Assignment2

Priyanka Kaushal (24200862)

Few declarations :-

1. Use **width = 80** and **wrap** to show complete output in PDF in str function in file.
2. Below code to suppress Warnings which will take unnecessary space in PDF.
3. Using knitr::kable to show output in tabular form

```
knitr::opts_chunk$set(warning = FALSE, message = FALSE)
```

Assignment 2 -

Task1 : Manipulation

1. Load the dataset pedestrian_2023.csv, save it as a tibble. Use a function from dplyr to remove the columns with a column name ending with "IN" or "OUT". What is the size (number of rows and columns) of this dataset?

I have loaded libraries - tibble , dplyr to answer this question

Step 1 Load csv file in p variable using **read.csv** function.

Step 2 Convert variable into tibble using **as.tibble** function.

step 3 Make p such that it does not have columns with column names ending with IN or OUT , used **select** function from dplyr with **"-"** and **ends_with** statement to deselect those columns .

Step 4 : Used method **nrow()** and **ncol()** to get the size of data loaded in p.

Ans :- Size of dataset loaded is as below:

1. Number of rows:- **8760**
2. Number of Columns :- **27**

```r
library(dplyr)
library(tibble)
#load file and convert it to tibble
p = read.csv('pedestrian_2023.csv', check.names = FALSE)
p = as.tibble(p)
p = p |>
  select(-ends_with(c('IN','OUT')))
cat("Number of columns in dataframe:",ncol(p),"\n")
```

```
Number of columns in dataframe: 27
```

```r
cat("Number of rows in dataframe:",nrow(p),"\n")
```

```
Number of rows in dataframe: 8760
```

2. Write some code to check that the variable Time is stored using an appropriate class for a date, and the other variables are numeric, fix them if they aren't.

On checking variables class using **sapply()** i found Time variable is loaded as **chr ~ character** and all other variables are of type **int ~ integer** , we need to do type casting to make it date and numeric respectively.

```r
knitr::kable(sapply(p,class))
```

	x
Time	character
Aston Quay/Fitzgeralds	integer
Baggot st lower/Wilton tce inbound	integer
Baggot st upper/Mespil rd/Bank	integer
Capel st/Mary street	integer
College Green/Bank Of Ireland	integer
College Green/Church Lane	integer
College st/Westmoreland st	integer
D'olier st/Burgh Quay	integer
Dame Street/Londis	integer
Grafton st/Monsoon	integer
Grafton Street / Nassau Street / Suffolk Street	integer
Grafton Street/CompuB	integer
Grand Canal st upp/Clanwilliam place	integer

	x
Grand Canal st upp/Clanwilliam place/Google	integer
Henry Street/Coles Lane/Dunnes	integer
Mary st/Jervis st	integer
North Wall Quay/Samuel Beckett bridge East	integer
North Wall Quay/Samuel Beckett bridge West	integer
O'Connell St/Parnell St/AIB	integer
O'Connell st/Princes st North	integer
Phibsborough Rd/Enniskerry Road	integer
Richmond st south/Portabello Harbour inbound	integer
Richmond st south/Portabello Harbour outbound	integer
Talbot st/Guineys	integer
Westmoreland Street East/Fleet street	integer
Westmoreland Street West/Carrolls	integer

I will use **mutate** function from dplyr package to to data type conversion for 26 columns first.

for Time i did separate conversion using **dmy_hms** from **lubridate** package format will be default to **YYYY-MM-DD HH:MM:SS TZ** , and for other 26 columns i used across option and deselected Time and did conversion using **as.numeric** function . This will convert the character to **POSIXct** class, this represent date and time objects in R.

