HSLU Hochschule Luzern

Comparing Skies

A Cross-Planetary Study of Earth's and Mars' Atmospheric Patterns

W.DWL03 DATA WAREHOUSE AND DATA LAKE SYSTEMS

AESM3003

Authors:

Joana Duarte —– joana.duartedossantos@stud.hslu.ch —– 23-576-309
Ansam Zedan —– ansam.zedan@stud.hslu.ch —– 22-888-200
Albin Plathottathil —– albin.plathottathil@stud.hslu.ch —– 23-576-143

GitHub Repository

June 15, 2024

Contents

1 Introduction

1.1 Background & Motivation

Earth has been the stage of multiple extreme weather phenomena, especially in the last decades[1]. Extreme temperatures are becoming more and more frequent, along with heavy rains that cause landslides, damage to infrastructure and disrupt transports, droughts that lead to seawater intrusion and harm to the ecosystems, and even tornadoes that cause destruction wherever they go.

The South region of Germany has been recently hit hard by torrential rains that have led thousands to evacuate [2]. Countries like the US observe over 1,150 tornadoes on average every year, which is more than Canada, Australia and all European countries combined [3], while at the same time have states where severe drought is leading to drastic measures for the upcoming Summer [4].

This situation is not exclusive to these areas. The entire world is suffering, one way or another with the consequences of the main issue: Global Warming.

Weather records have been kept since 1880. According to the Natural Resources Defense Council (NRDC), in the first 100 years of the records, the temperature increased 0.07ºC but since 1980 this records show that the temperature has been increasing at a rate of 0.18ºC per decade. While this may seem such a small number, this 0.18ºC per decade has led to record breaking temperatures in over 140 years of data, with the 5 warmest years being registered after 2015 [5].

Many researchers have debated about when this continuous rise will become irreversible, but it is not an easy task to make such prediction. But it is increasingly clear that at this rhythm, this tipping point of no return may be dangerously close [6] and it is crucial to explore all avenues for understanding and mitigating these phenomena.

Due to its proximity to Earth, scientists and space enthusiasts have kept their telescopes pointed at Nergal, the great hero, as the Babylonians called the “wandering star” we call Mars. Parallel studies on Mars reveal intriguing climatic dynamics, including significant dust storms and temperature fluctuations that mirror some of Earth’s own environmental challenges.

1.2 Problem Definition

Based on the premise that these climate changes may transform Earth into a planet that is inhabitable for humans and other beings that cannot withstand the extreme events, understanding extraterrestrial climates can provide new insights into our own planetary weather changes and sustainability.

More than looking for an alternative habitat, we are interested in knowing if we can use Mars’ weather data to better understand the principles of atmospheric dynamics, including wind patterns, storm formation, and temperature fluctuations.

Despite Mars’ thin atmosphere composed mainly of carbon dioxide, with surface pressures much lower than Earth’s, studying how weather phenomena occur in such an environment helps scientists refine their models of atmospheric behaviour, which can be applied to better predict and understand weather patterns on Earth, particularly in extreme or changing conditions.

[1] https://wmo.int/topics/extreme-weather

[2] https://www.theguardian.com/world/article/2024/jun/03/two-killed-in-floods-and-scores-missing-in-southern-germany

[3] https://edition.cnn.com/weather/us-leads-tornado-numbers-tornado-alley-xpn

[4] https://www.seattletimes.com/seattle-news/climate-lab/wa-drought-has-already-led-some-to-shut-off-water-to-farmers/

[5] https://www.nrdc.org/stories/global-warming-101#warming

[6] https://www.sgr.org.uk/resources/point-no-return-how-close-world-irreversible-climate-change

1.3 Research Questions

To address the complex issue of climate change and its potential parallels with Martian atmospheric dynamics, it is essential to define specific research questions that guided this study. These questions aim to uncover the potential benefits of studying extraterrestrial climates for enhancing our understanding of Earth's weather patterns. The following research questions have been formulated to direct the investigation:

- What similarities and differences in patterns can be identified between Mars and Earth, and how can these insights improve our models of atmospheric behaviour on Earth?
- Can weather data from Mars be utilised to enhance our understanding of atmospheric dynamics on Earth, thereby improving our ability to predict and mitigate extreme weather events?
- How can the integration of weather data from Mars and Earth in a Data Warehouse and Data Lake system enhance our ability to identify and analyse underlying climate trends and interactions?

2 State of the Art

Studies have shown that the Earth is experiencing increasingly volatile weather patterns characterised by rising global temperatures, intensified storm cycles, and unpredictable seasonal behaviour, contributing to widespread ecological and societal disruption [1, 2]. Recent research highlights a 1.2° C increase in global average temperatures over the past century, leading to rising sea levels and worsening natural disasters.

Mars, our neighbouring planet, during its early stages, went through profound volcanic activity, asteroid and comet collisions and even climate changes [3].

Researchers have focused multiple research questions that revolve around Mars [4, 5, 6, 7, 8]. However, the comprehensive comparative analysis of Earth and Mars weather patterns remains largely unexplored.

The two planets, though different in many aspects, share some similarities. Below, we highlight the key ones:

Similarities	
Composition	Similar geological composition (silicate and metal)
Vulcanism	Past and present vulcanic activity shaped the geography of both planets
Erosion	Landscape shaped by erosion caused by water and wind
Differences	
Atmosphere	Different chemical composition of both atmospheres [7]
Water	Unlike Mars, Earth has abundant water (in liquid state)
Magnetic Field	Earth: shielded of harmful radiation. Mars' magnetic field is weaker

Table 1: Adapted from Ely, D. R. (2024). Exploring Mars: The Red Planet Revealed. 3

The literature indicates a lack of cross-planetary studies, particularly in the use of Martian data to contextualise and potentially mitigate Earth's climactic crisis. This project aims to help bridging this gap by leveraging Data Warehouse and Data Lake Systems to compile weather datasets from both planets and analyse them to identify possible underlying climate trends and interactions.

[7] Earth's atmosphere is mostly composed of oxygen and nitrogen. Mars' is mostly composed of carbon dioxide

3 Methodology

Based on this shortage of relevant information, our group set out to collect data from relevant sources. We collected Mars and Earth's weather data, as well as Solar Flare records.

The procedure followed, from Amazon Web Services (AWS) infrastructure definition, to data acquisition, storing, cleaning and visualisations is described in the following sections.

3.1 Architecture

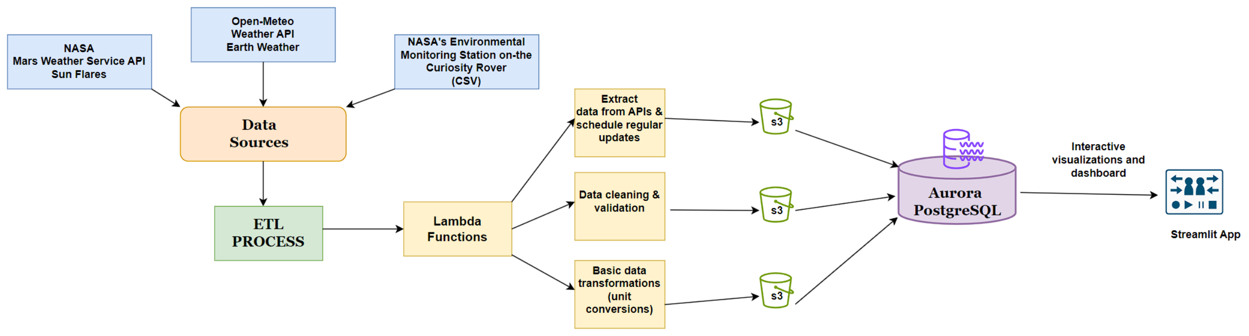

NASA Mars Weather Service API Sun Flares
Open-Meteo Weather API Earth Weather
NASA's Environmental Monitoring Station on-the Curiosity Rover (CSV)
Data Sources
ETL PROCESS
Lambda Functions
Extract data from APIs & schedule regular updates
Data cleaning & validation
Basic data transformations (unit conversions)
s3
s3
s3
Aurora PostgreSQL
Interactive visualizations and dashboard
Streamlit App

Figure 1: Project Schema

3.2 Data Lake

To store and manage the amount of raw data collected for this project, a Data Lake architecture was implemented on AWS. This allows us to store any format of data in its native format without the need for extensive transformation or schema enforcement. Thanks to it, the tasks of data management are simplified, while still allowing us to conduct analytics on it. The configuration steps used to set up the AWS architecture are included in the following sections.

3.2.1 Data Sources

Mars' Weather Data

Mars' weather data was obtained from Kaggle. This dataset is a compilation of detailed meteorological measurements taken by the Rover Environmental Monitoring Station (REMS) onboard NASA's Curiosity rover from 2012 to 2018, which is stationed inside Gale Crater on Mars. The dataset spans Sol 1 to Sol 1895, providing a comprehensive view of Martian weather conditions over several Martian years. Key Data Points Collected:

- Terrestrial Date: Earth date when the data was transmitted back to Earth.
- Sol: Martian day count since Curiosity's landing.
- Solar Longitude (Ls): Martian season by measuring the planet's position relative to the Sun, crucial for understanding seasonal weather patterns.
- Minimum Temperature: Recorded daily in ℃, offering insights into the thermal lows experienced at the Gale Crater.
- Maximum Temperature: Highlights the daily temperature highs in ℃, aiding in the assessment of diurnal temperature variations on Mars.
- Atmospheric Pressure: Measured in Pascals, it provides data on the thin Martian atmosphere's behaviour and dynamics.

- Wind Speed: Daily wind conditions, which are essential for understanding Martian dust storms and their impact on visibility and rover operations.
- Atmospheric Opacity: Clarity or haziness of the atmosphere, influenced by dust and weather patterns.
- Month: The Earth month corresponding to the data collection, simplifying longitudinal studies and comparisons with Earth's calendar.

Solar Flares Data

The Solar Flares data was collected from DONKI API provided by NASA's Mars Weather Service. This API offers detailed real-time and historical data about solar flare activities which are crucial in understanding space weather dynamics affecting both Mars and Earth. Key Data Points Collected:

- Flare Identifier (flrID): A unique identifier for each solar flare event, essential for tracking and historical data analysis.
- Instruments: Details the specific instruments used to detect and measure the solar flares, such as the GOES-P: EXIS 1.0-8.0. These instruments are critical for ensuring the accuracy and reliability of the data.
- Event Timing: Includes the beginTime, peakTime, and endTime of each flare, providing a timeline of the flare's activity which is vital for correlating solar events with atmospheric phenomena on Mars and Earth.
- Class Type: Each flare is categorised into a class (e.g., M2.3, M4.7), indicating the flare's intensity and potential impact on space weather conditions.
- Source Location: Specifies the solar coordinates where the flare originated, aiding in the study of solar active regions and their behaviours.
- Active Region Number: The NOAA active region number associated with the flare, which helps in the longitudinal study of active solar regions.
- Linked Events: Identifies any related space weather events, such as Coronal Mass Ejections (CMEs) or Solar Energetic Particle (SEP) events, providing a comprehensive view of interconnected solar activities.
- Link: Provides a direct URL to detailed information about the specific flare event, allowing for deeper investigation and verification of data.

