Practicum I CS5200

Modi, Harshkumar

Fall 2023

```
#install.packages("DBI")
#install.packages("tidyverse")
#install.packages("RMySQL")
#install.packages("dplyr")
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(RMySQL)
```

```
## Loading required package: DBI
```

```
library(DBI)
library(dplyr)
```

Connect to Database

```
con <- dbConnect(RMySQL::MySQL(),
                 host = "myrds.ccgfbvk5hpfc.us-east-1.rds.amazonaws.com",
                 port = 3306,
                 dbname = "birdStrikesDB",
                 username = "admin",
                 password = "CS5200Fall23!")
```

Creating database and tables

```sql
DROP TABLE IF EXISTS strikes;
```

```sql
DROP TABLE IF EXISTS conditions;
```

```sql
DROP TABLE IF EXISTS flights ;
```

```sql
DROP TABLE IF EXISTS airports;
```

Q4 (A) Creating flights table

```sql
CREATE TABLE IF NOT EXISTS flights (
  fid INTEGER NOT NULL AUTO_INCREMENT,
  date DATE NOT NULL,
  origin INTEGER NOT NULL,
  airline TEXT NOT NULL,
  aircraft TEXT NOT NULL,
  altitude INTEGER CHECK (altitude>=0),
  heavy INTEGER NOT NULL,
  PRIMARY KEY (fid)) ;
```

Q4 (B) Creating airports table

```sql
CREATE TABLE IF NOT EXISTS airports (
  aid INTEGER NOT NULL AUTO_INCREMENT,
  airportName TEXT NOT NULL,
  airportState TEXT NOT NULL,
  airportCode TEXT,
  PRIMARY KEY (aid));
```

Q4 (C) Adding foreign key origin to reference airports table

```sql
ALTER TABLE flights ADD FOREIGN KEY (origin) REFERENCES airports(aid);
```

Q4 (D) Creating table conditions

```sql
CREATE TABLE IF NOT EXISTS conditions (
  cid INTEGER NOT NULL AUTO_INCREMENT,
  sky_condition TEXT NOT NULL,
  explanation TEXT,
  PRIMARY KEY (cid));
```

Q4 (E) Creating table strikes

```
CREATE TABLE IF NOT EXISTS strikes (
  sid INTEGER NOT NULL AUTO_INCREMENT,
  fid INTEGER NOT NULL,
  numbirds INTEGER NOT NULL,
  IMPACT TEXT NOT NULL,
  damage INTEGER NOT NULL,
  altitude INTEGER CHECK (altitude>=0),
  conditions INTEGER NOT NULL,
  PRIMARY KEY (sid));
```

Q4 (F) Adding foreign key fid to reference flights table

```
ALTER TABLE strikes ADD FOREIGN KEY (fid) REFERENCES flights(fid);
```

Q4 (F) Adding foreign key conditions to reference conditions table

```
ALTER TABLE strikes ADD FOREIGN KEY (conditions) REFERENCES conditions(cid);
```

Q4 (G) Describing each table

```
DESCRIBE airports;
```

```
DESCRIBE flights;
```

```
DESCRIBE strikes;
```

```
DESCRIBE conditions;
```

Q5 Importing csv data

```
bds.raw <- read.csv("https://s3.us-east-2.amazonaws.com/artificium.us/datasets/BirdStrikesData-V2.csv",
#Checking all columns and it's datatype in the csv data
str(bds.raw)
```

```
## 'data.frame':    25558 obs. of  19 variables:
##  $ rid                  : int  202152 208159 207601 215953 219878 218432 221697 236635 207369 20437
##  $ aircraft             : chr  "Airplane" "Airplane" "Airplane" "Airplane" ...
##  $ airport              : chr  "LAGUARDIA NY" "DALLAS/FORT WORTH INTL ARPT" "LAKEFRONT AIRPORT" "SE
##  $ model                : chr  "B-737-400" "MD-80" "C-500" "B-737-400" ...
##  $ wildlife_struck      : int  859 424 261 806 942 537 227 320 9 4 ...
##  $ impact               : chr  "Engine Shut Down" "None" "None" "Precautionary Landing" ...
```

```
##  $ flight_date           : chr  "11/23/2000 0:00" "7/25/2001 0:00" "9/14/2001 0:00" "9/5/2002 0:00"
##  $ damage                : chr  "Caused damage" "Caused damage" "No damage" "No damage" ...
##  $ airline               : chr  "US AIRWAYS*" "AMERICAN AIRLINES" "BUSINESS" "ALASKA AIRLINES" ...
##  $ origin                : chr  "New York" "Texas" "Louisiana" "Washington" ...
##  $ flight_phase          : chr  "Climb" "Landing Roll" "Approach" "Climb" ...
##  $ remains_collected_flag: logi  FALSE FALSE FALSE TRUE FALSE FALSE ...
##  $ Remarks               : chr  "FLT 753. PILOT REPTD A HUNDRED BIRDS ON UNKN TYPE. #1 ENG WAS SHUT
##  $ wildlife_size         : chr  "Medium" "Small" "Small" "Small" ...
##  $ sky_conditions        : chr  "No Cloud" "Some Cloud" "No Cloud" "Some Cloud" ...
##  $ species               : chr  "Unknown bird - medium" "Rock pigeon" "European starling" "European
##  $ pilot_warned_flag     : chr  "N" "Y" "N" "Y" ...
##  $ altitude_ft           : chr  "1,500" "0" "50" "50" ...
##  $ heavy_flag            : chr  "Yes" "No" "No" "Yes" ...
```

