|  |  |
| --- | --- |
| Hotel Ratings and Reviews  MSA 8040 Fall 2017 | Lei Feng  Sandeep Gunda  Mohit Savani  Liying Wei  Young Jae Woo |

**Table of Contents**

Introduction2

Project Outline2

Data Discovery and Input3

Conceptual Data Model10

Key Data Entity Definitions11

Normalization of Database Tables:12

Data Analysis13

Conclusion20

Data Source21

**Introduction**

The advent of big data has led businesses into a new age of reason. Today, we have the technology to quantify our world in countless rational ways. However, there is just one flaw in analysis solely dependent on data: Human Nature. Real people don’t behave as rationally or as predictably as data might suggest. At times, we are tribal and emotional and the reasons behind the choices we make aren’t so easily measured. However, many businesses overlook this issue and simply focus on numerical data. In this, we saw a business problem and opportunity for the hotel industry, especially in terms of online booking, as the process of traveling involves so many human factors. We focused on whether a disparity between numerical customer ratings and actual reviews of hotels existed. In doing so, we gathered and analyzed structured data, which represents the numerical ratings, and unstructured data, which represents the actual reviews. The objective of conducting such comparative analysis was to observe if there is indeed a significant disparity between the two types of data.

**Project Outline:**

* Data Source Assessment and Selection.
  + TripAdvisor
  + Expedia
  + Kaggle
  + See source horelrevies.xlsx (Appendix A)
* Data Discovery.
  + Reviewed each data source and any associated documentation
  + Created high level context diagram and described the overall process and scope of the data considered
* Model the Data.
  + Created Conceptual Data Model (Entity Relationship Diagram below)
  + Utilized a third normal form (3NF) model.
* Analyze structured Database
  + MySQL implemented on all dataset and write queries and find the problem related outputs.
* Analyze unstructured Database.
  + Write Queries in MongoDB and create sentimental model by analyzing reviews.

**Data Discovery and Input**

The team accessed unstructured data of customer reviews through The Database and Information Systems Laboratory at University of Illinois at Urbana-Champaign. Specifically, we looked at the TripAdvisor Data Set from Review data sets for “Latent Aspect Rating Analysis”. This data set consists of documents of written customer reviews grouped by hotel.

For the structured data, links to the review sites for each hotel were contained in each of these documents. From this, we were able to design a web scraping tool to gather the numerical data, which was the structured data of our project. The data set that we gathered is attached (Appendix A).

For the unstructured data, we had reviews for the hotels in separate files. Also, as we deemed a sentiment analysis was essential we ran a sentiment analysis with Semantria on the first 15 reviews for each hotel file. Afterwards, we wrote a python script that reads each hotel file and takes the first 15 reviews and makes an insert statement for MongoDB. The following are the lines of code we created and utilized:

**Web Scrape Hotel Information (structured data)**

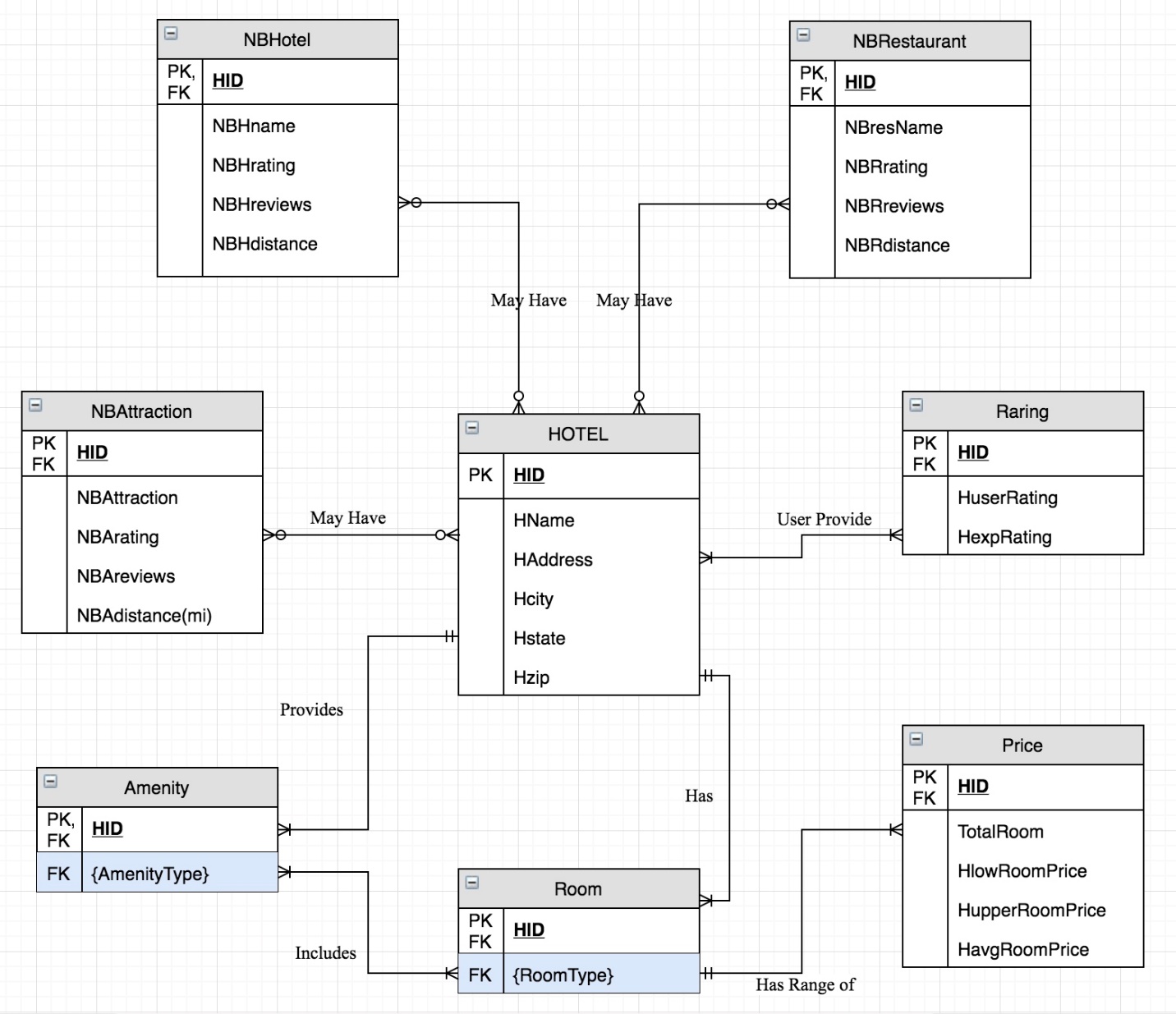
**import** requests  
**from** bs4 **import** BeautifulSoup **as** bs  
**import** csv  
**import** pandas **as** pd  
  
file = **"Hotel-2.0.xlsx"**hotel\_names = pd.ExcelFile(file)  
df1 = hotel\_names.parse(**'Sheet1'**)  
  
hotel = []  
hotel\_amenities = []  
hotel\_room\_types = []  
hotel\_nearby\_hotels = []  
hotel\_nearby\_restaurants = []  
hotel\_nearby\_attractions = []  
  
**for** page **in** df1[**'hotel\_website'**]:  
 **print**(page)  
 page\_data = requests.get(page)  
 soup = bs(page\_data.text, **"html.parser"**)  
  
 hotel\_id = page.split(**'-d'**)[1].split(**'-'**)[0]  
  
  
   
 hotel\_info\_container = soup.find(**'div'**, attrs={**'id'**: **'taplc\_location\_detail\_header\_hotels\_0'**})  
 hotel\_name = hotel\_info\_container.find(**'h1'**, attrs={**'id'**: **'HEADING'**}).text  
 hotel\_user\_rating = hotel\_info\_container.find(**'span'**, attrs={**'style'**: **'font-size:16px;'**})[**'alt'**].split(**' '**)[0]  
 hotel\_add\_str = hotel\_info\_container.find(**'span'**, attrs={**'class'**: **'street-address'**}).text.split(**","**)[0]  
 hotel\_add\_city = hotel\_info\_container.find(**'span'**, attrs={**'class'**: **'locality'**}).text.split(**","**)[0]  
 hotel\_add\_state = hotel\_info\_container.find(**'span'**, attrs={**'class'**: **'locality'**}).text.split(**", "**)[-1].split(**" "**)[0]  
 hotel\_add\_zip = hotel\_info\_container.find(**'span'**, attrs={**'class'**: **'locality'**}).text.split(**", "**)[-1].split(**" "**)[1][0:5]  
  
 amenities\_table = soup.find(**'div'**, attrs={**'class'**: **'ui\_columns section\_content'**})  
 amenities = []  
 **for** row **in** amenities\_table.findAll(**"li"**):  
 **if** (**not** [hotel\_id, row.get\_text().strip()] **in** amenities) **and** (len(row[**'class'**]) == 1):  
 amenities.append([hotel\_id, row.get\_text().strip()])  
  
 hotel\_details\_container = soup.find(**'div'**, attrs={**'class'**: **'details-top ui\_column is-4'**})  
 hotel\_price\_lower = \  
 hotel\_details\_container.find(**'ul'**, attrs={**'class'**: **'list price\_range'**}).find\_all(**'li'**, attrs={**'class'**: **'item'**})[  
 1].get\_text().split(**" ("**)[0].replace(**' '**, **''**).split(**'-'**)[0].replace(**'$'**, **''**)  
 hotel\_price\_upper = \  
 hotel\_details\_container.find(**'ul'**, attrs={**'class'**: **'list price\_range'**}).find\_all(**'li'**, attrs={**'class'**: **'item'**})[  
 1].get\_text().split(**" ("**)[0].replace(**' '**, **''**).split(**'-'**)[1].replace(**'$'**, **''**)  
  
 hotel\_expedia\_rating = hotel\_details\_container.find(**'div'**, attrs={**'title'**: **'Hotel class'**})  
 **if** (hotel\_expedia\_rating != None):  
 hotel\_expedia\_rating = float(hotel\_expedia\_rating[**'class'**][1].split(**'\_'**)[1].strip()) / 10  
 **else**:  
 hotel\_expedia\_rating = **''** room\_types\_table = hotel\_details\_container.find(**'ul'**, attrs={**'class'**: **'list room\_types'**})  
 room\_types = []  
 **for** row **in** room\_types\_table.findAll(**"li"**):  
 **if** (**not** row.get\_text() **in** room\_types) **and** (len(row[**'class'**]) == 1):  
 room\_types.append([hotel\_id, row.get\_text().replace(**','**,**''**).strip()])  
  
 hotel\_total\_rooms = \  
 hotel\_details\_container.find(**'ul'**, attrs={**'class'**: **'list number\_of\_rooms'**}).find\_all(**'li'**, attrs={**'class'**: **'item'**})[  
 1].get\_text().replace(**','**,**''**).strip()  
  
  
 **def** nearby\_adder(row, nearby\_list):  
 name = row.div.find(**'div'**, attrs={**'class'**: **'poiName'**})  
 **if** (name != None):  
 name = name.get\_text().strip()  
 **else**:  
 name = **''** rating = row.div.find(**'div'**, attrs={**'class'**: **'prw\_rup prw\_common\_bubble\_rating rating'**})  
 **if** (rating != None):  
 rating = rating.span[**'alt'**].split(**' '**)[0].strip()  
 **else**:  
 rating = **''** total\_reviews = row.div.find(**'div'**, attrs={**'class'**: **'reviewCount'**})  
 **if** (total\_reviews != None):  
 total\_reviews = total\_reviews.get\_text().split(**' '**)[0].replace(**','**, **''**).strip()  
 **else**:  
 total\_reviews = **''** distance = row.div.find(**'div'**, attrs={**'class'**: **'distance'**})  
 **if** (distance != None):  
 distance = distance.get\_text().split(**' '**)[0].strip()  
 **else**:  
 distance = **''  
  
 return** nearby\_list.append([hotel\_id, name, rating, total\_reviews, distance])  
  
  
 nearby\_hotels = soup.find\_all(**'div'**, attrs={**'class'**: **'prw\_rup prw\_common\_btf\_nearby\_poi\_grid poiGrid hotel'**})  
 **if** (nearby\_hotels != []):  
 nearby\_hotels = nearby\_hotels[0].find\_all(**'div'**)[1]  
 nearby\_hotels\_list = []  
 **for** row **in** nearby\_hotels:  
 nearby\_adder(row, nearby\_hotels\_list)  
 hotel\_nearby\_hotels.append(nearby\_hotels\_list)  
  
 nearby\_restaurants = soup.find\_all(**'div'**, attrs={**'class'**: **'prw\_rup prw\_common\_btf\_nearby\_poi\_grid poiGrid eatery'**})  
 **if** (nearby\_restaurants != []):  
 nearby\_restaurants = nearby\_restaurants[0].find\_all(**'div'**)[1]  
 nearby\_restaurants\_list = []  
 **for** row **in** nearby\_restaurants:  
 nearby\_adder(row, nearby\_restaurants\_list)  
 hotel\_nearby\_restaurants.append(nearby\_restaurants\_list)  
  
 nearby\_attractions = soup.find\_all(**'div'**,  
 attrs={**'class'**: **'prw\_rup prw\_common\_btf\_nearby\_poi\_grid poiGrid attraction'**})  
 **if** (nearby\_attractions != []):  
 nearby\_attractions = nearby\_attractions[0].find\_all(**'div'**)[1]  
 nearby\_attractions\_list = []  
 **for** row **in** nearby\_attractions:  
 nearby\_adder(row, nearby\_attractions\_list)  
 hotel\_nearby\_attractions.append(nearby\_attractions\_list)  
  
 hotel.append(  
 [hotel\_id, hotel\_name, hotel\_user\_rating, hotel\_expedia\_rating, hotel\_add\_str, hotel\_add\_city, hotel\_add\_state,  
 hotel\_add\_zip, hotel\_total\_rooms, hotel\_price\_lower, hotel\_price\_upper])  
 hotel\_amenities.append(amenities)  
 hotel\_room\_types.append(room\_types)  
  
