Case Study

How Does a Bike-Share Navigate Speedy Success?

Index

1. A clear statement of the business task (ASK)

About Cyclistic

Cyclistic, a Chicago-based bike-sharing company that offers both one-time and annual memberships, sets itself apart by also offering reclining bikes, hand tricycles, and cargo bikes, making bike-share more inclusive to people with disabilities and riders who can't use a standard two-wheeled bike. The majority of riders opt for traditional bikes; about 8% of riders use the assistive options. Cyclistic users are more likely to ride for leisure, but about 30% use them to commute to work each day.

The request

The company aims to design marketing strategies aimed at converting one-time riders into annual members.

The request for analytics to support marketing strategies will assess: **How do annual members and occasional cyclists use Cyclistic bikes differently??**

The Stakeholders

- **Lily Moreno**: The director of marketing is the sponsor of aim of the initiatives to converting one-time riders into annual members and this analysis.
- **Cyclistic marketing analytics team**: A team of data analysts who are responsible for collecting, analyzing, and reporting data that helps guide Cyclistic marketing strategy and so they are interested that the analysis was well done.
- **Cyclistic executive team**: They expect a good report detail-oriented to decide whether to approve the recommended marketing program
- **Occasional Cyclists**: This users could receive benefits becoming annual members

2. A description of all data sources used (Preparation)

The data source is a public folder of type **second part** published by the company *Motivate International Inc* under an open source license. The data is present on source divvy-tripdata and anonymous.

The folder contains data for a travel period divided by month of use, according to a format **Long data**, but there is no metadata file that explains the characteristics of the data.

Since the analysis is done only on the last 12 months and a manual operation of downloading and loading the 12 files on the systems was chosen without using automation tools such as API/ETL.

A folder was created on the cloud to make it accessible to a work team **/projects/2024/capstone/bike_share/dataset/csv** on which 12 ZIP files relating to the period have been uploaded 10/2023 - 9/2024.

The files are CSV compressed in ZIP divided by month of extraction.

To distinguish the versions of the files with respect to the sampling period of interest, the following nomenclature is used **AAAAMM-tripdata_AAAAMMDD.csv**. The first date indicates the reference period of the data and is static. The second date indicates the insertion/modification date to historicize the different versions of the file. Thanks this format is more easy find the last version of a same file or download the last version of all file if there are several update dates.

The files that make up the datasets have the following characteristics:

file name	dimension	MD5	righe
202310-tripdata-20241029.csv	105.3 Mb	4e71b4e49567fd13d0e01f72cbe5b2a7	537113
202311-tripdata-20241029.csv	71.3 Mb	9cf7cf1704779fbc1cd712de586523ad	362518
202312-tripdata-20241029.csv	43.9 Mb	0e2a48fcf487c75624672325aa7afc9b	224073
202401-tripdata-20241029.csv	28.7 Mb	7e4e37507051bf3b2e338a3126187c68	144873
202402-tripdata-20241029.csv	44.7 Mb	1e4d9a04cbbb14f9f5a36bf7c2c1c319	223164
202403-tripdata-20241029.csv	59.3 Mb	88267643679a17a3b218641832d6af0b	301687
202404-tripdata-20241029.csv	80.1 Mb	a89b2fd37bbd4d6113bb460fb2eb6984	415025
202405-tripdata-20241029.csv	117.7 Mb	875acd3626fb0d49ffe1dd1381d85aab	609493
202406-tripdata-20241029.csv	141.3 Mb	afe1ecf091a3eec10ce70166a7ada0ad	710721
202407-tripdata-20241029.csv	149.9 Mb	d0fbd3e9d186fd736bbb715865c1998f	748962
202408-tripdata-20241029.csv	150.9 Mb	d189e2cab64aea2c86e5a0d590d12800	755639
202409-tripdata-20241029.csv	161.0 Mb	95f1502c287f569c8ef5776bf711f583	821276

- Total rows: **5854544**
- Total size 1.1 Gb
- All files have a homogeneous structure with the same number of columns and the same labels

To use data of this large size it was not possible to open them with Google Sheets, even performing simple processing on individual files the system would crash, so we opted to use:

- **Microsfot Excel** using the function **Power Query** that It allows to handle big data like Tibble in R
- **BigQuery**.

Bigquery does not allow you to create tables directly from CSV files larger than 100Mb. A Google Bucket has been created to manage large files. The name the bucket is **ev-divvy-tripdata**, where the 12 files have been uploaded. Thanks to the presence of homogeneous files in buckets it was possible to load all the data present on multiple sources into a single table with a single operation, putting as input source *gs://ev-divvy-tripdata/*.

The following structure was created on Bigquery:

- dataset: **tripdata**
- table: **triptrack**

To verify that the data saved in the Cloud are correct, by downloading the files from buckets or Cloud, the hash was recalculated and compared with that in the table.

To verify further that the data was correctly imported and its integrity, we counted the data in the table that results 5854544 as the sum of the rows in the CSV files (Excluded header rows).

License, privacy, security and accessibility

The data provided is made available in open source mode under the following license.

This license allows to access, reproduce, analyze, copy, modify, distribute in your product or service and use the Data for any lawful purpose.

The data doesn't contain personal information.

The original data are stored online and there isn't required any authorization. the Local copy of data is stored in cloud and requires authorization to access, the license is also store together with the data, so if someone would share this data must export it following license indications.

ROCCC Analysis

- **Reliable**: the dataset isn't reliable, because there isn't information in the site store if the dataset covers all trips or are some samples and so if it is bias selection.
- **Original**: the data are original because use data published by a company that harvests data of bikes share of own service in Chicago.
- **Comprehensive**: the data aren't comprehensive, We cannot analyze customers' habits, due of in the data there isn't information about the relations between different trip of the same costumer, would be useful to indicate an anonymous identification of a customer like a pseudo id code and there isn't information to identify how much the stations cover the Chicago's neighborhoods, to understand if the problem to became member depends by far distance to reach the station and so it becomes difficult to use this service frequently or daily.
- **Current**: the datasets are currents, the datasets are upgraded each month with data of last month
- **Cited**: is define the source of the dataset

Conclusion: The dataset isn't ROCCC.

Process phase

Choos of the tools - Bigquery and Google sheet

To perform the process phase we will use Bigquery, because the dataset has more than a million data points and the spreadsheet cannot handle so many rows well. Using excel with powerquery we might be able to work with so many rows but the steps are similar to working in SQL it is preferable to work directly in SQL, for analysis you can export the data to R. Furthemore, in Bigquery there is also functions to do PIVOT to create complex relation with datas and it's possible to show the result in simple graphs (line chart or bar chart or dispersion chart). If you need to represent complex relationships with data or graphs there is a function in BigQuery to load the results of a query directly into the Google Sheet, allowing you to process the data with other operations and create graphs.

Data Integrity

We create a copy of original data so to rollback after manipulation.

Metadata of Dataset

the datasets have the following columns:

Name	type	description	
ride_id	[string]	unique identifier of the ride on bike	
rideable_type	[category]	type of bike [electric_bike, electric_scooter, classic_bike]	
started_at	[datetime]	start data of the ride	
ended_at	[datetime]	end data of the ride	
start_station_name	[string]	start station name	
start_station_id	[string]	start station identification code	
end_station_name	[string]	end station name	
end_station_id	[string]	end station identification code	
start_lat	[float]	position of start latitude	
start_lng	[float]	position of start longitude	
end_lat	[float]	position of end latitude	
end_lng	[float]	position of end longitude	
member_casual	[category]	type of customer [MEMBER	CASUAL]

3. Documentation of any cleaning or manipulation of data ## Data Cleaning Analysis

Analyze Qualitative values

```
SELECT distinct member_casual from `tripdata.triptrack` ;
```

member_casual
casual
member

```
SELECT count( distinct start_station_name) as start from `tripdata.triptrack` order by start;

SELECT count( distinct trim(start_station_name)) as start from `tripdata.triptrack` order by start;
```

Executing both queries result that the count is 1737, to indicate that it isn't necessary to apply trim operation on the start_station_name value.

```
SELECT count( distinct end_station_name) as end_name from `tripdata.triptrack` order by end_name;

SELECT count( distinct trim(end_station_name)) as end_name from `tripdata.triptrack` order by end_name
```

Executing both queries result that the count is 1749, to indicate that isn't necessary to apply trim operation on the end_station_name values.

```
SELECT distinct rideable_type from `tripdata.triptrack` ;
```

rideable_type
electric_bike
electric_scooter
classic_bike

Searching for Null values

```
SELECT
   count(month) as total,
   (select count(month)  FROM `tripdata.triptrack` where start_station_id is null) as null_start_station_id,
   (select count(month)  FROM `tripdata.triptrack` where end_station_id is null) as null_end_station_id,
   (select count(month)  FROM `tripdata.triptrack` where start_station_name is null) as null_start_station_name,
   (select count(month)  FROM `tripdata.triptrack` where end_station_name is null) as null_end_station_name,
   (select count(month)  FROM `tripdata.triptrack` where rideable_type is null) as null_rideable_type,
   (select count(month)  FROM `tripdata.triptrack` where member_casual is null) as null_member_casual,
   (select count(month)  FROM `tripdata.triptrack` where started_at is null) as null_started_at,
   (select count(month)  FROM `tripdata.triptrack` where ended_at is null) as null_ended_at,
   (select count(month)  FROM `tripdata.triptrack` where start_lat is null) as null_start_lat,
   (select count(month)  FROM `tripdata.triptrack` where end_lat is null) as null_end_lat,
   (select count(month)  FROM `tripdata.triptrack` where start_lng is null) as null_start_lng,
   (select count(month)  FROM `tripdata.triptrack` where end_lng is null) as null_end_lng
 FROM `tripdata.triptrack`
```

Column	number
total	5854544
start_station_name	1056535
end_station_name	1091792
start_station_id	1056535
end_station_id	1091792
rideable_type	0
member_casual	0
started_at	0
ended_at	0
start_lat	0
end_lat	7441
start_lng	0
end_lng	7441

the dataset presents some null values in the station names and station ids, the other informations are presents, but the null values don't influence our aim.

Searching for Outlayer values

Analyze min e max of quantity values and datetime

```
SELECT
   max(started_at)   as max_started_at,
   max(ended_at)   as min_ended_at,
   max(start_lat)   as max_start_lat,
   max(end_lat)   as max_end_lat,
   max(start_lng)   as max_start_lng,
   max(end_lng)   as max_end_lng,
   min(started_at)   as min_started_at,
   min(ended_at)   as min_ended_at,
   min(start_lat)   as min_start_lat,
   min(end_lat)  as min_end_lat,
   min(start_lng)   as min_start_lng,
   min(end_lng)   as min_end_lng
FROM `tripdata.triptrack`
```

Field	MIN	MAX
started_at	2024-09-30 23:54:05.552000 UTC	2023-10-01 00:00:05.000000 UTC
ended_at	2024-09-30 23:59:52.562000 UTC	00:02:02.000000 UTC
start_lat	42.07	2023-10-01 41.64
end_lat	87.96	16.06
start_lng	-87.52	-87.94
end_lng	1.72	-144.05

Thanks the max and min value for latitude and longitude it's possible to notice an error. Chicago has a latitude, longitude of 41.881832, -87.623177.

In the dataset thanks query that extract min and max instead there are values:

- latitude end station between 87.9 and 16, so none of the values are correct
- longituted end station between 1.72 and 182, very distant from -87.62.

On internet founded the approximatily latitude and longituted limits of Chicago, respectely [41,42.5] and [-88,-87]. Executing the following query to investigate

```
SELECT * `tripdata.triptrack`where end_lat not between 41 and 42.5;

SELECT count(*) as outlayer_lat_end from `tripdata.triptrack`where end_lat not between 41 and 42.5;

SELECT * from `tripdata.triptrack`where end_lng not between -88 and -87;

SELECT count(*) as outlayer_lng_end from `tripdata.triptrack`where end_lng not between -88 and -87;
```

when there is a end_lat or end_lng outlayer in the response there isn't the end_station_name and end_station_id

Outlayer count end station	# latitude	# longitude
	16	42

Searching for data where end date minor than start date

Using the following query we notice the there is an error in interval calculated, because a negative value results as minimum

```
SELECT min(DATETIME_DIFF( ended_at, started_at, MINUTE )) as min_length, max(DATETIME_DIFF( ended_at, started_at, MINUTE )) as
max_length  FROM `cyclistic_data.bike_trip_data`
```

min_length	max_length
-16656	1559

So in the dataset are present rows where the value of started_at are greater than end_at value.

To understand how much rows have this error , I execute the following query

```
SELECT count(*) as unvalid,
    (select count(*) FROM`cyclistic_data.bike_trip_data`) as total
FROM `cyclistic_data.bike_trip_data`
WHERE DATETIME_DIFF( ended_at, started_at, MINUTE ) <0

SELECT count(*) as unvalid,
    (select count(*) from`cyclistic_data.bike_trip_data`) as total
FROM `cyclistic_data.bike_trip_data`
WHERE DATETIME_DIFF( ended_at, started_at, MINUTE ) <1
```

Executing the query results 116 rows where the distance haa a negative value and 135832 rows where the time is under 1 minute. Respects 5854544 rows of all dataset, the rows with this error are 2% of the dataset.

Resume and conclusion

Dataset is composed by 5854544 rows.

field	format value	# NULL value	# Outlayer
ride_id	generic string no rule	0	0
rideable_type	["elettric_bike", "classik_bike, elettric_scooter"]	0	0
started_at	[format AAAA-MM-DD hh:mm:ss] in the correct interval 2023-10-01 - 2024-09-31	0	0
ended_at	[format AAAA-MM-DD hh:mm:ss] in the correct interval 2023-10-01 - 2024-09-31	0	135832
start_station_name		1056535	0
start_station_id	generic string no rule	1056535	0
end_station_name		1091792	0
end_station_id	generic string no rule	1091792	0
start_lat	between -90 and 90	1091792	0
start_lng	between -180 and 180	1091792	0
end_lat	between -90 and 90	1091792	16
end_lng	between -180 and 180	1091792	42
member_casual	["member", "casual"]	0	0

The dataset has 20% of null values on the information relating to the departure and arrival stations (name, id, longitude and latitude), furthermore it has few outlayers on the latitude and longitude values of the arrival stations. Since these columns are not considered useful to achieve the objective, they will be eliminated from the dataset that will be used in the analysis phase and therefore no interventions will be necessary to fill the null value and solve the outlayers' problem.

There are 135832 rows where the end date is wrong because minor the start or where the trip don't start if the difference is zero.

Due to the number being so low, we will remove these rows The other fields are complete and don't show any imputation errors.

Manipulation of data

We work on a copy of a original dataset called 'bike_trip_data' and all operations are recorded in a log file.

Columns deletion

In this table the columns of "start_lat", "end_lat","start_lng", "end_lng", "start_station_id", "end_station_id", "start_station_name" and "end_station_name" will be dropped.

```
ALTER TABLE
`graphite-space-440117-r9.cyclistic_data.bike_trip_data`
DROP COLUMN end_lat;

ALTER TABLE
`graphite-space-440117-r9.cyclistic_data.bike_trip_data`
DROP COLUMN end_lng;

ALTER TABLE
`graphite-space-440117-r9.cyclistic_data.bike_trip_data`
DROP COLUMN start_lat;

ALTER TABLE
`graphite-space-440117-r9.cyclistic_data.bike_trip_data`
DROP COLUMN start_lng;

ALTER TABLE
`graphite-space-440117-r9.cyclistic_data.bike_trip_data`
DROP COLUMN start_station_name;

ALTER TABLE
`graphite-space-440117-r9.cyclistic_data.bike_trip_data`
DROP COLUMN start_station_id;

ALTER TABLE
`graphite-space-440117-r9.cyclistic_data.bike_trip_data`
DROP COLUMN end_station_name;

ALTER TABLE
`graphite-space-440117-r9.cyclistic_data.bike_trip_data`
DROP COLUMN end_station_id;
```

Rows deletion

we delete all rows where the time of use is not useful for analysis they are 135832

```
DELETE FROM `cyclistic_data.bike_trip_data` WHERE DATETIME_DIFF( ended_at, started_at, MINUTE ) <1;
```

Operations to add columns and values

For help the analysis phase have been added this columns:

name	type	description
month	int64	Month of started trip
ride_length	int64	time interval between end date and start date
day_of_week	int64	day of week (1 = Sunday, 2 = Monday,... 7 = Saturday) of started trip

```
alter table `cyclistic_data.bike_trip_data` ADD COLUMN month INT64;
alter table `cyclistic_data.bike_trip_data` ADD COLUMN ride_length INT64;
alter table `cyclistic_data.bike_trip_data` ADD COLUMN day_of_week INT64;

alter table `cyclistic_data.bike_trip_data` ADD COLUMN season INT64;
alter table `cyclistic_data.bike_trip_data` ADD COLUMN season_name STRING;
```

We use the following operations to calculate values of the new columns

```
UPDATE  `cyclistic_data.bike_trip_data`  SET month = EXTRACT(MONTH from started_at)  WHERE 1=1 ;
UPDATE  `cyclistic_data.bike_trip_data`  SET day_of_week = EXTRACT(DAYOFWEEK from started_at)  WHERE 1=1 ;
UPDATE  `cyclistic_data.bike_trip_data`  SET ride_length = DATETIME_DIFF( ended_at, started_at, MINUTE )  WHERE 1=1 ;
```

Using following periods to define season of chicago the information

season	start month	end month
WINTER	december	february
SPRING	march	may
SUMMER	june	august

season	start month	end month
AUTUMN	september	november

```
UPDATE  `cyclistic_data.bike_trip_data`  SET season = 4, season_name="Autumn" WHERE month >8 AND month < 12 ;
UPDATE  `cyclistic_data.bike_trip_data`  SET season = 1, season_name="Winter" WHERE month <3 OR month = 12 ;
UPDATE  `cyclistic_data.bike_trip_data`  SET season = 2, season_name="Spring" WHERE month >2 AND month < 6 ;
UPDATE  `cyclistic_data.bike_trip_data`  SET season = 3, season_name="Summer" WHERE month >5 AND month < 9 ;
```

Validation

Run the following query to check:

- the number of data in the dataset
- if there are also negative elements in the ride_length column
- if all months and ride_length are filled in correctly

```
SELECT count(*) from tripdata.triptrack group by month
select * from tripdata.triptrack  order by ride_length DESC limit 100;
```

```
select ride_id,rideable_type, start_station_id, end_station_id, member_casual, month, day_of_week,
ride_length from tripdata.triptrack  order by ride_length DESC limit 100;

select rideable_type,count(rideable_type) as count_rideable_type from `tripdata.triptrack`
group by rideable_type;

select month,count(month) as count_rideable_type from `tripdata.triptrack`
group by month order by month ASC;
```

Selection of the most significant columns.

