Chitramoy Mukherjee

DSC 540-T302

Week-7 and Week-8

Date : 10/19/2023

In [2]:
```
# Load necessary libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Chapter-7

Data source downloaded in .txt format and convert into .csv format.

In [35]:
```
# Task 01 : Convert Text to CSV
# Read MetObjects.csv file and create dataframe and select sample records using head
file_path = 'C:\\Users\\14024\\OneDrive\\Desktop\\MS-DSC\\DSC-540\\Week-8\\MetObjects.

# Read the text file and convert into CSV file
df1 = pd.read_csv(file_path)

# Store this dataframe in csv file
df1.to_csv('MetObjects.csv',index = None)
```

```
C:\Users\14024\AppData\Local\Temp\ipykernel_8108\3696584842.py:6: DtypeWarning: Colum
ns (5,7,10,11,12,13,14,34,35,36,37,38,39,40,41,42,43,44,45,46) have mixed types. Spec
ify dtype option on import or set low_memory=False.
  df1 = pd.read_csv(file_path)
```

In [37]:
```
# Task 02 : Read CSV data file
df = pd.read_csv("MetObjects.csv", low_memory=False)
print(df.shape)
cols = set(df.columns)
cols
```

```
(484956, 54)
```

Out[37]:
```
{'AccessionYear',
 'Artist Alpha Sort',
 'Artist Begin Date',
 'Artist Display Bio',
 'Artist Display Name',
 'Artist End Date',
 'Artist Gender',
 'Artist Nationality',
 'Artist Prefix',
 'Artist Role',
 'Artist Suffix',
 'Artist ULAN URL',
 'Artist Wikidata URL',
 'City',
 'Classification',
 'Constituent ID',
 'Country',
 'County',
 'Credit Line',
 'Culture',
 'Department',
 'Dimensions',
 'Dynasty',
 'Excavation',
 'Gallery Number',
 'Geography Type',
 'Is Highlight',
 'Is Public Domain',
 'Is Timeline Work',
 'Link Resource',
 'Locale',
 'Locus',
 'Medium',
 'Metadata Date',
 'Object Begin Date',
 'Object Date',
 'Object End Date',
 'Object ID',
 'Object Name',
 'Object Number',
 'Object Wikidata URL',
 'Period',
 'Portfolio',
 'Region',
 'Reign',
 'Repository',
 'Rights and Reproduction',
 'River',
 'State',
 'Subregion',
 'Tags',
 'Tags AAT URL',
 'Tags Wikidata URL',
 'Title'}
```

In [38]:
```
# Task 03 : Find out the Nulls/NaN

df.isna().sum()
```

Out[38]:
```
Object Number                   0
Is Highlight                    0
Is Timeline Work                0
Is Public Domain                0
Object ID                       0
Gallery Number             435415
Department                      0
AccessionYear                3862
Object Name                  2266
Title                       28664
Culture                    276766
Period                     393813
Dynasty                    461755
Reign                      473720
Portfolio                  458442
Constituent ID             202443
Artist Role                202443
Artist Prefix              202443
Artist Display Name        202443
Artist Display Bio         204533
Artist Suffix              202491
Artist Alpha Sort          202443
Artist Nationality         202443
Artist Begin Date          202443
Artist End Date            202443
Artist Gender              378474
Artist ULAN URL            257515
Artist Wikidata URL        260754
Object Date                 13431
Object Begin Date               0
Object End Date                 0
Medium                       7215
Dimensions                  75058
Credit Line                   651
Geography Type             424997
City                       452202
State                      482335
County                     476397
Country                    408949
Region                     453456
Subregion                  462813
Locale                     469217
Locus                      477438
Excavation                 468385
River                      482864
Classification              78717
Rights and Reproduction    460427
Link Resource                   0
Object Wikidata URL        415802
Metadata Date              484956
Repository                      0
Tags                       292501
Tags AAT URL               292501
Tags Wikidata URL          292501
dtype: int64
```

In [42]:
```
# Task 04 : Drop columns having all Null
df.dropna(axis=1, how="all", inplace=True)
print(f"column {cols.difference(set(df.columns))} having all Null values")
```