Recheck the data types using **str()** function , now all variables are modified as per requirement i.e Time is in POSIXct and other 26 columns are in numeric format.

```
library(lubridate)
#type casting columns other than Time
p <- p |>
  mutate(
    across(-Time, as.numeric)
  )

#type casting Time column separately
p$Time = dmy_hm(p$Time,tz = Sys.timezone())

p_class = sapply(p,class)

str(p_class)
```

```
List of 27
 $ Time                                          : chr [1:2] "POSIXct" "POSIXt"
```

```
$ Aston Quay/Fitzgeralds                           : chr "numeric"
$ Baggot st lower/Wilton tce inbound               : chr "numeric"
$ Baggot st upper/Mespil rd/Bank                   : chr "numeric"
$ Capel st/Mary street                             : chr "numeric"
$ College Green/Bank Of Ireland                    : chr "numeric"
$ College Green/Church Lane                        : chr "numeric"
$ College st/Westmoreland st                       : chr "numeric"
$ D'olier st/Burgh Quay                            : chr "numeric"
$ Dame Street/Londis                               : chr "numeric"
$ Grafton st/Monsoon                               : chr "numeric"
$ Grafton Street / Nassau Street / Suffolk Street: chr "numeric"
$ Grafton Street/CompuB                            : chr "numeric"
$ Grand Canal st upp/Clanwilliam place             : chr "numeric"
$ Grand Canal st upp/Clanwilliam place/Google      : chr "numeric"
$ Henry Street/Coles Lane/Dunnes                   : chr "numeric"
$ Mary st/Jervis st                                : chr "numeric"
$ North Wall Quay/Samuel Beckett bridge East       : chr "numeric"
$ North Wall Quay/Samuel Beckett bridge West       : chr "numeric"
$ O'Connell St/Parnell St/AIB                      : chr "numeric"
$ O'Connell st/Princes st North                    : chr "numeric"
$ Phibsborough Rd/Enniskerry Road                  : chr "numeric"
$ Richmond st south/Portabello Harbour inbound     : chr "numeric"
$ Richmond st south/Portabello Harbour outbound    : chr "numeric"
$ Talbot st/Guineys                                : chr "numeric"
$ Westmoreland Street East/Fleet street            : chr "numeric"
$ Westmoreland Street West/Carrolls                : chr "numeric"
```

3. Load the dataset weather_2023.txt, save it as a tibble. Give meaningful names to the variables related to the weather. What is the size (number of rows and columns) of this dataset ?

- To load text file i am using **read.delim** function and provided separator as tab - **'\t'**
- to save it as a tibble i am using **as.tibble** function.
- to rename i am using **colnames()** function .
- to get the size i have used **str()** function which provides me
    1. Number of Rows : **8760**
    2. Number of Columns : **5**

```
library(tibble)
#load data from csv and save as tibble
w = read.delim('weather_2023.txt',sep = '\t')
w = as.tibble(w)
colnames(w) = c("TimeStamp","PrecipitationAmount(mm)","AirTemperature(C)"
                ,"MeanWindSpeed(kt)","CloudAmount(okta)")

str(w, width = 80, strict.width = "wrap")
```

```
tibble [8,760 x 5] (S3: tbl_df/tbl/data.frame)
$ TimeStamp : chr [1:8760] "2023-01-01T00:00:00Z" "2023-01-01T01:00:00Z"
   "2023-01-01T02:00:00Z" "2023-01-01T03:00:00Z" ...
$ PrecipitationAmount(mm): num [1:8760] 0.3 0 0 0 0 0 0 0 0.2 0 ...
$ AirTemperature(C) : num [1:8760] 6.7 6.2 4.9 4.4 4.1 4.4 4 4.3 5.2 5.5 ...
$ MeanWindSpeed(kt) : int [1:8760] 6 8 7 5 3 3 3 4 3 3 ...
$ CloudAmount(okta) : int [1:8760] 7 6 3 6 6 3 6 7 6 7 ...
```

4. Convert the variable containing the cloud amount information into an ordered factor. Print the levels and the output of a check to confirm it's ordered.

I have used **factor()** with **ordered =TRUE** to convert a column to an ordered factor , to print the levels in factor i have used **levels()** , and to check if its ordered or not i have used **is.ordered()** which will give output in boolean form TRUE if ordered or False if not in order.

```
#Factorise column CloudAmount
CloudAmount = factor(w$`CloudAmount(okta)`,ordered = TRUE)
print(levels(CloudAmount))
```

```
[1] "0" "1" "2" "3" "4" "5" "6" "7" "8"
```

```
print(is.ordered(CloudAmount))
```

```
[1] TRUE
```

5. Use the function skim_without_charts() from the package skimr on this weather dataset, and briefly explain in your own words what the function is doing.

