SAN JOSE STATE UNIVERSITY

Database Systems for Analytics (DATA 225)

Instructor – Simon Shim

Report on

Global Weather Repository

Group 8

Submitted By

Aiswarya Raghavadesikan

Neha Thakur

Saumya Varshney

Shikha Singh

Venkata Sai Sreelekha Gollu

TABLE OF CONTENT

I. Introduction

The phenomenon of weather significantly shapes our daily experiences. Key weather indicators such as temperature, precipitation, wind speed, and pressure play pivotal roles in various sectors, including agriculture, travel, and supply chain management. By delving into the analysis of temperature trends spanning multiple years, valuable insights can be extracted and visualized to benefit different industries. The escalating global temperatures have become a cause for widespread apprehension.

With the weather data analysis, we can deep dive into the unusual and concerning weather patterns to understand the ways to tackle natural calamities. Disaster preparedness could be done in advance with accurate predictions based on historical weather data. This could bring down the mortality rate in the event of a natural calamity.

We are utilizing Global Historical Climatology Network (GHCN) data from the National Oceanic and Atmospheric Administration (NOAA) website. In this project we are working on Cloud Analytics and Data Warehouse Implementation by building a data pipeline to analyze archival and real time weather data and gather useful insights from this data.

II. Dataset Description

We've sourced our data directly from the open data registry on AWS, courtesy of the National Oceanic and Atmospheric Administration (NOAA). This governmental entity diligently collects weather data from stations across the globe. Specifically, we've tapped into the NOAA Global Historical Climatology Network Daily (GHCN-D) dataset, offering a comprehensive overview of worldwide weather data.

Dataset link:

- https://registry.opendata.aws/noaa-ghcn/
- https://noaa-ghcn-pds.s3.amazonaws.com/index.html

The data is segregated into two distinct components. The initial segment encompasses comprehensive information pertaining to weather stations distributed globally. The subsequent segment comprises specific weather-related data, including minimum temperature, maximum temperature, snow depth, snowfall, and precipitation. This data is systematically collected by stations and subsequently stored in designated source locations.

Stations Details:

ID - 11 character station identification code
LATITUDE - is latitude of the station (in decimal degrees).
LONGITUDE - is the longitude of the station (in decimal degrees).
ELEVATION - is the elevation of the station (in meters, missing = -999.9).
STATE - is the U.S. postal code for the state (for U.S. stations only).
NAME - is the name of the station.
GSN FLAG - is a flag that indicates whether the station is part of the GCOS Surface Network (GSN)
HCN/CRN FLAG - is a flag that indicates whether the station is part of the U.S. Historical Climatology Network (HCN) or U.S. Climate Refererence Network (CRN).
WMO ID - is the World Meteorological Organization (WMO) number for the station.

Year Details:

ID - is the station identification code.
YEAR - is the year of the record.
MONTH - is the month of the record.
ELEMENT - is the element type.There are five core elements as well as a number of addition elements. The five core elements are:

- PRCP = Precipitation (tenths of mm)
- SNOW = Snowfall (mm)
- SNWD = Snow depth (mm)
- TMAX = Maximum temperature (tenths of degrees C)
- TMIN = Minimum temperature (tenths of degrees C)

III. Cloud Architecture

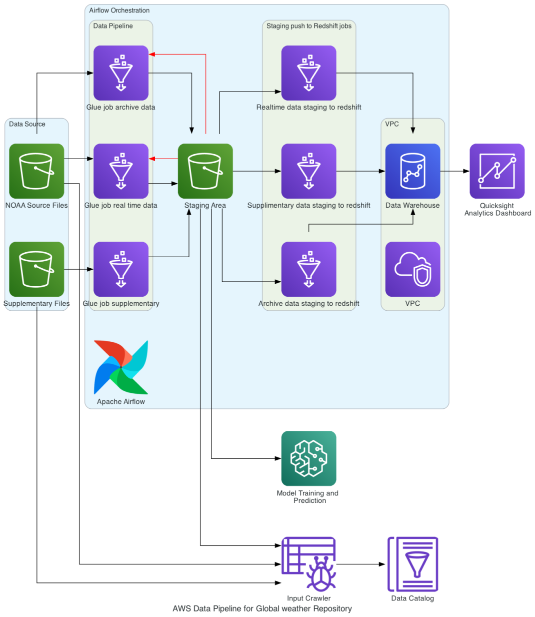


Fig. 1. Cloud Architecture

We have used below mentioned Amazon Web Services in this project to build the cloud architecture.

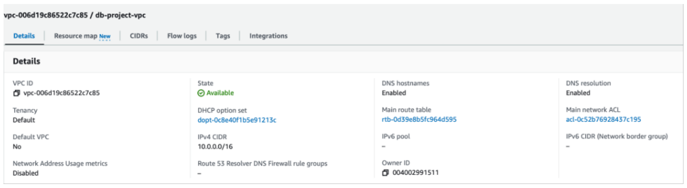
- VPC - A Virtual Private Cloud (VPC) is a virtual network dedicated to AWS (Amazon Web Services) account.
- S3 - Amazon Simple Storage Service (Amazon S3) is a scalable object storage service offered by Amazon Web Services (AWS). It is designed to store and retrieve any amount of data from anywhere on the web. Here we retrieved the data both historic and real time from S3

in which real time has daily data for the year 2023. The historical data we took from 2010 to 2022 is 12years. Here in the pipeline, we have loaded the data from s3.

- Glue - AWS Glue is a fully managed extract, transform, and load (ETL) service that makes it easy to prepare and load data for analysis, perform transformations to clean data and enforce integrity constraints, and then load it into staging area in parquet format.
- Glue Crawler - AWS Glue Crawler is a crucial component of the AWS Glue service that automatically discovers, catalogs, and organizes metadata about data sources.
- Glue Data Catalog - The Glue Data Catalog in AWS Glue is a centralized metadata repository that stores metadata about various data sources, making it easier to discover, manage, and analyze.
- Redshift - AWS Redshift is a cloud-based data warehouse service for powerful and scalable analytics, offering fast SQL queries and high-performance analysis on datasets.
- Apache Airflow - Apache Airflow is an open-source platform to schedule and monitor workflows. It is used to orchastrate our data pipe line.
- Sage Maker - Amazon SageMaker is a fully managed machine learning service that simplifies the cycle of machine learning models, from data labeling and model training to deployment and real-time predictions at scale.
- Quick Sight - It turns information into meaningful insights with easy-to-create, interactive visualizations, making data analysis for understanding and making decisions out of it.

IV. VPC Connection for cloud instance

A Virtual Private Cloud (VPC) is a virtual network dedicated to AWS (Amazon Web Services) account. It provides a logically isolated section of the AWS Cloud where we can launch AWS resources in a virtual network that we define. With a VPC, we have control over network environment, including the selection of your IP address range, creation of subnets, and configuration of route tables and network gateways.
We have created out VPC using amazon VPC. Below are the configurations which we have done.



vpc-006d19c86522c7c85 / db-project-vpc

Details

VPC ID vpc-006d19c86522c7c85	State Available	DNS hostnames Enabled	DNS resolution Enabled
Tenancy Default	DHCP option set dopt-0c8e40f1b5e91213c	Main route table rtb-0d39e8b5fc964d595	Main network ACL acl-0c52b76928437c195
Default VPC No	IPv4 CIDR 10.0.0.0/16	IPv6 pool –	IPv6 CIDR (Network border group) –
Network Address Usage metrics Disabled	Route 53 Resolver DNS Firewall rule groups –	Owner ID 004002991511	

Fig. 2. VPC connection in AWS instance

V. ETL Pipeline

A. *Data Sources*

NOAA GHCN-D AWS Bucket - This public bucket contains all the data collected by NOAA for the GHCN-D dataset from year 1750 to 2023. We are only considering the data from 2010 to 2023 for this project.

2010 - 2022 data comprises our historical data. This data is not updated unless a correction is issued and serves as a historical record of world climatology data. Each year is stored as a CSV, and the headers comprise of

- station id - id for the weather station where the data was collected from
- date id - date at which the data was collected in the format yyyymmdd (20231130)
- element id / datatype id - what category of data was recorded (maximum temperature, minimum temperature, precipitation, snow fall and snow depth)
- Data / value - the data recorded corresponding to element.

2023 data comprises our realtime data. This is a CSV which is updated daily for all the weather stations in the world. Additionally, we scraped a lot of supplemental data from https://www.ncdc.noaa.gov/cdo-web/webservices/v2#gettingStarted using REST APIs. This includes the following -

- stations.csv - data for each weather station like name, latitude, longitude and elevation
- locations.csv - names for each geographic location in the world, for example India, California, San Jose, Santa Clara county, Zip code 95129
- locationcategories.csv - name of geographic categories within which each location falls for example, countries, states, cities, climate regions, counties, zip code
- stationrelations.csv - relation between station, location and locationcategory, which allows us to identify all the stations within a geographic location
- dates.csv - every date from 2010 to 2023 along with information like year, quarter, week, month, day, day of year, month name, day name, leap year, which will allow us to perform various types of aggregate queries
- datatypes.csv - names and description of various types of data recorded at each station - temperature, precipitation, snowfall, moisture, etc.

B. *ETL Jobs*

1) Supplementary Data Ingestion Glue Job
   We load the supplementary data from its S3 bucket, apply transformations and enforce integrity constraints, and then load it into staging area in parquet format. This will be used in the processing of data in both realtime and archive data pipelines.
2) Archived Data Ingestion Glue Job
   Due to Glue resource and pricing constraints, we built a pipeline for each year between 2010-2022. Each pipelines loads the respective csv from the public bucket,

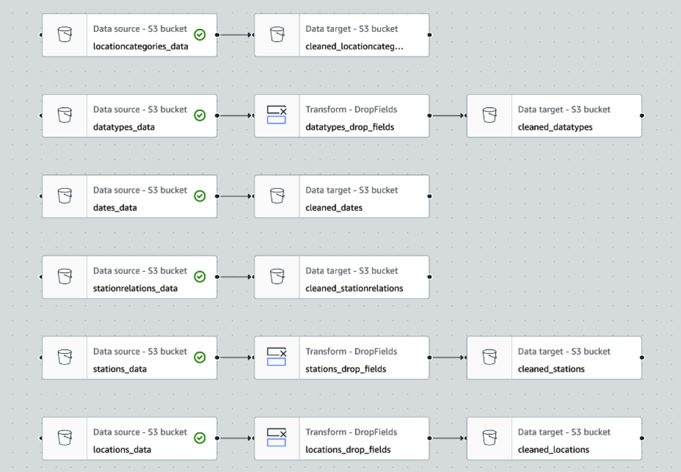


Fig. 3. Supplementary-data-pipeline in glue

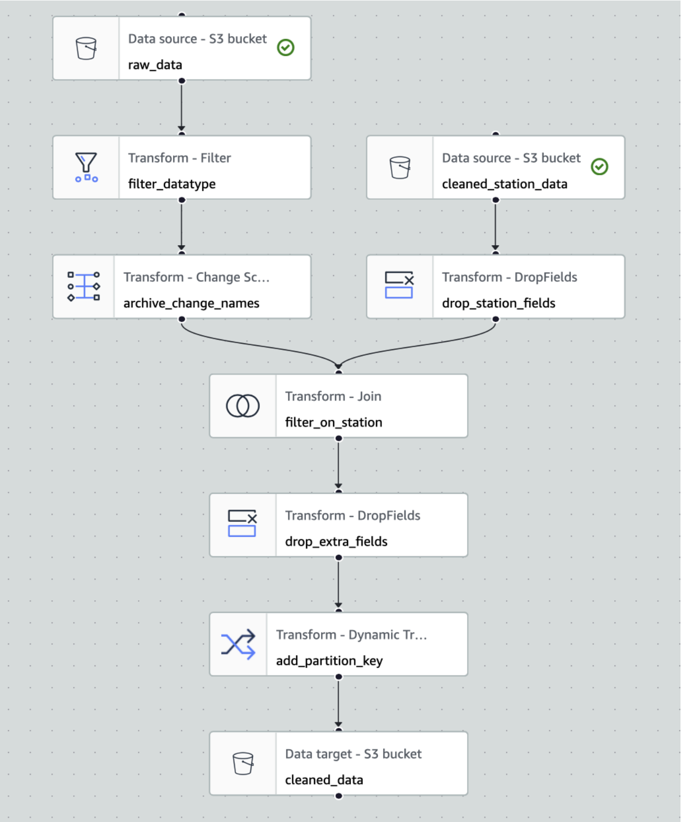


Fig. 4. 2010-data-pipeline in glue

transforms the schema for the warehouse compatibility, cleans up inconsistent data, enforces integrity constraints with the stations from supplementary data (only collect data for stations in supplementary data, drop data from all other stations), and loads them as year-based-partitioned parquet files in the staging area. This pipeline is supposed to run on demand, given the infrequent update nature of this data.

3) Realtime Data Ingestion Glue Job

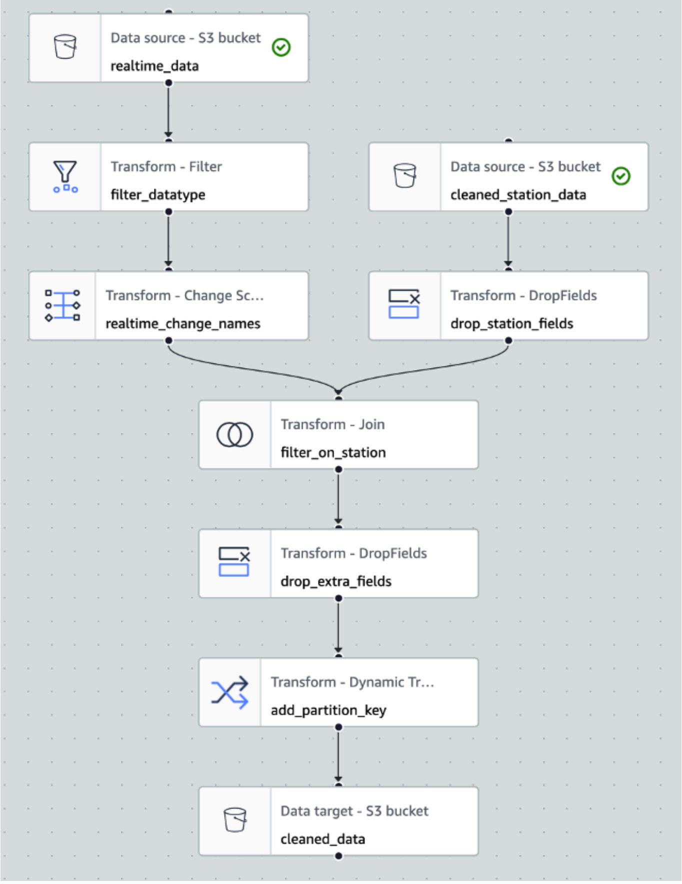


Fig. 5. realtime-data-pipeline in glue

This pipeline loads the 2023 csv from the public bucket, transforms the schema for the warehouse compatibility, cleans up inconsistent data, enforces integrity constraints with the stations from supplementary data (only collect data for stations in supplementary data, drop data from all other stations), and loads them as parquet files in the 2023 partition of staging area. This pipeline is supposed to run daily, given the frequent update nature of this data.

4) Supplementary Data Redshift Ingestion Glue Job
We extract the supplementary data from staging area, enforce the warehouse schema (transform), and load them into the Redshift Data Warehouse sitting inside a private VPC. These will act as the dimension tables for our snowflake schema. The job is configured to

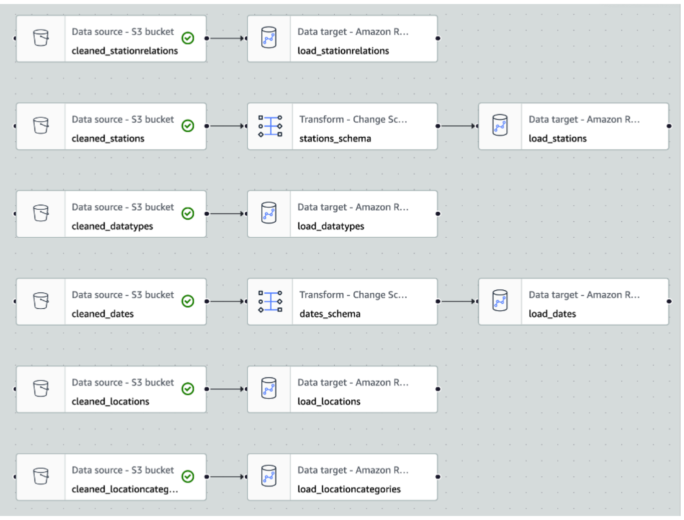


Fig. 6. supplementary-data-to-redshift in glue

ensure there is no data duplication in redshift upon data insertion.

5) Archive Data Redshift Load Job

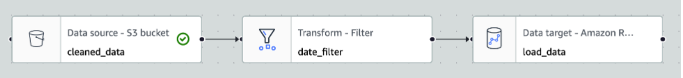


Fig. 7. archive-data-to-redshift in glue

We extract the archive data tables from staging area, enforce the warehouse schema (transform), and load them into the Redshift Data Warehouse sitting inside a private VPC. This will act as the fact table for our snowflake schema. The job is configured to ensure there is no data duplication in redshift upon data insertion.

6) Realtime Data Redshift Load Job

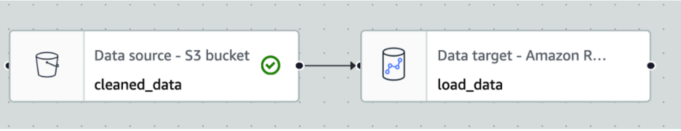


Fig. 8. realtime-data-to-redshift in glue

We extract the realtime data tables from staging area, enforce the warehouse schema (transform), and load them into the Redshift Data Warehouse sitting inside a private VPC. This will act as the fact table for our snowflake schema. The job is configured to ensure there is no data duplication in redshift upon data insertion.

VI. Data-warehouse Implementation

We have used Amazon Redshift for data-ware house implementation. Amazon Redshift is a fully managed, cloud-based data warehouse service provided by Amazon Web Services (AWS). It is designed for high-performance analysis and querying of large datasets using SQL queries.

A. *Data modeling*

We have implemented our OLAP (Online analytical processing) database using snowflake schema with fact and dimensions tables.A Snowflake Schema is a database schema commonly employed in data warehousing, particularly when designing structures for large-scale analytical databases. The Snowflake Schema takes normalization a step further, resulting in a more intricate, normalized architecture. This schema is aptly named for its resemblance to a snowflake when visualized, with multiple branches and levels radiating from a central point.

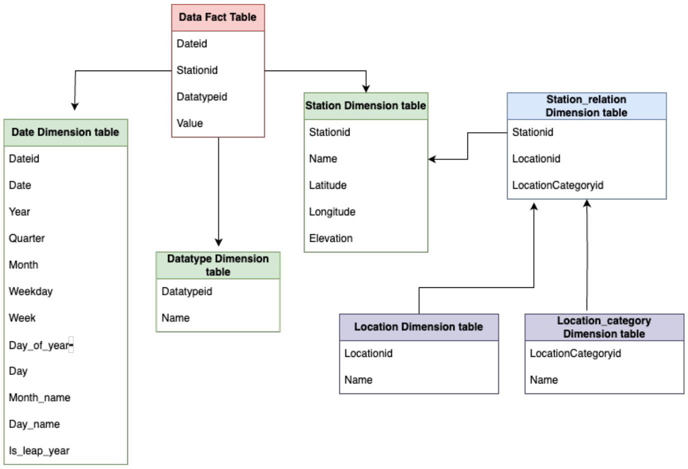


Fig. 9. Data-warehouse snowflake schema

The snowflake schema consists of following tables –

- Data (Fact table)
- Datatypes (Dimension table)
- Locations (Dimension table)
- Stations (Dimension table)
- Dates (Dimension table)
- Locationcategories (Dimension table)
- Stationrelations (Dimension table)

Attributes of the tables are explained as following -

1) **Data table:** This is the fact table contains following attributes -
   - Dateid- It is a identification field for date
   - Stationid- This represents the station identification code for the station under consideration while recording the climate parameters.
   - Name- Dateid- It is a identification field for date
   - Datatypeid- This contains element code indicating the element for which the name would be stored. For ex: TMAX, TMIN, SNOW, SNDP AND PRCP.
   - Value- This consists of measured data values for various datatypeid fields.
2) **Date table:** This is the dimension table of Data table contains following attributes -
   - Dateid- It is a identification field for date

- Date- This is represented in the format MM/DD/YYYY
- Year- Year is recorded in format YYYY
- Quarter- It represents the quarter of the year for which climate parameters are recorded.
- Month- It represents the month of the year for which climate parameters are recorded.
- Week- It represents the week of the year for which climate parameters are recorded.
- Weekday- It represents the weekday of the week for which climate parameters are recorded.
- Day_of_year- It represents the day of the year for which climate parameters are recorded.
- Day- It shows the count of the days of year on which parameters are recorded.
- Month_name- It represents the name of the month for which climate parameters are recorded.
- Day_name- It represents the name of the day for which climate parameters are recorded.
- Is_leap_year- It represents if the year under consideration is a leap year or not in form of TRUE/FALSE.

3) **Datatype table:** This is the dimension table of data table contains following attributes -
    - datatypeId- This contains element code indicating the element for which the name would be stored. For ex: TMAX, TMIN, SNOW, SNDP AND PRCP.
    - Name- This attribute gives the description of element.
4) **Stations table:** This is the dimension table of data table contains following attributes -
    - stationId- This is a unique identifier for each station. It is a combination of letters and numbers that uniquely identifies a specific weather station or monitoring station.
    - Name-This column contains the name associated with each station. For weather stations, it could be the name of the city, region, or a specific site where the monitoring station is located.
    - Latitude-This represents the geographic north-south coordinate of the station, measured in decimal degrees. Positive values indicate northern latitudes, and negative values indicate southern latitudes.
    - Longitude- This represents the geographic east-west coordinate of the station, measured in decimal degrees. Positive values indicate eastern longitudes, and negative values indicate western longitudes.
    - Elevation- This is the altitude of the station above a reference point (usually sea level). It provides information about the vertical position of the station in the Earth's atmosphere.
5) **Stationrelation table:** This is the dimension table of station table contains following attributes -
    - Stationid- This represents the station identification code for the station under consideration while recording the climate parameters.
    - Locationid- This gives the location identification code for the location station under consideration while recording the climate parameters.
    - Locationcategoryid- This represents the location category identification code for the station under consideration while recording the climate parameters.
6) **Locations table:** This is the dimension table of stationrelation table contains following attributes -
    - LocationId- This is a unique identifier for each location. It could be a numerical code or a combination of letters and numbers that uniquely identifies a specific geographic point, such as a weather station or monitoring station.
    - Name- This column contains the name associated with each location. For weather stations, it could be the name of the city, region, or a specific site where the monitoring station is located.
7) **Locationcategory table:** This is the dimension table of stationrelation table contains following attributes -
    - locationcategoryid- locationcategoryid is the location category identification code for the location.
    - Name- This attribute represents the name of location categories used in the dataset.

Redshift clusters have been created in VPC along with appropriate permissions and subnet mappings. A database titled ”noaadbproject” and a schema titled ”noaaschema” are created within the cluster.

VII. Apache Airflow

Apache Airflow is an open-source platform designed for orchestrating complex workflows and data processing pipelines. It allows users to define, schedule, and monitor workflows as Directed Acyclic Graphs (DAGs) of tasks. Airflow is particularly popular for ETL (Extract, Transform, Load) processes, data warehousing, machine learning workflows, and other scenarios where complex data workflows need to be managed and automated.

We orchestrate different Glue Job components using 2 pipelines -

- **Real-time data pipeline:**

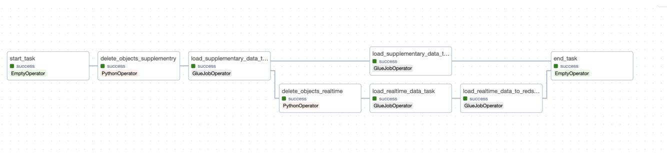


Fig. 10. Realtime Data Flow

This will clear the supplementary data and archived data (2010 to 2022) from staging area, run the supplementary data ingestion job, then run the archive data ingestion job followed by archive data redshift load job, while running the supplementary data redshift load job in parallel.

1) Start task: It is the dummy operator which is indicating the start of the flow.

2) Unload supplementary data objects: We used a python operator which facilitates the removal of supplementary data objects from the designated S3 staging bucket. The purpose of this step is to effectively clear the staging bucket, creating a preparatory state for the subsequent loading of updated data from the primary data source.

3) Loading supplementary data: We have used a Glue job operator which serves as the trigger mechanism for the execution of an AWS Glue job, specifically designed to perform transformations on supplementary data and prepare the data for staging.

4) Unload the real-time data objects: Python operator is used to enable the removal of real-time data objects from the designated S3 staging bucket.

5) Loading real-time data: An Glue job operator is triggered for the execution of a AWS Glue job designed specifically for performing transformations on real-time data and preparing it for staging.

6) Loading data to (Data-warehouse): Two glue jobs have been designed to load supplementary and real-time data to Redshift, thereby completing the orchestration process.

7) End task: This dummy operator is indicating the end of orchestration of data pipeline.

- **Archive data pipeline:**
  This will clear the supplementary data and archived data (2010 to 2022) from staging area, run the supplementary data ingestion job, then run the archive data ingestion job followed by archive data redshift load job, while running the supplementary data redshift load job in parallel.

1) Start task: It is the dummy operator which is indicating the start of the flow.

2) Unload supplementary data objects: We used a python operator which facilitates the removal of supplementary data objects from the designated S3 staging bucket. The purpose of this step is to effectively clear the staging bucket, creating a preparatory state for the subsequent loading of updated data from the primary data source.

3) Loading supplementary data: We have used a Glue job operator which serves as the trigger mechanism for the execution of an AWS Glue job, specifically designed to perform transformations on supplementary data and prepare the data for staging.

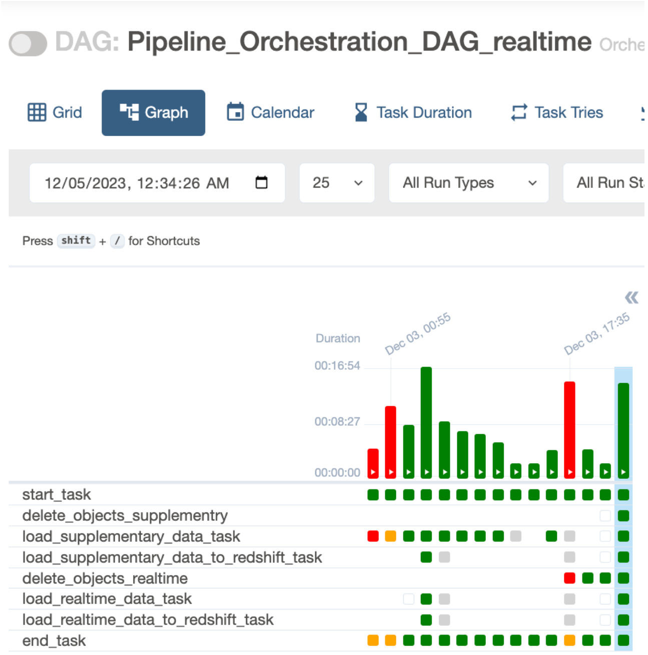


Fig. 11. Realtime data pipeline

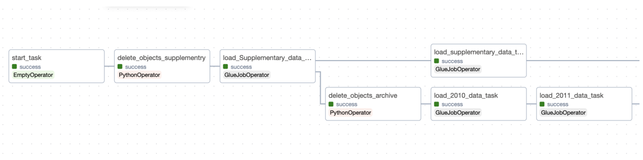


Fig. 12. Archival Data Flow

4) Unload the archival data objects: Python operator is used to enable the removal of archival data objects from year 2010-2022 from the designated S3 staging bucket.

5) Loading archival data: An Glue job operator is triggered for the execution of a AWS Glue job designed specifically for performing transformations on archival data and preparing it for staging.

6) Loading data to (Data-warehouse): Two glue jobs have been designed to load supplementary and archival data from year 2010 to 2022 to Redshift, thereby completing the orchestration process.

7) End task: This dummy operator is indicating the end of orchestration of data pipeline.

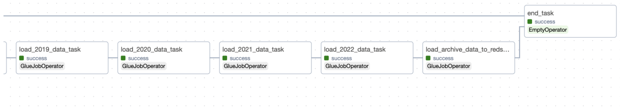
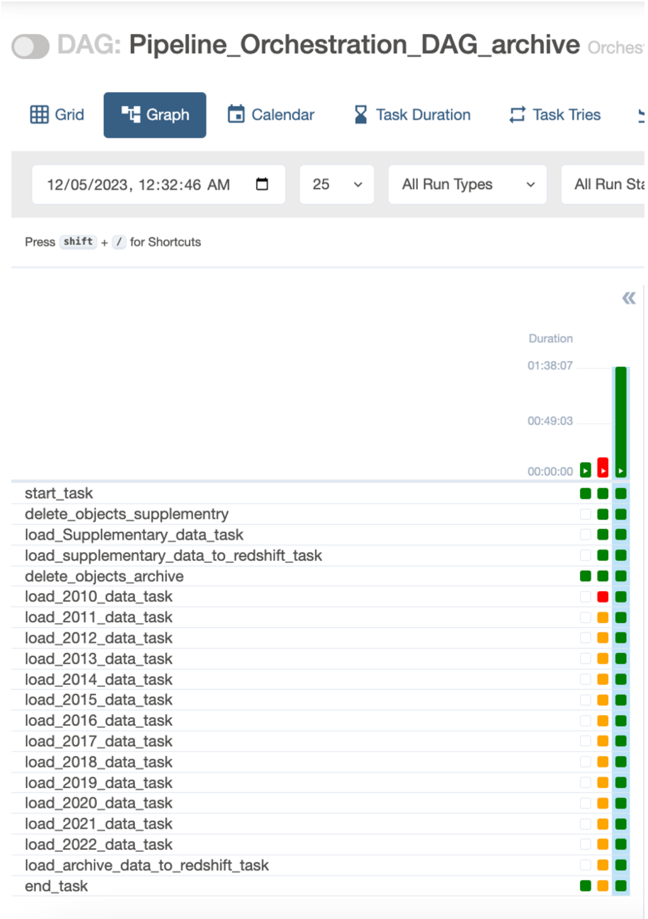


Fig. 13. Archival Data Flow



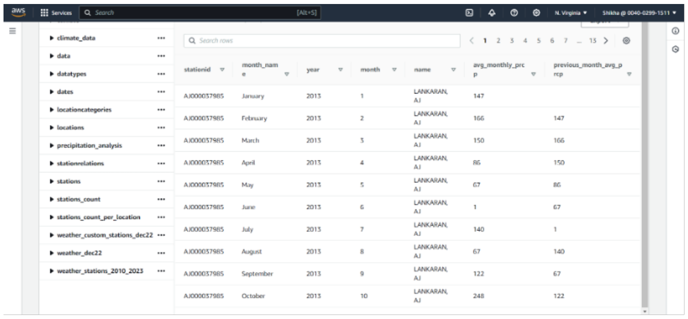
start_task
delete_objects_supplementry
load_Supplementary_data_task
load_supplementary_data_to_redshift_task
delete_objects_archive
load_2010_data_task
load_2011_data_task
load_2012_data_task
load_2013_data_task
load_2014_data_task
load_2015_data_task
load_2016_data_task
load_2017_data_task
load_2018_data_task
load_2019_data_task
load_2020_data_task
load_2021_data_task
load_2022_data_task
load_archive_data_to_redshift_task
end_task

Fig. 14. Archive data pipeline

VIII. Data Analysis and Visualizations

1) **Analysis of each month's precipitation and compare with previous month**

Create table noaadbproject.noaaschema.Precipitation_analysis AS WITH MonthlyAvg AS (SELECT d.stationid, da.month_name, da.year, da.month, s.name, AVG(d.value) AS avg_monthly_prcp FROM noaadbproject.noaaschema.Data d JOIN noaadbproject.noaaschema.dates da ON da.dateid = d.dateid JOIN noaadbproject.noaaschema.stations s ON d.stationid = s.id WHERE d.datatypeid = 'PRCP' AND da.year in (2013,2014,2015,2016,2017,2018,2019,2020,2021, 2022,2023) AND d.stationid = 'AJ000037985' GROUP BY d.stationid, da.year, da.month_name, da.month, s.name) SELECT stationid, month_name, year, month, name, avg_monthly_prcp, LAG(avg_monthly_prcp) OVER (PARTITION BY stationid ORDER BY year, month) AS previous_month_avg_prcp FROM MonthlyAvg ORDER BY year, month LIMIT 200;



stationid	month_name	year	month	name	avg_monthly_prcp	previous_month_avg_prcp
AJ000037985	January	2013	1	LANKARAN, AJ	147	
AJ000037985	February	2013	2	LANKARAN, AJ	166	147
AJ000037985	March	2013	3	LANKARAN, AJ	150	166
AJ000037985	April	2013	4	LANKARAN, AJ	86	150
AJ000037985	May	2013	5	LANKARAN, AJ	67	86
AJ000037985	June	2013	6	LANKARAN, AJ	1	67
AJ000037985	July	2013	7	LANKARAN, AJ	140	1
AJ000037985	August	2013	8	LANKARAN, AJ	67	140
AJ000037985	September	2013	9	LANKARAN, AJ	122	67
AJ000037985	October	2013	10	LANKARAN, AJ	248	122

Fig. 15. Query results saved in redshift table

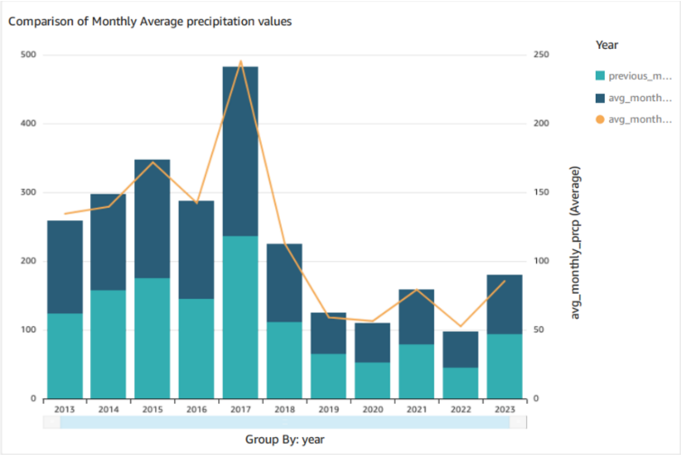


Fig. 16. Comparison of monthly precipitation values

Through this visualization we are showing the comparison of current and previous month averages of precipitation values for 10 years from 2013 to 2023.

2) **Count of stations based on different countries.**

CREATE TABLE noaadbproject.noaaschema.stations_count AS SELECT l.id AS location_id,sr.stationid, l.name AS location_name, lc.name AS location_category, COUNT (DISTINCT s.id) AS station_count FROM noaadbproject.noaaschema.stations s JOIN noaadbproject.noaaschema.stationrelations sr ON s.id = sr.stationid JOIN noaadbproject.noaaschema.locations l ON sr.locationid = l.id JOIN noaadbproject.noaaschema.locationcategories lc ON sr.locationcategoryid = lc.id GROUP BY l.id, l.name, lc.name, sr.stationid ORDER BY station_count DESC limit 60;

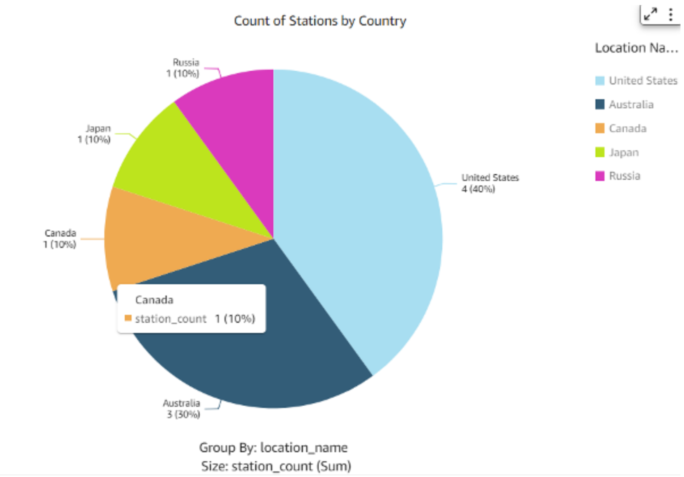


Fig. 17. Stations based on different countries

In this Pie chart we have taken Station count over various locations and used filter over some major countries like US, Australia, Canada, Japan, Russia. In the location names apart from countries we have regions, states many more as well. Based on our requirements we can explore.

3) **Average weather parameter by Year and Month.**

CREATE TABLE noaadbproject.noaaschema.climate AS SELECT d.value AS Value, d.datatypeid as climate_cond, s.name AS station_name, l.name AS location_name, lc.name AS location_category, s.latitude, s.longitude, dt.date, dt.quarter, dt.year, dt.month_name, dt.week, dt.dateid AS date_id, s.id AS station_id FROM noaadbproject.noaaschema.data d JOIN noaadbproject.noaaschema.stations s ON d.stationid = s.id JOIN noaadbproject.noaaschema.stationrelations sr ON s.id = sr.stationid JOIN noaadbproject.noaaschema.locations l ON sr.locationid = l.id JOIN noaadbproject.noaaschema.locationcategories lc ON sr.locationcategoryid = lc.id JOIN noaadbproject.noaaschema.dates dt ON d.dateid = dt.dateid WHERE d.value IS NOT NULL;

This visualization helps us to understand how the temperature changes year wise. As mentioned in heading it not only gives information about temperature but also all the parameters of weather like temperature max, temperature minimum, snowfall, snow depth.

4) **Fetching the weather type from 2010 to 2023 for selected stations.**

create table noaadbproject.noaaschema.weather_stations_2010_2023 as select A.datatypeid as weathertype, max(A.value) as value, S.name as StationName, D.date as date, D.year as year, D.month as month, A.dateid as date_id, S.id as Station_id from noaadbproject.noaaschema.data as A inner join noaadbproject.noaaschema.stations as S on A.stationid = S.id inner join noaadbproject.noaaschema.dates as D on A.dateid = D.dateid where A.dateid between ’20100101’ and ’20231130’ group by A.datatypeid, S.name, A.dateid, D.date, D.year, D.month, S.id having S.name like ’order by max(A.value) asc;

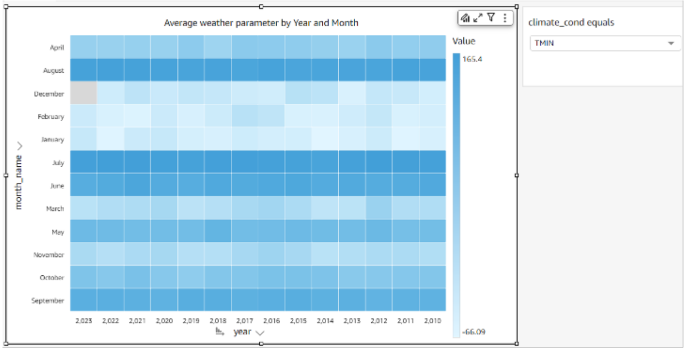


Fig. 18. Average weather parameter by Year and Month.

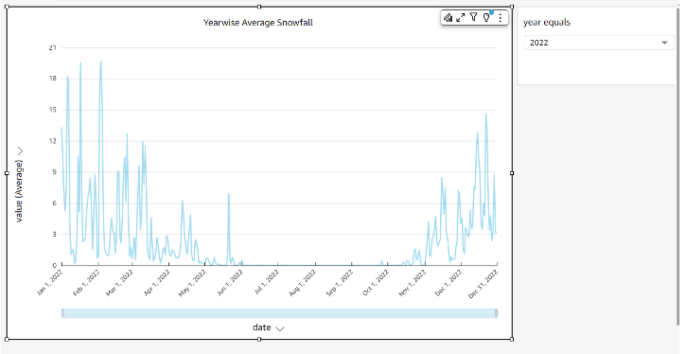


Fig. 19. Year-wise snow fall

5) **For a particular station location (Airports) and period (Dec 2022) checking weather types.**

create table noaadbproject.noaaschema.weather_custom_stations_dec22 as select A.datatypeid as weathertype, max(A.value) as value, S.name as StationName, D.date as date, D.year as year, D.month as month, A.dateid as date_id, S.id as Station_id from noaadbproject.noaaschema.data as A inner join noaadbproject.noaaschema.stations as S on A.stationid = S.id inner join noaadbproject.noaaschema.dates as D on A.dateid = D.dateid where A.dateid between ’20221201’ and ’20221231’ group by A.datatypeid,

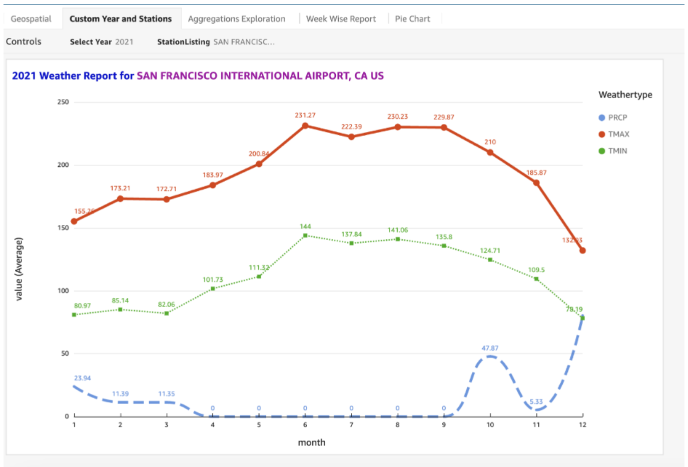


Fig. 20. weather type from 2010 to 2023 for selected stations

S.name, A.dateid, D.date, D.year, D.month, S.id having S.name like 'order by max(A.value) asc;

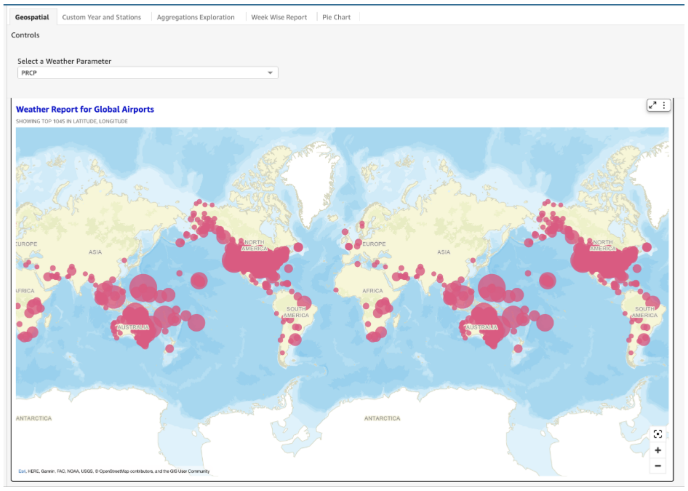

Geospatial
Custom Year and Stations
Aggregations Exploration
Week Wise Report
Pie Chart
Controls
Select a Weather Parameter
PRCP
Weather Report for Global Airports
SHOWING TOP 1045 IN LATITUDE, LONGITUDE
EUROPE
ASIA
NORTH AMERICA
AFRICA
SOUTH AMERICA
AUSTRALIA
ANTARCTICA
Esri, HERE, Garmin, FAO, NOAA, USGS, © OpenStreetMap contributors, and the GIS User Community

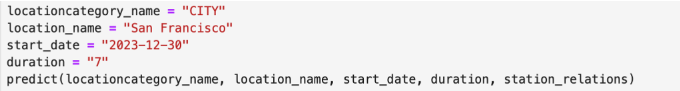
Fig. 21. weather type for selected station and year

IX. Recommendations/Modeling

Prediction Model: We have used the XGBoost Algorithm to train 5 different regression models to predict maximum and minimum temperature, precipitation, snow fall and snow depth. We used this model as this is known to work best for tabular data. R2 score for our model is 0.398.

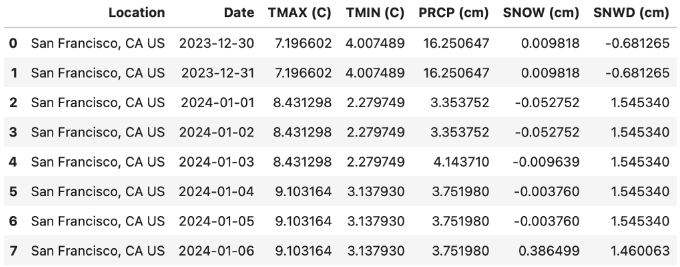
Modeling Predictions output:

1) San Francisco



```
locationcategory_name = "CITY"
location_name = "San Francisco"
start_date = "2023-12-30"
duration = "7"
predict(locationcategory_name, location_name, start_date, duration, station_relations)
```

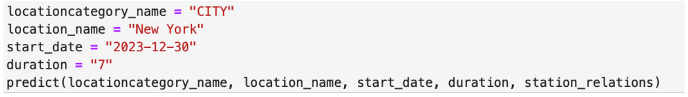
Fig. 22. Query for prediction of weather for San Francisco



	Location	Date	TMAX (C)	TMIN (C)	PRCP (cm)	SNOW (cm)	SNWD (cm)
0	San Francisco, CA US	2023-12-30	7.196602	4.007489	16.250647	0.009818	-0.681265
1	San Francisco, CA US	2023-12-31	7.196602	4.007489	16.250647	0.009818	-0.681265
2	San Francisco, CA US	2024-01-01	8.431298	2.279749	3.353752	-0.052752	1.545340
3	San Francisco, CA US	2024-01-02	8.431298	2.279749	3.353752	-0.052752	1.545340
4	San Francisco, CA US	2024-01-03	8.431298	2.279749	4.143710	-0.009639	1.545340
5	San Francisco, CA US	2024-01-04	9.103164	3.137930	3.751980	-0.003760	1.545340
6	San Francisco, CA US	2024-01-05	9.103164	3.137930	3.751980	-0.003760	1.545340
7	San Francisco, CA US	2024-01-06	9.103164	3.137930	3.751980	0.386499	1.460063

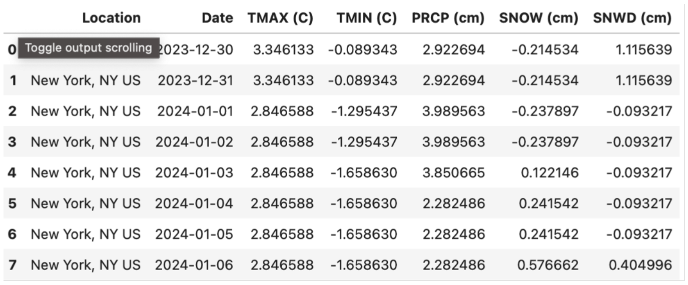
Fig. 23. Forecasting of San Francisco weather for next 7 days

2) New York



```
locationcategory_name = "CITY"
location_name = "New York"
start_date = "2023-12-30"
duration = "7"
predict(locationcategory_name, location_name, start_date, duration, station_relations)
```

Fig. 24. Query for prediction of weather for New York



	Location	Date	TMAX (C)	TMIN (C)	PRCP (cm)	SNOW (cm)	SNWD (cm)
0	Toggle output scrolling	2023-12-30	3.346133	-0.089343	2.922694	-0.214534	1.115639
1	New York, NY US	2023-12-31	3.346133	-0.089343	2.922694	-0.214534	1.115639
2	New York, NY US	2024-01-01	2.846588	-1.295437	3.989563	-0.237897	-0.093217
3	New York, NY US	2024-01-02	2.846588	-1.295437	3.989563	-0.237897	-0.093217
4	New York, NY US	2024-01-03	2.846588	-1.658630	3.850665	0.122146	-0.093217
5	New York, NY US	2024-01-04	2.846588	-1.658630	2.282486	0.241542	-0.093217
6	New York, NY US	2024-01-05	2.846588	-1.658630	2.282486	0.241542	-0.093217
7	New York, NY US	2024-01-06	2.846588	-1.658630	2.282486	0.576662	0.404996

Fig. 25. Forecasting of New York weather for next 7 days

For analysis and recommendations we tried to convert the dataset into a format with which we can do predictive analysis. Post checking the R2 score we have created a corelation matrix. We can see a white patch here. Those fields are highly positively correlated. So we could have selected only 1 field out of them, like day_of_year, and dropped quarter, month, and week. Tmax and Tmin are positively correlated. Latitude and Longitude are negatively correlated. These correlated fields are what is causing the low accuracy.

X. GitHub Repository

https://github.com/NehaBais/NOAA-aws-data-pipeline.git _NoSQL

XI. CONCLUSION

In conclusion, our ETL pipeline, orchestrated seamlessly on AWS, exemplifies a scalable and efficient approach to data management for enterprises. By leveraging AWS Data Pipeline, Glue jobs, and Redshift, we have established a reliable flow from diverse data sources to a centralized Data

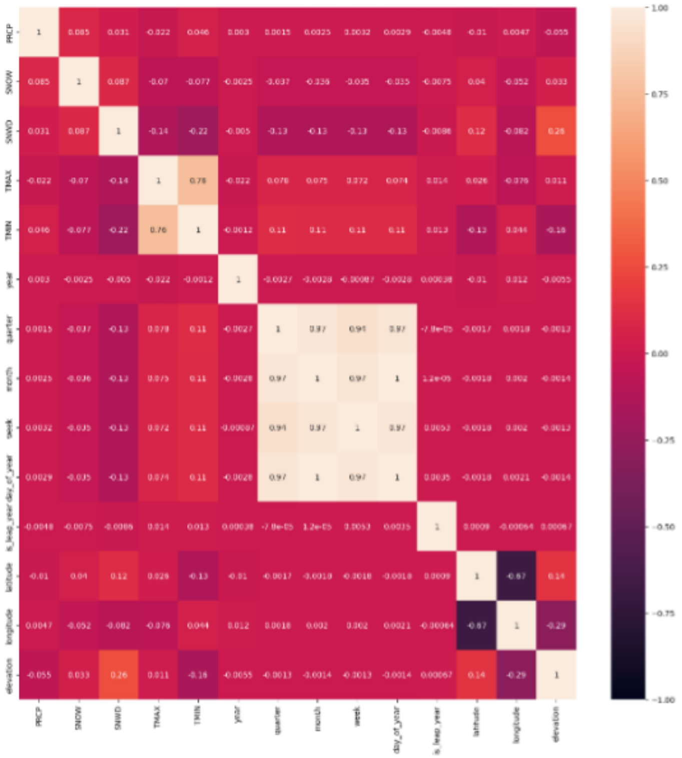


Fig. 26. Correlation analysis of the features

Warehouse. This not only enhances operational efficiency but also unlocks the potential for data-driven decision-making. The integration of visualization through Amazon Quick sight, coupled with model training and prediction, adds a layer of sophistication to our analytics capabilities. This end-to-end process not only caters to current business needs but also ensures adaptability to evolving data landscapes.

In summary, our ETL pipeline is a versatile solution that empowers enterprises across industries. It facilitates streamlined data processing, enhances decision-making, and positions businesses to stay agile in the face of evolving analytics requirements.