museum_art_curation Projects ▾ Branches ▾ Notebook Dataset ▾ Caveats ▾ Settings ▾

On branch **default**

[1]
```
LOAD DATASET MetObjects AS csv FROM MetObjects.csv @ url
'https://media.githubusercontent.com/media/metmuseum/openaccess/master/MetObjects.csv'
```

Console ▾ Timing Datasets ▾ Charts ▾

metobjects (**612686** rows)

Views

	Object_Number (string)	Is_Highlight (boolean)	Is_Timeline_Work (boolean)	Is_Public_Domain (boolean)	Object_ID (int)	Gallery_Number (short)	Department (string)	Acc
0	1979.486.1	false	false	false	1		The American Wing	197
1	1980.264.5	false	false	false	2		The American Wing	198
2	67.265.9	false	false	false	3		The American Wing	196
3	67.265.10	false	false	false	4		The American Wing	196
4	67.265.11	false	false	false	5		The American Wing	196
5	67.265.12	false	false	false	6		The American Wing	196
6	67.265.13	false	false	false	7		The American Wing	196
7	67.265.14	false	false	false	8		The American Wing	196
8	67.265.15	false	false	false	9		The American Wing	196
9	1979.486.3	false	false	false	10		The American Wing	197
10	1979.486.2	false	false	false	11		The American Wing	197
11	1979.486.7	false	false	false	12		The American Wing	197
12	1979.486.4	false	false	false	13		The American Wing	197
13	1979.486.5	false	false	false	14		The American Wing	197
14	16.74.49	false	false	false	15		The American Wing	191
15	16.74.27	false	false	false	16		The American Wing	191
16	16.74.28	false	false	false	17		The American Wing	191
17	16.74.29	false	false	false	18		The American Wing	191
18	16.74.30	false	false	false	19		The American Wing	191
19	16.74.31	false	false	false	20		The American Wing	191

[2]
```
SELECT * FROM metobjects WHERE Object_ID is not null;
```

Console ▾ Timing Datasets ▾ Charts ▾

metobjects (**484967** rows)

Views

	Object_Number (string)	Is_Highlight (boolean)	Is_Timeline_Work (boolean)	Is_Public_Domain (boolean)	Object_ID (int)	Gallery_Number (short)	Department (string)	Acc
0	1979.486.1	false	false	false	1		The American Wing	197
1	1980.264.5	false	false	false	2		The American Wing	198
2	67.265.9	false	false	false	3		The American Wing	196
3	67.265.10	false	false	false	4		The American Wing	196
4	67.265.11	false	false	false	5		The American Wing	196
5	67.265.12	false	false	false	6		The American Wing	196
6	67.265.13	false	false	false	7		The American Wing	196
7	67.265.14	false	false	false	8		The American Wing	196
8	67.265.15	false	false	false	9		The American Wing	196
9	1979.486.3	false	false	false	10		The American Wing	197
10	1979.486.2	false	false	false	11		The American Wing	197
11	1979.486.7	false	false	false	12		The American Wing	197
12	1979.486.4	false	false	false	13		The American Wing	197

museum_art_curation Projects ▾ Branches ▾ Notebook Dataset ▾ Caveats ▾ Settings ▾

14	16.74.49	false	false	false	15	The American Wing	191
15	16.74.27	false	false	false	16	The American Wing	191
16	16.74.28	false	false	false	17	The American Wing	191
17	16.74.29	false	false	false	18	The American Wing	191
18	16.74.30	false	false	false	19	The American Wing	191
19	16.74.31	false	false	false	20	The American Wing	191

```
[3] LOAD DATASET cmoa AS csv FROM cmoa.csv @ url 'https://raw.githubusercontent.com/cmoa/collection/master/cmoa.csv'
```

Console ▾ Timing Datasets ▾ Charts ▾

cmoa (**34595** rows)