  
**def** file\_writer(data):  
 **if** (data == hotel):  
 **for** i **in** range(len(data)):  
 writer.writerow(data[i])  
 **else**:  
 **for** i **in** range(len(data)):  
 **for** j **in** range(len(data[i])):  
 writer.writerow(data[i][j])  
  
**with** open(**'hotel.csv'**, **'a'**) **as** csv\_file:  
 writer = csv.writer(csv\_file,lineterminator=**'\n'**)  
 writer.writerow([**'hotel\_id'**,**'hotel\_name'**, **'hotel\_user\_rating'**, **'hotel\_expedia\_rating'**, **'hotel\_add\_str'**, **'hotel\_add\_city'**, **'hotel\_add\_state'**, **'hotel\_add\_zip'**, **'hotel\_total\_rooms'**, **'hotel\_price\_lower($)'**, **'hotel\_price\_upper($)'**])  
 file\_writer(hotel)  
  
**with** open(**'amenities.csv'**, **'a'**) **as** csv\_file:  
 writer = csv.writer(csv\_file,lineterminator=**'\n'**)  
 writer.writerow([**'hotel\_id'**,**'amenity'**])  
 file\_writer(hotel\_amenities)  
  
**with** open(**'room\_types.csv'**, **'a'**) **as** csv\_file:  
 writer = csv.writer(csv\_file,lineterminator=**'\n'**)  
 writer.writerow([**'hotel\_id'**, **'room\_type'**])  
 file\_writer(hotel\_room\_types)  
  
**with** open(**'nearby\_hotels.csv'**, **'a'**) **as** csv\_file:  
 writer = csv.writer(csv\_file,lineterminator=**'\n'**)  
 writer.writerow([**'hotel\_id'**,**'nearby\_hotel\_name'**,**'rating'**,**'reviews'**,**'distance'**])  
 file\_writer(hotel\_nearby\_hotels)  
  
**with** open(**'nearby\_restaurants.csv'**, **'a'**) **as** csv\_file:  
 writer = csv.writer(csv\_file,lineterminator=**'\n'**)  
 writer.writerow([**'hotel\_id'**, **'nearby\_restaurants\_name'**, **'rating'**, **'reviews'**, **'distance'**])  
 file\_writer(hotel\_nearby\_restaurants)  
  
**with** open(**'nearby\_attractions.csv'**, **'a'**) **as** csv\_file:  
 writer = csv.writer(csv\_file,lineterminator=**'\n'**)  
 writer.writerow([**'hotel\_id'**, **'nearby\_attraction\_name'**, **'rating'**, **'reviews'**, **'distance'**])  
 file\_writer(hotel\_nearby\_attractions)

**Mongo DB Insert Code**

**import** os  
**import** csv  
**import** re  
**from** numpy **import** \*  
**import** sys  
**from** datetime **import** datetime  
path = **'C:/Users/sandy/PycharmProjects/DataManagement/hotels'**files = [i **for** i **in** os.listdir(path) **if** os.path.isfile(os.path.join(path,i)) **and** \  
 **'hotel\_' in** i]  
**for** file **in** files:  
 filename = file  
 g = open(path + **"/"**+filename, **"r"**,newline=**"\n"**,errors = **'ignore'**)  
 d = g.read()  
 l = **''  
 for** i **in** range(1,16):  
 review = d.split(**"<Author>"**)[i]  
 author = review.split(**"<Content>"**)[0].replace(**"\r\n"**,**""**)  
 content = review.split(**"<Content>"**)[1].split(**"<Date>"**)[0].replace(**"\r\n"**,**""**)  
 date = review.split(**"<Date>"**)[1].split(**"\r\n"**)[0].replace(**"\r\n"**,**""**)  
 date = datetime.strptime(date, **'%b %d, %Y'**)  
 date = date.strftime(**'%Y-%m-%d'**)  
 overall = review.split(**"<Overall>"**)[1].split(**"<Value>"**)[0].replace(**"\r\n"**,**""**)  
 value = review.split(**"<Value>"**)[1].split(**"<Rooms>"**)[0].replace(**"\r\n"**,**""**)  
 rooms = review.split(**"<Rooms>"**)[1].split(**"<Location>"**)[0].replace(**"\r\n"**,**""**)  
 location = review.split(**"<Location>"**)[1].split(**"<Cleanliness>"**)[0].replace(**"\r\n"**,**""**)  
 cleanliness = review.split(**"<Cleanliness>"**)[1].split(**"<Check in / front desk>"**)[0].replace(**"\r\n"**,**""**)  
 checkin = review.split(**"<Check in / front desk>"**)[1].split(**"<Service>"**)[0].replace(**"\r\n"**,**""**)  
 service = review.split(**"<Service>"**)[1].split(**"<Business service>"**)[0].replace(**"\r\n"**,**""**)  
 bservice = review.split(**"<Business service>"**)[1].replace(**"\r\n"**,**""**)  
 hotel\_id = filename.split(**"\_"**)[1].split(**".dat"**)[0]  
  
 string = **"db.hotels.insert([{ "** + **"\"Hotel\_id\" : "** + hotel\_id + **",\"Author\" : \""** + author + **"\" , \"Date\" : new Date(\""** + date + **"\" ), \"Overall\" : "**+ overall  
 string1 = string + **", \"Value\" : "** + value + **", \"Room\" : "** + rooms + **", \"Location\" : "** + location + **", \"Cleanliness\" : "** + cleanliness  
 insert = string1 + **", \"Checkin/frontdesk\" : "** + checkin + **", \"Service\" : "** + service + **", \"BService\" : "** + bservice + **",\"Reviews\" : \""** + content + **"\",\"SentimentRate\" : 0.427, \"Sentiment\" : \"Positive\"}])"** *# print(insert)* l = l + insert + **"\n\n\n"** filewrite = open( **"C:/Users/sandy/PycharmProjects/DataManagement/inserts/"**+ str(filename.split(**"\_"**)[1].split(**".dat"**)[0]) + **"\_insert.txt"**,**"w"**)   
 filewrite.write(l)   
 filewrite.close()

**Conceptual Data Model**

Based on the structured data we gathered, we created the following conceptual data model (ER Diagram).



**Key Data Entity Definitions:**

**Hotel:** an establishment providing accommodations, meals, and other services for

travelers and tourists.

**Rating:** Hotel ratings are often used to classify hotels according to their quality. There is

a wide variety of rating schemes used by different organizations around the world. Many

have a system involving stars, with a greater number of stars indicating greater luxury.

**Amenity:** A hotel amenity is something of a premium nature provided in addition to the

room and its basics when renting a room at a hotel, motel, or other place of lodging. The

amenities provided in each hotel vary. In some places of lodging, certain amenities may

be standard with all rooms.

**Room:** Hotel room that shares a wall with an adjoining room and is connected by a

private door. Type of: bedchamber, bedroom, chamber, sleeping accommodation,

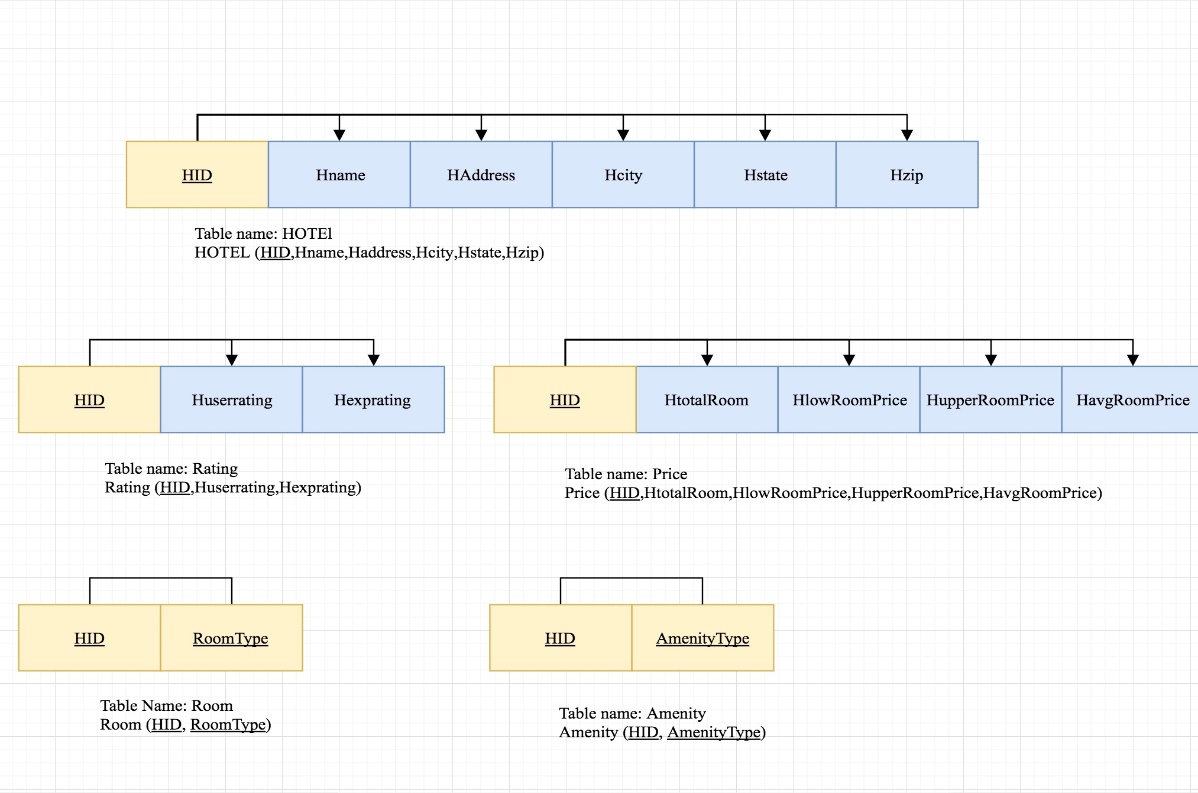
sleeping room. A room used primarily for sleeping.

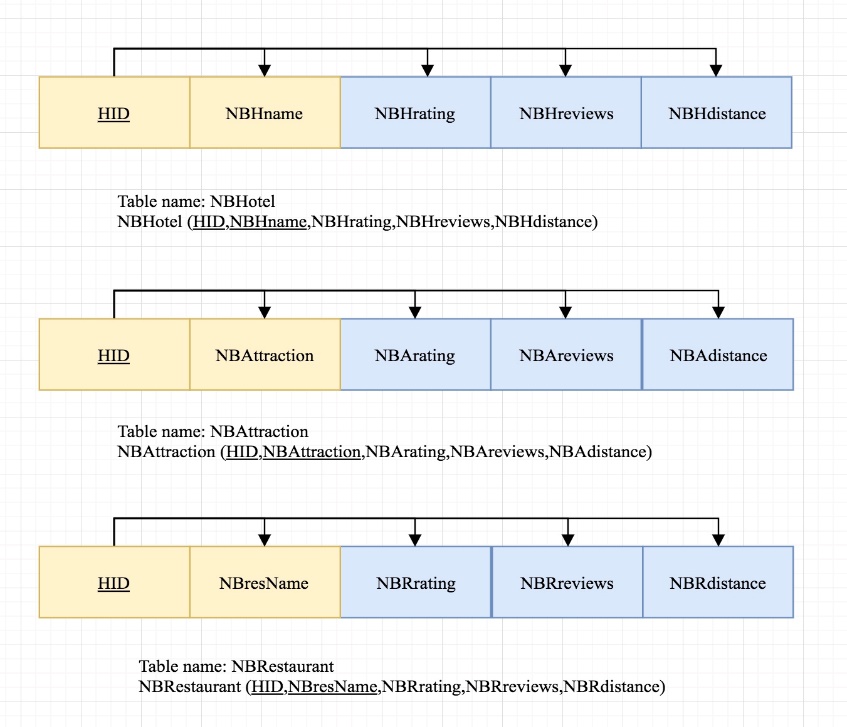
**Nearby:** Hotel nearby is the surrounding places of hotel where guest can easily

access.

**NORMALIZATION OF DATABASE TABLES:**

Database normalization is a process used to organize a database into table and columns. The idea is that a table should be about a specific topic and that only those columns which support that topic are included.





**Data Analysis**

This section contains data analysis queries that are designed to explore the structured data and unstructured data. Also, we will present our interpretation and some insights derived from the analysis.

**SQL Queries**

CREATE TABLE StateLookup(

StateName varchar (32),

StateAbbrev char (2)

);

# to match the abbreviated state in our original data

INSERT INTO StateLookup

VALUES ('Alabama', 'AL'),

('Alaska', 'AK'),

('Arizona', 'AZ'),

('Arkansas', 'AR'),

('California', 'CA'),

('Colorado', 'CO'),

('Connecticut', 'CT'),

('Delaware', 'DE'),

('District of Columbia', 'DC'),

('Florida', 'FL'),

('Georgia', 'GA'),

('Hawaii', 'HI'),

('Idaho', 'ID'),

('Illinois', 'IL'),

('Indiana', 'IN'),

('Iowa', 'IA'),

('Kansas', 'KS'),

('Kentucky', 'KY'),

('Louisiana', 'LA'),

('Maine', 'ME'),

('Maryland', 'MD'),

('Massachusetts', 'MA'),

('Michigan', 'MI'),

('Minnesota', 'MN'),

('Mississippi', 'MS'),

('Missouri', 'MO'),

('Montana', 'MT'),

('Nebraska', 'NE'),

('Nevada', 'NV'),

('New Hampshire', 'NH'),

('New Jersey', 'NJ'),

('New Mexico', 'NM'),

('New York', 'NY'),

('North Carolina', 'NC'),

('North Dakota', 'ND'),

('Ohio', 'OH'),

('Oklahoma', 'OK'),

('Oregon', 'OR'),

('Pennsylvania', 'PA'),

('Rhode Island', 'RI'),

('South Carolina', 'SC'),

('South Dakota', 'SD'),

('Tennessee', 'TN'),

('Texas', 'TX'),

('Utah', 'UT'),

('Vermont', 'VT'),

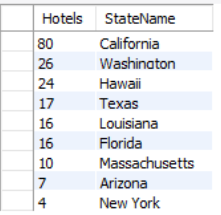
('Virginia', 'VA'),

('Washington', 'WA'),

('West Virginia', 'WV'),

('Wisconsin', 'WI'),

('Wyoming', 'WY')

SELECT \* FROM hotels.hotel;

1. **The number of hotels in the data by state**

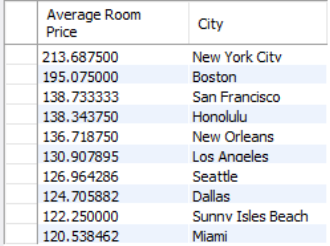
select count(HID) as 'Hotels' , StateName from hotel, statelookup

where hotel.Hstate = statelookup.StateAbbrev

group by(Hstate)

order by count(HID) desc;

1. **Top 10 most expensive cities to stay**

select avg(HavgRoomPrice) as 'Average Room Price', Hcity as 'City' from hotel,price

where hotel.HID = price.HID

group by(Hcity)

order by(avg(HavgRoomPrice)) desc

LIMIT 0, 10;

1. **Most commonly found amenities in affordable hotels where the average price per night is less than $100. Most hotels offer these basic amenities to attract customers**

select AmenityType, count(AmenityType) as 'Number' from amenity, price

where amenity.HID = price.HID and

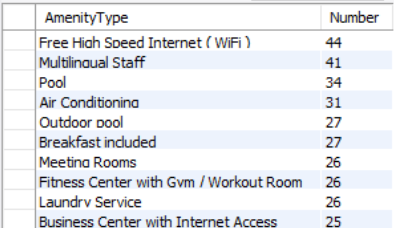
HavgRoomPrice <= 100.00

group by AmenityType

order by

count(AmenityType) desc

LIMIT 0, 10;



select round(avg(Huserrating - HexpRating), 2) as 'Average Difference' from rating;

select \* from rating

group by (Huserrating - HexpRating)

where (Huserrating - HexpRating) > round(avg(Huserrating - HexpRating), 2);

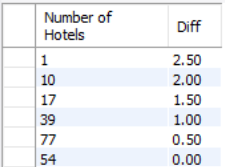
select \* from rating

group by (Huserrating - HexpRating)

having (Huserrating - HexpRating) > round(avg(Huserrating - HexpRating), 2);

1. **Based on the data from TripAdvisor, the number of hotels increases as the difference between the Expedia rating and TripAdvisor rating decrease**

select count(\*) as 'Number of Hotels', abs(Huserrating - HexpRating) as 'Diff' from rating

where abs(Huserrating - HexpRating) is not null

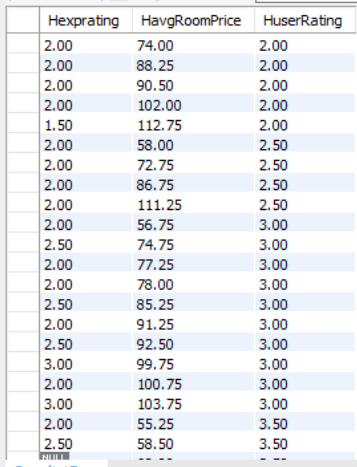
group by abs(Huserrating - HexpRating)

order by abs(Huserrating - HexpRating) desc;

1. **Study the ratings versus average prices of hotels**

select Hexprating, HavgRoomPrice, HuserRating from rating, price

where price.HID = rating.HID

****order by HuserRating,HavgRoomPrice;

**MongoDB queries**

**Average Sentiment Rate by Hotel**

db.hotels.aggregate( { $group :

{\_id : "Hotel\_id", avgSentRate :

{$avg : "$SentimentRate"} } } )

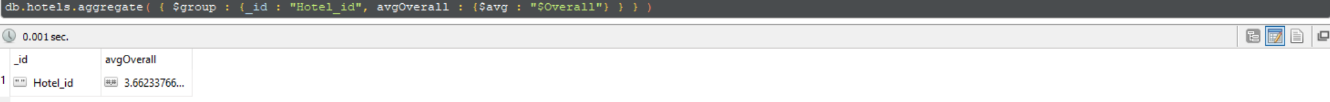


**Average Overall Rating by Hotel**

db.hotels.aggregate( { $group :

{\_id : "Hotel\_id", avgOverall :

{$avg : "$Overall"} } } )

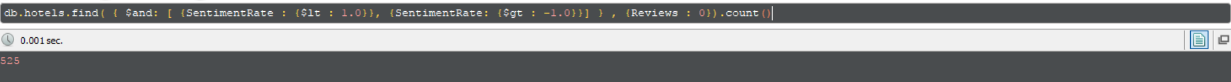


**Suppress Sentiment Rate Outliers** (SentimentRate > 1 and SentimentRate < -1)

db.hotels.find( { $and: [ {SentimentRate : {$lt : 1.0}},

{SentimentRate: {$gt : -1.0}}] } ,

{Reviews : 0}).



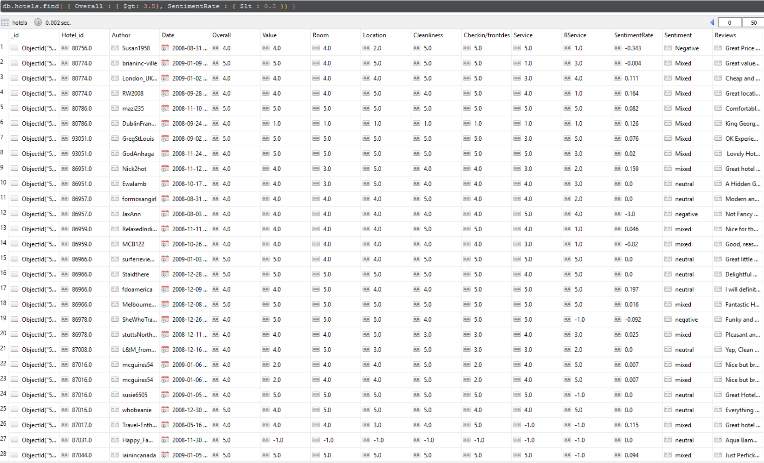
**Find “Not matching” Overall Rating and Sentiment Rate**

#Sentiment is abnormally low compared to the overall customer rating

db.hotels.find( { Overall : { $gt: 3.5},

SentimentRate :

{ $lt : 0.2 }} )

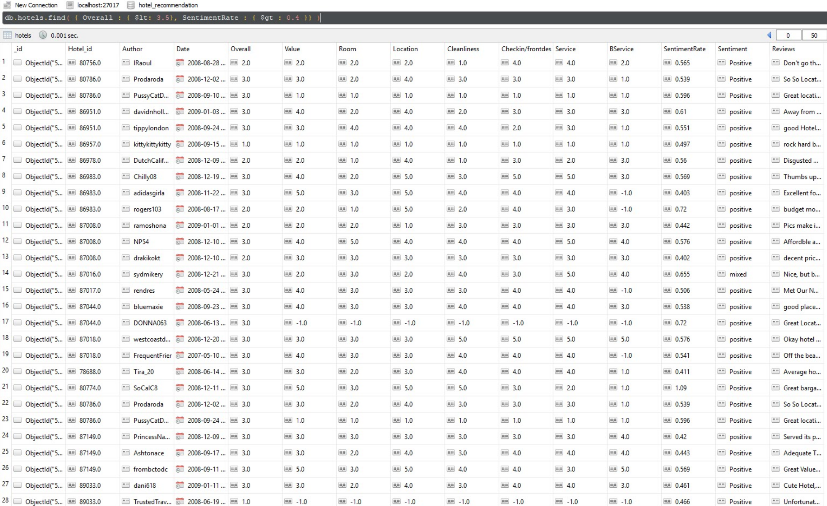


#Sentiment is abnormally high compared to the overall customer rating

db.hotels.find( { Overall : { $lt: 3.5},

SentimentRate :

{ $gt : 0.4 }} )

****

**Conclusion**

Today, hotels face fierce competition among each other and with alternative accommodation, such as Airbnb. Price competition always has limitations as there is a breakeven threshold of lowering prices. Based on our report, one method that hotels can utilizes to positively affect their sales and ultimately their bottom line is to carefully examine user written reviews and devise plans to manage them.

In addition, in writing this report we have worked end-to-end from query parsing, sentiment analysis, retrieving matching results, and ranking the results. We have gained overall understanding of how information retrieval systems work, especially in the case of datasets where text content is present.

**Data Source:**

http://times.cs.uiuc.edu/~wang296/Data/