|  |  |
| --- | --- |
| Data Quality **DATA CLEANING CHEAT SHEET - 1**  1) Duplicate Values 2) Missing Values 3) Invalid Values 4) Outliers  **Duplicates**   1. **duplicated** - returns a Series with True and False values that describe which rows in the DataFrame are duplicated and not 2. **drop\_duplicates** - return DataFrame with duplicate rows removed 3. **reset\_index** - allows you reset the index back to the default 0, 1, 2 etc indexes 4. **subset** - selecting particular rows and columns of data from a DataFrame (or Series) 5. **keep**  * first : drop duplicates except for the first occurrence. * last : drop duplicates except for the last occurrence. * False : drop all duplicates. | df**.**duplicated()  df**.**duplicated(keep **=** 'first')  df**.**duplicated(keep **=** 'last')  a **=** df**.**drop\_duplicates()  b **=** df**.**drop\_duplicates(keep **=** 'first')  c **=** df**.**drop\_duplicates(keep **=** 'last')  a**.**reset\_index()  b**.**reset\_index()  c**.**reset\_index()  a**.**reset\_index(drop **=** **True**)  b**.**reset\_index(drop **=** **True**)  c**.**reset\_index(drop **=** **True**)  df**.**drop\_duplicates (subset **=** ["name","place"]) |
| Missing Values  |  |  | | --- | --- | | **isnull** - returns a DataFrame object where all the values are replaced with a Boolean value True for NULL values, and otherwise False | df**.**isnull()  df**.**isnull()**.**sum()  df['Country']**.**isnull()  df['Country']**.**isnull()**.**sum() | | **notnull** - pandas function that will examine one or multiple values to validate that they are not null | df**.**notnull()  df**.**notnull()**.**sum()  df['Country']**.**notnull()  df['Country']**.**notnull()**.**sum() | | **notna** - returns a DataFrame object where all the values are replaced with a Boolean value True for NOT NA (not-a -number) values, and otherwise False | df**.**notna() | | **all** - returns one value for each column, True if ALL values in that column are True, otherwise False | df**.**isnull()**.**all(axis **=** 0)  df**.**isnull()**.**sum()**.**all()  df['Country']**.**isnull()**.**all()  df['Country']**.**isnull()**.**sum()**.**all() | | **any** - checks whether any value in the caller object (Dataframe or series) is not 0 and returns True for that | df**.**isnull()**.**any(axis **=** 0)  df**.**isnull()**.**sum()**.**any()  df['Country']**.**isnull()**.**any()  df['Country']**.**isnull()**.**sum()**.**any() | | |
| Treating Missing Values  |  |  | | --- | --- | | **dropna** - used to remove missing values | df**.**dropna(how **=** 'all')  df**.**dropna(how **=** 'all' , subset **=** ['Region','Country'])  df**.**dropna(how **=** 'any')  df**.**dropna(how **=** 'any' , subset **=** ['Region','Country']) | | **thresh** - takes integer value which tells minimum amount of na values to drop | df**.**dropna(thresh **=** 1) | | |
| Replacing Missing Values  |  |  | | --- | --- | | **fillna** - replaces the NULL values with a specified value   * forwardfill - replaces the NULL values with the value from the previous row (or previous column, if the axis parameter is set to 'columns' ) * backfill - used to backward fill the missing values in the dataset. It will backward fill the NaN values that are present in the pandas dataframe * value - pass in a value into the value= parameter | df**.**fillna(method **=** 'pad')  df**.**fillna(method **=** 'bfill')  df**.**fillna(value **=** 'not sure')  df['Order\_ID']**.**fillna(value **=** '9515842735')  df['Order\_ID']**.**fillna(df['Order\_ID']**.**mean()) | | **replace** - replaces the specified value with another specified value | df['Order\_ID'] **=** df['Order\_ID']**.**replace(np**.**nan , df['Order\_ID']**.**mean()) | | **interpolate** - used to fill NA values in the dataframe or series. But, this is a very powerful function to fill the missing values | df['Order\_ID']**.**interpolate()  df['Order\_ID']**.**interpolate(method **=** 'polynomial' , order **=** 2) | | |