Analysis of YELP dataset

1st Jagath Sai Narayana Kakaraparty
Dept. of ES Data Science
University at Buffalo
Buffalo, NY, USA
jagathsa@buffalo.edu

2nd Nisarg Negi
Dept. of ES Data Science
University at Buffalo
Buffalo, NY, USA
nisargne@buffalo.edu

3rd Siddhi Satish Jadhav
Dept. of ES Data Science
University at Buffalo
Buffalo, NY, USA
sjadhav3@buffalo.edu

***Abstract*—Consumer reviews have become an important deciding factor for businesses to flourish in today's world. Consumers frequently rely on certified sources for trustworthy reviews of various establishments such as restaurants, hotels, and so on. Yelp is a local business directory and review site that also has social networking capabilities. It enables people to rate and review businesses. The goal is to create a relational database system that will integrate existing Yelp data to give insightful analytics and assist existing and future business owners in making critical decisions about a new business or business expansion.**

I. Introduction

Every day, the internet generates an exponential amount of data. It is difficult to store massive volumes of data and extract insight from it. Yelp is a website that publishes reviews submitted by the public about local businesses. It allows business owners to enhance their services and users to select the best business accessible. The dataset consists of a subset of Yelp's businesses, reviews, and user information. It was originally created for the Yelp Dataset Challenge, allowing students to conduct research or analyze Yelp's data and report their findings. The collection contains information on businesses in 11 metropolitan areas across four nations. We intend to create an effective database system based on this dataset that can manage vast amounts of data and deliver insights such as reviews supplied by other consumers, ratings, tips, and so on which will be valuable to both consumers and businesses.

A. Targeted Audience

Owners that are prepared to monitor their firm and make significant decisions based on the visual plots, such as modifying the infrastructure or concentrating on an area that will help them expand, will be the system's users.

The dataset's real-world goal is to offer relevant insights and assist current and prospective business owners in making crucial choices about the

II. Data Description

The source data set is a JSON file, which is transformed into a CSV file using Python programs into tabular form. We chose to store the data in a database system for the reasons listed below: (I) The dataset is 4GB in size, making it challenging to store and access the data using CSV files. Postgres SQL, on the other hand, can store a large amount of data. Therefore, a database system would be a superior strategy for our instance. (II) The project's main goal is to establish connections between the tables in order to learn more about the businesses. Through the use of primary keys and foreign keys, Postgre SQL aids in achieving this whereas a relational schema cannot be established using Excel. (III) The yelp data has multiple groups that are there are multiple categories for each business. Using a database system, we can enlist the business based on the groups and perform further analysis. Such functionality is not present in excel. (IV) The main reason behind choosing a database system over CSV for the current data is faster access as the size of the data is quite huge. Indexing can be performed on the database tables for faster lookup whereas CSV depends on the size of the ram to access the data.

III. Data Import and processing

The dataset is an academic challenge distributed by Yelp. The dataset is in a compressed form. The uncompressed data is in form of JSON and hence requires extraction. Using python, the data is extracted into CSV and imported into Postgres via scripting.

The data that we imported was in JSON format, so we extracted and scraped the data into CSV format using python before loading it into Postgres SQL.

IV. Database and Table Formation

After extracting the data into CSV files, we had a total of 7 CSV files. We create a database called 'Yelp' in Postgres SQL and the tables created are as mentioned below.

'yelp_business' table is one of the base tables created using the yelp_business.csv by first extracting the CSV into a data frame object. We inserted the data into this table using the "to_sql" library which is used to insert the records present in the data frame object to a SQL database. We used this library for quicker insertion of data into the database. We then altered the table to add a primary key constraint on the column 'business_id'.

The 'yelp_user' table is one of the base tables created using the yelp_user.csv by first extracting the CSV into a data frame object. We inserted the data into this table using the "to_sql" library which is used to insert the records present in the data frame object to a SQL database. We used this library for quicker insertion of data into the database. We then altered the table to add a primary key constraint on the column 'user_id'.

```
CREATE TABLE YELP_BUSINESS
(BUSINESS_ID VARCHAR(50) NOT NULL,
NAME CHAR(30) NOT NULL,
NEIGHBORHOOD VARCHAR(30) NOT NULL,
ADDRESS VARCHAR(100) NOT NULL,
CITY VARCHAR(30) NOT NULL,
STATE CHAR(5) NOT NULL,
POSTAL_CODE INTEGER NOT NULL,
LATITUDE DOUBLE PRECISION NOT NULL,
LONGITUDE DOUBLE PRECISION NOT NULL,
STARS INTEGER NOT NULL,
REVIEW_COUNT INTEGER NOT NULL,
IS_OPEN INTEGER NOT NULL,
CATEGORIES VARCHAR(150) NOT NULL);
```

