Note

Getting Organized

First Loop

Second Loop

Cleaning Up Time and Date

What is the Most Popular Item?

When Are Men’s Basketballs Checked Out?

Final Comparison

Gym Equipment Cleaning Project

Ben Miller

2023-10-11

Note

I have obtained permission from my boss to publish this data. To ensure anonymity, I replaced the various student names and emails from the original data set with ‘NAME’ and ‘EMAIL.’ The data is otherwise unchanged from the original project, other than the redacted student information.

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.2.3
```

```
## Warning: package 'ggplot2' was built under R version 4.2.3
```

```
## Warning: package 'tibble' was built under R version 4.2.3
```

```
## Warning: package 'tidyr' was built under R version 4.2.3
```

```
## Warning: package 'readr' was built under R version 4.2.3
```

```
## Warning: package 'purrr' was built under R version 4.2.3
```

```
## Warning: package 'dplyr' was built under R version 4.2.3
```

```
## Warning: package 'forcats' was built under R version 4.2.3
```

```
## Warning: package 'lubridate' was built under R version 4.2.3
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.2     ✔ readr     2.1.4
## ✔ forcats   1.0.0     ✔ stringr   1.5.0
## ✔ ggplot2   3.4.2     ✔ tibble    3.2.1
## ✔ lubridate 1.9.2     ✔ tidyr     1.3.0
## ✔ purrr     1.0.1
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the ]8;;http://conflicted.r-lib.org/conflicted package]8;; to force all conflicts to become errors
```

```
library(janitor)
```

```
## Warning: package 'janitor' was built under R version 4.2.3
```

```
##
## Attaching package: 'janitor'
##
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```
library(ggthemes)
```

```
## Warning: package 'ggthemes' was built under R version 4.2.3
```

```
data <- read.csv("C:/Users/Ben/Downloads/Campus Rec/Equipment Data/MessyEquipData1.csv") #import data
```

This data is incredibly messy. It'll need to be cleaned if I want to analyze it.

Getting Organized

The first three rows of the data frame are titles from the csv that are unnecessary for my analysis. In the below chunk of code I will remove those rows using the slice function.

```
n <- nrow(data) #calculates number of rows in data set
clean_data <- data |>
  slice(4:n) #create a new data frame, clean_data, that only contains rows 4 through n
```

Nice! I have removed the first three rows from the data frame.

The removed three rows came before our column names in the csv file. As a result, when I imported the csv into R, R imported the wrong column names. After removing those rows the first row of clean_data now contains the column names. Let's make that first row our columns using the row-to_names function from the janitor package.

```
clean_data <- clean_data |>
  row_to_names(row_number = 1) #take the values from row #1 and make them columns
head(clean_data)
```

```
##                           Customer      Checked Out\n(User)
## 2 Category: Recreational Equipment
## 3             Equipment: Ab Roller
## 4                         Item: AB1
## 5                              NAME  Wed, Sep 20, 2023 8:35A
## 6
## 7                              NAME Sun, Sep 24, 2023 10:26A
##                   Due Date       Checked In\n(User) Overdue
## 2
## 3
## 4
## 5 Wed, Sep 20, 2023 11:59P  Wed, Sep 20, 2023 8:48A
## 6
## 7 Sun, Sep 24, 2023 11:59P Sun, Sep 24, 2023 10:37A
```

Let's rename the columns so they are easier to work with in R.

```
clean_data <- clean_data |>
  rename(CheckedOut = `Checked Out
(User)`)

clean_data <- clean_data |>
   rename(CheckedIn = `Checked In
(User)`)
```

Nice! Next, I'll want to remove the 'Due Date' and 'Overdue' columns from the data set, as I don't really care about those in my future analysis.

```
clean_data <- clean_data |>
  select(Customer,CheckedOut,CheckedIn)
head(clean_data)
```

```
##                           Customer              CheckedOut
## 2 Category: Recreational Equipment
## 3             Equipment: Ab Roller
## 4                        Item: AB1
## 5                             NAME  Wed, Sep 20, 2023 8:35A
## 6
## 7                             NAME Sun, Sep 24, 2023 10:26A
##                  CheckedIn
## 2
## 3
## 4
## 5  Wed, Sep 20, 2023 8:48A
## 6
## 7 Sun, Sep 24, 2023 10:37A
```

I'm noticing that rows with blank values in the 'Customer' column contain irrelevant information in other columns. I should remove rows from the data frame where the 'Customer' column does not have a value.

