Practicum I CS5200

Venkatesan, Jayaraman and Binu, Aby

Summer Full 2023

Connect to Database

using Amazon RDS MySQL as database server for this practicum

```
library(RMySQL)
```

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

```
# Settings
db_user <- 'thisispracticum'
db_password <- 'thisispracticum'
db_name <- 'test3_p'
db_host <- 'practicum1.c0lvnnjxlfni.us-east-1.rds.amazonaws.com'
db_port <- 3306

# 3. Read data from db
mydb <-  dbConnect(MySQL(), user = db_user, password = db_password,
                 dbname = db_name, host = db_host, port = db_port)
```

Create Database

Creating airport table with aid as primary key with auto increment

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

Creating airport table with fid as primary key with auto increment and other columns specified in question. Added extra constraint to altitude to hold only positive values and foreign key reference to above airport table

```
CREATE TABLE IF NOT EXISTS flights (
  fid INTEGER NOT NULL AUTO_INCREMENT,
  date DATE,
  origin INTEGER,
```

```
  airline TEXT,
  aircraft TEXT,
  altitude INTEGER CHECK (altitude >= 0),
  heavy BOOLEAN,
  PRIMARY KEY(fid),
  FOREIGN KEY (origin) REFERENCES airports(aid)
);
```

Creating condition lookup table

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

Creating strikes table with columns specifies in question with sid as synthetic primary key.
conditions is foreign key references to cid in conditions table

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

Adding foreign key reference

Adding fid as foreign key reference to fid in flights table

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

Validating table creations

The below chunks validates whether the expected tables are created

```
tables <- dbListTables(mydb)
expectedTables <- c('airports', 'flights', 'conditions', 'strikes')

# Check if all the expected tables exist
allExist <- all(expectedTables %in% tables)
```

```r
# Check if any unexpected tables exist
unexpectedTables <- setdiff(tables, expectedTables)

cat("All expected tables exist:", allExist, "\n")

if (length(unexpectedTables) > 0) {
  cat("Unexpected tables found:\n")
} else {
  cat("No unexpected tables found.")
}


query <- paste0("DESCRIBE flights")
table_info <- dbGetQuery(mydb, query)

col_names = c ('fid' , 'date','origin','airline','aircraft')

if(all(col_names %in% table_info$Field  )){
  cat("FLIGHTS TABLE - FIELDS ARE PROPER\n")
}

query <- paste0("DESCRIBE airports")
table_info <- dbGetQuery(mydb, query)

col_names = c ('aid' , 'airportState','airportCode')

if(all(col_names %in% table_info$Field  )){
  cat("AIRPORTS TABLE - FIELDS ARE PROPER\n")
}



query <- paste0("DESCRIBE conditions")
table_info <- dbGetQuery(mydb, query)

col_names = c ('cid' , 'sky_condition','explanation')

if(all(col_names %in% table_info$Field  )){
  cat("CONDITIONS TABLE - FIELDS ARE PROPER\n")
}
```

Loads and reads the CSV file.

```r
bds.raw <- read.csv("BirdStrikesData-V2.csv")
```

Cleaning CSV

Cleaning the CSV by adding sentinel airlines, aircrafts for empty values and changing heacy flags to 0/1 for the ease of database insertion

```
bds.raw$flight_date <- as.Date(bds.raw$flight_date, format="%m/%d/%Y")

bds.raw$airline = ifelse(bds.raw$airline=="","sentinel",bds.raw$airline)
bds.raw$aircraft = ifelse(bds.raw$aircraft=="","sentinel",bds.raw$aircraft)

bds.raw$altitude_ft <- as.numeric(gsub(",", "", bds.raw$altitude_ft))


bds.raw$heavy_flag = ifelse(bds.raw$heavy_flag=="No",0,bds.raw$heavy_flag)
bds.raw$heavy_flag = ifelse(bds.raw$heavy_flag=="Yes",1,bds.raw$heavy_flag)
```

Populating tables

Airport table population

Retrieving unique values from origin column and populating the airports table

```
temp1 <- bds.raw[!duplicated(bds.raw[c('origin')]),]

airports <- data.frame(aid = seq(1,length(temp1$origin)),
                           airportCode = "",
                           airportState = temp1$origin)


dbWriteTable(mydb, "airports", airports, append = T,row.names=FALSE)
```

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

Conditions table population

Retrieving unique values from sky_conditions column and populating the conditions table

```
temp2 <- bds.raw[!duplicated(bds.raw[c('sky_conditions')]),]



conditions <- data.frame(cid = seq(1,length(temp2$sky_conditions)),
                           sky_condition = temp2$sky_conditions,
                           explanation = "" )


dbWriteTable(mydb, "conditions", conditions, append = T,row.names=FALSE)
```

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

Flights table population

Retrive all the required columns from bds.raw and rename them to fit the table requirements. Also mapping each rows with approriate foreign key references to airports table

```
# retrive all needed rows
temp3 <- bds.raw[c('flight_date' , 'airline' , 'aircraft' , 'altitude_ft',
                   'heavy_flag' , 'origin')]

#change the names as needed
flights <- data.frame(fid = seq(1,length(temp3$origin)) ,
                      date = temp3$flight_date ,
                      origin = temp3$origin,
                      airline = temp3$airline,
                      aircraft = temp3$aircraft,
                      altitude = temp3$altitude_ft,
                      heavy = temp3$heavy_flag
                      )


rangex = 1:nrow(flights)

# for each row map the foreign key of airports to flight

for (r in rangex) {
  a <- airports$aid[which(airports$airportState == flights$origin[r])]

  flights$origin[r] <-  strtoi(a)

}

#write dataframe to table
dbWriteTable(mydb, "flights", flights, append = T,row.names=FALSE)
```

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

Strikes table population

Retrive all the required columns from bds.raw and rename them to fit the table requirements. Also mapping each rows with approriate foreign key references to flights table, airports table and conditions table

```
# retrive all needed rows
temp4 <- bds.raw[c('impact' , 'damage', 'altitude_ft', 'wildlife_struck' ,
                   'airline' , 'aircraft' , 'heavy_flag' , 'origin',
                   'sky_conditions' , 'flight_date')]



# as said in question preprocess the damage to 0/1 for ease of insertion
temp4$damage = ifelse(temp4$damage=="No damage",0,temp4$damage)
temp4$damage = ifelse(temp4$damage=="Caused damage",1,temp4$damage)


# add exta column for foreign key addition
temp4$fid <- vector("numeric", nrow(temp4))

#set default to 0
```

```
temp4$fid <- 0

strike_range = 1:nrow(temp4)
for (sr in strike_range) {

  # update sky_conditions with foreign key
  c <- conditions$cid[which(conditions$sky_condition == temp4$sky_conditions[sr])]

  temp4$sky_conditions[sr] <-  strtoi(c)

  # for each row map the foreign key of airports to flights
   o <- airports$aid[which(airports$airportState == temp4$origin[sr])]

  temp4$origin[sr] <-  strtoi(o)

  # for each row map the foreign key of flights to strikes

  fid <- -1

   # Perform the comparison without NA to retrieve fid
  if (anyNA(temp4$flight_date[sr] , temp4$altitude_ft )) {

    fid <- flights$fid[which(flights$aircraft == temp4$aircraft[sr] &
                               flights$airline == temp4$airline[sr] &
                               flights$origin == temp4$origin[sr]
                             )]


  } else {
     fid <- flights$fid[which(flights$aircraft == temp4$aircraft[sr] &
                              flights$airline == temp4$airline[sr] &
                              flights$origin == temp4$origin[sr] &

                              flights$date == temp4$flight_date[sr] &

                              flights$altitude == temp4$altitude_ft[sr]
                            )]


  }

  # if there are multiple flight choosing the first one (as pointed in question)
  if (length(fid) > 0) {
  fid <- min(fid)
  }

  temp4$fid[sr] <- strtoi(fid)


}


strikes <- data.frame(sid = seq(1,length(temp4$origin)) ,
```

```
                      fid = temp4$fid ,
                      damage = temp4$damage ,
                      altitude = temp4$altitude_ft,
                      conditions = temp4$sky_conditions,
                      numbirds = temp4$wildlife_struck,
                      impact = temp4$impact
                      )




# populate the strikes table
dbWriteTable(mydb, "strikes", strikes, append = T,row.names=FALSE)
```

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

7. Validating table population

Retrieving first 5 rows of each table to verify the presence of data after population

```
q1_val <- "select * from conditions";
q2_val <- "select * from airports limit 5";
q3_val <- "select * from flights limit 5";
q4_val <- "select * from strikes limit 5 ";

cond_res <- dbGetQuery(mydb, q1_val)
air_res <-  dbGetQuery(mydb, q2_val)
flight_res <- dbGetQuery(mydb, q3_val)
strike_res <- dbGetQuery(mydb, q4_val)

print(cond_res)
```

```
##   cid sky_condition explanation
## 1   1      No Cloud
## 2   2    Some Cloud
## 3   3      Overcast
```

```
print(air_res)
```

```
##   aid airportState airportCode
## 1   1     New York
## 2   2        Texas
## 3   3    Louisiana
## 4   4   Washington
## 5   5     Virginia
```

```
print(flight_res)
```

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

```
## 2   2 2001-07-25      2 AMERICAN AIRLINES Airplane        0     0
## 3   3 2001-09-14      3          BUSINESS Airplane       50     0
## 4   4 2002-09-05      4   ALASKA AIRLINES Airplane       50     1
## 5   5 2003-06-23      5   COMAIR AIRLINES Airplane       50     0
```

```
print(strike_res)
```

```
##   sid fid numbirds                impact damage altitude conditions
## 1   1   1      859      Engine Shut Down      1     1500          1
## 2   2   2      424                  None      1        0          2
## 3   3   3      261                  None      0       50          1
## 4   4   4      806 Precautionary Landing      0       50          2
## 5   5   5      942                  None      0       50          1
```

8. Query : To find the top 10 states with the greatest number of bird strike incidents

Below is the query to find top 10 bird strike incidents grouped by states (airportState in this schema)

```
# considered count(*) here and not sum because each row is an incident
SELECT a.airportState AS STATE, COUNT(*) AS INCIDENTS
FROM strikes AS s
JOIN flights AS f ON s.fid = f.fid
JOIN airports AS a ON f.origin = a.aid
GROUP BY a.airportState
ORDER BY INCIDENTS DESC
LIMIT 10;
```

Table 1: Displaying records 1 - 10

STATE	INCIDENTS
California	2520
Texas	2453
Florida	2055
New York	1319
Illinois	1008
Pennsylvania	986
Missouri	960
Kentucky	812
Ohio	778
Hawaii	729

9. Query: To find the airlines that had an above average number bird strike incidents.

Below is the query to find tabove average number of incidents by joing with flights table and retriving all incident counts above average

```
# considered count(*) here and not sum because each row is an incident
SELECT airline as airline , count(*)
FROM strikes
JOIN flights ON strikes.fid = flights.fid
GROUP BY airline
HAVING COUNT(*) > (
    SELECT AVG(incident_count)
    FROM (
        SELECT COUNT(*) AS incident_count
        FROM strikes
        JOIN flights ON strikes.fid = flights.fid
        GROUP BY airline
    ) AS avg_counts
);
```

Table 2: Displaying records 1 - 10

airline	count(*)
US AIRWAYS*	797
AMERICAN AIRLINES	2058
BUSINESS	3074
ALASKA AIRLINES	304
COMAIR AIRLINES	317
UNITED AIRLINES	506
AIRTRAN AIRWAYS	414
AMERICA WEST AIRLINES	157
HAWAIIAN AIR	332
DELTA AIR LINES	1349

10. Query : To find the (total) number of birds that struck aircraft by month

Create a SQL query against your database to find the (total) number of birds that struck aircraft by month. The data has some strikes for undefined months which will be shown as unknown

```
# considered sum() here and not count because ques ask total bird strike
sql_query <- "SELECT MONTH(f.date) AS month, SUM(s.numbirds) AS total_strikes
FROM strikes AS s
JOIN flights AS f ON s.fid = f.fid
GROUP BY month ORDER BY month";

birdStrikeByMonth <- dbGetQuery(mydb, sql_query)

birdStrikeByMonth$month = ifelse(is.na(birdStrikeByMonth$month),"unknown",
                                 birdStrikeByMonth$month)

head(birdStrikeByMonth , 6)
```

```
##     month total_strikes
## 1 unknown           141
## 2       1          3106
```

```
## 3       2          2602
## 4       3          3539
## 5       4          3802
## 6       5          4077
```

QUES 11 : Below is the plot to visualize bird strike by months

Using month 1 - 12 in x axis representing the months. The data has some strikes for undefined months so left as empty.(hence first column in graph will have empty axis label)

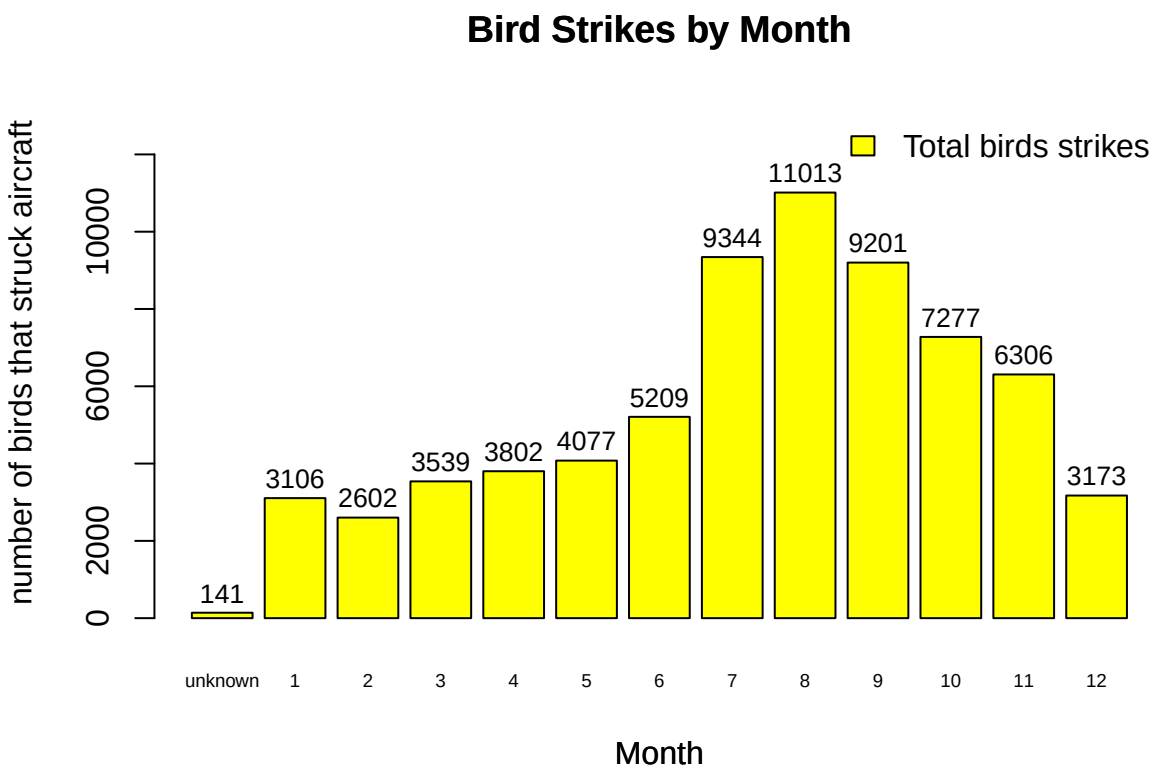
Y axis limit is normalised and increased to maxvalue of strikes * 1.2 to make data lables visible

```
names <- birdStrikeByMonth$month
values <- birdStrikeByMonth$total_strikes

# Create the column chart using barplot
x <- barplot(values, names.arg = names, xlab = "Month",
             main = "Bird Strikes by Month", col = "yellow",
             ylim = c(0, max(values) * 1.2),
             cex.names = 0.6)


title(main = "Bird Strikes by Month")
title(xlab = "Month")
title(ylab = "number of birds that struck aircraft")
legend("topright", legend = "Total birds strikes", fill = "yellow", bty = "n")

text(x , values+500, labels = as.character(values) , cex = 0.8)
```



QUES 12 : Below is the procedure created to insert a bird strike incident

The question is broken into 4 parts,

```
* Check if airport present ; else insert store its id
* Check if sky_condition is new ; if so insert in table and store its value
* Check if flight is new ; if so insert into flights table and store its id
* Insert strike to table with all the ids stored above as foreign key reference
```

```
stored_procedure <- "
CREATE PROCEDURE addNewBirdStrike(
    IN airline_inp VARCHAR(50),
    IN aircraft_inp VARCHAR(50),
    IN sky_cond_inp VARCHAR(50),
    IN origin_input VARCHAR(50),
    IN impact_inp VARCHAR(50),

    IN date_inp DATE,

    IN damage_inp BOOLEAN,
    IN heavy_inp BOOLEAN,

    IN num_of_birds_inp INTEGER,
    IN altitude_inp INTEGER
```

```
    )
BEGIN

  DECLARE conditionCount INT;
  DECLARE flightCount INT;
  DECLARE airportCount INT;

  DECLARE airportID INT;
  DECLARE conditionID INT;
  DECLARE flightsID INT;

  SET airportCount = 0;

  SELECT COUNT(*) INTO airportCount FROM airports WHERE
  airportState = origin_input;

  IF airportCount = 0 THEN
       insert into airports (airportState , airportCode)
       values (origin_input , '');
  end if;

  select aid into airportID from airports where airportState = origin_input;


  SET conditionCount = 0;
  SELECT COUNT(*) INTO conditionCount FROM conditions WHERE
  sky_condition = sky_cond_inp;

  IF conditionCount = 0 THEN
   insert into conditions (sky_condition , explanation) values (sky_cond_inp,'');
  END IF;

  select cid into  conditionID from conditions where sky_condition = sky_cond_inp;

  SET flightCount = 0;
  SELECT count(*) into flightCount from flights WHERE (airline=airline_inp
                                            AND aircraft = aircraft_inp
                                            AND date = date_inp
                                            AND altitude = altitude_inp
                                            AND heavy = heavy_inp
                                       AND origin = airportID) ;



  IF flightCount = 0 THEN
          insert into flights (airline , aircraft, altitude,heavy,date,origin)
          values ( airline_inp, aircraft_inp, altitude_inp, heavy_inp, date_inp,
          airportID);
  END IF;

  select fid into flightsID from flights where (airline=airline_inp
                                            AND aircraft = aircraft_inp
                                            AND date = date_inp
```

```
                                              AND altitude = altitude_inp
                                              AND heavy = heavy_inp
                                          AND origin = airportID);

  insert into strikes (fid, numbirds, impact, damage, altitude, conditions)
  values (flightsID,num_of_birds_inp,impact_inp,damage_inp,
                    altitude_inp,conditionID);
END;"

dbExecute(mydb,stored_procedure)
```

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

Testing new stored procedures

Here we test by inserting a completely new airport,condition,flight and strike

```
call_proc = "CALL addNewBirdStrike(
    'LUFTANSA',
    'NOT AIRPLANE',
    'dusty',
    'KARNATAKA',
    'more heavy',
    '1947-08-15',
    '1',
    '1',
    '101',
    '69'

    );"

res = dbSendQuery(mydb,call_proc)

q1 <- "select * from conditions where sky_condition = 'dusty'"
q2 <- "select * from flights where aircraft = 'NOT AIRPLANE'"
q3 <- "select * from airports where airportState = 'KARNATAKA'"
q4 <- "select * from strikes where (altitude = 69 and impact='more heavy')"

q1_res <- dbGetQuery(mydb, q1)
q2_res <- dbGetQuery(mydb, q2)
q3_res <- dbGetQuery(mydb, q3)
q4_res <- dbGetQuery(mydb, q4)


if(q1_res$sky_condition != 'dusty' ) {
  cat("ERROR ON CONDITION TABLE POPULATION!\n")
} else {
  cat("CONDITION TABLE POPULATED SUCCESSFULLY\n")
}
```

```
## CONDITION TABLE POPULATED SUCCESSFULLY
```

```r
if(q2_res$airline != 'LUFTANSA' &&  q2_res$origin != q3_res$aid) {
  cat("ERROR ON FLIGHTS TABLE POPULATION!\n")
} else {
  cat("FLIGHTS TABLE POPULATED SUCCESSFULLY\n")
}
```

```
## FLIGHTS TABLE POPULATED SUCCESSFULLY
```

```r
if(q2_res$origin != q3_res$aid && q3_res$airportState!= 'KARNATAKA' ) {
  cat("ERROR ON AIRPORT TABLE POPULATION!\n")
} else {
  cat("AIRPORT TABLE POPULATED SUCCESSFULLY\n")
}
```

```
## AIRPORT TABLE POPULATED SUCCESSFULLY
```

```r
if(q4_res$conditions != q1_res$cid && q4_res$fid != q2_res$fid ) {
  cat("ERROR ON STRIKES TABLE POPULATION!\n")
} else {
  cat("STRIKES TABLE POPULATED SUCCESSFULLY\n")
}
```

```
## STRIKES TABLE POPULATED SUCCESSFULLY
```