Nate Boyle
5/24/2023

SQL for Data Science Capstone Project:
Milestone 4 Peer-graded Assignment:
“Your Findings (Storytelling)”

Please find my answers to the assignment questions in Verdana and blue.

Review criteria

Your presentation will be a culmination of the other milestones you completed in this project-based course. You will create your presentation using any media you choose and use the Rich Text Editor feature to submit your presentation.

For presentation ideas:

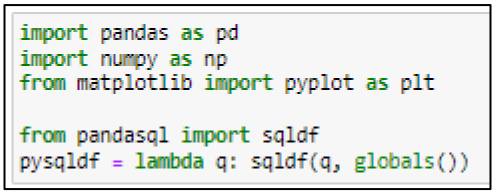
- Look at DataBricks and markdown (notebooks)
- Visualizations … raw data Infographics
- Presentation Styles / Audiences
- Reference SQL output vs. visualizations

Your presentation must include:

Build on Project Proposal

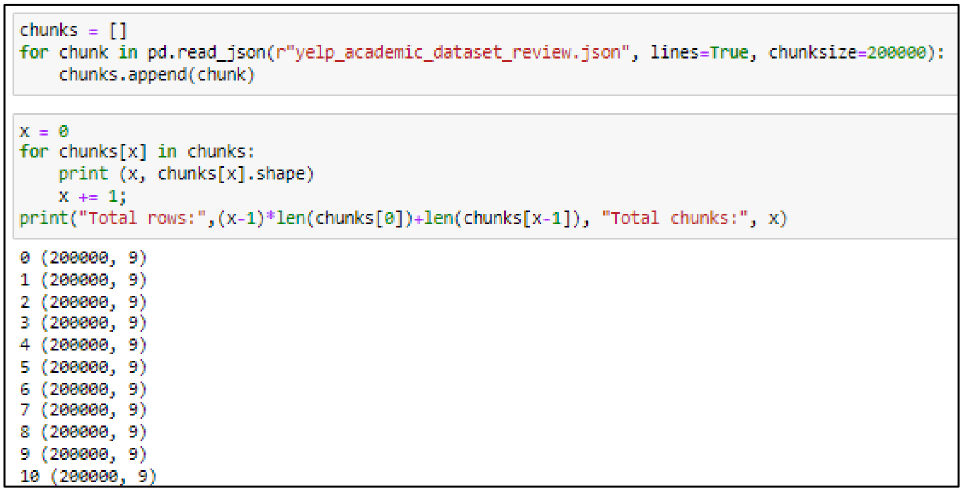
Build on your project proposal (from Milestone 1) that described the client or dataset you chose, the approach you were going to take, your initial hypotheses, and your initial approach. Include descriptive stats and any visualizations from your data exploration. You want to highlight key learnings from your data exploration and any Aha’s or changes to your plan as a results of your findings:

For a quick recap, I chose to use the Yelp dataset which can be found here https://www.yelp.com/dataset. I also used the following Python libraries to create data frames for and query the data using SQL.



```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt

from pandasql import sqldf
pysqldf = lambda q: sqldf(q, globals())
```

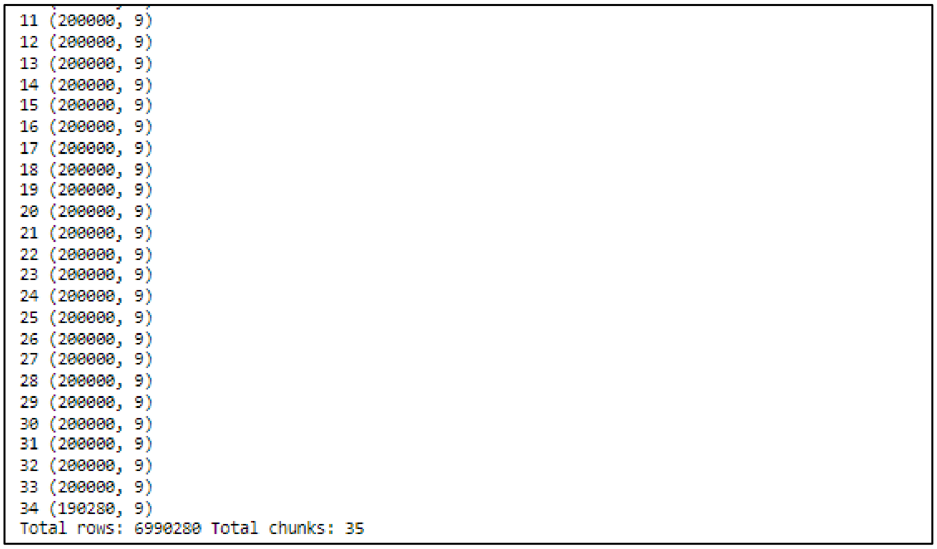
In particular I wound up using the ‘review’ table from the dataset as I believed it could provide useful insight on how customers appreciate and evaluate certain businesses. However, the review table was almost 7 million rows, so I had to break the table up into “chunks” as I kept running out of memory trying to store the entire table into one data frame. I also chose to store these chunks into an array so that they could be more easily accessed and evaluated. See below for code and initial data exploration.



```
chunks = []
for chunk in pd.read_json(r"yelp_academic_dataset_review.json", lines=True, chunksize=200000):
    chunks.append(chunk)
```

```
x = 0
for chunks[x] in chunks:
    print (x, chunks[x].shape)
    x += 1;
print("Total rows:",(x-1)*len(chunks[0])+len(chunks[x-1]), "Total chunks:", x)
```

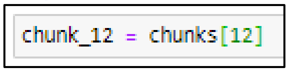
```
0 (200000, 9)
1 (200000, 9)
2 (200000, 9)
3 (200000, 9)
4 (200000, 9)
5 (200000, 9)
6 (200000, 9)
7 (200000, 9)
8 (200000, 9)
9 (200000, 9)
10 (200000, 9)
```



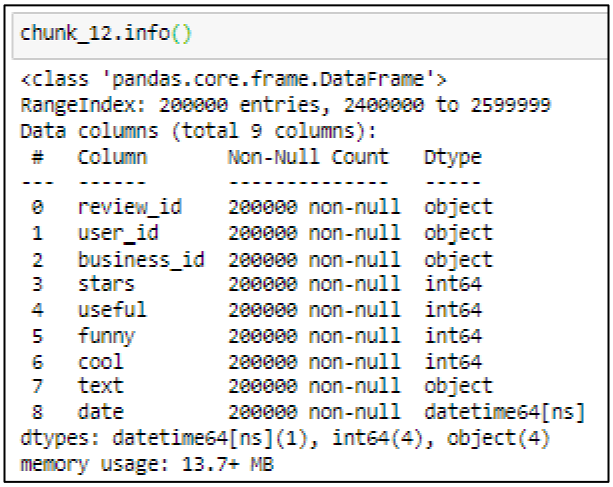
```
11 (200000, 9)
12 (200000, 9)
13 (200000, 9)
14 (200000, 9)
15 (200000, 9)
16 (200000, 9)
17 (200000, 9)
18 (200000, 9)
19 (200000, 9)
20 (200000, 9)
21 (200000, 9)
22 (200000, 9)
23 (200000, 9)
24 (200000, 9)
25 (200000, 9)
26 (200000, 9)
27 (200000, 9)
28 (200000, 9)
29 (200000, 9)
30 (200000, 9)
31 (200000, 9)
32 (200000, 9)
33 (200000, 9)
34 (190280, 9)
Total rows: 6990280 Total chunks: 35
```

As you can see, even after breaking up the table into chunks of 200,000 rows each, there still ended up being 35 data frames/chunks.

