Art Acquisition by Gender and Museum Director

SI 206 Final Project Report Winter 2025

By Evelyn Russell and Tessa Kipke

Link to GitHub repository: https://github.com/evelynr1/ET-206-Final-Project.git

A. Project Goals
    a. The initial goals of our project were to access two APIs (the Metropolitan Museum of Art and the Harvard Art Museums) and gather data about the pieces of art in the museum's possession—specifically, the title, the accession year, gender of the artist, and whether it has an image available online (using primaryImage field) for the Metropolitan Museum of Art and the verification level (rating of data completeness) for each piece for the Harvard Art Museums.
    b. Additionally, we planned to access and web-scrape (using BeautifulSoup) three websites: the Metropolitan Museum of Art's directors Wikipedia page, the Harvard Art Museum's directors Wikipedia page, and a Qualtrics web page showing information about the US gender pay gap. From these pages, we planned to gather data about each museum director, their tenure, and the historical gender pay gap in the US.
    c. Lastly, we planned to create a total of four visualizations.

Changes Made Since Our Presentation
a. Writing calculations to a file
    i. In the met_graphic.py file, we added a function (save_data_to_file) that creates a calculations.txt file and writes some of the Metropolitan Museum and Harvard Art Museums calculations that we used to create the visualizations. Specifically, it adds a counts of how many female and male artists' art were acquired by each director for both museums in an organized and well-formatted way.
    ii. In the harvard_graphics.py file, we added a function (write_calulations) that appends the aforementioned calculations.txt file with our additional calculations about the Harvard Art Museums (specifically, how many pieces of art were acquired in each timeframe, a count of the total number of artworks acquired, and counts of the distribution between male and female artists from the total).
b. Adding data to database in batches of 25
    i. In order to add only 25 rows of data to our database at a time, we altered two functions (add_harvard_art_from_json and add_met_art_from_json) in the database.py file. We created a count variable (new_inserts) that keeps track of how many rows of data had been successfully added to the database per function run and set the for loop to break once the count reached 25.

    ii. This changed how our code needs to be run; now, our database.py file needs to be run 5 times to capture all of our data. It adds 25 rows of data per run during the first 4 runs, then adds the remainder of the data during the 5th and final run, per suggestions from the teaching team. This prevents the need to run the file 30+ times to gather the 800+ rows of data from the Harvard Art Museums.

B. Achieved Goals

a. We successfully used the two APIs (the [Metropolitan Museum of Art] and the [Harvard Art Museums]) to gather data about the pieces of art in the museum's possession.
    i. The data we gathered from the Metropolitan Museum API consisted of the objectIDs (unique integer IDs) of works of art that were "highlights" (a field we were able to filter by using the API); the gender of the artists (artistGender), and the year of museum accession (accessionYear).
    ii. The data we gathered from the Harvard Art Museums API consisted of object IDs (unique integer IDs) of works of art, the accession year (accessionyear), and gender (gender) of the artist.
b. We also successfully accessed and web-scraped (using BeautifulSoup) two websites: the Metropolitan Museum of Art's directors [Wikipedia page] and the Harvard Art Museum's directors [Wikipedia page].
    i. From each page, we gathered data consisting of the names of each historical museum director, the year their tenure began (start_year), and the year their tenure ended (end_year).
    ii. We did not end up web-scraping the Qualtrics page with US gender gap data.
c. We successfully created four visualizations: Metropolitan Museum highlights by artist gender; artwork count by museum and artist gender; amount of art acquired by accession year at the Harvard Art Museums; and, the ratio of art by male and female artists at the Harvard Art Museums.

C. Problems We Faced

a. Throughout the project, we had issues with pushing and pulling on GitHub and not inadvertently overriding or causing conflicts with each other's code. We had success with separating the files into folders and making sure to only change one file at a time, but there were occasions where small changes were made and we got out-of-sync. To solve this, we learned to use Git commands like git --abort that reverted us out of in-progress commits. Additionally, we realized that because our code created files every time it was run, we needed to delete those files from our machines before pushing and pulling.

