Keyboard Switches Self-Study Part 2:

By: Alexander Cai

Introduction:

Idea:

When Wilson Leung and I first did our self-collected data analysis back in May 2023, we expected to leave the project there. However, I thought the project could be further improved, and had lots of potential to be even greater than it currently was. This was when I decided to self-collect data on myself, testing a bunch of different mechanical keyboard switches, and seeing if any of them had an impact on my words per minute and accuracy. This idea continued to lead me to pick up the ideas of the project and expand it, adding new skills I have acquired from various statistics courses, and implementing those techniques into this project.

Goal:

Do keyboard switches impact your typing performance?

Overview:

The project will be conducted on a singular keyboard, the Blade60. This will control variability in our scores. We attempt to achieve our goal by typing on a set of switches every day and recording those results into a Google Sheet. There will be multiple different Google Sheets that will hold different types of data (Descriptions, Summaries, Raw Data, Date Averages). We will create a testing procedure for testing each switch, and once all of the switches have been tested, we can move on to exploring/analyzing the data.

The project will consist of 3 different parts:

1) Data Collection
2) Database exploration (SQL)
3) Data Analysis (Python)

Data Collection:

Testing Procedure:

1) First, I will do 50 typing tests with my daily driver switch (Oil Kings) to determine my current WPM.
2) Decide the order of switches through a randomizer (Python)
3) Daily Process:
    a) Every day, we will record 10 typing trials on the baseline switch (Oil Kings)
    b) Change to the current switches that need to be tested that day
    c) Record a video of me talking about day-by-day updates, and a switch typing test for reference.
    d) Take a 2-minute break after every 10 trials.
    e) Record statistics like accuracy, WPM, date, and switch ID.
    f) Record the pitch of switches by placing the phone near my keyboard and performing 10 typing tests, marking the average decibels.
    g) Repeat steps d&e until all 50 iterations have been done.
    h) Record average WPM for baseline switch, WPM/accuracy for switch of the day. Rate the switch 1-10. Finally, subtract WPM for the switch of the day from the baseline switch
4) Repeat the process until all switches have been completed.

Updated Testing Procedure:

1. First, I will do 50 typing tests with my daily driver switch (Oil Kings) to determine my current WPM.
2. Decide the order of switches through a randomizer (Python), while also running new iterations based on added switches/ switches that need retesting.
3. Record the pitch of switches by pressing each switch with a keycap on it close to my microphone. Perform 30 presses, and record the average.
4. Daily Process:
    a. Every day, we will record 10 typing trials on the baseline switch (Oil Kings)
    b. Change to the current switches that need to be tested that day
    c. Record a video of me talking about day-by-day updates, and a switch typing test for reference.
    d. Run trials for the daily switch, taking a 2-minute break in between every 10 trials.
    e. Record statistics like accuracy, WPM, date, and switch ID.
    f. Repeat steps d&e until all 50 iterations have been done.
    g. Record average WPM for baseline switch, WPM/accuracy for switch of the day. Subtract WPM for the switch of the day from the baseline switch. Rate the switch on a scale of 1-10. Finally, write 2 words you would use to describe the daily switch.
    h. Mark down the daily average WPM and accuracy across baseline and daily switch.
5. Repeat the process until all switches have been completed.

Deciding the Order:

1) We will obtain our order by using Python:
    a) Create a list of switches
    b) Shuffle the switches using the random library
    c) Print the order.

Order:

1. Holy Panda
2. Kiwi
3. Xinhai
4. Type R
5. Tealios V2
6. Higanabana
7. Curry

New Order:

- Unfortunately, I lent the Kiwi Switches to a friend and do not know when I can get them back. For the sake of the experiment, we will swap the Kiwis for the curry switches.

- I was also able to order/borrow some switches from friends and others, making it so that I could add more switches to my list, but as a result, prioritize those switches first, because I want to return the switches as soon as possible.

1) Holy Panda
2) Curry
3) Banana Milk
4) Kiwi
5) Tealios V2
6) Higanbana
7) Brown
8) Type R
9) Xinhai

Final Order:

- I was able to purchase a few switches and borrow some of my friends'. For simplicity and to avoid ruining the current order, I decided to test the following switches after all the ones in my current line-up. The order was made through a random generator.

1. Holy Panda
2. Curry
3. Banana Milk
4. Kiwi
5. Tealios V2
6. Higanbana
7. Brown
8. Type R
9. Xinhai
10. Tangerine
11. Melodic
12. Sonja

Retesting:

I noticed a few switches performed much better or worse than others (Curry, Holy Panda, Tangerine). Because these switches performed so well/poorly, I wanted to check these statistics by rerunning tests for each of the following switches(in the following order):

1. Tangerine
2. Holy Panda
3. Curry

SQL Analysis:

Idea:

The goal of using SQL is to perform **exploratory data analysis** and **create a database/dataframe** with the following data accumulated. SQL is great at combining and sorting data in various ways. With these capabilities, SQL makes a great platform to perform basic summary statistics analysis through different sorts, specifically switch-by-switch comparisons. Another problem was the separation of data. Because I stored the data across different Google Sheets, I needed to find a way to eventually combine all of the data into one spreadsheet in which I can call when I perform my Data Analysis in Python. SQL is the perfect resource for this, because it allows easy and quick combination of different CSV files into one, given a set of commands.

Exploratory Database Analysis:

1) Sorting Switches by WPM:

- Original Summary Dataset(Before retesting):

id	switch	mean_wpm	benchmark_mean	adj_mean_wpm	accuracy	loudness(db)	enjoyment(/10)	notes	date
5	Oil King	76.9	0	0	0.89	62.7	8	Smooth, weighty	6/13/2025
12	Banana Milk	77.3	77.8	-0.5	0.9	69.5	8	satisfying, snappy	6/16/2025
3	Higanbana	77.8	77.2	0.6	0.93	76.6	7	loud, fun	6/20/2025
7	Holy Pandas	78.2	71.3	6.9	0.9	74.3	9	weighty, fun	6/14/2025
8	Tealios V2	79.5	79.6	-0.1	0.9	62.8	6	Buttery, low-resistance	6/18/2025
13	Tangerine	80.2	86.1	-5.9	0.91	68	6	Smooth, low-resistance	6/24/2025
1	EF Curry	80.6	73.6	7	0.91	62.6	6	smooth	6/15/2025
11	Brown	82.4	80.3	2.1	0.92	69.3	7	Scratchy, light	6/21/2025
9	Sonja	83.9	81	2.9	0.92	70.9	9	Scratchy, fun	6/26/2025
6	Kiwi	84.2	81.9	2.3	0.91	69.9	8	Scratchy, fun	6/17/2025
4	Type R	84.8	82.1	2.7	0.92	70.9	9	snappy, fun	6/22/2025
10	Melodic	86.7	83.3	3.4	0.93	70	8	satisfying, crunchy	6/25/2025
2	Xinhai	86.8	84.8	2	0.92	71.2	8	satisfying, snappy	6/22/2025

