Ronnald Le
CS 512
Data Science Tools & Programs
03/14/2023

Analyzing the Impact of Compliments on Yelp User Engagement

Problem Statement:

The problem is that we need to understand user behavior on Yelp and the impact of various compliments given to them by other users. Since we are not able to understand user behavior and preferences, it can be challenging to improve the platform experience for both businesses and customers. Not knowing the impact of compliments on users can lead to ineffective marketing strategies and miss opportunities that may come along that way when engaging with customers. The affected parties will include Yelp, business partners, and users. A good starting point would be to analyze user data to answer the questions being asked.

OSEMN Process:

Obtain
The Yelp dataset was obtained from a .tar file, and a python script called "pythonscriptjsoncsv.py" was used to extract it into .json files and a .csv file. The data was then uploaded to a Google Cloud Storage (GCS) bucket named "cs512_group2".

Scrub
Remove NA values from the dataset and other values that may affect the data. Partition the datatset into 160MB size each for dataprep to process successfully.

Explore
Use Dataprep to create a recipe that answers a specific question of interest. In this case, the question is about finding the ratio of "compliment_hot" and "review_count" for Yelp users and sorting them in descending order. I also use BigQuery to partition the dataset to answer the second question. The third question is done in R Studio to create a visualization.

Model
Create a Big Query dataset and a table to calculate the correlation between "compliment_funny" and "compliment_plain". Use a query to count the number of rows and find the correlation coefficient.

Interpret
Use R Studio to visualize the average number of "compliment_cute" being received by users who have been on Yelp before 2010 vs. after 2010. Filter out the dates for those who have been with Yelp before 2010 and after 2010, use the mean function, and remove any NA values. The resulting bar chart shows that those who join Yelp before 2010 have much more compliment_cute than those who joined after.

Description of Process:

Question 1: Which users have the highest ratio of "compliment_hot" to "review_count"?

To start working with the Yelp dataset, I used a Python script called "pythonscriptjsoncsv.py" to extract it from a .tar file into JSON and CSV files. Next, I uploaded the JSON files to my Google Cloud Storage (GCS) bucket, which I named "cs512_group2".

To work with the data in Dataprep, I created a new flow and selected "cloud storage" as the dataset. From there, I chose the "yelp_academic_data_set_user.json" file from my "cs512_group2" bucket. After processing the dataset, I split it into four 1GB chunks for easier handling.

Unfortunately, when attempting to upload the chunks back to the "cs512_group2" bucket, I encountered an issue: the 1GB chunks were too large for Dataprep to handle. To address this, I modified the script to partition the data into smaller chunks (~500MB each) and tried running it through Dataprep again.

Despite these efforts, Dataprep still failed to process the data. As a result, I turned to using the Google Cloud SDK Shell, which prompted me about billing issues.

```
C:\Users\hiimr\Documents\data>gsutil cp yelp_academic_dataset_business.json gs://my-bucket/
Copying file://yelp_academic_dataset_business.json [Content-Type=application/json]...
ResumableUploadAbortException: 403 The billing account for the owning project is disabled in state closed
```

Finally, I chose to use PySpark to partition the dataset. I started by loading up PySpark and using the spark.read.json method to read in the "yelp_academic_dataset_user.json" file. Through experimentation, I discovered that partitioning the dataset into 160MB files was the largest size that could be processed by Dataprep without crashing. To achieve this, I used PySpark to repartition the "yelp_academic_dataset_user.json" file into 20 parts for better parallelism. Next, I used the "df.write.json" command to write the partitioned data into separate files, resulting in the creation of a new file called "yelp_academic_dataset_user" in the "cs512_group2" bucket. This file contains multiple chunks, each ranging around 160MB in size. The entire process involved multiple stages to complete.

```
>>> df = spark.read.json("gs://cs512_group2/yelp_
academic_dataset_user.json")
[Stage 1:>
```

```
>>> df = df.repartition(20)
>>> df.write.json("gs://cs512_group2/output/yelp_
academic_dataset_user_20_parts.json")
```

After successfully repartionining the dataset into 20 chunks, I decided to check if the data is appropriately parsed. I went with the “show()” method to display a sample of data. The data should theoretically be parsed and loaded into dictionaries with the schema showing each of the columns as a struct of multiple fields.

