SQL-to-R Code

Prithviraj Lakkakula

Contents

In this post, I will query the data from different data tables in both SQL and R's dplyr with a goal of obtaining the same output. SQL code will be on the your left and R code will be on your right.

This is a work in progress as I continue to add the queries in the coming weeks starting with simple queries to more complicated queries.

Note: In the html file, two columns appear side by side but in pdf version you can view the code only one after the other (not side by side).

Introduction to SQL

Query 1: SELECTing single columns

SQL code

```
SELECT name
FROM people
LIMIT 10;
```

Table 1: Displaying records 1 - 10

name
50 Cent
A. Michael Baldwin
A. Raven Cruz
A.J. Buckley
A.J. DeLucia
A.J. Langer
Aaliyah
Aaron Ashmore
Aaron Hann
Aaron Hill

R code

```
people %>%
  select(name) %>%
  head(n = 10)
```

```
##                  name
## 1             50 Cent
## 2  A. Michael Baldwin
## 3       A. Raven Cruz
## 4        A.J. Buckley
## 5        A.J. DeLucia
## 6         A.J. Langer
## 7             Aaliyah
## 8       Aaron Ashmore
## 9          Aaron Hann
## 10         Aaron Hill
```

Query 2: SELECTing multiple columns

SQL code

```
SELECT name, birthdate
FROM people
LIMIT 10;
```

Table 2: Displaying records 1 - 10

name	birthdate
50 Cent	7/6/75
A. Michael Baldwin	4/4/63
A. Raven Cruz	
A.J. Buckley	2/9/78
A.J. DeLucia	
A.J. Langer	5/22/74
Aaliyah	1/16/79
Aaron Ashmore	10/7/79
Aaron Hann	
Aaron Hill	4/23/83

R code

```
people %>%
  select(name, birthdate) %>%
  head(n = 10)
```

```
##                  name birthdate
## 1             50 Cent    7/6/75
## 2  A. Michael Baldwin    4/4/63
## 3       A. Raven Cruz
## 4        A.J. Buckley    2/9/78
## 5        A.J. DeLucia
## 6         A.J. Langer   5/22/74
## 7             Aaliyah   1/16/79
## 8       Aaron Ashmore   10/7/79
## 9          Aaron Hann
## 10         Aaron Hill   4/23/83
```

Query 3: SELECTing all columns

SQL code

```
SELECT *
FROM people
LIMIT 10;
```

Table 3: Displaying records 1 - 10

id	name	birthdate	deathdate
1	50 Cent	7/6/75	
2	A. Michael Baldwin	4/4/63	
3	A. Raven Cruz		
4	A.J. Buckley	2/9/78	
5	A.J. DeLucia		
6	A.J. Langer	5/22/74	
7	Aaliyah	1/16/79	8/25/01
8	Aaron Ashmore	10/7/79	
9	Aaron Hann		
10	Aaron Hill	4/23/83	

R code

```
people %>%
  head(n = 10)
```

```
##    id               name birthdate deathdate
## 1   1            50 Cent    7/6/75
## 2   2 A. Michael Baldwin    4/4/63
## 3   3      A. Raven Cruz
## 4   4       A.J. Buckley    2/9/78
## 5   5       A.J. DeLucia
## 6   6        A.J. Langer   5/22/74
## 7   7            Aaliyah   1/16/79   8/25/01
## 8   8      Aaron Ashmore   10/7/79
## 9   9         Aaron Hann
## 10 10         Aaron Hill   4/23/83
```

Query 4: Excluding specific columns

SQL code

```
SELECT id, name, birthdate
FROM people
LIMIT 10;
```

Table 4: Displaying records 1 - 10

id	name	birthdate
1	50 Cent	7/6/75
2	A. Michael Baldwin	4/4/63
3	A. Raven Cruz	
4	A.J. Buckley	2/9/78
5	A.J. DeLucia	
6	A.J. Langer	5/22/74
7	Aaliyah	1/16/79
8	Aaron Ashmore	10/7/79

id	name	birthdate
9	Aaron Hann	
10	Aaron Hill	4/23/83

R code

```
people %>%
  select(-deathdate) %>%
  head(n = 10)
```

```
##    id              name birthdate
## 1   1            50 Cent    7/6/75
## 2   2 A. Michael Baldwin    4/4/63
## 3   3      A. Raven Cruz
## 4   4       A.J. Buckley    2/9/78
## 5   5       A.J. DeLucia
## 6   6        A.J. Langer   5/22/74
## 7   7            Aaliyah   1/16/79
## 8   8      Aaron Ashmore   10/7/79
## 9   9         Aaron Hann
## 10 10         Aaron Hill   4/23/83
```

Query 5: SELECTing DISTINCT columns

SQL code

```
SELECT DISTINCT language
FROM films
LIMIT 10;
```

Table 5: Displaying records 1 - 10

language
NA
German
English
Japanese
Danish
Italian
French
Swedish
Russian
None

R code

```
films %>%
  distinct(language) %>%
  head(n = 10)
```

```
## # A tibble: 10 x 1
##    language
##    <chr>
##  1 <NA>
##  2 German
##  3 English
##  4 Japanese
##  5 Danish
##  6 Italian
##  7 French
##  8 Swedish
##  9 Russian
## 10 None
```

SQL code

```
SELECT DISTINCT country
FROM films
LIMIT 10;
```

Table 6: Displaying records 1 - 10

country
USA
Germany
Japan
Denmark
UK
Italy
France
West Germany
Sweden
Soviet Union

R code

```
films %>%
  distinct(country) %>%
  head(n = 10)
```

```
## # A tibble: 10 x 1
##    country
##    <chr>
##  1 USA
##  2 Germany
##  3 Japan
##  4 Denmark
##  5 UK
##  6 Italy
##  7 France
##  8 West Germany
##  9 Sweden
## 10 Soviet Union
```

