INTRODUCTION

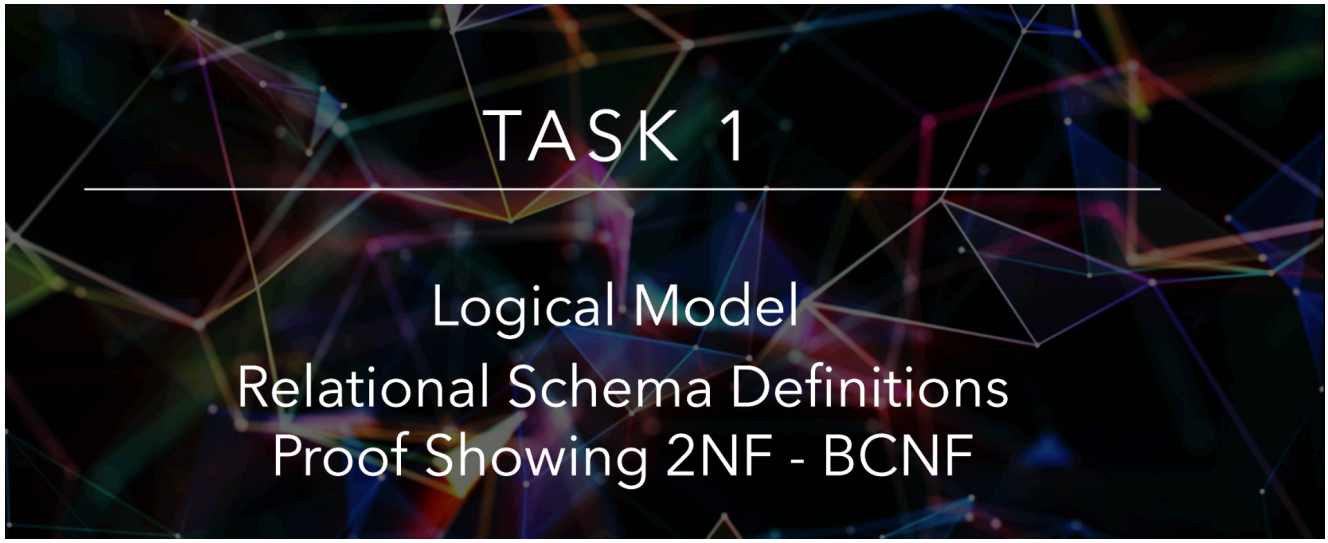
Wildlife Aircraft Collision App Design

Notes & Assumptions:

Below the key assumptions of the Bird Strike Application Database/Models are listed.

As I was designing this project I designed the app as a front end and a back end. I focused on the backend first to ensuring the functionality of the app. As I have more time I will further develop the app. I have provided both a front end and back end ERD, and the backend which I focused for functionality.

I have modeled my bird strike app in order to allow the users, there is only one user, for now and that is the pilot to add bird strike incidents to the app in order to inform and update airlines/airports to improve on the overall safety of flights.



RELATIONAL SCHEMA DEFINITIONS

Relational Schema Definitions for Front and Back End Model Diagram

The logical schemas shown above is defined below. A relation R with attributes A1, A2, A3…An where R(A1, A2, A3 …An), where the primary key is underlined and foreign keys are show in *italics.*

PilotLoginInfo (pilotID, userName, userPassword)

PilotInfo (*pilotID,* pInfoID, name, gender, DOB)

Airline (airlineID, airlineName, *pilotID*, *airportID*)

Airport (airportID, airportName)

Flight (flightNumID, *airlineID,* originState, damage, *flightTrackerID*)

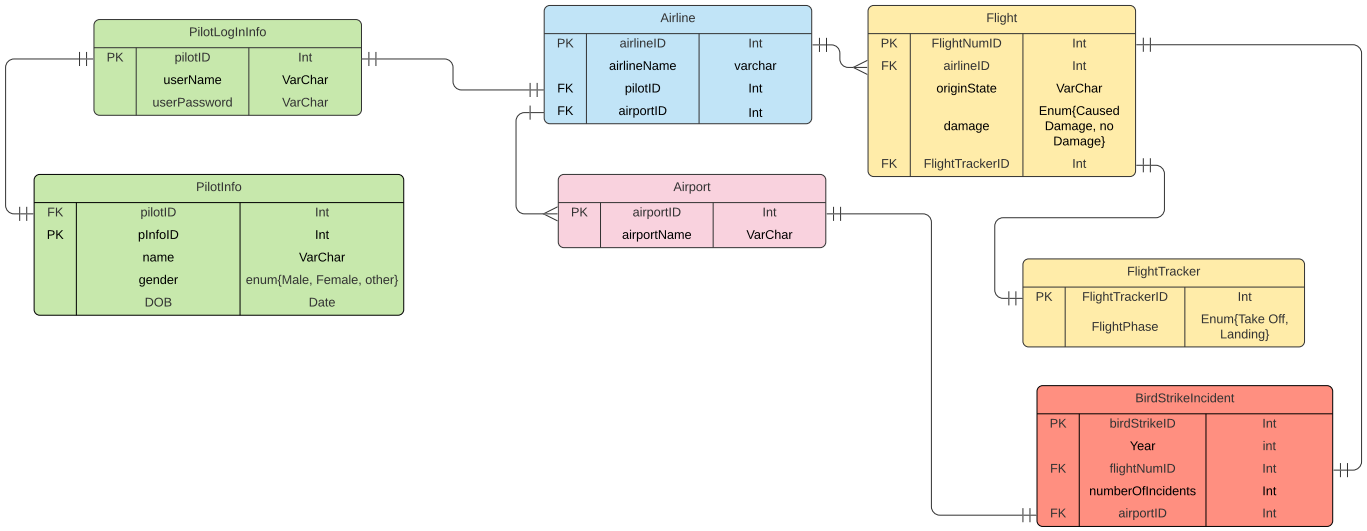
FlightTracker (flightTrackerID, flightPhase)