Removing N/A and empty values

```
print(sum(bds.raw$origin == "N/A"))
```

```
## [1] 449
```

```
print(sum(bds.raw$airport == ""))
```

```
## [1] 129
```

Filling N/A and empty values with “unknown” as per question

```
## removing blank and NA values and replacing with "unknown"
bds.raw$airport[bds.raw$airport == ""] <- "unknown"
bds.raw$origin[bds.raw$origin == "N/A"] <- "unknown"
```

Checking if N/A and empty values still persists

```
## checking again to see if blank values are removed
print(sum(bds.raw$origin == "N/A"))
```

```
## [1] 0
```

```
print(sum(bds.raw$airport == ""))
```

```
## [1] 0
```

Creating ids for tables according to their schema

```r
bds.raw <- bds.raw %>%
  mutate(
    aid = as.integer(factor(airport, levels = unique(airport))), #aid - creates a unique ID for airport
    cid = as.integer(factor(sky_conditions, levels = unique(sky_conditions))), #cid - creates a unique
    fid = seq_along(rid) #fid - creates unique ID for each row
  )
```

Creating dataframe airports

```r
#Create the airports table and populate it with unique airport data from the incidents table
airports <- unique(bds.raw[c("aid","airport", "origin")])
# mapping the dataframe airports to the table airports in database
colnames(airports) <- c("aid","airportName", "airportState")
head(airports, 1)
```

```
##   aid  airportName airportState
## 1   1 LAGUARDIA NY     New York
```

Creating dataframe flights

```r
flights <- bds.raw %>% select(fid, flight_date, origin = aid, airline, model, altitude_ft, heavy_flag)
colnames(flights) <- c("fid","date", "origin", "airline", "aircraft", "altitude", "heavy")
flights <- flights %>%
  mutate(
    date = as.Date(date, format = "%m/%d/%Y"),
    altitude = as.integer(gsub(",", "", altitude)),
    heavy = ifelse(heavy == "Yes", 1, 0)         # Converting 'Yes' to 1 and 'No' to 0
  )

#Displays first row of selected rows of dataframe
head(flights,1)
```

```
##   fid       date origin     airline  aircraft altitude heavy
## 1   1 2000-11-23      1 US AIRWAYS* B-737-400     1500     1
```

Creating dataframe conditions

```r
conditions <- bds.raw[c("cid","sky_conditions")]
colnames(conditions) <- c("cid","sky_condition")
conditions <- unique(conditions)
head(conditions,1)
```

```
##   cid sky_condition
## 1   1      No Cloud
```

Creating dataframe strikes

```
strikes <- bds.raw[c("fid","wildlife_struck", "impact", "damage", "altitude_ft", "cid")]
colnames(strikes) <- c("fid","numbirds", "IMPACT", "damage", "altitude", "conditions")
strikes$damage <- recode(strikes$damage,
                         `Caused damage` = 1,
                         `No damage` = 0)
strikes <- strikes %>% mutate(
  damage = as.integer(damage),
  conditions = as.integer(conditions),
  altitude = as.integer(gsub(",", "", altitude)))
head(strikes, 1)
```

```
##   fid numbirds           IMPACT damage altitude conditions
## 1   1      859 Engine Shut Down      1     1500          1
```

Q6 Inserting data (from dataframes) into the sql tables

```
dbWriteTable(con, name = "airports", value = airports, overwrite = FALSE, append = TRUE, row.names = FA
```

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

```
dbWriteTable(con, name = "flights", value = flights, overwrite = FALSE, append = TRUE, row.names = FALSE
```

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

```
dbWriteTable(con, name = "conditions", value = conditions, overwrite = FALSE, append = TRUE, row.names =
```

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

```
dbWriteTable(con, name = "strikes", value = strikes, overwrite = FALSE, append = TRUE, row.names = FALSE
```

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

Q7 Showing glimpses of data in each table

```
SELECT * from airports limit 10;
```

Table 1: Displaying records 1 - 10

aid	airportName	airportState	airportCode
1	LAGUARDIA NY	New York	NA
2	DALLAS/FORT WORTH INTL ARPT	Texas	NA
3	LAKEFRONT AIRPORT	Louisiana	NA

aid	airportName	airportState	airportCode
4	SEATTLE-TACOMA INTL	Washington	NA
5	NORFOLK INTL	Virginia	NA
6	GUAYAQUIL/S BOLIVAR	unknown	NA
7	NEW CASTLE COUNTY	Delaware	NA
8	WASHINGTON DULLES INTL ARPT	DC	NA
9	ATLANTA INTL	Georgia	NA
10	ORLANDO SANFORD INTL AIRPORT	Florida	NA

```
SELECT * FROM flights LIMIT 10;
```

Table 2: Displaying records 1 - 10

fid	date	origin	airline	aircraft	altitude	heavy
1	2000-11-23	1	US AIRWAYS*	B-737-400	1500	1
2	2001-07-25	2	AMERICAN AIRLINES	MD-80	0	0
3	2001-09-14	3	BUSINESS	C-500	50	0
4	2002-09-05	4	ALASKA AIRLINES	B-737-400	50	1
5	2003-06-23	5	COMAIR AIRLINES	CL-RJ100/200	50	0
6	2003-07-24	6	AMERICAN AIRLINES	A-300	0	0
7	2003-08-17	7	BUSINESS	LEARJET-25	150	0
8	2006-03-01	8	UNITED AIRLINES	A-320	100	0
9	2000-01-06	9	AIRTRAN AIRWAYS	DC-9-30	0	0
10	2000-01-07	10	AIRTOURS INTL	A-330	0	0

```
SELECT * FROM conditions LIMIT 10;
```

```
SELECT * FROM strikes LIMIT 10;
```

Q8 Finding the top 10 states with the greatest number of bird strike incidents

```
SELECT
    a.airportState AS state,
    COUNT(s.sid) AS number_of_incidents
FROM
    flights f
JOIN
    airports a ON f.origin = a.aid
JOIN
    strikes s ON f.fid = s.fid
GROUP BY
    a.airportState
ORDER BY
    number_of_incidents DESC
LIMIT 10;
```

Table 3: Displaying records 1 - 10

state	number_of_incidents
California	24990
Texas	24450
Florida	20450
New York	13160
Illinois	10070
Pennsylvania	9850
Missouri	9560
Kentucky	8060
Ohio	7730
Hawaii	7160

Q9 Finding the airlines that had an above average number bird strike incidents

```
WITH AirlineIncidents AS (
    SELECT
        f.airline,
        COUNT(s.sid) AS incidents
    FROM
        flights f
    JOIN
        strikes s ON f.fid = s.fid
    GROUP BY
        f.airline
),

AverageIncidents AS (
    SELECT
        AVG(incidents) AS avg_incidents
    FROM
        AirlineIncidents
)

SELECT
    ai.airline,
    ai.incidents
FROM
    AirlineIncidents ai, AverageIncidents av
WHERE
    ai.incidents > av.avg_incidents
ORDER BY
    ai.incidents DESC;
```

Table 4: Displaying records 1 - 10

airline	incidents
SOUTHWEST AIRLINES	46280
BUSINESS	30740
AMERICAN AIRLINES	20580
DELTA AIR LINES	13490
AMERICAN EAGLE AIRLINES	9320
SKYWEST AIRLINES	8910
US AIRWAYS*	7970
JETBLUE AIRWAYS	7080
UPS AIRLINES	5900
US AIRWAYS	5400

Q10 Finding the (total) number of birds that struck aircraft by month

```
# Execute the SQL query
query <- "
SELECT
    EXTRACT(MONTH FROM f.date) AS month,
    SUM(s.numbirds) AS total_birds_struck
FROM
    flights AS f
JOIN
    strikes AS s ON f.fid = s.fid
GROUP BY
    month
ORDER BY
    total_birds_struck DESC;
"
#Storing the results of the query in a dataframe
monthlyStrikes <- suppressWarnings(dbGetQuery(con, query))
monthlyStrikes$month <- months(as.Date(paste(monthlyStrikes$month, "1", sep="-"), format="%m-%d"))
monthlyStrikes$month[is.na(monthlyStrikes$month)] <- "Unknown"


#Printing only the first 6 months of data from the dataframe
print(monthlyStrikes[1:6,])
```

```
##       month total_birds_struck
## 1    August             110130
## 2      July              93440
## 3 September              92010
## 4   October              72779
## 5  November              63060
## 6      June              52090
```

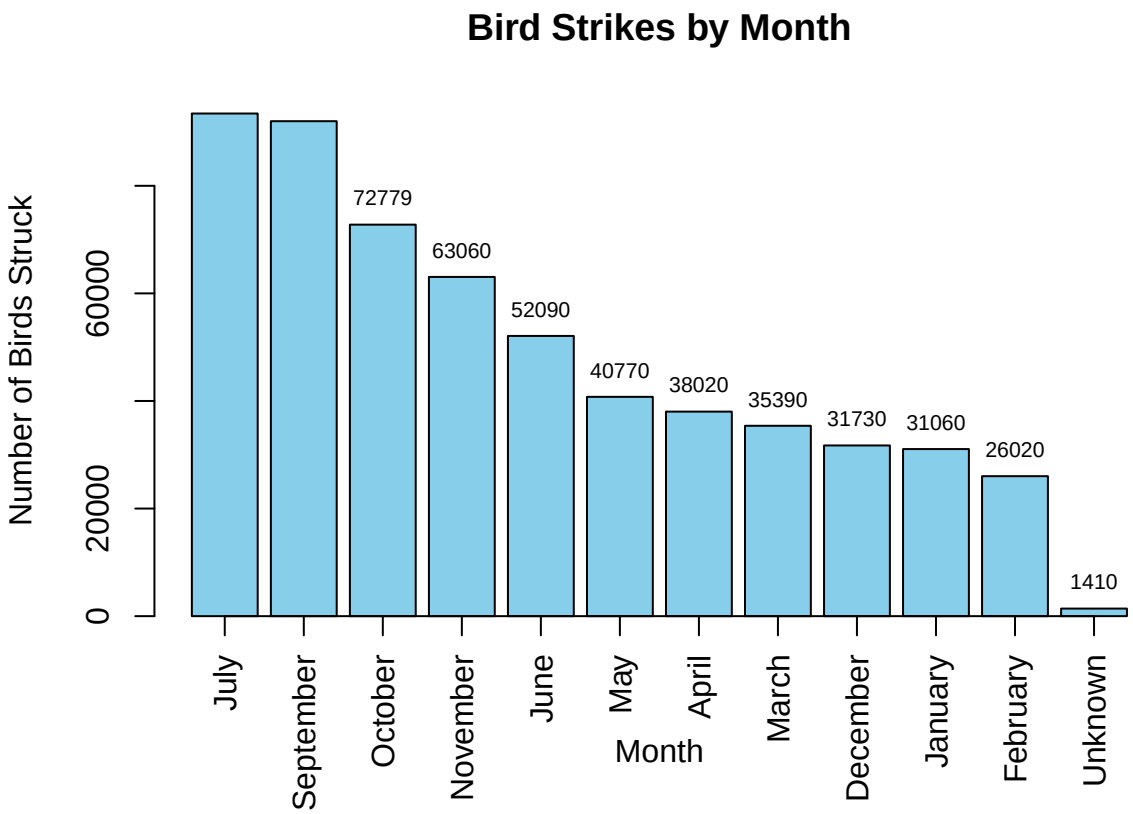
Q11 Building a column chart that plots month along the x-axis versus number of birds on the y-axis

```
# Assuming df is the dataframe you got from the previous question
monthlyStrikes <- monthlyStrikes[-1,]

# Creating a bar plot without x-axis labels
bp <- barplot(monthlyStrikes$total_birds_struck,
              names.arg=NULL, # suppress x-axis labels
              main="Bird Strikes by Month",
              xlab="Month",
              ylab="Number of Birds Struck",
              col="skyblue",
              border="black",
              ylim=c(0, max(monthlyStrikes$total_birds_struck) + 1500)) # +10 for a little extra space

# Adding rotated x-axis labels
axis(side=1, at=bp, labels=monthlyStrikes$month, las=2, srt=30, adj=1)

# Adding data labels
if (length(monthlyStrikes$total_birds_struck) > 1) {
  text(x=bp,
       y=monthlyStrikes$total_birds_struck + 5, # +5 to position the labels just above the bars
       labels=monthlyStrikes$total_birds_struck,
       cex=0.7,
       pos=3)
} else {
  text(x=bp[1],
       y=monthlyStrikes$total_birds_struck + 5,
       labels=monthlyStrikes$total_birds_struck,
       cex=0.7,
       pos=3)
}
```



Q12 Creating a stored procedure in MySQL that adds a new strike to the database

```
CREATE PROCEDURE IF NOT EXISTS AddBirdStrike(
    -- Parameters for Airports
    IN p_airportName TEXT,
    IN p_airportState TEXT,
    IN p_airportCode TEXT,

    -- Parameters for Flights
    IN p_date DATE,
    IN p_airline TEXT,
    IN p_aircraft TEXT,
    IN p_altitude INTEGER,
    IN p_heavy BOOLEAN,

    -- Parameters for Strikes
    IN p_numbirds INTEGER,
    IN p_impact TEXT,
    IN p_damage BOOLEAN,
    IN p_strike_altitude INTEGER,

    -- Parameters for Conditions
    IN p_sky_condition TEXT,
```

```
    IN p_explanation TEXT
)
BEGIN
    DECLARE v_origin INTEGER;
    DECLARE v_fid INTEGER;
    DECLARE v_conditions INTEGER;

    -- Check if airport exists, if not insert it
    SELECT aid INTO v_origin FROM airports WHERE airportCode = p_airportCode;
    IF v_origin IS NULL THEN
        INSERT INTO airports(airportName, airportState, airportCode)
        VALUES (p_airportName, p_airportState, p_airportCode);
        SET v_origin = LAST_INSERT_ID();
    END IF;

    -- Insert flight and get the fid
    INSERT INTO flights(date, origin, airline, aircraft, altitude, heavy)
    VALUES (p_date, v_origin, p_airline, p_aircraft, p_altitude, p_heavy);
    SET v_fid = LAST_INSERT_ID();

    -- Check if conditions exist, if not insert it
    SELECT cid INTO v_conditions FROM conditions WHERE sky_condition = p_sky_condition AND explanation =
    IF v_conditions IS NULL THEN
        INSERT INTO conditions(sky_condition, explanation)
        VALUES (p_sky_condition, p_explanation);
        SET v_conditions = LAST_INSERT_ID();
    END IF;

    -- Insert strike
    INSERT INTO strikes(fid, numbirds, impact, damage, altitude, conditions)
    VALUES (v_fid, p_numbirds, p_impact, p_damage, p_strike_altitude, v_conditions);

END;
```

Q13 Testing the defined procedure by adding new row of data

```
sample_airportName <- "Northeastern Airport"
sample_airportState <- "Northeastern Land"
sample_airportCode <- ""
sample_date <- "2023-10-31"
sample_airline <- "NEU"
sample_aircraft <- "Khoury"
sample_altitude <- 10000
sample_heavy <- TRUE
sample_numbirds <- 1
sample_impact <- "minor"
sample_damage <- FALSE
sample_strike_altitude <- 8000
sample_sky_condition <- "Clear"
sample_explanation <- "Normal Condition"

# Call the stored procedure using sprintf
```

```
query <- sprintf(
    "CALL AddBirdStrike('%s', '%s', '%s', '%s', '%s', '%s', %d, %d, %d, '%s', %d, '%s', '%s', '%s')",
    sample_airportName, sample_airportState, sample_airportCode,
    sample_date, sample_airline, sample_aircraft, sample_altitude, as.integer(sample_heavy),
    sample_numbirds, sample_impact, as.integer(sample_damage), sample_strike_altitude,
    sample_sky_condition, sample_explanation
)

# Execute the stored procedure
dbSendQuery(con, query)
```

```
## <MySQLResult:357392024,0,21>
```

Verifying the insertion by fetching the latest row in strikes table

```
result <- dbGetQuery(con, "SELECT * FROM strikes ORDER BY sid DESC LIMIT 1")
print(result)
```

```
##      sid   fid numbirds IMPACT damage altitude conditions
## 1 327680 25568        1  minor      0     8000          4
```

Verifying the insertion by fetching the latest row in flights table

```
result <- dbGetQuery(con, "SELECT * FROM flights ORDER BY fid DESC LIMIT 1")
print(result)
```

```
##     fid       date origin airline aircraft altitude heavy
## 1 25568 2023-10-31   1111     NEU   Khoury    10000     1
```

Verifying the insertion by fetching the latest row in airports table

```
result <- dbGetQuery(con, "SELECT * FROM airports ORDER BY aid DESC LIMIT 1")
print(result)
```

```
##    aid          airportName      airportState airportCode
## 1 1111 Northeastern Airport Northeastern Land
```

Disconnecting the database

```
dbDisconnect(con)
```

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