- Updated Summary Dataset:

id	switch	mean_wpm	benchmark_mean	adj_mean_wpm	accuracy	loudness (DB)	enjoyment(/10)	notes	date
5	Oil King	76.9	0	0	0.89	62.7	8	Smooth, weighty	6/13/2025
12	Banana Milk	77.3	77.8	-0.5	0.9	69.5	8	satisfying, snappy	6/16/2025
3	Higanbana	77.8	77.2	0.6	0.93	76.6	7	loud, fun	6/20/2025
8	Tealios V2	79.5	79.6	-0.1	0.9	62.8	6	Buttery, low-resistance	6/18/2025
13	Tangerine	81	82.1	-1.1	0.91	68	6	Smooth, low-resistance	6/27/2025
11	Brown	82.4	80.3	2.1	0.92	69.3	7	Scratchy, light	6/21/2025
1	EF Curry	83.4	89.7	-6.3	0.92	62.6	6	smooth	6/29/2025
9	Sonja	83.9	81	2.9	0.92	70.9	9	Scratchy, fun	6/26/2025
7	Holy Pandas	84.1	79.9	4.2	0.91	74.3	9	weighty, fun	6/28/2025
6	Kiwi	84.2	81.9	2.3	0.91	69.9	8	Scratchy, fun	6/17/2025
4	Type R	84.8	82.1	2.7	0.92	70.9	9	snappy, fun	6/22/2025
10	Melodic	86.7	83.3	3.4	0.93	70	8	satisfying, crunchy	6/25/2025
2	Xinhai	86.8	84.8	2	0.92	71.2	8	satisfying, snappy	6/22/2025

Observations:

1) The mean WPM ranges from 76.9 to 86.8. This is almost a 10 WPM difference across testing dates.

2) Accuracy may be correlated to mean WPM, as when I tend to type faster, there seems to be a positive increase in accuracy.

3) When comparing the original dataset to the updated one, Holy Pandas goes from being in the lower half of the average WPM to the upper half.

2) Sorting Switches by Adjusted Mean WPM:

- Original Summary Dataset:

	id	switch	mean_wpm	benchmark_mean	adj_mean_wpm	accuracy	loudness(db)	enjoyment(/10)	notes	date
1	13	Tangerine	80.2	86.1	-5.9	0.91	68	6	Smooth, low-resistance	6/24/2025
2	12	Banana Milk	77.3	77.8	-0.5	0.9	69.5	8	satisfying, snappy	6/16/2025
3	8	Tealios V2	79.5	79.6	-0.1	0.9	62.8	6	Buttery, low-resistance	6/18/2025
4	5	Oil King	76.9	0	0	0.89	62.7	8	Smooth, weighty	6/13/2025
5	3	Higanbana	77.8	77.2	0.6	0.93	76.6	7	loud, fun	6/20/2025
6	2	Xinhai	86.8	84.8	2	0.92	71.2	8	satisfying, snappy	6/22/2025
7	11	Brown	82.4	80.3	2.1	0.92	69.3	7	Scratchy, light	6/21/2025
8	6	Kiwi	84.2	81.9	2.3	0.91	69.9	8	Scratchy, fun	6/17/2025
9	4	Type R	84.8	82.1	2.7	0.92	70.9	9	snappy, fun	6/22/2025
10	9	Sonja	83.9	81	2.9	0.92	70.9	9	Scratchy, fun	6/26/2025
11	10	Melodic	86.7	83.3	3.4	0.93	70	8	satisfying, crunchy	6/25/2025
12	7	Holy Pandas	78.2	71.3	6.9	0.9	74.3	9	weighty, fun	6/14/2025
13	1	EF Curry	80.6	73.6	7	0.91	62.6	6	smooth	6/15/2025

- **Updated Summary Dataset:**

id	switch	mean_wpm	benchmark_mean	adj_mean_wpm	accuracy	loudness (DB)	enjoyment(/10)	notes	date
1	EF Curry	83.4	89.7	-6.3	0.92	62.6	6	smooth	6/29/2025
13	Tangerine	81	82.1	-1.1	0.91	68	6	Smooth, low-resistance	6/27/2025
12	Banana Milk	77.3	77.8	-0.5	0.9	69.5	8	satisfying, snappy	6/16/2025
8	Tealios V2	79.5	79.6	-0.1	0.9	62.8	6	Buttery, low-resistance	6/18/2025
5	Oil King	76.9	0	0	0.89	62.7	8	Smooth, weighty	6/13/2025
3	Higanbana	77.8	77.2	0.6	0.93	76.6	7	loud, fun	6/20/2025
2	Xinhai	86.8	84.8	2	0.92	71.2	8	satisfying, snappy	6/22/2025
11	Brown	82.4	80.3	2.1	0.92	69.3	7	Scratchy, light	6/21/2025
6	Kiwi	84.2	81.9	2.3	0.91	69.9	8	Scratchy, fun	6/17/2025
4	Type R	84.8	82.1	2.7	0.92	70.9	9	snappy, fun	6/22/2025
9	Sonja	83.9	81	2.9	0.92	70.9	9	Scratchy, fun	6/26/2025
10	Melodic	86.7	83.3	3.4	0.93	70	8	satisfying, crunchy	6/25/2025
7	Holy Pandas	84.1	79.9	4.2	0.91	74.3	9	weighty, fun	6/28/2025

Observations:

1) On average, we perform better with the “daily” switch in comparison to the benchmark (Oil Kings).

2) The EF Curry switch may be an outlier, as its adj_mean(adjusted means) of -6.3 is suspiciously low. However, this is likely due to its rather high benchmark mean for the day(06/29/2025).

3) Comparing both datasets adj_means, the switches we retested had significant changes. This shows that retesting our data had a significant impact on the way our data was ordered.

3) Sorting by Date:

- **Updated Date Averages Dataset:**

	A-Z date	123 avg_wpm	123 avg_accuracy
1	6/13/2025	76.9	0.89
2	6/14/2025	77	0.9
3	6/15/2025	79.4	0.91
4	6/16/2025	77.4	0.9
5	6/17/2025	83.8	0.91
6	6/18/2025	79.2	0.9
7	6/20/2025	77.7	0.93
8	6/21/2025	82	0.92
9	6/22/2025	84.3	0.93
10	6/23/2025	86.5	0.93
11	6/24/2025	81.2	0.91
12	6/25/2025	86.1	0.93
13	6/26/2025	83.4	0.92
14	6/27/2025	81.2	0.91
15	6/28/2025	83.7	0.93
16	6/29/2025	84.4	0.92

Observations:

1) With each passing date, there is a slight increase in WPM, going from the high 70s to mid-high 80s.

2) The same can be attributed to accuracy, as I started at 89, but then never achieved anything lower than 91 after the 16th.

---

4) Comparing Tactility:

- **Combining updated summary and updated descriptions:**

A-Z tactility	123 mean_wpm	123 mean_accuracy
No	80.8250017166	0.9112500027
Medium	84.1999969482	0.9100000262
Large	85.1999994914	0.9200000167
Low	82.4000015259	0.9200000167

Observations:

1) I performed better on tactile switches, just by observing the averages of both wpm and accuracy.

2) This suggests that I have a slight preference for tactile switches.

---

5) How many tests did I perform equal to or less than 75 WPM?:

- Before retesting dates (Using Raw Dataset)

123 count
187

- After retesting (Updated Raw Dataset)

123 count
27

Observations:

1) Altogether, I achieved 214 typing tests where I typed 75 WPM or less.

2) On average, I expect to achieve an average of about 14 (187/13) tests in a day with <=75 WPM, but only 9 (27/3) with retesting on days. This could likely be because:

a) I have familiarized myself with each switch; therefore, my performances have gotten better as a result.
b) I have slowly improved with typing since the start of the collection process
c) A mix of both (likely).

---

6) How many tests did I perform equal to or more than 95 WPM?:

- Before retesting dates (Using Raw Dataset)

123 count
33

- After retesting (Updated Raw Dataset)

123 count
6

Observations:

1) Altogether, we achieved 39 typing tests where I typed 95 WPM or more.

2) On average, I expect to achieve an average of about 3 (33/13) tests in a day with >=95 WPM, but 2 (6/3) with retesting on days.

a) These two are fairly close in trials, so we could assume that while I was getting better at typing (not achieving as many <=75 WPM), we were still achieving roughly the same amount of "very high" typing scores(>=95).

---

7) How many tests did I perform <=75 and >=95(Counted per day)?:

- **<=75 (Using Updated Raw Data):**

	A-Z date	123 count
1	6/13/2025	20
2	6/14/2025	27
3	6/15/2025	18
4	6/16/2025	26
5	6/17/2025	8
6	6/18/2025	20
7	6/20/2025	24
8	6/21/2025	11
9	6/22/2025	4
10	6/23/2025	3
11	6/24/2025	14
12	6/25/2025	3
13	6/26/2025	9
14	6/27/2025	13
15	6/28/2025	8
16	6/29/2025	6

- **>=95 (Using Updated Raw Data):**

A-Z date	123 count
6/13/2025	1
6/14/2025	2
6/15/2025	1
6/17/2025	3
6/18/2025	2
6/21/2025	3
6/22/2025	3
6/23/2025	4
6/24/2025	1
6/25/2025	9
6/26/2025	4
6/29/2025	6

Observations:

1) As time continued throughout the project, we can see fewer >=75 scores appear.

2) All of the >=95 scores achieved during the retesting days were performed on the last day. This is likely during the benchmark testing (recall the rather large daily benchmark recording of 89.7).

3) 4 dates are missing on the >=95 table: 6/16, 6/20, 6/27, 2/28.

4) There is a rather larger number of scores >=95 on 9/25. This was the date we were testing the Melodic, the only clicky switch on our list.

---

8) Of the "fun" switches out there, what were the adj. mean WPM, loudness, enjoyment, tactility, and date statistics?

- Using a TSV Vector and combining different tables:

A-Z date	A-Z switch	123 date_wpm	123 date_accuracy	123 adj_mean_wpm	A-Z sound	123 enjoyment
6/17/2025	Kiwi	83.8	0.91	2.3	High	8
6/20/2025	Higanbana	77.7	0.93	0.6	High	7
6/22/2025	Type R	84.3	0.93	2.7	High	9
6/26/2025	Sonja	83.4	0.92	2.9	Low	9
6/28/2025	Holy Pandas	83.7	0.93	4.2	High	9

Observations:

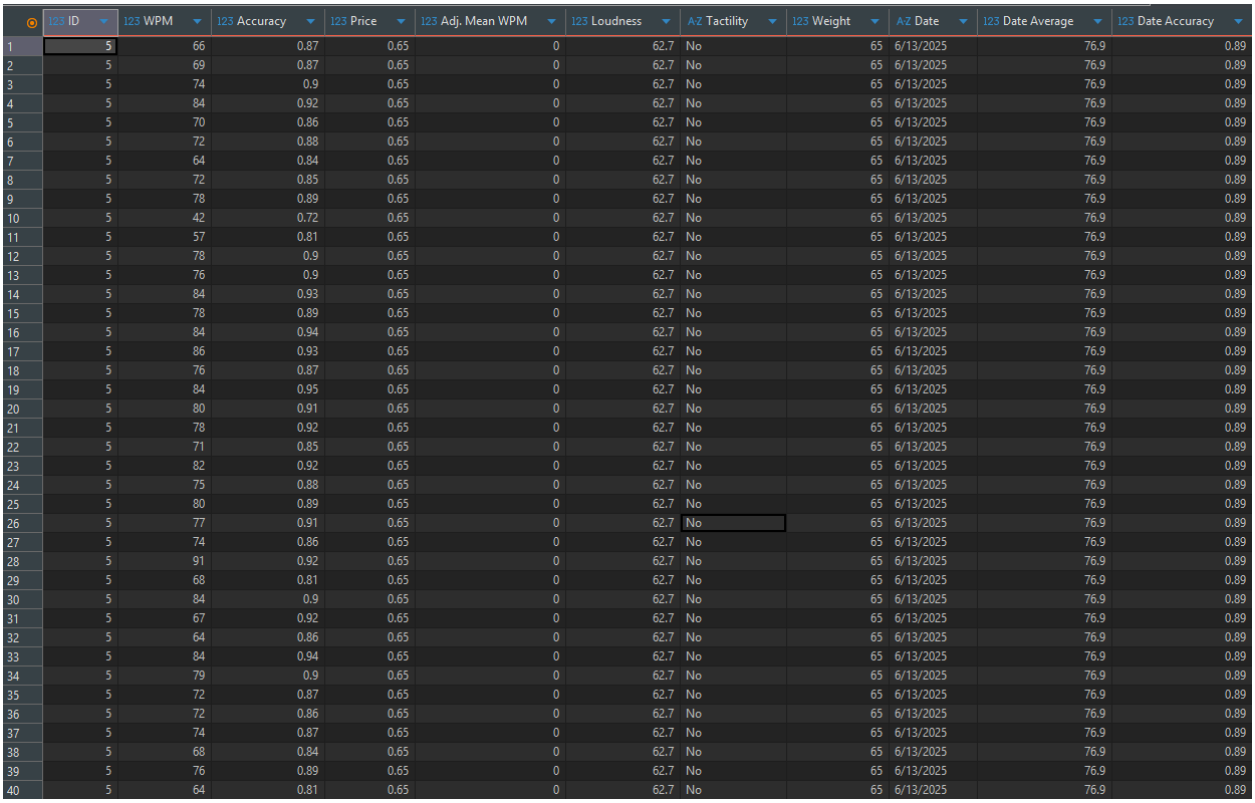
1) Of the switches I found 'fun', I see that I did better with the following switch compared to the benchmark.

2) I enjoyed typing on the switches that were 'fun', with all achieving at least a 7/10, most being a 9/10.

3) A majority of the switches I found fun were high-pitched switches(with the HMX Sonja being the one exception).

Creating the Ideal Database:

- Screenshot of the first 40 entries:



	ID	WPM	Accuracy	Price	Adj. Mean WPM	Loudness	Tactility	Weight	Date	Date Average	Date Accuracy
1	5	66	0.87	0.65	0	62.7	No	65	6/13/2025	76.9	0.89
2	5	69	0.87	0.65	0	62.7	No	65	6/13/2025	76.9	0.89
3	5	74	0.9	0.65	0	62.7	No	65	6/13/2025	76.9	0.89
4	5	84	0.92	0.65	0	62.7	No	65	6/13/2025	76.9	0.89
5	5	70	0.86	0.65	0	62.7	No	65	6/13/2025	76.9	0.89
6	5	72	0.88	0.65	0	62.7	No	65	6/13/2025	76.9	0.89
7	5	64	0.84	0.65	0	62.7	No	65	6/13/2025	76.9	0.89
8	5	72	0.85	0.65	0	62.7	No	65	6/13/2025	76.9	0.89
9	5	78	0.89	0.65	0	62.7	No	65	6/13/2025	76.9	0.89
10	5	42	0.72	0.65	0	62.7	No	65	6/13/2025	76.9	0.89
11	5	57	0.81	0.65	0	62.7	No	65	6/13/2025	76.9	0.89
12	5	78	0.9	0.65	0	62.7	No	65	6/13/2025	76.9	0.89
13	5	76	0.9	0.65	0	62.7	No	65	6/13/2025	76.9	0.89
14	5	84	0.93	0.65	0	62.7	No	65	6/13/2025	76.9	0.89
15	5	78	0.89	0.65	0	62.7	No	65	6/13/2025	76.9	0.89
16	5	84	0.94	0.65	0	62.7	No	65	6/13/2025	76.9	0.89
17	5	86	0.93	0.65	0	62.7	No	65	6/13/2025	76.9	0.89
18	5	76	0.87	0.65	0	62.7	No	65	6/13/2025	76.9	0.89
19	5	84	0.95	0.65	0	62.7	No	65	6/13/2025	76.9	0.89
20	5	80	0.91	0.65	0	62.7	No	65	6/13/2025	76.9	0.89
21	5	78	0.92	0.65	0	62.7	No	65	6/13/2025	76.9	0.89
22	5	71	0.85	0.65	0	62.7	No	65	6/13/2025	76.9	0.89
23	5	82	0.92	0.65	0	62.7	No	65	6/13/2025	76.9	0.89
24	5	75	0.88	0.65	0	62.7	No	65	6/13/2025	76.9	0.89
25	5	80	0.89	0.65	0	62.7	No	65	6/13/2025	76.9	0.89
26	5	77	0.91	0.65	0	62.7	No	65	6/13/2025	76.9	0.89
27	5	74	0.86	0.65	0	62.7	No	65	6/13/2025	76.9	0.89
28	5	91	0.92	0.65	0	62.7	No	65	6/13/2025	76.9	0.89
29	5	68	0.81	0.65	0	62.7	No	65	6/13/2025	76.9	0.89
30	5	84	0.9	0.65	0	62.7	No	65	6/13/2025	76.9	0.89
31	5	67	0.92	0.65	0	62.7	No	65	6/13/2025	76.9	0.89
32	5	64	0.86	0.65	0	62.7	No	65	6/13/2025	76.9	0.89
33	5	84	0.94	0.65	0	62.7	No	65	6/13/2025	76.9	0.89
34	5	79	0.9	0.65	0	62.7	No	65	6/13/2025	76.9	0.89
35	5	72	0.87	0.65	0	62.7	No	65	6/13/2025	76.9	0.89
36	5	72	0.86	0.65	0	62.7	No	65	6/13/2025	76.9	0.89
37	5	74	0.87	0.65	0	62.7	No	65	6/13/2025	76.9	0.89
38	5	68	0.84	0.65	0	62.7	No	65	6/13/2025	76.9	0.89
39	5	76	0.89	0.65	0	62.7	No	65	6/13/2025	76.9	0.89
40	5	64	0.81	0.65	0	62.7	No	65	6/13/2025	76.9	0.89

Reasoning:

- **ID:** Identifier for the switch.
- **WPM:** Our main response variable.
- **Accuracy:** Another possible response variable that can help be a good predictor, given it has so many trials.
- **Adj_Mean_WPM:** Can be useful if we want to compare the performances between switches.
- **Price/Tactility/Weight:** All predictors we can use to see if they may influence WPM/Accuracy, and be used to build a model.
- **Date/Date Average/Date Accuracy:** Will be useful if we are looking at a time series.

Python Analysis:

Idea/Questions of Interest:

While SQL is great for database analysis, it lacks in visualizations and different statistical analyses. That is where Python shines. Using various packages like Pandas, Matplotlib, Seaborn, and more, Python becomes a powerful software that allows us to explore a variety of graphs, tests, and other relevant statistics. After using SQL to construct our ideal database, we will try to answer the following:

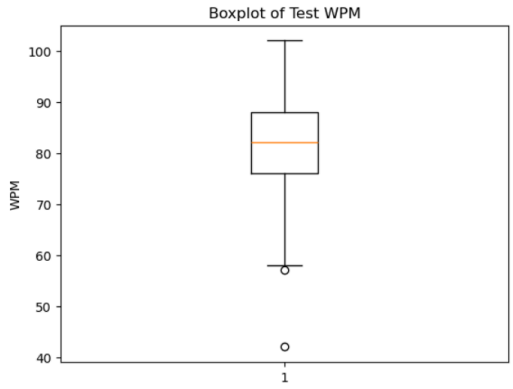
1) What do the following summary statistics of each predictor tell us?
2) What does the distribution of each switch look like?
3) Is there a correlation between weight and WPM?
4) Is there a correlation between price and WPM?
5) Construct a 95% confidence interval for WPM.
6) Construct a 95% confidence interval for Adjusted Mean WPM.
7) Is there a relationship between dates and both average WPM/accuracy?
8) Can we predict the WPM of a switch based on its ID?
9) Can we predict the WPM of a switch based on loudness, enjoyment, weight, sound, and tactility?
10) Given all of the predictors (except ID), can we predict the WPM of a test?

1) What do the following summary statistics of each predictor tell us?

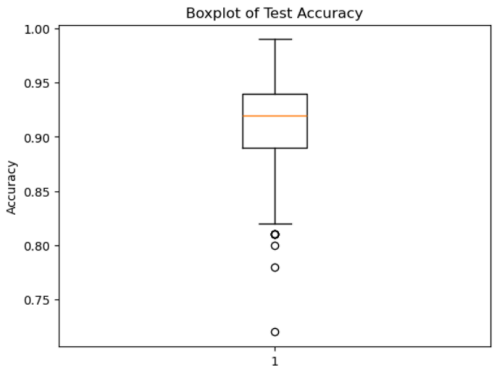
- Summary statistics of predictors:

	ID	WPM	Accuracy	Price	Adj. Mean WPM	Loudness	Weight	Date	Date Average	Date Accuracy	days_since_start	Tactility
count	950.000000	950.000000	950.000000	950.000000	950.000000	950.000000	950.000000	950	950.000000	950.000000	950.000000	950.000000
mean	6.684211	81.541053	0.913242	0.558947	0.473684	67.984211	62.421053	2025-06-21 05:03:09.473684224	81.561053	0.915263	8.210526	0.315789
min	1.000000	42.000000	0.720000	0.200000	-6.300000	62.600000	50.000000	2025-06-13 00:00:00	76.900000	0.890000	0.000000	0.000000
25%	4.000000	76.000000	0.890000	0.350000	-0.500000	62.700000	60.000000	2025-06-17 00:00:00	79.200000	0.910000	4.000000	0.000000
50%	6.000000	82.000000	0.920000	0.650000	0.000000	69.300000	65.000000	2025-06-22 00:00:00	82.000000	0.920000	9.000000	0.000000
75%	10.000000	88.000000	0.940000	0.700000	2.700000	70.900000	67.000000	2025-06-26 00:00:00	84.300000	0.930000	13.000000	1.000000
max	13.000000	102.000000	0.990000	1.200000	4.200000	76.600000	68.000000	2025-06-29 00:00:00	86.500000	0.930000	16.000000	1.000000
std	3.730271	8.149544	0.035539	0.231021	2.852142	4.531038	5.521448	NaN	3.134037	0.012559	4.982360	0.465074

- Boxplot of test WPM:



- Boxplot of test Accuracy:



Observations:

1) WPM: If I took the average of all my typing tests, I would type at an average of 81.5 WPM. This is fairly accurate to what my overall WPM is in Monkeytype (81 WPM). The distribution also seems to be normal, outside of 2 outliers.

2) Accuracy: If I took the average of all my typing tests, I would have a typing accuracy of 91.3%. Like WPM, we are normally distributed, outside of the 4 outliers present. There are likely more outliers for accuracy than WPM because we had a tighter spread for accuracy than WPM, meaning there is less variability.

