Practicum Notebook

Calvin Crosby, Kristine Umeh

Required packages to be installed:

Connecting to the database

```
# 1. Library
library(RMySQL)
# 2. Settings
db_user <- 'admin'
db_password <- [redacted]
db_name <- [redacted]
db_host <- [redacted]
db_port <- 3306

# 3. Read data from db
mydb <-  dbConnect(MySQL(), user = db_user, password = db_password,
                 dbname = db_name, host = db_host, port = db_port)
```

Drop statements to empty database

```
DROP TABLE IF EXISTS incidents
```

```
DROP TABLE IF EXISTS airlines
```

```
DROP TABLE IF EXISTS airports
```

Schema defined for airlines table

```
create table airlines(
  aid INT NOT NULL,
  code VARCHAR(200)NOT NULL DEFAULT 'unknown',
  airline VARCHAR(200) UNIQUE NOT NULL DEFAULT 'unknown',
  PRIMARY KEY (aid)
);
```

Schema defined for airports table

```
create table airports(
pid INT NOT NULL,
code VARCHAR(200)NOT NULL DEFAULT 'unknown',
name VARCHAR(200)UNIQUE NOT NULL DEFAULT 'unknown',
city VARCHAR(200) NOT NULL DEFAULT 'unknown',
state VARCHAR(200)NOT NULL DEFAULT 'unknown',
country VARCHAR(200)NOT NULL DEFAULT 'unknown',
PRIMARY KEY (pid)
);
```

Schema defined for incidents table

```
create table incidents (
  iid INT NOT NULL,
  dateOnly DATE DEFAULT NULL,
  depPort INT,
  arrPort INT,
  airline INT,
  aircraft VARCHAR(200)NOT NULL DEFAULT 'unknown',
  flightPhase VARCHAR(12)NOT NULL DEFAULT 'unknown',
  impact VARCHAR(200) NOT NULL DEFAULT 'unknown',
  PRIMARY KEY (iid),
  FOREIGN KEY (airline) REFERENCES airlines(aid),
  FOREIGN KEY (depPort) REFERENCES airports(pid),
  FOREIGN KEY (arrPort) References airports(pid)
)
```

```
install.packages("hash")
```

```
Error in install.packages : Updating loaded packages
```

A hash map is made to map possible encountered values to harmonized values. It is used by my harmonizer function

```
library(hash)

flightMap <- hash()
flightMap[["Take-off run"]] <- "takeoff"
flightMap[["Landing Roll"]] <- "landing"
flightMap[["Climb"]] <- "inflight"
flightMap[["Approach"]] <- "inflight"
flightMap[["Descent"]] <- "inflight"
flightMap[["Taxi"]] <- "takeoff"
flightMap[["Parked"]]<- "unknown"
flightMap[[" "]]<- "unknown"
```

Defining my phase harmonizer function. Tests are run at the end to show functionality

```
# This function is used to harmonize the flight phase data from the csv.
# It references the flightMap hashmap and returns the appropriate string corresponding
# to a known phase or it returns "unknown" if an unrecognized or NULL value is
# encountered.
phaseHarmonizer <- function(flightPhase){
  if (is.null(flightPhase)){
    return("unknown")
  }
  if (has.key(flightPhase,flightMap)){
    return(flightMap[[flightPhase]])
  }
  return("unknown")
}
phaseHarmonizer("Take-off run")
```

```
[1] "takeoff"
```

```
phaseHarmonizer('Take-off run')
```

```
[1] "takeoff"
```

```
phaseHarmonizer("Climbing")
```

```
[1] "unknown"
```

```
phaseHarmonizer("In the ocean lol")
```

```
[1] "unknown"
```

```
phaseHarmonizer("Landing Roll")
```

```
[1] "landing"
```

```
phaseHarmonizer(NULL)
```

```
[1] "unknown"
```

```
phaseHarmonizer("Taxi")
```

```
[1] "takeoff"
```

```
phaseHarmonizer("Parked")
```

```
[1] "unknown"
```

This reads in the csv file and stores it in a data frame. Duplicates are removed

```
require(readr)
library(tidyverse)
birds <- read_csv('BirdStrikesData.csv')
```

```
Rows: 25558 Columns: 26
-- Column specification --------------------------------------------------------------------------
Delimiter: ","
chr (18): Aircraft: Type, Airport: Name, Altitude bin, Aircraft: Make/Model, Wildlife: Number ...
dbl  (4): Record ID, Wildlife: Number Struck Actual, Aircraft: Number of engines?, Number of p...
lgl  (2): Remains of wildlife collected?, Remains of wildlife sent to Smithsonian

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
birds[!duplicated(birds$`Record ID`), ] #remove duplicates
```