```
clean_data <- clean_data |>
  filter(Customer != "")
head(clean_data)
```

```
##                           Customer              CheckedOut
## 1 Category: Recreational Equipment
## 2             Equipment: Ab Roller
## 3                        Item: AB1
## 4                             NAME  Wed, Sep 20, 2023 8:35A
## 5                             NAME Sun, Sep 24, 2023 10:26A
## 6                             NAME Sun, Sep 24, 2023 12:37P
##                  CheckedIn
## 1
## 2
## 3
## 4  Wed, Sep 20, 2023 8:48A
## 5 Sun, Sep 24, 2023 10:37A
## 6  Sun, Sep 24, 2023 1:32P
```

First Loop

Some rows in the data frame have irrelevant values in the 'Customer' column and no values in the 'CheckedOut' and 'CheckedIn' columns. Among these rows, I need to remove those with 'Customer' values like 'Category: Recreational Equipment' and 'Equipment: NAME OF UNIQUE EQUIPMENT.', as they do not contain relevant information.

However, I can't filter out rows solely based on empty 'CheckedIn' or 'CheckedOut' columns because some rows with missing 'CheckedIn' or 'CheckedOut' data still contain important 'Customer' information. Another issue is that future data will contain dozens of unique values for "Equipment:NAME OF UNIQUE EQUIPMENT", meaning that filtering by every unique "Equipment:NAME OF UNIQUE EQUIPMENT" value will be inefficient and potentially not reproducible.

Given these constraints, I came up with a solution that I believe is both efficient and reproducible for new data. I first use a for loop to identify rows where 'Customer' values start with 'Equipment:'. I then create a temporary column called 'Remove' with a value of 1 for such rows. After that, I filter out rows with a 'Remove' column value of 1. Finally, I remove the 'Remove' column.

```
clean_data <- clean_data |>
  filter(Customer != "Category: Recreational Equipment")

for(i in 1:nrow(clean_data)){
  equipment_vector <- startsWith(clean_data$Customer,"Equipment:")
} #This for loop iterates through every row in clean_data to create a logical vector indicating which rows have 'Cu
stomer' values starting with 'Equipment:'.

clean_data$Remove <- ifelse(equipment_vector == "TRUE",1,0) #This line adds a 'Remove' column in 'clean_data.' It a
ssigns a value of 1 if the corresponding 'equipment_vector' element is "TRUE," otherwise 0.

clean_data <- clean_data |>
  filter(Remove != 1) #filters out rows that have 'Remove' values of 1

clean_data$Remove <- NULL #removes the 'Remove' column
head(clean_data)
```

```
##    Customer              CheckedOut               CheckedIn
## 1 Item: AB1
## 2      NAME  Wed, Sep 20, 2023 8:35A  Wed, Sep 20, 2023 8:48A
## 3      NAME Sun, Sep 24, 2023 10:26A Sun, Sep 24, 2023 10:37A
## 4      NAME Sun, Sep 24, 2023 12:37P  Sun, Sep 24, 2023 1:32P
## 5 Item: AB2
## 6      NAME Tue, Sep 19, 2023 10:27A  Wed, Sep 20, 2023 3:27P
```

Second Loop

Nice! Now is what I found to be the most challenging part of this project…

In the clean_data data frame, the ‘Customer’ column now contains both item names and the names of individuals who checked them out. As you can see by looking at the messy csv, what is now the ‘Customer’ column follows a pattern where each item, such as ‘Item: AB1,’ is followed by the names of the individuals who rented it. My goal is to remove the individual names and retain only the item names in a new ‘Item’ column, effectively restructuring the data frame for analysis.

To achieve this, I created an object, named ‘cur_item,’ by extracting the first value from the ‘Customer’ column in clean_data. (At this point, the first value would be the first ‘Item:NAME OF ITEM’ in the data frame). Then, I made a for loop to iterate through each row in clean_data and check whether the ‘Customer’ column value in the current row starts with “Item:”. If it does, ‘cur_item’ is updated with the value from that row, effectively tracking the current item as the loop progresses through the data frame. Lastly, it populates the ‘Item’ column with the ‘cur_item’ value for each corresponding row.

In summary, the below code runs through each row of clean_data, detects item names in the ‘Customer’ column, and then transfers these item names to corresponding rows in the new ‘Item’ column.

```
clean_data$Item <- NA #Create Item column. Each value is NA for now

clean_data <- clean_data |>
  relocate(Item, .after = Customer) #move around columns for aesthetic purposes

cur_item <- clean_data$Customer[1]
for(i in 1:nrow(clean_data)){
  if(startsWith(clean_data$Customer[i],"Item:")){ #if we reach a new item name
  cur_item <- clean_data$Customer[i] #then replace value of 'cur_item' with the name of that new item
  }
  clean_data$Item[i] <- cur_item #then put the value of cur_item in each row of the new 'Item' column
}

clean_data$Item <- gsub("Item: ","",clean_data$Item) #remove "Item:" from each name in the Item column


clean_data$Customer <- NULL #I no longer need the Customer column


clean_data <- clean_data |>
  filter(CheckedOut != "")
head(clean_data)
```

```
##     Item              CheckedOut               CheckedIn
## 1    AB1  Wed, Sep 20, 2023 8:35A  Wed, Sep 20, 2023 8:48A
## 2    AB1 Sun, Sep 24, 2023 10:26A Sun, Sep 24, 2023 10:37A
## 3    AB1 Sun, Sep 24, 2023 12:37P  Sun, Sep 24, 2023 1:32P
## 4    AB2 Tue, Sep 19, 2023 10:27A  Wed, Sep 20, 2023 3:27P
## 5 BDN-01  Thu, Sep 21, 2023 2:02P  Thu, Sep 21, 2023 2:18P
## 6 BDN-01  Sat, Sep 23, 2023 5:41P  Sat, Sep 23, 2023 8:24P
```

Cleaning Up Time and Date

Great! Now we have a data frame with a properly dedicated Item column.

The ‘Checked Out’ and ‘Checked In’ columns contain information on the day of week, day, year, and time that the item was checked out and in. Unfortunately, this information is all clumped together in each observation in the two columns. I’ll need to separate the text in each column into individual columns for day, date, year, and time.

```
clean_data <- separate(clean_data, CheckedOut, c("Day","Date","Checked Out"), sep = ",") #creates dedicated Day and Date columns.

clean_data <- clean_data |>
  relocate(Day, .before = Item) |>
  relocate(Date, .before = Day)

#Knowing what day and day of week Items were checked back in is not relevant to my analysis. Thus, I can remove that information by first sorting it into NULL columns and then eliminating those columns.
clean_data <- separate(clean_data, CheckedIn, c("NULL","NULL1","Checked In"), sep = ",")
clean_data$`NULL`<- NULL
clean_data$NULL1 <- NULL

#Next, I'll want to create a column for year. While all observations take place in 2023 in this sample data, the original data frame had observations from both 2022 and 2023. As such, I implemented the below for loop to iterate through the data frame and populate a 'Year' column with the correct year value, extracted from the 'Checked In' column.
for(i in 1:nrow(clean_data)){
  clean_data$Year <- ifelse(startsWith(clean_data$`Checked In`," 2022"),2022,2023)
}

