BDA_Project_EDA

May 8, 2023

[1]:
```
# from google.colab import drive
# drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Fetching Data From Kaggle

Loading data from kaggle to drive and unzipping it. A kaggle.json file is required for authentication.

[2]:
```
!pip install kaggle
!mkdir ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
!kaggle datasets download -d yelp-dataset/yelp-dataset
!unzip yelp-dataset.zip
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: kaggle in /usr/local/lib/python3.10/dist-packages
(1.5.13)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-
packages (from kaggle) (2022.12.7)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-
packages (from kaggle) (1.26.15)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages
(from kaggle) (4.65.0)
Requirement already satisfied: python-dateutil in
/usr/local/lib/python3.10/dist-packages (from kaggle) (2.8.2)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.10/dist-
packages (from kaggle) (1.16.0)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.10/dist-
packages (from kaggle) (8.0.1)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-
packages (from kaggle) (2.27.1)
Requirement already satisfied: text-unidecode>=1.3 in
/usr/local/lib/python3.10/dist-packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied: charset-normalizer~=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from requests->kaggle) (2.0.12)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests->kaggle) (3.4)
```

```
cp: cannot stat 'kaggle.json': No such file or directory
chmod: cannot access '/root/.kaggle/kaggle.json': No such file or directory
Traceback (most recent call last):
  File "/usr/local/bin/kaggle", line 5, in <module>
    from kaggle.cli import main
  File "/usr/local/lib/python3.10/dist-packages/kaggle/__init__.py", line 23, in
<module>
    api.authenticate()
  File "/usr/local/lib/python3.10/dist-
packages/kaggle/api/kaggle_api_extended.py", line 164, in authenticate
    raise IOError('Could not find {}. Make sure it\'s located in'
OSError: Could not find kaggle.json. Make sure it's located in /root/.kaggle. Or
use the environment method.
Archive:  drive/MyDrive/IST718/Project/yelp-dataset.zip
  inflating: Dataset_User_Agreement.pdf
  inflating: yelp_academic_dataset_business.json
  inflating: yelp_academic_dataset_checkin.json
  inflating: yelp_academic_dataset_review.json
  inflating: yelp_academic_dataset_tip.json
  inflating: yelp_academic_dataset_user.json
```

Installing the Spark Library

[4]:
```
%%bash
pip install pyspark
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Collecting pyspark
  Downloading pyspark-3.4.0.tar.gz (310.8 MB)
                             310.8/310.8 MB 4.2 MB/s eta 0:00:00
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.10/dist-
packages (from pyspark) (0.10.9.7)
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py): started
  Building wheel for pyspark (setup.py): finished with status 'done'
  Created wheel for pyspark: filename=pyspark-3.4.0-py2.py3-none-any.whl
size=311317145
sha256=529882f2c4a2eb518557d938f89eff7b4017d9e72503f306bcf8fdf33d623f2a
  Stored in directory: /root/.cache/pip/wheels/7b/1b/4b/3363a1d04368e7ff0d408e57
ff57966fcdf00583774e761327
Successfully built pyspark
Installing collected packages: pyspark
Successfully installed pyspark-3.4.0
```

Importing required Libraries

[5]:
```
%matplotlib inline
from pyspark import SparkConf
from pyspark.sql import SparkSession
from pyspark.sql import SQLContext
from pyspark.sql.types import StructType, StructField, StringType, MapType
import pyspark.sql.functions as F
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import json
from pyspark.ml.feature import Tokenizer
from pyspark.ml import Pipeline
from pyspark.ml.classification import NaiveBayes
from pyspark.ml.evaluation import BinaryClassificationEvaluator
from pyspark.ml.feature import Tokenizer, StopWordsRemover, CountVectorizer
from pyspark.sql.functions import lower, regexp_replace
```

[6]:
```
sns.set_theme(style="whitegrid", palette="pastel")
```

Creating Spark Session

[7]:
```
# conf = SparkConf().set("spark.kryoserializer.buffer.max", "4g")
spark = SparkSession.builder.getOrCreate()
spark_context = spark.sparkContext
sqlContext = SQLContext(spark_context)
```

```
/usr/local/lib/python3.10/dist-packages/pyspark/sql/context.py:112:
FutureWarning: Deprecated in 3.0.0. Use SparkSession.builder.getOrCreate()
instead.
  warnings.warn(
```

1 Data Loading

1.1 Reviews Dataset

[43]:
```
reviews = spark.read.json('yelp_academic_dataset_review.json')
```

[44]:
```
reviews.show(5)
```

```
+--------------------+----+--------------------+-----+--------------------+-----
+--------------------+------+--------------------+
|         business_id|cool|                date|funny|           review_id|stars|
text|useful|             user_id|
+--------------------+----+--------------------+-----+--------------------+-----
+--------------------+------+--------------------+
|XQfwVwDr-v0ZS3_Cb…|   0|2018-07-07 22:09:11|    0|KU_O5udG6zpxOg-Vc…|
3.0|If you decide to …|     0|mh_-eMZ6K5RLWhZyI…|
|7ATYjTIgM3jUlt4UM…|   1|2012-01-03 15:28:18|    0|BiTunyQ73aT9WBnpR…|
5.0|I've taken a lot …|     1|OyoGAe7OKpv6SyGZT…|
```

```
|YjUWPpI6HXG530lwP…|   0|2014-02-05 20:30:30|    0|saUsX_uimxRlCVr67…|
3.0|Family diner. Had…|     0|8g_iMtfSiwikVnbP2…|
|kxX2SOes4o-D3ZQBk…|   1|2015-01-04 00:01:03|    0|AqPFMleE6RsU23_au…|
5.0|Wow!  Yummy, diff…|     1|_7bHUi9Uuf5__HHc_…|
|e4Vwtrqf-wpJfwesg…|   1|2017-01-14 20:54:15|    0|Sx8TMOWLNuJBWer-0…|
4.0|Cute interior and…|     1|bcjbaE6dDog4jkNY9…|
+--------------------+----+--------------------+-----+--------------------+-----
+--------------------+------+--------------------+
only showing top 5 rows
```

[45]:
```
columns_to_drop = ['cool','funny','average_stars']
reviews = reviews.drop(*columns_to_drop)
```

1.2 Businesses Dataset

[8]:
```
businesses = spark.read.json('yelp_academic_dataset_business.json')
```

[12]:
```
businesses.show(5)
```

```
+--------------------+--------------------+--------------------+---------------
----+-------------+--------------------+-------+----------+-----------+--------
-----------+----------+-----------+-----+-----+
|             address|          attributes|         business_id|
categories|         city|               hours|is_open|  latitude|   longitude|
name|postal_code|review_count|stars|state|
+--------------------+--------------------+--------------------+---------------
----+-------------+--------------------+-------+----------+-----------+--------
-----------+----------+-----------+-----+-----+
|1616 Chapala St, …|{null, null, null…|Pns2l4eNsfO8kk83d…|Doctors,
Traditio…|Santa Barbara|                null|
0|34.4266787|-119.7111968|Abby Rappoport, L…|      93101|           7|  5.0|
CA|
|87 Grasso Plaza S…|{null, null, null…|mpf3x-BjTdTEA3yCZ…|Shipping
Centers,…|       Affton|{8:0-18:30, 0:0-0…|      1| 38.551126|  -90.335695|
The UPS Store|      63123|          15|  3.0|   MO|
|5255 E Broadway Blvd|{null, null, null…|tUFrWirKiKi_TAnsV…|Department
Stores…|       Tucson|{8:0-23:0, 8:0-22…|      0| 32.223236| -110.880452|
Target|      85711|          22|  3.5|   AZ|
|         935 Race St|{null, null, u'no…|MTSW4McQd7CbVtyjq…|Restaurants,
Food…| Philadelphia|{7:0-21:0, 7:0-20…|      1|39.9555052| -75.1555641|  St
Honore Pastries|      19107|          80|  4.0|   PA|
|       101 Walnut St|{null, null, null…|mWMc6_wTdE0EUBKIG…|Brewpubs,
Breweri…|   Green Lane|{12:0-22:0, null,…|      1|40.3381827|
-75.4716585|Perkiomen Valley …|      18054|          13|  4.5|   PA|
+--------------------+--------------------+--------------------+---------------
----+-------------+--------------------+-------+----------+-----------+--------
-----------+----------+-----------+-----+-----+
```

only showing top 5 rows

[9]:
```
#change name for starts to avoid duplicates
businesses=businesses.withColumnRenamed("stars", "Restaurant_stars")
businesses=businesses.withColumnRenamed("name", "Restaurant_name")
businesses=businesses.withColumnRenamed("review_count",␣
 ↪"Restaurant_review_count")

#only keep restaurant in business table for this project
businesses=businesses.filter(F.col('categories').rlike('Restaurants'))
```

[14]:
```
businesses.count()
```

[14]: 52268

[10]:
```
columns_to_drop = ['postal_code']
businesses = businesses.drop(*columns_to_drop)
```

[16]:
```
# businesses = businesses.filter(col("categories").contains("Restaurants"))
# businesses = businesses.filter(F.col("is_open").contains("1"))
```

1.3 Users Dataset

[17]:
```
users = spark.read.json('yelp_academic_dataset_user.json')
```

[18]:
```
users.show(5)
```

```
+------------+--------------+--------------+---------------+-------------+-
-------------+--------------+--------------+----------------+--------------
-+-----------------+----------------+-----+--------------------+----+--------
----------+-----+------+-----------+------+--------------------+-------------
-----+
|average_stars|compliment_cool|compliment_cute|compliment_funny|compliment_hot|c
ompliment_list|compliment_more|compliment_note|compliment_photos|compliment_plai
n|compliment_profile|compliment_writer| cool|               elite|fans|
friends|funny|  name|review_count|useful|             user_id|
yelping_since|
+------------+--------------+--------------+---------------+-------------+-
-------------+--------------+--------------+----------------+--------------
-+-----------------+----------------+-----+--------------------+----+--------
----------+-----+------+-----------+------+--------------------+-------------
-----+
|         3.91|           467|            56|            467|          250|
18|            65|           232|             180|            844|
55|             239| 5994|               2007| 267|NSCy54eWehBJyZdG2…|
1259|Walker|         585|  7217|qVc8ODYU5SZjKXVBg…|2007-01-25 16:47:26|
|         3.74|          3131|           157|           3131|         1145|
```

```
251|            264|           1847|             1946|            7054|
184|
1521|27281|2009,2010,2011,20…|3138|ueRPE0CX75ePGMqOF…|13066|Daniel|
4333| 43091|j14WgRoU_-2ZE1aw1…|2009-01-25 04:35:42|
|         3.32|            119|             17|             119|            89|
3|             13|             66|               18|              96|
10|               35| 1003|2009,2010,2011,20…|  52|LuO3Bn4f3rlhyHIaN…| 1010|
Steph|         665|  2086|2WnXYQFK0hXEoTxPt…|2008-07-25 10:41:00|
|         4.27|             26|              6|              26|            24|
2|              4|             12|               9|              16|
1|               10|  299|      2009,2010,2011|  28|enx1vVPnfdNUdPho6…|  330|
Gwen|         224|   512|SZDeASXq7o05mMNLs…|2005-11-29 04:38:33|
|         3.54|              0|              0|               0|             1|
0|              1|              1|               0|               1|
0|               0|    7|                    |   1|PBK4q9KEEBHhFvSXC…|   15|
Karen|          79|    29|hA5lMy-EnncsH4JoR…|2007-01-05 19:40:59|
+-------------+---------------+---------------+-----------------+--------------+-
--------------+---------------+---------------+-----------------+---------------
-+------------------+-----------------+-----+--------------------+----+---------
-----------+-----+------+------------+------+--------------------+--------------
-----+
only showing top 5 rows
```

[19]:
```
columns_to_drop = ['elite','useful','yelping_since']
users = users.drop(*columns_to_drop)
```

[20]:
```
users=users.withColumnRenamed("name", "user_name")
```

1.4 Exploration and Data Joining

[21]:
```
reviews.printSchema()
```

