Law Enforcement Call Analytics utilizing Snowflake, dbt, Airflow, and Superset

Shao-Yu Huang
San Jose State University
shao-yu.huang@sjsu.edu

Lakshmi Bharathy Kumar
San Jose State University
lakshmibharathy.kumar@sjsu.edu

***Abstract* — In this project, we present a data pipeline built to analyze emergency dispatch data sourced from the City of San Francisco's open data portal. The objective is to provide insights into police response efficiency and workload distribution by district. The pipeline integrates modern data tools, including Apache Airflow, dbt (data build tool), Snowflake, and Apache Superset. This report outlines our ETL and ELT architecture, transformation logic, automation strategies, and final data visualizations.**

I. Problem Statement

Law enforcement agencies must ensure timely responses to emergency calls. However, with high dispatch requests, assessing performance and optimizing resource allocation is critical. This project aims to build an automated pipeline that calculates response time metrics, identifies high-demand districts, and surfaces these insights in real-time dashboards.

II. Requirements and Specifications

1. Download Docker Desktop and run “docker compose up”.
2. Set up Snowflake Variables:
    a. lab2_LawInforcement_url: https://data.sfgov.org/resource/gnap-fj3t.csv
    b. database: USER_DB_LION
    c. schema_name: raw
3. Set snowflake connections, and the id name is “snowflake_conn”

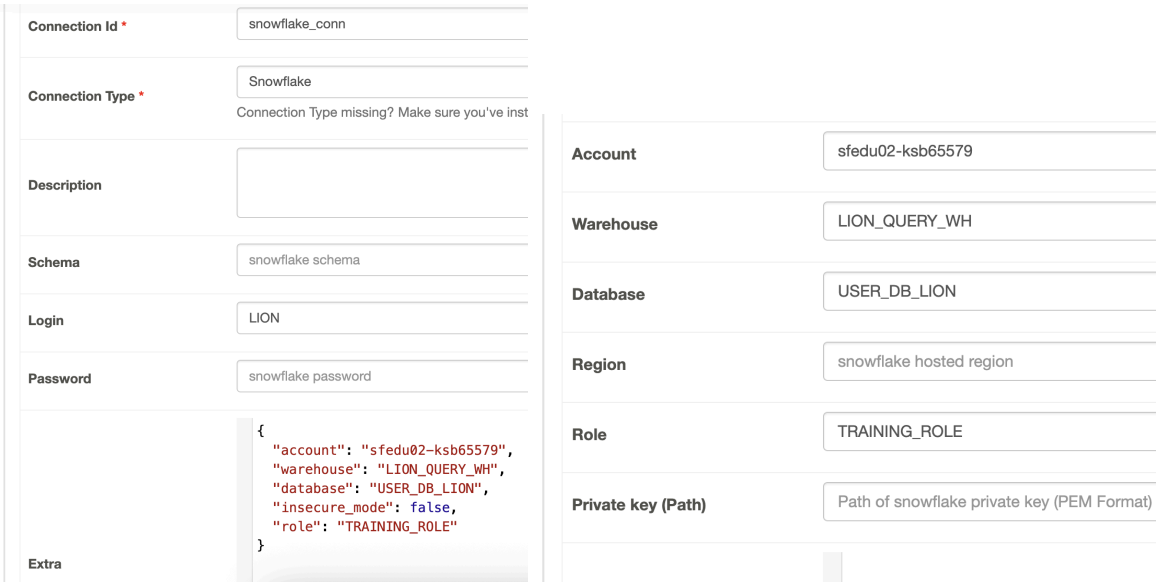

Fig 1. Snowflake connection in Airflow

III. Overall System Diagram

The data pipeline is structured to efficiently handle the extraction, transformation, loading, and visualization of law enforcement dispatch data. The architecture comprises the following components:

Data Source: The pipeline begins with ingesting raw data from the San Francisco Open Data portal, specifically the "Law Enforcement Dispatched Calls for Service" dataset.

Apache Airflow: Airflow orchestrates the workflow through Directed Acyclic Graphs (DAGs). It schedules and manages tasks such as data extraction, loading into Snowflake, and triggering dbt transformations.

Snowflake: Serving as the cloud data warehouse, Snowflake stores both raw and transformed data. The raw data is loaded into a designated schema, and subsequent transformations are also stored within Snowflake for consistency and performance.

dbt (Data Build Tool): dbt handles the transformation layer. It defines models that clean, aggregate, and structure the raw data into meaningful tables, facilitating easier analysis and reporting.

