Faitus Jeline Joseph
9/12/2022

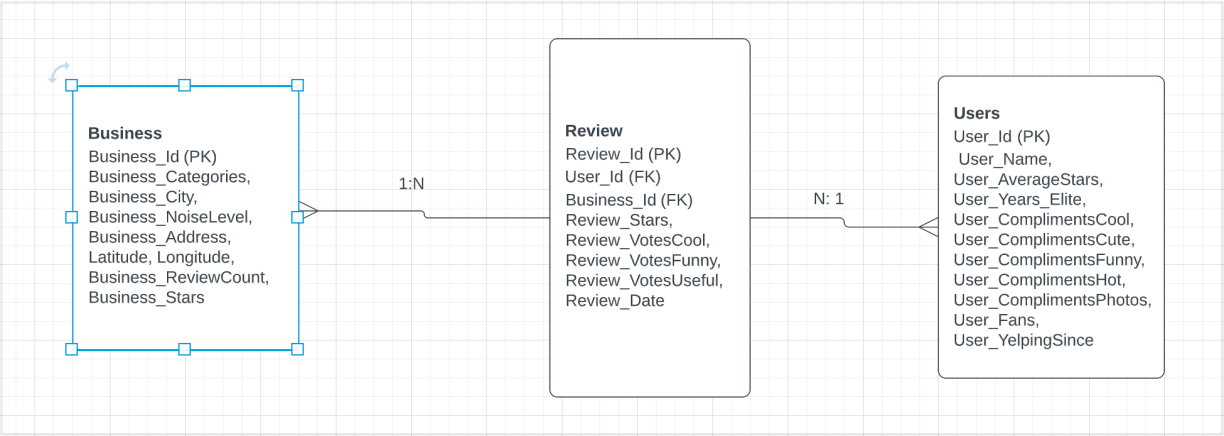
Approach:

1. Download the CSV
2. Export the CSV to a relational DB Mysql to analyze the data.
    1. yelp_data table
3. Separate the data into 3 different subject areas (Business, Users and Reviews)
    1. Business table (Dimension)
    2. Users table (Dimension)
    3. Reviews table (Fact)
4. Data Visualization using Python
    1. Top 10 Cities with the highest Yelp 5 Star reviewers
    2. Plot the number of Cool votes received based on the latitude and longitude in the US map
    3. Most popular star rating
    4. Yelp reviews over the years
    5. Relation between the restaurant's noice level and the Ratings

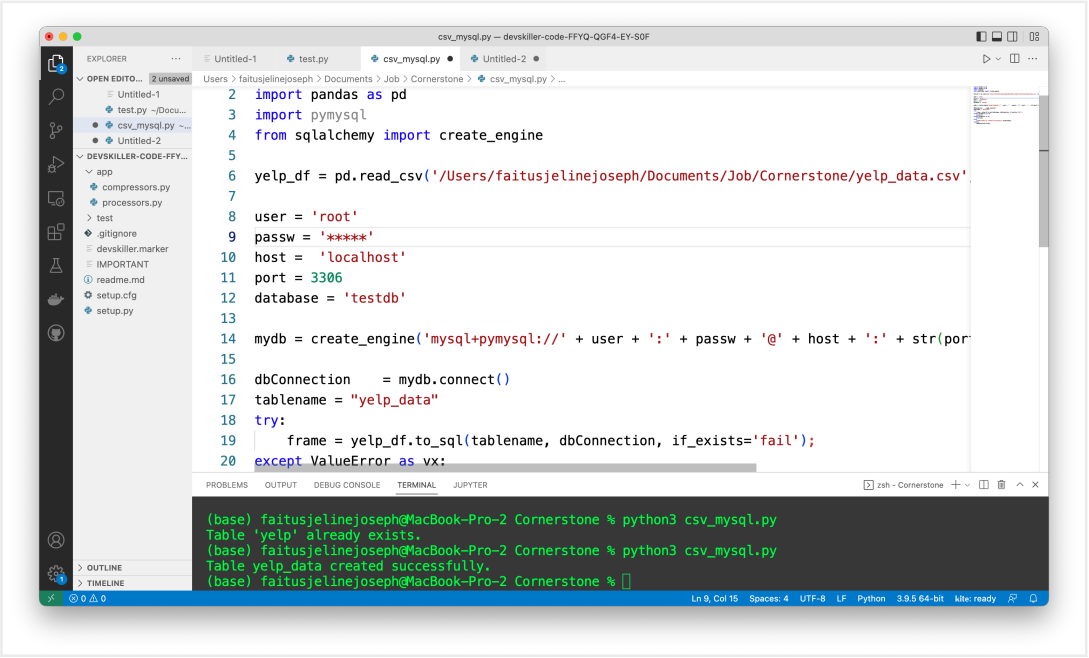
Tools and softwares used:

MySQL - Database
Python - For Data Ingestion and Visualization
VS Code - IDE
Jupyter Notebook - Data Visualization

Data Model:



Code used to load the CSV to MySQL:



DDL queries used:

select count(*) from yelp_data;
285764

Create table business as
(select distinct Business_Id, Business_Categories, Business_City,
Business_NoiseLevel,Business_Address,
Latitude,Longitude,Business_ReviewCount, Business_Stars
from yelp_data);

