Intermediate SQL

August 11, 2023

1 DataCamp - Intermediate SQL

- This Notebook contains solutions to all the exercises of the course Intermediate SQL. In this Notebook first the csv files are loaded from the datasets table as pandas dataframes, and then SQL Queries are run on them using the **duckdb** library.The syntax of running the SQL query in duckdb is the following:
  duckdb.query(**SQL-Query-here**).to_df().
- Backslashes (\) were put in place in order to move to the next line in the cell and are not part of the query
- All solutions are verified

2 Loading Datasets

[65]:
```
import pandas as pd
import pandas as pd
import duckdb
```

[67]:
```
films = pd.read_csv('./datasets/films.csv',\
                      header=None, names=['id', 'title', 'release_year',␣
 ↪'country', 'duration', 'language', 'certification',\
                                          'gross', 'budget'])

people = pd.read_csv('./datasets/people.csv',\
                      header=None, names=['id', 'name', 'birthdate',␣
 ↪'deathdate'])

reviews = pd.read_csv('./datasets/reviews.csv',\
                      header=None, names=['id', 'film_id', 'num_user',␣
 ↪'num_critic', 'imdb_score', 'num_votes',\
                                          'facebook_likes'])


roles = pd.read_csv('./datasets/roles.csv',\
                      header=None, names=['id', 'film_id', 'person_id', 'role'])
```

[68]:
```
films.head()
```

```
[68]:    id                                           title  release_year  \
      0   1  Intolerance: Love's Struggle Throughout the Ages        1916.0
      1   2                    Over the Hill to the Poorhouse        1920.0
      2   3                                    The Big Parade        1925.0
      3   4                                        Metropolis        1927.0
      4   5                                     Pandora's Box        1929.0

         country  duration language certification      gross     budget
      0      USA     123.0      NaN     Not Rated        NaN   385907.0
      1      USA     110.0      NaN           NaN  3000000.0   100000.0
      2      USA     151.0      NaN     Not Rated        NaN   245000.0
      3  Germany     145.0   German     Not Rated    26435.0  6000000.0
      4  Germany     110.0   German     Not Rated     9950.0        NaN
```

```
[69]: people.head()
```

```
[69]:    id               name   birthdate deathdate
      0   1            50 Cent  1975-07-06       NaN
      1   2  A. Michael Baldwin  1963-04-04       NaN
      2   3       A. Raven Cruz         NaN       NaN
      3   4        A.J. Buckley  1978-02-09       NaN
      4   5        A.J. DeLucia         NaN       NaN
```

```
[70]: reviews.head()
```

```
[70]:      id  film_id  num_user  num_critic  imdb_score  num_votes  facebook_likes
      0  3934    588.0     432.0         7.1      203461      46000             NaN
      1  3405    285.0     267.0         6.4      149998          0             NaN
      2   478     65.0      29.0         3.2        8465        491             NaN
      3    74     83.0      25.0         7.6        7071        930             NaN
      4  1254   1437.0     224.0         8.0      241030      13000             NaN
```

```
[71]: roles.head()
```

```
[71]:    id  film_id  person_id      role
      0   1        1       1630  director
      1   2        1       4843     actor
      2   3        1       5050     actor
      3   4        1       8175     actor
      4   5        2       3000  director
```

2.1 Chapter 1: Selecting Data

2.1.1 Querying Database

Count the number of records in the people table

```
[72]: duckdb.query(\
                 "SELECT COUNT(*) as count_records from people;"\
                ).to_df()
```

```
[72]:    count_records
      0           8397
```

Count the number of records with a birthdate in the people table, aliasing the result as count_birthdate.

```
[73]: duckdb.query(\
                 "SELECT COUNT(birthdate) as count_birthdate from people;"\
                ).to_df()
```

```
[73]:    count_birthdate
      0             6152
```

Count the records for languages and countries in the films table; alias as count_languages and count_countries.

```
[74]: duckdb.query(\
                 "SELECT COUNT(language) as count_languages\
                 , COUNT(country) as count_countries from films;"\
                ).to_df()
```

```
[74]:    count_languages  count_countries
      0             4955             4966
```

Return the unique countries represented in the films table using DISTINCT.

```
[75]: duckdb.query(\
                 "SELECT DISTINCT(country) FROM films"\
                ).to_df()
```

```
[75]:      country
      0        USA
      1    Germany
      2      Japan
      3    Denmark
      4         UK
      ..       …
      60  Slovenia
      61  Pakistan
      62     Chile
      63    Panama
      64  Slovakia

      [65 rows x 1 columns]
```

Return the number of unique countries represented in the films table, aliased as count_distinct_countries.

```
[76]: duckdb.query(\
                 "SELECT COUNT(DISTINCT(country)) as count_distinct_countries FROM␣
       ↪films"\
                ).to_df()
```

```
[76]:    count_distinct_countries
      0                        64
```

2.1.2 SQL Query Execution Order

If a column has space, it should be written in " ", eg."facebook likes"

2.2 Chapter 2: Filtering Records

Topics Covered: - Filtering Numbers - Multiple criteria - Filtering text - NULL values

2.2.1 Filtering Numbers

The WHERE clause allows you to filter based on text and numeric values in a table using comparison operators.