3) Loudness: Of the 13 switches we tested, the average loudness (measured in dB) was about 68 DB. All things considered, this is fairly loud. However, this is likely due to the method I record the sound of the switches. We also have a skewed left distribution, telling me that we have switches that hover around 68+ more than <68.

4) Price: The average cost of one switch in the experiment is about 56 cents, which is fairly expensive in today's market. However, this is likely due to the majority of our switches being "older".

5) Weight: The average weight of all of the switches I tried was about 62.4 grams. I believe this is about average to above-average weight for switches today.

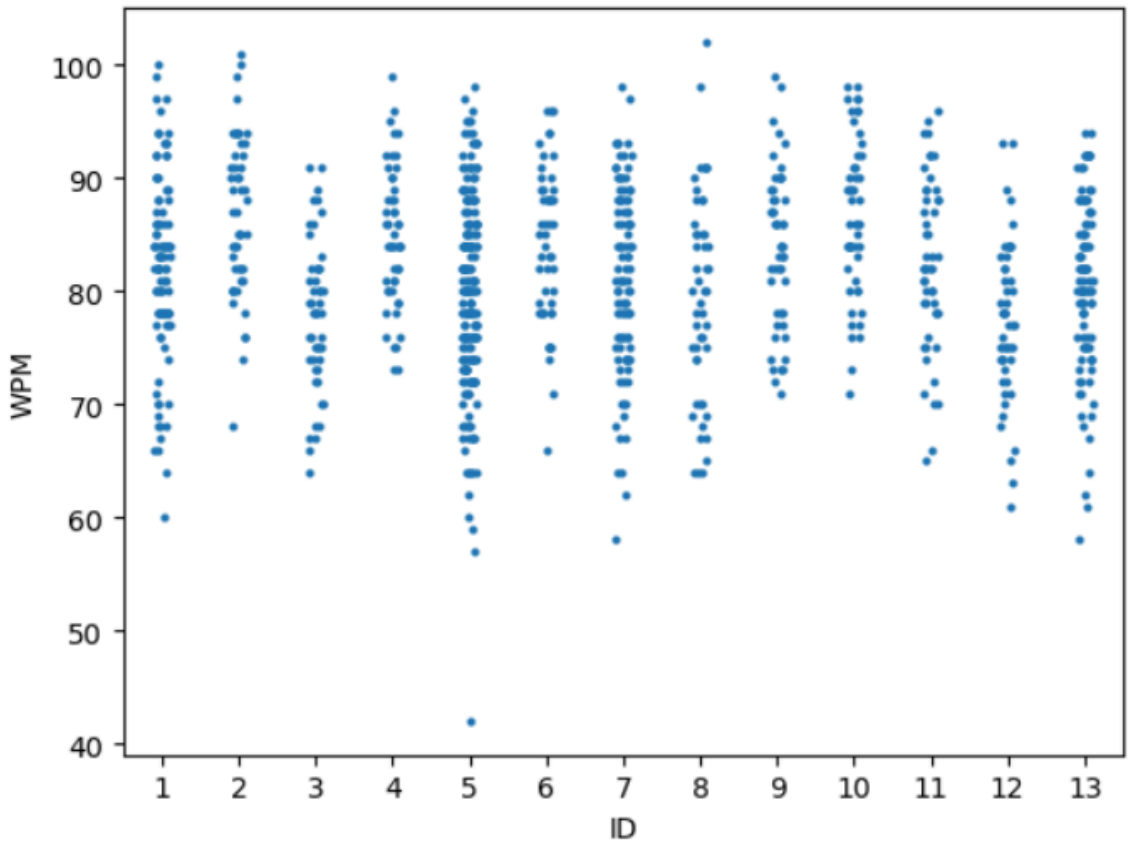
6) Date Average: When taking the average of all 16 testing days, my mean Daily WPM is 81.6. This is very close to our Average WPM.

7) Date Accuracy: My mean Daily accuracy is 91.5%, which is slightly higher than my Average Accuracy.

---

2) What does the distribution of each switch look like?

- Distribution of each switch:



Legend for ID:

1) Curry**, 2)** Xinhai**, 3)** Higanbana**, 4)** Type R**, 5)** Oil King**, 6)** Kiwi**, 7)** Holy Pandas**, 8)** Tealios V2**, 9)** Sonja**, 10)** Melodic**, 11)** Brown**, 12)** Banana Milk**, 13)** Tangerine

Observations:

1) Most of our tests were between mid-70s to low-90s.

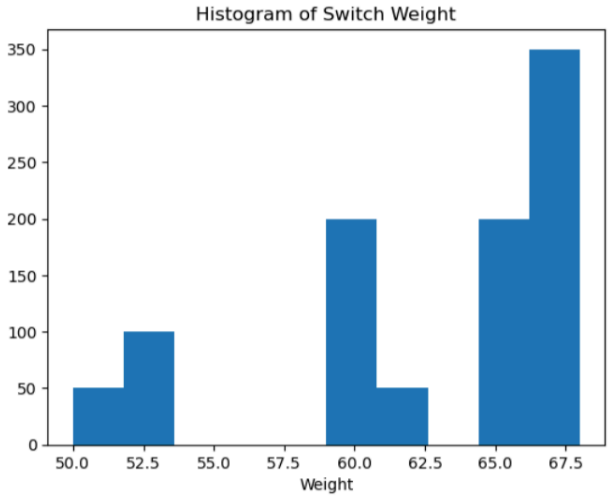
2) Some outliers from the Tealios V2 (ID 8) and the Oil King(ID 5) have an apparent outlier around ~40. The Tealios is an interesting case, as all of the other typing tests hovered around 65-90.

3) The Oil king has the widest variance among all of the switches, showing a large range of values ranging from ~40s to ~100.

3) Is there a correlation between weight and WPM?

Checking Pearson's Coefficient Conditions:

- As we were trying to pass the conditions for Pearson's Coefficient(passed Linearity and Independence), I noticed that I would severely fail the Nomality clause due to discrete data.



As a result, we will instead be running Spearman's coefficient, since it's better when working with ordinal/discrete data. We pass all other conditions needed for Spearman's.

Results:

Running Spearman's Coefficient

```
rho, p = spearmanr(x, y)
print(f"rho ={rho}")
print(f"p ={p}")
```

```
rho =-0.12002350544758929
p =0.00020897377317303676
```

Observations:

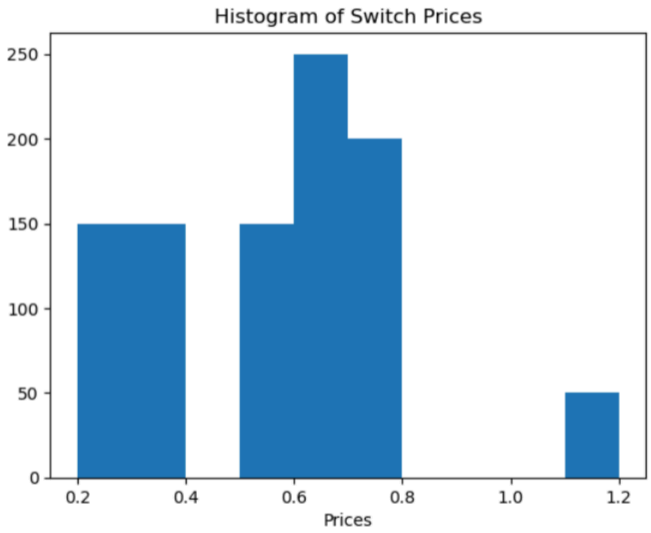
1) There is a weak negative monotonic relationship between Weight and WPM(ρ = -0.12).
2) Our small p-value (close to 0) indicates that there is a statistically significant monotonic relationship between Weight and WPM.

