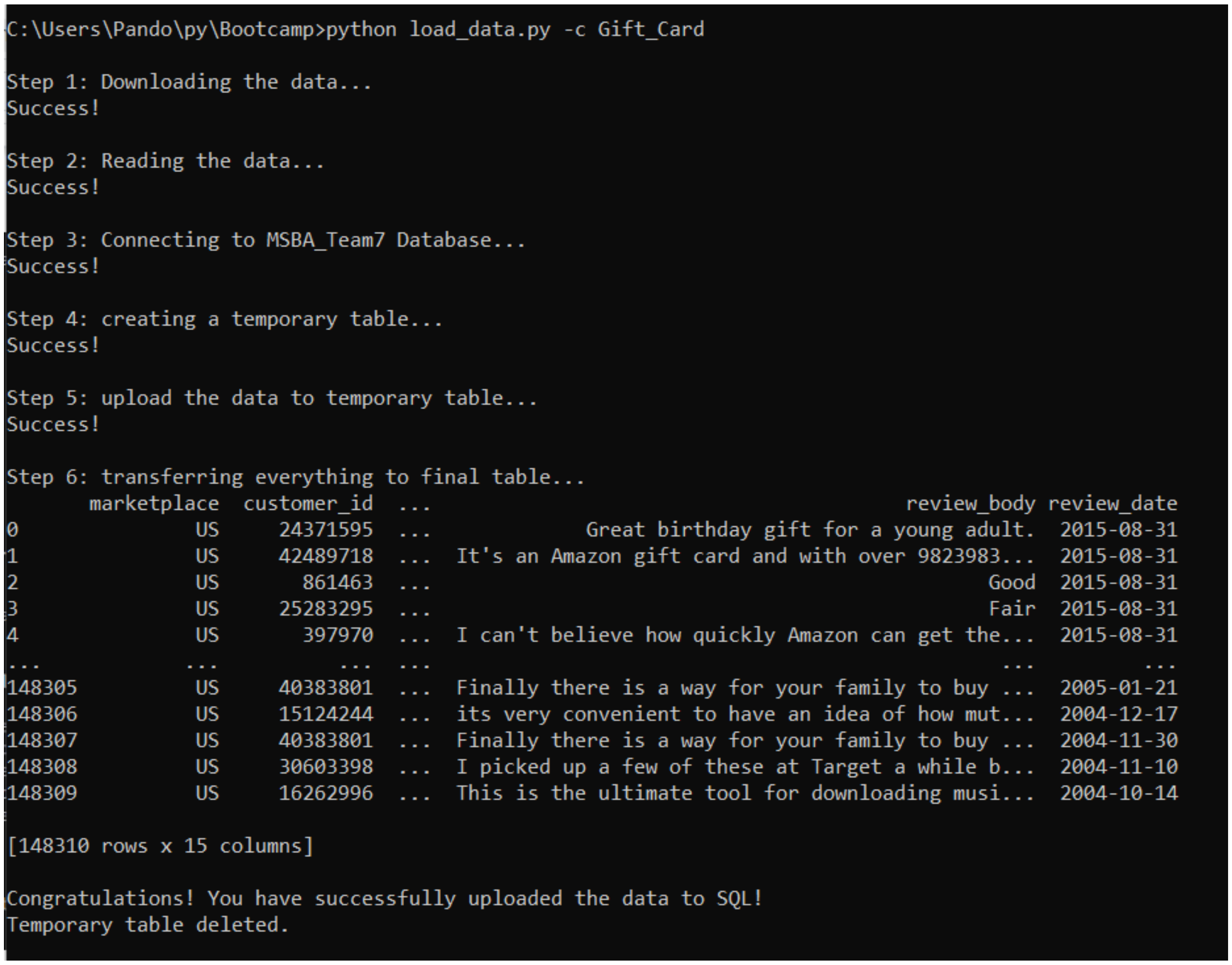
MSBA Team 7 Tech Bootcamp Project
Project Instructions:
https://canvas.emory.edu/courses/103848/files/8094720?module_item_id=1869139