Record ID <dbl>	Aircraft: Type <chr>	Airport: Name <chr>	Altitude bin <chr>	Aircraft: Make/Model <chr>
202152	Airplane	LAGUARDIA NY	> 1000 ft	B-737-400
208159	Airplane	DALLAS/FORT WORTH INTL ARPT	< 1000 ft	MD-80
207601	Airplane	LAKEFRONT AIRPORT	< 1000 ft	C-500
215953	Airplane	SEATTLE-TACOMA INTL	< 1000 ft	B-737-400
219878	Airplane	NORFOLK INTL	< 1000 ft	CL-RJ100/200
218432	Airplane	GUAYAQUIL/S BOLIVAR	< 1000 ft	A-300
221697	Airplane	NEW CASTLE COUNTY	< 1000 ft	LEARJET-25
236635	Airplane	WASHINGTON DULLES INTL ARPT	< 1000 ft	A-320
207369	Airplane	ATLANTA INTL	< 1000 ft	DC-9-30
204371	Airplane	ORLANDO SANFORD INTL AIRPORT	< 1000 ft	A-330

1-10 of 25,558 rows | 1-5 of 26 columns   Previous 1 2 3 4 5 6 ... 100 Next

This uses my custom phaseHarmonizer function to update the birds data frame flight Phase column. It checks for additional cases like empty strings and null values

```
nrows <- nrow(birds)

for (i in 1:nrows){
  if (is.na(birds[[i,14]]) | birds[[i,14]] == '')
    birds[[i,14]] == "unknown"
  else if (is.null(birds[[i,14]]))
    birds[[i,14]] == "unknown"
  else if (birds[[i,14]]== ""){
    birds[[i,14]] <- "unknown"
  }
  else{
    temp <- birds[[i,14]]
    temp <- phaseHarmonizer(temp)
    birds[[i,14]] <- temp
    }
  }
```

```
birds
```

Record ID <dbl>	Aircraft: Type <chr>	Airport: Name <chr>	Altitude bin <chr>	Aircraft: Make/Model <chr>
202152	Airplane	LAGUARDIA NY	> 1000 ft	B-737-400
208159	Airplane	DALLAS/FORT WORTH INTL ARPT	< 1000 ft	MD-80
207601	Airplane	LAKEFRONT AIRPORT	< 1000 ft	C-500
215953	Airplane	SEATTLE-TACOMA INTL	< 1000 ft	B-737-400
219878	Airplane	NORFOLK INTL	< 1000 ft	CL-RJ100/200
218432	Airplane	GUAYAQUIL/S BOLIVAR	< 1000 ft	A-300
221697	Airplane	NEW CASTLE COUNTY	< 1000 ft	LEARJET-25
236635	Airplane	WASHINGTON DULLES INTL ARPT	< 1000 ft	A-320
207369	Airplane	ATLANTA INTL	< 1000 ft	DC-9-30
204371	Airplane	ORLANDO SANFORD INTL AIRPORT	< 1000 ft	A-330

1-10 of 25,558 rows | 1-5 of 26 columns   Previous 1 2 3 4 5 6 ... 100 Next

```
library(anytime)

print(typeof(birds))
```

```
[1] "list"
```

```
print(typeof(birds[,12]))
```

```
[1] "list"
```

```
#airlineData is used to populate airline table, aid is auto-incremented
airlineData <- data.frame(airline = distinct(birds[,12])) #look up table for airlines
airlineData <- transmute(airlineData, airline=`Aircraft..Airline.Operator`,aid=1:n())

#airportsData is used to populate the airports table, pid is auto-incremented
airportsData <- data.frame(name = birds[,3],state=birds[,13])
airportsData <- distinct(airportsData, `Airport..Name`, .keep_all = TRUE)
airportsData <- transmute(airportsData, name = `Airport..Name`, state=`Origin.State`,pid=1:n())

airports.name <- data.frame(name = birds$`Airport: Name`)
airports.state <- data.frame(state = birds$`Origin State`)

airPFull <-cbind(airports.name,airports.state)
incidentsFull <-cbind(incidents.iid,incidents.date)


incidentsFull <- transmute(birds, iid = `Record ID`, dateOnly= anydate(`FlightDate`),airline=birds$`Aircraft: Air
line/Operator`, aircraft=`Aircraft: Make/Model`, flightPhase=`When: Phase of flight`, name=birds$`Airport: Name`,
impact=`Effect: Impact to flight`)

incidentsFull <- left_join(incidentsFull, airlineData, by = "airline")
incidentsFull <- left_join(incidentsFull, airportsData, by = "name")

# IncidentsFull is the dataframe for the incidents table
tail(incidentsFull)
```

iid <dbl>	dateOnly <date>	airline <chr>	aircraft <chr>	flightPhase <chr>	name <chr>	impact <chr>
319672	2011-12-29	SOUTHWEST AIRLINES	B-737-700	inflight	SACRAMENTO INTL	None
321151	2011-12-30	SKYWEST AIRLINES	EMB-120	inflight	REDDING MUNICIPAL	None
319677	2011-12-30	US AIRWAYS	A-321	landing	ORLANDO INTL	None
319680	<NA>	NA	EC-135	NA	NA	NA
319679	2011-12-31	DELTA AIR LINES	B-757-200	landing	DETROIT METRO WAYNE COUNTY ARPT	None
319593	2011-12-31	XTRA AIRWAYS	B-737-400	takeoff	ABRAHAM LINCOLN CAPITAL ARPT	None

6 rows | 1-7 of 10 columns

```
NA
```

```
DROP TABLE IF EXISTS airlinesAUX
```

```
DROP TABLE IF EXISTS airportsAUX
```

```
DROP TABLE IF EXISTS incidentsAUX
```

Writing data into our auxillary tables

```
dbWriteTable(mydb,"incidentsAUX",incidentsFull,overwrite=F,append=T)
```

```
[1] TRUE
```

```
dbWriteTable(mydb,"airlinesAUX",airlineData,overwrite=F,append=T)
```

```
[1] TRUE
```

```
dbWriteTable(mydb,"airportsAUX",airportsData,overwrite=F,append=T)
```

```
[1] TRUE
```

Inserting data into the airlines auxillary table

```
INSERT INTO airlines(aid,airline) SELECT aid,airline FROM airlinesAUX;
```

Showing data in airlines table in the database

```
SELECT * FROM airlines
```

aid <int>	code <chr>	airline <chr>
1	unknown	US AIRWAYS*
2	unknown	AMERICAN AIRLINES
3	unknown	BUSINESS
4	unknown	ALASKA AIRLINES
5	unknown	COMAIR AIRLINES
6	unknown	UNITED AIRLINES
7	unknown	AIRTRAN AIRWAYS
8	unknown	AIRTOURS INTL
9	unknown	AMERICA WEST AIRLINES
10	unknown	EXECUTIVE JET AVIATION

1-10 of 293 rows   Previous 1 2 3 4 5 6 ... 30 Next

Inserting data into airports table from airport auxillary table

```
INSERT INTO airports(pid,name,state) SELECT pid,name,state FROM airportsAUX;
```

Showing the airports table in the database

```
SELECT * FROM airports
```

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

1-10 of 1,000 rows   Previous 1 2 3 4 5 6 ... 100 Next

Inserting data into incidents table from the incidents auxillary table

```
INSERT INTO incidents(iid,dateOnly,depPort,arrPort,airline,aircraft,flightPhase,impact)
SELECT iid,dateOnly,pid,pid,aid,aircraft,flightPhase,impact
FROM incidentsAUX;
```

Taking a look at my incidents table in the database

```
SELECT * FROM incidents
```

iid <int>	dateOnly <chr>	depPort <int>	arrPort <int>	airline <int>	aircraft <chr>	flightPhase <chr>	impact <chr>
1195	2002-11-13	37	37	21	B-52H	inflight	None
3019	2002-10-10	707	707	21	T-38A	inflight	Precautionary Landing
3500	2001-05-15	37	37	21	B-52H	inflight	Precautionary Landing
3504	2001-05-23	37	37	21	B-52H	inflight	Precautionary Landing
3597	2001-04-18	123	123	21	AT-38B	inflight	None
4064	2000-04-06	37	37	21	B-52H	inflight	None
4074	2002-07-15	180	180	21	F-16D	takeoff	None
4076	2002-07-15	37	37	21	B-52H	inflight	Precautionary Landing
4090	2001-07-02	114	114	21	C-17A	inflight	Precautionary Landing
4091	2001-07-07	114	114	21	C-21A	takeoff	Aborted Take-off

1-10 of 1,000 rows   Previous 1 2 3 4 5 6 ... 100 Next

Query 1

```
SELECT
airlines.airline AS 'ARRIVING AIR LINE',
COUNT(incidents.iid)AS 'NUM OF INCIDENTS'
FROM incidents
INNER JOIN airports ON airports.pid = incidents.arrPort
INNER JOIN airlines ON airlines.aid = incidents.airline
WHERE
(
(airports.name like '%LaGuardia%')
AND
(incidents.flightPhase = 'landing')
)
GROUP BY airlines.airline
```

ARRIVING AIR LINE <chr>	NUM OF INCIDENTS <dbl>
AIR CANADA	1
AMERICAN AIRLINES	2
AMERICAN EAGLE AIRLINES	4
CHAUTAUQUA AIRLINES	2
CONTINENTAL AIRLINES	1
DELTA AIR LINES	4
JETBLUE AIRWAYS	1
PIEDMONT AIRLINES	2
PSA AIRLINES	1
UNITED AIRLINES	1

1-10 of 11 rows   Previous 1 2 Next

Query 2

```
SELECT * FROM (
SELECT depPort,arrPort,airports.pid,airports.name,
COUNT(iid) AS 'COUNT_OF_INCIDENTS'
FROM incidents
INNER JOIN airlines ON incidents.airline = airlines.aid
INNER JOIN airports ON incidents.depPort = airports.pid
WHERE
(
  (
    (
    UPPER(airlines.airline) NOT LIKE '%BUSINESS%'
    )
    AND
    (UPPER(airlines.airline) NOT LIKE '%MILITARY%'
    )
    AND
    (UPPER(airlines.airline)NOT LIKE '%CORP%'
    )
    AND
    (UPPER(airlines.airline)NOT LIKE '%EXEC%'
    )
    AND
    (UPPER(airlines.airline)NOT LIKE '%GOVERNMENT%'
    )
    AND
    (UPPER(airlines.airline)NOT LIKE '%PRIVATE%'
    )
    AND
    (UPPER(airlines.airline)NOT LIKE '%UNKNOWN%'
    )
  )
)
GROUP BY depPort, arrPort,airports.pid,airports.name
ORDER BY COUNT_OF_INCIDENTS DESC
) AS COUNT_OF_COMMERCIAL_INC
LIMIT 1
```

depPort <int>	arrPort <int>	pid <int>	name <chr>	COUNT_OF_INCIDENTS <dbl>
2	2	2	DALLAS/FORT WORTH INTL ARPT	798

1 row

Query 3

```
SELECT
YEAR(dateOnly) AS 'YEAR',
COUNT(*) AS 'BIRD_STRIKES_PER_YEAR'
FROM
incidents
GROUP BY YEAR(dateOnly)
ORDER BY YEAR(dateOnly) ASC
```

YEAR <int>	BIRD_STRIKES_PER_YEAR <dbl>
NA	129
2000	1367
2001	1230
2002	1681
2003	1568
2004	1692
2005	1853
2006	2159
2007	2301
2008	2258

1-10 of 13 rows   Previous 1 2 Next

Plotting the Bird Incidents from 2005 to 2011

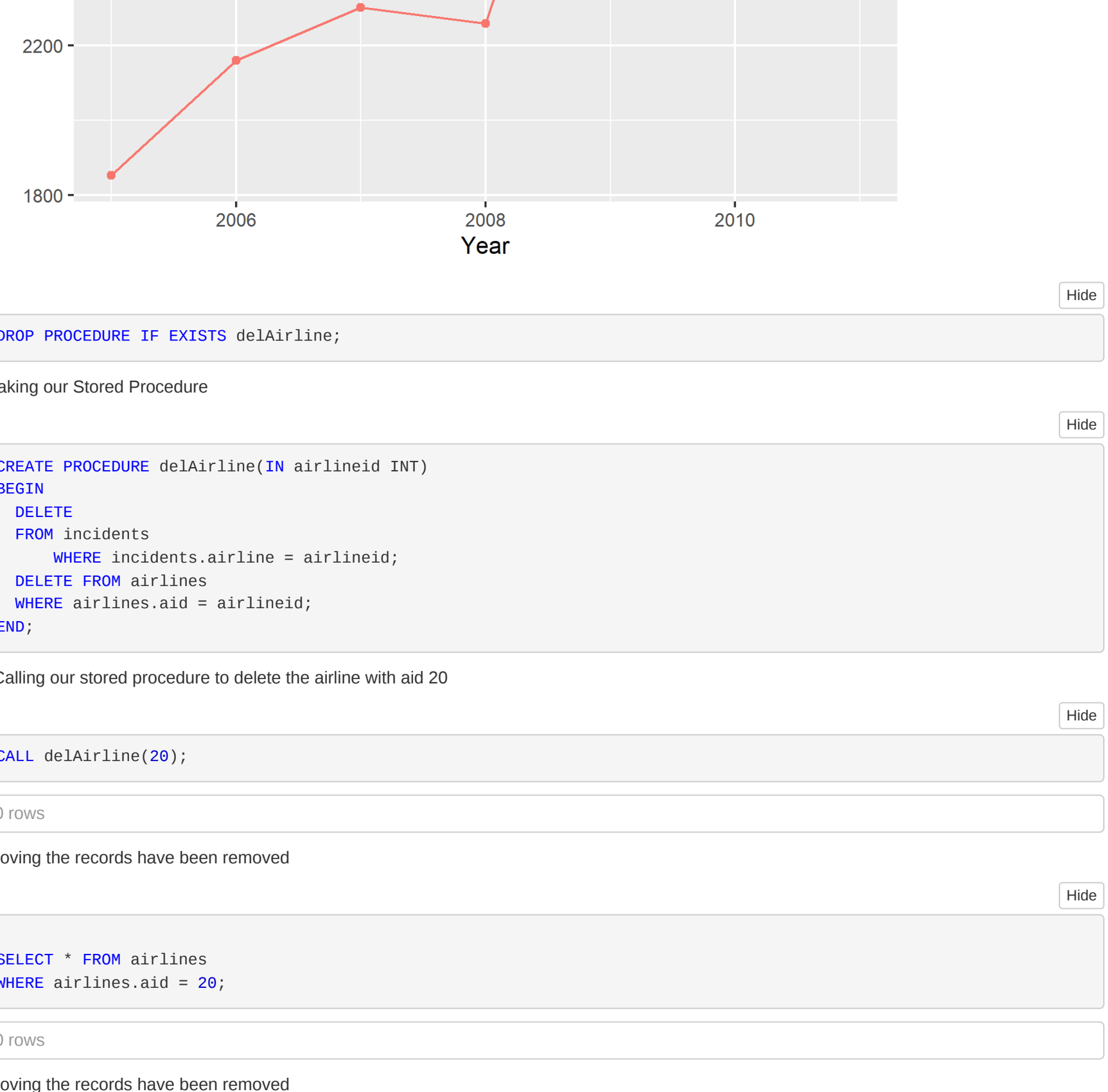
```
library(ggplot2)
library(sqldf)
frame <- dbGetQuery(mydb,"SELECT
YEAR(dateOnly) AS 'YEAR',
COUNT(*) AS 'BIRD_STRIKES_PER_YEAR'
FROM
incidents
GROUP BY YEAR(dateOnly)
ORDER BY YEAR(dateOnly) ASC")

filtered <- subset(frame, YEAR>=2005 & YEAR<=2011)
filteredmore <-subset(filtered,!is.null(YEAR))
print(filteredmore)
```

	YEAR <int>	BIRD_STRIKES_PER_YEAR <dbl>
7	2005	1853
8	2006	2159
9	2007	2301
10	2008	2258
11	2009	3247
12	2010	3121
13	2011	2952

7 rows

```
year <- filtered[,1]
bird_strikes <- filtered[,2]
#create data frame (frame) with above query using sqldf and pass into this:
ggplot(filtered, aes(x=year,y=bird_strikes,group=1,colour="Bird Strikes"))+geom_line()+ geom_point()+
  labs(title= "Bird Strikes Per Year vs. Year Recorded", x = "Year",y= "Bird Strikes")+scale_color_discrete(name
 = "Legend", labels = c("Bird Strikes "))+ theme(plot.title = element_text(hjust = 0.5))
```




```
DROP PROCEDURE IF EXISTS delAirline;
```

Making our Stored Procedure

```
CREATE PROCEDURE delAirline(IN airlineid INT)
BEGIN
  DELETE
  FROM incidents
      WHERE incidents.airline = airlineid;
  DELETE FROM airlines
  WHERE airlines.aid = airlineid;
END;
```

–Calling our stored procedure to delete the airline with aid 20

```
CALL delAirline(20);
```

0 rows

Proving the records have been removed

```
SELECT * FROM airlines
WHERE airlines.aid = 20;
```

0 rows

Proving the records have been removed

```
SELECT * FROM incidents
WHERE incidents.airline = 20;
```

0 rows

Disconnect from Database

```
dbDisconnect(mydb)
```