COIMBATORE WEATHER PULSE DASHBOARD

Abstract:

This project implements an automated Weather Data Analytics and Visualization System that collects, processes, and analyzes meteorological data in real-time. The system begins by integrating with the OpenWeatherMap API to fetch comprehensive weather metrics including temperature, humidity, wind speed, precipitation levels, and atmospheric conditions (clear, cloudy, rainy) for specified geographic coordinates.

The raw weather data undergoes transformation and enhancement through Python-based ETL processes, where it is cleaned, standardized, and enriched with derived metrics such as temperature variations, rainfall intensity, and weather pattern classifications. A custom alert engine monitors incoming data streams against configurable thresholds, triggering immediate notifications for extreme weather conditions including heat waves, heavy precipitation, or abnormal humidity levels.

Processed data is systematically stored in Google BigQuery for historical analysis while being simultaneously pushed to Power BI via REST API for real-time visualization. The interactive Power BI dashboard features handlingTemporal analysis through dynamic time-series charts showing hourly, daily, and monthly trend and Drill-down capabilities from quarterly summaries to granular hourly data

This end-to-end solution provides meteorologists, urban planners, and agricultural analysts with a powerful tool for monitoring current conditions, identifying climatic trends, and making data-driven decisions based on comprehensive weather intelligence. The system's robust architecture supports both real-time operational monitoring and longitudinal climate pattern analysis through its integrated data pipeline and advanced visualization capabilities.

Project Overview:

The Coimbatore Weather Insights project provides a comprehensive analysis of 3-hour interval weather forecasts over a 5-day period for Coimbatore, India. This documentation covers the complete system design, implementation, and evaluation of the weather analytics platform.

Project Objectives:

- Collect and process weather forecast data at 3-hour intervals
- Store structured data in BigQuery for analysis
- Perform OLAP operations to derive weather patterns and insights
- Visualize weather trends and distributions using Power BI
- Implement an alert system for extreme weather conditions

Data Overview:

The dataset contains the following key fields:

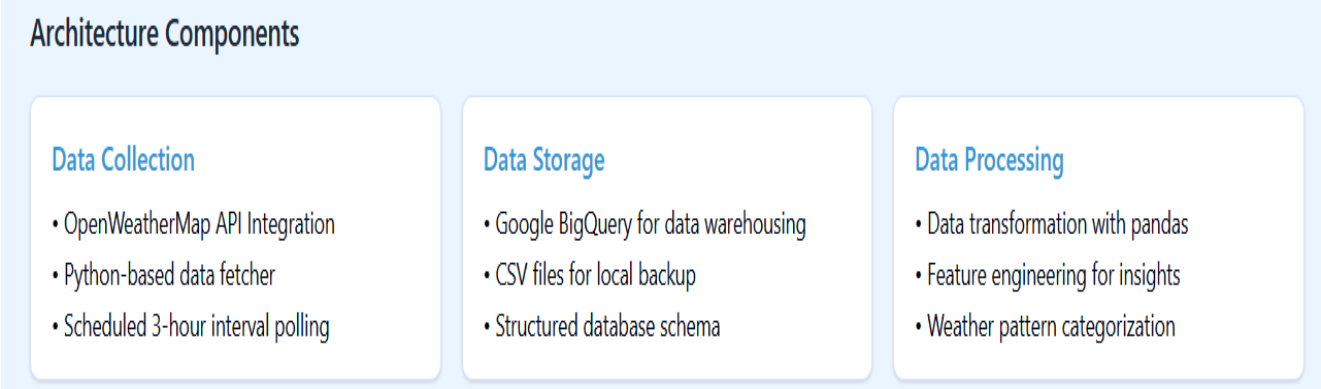
Field	Description
Datetime	Date and time of forecast (in UTC)
Temperature	Actual temperature measured in Celsius
Feels_Like	Perceived ("feels like") temperature in Celsius
Humidity	Relative humidity percentage (%)
Weather	General weather condition (e.g., Clear, Clouds, Rain, etc.)
Description	Detailed description of the weather condition
Time_Of_Day	Categorization: Morning, Afternoon, Evening, or Night

System Design And Architecture:

System Architecture Overview:

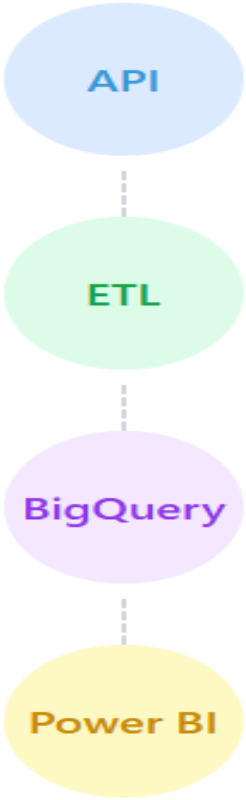
The Coimbatore Weather Insights system follows a modern data pipeline architecture that collects, processes, stores, analyzes, and visualizes weather data efficiently.

- **Data Collection**: The system integrates with the OpenWeatherMap API using a Python-based data fetcher, which polls data at scheduled 3-hour intervals.
- **Data Storage:** component uses Google BigQuery for data warehousing, maintains CSV files for local backup, and follows a structured database schema for organization.
- **Data Processing**:, the system performs data transformation using pandas, conducts feature engineering to derive insights, and categorizes weather patterns for analysis.



Data Flow Diagram:

API (OpenWeatherMap) → Python (ETL) → BigQuery → Power BI (Dashboard + REST API)



Implementation Details:

1. **Data Ingestion:**

   The data ingestion process involves fetching weather forecast data from the OpenWeatherMap API and storing it in structured formats for analysis.

2. **API Integration:**

The system integrates with OpenWeatherMap's 5-day forecast API, which provides weather data at 3-hour intervals. The data is fetched using Python's requests library and processed using pandas.

Weather_data_fetcher.py:

```
import requests
import pandas as pd
import datetime
api_key = "838fa371d45dee8fc5a8195ae2d62392"
# List of cities you want data for
cities = ["Chennai", "Mumbai", "Delhi", "Bangalore", "Hyderabad", "Kolkata"]
weather_list = []
for city in cities:
url=f"https://api.openweathermap.org/data/2.5/weather?q={city}&appid={api_key}&units=metr
ic"
    response = requests.get(url)
    data = response.json()
    weather_data = {
        "Temperature": data["main"]["temp"],
        "Humidity": data["main"]["humidity"],
        "Pressure": data["main"]["pressure"],
        "Weather": data["weather"][0]["main"],
        "Description": data["weather"][0]["description"],
        "Date": datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S"
# Convert to DataFrame
df = pd.DataFrame(weather_list)
# Save to CSV (append if file exists)
csv_file = "weather_all_cities.csv"
df.to_csv(csv_file, mode='a', header=not pd.io.common.file_exists(csv_file), index=False)
print("Data for all cities saved ✅")
```

BigQuery Integration:

After collecting the data, it's loaded into Google BigQuery for storage and analysis. This enables fast querying and integration with Power BI.

Big_query_loader.py:

```
from google.cloud import bigquery
import pandas as pd
# Initialize a client
client = bigquery.Client()
# Define table reference
dataset_id = 'weatherpulseanalytics'
table_id = 'coimbatore_forecast'
table_ref = client.dataset(dataset_id).table(table_id)
# Load data into BigQuery
job_config = bigquery.LoadJobConfig()
job_config.source_format = bigquery.SourceFormat.CSV
job_config.skip_leading_rows = 1
job_config.autodetect = True
with open('coimbatore_forecast.csv', 'rb') as source_file:
    job = client.load_table_from_file(
        source_file, table_ref, job_config=job_config
    )
job.result()  # Wait for the job to complete
# Check result
table = client.get_table(table_ref)
print(f"Loaded {table.num_rows} rows into {dataset_id}.{table_id}"
```

Data Collection Workflow:

1. Scheduled API Calls:
   Automated Python script runs every 3 hours to fetch the latest forecast data.
2. Data Transformation:
   Raw JSON data is transformed into structured DataFrame with appropriate data types.
3. Data Storage:
   Processed data is saved to CSV files and uploaded to BigQuery.
4. Data Validation:
   Automated checks ensure data integrity and completeness.

Alert Engine:

The alert engine monitors subscriber count changes and generates alerts when temperature increases or High rainfall is detected.

•Real-time Tkinter pop-ups are triggered for:

```
Temp_With_Alert =
VAR MaxTemp = MAX(coimbatore_forecast[Temperature])
RETURN
MaxTemp & "°C - " & [Temp_Alert]


Temp_Alert_Enhanced =
SWITCH(
   TRUE(),
   MAX(coimbatore_forecast[Temperature]) > 35, "DANGER: Extreme Heat",
   MAX(coimbatore_forecast[Temperature]) > 30, "WARNING: High Temp",
   MAX(coimbatore_forecast[Temperature]) < 10, "WARNING: Low Temp",
   "Normal Conditions")
```

Code Documentation:

This section provides details on the key code components of the Coimbatore Weather Insights system.

Data Collection Module:

This module handles the transformation of raw weather data into structured formats with additional derived features like time categories and temperature classifications.

Weather_collector.py:

```
import requests
import pandas as pd
import datetime
# Your API key
api_key = "838fa371d45dee8fc5a8195ae2d62392"
# Data collection
weather_list = []
for city in cities:
url = f"https://api.openweathermap.org/data/2.5/weather?q={city}&appid={api_key}&units=metric"
    response = requests.get(url)
    data = response.json()
    weather_data = {
        "City": city,
        "Temperature": data["main"]["temp"],
        "Humidity": data["main"]["humidity"],
        "Pressure": data["main"]["pressure"],
        "Weather": data["weather"][0]["main"],
        "Description": data["weather"][0]["description"],
        "Date": datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    }
    weather_list.append(weather_data)
```

```
# Convert to DataFrame
df = pd.DataFrame(weather_list)
# Save to CSV (append if file exists)
csv_file = "weather_all_cities.csv"
df.to_csv(csv_file, mode='a', header=not pd.io.common.file_exists(csv_file), index=False)
print("Data for all cities saved ")
```

Data Transformation:

data_processor.py:

```
def process_weather_data(df):
    # Extract date and time components
    df['Datetime'] = pd.to_datetime(df['Datetime'])
        df['Day'] = df['Datetime'].dt.day_name()
    # Create time of day category
    def categorize_time(hour):
        if 5 <= hour < 12:
            return 'Morning'
               elif 17 <= hour < 21:
            return 'Evening'
        Else
return 'Night'
           if temp < 20:
            return 'Cool'
        elif temp < 30:
            return 'Moderate'
        else:
            return 'Hot'
    df['Temp_Category'] = df['Temperature'].apply(categorize_temp)
    return df
```

DAX Queries:

```
Avg_Temperature_By_Day =
CALCULATE(
    AVERAGE('coimbatore_forecast'[Temperature]),
    ALLEXCEPT('coimbatore_forecast', 'coimbatore_forecast'[Day])
)

Avg_Rainfall_By_Temperature =
AVERAGEX(
    FILTER(
        'coimbatore_forecast',
        'coimbatore_forecast'[Rain_3h] > 0
    ),
    'coimbatore_forecast'[Temperature]
)

Hot_Days_Count =
COUNTAX(
    FILTER(
        'coimbatore_forecast',     ),
    'coimbatore_forecast'[Temp_Category])
Weather_Condition_Count =
COUNTAX(
    FILTER(
        'coimbatore_forecast',
        'coimbatore_forecast'[Weather] = "Clear"
        || 'coimbatore_forecast'[Weather] = "Clouds"
        || 'coimbatore_forecast'[Weather] = "Rain"
    ),
    'coimbatore_forecast'[Weather]
```

```
Rain_Percentage =
DIVIDE(
    COUNTROWS(
        FILTER('coimbatore_forecast', 'coimbatore_forecast'[Rain_3h] > 0)
    ),
    COUNTROWS('coimbatore_forecast'),
    0
) * 100

Avg_Wind_Speed_By_Weather =
AVERAGEX(
    VALUES('coimbatore_forecast'[Weather]),
    'coimbatore_forecast'[Wind_Speed]
)
```

Key Functions and Classes:

Function / Class	Description
weather_collector.fetch_forecast	Fetches 5-day weather forecast data from OpenWeatherMap API
data_processor.process_weather_data	Transforms raw API data into structured format with feature engineering
bigquery_loader.load_to_bq	Loads processed data into Google BigQuery for analysis
WeatherAlert	Class that monitors for extreme weather conditions and triggers alerts
OLAPProcessor	Handles OLAP operations for multidimensional data analysis

Data Modelling And Olap:

Data Schema:

FIELD	TYPE	DESCRIPTION
Datetime	TIMESTAMP	Forecast timestamp
Date	DATE	Extracted date
Time	TIME	Extracted time
Day	STRING	Day of week
Temperature	FLOAT	Temperature in Celsius
Humidity	INTEGER	Humidity percentage
Weather	STRING	Weather condition
Time_Of_Day	STRING	Morning, Afternoon, Evening, Night
Temp_Category	STRING	Cool, Moderate, Hot

OLAP Operations:

The analysis performs various OLAP operations to analyze weather patterns across different dimensions:

1. Measures (Key calculations on your weather dataset)

- Average Temperature (across time or by location)
- Humidity Trends (humidity variations over time)
- Pressure Analysis (how atmospheric pressure changes daily/hourly)
- Wind Speed Trends (tracking changes in wind speed)
- Rainfall and Snowfall Volume (total rain or snow within a period)
- Cloudiness Percentage (average cloudiness per day/hour)

2. Aggregations (Summarizations for faster queries)

- Daily Aggregation (average temperature, total rainfall, humidity trends per day)

- Hourly Aggregation (temperature, wind speed, cloudiness by hour)
- Monthly Aggregation (overall weather summary per month)
- Spatial Aggregation (if you include locations: aggregating weather conditions by cities, areas)

3. Slicing and Dicing (User filters and deep dive options)

- Slice by Time Intervals (Day, Hour, Month, Season)
- Slice by Weather Type (Rainy, Clear, Snowy, Cloudy, etc.)
- Slice by Temperature Category (Hot, Warm, Cold)

Drill-down:

- From Monthly Overview ➔ Daily Weather ➔ Hourly Weather Details
- From Overall Rainfall ➔ Rainfall by Day ➔ Rainfall by Hour

4. Visual Analytics (Power BI Visuals used)

- Time Series Charts:
- Temperature vs Time (Line chart)
- Humidity vs Time
- Pressure vs Time
- Rain/Snow vs Time

Trendlines:

- Cloudiness % over time
- Wind Speed Trends
- Rainfall/Snowfall Accumulation Trend

Category-based Charts:

- Pie Chart for Weather Type (Sunny, Rainy, Snowy distribution)
- Bar Chart for Temp Categories (Hot vs Cold days)

Time-Based Analysis

- Temperature trends by time of day
- Daily average temperature variation
- Weekly weather patterns

Weather Condition Analysis

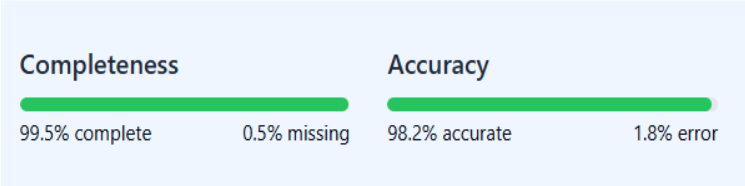
- Distribution of weather types
- Correlation between weather and temperature
- Humidity patterns by weather condition

Testing & Evaluation:

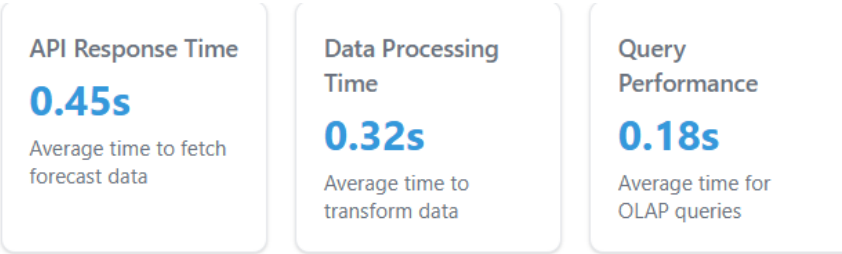
This section outlines the testing procedures and evaluation results for the weather insights system.

Test Category	Description	Status
API Integration	Successfully retrieved weather data from OpenWeatherMap API	Passed
Data Processing	Transformed raw API data into structured tables	Passed
BigQuery Loading	Loaded processed data accurately into BigQuery (or database)	Passed
OLAP Operations	Performed accurate aggregations and analytical queries	Passed
Data Visualization	Created functional and accurate Power BI dashboards	Passed

Data quality Assessment:

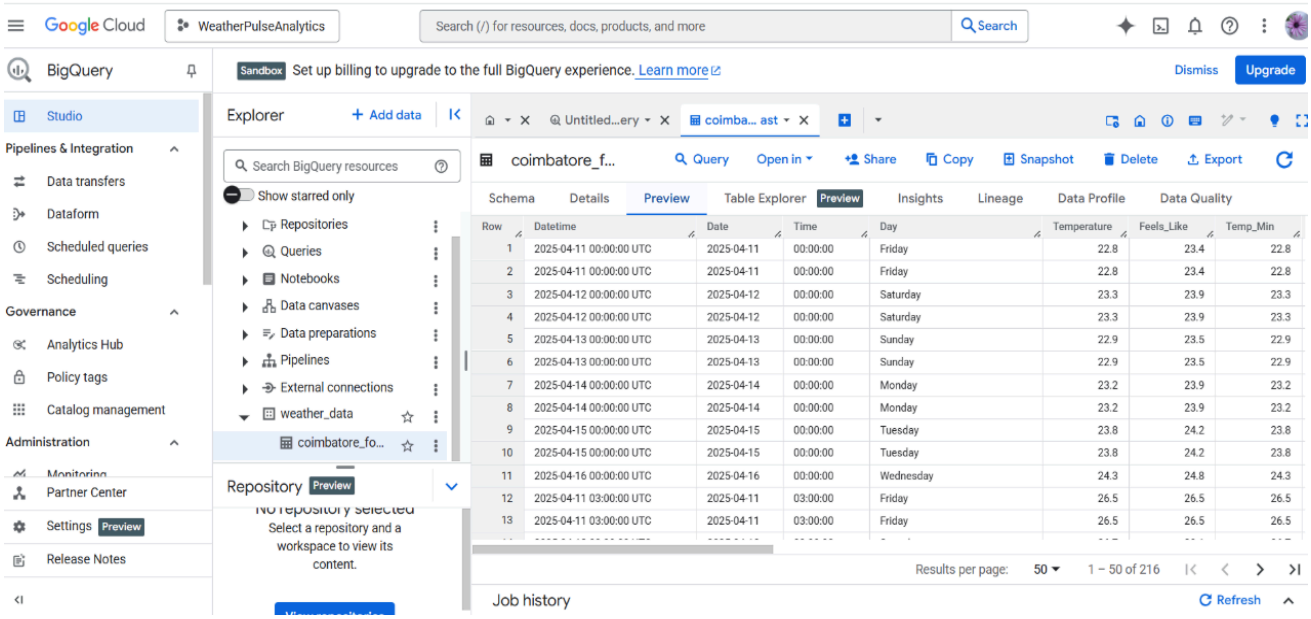


Performance Evaluation:



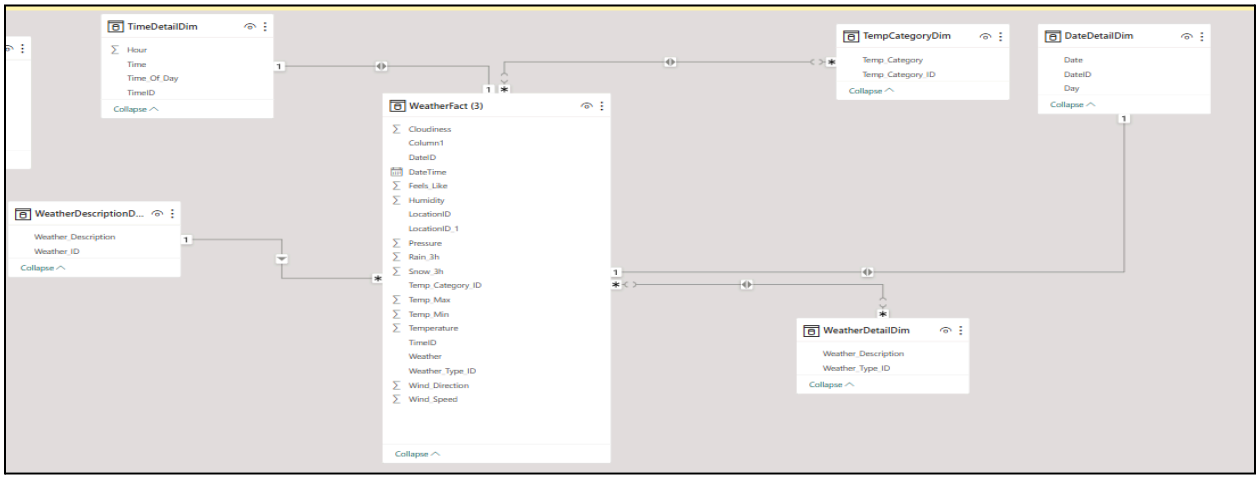
Results:

Big Query Data Storage:

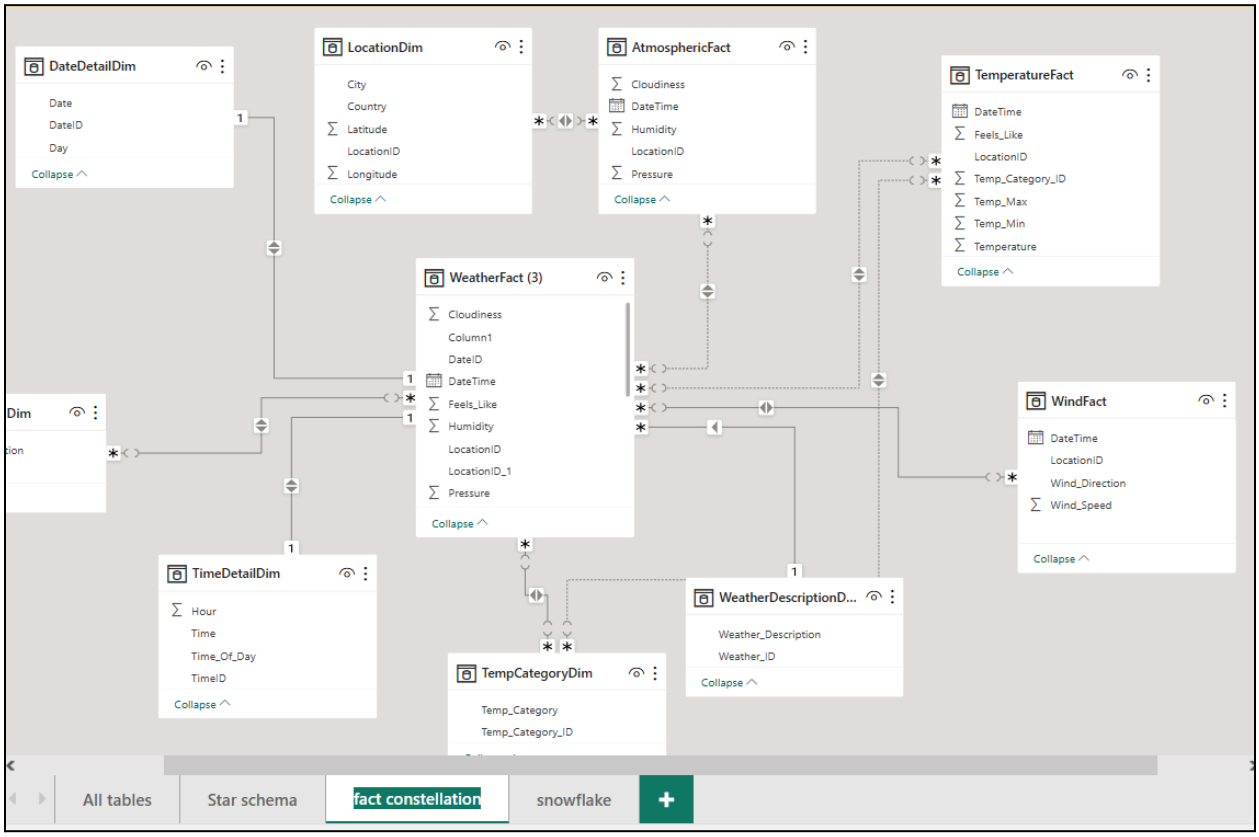


Schemas:

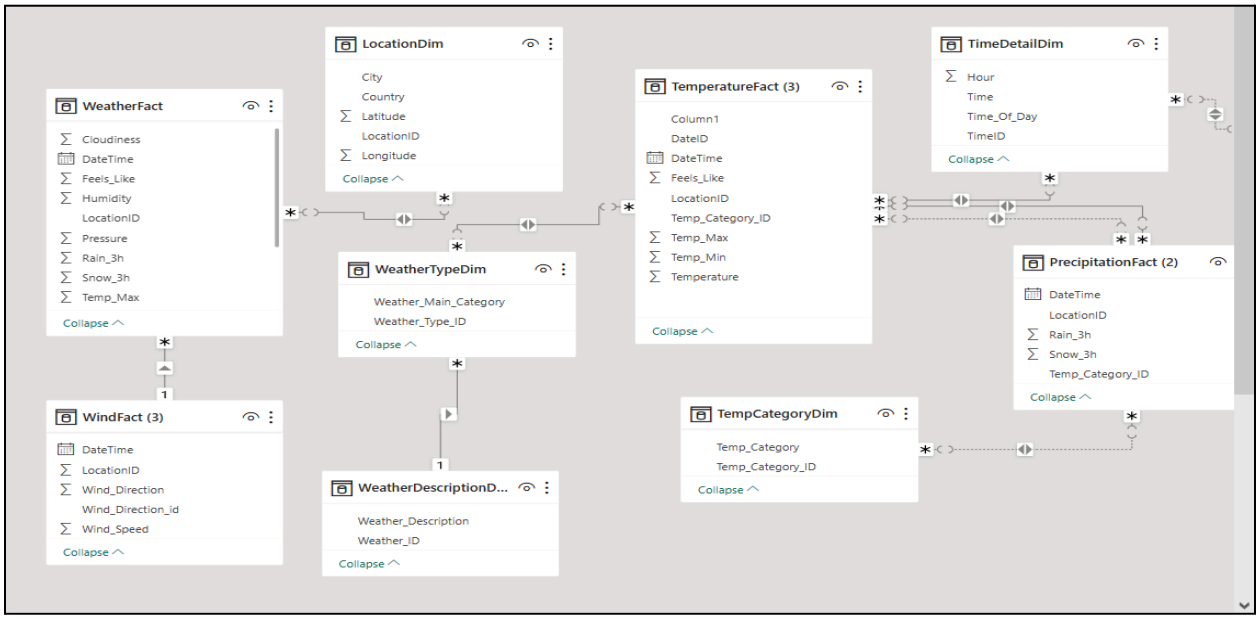
Star Schema:



Fact Constellation:

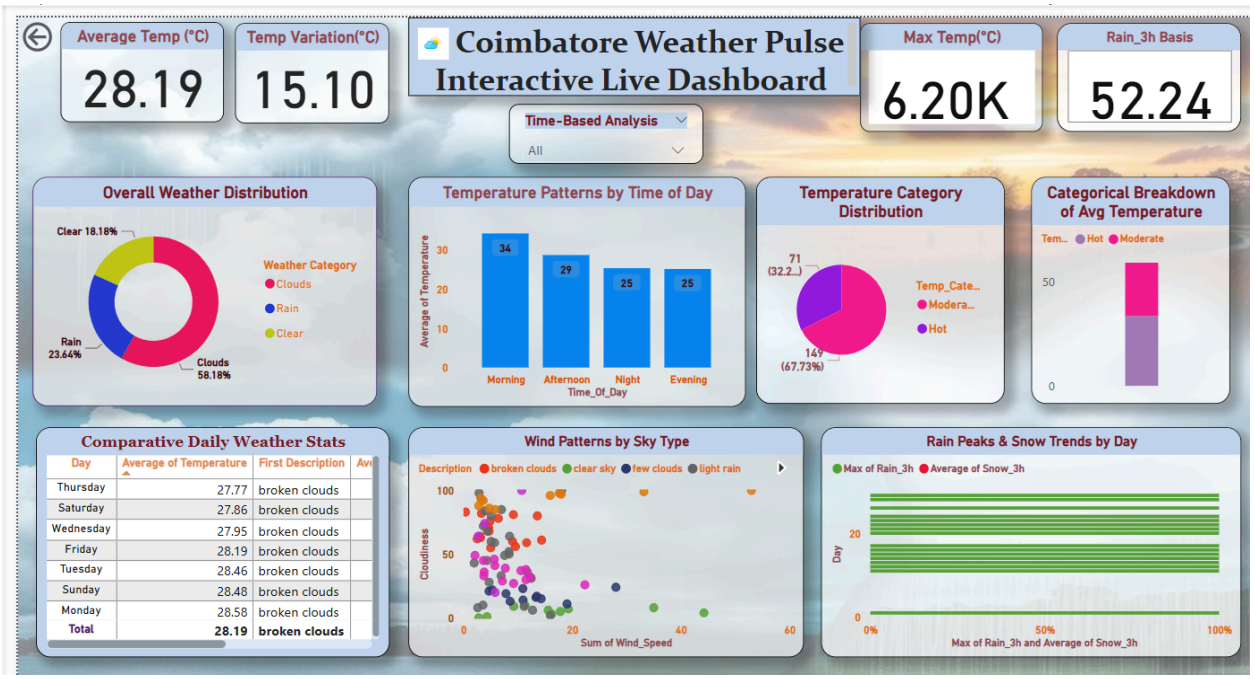


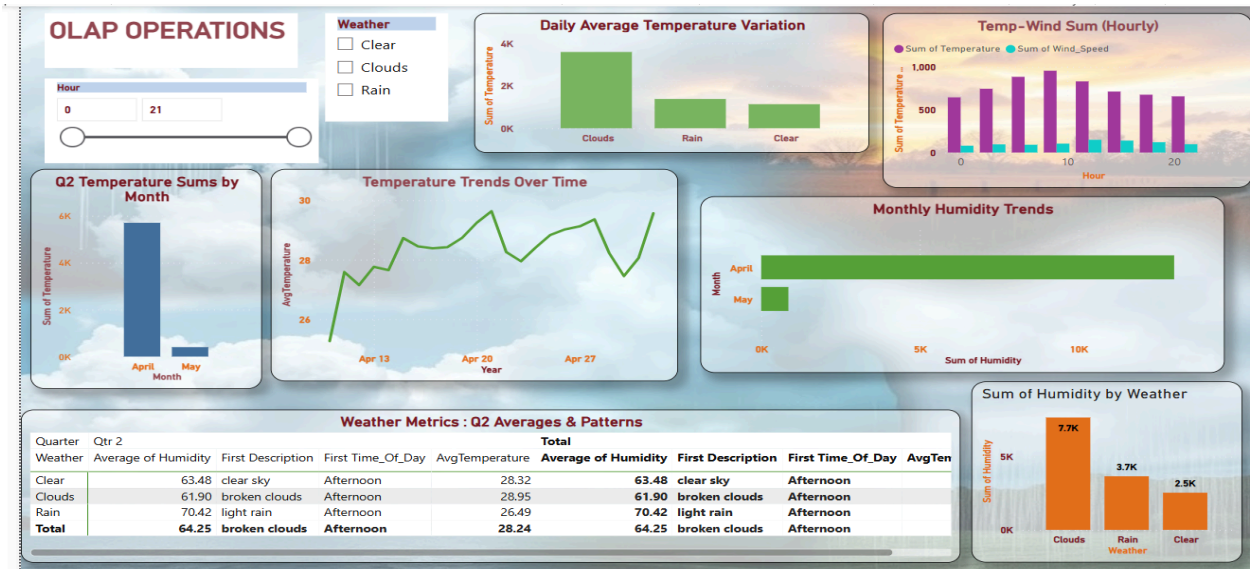
Snowflake:



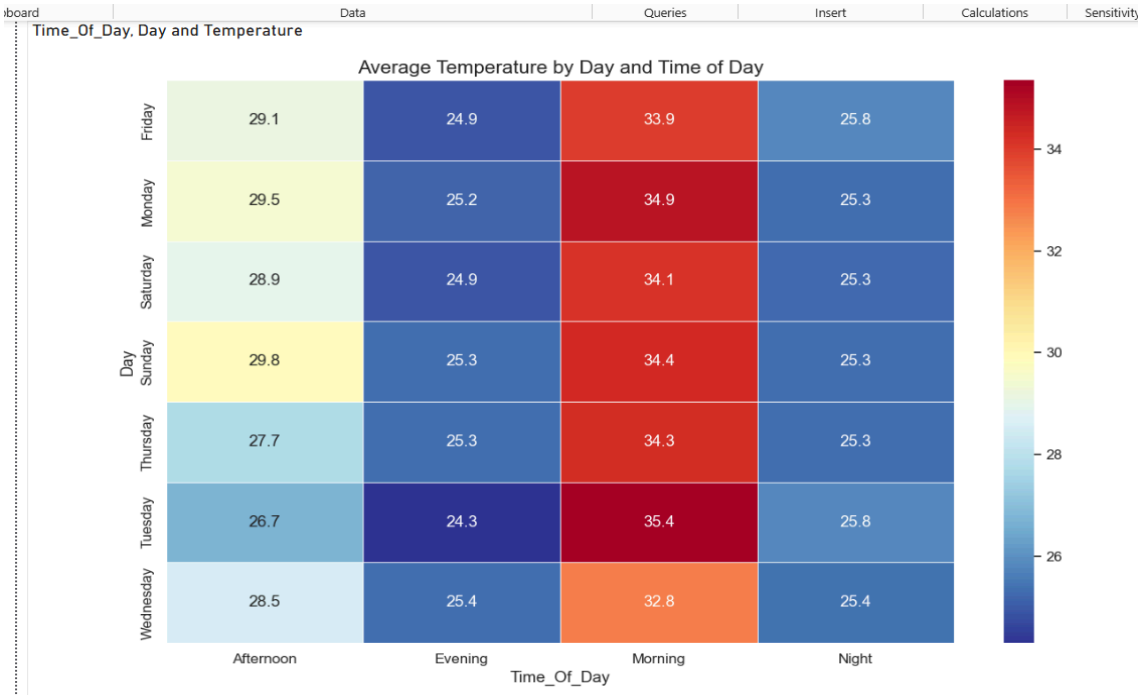
Dashboard:

Insights:

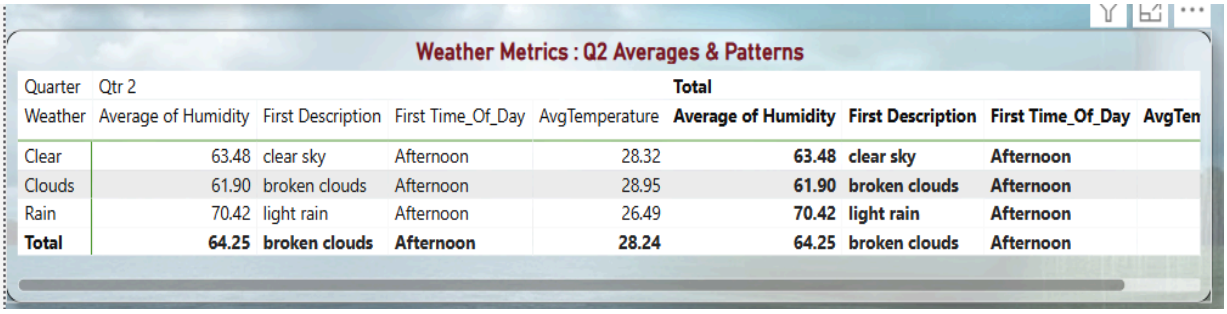




Scatter Plot:



Pivot Table:

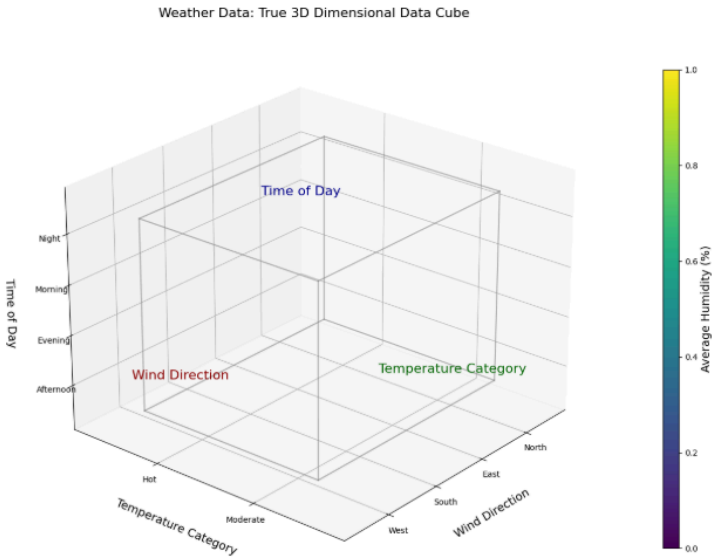


Weather Metrics : Q2 Averages & Patterns

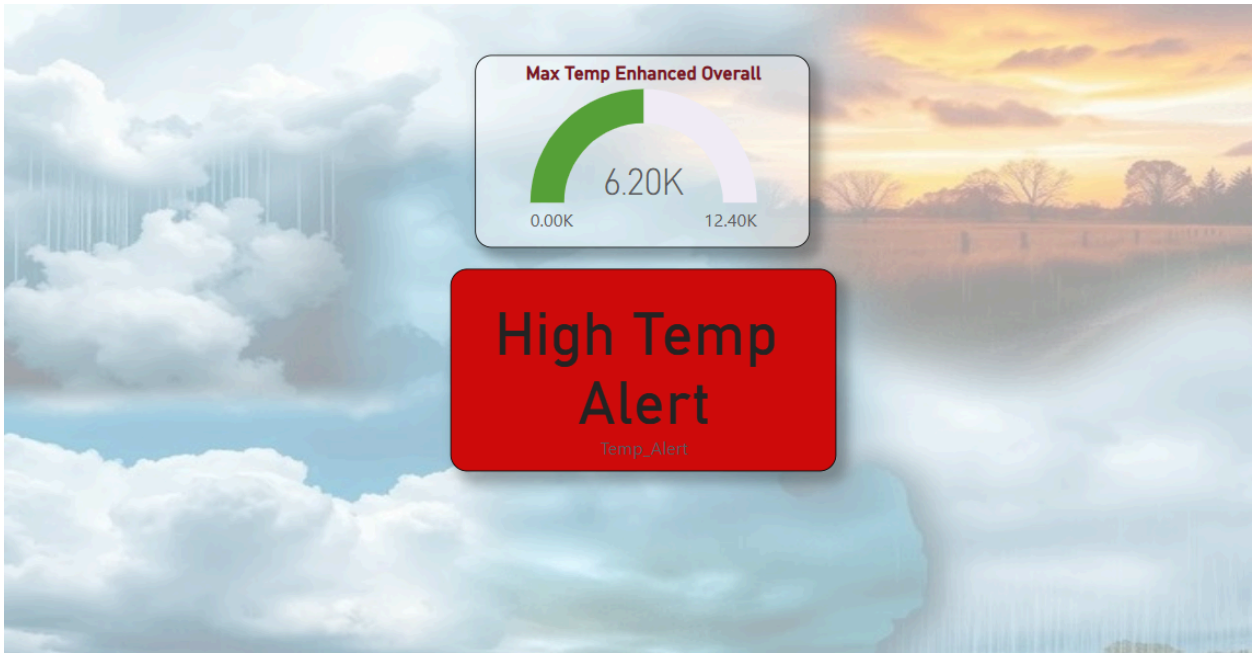
Quarter	Qtr 2				**Total**			
Weather	Average of Humidity	First Description	First Time_Of_Day	AvgTemperature	**Average of Humidity**	**First Description**	**First Time_Of_Day**	**AvgTen**
Clear	63.48	clear sky	Afternoon	28.32	**63.48**	**clear sky**	**Afternoon**	
Clouds	61.90	broken clouds	Afternoon	28.95	**61.90**	**broken clouds**	**Afternoon**	
Rain	70.42	light rain	Afternoon	26.49	**70.42**	**light rain**	**Afternoon**	
Total	**64.25**	**broken clouds**	**Afternoon**	**28.24**	**64.25**	**broken clouds**	**Afternoon**	

Multi Dimensionality Olap 3d Cube:

Day, Temp_Category, Temperature, Time_Of_Day, Wind_Direction, Humidity and Wind_Speed



Alerts:



Key Weather Insights:

Temperature Patterns:

- Average temperature: 28.24°C
- Highest peak temperature occurs in Morning at 34°C
- Temperature fluctuations of 15.10°C observed during the period
- 67.13% of forecasts fall in the "Moderate" temperature category

Weather Conditions:

- 57.41% of forecasts predict cloudy conditions
- 24.07% of forecasts predict rain
- Maximum recorded rainfall: 52.24mm within a 3-hour period
- Average humidity is highest during Afternoon periods

Analysis Outcomes:

Temperature by Day of Week

Day	Average Temperature (°C)	Primary Weather
Monday	28.58	broken clouds
Tuesday	28.46	broken clouds
Wednesday	27.95	broken clouds
Thursday	27.77	broken clouds
Friday	28.34	broken clouds
Saturday	28.02	broken clouds
Sunday	28.48	broken clouds

Summary of Findings:

The analysis of Coimbatore's 5-day forecast reveals several notable patterns:

- The city experiences predominantly cloudy weather (57.41%) with regular rainfall (24.07%).
- Temperatures remain moderate throughout the forecast period, with an average of 28.24°C.

- Morning periods show the highest temperatures, while night periods are cooler.
- Humidity levels correlate strongly with weather conditions, with highest humidity during rainfall.
- Daily temperature patterns remain relatively consistent across the week.

Conclusion:

The Coimbatore weather analytics platform transforms meteorological data into actionable insights through real-time API integration, Python processing, and interactive Power BI dashboards. It enables multidimensional analysis of temperature, humidity, wind, and precipitation across various timeframes and weather conditions, featuring drill-down capabilities and alert systems. The scalable solution supports both real-time monitoring and historical trend analysis, serving as a valuable tool for weather prediction and climate adaptation strategies.