DATA 607 LAB4 WEEK4

Farhod Ibragimov

2025-02-20

Let's load the necessary libraries:

```
# Loading libraries
library(tidyverse)
library(RMySQL)
library(DBI)
library(knitr)
library(tidyr)
library(kableExtra)
```

Loading and transforming data

I create a table in the `flight_data.csv` file to store the data from Week 4 assignment and uploaded it to Github.

This code reads the `flight_data.csv`from my Github URL "https://raw.githubusercontent.com/farhodibr/CUNY-SPS-MSDS/main/DATA607/LAB4/flight_data.csv" and creates `data` table.

```
# Reading CSV file from my Github and load it to the database
data <- read_csv("https://raw.githubusercontent.com/farhodibr/CUNY-SPS-MSDS/main/DATA607/LAB4/flight_da
data |>
  kable() |>
  kable_styling(full_width = F)
```

...1	...2	Los Angeles	Phoenix	San Diego	San Francisco	Seattle
ALASKA	On Time	497	221	212	503	1841
NA	Delayed	62	12	20	102	305
AM WEST	On Time	694	4840	383	320	201
NA	Delayed	117	415	65	129	61

The problems why this data table is not tidy and not suited well for analyses:

- "Los Angeles", "Phoenix", "San Diego", "San Francisco", "Seattle" are city names. In a tidy dataset, these should be values in a column named "city".
- "`...2`" column represents flight's status and its values "On Time" and "Delayed" should be variables since they have numerical values spread across the table

From here I'm tidying the data table to resolve the problems above:

```
# Checking column names
colnames(data)
```

```
## [1] "...1"          "...2"          "Los Angeles"   "Phoenix"
## [5] "San Diego"     "San Francisco" "Seattle"
```

The code below is renaming and standardizing column names, handling potential missing values in the "`Airline`" column, standardizing string values within the "`Status`" column (lowercase and underscores), making a readable table format.

```r
# Renaming first two columns
colnames(data)[1:2] <- c("Airline", "Status")

# Filling missing values in the Airline column
data <- data |>
  fill(Airline, .direction = "down")

# Turning all column names data in the table lowercase and replacing white spaces with underscore
colnames(data) <- tolower(colnames(data))
colnames(data) <- str_replace_all(colnames(data), " ", "_")

# Replacing white spaces with underscore in all string values in the Status column
data <- data |>
  mutate(across(status,
              str_replace_all, " ", "_")
         ) |>
  mutate(across(status, tolower))


data |>
  kable() |>
  kable_styling(full_width = F)
```

airline	status	los_angeles	phoenix	san_diego	san_francisco	seattle
ALASKA	on_time	497	221	212	503	1841
ALASKA	delayed	62	12	20	102	305
AM WEST	on_time	694	4840	383	320	201
AM WEST	delayed	117	415	65	129	61

Pivotting

This code first pivoting the table into a long format to create a "`city`" variable with city names as its values. At the same time it takes the data values associated with each city and puts them into a new column "`count`"

After that the code pivoting table into wider format to create two variables "`on_time`" and "`delayed`", and assigns corresponding numerical values from the"`count`" column , aligned with the corresponding "`airline`" and "`city`" variables. These "`on_time`" and "`delayed`" variables were values of "`status`" variable.

```r
# pivoting the data
airline_city_long_table <- data |>
  pivot_longer(
    cols = -c(airline, status),
    names_to = "city",
    values_to = "count"
  ) |>
  pivot_wider(
    names_from = status,
    values_from = count
  ) |>
  mutate(airline = str_to_lower(airline)) |>
  mutate(airline = str_replace_all(airline, "\\s+", "_")) |>
  arrange(airline, city)

colnames(airline_city_long_table) <- str_replace_all(colnames(airline_city_long_table), " ", "_")
colnames(airline_city_long_table) <- tolower(colnames(airline_city_long_table))
```

```
# Viewing the data


airline_city_long_table |>
  kable() |>
  kable_styling(full_width = F)
```

airline	city	on_time	delayed
alaska	los_angeles	497	62
alaska	phoenix	221	12
alaska	san_diego	212	20
alaska	san_francisco	503	102
alaska	seattle	1841	305
am_west	los_angeles	694	117
am_west	phoenix	4840	415
am_west	san_diego	383	65
am_west	san_francisco	320	129
am_west	seattle	201	61

This new `airline_city_long_table`is a tidy format of the original data table and meets tidy data principles:

1. Each variable is a column; each column is a variable.

   `airline, city, on_time, delayed` :each of these has distinct information and values.

2. Each observation is a row; each row is an observation.

   Each row is a unique observation of airline-specific city route combination , showing its on-time and delayed flight counts for that route.

3. Each value is a cell; each cell is a single value.

   There are no cell has combined info or lists, just one single value per cell.

Analyses

Below I have done a couple of analyses.

This one creates a separate `flight_status_proportions`table with proportions of on_time and `delayed` for each airline-city route observation.

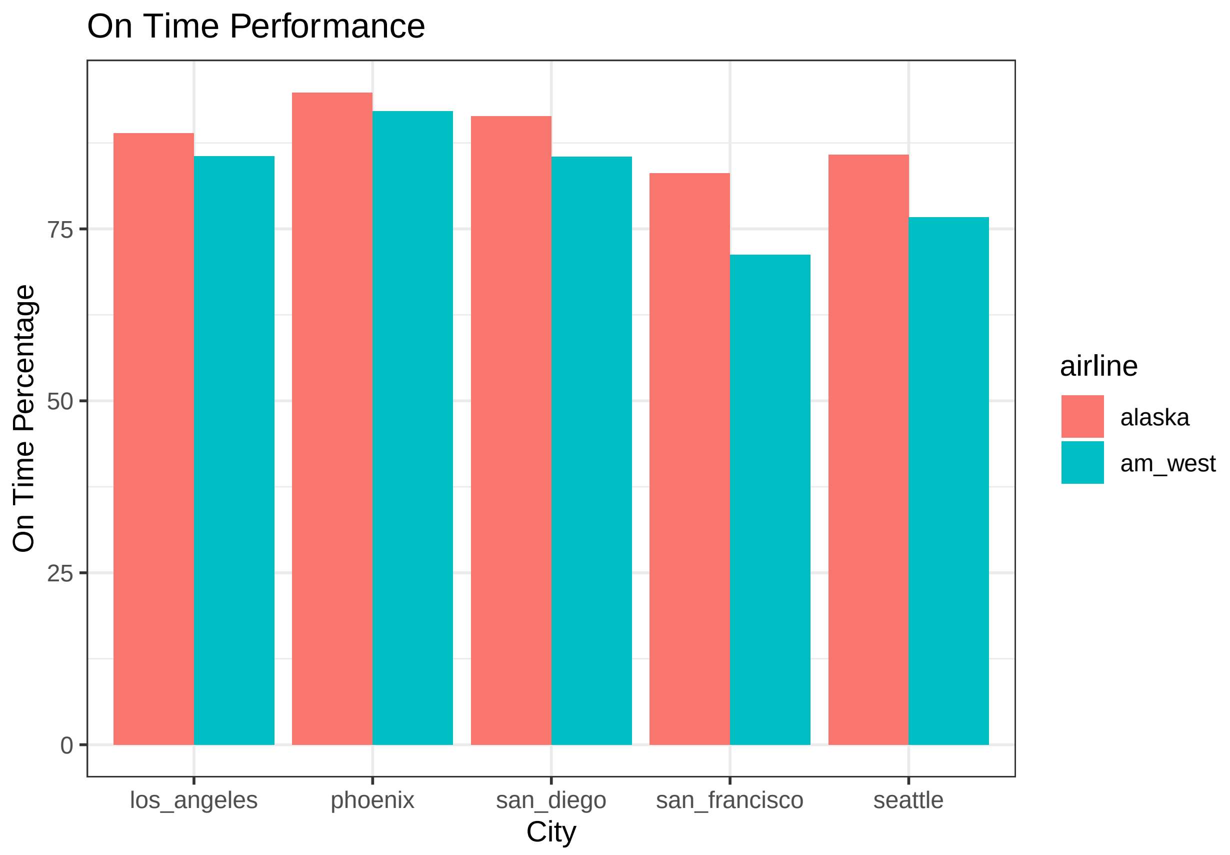
```
flight_status_proportions <- airline_city_long_table |>
  mutate(total_flights = on_time + delayed) |>
  mutate(on_time_percentage = round((on_time / total_flights) * 100, 2)) |>
  mutate(delayed_percentage = round((delayed / total_flights) * 100, 2)) |>
  select(airline, city, on_time_percentage, delayed_percentage)

flight_status_proportions |>
  kable() |>
  kable_styling(full_width = F)
```

airline	city	on_time_percentage	delayed_percentage
alaska	los_angeles	88.91	11.09
alaska	phoenix	94.85	5.15
alaska	san_diego	91.38	8.62
alaska	san_francisco	83.14	16.86
alaska	seattle	85.79	14.21
am_west	los_angeles	85.57	14.43
am_west	phoenix	92.10	7.90
am_west	san_diego	85.49	14.51
am_west	san_francisco	71.27	28.73
am_west	seattle	76.72	23.28

Let's create a column plot to compare on time and delay performances for each specific airline-city route observation:

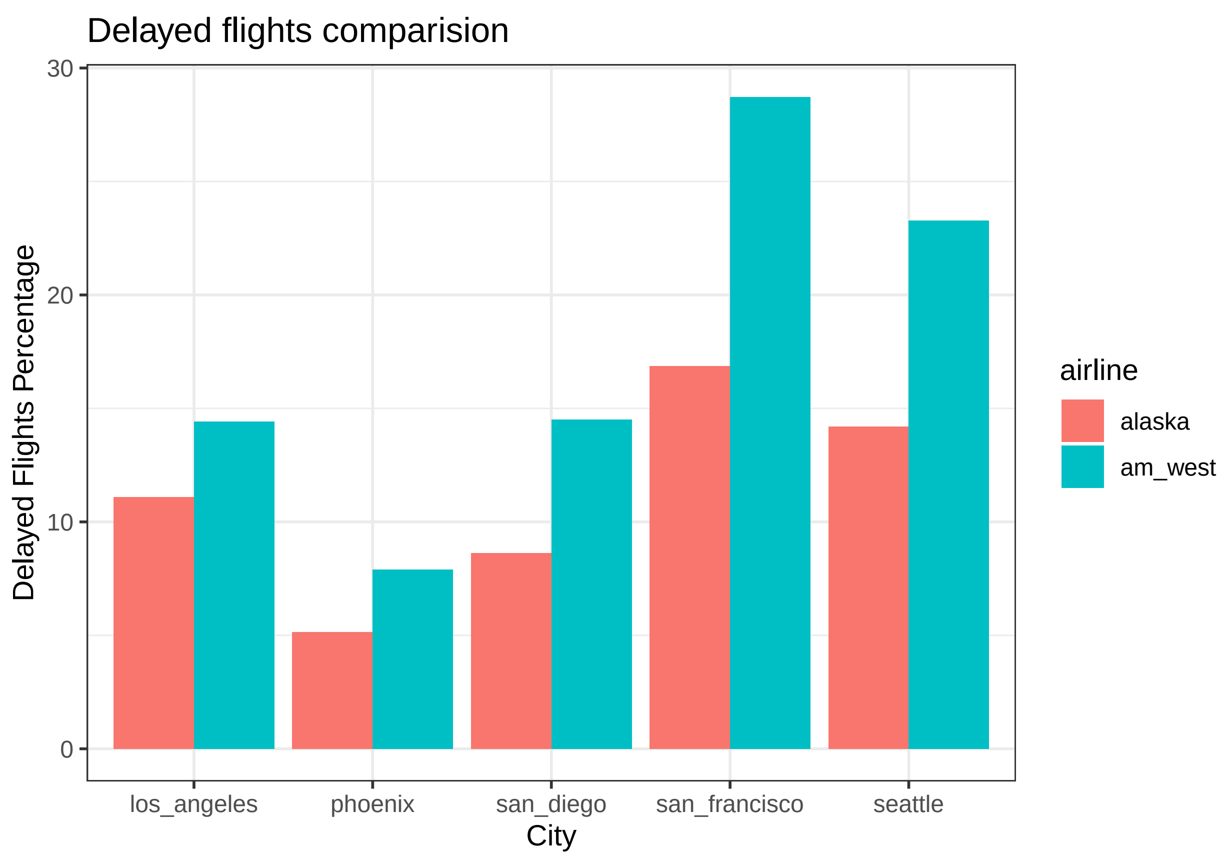
```
#analyzing the data
ggplot(
  flight_status_proportions,
  aes(
    x = city,
    y = on_time_percentage,
    fill = airline
  )
) +
  geom_col(position = "dodge") +
  labs(
    title = "On Time Performance",
    x = "City",
    y = "On Time Percentage"
  ) +
  theme_bw()
```



From this chart I can say that Alaska Airlines on time performance for each airline-city route is slightly better than AM West's.

This code creates a plot to compare delayed flights percentage for each airline-city observation:

```r
#analyzing the data
ggplot(
  flight_status_proportions,
  aes(
    x = city,
    y = delayed_percentage,
    fill = airline
  )
) +
  geom_col(position = "dodge") +
  labs(
    title = "Delayed flights comparision",
    x = "City",
    y = "Delayed Flights Percentage"
  ) +
  theme_bw()
```



From this plot I can say that AM West has more delayed flights percentage comparing to Alaska airlines. Especially on San Francisco and Seattle routes.

This code creates a new separate table to show total flights on each airline-city route observation:

```
airline_city_share_table <- airline_city_long_table |>

  group_by(airline, city) |>
  summarise(
    total_flights = sum(on_time + delayed),



  ) |>
  arrange(desc(total_flights))
```

```
## `summarise()` has grouped output by 'airline'. You can override using the
## `.groups` argument.
```

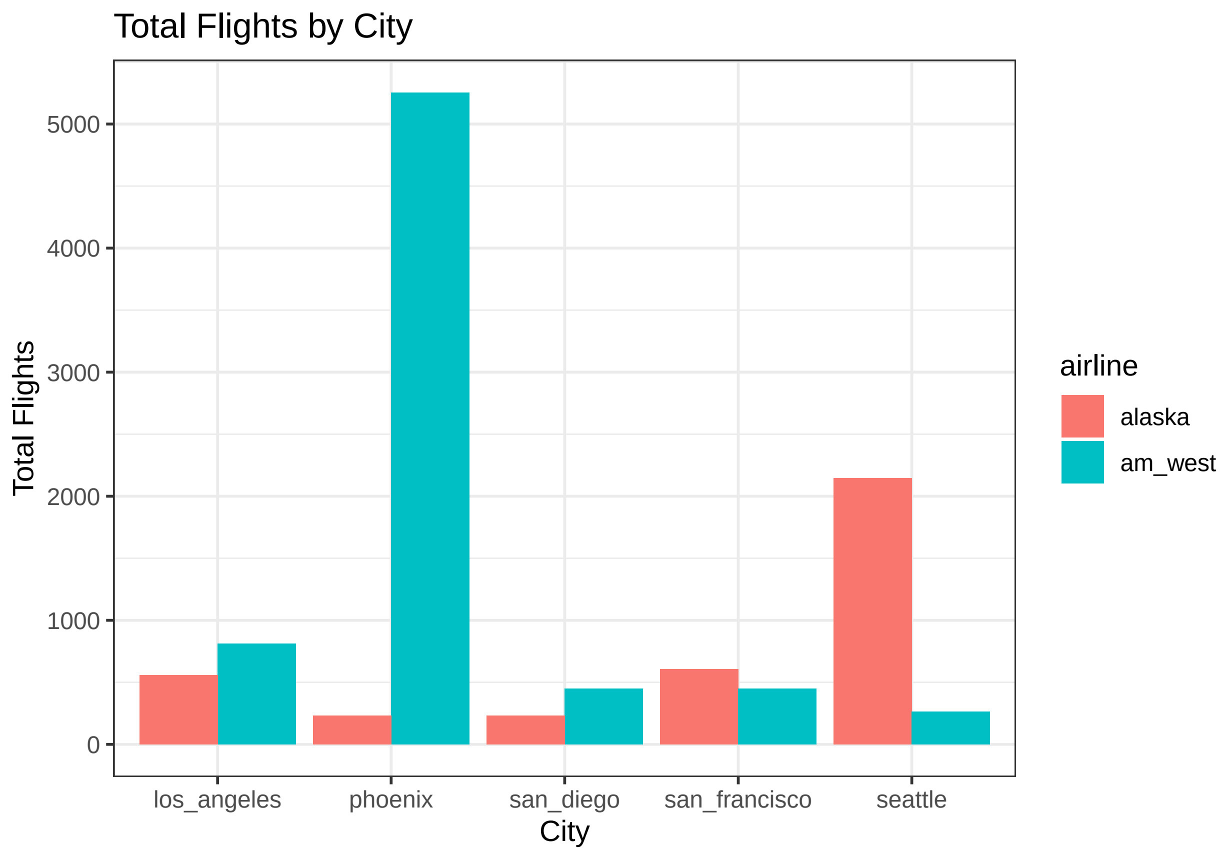
```
airline_city_share_table |>
  kable() |>
  kable_styling(full_width = F)
```

airline	city	total_flights
am_west	phoenix	5255
alaska	seattle	2146
am_west	los_angeles	811

alaska	san_francisco	605
alaska	los_angeles	559
am_west	san_francisco	449
am_west	san_diego	448
am_west	seattle	262
alaska	phoenix	233
alaska	san_diego	232

Let's compare each observation on this plot:

```r
ggplot(
  airline_city_share_table,
  aes(
    x = city,
    y = total_flights,
    fill = airline
  )
) +
  geom_col(position = "dodge") +
  labs(
    title = "Total Flights by City",
    x = "City",
    y = "Total Flights"
  ) +
  theme_bw()
```



From this plot I can see that AM West's busiest route is to Phoenix. And Alaska's busiest route is Seattle. Overall AM West Airlines has more flights combined from all routes than Alaska Airlines.

Upload to MySQL

Here is the code to connect to MySQL database. My credentials are stored in my Windows environment variables.

```
# Connecting to the MySQL database using the credentials stored in the environment variables
con <- dbConnect(
  drv = RMySQL::MySQL(),
  host = Sys.getenv("DB_HOST"),
  port = as.integer(Sys.getenv("DB_PORT")),
  dbname = Sys.getenv("DB_NAME"),
  username = Sys.getenv("DB_USER"),
  password <- Sys.getenv("DB_PASS")
)
```

This code creates new tables in MySQL and uploads the data into it

```
# more analysis
dbWriteTable(conn = con,
             value = data,
             name = "airline_city_messy_table",
             row.names = FALSE,
             overwrite = TRUE )
```

```
## [1] TRUE
```

```
dbWriteTable(conn = con,
             value = airline_city_long_table,
             name = "airline_city_long_table",
             row.names = FALSE,
             overwrite = TRUE )
```

```
## [1] TRUE
```

```
dbWriteTable(conn = con,
             value = flight_status_proportions,
             name = "flight_status_proportions",
             row.names = FALSE,
             overwrite = TRUE )
```

```
## [1] TRUE
```

```
dbWriteTable(conn = con,
             value = airline_city_share_table,
             name = "airline_city_share_table",
             row.names = FALSE,
             overwrite = TRUE )
```

```
## [1] TRUE
```

```
dbDisconnect(con)
```

```
## [1] TRUE
```