Name	type	description
ride_id	[string]	unique identifier of the ride on bike
rideable_type	[category]	electric_bike, electric_scooter, classic_bike
member_casual	[category]	Member, Casual
month	[1-12]	from the Started_at
dayofweek	[1-7]	from the Started_at
ride_length	[integer]	by difference of End_at end Started_at in minutes

4. Analysis Phase

Calculate Maximum, Minimum and average of ride_length

```
select
    CAST(avg(ride_length) AS INTEGER) as avg_ride_lenght,
    max(ride_length) as  max_ride_lenght,
    min(ride_length) as  min_ride_lenght,
    count(ride_id) as count_ride
from
    `cyclistic_data.bike_trip_data`
```

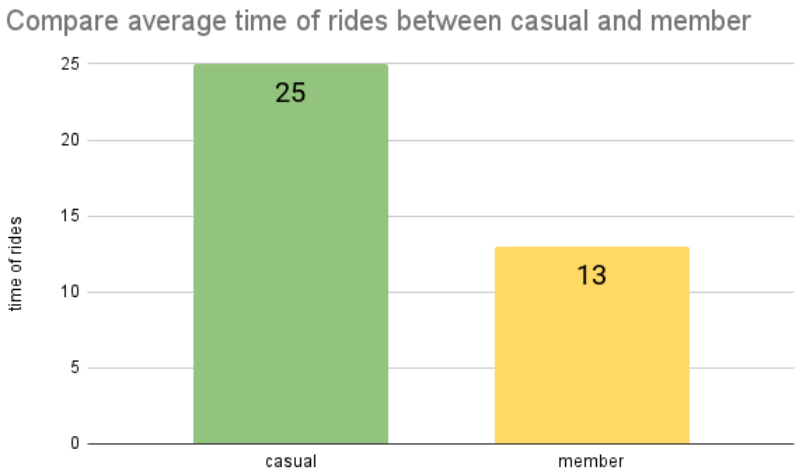
avg_ride_lenght	max_ride_lenght	min_ride_lenght	count_ride
17	1559	1	5718712

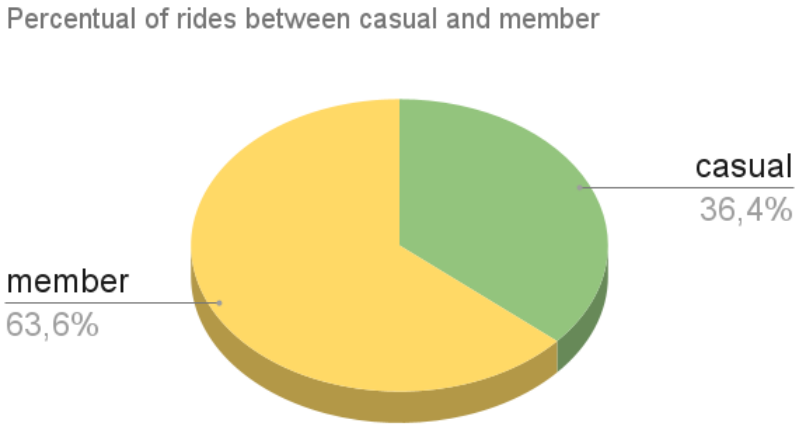
Calculate Maximum and average of ride_lenght by member_casual

```
select
    member_casual,
    CAST(avg(ride_length) AS INTEGER) as avg_ride_length,
    max(ride_length) as  max_ride_length,
    min(ride_length) as  min_ride_length,
    count(ride_id) as count_ride,
    count(ride_id)*100/5718712 as count_perc
from
    `cyclistic_data.bike_trip_data`
group by
    member_casual
order by
    avg_ride_lenght desc
```

the union of the results is the following:

member_casual	avg_ride_length	max_ride_length	min_ride_length	count_ride	count_perc
casual	25	1559	1	2067119	36.13%
member	13	1559	1	3651593	63.87%





> - the maximum and minimum distances are the same so it isn't useful for analysis
> - Count of rides of members is double then casuals
> - Average time of rides of casual greater than member sometime is the double

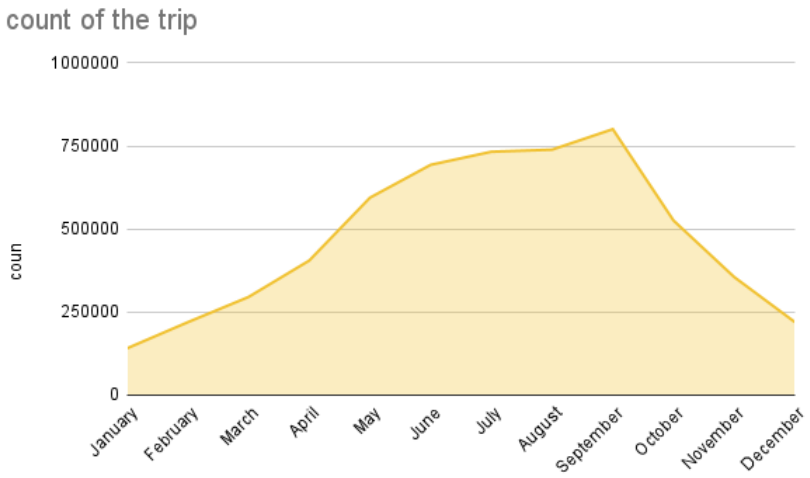
Calculate Maximum and average of ride_length by month

```
select
  month,
  FORMAT_DATETIME("%B",  started_at) as month_name,
  CAST(avg(ride_length) AS INTEGER) as avg_ride_lenght,
  count(ride_id) as count_trip
from
    `cyclistic_data.bike_trip_data`
group by month_name, month order by month ASC
```

month	month_name	avg_ride_lenght	count_trip
1	January	15	140537
2	February	15	219183
3	March	15	295099
4	April	17	404830
5	May	19	594130
6	June	20	693149
7	July	20	732543
8	August	19	738864
9	September	16	800533
10	October	16	525420
11	November	14	355114
12	December	13	219310

Average of rides lenght for Month

> The average monthly distance of all types of customers increases slightly during the summer season, but every month it is around 16 minutes



Count of rides for Month

> The number of trips is very different monthly. In winter there are around 100,000 races, but in summer the number increases six times.

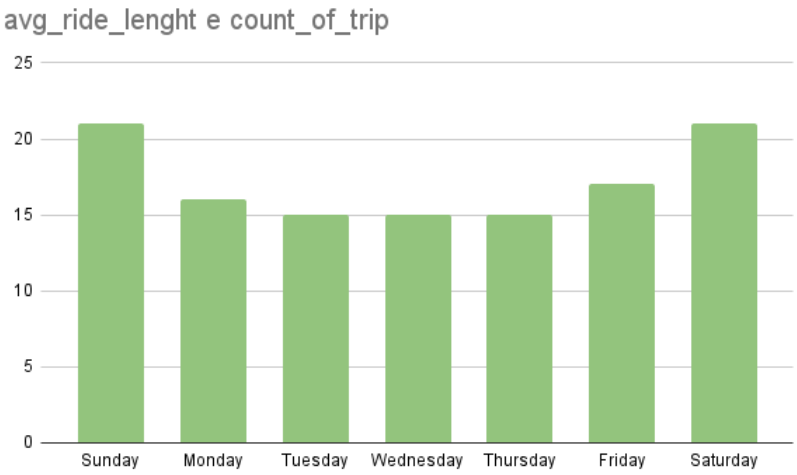
> Comparing the 2 graphs shows that in summer the number of races increases but the distances are short, and vice versa going towards winter the number of races decreases but the duration increases.

calculate avg of ride_length and the MODE respect day of week for Total dataset

```
select
    FORMAT_DATETIME("%A",  started_at) as day_name,day_of_week
    , CAST(avg(ride_length) AS INTEGER) as avg_ride_lenght, max(ride_length) as  max_ride_lenght, count(ride_id) as count_of_trip
from
    `cyclistic_data.bike_trip_data`
group by day_of_week,day_name order by day_of_week ASC
```

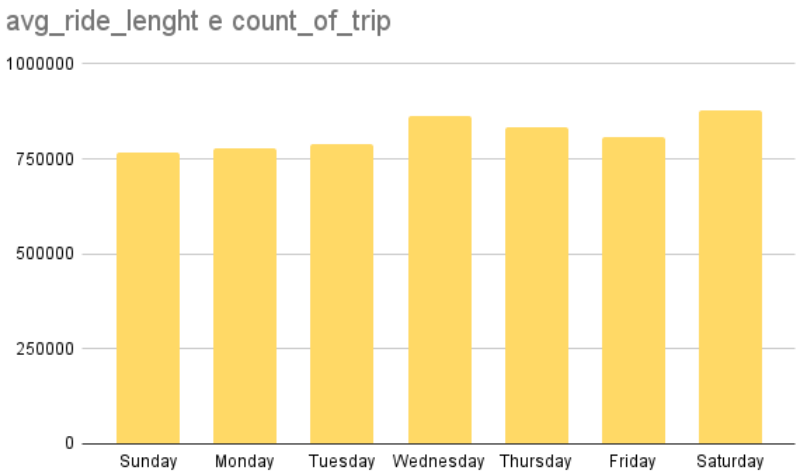
results are:

day_name	day_of_week	avg_ride_lenght	max_ride_lenght count_of_trip	
Sunday	1	21	1500	768568
Monday	2	16	1500	777879
Tuesday	3	15	1499	788371
Wednesday	4	15	1500	864628
Thursday	5	15	1499	833148
Friday	6	17	1500	806926
Saturday	7	21	1559	879192



AVG Ride_length

> The average duration of races increases slightly over the weekend, but there are no big differences



Count trip

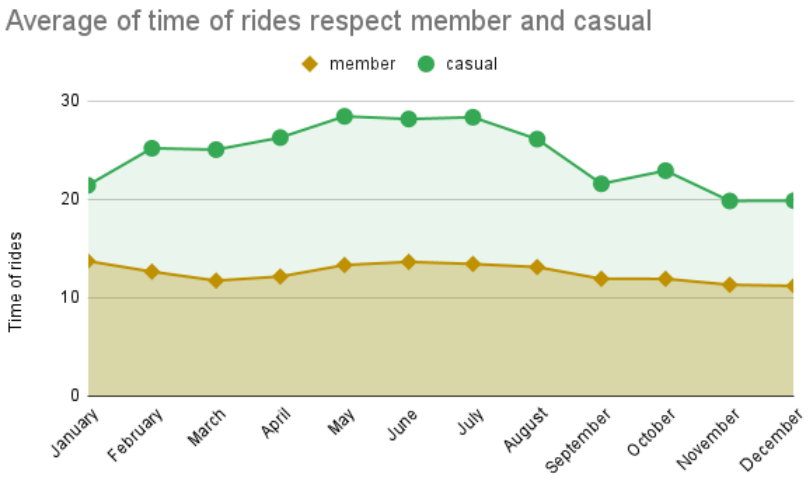
> the number of weekly trips is almost identical, compared to the number analyzing the entire year there are no significant differences

> analyzing both the two graphs over the entire year we do not notice a significant element where the days of the weeks can be useful for some advice.

Pivot AvgLenght vs Member_casual for each month

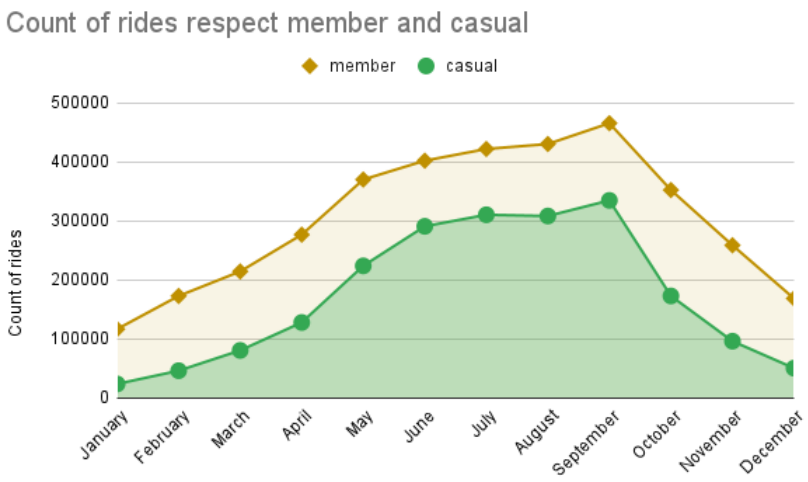
```
SELECT * FROM
(
select
    month,
    FORMAT_DATETIME("%B",  started_at) as month_name,
    member_casual,
    ride_id,
    ride_length

from
    `cyclistic_data.bike_trip_data`
)
PIVOT
(
  -- #2 aggregate
  avg(ride_length) as avg_ride_length,
  count(ride_id) as count_ride
  -- #3 pivot_column
  FOR member_casual in ("member","casual")
) order by month
```

month	month_name	avg_ride_length_member	count_ride_member	avg_ride_length_casual	count_ride_casual
1	January	13.720059264841895	116764	21.42115004416776	23773
2	February	12.64393597705536	172938	25.189706995350853	46245
3	March	11.730490204306188	214482	25.051490380440839	80617
4	April	12.148011545283444	276823	26.276719241916435	128007
5	May	13.318053998568308	370195	28.436711545761032	223935
6	June	13.636995395279905	402196	28.165212250775923	290953
7	July	13.424993187768791	422035	28.342632073891817	310508
8	August	13.108227220243613	430437	26.120352628012455	308427
9	September	11.915971457223421	465687	21.579418598400476	334846
10	October	11.913057047446381	352461	22.913447695696686	172959
11	November	11.308315176239709	258852	19.831771623278119	96262
12	December	11.196072853137991	168723	19.860418684642305	50587



Member	Casual	
mean	12,5	24,4
variance	0,8	10,4

> - The average travel time of casuals is longer than members, especially in the spring and summer months. Casuals have higher variance than members.
> - the Member variance is almost 0 to indicate that they always spend the same amount of time each month.



Calculated mean and variance starting from previous data

> The count's trend of rides in the year is the same for both members and casuals. The count of rides for both increase during spring and summer months

> - THe COUNT of trips by MEMBERs is greater than that of CASUALs on all months of year
> - THe AVERAGE DISTANCE by trips of CASUALs is greater than that of MEMBERs on all months of year

Ordening the results:

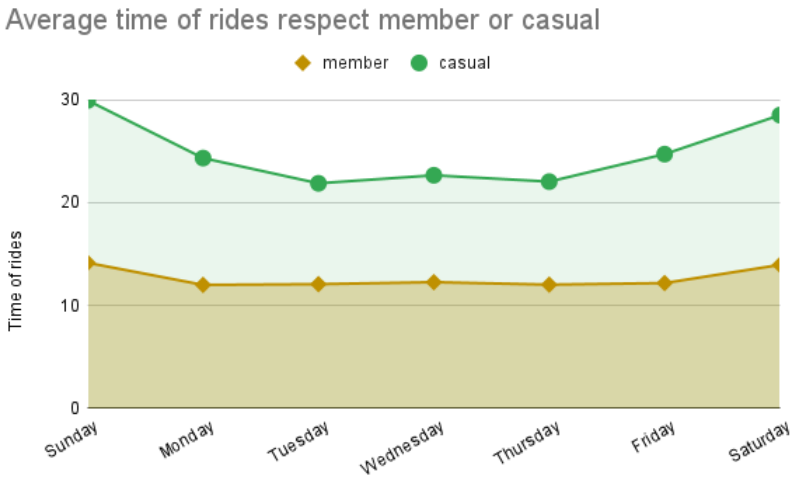
> - The average length used by MEMBER has max: 13 minutes, min: 11 minutes,average: 12 minutes
> - The average length used by CASUAL has max: 28 minutes, min: 19 minutes,average: 25 minutes

PIVOT of count and distance respect day of week and member_casual value

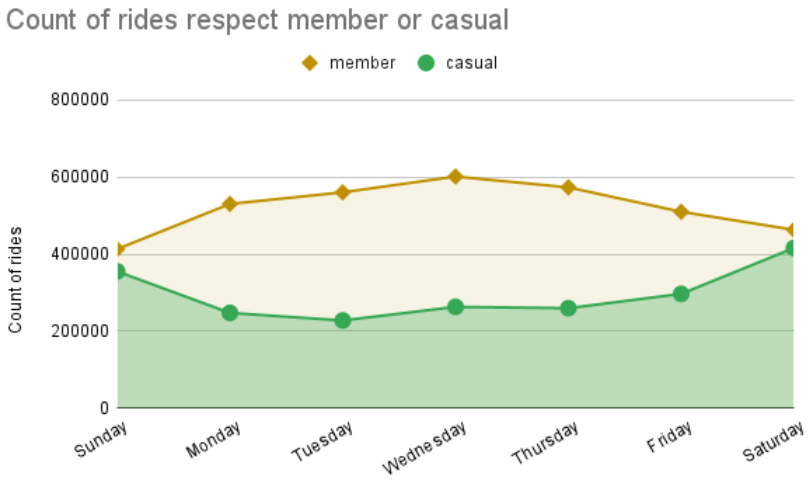
```
SELECT * FROM
(
select
    day_of_week,
    FORMAT_DATETIME("%A",  started_at) as day_name,
    member_casual,
    ride_length,
    ride_id

from
    `cyclistic_data.bike_trip_data`
)
PIVOT
(
  -- #2 aggregate
  avg(ride_length) as avg_ride_length, count(ride_id) as count_ride
  -- #3 pivot_column
  FOR member_casual in ("member","casual")
) order by day_of_week
```

day_of_week	day_name	avg_ride_length_member	count_ride_member	avg_ride_length_casual	count_ride_casual
1	Sunday	14.160626902803928	413206	29.955144331695539	355362

day_of_week	day_name	avg_ride_length_member	count_ride_member	avg_ride_length_casual	count_ride_casual
2	Monday	12.011196056393267	530276	24.3477461904743	247603
3	Tuesday	12.083290951756382	560385	21.890888036984727	227986
4	Wednesday	12.275715155203891	601338	22.67404003190396	263290
5	Thursday	12.036023024594471	573300	22.056282903851443	259848
6	Friday	12.189292269318853	509968	24.729793438802822	296958
7	Saturday	13.95906028675	463120	28.529511719125519	416072



> - **member**: the trend increase in work day
> - **casual**: the trend became greater in weekend

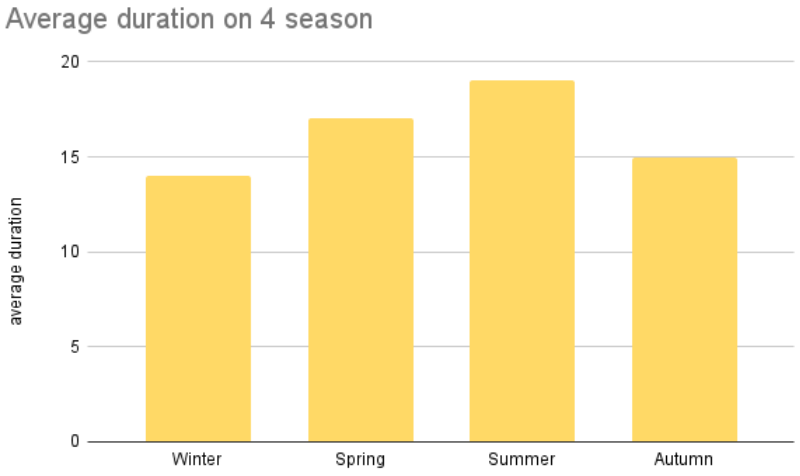


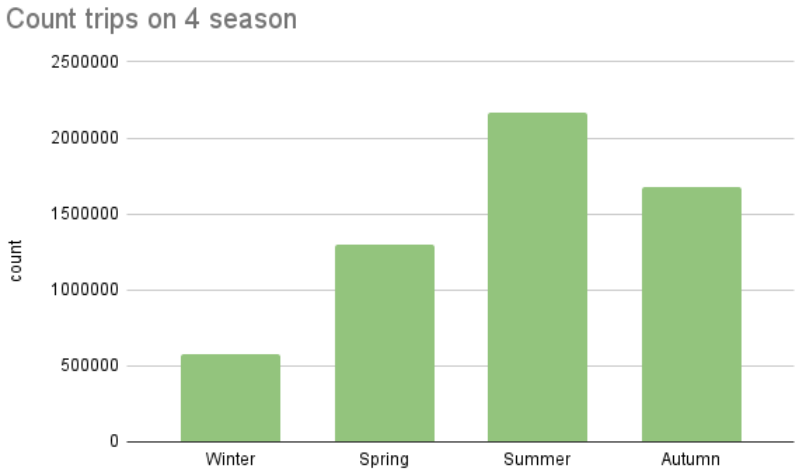
> - **member**: the trend is lower and constant
> - **casual**: the trend is greater during the weekend (max 30 sundsy and saturdsy, min 22 tuesday, wednesday, thursday)