#Below I remove the year text from the 'Checked Out' and 'Checked In' columns, as that information is now stored in its own column.
clean_data$`Checked Out` <- gsub(" 2023","",clean_data$`Checked Out`)

clean_data$`Checked In` <- gsub(" 2023","",clean_data$`Checked In`)

clean_data$`Checked Out` <- gsub(" 2022","",clean_data$`Checked Out`)

clean_data$`Checked In` <- gsub(" 2022","",clean_data$`Checked In`)

clean_data <- clean_data |>
  relocate(Year,.before=Date)

head(clean_data)
```

```
##   Year    Date Day   Item Checked Out Checked In
## 1 2023  Sep 20 Wed    AB1       8:35A      8:48A
## 2 2023  Sep 24 Sun    AB1      10:26A     10:37A
## 3 2023  Sep 24 Sun    AB1      12:37P      1:32P
## 4 2023  Sep 19 Tue    AB2      10:27A      3:27P
## 5 2023  Sep 21 Thu BDN-01       2:02P      2:18P
## 6 2023  Sep 23 Sat BDN-01       5:41P      8:24P
```

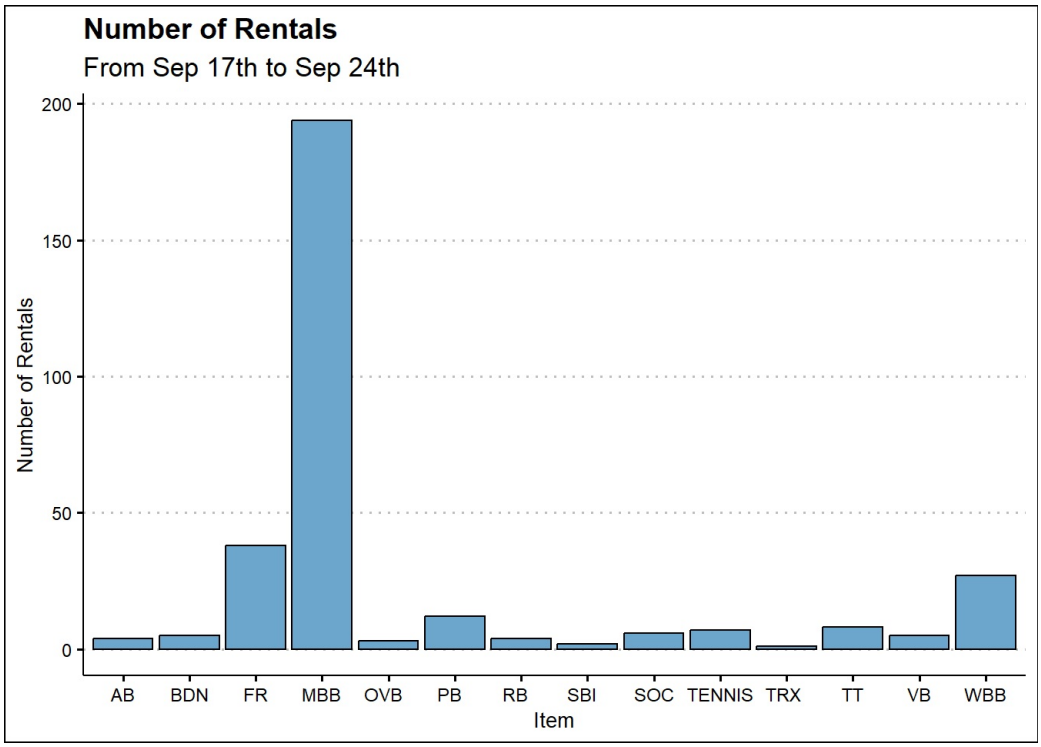
What is the Most Popular Item?

Nice! Now I have a clean data frame that I perform comprehensive analysis on.

Below I'll give two examples for how the cleaned data can be used to generate valuable insights.

Say that I wanted to know what item was the most popular. To answer this question I'll have to do some quick wrangling to remove identification numbers from items so I can easily group them. Then, I'll create a column chart that will answer the question.

```
#the regular expression "\\d" means "any single digit number". Thus, the below line removes identification numbers
from each item.
clean_data$Item <- gsub("\\d","",clean_data$Item)

clean_data$Item <- gsub("-","",clean_data$Item) #removes the "-" that separated the identification numbers from the
item.

#now the 'Item' column contains item information without any identification numbers, making it easy to group by ite
m

#The below code creates a data frame with two columns. 'Item', which has each item name, and 'Count', which contain
s the number of times a row's corresponding 'Item' was checked out. I'll use this data frame to create my visualiza
tion.
Rental_Item_Count <- clean_data |>
  group_by(Item) |>
  summarise(Count = n())

#Below is the code to produce a column chart with each item on the x-axis and the amount of times it was rented out
on the y-axis.

