

Sentiment Analysis of Yelp's Businesses Reviews, And User Data" Using 3-NF



What is Yelp?

Yelp is a popular online directory of local businesses from various industries: bars, restaurants, spas, gas stations etc. Results of a search query are filtered by geographical location, business type, price ranges, and unique features. The idea behind Yelp is for users to leave their rating and a review of their experience with an establishment. It can be a treasure chest of guidelines on how customers view your business, and what improvements can be made to see more happy customers. So, how exactly can Yelp reviews help us explore the satisfaction of our customers.



popular online directory of local businesses from various industries



The idea behind Yelp is for users to leave their rating and a review of their experience with an establishment.



Yelp reviews can be used to gain insights into customer experiences, identify areas for improvement, and gauge customer sentiment towards the business.

We use to create data pipeline that converts the non-relational Yelp dataset, which is dispersed over **JSON** files in an **Amazon S3 bucket**, into a 3NF-normalized dataset kept on Amazon Redshift. The resulting schema serves as the authoritative source for analytical queries and business intelligence (BI) tools, guaranteeing data consistency and referential integrity across tables. The data was further enhanced with demographic and weather information from outside data sources.

PROJECT DEVELOPMENT

Our project aims to create a data pipeline that converts non-relational Yelp dataset into a 3NF-normalized dataset kept on Amazon Redshift.



We use ETL tools such as Apache Spark, Amazon Redshift, and Apache Airflow to transform Yelp's data and integrate it with external demographic and weather data sources

Throughout the procedure, we use **ETL tools such as:**

#1 Apache Spark,

#2 Amazon Redshift, and

#3 Apache Airflow.

More to implement in upcoming stages

Data sources in implementation:



Our project relies on three main data sources to provide insights into Yelp data:



Yelp open dataset









US Cities Demographics dataset,



Historical Hourly Weather Data 2012-2017 dataset.

Few attributes that we are considering for our data loading and transformation (from our data model dictionary.)

-  *Businesses ; business_attributes ; business_categories and categories*
-  *business_hours*
-  *cities ; city_weather*
-  *checkins ; reviews*
-  *users, elite_years and friends*
-  *tips*

Brief :

This describe the process of remodeling a database schema for Yelp, a website that provides crowd-sourced reviews of local businesses. The schema includes several tables such as businesses, business_attributes, business_categories, categories, business_hours, addresses, cities, city_weather, checkins, reviews, users, elite_years, friends, and tips. The businesses table contains the name of the business, the current star rating, the number of reviews, and whether the business is currently open. Other tables such as business_attributes, business_categories, categories, business_hours, and addresses were created to normalize the data and ensure the data is stored efficiently. The information also highlights the challenges faced in converting nested dictionaries into data types and how data was grouped and aggregated to provide a neutral representation for easy querying.

Progress of project with ETL tools & How and why we are using:

Step 1: Load from S3



We are about to design data pipeline dynamically loads the JSON files from S3, processes them, and stores their normalized and enriched versions back into S3 in Parquet format. After this, Redshift takes over and copies the tables into a DWH.

Here, All three datasets reside in a Amazon S3 bucket, which is the easiest and safest option to store and retrieve any amount of data at any time from any other AWS service.

Step 2: Process with Spark



The data needs to be transformed into a relational form to be loaded into Amazon Redshift, which does not support nested data. Apache Spark is used to perform this transformation and execute the data processing pipeline in an Amazon EMR cluster. Spark also allows for data quality checks to be performed at this stage.

Step 3: Unload to S3

Parquet is a flat columnar storage format that is more efficient in terms of storage and performance compared to traditional row-oriented formats. It supports various data types without the need for post-processing, and is well-supported in the AWS ecosystem. We can use an AWS Glue crawler to discover and register the schema for our datasets to be used in Amazon Athena. However, our ultimate goal is to materialize the data for faster retrieval without prolonged load times.

Step 4: Load into Redshift



To load Parquet data into Redshift, we can use spark-redshift or an AWS Glue job. However, it's better to define tables manually for better control over data quality, consistency, and performance. This involves issuing SQL statements to first create tables and then copy data. AWS Glue's data catalog can help derive the correct data types to make this process easier and more transparent.

Step 5: Check data quality

To ensure that the output tables are of the right size, we can perform data quality checks at the end of the data pipeline. This involves comparing the number of rows in the source data with the number of rows in the destination tables, as well as checking for any missing or duplicate data. We can also verify the consistency of the data by running some statistical analysis and profiling on the data, such as checking for null values, identifying outliers and distributions, and performing other checks that are specific to the data being analyzed. Finally, we can set up alerts and notifications to monitor any issues with the data, such as sudden changes in the size or content of the data, or any discrepancies that are detected during the data quality checks.

Step 6: (Extension) !?!

If possible we will implement Airflow DAGS in our project !



Airflow DAGs (Directed Acyclic Graphs) are used for defining and scheduling data pipelines. They allow you to define a sequence of tasks and the dependencies between them, specifying when each task should run and what should happen if a task fails. With Airflow, you can build and schedule complex data pipelines that involve multiple steps, such as data ingestion, transformation, validation, and loading. Airflow DAGs make it easy to monitor and troubleshoot

your data pipelines, as you can see the status of each task, view logs, and receive alerts when something goes wrong.

Data Extraction Overview :

- In our project, we extract data from the Yelp open dataset, which is available in JSON format on the Amazon S3 bucket.
- To extract Yelp data from JSON files, we used Apache Spark. We utilized the Spark DataFrameReader to read the JSON files from the Amazon S3 bucket and create a Spark DataFrame for each file.
- During this step, we faced challenges with handling missing or malformed data in the JSON files. We also had to deal with schema inference issues, where Spark had difficulty in automatically detecting the schema of some JSON files. We used Spark's schema inference capabilities to overcome these challenges and wrote custom code to handle missing or malformed data.

Data Transformation Overview :

- ✚ From the Yelp Dataset, we combine the demographic data set and historic dataset and after combining the data sets, we send the resulted dataset to s3.
- ✚ The merged dataset is used for sentiment analysis.

DATA SERVING OVERVIEW :

- The transformed data is loaded into Amazon Redshift for analytical queries and business intelligence (BI) tools. The user can also take the data from amazon s3.

Challenges that I have observed: (so far.....)

#1 : While Loading large volumes of data into Redshift I felt like it might be time-consuming and may require tuning of the Redshift cluster to optimize performance.

#2 : When we are Transforming the data from its original format to 3NF can be challenging, especially when dealing with nested data structures. It requires a thorough understanding of the schema and a good knowledge of data processing tools like Spark.

References:

- <https://youtu.be/5Xv5G-DJS1A>

(.....yet to be added in final report.....)

THANK YOU