Practicum I CS5200

Samuel Ghebreyesus

Spring 2023

Connect to Database

```
library(RMySQL)
```

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

```
conn = dbConnect(RMySQL::MySQL(), host='localhost',
                            port=3306, user='root',
                  password='Asm@rino123')
dbSendQuery(conn, "CREATE DATABASE birdstrikedb")
```

```
## <MySQLResult:1513697560,0,0>
```

```
dbSendQuery(conn, "USE birdstrikedb")
```

```
## <MySQLResult:1512622072,0,1>
```

```
dbSendQuery(conn, "SET GLOBAL local_infile=1")
```

```
## <MySQLResult:1512622576,0,2>
```

```
conn = dbConnect(RMySQL::MySQL(), dbname='birdstrikedb', host='localhost',
                            port=3306,
                  user='root', password='Asm@rino123')
```

Create Database

```
sql <- "DROP TABLE IF EXISTS airports, conditions, airlines, incidents"
dbSendStatement(conn, sql)
```

```
## <MySQLResult:12,1,0>
```

```
airquery <- "CREATE TABLE airports(
  aid INT NOT NULL AUTO_INCREMENT,
  airportName TEXT NOT NULL,
  airportCode TEXT,
  state TEXT NOT NULL,
  PRIMARY KEY (aid)
)"

airresults <- dbSendQuery(conn, airquery)
dbClearResult(airresults)
```

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

```
condquery <- "CREATE TABLE conditions(
  cid INTEGER NOT NULL,
  `condition` TEXT NOT NULL,
  explanation TEXT,
  PRIMARY KEY (cid)
)"

condresults <- dbSendQuery(conn, condquery)
dbClearResult(condresults)
```

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

```
airlinequery <- "CREATE TABLE airlines(
  eid INT NOT NULL AUTO_INCREMENT,
  airlineName TEXT NOT NULL,
  airlineCode TEXT,
  flag TEXT,
  PRIMARY KEY (eid)
)"

airlineresults <- dbSendQuery(conn, airlinequery)
dbClearResult(airlineresults)
```

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

```
incquery <- "CREATE TABLE incidents(
  rid INTEGER NOT NULL,
  `dep.date` DATE,
  origin INTEGER,
  airline INTEGER,
  aircraft TEXT,
  `flight.phase` ENUM('takeoff', 'landing', 'inflight', 'unknown'),
  altitude INTEGER,
  CHECK (altitude >= 0),
  conditions INTEGER,
  warned TEXT,
  PRIMARY KEY (rid),
  FOREIGN KEY (origin) REFERENCES airports (aid),
```

```
  FOREIGN KEY (conditions) REFERENCES conditions (cid),
  FOREIGN KEY (airline) REFERENCES airlines (eid)
)"

incresults <- dbSendQuery(conn, incquery)
dbClearResult(incresults)
```

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

Tests table definition; attempt to insert a negative number into altitude column, which is illegal and should return an error

```
sql <- "insert into incidents (rid, altitude) values (9999999, -1)"
dbSendQuery(conn, sql)
```

Tests table definition; attempt to insert a value into flight phase column that isn't takeoff, landing, inflight or unknown, should return error

```
sql <- "insert into incidents(rid, `flight.phase`)
values (10000000, 'driving');"
dbSendQuery(conn, sql)
```

Load birdstrikes csv data file into a dataframe called bds.raw. Printed out first 5 rows just to show that it worked.

```
fn = "BirdStrikesData-V2.csv"
bds.raw <- read.csv(file = fn,
                    header = T,
                    stringsAsFactors = F)
head(bds.raw, 5)
```

```
##      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
##                  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
##       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
##
## 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 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
##   altitude_ft heavy_flag
## 1       1,500        Yes
## 2           0         No
## 3          50         No
## 4          50        Yes
## 5          50         No
```

Populate the tables with the data from the appropriate dataframe columns. I went through each row and where there was no airport or airline, I put in the value 'Unknown'. Used sqldf to get the necessary subset of the csv file and bulk loaded the subset into the tables.

```
options(sqldf.driver = 'SQLite')
df.airports <- sqldf::sqldf("select 1 as aid,
                            airport as 'airportName',
                            '' as airportCode,
                            origin as 'state' from `bds.raw`
                            group by airport, state")
```

```
##
## Attaching package: 'RSQLite'

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

```
numairports <- nrow(df.airports)
df.airports[,1] <- seq(1, numairports)
for (r in 1:numairports) {
  if (df.airports$airportName[r] == '') {
    df.airports$airportName[r] = 'Unknown'
  }
}
dbWriteTable(conn, "airports", df.airports, overwrite = F, append = T,
             row.names = FALSE)
```

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

```
df.airlines <- sqldf::sqldf("select 1 as eid,
                            airline as 'airlineName',
                            '' as airlineCode,
                            '' as 'flag' from `bds.raw`
                            group by airline")
numairlines <- nrow(df.airlines)
df.airlines[,1] <- seq(1, numairlines)
for (r in 1:numairlines) {
  if (df.airlines$airlineName[r] == '') {
    df.airlines$airlineName[r] = 'Unknown'
  }
}
dbWriteTable(conn, "airlines", df.airlines, overwrite = F, append = T,
             row.names = FALSE)
```

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

```
df.conditions <- sqldf::sqldf("select 1 as cid, sky_conditions as 'condition',
                              '' as explanation from `bds.raw` group by sky_conditions")
numconditions <- nrow(df.conditions)
df.conditions[,1] <- seq(1, numconditions)
dbWriteTable(conn, "conditions", df.conditions, overwrite = F, append = T,
             row.names = FALSE)
```

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

This code chunk was used to populate the incidents table. I had to find the aid, eid and cid values that matched the airport/airline names and conditions to satisfy the foreign key constraint. I also had to map the flight phase values to their proper value in the value set. Based on research, I mapped approach, descent and landing roll to landing. I also mapped Climb to inflight, Take-off run to takeoff and everything else(taxi, parked, blank) to unknown

```
df.incidents <- sqldf::sqldf("select * from `bds.raw`")
numincidents <- nrow(df.incidents)
for (r in 1:numincidents) {
  if (df.incidents$airport[r] == '') {
    df.incidents$airport[r] = 'Unknown'
  }
  if (df.incidents$airline[r] == '') {
    df.incidents$airline[r] = 'Unknown'
  }
    a <- df.airports$aid[which(df.incidents$airport[r] ==
                               df.airports$airportName
                              & df.incidents$origin[r] == df.airports$state)]
    b <- df.airlines$eid[which(df.incidents$airline[r] ==
                               df.airlines$airlineName)]
    c <- df.conditions$cid[which(df.incidents$sky_conditions[r] == df.conditions$condition)]
```

```
    df.incidents$airport[r] = a
    df.incidents$airline[r] = b
    df.incidents$sky_conditions[r] = c
    if (df.incidents$flight_phase[r] == 'Approach'
        || df.incidents$flight_phase[r] == 'Descent'
        || df.incidents$flight_phase[r] == 'Landing Roll') {
      df.incidents$flight_phase[r] = 'landing'
    } else if (df.incidents$flight_phase[r] == 'Climb') {
      df.incidents$flight_phase[r] = 'inflight'
    } else if (df.incidents$flight_phase[r] == 'Take-off run') {
      df.incidents$flight_phase[r] = 'takeoff'
    } else {
      df.incidents$flight_phase[r] = 'unknown'
    }
}
df.incidents2 <- sqldf::sqldf("select rid,
                              flight_date as `dep.date`,
                              airport as 'origin',
                              airline, aircraft,
                              flight_phase as `flight.phase`,
                              altitude_ft as 'altitude',
                              sky_conditions as 'conditions',
                              pilot_warned_flag as 'warned'
                              from `df.incidents`")
df.incidents2$dep.date = substr(df.incidents2$dep.date, 1,
                            nchar(df.incidents2$dep.date)-5)
df.incidents2$dep.date <- as.Date(df.incidents2$dep.date, format = "%m/%d/%Y")
df.incidents2$altitude <- gsub(',', '', df.incidents$altitude)
dbWriteTable(conn, "incidents", df.incidents2, overwrite = F, append=T,
             row.names = FALSE)
```

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

Display the first five rows of each table to show that the data loading worked

```
airports <- dbGetQuery(conn, "select * from airports")
head(airports, 5)
```

```
##   aid airportName airportCode      state
## 1   1     Unknown                Alabama
## 2   2     Unknown                 Alaska
## 3   3     Unknown                Arizona
## 4   4     Unknown               Arkansas
## 5   5     Unknown             California
```

```
airlines <- dbGetQuery(conn, "select * from airlines")
head(airlines, 5)
```

```
##   eid                 airlineName airlineCode flag
## 1   1                     Unknown
```

```
## 2   2 ABSA AEROLINHAS BRASILEIRAS
## 3   3                     ABX AIR
## 4   4                ACM AVIATION
## 5   5           ADI SHUTTLE GROUP
```

```
conditions <- dbGetQuery(conn, "select * from conditions")
head(conditions, 5)
```

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

```
incidents <- dbGetQuery(conn, "select * from incidents")
incidents$dep.date = as.Date(incidents$dep.date)
head(incidents, 5)
```

```
##    rid   dep.date origin airline aircraft flight.phase altitude conditions
## 1 1195 2002-11-13     89     198 Airplane      landing     2000          2
## 2 3019 2002-10-10    296     198 Airplane     inflight      400          1
## 3 3500 2001-05-15     89     198 Airplane      landing     1000          1
## 4 3504 2001-05-23     89     198 Airplane      landing     1800          1
## 5 3597 2001-04-18    952     198 Airplane      landing      200          3
##   warned
## 1      Y
## 2      Y
## 3      Y
## 4      Y
## 5      Y
```

SQL query to find the 10 states with the greatest number of incidents

```
select state, count(rid) as 'numIncidents' from (
  select incidents.rid, airports.state
  from incidents
  inner join airports on incidents.origin = airports.aid) as rid_state
  group by rid_state.state
  order by numIncidents desc limit 10
```

Table 1: Displaying records 1 - 10

state	numIncidents
California	2520
Texas	2453
Florida	2055
New York	1319
Illinois	1008
Pennsylvania	986
Missouri	960

state	numIncidents
Kentucky	812
Ohio	778
Hawaii	729

SQL query to find the airlines that had an above average number bird strike incidents

```
select * from (
select airlines.airlineName, count(incidents.rid) as 'numIncidents'
from airlines
inner join incidents on incidents.airline = airlines.eid
group by airlines.eid) as incident_airline_table
where numIncidents > (select count(rid) / count(distinct airline) from incidents)
order by numIncidents asc
```

Table 2: Displaying records 1 - 10

airlineName	numIncidents
MILITARY	89
COMMUTAIR	92
AIR CANADA	95
ALLEGIANT AIR	107
PIEDMONT AIRLINES	113
ATLANTIC COAST AIRLINES	115
REPUBLIC AIRLINES	120
ISLAND AIR	122
Unknown	129
ALOHA AIRLINES	135

SQL query to find the number of bird strike incidents by month and by flight phase, across all years and including all airlines and flights. I excluded from the data frame the rows that had unknown month as that would not be useful data for the analytics we are trying to run.

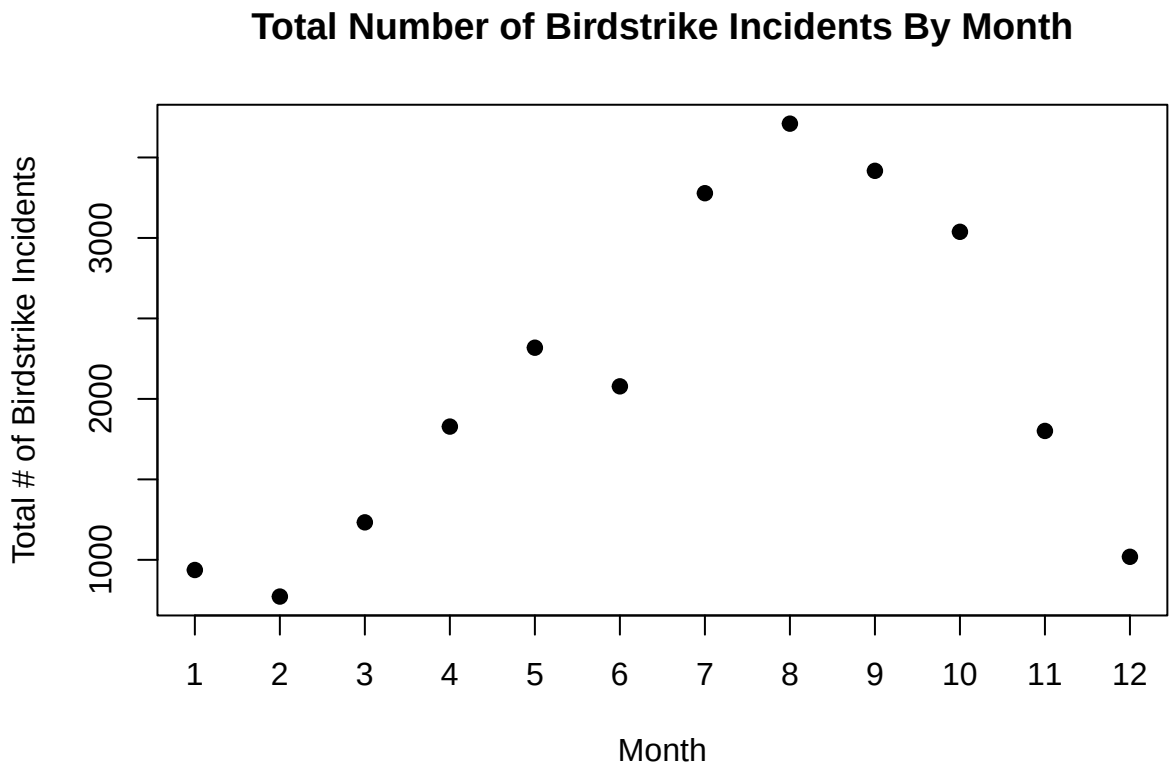
```
sql <- "select MONTH(`dep.date`) as 'Month', `flight.phase`, count(rid) as 'numIncidents'
from incidents
where MONTH(`dep.date`) IS NOT NULL
group by Month, `flight.phase`
order by Month, `flight.phase`"
month_phase_df_display <- dbGetQuery(conn, paste(sql, "limit 6"))
month_phase_df_full <- dbGetQuery(conn, sql)
month_phase_df_display
```

```
##   Month flight.phase numIncidents
## 1     1      takeoff          164
```

```
## 2     1      landing          576
## 3     1     inflight          193
## 4     1      unknown            4
## 5     2      takeoff          126
## 6     2      landing          477
```

Build a scatter plot that plots month along the x-axis versus number of incidents across all airlines and flight phases on the y-axis

```
sum_df <- aggregate(month_phase_df_full$numIncidents, list(month_phase_df_full$Month),
                    FUN=sum)
plot(sum_df$Group.1, sum_df$x, xlab="Month",
    ylab="Total # of Birdstrike Incidents",
    main="Total Number of Birdstrike Incidents By Month",
    pch=19, xaxt='n')
axis(side=1, at=seq(1, 12, by=1))
```



Stored procedure in MySQL that adds a new incident to the database. It takes in as parameters airport name, state, airline name, rid, date, aircraft, flight phase, condition and warned flag (y/n). If the airport or airline does not already exist in their respective tables then they are newly added to the airports/airlines table.

```
CREATE PROCEDURE INSERTNEWINCIDENT(
    IN newairportName TEXT,
    IN newstate TEXT,
    IN newairlineName TEXT,
    IN newrid INT,
    IN newdate DATE,
    IN newaircraft TEXT,
    IN newflightphase TEXT,
    IN newaltitude INT,
    IN newconditions INT,
    IN newWarned TEXT
)
BEGIN
    IF NOT EXISTS (SELECT * FROM AIRPORTS
    WHERE airportName = newairportName and
    state = newstate) THEN
        INSERT INTO airports (airportName, state)
        VALUES (newairportName, newstate);
    END IF;
    IF NOT EXISTS (SELECT * FROM AIRLINES WHERE airlineName = newairlineName)
    THEN
        INSERT INTO AIRLINES (airlineName)
        VALUES (newairlineName);
    END IF;
    INSERT INTO INCIDENTS(rid, `dep.date`,
    origin, airline, aircraft, `flight.phase`,
    altitude, conditions, warned)
    VALUES(newrid, newdate, (select aid from airports
where airportName = newairportName and state = newstate), (select eid from
airlines where airlineName = newairlineName),
newaircraft, newflightphase, newaltitude,
newconditions, newWarned);
END;
```

Calling my stored procedure to insert these 5 incidents. The first three incidents contain new airports, with two of them also including new airlines. The last two incidents contain airlines/airports that already exist.

```
CALL INSERTNEWINCIDENT("Asmara International", "Africa", "Africa Airways",
9999997, "2010-11-19", "Airplane", "unknown", 200, 2, "Y");
```

```
CALL INSERTNEWINCIDENT("Addis Airport", "Africa", "Ethiopia Airlines",
9999998, "2016-11-19", "Airplane", "unknown", 1200, 3, "N");
```

```
CALL INSERTNEWINCIDENT("Unknown", "Michigan", "ACM AVIATION", 9999999,
"2014-11-16", "Airplane", "landing", 1700, 1, "N");
```

```
CALL INSERTNEWINCIDENT("ANN ARBOR MUNICIPAL", "Michigan", "ACM AVIATION",
10000000, "2014-11-16", "Airplane", "landing", 1500, 3, "Y");
```

```
CALL INSERTNEWINCIDENT("Unknown", "Maryland", "American Airlines",
10000001, "2014-11-16", "Airplane", "landing", 1500, 3, "N");
```

Testing that the 3 new airports were added to the airports table and thus that the stored procedure worked

```
sql <- "select * from airports
where airportName = 'Asmara International'
or airportName = 'Addis Airport'
or (airportName = 'Unknown' and state ='Michigan');"
dbGetQuery(conn, sql)
```

```
##    aid          airportName airportCode    state
## 1 1142 Asmara International        <NA>   Africa
## 2 1143        Addis Airport        <NA>   Africa
## 3 1144              Unknown        <NA> Michigan
```

Testing that the 2 new airlines were added to the airlines table and thus that the stored procedure worked

```
sql <- "select * from airlines
where airlineName = 'Africa Airways' or airlineName = 'Ethiopia Airlines'"
dbGetQuery(conn, sql)
```

```
##   eid       airlineName airlineCode flag
## 1 294    Africa Airways        <NA> <NA>
## 2 295 Ethiopia Airlines        <NA> <NA>
```

Testing that the 5 incidents were added to the incidents table and thus that the stored procedure worked

```
sql <- "select * from incidents
where rid >= 9999997 and rid <= 10000001;"
dbGetQuery(conn, sql)
```

```
##        rid   dep.date origin airline aircraft flight.phase altitude conditions
## 1  9999997 2010-11-19   1142     294 Airplane      unknown      200          2
## 2  9999998 2016-11-19   1143     295 Airplane      unknown     1200          3
## 3  9999999 2014-11-16   1144       4 Airplane      landing     1700          1
```

```
## 4 10000000 2014-11-16     61       4 Airplane      landing     1500          3
## 5 10000001 2014-11-16     14      47 Airplane      landing     1500          3
##   warned
## 1      Y
## 2      N
## 3      N
## 4      Y
## 5      N
```

Test that only the three new airports were inserted; original airlines table had 1141 rows

```
sql <- "select count(*) from airports;"
dbGetQuery(conn, sql)
```

```
##   count(*)
## 1     1144
```

Test that only the two new airlines were inserted; original airports table had 293 rows

```
sql <- "select count(*) from airlines;"
dbGetQuery(conn, sql)
```

```
##   count(*)
## 1      295
```

Test that 5 incidents were inserted; original incidents table had 25558 rows

```
sql <- "select count(*) from incidents"
dbGetQuery(conn, sql)
```

```
##   count(*)
## 1    25563
```