Metropolitan Museum of Art: Artifacts Analysis

by: Alissa Trujillo

In [1]:
```
import numpy as np
import pandas as pd
from datetime import datetime
```

In [2]:
```
met_df = pd.read_csv("MetObjects.csv")
```

```
<ipython-input-2-d08dce636664>:1: DtypeWarning: Columns (5,7,10,11,12,13,14,3
4,35,36,37,38,39,40,41,42,43,44,45,46) have mixed types. Specify dtype option
on import or set low_memory=False.
  met_df = pd.read_csv("MetObjects.csv")
```

Transformation 1: Filter Out Missing Data

In [3]:
```
met_df.shape
```

Out[3]:
```
(477804, 54)
```

In [4]:
```
met_df = met_df.dropna(axis=1, how='all')
```

In [5]:
```
met_df.shape
```

Out[5]:
```
(477804, 53)
```

By dropping the columns that contain only NA's, we are able to eliminate one completely empty column from our dataset. The shape of the original dataset shows 54 columns, and after dropping essentially empty columns, the new shape of our dataset shows 53 columns.

Transformation 2: Manipulating Strings

In [6]:
```
pd.unique(met_df['Object Name'])
```

Out[6]:
```
array(['Coin', 'Peso', 'Centavos', ..., 'Book, print, ephemera',
       'Book; prints', 'Ephemera; postcard'], dtype=object)
```

We can see that the categories in this column are not consistenty capitalized and some of the items have multiple categories separated by different delimiters. We can fix the capitalization by applying the str.lower() function to this column, and then we can split the Object Name variables using the str.split() function.

In [7]:
```
met_df['Object Name'] = met_df['Object Name'].str.lower()
met_df['Object Name'] = met_df['Object Name'].str.split(":,")
```

In [8]:
```
met_df['Object Name'][477762:477768]
```

Out[8]:
```
477762           [drawing]
477763             [print]
477764             [print]
477765              [book]
477766              [book]
477767    [book, pamphlet]
Name: Object Name, dtype: object
```

Now we can see that the object types are contained in a list with consistent capitalization.

In [9]:
```
met_df['Primary Type'] = met_df['Object Name'].str[0]
met_df['Primary Type']
```

Out[9]:
```
0            coin
1            coin
2            coin
3            coin
4            coin
           ...
477799    drawing
477800    drawing
477801      print
477802    drawing
477803    drawing
Name: Primary Type, Length: 477804, dtype: object
```

I have also added a new column that denotes the main object type for each item in case we have transformations down the road that do not like the list type that 'Object Name' now has.

Transformation 3: Creating a Hierarchical Index

This dataset includes a unique object number for each item that can be used as an identifier. I will be creating a hierarchical index by first splitting the objects by department, using the department name as the outer index, and then using the object number as the inner index.

In [10]:
```
met_df2 = met_df.set_index(['Department', 'Object Number'])
met_df2.head()
```

Out[10]:

Department	Object Number	Is Highlight	Is Timeline Work	Is Public Domain	Object ID	Gallery Number	AccessionYear	Object Name
The American Wing	1979.486.1	False	False	False	1	NaN	1979.0	[coin]
	1980.264.5	False	False	False	2	NaN	1980.0	[coin]
	67.265.9	False	False	False	3	NaN	1967.0	[coin]
	67.265.10	False	False	False	4	NaN	1967.0	[coin]
	67.265.11	False	False	False	5	NaN	1967.0	[coin]

5 rows × 52 columns

Transformation 4: Pivoting Data

I am going to create a pivot table that takes a look at how many works in each department and gallery are public domain.