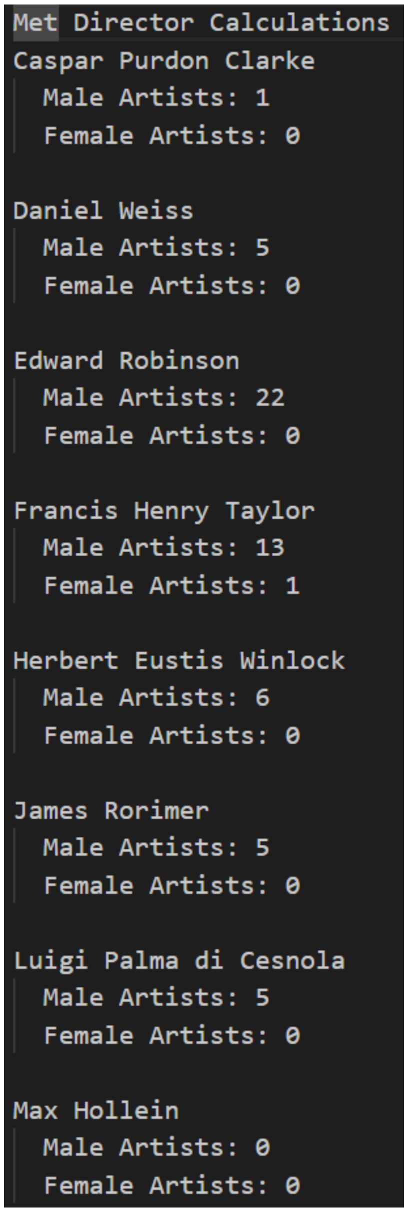
b. We had some difficulties accessing data from the Met API because the way that it's structured means that you have to run two requests: one search to get a list of objectIDs, and one request to access details about the art. The initial searches returned thousands or millions of objectIDs, so we had to find a way to filter them (isHighlight = 'true'). Another quirk with the API was that the objectIDs couldn't be filtered by relevant fields like artist gender (the searches were "fuzzy"), so we had to organize them after the fact.
c. The Harvard Art Museums API presented difficulties because the requests returned thousands of pages of art; we had to learn how to specify which page we wanted information from and iterate through them to get a random sample.
d. We had some issues doing database joins to get the data from the database—to create the graphics, we needed to join our Directors and Art tables using methods like ON, WHERE, BETWEEN, and GROUP BY. This required review of SQLite documentation and class materials.
e. We also had to review how to access and fetch data from our Gender table in order to use a shared integer ID between multiple tables.
f. In order to meet the requirement of adding 25 pieces of data at a time to the database, we had to insert limitation loops and print statements into our add_met_art_from_json and add_harvard_art_from_json functions, which took consultation with the teaching team and some experimentation.
g. We were confused about how to document the functions for this report and had to consult the teaching team for clarification.

D. The Calculations From the Data in the Database (i.e. a screenshot)

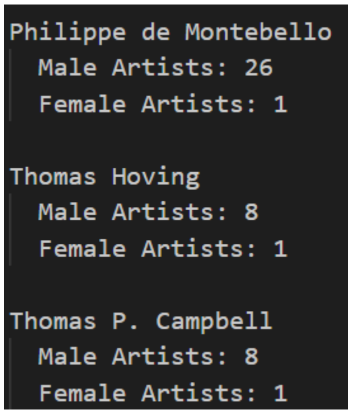
Summary:

- Met Director Calculations
- Harvard Director Calculations
- Harvard Art Museums Calculations
    - Harvard Art by Accession Year
    - Harvard Art by Artist Gender

Met Directors Calculations

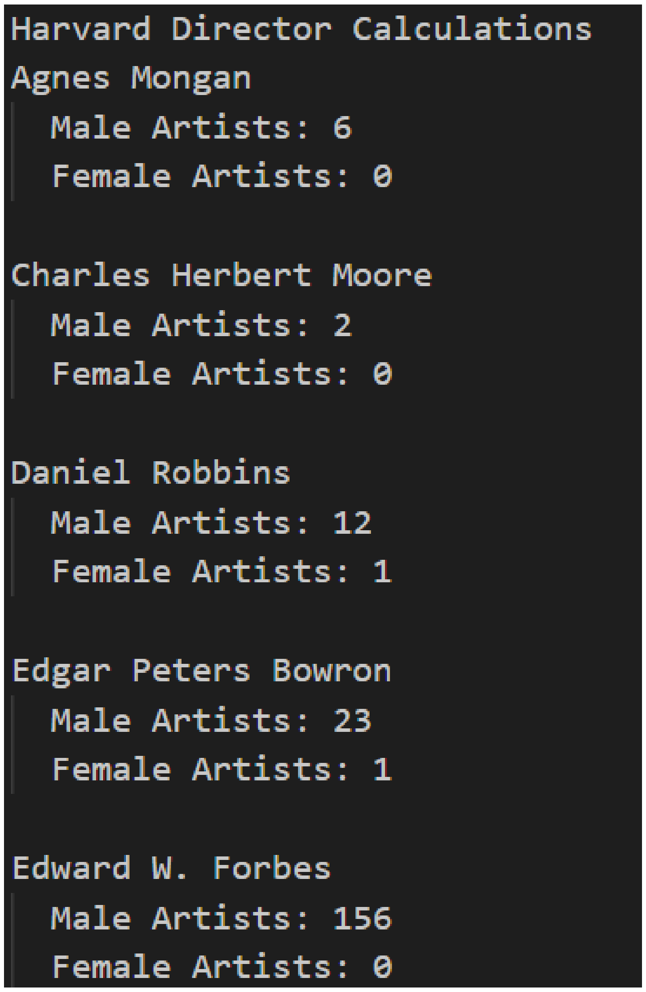


```
Met Director Calculations
Caspar Purdon Clarke
    Male Artists: 1
    Female Artists: 0

Daniel Weiss
    Male Artists: 5
    Female Artists: 0

Edward Robinson
    Male Artists: 22
    Female Artists: 0

Francis Henry Taylor
    Male Artists: 13
    Female Artists: 1

Herbert Eustis Winlock
    Male Artists: 6
    Female Artists: 0

James Rorimer
    Male Artists: 5
    Female Artists: 0

Luigi Palma di Cesnola
    Male Artists: 5
    Female Artists: 0

Max Hollein
    Male Artists: 0
    Female Artists: 0
```

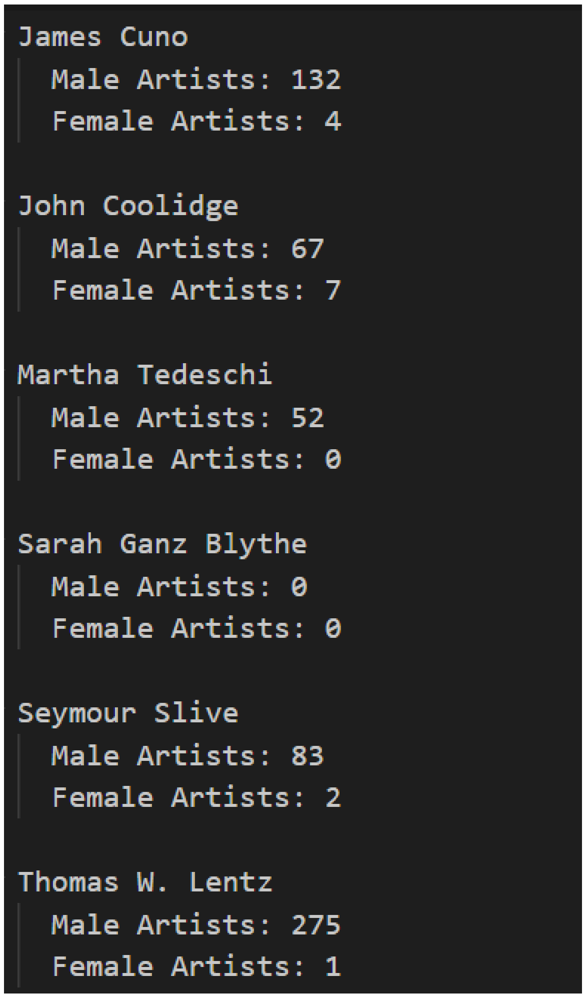


```
Philippe de Montebello
    Male Artists: 26
    Female Artists: 1

Thomas Hoving
    Male Artists: 8
    Female Artists: 1

Thomas P. Campbell
    Male Artists: 8
    Female Artists: 1
```

Harvard Directors Calculations



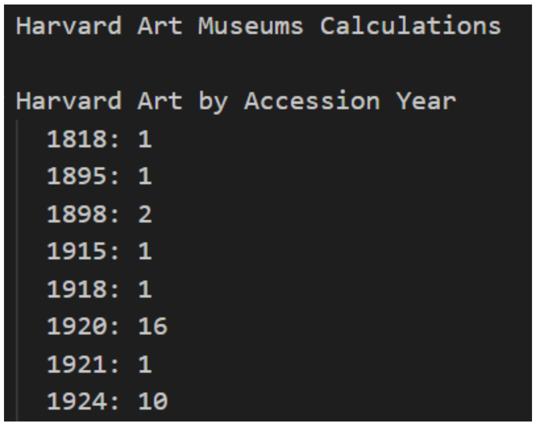
```
Harvard Director Calculations
Agnes Mongan
  Male Artists: 6
  Female Artists: 0

Charles Herbert Moore
  Male Artists: 2
  Female Artists: 0

Daniel Robbins
  Male Artists: 12
  Female Artists: 1

Edgar Peters Bowron
  Male Artists: 23
  Female Artists: 1

Edward W. Forbes
  Male Artists: 156
  Female Artists: 0
```



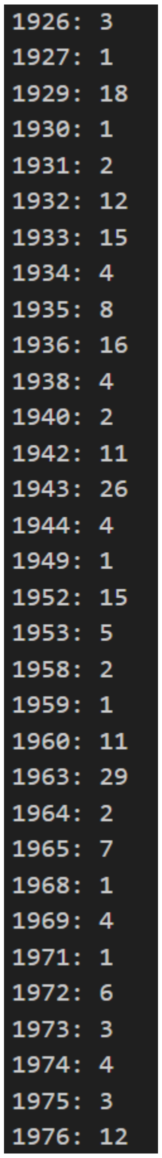
```
James Cuno
  Male Artists: 132
  Female Artists: 4

John Coolidge
  Male Artists: 67
  Female Artists: 7

Martha Tedeschi
  Male Artists: 52
  Female Artists: 0

Sarah Ganz Blythe
  Male Artists: 0
  Female Artists: 0

Seymour Slive
  Male Artists: 83
  Female Artists: 2

Thomas W. Lentz
  Male Artists: 275
  Female Artists: 1
```

Harvard Art Museums Calculations

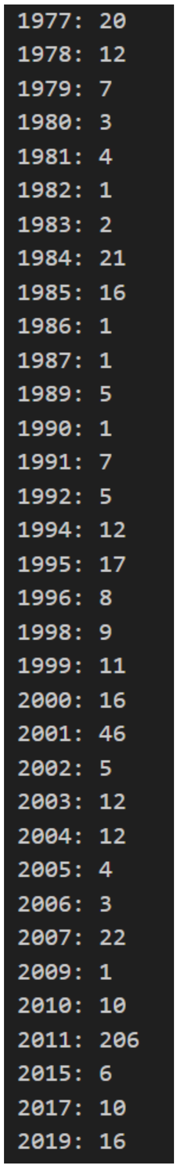
Harvard Art by Accession Year



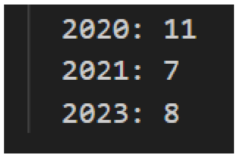
```
Harvard Art Museums Calculations

Harvard Art by Accession Year
  1818: 1
  1895: 1
  1898: 2
  1915: 1
  1918: 1
  1920: 16
  1921: 1
  1924: 10
```



```
1926: 3
1927: 1
1929: 18
1930: 1
1931: 2
1932: 12
1933: 15
1934: 4
1935: 8
1936: 16
1938: 4
1940: 2
1942: 11
1943: 26
1944: 4
1949: 1
1952: 15
1953: 5
1958: 2
1959: 1
1960: 11
1963: 29
1964: 2
1965: 7
1968: 1
1969: 4
1971: 1
1972: 6
1973: 3
1974: 4
1975: 3
1976: 12
```

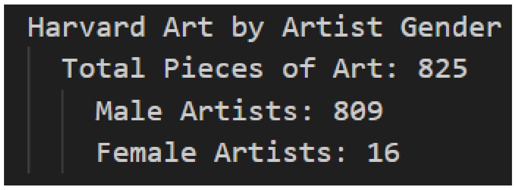


```
1977: 20
1978: 12
1979: 7
1980: 3
1981: 4
1982: 1
1983: 2
1984: 21
1985: 16
1986: 1
1987: 1
1989: 5
1990: 1
1991: 7
1992: 5
1994: 12
1995: 17
1996: 8
1998: 9
1999: 11
2000: 16
2001: 46
2002: 5
2003: 12
2004: 12
2005: 4
2006: 3
2007: 22
2009: 1
2010: 10
2011: 206
2015: 6
2017: 10
2019: 16
```



```
  2020: 11
  2021: 7
  2023: 8
```

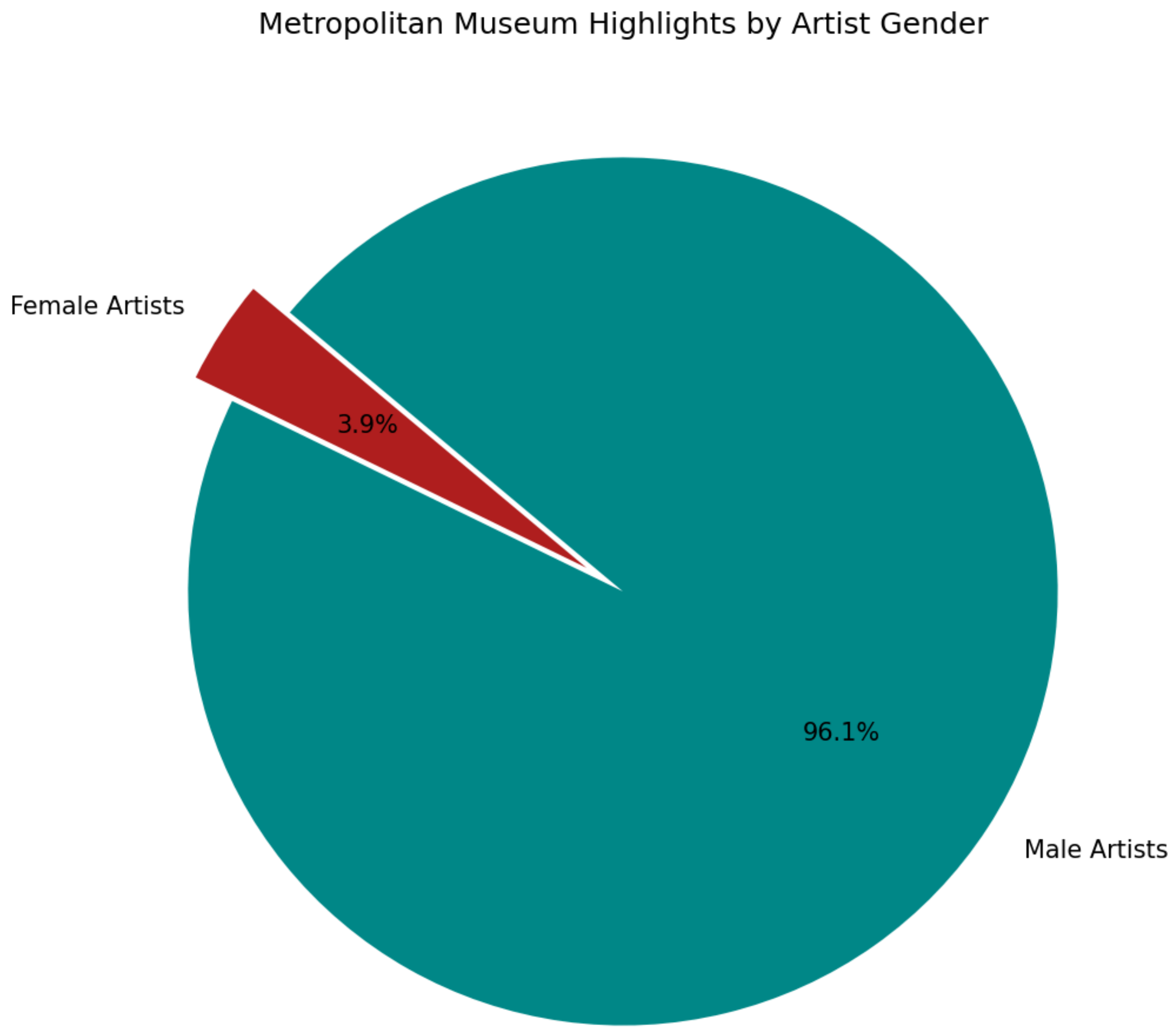
Harvard Art by Artist Gender



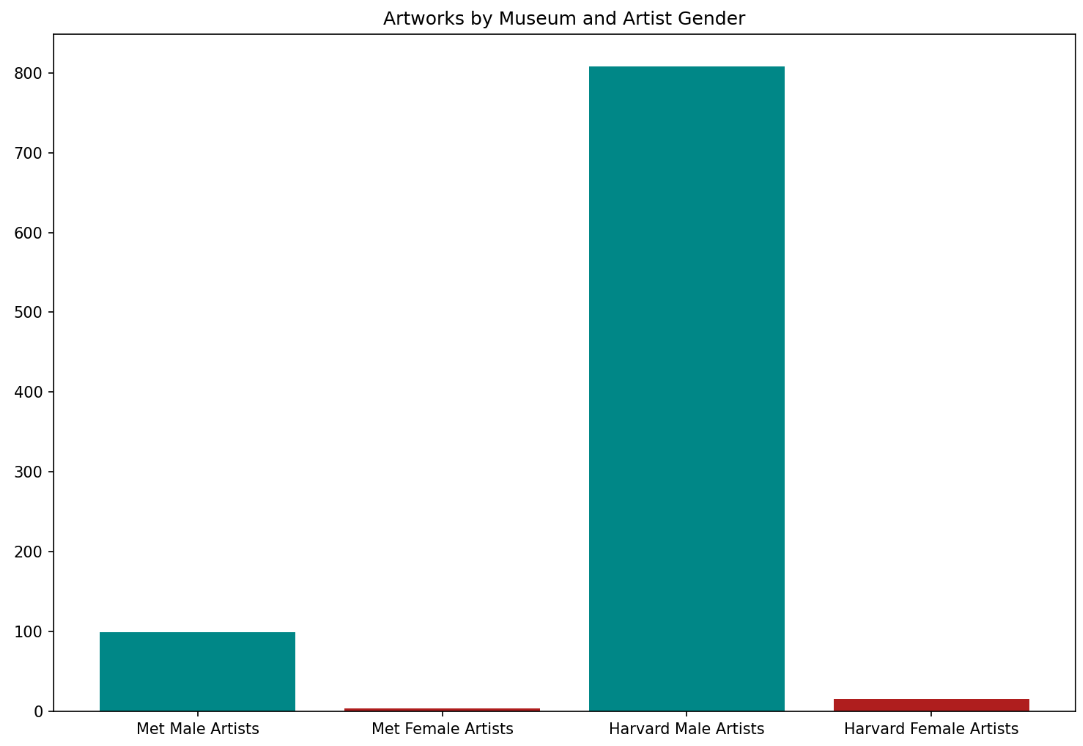
```
Harvard Art by Artist Gender
  Total Pieces of Art: 825
    Male Artists: 809
    Female Artists: 16
```

E. The Visualizations That We Created (i.e. screenshot or image file)

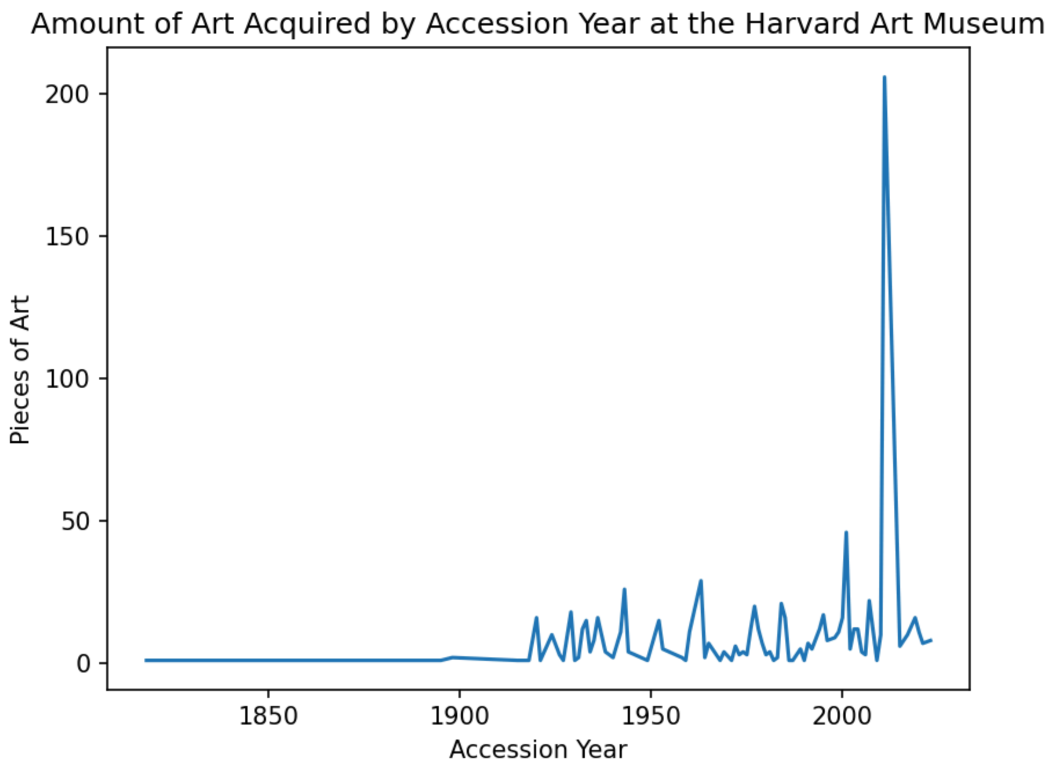
met_pie_chart.png



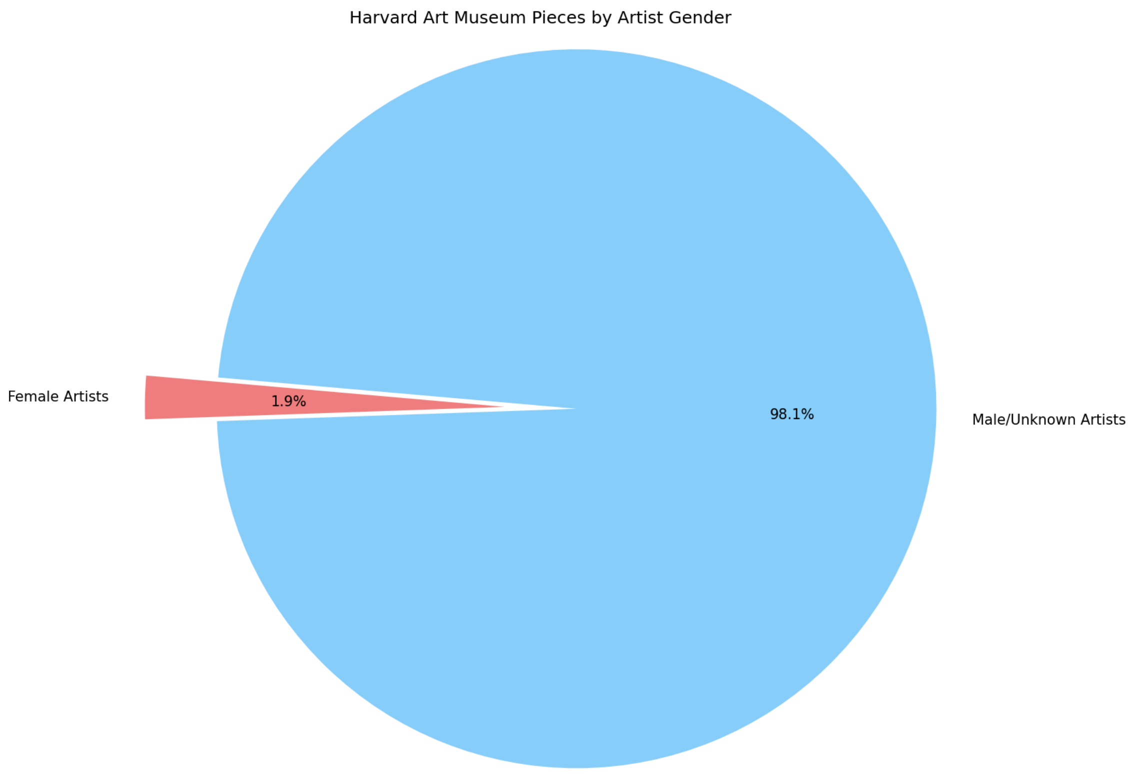
comparison_bar_chart.png



harvard_art_by_accession_year.png



harvard_art_by_gender.png



F. Instructions for Running Our Code

Due to the interactions between our code files necessary to create the database, graphics, and text file with the calculations (calcuations.txt), the code files must be run in the order specified below.

Steps	Folder	Instructions	Outcomes
# 1	Harvard Museum	Run harvard.py one time (~36 seconds)	Creates ‘harvard.json’ and ‘harvard_directors.json’
# 2	Met Museum	Run met.py one time (~43 seconds)	Creates ‘met.json’
# 3	Met Museum	Run metwiki.py one time (~.5 seconds)	Creates ‘met_directors.json’
# 4	Database	Run database.py five times (the first 4 times add 25 rows at a time, and the 5th	Creates ‘Art.db’ and uses ‘harvard.json’, ‘harvard_directors.json’, ‘met.json’, and ‘met_directors.json’ to insert data into

		run adds the remainder, preventing the need to run the file 30+ times)	‘Art.db’
# 5	Database	Run met_graphics.py one time	Uses ‘Art.db’ to create and save a pie chart (‘met_pie_chart.png’) and a bar chart (‘comparison_bar_chart.png’). Creates and writes calculations to ‘calculations.txt’
# 6	Database	Run harvard_graphics.py one time	Uses ‘Art.db’ to create and save a pie chart (‘harvard_art_by_gender.png’) and a line graph (‘harvard_art_by_accession_year.png’). Appends calculations to ‘calculations.txt’

G. Documentation for each function that you wrote. This includes describing the input and output for each function (20 points)

harvard.py		
Name and Purpose	Inputs	Outputs
get_harvard_api_key Retrieves API key from a text file	**filename** (str): text file where the API key is stored	**api_key** (str): the API key
get_art_data Retrieves list of art from Harvard Art Museums API including objectID, accession year, and artist gender	**harvard_api_key** (str): API key for the Harvard Art Museums API	**art_list** (list): list of dictionaries where each dictionary represents one piece of art
find_harvard_directors Scrapes Wikipedia page for bulleted list of Harvard directors, retrieves each director's name, start year, and end year	N/A	**directors_list** (list): list of dictionaries with director name, start year, and end year
write_data_to_file Saves data to a JSON file	**json_data** (dict): data to be written into JSON file **filename** (str): name of the file to write data to	None
main		

Retrieves API key, gets the art pieces and directors, and saves results to separate JSON files		

met.py		
Name and Purpose	Inputs	Outputs
get_objectIDs Gets a list of objectIDs from the Met Museum API for highlighted objects	N/A	**objectIDs** (list): a list of object IDs (integers) representing highlighted artworks
get_object_details Gets details for each objectID, filtering by accession year (must be greater than or equal to 1879) and handling missing artist gender cases (by assigning the gender to male/unknown)	**objectIDs** (list): a list of object IDs to retrieve details for	**result** (list): a list of dictionaries containing objectID, artistGender, and accessionYear
create_file_from_json_data Saves data to a JSON file	**filename** (str): name of the file to save data to **data** (dict): data to write to the JSON file	None
main Gets object IDs, gets the details of the object IDs, and saves the results to a JSON file		

met_wiki.py		
Name and Purpose	Inputs	Outputs
load_wiki_results Scrapes Wikipedia page for list of Met directors, extracting each director's name, start year, and end year of tenure	N/A	**objectIDs** (list): list of dictionaries with director name, start year, and end year
create_file_from_json_data Saves data to a JSON file	**filename** (str): name of the file to write data to **data** (dict): data to be	None

	written into a JSON file	

database.py		
Name and Purpose	Inputs	Outputs
set_up_database Sets up the database	**db_name** (str): name of the database	Returns the connection (**conn**) and cursor (**cur**) of the Art database
create_gender_table Creates and fills the Genders table within the database	The connection (**conn**) and cursor (**cur**) of the Art database	None
create_harvard_directors_table Creates the HarvardDirectors table within the database	The connection (**conn**) and cursor (**cur**) of the Art database	None
add_harvard_directors_from_json Fills the HarvardDirectors table in the database with data from the JSON file of filename (str) using the input connection (conn) and cursor (cur)	The connection (**conn**) and cursor (**cur**) of the Art database	None
create_harvard_art_table Creates the HarvardArt table within the database	The connection (**conn**) and cursor (**cur**) of the Art database	None
add_harvard_art_from_json Fills the HarvardArt table in the database with data from the JSON file of filename (str) using the input connection (conn) and cursor (cur)	The connection (**conn**) and cursor (**cur**) of the Art database **filename** (str): name of JSON file containing a list of dictionaries where each dictionary represents one piece of art	None
create_met_directors_table Creates the MetDirectors table within the database	The connection (**conn**) and cursor (**cur**) of the Art database	None
add_met_directors_from_json Fills the MetDirectors table in the	The connection (**conn**) and cursor (**cur**) of the Art	None

database with data from the JSON file of filename (str) using the input connection (conn) and cursor (cur)	database	
create_met_art_table Creates the MetArt table within the database	The connection (**conn**) and cursor (**cur**) of the Art database	None
add_met_art_from_json Fills the MetArt table in the database with data from the JSON file of filename (str) using the input connection (conn) and cursor (cur)	The connection (**conn**) and cursor (**cur**) of the Art database **filename** (str): name of JSON file containing a list of dictionaries where each dictionary represents one piece of art	None
main Sets up the database, adds all 5 tables and fills them with data from 4 JSON files		

met_graphics.py		
Name and Purpose	Inputs	Outputs
get_data Gets and sorts Met artwork acquisition data (stored in database) by museum director and artist gender	The connection (**conn**) and cursor (**cur**) of the Art database	**director_dict** (dict): dictionary with Met director names as keys and counts of male and female artworks as values
get_harvard_data Gets and sorts Harvard artwork acquisition data (stored in database) by museum director and artist gender	The connection (**conn**) and cursor (**cur**) of the Art database	**director_dict** (dict): dictionary with Harvard director names as keys and counts of male and female artworks as values
make_total_gender_pie Makes, displays, and saves pie chart (‘met_pie_chart.png’) comparing number of artworks by male and female artists at Met Museum	**director_dict** (dict): dictionary with Met director names as keys and counts of male and female artworks as values	None

make_comparison_chart Creates and saves bar chart Makes, displays, and saves bar chart (‘comparison_bar_chart.png’) comparing number of artworks by male and female artists between the Harvard and Met museums	**harvard_dict** (dict): dictionary with Harvard director names as keys and counts of male/female artworks as values **director_dict** (dict): dictionary with Met director names as keys and counts of male/female artworks as values	None
save_data_to_file Creates text file ‘calculations.txt’ and writes calculations (input as parameters data_dict1 and data_dict2) to the file	**data_dict1** (dict): dictionary of directors with counts of artworks by gender **data_dict2** (dict): dictionary of directors with counts of artworks by gender **filename** (str): name of txt file to write the data to	None
main Sets up connection to Art database, creates and saves ‘met_pie_chart.png’ and 'comparison_bar_chart.png' and creates and saves calculations to 'calculations.txt'		

harvard_graphics.py		
Name and Purpose	Inputs	Outputs
art_by_accession_year Creates, displays, and saves a pie chart (‘harvard_art_by_accession_year.png’) of number of pieces of art by gender at the Harvard Art Museums using data from the HarvardArt database	The connection (**conn**) and cursor (**cur**) of the Art database	**art_count_by_accession_year** (dict): dictionary containing art count by accession year (ascending by chronological order)

art_by_gender Creates, displays, and saves a line plot ('harvard_art_by_gender.png') of amount of art by accession year using data from the HarvardArt database	The connection (**conn**) and cursor (**cur**) of the Art database	**(art_by_women, art_by_men_or_unknown)** (tuple): tuple containing art count by gender – **art_by_women** (tuple): (id of gender in Genders table of database, count of art by women) – **art_by_men_or_unknown** (tuple): (id of gender in Genders table of database, count of art by men/unknown)
write_calculations Appends counts of art by accession year and art by gender for the harvard Art Museums to text file	**art_by_year** (dict): dictionary containing art count by accession year (ascending by chronological order) **art_by_women** (tuple): (id of gender in Genders table of database, count of art by women) **art_by_men_or_unknown** (tuple): (id of gender in database, count of art by men/unknown) **filename** (str): name of text file to write the calculations to	None
main Sets up connection to Art database, creates and saves 'harvard_art_by_accession_year.png' and 'harvard_art_by_gender.png' and appends calculations to 'calculations.txt' which is created in met_graphics.py		

H. Resources We Used

	Issue Description	Location of Resource	Result

Date			(did it solve the issue?)
4/5/25	Reviewed how to call an API	https://www.w3schools.com/python/ref_requests_response.asp	Yes
4/5/25	Reviewed how to use json.dump()	https://portal.perforce.com/s/article/JSON-Difference-between-json-dump-and-json-dumps	Yes
4/11	Reviewed how to make a pie chart in Matplotlib	https://www.w3schools.com/python/matplotlib_pie_charts.asp	It was a good start, but our database is not fully finished so we cannot test if it works yet. Update on 4/15: Yes
4/13	Investigated issues with filtering art by field with Met API	https://chatgpt.com/	Yes; it explained elements of the Met API's documentation that were unclear (certain fields couldn't be used for filtering)
4/14	Reviewed how to write docstrings in Python	https://peps.python.org/pep-0257/ https://www.programiz.com/python-programming/docstrings	Yes
4/14	Reviewed how to format SQL joins for graphic creation	https://www.w3schools.com/sql/sql_join.asp#gsc.tab=0	Yes
4/14	Reviewed how to create bar charts in matplotlib	SI 206 Lecture slides (L21)	Yes

4/15	How to add 25 things to the database at a time	##Code from Colleen on 4/15/25
		new_inserts = 0
		  for item in api_data:
		    if new_inserts >= 25:
		        break
		
		    execute statement
			Yes

		`if cur.rowcount == 1:` `new_inserts += 1` `conn.commit()`	
4/15	How to append to a text file	https://www.w3schools.com/python/python_file_write.asp	Yes
4/16	How to document the functions for this report	Eva's project (shown in office hour)	Yes
4/16	Still working on how to add 25 at a time to the database	https://www.sqlitetutorial.net/sqlite-count-function/#:~:text=The%20COUNT(*)%20function%20returns,rows%20including%20NULL%20and%20duplicates.	Yes