**RV UNIVERSITY**

**School of Computer Science and Engineering**

**Bengaluru – 560059**

****

**FUNDAMENTALS OF DATA ENGINEERING**

**COURSE CODE: CS3238**

**V Semester B.Tech (HONS.)**

**Project Report**

|  |  |
| --- | --- |
| **Name** | **Jeevan EG** |
| **USN** | **1RVU22CSE069** |
| **Academic Year** | **2024 - 2025** |
| **Project Title** | **Real-Time Weather Data Pipeline with DynamoDB, Snowflake, and AWS Lambda** |

**LIST OF CONTENTS**

1. **Introduction**
2. **Project Profile**

a. **Objectives**

b. **Dataset**

c. **Methodology**

1. **Observations and analysis**
2. **Results**
3. **Conclusion**

**ABSTRACT**

This project implements a real-time weather data pipeline utilizing AWS services and Snowflake for efficient data collection, storage, and analysis. The system automatically fetches weather data for multiple Indian cities every hour, processes it through AWS Lambda functions, stores it in DynamoDB, and ultimately loads it into Snowflake for analytical purposes. The pipeline demonstrates modern data engineering practices, incorporating serverless computing, stream processing, and cloud data warehousing.

**INTRODUCTION**

In the era of big data and real-time analytics, efficient data pipelines are crucial for processing and analysing weather data. This project presents an automated solution for collecting and processing weather data using cloud services. The implementation leverages AWS Lambda for serverless computing, DynamoDB for NoSQL storage, AWS S3 for data lake storage, and Snowflake for data warehousing.

**PROJECT PROFILE**

**a. Objectives**

* Implement an automated system for collecting real-time weather data
* Create a serverless architecture using AWS Lambda
* Establish a reliable data storage solution using DynamoDB
* Develop a data processing pipeline using DynamoDB Streams
* Set up an analytical database in Snowflake for data analysis
* Automate the entire data flow from collection to analysis

**b. Dataset**

The project utilizes real-time weather data from the WeatherAPI.com API, collecting the following parameters for ten Indian cities:

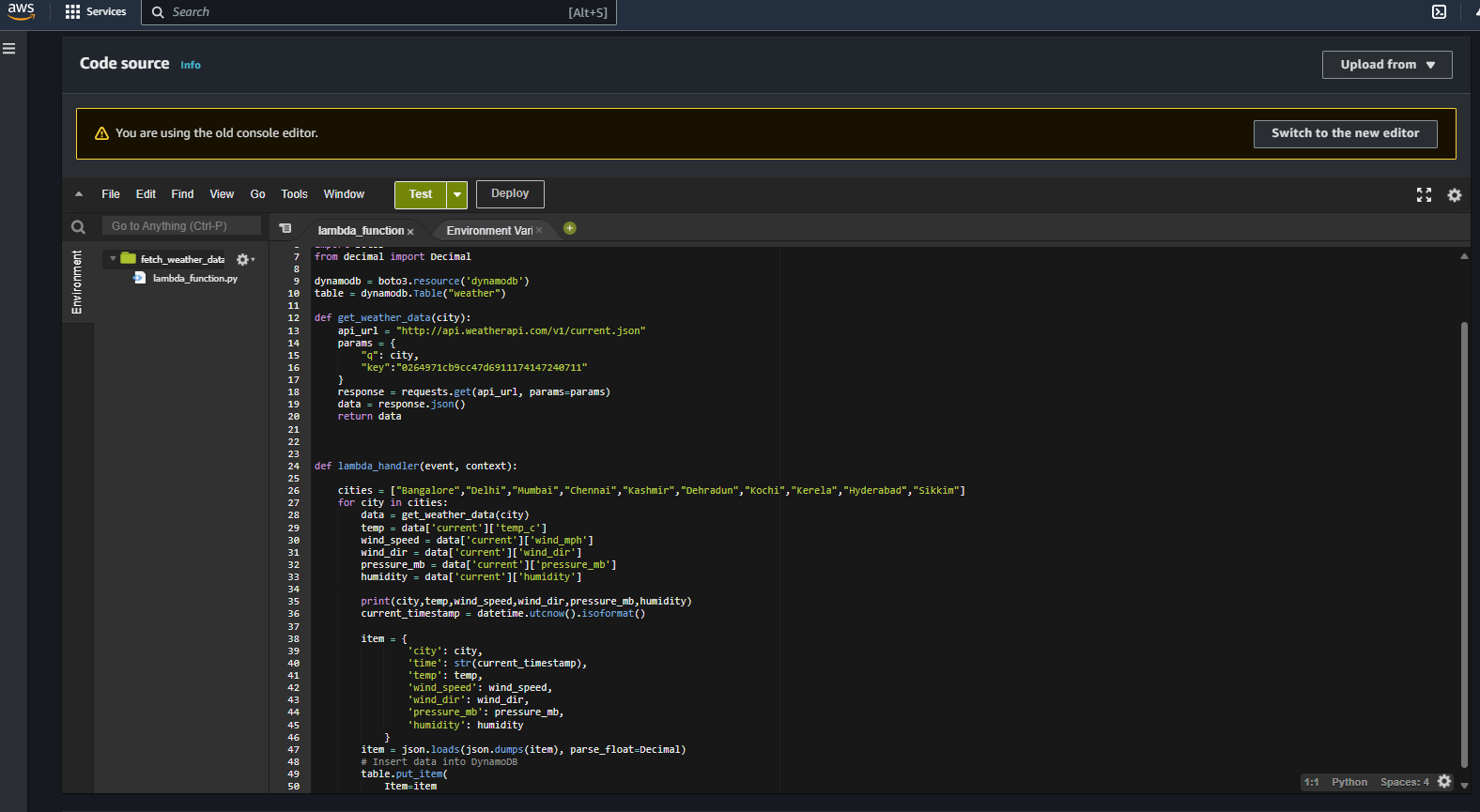
* Temperature (°C)
* Wind Speed (mph)
* Wind Direction
* Pressure (mb)
* Humidity (%)

Cities monitored: Bangalore, Delhi, Mumbai, Chennai, Kashmir, Dehradun, Kochi, Kerala, Hyderabad, and Sikkim.

