STAR
WARS

Exploring & Analyzing SWAPI Data

presented by The Death Star Group

Blair Sonnen, Jonathan Hays, Desmond Davis, Jesse Parent, Margarita Aynagoz

Executive Summary

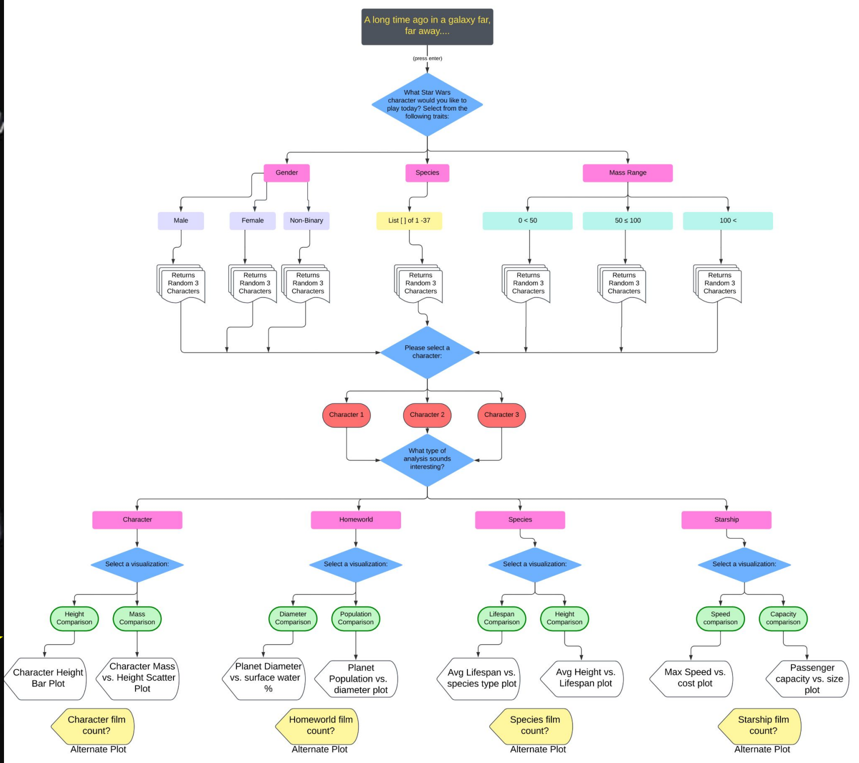
- ❑ Our Data Source: **Star Wars API**
  - Focus on pop culture, something fun and relatable.
  - API: *https://swapi.dev/api/*
  - ***Planets, Spaceships, Vehicles, People, Films and Species data***
- ❑ Our Exploration:
  - Assessing library data hypothesizing correlations.
  - Concluded with the realizations that:
    - Couldn’t do time-series forecasting.
    - Exploring the data was engaging.
    - We could improve user experience.
- ❑ Our Project:
  - An interactive application for API data exploration!

The Journey

- ❑ Let's do something fun!
  - Wait h/o…is this going to meet the project objectives?

- ❑ Questions we asked:
  - How to make this dataset *usable*?
  - How do we compile?
  - What is *even possible* to compare?
  - How can we *do this dynamically*?
  - What is relatable?
  - How *can we limit the data*?
  - Why make that choice –
    - Have a user *choose their own journey*?
    - The user can then *determine what's interesting to analyze.*
    - We create the application & analysis procedures.

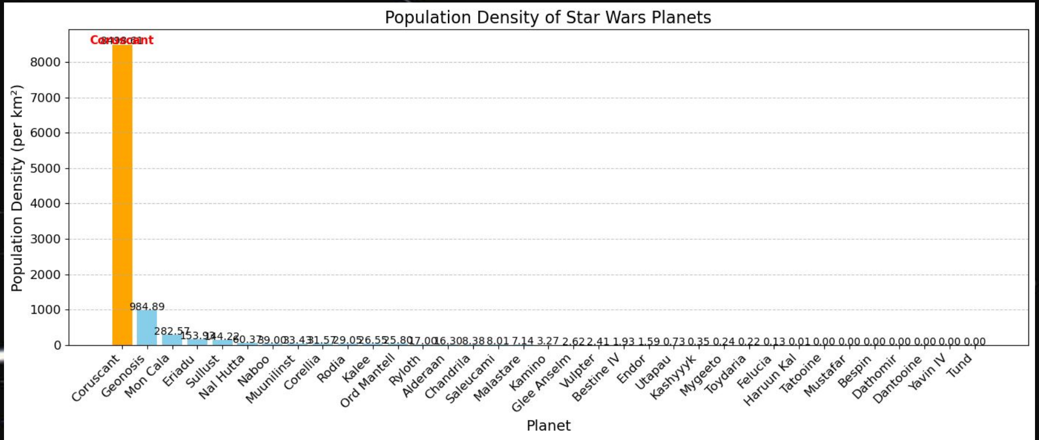
- ❑ Started with a workflow

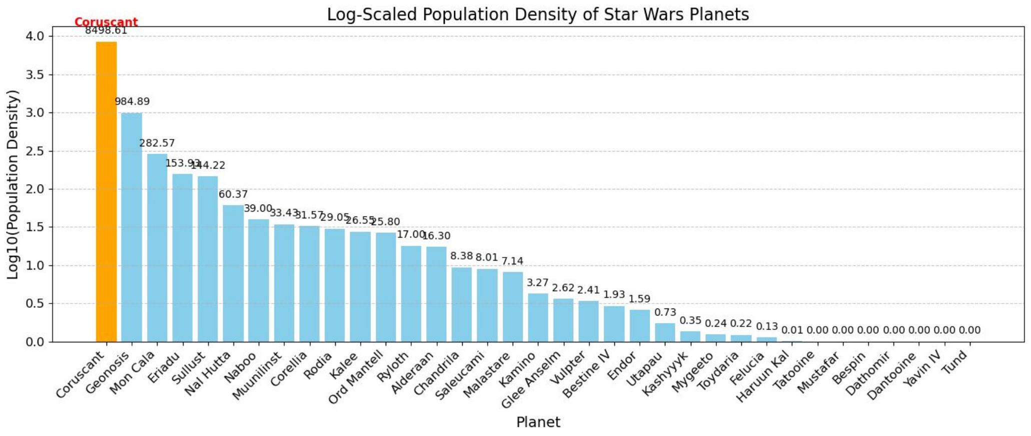

A long time ago in a galaxy far, far away....
(press enter)
What Star Wars character would you like to play today? Select from the following traits:
Gender
Species
Mass Range
Male
Female
Non-Binary
List [ ] of 1 -37
0 < 50
50 ≤ 100
100 <
Returns Random 3 Characters
Returns Random 3 Characters
Returns Random 3 Characters
Returns Random 3 Characters
Returns Random 3 Characters
Returns Random 3 Characters
Returns Random 3 Characters
Please select a character:
Character 1
Character 2
Character 3
What type of analysis sounds interesting?
Character
Homeworld
Species
Starship
Select a visualization:
Select a visualization:
Select a visualization:
Select a visualization:
Height Comparison
Mass Comparison
Diameter Comparison
Population Comparison
Lifespan Comparison
Height Comparison
Speed comparison
Capacity comparison
Character Height Bar Plot
Character Mass vs. Height Scatter Plot
Planet Diameter vs. surface water %
Planet Population vs. diameter plot
Avg Lifespan vs. species type plot
Avg Height vs. Lifespan plot
Max Speed vs. cost plot
Passenger capacity vs. size plot
Character film count?
Alternate Plot
Homeworld film count?
Alternate Plot
Species film count?
Alternate Plot
Starship film count?
Alternate Plot

Data Collection & Cleanup

- Data Source: Data pulled from API calls to URLS for Planets, Spaceships, Vehicles, People, Films and Species
  - Multiple pages of discrete data pages required specialized handling
  - A function was created to *fetch data from URLs asynchronously for time reduction*
  - Need to explode the arrays to get the data in a usable format
- Cleanup functions performed:
  - All data was type str.
  - Gender data needed to categorize all non-binary entries to common group.
  - Height and mass entries needed to be converted to int.
  - Character mass needed to be classified into user-friendly groupings.
  - Species data was inconsistent at best, and ranged from empty arrays to arrays with multiple URLs to bad entries.
    - Ultimately required parsing into “species classification.”
  - Character information for homeworld, species, starships, and vehicles needed to be exploded to become useful in dataframes .

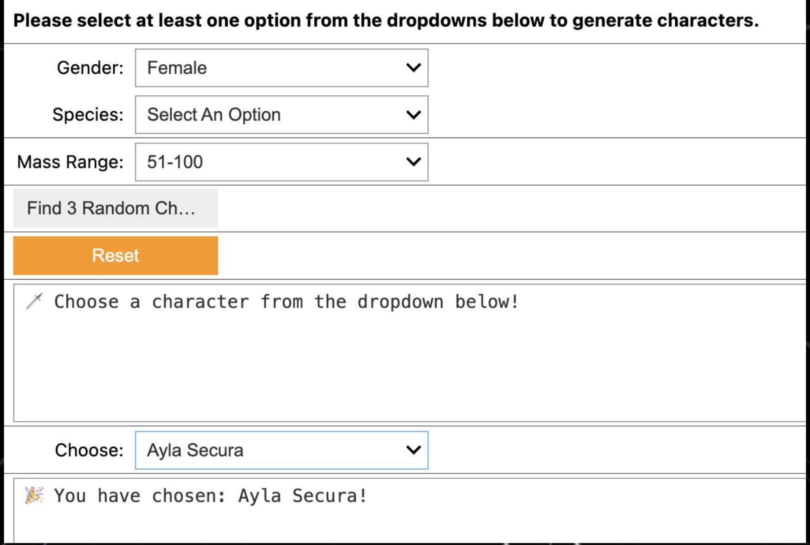
Visualizations





Approach & Methods

- ❑ Our approach required use of ipywidgets, IPython.display,for dataframe filtering on-the-fly.
  - ❑ In addition to standard standard tools like pandas, numpy, matplotlib.pyplot, requests,etc.
  - ❑ These tools helped create a visual and interactive user interface to filter our dataframes.



Please select at least one option from the dropdowns below to generate characters.

Gender: Female

Species: Select An Option

Mass Range: 51-100

```
Choose a character from the dropdown below!
```

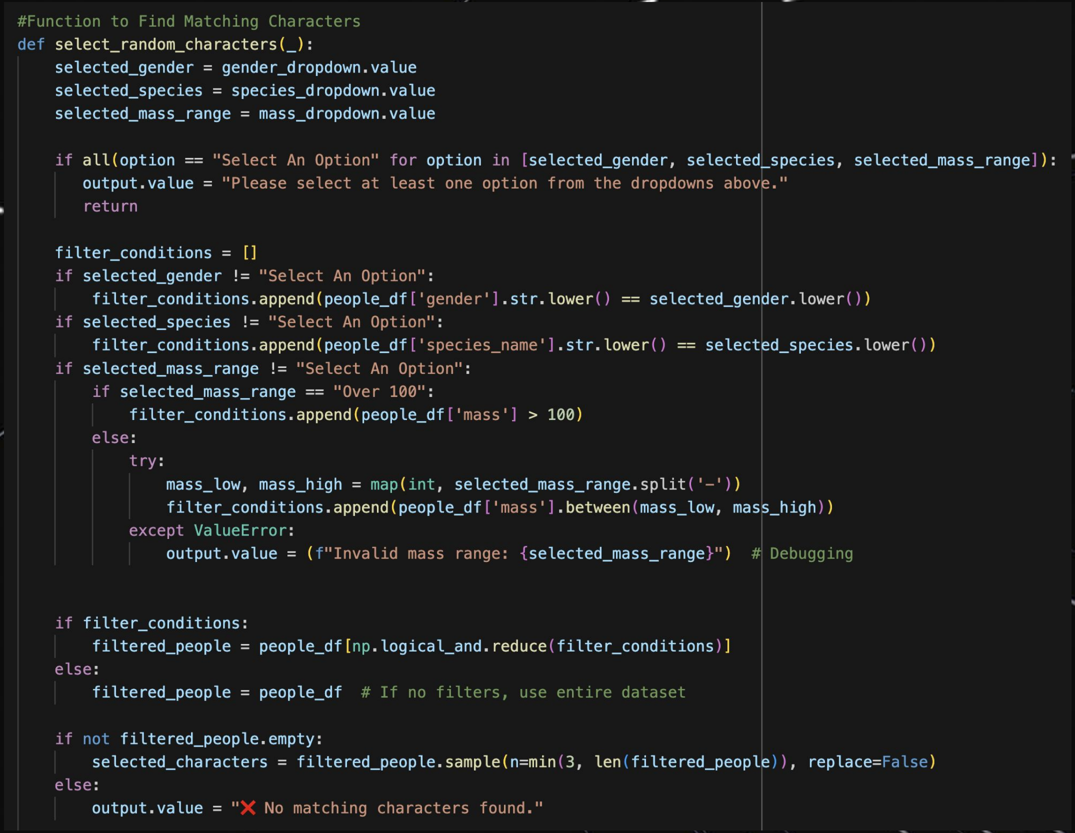
Choose: Ayla Secura

```
🎉 You have chosen: Ayla Secura!
```

- ❑ The user is free to select one filter to generate 3 random character options, or proceed to drill down with multiple filters in series.

Approach & Methods

- ❑ Guided Input Collection
- ❑ Ensuring Smooth Player Experience
- ❑ Apply Simple Filters
- ❑ Seamless Data Selection
- ❑ Keep Interaction Engaging



```python
#Function to Find Matching Characters
def select_random_characters(_):
    selected_gender = gender_dropdown.value
    selected_species = species_dropdown.value
    selected_mass_range = mass_dropdown.value

    if all(option == "Select An Option" for option in [selected_gender, selected_species, selected_mass_range]):
        output.value = "Please select at least one option from the dropdowns above."
        return

    filter_conditions = []
    if selected_gender != "Select An Option":
        filter_conditions.append(people_df['gender'].str.lower() == selected_gender.lower())
    if selected_species != "Select An Option":
        filter_conditions.append(people_df['species_name'].str.lower() == selected_species.lower())
    if selected_mass_range != "Select An Option":
        if selected_mass_range == "Over 100":
            filter_conditions.append(people_df['mass'] > 100)
        else:
            try:
                mass_low, mass_high = map(int, selected_mass_range.split('-'))
                filter_conditions.append(people_df['mass'].between(mass_low, mass_high))
            except ValueError:
                output.value = (f"Invalid mass range: {selected_mass_range}")  # Debugging


    if filter_conditions:
        filtered_people = people_df[np.logical_and.reduce(filter_conditions)]
    else:
        filtered_people = people_df  # If no filters, use entire dataset

    if not filtered_people.empty:
        selected_characters = filtered_people.sample(n=min(3, len(filtered_people)), replace=False)
    else:
        output.value = "❌ No matching characters found."
```

Key Code Snippets

Page Counts:

- SWAPI.dev APIs would not return all records at once
- Make initial calls to get total records

total_pages = int(total_records/10) + (1 if total_records%10 > 0 else 0)

Download All Pages:

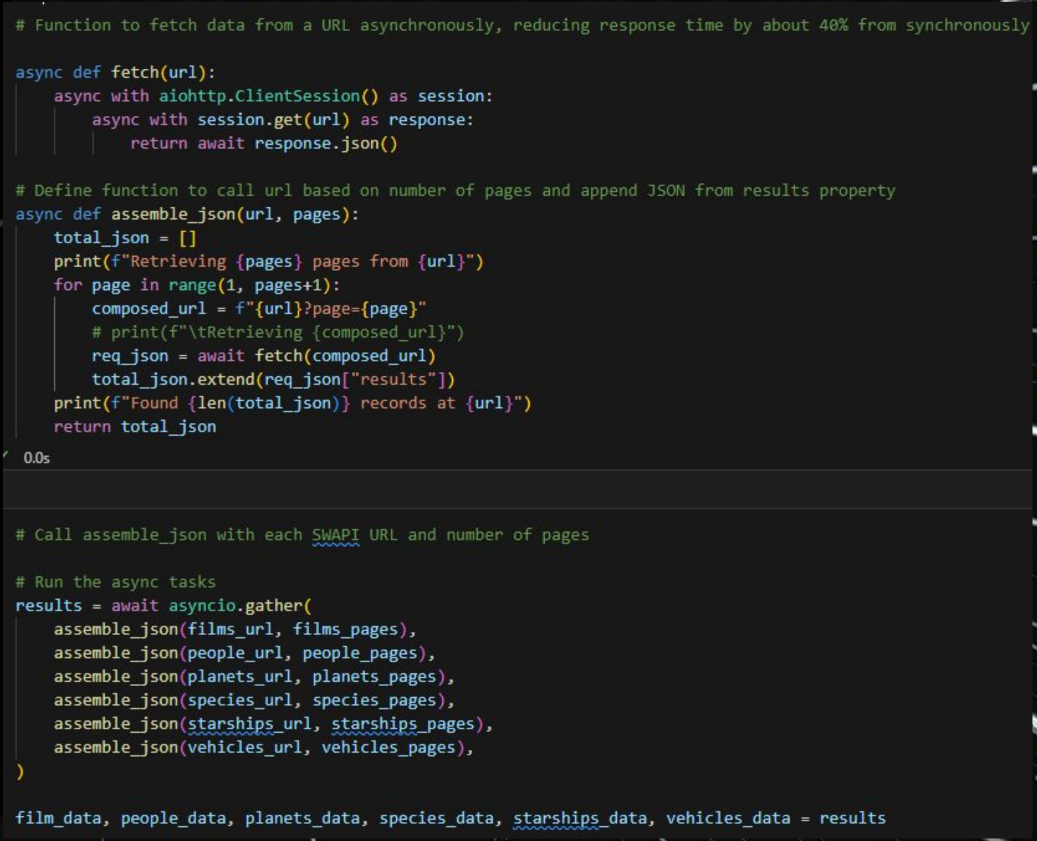
- Asynchronously request all APIs simultaneously
- Await each each page request in order
- Combine JSON from each response

Create Dataframes from JSON:

- Key information in “data’ attribute
- Display first 5 rows of each dataframe to examine structure

Cleanup Data:

- Consolidation of attributes
- Missing data filled in



```
# Function to fetch data from a URL asynchronously, reducing response time by about 40% from synchronously

async def fetch(url):
    async with aiohttp.ClientSession() as session:
        async with session.get(url) as response:
            return await response.json()

# Define function to call url based on number of pages and append JSON from results property
async def assemble_json(url, pages):
    total_json = []
    print(f"Retrieving {pages} pages from {url}")
    for page in range(1, pages+1):
        composed_url = f"{url}?page={page}"
        # print(f"\tRetrieving {composed_url}")
        req_json = await fetch(composed_url)
        total_json.extend(req_json["results"])
    print(f"Found {len(total_json)} records at {url}")
    return total_json
```

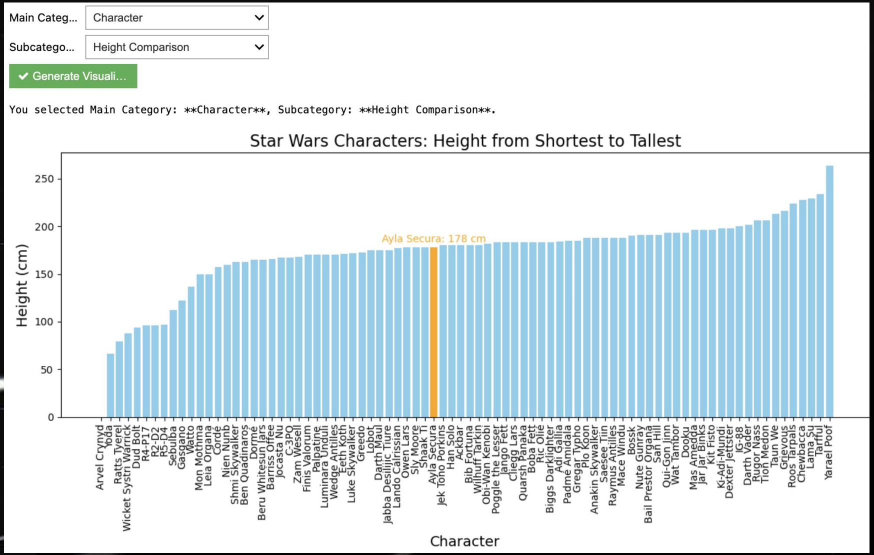
0.0s

```
# Call assemble_json with each SWAPI URL and number of pages

# Run the async tasks
results = await asyncio.gather(
    assemble_json(films_url, films_pages),
    assemble_json(people_url, people_pages),
    assemble_json(planets_url, planets_pages),
    assemble_json(species_url, species_pages),
    assemble_json(starships_url, starships_pages),
    assemble_json(vehicles_url, vehicles_pages),
)

film_data, people_data, planets_data, species_data, starships_data, vehicles_data = results
```

DEMO

Show Interactive UI with Visualizations

Dynamic visualizations



Main Categ... Character

Subcatego... Height Comparison

```
You selected Main Category: **Character**, Subcategory: **Height Comparison**.
```

Star Wars Characters: Height from Shortest to Tallest

- Users can filter, sort, and manipulate data to gain tailored insights.
- Interactive elements keep users engaged and encourage deeper exploration of data.

- Visual represented the variety of species in the dataset.
- Presented the analysis of species distribution revealing the diversity among characters in the universe.
- A comparison of character attributes among different species shows variability in height, mass, and affiliation.
- Individual character statistics (like mass, height, and age) provide insights into character design and powers.
- Data on character affiliations with starships and vehicles helped identify patterns in transportation.

Results & Findings

- **Improved Data Usability** – By transforming raw data into structured formats, we created a more accessible and meaningful dataset for analysis.
- **Enhanced Visualization Capabilities** – The exploded data allowed for deeper insights into character relationships, species distributions, and vehicle associations.
- **Better Data Classification** – Cleaning and categorizing inconsistent species and character attributes led to more precise and useful visual representations.

Challenges Faced During the Project:

- **API Integration Challenges** – Two different APIs were found and used, requiring careful comparison of data structures and formats each gave.
- **GitHub Collaboration Issues** – Merging code from different contributors led to version control conflicts that required manual resolution.
- **Data Inconsistencies** – Some datasets contained missing, duplicated, or poorly formatted data that needed extensive cleanup.
- **Combining Code** – Different coding styles and implementations had to be unified into a single connected framework.

Next Steps

- Next Steps: Further analysis on character relationships, species evolution.
- Add data from other sources like Wookiepedia to flesh it out more
- If we had more time, we'd love to incorporate AI & Machine Learning additions like a chatbot that provides Star Wars Trivia, lore, or character background upon selection.

Questions and feedback? Let's go!