Practicum I - Bird Strike Database

Connor Clancy - clancy.co@northeastern.edu

Spring 2023

Libary Imports

```
library(RMySQL)
```

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

```
library(sqldf)
```

```
## Loading required package: gsubfn

## Loading required package: proto

## Loading required package: RSQLite

##
## Attaching package: 'RSQLite'

## The following object is masked from 'package:RMySQL':
##
##     isIdCurrent

## sqldf will default to using MySQL
```

```
options(sqldf.driver = "SQLite")
```

Connect to Database

```
db_user <- 'admin'
db_password <- 'Northea$tern23'
db_name <- 'practicumOne'
db_host <- 'cclancy-cs5200.cbowkysg1oyc.us-east-2.rds.amazonaws.com'
db_port <- 3306

dbcon <- dbConnect(MySQL(), user = db_user, password = db_password,
                   dbname = db_name, host = db_host, port = db_port)
```

Create Helper R Functions

```
#' Method to check the existence of a foreign key and delete it if it exists.
#' If a foreign key already exists in a database this can cause problems for
#' dropping or altering the database.
#'
#' @param fk_name the name of the foreign key to be checked
#' @return void
#'
```

```
delete_fk <- function(fk_name) {
  # Query the database for the foreign key
  iac <- dbGetQuery(dbcon,
    sprintf(
      "SELECT TRUE
      FROM information_schema.table_constraints
      WHERE constraint_name = '%s';", fk_name
    )
  )

  # If the foreign key exist, delete it
  if (nrow(iac) > 0) {
    dbExecute(dbcon,
      sprintf(
        "ALTER TABLE incidents DROP FOREIGN KEY %s;", fk_name
      )
    )
  }
}

#' Method to take in a date string formatted m/d/yyyy and return it as a
#' properly formatted R date.
#'
#' @param ds the date string to be transformed
#' @return date formatted version of the input string.
#'
format_date <- function(ds) {
  td <- strsplit(ds, "/")[[1]]
  return(paste(td[3],td[1],td[2], sep="/"))
}
```

Create Database (Task 4)

4A - Incidents

```
DROP TABLE IF EXISTS incidents;
```

```
CREATE TABLE incidents (
  rid INT AUTO_INCREMENT PRIMARY KEY,
  `dep.date` DATE,
  origin INT,
  airline INT,
  aircraft VARCHAR(255) NOT NULL,
  `flight.phase` ENUM('Climb', 'Landing Roll', 'Approach', 'Take-off run',
                      'Descent', 'Taxi', 'Parked'),
  altitude INT,
  conditions INT,
  warned TINYINT,
  CHECK(altitude>=0)
);
```

4B - Airports

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

```
CREATE TABLE airports(
  aid INT AUTO_INCREMENT PRIMARY KEY,
  airportName VARCHAR(255) NOT NULL,
  airportCode VARCHAR(3),
  state VARCHAR(255)
);
```

4C - Foreign Key Incident Airports

```
delete_fk('FK_IncidentAirports')
```

```
ALTER TABLE incidents
ADD CONSTRAINT FK_IncidentAirports
FOREIGN KEY (origin) REFERENCES airports(aid);
```

4D - Conditions

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

```
CREATE TABLE conditions(
  cid INT AUTO_INCREMENT PRIMARY KEY,
  `condition` VARCHAR(255) NOT NULL,
  explanation VARCHAR(255)
);
```

```
delete_fk('FK_IncidentConditions')
```

```
ALTER TABLE incidents
ADD CONSTRAINT FK_IncidentConditions
FOREIGN KEY (conditions) REFERENCES conditions(cid);
```

4E - Airlines

```
DROP TABLE IF EXISTS airlines;
```

```
CREATE TABLE airlines(
  eid INT AUTO_INCREMENT PRIMARY KEY,
  airlineName VARCHAR(255) NOT NULL,
  airlineCode VARCHAR(2),
  flag VARCHAR(255)
);
```

4F - Join Incidents Airlines

```
delete_fk('FK_IncidentAirline')
```

```
ALTER TABLE incidents
ADD CONSTRAINT FK_IncidentAirline
FOREIGN KEY (airline) REFERENCES airlines(eid);
```

4G - Test Code to ensure the tables are properly set up

Test Inserts

```
INSERT INTO airlines (airlineName, airlineCode) VALUES
  ('American Airlines', 'AA'),
  ('Delta', 'DL'),
  ('jetBlue', 'B6'),
  ('Southwest', 'WN'),
  ('United', 'UA');
```

```
SELECT
  *
FROM airlines;
```

Airline Table Tests

```
INSERT INTO conditions (`condition`, explanation) VALUES
  ('clear', 'clear skies'),
  ('overcast', 'light clounds');
```

```
SELECT
  *
FROM conditions;
```

Conditions Table Tests

```
INSERT INTO airports (airportName, airportCode, state) VALUES
  ('Boston Logan International Airport', 'BOS', 'MA'),
  ('Seattle-Tacoma International Airport', 'SEA', 'WA'),
  ('Vancouver International Airport', 'YVR', 'BC');
```

```
SELECT
  *
FROM airports;
```

Airport Table Tests

```
INSERT INTO incidents (`dep.date`, origin, airline, aircraft, `flight.phase`,
                      altitude, conditions, warned) VALUES
  ('2023-02-24', 1, 3, 'A321-200', 'takeoff', 32, 2, 1),
  ('2023-02-26', 2, 1, '737 MAX 9', 'landing', 189, 1, 0);
```

```
SELECT
  *
FROM incidents;
```

Incidents Table Tests

```
DELETE FROM incidents WHERE 1=1;
```

```
DELETE FROM airlines WHERE 1=1;
```

```
DELETE FROM conditions WHERE 1=1;
```

```
DELETE FROM airports WHERE 1=1;
```

Clean-Up Test Data

Load Data (Task 5)

```
bds.raw <- read.csv("BirdStrikesData-V2.csv", header = TRUE)
head(bds.raw)
```

```
##      rid aircraft                     airport        model wildlife_struck
## 1 202152 Airplane                LAGUARDIA NY    B-737-400             859
## 2 208159 Airplane DALLAS/FORT WORTH INTL ARPT        MD-80             424
## 3 207601 Airplane            LAKEFRONT AIRPORT        C-500             261
## 4 215953 Airplane          SEATTLE-TACOMA INTL    B-737-400             806
## 5 219878 Airplane                NORFOLK INTL CL-RJ100/200             942
## 6 218432 Airplane          GUAYAQUIL/S BOLIVAR        A-300             537
##                  impact     flight_date         damage           airline
## 1      Engine Shut Down 11/23/2000 0:00 Caused damage       US AIRWAYS*
## 2                  None  7/25/2001 0:00 Caused damage AMERICAN AIRLINES
## 3                  None  9/14/2001 0:00     No damage          BUSINESS
## 4 Precautionary Landing   9/5/2002 0:00     No damage   ALASKA AIRLINES
## 5                  None  6/23/2003 0:00     No damage   COMAIR AIRLINES
## 6                  None  7/24/2003 0:00     No damage AMERICAN AIRLINES
##       origin flight_phase remains_collected_flag
## 1   New York        Climb                  FALSE
## 2      Texas Landing Roll                  FALSE
## 3  Louisiana     Approach                  FALSE
## 4 Washington        Climb                   TRUE
## 5   Virginia     Approach                  FALSE
## 6        N/A Take-off run                  FALSE
##
## 1  FLT 753. PILOT REPTD A HUNDRED BIRDS ON UNKN TYPE. #1 ENG WAS SHUT DOWN AND DIVERTED TO EWR. SLIG
## 2
## 3
## 4 NOTAM WARNING. 26 BIRDS HIT THE A/C, FORCING AN EMERGENCY LDG. 77 BIRDS WERE FOUND DEAD ON RWY/TWY
## 5
## 6
##   wildlife_size sky_conditions               species pilot_warned_flag
## 1        Medium       No Cloud Unknown bird - medium                 N
## 2         Small     Some Cloud           Rock pigeon                 Y
## 3         Small       No Cloud     European starling                 N
## 4         Small     Some Cloud     European starling                 Y
## 5         Small       No Cloud     European starling                 N
## 6         Small       No Cloud  Unknown bird - small                 N
##   altitude_ft heavy_flag
## 1       1,500        Yes
## 2           0         No
## 3          50         No
## 4          50        Yes
## 5          50         No
## 6           0         No
```

Populate Tables (Task 6)

Airlines Data Population

```
# TRIM() function included to remove a typo in the data.
bds.airlines <- sqldf("
                      SELECT DISTINCT
                        0 AS eid,
                        trim(airline, '*') AS airlineName,
                        NULL AS airlineCode,
                        NULL AS flag
                      FROM `bds.raw`
                      WHERE TRIM(airline, ' ') != ''
                      ORDER BY trim(airline, '*')
                      ")

# Create the artificial primary key with a counter
n.airlines <- nrow(bds.airlines)
bds.airlines[,1] <- seq(1, n.airlines)

# Display the data frame to make sure it looks correct
head(bds.airlines)
```

```
##   eid                 airlineName airlineCode flag
## 1   1 ABSA AEROLINHAS BRASILEIRAS          NA   NA
## 2   2                     ABX AIR          NA   NA
## 3   3                ACM AVIATION          NA   NA
## 4   4          ADI SHUTTLE GROUP          NA   NA
## 5   5                  AER LINGUS          NA   NA
## 6   6                    AERO AIR          NA   NA
```

```
dbWriteTable(
  dbcon,
  "airlines",
  bds.airlines,
  overwrite = F,
  append = T,
  row.names = FALSE
)
```

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

Airports Data Population

```
# Retrieve the relevant data from the raw dataframe.
bds.airports <- sqldf("
                      SELECT DISTINCT
                        0 AS aid,
                        airport AS airportName,
                        NULL AS airportCode,
                        CASE
                          WHEN origin = 'N/A' THEN NULL
                          ELSE origin
                        END AS state
                      FROM `bds.raw`
                      WHERE TRIM(airport, ' ') != ''
                      ORDER BY airport
                      ")

# Create the artificial primary key with a counter
n.airports <- nrow(bds.airports)
bds.airports[,1] <- seq(1, n.airports)

# Display the data frame to make sure it looks correct
head(bds.airports)
```

```
##   aid                  airportName airportCode        state
## 1   1         ABERDEEN REGIONAL AR          NA South Dakota
## 2   2         ABILENE REGIONAL ARPT          NA        Texas
## 3   3 ABRAHAM LINCOLN CAPITAL ARPT          NA     Illinois
## 4   4   ADAMS COUNTY- LEGION FIELD          NA    Wisconsin
## 5   5              ADAMS FIELD ARPT          NA     Arkansas
## 6   6          ADDINGTON FIELD ARPT          NA     Kentucky
```

```
dbWriteTable(
  dbcon,
  "airports",
  bds.airports,
  overwrite = F,
  append = T,
  row.names = FALSE
)
```

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

Condition Data Population

```
# Retrieve the relevant data from the raw data frame.
bds.conditions <- sqldf("
                      SELECT DISTINCT
                        0 AS cid,
                        sky_conditions AS condition,
                        NULL AS explanation
                      FROM `bds.raw`
                      ORDER BY sky_conditions
                      ")

# Create the artificial primary key with a counter
n.conditions <- nrow(bds.conditions)
bds.conditions[,1] <- seq(1, n.conditions)

# Display the data frame to make sure it looks correct
```

```r
head(bds.conditions)
```

```
##   cid  condition explanation
## 1   1   No Cloud          NA
## 2   2   Overcast          NA
## 3   3 Some Cloud          NA
```

```r
dbWriteTable(
  dbcon,
  "conditions",
  bds.conditions,
  overwrite = F,
  append = T,
  row.names = FALSE
)
```

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

Incidents Data Population

```r
# Retrieve the relevant data from the raw data frame.
bds.incidents <- sqldf("
                      SELECT
                        rid AS rid,
                        REPLACE(flight_date, ' 0:00', '') AS `dep.date`,
                        airport AS origin,
                        trim(airline, '*') AS airline,
                        model AS aircraft,
                        flight_phase AS `flight.phase`,
                        CAST(REPLACE(altitude_ft, ',','') AS INT) AS altitude,
                        sky_conditions AS conditions,
                        CASE
                          WHEN pilot_warned_flag = 'Y' THEN 1
                          ELSE 0
                        END AS warned
                      FROM `bds.raw`
                      ")

# Create the artificial primary key with a counter
n.incidents <- nrow(bds.incidents)
# bds.incidents[,1] <- seq(1, n.incidents)

# Format date column
for (r in 1:n.incidents) {
  bds.incidents$dep.date[r] = format_date(bds.incidents$dep.date[r])
}

# Link `airline` foreign key
for (r in 1:n.incidents) {
  eid <- bds.airlines$eid[
    which(bds.airlines$airlineName == bds.incidents$airline[r])
  ]

  # incidents with a blank airline get a NULL value
  if(length(eid) != 0) {
    bds.incidents$airline[r] <- eid
  } else {
    bds.incidents$airline[r] <- NA
  }
}

# Link `airport` foreign key
for (r in 1:n.incidents) {
  aid <- bds.airports$aid[
    which(bds.airports$airportName == bds.incidents$origin[r])
  ]

  # incidents with a blank airports get a NULL value
  if(length(aid) != 0) {
    bds.incidents$origin[r] <- aid
  } else {
    bds.incidents$origin[r] <- NA
  }
}

# Link `condition` foreign key
for (r in 1:n.incidents) {
  cid <- bds.conditions$cid[
    which(bds.conditions$condition == bds.incidents$conditions[r])
  ]

  # incidents with a blank condition get a NULL value
  if(length(cid) != 0) {
    bds.incidents$conditions[r] <- cid
  } else {
    bds.incidents$conditions[r] <- NA
  }
}

# Display the data frame to make sure it looks correct
head(bds.incidents)
```

```
##      rid   dep.date origin airline     aircraft flight.phase altitude
## 1 202152 2000/11/23    531     274    B-737-400        Climb     1500
## 2 208159  2001/7/25    209      46        MD-80 Landing Roll        0
## 3 207601  2001/9/14    538      70        C-500     Approach       50
## 4 215953   2002/9/5    912      36    B-737-400        Climb       50
## 5 219878  2003/6/23    715     101 CL-RJ100/200     Approach       50
## 6 218432  2003/7/24    396      46        A-300 Take-off run        0
##   conditions warned
## 1          1      0
## 2          3      1
## 3          1      0
## 4          3      1
## 5          1      0
## 6          1      0
```

```
dbWriteTable(
  dbcon,
  "incidents",
  bds.incidents,
  overwrite = F,
  append = T,
  row.names = FALSE
)
```

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

Show Table Heads (Task 7)

Airlines

```
SELECT
  *
FROM airlines
LIMIT 5;
```

Table 1: 5 records

eid	airlineName	airlineCode	flag
1	ABSA AEROLINHAS BRASILEIRAS	NA	NA
2	ABX AIR	NA	NA
3	ACM AVIATION	NA	NA
4	ADI SHUTTLE GROUP	NA	NA
5	AER LINGUS	NA	NA

Airports

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

Table 2: 5 records

aid	airportName	airportCode	state
1	ABERDEEN REGIONAL AR	NA	South Dakota
2	ABILENE REGIONAL ARPT	NA	Texas
3	ABRAHAM LINCOLN CAPITAL ARPT	NA	Illinois
4	ADAMS COUNTY- LEGION FIELD	NA	Wisconsin
5	ADAMS FIELD ARPT	NA	Arkansas

Conditions

```
SELECT
  *
FROM conditions
LIMIT 5;
```

Table 3: 3 records

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

Incidents

```sql
SELECT
  *
FROM incidents
LIMIT 5;
```

Table 4: 5 records

rid	dep.date	origin	airline	aircraft	flight.phase	altitude	conditions	warned
1195	2002-11-13	57	197	B-52H	Approach	2000	2	1
3019	2002-10-10	264	197	T-38A	Climb	400	1	1
3500	2001-05-15	57	197	B-52H	Approach	1000	1	1
3504	2001-05-23	57	197	B-52H	Approach	1800	1	1
3597	2001-04-18	920	197	AT-38B	Approach	200	3	1

States with the most bird strikes (Task 8)

```sql
SELECT
  a.state,
  COUNT(i.rid) AS strikeCount
FROM incidents AS i
INNER JOIN airports AS a
  ON i.origin = a.aid
GROUP BY
  a.state
ORDER BY
  COUNT(rid) DESC
LIMIT 10
```

Table 5: Displaying records 1 - 10

state	strikeCount
California	2499
Texas	2445
Florida	2045
New York	1316
Illinois	1007
Pennsylvania	985
Missouri	956
Kentucky	806
Ohio	773
Hawaii	716

Airlines with above average bird strikes (Task 9)

Since we need the airline and the number of associated strikes twice (once to calculate the average strikes per airline and a second time to display the above average strike airlines) it makes sense to use a Common Table Expression to create sub-queries and temporarily store the data to get the final answer.

```
WITH
  cteE AS (
    SELECT
      e.airlineName,
      COUNT(rid) AS strikes
    FROM incidents AS i
    INNER JOIN airlines AS e
      ON i.airline = e.eid
    GROUP BY
      e.airlineName
  ),
  cteAvg AS (
    SELECT AVG(strikes) AS avg_strikes FROM cteE
  )

SELECT
  e.airlineName,
  e.strikes
FROM cteE AS e
WHERE e.strikes > (SELECT avg_strikes FROM cteAvg)
ORDER BY strikes ASC
```

Table 6: Displaying records 1 - 10

airlineName	strikes
MILITARY	89
COMMUTAIR	92
AIR CANADA	95
ALLEGIANT AIR	107
PIEDMONT AIRLINES	113
ATLANTIC COAST AIRLINES	115
REPUBLIC AIRLINES	120
ISLAND AIR	122
ALOHA AIRLINES	135
SPIRIT AIRLINES	140

Birdstrikes by month and fligth phase (Task 10)

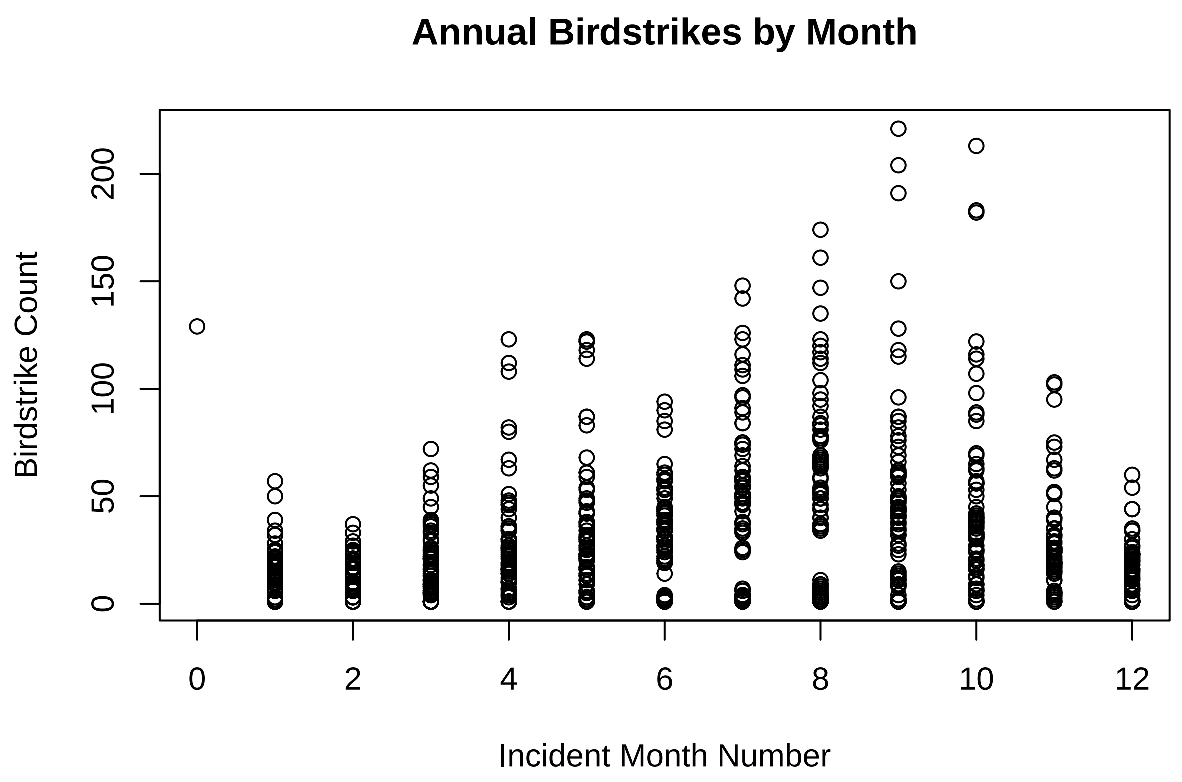
```
df.byMonth <- dbGetQuery(dbcon,
  "SELECT
    MONTH(`dep.date`) AS incident_month,
    YEAR(`dep.date`) AS incident_month,
    `flight.phase` AS flight_phase,
    COUNT(rid) AS incident_count
  FROM incidents
  GROUP BY
    MONTH(`dep.date`),
    YEAR(`dep.date`),
    `flight.phase`
  ")

head(df.byMonth)
```

```
##   incident_month incident_month flight_phase incident_count
## 1             11           2002     Approach             51
## 2             10           2002        Climb             32
## 3              5           2001     Approach             47
## 4              4           2001     Approach             48
## 5              4           2000     Approach             47
## 6              7           2002 Take-off run             51
```

Birdstrike Scatter Plot (Task 11)

```
plot(
  x = df.byMonth$incident_month,
  y = df.byMonth$incident_count,
  main = "Annual Birdstrikes by Month",
  xlab = "Incident Month Number",
  ylab = "Birdstrike Count"
)
```



Stored Procedure for adding an incident to the database (Task 12)

```
DROP PROCEDURE IF EXISTS spAddIncident;
```

```
# Define the signature of the procedure
CREATE PROCEDURE spAddIncident(
    IN dep_date DATE,
    IN airport_name VARCHAR(255),
    IN airline_name VARCHAR(255),
    IN aircraft_type VARCHAR(255),
    IN flight_phase VARCHAR(255),
    IN altitude_ft INT,
    IN condition_type VARCHAR(255),
    IN warned_flag TINYINT
)

BEGIN

    # Declare variables for foreign keys
    DECLARE airportId INT DEFAULT 0;
  DECLARE airlineId INT DEFAULT 0;
  DECLARE conditionsId INT DEFAULT 0;

  # Get airport primary key, or create a new airport if it does not exist
  SELECT aid INTO airportId FROM airports WHERE airportName = airport_name;
  IF airportId = 0 THEN
    INSERT INTO airports (airportName) VALUES (airport_name);
    SELECT aid INTO airportId FROM airports WHERE airportName = airport_name;
  END IF;

  # Get airline primary key, or create a new airline if it does not exist
  SELECT eid INTO airlineId FROM airlines WHERE airlineName = airline_name;
    IF airlineId = 0 THEN
        INSERT INTO airlines (airlineName) VALUES (airline_name);
        SELECT eid INTO airlineId FROM airlines WHERE airlineName = airline_name;
    END IF;

  # Get conditions primary key
  SELECT cid
  INTO conditionsId
  FROM conditions
  WHERE `condition` = condition_type;

  # Add new incident to the database
  INSERT INTO incidents (
    `dep.date`,
    origin,
    airline, aircraft,
    `flight.phase`,
    altitude,
    conditions,
    warned
  )
  VALUES (
    dep_date,
    airportId,
    airlineId,
    aircraft_type,
    flight_phase,
    altitude_ft,
    conditionsId,
    warned_flag
  );

END
```

```
CALL spAddIncident("2023-01-01", "Northeastern", "Avelo", "Connor 737 MAX 8",
                  "Taxi", 10, "Overcast", 0);
```

```
SELECT * FROM incidents WHERE `dep.date` = "2023-01-01"
```

Table 7: 1 records

rid	dep.date	origin	airline	aircraft	flight.phase	altitude	conditions	warned
321910	2023-01-01	1110	292	Connor 737 MAX 8	Taxi	10	2	0