---

4) Is there a correlation between price and WPM?

Checking Pearson's Coefficient Conditions:

- Much like with the correlation between weight and WPM, unfortunately, we are unable to pass the Normality clause due to the non-normal distribution of data:



Once again, we run Spearman's coefficient. Again, we pass all other conditions needed.

Results:

Running Spearman's Coefficient

```
rho, p = spearmanr(x, y)
print(f"rho ={rho}")
print(f"p ={p}")
```

```
rho =-0.09838049984067643
p =0.0024000569413966982
```

Observations:

1) There is a weak negative monotonic relationship between Weight and WPM(ρ = -0.10).
2) Our small p-value (close to 0) indicates that there is a statistically significant monotonic relationship between Weight and WPM.

---

5) Construct a 95% confidence interval for WPM.

Checking Confidence Interval Conditions:

- We pass all the conditions needed to run a Confidence Interval for WPM, including Random Sample, Independence, and Normality.

Results:

Making the Confidence Interval:

```
int_data = kb_df['WPM']
stats.t.interval(confidence = 0.95, df = len(int_data)-1, loc = np.mean(int_data), scale=stats.sem(int_data))
```

```
(np.float64(81.02216437454538), np.float64(82.05994088861252))
```

Observation:

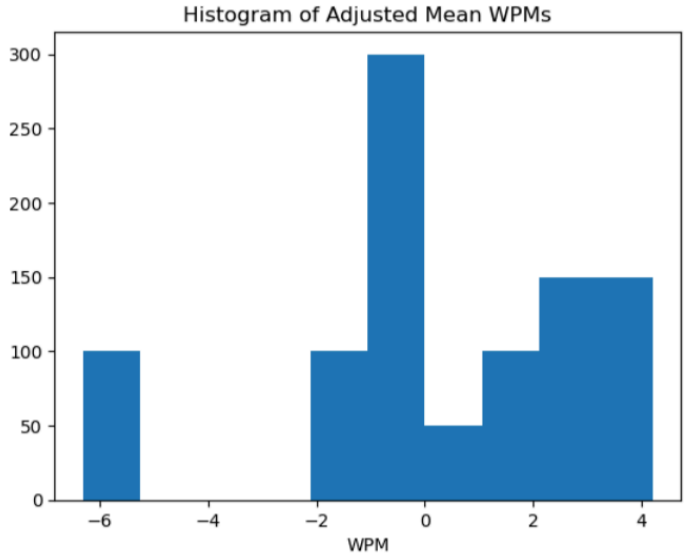
1) We are 95% confident that the confidence interval (81.022, 82.06) would contain the mean Words Per Minute when performing a MonkeyType test.

---

6) Construct a 95% confidence interval for adjusted mean WPM.

Checking Confidence Interval Conditions:

- Unfortunately, we would fail the Normality clause yet again for similar reasons. However, let's make a hypothetical confidence interval to see what one would look like.



Hypothetical Result

Making the Confidence Interval:

```
int_data = kb_df['Adj. Mean WPM']
stats.t.interval(confidence = 0.95, df = len(int_data)-1, loc = np.mean(int_data), scale=stats.sem(int_data))
```

```
(np.float64(0.2920859701090867), np.float64(0.655282450943545))
```

Observation:

1) We would be 95% confident that the confidence interval (0.29, 0.66) would contain the mean adjusted Words Per Minute(daily switch mean - daily benchmark mean), given that our data was normally distributed.

---

7) Is there a linear relationship between dates and average WPM/ accuracy?

Linear Regression Conditions:

- We pass all conditions needed to run linear regression, including linearity, homoscedasticity, normality, independence, and checking for outliers.

Results:

WPM Model:

```
WPM_model.summary()
```

OLS Regression Results

Dep. Variable:	WPM	**R-squared:**	0.070
Model:	OLS	**Adj. R-squared:**	0.069
Method:	Least Squares	**F-statistic:**	71.79
Date:	Wed, 20 Aug 2025	**Prob (F-statistic):**	9.05e-17
Time:	14:41:02	**Log-Likelihood:**	-3305.9
No. Observations:	950	**AIC:**	6616.
Df Residuals:	948	**BIC:**	6625.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	77.9778	0.492	158.540	0.000	77.013	78.943
days_since_start	0.4340	0.051	8.473	0.000	0.333	0.535

Omnibus:	8.076	**Durbin-Watson:**	1.753
Prob(Omnibus):	0.018	**Jarque-Bera (JB):**	7.992
Skew:	-0.214	**Prob(JB):**	0.0184
Kurtosis:	3.136	**Cond. No.**	18.7

Accuracy Model:

```
acc_model.summary()
```

OLS Regression Results

Dep. Variable:	Accuracy	**R-squared:**	0.049
Model:	OLS	**Adj. R-squared:**	0.048
Method:	Least Squares	**F-statistic:**	49.05
Date:	Wed, 20 Aug 2025	**Prob (F-statistic):**	4.72e-12
Time:	14:41:02	**Log-Likelihood:**	1846.7
No. Observations:	950	**AIC:**	-3689.
Df Residuals:	948	**BIC:**	-3680.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.9003	0.002	415.013	0.000	0.896	0.905
days_since_start	0.0016	0.000	7.004	0.000	0.001	0.002

Omnibus:	67.135	**Durbin-Watson:**	1.793
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	83.446
Skew:	-0.634	**Prob(JB):**	7.58e-19
Kurtosis:	3.706	**Cond. No.**	18.7

Finding r:

```
WPM_r = 0.070**0.5
ACC_r = 0.049**0.5

print(f'WPM model r = {WPM_r}')
print(f'ACC model r = {ACC_r}')
```

```
WPM model r = 0.2645751311064591
ACC model r = 0.22135943621178655
```

Observations:

1) Every day, we are expected to increase our average WPM by 0.43 and accuracy by 0.0016
2) We see a weak linear relationship between both WPM(0.26) and accuracy(0.22).

8) Can we predict the WPM of a switch based on its ID?

Checking ANOVA Regression Conditions:

- We pass all of ANOVA Regression's conditions, which included: Independence, Homoscedasticity, and Normality.

Results:

- **ANOVA Test Results:**

Conducting an ANOVA test:

```
from statsmodels.formula.api import ols

ID_model = ols('WPM ~ C(ID)', data=kb_df).fit()
anova_table = sm.stats.anova_lm(ID_model, typ=2)

print(anova_table)
```

```
                sum_sq     df         F        PR(>F)
C(ID)      6505.943947   12.0  8.98776  1.702210e-16
Residual  56521.955000  937.0      NaN           NaN
```

- **Predicting the WPM of each ID:**

Predicting the WPM of each ID:

```
# Predicted mean WPM per switch
pred_table = kb_df[['ID']].drop_duplicates()
pred_table['Predicted_WPM'] = ID_model.predict(pred_table)
pred_table.tail()
pred_table.sort_values(by = 'ID')
```

	ID	Predicted_WPM
110	1	81.970
530	2	86.780
350	3	77.820
470	4	84.760
0	5	79.685
230	6	84.180
50	7	81.110
290	8	79.460
710	9	83.920
650	10	86.680
410	11	82.360
170	12	77.300
590	13	80.560

- **Sorted by WPM:**

```
#Sorting by WPM:
pred_table.sort_values(by = 'Predicted_WPM')
```

	ID	Predicted_WPM
170	12	77.300
350	3	77.820
290	8	79.460
0	5	79.685
590	13	80.560
50	7	81.110
110	1	81.970
410	11	82.360
710	9	83.920
230	6	84.180
470	4	84.760
650	10	86.680
530	2	86.780

Legend for ID:

1) Curry**, 2)** Xinhai**, 3)** Higanbana**, 4)** Type R**, 5)** Oil King**, 6)** Kiwi**, 7)** Holy Pandas**, 8)** Tealios V2**, 9)** Sonja**, 10)** Melodic**, 11)** Brown**, 12)** Banana Milk**, 13)** Tangerine

Observations:

1) We see that our F-statistic of 8.99 and p-value of close to 0 show that ID is a significant predictor in determining a test's WPM.