Select film_ids and imdb_score with an imdb_score over 7.0

```
[77]: duckdb.query(\
                 "select film_id, imdb_score from reviews\
                 where imdb_score > 7"\
                ).to_df()
```

```
[77]:       film_id  imdb_score
      0       588.0      203461
      1       285.0      149998
      2        65.0        8465
      3        83.0        7071
      4      1437.0      241030
      …         …          …
      4955      2.0          75
      4956    514.0      181472
      4957     85.0       29738
      4958    118.0       29591
      4959   1123.0      387508

      [4960 rows x 2 columns]
```

Select the film_id and facebook_likes of the first ten records with less than 1000 likes from the reviews table.

[78]:
```
duckdb.query(\
             "select film_id, facebook_likes from reviews\
             where facebook_likes < 1000\
             limit 10"\
            ).to_df()
```

[78]:
```
Empty DataFrame
Columns: [film_id, facebook_likes]
Index: []
```

Count how many records have a num_votes of at least 100,000; use the alias films_over_100K_votes.

[79]:
```
duckdb.query(\
             "select count(*) as films_over_100K_votes\
             from reviews\
             where num_votes >= 100000"\
            ).to_df()
```

[79]:
```
   films_over_100K_votes
0                     46
```

Select and count the language field using the alias count_spanish. Apply a filter to select only Spanish from the language field.

[80]:
```
duckdb.query(\
             "select count(language) as count_spanish\
             from films\
             where language = 'Spanish'"\
            ).to_df()
```

[80]:
```
   count_spanish
0             40
```

2.2.2 Multiple criteria

Select the title and release_year for all German-language films released before 2000.

[81]:
```
duckdb.query(\
             "select title, release_year\
             from films\
             where language = 'German' and release_year < 2000"\
            ).to_df()
```

[81]:
```
                                title  release_year
0                          Metropolis        1927.0
1                       Pandora's Box        1929.0
2   The Torture Chamber of Dr. Sadism        1967.0
```

```
3                          Das Boot        1981.0
4                      Run Lola Run        1998.0
5                    Aimee & Jaguar        1999.0
```

Update the query from the previous step to show German-language films released after 2000 rather than before.

[82]:
```
duckdb.query(\
             "select title, release_year\
             from films\
             where language = 'German' and release_year > 2000"\
            ).to_df()
```

[82]:
```
                        title  release_year
0              Good Bye Lenin!        2003.0
1                     Downfall        2004.0
2                 Summer Storm        2004.0
3          The Lives of Others        2006.0
4   The Baader Meinhof Complex        2008.0
5                     The Wave        2008.0
6                        Cargo        2009.0
7                 Soul Kitchen        2009.0
8             The White Ribbon        2009.0
9                            3        2010.0
10             Animals United        2010.0
11          Buen DÃa, RamÃ³n       2013.0
```

Select all details for German-language films released after 2000 but before 2010 using only WHERE and AND.

[83]:
```
duckdb.query(\
             "select *\
             from films\
             where release_year <2010 and release_year > 2000 and language =␣
 ↪'German'"\
            ).to_df()
```

[83]:
```
     id                       title  release_year      country  duration  \
0  1952             Good Bye Lenin!        2003.0      Germany     121.0
1  2130                    Downfall        2004.0      Germany     178.0
2  2224                Summer Storm        2004.0      Germany      98.0
3  2709         The Lives of Others        2006.0      Germany     137.0
4  3100  The Baader Meinhof Complex        2008.0      Germany     184.0
5  3143                    The Wave        2008.0      Germany     107.0
6  3220                       Cargo        2009.0  Switzerland     112.0
7  3346                Soul Kitchen        2009.0      Germany      99.0
8  3412            The White Ribbon        2009.0      Germany     144.0
```

```
  language certification       gross      budget
0   German             R   4063859.0   4800000.0
1   German             R   5501940.0  13500000.0
2   German             R     95016.0   2700000.0
3   German             R  11284657.0   2000000.0
4   German             R    476270.0  20000000.0
5   German           NaN         NaN   5000000.0
6   German           NaN         NaN   4500000.0
7   German           NaN    274385.0   4000000.0
8   German             R   2222647.0  12000000.0
```

Select the title and release_year for films released in 1990 or 1999 using only WHERE and OR.

