```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler, MinMaxScaler

# Load the dataset
file_path = 'Novanectar.csv'  # Replace with your file path
data = pd.read_csv(file_path)


# Display missing values
print("Missing values before handling:")
print(data.isnull().sum())

# Fill missing values with mean for numerical columns and mode for categorical columns
for column in data.columns:
    if data[column].dtype == 'object':
        data[column].fillna(data[column].mode()[0], inplace=True)
    else:
        data[column].fillna(data[column].mean(), inplace=True)
```

```
Missing values before handling:
Date              GameID         Drive        qtr       down       time           TimeUnder        TimeSecs          PlayTimeDiff
dtype: int64
```

```
# Using Z-score to detect outliers
from scipy.stats import zscore

z_scores = np.abs(zscore(data.select_dtypes(include=[np.number])))
outliers = (z_scores > 3).any(axis=1)
data_cleaned = data[~outliers]


if 'Date' in data.columns:
    data['Date'] = pd.to_datetime(data['Date'], errors='coerce')
    data['Date'] = data['Date'].dt.strftime('%Y-%m-%d')


data_cleaned.drop_duplicates(inplace=True)


if 'Date' in data.columns:
    data_cleaned['Year'] = pd.to_datetime(data_cleaned['Date']).dt.year
    data_cleaned['Month'] = pd.to_datetime(data_cleaned['Date']).dt.month
    data_cleaned['Day'] = pd.to_datetime(data_cleaned['Date']).dt.day


# Normalization
scaler = MinMaxScaler()
numeric_columns = data_cleaned.select_dtypes(include=[np.number]).columns
data_cleaned[numeric_columns] = scaler.fit_transform(data_cleaned[numeric_columns])
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-7-6fa8879419a0> in <cell line: 4>()
      2 scaler = MinMaxScaler()
      3 numeric_columns = data_cleaned.select_dtypes(include=[np.number]).columns
----> 4 data_cleaned[numeric_columns] =
scaler.fit_transform(data_cleaned[numeric_columns])

                                  5 frames
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py in
check_array(array, accept_sparse, accept_large_sparse, dtype, order, copy,
force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_features,
estimator, input_name)
    776         )
    777         if all(isinstance(dtype_iter, np.dtype) for dtype_iter in
dtypes_orig):
--> 778             dtype_orig = np.result_type(*dtypes_orig)
    779
    780     elif hasattr(array, "iloc") and hasattr(array, "dtype"):
```

Next steps: Explain error

```
data_cleaned.hist(bins=50, figsize=(20, 15))
plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-9-d3b8f83c564c> in <cell line: 1>()
----> 1 data_cleaned.hist(bins=50, figsize=(20, 15))
      2 plt.show()

 1 frames
/usr/local/lib/python3.10/dist-packages/pandas/plotting/_matplotlib/hist.py in
hist_frame(data, column, by, grid, xlabelsize, xrot, ylabelsize, yrot, ax, sharex,
sharey, figsize, layout, bins, legend, **kwds)
    512
    513     if naxes == 0:
--> 514         raise ValueError(
    515             "hist method requires numerical or datetime columns, nothing to
plot."
    516         )

ValueError: hist method requires numerical or datetime columns, nothing to plot.
```

Next steps: Explain error

```
print("Missing values after handling:")
print(data_cleaned.isnull().sum())
print("Duplicates after handling:", data_cleaned.duplicated().sum())
```

```
Missing values after handling:
Date              GameID         Drive        qtr       down       time           TimeUnder        TimeSecs          PlayTimeDiff
dtype: int64
Duplicates after handling: 0
```

```
# Create a log file to document steps
with open("data_cleaning_log.txt", "w") as log_file:
    log_file.write("Data Cleaning and Processing Log\n")
    log_file.write("Missing values before handling:\n")
    log_file.write(str(data.isnull().sum()) + "\n")
    log_file.write("Missing values after handling:\n")
    log_file.write(str(data_cleaned.isnull().sum()) + "\n")
    log_file.write("Duplicates after handling: " + str(data_cleaned.duplicated().sum()) + "\n")

# Save the cleaned dataset
data_cleaned.to_csv('Novanectar_cleaned.csv', index=False)
print("Data cleaning and processing complete. Cleaned data saved to 'Novanectar_cleaned.csv'.")
```

```
Data cleaning and processing complete. Cleaned data saved to 'Novanectar_cleaned.csv'.
```

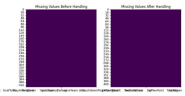
```
# Load the original and cleaned datasets
original_data = pd.read_csv('Novanectar.csv')
cleaned_data = pd.read_csv('Novanectar_cleaned.csv')

# 1. Visualize Missing Values
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
sns.heatmap(original_data.isnull(), cbar=False, cmap='viridis')
plt.title('Missing Values Before Handling')

plt.subplot(1, 2, 2)
sns.heatmap(cleaned_data.isnull(), cbar=False, cmap='viridis')
plt.title('Missing Values After Handling')

plt.tight_layout()
plt.show()
```

```
<ipython-input-14-675061a53b98>:15: UserWarning: Tight layout not applied. The left a
  plt.tight_layout()
```



```
# Using box plots to visualize outliers
numeric_columns = original_data.select_dtypes(include=[np.number]).columns

plt.figure(figsize=(15, 10))
for i, column in enumerate(numeric_columns):
    plt.subplot(len(numeric_columns) // 2 + 1, 2, i + 1)
    sns.boxplot(data=original_data[column])
    plt.title(f'Outliers in {column} Before Removal')
```

```
plt.tight_layout()
plt.show()

plt.figure(figsize=(15, 10))
for i, column in enumerate(numeric_columns):
    plt.subplot(len(numeric_columns) // 2 + 1, 2, i + 1)
    sns.boxplot(data=cleaned_data[column])
    plt.title(f'Outliers in {column} After Removal')

plt.tight_layout()
plt.show()

# 3. Histograms of Numerical Columns
plt.figure(figsize=(20, 15))
for i, column in enumerate(numeric_columns):
    plt.subplot(len(numeric_columns) // 2 + 1, 2, i + 1)
    sns.histplot(cleaned_data[column], kde=True)
    plt.title(f'Distribution of {column} After Cleaning')
plt.tight_layout()
plt.show()
```

```
<Figure size 1500x1000 with 0 Axes>
<Figure size 1500x1000 with 0 Axes>
<Figure size 2000x1500 with 0 Axes>
```