STAT 209: SQL Part II

Code

Basic query components in SQL (cont'd)

Goal

Learn SQL equivalents of data-wrangling tools beyond the basic "five verbs"; particularly "joins", and get some practice combining the fundamental ingredients into productive queries – and getting used to some of the "quirks" of SQL's requirements about query structure – to access data from large databases stored on a remote server in a resource-efficient way.

Setting up the connection

We need to load packages and set up a connection to the server again. We'll again do some work with the `"airlines"` database.

Code:

Code

For convenience here is the list of basic verbs again:

`SELECT` allows you to list the columns, or functions operating on columns, that you want to retrieve. This is an analogous operation to the `select()` verb in `dplyr`, potentially combined with `mutate()`.

`FROM` specifies the table where the data are.

`JOIN` allows you to stitch together two or more tables using a key. This is analogous to the `join()` commands in `dplyr`.

`WHERE` allows you to filter the records according to some criteria. This is an analogous operation to the `filter()` verb in `dplyr`.

`GROUP BY` allows you to aggregate the records according to some shared value. This is an analogous operation to the `group_by()` verb in `dplyr`.

`HAVING` is like a `WHERE` clause that operates on the result set—not the records themselves. This is analogous to applying a second `filter()` command in `dplyr`, after the rows have already been aggregated.

`ORDER BY` is exactly what it sounds like—it specifies a condition for ordering the rows of the result set. This is analogous to the `arrange()` verb in `dplyr`.

`LIMIT` restricts the number of rows in the output. This is similar to the R command `head()`, but somewhat more versatile.

Image Source: Baumer et al. *Modern Data Science with R*.

Remember: Verbs lower in the list must appear after verbs higher in the list when constructing queries.

Here's the `dplyr` to `SQL` translation summary again:

Concept	SQL	R
Filter by rows & columns	`SELECT col1, col2 FROM a WHERE col3 = 'x'`	`a %>% filter(col3 == "x") %>% select(col1, col2)`
Aggregate by rows	`SELECT id, sum(col1) FROM a GROUP BY id`	`a %>% group_by(id) %>% summarize(sum(col1))`
Combine two tables	`SELECT * FROM a JOIN b ON a.id = b.id`	`a %>% inner_join(b, by = c("id" = "id"))`

Table 12.1: Equivalent commands in SQL and R, where a and b are SQL tables and R `data.frames`.

Image Source: Baumer et al. *Modern Data Science with R*

And, it bears repeating, in all caps this time:

IMPORTANT: ALWAYS `LIMIT` YOUR QUERIES, LEST YOU TRY TO FETCH HUNDREDS OF MILLIONS OF RECORDS AND BREAK EVERYTHING FOR EVERYONE

One last reminder: to designate a code chunk "SQL", use `{sql connection=db}` in the chunk options (where `db` is whatever you named your connection in a previous R code chunk) in place of the `r` that is usually there.

`JOIN` (and its variants)

Recall that in the last part of the lab, you constructed a query to show the airlines with at least 1000 flights on June 29th, 2012 in order of the average arrival delay time.

Here is one way you might have written that query:

Code

8 records

carrier	num_flights	avg_delay
US	1192	3.9010
DL	2224	8.0692
OO	1846	12.1056
AA	1480	14.9311
EV	2327	17.1104
MQ	1419	19.1832
WN	3434	24.2266
UA	1644	25.6448

The output above is useful if we know what airlines the two digit IDs correspond to. Some of them are easily recognized; others not so much. It would make our results more useful if we returned the actual full names of the carriers. Since this is information about a carrier not about a flight, it is stored in a different table; namely, `carriers`.

To see the name of the tables available, we can use `SHOW TABLES`; and to see the structure of the `carriers` table, we can use `DESCRIBE`.

Code:

Code

4 records

Tables_in_airlines
airports
carriers
flights
planes

Code

2 records

Field	Type	Null	Key	Default	Extra
carrier	varchar(7)	NO	PRI		
name	varchar(255)	NO			

Looks like `name` is the variable/field we want.

The four join types we learned in `dplyr` have equivalents in `SQL`, which are summarized below. The equivalent of the `by=` argument that we use in `dplyr` to specify the column used to align the tables is the `ON` keyword.

dplyr	SQL
`inner_join()`	`JOIN`
`left_join()`	`LEFT JOIN`
`right_join()`	`RIGHT JOIN`
`full_join()`	`CROSS JOIN`
`by=`	`ON`

To attach the carrier's full name to our result set from above, we could probably use any of these, but I think a left join makes the most sense (since we want to make sure to include each airline in our result set, even if for some reason its full name is missing from the carrier list; which it shouldn't be in this data, but still).

Here's what that would look like as an SQL query:

Code:

Code

8 records

carrier	name	num_flights	avg_delay
US	US Airways Inc.	1192	3.9010
DL	Delta Air Lines Inc.	2224	8.0692
OO	SkyWest Airlines Inc.	1846	12.1056
AA	American Airlines Inc.	1480	14.9311
EV	ExpressJet Airlines Inc.	2327	17.1104
MQ	Envoy Air	1419	19.1832
WN	Southwest Airlines Co.	3434	24.2266
UA	United Air Lines Inc.	1644	25.6448

Merging multiple queries: `UNION` (cf. `bind_rows()`)

If we want to take two queries and merge their output into a single result set, we can simply concatenate the queries with the keyword `UNION`. For example, if there are two particular days of interest for which we want to compute some things, we can write queries for each one and merge them. Suppose for whatever reason we are interested in flights either on June 29th, 2012 or on October 13, 2014.

Code:

Code

Displaying records 1 - 10

carrier	name	year	month	day	num_flights
WN	Southwest Airlines Co.	2012	6	29	3434
WN	Southwest Airlines Co.	2014	10	13	3390
DL	Delta Air Lines Inc.	2014	10	13	2425
EV	ExpressJet Airlines Inc.	2012	6	29	2327
DL	Delta Air Lines Inc.	2012	6	29	2224
EV	ExpressJet Airlines Inc.	2014	10	13	1962
OO	SkyWest Airlines Inc.	2012	6	29	1846
OO	SkyWest Airlines Inc.	2014	10	13	1746
UA	United Air Lines Inc.	2012	6	29	1644
AA	American Airlines Inc.	2014	10	13	1497

Often, perhaps most of the time, there will be a more concise way to write a query like this, by, for example, writing conjunctions or disjunctions of `WHERE` statements and/or adding additional variables to the `GROUP BY` clause. For example, the following is equvalent.

Code

Displaying records 1 - 10

carrier	name	year	month	day	num_flights
WN	Southwest Airlines Co.	2012	6	29	3434
WN	Southwest Airlines Co.	2014	10	13	3390
DL	Delta Air Lines Inc.	2014	10	13	2425
EV	ExpressJet Airlines Inc.	2012	6	29	2327
DL	Delta Air Lines Inc.	2012	6	29	2224
EV	ExpressJet Airlines Inc.	2014	10	13	1962
OO	SkyWest Airlines Inc.	2012	6	29	1846
OO	SkyWest Airlines Inc.	2014	10	13	1746
UA	United Air Lines Inc.	2012	6	29	1644
AA	American Airlines Inc.	2014	10	13	1497

but other times the union we want may not lend itself quite so easily to a concise conjunction or disjunction like this.

Exercises

The following exercises use tables from the `imdb` database instead of the `airlines` database, so you'll need to open a new connection.

Relevant tables are:

1. `name` : records are people (actors, etc.)
2. `title` : records are works (movies, etc.)
3. `cast_info` : records are roles in works, indexed by person
4. `char_name` : records are roles in works, indexed by character

Open the connection first:

Code

Code

Displaying records 1 - 10

Tables_in_imdb
aka_name
aka_title
cast_info
char_name
comp_cast_type
company_name
company_type
complete_cast
info_type
keyword

Exercise 1 Find a movie of your choice in the `title` table. You can use `WHERE <field> LIKE '%<sub string>%'` to get entries that match part of a string.

Sample solution

Code

3 records

id	title	imdb_index	kind_id	production_year	imdb_id	phonetic_code	episode_of_id	season_nr	episode_nr	series_years	md5sum

id	title	imdb_index	kind_id	production_year	imdb_id	phonetic_code	episode_of_id	season_nr	episode_nr	series_years	md5sum
78460	Adults Recat to the Simpsons (30th Anniversary)	NA	7	2017	NA	A3432	78406	NA	NA	NA	2ae09eed7d576cc
70273	(2016-05-18)	NA	7	2016	NA	NA	68058	NA	NA	NA	511dfc14cfff7589d:
60105	(2014-04-11)	NA	7	2014	NA	NA	59138	NA	NA	NA	c6cdce7e667e077

Code

1 records

id	title	imdb_index	kind_id	production_year	imdb_id	phonetic_code	episode_of_id	season_nr	episode_nr	series_years	md5sum
3944746	Memento	NA	1	2000	NA	M53	NA	NA	NA	NA	4cd6aeb9bfe39114

Exercise 2 Find Viola Davis's `person_id` in the `name` table.

Sample solution:

Code

3 records

id	name	imdb_index	imdb_id	gender	name_pcode_cf	name_pcode_nf	surname_pcode	md5sum
235	-Alverio, Esteban Rodriguez	NA	NA	m	A4162	E2315	A416	f5c410bff6839b545d04c531f776e8f2
921	Aaberge, Theodor Olai	NA	NA	m	A1623	T3641	A162	eef277cb705ce78a3b41ed22b5d56292
1698	Aarudra	NA	NA	m	A636	NA	NA	59b3b4b95e3223cb1471724d42c71654

Code

3 records

id	name	imdb_index	imdb_id	gender	name_pcode_cf	name_pcode_nf	surname_pcode	md5sum
2977373	Davis, Viola	II	NA	f	D1214	V4312	D12	cece125825cb648dcadaa25ed1f08cf2
2977372	Davis, Viola	I	NA	f	D1214	V4312	D12	4c0de69942ada8a98c00b22ec7c22ef4
4273298	Davis, Viola	III	NA	NA	D1214	V4312	D12	15ddfd05d8c7d49215107ece595dc2a3

Code

1 records

id	name	imdb_index
2977372	Davis, Viola	I

Exercise 3 Get a list of Viola Davis's roles, using the `cast_info` table, showing her name in the output (requires a join)

Sample solution:

Code

3 records

id	person_id	movie_id	person_role_id	note	nr_order	role_id
1	1	3432997	1	NA	31	1
2	2	1901690	2	NA	NA	1
3	3	4027567	2	NA	25	1

Code

5 records

person_id	movie_id	role_id	person_role_id	name
2977372	3206068	2	418310	Davis, Viola
2977372	3206331	2	637	Davis, Viola
2977372	3206767	2	637	Davis, Viola
2977372	3208095	2	29580	Davis, Viola
2977372	3210140	2	29580	Davis, Viola

Exercise 4 Add the character names to the previous output, by joining with `char_info`.

Sample solution:

Code

Displaying records 1 - 10

id	name	imdb_index	imdb_id	name_pcode_nf	surname_pcode	md5sum
9201	U.S.S. Soldier	NA	NA	U2436	S436	ff2690cc3b3b8486ecc17fc597bc9d47
10644	Count Rood	NA	NA	C5363	R3	10c0099f53636b7a10e36ecd2b12b510
75784	Himself - Chicago Bears	NA	NA	H5241	B62	60ea011c0d7f7d4b12d4af090867b785
97621	Samatan	NA	NA	S535	NA	2dc52183f09c00704d3c65be1ce22bc9
31691	Man mistaken for alex	NA	NA	M5232	A42	7c73d4f4faa6f1865fa5dc953fda635d
22621	Paulie Gigante	NA	NA	P4253	G253	20152b2de417cdca2387e329f12bbc3d
25308	Lil Vito	NA	NA	L413	V3	6330a6bc9354d06814c948ed04453d80
81018	David’s uncle	NA	NA	D1325	U524	f373c29e5da0e060733736345649ea4e
29837	Septimus	NA	NA	S1352	NA	2c0f06f1cbdf55b561680ae231fbb2bd
88446	Emin, Nezahat’in Babasi	NA	NA	E5235	B12	456ab69926d681d9c7e870f5f3b82d66

Code

Displaying records 1 - 10

title	name
Antwone Fisher	Eva May
Beautiful Creatures	Amma
Beyond All Boundaries	Hortense Johnson - Arsenal Worker
Beyond Babyland	Narrator
Black Theater Today: 2005	NA
Blackhat	Carol Barrett
Custody	Martha Schulman
Dark Girls	Herself
Disturbia	Detective Parker
Doubt	Mrs. Miller

Exercise 5 Find Viola Davis’s full filmography, in chronological order. Include each movie’s `title`, `production_year`, and the name of the character that she played.

Sample solution:

Getting credit

DM me your query from exercise 5 (due Thursday 11/21 by class time).