Big Data Project

Group members:
Akshay Kshirsagar
Karan Manghi
Tejas Arya
Aamir Jamal

Introduction:

What has been done so far:

We have chosen the YELP dataset. Initially we extracted the data tables from the zip file. We wrote a Python script to load the data from JSON format into standard Python list and Dictionary Data structures.
We ran some checks on it to make sure any data field was blank or invalid.

1) Extract data from given Zip files.
2) Load JSON data using Python script.
3) Preprocessing steps, which include cleaning the data for any blank values or invalid entries.
4) Segregate the data so we can make some informed decisions, such as the count of unique categories, i.e the services provided by YELP.
5) Figure out the various entities which will be used in our system, such as Business, User etc.
6) Preparation for ER diagram, included distinguishing unique categories and dividing them into subsections.
7) Created tables based on our initial assumptions, like business_id , its name and address must be in the same table.
8) Completed an initial ER diagram based on given attributes.
9) Checked for primary key in column by running a script to see if that attribute was unique and could uniquely identify a record in the table. Started dividing the tables based on Business Categories.
10) The number of unique categories (i.e Restaurant, Shopping salon etc) in this dataset ranged to around 1130. We made a design choice to separate the Business table into sub tables to account for businesses which fall in the same category, based on their fields.
11) We chose to keep 4 main categories , which are Restaurants, Home Services, Shopping and Beauty & Spas. All other businesses are kept in the main Business table for the sake of simplicity.
12) Inserted the data into database by connecting Python with SQL data base.

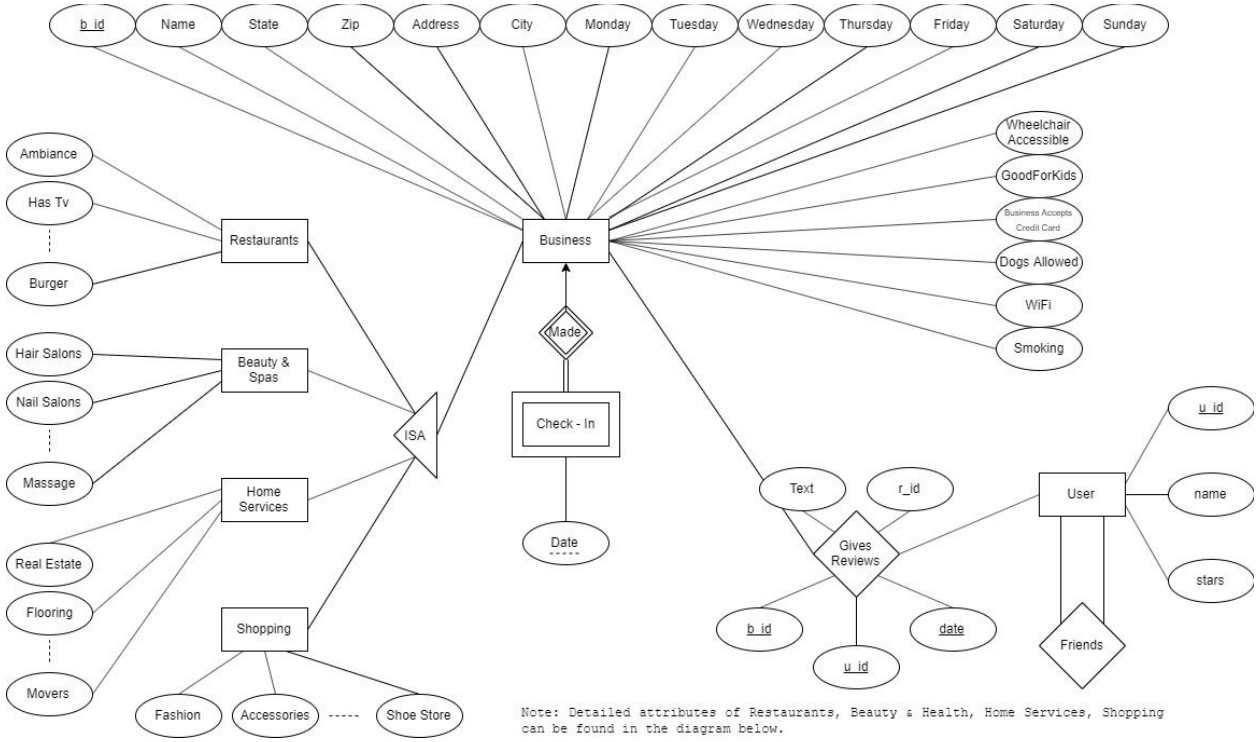
13) Ran queries to check for Normalization process, i.e performed functional dependency tests on it.
14) Now that our tables were Normalized and our ER diagram was finalized, we inserted the modified tables into database, linked them with Primary key and Foreign key constraint.
15) The queries we have inserted in the python script and SQL script are edited so that they accept the data which is thoroughly organized before importing. They will not work directly on YELP dataset.
    Example: We have distributed the “attributes” and “categories” of the business into separate tables so we can insert the multivalued data from that dictionary directly into SQL database. This would help us to extract relevant “attributes” into our new table schema.

