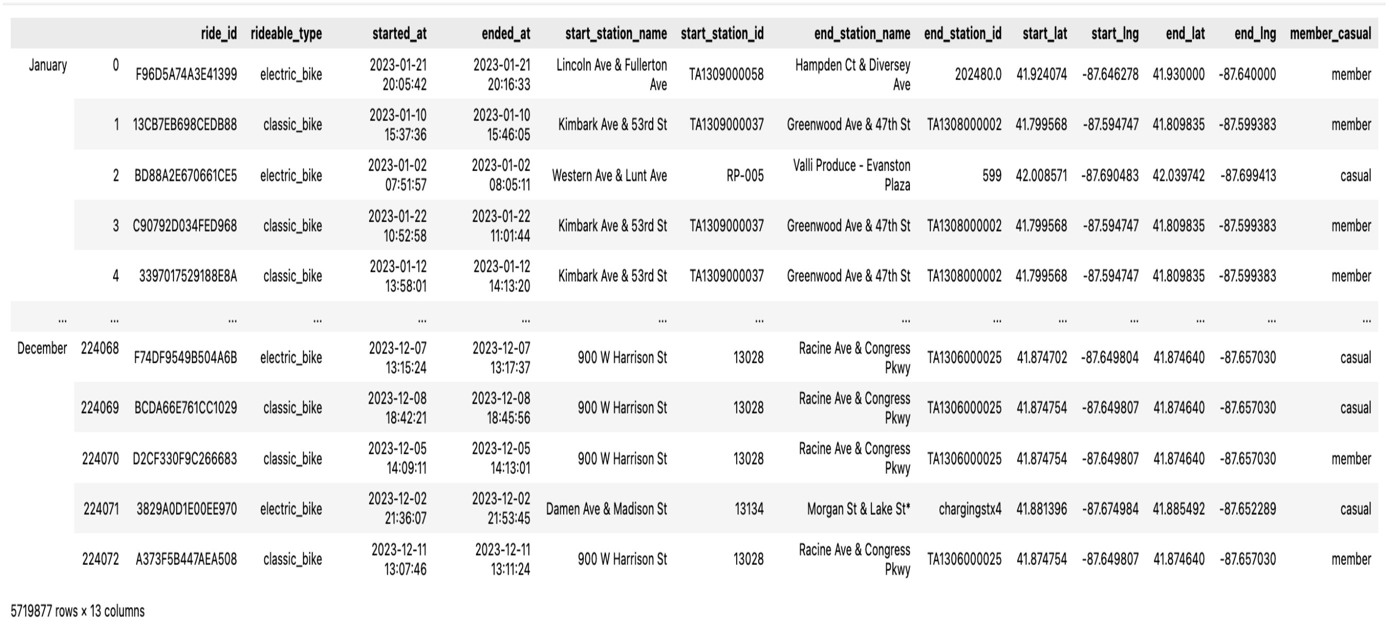
Bike-Share Case Study

This report provides the results and step-by-step explanation of the data analysis performed for a bike-sharing case study. The data belongs to a bike-sharing company that has two kinds of users: annual members and casual riders. The goal of the study was to identify how annual members and casual riders use the bikes differently in order to help the stake-holders decide whether or not to target converting casual riders into annual members in the next marketing campaign. The library Panadas from Python was used to perform the analysis using data collected from January-December 2023 and downloaded from https://divvy-tripdata.s3.amazonaws.com/index.html.

The code that was used to perform the data analysis can be found in the Jupyter Notebook bike_share_analysis.ipynb.