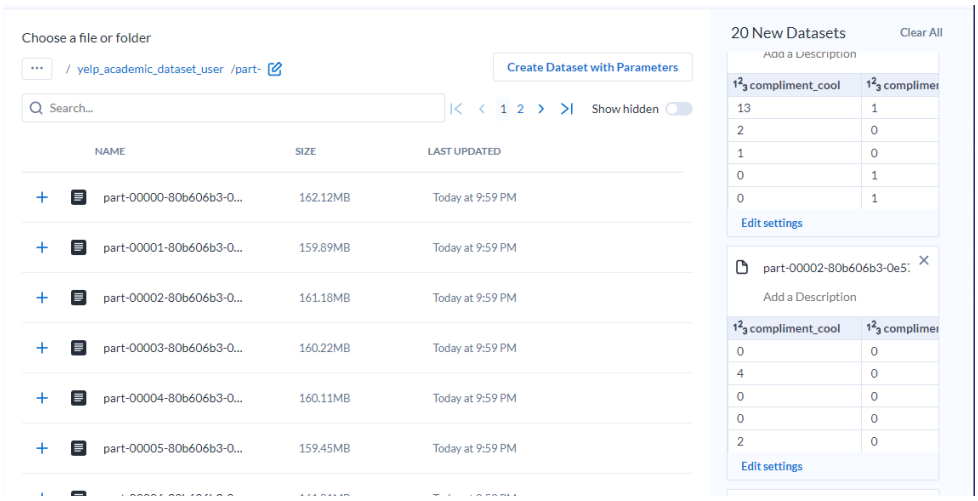
```
>>> spark = SparkSession.builder.appName("YelpDat
a").getOrCreate()
>>> yelp_data = spark.read.json("gs://cs512_group
2/yelp_academic_dataset_user.json")
[Stage 5:>
[Stage 5:==>
```

I decided to use PySpark’s dataframe API to check if the data is appropriately parsed. I load the JSON data from GCS bucket and have it display the first 5 rows of the DataFrame. Since the data is appropriately parsed, I am able to see a tabular display of the first 5 rows for the Yelp data. If it was not appropriately parsed, then I may see errors or weird outputs when displaying the data.

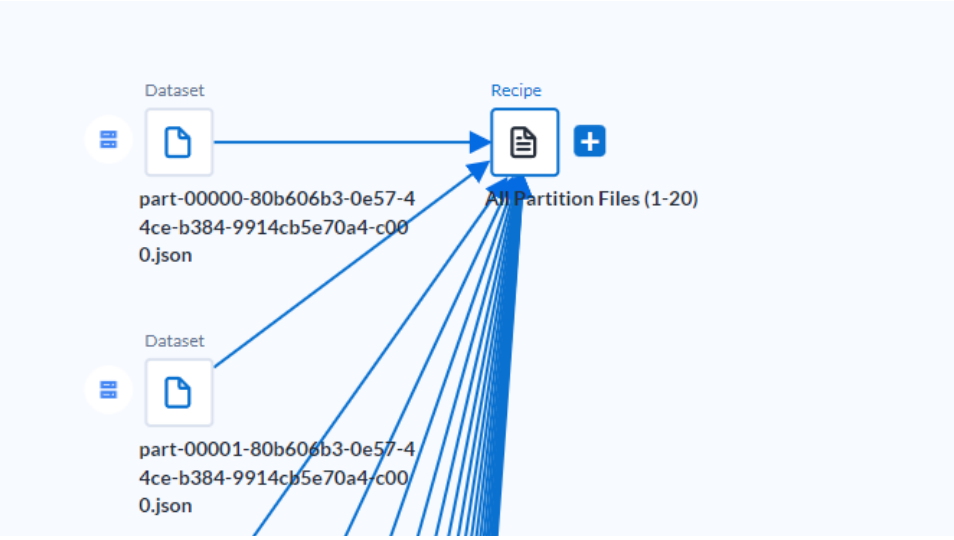
```
>>> yelp_data.show(5)
[Stage 6:>

+-------------+---------------+---------------+--
--------------+--------------+---------------+---
------------+---------------+-----------------+--
-------------+-----------------+--------------
--+-----+--------------------+----+-------------
------+-----+------+-----------+------+---------
----------+-------------------+
|average_stars|compliment_cool|compliment_cute|co
mpliment_funny|compliment_hot|compliment_list|com
pliment_more|compliment_note|compliment_photos|co
mpliment_plain|compliment_profile|compliment_writ
er| cool|               elite|fans|              f
riends|funny|  name|review_count|useful|
    user_id|      yelping_since|
```

The final step is to edit the recipe in dataprep after successfully uploading all of the partitions.

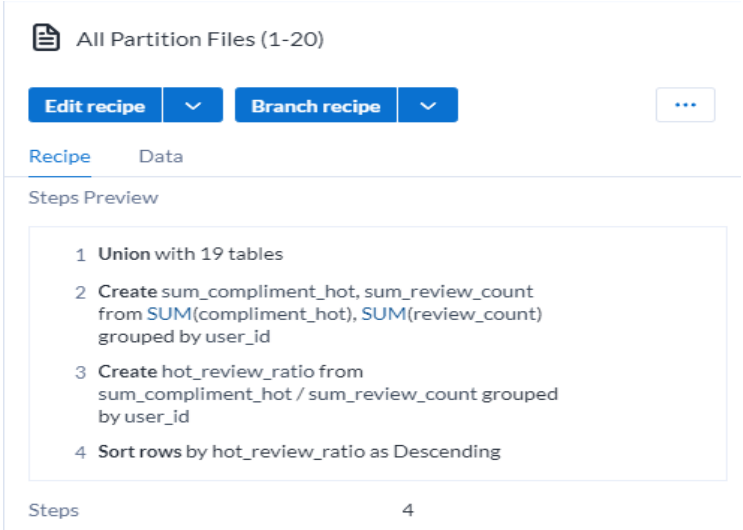


I accessed the "Final Project Flow" view and selected all of the partitions. I then used the "union" option to combine all of the .json files into a single recipe. Using this recipe, I was able to answer my question of interest. During the recipe editing process, I could preview my newly created column called "hot_review_ratio", which calculates the ratio of "compliment_hot" to "review_count".

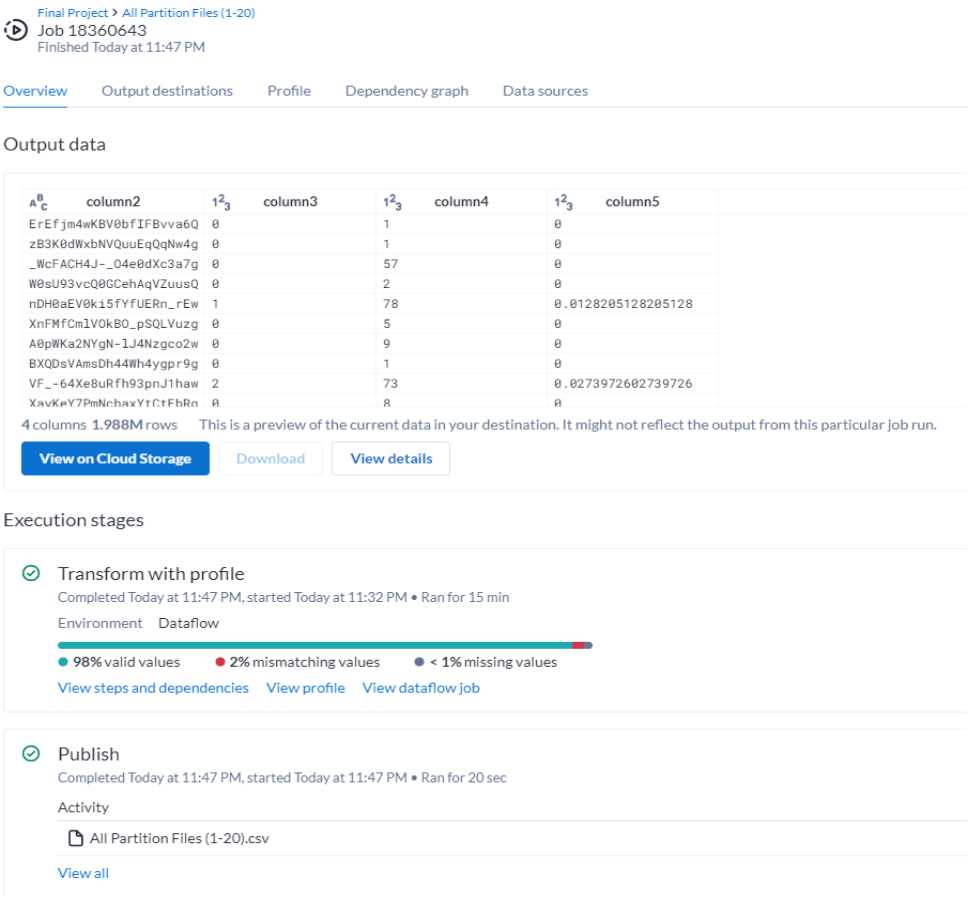


user_id	sum_compliment_hot	sum_review_count	hot_review_ratio
12,741 Categories	0.0 - 2,420.0	1.00 - 5,434.00	0 - 1
jsp8h4yF3VI7_sk3EyvEIQ	2420	586	4.129692832764505
wIs7BiEcGzb2Z7eXWvk1pA	619	247	2.506072874493927
F_5_UNX-wrAFCXuAkBZRDw	2103	1237	1.700080840743735
zZW_RGlpS4p2tu0KTz7hxQ	21	13	1.6153846153846154
7n_Z6Du3NrZB1GhxpQDW7Q	166	111	1.4954954954954955
MtDsCRx8tRZTN8V6YMtKLA	946	982	0.9633401221995926
wzUFZ3DaU7k9R2S54IMPfg	676	758	0.8918205804749341
24IYk1sP7mtJU_vq65JnJA	96	118	0.8135593220338984
y9ny5Al52eLS4xwNE1unqg	7	9	0.7777777777777778
bpqcoOFZArVLzCP9q2IaMQ	160	208	0.7692307692307693
Ad-v2Z4vvuzxZxbpywIm0A	330	449	0.734966592427617
r7XYI0fArgMPVqpv_3TMUw	51	70	0.7285714285714285
nLtUXHvuAD_pboY8hS_ZYg	8	11	0.7272727272727273
CqJVkZYzNiPgOVaOB1bMdw	119	168	0.7083333333333334
akVV8m8TdYgTrHw1PTsoaA	104	148	0.7027027027027027
Uskqv_NRH83oJmS-9yADvA	2	3	0.6666666666666666
JMtTO8a2ji_D4j3McSiJsA	36	54	0.6666666666666666
XUqfnt21HvIHUEJ0mPBR-Q	19	31	0.6129032258064516
zgV0ZroIF956gw4cul8MHA	412	693	0.5945165945165946
HEUnr3otTggyn3ejKoC3Bg	4	7	0.5714285714285714
jvjLESkwupvuCp-mLFky0w	6	11	0.5454545454545454

Based on the question of interest, I sorted the recipe to display the "hot_review_ratio" in descending order. The result showed that the top Yelp user has a ratio of 4.13, indicating that these Yelp users are highly active in this case.

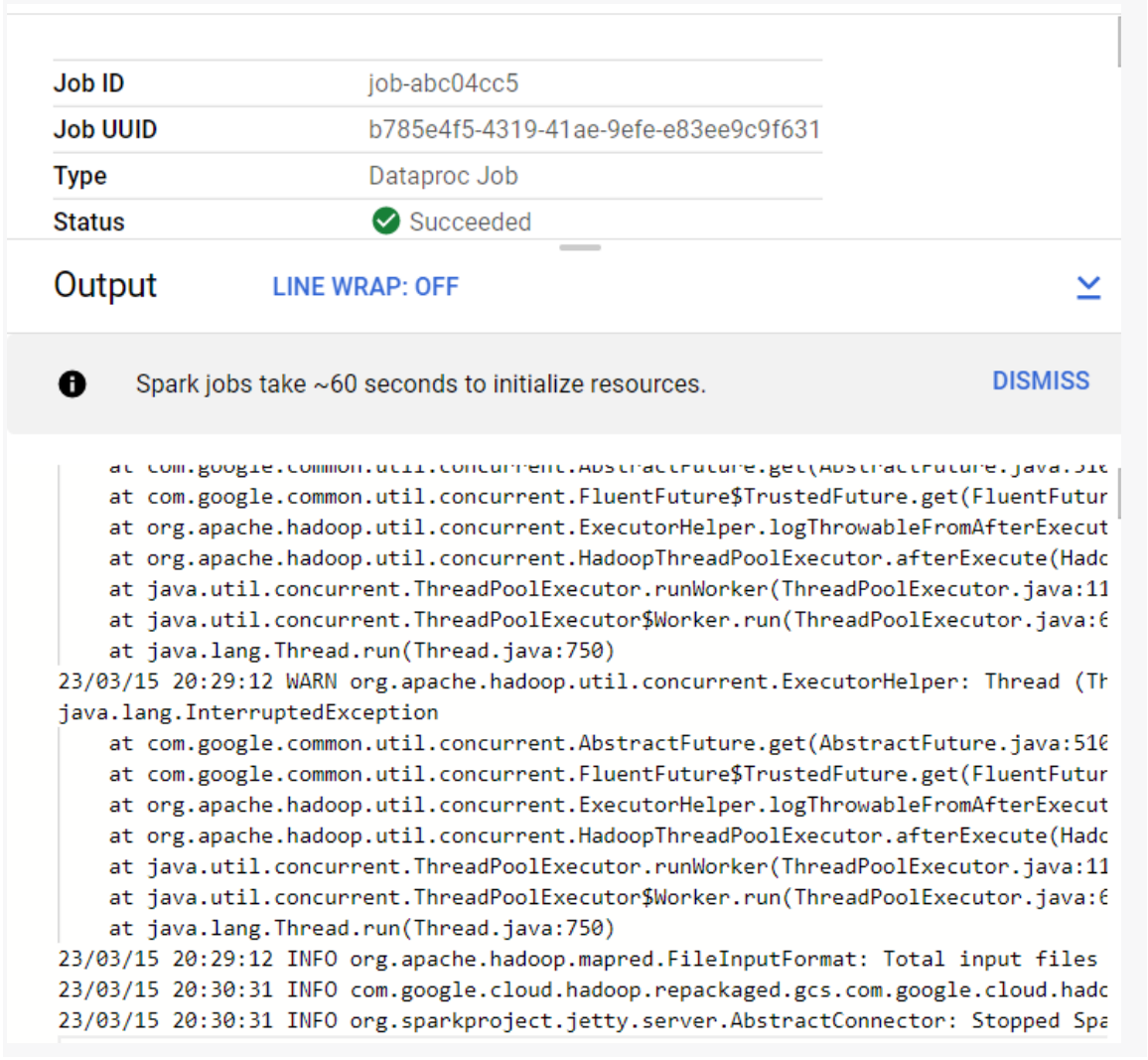


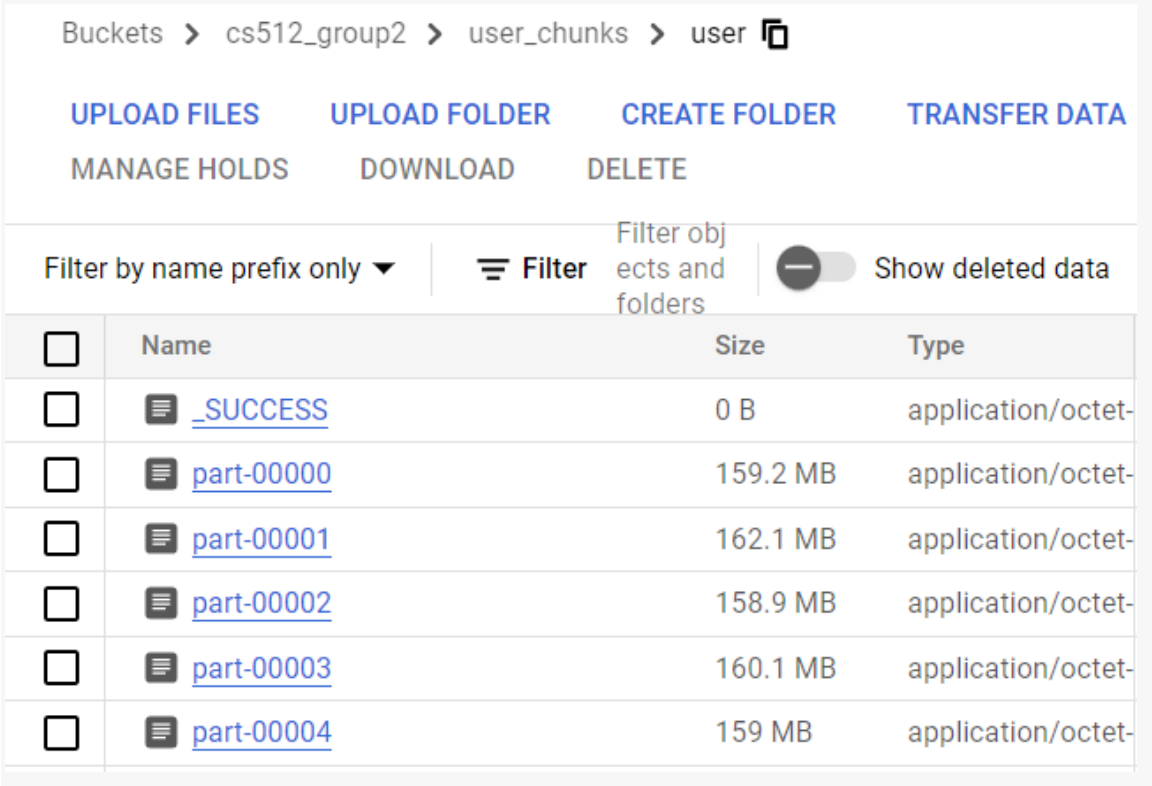
Once the recipe looks good, I ran the job that lasted for 15 minutes. The job run was a success and the .csv files were published into a folder called “All Partition Files (1-20)”.



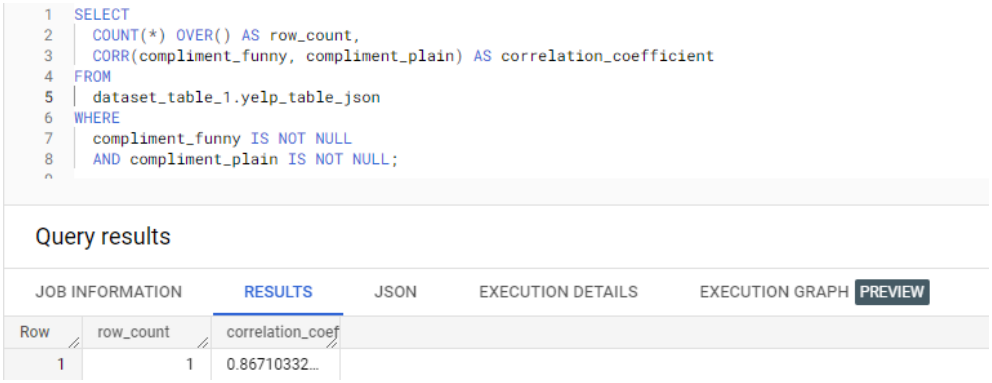
Question 2: Is there a correlation between the number of "compliment_funny" a user receives and the number of "compliment_plain" they receive? If so, what is the correlation coefficient?

I went into Dataproc to create a cluster, and after successfully creating it, I created a job with a Python script I had prepared. I uploaded the Python script to my bucket and used the Hadoop jar file to connect to BigQuery. Once the job was successfully executed, I proceeded to BigQuery to calculate the correlation between the “compliment_funny” and “compliment_plain” columns.


Job ID
job-abc04cc5
Job UUID
b785e4f5-4319-41ae-9efe-e83ee9c9f631
Type
Dataproc Job
Status
Succeeded
Output
LINE WRAP: OFF
Spark jobs take ~60 seconds to initialize resources.
DISMISS
at com.google.common.util.concurrent.FluentFuture$TrustedFuture.get(FluentFutur
at org.apache.hadoop.util.concurrent.ExecutorHelper.logThrowableFromAfterExecut
at org.apache.hadoop.util.concurrent.HadoopThreadPoolExecutor.afterExecute(Hadc
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:11
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:6
at java.lang.Thread.run(Thread.java:750)
23/03/15 20:29:12 WARN org.apache.hadoop.util.concurrent.ExecutorHelper: Thread (Th
java.lang.InterruptedException
at com.google.common.util.concurrent.AbstractFuture.get(AbstractFuture.java:510
at com.google.common.util.concurrent.FluentFuture$TrustedFuture.get(FluentFutur
at org.apache.hadoop.util.concurrent.ExecutorHelper.logThrowableFromAfterExecut
at org.apache.hadoop.util.concurrent.HadoopThreadPoolExecutor.afterExecute(Hadc
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:11
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:6
at java.lang.Thread.run(Thread.java:750)
23/03/15 20:29:12 INFO org.apache.hadoop.mapred.FileInputFormat: Total input files
23/03/15 20:30:31 INFO com.google.cloud.hadoop.repackaged.gcs.com.google.cloud.hadc
23/03/15 20:30:31 INFO org.sparkproject.jetty.server.AbstractConnector: Stopped Spa



To answer the second question, I chose to utilize Big Query. First, I created a new dataset and table, uploading the “user_chunks” data from my Google Cloud Storage bucket. I then constructed a query to count the number of rows and apply the correlate function to determine the correlation between the number of "compliment_funny" a user receives and the number of "compliment_plain". The resulting correlation coefficient was 0.867, indicating a strong positive correlation. This suggests that users who receive many "compliment_funny" also tend to receive many "compliment_plain", and vice versa.



```sql
SELECT
  COUNT(*) OVER() AS row_count,
  CORR(compliment_funny, compliment_plain) AS correlation_coefficient
FROM
  dataset_table_1.yelp_table_json
WHERE
  compliment_funny IS NOT NULL
  AND compliment_plain IS NOT NULL;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH PREVIEW

Row	row_count	correlation_coef
1	1	0.86710332...

Question 3: What is the average number of "compliment_cute" received by users who have been on Yelp since before 2010, and how does this compare to the average number of "compliment_cute" received by users who joined Yelp after 2010?

For the last question, I used R Studio to visualize the average number of “compliment_cute” received by Yelp users who joined before 2010 and those who joined after. I first filtered the data for users who joined before 2010 and those who joined after, then used the mean function to calculate the average number of “compliment_cute” received. I removed any NA values that could affect the data. I created a bar chart to display the results, and while I tried to improve its appearance, this is the final version. The chart shows that users who joined Yelp before 2010 received significantly more “compliment_cute” than those who joined after.

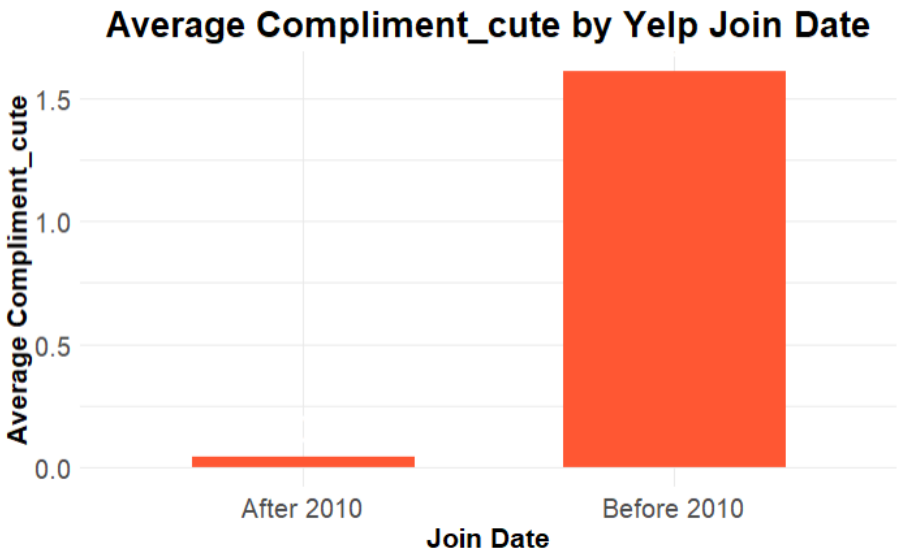
```
# Filter the data for users before 2010 and after 2010 timeframe
users_before_2010 <- yelp_data |>
  filter(as.Date(yelping_since) < as.Date("2010-01-01"))
users_after_2010 <- yelp_data |>
  filter(as.Date(yelping_since) >= as.Date("2010-01-01"))

# Find the average number of compliment_cute for each group
avg_cute_before_2010 <- mean(users_before_2010$compliment_cute, na.rm = TRUE)
avg_cute_after_2010 <- mean(users_after_2010$compliment_cute, na.rm = TRUE)

cat("Average compliment_cute before 2010:", avg_cute_before_2010, "\n")
cat("Average compliment_cute after 2010:", avg_cute_after_2010, "\n")

# Visualization to compare the averages
df <- data.frame(
  group = c("Before 2010", "After 2010"),
  avg_cute = c(avg_cute_before_2010, avg_cute_after_2010)
)

ggplot(df, aes(x = group, y = avg_cute)) +
  geom_bar(stat = "identity", fill = "#FF5733", width = 0.6) +
  geom_text(aes(label = round(avg_cute, 2)), vjust = -0.5, color = "white", size = 8) +
  theme_minimal() +
  theme(
    plot.title = element_text(face = "bold", size = 20, hjust = 0.5),
    axis.text = element_text(size = 15),
    axis.title = element_text(face = "bold", size = 16),
    legend.title = element_blank(),
    legend.position = "none"
  ) +
  labs(title = "Average Compliment_cute by Yelp Join Date",
       x = "Join Date", y = "Average Compliment_cute")
```


Average Compliment_cute by Yelp Join Date
Average Compliment_cute
1.5
1.0
0.5
0.0
After 2010
Before 2010
Join Date