Author

[Email address]

Abstract

[Draw your reader in with an engaging abstract. It is typically a short summary of the document.   
When you’re ready to add your content, just click here and start typing.]

Data Engineering Project

Designing and Implementing a Data Pipeline to Analyse Scientific Publications

Contents

[Abstract 2](#_Toc154396361)

[Introduction 2](#_Toc154396362)

[Description 2](#_Toc154396363)

[Graph Database structure: 3](#_Toc154396364)

[Pre-processing Strategies 3](#_Toc154396365)

[Handling Missing Values 3](#_Toc154396366)

[Data Type Standardisation 4](#_Toc154396367)

[Enrichment through External Sources 4](#_Toc154396368)

[Pipeline for Incremental Updates 4](#_Toc154396369)

[Data Modelling 4](#_Toc154396370)

[Data Warehouse schema: 4](#_Toc154396371)

[Analysis 5](#_Toc154396372)

[Business Intelligence Queries and Graph Analytics 5](#_Toc154396373)

[Graph Analytics: 7](#_Toc154396374)

[Performance Considerations and Scalability 9](#_Toc154396375)

[Conclusion 9](#_Toc154396376)

[Individual Contribution Statement 9](#_Toc154396377)

[Appendix 9](#_Toc154396378)

# Abstract

This project aims to offer a hands-on immersion into the realm of data engineering, modelling, and analytics, specifically focusing on the challenges posed by real-world scientific publication data. Its primary goal is to develop proficiency in constructing comprehensive analytical structures and ensuring the repeatability of the data pipeline. Throughout this endeavour, we'll explore a wide spectrum of analytical viewpoints, ranging from establishing data warehouses and data marts for Business Intelligence (BI) inquiries to utilising graph databases for tasks such as predicting co-authorship and conducting community analysis. Furthermore, we'll delve into the intricate processes of data transformations, cleansing, and augmentation, equipping participants to navigate the intricacies of genuine data engineering projects.

# Introduction

In today's data-driven world, managing and utilizing diverse datasets is crucial for making informed decisions and boosting operational efficiency. This project dives deep into handling data, using modern tools and methods to streamline the processing of incoming JSON files.

The main objective here is to unlock the potential of the received JSON data. We're extracting relevant information and shaping it into structured repositories ready for analysis and exploration. This involves careful analysis, building databases, and setting up automated pipelines for smooth data flow.

By bringing together Docker, pgAdmin, Neo4j, and Airflow, this project aims to create a unified system. It not only handles and refines raw JSON data but also constructs databases and orchestrates workflows with exceptional efficiency.

This project follows a meticulously planned sequence: reducing data, analyzing it, setting up databases, and automating pipelines. It's a blueprint showcasing the power of modern data engineering tools. These tools work together to handle and transform data efficiently while ensuring scalability, reliability, and maintainability in a constantly changing data landscape.

This introduction marks the beginning of an exploration into data engineering, where advanced technologies form the foundation for smooth data processing and informed decision-making.

# Description

To understand existing dataset, there was done Exploratory Data Analysis (EDA) summary.

**Dataset Overview**

**Records:** The dataset contain 10,00 records.

**Columns:** The dataset includes various column such as **id**, **submitter**, **authors**, **title**, **comments**, **journal-ref**, **doi**, **report-no**, **categories**, **license**, **abstract**, **versions**, **update\_date**, and **authors\_parsed**.

**Data Inspection**

**Data Types:** Mostly categorical/object data types.

**Missing Values:** several columns have missing values: **comments**, **journal-ref**, **doi**, **report-no**, and **license** significant number of missing values.

**Duplicate Records:** no explicit identification of duplicate records.

**Value Counts (Top records)**

**id**: All unique identifiers, each occurring once.

**submitter:** Diverse submitters with varying submission frequencies.

**authors:** Several authors contributing across multiple papers.

**title:** Unique titles, some with repetitions suggesting multiple submissions.

**comments, journal-ref, doi, report-no:** Various descriptive entries with differing frequencies.

**categories:** Diverse categories with astro-ph being the most frequent.

**license:** Multiple licenses used; one dominant license used in the majority.

**Insights**

**Peculiar Entries:** Some comments and abstract entries contain significant text, potentially indicating special cases or detailed reports.

**Missing Values:** Several columns have a considerable number of missing values, requiring further investigation, or handling during preprocessing.

Also, further exploration of dataset was done by using various columns like abstracts, version information, update dates and author details across different categories like astro-ph, hep-ph,quant-ph etc.

Graph Database structure: description of the nodes and relationships chosen for the graph view, depicting entities like authors, papers, journals etc.

The ArXiv dataset comprises various fields such as id, submitter, authors, title, abstract, among others. Upon initial inspection of a subset of the data, it was observed that several columns contained missing values. For instance:

comments: 36480 missing entries

journal-ref: 126845 missing entries

doi: 98608 missing entries

These missing values pose a challenge in ensuring data completeness and reliability. Additionally, there were diverse data types across columns, requiring standardisation for seamless integration into the DW and Graph Database.

## Pre-processing Strategies

### Handling Missing Values

To address missing data, multiple strategies were employed:

Imputation: For certain columns like comments, journal-ref, and doi, missing values were filled using appropriate imputation techniques such as mean, median, or mode based on the data distribution.

Dropping Columns: Columns with an overwhelming number of missing entries, such as report-no with 228332 missing values, were considered for removal to maintain data quality.

### Data Type Standardisation

Uniformity in data types across columns is crucial for consistency and query efficiency. For instance, converting update date from an object type to a datetime format facilitates chronological analysis and time-based queries.

### Enrichment through External Sources

Leveraging the Google Scholar API, additional information like publication counts, citation metrics, and author profiles were integrated to augment the dataset. This enriched data will significantly enhance the DW’s ability to provide comprehensive insights and facilitate more nuanced BI queries.

### Pipeline for Incremental Updates

To simulate incremental updates to the DW and Graph Database, the dataset was divided into partitions. Each partition was sequentially fed into the pipeline, mimicking a real-time or periodic update scenario. This iterative approach ensures scalability and efficiency in handling large volumes of data.

## Data Modelling

### Data Warehouse schema:

There is a fact table (ArXiv data) & dimension tables.

**Dimension Tables:**

dim\_submitter: Stores details about the submitter of the publication.

dim\_authors: Contains information about authors, including their names and genders.

dim\_title: Holds data regarding publication titles, both original and refined versions.

dim\_journal\_ref: Stores references to journals, with resolved and refined venue information.

dim\_license: Stores types of licenses under which publications are distributed.

dim\_field\_of\_study: Contains fields of study associated with publications.

**Fact Table:** fact\_publications

**Junction Table:**

publication\_author\_junction: Handles the many-to-many relationship between publications and authors by linking publication IDs with author IDs.

This schema is designed to efficiently store and manage data related to publications, their authors, associated metadata, and various attributes like journals, licenses, and fields of study. The fact table acts as a centerpiece aggregating key information about publications, while the dimension tables provide additional context and details for analysis and reporting purposes.

A diagram of a database

Description automatically generated

Figure : Star schema in data warehousing.

# Analysis

## Business Intelligence Queries and Graph Analytics

In leveraging the enriched dataset, a set of specific Business Intelligence (BI) queries has been meticulously formulated to illuminate the multifaceted capabilities inherent within the Data Warehouse (DWH) schema. These queries aim to delve deeply into the vast landscape of scientific publication data, offering nuanced insights and strategic information retrieval.

**Detailed Publication Information**: Uncover comprehensive details for each publication, including an exhaustive list of authors.

**Yearly Publication Count:** Analyze the annual count of publications to discern temporal trends in research output.

**Top 5 Publishers:** Identify the top five most prolific publishers (submitters) based on their contributions to the dataset.

**High Energy Physics Publications:** Isolate publications specifically related to the field of 'High Energy Physics' for targeted analysis.

**Prolific Authors:** Identify authors with the highest publication counts, highlighting their substantial contributions.

**Publications Without Specified Licence:** Quantify the number of publications lacking a specified licence for legal usage or distribution.

**Journal Publication Counts:** Summarise the count of publications associated with each distinct journal.

**Licence Type Distribution**: Categorize publications based on licence types and ascertain the distribution within the dataset.

**Top Authors in Each Field**: Identify and rank top authors within various fields of study for individualised recognition.

**Field-Specific Publication Trends:** Analyse changes in publication counts across diverse fields of study over successive years.

... and more such queries that span a wide spectrum of analytical dimensions, aiming to extract valuable insights from the dataset.

These BI queries are strategically designed to mine comprehensive intelligence from the dataset, enabling nuanced analysis and strategic decision-making in the domain of scientific publications. They showcase the extensive capabilities of the Data Warehouse schema in addressing diverse analytical needs and unveiling actionable insights within this expansive repository of scientific knowledge.

**BI Queries**: Below are some highlighted examples of SQL queries that were used:

A white background with red text

Description automatically generated

Figure : BI query example 1.

A white background with text

Description automatically generated

Figure : BI query example 2.

A computer screen shot of a computer code

Description automatically generated

Figure : BI query example 3.

A computer screen shot of a computer

Description automatically generated

Figure : BI query example 4.

Graph Analytics:  in this project harnessing the power of Neo4j graph analytics to explore relationships amount authors, publications, categories, journals, versions, dates, and licenses. This approach leverages the interconnected nature of scholarly work, enabling insightful queries and analyses.

#### Graph structure

The nodes encompass various entities like authors, publications, categories, journals, versions, dates, and licenses. Each entity is interconnected through specific relationships:

* **AUTHORED\_BY:** Connects authors to publications, detailing author contributions.
* **PUBLISHED\_IN:** Links publications to journals where they are published.
* **HAS\_CATEGORY:** Associates publications with their respective categories.
* **HAS\_VERSION:** Tracks different versions of publications.
* **UPDATED\_ON:** Relates publications to their update dates.
* **LICENSED\_UNDER:** Indicates the license under which a publication is distributed.
* **CO\_AUTHORED:** Establishes collaborations between authors who have co-authored papers.

#### Graph Queries:

The queries designed for this project offer versatile insights into the graph data:

* **Author's Publications:** Fetches publications authored by any author.
* **Co-Authors Identification:** Discovers pairs of authors who have collaborated on publications.
* **Category-wise Publications:** Lists publications associated with different categories and journals.
* **License-based Publications:** Retrieves publications and their associated licenses.
* **PageRank for Influential Papers:** Applies the PageRank algorithm to identify influential publications based on their citations and connections within the graph.
* **Database Setup using Docker:**
* Utilize Docker to set up environments for various databases:
* PostgreSQL (via pgAdmin):
* Create a PostgreSQL database using pgAdmin to store structured data.
* Define schemas, tables, and relationships as per the previously defined schema.
* Neo4j:
* Implement a graph database using Neo4j to manage and query interconnected data.
* Define nodes and relationships based on the data's graph structure.
* Pipeline Creation with Airflow:
* Implement Airflow to automate data processing tasks:
* Data Ingestion:
* Schedule tasks to fetch incoming JSON files and trigger the data reduction process.
* Data Transformation:
* Develop tasks to transform the reduced data into formats suitable for storage in PostgreSQL and Neo4j databases.
* Database Population:
* Automate the population of the PostgreSQL and Neo4j databases with the transformed data.
* Data Validation & Quality Checks:
* Incorporate tasks to ensure data integrity, perform validation, and conduct quality checks during the pipeline.
* Monitoring & Maintenance:
* Set up monitoring mechanisms within Airflow to track the pipeline's performance, detect errors, and ensure timely data processing.
* Establish a maintenance schedule for databases, including backups, updates, and optimizations.
* Key Tools Used:
* JSON Handling: Python libraries like json for parsing and reducing JSON files.
* Databases: PostgreSQL managed through pgAdmin for structured data, Neo4j for graph data.
* Containerization: Docker for creating and managing database environments.
* Workflow Management: Apache Airflow for orchestrating and automating data pipelines.

## Performance Considerations and Scalability

Scalability Analysis: insights into how the project would scale with increased dataset size, optimizations applied, and potential challenges.

# Conclusion

This project is a testament to the power of modern tools in handling incoming JSON data. We've explored data ingestion, reduction, and analysis, uncovering valuable insights hidden in the raw information. By using Docker, pgAdmin, and Neo4j, we've carefully built structured databases that form a strong base for smart decision-making.

With the addition of Airflow, our processes have become more automated. This means our pipelines manage data transformation and database population accurately and reliably. Automation boosts efficiency and ensures the processed data's quality and integrity.

As this project concludes, these tools combine to create a scalable, reliable framework for data engineering. The documentation we've created serves as both a knowledge hub and a guide for future data initiatives.

This project truly represents modern data engineering—a blend of advanced tools and smart strategies. It lays the groundwork for a data-focused future where information fuels informed decisions and meaningful actions. It's a robust foundation for navigating the evolving data landscape and tapping into its potential in various forms.

# Individual Contribution Statement

# Appendix