```
column {'Metadata Date'} having all Null values
```

In [43]:
```
# Task 05 : Fill in missing data with 'NA'

df_fill_missing = df.fillna('NA')
df_fill_missing
```

Out[43]:

	Object Number	Is Highlight	Is Timeline Work	Is Public Domain	Object ID	Gallery Number	Department	AccessionYear	Obje Nam
0	1979.486.1	False	False	False	1	NA	The American Wing	1979.0	Co
1	1980.264.5	False	False	False	2	NA	The American Wing	1980.0	Co
2	67.265.9	False	False	False	3	NA	The American Wing	1967.0	Co
3	67.265.10	False	False	False	4	NA	The American Wing	1967.0	Co
4	67.265.11	False	False	False	5	NA	The American Wing	1967.0	Co
...	...	...	...	...	...	...	...	...	
484951	55.621.134	False	False	False	900605	NA	Drawings and Prints	1955	Pri
484952	1977.646	False	False	False	900606	NA	Drawings and Prints	1977	Pri
484953	33.40.1	False	False	False	900633	NA	Drawings and Prints	1933	Pri
484954	170.1 C42	True	False	False	900717	NA	The Libraries	NA	N
484955	17.3.3457	False	False	False	900748	NA	Drawings and Prints	1917	Pri

484956 rows × 53 columns

In [55]:
```
# Task 06 : Check duplicates based on object number as key

# List Duplicates for column Object Number
df.duplicated(subset="Object Number").sum()

# List duplicates for object number and keep first value
df.duplicated(subset="Object Number", keep="first")

# Drop duplicates
df.drop_duplicates(subset="Object Number", keep="first", inplace=True)

# Get Duplicate counts
df.duplicated(subset="Object Number",keep="first").sum()

print(f"{df.duplicated().sum()} duplicates considering all columns")
```

```
0 duplicates considering all columns
```

Chapter - 8

In [56]:
```
# Task 01 : Create multiindex with AccessionYear and Department Columns

df.set_index(["AccessionYear", "Department"], inplace=True)
df.sort_index(inplace=True)
df.head()
```

Out[56]:

		Object Number	Is Highlight	Is Timeline Work	Is Public Domain	Object ID	Gallery Number	Object Name	
AccessionYear	**Department**								
1870.0	**Greek and Roman Art**	70.1	True	True	True	239584	169.0	Sarcophagus, garland	sa
1871.0	**European Paintings**	71.125	False	False	True	435655	NaN	Painting	
	European Paintings	71.156–57	False	False	True	435763	NaN	Painting	v
	European Paintings	71.110	False	False	True	435771	NaN	Painting	T
	European Paintings	71.19	False	False	True	435918	NaN	Painting	'

5 rows × 51 columns

In [60]:

```
# Task 02 : Group by with indexes

df.groupby(level=1).sum()
```

Out[60]:

	Is Highlight	Is Timeline Work	Is Public Domain	Object ID	Object Begin Date	Object End Date
Department						
Ancient Near Eastern Art	75	286	6190	2022922072	-7979004	-5766176
Arms and Armor	53	173	7079	801321000	21325915	23433651
Arts of Africa, Oceania, and the Americas	104	679	6370	4133923955	15138889	18527029
Asian Art	214	762	31295	3134972272	53468889	58087205
Costume Institute	134	272	8314	4494156015	60307318	60934548
Drawings and Prints	86	718	64793	93353716609	307040934	309565554
Egyptian Art	111	347	12192	15664315090	-44205316	-37536200
European Paintings	121	645	2271	1153390646	4389657	4423165
European Sculpture and Decorative Arts	82	670	33793	9688756570	74267201	76638954
Greek and Roman Art	114	535	29877	16178605951	-29507954	-11516380
Islamic Art	117	498	13013	7504585457	16883232	19999133
Medieval Art	47	346	6759	3370281564	6151143	7158247
Modern and Contemporary Art	178	367	180	7332454535	27696930	27758002
Musical Instruments	105	166	2280	2666243374	9015475	9353469
Photographs	113	382	6334	14442744063	71207384	71429206
Robert Lehman Collection	98	83	2272	1190809677	4257011	4370903
The American Wing	434	778	11808	1024517637	33858244	34235275
The Cloisters	57	186	2235	1095516820	3122995	3294467
The Libraries	512	0	147	428034825	1000561	987299

In [64]:
```
df.groupby(level=0).sum()
```

Out[64]:

	index	Is Highlight	Is Timeline Work	Is Public Domain	Object ID	Object Begin Date	Object End Date
0	0	1	1	1	239584	200	225
1	1	0	0	1	435655	1644	1644
2	2	0	0	1	435763	1520	1530
3	3	0	0	1	435771	1653	1653
4	4	0	0	1	435918	1662	1662
...	...	...	...	...	...	...	...
481651	481651	0	0	0	898528	0	0
481652	481652	0	0	0	898872	0	0
481653	481653	0	0	0	899651	0	0
481654	481654	0	0	0	900216	0	0
481655	481655	1	0	0	900717	1839	1839

481656 rows × 7 columns

In [65]:
```
df.reset_index(inplace=True)
```

In [67]:
```
# Task 03 : Reshape and Convert it into Dataframe and stack the result

sr1 = np.copy(df.Department.unique())
sr1 = sr1[:-1]
sr1
```

Out[67]:
```
array(['Greek and Roman Art', 'European Paintings',
       'European Sculpture and Decorative Arts', 'The American Wing',
       'Medieval Art', 'Egyptian Art', 'Ancient Near Eastern Art',
       'Islamic Art', 'Asian Art', 'Drawings and Prints',
       'Arts of Africa, Oceania, and the Americas', 'Costume Institute',
       'Arms and Armor', 'Photographs', 'Musical Instruments',
       'The Cloisters', 'Modern and Contemporary Art',
       'Robert Lehman Collection'], dtype=object)
```

In [69]:
```
sr1 = sr1.reshape(3,6)
sr1
df2 = pd.DataFrame(sr1)
df2
```

Out[69]:

	0	1	2	3	4	5
0	Greek and Roman Art	European Paintings	European Sculpture and Decorative Arts	The American Wing	Medieval Art	Egyptian Art
1	Ancient Near Eastern Art	Islamic Art	Asian Art	Drawings and Prints	Arts of Africa, Oceania, and the Americas	Costume Institute
2	Arms and Armor	Photographs	Musical Instruments	The Cloisters	Modern and Contemporary Art	Robert Lehman Collection

In [70]:
```
result1 = df2.stack()
result1
```

Out[70]:
```
0  0                      Greek and Roman Art
   1                       European Paintings
   2      European Sculpture and Decorative Arts
   3                        The American Wing
   4                             Medieval Art
   5                             Egyptian Art
1  0                 Ancient Near Eastern Art
   1                              Islamic Art
   2                                Asian Art
   3                      Drawings and Prints
   4   Arts of Africa, Oceania, and the Americas
   5                        Costume Institute
2  0                           Arms and Armor
   1                              Photographs
   2                      Musical Instruments
   3                            The Cloisters
   4              Modern and Contemporary Art
   5                 Robert Lehman Collection
dtype: object
```

In [71]:
```
result1.unstack()
```

Out[71]:

	0	1	2	3	4	5
0	Greek and Roman Art	European Paintings	European Sculpture and Decorative Arts	The American Wing	Medieval Art	Egyptian Art
1	Ancient Near Eastern Art	Islamic Art	Asian Art	Drawings and Prints	Arts of Africa, Oceania, and the Americas	Costume Institute
2	Arms and Armor	Photographs	Musical Instruments	The Cloisters	Modern and Contemporary Art	Robert Lehman Collection

In [74]:
```
# Task 04 : Split and Merge

# Split dataframe into 2 dataframe with 20 columns in first and rest in other datafram

df1 = df.iloc[:, :20]
df2 = df.iloc[:, 20:]
```

```
df2["Object Number"] = df1["Object Number"]
print(f"Shape of df : {df.shape}")
print(f"Shape of df1 : {df1.shape}")
print(f"Shape of df2 : {df2.shape}")
```

```
Shape of df : (481656, 55)
Shape of df1 : (481656, 20)
Shape of df2 : (481656, 36)
```

In [75]:
```
df1 = df1.merge(df2, how="inner")
print(f"Shape of df1 : {df1.shape}")
```

```
Shape of df1 : (481656, 55)
```

Chapter - 10

In [78]:
```
# Create subset with Department and Country
df1 = df[["Department",  "Country"]]
#Drop Null from Country column
df1.dropna(subset=["Country"], inplace=True)
# df1
# Replace new values with get-dummies method.
df1 = pd.get_dummies(df1, columns=["Country"], prefix="", prefix_sep="")
# Group data frame result based on department column.
df1 = df1.groupby("Department").sum()
df1
```

```
C:\Users\14024\AppData\Local\Temp\ipykernel_8108\1207400073.py:4: SettingWithCopyWarn
ing:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  df1.dropna(subset=["Country"], inplace=True)
```

Out[78]:

	Afghanistan	Afghanistan or Iran	Afghanistan or Northeastern Iran	Afghanistan or Turkmenistan	Africa	Alamania	Alamania|Fra
Department							
Ancient Near Eastern Art	0.0	0.0	0.0	0.0	0.0	0.0	
Arts of Africa, Oceania, and the Americas	0.0	0.0	0.0	0.0	0.0	0.0	
Egyptian Art	0.0	0.0	0.0	0.0	1.0	0.0	
Islamic Art	20.0	1.0	2.0	1.0	0.0	0.0	
Medieval Art	0.0	0.0	0.0	0.0	0.0	3.0	
Modern and Contemporary Art	0.0	0.0	0.0	0.0	0.0	0.0	
Musical Instruments	37.0	0.0	0.0	0.0	0.0	0.0	
The American Wing	0.0	0.0	0.0	0.0	0.0	0.0	
The Cloisters	0.0	0.0	0.0	0.0	0.0	0.0	
The Libraries	0.0	0.0	0.0	0.0	0.0	0.0	

10 rows × 959 columns

In [80]:

```
# Task 01 : Mapping

mapping = {
    "Mexico" : "GR1",
    "United States" : "GR1",
    "Canada" : "GR1",
    "England" : "GR2",
    "Spain" : "GR2",
    "Netharlands" : "GR2",
    "Italy" : "GR2",
    "Ireland" : "GR2",
    "France" : "GR3",
    "China" : "GR3",
    "India" : "GR3",
    "Japan" : "GR4",
    "Afghanistan" : "GR4",
    "Alamania" : "GR4",
}
by_column = df1.groupby(mapping, axis=1)
by_column.sum()
```

Out[80]:

Department	GR1	GR2	GR3	GR4
Ancient Near Eastern Art	0.0	0.0	0.0	0.0
Arts of Africa, Oceania, and the Americas	2084.0	0.0	22.0	0.0
Egyptian Art	0.0	1.0	0.0	0.0
Islamic Art	21.0	145.0	1164.0	20.0
Medieval Art	0.0	298.0	334.0	3.0
Modern and Contemporary Art	19.0	2.0	2.0	0.0
Musical Instruments	808.0	283.0	747.0	251.0
The American Wing	8116.0	1023.0	1158.0	21.0
The Cloisters	0.0	239.0	478.0	0.0
The Libraries	47.0	35.0	95.0	6.0

In [82]:
```
map_series = pd.Series(mapping)
df1.groupby(map_series, axis=1).sum()
```

Out[82]:

Department	GR1	GR2	GR3	GR4
Ancient Near Eastern Art	0.0	0.0	0.0	0.0
Arts of Africa, Oceania, and the Americas	2084.0	0.0	22.0	0.0
Egyptian Art	0.0	1.0	0.0	0.0
Islamic Art	21.0	145.0	1164.0	20.0
Medieval Art	0.0	298.0	334.0	3.0
Modern and Contemporary Art	19.0	2.0	2.0	0.0
Musical Instruments	808.0	283.0	747.0	251.0
The American Wing	8116.0	1023.0	1158.0	21.0
The Cloisters	0.0	239.0	478.0	0.0
The Libraries	47.0	35.0	95.0	6.0

In [83]:
```
df1.groupby(len).sum()
```

Out[83]:

	Afghanistan	Afghanistan or Iran	Afghanistan or Northeastern Iran	Afghanistan or Turkmenistan	Africa	Alamania	Alamania|Franc
Department							
11	20.0	1.0	2.0	1.0	0.0	0.0	0
12	0.0	0.0	0.0	0.0	1.0	3.0	1
13	0.0	0.0	0.0	0.0	0.0	0.0	0
17	0.0	0.0	0.0	0.0	0.0	0.0	0
19	37.0	0.0	0.0	0.0	0.0	0.0	0
24	0.0	0.0	0.0	0.0	0.0	0.0	0
27	0.0	0.0	0.0	0.0	0.0	0.0	0
41	0.0	0.0	0.0	0.0	0.0	0.0	0

8 rows × 959 columns

Chapter - 11

In [119…

```
# Task 01 : Convert between string and date time

df = pd.read_excel("BOING-BOING-CANDY-HIERARCHY-2016-SURVEY-Responses.xlsx")
df.head()
```

Out[119]:

	Timestamp	Are you going actually going trick or treating yourself?	Your gender:	How old are you?	Which country do you live in?	Which state, province, county do you live in?	[100 Grand Bar]	[Anonymous brown globs that come in black and orange wrappers]	[Any full-sized candy bar]	[Black Jacks]	..
0	2016-10-24 05:09:23.033	No	Male	22	Canada	Ontario	JOY	DESPAIR	JOY	MEH	..
1	2016-10-24 05:09:54.798	No	Male	45	usa	il	MEH	MEH	JOY	JOY	..
2	2016-10-24 05:13:06.734	No	Female	48	US	Colorado	JOY	DESPAIR	JOY	MEH	..
3	2016-10-24 05:14:17.192	No	Male	57	usa	il	JOY	MEH	JOY	MEH	..
4	2016-10-24 05:14:24.625	Yes	Male	42	USA	South Dakota	MEH	DESPAIR	JOY	DESPAIR	..

5 rows × 123 columns

In [120…

```
dtstr = df["Timestamp"].map(lambda dt: dt.strftime("%Y/%m/%d %H:%M:%S"))
print(dtstr.head())
```

```
0    2016/10/24 05:09:23
1    2016/10/24 05:09:54
2    2016/10/24 05:13:06
3    2016/10/24 05:14:17
4    2016/10/24 05:14:24
Name: Timestamp, dtype: object
```

In [101…

```
# Import module datetime
from datetime import datetime
dt = dtstr.map(lambda dtstr: datetime.strptime(dtstr, "%Y/%m/%d %H:%M:%S"))
print(dt.head())
```

```
0   2016-10-24 05:09:23
1   2016-10-24 05:09:54
2   2016-10-24 05:13:06
3   2016-10-24 05:14:17
4   2016-10-24 05:14:24
Name: Timestamp, dtype: datetime64[ns]
```

In [102…

```
# Generate date range
dt_range = pd.date_range(end='2020-07-06', periods = 200)
dt_range
```

Out[102]:

```
DatetimeIndex(['2019-12-20', '2019-12-21', '2019-12-22', '2019-12-23',
               '2019-12-24', '2019-12-25', '2019-12-26', '2019-12-27',
               '2019-12-28', '2019-12-29',
               ...
               '2020-06-27', '2020-06-28', '2020-06-29', '2020-06-30',
               '2020-07-01', '2020-07-02', '2020-07-03', '2020-07-04',
               '2020-07-05', '2020-07-06'],
              dtype='datetime64[ns]', length=200, freq='D')
```

In [121…

```
# Hourly range
dt_range = pd.date_range(end='2020-07-06 23:59:00', periods = 24, freq="1H")
dt_range
```

Out[121]:

```
DatetimeIndex(['2020-07-06 00:59:00', '2020-07-06 01:59:00',
               '2020-07-06 02:59:00', '2020-07-06 03:59:00',
               '2020-07-06 04:59:00', '2020-07-06 05:59:00',
               '2020-07-06 06:59:00', '2020-07-06 07:59:00',
               '2020-07-06 08:59:00', '2020-07-06 09:59:00',
               '2020-07-06 10:59:00', '2020-07-06 11:59:00',
               '2020-07-06 12:59:00', '2020-07-06 13:59:00',
               '2020-07-06 14:59:00', '2020-07-06 15:59:00',
               '2020-07-06 16:59:00', '2020-07-06 17:59:00',
               '2020-07-06 18:59:00', '2020-07-06 19:59:00',
               '2020-07-06 20:59:00', '2020-07-06 21:59:00',
               '2020-07-06 22:59:00', '2020-07-06 23:59:00'],
              dtype='datetime64[ns]', freq='H')
```

In [129…

