Exploratory Data Analysis

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

```
df_train = pd.read_csv("/content/train.csv")
df_test = pd.read_csv("/content/test (1).csv")
```

```
df_test_format = df_test[['date', 'Item Id']].copy()
```

```
df_train.head(5)
```

	ID	date	Item Id	Item Name	ad_spend	anarix_id	units	unit_price
0	2022-04-12_B09KDTS4DC	2022-04-12	B09KDTS4DC	NapQueen Elizabeth 8" Gel Memory Foam Mattress...	NaN	NAPQUEEN	0.0	0.0
1	2022-04-12_B09MR2MLZH	2022-04-12	B09MR2MLZH	NapQueen 12 Inch Bamboo Charcoal Queen Size Me...	NaN	NAPQUEEN	0.0	0.0
2	2022-04-12_B09KSYL73R	2022-04-12	B09KSYL73R	NapQueen Elsa 8" Innerspring Mattress, Twin XL	NaN	NAPQUEEN	0.0	0.0
3	2022-04-12_B09KT5HMNY	2022-04-12	B09KT5HMNY	NapQueen Elsa 6" Innerspring Mattress, Twin	NaN	NAPQUEEN	0.0	0.0
4	2022-04-12_B09KTF8ZDQ	2022-04-12	B09KTF8ZDQ	NapQueen Elsa 6" Innerspring Mattress, Twin XL	NaN	NAPQUEEN	0.0	0.0

```
df_test.head(5)
```

	ID	date	Item Id	Item Name	ad_spend	anarix_id	unit_price
0	2024-07-01_B09KDR64LT	2024-07-01	B09KDR64LT	NapQueen Elizabeth 10" Gel Memory Foam Mattres...	NaN	NAPQUEEN	0.0
1	2024-07-01_B09KDTS4DC	2024-07-01	B09KDTS4DC	NapQueen Elizabeth 8" Gel Memory Foam Mattress...	NaN	NAPQUEEN	0.0
2	2024-07-01_B09KDTHJ6V	2024-07-01	B09KDTHJ6V	NapQueen Elizabeth 12" Gel Memory Foam Mattres...	NaN	NAPQUEEN	0.0

Next steps: Generate code with `df_test` View recommended plots New interactive sheet

```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 101490 entries, 0 to 101489
Data columns (total 8 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   ID          101490 non-null  object
 1   date        101490 non-null  object
 2   Item Id     101488 non-null  object
 3   Item Name   99658 non-null   object
 4   ad_spend    77303 non-null   float64
 5   anarix_id   101490 non-null  object
 6   units       83592 non-null   float64
 7   unit_price 101490 non-null float64
dtypes: float64(3), object(5)
memory usage: 6.2+ MB
```

```
df_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2833 entries, 0 to 2832
Data columns (total 7 columns):
 #    Column      Non-Null Count   Dtype
----  ------------------------------------  ----------
 0    ID          2833 non-null    object
 1    date        2833 non-null    object
 2    Item Id     2833 non-null    object
 3    Item Name   2489 non-null    object
 4    ad_spend    1382 non-null    float64
 5    anarix_id   2833 non-null    object
 6    unit_price  2833 non-null    float64
```

```
dtypes: float64(2), object(5)
memory usage: 155.1+ KB
```

```
# Checking for missing values in the training dataset
print("Missing values in training data:")
print(df_train.isnull().sum())
```

```
Missing values in training data:
ID                 0
date               0
Item Id            2
Item Name       1832
ad_spend       24187
anarix_id          0
units          17898
unit_price         0
dtype: int64
```

```
# Checking for missing values in the testing dataset
print("Missing values in testing data:")
print(df_test.isnull().sum())
```

```
Missing values in testing data:
ID                0
date              0
Item Id           0
Item Name       344
ad_spend       1451
anarix_id         0
unit_price        0
dtype: int64
```

```
# Filling null values with 0
df_train = df_train.fillna({'ad_spend': 0, 'units': 0})
df_test = df_test.fillna({'ad_spend': 0})
```

```
df_train
```

