

DIVVY BIKE SHARING ANALYSIS

PowerBI Report

by

William Gan

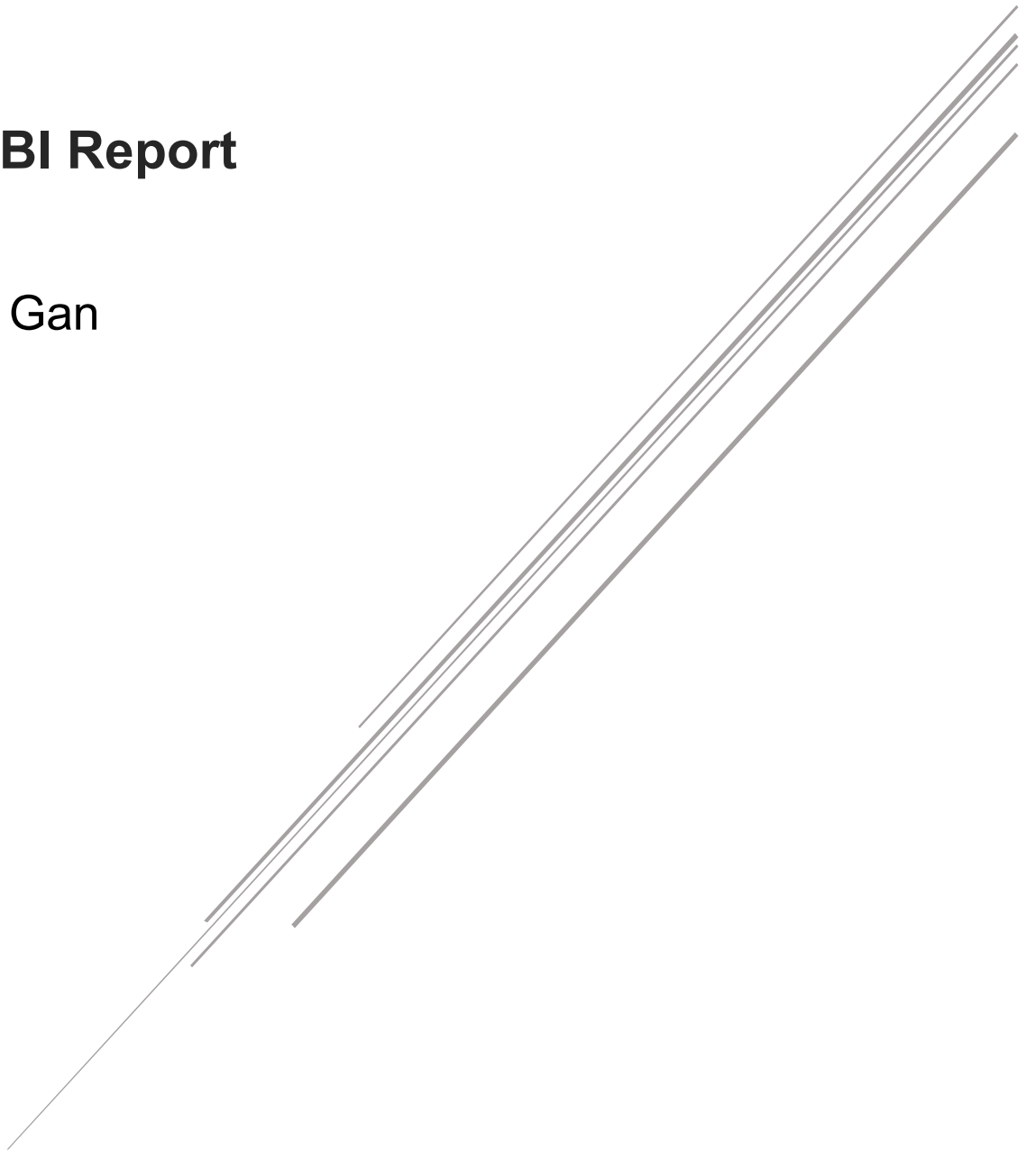


Table of Contents

1. BI Data Source and Requirements	4
1.1. Introduction to the Dataset	4
1.2. Business Objective	4
1.3. Key Performance Indicators (KPI)	5
1.4. Key User Groups	5
1.5. Importing Data Source	6
1.6. Overview of the Dataset	8
2. Data Cleansing and Pre-Processing	10
2.1. Trimming the whitespaces of text fields	11
2.2. Removing Blank Rows and Duplicates from the table	12
2.3. Filtering the <ride_id> column (M Language)	13
2.4. Filtering the <rideable_type> and <member_casual> column	14
2.5. Changing the data type of <started_at> and <ended_at> columns	15
2.6. Changing the data type of columns containing coordinates	16
2.7. Removing errors from the table	17
3. Data Modelling and Relationship	18
3.1. Creation of two ID columns in the <trips> table (M Language)	18
3.2. Creating the <datetime> table	20
3.3. Creating the <coordinates> Table	27
3.4. Creating the <stations> table	28
3.5. Transforming the <trips> table into a fact table	35
3.6. Data Modelling Relations and Summary	36
3.7. Filtering out invalid durations	39
4. Executive Summary	41
5. Data Analysis	42
5.1. Introduction	42
5.2. Analysis of Total Trips	44
5.3. Analysis of Total Ride Duration (New column using DAX)	46
5.4. Analysis of Median Ride Duration	48
5.5. Distribution of Bike Usage by User Type	50

5.6.	Weekly Distribution of Ride Duration (New measure using DAX).....	52
5.7.	Data Visualization Dashboard (New measure with DAX)	55
6.	Conclusions and Recommendations.....	56
6.1.	Pricing Information of Bike Sharing Service	56
6.2.	Correlation Between Usage Patterns and Pricing	57
6.3.	Recommendations and Justification.....	58
7.	References	61



BUSINESS INTELLIGENCE DESIGN SCOPE

Divvy Bike Sharing Analysis

Gan Hansong

1. BI Data Source and Requirements

1.1. Introduction to the Dataset

The dataset used in this analysis is sourced from <https://www.kaggle.com/datasets/hswg94/chicago-bike-rental/data> that is originated from Divvy Bikes at <https://divvybikes-marketing-staging.lyft.net/system-data>. Divvy bikes is a popular bike sharing service in Chicago, Illinois. It includes trip related data, such as user type (member or casual), start and end datetimes and the type of bike used (classic or electric). Analyzing this dataset brings an understanding to usage behavior of the bike sharing service and allows to identify key areas that require improvement. Additionally, it serves as a foundation dataset that can be used for performing and conducting data analysis for identifying patterns and trends to create strategic decisions, enhance service quality and understand customer demands.

1.2. Business Objective

The objective of this project is to analyze the data and identify strategies that can aid the increase of non-member to membership conversions and attract new customers to utilize the bike sharing service. In the analysis, the specific questions will be answered and followed by providing recommendations.

- How do annual members and casual riders use the bike service differently?
- Why would casual riders switch to purchase memberships?
- How to influence new customers to utilize the bike sharing service?

1.3. Key Performance Indicators (KPI)

To effectively measure the success of the business objectives, the following KPIs were identified:

- Total Number of Trips - Measures the overall usage of the bike-sharing service.
- Member vs. Casual Trips - Tracks the proportion of trips made by members compared to casual users.
- Ride Duration - Analyzes the typical duration of rides, segmented by user type and bike type.
- Peak Usage Times - Identifies the times of day and days of the week when the service is most utilized.

1.4. Key User Groups

This report may serve useful for various stakeholders within the organization to aid in decision making.

For example, business analysts may use the insights to understand usage patterns and support strategic decision-making, while marketing teams will identify target segments and develop campaigns to increase memberships and attract new users. High-level executives could make high-level strategic decisions based on the insights generated from the report.

1.5. Importing Data Source

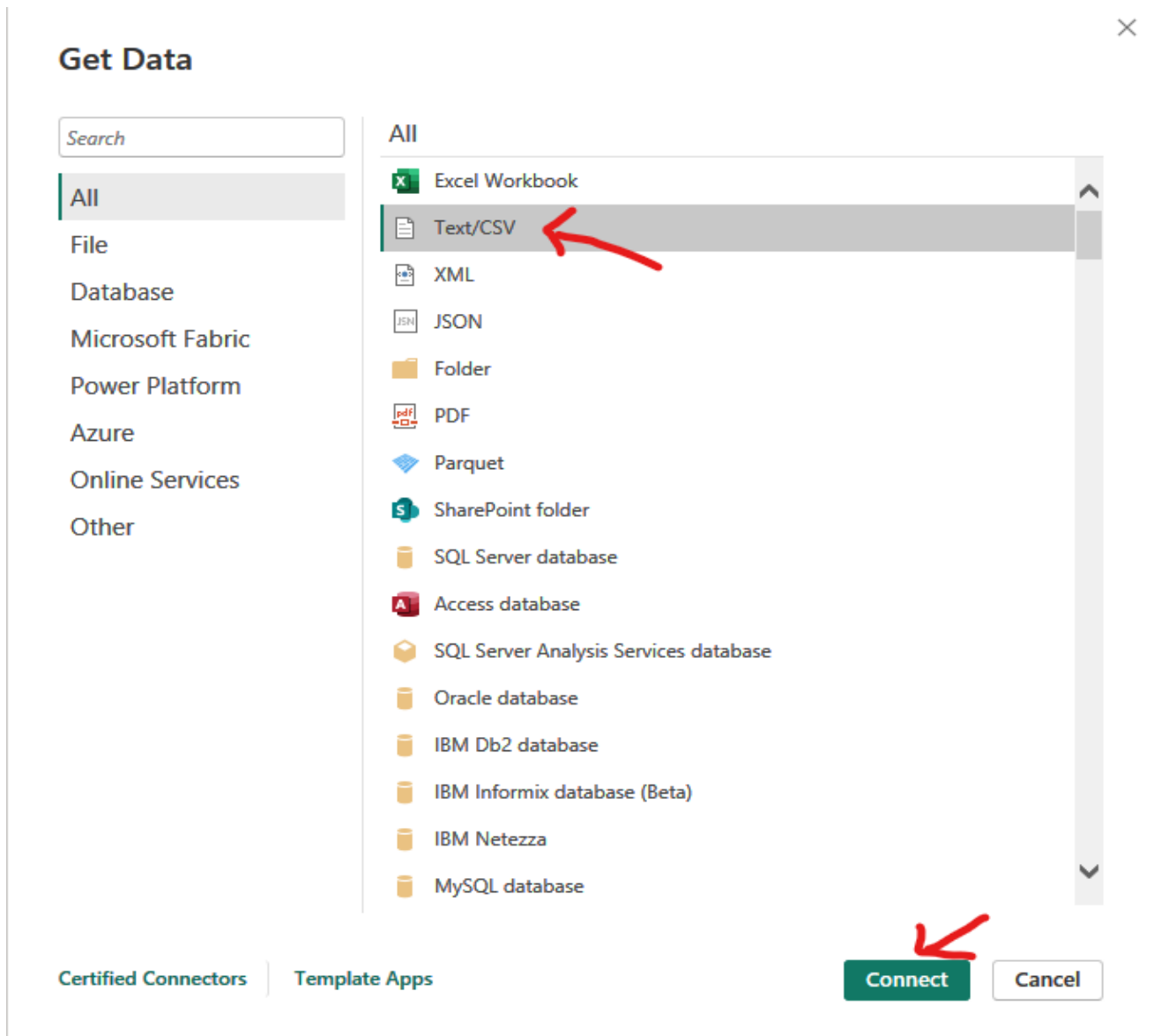


Figure 1 - Importing Data as Text/CSV

The initial steps begin by connecting to the dataset file **chicago_bike_sharing_service.csv** with the steps shown in Figure 1.

chicago_bike_rental_dataset.csv

File Origin

Delimiter

Data Type Detection

1252: Western European (Windows)

Comma

Based on first 200 rows

ride_id	rideable_type	started_at	ended_at	start_station_name	start_station_id	end_station
C2F7DD78E82EC875	electric_bike	13/1/2022 7:59:47 pm	13/1/2022 8:02:44 pm	Glenwood Ave & Touhy Ave	525	Clark St & Touhy
A6CF8980A652D272	electric_bike	10/1/2022 4:41:56 pm	10/1/2022 4:46:17 pm	Glenwood Ave & Touhy Ave	525	Clark St & Touhy
BD0F91DFF741C66D	classic_bike	25/1/2022 12:53:40 pm	25/1/2022 12:58:01 pm	Sheffield Ave & Fullerton Ave	TA1306000016	Greenview Ave & 8
CBB80ED419105406	classic_bike	4/1/2022 8:18:04 am	4/1/2022 8:33:00 am	Clark St & Bryn Mawr Ave	KA1504000151	Paulina St & Mor
DDC963BFDDA51EEA	classic_bike	20/1/2022 9:31:10 am	20/1/2022 9:37:12 am	Michigan Ave & Jackson Blvd	TA1309000002	State St & Rando
A39C6F6CC0586C0B	classic_bike	12/1/2022 2:48:09 am	12/1/2022 2:51:31 am	Wood St & Chicago Ave	637	Honore St & Divi
BDC4AB637EDF981B	classic_bike	31/1/2022 2:32:52 am	31/1/2022 2:49:26 am	Oakley Ave & Irving Park Rd	KA1504000158	Broadway & Shei
81751A3186E59A6B	classic_bike	22/1/2022 8:20:02 pm	22/1/2022 8:32:06 pm	Sheffield Ave & Fullerton Ave	TA1306000016	Damen Ave & Ch
154222B86A338ABD	electric_bike	17/1/2022 3:34:41 pm	17/1/2022 4:00:08 pm	Racine Ave & 15th St	13304	Clinton St & Was
72DC25B2DD467EEF	classic_bike	28/1/2022 11:27:53 pm	28/1/2022 11:35:16 pm	LaSalle St & Jackson Blvd	TA1309000004	Clinton St & Was
F20394FE09C63DB2	classic_bike	12/1/2022 2:27:59 am	12/1/2022 2:34:20 am	LaSalle St & Jackson Blvd	TA1309000004	Clinton St & Was
006D3F363CF9600B	electric_bike	29/1/2022 8:30:43 pm	29/1/2022 8:43:04 pm	Clarendon Ave & Leland Ave	TA1307000119	Broadway & Grai
FB258EEBE89F4E17	classic_bike	3/1/2022 1:56:18 am	3/1/2022 2:05:38 am	Sheffield Ave & Fullerton Ave	TA1306000016	Stockton Dr & W
2EF51270B08DEE03	classic_bike	21/1/2022 6:03:06 am	21/1/2022 6:09:59 am	Rush St & Superior St	15530	Larrabee St & Kir
E119D4E6BF686601	electric_bike	8/1/2022 1:36:40 pm	8/1/2022 1:46:40 pm	Michigan Ave & Jackson Blvd	TA1309000002	St. Clair St & Erie
7DD70C4AA0F3EA7B	classic_bike	14/1/2022 1:47:07 pm	14/1/2022 1:47:39 pm	Lincoln Park Conservatory	LP-	Lincoln Park Con
0AA241497B970400	classic_bike	12/1/2022 10:09:13 pm	12/1/2022 10:10:53 pm	LaSalle St & Jackson Blvd	TA1309000004	Clark St & Ida B V
F5E6D97D85BE8039	classic_bike	31/1/2022 8:07:45 pm	31/1/2022 8:12:48 pm	Michigan Ave & Jackson Blvd	TA1309000002	Indiana Ave & Rc
78168928DF06A497	classic_bike	29/1/2022 1:21:59 am	29/1/2022 1:26:54 am	Michigan Ave & Jackson Blvd	TA1309000002	Indiana Ave & Rc
15C176B55C2BC059	classic_bike	20/1/2022 5:00:40 am	20/1/2022 5:09:56 am	Clark St & Bryn Mawr Ave	KA1504000151	Clark St & Schreil

Extract Table Using Examples

Load

Transform Data

Cancel

Figure 2 - Loading Dataset

After the dataset is connected, load the data. As shown in Figure 2, the **chicago_bike_rental_dataset.csv** file will be previewed, allowing to verify the data structure and content.

1.6. Overview of the Dataset

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
CD3195185A7CF182	classic_bike	26/1/2022 4:14:17 am	26/1/2022 4:18:40 am	University Ave & 57th St	KA1503000071	Ellis Ave & 60th St	KA1503000014	41.791478	-87.599861	41.78509714636	-87.6010727606	member
1A5EAD595A0F926C	classic_bike	26/1/2022 10:48:37 pm	26/1/2022 10:54:37 pm	University Ave & 57th St	KA1503000071	Ellis Ave & 60th St	KA1503000014	41.791478	-87.599861	41.78509714636	-87.6010727606	member
3380F18E48C9DC48	classic_bike	26/1/2022 4:24:45 am	26/1/2022 4:30:02 am	University Ave & 57th St	KA1503000071	Ellis Ave & 60th St	KA1503000014	41.791478	-87.599861	41.78509714636	-87.6010727606	member
A6F36553FF4A8796	classic_bike	26/1/2022 6:39:30 am	26/1/2022 6:43:51 am	University Ave & 57th St	KA1503000071	Ellis Ave & 60th St	KA1503000014	41.791478	-87.599861	41.78509714636	-87.6010727606	member
120E5902173C678A	classic_bike	19/1/2022 12:37:58 am	19/1/2022 12:41:50 am	University Ave & 57th St	KA1503000071	Ellis Ave & 60th St	KA1503000014	41.791478	-87.599861	41.78509714636	-87.6010727606	member
A623647C2DC21738	classic_bike	26/1/2022 8:25:19 pm	26/1/2022 8:29:42 pm	University Ave & 57th St	KA1503000071	Ellis Ave & 60th St	KA1503000014	41.791478	-87.599861	41.78509714636	-87.6010727606	member
66398EA18C3C29D6	classic_bike	26/1/2022 11:50:16 pm	26/1/2022 11:56:43 pm	University Ave & 57th St	KA1503000071	Ellis Ave & 60th St	KA1503000014	41.791478	-87.599861	41.78509714636	-87.6010727606	member
FEA58E39F3012E97	classic_bike	12/1/2022 9:23:22 pm	12/1/2022 9:27:28 pm	University Ave & 57th St	KA1503000071	Ellis Ave & 60th St	KA1503000014	41.791478	-87.599861	41.78509714636	-87.6010727606	member
E9C72A6080AA32E0	classic_bike	19/1/2022 1:38:01 am	19/1/2022 2:23:12 am	University Ave & 57th St	KA1503000071	Ellis Ave & 60th St	KA1503000014	41.791478	-87.599861	41.78509714636	-87.6010727606	member
F06E19C8A47E1103	classic_bike	26/1/2022 6:27:41 am	26/1/2022 6:44:16 am	University Ave & 57th St	KA1503000071	Ellis Ave & 60th St	KA1503000014	41.791478	-87.599861	41.78509714636	-87.6010727606	member
AE58E4D7396E683F	classic_bike	26/1/2022 9:14:03 pm	26/1/2022 9:18:25 pm	University Ave & 57th St	KA1503000071	Ellis Ave & 60th St	KA1503000014	41.791478	-87.599861	41.78509714636	-87.6010727606	member
5734024939211070	classic_bike	26/1/2022 1:33:00 am	26/1/2022 1:37:28 am	University Ave & 57th St	KA1503000071	Ellis Ave & 60th St	KA1503000014	41.791478	-87.599861	41.78509714636	-87.6010727606	member
CD9846D72928121E	classic_bike	12/1/2022 1:09:54 am	12/1/2022 1:15:44 am	University Ave & 57th St	KA1503000071	Ellis Ave & 60th St	KA1503000014	41.791478	-87.599861	41.78509714636	-87.6010727606	member
E7FA8BA764E40886	classic_bike	26/1/2022 4:35:00 am	26/1/2022 4:38:54 am	University Ave & 57th St	KA1503000071	Ellis Ave & 60th St	KA1503000014	41.791478	-87.599861	41.78509714636	-87.6010727606	member
F3EAD078BC97AE55	classic_bike	26/1/2022 1:02:10 am	26/1/2022 1:06:41 am	University Ave & 57th St	KA1503000071	Ellis Ave & 60th St	KA1503000014	41.791478	-87.599861	41.78509714636	-87.6010727606	member
033F630AAFE4A95F	classic_bike	19/1/2022 1:09:14 am	19/1/2022 1:13:22 am	University Ave & 57th St	KA1503000071	Ellis Ave & 60th St	KA1503000014	41.791478	-87.599861	41.78509714636	-87.6010727606	member
82EF5097A1AB901D	classic_bike	12/1/2022 10:31:42 pm	12/1/2022 10:35:14 pm	University Ave & 57th St	KA1503000071	Ellis Ave & 60th St	KA1503000014	41.791478	-87.599861	41.78509714636	-87.6010727606	member
ED1AD879D7997853	classic_bike	19/1/2022 3:14:19 am	19/1/2022 3:18:23 am	University Ave & 57th St	KA1503000071	Ellis Ave & 60th St	KA1503000014	41.791478	-87.599861	41.78509714636	-87.6010727606	member
504F6AD8C84202AA	classic_bike	19/1/2022 8:08:34 am	19/1/2022 8:12:12 am	University Ave & 57th St	KA1503000071	Ellis Ave & 60th St	KA1503000014	41.791478	-87.599861	41.78509714636	-87.6010727606	member
DA4736DE0CD86892	classic_bike	5/1/2022 1:09:53 am	5/1/2022 1:15:54 am	University Ave & 57th St	KA1503000071	Ellis Ave & 60th St	KA1503000014	41.791478	-87.599861	41.78509714636	-87.6010727606	member
6236871294D1DA44	classic_bike	5/1/2022 1:06:15 am	5/1/2022 1:11:12 am	University Ave & 57th St	KA1503000071	Ellis Ave & 60th St	KA1503000014	41.791478	-87.599861	41.78509714636	-87.6010727606	member
A204ABFD56857D80	classic_bike	16/2/2022 6:15:45 am	16/2/2022 6:19:54 am	University Ave & 57th St	KA1503000071	Ellis Ave & 60th St	KA1503000014	41.791478	-87.599861	41.78509714636	-87.6010727606	member
079D540192F51184	classic_bike	16/2/2022 6:15:14 am	16/2/2022 6:20:12 am	University Ave & 57th St	KA1503000071	Ellis Ave & 60th St	KA1503000014	41.791478	-87.599861	41.78509714636	-87.6010727606	member
184CC3922738129A	classic_bike	23/2/2022 8:20:26 pm	23/2/2022 8:24:35 pm	University Ave & 57th St	KA1503000071	Ellis Ave & 60th St	KA1503000014	41.791478	-87.599861	41.78509714636	-87.6010727606	member
0403F066D21F9477	classic_bike	9/2/2022 7:24:14 pm	9/2/2022 7:28:51 pm	University Ave & 57th St	KA1503000071	Ellis Ave & 60th St	KA1503000014	41.791478	-87.599861	41.78509714636	-87.6010727606	member
18F3B9808C903021	classic_bike	16/2/2022 6:36:08 am	16/2/2022 6:40:28 am	University Ave & 57th St	KA1503000071	Ellis Ave & 60th St	KA1503000014	41.791478	-87.599861	41.78509714636	-87.6010727606	member
6A4D80954668F117	classic_bike	16/2/2022 11:31:47 pm	16/2/2022 11:35:33 pm	University Ave & 57th St	KA1503000071	Ellis Ave & 60th St	KA1503000014	41.791478	-87.599861	41.78509714636	-87.6010727606	member
6668A9735436D424	classic_bike	16/2/2022 1:30:02 am	16/2/2022 1:34:55 am	University Ave & 57th St	KA1503000071	Ellis Ave & 60th St	KA1503000014	41.791478	-87.599861	41.78509714636	-87.6010727606	member
C1219A6A8E4D0CAF	classic_bike	23/2/2022 7:21:55 am	23/2/2022 7:25:08 am	University Ave & 57th St	KA1503000071	Ellis Ave & 60th St	KA1503000014	41.791478	-87.599861	41.78509714636	-87.6010727606	member
A8485DCAEC7610DF	classic_bike	2/2/2022 5:29:16 am	2/2/2022 5:32:06 am	University Ave & 57th St	KA1503000071	Ellis Ave & 60th St	KA1503000014	41.791478	-87.599861	41.78509714636	-87.6010727606	member
B7F908A01E3062AF	classic_bike	9/2/2022 5:25:50 am	9/2/2022 5:31:09 am	University Ave & 57th St	KA1503000071	Ellis Ave & 60th St	KA1503000014	41.791478	-87.599861	41.78509714636	-87.6010727606	member
3B9C24DA4C5AD289	classic_bike	2/2/2022 3:58:10 am	2/2/2022 4:01:59 am	University Ave & 57th St	KA1503000071	Ellis Ave & 60th St	KA1503000014	41.791478	-87.599861	41.78509714636	-87.6010727606	member