Earth's Weather Data

Earth's weather data was collected from Open-Meteo's API. It is possible to obtain multiple weather features from the API, but for the purpose of this project we collected only

- Date: the date is received in Unix Timestamp format (then converted to yyyy-mm-dd hh:mm:ss during data cleaning process)
- Coordinates: Latitude and Longitude of the selected locations. Geographical WGS84 coordinates of the location.
- Temperature: Air temperature collected 2m above the surface. Measured in degrees Celsius.
- Relative Humidity: collected 2m above the surface
- Rain: in mm. Liquid precipitation of the previous hour.
- Direct Radiation

The data was collected from 01.01.1940 to the present and corresponds to hourly observations of 11 different locations. These locations were chosen to observe a diverse range of climates, from desert region, to extreme subarctic, or even Mediterranean climates. This variety in terms of climate conditions available led us to depict more than one region to ensure that the scope of our analysis is not limited by where the observation takes place.

- The Amazon, Brazil. World's most biologically diversified location and largest rainforest [8],
- Athens, Greece,
- Death Valley, California (where the highest temperature was ever recorded anywhere in the world) [9],
- Everest, between Nepal and the Tibet Autonomous Region of China,
- London, UK,
- McMurdo, Antarctica,
- Oymyakon, Russia,
- Serengeti, Africa,
- Sydney, Australia,
- Zurich, Switzerland.

3.2.2 Data Collection

The data collection process is composed of 3 separate Lambda Functions, 2 of them are responsible for the API calls, and one for the Kaggle data which was not obtained via API. There are also 3 separate S3 Buckets where each type of data collected.

We created S3 buckets not only to store our data but also for the environment layers for our lambda functions.

Here are the detailed instructions on how to set up an Amazon S3 bucket, which is essential for storing data collected from various sources. The steps outlined ensure that the bucket is configured for optimal performance and security:

1. **Navigate to the S3 Management Console**:
   - Access the AWS Management Console.
   - In the search bar at the top, type "S3" and open the S3 service page.
   - Click on the `Create bucket` button.
2. **Set the Bucket Name and Region**:
   - **Bucket Name**: Enter a unique name for your bucket. Remember, bucket names must be globally unique across all existing AWS users.
   - **Region**: Select the AWS Region where this bucket will reside. It is advisable to choose a region that is geographically closest to your users or yourself to minimize latency and costs.
3. **Configure Options**:
   - Disable ACLs (Access Control Lists) to ensure that the bucket uses the newer and more secure access control settings.
   - Enable **Bucket Versioning** to keep multiple versions of an object in case it is deleted or overwritten.
   - Enable **Bucket Key** for enhanced encryption options.

Note: It is recommended to select Linux 2 for the environment as it is considered the most stable version for running AWS-related operations.

In order to execute the Lambda function, it was necessary to create layers that included all of the required libraries. The following describes the process of creating these layers in the AWS Cloud9 environment:

[8] https://www.worldwildlife.org/stories/what-animals-live-in-the-amazon-and-8-other-amazon-facts
[9] https://www.visitcalifornia.com/experience/things-do-death-valley-national-park

1. **Access the AWS Cloud9 Console**: Use the search function within the AWS Management Console to find and select Cloud9.
2. **Initiate a New Environment**: Navigate to the Cloud9 dashboard and click on “Create environment”.
3. **Configure Your Environment**:
    - **Name Your Environment**: Assign a name of your choice. For simplicity, default settings can be used with access via SSH.
    - Confirm the settings by clicking “Create”.
    - Monitor the creation process via the loading bar at the top of the console.
4. **Access the IDE**: Once the environment setup is complete, open the integrated development environment to begin your work.

Listing 1: Setting Up Python and Dependencies in Cloud9

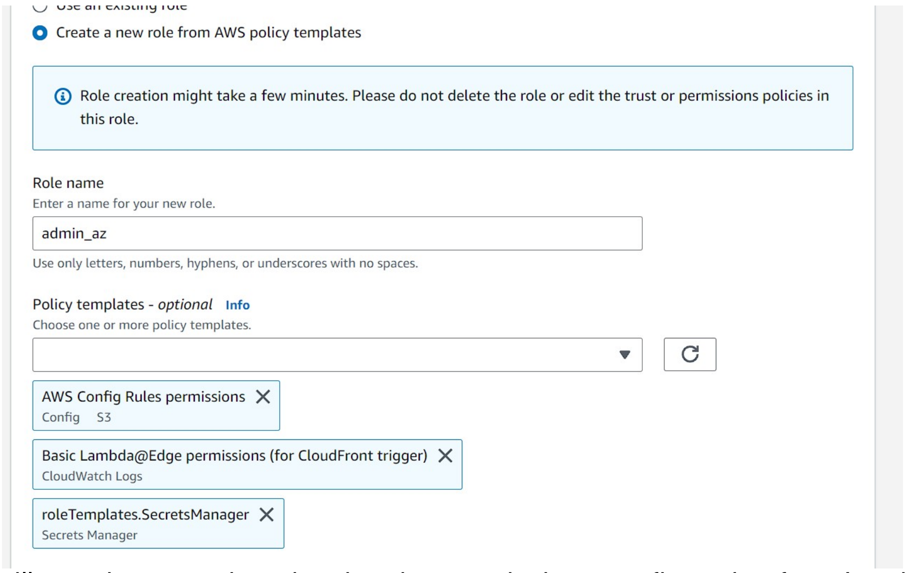
```
# Install/build python3.9 in the Cloud9 Linux session.

# PHASE 1:
cd /opt
sudo wget https://www.python.org/ftp/python/3.9.6/Python-3.9.6.tgz
sudo tar xzf Python-3.9.6.tgz
cd Python-3.9.6
sudo ./configure --enable-optimizations
sudo make altinstall
sudo rm -f /opt/Python-3.9.6.tgz

# Return to home directory
cd ~/
# Install virtualenv
python3 -m pip install --user virtualenv

# PHASE 2: Create a virtual environment in python 3.9 and add the needed libraries/packages
mkdir folder
cd folder
virtualenv v-env --python=python3.9
source ./v-env/bin/activate
pip install pandas numpy tweepy requests psycopg2-binary
deactivate

# PHASE 3: Ship all the libraries/packages into the AWS lambda service
mkdir python
cd python
cp -r ../v-env/lib/python3.9/site-packages/* .
cd ..
zip -r my_custom_layer.zip python
# save to s3 bucket
aws s3 cp my_custom_layer.zip s3://databaseenv/
aws lambda publish-layer-version --layer-name my_custom_lambda_layer --zip-file fileb://
    ↪ my_custom_layer.zip --compatible-runtimes python3.9
```

Final Steps:

- Download the `my_custom_layer.zip` file from Cloud9, which can be located in the sidebar.
- Since the zip file exceeds 50MB, upload it to an S3 bucket. Then, link it to an AWS Lambda layer for access, as direct additions of files this size are not supported.

To establish a new AWS Lambda function[10] we followed the steps outlined below. These steps ensure that the function is configured with the necessary permissions and runtime environment suited for interfacing with other AWS services such as S3, AWS Glue, and CloudWatch Logs.

1. **Access the AWS Lambda Console**:
   - Navigate to the AWS Management Console.
   - Use the search function to locate and open the AWS Lambda service.
2. **Initialize the Function Creation**:
   - Click on the "Create function" button which is prominently displayed.
   - Select "Author from scratch" to start configuring a new function.
3. **Configure Function Basics**:
   - **Function Name**: Enter a name for your Lambda function.
   - **Runtime Selection**: Choose Python 3.9, as recommended for compatibility with previously created environments.
   - **Architecture**: Select the appropriate instruction set architecture that your Lambda function will utilize to execute.
4. **Set Permissions**:
   - Under the Permissions settings, opt to either choose an existing role or create a new execution role. This role should have sufficient permissions to interact with necessary AWS services.



Create a new role from AWS policy templates

Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.

Role name
Enter a name for your new role.
admin_az
Use only letters, numbers, hyphens, or underscores with no spaces.

Policy templates - *optional* Info
Choose one or more policy templates.

AWS Config Rules permissions
Config S3

Basic Lambda@Edge permissions (for CloudFront trigger)
CloudWatch Logs

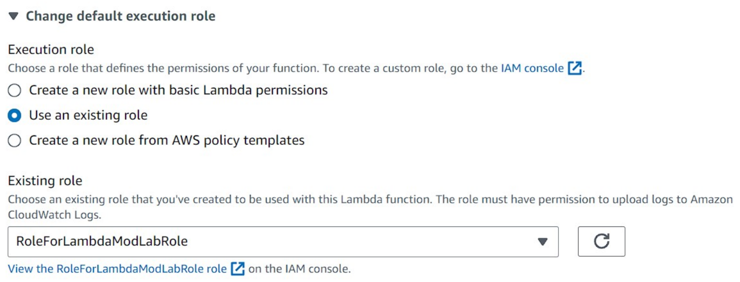
roleTemplates.SecretsManager
Secrets Manager

Figure 2: An illustrative snapshot showing the permissions configuration for a Lambda function

   - Ensure the role includes permissions to:
     - Read from and write to the specified S3 bucket.
     - Access AWS Secrets Manager for API keys management.
     - Log activities to Amazon CloudWatch for effective monitoring and troubleshooting.
   - By default, AWS Lambda will configure an execution role that can upload logs to Amazon CloudWatch Logs.

Note: The specific execution role used in the setup, named "Labrole", differs from typical configurations due to restrictions in the AWS Academy Learner Lab environment. This role does not match the one depicted in earlier graphical resources but is adequate for the lab's scope.

[10]Further details on each lambda function will be provided according to the data



▼ Change default execution role

Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.

- Create a new role with basic Lambda permissions
- [✓ Use an existing role]
- Create a new role from AWS policy templates

Existing role

Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

RoleForLambdaModLabRole

View the RoleForLambdaModLabRole role on the IAM console.

Figure 3: A representative image of the AWS role selection interface used during Lambda setup

Here are the steps to create Lambda Layers which can be added to any Lambda function to extend its capabilities with additional libraries or custom runtime environments.

1. **Navigate to the Lambda Console**:
   - Access the AWS Management Console and locate the AWS Lambda service.
2. **Create a New Layer**:
   - In the Lambda dashboard, select “Layers” from the sidebar, then click on “Create layer”.
   - **Name Your Layer**: Provide a meaningful name that reflects its purpose.
   - **Upload the ZIP File**: Choose “Upload a .zip file” and select the ‘layer.zip‘ file previously prepared in Cloud9.
   - **Specify the Compatible Runtime**: Set this to Python 3.9 and architecture to x86_64, matching the environment prepared earlier.
   - Confirm by clicking on “Create”.

Adding the Layer to a Lambda Function:

1. **Access the Lambda Function**:
   - Return to the main page of the AWS Lambda service and select your function.
2. **Integrate the Layer**:
   - Scroll to the bottom of the function configuration page and click on “Add a layer”.
   - Choose “Custom layers”, select the newly created layer and the appropriate version.
   - Complete the integration by clicking on “Add”.