	ID	date	Item Id	Item Name	ad_spend	anarix_id	units	unit_price
0	2022-04-12_B09KDTS4DC	2022-04-12	B09KDTS4DC	NapQueen Elizabeth 8" Gel Memory Foam Mattress...	0.00	NAPQUEEN	0.0	0.00
1	2022-04-12_B09MR2MLZH	2022-04-12	B09MR2MLZH	NapQueen 12 Inch Bamboo Charcoal Queen Size Me...	0.00	NAPQUEEN	0.0	0.00
2	2022-04-12_B09KSYL73R	2022-04-12	B09KSYL73R	NapQueen Elsa 8" Innerspring Mattress, Twin XL	0.00	NAPQUEEN	0.0	0.00
3	2022-04-12_B09KT5HMNY	2022-04-12	B09KT5HMNY	NapQueen Elsa 6" Innerspring Mattress, Twin	0.00	NAPQUEEN	0.0	0.00
4	2022-04-12_B09KTF8ZDQ	2022-04-12	B09KTF8ZDQ	NapQueen Elsa 6" Innerspring Mattress, Twin XL	0.00	NAPQUEEN	0.0	0.00
...	...	...	...	...	...	...	...	...
101485	2024-05-31_B0CR4BGLK5	2024-05-31	B0CR4BGLK5	NaN	604.73	NAPQUEEN	0.0	0.00
101486	2024-05-31_B0CR4BG4ZW	2024-05-31	B0CR4BG4ZW	NaN	261.21	NAPQUEEN	2.0	225.32
101487	2024-05-31_B0CR49NR3B	2024-05-31	B0CR49NR3B	NaN	0.00	NAPQUEEN	0.0	0.00
101488	2024-05-31_B0CR49N6MQ	2024-05-31	B0CR49N6MQ	NaN	0.00	NAPQUEEN	0.0	0.00
101489	2024-05-31_B0CR4BK4FW	2024-05-31	B0CR4BK4FW	NaN	0.00	NAPQUEEN	0.0	0.00

101490 rows × 8 columns

```
df_test
```

	ID	date	Item Id	Item Name	ad_spend	anarix_id	unit_price
0	2024-07-01_B09KDR64LT	2024-07-01	B09KDR64LT	NapQueen Elizabeth 10" Gel Memory Foam Mattres...	0.00	NAPQUEEN	0.0
1	2024-07-01_B09KDTS4DC	2024-07-01	B09KDTS4DC	NapQueen Elizabeth 8" Gel Memory Foam Mattress...	0.00	NAPQUEEN	0.0
2	2024-07-01_B09KDTHJ6V	2024-07-01	B09KDTHJ6V	NapQueen Elizabeth 12" Gel Memory Foam Mattres...	0.00	NAPQUEEN	0.0
3	2024-07-01_B09KDQ2BWY	2024-07-01	B09KDQ2BWY	NapQueen Elizabeth 12" Gel Memory Foam Mattres...	0.00	NAPQUEEN	0.0
4	2024-07-01_B09KDYY3SB	2024-07-01	B09KDYY3SB	NapQueen Elizabeth 10" Gel Memory Foam Mattres...	101.72	NAPQUEEN	1094.5
...	...	...	...	...	...	...	...
2828	2024-07-28_B0BRCW2B64	2024-07-28	B0BRCW2B64	NapQueen Anula Green Tea 12", Queen	11.78	NAPQUEEN	0.0
2829	2024-07-28_B0CFV6V981	2024-07-28	B0CFV6V981	NaN	1.17	NAPQUEEN	0.0

Next steps: Generate code with `df_test` View recommended plots New interactive sheet

```
# Checking for missing values in the training dataset
print("Missing values in training data:")
print(df_train.isnull().sum())
```

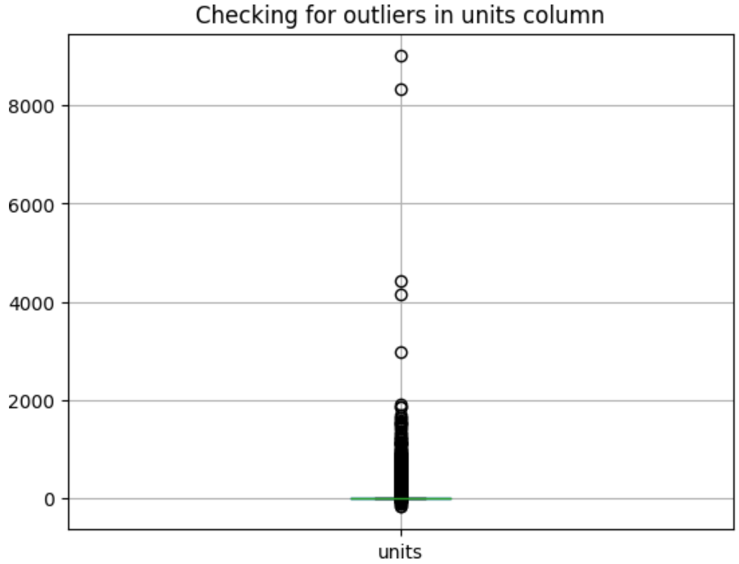
```
Missing values in training data:
ID               0
date             0
Item Id          2
Item Name     1832
ad_spend         0
anarix_id        0
units            0
unit_price       0
dtype: int64
```

```
# Checking for missing values in the testing dataset
print("Missing values in testing data:")
print(df_test.isnull().sum())
```

```
Missing values in testing data:
ID              0
date            0
Item Id         0
Item Name     344
ad_spend        0
anarix_id       0
unit_price      0
dtype: int64
```

```
# Checking for outliers
df_train.boxplot(column='units')
plt.title('Checking for outliers in units column')
```

```
Text(0.5, 1.0, 'Checking for outliers in units column')
```



