SI 206 - Final Project Report

Priya Ganji - Art Institute of Chicago
Sneha Yedulla - The Metropolitan Museum of Art
Paru Banjara - Harvard Art Museum

GitHub Repository Link: https://github.com/priyavganji/finalproject206

Introduction

Our final project focused on collecting data about individual artworks from three separate museums. These museums are the Art Institute of Chicago (which Priya worked on), the Metropolitan Museum of Art (which Sneha worked on), and lastly the Harvard At Museum (which Paru worked on). As all of us are interested in art and visiting museums, we thought these would be great API's to work with.

Goal

Our goal of this project was to answer the question: What is the average duration an artwork took in each of the 3 museums? So, our project focused on the start dates and end dates (in years) for artworks in the 3 museums. We hoped to discover which museum has artworks that took the least and most amounts of time, on average. Even though this was our main question, we have researched and made many other interesting findings along the way.

We feel our findings can lead to more detailed research in the future regarding why some pieces took longer than others, maybe for historical reasons or the typical lifestyle during a certain time period.

We divided the report for each of us to talk about our individual findings and work, before coming together for a combined analysis.

Priya (p. 2-6)

Problems:

Some problems I had faced were during the early stages when I was trying to collect data. As it was a public API, it took a long time for me to understand and read the documentation provided to pull the exact data I wanted. Modifying the URL correctly was a trial and error process. After getting json data from the API, the next struggle was going into the numerous dictionaries and lists to get the exact information I wanted. This was a tedious process and I used json validators and cleaners to understand and look at my data better.

Calculation file:

This is my calculation file. This is a dictionary with detailed keys to represent all my calculations. I have put everything into a json file. We must also keep in mind the API is changing on a daily basis as the museums input data, hence these numbers might change frequently.

```
{} AIChicago_cal.json > ...
{
    "AVG start year is: ": 1729.236,
    "MAX start year is: ": 2021,
    "MIN start year is: ": -945,
    "AVG end year is: ": 1749.148,
    "MAX end year is: ": 2021,
    "MIN end year is: ": -800,
    "Average # of years an art piece took": 19.912
}
```

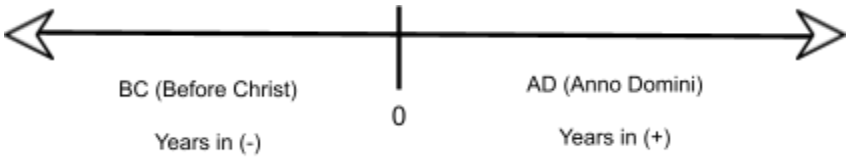

Here, my main calculation states that the average # of years an art piece taken from the Art Institute of Chicago is 19 years to complete, depending on the artworks in my database.

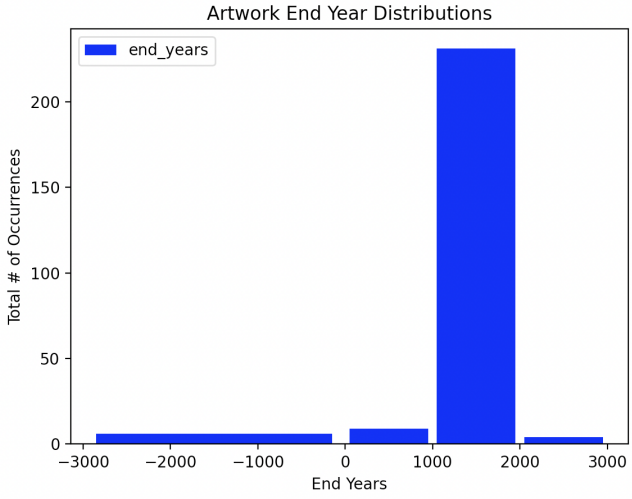
Visualizations:
I have created two visualizations for my database.

The first visualization is a histogram that showcases the number of art pieces corresponding to their end year. It is divided up by 1000 years.

A very important thing to note is the negative numbers stand for BC (Before Christ) on the timeline, hence they are negative, while the positive numbers correspond to AD (Anno Domini). We can technically say we are living in 2022 AD according to the chart.

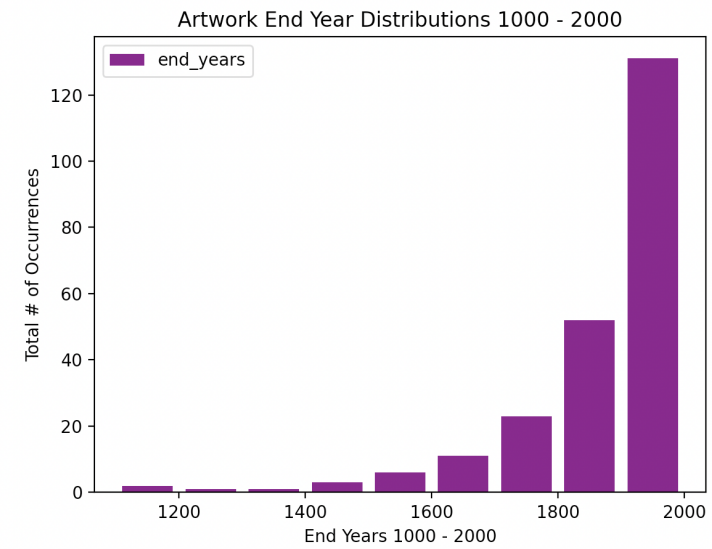
Here is an image to clarify this a little more:





As you can see, the first visualization here has the most pieces of artwork that were finished being created (end years) from the years 1000 - 2000, which are over 200 pieces from my database.