Query 1: Create YELP_BUSINESS Table

```
ALTER TABLE YELP_BUSINESS
ADD PRIMARY KEY (BUSINESS_ID);
```

Query 2: Alter YELP_BUSINESS Table

```
CREATE TABLE YELP_USER
(USER_ID VARCHAR(50),
NAME CHAR(50),
REVIEW_COUNT INTEGER DEFAULT 0,
YELPING_SINCE CHAR(10) DEFAULT 'NONE');
```

Query 3: Create YELP_USER Table

```
ALTER TABLE YELP_USER
ADD PRIMARY KEY (USER_ID);
```

Query 4: Alter YELP_USER Table

yelp_business table consists of a 'categories' column in the form of lists; hence, we need to remove the partial dependency. So, we created a new table 'yelp_categories' which consisted of the categories from the 'yelp_business' table, making it follow 1 NF. Insertion into the table is achieved using python code. We then altered the table to add a foreign key constraint on the column 'business_id' which is referenced from the 'yelp_business' table and we also add a delete cascade constraint on the table.

```
CREATE TABLE YELP_CATEGORIES
(BUSINESS_ID VARCHAR(50) NOT NULL,
CATEGORIES CHAR(50) NOT NULL);
```

Query 5: Create YELP_CATEGORIES Table

The 'yelp_checkin' table table created using the yelp_checkin.csv by first extracting the CSV into a data frame object. We inserted the data into this table using the "to_sql" library which is used to insert the records present in the data frame object to a SQL database. We used this library for quicker insertion of data into the database. We then altered the table to add a foreign key constraint on the column 'business_id' which is referenced from the 'yelp_business' table and we also add a delete cascade constraint on the table.

```
ALTER TABLE YELP_CATEGORIES
ADD FOREIGN KEY (BUSINESS_ID)
REFERENCES YELP_BUSINESS(BUSINESS_ID)
ON DELETE CASCADE;
```

Query 6: Alter YELP_CATEGORIES Table

```
CREATE TABLE YELP_CHECKIN
(BUSINESS_ID VARCHAR(50) NOT NULL,
WEEKDAY VARCHAR(20) NOT NULL,
HOUR TIME NOT NULL,
CHECKINS INTEGER NOT NULL);
```

Query 7: Create YELP_CHECKIN Table

```
ALTER TABLE YELP_CHECKIN
ADD FOREIGN KEY (BUSINESS_ID)
REFERENCES YELP_BUSINESS(BUSINESS_ID)
ON DELETE CASCADE;
```

Query 8: Alter YELP_CHECKIN Table

The 'yelp_business_hours' table is created using the yelp_business_hours.csv by first extracting the CSV into a data frame object. We inserted the data into this table using the "to_sql" library which is used to insert the records present in the data frame object to a SQL database. We used this library for quicker insertion of data into the database. We then altered the table to add a foreign key constraint on the column 'business_id' which is referenced from the 'yelp_business' table and we also add a delete cascade constraint on the table.

```
CREATE TABLE YELP_BUSINESS_HOURS
(BUSINESS_ID VARCHAR(50),
MONDAY VARCHAR(20) DEFAULT 'NONE',
TUESDAY VARCHAR(20) DEFAULT 'NONE',
WEDNESDAY VARCHAR(20) DEFAULT 'NONE',
THURSDAY VARCHAR(20) DEFAULT 'NONE',
FRIDAY VARCHAR(20) DEFAULT 'NONE',
SATURDAY VARCHAR(20) DEFAULT 'NONE',
SUNDAY VARCHAR(20) DEFAULT 'NONE');
```

Query 9: Create YELP_BUSINESS_HOURS Table

The 'yelp_tip' table is created using the yelp_tips.csv by first extracting the CSV into a data frame object. We inserted the data into this table using the "to_sql" library which is used to insert the records present in the data frame object to a SQL database. We used this library for quicker insertion of data into the database. We then altered the table to add a

```
ALTER TABLE YELP_BUSINESS_HOURS
ADD FOREIGN KEY (BUSINESS_ID)
REFERENCES YELP_BUSINESS(BUSINESS_ID)
ON DELETE CASCADE;
```

