**YELP REVIEW ANALYSIS**

**A report on**

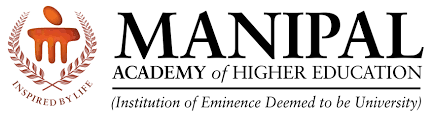
**Big Data Analysis Lab Project**

**[CSE-3263]**

Submitted By

**Aditya Nerusu – 210962196**

**Sai Sujan Korrapati - 210962194**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**MANIPAL INSTITUTE OF TECHNOLOGY,**

**MANIPAL ACADEMY OF HIGHER EDUCATION**

**April 2024**

Sai Sujan Korrapati (210962194)

Aditya Nerusu (210962196)

Department of Computer Science and Engineering,

Manipal Institute of Technology

Manipal Academy of Higher Education, India

[korrapatisaisujan@gmail.com](mailto:korrapatisaisujan@gmail.com)

adityaadi2744@gmail.com

**Abstract-**The goal of this project is to demostrate the use of PySpark and Spark SQL to query and analyze the Yelp Open Dataset. Specifically, the aim is to analyze the Yelp Reviews dataset, which consists of 6.7 million user-generated reviews of businesses on Yelp. I also perform JOIN operations with the Yelp Business and Yelp User datasets to describe relations between review ratings and characteristics of the business, such as geographic location. To perform some of these queries, I demonstrate the use of user-defined functions (UDFs) in Spark SQL queries. Lastly, I briefly examine how partitioning of the underlying data abstraction changes computational speed.

**Keywords-** PySpark, Spark SQL, Yelp Open Dataset, user-generated reviews, JOIN operations, user-defined functions (UDFs), partitioning, computational speed.

## **I.Introduction**

### **Apache Spark**

Apache Spark is a unified analytics engine for large-scale data processing. Released in 2010, in enables cluster-computing and can be used interactively from Java, Scala, Python, R, and SQL shells. It also provides access to a number of libraries, including SQL and DataFrames, MLlib for machine learning, GraphX and Spark Streaming . Users can access data from a large range of data sources, including HDFS and Apache Hive, and can run Spark on cluster managers like Mesos and Hadoop YARN or in Spark's standalone cluster mode.

#### **Resilient Distributed Datasets (RDDs)**

The fundamental data abstraction in Spark is the Resilient Distributed Dataset (RDD). RDDs are immutable collections of Java or Python objects, partitioned across a cluster and able to be operated on in parallel . A slightly more complete and technical definition is provided in the original paper that introduced RDDs

We present Resilient Distributed Datasets (RDDs), a distributed memory abstraction that lets programmers perform in-memory computations on large clusters in a fault-tolerant manner.

### **Spark SQL**

Released in 2014, Spark SQL is a Spark module for structured data processing. Since it provides Spark with more information about the structure of the data and the computation being executed, Spark SQL can conduct more optimizations than the basic Spark RDD API .

The creators of Spark SQL point out that it was designed to bridge the gap between relational and procedural systems, allowing users to easily combine the two . It does so via two contributions: DataFrame API and an extensible optimizer called Catalyst. These innovations will be covered in the next section.

#### **DataFrames and DataFrame API**

­­

#### **Programming interface**

The interfaces to Spark SQL are summarized in Figure 1. Spark SQL is a library that runs on top of Spark. The SQL inferfaces can be accessed using Open Database Connectivity (ODBC) or Java Database Connectivity (JDBC), or alternatively through a command-line console. Both Spark SQL and DataFrame API queries pass through the the Catalyst relational optimizer, whereas procedural code written directly in supported programming languages (Java, Scala and Python) does not. At the very bottom, below Spark, lie RDDs—Spark's fundamental data abstractions that were covered above.

Figure 1: Interfaces to Spark SQL and interaction with A diagram of a software development

Description automatically generated

## **The Yelp Open Dataset**

The [Yelp Open Dataset](https://www.yelp.com/dataset) is a subset of internal data collected by Yelp, a crowd-sourced review forum and business directory service. Yelp has provided this data free of charge for personal, academic and educational purposes.

The dataset is composed of six individual JSON files with cross-referenced variables that allow users to combine multiple files. The core of the dataset and the focus of this project is the **reviews** table, which contains information on roughly 6.7 million user-written reviews of businesses on Yelp.