So I wanted to take a deeper look at each of the 100 year intervals to see if there is any time frame that created the most art. In a sense, trying to see if the renaissance or other time periods with historical context can show through in my database.



There is a very clear trend here. It is constantly changing each time I run my code, but it is very clear that most of the artworks found in the museum are from 1900 to 2000.

Instructions for running the code:
In order to run my portion of the code, you will just have to run the file. The database will keep adding each time you run it, and update the json file as well as the data for the calculations and visualizations. One visualization will show up after another, hence commenting out one and then the other could also be another way to more easily run the code if needed.

Code documentation:
This is my code documentation for the 7 functions I created:

```
def setUpDatabase(db_name):
```

This function takes in a database name, that was implemented in main when called, and created the SQL path and connection needed for cur and conn to be used in the rest of the code. This function is returning cur, conn.

```
def create_artworks_table(cur, conn):
```

This function creates a database if it does not exist, called Artworks in SQL with 4 columns: id, Title, date_start, and date_end. The id is the primary key, date start is the year the art piece was started and year end is the year the art piece was finished being created. Nothing is returned in this function except the table is created when conn.commit() is run.

```
def add_artworks_from_json(page, cur, conn):
```

This function takes in a page number, cur and conn. It requests the json data from the specific page provided, which is then inputted into the url for pulling data. This function goes into the url json data and searches for the necessary keys, adding 25 rows of data into the table. Nothing is returned in the python file.

```
def do_calc(cur, conn, file_name):
```

This is my longest function. It takes in cor and conn to be connected with the SQL database but also a file name. This function does 7 total calculations, inputs them into a dictionary with detailed keys, and then puts them into a json file, with the filename provided.

These are the seven calculations that are done in this function for the artworks in the table:

- Average of the start years
- Minimum start year (to find the earliest start date)
- Maximum start year (to find the latest start date)
- Average of the end years
- Minimum end year (to find the earliest end date)
- Maximum end year (to find the latest end date)
- The average of (end date - start date) to find the average amount of time taken for the pieces

```
def visual(cur, conn):
```

This first visualization displays a histogram of the end years of the pieces for each 1000 year time frame. Only cur, and conn are the inputs of this function to be connected to the database. Nothing is returned in this function except the visualization is shown.

```
def visual2(cur, conn):
```

This second visualization displays a histogram of the end years of the pieces for each 100 year time frame from 1000 to 2000 as this was the time frame with the most artworks. Only cur, and

conn are the inputs of this function to be connected to the database. Nothing is returned in this function except the visualization is shown as well.

Documentation of resources:

Date:	Issue Description:	Location of Resource:	Result
11/29/22	I was having a hard time viewing my json file clearly. It had a lot of dictionaries and lists inside one another.	json validator (https://jsonformatter.org/)	This resource totally helped, I was able to even check the number of artworks pulled (which is 25 each time)!
12/05/22	It was a little confusing for me in regards to syntax when doing the calculations. I wasn't sure what functions were available in SQL already.	I used Professor Ericson's slides which had showcased examples to sample calculations.	This really helped as I was able to practice with these functions and later implement them into my code as well.
12/06/22	When creating the visualizations, I felt I didn't have much practice with this in class, and really wanted some help creating them.	https://www.youtube.com/watch?v=woBnzolCH5Q	This video walked me through what each step meant in creating a histogram using mat plot lib, and I got two visualizations to work within my code.
All throughout	As this was a big final project, I had many general questions and concerns I wanted to get resolved.	I went to Office hours as often as I could, talked to Holden one-on-one, and Antionette as well to get help with the project. I attended the in class office hours as well and talked to Professor Ericson regarding some of the questions I had with the rubric.	All of them had been so helpful, and walked me through all the processes. I walked out each time feeling better, and getting my questions answered. Thank you to all of them for their help!

Sneha (p. 7-11)

My Goals:

Goals specific to my portion of the project were to gather data on specific years that artworks were started and finished, the names of artists, the names of artwork's departments, and the artwork's object IDs from the Metropolitan Museum of Art API. After gathering this data into a database table, my goals were to calculate (1) the minimum, maximum, and average start and end years, (2) the longest artwork duration, and (3) the average artwork duration. After doing these calculations, my final goal was to create an engaging visual that captured an aspect of my data really well. I decided to make a histogram showing the distribution of the end years of the artworks. While these were goals specific to my portion of the project, one overall group goal I played a huge role in was writing a detailed combined analysis for the report which compares all of our data.