SQL code

```
SELECT DISTINCT certification
FROM films
LIMIT 10;
```

Table 7: Displaying records 1 - 10

certification
Not Rated
NA
Passed
Unrated
Approved
G
PG
R
PG-13
M

R code

```
films %>%
  distinct(certification) %>%
  head(n = 10)
```

```
## # A tibble: 10 x 1
##    certification
##    <chr>
##  1 Not Rated
##  2 <NA>
##  3 Passed
##  4 Unrated
##  5 Approved
##  6 G
##  7 PG
##  8 R
##  9 PG-13
## 10 M
```

SQL code

```
SELECT DISTINCT role
FROM roles
LIMIT 10;
```

Table 8: 2 records

role
director
actor

R code

```
roles %>%
  distinct(role) %>%
  head(n = 10)
```

```
## # A tibble: 2 x 1
##   role
##   <chr>
## 1 director
## 2 actor
```

Query 6: Learning to COUNT

SQL code: Count the number of rows in `people` table

```
SELECT COUNT(*)
FROM people;
```

Table 9: 1 records

COUNT(*)
8397

R code: Count the number of rows in `people` table

```
people %>%
  count()
```

```
##      n
## 1 8397
```

SQL code: Count the number of birth dates in the `people` table

```
SELECT COUNT(birthdate)
FROM people;
```

Table 10: 1 records

COUNT(birthdate)
8397

R code: Count the number of birth dates in the `people` table

```
people %>% select(birthdate) %>%
  count()
```

```
##      n
## 1 8397
```

SQL code: Count the number of DISTINCT birth dates in the `people` table

```
SELECT COUNT(DISTINCT birthdate)
FROM people;
```

Table 11: 1 records

COUNT(DISTINCT birthdate)
5399

R code: Count the number of DISTINCT birth dates in the `people` table

```
people %>% select(birthdate) %>%
  n_distinct()
```

```
## [1] 5399
```

SQL code: Count the number of DISTINCT languages in the `films` table

```
SELECT COUNT(DISTINCT language)
FROM films;
```

Table 12: 1 records

COUNT(DISTINCT language)
47

R code: Count the number of DISTINCT languages in the `films` table

```
films %>% select(language) %>%
  n_distinct()
```

```
## [1] 48
```

::::

SQL code: Count the number of DISTINCT languages in the `films` table

```
SELECT COUNT(DISTINCT country)
FROM films;
```

Table 13: 1 records

COUNT(DISTINCT country)
64

R code: Count the number of DISTINCT languages in the `films` table

```
films %>% select(country) %>%
  n_distinct()
```

```
## [1] 65
```

Query 7: Filtering of numeric values

SQL code: selects all details for films with a budget over ten thousand dollars

```
SELECT *
FROM films
WHERE budget > 10000
LIMIT 5;
```

Table 14: 5 records

id	title	release_year	country	duration	language	certification	gross	budget
1	Intolerance: Love's Struggle Throughout the Ages	1916	USA	123	NA	Not Rated	NA	385907
2	Over the Hill to the Poorhouse	1920	USA	110	NA	NA	3000000	100000
3	The Big Parade	1925	USA	151	NA	Not Rated	NA	245000
4	Metropolis	1927	Germany	145	German	Not Rated	26435	6000000
6	The Broadway Melody	1929	USA	100	English	Passed	2808000	379000

R code: selects all details for films with a budget over ten thousand dollars

```
films %>%
  filter(budget > 10000) %>%
  head(n = 5)
```

```
## # A tibble: 5 x 9
##      id title       release_year country duration language certification   gross
##   <dbl> <chr>              <dbl> <chr>      <dbl> <chr>    <chr>           <dbl>
## 1     1 Intoleranc~         1916 USA          123 <NA>     Not Rated          NA
## 2     2 Over the H~         1920 USA          110 <NA>     <NA>          3000000
## 3     3 The Big Pa~         1925 USA          151 <NA>     Not Rated          NA
## 4     4 Metropolis          1927 Germany      145 German   Not Rated       26435
## 5     6 The Broadw~         1929 USA          100 English  Passed        2808000
## # ... with 1 more variable: budget <dbl>
```

SQL code: selects all details for all films released in 2016

```
SELECT *
FROM films
WHERE release_year = 2016
LIMIT 5;
```

Table 15: 5 records

id	title	release_year	country	duration	language	certification	gross	budg
4821	10 Cloverfield Lane	2016	USA	104	English	PG-13	71897215	1.5e+
4822	13 Hours	2016	USA	144	English	R	52822418	5.0e+
4823	A Beginner's Guide to Snuff	2016	USA	87	English	NA	NA	N
4824	Airlift	2016	India	130	Hindi	NA	NA	4.4e+
4825	Alice Through the Looking Glass	2016	USA	113	English	PG	76846624	1.7e+

R code: selects all details for all films released in 2016

```
films %>%
  filter(release_year == 2016) %>%
  head(n = 5)
```

```
## # A tibble: 5 x 9
##      id title       release_year country duration language certification   gross
##   <dbl> <chr>              <dbl> <chr>      <dbl> <chr>    <chr>           <dbl>
## 1  4821 10 Cloverf~         2016 USA          104 English  PG-13          7.19e7
## 2  4822 13 Hours            2016 USA          144 English  R              5.28e7
## 3  4823 A Beginner~         2016 USA           87 English  <NA>          NA
## 4  4824 Airlift             2016 India        130 Hindi    <NA>          NA
## 5  4825 Alice Thro~         2016 USA          113 English  PG             7.68e7
## # ... with 1 more variable: budget <dbl>
```

SQL code: selects number of films released before 2000

```
SELECT COUNT(release_year)
FROM films
WHERE release_year <2000;
```

Table 16: 1 records

COUNT(release_year)
1337

R code: selects number of films released before 2000

```
films %>%
  count(release_year < 2000)
```

```
## # A tibble: 3 x 2
##   `release_year < 2000`     n
##   <lgl>                 <int>
## 1 FALSE                  3589
## 2 TRUE                   1337
## 3 NA                       42
```