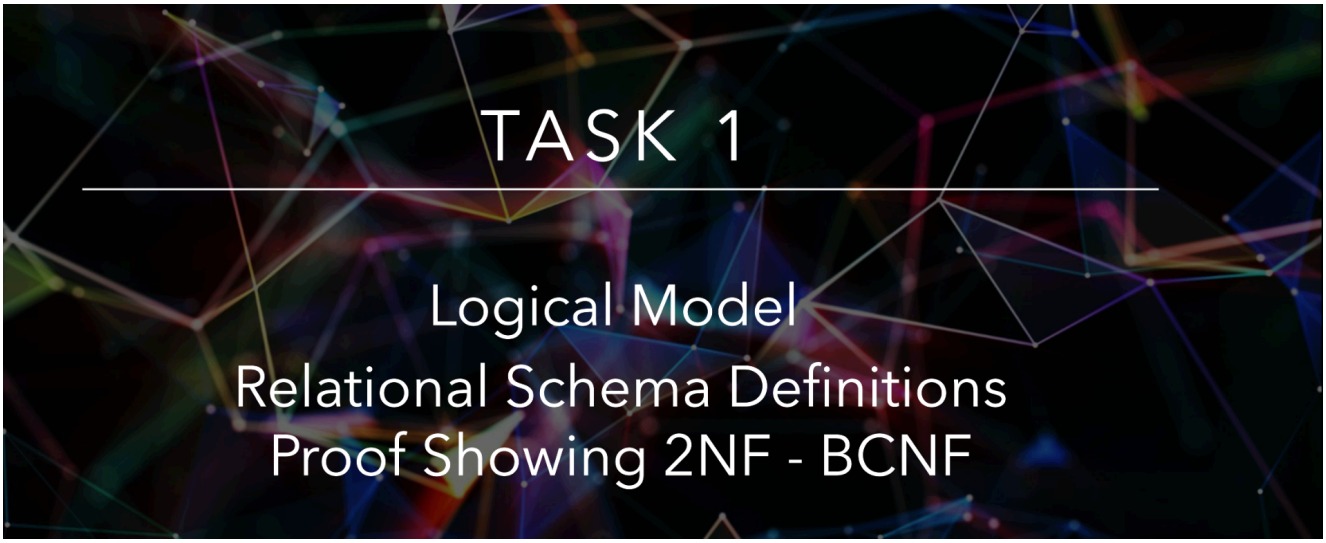
BirdStrikeIncident(birdStrikeID, year, *flightNumID,* numberOfIncidents, *airportID*)

Schreiner_Practicum1

Dominique Schreiner |

Front End & Back End ERD





For the Practicum I focused on implementing the back end, due to time constraints. I used the following Schema with the tables, focusing on functionality.

RELATIONAL SCHEMA DEFINITIONS

Relational Schema Definitions for Front and Back End Model Diagram

The logical schemas shown above is defined below. A relation R with attributes A1, A2, A3…An where R(A1, A2, A3 …An), where the primary key is underlined and foreign keys are show in *italics.*

Airline (airLineID, airlineName)

BirdStrikeIncident (*airlineID*, *airportID,* Year, *FlightPhaseID,* StrikeNum)

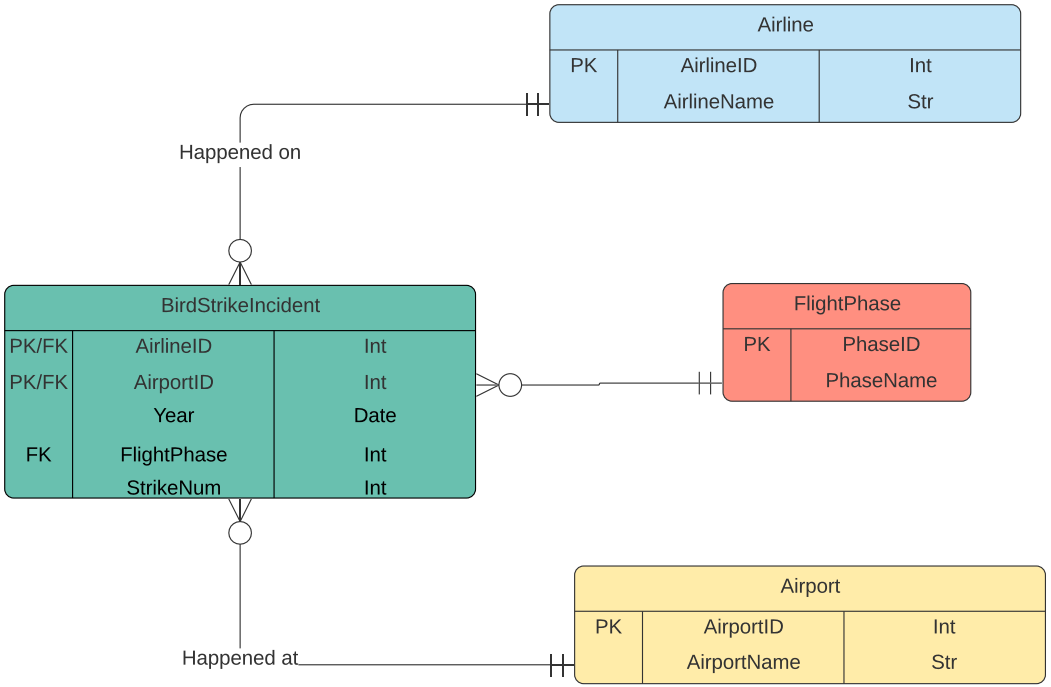
FlightPhase (phaseID, phaseName)

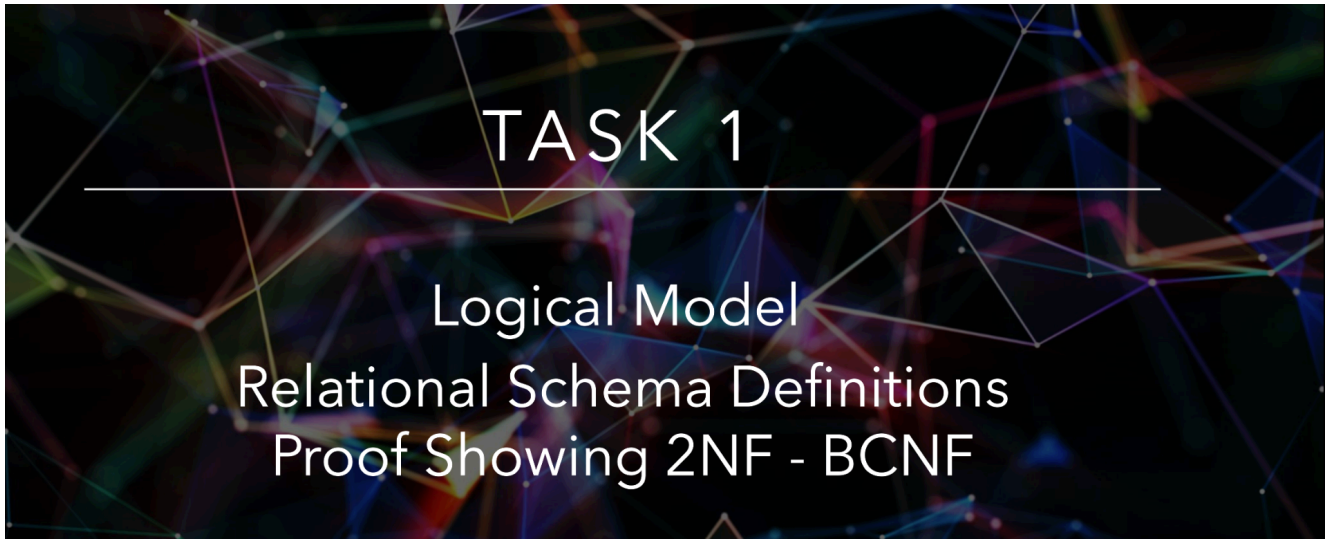
Airport (airportID, airportName)

Schreiner_Practicum1

Dominique Schreiner |

Back End ERD





RELATIONAL SCHEMA DEFINITIONS - Back End ERD

A relational schema is considered to be in BCNF for every one of its dependencies X -> Y one of the following conditions must be met:

- X → Yis a tribal functional dependency (i.e Y is a subset of X) (be in 3NF no transitive relations)
- X is a superkey for the schema

A relational schema is considered to be in 2NF it dependencies must meet the following criteria:

- Follow 1NF
- Have no partial dependencies

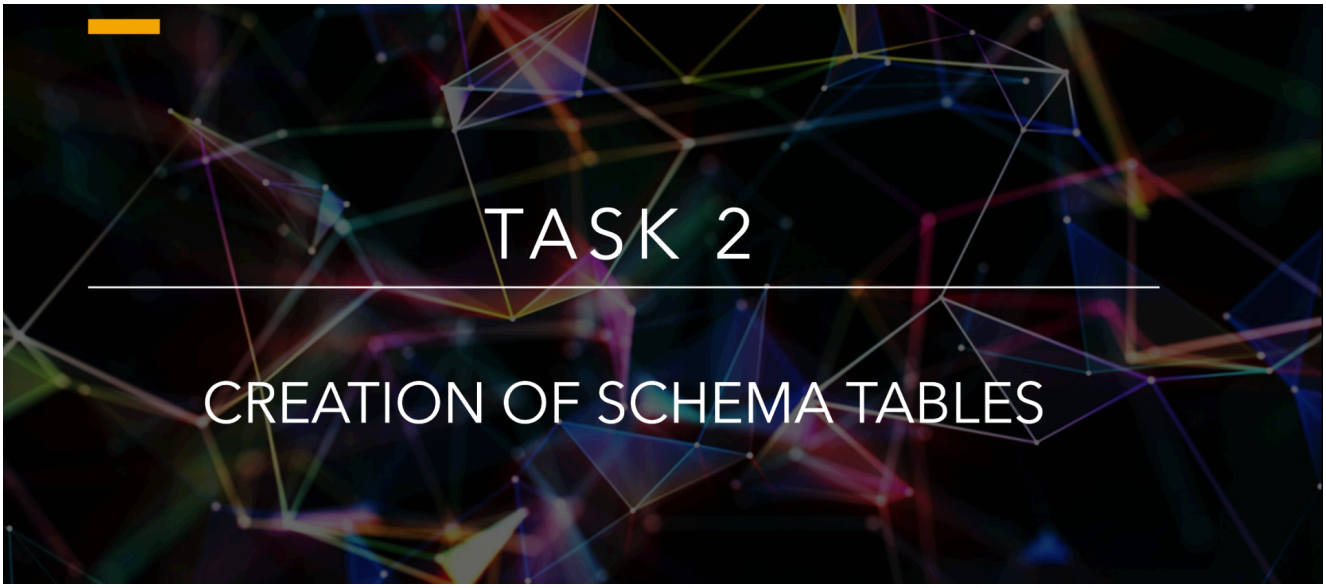
All below meet 2NF

Airline
airLineID →{ airlineName}

BirdStrikeIncident
airlineID, airportID form a composite primary key, therefore tribal FD hence in BCNF

FlightPhase
phaseID → { phaseName}

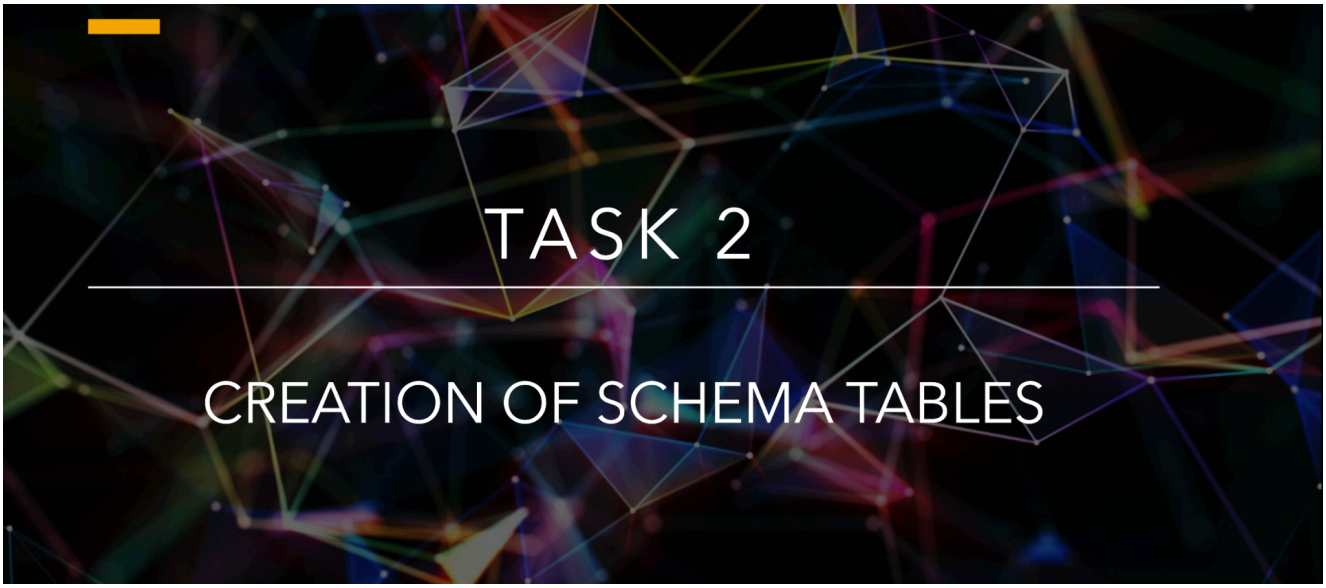
Airport
airportID → {airportName}



I begun to create the tables for both the front end and back end model as seen above. As I was working to expand my app. I performed this in MySQL Workbench.

```
create table PilotLoginInfo(
pilotID int primary key,
userName varchar(30),
userPassword varchar(30)
);

create table PilotInfo(
pInfoID int primary key,
pilotID int,
name varchar(50),
gender enum('male', 'female', 'other'),
dob date,
foreign key (pilotID) references PilotLoginInfo(pilotID)
);

create table Airport(
airportID int primary key,
airportName varchar(50)
);

create table Airline(
airlineID int primary key,
airlineName varchar(50),
pilotID int,
airportID int,
foreign key (pilotID) references PilotLoginInfo(pilotID),
foreign key (airportID) references Airport (airportID)
);
```

```
create table FlightTracker(
flightTrackerID int primary key,
flightPhase enum('Take Off', 'Landing')
);


create table Flight (
flightNumID int primary key,
airlineID int,
originState varchar(50),
damage enum('Caused Damage', 'No Damage'),
flightTrackerID int,
foreign key (airlineID) references Airline(airlineID),
foreign key (flightTrackerID) references FlightTracker(FlightTrackerID)
);

create table BirdStrikeIncident(
birdStrikeID int primary key,
year int,
flightNumID int,
numberOfIncidents int,
airportID int,
foreign key (flightNumID) references Flight(flightNumID),
foreign key (airportID) references Airport(airportID)
);
```




```
DROP TABLE IF EXISTS
Airline;


CREATE TABLE Airline (
airlineID INT NOT NULL,
airlineName VARCHAR(50),
PRIMARY KEY (airlineID)
);

DROP TABLE IF EXISTS
Airport;

CREATE TABLE Airport (
airportID  INT NOT NULL,
airportName VARCHAR(255),
PRIMARY KEY (airportID)
);

DROP TABLE IF EXISTS FlightPhase;

CREATE TABLE FlightPhase
(PhaseID INT NOT NULL,
 PhaseName VARCHAR(20),
 PRIMARY KEY (PhaseID)
)

DROP TABLE IF EXISTS
birdstrikeincident;

CREATE TABLE birdstrikeincident (
Year DATE NOT NULL,
StrikeNum INT NOT NULL,
airportID  INT NOT NULL,
airlineID INT NOT NULL,
```

```
phaseID INT NOT NULL,
PRIMARY KEY (airportID, airlineID),
FOREIGN KEY (airportID) REFERENCES Airport (airportID),
FOREIGN KEY (airlineID) REFERENCES Airline (airlineID),
FOREIGN KEY (phaseID) REFERENCES FlightPhase (phaseID)
```

I first cleaned the dataset, selected the columns, named them accordingly, and converted the time column to a year-mm-dd format.

Then I created the data frames and populate the data into tables with dbWriteTable function.

Creating DataFrame for birdStrikeIncident Table = birdStrikesDF

```
birdStrikesDF = BirdStrikesNew %>% select(Year, StrikeNum)
n = nrow(birdStrikesDF)

airportID = vector(length = n)
airlineID = vector(length = n)
phaseID = vector(length = n)

for (i in 1:n) {

  obs = BirdStrikesNew[i,]

  Airport = obs$Airport
  Airport_ind = which(AirportDF$airportName == Airport)
  airportID[i] = AirportDF$airportID[Airport_ind]

  # print(airportID[i])

  Airline = obs$Airline
  Airline_ind = which(AirlineDF$airlineName == Airline)
  airlineID[i] = AirlineDF$airlineID[Airline_ind]

  # print(airlineID[i])

  FlightPhase = obs$FlightPhase
  Phase_ind = which(FlightPhaseDF$PhaseName == FlightPhase)
  phaseID[i] = FlightPhaseDF$PhaseID[Phase_ind]

  # print(phaseID[i])

}

birdStrikesDF$airportID = airportID
birdStrikesDF$airlineID = airlineID
birdStrikesDF$phaseID = phaseID

head(birdStrikesDF)
```

Creating DataSet to use for FlightPhase Table = FlightPhaseDF

```
FlightPhase = unique(BirdStrikesNew$FlightPhase)
n = length(FlightPhase)
PhaseID = c(1:n)

FlightPhaseDF = data.frame(PhaseID = PhaseID,
                           PhaseName = FlightPhase)
```

Creating DataSet to use for Airline Table = AirlineDF

```
airlineName = unique(BirdStrikesNew$Airline)
n = length(airlineName)
airlineID = c(1:n)

AirlineDF = data.frame(airlineID = airlineID,
                           airlineName = airlineName)
```

Creating DataSet to use for Airport Table = AirportDF

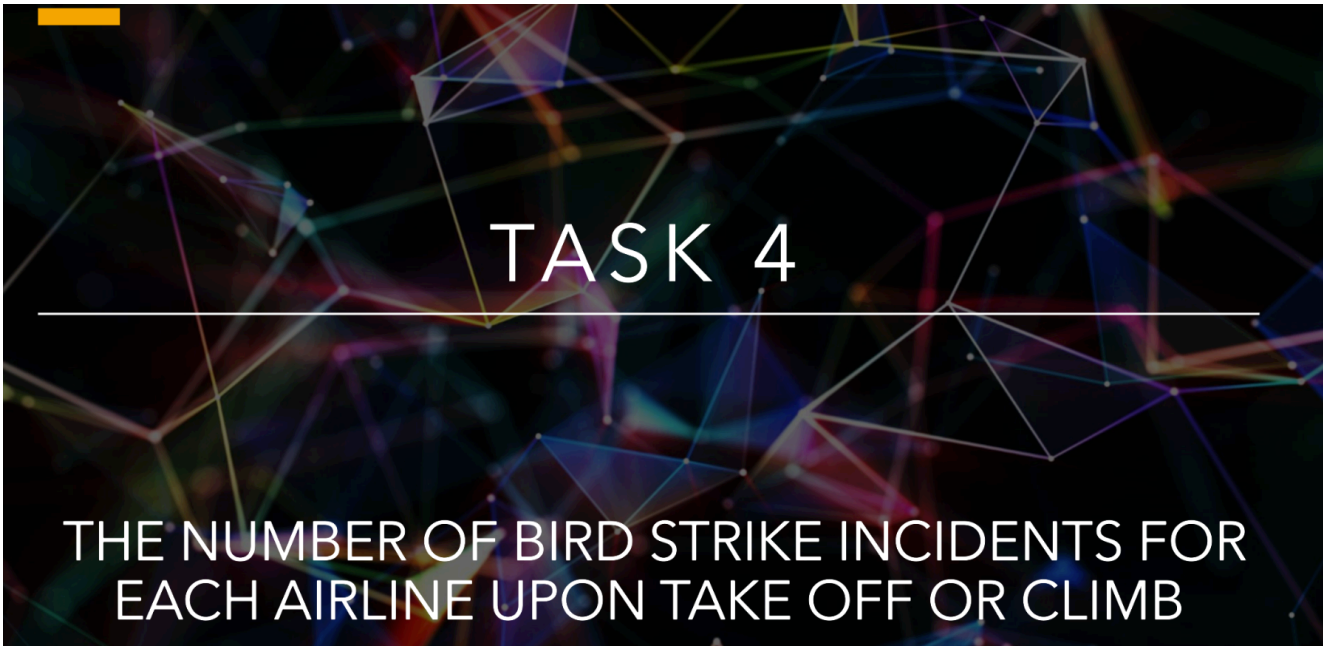
```
airportName = unique(BirdStrikesNew$Airport)
n = length(airportName)
airportID = c(1:n)

AirportDF = data.frame(airportID = airportID,
                           airportName = airportName)
```

```
RMySQL::dbWriteTable(mydb, "airport", AirportDF,
                     append=F, overwrite = T, row.names=FALSE)
```

```
RMySQL::dbWriteTable(mydb, "airline", AirlineDF,
                    append=F, overwrite = T, row.names=FALSE)
```

```
RMySQL::dbWriteTable(mydb, "flightphase", FlightPhaseDF,
                     append=F, overwrite = T, row.names=FALSE)
```

```
RMySQL::dbWriteTable(mydb, "birdstrikeincident", birdStrikesDF,
                     append=F, overwrite = T, row.names=FALSE)
```



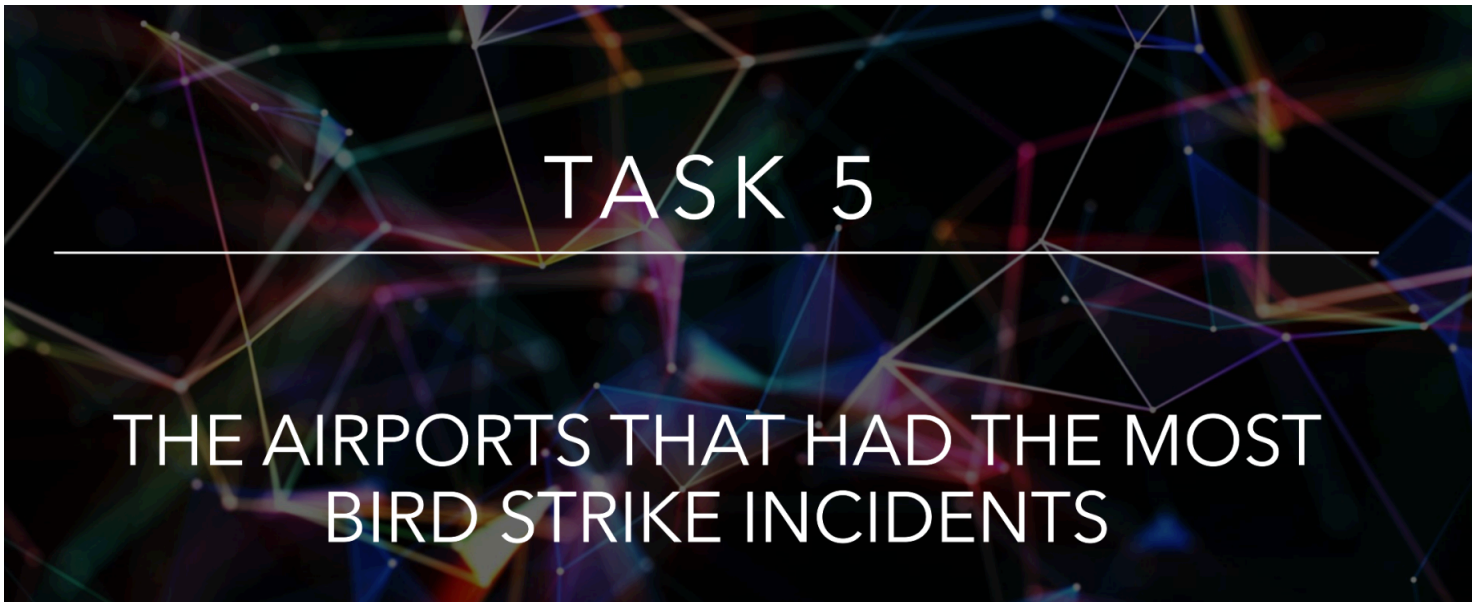
Below I created a query to find all the bird strike incidents for each airline upon take off climb. Take Off is represented with ID = 4 and Climb has a phaseID represented as ID = 1. In order to show all incidents for each record of the airlines I grouped by airlineID to show all records.

```
SELECT phaseID, airlineID, COUNT(StrikeNum)
FROM birdstrikeincident
WHERE phaseID = '1'
OR phaseID ='4'
GROUP BY airlineID;
```

Displaying records 1 - 10

phaseID	airlineID	COUNT(StrikeNum)
1	1	343
1	4	87
4	2	771
1	3	1287
4	7	170
4	12	124
1	6	192
1	13	517
1	15	87
1	16	99

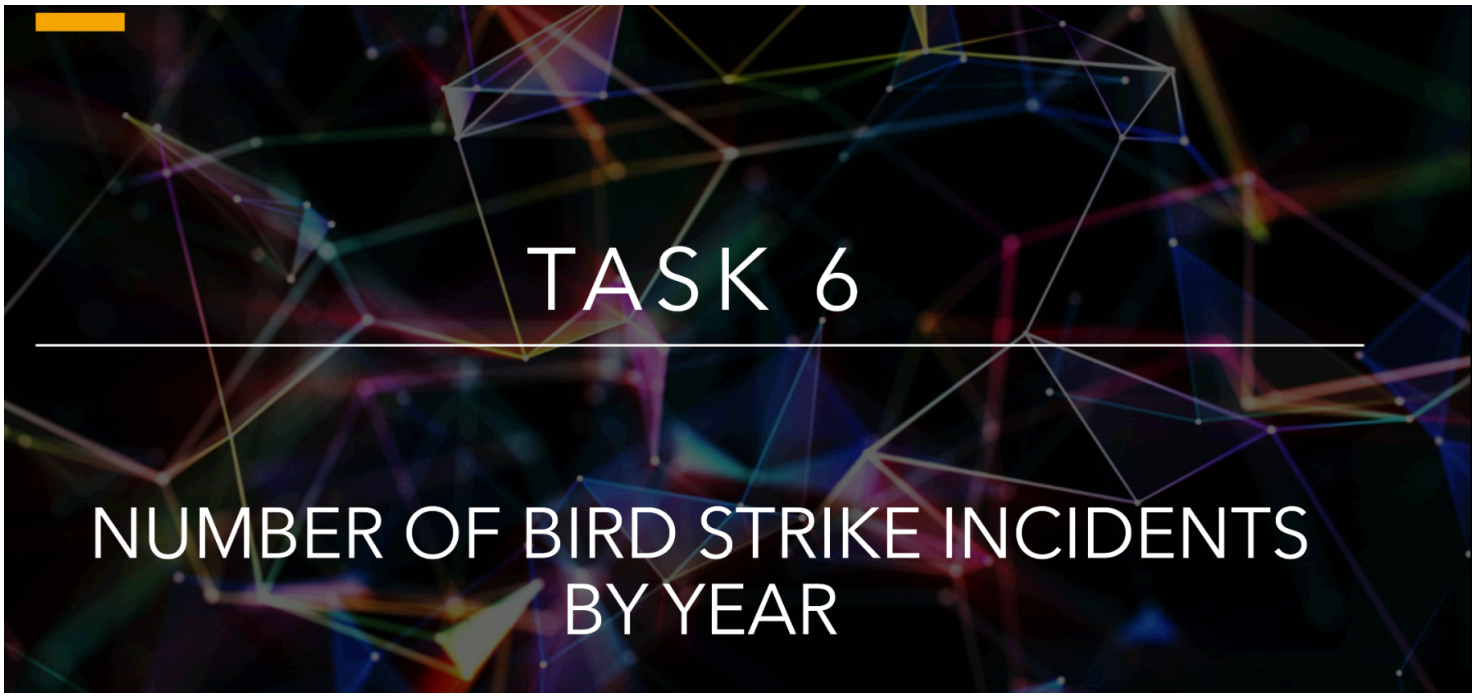
Task 5



In this query all the only information we are concerned with is the airlines with the most bird incidents, during any phase of flight. Therefore, I chose to display the airports represented with airport and then sum the number of strike occurrences per airport and displayed in descending order to demonstrate the airports with the most incidents.

Displaying records 1 - 10

airlineID	airportID	SUM(StrikeNum)	phaseID
2	2	2933	2
3	106	1856	1
1	1	1579	1
2	52	1396	3
13	91	1376	3
17	35	1210	4
47	107	1207	4
4	4	1181	1
5	5	1119	3
6	68	1027	1



In order to find every bird strike incident by the year, I used the GROUP BY Year statement. I also summed the incidents to return records of all the strikes occurring in each in the dataset.

Displaying records 1 - 10

year(Year)	SUM(StrikeNum)	airlineID
2010	7657	2
2009	7169	39
2011	7105	3
2003	6193	5
2006	5839	6
2002	5505	4
2008	5350	29
2005	5138	29
2007	5069	20
2001	4721	2

Task 7 Creating DataFrame to Plot

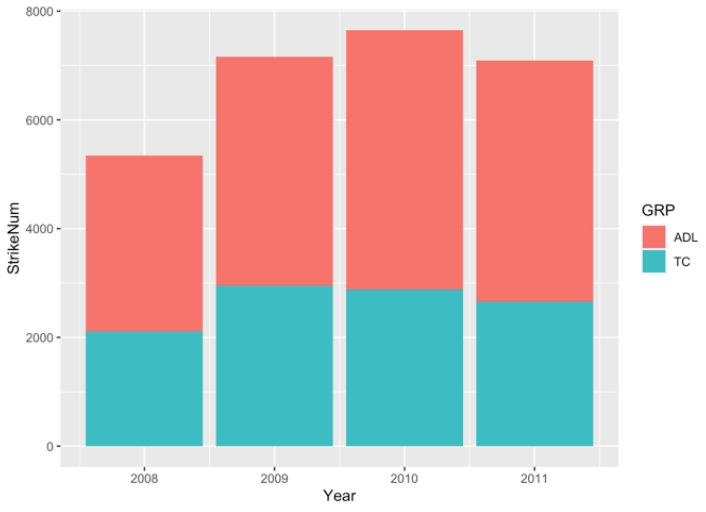
TASK 7

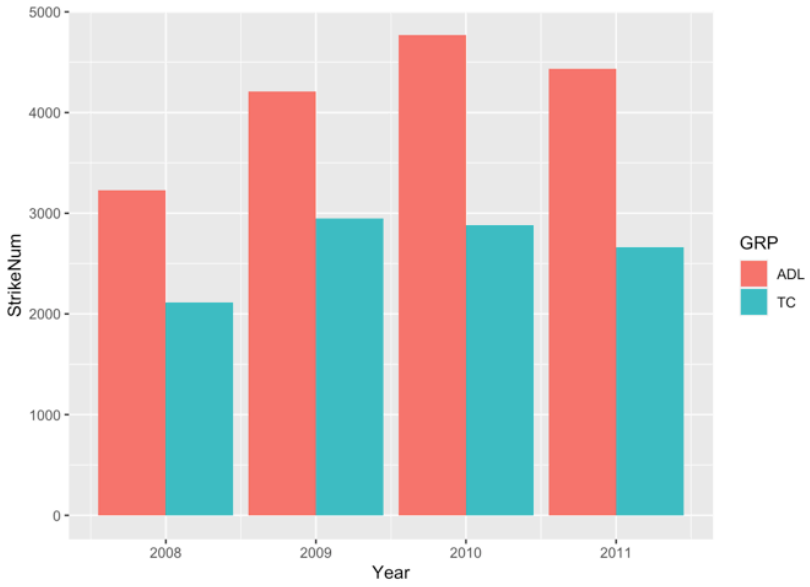
BIRDSTRIKES FROM 2008 – 2011 VISUALIZATION

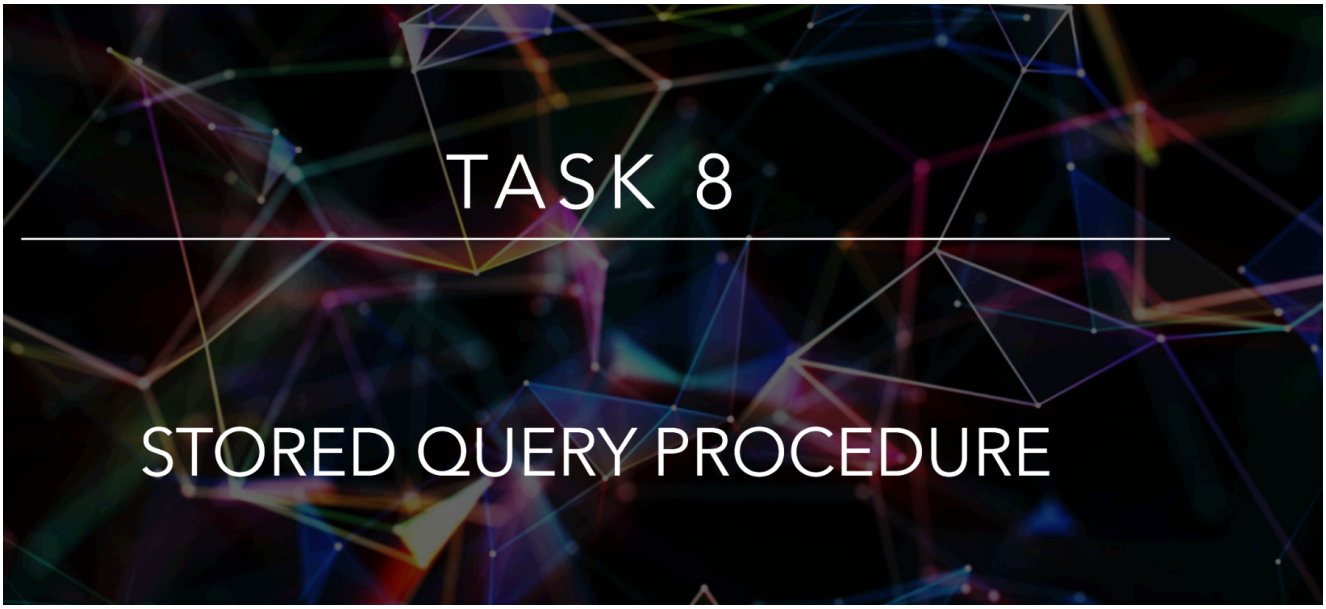
The graphs below visualizes the number of bird strikes incidents per year from 2008 to 2011 during take-off/climbing and during descent/ approach/landing. I categorized the flight phases into two groups ascending and descending.

ADL = Phases: Approach, Descent, Landing
TC = Phases: Take-Off, Climb

There are two types of representations: stacked & side by side to show how these two phases compare in number of strikes from years 2008 - 2011.




5000
4000
3000
2000
1000
0
StrikeNum
2008
2009
2010
2011
Year
GRP
ADL
TC



In order to create a stored query delete procedure, I used MySQL workbench. See the diagram below. I then integrated the stored query into R Studio.

NOTE: I did run into an error only when knitting my RNotebook into an HTML format. It is commented out on my code. I included the call function which executes without error in my R Studio Notebook file.

```
select count(*) from BirdStrikeIncident;

CALL sp_DeleteIncident ('ALASKA AIRLINES');

SELECT
    COUNT(*)
FROM
    BirdStrikeIncident
WHERE
    airlineID = 4;

/*
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_DeleteIncident`(IN airlineNameIn varchar(50))
BEGIN
DECLARE airlineIDOut INT;

select airlineID INTO airlineIDOut from airline
where airlineName = airlineNameIn;

SELECT airlineIDOut;

delete from BirdStrikeIncident
where airlineID = airlineIDOut;

END
*/
```