Views

	title (string)	creation_date (string)	creation_date_earliest (date)	creation_date_latest (date)
0	Keith Haring	1984	1984-00-01	1984-00-01
1	Untitled	1964-1965	1964-00-01	1965-00-01
2				
3	TBF 10/30/14"	paintings	?	?
4	Trans East West (Tew) No. 5: The Attack of the Embassy (Reconstruction)	1999	1999-00-01	1999-00-01
5	Trans East West (Tew) No. 6: Rebuilding Beyrut	1999	1999-00-01	1999-00-01
6	Trans East West (Tew) No. 7: Bombing the Power Plant	1999	1999-00-01	1999-00-01
7	Trans East West (Tew) No. 8: Suddenly It was Completely Dark (Maybe Because I Closed My Eyes)	1999	1999-00-01	1999-00-01
8	"Trans East West (Tew) No. 9: The ""Meridien"" Will Have a Nice View"	1999	1999-00-01	1999-00-01
9	Trans East West (Tew) No. 10: To the People of Damascus	1999	1999-00-01	1999-00-01
10	Trans East West (Tew) No. 11: Former Restaurant	1999	1999-00-01	1999-00-01
11	Trans East West (Tew) No. 12: The Neighbours	1999	1999-00-01	1999-00-01
12	Trans East West (Tew) No. 13: The President	1999	1999-00-01	1999-00-01
13	Trans East West (Tew) No. 14: To the People of Damascus	1999	1999-00-01	1999-00-01
14	Trans East West (Tew) No. 15: Cinema Beyrut	1999	1999-00-01	1999-00-01
15	Trans East West (Tew) No. 16: Secret Excavations in Pigman's Land	1999	1999-00-01	1999-00-01
16	Snuff bottle: Carved Glass	late 18th century	1770-00-01	1799-00-01
17	Pill Lamp	1968	1968-00-01	1968-00-01
18	Pill Lamp	1968	1968-00-01	1968-00-01
19	Pill Lamp	1968	1968-00-01	1968-00-01

```
[4] LOAD DATASET moma AS csv FROM Artworks.csv @ url
    'https://media.githubusercontent.com/media/MuseumofModernArt/collection/master/Artworks.csv'
```

Console ▾ Timing Datasets ▾ Charts ▾

moma (**153825** rows)

Views

	Title (string)	Artist (string)	ConstituentID (float)	ArtistBio (str
0	Ferdinandsbrücke Project, Vienna, Austria (Elevation, preliminary version)	Otto Wagner	6210	(Austrian, 1841–1918)
1	City of Music, National Superior Conservatory of Music and Dance, Paris, France, View from interior courtyard	Christian de Portzamparc	7470	(French, born 1944)
2	Villa near Vienna Project, Outside Vienna, Austria, Elevation	Emil Hoppe	7605	(Austrian, 1876–1957)
3	The Manhattan Transcripts Project, New York, New York, Introductory panel to Episode 1: The Park	Bernard Tschumi	7056	(French and Swiss, born S
4	Villa, project, outside Vienna, Austria, Exterior perspective	Emil Hoppe	7605	(Austrian, 1876–1957)
5	The Manhattan Transcripts Project, New York, New York, Episode 1: The Park	Bernard Tschumi	7056	(French and Swiss, born S
6	The Manhattan Transcripts Project, New York, New York, Episode 1: The Park	Bernard Tschumi	7056	(French and Swiss, born S
7	The Manhattan Transcripts Project, New York, New York, Episode 1: The Park	Bernard Tschumi	7056	(French and Swiss, born S
8	The Manhattan Transcripts Project, New York, New York, Episode 1: The Park	Bernard Tschumi	7056	(French and Swiss, born S

10	The Manhattan Transcripts Project, New York, New York, Episode 1: The Park	Bernard Tschumi	7056	(French and Swiss, born S
11	The Manhattan Transcripts Project, New York, New York, Episode 1: The Park	Bernard Tschumi	7056	(French and Swiss, born S
12	The Manhattan Transcripts Project, New York, New York, Episode 1: The Park	Bernard Tschumi	7056	(French and Swiss, born S
13	The Manhattan Transcripts Project, New York, New York, Episode 1: The Park	Bernard Tschumi	7056	(French and Swiss, born S
14	The Manhattan Transcripts Project, New York, New York, Episode 1: The Park	Bernard Tschumi	7056	(French and Swiss, born S
15	The Manhattan Transcripts Project, New York, New York, Episode 1: The Park	Bernard Tschumi	7056	(French and Swiss, born S
16	The Manhattan Transcripts Project, New York, New York, Episode 1: The Park	Bernard Tschumi	7056	(French and Swiss, born S
17	The Manhattan Transcripts Project, New York, New York, Episode 1: The Park	Bernard Tschumi	7056	(French and Swiss, born S
18	The Manhattan Transcripts Project, New York, New York, Episode 1: The Park	Bernard Tschumi	7056	(French and Swiss, born S
19	The Manhattan Transcripts Project, New York, New York , Episode 1: The Park	Bernard Tschumi	7056	(French and Swiss, born S

