Available online www.ejaet.com

European Journal of Advances in Engineering and Technology, 2024, 11(1): 52-55

Research Article ISSN: 2394-658X

Automating BigQuery Dependency Management with Email Alerts: The BigQuery_dependency_email_trigger Library

Preyaa Atri

Email id – Preyaa.atri91@gmail.com

ABSTRACT

This paper introduces BigQuery_dependency_email_trigger, a Python library designed to automate dependency checks for BigQuery datasets and tables. It monitors the results of user-specified BigQuery queries and sends notification emails based on pre-defined conditions. This approach streamlines data pipeline monitoring by automating dependency checks and alerting users of potential issues. The library offers configurable retry attempts, warning thresholds, and email content, allowing for flexible integration into existing data workflows.

Key words: BigQuery, Data Pipelines, Dependency Management, Monitoring, Email Alerts, Python Library

INTRODUCTION

Data pipelines are the backbone of modern data infrastructures, orchestrating data movement and transformation processes (Khan et al., 2023). Ensuring data dependencies are met is crucial for data pipeline integrity. Traditional methods for dependency check often involve manual intervention or custom scripts, leading to inefficiencies and potential human errors.

BigQuery_dependency_email_trigger addresses this challenge by providing an automated solution for monitoring BigQuery dependency queries. It simplifies dependency management by offering the following functionalities:

A. **Automated Dependency Checks:** Executes user-defined SQL queries against BigQuery to check for specific data conditions.
B. **Configurable Retries:** Allows users to specify the number of retry attempts for failed dependency checks, mitigating transient errors.
C. **Email Alerts:** Sends informative emails based on the outcome of the dependency check. Users can configure warning and error emails with customizable content and subject lines.

By automating dependency checks and providing timely notifications, BigQuery_dependency_email_trigger empowers data engineers and analysts to proactively identify and address potential issues within data pipelines.

PROBLEM STATEMENT

Data pipelines often rely on the successful completion of upstream tasks to generate downstream results. Manually monitoring these dependencies can be time-consuming and error-prone. Existing solutions might involve custom scripting or third-party tools that lack flexibility or require significant development effort.

SOLUTION

BigQuery_dependency_email_trigger offers an automated approach to managing BigQuery data dependencies. The core functionality revolves around the bigquery_dependency_email_trigger function, which accepts a comprehensive set of parameters:

A. **dependency_query:** The SQL query to execute against BigQuery to check the dependency condition.
B. **project:** The GCP project ID containing the BigQuery dataset and table referenced in the query.
C. **expected_dependent_result:** The anticipated outcome of the dependency query.
D. **number_of_tries:** The total number of retries to attempt for failed dependency checks.
E. **num_of_tries_before_warn_email:** The number of failed attempts before triggering a warning email notification.
F. **time_interval:** The time interval (in seconds) to wait between retry attempts.

G. **warn_email_content/subject:** Content and subject line for the warning email notification.
H. **error_email_content/subject:** Content and subject line for the error email notification.
I. **SMTP server details:** Configuration details for the SMTP server used for sending emails (server address, port, sender email, username, and password).

The function executes the dependency_query and compares the result to the expected_dependent_result. If a match is not found within the specified number_of_tries, the library implements a retry mechanism with a configurable time_interval between attempts. Depending on the number of failed attempts relative to the num_of_tries_before_warn_email threshold, the library triggers either a warning or error email notification.

INSTALLATION

BigQuery_dependency_email_trigger can be easily installed using pip:

```
pip install bigquery-dependency-email-trigger
```

A. Usage with Example

BigQuery_dependency_email_trigger simplifies dependency checks by offering a single function, bigquery_dependency_email_trigger. Let's walk through a detailed example demonstrating its practical application:

Scenario: We have a data pipeline where a web scraping job populates a BigQuery table named product_data daily. Downstream processes rely on this table being populated with fresh data. BigQuery_dependency_email_trigger can be used to automate a dependency check and send email alerts if the data isn't loaded as expected.

Python

```
import time
import pandas_gbq
from bigquery_dependency_email_trigger import
bigquery_dependency_email_trigger

# Define parameters
dependency_query = "SELECT COUNT(*) FROM `your-project-
id.marketing_data.product_data` WHERE load_date = CURRENT_DATE()"
project = "your-gcp-project-id"
expected_dependent_result = 1000  # Assuming we expect at least 1000 new
products daily
number_of_tries = 3
num_of_tries_before_warn_email = 2
time_interval = 3600  # Check every hour

# Configure email notification (replace with your details)
warn_email_content = "Attention! The product data table might not be loaded
for today."
warn_email_subject = "Warning: Dependency Check for Daily Product Data Load"
error_email_content = "Urgent! Dependency check failed. Product data load
might be broken."
error_email_subject = "Error: Daily Product Data Load Failure"

# SMTP server details (replace with your configuration)
SMTP_SERVER = "smtp.example.com"
SMTP_PORT = 587
SENDER_EMAIL = "data_monitoring@yourcompany.com"
SMTP_USER = "your_smtp_username"  # Optional, if authentication required
SMTP_PASSWORD = "your_smtp_password"  # Optional, if authentication required

# Trigger the dependency check
result = bigquery_dependency_email_trigger(
    dependency_query, project, expected_dependent_result,
    number_of_tries, num_of_tries_before_warn_email, time_interval,
    warn_email_content, warn_email_subject, error_email_content,
error_email_subject,
    SMTP_SERVER, SMTP_PORT, SENDER_EMAIL, SMTP_USER, SMTP_PASSWORD
)

if result:
  print("Product data for today seems to be loaded successfully.")
else:
  print("Dependency check failed. Please investigate the product data load
job.")
```

B. Explanation:

[1]. We define the dependency_query to check how many rows were loaded into the product_data table with today's date (CURRENT_DATE() function).
[2]. Other parameters like project, expected_dependent_result, retry configuration, and email notification details are set.
[3]. The bigquery_dependency_email_trigger function executes the check.
[4]. The script prints a message depending on whether the dependency was met or not.

This example demonstrates how BigQuery_dependency_email_trigger can be seamlessly integrated into existing Python scripts to automate dependency checks and send informative email notifications. By customizing the parameters and queries, you can tailor the solution to fit various data pipeline monitoring scenarios within your BigQuery environment.

IMPACT

By automating dependency checks and generating email alerts, BigQuery_dependency_email_trigger offers several advantages:

[1]. Improved Efficiency: Reduces manual effort associated with dependency monitoring, freeing up valuable time for data engineers.
[2]. Proactive Issue Detection: Enables faster identification and resolution of data pipeline issues by providing timely notifications.
[3]. Enhanced Data Quality: Contributes to maintaining data quality by alerting users to potential discrepancies in dependent datasets.
[4]. Simplified Workflows: Integrates seamlessly into existing data pipelines, streamlining overall data management processes.

BigQuery_dependency_email_trigger extends its positive influence beyond just expediting data pipeline monitoring. By automating dependency checks and generating email alerts, it creates a ripple effect that impacts various adjacent fields within the data ecosystem:

[1]. Data Engineering Efficiency: Automating dependency checks liberates data engineers from repetitive manual tasks and process dependency (Schreiber, 2020). This reclaimed time allows them to focus on higher-value activities such as data pipeline optimization, data quality improvement, and building new data processing tools.
[2]. Data Quality Management: Timely alerts about unmet dependencies enable data engineers to proactively address potential issues before they impact downstream data products. This proactive approach helps to ensure data accuracy and consistency throughout the data pipeline.
[3]. Alert Fatigue Reduction: Traditional monitoring approaches often generate a high volume of alerts, leading to alert fatigue among data engineers. BigQuery_dependency_email_trigger focuses on critical dependency checks, delivering only relevant notifications. This targeted approach reduces alert fatigue and ensures that data engineers prioritize the most important issues.
[4]. Improved Collaboration: Email alerts can be directed to various stakeholders within the data team, fostering improved communication and collaboration. Data analysts, data scientists, and data owners can be made aware of potential issues impacting their work, enabling a more coordinated response (Mastrianni et al., 2022).
[5]. Enhanced Data Governance: By establishing clear data dependencies and automating checks, BigQuery_dependency_email_trigger strengthens data governance practices. It provides an auditable trail of dependency checks and facilitates the enforcement of data quality standards within the organization.
[6]. Cost Optimization: Proactive identification and resolution of data pipeline issues through dependency checks can minimize the reprocessing of erroneous data. This translates to cost savings by optimizing resource utilization in cloud environments (Al-Arasi & Saif, 2020) like BigQuery.

This library's impact extends far beyond automating dependency checks. It fosters a data ecosystem that is efficient, proactive, collaborative, and governed, ultimately leading to higher quality data and better overall data management practices.

DEPENDENCIES

BigQuery_dependency_email_trigger relies on several external libraries to function effectively:

[1]. **pandas-gbq:** This library facilitates interacting with BigQuery from Python, enabling the execution of the dependency_query and retrieval of results.
[2]. **smtplib:** Python's built-in smtplib module handles sending email notifications via the specified SMTP server.

FUTURE SCOPE AND CONSIDERATIONS

While BigQuery_dependency_email_trigger offers a valuable solution for BigQuery dependency management, there's room for future enhancements:

[1]. **Supporting External Data Sources:** Extending functionality to monitor dependencies on external data sources beyond BigQuery would broaden its applicability.

[2]. **Advanced Email Configuration:** Features like recipient lists, email attachments, and integration with external notification services could improve user experience.

[3]. **Data Quality Checks:** Expanding the library to support more complex data quality checks within the dependency_query would enable a wider range of monitoring scenarios.

[4]. **Error Handling and Logging:** Implementing robust error handling mechanisms and detailed logging capabilities would improve troubleshooting and debugging.

In conclusion, BigQuery_dependency_email_trigger is a valuable Python library that automates dependency checks for BigQuery datasets and tables. By streamlining dependency management and providing email notifications, it empowers data engineers and analysts to proactively maintain data pipeline health and data quality. Future development efforts focused on the areas mentioned above can further enhance the library's capabilities and broaden its adoption within the data engineering community.

REFERENCES

[1]. Khan, N. Nikolov, M. Matskin, R. Prodan, D. Roman, B. Şahinet al., "Smart data placement using storage-as-a-service model for big data pipelines", Sensors, vol. 23, no. 2, p. 564, 2023. https://doi.org/10.3390/s23020564

[2]. Python Software Foundation, "smtplib — SMTP protocol client," Python documentation, Available: https://docs.python.org/3/library/smtplib.html.

[3]. Google, "Introduction to pandas-gbq," Google Cloud Python Client Library Documentation, Available: https://googleapis.dev/python/pandas-gbq/latest/intro.html.

[4]. C. Schreiber, "Automated sustainability compliance checking using process mining and formal logic",, 2020. https://doi.org/10.1145/3401335.3401355

[5]. A. Mastrianni, L. Almengor, & A. Sarcevic, "Alerts as coordination mechanisms", Proceedings of the ACM on Human-Computer Interaction, vol. 6, no. GROUP, p. 1-14, 2022. https://doi.org/10.1145/3492828

[6]. R. Al-Arasi and A. Saif, "Task scheduling in cloud computing based on metaheuristic techniques: a review paper", EAI Endorsed Transactions on Cloud Systems, vol. 6, no. 17, p. 162829, 2020. https://doi.org/10.4108/eai.13-7-2018.162829