```
# Convert timestamps to periods and back
# import parse
from dateutil.parser import parse
from pandas.tseries.offsets import Hour, Minute
df.set_index("Timestamp", inplace=True)
dt = parse("2016-10-24 05:09:23")
one_hour = Hour(1)
end_date = dt + one_hour
df[(df.index > dt) & (df.index < end_date)]
```

```
Out[129]:
```

Timestamp	Are you going actually going trick or treating yourself?	Your gender:	How old are you?	Which country do you live in?	Which state, province, county do you live in?	[100 Grand Bar]	[Anonymous brown globs that come in black and orange wrappers]	[Any full-sized candy bar]	[Black Jacks]	[B c
2016-10-24 05:09:23.033	No	Male	22	Canada	Ontario	JOY	DESPAIR	JOY	MEH	
2016-10-24 05:09:54.798	No	Male	45	usa	il	MEH	MEH	JOY	JOY	D
2016-10-24 05:13:06.734	No	Female	48	US	Colorado	JOY	DESPAIR	JOY	MEH	
2016-10-24 05:14:17.192	No	Male	57	usa	il	JOY	MEH	JOY	MEH	
2016-10-24 05:14:24.625	Yes	Male	42	USA	South Dakota	MEH	DESPAIR	JOY	DESPAIR	
...	...	...	...	...	...	...	...	...	...	
2016-10-24 06:06:09.860	No	Male	48	Canada	Nova Scotia	MEH	JOY	JOY	DESPAIR	
2016-10-24 06:07:43.121	No	Male	46	USA	Oklahoma, Rogers County	JOY	DESPAIR	JOY	DESPAIR	

2016-10-24 06:08:19.810	Yes	Male	43	USA	NC	JOY	DESPAIR	JOY	MEH
2016-10-24 06:08:43.022	No	Male	33	USA	New York	MEH	DESPAIR	JOY	MEH
2016-10-24 06:09:22.934	No	Male	48	usa	ga	JOY	DESPAIR	JOY	MEH

In [133…
```python
prd = df.index.to_period("H")
prd
```

Out[133]:
```
PeriodIndex(['2016-10-24 05:00', '2016-10-24 05:00', '2016-10-24 05:00',
             '2016-10-24 05:00', '2016-10-24 05:00', '2016-10-24 05:00',
             '2016-10-24 05:00', '2016-10-24 05:00', '2016-10-24 05:00',
             '2016-10-24 05:00',
             ...
             '2016-10-29 07:00', '2016-10-29 10:00', '2016-10-29 11:00',
             '2016-10-29 12:00', '2016-10-29 14:00', '2016-10-29 16:00',
             '2016-10-30 06:00', '2016-10-30 11:00', '2016-10-30 16:00',
             '2016-10-30 17:00'],
            dtype='period[H]', name='Timestamp', length=1259)
```

In [127…
```python
prd.to_timestamp(how='end')
```

Out[127]:
```
DatetimeIndex(['2016-10-24 05:59:59.999999999',
               '2016-10-24 05:59:59.999999999',
               '2016-10-24 05:59:59.999999999',
               '2016-10-24 05:59:59.999999999',
               '2016-10-24 05:59:59.999999999',
               '2016-10-24 05:59:59.999999999',
               '2016-10-24 05:59:59.999999999',
               '2016-10-24 05:59:59.999999999',
               '2016-10-24 05:59:59.999999999',
               '2016-10-24 05:59:59.999999999',
               ...
               '2016-10-29 07:59:59.999999999',
               '2016-10-29 10:59:59.999999999',
               '2016-10-29 11:59:59.999999999',
               '2016-10-29 12:59:59.999999999',
               '2016-10-29 14:59:59.999999999',
               '2016-10-29 16:59:59.999999999',
               '2016-10-30 06:59:59.999999999',
               '2016-10-30 11:59:59.999999999',
               '2016-10-30 16:59:59.999999999',
               '2016-10-30 17:59:59.999999999'],
              dtype='datetime64[ns]', name='Timestamp', length=1259, freq=None)
```

In [126…
```
df.reset_index(inplace=True)
```