# Assignment 10Dply B: Comparison between dplyr and SQL SELECT

*Joshua Conte*

*July 16, 2017*

## Comparison between dplyr and SQL SELECT

For this assignment, I need to: extract aggregated statistics using dplyr and compare dplyr to SQL SELECT.

### Setting up R

First I configure R studio with the parameters below:

```r
# clears the console in RStudio
cat("\014")
```

```r
# clears environment
rm(list = ls())

# Set working directory
setwd("C:/R/DA5020/Week_10/")

# Load required packages
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
```

## Begin Assignment 10Dply B

### Assignment Tasks

#### Task 1

Revisit your Bird Strike database created for the last assignment. Connect to the database.

```r
# connect to the database file using src_sqlite
birdstrike<- src_sqlite("birdstrike.db", create = F)

# List the tables in the database
src_tbls(birdstrike)
```

```
## [1] "table1" "table2"
```

**Querying the database**
dplyr translates the R code we write to SQL code. Inorder to connect to the tables, I needed to use 'tbl', as shown below.

For table1:

```
# Load table1
table1 <- tbl(birdstrike,"table1")

# This shows the first 3 rows of table1
head(table1,3)
```

```
## Source:   query [?? x 6]
## Database: sqlite 3.19.3 [birdstrike.db]
##
## # A tibble: ?? x 6
##   RecordID AircraftType AircraftAirlineOperator              AirportName
##      <int>        <chr>                   <chr>                    <chr>
## 1   200508     Airplane    CONTINENTAL AIRLINES NEWARK LIBERTY INTL ARPT
## 2   206593     Airplane         UNITED AIRLINES                  UNKNOWN
## 3   206594     Airplane         UNITED AIRLINES      DENVER INTL AIRPORT
## # ... with 2 more variables: AircraftMakeModel <chr>, FlightDate <chr>
```

For table2:

```
# Load table2
table2 <- tbl(birdstrike,"table2")

# This shows the first 3 rows of table2
head(table2,3)
```

```
## Source:   query [?? x 2]
## Database: sqlite 3.19.3 [birdstrike.db]
##
## # A tibble: ?? x 2
##   RecordID ConditionsPrecipitation
##      <int>                   <chr>
## 1   200508                     Fog
## 2   206593                    <NA>
## 3   206594                    <NA>
```

**Task 2**

Write a dpyr statement that counts the number of incidents where the incident reported fog during the incident.

I can pipe dplyr operations together with %>%. The pipeline %>% takes the output from the left-hand side of the pipe as the first argument to the function on the right hand side.

```
# Grouped by ConditionsPrecipitation and found the count for each
# ConditionsPrecipitation type as BirdStrikes.  Then filtered for Fog.
table2%>%group_by(ConditionsPrecipitation)%>%
  summarize(BirdStrikes=n())%>%
  filter(ConditionsPrecipitation=='Fog')
```

```
## Source:   query [?? x 2]
## Database: sqlite 3.19.3 [birdstrike.db]
##
## # A tibble: ?? x 2
##   ConditionsPrecipitation BirdStrikes
##                     <chr>       <int>
## 1                     Fog        1061
```

### Task 3

Write a function **CountIncidents(aircraftType)** that accepts an aircraft type and returns the number of bird strike incidents for the aircraft type.

```r
CountIncidents <- function(x) {
  # This pipes dplyr operations together with %>%.
  # Grouped by AircraftType, then found the count for each AircraftType type as
  # BirdStrikes. Then filtered for 'x'.
  table1 %>% group_by(AircraftType) %>%
    summarize(BirdStrikes = n()) %>%
    filter(AircraftType == x)
}

# This calls the function with argument in quotes
CountIncidents("Helicopter")
```

```
## Source:   query [?? x 2]
## Database: sqlite 3.19.3 [birdstrike.db]
##
## # A tibble: ?? x 2
##   AircraftType BirdStrikes
##          <chr>       <int>
## 1   Helicopter        1010
```

### Task 4

Write a function **Incidents(Airline)** that accepts an airline and returns a dataframe that contains all incidents for that airline. Limit the columns to: **AirportName**, **AircraftModel**, and **FlightDate**.

```r
Incidents <- function(x) {
  # THis function accepts an airline and returns a dataframe that contains all
  # incidents for that airline. Limiting the columns to: AirportName, AircraftModel,
  # and FlightDate

  # This pipes dplyr operations together with %>%.
  # Select AircraftAirlineOperator, AirportName, AircraftMakeModel, FlightDate then
  # filtered for AircraftAirlineOperator 'x'. Ungrouped AircraftAirlineOperator,
  # then arrange by date oldest to newest
  table1 %>% select(AircraftAirlineOperator,
                    AirportName,
                    AircraftMakeModel,
                    FlightDate) %>%
    filter(AircraftAirlineOperator == x) %>%
    ungroup() %>%
    select(-AircraftAirlineOperator) %>%
    arrange(FlightDate)
}

# This calls the function with argument in quotes
Incidents.df <- Incidents("CONTINENTAL AIRLINES")

# Make it a data frame
Incidents.df <- data.frame(Incidents.df)

# Summarize the data frame
summary(Incidents.df)
```

```
##  AirportName        AircraftMakeModel   FlightDate
##  Length:833         Length:833         Length:833
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
```

```
# Show first three lines
head(Incidents.df, 3)
```

```
##                AirportName AircraftMakeModel FlightDate
## 1 NEWARK LIBERTY INTL ARPT         B-757-200 2000-01-01
## 2  LAFAYETTE REGIONAL (LA)            ATR-42 2000-01-04
## 3                  UNKNOWN         B-737-800 2000-01-27
```

**Task 5**

Write a function **CountIncidentsByAirline()** that creates a data frame where the first column is a name of an Aircraft and the second column is the total number of incidents the Airline had.

```
CountIncidentsByAirline <- function(x) {
  # THis function accepts an airline and returns a dataframe that contains all
  # incidents for that airline. Limiting the columns to: AirportName, AircraftModel,
  # and FlightDate

  # This pipes dplyr operations together with %>%.
  # Select AircraftAirlineOperator, AircraftMakeModel, then filtered for
  # AircraftAirlineOperator 'x', ungrouped AircraftAirlineOperator.
  # Then group by AircraftMakeModel and summarize by birdstikes and
  # order data by largest to smallest birdstrikes.
  table1 %>% select(AircraftMakeModel) %>%
  group_by(AircraftMakeModel) %>%
  summarize(BirdStrikes = n()) %>%
  arrange(desc(BirdStrikes))
}

# This calls the function with argument in quotes
IncidentsByAirline.df <- CountIncidentsByAirline()

# Make it a data frame
IncidentsByAirline.df <- data.frame(IncidentsByAirline.df)

# Summarize the data frame
summary(IncidentsByAirline.df)
```

```
##  AircraftMakeModel   BirdStrikes
##  Length:526         Min.   :    1.0
##  Class :character   1st Qu.:    2.0
##  Mode  :character   Median :   10.5
##                     Mean   :  189.0
##                     3rd Qu.:   63.0
##                     Max.   :24637.0
```

```
# Show first three lines
head(IncidentsByAirline.df,3)
```

```
##   AircraftMakeModel BirdStrikes
## 1           UNKNOWN       24637
## 2         B-737-300        5524
## 3             A-320        4654
```

**Task 6:**

Use **show_query()** to extract the SQL statement from one of your dplyr operations, then compare it with the corresponding SQL statement you wrote last week. State your preference for retrieving data from a large data set.
I selected the statement from task2, a statement that counts the number of incidents where the incident reported fog during the incident:

```
fog_dplyr<-table2%>%group_by(ConditionsPrecipitation)%>%
  summarize(BirdStrikes=n())%>%
  filter(ConditionsPrecipitation=='Fog')

show_query(fog_dplyr)
```

```
## <SQL>
## SELECT *
## FROM (SELECT `ConditionsPrecipitation`, COUNT() AS `BirdStrikes`
## FROM `table2`
## GROUP BY `ConditionsPrecipitation`)
## WHERE (`ConditionsPrecipitation` = 'Fog')
```

**SQL statement wrote last week:**
SELECT ConditionsPrecipitation, count(*) AS BirdStrikes
FROM table2
WHERE ConditionsPrecipitation
LIKE ‘fog’

The statements are similar, however, the one I used last week was a little smaller since I utilized the LIKE statement. Overall, I prefer to use SQL over dplyr because there is so much more support for SQL. It took me a while to find the basics of dplyr, but for SQL, it’s just a quick search on stackoverflow. Plus, SQL is used in a lot of different places, so it’s easier to port the code from one place to another.