In [103…
```
!pip install ipython-sql

!pip install sqlalchemy
```

```
Requirement already satisfied: ipython-sql in c:\users\r_m_l\anaconda3\lib\site-packages (0.5.0)
Requirement already satisfied: prettytable in c:\users\r_m_l\anaconda3\lib\site-packages (from ipython-sql) (3.10.0)
Requirement already satisfied: ipython in c:\users\r_m_l\anaconda3\lib\site-packages (from ipython-sql) (8.20.0)
Requirement already satisfied: sqlalchemy>=2.0 in c:\users\r_m_l\anaconda3\lib\site-packages (from ipython-sql) (2.0.25)
Requirement already satisfied: sqlparse in c:\users\r_m_l\anaconda3\lib\site-packages (from ipython-sql) (0.5.0)
Requirement already satisfied: six in c:\users\r_m_l\anaconda3\lib\site-packages (from ipython-sql) (1.16.0)
Requirement already satisfied: ipython-genutils in c:\users\r_m_l\anaconda3\lib\site-packages (from ipython-sql) (0.2.0)
Requirement already satisfied: typing-extensions>=4.6.0 in c:\users\r_m_l\anaconda3\lib\site-packages (from sqlalchemy>=2.0->ipython-sql) (4.9.0)
Requirement already satisfied: greenlet!=0.4.17 in c:\users\r_m_l\anaconda3\lib\site-packages (from sqlalchemy>=2.0->ipython-sql) (3.0.3)
Requirement already satisfied: decorator in c:\users\r_m_l\anaconda3\lib\site-packages (from ipython->ipython-sql) (5.1.1)
Requirement already satisfied: jedi>=0.16 in c:\users\r_m_l\anaconda3\lib\site-packages (from ipython->ipython-sql) (0.18.1)
Requirement already satisfied: matplotlib-inline in c:\users\r_m_l\anaconda3\lib\site-packages (from ipython->ipython-sql) (0.1.6)
Requirement already satisfied: prompt-toolkit<3.1.0,>=3.0.41 in c:\users\r_m_l\anaconda3\lib\site-packages (from ipython->ipython-sql) (3.0.43)
Requirement already satisfied: pygments>=2.4.0 in c:\users\r_m_l\anaconda3\lib\site-packages (from ipython->ipython-sql) (2.15.1)
Requirement already satisfied: stack-data in c:\users\r_m_l\anaconda3\lib\site-packages (from ipython->ipython-sql) (0.2.0)
Requirement already satisfied: traitlets>=5 in c:\users\r_m_l\anaconda3\lib\site-packages (from ipython->ipython-sql) (5.7.1)
Requirement already satisfied: colorama in c:\users\r_m_l\anaconda3\lib\site-packages (from ipython->ipython-sql) (0.4.6)
Requirement already satisfied: wcwidth in c:\users\r_m_l\anaconda3\lib\site-packages (from prettytable->ipython-sql) (0.2.5)
Requirement already satisfied: parso<0.9.0,>=0.8.0 in c:\users\r_m_l\anaconda3\lib\site-packages (from jedi>=0.16->ipython->ipython-sql) (0.8.3)
Requirement already satisfied: executing in c:\users\r_m_l\anaconda3\lib\site-packages (from stack-data->ipython->ipython-sql) (0.8.3)
Requirement already satisfied: asttokens in c:\users\r_m_l\anaconda3\lib\site-packages (from stack-data->ipython->ipython-sql) (2.0.5)
Requirement already satisfied: pure-eval in c:\users\r_m_l\anaconda3\lib\site-packages (from stack-data->ipython->ipython-sql) (0.2.2)
Requirement already satisfied: sqlalchemy in c:\users\r_m_l\anaconda3\lib\site-packages (2.0.25)
Requirement already satisfied: typing-extensions>=4.6.0 in c:\users\r_m_l\anaconda3\lib\site-packages (from sqlalchemy) (4.9.0)
Requirement already satisfied: greenlet!=0.4.17 in c:\users\r_m_l\anaconda3\lib\site-packages (from sqlalchemy) (3.0.3)
```

In [104…
```
!pip install psycopg2
```

```
Requirement already satisfied: psycopg2 in c:\users\r_m_l\anaconda3\lib\site-packages (2.9.9)
```

In [106…
```
%load_ext sql
```

```
The sql extension is already loaded. To reload it, use:
  %reload_ext sql
```

In [107…
```
%sql postgresql://postgres:connie@localhost:5432/nps
```

Net Promoter Score

There are 2 sheets of data in a database called score. Each sets represents information regarding a survey given to customers. The goal of this workbook is analyze the data and calculate the net promoter score and identify any outliers or patterns.

