Analysis of Wildlife Strikes to Aircraft

Practicum I CS5200

Bhavitha Naga Sai Kandru

Spring 2024

Q3) Add an R code chunk that connects to your MySQL database. Use headers for all other questions with appropriate titles so you (and we) can navigate the notebook more easily. If you have difficulty connecting to or setting up MySQL, then use SQLite and proceed. You can always come back to this question and change your configuration so that you connect to MySQL. This is the benefit of relational databases: you can easily switch between databases without changing your code. Do not echo the code in the notebook and supress any warnings or other messages. Add an appropriate code chunk label.

Connecting to database

```
# 1. Library
library(RMariaDB)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# 2. Settings freemysqlhosting.net (max 5MB)
db_name <- "sql5690911"
db_user <- "sql5690911"
db_host <- "sql5.freemysqlhosting.net"
db_pwd <- "6vXhYueYTL"
db_port <- 3306

# 3. Connect to remote server database
con <-  dbConnect(RMariaDB::MariaDB(), user = db_user, password = db_pwd,
                      dbname = db_name, host = db_host, port = db_port)
```

Q4) In a single R code chunk that is not echoed (i.e., set "echo = F" for the code chunk and supress all messages), create the database schema described below (do not use {sql} code chunks). Add appropriate constraints, primary key and foreign key definitions. In the schema definitions below, primary keys are underlined and foreign keys are bolded.

Q4(A): # Create Database # Create airports table

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

Q4(B-C) # Creating flights table and linking

```
CREATE TABLE IF NOT EXISTS flights (
    fid integer PRIMARY KEY,
    `date` date,
    origin integer,
    airline text,
    aircraft text,
    altitude integer CHECK (altitude >= 0),
    heavy bit(1),
    FOREIGN KEY (origin) REFERENCES airports(aid)
) ENGINE=InnoDB;
```

Q4(D) # Creating conditions table

```
CREATE TABLE IF NOT EXISTS conditions (
    cid INTEGER PRIMARY KEY,
    sky_condition TEXT,
    explanation TEXT
) ENGINE=InnoDB;
```

Q4(E-F) # Creating strikes table and linking

```
CREATE TABLE IF NOT EXISTS strikes (
    sid INTEGER AUTO_INCREMENT PRIMARY KEY,
    fid INTEGER,
    numbirds INTEGER,
    impact TEXT,
    damage TINYINT(1),
    altitude INTEGER,
    conditions INTEGER
) ENGINE=InnoDB;
```

Q4(G)

Test code-1

```
SHOW TABLES;
```

Test code-2

```
DESC flights;
```

Q5

Reading data from csv into bds.raw dataframe

```
bds.raw <- read.csv("BirdStrikesData-V3.csv")
head(bds.raw)
```

```
##      rid aircraft                    airport        model
## 1 202152 Airplane                LAGUARDIA NY    B-737-400
## 2 208159 Airplane DALLAS/FORT WORTH INTL ARPT        MD-80
## 3 207601 Airplane           LAKEFRONT AIRPORT        C-500
## 4 215953 Airplane         SEATTLE-TACOMA INTL    B-737-400
## 5 219878 Airplane                NORFOLK INTL CL-RJ100/200
## 6 218432 Airplane         GUAYAQUIL/S BOLIVAR        A-300
##                  impact     flight_date    damage           airline     origin
## 1      Engine Shut Down 11/23/2000 0:00    Damage        US AIRWAYS   New York
## 2                  None  7/25/2001 0:00    Damage AMERICAN AIRLINES      Texas
## 3                  None  9/14/2001 0:00 No damage          BUSINESS  Louisiana
## 4 Precautionary Landing   9/5/2002 0:00 No damage   ALASKA AIRLINES Washington
## 5                  None  6/23/2003 0:00 No damage   COMAIR AIRLINES   Virginia
## 6                  None  7/24/2003 0:00 No damage AMERICAN AIRLINES        N/A
##   flight_phase wildlife_size sky_conditions pilot_warned_flag altitude_ft
## 1        Climb        Medium       No Cloud                 N       1,500
## 2 Landing Roll         Small     Some Cloud                 Y           0
## 3     Approach         Small       No Cloud                 N          50
## 4        Climb         Small     Some Cloud                 Y          50
## 5     Approach         Small       No Cloud                 N          50
## 6 Take-off run         Small       No Cloud                 N           0
##   heavy_flag
## 1        Yes
## 2         No
## 3         No
## 4        Yes
## 5         No
## 6         No
```

Q6

Storing neccessary columns into dataframe

```
df.bird <- data.frame(
  rid = bds.raw$rid,
  date = bds.raw$flight_date,
  airline = bds.raw$airline,
  aircraft = bds.raw$aircraft,
  heavy = bds.raw$heavy_flag,
  impact = bds.raw$impact,
  airportState = bds.raw$origin,
  sky_condition = bds.raw$sky_conditions,
  altitude = bds.raw$altitude_ft,
  damage = bds.raw$damage,
  numbirds = bds.raw$wildlife_size
)
```

Modifications

```
df.bird$airline[which(df.bird$airline =='')] <- 'sentinel'


# Updating the impact as TRUE or FALSE based on damage caused or not for the impact field.
df.bird$damage <- ifelse(df.bird$damage == 'Caused damage', 'TRUE', 'FALSE')

df.bird$heavy <- ifelse(df.bird$heavy == 'Yes', 'TRUE', 'FALSE')

df.bird$date[which(df.bird$date =='')] <- 'NoDate'
```

Creating dataframes

```
df.airports <- data.frame(
  aid = seq_len(length(unique(df.bird$airportState))),
  airportName = unique(df.bird$airport),
  airportState = unique(df.bird$airportState),
  airportCode = "ZZZ",
  stringsAsFactors = FALSE
)

df.flights <- data.frame(
  fid = (df.bird$rid),
  date = as.Date(df.bird$date, format = "%m/%d/%Y"),
  origin = df.bird$airportState)
df.flights <- df.flights %>%
              left_join(df.airports %>%
                              select(aid,airportState), by = c('origin' ='airportState')) %>%
                              mutate(origin = ifelse(!is.na(aid),aid, origin)) %>%
                              select(-aid)
```

```
 df.flights$airline = df.bird$airline
  df.flights$aircraft = df.bird$aircraft
  df.flights$altitude = df.bird$altitude
 df.flights$heavy = as.logical(df.bird$heavy)
  df.flights$stringsAsFactors = FALSE



df.conditions <- data.frame(
  cid = seq_len(length(unique(df.bird$sky_condition))),
  sky_condition = unique(df.bird$sky_condition),
  explanation = "",
  stringsAsFactors = FALSE
)

df.strikes <- data.frame(
  sid = seq_len(nrow(df.bird)),
  fid = df.flights$fid,
  numbirds = df.bird$numbirds,
  impact = df.bird$impact,
  damage = as.logical(df.bird$damage),
  altitude = df.bird$altitude,
  conditions = df.bird$sky_condition)

  df.strikes <- df.strikes %>%
                left_join(df.conditions %>%
                              select(cid,sky_condition), by = c('conditions' ='sky_condition')) %>%
                              mutate(conditions = ifelse(!is.na(cid),cid, sky_condition)) %>%
                              select(-cid)

  df.strikes$stringsAsFactors = FALSE

  df.flights <- head(df.flights, 500)
  df.strikes <- head(df.strikes, 500)
```

Pushing data into tables

```
# Drop child tables
dbRemoveTable(con, "strikes")
dbRemoveTable(con, "conditions")
dbRemoveTable(con, "flights")


# Insert data into the airports table
dbWriteTable(con, "airports", df.airports, overwrite = TRUE, row.names = FALSE)

# Insert data into the flights table
dbWriteTable(con, "flights", df.flights, overwrite = TRUE, row.names = FALSE)

# Insert data into the conditions table
```

```r
dbWriteTable(con, "conditions", df.conditions, overwrite = TRUE, row.names = FALSE)

# Insert data into the strikes table
dbWriteTable(con, "strikes", df.strikes, overwrite = TRUE, row.names = FALSE)
```

Q7

Displaying data from 'airport' table

```r
dbGetQuery(con, "SELECT * from airports Limit 10")
```

```
##    aid airportName airportState airportCode
## 1    1    New York     New York         ZZZ
## 2    2       Texas        Texas         ZZZ
## 3    3   Louisiana    Louisiana         ZZZ
## 4    4  Washington   Washington         ZZZ
## 5    5    Virginia     Virginia         ZZZ
## 6    6         N/A          N/A         ZZZ
## 7    7    Delaware     Delaware         ZZZ
## 8    8          DC           DC         ZZZ
## 9    9     Georgia      Georgia         ZZZ
## 10  10     Florida      Florida         ZZZ
```

Displaying data from 'conditions' table

```r
dbGetQuery(con, "SELECT * from conditions Limit 10")
```

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

Displaying data from 'flights' table

```r
dbGetQuery(con, "SELECT * from flights Limit 10")
```

```
##       fid       date origin           airline aircraft altitude heavy
## 1  202152 2000-11-23      1        US AIRWAYS Airplane    1,500     1
## 2  208159 2001-07-25      2 AMERICAN AIRLINES Airplane        0     0
## 3  207601 2001-09-14      3          BUSINESS Airplane       50     0
## 4  215953 2002-09-05      4   ALASKA AIRLINES Airplane       50     1
```

```
## 5  219878 2003-06-23      5   COMAIR AIRLINES Airplane       50     0
## 6  218432 2003-07-24      6 AMERICAN AIRLINES Airplane        0     0
## 7  221697 2003-08-17      7          BUSINESS Airplane      150     0
## 8  236635 2006-03-01      8   UNITED AIRLINES Airplane      100     0
## 9  207369 2000-01-06      9   AIRTRAN AIRWAYS Airplane        0     0
## 10 204371 2000-01-07     10     AIRTOURS INTL Airplane        0     0
##    stringsAsFactors
## 1                 0
## 2                 0
## 3                 0
## 4                 0
## 5                 0
## 6                 0
## 7                 0
## 8                 0
## 9                 0
## 10                0
```

Displaying data from 'strikes' table

```
dbGetQuery(con, "SELECT * from strikes Limit 10")
```

```
##    sid    fid numbirds                impact damage altitude conditions
## 1    1 202152   Medium      Engine Shut Down      0    1,500          1
## 2    2 208159    Small                  None      0        0          2
## 3    3 207601    Small                  None      0       50          1
## 4    4 215953    Small Precautionary Landing      0       50          2
## 5    5 219878    Small                  None      0       50          1
## 6    6 218432    Small                  None      0        0          1
## 7    7 221697    Small                 Other      0      150          1
## 8    8 236635    Small                 Other      0      100          2
## 9    9 207369    Small      Aborted Take-off      0        0          2
## 10  10 204371    Small                  None      0        0          2
##    stringsAsFactors
## 1                 0
## 2                 0
## 3                 0
## 4                 0
## 5                 0
## 6                 0
## 7                 0
## 8                 0
## 9                 0
## 10                0
```

}}

Q8

Top Airports with Strikes

```
SELECT airportState AS State, COUNT(*) AS Incidents
FROM strikes
JOIN flights ON strikes.fid = flights.fid
JOIN airports ON flights.origin = airports.aid
GROUP BY airportState
ORDER BY Incidents DESC
LIMIT 10;
```

Table 1: Displaying records 1 - 10

State	Incidents
Texas	49
California	39
Florida	29
New York	26
Pennsylvania	23
Kentucky	18
Illinois	18
DC	16
Ohio	16
Massachusetts	15

Q9

Analysis by Airline

```
SELECT airline AS Airline, COUNT(*) AS Incidents
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
```

Table 2: Displaying records 1 - 10

Airline	Incidents
ALASKA AIRLINES	9
AMERICAN AIRLINES	37
AMERICAN EAGLE AIRLINES	23
ATLANTIC COAST AIRLINES	8
BUSINESS	74
COMAIR AIRLINES	15
CONTINENTAL AIRLINES	12
DELTA AIR LINES	32

Airline	Incidents
EXPRESSJET (CONTINENTAL EXPRS)	8
HAWAIIAN AIR	12

Q10

Analysis by Month

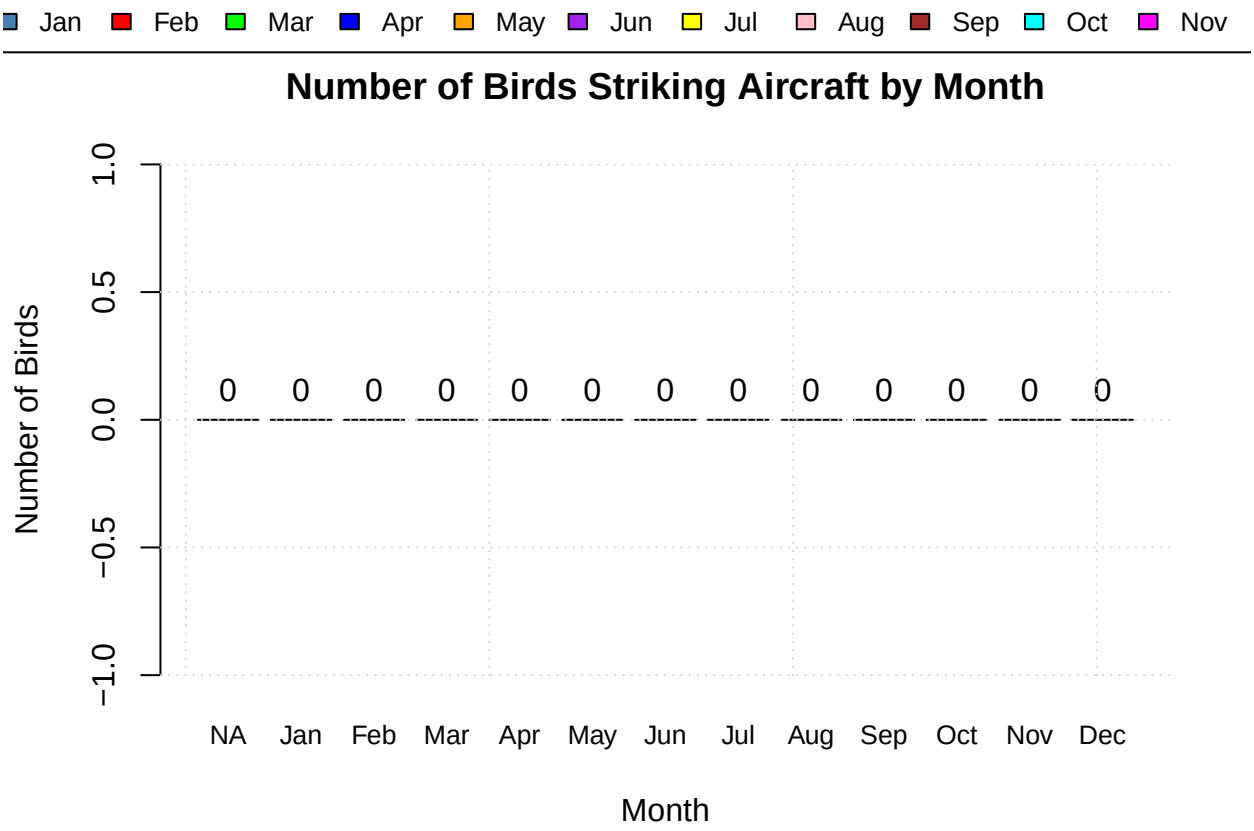
```
# Execute the SQL query
result <- dbGetQuery(con, "SELECT MONTH(date) AS Month, SUM(numbirds) AS TotalBirds FROM strikes JOIN fl

# Display 10 rows of the dataframe
head(result, 10)
```

```
##    Month TotalBirds
## 1     NA          0
## 2      1          0
## 3      2          0
## 4      3          0
## 5      4          0
## 6      5          0
## 7      6          0
## 8      7          0
## 9      8          0
## 10     9          0
```

Q11

Trend by Month

```
# Plot the column chart with increased y-axis scale
colors <- c("steelblue", "red", "green", "blue", "orange", "purple", "yellow", "pink", "brown", "cyan",

month_labels <- c("NA", month.abb)

# Add data labels
text(x = barplot(result$TotalBirds, names.arg = month_labels, col = colors, xlab = "Month", ylab = "Numb

grid()

# Add title
title(main = "Number of Birds Striking Aircraft by Month")

# Create legend
legend("top", legend = month.abb, fill = colors, title = "Month", horiz = TRUE, xpd = TRUE, inset = c(0
```



Q12

Creating new stored procedure called add_strike

```
DROP PROCEDURE IF EXISTS add_strike;
```

```
create procedure add_strike (
      in rid int,
      in `date` date,
      in airline varchar(50),
      in aircraft varchar(30),
      in altitude_ft int,
      in heavy_flag boolean,
      in wildlife_struck int,
      in impact varchar(30),
      in damage varchar(30),
      in skyCondition varchar(30),
      in explanations varchar(30),
      in airport_state varchar(30),
      in airport_code varchar(30)
```

```
)

begin

SET @next_sid := (SELECT MAX(sid) + 1 FROM strikes);
SET @next_aid := (SELECT MAX(aid) + 1 FROM airports);
SET @next_cid := (SELECT MAX(cid) + 1 FROM conditions);

if airport_state not in (select airportState from airports) then
insert into airports (aid, airportState, airportCode) values (@next_aid, airport_state, airport_code);
end if;

if skyCondition not in (select sky_condition from conditions) then
insert into conditions (cid, sky_condition, explanation) values (@next_cid, skyCondition, explanations)
end if;

insert into flights (fid, `date`, origin, airline, aircraft, altitude, heavy) values (rid, `date`, (sel

insert into strikes (sid, fid, numbirds, impact, damage, altitude, conditions)
values (@next_sid,rid, wildlife_struck, impact, damage, altitude_ft, (select cid from conditions as c wh


end
```

Stored Procedure Testing1 where input is of type that does not change airports and conditions tables

```
CALL add_strike (95000, "2023-04-01", "Southwest Airlines", "Airplane", 500, 1, 20, "Engine Shut Down",
```

```
select * from airports
```

Table 3: Displaying records 1 - 10

aid	airportName	airportState	airportCode
1	New York	New York	ZZZ
2	Texas	Texas	ZZZ
3	Louisiana	Louisiana	ZZZ
4	Washington	Washington	ZZZ
5	Virginia	Virginia	ZZZ
6	N/A	N/A	ZZZ
7	Delaware	Delaware	ZZZ
8	DC	DC	ZZZ
9	Georgia	Georgia	ZZZ
10	Florida	Florida	ZZZ

```
select * from flights where fid = 95000
```

Table 4: 1 records

fid	date	origin	airline	aircraft	altitude	heavy	stringsAsFactors
95000	2023-04-01	5	Southwest Airlines	Airplane	500	1	NA

```
select * from conditions
```

Table 5: 3 records

cid	sky_condition	explanation
1	No Cloud	
2	Some Cloud	
3	Overcast	

```
select * from strikes where fid = 95000
```

Table 6: 1 records

sid	fid	numbirds	impact	damage	altitude	conditions	stringsAsFactors
501	95000	20	Engine Shut Down	1	500	3	NA

Stored Procedure Testing 2 where input is of type that updates airports and conditions table as well

```
CALL add_strike (8765, "2024-10-12", "MB Airlines", "Airplane", 5650, 1, 27, "Precautionary Landing", FA
```

```
select * from airports
```

Table 7: Displaying records 1 - 10

aid	airportName	airportState	airportCode
1	New York	New York	ZZZ
2	Texas	Texas	ZZZ
3	Louisiana	Louisiana	ZZZ
4	Washington	Washington	ZZZ
5	Virginia	Virginia	ZZZ
6	N/A	N/A	ZZZ
7	Delaware	Delaware	ZZZ
8	DC	DC	ZZZ
9	Georgia	Georgia	ZZZ
10	Florida	Florida	ZZZ

```
select * from flights where fid = 95001
```

Table 8: 0 records

fid	date	origin	airline	aircraft	altitude	heavy	stringsAsFactors

```
select * from conditions
```

Table 9: 4 records

cid	sky_condition	explanation
1	No Cloud	
2	Some Cloud	
3	Overcast	
4	Snow	

```
select * from strikes where fid = 95001
```

Table 10: 0 records

sid	fid	numbirds	impact	damage	altitude	conditions	stringsAsFactors

Deleting the added entries

```
DELETE FROM flights WHERE fid = 95000;
```

```
DELETE FROM strikes WHERE fid = 95000;
```

```
DELETE FROM flights WHERE fid = 95001;
```

```
DELETE FROM strikes WHERE fid = 95001;
```

```
DELETE FROM airports where airportState = 'Vizag';
```

```
DELETE FROM conditions where sky_condition = 'Snow';
```

Dropping the Stored method

```
DROP PROCEDURE IF EXISTS add_strike;
```

Drop strikes table if exists

```
drop table if exists strikes;
```

Drop conditions table if exists

```
drop table if exists conditions;
```

Drop flights table if exists before

```
drop table if exists flights;
```

Drop airports table if exists

```
drop table if exists airports;
```

Disconnecting the Database

```
dbDisconnect(con)
```