[84]:
```
duckdb.query(\
             "select title, release_year\
             from films\
             where release_year = 1990 or release_year = 1999"\
            ).to_df()
```

[84]:
```
                              title  release_year
0                     Arachnophobia        1990.0
1      Back to the Future Part III        1990.0
2                    Child's Play 2        1990.0
3                Dances with Wolves        1990.0
4                   Days of Thunder        1990.0
..                              …             …
193                Twin Falls Idaho        1999.0
194  Universal Soldier: The Return        1999.0
195                   Varsity Blues        1999.0
196                  Wild Wild West        1999.0
197                  Wing Commander        1999.0

[198 rows x 2 columns]
```

In the above query, Filter the records to only include English or Spanish-language films.

[85]:
```
duckdb.query(\
             "select title, release_year\
             from films\
             where (release_year = 1990 or release_year = 1999) and (language =␣
↪'English' or language = 'Spanish') "\
            ).to_df()
```

[85]:
```
                              title  release_year
0                     Arachnophobia        1990.0
1      Back to the Future Part III        1990.0
```

```
2                     Child's Play 2        1990.0
3                 Dances with Wolves        1990.0
4                    Days of Thunder        1990.0
..                               …             …
191                 Twin Falls Idaho        1999.0
192  Universal Soldier: The Return        1999.0
193                    Varsity Blues        1999.0
194                   Wild Wild West        1999.0
195                   Wing Commander        1999.0

[196 rows x 2 columns]
```

In the above query, Finally, restrict the query to only return films worth more than $2,000,000 gross.

[86]:
```
duckdb.query(\
             "select title, release_year\
             from films\
             where (release_year = 1990 or release_year = 1999) and (language =␣
 ↪'English' or language = 'Spanish') \
             and gross > 2000000"\
            ).to_df()
```

[86]:
```
                               title  release_year
0                       Arachnophobia        1990.0
1       Back to the Future Part III        1990.0
2                     Child's Play 2        1990.0
3                 Dances with Wolves        1990.0
4                    Days of Thunder        1990.0
..                               …             …
163                         Trippin'        1999.0
164  Universal Soldier: The Return        1999.0
165                    Varsity Blues        1999.0
166                   Wild Wild West        1999.0
167                   Wing Commander        1999.0

[168 rows x 2 columns]
```

Select the title and release_year of all films released between 1990 and 2000 (inclusive) using BETWEEN.

[87]:
```
duckdb.query(\
             "select title, release_year\
             from films\
             where release_year between 1990 and 2000"\
            ).to_df()
```

[87]:
```
                               title  release_year
0                     Arachnophobia        1990.0
1     Back to the Future Part III        1990.0
2                     Child's Play 2        1990.0
3                Dances with Wolves        1990.0
4                   Days of Thunder        1990.0
..                               …           …
952                         Whipped        2000.0
953                    Woman on Top        2000.0
954                     Wonder Boys        2000.0
955                           X-Men        2000.0
956            You Can Count on Me        2000.0

[957 rows x 2 columns]
```

Build on your previous query to select only films with a budget over $100 million.

[88]:
```
duckdb.query(\
             "select title, release_year\
             from films\
             where (release_year between 1990 and 2000) and budget > 100000000"\
            ).to_df()
```

[88]:
```
                                         title  release_year
0                   Terminator 2: Judgment Day        1991.0
1                                   True Lies        1994.0
2                                  Waterworld        1995.0
3                              Batman & Robin        1997.0
4                                Dante's Peak        1997.0
5                            Princess Mononoke        1997.0
6                      Speed 2: Cruise Control        1997.0
7                           Starship Troopers        1997.0
8                                     Titanic        1997.0
9                          Tomorrow Never Dies        1997.0
10                                 A Bug's Life        1998.0
11                                        Antz        1998.0
12                                  Armageddon        1998.0
13    Les couloirs du temps: Les visiteurs II        1998.0
14                               Lethal Weapon 4        1998.0
15                                       Tango        1998.0
16                                Godzilla 2000        1999.0
17  Star Wars: Episode I - The Phantom Menace        1999.0
18                               Stuart Little        1999.0
19    The Messenger: The Story of Joan of Arc        1999.0
20                        The World Is Not Enough        1999.0
21                              Wild Wild West        1999.0
22                                    Dinosaur        2000.0
```

```
23                                    Gladiator        2000.0
24             How the Grinch Stole Christmas        2000.0
25                     Mission: Impossible II        2000.0
26                                 The Patriot        2000.0
27                           The Perfect Storm        2000.0
```

Using the above query, Now, restrict the query to only return Spanish-language films.

