**PRODUCT SALES ANALYSIS**

**PHASE-3**

**NAME: NAGENTHIRA KUMAR D**

**NM ID:au211521106104**

# STEPS FOR CLEANING DATASET USING PYTHON:

1.Handle Missing Values: Locate and deal with any missing data points. Rows with missing data can either be eliminated or filled in utilizing methods like mean, median, or interpolation, depending on the circumstances.

2. Remove Duplicates: - Search for and eliminate records that are identical, particularly if the dataset was produced from various sources. Analysis results may be skewed by duplicates.

3. Data Type Conversion:- Ascertain that the data types are suitable for analysis. For instance, dates should be formatted as dates, and categorical variables should have the proper labels. Numerical data should also be presented in its proper format.

4. Check for Outliers: - Find anomalies in the numerical data that can have a big impact on the analysis. Choose whether to eliminate outliers or to change the data to lessen their influence.

5.Normalize and Standardize Data:- If you intend to employ methods that are sensitive to the scale of variables, standardize or normalize numerical data. It is possible to apply normalization (scaling values between 0 and 1) or standardization (subtracting the mean and dividing by the standard deviation).

6. Validate Categorical Data: Verify that categorical variables only contain valid values by validating them. Correct categories that are inconsistent or misspelled.

7. Address Inconsistent Data: Keep an eye out for errors made when entering data, particularly in text areas. For instance, when referring to the same area, "NY," "New York," and "New York City" should all be used consistently.

8. Check Integrity Constraints - Verify that the connections between various columns make sense. For instance, the time of arrival should be later than the time of departure.

9. Extract pertinent data from text fields using parsing and extraction software. For instance, if a field contains both a date and an hour, separate them into different columns.

10. Validate the coordinates in the dataset if it contains geographic information to make sure they are within the expected range for the area of interest.

11. Validate data across various fields via cross-field validation. Make sure the estimated speed is within acceptable bounds, for instance, if you have a distance field and a time field.

12. note Changes: Keep a note of any modifications that were made during cleaning. This documentation is useful for reproducibility and transparency.

13. Examine the Clean Dataset:On the cleaned dataset, run preliminary analysis to make sure the data behaves as predicted. This process helps identify any problems that might have gone unnoticed during cleaning.

*# Import necessary libraries*

import pandas as pd

*# Load your dataset into a pandas DataFrame*

df = pd.read\_csv('/content/statsfinal.csv')

*# Drop duplicate rows*

df = df.drop\_duplicates()

*# Handle missing values*

df = df.dropna() *# Drop rows with any NaN values # OR fill missing values with a specific value*

*# df = df.fillna(value)*

*# Remove unwanted columns*

*# df = df.drop(columns=['Q-P1'])*

*# Convert data types if needed*

*# df['Q-P1'] = df['Q-P1'].astype('desired\_data\_type')*

*# Remove leading/trailing whitespaces from string columns # df['Q-P1'] = df['Q-P1'].str.strip()*

*# Perform other data cleaning operations as per your specific requirements*

*# Save the cleaned data to a new CSV file* df.to\_csv('cleaned\_data.csv', index=False) from google.colab import files

*# Assuming your cleaned data file is named 'cleaned\_data.csv' # Replace 'cleaned\_data.csv' with the actual file name if it's different*

files.download('cleaned\_data.csv')

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object> df.info()

<class 'pandas.core.frame.DataFrame'>

Int64Index: 4600 entries, 0 to 4599 Data columns (total 10 columns):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # |  | Column | Non-Null Count |  | Dtype |
| 0 |  | Unnamed: 0 | 4600 non-null |  | int64 |

1 Date 4600 non-null object

2 Q-P1 4600 non-null int64

1. Q-P2 4600 non-null int64
2. Q-P3 4600 non-null int64
3. Q-P4 4600 non-null int64
4. S-P1 4600 non-null float64
5. S-P2 4600 non-null float64
6. S-P3 4600 non-null float64
7. S-P4 4600 non-null float64 dtypes: float64(4), int64(5), object(1) memory usage: 395.3+ KB

df.tail(3)