Query 10: Alter YELP_BUSINESS_HOURS Table

foreign key constraint on the column 'business_id' which is referenced from the 'yelp_business' table and we also add a delete cascade constraint on the table.

```
CREATE TABLE YELP_TIP
(TEXT VARCHAR(200) DEFAULT 'NONE',
LIKES INTEGER DEFAULT 0,
BUSINESS_ID VARCHAR(50) NOT NULL,
USER_ID VARCHAR(50) NOT NULL,
FOREIGN KEY (BUSINESS_ID)
REFERENCES YELP_BUSINESS(BUSINESS_ID),
FOREIGN KEY (USER_ID) REFERENCES YELP_USER(USER_ID));
```

Query 11: Create YELP_TIP Table

```
ALTER TABLE YELP_TIP
ADD FOREIGN KEY (BUSINESS_ID)
REFERENCES YELP_BUSINESS(BUSINESS_ID)
ON DELETE CASCADE;
```

Query 12: Alter YELP_TIP Table

The 'yelp_review' table is created using the yelp_review.csv by first extracting the CSV into a data frame object. We inserted the data into this table using the "to_sql" library which is used to insert the records present in the data frame object to a SQL database. We used this library for quicker insertion of data into the database. We then altered the table to add a primary key constraint on the column 'review_id' and a foreign key constraint on the column 'business_id' which is referenced from the 'yelp_business' table and on the column 'user_id' which is referenced from the 'yelp_user' table. We also add a delete cascade constraint on the table.

```
CREATE TABLE YELP_REVIEW
(REVIEW_ID VARCHAR(50) NOT NULL,
USER_ID VARCHAR(50) NOT NULL,
BUSINESS_ID VARCHAR(50) NOT NULL,
STARS INTEGER DEFAULT 0,
TEXT VARCHAR(300) DEFAULT 'NONE');
```

Query 13: Create YELP_REVIEW Table

V. Problems faced and their solution

When we tried to create tables by defining the constrains and types for each of the attributes in the tables, when we inserted data into the tables, the data was inserted into the

```
ALTER TABLE YELP_REVIEW
ADD PRIMARY KEY (REVIEW_ID);
ALTER TABLE YELP_REVIEW
ADD FOREIGN KEY (BUSINESS_ID)
REFERENCES YELP_BUSINESS(BUSINESS_ID)
ON DELETE CASCADE;
ALTER TABLE YELP_REVIEW
ADD FOREIGN KEY (USER_ID)
REFERENCES YELP_USER(USER_ID)
ON DELETE CASCADE;
```

Query 14: Alter YELP_REVIEW Table

table the with the original data types and with no constraints. Hence, we altered each of the tables to define the data types and constraints separately.

```
ALTER TABLE YELP_BUSINESS
ALTER COLUMN BUSINESS_ID TYPE VARCHAR(500),
ALTER COLUMN NAME TYPE CHAR(200),
ALTER COLUMN ADDRESS TYPE VARCHAR(200),
ALTER COLUMN CITY TYPE VARCHAR(50),
ALTER COLUMN STATE TYPE CHAR(5),
ALTER COLUMN POSTAL_CODE TYPE CHAR(10),
ALTER COLUMN LATITUDE TYPE DOUBLE PRECISION,
ALTER COLUMN LONGITUDE TYPE DOUBLE PRECISION,
ALTER COLUMN STARS TYPE DECIMAL(1,0),
ALTER COLUMN REVIEW_COUNT TYPE NUMERIC(100,0),
ALTER COLUMN IS_OPEN TYPE NUMERIC(1,0);
```

Query 15: Alter YELP_BUSINESS Table to change data types

```
ALTER TABLE YELP_BUSINESS
ALTER COLUMN BUSINESS_ID SET NOT NULL,
ALTER COLUMN ADDRESS SET NOT NULL,
ALTER COLUMN CITY SET DEFAULT 'NONE',
ALTER COLUMN STATE SET DEFAULT 'NONE',
ALTER COLUMN POSTAL_CODE SET DEFAULT '0',
ALTER COLUMN LATITUDE SET DEFAULT '0.0000',
ALTER COLUMN LONGITUDE SET DEFAULT '0.0000',
ALTER COLUMN STARS SET NOT NULL,
ALTER COLUMN REVIEW_COUNT SET NOT NULL,
ALTER COLUMN IS_OPEN SET NOT NULL;
```

Query 16: Alter YELP_BUSINESS Table to add constraints

```
ALTER TABLE YELP_USER
ALTER COLUMN USER_ID TYPE VARCHAR(500),
ALTER COLUMN NAME TYPE CHAR(200),
ALTER COLUMN REVIEW_COUNT TYPE NUMERIC(100,0),
ALTER COLUMN YELPING_SINCE TYPE DATE
USING TO_DATE(YELPING_SINCE, 'YYYY-MM-DD');
```