Clean Up the Cloud9 Environment, to manage resources effectively and avoid unnecessary costs, these are the steps to delete the Cloud9 environment after its purpose has been served and we made sure it worked in the lambda function:

1. **Return to the Cloud9 Console**:
   - Locate and open the Cloud9 service from the AWS Management Console.
2. **Select the Environment for Deletion**:
   - Choose the environment you intend to remove from the list of available environments.
3. **Delete the Environment**:
   - Click on “Delete” and confirm your choice by following the prompts to finalize the deletion.

After downloading the Mars weather data, it was uploaded to a designated Amazon S3 bucket. This method ensured that the data remained secure and easily accessible within our AWS environment. The S3 bucket serves as a centralized repository, facilitating streamlined access and subsequent data processing stages.

The pipeline using a lambda function to insert and clean the data can be found in the appendix 4.

To deal with the multiple locations, avoid API request limitations and manage AWS resources, Earth's weather data was collected by using a Step Function that is responsible for running the Lambda Function (demonstrated in the implementation code shown in the appendix 5) 11 times, one for each location. As mentioned, the Lambda Function is then responsible for storing the data in the S3 bucket created for this purpose. Since the collection is triggered weekly and the amount of data is not large, the Step Function does not run the Lambdas in parallel.

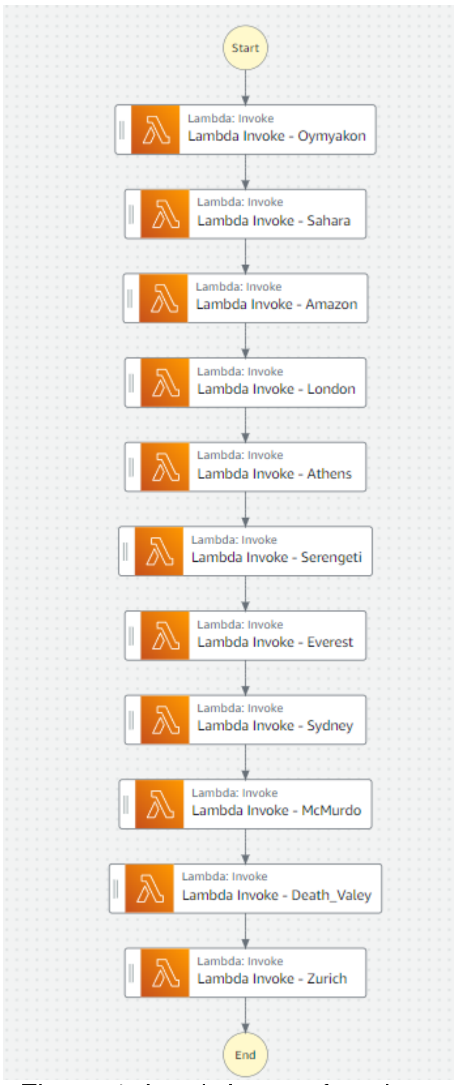


Figure 4: Lambda step functions

After creating the Lambda Function as described above, the procedure to create the Step Function is as follows:

1. **Navigate to the Step Function Console**:
   - Choose "Create state machine", ensuring that both the state machine and the Lambda Function are in the same AWS account and Region.
2. **Select "Blank" in the Template Dialog Box**:
   - There are multiple templates available, but **Blank** is the best suited.

3. **Select "Actions" Panel**:
   - Drag and drop "AWS Lambda Invoke" into the empty state labelled "Drag first state here".
   - **Provide Location Information to the Lambda**: The Lambda Function expects to receive the name and coordinates of the location from the Payload of the Lambda Invoke with the following structure:

```
          {
             "city": "Oymyakon",
             "latitude": 63.4608,
             "longitude": 142.785
           }
```

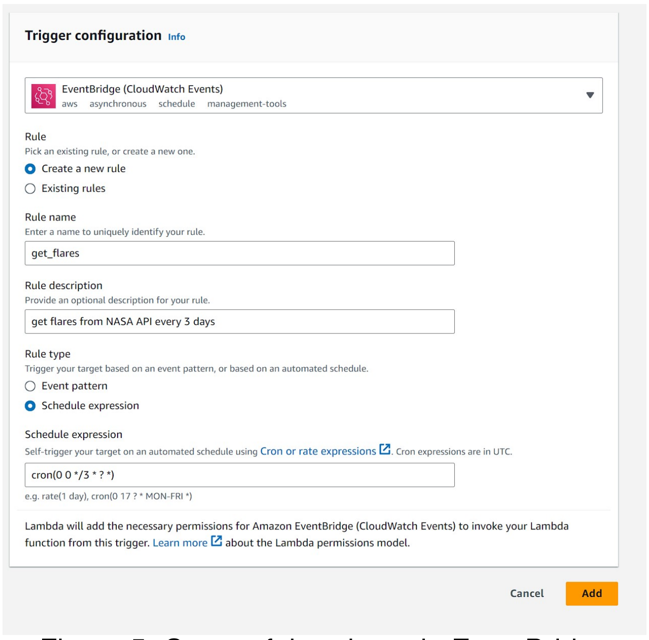
4. **Create the State Machine**: In the "Confirm role creation" dialog box, select "Confirm" to proceed.
5. **Run the State Machine**: Select "Start execution".

To manage the collection of solar flares data from the NASA DONKI API without exceeding the API request limits and to efficiently utilize AWS resources, a Lambda Function was initiated every seven days, which fetches the latest solar flares data. The Lambda Function is configured to handle the API's data structure, ensuring that the data is collected consistently on schedule. Once retrieved, the data is directly stored in a designated S3 bucket. This setup not only automates the data collection process but also secures the data in a scalable and accessible manner.

First configure trigger for lambda function to automate the process of retrieving solar flares data every 7 days:

1. Navigate to the Lambda Console and click on "Add trigger".
2. Choose Amazon EventBridge (CloudWatch Events) as the trigger source.
3. Set up a new rule in Amazon EventBridge:

   - Define a schedule using a cron expression: `cron(0 0 */7 * ?  *)`, which configures the function to execute every 7 days at midnight UTC.



EventBridge (CloudWatch Events)
aws asynchronous schedule management-tools

Rule
Pick an existing rule, or create a new one.
[✓ Create a new rule] Existing rules

Rule name
Enter a name to uniquely identify your rule.
get_flares

Rule description
Provide an optional description for your rule.
get flares from NASA API every 3 days

Rule type
Trigger your target based on an event pattern, or based on an automated schedule.
Event pattern [✓ Schedule expression]

Schedule expression
Self-trigger your target on an automated schedule using Cron or rate expressions. Cron expressions are in UTC.
cron(0 0 */3 * ? *)
e.g. rate(1 day), cron(0 17 ? * MON-FRI *)

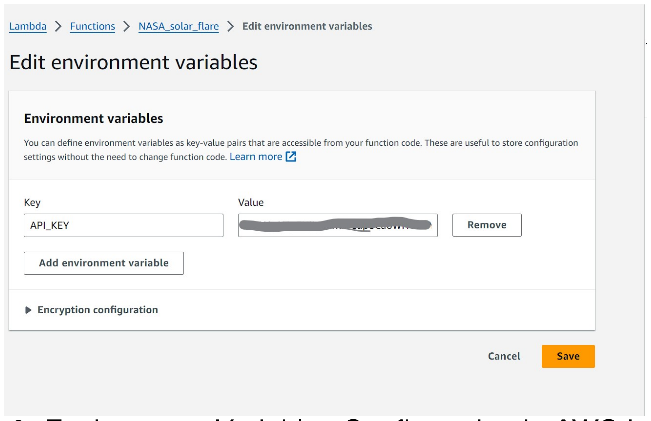
Lambda will add the necessary permissions for Amazon EventBridge (CloudWatch Events) to invoke your Lambda function from this trigger. Learn more about the Lambda permissions model.

Figure 5: Setup of the trigger in EventBridge

4. Confirm by clicking on "Add".

Store sensitive information securely using environment variables in the Lambda function configuration:

- Key: `API_KEY`, Value: *Your actual NASA API key*.
- Key: `BUCKET_NAME`, Value: *Name of your S3 bucket*.



Lambda > Functions > NASA_solar_flare > Edit environment variables

Edit environment variables

Environment variables

You can define environment variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code. Learn more

Key	Value
API_KEY	[redacted]

Figure 6: Environment Variables Configuration in AWS Lambda

The following Python code 6 is used in the AWS Lambda function to retrieve solar flares data from NASA's DONKI API and store it in an S3 bucket.

Finally deploy and test the Lambda function to ensure that the function executes correctly by manually invoking it or waiting for the next scheduled event to trigger the function. The data collected by each team member is then saved to an S3 bucket before any processing is done to the data.

3.2.3 Data Cleaning

Earth's Weather Data

Upon reviewing Earth's weather data, it was found that the dataset was predominantly clean and well-structured. However, initial observations revealed a few instances of missing values in the "Rain" and "Direct Radiation" records during the first 7 hours of data collection. Given the negligible proportion of these missing values (approximately 0.000948% of the dataset), the decision was made to remove these rows instead of applying imputation techniques. This approach helped maintain the integrity of the dataset without introducing potential biases that could arise from estimating missing data.

Furthermore, dates were converted from Unix Timestamp format to a more conventional `yyyy-mm-dd hh:mm:ss` format to facilitate easier analysis and visualization. In addition, we transformed the weather data from JSON to a DataFrame format was crucial for standardized processing and analysis, allowing us to integrate this data effectively with other datasets in our study.

Solar Flares Data

The solar flares data required several cleaning steps:

- **Datetime Conversion:** All date and time information was converted to a standardized datetime format to ensure consistency across temporal data.
- **Mapping Solar Flares Intensity:** The intensity of solar flares was categorized using a predefined scale to simplify analysis and interpretation, as follows:
  - 'X': Most intense
  - 'M': Medium
  - 'C': Small
  - 'B': Very Small

  - 'A': Smallest

  This classification allows for a more structured analysis of solar flare impacts on atmospheric conditions.

Mars Weather Data from Kaggle

For the Mars weather dataset the cleaning procedure included:

- **Removing Empty Columns:** The 'Wind Speed' column was consistently empty across the dataset and was therefore dropped to streamline the dataset structure.
- **Handling Missing Data:** Rows containing null values in critical fields such as 'min_temp', 'max_temp', and 'pressure' were removed to maintain data quality.
- **Date Conversion:** The 'Terrestrial Date' was converted from a string format to datetime, enhancing the uniformity of temporal data across our study.
- **Renaming Columns:** The 'Season' column was renamed to 'Month' to accurately reflect the data content, which represented the Earth month corresponding to the Martian data collection period.

These steps ensured that our datasets were not only clean but also formatted consistently to aid in comparative analysis across different planetary conditions. This meticulous approach to data cleaning is crucial for deriving reliable insights from complex, multi-source datasets in our cross-planetary study.

