Technical documentation

Project: GitHub Events Analyzer

Date: May 4, 2025

Version: 1.0

Organization: GDC Fusion - Global Development Club | Wizeline

---

Introduction

GitHub is the leading collaboration platform for software development, generating millions of daily events that reflect technological trends, the most widely used programming languages, and the behavior of open-source communities. Analyzing this information enables organizations to gain key insights to stay at the forefront of the technology sector.

As part of the GDC Fusion Hackadata, this project proposes a solution for processing public GitHub events from GH Archive, transforming raw data into structured and actionable information. The solution is based on a modern data pipeline implemented using a Medallion architecture, which organizes the data into layers (Bronze, Silver, Gold) to facilitate its cleaning, traceability, and analysis.

Objectives

General Objective

To transform chaotic GitHub event data into structured and meaningful information through a robust data architecture, enabling predictive analysis and visualizations with strategic value.

Specific Objectives

1. To implement a cloud-based storage system to host raw files and design a transformation pipeline based on the Medallion architecture, using tools such as DBT, ensuring data availability, cleanliness, and progressive structuring.
2. To apply exploratory data analysis, segmentation, and predictive modeling techniques to the processed data in order to identify patterns, anomalies, and trends, and to develop interactive dashboards with a narrative approach that effectively communicate key findings to both technical and strategic audiences.

Technical Requirements

1. **Programming Language:** Python 3.8 or later
2. **Database:** SQL
3. **Specific Python Libraries:** Pandas, Snowpark
4. **Platforms:** Active accounts on Snowflake and DBT Cloud for data management, storage, and transformation

Methodology

The proposed methodology follows a series of sequential steps organized into three main stages: **Cloud-Based Data Implementation and Transformation**, **Data Analysis and Predictive Modeling**, and **Visualization and Communication of Findings**. Each of these stages is broken down into specific subprocesses that enable the project to be structured and developed in an orderly manner, from data management to the presentation of results, as shown in **Figure 1**.

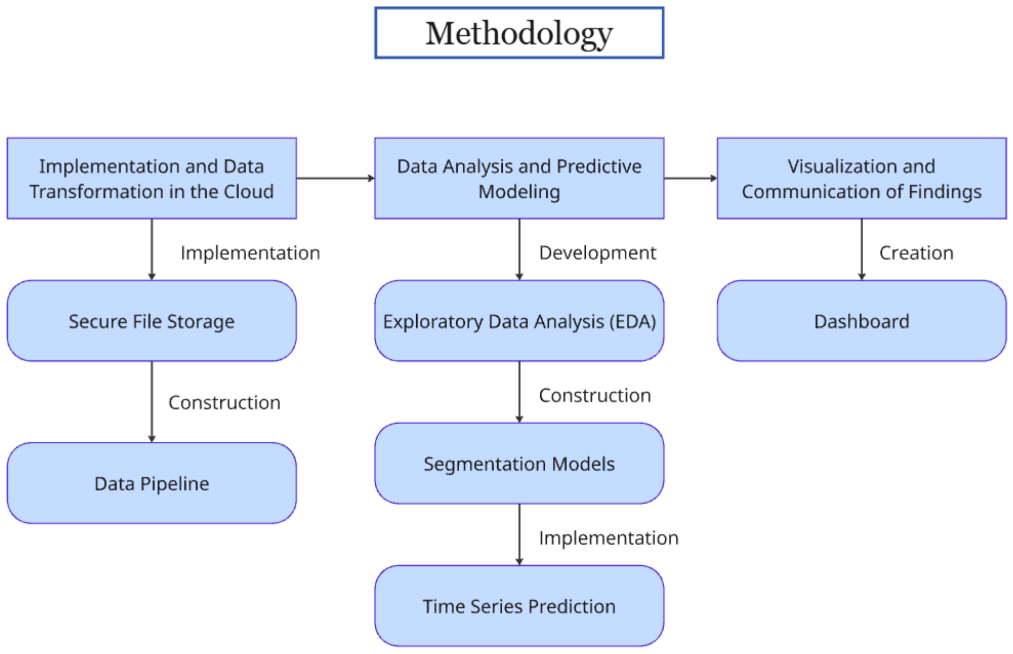

Figura 1. Diagrama de Metodología

Cloud-Based Data Implementation and Transformation

This project focused on transforming raw data from the GH Archive (January 9–11, 2015) into analysis-ready tables in the cloud. Python scripts were used to automate the download and initial storage of data in Google Cloud Storage (GCS). To overcome Snowflake's limitation in accessing external URLs directly, a Python script was implemented to download the files locally and then upload them to GCS. Finally, within Snowflake, Python scripts were used to clean, normalize, and generate new analytical variables, resulting in structured tables ready for exploration and visualization.

Following the transformation in Snowflake, the tables **ACTOR, GITHUB_EVENTS_WIDE, GITHUB_RAW_DATA, GLOBAL_EVENTS, ORG,** and **REPO** were created, organizing and structuring GH Archive data for analysis. We will explore these tables in greater detail below.