Query 17: Alter YELP_USER Table to change data types

VI. Functional dependencies and BCNF check

In order to maintain data integrity by removing redundancies in a database, it is necessary to normalize the database relations. The database is already normalized since we handled the partial dependency that one of the table was holding, tables

Fig. 1: Entity Relationship Diagram

```
ALTER TABLE YELP_USER
ALTER COLUMN USER_ID SET NOT NULL,
ALTER COLUMN NAME SET DEFAULT 'UNKNOWN',
ALTER COLUMN REVIEW_COUNT SET DEFAULT 0,
ALTER COLUMN YELPING_SINCE SET DEFAULT CURRENT_DATE;
```

Query 18: Alter YELP_USER Table to add constraints

```
ALTER TABLE YELP_CATEGORIES
ALTER COLUMN BUSINESS_ID TYPE VARCHAR(500),
ALTER COLUMN CATEGORIES TYPE CHAR(200);
```

Query 19: Alter YELP_CATEGORIES Table to change data types

```
ALTER TABLE YELP_CATEGORIES
ALTER COLUMN BUSINESS_ID SET NOT NULL,
ALTER COLUMN CATEGORIES SET NOT NULL;
```

Query 20: Alter YELP_CATEGORIES Table to add constraints

```
ALTER TABLE YELP_CHECKIN
ALTER COLUMN BUSINESS_ID SET NOT NULL,
ALTER COLUMN WEEKDAY SET NOT NULL,
ALTER COLUMN HOUR SET NOT NULL,
ALTER COLUMN CHECKINS SET NOT NULL;
```

Query 21: Alter YELP_CHECKIN Table to change data types

```
ALTER TABLE YELP_CHECKIN
ALTER COLUMN BUSINESS_ID TYPE VARCHAR(500),
ALTER COLUMN WEEKDAY TYPE CHAR(200),
ALTER COLUMN HOUR TYPE VARCHAR(50),
ALTER COLUMN CHECKINS TYPE NUMERIC(100,0);
```

Query 22: Alter YELP_CHECKIN Table to add constraints

are already in 1NF. Since the primary keys alone are able

```
ALTER TABLE YELP_BUSINESS_HOURS
ALTER COLUMN BUSINESS_ID TYPE VARCHAR(500),
ALTER COLUMN MONDAY TYPE VARCHAR(500),
ALTER COLUMN TUESDAY TYPE VARCHAR(500),
ALTER COLUMN WEDNESDAY TYPE VARCHAR(500),
ALTER COLUMN THURSDAY TYPE VARCHAR(500),
ALTER COLUMN FRIDAY TYPE VARCHAR(500),
ALTER COLUMN SATURDAY TYPE VARCHAR(500),
ALTER COLUMN SUNDAY TYPE VARCHAR(500);
```

Query 23: Alter YELP_BUSINESS_HOURS Table to change data types

```
ALTER TABLE YELP_BUSINESS_HOURS
ALTER COLUMN BUSINESS_ID SET NOT NULL,
ALTER COLUMN MONDAY SET DEFAULT 'NONE',
ALTER COLUMN TUESDAY SET DEFAULT 'NONE',
ALTER COLUMN WEDNESDAY SET DEFAULT 'NONE',
ALTER COLUMN THURSDAY SET DEFAULT 'NONE',
ALTER COLUMN FRIDAY SET DEFAULT 'NONE',
ALTER COLUMN SATURDAY SET DEFAULT 'NONE',
ALTER COLUMN SUNDAY SET DEFAULT 'NONE';
```

Query 24: Alter YELP_BUSINESS_HOURS Table to add constraints

```
ALTER TABLE YELP_TIP
ALTER COLUMN TEXT TYPE VARCHAR(800),
ALTER COLUMN LIKES TYPE NUMERIC(100,0),
ALTER COLUMN BUSINESS_ID TYPE VARCHAR(200),
ALTER COLUMN USER_ID TYPE VARCHAR(200);
```

Query 25: Alter YELP_TIP Table to change data types

```
ALTER TABLE YELP_TIP
ALTER COLUMN TEXT SET DEFAULT 'NONE',
ALTER COLUMN LIKES SET DEFAULT 0,
ALTER COLUMN BUSINESS_ID SET NOT NULL,
ALTER COLUMN USER_ID SET NOT NULL;
```

Query 26: Alter YELP_TIP Table to add constraints

```
ALTER TABLE YELP_REVIEW
ALTER COLUMN REVIEW_ID TYPE VARCHAR(500),
ALTER COLUMN USER_ID TYPE CHAR(200),
ALTER COLUMN BUSINESS_ID TYPE VARCHAR(200),
ALTER COLUMN STARS TYPE NUMERIC(100,0),
ALTER COLUMN TEXT TYPE VARCHAR(100000);
```