In [108…
```
%%sql
-- Looking at a sample of the sheet of data --
```

```
SELECT * FROM customer
LIMIT 10;
```

```
 * postgresql://postgres:***@localhost:5432/nps
10 rows affected.
```

Out[108…

id	created_at	is_premier	is_spam
1	2018-01-01	False	False
2	2018-01-01	False	False
3	2018-01-01	True	False
4	2018-01-01	False	False
5	2018-01-01	False	False
6	2018-01-01	False	False
7	2018-01-01	False	False
8	2018-01-01	False	False
9	2018-01-01	False	False
10	2018-01-01	False	False

In [109…

```
%%sql
-- From the output above, I can see an ID, a date and two boolean columns. I want to know what time frame is included in the data --

(SELECT * FROM customer
 Order by created_at ASC
 limit 5)
union all

(SELECT * FROM customer
 ORDER by created_at DESC
 limit 5);
```

```
 * postgresql://postgres:***@localhost:5432/nps
10 rows affected.
```

Out[109…

id	created_at	is_premier	is_spam
2	2018-01-01	False	False
3	2018-01-01	True	False
4	2018-01-01	False	False
5	2018-01-01	False	False
1	2018-01-01	False	False
186606	2018-12-30	True	False
186607	2018-12-30	False	False
186604	2018-12-30	False	False
186605	2018-12-30	False	False
186608	2018-12-30	False	False

In [110…

```
%%sql
-- Next I want to see the distinct values in my boolean columns --

SELECT DISTINCT is_premier, is_spam FROM customer;
```

```
 * postgresql://postgres:***@localhost:5432/nps
4 rows affected.
```

Out[110…

is_premier	is_spam
False	False
False	True
True	False
True	True

In [111…

```
%%sql
-- I also want to knwo how many rows of data I'll be looking at as well --

SELECT COUNT (*) FROM customer;
```

```
 * postgresql://postgres:***@localhost:5432/nps
1 rows affected.
```

Out[111…

count
188323

In [112…

```
%%sql
-- I am repeating the same steps above for the secong sheet of data, bringing in a sample of the data --

SELECT * FROM score
LIMIT 10;
```

```
 * postgresql://postgres:***@localhost:5432/nps
10 rows affected.
```

Out[112…

id	customer_id	created_at	score
1	1	2018-01-01	5
2	2	2018-01-01	0
3	3	2018-01-01	10
4	4	2018-01-01	6
5	5	2018-01-01	10
6	6	2018-01-01	0
7	7	2018-01-01	1
8	8	2018-01-01	2
9	9	2018-01-01	1
10	10	2018-01-01	10

In [113…

```
%%sql

-- From the output above, I have an id column, a customer_id and date column similar to the first sheet of data and a score --
-- Identifying the date range, do we have the same dates as the first sheet of data --

(SELECT * FROM score
 Order by created_at ASC
 limit 5)
union all

(SELECT * FROM score
 ORDER by created_at DESC
 limit 5);
```

```
 * postgresql://postgres:***@localhost:5432/nps
10 rows affected.
```

Out[113…

id	customer_id	created_at	score
2	2	2018-01-01	0
3	3	2018-01-01	10
4	4	2018-01-01	6
5	5	2018-01-01	10
1	1	2018-01-01	5
1556593	2121	2018-12-30	10
1556594	2136	2018-12-30	10
1556591	2093	2018-12-30	10
1556592	2108	2018-12-30	10
1556595	2139	2018-12-30	10

In [114…

```
%%sql

-- I want to know the lowest and highest score recorded --

(SELECT * FROM score
ORDER by score ASC
LIMIT 2)
union all

(SELECT * FROM score
ORDER by score DESC
LIMIT 2);
```

```
 * postgresql://postgres:***@localhost:5432/nps
4 rows affected.
```

Out[114…

id	customer_id	created_at	score
2	2	2018-01-01	0
6	6	2018-01-01	0
3	3	2018-01-01	10
5	5	2018-01-01	10

In [115…

```
%%sql

-- How many rows of data are included in the data set --
```

```
SELECT COUNT (*) FROM score;
```

```
 * postgresql://postgres:***@localhost:5432/nps
1 rows affected.
```