```
root
 |-- business_id: string (nullable = true)
 |-- date: string (nullable = true)
 |-- review_id: string (nullable = true)
 |-- stars: double (nullable = true)
 |-- text: string (nullable = true)
 |-- useful: long (nullable = true)
 |-- user_id: string (nullable = true)
```

[22]:
```
businesses.printSchema()
```

```
root
 |-- address: string (nullable = true)
 |-- attributes: struct (nullable = true)
 |    |-- AcceptsInsurance: string (nullable = true)
```

```
|    |-- AgesAllowed: string (nullable = true)
|    |-- Alcohol: string (nullable = true)
|    |-- Ambience: string (nullable = true)
|    |-- BYOB: string (nullable = true)
|    |-- BYOBCorkage: string (nullable = true)
|    |-- BestNights: string (nullable = true)
|    |-- BikeParking: string (nullable = true)
|    |-- BusinessAcceptsBitcoin: string (nullable = true)
|    |-- BusinessAcceptsCreditCards: string (nullable = true)
|    |-- BusinessParking: string (nullable = true)
|    |-- ByAppointmentOnly: string (nullable = true)
|    |-- Caters: string (nullable = true)
|    |-- CoatCheck: string (nullable = true)
|    |-- Corkage: string (nullable = true)
|    |-- DietaryRestrictions: string (nullable = true)
|    |-- DogsAllowed: string (nullable = true)
|    |-- DriveThru: string (nullable = true)
|    |-- GoodForDancing: string (nullable = true)
|    |-- GoodForKids: string (nullable = true)
|    |-- GoodForMeal: string (nullable = true)
|    |-- HairSpecializesIn: string (nullable = true)
|    |-- HappyHour: string (nullable = true)
|    |-- HasTV: string (nullable = true)
|    |-- Music: string (nullable = true)
|    |-- NoiseLevel: string (nullable = true)
|    |-- Open24Hours: string (nullable = true)
|    |-- OutdoorSeating: string (nullable = true)
|    |-- RestaurantsAttire: string (nullable = true)
|    |-- RestaurantsCounterService: string (nullable = true)
|    |-- RestaurantsDelivery: string (nullable = true)
|    |-- RestaurantsGoodForGroups: string (nullable = true)
|    |-- RestaurantsPriceRange2: string (nullable = true)
|    |-- RestaurantsReservations: string (nullable = true)
|    |-- RestaurantsTableService: string (nullable = true)
|    |-- RestaurantsTakeOut: string (nullable = true)
|    |-- Smoking: string (nullable = true)
|    |-- WheelchairAccessible: string (nullable = true)
|    |-- WiFi: string (nullable = true)
|-- business_id: string (nullable = true)
|-- categories: string (nullable = true)
|-- city: string (nullable = true)
|-- hours: struct (nullable = true)
|    |-- Friday: string (nullable = true)
|    |-- Monday: string (nullable = true)
|    |-- Saturday: string (nullable = true)
|    |-- Sunday: string (nullable = true)
|    |-- Thursday: string (nullable = true)
|    |-- Tuesday: string (nullable = true)
```

```
 |    |-- Wednesday: string (nullable = true)
 |-- is_open: long (nullable = true)
 |-- latitude: double (nullable = true)
 |-- longitude: double (nullable = true)
 |-- Restaurant_name: string (nullable = true)
 |-- Restaurant_review_count: long (nullable = true)
 |-- Restaurant_stars: double (nullable = true)
 |-- state: string (nullable = true)
```

[23]:
```
users.printSchema()
```

```
root
 |-- average_stars: double (nullable = true)
 |-- compliment_cool: long (nullable = true)
 |-- compliment_cute: long (nullable = true)
 |-- compliment_funny: long (nullable = true)
 |-- compliment_hot: long (nullable = true)
 |-- compliment_list: long (nullable = true)
 |-- compliment_more: long (nullable = true)
 |-- compliment_note: long (nullable = true)
 |-- compliment_photos: long (nullable = true)
 |-- compliment_plain: long (nullable = true)
 |-- compliment_profile: long (nullable = true)
 |-- compliment_writer: long (nullable = true)
 |-- cool: long (nullable = true)
 |-- fans: long (nullable = true)
 |-- friends: string (nullable = true)
 |-- funny: long (nullable = true)
 |-- user_name: string (nullable = true)
 |-- review_count: long (nullable = true)
 |-- user_id: string (nullable = true)
```

[24]:
```
df = reviews.join(businesses,on ='business_id', how = 'inner')
df = df.join(users,on ='user_id', how = 'inner')
```

[25]:
```
# df.show(3) # Takes a lot of time to execute
```

2 Data Cleaning

[26]:
```
df.count()
```

[26]: 4724464

[27]:
```
df.printSchema()
```

```
root
```

```
|-- user_id: string (nullable = true)
|-- business_id: string (nullable = true)
|-- date: string (nullable = true)
|-- review_id: string (nullable = true)
|-- stars: double (nullable = true)
|-- text: string (nullable = true)
|-- useful: long (nullable = true)
|-- address: string (nullable = true)
|-- attributes: struct (nullable = true)
|    |-- AcceptsInsurance: string (nullable = true)
|    |-- AgesAllowed: string (nullable = true)
|    |-- Alcohol: string (nullable = true)
|    |-- Ambience: string (nullable = true)
|    |-- BYOB: string (nullable = true)
|    |-- BYOBCorkage: string (nullable = true)
|    |-- BestNights: string (nullable = true)
|    |-- BikeParking: string (nullable = true)
|    |-- BusinessAcceptsBitcoin: string (nullable = true)
|    |-- BusinessAcceptsCreditCards: string (nullable = true)
|    |-- BusinessParking: string (nullable = true)
|    |-- ByAppointmentOnly: string (nullable = true)
|    |-- Caters: string (nullable = true)
|    |-- CoatCheck: string (nullable = true)
|    |-- Corkage: string (nullable = true)
|    |-- DietaryRestrictions: string (nullable = true)
|    |-- DogsAllowed: string (nullable = true)
|    |-- DriveThru: string (nullable = true)
|    |-- GoodForDancing: string (nullable = true)
|    |-- GoodForKids: string (nullable = true)
|    |-- GoodForMeal: string (nullable = true)
|    |-- HairSpecializesIn: string (nullable = true)
|    |-- HappyHour: string (nullable = true)
|    |-- HasTV: string (nullable = true)
|    |-- Music: string (nullable = true)
|    |-- NoiseLevel: string (nullable = true)
|    |-- Open24Hours: string (nullable = true)
|    |-- OutdoorSeating: string (nullable = true)
|    |-- RestaurantsAttire: string (nullable = true)
|    |-- RestaurantsCounterService: string (nullable = true)
|    |-- RestaurantsDelivery: string (nullable = true)
|    |-- RestaurantsGoodForGroups: string (nullable = true)
|    |-- RestaurantsPriceRange2: string (nullable = true)
|    |-- RestaurantsReservations: string (nullable = true)
|    |-- RestaurantsTableService: string (nullable = true)
|    |-- RestaurantsTakeOut: string (nullable = true)
|    |-- Smoking: string (nullable = true)
|    |-- WheelchairAccessible: string (nullable = true)
|    |-- WiFi: string (nullable = true)
```

```
|-- categories: string (nullable = true)
|-- city: string (nullable = true)
|-- hours: struct (nullable = true)
|    |-- Friday: string (nullable = true)
|    |-- Monday: string (nullable = true)
|    |-- Saturday: string (nullable = true)
|    |-- Sunday: string (nullable = true)
|    |-- Thursday: string (nullable = true)
|    |-- Tuesday: string (nullable = true)
|    |-- Wednesday: string (nullable = true)
|-- is_open: long (nullable = true)
|-- latitude: double (nullable = true)
|-- longitude: double (nullable = true)
|-- Restaurant_name: string (nullable = true)
|-- Restaurant_review_count: long (nullable = true)
|-- Restaurant_stars: double (nullable = true)
|-- state: string (nullable = true)
|-- average_stars: double (nullable = true)
|-- compliment_cool: long (nullable = true)
|-- compliment_cute: long (nullable = true)
|-- compliment_funny: long (nullable = true)
|-- compliment_hot: long (nullable = true)
|-- compliment_list: long (nullable = true)
|-- compliment_more: long (nullable = true)
|-- compliment_note: long (nullable = true)
|-- compliment_photos: long (nullable = true)
|-- compliment_plain: long (nullable = true)
|-- compliment_profile: long (nullable = true)
|-- compliment_writer: long (nullable = true)
|-- cool: long (nullable = true)
|-- fans: long (nullable = true)
|-- friends: string (nullable = true)
|-- funny: long (nullable = true)
|-- user_name: string (nullable = true)
|-- review_count: long (nullable = true)
```

[28]:
```
df.describe()
```

[28]:
```
DataFrame[summary: string, user_id: string, business_id: string, date: string,
review_id: string, stars: string, text: string, useful: string, address: string,
categories: string, city: string, is_open: string, latitude: string, longitude:
string, Restaurant_name: string, Restaurant_review_count: string,
Restaurant_stars: string, state: string, average_stars: string, compliment_cool:
string, compliment_cute: string, compliment_funny: string, compliment_hot:
string, compliment_list: string, compliment_more: string, compliment_note:
string, compliment_photos: string, compliment_plain: string, compliment_profile:
```

```
string, compliment_writer: string, cool: string, fans: string, friends: string,
funny: string, user_name: string, review_count: string]
```

2.1 Dropping Columns

[29]:
```
# friends do not provide any information and Restaurant_starts and hours are␣
↪redudant information
columns_drop = ['friends', 'Restaurant_stars', 'hours']
df = df.drop(*columns_drop)
```

[30]:
```
columns_drop = [
    'compliment_cute',
    'compliment_funny',
    'compliment_hot',
    'compliment_list',
    'compliment_more',
    'compliment_note',
    'compliment_photos',
    'compliment_plain',
    'compliment_profile',
    'compliment_writer'
   ]
df = df.drop(*columns_drop)
```

2.2 Checking Null Values

[31]:
```
df.dropna().count()
```

[31]:
```
4719396
```

[32]:
```
df = df.dropna()
```

2.3 Type Casting

[33]:
```
import datetime
from pyspark.sql.functions import month
```

[34]:
```
df = df.withColumn('date', F.to_date(F.unix_timestamp('date', 'yyyy-MM-dd HH:mm:
↪ss').cast('timestamp')))
df = df.withColumn('year', F.year(F.col('date')))
df = df.withColumn('stars', df['stars'].cast('int'))
```

[35]:
```
df.dtypes
```

[35]:
```
[('user_id', 'string'),
 ('business_id', 'string'),
 ('date', 'date'),
```

