IE 5374 Project - Sec3 - Group16 1

Fengbo Ma Ankita Yadav Zeeshan Ali Shaikh

10/12/2021

Contents

IE 5374 Foundations of Data Analytics

Project 1

Project proposal

The group obtained numbers of data sets about January and February flight information for the years 2019 and 2020. The open sources data sets were provided by the Bureau of Transportation Statistics, Govt. of the USA. The group was managed to provide reasonable analytics and visualizations based on business questions that stockholders might pay special attention to. The group performed a series of analyses on the messy, complex, and real-life air-line data set obtained from the source mentioned above and combined them together in order to visualize it. The group mapped and converted the relevant information into a single data set.

The final data set will be more reliable, organized, convenient to use that include information about origin airport, destination airport, airplane information, departure time and arrival time, etc.

Data Acquisition

Data acquisition is the process of collecting and acquiring data from disparate source systems that capture the real-world physical phenomena and converting them into a digital form that can be manipulated by a computer and software.

Data Wrangling

Discovering

In this step, we explore the dataset, find important trends or missing values, and conceptualize in order to use it.

```
#import data for Jan 2019
df119 = read_csv("Jan_2019_ontime.csv")

#import data for Jan 2020
df120 = read_csv("Jan_2020_ontime.csv")

#import data for Feb 2019
df219 = read_csv("Feb_2019_ontime.csv")

#import data for Feb 2020
df220 = read_csv("Feb_2020_ontime.csv")

#import data frame about air carrier name
dfc = read_csv("IATA Code.csv")

#import data indicating brand code share partners for air carriers
dfbcp = read_csv("BrandedCodesharePartners.csv")
```

Import data

Summary Statisics In this section, we check all the column names in the data frames for January and February for both the years 2019 and 2020 from which we extract the unique carrier id of flights and check whether the data frames consist of NA values, and displaying the number of NA values in each column.

```
#display colume name for all the data frame
colnames(df119)
```

```
##  [1] "DAY_OF_MONTH"          "DAY_OF_WEEK"           "OP_UNIQUE_CARRIER"
##  [4] "OP_CARRIER_AIRLINE_ID" "OP_CARRIER"            "TAIL_NUM"
##  [7] "OP_CARRIER_FL_NUM"     "ORIGIN_AIRPORT_ID"     "ORIGIN_AIRPORT_SEQ_ID"
## [10] "ORIGIN"                "DEST_AIRPORT_ID"       "DEST_AIRPORT_SEQ_ID"
## [13] "DEST"                  "DEP_TIME"              "DEP_DEL15"
## [16] "DEP_TIME_BLK"          "ARR_TIME"              "ARR_DEL15"
## [19] "CANCELLED"             "DIVERTED"              "DISTANCE"
## [22] "...22"
```

```
colnames(df120)
```

```
##  [1] "DAY_OF_MONTH"          "DAY_OF_WEEK"           "OP_UNIQUE_CARRIER"
##  [4] "OP_CARRIER_AIRLINE_ID" "OP_CARRIER"            "TAIL_NUM"
##  [7] "OP_CARRIER_FL_NUM"     "ORIGIN_AIRPORT_ID"     "ORIGIN_AIRPORT_SEQ_ID"
## [10] "ORIGIN"                "DEST_AIRPORT_ID"       "DEST_AIRPORT_SEQ_ID"
## [13] "DEST"                  "DEP_TIME"              "DEP_DEL15"
## [16] "DEP_TIME_BLK"          "ARR_TIME"              "ARR_DEL15"
## [19] "CANCELLED"             "DIVERTED"              "DISTANCE"
## [22] "...22"
```

```
colnames(df219)
```

```
##  [1] "DAY_OF_MONTH"          "DAY_OF_WEEK"           "OP_UNIQUE_CARRIER"
##  [4] "OP_CARRIER_AIRLINE_ID" "OP_CARRIER"            "TAIL_NUM"
##  [7] "OP_CARRIER_FL_NUM"     "ORIGIN_AIRPORT_ID"     "ORIGIN_AIRPORT_SEQ_ID"
## [10] "ORIGIN"                "DEST_AIRPORT_ID"       "DEST_AIRPORT_SEQ_ID"
## [13] "DEST"                  "DEP_TIME"              "DEP_DEL15"
## [16] "DEP_TIME_BLK"          "ARR_TIME"              "ARR_DEL15"
## [19] "CANCELLED"             "DIVERTED"              "DISTANCE"
## [22] "...22"
```

```
colnames(df220)
```

```
##  [1] "DAY_OF_MONTH"          "DAY_OF_WEEK"           "OP_UNIQUE_CARRIER"
##  [4] "OP_CARRIER_AIRLINE_ID" "OP_CARRIER"            "TAIL_NUM"
##  [7] "OP_CARRIER_FL_NUM"     "ORIGIN_AIRPORT_ID"     "ORIGIN_AIRPORT_SEQ_ID"
## [10] "ORIGIN"                "DEST_AIRPORT_ID"       "DEST_AIRPORT_SEQ_ID"
## [13] "DEST"                  "DEP_TIME"              "DEP_DEL15"
## [16] "DEP_TIME_BLK"          "ARR_TIME"              "ARR_DEL15"
## [19] "CANCELLED"             "DIVERTED"              "DISTANCE"
## [22] "...22"
```

```
#display all the air carrier name in the data frame
unique (df119$OP_UNIQUE_CARRIER)
```

```
##  [1] "9E" "AA" "MQ" "G4" "OH" "B6" "YV" "EV" "F9" "YX" "HA" "NK" "OO" "WN" "AS"
## [16] "UA" "DL"
```

```
unique (df120$OP_UNIQUE_CARRIER)
```

```
##  [1] "EV" "WN" "MQ" "B6" "HA" "AA" "F9" "YX" "9E" "YV" "OH" "NK" "DL" "OO" "UA"
## [16] "G4" "AS"
```

```
unique (df219$OP_UNIQUE_CARRIER)
```

```
##  [1] "AA" "NK" "MQ" "G4" "YX" "EV" "F9" "HA" "WN" "9E" "OH" "YV" "AS" "UA" "B6"
## [16] "DL" "OO"
```

```
unique (df220$OP_UNIQUE_CARRIER)
```

```
##  [1] "MQ" "B6" "OH" "AA" "NK" "YX" "F9" "DL" "YV" "EV" "HA" "9E" "G4" "UA" "AS"
## [16] "WN" "OO"
```

```
# first time checking for NA variable
kable(
  df119 %>%
    summarise_all(funs(sum(is.na(.)))) %>%
    t()
,caption="Check for NA January 2019")
```

Table 1: Check for NA January 2019

DAY_OF_MONTH	0
DAY_OF_WEEK	0
OP_UNIQUE_CARRIER	0
OP_CARRIER_AIRLINE_ID	0
OP_CARRIER	0
TAIL_NUM	2543
OP_CARRIER_FL_NUM	0
ORIGIN_AIRPORT_ID	0
ORIGIN_AIRPORT_SEQ_ID	0
ORIGIN	0
DEST_AIRPORT_ID	0
DEST_AIRPORT_SEQ_ID	0
DEST	0
DEP_TIME	16352
DEP_DEL15	16355
DEP_TIME_BLK	0
ARR_TIME	17061
ARR_DEL15	18022
CANCELLED	0
DIVERTED	0
DISTANCE	0
...22	583985

```
kable(
  df120 %>%
    summarise_all(funs(sum(is.na(.))))%>%
    t()
  ,caption="Check for NA January 2020")
```

Table 2: Check for NA January 2020

DAY_OF_MONTH	0
DAY_OF_WEEK	0
OP_UNIQUE_CARRIER	0
OP_CARRIER_AIRLINE_ID	0
OP_CARRIER	0
TAIL_NUM	698
OP_CARRIER_FL_NUM	0
ORIGIN_AIRPORT_ID	0
ORIGIN_AIRPORT_SEQ_ID	0
ORIGIN	0
DEST_AIRPORT_ID	0
DEST_AIRPORT_SEQ_ID	0
DEST	0
DEP_TIME	6664
DEP_DEL15	6699
DEP_TIME_BLK	0
ARR_TIME	7075
ARR_DEL15	8078
CANCELLED	0
DIVERTED	0
DISTANCE	0
...22	607346

```
kable(
  df219 %>%
    summarise_all(funs(sum(is.na(.))))%>%
    t()
,caption="Check for NA February 2019")
```

Table 3: Check for NA February 2019

DAY_OF_MONTH	0
DAY_OF_WEEK	0
OP_UNIQUE_CARRIER	0
OP_CARRIER_AIRLINE_ID	0
OP_CARRIER	0
TAIL_NUM	1393
OP_CARRIER_FL_NUM	0
ORIGIN_AIRPORT_ID	0
ORIGIN_AIRPORT_SEQ_ID	0
ORIGIN	0
DEST_AIRPORT_ID	0
DEST_AIRPORT_SEQ_ID	0

Table 3: Check for NA February 2019

DEST	0
DEP_TIME	14823
DEP_DEL15	14827
DEP_TIME_BLK	0
ARR_TIME	15700
ARR_DEL15	16861
CANCELLED	0
DIVERTED	0
DISTANCE	0
... 22	533175

```
kable(
  df220 %>%
    summarise_all(funs(sum(is.na(.))))%>%
    t()
,caption="Check for NA February 2020")
```

Table 4: Check for NA February 2020

DAY_OF_MONTH	0
DAY_OF_WEEK	0
OP_UNIQUE_CARRIER	0
OP_CARRIER_AIRLINE_ID	0
OP_CARRIER	0
TAIL_NUM	433
OP_CARRIER_FL_NUM	0
ORIGIN_AIRPORT_ID	0
ORIGIN_AIRPORT_SEQ_ID	0
ORIGIN	0
DEST_AIRPORT_ID	0
DEST_AIRPORT_SEQ_ID	0
DEST	0
DEP_TIME	4938
DEP_DEL15	4951
DEP_TIME_BLK	0
ARR_TIME	5287
ARR_DEL15	6192
CANCELLED	0
DIVERTED	0
DISTANCE	0
... 22	574268

```
#NA number exist for all data frames
```

The result could clearly indicate that in the above tables there are a couple of columns which do have NA values.

Discovering Discrete Data In this step, we determine the total number of records of data which is available for us in each csv file using the DAY_OF_MONTH attribute.

```
# count for max number row in the data set using summarize function rather than dimension function
# Summarize function provide only one out out that we need

kable(
  df119 %>%
    summarize(DAY_OF_MONTH=n()),
caption="Number of rows in January 2019 dataset")
```

Table 5: Number of rows in January 2019 dataset

DAY_OF_MONTH
583985

```
kable(
  df120 %>%
    summarize(DAY_OF_MONTH=n()),
caption="Number of rows in January 2020 dataset")
```

Table 6: Number of rows in January 2020 dataset

DAY_OF_MONTH
607346

```
kable(
  df219 %>%
    summarize(DAY_OF_MONTH=n()),
caption="Number of rows in February 2019 dataset")
```

Table 7: Number of rows in February 2019 dataset

DAY_OF_MONTH
533175

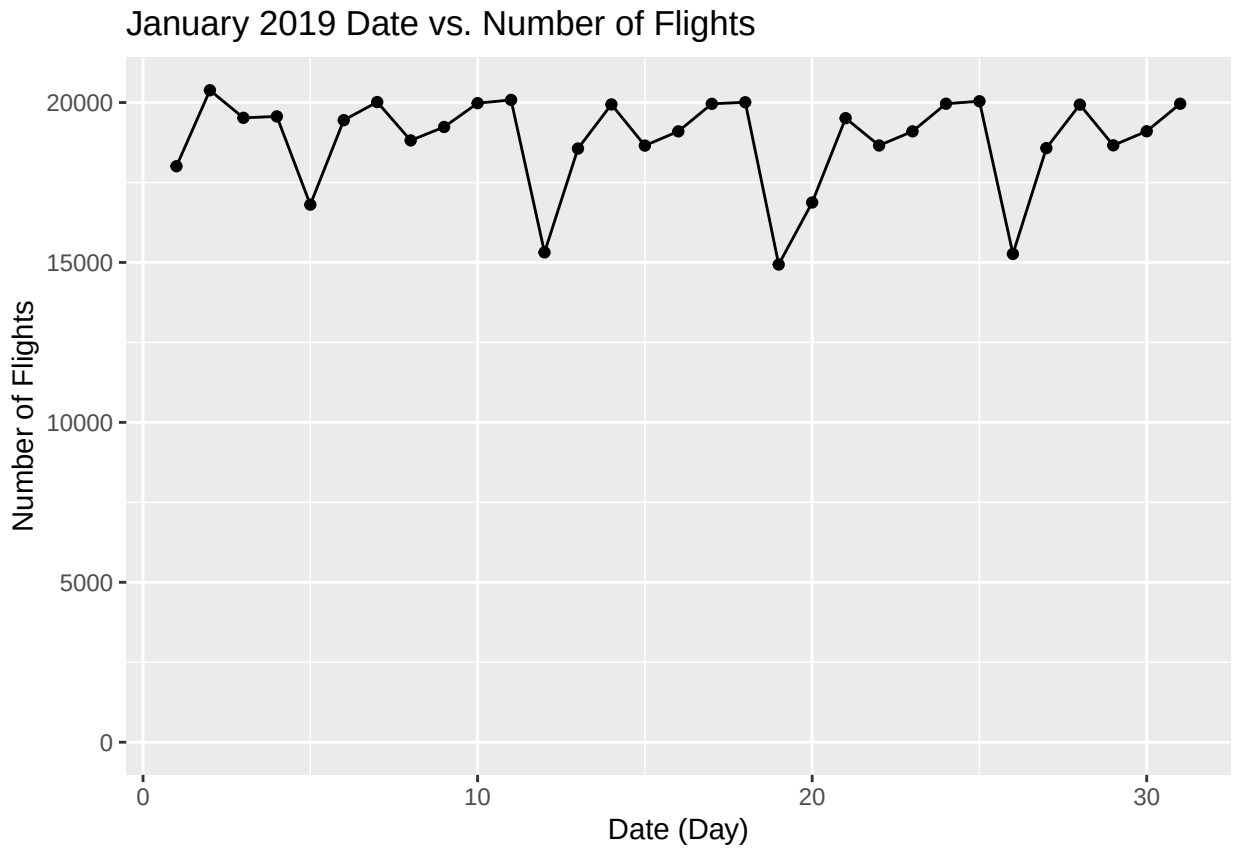
```
kable(
  df220 %>%
    summarize(DAY_OF_MONTH=n()),
caption="Number of rows in February 2020 dataset")
```