[5]

```
LOAD DATASET tate AS csv FROM artwork_data.csv @ url
'https://raw.githubusercontent.com/tategallery/collection/master/artwork_data.csv'
```

Console Timing Datasets Charts

tate (**69201** rows)

Views

	id (int)	accession_number (string)	artist (string)	artistRole (string)	artistId (short)	title (string)
0	1035	A00001	Blake, Robert	artist	38	A Figure Bowing before a Seated Old Man with his Arm Outstretched in Benediction. Verso: Ir
1	1036	A00002	Blake, Robert	artist	38	Two Drawings of Frightened Figures, Probably for ‘The Approach of Doom’
2	1037	A00003	Blake, Robert	artist	38	The Preaching of Warning. Verso: An Old Man Enthroned Between Two Groups of Figures, by
3	1038	A00004	Blake, Robert	artist	38	Six Drawings of Figures with Outstretched Arms
4	1039	A00005	Blake, William	artist	39	The Circle of the Lustful: Francesca da Rimini (‘The Whirlwind of Lovers’)
5	1040	A00006	Blake, William	artist	39	Ciampolo the Barrator Tormented by the Devils
6	1041	A00007	Blake, William	artist	39	The Baffled Devils Fighting
7	1042	A00008	Blake, William	artist	39	The Six-Footed Serpent Attacking Agnolo Brunelleschi
8	1043	A00009	Blake, William	artist	39	The Serpent Attacking Buoso Donati
9	1044	A00010	Blake, William	artist	39	The Pit of Disease: The Falsifiers
10	1045	A00011	Blake, William	artist	39	Dante Striking against Bocca Degli Abati
11	1046	A00012	Blake, William	artist	39	Job and his Family
12	1047	A00013	Blake, William	artist	39	Satan before the Throne of God
13	1048	A00014	Blake, William	artist	39	Job’s Sons and Daughters Overwhelmed by Satan
14	1049	A00015	Blake, William	artist	39	The Messengers tell Job of his Misfortunes
15	1050	A00016	Blake, William	artist	39	Satan Going Forth from the Presence of the Lord, and Job’s Charity
16	1051	A00017	Blake, William	artist	39	Satan Smiting Job with Sore Boils
17	1052	A00018	Blake, William	artist	39	Job’s Comforters
18	1053	A00019	Blake, William	artist	39	Job’s Despair
19	1054	A00020	Blake, William	artist	39	The Vision of Eliphaz