```
 ('review_id', 'string'),
 ('stars', 'int'),
 ('text', 'string'),
 ('useful', 'bigint'),
 ('address', 'string'),
 ('attributes',
  'struct<AcceptsInsurance:string,AgesAllowed:string,Alcohol:string,Ambience:str
ing,BYOB:string,BYOBCorkage:string,BestNights:string,BikeParking:string,Business
AcceptsBitcoin:string,BusinessAcceptsCreditCards:string,BusinessParking:string,B
yAppointmentOnly:string,Caters:string,CoatCheck:string,Corkage:string,DietaryRes
trictions:string,DogsAllowed:string,DriveThru:string,GoodForDancing:string,GoodF
orKids:string,GoodForMeal:string,HairSpecializesIn:string,HappyHour:string,HasTV
:string,Music:string,NoiseLevel:string,Open24Hours:string,OutdoorSeating:string,
RestaurantsAttire:string,RestaurantsCounterService:string,RestaurantsDelivery:st
ring,RestaurantsGoodForGroups:string,RestaurantsPriceRange2:string,RestaurantsRe
servations:string,RestaurantsTableService:string,RestaurantsTakeOut:string,Smoki
ng:string,WheelchairAccessible:string,WiFi:string>'),
 ('categories', 'string'),
 ('city', 'string'),
 ('is_open', 'bigint'),
 ('latitude', 'double'),
 ('longitude', 'double'),
 ('Restaurant_name', 'string'),
 ('Restaurant_review_count', 'bigint'),
 ('state', 'string'),
 ('average_stars', 'double'),
 ('compliment_cool', 'bigint'),
 ('cool', 'bigint'),
 ('fans', 'bigint'),
 ('funny', 'bigint'),
 ('user_name', 'string'),
 ('review_count', 'bigint'),
 ('year', 'int')]
```

[35]:

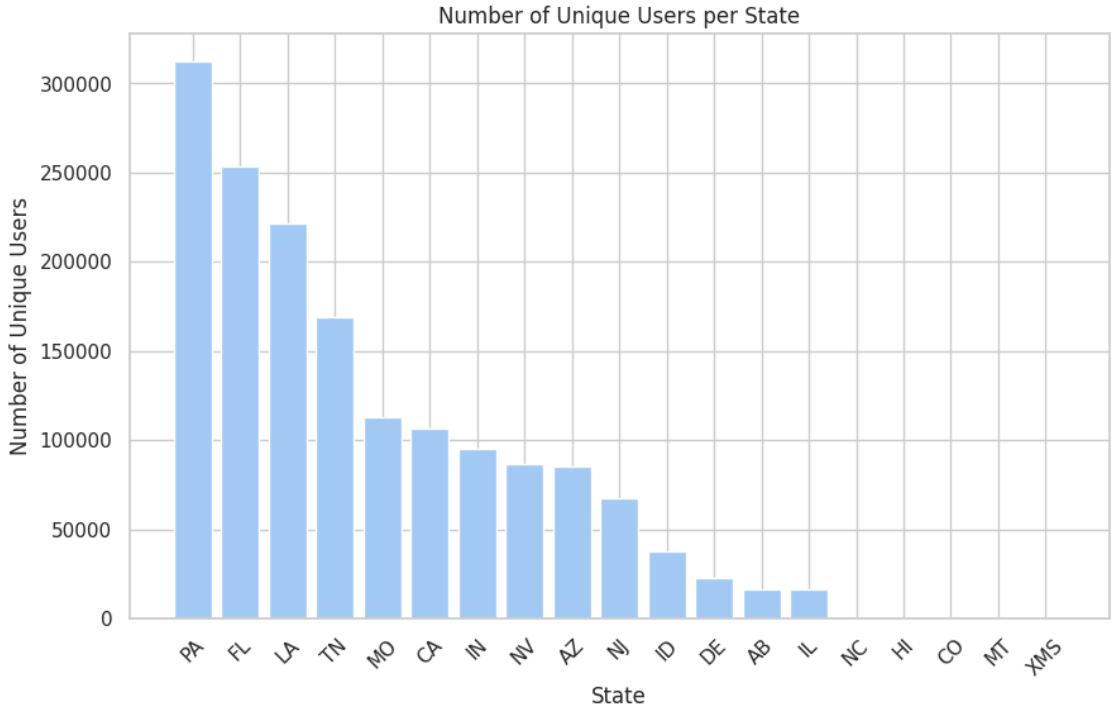
3 EDA

3.1 Unique Users By State

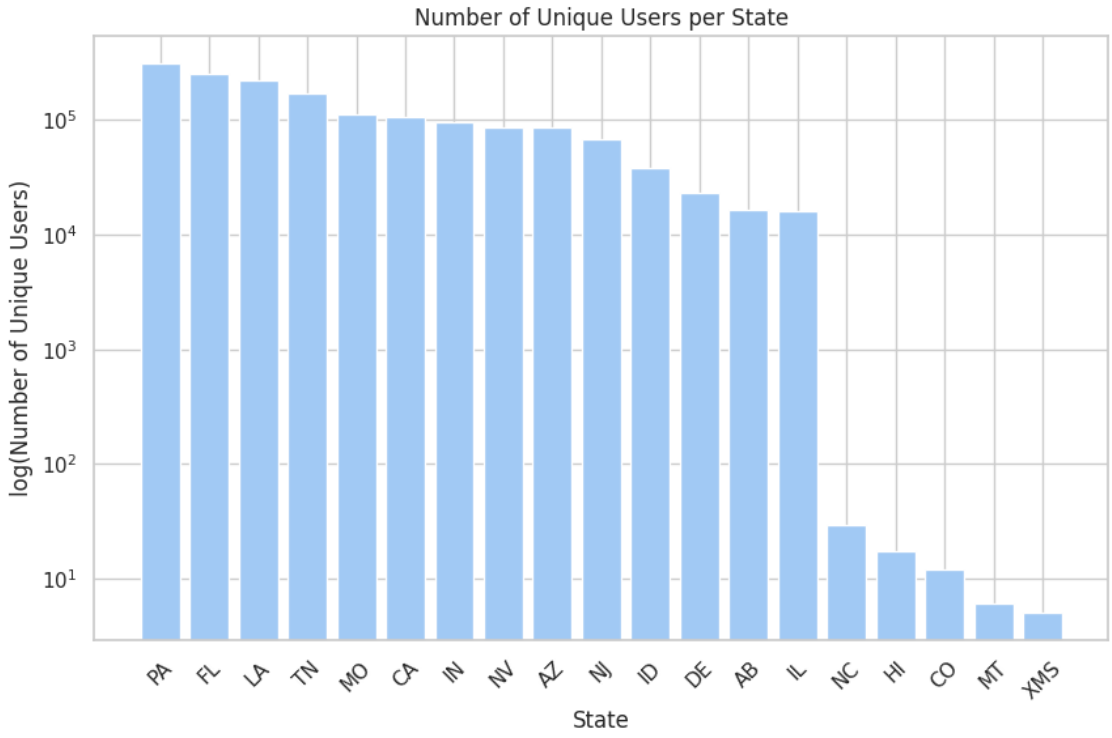
[36]:
```
unique_users_per_state = df.groupBy('state').agg(F.countDistinct('user_id').
  ↪alias('unique_users')).orderBy('unique_users', ascending=False)
pandas_df = unique_users_per_state.toPandas()
```

[37]:
```
# Plot the number of unique users per state
plt.figure(figsize=(10, 6))
plt.bar(pandas_df['state'], pandas_df['unique_users'])
```

```
plt.xlabel('State')
plt.ylabel('Number of Unique Users')
plt.title('Number of Unique Users per State')
plt.xticks(rotation=45)
plt.show()
```



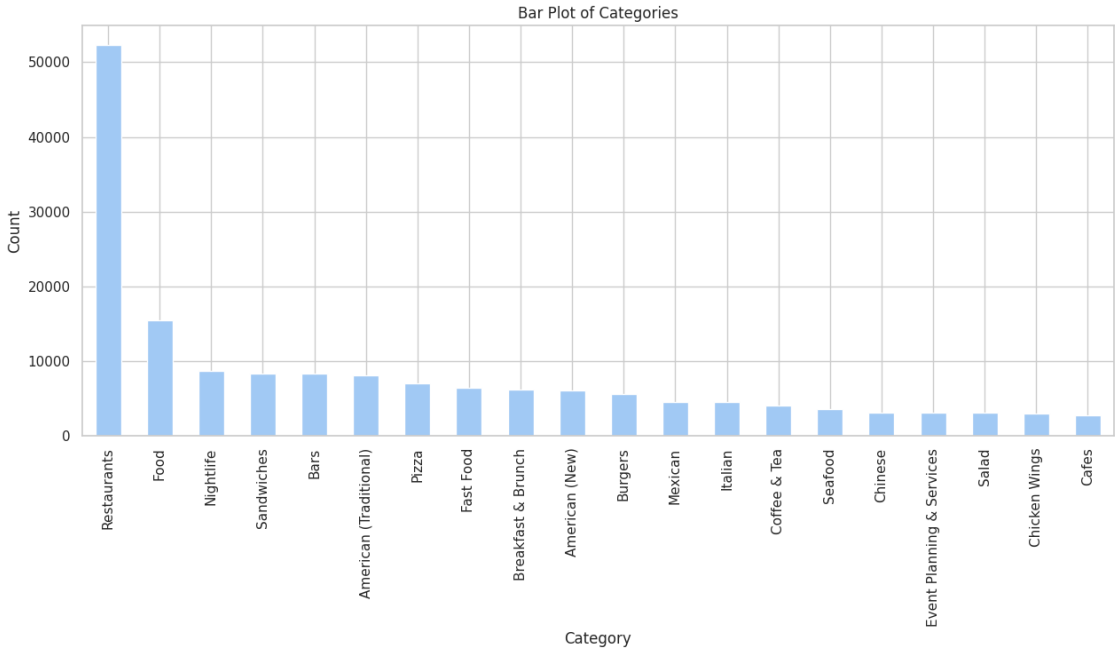
[38]:
```
plt.figure(figsize=(10, 6))
plt.bar(pandas_df['state'], pandas_df['unique_users'])
plt.yscale('log')
plt.xlabel('State')
plt.ylabel('log(Number of Unique Users)')
plt.title('Number of Unique Users per State')
plt.xticks(rotation=45)
plt.show()
```



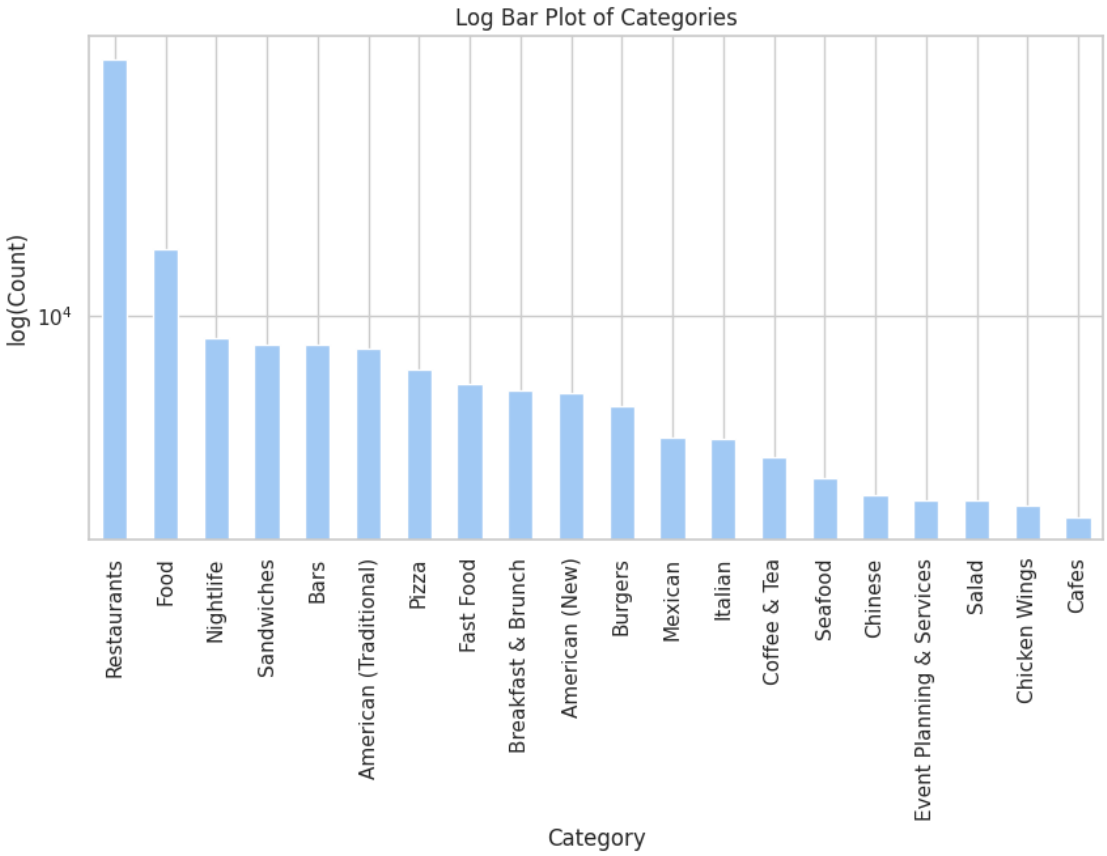
3.2 Categories Distribution

[ ]:
```
# Split the categories column into separate columns
categories_df = businesses.withColumn('categories', F.
 ↪split(businesses['categories'], ', '))