Correlation

```
#Correlation coefficient between "ad_spend" and "units"
corr_coef = df_train['ad_spend'].corr(df_train['units'])
print(corr_coef)
```

```
0.74453011658838
```

```
#Correlation coefficient between "ad_spend" and "unit_price"
corr_coef = df_train['ad_spend'].corr(df_train['unit_price'])
print(corr_coef)
```

```
0.054472062228094906
```

Pearson correlation

```
from scipy.stats import pearsonr
r, p = pearsonr(df_train["ad_spend"], df_train["units"])
print(r)
```

```
0.7445301165883786
```

Feature Engineering

```
# Selected features
df_train_f = df_train[['ad_spend', 'units', 'unit_price']].copy()
```

```
df_test_f = df_test[['ad_spend', 'unit_price']].copy()
```

Separating target and predictor variables

```
X = df_train_f.drop(columns = 'units', axis=1)
Y = df_train_f['units']
```

```
print(X)
```

```
        ad_spend  unit_price
0           0.00        0.00
1           0.00        0.00
2           0.00        0.00
3           0.00        0.00
4           0.00        0.00
...          ...         ...
101485    604.73        0.00
101486    261.21      225.32
101487      0.00        0.00
101488      0.00        0.00
```

```
101489      0.00        0.00

[101490 rows x 2 columns]
```

```
print(Y)
```

```
0         0.0
1         0.0
2         0.0
3         0.0
4         0.0
         ...
101485    0.0
101486    2.0
101487    0.0
101488    0.0
101489    0.0
Name: units, Length: 101490, dtype: float64
```

Model Selection

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Training a Linear Regression model
model = LinearRegression()
model.fit(X, Y)

# Using the trained model to make predictions on the testing data
predictions = model.predict(df_test_f[['ad_spend', 'unit_price']])
predictions = [round(x) for x in predictions]

# Saving the predicted data to a new CSV file
predicted_df = df_test_format.copy()
predicted_df['units'] = predictions

# Calculating the Mean Squared Error(MSE)
mse = mean_squared_error(predicted_df['units'], predictions)
print("Mean Squared Error (MSE):", mse)

# Calculating the accuracy
accuracy = 1 - (mse / (predicted_df['units'].var() * len(predicted_df)))
print("Accuracy:", accuracy)

predicted_df.to_csv('predicted_data_lr.csv', index=False)
```

```
Mean Squared Error (MSE): 0.0
Accuracy: 1.0
```

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error

# Training a Decision Tree model
model = DecisionTreeRegressor()
model.fit(X, Y)

# Using the trained model to make predictions on the testing data
predictions = model.predict(df_test_f[['ad_spend', 'unit_price']])
predictions = [round(x) for x in predictions]

# Saving the predicted data to a new CSV file
predicted_df = df_test_format.copy()
predicted_df['units'] = predictions

# Calculating the Mean Squared Error(MSE)
mse = mean_squared_error(predicted_df['units'], predictions)
print("Mean Squared Error (MSE):", mse)

# Calculating the accuracy
accuracy = 1 - (mse / (predicted_df['units'].var() * len(predicted_df)))
print("Accuracy:", accuracy)

predicted_df.to_csv('predicted_data_dt.csv', index=False)
```

```
Mean Squared Error (MSE): 0.0
Accuracy: 1.0
```

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
```

```
# Training a Random Forest model
model = RandomForestRegressor()
model.fit(X, Y)

# Using the trained model to make predictions on the testing data
predictions = model.predict(df_test_f[['ad_spend', 'unit_price']])
predictions = [round(x) for x in predictions]

# Saving the predicted data to a new CSV file
predicted_df = df_test_format.copy()
predicted_df['units'] = predictions

# Calculating the Mean Squared Error(MSE)
mse = mean_squared_error(predicted_df['units'], predictions)
print("Mean Squared Error (MSE):", mse)

# Calculating the accuracy
accuracy = 1 - (mse / (predicted_df['units'].var() * len(predicted_df)))
print("Accuracy:", accuracy)

predicted_df.to_csv('predicted_data_rf.csv', index=False)
```

```
Mean Squared Error (MSE): 0.0
Accuracy: 1.0
```

Hyperparameter Tuning

```
# Importing GridSearchCV to search and find the optimal combination of hyperparameters for a given model
# (creating a 'grid' of possible combinations)
from sklearn.model_selection import GridSearchCV


param_grid = {
    'max_depth': [3, 5, 10],
    'min_samples_split': [2, 5, 10]
}

grid_search = GridSearchCV(DecisionTreeRegressor(), param_grid, cv=5)
grid_search.fit(X, Y)

print("Best parameters:", grid_search.best_params_)
print("Best score:", grid_search.best_score_)

best_model = grid_search.best_estimator_
predictions = best_model.predict(df_test_f[['ad_spend', 'unit_price']])
```

```
Best parameters: {'max_depth': 10, 'min_samples_split': 2}
Best score: 0.37093712066890766
```

```
param_grid = {
    'max_depth': [3, 5, 10],
    'min samples split': [2, 5, 10]
```