Create table users as

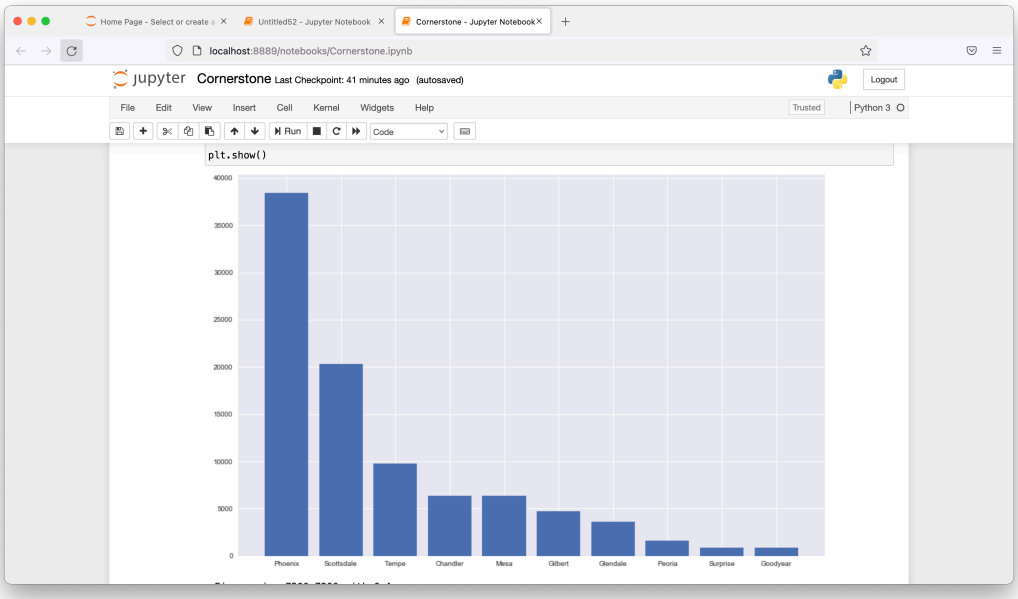
```
(select distinct User_Id,
User_Name,User_AverageStars,User_Years_Elite,User_ComplimentsCool,
User_ComplimentsCute, User_ComplimentsFunny, User_ComplimentsHot,
User_ComplimentsPhotos, User_Fans, User_YelpingSince
from yelp_data)
;

Create table review as
(select distinct Review_Id, User_Id,Business_Id,Review_Stars,
Review_VotesCool,Review_VotesFunny, Review_VotesUseful, Review_Date
from yelp_data)
;

select * from business;
select * from users;
select * from review;
```

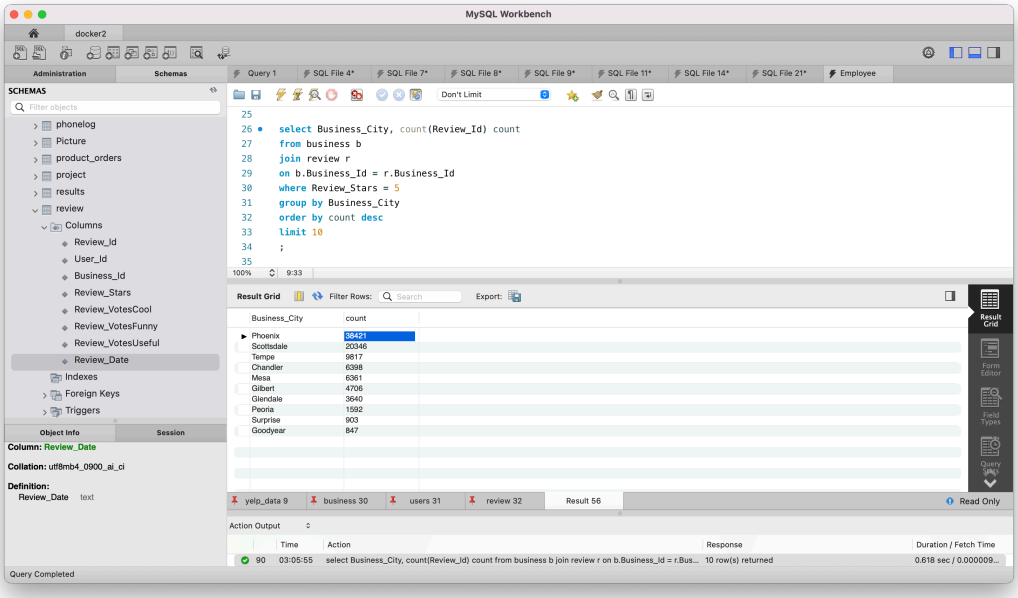
Data Visualization:

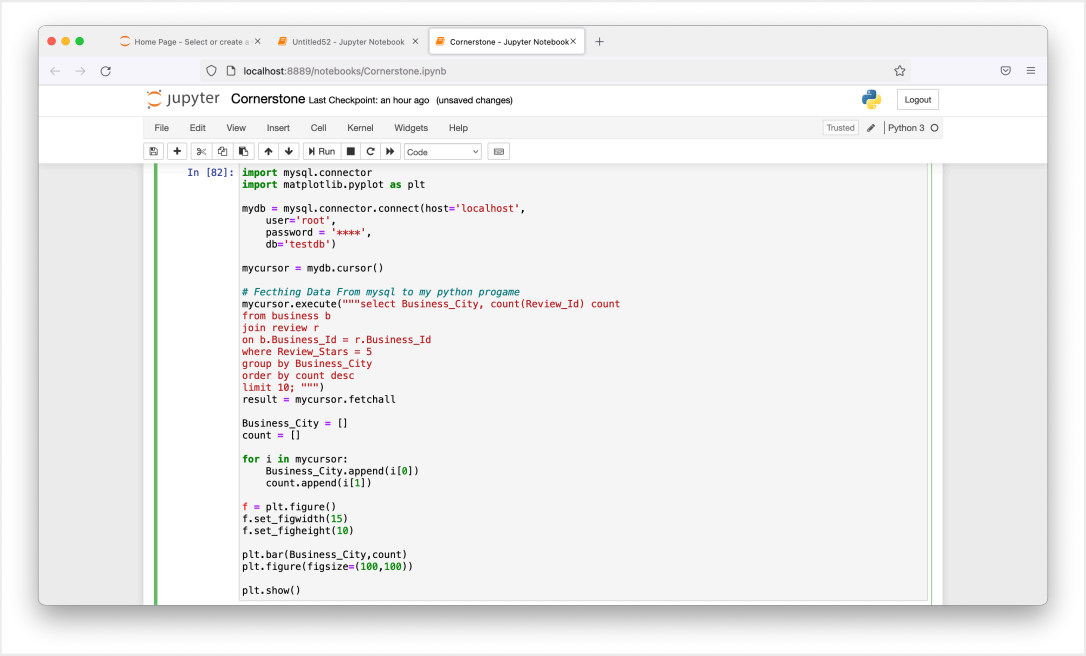
Note: For all the data visualizations, I have attached the screenshots of data visualization, query used and the python code

1. **Top 10 Cities with the highest Yelp 5 Star reviewers**

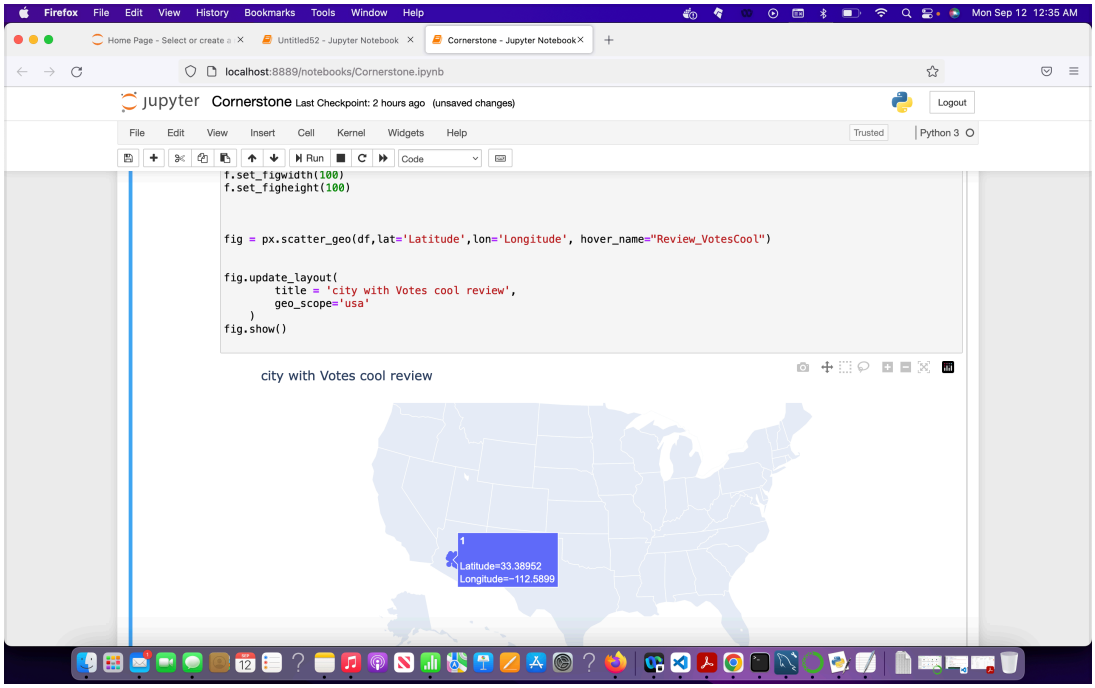
Key observations : Phoenix has the highest Yelp reviewers with a total of 38421 reviews


Home Page - Select or create a
Untitled52 - Jupyter Notebook
Cornerstone - Jupyter Notebook
localhost:8889/notebooks/Cornerstone.ipynb
jupyter Cornerstone Last Checkpoint: 41 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
plt.show()
40000
35000
30000
25000
20000
15000
10000
5000
0
Phoenix
Scottsdale
Tempe
Chandler
Mesa
Gilbert
Glendale
Peoria
Surprise
Goodyear