# Convert the DataFrame to a Pandas DataFrame
pandas_df = categories_df.toPandas()

# Flatten the categories into a single column
categories = [item for sublist in pandas_df['categories'] for item in sublist]

# Create a Pandas DataFrame with the category counts
category_counts = pd.Series(categories).value_counts()
```

[34]:
```
# Create top 20 categories bar plot
plt.figure(figsize=(15, 6))
category_counts[:20].plot(kind='bar')
plt.xlabel('Category')
plt.ylabel('Count')
plt.title('Bar Plot of Categories')
plt.xticks(rotation=90)
plt.show()
```



[33]:
```
# Create top 20 categories bar plot (Log Plot)
plt.figure(figsize=(10, 5))
category_counts[:20].plot(kind='bar')
plt.yscale('log')
plt.xlabel('Category')
plt.ylabel('log(Count)')
plt.title('Log Bar Plot of Categories')
plt.xticks(rotation=90)
plt.show()
```



[20]:
```
# Unique Categories
len(set(categories))
```

[20]: 728

[14]:
```
category_counts
```

[14]:
```
Restaurants                 52268
Food                        15472
Nightlife                    8723
Sandwiches                   8366
Bars                         8337
                            ...
Home Theatre Installation       1
Homeowner Association           1
Kids Hair Salons                1
Calabrian                       1
Gemstones & Minerals            1
Length: 728, dtype: int64
```

3.3 Open Businesses

[ ]:
```
# businesses.select(F.col("city")).distinct().collect()
```

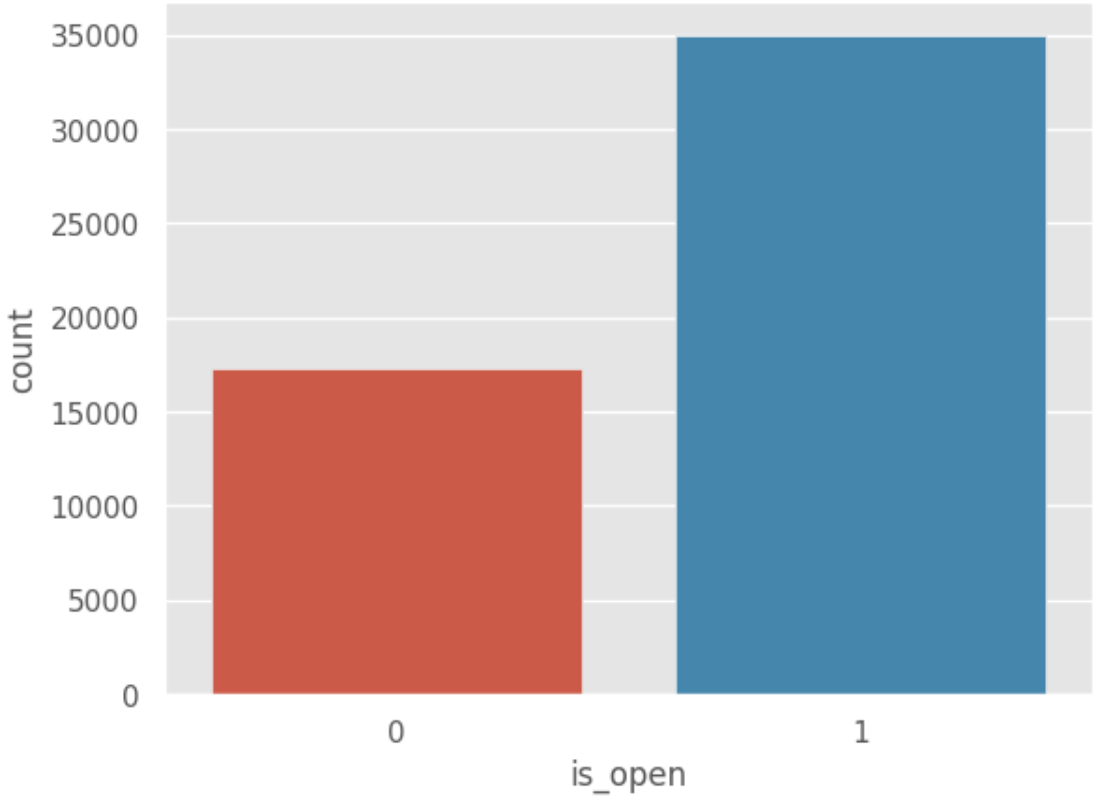
[40]:
```
businesses_open = businesses.groupBy('is_open').count().toPandas()
businesses_open
```

[40]:

	is_open	count
0	0	17281
1	1	34987

[41]:
```
# sns.set_style('white')
plt.style.use('ggplot')
sns.barplot(x='is_open', y='count', data=businesses_open)
```

[41]: <Axes: xlabel='is_open', ylabel='count'>



[41]:

3.3.1 Selecting Only Open Businesses

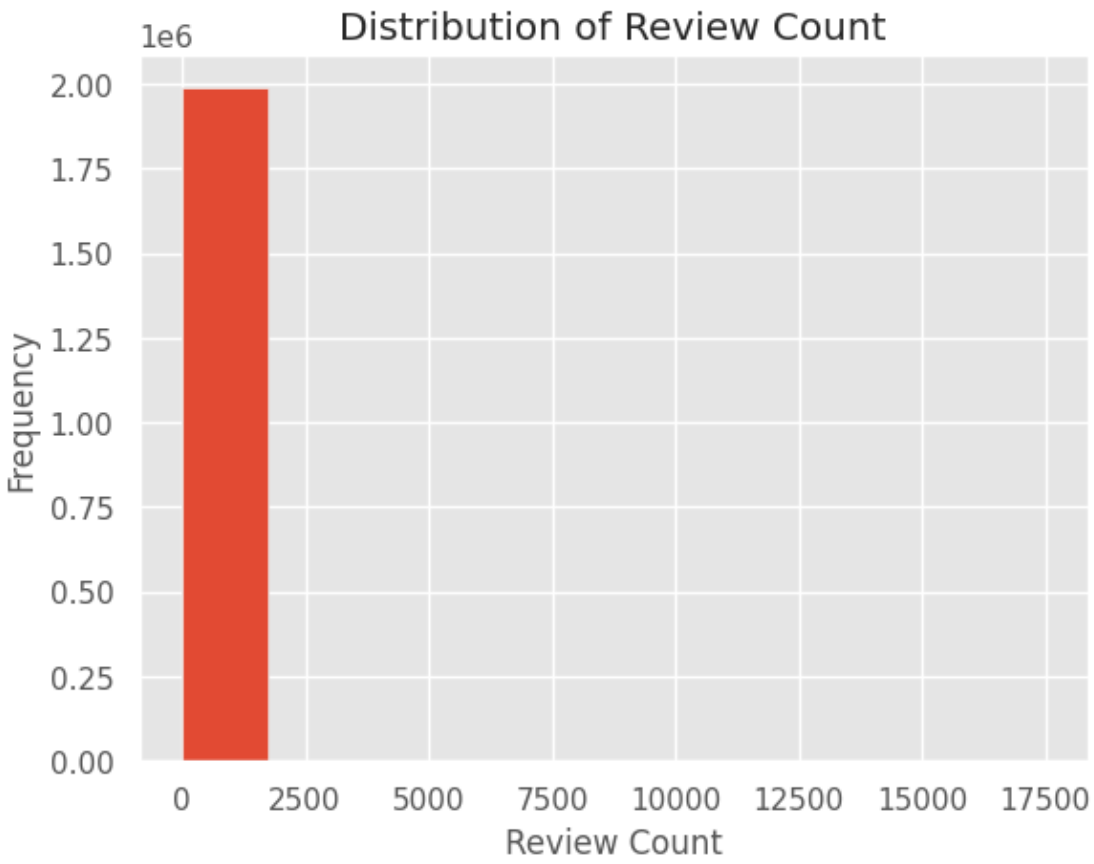
[42]:
```
df = df.filter(F.col("is_open").contains("1"))
```

##Distribution of review_count:

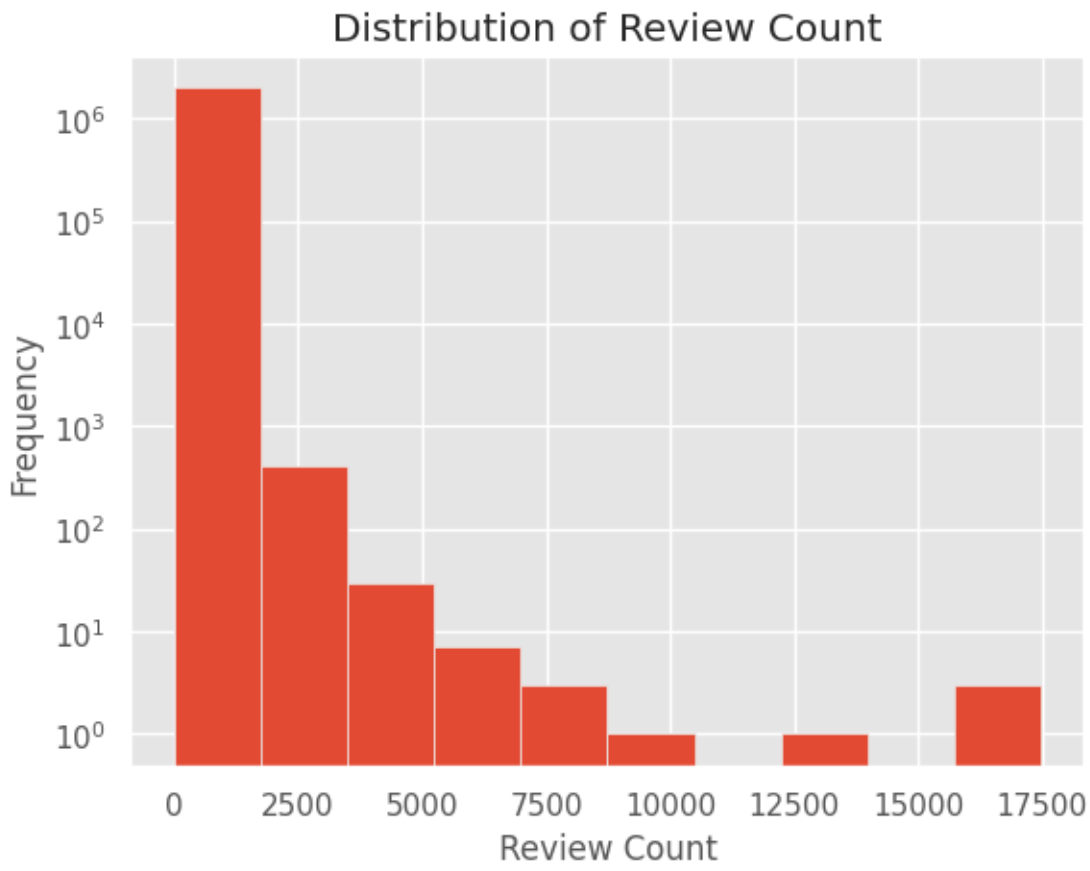
[43]:
```
import matplotlib.pyplot as plt
import pyspark.sql.functions as F

users.select("review_count").groupBy().agg(F.mean("review_count").
 ↪alias("mean_review_count"),
                                           F.max("review_count").
 ↪alias("max_review_count"),
                                           F.min("review_count").
 ↪alias("min_review_count")).show()
```

```
+------------------+----------------+----------------+
| mean_review_count|max_review_count|min_review_count|
+------------------+----------------+----------------+
|23.394409267683386|           17473|               0|
+------------------+----------------+----------------+
```

[44]:
```
users.select("review_count").toPandas().hist()
plt.xlabel("Review Count")
plt.ylabel("Frequency")
plt.title("Distribution of Review Count")
plt.show()
```



[45]:
```
users.select("review_count").toPandas().hist()
plt.yscale('log')
plt.xlabel("Review Count")
plt.ylabel("Frequency")
plt.title("Distribution of Review Count")
plt.show()
```



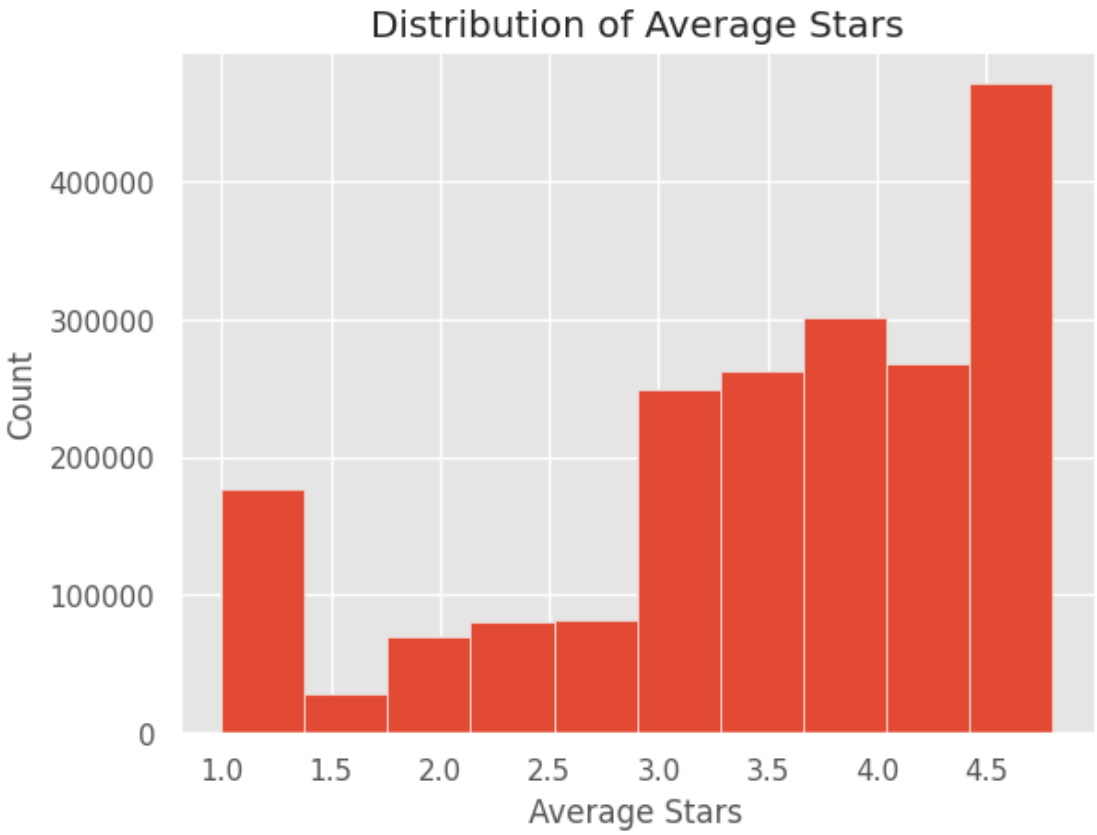
##Top 10 users with the most reviews:

[46]:
```
users.groupBy("user_name", "user_id").agg(F.sum("review_count").
  ↪alias("total_reviews")).orderBy(F.desc("total_reviews")).show(10)
```

```
+---------+--------------------+-------------+
|user_name|             user_id|total_reviews|
+---------+--------------------+-------------+
|      Fox|Hi10sGSZNxQH3NLyW…|        17473|
|   Victor|8k3aO-mPeyhbR5HUu…|        16978|
|    Bruce|hWDybu_KvYLSdEFzG…|        16567|
|    Shila|RtGqdDBvvBCjcu5dU…|        12868|
|      Kim|P5bUL3Engv-2z6kKo…|         9941|
|   Nijole|nmdkHL2JKFx55T3nq…|         8363|
|  Vincent|bQCHF5rn5lMI9c5kE…|         8354|
|   George|8RcEwGrFIgkt9WQ35…|         7738|
|  Kenneth|Xwnf20FKuikiHcSpc…|         6766|
| Jennifer|CxDOIDnH8gp9KXzpB…|         6679|
+---------+--------------------+-------------+
only showing top 10 rows
```

##**Histograms for average stars**

[47]:
```
histogram = users.select('average_stars').rdd.flatMap(lambda x: x).histogram(20)
plt.hist(histogram[0][:-1], weights=histogram[1])
plt.title('Distribution of Average Stars')
plt.xlabel('Average Stars')
plt.ylabel('Count')
plt.show()
```



3.4 Users VS Businesses

[35]:
```
selected_df = businesses.select('business_id', 'Restaurant_name',␣
 ↪'Restaurant_stars', 'Restaurant_review_count', 'state')
```

[36]:
```
filtered_df = businesses.filter(businesses['state'] == 'PA')
```

[39]:
```
from pyspark.sql.functions import avg, count, max, min
```

```
grouped_df = businesses.groupBy('state').agg(avg('Restaurant_stars'),␣
 ↪count('business_id'))
max_review_count = businesses.groupBy('state').
 ↪agg(max('Restaurant_review_count'))
```

[40]:
```
grouped_df
```

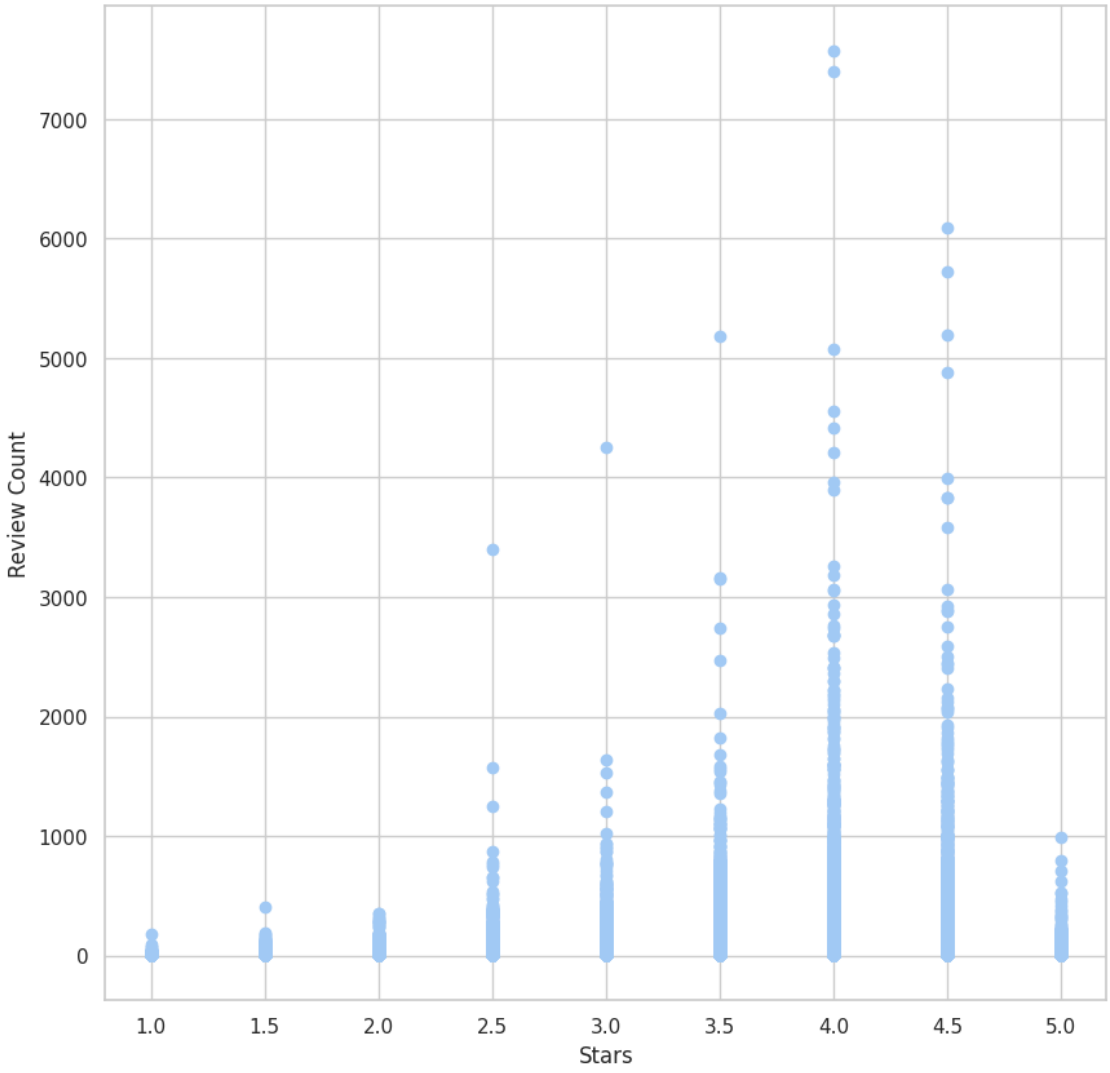
[40]:
```
DataFrame[state: string, avg(Restaurant_stars): double, count(business_id):
bigint]
```

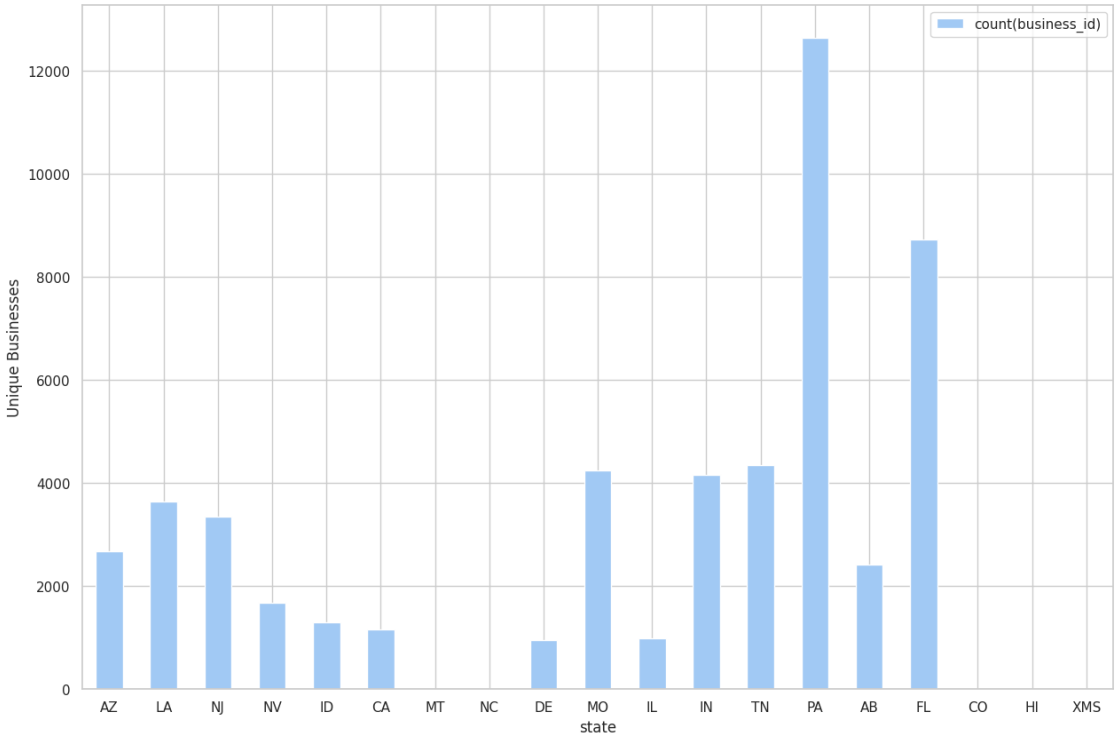
[41]:
```
max_review_count
```

[41]:
```
DataFrame[state: string, max(Restaurant_review_count): bigint]
```

[46]:
```
joined_df = reviews.join(businesses, 'business_id', 'inner')
```

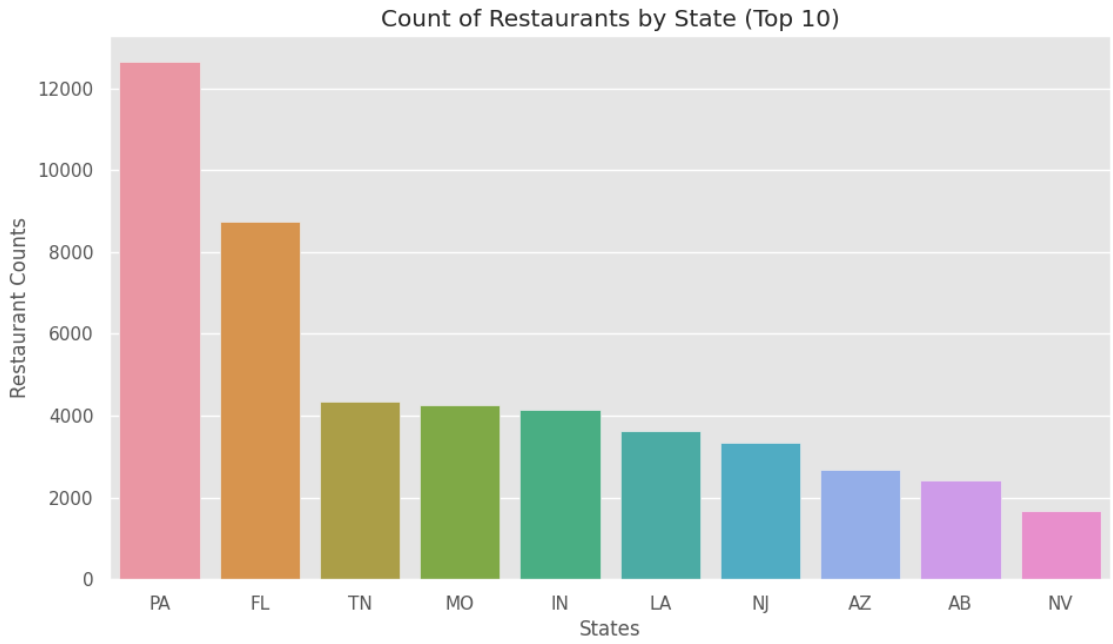
[48]:
```
import matplotlib.pyplot as plt
import pandas as pd