<u>Part 1: Upload Data to the Database</u> (Python: load_data.py)

```
import requests
import pandas as pd
from mysql.connector import Error
import sqlalchemy
import pyodbc
import csv
import argparse
import sys

def main():
    # PARAMETERS
    # MY SQL Server Confidentials
    host='msba-bootcamp-prod.cneftpdd0l3q.us-east-1.rds.amazonaws.com'
    user='HSHOU3'
    password = open("mysql_pass.txt","r").readline().strip()

    # databases
    Database = "MSBA_Team7"
    table = "try_and_error"
    final_table = "reviews_raw"
    create_table = f'''
                CREATE TABLE IF NOT EXISTS {table} (
                    marketplace varchar(256),
                    customer_id INT,
                    review_id varchar(256),
                    product_id varchar(256),
                    product_parent INT,
                    product_title varchar(256),
                    product_category varchar(256),
                    star_rating INT,
                    helpful_votes INT,
                    total_votes INT,
                    vine varchar(256),
                    verified_purchase varchar(256),
                    review_headline varchar(1000),
                    review_body varchar(1000),
                    review_date DATE)
    '''
    insert_function = f'''
                INSERT INTO {table}
                (marketplace, customer_id, review_id,
                product_id,product_parent,product_title,product_category,
                star_rating,helpful_votes,total_votes,vine,verified_purchase,
                review_headline,review_body,review_date)
                VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)
                '''

    possible_categories = {'Wireless','Watches','Video_Games','Video_DVD',
                'Video','Toys','Tools','Sports','Software','Shoes','Pet_Products',
                'Personal_Care_Appliances','PC','Outdoors','Office_Products',
                'Musical_Instruments','Music','Mobile_Electronics','Mobile_Apps',
                'Major_Appliances','Luggage','Lawn_and_Garden','Kitchen',
                'Jewelry','Home_Improvement','Home_Entertainment','Home',

'Health_Personal_Care','Grocery','Gift_Card','Furniture','Electronics',
                'Digital_Video_Games','Digital_Video_Download','Digital_Software',

'Digital_Music_Purchase','Digital_Ebook_Purchase','Digital_Ebook_Purchase',
                'Camera','Books','Beauty','Baby','Automotive','Apparel'}

    #Arguments
    parser = argparse.ArgumentParser(description='Return data for the given product
category.')
    parser.add_argument('-c', '--category',
                    help='Input category name like "Firstword_Secondword_Thirdword"
to get data')
    args = parser.parse_args()

    #filter out the non-existing product categories
    if args.category:
        if args.category not in possible_categories:
            sys.exit("Please input a valid product category or check your input
format.")
        else:
            category = args.category

    #url and file names
    url =
f"https://s3.amazonaws.com/amazon-reviews-pds/tsv/amazon_reviews_us_{category}_v1_0
0.tsv.gz"
    filename = f"{category}_reviews.tsv.gz"
    # filename = "gift_reviews_copy.csv"  #TESTING

    #Starting the Process
    print("\nStep 1: Downloading the data...")
    download(filename,url)

    #Read the data & Preview
    print("\nStep 2: Reading the data...")
    data = read_data(filename)

    #Use pyodbc to connect to SQL Server
    print(f"\nStep 3: Connecting to {Database} Database...")
    try:
        params = 'DRIVER={MySQL ODBC 8.0 Unicode Driver};SERVER='+host
+';PORT=3306;DATABASE=' + Database + ';UID=' + user + ';PWD=' + password
        conn = pyodbc.connect(params)
        cursor = conn.cursor()
        cursor.execute(f"USE {Database}")
        print("Success!")
    except Error as e:
        if e.__class__ == pyodbc.ProgrammingError:
            conn == reinit()
            cursor = conn.cursor()

    #Delete and recreate the table if it exists
    print("\nStep 4: creating a temporary table...")
    delete(cursor,table)
    create(cursor,create_table)
    print("Success!")

    #Upload data to MySQL Server
    print("\nStep 5: upload the data to temporary table...")
    try:
        engine =
sqlalchemy.create_engine(f"mysql://{user}:{password}@{host}:3306/{Database}")
        data.to_sql(table,engine,index=False,if_exists="append")
        print("Success!")

        #Transfer everything from temporary table to final table
        print("\nStep 6: transferring everything to final table...")
        delete(engine,final_table)
        engine.execute(f"CREATE TABLE {final_table} LIKE {table}")
        engine.execute(f"INSERT INTO {final_table} SELECT * FROM {table}")
        print(pd.DataFrame(engine.execute(f"SELECT * FROM
{final_table}").fetchall()))
        print("\nCongratulations! You have successfully uploaded the data to SQL!")
    except Error as e:
        print("Error occurred during data uploading:",e)

    #delete temporary table
    delete(cursor,table)
    print("Temporary table deleted.")


#download the tsv.gz file
def download(filename,url):
    try:
        with open(filename, "wb") as f:
            r = requests.get(url)
            f.write(r.content)
        print("Success!")
    except error as e:
        print("Problem occurred during downloading data:",e)


#reading the data
def read_data(filename):
    try:
        data = pd.read_csv(filename,sep='\t', compression='gzip')
        #data = pd.read_csv(filename)   # TESTING
        #convert NaN data to None values
        data = data.where((pd.notnull(data)), None)
        #data = data.iloc[: , 1:]  #TESING: use if csv have 1 column of index
        print("Success!")
        return data
    except error as e:
        print("\nProblem occurred during downloading data:",e)

#delete and recreate a new table for testing the code
def delete(cursor,table):
    try:
        cursor.execute(f"DROP TABLE IF EXISTS {table}")
    except Error as e:
        print("error in deleting the table.")

#delete and recreate a new table for testing the code
```

```python
def create(cursor,create_table):
    try:
        cursor.execute(create_table)
    except Error as e:
        print("error in creating new table.")


if __name__ == "__main__":main()
```