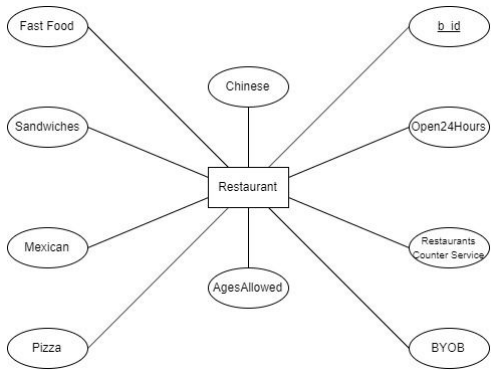
What we plan to do:

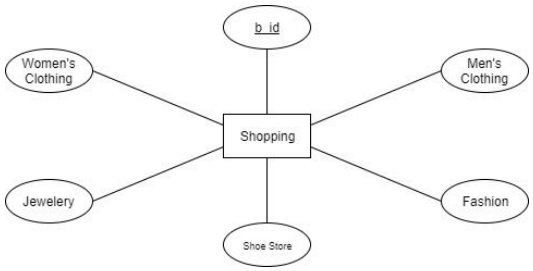
1) Our next step would be to create a basic interface and initiate proper connection between front end and back end.
2) Try out some basic queries to ensure proper functioning.
3) Provide menu to select , on what attribute user wants to search. We will provide options of categories , address , zip code to make efficient queries.
4) We plan to provide user with options to implement CRUD operations. User will be able to create new tables, delete existing records and update a record from the table.

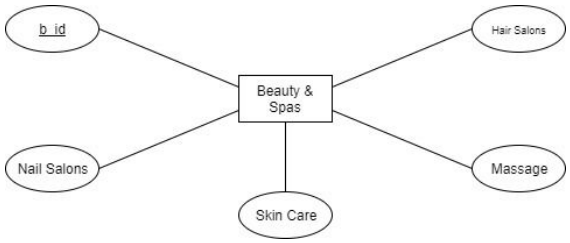
List of deliverables:

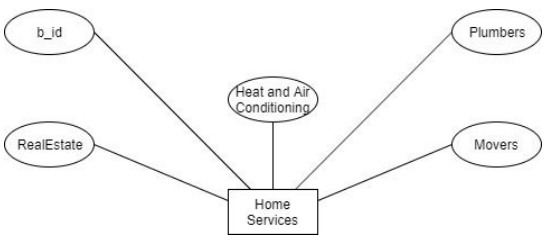
1) Interface, so that user can effectively interact with the database.
2) Query support system for retrieval and modification of records in the database.
3) Scripts to import , extract data from the YELP dataset.
4) Script to check if data is valid and in required format.


b_id
Name
State
Zip
Address
City
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
Ambiance
Has Tv
Burger
Restaurants
Business
Wheelchair Accessible
GoodForKids
Business Accepts Credit Card
Dogs Allowed
WiFi
Smoking
Hair Salons
Nail Salons
Massage
Beauty & Spas
Home Services
Real Estate
Flooring
Movers
ISA
Made
Check - In
Date
Shopping
Fashion
Accessories
Shoe Store
Text
r_id
Gives Reviews
b_id
u_id
date
User
u_id
name
stars
Friends
Note: Detailed attributes of Restaurants, Beauty & Health, Home Services, Shopping can be found in the diagram below.


