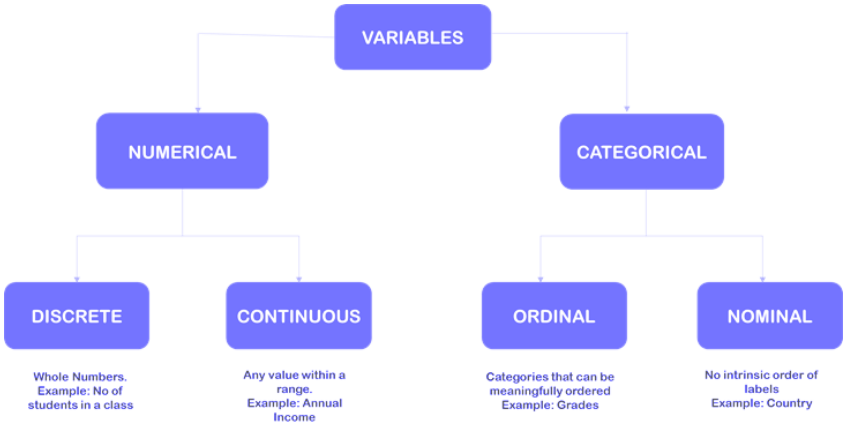
Standard Structure of Exploratory Data Analysis

Libraries used: Pandas, NumPy, Seaborn

1. Identification of variables and data types:



   a. df.dtype
   b. df.shape
   c. df.info()

2. Analysing the basic metrics:

   a. df.describe(include='all')
   b. Including only numeric columns in a DataFrame description: df.describe(include=[numpy.number])
   c. Including only string columns in a DataFrame description: df.describe(include=[np.object])
   d. Including only categorical columns from a DataFrame description: df.describe(include=['category'])
   e. Analysing Null Values: df.isnull().sum()/len(df)*100
   f. Conversions:
      i. To datetime: df['date']=pd.to_datetime(df['date'])
      ii. To other datatype: df['num']=df['num'].astype('int64')

3. Brainstorming:

Domain understanding is a must have while solving business use-case. It helps in identifying important variables from the list of available variables. This might include dropping of redundant info/noise from the dataset.

Few things to keep in mind:

a. While doing EDA it might be helpful to have an understanding of how the data is going to be used

b. An outcome of EDA could be the insight that data is insufficient for our problem statement.

c. EDA can also help in brainstorming other use cases which the business might not have thought of.

4. Non-Graphical Analysis:

a. `df.column.value_counts()`

b. `df.column.value_counts(normalize=True)*100`

c. The nunique() function in Pandas returns a series with several distinct observations in a column: `df['column'].nunique()`

d. `df.drop_duplicates()`

e. Similarly, the unique() function of pandas returns the list of unique values in the dataset: `df['column'].unique()`

f. For categorical variables, check `df.groupby(var_x).agg({var_y:['count', 'mean'']})`

5. Visual Analysis:

a. **Histogram/Kdeplot**

i. Distribution of the data — Whether the data is normally distributed or if it's skewed (to the left or right)

ii. To identify outliers — Extremely low or high values that do not fall near any other data points

Visualization Examples:

1. `df['col'].hist(bins=25)`
2. `seaborn.kdeplot(data=df, x="col")`
3. `seaborn.distplot(data=df, x="col")`

b. **Box Plot:** visual representation of the statistical summary of a given data set. The Summary includes:

i. Minimum
ii. First Quartile
iii. Median (Second Quartile)

iv. Third Quartile
v. Maximum

Also, It is also used to identify the outliers in the dataset

```
seaborn.boxplot(x='Col1', y='Col2', data=df)
```

6. Bivariate Analysis:

a. Correlation- `seaborn.heatmap(df.corr(),annot=True)`
b. Scatterplot- `seaborn.scatterplot(x='col1', y='col2', df=data)`
c. Pairplot – `seaborn.pairplot(df.select_dtypes(include='float64'))`

7. Correlation and Covariance:

a. For sample: `Covariance_S= numpy.cov(X,Y)[0][1]`
b. For population: `Covariance_P= numpy.cov(X,Y, bias=True)[0][1]`
c. Correlation Coefficient: `numpy.corrcoef(X,Y)[0][1]`
d. Visualization: `seaborn.heatmap(df.corr(), annot=True)`

8. Variable transformations:

Feature Transformation and Scaling Techniques to Boost Your Model Performance

a. **MinMaxScaler:** It just scales all the data between 0 and 1.
b. **StandardScaler:** scales the values such that the mean is 0 and the standard deviation is 1(or the variance).
c. **RobustScaler:** The Robust Scaler, as the name suggests, is not sensitive to outliers. This scaler-
    i. removes the median from the data
    ii. scales the data by the InterQuartile Range(IQR)
d. **Log Transformer:** primarily used to convert a skewed distribution to a normal distribution/less-skewed distribution `np.log(df['col'])`

Note: You can also check the **value of kurtosis** and the **value of excess kurtosis** which evaluates how thick or thin the tails of a given probability distribution are compared to the normal distribution.

9. Missing value treatment:

a. Deletion of missing rows (rows % < 3)
b. Mean/ Mode/ Median Imputation (rows % between 3-30)
c. KNN Imputation
d. Drop column (if you know business) (rows % >50%)
e. For time series data: Forward fill and backward fill

f. df.fillna(method="ffill")
df.fillna(method="bfill")

10. Outlier treatment:

a. **Z-score:**

```
z_score = (i-mean)/std
if (np.abs(z_score) > threshold):
        outliers.append(i)
```

b. **Interquartile Range(IQR):** compute lower bound = (Q1–1.5*IQR), upper bound = (Q3+1.5*IQR)

11. Business Insights and Recommendations:

a. Recommendations should be one-sentence, succinct, and start with an action verb (create, establish, fund, facilitate, coordinate, etc.).
They should use a **SMART** format - Specific, Measurable, Attainable, Realistic, Timely

b. PRO TIP ► Recommendations stem from the findings. Link each of your recommendations to the finding that supports it, to highlight the direct connection between assessment and action.

   i. You can show this link visually by using a two-columned table: the first column lists the finding, and the respective recommendation is listed adjacently in the second column.
   ii. You can also follow the following framework for writing recommendations- "Our insights show that X, so we can do Y in order to achieve Z"

Mnemonic to remember all of this?

Brag to business using **SMART recommendations**

B- **BR**ainstorming

A- **A**nalysis of variables within dataset

G- **G**raphical Analysis/ Visualization

T- **T**ransformation (for skewed variables)

O- **O**utliers & Missing Values Treatment

B- **B**usiness - Smart recommendations

Policy for sharing projects?

Many of the case studies in the Scaler DSML course are built in partnership with other companies. These partner companies have shared their datasets and problem statements with Scaler and, in doing so, they have given us a huge responsibility to keep their assets protected. Hence, we ask all students to sign an NDA in the beginning of the course.

At Scaler, we also understand that our students who work hard to solve these business cases would like to showcase their solutions on their Github, Kaggle, and Resume.

As such, here are the guidelines:

1. You may **not** post the Business Case Study problem statement anywhere.
2. You may **not** share any datasets anywhere, except for the publicly available datasets from Kaggle or public domain.
3. You may upload your solutions to your Github or Kaggle profiles as long as the solution:
4. does not contain any confidential information or intellectual property owned by the partner company or Scaler.
5. does not compromise Scaler's confidentiality agreement with the partner company
6. does not compromise the NDA signed by the students
7. does not leak the shared datasets.
8. It is acceptable if small snippets or snapshots of the datasets may appear in your solution, as long as it does not compromise the overall privacy of the dataset, the partner company, or the stakeholders of the partner company.
9. You may list the project titles, a short description of the problem, and a short summary of your insights on your resume and on relevant platforms.
10. We recommend that instead of sharing the names of partner companies you use a generic description that makes you look professional. eg:
11. Instead of saying "worked on a project from Uber", you can say "worked on a project from **a top ride-sharing company**"