Unnamed: 0 Date Q-P1 Q-P2 Q-P3 Q-P4 S-P1 S-

P2 \

4597 4597 01-02-2023 6289 3143 3588 474 19936.13

19926.62

4598 4598 02-02-2023 3122 1188 5899 517 9896.74

7531.92

4599 4599 03-02-2023 1234 3854 2321 406 3911.78

24434.36

|  |  |  |
| --- | --- | --- |
|  | S-P3 | S-P4 |
| 4597 | 19446.96 | 3379.62 |
| 4598 | 31972.58 | 3686.21 |
| 4599 | 12579.82 | 2894.78 |

df.head(3)

Unnamed: 0 Date Q-P1 Q-P2 Q-P3 Q-P4 S-P1 S-P2

\

0 0 13-06-2010 5422 3725 576 907 17187.74 23616.50

1 1 14-06-2010 7047 779 3578 1574 22338.99 4938.86

2 2 15-06-2010 1572 2082 595 1145 4983.24 13199.88

S-P3 S-P4 0 3121.92 6466.91

1 19392.76 11222.62

2 3224.90 8163.85

df.describe()

Unnamed: 0 Q-P1 Q-P2 Q-P3 Q-P4

\

count 4600.000000 4600.000000 4600.000000 4600.000000 4600.000000

mean 2299.500000 4121.849130 2130.281522 3145.740000 1123.500000

std 1328.049949 2244.271323 1089.783705 1671.832231 497.385676

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| min | 0.000000 | 254.000000 | 251.000000 | 250.000000 | 250.000000 |
|  |  |  |  |  |  |
| 25% | 1149.750000 | 2150.500000 | 1167.750000 | 1695.750000 | 696.000000 |
|  |  |  |  |  |  |
| 50% | 2299.500000 | 4137.000000 | 2134.000000 | 3202.500000 | 1136.500000 |
|  |  |  |  |  |  |
| 75% | 3449.250000 | 6072.000000 | 3070.250000 | 4569.000000 | 1544.000000 |
|  |  |  |  |  |  |
| max | 4599.000000 | 7998.000000 | 3998.000000 | 6000.000000 | 2000.000000 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | S-P1 | S-P2 | S-P3 | S-P4 |
| count | 4600.000000 | 4600.000000 | 4600.000000 | 4600.000000 |
| mean | 13066.261743 | 13505.984848 | 17049.910800 | 8010.555000 |
| std | 7114.340094 | 6909.228687 | 9061.330694 | 3546.359869 |
| min | 805.180000 | 1591.340000 | 1355.000000 | 1782.500000 |
| 25% | 6817.085000 | 7403.535000 | 9190.965000 | 4962.480000 |
| 50% | 13114.290000 | 13529.560000 | 17357.550000 | 8103.245000 |
| 75% | 19248.240000 | 19465.385000 | 24763.980000 | 11008.720000 |
| max | 25353.660000 | 25347.320000 | 32520.000000 | 14260.000000 |

df.isna().sum()

Unnamed: 0 0

Date 0

Q-P1 0

Q-P2 0

Q-P3 0

Q-P4 0

S-P1 0

S-P2 0

S-P3 0

S-P4 0

dtype: int64

df.dropna(inplace=True) df.isna().sum()

Unnamed: 0 0

Date 0

Q-P1 0

Q-P2 0

Q-P3 0

Q-P4 0

S-P1 0

S-P2 0

S-P3 0

S-P4 0

dtype: int64

df[df.duplicated()]

Empty DataFrame

Columns: [Unnamed: 0, Date, Q-P1, Q-P2, Q-P3, Q-P4, S-P1, S-P2, S-P3, S-P4]

Index: []

