Out[115…

count
1577578

In [116…

```
%%sql

-- EDA: looking at the customers with the highest number of responses and total response count --

SELECT customer_id, COUNT (score.id) AS cnt FROM score
INNER JOIN customer ON customer_id = customer.id
GROUP BY customer_id ORDER BY cnt DESC
LIMIT 10;
```

```
 * postgresql://postgres:***@localhost:5432/nps
10 rows affected.
```

Out[116…

customer_id	cnt
31	38
928	38
4271	38
5333	37
1253	37
1259	36
1030	36
2327	36
564	36
2335	36

In [117…

```
%%sql

-- From the output above, the customer_ids 31, 928 and 4271 have the most response to the survey, but how many responses most customers leave --

SELECT cnt, COUNT (cnt) AS count_of_count FROM
(
    SELECT customer_id, count (score.id) AS cnt FROM score
    INNER JOIN customer ON customer_id = customer.id
    GROUP BY customer_id
```

```
) a
GROUP BY cnt
ORDER BY count_of_count DESC
LIMIT 100;
```

```
 * postgresql://postgres:***@localhost:5432/nps
38 rows affected.
```

Out[117…

cnt	count_of_count
6	18779
5	17218
7	17094
4	15642
8	14108
3	12983
9	11978
10	10556
2	10191
11	9001
12	7833
13	6698
1	6302
14	5707
15	4908
16	4059
17	3341
18	2749
19	2155
20	1751
21	1415
22	1041
23	779
24	574
25	424
26	318
27	220

cnt	count_of_count
28	186
29	112
30	73
31	46
33	25
32	25
34	12
35	9
36	6
38	3
37	2

In [118…

```
%%sql

--  From the output above, we can see that most most customers leave between 2-10 responses per survey, with 6 being the highest --
--  Next, since we have a date associated with each survey, on average what are the scores across all customers --

SELECT week, ROUND(AVG(average_weekly_score),2) AS average_score FROM
(
    SELECT TO_CHAR(score.created_at, 'IYYY-IW') AS week, customer_id, AVG(score) AS average_weekly_score FROM score
    GROUP BY week, customer_id
) a
GROUP BY week
ORDER BY week
LIMIT 100;
```

```
 * postgresql://postgres:***@localhost:5432/nps
52 rows affected.
```

Out[118...

week	average_score
2018-01	5.12
2018-02	5.80
2018-03	5.74
2018-04	5.50
2018-05	6.33
2018-06	7.02
2018-07	7.01
2018-08	6.89
2018-09	7.38
2018-10	7.75
2018-11	7.80
2018-12	7.81
2018-13	7.84
2018-14	7.95
2018-15	7.93
2018-16	7.92
2018-17	7.94
2018-18	8.03
2018-19	8.06
2018-20	7.99
2018-21	7.93
2018-22	8.06
2018-23	8.30
2018-24	8.32
2018-25	8.34
2018-26	8.36
2018-27	8.35

week	average_score
2018-28	8.34
2018-29	8.33
2018-30	8.29
2018-31	8.38
2018-32	8.48
2018-33	8.45
2018-34	8.45
2018-35	8.25
2018-36	7.75
2018-37	7.48
2018-38	7.38
2018-39	7.31
2018-40	7.69
2018-41	7.92
2018-42	7.94
2018-43	7.97
2018-44	8.04
2018-45	8.12
2018-46	8.16
2018-47	8.15
2018-48	8.17
2018-49	8.28
2018-50	8.31
2018-51	8.34
2018-52	8.35

In [119…

```
%%sql

-- Categorizing customers to calculate the net promoter score --
```

```
-- Any scores above 7 would be considered a promoter, above 5 is passive and anything else will be a distractor --

SELECT count(*) FROM
(
    SELECT CASE
        WHEN average_weekly_score > 7 THEN 'promoter'
        WHEN average_weekly_score > 5 THEN 'passive'
        ELSE 'detractor'
    END AS nps_class, week FROM
    (
        SELECT TO_CHAR(score.created_at, 'IYYY-IW') AS week, customer_id, AVG(score) AS average_weekly_score FROM score
        GROUP BY week, customer_id
    ) a
) b
LIMIT 10;
```

```
 * postgresql://postgres:***@localhost:5432/nps
1 rows affected.
```

Out[119…

count
951289

In [120…

```
%%sql

-- Categorizing customers to calculate the net promoter score --
-- Any scores above 7 would be considered a promoter, above 5 is passive and anything else will be a distractor --
-- Output will list how many in each category and the total column and list them by week and month --
-- Calculate the net promoter score --


SELECT *,
ROUND(((CAST(promoter AS DECIMAL) / total) - (CAST(detractor AS DECIMAL) / total)) * 100, 0) AS nps,
CEIL(CAST(SUBSTRING(week, 6, 8) AS DECIMAL)*12/52) AS month
FROM
(
    SELECT week,
    SUM(CASE WHEN nps_class = 'promoter' THEN 1 ELSE 0 END) AS "promoter",
    SUM(CASE WHEN nps_class = 'passive' THEN 1 ELSE 0 END) AS "passive",
    SUM(CASE WHEN nps_class = 'detractor' THEN 1 ELSE 0 END) AS "detractor",
        COUNT(*) AS "total"
    FROM

    (
        SELECT CASE
        WHEN average_weekly_score > 7 THEN 'promoter'
        WHEN average_weekly_score > 5 THEN 'passive'
        ELSE 'detractor'
        END AS nps_class, week
        FROM
```

```
        (
            SELECT TO_CHAR(score.created_at, 'IYYY-IW') AS week, customer_id, AVG(score) AS average_weekly_score FROM score
            GROUP BY week, customer_id
        ) a
    ) b
GROUP BY week
ORDER BY week
) c



limit 100;
```

```
 * postgresql://postgres:***@localhost:5432/nps
52 rows affected.
```

Out[120…

week	promoter	passive	detractor	total	nps	month
2018-01	26	6	33	65	-11	1
2018-02	65	10	55	130	8	1
2018-03	74	10	64	148	7	1
2018-04	80	9	76	165	2	1
2018-05	202	21	121	344	24	2
2018-06	431	37	187	655	37	2
2018-07	518	37	226	781	37	2
2018-08	565	48	262	875	35	2
2018-09	831	63	279	1173	47	3
2018-10	1258	82	319	1659	57	3
2018-11	1429	104	354	1887	57	3
2018-12	1562	98	386	2046	57	3
2018-13	1767	112	433	2312	58	3
2018-14	2195	177	482	2854	60	4
2018-15	2692	188	592	3472	60	4
2018-16	2873	208	637	3718	60	4
2018-17	3130	218	699	4047	60	4
2018-18	3658	248	758	4664	62	5
2018-19	4271	332	853	5456	63	5
2018-20	4494	348	949	5791	61	5
2018-21	4731	362	1031	6124	60	5
2018-22	5502	344	1110	6956	63	6
2018-23	6663	381	1106	8150	68	6
2018-24	7208	423	1170	8801	69	6
2018-25	7523	436	1196	9155	69	6
2018-26	8127	469	1239	9835	70	6
2018-27	9165	546	1443	11154	69	7

week	promoter	passive	detractor	total	nps	month
2018-28	10204	670	1614	12488	69	7
2018-29	10925	652	1771	13348	69	7
2018-30	11588	679	1915	14182	68	7
2018-31	13236	789	2022	16047	70	8
2018-32	15596	866	2165	18627	72	8
2018-33	16617	824	2412	19853	72	8
2018-34	17352	814	2491	20657	72	8
2018-35	18175	971	3169	22315	67	9
2018-36	18052	1035	4759	23846	56	9
2018-37	18185	1252	5776	25213	49	9
2018-38	19004	1311	6373	26688	47	9
2018-39	19552	1323	6862	27737	46	9
2018-40	22186	1440	6044	29670	54	10
2018-41	24298	1534	5556	31388	60	10
2018-42	25521	1607	5721	32849	60	10
2018-43	26678	1633	5892	34203	61	10
2018-44	29207	1717	6067	36991	63	11
2018-45	32375	1777	6321	40473	64	11
2018-46	34426	1755	6554	42735	65	11
2018-47	36005	1828	6865	44698	65	11
2018-48	39717	2071	7401	49189	66	12
2018-49	47968	2394	8166	58528	68	12
2018-50	53318	2617	8853	64788	69	12
2018-51	57003	2750	9146	68899	69	12
2018-52	60818	2915	9727	73460	70	12

Observation: In January, the net promoter scores were the lowest and has been an upward trend since February. In september the net promoter scores decreased but bounce back shortly after.

Furter analysis: Were there any processes changes between January and February to explain the higher engagement with the survey and the higher score ? Could the decrease in september be caused by seasonality, or was there a new product introduced that were not favorable to the customers.

In [ ]: