CS5200 - Practicum I

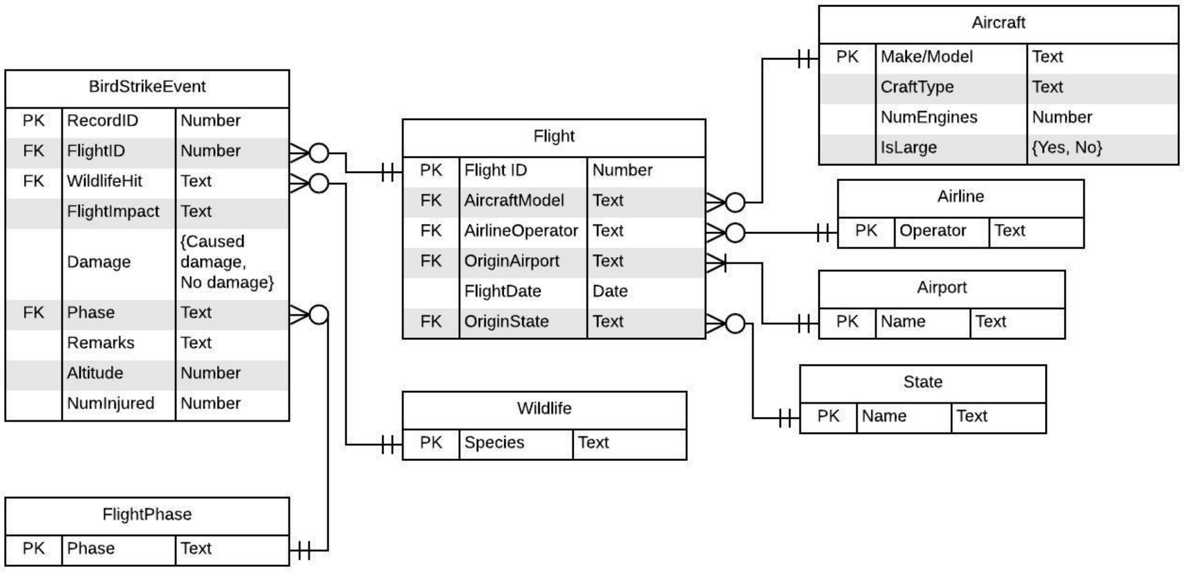
by Ernie Gurish

Practicum I - Bird Strike Data

(1) Create our ERD

First we create our logical model with an Entity-Relationship Diagram (ERD) using Crow's Foot notation. For the sake of simplicity, I have left lookup tables for categorical values out of both the ERD and the implementation, though a true implementation should likely have them.

The ERD/model is pictured below and the link to it is: https://lucid.app/documents/view/f4bde2bb-9f7f-4e56-96f8-84c47222620b



#

Connect to our database

Before we can create our tables with DDL statements, we must connect to our existing (but empty) database file. Here I used MySQL installed locally on my computer.

```
# Make sure our MySQL connection library is installed
if("RMariaDB" %in% rownames(installed.packages()) == FALSE) {
  install.packages("RMariaDB")
}
library(RMariaDB)

dbfile <- "EGpracticumI"
usr <- "rstudio"
pwd <- "cs5200p1"
```

```
# Connect to the local database
dbcon <- dbConnect(RMariaDB::MariaDB(), username=usr, password=pwd, dbname=dbfile)

# Check that our connection was successful and no tables currently exist
dbListTables(dbcon)
```

```
## character(0)
```

(2) Create our tables via DDL statements

Now we use our CREATE TABLE statements to realize the ERD show above. We will set our primary keys, foreign keys, and any basic constraints or default values. We can drop each table first if we need to reset any data, as shown in the R chunk below.

```
dbSendStatement(dbcon, "DROP TABLE IF EXISTS bird_strike_event;")
dbSendStatement(dbcon, "DROP TABLE IF EXISTS flight;")
```

```
## Warning in result_create(conn@ptr, statement, is_statement): Cancelling previous
## query
```

```
dbSendStatement(dbcon, "DROP TABLE IF EXISTS aircraft;")
```

```
## Warning in result_create(conn@ptr, statement, is_statement): Cancelling previous
## query
```

```
dbSendStatement(dbcon, "DROP TABLE IF EXISTS wildlife;")
```

```
## Warning in result_create(conn@ptr, statement, is_statement): Cancelling previous
## query
```

```
dbSendStatement(dbcon, "DROP TABLE IF EXISTS flight_phase;")
```

```
## Warning in result_create(conn@ptr, statement, is_statement): Cancelling previous
## query
```

```
dbSendStatement(dbcon, "DROP TABLE IF EXISTS airline;")
```

```
## Warning in result_create(conn@ptr, statement, is_statement): Cancelling previous
## query
```

```
dbSendStatement(dbcon, "DROP TABLE IF EXISTS airport;")
```

```
## Warning in result_create(conn@ptr, statement, is_statement): Cancelling previous
## query
```

```
dbSendStatement(dbcon, "DROP TABLE IF EXISTS state;")
```

```
## Warning in result_create(conn@ptr, statement, is_statement): Cancelling previous
## query
```

```
CREATE TABLE IF NOT EXISTS aircraft (
    make_model VARCHAR(120),
    craft_type VARCHAR(50) NOT NULL DEFAULT "Airplane",
    num_engines INTEGER NOT NULL,
    is_large VARCHAR(5),
    PRIMARY KEY (make_model)
);
```

```
CREATE TABLE IF NOT EXISTS wildlife (
    species VARCHAR(120),
    PRIMARY KEY (species)
);
```

```
CREATE TABLE IF NOT EXISTS flight_phase (
    phase VARCHAR(120),
    PRIMARY KEY (phase)
);
```

```
CREATE TABLE IF NOT EXISTS airline (
    operator VARCHAR(120),
    PRIMARY KEY (operator)
);
```

```
CREATE TABLE IF NOT EXISTS airport (
    name VARCHAR(120),
    PRIMARY KEY (name)
);
```

```
CREATE TABLE IF NOT EXISTS state (
    name VARCHAR(50),
    PRIMARY KEY (name)
);
```

```
CREATE TABLE IF NOT EXISTS flight (
    flight_id INTEGER AUTO_INCREMENT,
    aircraft_model VARCHAR(120) NOT NULL,
    airline_operator VARCHAR(120) NOT NULL,
    origin_airport VARCHAR(120) NOT NULL,
    flight_date DATE NOT NULL,
    origin_state VARCHAR(50) NOT NULL,
    PRIMARY KEY (flight_id),
    FOREIGN KEY (aircraft_model) REFERENCES aircraft(make_model),
    FOREIGN KEY (airline_operator) REFERENCES airline(operator),
    FOREIGN KEY (origin_airport) REFERENCES airport(name),
    FOREIGN KEY (origin_state) REFERENCES state(name)
);
```

```
CREATE TABLE IF NOT EXISTS bird_strike_event (
    record_id INTEGER,
    flight_id INTEGER,
    wildlife_hit VARCHAR(120),
    flight_impact VARCHAR(120) NOT NULL DEFAULT "None",
    damage VARCHAR(20) DEFAULT "No damage",
    phase VARCHAR(120),
    remarks VARCHAR(1000) DEFAULT "None",
    altitude INTEGER NOT NULL,
    num_injured INTEGER NOT NULL,
    PRIMARY KEY (record_id),
    FOREIGN KEY (flight_id) REFERENCES flight(flight_id),
    FOREIGN KEY (wildlife_hit) REFERENCES wildlife(species),
    FOREIGN KEY (phase) REFERENCES flight_phase(phase)
);
```

(3) Load bird strike data from CSV

Start by loading in our CSV…

```
fpath <- "c:/Users/emgur/Desktop/NEU/CS5200/Practicum I"
fname <- "BirdStrikesData.csv"
fileName <- paste(fpath, fname, sep = "/")
birdDF <- read.csv(fileName, header = TRUE, stringsAsFactors = FALSE)
```

…and taking an initial look at the data.

```
head(birdDF, 5)
```

```
##   ï..Record.ID Aircraft..Type               Airport..Name Altitude.bin
## 1       202152       Airplane                 LAGUARDIA NY    > 1000 ft
## 2       208159       Airplane DALLAS/FORT WORTH INTL ARPT    < 1000 ft
## 3       207601       Airplane            LAKEFRONT AIRPORT    < 1000 ft
## 4       215953       Airplane          SEATTLE-TACOMA INTL    < 1000 ft
## 5       219878       Airplane                 NORFOLK INTL    < 1000 ft
##   Aircraft..Make.Model Wildlife..Number.struck Wildlife..Number.Struck.Actual
## 1            B-737-400                Over 100                            859
## 2                MD-80                Over 100                            424
## 3                C-500                Over 100                            261
## 4            B-737-400                Over 100                            806
## 5         CL-RJ100/200                Over 100                            942
##   Effect..Impact.to.flight      FlightDate Effect..Indicated.Damage
## 1         Engine Shut Down 11/23/2000 0:00           Caused damage
## 2                     None  7/25/2001 0:00           Caused damage
## 3                     None  9/14/2001 0:00               No damage
## 4    Precautionary Landing   9/5/2002 0:00               No damage
## 5                     None  6/23/2003 0:00               No damage
##   Aircraft..Number.of.engines. Aircraft..Airline.Operator Origin.State
## 1                            2               US AIRWAYS*     New York
## 2                            2         AMERICAN AIRLINES        Texas
## 3                            2                  BUSINESS    Louisiana
## 4                            2           ALASKA AIRLINES   Washington
## 5                            2           COMAIR AIRLINES     Virginia
##   When..Phase.of.flight Conditions..Precipitation
## 1                 Climb                      None
## 2          Landing Roll                      None
## 3              Approach                      None
## 4                 Climb                      None
## 5              Approach                      None
##   Remains.of.wildlife.collected. Remains.of.wildlife.sent.to.Smithsonian
## 1                          FALSE                                   FALSE
## 2                          FALSE                                   FALSE
## 3                          FALSE                                   FALSE
## 4                           TRUE                                   FALSE
## 5                          FALSE                                   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
##   Wildlife..Size Conditions..Sky     Wildlife..Species
## 1        Medium        No Cloud Unknown bird - medium
## 2         Small      Some Cloud           Rock pigeon
## 3         Small        No Cloud     European starling
## 4         Small      Some Cloud     European starling
## 5         Small        No Cloud     European starling
##   Pilot.warned.of.birds.or.wildlife. Cost..Total.. Feet.above.ground
## 1                                  N        30,736             1,500
## 2                                  Y             0                 0
## 3                                  N             0                50
## 4                                  Y             0                50
## 5                                  N             0                50
##   Number.of.people.injured Is.Aircraft.Large.
## 1                        0                Yes
## 2                        0                 No
## 3                        0                 No
## 4                        0                Yes
## 5                        0                 No
```

We can then:
1. Remove unnecessary characters from our column names to make them easier to access

```r
# Replace unwanted characters with empty character
for (i in 1:ncol(birdDF)) {
  colnames(birdDF)[i] <- gsub("[.ï]", "", colnames(birdDF)[i])
}
```

2. Remove columns not needed for our database requirements

```r
# Remove columns containing data not needed for our app
birdDF <- subset(birdDF, select = -c(WildlifeNumberStruckActual, Altitudebin,
                                     WildlifeNumberstruck, WildlifeNumberStruckActual,
                                     ConditionsPrecipitation, Remainsofwildlifecollected,
                                     RemainsofwildlifesenttoSmithsonian, WildlifeSize,
                                     ConditionsSky,Pilotwarnedofbirdsorwildlife,
                                     CostTotal))
```

3. Remove data we don't know or fill in data based on our assumptions

I have kept some columns not strictly needed for the practicum (e.g. damage, pilot remarks, etc.) under the assumption that an app for pilots to report bird strikes to might want to collect this data.

```r
# Remove rows lacking flight or aircraft data. All the records this one line removes
# are the rows with missing airport info, size info, as well as aircraft type
birdDF <- birdDF[-which(birdDF$AircraftType == ""), ]

# Convert it to correct date formatting
birdDF$FlightDate <- as.Date(birdDF$FlightDate, "%m/%d/%Y")
# Store it back into character as otherwise MySQL interprets it as a double
birdDF$FlightDate <- as.character(birdDF$FlightDate)

# Fix blank number of engines to be 2 as it is the most common
birdDF$AircraftNumberofengines[which(birdDF$AircraftNumberofengines == "")] <- "2"
birdDF$AircraftNumberofengines[which(birdDF$AircraftNumberofengines == "C")] <- "2"

# A few aircraft models have multiple engine numbers listed (important as Model is our PK)
# Fix this under the assumption it should be most frequently appearing number of engines
# for that particular model number
```

```
birdDF$AircraftNumberofengines[which(birdDF$AircraftMakeModel == "A-300" &
                                       birdDF$AircraftNumberofengines == "4")] <- "2"
birdDF$AircraftNumberofengines[which(birdDF$AircraftMakeModel == "B-747-8 SERIES" &
                                       birdDF$AircraftNumberofengines == "4")] <- "2"

# This should also be an integer, not character field
birdDF$AircraftNumberofengines <- as.integer(birdDF$AircraftNumberofengines)

# Feet above ground should be integer as well, but must remove commas first
birdDF$Feetaboveground <- gsub("[,]", "", birdDF$Feetaboveground)
birdDF$Feetaboveground <- as.integer(birdDF$Feetaboveground)

# Set remarks to "None" if empty
birdDF$Remarks[which(birdDF$Remarks == "")] <- "None"

# Remove unwanted * character from some US AIRWAYS entries
birdDF$AircraftAirlineOperator[which(birdDF$AircraftAirlineOperator ==
                                                          "US AIRWAYS*")] <- "US AIRWAYS"

# Set all "Unknown bird" entries to say only that (a lot of them include size information)
birdDF$WildlifeSpecies[which(grepl("Unknown bird",
                                   birdDF$WildlifeSpecies))] <- "Unknown bird"

# Also, we use a flight_no field to track our flights so we can add that
# (arbitrary) PK here to make the import process easier
birdDF$flight_id <- 1:nrow(birdDF)
```

Now that our data is clean, we can begin loading the information into our tables. We will check to make sure we successfully loaded the data after each table by querying a small sample.

Note that we will use sqldf to format our tables to be written out to the database.

```
if("sqldf" %in% rownames(installed.packages()) == FALSE) {
  install.packages("sqldf")
}
library(sqldf)
```

```
## Loading required package: gsubfn

## Loading required package: proto

## Loading required package: RSQLite
```

Start with some of our smaller tables:

```
df.wildlife <- sqldf("SELECT DISTINCT WildlifeSpecies as species
                     FROM birdDF
                     ORDER BY species;")
dbWriteTable(dbcon, "wildlife", df.wildlife, append = TRUE)

dbGetQuery(dbcon, "SELECT * FROM wildlife LIMIT 10;")
```

```
##                    species
## 1      Acadian flycatcher
## 2      American alligator
## 3         American avocet
## 4        American bittern
## 5     American black duck
## 6           American coot
## 7           American crow
## 8  American golden-plover
## 9      American goldfinch
## 10       American kestrel
```

```
df.phase <- sqldf("SELECT DISTINCT WhenPhaseofflight as phase
                    FROM birdDF
                    ORDER BY phase;")
dbWriteTable(dbcon, "flight_phase", df.phase, append = TRUE)

dbGetQuery(dbcon, "SELECT * FROM flight_phase LIMIT 10;")
```

```
##          phase
## 1     Approach
## 2        Climb
## 3      Descent
## 4 Landing Roll
## 5       Parked
## 6 Take-off run
## 7         Taxi
```

```
df.airline <- sqldf("SELECT DISTINCT AircraftAirlineOperator as operator
                    FROM birdDF
                    ORDER BY operator;")
dbWriteTable(dbcon, "airline", df.airline, append = TRUE)

dbGetQuery(dbcon, "SELECT * FROM airline LIMIT 10;")
```

```
##                        operator
## 1  ABSA AEROLINHAS BRASILEIRAS
## 2                      ABX AIR
## 3                 ACM AVIATION
## 4            ADI SHUTTLE GROUP
## 5                   AER LINGUS
## 6                     AERO AIR
## 7                     AEROFLOT
## 8        AEROLINEAS EJECUTIVAS
## 9                  AEROLITORAL
## 10                  AEROMEXICO
```

```
df.airport <- sqldf("SELECT DISTINCT AirportName as name
                    FROM birdDF
                    ORDER BY name;")
dbWriteTable(dbcon, "airport", df.airport, append = TRUE)

dbGetQuery(dbcon, "SELECT * FROM airport LIMIT 10;")
```

```
##                             name
## 1          ABERDEEN REGIONAL AR
## 2         ABILENE REGIONAL ARPT
## 3  ABRAHAM LINCOLN CAPITAL ARPT
## 4    ADAMS COUNTY- LEGION FIELD
## 5              ADAMS FIELD ARPT
## 6          ADDINGTON FIELD ARPT
## 7                       ADDISON
## 8       ADIRONDAK REGIONAL ARPT
## 9              AIRBORNE AIRPARK
## 10      AKRON-CANTON MUNICIPAL
```

```
df.state <- sqldf("SELECT DISTINCT OriginState as name
                    FROM birdDF
                    ORDER BY name;")
dbWriteTable(dbcon, "state", df.state, append = TRUE)

dbGetQuery(dbcon, "SELECT * FROM state LIMIT 10;")
```

```
##                name
## 1           Alabama
## 2            Alaska
## 3           Alberta
## 4           Arizona
## 5          Arkansas
## 6  British Columbia
## 7        California
## 8          Colorado
## 9       Connecticut
## 10               DC
```

```
df.aircraft <- sqldf("SELECT DISTINCT AircraftMakeModel as make_model,
                    AircraftType as craft_type,
                    AircraftNumberofengines as num_engines,
                    IsAircraftLarge as is_large
                    FROM birdDF
                    ORDER BY AircraftMakeModel;")
dbWriteTable(dbcon, "aircraft", df.aircraft, append = TRUE)

dbGetQuery(dbcon, "SELECT * FROM aircraft LIMIT 10;")
```

```
##        make_model craft_type num_engines is_large
## 1           A-10A   Airplane           2       No
## 2  A-23 MUSKATEER   Airplane           2       No
## 3           A-300   Airplane           2       No
## 4           A-310   Airplane           2       No
## 5           A-318   Airplane           2       No
## 6           A-319   Airplane           2       No
## 7           A-320   Airplane           2       No
## 8           A-321   Airplane           2       No
## 9           A-330   Airplane           2       No
## 10          A-340   Airplane           4       No
```

Now we load our larger tables that require foreign key checking. We could have loaded these tables first by using dbSendStatement() to set foreign key checks off before we uploaded our data. However, doing it this way has the added benefit of making sure our foreign key tables are set up properly.

```
df.flight <- sqldf("SELECT flight_id,
                    AircraftMakeModel as aircraft_model,
                    AircraftAirlineOperator as airline_operator,
                    AirportName as origin_airport,
                    FlightDate as flight_date,
                    OriginState as origin_state
                    FROM birdDF
                    ORDER BY flight_id;")
dbWriteTable(dbcon, "flight", df.flight, append = TRUE)

dbGetQuery(dbcon, "SELECT * FROM flight LIMIT 10;")
```

```
##    flight_id aircraft_model  airline_operator                origin_airport
## 1          1      B-737-400        US AIRWAYS                  LAGUARDIA NY
## 2          2          MD-80 AMERICAN AIRLINES  DALLAS/FORT WORTH INTL ARPT
## 3          3          C-500          BUSINESS             LAKEFRONT AIRPORT
## 4          4      B-737-400   ALASKA AIRLINES           SEATTLE-TACOMA INTL
## 5          5    CL-RJ100/200   COMAIR AIRLINES                  NORFOLK INTL
## 6          6          A-300 AMERICAN AIRLINES           GUAYAQUIL/S BOLIVAR
## 7          7     LEARJET-25          BUSINESS             NEW CASTLE COUNTY
## 8          8          A-320   UNITED AIRLINES  WASHINGTON DULLES INTL ARPT
## 9          9        DC-9-30   AIRTRAN AIRWAYS                  ATLANTA INTL
## 10        10          A-330     AIRTOURS INTL ORLANDO SANFORD INTL AIRPORT
##    flight_date origin_state
## 1   2000-11-23     New York
## 2   2001-07-25        Texas
## 3   2001-09-14    Louisiana
## 4   2002-09-05   Washington
## 5   2003-06-23     Virginia
## 6   2003-07-24          N/A
## 7   2003-08-17     Delaware
## 8   2006-03-01           DC
## 9   2000-01-06      Georgia
## 10  2000-01-07      Florida
```

```
df.birdstrike <- sqldf("SELECT RecordID as record_id,
                         flight_id,
                         WildlifeSpecies as wildlife_hit,
                         EffectImpacttoflight as flight_impact,
                         EffectIndicatedDamage as damage,
                         Whenphaseofflight as phase,
                         Remarks as remarks,
                         Feetaboveground as altitude,
                         Numberofpeopleinjured as num_injured
                      FROM birdDF
                      ORDER BY record_id;")
dbWriteTable(dbcon, "bird_strike_event", df.birdstrike, append = TRUE)

dbGetQuery(dbcon, "SELECT * FROM bird_strike_event LIMIT 10;")
```

```
##    record_id flight_id     wildlife_hit         flight_impact    damage
## 1       1195       746     Unknown bird                  None No damage
## 2       3019      7843     Unknown bird Precautionary Landing No damage
## 3       3500      6006     Unknown bird Precautionary Landing No damage
## 4       3504       286     Unknown bird Precautionary Landing No damage
## 5       3597      5931 Upland sandpiper                  None No damage
## 6       4064        43     Unknown bird                  None No damage
## 7       4074       569    Mourning dove                  None No damage
## 8       4076      7339     Barn swallow Precautionary Landing No damage
## 9       4090       304     Unknown bird Precautionary Landing No damage
## 10      4091       308     Unknown bird      Aborted Take-off No damage
##           phase
## 1      Approach
## 2         Climb
## 3      Approach
## 4      Approach
## 5      Approach
## 6      Approach
## 7  Take-off run
## 8         Climb
## 9         Climb
## 10 Take-off run
##                                                                                      remarks
## 1                                                                                      None.
## 2                                                                                       None
## 3                                                                                       None
## 4                                                                                       None
## 5                                                                                       None
## 6       A bird struck the left inboard flap and one was ingested into the #7 engine intake.
## 7                                                                                       None
## 8  During touch and go bird struck the top of the nose radome between the #! and #2 window.
## 9                                                                                       None
## 10                                                                                      None
##    altitude num_injured
## 1      2000           0
## 2       400           0
## 3      1000           0
## 4      1800           0
## 5       200           0
## 6      1000           0
## 7         0           0
## 8       500           0
## 9        50           0
## 10        0           0
```

(4) SQL Query: Number of bird strikes for each airline upon take-off or climb

```sql
SELECT COUNT(record_id) AS NumBirdStrikes, F.airline_operator AS Airline
FROM bird_strike_event as B
  INNER JOIN flight AS F
  ON B.flight_id = F.flight_id
WHERE B.phase LIKE "Take-off%" OR B.phase LIKE "Climb"
GROUP BY Airline
ORDER BY NumBirdStrikes DESC;
```

Table 1: Displaying records 1 - 10

NumBirdStrikes	Airline
1544	SOUTHWEST AIRLINES
1287	BUSINESS
771	AMERICAN AIRLINES
575	US AIRWAYS
517	DELTA AIR LINES
324	AMERICAN EAGLE AIRLINES

NumBirdStrikes	Airline
282	SKYWEST AIRLINES
240	JETBLUE AIRWAYS
192	UNITED AIRLINES
178	FRONTIER AIRLINES

(5) SQL Query: Airport(s) that had the most bird strike incidents

```sql
SELECT MAX(NumBirdStrikes) as MostBirdStrikes, Airport
FROM (SELECT COUNT(record_id) AS NumBirdStrikes, F.origin_airport AS Airport
      FROM bird_strike_event AS B
        INNER JOIN flight AS F
        ON B.flight_id = F.flight_id
      GROUP BY Airport
      ORDER BY NumBirdStrikes DESC) AS NumStrikes;
```

Table 2: 1 records

MostBirdStrikes	Airport
803	DALLAS/FORT WORTH INTL ARPT

(6) SQL Query: Number of bird strike incidents by year

```sql
SELECT COUNT(B.record_id) AS NumBirdStrikes, EXTRACT(YEAR FROM F.flight_date) AS Year
FROM bird_strike_event as B
  INNER JOIN flight as F
  ON B.flight_id = F.flight_id
GROUP BY Year
ORDER BY Year;
```

Table 3: Displaying records 1 - 10

NumBirdStrikes	Year
1367	2000
1230	2001
1681	2002
1568	2003
1692	2004
1853	2005
2159	2006
2301	2007
2258	2008
3247	2009

(7) Visualizing the number of bird strike incidents per year (2008-2011) during take-off/climb and during decent/approach/landing

First we collect our take-off/climb data into a dataframe:

```
df.climbdata <- dbGetQuery(dbcon, "SELECT COUNT(B.record_id) AS NumBirdStrikes,
                                    EXTRACT(YEAR FROM F.flight_date) AS Year,
                                    \"Take-off/Climb\"
                                  FROM bird_strike_event as B
                                    INNER JOIN flight as F
                                    ON B.flight_id = F.flight_id
                                  WHERE (B.phase LIKE \"Take-off%\" OR
                                    B.phase LIKE \"Climb\")
                                  GROUP BY Year
                                  HAVING Year BETWEEN 2008 AND 2011
                                  ORDER BY Year;")
# Make sure the query worked
head(df.climbdata)
```

```
##   NumBirdStrikes Year Take-off/Climb
## 1            810 2008 Take-off/Climb
## 2           1127 2009 Take-off/Climb
## 3           1062 2010 Take-off/Climb
## 4           1030 2011 Take-off/Climb
```

Second get our descent/approach/landing data into a dataframe:

```
df.landdata <- dbGetQuery(dbcon, "SELECT COUNT(B.record_id) AS NumBirdStrikes,
                                    EXTRACT(YEAR FROM F.flight_date) AS Year,
                                    \"Descent/Approach/Landing\"
                                  FROM bird_strike_event as B
                                    INNER JOIN flight as F
                                    ON B.flight_id = F.flight_id
                                  WHERE (B.phase LIKE \"Descent\" OR
                                    B.phase LIKE \"Approach\" OR
                                    B.phase LIKE \"Landing%\")
                                  GROUP BY Year
                                  HAVING Year BETWEEN 2008 AND 2011
                                  ORDER BY Year;")
# Make sure the query worked
head(df.landdata)
```

```
##   NumBirdStrikes Year Descent/Approach/Landing
## 1           1442 2008 Descent/Approach/Landing
## 2           2109 2009 Descent/Approach/Landing
## 3           2053 2010 Descent/Approach/Landing
## 4           1913 2011 Descent/Approach/Landing
```

Next we can the join the two tables by year to consolidate our tables:

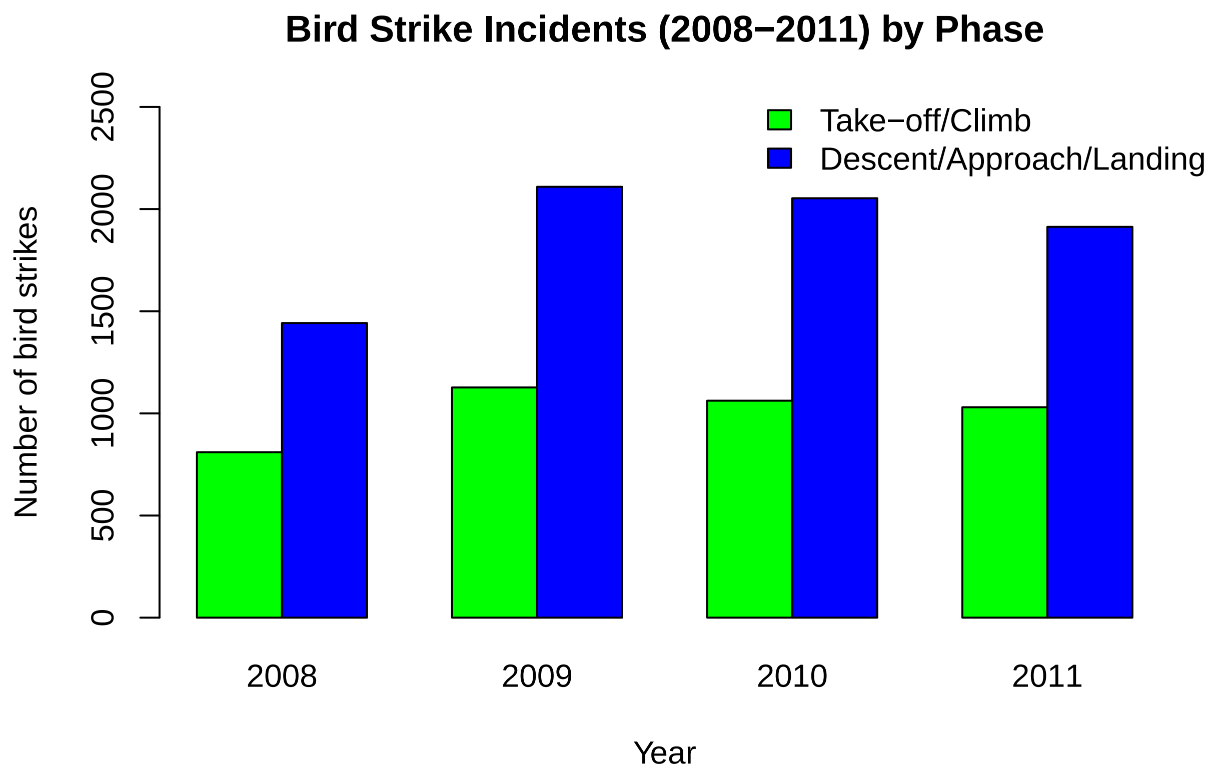
```
df.yearstrikes <- sqldf("SELECT C.NumBirdStrikes AS \"Take-off/Climb\",
                          L.NumBirdStrikes AS \"Descent/Approach/Landing\",
                          C.Year
                        FROM `df.climbdata` as C
                         INNER JOIN `df.landdata` as L
                         ON C.Year = L.Year
                        ORDER BY C.Year;", method = "raw")

# Make sure our join worked
head(df.yearstrikes, 4)
```

```
##   Take-off/Climb Descent/Approach/Landing Year
## 1            810                     1442 2008
## 2           1127                     2109 2009
## 3           1062                     2053 2010
## 4           1030                     1913 2011
```

Now we have our data together in one dataframe and we can finally reformat it into a matrix to easily create our barchart.

```
# Initialize our matrix
data.plot <- matrix(nrow = 2, ncol = 4)
# The take-off/climb & descent/landing columns become our rows
rownames(data.plot) <- c(colnames(df.yearstrikes)[1], colnames(df.yearstrikes)[2])
# Year becomes our column labels (& is to be the x-axis of our chart)
colnames(data.plot) <- df.yearstrikes$Year
# Fill in our number of strikes (number is our y-axis) for both rows
data.plot[1, ] <- df.yearstrikes$`Take-off/Climb`
data.plot[2, ] <- df.yearstrikes$`Descent/Approach/Landing`
# Finally plot it out
barplot(data.plot,
        main = "Bird Strike Incidents (2008-2011) by Phase",
        xlab = "Year",
        ylab = "Number of bird strikes",
        legend = rownames(data.plot),
        col = c("green", "blue"),
        ylim = c(0, 2500),
        args.legend = list(x = "topright", bty = "n", inset=c(-0.05,-0.05)),
        beside = TRUE)
```



I found the tutorial at this link (https://www.dataanalytics.org.uk/legends-on-graphs-and-charts/) to be very helpful for figuring out the correct formatting of the graph above.

(8) Creating a stored procedure to remove a bird strike incident

First we create our stored procedure. Since our bird_strike_event table has a primary key composed of solely a record_id, that is all the information we need to remove a bird strike incident. Additionally, our record_id is not referenced in any other table so we do not have to concern ourselves with updating any additional tables. I utilized this link (https://www.w3resource.com/mysql/mysql-procedure.php) to help ensure my stored procedure code was correct.

```
CREATE PROCEDURE RemoveBirdStrike(strike_id INTEGER)
BEGIN
  DELETE FROM bird_strike_event
  WHERE record_id = strike_id;
END;
```

Now that our procedure has been created, we can test it on our bird strike incident table. We can choose an arbitrary record_id to delete as a test. We will try to remove record_id 1195 from the result set below by CALLing our stored procedure:

```
SELECT *
FROM bird_strike_event
LIMIT 5;
```

Table 4: 5 records

record_id	flight_id	wildlife_hit	flight_impact	damage	phase	remarks	altitude	num_injured
1195	746	Unknown bird	None	No damage	Approach	None.	2000	0
3019	7843	Unknown bird	Precautionary Landing	No damage	Climb	None	400	0
3500	6006	Unknown bird	Precautionary Landing	No damage	Approach	None	1000	0
3504	286	Unknown bird	Precautionary Landing	No damage	Approach	None	1800	0
3597	5931	Upland sandpiper	None	No damage	Approach	None	200	0

```
CALL RemoveBirdStrike(1195);
```

We can double check that it is gone with:

```
SELECT *
FROM bird_strike_event
LIMIT 5;
```

Table 5: 5 records

record_id	flight_id	wildlife_hit	flight_impact	damage	phase	remarks	altitude	num_injured
3019	7843	Unknown bird	Precautionary Landing	No damage	Climb	None	400	0
3500	6006	Unknown bird	Precautionary Landing	No damage	Approach	None	1000	0
3504	286	Unknown bird	Precautionary Landing	No damage	Approach	None	1800	0
3597	5931	Upland sandpiper	None	No damage	Approach	None	200	0
4064	43	Unknown bird	None	No damage	Approach	A bird struck the left inboard flap and one was ingested into the #7 engine intake.	1000	0

And, to be even more sure that the deletion was successful, we can run a query looking for record_id 1195 specifically:

```
SELECT *
FROM bird_strike_event
WHERE record_id = 1195;
```

Table 6: 0 records

record_id	flight_id	wildlife_hit	flight_impact	damage	phase	remarks	altitude	num_injured

Lastly, disconnect from our database

```
dbDisconnect(dbcon)
```