[6]

```python
import pandas as pd
import numpy as np
df_tate = vizierdb.get_data_frame('tate');
df_moma = vizierdb.get_data_frame('moma');
df_meta = vizierdb.get_data_frame('metobjects');
df_cmoa = vizierdb.get_data_frame('cmoa');
data = df_meta;
data.columns = [x.replace('_', ' ') for x in data.columns]
# Find the oldest objects based on 'Object Begin Date' and 'Object End Date'
oldest_object = data.loc[data['Object Begin Date'].idxmin()]

# Get the 'Object Date', 'Object Begin Date', and 'Object End Date' of the oldest object
oldest_object_date = oldest_object['Object Date']
oldest_object_begin_date = oldest_object['Object Begin Date']
```

```
# Print the oldest object information and date range
print("Oldest Object Date:", oldest_object_date)
print("Oldest Object Begin Date:", oldest_object_begin_date)
print("Oldest Object End Date:", oldest_object_end_date)

# Initialize a set to store unique non-anonymous artist names
unique_non_anonymous_artist_names = set()

# Iterate through the 'Artist Display Name' column and add unique non-anonymous names to the set
for artists in data['Artist Display Name']:
    if pd.notnull(artists):
        # Split the string into individual artist names using the '|' delimiter
        artist_names = artists.split('|')
        # Check if the artist name is not anonymous and not unknown, then add it to the set
        non_anonymous_names = [artist.strip() for artist in artist_names if artist.strip() and not artist.strip().startswi
        unique_non_anonymous_artist_names.update(non_anonymous_names)

# Get the number of unique non-anonymous artist names
num_unique_non_anonymous_artists = len(unique_non_anonymous_artist_names)

# Print the number of unique non-anonymous artist names
print("Number of unique non-anonymous artist names:", num_unique_non_anonymous_artists)


from collections import Counter

# Function to clean and extract artist names
def extract_artist_names(artists):
    if pd.notnull(artists):
        # Split the string into individual artist names using the '|' delimiter
        artist_names = artists.split('|')
        # Clean and return the artist names (removing empty strings, whitespaces, and anonymous names)
        return [artist.strip() for artist in artist_names if artist.strip() and not artist.strip().startswith(('Anonymous'
    else:
        return []

# Extract artist names using the function and flatten the list
all_artist_names = [name for sublist in data['Artist Display Name'].apply(extract_artist_names) for name in sublist]

# Count the occurrences of each artist name
artist_name_counts = Counter(all_artist_names)

# Find the most frequent artist name
most_frequent_artist_name = artist_name_counts.most_common(1)[0]

# Print the most frequent artist name and its count
print("Most Frequent Artist Name:", most_frequent_artist_name[0])
print("Count:", most_frequent_artist_name[1])

# Initialize variables to store oldest object information
oldest_accession_year = float('inf')  # Set to positive infinity initially
oldest_object_titles = []

# Iterate through the data to find the oldest object(s)
for index, row in data.iterrows():
    accession_year = row['AccessionYear']
    try:
        # Try converting the accession year to an integer (ignore non-four-digit entries)
        accession_year = int(accession_year)
        # Check if the accession year is a valid four-digit number
        if 1000 <= accession_year <= 9999:
            # Compare accession years to find the oldest object(s)
            if accession_year < oldest_accession_year:
                oldest_accession_year = accession_year
                oldest_object_titles = [row['Title']]
            elif accession_year == oldest_accession_year:
                oldest_object_titles.append(row['Title'])
    except ValueError:
        # Handle invalid accession years (non-integer values) if any
        pass
```

```
print("Oldest Object(s) Title(s):")
for title in oldest_object_titles:
    print(title)


# Check the number of unique values in the original "Medium" column
original_unique_values = data['Medium'].nunique()

# Clean the "Medium" column by converting to uppercase and stripping whitespaces
data = data.assign(Medium=data['Medium'].str.upper().str.strip())

# Check the number of unique values in the cleaned "Medium" column
cleaned_unique_values = data['Medium'].nunique()

# Print the number of unique values before and after the change
print(f"Number of unique values in 'Medium' before cleaning: {original_unique_values}")