[89]:
```
duckdb.query(\
             "select title, release_year\
             from films\
             where (release_year between 1990 and 2000) and budget > 100000000␣
 ↪and language = 'Spanish'"\
            ).to_df()
```

[89]:
```
   title  release_year
0  Tango        1998.0
```

In the abive query, Finally, amend the query to include all Spanish-language or French-language films with the same criteria.

[90]:
```
duckdb.query(\
             "select title, release_year\
             from films\
             where (release_year between 1990 and 2000) and budget > 100000000␣
 ↪and (language = 'Spanish' or language = 'French')"\
            ).to_df()
```

[90]:
```
                                      title  release_year
0                                     Tango        1998.0
1  Les couloirs du temps: Les visiteurs II        1998.0
```

2.2.3 Filtering text

To filter a pattern in text instead of the entire text, we can use: - LIKE - NOT LIKE - IN

Select the names of all people whose names begin with 'B'.

[91]:
```
duckdb.query(\
             "select name from people where name like 'B%'"\
            ).to_df()
```

[91]:
```
              name
0       B.J. Novak
1     Babak Najafi
2      Babar Ahmed
3   Bahare Seddiqi
4         Bai Ling
```

```
..              …
440   Buster Keaton
441   Busy Philipps
442     Buzz Aldrin
443    Byron Howard
444      Byron Mann

[445 rows x 1 columns]
```

Select the names of people whose names have 'r' as the second letter.

```
[92]: duckdb.query(\
                  "select name from people where name like '_r%'"\
                 ).to_df()
```

```
[92]:                   name
      0             Ara Celi
      1        Aramis Knight
      2    Arben Bajraktaraj
      3     Arcelia RamÃrez
      4           Archie Kao
      ..                 …
      525        Troy Garity
      526        Troy Miller
      527         Troy Nixey
      528     Ursula Andress
      529      Wray Crawford

      [530 rows x 1 columns]
```

Select the names of people whose names don't start with 'A'.

```
[93]: duckdb.query(\
                  "select name from people where name not like 'A%'"\
                 ).to_df()
```

```
[93]:                      name
      0                   50 Cent
      1             Ã lex Angulo
      2      Ã lex de la Iglesia
      3            Ã ngela Molina
      4                B.J. Novak
      …                       …
      7763           Zohra Segal
      7764       Zooey Deschanel
      7765          Zoran Lisinac
      7766          Zubaida Sahar
      7767         Zuhair Haddad
```

```
[7768 rows x 1 columns]
```

Select the title and release_year of all films released in 1990 or 2000 that were longer than two hours.

```
[94]: duckdb.query(\
                   "select title, release_year from films where release_year in␣
      ↪(1990, 2000) and duration > 120"\
                 ).to_df()
```

```
[94]:                         title  release_year
      0           Dances with Wolves        1990.0
      1                   Die Hard 2        1990.0
      2                        Ghost        1990.0
      3                   Goodfellas        1990.0
      4             Mo' Better Blues        1990.0
      5                 Pretty Woman        1990.0
      6      The Godfather: Part III        1990.0
      7       The Hunt for Red October        1990.0
      8         All the Pretty Horses        2000.0
      9                Almost Famous        2000.0
      10                  Bamboozled        2000.0
      11                   Cast Away        2000.0
      12                   Chocolat        2000.0
      13          Dancer in the Dark        2000.0
      14             Erin Brockovich        2000.0
      15           Finding Forrester        2000.0
      16                        Fiza        2000.0
      17                   Gladiator        2000.0
      18        Gone in Sixty Seconds        2000.0
      19           Keeping the Faith        2000.0
      20           Love & Basketball        2000.0
      21                Men of Honor        2000.0
      22       Mission: Impossible II        2000.0
      23                Pandaemonium        2000.0
      24              Pay It Forward        2000.0
      25                     Pollock        2000.0
      26               Proof of Life        2000.0
      27                      Quills        2000.0
      28              Reindeer Games        2000.0
      29               Space Cowboys        2000.0
      30                 The 6th Day        2000.0
      31              The Contender        2000.0
      32              The Family Man        2000.0
      33           The House of Mirth        2000.0
      34   The Legend of Bagger Vance        2000.0
```

```
35                 The Patriot        2000.0
36            The Perfect Storm       2000.0
37                Thirteen Days       2000.0
38                      Traffic       2000.0
39               Vertical Limit       2000.0
40           What Lies Beneath        2000.0
41              What Women Want       2000.0
```

Select the title and language of all films in English, Spanish, or French using IN.

[95]:
```
duckdb.query(\
             "select title, language from films where language in ('English',␣
 ↪'Spanish', 'French')"\
            ).to_df()
```

[95]:
```
                                title language
0                  The Broadway Melody  English
1                        Hell's Angels  English
2                   A Farewell to Arms  English
3                         42nd Street  English
4                   She Done Him Wrong  English
…                                   …        …
4742                    The Blue Room   French
4743     Animal Kingdom: Let's go Ape   French
4744                        Evolution   French
4745  They Will Have to Kill Us First   French
4746                    Irreplaceable   French

[4747 rows x 2 columns]
```

Select the title, certification and language of all films certified NC-17 or R that are in English, Italian, or Greek.

[96]:
```
duckdb.query(\
             "select title, certification, language from films where \
             certification in ('NC-17', 'R') and language in ('English',␣
 ↪'Italian', 'Greek')"\
            ).to_df()
```

[96]:
```
                          title certification language
0                Pink Flamingos         NC-17  English
1                 The Evil Dead         NC-17  English
2                     Showgirls         NC-17  English
3                       Orgazmo         NC-17  English
4                        L.I.E.         NC-17  English
…                             …            …        …
2001             The Neon Demon             R  English
2002            The Perfect Match           R  English
```

```
2003  The Purge: Election Year            R  English
2004                  The Veil            R  English
2005                  Triple 9            R  English

[2006 rows x 3 columns]
```

- **Count the unique titles from the films database and use the alias provided (nineties_english_films_for_teens).**
- **Filter to include only movies with a release_year from 1990 to 1999, inclusive.**
- **Add another filter narrowing your query down to English-language films.**
- **Add a final filter to select only films with 'G', 'PG', 'PG-13' certifications.**

[97]:
```
duckdb.query(\
             "select count(distinct(title)) as nineties_english_films_for_teens␣
 ↪from films\
             where (release_year between 1990 and 1999)\
             and language = 'English'\
             and certification in ('G', 'PG', 'PG-13')"\
            ).to_df()
```

[97]:
```
   nineties_english_films_for_teens
0                               310
```

2.2.4 NULL Values

Select the title of every film that doesn't have a budget associated with it and use the alias no_budget_info.

[98]:
```
duckdb.query(\
             "select title as no_budget_info from films where budget is null"\
            ).to_df()
```

[98]:
```
                no_budget_info
0                Pandora's Box
1        The Prisoner of Zenda
2                The Blue Bird
3                        Bambi
4                   State Fair
..                         …
425                Unforgotten
426                      Wings
427                 Wolf Creek
428          Wuthering Heights
429  Yu-Gi-Oh! Duel Monsters

[430 rows x 1 columns]
```

Count the number of films with a language associated with them and use the alias

count_language_known.

```
[99]: duckdb.query(\
                  "select count(*) as count_language_known from films where language␣
       ↪is not null"\
                 ).to_df()
```

```
[99]:    count_language_known
      0                  4955
```

2.3 Chapter 3: Aggregate Functions

Topics Covered:

- Summarizing Data
- Summarizing Subsets
- Aliasing and Arithmetic

2.3.1 Summarizing Data

- Count, MIN and MAX can work on various data types

Use the SUM() function to calculate the total duration of all films and alias with total_duration.

```
[100]: duckdb.query(\
                   "select sum(duration) as total_duration from films"\
                  ).to_df()
```

```
[100]:    total_duration
       0        534882.0
```

Calculate the average duration of all films and alias with average_duration.

```
[101]: duckdb.query(\
                   "select avg(duration) as average_duration from films"\
                  ).to_df()
```

```
[101]:    average_duration
       0        107.947931
```

Find the most recent release_year in the films table, aliasing as latest_year.

```
[102]: duckdb.query(\
                   "select max(release_year) as latest_year from films"\
                  ).to_df()
```

```
[102]:    latest_year
       0       2016.0
```

Find the duration of the shortest film and use the alias shortest_film.

```
[103]: duckdb.query(\
                    "select min(duration) as shortest_film from films"\
                   ).to_df()
```

```
[103]:    shortest_film
       0            7.0
```

