Zeinab Ghandour
Woojin Kang

GitHub Repo: https://github.com/zghandou/si206-finalproj.git

Report

1. The goals for your project:

Animal Crossing: New Horizons is a popular Nintendo game that allows users to create their own personalized island. On this island, characters called “Villagers” inhabit the user’s island. These villagers are all cartoonish animals with unique qualities. Our project aimed to extract these qualities from villagers to represent the diversity of characteristics on the island and the types of villagers that inhabit it. We wanted to collect data on some villagers related to their identity. Traits such as gender, species, and personality were some that we set out to find. We also wanted to collect the variety of styles available to villagers and the average price that it costs villagers to dress in that style.

2. The goals that were achieved:

We were able to collect information on 100 villagers including their hobbies (education, fashion, etc), their personality types(cranky, lazy, etc), their species, and their gender. We were able to find that most villagers on the island are cranky, which means that they might not be enjoying island life all that much! We also found that cats are the species most commonly found on the island. Additionally, we gathered data about different styles villagers could wear, the most popular clothing types, and the average cost of each clothing style. Through the clothing calculations, we were able to determine that villagers that want to look “cool” pay a lot more bells (game currency) in comparison to the other styles available to them in the game. We also found that the clothing style with the most number of items fell under the category “simple.” Overall, we were able to gain detailed information about the demographic of villagers, their personalities, and their styles.

3. The problems that you faced:

When creating tables in our database, we struggled with formatting nested dictionaries into rows. We ended up referencing lecture videos, past homework, and discussion assignments to understand proper indexing. We also struggled with organizing functions. The order of the functions affected how data was loaded into the tables. We also ran into issues while pushing and committing into GitHub while using a collaborative repository. Additionally, we realized we had to use different approaches to limiting information to less than 25 at a time for our differing

file functions and APIs. When it came to creating visualizations, we had trouble creating a pi chart representing the gender distribution percentages.

4. Calculations from the data in the database

Clothing_Style_Calculation

Clothing Style	total amount	average style cost
Active	17	244.23529411764700
Simple	28	341.85714285714300
Elegant	14	536.9285714285710
Cute	21	550.0952380952380
Cool	8	733.625
Normal	7	424.7142857142860
Gorgeous	5	526.0

Clothing_Style_Calculation.csv

```
Clothing Style,total amount,average style cost
Active,17,244.23529411764707
Simple,28,341.85714285714283
Elegant,14,536.9285714285714
Cute,21,550.0952380952381
Cool,8,733.625
Normal,7,424.7142857142857
Gorgeous,5,526.0
```

These are the Species and Personality calculations in the VS code write-out file and as a nicer table:

species.csv

```
Villager Species,Number of Villagers
Anteater,7
Bear,15
Bird,13
Bull,6
Cat,23
Cub,16
Chicken,9
Cow,4
Alligator,7
```

personality types.csv

```
Personality Type,Number of Villagers
Cranky,16
Jock,15
Peppy,14
Snooty,13
Normal,13
Smug,7
Lazy,16
Uchi,6
```