print(f"Number of unique values in 'Medium' after cleaning: {cleaned_unique_values}")

# Now the "Medium" column in the DataFrame is cleaned and standardized
# You can continue working with the updated DataFrame

# Create a boolean series for problematic cases (end date before begin date)
problematic_cases = data['Object End Date'] < data['Object Begin Date']

# Extract end dates from "Object Date" column where there is a dash and "B.C." ending
regex_pattern = r'(\d{1,4})\s*-\s*B\.C\.'
data.loc[problematic_cases, 'Object End Date'] = data.loc[problematic_cases, 'Object Date'].str.extract(regex_pattern, exp

# Convert the "Object End Date" column to numeric, handling errors and replacing 0 values with NaN
data['Object End Date'] = pd.to_numeric(data['Object End Date'], errors='coerce').replace(0, np.nan)

# Fill remaining problematic cases where end date is before begin date with NaN values
data.loc[problematic_cases, 'Object End Date'] = np.nan


# Split the "Tags" column and explode the entries to new rows
tags_df = data.assign(Tags=data['Tags'].str.split(',')).explode('Tags')

data = tags_df;

# Initialize empty lists for each artist-related column
artist_roles = []
artist_display_names = []
artist_begin_dates = []
artist_end_dates = []
artist_gender = []
artist_nationality = []
object_ids = []

# Iterate over the rows and columns to extract artist information
for index, row in data.iterrows():
    object_id = row['Object ID']

    # Split the columns into lists
    roles = str(row['Artist Role']).split('|')
    display_names = str(row['Artist Display Name']).split('|')
    begin_dates = str(row['Artist Begin Date']).split('|')
    end_dates = str(row['Artist End Date']).split('|')
    gender = str(row['Artist Gender']).split('|')
    nationality = str(row['Artist Nationality']).split('|')

    # Append the corresponding elements to the lists
    for role, display_name, begin_date, end_date, gender, nationality in zip(roles, display_names, begin_dates, end_dates,
        artist_roles.append(role.strip())
        artist_display_names.append(display_name.strip())
        artist_begin_dates.append(begin_date.strip())
        artist_end_dates.append(end_date.strip())
        artist_gender.append(gender.strip())
        artist_nationality.append(nationality.strip())
        object_ids.append(object_id)
    'Object ID': object_ids,
    'Artist Role': artist_roles,
    'Artist Display Name': artist_display_names,
    'Artist Begin Date': artist_begin_dates,
    'Artist End Date': artist_end_dates,
    'Artist Gender': artist_gender,
    'Artist Nationality': artist_nationality
})

# Print the new DataFrame with artist information
final_data = artists_df
# Additional mapping between nationalities and countries
# Extended mapping between nationalities and countries
nationality_to_country_mapping = {
    'American': 'United States',
    'British': 'United Kingdom',
    'French': 'France',
    'German': 'Germany',
    'Italian': 'Italy',
    'Japanese': 'Japan',
    'Chinese': 'China',
    'Indian': 'India',
    'Australian': 'Australia',
    'Canadian': 'Canada',
    'Brazilian': 'Brazil',
    'Russian': 'Russia',
    'South African': 'South Africa',
    'South Korean': 'South Korea',
    'Mexican': 'Mexico',
    'Nigerian': 'Nigeria',
    'Argentinian': 'Argentina',
    'Egyptian': 'Egypt',
    'Thai': 'Thailand',
    'Spanish': 'Spain',
    'Swedish': 'Sweden',
    'Netherlands': 'Netherlands',
    'Turkish': 'Turkey',
    'Switzerland': 'Switzerland',
    'Norwegian': 'Norway',
    'Danish': 'Denmark',
    'Ireland': 'Ireland',
    'Kenyan': 'Kenya',
    'Nigerien': 'Niger',
    'Moroccan': 'Morocco',
    'Ethiopian': 'Ethiopia',
    'Ghanaian': 'Ghana',
    'Ugandan': 'Uganda',
    'Senegalese': 'Senegal',
    'Cameroonian': 'Cameroon',
    'Tanzanian': 'Tanzania',
    'Rwandan': 'Rwanda',
    'Sudanese': 'Sudan',
    'Ivorian': 'Ivory Coast',
    'Chinese': 'China',
    'Japanese': 'Japan',
    'Indian': 'India',
    'Indonesian': 'Indonesia',
    'South Korean': 'South Korea',
    'Vietnamese': 'Vietnam',
    'Filipino': 'Philippines',
    'Thai': 'Thailand',
    'Malaysian': 'Malaysia',
    'Singaporean': 'Singapore',
    'Bangladeshi': 'Bangladesh',
    'Pakistani': 'Pakistan',
    'Sri Lankan': 'Sri Lanka',
    'Nepali': 'Nepal',
    'Mongolian': 'Mongolia',
    'Kazakhstani': 'Kazakhstan',
    # Add more mappings as needed
}
```

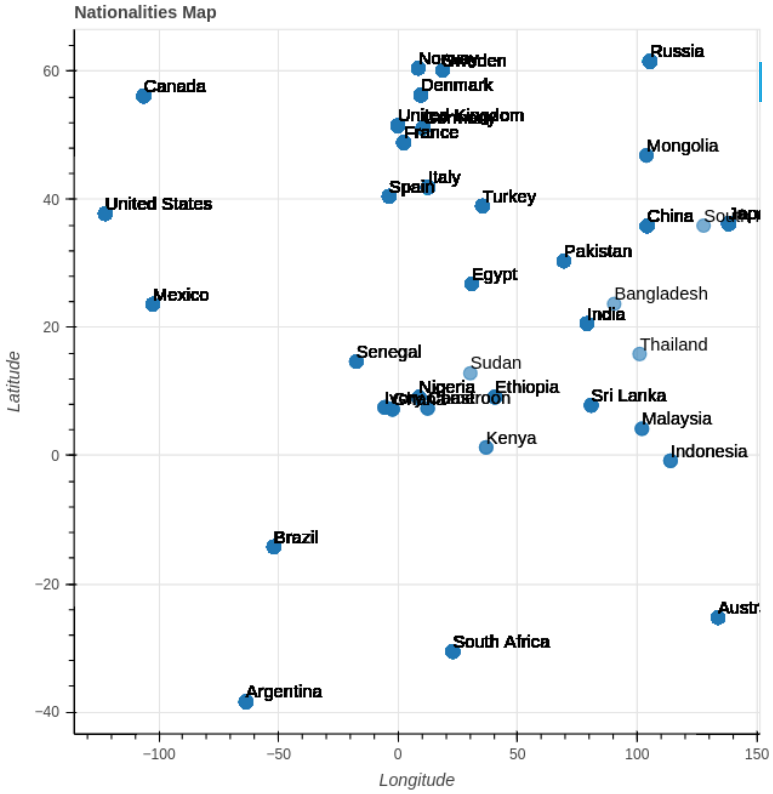
```
final_data['Country'] = final_data['Artist Nationality'].map(nationality_to_country_mapping)

# Filter out rows with missing countries
final_data = final_data.dropna(subset=['Country'])

# Extended manual latitude and longitude for more countries (replace with actual values)
country_coordinates = {
    'United States': {'Latitude': 37.7749, 'Longitude': -122.4194},
    'United Kingdom': {'Latitude': 51.509865, 'Longitude': -0.118092},
    'France': {'Latitude': 48.8566, 'Longitude': 2.3522},
    'Germany': {'Latitude': 51.1657, 'Longitude': 10.4515},
    'Italy': {'Latitude': 41.9028, 'Longitude': 12.4964},
    'Japan': {'Latitude': 35.6895, 'Longitude': 139.6917},
    'China': {'Latitude': 39.9042, 'Longitude': 116.4074},
    'India': {'Latitude': 20.5937, 'Longitude': 78.9629},
    'Australia': {'Latitude': -25.2744, 'Longitude': 133.7751},
    'Canada': {'Latitude': 56.1304, 'Longitude': -106.3468},
    'Brazil': {'Latitude': -14.235, 'Longitude': -51.9253},
    'Russia': {'Latitude': 61.524, 'Longitude': 105.3188},
    'South Africa': {'Latitude': -30.5595, 'Longitude': 22.9375},
    'South Korea': {'Latitude': 35.9078, 'Longitude': 127.7669},
    'Mexico': {'Latitude': 23.6345, 'Longitude': -102.5528},
    'Nigeria': {'Latitude': 9.0820, 'Longitude': 8.6753},
    'Argentina': {'Latitude': -38.4161, 'Longitude': -63.6167},
    'Egypt': {'Latitude': 26.8206, 'Longitude': 30.8025},
    'Thailand': {'Latitude': 15.8700, 'Longitude': 100.9925},
    'Spain': {'Latitude': 40.4637, 'Longitude': -3.7492},
    'Sweden': {'Latitude': 60.1282, 'Longitude': 18.6435},
    'Netherlands': {'Latitude': 52.3676, 'Longitude': 4.9041},
    'Turkey': {'Latitude': 38.9637, 'Longitude': 35.2433},
    'Switzerland': {'Latitude': 46.8182, 'Longitude': 8.2275},
    'Norway': {'Latitude': 60.4720, 'Longitude': 8.4689},
    'Denmark': {'Latitude': 56.2639, 'Longitude': 9.5018},
    'Ireland': {'Latitude': 53.1424, 'Longitude': -7.6921},
    'Kenya': {'Latitude': 1.2921, 'Longitude': 36.8219},
    'Niger': {'Latitude': 17.6078, 'Longitude': 8.0817},
    'Morocco': {'Latitude': 31.7917, 'Longitude': -7.0926},
    'Ethiopia': {'Latitude': 9.1450, 'Longitude': 40.4897},
    'Ghana': {'Latitude': 7.2500, 'Longitude': -2.3333},
    'Uganda': {'Latitude': 1.3733, 'Longitude': 32.2903},
    'Senegal': {'Latitude': 14.6928, 'Longitude': -17.4467},
    'Cameroon': {'Latitude': 7.3697, 'Longitude': 12.3547},
    'Tanzania': {'Latitude': -6.369028, 'Longitude': 34.888822},
    'Rwanda': {'Latitude': -1.9403, 'Longitude': 29.8739},
    'Sudan': {'Latitude': 12.8628, 'Longitude': 30.2176},
    'Ivory Coast': {'Latitude': 7.5400, 'Longitude': -5.5471},
    'China': {'Latitude': 35.8617, 'Longitude': 104.1954},
    'Japan': {'Latitude': 36.2048, 'Longitude': 138.2529},
    'India': {'Latitude': 20.5937, 'Longitude': 78.9629},
    'Indonesia': {'Latitude': -0.7893, 'Longitude': 113.9213},
    'South Korea': {'Latitude': 35.9078, 'Longitude': 127.7669},
    'Vietnam': {'Latitude': 14.0583, 'Longitude': 108.2772},
    'Philippines': {'Latitude': 12.8797, 'Longitude': 121.7740},
    'Thailand': {'Latitude': 15.8700, 'Longitude': 100.9925},
    'Malaysia': {'Latitude': 4.2105, 'Longitude': 101.9758},
    'Singapore': {'Latitude': 1.3521, 'Longitude': 103.8198},
    'Bangladesh': {'Latitude': 23.6850, 'Longitude': 90.3563},
    'Pakistan': {'Latitude': 30.3753, 'Longitude': 69.3451},
    'Sri Lanka': {'Latitude': 7.8731, 'Longitude': 80.7718},
    'Nepal': {'Latitude': 28.3949, 'Longitude': 84.1240},
    'Mongolia': {'Latitude': 46.8625, 'Longitude': 103.8467},
    'Kazakhstan': {'Latitude': 48.0196, 'Longitude': 66.9237},
    # Add more mappings as needed
}

# Merge coordinates with the final_data DataFrame
final_data = pd.merge(final_data, pd.DataFrame(country_coordinates).T, left_on='Country', right_index=True, how='left')

from bokeh.plotting import figure, show
from bokeh.io import output_notebook
from bokeh.models import ColumnDataSource
```

museum_art_curation Projects Branches Notebook Dataset Caveats Settings

```
# Create a Bokeh figure
p = figure(title="Nationalities Map",
           x_axis_label="Longitude", y_axis_label="Latitude")

# Create a ColumnDataSource from the DataFrame
source = ColumnDataSource(final_data)

# Add circle glyphs for each country
p.circle(x='Longitude', y='Latitude', size=10, color=Category20_20[0], alpha=0.6, source=source)
p.text(x='Longitude', y='Latitude', text='Country', text_font_size='10pt', source=source)

# Show the plot
show(p)
```

Console Timing Datasets Charts

```
Oldest Object Date: ca. 400,000–240,000 B.C.
Oldest Object Begin Date: -400000.0
Oldest Object End Date: -240000.0
Number of unique non-anonymous artist names: 57858
Most Frequent Artist Name: Walker Evans
Count: 7326
1870
Oldest Object(s) Title(s):
Marble sarcophagus with garlands
Number of unique values in 'Medium' before cleaning: 65214
Number of unique values in 'Medium' after cleaning: 63839
```



Connected to vizier @ http://localhost:5001/vizier-db/api/v1/