Query 8: Filtering text

SQL code: gets the titles of all films which were filmed in China

```
SELECT title
FROM films
WHERE country = 'China' -- in PostgreSQL you must use single quotes
LIMIT 5;
```

Table 17: 5 records

title
The Last Emperor
Hero
Hero
House of Flying Daggers
The Promise

R code: gets the titles of all films which were filmed in China

```
films %>%
  filter(country == "China") %>% # here you must use double quotes around text
  select(title) %>%
  head(n = 5)
```

```
## # A tibble: 5 x 1
##   title
##   <chr>
## 1 The Last Emperor
## 2 Hero
## 3 Hero
## 4 House of Flying Daggers
## 5 The Promise
```

SQL code: gets all the details for all French language films

```
SELECT *
FROM films
WHERE language = 'French' -- in PostgreSQL you must use single quotes
LIMIT 5;
```

Table 18: 5 records

id	title	release_year	country	duration	language	certification	gross	budget
108	Une Femme Mariée	1964	France	94	French	NA	NA	1.2e+05
111	Pierrot le Fou	1965	France	110	French	Not Rated	NA	3.0e+05
140	Mississippi Mermaid	1969	France	123	French	R	26893	1.6e+06
423	Subway	1985	France	98	French	R	NA	1.7e+07
662	Les visiteurs	1993	France	107	French	R	700000	5.0e+07

R code: gets all the details for all French language films

```
films %>%
  filter(language == "French") %>% # here you must use double quotes around text
  head(n = 5)
```

```
## # A tibble: 5 x 9
##      id title release_year country duration language certification  gross budget
##   <dbl> <chr>        <dbl> <chr>      <dbl> <chr>    <chr>          <dbl>  <dbl>
## 1   108 Une ~         1964 France        94 French   <NA>              NA  1.2e5
## 2   111 Pier~         1965 France       110 French   Not Rated         NA  3  e5
## 3   140 Miss~         1969 France       123 French   R              26893  1.6e6
## 4   423 Subw~         1985 France        98 French   R                 NA  1.7e7
## 5   662 Les ~         1993 France       107 French   R             700000  5  e7
```

SQL code: Get the name and birth date of the person born on November 11th, 1974.

```
SELECT name birthdate
FROM people
WHERE birthdate = '1974-11-11' -- in PostgreSQL you must use single quotes
LIMIT 5;
```

Table 19: 0 records

birthdate

R code: Get the name and birth date of the person born on November 11th, 1974.

```
people %>%
  select(name, birthdate) %>%
  filter(birthdate == "1974-11-11") %>% # here you must use double quotes around text
  head(n = 5)
```

```
## [1] name      birthdate
## <0 rows> (or 0-length row.names)
```

SQL code: Get the number of Hindi language films

```
SELECT COUNT(language)
FROM films
WHERE language = 'Hindi'; -- in PostgreSQL you must use single quotes
```

Table 20: 1 records

COUNT(language)
28

R code: Get the number of Hindi language films

```
films %>%
  filter(language == "Hindi") %>% # here you must use double quotes around text
  count()
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1    28
```

Query 9: Use WHERE and AND for multiple conditions

SQL code: Gets the titles of films released between 1994 and 2000.

```
SELECT title
FROM films
WHERE release_year > 1994
AND release_year < 2000
LIMIT 5;
```

Table 21: 5 records

title
Ace Ventura: When Nature Calls
Apollo 13
Assassins
Babe
Bad Boys

R code: Gets the titles of films released between 1994 and 2000.

```
films %>%
  filter(release_year > 1994 &
           release_year < 2000) %>%
  select(title) %>%
  head(n = 5)
```

```
## # A tibble: 5 x 1
##   title
##   <chr>
## 1 Ace Ventura: When Nature Calls
## 2 Apollo 13
## 3 Assassins
## 4 Babe
## 5 Bad Boys
```

SQL code: Get the title and release year for all Spanish language films released before 2000

```
SELECT title, release_year
FROM films
WHERE language = 'Spanish'
AND release_year < 2000;
```

Table 22: 3 records

title	release_year
El Mariachi	1992
La otra conquista	1998
Tango	1998

R code: Get the title and release year for all Spanish language films released before 2000

```
films %>%
  filter(language ==  "Spanish" &
           release_year < 2000) %>%
  select(title, release_year)
```

```
## # A tibble: 3 x 2
##   title             release_year
##   <chr>                    <dbl>
## 1 El Mariachi               1992
## 2 La otra conquista         1998
## 3 Tango                     1998
```

SQL code: Get all details for Spanish language films released after 2000, but before 2010.

```
SELECT *
FROM films
WHERE language = 'Spanish'
AND release_year > 2000
AND release_year < 2010
LIMIT 5;
```

Table 23: 5 records

id	title	release_year	country	duration	language	certification	gross	budget
1695	Y Tu Mamá También	2001	Mexico	106	Spanish	R	13622333	2000000
1757	El crimen del padre Amaro	2002	Mexico	118	Spanish	R	5709616	1800000
1807	Mondays in the Sun	2002	Spain	113	Spanish	R	146402	4000000
2173	Live-In Maid	2004	Argentina	83	Spanish	Unrated	NA	800000
2175	Maria Full of Grace	2004	Colombia	101	Spanish	R	6517198	3000000

R code: Get all details for Spanish language films released after 2000, but before 2010.

```
films %>%
  filter(language ==  "Spanish" &
           release_year > 2000 &
           release_year < 2010) %>%
  head(n = 5)
```

```
## # A tibble: 5 x 9
##      id title       release_year country duration language certification   gross
##   <dbl> <chr>              <dbl> <chr>      <dbl> <chr>    <chr>           <dbl>
## 1  1695 Y Tu Mamá ~         2001 Mexico       106 Spanish  R              1.36e7
## 2  1757 El crimen ~         2002 Mexico       118 Spanish  R              5.71e6
## 3  1807 Mondays in~         2002 Spain        113 Spanish  R              1.46e5
## 4  2173 Live-In Ma~         2004 Argent~       83 Spanish  Unrated       NA
## 5  2175 Maria Full~         2004 Colomb~      101 Spanish  R              6.52e6
## # ... with 1 more variable: budget <dbl>
```

Query 10: Use WHERE and OR for multiple conditions

SQL code: Gets all films release in either 1994 or 2000

```
SELECT title
FROM films
WHERE release_year = 1994
OR release_year = 2000
LIMIT 5;
```

Table 24: 5 records

title
3 Ninjas Kick Back
A Low Down Dirty Shame
Ace Ventura: Pet Detective
Baby's Day Out
Beverly Hills Cop III

R code: Gets all films release in either 1994 or 2000

```r
films %>%
  filter(release_year == 1994 |
           release_year == 2000) %>%
  select(title) %>%
  head(n = 5)
```

```
## # A tibble: 5 x 1
##   title
##   <chr>
## 1 3 Ninjas Kick Back
## 2 A Low Down Dirty Shame
## 3 Ace Ventura: Pet Detective
## 4 Baby's Day Out
## 5 Beverly Hills Cop III
```

Query 11: Combining AND and OR with WHERE in SQL

SQL code: Gets all films release in either 1994 or 2000

```sql
SELECT title
FROM films
WHERE (release_year = 1994 OR release_year = 1995)
AND (certification = 'PG' OR certification = 'R')
LIMIT 5;
```

Table 25: 5 records

title
3 Ninjas Kick Back
A Low Down Dirty Shame
Baby's Day Out
Beverly Hills Cop III
Bullets Over Broadway

R code: Gets all films release in either 1994 or 2000

```r
films %>%
  filter((release_year == 1994 | release_year == 1995) &
           (certification == "PG" | certification == "R")) %>%
  select(title) %>%
  head(n = 5)
```

```
## # A tibble: 5 x 1
##   title
##   <chr>
## 1 3 Ninjas Kick Back
## 2 A Low Down Dirty Shame
## 3 Baby's Day Out
## 4 Beverly Hills Cop III
## 5 Bullets Over Broadway
```

SQL code: Get the title and release year for films released in the 90s.

```
SELECT title, release_year
FROM films
WHERE release_year >= 1994
AND release_year < 2000
LIMIT 5;
```

Table 26: 5 records

title	release_year
3 Ninjas Kick Back	1994
A Low Down Dirty Shame	1994
Ace Ventura: Pet Detective	1994
Baby's Day Out	1994
Beverly Hills Cop III	1994

R code: Get the title and release year for films released in the 90s.

```
films %>%
  filter(release_year >= 1994 & release_year < 2000)  %>%
  select(title, release_year) %>%
  head(n = 5)
```

```
## # A tibble: 5 x 2
##   title                       release_year
##   <chr>                              <dbl>
## 1 3 Ninjas Kick Back                  1994
## 2 A Low Down Dirty Shame              1994
## 3 Ace Ventura: Pet Detective          1994
## 4 Baby's Day Out                      1994
## 5 Beverly Hills Cop III               1994
```

SQL code: filter the records to only include French or Spanish language films in 1990s.

```
SELECT title, release_year
FROM films
WHERE (release_year >= 1990 AND release_year < 2000)
AND (language = 'French' OR language = 'Spanish')
LIMIT 5;
```

Table 27: 5 records

title	release_year
El Mariachi	1992
Les visiteurs	1993
The Horseman on the Roof	1995
When the Cat's Away	1996
The Chambermaid on the Titanic	1997

R code: filter the records to only include French or Spanish language films in 199Os.

```r
films %>%
  filter((release_year >= 1990 & release_year < 2000) &
           (language == "French" | language == "Spanish")) %>%
  select(title, release_year) %>%
  head(n = 5)
```

```
## # A tibble: 5 x 2
##   title                           release_year
##   <chr>                                  <dbl>
## 1 El Mariachi                             1992
## 2 Les visiteurs                           1993
## 3 The Horseman on the Roof                1995
## 4 When the Cat's Away                     1996
## 5 The Chambermaid on the Titanic          1997
```

SQL code: filter the records to only include French or Spanish language films in 1990s with gross greater than 2 million.

```sql
SELECT title, release_year
FROM films
WHERE (release_year >= 1990 AND release_year < 2000)
AND (language = 'French' OR language = 'Spanish')
AND gross > 2000000
LIMIT 5;
```

Table 28: 2 records

title	release_year
El Mariachi	1992
The Red Violin	1998

R code: filter the records to only include French or Spanish language films in 199Os with gross greater than 2 million.

```r
films %>%
  filter((release_year >= 1990 & release_year < 2000) &
           (language == "French" | language == "Spanish") &
           gross > 2000000) %>%
  select(title, release_year) %>%
  head(n = 5)
```

```
## # A tibble: 2 x 2
##   title          release_year
##   <chr>                 <dbl>
## 1 El Mariachi            1992
## 2 The Red Violin         1998
```

Query 12: `BETWEEN` with `AND` and `OR`

SQL code: `BETWEEN` is used to filter values in a specified range and it is always inclusive

```
SELECT title, release_year
FROM films
WHERE release_year
BETWEEN 1994 AND 2000
LIMIT 5;
```

Table 29: 5 records

title	release_year
3 Ninjas Kick Back	1994
A Low Down Dirty Shame	1994
Ace Ventura: Pet Detective	1994
Baby's Day Out	1994
Beverly Hills Cop III	1994

R code: `between` is used in filter our values based on a range of values

```
films %>%
  filter(between(release_year, 1994, 2000))%>%
  select(title, release_year) %>%
  head(n = 5)
```

```
## # A tibble: 5 x 2
##   title                       release_year
##   <chr>                              <dbl>
## 1 3 Ninjas Kick Back                  1994
## 2 A Low Down Dirty Shame              1994
## 3 Ace Ventura: Pet Detective          1994
## 4 Baby's Day Out                      1994
## 5 Beverly Hills Cop III               1994
```

SQL code: `BETWEEN` when used with both `AND` and `OR`

```
SELECT title, release_year
FROM films
WHERE release_year BETWEEN 1990 AND 2000
AND language = 'Spanish'
AND budget > 100000000
LIMIT 5;
```

Table 30: 1 records

title	release_year
Tango	1998

R code:

```r
films %>%
  filter(between(release_year, 1990, 2000) &
           language == "Spanish" &
           budget > 100000000)%>%
  select(title, release_year) %>%
  head(n = 5)
```

```
## # A tibble: 1 x 2
##   title release_year
##   <chr>        <dbl>
## 1 Tango         1998
```

Query 13: `WHERE IN`

SQL code: If you have many either or conditions on a column, we need to specify several `OR` conditions using `WHERE`. Instead we can used `WHERE IN`.

```sql
SELECT title, release_year
FROM films
WHERE release_year IN (1990, 2000)
AND duration > 120
LIMIT 5;
```

Table 31: 5 records

title	release_year
Dances with Wolves	1990
Die Hard 2	1990
Ghost	1990
Goodfellas	1990
Mo' Better Blues	1990

R code:

```r
films %>%
  filter(release_year %in% c(1990, 2000) &
           duration > 120) %>%
  select(title, release_year) %>%
  head(n = 5)
```

```
## # A tibble: 5 x 2
##   title              release_year
##   <chr>                     <dbl>
## 1 Dances with Wolves         1990
## 2 Die Hard 2                 1990
## 3 Ghost                      1990
## 4 Goodfellas                 1990
## 5 Mo' Better Blues           1990
```

SQL code:

```
SELECT title, release_year
FROM films
WHERE language IN ('English', 'Spanish', 'French')
LIMIT 5;
```

Table 32: 5 records

title	release_year
The Broadway Melody	1929
Hell's Angels	1930
A Farewell to Arms	1932
42nd Street	1933
She Done Him Wrong	1933

R code:

```
films %>%
  filter(language %in% c("English", "Spanish", "French")) %>%
  select(title, release_year) %>%
  head(n = 5)
```

```
## # A tibble: 5 x 2
##   title                release_year
##   <chr>                       <dbl>
## 1 The Broadway Melody          1929
## 2 Hell's Angels                1930
## 3 A Farewell to Arms           1932
## 4 42nd Street                  1933
## 5 She Done Him Wrong           1933
```

SQL code:

```
SELECT title, certification
FROM films
WHERE certification IN ('NC-17', 'R')
LIMIT 5;
```

Table 33: 5 records

title	certification
Psycho	R
A Fistful of Dollars	R
Rosemary's Baby	R
Mississippi Mermaid	R
The Wild Bunch	R

R code:

```r
films %>%
  filter(certification %in% c("NC-17", "R")) %>%
  select(title, certification) %>%
  head(n = 5)
```

```
## # A tibble: 5 x 2
##   title                certification
##   <chr>                <chr>
## 1 Psycho               R
## 2 A Fistful of Dollars R
## 3 Rosemary's Baby      R
## 4 Mississippi Mermaid  R
## 5 The Wild Bunch       R
```

Query 14: Intro to `NULL` and `IS NULL`

SQL code: In SQL, `NULL` represents a missing or unknown value. One can check `NULL` values using `IS NULL`. For example, to count the number of missing birth dates in the people table;

```sql
SELECT COUNT(*)
FROM people
WHERE birthdate IS NULL
LIMIT 5;
```

Table 34: 1 records

COUNT(*)
0

R code:

```r
people %>%
  filter(is.na(birthdate)) %>% count() %>%
  head(n = 5)
```

```
##   n
## 1 0
```

SQL code: Sometimes, you will want to filter out missing values so you only get results which are not NULL. To do this, you can use the IS NOT NULL operator.

```sql
SELECT name
FROM people
WHERE birthdate IS NOT NULL
LIMIT 5;
```

Table 35: 5 records

name
50 Cent
A. Michael Baldwin
A. Raven Cruz
A.J. Buckley
A.J. DeLucia

R code:

```
people %>%
  filter(!is.na(birthdate)) %>%
  select(name) %>%
  head(n = 5)
```

```
##                 name
## 1            50 Cent
## 2 A. Michael Baldwin
## 3      A. Raven Cruz
## 4       A.J. Buckley
## 5       A.J. DeLucia
```

SQL code: Get the number of films which don't have a language associated with them

```
SELECT COUNT(*)
FROM films
WHERE language IS NULL
LIMIT 5;
```

Table 36: 1 records

COUNT(*)
11

R code:

```
films %>%
  filter(is.na(language)) %>% count() %>%
  head(n = 5)
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1    11
```

Query 15: Aggregate Functions

Essentially SQL has some aggregate functions that you can perform on your variables.

SQL code: The following query gives the average value from the budget column of the films table.

```
SELECT AVG(budget) AS budget_ave
FROM films
LIMIT 5;
```

Table 37: 1 records

budget_ave
39902826

R code:

```
films %>%
  summarize(budget_avg = mean(budget, na.rm = TRUE)) %>%
  head(n = 5)
```

```
## # A tibble: 1 x 1
##   budget_avg
##        <dbl>
## 1  39902826.
```

SQL code: The following query gives the maximum value from the budget column of the films table.

```
SELECT MAX(budget) AS budget_max
FROM films
LIMIT 5;
```

Table 38: 1 records

budget_max
12215500000

R code:

```
films %>%
  summarize(budget_max = max(budget, na.rm = TRUE)) %>%
  head(n = 5)
```

```
## # A tibble: 1 x 1
##    budget_max
##         <dbl>
## 1 12215500000
```

SQL code: The following query gives the minimum value from the budget column of the films table.

```
SELECT MIN(budget) AS budget_min
FROM films
LIMIT 5;
```

Table 39: 1 records

budget_min
218

R code:

```
films %>%
  summarize(budget_min = min(budget, na.rm = TRUE)) %>%
  head(n = 5)
```

```
## # A tibble: 1 x 1
##   budget_min
##        <dbl>
## 1        218
```

SQL code: The following query gives the minimum value from the budget column of the films table.

```
SELECT SUM(budget) AS budget_sum
FROM films
LIMIT 5;
```

Table 40: 1 records

budget_sum
1.81079e+11

R code:

```
films %>%
  summarize(budget_sum = sum(budget, na.rm = TRUE)) %>%
  head(n = 5)
```

```
## # A tibble: 1 x 1
##     budget_sum
##          <dbl>
## 1 181079025606
```

Query 16: Combining Aggregate functions with WHERE

SQL code: Gets the total budget of movies made in the year 2010 or later.

```
SELECT SUM(budget)
FROM films
WHERE release_year >=2010
LIMIT 5;
```

Table 41: 1 records

SUM(budget)
54913578440

R code:

```
films %>%
  filter(release_year >= 2010) %>%
  summarize(budget_sum = sum(budget, na.rm = TRUE)) %>%
  head(n = 5)
```

```
## # A tibble: 1 x 1
##    budget_sum
##         <dbl>
## 1 54913578440
```

SQL code: Get the amount grossed by the best performing film between 2000 and 2012, inclusive.

```
SELECT MAX(gross)
FROM films
WHERE release_year BETWEEN 2000 AND 2012
LIMIT 5;
```

Table 42: 1 records

MAX(gross)
760505847

R code:

```
films %>%
  filter(between(release_year, 2000, 2012)) %>%
  summarize(gross_max = max(gross, na.rm = TRUE)) %>%
  head(n = 5)
```

```
## # A tibble: 1 x 1
##   gross_max
##       <dbl>
## 1 760505847
```

SQL code: Get the number of decades the films table covers. Alias the result as number_of_decades.

```
SELECT (MAX(release_year) - MIN(release_year)) / 10 AS number_of_decades
FROM films
LIMIT 5;
```

Table 43: 1 records

number_of_decades
10

R code:

```
films %>%
  summarize(number_of_decades = (max(release_year, na.rm = TRUE) -
                                   min(release_year, na.rm = TRUE)) / 10) %>%
  head(n = 5)
```

```
## # A tibble: 1 x 1
##   number_of_decades
##               <dbl>
## 1                10
```

Query 17: Sorting and Grouping

SQL code: In SQL, the `ORDER BY` keyword is used to sort results in ascending or descending order according to the values of one or more columns. By default `ORDER BY` will sort in ascending order. In order to sort the results in descending order, you can use `DESC` order.

For example, the following query gives you the titles of films sorted by release year, from newest to oldest.

```
SELECT title
FROM films
ORDER BY release_year DESC
LIMIT 5;
```

Table 44: 5 records

title
10 Cloverfield Lane
13 Hours
A Beginner's Guide to Snuff
Airlift
Alice Through the Looking Glass

R code: Analogous verb in R for sorting is `arrange`. For `arrange` the default is ascending but for getting descending order `desc` can be used like in SQL.

```
films %>%
  arrange(desc(release_year)) %>%
  select(title) %>%
  head(n = 5)
```

```
## # A tibble: 5 x 1
##   title
##   <chr>
## 1 10 Cloverfield Lane
## 2 13 Hours
## 3 A Beginner's Guide to Snuff
## 4 Airlift
## 5 Alice Through the Looking Glass
```

SQL code: For text values, ORDER BY sorts a column by default alphabetically (A to Z)

```
SELECT name
FROM people
ORDER BY name
LIMIT 5;
```

Table 45: 5 records

name
50 Cent
<83>mile Gaudreault
<83>milie Dequenne
<83>ric Tessier
<83>tienne Faure

R code:

```
people %>%
  arrange(name) %>%
  select(name) %>%
  head(n = 5)
```

```
##                   name
## 1 \x83mile Gaudreault
## 2  \x83milie Dequenne
## 3     \x83ric Tessier
## 4    \x83tienne Faure
## 5      \xe7lex Angulo
```

SQL code: Get the title of films released in 2000 or 2012, in the order they were released.

```
SELECT title
FROM films
WHERE release_year IN (2000, 2012)
ORDER BY release_year
LIMIT 5;
```

Table 46: 5 records

title
102 Dalmatians
28 Days
3 Strikes
Aberdeen
All the Pretty Horses

R code: Get the title of films released in 2000 or 2012, in the order they were released.

```
films %>%
  filter(release_year %in% c(2000, 2012)) %>%
  arrange(release_year) %>%
  select(title) %>%
  head(n = 5)
```

```
## # A tibble: 5 x 1
##   title
##   <chr>
## 1 102 Dalmatians
## 2 28 Days
## 3 3 Strikes
## 4 Aberdeen
## 5 All the Pretty Horses
```

SQL code: Get all details for all films except those released in 2015 and order them by duration.

```
SELECT *
FROM films
WHERE release_year < 2015
OR release_year > 2015
ORDER BY duration
LIMIT 5;
```

Table 47: 5 records

id	title	release_year	country	duration	language	certification	gro
1398	Hum To Mohabbat Karega	2000	India	NA	Hindi	NA	N
2326	Dil Jo Bhi Kahey...	2005	India	NA	English	NA	1293
2712	The Naked Ape	2006	USA	NA	English	NA	N
3208	Black Water Transit	2009	USA	NA	English	NA	N
3504	Harry Potter and the Deathly Hallows: Part I	2010	UK	NA	English	NA	N

R code: Get all details for all films except those released in 2015 and order them by duration.

```
films %>%
  filter(release_year < 2015 | release_year > 2015) %>%
  arrange(duration) %>%
  head(n = 5)
```

```
## # A tibble: 5 x 9
##      id title release_year country duration language certification  gross budget
##   <dbl> <chr>        <dbl> <chr>      <dbl> <chr>    <chr>          <dbl>  <dbl>
## 1  2926 The ~         2007 USA            7 English  <NA>              NA  1.3e4
## 2  4098 Vess~         2012 USA           14 English  <NA>              NA NA
## 3  2501 Wal-~         2005 USA           20 English  Not Rated         NA  1.5e6
## 4   566 Mari~         1990 USA           34 English  <NA>          333658  3.4e4
## 5  2829 Jesu~         2007 USA           35 English  <NA>              NA NA
```

Query 18: SORTING single columns by DESC

To order results in descending order, you can put the keyword `DESC` after your `ORDER BY`.

SQL code: Gets all the names in the people table, in reverse alphabetical order.

```
SELECT name
FROM people
ORDER BY name DESC
LIMIT 5;
```

Table 48: 5 records

name
Zuhair Haddad
Zubaida Sahar
Zoran Lisinac
Zooey Deschanel
Zohra Segal

R code: Gets all the names in the people table, in reverse alphabetical order.

```
people %>%
  arrange(desc(name)) %>%
  select(name) %>%
  head(n = 5)
```

```
##              name
## 1   Zuhair Haddad
## 2   Zubaida Sahar
## 3   Zoran Lisinac
## 4 Zooey Deschanel
## 5     Zohra Segal
```

Query 19: SORTING multiple columns

ORDER BY can also be used to sort on multiple columns. It will sort by the first column specified, then sort by the next, then the next, and so on. For example, the following query sorts birth dates first (oldest to newest) and then sorts on the names in alphabetical order. THE ORDER OF COLUMNS IS IMPORTANT!

Note: To specify multiple columns you separate the column names with a comma.

SQL code: Gets all the names in the people table, in reverse alphabetical order.

```
SELECT birthdate, name
FROM people
ORDER BY birthdate, name
LIMIT 5;
```

Table 49: 5 records

birthdate	name
	<83>mile Gaudreault
	<83>tienne Faure
	A. Raven Cruz
	A.J. DeLucia
	Aaron Hann

R code: Gets all the names in the people table, in reverse alphabetical order.

```
people %>%
  arrange(birthdate, name) %>%
  select(birthdate, name) %>%
  head(n = 5)
```

```
##   birthdate                name
## 1           \x83mile Gaudreault
## 2              \x83tienne Faure
## 3                 A. Raven Cruz
## 4                  A.J. DeLucia
## 5                    Aaron Hann
```

SQL code: Get the `names` and `birthdates` of people ordered by name and birth date. Notice how the second column you order on only steps in when the first column is not decisive to tell the order. The second column acts as a tie breaker

```
SELECT name, birthdate
FROM people
ORDER BY name, birthdate
LIMIT 5;
```

Table 50: 5 records

name	birthdate
50 Cent	7/6/75
<83>mile Gaudreault	
<83>milie Dequenne	8/29/81
<83>ric Tessier	1883-05-28
<83>tienne Faure	

R code: Get the `names` and `birthdates` of people ordered by name and birth date. Notice how the second column you order on only steps in when the first column is not decisive to tell the order. The second column acts as a tie breaker

```
people %>%
  arrange(name, birthdate) %>%
  select(name, birthdate) %>%
  head(n = 5)
```

```
##                   name  birthdate
## 1 \x83mile Gaudreault
## 2  \x83milie Dequenne    8/29/81
## 3     \x83ric Tessier 1883-05-28
## 4    \x83tienne Faure
## 5      \xe7lex Angulo    4/12/53
```

Query 20: GROUP BY

- We saw `ORDER BY` for sorting. Here we use `GROUP BY` for aggregating results.
- Commonly, `GROUP BY` is used with aggregate functions like `COUNT()` or `MAX()`.
- Note that `GROUP BY` always goes after the `FROM` clause!
- Overall, `GROUP BY` is performing operations within groups.
- Warning: SQL will return an error if you try to `SELECT` a field that is not in your `GROUP BY` clause without using it to calculate some kind of value about the entire group.
- Note that you can combine `GROUP BY` with `ORDER BY` to group your results, calculate something about them, and then order your results.
- **NOTE**: `ORDER BY` always goes after `GROUP BY`

SQL code: Gets the release year and count of films released in each year

```
SELECT release_year, COUNT(*)
FROM films
GROUP BY release_year
LIMIT 5;
```

Table 51: 5 records

release_year	COUNT(*)
NA	42
1916	1
1920	1
1925	1
1927	1

R code: Gets the release year and count of films released in each year

```
films %>%
  group_by(release_year) %>%
  select(release_year) %>% count() %>%
  head(n = 5)
```

```
## # A tibble: 5 x 2
## # Groups:   release_year [5]
```

```
##   release_year     n
##          <dbl> <int>
## 1         1916     1
## 2         1920     1
## 3         1925     1
## 4         1927     1
## 5         1929     2
```

SQL code: Get the release year and average duration of all films, grouped by release year

```
SELECT release_year, AVG(duration)
FROM films
GROUP BY release_year
LIMIT 5;
```

Table 52: 5 records

release_year	AVG(duration)
NA	77.43902
1916	123.00000
1920	110.00000
1925	151.00000
1927	145.00000

R code: Get the release year and average duration of all films, grouped by release year

```
films %>%
  group_by(release_year) %>% summarize(duration_avg = mean(duration)) %>%
  select(release_year, duration_avg) %>%
  head(n = 5)
```

```
## # A tibble: 5 x 2
##   release_year duration_avg
##          <dbl>        <dbl>
## 1         1916          123
## 2         1920          110
## 3         1925          151
## 4         1927          145
## 5         1929          105
```

SQL code: Get the release year and largest budget for all films, grouped by release year

```
SELECT release_year, MAX(budget)
FROM films
GROUP BY release_year
LIMIT 5;
```

Table 53: 5 records

release_year	MAX(budget)
NA	15000000
1916	385907
1920	100000
1925	245000
1927	6000000

R code: Get the release year and largest budget for all films, grouped by release year

```
films %>%
  group_by(release_year) %>% summarize(budget_max = max(budget)) %>%
  select(release_year, budget_max) %>%
  head(n = 5)
```

```
## # A tibble: 5 x 2
##   release_year budget_max
##          <dbl>      <dbl>
## 1         1916     385907
## 2         1920     100000
## 3         1925     245000
## 4         1927    6000000
## 5         1929          NA
```

SQL code: Get the country, release year, and lowest amount grossed per release year per country. Order your results by country and release year.

```
SELECT country, release_year, MIN(gross)
FROM films
GROUP BY release_year, country
ORDER BY country, release_year
LIMIT 5;
```

Table 54: 5 records

country	release_year	MIN(gross)
NA	NA	NA
NA	2014	NA
Afghanistan	2003	1127331
Argentina	2000	1221261
Argentina	2004	304124

R code: Get the country, release year, and lowest amount grossed per release year per country. Order your results by country and release year.

```
films %>%
  group_by(release_year, country) %>% summarize(gross_min = min(gross)) %>%
  arrange(country, release_year) %>%
  select(country, release_year, gross_min) %>%
  head(n = 5)
```

```
## # A tibble: 5 x 3
## # Groups:   release_year [5]
##   country     release_year gross_min
##   <chr>              <dbl>     <dbl>
## 1 Afghanistan         2003   1127331
## 2 Argentina           2000   1221261
## 3 Argentina           2004        NA
## 4 Argentina           2009  20167424
## 5 Aruba               1998  10076136
```

Joining Data

Query 21: INNER JOINS

- `INNER JOIN` and `LEFT JOIN` are probably the two most common joins.
- `id` field is called `key` since it is used to reference both the left and right tables
- `INNER JOIN` gathers the value columns from both the left and right tables with common `key ids` in both.

::::

SQL code: INNER JOIN and SELECTing all columns

```
SELECT city.name AS city, country.region AS region
FROM cities AS city
  -- Inner join to countries
  INNER JOIN countries AS country
  -- Match on the country codes
  ON city.country_code = country.code
LIMIT 5;
```

Table 55: 5 records

city	region
Abidjan	Western Africa
Abu Dhabi	Middle East
Abuja	Western Africa
Accra	Western Africa
Addis Ababa	Eastern Africa

R code: INNER JOIN and SELECTing all columns

```
cities %>%
  inner_join(countries, by = c("country_code" = "code")) %>%
  select(city = name, region) %>%
  head(n = 5)
```

```
## # A tibble: 5 x 2
##   city        region
##   <chr>       <chr>
```

```
## 1 Abidjan     Western Africa
## 2 Abu Dhabi   Middle East
## 3 Abuja       Western Africa
## 4 Accra       Western Africa
## 5 Addis Ababa Eastern Africa
```

Query 22: INNER JOINS - 2

::::

SQL code:

```
SELECT c.code AS country_code, year, inflation_rate
FROM countries AS c
  INNER JOIN economies AS e
  ON c.code = e.code
LIMIT 5;
```

Table 56: 5 records

country_code	year	inflation_rate
AFG	2010	2.179
AFG	2015	-1.549
NLD	2010	0.932
NLD	2015	0.220
ALB	2010	3.605

R code:

```
countries %>%
  inner_join(economies, by = "code") %>%
  select(country_code = code, year, inflation_rate) %>%
  head(n = 5)
```

```
## # A tibble: 5 x 3
##   country_code  year inflation_rate
##   <chr>        <dbl>          <dbl>
## 1 AFG           2010          2.18
## 2 AFG           2015         -1.55
## 3 NLD           2010          0.932
## 4 NLD           2015          0.22
## 5 ALB           2010          3.60
```

Query 23: INNER JOIN via USING

::::

SQL code:

```
SELECT c.continent, l.name AS language, l.official
FROM countries AS c
  INNER JOIN languages AS l
  USING (code)
LIMIT 5;
```

Table 57: 5 records

continent	language	official
Asia	Dari	1
Asia	Other	0
Asia	Pashto	1
Asia	Turkic	0
Europe	Dutch	1

R code:

```
countries %>%
  inner_join(languages, by = "code") %>%
  select(continent, language = name, official) %>%
  head(n = 5)
```

```
## # A tibble: 5 x 3
##   continent language official
##   <chr>     <chr>    <lgl>
## 1 Asia      Dari     TRUE
## 2 Asia      Pashto   TRUE
## 3 Asia      Turkic   FALSE
## 4 Asia      Other    FALSE
## 5 Europe    Dutch    TRUE
```

Query 24: CASE WHEN and THEN

- Often it's useful to look at a numerical field not as raw data, but instead as being in different categories or groups.
- You can use `CASE` with `WHEN`, `THEN`, `ELSE`, and `END` to define a new grouping field.

::::

SQL code:

```
SELECT continent, code, surface_area,
    CASE WHEN surface_area > 2000000 THEN 'large'
        WHEN surface_area > 350000 AND surface_area < 2000000 THEN 'medium'
        ELSE 'small' END
        AS geosize_group
FROM countries
LIMIT 5;
```

Table 58: 5 records

continent	code	surface_area	geosize_group
Asia	AFG	652090	medium
Europe	NLD	41526	small
Europe	ALB	28748	small
Africa	DZA	2381740	large
Oceania	ASM	199	small

R code:

```
countries %>%
  mutate(geosize_group = case_when(
    surface_area > 2000000 ~ "large",
    surface_area > 350000 & surface_area < 2000000 ~ "medium",
    surface_area < 350000 ~ "small"
  )) %>%
  select(continent, code, surface_area, geosize_group) %>%
  head(n = 5)
```

```
## # A tibble: 5 x 4
##   continent code  surface_area geosize_group
##   <chr>     <chr>        <dbl> <chr>
## 1 Asia      AFG         652090 medium
## 2 Europe    NLD          41526 small
## 3 Europe    ALB          28748 small
## 4 Africa    DZA        2381740 large
## 5 Oceania   ASM            199 small
```