Table 8: Number of rows in February 2020 dataset

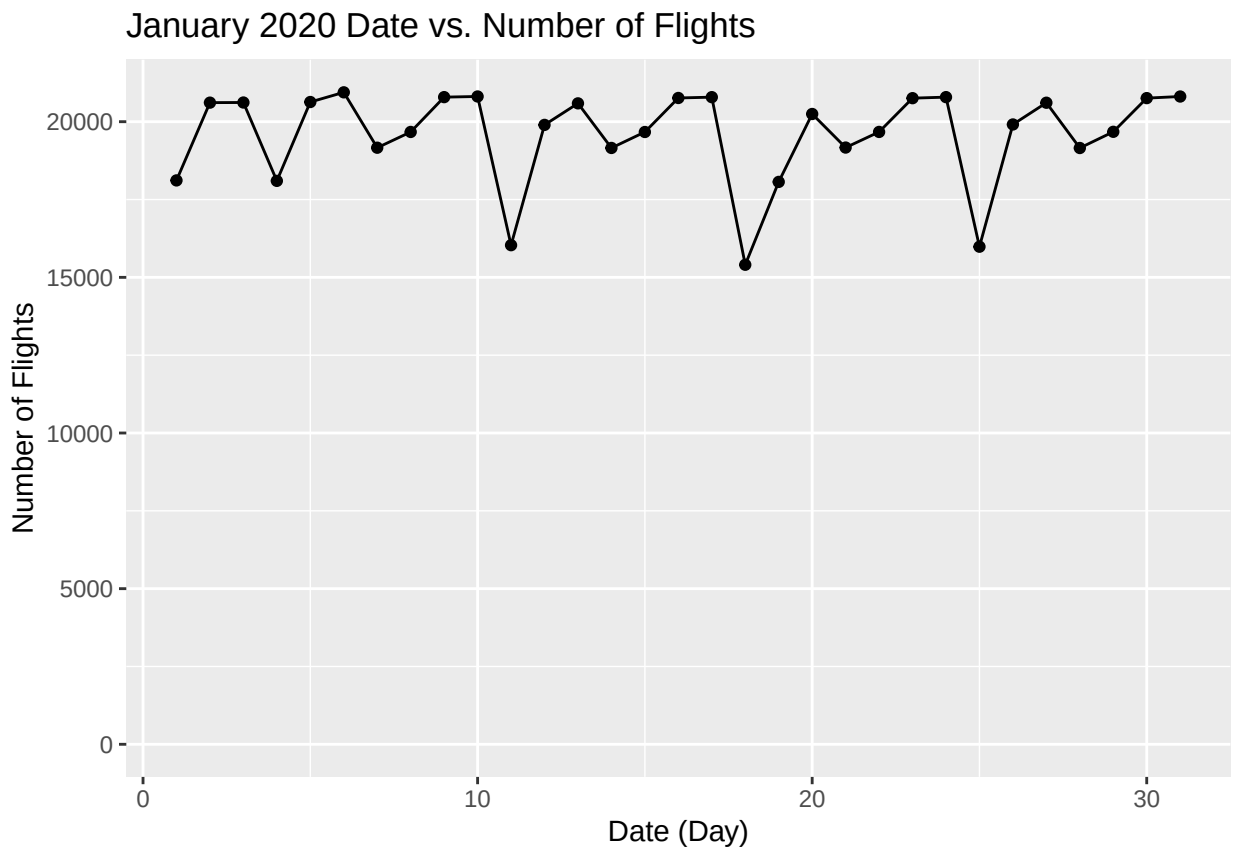
DAY_OF_MONTH
574268

Data discovering using visualizations Displaying the total number of flights using the bar chart for all 30 days for January 2019, January 2020, February 2019, and February 2020.

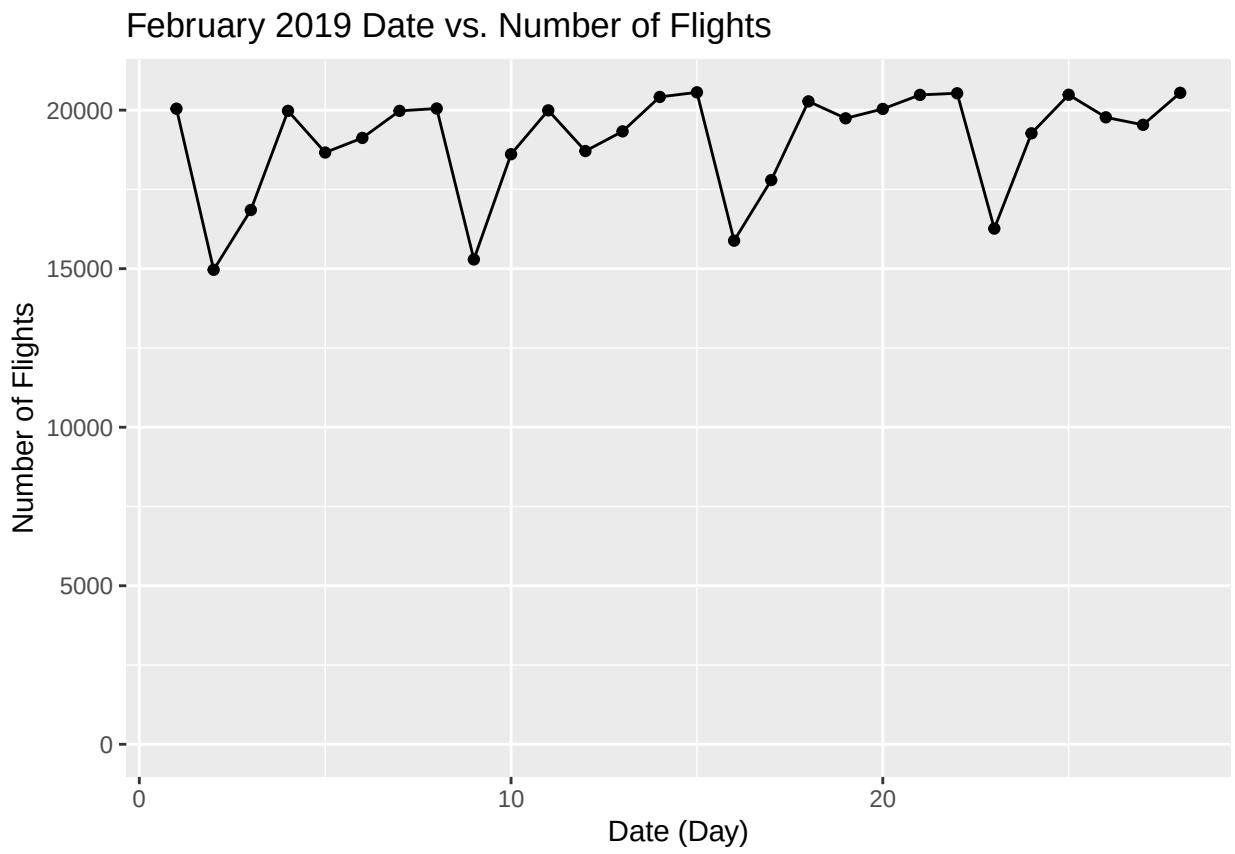
```
#Data discovering using visualizations
#Display number of flight in each day

# plot line chart for Jan 2019
df119 %>%
  count(DAY_OF_MONTH) %>%
  ggplot(aes(x=DAY_OF_MONTH, y=n)) +
  geom_line(stat = "identity") +
  ggtitle("January 2019 Date vs. Number of Flights")+
  geom_point()+
  expand_limits(y=0)+
  labs(x="Date (Day)", y="Number of Flights")
```



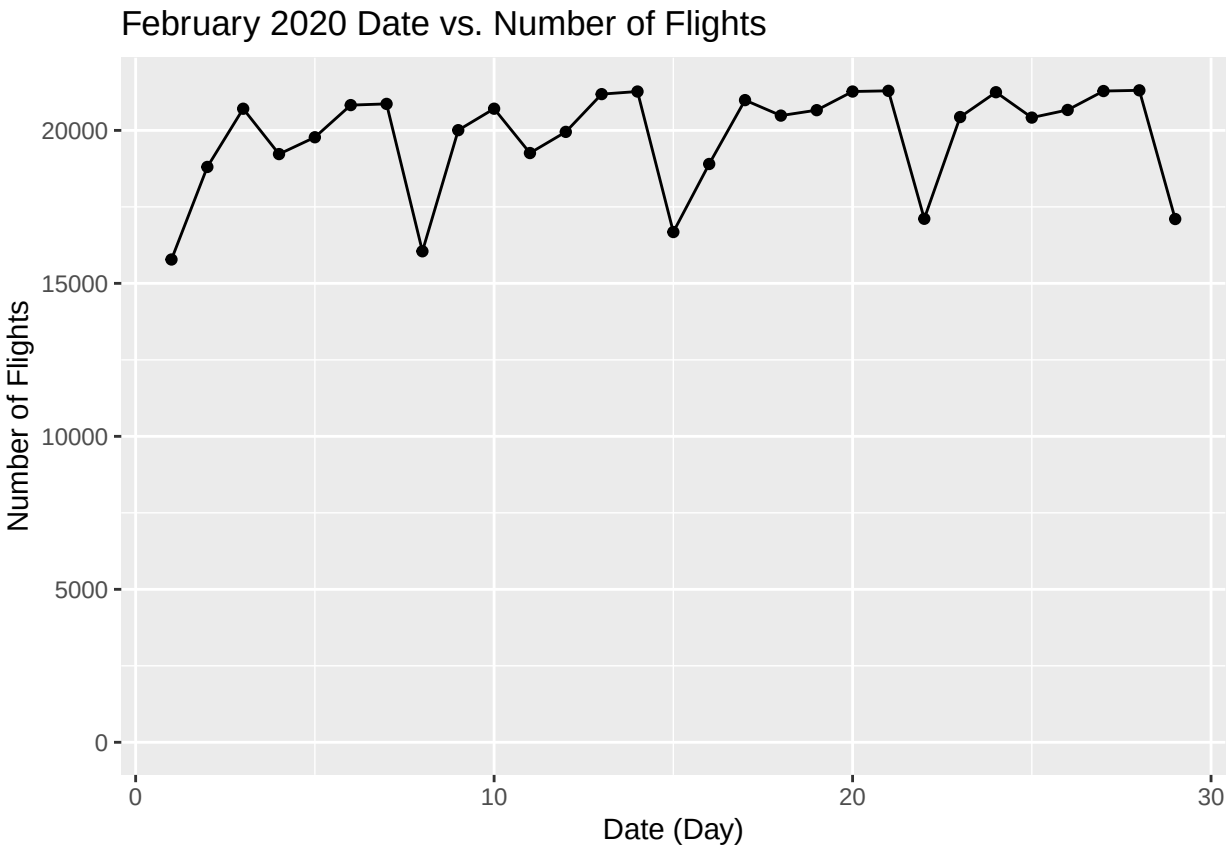

```
# plot line chart for Feb 2019
df120 %>%
  count(DAY_OF_MONTH) %>%
  ggplot(aes(x=DAY_OF_MONTH, y=n)) +
  geom_line(stat = "identity") +
  ggtitle("January 2020 Date vs. Number of Flights")+
  geom_point()+
  expand_limits(y=0)+
  labs(x="Date (Day)", y="Number of Flights")
```




```
# plot line chart for Jan 2020
df219 %>%
  count(DAY_OF_MONTH) %>%
  ggplot(aes(x=DAY_OF_MONTH, y=n)) +
  geom_line(stat = "identity") +
  ggtitle("February 2019 Date vs. Number of Flights")+
  geom_point()+
  expand_limits(y=0)+
  labs(x="Date (Day)", y="Number of Flights")
```




```
# plot line chart for Feb 2020
df220 %>%
  count(DAY_OF_MONTH) %>%
  ggplot(aes(x=DAY_OF_MONTH, y=n)) +
  geom_line(stat = "identity") +
  ggtitle("February 2020 Date vs. Number of Flights")+
  geom_point()+
  expand_limits(y=0)+
  labs(x="Date (Day)", y="Number of Flights")
```



Structuring

In this step, we transform the unusable raw data into organised form which can be further used for analysis.

Convert String to Dates (date change) Earlier we had a separate date and month column , and also the year was unspecified, so we made a new column known as Day.of.month which displays the entire date in "YYYY-MM-DD" format.

```
#In original data sets, dates are not in date format
#converting everything into date using function as.Date

#Converting for Jan 2019
df119 = df119 %>%
  mutate(df119,Day.of.month = paste(Year="2019", Month="01", DAY_OF_MONTH, sep = "-"))
df119$Day.of.month = as.Date(df119$Day.of.month)

#Converting for Jan 2019
df120 = df120 %>%
  mutate(df120,Day.of.month = paste(Year="2020", Month="01", DAY_OF_MONTH, sep = "-"))
df120$Day.of.month = as.Date(df120$Day.of.month)

#Converting for Jan 2019
df219 = df219 %>%
  mutate(df219,Day.of.month = paste(Year="2019", Month="02", DAY_OF_MONTH, sep = "-"))
```

```
df219$Day.of.month = as.Date(df219$Day.of.month)

#Converting for Jan 2019
df220 = df220 %>%
  mutate(df220,Day.of.month = paste(Year="2020", Month="02", DAY_OF_MONTH, sep = "-"))
df220$Day.of.month = as.Date(df220$Day.of.month)
```

Time Format Change In the original data set, the time was in the incorrect format (i.e. 0601 and not 06:01), hence, we altered the formatting to put it into a standard format for both arrival and departure time.

```
# Converting for Jan 2019
df119 = df119 %>%
  mutate(
    Arr.time = format(strptime(sprintf('%04d',as.integer(ARR_TIME)), format='%H%M'), '%H:%M'),
    Dep.time = format(strptime(sprintf('%04d',as.integer(DEP_TIME)), format='%H%M'), '%H:%M')
        )
#Converting for Feb 2019
df120 = df120 %>%
  mutate(
    Arr.time = format(strptime(sprintf('%04d',as.integer(ARR_TIME)), format='%H%M'), '%H:%M'),
    Dep.time = format(strptime(sprintf('%04d',as.integer(DEP_TIME)), format='%H%M'), '%H:%M')
        )
#Converting for Jan 2020
df219 = df219 %>%
  mutate(
    Arr.time = format(strptime(sprintf('%04d',as.integer(ARR_TIME)), format='%H%M'), '%H:%M'),
    Dep.time = format(strptime(sprintf('%04d',as.integer(DEP_TIME)), format='%H%M'), '%H:%M')
        )
#Converting for Feb 2020
df220 = df220 %>%
  mutate(
    Arr.time = format(strptime(sprintf('%04d',as.integer(ARR_TIME)), format='%H%M'), '%H:%M'),
    Dep.time = format(strptime(sprintf('%04d',as.integer(DEP_TIME)), format='%H%M'), '%H:%M')
        )
```

Creating Categories from continous Data (Weekday change) In this section, we converted the attribute Day.of.week from decimal number (7) to an ordered factor of character strings, such as "Sunday" using wday() with label=TRUE.

```
# weekday was shown as  integers where 1 as Monday and 7 as Sunday
#In this section, replace everything from a number to words

#replacing for the Jan 2019 data sets
df119 <- df119 %>%
  mutate(Day.of.week  = wday(Day.of.month,label=TRUE,abbr = FALSE)
  )

#replacing for the Jan 2020 data sets
df120 <- df120 %>%
  mutate(Day.of.week = wday(Day.of.month,label=TRUE,abbr = FALSE)
  )
```

```
#replacing for the Feb 2019 data sets
df219 <- df219 %>%
  mutate(Day.of.week = wday(Day.of.month,label=TRUE,abbr = FALSE)
  )

#replacing for the Feb 2020 data sets
df220 <- df220 %>%
  mutate(Day.of.week = wday(Day.of.month,label=TRUE,abbr = FALSE)
  )
```

Combine Text Column (Merging) In this section, we are merging the two attributes namely, OP_UNIQUE_CARRIER and OP_CARRIER_FL_NUM creating a new column called Flight Number.

```
#Combine two column to form a new information

#Combine for Jan 2019 data set
df119 = df119 %>%
  unite('Flight Number',OP_UNIQUE_CARRIER,OP_CARRIER_FL_NUM,remove=FALSE,sep="-")

#Combine for Jan 2020 data set
df120 = df120 %>%
  unite('Flight Number',OP_UNIQUE_CARRIER,OP_CARRIER_FL_NUM,remove=FALSE,sep="-")

#Combine for Feb 2019 data set
df219 = df219 %>%
  unite('Flight Number',OP_UNIQUE_CARRIER,OP_CARRIER_FL_NUM,remove=FALSE,sep="-")

#Combine for Feb 2020 data set
df220 = df220 %>%
  unite('Flight Number',OP_UNIQUE_CARRIER,OP_CARRIER_FL_NUM,remove=FALSE,sep="-")
```

Cleaning

This step includes deleting empty rows or cells, fixing NA values, examining outliers.

Remove NA (Fix missing data) In our data sets, we had the last column filled with NA's, so we removed them and checked again whether there is any NA's left or not, in the other half of the code, we replaced all the NA's by '0' so that we don't lose our data from the data set. At the end of this section, we have displayed the result of values with no NA's in the data set.

```
#Remove NAs

df119 = select(df119,-TAIL_NUM,-...22)
df120 = select(df120,-TAIL_NUM,-...22)
df219 = select(df219,-TAIL_NUM,-...22)
df220 = select(df220,-TAIL_NUM,-...22)

# Checking the NA values for the entire data set.

kable(
  df119 %>%
```

```
    summarise_all(funs(sum(is.na(.)))) %>%
    t()
,caption="2nd Time Check for NA January 2019")
```

Table 9: 2nd Time Check for NA January 2019

DAY_OF_MONTH	0
DAY_OF_WEEK	0
Flight Number	0
OP_UNIQUE_CARRIER	0
OP_CARRIER_AIRLINE_ID	0
OP_CARRIER	0
OP_CARRIER_FL_NUM	0
ORIGIN_AIRPORT_ID	0
ORIGIN_AIRPORT_SEQ_ID	0
ORIGIN	0
DEST_AIRPORT_ID	0
DEST_AIRPORT_SEQ_ID	0
DEST	0
DEP_TIME	16352
DEP_DEL15	16355
DEP_TIME_BLK	0
ARR_TIME	17061
ARR_DEL15	18022
CANCELLED	0
DIVERTED	0
DISTANCE	0
Day.of.month	0
Arr.time	17061
Dep.time	16352
Day.of.week	0

```
kable(
  df120 %>%
    summarise_all(funs(sum(is.na(.))))%>%
    t()
  ,caption="2nd Time Check for NA January 2020")
```

Table 10: 2nd Time Check for NA January 2020

DAY_OF_MONTH	0
DAY_OF_WEEK	0
Flight Number	0
OP_UNIQUE_CARRIER	0
OP_CARRIER_AIRLINE_ID	0
OP_CARRIER	0
OP_CARRIER_FL_NUM	0
ORIGIN_AIRPORT_ID	0
ORIGIN_AIRPORT_SEQ_ID	0
ORIGIN	0
DEST_AIRPORT_ID	0

Table 10: 2nd Time Check for NA January 2020

DEST_AIRPORT_SEQ_ID	0
DEST	0
DEP_TIME	6664
DEP_DEL15	6699
DEP_TIME_BLK	0
ARR_TIME	7075
ARR_DEL15	8078
CANCELLED	0
DIVERTED	0
DISTANCE	0
Day.of.month	0
Arr.time	7075
Dep.time	6664
Day.of.week	0

```
kable(
  df219 %>%
    summarise_all(funs(sum(is.na(.))))%>%
    t()
,caption="2nd Time Check for NA February 2019")
```

Table 11: 2nd Time Check for NA February 2019

DAY_OF_MONTH	0
DAY_OF_WEEK	0
Flight Number	0
OP_UNIQUE_CARRIER	0
OP_CARRIER_AIRLINE_ID	0
OP_CARRIER	0
OP_CARRIER_FL_NUM	0
ORIGIN_AIRPORT_ID	0
ORIGIN_AIRPORT_SEQ_ID	0
ORIGIN	0
DEST_AIRPORT_ID	0
DEST_AIRPORT_SEQ_ID	0
DEST	0
DEP_TIME	14823
DEP_DEL15	14827
DEP_TIME_BLK	0
ARR_TIME	15700
ARR_DEL15	16861
CANCELLED	0
DIVERTED	0
DISTANCE	0
Day.of.month	0
Arr.time	15700
Dep.time	14823
Day.of.week	0

```
kable(
  df220 %>%
    summarise_all(funs(sum(is.na(.))))%>%
    t()
,caption="2nd Time Check for NA February 2020")
```

Table 12: 2nd Time Check for NA February 2020

DAY_OF_MONTH	0
DAY_OF_WEEK	0
Flight Number	0
OP_UNIQUE_CARRIER	0
OP_CARRIER_AIRLINE_ID	0
OP_CARRIER	0
OP_CARRIER_FL_NUM	0
ORIGIN_AIRPORT_ID	0
ORIGIN_AIRPORT_SEQ_ID	0
ORIGIN	0
DEST_AIRPORT_ID	0
DEST_AIRPORT_SEQ_ID	0
DEST	0
DEP_TIME	4938
DEP_DEL15	4951
DEP_TIME_BLK	0
ARR_TIME	5287
ARR_DEL15	6192
CANCELLED	0
DIVERTED	0
DISTANCE	0
Day.of.month	0
Arr.time	5287
Dep.time	4938
Day.of.week	0

```
# Replacing NA values to 0

df119 <- mutate_at(df119, c("DEP_TIME", "ARR_TIME", "Arr.time", "Dep.time","ARR_DEL15","DEP_DEL15"), ~r

df120 <- mutate_at(df120, c("DEP_TIME", "ARR_TIME", "Arr.time", "Dep.time","ARR_DEL15","DEP_DEL15"), ~r

df219 <- mutate_at(df219, c("DEP_TIME", "ARR_TIME", "Arr.time", "Dep.time","ARR_DEL15","DEP_DEL15"), ~r

df220 <- mutate_at(df220, c("DEP_TIME", "ARR_TIME", "Arr.time", "Dep.time","ARR_DEL15","DEP_DEL15"), ~r

#Validating
kable(
  df119 %>%
    summarise_all(funs(sum(is.na(.)))) %>%
    t()
,caption="3rd time Check for NA January 2019")
```

Table 13: 3rd time Check for NA January 2019

DAY_OF_MONTH	0
DAY_OF_WEEK	0
Flight Number	0
OP_UNIQUE_CARRIER	0
OP_CARRIER_AIRLINE_ID	0
OP_CARRIER	0
OP_CARRIER_FL_NUM	0
ORIGIN_AIRPORT_ID	0
ORIGIN_AIRPORT_SEQ_ID	0
ORIGIN	0
DEST_AIRPORT_ID	0
DEST_AIRPORT_SEQ_ID	0
DEST	0
DEP_TIME	0
DEP_DEL15	0
DEP_TIME_BLK	0
ARR_TIME	0
ARR_DEL15	0
CANCELLED	0
DIVERTED	0
DISTANCE	0
Day.of.month	0
Arr.time	0
Dep.time	0
Day.of.week	0

```
kable(
  df120 %>%
    summarise_all(funs(sum(is.na(.))))%>%
    t()
  ,caption="3rd time Check for NA January 2020")
```

Table 14: 3rd time Check for NA January 2020

DAY_OF_MONTH	0
DAY_OF_WEEK	0
Flight Number	0
OP_UNIQUE_CARRIER	0
OP_CARRIER_AIRLINE_ID	0
OP_CARRIER	0
OP_CARRIER_FL_NUM	0
ORIGIN_AIRPORT_ID	0
ORIGIN_AIRPORT_SEQ_ID	0
ORIGIN	0
DEST_AIRPORT_ID	0
DEST_AIRPORT_SEQ_ID	0
DEST	0
DEP_TIME	0
DEP_DEL15	0
DEP_TIME_BLK	0

Table 14: 3rd time Check for NA January 2020

ARR_TIME	0
ARR_DEL15	0
CANCELLED	0
DIVERTED	0
DISTANCE	0
Day.of.month	0
Arr.time	0
Dep.time	0
Day.of.week	0

```
kable(
  df219 %>%
    summarise_all(funs(sum(is.na(.))))%>%
    t()
,caption="3rd time Check for NA February 2019")
```

Table 15: 3rd time Check for NA February 2019

DAY_OF_MONTH	0
DAY_OF_WEEK	0
Flight Number	0
OP_UNIQUE_CARRIER	0
OP_CARRIER_AIRLINE_ID	0
OP_CARRIER	0
OP_CARRIER_FL_NUM	0
ORIGIN_AIRPORT_ID	0
ORIGIN_AIRPORT_SEQ_ID	0
ORIGIN	0
DEST_AIRPORT_ID	0
DEST_AIRPORT_SEQ_ID	0
DEST	0
DEP_TIME	0
DEP_DEL15	0
DEP_TIME_BLK	0
ARR_TIME	0
ARR_DEL15	0
CANCELLED	0
DIVERTED	0
DISTANCE	0
Day.of.month	0
Arr.time	0
Dep.time	0
Day.of.week	0

```
kable(
  df220 %>%
    summarise_all(funs(sum(is.na(.))))%>%
    t()
,caption="3rd time Check for NA February 2020")
```

Table 16: 3rd time Check for NA February 2020

DAY_OF_MONTH	0
DAY_OF_WEEK	0
Flight Number	0
OP_UNIQUE_CARRIER	0
OP_CARRIER_AIRLINE_ID	0
OP_CARRIER	0
OP_CARRIER_FL_NUM	0
ORIGIN_AIRPORT_ID	0
ORIGIN_AIRPORT_SEQ_ID	0
ORIGIN	0
DEST_AIRPORT_ID	0
DEST_AIRPORT_SEQ_ID	0
DEST	0
DEP_TIME	0
DEP_DEL15	0
DEP_TIME_BLK	0
ARR_TIME	0
ARR_DEL15	0
CANCELLED	0
DIVERTED	0
DISTANCE	0
Day.of.month	0
Arr.time	0
Dep.time	0
Day.of.week	0

Check Outliers In this section we are checking for outliers, so the data set was inconsistent, we first convert the values in minutes, then we filter out the values which are causing errors in making the decisions.

We manage to check outliers by looking at the depart time and arriving time. In the data set there are some outliers such as the duration of the flight is less than 10 minutes. It might either causing by a human error or a emergency call back. Under both situation, it does not help us to do the data driven-decision. Remove outliers for all the datasets rather than fix them. We do not have a fitable prediction model for the task.

```
# Adding new column to check for time
# subtracting arriving time and departing time to find the flight duration
df119$time_check = round(abs(as.double(as.POSIXct(df119$Arr.time, format = "%H:%M")- as.POSIXct(df119$D

#Filter out any flight shorter than 60min*0.2
df119 = df119 %>%
  filter(time_check >0.2)

#Display output flight time in hour
kable(
  df119 %>%
    count(time_check) %>%
    head(5)
  ,caption = "Flight time check January 2019", col.names = c("Time Check (hours)" , "Number of Flights"
  )
```

Table 17: Flight time check January 2019

Time Check (hours)	Number of Flights
0.22	350
0.23	359
0.25	392
0.27	348
0.28	365

```
# subtracting arriving time and departing time to find the flight duration
df120$time_check = round(abs(as.double(as.POSIXct(df120$Arr.time, format = "%H:%M")- as.POSIXct(df120$D

#Filter out any flight shorter than 60min*0.2
df120 = df120 %>%
  filter(time_check >0.2)

#Display output flight time in hour
kable(
  df120 %>%
    count(time_check) %>%
    head(5)
  ,caption = "Flight time check January 2020", col.names = c("Time Check (hours)" , "Number of Flights"
  )
```

Table 18: Flight time check January 2020

Time Check (hours)	Number of Flights
0.22	454
0.23	404
0.25	394
0.27	385
0.28	358

```
# subtracting arriving time and departing time to find the flight duration
df219$time_check = round(abs(as.double(as.POSIXct(df219$Arr.time, format = "%H:%M")- as.POSIXct(df219$D

#Filter out any flight shorter than 60min*0.2
df219 = df219 %>%
  filter(time_check >0.2)

#Display output flight time in hour
kable(
  df219 %>%
    count(time_check) %>%
    head(5)
  ,caption = "Flight time check February 2019", col.names = c("Time Check (hours)" , "Number of Flights
  )
```

Table 19: Flight time check February 2019

Time Check (hours)	Number of Flights
0.22	318
0.23	308
0.25	307
0.27	335
0.28	333

```
# subtracting arriving time and departing time to find the flight duration
df220$time_check = round(abs(as.double(as.POSIXct(df220$Arr.time, format = "%H:%M")- as.POSIXct(df220$D

#Filter out any flight shorter than 60min*0.2
df220 = df220 %>%
  filter(time_check >0.2)

#Display output flight time in hour
kable(
  df220 %>%
    count(time_check) %>%
    head(5)
  ,caption = "Flight time check February 2020", col.names = c("Time Check (hours)" , "Number of Flights
  )
```

Table 20: Flight time check February 2020

Time Check (hours)	Number of Flights
0.22	335
0.23	373
0.25	366
0.27	372
0.28	378

Enriching

This involves incorporating data from other datasets and removing the irrelevant ones for easy data manipulation.

Add attributes In this section, we are working on the datasets, where we are joining all the data set of four months data to the dfc dataset, using a common column name.

```
# Adding extra column by using an additional table
# Giving companies name by using their IATA code

#select useful column for the project
#In this case, get rid og the second column in IATA code dataset
dfc = select(dfc,-2)
colnames(dfc)[1] = "OP_UNIQUE_CARRIER"
```

```
#convert everything into upper case to aviod unnecessary chaos
#for example: Southwestern Airline vs. SouthWestern Airline
dfc = dfc %>%
  mutate(`Air Carrier Name`= toupper(`Air Carrier Name`))

#left join and form a new column
df119 = left_join(df119,dfc)
df120 = left_join(df120,dfc)
df219 = left_join(df219,dfc)
df220 = left_join(df220,dfc)
```

Add Category We would like to add a category to each company and group them by their their business partner. Several company share common air routes. In the transportation field, they are so called Branded Code share Partners. Rather than splitting by using if else statement (where we perhapes need to run then thousands of loops with 20 elseif statement), we put everything into another data set and perform a left join. (Due to the limited of our device computational power, we consider it as a good way to get around the memoery restrictions)

```
#Introducing the new dataset and change col name into a share-in-common name
colnames(dfbcp)[1] = "Air Carrier Name"
dfbcp = dfbcp %>%
  mutate(`Air Carrier Name`= toupper(`Air Carrier Name`))

#left join two data sets
df119 = left_join(df119,dfbcp,by= "Air Carrier Name")
df120 = left_join(df120,dfbcp,by= "Air Carrier Name")
df219 = left_join(df219,dfbcp,by= "Air Carrier Name")
df220 = left_join(df220,dfbcp,by= "Air Carrier Name")
```

Validating

In this step, we verify whether the data is of high quality, dependable, and logical.