Fast Food
Chinese
Sandwiches
Restaurant
Mexican
AgesAllowed
Pizza
b_id
Open24Hours
Restaurants Counter Service
BYOB


b_id
Women's Clothing
Men's Clothing
Shopping
Jewelery
Fashion
Shoe Store


b_id
Hair Salons
Beauty & Spas
Nail Salons
Massage
Skin Care


b_id
Plumbers
Heat and Air Conditioning
RealEstate
Movers
Home Services

ER To Table :

business(b_id, name, phone#, mon, tues, wed, thurs, fri, sat, sunday, state, zip, address, city, BusinessAcceptsCreditCards, GoodForKids, WiFi, WheelchairAccessible, DogsAllowed, Smoking)
PK : b_id

restaurant(b_id, pizza, fast food, sandwiches, mexican, chinese, Open24Hours, RestaurantsCounterService, BYOB, AgesAllowed)
PK: b_id
FK:b_id

beauty&spas(b_id, hair salons, nail salons, skin care, massage)
PK: b_id
FK:b_id

homeservices(b_id, real estate, movers, heat and air conditioning, plumbers)
PK: b_id
FK:b_id

shopping(b_id, fashion, women's clothing, jewellery, men's clothing, shoe store)
PK: b_id
FK:b_id

chekin(b_id,date)
PK: b_id, date
FK: b_id
Descriminator: date

user(u_id,name, stars)
PK:u_id

friends(u_id,friends_id)
PK:(u_id,friends_id)
FK1:u_id
FK2:friends_id

givesReview(b_id,u_id,text,r_id,date)
PK: (b_id,u_id)

FK1: b_id
FK2: u_id

Normalization:

Query for finding the functional dependency :

```
SELECT candidate_key
FROM table_name
GROUP BY candidate_key
HAVING COUNT(column_name) >1 ;
```

business
Candidate key = b_id

2NF : because no partial dependency
3NF: because no transitive dependency
BCNF: because there exists only one candidate key

restaurant
Candidate key = b_id

2NF : because no partial dependency
3NF: because no transitive dependency
BCNF: because there exists only one candidate key

beauty&spas
Candidate key = b_id

2NF : because no partial dependency
3NF: because no transitive dependency
BCNF: because there exists only one candidate key

shopping
Candidate key = b_id

2NF : because no partial dependency
3NF: because no transitive dependency
BCNF: because there exists only one candidate key

homeservices
Candidate key = b_id

2NF : because no partial dependency
3NF: because no transitive dependency
BCNF: because there exists only one candidate key

checkin
No need to normalise because the table has no non key attributes

user
Candidate key = b_id

2NF : because no partial dependency
3NF: because no transitive dependency
BCNF: because the primary key is the candidate key

friends
No need to normalise because the table has no non key attributes

givesReview
Candidate key = b_id, u_id

2NF : because no partial dependency
3NF: because no transitive dependency
BCNF: because the primary key is the candidate key

Note : r_id is not selected as a part of the primary key as it is a surrogate key

Query Index:

We observed that most of the queries in our database would need access business ID, business name and zip code. And hence we decided to make an index on b_id, zip, name.
The typical queries that we based our inference on:

1.) Find business details based on business name
2.) Find businesses in the given zip code
3.) Find a business providing a particular type of service in a particular area

Hence the query :

```
CREATE INDEX id_index
ON business(b_id);

CREATE INDEX name_index
ON business(name);

CREATE INDEX zip_index
ON business(zip);
```

Query Flow:

For example, if you search based only on the name of a restaurant, the query will hit the “business” table directly and get all the information in the table regarding all the businesses with that name.

So, since the above query would not make a lot of sense, you will probably want to search the name and the zip of a business. And hence in that case, the query will directly hit the business table and get all the information about the business located in that zip area.

You could also query the type of the business. So if you want to find restaurants or shopping services, the query will directly hit the restaurant or shopping table and retrieve the information.