3.3 Data Warehouse

3.3.1 Architecture

Database Setup

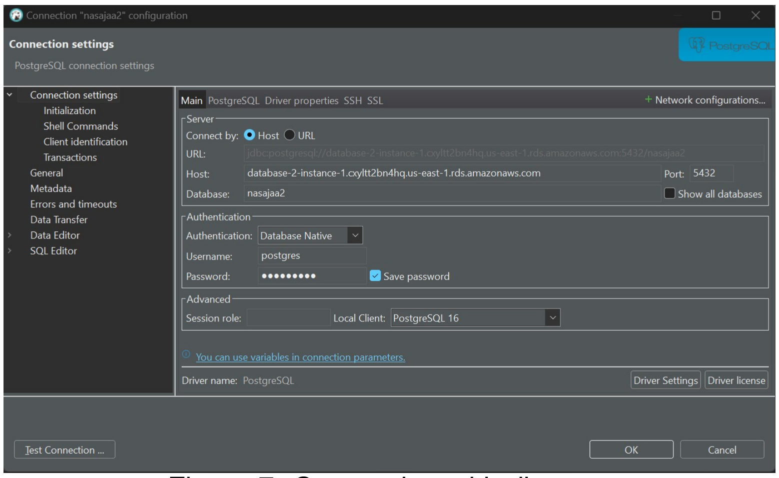
To support our study, we required a database that was not only reliable and scalable but also robust and adaptable to our diverse data handling needs. We selected Amazon Aurora PostgreSQL because it combines the advanced performance and scalability of Aurora with the flexibility and extensive feature set of PostgreSQL. This choice offered us powerful data processing capabilities and extensibility, allowing us to customize the database to meet the specific demands of our project. Furthermore, PostgreSQL's reputation for reliability and data integrity ensured the accuracy and consistency of our data. Being an open-source solution, it is cost-effective and supported by a vibrant community that contributes to its active development and standards compliance, facilitating easy data sharing and integration with other systems. Here is a detailed step-by-step guide on how to set up the Aurora PostgreSQL database:

1. **Access the RDS Dashboard:** Begin by navigating to the Amazon RDS dashboard. This can be done by typing "RDS" into the search bar of the AWS Management Console and selecting RDS from the search results.
2. **Create a New Database:** Click on the 'Create database' button to start the setup process.
3. **Select the Database Engine:** Choose 'Aurora' and then select the option for 'PostgreSQL Compatible' to ensure compatibility with PostgreSQL-based applications.
4. **Select the Engine Version:** It's recommended to use the default version provided to ensure stability and compatibility with most current applications.
5. **Configure Public Access:** Set 'Public access' to 'Yes' to allow connections from outside the AWS environment, which is useful for development and testing phases.
6. **Choose Instance Specifications:** Opt for the lowest memory configuration available to minimize costs and avoid potential errors during database creation. Ensure that monitoring features are not enabled to simplify the creation process.

After the database has been successfully created, it's essential to be able to interact with it effectively:

7. **Download DBeaver:** To manage the database, download and install DBeaver, which is an open-source SQL client and database administration tool. It provides a user-friendly graphical interface for more straightforward database querying and management.
8. **Connect to the Database:** Open DBeaver and create a new connection to your Aurora PostgreSQL database. You will need to specify the database endpoint, username, and password 9 that were configured during the database setup.

These steps outline the process of setting up and accessing a scalable cloud-based database which supports the high-performance requirements of our data-intensive application.


Connection "nasajaa2" configuration

Connection settings

PostgreSQL connection settings

Main | PostgreSQL | Driver properties | SSH | SSL

Server

Connect by: [✓ Host] URL

URL: jdbc:postgresql://database-2-instance-1.cxyltt2bn4hq.us-east-1.rds.amazonaws.com:5432/nasajaa2

Host: database-2-instance-1.cxyltt2bn4hq.us-east-1.rds.amazonaws.com Port: 5432

Database: nasajaa2 Show all databases

Authentication

Authentication: Database Native

Username: postgres

Password: ●●●●●●●●● [✓ Save password]

Advanced

Session role: Local Client: PostgreSQL 16

You can use variables in connection parameters.

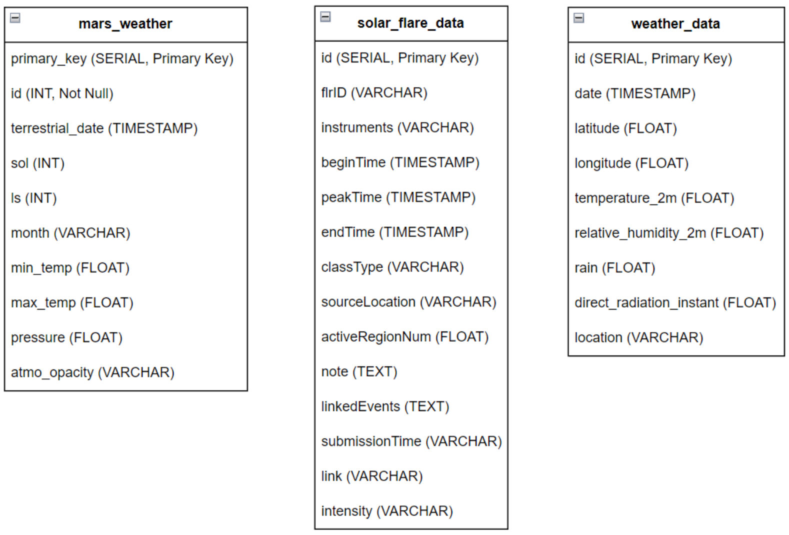
Driver name: PostgreSQL

Figure 7: Connection with dbeaver

Database schema

Our project utilizes a structured database schema designed to efficiently organize and store the different datasets involved in our cross-planetary study. This schema includes three primary tables: mars-weather, solar-flare-data, and weather-data. Each table is tailored to handle specific types of data from our sources, facilitating effective data retrieval, manipulation, and analysis.

You can find the SQL queries for creating the tables in the appendix 8.



mars_weather
primary_key (SERIAL, Primary Key)
id (INT, Not Null)
terrestrial_date (TIMESTAMP)
sol (INT)
ls (INT)
month (VARCHAR)
min_temp (FLOAT)
max_temp (FLOAT)
pressure (FLOAT)
atmo_opacity (VARCHAR)

solar_flare_data
id (SERIAL, Primary Key)
flrID (VARCHAR)
instruments (VARCHAR)
beginTime (TIMESTAMP)
peakTime (TIMESTAMP)
endTime (TIMESTAMP)
classType (VARCHAR)
sourceLocation (VARCHAR)
activeRegionNum (FLOAT)
note (TEXT)
linkedEvents (TEXT)
submissionTime (VARCHAR)
link (VARCHAR)
intensity (VARCHAR)

weather_data
id (SERIAL, Primary Key)
date (TIMESTAMP)
latitude (FLOAT)
longitude (FLOAT)
temperature_2m (FLOAT)
relative_humidity_2m (FLOAT)
rain (FLOAT)
direct_radiation_instant (FLOAT)
location (VARCHAR)

Figure 8: Relational Database Structure

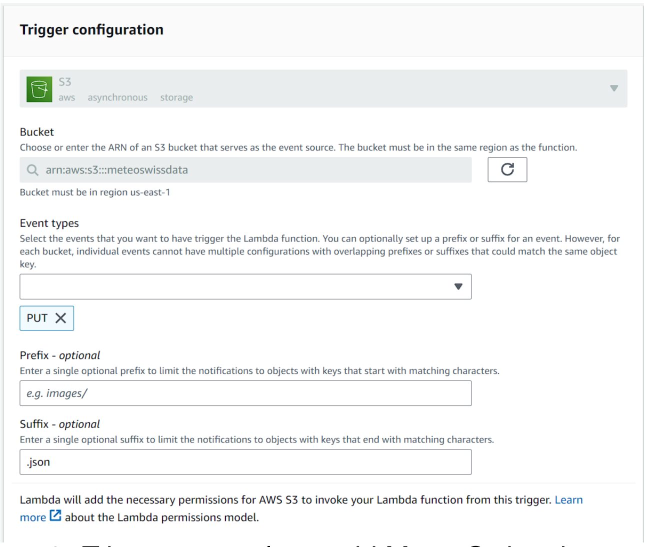
Automating Data Ingestion

To streamline the data ingestion process into our Aurora PostgreSQL database and manage new data as it arrives efficiently, we employed AWS Lambda functions activated by new file uploads in designated S3 buckets. Each bucket has a specific trigger set up to handle files according to their types. Below are the detailed steps to configure these S3 triggers and the core implementation in the Lambda function for processing and storing the data:

Setting Up the S3 Triggers

Each S3 bucket associated with different data types (Mars weather data, solar flares, and Earth weather data) has its own trigger. Here's how to set up each trigger:

1. **Navigate to the Lambda Configuration:** Open your Lambda function's configuration page through the AWS Management Console.
2. **Add New Triggers:** Begin by clicking on "Add trigger" and select S3 from the list of available AWS services.
3. **Configure Trigger Details for Each Bucket:**
    - **Mars and Solar Flares Data:**
        - Select the respective bucket for Mars or solar flares data.
        - Set the event type to "PUT" for uploading new files.
        - Specify the suffix as ".csv" to trigger the function only for CSV files.
    - **Weather Data:**
        - Choose the S3 bucket used for weather data.
        - Select "PUT" as the event type for new uploads.
        - Specify the suffix as ".json" to limit the trigger to JSON files.
4. **Save Each Configuration:** After defining the conditions for each trigger, save the configurations. This action will automatically establish the necessary permissions for the Lambda function to access the specified S3 buckets.


Trigger configuration

S3
aws asynchronous storage

Bucket
Choose or enter the ARN of an S3 bucket that serves as the event source. The bucket must be in the same region as the function.
arn:aws:s3:::meteoswissdata
Bucket must be in region us-east-1

Event types
Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.
PUT

Prefix - *optional*
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters.
e.g. images/

Suffix - *optional*
Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters.
.json

Lambda will add the necessary permissions for AWS S3 to invoke your Lambda function from this trigger. Learn more about the Lambda permissions model.

Figure 9: Trigger example to add MeteoSwiss data to DB

Our Lambda function is equipped to handle and process the data files retrieved from these S3 buckets. The function's capabilities include:

- **Data Retrieval:** It pulls new data files using Boto3, the AWS SDK for Python.

- **Data Cleaning:** Applies predefined preprocessing functions to clean the data, including removing unneeded columns, converting date formats, and mapping categorical data correctly.
- **Data Insertion:** The cleaned data is loaded into the Aurora PostgreSQL database via SQLAlchemy, providing robust data management and relational mapping.

Code and Implementation

Detailed in the appendix 7 the Python code includes comprehensive error handling to ensure stability in production. We also made sure adhere to best practices by securing AWS credentials in a configuration file rather than embedding them directly in the code. The function is designed to manage various file formats and schemas, applying specific preprocessing techniques depending on the source of the data.

3.4 Visualization

Visualizing the data collected is crucial for understanding it by providing a quick way to identify patterns, trends, outlier values, and relationships that may not be apparent from looking at the raw data.

Our group developed a Streamlit app, accessible at https://comparingskies-nasaweatherexploration.streamlit.app/, which includes multiple interactive plots. These plots display the evolution of datasets over time and the relationships between them, as they are all based on time-series observations.

The app was designed from the perspective of a group of space enthusiasts and is organized into several sections:

- **Home**: The app's homepage.
- **About**: Provides an introduction to the app, the datasets, and the analysis.
- **Earth**: Features interactive plots of Earth's weather data by location and attribute (Temperature, Rain, Humidity, and Direct Radiation), comparisons of Mars minimum temperature vs. Earth temperature based on solar flare impact intensity, and analysis of Earth's temperature in conjunction with solar events.
- **Mars**: Displays interactive plots of Mars' weather features (Temperature and Pressure).
- **Flares**: Shows an interactive plot of solar flare events over time.

To ensure the app automatically retrieves the latest data, we implemented a module named `db_connect.py`, which establishes a connection to our Aurora PostgreSQL database. This module uses SQLAlchemy to execute database queries efficiently and securely. The database credentials are loaded from a TOML configuration file to maintain security and flexibility.

Here is an overview of the code in the `db_connect.py` file, which is further detailed in the appendix under the title "Database Connection for Streamlit App":

Listing 2: Database Connection Functions

```python
import sqlalchemy
from sqlalchemy.sql import text
import toml
import pandas as pd

# Load database credentials from secrets.toml
config = toml.load("config\secrets.toml")
db_config = config['database']

# Create the database URL
database_url = f"postgresql://{db_config['user']}:{db_config['password']}@{db_config['host
    ↪ ']}:{db_config['port']}/{db_config['db_name']}"

# Create an engine
```

```
engine = sqlalchemy.create_engine(database_url)

def weather_data_location(city_name):
    try:
        with engine.connect() as connection:
            result = pd.read_sql_query(f"SELECT * FROM weather_data WHERE location = '{
                ↪ city_name}';", connection)
            return result
    except Exception as e:
        print(f"An error occurred: {e}")
```

In the Streamlit app, data retrieval is implemented as follows:

Listing 3: Retrieving Data in Streamlit

```
import db_connect
data = db_connect.weather_data_location('Sydney')
```

This integration allows our Streamlit app to dynamically fetch and display the most current data, making our visualizations both interactive and up-to-date.

4 Research

4.1 Contributor Spotlight

Dr. Luna Starfield, Associate Professor of Planetary Science at the Institute for Space and Environmental Studies, specializes in climatology and planetary science, focusing on comparative planetary environments. Her scholarly pursuits include exploring planetary weather patterns, investigating the viability of life on other planets, and studying the effects of climate change. Dr. Starfield's contributions to our project are pivotal, providing essential data and insights that enhance our analysis of complex weather patterns and climate interactions across planets. Her ongoing research and educational endeavors continue to propel forward the boundaries of planetary science, making significant impacts on both academic and practical fronts.

4.2 Research Results

While introducing this project, we defined the key research questions that guided our study and outlined the methodological approach employed to address them. This chapter aims to address those research questions, drawing upon the findings from our data analysis and interpretation.

What similarities and differences in patterns can be identified between Mars and Earth, and how can these insights improve our models of atmospheric behavior on Earth?

Based on our observation, we concluded that in the timeframe studied both planets display a cyclic fluctuations of their temperatures.

- Both planets experience daily temperature cycles resulting from its rotation cycle. Similarly to Earth, that completes a rotation around itself in 23.9 hours, Mars' takes 24.6.
- Both planets display cyclic patterns that correspond to their planetary seasons. Unlike Earth's 365 days to complete a revolution around the Sun, Mars takes 687 to complete this cycle.

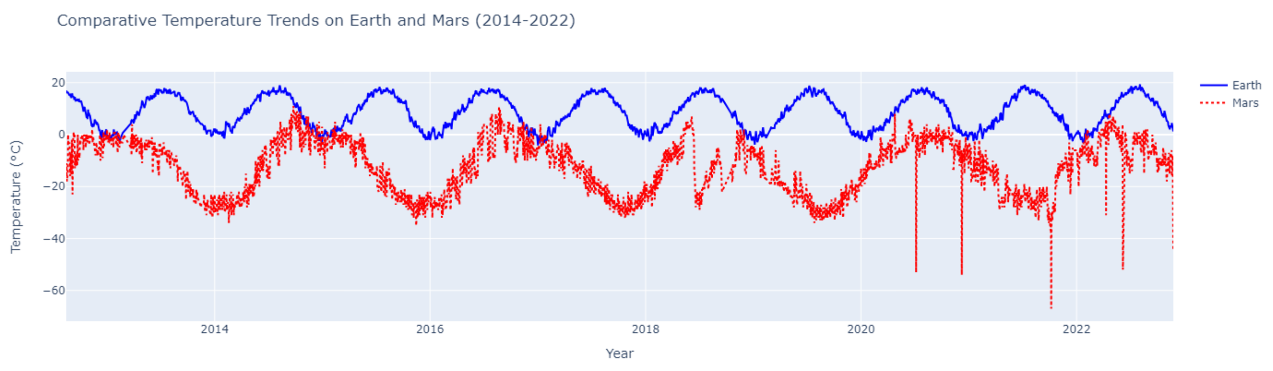


Figure 10: Time Series of Mars and Earth Temperature Data

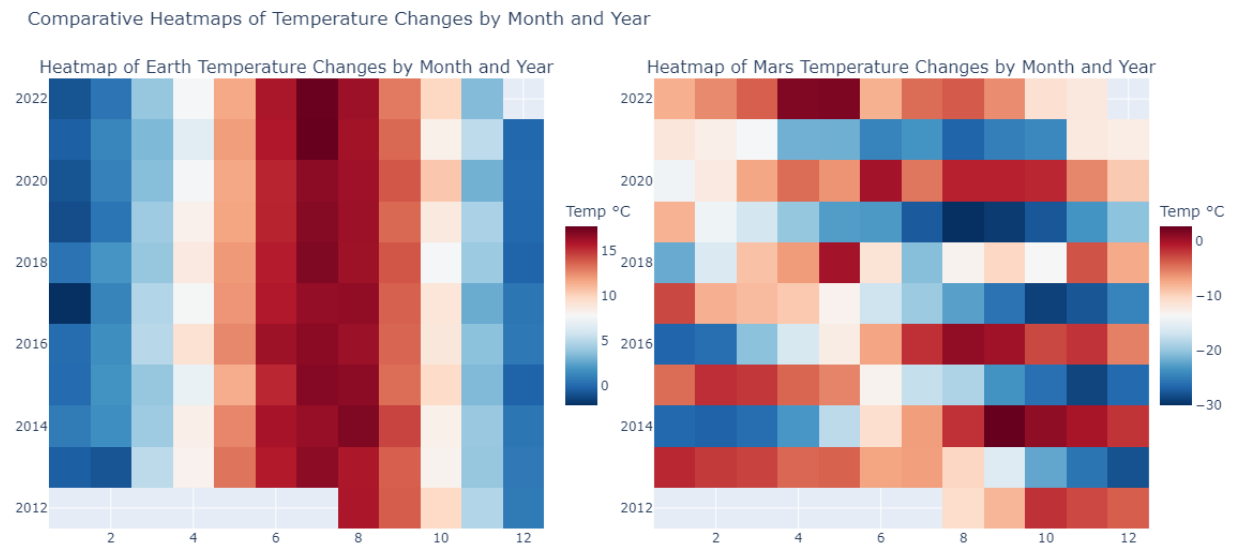


Figure 11: Comparative Heatmaps of Temperature Changes by Month and Year

The visualizations provided inconclusive insights, prompting us to conduct a more detailed statistical analysis to better understand the data.

Statistical and Seasonal Analysis of Temperature Data

The decade-long temperature data from Earth and Mars provide profound insights into the climatic behaviors of both planets. This analysis, particularly relevant to Dr. Luna Starfield's interests in comparative planetary environments, is detailed below.

Statistical Overview

Mean Temperature

- **Earth:** Averages at approximately 8.55℃, reflecting moderate global climates.
- **Mars:** Averages at about -12.74℃, significantly colder, indicative of Mars' thinner atmosphere and greater distance from the sun.

Median Temperature

- **Earth:** The median of 8.67℃ suggests a slightly skewed distribution, with more days colder than the mean.

- **Mars:** The median of -12.0℃ is slightly higher than the mean, suggesting a slight skew towards warmer temperatures.

Mode Temperature

- **Earth:** The mode at 12.77℃ indicates the most recurrent temperature, considerably higher than both mean and median, suggesting occasional peaks.
- **Mars:** The mode at -4.0℃, much higher than the median or mean, suggests infrequent but significant temperature spikes.

Seasonal Temperature Analysis

This analysis helps uncover how each planet's temperatures vary throughout the year and over the years:

- **Earth**
  - Warmer temperatures typically in the third quarter (July-September), aligning with summer in the northern hemisphere, and cooler temperatures in the first quarter (January-March).
  - A slight trend of increasing temperatures in the second quarter (April-June) may indicate gradual warming or changes in seasonal weather patterns.
- **Mars**
  - Seasonal variations are more extreme due to its eccentric orbit and thin atmosphere, with least negative temperatures during the third quarter (July-September).
  - Extremely low temperatures in the fourth quarter (October-December) and first quarter (January-March), with 2019 showing the lowest temperatures, suggesting possible atmospheric or orbital influences.

Volatility Analysis

- **Earth:** Volatility of 6.32 shows considerable variability, influenced by complex climate systems including ocean currents and varying geographical landscapes.
- **Mars:** Higher volatility of 10.74 reflects more extreme temperature fluctuations, likely exacerbated by its thin atmosphere which cannot retain heat effectively.

Implications for Comparative Planetary Studies

- Understanding Earth's moderate seasonal shifts versus Mars' extreme shifts can aid in developing better predictive models for Earth's changing climate and for planning activities on Mars.
- Mars offers a simpler model for studying atmospheric behavior, which could be leveraged to refine Earth's more complex climate models.

Recommendations for Future Research

- Continued monitoring over subsequent years to verify these trends and refine predictive models.
- Leveraging data from other planets in our solar system to enhance understanding of atmospheric processes.
- Enhancing remote sensing technologies to improve data accuracy and resolution for both Earth and Mars.

Dr. Luna Starfield can use these insights to advance her research into the implications of Martian weather patterns for Earth, enhancing understanding within the scientific community and informing both academic content and future exploratory missions. This analysis not only aids in academic pursuits but also has practical implications in planning for future human activities on Mars.

Can weather data from Mars be utilized to enhance our understanding of atmospheric dynamics on Earth, thereby improving our ability to predict and mitigate extreme weather events?

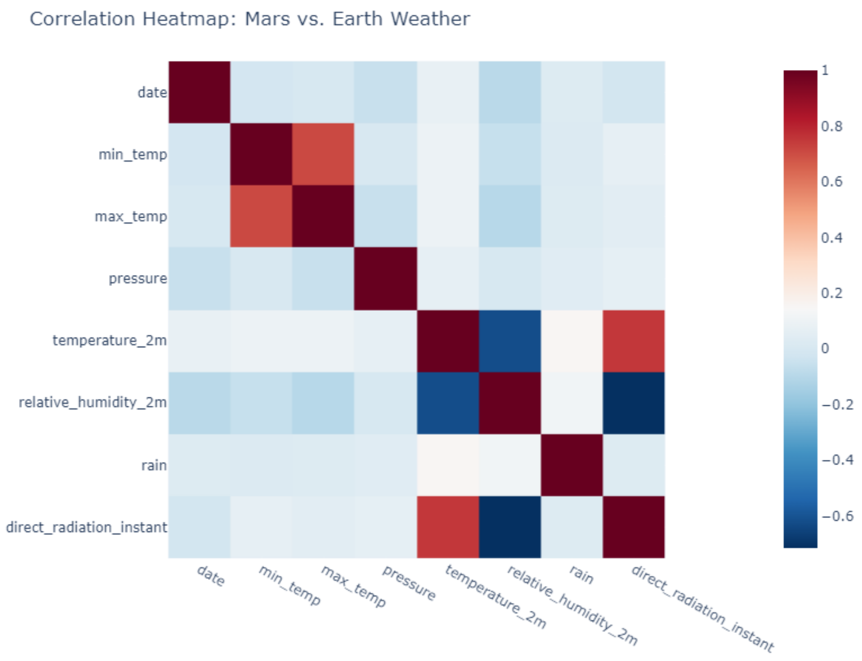


Figure 12: Correlation Heatmap Mars vs. Earth Weather

The heatmap, designed to visualize correlation coefficients between various weather parameters from Mars and Earth, indicates minimal to no significant correlations. These results suggest that weather parameters from Mars, such as maximum and minimum temperatures and atmospheric pressure, do not align in any meaningful way with Earth's temperature, relative humidity, and rainfall.

Key Observations

- **Lack of Significant Correlation:** No significant correlations are evident across the assessed variables, indicating a lack of direct or useful predictive relationships between the atmospheric conditions of the two planets.
- **Isolated Data Points:** The absence of strong correlations implies that Martian weather data, given the variables analyzed, does not closely align with Earth's atmospheric parameters, limiting its utility for predicting or understanding Earth's weather patterns.

Linear Regression Analysis

To explore potential correlations between Martian and Earth weather patterns, we conducted a linear regression analysis using Python's `scikit-learn` library. The aim was to determine if weather variables from Mars could predict Earth's surface temperatures.

The linear regression model attempted to predict Earth's surface temperature based on Mars' maximum temperature and atmospheric pressure.

Model Outcomes

- **Coefficients:** The regression model coefficients for Mars' maximum temperature (0.00829759) and pressure (0.00844555) are very small, suggesting a negligible influence of these Martian parameters on Earth's temperature.
- **Intercept:** The intercept of 1.609 indicates the baseline prediction for Earth's temperature when Mars' measurements are at zero, which is abstract considering the different environmental conditions of the two planets.

Interpretation and Implications

- **Ineffectiveness for Prediction:** The results reinforce the indication from the heatmap that Martian weather data offers little practical value in predicting Earth's weather conditions.
- **Scientific Exploration:** Although the Martian data does not provide predictive power for Earth's weather, it remains valuable for theoretical or experimental studies in comparative planetary science.
- **Utility in Comparative Planetary Studies:** The evident lack of correlation might be of interest in understanding the differences in planetary atmospheres, potentially impacting fields like astrobiology and the study of exoplanets.

Conclusion

This analysis underscores that the Martian atmospheric data, represented by the studied variables and timeframe, does not correlate significantly with Earth's weather parameters. While not directly enhancing our predictive capabilities for Earth's extreme weather events, Martian data can still inform broader scientific inquiries when used in comparative planetary research or theoretical modeling. This emphasizes the need for tailored, planet-specific models in atmospheric sciences, highlighting the complexity and distinctiveness of planetary environments.

This prompted an investigation into the potential impacts of solar flares on Martian and Earthly weather conditions, focusing on variations in atmospheric parameters that coincide with the timing of these solar events. The study involved analyzing correlations between solar flare intensity and key atmospheric variables, such as temperature, pressure, and atmospheric opacity on Mars, as well as temperature, humidity, and radiation levels on Earth.

Methodology and Analysis

The correlation analysis was conducted to ascertain patterns that could illuminate how weather conditions on both planets might interact or reflect upon each other under the influence of solar flare intensity.

Analysis of Mars Correlation Matrix

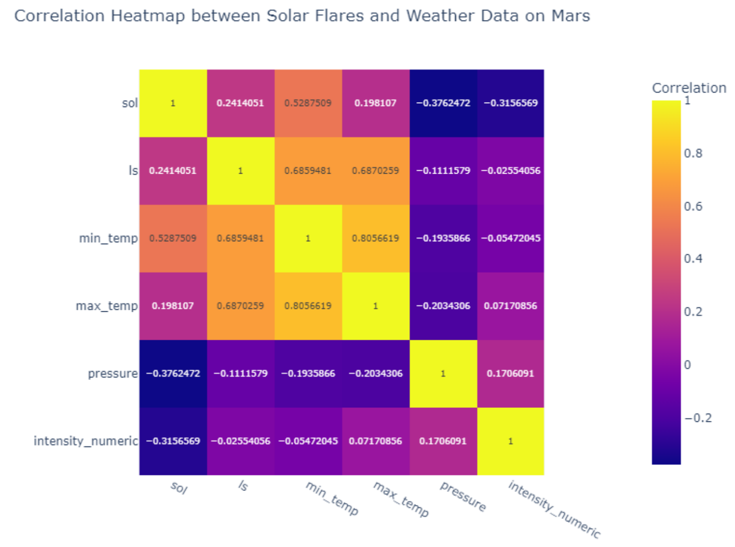


Figure 13: Correlation Heatmap between Solar Flares and Weather Data on mars

- **Seasonal Impact (Sol and Ls):** A moderate positive correlation was observed between 'sol' (Mars solar day) and 'min_temp' (0.53), indicating rising temperatures as the Martian day progresses. Strong correlations between 'ls' (solar longitude) and both 'min_temp' and 'max_temp' (0.69) suggest significant temperature shifts influenced by seasonal changes as Mars orbits the Sun.
- **Temperature and Atmospheric Pressure:** Both 'min_temp' and 'max_temp' exhibit a negative correlation with atmospheric pressure, though these are weak, indicating that higher temperatures may slightly correspond to lower pressures.
- **Solar Flares Impact:** Correlation between solar flare intensity and Martian atmospheric parameters is generally low. The most substantial correlation is between solar flare intensity and 'pressure' (0.17), indicating a minor increase in atmospheric pressure corresponding with increased solar flare activity.

Analysis of Earth Correlation Matrix

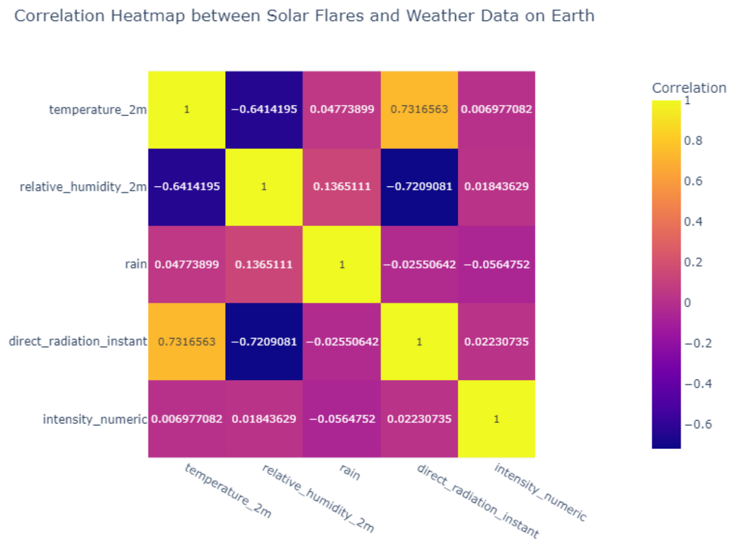


Figure 14: Correlation Heatmap between Solar Flares and Weather Data on earth

- **Temperature and Humidity:** A strong negative correlation exists between 'temperature_2m' and 'relative_humidity_2m' (-0.64), indicating that higher temperatures often lead to lower relative humidity levels.
- **Radiation and Weather:** Direct solar radiation ('direct_radiation_instant') shows a strong positive correlation with 'temperature_2m' (0.73) and a negative correlation with 'relative_humidity_2m' (-0.72), underscoring the significant impact of solar radiation on Earth's surface conditions.
- **Solar Flares Influence:** Solar flare intensity demonstrates very weak correlations with all of Earth's weather parameters, with the most significant being a negligible positive correlation with 'direct_radiation_instant' (0.02). This suggests minimal direct influence of solar flares on near-surface weather conditions.

Conclusion

The findings highlight that while Martian weather shows some responsiveness to solar flares, particularly in atmospheric pressure changes, Earth's weather conditions appear largely unaffected by solar events in terms of immediate changes. These insights contribute to our understanding of planetary weather systems and underscore the need for comprehensive, multidimensional studies to fully grasp these dynamics. Further, while the analysis can help refine atmospheric models on both planets, the minimal impact of solar flares on Earth's immediate weather conditions suggests that predictive models based on solar activity should be approached with caution.

How can the integration of weather data from Mars and Earth in a Data Warehouse and Data Lake system enhance our ability to identify and analyze underlying climate trends and interactions?

Integrating weather data from Mars and Earth into a Data Warehouse and Data Lake system can significantly enhance our understanding of climate trends and interactions by providing a unified, scalable platform that can seamlessly integrate and process the new information as it is collected, enabling continuous refinement of climate models and analyses.

From our experience and observation, besides scalability, this integration offers several key benefits, such as:

- **Centralized Data Management**: A Data Warehouse provides a structured and centralized repository where historical and real-time weather data from both planets can be stored. This facilitates easier access and management of data, ensuring consistency and accuracy in the datasets being analyzed.

  A Data Lake provides the flexibility of storing a wide variety of data formats in their raw form without the need to model the data during ingestion.

  This approach facilitates the identification of similarities, differences, potential interactions between the two planetary climates and their overall study by allowing researchers to develop advanced data models and simulations that incorporate multiple variables from different planetary environments. This can lead to more accurate predictions and a deeper comprehension of climate dynamics across the solar system.
- **Comparative Analysis**: With the datasets stored in an integrated system, it becomes easier to conduct comparative analyses. We can systematically compare specific weather events on Mars and Earth, such as dust storms and hurricanes, to understand their similarities and differences and how the Solar Flares impact these events. This can lead to a better understanding of the underlying mechanisms driving climate systems and the factors influencing their evolution.
- **Long-Term Trend Identification**: With a centralized repository containing both Earth and Mars weather data spanning extended periods, it becomes possible to detect long-term climate trends and cycles that may not be apparent when analyzing data from a single planet in isolation. This can provide insights into the broader dynamics of planetary climate systems.

- A **unified data system** encourages collaboration between planetary scientists, climatologists, data scientists, and other researchers. Sharing a common platform facilitates the exchange of ideas and methodologies, fostering innovative approaches to studying and mitigating climate change. This is a key benefit of this centralized system that we identified from the start in this project and that shaped our choice of the personas.