Analysis vs Season

```
select  season, season_name, count(ride_id) as count_ride, avg(ride_length) as avg_ride_length from `cyclistic_data.bike_trip_data`
group by season, season_name
```

season	season_name	count_ride	avg_ride_length
1	Winter	579030	14.431865361034822
2	Spring	1294059	17.433718246231443
3	Summer	2164556	19.331639837453917
4	Autumn	1681067	15.331391312779319

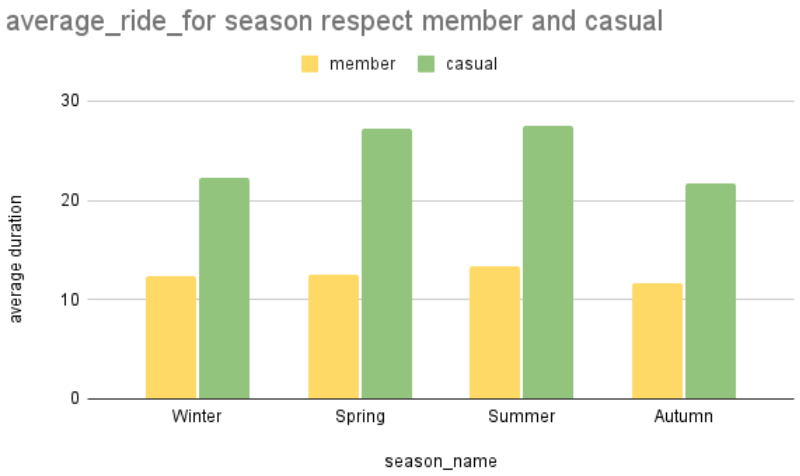


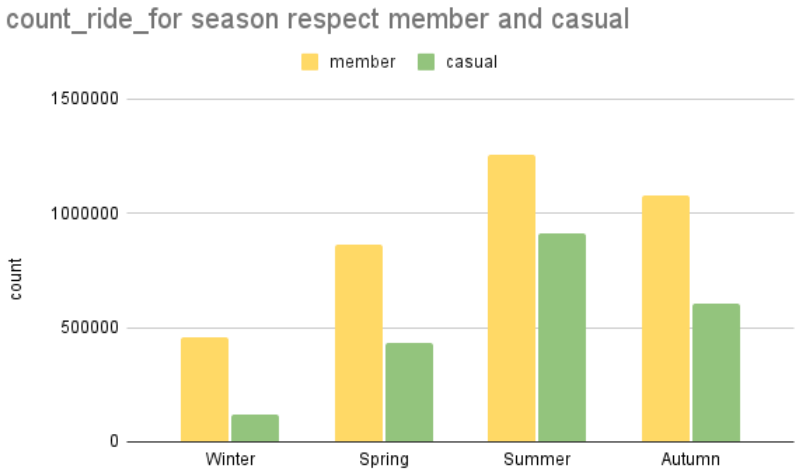



```
select * from(
select  season, season_name,member_casual,ride_length,ride_id from `cyclistic_data.bike_trip_data`
)
PIVOT
(
  -- #2 aggregate
  avg(ride_length) as avg_ride_length, count(ride_id) as count_ride
  -- #3 pivot_column
  FOR member_casual in ("member","casual")
) order by season
```

season	season_name	avg_ride_length_member	count_ride_member	avg_ride_length_casual	count_ride_casual
1	Winter	12.385146970605868	458425	22.211533518510837	120605
2	Spring	12.546842716192673	861500	27.166594614838651	432559
3	Summer	13.384280144229406	1254668	27.532607309910603	909888

season	season_name	avg_ride_length_member	count_ride_member	avg_ride_length_casual	count_ride_casual
4	Autumn	11.768970287836614	1077000	21.682884514466107	604067

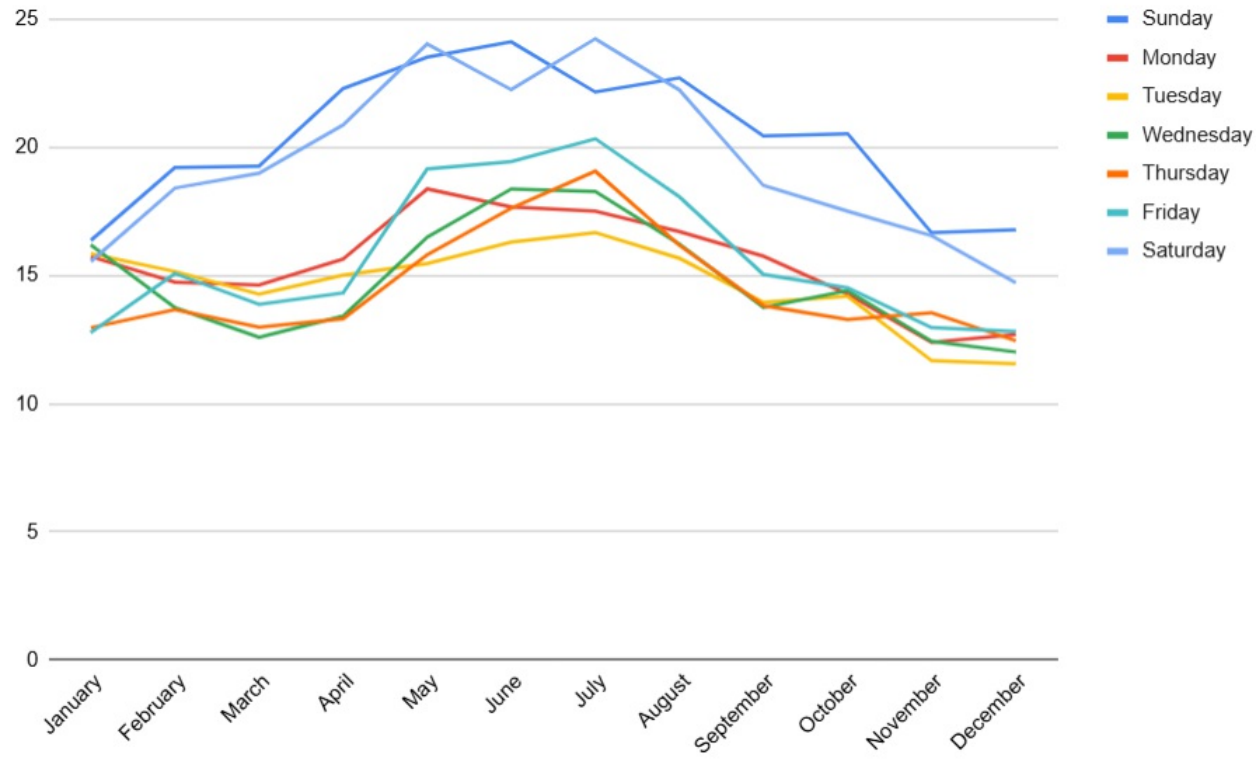




Analyze the behavior of cyclists on the day of the week for each month

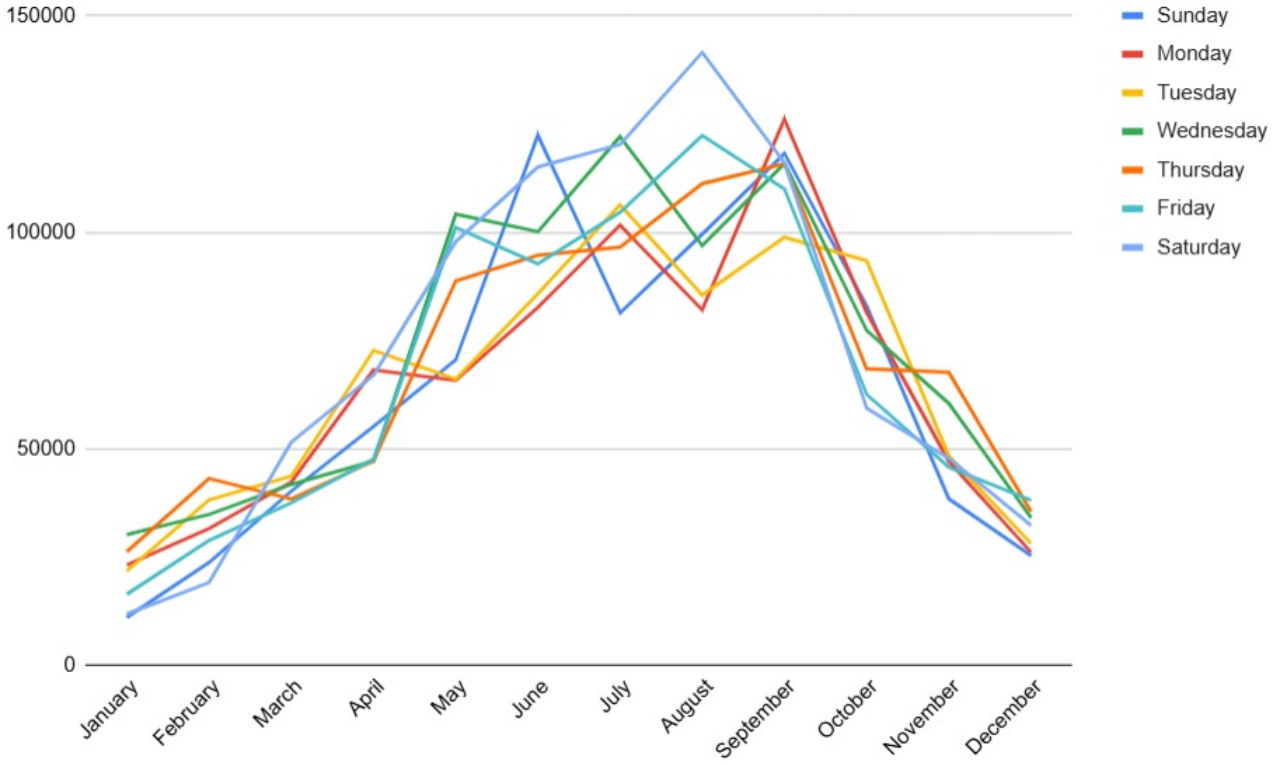
```
SELECT * FROM
(
  -- #1 from_item
  SELECT
    month,
    FORMAT_DATETIME("%B",  started_at) as month_name,
    FORMAT_DATETIME("%A",  started_at) as day_name,
    ride_length
  FROM `cyclistic_data.bike_trip_data`
)
PIVOT
(
  -- #2 aggregate
  count(ride_length) AS count_ride, avg(ride_length) AS avg_ride_length
  -- #3 pivot_column
  FOR day_name in ("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday")
) ORDER BY month ASC
```

Average of length Ride for weekday and month



- the trend of the average race time is always the same for all day of the week
- On sunday and saturday the time is greater than on other days
- There is a slight increase in trip times for all days in the summer months

Count of Ride for weekday and month



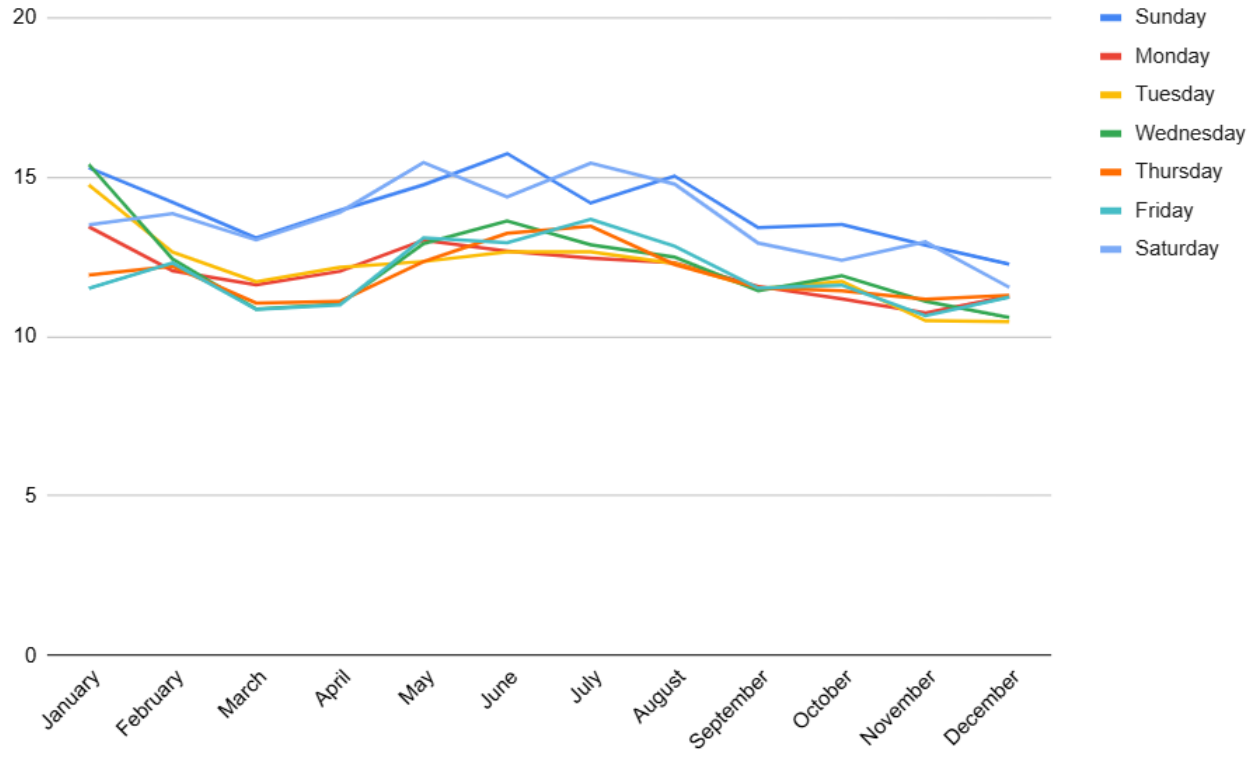
We notice a slight difference between summer/spring months and Fall/Winter months.

- During Fall/Winter months the service is used most frequently on workdays.
- During Spring/Summer months the use of service tends to become more frequent on weekends.
- the trend is similar throughout the day of each month
- there is a strong increase in the number of trips in the summer period, 6 times greater than in the winter months.

Calculate average of ride_length and the MODE respect day of week for MEMBER o CASUAL kind

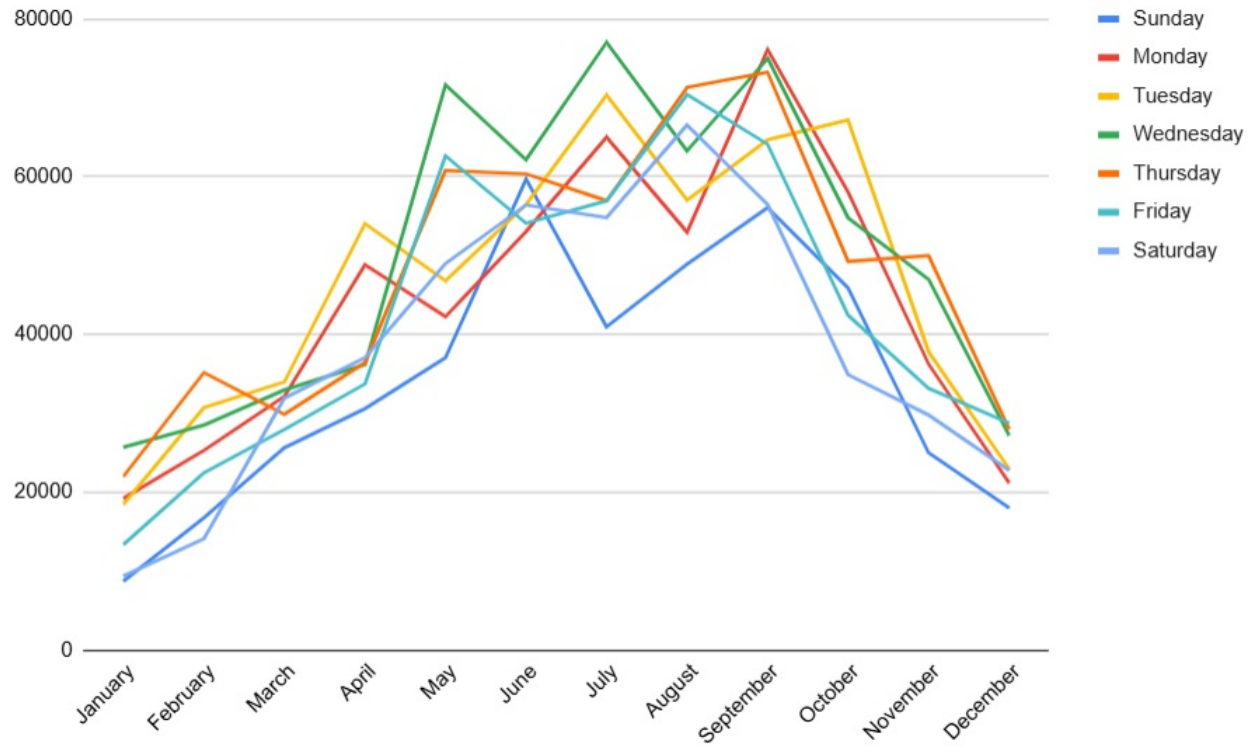
```
SELECT * FROM
(
  -- #1 from_item
  SELECT
    month,
    FORMAT_DATETIME("%B",  started_at) as month_name,
    FORMAT_DATETIME("%A",  started_at) as day_name,
    ride_length
  FROM `cyclistic_data.bike_trip_data`
  WHERE member_casual="casual"
  --- WHERE member_casual="casual"
)
PIVOT
(
  -- #2 aggregate
  avg(ride_length) AS avg_ride_length,
  count(ride_length) AS count_ride_length
  -- #3 pivot_column
  FOR day_name in ("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday")
) ORDER BY month ASC
```

Member length of rides for weekdays and months



- the average distance of trips made by members on weekends is higher than on weekdays
- Saturday is always constant throughout the month and has the highest value
- Sunday is similar to Saturday but in August the distance becomes black to zero.

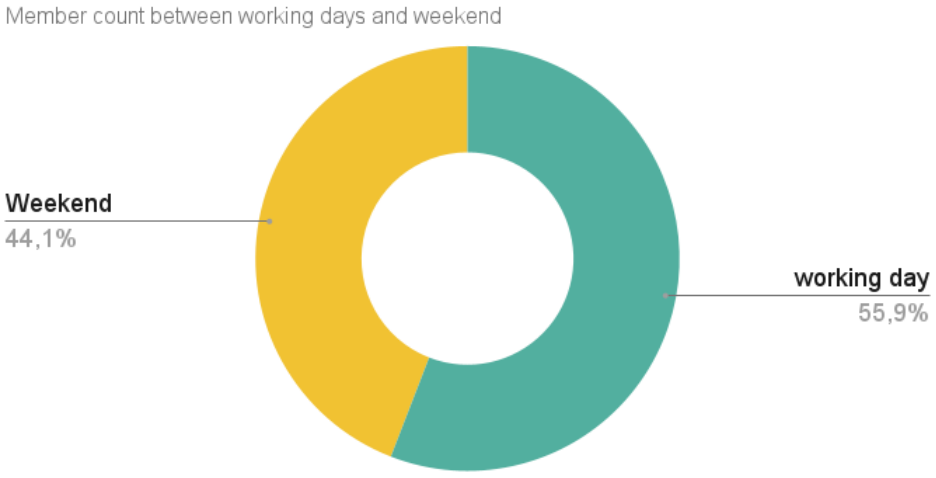
Member count of rides for weekdays and months

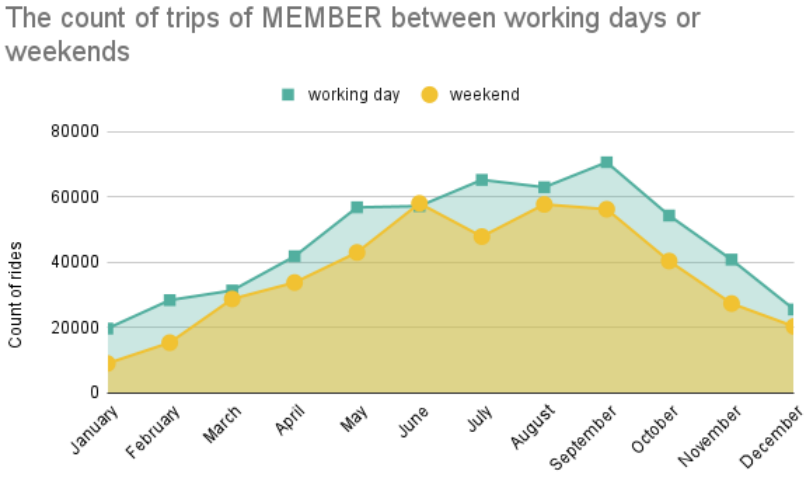


- The count of trips made by members has the same trend for every day in all months of the year
- Saturday has the lowest value, so members have decreased this month but they use the bicycle for a greater distance.