Figure 3 - Overview of the Dataset

Figure 3 is a preview of the dataset that was loaded, details of the columns can be found in Table 1, including the column name and its description.

Data Description	
Column Name	Description
ride_id	Unique identifier for each ride
rideable_type	Type of bike used for the ride
started_at	Timestamp when the ride started
ended_at	Timestamp when the ride ended
start_station_name	Name of the station where the ride started
start_station_id	Unique identifier for the start station
end_station_name	Name of the station where the ride ended
end_station_id	Unique identifier for the end station
start_lat	Latitude of the start location
start_lng	Longitude of the start location
end_lat	Latitude of the end location
end_lng	Longitude of the end location
member_casual	Type of user (member or casual)

Table 1 - Data Description

2.Data Cleansing and Pre-Processing

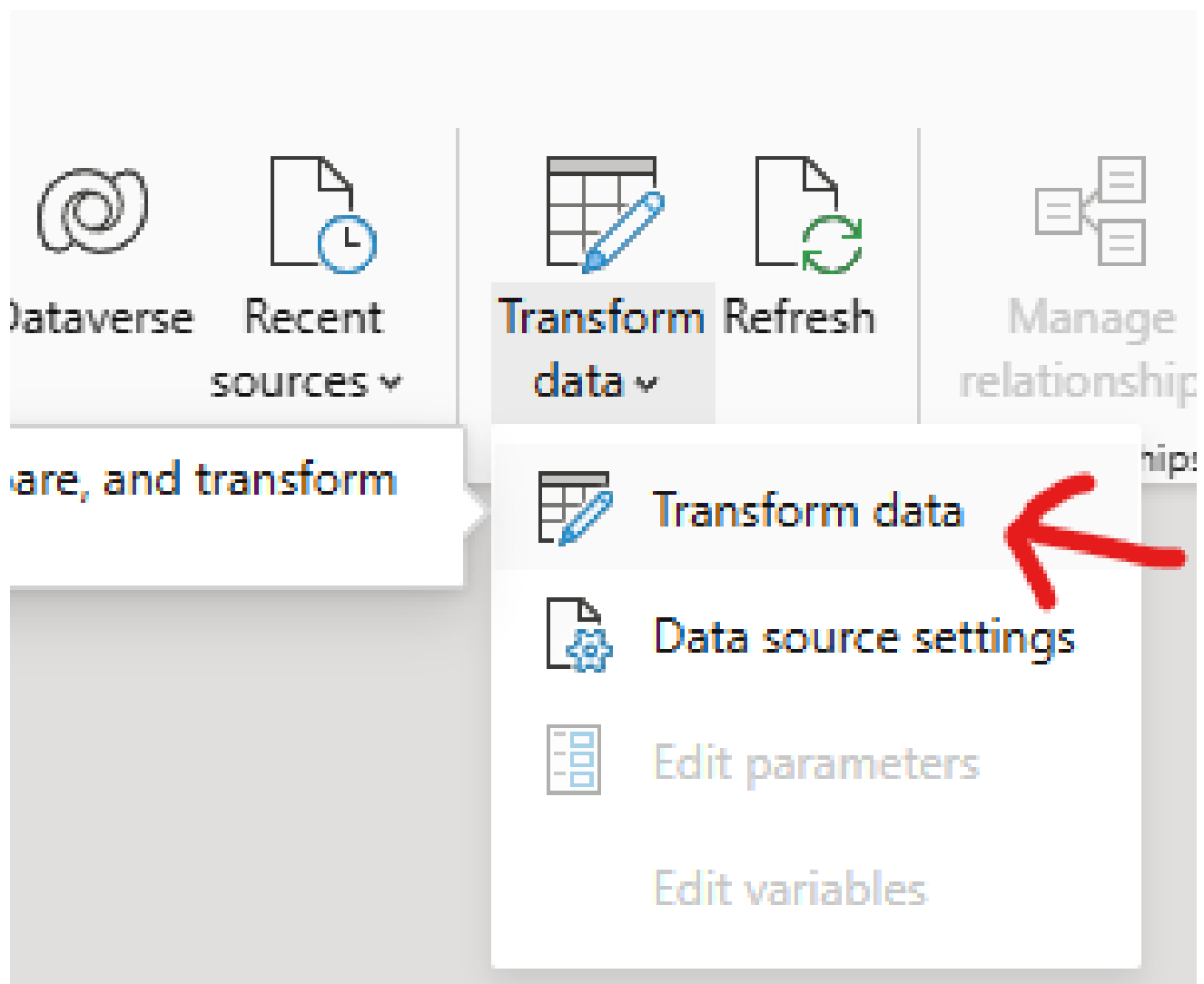
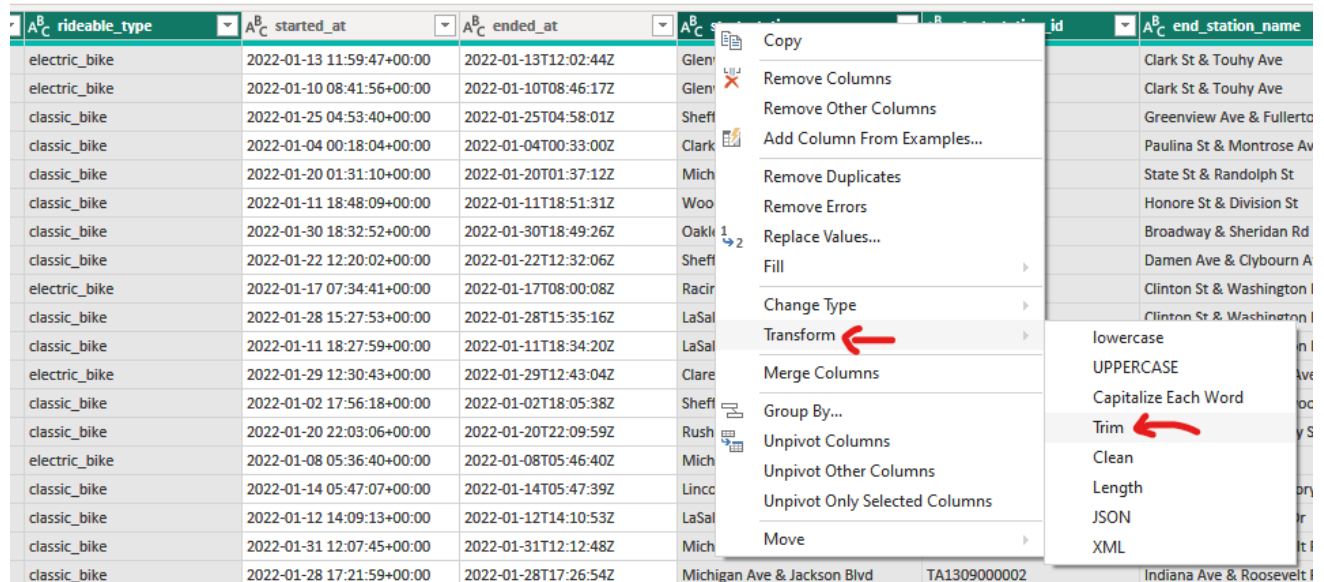


Figure 4 - Transform Data Menu

To start off the data preparation process, the feature 'transform data' in PowerBI as shown in Figure 4 is used, it opens a window that allows allow performing actions related to data preparation, such as data cleansing, and it will be primarily used to prepare the data before performing the steps for data modelling.

The process of data cleansing and pre-processing is to acquire a clean and reliable dataset that is ready for producing accurate analysis and report, the goal is to free it from inconsistencies and errors.

2.1. Trimming the whitespaces of text fields



The screenshot shows a data table with columns: `rideable_type`, `started_at`, `ended_at`, `start_station_name`, `end_station_name`, and `id`. A context menu is open over the `start_station_name` column, with the 'Transform' option selected. The 'Trim' option is highlighted in the submenu. Red arrows point to the 'Transform' and 'Trim' options.

<code>rideable_type</code>	<code>started_at</code>	<code>ended_at</code>	<code>start_station_name</code>	<code>end_station_name</code>	<code>id</code>
electric_bike	2022-01-13 11:59:47+00:00	2022-01-13T12:02:44Z	Glen	Clark St & Touhy Ave	
electric_bike	2022-01-10 08:41:56+00:00	2022-01-10T08:46:17Z	Glen	Clark St & Touhy Ave	
classic_bike	2022-01-25 04:53:40+00:00	2022-01-25T04:58:01Z	Sheff	Greenview Ave & Fullerton	
classic_bike	2022-01-04 00:18:04+00:00	2022-01-04T00:33:00Z	Clark	Paulina St & Montrose Ave	
classic_bike	2022-01-20 01:31:10+00:00	2022-01-20T01:37:12Z	Mich	State St & Randolph St	
classic_bike	2022-01-11 18:48:09+00:00	2022-01-11T18:51:31Z	Woo	Honore St & Division St	
classic_bike	2022-01-30 18:32:52+00:00	2022-01-30T18:49:26Z	Oakl	Broadway & Sheridan Rd	
classic_bike	2022-01-22 12:20:02+00:00	2022-01-22T12:32:06Z	Sheff	Damen Ave & Clybourn Ave	
electric_bike	2022-01-17 07:34:41+00:00	2022-01-17T08:00:08Z	Racine	Clinton St & Washington St	
classic_bike	2022-01-28 15:27:53+00:00	2022-01-28T15:35:16Z	LaSal	Clinton St & Washington St	
classic_bike	2022-01-11 18:27:59+00:00	2022-01-11T18:34:20Z	LaSal		
electric_bike	2022-01-29 12:30:43+00:00	2022-01-29T12:43:04Z	Clare		
classic_bike	2022-01-02 17:56:18+00:00	2022-01-02T18:05:38Z	Sheff		
classic_bike	2022-01-20 22:03:06+00:00	2022-01-20T22:09:59Z	Rush		
electric_bike	2022-01-08 05:36:40+00:00	2022-01-08T05:46:40Z	Mich		
classic_bike	2022-01-14 05:47:07+00:00	2022-01-14T05:47:39Z	Lincoln		
classic_bike	2022-01-12 14:09:13+00:00	2022-01-12T14:10:53Z	LaSal		
classic_bike	2022-01-31 12:07:45+00:00	2022-01-31T12:12:48Z	Mich		
classic_bike	2022-01-28 17:21:59+00:00	2022-01-28T17:26:54Z	Michigan Ave & Jackson Blvd	Indiana Ave & Roosevelt St	TA1309000002

Figure 5 - Trimming Whitespaces

The first step in the data cleaning process involves trimming whitespaces from text fields to remove any leading, trailing, or double spaces within the fields (Figure 5). This helps maintain data consistency by preventing unintended issues with entries being treated as distinct values.

2.2. Removing Blank Rows and Duplicates from the table

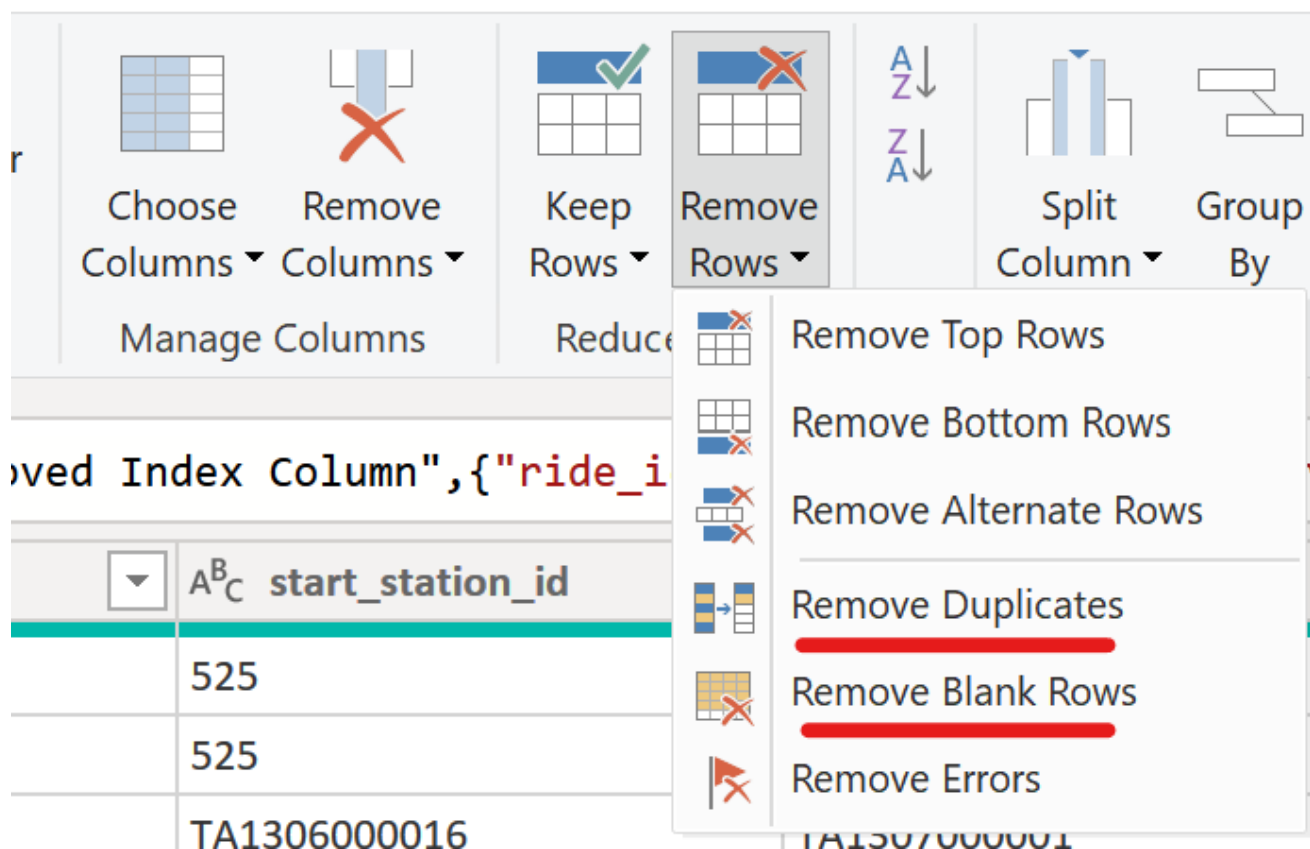


Figure 6 – Removing Blank Rows and Duplicates

To ensure that all entries are complete and to prevent any errors during analysis, blank rows and duplicates were removed from the table, illustrated in Figure 6. This is essential for maintaining the accuracy of any subsequent calculations and analysis.

2.3. Filtering the <ride_id> column (M Language)

```
1 let
2 ... Source = Csv.Document(File.Contents("G:\My Drive\MDIS Study\Databases and Business Intelligence\
3 ... #"Promoted Headers" = Table.PromoteHeaders(Source, [PromoteAllScalars=true]),
4 ... #"Trimmed Text" = Table.TransformColumns(#"Promoted Headers",{{"ride_id", Text.Trim, type text}},
5 ... #"Removed Duplicates" = Table.Distinct(#"Trimmed Text"),
6 ... #"Filtered Rows" = Table.SelectRows(#"Removed Duplicates", each Text.Length([ride_id]) = 16)
7 in
8 ... #"Filtered Rows"
```

Figure 7 – Filtering Column to 16 Characters in Length

The following lines shown in Figure 7 was added in the advanced editor to display only rows where the <ride_id> column contains 16 characters. This ensures that all entries are of the correct length, and to eliminate any invalid entries while maintaining integrity.

2.4. Filtering the <rideable_type> and <member_casual> column

Filter Rows

Apply one or more filter conditions to the rows in this table.

☒ Basic ☐ Advanced

Keep rows where 'rideable_type'

equals classic_bike

☐ And ☒ Or

equals electric_bike

OK Cancel

Figure 8 - Filtering the columns to include only relevant data

Filtering the <rideable_type> column shown in Figure 8 ensures that only the relevant bike types are included in the analysis. Keeping only rows that are either "classic_bike" or "electric_bike" will exclude any irrelevant or dirty data, which could skew the analysis results. Likewise, the <member_casual> column was filtered to include only rows where the field were either "casual" or "member"

2.5. Changing the data type of <started_at> and <ended_at> columns

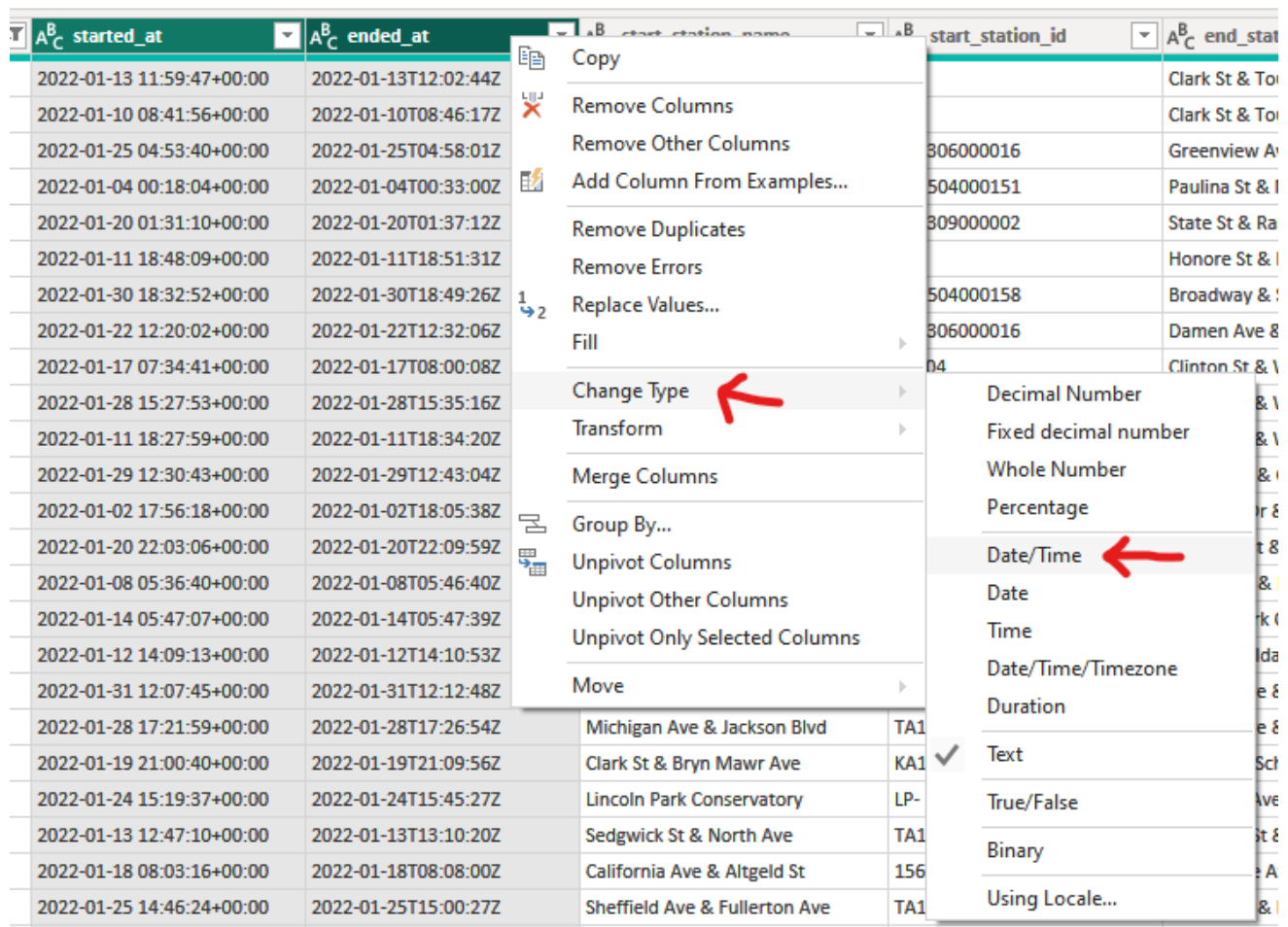


Figure 9 - Changing the data type of <started_at> and <ended_at> columns

To ensure proper representation and functionality of date and time related calculations, the data types of the <started_at> and <ended_at> columns were changed to Date/Time. Figure 9 illustrates this process, which is crucial for extracting day information and calculating ride durations.

2.6. Changing the data type of columns containing coordinates

The screenshot shows a data transformation interface with a table containing coordinate data. The columns are labeled `ABC start_lat`, `ABC start_lng`, `ABC end_lat`, and `ABC end_lng`. A context menu is open over the `ABC end_lng` column, with the 'Change Type' option selected. The 'Decimal Number' option is highlighted in the submenu, indicated by a red arrow. The right sidebar shows 'Query Settings' with 'fact_table' as the source and 'Promoted Headers' and 'Trimmed Text' as applied steps.

<code>A^B_C start_lat</code>	<code>A^B_C start_lng</code>	<code>A^B_C end_lat</code>	<code>A^B_C end_lng</code>	
42.0128005	-87.665906	42.01256011541	-87.674	
42.012763	-87.6659675	42.01256011541	-87.674	
41.9256018819	-87.6537080423	41.92533	-87.665	
41.983593	-87.669154	41.961507	-87.671	
41.87785	-87.62408	41.8846210725793	-87.627	
41.895634	-87.672069	41.903119	-87.673	
41.95434085219	-87.6860796243	41.952833	-87.649	
41.9256018819	-87.6537080423	41.931931	-87.677	
41.8612513333333	-87.6565001666666	41.88338	-87.641	
41.878166	-87.631929	41.88338	-87.641	
41.878166	-87.631929	41.88338	-87.641	
41.967962	-87.6500285	41.9947796884	-87.660	
41.9256018819	-87.6537080423	41.93132	-87.638	
41.89576474564	-87.6259080327	41.897764	-87.642	
41.8778493333333	-87.624056	41.8943451374242	-87.622	
41.9239313113661	-87.6358245313167	41.9239313113661	-87.635	
41.878166	-87.631929	41.8759326655	-87.630	
41.87785	-87.62408	41.867888	-87.625	
41.87785	-87.62408	41.867888	-87.623041	member
41.983593	-87.669154	41.99925182186	-87.6713773393	member
41.9239313113661	-87.6358245313167	41.8839840647265	-87.6246839761734	member
41.911386	-87.638677	41.911386	-87.638677	member
41.9266475	-87.6976608333333	41.9201955620056	-87.6926591992378	casual

Figure 10 - Adding the `day_of_week_name` column using a custom column formula

The data type of the coordinate related columns was changed to better represent the type of data as geographical and mapping related tools may require a decimal format to function as intended, converting the data type is essential to ensure compatibility. Figure 10 illustrates the process of changing the data type to Decimal Number.

2.7. Removing errors from the table

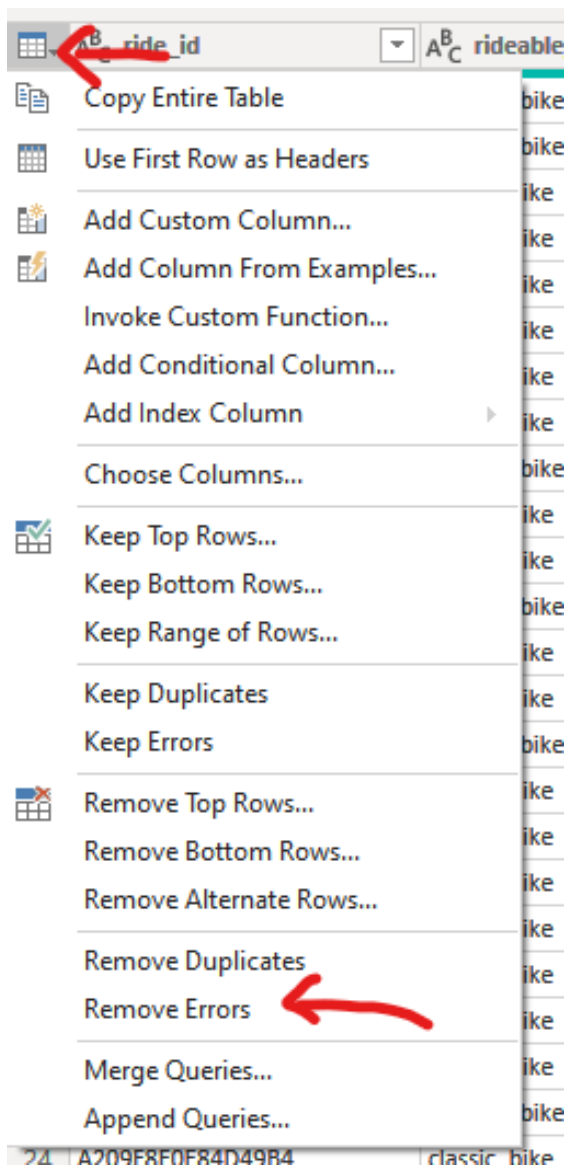


Figure 11 - Removing Errors from the Table

The final step in the data cleansing process is to remove any errors from the table as shown in Figure 11. This step is critical to ensure the integrity and accuracy of the dataset. Errors can occur during data cleaning, such as converting the data types on fields with dirty data. Removing these errors ensures that any issues introduced during the cleansing process are resolved before finalizing the dataset.

3.Data Modelling and Relationship

3.1. Creation of two ID columns in the <trips> table (M Language)

In this step, columns for unique identifiers of the coordinates and datetime information is created to serve its purpose in establishing a clear relationship to the <trips> tables. This step helps in organizing the data effectively and to facilitates future querying and analysis.

```
11 ...#"Removed Errors" = Table.RemoveRowsWithErrors("#Changed Type"),
12
13 ...// Adding coords_id and datetime_id
14 ...#"Index" = Table.AddIndexColumn("#Removed Errors", "Index", 1, 1, Int64.Type),
15 ...#"Coords_ID" = Table.AddColumn("#Index", "coords_id", each "CO" & Text.PadStart(Number.ToText([Index], "X"), 8, "0"), type text),
16 ...#"Datetime_ID" = Table.AddColumn("#Coords_ID", "datetime_id", each "DT" & Text.PadStart(Number.ToText([Index], "X"), 8, "0"), type text),
17 ...#"Removed Index Column" = Table.RemoveColumns("#Datetime_ID",{"Index"})
18 in
19 ...#"Removed Index Column"
```

Figure 12 – Creation of Columns with M Language

In the advanced editor, the highlighted text shown in Figure 12 are lines of code to generates unique IDs, this is to provide a basis for the new IDs.

The <coords_id> field is created by concatenating a custom prefix "CO" with values from the created index column in line 14, the indexes are then converted to hexadecimal and padded onto 8 digits before the concatenation, this is done to create unique identifiers for the coordinate fields.

A similar process was replicated to create the <datetime_id> column using the prefix "DT", shown in line 16

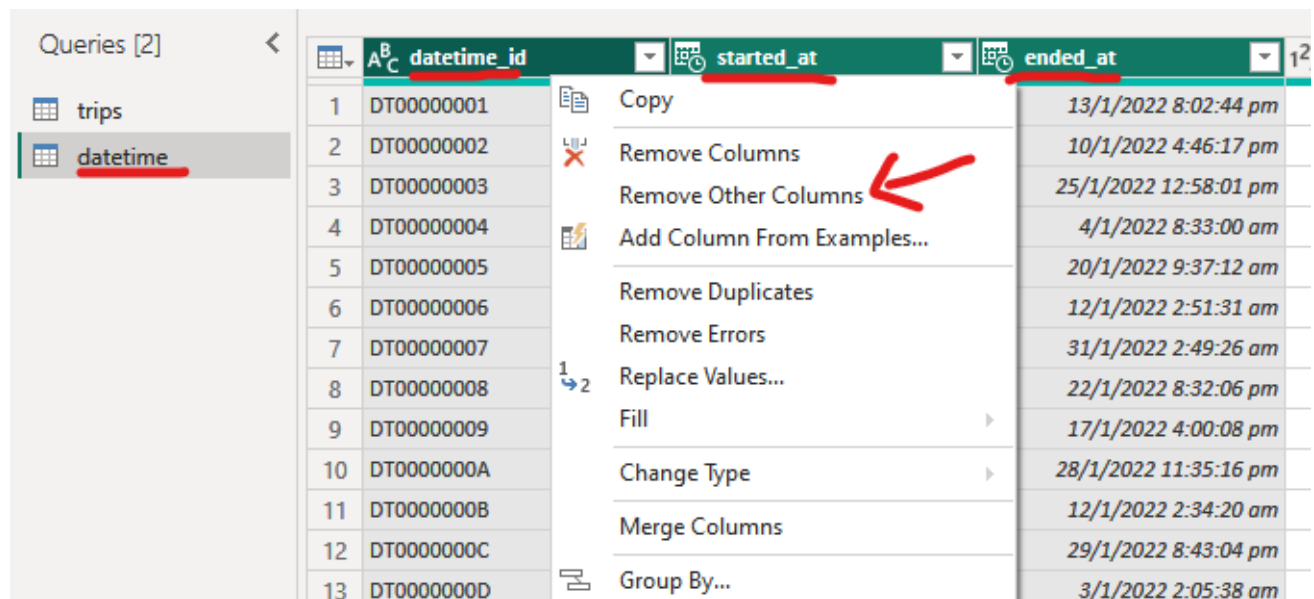
	1.2 end_lat	1.2 end_lng	A ^B _C member_casual	A ^B _C coords_id	A ^B _C datetime_id
6	42.01256012	-87.67436712	casual	CO00000001	DT00000001
5	42.01256012	-87.67436712	casual	CO00000002	DT00000002
4	41.92533	-87.6658	member	CO00000003	DT00000003
4	41.961507	-87.671387	casual	CO00000004	DT00000004
8	41.88462107	-87.62783423	member	CO00000005	DT00000005
9	41.903119	-87.673935	member	CO00000006	DT00000006
2	41.952833	-87.649993	member	CO00000007	DT00000007
4	41.931931	-87.677856	member	CO00000008	DT00000008
7	41.88338	-87.64117	member	CO00000009	DT00000009
9	41.88338	-87.64117	member	CO0000000A	DT0000000A
9	41.88338	-87.64117	member	CO0000000B	DT0000000B
5	41.99477969	-87.66028453	member	CO0000000C	DT0000000C
4	41.93132	-87.638742	member	CO0000000D	DT0000000D
3	41.897764	-87.642884	member	CO0000000E	DT0000000E
6	41.89434514	-87.62279838	casual	CO0000000F	DT0000000F
3	41.92393131	-87.63582453	member	CO00000010	DT00000010
9	41.87593267	-87.63058454	member	CO00000011	DT00000011
8	41.867888	-87.623041	member	CO00000012	DT00000012
8	41.867888	-87.623041	member	CO00000013	DT00000013

Figure 13 - Updated Table with Coordinate and Datetime Identifiers

The highlighted columns in Figure 13, <coords_id> and <datetime_id> were results of the columns created into the <trips> table using M language shown in Figure 12. This concludes the creation of the two IDs.

3.2. Creating the <datetime> table

3.2.1. Duplicating Table and Removing Columns



The screenshot shows a data table with three columns: `datetime_id`, `started_at`, and `ended_at`. The `datetime_id` column is highlighted, and a context menu is open over it. The menu options are: Copy, Remove Columns, Remove Other Columns (indicated by a red arrow), Add Column From Examples..., Remove Duplicates, Remove Errors, Replace Values..., Fill, Change Type, Merge Columns, and Group By... The table contains 13 rows of data, with `datetime_id` values ranging from DT000000001 to DT00000000D.

	<code>datetime_id</code>	<code>started_at</code>	<code>ended_at</code>
1	DT000000001		13/1/2022 8:02:44 pm
2	DT000000002		10/1/2022 4:46:17 pm
3	DT000000003		25/1/2022 12:58:01 pm
4	DT000000004		4/1/2022 8:33:00 am
5	DT000000005		20/1/2022 9:37:12 am
6	DT000000006		12/1/2022 2:51:31 am
7	DT000000007		31/1/2022 2:49:26 am
8	DT000000008		22/1/2022 8:32:06 pm
9	DT000000009		17/1/2022 4:00:08 pm
10	DT00000000A		28/1/2022 11:35:16 pm
11	DT00000000B		12/1/2022 2:34:20 am
12	DT00000000C		29/1/2022 8:43:04 pm
13	DT00000000D		3/1/2022 2:05:38 am

Figure 14 - Duplicating the <trips> Table and Removing Columns

The datetime table is created to separate the date and time related information of trips into a dedicated table to help reduce redundancy and improve efficiency of queries. The trips table was duplicated, with all other columns removed, except the columns, this process is shown in Figure 14.

3.2.2. Adding <duration_sec> column (custom column formula).

Custom Column

Add a column that is computed from the other columns.

New column name
duration_sec

Custom column formula ⓘ
= Duration.TotalSeconds([ended_at]-[started_at])

Available columns
datetime_id
started_at
ended_at

<< Insert

[Learn about Power Query formulas](#)

✓ No syntax errors have been detected.

OK Cancel

Figure 15 - Adding the <duration_sec> Column

Figure 15 demonstrates the addition of a custom column named <duration_sec> using a custom column. The formula calculates the duration of each ride in seconds by subtracting the started_at timestamp from the ended_at timestamp and converting the result to seconds using the “Duration.TotalSeconds()” function. The purpose of this <duration_sec> column is to provide a measurement of the duration of each trip in seconds.

3.2.3. Adding <day_of_week> column (custom column formula).

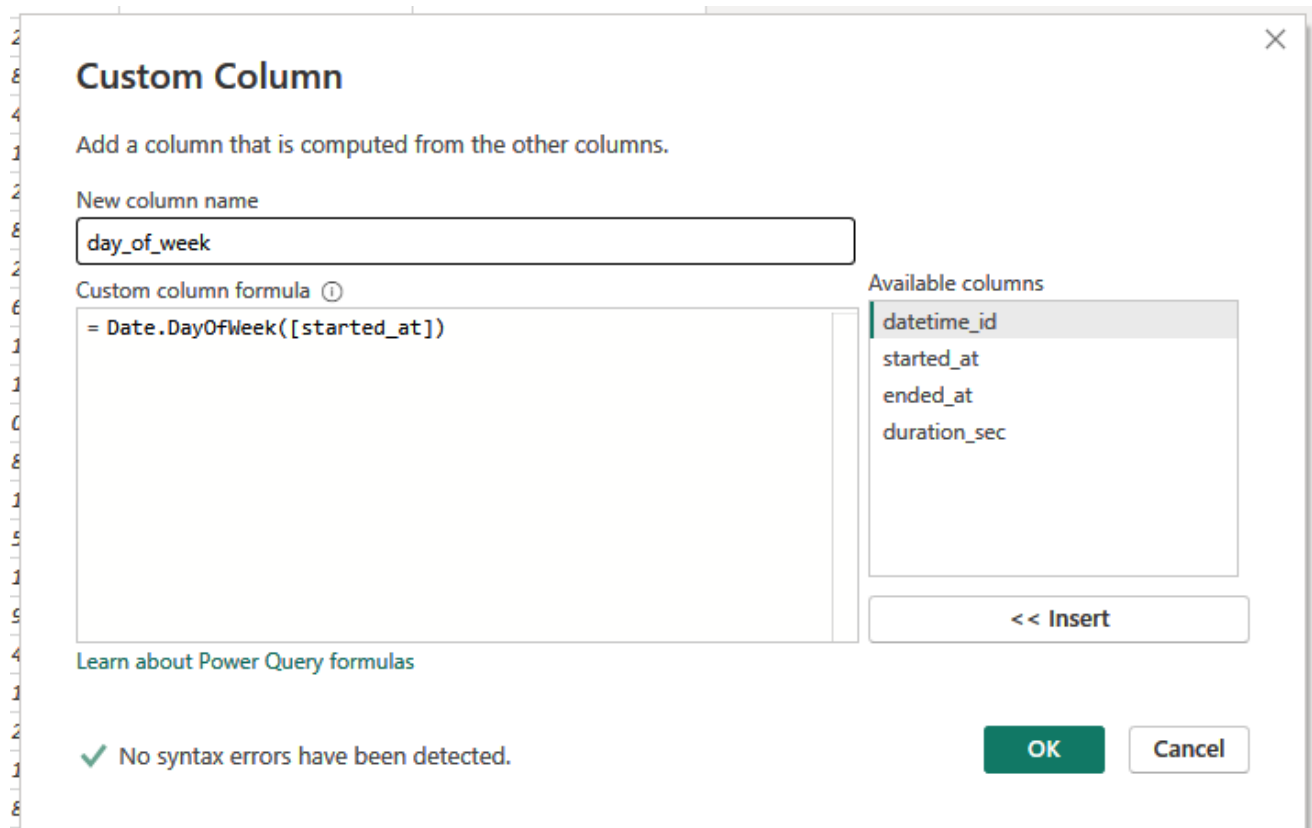


Figure 16 - Adding the <day_of_week> Column

The image shown in Figure 16 is the step taken to produce a custom column named <day_of_week> using a custom column formula. The formula “Date.DayOfWeek([started_at])” is used to extract the day of the week from the “started_at” timestamp. The purpose of this <day_of_week> column is to identify the day of the week for the start of each trip.

