**Database Management Systems**

**Project Report**

**On**

**Analyzing the impact of Social Media Marketing platforms on small business performances**

**Team Composition and Responsibilities:**

1. Devyani Deore:
   1. Implemented the ER Diagram for the given dataset.
   2. Created DDL and DML commands for all the entities as per the ER diagram and dataset.
2. Sai Sanjith Sivapuram:
   1. Implemented a Data Transformation script in Python for pre-processing and converting data from JSON to CSV format.
   2. Executed SQL queries for the entities/tables to produce Exploratory Data Analysis.
3. Gokulnath Anand:
   1. Executed SQL queries between entities/tables to produce Exploratory Data Analysis.
   2. Prepared the final Report as per the given format with all procedures and conclusions.

**Project Goal:**

The primary goal of this project is to obtain various kinds of analysis using PostgreSQL against the Yelp Dataset. This analysis helps us understand the statistical data of different businesses and how they improved over time.

This project has allowed us to utilize the following DBMS concepts learned in class:

1. Ability to understand real-world datasets and conceptualize a model (Entity Relationship Diagram).
2. Ability to convert the model into a proper SQL schema. Especially how to insert CSV file values into a table.
3. Ability to write DDL and DML commands in SQL.
4. Ability to execute join commands on the dataset.
5. Ability to execute Basic and Advanced SQL Queries.

**Attached Files**

1. Raw Yelp Dataset (JSON format): yelp\_dataset.tar
2. Data Transformation and Pre-Processing: Yelp\_Data\_Transformation.py
3. Processed Data: Yelp\_Transformed\_data.zip
4. Transformed CSV files: yelp\_user.csv, yelp\_business.csv, yelp\_review.csv
5. DDL and DML statement: Yelp\_DDL\_DML.sql
6. SQL Queries: Yelp\_Queries.sql

**About the Dataset:**

The dataset originated from the link <https://www.yelp.com/dataset>. The dataset contains the following JSON data:

1. business.json
2. checkin.json
3. review.json
4. tip.json
5. user.json

Following is the snapshot of the business data in JSON format:

A screen shot of a computer screen

Description automatically generated

**Dataset Transformation and Pre-Processing:**

As the Yelp data is in the form of JSON to make it compatible with PostgreSQL, we have created a Python script to pre-process and convert the JSON data to CSV format.

The following are the steps performed:

1. Imported Pandas and JSON library in Python.
2. Loaded the data in the form of text using Python’s built-in read functionality.
3. Converted the JSON strings to actual JSON object format called dictionary.
4. Created data frames for respective JSON data and removed all the unwanted columns and rows as necessary to ensure relevance and optimum performance of queries for Exploratory Data Analysis.
5. Saved the file in the form of CSV with the following naming structure yelp\_filename.csv.

All the steps are included in the Yelp\_Data\_Transformation.py file.

**Database Design:**

The ER Diagram below is a complete design of the final transformed and pre-processed data i.e. considering the selected data alone.

A diagram of a company

Description automatically generated

**Database Schema:**

We have converted our conceptual schema i.e. the ER Diagram into the following SQL Schema in Yelp\_DDL\_DML.sql.

The following are the commands:

**DDL Commands:**

drop table if exists business;

CREATE TABLE business(

business\_id varchar(50) NOT NULL PRIMARY KEY,

name varchar(300) NOT NULL,

address varchar(500),

city varchar(70) NOT NULL,

state varchar(4) NOT NULL,

postal\_code varchar(20),

latitude double precision NOT NULL,

longitude double precision NOT NULL,

stars double precision NOT NULL,

review\_count integer NOT NULL,

is\_open integer NOT NULL,

categories varchar(1000)

);

drop table if exists users;

CREATE TABLE users (

user\_id VARCHAR(255) PRIMARY KEY,

name VARCHAR(100),

review\_count INTEGER,

yelping\_since TIMESTAMP,

average\_stars DECIMAL(2, 1)

);

drop table if exists review;

CREATE TABLE review (

review\_id VARCHAR(255) PRIMARY KEY,

user\_id VARCHAR(255),

business\_id VARCHAR(255),

stars DECIMAL(2, 1),

useful INTEGER,

funny INTEGER,

cool INTEGER,

date TIMESTAMP,

FOREIGN KEY (business\_id) REFERENCES business(business\_id),

FOREIGN KEY (user\_id) REFERENCES users(user\_id)

);

**DML Commands:**

COPY business FROM '/Users/sanjithsivapuram/Public/yelp\_business.csv' WITH CSV HEADER DELIMITER ',' NULL '';

COPY users FROM '/Users/sanjithsivapuram/Public/yelp\_user.csv' WITH CSV HEADER DELIMITER ',' NULL '';

COPY review FROM '/Users/sanjithsivapuram/Public/yelp\_review.csv' WITH CSV HEADER DELIMITER ',' NULL '';

**NOTE:** The path mentioned in the COPY command is the location in our system which is accessible by the PgAdmin4 application. To execute the above commands ensure that the CSV file generated or used from the given data is in an accessible location by the SQL file.