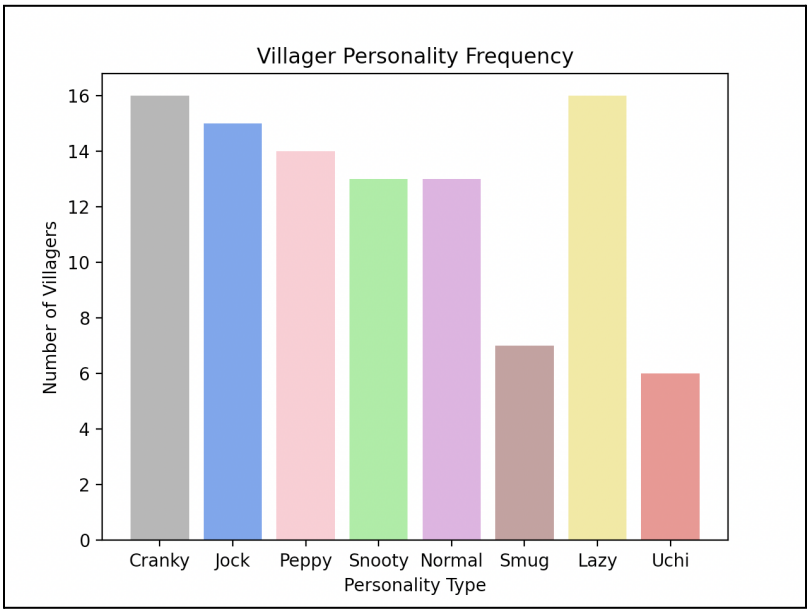
species

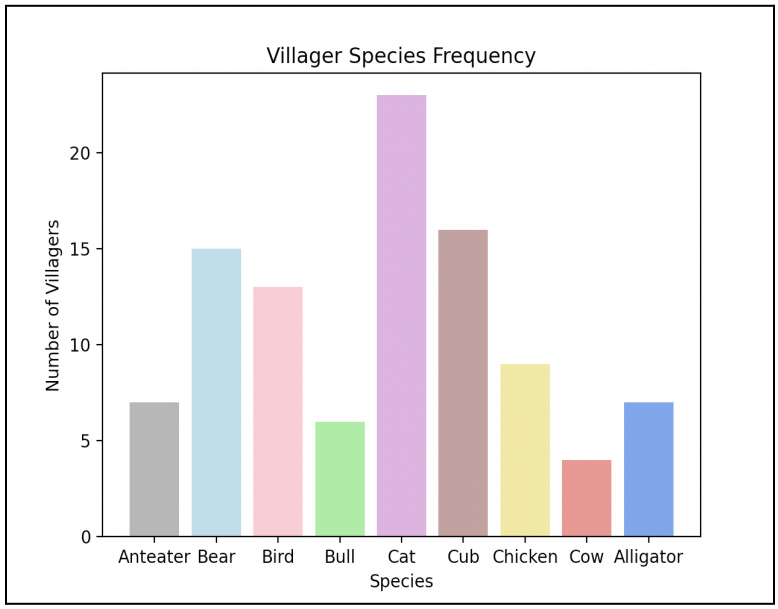
Villager Species	Number of Villagers
Anteater	7
Bear	15
Bird	13
Bull	6
Cat	23
Cub	16
Chicken	9
Cow	4
Alligator	7

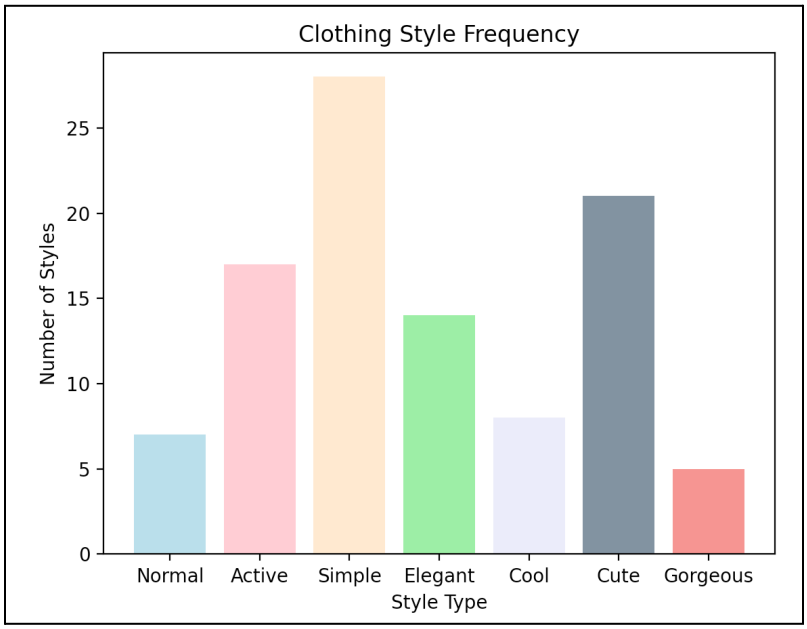
personality types

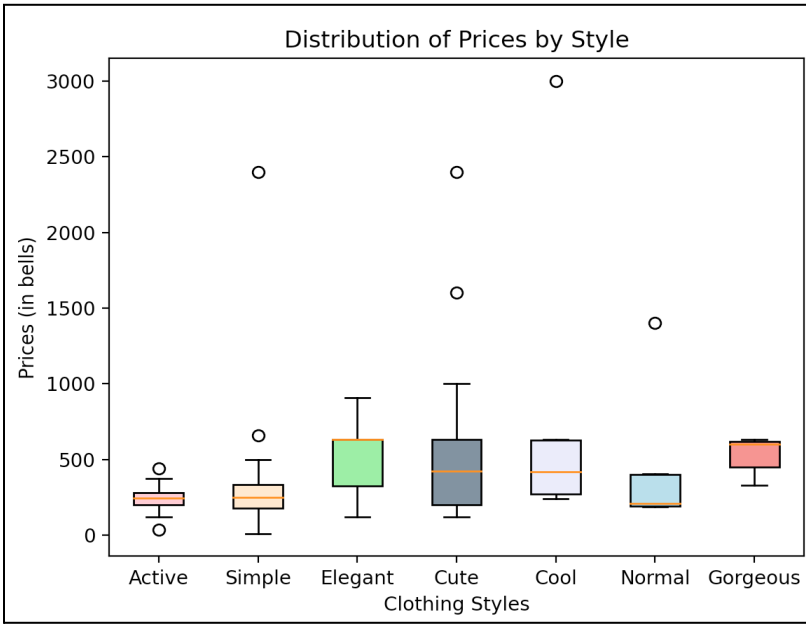
Personality Type	Number of Villagers
Cranky	16
Jock	15
Peppy	14
Snooty	13
Normal	13
Smug	7
Lazy	16
Uchi	6

5. The visualization that you created (i.e. screenshot or image file):









6. Instructions for running your code:

In order to run our code you must first download matplotlib and sqlite3. Then you must import JSON, os, requests, sqlite3, CSV, and matplotlib.pyplot as plt. You will then be able to run the nook.py file. Upon running the nook.py code, the terminal will prompt you to enter a start number. You will want to start at 0, so enter 0. The code will then ask for a stopping number. Enter any number below 25. This will control how much data is loaded into the 'Clothes' table of our 'island' database. To continue loading more data, run the nook.py file again. It will prompt you to enter another pair of start and stop numbers. You should continue from the number you left off. To retrieve the visualizations of the data, simply run the nook_calc.py file. For the acnh.py file, it should first be run 4 times to store the items in the database. After the acnh_calc.py should be run once which will generate the different graphs and the csv files.

7. Documentation for each function that you wrote. This includes the input and output for each function:

nook.py

setUpDatabase: creates a connection to our database – 'island'. It has db_name as an input and returns cur and conn.

make_style_table: creates foreign keys for the same styles within the 'Clothes' table. This will allow us to link the 'Styles' table and 'Clothes' table to compute calculations. This outputs a table.

clothes: grabs clothing data – style, price, and clothing item name from the nookipedia API. It then inserts that data into a table named 'Clothes'. This outputs a table.

main: calls in all the functions and connects functions together within the nook.py file by setting the outputs of functions to a variable. Those variables can then be passed into other functions. The main function also hosts the input prompts that limit how many items of data are loaded into the database at a time.

nook_calc.py

setUpDatabase: creates a connection to our database – 'island'. It has db_name as an input and returns cur and conn.