Goal Achievement:

All of my goals were completed. I successfully made a database table, performed calculations which I neatly arranged in a json file, and made an engaging visual. The goal to calculate the average artwork duration at the Metropolitan Museum of Art was to be able to eventually compare that to the averages my partners calculated for their museums. I found the average artwork took 25.63 years at the MET. Furthermore, the goal to calculate the average start year and the average end year for artworks at the MET was also to be able to compare those to the averages my partners calculated. I found the average start year to be 1813 and the average end year to be 1839. Lastly, I successfully created a histogram showing the distribution of the end years of the artworks. These dates ranged from 1650 to 1927. While I achieved all my goals, it was not without facing a couple of problems.

Problems:

The first problem I faced was not understanding the documentation in the API. I did not know how to properly pull the data I needed. I spent a lot of time reading the documentation before finally understanding how to pull the data. I had to work on the base URL several times before I got it right. The second problem I faced was having confusion on how to do my calculations, but I viewed Professor Ericson's slides of sample calculation examples and figured it out. The third problem I faced was not feeling confident in making a histogram. Watching a YouTube video on it really helped. While problem solving, I made a table of all the resources I used.

Documentation of Resources:

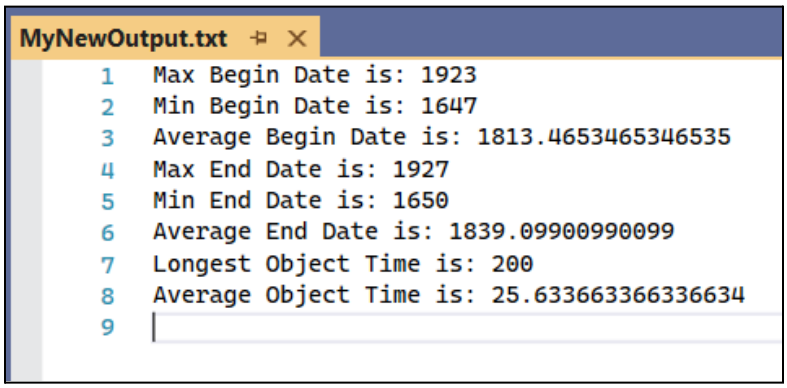
Date:	Issue Description:	Location of Resource:	Result
12/01/2022	Needed to get a better understanding of API	https://betterprogramming.pub/how-to-rea	I understood how to properly pull the data

	documentation, so I read this page online.	d-and-understand-api-documentations-2f894b51d0b7	I needed for my API.
12/01/2022	Same issue as above: I was struggling to understand the documentation.	https://api-toolkit.herokuapp.com/5	Paru recommended this website to me, which helped me understand which information I needed to extract from the API.
12/04/22	I was a little lost on which functions were available in SQL to use for my calculations portion.	Professor Ericson's slides of sample calculation examples.	Viewing the sample calculations gave me a better understanding of how to do my calculations.
12/06/22	When trying to create my visualization, I felt a bit inexperienced, so I watched a youtube video on it before starting.	https://youtu.be/4SL89I9VS44	This video did a great job explaining how to create a histogram using mat plot lib. I was able to successfully make my own histogram.

Calculation file:

As I stated in my "goals" section above, my goals were to calculate (1) the minimum, maximum, and average start and end years, (2) the longest artwork duration, and (3) the average artwork duration. This is my calculation file with all that data. I have made it so that all the data can be neatly extracted into either a txt file or a json file. The units for all data are years.

Here is the txt file:



Here is the json file:

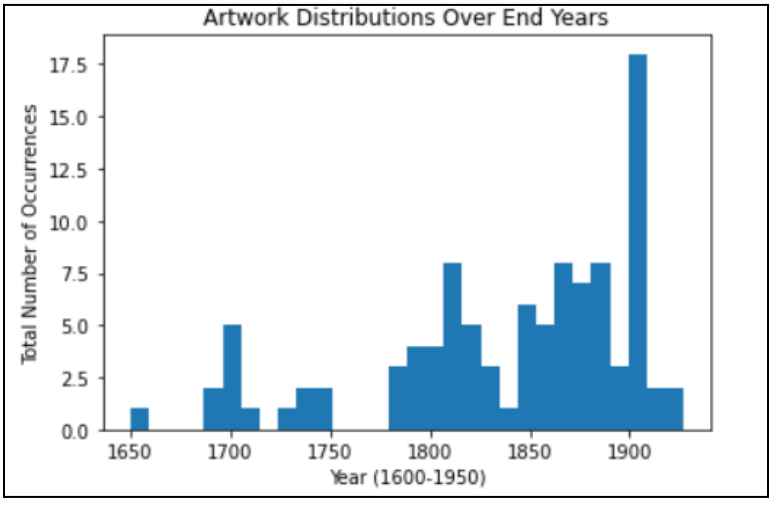
Output.json

Schema:


```
[
    {
        "Max Begin Date is": " 1923"
    },
    {
        "Min Begin Date is": " 1647"
    },
    {
        "Average Begin Date is": " 1813.4653465346535"
    },
    {
        "Max End Date is": " 1927"
    },
    {
        "Min End Date is": " 1650"
    },
    {
        "Average End Date is": " 1839.09900990099"
    },
    {
        "Longest Object Time is": " 200"
    },
    {
        "Average Object Time is": " 25.633663366336634"
    }
]
```