This function skims over a data frame, and provide useful general and statistical information about it ,such as

1. Number of rows & columns ,column type frequency i.e what are the different data types of columns and the frequency or number of columns having particular data type in data frame .
2. For particular column Time - missing values count ,completeness status ,min/max ,unique value count & whitespace count.
3. Statistical summary like mean , standard deviation , 0 , 25 ,50 ,75 & 100 percentile for all columns .

```
library(skimr)
skim_without_charts(w)
```

Table 2: Data summary

Name	w
Number of rows	8760
Number of columns	5
_____	
Column type frequency:	
character	1
numeric	4
_____	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
TimeStamp	0	1	20	20	0	8760	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
PrecipitationAmount(mm)	0	1	0.11	0.50	0.0	0.0	0.0	0.0	10.6
AirTemperature(C)	0	1	10.74	4.85	-4.3	7.4	10.7	14.3	25.5
MeanWindSpeed(kt)	0	1	8.99	4.21	0.0	6.0	9.0	11.0	31.0
CloudAmount(okta)	0	1	5.62	2.10	0.0	5.0	7.0	7.0	8.0

6. Check that the variable Time in the weather dataset is of an appropriate class for a date (fix it if it isn't), and confirm that the range of Time in the two dataset is the same.

Check the class of all variables using **sapply().**

```
knitr::kable(sapply(w, class))
```

	x
TimeStamp	character
PrecipitationAmount(mm)	numeric
AirTemperature(C)	numeric
MeanWindSpeed(kt)	integer
CloudAmount(okta)	integer

I will change class for columns TimeStamp to type **POSIXct** using **ymd_hms()** and columns MeanWindSpeed(kt) ,CloudAmount(okta) to numeric using **as.numeric()**

```
library(dplyr)
#type casting columns to numeric and Timestamp column to datetime format
w <- w |>
  mutate(
    across(c(`MeanWindSpeed(kt)`,`CloudAmount(okta)`), as.numeric),
    TimeStamp = ymd_hms(w$TimeStamp, tz = Sys.timezone())
  )

w_class = sapply(w,class)

str(w_class)
```

```
List of 5
 $ TimeStamp              : chr [1:2] "POSIXct" "POSIXt"
 $ PrecipitationAmount(mm): chr "numeric"
 $ AirTemperature(C)      : chr "numeric"
 $ MeanWindSpeed(kt)      : chr "numeric"
 $ CloudAmount(okta)      : chr "numeric"
```

to get the range of date column from both datasets use **range()** ,it is confirm that the range is same for both .

```r
range(p$Time)
```

```
[1] "2023-01-01 00:00:00 GMT" "2023-12-31 23:00:00 GMT"
```

```r
range(w$TimeStamp)
```

```
[1] "2023-01-01 00:00:00 GMT" "2023-12-31 23:00:00 GMT"
```

7. Join the two dataset. What is the size (number of rows and columns) this dataset?

I am using **merge()** to join 2 datasets on column Time from pedestrian data and column TimeStamp from weather .

Size of merged data is as:

- Number of Columns - **31**
- Number of Rows - **8760**

```r
# join datsets w & p pn column Time from p & timestamp from w
d= merge(p,w,by.x = 'Time' ,by.y = 'TimeStamp')
cat("Number of Columns in merged data",ncol(d),"\n")
```

```
Number of Columns in merged data 31
```

```r
cat("Number of Rows in merged data",nrow(d),"\n")
```

```
Number of Rows in merged data 8760
```

8. Add two columns one containing the name of the day of the week and the other the month. Check that these two columns are ordered factors.

