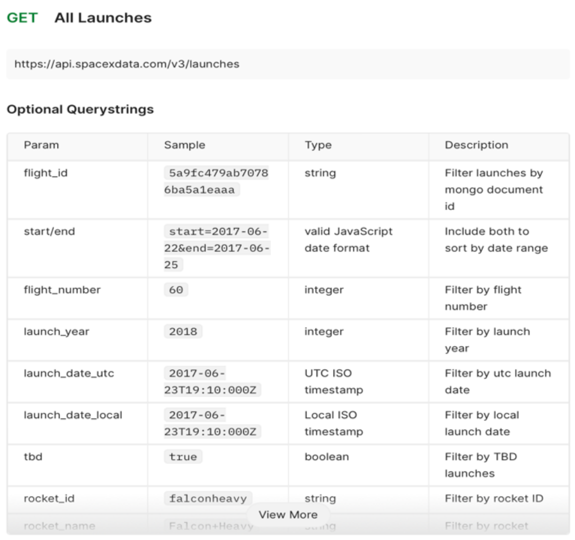
Rouxan Potgieter 231013
DV200 API Research

SpaceX API | Openweathermap API | Last.fm API

SpaceX API is an open-source REST API that provides data on SpaceX rockets, launches, capsules, launchpads, etc.

Dataset Characteristics:
It provides data on capsules, cores, history, info (about SpaceX), landing and launching pads, launches, missions, payloads, rockets, and ships. Each of these can be considered their own object with its own attributes, for example, here's some of a launch's attributes:

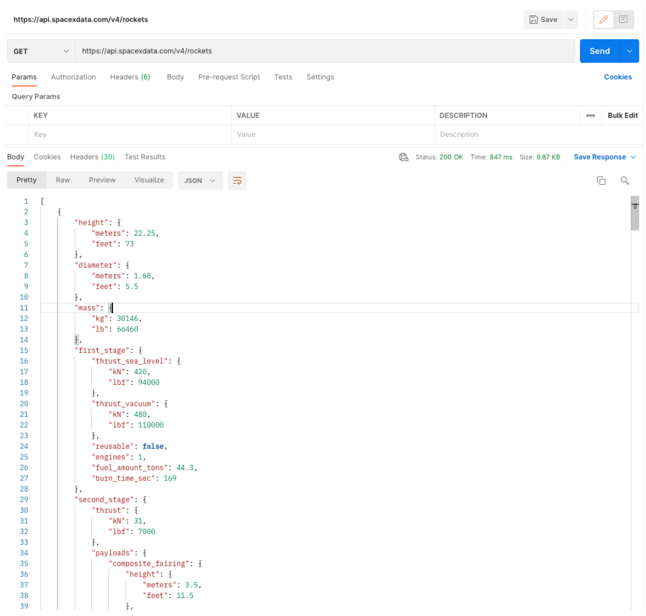

GET All Launches

https://api.spacexdata.com/v3/launches

Optional Querystrings

Param	Sample	Type	Description
flight_id	`5a9fc479ab70786ba5a1eaaa`	string	Filter launches by mongo document id
start/end	`start=2017-06-22&end=2017-06-25`	valid JavaScript date format	Include both to sort by date range
flight_number	`60`	integer	Filter by flight number
launch_year	`2018`	integer	Filter by launch year
launch_date_utc	`2017-06-23T19:10:000Z`	UTC ISO timestamp	Filter by utc launch date
launch_date_local	`2017-06-23T19:10:000Z`	Local ISO timestamp	Filter by local launch date
tbd	`true`	boolean	Filter by TBD launches
rocket_id	`falconheavy`	string	Filter by rocket ID
rocket_name	`Falcon+Heavy`	string	Filter by rocket