Checking NA value After joining the datasets there is a possibility that our data set might again have redundancies, so we again check for NA values using the summarize function.

```
#Re-run NA checks

kable(
  df119 %>%
    summarise_all(funs(sum(is.na(.))))
,caption = "Double check for NA value January 2019")
```

Table 21: Double check for NA value January 2019

DAY	DAY_OF_MONTH	DAY_OF_WEEK	OP_UNIQUE_CARRIER	OP_CARRIER_AIRLINE_ID	TAIL_NUM	OP_CARRIER_FLIGHT_NUM	ORIGIN_AIRPORT_ID	ORIGIN_AIRPORT_SEQ_ID	ORIGIN_CITY_MARKET_ID	ORIGIN	DEST_AIRPORT_ID	DEST_AIRPORT_SEQ_ID	DEST_CITY_MARKET_ID	DEST	CRS_DEP_TIME	DEP_TIME	DEP_DELAY	DEP_DELAY_NEW	DEP_DEL15	DEP_DELAY_GROUP	CRS_ARR_TIME	ARR_TIME	ARR_DELAY	ARR_DELAY_NEW	ARR_DEL15	ARR_DELAY_GROUP	CANCELLED	DIVERTED	CRS_ELAPSED_TIME	ACTUAL_ELAPSED_TIME	AIR_TIME	FLIGHTS	DISTANCE	DISTANCE_GROUP	CARRIER_DELAY	WEATHER_DELAY	NAS_DELAY	SECURITY_DELAY	LATE_AIRCRAFT_DELAY	Unnamed: 36
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

```
kable(
  df120 %>%
    summarise_all(funs(sum(is.na(.))))
,caption = "Double check for NA value January 2020")
```

Table 22: Double check for NA value January 2020

DAY	DAY_OF_MONTH	DAY_OF_WEEK	OP_UNIQUE_CARRIER	OP_CARRIER_AIRLINE_ID	TAIL_NUM	OP_CARRIER_FLIGHT_NUM	ORIGIN_AIRPORT_ID	ORIGIN_AIRPORT_SEQ_ID	ORIGIN_CITY_MARKET_ID	ORIGIN	DEST_AIRPORT_ID	DEST_AIRPORT_SEQ_ID	DEST_CITY_MARKET_ID	DEST	CRS_DEP_TIME	DEP_TIME	DEP_DELAY	DEP_DELAY_NEW	DEP_DEL15	DEP_DELAY_GROUP	CRS_ARR_TIME	ARR_TIME	ARR_DELAY	ARR_DELAY_NEW	ARR_DEL15	ARR_DELAY_GROUP	CANCELLED	DIVERTED	CRS_ELAPSED_TIME	ACTUAL_ELAPSED_TIME	AIR_TIME	FLIGHTS	DISTANCE	DISTANCE_GROUP	CARRIER_DELAY	WEATHER_DELAY	NAS_DELAY	SECURITY_DELAY	LATE_AIRCRAFT_DELAY	Unnamed: 36
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

```
kable(
  df219 %>%
    summarise_all(funs(sum(is.na(.))))
,caption = "Double check for NA value February 2019")
```

Table 23: Double check for NA value February 2019

DAY	DAY_OF_MONTH	DAY_OF_WEEK	OP_UNIQUE_CARRIER	OP_CARRIER_AIRLINE_ID	TAIL_NUM	OP_CARRIER_FLIGHT_NUM	ORIGIN_AIRPORT_ID	ORIGIN_AIRPORT_SEQ_ID	ORIGIN_CITY_MARKET_ID	ORIGIN	DEST_AIRPORT_ID	DEST_AIRPORT_SEQ_ID	DEST_CITY_MARKET_ID	DEST	CRS_DEP_TIME	DEP_TIME	DEP_DELAY	DEP_DELAY_NEW	DEP_DEL15	DEP_DELAY_GROUP	CRS_ARR_TIME	ARR_TIME	ARR_DELAY	ARR_DELAY_NEW	ARR_DEL15	ARR_DELAY_GROUP	CANCELLED	DIVERTED	CRS_ELAPSED_TIME	ACTUAL_ELAPSED_TIME	AIR_TIME	FLIGHTS	DISTANCE	DISTANCE_GROUP	CARRIER_DELAY	WEATHER_DELAY	NAS_DELAY	SECURITY_DELAY	LATE_AIRCRAFT_DELAY	Unnamed: 36
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

```
kable(
  df220 %>%
    summarise_all(funs(sum(is.na(.))))
,caption = "Double check for NA value February 2020")
```

Table 24: Double check for NA value February 2020

DAY	DAY_OF_MONTH	DAY_OF_WEEK	OP_UNIQUE_CARRIER	OP_CARRIER_AIRLINE_ID	TAIL_NUM	OP_CARRIER_FLIGHT_NUM	ORIGIN_AIRPORT_ID	ORIGIN_AIRPORT_SEQ_ID	ORIGIN_CITY_MARKET_ID	ORIGIN	DEST_AIRPORT_ID	DEST_AIRPORT_SEQ_ID	DEST_CITY_MARKET_ID	DEST	CRS_DEP_TIME	DEP_TIME	DEP_DELAY	DEP_DELAY_NEW	DEP_DEL15	DEP_DELAY_GROUP	CRS_ARR_TIME	ARR_TIME	ARR_DELAY	ARR_DELAY_NEW	ARR_DEL15	ARR_DELAY_GROUP	CANCELLED	DIVERTED	CRS_ELAPSED_TIME	ACTUAL_ELAPSED_TIME	AIR_TIME	FLIGHTS	DISTANCE	DISTANCE_GROUP	CARRIER_DELAY	WEATHER_DELAY	NAS_DELAY	SECURITY_DELAY	LATE_AIRCRAFT_DELAY	Unnamed: 36
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Every data sets do not contain any NA values by now.

Checking date type Double checking data type for dates. Select everything is not in date format.

```
#Select everything is not in date format.
kable(
  df119 %>%
    filter(is.Date(Day.of.month)==FALSE) %>%
             select(Day.of.month),
           caption="is.Date() Check for January-2019"
  )
```

Table 25: is.Date() Check for January-2019

Day.of.month

```
#Select everything is not in date format.
kable(
  df120 %>%
    filter(is.Date(Day.of.month)==FALSE) %>%
             select(Day.of.month),
           caption="is.Date() Check for January-2020"
  )
```

Table 26: is.Date() Check for January-2020

Day.of.month

```
#Select everything is not in date format.
kable(
  df219 %>%
    filter(is.Date(Day.of.month)==FALSE) %>%
             select(Day.of.month),
           caption="is.Date() Check for February-2019"
  )
```

Table 27: is.Date() Check for February-2019

Day.of.month

```
#Select everything is not in date format.
kable(
  df220 %>%
    filter(is.Date(Day.of.month)==FALSE) %>%
             select(Day.of.month),
           caption="is.Date() Check for February-2020"
  )
```

Table 28: is.Date() Check for February-2020

Day.of.month

The empty values means that the dates are in the correct format. Nothing shows up. That means everything is now in date format.

Check duplication There's a possibility that a single carrier with the same flight number/id is flying from similar or different locations, so in this section, we are trying to check for duplication and listing them down, for the same number of flights that are on the same timing, date, and having the same distance. Check duplication by group by every possible categories.

```
#use group by + count to find if everything is duplicate
#Check n for count

#check for Jan 2019

kable(
  df119 %>%
    group_by(`Flight Number`, Day.of.month, Dep.time, Arr.time, DISTANCE) %>%
    count(sort = TRUE) %>%
    head(10)
  ,caption="Duplication Check for January 2019")
```

Table 29: Duplication Check for January 2019

Flight Number	Day.of.month	Dep.time	Arr.time	DISTANCE	n
EV-2860	2019-01-06	09:50	12:29	853	2
EV-2861	2019-01-06	05:39	08:23	437	2
EV-2862	2019-01-02	19:13	20:23	247	2
EV-2862	2019-01-03	20:01	21:04	247	2
EV-2862	2019-01-04	18:25	19:33	247	2
EV-2862	2019-01-05	18:09	19:32	247	2
EV-2862	2019-01-06	07:34	09:04	383	2
EV-2863	2019-01-01	11:06	13:57	853	2
EV-2863	2019-01-02	12:45	15:24	853	2
EV-2863	2019-01-03	12:43	15:17	853	2

```
#check for Jan 2020
kable(
  df120 %>%
    group_by(`Flight Number`, Day.of.month, Dep.time, Arr.time, DISTANCE) %>%
    count(sort = TRUE)%>%
    head(10)
  ,caption="Duplication Check for January 2020")
```

Table 30: Duplication Check for January 2020

Flight Number	Day.of.month	Dep.time	Arr.time	DISTANCE	n
EV-3965	2020-01-02	12:01	15:17	744	2
EV-3965	2020-01-03	12:05	15:15	744	2
EV-3965	2020-01-04	12:07	15:11	744	2
EV-3965	2020-01-05	05:58	07:07	174	2
EV-3965	2020-01-06	06:17	08:34	1091	2
EV-3965	2020-01-07	06:15	08:30	1091	2
EV-3965	2020-01-08	06:18	08:15	1091	2
EV-3965	2020-01-09	06:15	08:38	1091	2
EV-3965	2020-01-10	06:11	08:39	1091	2
EV-3965	2020-01-11	06:16	08:44	1091	2

```
#check for Feb 2019
kable(
  df219 %>%
    group_by(`Flight Number`, Day.of.month, Dep.time, Arr.time, DISTANCE) %>%
    count(sort = TRUE)%>%
    head(10)
  ,caption="Duplication Check for February 2019")
```

Table 31: Duplication Check for February 2019

Flight Number	Day.of.month	Dep.time	Arr.time	DISTANCE	n
EV-3965	2019-02-01	10:55	12:47	409	2
EV-3965	2019-02-02	10:55	12:26	409	2
EV-3965	2019-02-03	11:03	12:31	409	2
EV-3965	2019-02-04	10:50	12:23	409	2
EV-3965	2019-02-05	09:53	12:17	463	2
EV-3965	2019-02-06	10:49	12:37	409	2
EV-3965	2019-02-07	10:56	12:54	409	2
EV-3965	2019-02-08	10:53	12:38	409	2
EV-3965	2019-02-09	15:37	17:16	409	2
EV-3965	2019-02-10	13:57	15:53	409	2

```
#check for Feb 2020
kable(
  df220 %>%
    group_by(`Flight Number`, Day.of.month, Dep.time, Arr.time, DISTANCE) %>%
    count(sort = TRUE)%>%
    head(10)
  ,caption="Duplication Check for February 2020")
```

Table 32: Duplication Check for February 2020

Flight Number	Day.of.month	Dep.time	Arr.time	DISTANCE	n
EV-3965	2020-02-01	06:27	08:37	1091	2
EV-3965	2020-02-02	06:19	08:23	1091	2

Flight Number	Day.of.month	Dep.time	Arr.time	DISTANCE	n
EV-3965	2020-02-03	06:12	08:14	1091	2
EV-3965	2020-02-04	06:10	08:26	1091	2
EV-3965	2020-02-05	06:16	09:03	1091	2
EV-3965	2020-02-06	06:17	10:11	1091	2
EV-3965	2020-02-07	06:56	09:16	1091	2
EV-3965	2020-02-08	06:18	08:46	1091	2
EV-3965	2020-02-09	06:10	08:57	1091	2
EV-3965	2020-02-10	06:17	09:14	1091	2

Publishing

Bind Creating one large data frame by combining all the four data frames using the rbind function for working on business problems and checking for the NA values again, if any cleaning needs to be done.

```
df = rbind(df119, df120,df219,df220)

df %>%
  summarise_all(funs(sum(is.na(.))))
```

```
## # A tibble: 1 x 28
##   DAY_OF_MONTH DAY_OF_WEEK `Flight Number` OP_UNIQUE_CARRIER OP_CARRIER_AIRLINE~
##          <int>       <int>           <int>              <int>                <int>
## 1            0           0               0                  0                    0
## # ... with 23 more variables: OP_CARRIER <int>, OP_CARRIER_FL_NUM <int>,
## #   ORIGIN_AIRPORT_ID <int>, ORIGIN_AIRPORT_SEQ_ID <int>, ORIGIN <int>,
## #   DEST_AIRPORT_ID <int>, DEST_AIRPORT_SEQ_ID <int>, DEST <int>,
## #   DEP_TIME <int>, DEP_DEL15 <int>, DEP_TIME_BLK <int>, ARR_TIME <int>,
## #   ARR_DEL15 <int>, CANCELLED <int>, DIVERTED <int>, DISTANCE <int>,
## #   Day.of.month <int>, Arr.time <int>, Dep.time <int>, Day.of.week <int>,
## #   time_check <int>, Air Carrier Name <int>, Codeshare Partner <int>
```

Publishing Github The data by know is fully unified. The data set is now cleaned (NA removed and no more ouliers), multiple columns added to the data base and enriched the entire data set.

The progress should be available to others and ready for perform futhur analysis. Since the group do not have access to any Northeastern University code sharing platform, the group decide to push everything to Github as an alternative for publishing the code. Github link: https://github.com/FengboMa/IE5374_sec3_Group16_project_1.git

Decision-making with data

1. Display the total number of unique Air carrier names.

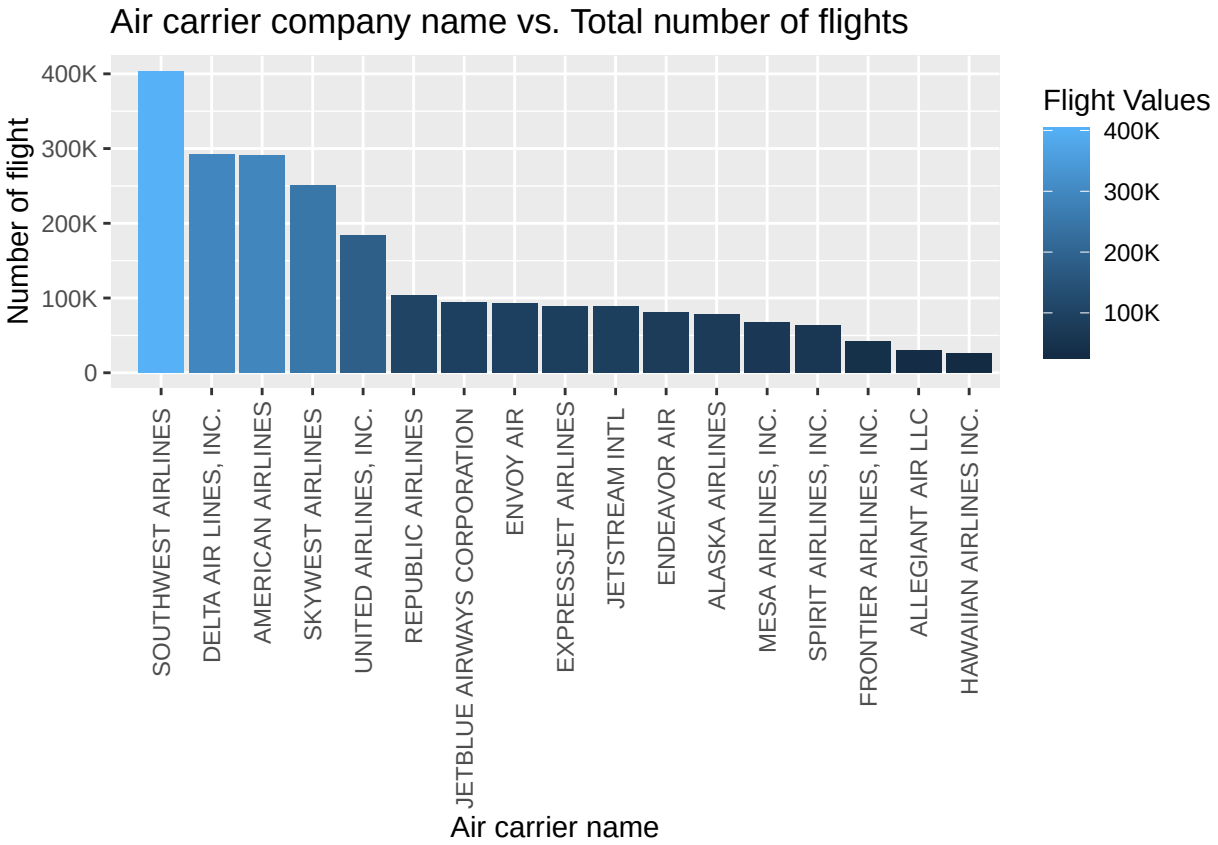
```
#Display air carrier name
unique_airline = unique(df$`Air Carrier Name`)
unique_airline
```

```
##  [1] "ENDEAVOR AIR "                "AMERICAN AIRLINES"
##  [3] "ENVOY AIR "                   "ALLEGIANT AIR LLC "
##  [5] "JETSTREAM INTL "              "JETBLUE AIRWAYS CORPORATION "
##  [7] "MESA AIRLINES, INC. "         "EXPRESSJET AIRLINES"
##  [9] "FRONTIER AIRLINES, INC. "     "REPUBLIC AIRLINES "
## [11] "HAWAIIAN AIRLINES INC. "      "SPIRIT AIRLINES, INC. "
## [13] "SKYWEST AIRLINES "            "SOUTHWEST AIRLINES "
## [15] "ALASKA AIRLINES"              "UNITED AIRLINES, INC. "
## [17] "DELTA AIR LINES, INC. "
```

```
#count total number flight for each carrier
total_number = length(unique_airline)
paste("There are total", total_number,"unique Air carriers.")
```

```
## [1] "There are total 17 unique Air carriers."
```

```
#plot the bar chart to demonstrate the number of flight for each air carrier
df %>%
  count(`Air Carrier Name`) %>%
  ggplot(aes(x=reorder(`Air Carrier Name`,-n), y=n,fill= n)) +
  scale_fill_continuous(name = 'Flight Values',labels = scales::label_number_si())+
  geom_bar(stat = "identity") +
  ggtitle("Air carrier company name vs. Total number of flights")+
  labs(x= "Air carrier name", y= "Number of flight")+
  expand_limits(x = c(0, NA), y = c(0,NA)) +
  scale_y_continuous(labels = scales::label_number_si())+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```



It can be depicted that Southwest Airlines has the highest count of flights which is about 400K, whereas Hawaiian Airlines has the least number of flights about 20K. Furthermore, Delta Airlines and American Airlines have the same number of flights around 290K.

2. Which day of the weekday had the most flights?

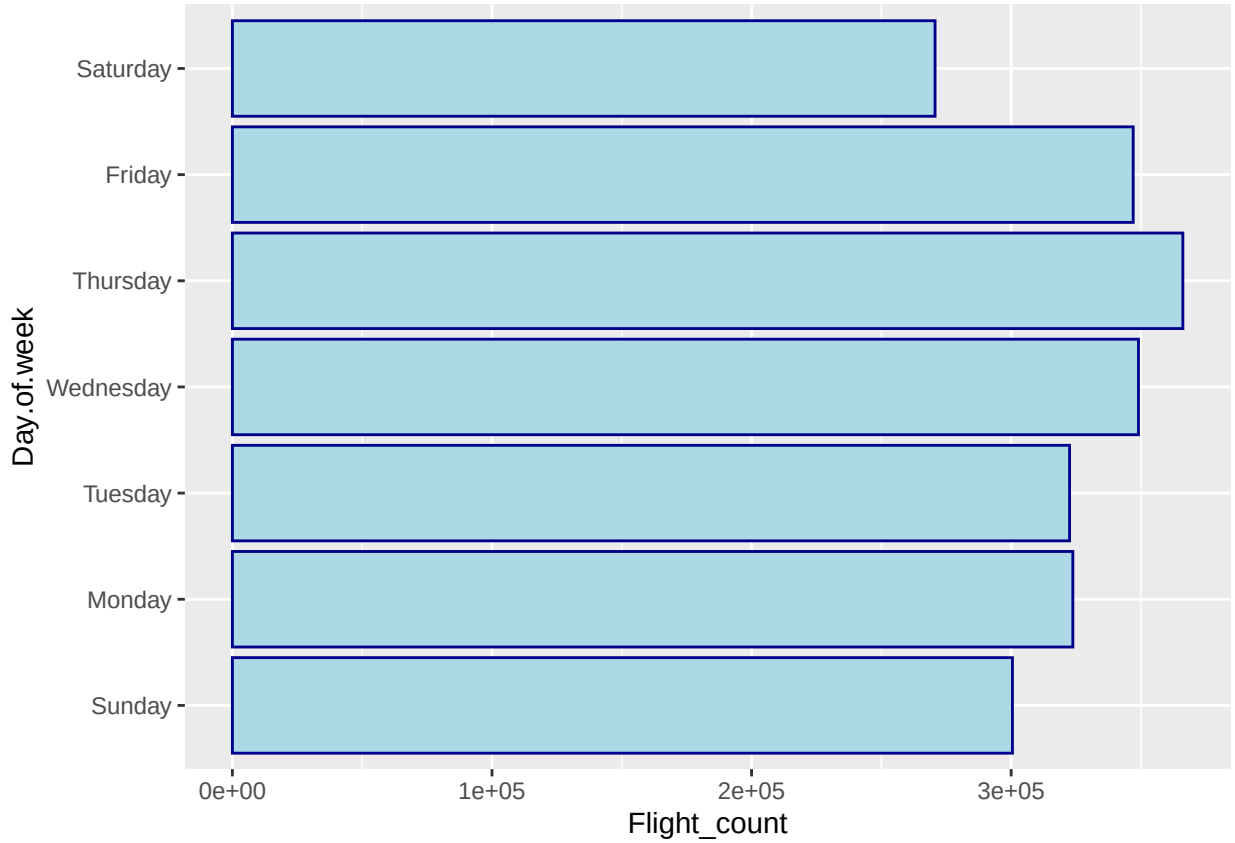
```
# Counting the number of flights for each weekdays
kable(
  Task2<-
    df%>%
    select(Day.of.week)%>%
    group_by(Day.of.week)%>%
    arrange(Day.of.week, order=TRUE)%>%
    summarise(Flight_count=n()),
    caption = 'Total Flights on Each Day'
)
```

Table 33: Total Flights on Each Day

Day.of.week	Flight_count
Sunday	300485
Monday	323760
Tuesday	322510
Wednesday	349044

Day.of.week	Flight_count
Thursday	366146
Friday	347010
Saturday	270685

```
#Create a bar chart for each weekday
Task2 %>%
  ggplot(aes(Flight_count, Day.of.week,))+
  geom_bar(stat= "identity", color="darkblue", fill="lightblue" )
```




```
#Slice out the weekday with the most flights
kable(
  df%>%
    select(Day.of.week)%>%
    group_by(Day.of.week)%>%
    summarise(Flight_count=n())%>%
    arrange(desc(Flight_count))%>%
    slice(1)
  ,caption = "Weekday with the Most Flights")
```

Table 34: Weekday with the Most Flights

Day.of.week	Flight_count
Thursday	366146

It can be depicted that Thursday had the highest number of flights which was approximately 366K and is represented in the table. From the bar graph, we can clearly figure out that Friday and Wednesday are the second-most and third most respectively. On the contrary, Saturday had the least number of flights about 275K.

3. Which flight had the largest distance? And was the distance the reason for the delay? And Which flight had the smallest distance and was there a delay?

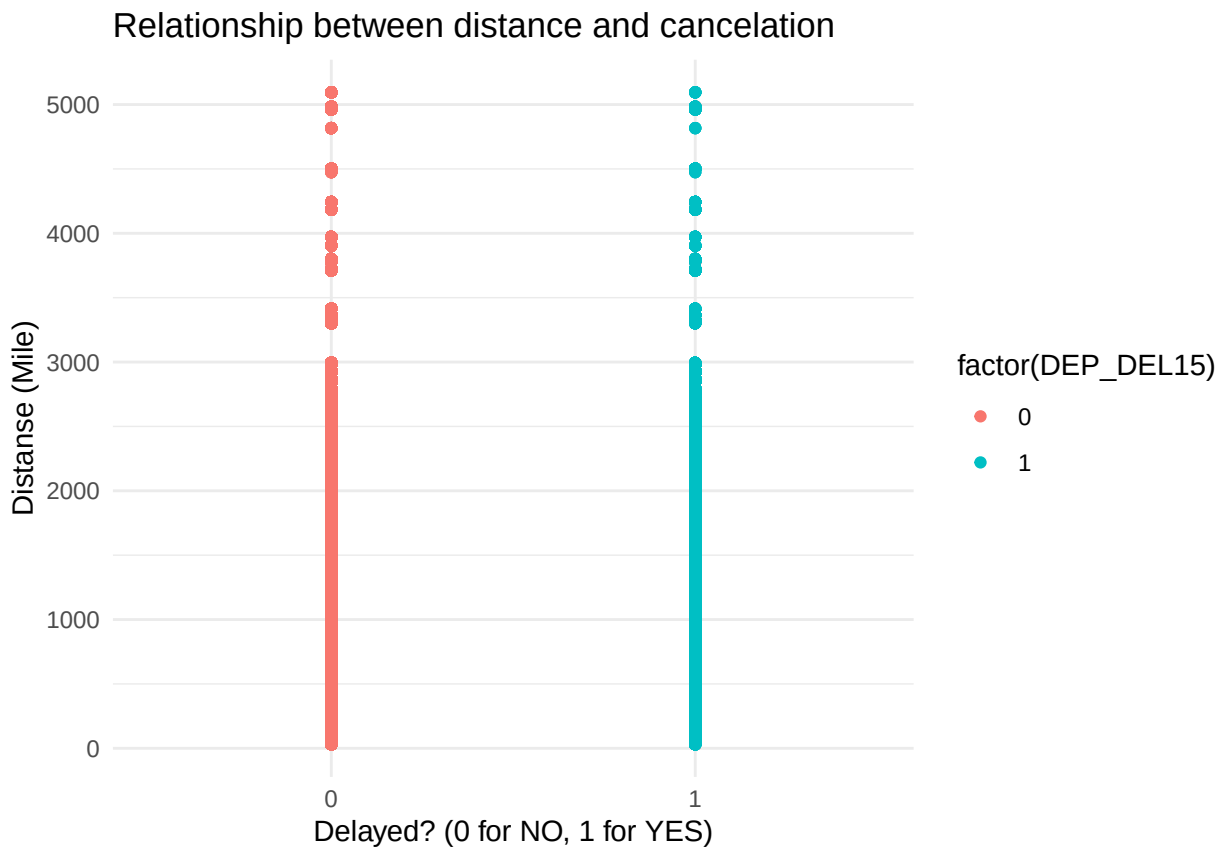
By finding the largest distance, filter out the column with distance is critical. Then use the max function to point out the max distance. To examine any relationships between distance and change of delay, we plot a point chart to find out if there is any correlation.

```
#Display the largest distance by using max function
kable(
  max(df$DISTANCE),
  caption = "The Largest Distance", col.names = c("Distance (mile)") )
```

Table 35: The Largest Distance

Distance (mile)
5095

```
#Dot plot, compare delay with distance, Does the delay has anything to do with distance?
df %>%
  select(DISTANCE, DEP_DEL15) %>%
  ggplot( aes(x=factor(DEP_DEL15), y=DISTANCE, color=factor(DEP_DEL15), fill=factor(DEP_DEL15))) +
  geom_point() +
  ggtitle("Relationship between distance and cancelation")+
  labs(x= "Delayed? (0 for NO, 1 for YES)", y= "Distanse (Mile)")+
  theme_minimal()
```




```
# using max min and mean to output relationship in numerical perspective for delayed flight
kable(
  df %>%
  filter(DEP_DEL15 == 1) %>%
  summarise(max_distance=max(DISTANCE), min_distance=min(DISTANCE), avg_distance=mean(DISTANCE)),
  caption="Summary Statistics for Delayed flight"
)
```

Table 36: Summary Statistics for Delayed flight

max_distance	min_distance	avg_distance
5095	31	809.1967

```
# using max min and mean to output relationship in numerical perspective for Non-delayed flight
kable(
  df %>%
  filter(DEP_DEL15 == 0) %>%
  summarise(max_distance=max(DISTANCE), min_distance=min(DISTANCE), avg_distance=mean(DISTANCE)),
  caption="Summary Statistics for Non Delayed flight"
)
```

Table 37: Summary Statistics for Non Delayed flight

max_distance	min_distance	avg_distance
5095	31	796.5901

```
# Which flight had the smallest distance and was there a delay?
kable(
  df %>%
  filter(DISTANCE == min(df$DISTANCE)) %>%
  count(DEP_DEL15)
  ,caption="Smallest Distance and Delay", col.names = c("If Delayed?", "Number of the observation")
)
```

Table 38: Smallest Distance and Delay

If Delayed?	Number of the observation
0	183
1	39

By looking at the plot, the correlation between distance and delay is not strong by looking at the plot. To further analysis the problem, information such as max, min and mean could be calculated and outputed as a numerical. As there are no distinct differences between the two observations, it is confirmed that the chance of delay is not related to distance.

4. How many numbers of flights were canceled in week 1? which weekday had the most flight canceled for both the year?

```
# Select information from the 1st week where should be from Jan 1, 2019 to Jan 7, 2019. and count the n
c <-
  df %>%
  select(Day.of.month,Day.of.week, CANCELLED) %>%
  filter(Day.of.month >= "2019-01-01" & Day.of.month<= "2019-01-07") %>%
  group_by(Day.of.week) %>%
  summarise(CancelledFlights = n())

# Present as a table
kable (c,
       caption="Numbers of flights cancelled in week 1"
)
```

Table 39: Numbers of flights cancelled in week 1

Day.of.week	CancelledFlights
Sunday	19403
Monday	20122
Tuesday	18142
Wednesday	20494

Day.of.week	CancelledFlights
Thursday	19610
Friday	19755
Saturday	16977

```
# How many numbers of flights were canceled in week 1?
value = sum(c$CancelledFlights)
paste("The total number of Cancelled Flights in Week 1:", value)
```

```
## [1] "The total number of Cancelled Flights in Week 1: 134503"
```

```
# Select and group by weeks and count for which weekday have the most flight canceled
kable(
  df%>%
  select(Day.of.week,CANCELLED)%>%
  group_by(Day.of.week)%>%
  arrange(Day.of.week, order=TRUE)%>%
  summarise(Values=n()),
  caption = 'Total Flights for both years grouped by week',
)
```

Table 40: Total Flights for both years grouped by week

Day.of.week	Values
Sunday	300485
Monday	323760
Tuesday	322510
Wednesday	349044
Thursday	366146
Friday	347010
Saturday	270685

```
# Slice out Weekday that had the most cancelled flights for both years
kable(
  df%>%
  select(Day.of.week)%>%
  group_by(Day.of.week)%>%
  summarise(Values=n())%>%
  arrange(desc(Values))%>%
  slice(1),
  caption = 'Weekday that had the most cancelled flights for both years'
)
```

Table 41: Weekday that had the most cancelled flights for both years

Day.of.week	Values
Thursday	366146

First, we displayed the total number of flights canceled through the four months, in which we found that the total number of flights canceled in week 1 was 134503 and Thursday had the most number of flights canceled which was 366146 flights.

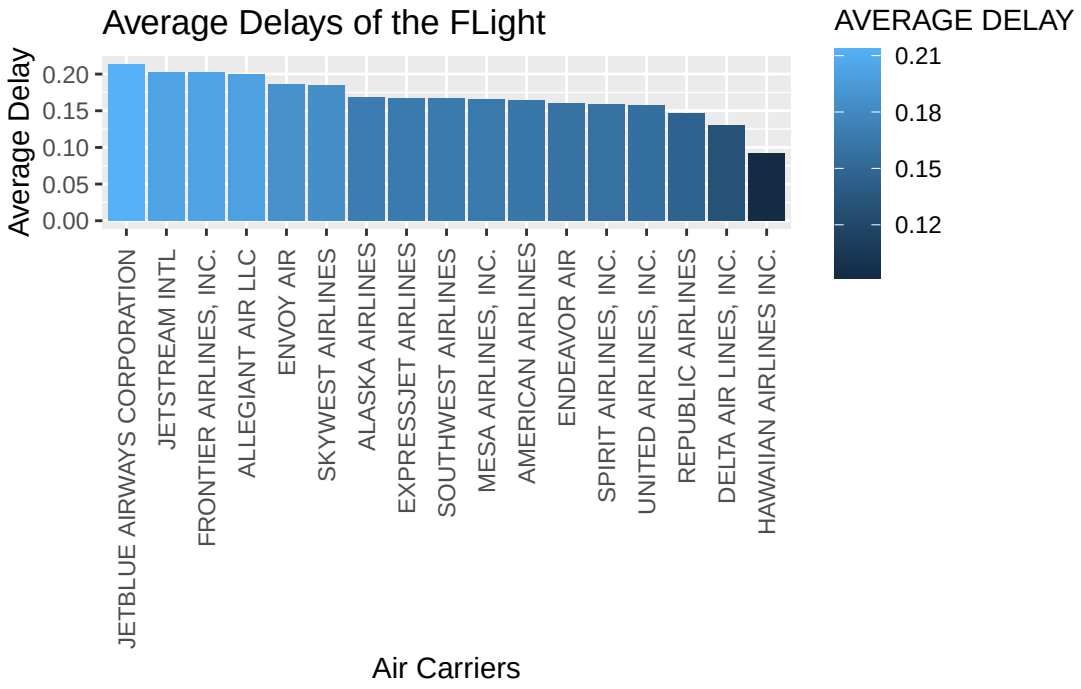
5. The average delay rate of each air carrier?

```
# Select air carrier name and delay times
A = df %>%
  select(`Air Carrier Name`,DEP_DEL15)%>%
  filter(DEP_DEL15 == 1)%>%
  group_by(`Air Carrier Name`)%>%
  summarise(Value=n()) %>%
  arrange(desc(Value))


#Combine all the air carrier name and total number
B = df %>%
  select(`Air Carrier Name`)%>%
  group_by(`Air Carrier Name`)%>%
  summarise(Value1=n()) %>%
  arrange(desc(Value1))

# Left join to columms for further calculation
C=left_join(A,B, by= 'Air Carrier Name')

# Cancellation rate  = # of Cancels / # total flight
C$`AVERAGE DELAY` = C$Value/C$Value1

#Rename columns
colnames(C)[2] = "Delayed Flight"
colnames(C)[3] = "Total flight"

#Bar chart for air carrier name vs average delay
kable(C)
```

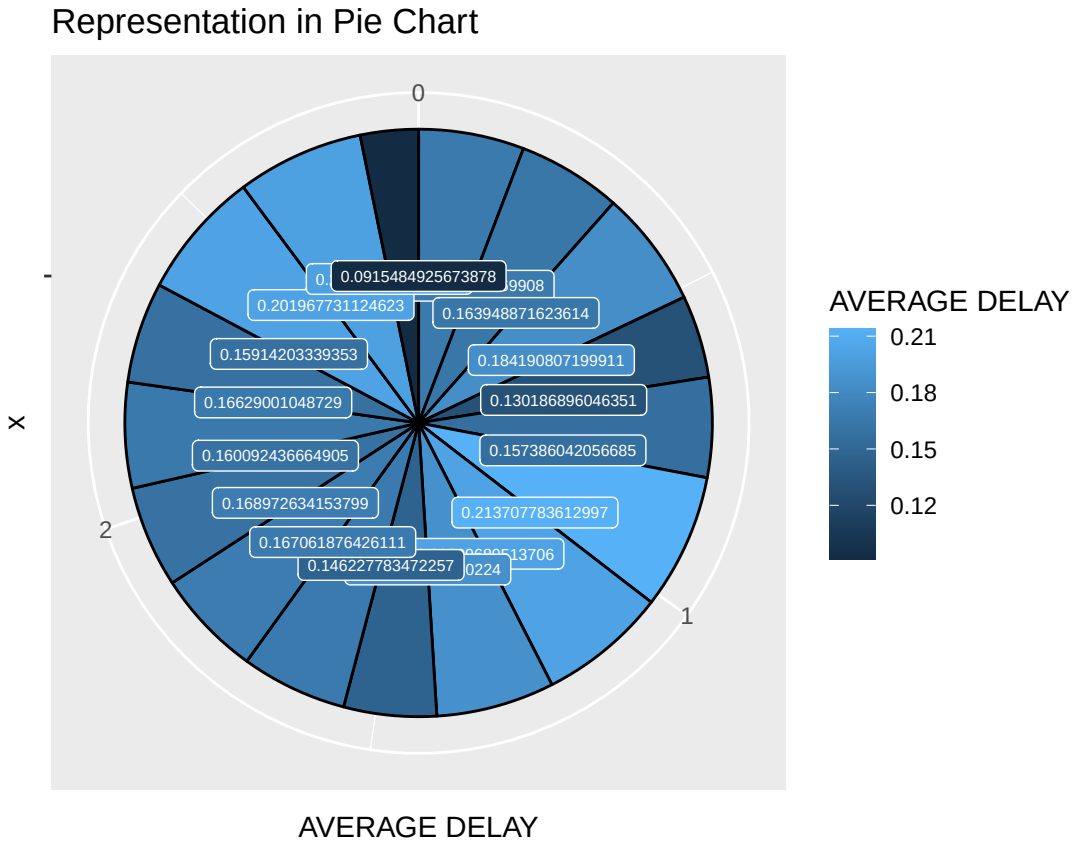
Air Carrier Name	Delayed Flight	Total flight	AVERAGE DELAY
SOUTHWEST AIRLINES	67278	404002	0.1665289
AMERICAN AIRLINES	47817	291658	0.1639489
SKYWEST AIRLINES	46212	250892	0.1841908
DELTA AIR LINES, INC.	38019	292034	0.1301869
UNITED AIRLINES, INC.	28920	183752	0.1573860
JETBLUE AIRWAYS CORPORATION	20021	93684	0.2137078
JETSTREAM INTL	17908	88611	0.2020968
ENVOY AIR	17392	93221	0.1865674
REPUBLIC AIRLINES	15122	103414	0.1462278
EXPRESSJET AIRLINES	14936	89404	0.1670619
ALASKA AIRLINES	13263	78492	0.1689726
ENDEAVOR AIR	13024	81353	0.1600924
MESA AIRLINES, INC.	11258	67701	0.1662900
SPIRIT AIRLINES, INC.	10046	63126	0.1591420
FRONTIER AIRLINES, INC.	8437	41774	0.2019677

Air Carrier Name	Delayed Flight	Total flight	AVERAGE DELAY
ALLEGIANT AIR LLC	6044	30219	0.2000066
HAWAIIAN AIRLINES INC.	2408	26303	0.0915485

```
C %>%
  ggplot(aes(x=reorder(`Air Carrier Name`,-`AVERAGE DELAY`), y=`AVERAGE DELAY`,fill= `AVERAGE DELAY`))+
  geom_bar(stat = "identity")+
  ggtitle("Average Delays of the FLight")+
  labs(x= "Air Carriers", y= "Average Delay")+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1, ))+
  theme(plot.margin=unit(c(1,1,1.5,1.2),"cm"))
```



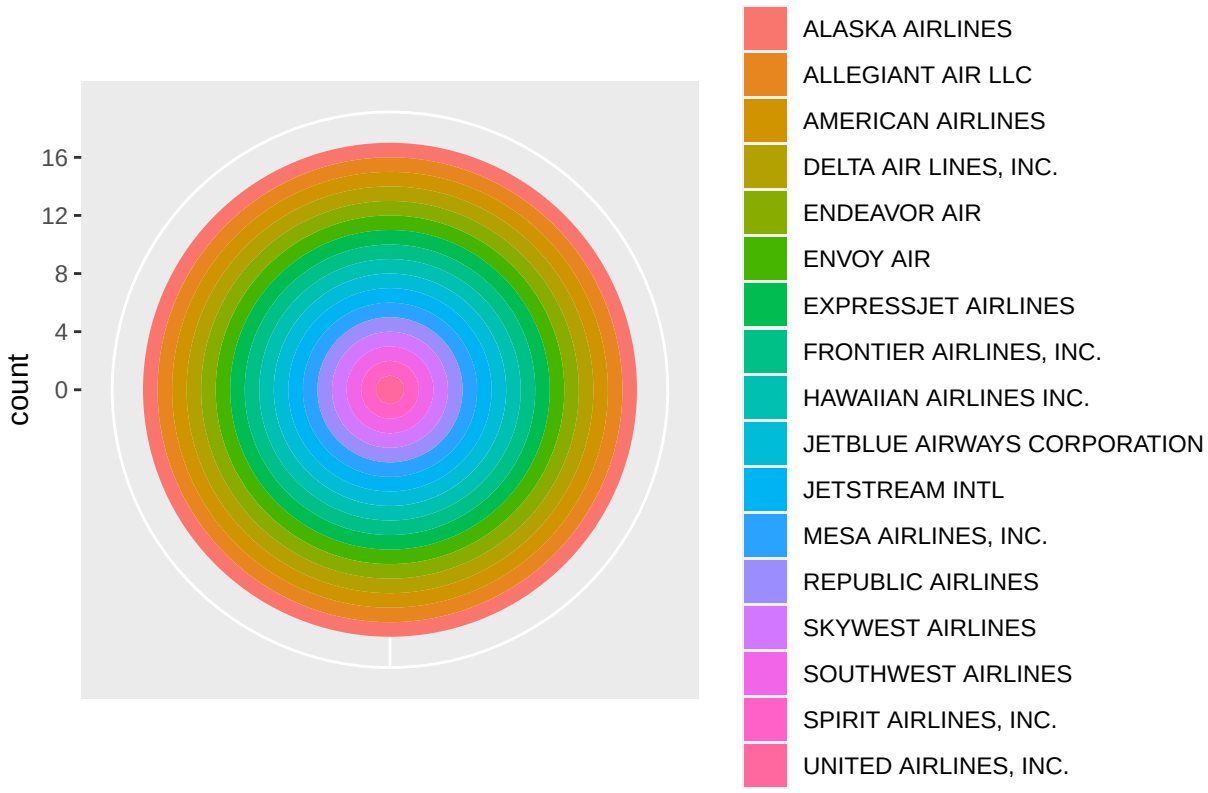

```
#Create a pie chart for the same task
C%>%
  ggplot( aes(x = "", y = `AVERAGE DELAY`, fill = `AVERAGE DELAY`)) +
  geom_col(color = "black") +
   geom_label(aes(label = `AVERAGE DELAY`),
              position = position_stack(vjust = 1),
             size=2,
             color = "white",
            show.legend = FALSE) +
  coord_polar(theta = "y")+
ggtitle("Representation in Pie Chart")
```




```
# using highcharter package to create a pie chart with improved resolution and detail
# The output could not kint into a pdf
dout <- data_to_hierarchical(C, c(`Air Carrier Name`), `AVERAGE DELAY`) %>%
  hchart(type = "sunburst") %>%
  hc_title(text = "Average Delays of the Flight")
  # hc_subtitle(text = "#techanswers88",style = list(fontSize = "16px",fontWeight = "bold", color = "re

#printing the plot
dout
```

```
# Sniped the output plot and insert it as a png file
include_graphics("pie.png")
```


```
#Other alternative for pie chart using ggplot
C %>%
  ggplot( aes(x = factor(1), fill = factor(`Air Carrier Name`))) +
  geom_bar(width = 1) +
  coord_polar() +
  labs(x = NULL, fill = NULL)
```



6. Which Branded Codeshare Partners has the most flights in the given four months.

```
kable(
  df%>%
    select(`Air Carrier Name`, `Codeshare Partner`)%>%
    group_by(`Codeshare Partner`)%>%
    summarise(Value1=n()) %>%
    arrange(desc(Value1))%>%
    slice(1)
  ,caption = "Branded Codeshare Partners with the Most flights"
  ,col.names = c("Branded Codeshare", "Number of Flights")
)
```

Table 43: Branded Codeshare Partners with the Most flights

Branded Codeshare	Number of Flights
American Airlines Branded Codeshare Partners	1024458

From the table, it is clear that American Airlines Branded Codeshare Partners has the most flights for all the four months.

7. At which destination with the airport id had the most flight canceled? – least – comparison graphs

```
kable(
  df%>%
  select(DEST_AIRPORT_ID,CANCELLED)%>%
  group_by(DEST_AIRPORT_ID)%>%
  arrange(DEST_AIRPORT_ID, order=TRUE)%>%
  summarise(Value=n()) %>%
  slice(1),
  caption = 'the airport id which had the most flight cancelled',
)
```

Table 44: the airport id which had the most flight cancelled

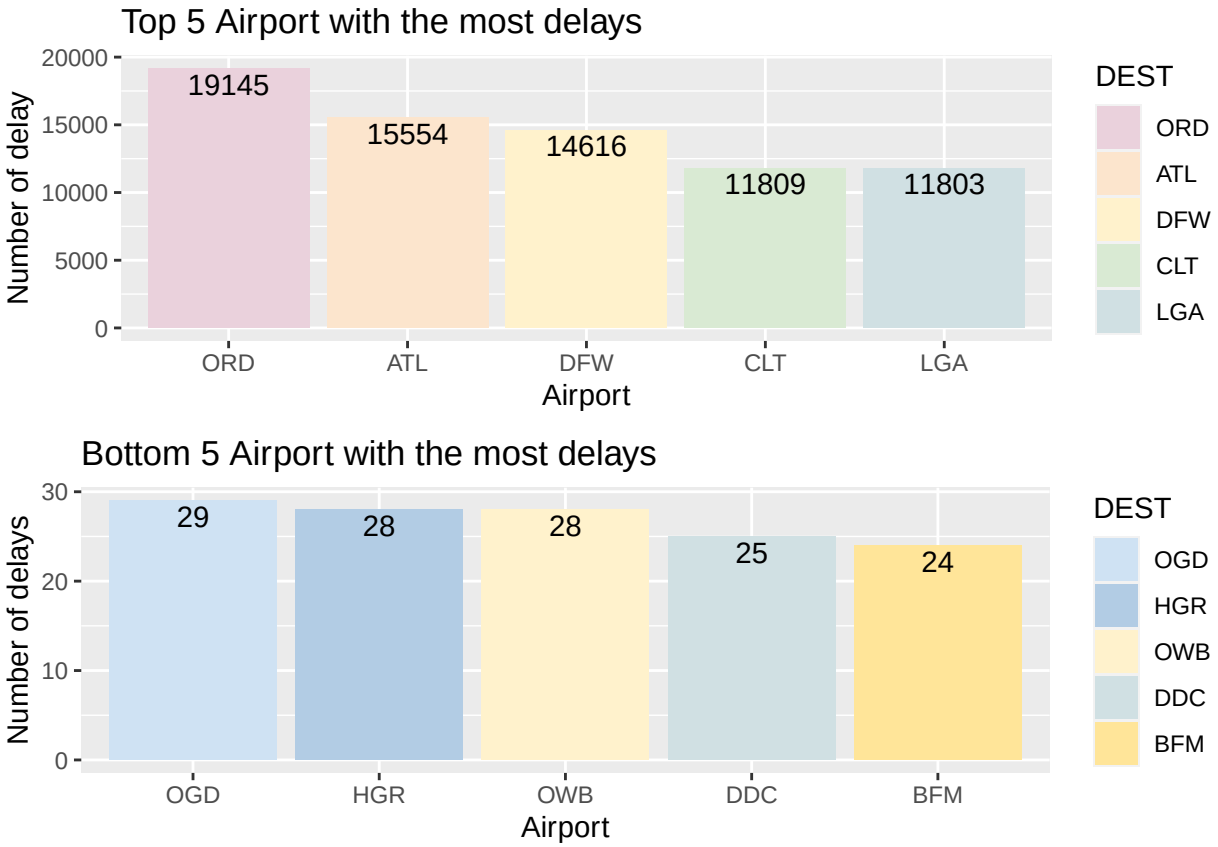
DEST_AIRPORT_ID	Value
10135	1472

The airport id 10135 had the greatest number of flights canceled which was 1472.

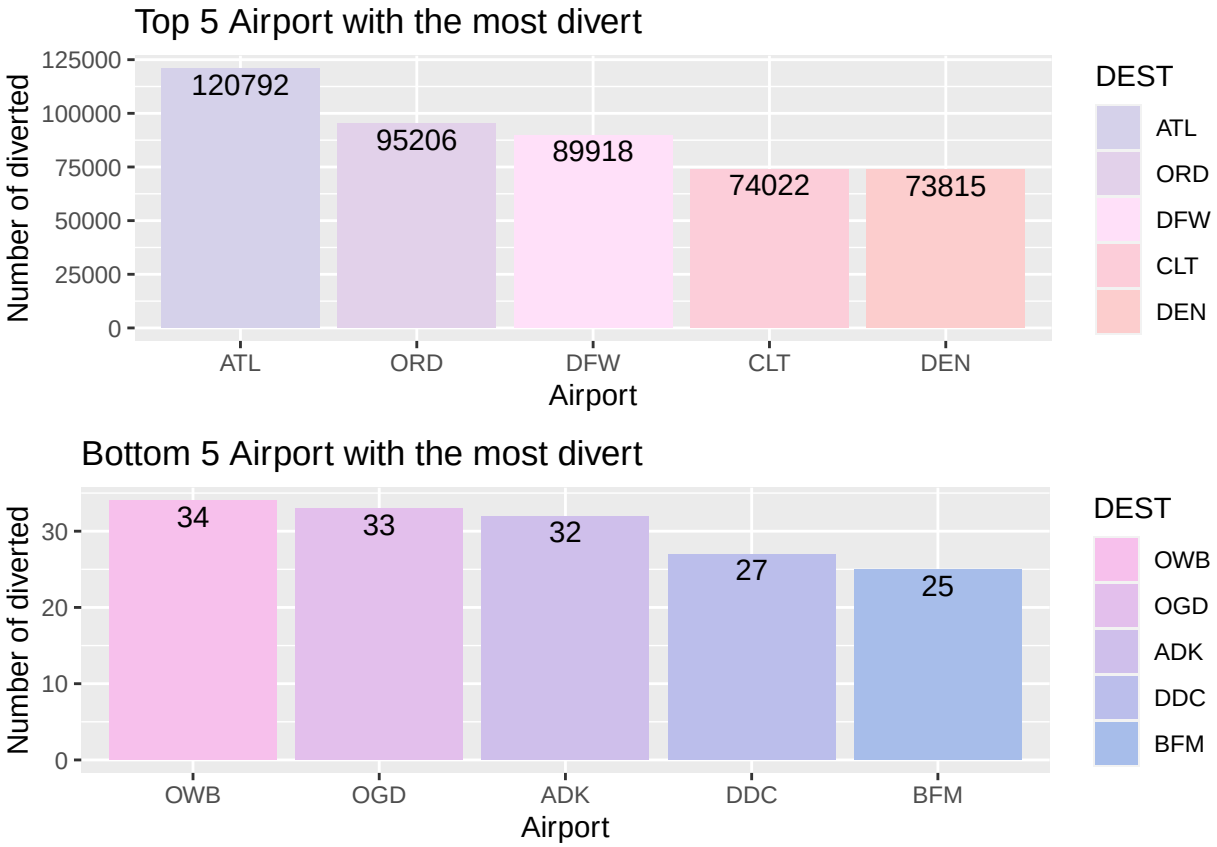
8. At which destination had the most flight delays and the most flight diverted? – least

```
# Create the first plot by filter out designation with number of delays. Then plot them in decreasing o
plot81 =
df %>%
  filter (DEP_DEL15 == 1) %>%
  count(DEST, sort=TRUE) %>%
  slice (1:5) %>%
  ggplot(aes(x=reorder(DEST,-n), y=n, fill= DEST, label= n)) +
  geom_bar(stat = "identity") +
  scale_fill_manual(values = c("ORD" = "#EAD1DC","ATL" = "#FCE5CD","DFW" = "#FFF2CC","CLT"="#D9EAD3","L
  ggtitle("Top 5 Airport with the most delays")+
  labs(x= "Airport", y= "Number of delay")+
  geom_text(aes(label=n), position=position_dodge(width=0.9), vjust=1.25)

# Create the second plot by filter out designation with number of delays. Then select the last five airp
plot82 =
df %>%
  filter (DEP_DEL15 == 0) %>%
  count(DEST, sort=TRUE) %>%
  tail(n=5) %>%
  ggplot(aes(x=reorder(DEST,-n), y=n, fill= DEST, label= n)) +
  geom_bar(stat = "identity") +
  scale_fill_manual(values = c("OGD" = "#cfe2f3","HGR" = "#b2cce4","OWB" = "#fff2cc","DDC"="#d0e0e3","BI
  ggtitle("Bottom 5 Airport with the most delays")+
  labs(x= "Airport", y= "Number of delays")+
  geom_text(aes(label=n), position=position_dodge(width=0.9), vjust=1.25)

# Create the third plot by filter out designation with number of divert Then plot them in decreasing or
```

```
plot83 =
df %>%
  filter (DIVERTED == 0) %>%
  count(DEST, sort=TRUE) %>%
  slice (1:5)  %>%
  ggplot(aes(x=reorder(DEST,-n), y=n, fill= DEST, label= n)) +
  geom_bar(stat = "identity") +
  scale_fill_manual(values = c("ATL" = "#d5d1ea","ORD" = "#e2d1ea","DFW" = "#ffe0f9","CLT"="#fccdd9","D
  ggtitle("Top 5 Airport with the most divert")+
  labs(x= "Airport", y= "Number of diverted")+
  geom_text(aes(label=n), position=position_dodge(width=0.9), vjust=1.25)

# Create the fourth plot by filter out designation with number of divert Then select the last five airp
plot84 =
df %>%
  filter (DIVERTED == 0) %>%
  count(DEST, sort=TRUE) %>%
  tail(n=5) %>%
  ggplot(aes(x=reorder(DEST,-n), y=n, fill= DEST, label= n)) +
  geom_bar(stat = "identity") +
  ggtitle("Bottom 5 Airport with the most divert")+
  scale_fill_manual(values = c("OWB" = "#f7c0ec","OGD" = "#e3bfec","ADK" = "#cfbfeb","DDC"="#bbbeeb","BI
  labs(x= "Airport", y= "Number of diverted")+
  geom_text(aes(label=n), position=position_dodge(width=0.9), vjust=1.25)

# Combine Plots and present
grid.arrange(plot81,plot82)
```




```
grid.arrange(plot83,plot84)
```



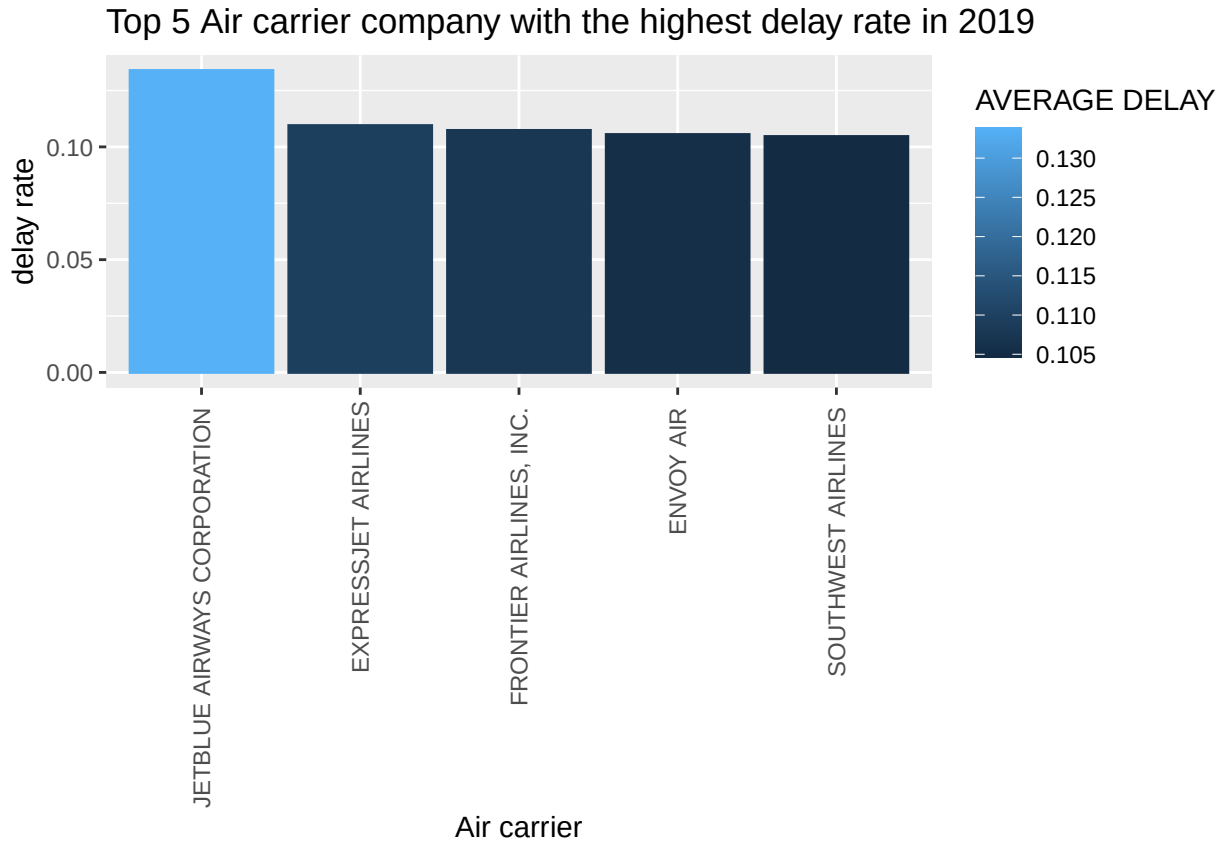
The first two charts segregates the top 5 and bottom 5 airports with the most delays whereas the other two charts displays the same but for most diversions.

From the bar charts, it is easy to point out that the airport with the most delays is ORD (Chicago O'Hare International Airport), total of 18229 delays in four months. The airport with the most diverts is ATL (Atlanta Hartsfield-Jackson International Airport), total of 120572 diverts in four months. The airport with the least delays and diverts is BFM (Mobile Downtown Airport), total of 24 delays and 25 diverts in four months.

9. The carrier performance in 2019 vs 2020, analyzing the delay rate for the best top 5 carriers and the worst 5 carriers in both years

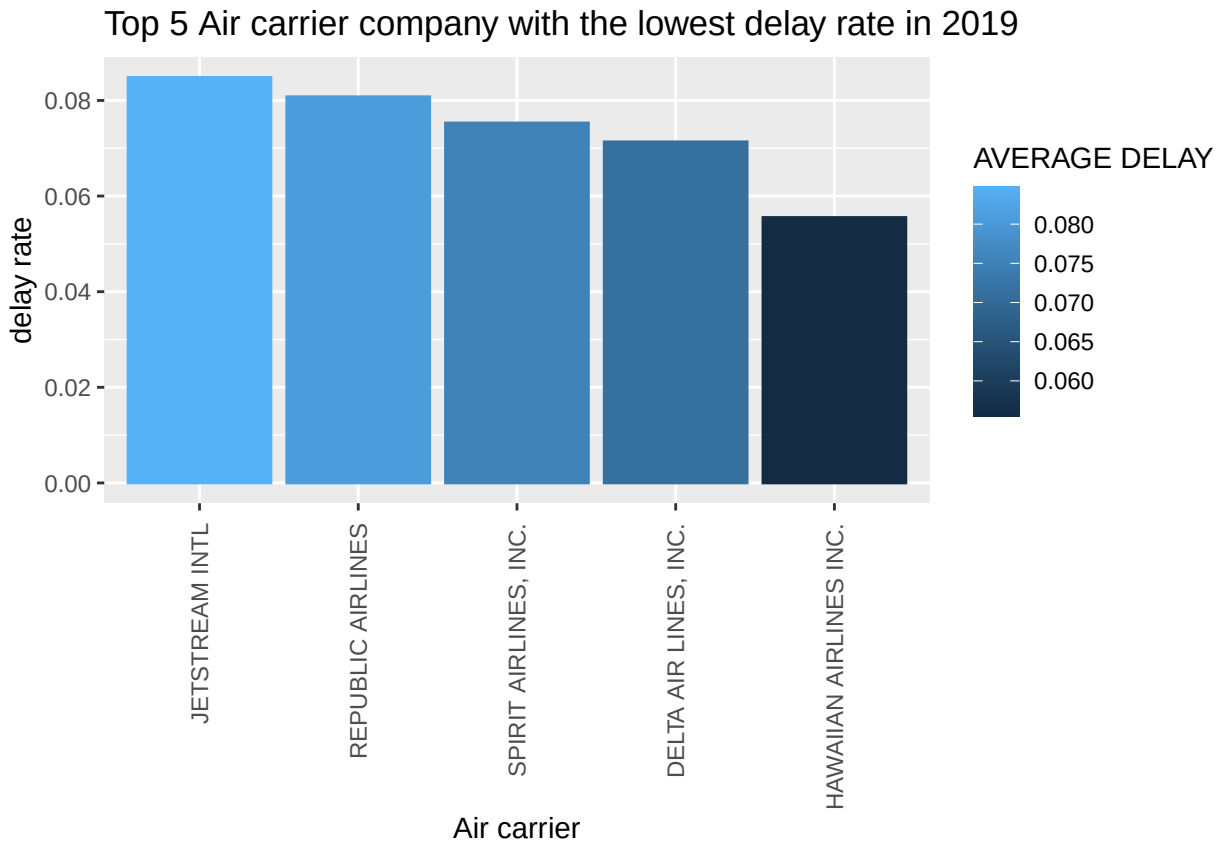
```
# Select the first year, and pick out the delay rate
DE20191 = df %>%
  filter(Day.of.month >= "2019-01-01" & Day.of.month<= "2019-02-28") %>%
  select(`Air Carrier Name`,DEP_DEL15)%>%
  filter(DEP_DEL15 == 1)%>%
  group_by(`Air Carrier Name`)%>%
  summarise(Value=n()) %>%
  arrange(desc(Value))

# Group by colname and find delay calculated earlier
DE20192 = df %>%
  select(`Air Carrier Name`)%>%
  group_by(`Air Carrier Name`)%>%
  summarise(Value1=n()) %>%
```

```
  arrange(desc(Value1))

#Left join
DE2019=left_join(DE20191,DE20192, by= 'Air Carrier Name')

#Delay rate
DE2019$`AVERAGE DELAY` = DE2019$Value/DE2019$Value1

#modify name
colnames(DE2019)[2] = "Delayed Flight"
colnames(DE2019)[3] = "Total flight"

# Plot Top 5 Air carrier company with the highest delay rate in 2019
DE2019 %>%
  arrange(desc(`AVERAGE DELAY`)) %>%
  top_n(5) %>%
  ggplot(aes(x=reorder(`Air Carrier Name`,-`AVERAGE DELAY`), y=`AVERAGE DELAY`, color=`AVERAGE DELAY`, f
  geom_bar(stat = "identity") +
  ggtitle("Top 5 Air carrier company with the highest delay rate in 2019")+
  labs(x= "Air carrier", y= "delay rate")+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```



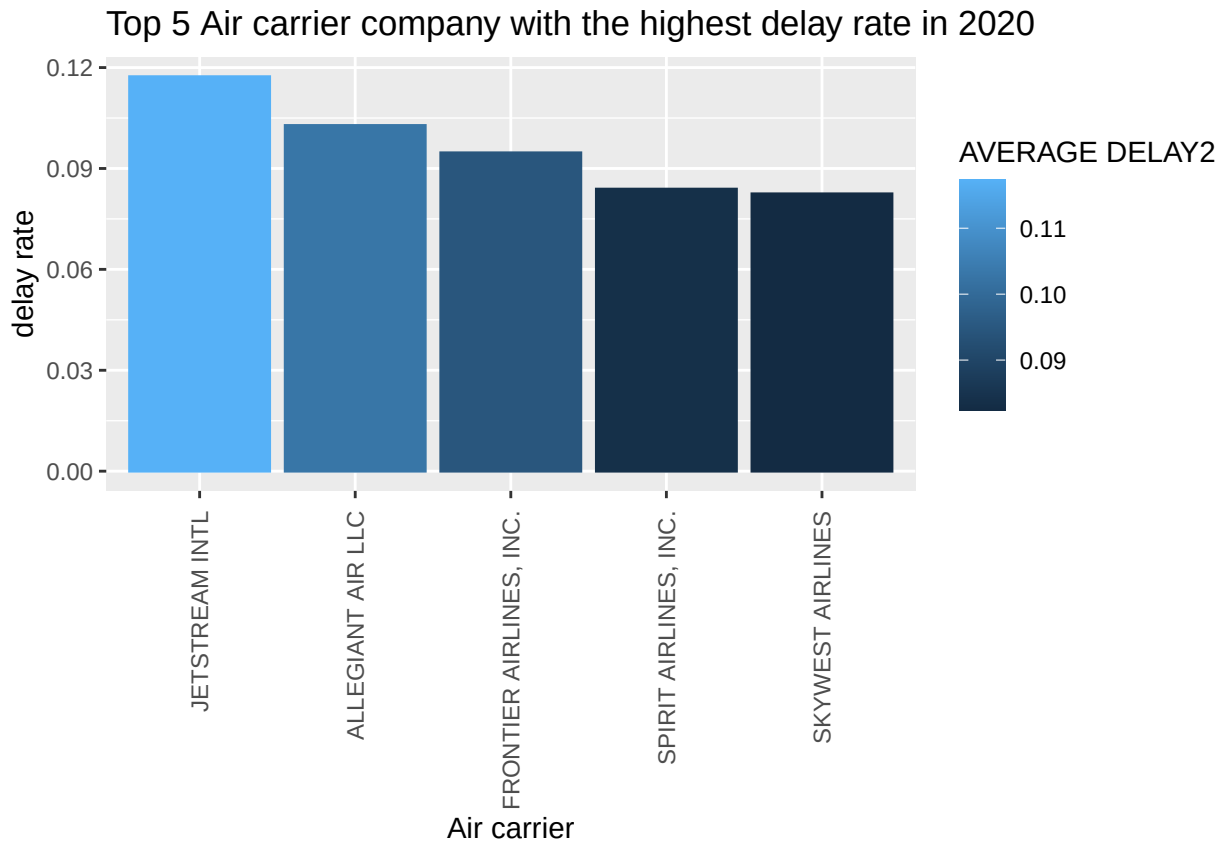

```
# Plot Top 5 Air carrier company with the lowest delay rate in 2019
DE2019 %>%
  arrange(desc(`AVERAGE DELAY`)) %>%
```

```
  tail(n=5) %>%
  ggplot(aes(x=reorder(`Air Carrier Name`,-`AVERAGE DELAY`), y=`AVERAGE DELAY`, color=`AVERAGE DELAY`, f
  geom_bar(stat = "identity") +
  ggtitle("Top 5 Air carrier company with the lowest delay rate in 2019")+
  labs(x= "Air carrier", y= "delay rate")+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```



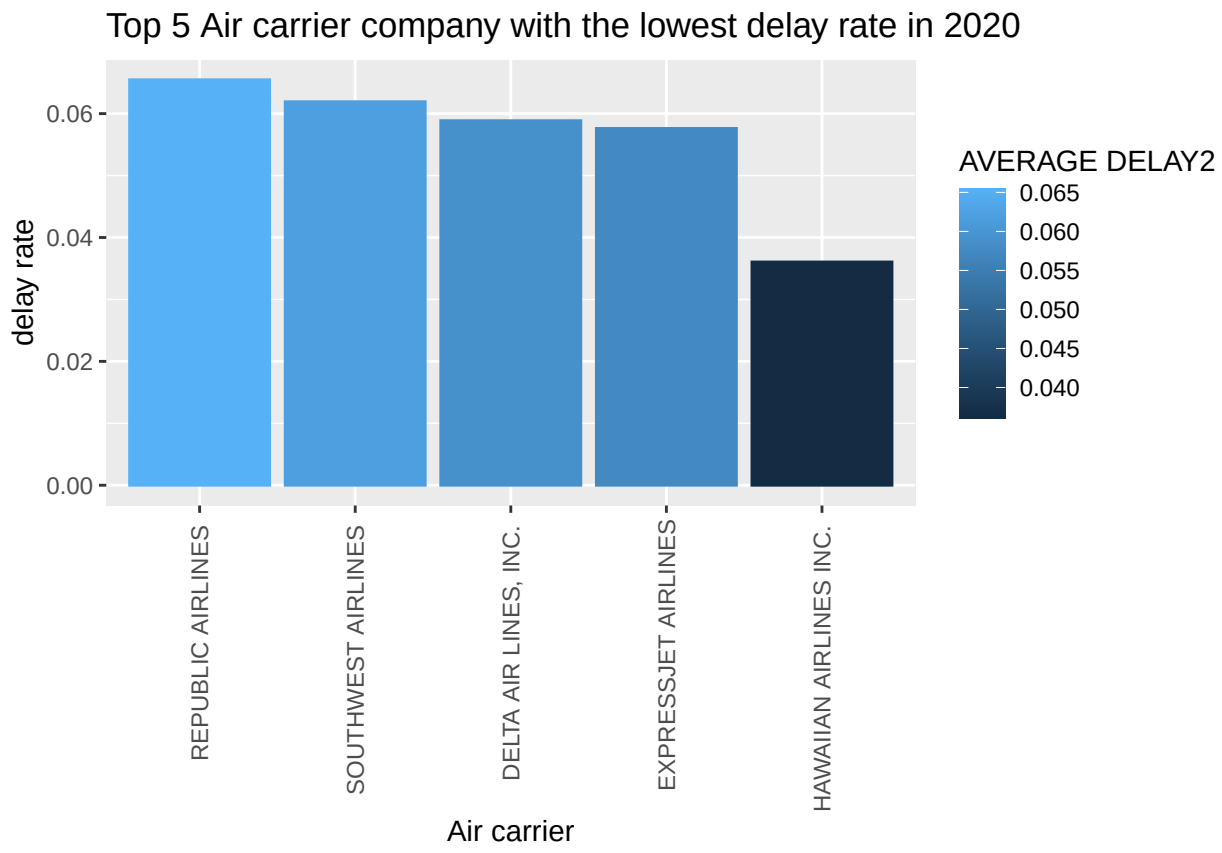

```
# Select the second year, and pick out the delay rate
DE20201 = df %>%
  filter(Day.of.month >= "2020-01-01" & Day.of.month<= "2020-02-29") %>%
  select(`Air Carrier Name`,DEP_DEL15)%>%
  filter(DEP_DEL15 == 1)%>%
  group_by(`Air Carrier Name`)%>%
  summarise(Value=n()) %>%
  arrange(desc(Value))

# Group by colname and find delay calculated earlier
DE20202 = df %>%
  select(`Air Carrier Name`)%>%
  group_by(`Air Carrier Name`)%>%
  summarise(Value1=n()) %>%
  arrange(desc(Value1))

# Left join
DE2020=left_join(DE20201,DE20202, by= 'Air Carrier Name')
```

```r
# Calculate delay rate
DE2020$`AVERAGE DELAY2` = DE2020$Value/DE2020$Value1

# Rename column
colnames(DE2020)[2] = "Delayed Flight2"
colnames(DE2020)[3] = "Total flight2"

# Top 5 Air carrier company with the highest delay rate in 2020
DE2020 %>%
  arrange(desc(`AVERAGE DELAY2`)) %>%
  top_n(5) %>%
  ggplot(aes(x=reorder(`Air Carrier Name`,-`AVERAGE DELAY2`), y=`AVERAGE DELAY2`, color=`AVERAGE DELAY2
  geom_bar(stat = "identity") +
  ggtitle("Top 5 Air carrier company with the highest delay rate in 2020")+
  labs(x= "Air carrier", y= "delay rate")+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```

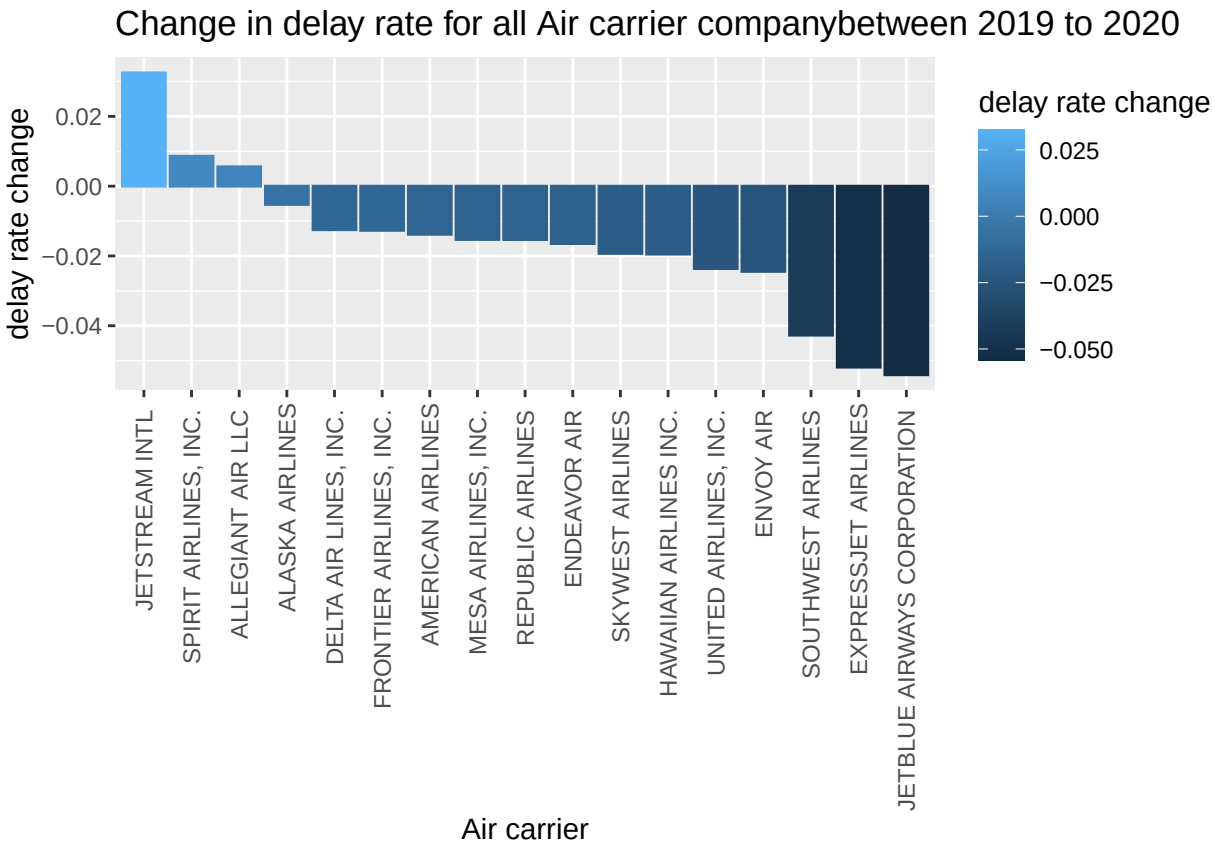



```r
#Top 5 Air carrier company with the lowest delay rate in 2020
DE2020 %>%
  arrange(desc(`AVERAGE DELAY2`)) %>%
  tail(n=5) %>%
  ggplot(aes(x=reorder(`Air Carrier Name`,-`AVERAGE DELAY2`), y=`AVERAGE DELAY2`, color=`AVERAGE DELAY2
  geom_bar(stat = "identity") +
  ggtitle("Top 5 Air carrier company with the lowest delay rate in 2020")+
  labs(x= "Air carrier", y= "delay rate")+
```

```
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```




```
# Join two data together
DE = left_join(DE2019, DE2020)
DE$`delay rate change` = DE$`AVERAGE DELAY2`-DE$`AVERAGE DELAY`

# Compare delay rate for both year
DE %>%
  select(`Air Carrier Name`,`delay rate change`) %>%
  ggplot(aes(x=reorder(`Air Carrier Name`,-`delay rate change`), y=`delay rate change`, color=`delay ra
  geom_bar(stat = "identity") +
  ggtitle("Change in delay rate for all Air carrier companybetween 2019 to 2020")+
  labs(x= "Air carrier", y= "delay rate change")+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```



From the first graph, the top company with highest delay rate in 2019 are: JetBlue, ExpressJet, Frontier, Envoy and Southwest. From the second graph, the top company with lowest delay rate in 2019 are: JetStream, Republic, Spririt, Delta and Hawaiian. From the third graph, the top company with highest delay rate in 2020 are: JetStream, Allegiant, Frontier, Spirit and Skywest From the fourth graph, the top company with lowest delay rate in 2019 are: Republic, Southwest, Delta, ExpressJet and Hawaiian From 2019 to 2020, most of the company managed to cut down the delay rate, except three companies, JetStream, Spirit and Allegant.

The result fits daily life experience, lower class budget airlines such as Southwest usually have a higher delay rate. As a comparison, Hawaiian Airline has its reputation in on-time departure. Within reasonable constraints, fly with Hawaiian Airlines for better on-time flight experience. ## Conclusion

To begin with, the group used various Data aquisition and Data wrangling techniques in order to filter out irrelevant data and make it more stable for our analysis in order to answer our business questions.

Initially, the group researched on various websites and open source data sets sharing platforms and finalized the January and February flight information for the years 2019 which was available on kaggle. The group chose this dataset because it had NA values, had some discrepancies like wrong date and time format, and the dataset had enough number of rows for manipulation. Furthermore, the group decided to add two more datasets for the year 2020 for both the months in order to merge it into a final one which made it easier to perform analysis.

While cleaning our dataset, the group realized that there were some flights which were in the air for less than 10 mins which is impractical and hard to predict the actual flight's duration hence, the group fixed this issue in the 'Check Outliers' section. Alternatively, the group can perform unsupervised Machine learning to predict the actual flight duration.Because of the difficulty level of the Machine learning and the outcome of the procedure does not align with the original goal of the project, the group decided to replace those NA values with zeros. The replacement does not have any impact on the data-driving decisions about

flight duration. Any column with NA values that would not participate in any data driving decisions were abandoned.

Under structuring section, multiple dates and time changes was being addressed. One of the encountered issue was, the original data saved weekdays as integer numbers("1" as Monday, "7" as Sunday). By default, the ISO standard indicate the number 1 as Sunday. The team went pass that by directly converting date to weekdays.

Package ggplot2 as one of the critical package that should be mastered, the team present several plots with ggplots2 including bar chart, line chart, etc. For the practicing purpose, the team managed to present many plot types if possible. For problem 5, the group was planned to generate a pie chart using ggplots. Due to the over-complicated air carrier names, ggplots could not present the plot with well structured labels. The team then compute several other plot packages to solve the issue. Some of the pie chart could not able to knit into a pdf file.

As the project requirement stated, the final submission should be in pdf format by using knit to pdf in-build function within Rstudio. With computation restrictions such as short in memory, the team spent more than necessary amount of time to knit the pdf file. Some minor tips was discovered during attempts, such as add cache = TRUE for R code block to aviod calculating same factors repeatedly. Due to the fact that a large data set takes overwhelming amont of memory, kniting would still causes the device not responding or even crushing.

As a conclusion, the team successfully encountered all the constrains and met the initial objectives. Proper procedures of data wrangling and data driven business question were mastered after the complement of the project.