**II.Literature Review:**

The analysis of user-generated content, such as online reviews, has gained significant attention in the research community due to its importance in understanding consumer behavior, sentiment analysis, and business intelligence. In the context of analyzing Yelp reviews using PySpark and Spark SQL, several relevant studies and methodologies can be reviewed:

**Text Mining and Sentiment Analysis:** Numerous studies have explored techniques for text mining and sentiment analysis of online reviews. Research by Liu (2012) provides an overview of sentiment analysis methods, including lexicon-based approaches, machine learning algorithms, and hybrid models. Applying these techniques to Yelp reviews can facilitate the extraction of sentiment polarity and sentiment trends over time.

**Big Data Analytics:** With the advent of big data technologies like Apache Spark, researchers have investigated the application of distributed computing frameworks for analyzing large-scale datasets, including online reviews. Studies by Zaharia et al. (2016) and Armbrust et al. (2015) discuss the architecture and capabilities of Spark for processing big data efficiently. Leveraging PySpark and Spark SQL for Yelp review analysis aligns with the trend of utilizing distributed computing for scalable data analytics.

**User Behavior Analysis**: Understanding user behavior on review platforms like Yelp is crucial for businesses and researchers. Studies by Chevalier and Mayzlin (2006) and Luca (2016) examine factors influencing user behavior, such as review characteristics, reviewer demographics, and social influence. Analyzing Yelp reviews with PySpark and Spark SQL can provide insights into user engagement patterns, review trends, and the impact of user-generated content on businesses.

**Business Intelligence and Decision Making:** Yelp reviews offer valuable insights for businesses in various industries, aiding in decision-making processes, reputation management, and competitive analysis. Research by Duan et al. (2008) and Ghose and Ipeirotis (2011) explores the use of online reviews for business intelligence and marketing strategies. Analyzing Yelp data with PySpark enables businesses to extract actionable insights from large volumes of user-generated content.

**Natural Language Processing (NLP):** NLP techniques play a crucial role in extracting meaningful information from text data, including online reviews. Studies by Manning et al. (2020) and Jurafsky and Martin (2019) provide comprehensive coverage of NLP fundamentals and advanced techniques, such as sentiment analysis, topic modeling, and named entity recognition. Integrating NLP libraries with PySpark enables the application of sophisticated text analysis algorithms to Yelp reviews.

**Optimization and Performance Tuning:** As data volumes continue to grow, optimizing query performance becomes essential for efficient data analysis. Research by Zhou et al. (2018) and Meng et al. (2016) discusses optimization techniques for distributed computing platforms like Spark, including query optimization, partitioning strategies, and resource management. Understanding these optimization principles can enhance the efficiency of Yelp review analysis using PySpark and Spark SQL.

**III.Research Gaps:**

**Nuanced Sentiment Analysis:** Develop more nuanced sentiment analysis techniques tailored to the diverse content and subjective expressions found in Yelp reviews

**Temporal Dynamics:** Investigate how sentiment, review volume, and user behavior vary over different time periods, such as weekdays versus weekends, seasons, and significant events like holidays.

**User Behavior Modeling:** Develop predictive models to understand user engagement patterns, review submission frequency, and factors influencing user interactions with Yelp's platform.

**Business Impact Analysis:** Examine causal relationships between review content, ratings, and business outcomes, considering factors like consumer decisions, business reputation, and financial performance.

## **Multimodal Data Integration**: Explore techniques for integrating multimodal data sources (e.g., images, videos) into Yelp review analysis, examining their impact on consumer perceptions and decision-making.

## **Privacy and Ethical Considerations:** Address privacy and ethical concerns associated with analyzing user-generated content, focusing on methods for data anonymization and ethical guidelines for research on online review

## **IV.Methodology:**

## **Data Reading and Preprocessing:**

## Utilize PySpark's spark.read.json(path) method to read the Yelp Open Dataset JSON files into DataFrames.

## Conduct basic checks to ensure data integrity and completeness.

## Infer the schema of the dataset using printSchema() method to understand the structure of the data.

## **Querying the Data:**

## Perform basic SQL queries to count the number of reviews, unique businesses, and unique users.

## Utilize Spark SQL's DataFrame API to compute descriptive statistics for variables such as star ratings and review counts.

## **User-Defined Functions (UDFs):**

## Define user-defined functions (UDFs) to extract specific information from textual data, such as dates.

## Implement UDFs to query textual patterns in reviews and extract relevant insights.

## **JOIN Queries with Additional Yelp Datasets:**

## Join the Yelp Reviews dataset with the Yelp Business and Yelp User datasets to analyze relationships between review ratings and business/user characteristics.

## Utilize SQL JOIN operations to merge datasets based on common keys, such as business\_id and user\_id.

## **Partitioning Evaluation:**

## Investigate the impact of partitioning on computational speed by analyzing the number of partitions in the underlying RDDs.

## Experiment with different partitioning strategies, such as repartitioning the DataFrame, to evaluate changes in query execution time.

## **V. Results and Discussion:**

## **How partitions change speed**

Lastly, Spark allows us to examine how the number of partitions interacts with computational speed. To do so, we must analyze the number of partitions that the underlying RDD of our tables is divided into.

Spark automatically partitions DataFrames, so the first thing we want to learn is how many partitions are currently set up for the RDD underneath our reviews tables. We do so using the rdd.getNumPartitions():

To test various partition quantities, we will need to be able to manually alter the number of partitions for a given DataFrame. PySpark makes this very easy to do using the repartition() method.

### **Speed test 1: simple query**

In our first test, we will vary the number of RDD partitions from 20 to 40 (using increments of 1). The query we are running with each partition is a simple selection of three columns from the **reviews** table.

A graph with dots and lines

Description automatically generated

### **Speed test 2: query involving UDF and COUNT()**

In the second test, we again vary the number of partitions from 20 to 60. This time, the query is a bit more involved, applying our udf\_year as well as a COUNT() function.

A line of dots on a white background

Description automatically generated

Interestingly, query execution time does not show a monotonic trend as the number of partitions varies. For our simple query (Speed Test 1), between 30 and 35 partitions seem to be the optimal level for this particular dataset. For the slightly more involved query (Speed Test 2), query execution time does not appear to vary much with the number of partitions. If anything, this test suggests that around 45-50 partitions is optimal. This is quite close to the 40 partitions Spark originally chose.

To determine whether other factors are distoring the query execution time, more research would need to be done on the architecture of the cluster we are running, and how the partitions are distributed between computers in our setup.

## **Conclusion**

The Spark SQL module adds powerful functionality to the existing Apache Spark stack, allowing the user to combine relational with procedural processing. High processing speed is provided through the Catalyst optimizer, which Spark SQL runs through. Moreover, user-defined functions give the user even more flexibility, allowing user-generated functions to be used in queries. These can be extended to complex functions, such as machine-learning predictors—this would be an ample topic for further demonstrative research projects. Our tests suggests that, to optimize query execution time, the RDD underlying the Yelp dataset should be divided into about 30-35 partitions, very close to Spark's original paritioning decision.

## **References**

[1] Apache Spark project. [http://spark.apache.org](http://spark.apache.org/).

[2] Armbrust, M., Xin, R. S., Lian, C., Huai, Y., Liu, D., Bradley, J. K., ... & Zaharia, M. (2015, May). Spark SQL: Relational data processing in spark. In Proceedings of the 2015 ACM SIGMOD international conference on management of data (pp. 1383-1394). ACM.

[3] Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., ... & Stoica, I. (2012, April). Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation (pp. 2-2). USENIX Association.

[4] Laskowski, J. (2019). The Internals of Apache Spark. Gitbook. <https://legacy.gitbook.com/book/jaceklaskowski/mastering-apache-spark/details>.

[5] Apache Spark: Spark SQL, DataFrames and Datasets Guide. <https://spark.apache.org/docs/latest/sql-programming-guide.html>.

[6] Armbrust, M. (2017). Apache® Spark™ DataFrames: Simple and Fast Analysis of Structured Data. Databricks webinar. <https://pages.databricks.com/Spark-DataFrames-Structured-Data.html>.

[7] Armbrust, M., Huai, Y., Liang, C., Xin, R., Zaharia, M. (2015). Deep Dive into Spark SQL’s Catalyst Optimizer. <https://databricks.com/blog/2015/04/13/deep-dive-into-spark-sqls-catalyst-optimizer.html>.

[8] Horrocks, I. (2003). Database Connectivity: ODBC, JDBC and SQLJ. <http://www.cs.man.ac.uk/~horrocks/Teaching/cs2312/Lectures/Handouts/jdbc.pdf>.

[9] Gupta, A. (2016). Complete Guide on DataFrame Operations in PySpark. Analytics Vidhya. <https://www.analyticsvidhya.com/blog/2016/10/spark-dataframe-and-operations/>.

[10] Powers, M. (2016). Managing Spark Partitions with Coalesce and Repartition. Medium. <https://medium.com/@mrpowers/managing-spark-partitions-with-coalesce-and-repartition-4050c57ad5c4>.

[11] Yelp Open Dataset. <https://www.yelp.com/dataset>.