Command Prompt:



```
C:\Users\Pando\py\Bootcamp>python load_data.py -c Gift_Card

Step 1: Downloading the data...
Success!

Step 2: Reading the data...
Success!

Step 3: Connecting to MSBA_Team7 Database...
Success!

Step 4: creating a temporary table...
Success!

Step 5: upload the data to temporary table...
Success!

Step 6: transferring everything to final table...
       marketplace  customer_id  ...                                       review_body review_date
0               US     24371595  ...                 Great birthday gift for a young adult.  2015-08-31
1               US     42489718  ...  It's an Amazon gift card and with over 9823983...  2015-08-31
2               US       861463  ...                                               Good  2015-08-31
3               US     25283295  ...                                               Fair  2015-08-31
4               US       397970  ...  I can't believe how quickly Amazon can get the...  2015-08-31
...            ...          ...  ...                                                ...         ...
148305          US     40383801  ...  Finally there is a way for your family to buy ...  2005-01-21
148306          US     15124244  ...  its very convenient to have an idea of how mut...  2004-12-17
148307          US     40383801  ...  Finally there is a way for your family to buy ...  2004-11-30
148308          US     30603398  ...  I picked up a few of these at Target a while b...  2004-11-10
148309          US     16262996  ...  This is the ultimate tool for downloading musi...  2004-10-14

[148310 rows x 15 columns]

Congratulations! You have successfully uploaded the data to SQL!
Temporary table deleted.
```

Part 2: Data Cleaning & Exploratory Analysis (SQL)

Q1: Append the data from the msba_db1.reviews_pet_products table to your reviews_raw table.

```
INSERT INTO reviews_raw
SELECT * FROM MSBA_DB1.reviews_pet_products;
```

Q2: Write queries to address the issues in your reviews_raw

```
-- Missing values
-- Check which column has missing value
select * from MSBA_Team7.reviews_raw where product_title is NULL;
-- missing value, suggest using product_id to index the correct
product_title
select * from MSBA_Team7.reviews_raw where review_headline is NULL;
-- missing value, suggest to keep it blank
select * from MSBA_Team7.reviews_raw where review_body is NULL;
-- 650 missing value, this column does not affect the analysis, suggest to
keep it blank
select * from MSBA_Team7.reviews_raw where review_date is NULL;
-- missing value, review_date data incorrectly been put on review_body,
suggest to alter the column
```

Data Types

```
-- #Check data type of each column
describe MSBA_Team7.reviews_raw
```

Field	*Data type*	*Reason*
marketplace	*varchar(256)*	*Nations' name only contains string, using varchar to represent*
customer_id	*int*	*Customer's id only contains integer, using int to represent*
review_id	*varchar(256)*	*Reviewers's id contains integer and string, using varchar to represent*
product_id	*varchar(256)*	*Product's id contains integer and string, using varchar to represent*
product_parent	*int*	*Product's parent only contains integer, using int to represent*
product_title	*varchar(256)*	*Product's title only contains string, using varchar to represent*

product_category	*varchar(256)*	*Product's category only contains string, using varchar to represent*
star_rating	*int*	*Star rating only contains integer, using int to represent*
helpful_votes	*int*	*Votes only contains integer, using int to represent*
total_votes	*int*	*Total votes only contains integer, using int to represent*
vine	*varchar(256)*	*Vine only contains string, using varchar to represent*
verified_purchase	*varchar(256)*	*Verified purchase only contains string, using varchar to represent*
review_headline	*longtext*	*Review headline contains lots of string, using longtext to represent*
review_body	*longtext*	*Review may contains lots of string, using longtext to represent*
review_date	*date*	*Using date to represent the date of review*

Text Processing

```
-- Avoiding error code #1175
SET SQL_SAFE_UPDATES = 0;
-- Replace punctuation
UPDATE MSBA_Team7.reviews_raw
SET review_body = REPLACE(review_body,':)','');
UPDATE MSBA_Team7.reviews_raw
SET review_body = REPLACE(review_body,',','');
UPDATE MSBA_Team7.reviews_raw
SET review_body = REPLACE(review_body,'.','');
UPDATE MSBA_Team7.reviews_raw
SET review_body = REPLACE(review_body,'and','');
UPDATE MSBA_Team7.reviews_raw
SET review_body = REPLACE(review_body,'or','');
UPDATE MSBA_Team7.reviews_raw
SET review_body = REPLACE(review_body,'or','')
```

Data domain validation and standardization.

```sql
-- Convert the review headline and comment into lowercase
SET SQL_SAFE_UPDATES = 0;
UPDATE reviews_raw SET review_headline = LOWER(review_headline);
SET SQL_SAFE_UPDATES = 0;
UPDATE reviews_raw SET review_body = LOWER(review_body);

CREATE TABLE reviews LIKE reviews_raw;
INSERT INTO reviews SELECT * FROM reviews_raw;
```

Redundant or irrelevant attributes.

```sql
-- Drop marketplace
ALTER TABLE MSBA_Team7.reviews_raw
DROP COLUMN marketplace;
```

Q3: Write queries to answer the data analysis questions

1. There are 2791929 rows * 15 columns in the dataset.

```
-- count the rows
SELECT COUNT(*) as No_of_Rows
FROM MSBA_Team7.reviews_raw;
-- count the columns
SELECT TABLE_NAME , count(COLUMN_NAME)
FROM information_schema.columns
GROUP BY TABLE_NAME;
```

2. There are 241117 different products in the dataset.

```
SELECT COUNT(DISTINCT product_id)
FROM MSBA_Team7.reviews_raw;
```

3. There are 207944 different customers in the dataset.

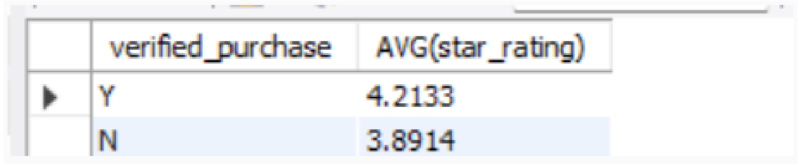
```
-- count the number of different customers
SELECT COUNT(DISTINCT customer_id)
FROM MSBA_Team7.reviews_raw;
```

4. There are 241255 combinations of product_id and product_parent.

```
-- count different combinations of product_id and product_parent
SELECT COUNT(DISTINCT product_id, product_parent)
FROM MSBA_Team7.reviews_raw;
```

5. The average product rating is 4.1749. The average rating is 4.2133 for verified purchase, and 3.8914 for unverified purchase. The average of two groups is different.

```
-- average of ratings
SELECT AVG(star_rating)
FROM MSBA_Team7.reviews_raw;
-- average of verified product ratings
SELECT verified_purchase, AVG(star_rating)
FROM MSBA_Team7.reviews_raw
GROUP BY verified_purchase
```



verified_purchase	AVG(star_rating)
Y	4.2133
N	3.8914

The remaining data analysis questions are answered using R in part 4

<u>Part 3: Download & Upload Data to R (Python)</u>

File: download_data.py

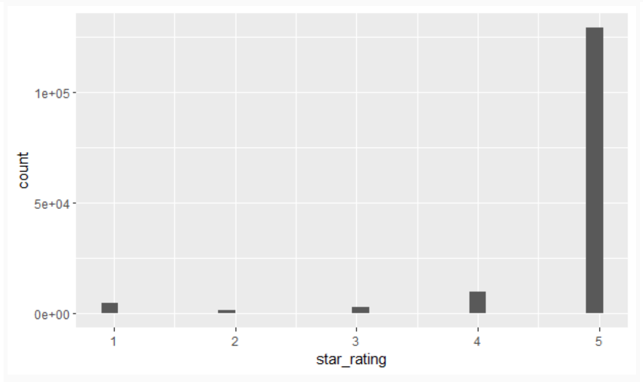
```
#Possible code for Data extraction from MySQL to Python...
from importlib import metadata
from multiprocessing.sharedctypes import Array
import sqlalchemy as db
import pandas as pd
from sqlalchemy import create_engine

engine =
create_engine("mysql+pymysql://insert_username:insert_password@msba-bootcam
p-prod.cneftpdd0l3q.us-east-1.rds.amazonaws.com/MSBA_Team7")
cnxn = engine.connect()
print(engine.execute("SELECT * FROM reviews_raw").fetchall())
metadata = db.MetaData()
reviews_raw = db.Table('reviews_raw', metadata, autoload=True,
autoload_with=engine)

df = pd.DataFrame(reviews_raw)
print('DataFrame:\n',df)

csv_data = df.to_csv()
print('n\CSV String:\n', csv_data)

#organizing into a dataframe using PANDAS
#columns = ['customer_id', 'review_id', 'product_id', 'product_parent',
'product_title', 'product_category', 'star_rating', 'helpful_votes',
'total_votes', 'vine', 'verified_purchase', 'review_headline',
'review_body', 'review_date', 'y_year']
```

Part 4: Further Analysis & Creating visualizations

Load results in R, do the remaining data analysis questions, and create any visualizations you need for the report.

6. The distribution of ratings looks like exponential distribution, visualization shown below. Most of the star ratings are five.

```
# Data visualization with ggplot2 library
library(ggplot2)
ggplot(df_reviews, aes(x = star_rating)) + geom_histogram()
```



7. a metric that captures the "fairness" or "bias" of a reviewer
   Consider vine = Y might be fraud reviews with marketing purpose, and helpful_votes = 0 means the star rating is not very useful. Then we look into this group especially with rating star = 5, these reviews are suspicious reviews. Also, those reviews with headlines but no review body are also suspicious. So the metric condition is: (vine = Y & helpful_votes = 0 & star_rating = 5 & review_body = NA)

```
unfair_checking <- df_reviews %>%
  filter(vine == 1 & helpful_votes < 2 & star_rating > 4)
mean(unfair_checking$star_rating)
```

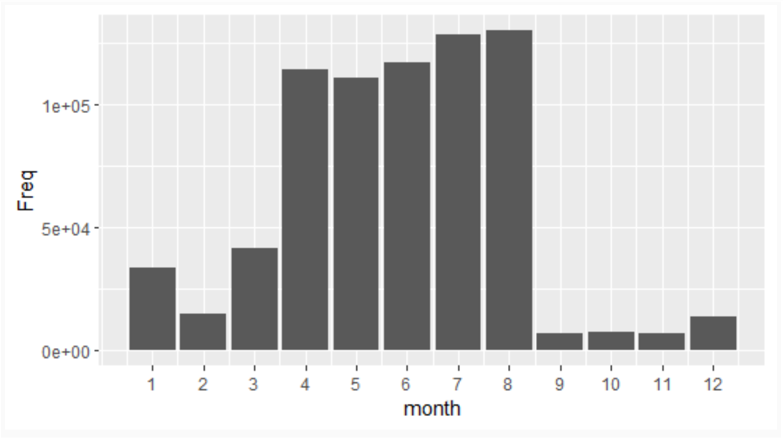
8. Aggregate number of reviews to the month level
   a. SQL part

```
-- convert data format to month
SELECT
EXTRACT(YEAR FROM review_date) AS year,
            EXTRACT(MONTH FROM review_date) AS month
FROM reviews_raw;
```

   b. R Part

```
library("ggplot2")
month_reviews <- 'Q8_Month_date.csv'
mon_reviews <- read_csv(month_reviews)
```

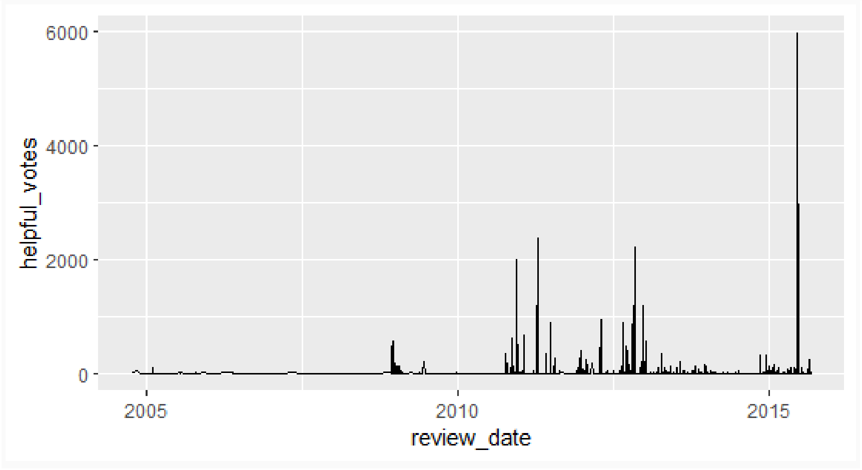
```
mon_total <- mon_reviews%>%
  group_by(month)%>%
  summarise(Freq=n())
ggplot(mon_total, aes(x = month, y=Freq)) +
  geom_bar(stat='identity') +
  scale_x_continuous(breaks = seq(1, 12, by = 1))
```



c. There exists a seasonality. In the second quarter, the frequency is extremely high. While in the first and the last quarter, the frequency is reduced.

9. Time series plot with helpful reviews

```
library(magrittr)
library(ggplot2)
# Filter for the total votes >= 1
df_filtered_reviews <- df_reviews %>%
 filter(helpful_votes >= 1)
ggplot(data.frame(df_filtered_reviews), mapping = aes(x = review_date, y =
helpful_votes)) +
  geom_line()
```



There are more helpful reviews during years of 2011 and 2016, and the number of helpful votes peaked in 2016, reaching about 6000 votes.

10. Time series plot on adjusted helpful reviews
    While grouping the yearly helpful reviews, we can find that it shows a rising trend through the years, which may indicate the improving quality of reviews.

```
df_reviews <- df_reviews %>%
  group_by(y_year)%>%
  summarize(total=n()) %>%
  mutate(helpful = as.integer(any(df_reviews$helpful_votes!=0))) %>%
  summarize(helpful=n())

ggplot(df_reviews, aes(x = y_year, y=helpful/total)) +
  geom_bar(stat='identity')
```

11. Decision on retaining vine program on Amazon
    Based on the outcome found in question 7, we think the vine program can be retained, but it is better to manage the use of vine programs since the unfair star rating is very influential for market purchase.

Part 5: Write the report (report.doc)

- Explains all the data cleaning that you did and why you did it
- Describes any insights that you found as you answered the data analysis questions
- Answers all the questions from Data analysis questions section and includes all the visualizations that you created during Part 4
- Lists any difficulties that you encountered throughout the project or any bugs that were hard to fix and explain how you solved them
- Specifies the most useful or interesting thing(s) that you learned as part of this project or the Technology Bootcamp in general,

Tech Bootcamp Project Report
MSBA Team 7
Chris Chou, Zackery Goldberg, Pandora Shou, Tiffany Zhao

Amazon Web Services offer access to datasets that contain review information about different types of products. For this project, we use Python to transfer data between a web server, SQL, and R; then, we use SQL to clean and analyze the data; finally, we use R to conduct exploratory data analysis.

Cleaning the Data
After uploading the data to the SQL server, we cleaned the data step by step before further analysis.
The first step is text processing. When we looked at the review_body column, there were a lot of unnecessary symbols, such as ':)', ',', 'and', 'or', etc. We removed these characters to reduce the data size so that text mining will be easier in the future.
Next, we standardized the data. We noticed that the headline column and review_body column are in mixed uppercase and lowercase, which is a problem if we want to filter distinct values. Therefore, we changed the data in these two columns into lowercase.
Lastly, we removed redundant attributes. After exploratory data analysis, we found only one value in the marketplace column -- "US". A column that contains one class doesn't add any value to our prediction. Therefore, we dropped the marketplace column.

Analyzing the Data
Next, we analyzed the cleaned data to dig for more insights. For summary, we have 2791929 rows and 15 columns in total after appending the pet. Then we used Distinct Count to count the unique values of products and customers, and found there exist 241117 unique products and 207944 customers. By calculating the unique combinations of products and customers, we found there are 241255 types of combinations, which is higher than both the product type and the number of customers, which indicates that there may be customers purchasing different types of products, which coordinates with real-life situations.
After analyzing the basic attributes of the data frame, we look into the statistics side.
By comparing and visualizing the star rating for all purchases and verified purchases, we concluded that the verified purchases have lower star rating compared to average. This is actually a warning for the customers, as it indicates potential marketing fraud which involves some unfair star rating for unverified purchases. These unverified purchases might have been returned as soon as delivered or just from the partner's help to make the rating look better, which is suspicious and unreliable in either situation.
For better visualization, we connect SQL databases to R through the "RMySQL" package, and we fetch the data into a new data frame.
Then we used the ggplot package to plot the distribution of star rating, the plot shows star rating aggregates at 5 points, which seems like an overall good comment.
However in question 7 when we created a metric and excluded those unfair data, the average decreased a lot. This proves the validity of out metric and also indicates the potential unfair and bias in star rating. We also aggregate time series data into monthly and yearly data in question

8 and question 10 relatively. The trend shows that there exists a seasonality. In the second quarter, the frequency is extremely high. While in the first and the last quarter, the frequency is reduced.
Based on unfair validation and data analysis, we concluded that the vine program can be retained, but it should be managed on the star rating and reviews function to avoid potential unfair or biased comments.

Overcoming the Challenges
Throughout the project, we encountered several technical difficulties. When we attempted to upload data to the SQL server using Python, our biggest challenge is cutting down the program running time. Initially, we picked the sql-connector package to connect to the SQL server, which allowed us to upload data one row at a time. This way, we wrote simpler code, but uploading the entire gift card dataset (148,310 rows) took 2 hours, not to mention that the gift card dataset is the smallest of all datasets! To optimize the algorithm, we found the to_sql function from the sqlalchemy package, which inserts data by bulks. Since sqlalchemy requires that we use pyodbc as the connection driver, we had to re-do all the code about connecting to the SQL server. However, our efforts paid off. We eventually cut down the python program running time to under 3 minutes. As the result, we are more flexible in test-running the python script, and we can manipulate the data using MySQL without worrying about accidentally making mistakes.

Learning from the Project and the Bootcamp
Looking back at the 8 days of the intense boot camp, we have not only improved our coding skills but also learned a lot about rules in the programming world.
First, always keep in mind what we want to accomplish. If we are trying to query and analyze the data, we can use SQL. If we are trying to conduct hypothesis tests and create visualizations, we can use R. If we are trying to build machine learning models or even use one language to do everything, python will be our best choice. By using the right language for the right task, we can maximize each language's utility. What we aim for also influences our choice of the package/driver. Therefore, we should be clear about our goal before jumping into coding.

Also, coding is not done after a script can function properly. The understandability and speed of the script also impact the performance a lot. If we just pile everything together without formatting and commenting, when we go back to one particular step for debugging, even the coder cannot trace the code. The speed of running the script also matters. From the project perspective, cutting down running time allows more opportunities to make mistakes. The other team members have more time to finish their parts of the project. There is also less chance of program interruptions, which leads to fewer potential data corruptions. Therefore, even when the code can achieve the desired function, we should always aim to optimize the algorithm.