Summary
# ------------------------------
print("Cleaned dataset shape:", df.shape)
print(df.info())
print(df.describe())
print(df['Active'].value_counts())
```

```
# ------------------------------
# 7️⃣ Basic Visualizations
# ------------------------------

# Distribution of Current Price (log scale)
plt.figure(figsize=(10,5))
sns.histplot(df['Current Price (USD)'], bins=50, log_scale=True)
plt.title('Distribution of Current Price (USD)')
plt.show()

# Market Cap vs Trading Volume scatter
plt.figure(figsize=(10,6))
sns.scatterplot(x='Market Cap (USD)', y='Trading Volume (24h)', data=df, hue='Active')
plt.xscale('log')
plt.yscale('log')
plt.title('Market Cap vs Trading Volume')
plt.show()

# Top 10 Coins by Market Cap
top_marketcap = df.sort_values('Market Cap (USD)', ascending=False).head(10)
plt.figure(figsize=(12,6))
sns.barplot(x='Coin Name', y='Market Cap (USD)', data=top_marketcap)
plt.xticks(rotation=45)
plt.title('Top 10 Meme Coins by Market Cap')
plt.show()
```

```
# ------------------------------
# 8️⃣ Deeper Analysis
# ------------------------------

# Most Volatile Coins
top_volatility = df.sort_values('Price Range Ratio', ascending=False).head(10)
plt.figure(figsize=(12,6))
sns.barplot(x='Coin Name', y='Price Range Ratio', data=top_volatility)
plt.xticks(rotation=45)
plt.title('Top 10 Most Volatile Meme Coins')
plt.show()

# Coins with Highest Liquidity Ratio
top_liquidity = df.sort_values('Liquidity Ratio', ascending=False).head(10)
plt.figure(figsize=(12,6))
sns.barplot(x='Coin Name', y='Liquidity Ratio', data=top_liquidity)
plt.xticks(rotation=45)
plt.title('Top 10 Coins by Liquidity Ratio')
plt.show()

# Top Coins by Market Cap per Holder
top_popularity = df.sort_values('Market Cap per Holder', ascending=False).head(10)
plt.figure(figsize=(12,6))
sns.barplot(x='Coin Name', y='Market Cap per Holder', data=top_popularity)
plt.xticks(rotation=45)
```

```
plt.title('Top 10 Meme Coins by Market Cap per Holder')
plt.show()

# Returns Analysis
plt.figure(figsize=(10,5))
sns.histplot(df['Returns (%)'], bins=50, kde=True)
plt.title('Distribution of Returns (%) Across Meme Coins')
plt.show()

top_returns = df.sort_values('Returns (%)', ascending=False).head(10)
plt.figure(figsize=(12,6))
sns.barplot(x='Coin Name', y='Returns (%)', data=top_returns)
plt.xticks(rotation=45)
plt.title('Top 10 Meme Coins by Returns (%)')
plt.show()

# ------------------------------
# 🥧 Pie Charts (Dynamic & Error-Proof)
# ------------------------------

# Active vs Inactive Coins
active_counts = df['Active'].value_counts()
labels = active_counts.index.map({True: 'Active', False: 'Inactive'})
plt.figure(figsize=(6,6))
plt.pie(active_counts, labels=labels, autopct='%1.1f%%', colors=['#ff69b4','#87cefa'])
plt.title('Active vs Inactive Meme Coins')
```

```
plt.show()

# Top 5 Market Cap Coins
top5_marketcap = df.sort_values('Market Cap (USD)', ascending=False).head(5)
plt.figure(figsize=(7,7))
plt.pie(top5_marketcap['Market Cap (USD)'],
    labels=top5_marketcap['Coin Name'],
    autopct='%1.1f%%',
    colors=sns.color_palette('pastel')[0:5])
plt.title('Top 5 Meme Coins by Market Cap')
plt.show()

# Top 5 Coins by Trading Volume
top5_volume = df.sort_values('Trading Volume (24h)', ascending=False).head(5)
plt.figure(figsize=(7,7))
plt.pie(top5_volume['Trading Volume (24h)'],
    labels=top5_volume['Coin Name'],
    autopct='%1.1f%%',
    colors=sns.color_palette('Set2')[0:5])
plt.title('Top 5 Meme Coins by 24h Trading Volume')
plt.show()

# Coin Age Distribution
bins = [0,30,90,180,365, np.inf]
labels_age = ['<1 Month','1-3 Months','3-6 Months','6-12 Months','>1 Year']
df['Age Group'] = pd.cut(df['Coin Age (days)'], bins=bins, labels=labels_age)
```

```
age_group_counts = df['Age Group'].value_counts().sort_index()

plt.figure(figsize=(7,7))

plt.pie(age_group_counts, labels=age_group_counts.index, autopct='%1.1f%%',
colors=sns.color_palette('bright', len(age_group_counts)))

plt.title('Distribution of Meme Coins by Age Group')

plt.show()


# ------------------------------

# 1 0️⃣ Save Cleaned Dataset

# ------------------------------

df.to_csv('cleaned_crypto_solana_meme_coins.csv', index=False)

print("Cleaned dataset saved as 'cleaned_coins.csv'")
```