---
title: "Practicum I CS5200"
Author: Pegah Zargarian
Date: Fall 2023
output:
html_document:
df_print: paged

Connect to Database

```
# Load the library
library(RMySQL)
```

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

```
library(DBI)


dbcon <- dbConnect(MySQL(),
                   dbname = 'sql3653854',
                   user = 'sql3653854',
                   password = 'hLYvdVVihH',
                   host = 'sql3.freemysqlhosting.net')
```

Create Database

```
CREATE TABLE IF NOT EXISTS airports (
    aid INT PRIMARY KEY NOT NULL,
    airportName VARCHAR(100),
    /*origin in CSV */
    airportState VARCHAR(50),
    airportCode VARCHAR(50)
);
```

```
CREATE TABLE  IF NOT EXISTS flights(
  fid INTEGER PRIMARY KEY NOT NULL,
  flight_date DATE,
  origin_aid INTEGER,
  airline VARCHAR(100),
  aircraft VARCHAR(50),
  heavy TINYINT(1),
  FOREIGN KEY (origin_aid) REFERENCES airports(aid)
  );
```

```
CREATE VIEW flightsView AS
SELECT
  fid,
  flight_date,
  origin_aid,
  airline,
  aircraft,
  CASE
    WHEN heavy = 1 THEN 'TRUE'
    ELSE 'FALSE'
```

```
  END AS heavy
FROM flights;
```

```
CREATE TABLE IF NOT EXISTS conditions(
    cid INT PRIMARY KEY NOT NULL,
    sky_condition VARCHAR(50) UNIQUE,
    explanation TEXT

);
```

```
CREATE TABLE IF NOT EXISTS strikes(
    sid INT PRIMARY KEY NOT NULL,
    numbirds INT,
    impact TEXT,
    damage TINYINT(1),
    altitude INT CHECK (altitude >= 0),
    conditions_id INT,
    fid INT,
    FOREIGN KEY (conditions_id) REFERENCES conditions(cid),
    FOREIGN KEY (fid) REFERENCES flights(fid)

);
```

```
CREATE VIEW strikesView AS
SELECT
    sid,
    numbirds,
    impact,
    CASE
        WHEN damage= 1 THEN 'TRUE'
        ELSE 'FALSE'
    END AS damage,
    altitude,
    conditions_id,
    fid
FROM strikes;
```

```
bds.raw <- read.csv("BirdStrikesData-V2.csv")
# Specify the columns to handle missing values
columns_to_handle <- c("aircraft",  "airport","impact", "airline" ,"origin","sky_conditions")

# Loop through each column and replace missing values
for (col in columns_to_handle) {
  bds.raw[[col]][is.na(bds.raw[[col]]) | bds.raw[[col]] == ""] <- "Unknown"
}
bds.raw$altitude_ft[is.na(bds.raw$altitude_ft) | bds.raw$altitude_ft ==""] <- 0
```

```
SELECT * FROM airports WHERE airportState =NULL;
```

Table 2: 0 records

aid	airportName	airportState	airportCode

```sql
SELECT * FROM flights WHERE flight_date = NULL;
```

Table 3: 0 records

fid	flight_date	origin_aid	airline	aircraft	heavy

```sql
SELECT * FROM strikes WHERE altitude = NULL;
```

Table 4: 0 records

sid	numbirds	impact	damage	altitude	conditions_id	fid

```r
# airports
# Creating a new data frame with the columns needed
n.flights <- nrow(bds.raw)
airports_subset_df<- data.frame(
  aid = 100 + seq(1,n.flights),
  airportName = bds.raw$airport,
  airportState = bds.raw$origin,
  airportCode = ""
)

# Bulk insert into the 'flights' table without the 'origin' column
dbWriteTable(dbcon, "airports", airports_subset_df, row.names = FALSE, append = TRUE)
```

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

```r
#conditions
# Creating a new data frame with the columns needed
n.conditions <- nrow(bds.raw)
conditions_df <- data.frame(
  cid = 100 + seq(1,n.conditions),
  sky_condition = bds.raw$sky_conditions,
  explanation =""
)


dbWriteTable(dbcon, "conditions", conditions_df , row.names = FALSE, append = TRUE)
```

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

```r
#flights
# Assuming you have the aid values from the airports_subset_df
airports_aid <- airports_subset_df$aid

# Creating a new data frame with the columns needed
n.flights <- nrow(bds.raw)
flights_df <- data.frame(
  fid = bds.raw$rid,
  flight_date = as.Date(bds.raw$flight_date, format = "%m/%d/%Y %H:%M"),
  airline = bds.raw$airline,
  aircraft = bds.raw$aircraft,
  origin_aid = 0,  # Placeholder for foreign key
```

```
  heavy =  ifelse(bds.raw$heavy_flag == 'Yes', 1, 0)
)

# Match airline and airportState to find the corresponding aid
for (r in 1:n.flights) {
  match_idx <- which(airports_subset_df$airportState == bds.raw$origin[r] &
                     airports_subset_df$airportName == bds.raw$airport[r])

  if (length(match_idx) > 0) {
    flights_df$origin_aid[r] <- airports_aid[match_idx[1]]
  }
}

# Bulk insert into the 'strikes'
dbWriteTable(dbcon, "flights", flights_df, row.names = FALSE, append= TRUE)
```

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

```
UPDATE flights
SET flight_date = '1900-01-01'
WHERE flight_date IS NULL;
```

```
#strikes

# Creating a new data frame with the columns needed
n.flights <- nrow(bds.raw)
strikes_df <- data.frame(
  sid = 10000 + seq(1,n.flights),
  numbirds = bds.raw$wildlife_struck,
  damage = ifelse(bds.raw$damage == 'Caused damage', 1, 0),
  impact = bds.raw$impact,
  altitude = bds.raw$altitude_ft,
  conditions_id = 0,
  fid = 0
)


# Match airline and airportState to find the corresponding aid
for (r in 1:n.flights)   {
  conditionsRow <- conditions_df$cid[which(conditions_df$sky_condition==bds.raw$sky_conditions[r])]
  strikes_df$conditions_id[r] <- conditionsRow   }

flightDate <- as.Date(bds.raw$flight_date, format = "%m/%d/%Y")
for (r in 1:n.flights) {
  fidRow <- flights_df$fid[which(flights_df$fid == bds.raw$rid[r])]
  strikes_df$fid[r] <- fidRow
}


# Bulk insert into the 'strikes'
dbWriteTable(dbcon, "strikes", strikes_df, row.names = FALSE, append = TRUE)
```

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

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

Table 5: 5 records

aid	airportName	airportState	airportCode
101	LAGUARDIA NY	New York	
102	DALLAS/FORT WORTH INTL ARPT	Texas	
103	LAKEFRONT AIRPORT	Louisiana	
104	SEATTLE-TACOMA INTL	Washington	
105	NORFOLK INTL	Virginia	

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

Table 6: 3 records

cid	sky_condition	explanation
101	No Cloud	
102	Some Cloud	
113	Overcast	

```
SELECT * FROM flightsView
LIMIT 5;
```

Table 7: 5 records

fid	flight_date	origin_aid	airline	aircraft	heavy
1195	2002-11-13	143	MILITARY	Airplane	FALSE
3019	2002-10-10	7961	MILITARY	Airplane	FALSE
3500	2001-05-15	143	MILITARY	Airplane	FALSE
3504	2001-05-23	143	MILITARY	Airplane	FALSE
3597	2001-04-18	339	MILITARY	Airplane	FALSE

```
SELECT * FROM strikesView
LIMIT 5;
```

Table 8: 5 records

sid	numbirds	impact	damage	altitude	conditions_id	fid
10001	859	Engine Shut Down	TRUE	1	101	202152
10002	424	None	TRUE	0	102	208159
10003	261	None	FALSE	50	101	207601
10004	806	Precautionary Landing	FALSE	50	102	215953
10005	942	None	FALSE	50	101	219878

```
SELECT
    a.airportState AS state,
    COUNT(*) AS numbirds
```

```
FROM
    strikes s
JOIN
    flights f ON s.fid = f.fid
JOIN
    airports a ON f.origin_aid = a.aid
GROUP BY
    a.airportState
ORDER BY
    numbirds DESC
LIMIT 10;
```

Table 9: Displaying records 1 - 10

state	numbirds
California	2520
Texas	2453
Florida	2055
New York	1319
Illinois	1008
Pennsylvania	986
Missouri	960
Kentucky	812
Ohio	778
Hawaii	729

```
SELECT f.airline as airline, COUNT(s.sid) as num_of_strikes
FROM flights AS f JOIN strikes AS s ON f.fid = s.fid
GROUP BY f.airline HAVING num_of_strikes > (SELECT AVG(incident_count)
FROM (SELECT count(*) as incident_count FROM strikes
GROUP BY fid) AS average_incidents)
ORDER BY num_of_strikes DESC;
```

Table 10: Displaying records 1 - 10

airline	num_of_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

```
SELECT
    MONTH(f.flight_date) AS month,
     SUM(s.numbirds) AS total_birds
```

```
FROM
    strikes s
JOIN
    flights f ON s.fid = f.fid
GROUP BY
    month;
```

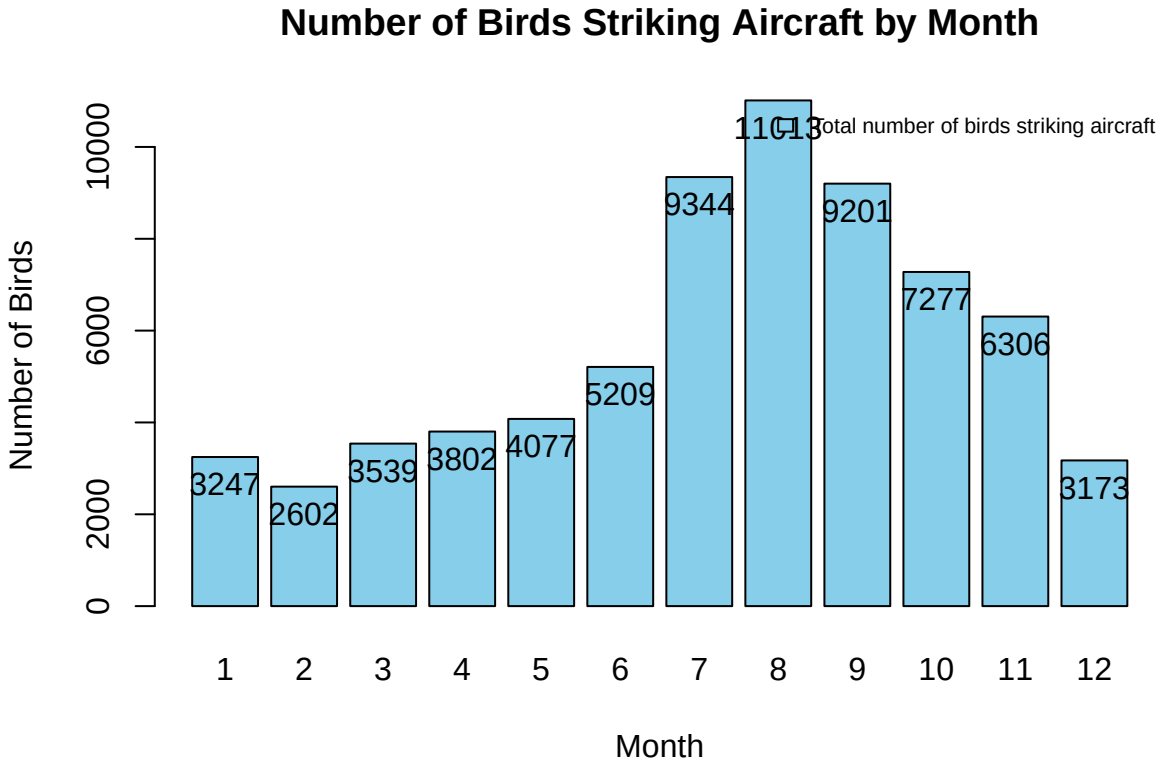
```
head(df.strikesByMonth)
```

```
##   month total_birds
## 1     1        3247
## 2     2        2602
## 3     3        3539
## 4     4        3802
## 5     5        4077
## 6     6        5209
```

```
barplot(df.strikesByMonth$total_birds,
        names.arg = df.strikesByMonth$month,
        xlab = "Month", ylab = "Number of Birds",
        main = "Number of Birds Striking Aircraft by Month",
        col = "skyblue")

# Adding data labels
text(x = barplot(df.strikesByMonth$total_birds, col = "skyblue", plot = FALSE),
     y = df.strikesByMonth$total_birds + 5,  # Adjust the 5 for proper positioning
     labels = df.strikesByMonth$total_birds, pos = 1)

# Adding a legend-like text annotation
legend("topright", legend = "Total number of birds striking aircraft",
       fill = "skyblue", bty = "n", cex = 0.65)
```




```
DROP PROCEDURE IF EXISTS addStrike;
```

```
CREATE PROCEDURE addStrike(
  IN sp_numBirds INT,
  IN sp_impact TEXT,
  IN sp_altitude INT,
  IN sp_fid INT,
  IN sp_flight_date DATE,
  IN sp_airline VARCHAR(100),
  IN sp_aircraft VARCHAR(50),
  IN sp_airportName VARCHAR(100),
  IN sp_airportState VARCHAR(50),
  IN sp_skyCondition VARCHAR(50),
  IN sp_heavy TINYINT(1),
  IN sp_damage TINYINT(1)
)
BEGIN
  DECLARE fid_procedure INT DEFAULT NULL;
  DECLARE aid_procedure INT DEFAULT NULL;
  DECLARE cid_procedure INT DEFAULT NULL;
  DECLARE sid_procedure INT DEFAULT NULL;
  DECLARE aid_max INT DEFAULT NULL;
  DECLARE fid_max INT DEFAULT NULL;
  DECLARE cid_max INT DEFAULT NULL;
  DECLARE sid_max INT DEFAULT NULL;

  IF sp_airportState IS NOT NULL THEN
    SELECT aid INTO aid_procedure FROM airports
    WHERE airportState = sp_airportState AND airportName = sp_airportName;
```

```
    /* If the airportState does not exist, insert it and grab the aid */
    IF aid_procedure IS NULL THEN
      SELECT COALESCE(MAX(aid), 1) INTO aid_max FROM airports;
      SET aid_procedure = aid_max + 1;
      INSERT INTO airports (aid, airportName, airportState)
      VALUES (aid_procedure, sp_airportName, sp_airportState);
    END IF;
  END IF;

  /* Check if a skyCondition exists */
  IF sp_skyCondition IS NOT NULL THEN
    SELECT cid INTO cid_procedure FROM conditions WHERE sky_condition = sp_skyCondition;
    /* Handle if the cid doesn't exist */
    IF cid_procedure IS NULL THEN
      SELECT COALESCE(MAX(cid), 1) INTO cid_max FROM conditions;
      SET cid_procedure = cid_max + 1;
      INSERT INTO conditions (cid, sky_condition) VALUES (cid_procedure, sp_skyCondition);
    END IF;
  END IF;

  /* Check if the fid already exists */
  IF sp_fid IS NOT NULL THEN
    SELECT fid INTO fid_procedure FROM flights WHERE fid = sp_fid;
    /* Handle if fid doesn't exist */
    IF fid_procedure IS NULL THEN
      SELECT COALESCE(MAX(fid), 1) INTO fid_max FROM flights;
      SET fid_procedure = fid_max + 1;
      INSERT INTO flights (fid, flight_date, aircraft, airline, origin_aid, heavy)
      VALUES (fid_procedure, sp_flight_date, sp_aircraft,
      UPPER(sp_airline), aid_procedure, sp_heavy);
    END IF;
  END IF;

  /* Check if sid_procedure exists */
  IF fid_procedure IS NOT NULL AND cid_procedure IS NOT NULL
  AND sid_procedure IS NOT NULL THEN
    SELECT sid INTO sid_procedure FROM strikes
    WHERE fid = sp_fid AND cid = cid_procedure;
    /* Handle if sid doesn't exist */
    IF sid_procedure IS NULL THEN
      SELECT COALESCE(MAX(sid), 1) INTO sid_max FROM strikes;
      SET sid_procedure = sid_max + 1;
      INSERT INTO strikes (sid, numbirds, damage, impact, altitude, cid, fid)
      VALUES (sid_procedure, sp_numBirds, sp_damage,
      sp_impact, sp_altitude, cid_procedure, fid_procedure);
    END IF;
  END IF;

END;
```

```
# Test Case 1: Insert a New Bird Strike with New Airport and New Flight
dbSendQuery(dbcon, "
  CALL addStrike(
    8, 'Tail damage', 30, NULL, '2023-10-25', 'Delta Airlines', 'Airplane XYZ',
```

```
    'ABC Airport', 'XYZ State', 'Some Clouds', 'Yes', 'Caused damage'
  )
")
```

```
## <MySQLResult:804232672,0,41>
```

```
# Test Case 2: Insert a New Bird Strike with Existing Airport and Existing Flight
dbSendQuery(dbcon, "
  CALL addStrike(
    5, 'Wing damage', 25, 101, '2023-10-26', 'American Airlines', 'Airplane ABC',
    'MNO Airport', 'PQRS State', 'Clear', 'No', 'No damage'
  )
")
```

```
## <MySQLResult:1334358472,0,42>
```

```
# Test Case 3: Insert a New Bird Strike with Existing Airport and New Flight
dbSendQuery(dbcon, "
  CALL addStrike(
    7, 'Engine shutdown', 28, NULL, '2023-10-27', 'United Airlines', 'Airplane XYZ',
    'ABC Airport', 'XYZ State', 'Overcast', 'Yes', 'Caused damage'
  )
")
```

```
## <MySQLResult:1334360040,0,43>
```

```
# Test Case 4: Insert a New Bird Strike with New Airport and Existing Flight
dbSendQuery(dbcon, "
  CALL addStrike(
    6, 'Fuselage damage', 22, 102, '2023-10-28', 'Delta Airlines', 'Airplane XYZ',
    'New Airport', 'New State', 'Some Clouds', 'No', 'No damage'
  )
")
```

```
## <MySQLResult:536142704,0,44>
```

```
SELECT * FROM flights
WHERE airline = 'Delta Airlines'
```

Table 11: 1 records

fid	flight_date	origin_aid	airline	aircraft	heavy
321911	2023-10-28	25661	DELTA AIRLINES	Airplane XYZ	0

```
SELECT * FROM flights
WHERE flight_date = '2023-10-26'
```

Table 12: 1 records

fid	flight_date	origin_aid	airline	aircraft	heavy
321910	2023-10-26	25660	AMERICAN AIRLINES	Airplane ABC	0

```
SELECT * FROM airports
WHERE airportName = "ABC Airport"
```

Table 13: 1 records

aid	airportName	airportState	airportCode
25659	ABC Airport	XYZ State	NA

```
SELECT * FROM airports
WHERE airportName = "New Airport"
```

Table 14: 1 records

aid	airportName	airportState	airportCode
25661	New Airport	New State	NA

```
dbDisconnect(dbcon)
```

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