I am using **wday()** and **month()** function from lubridate package to get week day and month name

using **class()** function we checked both new added columns are ordered factor.

```r
#add 2 columns for weekday & month name
d$day_of_week = wday(d$Time,label = TRUE)
d$month_name = month(d$Time,label = TRUE)

class(d$month_name)
```

```
[1] "ordered" "factor"
```

```r
class(d$day_of_week)
```

```
[1] "ordered" "factor"
```

9. Use dplyr::relocate() to put the new columns with the month and day of the week as the second and third columns of the dataset. Print the column names.

To relocate i have used .after = Time to put the 2 columns at 2nd & 3rd place in column list as Time is the 1st column . to print column names i am using **colnames()** function.

```r
#chnage the location of columns using relocate
d = d |>
  relocate(month_name,day_of_week, .after = Time)

print(colnames(d))
```

```
 [1] "Time"
 [2] "month_name"
 [3] "day_of_week"
 [4] "Aston Quay/Fitzgeralds"
 [5] "Baggot st lower/Wilton tce inbound"
 [6] "Baggot st upper/Mespil rd/Bank"
 [7] "Capel st/Mary street"
 [8] "College Green/Bank Of Ireland"
 [9] "College Green/Church Lane"
[10] "College st/Westmoreland st"
[11] "D'olier st/Burgh Quay"
[12] "Dame Street/Londis"
[13] "Grafton st/Monsoon"
[14] "Grafton Street / Nassau Street / Suffolk Street"
[15] "Grafton Street/CompuB"
[16] "Grand Canal st upp/Clanwilliam place"
```

```
[17] "Grand Canal st upp/Clanwilliam place/Google"
[18] "Henry Street/Coles Lane/Dunnes"
[19] "Mary st/Jervis st"
[20] "North Wall Quay/Samuel Beckett bridge East"
[21] "North Wall Quay/Samuel Beckett bridge West"
[22] "O'Connell St/Parnell St/AIB"
[23] "O'Connell st/Princes st North"
[24] "Phibsborough Rd/Enniskerry Road"
[25] "Richmond st south/Portabello Harbour inbound"
[26] "Richmond st south/Portabello Harbour outbound"
[27] "Talbot st/Guineys"
[28] "Westmoreland Street East/Fleet street"
[29] "Westmoreland Street West/Carrolls"
[30] "PrecipitationAmount(mm)"
[31] "AirTemperature(C)"
[32] "MeanWindSpeed(kt)"
[33] "CloudAmount(okta)"
```

Task2 : Analysis

1. Use functions from base R to compute which months had the highest and the lowest overall pedestrian traffic (i.e. total pedestrian traffic in all location in the whole month).

I have used **sapply** to get all numeric columns from pedestrian dataset , then created a column named total_traffic which is **rowsum()** of all numeric columns in that row this have value shows total pedestrain traffic .

using **aggregate()** function to group data by month name , using **which.max** & **which.min** to get index for minimum & maximum value in data.

```r
# get all numeric column from dataframe
numeric_columns <- sapply(p, is.numeric)

#create column with total_traffic information using rowsum
p$total_traffic <- as.numeric(rowSums(p[,numeric_columns],na.rm = TRUE))

month_name <- factor(month(p$Time,label = TRUE))

# using aggregate we will create a value that holds total_traffic
#value grouping by each month
traffic_by_month <- aggregate(total_traffic ~ month_name, data = p, sum)
```

```
df= data.frame(traffic_by_month)

#using which.max /which.min  to get index for row which holds max or min value
max_row = which.max(df$total_traffic)
min_row = which.min(df$total_traffic)

#load the value of max & min information
max_month = df[max_row,]
min_month = df[min_row,]

#print the lowest & highest traffic facing month
cat("Month of" ,as.character(max_month$month_name)
    ,"has highest traffic of",max_month$total_traffic,"\n")
```

```
Month of Mar has highest traffic of 25002430
```

```
cat("Month of",as.character(min_month$month_name)
    ,"has lowest traffic of",min_month$total_traffic)
```

```
Month of Jun has lowest traffic of 15128010
```

2. Use ggplot2 to create a plot displaying three time series of daily pedestrian footfall in three locations of your choice. Add two vertical bars to mark St. Patrick's day and Christmas Day.

It is observed that on these 2 days footfall is going down when compared to other consecutive days.

Here i am pivoting data to put all 3 selected locations in a column Location and its respective value for each row in column footfall using **pivot_longer()**

then using this new dataset to group by each date & location , show summary as sum of footfall for data summed up from each hour information in a day and putting the data in **p_pivot** which i will use for plotting.

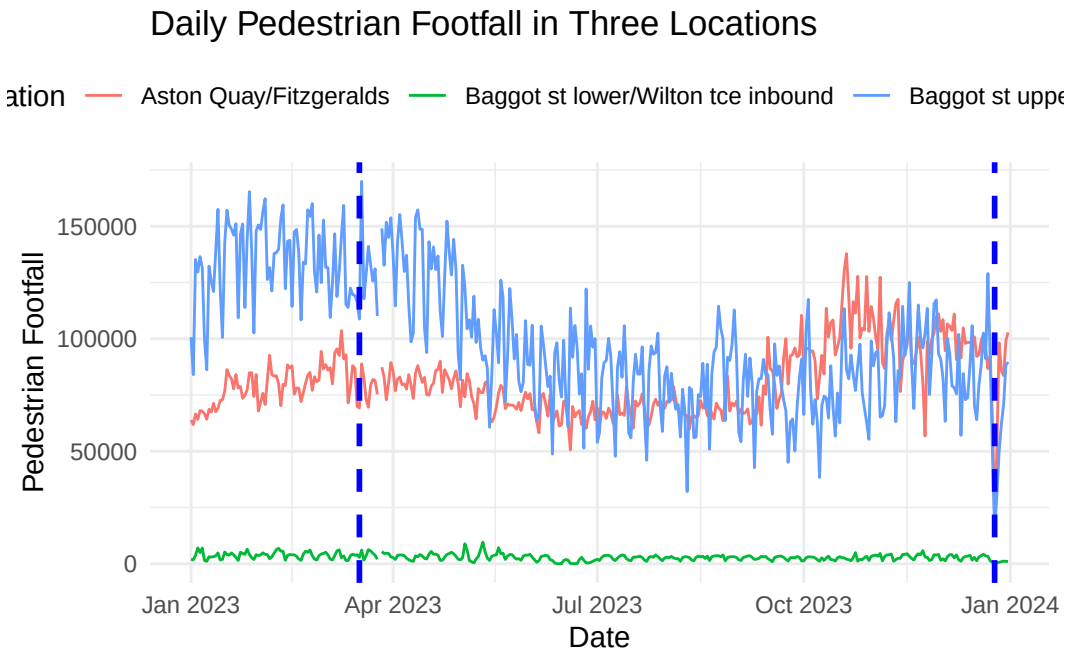
using **ggplot() + geom_line()** to plot a line chart showing time series for 3 locationshowing date on x-axis and total footfall in Y axis and used **geom_vline()** to add 2 vertical lines to show the bar for Christmas & St.Patrick's day

```r
library(ggplot2)
library(dplyr)
library(tidyr)

#pivot data to have each location value in rows .
p_pivot <- p |>
  select(c("Time","Aston Quay/Fitzgeralds"
                        ,"Baggot st lower/Wilton tce inbound"
                       ,"Baggot st upper/Mespil rd/Bank"))  |>
  pivot_longer(cols = c("Aston Quay/Fitzgeralds"
                        ,"Baggot st lower/Wilton tce inbound"
                       ,"Baggot st upper/Mespil rd/Bank"),
               names_to = "Location",
               values_to = "Footfall")

#group by date & location and sum footfall
p_pivot <- p_pivot  |>
            group_by(Time = as.Date(Time),Location) |>
            summarize(Total = sum(Footfall))

#plot graph with 3 location time series showing total sum of foootfall
ggplot(p_pivot, aes(x = Time, y = Total, color = Location)) +
  geom_line() +
  geom_vline(xintercept = as.Date("2023-03-17"), linetype = "dashed"
             , color = "blue", size = 1) +  # St. Patrick's Day
  geom_vline(xintercept = as.Date("2023-12-25"), linetype = "dashed"
             , color = "blue", size = 1) +  # Christmas Day
  labs(
    title = "Daily Pedestrian Footfall in Three Locations",
    x = "Date",
    y = "Pedestrian Footfall"
  ) +
  theme_minimal() +
  theme(legend.position = "top")
```



