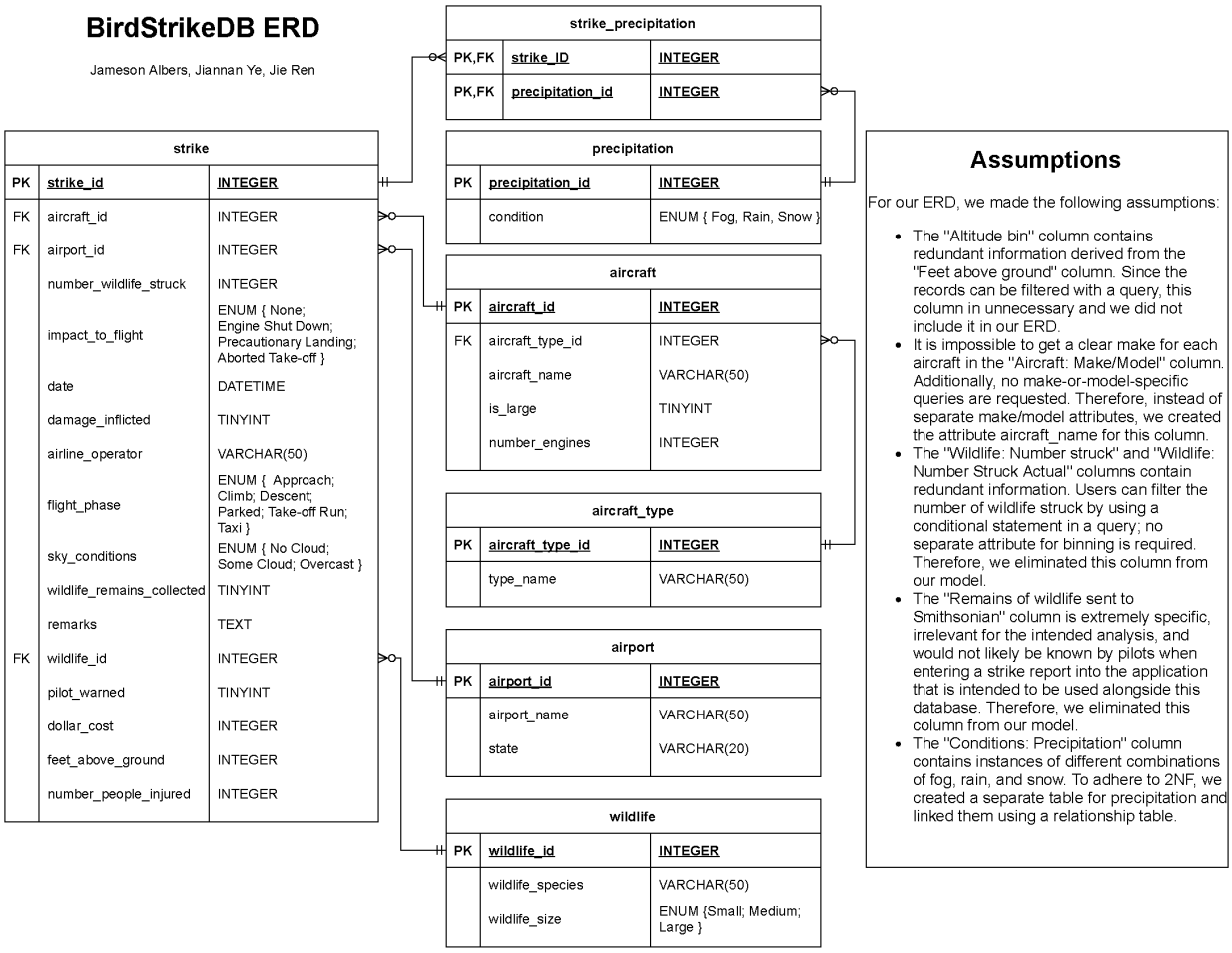
Practicum 1

Jie Ren Jiannan Ye Jameson Albers

Q1

See the Entity-Relationship diagram that was part of the submission. The image is also present below in the knitted PDF:



Q2

The first step is to connect to a local MySQL database. The database used by our group was named "bird_strike". You can use a different connection method to connect to any local or remote database, as long as the correct syntax is used.

```r
#install.packages('RMySQL') # Uncomment to install RMySQL package
library(RMySQL)
```

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

```r
# Connect to local MySQL database
db_user <- 'root'
db_password <- 'abcd1234' # Substitute your root-user password
db_name <- 'bird_strike' # Substitute your empty MySQL database name
db_host <- 'localhost'
db_port <- 3306

# Create new database connection
dbcon <- dbConnect(MySQL(), user = db_user, password = db_password,
                   dbname = db_name, host = db_host, port = db_port)
```

We now begin our data definition language. For the strike table, we created each attribute to not be null, as this ensures that any incomplete entries that would not provide useful information are excluded. Additionally, we used ENUM types instead of separate tables for the `impact_to_flight`, `flight_phase`, and `sky_conditions` attributes. This is because the distribution of each value was fairly linear, and greatly simplified the cleaning and insertion of our data. Additionally, since most of these values were not equivalent to NULL, the space savings from creating a separate table would be minimal, especially since ENUM types are resolved to an integer instead of a string, which saves a lot of space.

```sql
DROP TABLE IF EXISTS strike
```

```sql
CREATE TABLE strike (
  strike_id INTEGER PRIMARY KEY NOT NULL,
  aircraft_id INTEGER NOT NULL REFERENCES aircraft(aircraft_id),
  airport_id INTEGER NOT NULL REFERENCES airport(airport_id),
  wildlife_id INTEGER NOT NULL REFERENCES wildlife(wildlife_id),
  number_wildlife_struck INTEGER NOT NULL,
  impact_to_flight ENUM('None', 'Engine Shut Down', 'Aborted Take-off', 'Precautionary Landing', 'Other
  strike_date DATE,
  damage_inflicted INTEGER NOT NULL,
  airline_operator VARCHAR(50),
  flight_phase ENUM('Approach', 'Climb', 'Descent', 'Parked', 'Take-off run', 'Taxi', 'Landing Roll'),
  sky_conditions ENUM('No Cloud', 'Some Cloud', 'Overcast') NOT NULL,
  wildlife_remains_collected INTEGER NOT NULL,
  remarks TEXT NOT NULL,
  pilot_warned INTEGER,
  dollar_cost INTEGER NOT NULL,
  feet_above_ground INTEGER,
  number_people_injured INTEGER NOT NULL
)
```

We now begin the DDL for the aircraft table:

```sql
DROP TABLE IF EXISTS aircraft
```

```sql
CREATE TABLE aircraft (
  aircraft_id INTEGER NOT NULL PRIMARY KEY,
  aircraft_type_id INTEGER NOT NULL REFERENCES aircraft_type(aircraft_type_id),
  aircraft_name VARCHAR(50) NOT NULL,
  is_large INTEGER,
  number_engines INTEGER
)
```

Now, the aircraft_type table:

```
DROP TABLE IF EXISTS aircraft_type
```

```
CREATE TABLE aircraft_type (
  aircraft_type_id INTEGER NOT NULL PRIMARY KEY,
  type_name VARCHAR(50)
)
```

Next, the airport table:

```
DROP TABLE IF EXISTS airport
```

```
CREATE TABLE airport (
  airport_id INTEGER NOT NULL PRIMARY KEY,
  airport_name VARCHAR(50),
  state VARCHAR(20)
)
```

The wildlife table:

```
DROP TABLE IF EXISTS wildlife
```

```
CREATE TABLE wildlife (
  wildlife_id INTEGER NOT NULL PRIMARY KEY,
  wildlife_species VARCHAR(50) NOT NULL,
  wildlife_size ENUM('Small', 'Medium', 'Large')
)
```

The precipitation table:

```
DROP TABLE IF EXISTS precipitation
```

```
CREATE TABLE precipitation (
  precipitation_id INTEGER NOT NULL PRIMARY KEY,
  condition_name ENUM('Fog', 'Rain', 'Snow', 'None')
)
```

The strike_precipitation relationship table:

```
DROP TABLE IF EXISTS strike_precipitation
```

```
CREATE TABLE strike_precipitation (
  strike_id INTEGER NOT NULL REFERENCES strike(strike_id),
  precipitation_id INTEGER NOT NULL REFERENCES precipitation(precipitation_id),
  PRIMARY KEY (strike_id, precipitation_id)
)
```

Now that our database schema is defined, we can import the data from a CSV file, clean it in R, and insert it into our database.

Q3

The first step is to read the raw data from the CSV file into an R data frame:

```
fn = "BirdStrikesData.csv" # Path to CSV file
data <- read.csv(file = fn,
                 header = T,
                 stringsAsFactors = F)
#head(data) # Uncomment to check data frame
```

Next, we remove any records without flight or aircraft information, and transform the attribute values to fit the data types of the corresponding columns of our database schema:

```
# Omit records without flight or aircraft information.
valid_data <- subset(data, Aircraft..Type != "" & Aircraft..Make.Model != "" & Aircraft..Number.of.engi

# Modify data format
valid_data[valid_data == "Yes"] <- 1
valid_data[valid_data == TRUE] <- 1
valid_data[valid_data == "Y"] <- 1
valid_data[valid_data == "Caused damage"] <- 1
valid_data[valid_data == "No"] <- 0
valid_data[valid_data == FALSE] <- 0
valid_data[valid_data == "N"] <- 0
valid_data[valid_data == "No damage"] <- 0
valid_data$FlightDate = as.Date(valid_data$FlightDate, "%m/%d/%Y")

#head(valid_data) # Uncomment to check data frame
```

Next, we retrieve the unique precipitation values:

```
# Retrieve precipitation results
precipitations <- valid_data[,c('Conditions..Precipitation'), drop = F]
precipitations <- precipitations[!duplicated(precipitations), ]
print(precipitations)
```

```
## [1] "None"            "Snow"            "Fog"             "Rain"
## [5] "Fog, Rain"       "Rain, Snow"      "Fog, Rain, Snow" "Fog, Snow"
```

Since we have 3 possible values for precipitation, we will create a record for each of these unique individual values:

```
/* Insert precipitations */
INSERT INTO precipitation VALUES
(1, 'Snow'),
(2, 'Fog'),
(3, 'Rain');
```

Query the precipitation table to confirm our insertions were successful:

```
SELECT * FROM precipitation;
```

Table 1: 3 records

precipitation_id	condition_name
1	Snow
2	Fog
3	Rain

Next, we create a data frame with the unique aircraft types:

```
# Retrieve aircraft types
aircraft_type <- valid_data[,c('Aircraft..Type'), drop = F]
aircraft_type <- aircraft_type[!duplicated(aircraft_type), ]
print(aircraft_type)
```

```
## [1] "Airplane"
```

There is only one, so we can easily insert it:

```
/* Insert precipitations */
INSERT INTO aircraft_type VALUES (1, 'Airplane');
```

Confirm our insertion was successful:

```
SELECT * FROM aircraft_type;
```

Table 2: 1 records

aircraft_type_id	type_name
1	Airplane

Now, we create a data frame for the aircraft, and insert the values into our database:

```
# Insert aircraft
aircrafts <- valid_data[,c('Aircraft..Make.Model', 'Is.Aircraft.Large.', 'Aircraft..Number.of.engines.')
aircrafts <- cbind(1, aircrafts[, 1:ncol(aircrafts)])
aircrafts <- aircrafts[!duplicated(aircrafts), ]
colnames(aircrafts) <- c("aircraft_type_id","aircraft_name", "is_large", "number_engines")
aircrafts <- cbind(1, aircrafts[, 1:ncol(aircrafts)])
n.aircrafts <- nrow(aircrafts)
aircrafts[,1] <- seq(1, n.aircrafts)
colnames(aircrafts)[1] <- "aircraft_id"
dbSendQuery(dbcon, "SET GLOBAL local_infile = true;")
```

```
## <MySQLResult:0,0,18>
```

```
dbWriteTable(dbcon, value = aircrafts, name = "aircraft", append = T, row.names = F)
```

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

Confirm successful insertion:

```
SELECT * FROM aircraft;
```

Table 3: Displaying records 1 - 10

aircraft_id	aircraft_type_id	aircraft_name	is_large	number_engines
1	1	B-737-400	1	2
2	1	MD-80	0	2
3	1	C-500	0	2
4	1	CL-RJ100/200	0	2
5	1	A-300	0	2
6	1	LEARJET-25	0	2
7	1	A-320	0	2
8	1	DC-9-30	0	2
9	1	A-330	0	2
10	1	FOKKER F100	0	2

Now, we insert the airports:

```
# Insert airports
airports <- valid_data[,c('Airport..Name','Origin.State')]
airports <- airports[!duplicated(airports), ]
```

```
airports <- cbind(1, airports[, 1:ncol(airports)])
n.airports <- nrow(airports)
airports[,1] <- seq(1, n.airports)
colnames(airports) <- c("airport_id", "airport_name","state")
dbWriteTable(dbcon, value = airports, name = "airport", append = T, row.names = F)
```

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

Confirm successful insertion:

```
SELECT * FROM airport;
```

Table 4: Displaying records 1 - 10

airport_id	airport_name	state
1	LAGUARDIA NY	New York
2	DALLAS/FORT WORTH INTL ARPT	Texas
3	LAKEFRONT AIRPORT	Louisiana
4	SEATTLE-TACOMA INTL	Washington
5	NORFOLK INTL	Virginia
6	GUAYAQUIL/S BOLIVAR	N/A
7	NEW CASTLE COUNTY	Delaware
8	WASHINGTON DULLES INTL ARPT	DC
9	ATLANTA INTL	Georgia
10	ORLANDO SANFORD INTL AIRPORT	Florida

Next, insert the wildlife:

```
# Insert wildlife
wildlife <- valid_data[,c('Wildlife..Species', 'Wildlife..Size')]
wildlife <- wildlife[!duplicated(wildlife), ]
wildlife <- cbind(1, wildlife[, 1:ncol(wildlife)])
n.wildlife <- nrow(wildlife)
wildlife[,1] <- seq(1, n.wildlife)
colnames(wildlife) <- c("wildlife_id", "wildlife_species","wildlife_size")
dbWriteTable(dbcon, value = wildlife, name = "wildlife", append = T, row.names = F)
```

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

Confirm insertion:

```
SELECT * FROM wildlife;
```

Table 5: Displaying records 1 - 10

wildlife_id	wildlife_species	wildlife_size
1	Unknown bird - medium	Medium
2	Rock pigeon	Small
3	European starling	Small
4	Unknown bird - small	Small
5	Canada goose	Large
6	Snow goose	Large
7	Black-headed munia	Small
8	Ring-billed gull	Medium
9	Sandhill crane	Large

wildlife_id	wildlife_species	wildlife_size
10	Western meadowlark	Small

Now, we insert the strike information:

```
# Insert strikes
strikes <- valid_data[,c('ï..Record.ID', 'Wildlife..Number.Struck.Actual', 'Effect..Impact.to.flight',
colnames(strikes) <- c("strike_id", "number_wildlife_struck", "impact_to_flight","strike_date", "damage_

strikes <- cbind(strikes[1],1,strikes[,2:ncol(strikes)])
strikes <- cbind(strikes[1],1,strikes[,2:ncol(strikes)])
strikes <- cbind(strikes[1],1,strikes[,2:ncol(strikes)])
colnames(strikes)[2] <- "aircraft_id"
colnames(strikes)[3] <- "airport_id"
colnames(strikes)[4] <- "wildlife_id"

n.strikes <- nrow(strikes)
# process each row (strike) one by one
for (r in 1:n.strikes) {
  # find the aircraft, airport and wildlife PK for that strike
  a <- aircrafts$aircraft_id[which(aircrafts$aircraft_name == valid_data$Aircraft..Make.Model[r] & airc

  b <- airports$airport_id[which(airports$airport_name == valid_data$Airport..Name[r] & airports$state =

  c <- wildlife$wildlife_id[which(wildlife$wildlife_species == valid_data$Wildlife..Species[r] & wildli
Wildlife..Size[r])]

  strikes$aircraft_id[r] <- a
  strikes$airport_id[r] <- b
  strikes$wildlife_id[r] <- c
}

dbWriteTable(dbcon, value = strikes, name = "strike", append = T, row.names = F)
```

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

Confirm insertion:

```
SELECT strike_id,strike_date,airline_operator,impact_to_flight,flight_phase FROM strike;
```

Table 6: Displaying records 1 - 10

strike_id	strike_date	airline_operator	impact_to_flight	flight_phase
200011	2000-04-06	ABX AIR	None	Landing Roll
200012	2000-04-10	AIR MIDWEST	None	Landing Roll
200022	2000-02-28	AMERICAN EAGLE AIRLINES	None	Approach
200023	2000-08-25	US AIRWAYS*	None	Descent
200028	2000-03-22	UNITED AIRLINES	None	Approach
200029	2000-05-14	BUSINESS EXPRESS	None	Approach
200031	2000-06-12	GREAT LAKES AIRLINES	None	Approach
200036	2000-09-29	BUSINESS	None	Climb
200037	2001-02-01	UNITED AIRLINES	None	Take-off run
200042	2000-08-31	FEDEX EXPRESS	None	Approach

Next, we insert the rows for the strike_precipitation relationship table:

```
# Insert snow precipitations
snow <- dplyr::filter(valid_data, grepl('Snow', Conditions..Precipitation))
snow <- snow[,c('ï..Record.ID'), drop = F]
snow <- cbind(snow[1], 1)
colnames(snow) <- c("strike_id","precipitation_id")
dbWriteTable(dbcon, value = snow, name = "strike_precipitation", append = T, row.names = F)
```

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

```
# Insert fog precipitations
fog <- dplyr::filter(valid_data, grepl('Fog', Conditions..Precipitation))
fog <- fog[,c('ï..Record.ID'), drop = F]
fog <- cbind(fog[1], 2)
colnames(fog) <- c("strike_id","precipitation_id")
dbWriteTable(dbcon, value = fog, name = "strike_precipitation", append = T, row.names = F)
```

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

```
# Insert rain precipitations
rain <- dplyr::filter(valid_data, grepl('Rain', Conditions..Precipitation))
rain <- rain[,c('ï..Record.ID'), drop = F]
rain <- cbind(rain[1], 3)
colnames(rain) <- c("strike_id","precipitation_id")
dbWriteTable(dbcon, value = rain, name = "strike_precipitation", append = T, row.names = F)
```

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

Confirm insertion:

```
SELECT * FROM strike_precipitation;
```

Table 7: Displaying records 1 - 10

strike_id	precipitation_id
200011	3
200012	3
200097	1
200120	3
200194	2
200200	2
200218	3
200219	1
200259	3
200267	1

Q4

The first query will return the number of bird strikes during the climb or take-off phases by airline:

```
SELECT airline_operator, Count(*) AS num_strikes
FROM strike
WHERE flight_phase = 'Climb' OR 'Take-off Run'
GROUP BY airline_operator
```

Table 8: Displaying records 1 - 10

airline_operator	num_strikes
BUSINESS	672
UNITED AIRLINES	62
NORTHWEST AIRLINES	35
AMERICAN AIRLINES	414
MESA AIRLINES	35
AMERICA WEST AIRLINES	28
AIR TRANSPORT INTL	3
PRIVATELY OWNED	105
US AIRWAYS*	179
ATLANTIC SOUTHEAST	56

Q5

The second query will return the airports with the greatest number of bird strikes:

```
SELECT airport_name, Count(*) AS num_strikes
FROM strike s JOIN Airport a ON (s.airport_id = a.airport_id)
GROUP BY airport_name
ORDER BY num_strikes DESC
```

Table 9: Displaying records 1 - 10

airport_name	num_strikes
DALLAS/FORT WORTH INTL ARPT	802
SACRAMENTO INTL	676
SALT LAKE CITY INTL	479
DENVER INTL AIRPORT	476
KANSAS CITY INTL	452
PHILADELPHIA INTL	442
ORLANDO INTL	408
BALTIMORE WASH INTL	401
LOUISVILLE INTL ARPT	394
JOHN F KENNEDY INTL	389

Q6

The third query will return the number of strikes by year:

```
SELECT YEAR(strike_date) AS Year, Count(strike_date) AS Strikes
FROM strike
GROUP BY YEAR(strike_date)
```

Table 10: Displaying records 1 - 10

Year	Strikes
2000	1349
2001	1195
2007	2288

Year	Strikes
2006	2148
2009	3206
2008	2242
2004	1690
2002	1642
2003	1566
2005	1851

Q7

We will now create a column chart that compares the number of bird strikes during take-off or climb and compare it to the number of strikes during approach, descent, and landing from 2008-2011.

The first step is to return the climb-phase strikes and return a data frame containing the information we want:

```
SELECT YEAR(strike_date) AS Year, COUNT(*) AS Num, 'Climbing' AS Phase
FROM strike
WHERE YEAR(strike_date) >= 2008 AND YEAR(strike_date) <=2011 AND (flight_phase = 'Take-off run' OR flig
GROUP BY YEAR(strike_date)
```

Next, we do the same for the descent phase:

```
SELECT YEAR(strike_date) AS Year, COUNT(*) AS Num, 'Descending' AS Phase
FROM strike
WHERE YEAR(strike_date) >= 2008 AND YEAR(strike_date) <=2011 AND (flight_phase = 'Approach' OR flight_p
GROUP BY YEAR(strike_date)
```

Now, we combine the results of the two queries and create our chart:

```
#install.packages("ggplot2","dplyr") # Uncomment to install packages
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

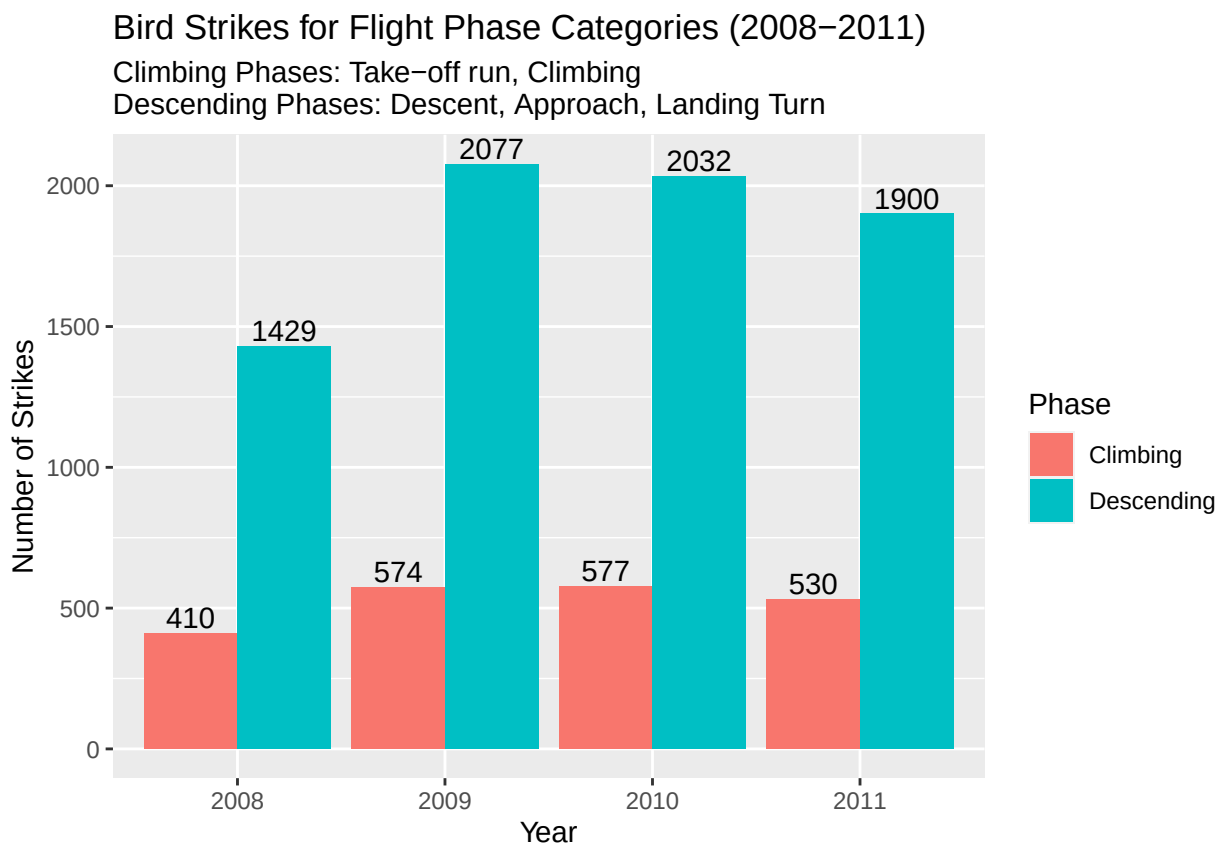
```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# Concatenate climb/descent data into the same data frame
concatenated_data = rbind(raw_data_climbing_q7,raw_data_descending_q7) %>%
  mutate(Year=as.character(Year)) # Character type makes plot look better
#head(concatenated_data) # Uncomment to check data frame

# Create column chart object
chart <- ggplot(concatenated_data, aes(x=Year, y=Num, fill=Phase)) +
                geom_bar(stat='identity',position=position_dodge()) +
                geom_text(aes(label=Num), position=position_dodge(width=0.9), vjust=-0.25) +
                labs(title='Bird Strikes for Flight Phase Categories (2008-2011)',
```

```
                    subtitle='Climbing Phases: Take-off run, Climbing\nDescending Phases: Descent, Appr
                    x='Year', y='Number of Strikes')

chart
```



Q8

Our stored procedure will take the name of an airline as an input, then delete the rows in the strike and strike_precipitation tables corresponding to that airline. It is important do delete from both tables to assure data integrity.

```
DROP PROCEDURE IF EXISTS delete_airline
```

We can delete the appropriate rows from both tables by using a left join:

```
CREATE PROCEDURE delete_airline (IN airline VARCHAR(50))
BEGIN
  DELETE strike, strike_precipitation
  FROM strike
    LEFT JOIN strike_precipitation
    ON strike.strike_id = strike_precipitation.strike_id
  WHERE strike.airline_operator=airline;
END
```

We query the number of strikes by airline:

```
SELECT airline_operator, Count(*) AS num_strikes
FROM strike
GROUP BY airline_operator
ORDER BY num_strikes DESC
```

Table 11: Displaying records 1 - 10

airline_operator	num_strikes
SOUTHWEST AIRLINES	4628
BUSINESS	3046
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

The 'BUSINESS' airline has a lot of records, so we will select it to be our test case. Let's query the strike_precipitation table to see if it has any records for precipitation:

```
SELECT s.airline_operator, p.precipitation_id, COUNT(*) AS Total
FROM strike AS s
  JOIN strike_precipitation AS p
  ON s.strike_id=p.strike_id
WHERE s.airline_operator='BUSINESS'
GROUP BY precipitation_id
```

Table 12: 3 records

airline_operator	precipitation_id	Total
BUSINESS	2	99
BUSINESS	3	219
BUSINESS	1	10

It has records for each precipitation type, and will be a good test case. We call our stored procedure with the 'BUSINESS' airline:

```
CALL delete_airline('BUSINESS')
```

Now, we repeat our queries to confirm that all records from this airline were deleted:

```
SELECT airline_operator, Count(*) AS num_strikes
FROM strike
GROUP BY airline_operator
ORDER BY num_strikes DESC
```

Table 13: Displaying records 1 - 10

airline_operator	num_strikes
SOUTHWEST AIRLINES	4628
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
UNITED AIRLINES	506

All the 'BUSINESS' strikes are gone! Next, we need to check the strike_precipitation table

```
SELECT s.airline_operator, p.precipitation_id, COUNT(*) AS Total
FROM strike AS s
  JOIN strike_precipitation AS p
  ON s.strike_id=p.strike_id
WHERE s.airline_operator='BUSINESS'
GROUP BY precipitation_id
```

Table 14: 0 records

airline_operator	precipitation_id	Total

Finally, we disconnect from the database.

```
dbDisconnect(dbcon)
```

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