Cleaning:

- *read_data*:
  The original data was stored such that each month was in a separate .csv file. So in this method the data from each month is read and stored into a DataFrame (DF), and then the 12 DFs are concatenated into one multi-index DF. Using a multi-index DF has several advantages. First, the distinction between the different seasons/months can still be maintained (data from different months are not fused together into one large DF). Second, the multi-index DF facilitates finding and aggregating values across different months when needed. In Figure 1 we can see the first and last 5 entries of bike rides from the multi-index DF:



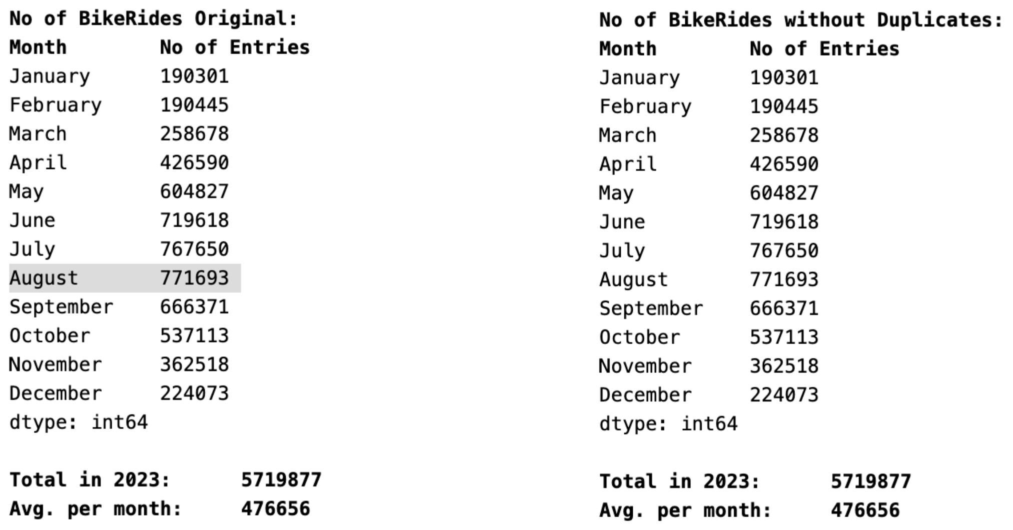
		ride_id	rideable_type	started_at	ended_at	start_station_name	start_station_id	end_station_name	end_station_id	start_lat	start_lng	end_lat	end_lng	member_casual
January	0	F96D5A74A3E41399	electric_bike	2023-01-21 20:05:42	2023-01-21 20:16:33	Lincoln Ave & Fullerton Ave	TA1309000058	Hampden Ct & Diversey Ave	202480.0	41.924074	-87.646278	41.930000	-87.640000	member
	1	13CB7EB698CEDB88	classic_bike	2023-01-10 15:37:36	2023-01-10 15:46:05	Kimbark Ave & 53rd St	TA1309000037	Greenwood Ave & 47th St	TA1308000002	41.799568	-87.594747	41.809835	-87.599383	member
	2	BD88A2E670661CE5	electric_bike	2023-01-02 07:51:57	2023-01-02 08:05:11	Western Ave & Lunt Ave	RP-005	Valli Produce - Evanston Plaza	599	42.008571	-87.690483	42.039742	-87.699413	casual
	3	C90792D034FED968	classic_bike	2023-01-22 10:52:58	2023-01-22 11:01:44	Kimbark Ave & 53rd St	TA1309000037	Greenwood Ave & 47th St	TA1308000002	41.799568	-87.594747	41.809835	-87.599383	member
	4	3397017529188E8A	classic_bike	2023-01-12 13:58:01	2023-01-12 14:13:20	Kimbark Ave & 53rd St	TA1309000037	Greenwood Ave & 47th St	TA1308000002	41.799568	-87.594747	41.809835	-87.599383	member
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
December	224068	F74DF9549B504A6B	electric_bike	2023-12-07 13:15:24	2023-12-07 13:17:37	900 W Harrison St	13028	Racine Ave & Congress Pkwy	TA1306000025	41.874702	-87.649804	41.874640	-87.657030	casual
	224069	BCDA66E761CC1029	classic_bike	2023-12-08 18:42:21	2023-12-08 18:45:56	900 W Harrison St	13028	Racine Ave & Congress Pkwy	TA1306000025	41.874754	-87.649807	41.874640	-87.657030	casual
	224070	D2CF330F9C266683	classic_bike	2023-12-05 14:09:11	2023-12-05 14:13:01	900 W Harrison St	13028	Racine Ave & Congress Pkwy	TA1306000025	41.874754	-87.649807	41.874640	-87.657030	member
	224071	3829A0D1E00EE970	electric_bike	2023-12-02 21:36:07	2023-12-02 21:53:45	Damen Ave & Madison St	13134	Morgan St & Lake St*	chargingstx4	41.881396	-87.674984	41.885492	-87.652289	casual
	224072	A373F5B447AEA508	classic_bike	2023-12-11 13:07:46	2023-12-11 13:11:24	900 W Harrison St	13028	Racine Ave & Congress Pkwy	TA1306000025	41.874754	-87.649807	41.874640	-87.657030	member

