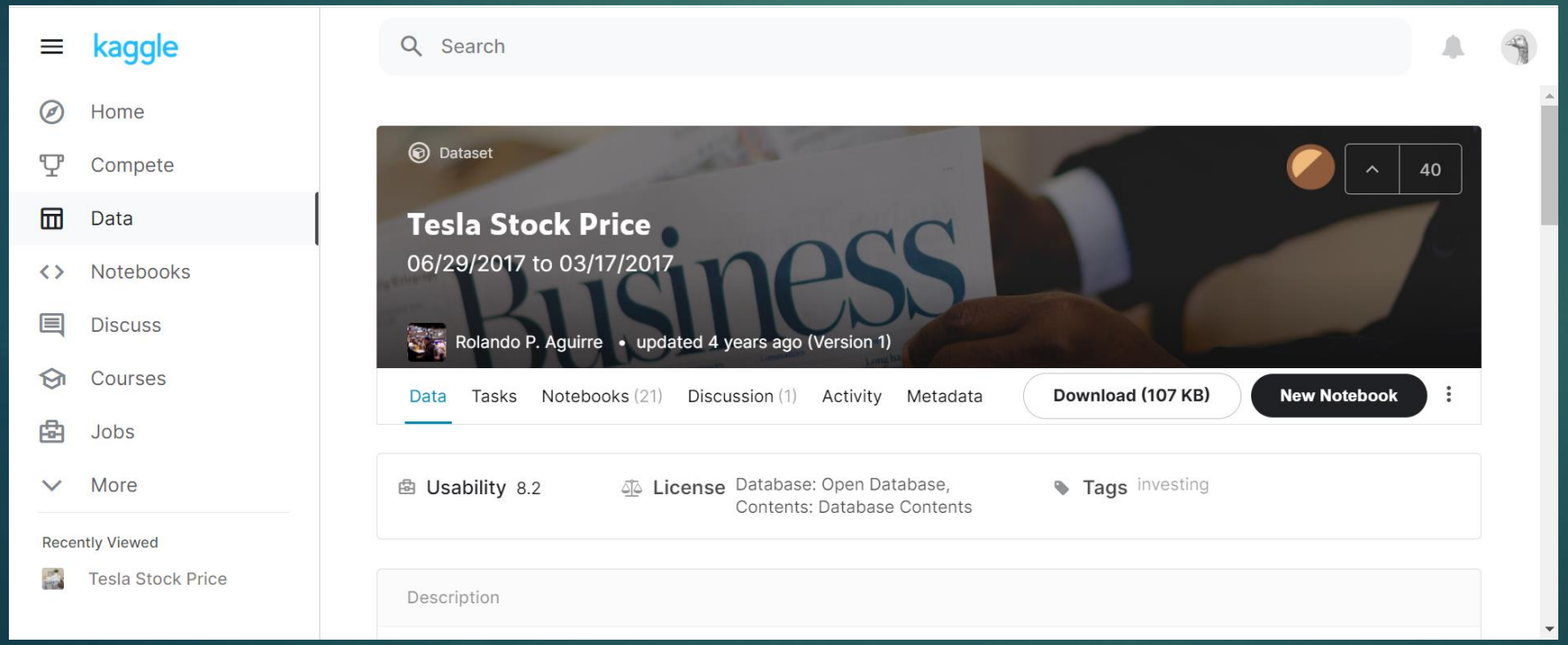
Assignment – Your own window Functions

BY RANDY LEON

1. Find a dataset that includes a time series (e.g. end of day stock prices for several instruments).
2. Use window functions to calculate year to date and six day moving averages.
3. Present your code in a three to five minute presentation.
4. You may work in a small team on this assignment.

STEP ONE: Find a dataset that includes a time series (e.g. end of day stock prices for several instruments).

I chose a dataset I found on Kaggle.com, the historical stock price for $TSLA (Tesla Stock, since it IPO'd up until 2017-03-17.



	Date date	Open numeric	High numeric	Low numeric	Close numeric	Volume (in Thousands) numeric	Adjusted Close numeric
1	2010-06-29	19	25	17.540001	23.889999	18766.3	23.889999
2	2010-06-30	25.790001	30.42	23.299999	23.83	17187.1	23.83
3	2010-07-01	25	25.92	20.27	21.959999	8218.8	21.959999
4	2010-07-02	23	23.1	18.709999	19.200001	5139.8	19.200001
5	2010-07-06	20	20	15.83	16.110001	6866.9	16.110001
6	2010-07-07	16.4	16.629999	14.98	15.8	6921.7	15.8
7	2010-07-08	16.139999	17.52	15.57	17.459999	7711.4	17.459999
8	2010-07-09	17.58	17.9	16.549999	17.4	4050.6	17.4
9	2010-07-12	17.950001	18.07	17	17.049999	2202.5	17.049999
10	2010-07-13	17.389999	18.639999	16.9	18.139999	2680.1	18.139999
11	2010-07-14	17.940001	20.15	17.76	19.84	4195.2	19.84
12	2010-07-15	19.940001	21.5	19	19.889999	3739.8	19.889999
13	2010-07-16	20.700001	21.299999	20.049999	20.639999	2621.3	20.639999
14	2010-07-19	21.370001	22.25	20.92	21.91	2486.5	21.91
15	2010-07-20	21.85	21.85	20.049999	20.299999	1825.3	20.299999
16	2010-07-21	20.66	20.9	19.5	20.219999	1252.5	20.219999

STEP TWO: Use window functions to calculate year to date and six day moving averages.

