Categorical column:

* A column with only one category, drop it as it has no contribution in the analysis and modelling.
* if a categorical column has a small number of unique categories (e.g., less than 10), it is typically manageable for analysis. If it has a moderate number of categories (e.g., 10-50), it can still be handled effectively with appropriate techniques. However, if the number of categories is very high (e.g., hundreds or more), it may be necessary to apply dimensionality reduction or other encoding strategies to address the challenges associated with high dimensionality.
* If column has nan value then can’t convert it into int type, to do that first have to handle the nan

But if want to retian the null value then use

df['year'] = df['year'].astype(float), or

df['year'] = pd.to\_numeric(df['year'], errors = 'coerce'), or

df["year"] = df["year"].astype('Int64')

* To extract the coordinates from a dictionary-like object stored in a column with the datatype 'object'

# Example DataFrame

**df = pd.DataFrame({'location': [**

**{'type': 'Point', 'coordinates': [6.08333, 50.775]},**

**{'type': 'Point', 'coordinates': [9.9389, 53.5503]}**

**]})**

# Extract coordinates using apply and lambda function

**df['latitude'] = df['location'].apply(lambda x: x['coordinates'][0])**

**df['longitude'] = df['location'].apply(lambda x: x['coordinates'][1])**

But if there is null value in the column then it will give error

**# Example DataFrame**

**df = pd.DataFrame({'location': [**

**{'type': 'Point', 'coordinates': [6.08333, 50.775]},**

**{'type': 'Point', 'coordinates': [9.9389, 53.5503]},**

**None # Adding a "NaN" value**

**]})**

**# Extract coordinates using apply and lambda function**

**df['latitude'] = df['location'].apply(lambda x: x['coordinates'][0] if isinstance(x, dict) else None)**

**df['longitude'] = df['location'].apply(lambda x: x['coordinates'][1] if isinstance(x, dict) else None)**

* **Convert a objevct type ccolumn with date values in date format**

df["airdate"] = **pd.to\_datetime**(df["airdate"])

**Convert the 'Time' column to datetime format**

df["airtime"] = pd.to\_datetime(df["airtime"], **format** = '%H:%M')

**Convert the 'Time' column to 12-hour format**

df['airtime'] = df['airtime'].**dt.strftime**('%I:%M %p')