3.2.4. Changing data type of <duration_sec> and <day_of_week> columns.

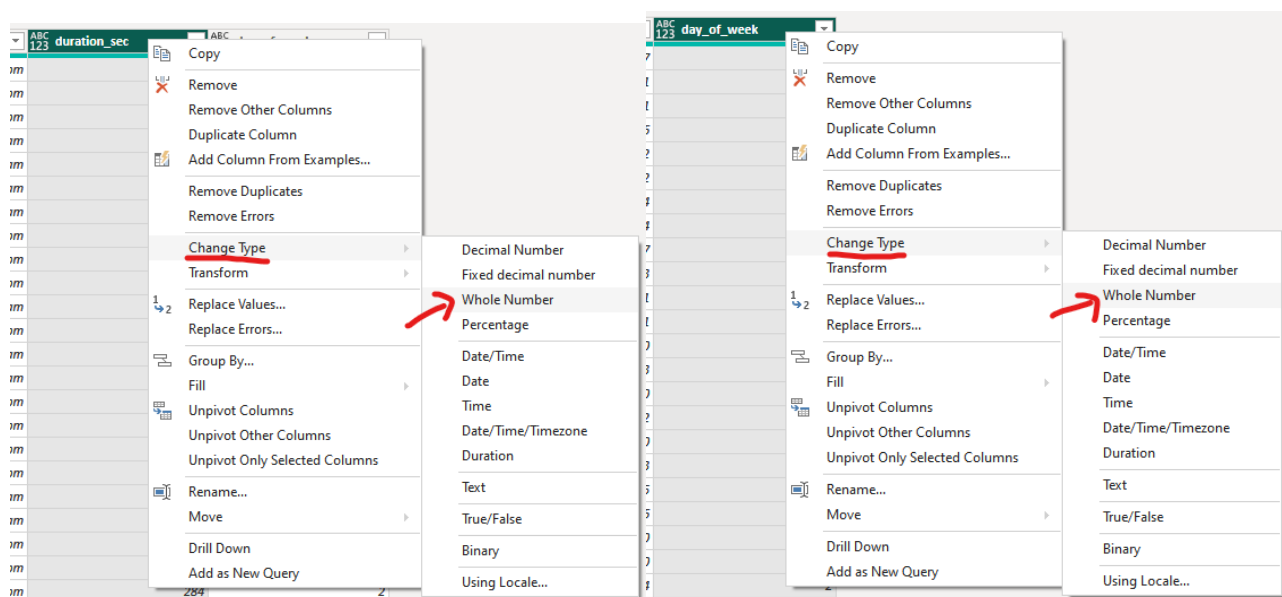


Figure 17 - Changing Data Type of <duration_sec> and <day_of_week>

The image in Figure 17 illustrates the process of changing the data types of the <duration_sec> and <day_of_week> columns. The <duration_sec> is changed to store the duration which are calculated in seconds as integers. The <day_of_week> column is changed to ensure that the day of the week is stored as an integer, with values ranging from 0 (Sunday) to 6 (Saturday). This is to facilitate compatibility with datetime related data processing and analysis functions, such as data sorting that expect these values to be in integer format.

3.2.5. Adding <day_of_week_name> column (custom column formula).

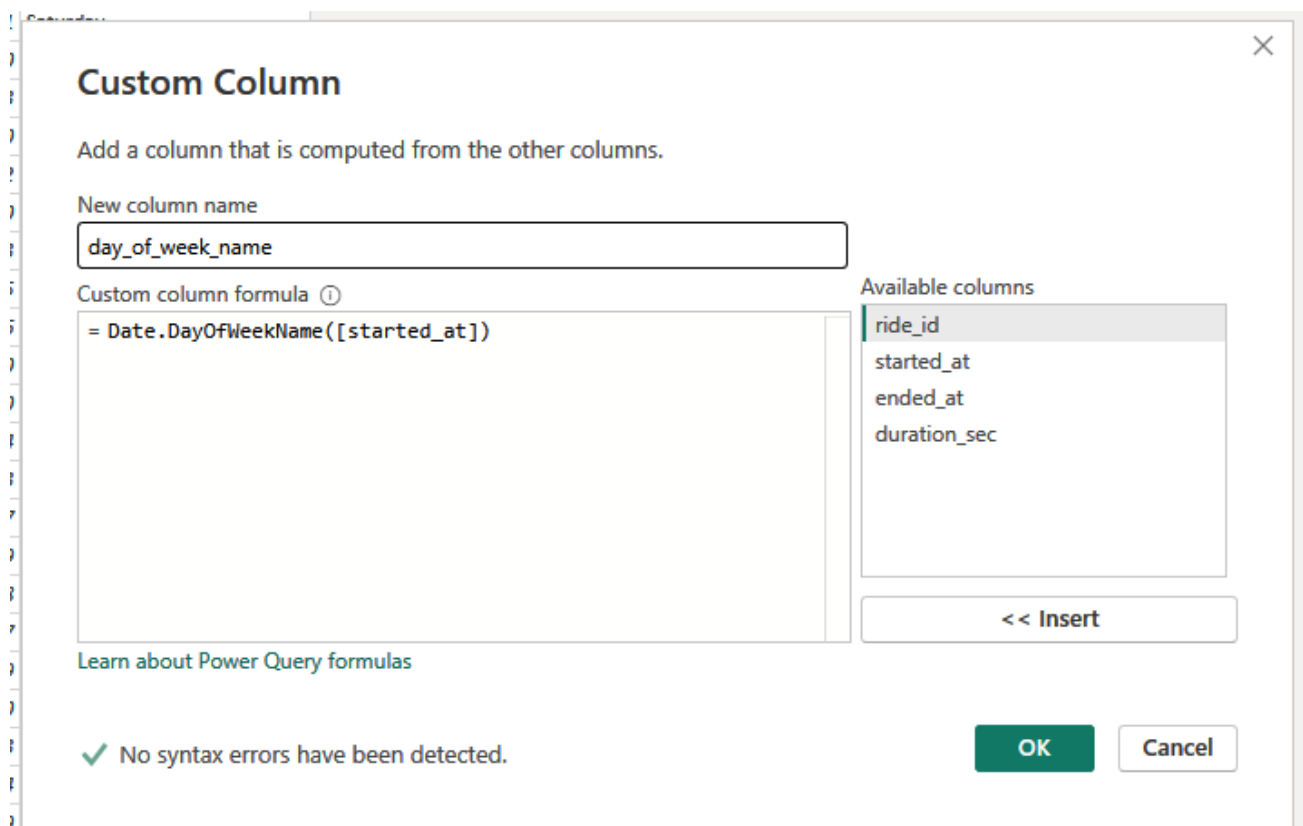


Figure 18 - Adding the <day_of_week_name> Column

The image in Figure 18 shows the addition of a custom column named <day_of_week_name> using a custom column formula. The formula “Date.DayOfWeekName([started_at])” is used to extract the name of the day of the week from the <started_at> column.

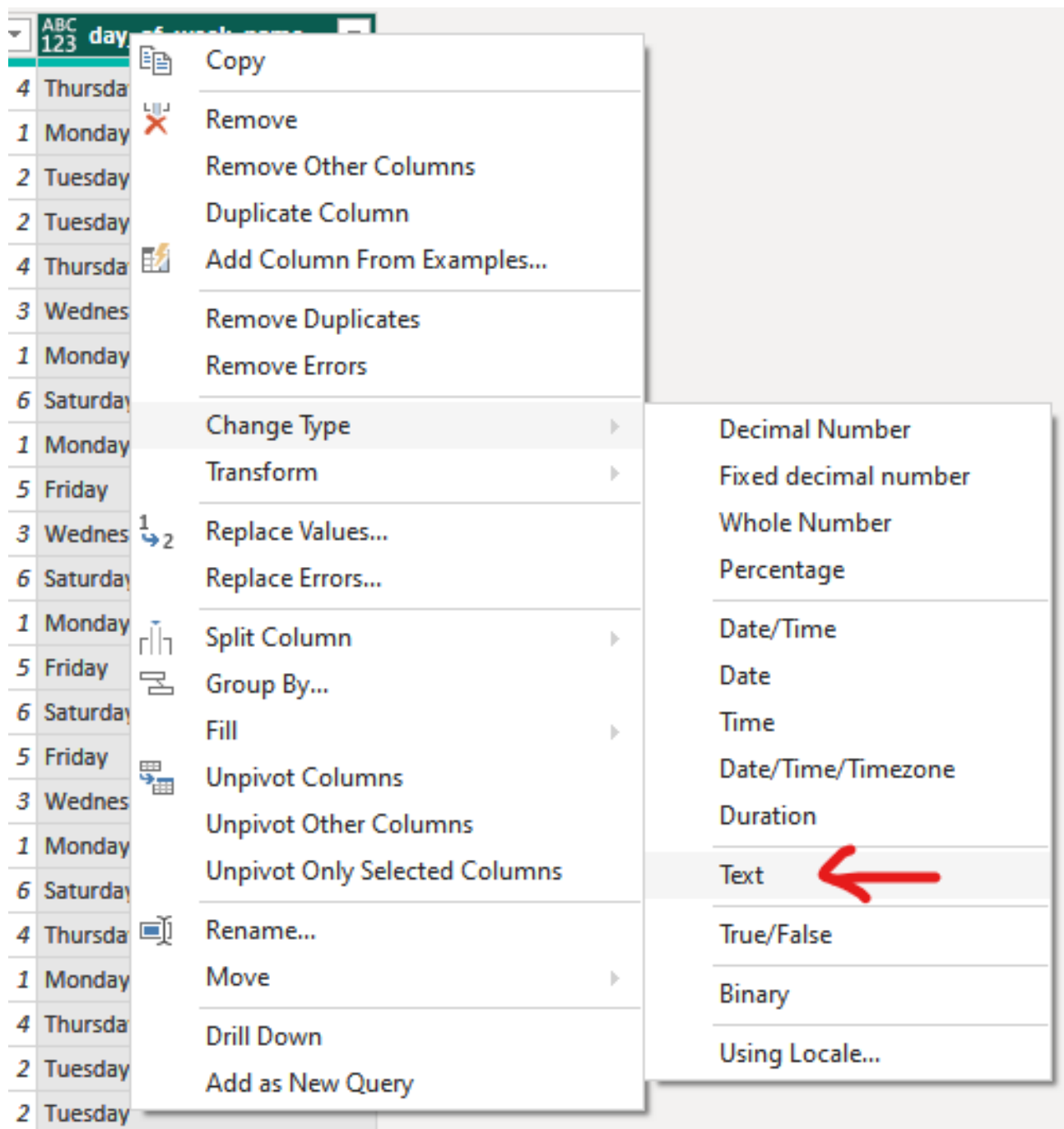


Figure 19 - Changing Data Type of <day_of_week_name> Column

Figure 19 demonstrates the process of changing the data type of the <day_of_week_name> column to "Text" to better represent the data as day names are treated as text strings rather than any other data type, which aligns with their nature as names.

3.2.6. Overview of the <datetime> table

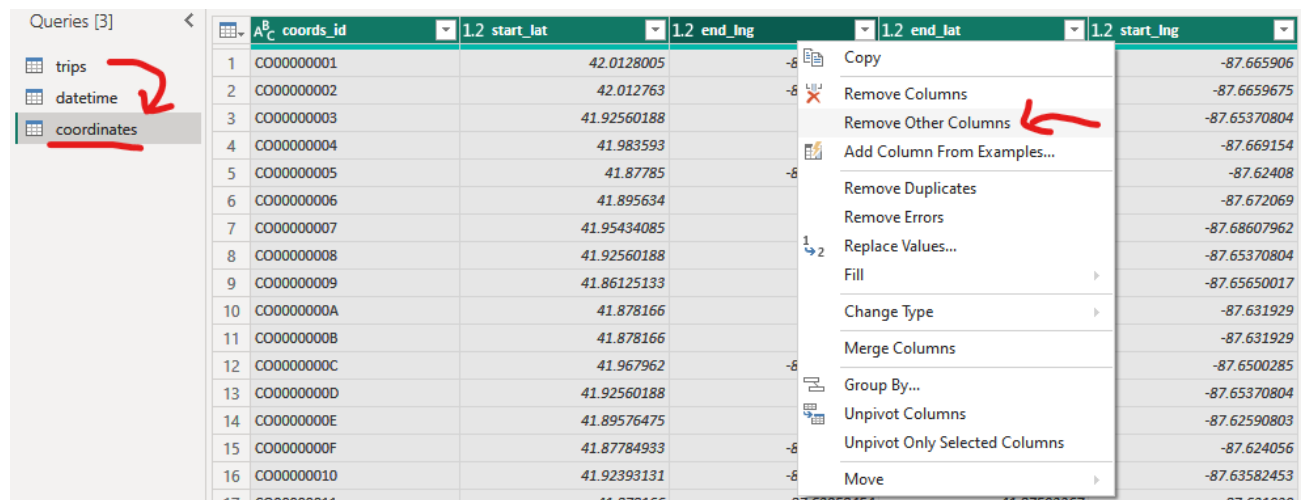
	datetime_id	started_at	ended_at	duration_sec	day_of_week	day_of_week_name
1	DT00000001	13/1/2022 7:59:47 pm	13/1/2022 8:02:44 pm	177	4	Thursday
2	DT00000002	10/1/2022 4:41:56 pm	10/1/2022 4:46:17 pm	261	1	Monday
3	DT00000003	25/1/2022 12:53:40 pm	25/1/2022 12:58:01 pm	261	2	Tuesday
4	DT00000004	4/1/2022 8:18:04 am	4/1/2022 8:33:00 am	896	2	Tuesday
5	DT00000005	20/1/2022 9:31:10 am	20/1/2022 9:37:12 am	362	4	Thursday
6	DT00000006	12/1/2022 2:48:09 am	12/1/2022 2:51:31 am	202	3	Wednesday
7	DT00000007	31/1/2022 2:32:52 am	31/1/2022 2:49:26 am	994	1	Monday
8	DT00000008	22/1/2022 8:20:02 pm	22/1/2022 8:32:06 pm	724	6	Saturday
9	DT00000009	17/1/2022 3:34:41 pm	17/1/2022 4:00:08 pm	1527	1	Monday
10	DT0000000A	28/1/2022 11:27:53 pm	28/1/2022 11:35:16 pm	443	5	Friday
11	DT0000000B	12/1/2022 2:27:59 am	12/1/2022 2:34:20 am	381	3	Wednesday
12	DT0000000C	29/1/2022 8:30:43 pm	29/1/2022 8:43:04 pm	741	6	Saturday
13	DT0000000D	3/1/2022 1:56:18 am	3/1/2022 2:05:38 am	560	1	Monday
14	DT0000000E	21/1/2022 6:03:06 am	21/1/2022 6:09:59 am	413	5	Friday
15	DT0000000F	8/1/2022 1:36:40 pm	8/1/2022 1:46:40 pm	600	6	Saturday
16	DT00000011	12/1/2022 10:09:13 pm	12/1/2022 10:10:53 pm	100	3	Wednesday
17	DT00000012	31/1/2022 8:07:45 pm	31/1/2022 8:12:48 pm	303	1	Monday
18	DT00000013	29/1/2022 1:21:59 am	29/1/2022 1:26:54 am	295	6	Saturday
19	DT00000014	20/1/2022 5:00:40 am	20/1/2022 5:09:56 am	556	4	Thursday
20	DT00000015	24/1/2022 11:19:37 pm	24/1/2022 11:45:27 pm	1550	1	Monday

Figure 20 - Overview of the <datetime> Table

Figure 20 provides an overview of the <datetime> table from the outcome of the data transformation process. The table is now structured in a way to facilitate data analysis, such as identifying trends based on the day of the week, calculating average ride durations etc.

3.3. Creating the <coordinates> Table

3.3.1. Duplicating <trips> table and Removing Columns



The screenshot shows a data table with columns: A^BC coords_id, 1.2 start_lat, 1.2 end_lng, 1.2 end_lat, and 1.2 start_lng. A context menu is open over the table, with the 'Remove Other Columns' option highlighted by a red arrow. Another red arrow points to the 'coordinates' table in the 'Queries [3]' sidebar on the left.

	A ^B C coords_id	1.2 start_lat	1.2 end_lng	1.2 end_lat	1.2 start_lng
1	CO00000001	42.0128005	-87.67436712	42.01256012	-87.665906
2	CO00000002	42.012763	-87.67436712	42.01256012	-87.6659675
3	CO00000003	41.92560188	-87.6658	41.92533	-87.65370804
4	CO00000004	41.983593	-87.671387	41.961507	-87.669154
5	CO00000005	41.87785	-87.62783423	41.88462107	-87.62408
6	CO00000006	41.895634	-87.673935	41.903119	-87.672069
7	CO00000007	41.95434085	-87.649993	41.952833	-87.68607962
8	CO00000008	41.92560188	-87.677856	41.931931	-87.65370804
9	CO00000009	41.86125133	-87.64117	41.88338	-87.65650017
10	CO0000000A	41.878166	-87.64117	41.88338	-87.631929
11	CO0000000B	41.878166	-87.64117	41.88338	-87.631929
12	CO0000000C	41.967962	-87.66028453	41.99477969	-87.6500285
13	CO0000000D	41.92560188	-87.638742	41.93132	-87.65370804
14	CO0000000E	41.89576475	-87.642884	41.897764	-87.62590803
15	CO0000000F	41.87784933	-87.62279838	41.89434514	-87.624056

Figure 21 - Duplicating <trips> table and Removing Columns

Figure 21 shows the process of creating a <coordinates> table by duplicating the <trips> table and removing unnecessary columns.

3.3.2. Overview of the <coordinates> table

	A ^B C coords_id	1.2 start_lat	1.2 end_lng	1.2 end_lat	1.2 start_lng
1	CO00000001	42.0128005	-87.67436712	42.01256012	-87.665906
2	CO00000002	42.012763	-87.67436712	42.01256012	-87.6659675
3	CO00000003	41.92560188	-87.6658	41.92533	-87.65370804
4	CO00000004	41.983593	-87.671387	41.961507	-87.669154
5	CO00000005	41.87785	-87.62783423	41.88462107	-87.62408
6	CO00000006	41.895634	-87.673935	41.903119	-87.672069
7	CO00000007	41.95434085	-87.649993	41.952833	-87.68607962
8	CO00000008	41.92560188	-87.677856	41.931931	-87.65370804
9	CO00000009	41.86125133	-87.64117	41.88338	-87.65650017
10	CO0000000A	41.878166	-87.64117	41.88338	-87.631929
11	CO0000000B	41.878166	-87.64117	41.88338	-87.631929
12	CO0000000C	41.967962	-87.66028453	41.99477969	-87.6500285
13	CO0000000D	41.92560188	-87.638742	41.93132	-87.65370804
14	CO0000000E	41.89576475	-87.642884	41.897764	-87.62590803
15	CO0000000F	41.87784933	-87.62279838	41.89434514	-87.624056

Figure 22 - Overview of the <coordinates> Table

This outcome of the <coordinates> table is shown in Figure 22, it contains the geographical coordinates associated with the start and end points of each ride.

3.4. Creating the <stations> table

3.4.1. Duplicating the <fact_table> and rename to <stations>

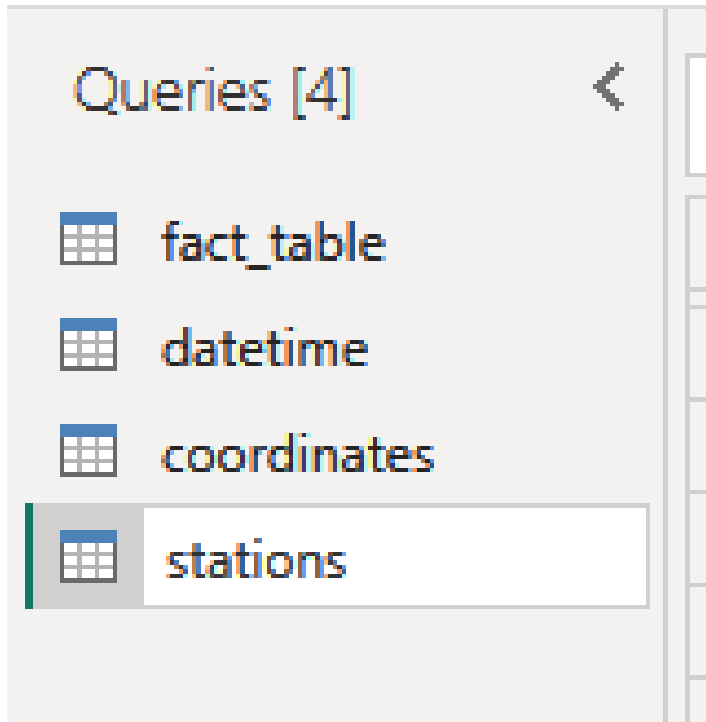


Figure 23 - Duplicating and Renaming the <fact_table>

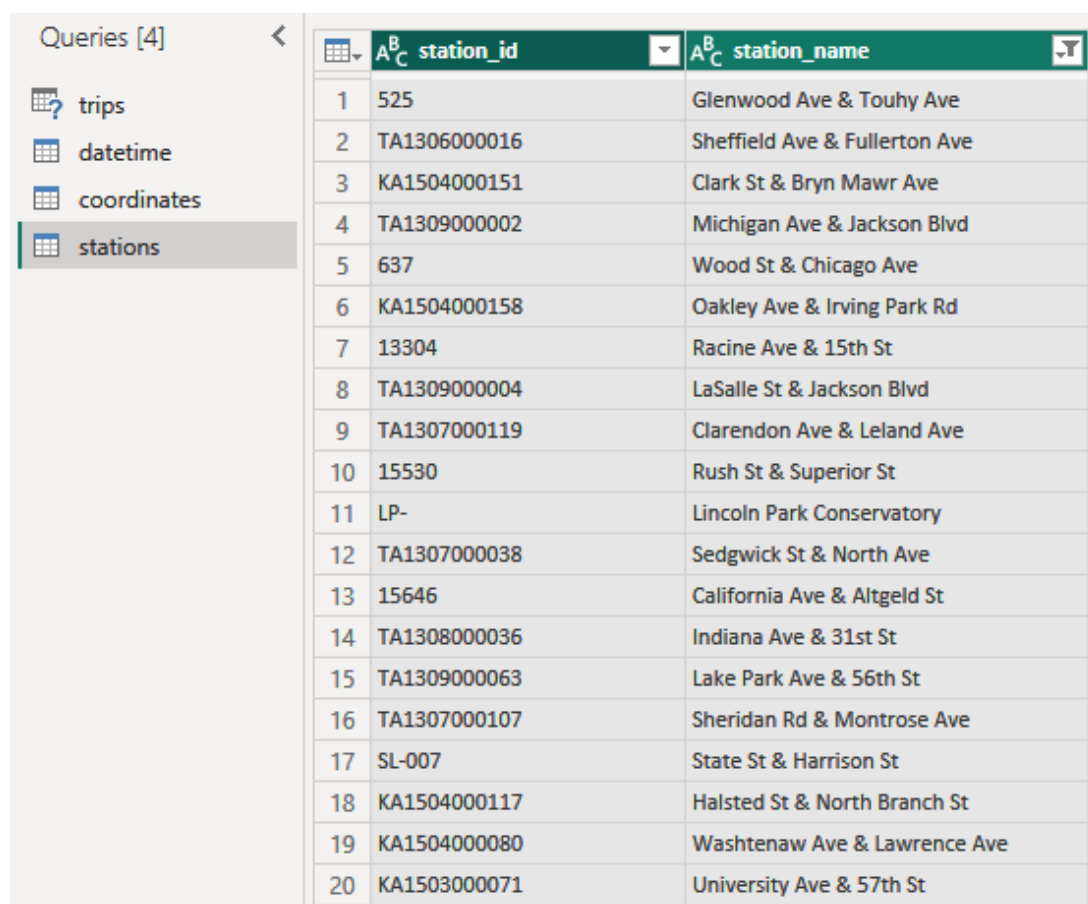
Figure 23 illustrates the initial step in creating the <stations> table by duplicating and renaming the <fact_table>. This goal of this table is to provide unique records for each station, this table provides unique station IDs and its corresponding name. The purpose of this table is to reduce data redundancy and improve data integrity

3.4.2. Constructing the <stations> table (M Language)

```
19 .....//Create the StartStations and EndStations table
20 .....StartStations = Table.SelectColumns(#"Removed Index Column", {"start_station_id", "start_station_name"}),
21 .....EndStations = Table.SelectColumns(#"Removed Index Column", {"end_station_id", "end_station_name"}),
22
23 .....//Rename the Columns of the Start and End Stations table
24 .....RenamedStartStations = Table.RenameColumns(StartStations, {"start_station_id", "station_id"}, {"start_station_name", "station_name"}),
25 .....RenamedEndStations = Table.RenameColumns(EndStations, {"end_station_id", "station_id"}, {"end_station_name", "station_name"}),
26
27 .....//Combine both StartStations and EndStations tables
28 .....CombinedStations = Table.Combine({RenamedStartStations, RenamedEndStations}),
29
30 .....//Remove duplicates to ensure each station appears only once
31 .....stationsList = Table.Distinct(CombinedStations)
32 in
33 .....#"stationsList"
```

Figure 24 – Using M Language to construct the <stations> table

Figure 24 shows the M Language code used to construct the <stations> table. First, columns for start and end stations are selected from the original table and renamed to have a consistent naming convention for station_id and station_name. Next, the renamed start and end stations columns are appended together. Finally, duplicates are removed.

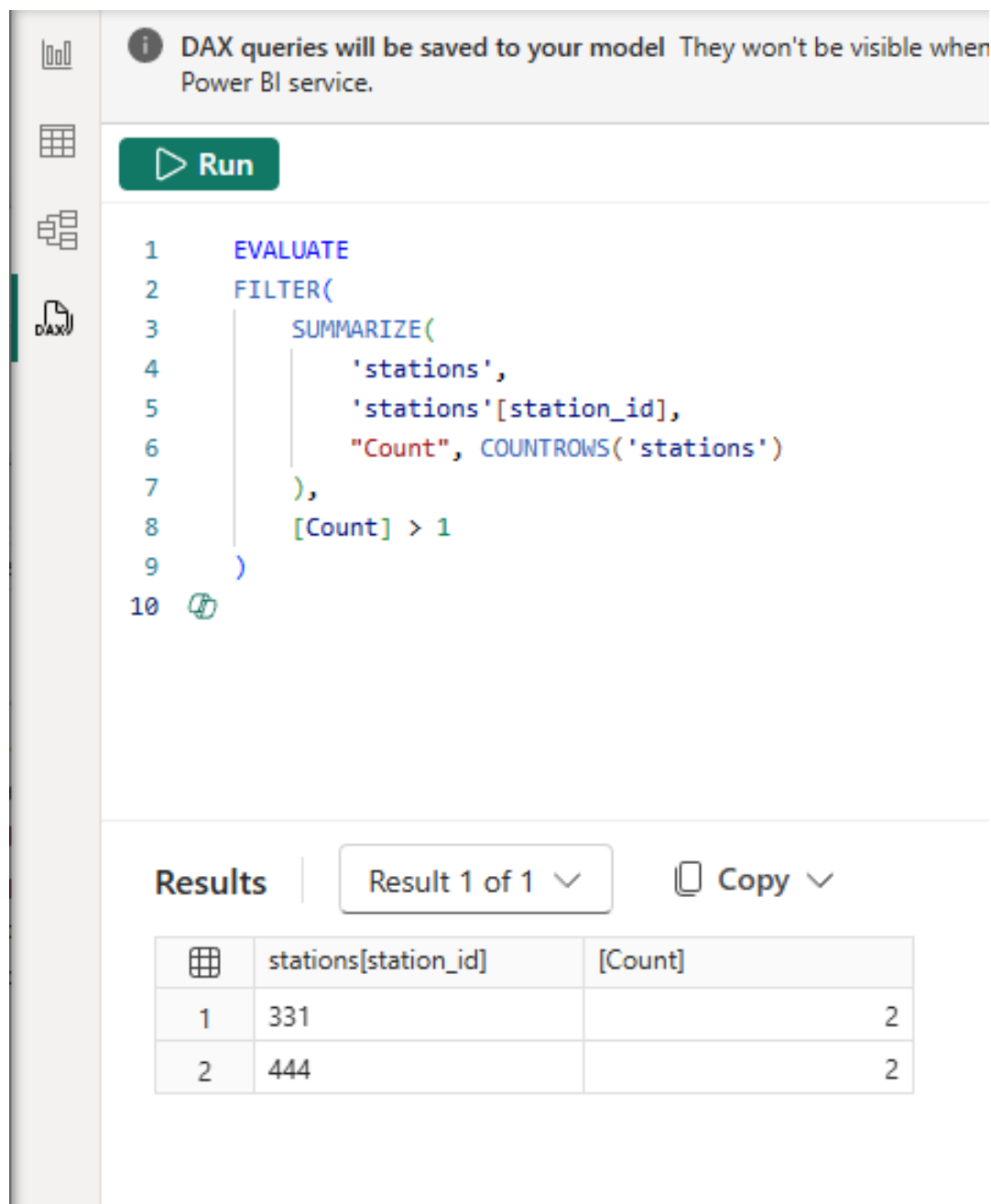


	station_id	station_name
1	525	Glenwood Ave & Touhy Ave
2	TA1306000016	Sheffield Ave & Fullerton Ave
3	KA1504000151	Clark St & Bryn Mawr Ave
4	TA1309000002	Michigan Ave & Jackson Blvd
5	637	Wood St & Chicago Ave
6	KA1504000158	Oakley Ave & Irving Park Rd
7	13304	Racine Ave & 15th St
8	TA1309000004	LaSalle St & Jackson Blvd
9	TA1307000119	Clarendon Ave & Leland Ave
10	15530	Rush St & Superior St
11	LP-	Lincoln Park Conservatory
12	TA1307000038	Sedgwick St & North Ave
13	15646	California Ave & Altgeld St
14	TA1308000036	Indiana Ave & 31st St
15	TA1309000063	Lake Park Ave & 56th St
16	TA1307000107	Sheridan Rd & Montrose Ave
17	SL-007	State St & Harrison St
18	KA1504000117	Halsted St & North Branch St
19	KA1504000080	Washtenaw Ave & Lawrence Ave
20	KA1503000071	University Ave & 57th St

Figure 25 - Result of <stations> Table with M Language Script

Figure 25 shows the outcome after running the M Language script to construct the <stations> table. The resulting table includes columns for station_id and station_name, that should display unique records for each station.

3.4.3. Ensuring the values in “station_id” column is unique (DAX Queries)



The screenshot shows the Power BI DAX query editor. At the top, a message states: "DAX queries will be saved to your model. They won't be visible when Power BI service." Below this is a green "Run" button. The DAX query is as follows:

```
1  EVALUATE
2  FILTER(
3      SUMMARIZE(
4          'stations',
5          'stations'[station_id],
6          "Count", COUNTROWS('stations')
7      ),
8      [Count] > 1
9  )
10
```

Below the query editor, the "Results" section shows "Result 1 of 1" and a "Copy" button. The results are displayed in a table:

	stations[station_id]	[Count]
1	331	2
2	444	2

Figure 26 – Dax Query for identifying non unique station id values

Before finalizing the creation of the <stations> table, it is crucial to identify any non-unique IDs to prevent potential errors. The following DAX query shown in Figure 26 was used to discover any non-unique values in the station_id column. The query results reveal that station_id values "331" and "444" each have two entries, the duplicates will need to be investigated and resolved. Having unique station IDs is crucial to maintain data integrity and to allow forming a relationship with the fact table <trips>.

3.4.4. Investigating duplicate station_id <331>

A ^B _C end_station_name	A ^B _C end_station_id	A ^B _C start_lat	A ^B _C start_lng	A ^B _C end_lat	A ^B _C end_lng
Halsted St & Clybourn Ave	331	41.8902403333333	-87.6343503333333	41.909668	-87.648128
Halsted St & Clybourn Ave	331	41.889914393	-87.634370208	41.909668	-87.648128
Pulaski Rd & 21st St	331	41.84	-87.73	41.85	-87.72

Figure 27 – Investigating the names and coordinates of station ID 331

A ^B _C station_id	A ^B _C station_name
1 331	Halsted St & Clybourn Ave
2 331	Pulaski Rd & 21st St

Figure 28 – Station Names that are found under Station ID 331

Replace Values

Replace one value with another in the selected columns.

Value To Find

Replace With

Advanced options

OK Cancel

Figure 29 – Merging the Station Names

During the investigation of the duplicate station_id <331>, it was observed that the coordinates for "Halsted St & Clybourn Ave" and "Pulaski Rd & 21st St" are in proximity (shown in Figure 27), this likely have led to a glitch that may have caused them to be assigned the same station_id. To correct this, the two locations are combined under a single ID, with the name updated to "Halsted St & Clybourn Ave, Pulaski Rd & 21st St", this step is shown in Figure 29 to accurately reflect both locations. The redundant record was then removed in section [3.4.6](#).

3.4.5. Investigating duplicate station_id <444> (DAX Queries)

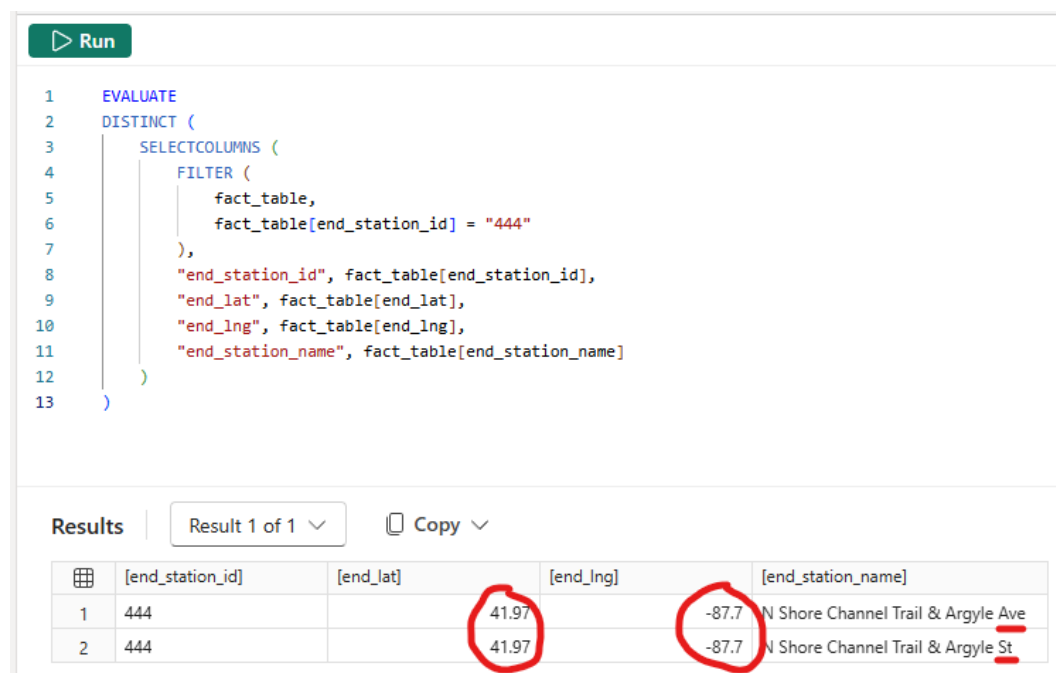


Figure 30 – Dax Query for identifying non unique station id values

The following DAX Query shown in Figure 30 is executed to identify the station names and coordinates associated with station_id <444>. The findings indicate that there is a high likelihood for the error in the name as the coordinates are nearly identical.

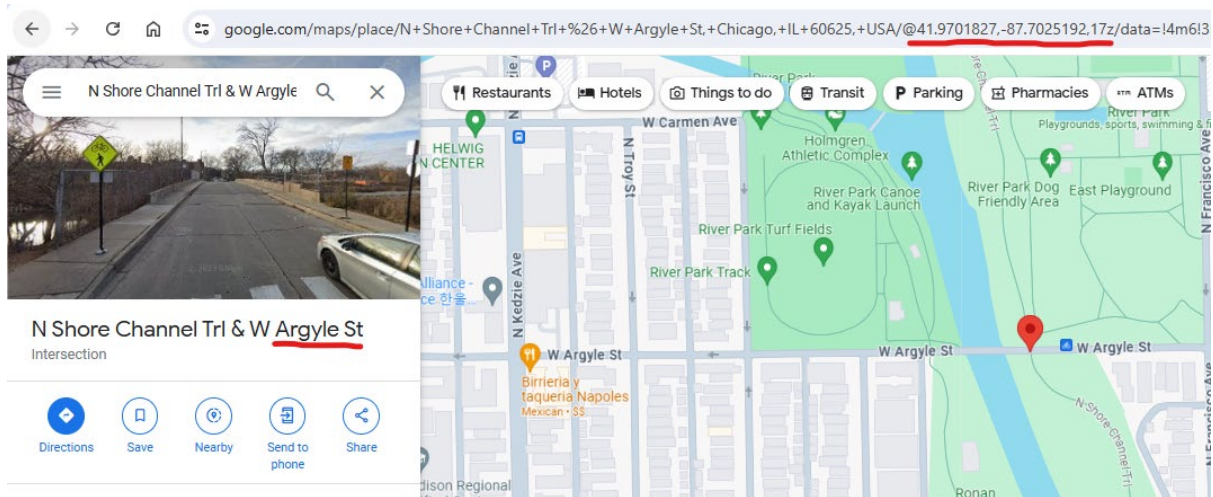


Figure 31 – Identifying the correct station name

The coordinates placed in google maps shows the location pointing to Argyle St, which should be the correct name of the station as shown in Figure 31.

3.4.6. Removing the two invalid records from <stations> table

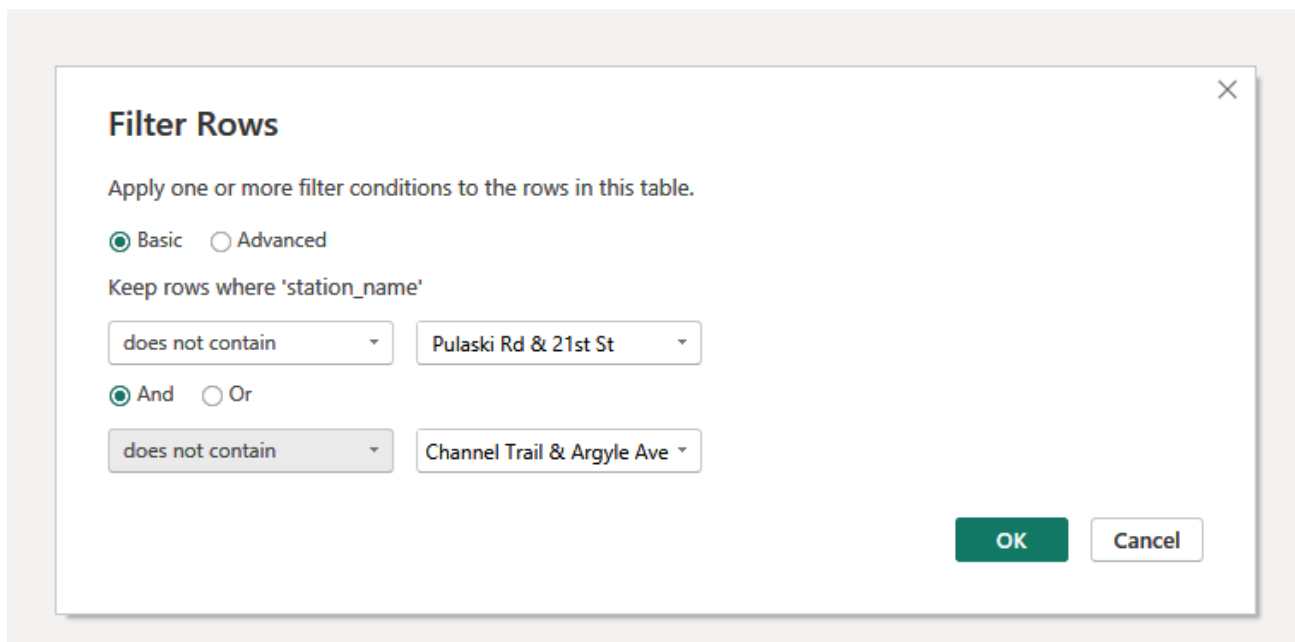


Figure 32 – Removing any invalid records by filtering rows

To remove the two invalid records from the <stations> table, the "Filter Rows" dialog box shown in Figure 32 is utilized. The filter conditions are then set to exclude rows where the station_name contains "Pulaski Rd & 21st St" or "Channel Trail & Argyle Ave". These filter conditions will effectively removing the duplicate entries highlighted at section [3.4.4](#) and section [3.4.5](#) from the table.

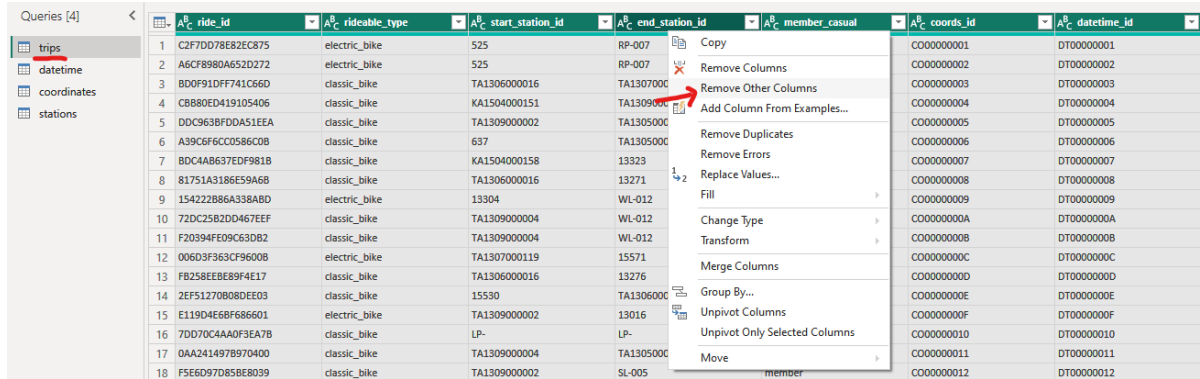
3.4.7. Conforming the data in <station_id> column is unique (DAX Queries)



Figure 33 - Using DAX queries to ensure the station_id column contains unique values

DAX Query as shown on Figure 33 is used to verify the uniqueness of the station_id column. The resulting empty table confirmed that all station_id values are unique, conforming the integrity of the data and finalizing the creation of the <stations> table.