In [11]:
```
met_df3 = met_df.pivot_table(index='Department', columns='Gallery Number',
                             values='Is Public Domain', aggfunc='sum',
                             fill_value=0)
met_df3
```

Out[11]:

Gallery Number	2.0	7.0	10.0	14.0	16.0	19.0	100.0	101.0	103.0	104.0	...	959	961	962
Department														
Ancient Near Eastern Art	0	0	0	0	0	0	0	0	0	0	...	0	0	0
Arms and Armor	1	0	2	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	11	0	...	0	0	0
Asian Art	0	0	0	0	0	0	0	0	0	0	...	0	0	0
Costume Institute	0	0	0	0	0	0	0	0	0	0	...	0	0	0
Drawings and Prints	0	0	1	0	0	0	0	0	0	0	...	0	0	0
Egyptian Art	0	0	0	0	0	0	12	144	121	74	...	0	0	0
European Paintings	0	0	0	0	1	0	0	0	0	0	...	0	0	0
European Sculpture and Decorative Arts	0	0	0	0	0	0	0	0	0	0	...	0	0	0
Greek and Roman Art	0	0	0	0	0	0	0	0	0	0	...	0	0	0
Islamic Art	1	0	2	0	0	0	0	0	0	0	...	0	0	0
Medieval Art	0	0	0	0	0	0	0	0	0	0	...	0	0	0
Modern and Contemporary Art	0	0	0	0	0	0	0	0	0	0	...	0	0	0
Musical Instruments	0	0	0	0	0	0	0	0	0	0	...	0	0	0
Photographs	0	0	0	0	0	0	0	0	0	0	...	0	0	0
Robert Lehman Collection	0	0	0	0	0	0	0	0	0	0	...	0	7	2
The American Wing	0	0	0	0	0	0	0	0	0	0	...	0	0	0
The Cloisters	1	1	12	5	1	1	0	0	0	0	...	0	0	0
The Libraries	0	0	0	0	0	0	0	0	0	0	...	0	0	0

19 rows × 609 columns

The department is indexed on the left hand side and the gallery number is denoted by the column name at the top. The aggfunc sum function allows us to look at the number of "true" values in each column, denoting whether the work is in the public domain. The fill value fills in the NA values with zeroes, defaulting to false. We can see by looking at this data that the Egyptian Art department has many public domain artifacts spread across multiple departments. We can also see that Gallery 10 contains public domain works from 4 different departments.

Transformation 5: Grouping Within a Series

In [12]:
```
met_df['AccessionYear'] = pd.to_numeric(met_df['AccessionYear'], errors='coerc
met_df['AccessionYear'] = met_df['AccessionYear'].astype('Int64')
grouped = met_df['AccessionYear'].groupby(met_df['Department'], dropna=True)
grouped.mean()
```

Out[12]:
```
Department
Ancient Near Eastern Art                     1948.864182
Arms and Armor                               1933.492283
Arts of Africa, Oceania, and the Americas    1979.635719
Asian Art                                    1945.727474
Costume Institute                            1983.608722
Drawings and Prints                          2071.813985
Egyptian Art                                 1922.552621
European Paintings                            1951.71303
European Sculpture and Decorative Arts       1936.888021
Greek and Roman Art                          1958.603716
Islamic Art                                  1937.850813
Medieval Art                                 1928.250213
Modern and Contemporary Art                  1980.139716
Musical Instruments                          1925.859046
Photographs                                   1989.18348
Robert Lehman Collection                     1975.012374
The American Wing                            1955.900548
The Cloisters                                1951.875922
The Libraries                                     2009.0
Name: AccessionYear, dtype: Float64
```

Here I used the Groupby funtion to see the average Accession Year for the articles in each department. The Egyptian Art has the oldest average accession year with a mean of 1922. The average Accession Year for Drawings and Prints came out to be 2071, which is a year in the future, indicating that there are erroneous data points in this category.

In [13]:
```
met_df = met_df.replace(19171917, 1917)
```

After doing some investigation, the date that was throwing off the averages was a value of '19171917' which we can assume was intended to be 1917. That value is now fixed so it fits the normal range of data.

Transformation 6: Grouping with Functions

In [14]:
```
grouped2 = met_df['Is Highlight'].groupby(met_df['Primary Type']).count()
grouped2.sort_values(ascending=False)
```

Out[14]:
```
Primary Type
print                              99527
photograph                         28458
drawing                            25791
book                               13394
fragment                            9568
                                   ...
hexagonal vessel with cover            1
hexagonal box with inverted corners    1
herm of hermes propylaios              1
herm of hermes                         1
ṭāśā                                   1
Name: Is Highlight, Length: 27601, dtype: int64
```

I used two functions in this Groupby transformation: **count()** and **sort_values()**.

For this transformation, I grouped the data by the Primary Type column we created earlier. I focused on the 'Is Highlight' variable, indicating whether the object is a highlighted item in the museum. Then I applied the count() function in order to count the number of true values there were for each object type. Lastly, I applied sort_values() to sort this information in descending order.

The result is a series that shows the Primary Types that have the most highlighted items. We can see that the most popular types are print, photograph, drawing, book and fragment. These are all popular mediums, as opposed to the items at the bottom of the list which are more specific or obscure.

In [15]:
```
grouped2.agg(['mean', 'max', 'min', 'std'])
```

Out[15]:
```
mean       17.249846
max     99527.000000
min         1.000000
std       660.787170
Name: Is Highlight, dtype: float64
```

We can also pass function arguments to the 'agg' like above to get some descriptive statistics about the resulting data.

Transformation 7: Split/Apply/Combine

Using the apply method, we can split the data by department, apply the "newest" function defined below, and then combine them into one table to see the newest pieces in each department.

In [16]:
```
def newest(met_df, n=5, column='AccessionYear'):
    return met_df.sort_values(by=column, na_position='first')[-n:]
```

In [17]:
```
met_df.groupby('Department').apply(newest)
```

Out[17]:

		Object Number	Is Highlight	Is Timeline Work	Is Public Domain	Object ID	Gallery Number	Departmer
Department								
Ancient Near Eastern Art	**202072**	2012.454	False	True	True	328241	NaN	Ancier Nea Eastern Ar
	202228	2014.717	False	False	True	329087	403.0	Ancier Nea Eastern Ar
	202071	2015.789	False	False	True	328186	403.0	Ancier Nea Eastern Ar
	465563	2019.288.1–.32	False	False	False	820847	NaN	Ancier Nea Eastern Ar
	456716	2019.116	False	False	True	781871	NaN	Ancier Nea Eastern Ar
...	**...**	...	...	...	...	...	...	.
The Libraries	**396082**	DC611.N846 A5 1788	True	False	True	680318	NaN	Th Librarie
	395630	AY831 .Z7 1792	True	False	True	679732	NaN	Th Librarie
	395564	AY834 .A46 1813	True	False	False	679638	NaN	Th Librarie
	396522	BX2016.A5 F7 1792	True	False	True	681246	NaN	Th Librarie
	472993	HG1552.D28 A3 2007a	True	False	False	839191	NaN	Th Librarie

95 rows × 54 columns

Transformation 8: Cross-Tabulations

I performed a cross-tabulation of the data by taking a look at the number of objects of each type in each department. The margins=True argument includes a column on the right that gives us a total count for all of the items in the department as well as a row at the bottom that gives the total counts for each object type.

In [18]:
```
pd.crosstab(met_df['Department'], met_df['Primary Type'], margins=True)
```

Out[18]:

Primary Type	"autophone" organette	"basso"	"chanot model" violin	"humantone" nose flute	"japanese fiddle"	"komo" trumpet	"ladies in blue" fresco	"mu ("r
Department								
Ancient Near Eastern Art	0	0	0	0	0	0	0	
Arms and Armor	0	0	0	0	0	0	0	
Arts of Africa, Oceania, and the Americas	0	0	0	0	0	0	0	
Asian Art	0	0	0	0	0	0	0	
Costume Institute	0	0	0	0	0	0	0	
Drawings and Prints	0	0	0	0	0	0	0	
Egyptian Art	0	0	0	0	0	0	0	
European Paintings	0	0	0	0	0	0	0	
European Sculpture and Decorative Arts	0	0	0	0	0	0	0	
Greek and Roman Art	0	0	0	0	0	0	1	
Islamic Art	0	0	0	0	0	0	0	
Medieval Art	0	0	0	0	0	0	0	
Modern and Contemporary Art	0	0	0	0	0	0	0	
Musical Instruments	1	1	1	1	1	1	0	
Photographs	0	0	0	0	0	0	0	
Robert Lehman Collection	0	0	0	0	0	0	0	
The American Wing	0	0	0	0	0	0	0	
The Cloisters	0	0	0	0	0	0	0	
All	1	1	1	1	1	1	1	

19 rows × 27602 columns

Transformation 9: Convert Between String and Datetime

In [19]:
```
met_df['AccessionYearDT2'] = pd.to_datetime(met_df['AccessionYear'], format='%
```

In [20]:
```
met_df['AccessionYearDT2']
```

Out[20]:
```
0         1979-01-01
1         1980-01-01
2         1967-01-01
3         1967-01-01
4         1967-01-01
             ...
477799    1923-01-01
477800    1923-01-01
477801    1953-01-01
477802    1923-01-01
477803    1923-01-01
Name: AccessionYearDT2, Length: 477804, dtype: datetime64[ns]
```

Now there is a new column in the data that expresses the Accession Year in datetime format.

Transformation 10: Period Frequency Conversions

Since the objects are denoted by Accession Year, the periods I will be taking a look at will be decades. I will first find out what the earliest included decade in the museum.

In [21]:
```
met_df['AccessionYear'].min()
```

Out[21]:
```
1870
```

Now I will compute a datetime object with the decades from 1870 to now.

In [22]:
```
p = pd.date_range('1870-01-01', '2020-01-01', freq='10AS-JAN')
p
```

Out[22]:
```
DatetimeIndex(['1870-01-01', '1880-01-01', '1890-01-01', '1900-01-01',
               '1910-01-01', '1920-01-01', '1930-01-01', '1940-01-01',
               '1950-01-01', '1960-01-01', '1970-01-01', '1980-01-01',
               '1990-01-01', '2000-01-01', '2010-01-01', '2020-01-01'],
              dtype='datetime64[ns]', freq='10AS-JAN')
```

I can now use the resample metric to find out statistics about the artifacts within each individual decade.

In [23]:
```
met_df4 = met_df.set_index('AccessionYearDT2')
met_df4.resample('10AS-JAN').mean()
```

Out[23]:

AccessionYearDT2	Is Highlight	Is Timeline Work	Is Public Domain	Object ID	AccessionYear	Object Begin Date
1870-01-01	0.001484	0.013353	0.880415	226706.375668	1874.817359	-993.022700
1880-01-01	0.006466	0.015542	0.722632	358641.563698	1885.798866	1238.652411
1890-01-01	0.001980	0.014176	0.902856	214006.296435	1893.729518	911.103919
1900-01-01	0.003249	0.014223	0.591328	274060.015714	1906.807117	867.918553
1910-01-01	0.004825	0.020500	0.657060	350147.430311	1914.62944	588.640134
1920-01-01	0.003173	0.015802	0.612731	390862.233128	1924.345209	896.230679
1930-01-01	0.003319	0.014702	0.597279	311328.143512	1934.300052	1185.405821
1940-01-01	0.002933	0.012965	0.449098	358367.710510	1944.657709	1520.605615
1950-01-01	0.003162	0.012988	0.471163	437882.982932	1954.396133	1624.481954
1960-01-01	0.001769	0.007418	0.444015	507788.687528	1963.564216	1752.628026
1970-01-01	0.008268	0.027059	0.410451	327947.962328	1975.079042	1647.317188
1980-01-01	0.007065	0.031049	0.353103	305912.118618	1984.302198	1558.724724
1990-01-01	0.009217	0.037239	0.272922	291432.726115	1994.522784	1677.514085
2000-01-01	0.005831	0.025525	0.414339	256887.199390	2005.568508	1708.749831
2010-01-01	0.006579	0.005817	0.634202	624367.521364	2013.256135	820.223137
2020-01-01	0.006013	0.002830	0.253626	715337.164839	2020.570923	1597.043509

Since some of the columns are boolean variables, the range from 0-1 gives us a percentage of the artifacts that are museum highlights, timeline works, and in the public domain.