Tabla 1 (ACTOR). Contains basic information about GitHub users who perform actions.

Field	Description	Data type
avatar_url	URL of the profile image (avatar) of the GitHub user who triggered the event.	Text(String)
gravatar_id	Unique identifier associated with the GitHub user's Gravatar account.	Text(String)
id	Unique numerical identifier assigned to each user within the GitHub platform.	Number
login	Unique username that identifies the user on GitHub.	Text(String)
url	Web address (URL) of the GitHub user's public profile.	Text(String)

Tabla 2 (GITHUB_EVENTS_WIDE). Contains detailed information about various events that occur on the GitHub platform.

Field	Description	Data type
id	Unique identifier for each	Number

	event recorded in the GitHub log.	
issue_closed_at	Timestamp indicating when an issue was closed in a GitHub repository.	Timestamp_NTZ
issue_url	Specific URL of the issue associated with the event.	Text(String)
language	Primary programming language associated with the repository where the event occurred.	Text(String)
merge_at	Timestamp indicating when a pull request was merged into the repository.	Timestamp_NTZ
payload_created_at	Timestamp indicating when the event was created on GitHub.	Timestamp_NTZ
release_published_at	Timestamp indicating when a new release of the project was published in the repository.	Timestamp_NTZ
review_submitted_at	Timestamp indicating when a code review was submitted for a pull request.	Timestamp_NTZ

Tabla 3 (GITHUB_RAW_DATA). Contains the raw GitHub event data in its original JSON format within the "VARIANT_COL" column.

Field	Description	Data type
vatiant_col	To store semi-structured data, such as JSON.	Variant

Tabla 4 (GLOBAL_EVENTS). It provides an overview of the events that occur on GitHub, including information about who performed the action, when it occurred, where it occurred

(repository and organization), whether it was public, and the type of event.

Field	Description	Data type
actor_id	Unique numerical identifier of the actor or user who performed the event.	Number
created_at	Timestamp indicating when the event occurred on GitHub.	Timestamp_NTZ
id	Unique identifier for each globally recorded event.	Number
org_id	Unique numerical identifier of the organization to which the repository where the event occurred belongs.	Number
public	Indicates whether the event was public or private.	Boolean
repo_id	Unique numerical identifier of the repository where the event occurred.	Number
type	Type of event that occurred on GitHub.	Text(String)

Tabla 5 (ORG). Stores basic identification and profile information of organizations with recorded activity in the GH Archive.

Field	Description	Data type
avatar_url	URL of the profile image (avatar) of the organization on GitHub.	Text(String)
gravatar_id	Unique identifier associated with the organization's Gravatar account.	Text(String)

id	Unique numerical identifier assigned to each organization within the GitHub platform.	Number
login	Unique username that identifies the organization on GitHub.	Text(String)
url	Web address (URL) of the organization's public profile on GitHub.	Text(String)

Tabla 6 (REPO). Stores basic identification information of repositories with recorded activity in the GH Archive.

Field	Description	Data type
id	Unique numerical identifier assigned to each repository within the GitHub platform.	Number
name	Name of the repository.	Text(String)
url	Web address (URL) of the repository on GitHub.	Text(String)

Data Analysis and Predictive Modeling

To address the data analysis task, the process began with an exploratory data analysis (EDA) phase, which enabled a comprehensive understanding of the data structure, distribution, and relevant behavioral patterns. Based on this exploration, segmentation models were built to identify groups with similar characteristics and potential anomalies. Finally, time series prediction models were implemented to anticipate trends and support strategic, data-driven decision-making.

Implementation

The project implementation can be found at the following URL: https://github.com/ValenCedano/Wizeline-Hackathon-2025

The repository is organized into three top-level entries:

1 . code/snowflake/

Contains all Snowflake-related data-engineering scripts and jobs:

- ❖ gcp_stage_bucket_integration.sql – defines the external stage and bucket integration.
- ❖ gcp_load_raw_data.sql – loads raw event data from GCS into Snowflake tables.
- ❖ sql_create_replace_global_tables.sql – (re)creates the global ("raw" → "staging" → "production") tables.
- ❖ sql_populate_global_tables.sql – transforms and ingests staging data into production tables.
- ❖ sql_create_rule_external_connection.sql – sets up network access rules / external connections.
- ❖ sql_populate_events_wide_view.sql – builds a wide, denormalized view for downstream analytics.
- ❖ sql_remove_all_data.sql – cleans out all data objects (for full re-load).
- ❖ py_transformation_load.py – Python orchestration script that invokes the above SQL steps in order (with any necessary parameterization or logging).

2. visualization/

Houses all BI artifacts and reference images:

- ❖ GH Archive analysis.pbix – Power BI report file connecting to the Snowflake "wide view" and delivering interactive dashboards.
- ❖ medallion_architecture.jpeg – a diagram of the medallion (bronze–silver–gold) data-lake architecture.

- metodologia.jpeg – a flowchart depicting the overall data-ingestion and transformation methodology.

3. README.md