Query 27: Alter YELP_REVIEW Table to change data types

```
ALTER TABLE YELP_REVIEW
ALTER COLUMN REVIEW_ID SET NOT NULL,
ALTER COLUMN USER_ID SET NOT NULL,
ALTER COLUMN BUSINESS_ID SET NOT NULL,
ALTER COLUMN STARS SET DEFAULT 0,
ALTER COLUMN TEXT SET DEFAULT 'NONE';
```

Query 28: Alter YELP_REVIEW Table to add constraints

to determine every other attribute, the relations are in 2NF. Without the primary keys, the other attributes in the relations cannot be determined and so, relations are in 3NF. BCNF is a normal form that helps in dealing with any of the anomalies present in the database systems while designing them. We prove the relations to be in BCNF in this section. We also mention the functional dependencies that each of the relations that this database holds.

a. yelp_business

{business_id } → {business_id, name, address, city, state, postal_code, latitude, longitude, stars, review_count, is_open}

Since all the attributes can be defined by business_id alone, there exists only one FD with others being subsets of this FD. For the functional dependency shown above, the attribute on the left side i.e. business_id is a key and there exist no transitive dependencies which means the table is in BCNF.

b. yelp_user

{user_id } → {user_id, name, review_count, yelping_since}

Since all the attributes can be defined by user_id alone, there exists only one FD with others being subsets of this FD. For the functional dependency shown above, the attribute on the left side i.e. user_id is a key and there exist no transitive dependencies which means the table is in BCNF.

c. yelp_categories

{business_id } → {business_id, categories}

Since all the attributes can be defined by business_id alone, there exists only one FD with others being subsets of this FD. For the functional dependency shown above, the attribute on the left side i.e. user_id is a key and there exist no transitive dependencies which means the table is in BCNF.

d. yelp_checkin

{business_id, weekday, hour } → {business_id, weekday, hour, checkins}

Since checkins can be defined by business_id, weekday, and hour there exists only one FD. For the functional dependency shown above, the attributes on the left side i.e. business_id, weekday, hour form a key, and there exists no transitive dependencies which means the table is in BCNF.

e. yelp_business_hours {business_id } → {business_id, monday, tuesday, wednesday, thursday, friday, saturday, sunday}

Since all the attributes can be defined by user_id alone, there exists only one FD with others being subsets of this FD. For the functional dependency shown above, the attribute on the left side i.e. business_id is a key and there exists no transitive dependencies which means the table is in BCNF.

f. yelp_tip {business_id } → {business_id, monday, tuesday, wednesday, thursday, friday, saturday, sunday}

Since all the attributes can be defined by business_id and user_id, there exists only one FD with others being subsets of this FD. • For the functional dependency shown above, the attributes on the left side i.e. business_id and user_id together form a key and there exists no transitive dependencies which means the table is in BCNF.

g. yelp_review {review_id } → {review_id, business_id, user_id, stars, text}

Since all the attributes can be defined by review_id alone, there exists only one FD with others being subsets of this FD. • For the functional dependency shown above, the attribute on the left side i.e. review_id is a key and there exists no transitive dependencies which means the table is in BCNF.

VII. Improvements

Since the query execution time was large on some of the tables, we performed indexing on the tables. This section discusses those queries and the indexes used.

```
EXPLAIN ANALYSE SELECT B.NAME,
B.LATITUDE, B.LONGITUDE,
C.CATEGORIES,B.CITY,B.STATE,B.REVIEW_COUNT
FROM YELP_BUSINESS AS B
INNER JOIN YELP_CATEGORIES AS C
ON B.BUSINESS_ID = C.BUSINESS_ID
WHERE B.CITY = 'LAS VEGAS'
AND C.CATEGORIES = 'RESTAURANTS';
```

Query 29: Explain query

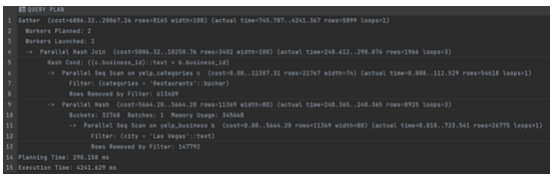


Fig. 2: Explain query before indexing

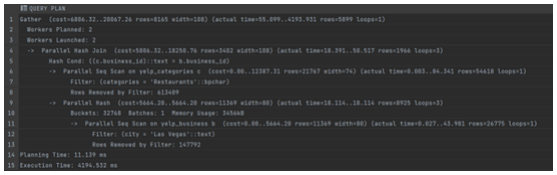


Fig. 3: Explain query after indexing

```
SELECT T.TEXT, B.NAME
FROM YELP_BUSINESS AS B
INNER JOIN YELP_CATEGORIES AS C
ON B.BUSINESS_ID = C.BUSINESS_ID
INNER JOIN YELP_TIP AS T
ON B.BUSINESS_ID = T.BUSINESS_ID
WHERE C.CATEGORIES = 'RESTAURANTS'
AND B.CITY = 'LAS VEGAS' LIMIT 10;
```

Query 30: Select query 1

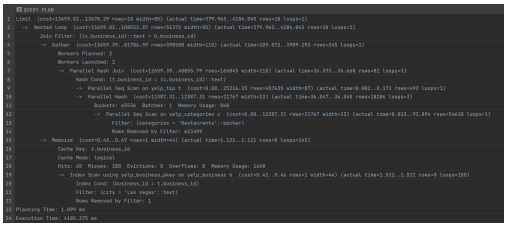


Fig. 4: Select query 1 before indexing

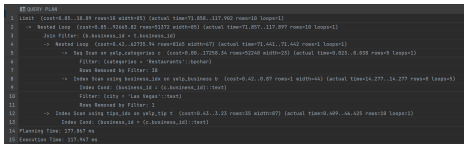


Fig. 5: Select query 1 after indexing

```
SELECT CAT, COUNT(CAT) FROM
(SELECT C.CATEGORIES AS CAT
FROM YELP_BUSINESS
AS B INNER JOIN YELP_CATEGORIES AS C
ON B.BUSINESS_ID = C.BUSINESS_ID
WHERE B.CITY = 'LAS VEGAS' )
AS TEMP GROUP BY CAT HAVING COUNT(CAT) =1;
```

Query 31: Select query 2

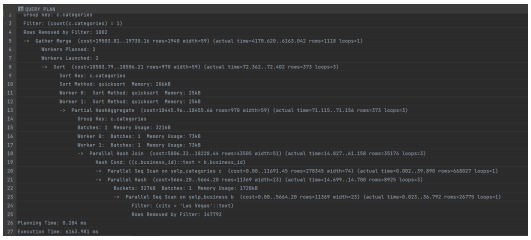


Fig. 6: Select query 2 before indexing

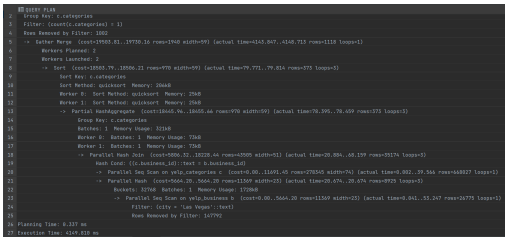


Fig. 7: Select query 2 after indexing

```
SELECT R.STARS, R."TEXT"
FROM YELP_REVIEW R INNER JOIN
YELP_CATEGORIES C
ON C.BUSINESS_ID = R.BUSINESS_ID
WHERE C.CATEGORIES = 'RESTAURANTS' LIMIT 1000;
```

Query 32: Select query 3

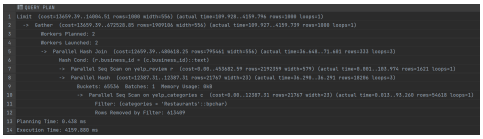


Fig. 8: Select query 3 before indexing

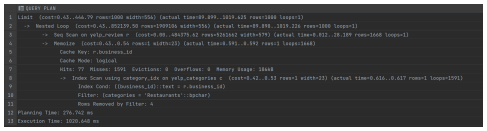


Fig. 9: Select query 3 after indexing

Indexes created

```
CREATE INDEX REVIEW_IDX
ON YELP_REVIEW(REVIEW_ID, BUSINESS_ID);
```

Query 33: Indexing query 1

```
CREATE INDEX BUSINESS_IDX
ON YELP_BUSINESS(BUSINESS_ID);
```

Query 34: Indexing query 2

```
CREATE INDEX CATEGORY_IDX
ON YELP_CATEGORIES(BUSINESS_ID);
```

Query 35: Indexing query 3

```
CREATE INDEX TIPS_IDX
ON YELP_TIP(BUSINESS_ID, USER_ID);
```

Query 36: Indexing query 4

VIII. CRUD OPERATIONS

a. Insert query

Fig. 10: Insert query

b. Select query

Fig. 11: Select query

c. Index creation query

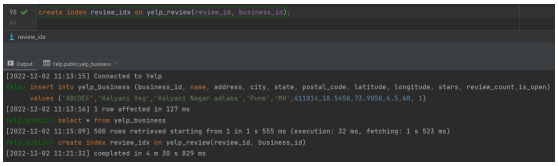


Fig. 12: Indexing query

d. Alter query

```
Yelp.public> ALTER TABLE yelp_business
                ALTER COLUMN business_id TYPE varchar(500),
                ALTER COLUMN name TYPE char(200),
                ALTER COLUMN address TYPE varchar(500),
                ALTER COLUMN city TYPE varchar(500),
                ALTER COLUMN state TYPE char(50),
                ALTER COLUMN postal_code TYPE char(50),
                ALTER COLUMN latitude TYPE double precision,
                ALTER COLUMN longitude TYPE double precision,
                ALTER COLUMN stars TYPE decimal(1,0),
                ALTER COLUMN review_count TYPE numeric(100,0),
                Alter COLUMN is_open TYPE Numeric(1,0)
[2022-12-02 11:25:11] completed in 42 s 981 ms
```

Fig. 13: Alter query

e. Delete query

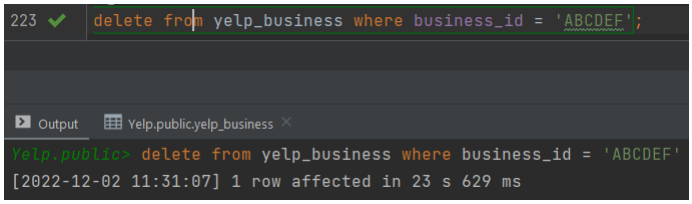


Fig. 14: Delete query

f. Order by query

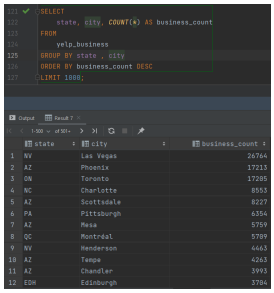


Fig. 15: Order by query

g. Join query

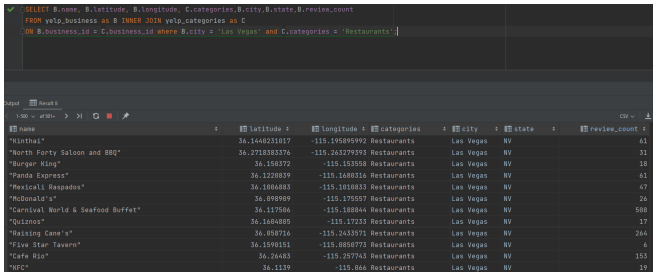


Fig. 16: Join query

IX. WEB APPLICATION

Users can visualize states with most number of businesses by selecting the number of states they want to view the data for which is shown by a Bubble plot.

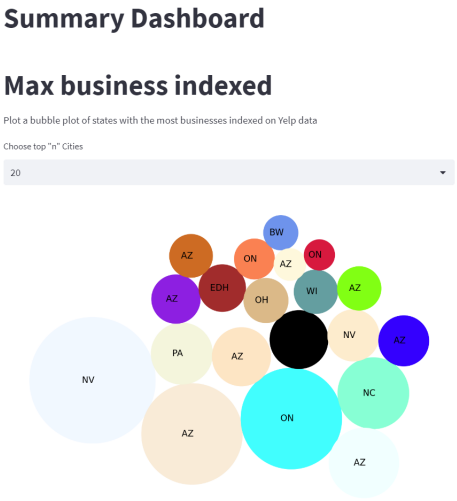


Fig. 17: Bubble plot of states with most businesses

Users can visualize the locations of the businesses with specific categories by selecting the state, business, and the category they wish to view the information about.

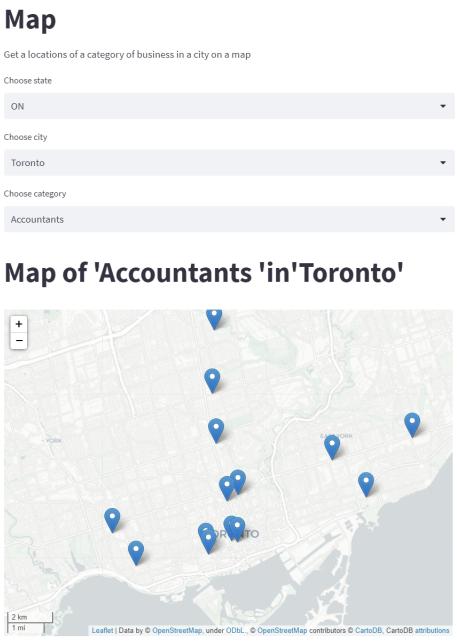


Fig. 18: Locations of business categories in a city

Users can visualize the most used positive and negative reviews by the consumers.