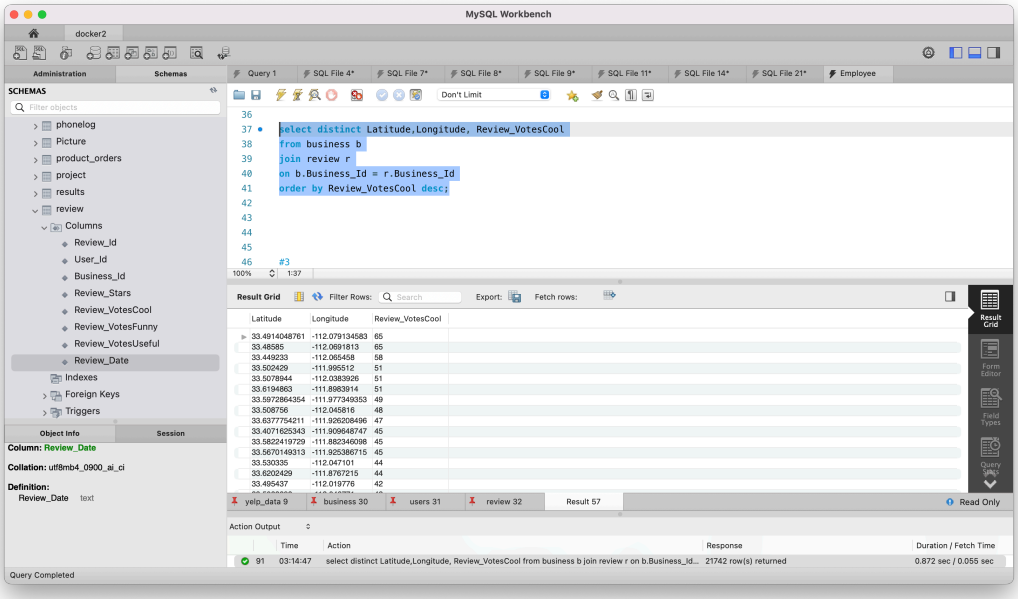

MySQL Workbench
docker2
Administration
Schemas
Query 1
SQL File 4*
SQL File 7*
SQL File 8*
SQL File 9*
SQL File 11*
SQL File 14*
SQL File 21*
Employee
SCHEMAS
phonelog
Picture
product_orders
project
results
review
Columns
Review_Id
User_Id
Business_Id
Review_Stars
Review_VotesCool
Review_VotesFunny
Review_VotesUseful
Review_Date
Indexes
Foreign Keys
Triggers
Object Info
Session
Column: Review_Date
Collation: utf8mb4_0900_ai_ci
Definition:
Review_Date text
Don't Limit
25
26 • select Business_City, count(Review_Id) count
27 from business b
28 join review r
29 on b.Business_Id = r.Business_Id
30 where Review_Stars = 5
31 group by Business_City
32 order by count desc
33 limit 10
34 ;
35
100% 9:33
Result Grid
Filter Rows:
Export:
Business_City
count
Phoenix 38421
Scottsdale 20346
Tempe 9817
Chandler 6398
Mesa 6361
Gilbert 4706
Glendale 3640
Peoria 1592
Surprise 903
Goodyear 847
yelp_data 9
business 30
users 31
review 32
Result 56
Read Only
Action Output
Time
Action
Response
Duration / Fetch Time
90 03:05:55 select Business_City, count(Review_Id) count from business b join review r on b.Business_Id = r.Bus... 10 row(s) returned 0.618 sec / 0.000009...
Query Completed

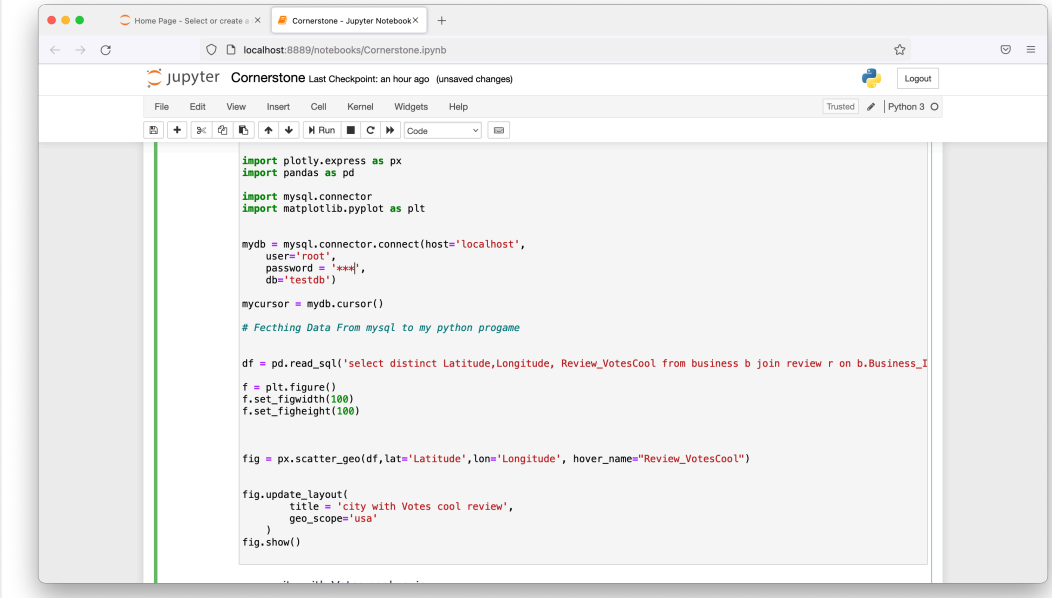


2. **Plot the number of Cool votes received based on the latitude and longitude in the US map**



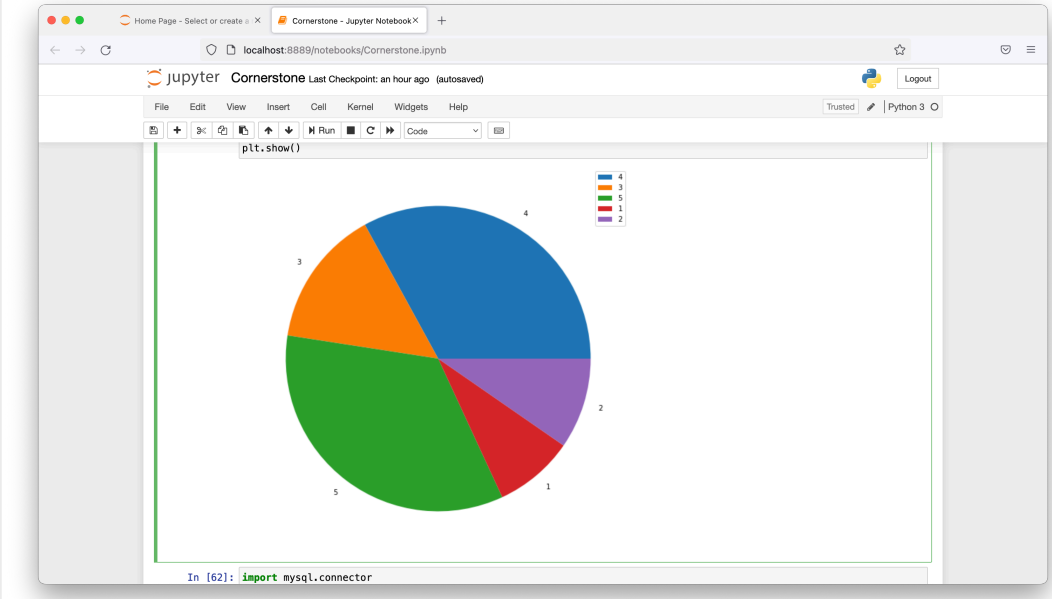

city with Votes cool review
2
Latitude=33.85766
Longitude=−112.1133

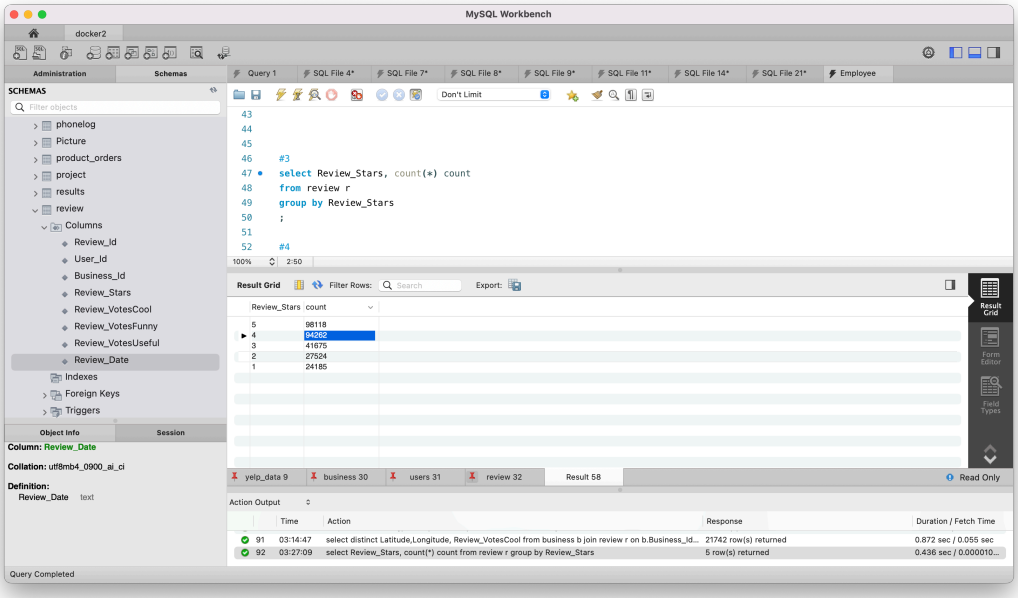

MySQL Workbench
docker2
Administration
Schemas
SCHEMAS
phonelog
Picture
product_orders
project
results
review
Columns
Review_Id
User_Id
Business_Id
Review_Stars
Review_VotesCool
Review_VotesFunny
Review_VotesUseful
Review_Date
Indexes
Foreign Keys
Triggers
Object Info
Session
Column: Review_Date
Collation: utf8mb4_0900_ai_ci
Definition:
Review_Date text
Query 1
SQL File 4*
SQL File 7*
SQL File 8*
SQL File 9*
SQL File 11*
SQL File 14*
SQL File 21*
Employee
Don't Limit
select distinct Latitude,Longitude, Review_VotesCool
from business b
join review r
on b.Business_Id = r.Business_Id
order by Review_VotesCool desc;
#3
Result Grid
Filter Rows:
Export:
Fetch rows:
Latitude
Longitude
Review_VotesCool
Action Output
Time
Action
Response
Duration / Fetch Time
Query Completed

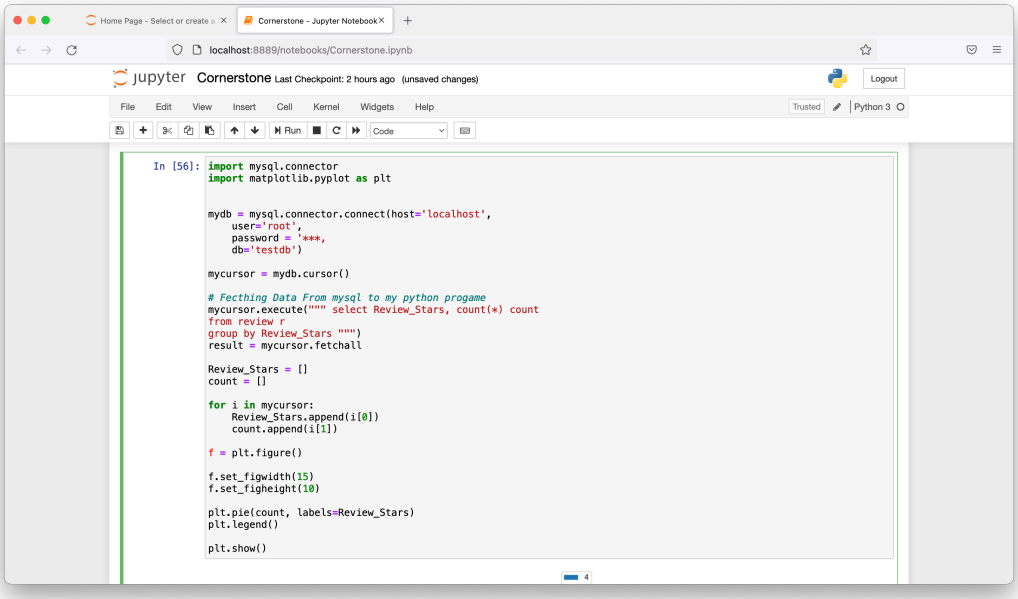


3. **Most popular star rating**

Key Observation: Star 5 rating is the most popular with the count of '98118', followed by 4 Star rating with the count of '94262'

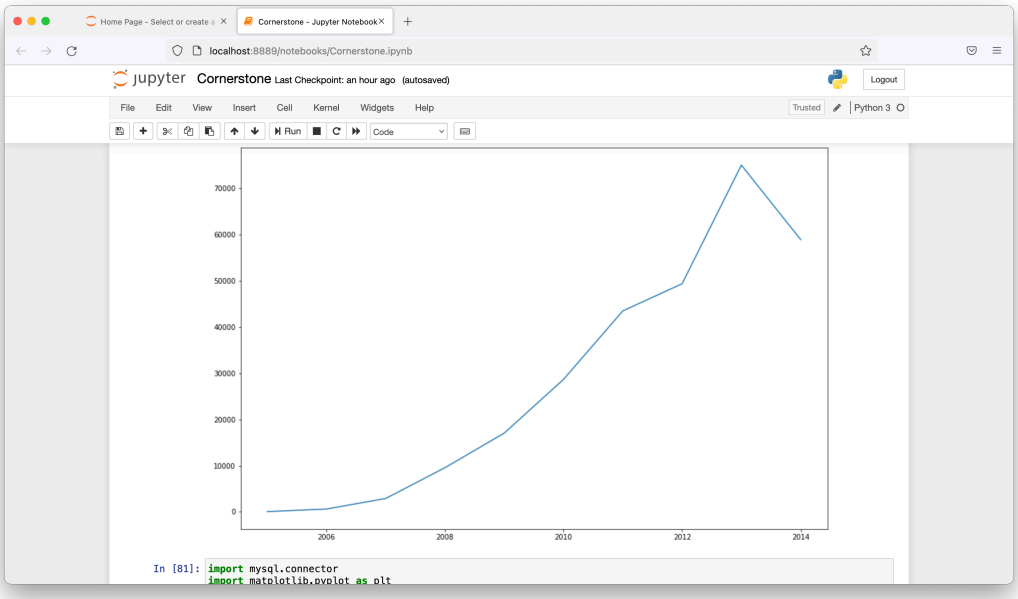


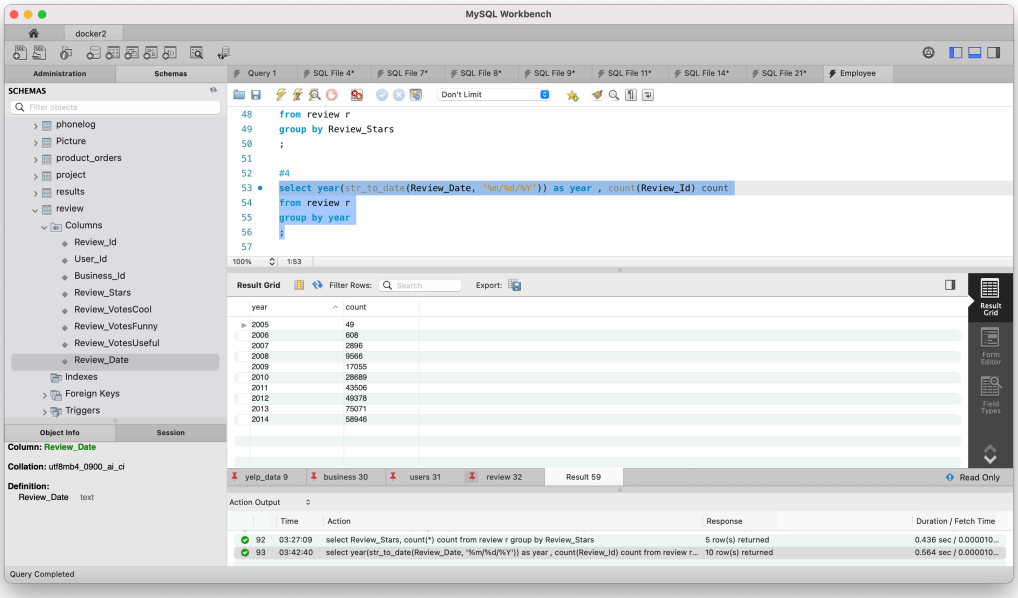


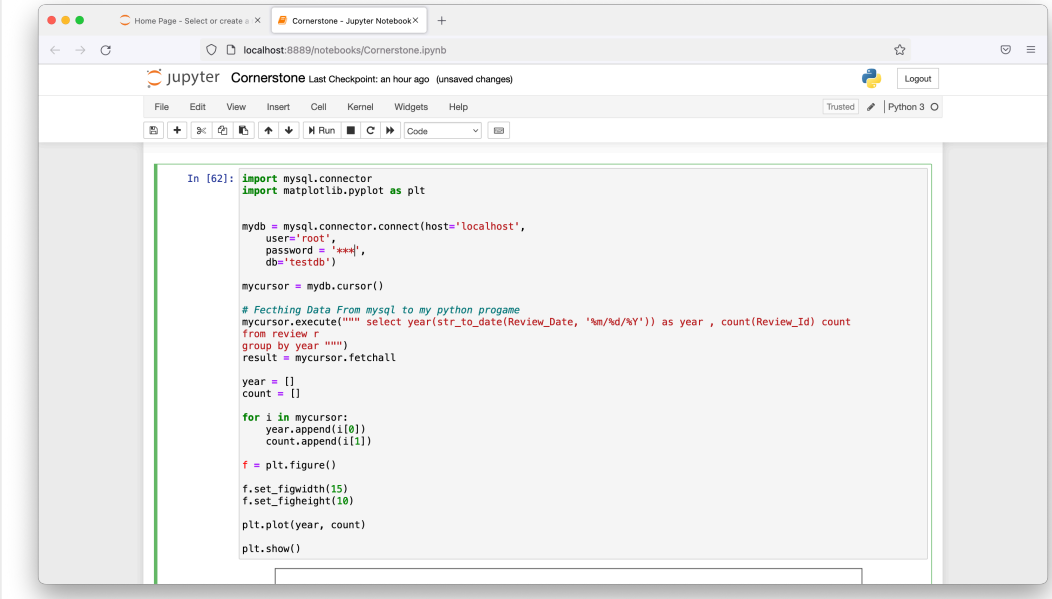


4. **Yelp reviews over the years**

Key Observations: There is a steady increase is the number of Yelp review from 2005 to 2013. But there is a sight drop in 2014

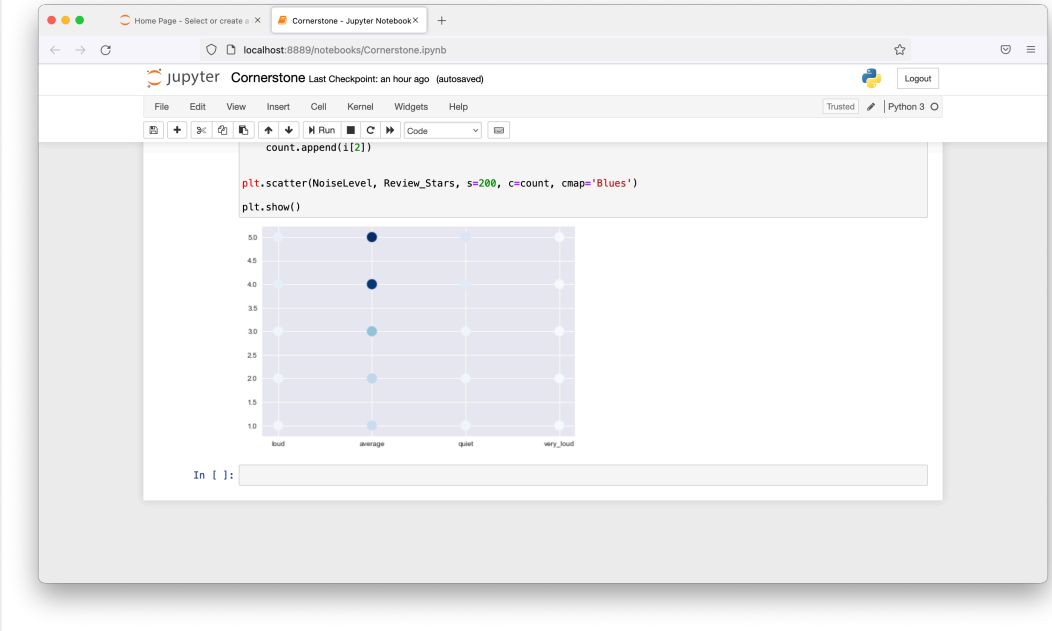

Home Page - Select or create a
Cornerstone - Jupyter Notebook
localhost:8889/notebooks/Cornerstone.ipynb
jupyter Cornerstone Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
70000
60000
50000
40000
30000
20000
10000
0
2006
2008
2010
2012
2014
In [81]:
import mysql.connector
import matplotlib.pyplot as plt

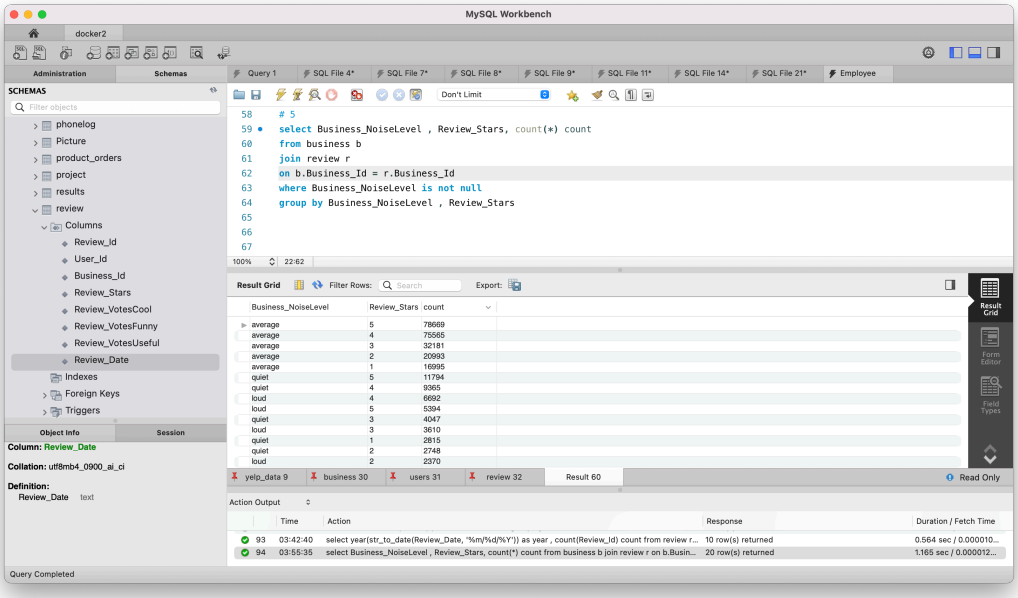

MySQL Workbench
docker2
Administration
Schemas
SCHEMAS
Filter objects
phonelog
Picture
product_orders
project
results
review
Columns
Review_Id
User_Id
Business_Id
Review_Stars
Review_VotesCool
Review_VotesFunny
Review_VotesUseful
Review_Date
Indexes
Foreign Keys
Triggers
Object Info
Session
Column: Review_Date
Collation: utf8mb4_0900_ai_ci
Definition:
Review_Date text
Query 1
SQL File 4*
SQL File 7*
SQL File 8*
SQL File 9*
SQL File 11*
SQL File 14*
SQL File 21*
Employee
Don't Limit
48 from review r
49 group by Review_Stars
50 ;
51
52 #4
53 select year(str_to_date(Review_Date, '%m/%d/%Y')) as year , count(Review_Id) count
54 from review r
55 group by year
56 ;
57
Result Grid
Filter Rows:
Search
Export:
year
count
2005 49
2006 608
2007 2896
2008 9566
2009 17055
2010 28689
2011 43506
2012 49378
2013 75071
2014 58946
yelp_data 9
business 30
users 31
review 32
Result 59
Read Only
Action Output
Time
Action
Response
Duration / Fetch Time
92 03:27:09 select Review_Stars, count(*) count from review r group by Review_Stars 5 row(s) returned 0.436 sec / 0.000010...
93 03:42:40 select year(str_to_date(Review_Date, '%m/%d/%Y')) as year , count(Review_Id) count from review r... 10 row(s) returned 0.564 sec / 0.000010...
Query Completed

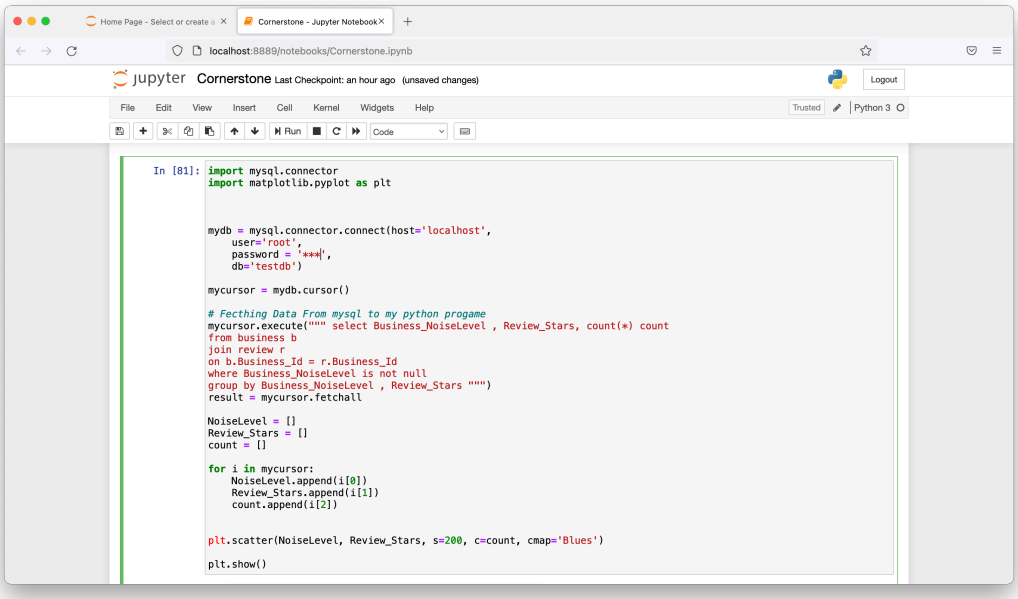


5. **Relation between the restaurant's noice level and the Ratings**

Key Observation: Majority of the users rated 5 for the Average noisy restaurants when compared to the quiet and loud noisy restaurants




MySQL Workbench
docker2
Administration
Schemas
SCHEMAS
Filter objects
phonelog
Picture
product_orders
project
results
review
Columns
Review_Id
User_Id
Business_Id
Review_Stars
Review_VotesCool
Review_VotesFunny
Review_VotesUseful
Review_Date
Indexes
Foreign Keys
Triggers
Object Info
Session
Column: Review_Date
Collation: utf8mb4_0900_ai_ci
Definition:
Review_Date text
Query 1
SQL File 4*
SQL File 7*
SQL File 8*
SQL File 9*
SQL File 11*
SQL File 14*
SQL File 21*
Employee
Don't Limit
58 # 5
59 • select Business_NoiseLevel , Review_Stars, count(*) count
60 from business b
61 join review r
62 on b.Business_Id = r.Business_Id
63 where Business_NoiseLevel is not null
64 group by Business_NoiseLevel , Review_Stars
65
66
67
100% 22:62
Result Grid
Filter Rows:
Search
Export:
Business_NoiseLevel
Review_Stars
count
average 5 78669
average 4 75565
average 3 32181
average 2 20993
average 1 16995
quiet 5 11794
quiet 4 9365
loud 4 6692
loud 5 5394
quiet 3 4047
loud 3 3610
quiet 1 2815
quiet 2 2748
loud 2 2370
yelp_data 9
business 30
users 31
review 32
Result 60
Read Only
Action Output
Time
Action
Response
Duration / Fetch Time
93 03:42:40 select year(str_to_date(Review_Date, '%m/%d/%Y')) as year , count(Review_Id) count from review r... 10 row(s) returned 0.564 sec / 0.000010...
94 03:55:35 select Business_NoiseLevel , Review_Stars, count(*) count from business b join review r on b.Busin... 20 row(s) returned 1.165 sec / 0.000012...
Query Completed


Home Page - Select or create a
Cornerstone - Jupyter Notebook
localhost:8889/notebooks/Cornerstone.ipynb
jupyter Cornerstone Last Checkpoint: an hour ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [81]:
import mysql.connector
import matplotlib.pyplot as plt
mydb = mysql.connector.connect(host='localhost',
user='root',
password = '***',
db='testdb')
mycursor = mydb.cursor()
# Fecthing Data From mysql to my python progame
mycursor.execute(""" select Business_NoiseLevel , Review_Stars, count(*) count
from business b
join review r
on b.Business_Id = r.Business_Id
where Business_NoiseLevel is not null
group by Business_NoiseLevel , Review_Stars """)
result = mycursor.fetchall
NoiseLevel = []
Review_Stars = []
count = []
for i in mycursor:
NoiseLevel.append(i[0])
Review_Stars.append(i[1])
count.append(i[2])
plt.scatter(NoiseLevel, Review_Stars, s=200, c=count, cmap='Blues')
plt.show()