Case Study : Bike Sharing

Install the below packages

>install.packages(“tidyverse”)

> install.packages()

> install.packages("skimr")

Install the below packages

>library(tidyverse)

> library(skimr) # to get summary data

> library(janitor)

> library(dplyr)

STEP 1: COLLECT DATA

It will read the csv from the mentioned path

> Trips_Apr20 <- read_csv('./trip_data/202004-divvy-tripdata.csv')

> Trips_May20 <- read_csv('./trip_data/202005-divvy-tripdata.csv')

> Trips_Jun20 <- read_csv('./trip_data/202006-divvy-tripdata.csv')

> Trips_Aug20 <- read_csv('./trip_data/202008-divvy-tripdata.csv')

> Trips_Sep20 <- read_csv('./trip_data/202009-divvy-tripdata.csv')

> Trips_Oct20 <- read_csv('./trip_data/202010-divvy-tripdata.csv')

> Trips_Nov20 <- read_csv('./trip_data/202011-divvy-tripdata.csv')

> Trips_Dec20 <- read_csv('./trip_data/202012-divvy-tripdata.csv')

> Trips_Jan21 <- read_csv('./trip_data/202101-divvy-tripdata.csv')

> Trips_Feb21 <- read_csv('./trip_data/202102-divvy-tripdata.csv')

> Trips_Mar21 <- read_csv('./trip_data/202103-divvy-tripdata.csv')

> Trips_Apr21 <- read_csv('./trip_data/202104-divvy-tripdata.csv')

STEP 2: WRANGLE DATA AND COMBINE INTO A SINGLE FILE

It will show the columns present in all files

```
> colnames(Trips_Apr20)
> colnames(Trips_May20)
> colnames(Trips_Jun20)
> colnames(Trips_Jul20)
> colnames(Trips_Aug20)
> colnames(Trips_Sep20)
> colnames(Trips_Oct20)
> colnames(Trips_Nov20)
> colnames(Trips_Dec20)
> colnames(Trips_Jan21)
> colnames(Trips_Feb21)
> colnames(Trips_Mar21)
> colnames(Trips_Apr21)
```

Inspect the data frames and look for incongruencies

```
#================================================
> str(Trips_Apr20)
> str(Trips_May20)
> str(Trips_Jun20)
> str(Trips_Jul20)
> str(Trips_Aug20)
> str(Trips_Sep20)
> str(Trips_Oct20)
> str(Trips_Nov20)
> str(Trips_Dec20)
> str(Trips_Jan21)
> str(Trips_Feb21)
> str(Trips_Mar21)
> str(Trips_Apr21)
```

we can compare column datatype across all data frame by using compare_df_cols when we have large dataset, that would be more easy

it will check for the mismatch in dataset

```
> compare_df_cols(Trips_Apr20, Trips_May20, Trips_Jun20, Trips_Jul20,
+             Trips_Aug20, Trips_Sep20, Trips_Oct20, Trips_Nov20, Trips_Dec20,
+             Trips_Jan21, Trips_Feb21, Trips_Mar21, Trips_Apr21, return = "mismatch")
```

it was found out that end_station_id and start_station_id was having numeric datatype in some and character data type in some

Convert end_station_id and start_station_id to character so that they can stack correctly

```
> Trips_Apr20 <- mutate(Trips_Apr20, end_station_id =
+                 as.character(end_station_id), start_station_id =
+                 as.character(start_station_id))
> Trips_May20 <- mutate(Trips_May20, end_station_id =
+                 as.character(end_station_id), start_station_id =
+                 as.character(start_station_id))
> Trips_Jun20 <- mutate(Trips_Jun20, end_station_id =
+                 as.character(end_station_id), start_station_id =
+                 as.character(start_station_id))
> Trips_Jul20 <- mutate(Trips_Jul20, end_station_id =
+                 as.character(end_station_id), start_station_id =
+                 as.character(start_station_id))
> Trips_Aug20 <- mutate(Trips_Aug20, end_station_id =
+                 as.character(end_station_id), start_station_id =
+                 as.character(start_station_id))
> Trips_Sep20 <- mutate(Trips_Sep20, end_station_id =
+                 as.character(end_station_id), start_station_id =
+                 as.character(start_station_id))
> Trips_Oct20 <- mutate(Trips_Oct20, end_station_id =
+                 as.character(end_station_id), start_station_id =
+                 as.character(start_station_id))
```

```
> Trips_Nov20 <- mutate(Trips_Nov20, end_station_id =
+                       as.character(end_station_id), start_station_id =
+                       as.character(start_station_id))
> Trips_Apr21 <- mutate(Trips_Apr21, end_station_id =
+                       as.character(end_station_id), start_station_id =
+                       as.character(start_station_id))
```