5719877 rows × 13 columns

Figure 1: First and last 5 entries of bike rides from the original_data

In Figure 1 the multi-index of the DF is shown in the first two columns (month, row_index). Then looking at the entries themselves we can see that the data consists of 13 columns: 1) ride id, 2) type of bike, 3-4) date and time for the start and end of the ride, 5-12) the name, id, latitude and longitude of the start and end stations, and 13) whether the rider was a casual rider or a member.

- *count_entries*:
  The method *count_entries* is used to find the number of entries within each month, and the average per month. The result is shown in Figure 2 left. The dataset contains in total almost 5.7 Million entries, with an average of approximately 480,000 entries per month. From Decemeber to March the number of rides is relatively lower than the average, which is expected as these are cold months. This is confirmed by the peak highlighted in August. After retrieving this information for the original dataset, the duplicates are dropped, and the method is called again. The result of running the method after dropping the duplicates is shown in Figure 2 right. The number of entries before and after is identical, therefore the original dataset did not have any duplicates.



```
No of BikeRides Original:
Month          No of Entries
January        190301
February       190445
March          258678
April          426590
May            604827
June           719618
July           767650
August         771693
September      666371
October        537113
November       362518
December       224073
dtype: int64

Total in 2023:        5719877
Avg. per month:       476656
```

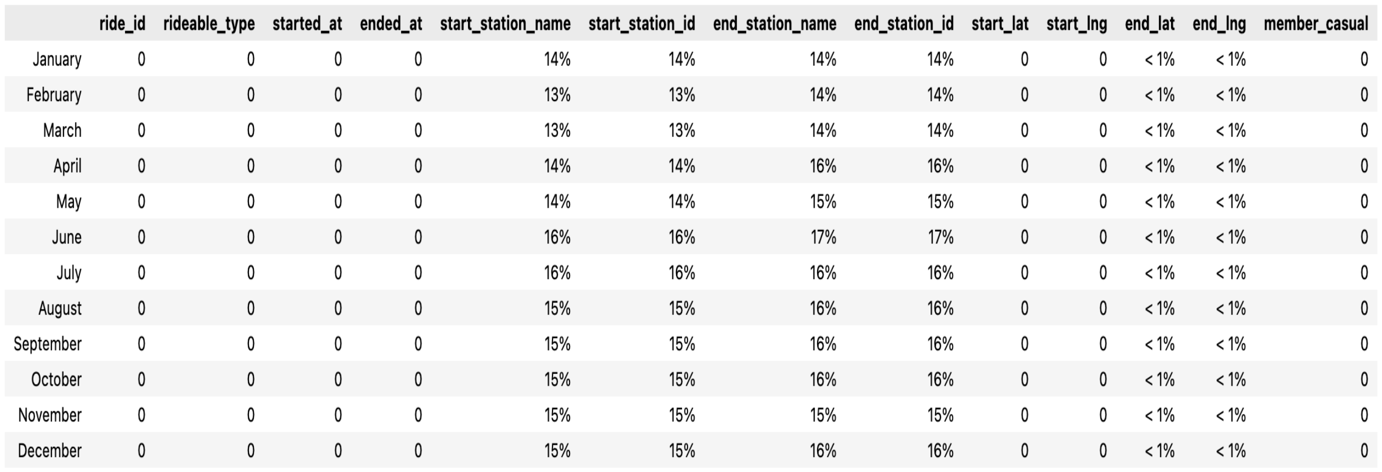
```
No of BikeRides without Duplicates:
Month          No of Entries
January        190301
February       190445
March          258678
April          426590
May            604827
June           719618
July           767650
August         771693
September      666371
October        537113
November       362518
December       224073
dtype: int64

Total in 2023:        5719877
Avg. per month:       476656
```

