Facebook marketplace

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

# Load dataset
file_path ='E:/Trimester 2/Data science/finlatics/MLResearch/Facebook Dataset/Facebook_Marketplace_data.csv'
dataset = pd.read_csv(file_path)

# Display basic information and the first few rows of the dataset
dataset_info = dataset.info()
dataset_head = dataset.head()

dataset_info, dataset_head

# Remove redundant columns
cleaned_dataset = dataset.drop(columns=['Column1', 'Column2', 'Column3', 'Column4'])

# Convert `status_published` to datetime format
cleaned_dataset['status_published'] = pd.to_datetime(cleaned_dataset['status_published'], errors='coerce')

# Check for missing or invalid values after conversion
missing_status_published = cleaned_dataset['status_published'].isnull().sum()
```

```
# Display the updated dataset information
cleaned_dataset_info = cleaned_dataset.info()
missing_status_published, cleaned_dataset_info
```

```
# Extract hour from `status_published`
cleaned_dataset['hour_published'] = cleaned_dataset['status_published'].dt.hour

# Group by hour and calculate average reactions
hourly_reactions =
cleaned_dataset.groupby('hour_published')['num_reactions'].mean()

# Plot the results
plt.figure(figsize=(10, 6))
sns.lineplot(data=hourly_reactions, marker='o')
plt.title('Average Number of Reactions by Hour of Publication', fontsize=14)
plt.xlabel('Hour of the Day (24-hour format)', fontsize=12)
plt.ylabel('Average Number of Reactions', fontsize=12)
plt.grid()
plt.show()

hourly_reactions
```

```
# Calculate correlation
correlation_matrix = dataset[['num_reactions', 'num_comments', 'num_shares']].corr()

# Plot heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
plt.title('Correlation between Reactions, Comments, and Shares')
plt.show()
```

```
# Count the different types of posts
post_type_counts = dataset['status_type'].value_counts()

# Plot the counts
plt.figure(figsize=(8, 6))
sns.barplot(x=post_type_counts.index, y=post_type_counts.values, palette='viridis')
plt.title('Count of Different Post Types')
plt.xlabel('Post Type')
plt.ylabel('Count')
plt.show()
```

```
# Group by `status_type` and calculate averages
averages_by_post_type = dataset.groupby('status_type')[['num_reactions',
'num_comments', 'num_shares']].mean()

# Display the averages
print(averages_by_post_type)

# Plot the averages
averages_by_post_type.plot(kind='bar', figsize=(10, 6))
plt.title('Average Engagement Metrics by Post Type')
plt.xlabel('Post Type')
plt.ylabel('Average Value')
plt.legend(loc='upper right')
plt.grid()
plt.show()
```

```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

# Select relevant columns
columns = ['num_reactions', 'num_comments', 'num_shares', 'num_likes', 'num_loves',
        'num_wows', 'num_hahas', 'num_sads', 'num_angrys']
data = dataset[columns]

# Standardize the data
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data)

# Use Elbow Method to determine the number of clusters
sse = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(data_scaled)
    sse.append(kmeans.inertia_)

# Plot the Elbow Method graph
plt.figure(figsize=(10, 6))
plt.plot(range(1, 11), sse, marker='o')
plt.title('Elbow Method for Optimal Clusters')
plt.xlabel('Number of Clusters')
plt.ylabel('SSE')
```

```
plt.grid()
plt.show()

# Train K-Means with the optimal number of clusters (e.g., 3 based on elbow plot)
kmeans = KMeans(n_clusters=3, random_state=42)
dataset['cluster'] = kmeans.fit_predict(data_scaled)

# Display cluster counts
print(dataset['cluster'].value_counts())
```

```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import pandas as pd

# Define columns for clustering
columns = ['status_type', 'num_reactions', 'num_comments', 'num_shares',
    'num_likes', 'num_loves', 'num_wows', 'num_hahas', 'num_sads', 'num_angrys']
```

```
# Extract relevant data
data = dataset[columns]

# Preprocessor for scaling and encoding
preprocessor = ColumnTransformer(
    transformers=[
        ('status_type', OneHotEncoder(), ['status_type']),
        ('numeric', StandardScaler(), ['num_reactions', 'num_comments', 'num_shares',
                                       'num_likes', 'num_loves', 'num_wows',
                                       'num_hahas', 'num_sads', 'num_angrys'])
    ]
)

# Preprocess data
data_preprocessed = preprocessor.fit_transform(data)

# Train K-Means with 10 clusters
optimal_clusters = 10
kmeans = KMeans(n_clusters=optimal_clusters, random_state=42)
cluster_labels = kmeans.fit_predict(data_preprocessed)

# Add cluster labels to the dataset
dataset['cluster'] = cluster_labels

# Perform PCA for dimensionality reduction (to 2 components)
pca = PCA(n_components=2)
reduced_data = pca.fit_transform(data_preprocessed)
```

```
# Reduce the cluster centers for visualization
cluster_centers_reduced = pca.transform(kmeans.cluster_centers_)


# Plot the clusters and their centers
plt.figure(figsize=(12, 8))
scatter = plt.scatter(reduced_data[:, 0], reduced_data[:, 1], c=cluster_labels,
cmap='viridis', s=50, alpha=0.7)
plt.scatter(cluster_centers_reduced[:, 0], cluster_centers_reduced[:, 1], c='red',
s=200, label='Cluster Centers', edgecolors='black')
plt.title('Cluster Visualization with 10 Clusters', fontsize=14)
plt.xlabel('PCA Component 1', fontsize=12)
plt.ylabel('PCA Component 2', fontsize=12)
plt.legend()
plt.colorbar(scatter, label='Cluster Label')
plt.grid()
plt.show()
```