double check column datatype across all dataframe

```
> compare_df_cols(Trips_Apr20, Trips_May20, Trips_Jun20, Trips_Jul20,
+                 Trips_Aug20, Trips_Sep20, Trips_Oct20, Trips_Nov20, Trips_Dec20,
+                 Trips_Jan21, Trips_Feb21, Trips_Mar21, Trips_Apr21, return = "mismatch")
# Now all the columns seems to be matching
```

Stack individual data frames into one big data frame

```
> all_trips <- bind_rows(Trips_Apr20, Trips_May20, Trips_Jun20, Trips_Jul20,
+                        Trips_Aug20, Trips_Sep20, Trips_Oct20, Trips_Nov20, Trips_Dec20,
+                        Trips_Jan21, Trips_Feb21, Trips_Mar21, Trips_Apr21)
```

Remove unused column

```
> all_trips <- all_trips %>%
+   select(-c(start_lat, start_lng, end_lat, end_lng))

# removed the lat and lng columns

> all_trips <- all_trips %>% rename(trip_id= ride_id ,ride_type =
+                                     rideable_type
+                                   ,start_time = started_at,end_time =ended_at
+                                   ,from_station_name = start_station_name
+                                   ,from_station_id = start_station_id
+                                   ,to_station_name = end_station_name
```

```
+                    ,to_station_id = end_station_id
+                    ,usertype = member_casual)
```

Rename the columns to the name we want

STEP 3: CLEAN UP AND ADD DATA TO PREPARE FOR ANALYSIS

Inspect the new table that has been created

```
> colnames(all_trips)
> dim(all_trips)
```

Shows the total count of data in the tables

```
> head(all_trips)

> summary(all_trips)
```

shows a basic information of the data currently with us

```
> skim(all_trips)
```

Add columns that list the date, month, day, and year of each ride

This will allow us to aggregate ride data for each month, day, or year … before completing these operations we could only aggregate at the ride level

```
> all_trips$date <- as.Date(all_trips$start_time)   #The default format is yyyy-mm-dd
> all_trips$month <- format(as.Date(all_trips$date), "%m")
> all_trips$day <- format(as.Date(all_trips$date), "%d")
> all_trips$year <- format(as.Date(all_trips$date), "%Y")
> all_trips$day_of_week <- format(as.Date(all_trips$date), "%A")

> all_trips$ride_length <- difftime(all_trips$end_time,all_trips$start_time)
> is.factor(all_trips$ride_length)
```

```
> all_trips$ride_length <- as.numeric(as.character(all_trips$ride_length))
> is.numeric(all_trips$ride_length)

> skim(all_trips$ride_length)
```

Add a “ride_length” calculation to all_trips (in seconds)

```
> all_trips_v2 <- all_trips[!(all_trips$ride_length<0),]
```

Remove “bad” data

The data frame includes a few hundred entries when bikes were taken out of docks and checked for quality by Divvy or ride_length was negative

```
> skim(all_trips_v2)
```

STEP 4: CONDUCT DESCRIPTIVE ANALYSIS

```
# Descriptive analysis on ride_length (all figures in seconds)
> summary(all_trips_v2$ride_length)
```

Export to CSV file for further analysis and view the data

```
> write.csv(all_trips_v2, "data.csv")
> save.image("E:/Google_DA/8/bike_data_report.RData")
> View(all_trips_v2)
```