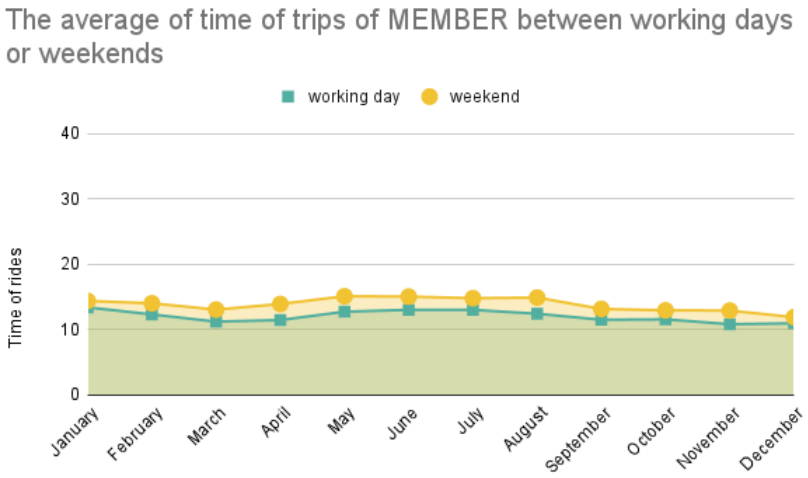
Calculated respect working day and weekend

month_name	COUNT WORKDAY	AVG WORKDAY	COUNT WEEKEND	AVG WEEKEND
January	19736,2	13,40947133	9041,5 14,40122708	
February	28425,8	12,33294523	15404,5 14,0367029	
March	31383,6	11,2251734	28782 13,06597484	
April	41836,6	11,47145423	33820 13,93211192	
May	56829	12,74971958	43025 15,11134556	
June	57217,2	13,03167231	58055 15,06097321	
July	65252,6	13,02877173	47886 14,81446659	
August	62994,2	12,44304317	57733 14,91031507	
September	70633,8	11,51564924	56259 13,17258379	
October	54332,8	11,57601238	40398,5 12,95802295	
November	40815,2	10,83697803	27388 12,92142822	
December	25596,4 10,9743638	20370,5 11,90997178		





> The service is used by Members more frequent in working day than weekend (55.9% - 44.1%), this trend is similar in every months

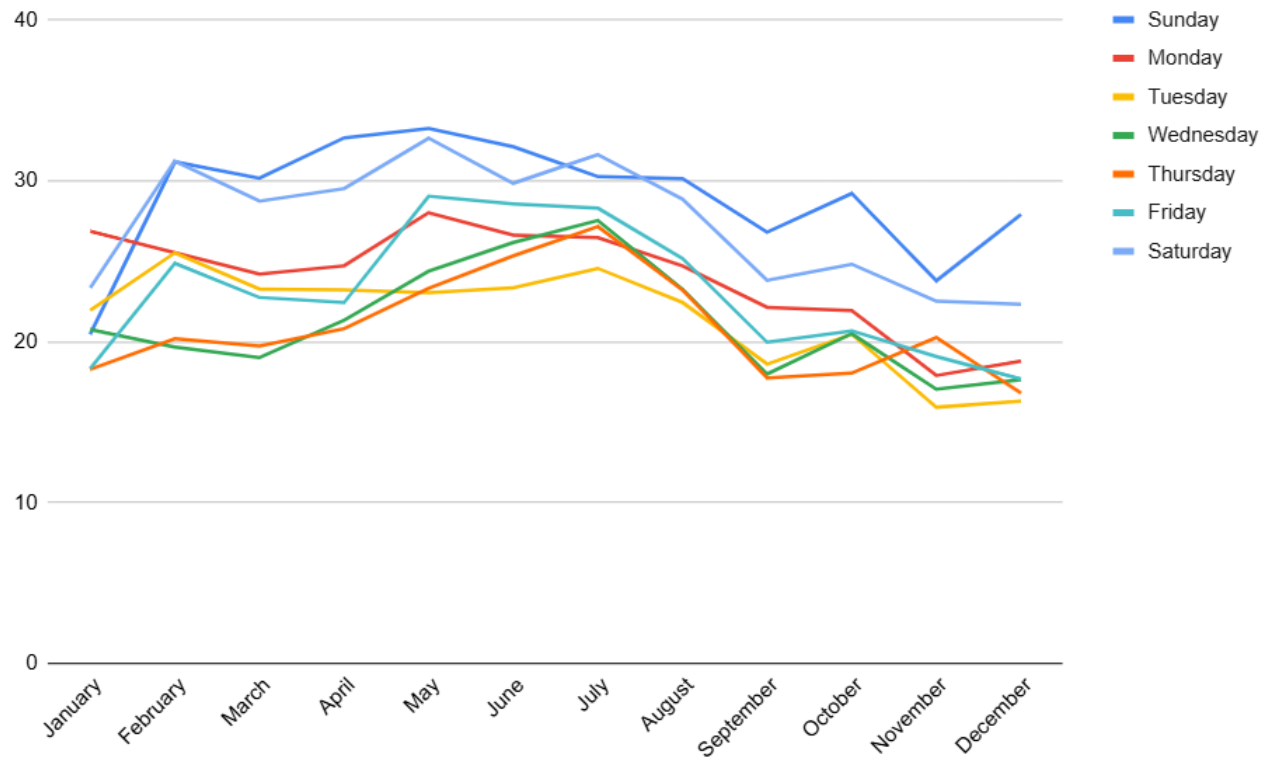


> member: trend in every months of year is similar in all days

Calculate avg of ride_length and the MODE respect day of week for CASUAL dataset

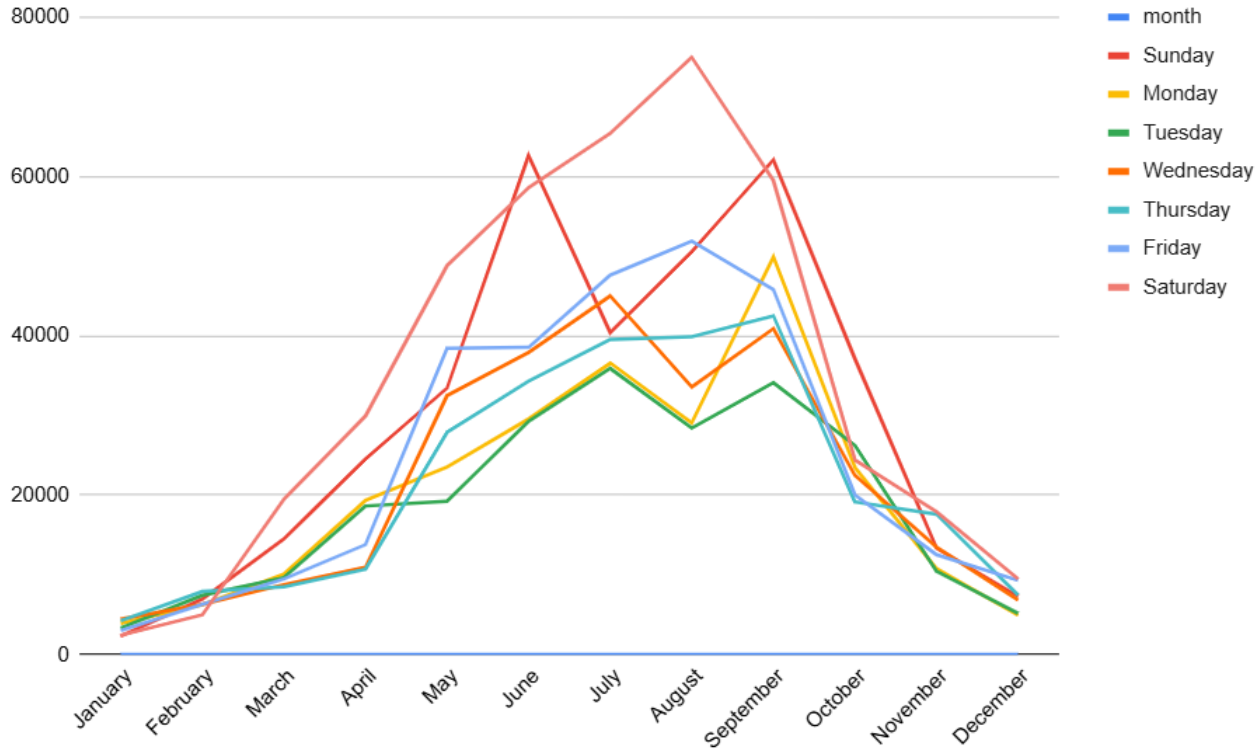
```
SELECT * FROM
(
  -- #1 from_item
  SELECT
    month,
    FORMAT_DATETIME("%B",  started_at) as month_name,
    FORMAT_DATETIME("%A",  started_at) as day_name,
    ride_length
  FROM `cyclistic_data.bike_trip_data`
  WHERE member_casual="casual"
)
PIVOT
(
  -- #2 aggregate
  avg(ride_length) AS avg_ride_length,
  count(ride_length) AS count_ride_length
  -- #3 pivot_column
  FOR day_name in ("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday")
) ORDER BY month ASC
```

Casual Average length of rides for weekdays and months



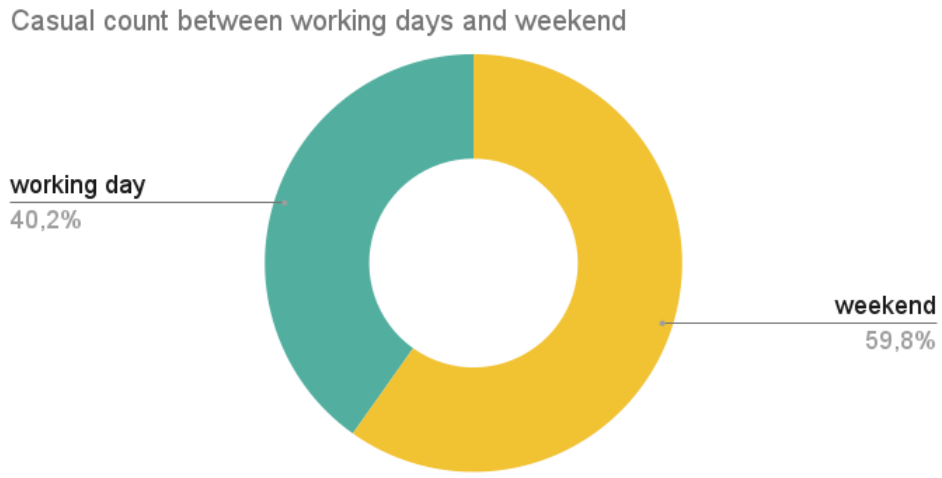
> - the averages distances on all days are similar,
> - there are many variability in between january and may

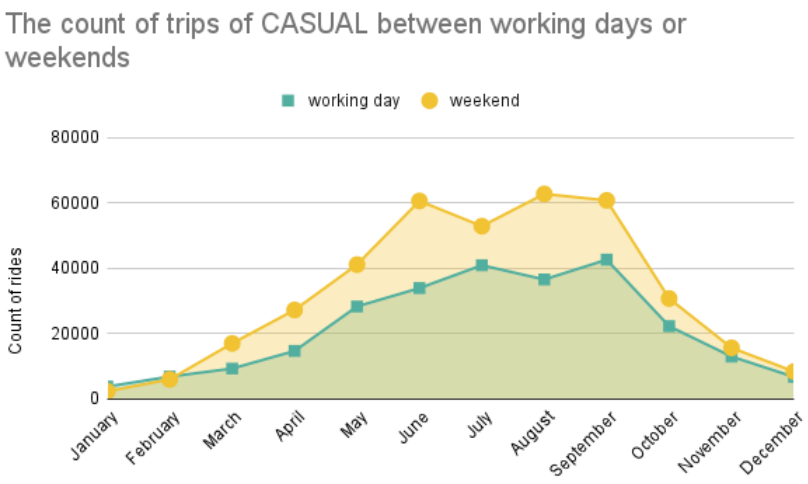
Casual Count of rides for weekdays and months



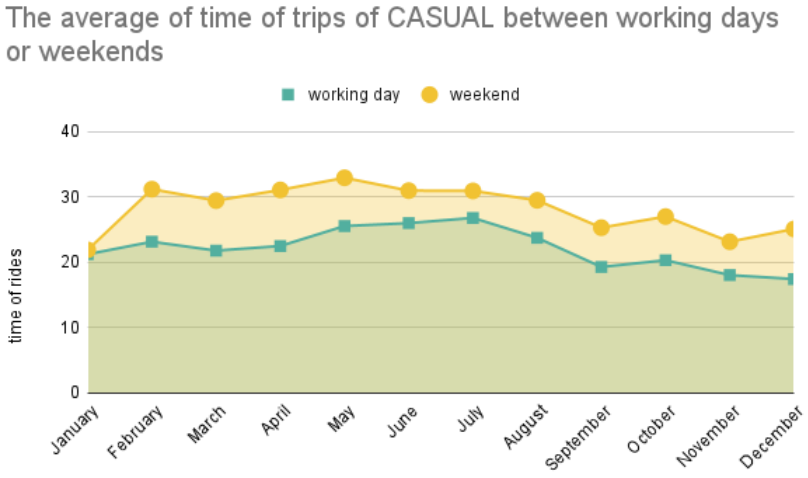
- the count increases for all days on the summer
- the tendencies are similars for all days on all months
- during december, janury, february the count si very low
- Th count is most on sunday and saturday, on other days the count has the same average for all months

Calculated respect working day and weekend





- the service is used by Casuals more frequent during the weekend (59.8% - 40.2%)
- that casuals in weekend increase the use during the spring and summer



casual: the service in every months of year is use smore in week end than working days

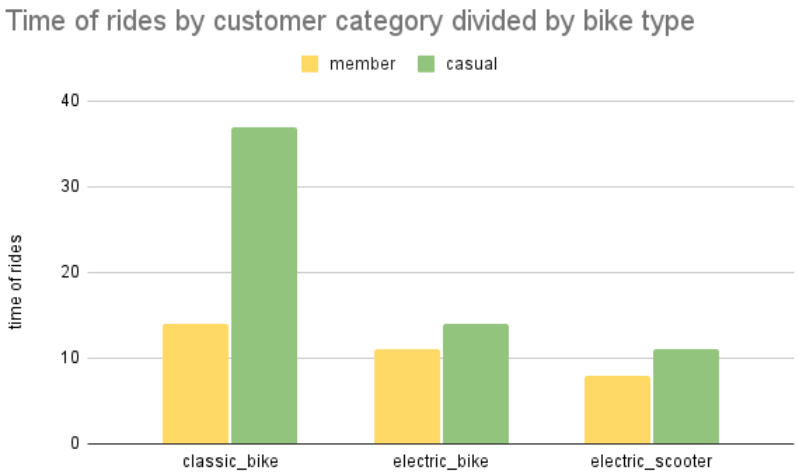
Analysis member-casual respect rideable_type

```
select * from(
select  rideable_type,member_casual,ride_length,ride_id from `cyclistic_data.bike_trip_data`
)
PIVOT
(
  -- #2 aggregate
  avg(ride_length) as avg_ride_length,
  count(ride_id) as count_ride
  -- #3 pivot_column
  FOR member_casual in ("member","casual")
) order by rideable_type
```

rideable_type	avg_ride_length_member	count_ride_member	avg_ride_length_casual	count_ride_casual
classic_bike	14.246873641650241	1799058	37.879924041505731	965264
electric_bike	11.058633345227463	1796401	14.73946325233608	1020405
electric_scooter	8.160526597071291	56134	11.972154696132591	81450

Starting by these data we calculate percentual of count

category	classic_bike	electric_bike	electric_scooter	total
member	49,27%	49,19%	1,54%	3651593
casual	46,70%	49,36%	3,94%	2067119



> All kind of customer use the same genre of bike in the same percentual

> casual when use classic_bike tends to do longer rides then member

Result of the case study - Summary of analysis

- Casuals rides are 28% less than members.
- The duration of the casuals' races is longer than that of the members (25 minutes casuals vs 13 members)
- Casuals increase their use of the service in the spring and summer more than members
- Casuals prefer to use the service on weekends (average 30 minutes)
- The rides count of members always has the same trend throughout the months on both working days and weekends
- Casuals users prefer the classic bikes for traveling for a long periods, more than members (45 minutes versus 12 minutes)
- Members use the service with a constant duration (low variance 0,8) as if the use depended on a scheduled event, vice versa the duration of the
- Casuals has a high variance to indicate that the use does not follow a scheduled need.

Insights

- Casuals use the service for pleasure, in fact they prefer spring and summer weekends and use the service for a long time
- Members use the service to cycle for usual services such as going to work or other repetitive activity, to reach easily a destination.

Reccomendation

The result of this study : To evaluate subscription strategies that favor those who use the service for:

1. longer periods
2. spring and summer months
3. using classic bike especially on weekends

Advices to improve strategy

Connecting the trips for the same user to the dataset would allow us to better understand the service's habits of use. Thus, we could identify homogeneous groups to which specific strategies can be dedicated through clustering.

LOG FILE - TABLE bike_trip_data

DATE	OPERATION	ELEMENT	DESCRIPTION
2024.10.28	COPY	all data	copied all data from 'cyclistic_data.bike_trip' to 'cyclistic_data.bike_trip_data'
2024.10.28	DEL COLUMN	start_lan	unuseful for analysis and with null elements
2024.10.28	DEL COLUMN	start_lnge	unuseful for analysis and with null elements
2024.10.28	DEL COLUMN	end_lan	unuseful for analysis and with null elements
2024.10.28	DEL COLUMN	end_lng	unuseful for analysis and with null elements
2024.11.03	DEL COLUMN	start_station_name	unuseful for analysis and with null elements
2024.11.03	DEL COLUMN	end_station_name	unuseful for analysis and with null elements
2024.11.03	DEL COLUMN	start_station_id	unuseful for analysis and with null elements
2024.11.03	DEL COLUMN	end_station_id	unuseful for analysis and with null elements
2024.11.04	ADD COLUMN	month	useful for analysis
2024.11.04	ADD COLUMN	day_of_week	useful for analysis
2024.11.04	ADD COLUMN	ride_length	useful for analysis
2024.11.04	UPDATE VALUE	month	calculated from star_data_trip
2024.11.04	UPDATE VALUE	day_of_week	calculated from star_data_trip
2024.11.04	UPDATE VALUE	ride_length	calculated like time interval between end dates and start dates of the trips
2024.11.05	DEL ROW	some rows	deleted rows where interval between end_at - start_at is less the one minute.

1. A clear statement of the business task (ASK)

About Cyclistic

Cyclistic, a Chicago-based bike-sharing company that offers both one-time and annual memberships, sets itself apart by also offering reclining bikes, hand tricycles, and cargo bikes, making bike-share more inclusive to people with disabilities and riders who can’t use a standard two-wheeled bike. The majority of riders opt for traditional bikes; about 8% of riders use the assistive options. Cyclistic users are more likely to ride for leisure, but about 30% use them to commute to work each day.

The request

The company aims to design marketing strategies aimed at converting one-time riders into annual members.

The request for analytics to support marketing strategies will assess: **How do annual members and occasional cyclists use Cyclistic bikes differently??**

The Stakeholders

- **Lily Moreno**: The director of marketing is the sponsor of aim of the initiatives to converting one-time riders into annual members and this analysis.
- **Cyclistic marketing analytics team**: A team of data analysts who are responsible for collecting, analyzing, and reporting data that helps guide Cyclistic marketing strategy and so they are interested that the analysis was well done.
- **Cyclistic executive team**: They expect a good report detail-oriented to decide whether to approve the recommended marketing program
- **Occasional Cyclists**: This users could receive benefits becoming annual members

2. A description of all data sources used (Preparation)

The data source is a public folder of type **second part** published by the company *Motivate International Inc* under an open source license. The data is present on source divvy-tripdata and anonymous.

The folder contains data for a travel period divided by month of use, according to a format **Long data**, but there is no metadata file that explains the characteristics of the data.

Since the analysis is done only on the last 12 months and a manual operation of downloading and loading the 12 files on the systems was chosen without using automation tools such as API/ETL.

A folder was created on the cloud to make it accessible to a work team **/projects/2024/capstone/bike_share/dataset/csv** on which 12 ZIP files relating to the period have been uploaded 10/2023 - 9/2024.

The files are CSV compressed in ZIP divided by month of extraction.

To distinguish the versions of the files with respect to the sampling period of interest, the following nomenclature is used **AAAAMM-tripdata_AAAAMMDD.csv**. The first date indicates the reference period of the data and is static. The second date indicates the insertion/modification date to historicize the different versions of the file. Thanks this format is more easy find the last version of a same file or download the last version of all file if there are several update dates.

The files that make up the datasets have the following characteristics:

file name	dimension	MD5	righe
202310-tripdata-20241029.csv	105.3 Mb	4e71b4e49567fd13d0e01f72cbe5b2a7	537113
202311-tripdata-20241029.csv	71.3 Mb	9cf7cf1704779fbc1cd712de586523ad	362518
202312-tripdata-20241029.csv	43.9 Mb	0e2a48fcf487c75624672325aa7afc9b	224073
202401-tripdata-20241029.csv	28.7 Mb	7e4e37507051bf3b2e338a3126187c68	144873
202402-tripdata-20241029.csv	44.7 Mb	1e4d9a04cbbb14f9f5a36bf7c2c1c319	223164
202403-tripdata-20241029.csv	59.3 Mb	88267643679a17a3b218641832d6af0b	301687
202404-tripdata-20241029.csv	80.1 Mb	a89b2fd37bbd4d6113bb460fb2eb6984	415025
202405-tripdata-20241029.csv	117.7 Mb	875acd3626fb0d49ffe1dd1381d85aab	609493
202406-tripdata-20241029.csv	141.3 Mb	afe1ecf091a3eec10ce70166a7ada0ad	710721
202407-tripdata-20241029.csv	149.9 Mb	d0fbd3e9d186fd736bbb715865c1998f	748962
202408-tripdata-20241029.csv	150.9 Mb	d189e2cab64aea2c86e5a0d590d12800	755639
202409-tripdata-20241029.csv	161.0 Mb	95f1502c287f569c8ef5776bf711f583	821276

- Total rows: **5854544**
- Total size 1.1 Gb
- All files have a homogeneous structure with the same number of columns and the same labels

To use data of this large size it was not possible to open them with Google Sheets, even performing simple processing on individual files the system would crash, so we opted to use:

- **Microsfot Excel** using the function **Power Query** that It allows to handle big data like Tibble in R
- **BigQuery**.

Bigquery does not allow you to create tables directly from CSV files larger than 100Mb. A Google Bucket has been created to manage large files. The name the bucket is **ev-divvy-tripdata**, where the 12 files have been uploaded. Thanks to the presence of homogeneous files in buckets it was possible to load all the data present on multiple sources into a single table with a single operation, putting as input source *gs://ev-divvy-tripdata/*.

The following structure was created on Bigquery:

- dataset: **tripdata**
- table: **triptrack**

To verify that the data saved in the Cloud are correct, by downloading the files from buckets or Cloud, the hash was recalculated and compared with that in the table.

To verify further that the data was correctly imported and its integrity, we counted the data in the table that results 5854544 as the sum of the rows in the CSV files (Excluded header rows).

License, privacy, security and accessibility

The data provided is made available in open source mode under the following license.

This license allows to access, reproduce, analyze, copy, modify, distribute in your product or service and use the Data for any lawful purpose.

The data doesn't contain personal information.

The original data are stored online and there isn't required any authorization. the Local copy of data is stored in cloud and requires authorization to access, the license is also store together with the data, so if someone would share this data must export it following license indications.

ROCCC Analysis

- **Reliable**: the dataset isn't reliable, because there isn't information in the site store if the dataset covers all trips or are some samples and so if it is bias selection.
- **Original**: the data are original because use data published by a company that harvests data of bikes share of own service in Chicago.
- **Comprehensive**: the data aren't comprehensive, We cannot analyze customers' habits, due of in the data there isn't information about the relations between different trip of the same costumer, would be useful to indicate an anonymous identification of a customer like a pseudo id code and there isn't information to identify how much the stations cover the Chicago's neighborhoods, to understand if the problem to became member depends by far distance to reach the station and so it becomes difficult to use this service frequently or daily.
- **Current**: the datasets are currents, the datasets are upgraded each month with data of last month
- **Cited**: is define the source of the dataset

Conclusion: The dataset isn't ROCCC.

Process phase

Choos of the tools - Bigquery and Google sheet

To perform the process phase we will use Bigquery, because the dataset has more than a million data points and the spreadsheet cannot handle so many rows well. Using excel with powerquery we might be able to work with so many rows but the steps are similar to working in SQL it is preferable to work directly in SQL, for analysis you can export the data to R. Furthemore, in Bigquery there is also functions to do PIVOT to create complex relation with datas and it's possible to show the result in simple graphs (line chart or bar chart or dispersion chart). If you need to represent complex relationships with data or graphs there is a function in BigQuery to load the results of a query directly into the Google Sheet, allowing you to process the data with other operations and create graphs.

Data Integrity

We create a copy of original data so to rollback after manipulation.

Metadata of Dataset

the datasets have the following columns:

Name	type	description	
ride_id	[string]	unique identifier of the ride on bike	
rideable_type	[category]	type of bike [electric_bike, electric_scooter, classic_bike]	
started_at	[datetime]	start data of the ride	
ended_at	[datetime]	end data of the ride	
start_station_name	[string]	start station name	
start_station_id	[string]	start station identification code	
end_station_name	[string]	end station name	
end_station_id	[string]	end station identification code	
start_lat	[float]	position of start latitude	
start_lng	[float]	position of start longitude	
end_lat	[float]	position of end latitude	
end_lng	[float]	position of end longitude	
member_casual	[category]	type of customer [MEMBER	CASUAL]

3. Documentation of any cleaning or manipulation of data ## Data Cleaning Analysis

Analyze Qualitative values

```
SELECT distinct member_casual from `tripdata.triptrack` ;
```

member_casual
casual
member

```
SELECT count( distinct start_station_name) as start from `tripdata.triptrack` order by start;

SELECT count( distinct trim(start_station_name)) as start from `tripdata.triptrack` order by start;
```

Executing both queries result that the count is 1737, to indicate that it isn't necessary to apply trim operation on the start_station_name value.

```
SELECT count( distinct end_station_name) as end_name from `tripdata.triptrack` order by end_name;

SELECT count( distinct trim(end_station_name)) as end_name from `tripdata.triptrack` order by end_name
```

Executing both queries result that the count is 1749, to indicate that isn't necessary to apply trim operation on the end_station_name values.

```
SELECT distinct rideable_type from `tripdata.triptrack` ;
```

rideable_type
electric_bike
electric_scooter
classic_bike

Searching for Null values

```
SELECT
   count(month) as total,
   (select count(month)  FROM `tripdata.triptrack` where start_station_id is null) as null_start_station_id,
   (select count(month)  FROM `tripdata.triptrack` where end_station_id is null) as null_end_station_id,
   (select count(month)  FROM `tripdata.triptrack` where start_station_name is null) as null_start_station_name,
   (select count(month)  FROM `tripdata.triptrack` where end_station_name is null) as null_end_station_name,
   (select count(month)  FROM `tripdata.triptrack` where rideable_type is null) as null_rideable_type,
   (select count(month)  FROM `tripdata.triptrack` where member_casual is null) as null_member_casual,
   (select count(month)  FROM `tripdata.triptrack` where started_at is null) as null_started_at,
   (select count(month)  FROM `tripdata.triptrack` where ended_at is null) as null_ended_at,
   (select count(month)  FROM `tripdata.triptrack` where start_lat is null) as null_start_lat,
   (select count(month)  FROM `tripdata.triptrack` where end_lat is null) as null_end_lat,
   (select count(month)  FROM `tripdata.triptrack` where start_lng is null) as null_start_lng,
   (select count(month)  FROM `tripdata.triptrack` where end_lng is null) as null_end_lng
 FROM `tripdata.triptrack`
```

Column	number
total	5854544
start_station_name	1056535
end_station_name	1091792
start_station_id	1056535
end_station_id	1091792
rideable_type	0
member_casual	0
started_at	0
ended_at	0
start_lat	0
end_lat	7441
start_lng	0
end_lng	7441

the dataset presents some null values in the station names and station ids, the other informations are presents, but the null values don't influence our aim.

Searching for Outlayer values

Analyze min e max of quantity values and datetime

```
SELECT
   max(started_at)   as max_started_at,
   max(ended_at)   as min_ended_at,
   max(start_lat)   as max_start_lat,
   max(end_lat)   as max_end_lat,
   max(start_lng)   as max_start_lng,
   max(end_lng)   as max_end_lng,
   min(started_at)   as min_started_at,
   min(ended_at)   as min_ended_at,
   min(start_lat)   as min_start_lat,
   min(end_lat)  as min_end_lat,
   min(start_lng)   as min_start_lng,
   min(end_lng)   as min_end_lng
FROM `tripdata.triptrack`
```

Field	MIN	MAX
started_at	2024-09-30 23:54:05.552000 UTC	2023-10-01 00:00:05.000000 UTC
ended_at	2024-09-30 23:59:52.562000 UTC	00:02:02.000000 UTC
start_lat	42.07	2023-10-01 41.64
end_lat	87.96	16.06
start_lng	-87.52	-87.94
end_lng	1.72	-144.05

Thanks the max and min value for latitude and longitude it's possible to notice an error. Chicago has a latitude, longitude of 41.881832, -87.623177.

In the dataset thanks query that extract min and max instead there are values:

- latitude end station between 87.9 and 16, so none of the values are correct
- longituted end station between 1.72 and 182, very distant from -87.62.

On internet founded the approximatily latitude and longituted limits of Chicago, respectely [41,42.5] and [-88,-87]. Executing the following query to investigate

```
SELECT * `tripdata.triptrack`where end_lat not between 41 and 42.5;

SELECT count(*) as outlayer_lat_end from `tripdata.triptrack`where end_lat not between 41 and 42.5;

SELECT * from `tripdata.triptrack`where end_lng not between -88 and -87;

SELECT count(*) as outlayer_lng_end from `tripdata.triptrack`where end_lng not between -88 and -87;
```

when there is a end_lat or end_lng outlayer in the response there isn't the end_station_name and end_station_id

Outlayer count end station	# latitude	# longitude
	16	42

Searching for data where end date minor than start date

Using the following query we notice the there is an error in interval calculated, because a negative value results as minimum

```
SELECT min(DATETIME_DIFF( ended_at, started_at, MINUTE )) as min_length, max(DATETIME_DIFF( ended_at, started_at, MINUTE )) as
max_length  FROM `cyclistic_data.bike_trip_data`
```

min_length	max_length
-16656	1559

So in the dataset are present rows where the value of started_at are greater than end_at value.

To understand how much rows have this error , I execute the following query

```
SELECT count(*) as unvalid,
    (select count(*) FROM`cyclistic_data.bike_trip_data`) as total
FROM `cyclistic_data.bike_trip_data`
WHERE DATETIME_DIFF( ended_at, started_at, MINUTE ) <0

SELECT count(*) as unvalid,
    (select count(*) from`cyclistic_data.bike_trip_data`) as total
FROM `cyclistic_data.bike_trip_data`
WHERE DATETIME_DIFF( ended_at, started_at, MINUTE ) <1
```

Executing the query results 116 rows where the distance haa a negative value and 135832 rows where the time is under 1 minute. Respects 5854544 rows of all dataset, the rows with this error are 2% of the dataset.

Resume and conclusion

Dataset is composed by 5854544 rows.

field	format value	# NULL value	# Outlayer
ride_id	generic string no rule	0	0
rideable_type	["elettric_bike", "classik_bike, elettric_scooter"]	0	0
started_at	[format AAAA-MM-DD hh:mm:ss] in the correct interval 2023-10-01 - 2024-09-31	0	0
ended_at	[format AAAA-MM-DD hh:mm:ss] in the correct interval 2023-10-01 - 2024-09-31	0	135832
start_station_name		1056535	0
start_station_id	generic string no rule	1056535	0
end_station_name		1091792	0
end_station_id	generic string no rule	1091792	0
start_lat	between -90 and 90	1091792	0
start_lng	between -180 and 180	1091792	0
end_lat	between -90 and 90	1091792	16
end_lng	between -180 and 180	1091792	42
member_casual	["member", "casual"]	0	0

The dataset has 20% of null values on the information relating to the departure and arrival stations (name, id, longitude and latitude), furthermore it has few outlayers on the latitude and longitude values of the arrival stations. Since these columns are not considered useful to achieve the objective, they will be eliminated from the dataset that will be used in the analysis phase and therefore no interventions will be necessary to fill the null value and solve the outlayers' problem.

There are 135832 rows where the end date is wrong because minor the start or where the trip don't start if the difference is zero.

Due to the number being so low, we will remove these rows The other fields are complete and don't show any imputation errors.

Manipulation of data

We work on a copy of a original dataset called 'bike_trip_data' and all operations are recorded in a log file.

Columns deletion

In this table the columns of "start_lat", "end_lat","start_lng", "end_lng", "start_station_id", "end_station_id", "start_station_name" and "end_station_name" will be dropped.

```
ALTER TABLE
`graphite-space-440117-r9.cyclistic_data.bike_trip_data`
DROP COLUMN end_lat;

ALTER TABLE
`graphite-space-440117-r9.cyclistic_data.bike_trip_data`
DROP COLUMN end_lng;

ALTER TABLE
`graphite-space-440117-r9.cyclistic_data.bike_trip_data`
DROP COLUMN start_lat;

ALTER TABLE
`graphite-space-440117-r9.cyclistic_data.bike_trip_data`
DROP COLUMN start_lng;

ALTER TABLE
`graphite-space-440117-r9.cyclistic_data.bike_trip_data`
DROP COLUMN start_station_name;

ALTER TABLE
`graphite-space-440117-r9.cyclistic_data.bike_trip_data`
DROP COLUMN start_station_id;

ALTER TABLE
`graphite-space-440117-r9.cyclistic_data.bike_trip_data`
DROP COLUMN end_station_name;

ALTER TABLE
`graphite-space-440117-r9.cyclistic_data.bike_trip_data`
DROP COLUMN end_station_id;
```

Rows deletion

we delete all rows where the time of use is not useful for analysis they are 135832

```
DELETE FROM `cyclistic_data.bike_trip_data` WHERE DATETIME_DIFF( ended_at, started_at, MINUTE ) <1;
```

Operations to add columns and values

For help the analysis phase have been added this columns:

name	type	description
month	int64	Month of started trip
ride_length	int64	time interval between end date and start date
day_of_week	int64	day of week (1 = Sunday, 2 = Monday,... 7 = Saturday) of started trip

```
alter table `cyclistic_data.bike_trip_data` ADD COLUMN month INT64;
alter table `cyclistic_data.bike_trip_data` ADD COLUMN ride_length INT64;
alter table `cyclistic_data.bike_trip_data` ADD COLUMN day_of_week INT64;

alter table `cyclistic_data.bike_trip_data` ADD COLUMN season INT64;
alter table `cyclistic_data.bike_trip_data` ADD COLUMN season_name STRING;
```

We use the following operations to calculate values of the new columns

```
UPDATE  `cyclistic_data.bike_trip_data`  SET month = EXTRACT(MONTH from started_at)  WHERE 1=1 ;
UPDATE  `cyclistic_data.bike_trip_data`  SET day_of_week = EXTRACT(DAYOFWEEK from started_at)  WHERE 1=1 ;
UPDATE  `cyclistic_data.bike_trip_data`  SET ride_length = DATETIME_DIFF( ended_at, started_at, MINUTE )  WHERE 1=1 ;
```

Using following periods to define season of chicago the information

season	start month	end month
WINTER	december	february
SPRING	march	may
SUMMER	june	august

season	start month	end month
AUTUMN	september	november

```
UPDATE  `cyclistic_data.bike_trip_data`  SET season = 4, season_name="Autumn" WHERE month >8 AND month < 12 ;
UPDATE  `cyclistic_data.bike_trip_data`  SET season = 1, season_name="Winter" WHERE month <3 OR month = 12 ;
UPDATE  `cyclistic_data.bike_trip_data`  SET season = 2, season_name="Spring" WHERE month >2 AND month < 6 ;
UPDATE  `cyclistic_data.bike_trip_data`  SET season = 3, season_name="Summer" WHERE month >5 AND month < 9 ;
```

Validation

Run the following query to check:

- the number of data in the dataset
- if there are also negative elements in the ride_length column
- if all months and ride_length are filled in correctly

```
SELECT count(*) from tripdata.triptrack group by month
select * from tripdata.triptrack  order by ride_length DESC limit 100;
```

```
select ride_id,rideable_type, start_station_id, end_station_id, member_casual, month, day_of_week,
ride_length from tripdata.triptrack  order by ride_length DESC limit 100;

select rideable_type,count(rideable_type) as count_rideable_type from `tripdata.triptrack`
group by rideable_type;

select month,count(month) as count_rideable_type from `tripdata.triptrack`
group by month order by month ASC;
```

Selection of the most significant columns.

