Practicum I CS5200

Agnibha Chatterjee [chatterjee.ag@northeastern.edu]
Om Agarwal [agarwal.o@northeastern.edu]

Fall 2023

Some considerations made

1. All cells with missing values of N/A's have been converted to Unknown
2. Missing altitudes have been given the value 999999
3. Missing dates have been given the value 01/01/1970

```
# Clearing the environment (R environment)
rm(list = ls())
```

Connect to Database

```
library(RMySQL)
```

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

```
library(DBI)

db.user <- 'admin'
db.password <- 'omagarwal86'
db.name <- 'dbbirdstrikes'
db.host <- 'dbms-practicum-1.ckaxw7lvptoz.us-east-2.rds.amazonaws.com'
db.port <- 3306

dbConn <-  dbConnect(RMySQL::MySQL(), user = db.user, password = db.password,
                           dbname = db.name, host = db.host, port = db.port)
```

```
SET FOREIGN_KEY_CHECKS=0;
```

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

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

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

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

```
SET FOREIGN_KEY_CHECKS=1;
```

Create Database

Question 4.A

```
-- This SQL chunk creates table flights
CREATE TABLE flights(
```

```
fid INTEGER PRIMARY KEY,
date DATE DEFAULT NULL DEFAULT '1970-01-01',
origin INTEGER NOT NULL DEFAULT 999,
airline TEXT NOT NULL ,
aircraft TEXT NOT NULL ,
altitude INTEGER NOT NULL DEFAULT 999999,
heavy BOOLEAN NOT NULL DEFAULT FALSE,
CHECK (altitude >= 0)
);
```

Question 4.B

```
-- This SQL chunk creates table airports
CREATE TABLE airports(
aid INTEGER AUTO_INCREMENT PRIMARY KEY,
airportName TEXT NOT NULL ,
airportState TEXT NOT NULL ,
airportCode TEXT DEFAULT ''
);
```

Question 4.C

```
-- This SQL chunk add a foreign key constraint on flights
ALTER TABLE flights
ADD CONSTRAINT fk_origin
FOREIGN KEY (origin) REFERENCES airports(aid)
ON DELETE CASCADE
ON UPDATE CASCADE;
```

Question 4.D

```
-- This SQL chunk creates table conditions
CREATE TABLE conditions(
cid INTEGER AUTO_INCREMENT PRIMARY KEY,
sky_condition TEXT NOT NULL,
explanation TEXT DEFAULT ''
);
```

Question 4.E

```
-- This SQL chunk creates table strikes
CREATE TABLE strikes(
sid INTEGER AUTO_INCREMENT PRIMARY KEY,
fid INTEGER NOT NULL,
numbirds INTEGER NOT NULL,
impact TEXT NOT NULL ,
damage TEXT NOT NULL,
altitude INTEGER NOT NULL DEFAULT 999999,
conditions INTEGER,
CHECK (altitude >= 0),
CONSTRAINT fk_conditions
FOREIGN KEY (conditions) REFERENCES conditions(cid)
```

```
ON DELETE CASCADE
ON UPDATE CASCADE
);
```

Question 4.F

```
-- This SQL chunk add a foreign key constraint on strikes
ALTER TABLE strikes
ADD CONSTRAINT fk_fid
FOREIGN KEY (fid) REFERENCES flights(fid)
ON DELETE CASCADE
ON UPDATE CASCADE;
```

Question 4.G

```
INSERT INTO airports VALUES(101, "Temp Airport", "Temp State", '');
```

```
INSERT INTO flights VALUES(102, "1980-01-03", 101, "Temp Airline",
  "Temp Aircraft", 5050, FALSE);
```

```
INSERT INTO conditions VALUES(301, "Sunny", '');
```

```
  INSERT INTO strikes VALUES(401, 102, 500, "None", "No Damage", 5050, 301);
```

Question 5 and 6 (We have read the CSV file from its URL)

```
# This r chunk reads the csv and populates the tables

# Reading the csv file froms its url
bds.raw <- read.csv(
  url("https://s3.us-east-2.amazonaws.com/artificium.us/datasets/BirdStrikesData-V2.csv"))
bds.airports <- bds.raw[, c("airport", "origin")]
# Replacing all N/A and '' with Unknown
bds.airports[] <- lapply(bds.airports,
                         function(col) ifelse(col %in% c('N/A', ''), 'Unknown', col))

  # Table Airports
  # Grouping airports by their state
  grouped.bds.airports <- lapply(split(bds.airports, bds.airports$origin), unique)
  # Merging each group
  airports <- do.call(rbind, grouped.bds.airports)
  row.names(airports) <- NULL
  colnames(airports) <- c("airportName", "airportState")
  n.airports <- nrow(airports)
  airports$aid <- 1:n.airports


   # Table Flights
  bds.flights <- bds.raw[, c("rid", "flight_date", "airport", "origin", "airline",
                             "aircraft", "altitude_ft", "heavy_flag")]
  bds.flights$origin <- bds.airports$origin
  bds.flights$airport <- bds.airports$airport
  colnames(bds.flights) <- c("fid", "date", "airport", "origin", "airline",
```

```
                              "aircraft", "altitude", "heavy")
# Replacing all empty values with Unknown
bds.flights[] <- lapply(bds.flights, function(col) ifelse(col == '', 'Unknown', col))
n.flights <- nrow(bds.flights)

# Function to format dates
clean_date <- function(date) {
  if (date == 'Unknown') {
    return("1970-01-01")
  } else {
    split.timestamp <- strsplit(date, " ")
    split.date <- unlist(strsplit(split.timestamp[[1]][1], '/'))
    split.date[1:2] <- sprintf("%02d", as.integer(split.date[1:2]))
    return(paste0(split.date[3], "-", split.date[1], "-", split.date[2]))
  }
}

for (i in 1:n.flights) {
# Setting origin in flights from the airports dataframe
bds.flights$origin[i] <- which(airports$airportName == bds.flights$airport[i] &
                                 airports$airportState == bds.flights$origin[i])

bds.flights$date[i] <- clean_date(bds.flights$date[i])

# Replacing * in airline name (UNITED AIRWAYS* -> UNITED AIRWAYS)
bds.flights$airline[i] <- gsub("*", "", bds.flights$airline[i])
if (bds.flights$altitude[i] == 'Unknown') {
  # Setting Unknoun altitudes to a value of 999999
  bds.flights$altitude[i] <- 999999
} else {
  # Removing comma from flights
  bds.flights$altitude[i] <- as.integer(gsub(",", "", bds.flights$altitude[i]))
}
bds.flights$heavy[i] <- !grepl("Unknown|No", bds.flights$heavy[i])

}
bds.flights$airport <- NULL
bds.flights$fid <- as.integer(bds.flights$fid)
bds.flights$origin <- as.integer(bds.flights$origin)
bds.flights$altitude <- as.integer(bds.flights$altitude)


# Table Conditions
bds.conditions <- bds.raw[, c("sky_conditions", "Remarks")]
# Removing duplicate values from the bds.conditions dataframe
bds.conditions <- bds.conditions[!duplicated(bds.conditions$sky_conditions), ]
bds.conditions$cid <- 1:nrow(bds.conditions)
bds.conditions$Remarks <- NULL
# renaming columns
colnames(bds.conditions) <- c("sky_condition", "cid")

# Table Strikes
bds.strikes <- bds.raw[, c("wildlife_struck", "impact", "damage", "sky_conditions")]
```

```
  bds.strikes$fid <- bds.flights$fid
  bds.strikes$altitude <- bds.flights$altitude
  # renaming columns
  colnames(bds.strikes) <- c("numbirds", "impact", "damage", "conditions", "fid",
                             "altitude")
  bds.strikes[] <- lapply(bds.strikes,
                          function(col) ifelse(col == '', 'Unknown', col))

  for (i in 1:nrow(bds.strikes)) {
    # Setting conditions to its correct cid value
    bds.strikes$conditions[i] <- which(
      bds.conditions$sky_condition == bds.strikes$conditions[i])
  }
  bds.strikes$conditions <- as.integer(bds.strikes$conditions)
  bds.strikes$numbirds <- as.integer(bds.strikes$numbirds)
  bds.strikes$sid <- 1:nrow(bds.strikes)

  # Writing to table
  dbWriteTable(dbConn, "airports", airports, append = T, row.names = F)
```

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

```
  dbWriteTable(dbConn, "flights", bds.flights, append = T, row.names = F)
```

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

```
  dbWriteTable(dbConn, "conditions", bds.conditions, append = T, row.names = F)
```

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

```
  dbWriteTable(dbConn, "strikes", bds.strikes, append = T, row.names = F)
```

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

Question 7

```
SELECT * FROM flights LIMIT 5;
```

Table 1: 5 records

fid	date	origin	airline	aircraft	altitude	heavy
1195	2002-11-13	412	MILITARY	Airplane	2000	0
3019	2002-10-10	150	MILITARY	Airplane	400	0
3500	2001-05-15	412	MILITARY	Airplane	1000	0
3504	2001-05-23	412	MILITARY	Airplane	1800	0
3597	2001-04-18	852	MILITARY	Airplane	200	0

```
SELECT * FROM airports LIMIT 5;
```

Table 2: 5 records

aid	airportName	airportState	airportCode
1	BIRMINGHAM-SHUTTLESWORTH INTL	Alabama	NA
2	TROY MUNICIPAL ARPT	Alabama	NA

aid	airportName	airportState	airportCode
3	HUNTSVILLE INTL	Alabama	NA
4	MONTGOMERY REGIONAL ARPT	Alabama	NA
5	MOBILE DOWNTOWN ARPT	Alabama	NA

```
-- Displaying everything as there are only 3 rows
SELECT * FROM conditions;
```

Table 3: 3 records

cid	sky_condition	explanation
1	No Cloud	NA
2	Some Cloud	NA
3	Overcast	NA

```
SELECT * FROM strikes LIMIT 5;
```

Table 4: 5 records

sid	fid	numbirds	impact	damage	altitude	conditions
1	202152	859	Engine Shut Down	Caused damage	1500	1
2	208159	424	None	Caused damage	0	2
3	207601	261	None	No damage	50	1
4	215953	806	Precautionary Landing	No damage	50	2
5	219878	942	None	No damage	50	1

Question 8

```
# The idea in this query was to JOIN strikes, flights and airport on their respective
# primary keys
SELECT a.airportState, COUNT(*) as no_of_impacts
FROM strikes s
JOIN flights f ON f.fid = s.fid
JOIN airports a on a.aid = f.origin
GROUP BY a.airportState
ORDER BY COUNT(*) DESC
LIMIT 10;
```

Table 5: Displaying records 1 - 10

airportState	no_of_impacts
California	2520
Texas	2453
Florida	2055
New York	1319
Illinois	1008
Pennsylvania	986
Missouri	960
Kentucky	812

airportState	no_of_impacts
Ohio	778
Hawaii	729

Question 9

```
# The idea was to find the count of each airline, give the count column an alias and find
# the average of this count column, this would then be used to find airline having a count
# greater than the average count
SELECT airline, COUNT(f.fid) AS num_bird_strikes
FROM flights f
JOIN strikes s ON s.fid = f.fid
GROUP BY f.airline
HAVING COUNT(*) >
            (SELECT AVG(count)
              FROM (SELECT COUNT(*) AS count
              FROM strikes st
              JOIN flights fg ON fg.fid = st.fid
              GROUP BY fg.airline) AS avg)
ORDER BY num_bird_strikes DESC;
```

Table 6: Displaying records 1 - 10

airline	num_bird_strikes
SOUTHWEST AIRLINES	4628
BUSINESS	3074
AMERICAN AIRLINES	2058
DELTA AIR LINES	1349
AMERICAN EAGLE AIRLINES	932
SKYWEST AIRLINES	891
US AIRWAYS*	797
JETBLUE AIRWAYS	708
UPS AIRLINES	590
US AIRWAYS	540

Question 10

```
# The idea here was to join strikes and flights on sid and then perform the grouping
# As the missing dates have the value 1970-01-01, we've excluded them from the result
# of this query
query <- "SELECT MONTHNAME(f.date) AS month_name, SUM(s.numbirds) AS cnt
FROM flights f
JOIN strikes s ON s.fid = f.fid AND YEAR(f.date) > 1970
GROUP BY month_name
ORDER BY cnt DESC;"

total.per.month <- dbGetQuery(conn = dbConn, statement = query)
```

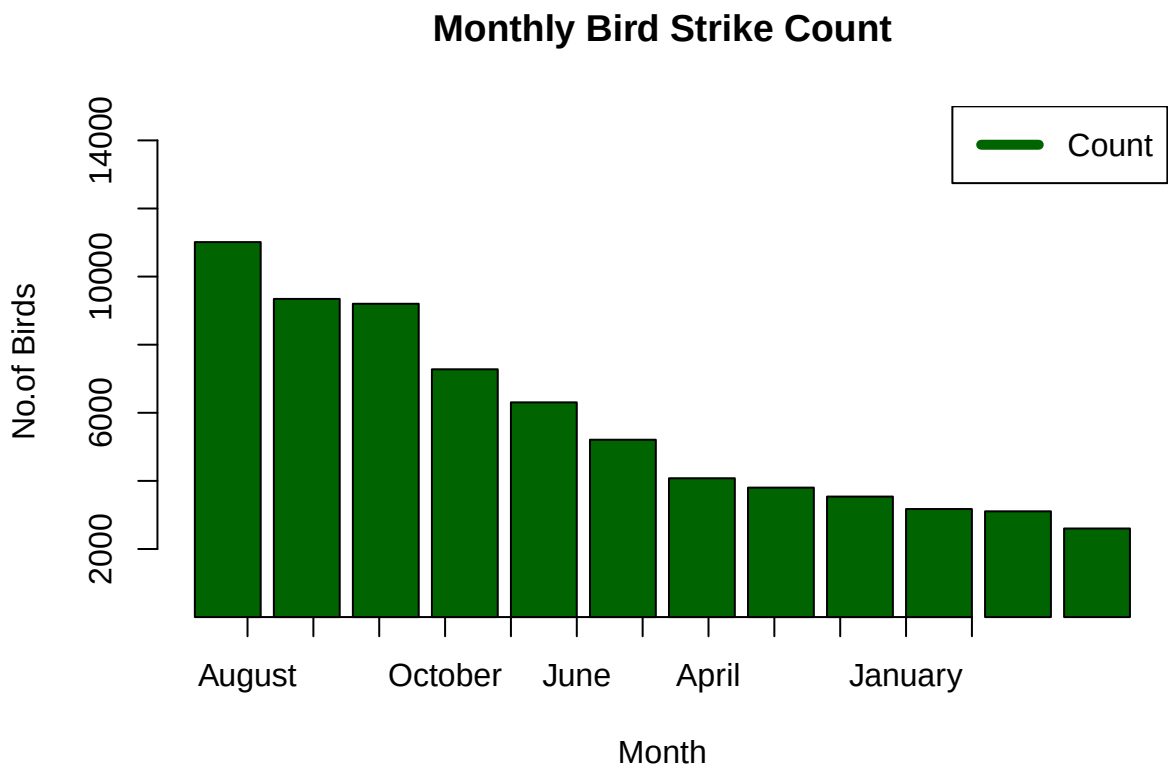
```
## Warning in .local(conn, statement, ...): Decimal MySQL column 1 imported as
## numeric
```

```r
head(total.per.month, n = 6)
```

```
##   month_name   cnt
## 1     August 11013
## 2       July  9344
## 3  September  9201
## 4    October  7277
## 5   November  6306
## 6       June  5209
```

Question 11

```r
total.per.month <- na.omit(total.per.month)
barplot(total.per.month$cnt, main = "Monthly Bird Strike Count", xlab = "Month",
        ylab = "No.of Birds", col = "darkgreen", ylim = c(1, 15000))
legend("topright", legend = "Count", col = "darkgreen", lwd = 5)
axis(1, at = 1:12, labels = total.per.month$month_name)
```



Question 12

Dropping procedure before creating it

```r
DROP PROCEDURE IF EXISTS CreateStrike;
```

```r
CREATE PROCEDURE CreateStrike(
    IN incidentDate DATE,
```

```
    IN airport TEXT,
    IN state TEXT,
    IN i_airline TEXT,
    IN i_aircraft TEXT,
    IN i_altitude INT,
    IN isHeavy BOOLEAN,
    IN numBirds INT,
    IN i_impact TEXT,
    IN i_damage TEXT,
    IN incidentCondition TEXT
)
BEGIN

-- Basic idea
-- 1. Retrieve aid, fid and cid
-- 2. Check if there is an existing strike
-- 3. If yes, issue warning and exit
-- 4. If no, then create the strike, and/or flight, airport, condition

    DECLARE m_origin INT;
    DECLARE m_fid INT;
    DECLARE m_cid INT;
    DECLARE m_sid INT;
    DECLARE err BOOLEAN;

    SET err = FALSE;

    SELECT aid INTO m_origin
    FROM airports
    WHERE airportName = airport AND airportState = state
    LIMIT 1;

    SELECT fid INTO m_fid
    FROM flights f
    WHERE f.date = incidentDate
        AND LOWER(f.airline) = LOWER(i_airline)
        AND LOWER(f.aircraft) = LOWER(i_aircraft)
        AND f.altitude = i_altitude
        AND f.HEAVY = isHeavy
    LIMIT 1;

    SELECT cid INTO m_cid
    FROM conditions
    WHERE LOWER(sky_condition) = LOWER(incidentCondition)
    LIMIT 1;

    SELECT sid INTO m_sid
    FROM strikes
    WHERE fid = m_fid
      AND numbirds = numBirds
      AND LOWER(impact) = LOWER(i_impact)
      AND LOWER(damage) = LOWER(i_damage)
      AND altitude = i_altitude
```

```
      AND conditions = m_cid
    LIMIT 1;

    IF m_sid IS NOT NULL THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Denied insertion of duplicate strike';
        SET err = TRUE;
    END IF;

    IF err = FALSE THEN
      IF m_origin IS NULL THEN
        INSERT INTO airports (airportName, airportState) VALUES (airport, state);
        SET m_origin = LAST_INSERT_ID();
    END IF;

    IF m_fid IS NULL THEN
        SELECT MAX(fid) INTO m_fid FROM flights;
        SET m_fid = m_fid + 1;
        INSERT INTO flights (fid, date, origin, airline, aircraft, altitude, HEAVY)
        VALUES (m_fid, incidentDate, m_origin, i_airline, i_aircraft, i_altitude, isHeavy);
    END IF;

    IF m_cid IS NULL THEN
        INSERT INTO conditions (sky_condition) VALUES (incidentCondition);
        SET m_cid = LAST_INSERT_ID();
    END IF;

     INSERT INTO strikes (fid, numbirds, impact, damage, altitude, conditions)
     VALUES (m_fid, numBirds, i_impact, i_damage, i_altitude, m_cid);
    END IF;

END;
```

```
-- Testing CreateStrike
CALL CreateStrike('2012-03-02','Kempegowda Airport', 'Bangalore', 'UNITED AIRWAYS',
'Airplane', 6970, FALSE, 600, 'None', 'No Damage', 'Rainy');
```

```
# This R chunk always displays the sid of the last inserted strike
# If you have inserted a strike, run this to get its sid
sql <- "SELECT MAX(sid) as id FROM strikes;"
res <- dbGetQuery(dbConn, sql)
print(paste("The id of the the last inserted strike is", res$id[[1]][1]))
```

```
## [1] "The id of the the last inserted strike is 25559"
```

Disconnecting from database

```
# Disconnecting from the database
status <- dbDisconnect(dbConn)
if (status) {
  print("Disconnected!")
} else {
  print("Couldn't disconnect!")
}
```

```
## [1] "Disconnected!"
```