2) Of the switches I tested, it seems that we are likely to perform the best with the Xinhais, but worst with the Banana Milks.

3) We also see a range of about 10 WPM between the best/worst performing switches.

---

9) Can we predict the WPM of a switch based on Loudness, Enjoyment, Weight, Sound, and Tactility?

Checking Conditions for Linear Regression:

- We pass all the conditions needed to conduct linear regression, including: Linearity, Independence, Homoscedasticity, Normality, Outliers, and Collinearity. Conditions like Linearity and Homoscedasticity were checked both in a full model setting and individually.

Results:

- **After splitting the dataset:**

Splitting and Training/Testing our data:

```
#Splitting our data:
X_trainset, X_testset, y_trainset, y_testset = train_test_split(x, y, test_size = 0.3, random_state = 123)
```

```
predict_model = LinearRegression()
#Fitting the model:
predict_model.fit(X_trainset, y_trainset)

#Make the predictions:
y_pred = predict_model.predict(X_testset)

#Performance Measurements:
mse = mean_squared_error(y_testset, y_pred)
r2 = r2_score(y_testset, y_pred)

print(f'mse = {mse}')
print(f'r2 = {r2}')
```

```
mse = 61.01553676180331
r2 = 0.014121616359920153
```

- **Inputting values:**

Estimating our WPM given values:

```
#Making an entry:
new_data = pd.DataFrame({
        'Loudness': [65],
        'Weight': [60],
        'Price': [0.6],
        'Tactility': [1]
})

predicted_value = predict_model.predict(new_data)

print(f'The predicted value of a typing test given a a loudness of 65DB, 60g weight, 60 cents a switch, and is a tactile switch is: {predicted_value}')
```

```
The predicted value of a typing test given a a loudness of 65DB, 60g weight, 60 cents a switch, and is a tactile switch is: [83.60921039]
```

What if we made an entry that does not contain values within the range we have sampled from?

```
newer_data = pd.DataFrame({
    'Loudness': [30],
    'Weight': [85],
    'Price': [1.5],
    'Tactility': [0]
})

new_predicted_value = predict_model.predict(newer_data)

print(f'The predicted value of a typing test given a a loudness of 30DB, 85g weight, $1.50 a switch, and is a linear switch is: {new_predicted_value}')
```

```
The predicted value of a typing test given a a loudness of 30DB, 85g weight, $1.50 a switch, and is a linear switch is: [79.45284819]
```

Observations:

1) I achieve an MSE of 61.0155. If we were to estimate, that means on average, our model will likely be about ~8 WPM off.

2) An r2 of 0.014 tells us that the model does not account for much of the variability in WPM at all.

---

10) Given all the predictors(except ID), can we predict the WPM of a test?

Checking for Overfitting (Random Foresting):

```
x = kb_df[['Accuracy', 'Loudness', 'Weight', 'Price', 'Tactility', 'Date Average', 'Date Accuracy', 'days_since_start' ]]
y = kb_df['WPM']

x_trainset, x_testset, y_trainset, y_testset = train_test_split(x, y, test_size = 0.3, random_state = 123)

#Fitting the Random Forest:
rf = RandomForestRegressor(random_state = 123)
rf.fit(x_trainset, y_trainset)

y_train_pred = rf.predict(x_trainset)
y_test_pred = rf.predict(x_testset)

train_r2 = r2_score(y_trainset, y_train_pred)
test_r2 = r2_score(y_testset, y_test_pred)

print(f'train r2: {train_r2}')
print(f'test r2: {test_r2}')
```

```
train r2: 0.8316037294600211
test r2: 0.5934838671331985
```

Results:

```
#Predicting using somewhat average numbers:
rf = RandomForestRegressor(n_estimators = 75, random_state = 123)
rf.fit(x_trainset, y_trainset)

rf_data = [['92', '60', '65', '0.65', '0', '82', '91', '7']]

rf_prediction = rf.predict(rf_data)
print('Predicted WPM', rf_prediction[0])
```

```
Predicted WPM 94.23155555555556
```

```
#Low Numbers:
rf_data = [['82', '50', '40', '0.25', '0', '70', '81', '1']]

rf_low_prediction = rf.predict(rf_data)
print('Predicted WPM', rf_low_prediction[0])
```

```
Predicted WPM 91.94766666666668
```

Observation:

1) The data is capturing some noise: our training set explains about 83% of the variance in new unseen data, while the test set only captures about 59%. Nevertheless, we can capture a good amount of unseen data with the following.

SQL Conclusions:

---

1) Sorting Switches by WPM:

Observations:

1) The mean WPM ranges from 76.9 to 86.8. This is almost a 10 WPM difference across testing dates.

2) Accuracy may be correlated to mean WPM, as when I tend to type faster, there seems to be a positive increase in accuracy.

3) When comparing the original dataset to the updated one, Holy Pandas goes from being in the lower half of the average WPM to the upper half.

---

2) Sorting Switches by Adjusted Mean WPM:

Observations:

1) On average, we perform better with the “daily” switch in comparison to the benchmark (Oil Kings).

2) The EF Curry switch may be an outlier, as its adj_mean(adjusted means) of -6.3 is suspiciously low. However, this is likely due to its rather high benchmark mean for the day(06/29/2025).

3) Comparing the adjusted means, the switches we retested had significant changes. This shows that retesting our data had a significant impact on the way our data was ordered.

---

3) Sorting by Date:

Observations:

1) With each passing date, there is a slight increase in WPM, going from the high 70s to mid-high 80s.

2) The same can be attributed to accuracy, as I started at 89, but then never achieved anything lower than 91 after the 16th.

---

4) Comparing Tactility:

Observations:

1) I performed better on tactile switches, just by observing the averages of both wpm and accuracy.

2) This suggests that I have a slight preference for tactile switches.

---

5) How many tests did I perform equal to or less than 75 WPM?:

Observations:

1) Altogether, I achieved 214 typing tests where I typed 75 WPM or less.

2) On average, I expect to achieve an average of about 14 (187/13) tests in a day with <=75 WPM, but only 9 (27/3) with retesting on days. This could likely be because:

a) I have familiarized myself with each switch; therefore, my performances have gotten better as a result.
b) I have slowly improved with typing since the start of the collection process
c) A mix of both (likely).

---

6) How many tests did I perform equal to or more than 95 WPM?:

Observations:

1) Altogether, we achieved 39 typing tests where I typed 95 WPM or more.

2) On average, I expect to achieve an average of about 3 (33/13) tests in a day with >=95 WPM, but 2 (6/3) with retesting on days.

a) These two are fairly close in trials, so we could assume that while I was getting better at typing (not achieving as many <=75 WPM), we were still achieving roughly the same amount of "very high" typing scores(>=95).

---

7) How many tests did I perform <=75 and >=95(Counted per day)?:

Observations:

1) As time continued throughout the project, we can see fewer >=75 scores appear.

2) All of the >=95 scores achieved during the retesting days were performed on the last day. This is likely during the benchmark testing (recall the rather large daily benchmark recording of 89.7).

3) 4 dates are missing on the >=95 table: 6/16, 6/20, 6/27, 2/28.
4) There is a rather larger number of scores >=95 on 9/25. This was the date we were testing the Melodic, the only clicky switch on our list.

---

8) Of the "fun" switches out there, what were the adj. mean WPM, loudness, enjoyment, tactility, and date statistics?

Observations:

1) Of the switches I found 'fun', I see that I did better with the following switch compared to the benchmark.

2) I enjoyed typing on the switches that were 'fun', with all achieving at least a 7/10, most being a 9/10.

3) A majority of the switches I found fun were high-pitched switches(with the HMX Sonja being the one exception).

Python Conclusions:

---

1) What do the following summary statistics of each predictor tell us?

WPM: If I took the average of all my typing tests, I would type at an average of 81.5 WPM. This is fairly accurate to what my overall WPM is in Monkeytype (81 WPM). The distribution also seems to be normal, outside of 2 outliers.

Accuracy: If I took the average of all my typing tests, I would have a typing accuracy of 91.3%. Like WPM, we are normally distributed, outside of the 4 outliers present. There are likely more outliers for accuracy than WPM because we had a tighter spread for accuracy than WPM, meaning there is less variability.

Loudness: Of the 13 switches we tested, the average loudness (measured in dB) was about 68 DB. All things considered, this is fairly loud. However, this is likely due to the method I record the sound of the switches. We also have a skewed left distribution, telling me that we have switches that hover around 68+ more than <68.

Price: The average cost of one switch in the experiment is about 56 cents, which is fairly expensive in today's market. However, this is likely due to the majority of our switches being "older".

Weight: The average weight of all of the switches I tried was about 62.4 grams. I believe this is about average to above-average weight for switches today.

Date Average: When taking the average of all 16 testing days, my mean Daily WPM is 81.6. This is very close to our Average WPM.

Date Accuracy: My mean Daily accuracy is 91.5%, which is slightly higher than my Average Accuracy.

Conclusions:

1) We can see some outliers present in the WPM and Accuracy box plots, telling us that there were some extreme cases while I was typing.

2) Predictors may be skewed thanks to the inclusion of multiple runs of the Gateron Oil Kings, Tangerines, Holy Pandas, and Curry Switches.

---

2) What does the distribution of each switch look like?

1) Most of our tests were between mid-70s to low-90s.

2) We see some outliers from the Tealios V2 (ID 8), and the Oil King, which has a very obvious outlier around ~40. The Tealios is an interesting case, as all of the other typing tests hovered around 65-90.

3) The Oil king seems to have the widest variance among all of the switches, showing a large range of values ranging from ~40s to ~100. This is likely due to typing daily on them before every experiment.

---

3) Is there a correlation between weight and WPM?

- There is a weak negative monotonic relationship between Weight and WPM(= -0.12).

- Our small p-value (close to 0) indicates that there is a statistically significant monotonic relationship between Weight and WPM.

Conclusion:

1) While our relationship is unlikely to occur purely through chance(small p-value), the effect of the relationship is weak. This likely means that other factors have a larger effect on WPM than Weight.

---

4) Is there a correlation between price and WPM?:

- There is a weak negative monotonic relationship between Weight and WPM(= -0.10).

- Our small p-value (close to 0) indicates that there is a statistically significant monotonic relationship between Weight and WPM.

Conclusions:

1) While this relationship is unlikely to occur purely through chance, the effect of the relationship is weak.

2) Seeing that we achieve similar results in both tests, I believe a reason for this is our repetition in data(for every one different price/weight category, there are 50 entries). This similarity in sample sizes/ranks leads to us having very little variety, resulting in our correlation being created through 13 categories.

---

5) Construct a 95% confidence interval for WPM.

- We are 95% confident that the confidence interval (81.022, 82.06) would contain the mean Words Per Minute when performing a MonkeyType test.

Conclusion:

1) Our confidence interval has a range of about 1 word per minute. This is likely due to the high volume of data we had, creating a more precise and accurate range of our mean WPM.

---

6) Construct a 95% confidence interval for adjusted WPM.

- We would be 95% confident that the confidence interval (0.29, 0.66) would contain the mean adjusted Words Per Minute(daily switch mean - daily benchmark mean), given that our data was normally distributed.

Conclusions:

1) We are fairly confident of seeing a slight increase in our WPM in comparison to the benchmark switch.

2) The lack of categories and data is starting to show when we check for normality in all of these tests, because we don't have enough variance in our data; we are left with clusters of data and spread-out distributions, rather than a cohesive histogram.

---

7) Is there a relationship between dates and average WPM/ accuracy?

- Every day, we are expected to increase our average WPM by 0.43 and accuracy by 0.0016

- We see a weak linear relationship between both WPM(0.26) and accuracy(0.22).

Conclusions:

1) This is likely because I cannot continuously improve in typing accuracy and WPM every day, as many factors affect these statistics.

2) It's also unreasonable to expect the same amount of growth day-to-day, as while I may improve drastically after a few days of continuous typing, eventually I am going to plateau in performance, only increasing in small increments.

---

8) Can we predict the WPM of a switch based on its ID?

- We see that our F-statistic of 8.99 and p-value of close to 0 show that ID is a significant predictor in determining a test's WPM.

- Of the switches I tested, it seems that we are likely to perform the best with the Xinhais, but worst with the Banana Milks.

- We also see a range of about 10 WPM between the best/worst performing switches.

Conclusions:

1) Given that each switch has many different factors (typing feel, weight, sound, bias, etc), it's no surprise that the switch itself would have a large impact on my WPM performance.

2) When looking at the order of the predicted performances, it seems that it's taking all of the data that we have accumulated and taking the average of each. This is slightly different from our summary table, as for switches I retested, I only took the average of the 50 new retests. The following table instead takes the average for all tests associated with a test, giving us an unbalanced result, where each switch is weighed differently.

---

9) Can we predict the WPM of a switch based on loudness, weight, price, and tactility?

- I achieve an MSE of 61.0155. If we were to estimate, that means on average, our model will likely be about ~8 WPM off.

- An r^2 of 0.014 tells us that the model does not account for much of the variability in WPM at all.

Conclusion:

1) Because of our high MSE and low r2, we can only make rough estimates of what our data will be like with the following set of predictors

---

10) Given all the predictors, can we predict the WPM of a test?

- It seems that our data is capturing some noise: our training set explains about 83% of the variance in new unseen data, while the test set only captures about 59%. Nevertheless, we can capture a good amount of unseen data with the following.

Conclusions:

1) Our model is likely capturing noise and overfitting. When comparing our test and training r2, we see a fairly large difference between the two (0.24 difference).

2) The predictions are fairly large, too, given that I selected predictors like Date WPM to be fairly low(70), yet we achieve a high WPM (92). While not impossible, these inaccuracies are likely due to the noise and imperfect model fit.

Future Improvements:

1st Improvement: Sample Size:

While I was able to improve and achieve better results with the use of our current dataset, it's obvious that if we had a larger variety in data (more switches), we likely would have been able to achieve more precise and accurate analysis data.

2nd Improvement: Sampling Methods:

I wish I had recorded more data when performing each typing test. For example, why not also record the Average DB of every typing test? This would have given us far more accurate sound measurements rather than using the average of 30 clicks of each switch.