Name	type	description
ride_id	[string]	unique identifier of the ride on bike
rideable_type	[category]	electric_bike, electric_scooter, classic_bike
member_casual	[category]	Member, Casual
month	[1-12]	from the Started_at
dayofweek	[1-7]	from the Started_at
ride_length	[integer]	by difference of End_at end Started_at in minutes

4. Analysis Phase

Calculate Maximum, Minimum and average of ride_lenght

```
select
    CAST(avg(ride_length) AS INTEGER) as avg_ride_lenght,
    max(ride_length) as  max_ride_lenght,
    min(ride_length) as  min_ride_lenght,
    count(ride_id) as count_ride
from
    `cyclistic_data.bike_trip_data`
```

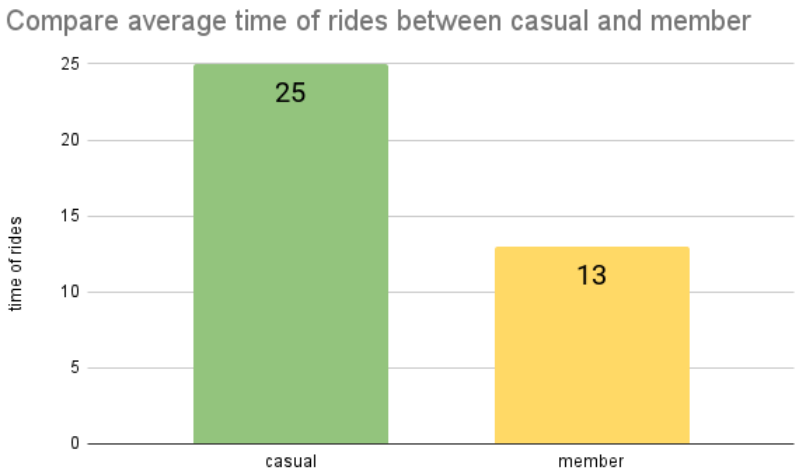
avg_ride_lenght	max_ride_lenght	min_ride_lenght	count_ride
17	1559	1	5718712

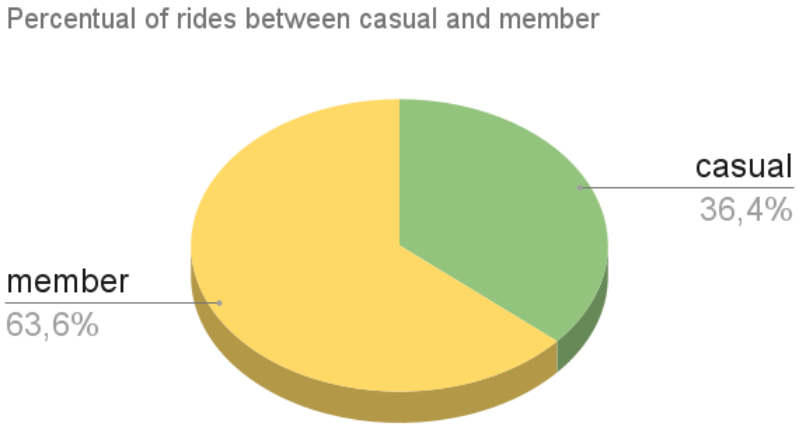
Calculate Maximum and average of ride_lenght by member_casual

```
select
    member_casual,
    CAST(avg(ride_length) AS INTEGER) as avg_ride_lenght,
    max(ride_length) as  max_ride_lenght,
    min(ride_length) as  min_ride_lenght,
    count(ride_id) as count_ride,
    count(ride_id)*100/5718712 as count_perc
from
    `cyclistic_data.bike_trip_data`
group by
    member_casual
order by
    avg_ride_lenght desc
```

the union of the results is the following:

member_casual	avg_ride_lenght	max_ride_lenght	min_ride_lenght	count_ride	count_perc
casual	25	1559	1	2067119	36.13%
member	13	1559	1	3651593	63.87%





> - the maximum and minimum distances are the same so it isn't useful for analysis
> - Count of rides of members is double then casuals
> - Average time of rides of casual greater than member sometime is the double

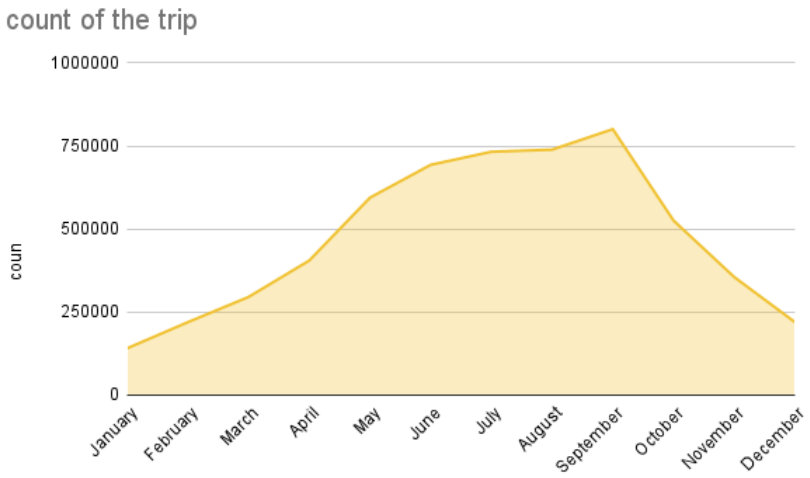
Calculate Maximum and average of ride_length by month

```
select
  month,
  FORMAT_DATETIME("%B",  started_at) as month_name,
  CAST(avg(ride_length) AS INTEGER) as avg_ride_lenght,
  count(ride_id) as count_trip
from
    `cyclistic_data.bike_trip_data`
group by month_name, month order by month ASC
```

month	month_name	avg_ride_lenght	count_trip
1	January	15	140537
2	February	15	219183
3	March	15	295099
4	April	17	404830
5	May	19	594130
6	June	20	693149
7	July	20	732543
8	August	19	738864
9	September	16	800533
10	October	16	525420
11	November	14	355114
12	December	13	219310

Average of rides lenght for Month

> The average monthly distance of all types of customers increases slightly during the summer season, but every month it is around 16 minutes



Count of rides for Month

> The number of trips is very different monthly. In winter there are around 100,000 races, but in summer the number increases six times.

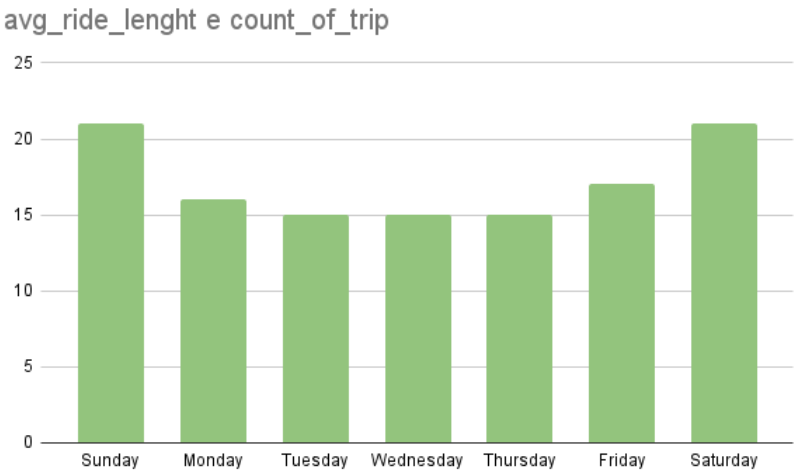
> Comparing the 2 graphs shows that in summer the number of races increases but the distances are short, and vice versa going towards winter the number of races decreases but the duration increases.

calculate avg of ride_length and the MODE respect day of week for Total dataset

```
select
    FORMAT_DATETIME("%A",  started_at) as day_name,day_of_week
    , CAST(avg(ride_length) AS INTEGER) as avg_ride_lenght, max(ride_length) as  max_ride_lenght, count(ride_id) as count_of_trip
from
    `cyclistic_data.bike_trip_data`
group by day_of_week,day_name order by day_of_week ASC
```

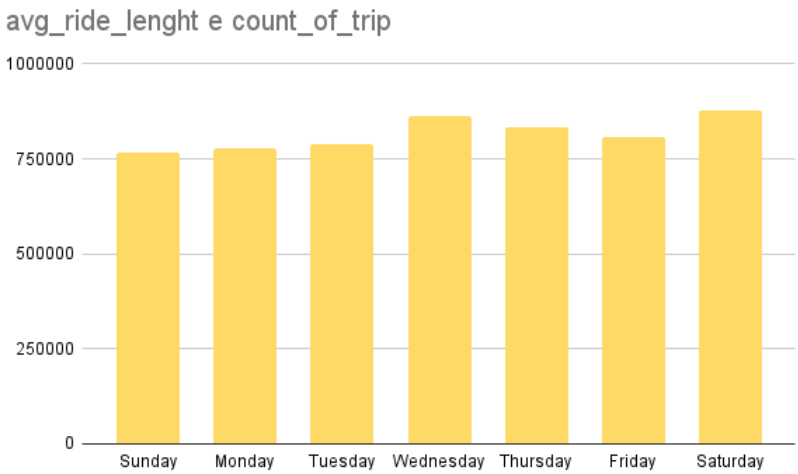
results are:

day_name	day_of_week	avg_ride_lenght	max_ride_lenght count_of_trip	
Sunday	1	21	1500	768568
Monday	2	16	1500	777879
Tuesday	3	15	1499	788371
Wednesday	4	15	1500	864628
Thursday	5	15	1499	833148
Friday	6	17	1500	806926
Saturday	7	21	1559	879192



AVG Ride_length

> The average duration of races increases slightly over the weekend, but there are no big differences



Count trip

> the number of weekly trips is almost identical, compared to the number analyzing the entire year there are no significant differences

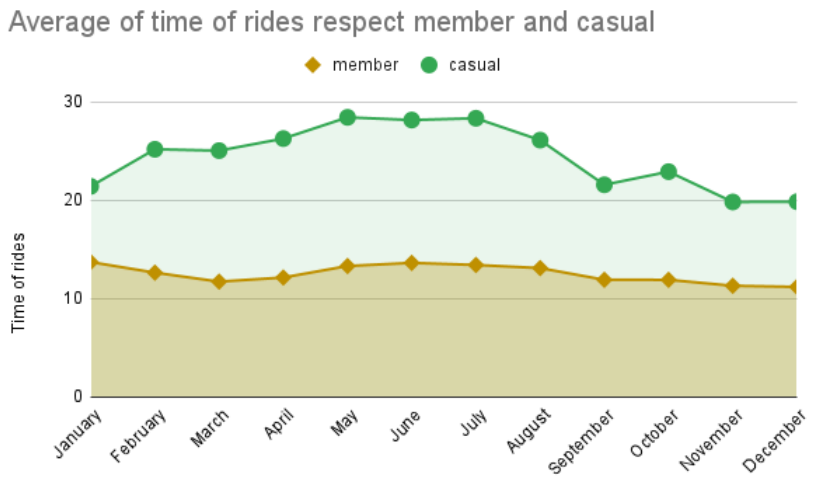
> analyzing both the two graphs over the entire year we do not notice a significant element where the days of the weeks can be useful for some advice.

Pivot AvgLenght vs Member_casual for each month

```
SELECT * FROM
(
select
    month,
    FORMAT_DATETIME("%B",  started_at) as month_name,
    member_casual,
    ride_id,
    ride_length

from
    `cyclistic_data.bike_trip_data`
)
PIVOT
(
  -- #2 aggregate
  avg(ride_length) as avg_ride_length,
  count(ride_id) as count_ride
  -- #3 pivot_column
  FOR member_casual in ("member","casual")
) order by month
```

month	month_name	avg_ride_length_member	count_ride_member	avg_ride_length_casual	count_ride_casual
1	January	13.720059264841895	116764	21.42115004416776	23773
2	February	12.64393597705536	172938	25.189706995350853	46245
3	March	11.730490204306188	214482	25.051490380440839	80617
4	April	12.148011545283444	276823	26.276719241916435	128007
5	May	13.318053998568308	370195	28.436711545761032	223935
6	June	13.636995395279905	402196	28.165212250775923	290953
7	July	13.424993187768791	422035	28.342632073891817	310508
8	August	13.108227220243613	430437	26.120352628012455	308427
9	September	11.915971457223421	465687	21.579418598400476	334846
10	October	11.913057047446381	352461	22.913447695696686	172959
11	November	11.308315176239709	258852	19.831771623278119	96262
12	December	11.196072853137991	168723	19.860418684642305	50587

Bar chart with Avg ride Length

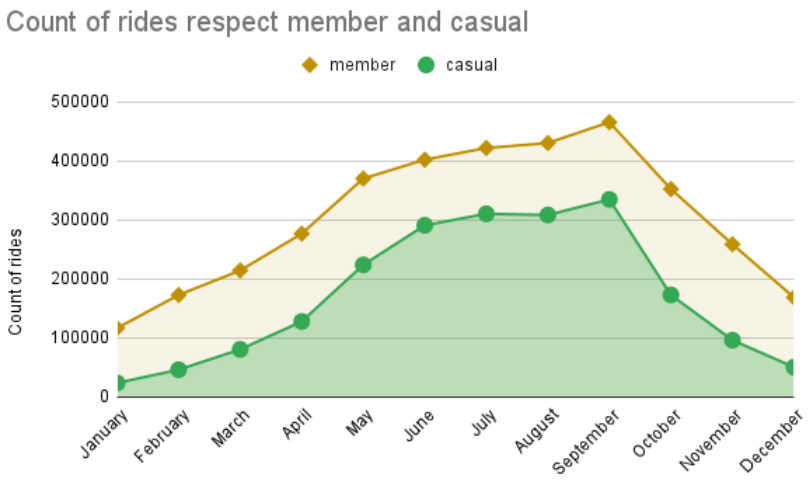


Member	Casual	
mean	12,5	24,4

Member	Casual	
variance	0,8	10,4

> - The average travel time of casuals is longer than members, especially in the spring and summer months. Casuals have higher variance than members.
> - the Member variance is almost 0 to indicate that they always spend the same amount of time each month.

Bar Chart with Count ride



Calculated mean and variance starting from previous data

> The count's trend of rides in the year is the same for both members and casuals. The count of rides for both increase during spring and summer months

> - THe COUNT of trips by MEMBERs is greater than that of CASUALs on all months of year
> - THe AVERAGE DISTANCE by trips of CASUALs is greater than that of MEMBERs on all months of year

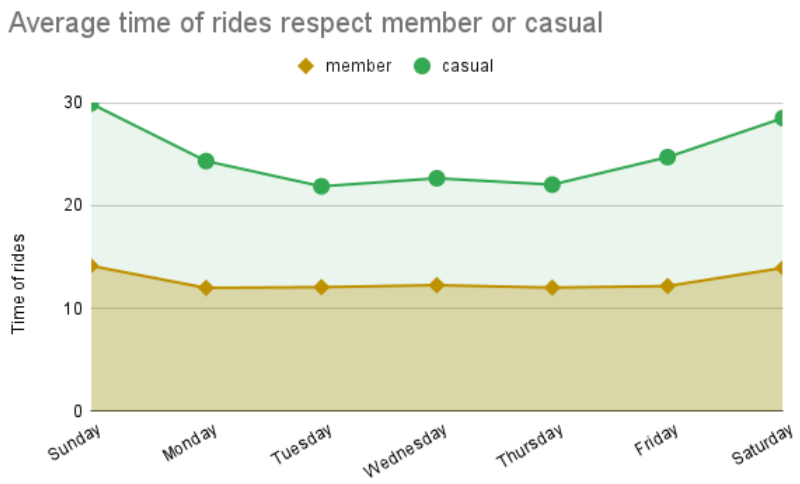
Ordering the results:

> - The average length used by MEMBER has max: 13 minutes, min: 11 minutes,average: 12 minutes
> - The average length used by CASUAL has max: 28 minutes, min: 19 minutes,average: 25 minutes

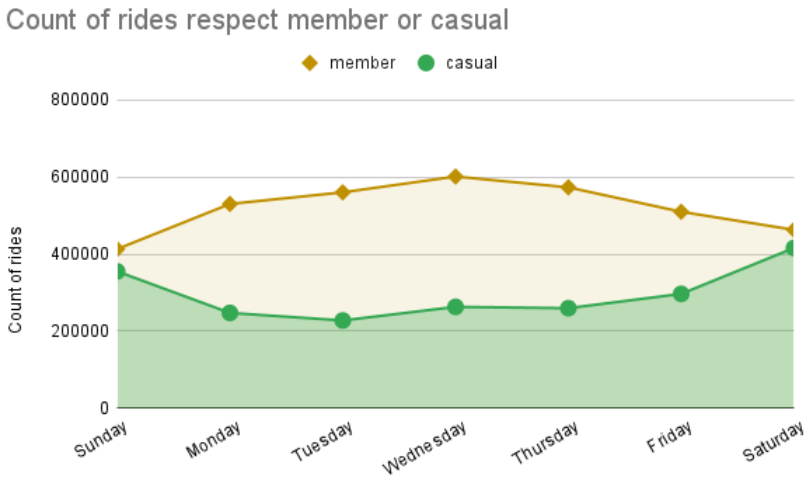
PIVOT of count and distance respect day of week and member_casual value

```
SELECT * FROM
(
select
    day_of_week,
    FORMAT_DATETIME("%A",  started_at) as day_name,
    member_casual,
    ride_length,
    ride_id

from
    `cyclistic_data.bike_trip_data`
)
PIVOT
(
  -- #2 aggregate
  avg(ride_length) as avg_ride_length, count(ride_id) as count_ride
  -- #3 pivot_column
  FOR member_casual in ("member","casual")
) order by day_of_week
```

day_of_week	day_name	avg_ride_length_member	count_ride_member	avg_ride_length_casual	count_ride_casual
1	Sunday	14.160626902803928	413206	29.955144331695539	355362
2	Monday	12.011196056393267	530276	24.3477461904743	247603
3	Tuesday	12.083290951756382	560385	21.890888036984727	227986
4	Wednesday	12.275715155203891	601338	22.67404003190396	263290
5	Thursday	12.036023024594471	573300	22.056282903851443	259848
6	Friday	12.189292269318853	509968	24.729793438802822	296958
7	Saturday	13.95906028675	463120	28.529511719125519	416072



- **member**: the trend increase in work day
- **casual**: the trend became greater in weekend

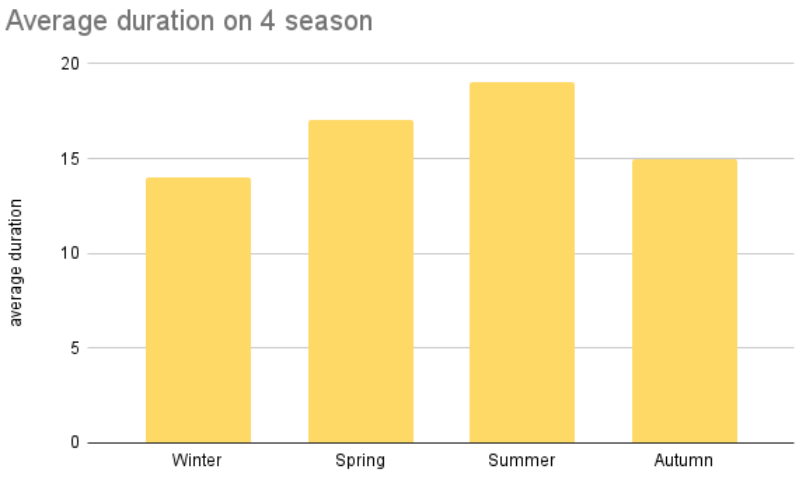


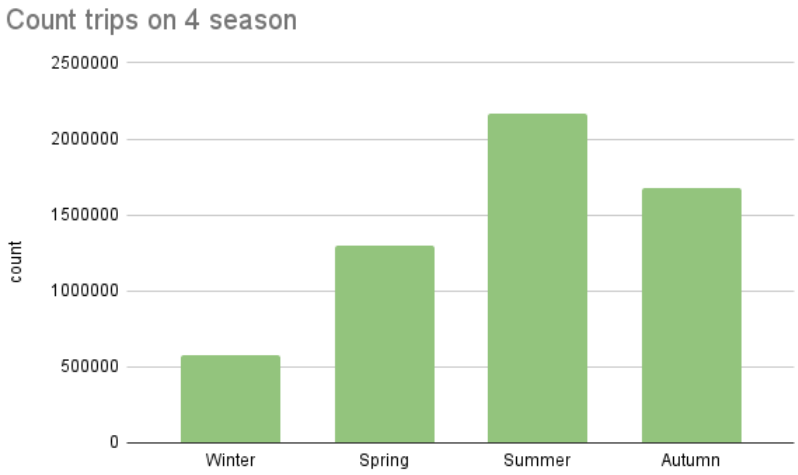
> - **member**: the trend is lower and constant
> - **casual**: the trend is greater during the weekend (max 30 sundsy and saturdsy, min 22 tuesday, wednesday, thursday)

Analysis vs Season

```
select  season, season_name, count(ride_id) as count_ride, avg(ride_length) as avg_ride_length from `cyclistic_data.bike_trip_data`
group by season, season_name
```

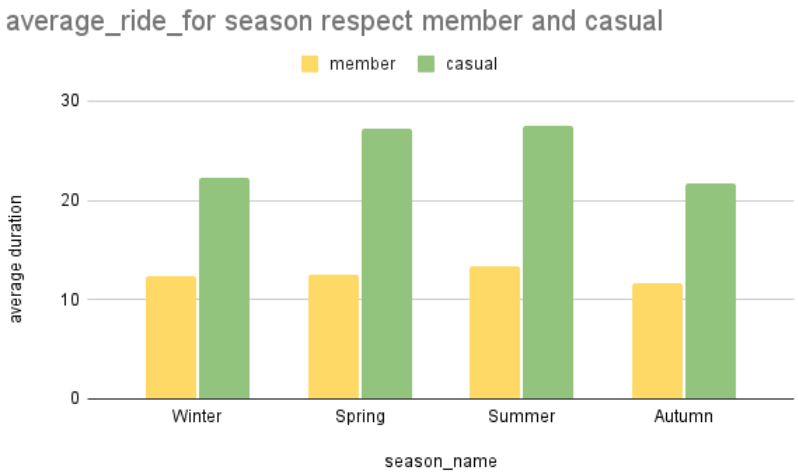
season	season_name	count_ride	avg_ride_length
1	Winter	579030	14.431865361034822
2	Spring	1294059	17.433718246231443
3	Summer	2164556	19.331639837453917
4	Autumn	1681067	15.331391312779319

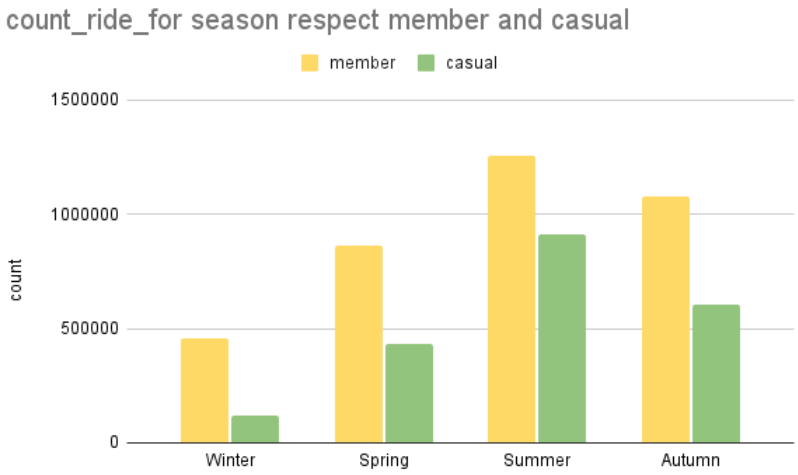





```
select * from(
select  season, season_name,member_casual,ride_length,ride_id from `cyclistic_data.bike_trip_data`
)
PIVOT
(
  -- #2 aggregate
  avg(ride_length) as avg_ride_length, count(ride_id) as count_ride
  -- #3 pivot_column
  FOR member_casual in ("member","casual")
) order by season
```

season	season_name	avg_ride_length_member	count_ride_member	avg_ride_length_casual	count_ride_casual
1	Winter	12.385146970605868	458425	22.211533518510837	120605
2	Spring	12.546842716192673	861500	27.166594614838651	432559
3	Summer	13.384280144229406	1254668	27.532607309910603	909888
4	Autumn	11.768970287836614	1077000	21.682884514466107	604067

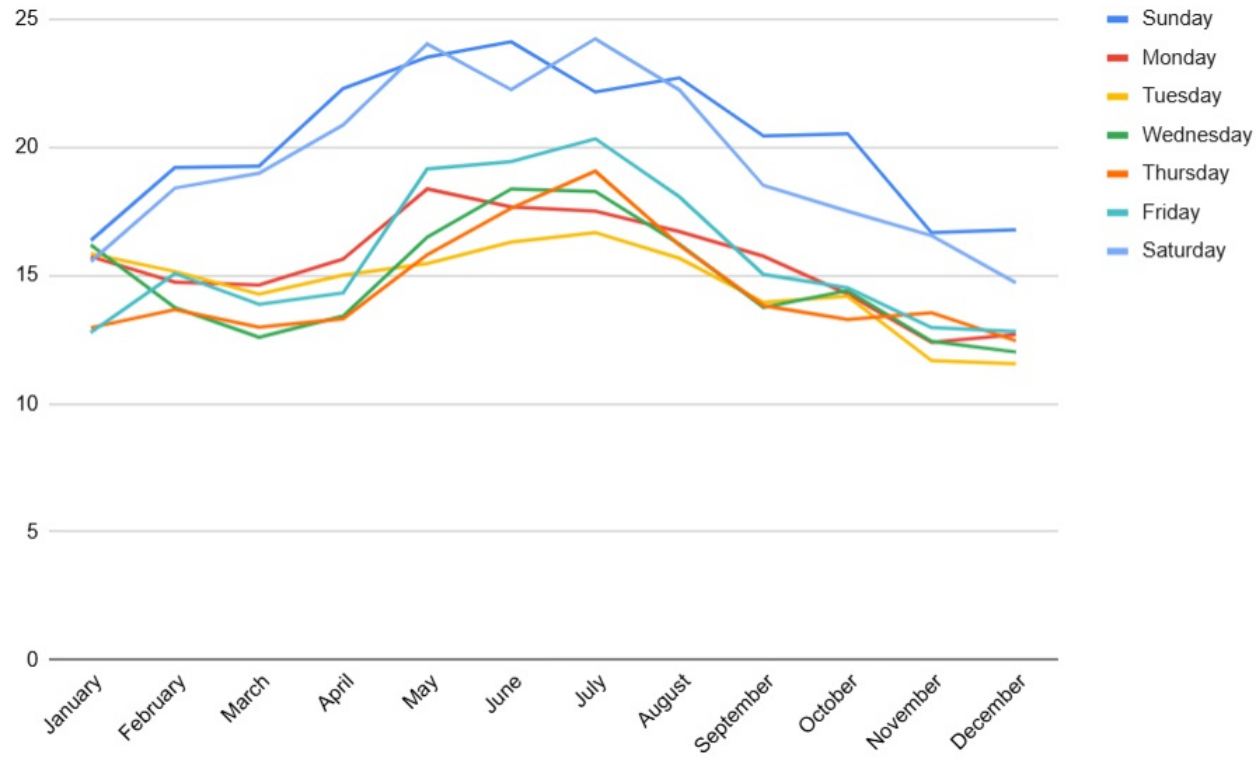




Analyze the behavior of cyclists on the day of the week for each month

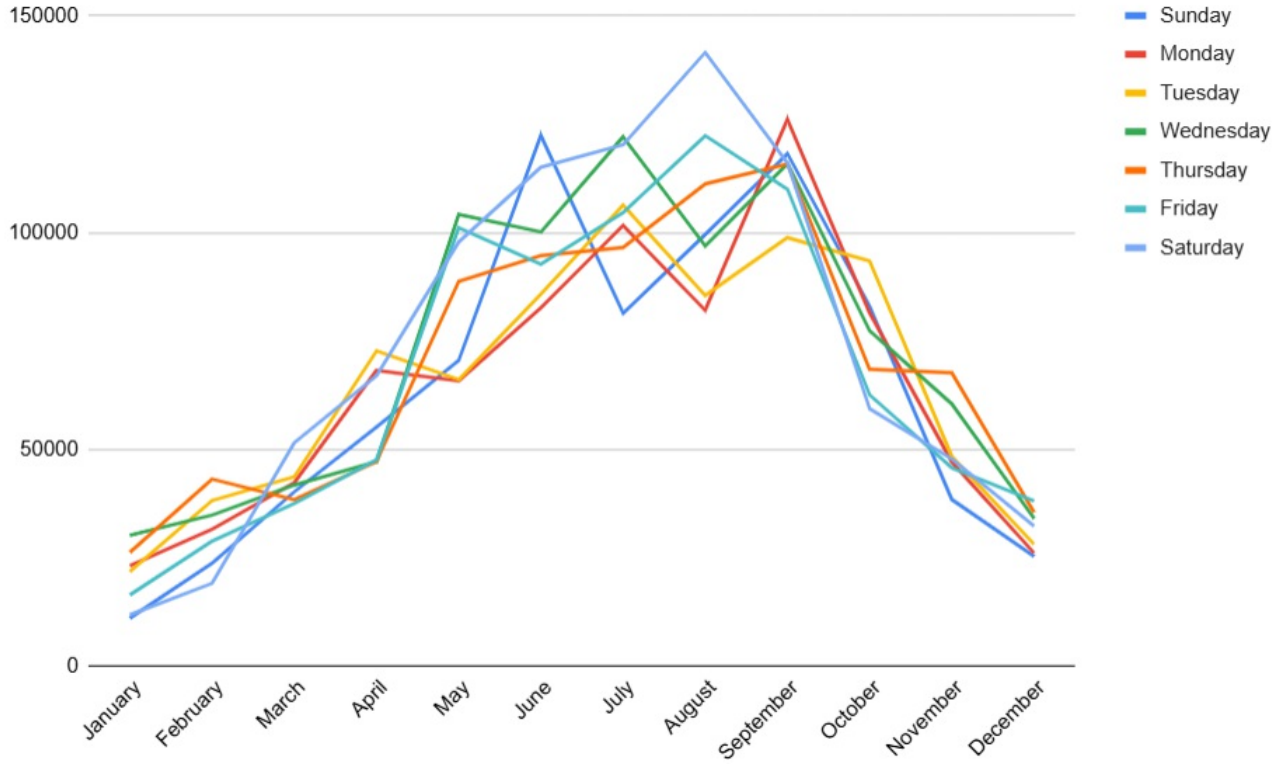
```
SELECT * FROM
(
  -- #1 from_item
  SELECT
    month,
    FORMAT_DATETIME("%B",  started_at) as month_name,
    FORMAT_DATETIME("%A",  started_at) as day_name,
    ride_length
  FROM `cyclistic_data.bike_trip_data`
)
PIVOT
(
  -- #2 aggregate
  count(ride_length) AS count_ride, avg(ride_length) AS avg_ride_length
  -- #3 pivot_column
  FOR day_name in ("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday")
) ORDER BY month ASC
```

Average of length Ride for weekday and month



> - the trend of the average race time is always the same for all day of the week
> - On sunday and saturday the time is greater than on other days
> - There is a slight increase in trip times for all days in the summer months

Count of Ride for weekday and month

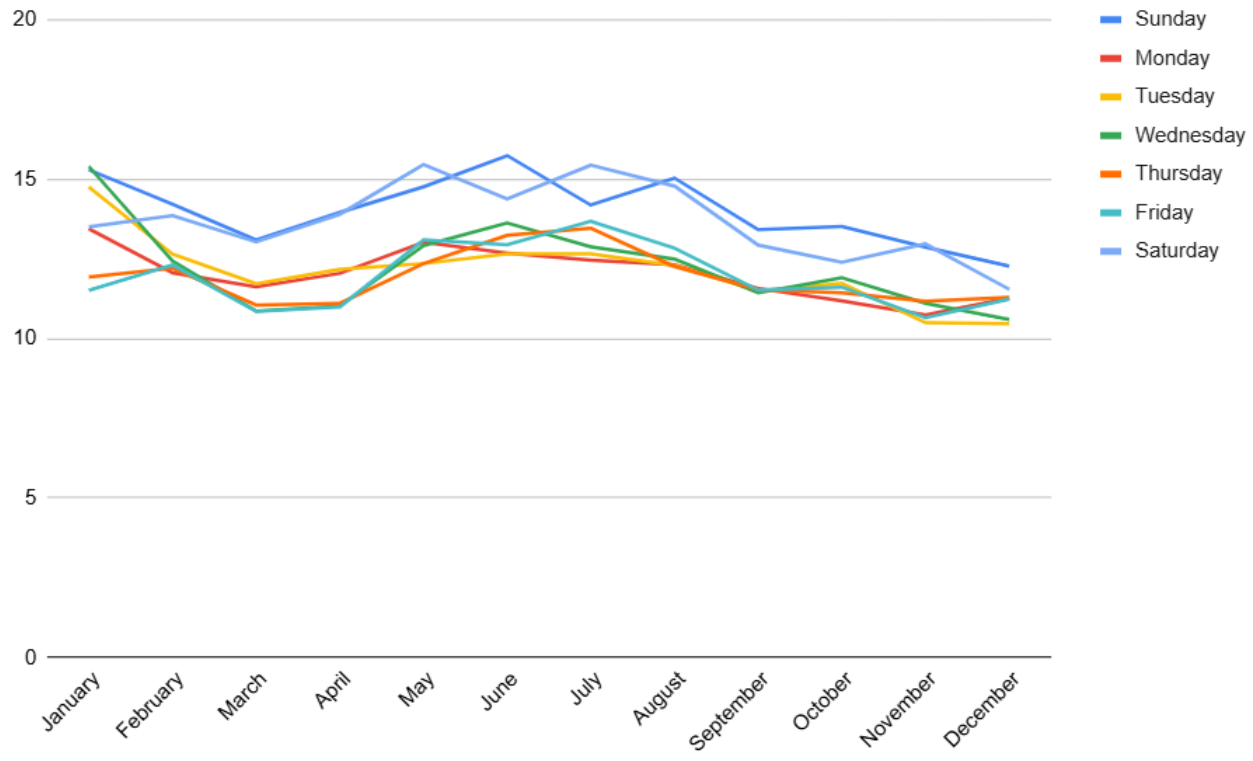


> We notice a slight difference between summer/spring months and Fall/Winter months.
>
> - During Fall/Winter months the service is used most frequently on workdays.
> - During Spring/Summer months the use of service tends to become more frequent on weekends.
> - the trend is similar throughout the day of each month
> - there is a strong increase in the number of trips in the summer period, 6 times greater than in the winter months.

Calculate average of ride_length and the MODE respect day of week for MEMBER o CASUAL kind

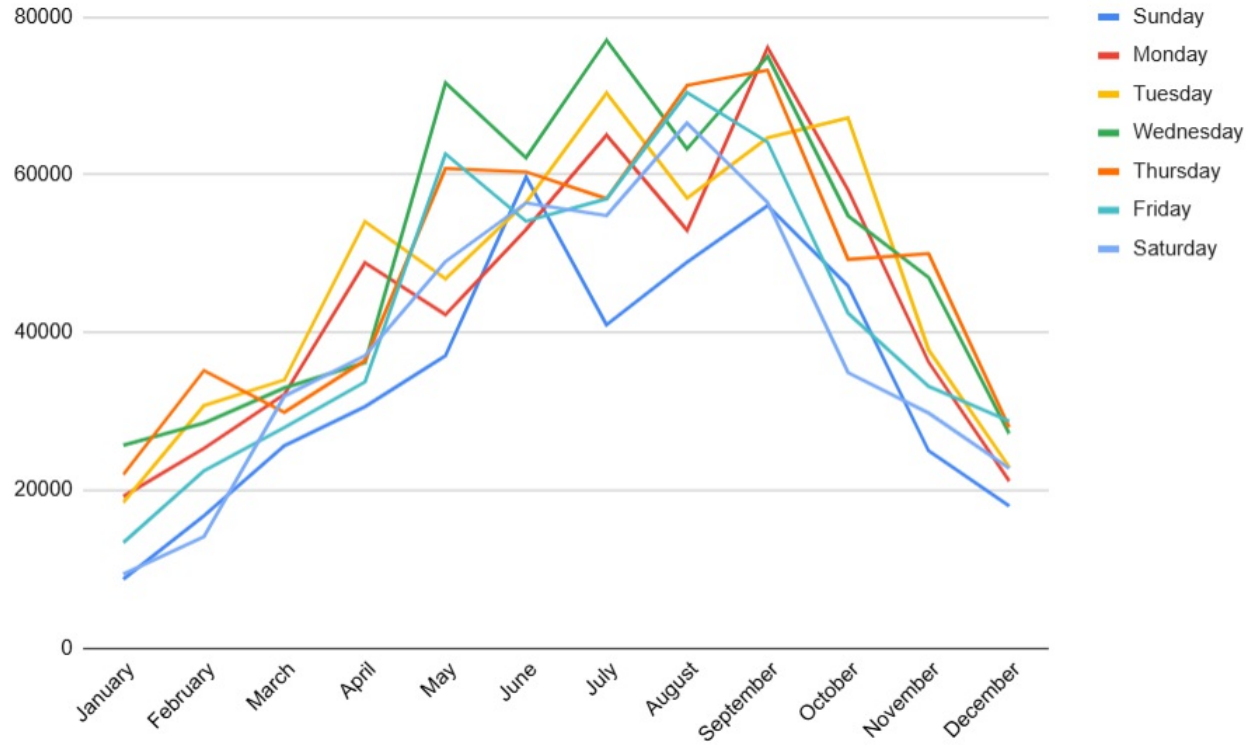
```
SELECT * FROM
(
  -- #1 from_item
  SELECT
    month,
    FORMAT_DATETIME("%B",  started_at) as month_name,
    FORMAT_DATETIME("%A",  started_at) as day_name,
    ride_length
  FROM `cyclistic_data.bike_trip_data`
  WHERE member_casual="casual"
  --- WHERE member_casual="casual"
)
PIVOT
(
  -- #2 aggregate
  avg(ride_length) AS avg_ride_length,
  count(ride_length) AS count_ride_length
  -- #3 pivot_column
  FOR day_name in ("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday")
) ORDER BY month ASC
```

Member length of rides for weekdays and months



- the average distance of trips made by members on weekends is higher than on weekdays
- Saturday is always constant throughout the month and has the highest value
- Sunday is similar to Saturday but in August the distance becomes black to zero.

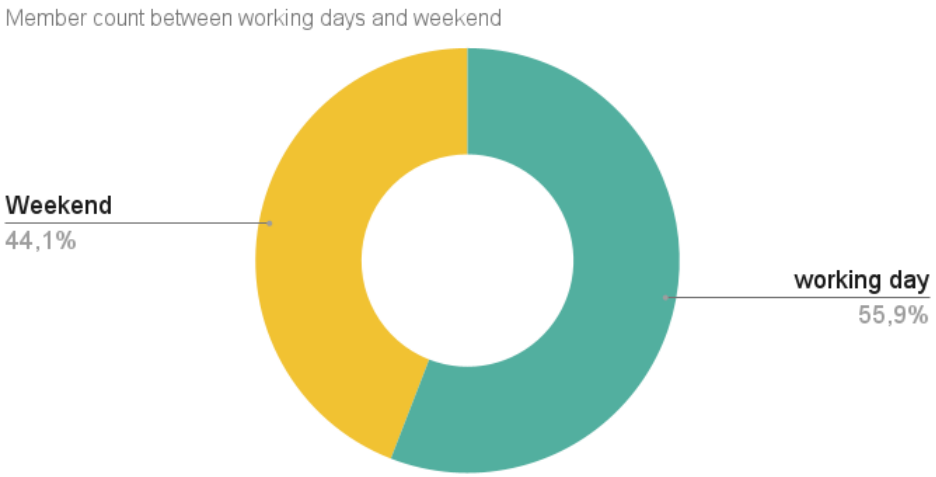
Member count of rides for weekdays and months

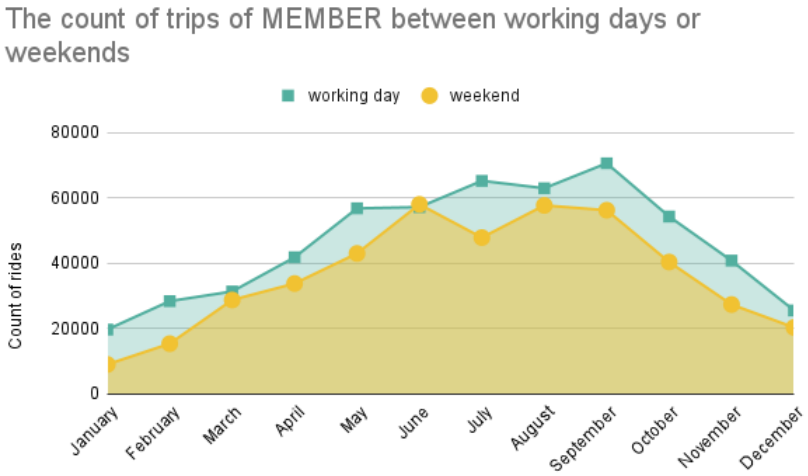


- The count of trips made by members has the same trend for every day in all months of the year
- Saturday has the lowest value, so members have decreased this month but they use the bicycle for a greater distance.

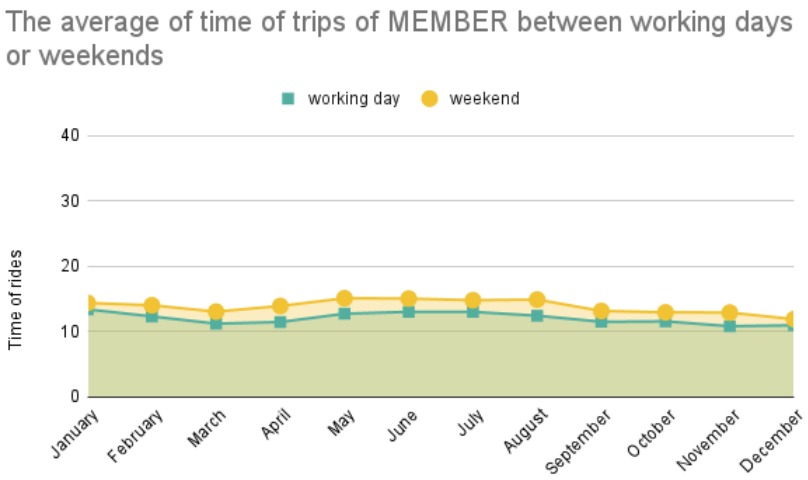
Calculated respect working day and weekend

month_name	COUNT WORKDAY	AVG WORKDAY	COUNT WEEKEND	AVG WEEKEND
January	19736,2	13,40947133	9041,5 14,40122708	
February	28425,8	12,33294523	15404,5 14,0367029	
March	31383,6	11,2251734	28782 13,06597484	
April	41836,6	11,47145423	33820 13,93211192	
May	56829	12,74971958	43025 15,11134556	
June	57217,2	13,03167231	58055 15,06097321	
July	65252,6	13,02877173	47886 14,81446659	
August	62994,2	12,44304317	57733 14,91031507	
September	70633,8	11,51564924	56259 13,17258379	
October	54332,8	11,57601238	40398,5 12,95802295	
November	40815,2	10,83697803	27388 12,92142822	
December	25596,4 10,9743638	20370,5 11,90997178		





> The service is used by Members more frequent in working day than weekend (55.9% - 44.1%), this trend is similar in every months

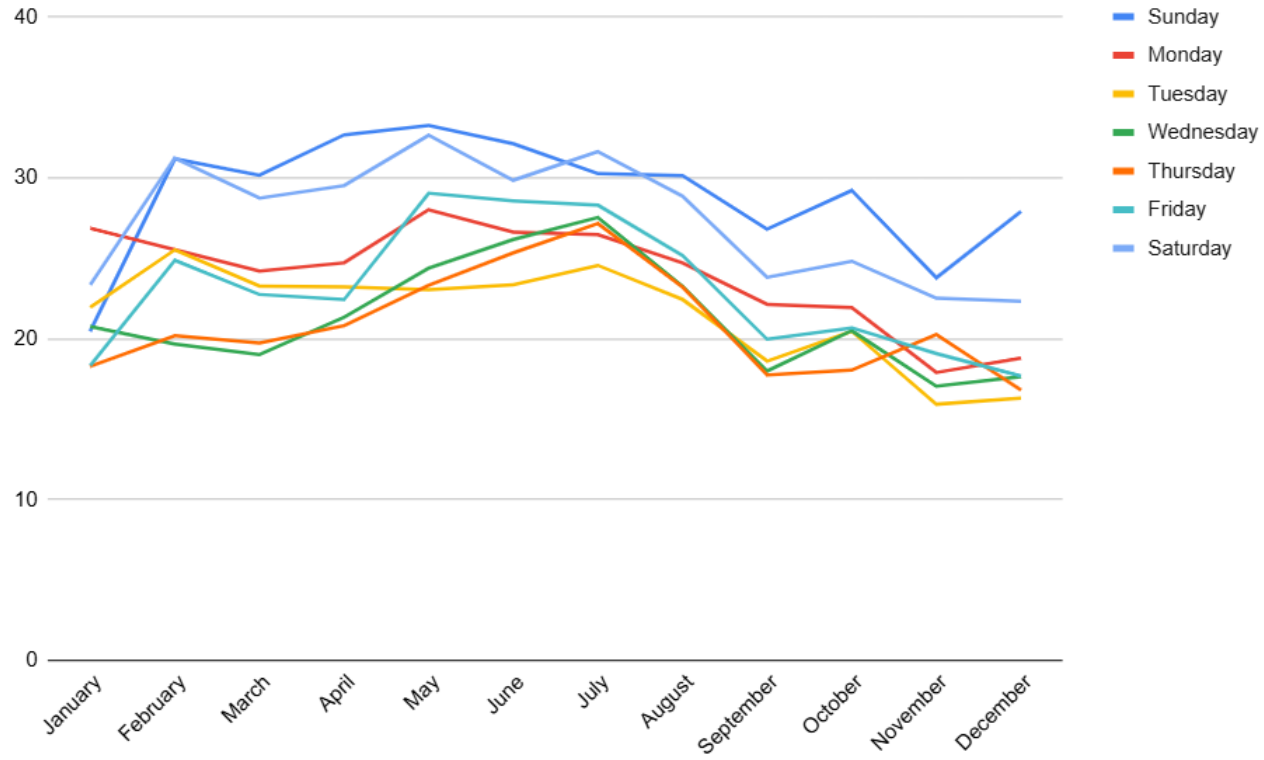


> member: trend in every months of year is similar in all days

Calculate avg of ride_length and the MODE respect day of week for CASUAL dataset

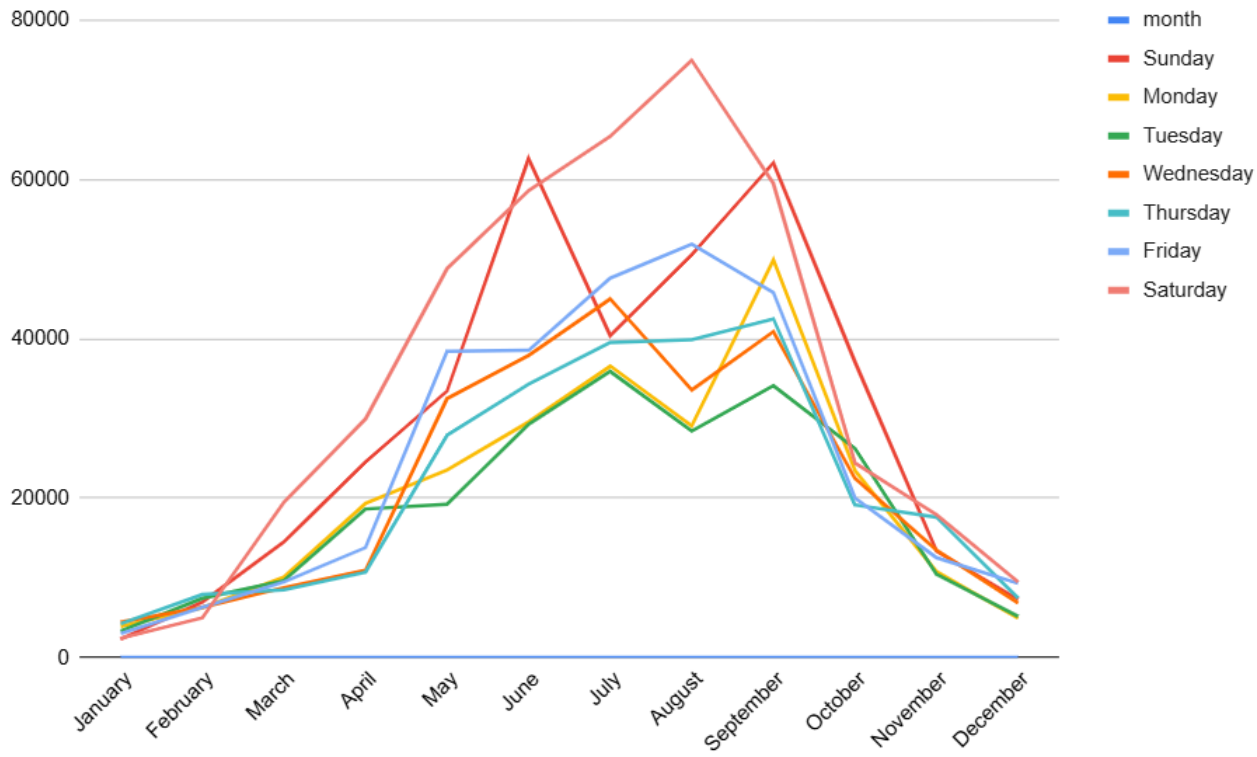
```
SELECT * FROM
(
  -- #1 from_item
  SELECT
    month,
    FORMAT_DATETIME("%B",  started_at) as month_name,
    FORMAT_DATETIME("%A",  started_at) as day_name,
    ride_length
  FROM `cyclistic_data.bike_trip_data`
  WHERE member_casual="casual"
)
PIVOT
(
  -- #2 aggregate
  avg(ride_length) AS avg_ride_length,
  count(ride_length) AS count_ride_length
  -- #3 pivot_column
  FOR day_name in ("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday")
) ORDER BY month ASC
```

Casual Average length of rides for weekdays and months



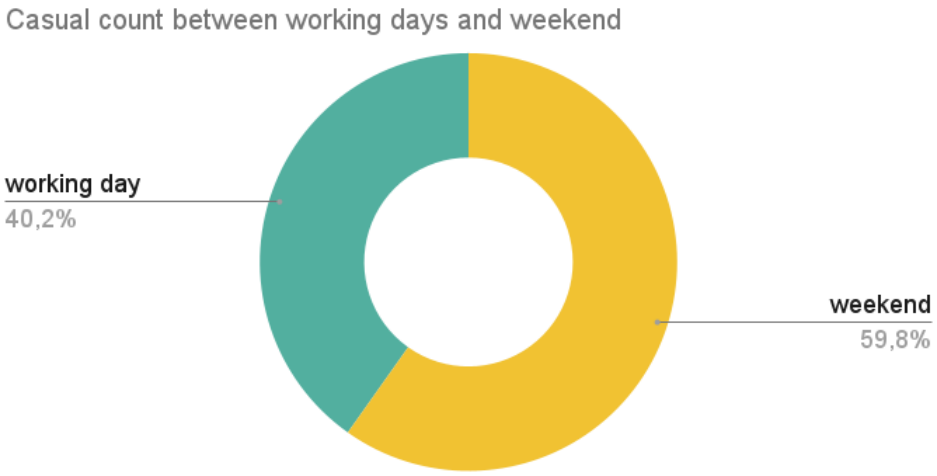
- the averages distances on all days are similar,
- there are many variability in between january and may

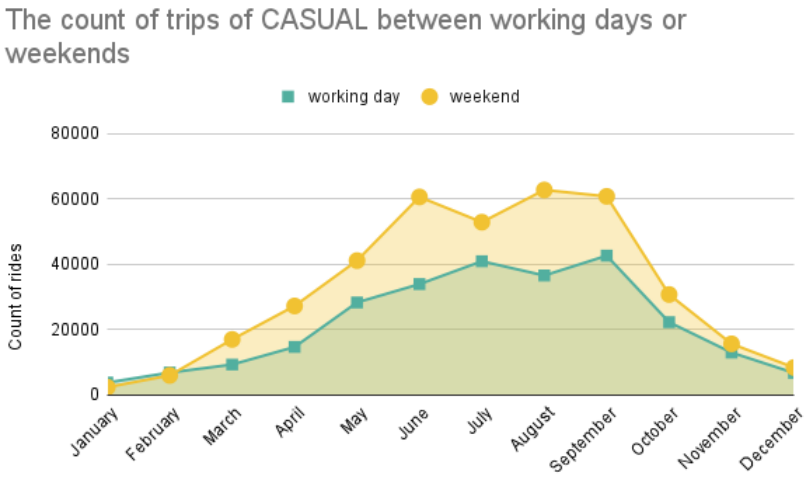
Casual Count of rides for weekdays and months



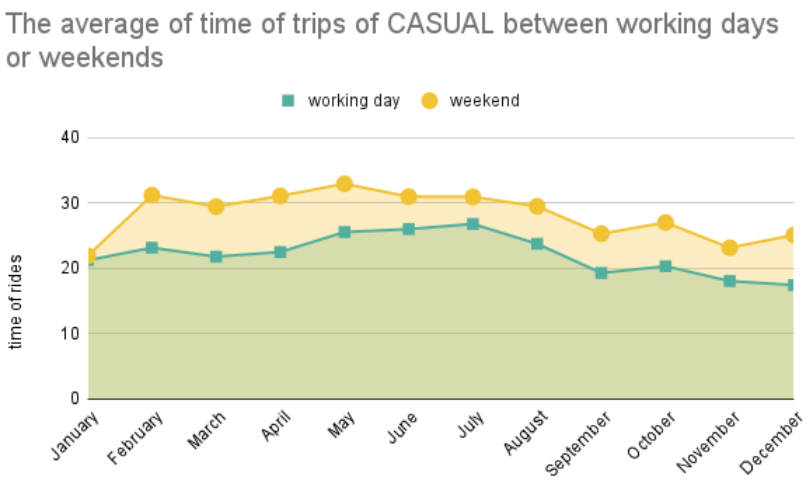
- the count increases for all days on the summer
- the tendencies are similars for all days on all months
- during december, janury, february the count si very low
- Th count is most on sunday and saturday, on other days the count has the same average for all months

Calculated respect working day and weekend





- the service is used by Casuals more frequent during the weekend (59.8% - 40.2%)
- that casuals in weekend increase the use during the spring and summer



> casual: the service in every months of year is use smore in week end than working days

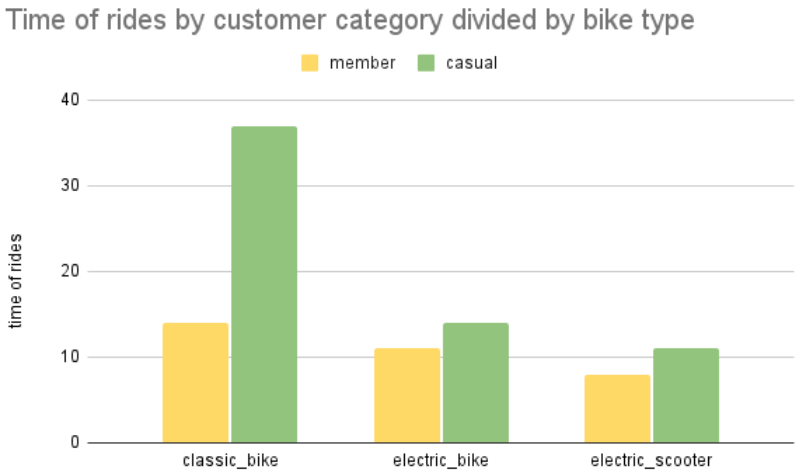
Analysis member-casual respect rideable_type

```
select * from(
select  rideable_type,member_casual,ride_length,ride_id from `cyclistic_data.bike_trip_data`
)
PIVOT
(
  -- #2 aggregate
  avg(ride_length) as avg_ride_length,
  count(ride_id) as count_ride
  -- #3 pivot_column
  FOR member_casual in ("member","casual")
) order by rideable_type
```

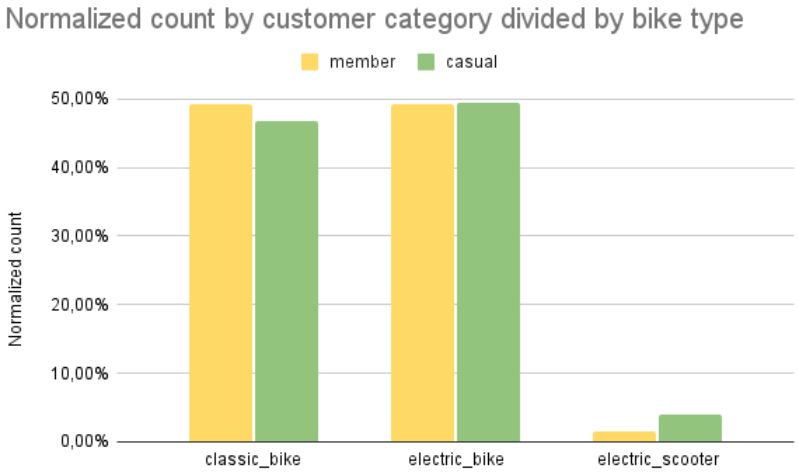
rideable_type	avg_ride_length_member	count_ride_member	avg_ride_length_casual	count_ride_casual
classic_bike	14.246873641650241	1799058	37.879924041505731	965264
electric_bike	11.058633345227463	1796401	14.73946325233608	1020405
electric_scooter	8.160526597071291	56134	11.972154696132591	81450

Starting by these data we calculate percentual of count

category	classic_bike	electric_bike	electric_scooter	total
member	49,27%	49,19%	1,54%	3651593
casual	46,70%	49,36%	3,94%	2067119



> All kind of customer use the same genre of bike in the same percentual



> casual when use classic_bike tends to do longer rides then member

Result of the case study - Reccomendation and

- Casuals rides are 28% less than members.
- The duration of the casuals' races is longer than that of the members (25 minutes casuals vs 13 members)
- Casuals increase their use of the service in the spring and summer more than members
- Casuals prefer to use the service on weekends (average 30 minutes)
- The rides count of members always has the same trend throughout the months on both working days and weekends
- Casuals users prefer the classic bikes for traveling for a long periods, more than members (45 minutes versus 12 minutes)
- Members use the service with a constant duration (low variance 0,8) as if the use depended on a scheduled event, vice versa the duration of the
- Casuals has a high variance to indicate that the use does not follow a scheduled need.

Insights

- Casuals use the service for pleasure, in fact they prefer spring and summer weekends and use the service for a long time
- Members use the service to cycle for usual services such as going to work or other repetitive activity, to reach easily a destination.

Reccomendation

The result of this study : To evaluate subscription strategies that favor those who use the service for:

1. longer periods
2. spring and summer months
3. using classic bike especially on weekends

Advices to improve strategy

Connecting the trips for the same user to the dataset would allow us to better understand the service's habits of use. Thus, we could identify homogeneous groups to which specific strategies can be dedicated through clustering.

LOG FILE - TABLE bike_trip_data

DATE	OPERATION	ELEMENT	DESCRIPTION
2024.10.28	COPY	all data	copied all data from 'cyclistic_data.bike_trip' to 'cyclistic_data.bike_trip_data'
2024.10.28	DEL COLUMN	start_lan	unuseful for analysis and with null elements
2024.10.28	DEL COLUMN	start_lnge	unuseful for analysis and with null elements
2024.10.28	DEL COLUMN	end_lan	unuseful for analysis and with null elements
2024.10.28	DEL COLUMN	end_lng	unuseful for analysis and with null elements
2024.11.03	DEL COLUMN	start_station_name	unuseful for analysis and with null elements
2024.11.03	DEL COLUMN	end_station_name	unuseful for analysis and with null elements
2024.11.03	DEL COLUMN	start_station_id	unuseful for analysis and with null elements
2024.11.03	DEL COLUMN	end_station_id	unuseful for analysis and with null elements
2024.11.04	ADD COLUMN	month	useful for analysis
2024.11.04	ADD COLUMN	day_of_week	useful for analysis
2024.11.04	ADD COLUMN	ride_length	useful for analysis
2024.11.04	UPDATE VALUE	month	calculated from star_data_trip
2024.11.04	UPDATE VALUE	day_of_week	calculated from star_data_trip
2024.11.04	UPDATE VALUE	ride_length	calculated like time interval between end dates and start dates of the trips
2024.11.05	DEL ROW	some rows	deleted rows where interval between end_at - start_at is less the one minute.