Apache Superset: Superset connects directly to Snowflake to visualize the transformed data. It provides dashboards and charts that offer insights into key metrics such as response times and incident counts.

IV. Data Flow Overview:

Extraction: Airflow initiates the process by extracting data from the public dataset.

Loading: The extracted data is loaded into the raw schema within Snowflake.

Transformation: DBT models are executed to transform the raw data into structured formats suitable for analysis.

Visualization: Superset accesses the transformed data in Snowflake to generate interactive dashboards and visualizations.

Fig 2. Overall system architecture of the end-to-end data analytics pipeline integrating ETL and ELT processes using Airflow, Snowflake, dbt Cloud, and Superset.

IV. Detailed table structure

The primary dataset used in this project is the Law Enforcement Dispatched Calls for Service – Real-Time dataset, made publicly available by the City and County of San Francisco [1]. This dataset provides near real-time information on incidents reported through the 911 dispatch system and responded to by the San Francisco Police Department (SFPD).

Although the full dataset contains a wide range of attributes related to law enforcement operations, this project focuses on a curated subset of columns that are directly relevant to response performance analysis. Specifically, the following fields are used:

id: Unique identifier for each incident

cad_number: Dispatch number assigned by the 911 system

received_datetime: When the call was received or the incident was self-initiated

dispatch_datetime: When the first unit was dispatched

onscene_datetime: When the first unit arrived at the incident location

police_district: The district in which the incident occurred

response_time_min: A derived field representing the number of minutes between dispatch and on-scene arrival

In addition to the core fields extracted from the original dataset, several computed columns were included in the raw_incidents table to enhance the granularity of incident lifecycle analysis. These fields provide further insights into different stages of a police unit's response timeline:

enroute_to_dispatch_min (float): Represents the duration (in minutes) between the dispatch time and the moment the responding unit marked itself en route. This helps assess potential delays in mobilization after being dispatched.

onscene_to_enroute_min (float): Captures the time taken by the unit to arrive at the scene after it began en route. This metric reflects real-world travel or navigation efficiency.

total_incident_duration_min (float): Measures the total duration from the initial dispatch to the arrival on scene. This holistic metric is useful for evaluating total response efficiency.

These computed columns were added to the raw schema to support more detailed time-based performance analytics in both the dbt transformation stage and Superset dashboard visualizations. Their inclusion supports a deeper breakdown of response delays and variations across districts and periods.

Table: raw.law_enforcement_calls

FIeld	**Type**	**Constraints**
Id	string	Primary key
cad_number	String	Not null
received_datetime	Timestamp	Not null
dispatch_datetime	Timestamp	Not null
onscence_datetime	Timestamp	Not null
close_datetime	Timestamp	Not null
police_district	string	Not null
dispatch_to_received_min	float	computed
enroute_to_dispatch_min	float	computed
onscene_to_enroute_min	float	computed
total_incident_duration_min	float	computed

Table. 1. Field definitions for the raw.law_enforcement_calls table.

Table: analytics.final_district_performance

Field	Type	Description
police_district	string	Group identifier
Avg_response_time_mi	float	Average total response time per district
district_rank	Integer	Rank based on response time
total_case	Integer	Number of dispatch cases in

Table. 2. Field definitions for the analytics.final_district_performance table.

V. Airflow

Airflow ETL pipeline:

The Airflow DAG for ETL is named "law_enforcement_ETL" and orchestrates the ETL process for importing and preparing law enforcement dispatch data. This DAG is scheduled to run daily at 2:30 AM (PDT), as specified by the cron expression 30 2 * * *.
The ETL pipeline consists of three main tasks:

1. **extract_data:**
   This task downloads raw data from the San Francisco open data API [1] and stores in Snowflake. It ensures that only new or updated records are fetched using idempotent logic with timestamps or record IDs.
2. **transform_data:**
   This task processes the extracted data by cleaning and standardizing timestamps, filtering necessary columns, and computing additional fields such as response_time_min, enroute_to_dispatch_min, and total_incident_duration_min. This logic is implemented in Python and/or SQL and prepares the dataset for loading into the warehouse.
3. **load_data:**
   This task loads the transformed dataset into the RAW.LAW_ENFORCEMENT_CALLS table within the Snowflake data warehouse. It uses transactions and error handling to ensure data consistency and prevent duplicate entries.

Airflow ELT pipeline:

This project implements an automated **ELT (Extract, Load, Transform)** pipeline for analyzing emergency law enforcement calls using **dbt (Data Build Tool)** and **Apache Airflow**. The pipeline is designed to extract meaningful insights from raw call data by transforming it into clean, analytical models that support operational dashboards and decision-making.

1. **Extract & Load:**

    - Raw data from emergency law enforcement calls is ingested and loaded into Snowflake (schema: `raw`).

    - This dataset contains fields such as `cad_number`, `received_datetime`, `dispatch_datetime`, `onscene_datetime`, and `police_district`.

2. **Transform (dbt):**

    - The dbt project transforms raw call data into structured models:

    - `law_enforcement_calls`: a cleaned base model selecting and calculating response intervals

    - `police_district_summary`: aggregates total cases and average response time per district

    - Models are materialized as views or tables in the `analytics` schema.

3. **Scheduled Orchestration (Airflow):**

    - A dedicated Airflow DAG (`dbt_build.py`) orchestrates the dbt workflow using `BashOperator` tasks:

    - `dbt run`: builds all dbt models

    - `dbt test`: validates model quality with schema and custom tests

    - `dbt snapshot`: optionally captures slowly changing dimensions

    - The DAG uses environment variables sourced from Airflow's Snowflake connection and is scheduled manually or periodically.

Each task is represented as a separate node in the DAG and runs sequentially from extraction to transformation to loading. The DAG is monitored through Airflow's Web UI, as shown in the figure, where task statuses are visualized with color-coded blocks. The execution history demonstrates that the pipeline has run multiple times successfully, with retry and failure handling implemented as needed.

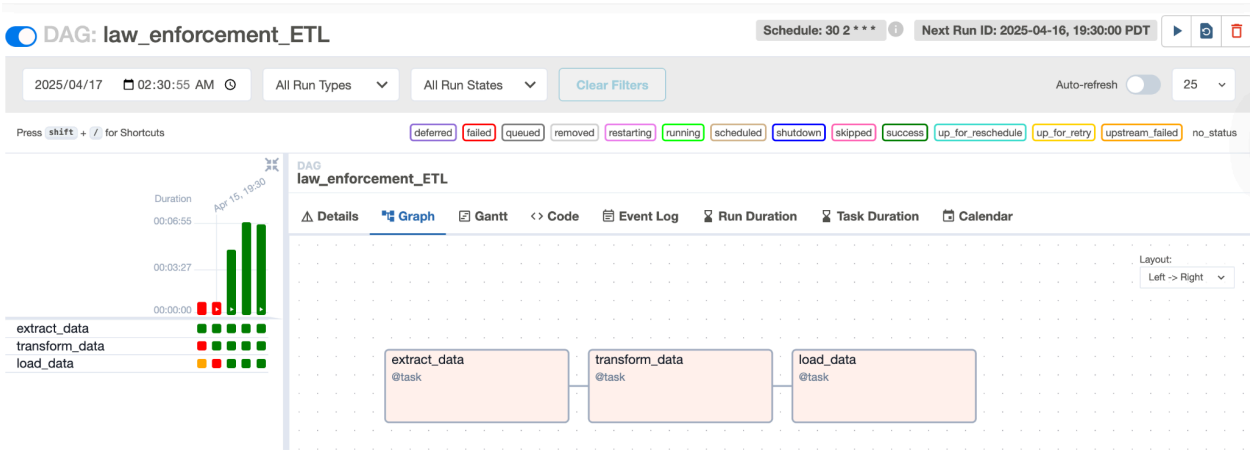

Fig. 3. Airflow DAG Pipeline of the law_enforcement_ETL.

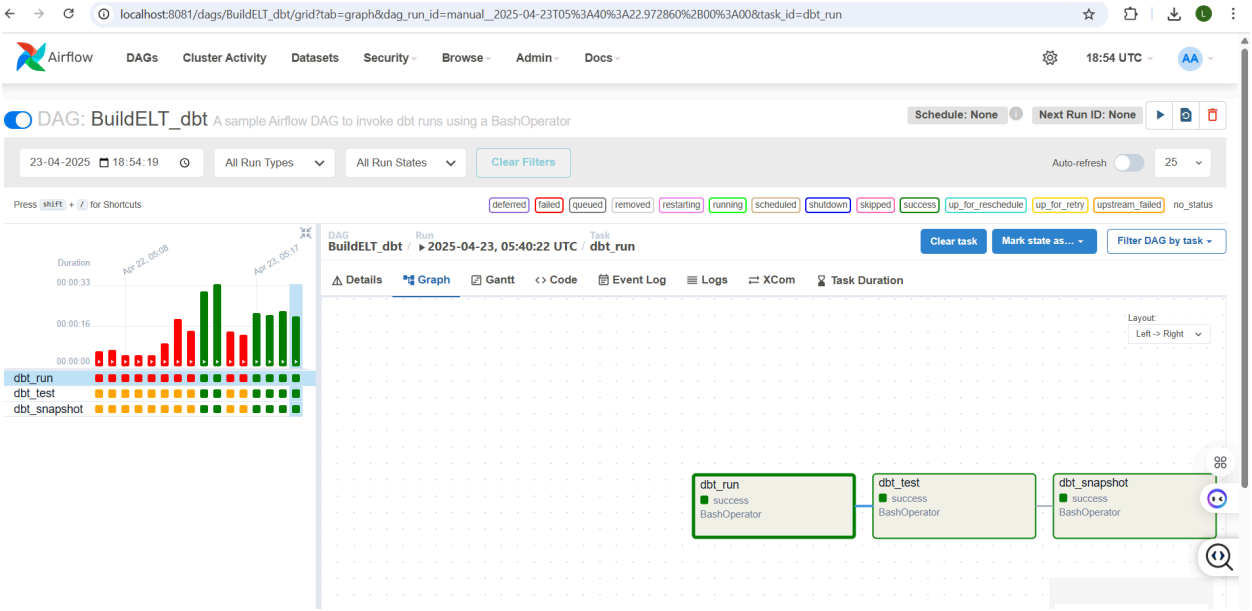

Fig. 4. Airflow DAG Pipeline of BuildELT.

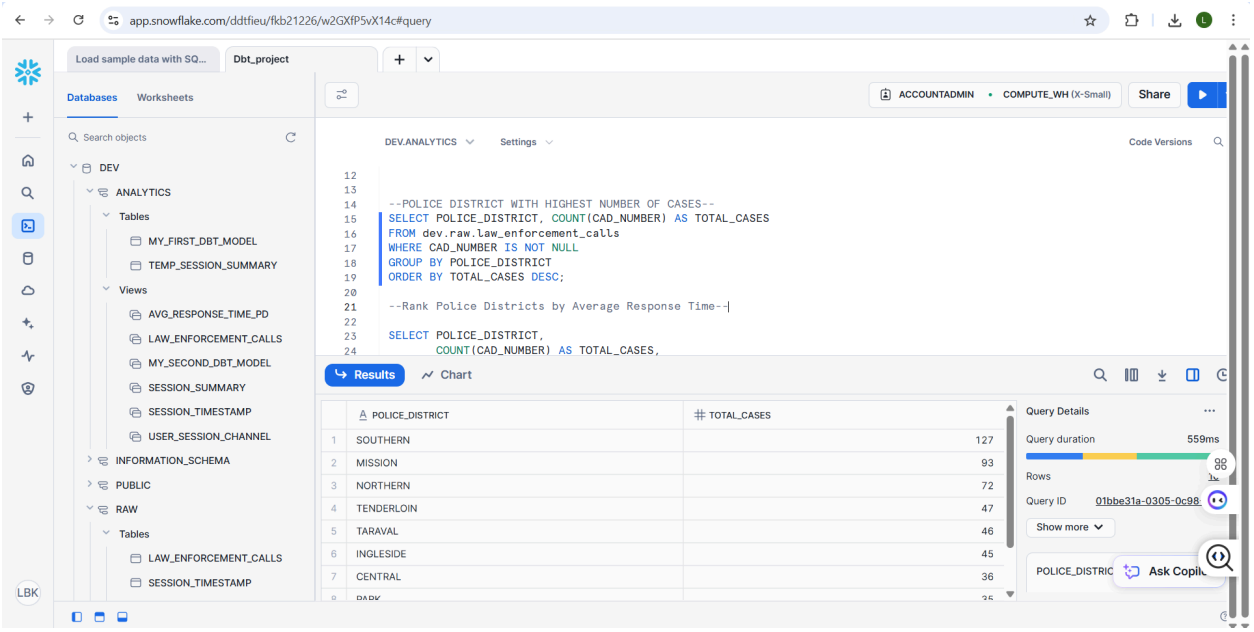

Fig. 5. Final tables and views in Snowflake

VI. Dbt

Data Build Tool (dbt) is an open-source command-line tool that enables data analysts and engineers to transform raw data into analytics-ready datasets within modern data warehouses. Adopting a modular SQL-based workflow and software engineering best practices—such as version control, testing, and documentation—dbt simplifies the development and maintenance of data transformation pipelines.

In this project, dbt was used to (1) run transformation models, (2) validate data quality with tests, and (3) track historical records using snapshots. Fig. 6 shows the result of executing the dbt run command, which materializes transformation logic into tables or views. Fig. 7 demonstrates the output of dbt test, confirming the integrity of the dataset using not-null and uniqueness constraints. Fig. 8 shows the execution of the dbt snapshot command, which captures slowly changing dimensions over time for audit or trend analysis.

```
(dbt_env) (base) shao@Mac lab2 % dbt run
06:48:50  Running with dbt=1.9.4
06:48:50  Registered adapter: snowflake=1.9.2
06:48:51  Found 2 models, 1 snapshot, 2 data tests, 2 sources, 474 macros
06:48:51
06:48:51  Concurrency: 1 threads (target='dev')
06:48:51
06:48:52  1 of 1 START sql table model analytics.Avg_response_time_pd ..... [RUN]
06:48:53  1 of 1 OK created sql table model analytics.Avg_response_time_pd ..... [SUCCESS 1 in 1.20s]
06:48:53
06:48:53  Finished running 1 table model in 0 hours 0 minutes and 2.09 seconds (2.09s).
06:48:53
```

Fig. 6. dbt result for “dbt run” command.

```
(dbt_env) (base) shao@Mac lab2 % dbt test
06:49:30  Running with dbt=1.9.4
06:49:30  Registered adapter: snowflake=1.9.2
06:49:30  Found 2 models, 1 snapshot, 2 data tests, 2 sources, 474 macros
06:49:30
06:49:30  Concurrency: 1 threads (target='dev')
06:49:30
06:49:30  1 of 2 START test source_not_null_raw_law_enforcement_calls_cad_number ......... [RUN]
06:49:31  1 of 2 PASS source_not_null_raw_law_enforcement_calls_cad_number ............... [PASS in 0.46s]
06:49:31  2 of 2 START test source_not_null_raw_law_enforcement_calls_police_district .... [RUN]
06:49:31  2 of 2 PASS source_not_null_raw_law_enforcement_calls_police_district .......... [PASS in 0.39s]
06:49:31
06:49:31  Finished running 2 data tests in 0 hours 0 minutes and 1.53 seconds (1.53s).
06:49:32
06:49:32  Completed successfully
06:49:32
06:49:32  Done. PASS=2 WARN=0 ERROR=0 SKIP=0 TOTAL=2
```

Fig. 7. dbt result for “dbt test” command.

```
(dbt_env) (base) shao@Mac lab2 % dbt snapshot
06:49:57  Running with dbt=1.9.4
06:49:58  Registered adapter: snowflake=1.9.2
06:49:58  Found 2 models, 1 snapshot, 2 data tests, 2 sources, 474 macros
06:49:58
06:49:58  Concurrency: 1 threads (target='dev')
06:49:58
06:49:58  1 of 1 START snapshot snapshots.district_response_snapshot ..................... [RUN]
06:50:00  1 of 1 OK snapshotted snapshots.district_response_snapshot ..................... [SUCCESS 1 in 1.20s]
06:50:00
06:50:00  Finished running 1 snapshot in 0 hours 0 minutes and 2.06 seconds (2.06s).
06:50:00
06:50:00  Completed successfully
06:50:00
06:50:00  Done. PASS=1 WARN=0 ERROR=0 SKIP=0 TOTAL=1
```

Fig. 8. dbt result for “dbt snapshot” command.

Ⅶ. Visulization

Superset Dashboard Description

The Superset dashboard provides a visual analytical interface to explore and evaluate police district performance based on operational response efficiency. It is built upon a transformed dataset created using dbt (Data Build Tool) and sourced from emergency call records stored in Snowflake. The purpose of the dashboard is to deliver actionable insights into emergency response operations by evaluating both workload (number of cases) and responsiveness (average response time) across various police districts.

Key Visualizations Included:

1. **Bar Chart** – Total Cases by Police District
    - This chart visualizes the distribution of emergency call volume across all police districts.
    - It enables comparative analysis of incident density, helping to identify high-activity zones.

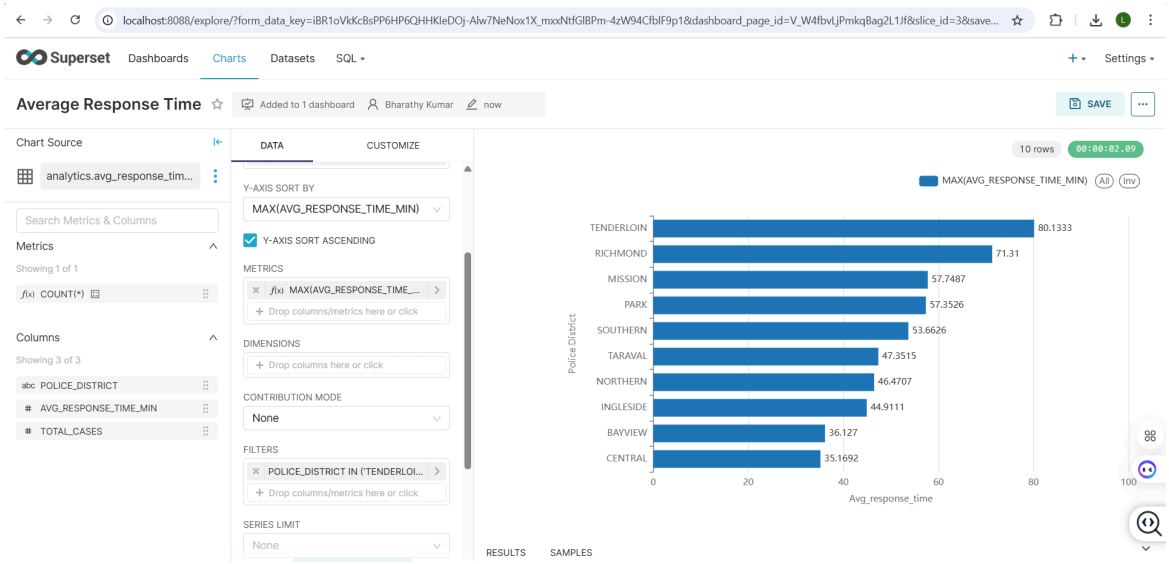

Fig.9. Bar Chart

2. **Pie Chart** – Proportion of Cases per District

- A proportional view of case volume, representing each district's share of the total incidents.
- Useful for quickly identifying which districts contribute most heavily to the call load.

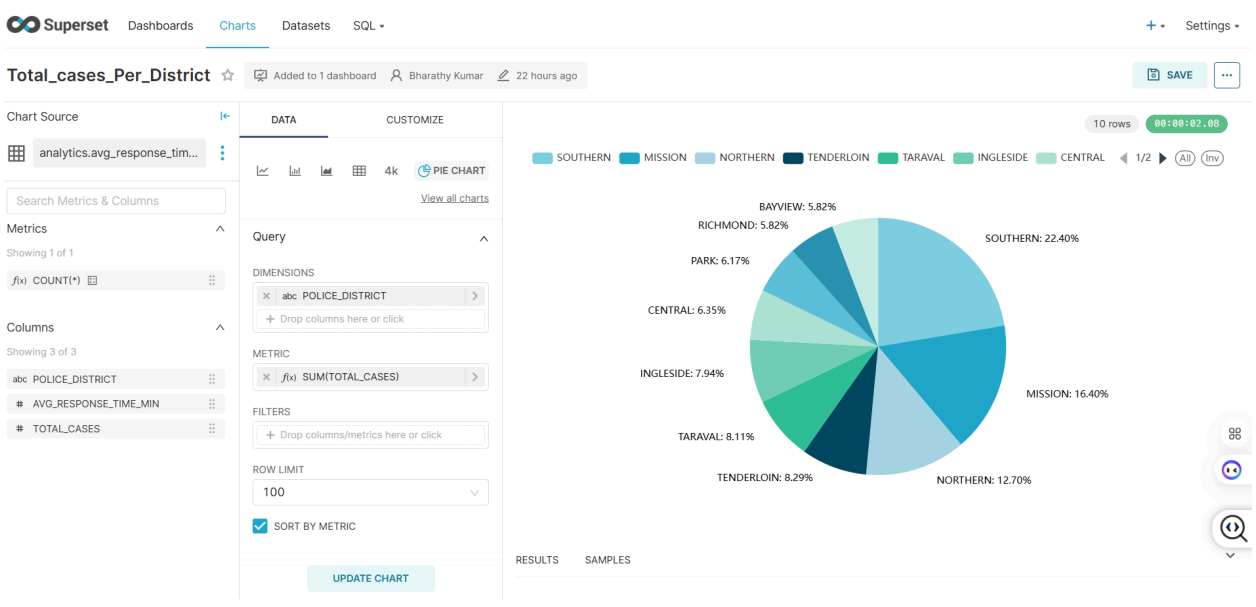

Fig. 10.Pie chart

3. **Bubble Chart** – Case Volume vs. Response Time

Plots police districts as individual bubbles using:

- X-axis: Average Response Time
- Y-axis: Total Number of Cases
- Bubble Size: Total Case Count

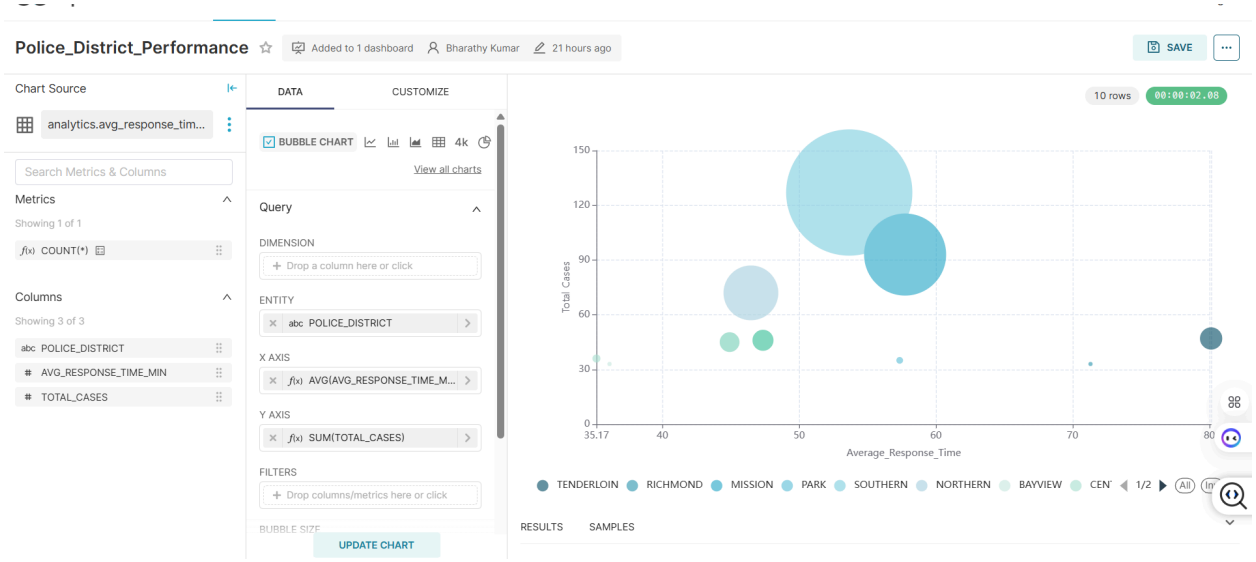

Fig. 11.Bubble Chart

Highlights high-performing districts (high volume + fast response) and underperforming areas (slow response or low efficiency). Utilizes the Avg_response_time dbt model as its source.

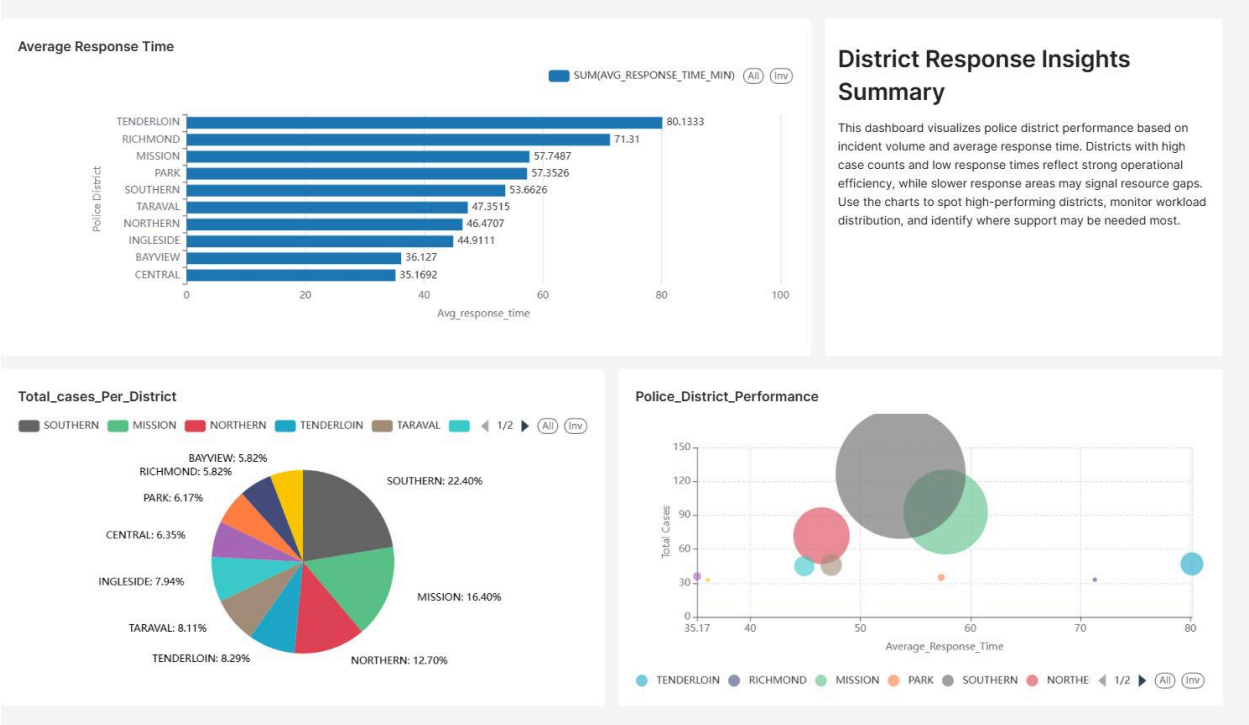

Fig 12. Output result of the Law_Enforcement_Performance_Analysis.

VIII. Conclusion

This lab successfully demonstrates the implementation of an end-to-end data analytics pipeline by integrating Apache Airflow, dbt Cloud, Snowflake, and Superset. Raw law enforcement data from San Francisco's open data portal was ingested into Snowflake using Airflow ETL tasks, ensuring automation and traceability. dbt Cloud was then used to perform ELT transformations, such as calculating daily case counts and average response times and creating clean and analysis-ready tables. These models were scheduled and orchestrated through Airflow, enabling a robust and repeatable workflow. Finally, the processed data was visualized in Superset, providing stakeholders with interactive dashboards to monitor public safety trends and district-wise response performance. This lab highlights the power of combining cloud data warehousing, data transformation, workflow orchestration, and modern BI tools to derive actionable insights from open datasets.

IX. References

[1] San Francisco Open Data, “Police Department Incident Reports.” https://data.sfgov.org/resource/gnap-fj3t.csv

[2] dbt Labs, “dbt (Data Build Tool).”. Available: https://www.getdbt.com/

[3] The Apache Software Foundation, “Apache Airflow.” https://airflow.apache.org/

[4] The Apache Software Foundation, “Apache Superset.” https://superset.apache.org/

[5]Snowflake Inc., “Snowflake Cloud Data Platform.” https://www.snowflake.com/

Appendix A — Code Repository Structure

All relevant source code for this project is available in the GitHub repository Pair20_Lab2. The repository is structured into the following subdirectories:

- ETL_code: Contains all scripts for data extraction, transformation, and loading.
- ELT_code: Contains dbt models for building analysis-ready tables, designed to be scheduled and run via Apache Airflow for automated ELT.
- Pair20_Lab2_DBT: The complete dbt project folder, including configuration, models, and snapshots.
- dbt_input: Raw input models used as the basis for data transformation in dbt.
- dbt_output: Materialized models generated after transformations.
- dbt_snapshot: Snapshot definitions and outputs for tracking slowly changing dimensions.