For my analysis I chose chunk 12, my birthday is 12/12 so I am partial to the number.



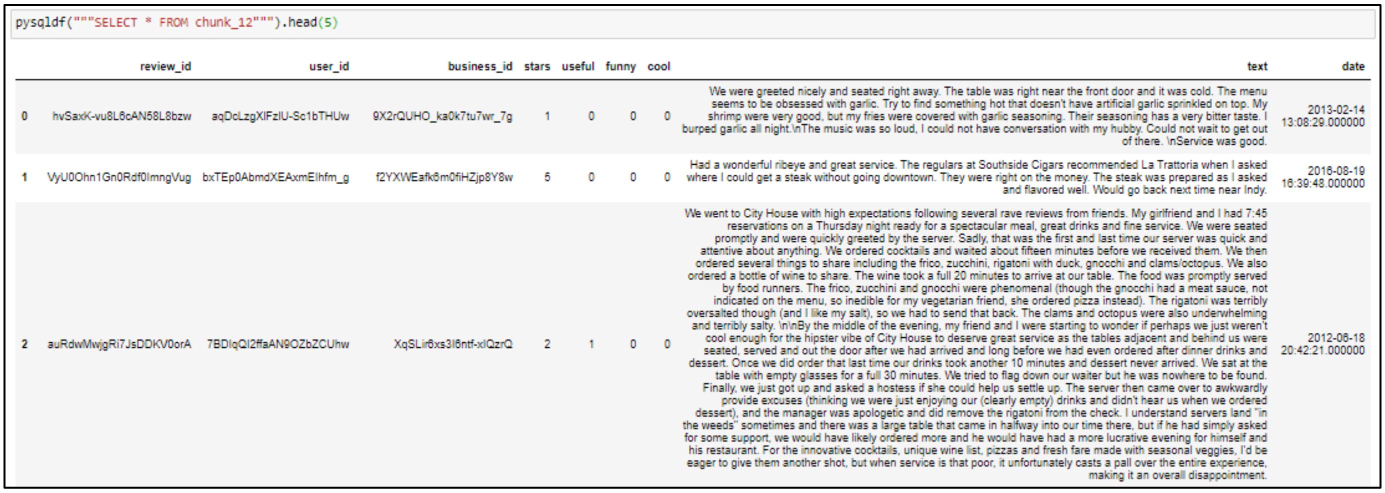
```
chunk_12 = chunks[12]
```



```
chunk_12.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200000 entries, 2400000 to 2599999
Data columns (total 9 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   review_id    200000 non-null  object
 1   user_id      200000 non-null  object
 2   business_id  200000 non-null  object
 3   stars        200000 non-null  int64
 4   useful       200000 non-null  int64
 5   funny        200000 non-null  int64
 6   cool         200000 non-null  int64
 7   text         200000 non-null  object
 8   date         200000 non-null  datetime64[ns]
dtypes: datetime64[ns](1), int64(4), object(4)
memory usage: 13.7+ MB
```

Per the screenshot above, we can see that the review table is comprised of 9 columns, containing certain the primary key 'review_id', the foreign keys 'user_id' and 'business_id', and then several fields describing the review. Below is a snippet of the data within these fields.



```
pysqldf("""SELECT * FROM chunk_12""").head(5)
```

	review_id	user_id	business_id	stars	useful	funny	cool	text	date
0	hvSaxK-vu8L6cAN58L8bzw	aqDcLzgXIFzIU-Sc1bTHUw	9X2rQUHO_ka0k7tu7wr_7g	1	0	0	0	We were greeted nicely and seated right away. The table was right near the front door and it was cold. The menu seems to be obsessed with garlic. Try to find something hot that doesn't have artificial garlic sprinkled on top. My shrimp were very good, but my fries were covered with garlic seasoning. Their seasoning has a very bitter taste. I burped garlic all night.\nThe music was so loud, I could not have conversation with my hubby. Could not wait to get out of there. \nService was good.	2013-02-14 13:08:29.000000
1	VyU0Ohn1Gn0Rdf0ImngVug	bxTEp0AbmdXEAxmElhfm_g	f2YXWEafk6m0fiHZjp8Y8w	5	0	0	0	Had a wonderful ribeye and great service. The regulars at Southside Cigars recommended La Trattoria when I asked where I could get a steak without going downtown. They were right on the money. The steak was prepared as I asked and flavored well. Would go back next time near Indy.	2016-08-19 16:39:48.000000
2	auRdwMwjgRi7JsDDKV0orA	7BDIqQI2ffaAN9OZbZCUhw	XqSLir6xs3l6ntf-xIQzrQ	2	1	0	0	We went to City House with high expectations following several rave reviews from friends. My girlfriend and I had 7:45 reservations on a Thursday night ready for a spectacular meal, great drinks and fine service. We were seated promptly and were quickly greeted by the server. Sadly, that was the first and last time our server was quick and attentive about anything. We ordered cocktails and waited about fifteen minutes before we received them. We then ordered several things to share including the frico, zucchini, rigatoni with duck, gnocchi and clams/octopus. We also ordered a bottle of wine to share. The wine took a full 20 minutes to arrive at our table. The food was promptly served by food runners. The frico, zucchini and gnocchi were phenomenal (though the gnocchi had a meat sauce, not indicated on the menu, so inedible for my vegetarian friend, she ordered pizza instead). The rigatoni was terribly oversalted though (and I like my salt), so we had to send that back. The clams and octopus were also underwhelming and terribly salty. \n\nBy the middle of the evening, my friend and I were starting to wonder if perhaps we just weren't cool enough for the hipster vibe of City House to deserve great service as the tables adjacent and behind us were seated, served and out the door after we had arrived and long before we had even ordered after dinner drinks and dessert. Once we did order that last time our drinks took another 10 minutes and dessert never arrived. We sat at the table with empty glasses for a full 30 minutes. We tried to flag down our waiter but he was nowhere to be found. Finally, we just got up and asked a hostess if she could help us settle up. The server then came over to awkwardly provide excuses (thinking we were just enjoying our (clearly empty) drinks and didn't hear us when we ordered dessert), and the manager was apologetic and did remove the rigatoni from the check. I understand servers land "in the weeds" sometimes and there was a large table that came in halfway into our time there, but if he had simply asked for some support, we would have likely ordered more and he would have had a more lucrative evening for himself and his restaurant. For the innovative cocktails, unique wine list, pizzas and fresh fare made with seasonal veggies, I'd be eager to give them another shot, but when service is that poor, it unfortunately casts a pall over the entire experience, making it an overall disappointment.	2012-06-18 20:42:21.000000

- Include Client/Hypotheses/Approach – fictional coffee company CoffeeKing, using the Yelp dataset I formed the hypothesis that the 'text' field for a review within the review table would have a relationship with the number of 'stars' for that review, which is essentially the rating a user is giving a business, and that the fictional client could leverage this relationship for business decision making.
- Include artifacts from previous modules – A/B testing from course 2 in the specialization "***Data Wrangling, Analysis and A/B Testing with SQL***".
- Include results (good and bad paths); Correlations / regressions – As can be seen in the insights discovered section below correlations were discovered between several keywords in the text of a review and the number of stars for said review.
- Graphics / Visualizations – Please see the insights discovered section below.

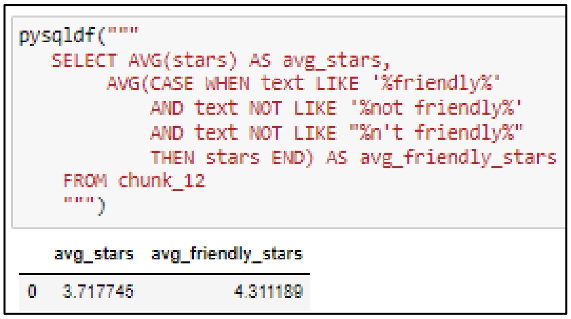
Discuss Insights Discovered

Discuss insights discovered (results from your diving deeper / going broader analysis). This is where you put your spin on what you've discovered.

- Discuss your hypotheses and any direct outcomes from whether you were right or wrong. Did you change your hypotheses? Or create new ones?

One of my original hypotheses was that certain keywords, typically those with a positive connotation, used in a review would correlate with a positive rating (number of stars). As we can see below, the average number of stars for reviews containing the words 'friendly', 'clean', or 'tasty' is higher than the baseline average number of stars for a population of 200,000 reviews. Please see screenshots below.

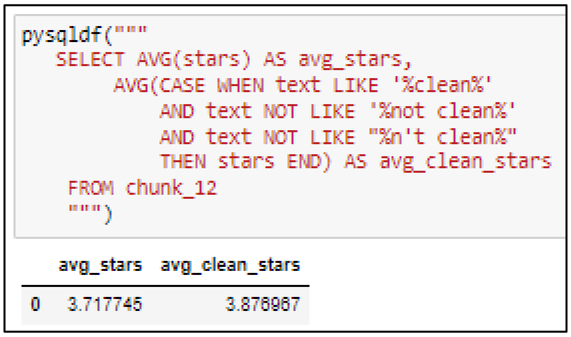
Here is a comparison of the average stars for all reviews and then the average number of stars for reviews that contain the word 'friendly', but I also made sure that it did not contain reviews that had the word combinations 'not friendly' or 'wasn't friendly' or 'weren't friendly'.



```
pysqldf("""
   SELECT AVG(stars) AS avg_stars,
        AVG(CASE WHEN text LIKE '%friendly%'
            AND text NOT LIKE '%not friendly%'
            AND text NOT LIKE "%n't friendly%"
            THEN stars END) AS avg_friendly_stars
    FROM chunk_12
    """)
```

```
   avg_stars  avg_friendly_stars
0   3.717745            4.311189
```

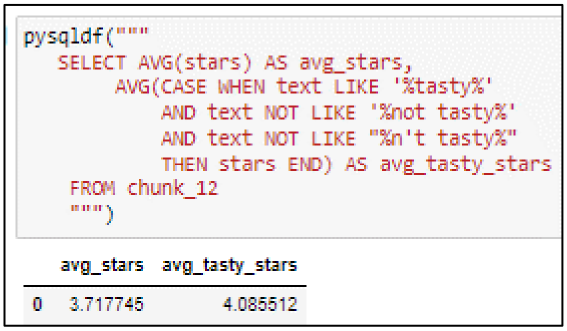
Here is the average and coding used for the keyword 'clean'.



```
pysqldf("""
   SELECT AVG(stars) AS avg_stars,
        AVG(CASE WHEN text LIKE '%clean%'
            AND text NOT LIKE '%not clean%'
            AND text NOT LIKE "%n't clean%"
            THEN stars END) AS avg_clean_stars
    FROM chunk_12
    """)
```

```
   avg_stars  avg_clean_stars
0   3.717745         3.878967
```

Here is the average and coding used for the keyword 'tasty'.

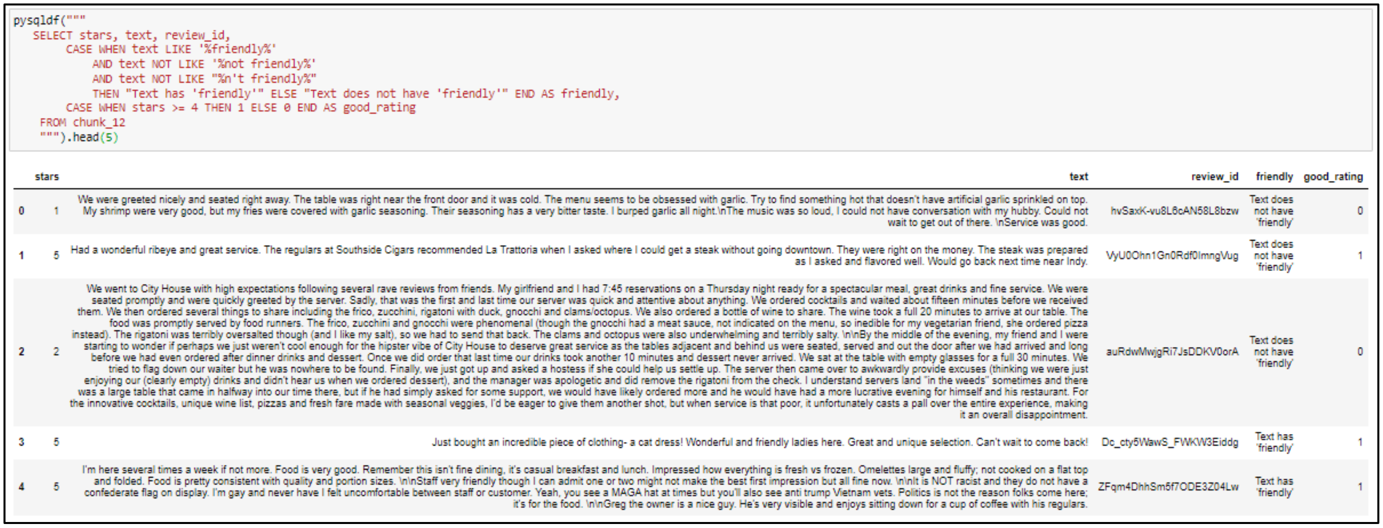


```
pysqldf("""
   SELECT AVG(stars) AS avg_stars,
        AVG(CASE WHEN text LIKE '%tasty%'
            AND text NOT LIKE '%not tasty%'
            AND text NOT LIKE "%n't tasty%"
            THEN stars END) AS avg_tasty_stars
    FROM chunk_12
    """)
```

```
   avg_stars  avg_tasty_stars
0   3.717745         4.085512
```

- Discuss any metrics you created and why?

Yes, I created a binary metric called ‘good_rating’, which returns a 1 if the number of stars for a review is 4 or higher, and a 0 if it is not. This rating classification is ultimately subjective in nature, but given that the maximum number of stars 5, I felt that most people would consider 4 or more stars as good, and anything less than that would be an average or bad rating. I also made an additional metric that was modified for each keyword, it basically was a column stating whether or not the keyword I was looking for was contained within the text for a review. Below is a screenshot of these metrics being added to a table for the ‘friendly’ keyword.



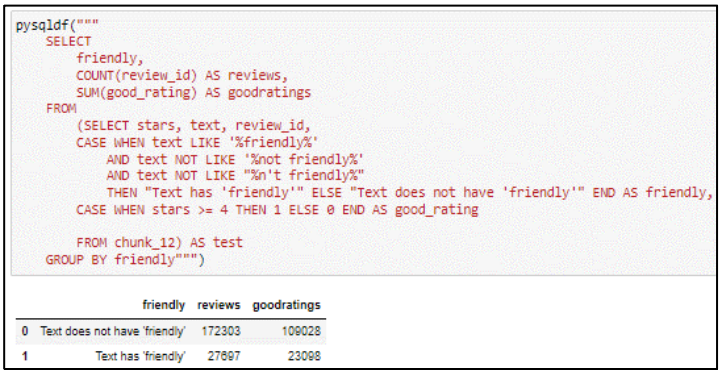
```
pysqldf("""
   SELECT stars, text, review_id,
        CASE WHEN text LIKE '%friendly%'
            AND text NOT LIKE '%not friendly%'
            AND text NOT LIKE "%n't friendly%"
            THEN "Text has 'friendly'" ELSE "Text does not have 'friendly'" END AS friendly,
        CASE WHEN stars >= 4 THEN 1 ELSE 0 END AS good_rating
    FROM chunk_12
    """).head(5)
```

	stars	text	review_id	friendly	good_rating
0	1	We were greeted nicely and seated right away. The table was right near the front door and it was cold. The menu seems to be obsessed with garlic. Try to find something hot that doesn't have artificial garlic sprinkled on top. My shrimp were very good, but my fries were covered with garlic seasoning. Their seasoning has a very bitter taste. I burped garlic all night.\nThe music was so loud, I could not have conversation with my hubby. Could not wait to get out of there. \nService was good.	hvSaxK-vu8L6cAN58L8bzw	Text does not have 'friendly'	0
1	5	Had a wonderful ribeye and great service. The regulars at Southside Cigars recommended La Trattoria when I asked where I could get a steak without going downtown. They were right on the money. The steak was prepared as I asked and flavored well. Would go back next time near Indy.	VyU0Ohn1Gn0Rdf0ImngVug	Text does not have 'friendly'	1
2	2	We went to City House with high expectations following several rave reviews from friends. My girlfriend and I had 7:45 reservations on a Thursday night ready for a spectacular meal, great drinks and fine service. We were seated promptly and were quickly greeted by the server. Sadly, that was the first and last time our server was quick and attentive about anything. We ordered cocktails and waited about fifteen minutes before we received them. We then ordered several things to share including the frico, zucchini, rigatoni with duck, gnocchi and clams/octopus. We also ordered a bottle of wine to share. The wine took a full 20 minutes to arrive at our table. The food was promptly served by food runners. The frico, zucchini and gnocchi were phenomenal (though the gnocchi had a meat sauce, not indicated on the menu, so inedible for my vegetarian friend, she ordered pizza instead). The rigatoni was terribly oversalted though (and I like my salt), so we had to send that back. The clams and octopus were also underwhelming and terribly salty. \n\nBy the middle of the evening, my friend and I were starting to wonder if perhaps we just weren't cool enough for the hipster vibe of City House to deserve great service as the tables adjacent and behind us were seated, served and out the door after we had arrived and long before we had even ordered after dinner drinks and dessert. Once we did order that last time our drinks took another 10 minutes and dessert never arrived. We sat at the table with empty glasses for a full 30 minutes. We tried to flag down our waiter but he was nowhere to be found. Finally, we just got up and asked a hostess if she could help us settle up. The server then came over to awkwardly provide excuses (thinking we were just enjoying our (clearly empty) drinks and didn't hear us when we ordered dessert), and the manager was apologetic and did remove the rigatoni from the check. I understand servers land "in the weeds" sometimes and there was a large table that came in halfway into our time there, but if he had simply asked for some support, we would have likely ordered more and he would have had a more lucrative evening for himself and his restaurant. For the innovative cocktails, unique wine list, pizzas and fresh fare made with seasonal veggies, I'd be eager to give them another shot, but when service is that poor, it unfortunately casts a pall over the entire experience, making it an overall disappointment.	auRdwMwjgRi7JsDDKV0orA	Text does not have 'friendly'	0
3	5	Just bought an incredible piece of clothing- a cat dress! Wonderful and friendly ladies here. Great and unique selection. Can't wait to come back!	Dc_cty5WawS_FWKW3Eiddg	Text has 'friendly'	1
4	5	I'm here several times a week if not more. Food is very good. Remember this isn't fine dining, it's casual breakfast and lunch. Impressed how everything is fresh vs frozen. Omelettes large and fluffy; not cooked on a flat top and folded. Food is pretty consistent with quality and portion sizes. \n\nStaff very friendly though I can admit one or two might not make the best first impression but all fine now. \n\nIt is NOT racist and they do not have a confederate flag on display. I'm gay and never have I felt uncomfortable between staff or customer. Yeah, you see a MAGA hat at times but you'll also see anti trump Vietnam vets. Politics is not the reason folks come here; it's for the food. \n\nGreg the owner is a nice guy. He's very visible and enjoys sitting down for a cup of coffee with his regulars.	ZFqm4DhhSm5f7ODE3Z04Lw	Text has 'friendly'	1

- Discuss discoveries about relationships in the data / themes discovered.

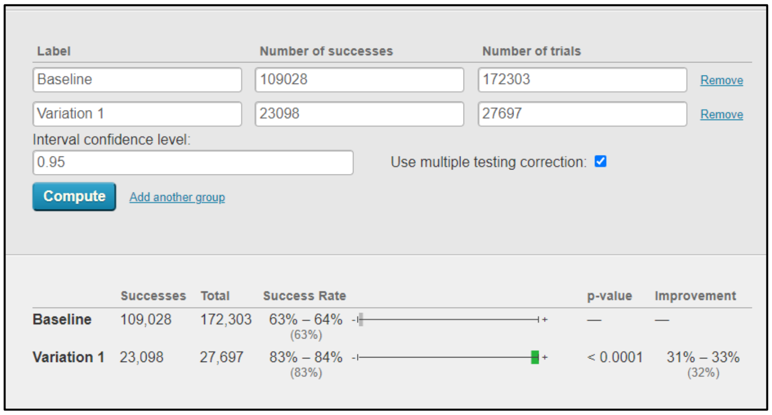
I used these metrics to create AB testing for each word, and then used the results of that testing and plugged them into the AB testing GitHub site, https://thumbtack.github.io/abba/demo/abba.html, we used in the second course in this specialization. I also used the results from the A/B testing to create 100% stacked bar charts for each word so it would be more visually apparent how much the portion of reviews with certain words had good ratings compared to reviews without those words.

Here are the AB testing results for the ‘friendly’ keyword.



```
pysqldf("""
    SELECT
        friendly,
        COUNT(review_id) AS reviews,
        SUM(good_rating) AS goodratings
    FROM
        (SELECT stars, text, review_id,
        CASE WHEN text LIKE '%friendly%'
            AND text NOT LIKE '%not friendly%'
            AND text NOT LIKE "%n't friendly%"
            THEN "Text has 'friendly'" ELSE "Text does not have 'friendly'" END AS friendly,
        CASE WHEN stars >= 4 THEN 1 ELSE 0 END AS good_rating

        FROM chunk_12) AS test
    GROUP BY friendly""")
```

	friendly	reviews	goodratings
0	Text does not have 'friendly'	172303	109028
1	Text has 'friendly'	27697	23098



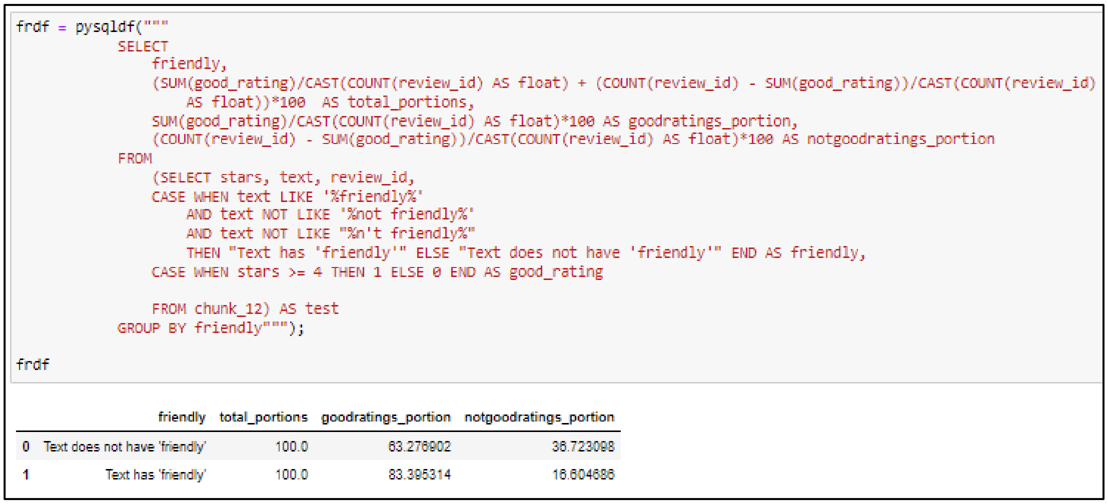
Label	Number of successes	Number of trials	
Baseline	109028	172303	Remove
Variation 1	23098	27697	Remove

Interval confidence level: 0.95

Use multiple testing correction: ☑

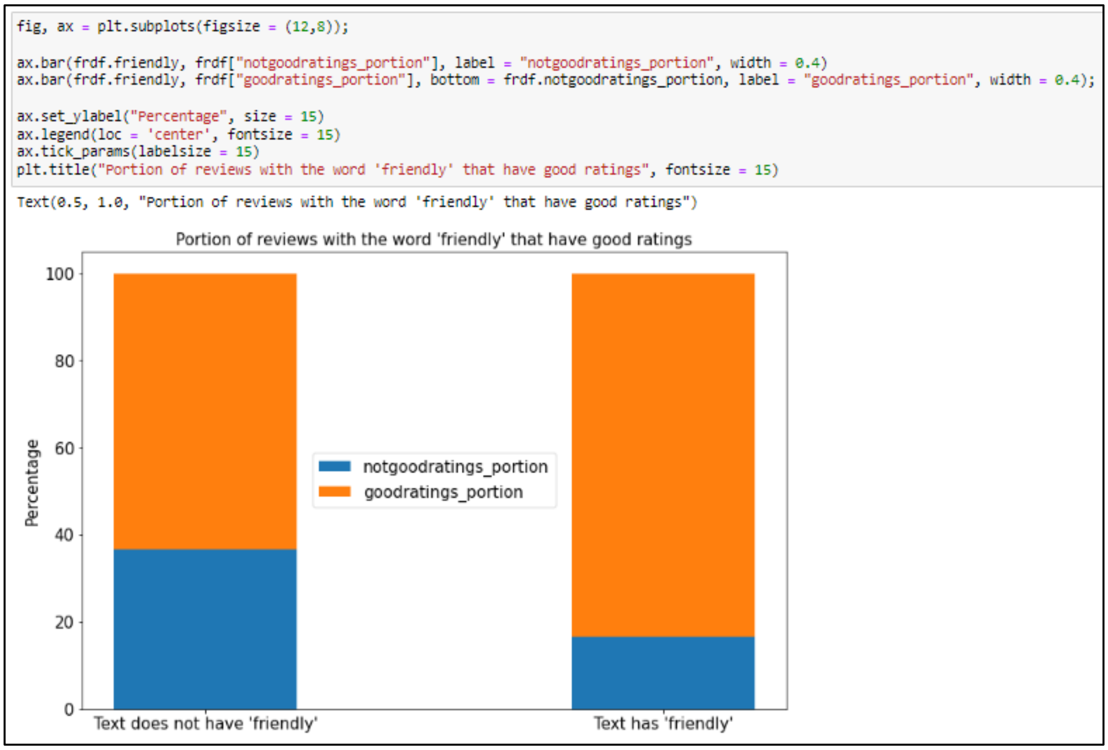
	Successes	Total	Success Rate	p-value	Improvement
Baseline	109,028	172,303	63% – 64% (63%)	—	—
Variation 1	23,098	27,697	83% – 84% (83%)	< 0.0001	31% – 33% (32%)

Here is the stacked bar chart, and its code, for the ‘friendly’ A/B testing.



```
frdf = pysqldf("""
            SELECT
                friendly,
                (SUM(good_rating)/CAST(COUNT(review_id) AS float) + (COUNT(review_id) - SUM(good_rating))/CAST(COUNT(review_id)
                    AS float))*100  AS total_portions,
                SUM(good_rating)/CAST(COUNT(review_id) AS float)*100 AS goodratings_portion,
                (COUNT(review_id) - SUM(good_rating))/CAST(COUNT(review_id) AS float)*100 AS notgoodratings_portion
            FROM
                (SELECT stars, text, review_id,
                CASE WHEN text LIKE '%friendly%'
                    AND text NOT LIKE '%not friendly%'
                    AND text NOT LIKE "%n't friendly%"
                    THEN "Text has 'friendly'" ELSE "Text does not have 'friendly'" END AS friendly,
                CASE WHEN stars >= 4 THEN 1 ELSE 0 END AS good_rating

                FROM chunk_12) AS test
            GROUP BY friendly""");

frdf
```

	friendly	total_portions	goodratings_portion	notgoodratings_portion
0	Text does not have 'friendly'	100.0	63.276902	36.723098
1	Text has 'friendly'	100.0	83.395314	16.604686

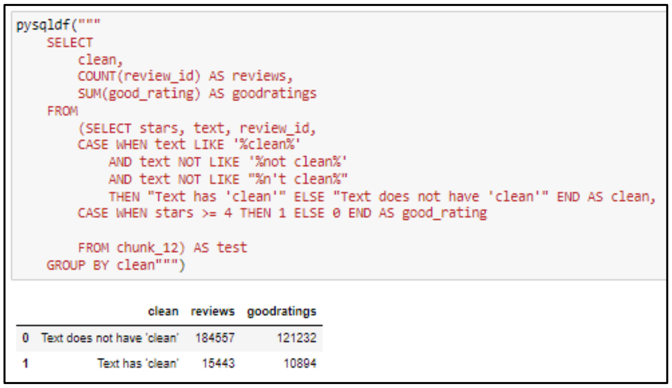


```
fig, ax = plt.subplots(figsize = (12,8));

ax.bar(frdf.friendly, frdf["notgoodratings_portion"], label = "notgoodratings_portion", width = 0.4)
ax.bar(frdf.friendly, frdf["goodratings_portion"], bottom = frdf.notgoodratings_portion, label = "goodratings_portion", width = 0.4);

ax.set_ylabel("Percentage", size = 15)
ax.legend(loc = 'center', fontsize = 15)
ax.tick_params(labelsize = 15)
plt.title("Portion of reviews with the word 'friendly' that have good ratings", fontsize = 15)
```

```
Text(0.5, 1.0, "Portion of reviews with the word 'friendly' that have good ratings")
```

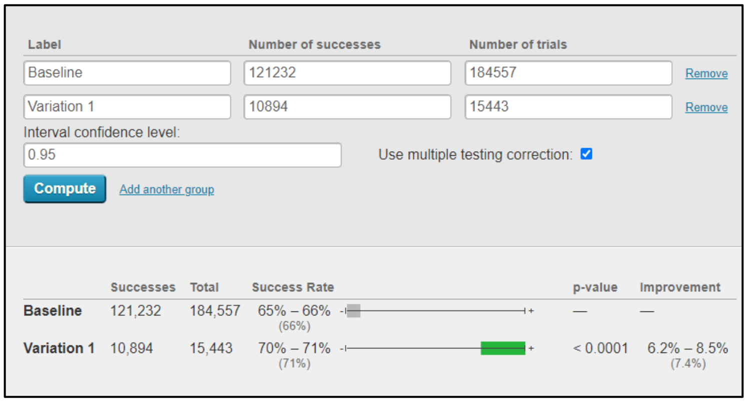
Portion of reviews with the word 'friendly' that have good ratings
Percentage
0
20
40
60
80
100
Text does not have 'friendly'
Text has 'friendly'
notgoodratings_portion
goodratings_portion

Here are the results of the 'clean' A/B testing once plugged into the GitHub site.



```
pysqldf("""
    SELECT
        clean,
        COUNT(review_id) AS reviews,
        SUM(good_rating) AS goodratings
    FROM
        (SELECT stars, text, review_id,
        CASE WHEN text LIKE '%clean%'
            AND text NOT LIKE '%not clean%'
            AND text NOT LIKE "%n't clean%"
            THEN "Text has 'clean'" ELSE "Text does not have 'clean'" END AS clean,
        CASE WHEN stars >= 4 THEN 1 ELSE 0 END AS good_rating

        FROM chunk_12) AS test
    GROUP BY clean""")
```

	clean	reviews	goodratings
0	Text does not have 'clean'	184557	121232
1	Text has 'clean'	15443	10894



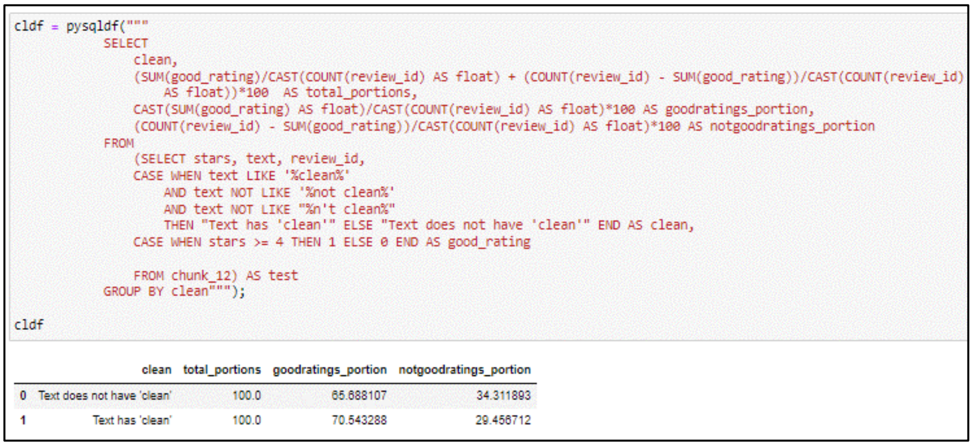
Label	Number of successes	Number of trials	
Baseline	121232	184557	Remove
Variation 1	10894	15443	Remove

Interval confidence level: 0.95

Use multiple testing correction: ☑

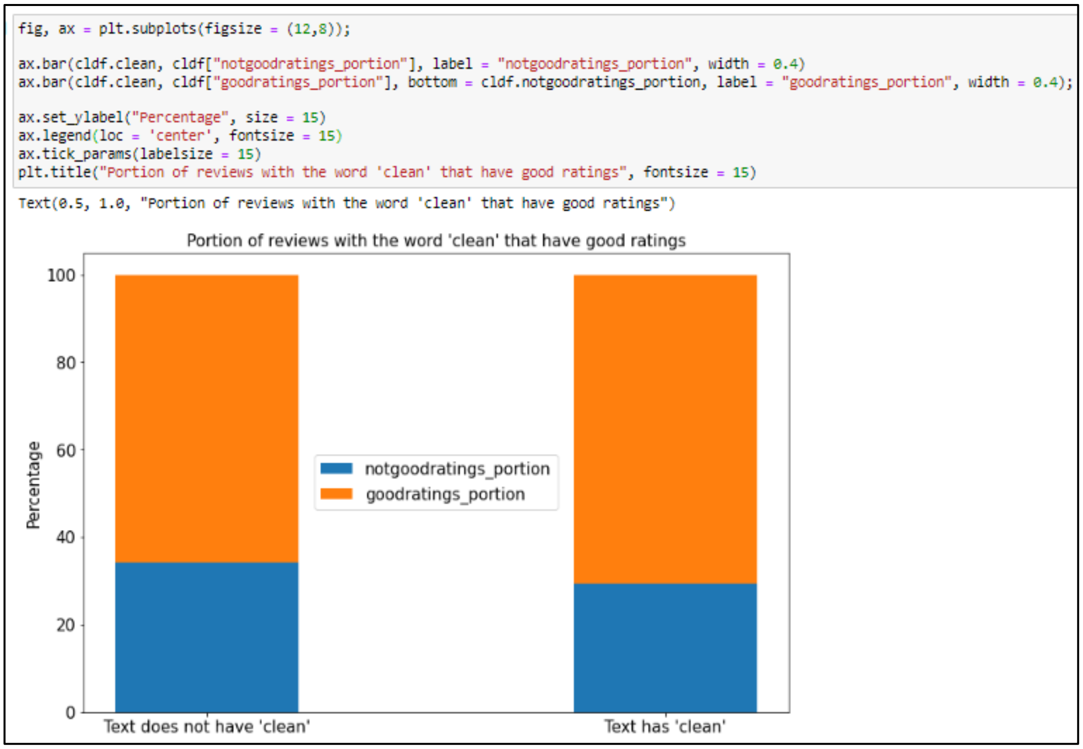
	Successes	Total	Success Rate	p-value	Improvement
Baseline	121,232	184,557	65% – 66% (66%)	—	—
Variation 1	10,894	15,443	70% – 71% (71%)	< 0.0001	6.2% – 8.5% (7.4%)

Here is the stacked bar chart, and its code, for the 'clean' A/B testing.



```
cldf = pysqldf("""
            SELECT
                clean,
                (SUM(good_rating)/CAST(COUNT(review_id) AS float) + (COUNT(review_id) - SUM(good_rating))/CAST(COUNT(review_id)
                    AS float))*100  AS total_portions,
                CAST(SUM(good_rating) AS float)/CAST(COUNT(review_id) AS float)*100 AS goodratings_portion,
                (COUNT(review_id) - SUM(good_rating))/CAST(COUNT(review_id) AS float)*100 AS notgoodratings_portion
            FROM
                (SELECT stars, text, review_id,
                CASE WHEN text LIKE '%clean%'
                    AND text NOT LIKE '%not clean%'
                    AND text NOT LIKE "%n't clean%"
                    THEN "Text has 'clean'" ELSE "Text does not have 'clean'" END AS clean,
                CASE WHEN stars >= 4 THEN 1 ELSE 0 END AS good_rating

                FROM chunk_12) AS test
            GROUP BY clean""");

cldf
```

	clean	total_portions	goodratings_portion	notgoodratings_portion
0	Text does not have 'clean'	100.0	65.688107	34.311893
1	Text has 'clean'	100.0	70.543288	29.456712

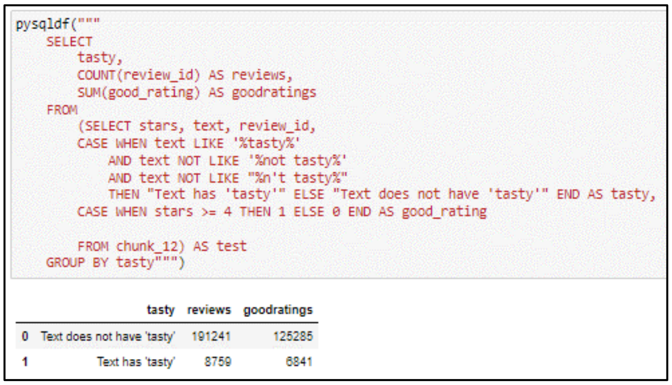


```
fig, ax = plt.subplots(figsize = (12,8));

ax.bar(cldf.clean, cldf["notgoodratings_portion"], label = "notgoodratings_portion", width = 0.4)
ax.bar(cldf.clean, cldf["goodratings_portion"], bottom = cldf.notgoodratings_portion, label = "goodratings_portion", width = 0.4);

ax.set_ylabel("Percentage", size = 15)
ax.legend(loc = 'center', fontsize = 15)
ax.tick_params(labelsize = 15)
plt.title("Portion of reviews with the word 'clean' that have good ratings", fontsize = 15)
```

```
Text(0.5, 1.0, "Portion of reviews with the word 'clean' that have good ratings")
```

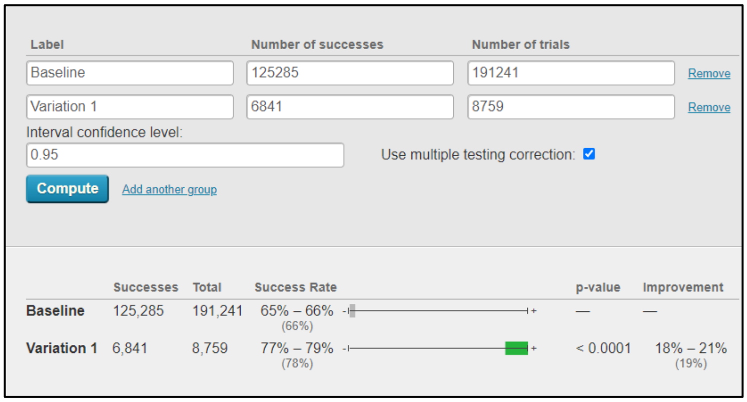
Portion of reviews with the word 'clean' that have good ratings
Percentage
0
20
40
60
80
100
notgoodratings_portion
goodratings_portion
Text does not have 'clean'
Text has 'clean'

Here are the results of the ‘tasty’ A/B testing once plugged into the GitHub site.



```
pysqldf("""
    SELECT
        tasty,
        COUNT(review_id) AS reviews,
        SUM(good_rating) AS goodratings
    FROM
        (SELECT stars, text, review_id,
        CASE WHEN text LIKE '%tasty%'
            AND text NOT LIKE '%not tasty%'
            AND text NOT LIKE "%n't tasty%"
            THEN "Text has 'tasty'" ELSE "Text does not have 'tasty'" END AS tasty,
        CASE WHEN stars >= 4 THEN 1 ELSE 0 END AS good_rating

        FROM chunk_12) AS test
    GROUP BY tasty""")
```

	tasty	reviews	goodratings
0	Text does not have 'tasty'	191241	125285
1	Text has 'tasty'	8759	6841



Label	Number of successes	Number of trials	
Baseline	125285	191241	Remove
Variation 1	6841	8759	Remove

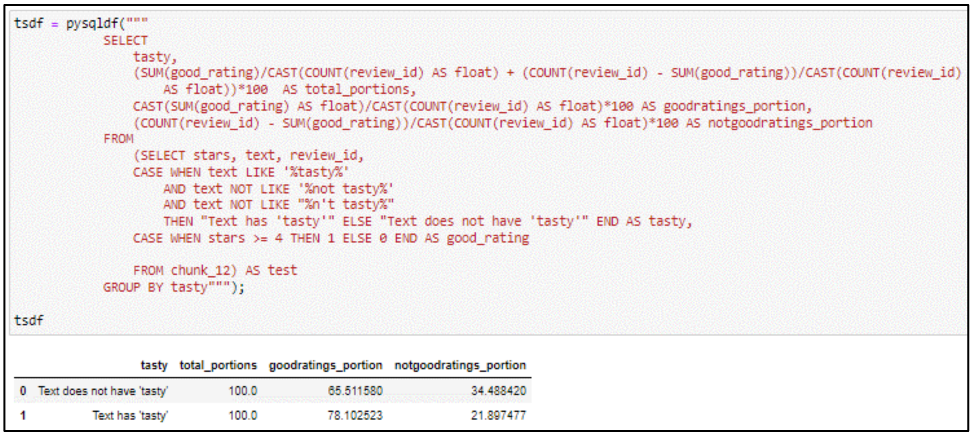
Interval confidence level: 0.95

Use multiple testing correction: ☑

Compute   Add another group

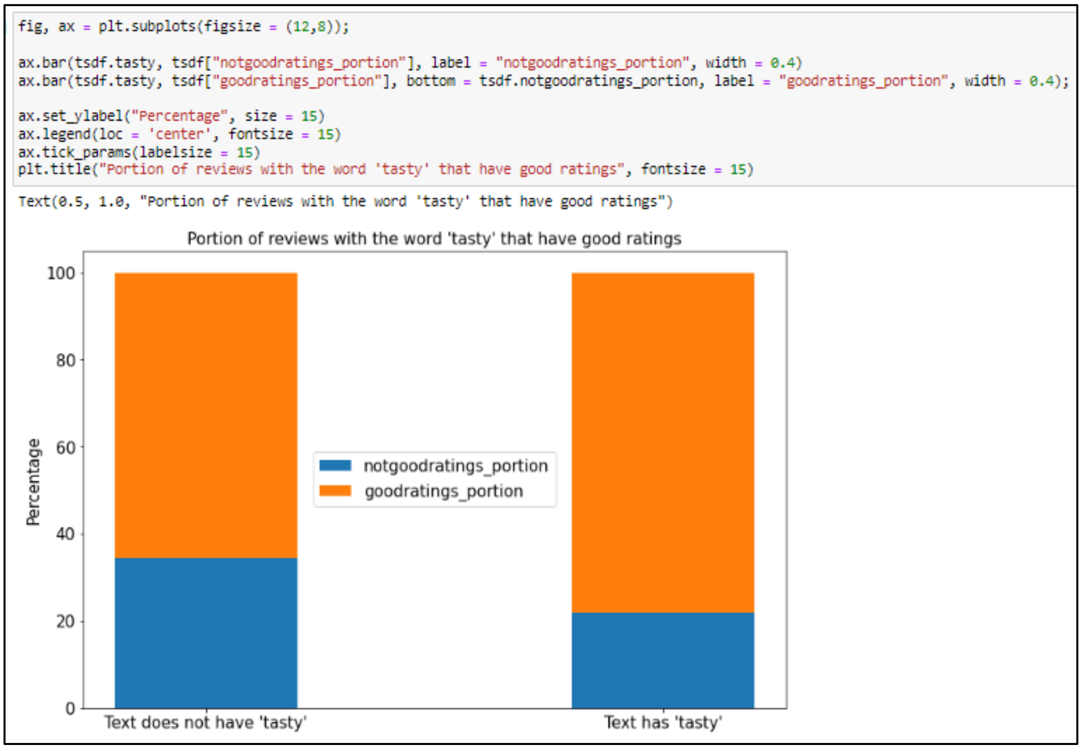
	Successes	Total	Success Rate	p-value	Improvement
Baseline	125,285	191,241	65% – 66% (66%)	—	—
Variation 1	6,841	8,759	77% – 79% (78%)	< 0.0001	18% – 21% (19%)

Here is the stacked bar chart, and its code, for the ‘tasty’ A/B testing.



```
tsdf = pysqldf("""
            SELECT
                tasty,
                (SUM(good_rating)/CAST(COUNT(review_id) AS float) + (COUNT(review_id) - SUM(good_rating))/CAST(COUNT(review_id)
                    AS float))*100  AS total_portions,
                CAST(SUM(good_rating) AS float)/CAST(COUNT(review_id) AS float)*100 AS goodratings_portion,
                (COUNT(review_id) - SUM(good_rating))/CAST(COUNT(review_id) AS float)*100 AS notgoodratings_portion
            FROM
                (SELECT stars, text, review_id,
                CASE WHEN text LIKE '%tasty%'
                    AND text NOT LIKE '%not tasty%'
                    AND text NOT LIKE "%n't tasty%"
                    THEN "Text has 'tasty'" ELSE "Text does not have 'tasty'" END AS tasty,
                CASE WHEN stars >= 4 THEN 1 ELSE 0 END AS good_rating

                FROM chunk_12) AS test
            GROUP BY tasty""");

tsdf
```

	tasty	total_portions	goodratings_portion	notgoodratings_portion
0	Text does not have 'tasty'	100.0	65.511580	34.488420
1	Text has 'tasty'	100.0	78.102523	21.897477



```
fig, ax = plt.subplots(figsize = (12,8));

ax.bar(tsdf.tasty, tsdf["notgoodratings_portion"], label = "notgoodratings_portion", width = 0.4)
ax.bar(tsdf.tasty, tsdf["goodratings_portion"], bottom = tsdf.notgoodratings_portion, label = "goodratings_portion", width = 0.4);

ax.set_ylabel("Percentage", size = 15)
ax.legend(loc = 'center', fontsize = 15)
ax.tick_params(labelsize = 15)
plt.title("Portion of reviews with the word 'tasty' that have good ratings", fontsize = 15)
```

```
Text(0.5, 1.0, "Portion of reviews with the word 'tasty' that have good ratings")
```

Portion of reviews with the word 'tasty' that have good ratings
Percentage
0
20
40
60
80
100
notgoodratings_portion
goodratings_portion
Text does not have 'tasty'
Text has 'tasty'

As we can see from the A/B testing results on the previous pages, all of the keywords had a statistically significant effect on whether or not there were good ratings, with p-values of less than 0.0001 each. The 'friendly' keyword, however, had the most dramatic effect, with a 31%-33% improvement on good ratings compared to reviews without the keyword 'friendly' whereas 'clean' only has a 6%-8% improvement and 'tasty' has an 18%-21% improvement. Not only that, but the population of reviews with the keyword 'friendly' is much larger than the other two, with a total of 27,697 reviews, whereas 'clean' was contained within 15,443 reviews, and 'tasty' within 8,759 reviews, meaning that people felt the need to bring up how friendly staff were much more than they felt the need to mention how clean facilities were or how tasty food was, although for that last one it makes sense because not all businesses sell food.

Recommendations and Actions

Summarize the insights you found and make recommendations on what your client should do. What is the next steps or the action that should be taken as a result of your analysis?

Out of the three keywords chosen, all were statistically significant when it came to having a good rating, i.e., 4 or more stars. And while many customers do not take the time to actually read through Yelp reviews, they can see from a quick Google search the average number of stars a business has, so it would behoove business owners to take note of the verbiage used in the reviews of their business(es).

Another insight I had that became apparent later on in my analysis after thinking about the differences between the results of the different keywords, was that a machine can clean a business, it can make tasty food, but it cannot, at the time of writing this assignment, produce authentic friendliness on par with a human being. As businesses become more and more inclined to use machines and forms of A.I. to automate processes, it would benefit them to take heed and note that some aspects of what people like about their business(es) are the human aspects. Out of the 200,000 reviews analyzed, friendliness had a far greater impact on a positive rating than other aspects which could be reproduced by a machine.