get_styles_and_prices: grabs clothing 'style' and clothing 'price' from the 'Clothes' table by using the foreign key – 'style_id' – we made in the function *make_style_table*. This outputs a list of tuples. The tuples include the clothing style and the price associated with the clothing item.

get_styles: keeps track of all the unique styles and the count of how many clothing pieces are associated with that style from the ‘Clothes’ table. This is accomplished by using the foreign key – ‘style_id’ – to link the ‘Styles’ table with the ‘Clothes’ table. As a result, the function outputs a dictionary with keys as the style and the value as the number of clothing items with that specific style.

style_avg_price: create two dictionaries. One dictionary calculates the average price of all clothes of a certain style. The second dictionary keeps track of the total amount of clothes with that specific style from the ‘Clothes’ table, the aggregate price of unique clothing styles within the ‘Clothes’ table, and a list of all individual prices of clothing items for each unique style found in the ‘Clothes’ table. As a result, the function outputs a dictionary with keys as the clothing style and its value as the average price of that unique style and a nested dictionary with keys as the clothing style with the value set to an inner dictionary.

write_out_averages: writes the style name, average style price, and total amount found of each style within the ‘Clothes’ table to a CSV file. To do this, it takes the nested dictionary from *style_avg_price* and uses indexing to access the count and total price values. Using these values, we calculated the average price. As a result, the output is a CSV file that includes all unique clothing styles, the average price of all clothing price of a specific style, and the total count of all clothing items of a respective clothing style.

box_plot: creates box plots of the price distribution of each style by using the dictionary of calculations outputted from the *style_avg_price* function.

styles_graph: creates a bar graph for the total amount found of each style within the ‘Clothes’ table.

acnh.py

def setUpDatabase: this functions makes a connection for our database file in SQLite. We used the same function that was given to us in homework assignments and discussion activities. It has db_name as an input and returns cur and conn.

def get_data: this function is used to make a request to the “ACNH” API and outputs all the information pertaining to each villager from the game. The input is a villager id that is

formatted at the end of the API URL and changes depending on the unique id for a villager (ex: id 7 = Olaf)

def create_villager_table: takes cur and conn in the parameters and uses SQL statements to create the overall Villager table that many other tables feed into. The output is a table with the corresponding column names.

def make_gender_table: this takes the data returned in the get_data function and outputs a table with foreign keys/ids for each villager gender in the game (0 = male, 1 = female).

def make_species_table: this takes in data from the API and outputs a table containing each species of villager along with its foreign key.

def make_p_table: this takes in the data from the get_data function and outputs a separate table with foreign keys for each personality type string.

def make_hobby_table: this takes in the data from the get_data function and outputs a separate table with foreign keys for each villager hobby string.

def add_villager: takes in information from the API used in the get_data function and outputs a table with all the villager names along with the values from the foreign key tables for species, personality, hobby, and gender. This is also the function that limits 25 or fewer items being added to the database (lines 120-122). I used SQL to insert the data I specified into its designated column within the villager table.

def main: the main function in the “acnh.py” file is used to call all of the functions written in the code above it, within this nothing would actually happen within the file. I set up the database and named it “island.db” in the main as well as made a for loop to limit the overall number of items in the database.

acnh_calc.py

def setUpDatabase: this function makes a connection for our database file in SQLite. We used the same function that was given to us in homework assignments and discussion activities. It has db_name as an input and returns cur and conn.

def calc_gender: this function calculates the number of villagers that are female or male. It outputs the count of female and male villagers. —> ([“Male”: 46, “Female”: 54])

def calc_personality: this function calculates the number of villagers within each personality type. It sets up an empty dictionary and makes a query to select all the personality types, goes through that and returns the villagers per personality type (returns the dictionary as well).

```
{'Cranky': 16, 'Jock': 15, 'Peppy': 14, 'Snooty': 13, 'Normal': 13, 'Smug': 7,
'Lazy': 16, 'Uchi': 6}
```

def calc_species: this function calculates the number of villagers per species. It sets up an empty dictionary and makes a query to select all the species, goes through that, and returns the villagers per species (returns the dictionary as well).

```
{'Anteater': 7, 'Bear': 15, 'Bird': 13, 'Bull': 6, 'Cat': 23, 'Cub': 16, 'Chick
en': 9, 'Cow': 4, 'Alligator': 7}
```

def write_data: takes in data from the calculations on species made in the calc_personality function and writes a csv file (output) with this information.

def write_data2: takes in data from the calculations on species made in the calc_species function and writes a csv file (output) with this information.

def write_data3: takes in data from the calculations on gender made in the cal_gender function and writes a csv file (output) with this information.

def personality_graph: this is the function that takes in the dictionary items from the calc_personality function and outputs a bar graph of the villagers per personality type. The personalities are keys and the number of villagers is the value of those keys.

def species_graph: this is the function that takes in the dictionary items from the calc_species function and outputs a bar graph of the villagers per animal species. The species are keys and the number of villagers is the value of those keys.

def main: the main function in the “acnh_calc.py” file is used to call all of the functions used to calculate information from our tables and output the visualizations. Without this nothing would actually happen within the file. I also called the functions used to write out to a CSV file in the main function. I set up the database and named it “island.db” in the main as well as

8. **Resource Documentation:**

Date: 12/12/2022

- Issue description: box plot was not showing up and issues with styling the plot
- Location of resource: https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.boxplot.html
- Result: we were able to successfully visualize the spread of clothing process for each unique clothing style

Date: 12/12/2022

- Issue description: the gender pi chart was not showing up
- Location of resource: https://matplotlib.org/stable/gallery/pie_and_polar_charts/pie_features.html
- Result: I was not able to use this to make my pi chart appear, the problem persisted so I created a different chart instead.

Date: 12/10/2022

- Issue description: trouble writing in csv file for calculations about clothing styles
- Location of resource: https://www.pythontutorial.net/python-basics/python-write-csv-file/
- Result: we were able to write in the csv file after figuring out proper indexing into tuples using dictionaries as input.

Date: 12/10/2022

- Issue description: Confusion surrounding how to use JOIN
- Location of resource: Lecture recording (11/22) and Discussion 12 assignment
- Result: we were able to write a join statement and understand when to use it.

Date: 12/9/2022

- Issue description: trouble creating a single database, previously we had two separate databases
- Location of resource: office hours
- Result: we were able to create one database called ‘island’. We figured out that only one person had to create the database and reference the database consistently across files.

Date: 12/8/2022

- Issue description: Was not able to make commits to the shared repo after a wrong merge was made.
- Location of resource: Office Hours
- Result: Was able to work past this and reversed the error that prevented us from pushing/pulling code.

Date: 12/8/2022

- Issue description: 25 item limit wasn't working
- Location of resource: Office Hours
- Result: Was able to modify existing code with gsi to make my limit work within my database.

Date: 11/26/2022
- Issue description: Needed guidance for using SQL commands and setting up databases with data inserted.
- Location of resource: Homework #7, Also peer tutoring session to help me understand how all of it works together.
- Result: Was able to use/modify what I did in homework 7 with the pokemon database to create my own tables.

Date: 11/20/2022
- Issue description: Creating function correctly with API url and parameters, also not being able to output the API's data.
- Location of resource: Homework #6
- Result: Was able to use/modify what I did in homework 6 to use the API I chose and was able to see all the data.

Date: 11/15/2022
- Issue description: Needed two APIs to pull data from the game "Animal Crossing"
- Location of resource: https://acnhapi.com/doc and https://api.nookipedia.com/
- Result: Was able to find two APIs that worked for out project goals.

Date: 11/15/2022
- Issue description: understanding how to read in an API using a special access key.
- Location of resource: Lecture 17 – JSON and APIs and https://www.askpython.com/python/examples/pull-data-from-an-api
- Result: I was able to read in a file. I had to use parameters and insert the API key as a one of the headers.