TotalChart <- ggplot(data=Rental_Item_Count,mapping=aes(x=Item,y=Count)) + geom_col(fill="skyblue3",color="black")
+ labs(x="Item",y="Number of Rentals",title = "Number of Rentals",subtitle = "From Sep 17th to Sep 24th") + theme_c
lean()


TotalChart #This chart reveals that men's basketballs are by far the most popular item
```



When Are Men's Basketballs Checked Out?

Now that we know that men's basketballs are the gym's most popular rental item, we may want to find out if there are any notable trends in their checkout patterns. Exploring this could offer valuable insights that would allow us to optimize our open recreation hours. This, in turn, would ensure a more efficient and tailored utilization of resources.

In order to answer this question, I'll first need to create a new column that contains information on the specific time bin in which each item was checked out. Then, I'll filter the data to only include men's basketballs. Finally, I'll create a visualization that displays what time bin has the most basketball rentals.

```
#The code below adds a 'Time' column, which I will populate with values of AM or PM depending on when the item was
checked out. I plan to use this to create separate 'AM' and 'PM' data frames, establish time bins for each observat
ion in those data frames, and then merge them back into a single data frame.

clean_data$Time <- grepl("P",clean_data$`Checked Out`)

clean_data$Time <- gsub("TRUE","PM",clean_data$Time)

clean_data$Time <- gsub("FALSE","AM",clean_data$Time)

PM <- clean_data |>
  filter(Time == "PM")
```

Below, I create individual columns for each time bin, assigning a value of 1 if an item in that row was checked out during that specific time frame. Then, I'll use several for loops to populate a new 'TimeBin' column with corresponding text. For instance, if the 'TwelvetoTwo' column in the PM data frame has a value of 1, the 'TimeBin' value for that row will be assigned the text '12PM-2PM,' and so forth. Following this, I'll eliminate the 'TwelvetoTwo' (and similar) columns. The end result will be the cleaned data frame, now with a 'TimeBin' column that shows what 2 hour time bin each item was checked out during.

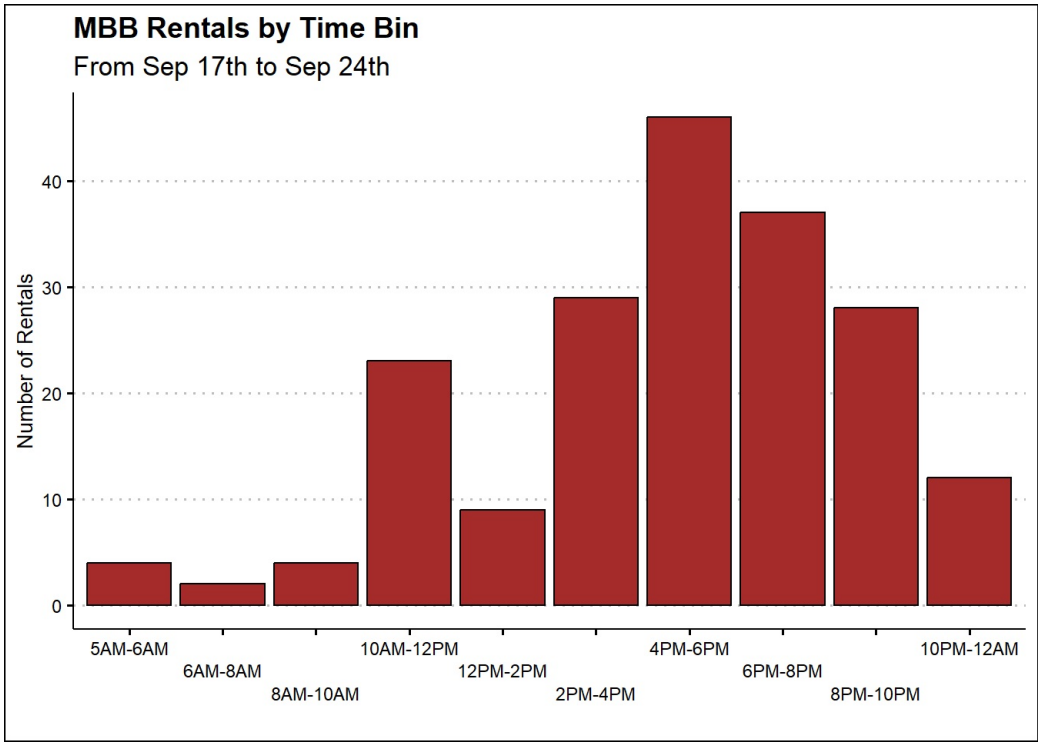
```
PM$TwelveToTwo <- ifelse(startsWith(PM$`Checked Out`, " 12") | startsWith(PM$`Checked Out`, " 1:"), 1, 0)

PM$TwoToFour <- ifelse(startsWith(PM$`Checked Out`, " 2") | startsWith(PM$`Checked Out`, " 3"), 1, 0)

PM$FourToSix <- ifelse(startsWith(PM$`Checked Out`, " 4") | startsWith(PM$`Checked Out`, " 5"), 1, 0)

PM$SixToEight <- ifelse(startsWith(PM$`Checked Out`, " 6") | startsWith(PM$`Checked Out`, " 7"), 1, 0)

PM$EightToTen <- ifelse(startsWith(PM$`Checked Out`, " 8") | startsWith(PM$`Checked Out`, " 9"), 1, 0)

PM$TenToMidnight <- ifelse(startsWith(PM$`Checked Out`, " 10") | startsWith(PM$`Checked Out`, " 11"), 1, 0)

PM$TimeBin <- NA

for(i in 1:nrow(PM)){
if(PM$TwelveToTwo[i] == 1){
  PM$TimeBin[i] <- "12PM-2PM"
}
}

for(i in 1:nrow(PM)){
if(PM$TwoToFour[i] == 1){
  PM$TimeBin[i] <- "2PM-4PM"
}
}

for(i in 1:nrow(PM)){
if(PM$FourToSix[i] == 1){
  PM$TimeBin[i] <- "4PM-6PM"
}
}


for(i in 1:nrow(PM)){
if(PM$SixToEight[i] == 1){
  PM$TimeBin[i] <- "6PM-8PM"
}
}

for(i in 1:nrow(PM)){
if(PM$EightToTen[i] == 1){
  PM$TimeBin[i] <- "8PM-10PM"
}
}

for(i in 1:nrow(PM)){
if(PM$TenToMidnight[i] == 1){
  PM$TimeBin[i] <- "10PM-12AM"
}
}

PM$TwelveToTwo <- NULL
PM$TwoToFour <- NULL
PM$FourToSix <- NULL
PM$SixToEight <- NULL
PM$EightToTen <- NULL
PM$TenToMidnight <- NULL
PM$Time <- NULL

#and now the same process but for AM

AM <- clean_data |>
  filter(Time == "AM")

AM$FiveToSix <- ifelse(startsWith(AM$`Checked Out`, " 5"),1,0)

AM$SixToEight <- ifelse(startsWith(AM$`Checked Out`, " 6") | startsWith(AM$`Checked Out`, " 7"), 1, 0)

AM$EightToTen <- ifelse(startsWith(AM$`Checked Out`, " 8") | startsWith(AM$`Checked Out`, " 9"), 1, 0)

AM$TenToTwelve <- ifelse(startsWith(AM$`Checked Out`, " 10") | startsWith(AM$`Checked Out`, " 11"), 1, 0)
```

```
AM$Time <- NULL


AM$TimeBin <- NA

for(i in 1:nrow(AM)){
if(AM$FiveToSix[i] == 1){
  AM$TimeBin[i] <- "5AM-6AM"
}
}

for(i in 1:nrow(AM)){
if(AM$SixToEight[i] == 1){
  AM$TimeBin[i] <- "6AM-8AM"
}
}

for(i in 1:nrow(AM)){
if(AM$EightToTen[i] == 1){
  AM$TimeBin[i] <- "8AM-10AM"
}
}

for(i in 1:nrow(AM)){
if(AM$TenToTwelve[i] == 1){
  AM$TimeBin[i] <- "10AM-12PM"
}
}

AM$FiveToSeven <- NULL
AM$FiveToSix <- NULL
AM$SixToEight <- NULL
AM$EightToTen <- NULL
AM$TenToTwelve <- NULL

TimeBin_df <- rbind(AM,PM) #merge the two data frames

head(TimeBin_df)
```

```
##   Year    Date Day Item Checked Out Checked In   TimeBin
## 1 2023  Sep 20 Wed   AB       8:35A      8:48A  8AM-10AM
## 2 2023  Sep 24 Sun   AB      10:26A     10:37A 10AM-12PM
## 3 2023  Sep 19 Tue   AB      10:27A      3:27P 10AM-12PM
## 4 2023  Sep 17 Sun   FR      10:22A     10:28A 10AM-12PM
## 5 2023  Sep 18 Mon   FR       9:57A     12:13P  8AM-10AM
## 6 2023  Sep 19 Tue   FR      10:06A     11:25A 10AM-12PM
```

Next, I'll create a column chart that displays the peak rental hours for men's basketballs.

```
TimeBin_df$TimeBin <- factor(TimeBin_df$TimeBin,levels=c("5AM-6AM","6AM-8AM","8AM-10AM","10AM-12PM","12PM-2PM","2PM
-4PM","4PM-6PM","6PM-8PM","8PM-10PM","10PM-12AM"))
#create levels to 'TimeBin' so the x-axis is ordered correctly when creating visualizations

MBB <- TimeBin_df |>
  filter(Item=="MBB") |>
  group_by(TimeBin) |>
  summarise(Count = n())

MBBchart <- ggplot(data=MBB,mapping=aes(x=TimeBin,y=Count)) + geom_col(fill="brown",color="black") + labs(x = "", y
= "Number of Rentals",title = "MBB Rentals by Time Bin",subtitle = "From Sep 17th to Sep 24th") + theme_clean() + s
cale_x_discrete(guide = guide_axis(n.dodge=3))

MBBchart
```



After analyzing the chart, the highest demand for basketball rentals appears to occur between 4pm and 6pm. To enhance the overall efficiency of the gym's open recreation hours, it would be advisable schedule them during this peak period, thus ensuring optimal utilization of the facility and a better experience for our users.

Final Comparison

Below, we can compare the original messy data frame with the wrangled data set.

```
head(data,n=10)
```

```
##                                 Equipment.History.Detail
## 1                                               Parameters
## 2  Sun, Sep 17, 2023 12:00 AM to Sun, Sep 24, 2023 11:59 PM
## 3                       Categories: Recreational Equipment
## 4                                                 Customer
## 5                         Category: Recreational Equipment
## 6                                     Equipment: Ab Roller
## 7                                                 Item: AB1
## 8                                                      NAME
## 9
## 10                                                     NAME
##                          X                     X.1                      X.2
## 1
## 2
## 3
## 4       Checked Out\n(User)                Due Date       Checked In\n(User)
## 5
## 6
## 7
## 8   Wed, Sep 20, 2023 8:35A Wed, Sep 20, 2023 11:59P  Wed, Sep 20, 2023 8:48A
## 9
## 10 Sun, Sep 24, 2023 10:26A Sun, Sep 24, 2023 11:59P Sun, Sep 24, 2023 10:37A
##        X.3
## 1
## 2
## 3
## 4  Overdue
## 5
## 6
## 7
## 8
## 9
## 10
```

```
head(TimeBin_df)
```

```
##   Year    Date Day Item Checked Out Checked In   TimeBin
## 1 2023  Sep 20 Wed   AB       8:35A      8:48A  8AM-10AM
## 2 2023  Sep 24 Sun   AB      10:26A     10:37A 10AM-12PM
## 3 2023  Sep 19 Tue   AB      10:27A      3:27P 10AM-12PM
## 4 2023  Sep 17 Sun   FR      10:22A     10:28A 10AM-12PM
## 5 2023  Sep 18 Mon   FR       9:57A     12:13P  8AM-10AM
## 6 2023  Sep 19 Tue   FR      10:06A     11:25A 10AM-12PM
```