3. Create a table displaying the minimum and maximum temperature, the mean daily precipitation amount, and the mean daily wind speed by season (Winter: December to February, Spring: March to May, Summer: June to August, and Autumn: September to November).

I have created a new column using mutate() and **case_when()** statement . group the data by season column and **summarise** max & min for Airtemperature and mean for rain and windspeed columns .I have then put this all modification in new dataframe **w_summary** to keep the old one as is

```
#add column season using case when statement for month from Time column
w_summary <- w |>
  mutate(Season = case_when(
      month(TimeStamp) %in% c(12, 1, 2) ~ "Winter",  # Dec, January, February
      month(TimeStamp) %in% c(3, 4, 5) ~ "Spring",   # March, April, May
      month(TimeStamp) %in% c(6, 7, 8) ~ "Summer",   # June, July, August
      month(TimeStamp) %in% c(9, 10, 11) ~ "Autumn",   # Sep, Oct, Nov
      TRUE ~ "Unknown"  # In case of unexpected values
    )) |>
  group_by(Season) |>  # Group by Season
  summarize(
    Max_Temp = max(`AirTemperature(C)`, na.rm = TRUE),  # Max of Col1
    Min_Temp = min(`AirTemperature(C)`, na.rm = TRUE),  # Min of Col1
```

```
    Mean_Rain = mean(`PrecipitationAmount(mm)`, na.rm = TRUE), # Mean of Col2
    Mean_WindSpeed = mean(`MeanWindSpeed(kt)`, na.rm = TRUE)
  )

knitr::kable(w_summary)
```

Season	Max_Temp	Min_Temp	Mean_Rain	Mean_WindSpeed
Autumn	25.5	-2.0	0.1486957	8.343249
Spring	19.4	-4.3	0.1053466	8.634798
Summer	24.0	3.8	0.1439764	8.504076
Winter	13.8	-4.3	0.0584259	10.495833

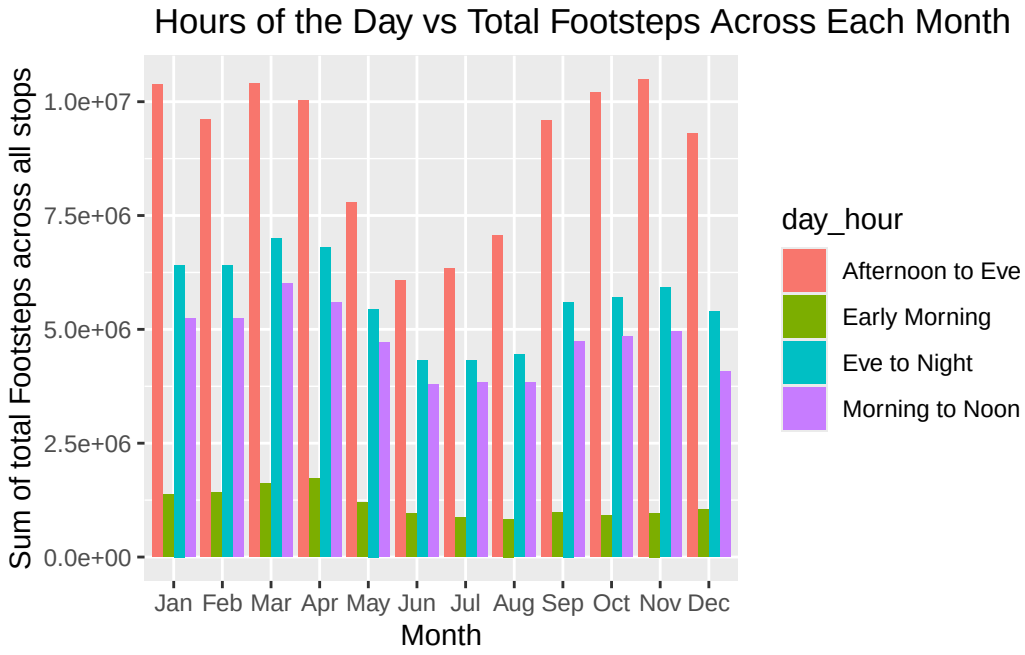
Task3 :Creativity

Visualisation 1 :- Plot

This plot is showing which time of day is having maximum traffic across each month , and we can observe in graph that Traffic is always at its peak in Afternoon to Evening buckets i.e 12 to 17 pm .

```
p_summary <- p |>
  mutate(day_hour = case_when(
      hour(Time) %in% c(0:5) ~ "Early Morning",
      hour(Time) %in% c(6:11) ~ "Morning to Noon",
      hour(Time) %in% c(12:17) ~ "Afternoon to Eve",
      hour(Time) %in% c(18:23) ~ "Eve to Night",   # Sep, Oct, Nov
      TRUE ~ "Unknown"  # In case of unexpected values
    )) |>
  group_by(day_hour,month_name= month(Time,label= TRUE )) |>
  summarize(
    traffic_by_hour_bucket = sum(total_traffic, na.rm = TRUE),
      )

ggplot(p_summary, aes(x = month_name, y = traffic_by_hour_bucket,
                      fill = day_hour)) +  geom_bar(stat = "identity"
                                                    , position = "dodge") +
  labs(title = " Hours of the Day vs Total Footsteps Across Each Month",
       x = "Month", y = "Sum of total Footsteps across all stops")
```



Visualisation 2 :- Table

This table shows Weekdays vs Weekend Traffic scenario at 2 stations across all months in a year and we notice that there is always more footsteps observed on weekdays when compared against traffic on weekend.

```
p_summary1 <- p  |>
  mutate(day_flag = case_when(
      weekdays(Time) %in% c('Saturday','Sunday') ~ "Wknd",
      weekdays(Time) %in% c('Monday','Tuesday','Wednesday'
                            ,'Thursday','Friday') ~ "Wkdy",
      TRUE ~ "Unknown"  # In case of unexpected values
    )
  ) |>
  select(c("Time","day_flag","Aston Quay/Fitzgeralds"
                        ,"Baggot st upper/Mespil rd/Bank"))  |>
  pivot_longer(cols = c("Aston Quay/Fitzgeralds"
                        ,"Baggot st upper/Mespil rd/Bank"),
               names_to = "Location",
               values_to = "Footfall") |>
  group_by(m_name = month(Time,label= TRUE),day_flag,Location) |>
summarize(
    traffic_by_day_flag = sum(Footfall,na.rm = TRUE),  # Max of Col1
```

```
      )|>
  arrange(desc(traffic_by_day_flag))

#knitr::kable(p_summary1)

reshaped_data <- p_summary1 |>
  pivot_wider(
    names_from = c( Location ,day_flag),
    values_from = traffic_by_day_flag,
    names_glue = "{substr(Location,1,10)}_{day_flag}"
  )

knitr::kable(reshaped_data,align = 'ccc')
```

m_name	Baggot st _Wkdy	Aston Quay_Wkdy	Baggot st _Wknd	Aston Quay_Wknd
Mar	3091693	1926469	958416	659436
Jan	3056461	1620700	1010838	661672
Feb	2865516	1614362	1036056	647647
Apr	2785796	1627885	1112237	821448
May	2342011	1686283	719446	584523
Oct	1813557	2319283	607816	900231
Nov	2190824	2125906	626397	821389
Jun	2067252	1461627	539432	535860
Dec	1659767	1932233	759365	947708
Jul	1896936	1442495	591820	670853
Aug	1878999	1595239	576738	548185
Sep	1686857	1642230	561835	735942