3.5. Transforming the <trips> table into a fact table



	ride_id	rideable_type	start_station_id	end_station_id	member_casual	coords_id	datetime_id
1	C2F7D078C82EC875	electric_bike	525	RP-007		CO00000001	DT00000001
2	A6CF8980A652D272	electric_bike	525	RP-007		CO00000002	DT00000002
3	B0DF91DFF741C66D	classic_bike	TA1306000016	TA13070000		CO00000003	DT00000003
4	C8B80ED419105406	classic_bike	KA1504000151	TA13090000		CO00000004	DT00000004
5	DDC9638FDDA51EEA	classic_bike	TA1309000002	TA13050000		CO00000005	DT00000005
6	A39C6F6CC0586C0B	classic_bike	637	TA13050000		CO00000006	DT00000006
7	BDCAAB637EDF981B	classic_bike	KA1504000158	13323		CO00000007	DT00000007
8	81751A3186E59A6B	classic_bike	TA1306000016	13271		CO00000008	DT00000008
9	154222B86A338ABD	electric_bike	13304	WL-012		CO00000009	DT00000009
10	72DC25B2DD467EEF	classic_bike	TA1309000004	WL-012		CO0000000A	DT0000000A
11	F20394FE09C63DB2	classic_bike	TA1309000004	WL-012		CO0000000B	DT0000000B
12	006D3F363CF9600B	electric_bike	TA1307000119	15571		CO0000000C	DT0000000C
13	F8258EEBE89F4E17	classic_bike	TA1306000016	13276		CO0000000D	DT0000000D
14	2EF51270B080DE03	classic_bike	15530	TA13060000		CO0000000E	DT0000000E
15	E119D4E6BF686601	electric_bike	TA1309000002	13016		CO0000000F	DT0000000F
16	7DD70C4AA0F3EA7B	classic_bike	LP-	LP-		CO00000010	DT00000010
17	0AA241497B970400	classic_bike	TA1309000004	TA13050000		CO00000011	DT00000011
18	F5E6D97D85BE8039	classic_bike	TA1309000002	SL-005	member	CO00000012	DT00000012

Figure 34 – Transforming the trips table to a fact table

The image in Figure 34 shows the process of removing unnecessary columns from the <trips> table. This step is to prepare the <trips> table for use as a fact_table.

3.6. Data Modelling Relations and Summary

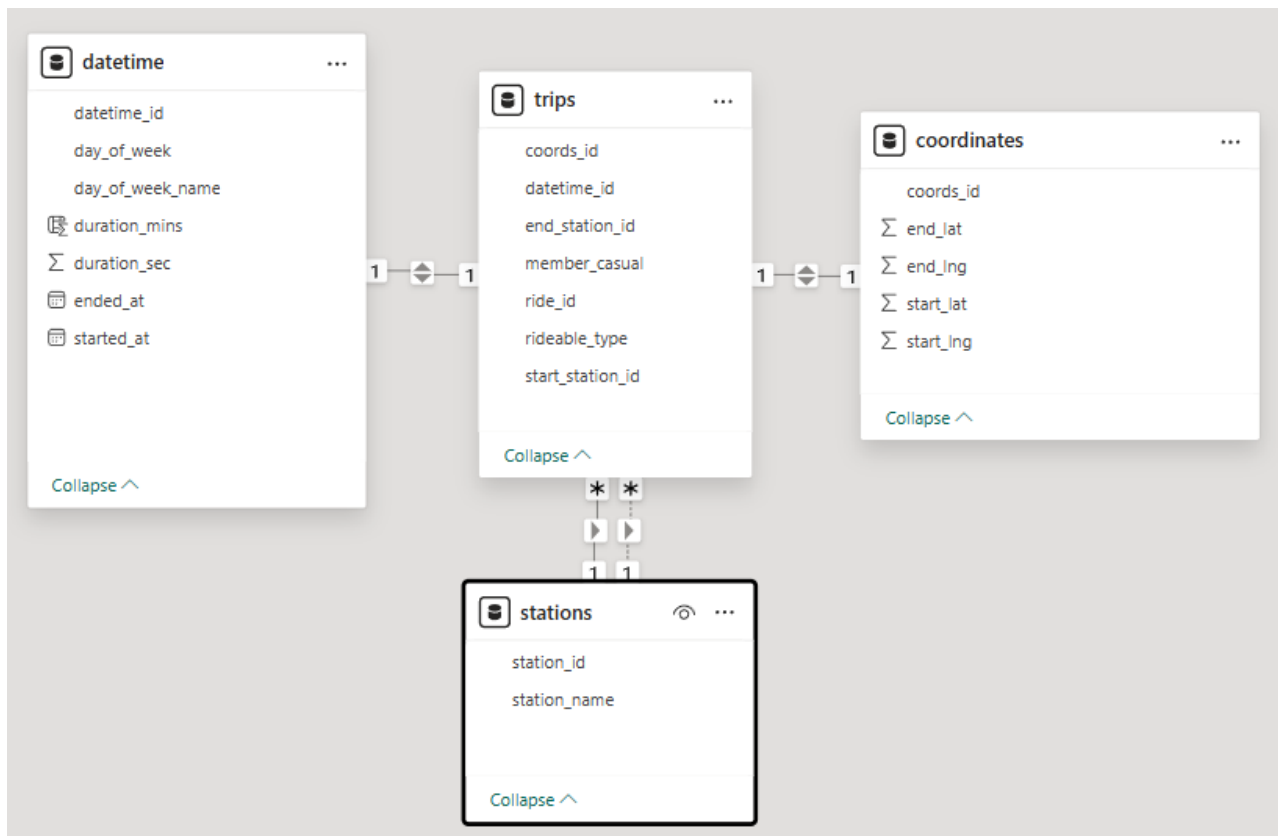


Figure 35 - Star schema data model showing relationships between tables

The diagram in Figure 35 represents a star schema model consisting of four interconnected tables: <datetime>, <trips>, <coordinates>, and <stations>. In this star schema, the <trips> table serves as the fact table.

Relationships between the tables are: <datetime> to <trips> via **datetime_id**, <coordinates> to <trips> via **coords_id**, and <stations> to <trips> via **start_station_id** and **end_station_id**.

This star schema model organizes data, ensures integrity, and optimizes analytical queries. In Table 2, Table 3, Table 4, and Table 5 are data dictionaries that outlines the columns, data types, and descriptions of each table.

trips table		
Column Name	Data Type	Description
coords_id	String	A unique identifier for each coordinate record.
datetime_id	String	A unique identifier for each datetime record.
end_station_id	String	The ID of the station where the ride ended.
member_casual	String	Indicates if the rider is a member or a casual user.
ride_id	String	A unique identifier for each ride.
rideable_type	String	The type of bike used for the ride (e.g., electric, classic).
start_station_id	String	The ID of the station where the ride started.

Table 2 – Data Dictionary of trips table

datetime table		
Column Name	Data Type	Description
datetime_id	String	A unique identifier for each datetime record.
day_of_week	Integer	The day of the week when the ride started, represented as a whole number (0 for Sunday, 1 for Monday, etc.).
day_of_week_name	String	The full name of the day of the week when the ride started (e.g., "Monday", "Tuesday").
duration_mins	Integer	The duration of the ride in minutes.
duration_sec	Integer	The duration of the ride in seconds.
ended_at	DateTime	The timestamp indicating when the ride ended.
started_at	DateTime	The timestamp indicating when the ride started.

Table 3 - Data Dictionary of datetime table

coordinates table		
Column Name	Column Name	Column Name
coords_id	String	A unique identifier for each coordinate record.
end_lat	Decimal	The latitude coordinate where the ride ended.
end_lng	Decimal	The longitude coordinate where the ride ended.
start_lat	Decimal	The latitude coordinate where the ride started.
start_lng	Decimal	The longitude coordinate where the ride started.

Table 4 - Data Dictionary of coordinates table

stations table		
Column Name	Data Type	Description
station_id	String	A unique identifier for each station.
station_name	String	The name of the station.

Table 5 - Data Dictionary of stations table

3.7. Filtering out invalid durations

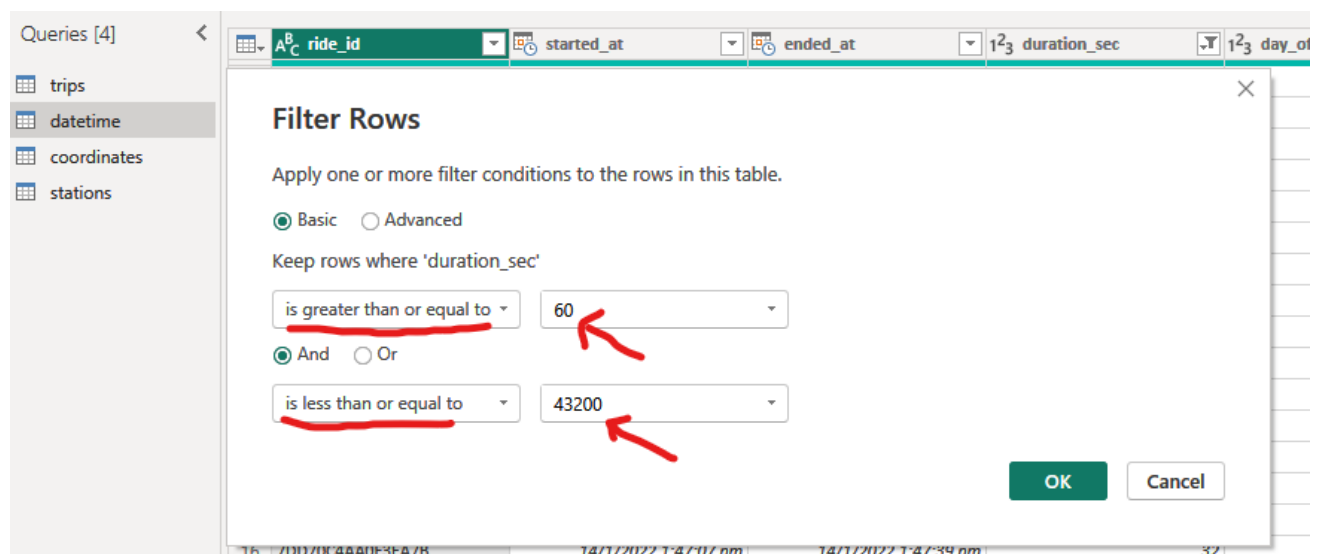


Figure 36 - Filtering the datetime table to include only durations between 60 seconds and 12 hours.

After establishing the relationships between the tables, an important step to take prior to generating any data visualizations is to filter the <datetime> table to retain only data with durations between 60 seconds and 12 hours as shown in Figure 36.

This step enhances the data integrity by removing data that does not reflect typical user riding behaviour. Ensuring that the data falls within this range helps to eliminate any outliers that could skew the analysis and lead to inaccurate insights.

The purpose of structuring and cleansing of the data is to ensure data integrity during data analysis.



BUSINESS INTELLIGENCE SOLUTION

Divvy Bike Sharing Analysis

William Gan

4. Executive Summary

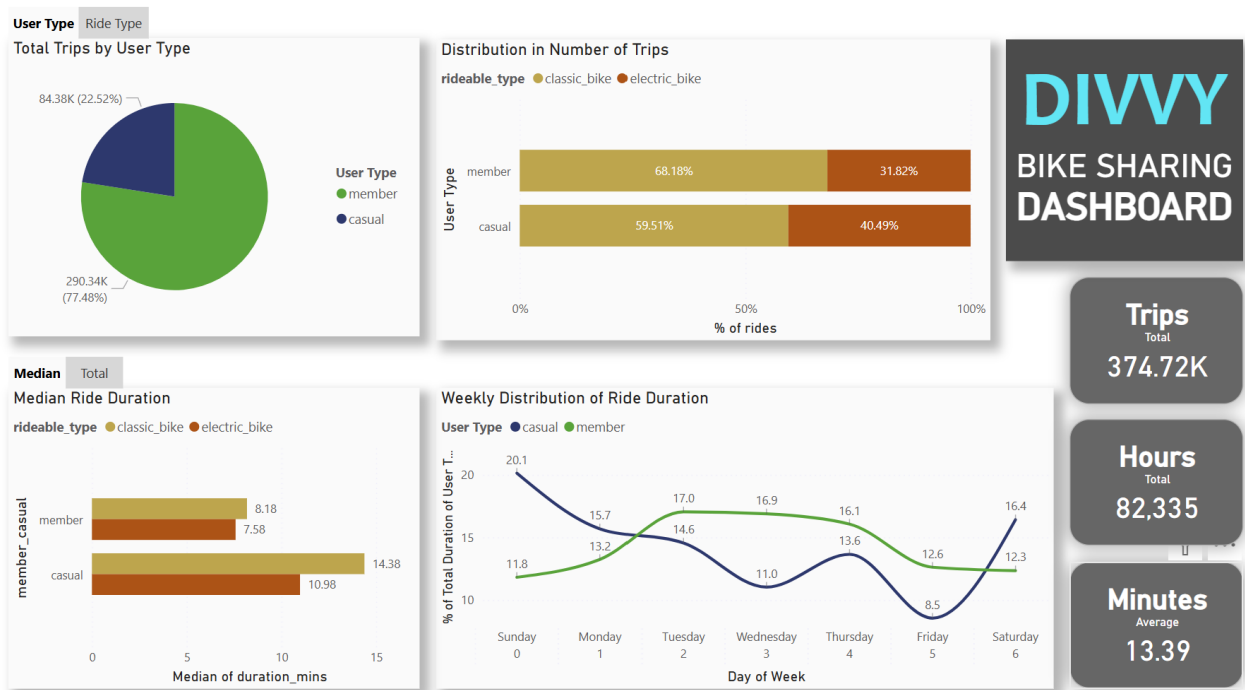


Figure 37 - Bike Sharing Utilisation Dashboard

This Business Intelligence report analyzed the Divvy Bikes dataset of 3 months from January 2022 to March 2022 to identify strategies for increasing membership conversions and attracting new customers. The objective is to fill the service gap and convert casual riders or attract new customers. Figure 37 is a dashboard that was created.

The key findings include that members account for 77.48% of total trips, while casual riders who make up the remaining 22.52% tend to have longer ride durations. Casual riders prefer electric bikes and use the service more on weekends. Recommendations include introducing a new weekly pass introduced to provide significant cost benefits for the identified targeted audience, tourists and short-term visitors, and strategies for advertisements to promote the new plan discussed to bring awareness to the new plans to increase user engagement.

5. Data Analysis

5.1. Introduction

This section showcases the steps to discover solutions by performing data analysis and addresses the questions highlighted in and provides information about the data collected.

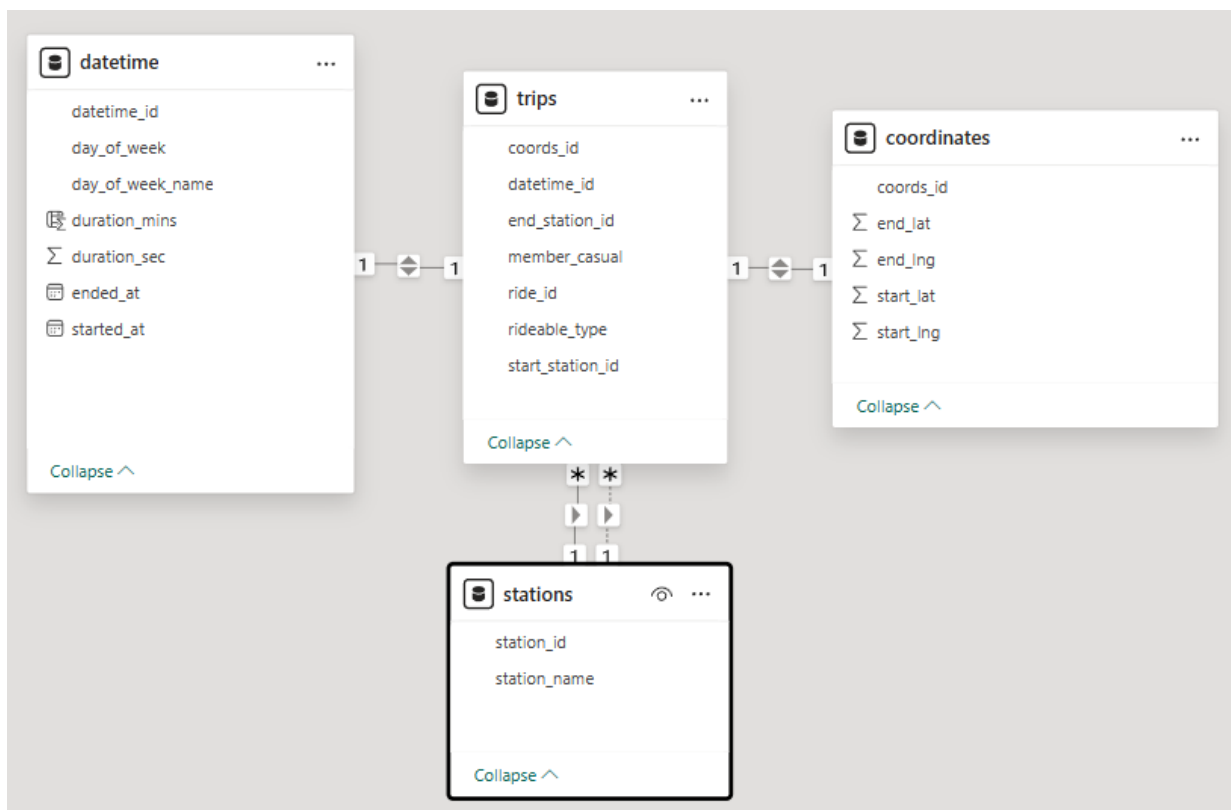


Figure 38 - Overview of the star schema model

The dataset used in this analysis is sourced from Divvy Bikes, a popular bike-sharing service in Chicago, Illinois.

Figure 38 shows the data model that was produced from the steps taken in section 3, providing a visual representation of how the tables are interconnected. It includes trip-related data such as user type (member or casual), start and end datetimes, and the type of bike used (classic or electric).

The data will be examined to answer the following key questions:

- How do annual members and casual riders use the bike service differently?
- Why would casual riders switch to purchase memberships?
- How to influence new customers to utilize the bike sharing service

5.2. Analysis of Total Trips

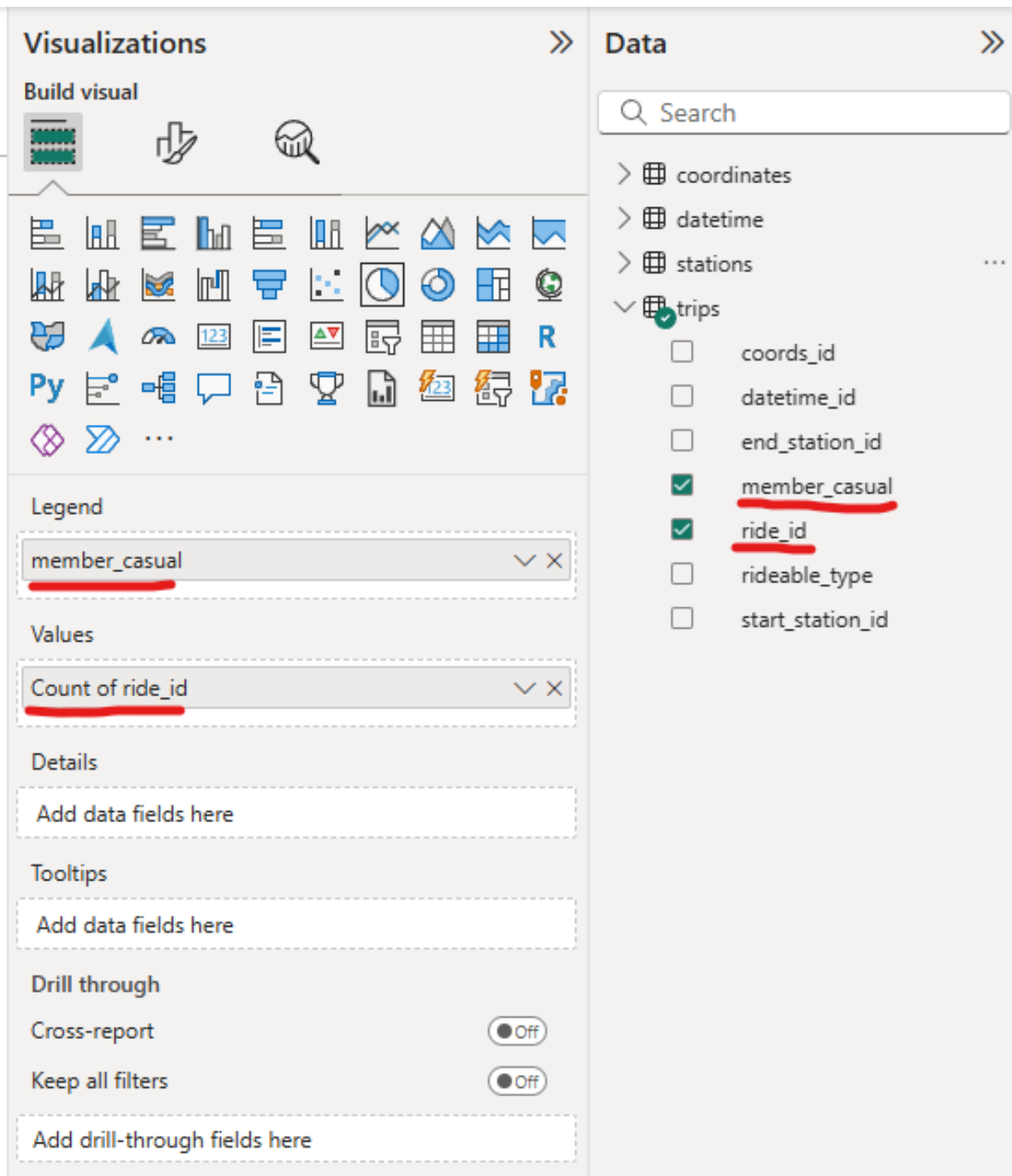


Figure 39 - Visualization Settings for Total Trips Pie Chart

The <Total Trips> pie chart is created from the steps taken as shown in Figure 39. In the legend, member_casual field was selected, and in the values field, the count of ride_id.

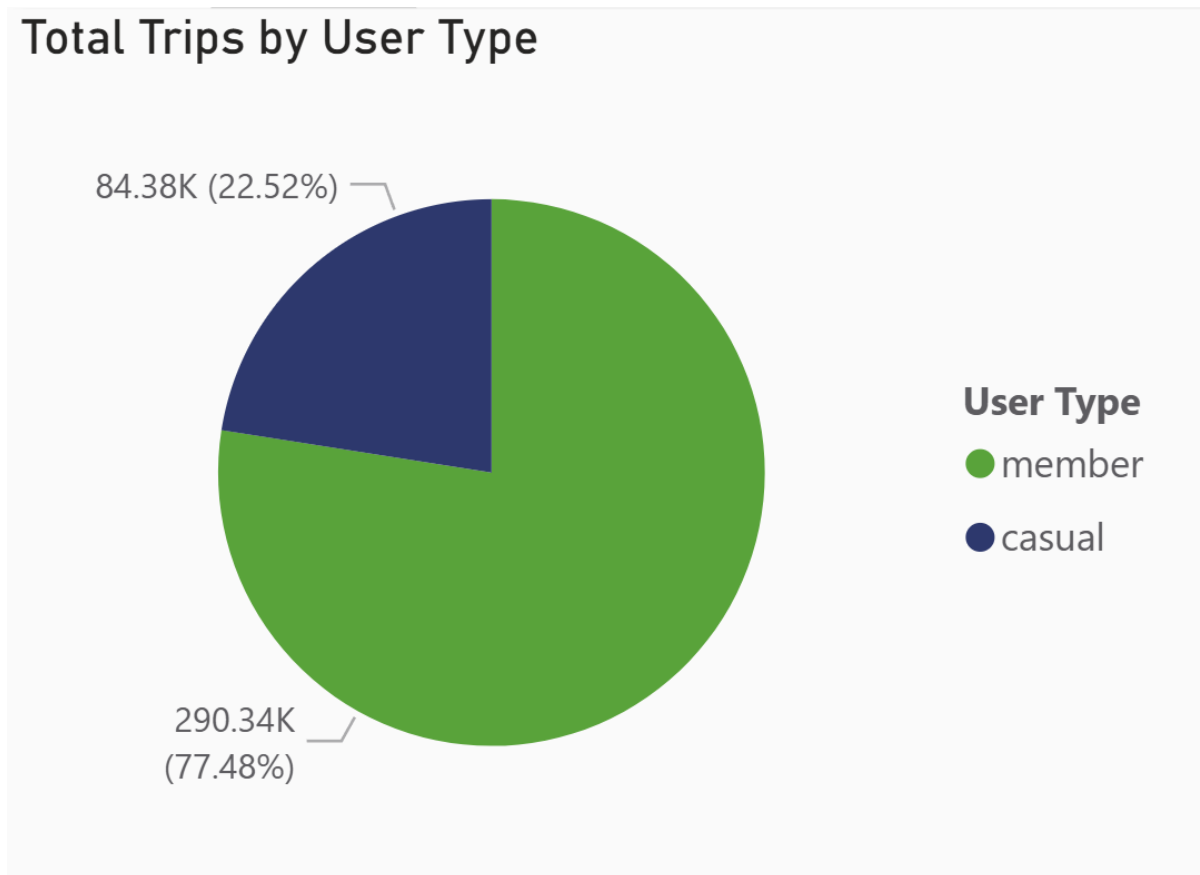


Figure 40 – Pie Chart of Total Trips by User Type

The resulting pie chart, shown in Figure 40 illustrates the proportion of trips taken by casual riders versus annual members. This visualization is crucial for identifying which group utilizes the bike-sharing service more frequently.

- Members account for 77.48% (290.34K) of the total trips.
- Casual users account for 22.52% (84.38K) of the total trips.

5.3. Analysis of Total Ride Duration (New column using DAX)

```
duration_mins = 'datetime'[duration_sec]/60
```

Figure 41 - Formula to calculate duration in minutes

To conduct this analysis, a new column using DAX expression for total ride duration in minutes named 'duration_mins' was created, this calculates the ride duration in minutes. The formula shown in Figure 41 was used to convert the duration from seconds to minutes to provide a more relatable understanding of the ride times.

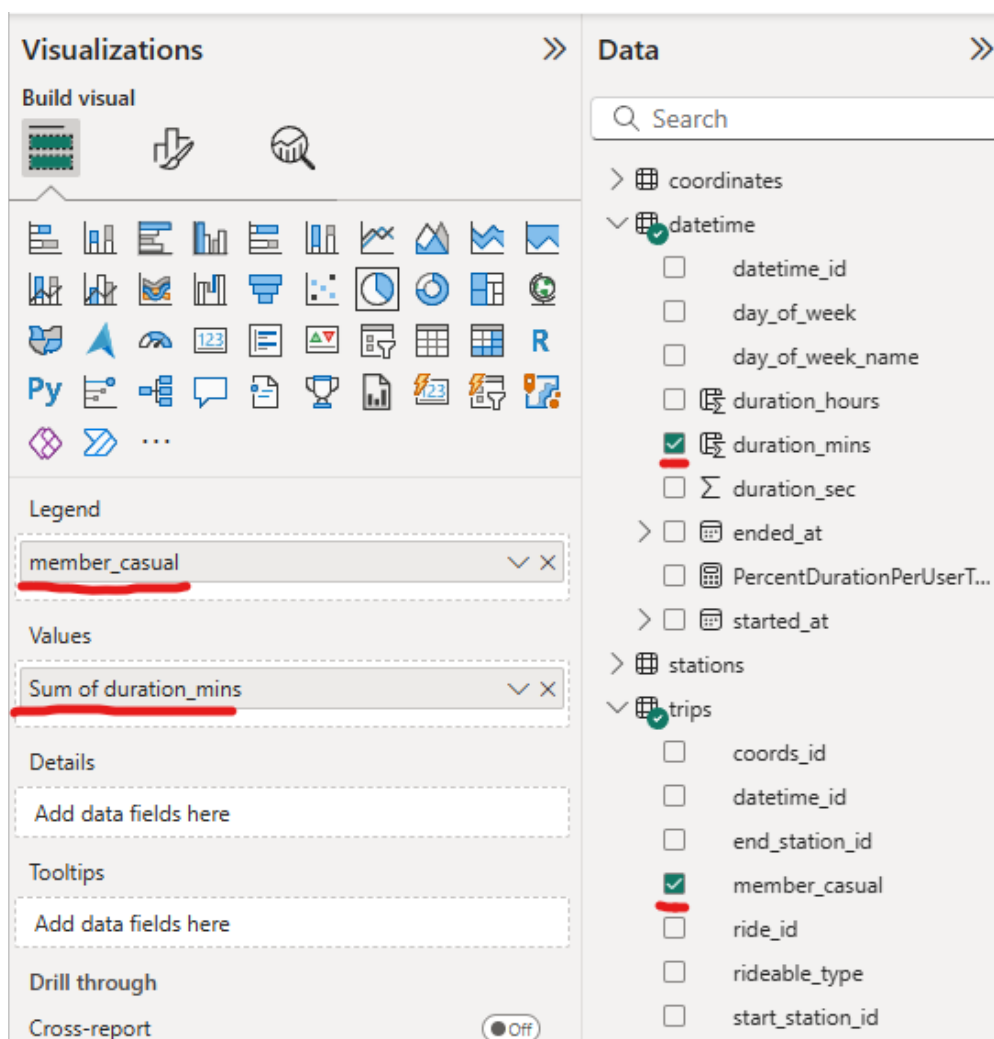


Figure 42 - Visualization Settings for Total Duration Pie Chart

A pie chart was generated with the setting shown in Figure 42 to display the total ride duration segmented by user type. The member_casual field is used to distinguish between annual members and casual riders, while the “duration_mins” measure is used for aggregating the total ride time in minutes for each user type.

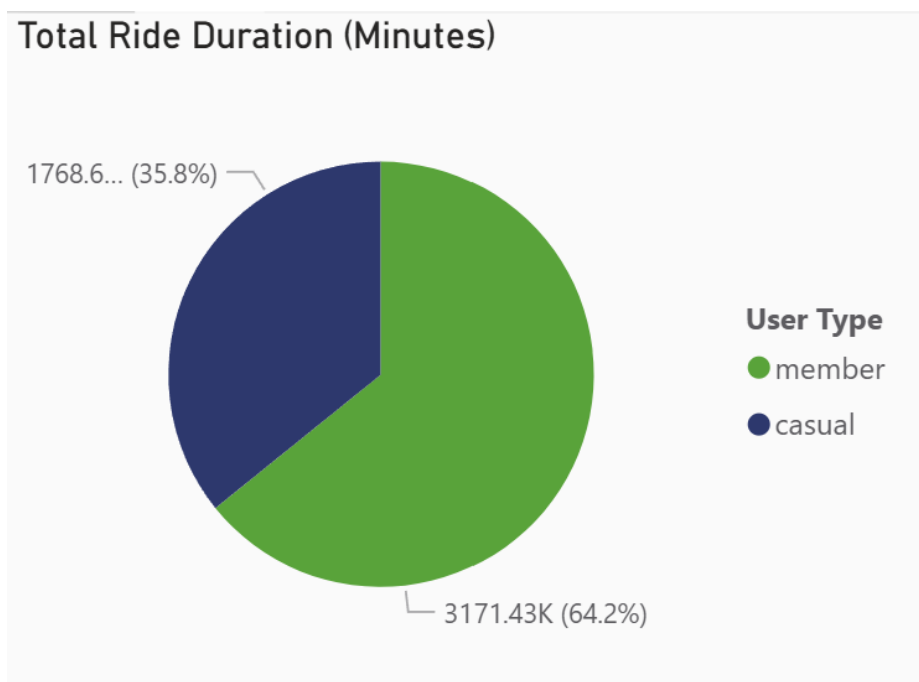


Figure 43 – Total Ride Duration Pie Chart

The resulting pie chart shown in Figure 43 visualizes the total ride duration for each user group, providing a view of how much time annual members and casual riders spend on rides.

- Members have a total ride duration of ~3171 thousand minutes, making up 64.2% of the total duration.
- Casual users have a total ride duration of ~1768 thousand minutes, making up 35.8% of the total duration

5.4. Analysis of Median Ride Duration

The purpose of this analysis is to visualize the median ride duration for different user types (annual members versus casual riders). Understanding the median ride duration helps in identifying a typical user usage behavior.

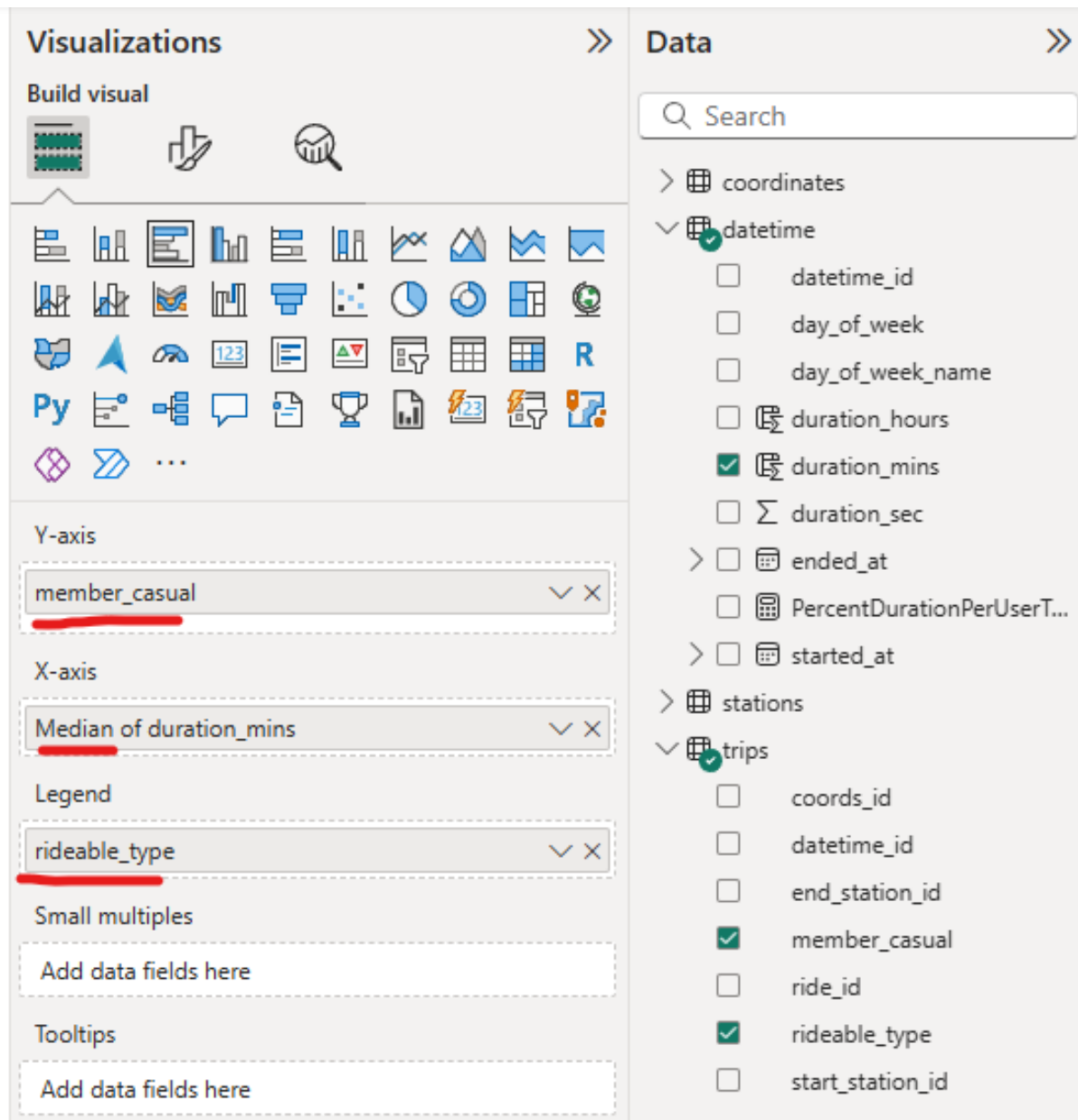


Figure 44 - Visualization Settings for Median Ride Duration Bar Chart

To perform this analysis, a measure for the median ride duration in minutes was calculated with the setting shown in Figure 44. A median form measure provides

a central value that represents the typical ride duration for each user type, excluding the influence of outliers.

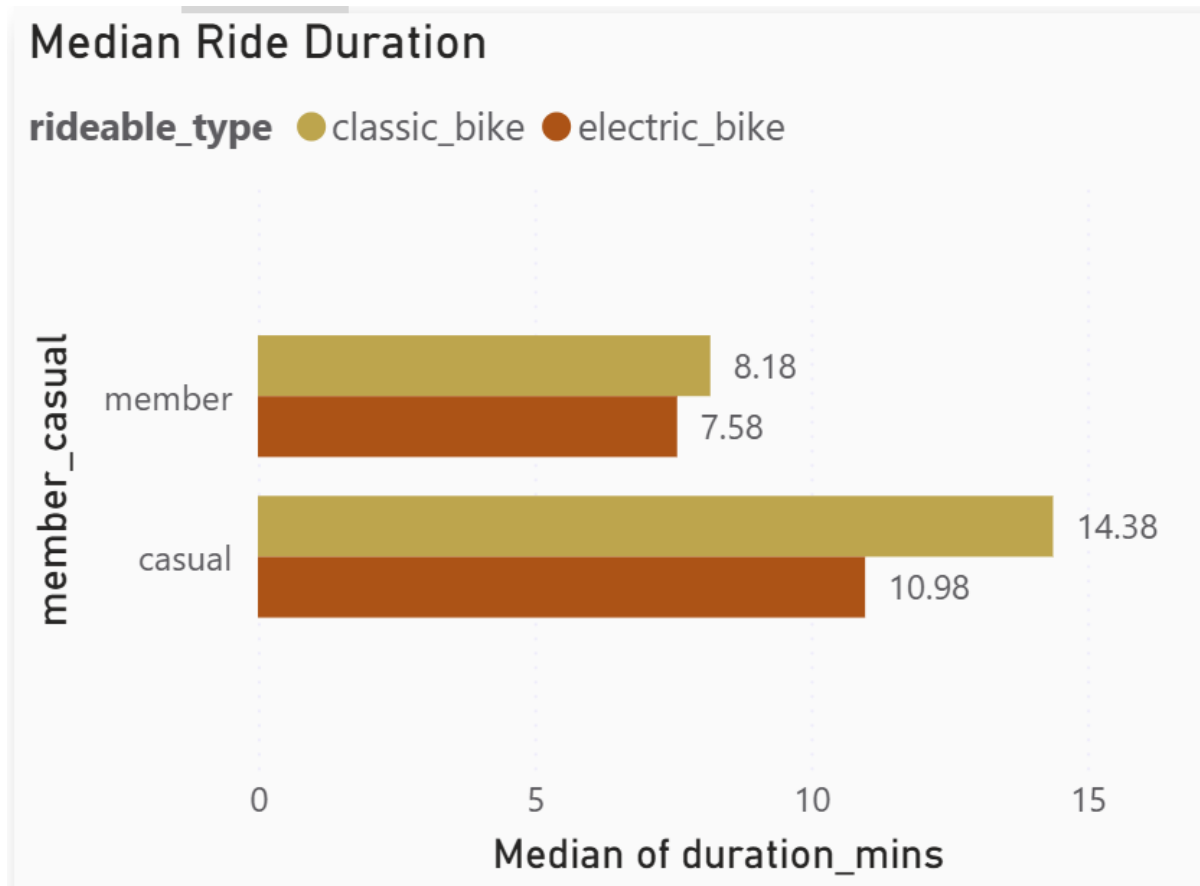


Figure 45 – Bar Chart for Median Ride Duration

	Classic bikes	Electric bikes
Members	8.18 minutes	7.58 minutes
Casual	14.38 minutes	10.98 minutes

Table 6 - Median Ride Duration of Users by Bike Type

The resulting bar chart, as shown in Figure 45, illustrates the median ride duration for annual members and casual riders, segmented by the type of bike used. This visualization helps in understanding the typical ride duration for each

user group and bike type, offering valuable insights into user preferences and behaviour. Table 6 summarizes the findings of median ride duration with user type and bike type.

5.5. Distribution of Bike Usage by User Type

The aim of this analysis is to visualize the distribution of bike-sharing trips by user type and bike type. Understanding the distribution helps in identifying user and bike preferences.

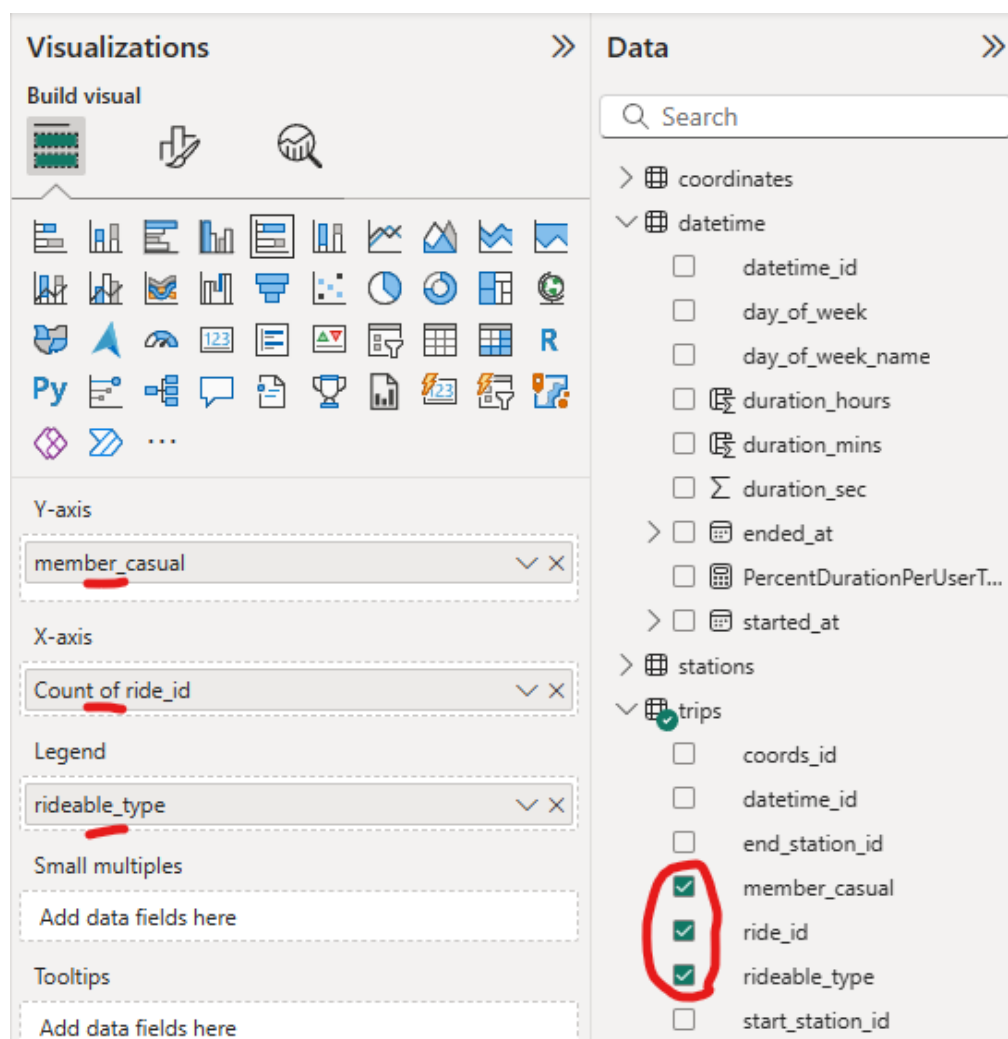


Figure 46 - Visualization Settings for Distribution of Bike Usage

A stacked bar chart was created using the settings shown on Figure 46 to display the distribution of trips by user type and bike type shown on Figure 47.