Figure 2: No of entries and average before and after removing duplicates

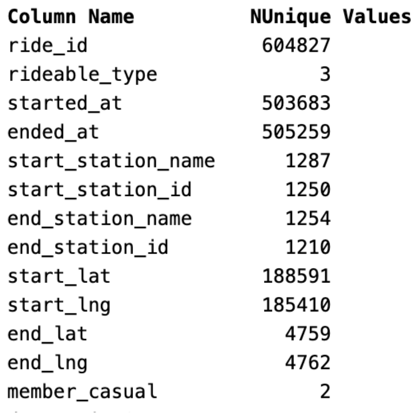
- *get_null_percentage*:
  This method calculates the percentage of null values for each column and month. The results are shown in Figure 3. As we can see the columns *start_station_name*, *start_station_id*, *end_station_name*, *end_station_id* in every month have 13-17% null values. The columns *end_lat* and *end_long* have less than 1% null values. In order to explore the dataset, as well as these null values a bit further, the number of unique (distinct) values for each column in original_data["May"] is calculated and shown in Figure 4. The choice of the month of May is random and should not make a difference.

  - *ride_id_unique* = 604827: as expected, *ride_id* has as many unique values as the number of entries in the dataframe.
  - *rideable_type_unique* = 3: these 3 unique values are the types of bikes: [electric_bike, classic_bike, docked_bike].
  - *started(ended)_at_unique* = 503683, 505259: since these are datetimes (yy-mm-dd hh:mm:ss), one may have expected that they would have as many distinct values as the number of entries, since it seems unlikely for more than one rider to have rented a bike at the exact same time down to the second. However, the number of unique values in these columns is less than the number of entries by 17%. To ensure that these are not duplicate entries but with different



	ride_id	rideable_type	started_at	ended_at	start_station_name	start_station_id	end_station_name	end_station_id	start_lat	start_lng	end_lat	end_lng	member_casual
January	0	0	0	0	14%	14%	14%	14%	0	0	< 1%	< 1%	0
February	0	0	0	0	13%	13%	14%	14%	0	0	< 1%	< 1%	0
March	0	0	0	0	13%	13%	14%	14%	0	0	< 1%	< 1%	0
April	0	0	0	0	14%	14%	16%	16%	0	0	< 1%	< 1%	0
May	0	0	0	0	14%	14%	15%	15%	0	0	< 1%	< 1%	0
June	0	0	0	0	16%	16%	17%	17%	0	0	< 1%	< 1%	0
July	0	0	0	0	16%	16%	16%	16%	0	0	< 1%	< 1%	0
August	0	0	0	0	15%	15%	16%	16%	0	0	< 1%	< 1%	0
September	0	0	0	0	15%	15%	16%	16%	0	0	< 1%	< 1%	0
October	0	0	0	0	15%	15%	16%	16%	0	0	< 1%	< 1%	0
November	0	0	0	0	15%	15%	15%	15%	0	0	< 1%	< 1%	0
December	0	0	0	0	15%	15%	16%	16%	0	0	< 1%	< 1%	0

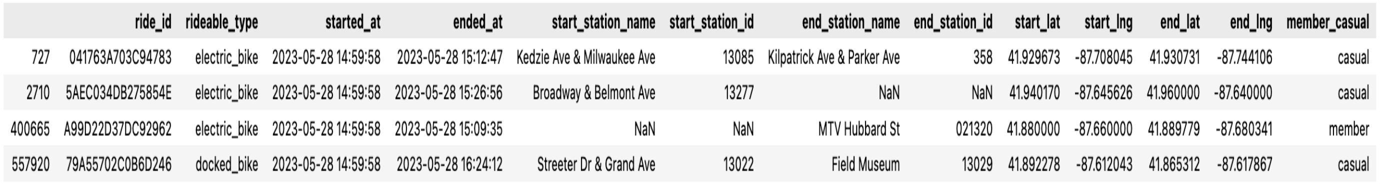
Figure 3: Percentage of null values for each column across the months



```
Column Name          NUnique Values
ride_id               604827
rideable_type              3
started_at            503683
ended_at              505259
start_station_name      1287
start_station_id        1250
end_station_name        1254
end_station_id          1210
start_lat             188591
start_lng             185410
end_lat                 4759
end_lng                 4762
member_casual              2
```

Figure 4: No. of unique values for every column in original_data[“May”]

ride_id, one of these incidents has been retrieved and is shown in Figure 9. By looking at the values, it is clear that they are indeed different entries but with the exact same start time.



	ride_id	rideable_type	started_at	ended_at	start_station_name	start_station_id	end_station_name	end_station_id	start_lat	start_lng	end_lat	end_lng	member_casual
727	041763A703C94783	electric_bike	2023-05-28 14:59:58	2023-05-28 15:12:47	Kedzie Ave & Milwaukee Ave	13085	Kilpatrick Ave & Parker Ave	358	41.929673	-87.708045	41.930731	-87.744106	casual
2710	5AEC034DB275854E	electric_bike	2023-05-28 14:59:58	2023-05-28 15:26:56	Broadway & Belmont Ave	13277	NaN	NaN	41.940170	-87.645626	41.960000	-87.640000	casual
400665	A99D22D37DC92962	electric_bike	2023-05-28 14:59:58	2023-05-28 15:09:35	NaN	NaN	MTV Hubbard St	021320	41.880000	-87.660000	41.889779	-87.680341	member
557920	79A55702C0B6D246	docked_bike	2023-05-28 14:59:58	2023-05-28 16:24:12	Streeter Dr & Grand Ave	13022	Field Museum	13029	41.892278	-87.612043	41.865312	-87.617867	casual

Figure 5: Entries from original_data[“May”] that have the same *started_at*

– *start(end)_station_name(id)_unique* = 1287, 1250, 1254, 1210: since there is a limited number of stations, it is expected that these columns have a smaller number of unique values than the number of entries. However, one would have expected the number of unique station names and station ids to be the same, whereas the ids are less than the names by a small fraction. Which could either by accounted for by the null values or could mean that there are stations that have the different names but the same id.

– *start(end)_lat(long)_unique* = 188591, 185419, 4759, 4762: the start latitude and longitude numbers seem to be as expected, which is less than the total number of entries, but more than the number of stations (this is based on the assumption that the exact location where a bike is docked can vary within the station especially that the values are given to the 6^{th} decimal place).

However, there is a large difference between the number of values in the start (∼188000) and the numbers in the end (∼4800) latitude and longitude. This difference cannot be accounted for by the null values in the end columns, since these were less than 1%. If these values are true, that would mean that users rode there bikes from many start points, but ended up in a much smaller set of points. Which cannot be the case since that would have been reflected in the number of end stations.

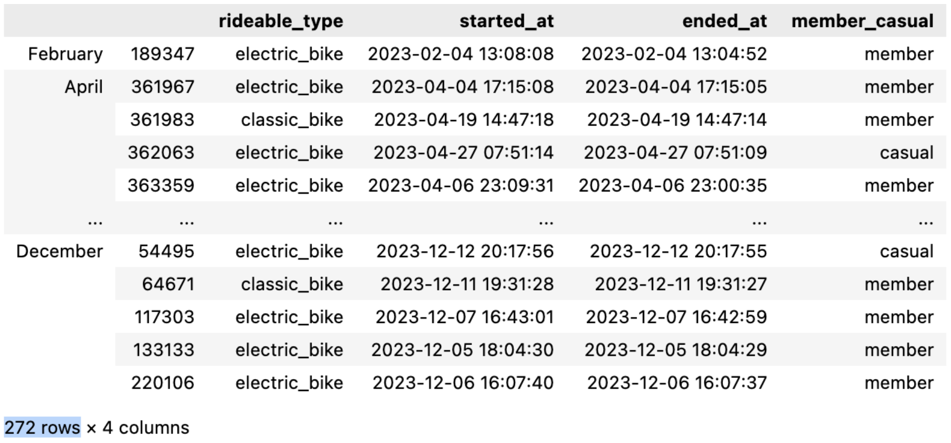
  – *member_casual_unique* = 2: these 2 unique values are: [member, casual].

- *clean_data*:
  After exploring the dataset, we can see that the extractable information can be divided into information about the:

  1. rider (casual/member)
  2. bike (electric/classic/docked)
  3. ride (start-end: time, date, location)

  Since, the goal of the analysis is to find out whether to target converting casual riders into members or not, it seems that all the information is relevant to the analysis, with the exception of the ride location. The geographical location would have been important if for example the goal of the analysis was to find out whether more stations should be added and where to do so. Therefore, since the locations seem to be irrelevant, contain null values and discrepancies, these columns will be dropped. The column *ride_id* also does not provide any valuable information for the current analysis. Thus, the method *clean_data1* drops all columns related to geographical location and ride id, which leaves: *rideable_type*, *started_at*, *ended_at*, *member_casual*.

  Next, is data formatting. We have already looked at the columns *rideable_type* and *member_casual*, and ensured that they have the expected values. As for the columns *started_at* and *ended_at*, we need to make sure that the *ended_at* time always comes after *started_at* time. In order to compare the values, they are first converted in the method *clean_data1* from strings of characters to a numerical datetime format. Next, by comparing the values and filtering them, we find that there are indeed cases where the *ended_at* time is actually before the *started_at* time. These cases for the month of May are shown in Figure 6.



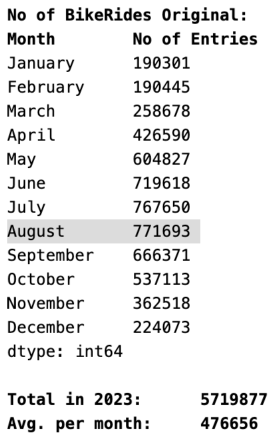
		rideable_type	started_at	ended_at	member_casual
February	189347	electric_bike	2023-02-04 13:08:08	2023-02-04 13:04:52	member
April	361967	electric_bike	2023-04-04 17:15:08	2023-04-04 17:15:05	member
	361983	classic_bike	2023-04-19 14:47:18	2023-04-19 14:47:14	member
	362063	electric_bike	2023-04-27 07:51:14	2023-04-27 07:51:09	casual
	363359	electric_bike	2023-04-06 23:09:31	2023-04-06 23:00:35	member
...	...	...	...	...	...
December	54495	electric_bike	2023-12-12 20:17:56	2023-12-12 20:17:55	casual
	64671	classic_bike	2023-12-11 19:31:28	2023-12-11 19:31:27	member
	117303	electric_bike	2023-12-07 16:43:01	2023-12-07 16:42:59	member
	133133	electric_bike	2023-12-05 18:04:30	2023-12-05 18:04:29	member
	220106	electric_bike	2023-12-06 16:07:40	2023-12-06 16:07:37	member

272 rows × 4 columns

Figure 6: Entries in May when the ended_at time is before the started_at time

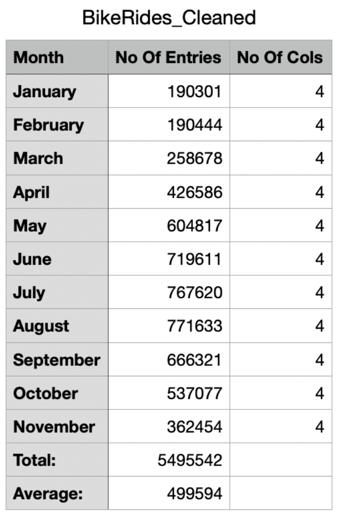
  Looking at the entries in Figure 6, we can see that these are cases when the *ended_at* time is before the *started_at* time by just a few seconds. It can be that in these incidents the start and end time were switched due to some glitch, perhaps the bike rental time being only a few seconds (shorter than the server response time). In the entire dataset of approximately 5.5 Million entries, there is a total of 262 entries that have this issue. Since, the dataset is large, we can simply drop these entries.

Finally, after dropping the columns related to location, and the entries with the switched times, the method *count_entries* is called once again and used to compare the cleaned dataset to the original one. The result is shown in Figure 7. As we can see the number of columns has changed from 13 to 4, and the number of entries is slightly less due to removing the entries where the *ended_at* time is before the *started_at* time.



```
No of BikeRides Original:
Month        No of Entries
January      190301
February     190445
March        258678
April        426590
May          604827
June         719618
July         767650
August       771693
September    666371
October      537113
November     362518
December     224073
dtype: int64

Total in 2023:      5719877
Avg. per month:     476656
```



BikeRides_Cleaned

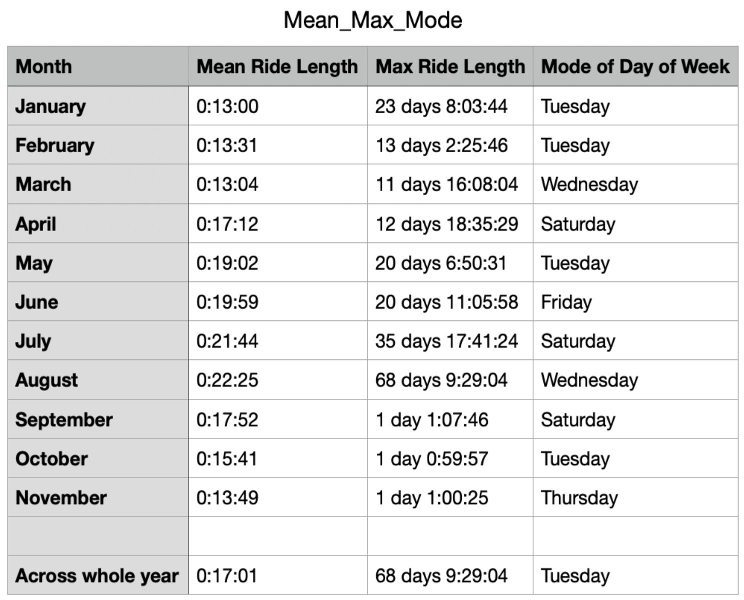
Month	No Of Entries	No Of Cols
January	190301	4
February	190444	4
March	258678	4
April	426586	4
May	604817	4
June	719611	4
July	767620	4
August	771633	4
September	666321	4
October	537077	4
November	362454	4
Total:	5495542	
Average:	499594	

Figure 7: No of entries and columns before and after cleaning

Analysis:

- *prepare_data*:
  The method *prepare_data* adds two new columns: *ride_length*: the difference between the columns *ended_at* and *started_at* times, and *day_of_week*: extracted from the date in *started_at*.

- *statistics*:
  The method *statistics* calculates for each month the average and maximum ride length, as well as the mode of the day of the week; i.e. the day of the week that occurred the most. The results are shown in Figure 8. The mean ride length varies between 13 - 22 minutes across the months, whereas the maximum ride length is 68 days.



Mean_Max_Mode

Month	Mean Ride Length	Max Ride Length	Mode of Day of Week
January	0:13:00	23 days 8:03:44	Tuesday
February	0:13:31	13 days 2:25:46	Tuesday
March	0:13:04	11 days 16:08:04	Wednesday
April	0:17:12	12 days 18:35:29	Saturday
May	0:19:02	20 days 6:50:31	Tuesday
June	0:19:59	20 days 11:05:58	Friday
July	0:21:44	35 days 17:41:24	Saturday
August	0:22:25	68 days 9:29:04	Wednesday
September	0:17:52	1 day 1:07:46	Saturday
October	0:15:41	1 day 0:59:57	Tuesday
November	0:13:49	1 day 1:00:25	Thursday
Across whole year	0:17:01	68 days 9:29:04	Tuesday

Figure 8: The mean and max ride length, and mode of day of the week

Figure 9: The mean and max ride length, and mode of day of the week for members and casual riders separately