**Queries For Analysis:**

-- EDA For Businesses

-- Retrieve all the business data

Getting an idea of the complete business data.

select \* from business;

-- Businesses per state

This query provides the total number of businesses per state.

select state, count(business\_id) as business\_per\_state

from business

group by state

order by business\_per\_state desc;

-- Businesses per state and city

This query provides the total number of businesses per state and city.

select state, city, count(business\_id) as business\_per\_state\_city

from business

group by state, city

order by state, city, business\_per\_state\_city desc;

-- Total Business for each business

This query provides the total number of business outlets for every business.

select name, count(business\_id) as total\_business

from business

group by name

order by total\_business desc;

-- Relation between Businesses and Stars

This query gives us the number of business with different ratings.

select stars, count(business\_id) as total\_business

from business

group by stars

order by stars desc;

-- All different ratings for each business

This query gives us the count of the business with different ratings.

select name, stars, count(business\_id) as total\_business

from business

group by name, stars

order by name, stars desc;

-- Business Categories and Ratings

SELECT categories, AVG(stars) AS average\_rating, COUNT(business\_id) AS number\_of\_businesses

FROM business

GROUP BY categories

ORDER BY average\_rating DESC, number\_of\_businesses DESC;

-- EDA for Reviews

-- Retrieve all the review data

select \* from review;

-- Trend in Average Ratings Over Time

This query can help you understand if there is a general trend of improving ratings over time, which might suggest that businesses are benefitting from customer feedback on Yelp.

SELECT DATE\_TRUNC('year', r.date) AS year, AVG(r.stars) AS average\_rating

FROM review r

GROUP BY year

ORDER BY year;

-- Reviews over time

This query can show whether businesses are receiving more reviews over time, which might suggest increased visibility or engagement due to being on Yelp.

SELECT b.business\_id, b.name, DATE\_TRUNC('year', r.date) AS year, COUNT(r.review\_id) AS total\_reviews

FROM business b

JOIN review r ON b.business\_id = r.business\_id

GROUP BY b.business\_id, b.name, year

ORDER BY b.business\_id, year;

-- Correlation between business and reviews

A higher number of reviews combined with a higher average rating might indicate positive customer engagement and satisfaction.

SELECT b.business\_id, b.name, AVG(r.stars) AS average\_rating, COUNT(r.review\_id) AS review\_count

FROM business b

JOIN review r ON b.business\_id = r.business\_id

GROUP BY b.business\_id, b.name

ORDER BY review\_count DESC, average\_rating DESC;

-- Top Businesses in terms of reviews and rating

SELECT b.business\_id, b.name, AVG(r.stars) AS average\_rating, COUNT(r.review\_id) AS review\_count

FROM business b

JOIN review r ON b.business\_id = r.business\_id

WHERE b.stars >= 4

GROUP BY b.business\_id, b.name

ORDER BY review\_count DESC, average\_rating DESC

LIMIT 10;

-- EDA for Users

-- Retrieve all the users data

select \* from users;

-- Retrieving the number of users over time

SELECT DATE\_TRUNC('year', u.yelping\_since) AS year, COUNT(u.user\_id) AS number\_of\_users

FROM users u

GROUP BY year

ORDER BY year;

**Instructions for reproducing the project:**

1. Unzip the yelp\_dataset.tar to get the raw data.
2. Open the Yelp\_Data\_Transformation.py file and change the value path for the files in the variable named file path for all three keys user, review, and business.
3. Execute the file in the terminal. Make sure you have a Python compiler in your system and the command to execute will be ‘python Yelp\_Data\_Transformation.py’.
4. Once executed the CSV files are generated near the .py file and store them in a place i.e. accessible by the SQL compiler.
5. Run the Yelp\_DDL\_DML.sql file in the SQL compiler to get the required tables with the data. Ensure that the paths for the COPY command in the file are changed pointing to the location of the files in your system (the file path must be accessible by SQL compiler to read).
6. Run the Yelp\_Queries.sql file in the SQL compiler where your tables are in the database to get the queries mentioned above as per the project goal.

**Conclusion:**

In a nutshell, we can summarize that SQL helps us a lot in performing Exploratory Data Analysis. It is easy to access a huge amount of data and it ensures consistency and reliability. SQL is useful for making big databases for various companies. Analysis of the dataset using PostgreSQL has come out with valuable insights that can be utilized for future decision-making by focusing on emerging trends.