SIX DAY MOVING AVERAGE, CODE AND SCREENSHOT

Stocks/postgres@PostgreSQL 11

Query Editor | Query History

```
select * from public."StockA";

SELECT "Date", "Open", "High ", "Low", "Close", "Volume (in Thousands)", "Adjusted Close", AVG ("Adjusted Close") OVER
(ORDER BY "Date" ASC
 ROWS BETWEEN
 6 PRECEDING AND CURRENT ROW) as "6_Day_MVG_AVG"
 FROM public."StockA"
 ORDER BY "Date" ASC;
```

Data Output | Explain | Messages | Notifications

	Date date	Open numeric	High numeric	Low numeric	Close numeric	Volume (in Thousands) numeric	Adjusted Close numeric	6_Day_MVG_AVG numeric
1	2010-06...	19	25	17.540001	23.889999	18766.3	23.889999	23.8899990000000000
2	2010-06...	25.790001	30.42	23.299999	23.83	17187.1	23.83	23.8599995000000000
3	2010-07...	25	25.92	20.27	21.959999	8218.8	21.959999	23.2266660000000000
4	2010-07...	23	23.1	18.709999	19.200001	5139.8	19.200001	22.2199997500000000
5	2010-07...	20	20	15.83	16.110001	6866.9	16.110001	20.9980000000000000
6	2010-07...	16.4	16.629999	14.98	15.8	6921.7	15.8	20.1316666666666667
7	2010-07...	16.139999	17.52	15.57	17.459999	7711.4	17.459999	19.7499998571428571
8	2010-07...	17.58	17.9	16.549999	17.4	4050.6	17.4	18.8228571428571429
9	2010-07...	17.950001	18.07	17	17.049999	2202.5	17.049999	17.8542855714285714
10	2010-07...	17.389999	18.639999	16.9	18.139999	2680.1	18.139999	17.3085712857142857
11	2010-07...	17.940001	20.15	17.76	19.84	4195.2	19.84	17.3999997142857143
12	2010-07...	19.940001	21.5	19	19.889999	3739.8	19.889999	17.9399994285714286

I USED '6 PRECEDING AND CURRENT ROW'

THOUGH THE DAYS ARE NOT IN CHRONOLOGICAL ORDER (THESE DATES SKIP WEEKENDS) IT DOESN'T MATTER BECAUSE THE STOCK MARKET IS CLOSED ON THE WEEKEND.

YEAR TO DATE MOVING AVERAGE, CODE AND SCREENSHOTS

Stocks/postgres@PostgreSQL 11

Query Editor | Query History

```
select EXTRACT(YEAR FROM "Date"), ROUND("Adjusted Close",2),
    AVG ("Adjusted Close") OVER(
        PARTITION BY EXTRACT(YEAR FROM "Date")
        ORDER BY EXTRACT(YEAR FROM "Date"))
        AS Adjusted_Close_Yearly_AVG
FROM public."StockA";
```

I WAS ABLE TO CALCULATE THIS, HOWEVER, I TRIED HAVING IT SO ONLY DISTINCT YEARS CAME UP.

AS YOU WILL SEE IN SUBSEQUENT SCREENSHOTS, I WAS ABLE TO CALCULATE THE ADJUSTED CLOSING PRICE FOR EACH YEAR FOR $TSLA.

Data Output

	date_part double precision	round numeric	adjusted_close_yearly_avg numeric
1	2010	23.89	23.341846153846153
2	2010	23.83	23.341846153846153
3	2010	21.96	23.341846153846153
4	2010	19.20	23.341846153846153
5	2010	16.11	23.341846153846153
6	2010	15.80	23.341846153846153
7	2010	17.46	23.341846153846153
8	2010	17.40	23.341846153846153
9	2010	17.05	23.341846153846153
10	2010	18.14	23.341846153846153
11	2010	19.84	23.341846153846153
12	2010	19.89	23.341846153846153
13	2010	20.64	23.341846153846153
14	2010	21.91	23.341846153846153
15	2010	20.30	23.341846153846153
16	2010	20.22	23.341846153846153

YEAR TO DATE MOVING AVERAGE, CODE AND SCREENSHOTS CONT.

Data Output

	date_part double precision	round numeric	adjusted_close_yearly_avg numeric
126	2010	25.55	23.3418461538461538
127	2010	26.41	23.3418461538461538
128	2010	27.73	23.3418461538461538
129	2010	26.50	23.3418461538461538
130	2010	26.63	23.3418461538461538
131	2011	26.62	26.8047618611111111
132	2011	26.67	26.8047618611111111
133	2011	26.83	26.8047618611111111
134	2011	27.88	26.8047618611111111
135	2011	28.24	26.8047618611111111
136	2011	28.45	26.8047618611111111
137	2011	26.96	26.8047618611111111
138	2011	26.96	26.8047618611111111
139	2011	26.22	26.8047618611111111
140	2011	25.75	26.8047618611111111
141	2011	25.64	26.8047618611111111

Data Output

	date_part double precision	round numeric	adjusted_close_yearly_avg numeric
624	2012	34.59	31.1686000360000000
625	2012	34.61	31.1686000360000000
626	2012	34.43	31.1686000360000000
627	2012	34.00	31.1686000360000000
628	2012	34.28	31.1686000360000000
629	2012	33.59	31.1686000360000000
630	2012	33.69	31.1686000360000000
631	2012	33.22	31.1686000360000000
632	2012	33.87	31.1686000360000000
633	2013	35.36	104.4012297261904762
634	2013	34.77	104.4012297261904762
635	2013	34.40	104.4012297261904762
636	2013	34.34	104.4012297261904762
637	2013	33.68	104.4012297261904762
638	2013	33.64	104.4012297261904762
639	2013	33.53	104.4012297261904762