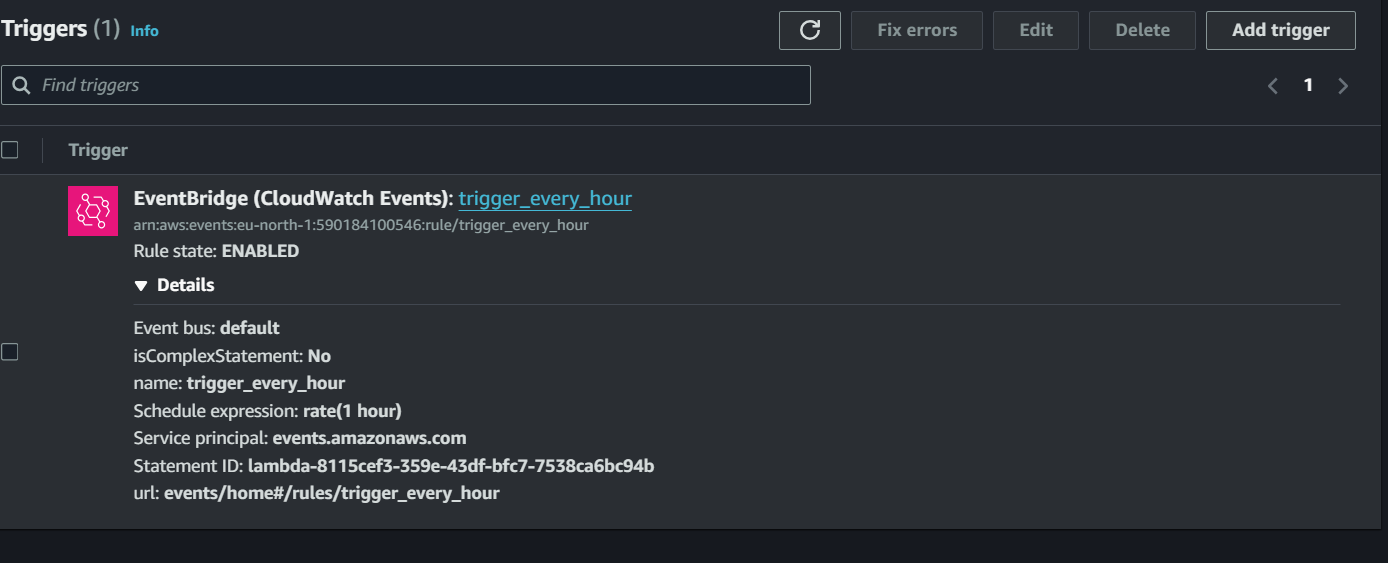
**c. Methodology**

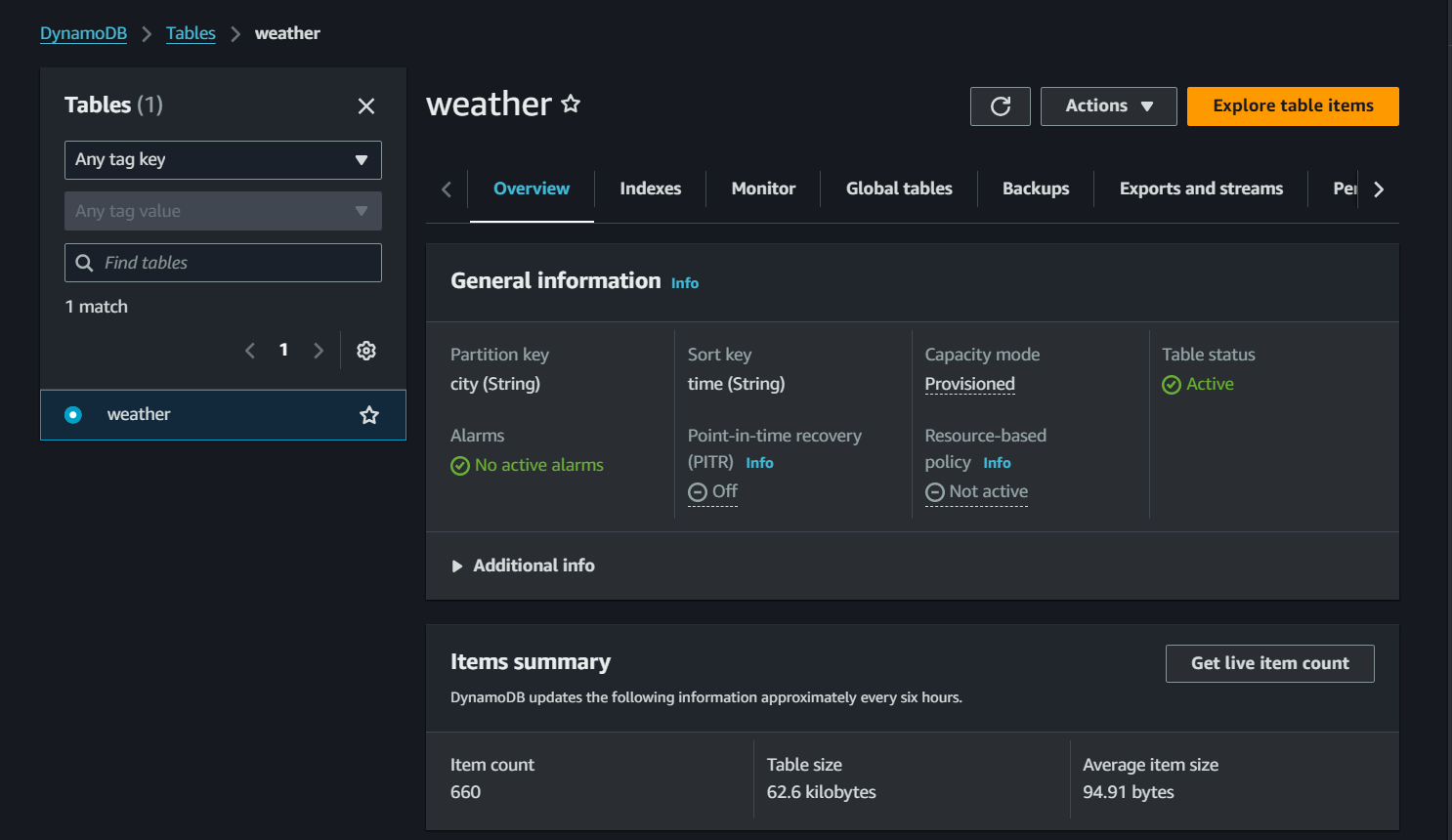
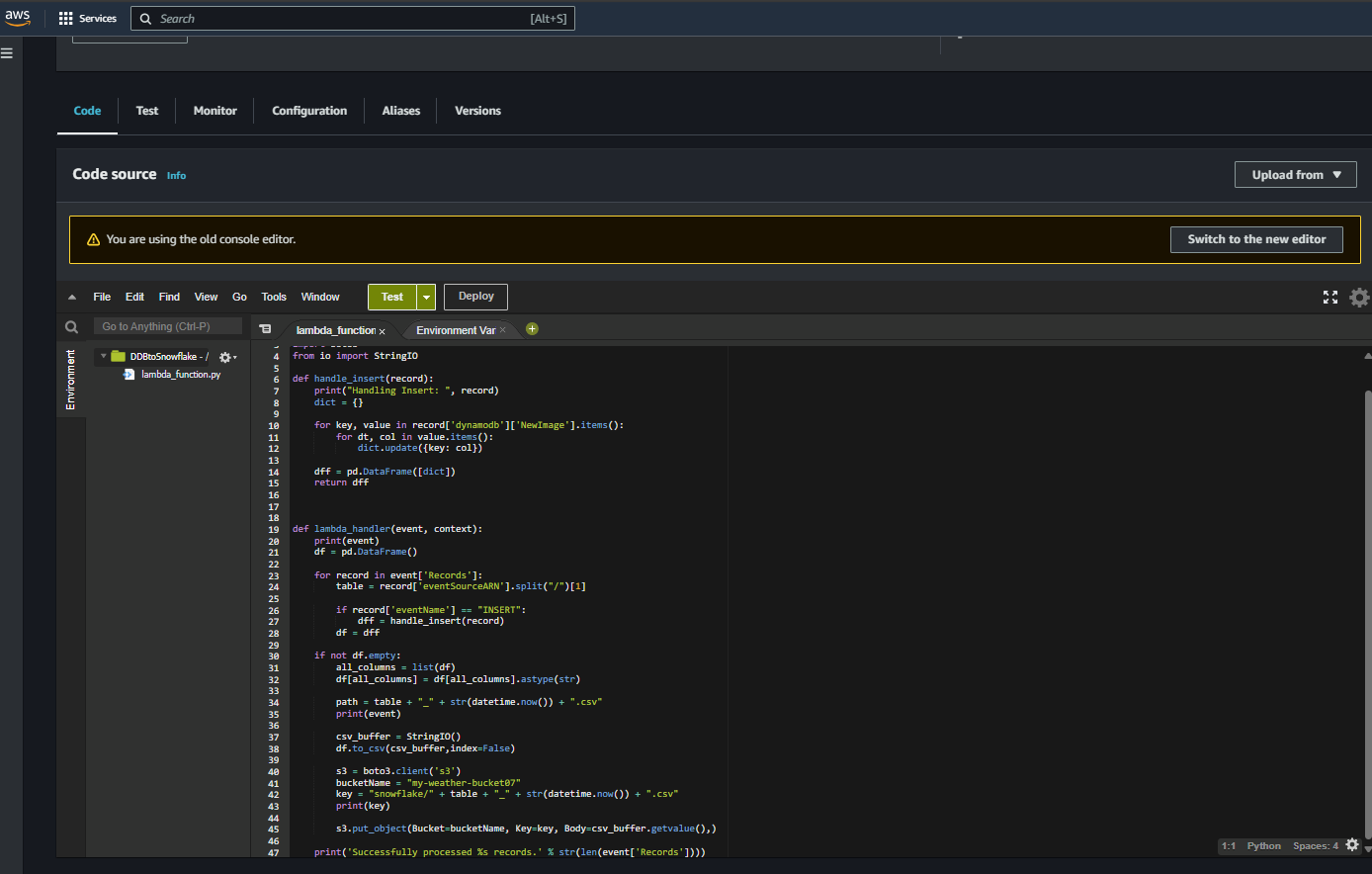
The project implementation follows a systematic approach with the following components:

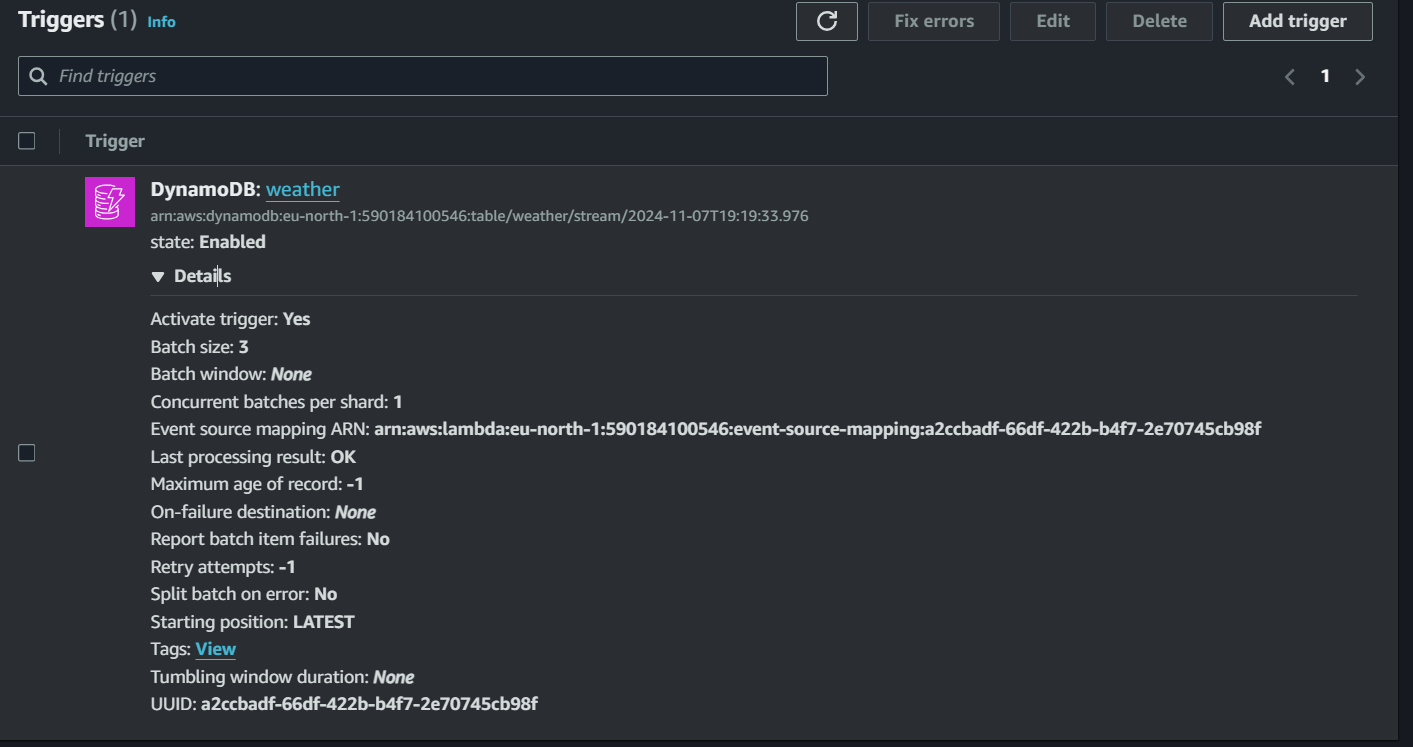
1. **AWS Lambda Function for Data Collection**
   * Created a Lambda function to fetch weather data
   * Implemented city-wise data collection
   * Set up hourly triggers using EventBridge
   * *Lambda Function:*

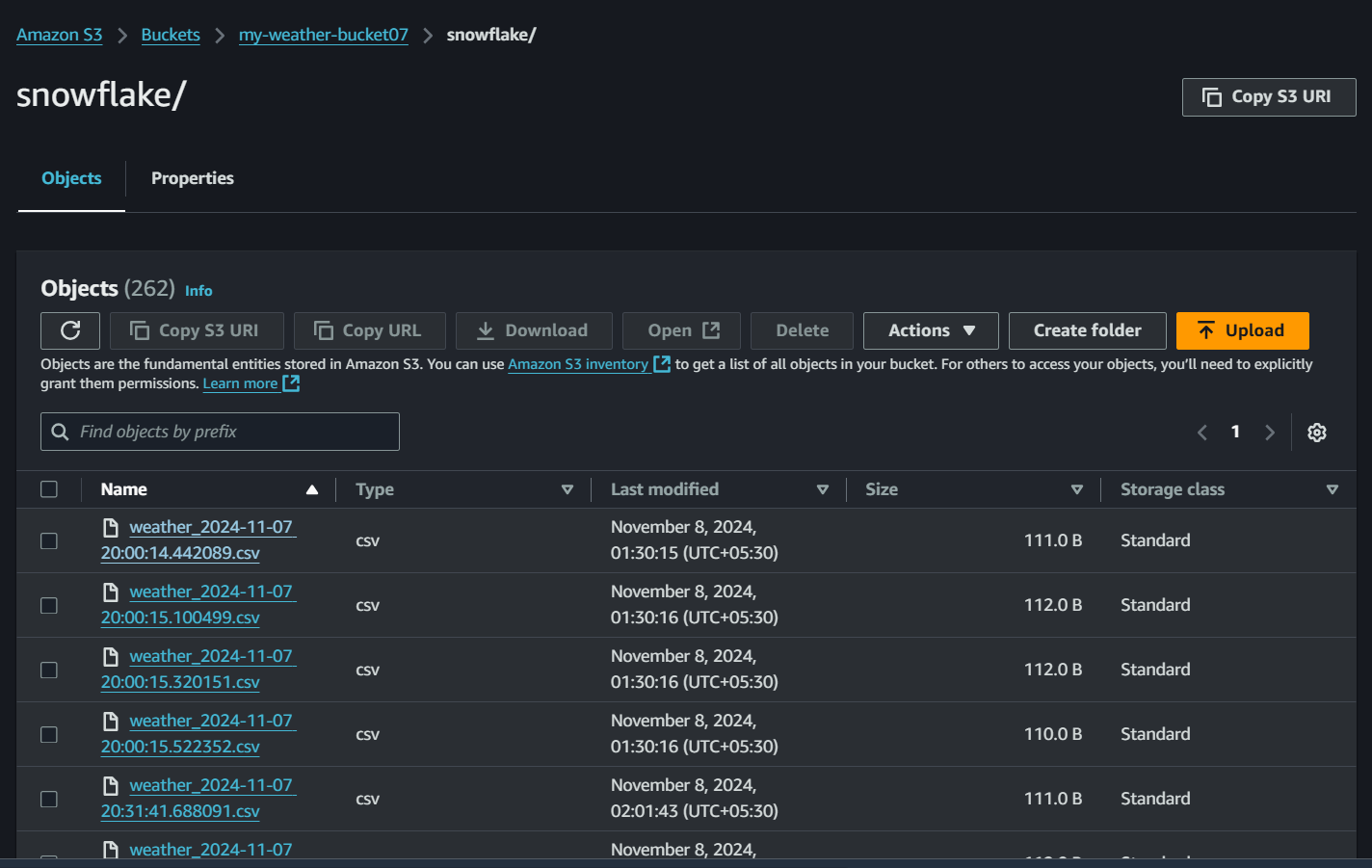


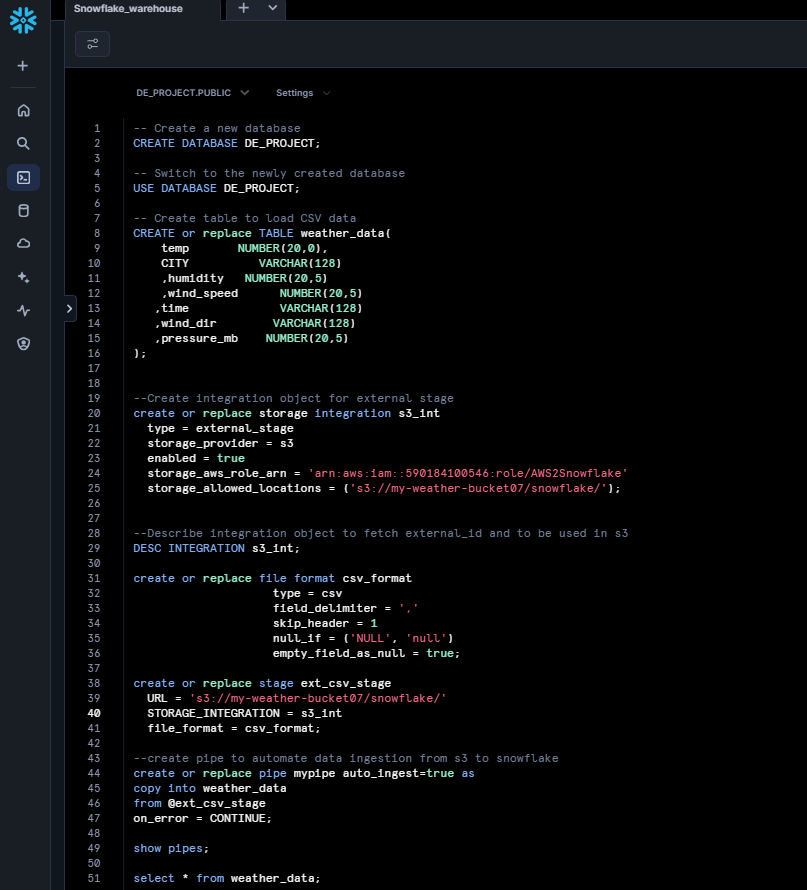
Trigger:



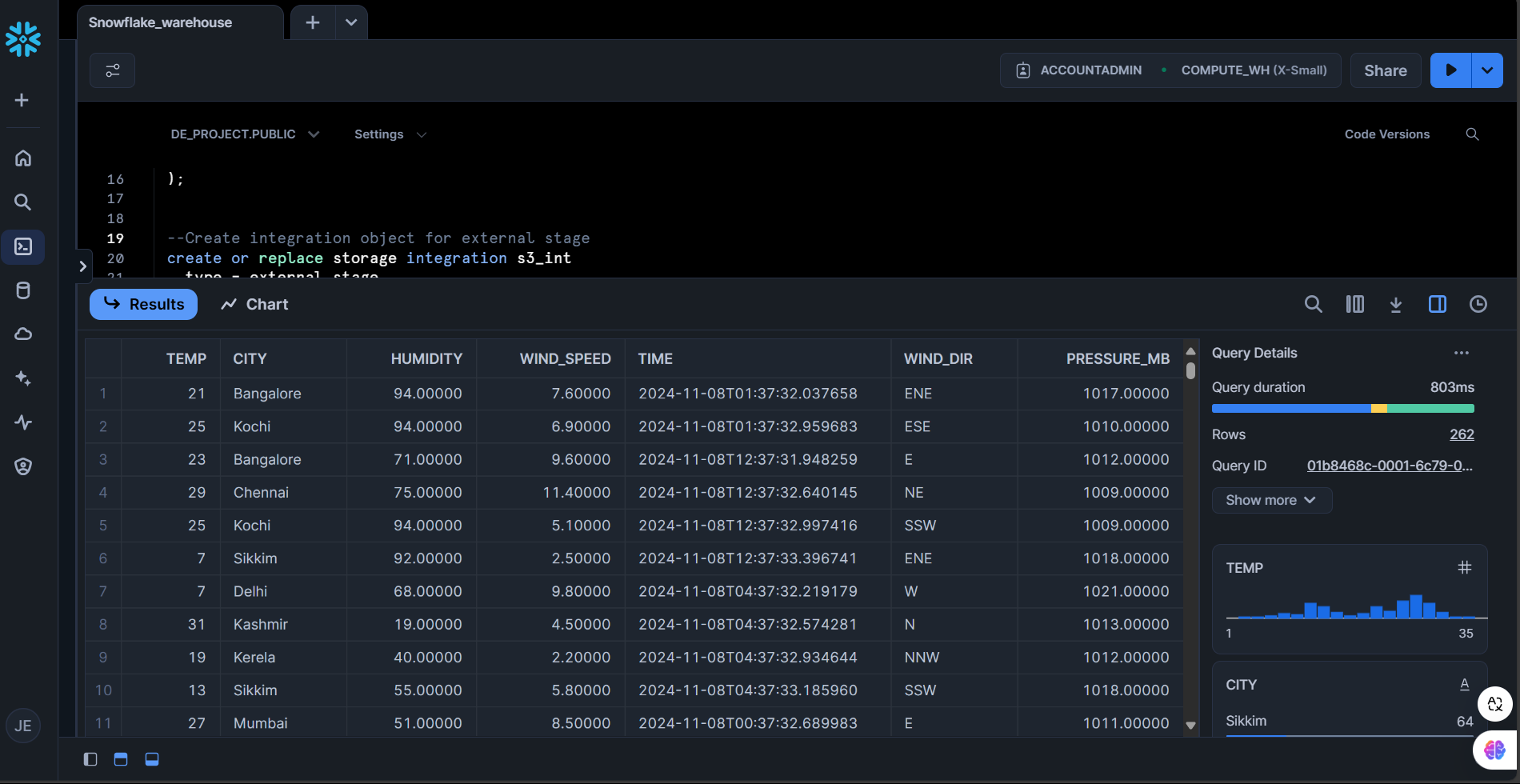
1. **DynamoDB Setup**
   * Created a DynamoDB table with appropriate schema
   * Configured partition key (city) and sort key (timestamp)
   * Enabled DynamoDB Streams for real-time data processing *DynamoDB*
   * *Config:* **
2. **Stream Processing Lambda Function**
   * Implemented a second Lambda function for stream processing
   * Created data transformation logic using pandas
   * Set up S3 bucket integration
   * *Lambda Function:* **

Trigger function: 

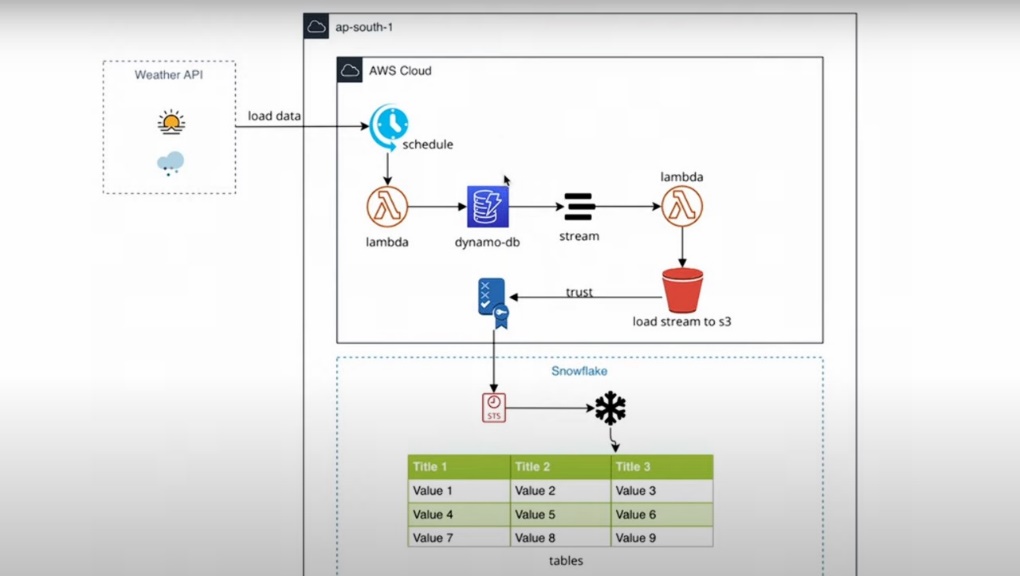
1. **S3 Integration**
   * Created an S3 bucket for storing processed data
   * Implemented CSV file storage structure
   * Set up appropriate IAM roles and permissions *[Screenshot needed: S3 bucket configuration]*
   * *S3 Bucket:* **
2. **Snowflake Integration**
   * Configured Snowflake warehouse
   * Set up external stage for S3 connection
   * Created Snowpipe for automatic data loading
   * *Snowflake Interface:*



*Snowflake warehouse Data:*

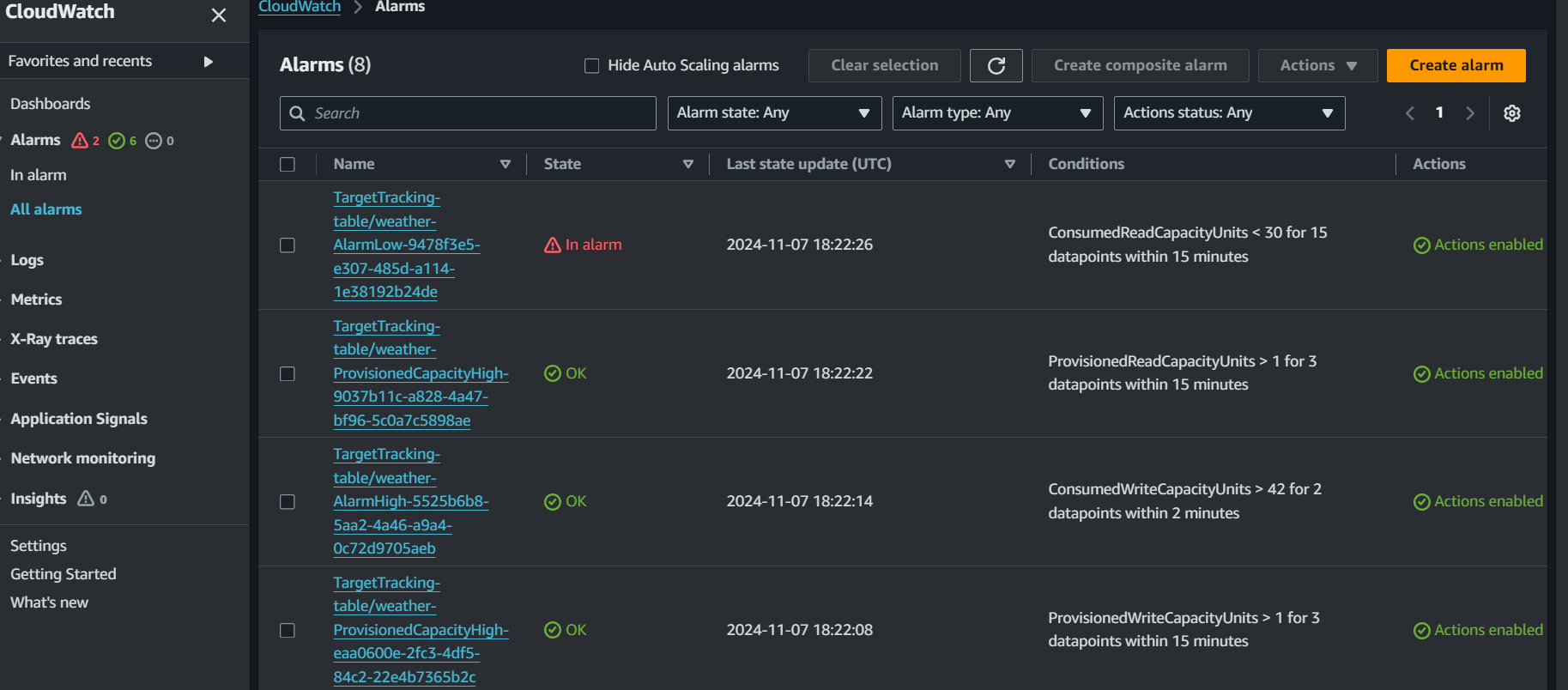
**

Architecture Diagram:



**OBSERVATIONS AND ANALYSIS**

The implementation revealed several key insights:

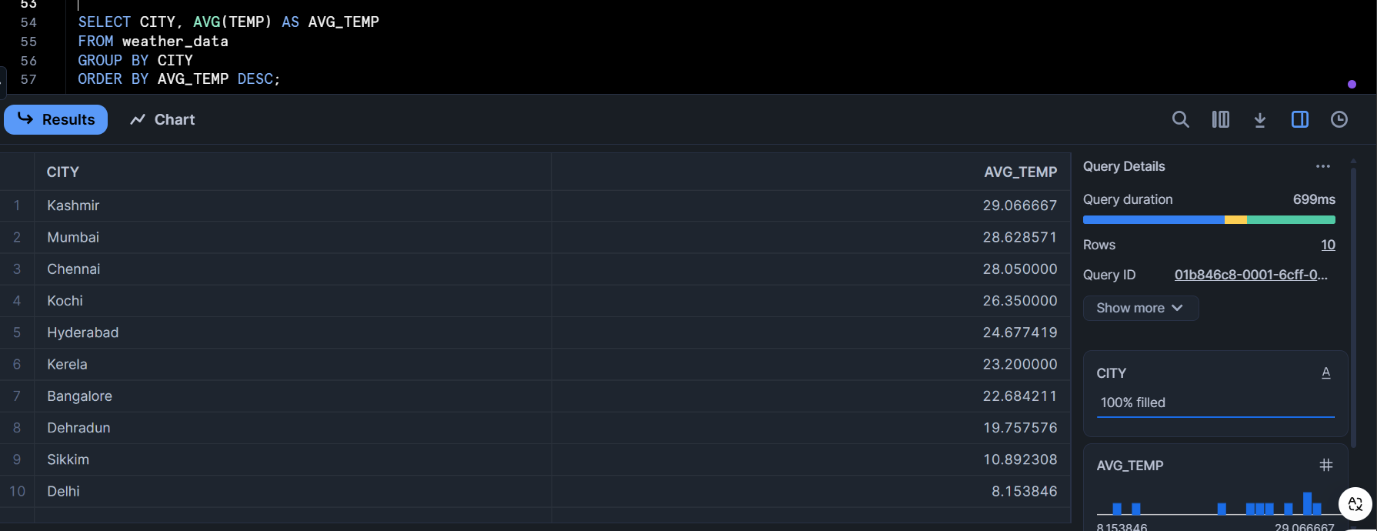
1. **System Performance**
   * Lambda functions execute reliably within the specified timeout
   * DynamoDB provides consistent performance for data storage
   * Stream processing shows minimal latency
   * *Cloud Watch:*
2. **Data Quality**
   * Weather data is consistently formatted
   * No data loss observed during stream processing
   * Successful transformation to CSV format
3. **Integration Efficiency**
   * Seamless data flow between services
   * Reliable trigger mechanisms
   * Efficient data transformation process

**RESULTS**

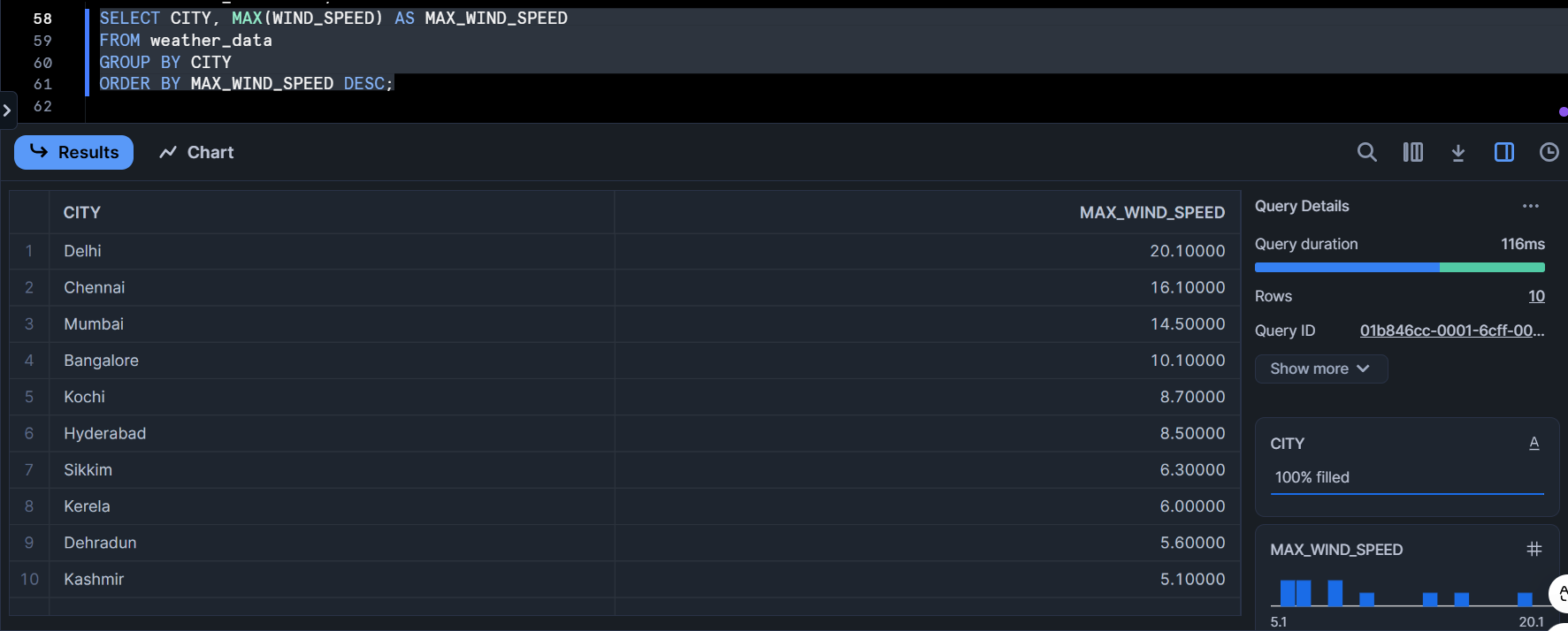
The project successfully achieved its objectives with the following outcomes:

1. **Automated Data Collection**
   * Hourly weather data collection for 10 cities
   * Reliable API integration
   * Consistent data formatting
2. **Data Processing Pipeline**
   * Efficient stream processing
   * Successful CSV conversion
   * Reliable S3 storage
3. **Analytics Capability**
   * Successfully loaded data into Snowflake
   * Enabled complex SQL queries
   * Real-time data availability.

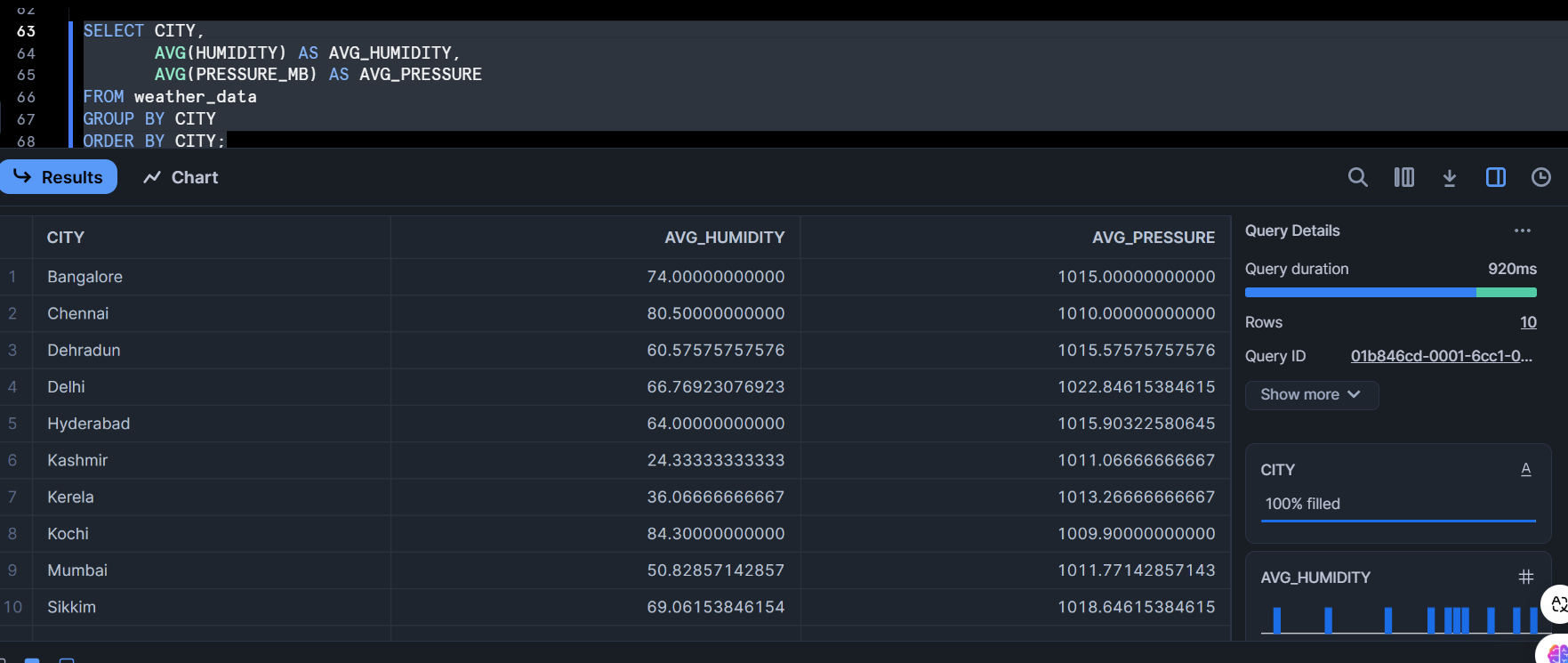
**Some Basic Analytics done using Snowflake Warehouse:**

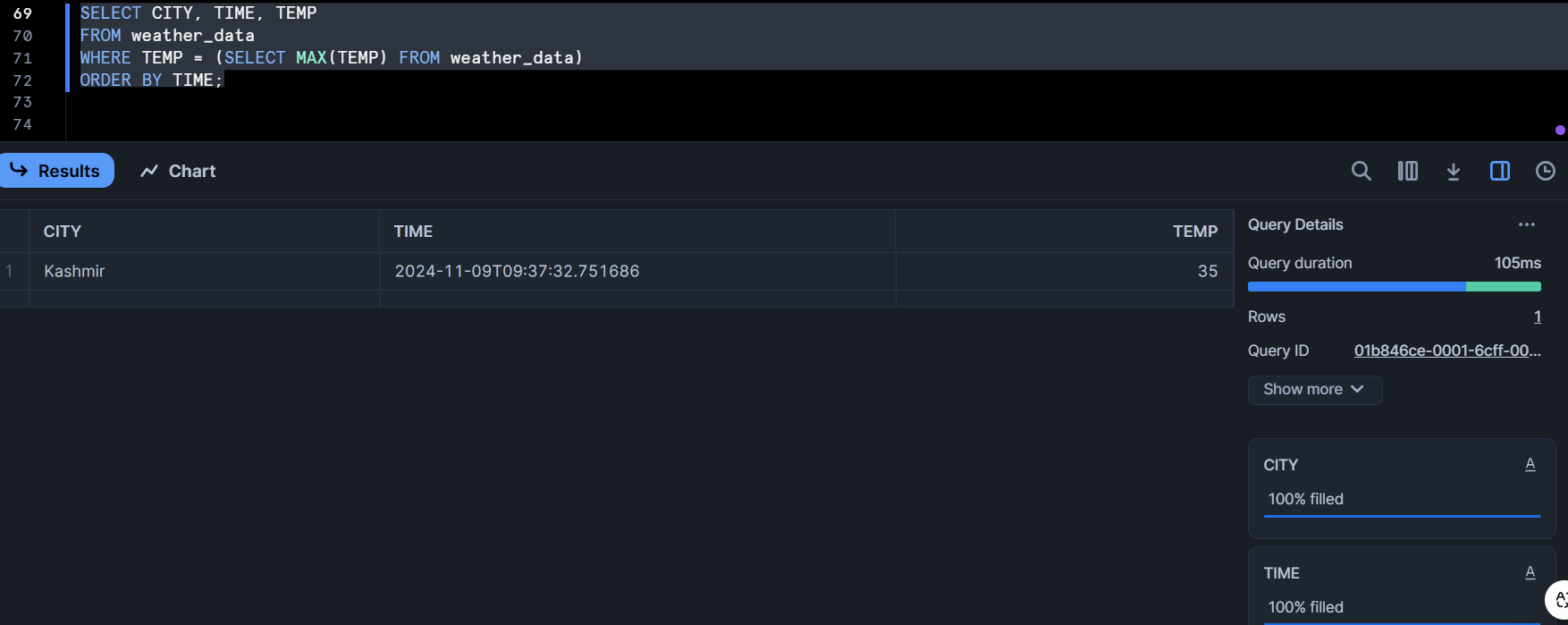
Average Temperature per City

Maximum Wind Speed per City

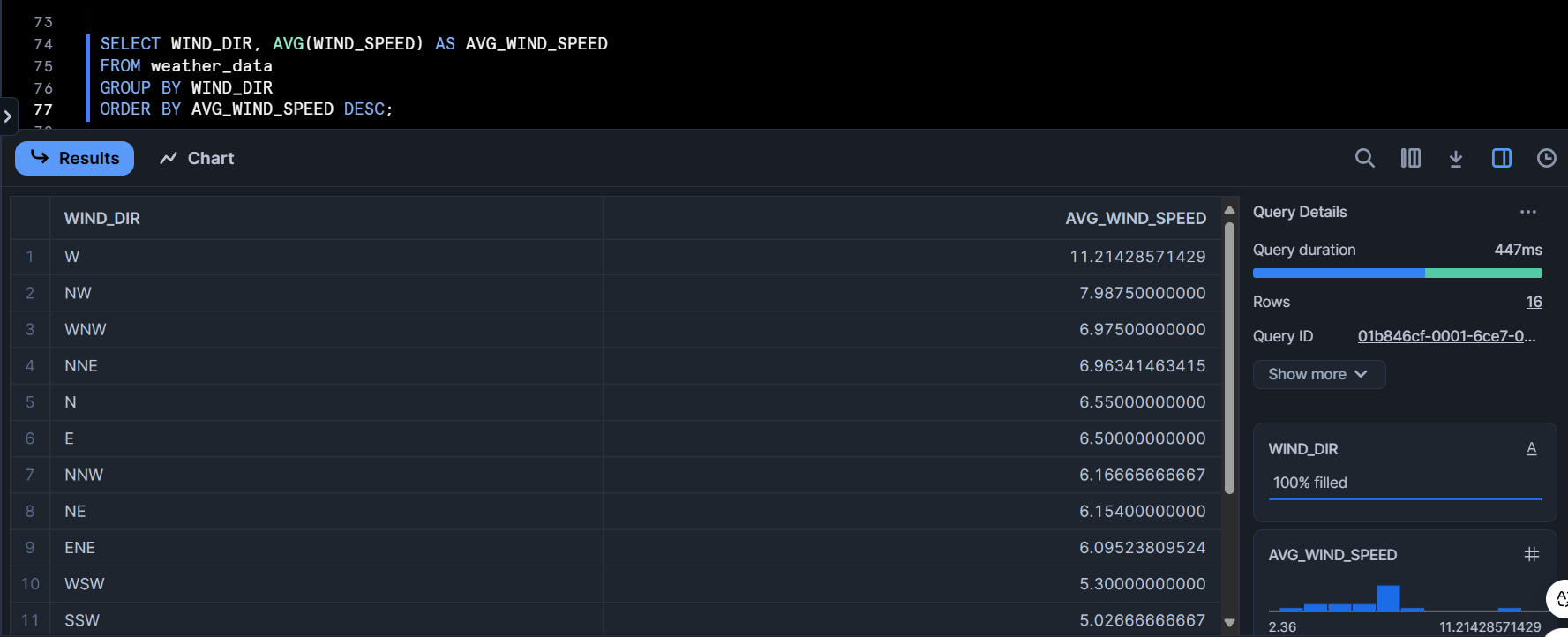


Average Humidity and Pressure by City

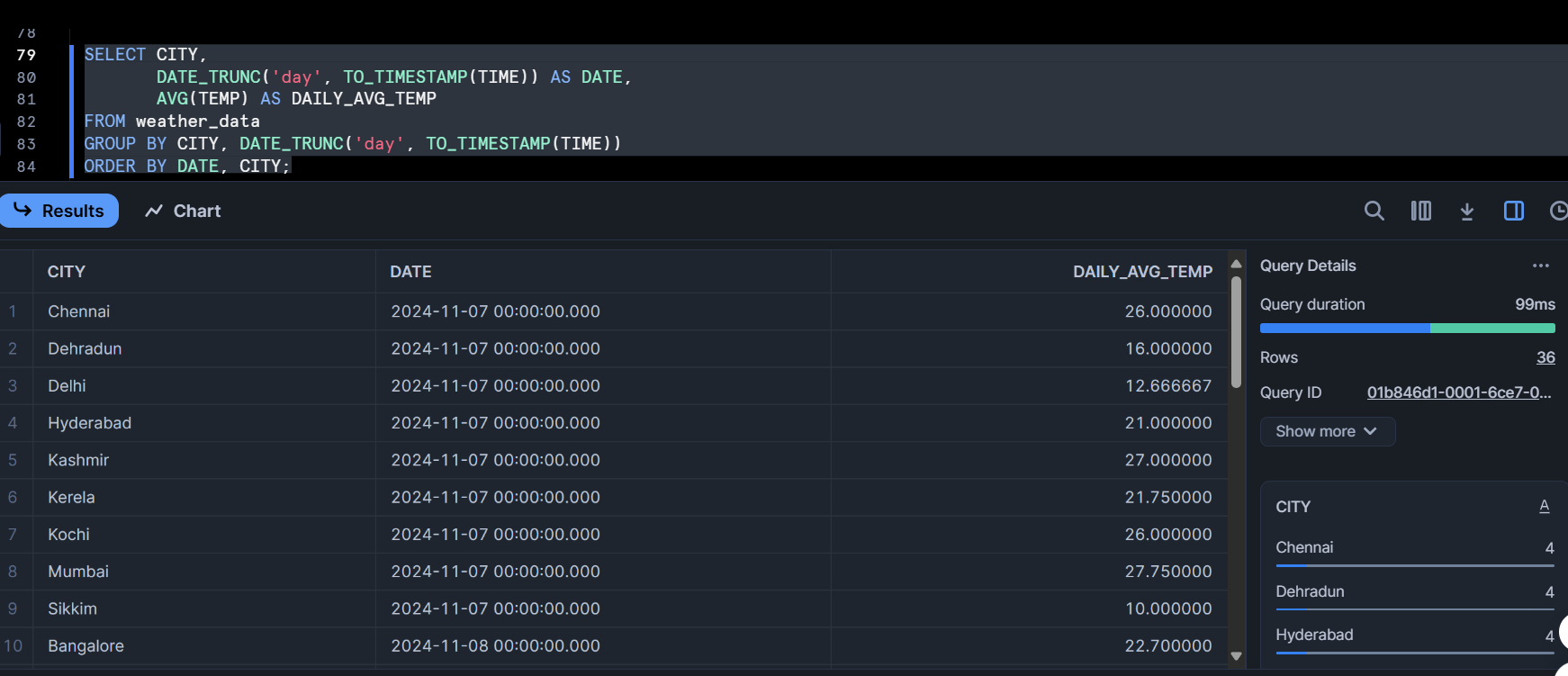
****

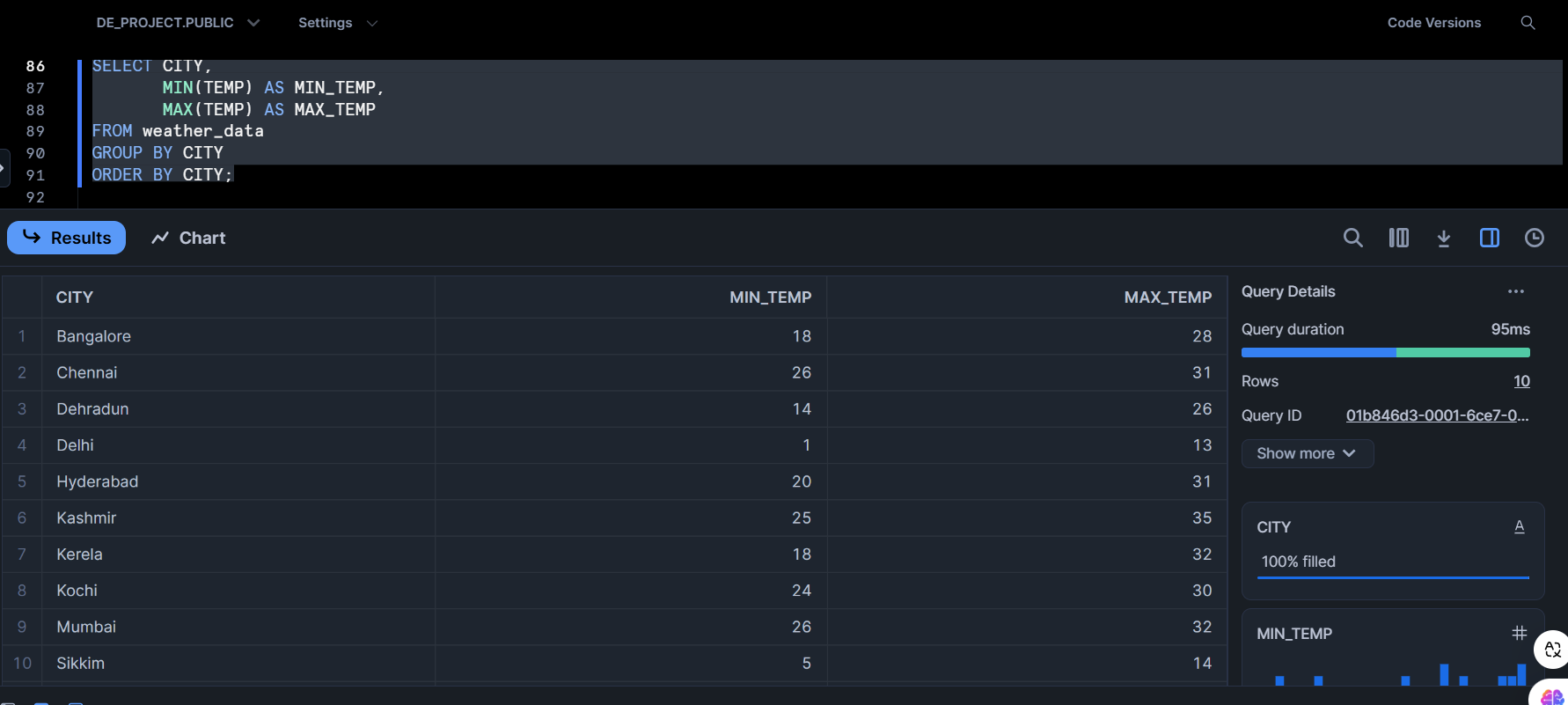
Time of Highest Temperature Recorded ****

Wind Speed Analysis by Direction

****

Daily Average Temperature for Each City

****

Calculate Temperature Range (Min and Max) per City****

**CONCLUSION**

The implemented weather data pipeline demonstrates the effectiveness of modern cloud services in building robust data engineering solutions. The serverless architecture provides scalability and cost-efficiency, while the combination of DynamoDB and Snowflake offers both transactional and analytical capabilities. The system successfully automates the entire process from data collection to analysis, providing a foundation for weather data analytics and forecasting applications.

Future enhancements could include:

* Adding more cities and weather parameters
* Implementing data validation layers
* Creating visualization dashboards
* Adding machine learning capabilities for weather prediction