Similarly, here's an extract of a rocket object's attributes (and proof that I have access to the API – screenshot of postman):



https://api.spacexdata.com/v4/rockets

GET https://api.spacexdata.com/v4/rockets

Status: 200 OK Time: 847 ms Size: 8.87 KB

```
[
    {
        "height": {
            "meters": 22.25,
            "feet": 73
        },
        "diameter": {
            "meters": 1.68,
            "feet": 5.5
        },
        "mass": {
            "kg": 30146,
            "lb": 66460
        },
        "first_stage": {
            "thrust_sea_level": {
                "kN": 420,
                "lbf": 94000
            },
            "thrust_vacuum": {
                "kN": 480,
                "lbf": 110000
            },
            "reusable": false,
            "engines": 1,
            "fuel_amount_tons": 44.3,
            "burn_time_sec": 169
        },
        "second_stage": {
            "thrust": {
                "kN": 31,
                "lbf": 7000
            },
            "payloads": {
                "composite_fairing": {
                    "height": {
                        "meters": 3.5,
                        "feet": 11.5
                    },
```

Rockets	**Launches**
• id • active • stages • boosters • cost_per_launch • success_rate_pct • first_flight • country • company • height (object: meters, feet) • diameter (object: meters, feet) • mass (object: kg, lb) • payload_weights (array of objects: id, name, kg, lb) • first_stage (object: reusable, engines, fuel_amount_tons, burn_time_sec, thrust_sea_level, thrust_vacuum) • second_stage (object: engines, fuel_amount_tons, burn_time_sec, thrust, payloads) • engines (object: number, type, version, layout, engine_loss_max, propellant_1, propellant_2, thrust_sea_level, thrust_vacuum, thrust_to_weight) • landing_legs (object: number, material) • wikipedia • description • rocket_id • rocket_name • rocket_type	• id • mission_name • launch_year • launch_date_utc • launch_success • rocket (object containing rocket details like rocket_id, rocket_name) • launch_site (object containing site details like site_name_long) • payloads (array containing payload details) • links (object containing media links like video_link, mission_patch) • details
Capsules	**Dragons (spacecraft)**
• id • capsule_serial • status	• d • name • type

• original_launch • missions (array of mission names and flight numbers) • landings • type • details	• active • crew_capacity • orbit_duration_yr • dry_mass_kg • dry_mass_lb • first_flight • heat_shield (object containing heat shield details) • thrusters (array of thruster details) • launch_payload_mass • launch_payload_vol • return_payload_mass • return_payload_vol • flickr_images • wikipedia • description
Landing Pads	**Missions**
• id • full_name • status • location (object containing location details like name, region, latitude, longitude) • landing_type • attempted_landings • successful_landings • wikipedia • details	• mission_name • mission_id • manufacturers (array of strings) • payload_ids (array of strings) • wikipedia • website • twitter • description • objectives (array of strings or objects detailing specific mission objectives) • launches (array of launch ids associated with the mission) • success_rate_pct • partners (array of strings or objects detailing partner organizations)
History	
• id • title • event_date_utc • event_date_unix • flight_number (optional, if applicable) • details • links (object: article, wikipedia, reddit; links to external sources providing more information) • location (object: optional, may include name, region, latitude, longitude if applicable to the historical event)	

Outline of comparative data:
For Rocket vs. Rocket Comparison:

- active: Indicates whether the rocket is currently active, allowing comparison of operational versus retired rockets.

- stages: The number of stages in the rocket can be compared to understand complexity and mission capability.
- boosters: Number of boosters used, which affects lift capacity and mission scope.
- cost_per_launch: Provides a financial perspective, comparing the investment required for each rocket launch.
- success_rate_pct: Offers insight into the reliability and success history of each rocket.
- first_flight: The date of the first flight allows comparison of rocket age and technological evolution.
- country: Country of origin can be interesting for comparing rockets from different space agencies or companies.
- company: The manufacturing company, useful for comparing rockets developed by different entities.
- height, diameter, mass: Physical dimensions and mass can be compared to understand size and build differences.
- payload_weights: Maximum payload capacities to various orbits (e.g., LEO, GTO) provide a direct comparison of lift capabilities.
- engines.number: The number of engines can be compared to understand thrust and power differences.

For Launch vs. Launch Comparison:

- mission_name: Identifies the mission, useful for thematic or objective comparisons.
- launch_date_utc: The date and time of launch allow for chronological comparisons and understanding temporal trends.
- launch_success: Indicates whether the launch was successful, allowing for comparison of mission outcomes.
- rocket.rocket_id, rocket.rocket_name: Identifies the rocket used, useful for comparing launches using the same or different rockets.
- payloads: (Nested within launch data, if available) Allows comparison of payload types, destinations (e.g., orbits), and capacities.
- launch_site.site_name_long: The launch site can be compared to understand geographic launch distribution and site-specific success rates.
- cost_per_launch (If available at the launch level, or inferred from the rocket data): Comparing the cost of different launches can provide financial insights.
- flight_number: Provides a sequence for comparison, useful for analyzing progress or changes over time.
- launch_year: Year of the launch, useful for annual comparisons and trend analysis.

Outline of Timeline Data:
Launches Per Pad Over Time

- launch_date_utc: The UTC date and time of the launch to determine when each launch occurred.

- launch_site.site_id or launch_site.site_name_long: Identifies the launch pad used for each launch. This allows you to group launches by location and plot them over time.

Successful/Failed Launches Over Time

- launch_date_utc: Essential for placing each launch on the timeline.
- launch_success: Indicates whether a launch was successful. By plotting this attribute over time, you can visualize trends in launch success rates.

And for successful/failed launches per pad:

- launch_site.site_id or launch_site.site_name_long: To segment the data by launch pad in addition to the success/failure status.

Rockets Launched Over Time

- launch_date_utc: As with the other visualizations, this places each launch on the timeline.
- rocket.rocket_id or rocket.rocket_name: Allows you to identify which rocket was used for each launch. This way, you can visualize the frequency and distribution of rocket types over time.

Payload Mass to Various Orbits Over Time

This visualization can show how the capability of SpaceX rockets to carry payloads has evolved, indicating technological advancements and the increasing scope of missions over time.

- launch_date_utc: To determine when each launch occurred, allowing you to plot this data over time.
- rocket.rocket_id or rocket.rocket_name: To identify the rocket used for each launch. This is useful if you want to differentiate capabilities by rocket type.
- payloads.payload_type: To categorize the payload (e.g., satellite, Dragon spacecraft, etc.).
- payloads.orbit: Identifies the orbit destination of the payload (e.g., LEO, GTO, ISS, etc.), allowing you to segment the data based on orbit type.
- payloads.payload_mass_kg or payloads.payload_mass_lbs: The mass of the payload, which is the key data point for this visualization.

Launch Frequency by Year

This visualization can provide insights into SpaceX's operational tempo, showing how the number of launches has changed year by year, reflecting the company's growth and the increasing demand for space launches.

- launch_date_utc: To extract the year of each launch and aggregate launch counts by year.
- launch_success: Optionally, you can differentiate between successful and unsuccessful launches within each year.
- rocket.rocket_id or rocket.rocket_name: To segment the data by rocket type, if you wish to show how the launch frequency of different rockets has evolved.

The OpenWeatherMap API

Provides comprehensive weather data, which can be utilised for various visualisations, including comparative and timeline analyses. Below is a brief compilation of key objects and their attributes provided by the API, followed by outlines for utilising this data in comparative and timeline visualizations.

Key Objects and Attributes

Current Weather Data Object

- coord: { lon, lat } - Longitude and latitude of the location.
- weather: Array of objects containing id, main (weather condition), description, icon.
- base: Internal parameter.
- main: Contains temp, feels_like, temp_min, temp_max, pressure, humidity.
- visibility: Visibility distance in meters.
- wind: Contains speed, deg (direction), and sometimes gust.
- clouds: { all } - Cloudiness percentage.
- dt: Time of data calculation, UTC.
- sys: Contains country, sunrise, and sunset times.
- timezone: Shift in seconds from UTC.
- id: City ID.
- name: City name.
- cod: Internal parameter.

Forecast Weather Data Object

- city: Contains id, name, coord, country, population, timezone, sunrise, sunset.
- cod, message, cnt: Internal parameters or metadata.
- list: Array of forecast data objects, each containing:
- dt: Time of forecasted data, UTC.
- main: Similar to current weather's main, with forecasted temperature, pressure, etc.
- weather: Array of weather condition objects.
- clouds, wind, visibility, pop: Forecasted cloudiness, wind data, visibility, and probability of precipitation.
- sys: Contains pod (part of the day).
- dt_txt: Forecasted data time in text.

Comparative Data Outline

You can compare current weather conditions (temp, humidity, pressure, wind.speed) across multiple locations to visualize climate diversity. Comparing forecast data (list.main.temp, list.weather.main, list.wind.speed) between cities over a short term (e.g., 5 days) can reveal expected weather patterns, preparing viewers for upcoming conditions. Additionally, comparing historical weather data, if available, can show how weather conditions have changed over time in different regions, highlighting climate change effects or seasonal variations.

Timeline Data Outline
Utilizing the forecast and historical weather data from OpenWeatherMap, you can create timeline visualizations that depict how weather conditions (e.g., temp, rain, snow, wind.speed) evolve over time within a specific location. For instance, plotting temperature (main.temp) and precipitation (rain, snow) forecasts over a week can illustrate expected weather trends, aiding in planning and preparedness. Similarly, historical weather data can be used to construct timelines showing temperature or precipitation changes over months or years, offering insights into climate patterns and anomalies. This approach is invaluable for environmental studies, agricultural planning, or simply for individuals interested in weather trends.

Last.fm API
Offers data related to music tracks, artists, albums, and user listening habits, making it ideal for creating engaging visualizations around music preferences and trends. Here's a brief overview of key objects and their attributes, followed by outlines for comparative and timeline data utilization.

Key Objects and Attributes
Artist Object
- name: Artist's name.
- mbid: MusicBrainz Identifier.
- url: URL to the Last.fm artist page.
- image: Array of image URLs in various sizes.
- streamable: Indicates if the artist's music is available for streaming.
- listeners: Number of listeners.
- playcount: Total plays across all tracks.

Album Object
- artist: Artist name or object containing artist details.
- name: Album name.
- mbid: MusicBrainz Identifier.
- url: URL to the Last.fm album page.
- image: Array of image URLs in various sizes.
- listeners: Number of listeners.
- playcount: Total album plays.

Track Object
- name: Track name.
- mbid: MusicBrainz Identifier.
- url: URL to the Last.fm track page.
- duration: Track duration in milliseconds.
- streamable: Indicates if the track is available for streaming.
- listeners: Number of listeners.
- playcount: Total track plays.
- artist: Artist name or object containing artist details.

- album: Album name or object containing album details (if applicable).

Comparative Data Outline
The Last.fm API's rich dataset supports a wide range of comparative visualizations. For instance, you can compare artists by their listeners and playcount to visualize relative popularity. Albums and tracks can also be compared within or across artists based on playcount, listeners, or other metrics like duration for tracks. This data can highlight popular music trends, artist dominance in certain genres, or the impact of specific albums or tracks on an artist's popularity. Moreover, comparing the tag data (genre tags) associated with different tracks or artists can offer insights into the music genre landscape and how it shifts across different artists or over time.

Timeline Data Outline
For timeline visualizations, the Last.fm API facilitates tracking the popularity of artists, albums, or tracks over time. While Last.fm's API doesn't directly provide time-series data, you can infer trends over time by periodically fetching playcount and listeners data for artists, albums, or tracks and storing this data with timestamps. Over time, this collected data can reveal trends, such as an artist's growing popularity, seasonal variations in track listens, or the long-term impact of an album release on an artist's playcounts. The API's terms of use does allow for this (because I am using it for academic/educational purposes).