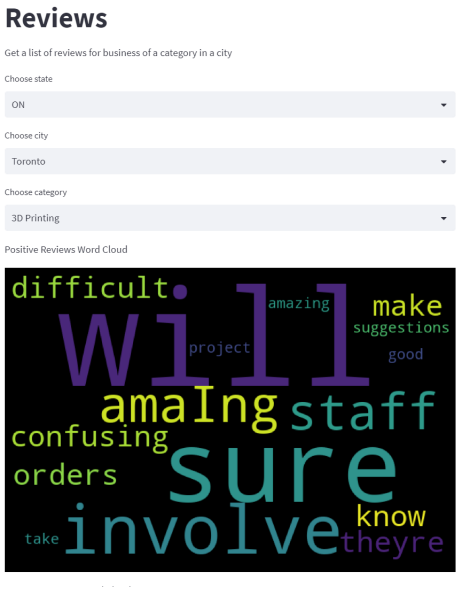


Fig. 19: Positive reviews for a business category in a city

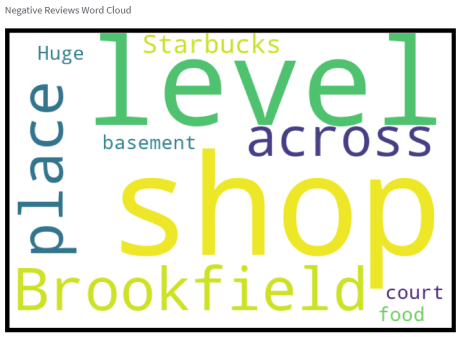


Fig. 20: Negative reviews for a business category in a city

Users can visualize the possible business opportunities for a particular category in a city since those cities would have the least number of businesses in that particular category.

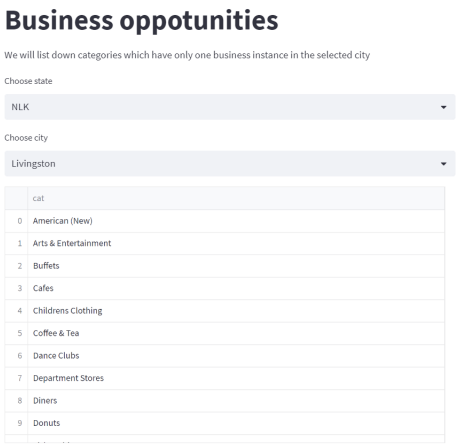
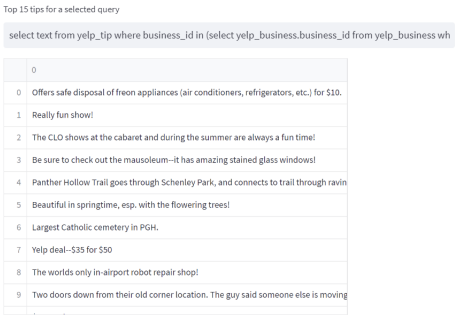


Fig. 21: Business opportunities based on categories in a city

Users can query the database to get any relevant information they would like to fetch.


Top 15 tips for a selected query

select text from yelp_tip where business_id in (select yelp_business.business_id from yelp_business wh

	0
0	Offers safe disposal of freon appliances (air conditioners, refrigerators, etc.) for $10.
1	Really fun show!
2	The CLO shows at the cabaret and during the summer are always a fun time!
3	Be sure to check out the mausoleum--it has amazing stained glass windows!
4	Panther Hollow Trail goes through Schenley Park, and connects to trail through ravin
5	Beautiful in springtime, esp. with the flowering trees!
6	Largest Catholic cemetery in PGH.
7	Yelp deal--$35 for $50
8	The worlds only in-airport robot repair shop!
9	Two doors down from their old corner location. The guy said someone else is moving

Fig. 22: Top 15 tips

X. Further Enhancements

The database that we designed is good for businesses to get enough insights, but there is a scope for a few improvements. Since this is an already populated database and static in nature, we can improve the time taken to fetch the data by creating views for the queries that are frequently accessed on the front end.

References

[1] https://www.kaggle.com/code/jagangupta/what-s-in-a-review-yelp-ratings-eda

[2] https://www.yelp.com/dataset

[3] https://github.com/sai2499/Kryptonite_Yelp$_Dataset$

[4] https://www.yelp.com/dataset/documentation/main

[5] https://github.com/Yelp/dataset-examples

[6] https://towardsdatascience.com/converting-yelp-dataset-to-csv-using-pandas-2a4c8f03bd88