2.3.2 Summarizing Subsets

Use SUM() to calculate the total gross for all films made in the year 2000 or later, and use the alias total_gross.

```
[104]: duckdb.query(\
                   "select sum(gross) as total_gross from films\
                   where release_year >= 2000"\
                   ).to_df()
```

```
[104]:     total_gross
       0  1.509009e+11
```

Calculate the average amount grossed by all films whose titles start with the letter 'A' and alias with avg_gross_A.

```
[105]: duckdb.query(\
                   "select avg(gross) as avg_gross_A from films where title like 'A%'"\
                   ).to_df()
```

```
[105]:     avg_gross_A
       0  4.789324e+07
```

Calculate the lowest gross film in 1994 and use the alias lowest_gross.

```
[106]: duckdb.query(\
                   "select min(gross) as lowest_gross from films where release_year =␣
         ↪1994"\
                   ).to_df()
```

```
[106]:    lowest_gross
       0      125169.0
```

Calculate the highest gross film between 2000 and 2012, inclusive, and use the alias highest_gross

```
[107]: duckdb.query(\
                   "select max(gross) as highest_gross from films where release_year␣
         ↪between 2000 and 2012"\
                   ).to_df()
```

```
[107]:    highest_gross
        0    760505847.0
```

Calculate the average facebook_likes to one decimal place and assign to the alias, avg_facebook_likes.

```
[108]: duckdb.query(\
                   "select round(avg(facebook_likes), 1) as avg_facebook_likes from␣
         ↪reviews"\
                   ).to_df()
```

```
[108]:    avg_facebook_likes
        0                NaN
```

Calculate the average budget from the films table, aliased as avg_budget_thousands, and round to the nearest thousand.

```
[109]: duckdb.query(\
                   "select round(avg(budget), -3) as avg_budget_thousands from films"\
                   ).to_df()
```

```
[109]:    avg_budget_thousands
        0            39903000.0
```

2.3.3 Aliasing and arithmetic

- Arithmetic on same datatype gives the same datatype in SQL, eg. int divided by int gives int

Select the title and duration in hours for all films and alias as duration_hours; since the current durations are in minutes, you'll need to divide duration by 60.0.

```
[110]: duckdb.query(\
                   "select title, duration / 60.0 as duration_hours from films"\
                   ).to_df()
```

```
[110]:                                                  title  duration_hours
        0     Intolerance: Love's Struggle Throughout the Ages        2.050000
        1                        Over the Hill to the Poorhouse        1.833333
        2                                        The Big Parade        2.516667
        3                                            Metropolis        2.416667
        4                                          Pandora's Box        1.833333
        …                                                    …              …
        4963                                        Unforgotten        0.750000
        4964                                              Wings        0.500000
        4965                                         Wolf Creek             NaN
        4966                                   Wuthering Heights        2.366667
        4967                            Yu-Gi-Oh! Duel Monsters        0.400000

        [4968 rows x 2 columns]
```

Calculate the percentage of people who are no longer alive and alias the result as percentage_dead.

```
[111]: duckdb.query(\
                 "SELECT count(deathdate) * 100.0 / count(id) AS percentage_dead␣
        ↪FROM people;"\
                 ).to_df()
```

```
[111]:    percentage_dead
       0         9.372395
```

Find how many decades (period of ten years) the films table covers by using MIN() and MAX(); alias as number_of_decades.

```
[112]: duckdb.query(\
                 "SELECT (MAX(release_year) - MIN(release_year)) / 10.0 AS␣
        ↪number_of_decades FROM films;"\
                 ).to_df()
```

```
[112]:    number_of_decades
       0               10.0
```

2.4 Sorting and Grouping

2.4.1 Sorting Results

Select the name of each person in the people table, sorted alphabetically.

```
[113]: duckdb.query(\
                 "SELECT name from people order by name"\
                 ).to_df()
```

```
[113]:                     name
       0                50 Cent
       1     A. Michael Baldwin
       2          A. Raven Cruz
       3           A.J. Buckley
       4           A.J. DeLucia
       …                      …
       8392        Ã"scar Jaenada
       8393   Ã‰mile Gaudreault
       8394    Ã‰milie Dequenne
       8395        Ã‰ric Tessier
       8396      Ã‰tienne Faure

       [8397 rows x 1 columns]
```

Select the title and duration for every film, from longest duration to shortest.

```
[114]: duckdb.query(\
                "SELECT title, duration from films order by duration desc"\
                ).to_df()
```

```
[114]:                          title  duration
       0                       Carlos     334.0
       1            Blood In, Blood Out     330.0
       2                 Heaven's Gate     325.0
       3       The Legend of Suriyothai     300.0
       4                      Das Boot     293.0
       …                             …         …
       4963                      Barfi       NaN
       4964                    Destiny       NaN
       4965          Karachi se Lahore       NaN
       4966           Romantic Schemer       NaN
       4967                 Wolf Creek       NaN

       [4968 rows x 2 columns]
```