5 Conclusions

This project aimed to investigate the potential of using Martian weather data to enhance our understanding of atmospheric dynamics on Earth and improve our ability to predict and mitigate extreme weather events. Through the integration of weather data from Mars and Earth into a Data Warehouse and Data Lake system, we were able to conduct comprehensive analysis of underlying climate trends and interactions.

Our research revealed both similarities and differences in the temperature patterns observed on Mars and Earth. While both planets exhibit cyclic fluctuations in temperature due to their respective rotations and orbital cycles, the magnitude and extremity of these fluctuations differ significantly. Mars experiences more extreme temperature variations due to its thin atmosphere and greater distance from the Sun.

Statistical analysis of the temperature data provided valuable insights into the seasonal variations, volatility, and long-term trends on each planet. These findings can aid in refining predictive models for Earth's changing climate and inform planning for future human activities on Mars.

However, our correlation and regression analysis indicated a lack of significant direct relationships between Martian weather parameters and Earth's atmospheric conditions. This suggests that the Martian data, within the studied timeframe and variables, may have limited practical value in directly predicting or mitigating extreme weather events on Earth.

Nonetheless, the integration of weather data from both planets into a unified Data Warehouse and Data Lake system facilitated centralized data management, comparative analysis, and the identification of long-term climate trends. This approach encourages collaboration among researchers, fostering innovative approaches to studying and mitigating climate change.

5.1 Future Work

While this study provided valuable insights, there are several avenues for future research and improvement:

1. **Expand the scope of data collection**: Incorporate additional weather parameters and extend the timeframe of data collection to capture a more comprehensive picture of planetary climate dynamics. This could potentially reveal previously undetected correlations or patterns.
2. **Leverage data from other planets**: Integrate weather data from other planets in our solar system to further enhance our understanding of atmospheric processes and refine predictive models.
3. **Improve remote sensing technologies**: Invest in advanced remote sensing technologies to improve the accuracy and resolution of data collected from both Earth and Mars, enabling more precise analyses and simulations.
4. **Develop specialized climate models**: Utilize the integrated data to develop specialized climate models tailored to each planet's unique atmospheric conditions, accounting for factors such as atmospheric composition, surface features, and orbital dynamics.
5. **Explore interdisciplinary collaborations**: Further foster collaborations between planetary scientists, climatologists, data scientists, and other relevant disciplines to leverage diverse perspectives and expertise in addressing the complex challenges of climate change.

6. **Investigate the impact of solar activity**: Conduct in-depth analyses of the influence of solar flares and other space weather phenomena on the atmospheric dynamics of both Earth and Mars, potentially revealing new insights into the interconnectedness of our solar system.

By continuing to explore the connections between planetary weather patterns and leveraging the power of data integration and advanced analytics, we can deepen our understanding of Earth's climate and develop more effective strategies for mitigating the impacts of climate change.

References

[1] S. George Philander. *Is the Temperature Rising?: The Uncertain Science of Global Warming*. Princeton University Press, Princeton, NJ, 2018.

[2] James L. Green, Chuanfei Dong, Michael Hesse, C. Alex Young, and Vladimir Airapetian. Space weather observations, modeling, and alerts in support of human exploration of mars. *Frontiers in Astronomy and Space Sciences*, 9:n/a, 2022.

[3] D.R. Ely. *Exploring Mars: The Red Planet Revealed*. David R Ely, 2024.

[4] Christina Plainaki, Jean Lilensten, Aikaterini Radioti, Maria Andriopoulou, Anna Milillo, Tom A. Nordheim, Iannis Dandouras, Athena Coustenis, Davide Grassi, Valeria Mangano, Stefano Massetti, Stefano Orsini, and Alice Lucchetti. Planetary space weather: scientific aspects and future perspectives. *Journal of Space Weather and Space Climate*, 6(A31):56, 2016.

[5] A. Mittelholz, C. L. Johnson, M. Fillingim, S. P. Joy, J. Espley, J. Halekas, S. Smrekar, and W. B. Banerdt. Space weather observations with insight. *Geophysical Research Letters*, 48(22):e2021GL095432, 2021. e2021GL095432 2021GL095432.

[6] Erika Palmerio, Christina O Lee, Ian G Richardson, Teresa Nieves-Chinchilla, Luiz F G Dos Santos, Jacob R Gruesbeck, Nariaki V Nitta, M Leila Mays, Jasper S Halekas, Cary Zeitlin, Shaosui Xu, Mats Holmström, Yoshifumi Futaana, Tamitha Mulligan, Benjamin J Lynch, and Janet G Luhmann. Cme evolution in the structured heliosphere and effects at earth and mars during solar minimum. *Space Weather*, 20(9):n/a, 2022.

[7] Hui Huang, Jianpeng Guo, Christian Mazelle, Emmanuel Penou, Haibo Lin, and Dan Zhao. Properties of interplanetary fast shocks close to the martian environment. *The Astrophysical Journal*, 914(1):14, 2021.

[8] C. O. Lee, T. Hara, J. S. Halekas, E. Thiemann, P. Chamberlin, F. Eparvier, R. J. Lillis, D. E. Larson, P. A. Dunn, J. R. Espley, J. Gruesbeck, S. M. Curry, J. G. Luhmann, and B. M. Jakosky. Maven observations of the solar cycle 24 space weather conditions at mars. *Journal of Geophysical Research: Planets*, 122(1):e2016JA023495, 2017.

A Additional Lambda Functions and Graphs

Listing 4: Mars Lambda function

```python
import boto3
import pandas as pd
import os
from io import StringIO

s3_client = boto3.client('s3')

def lambda_handler(event, context):

    # Get the bucket name and key from the event (assuming an S3 trigger)
    bucket = os.environ['BUCKET_NAME']
    key = "weather_mars_combined.csv"

    try:
        # Read the CSV data from S3
        response = s3_client.get_object(Bucket=bucket, Key=key)
        csv_data = response['Body'].read().decode('utf-8')

        # Create a DataFrame
        df = pd.read_csv(StringIO(csv_data))
        print(df.head(5))
        print(df.columns)

        # Clean the data (remove rows with NaN values)
        df.drop(columns='wind_speed', inplace=True)
        df.dropna(subset=['min_temp','max_temp','pressure'], how='all', inplace=True)

        print(df.info())
        # Convert the DataFrame back to CSV
        csv_cleaned = df.to_csv(index=False)

        # Write the cleaned data back to S3
        output_key = "cleaned/" + key # Save to a different location to avoid overwriting
        s3_client.put_object(Bucket=bucket, Key=output_key, Body=csv_cleaned)

        return {
            'statusCode': 200,
            # 'body': 'Data read into csv'
            'body': 'Data cleaned and saved to {}'.format(output_key)
        }

    except Exception as e:
        return {
            'statusCode': 500,
            'body': str(e)
        }
```

Listing 5: Earth (Open-Meteo) Lambda function

```python
import json
import openmeteo_requests # Open-Meteo weather API
import requests_cache # chaching responses to reduce repetitive requests
import pandas as pd
from datetime import datetime, timezone, timedelta
import time
from retry_requests import retry
import boto3
```

```
s3 = boto3.client('s3')


def lambda_handler(event, context):
    openmeteo = openmeteo_requests.Client()
    coordinates = [
     {"latitude": event["latitude"], "longitude": event["longitude"]}
    ] # this information comes from the Payload of the Lambda Invoke in the Step Function
    url = "https://archive-api.open-meteo.com/v1/archive"

    now_utc = datetime.now(timezone.utc)

    end_date = now_utc.strftime("%Y-%m-%d")

    start_date = (now_utc - timedelta(days=7)).strftime("%Y-%m-%d")

    base_params = {
        "start_date": start_date,
        "end_date": end_date,
        "hourly": ["temperature_2m", "relative_humidity_2m", "rain", "
            ↪ direct_radiation_instant"]
    }

    results = []

    for coords in coordinates:
        params = base_params.copy()
        params["latitude"] = coords["latitude"]
        params["longitude"] = coords["longitude"]
        responses = openmeteo.weather_api(url, params=params)
        response = responses[0]
        hourly = response.Hourly()
        hourly_temperature_2m = hourly.Variables(0).ValuesAsNumpy()
        hourly_relative_humidity_2m = hourly.Variables(1).ValuesAsNumpy()
        hourly_rain = hourly.Variables(2).ValuesAsNumpy()
        hourly_direct_radiation_instant = hourly.Variables(3).ValuesAsNumpy()
        hourly_data = {
            "date": pd.date_range(
                start=pd.to_datetime(hourly.Time(), unit="s", utc=True),
                end=pd.to_datetime(hourly.TimeEnd(), unit="s", utc=True),
                freq=pd.Timedelta(seconds=hourly.Interval()),
                inclusive="left"
            ).strftime('%Y-%m-%d %H:%M:%S').tolist(), # Convert DatetimeIndex to strings
            "latitude": params["latitude"],
            "longitude": params["longitude"],
            "temperature_2m": hourly_temperature_2m.tolist(),
            "relative_humidity_2m": hourly_relative_humidity_2m.tolist(),
            "rain": hourly_rain.tolist(),
            "direct_radiation_instant": hourly_direct_radiation_instant.tolist()
        }
        results.append({
            "coordinates": coords,
            "weather_data": hourly_data
        })
    results_json = json.dumps(results)

    obj_name = "weather_data_" + event["city"].lower() + ".json"

    bucket_name = "openmeteo-bucket"
    object_key = obj_name
    s3.put_object(Body=results_json, Bucket=bucket_name, Key=object_key)
    print("Data saved to S3 bucket successfully")
    return {
        'statusCode': 200,
        'body': json.dumps('Data saved to S3 bucket successfully')
    }
```

Listing 6: Solar flare lambda function

```
import requests
import json
import pandas as pd
from datetime import datetime, timedelta
import logging
import boto3
import os

# Initialize S3 client
s3 = boto3.client('s3')

# Setup logger
logging.basicConfig(level=logging.INFO)
logger = logging.getLogger()

def lambda_handler(event, context):
    api_key = os.environ['API_KEY']
    bucket_name = os.environ['BUCKET_NAME']

    end_date = datetime.now()
    start_date = end_date - timedelta(days=30)
    start_date_str = start_date.strftime('%Y-%m-%d')
    end_date_str = end_date.strftime('%Y-%m-%d')

    solar_flare_url = f'https://api.nasa.gov/DONKI/FLR?startDate={start_date_str}&endDate={
        ↪ end_date_str}&api_key={api_key}'

    try:
        response = requests.get(solar_flare_url)
        response.raise_for_status() # Raises an HTTPError for bad responses
        data = response.json()

        # Process and save data
        if data:
            normalized_data = pd.json_normalize(data)
            file_name = f"solar_data_{start_date_str}_to_{end_date_str}.csv"
            file_path = f"/tmp/{file_name}"
            normalized_data.to_csv(file_path, index=False)

            # Upload to S3
            s3.upload_file(file_path, 'solarflaredata', file_name)
            logger.info(f"Data saved to S3 successfully, number of entries: {len(data)}")
        else:
            logger.info("No data available for the given date range.")

        return {
            'statusCode': 200,
            'body': json.dumps('Data processing and saving successful!')
        }

    except requests.exceptions.RequestException as e:
        logger.error(f"Request failed: {e}")
        return {
            'statusCode': 500,
            'body': json.dumps('Data processing failed!')
        }
```

Listing 7: Data Ingestion

```
import boto3 # access the data
import pandas as pd
import json
from sqlalchemy import create_engine
import os

# -------------------------------------------------------
# preprocessing functions

def process_flare_data(df):
    print(df.head())
    if 'Unnamed: 0' in df.columns:
        df = df.drop(columns=['Unnamed: 0'])
        print(df.head())
    # Convert date fields to datetime
    df['beginTime'] = pd.to_datetime(df['beginTime'], format='%Y-%m-%dT%H:%MZ', errors='
        ↪ coerce')
    df['endTime'] = pd.to_datetime(df['endTime'], format='%Y-%m-%dT%H:%MZ', errors='coerce')

    # Map flare classes to intensity descriptions
    flare_intensity = {
        'X': 'Most intense',
        'M': 'Medium',
        'C': 'Small',
        'B': 'Very Small',
        'A': 'Smallest'
    }
    df['intensity'] = df['classType'].str[0].map(flare_intensity)
    print('columns:')
    print(df.columns)
    return df.reset_index(drop=True)

def process_kaggle_data(df):
    df.drop(columns='wind_speed', inplace=True)
    df.dropna(subset=['min_temp','max_temp','pressure'], how='all', inplace=True)
    df['terrestrial_date'] = pd.to_datetime(df['terrestrial_date'])
    if 'season' in df.columns:
        df = df.rename(columns={'season': 'month'})
    df = df.drop(columns=['index'])
    print(df.head())
    return df.reset_index(drop=True)

def process_meteo_data(json, file_name):
    df_weather_data = pd.DataFrame(json)
    df_weather_data['location'] = file_name.split('.json')[0].split('_')[-1]
    df_weather_data = df_weather_data.dropna()
    df_weather_data['date'] = pd.to_datetime(df_weather_data['date'], unit='ms')
    return df_weather_data.reset_index(drop=True)
# -------------------------------------------------------

def lambda_handler(event, context):
    # put aws credintials in a json file and read them
    with open('../awsconfig.json', 'r') as jsonfile:
```

```
    data = json.load(jsonfile)
YOUR_ACCESS_KEY = data['aws_access_key_id']
YOUR_SECRET_KEY = data['aws_secret_access_key']
YOUR_REGION = data['region_name']
SESSION_TOKEN = data['aws_session_token']

# AWS credentials
s3 = boto3.client('s3',
                region_name = YOUR_REGION,
                aws_access_key_id = YOUR_ACCESS_KEY,
                aws_secret_access_key = YOUR_SECRET_KEY,
                aws_session_token = SESSION_TOKEN
                )

# bucket names
buckets = ['solarflaredata'] # 'marskaggledata', 'meteoswissdata',

# Connect to Aurora PostgreSQL
# how to write it: engine = create_engine('postgresql+psycopg2://username:password@host:
    ↪ port/database')
# for security reasons it's added to the config file
engine_conn = data['engine_conn']
engine = create_engine(engine_conn)

for bucket in buckets:
    if bucket == 'solarflaredata':
        response = s3.list_objects_v2(Bucket=bucket)
        files = [file['Key'] for file in response.get('Contents', []) if file['Key'].
            ↪ endswith('.csv')]

        for file_key in files:
            obj = s3.get_object(Bucket=bucket, Key=file_key)
            data = pd.read_csv(obj['Body'])
            print("Initial Data Loaded into DataFrame:")
            print(data.head())
            processed_data = process_flare_data(data)
            processed_data.columns = [column.lower() for column in processed_data.columns
                ↪ ]
            # Load processed data into the database
            processed_data.to_sql('solar_flare_data', con=engine, if_exists='append',
                ↪ index=False)
            print(f"Data from {file_key} in {bucket} processed and loaded into database."
                ↪ )

    elif bucket == 'marskaggledata':
        response = s3.list_objects_v2(Bucket=bucket)
        files = [file['Key'] for file in response.get('Contents', []) if file['Key'].
            ↪ endswith('.csv')]

        data_frames = []

        for file_key in files:
            obj = s3.get_object(Bucket=bucket, Key=file_key)
            data = pd.read_csv(obj['Body'])
            processed_data = process_kaggle_data(data)
            data_frames.append(data)

        concatenated_df = pd.concat(data_frames, ignore_index=True)
        concatenated_df = concatenated_df.drop(columns=['index'])
        if 'season' in concatenated_df.columns:
            concatenated_df = concatenated_df.drop(columns=['season'])
            cleaned_data = concatenated_df.drop_duplicates().reset_index(drop=True)
            cleaned_data.to_sql('mars_weather', con=engine, if_exists='append', index=False)
            print(f"Data from {file_key} in {bucket} processed and loaded into database.")

        elif bucket == 'meteoswissdata':
            response = s3.list_objects_v2(Bucket=bucket)
            files = [file['Key'] for file in response.get('Contents', []) if file['Key'].
                ↪ endswith('.json')]

            for file_key in files:
                obj = s3.get_object(Bucket=bucket, Key=file_key)
                json_data = json.loads(obj['Body'].read().decode('utf-8'))
                processed_data = process_meteo_data(json_data, file_key)
                processed_data.to_sql('weather_data', con=engine, if_exists='append', index=
                    ↪ False)
                print(f"Data from {file_key} in {bucket} processed and loaded into database."
                    ↪ )


    return {
        'statusCode': 200,
        'body': json.dumps('Data processed and stored successfully!')
    }
```

B SQL code queries

Listing 8: Create database tables

```
CREATE TABLE mars_weather (
        primary_key SERIAL PRIMARY KEY,
    id INT NOT NULL,
    terrestrial_date TIMESTAMP,
    sol INT,
    ls INT,
    month VARCHAR(255),
    min_temp FLOAT,
    max_temp FLOAT,
    pressure FLOAT,
    atmo_opacity VARCHAR(255)
);

CREATE TABLE solar_flare_data (
    id SERIAL PRIMARY KEY,
    flrID VARCHAR(255),
    instruments VARCHAR(255),
    beginTime TIMESTAMP,
    peakTime TIMESTAMP,
    endTime TIMESTAMP,
    classType VARCHAR(255),
    sourceLocation VARCHAR(255),
    activeRegionNum FLOAT,
    note TEXT,
    linkedEvents TEXT,
    submissionTime VARCHAR(255),
    link VARCHAR(255),
    intensity VARCHAR(255)
);

CREATE TABLE weather_data (
    id SERIAL PRIMARY KEY,
    date TIMESTAMP,
    latitude FLOAT,
    longitude FLOAT,
    temperature_2m FLOAT,
    relative_humidity_2m FLOAT,
    rain FLOAT,
    direct_radiation_instant FLOAT,
    location VARCHAR(255)
);
```

C Credentials

Listing 9: DB credentials

```
awsconfig:
{
    "aws_access_key_id" :"ASIA2CEWIKRHPVLQQVN5",
    "aws_secret_access_key" :"A8cHkVwCsZ9JfK9GY8nRb5xrkQC/vX6dWDCC/ydW",
    "region_name" :"us-east-1",
    "aws_session_token" :"IQoJb3JpZ2luX2VjELH//////////wEaCXVzLXdlc3
        ↪ QtMiJIMEYCIQDXyIzZtXUPpiataOJAuKPMOJ2KcQHYkoMwJZyEIt78ZAIhAN22wZw5oUZU+9
        ↪ sduqzwZReac46YCVLyzCMqZhzHF8I9Kq4CCCoQAhoMNjkxODA0ODUzMzI2IgxYCzM2KMNxUZ4
        ↪ BUbwqiwKOFy7qDxfJV0jsGjPr11ma2FFcPPLlKBvlz6iEK3bSWEba9gXNXEeHzQJqRXDKzN6kq3
        ↪ mZWFiFmfyskjQw+ONlQJisT0AkWSjDV0Fx8p9M3EUAlAlqnp2xuDQVIbhDk29a7
        ↪ oOEYmgZFJKRgzrcVuvqOccrlF4WxGcPUQO64MjxPbbB9VTuOlwgD6HegTN2uya24ab0H4Lu0
        ↪ JylHjEaPyvwWETC8LnKsXNYREw0EWZJrUupFOPu5AGFyAuW9sinxlHMMOyUTODGcQFPEO1
        ↪ cUAWcq3lI/Hd2MucSjvPBgaj9BrBZpgup7EM9RifYzXF85EFKMAZJj26Oeo/e/UBdLLm0l2cG1
        ↪ MXDM2vY00ow2aKssgY6nAEKH9vXWRAlZLHKTOPBl6JIrEdnFfllYayJ1Swljw6lH53UIJn9hmag
        ↪ 9bXKwH6yXub2YS643Zy7Gk9MekNlXtlT33QuGSJmGQXxbAm/yH6Br/+dUwwjdjN0FzEjdLRznT8
        ↪ XU2GK0gu5CSuvDJQE519+LHX8lgfHoTD4apsaF5kQsVDByNo91O+jFrM6rSmTQeA7HxJKhc2SVw
        ↪ 3reCo=",
    "engine_conn" :"postgresql+psycopg2://postgres:Inter1986@database-2-instance-1.
        ↪ cxyltt2bn4hq.us-east-1.rds.amazonaws.com:5432/nasajaa2"
}

awsacademy:
{
Ansam: {email:zedanansam@gmail.com, password:Tvu-+5,$RCNcW&U},
Joana: {email:joana.duartedossantos@stud.hslu.ch, password:DWL_jo@987},
Albin: {email:albin.plathottathil@gmail.com, password:Password123}
}
```

D Evaluation Grid

Evaluation Criteria	Maximum possible points	Joana	Ansam	Albin
Introduction				
Table of contents	5	X	X	X
Topic introduction	5	X	X	X
Problem definition & Goals	5	X	X	X
Business/Research questions & Limitations	5	X	X	X
State of the art				
Literature review (Academic / Business)	5	X	X	X
Methodology				
Data Lake	5	X	X	X
Data lake architecture, system, etc. in technical details	5	X	X	X
Design choices / trade-offs / assumptions	5	X	X	X
Data sources definitions (technical)	5	X	X	X
Data imputation, data cleaning per data source	5	X	X	X
Data analysis per data source	5	X	X	X
The database schema, model, table descriptions, and files.	5	X	X	X
Data Lake consumers (personas)	5	X	X	X
Data Warehouse	5	X	X	X
Data warehouse architecture, system, etc. in technical details	5	X	X	X
Visualization design justification	5	X	X	X
Tableau visualization (with public link)	5	X	X	X
Answers to research/business questions	5	X	X	X
Conclusions				
Proper discussion of the solution	5	X	X	X
Proper discussions of the project outcomes.	5	X	X	X
Future work	5	X	X	X

Table 2: Assessment Table