Visualization:

I made a histogram showing the distribution of the end years of the artworks. I labeled the x-axis with the years the artworks were finished in and the y-axis with the total number of occurrences. As you can see, there seems to be a significantly large amount of artworks that were finished being created between 1900 and 1925. Also, a trend I noticed is that the museum has increasingly more artworks as time gets more recent from 1650 to 1930. In other words, the museum has increasingly more amounts of newer artworks.

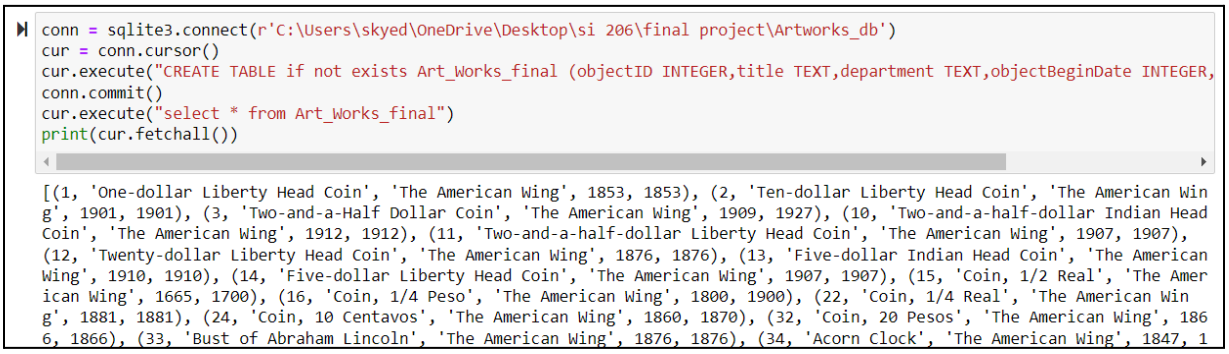


Instructions for running the code:
I did my data analysis in Jupyter notebook so my code is containerized, modular, and readable. Afterwards, I converted the .ipynb file to a .py file and added descriptive comments that explain the role of each function in the code before putting both files in my group's github repository. To run my code, you will open the .ipynb file in Jupyter notebook and just run the file. You will just need to change the path names of where files are located on your computer. The database will add data and the visualization will display when run.

Code documentation (explains role of each cell and its input and output):

```
def setUpDatabase(db_name):
    path = os.path.dirname(os.path.abspath('./'))
    conn = sqlite3.connect(path+'/'+db_name)
    return conn
```

This function takes in a database name and creates the SQL path and connection needed for cur and conn to be used in the code that follows. This function returns conn.



```
conn = sqlite3.connect(r'C:\Users\skyed\OneDrive\Desktop\si 206\final project\Artworks_db')
cur = conn.cursor()
cur.execute("CREATE TABLE if not exists Art_Works_final (objectID INTEGER,title TEXT,department TEXT,objectBeginDate INTEGER,
conn.commit()
cur.execute("select * from Art_Works_final")
print(cur.fetchall())
```

```
[(1, 'One-dollar Liberty Head Coin', 'The American Wing', 1853, 1853), (2, 'Ten-dollar Liberty Head Coin', 'The American Win
g', 1901, 1901), (3, 'Two-and-a-Half Dollar Coin', 'The American Wing', 1909, 1927), (10, 'Two-and-a-half-dollar Indian Head
Coin', 'The American Wing', 1912, 1912), (11, 'Two-and-a-half-dollar Liberty Head Coin', 'The American Wing', 1907, 1907),
(12, 'Twenty-dollar Liberty Head Coin', 'The American Wing', 1876, 1876), (13, 'Five-dollar Indian Head Coin', 'The American
Wing', 1910, 1910), (14, 'Five-dollar Liberty Head Coin', 'The American Wing', 1907, 1907), (15, 'Coin, 1/2 Real', 'The Amer
ican Wing', 1665, 1700), (16, 'Coin, 1/4 Peso', 'The American Wing', 1800, 1900), (22, 'Coin, 1/4 Real', 'The American Win
g', 1881, 1881), (24, 'Coin, 10 Centavos', 'The American Wing', 1860, 1870), (32, 'Coin, 20 Pesos', 'The American Wing', 186
6, 1866), (33, 'Bust of Abraham Lincoln', 'The American Wing', 1876, 1876), (34, 'Acorn Clock', 'The American Wing', 1847, 1
```

This cell accepts the filename of the database as a parameter, and returns a list of dictionaries. The return value is in the format: [(objectID, title, department, objectBeginDate, objectEndDate)...].

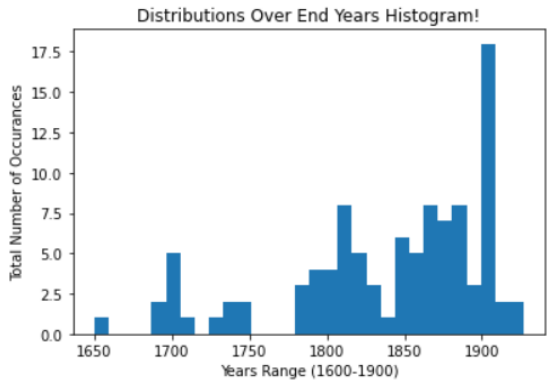
```
## Check for Data!
cur.execute("select * from Art_Works_final")
re= pd.DataFrame(cur.fetchall())
re
```

	0	1	2	3	4
0	1	One-dollar Liberty Head Coin	The American Wing	1853	1853
1	2	Ten-dollar Liberty Head Coin	The American Wing	1901	1901
2	3	Two-and-a-Half Dollar Coin	The American Wing	1909	1927
3	10	Two-and-a-half-dollar Indian Head Coin	The American Wing	1912	1912
4	11	Two-and-a-half-dollar Liberty Head Coin	The American Wing	1907	1907
...	...	...	...	...	...
96	454	Bench	The American Wing	1697	1700
97	457	Sleigh Seat	The American Wing	1797	1800
98	465	Figure of Benjamin Franklin	The American Wing	1835	1845
99	466	Plaque Portrait of Benjamin Franklin	The American Wing	1800	1883
100	472	Portrait of Benjamin Franklin	The American Wing	1776	1883

101 rows × 5 columns

This cell creates a database, called Art_Works_final, with 5 columns: objectID, title, department, objectBeginDate, objectEndDate. Nothing is returned in this function except the table is created when it is run.

```
import matplotlib.pyplot as plt
x = re['objectEndDate']
plt.hist(x, bins=30)
plt.ylabel('Total Number of Occurances')
plt.xlabel('Years Range (1600-1900)')
plt.title('Distributions Over End Years Histogram!')
plt.show()
```



This cell takes in the end dates of the artworks, and outputs my histogram.

Paru (p. 12-18)

Goals:

The specific goal that was relevant to my part of the project was to identify the minimum and the maximum number of paintings along with their corresponding artist names at the Harvard Art Museums from the year 1000 to the year 2000. I successfully achieved this goal by finding the highest number of paintings which was 193 by an unknown artist and lowest number of paintings by Jennifer Bartlett which was 1.

Problems:

The main problems I faced were understanding the API, importing the data from the API, understanding the data model on the API, joining tables and making visualizations. Firstly, I found it difficult to understand how everything on the API worked. After reading through the API documentation carefully and following the instructions as suggested on the API and using another resource from Harvard Art Museums, I started to make progress. The data were organized into different categories, such as object, person, century, classification, and more. Therefore, I had to specify those categories in the URL to import the data I wanted to collect.

After being able to make API calls and collect the data from the API, the next problem was understanding how the data were stored. I saw that the data was stored in a list of dictionaries. For the data model understanding purpose, I stored the data into a json file and formatted it on VS code. It made it easier to see and understand the nested dictionaries. For my calculation and visualizations, I used the data stored in the database. For the join tables part, I used the lecture slides and W3schools website for reference. In order to make the visualizations, I also used different resources that helped me to make progress.

Calculations:

I have included my calculations file below. I imported all the painting records from the year 1000 to the year 2000 in order to accurately calculate the data. I have two tables in my database. One table has two columns which contain id (for the artist) and displayname of the artists and the other table has six columns - id (for the art), title, date end, date begin, culture, and person id as a foreign key. The person table has 1115 rows and the object table has 2573 rows as of December 9, 2022.

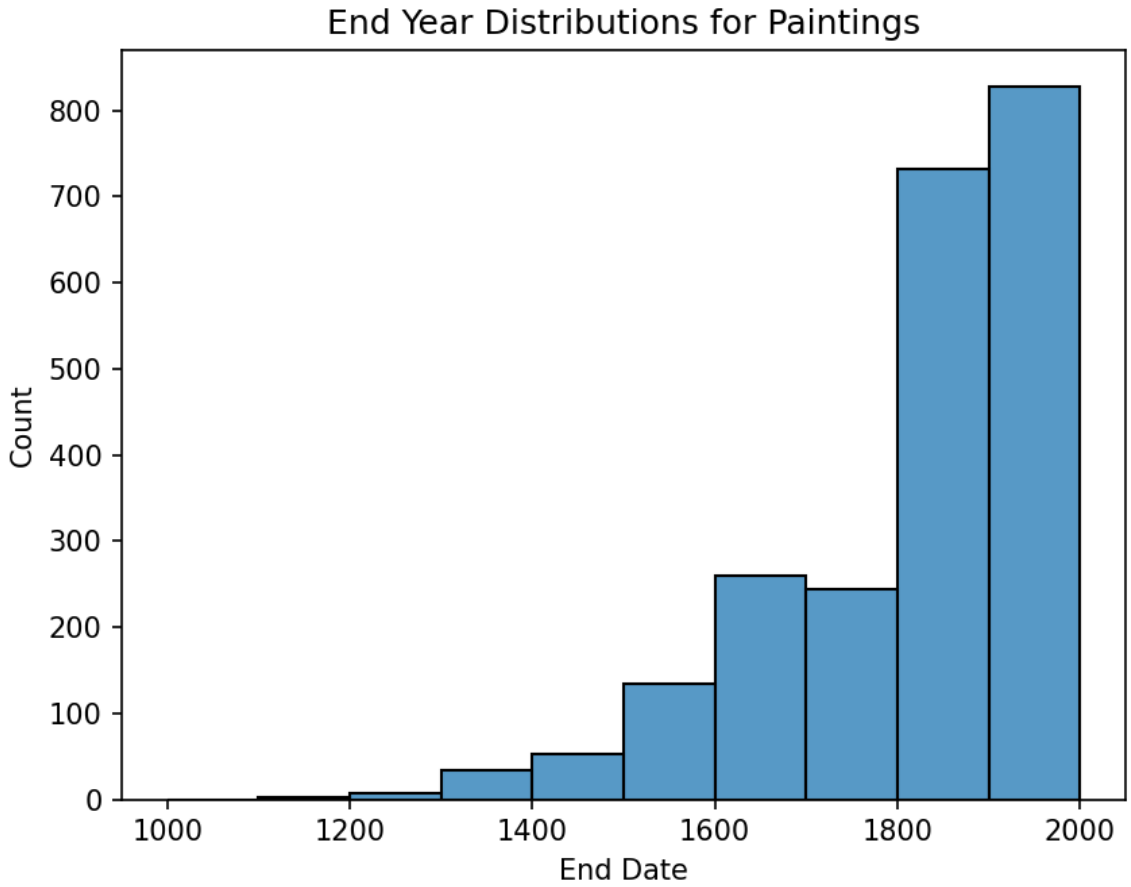
For the data calculation, I calculated the following:

- The name of the artist who created the most paintings and the number of paintings.
- The name of the artist who created the least number of paintings and the number of paintings.
- Maximum, minimum, and average end date for the paintings.
- Maximum, minimum, and average begin date for the paintings.

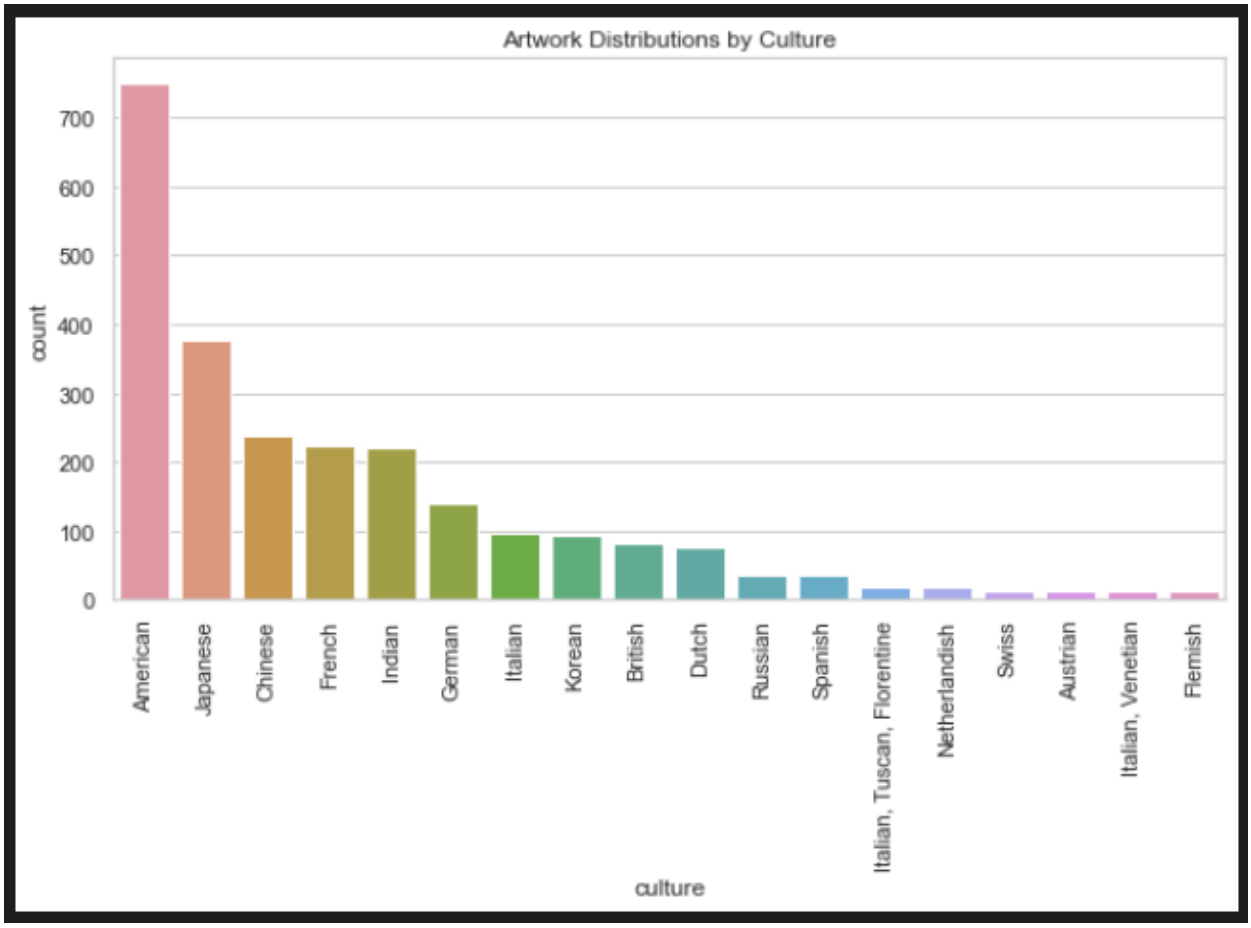
- The average time taken for the completion of a painting throughout the years 1000 to 2000.

```
harvard_api.txt
1   Maximum number of Paintings by Unknown Artist which is 197
2   Minimum number of paintings by Jennifer Bartlett which is 1
3   Maximum End Date: Year 2000
4   Minimum End Date: Year 1140
5   Average End Date: 1816.66 years.
6   Maximum Begin Date: Year 2000
7   Minimum Begin Date: Year 1127
8   Average Begin Date: 1800.71 years.
9   Average Paintings Completion Time: 15.96 years
```

Visualizations:
I have included my visualizations below.



The first visualization shows the number of artworks by their end date, ie. the completion date of the artworks. The x-axis contains the years from 1000 to 2000 and the y-axis contains the paintings count. One interesting thing to notice is that there weren't any artworks completed in the earlier years up until the 1200s. The completion number slowly started to increase after the 1500s. In the 1800s, the number of paintings was doubled in comparison to 1700s. The highest number of paintings were created in the 1900s. It clearly shows that the number of artworks has been in an increasing trend except for the 1700s when there is a little bit of decline in comparison to the 1600s.



The second visualization shows the artwork distributions by culture. The x-axis contains the name of the countries and the y-axis contains the counts of paintings.
The Harvard Art Museum has the highest number of American paintings. The second highest is Japanese and the third highest is Chinese paintings. The museum has the least number of Swiss, Austrian, Italian, and Flemish paintings.

Instructions for running code:
I have set the URL to pull a lot of data from the API, so my program takes some time (about 1 minute) to finish running and importing the data. It stores data into the database in the number of 25 rows at a time. It also shows the visualizations after the program stops running.

Documentation for each function (including input and output):
I have created several functions that have specific roles. I have a total of 11 functions in my python file. Here are the descriptions of each function.

1)
```
def setUpDatabase(db_name):
```
This function takes the name of a database that is passed from the main. Then, it establishes a path and creates a cursor and a connection with the database. It returns the cursor and connection.

2)
```
def create_object_table(cur):
```

This function takes a cursor object. Then, It creates an object table in the database by adding all the specified column names for the table which include id, title, datebegin, dateend, culture and personid (foreign key). This function does not return anything.

3)
```
def create_people_table(cur):
```

This function also takes a cursor object as a parameter. It creates a person table in the database with the column names specified, ie. id and displayname of the artists. This function also doesn't return anything.

4)
```
def get_object_records(cur):
```

The parameter for this function is a cursor object. This function calls the API to make a request for importing the data. I have specified in the URL to get the paintings records that have only one artist working on a painting from years 1000 to 2000. It pulls 100 records at a time from the API, but only 25 rows are stored in the database. It returns a list of dictionaries containing the records of paintings.

5)

```
def get_person_records(cur):
```

This function takes a cursor object as a parameter. It imports information about the artists from the API given the constraints in the URL. This function also inputs only 25 rows in a table at a time. It returns a list of dictionaries containing ids and artist names.

6)

```
def add_object_records(records, cur):
```

This function takes the list of records containing the dictionaries and a cursor object. The role of this function is to add object records to the database table after converting them into a list of tuples. This function doesn't return anything.

7)

```
def join_tables(cur, conn):
```

This function takes a cursor object and a connection object. The role of this function is to join the object and the person table by the foreign key on the object table which is the personid. Personid is the primary key on the person table. Nothing is returned by this function.

8)

```
def calculate_min_max_avg(cur, conn):
```

This function takes a cursor object and a connection object as parameters. The roles of this function are to calculate the maximum, minimum, and average date end (artwork completion date), maximum, minimum, and average date begin (artwork start date), and the average time taken for an artwork to complete. This function is also used to calculate the maximum number of artworks done by an artist and their name and minimum number of artworks done by an artist including their name. In order to calculate that, I joined the two tables by the personid foreign key on the object table. The personid is the primary key on the person table. I sorted the artwork count in descending order so the least number of artworks done by an artist was at the bottom of the list and maximum number of artworks done by an artist was at the top of the list. There were several hundred artists who only completed one painting, but I selected the one artist with the least number of paintings from the bottom of the list. This function outputs the result into a text file and doesn't return anything.

9)

```
def visualization_enddate(cur, conn):
```

Like a lot of other functions, this function also takes a cursor object and a connection object passed from the main. This function creates a histogram for the end dates showing the artwork distributions throughout the years. This function doesn't return anything.

10)

```
def visualization_culture(cur, conn):
```

This function takes a cursor object and a connection object. It creates a barplot for the distribution of artwork by culture on the x-axis and their corresponding count values on the y-axis. This function doesn't return anything.

11)

```
def main():
```

This function is used to call all the other functions and to set up a database. This function doesn't take anything and doesn't return anything.

Documentation of the resources used:

Date	Issue Description	Location of Resource	Result (did it solve the issue?
2022/11/30	I found this resource("API toolkit & guide") on the Harvard API documentation page. I was struggling to understand the documentation prior to finding this resource.	https://api-toolkit.herokuapp.com/5	After I started reading through this toolkit, it made the understanding so much easier because it clearly stated the necessary steps I needed to take to extract the information from API.
2022/12/03	I wanted to add the years 1000 to 2000 in the URL in order to get the artworks records from 1000 to 2000. The API documentation suggested using elastic search for that, so I used this resource.	https://www.elastic.co/guide/en/elasticsearch/reference/5.6/query-dsl-query-string-query.html#query-string-syntax	I found on this website that I could just insert [1000 TO 2000] into my URL. It resolved the issue.

2022/12/04	I used this resource to understand how a join table would work because I was struggling to understand it prior to that. I also referenced the lecture slides to understand the join table and keys.	https://www.w3schools.com/sql/sql_join.asp	It became clear after I went through examples on this website as well as on the lecture slides.
2022/12/08	I used this resource to understand how to create a barplot using seaborn.	https://seaborn.pydata.org/generated/seaborn.barplot.html	I was able to create a barplot after referencing this resource.
2022/12/08	I used a few other resources in order to understand pandas which I used for one of my visualizations. The first resource I used to understand how notna function in pandas worked. It turns out that this function returns a boolean list of values and filters out the non-existing values like none. The second resource I used to understand pandas in general. The third resource I used to understand how to create a connection to my sqlite3 database so that the pandas object could read my database.	https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.notna.html https://www.w3schools.com/python/pandas/pandas_dataframes.asp https://towardsdatascience.com/an-easy-beginners-guide-to-sqlite-in-python-and-pandas-fbf1f38f6800	By utilizing all these resources, I was able to create my visualization.

Combined Analysis

We each worked on our databases, calculations, and visualizations separately for better group management. Hence, our 3 files can be run in any order, to obtain the final calculations and visualizations. Then, we compared our data. We put our data into a table for easy viewing before elaborating on it.

Final Comparisons:

	AVG Artwork duration (years)	AVG Start date (year date)	AVG End date (year date)
Art Institute of Chicago	19.12	1729	1749
Harvard Art Museum	15.96	1800	1816
MET	25.63	1813	1849

As of recently, the average number of years a painting took at the Art Institute of Chicago is 19.912 years. The average number of years a painting took for the Harvard Art Museum is 15.96 years. And lastly, the average number of years a painting took for the Metropolitan Museum of Art is 25.63 years. Comparing these 3 findings reveals that the Metropolitan Museum of Art has artworks that took the longest amount of time and the Harvard Art Museum has artworks that took the shortest amount of time.

Next, the average start year of pieces at the Art institute of Chicago is the year 1729, at the Harvard Art Museum is the year 1800, and lastly at the Metropolitan Museum Of Art is the year 1813. Comparing these 3 findings reveals that the Art Institute of Chicago has artworks that were started the earliest and the Metropolitan Museum Of Art has artworks that were started the latest. This is all based on averages with the pieces in our databases.

Next, the average end year of paintings at the Art institute of Chicago is 1749, at the Harvard Art Museum is 1816, and at the Metropolitan Museum Of Art is 1849 (in year dates). Comparing these 3 findings reveals that the Art institute of Chicago has artworks that were ended the earliest and Metropolitan Museum Of Art has artworks that were ended the latest. Again, this is all based on averages with the pieces in our databases.

Overall, we think our program is great at obtaining information on the 3 different museums we focused on. We were able to understand a great deal about the breadth and depths of the artworks in each museum by looking at their start and end dates. For future analysis we would take this program a step further by discussing the location from which these pieces come from, or the departments as well.