Select the release_year, duration, and title of films ordered by their release year and duration, in that order.

```
[115]: duckdb.query(\
                "SELECT release_year, duration, title from films order by␣
        ↪release_year, duration"\
                ).to_df()
```

```
[115]:       release_year  duration                                           title
       0           1916.0     123.0  Intolerance: Love's Struggle Throughout the Ages
       1           1920.0     110.0                  Over the Hill to the Poorhouse
       2           1925.0     151.0                                  The Big Parade
       3           1927.0     145.0                                      Metropolis
       4           1929.0     100.0                             The Broadway Melody
       …                …         …                                               …
       4963           NaN     197.0                               Deadline Gallipoli
       4964           NaN     240.0                                            Emma
       4965           NaN     286.0                                     The Company
       4966           NaN     334.0                                          Carlos
       4967           NaN       NaN                                      Wolf Creek

       [4968 rows x 3 columns]
```

Select the certification, release_year, and title from films ordered first by certification (alphabetically) and second by release year, starting with the most recent year.

```
[116]: duckdb.query(\
                "SELECT certification, release_year, title from films order by␣
        ↪certification, release_year desc"\
            ).to_df()
```

```
[116]:      certification  release_year                         title
       0         Approved        1967.0            You Only Live Twice
       1         Approved        1967.0                    Point Blank
       2         Approved        1967.0                  In Cold Blood
       3         Approved        1966.0                   Torn Curtain
       4         Approved        1966.0  The Good, the Bad and the Ugly
       …              …             …                             …
       4963           NaN           NaN                        Trapped
       4964           NaN           NaN                        Twisted
       4965           NaN           NaN                  Unforgettable
       4966           NaN           NaN                    Unforgotten
       4967           NaN           NaN                          Wings

       [4968 rows x 3 columns]
```

2.4.2 Grouping Data

- To summarize / aggregate data for a specific group of results, we use group by

Select the release_year and count of films released in each year aliased as film_count.

```
[117]: duckdb.query(\
                    "SELECT release_year, count(*) as film_count from films group by␣
         ↪release_year"\
                   ).to_df()
```

```
[117]:    release_year  film_count
       0         1916.0           1
       1         1920.0           1
       2         1925.0           1
       3         1927.0           1
       4         1929.0           2
       ..           …          …
       87        2013.0         236
       88        2014.0         252
       89        2015.0         226
       90        2016.0         106
       91           NaN          42

       [92 rows x 2 columns]
```

Select the release_year and average duration aliased as avg_duration of all films, grouped by release_year.

```
[118]: duckdb.query(\
            "SELECT release_year, avg(duration) as avg_duration from films␣
↪group by release_year"\
            ).to_df()
```

```
[118]:     release_year  avg_duration
      0          1916.0    123.000000
      1          1920.0    110.000000
      2          1925.0    151.000000
      3          1927.0    145.000000
      4          1929.0    105.000000
      ..            ...           ...
      87         2013.0    108.140426
      88         2014.0    105.426295
      89         2015.0    106.098214
      90         2016.0    109.632075
      91            NaN     77.439024

      [92 rows x 2 columns]
```

Select the release_year, country, and the maximum budget aliased as max_budget for each year and each country; sort your results by release_year and country.

```
[119]: duckdb.query(\
            "SELECT release_year, country, max(budget) as max_budget from films␣
↪group by release_year, country\
            order by release_year, country"\
            ).to_df()
```

```
[119]:      release_year  country  max_budget
      0           1916.0      USA    385907.0
      1           1920.0      USA    100000.0
      2           1925.0      USA    245000.0
      3           1927.0  Germany   6000000.0
      4           1929.0  Germany         NaN
      ..             ...      ...         ...
      500            NaN   Poland         NaN
      501            NaN   Sweden         NaN
      502            NaN       UK         NaN
      503            NaN      USA   5000000.0
      504            NaN      NaN         NaN

      [505 rows x 3 columns]
```

2.4.3 Filtering grouped data (HAVING)

- **Select country from the films table, and get the distinct count of certification aliased as certification_count.**
- **Group the results by country.**

- **Filter the unique count of certifications to only results greater than 10.**

[120]:
```
duckdb.query(\
            "SELECT country, count(distinct(certification)) as␣
↪certification_count from films\
            group by country\
            having count(distinct(certification)) > 10"\
            ).to_df()
```

[120]:
```
  country  certification_count
0     USA                   12
```

- **Select the country and the average budget as average_budget, rounded to two decimal, from films.**
- **Group the results by country.**
- **Filter the results to countries with an average budget of more than one billion (1000000000).**
- **Sort by descending order of the average_budget.**

[121]:
```
duckdb.query(\
            "SELECT country, round(avg(budget), 2) as average_budget from films\
            group by country\
            having round(avg(budget), 2) > 1000000000\
            order by average_budget desc"\
            ).to_df()
```

[121]:
```
       country  average_budget
0  South Korea    1.383960e+09
1      Hungary    1.260000e+09
```

Bringing it all together - Final Exercise

- **Select the release_year for each film in the films table, filter for records released after 1990, and group by release_year.**

[122]:
```
duckdb.query(\
            "SELECT release_year from films where release_year > 1990 group by␣
↪release_year"\
            ).to_df()
```

[122]:
```
    release_year
0         1991.0
1         1992.0
2         1993.0
3         1994.0
4         1995.0
5         1996.0
6         1997.0
```

```
7          1998.0
8          1999.0
9          2000.0
10         2001.0
11         2002.0
12         2003.0
13         2004.0
14         2005.0
15         2006.0
16         2007.0
17         2008.0
18         2009.0
19         2010.0
20         2011.0
21         2012.0
22         2013.0
23         2014.0
24         2015.0
25         2016.0
```

Modify the query to include the average budget aliased as avg_budget and average gross aliased as avg_gross for the results we have so far.

```
[123]: duckdb.query(\
                   "SELECT release_year, avg(budget) as avg_budget, avg(gross) as␣
        ↪avg_gross\
                   from films\
                   where release_year > 1990 group by release_year"\
                   ).to_df()
```

```
[123]:    release_year    avg_budget     avg_gross
       0        1991.0  2.517655e+07  5.384450e+07
       1        1992.0  2.598203e+07  6.366520e+07
       2        1993.0  2.072979e+07  4.530209e+07
       3        1994.0  2.901377e+07  5.939567e+07
       4        1995.0  3.277500e+07  4.490952e+07
       5        1996.0  3.162061e+07  4.204417e+07
       6        1997.0  5.942449e+07  4.479377e+07
       7        1998.0  4.046000e+07  3.837701e+07
       8        1999.0  3.898178e+07  3.807218e+07
       9        2000.0  3.493138e+07  4.217263e+07
       10       2001.0  3.768731e+07  4.325572e+07
       11       2002.0  3.259851e+07  4.351115e+07
       12       2003.0  3.720865e+07  4.872775e+07
       13       2004.0  4.686534e+07  4.072653e+07
       14       2005.0  7.032394e+07  4.115914e+07
       15       2006.0  9.396893e+07  3.923786e+07
```

```
16        2007.0  3.527113e+07  4.626750e+07
17        2008.0  4.180489e+07  4.457351e+07
18        2009.0  3.707329e+07  4.620744e+07
19        2010.0  4.609466e+07  4.990833e+07
20        2011.0  3.777525e+07  4.578584e+07
21        2012.0  4.133182e+07  6.287353e+07
22        2013.0  4.051904e+07  5.615836e+07
23        2014.0  3.532580e+07  6.241214e+07
24        2015.0  3.929833e+07  7.257330e+07
25        2016.0  5.664274e+07  7.692404e+07
```

Modify the query once more so that only years with an average budget of greater than 60 million are included.

[124]:
```
duckdb.query(\
            "SELECT release_year, AVG(budget) AS avg_budget, AVG(gross) AS␣
 ↪avg_gross\
            FROM films\
            WHERE release_year > 1990\
            GROUP BY release_year\
            having avg(budget) > 60000000"\
            ).to_df()
```

[124]:
```
   release_year    avg_budget     avg_gross
0        2005.0  7.032394e+07  4.115914e+07
1        2006.0  9.396893e+07  3.923786e+07
```

Finally, order the results from the highest average gross and limit to one.

[125]:
```
duckdb.query(\
            "SELECT release_year, AVG(budget) AS avg_budget, AVG(gross) AS␣
 ↪avg_gross\
            FROM films\
            WHERE release_year > 1990\
            GROUP BY release_year\
            having avg(budget) > 60000000\
            order by avg_gross desc\
            limit 1"\
            ).to_df()
```

[125]:
```
   release_year    avg_budget     avg_gross
0        2005.0  7.032394e+07  4.115914e+07
```

[ ]: