AI and DS-1

Experiment1

Aim: Introduction to Data science and Data preparation using Pandas.

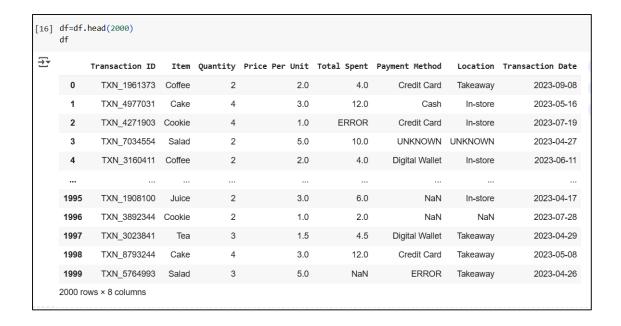
Theory:

1. Load data in Pandas:

We first mounted Google Drive to the Colab environment to access files stored in drive. Then, we used read_csv to read the contents of cafe_sales.csv and load it into the DataFrame df for analysis.

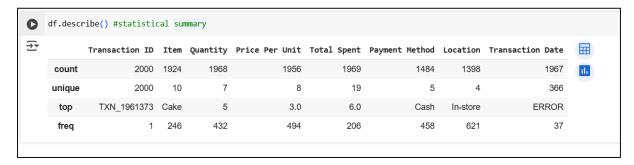
	<pre>from google.colab import drive import pandas as pd drive.mount('/content/drive') df = pd.read_csv('/content/drive/My Drive/cafe_sales.csv') df</pre>											
₹		l at /content/dr										
		Transaction ID				•	Payment Method		Transaction Date			
	0	TXN_1961373	Coffee	2	2.0	4.0	Credit Card	Takeaway	2023-09-08			
	1	TXN_4977031	Cake	4	3.0	12.0	Cash	In-store	2023-05-16			
	2	TXN_4271903	Cookie	4	1.0	ERROR	Credit Card	In-store	2023-07-19			
	3	TXN_7034554	Salad	2	5.0	10.0	UNKNOWN	UNKNOWN	2023-04-27			
	4	TXN_3160411	Coffee	2	2.0	4.0	Digital Wallet	In-store	2023-06-11			
	9995	TXN_7672686	Coffee	2	2.0	4.0	NaN	UNKNOWN	2023-08-30			
	9996	TXN_9659401	NaN	3	NaN	3.0	Digital Wallet	NaN	2023-06-02			
	9997	TXN_5255387	Coffee	4	2.0	8.0	Digital Wallet	NaN	2023-03-02			
	9998	TXN_7695629	Cookie	3	NaN	3.0	Digital Wallet	NaN	2023-12-02			
	9999	TXN_6170729	Sandwich	3	4.0	12.0	Cash	In-store	2023-11-07			
	10000 rows × 8 columns											

By using df.head(2000), we limited the dataset to 2000 rows.

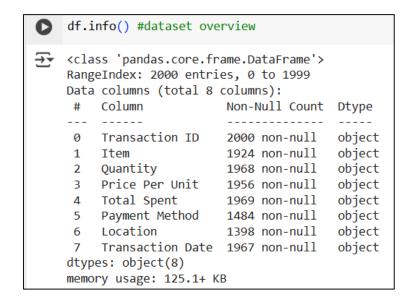


2. Description of dataset:

df.describe() shows statistical summary of columns in a dataframe.



df.info() provides complete overview of the dataframe by providing column names and their datatypes, non null values count for each column, memory usage.



3. Dropping columns that aren't useful:

df.drop(['column name'], axis=1) removes the Transaction ID column, where axis=1 specifies we are dropping a column and not a row.

df.dr df	op(['Tra	nsaction I	D'], axis=1, inp	olace=True)			
3	Item	Quantity	Price Per Unit	Total Spent	Payment Method	Location	Transaction Date
0	Coffee	2	2.0	4.0	Credit Card	Takeaway	2023-09-08
1	Cake	4	3.0	12.0	Cash	In-store	2023-05-16
2	Cookie	4	1.0	ERROR	Credit Card	In-store	2023-07-19
3	Salad	2	5.0	10.0	UNKNOWN	UNKNOWN	2023-04-27
4	Coffee	2	2.0	4.0	Digital Wallet	In-store	2023-06-11
1995	Juice	2	3.0	6.0	NaN	In-store	2023-04-17
1996	Cookie	2	1.0	2.0	NaN	NaN	2023-07-28
1997	Tea	3	1.5	4.5	Digital Wallet	Takeaway	2023-04-29
1998	Cake	4	3.0	12.0	Credit Card	Takeaway	2023-05-08
1999	Salad	3	5.0	NaN	ERROR	Takeaway	2023-04-26

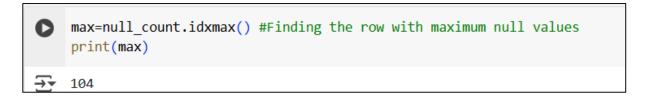
4. Dropping rows with maximum missing values:

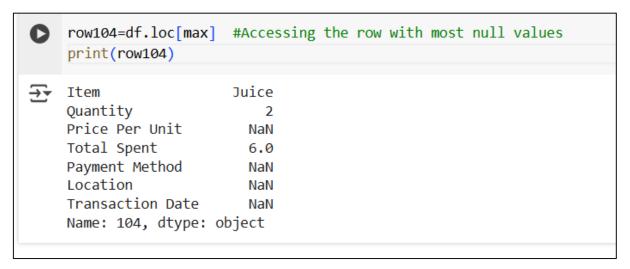
In order to drop rows with maximum missing values we first calculated number of null values in each row and using idxmax found out the row with the maximum null values (here it is row 104) and then dropped the row using df.drop(max,axis=0)

```
null_count = df.isnull().sum(axis=1) #calculating no. of null values in each row

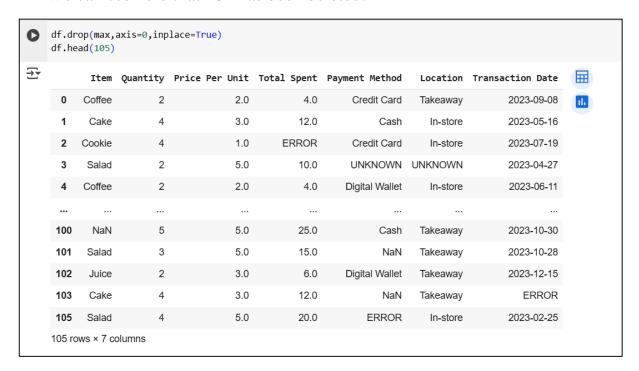
print(null_count)

0 0
1 0
2 0
3 0
4 0
...
1995 1
1996 2
1997 0
1998 0
1999 1
Length: 2000, dtype: int64
```





We can see here that 104 has been deleted.



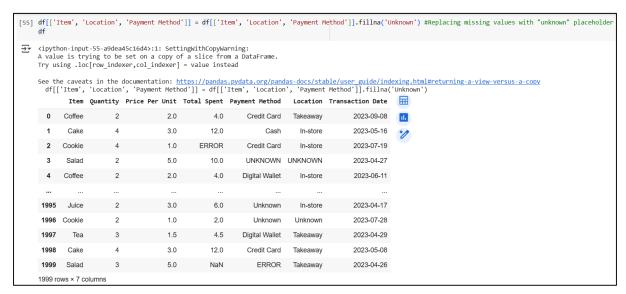
Alternatively, we can get the count of all rows that have max missing values and drop them at once.

```
max_null_count = df.isnull().sum(axis=1).max() # Get max missing count
rows_to_drop = df[df.isnull().sum(axis=1) == max_null_count].index # Find all such rows
print(rows_to_drop)
df.drop(rows_to_drop, axis=0, inplace=True) # Drop the rows

Index([104, 1379, 2853, 5851], dtype='int64')
```

5. Handling missing data:

For the missing data in the categorical columns Item, Location, and Payment Method, we have replaced the missing values with the placeholder 'unknown'.



For handling missing values in transaction date we first converted all invalid values to NaT and then replaced the missing values (including NaT) to a default placeholder date.

For missing values in Quantity column, we converted invalid values to NaN and then replaced them with the median.

Filled the missing values in Price per unit using mode (most frequent value) based on each item.

	<pre>df.loc[:, for item i mode_v</pre>	<pre>rice Per Unit'] == 'unknown', 'Price Per Unit'] = pd.NA rice Per Unit'] = pd.to_numeric(df['Price Per Unit'], errors='coerce') df['Item'].unique(): ue = df.loc[df['Item'] == item, 'Price Per Unit'].mode()[0] df['Item'] == item) & df['Price Per Unit'].isna(), 'Price Per Unit'] = mode_valuer Unit']</pre>
	Pri	Per Unit
	0	2.0
	1	3.0
	2	1.0
	3	5.0
	4	2.0
	1995	3.0
	1996	1.0
	1997	1.5
	1998	3.0

Lastly replaced missing values in Total Spent column with 0

.65]	df.loc	[:, 'To	tal Spent'] = df['Total Sp	oent'].fillna	(0)		
D	df							
		Item	Quantity	Price Per Unit	Total Spent	Payment Method	Location	Transaction Date
	0	Coffee	2	2.0	4.0	Credit Card	Takeaway	2023-09-08
	1	Cake	4	3.0	12.0	Cash	In-store	2023-05-16
	2	Cookie	4	1.0	ERROR	Credit Card	In-store	2023-07-19
	3	Salad	2	5.0	10.0	UNKNOWN	UNKNOWN	2023-04-27
	4	Coffee	2	2.0	4.0	Digital Wallet	In-store	2023-06-11
	1995	Juice	2	3.0	6.0	Unknown	In-store	2023-04-17
	1996	Cookie	2	1.0	2.0	Unknown	Unknown	2023-07-28
	1997	Tea	3	1.5	4.5	Digital Wallet	Takeaway	2023-04-29
	1998	Cake	4	3.0	12.0	Credit Card	Takeaway	2023-05-08
	1999	Salad	3	5.0	0	ERROR	Takeaway	2023-04-26
	1999 rd	ows × 7 co	olumns					

6. Create dummy variables:

Dummy variables are used to convert categorical values into numerical format so that machine learning models and statistical analysis can process them.

Here we have used pd.get_dummies() to convert Payment method and location to numeric format.

0	df_dummies = pd.get_dummies(df, columns=['Payment Method', 'Location'], drop_first=True)										
	df_du	mmies[df_d	ummies.sel	lect_dtypes('bo	ool').columns] = df_dummies.se	elect_dtypes('bool').astyp	e(int)			
0	df_du	mmies									
∑ *		Item (Quantity	Price Per Unit	Total Spent	Transaction Date	Payment Method_Credit Card	Payment Method_Digital Wallet	Payment Method_Unknown	Location_Takeaway	Location_Unknown
	0	Coffee	2	2.0	4.0	2023-09-08	1	0	0	1	0
	1	Cake	4	3.0	12.0	2023-05-16	0	0	0	0	0
	2	Cookie	4	1.0	Unknown	2023-07-19	1	0	0	0	0
	3	Salad	2	5.0	10.0	2023-04-27	0	0	1	0	1
	4	Coffee	2	2.0	4.0	2023-06-11	0	1	0	0	0

7. Find out outliers (manually):

In the cafe sales dataset, the value 25 in Total Spent column can be considered as an outlier since it is significantly higher than the other values in that column that are below 20.

8. Standardization and normalization of columns:

Standardizing data: Centers data around 0 with a standard deviation of 1.

Formula:
$$Z = \frac{X-Mean}{Standard deviation}$$

0		, 'Quantity_standardi tity_standardized']	ed'] = (df['Quantity'] - df['Quantity'].mean()) / df['Quantity'].s
₹	Qu	uantity_standardized	
	0	-0.756744	
	1	0.672383	
	2	0.672383	
	3	-0.756744	
	4	-0.756744	

Normalizing data: Scales values between 0 and 1.

Formula: Xnormalized =
$$\frac{X-X\min}{Xmax-Xmin}$$

```
df.loc[:, 'Total_Spent_normalized'] = (df['Total Spent'] - df['Total Spent'].min()) / (df['Total Spent'].max() - df['Total Spent'].min())
df['Total_Spent_normalized']
        Total Spent normalized
   0
   1
                              0.48
                              0.00
                              0.40
                              0.16
  1995
                              0.24
  1996
                              0.08
                              0.18
  1997
  1998
                              0.48
                              0.00
  1999
 1999 rows × 1 columns
```

Conclusion:

In this experiment, we processed the data set which contained missing and invalid values and transformed it into a clean dataset which can be used for further analysis. Also, we created dummy variables so that categorical values can be converted to numeric values making them suitable for various machine learning algorithms. Lastly, we performed normalization and standardization on the columns to ensure consistency in data scaling.