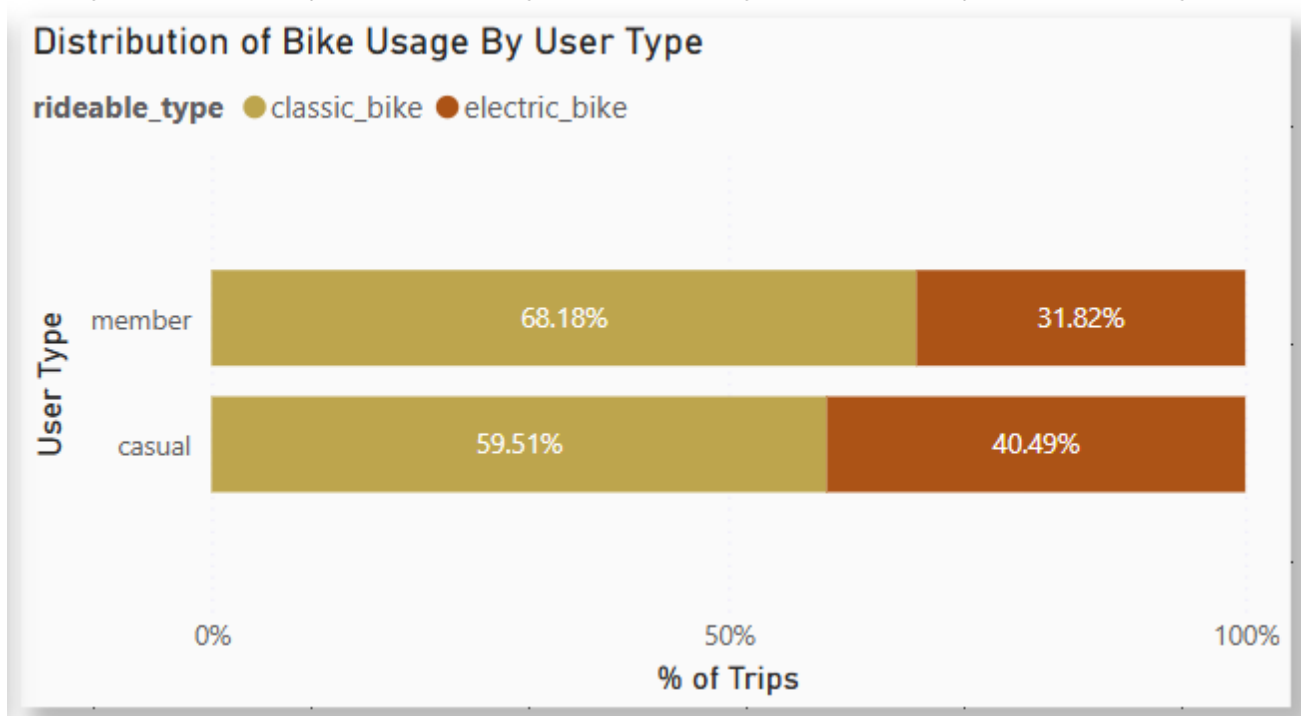


Figure 47 – Bar Chart for Distribution of Bike Usage By User

	Classic bikes	Electric bikes
Members	68.18%	31.82%
Casual	59.51%	40.49%

Table 7 – Readings of Distribution of Bike Usage

The resulting bar chart shown in Figure 47 illustrates the distribution of trips for annual members and casual riders, segmented by the type of bike used. This visualization provides a clear comparison of preferences between user groups. Table 7 summarizes the distribution percentages for user type and bike type.

5.6. Weekly Distribution of Ride Duration (New measure using DAX)

The purpose of this analysis is to visualize the weekly distribution of ride duration for annual members and casual riders. Understanding how ride durations vary throughout the days in a week helps in identifying peak usage times.

```
1 PercentDurationPerUserType =
2
3 VAR TotalDurationCasual =
4 CALCULATE(
5     SUM('datetime'[duration_sec]),
6     FILTER(ALL('trips'), 'trips'[member_casual] = "casual")
7 )
8
9 VAR TotalDurationMember =
10 CALCULATE(
11     SUM('datetime'[duration_sec]),
12     FILTER(ALL('trips'), 'trips'[member_casual] = "member")
13 )
14
15 VAR GroupDuration =
16     CALCULATE(
17         SUM('datetime'[duration_sec]),
18         'trips'[member_casual] = SELECTEDVALUE('trips'[member_casual])
19     )
20
21 VAR TotalGroupDuration =
22     SWITCH(
23         SELECTEDVALUE('trips'[member_casual]),
24         "casual", TotalDurationCasual,
25         "member", TotalDurationMember
26     )
27
28 RETURN
29 DIVIDE(GroupDuration, TotalGroupDuration, 0) * 100
30
```

Figure 48 – DAX Measure formula for Percent Duration Per User Type

In Figure 48, a DAX measure, “PercentDurationPerUserType” was created to calculate the percentage of the total ride duration for each user type. This involves summing up the duration in seconds for casual and member riders separately, and then calculating the percentage of the group duration relative to the total duration for each day of the week. It is used to show a comparative view of how ride durations are distributed across different days and user types.

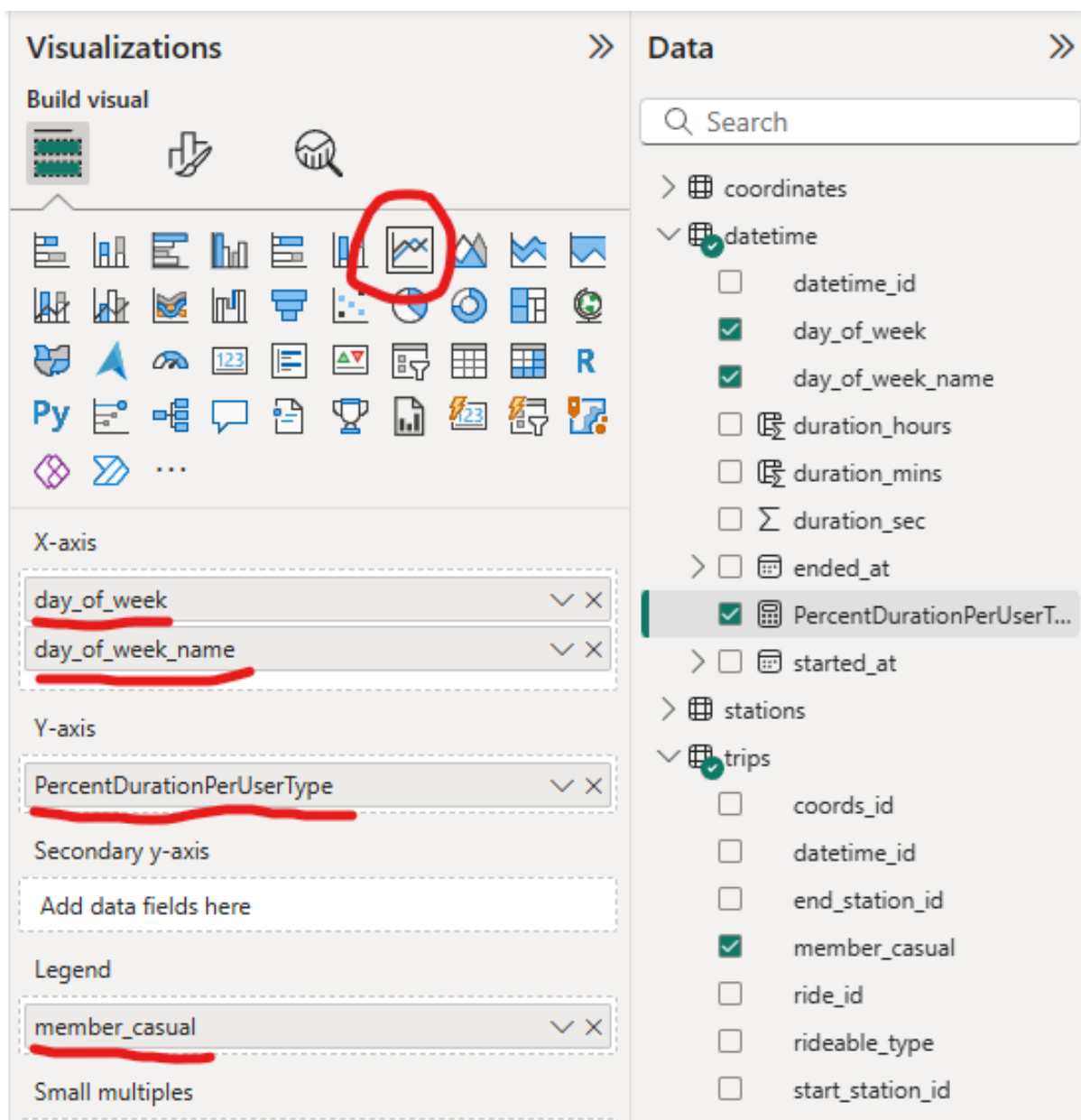


Figure 49 – Visualization Setting for Weekly Distribution of Ride Duration

A line chart was created to display the weekly distribution of ride duration using the settings shown in Figure 49.

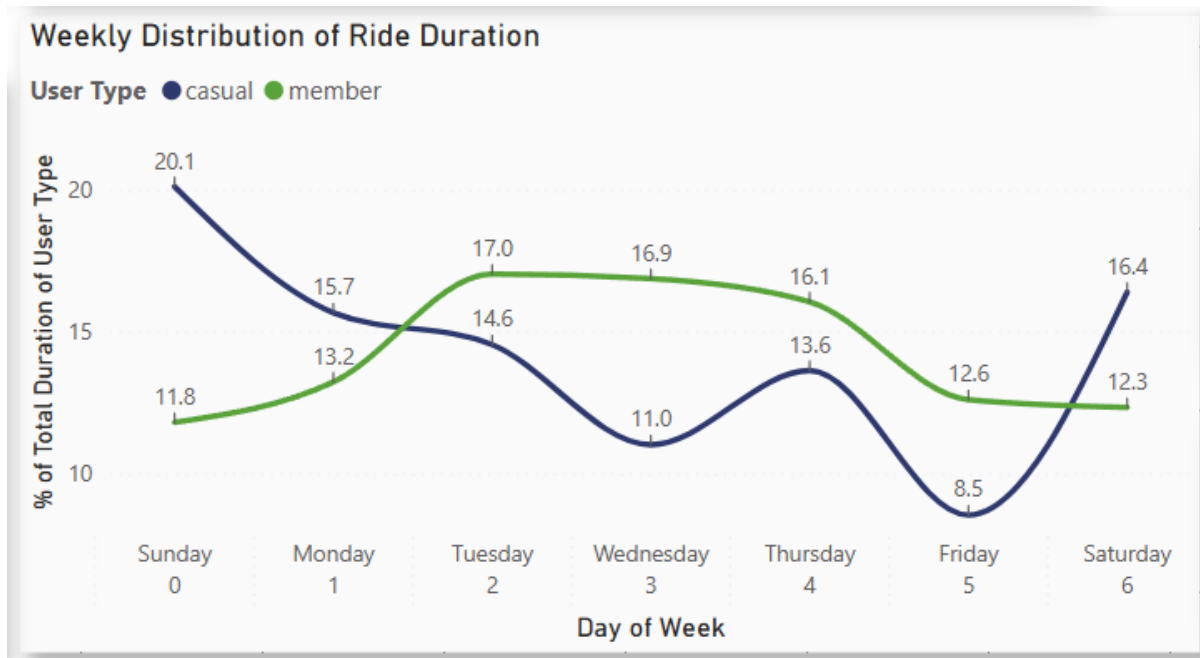
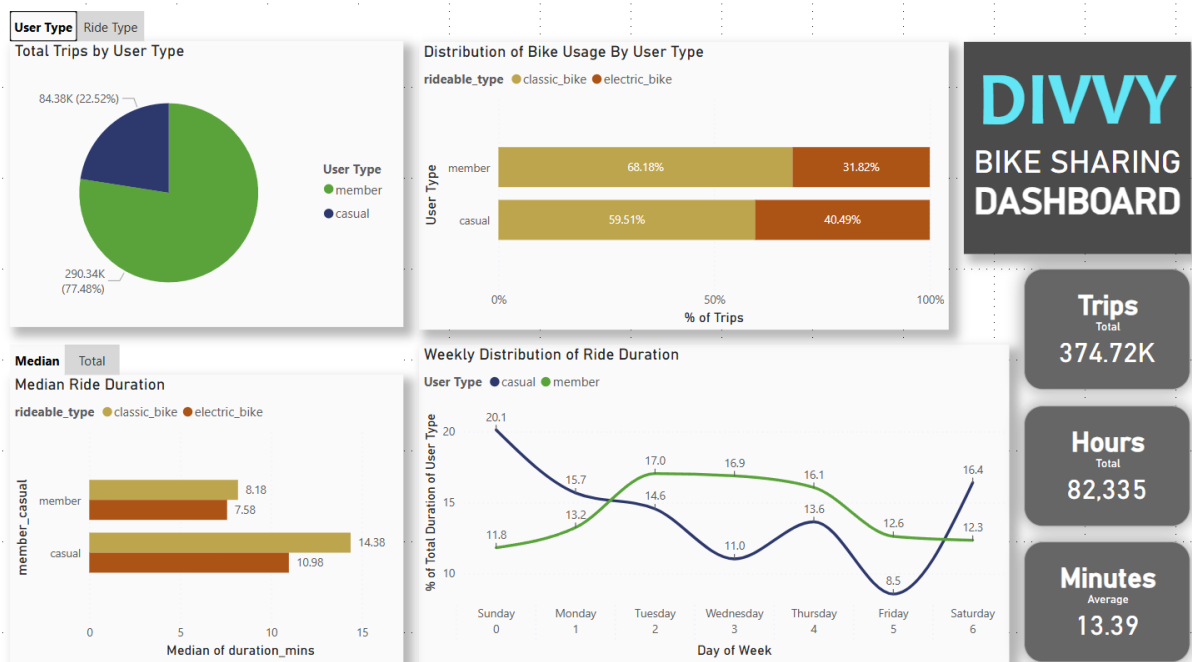


Figure 50 – Line Chart for Weekly Distribution of Ride Duration

The resulting line chart shown in Figure 50 illustrates the weekly distribution of ride duration for both members and casual riders. The chart provides a visual comparison of how ride durations are distributed throughout the week for each user group.

5.7. Data Visualization Dashboard (New measure with DAX)



```
1 duration_hours = 'datetime'[duration_sec]/(60*60)
```

Figure 52 - DAX Measure Formula used for Hours Total

The data visualization dashboard produced in Figure 51 consolidates the charts created for data analysis, it offers an interactive view of the usage of the bike sharing service.

Figure 52 is a DAX measure for the total hours card that was created for the dashboard shown in Figure 51. This dashboard is created to allow relevant stakeholders to grasp key trends and patterns, and to aid in decision making and strategic planning.

6. Conclusions and Recommendations

Before continuing, it's crucial to understand the current pricing information on the bike sharing service.

6.1. Pricing Information of Bike Sharing Service

6.1.1. Pricing Model of Classic Bike Rental

Classic Bike Rental				
		Unlock Price	Free Duration	Cost/Min
Casual	Single Ride	\$1	-	\$0.18
	Day Pass (\$18.10/Day)	Free	180 Mins/Ride	\$0.18
Member	Divvy (\$143.90/Year) (\$11.99/Month)	Free	45 Mins/Ride	\$0.18

Table 8 – Pricing Information of Classic Bike

6.1.2. Pricing Model of Electric Bike Rental

Electric Bike Rental				
		Unlock Price	Free Duration	Cost/Min
Casual	Single Ride	\$1	-	\$0.44
	Day Pass (\$18.10/Day)	Free	-	\$0.44
Member	Divvy (\$143.90/Year) (\$11.99/Month)	Free	-	\$0.18

Table 9 – Pricing Information of Electric Bike

6.2. Correlation Between Usage Patterns and Pricing

6.2.1. Casual Riders have a higher median duration

The higher median duration on both type of bikes for casual riders can be attributed to the **180 free minutes provided on the day pass compared to 45 minutes on the member plan.**

6.2.2. Preference for Electric Bikes for Casual Riders

The pricing structure indicates that electric bikes are significantly more expensive per minute compared to classic bikes. However, **casual riders preference for electric bikes suggests they are willing to pay more for the convenience and speed of electric bikes, or possibly for leisure purposes.**

6.2.3. Weekday vs Weekend Usage

Casual users have higher usage on weekends compared to weekdays, indicating a preference for leisure rides. It may make less sense to casual riders to commit to a full month membership

6.3. Recommendations and Justification

6.3.1. A New Pricing Plan and Explanation

Weekly Pass (New) (\$24.99 / Week)			
	Unlock Price	Free Duration	Cost/Min
Classic Bike	Free	60 Mins / Ride	\$0.18
Electric Bike	Free	90 Mins / Week	\$0.18

Table 10 – Illustration of a recommended new pricing plan

The new pricing plan shown in Table 10 is an illustration to target new customers and convert casual riders on Single Ride or Day Pass to a newly proposed weekly pass plan.

To justify, casual Riders make fewer but longer trips on average and have a stronger preference for electric bikes than Members. Casual Riders also have a higher duration usage on weekends than weekdays. In addition, the cost efficiency of the new plan is shown in Table 11.

The target audience for this plan are tourists and short term visitors as they are likely visiting the city for leisure or business and may not be familiar with the infrastructure of the city. Tourists and visitors are usually in for a short stay and would most likely prefer a convenient and flexible transportation which will allow for easier navigation and better comfort.

6.3.2. Cost Efficiency of Electric Bike Usage per Plan

Single Ride

1. Unlock Fee: \$1
2. Cost per Minute: \$0.44
3. Cost for 90 Minutes: $90 * \$0.44 = \39.60
4. Total Cost for 90 Minutes: $\$1 + \$39.60 = \underline{\$40.60}$

Day Pass

1. Day Pass Price: \$18.10
2. Cost per Minute: \$0.44
3. Cost for 90 Minutes: $90 * \$0.44 = \39.60
4. Total Cost for 90 Minutes: $\$18.10 + \$39.60 = \underline{\$57.70}$

Monthly Pass

1. Monthly Pass Cost: \$11.99
2. Cost per Minute: \$0.18
3. Cost for 90 Minutes: $90 * \$0.18 = \16.20
4. Total Cost for 90 Minutes: $\$11.99 + \$16.20 = \underline{\$28.19}$

Weekly Pass

1. Pass Cost: \$24.99
2. Free Duration: 90 minutes
3. Cost/Minute Exceeding Free Duration: \$0.18
4. Total Cost for 90 Minutes: $\underline{\$24.99}$

Table 11 – Cost Efficiency Calculations and Summary

Table 11 compares the cost of riding an e-bike for the first 90 minutes using different pricing plans. The weekly pass is the most cost-effective option, followed by the monthly pass. The single ride option is more expensive, with the day pass being the most expensive overall. Hence, the illustrated weekly pass is a compelling choice for users who are committing short term, with the habit of longer ride durations and preference for electric bikes.

6.3.3. Targeted advertisements to promote illustrated weekly plan

Targeted advertisements can be used to promote the new weekly membership plan to casual riders who often use the service. This can potentially increase user retention and result in a steady revenue stream. The new membership plan could also be promoted at non-customers for those who disinterested with the previous existing options.

7. References

DougKlopfenstein (2024) Expressions, values, and let expression - powerquery m, PowerQuery M | Microsoft Learn. Available at: <https://learn.microsoft.com/en-us/powerquery-m/expressions-values-and-let-expression> (Accessed: 01 July 2024).

Kfollis (2024) Switch function (DAX) - dax, function (DAX) - DAX | Microsoft Learn. Available at: <https://learn.microsoft.com/en-us/dax/switch-function-dax> (Accessed: 07 July 2024).

Kfollis and jeroenterheerdt (2023) DAX Queries Microsoft Learn. Available at: <https://learn.microsoft.com/en-us/dax/dax-queries> (Accessed: 04 July 2024).

Peter-Myers (2022) Use variables to improve your dax formulas - dax, DAX | Microsoft Learn. Available at: <https://learn.microsoft.com/en-us/dax/best-practices/dax-variables> (Accessed: 11 August 2024).

Power BI Resources (2024) Power BI Guide. Available at: <https://pbi.guide/resources/> (Accessed: 01 July 2024).