

Retrieving data using the SQL SELECT statement

Suman Sahil, Steve Simon

Creation date: 2017-09-12

Working with the airlines dataset

- Database airlines_db

- Single table, airlines_table.
- Number of passengers bumped from their flights due to overbooking.
- For years 2016, 2017
- Also rate of bumping per ten thousand passengers.
 - More details at the [Data Description website](#).

The first data set is small. So small that you might wonder you would need a database for it.

The airlines_db database has a single table and that table only has five fields and twelve records. I wanted a database small enough that you could list the entire thing on a single slide.

You can check out the original source, though I did make a few minor modifications.

Select an entire table

– SQL code

```
select *  
from airlines_table
```

To select an entire table in SQL use the wild card symbol, asterisk. The asterisk is shorthand for “every field in the table.” If you are working directly with your database, you do not need any extra code, but in R and SAS there’s just a bit more to it.

For most of the lectures, I will not show the extra SAS and R code, but I wanted to illustrate it at least for the very first lecture.

Select an entire table in R (1/2)

— R code

```
library(sqldf)
db <- dbConnect(SQLite(),
  dbname="../data/airlines_db.sqlite")
airlines_data <- dbGetQuery(conn=db,
  "select *
    from airlines_table")
airlines_data
dbDisconnect(conn=db)
```

In R, you need a third party extension. There are several that will work. I've had pretty good luck with the sqldf library. There is a generic library dbi and an extension to Oracle called ROracle.

You also need to connect to the database before you can extract any information. use the dbConnect function for this. For a password protected database, you would need extra arguments for the user name and password.

Finally, enclose your SQL code in quotes and pass it to the dbGetQuery function. This function produces a data frame which I have stored in airlines_data.

Always remember to disconnect when you are done. This doesn't have to be right away, but you should disconnect somewhere rather than leaving the connection open when your program is done.

Select an entire table in R (2/2)

```
##      Airline b2017 b2016 r2017 r2016
## 1      DELTA    679    912  0.07  0.09
## 2      VIRGIN    165     77  0.27  0.13
## 3     JETBLUE   1475   2140  0.54  0.82
## 4      UNITED   2067   2874  0.30  0.45
## 5    HAWAIIAN    92     30  0.11  0.04
## 6 EXPRESSJET    785   2541  0.67  1.58
## 7     SKYWEST    917   2177  0.37  0.96
## 8    AMERICAN   4517   6598  0.46  0.66
## 9      ALASKA    658    734  0.35  0.41
## 10 SOUTHWEST   6678  11907  0.58  1.06
## 11 FRONTIER    540    688  0.45  0.63
## 12     SPIRIT   1502   1418  0.88  0.93
```

Here's what the data frame looks like.

Select entire table in SAS (1/2)

– SAS code

```
libname sql_lib odbc
  datasrc='sqlite3';
proc sql;
  create table full_table as
  select *
  from sql_lib.airlines_table;
quit;
proc print
  data=full_table;
run;
```

This program shows how to use the SELECT statement for SQL within a SAS program. This code shows all the steps that you need for a simple query that selects every record and all fields within a single table.

First you need to point to the database with a libname statement. Then you insert the code into proc sql.

By default, proc sql will just display the results of your query. To save a file for further work, use the create table as statement.

Notice that proc sql requires a quit statement rather than a run statement at the end.

Select entire table in SAS (2/2)

The SAS System

Obs	Airline	b2017	b2016	r2017	r2016
1	DELTA	679	912	0.07	0.09
2	VIRGIN	165	77	0.27	0.13
3	JETBLUE	1475	2140	0.54	0.82
4	UNITED	2067	2874	0.30	0.45
5	HAWAIIAN	92	30	0.11	0.04
6	EXPRESSJET	785	2541	0.67	1.58
7	SKYWEST	917	2177	0.37	0.96
8	AMERICAN	4517	6598	0.46	0.66
9	ALASKA	658	734	0.35	0.41
10	SOUTHWEST	6678	11907	0.58	1.06
11	FRONTIER	540	688	0.45	0.63
12	SPIRIT	1502	1418	0.88	0.93

SAS Output

Here's what the output looks like.

Selecting a single field

– SQL code

```
select airline  
from airlines_table
```

To select a single field, list that field's name after the select statement.

Selecting a single field in R (1/2)

— R code

```
db <- dbConnect(SQLite(),  
  dbname="../data/airlines_db.sqlite")  
airline_names <- dbGetQuery(conn=db,  
  "select Airline  
    from airlines_table")  
airline_names
```

In R, you need to connect again (skip this step if you didn't disconnect earlier). Then call the `dbGetQuery` function with the SQL code inserted. Keep the connection open for now to save time with the next couple of queries.

Selecting a single field in R (2/2)

```
##      Airline
## 1      DELTA
## 2     VIRGIN
## 3   JETBLUE
## 4     UNITED
## 5   HAWAIIAN
## 6 EXPRESSJET
## 7    SKYWEST
## 8   AMERICAN
## 9     ALASKA
## 10  SOUTHWEST
## 11  FRONTIER
## 12    SPIRIT
```

Here's what the output looks like.

Select a single field in SAS (1/2)

– SAS code

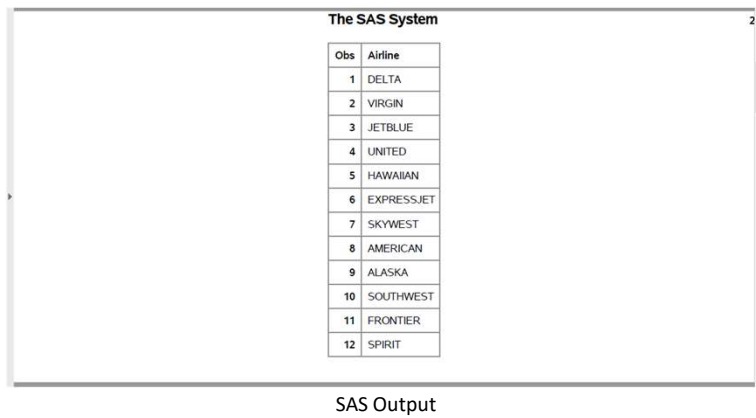
```
proc sql;  
  create table single_column as  
  select Airline  
  from sql_lib.airlines_table;  
quit;  
proc print  
  data=single_column;  
run;
```

This is how you select a single field in SAS.

As before, place the SQL query inside proc sql and use the create table as statement to store the results in a SAS data set.

Watch your semicolons carefully in SAS!

Select a single field in SAS (2/2)



The screenshot shows the SAS Output window titled "The SAS System". It displays a table with two columns: "Obs" (Observation) and "Airline". The table contains 12 rows of data, with observation numbers 1 through 12 and corresponding airline names. The window has a scroll bar on the right side.

Obs	Airline
1	DELTA
2	VIRGIN
3	JETBLUE
4	UNITED
5	HAWAIIAN
6	EXPRESSJET
7	SKYWEST
8	AMERICAN
9	ALASKA
10	SOUTHWEST
11	FRONTIER
12	SPIRIT

Here's what the output looks like.

Selecting multiple fields

– SQL code

```
select Airline, r2016, r2017  
from airlines_table
```

To select multiple fields, list them after the select statement separated by commas. Don't leave out the commas. I usually do, and it inevitably leads to a cryptic error message.

Selecting multiple fields in R (1/2)

```
airline_bump_rates <- dbGetQuery(conn=db,  
  "select  
    Airline, r2016, r2017  
  from airlines_table")  
airline_bump_rates
```

Here's the code in R.

Selecting multiple fields in R (2/2)

```
##      Airline r2016 r2017
## 1      DELTA  0.09  0.07
## 2      VIRGIN  0.13  0.27
## 3     JETBLUE  0.82  0.54
## 4      UNITED  0.45  0.30
## 5    HAWAIIAN  0.04  0.11
## 6 EXPRESSJET  1.58  0.67
## 7     SKYWEST  0.96  0.37
## 8    AMERICAN  0.66  0.46
## 9      ALASKA  0.41  0.35
## 10 SOUTHWEST  1.06  0.58
## 11  FRONTIER  0.63  0.45
## 12     SPIRIT  0.93  0.88
```

Here's what the output looks like.

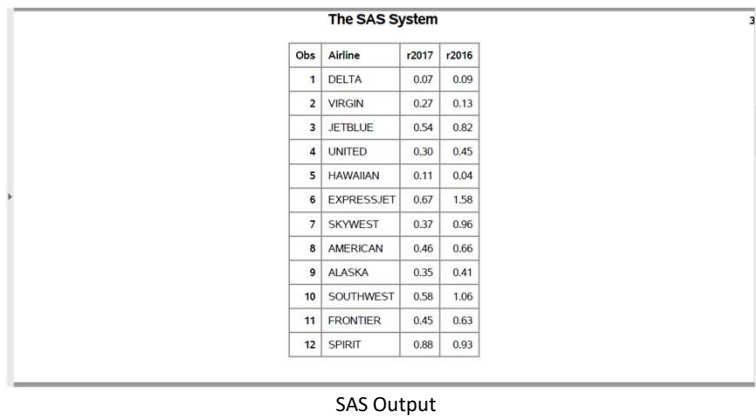
Select a multiple fields in SAS (1/2)

– SAS code

```
proc sql;  
  create table multiple_columns as  
  select Airline, r2017, r2016  
  from sql_lib.airlines_table;  
quit;  
proc print  
  data=multiple_columns;  
run;
```

This is how you select multiple fields in SAS.

Select a multiple fields in SAS (2/2)



The screenshot shows the SAS Output window titled "The SAS System". It displays a table with 4 columns: "Obs", "Airline", "r2017", and "r2016". The table contains 12 rows of data, numbered 1 through 12 in the "Obs" column. The "Airline" column lists various airlines, and the "r2017" and "r2016" columns contain numerical values. The table is presented in a scrollable window with a vertical scrollbar on the right side.

Obs	Airline	r2017	r2016
1	DELTA	0.07	0.09
2	VIRGIN	0.27	0.13
3	JETBLUE	0.54	0.82
4	UNITED	0.30	0.45
5	HAWAIIAN	0.11	0.04
6	EXPRESSJET	0.67	1.58
7	SKYWEST	0.37	0.96
8	AMERICAN	0.46	0.66
9	ALASKA	0.35	0.41
10	SOUTHWEST	0.58	1.06
11	FRONTIER	0.45	0.63
12	SPIRIT	0.88	0.93

SAS Output

Here's what the output looks like.

Changing field names

– SQL code

```
select
  Airline,
  r2017 as current_rate,
  r2016 as previous_rate
from airlines_table
```

Use the AS keyword to change the name of a field. Notice where the commas go. This code renames the fields in the output, but the names in the original database remain the same.

Changing field names in R (1/2)

```
changed_names <- dbGetQuery(conn=db,  
  "select  
    Airline,  
    r2017 as current_rate,  
    r2016 as previous_rate  
  from airlines_table")  
changed_names  
dbDisconnect(conn=db)
```

Here's the R code. Since this is the last query in R, you need to disconnect here.

Changing field names in R (2/2)

```
##      Airline current_rate previous_rate
## 1      DELTA          0.07           0.09
## 2      VIRGIN          0.27           0.13
## 3     JETBLUE          0.54           0.82
## 4      UNITED          0.30           0.45
## 5    HAWAIIAN          0.11           0.04
## 6 EXPRESSJET          0.67           1.58
## 7     SKYWEST          0.37           0.96
## 8    AMERICAN          0.46           0.66
## 9      ALASKA          0.35           0.41
## 10 SOUTHWEST          0.58           1.06
## 11 FRONTIER          0.45           0.63
## 12    SPIRIT          0.88           0.93
```

Here's what the output looks like.

Renaming fields (1/2)

– SAS code

```
proc sql;
  create table renamed_fields as
  select
    Airline,
    r2017 as current_rate,
    r2016 as previous_rate
  from sql_lib.airlines_table;
quit;
proc print
  data=renamed_fields;
run;
```

You can rename fields in proc sql, but be careful. Sometimes SAS retains the original name as the variable label. If you have trouble with renaming, you may want to do the renaming in SAS itself.

Renaming fields (2/2)

The SAS System

Obs	Airline	current_rate	previous_rate
1	DELTA	0.07	0.09
2	VIRGIN	0.27	0.13
3	JETBLUE	0.54	0.82
4	UNITED	0.30	0.45
5	HAWAIIAN	0.11	0.04
6	EXPRESSJET	0.67	1.58
7	SKYWEST	0.37	0.96
8	AMERICAN	0.46	0.66
9	ALASKA	0.35	0.41
10	SOUTHWEST	0.58	1.06
11	FRONTIER	0.45	0.63
12	SPIRIT	0.88	0.93

SAS Output

Here's what the output looks like.

Your homework

– Database crawling_db

- Single table, crawling_table.
 - Birth_month (self-explanatory)
 - Temperature (Average temperature (F) six months after birth)
 - avg_crawling_age (in weeks)
- More details at the [Data Description website](#).
- Read all three fields and all records
 - Change Temperature to Temperature_F
 - Put your code and the output in a single PDF file

The database for your first homework assignment is also very small. It has three fields and twelve records. This data came from a research publication in Infant Behavior and Development. The original data set included 414 infants, but the results were averaged by birth month. The hypothesis is that the colder temperature around the time when infants might start to crawl might adversely influence the age at which the infants started crawling because the bulky clothes needed in cold weather might restrict movement and interfere with their normal development.

You can check out the original source on the data description website.