selected_df_pd = selected_df.toPandas()

plt.figure(figsize=(10,10))
plt.scatter(selected_df_pd['Restaurant_stars'],␣
 ↪selected_df_pd['Restaurant_review_count'])
plt.xlabel('Stars')
plt.ylabel('Review Count')
plt.show()

grouped_df_pd = grouped_df.toPandas()
# plt.figure(figsize=(10,10))
grouped_df_pd.plot.bar(x='state', y='count(business_id)', rot=0,␣
 ↪figsize=(15,10))
plt.ylabel('Unique Businesses')
plt.show()
```


Review Count
7000
6000
5000
4000
3000
2000
1000
0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Stars



3.5 Open businesses counts by state (Top 10)

[51]:
```
business_counts = businesses.groupBy('state').count().sort('count').
  ↪orderBy(['count'], ascending=[0]).toPandas()
plt.figure(figsize=(11,6))
states = business_counts['state'].values[:10]
counts = business_counts['count'].values[:10]
sns.barplot(x = states, y = counts)
plt.ylabel('Restaurant Counts')
plt.xlabel('States')
plt.title('Count of Restaurants by State (Top 10)')
plt.show()
```



[52]:
```
business_counts.head()
```

[52]:

	state	count
0	PA	12641
1	FL	8731
2	TN	4352
3	MO	4247
4	IN	4150

3.5.1 By average count

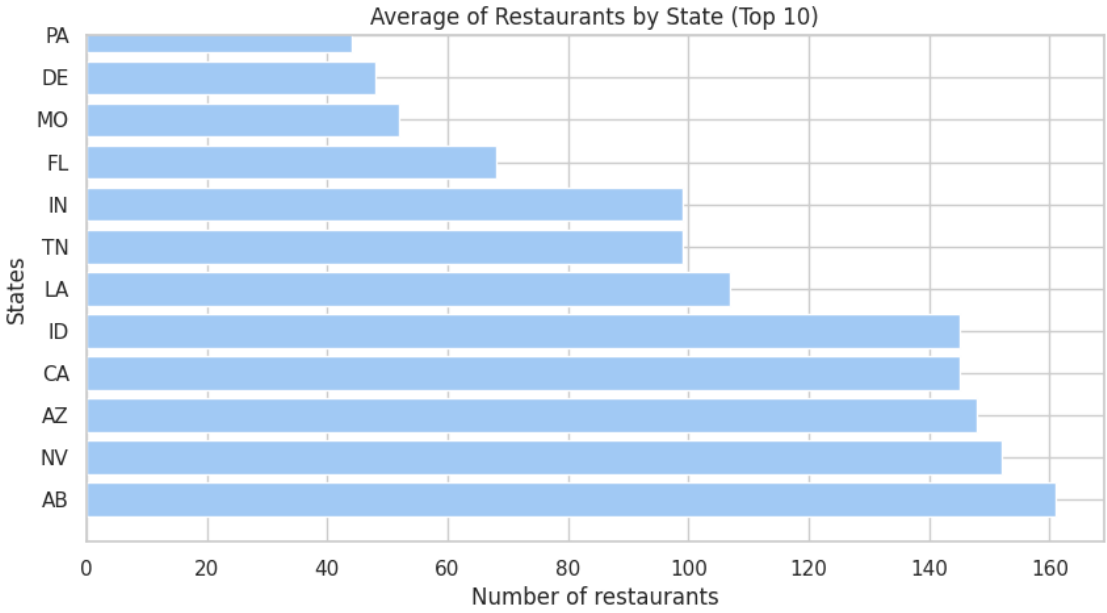
[50]:
```
state_avg = businesses.groupBy('city','state').count().groupBy('state').
 ↪agg({'count':'avg'}).sort(F.desc('avg(count)')).toPandas()
state_avg = state_avg.round(0)
state_avg.head()
```

[50]:

	state	avg(count)
0	AB	161.0
1	NV	152.0
2	AZ	148.0
3	CA	145.0
4	ID	145.0

[51]:
```
plt.figure(figsize=(10,5))
plt.barh(state_avg.state.values, state_avg['avg(count)'])
# sns.barplot(x=state_avg.state.values, y=state_avg['avg(count)'], orient='h')
```

```
plt.title("Average of Restaurants by State (Top 10)")
plt.xlabel("Number of restaurants")
plt.ylabel("States")
plt.ylim((-1,11))
plt.show()
```



3.5.2 By City

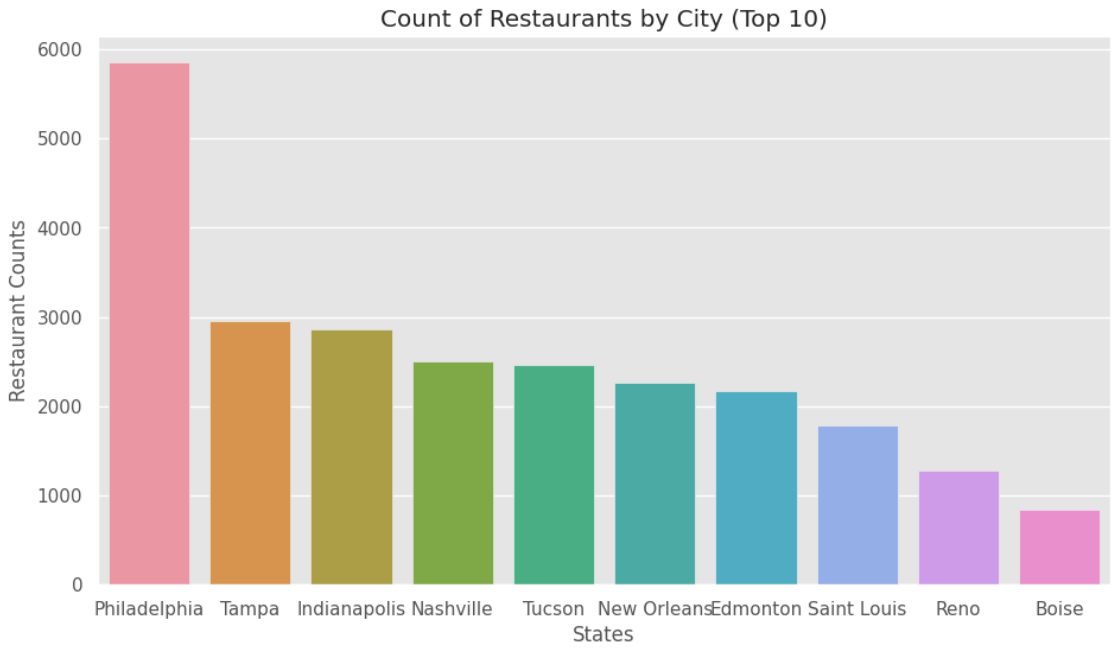
[55]:
```
business_counts_city = businesses.groupBy('city').count().sort('count').
↪orderBy(['count'], ascending=[0]).toPandas()
business_counts_city.head()
```

[55]:

	city	count
0	Philadelphia	5852
1	Tampa	2960
2	Indianapolis	2862
3	Nashville	2502
4	Tucson	2466

[56]:
```
plt.figure(figsize=(11,6))
states = business_counts_city['city'].values[:10]
counts = business_counts_city['count'].values[:10]
sns.barplot(x = states, y = counts)
plt.ylabel('Restaurant Counts')
plt.xlabel('States')
plt.title('Count of Restaurants by City (Top 10)')
```

```
plt.show()
```



Based on the information above, Pennsylvania state has the highest count of restaurants, and among the top 10 cities in the USA with the most restaurants, Philadelphia stands out significantly. Additionally, on average, Alberta has a greater number of restaurants per city compared to other states.

3.6 Reviews distribution over time

[57]:
```
df.count()
```

[57]: 3769692

[58]:
```
reviews_res = df.groupBy('year').count().orderBy('count').toPandas()
reviews_res = reviews_res.dropna()
reviews_res.head()
```

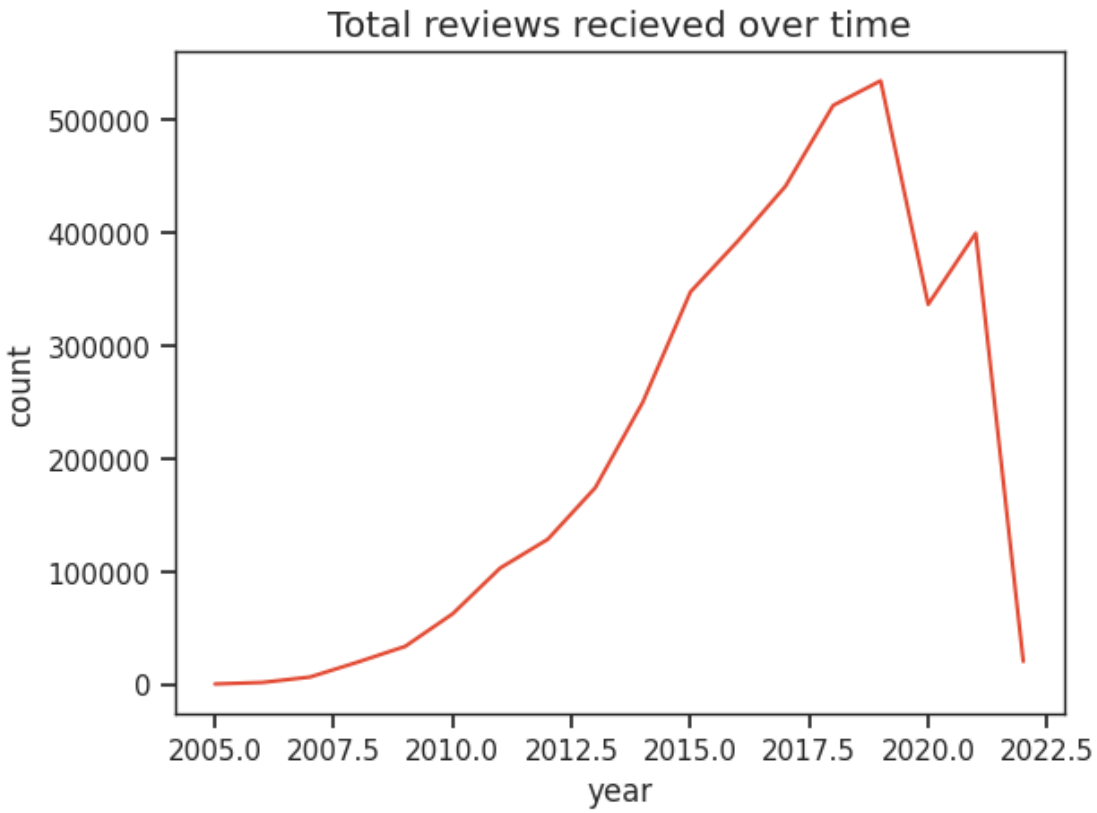
[58]:

	year	count
0	2005	399
1	2006	1757
2	2007	6553
3	2008	19742
4	2022	20445

[59]:
```
sns.set_style('ticks')
```

```
sns.lineplot(x="year", y="count", data=reviews_res).set_title('Total reviews␣
 ↪recieved over time')
```

[59]:
```
Text(0.5, 1.0, 'Total reviews recieved over time')
```



Based on the data provided in the dataset, it appears that 2017 had the highest number of received reviews. Additionally, there was exponential growth in the number of reviews from 2010 to 2016. One possibility is that there was an increase in the use of online review platforms, which may have made it easier for people to leave reviews. This could have resulted in a higher volume of reviews overall, and particularly in 2017 when the platform may have reached peak popularity.

However, we can observe a steep decrease in the reviews as of 2019, this could be because of COVID-19 when people stater of avoid going out and started to social distance.

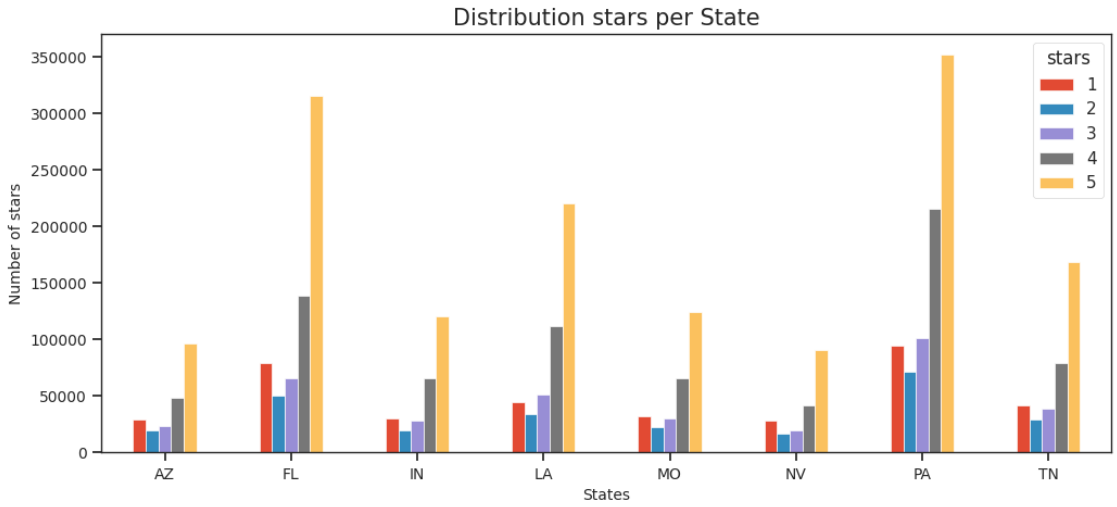
3.7 Stars per state

[60]:
```
state_df = df.groupBy('state', 'stars').count().orderBy('count').toPandas()
state_df['stars'] = state_df.stars.astype(int)
```

[61]:
```
top_restaurant = state_df.sort_values(by=['count'], ascending = False).head(50)
top_restaurant = top_restaurant.pivot(index = 'state', columns = 'stars',␣
 ↪values = 'count')
```

```
top_restaurant = top_restaurant.dropna()
```

[62]:
```
top_restaurant.plot.bar(figsize=(12,5))
plt.ylabel("Number of stars", fontsize = 10)
plt.xlabel("States", fontsize = 10)
plt.title("Distribution stars per State", fontsize = 15)
plt.xticks(size = 10)
plt.yticks(size = 10)
plt.xticks(rotation = 0)
plt.show()
```



According to the data, Pensilvania (PA) received the largest number of 5-star ratings, with Florida (FL) coming in second place. These results are consistent with the proportion of restaurants per state that we observed earlier.

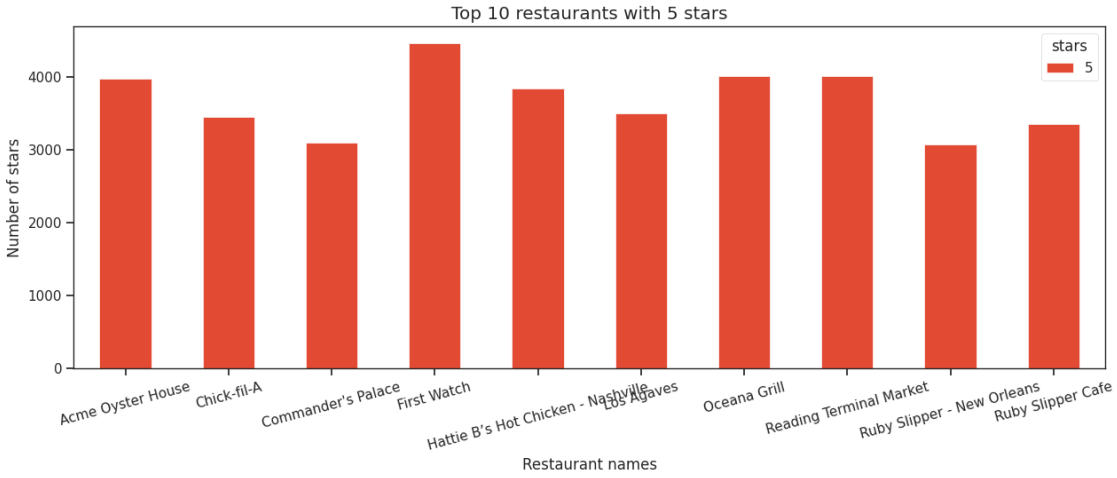
3.8 Top 10 Resteraunts with 5 Stars

[63]:
```
stars_df = df.groupBy('stars', 'Restaurant_name').count().orderBy('count').
↪toPandas()
stars_df['stars'] = stars_df.stars.astype(int)
```

[64]:
```
top_restaurant = stars_df[stars_df['stars']==5].sort_values(by=['count'],␣
↪ascending = False).head(10)
top_restaurant = top_restaurant.pivot(index = 'Restaurant_name', columns =␣
↪'stars', values = 'count')
```

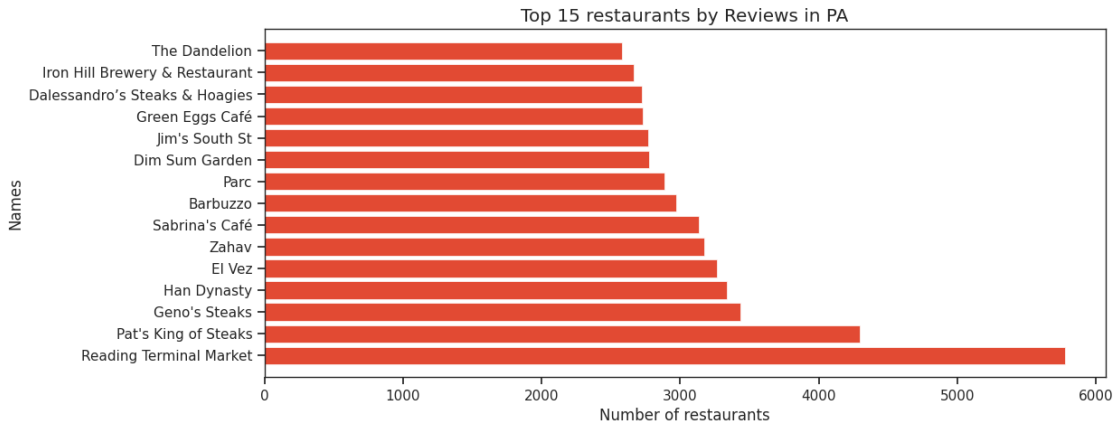
[65]:
```
top_restaurant.plot.bar(figsize=(15,5))
plt.ylabel("Number of stars")
plt.xlabel("Restaurant names")
plt.title("Top 10 restaurants with 5 stars")
```

```
plt.xticks(rotation=15)
plt.show()
```

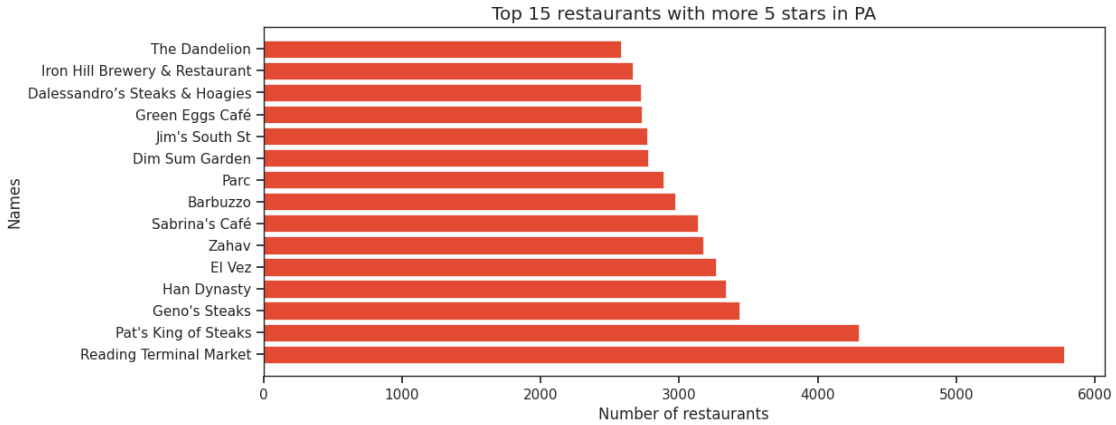


[66]:
```
# Top 15 in PA
data = df.filter(F.col('state')=='PA')
stars_df = data.groupBy('Restaurant_name').count().orderBy('count').toPandas()
# stars_df['review_count'] = stars_df.review_count.astype(int)
top_restaurant = stars_df.sort_values(by=['count'], ascending = False).head(15)
```

[67]:
```
plt.figure(figsize=(12,5))
plt.barh(top_restaurant.Restaurant_name, top_restaurant['count'])
plt.title("Top 15 restaurants by Reviews in PA")
plt.xlabel("Number of restaurants")
plt.ylabel("Names")
plt.ylim((-1,15))
plt.show()
```



[68]:
```
plt.figure(figsize=(12,5))
plt.barh(top_restaurant.Restaurant_name, top_restaurant['count'])
plt.title("Top 15 restaurants with more 5 stars in PA")
plt.xlabel("Number of restaurants")
plt.ylabel("Names")
plt.ylim((-1,15))
plt.show()
```



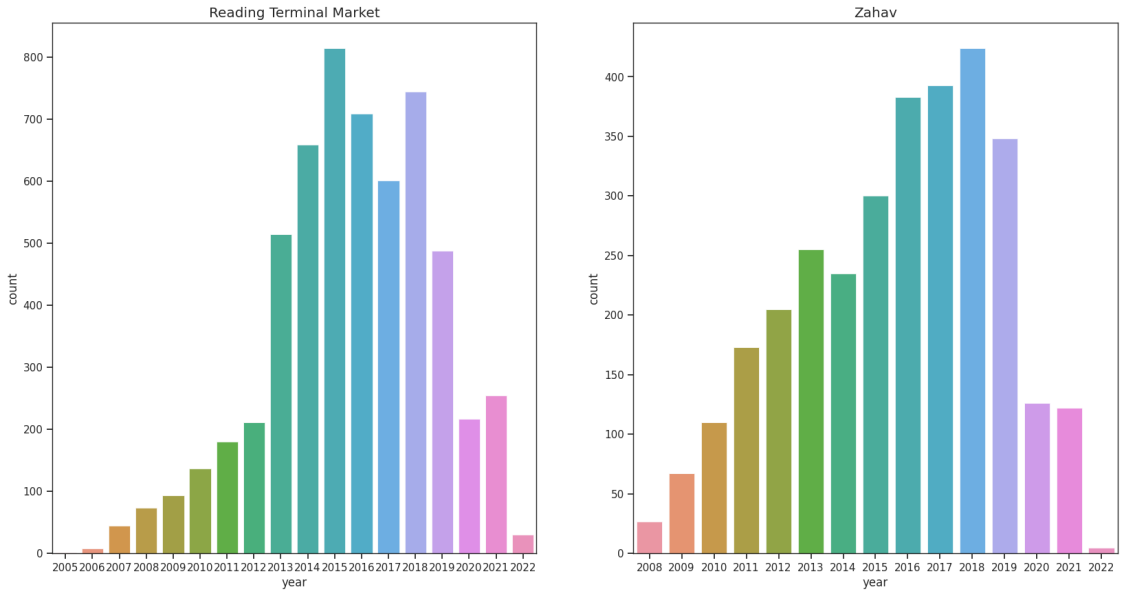
3.9 Top 2 Resterauntes in PA Trend over time

[69]:
```
# Top 4: Trend over time (this cell takes 15 mins to execute)
top_1 = df.filter(df.Restaurant_name == 'Reading Terminal Market').
 ↪groupBy('year').count().orderBy('count').toPandas()
top_2 = df.filter(df.Restaurant_name == 'Zahav').groupBy('year').count().
 ↪orderBy('count').toPandas()
# top_3 = df.filter(df.Restaurant_name == "Dalessandro's Steak & Hoagies").
 ↪groupBy('year').count().orderBy('count').toPandas()
# top_4 = df.filter(df.Restaurant_name == 'Barbuzzo').groupBy('year').count().
 ↪orderBy('count').toPandas()
```

[76]:
```
plt.figure(figsize=(20,10))

plt.subplot(1,2,1)
sns.barplot(x="year", y="count", data=top_1).set_title('Reading Terminal␣
 ↪Market')

plt.subplot(1,2,2)
sns.barplot(x="year", y="count", data=top_2).set_title('Zahav')

# plt.subplot(2,2,3)
```

```
# sns.barplot(x="year", y="count", data=top_3).set_title("Dalessandro's Steak &␣
 ↪Hoagies")

# plt.subplot(2,2,4)
# sns.barplot(x="year", y="count", data=top_4).set_title('Barbuzzo')
```

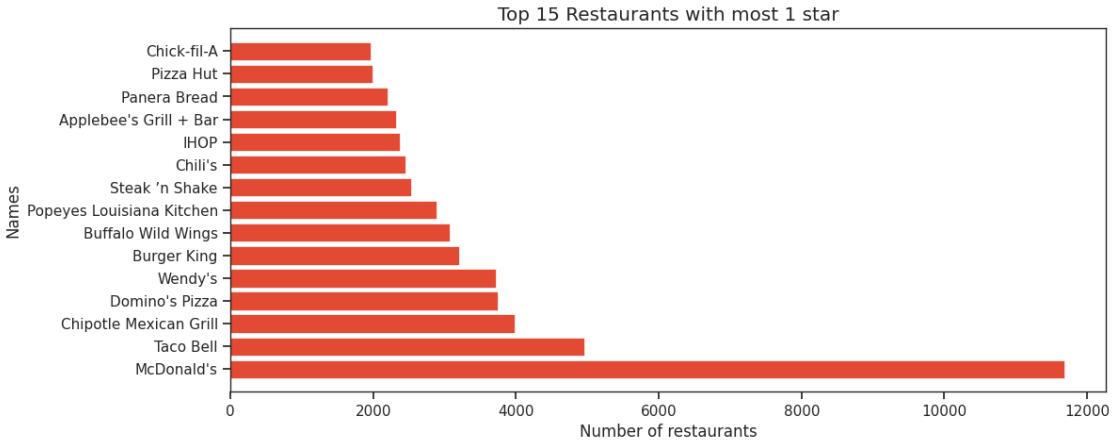
[76]: Text(0.5, 1.0, 'Zahav')



3.10 Worst Restaurants

[79]:
```
stars_df = df.groupBy('stars', 'Restaurant_name').count().orderBy('count').
 ↪toPandas()
stars_df['stars'] = stars_df.stars.astype(int)
top_restaurant = stars_df[stars_df['stars']==1].sort_values(by=['count'],␣
 ↪ascending = False).head(15)
```

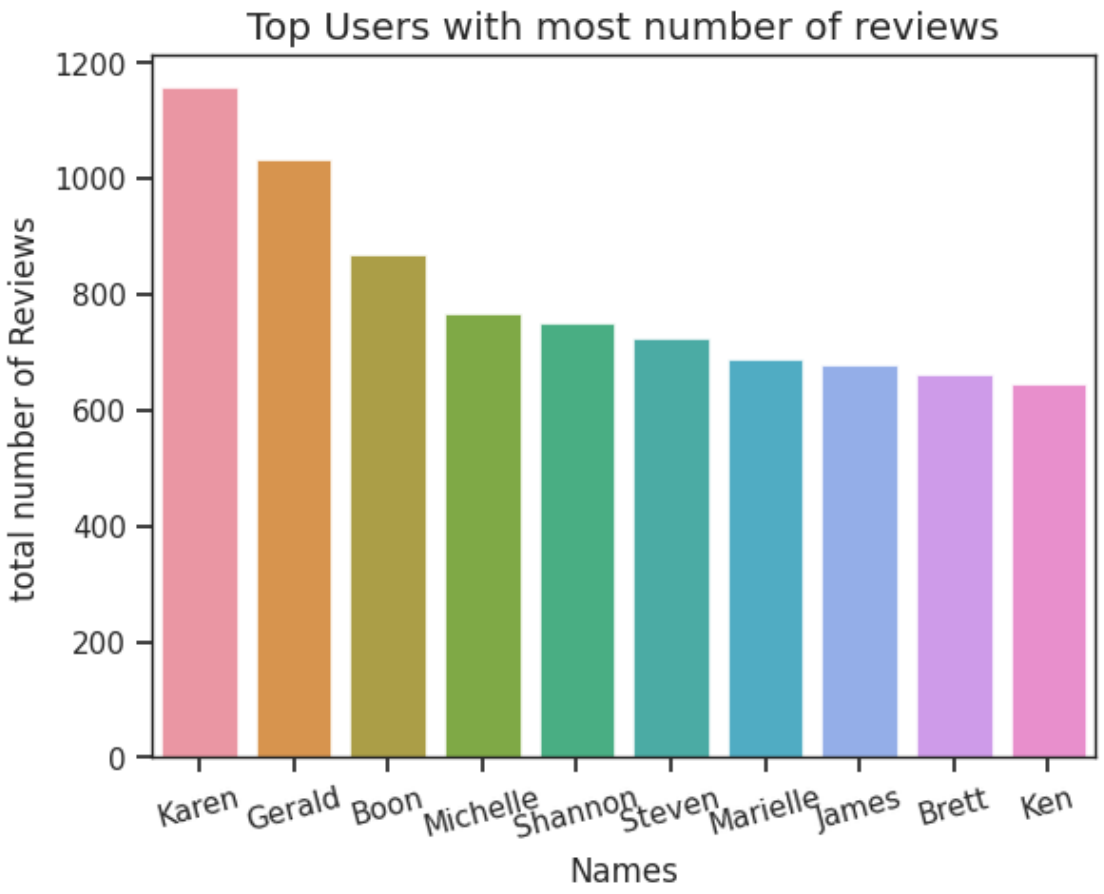
[80]:
```
plt.figure(figsize=(12,5))
plt.barh(top_restaurant.Restaurant_name, top_restaurant['count'])
plt.title("Top 15 Restaurants with most 1 star")
plt.xlabel("Number of restaurants")
plt.ylabel("Names")
plt.ylim((-1,15))
plt.show()
```



3.11 Top 10 users with most reviews

[81]:
```
df_user = df.groupBy('user_name', 'user_id').count().sort(F.desc('count')).
↪toPandas()
df_user = df_user.head(10)
```

[82]:
```
plt.figure()
sns.barplot(x='user_name', y='count', data=df_user)
plt.title("Top Users with most number of reviews")
plt.xlabel("Names")
plt.ylabel("total number of Reviews")
plt.xticks(rotation=15)
plt.show()
```



[82]:

[82]:

4 Word Cloud Based on Reviews

[83]:
```
punct_remover = F.udf(lambda x: remove_punctuations(x))
```

[84]:
```
import string
def remove_punctuations(text):
    punctuations = set(string.punctuation)
    no_punct_text = ''.join(char.lower() for char in text if char not in␣
 ↪punctuations)
    return no_punct_text
```

[85]:
```
df_word = df.select('*', punct_remover('text')).drop('text')
df_word = df_word.withColumnRenamed('<lambda>(text)', 'text')
```

[86]:
```
# !pip install wordcloud
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
```

[87]:
```
word_visual =df_word.select('text').limit(100000).toPandas()
```

[88]:
```
text = " ".join(review for review in word_visual.text)
```

[89]:
```
plt.figure(figsize = (20,10))
stopwords = set(STOPWORDS)
stopwords.update(["place", "food", "ordered","restaurant"])
wordcloud = WordCloud(stopwords=stopwords,␣
 ↪background_color="white",max_words=1000,width=800, height=400).
 ↪generate(text[:10000000])

plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

[89]: