Bird Strikes on Aircraft Project

CS 5200

Fall 2023

Practicum 1

Group 5: Mingxi Li,

Si Wu,

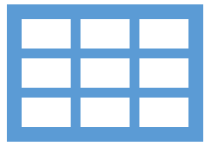
Zheng Zheng,

Jiawei Zhou

Content

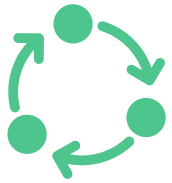
- Introduction (Zheng)
- Database Creation (Zheng)
- Data Implementation (Mingxi Li)
- Data Analysis
  - Part 1 (Si Wu)
  - Part 2 (Jiawei Zhou)
- Conclusion (Jiawei Zhou)

Introduction



Input

CSV file with 26 column.



Method&Tools

Conceptual & Logic Schema Lucid

RStudio Create Database MySQL

RStudio Data Preprocessing

RStudio Data Loading

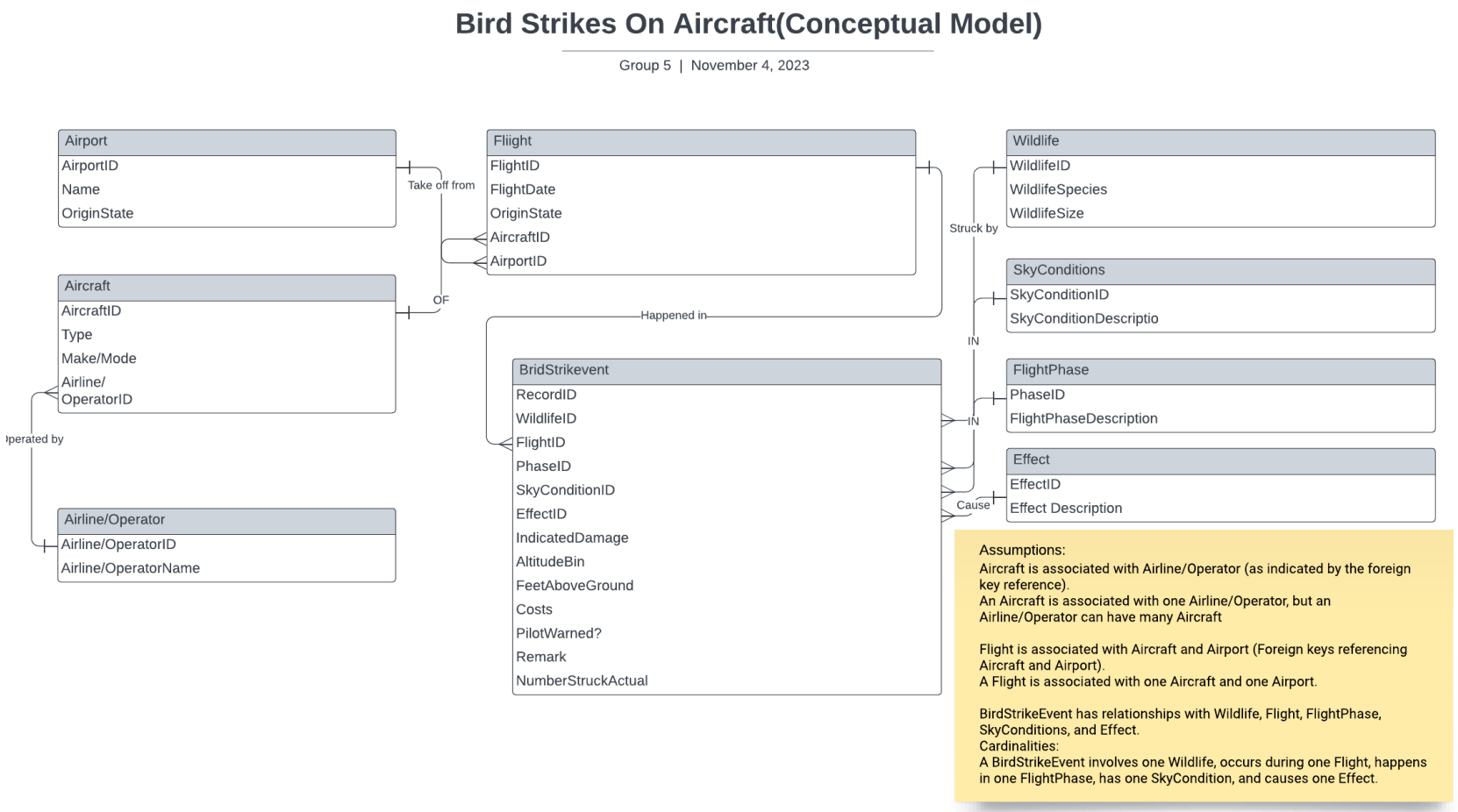
Output

A 2NF database

Some analysis based on the database

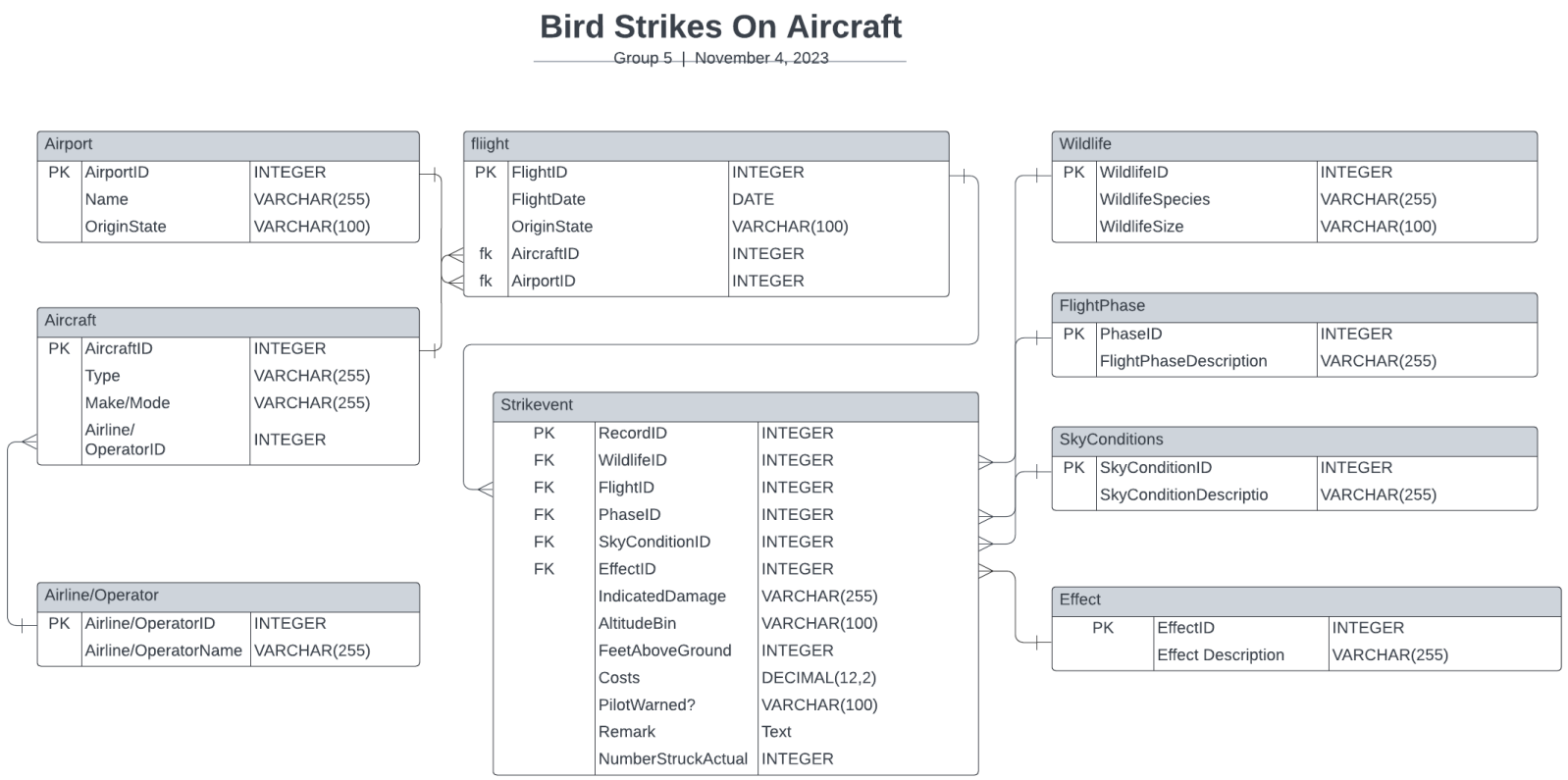
RStudio

Database Creation
-Conceptual Model/Schema



- 9 Entities
  - 1. Aircraft
  - 2. Airport
  - 3. Flight
  - 4. FlightPhase
  - 5. SkyConditions
  - 6. Wildlife
  - 7. BirdStrikeEvent
  - 8. Effect
  - 9. Airline/Operator
- Assumptions:
  - Aircraft is associated with Airline/Operator (as indicated by the foreign key reference).
  - An Aircraft is associated with one Airline/Operator, but an Airline/Operator can have many Aircraft
  - Flight is associated with Aircraft and Airport
  - A Flight is associated with one Aircraft and one Airport.
  - BirdStrikeEvent has relationships with Wildlife, Flight, FlightPhase, SkyConditions, and Effect.

Database Creation
Logical Schema



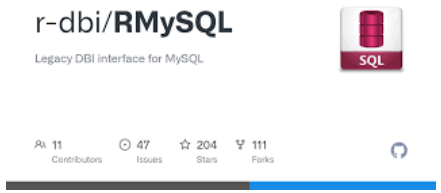
- Deleted 5 irrelevant attributes
  - Number_of_Engines
  - IsLarge
  - RemainsCollected
  - RemainsSent
  - Remark
  - NumberStruckRange

- Set most attributes as Character datatype with some limitation except for the IDs or Dates.
  - Name for Airport: VARCHAR(100)

Database Creation

-R codes for Database Creation

- Connect to MySQL sever




```r
{r}
# Load the required library
library(RMySQL)

# Create a connection
con <- dbConnect(MySQL(), user='root',
                 password='', host='localhost')

```

- Create database

```r
# Create the database
dbGetQuery(con, "DROP DATABASE IF EXISTS BirdStrikesOnAircraft")
dbGetQuery(con, "CREATE DATABASE BirdStrikesOnAircraft")

# Use the created database
dbGetQuery(con, "USE BirdStrikesOnAircraft")
```

- Create table

```r
# 1. Aircraft
dbGetQuery(con, "DROP TABLE IF EXISTS Aircraft")
dbGetQuery(con, "CREATE TABLE Aircraft(
    AircraftID INTEGER AUTO_INCREMENT PRIMARY KEY,
    Type VARCHAR(255),
    Make_Model VARCHAR(255),
    Number_of_Engines INTEGER,
    Airline_OperatorID INTEGER,
    IsLarge VARCHAR(255),
    FOREIGN KEY (Airline_OperatorID) REFERENCES Airline_Operator
(Airline_OperatorID)
)")
```

Take-away:
Always create the tables with no reference key first.

Database Implementation

- Step 1: Load Data from CSV Files

```
# Load data from CSV
csv_data <- read.csv('BirdStrikesData.csv')
```

- Step 2: Data Preprocessing
  Rename Variables to Meet Database Naming Conventions

- Data Type Conversion:
  - Integer Conversion,
  - String Conversion,
  - Date and Time Conversion,
  - String to Integer Conversion

```
# Aircraft Table
csv_data$Type <- as.character(csv_data$Type)
csv_data$Make_Model <- as.character(csv_data$Make_Model)
```

```
# Flight Table
csv_data$FlightDate <- as.Date(csv_data$FlightDate, format="%m/%d/%Y")
```

```
# BirdStrikeEvent Table
csv_data$IndicatedDamage <- ifelse(csv_data$IndicatedDamage == 'Caused damage', 1, 0)
```

Database Implementation

- Step 2: Data Preprocessing
  - Handle Missing Values
    - Identify categorical data columns and replace missing values in categorical columns with 'Unknown'.
    - Identify numerical data columns and replace missing values in numerical columns with 0.
    - Replace non-date values with 1900-01-01 in the "FlightDate" column.

```r
# Identify columns with categorical data
categorical_cols <- c(
  "Type", "AirportName", "AltitudeBin", "Make_Model", "EffectDescription",
  "OriginState", "FlightPhaseDescription", "Precipitation", "RemainsCollected",
  "RemainsSent", "Remarks", "WildlifeSize", "SkyConditionDescription", "WildlifeSpecies",
  "PilotWarned", "IsLarge", "Airline_OperatorName", "NumberStruckRange"
)

# Replace NA values with 'Unknown' for these columns
csv_data[categorical_cols] <- lapply(csv_data[categorical_cols], function(col) {
    ifelse(is.na(col)| col == "", 'Unknown', col)
})

# Identify columns with numerical data
numerical_cols <- c(
  "RecordID", "NumberStruckActual", "NumberOfEngines", "Costs",
  "FeetAboveGround", "NumberOfPeopleInjured"
)

# Replace NA values with 0 for these columns
csv_data[numerical_cols] <- lapply(csv_data[numerical_cols], function(col) {
    ifelse(is.na(col)| col == "", 0, col)
})
# Replace 'Unknown' or other non-date values in FlightDate with '1900-01-01'
csv_data$FlightDate[csv_data$FlightDate == ""  | !grepl("^\\d{4}-\\d{2}-\\d{2}$", csv_data$FlightDate)] <-
'1900-01-01'
```

Take-away: Use regular expression to verify if the date conforms to the specified format, which is YYYY-MM-DD (year-month-day).

Database Implementation

- Step 3: Create Data Table Identifiers and Merge with Original Data Frame

```
# Airline/Operator IDs
airline_operator_ids <- unique(csv_data$Airline_OperatorName)
airline_operator_ids <- data.frame(Airline_OperatorID = seq_along(airline_operator_ids),
                                   Airline_OperatorName = airline_operator_ids)
```

Take-away: Use "seq_along" function to assign a unique identifier to each distinct entity. These identifiers are integers that start from 1 and increment sequentially.

```
# Merge Airline/Operator IDs
csv_data <- merge(csv_data, airline_operator_ids, by="Airline_OperatorName", all.x = TRUE)
```

Database Implementation

- Step 4: Data Insertion

```
#1. Airline/Operator
for(i in 1:nrow(airline_operator_ids)) {
  query <- sprintf("INSERT INTO Airline_Operator (Airline_OperatorID, Airline_OperatorName) VALUES (%d, '%s')",
                   airline_operator_ids$Airline_OperatorID[i], airline_operator_ids$Airline_OperatorName[i])
  dbGetQuery(con, query)
}
```

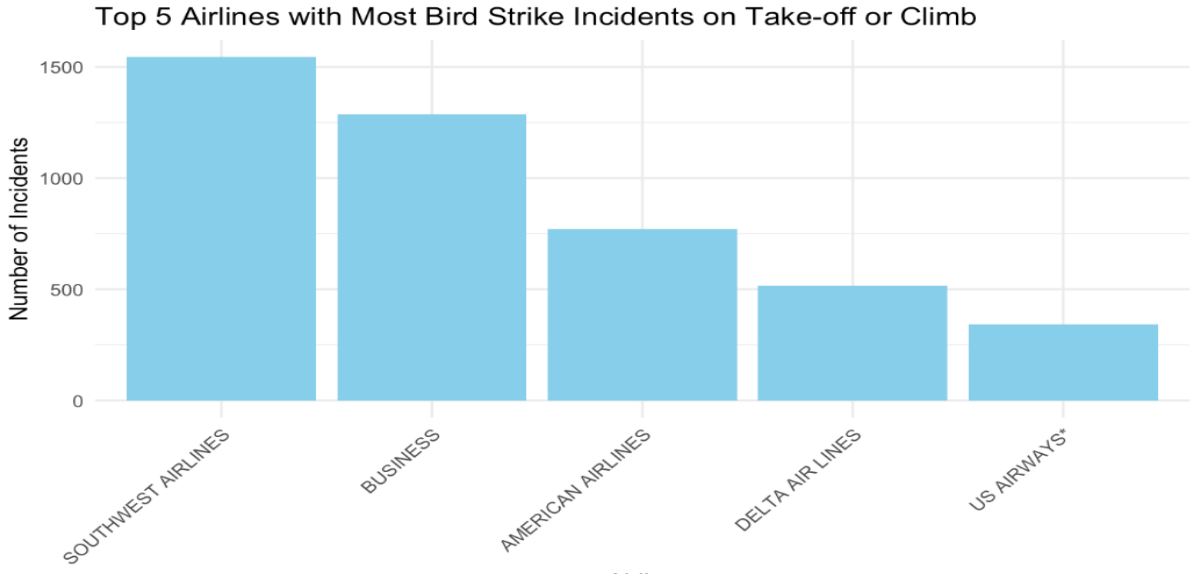
Take-away: The sprintf function is a powerful tool for string formatting. It could create string templates with placeholders and insert actual variable values into these placeholders to generate the final string

The Number of Bird Strike Incidents for each Airline upon Take-off or Climb

```sql
SELECT
    ao.Airline_OperatorName AS Airline,
    COUNT(*) AS NumberOfIncidents
FROM
    BirdStrikeEvent bse
JOIN
    Flight f ON bse.FlightID = f.FlightID
JOIN
    Aircraft ac ON f.AircraftID = ac.AircraftID
JOIN
    Airline_Operator ao ON ac.Airline_OperatorID = ao.Airline_OperatorID
JOIN
    FlightPhase fp ON bse.PhaseID = fp.PhaseID
WHERE
    fp.FlightPhaseDescription IN ('Take-off run', 'Climb')
GROUP BY
    ao.Airline_OperatorName
ORDER BY
    NumberOfIncidents DESC;
```

Airline <chr>	NumberOfIncidents <dbl>
SOUTHWEST AIRLINES	1544
BUSINESS	1287
AMERICAN AIRLINES	771
DELTA AIR LINES	517
US AIRWAYS*	343
AMERICAN EAGLE AIRLINES	324
SKYWEST AIRLINES	282
JETBLUE AIRWAYS	240
US AIRWAYS	232
UNITED AIRLINES	192

1-10 of 210 rows Previous 1 2 3 4 5 6 ... 21 Next



The Airport that Had the Most Bird Strike Incidents

```sql
SELECT
    ap.AirportName AS Airport,
    COUNT(*) AS NumberOfBirdStrikes
FROM
    BirdStrikeEvent bse
JOIN
    Flight f ON bse.FlightID = f.FlightID
JOIN
    Airport ap ON f.AirportID = ap.AirportID
GROUP BY
    ap.AirportName
ORDER BY
    NumberOfBirdStrikes DESC

LIMIT 1;
```

Airport	NumberOfBirdStrikes
<chr>	<dbl>
DALLAS/FORT WORTH INTL ARPT	803

1 row

The Number of Bird Strike Incidents by Year

```
SELECT
    YEAR(f.FlightDate) AS Year,
    COUNT(*) AS NumberOfStrikes
FROM
    Flight f
JOIN
    Aircraft a ON f.AircraftID = a.AircraftID
JOIN
    Airline_Operator ao ON a.Airline_OperatorID = ao.Airline_OperatorID
JOIN
    BirdStrikeEvent bse ON f.FlightID = bse.FlightID

WHERE
    YEAR(f.FlightDate) <> 1900
GROUP BY
    Year
ORDER BY
    Year;
```

Year <int>	NumberOfStrikes <dbl>
2000	1367
2001	1230
2002	1681
2003	1568
2004	1692
2005	1853
2006	2159
2007	2301
2008	2258
2009	3247

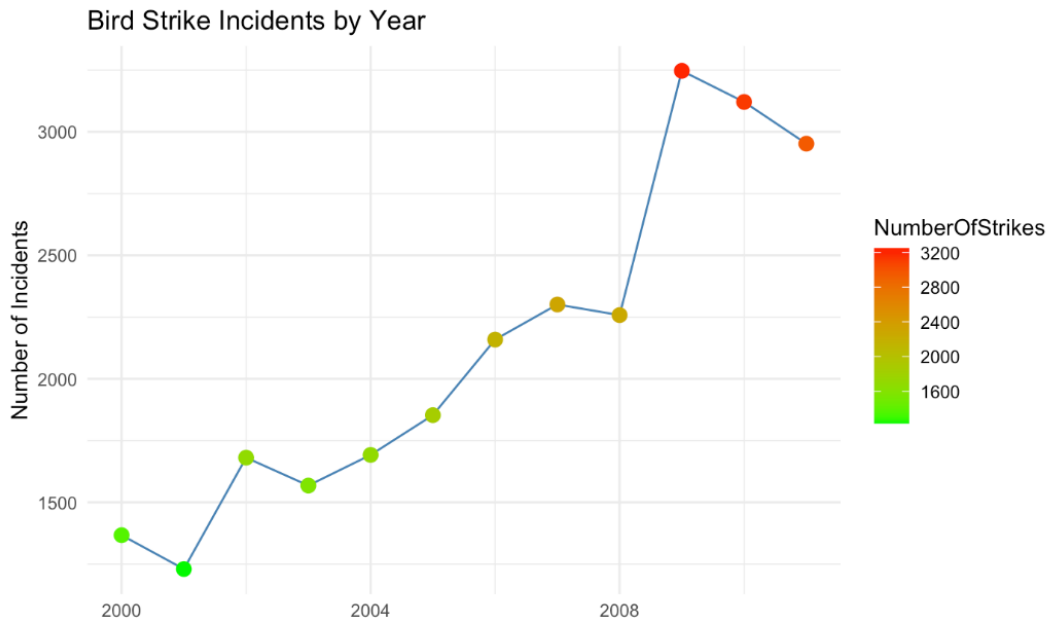
1-10 of 12 rows

Previous 1 2 Next

SQL: Visualization

```
# Plot for bird strike incidents by year from Question 6
ggplot(df6, aes(x = Year, y = NumberOfStrikes)) +
  geom_line(color="steelblue") +
  geom_point(aes(color = NumberOfStrikes), size = 3) +
  scale_color_gradient(low = "green", high = "red") +
  labs(title = "Bird Strike Incidents by Year",
       y = "Number of Incidents",
       x = "Year") +
  theme_minimal()
```

Trend of Bird Strike Incidents by Year



Visualization of The Number of Bird Strikes Incidents per Year from 2008 to 2011 during take-off/climbing and during descent/approach/landing

```
SELECT
    YEAR(f.FlightDate) AS Year,
    CASE
        WHEN fp.FlightPhaseDescription IN ('Take-off run', 'Climb') THEN 'Group 1: Take-off & Climbing'
        WHEN fp.FlightPhaseDescription IN ('Descent', 'Approach', 'Landing Roll') THEN 'Group 2: Descent, Approac
h & Landing'
    END AS FlightPhaseGroup,
    COUNT(*) AS NumberOfIncidents
FROM
    BirdStrikeEvent bse
JOIN
    Flight f ON bse.FlightID = f.FlightID
JOIN
    FlightPhase fp ON bse.PhaseID = fp.PhaseID
WHERE
    YEAR(f.FlightDate) BETWEEN 2008 AND 2011
    AND fp.FlightPhaseDescription IN ('Take-off run', 'Climb', 'Descent', 'Approach', 'Landing Roll')
GROUP BY
    YEAR(f.FlightDate),
    FlightPhaseGroup
ORDER BY
    Year, FlightPhaseGroup;
```

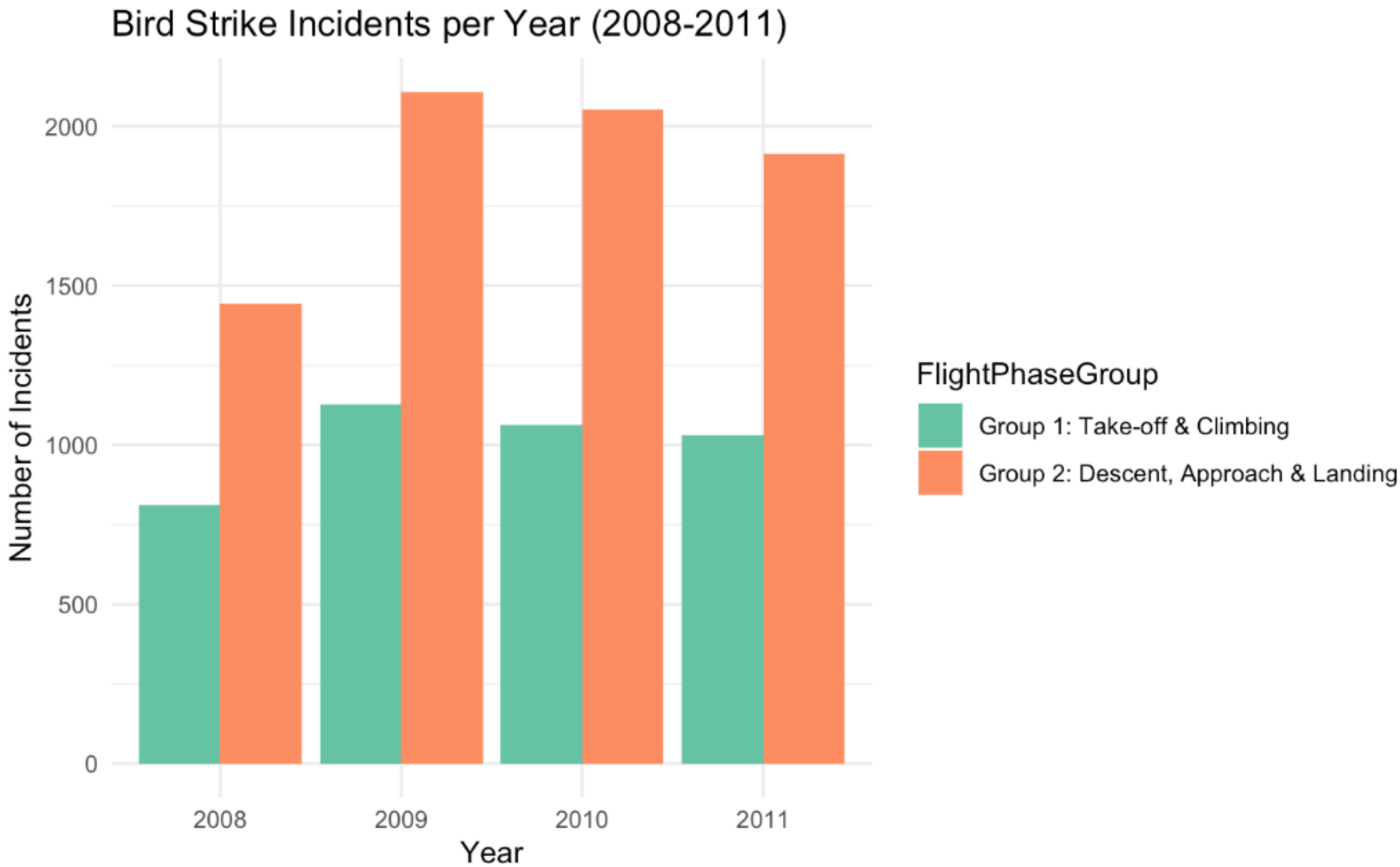
Year	FlightPhaseGroup	NumberOfIncidents
<int>	<chr>	<dbl>
2008	Group 1: Take-off & Climbing	810
2008	Group 2: Descent, Approach & Landing	1442
2009	Group 1: Take-off & Climbing	1127
2009	Group 2: Descent, Approach & Landing	2109
2010	Group 1: Take-off & Climbing	1062
2010	Group 2: Descent, Approach & Landing	2053
2011	Group 1: Take-off & Climbing	1030
2011	Group 2: Descent, Approach & Landing	1913

```
library(ggplot2)
library(DBI)

query <- "
SELECT
    YEAR(f.FlightDate) AS Year,
    CASE
        WHEN fp.FlightPhaseDescription IN ('Take-off run', 'Climb') THEN 'Group 1: Take-off & Climbing'
        WHEN fp.FlightPhaseDescription IN ('Descent', 'Approach', 'Landing Roll') THEN 'Group 2: Descent, Approac
h & Landing'
    END AS FlightPhaseGroup,
    COUNT(*) AS NumberOfIncidents
FROM
    BirdStrikeEvent bse
JOIN
    Flight f ON bse.FlightID = f.FlightID
JOIN
    FlightPhase fp ON bse.PhaseID = fp.PhaseID
WHERE
    YEAR(f.FlightDate) BETWEEN 2008 AND 2011
    AND fp.FlightPhaseDescription IN ('Take-off run', 'Climb', 'Descent', 'Approach', 'Landing Roll')
GROUP BY
    YEAR(f.FlightDate),
    FlightPhaseGroup
ORDER BY
    Year, FlightPhaseGroup;
"

data <- dbGetQuery(con, query)

ggplot(data, aes(x=factor(Year), y=NumberOfIncidents, fill=FlightPhaseGroup)) +
  geom_bar(stat="identity", position="dodge") +
  labs(title="Bird Strike Incidents per Year (2008-2011)",
       x="Year",
       y="Number of Incidents") +
  scale_fill_brewer(palette="Set2") +
  theme_minimal()
```


Bird Strike Incidents per Year (2008-2011)
Number of Incidents
2000
1500
1000
500
0
2008
2009
2010
2011
Year
FlightPhaseGroup
Group 1: Take-off & Climbing
Group 2: Descent, Approach & Landing

A Stored Procedure that Removes a Bird Strike Incident from the Database

```
CREATE PROCEDURE DeleteBirdStrikeEvent(IN incidentRecordID INT)
BEGIN
  DELETE FROM BirdStrikeEvent WHERE RecordID = incidentRecordID;
END
```

```
SELECT * FROM BirdStrikeEvent WHERE RecordID = 1195;
```

RecordID <int>	WildlifeID <int>	FlightID <int>	PhaseID <int>	SkyConditionID <int>	EffectID <int>	IndicatedDamage <int>	AltitudeBin <chr>
1195	14	747	3	3	2	0	> 1000 ft

1 row | 1-8 of 17 columns

```
CREATE PROCEDURE DeleteBirdStrikeEvent(IN incidentRecordID INT)
BEGIN
  DELETE FROM BirdStrikeEvent WHERE RecordID = incidentRecordID;
END
```

```
CALL DeleteBirdStrikeEvent(1195);
```

```
SELECT * FROM BirdStrikeEvent WHERE RecordID = 1195;
```

0 rows | 1-8 of 17 columns

Conclusion

- Flight Phase Risks: The data reveals that the phases of “Descent, Approach & Landing” (Group 2) consistently experience more bird strike incidents than the “Take-off & Climbing” phases (Group 1). This heightened risk during descent and landing emphasizes the importance of reinforced safety measures during these specific phases.
- Airline Operator Incidence: Certain airline operators experienced a higher number of incidents during the ‘Take-Off’ and ‘Climb’ phases compared to others. This suggests that some airlines might need to review their preventive measures or routes to ensure greater safety.
- Airport Vulnerability: Our study also highlighted that some airports reported a notably higher number of bird strike incidents. These airports, in particular, may benefit from enhanced bird management and deterrent strategies.


Thank you