Homework 4

Khang Thai

2025-05-18

```
library(RMariaDB)
```

```
## Warning: package 'RMariaDB' was built under R version 4.3.3
```

```
library(DBI)
```

```
## Warning: package 'DBI' was built under R version 4.3.3
```

```
con <- dbConnect(RMariaDB::MariaDB(),
host = "relational.fel.cvut.cz",
port = 3306,
username = "guest",
password = "ctu-relational",
dbname = "imdb_ijs"
)
```

Question 1)

(a)

```
dbGetQuery(con, "
  SELECT name, year
  FROM movies
  WHERE year = (SELECT MIN(year) FROM movies);
")
```

```
##                             name year
## 1         Roundhay Garden Scene 1888
## 2 Traffic Crossing Leeds Bridge 1888
```

(b)

```
dbGetQuery(con, "
  SELECT name, year, rank
  FROM movies
  WHERE year = (SELECT MAX(year)
                FROM movies
                WHERE rank IS NOT NULL)
  ORDER BY rank DESC
  LIMIT 10;
")
```

```
##                                      name year rank
## 1                        Dawn of the Friend 2004  9.9
```

```
## 2  Magical Time Traveling Thugtastic Jug, The 2004  9.8
## 3            Dimensia Minds Trilogy: The Reds 2004  9.8
## 4                                    Accordon 2004  9.7
## 5                             Tomorrow's Memoir 2004  9.7
## 6                           Milton Is a Shitbag 2004  9.7
## 7                                    Cashback 2004  9.7
## 8                           Devils Are Dreaming 2004  9.7
## 9                                   Sanhedrin 2004  9.7
## 10                            Earl's Your Uncle 2004  9.6
```

(c)

```
genre_rating <- dbGetQuery(con, "
  SELECT genre, COUNT(*) AS num_movies, AVG(rank) AS avg_rating
  FROM movies_genres
  JOIN movies ON movie_id = id
  WHERE rank IS NOT NULL
  GROUP BY genre
  ORDER BY avg_rating DESC
")
genre_rating
```

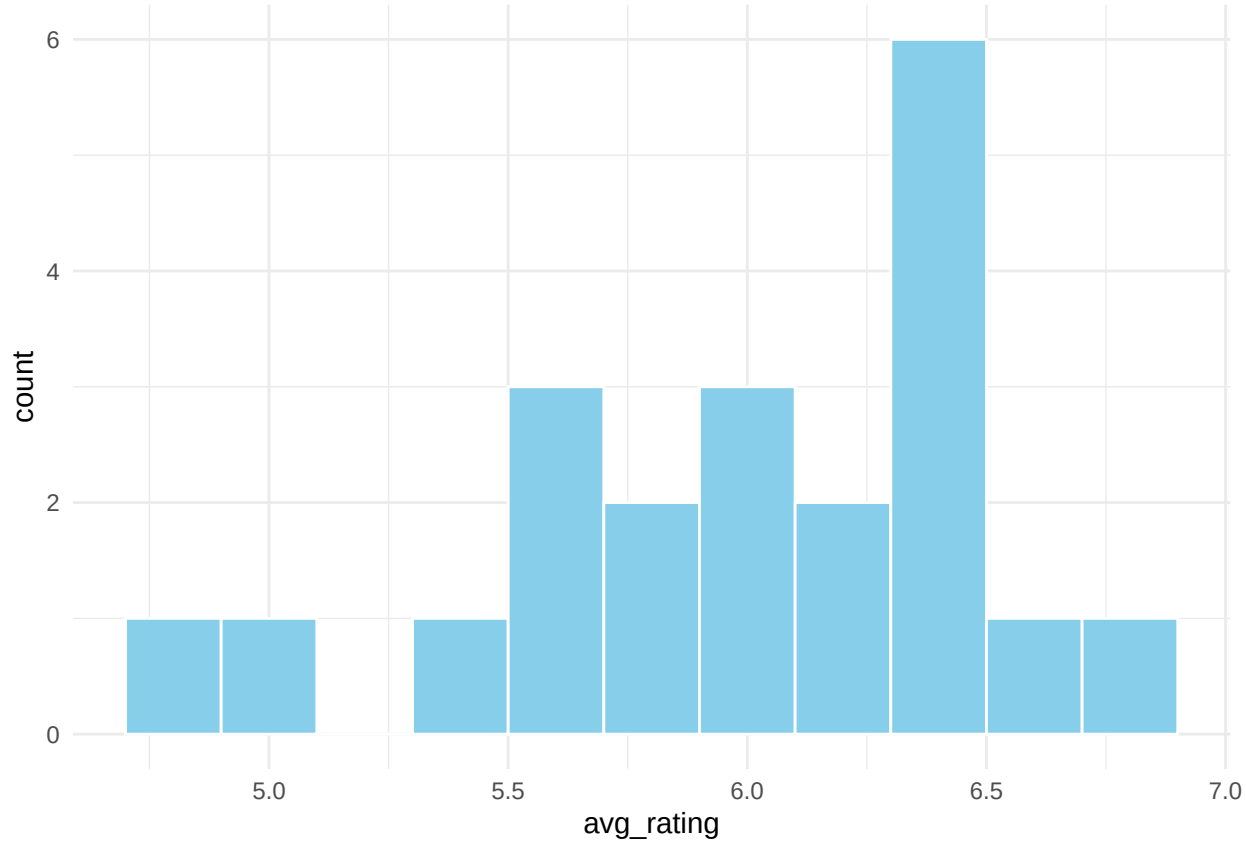
```
##          genre num_movies avg_rating
## 1    Film-Noir        396   6.701768
## 2    Animation       4039   6.557836
## 3  Documentary       3753   6.496829
## 4        Adult         82   6.428049
## 5        Music        817   6.415912
## 6        Short       9525   6.394257
## 7          War       1722   6.376945
## 8       Family       4281   6.314623
## 9      Romance       5189   6.156967
## 10       Drama      23269   6.137767
## 11     Musical       2284   6.087347
## 12     Mystery       1891   5.933210
## 13      Comedy      19105   5.905480
## 14       Crime       4272   5.860042
## 15     Fantasy       1907   5.846880
## 16     Western       2225   5.652090
## 17   Adventure       3493   5.581792
## 18    Thriller       5189   5.527481
## 19      Action       5449   5.339237
## 20      Sci-Fi       2416   5.008651
## 21      Horror       3895   4.756406
```

(d)

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

```
ggplot(genre_rating, aes(x = avg_rating)) +
  geom_histogram(binwidth = 0.2, fill = "skyblue", color = "white") +
  theme_minimal()
```



Question 2)

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.3.3
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
director_tbl <- tbl(con, "directors")
movies_directors_tbl <- tbl(con, "movies_directors")

result <- movies_directors_tbl %>%
  inner_join(director_tbl, by = c("director_id" = "id")) %>%
  group_by(director_id, first_name, last_name) %>%
  summarise(movie_count = n()) %>%
  arrange(desc(movie_count)) %>%
  head(10)
```

```
show_query(result)
```

```
## `summarise()` has grouped output by "director_id" and "first_name". You can
## override using the `.groups` argument.
```

```
## <SQL>
## SELECT `director_id`, `first_name`, `last_name`, COUNT(*) AS `movie_count`
## FROM (
##   SELECT `movies_directors`.*, `first_name`, `last_name`
##   FROM `movies_directors`
##   INNER JOIN `directors`
##     ON (`movies_directors`.`director_id` = `directors`.`id`)
## ) `q01`
## GROUP BY `director_id`, `first_name`, `last_name`
## ORDER BY `movie_count` DESC
## LIMIT 10
```

```
collect(result)
```

```
## `summarise()` has grouped output by "director_id" and "first_name". You can
## override using the `.groups` argument.
```

```
## # A tibble: 10 x 4
## # Groups:   director_id, first_name [10]
##    director_id first_name                last_name movie_count
##          <int> <chr>                      <chr>         <int64>
##  1       25116 Dave                       Fleischer         616
##  2       56530 Georges                    Méliès            554
##  3       30570 D.W.                       Griffith          530
##  4        1958 Gilbert M. 'Broncho Billy' Anderson          360
##  5       24576 Louis                      Feuillade         345
##  6       72184 Mack                       Sennett           330
##  7       41763 Seymour                    Kneitel           320
##  8       26173 Friz                       Freleng           316
##  9       24941 Bud                        Fisher            312
## 10       38415 Chuck (I)                  Jones             293
```

Question 3)

```
library(reticulate)
virtualenv_install("stats167_venv", packages = "pymysql")
use_virtualenv("stats167_venv")
```

(a)

```
import pymysql

con = pymysql.connect(
  host = "relational.fel.cvut.cz",
  port = 3306,
  user = "guest",
  password = "ctu-relational",
  database = "imdb_ijs"
)
```

```
with con.cursor() as cur:
    query = """
        SELECT role, COUNT(DISTINCT roles.actor_id) AS num_actresses
        FROM roles
        JOIN actors ON roles.actor_id = actors.id
        WHERE actors.gender = 'F' AND role IS NOT NULL
        GROUP BY role
        ORDER BY num_actresses
        LIMIT 20;
    """
    cur.execute(query)
    results = cur.fetchall()
```

```
## 20
```

```
for row in results:
    print(row)
```

```
## ('Þórkatla', 1)
## ('#1 Woman', 1)
## ('#2 Party Babe Hybrid', 1)
## ('#22 HeidiBowl/Heidi', 1)
## ('#23 Bad Hair Days', 1)
## ('#23 Bad Hair Days/#43 Passing', 1)
## ('#3 Party Babe Hybrid', 1)
## ('#43 Passing on Big Role: Mons', 1)
## ('#43 Passing on Big Role: Pear', 1)
## ('#1 Party Babe Hybrid', 1)
## ('#1 Fan', 1)
## ('"Statue of Liberty"', 1)
## (' (1991 reissue only)', 1)
## (' (episode 4: The Criminal)', 1)
## (' (segment "La voce umana")', 1)
## (' (segment Red Peppers) (segme', 1)
## ('"Astoria" Owner', 1)
## ('"Betsy Ross"', 1)
## ('"Frank" Hickson', 1)
## ('"Fred" Lincoln', 1)
```

(b)

```
con = pymysql.connect(
  host = "relational.fel.cvut.cz",
  port = 3306,
  user = "guest",
  password = "ctu-relational",
  database = "imdb_ijs"
)

with con.cursor() as cur:
    query = """
        SELECT actors.first_name, actors.last_name, sub.role_count,
        RANK() OVER(
          ORDER BY sub.role_count DESC) AS rank
```

```
        FROM (
          SELECT actor_id, COUNT(*) AS role_count
          FROM roles
          WHERE role LIKE '%Gamgee%'
          GROUP BY actor_id
        ) AS sub
        JOIN actors ON sub.actor_id = actors.id;

    """
    cur.execute(query)
    results = cur.fetchall()
```

```
## 7
```

```
for row in results:
    print(row)
```

```
## ('Sean', 'Astin', 6, 1)
## ('Norman', 'Forsey', 1, 2)
## ('Maisie', 'McLeod-Riera', 1, 2)
## ('Michael', 'Scholes', 1, 2)
## ('Serge', 'Lhorca', 1, 2)
## ('Alexandra', 'Astin', 1, 2)
## ('Roddy', 'McDowall', 1, 2)
```

Question 4)

(a)

```
import pymysql

con = pymysql.connect(
  host = "relational.fel.cvut.cz",
  port = 3306,
  user = "guest",
  password = "ctu-relational",
  database = "imdb_ijs"
)

with con.cursor() as cur:
    query = """
        WITH director_stats AS (
          SELECT
            md.director_id,
            AVG(m.rank) AS average_rating,
            COUNT(*) AS movie_count
          FROM movies m
          JOIN movies_directors md ON m.id = md.movie_id
          WHERE m.rank IS NOT NULL
          GROUP BY md.director_id
          HAVING COUNT(*) >= 5
        ), ranked_movies AS(
          SELECT
            d.first_name,
```

```
            d.last_name,
            m.name AS movie_title,
            m.rank AS movie_rating,
            ds.average_rating,
            RANK() OVER (PARTITION BY d.id
            ORDER BY m.rank DESC) AS rank
          FROM director_stats ds
          JOIN directors d ON d.id = ds.director_id
          JOIN movies_directors md ON d.id = md.director_id
          JOIN movies m ON m.id = md.movie_id
          WHERE m.rank IS NOT NULL
        )
        SELECT *
        FROM ranked_movies
        WHERE rank <= 3
        ORDER BY last_name, movie_rating DESC
        LIMIT 20;

    """
    cur.execute(query)
    results = cur.fetchall()
```

```
## 20
```

```
for row in results:
    print(row)
```

```
## ('Veikko', 'Aaltonen', 'Rakkaudella, Maire', 6.9, 6.142857210976737, 1)
## ('Veikko', 'Aaltonen', 'Is meidn', 6.8, 6.142857210976737, 2)
## ('Veikko', 'Aaltonen', 'Tuhlaajapoika', 6.7, 6.142857210976737, 3)
## ('Paul', 'Aaron', 'Maxie', 5.0, 4.3599999904632565, 1)
## ('Paul', 'Aaron', 'Different Story, A', 4.8, 4.3599999904632565, 2)
## ('Paul', 'Aaron', 'Force of One, A', 4.5, 4.3599999904632565, 3)
## ('George', 'Abbott', 'Manslaughter', 7.4, 6.533333381017049, 1)
## ('George', 'Abbott', 'Damn Yankees!', 7.1, 6.533333381017049, 2)
## ('George', 'Abbott', 'Pajama Game, The', 6.9, 6.533333381017049, 3)
## ('Vadim', 'Abdrashitov', 'Sluga', 8.6, 6.342857258660453, 1)
## ('Vadim', 'Abdrashitov', 'Ostanovilsya poyezd', 8.1, 6.342857258660453, 2)
## ('Vadim', 'Abdrashitov', 'Plyumbum, ili opasnaya igra', 7.0, 6.342857258660453, 3)
## ('Lasse', 'Åberg', 'Sllskapsresan', 7.2, 5.799999952316284, 1)
## ('Lasse', 'Åberg', 'Repmnad', 6.2, 5.799999952316284, 2)
## ('Lasse', 'Åberg', 'Sllskapsresan 2 - Snowroller', 6.2, 5.799999952316284, 2)
## ('Salah', 'Abouseif', 'Zawja al-thaniya, al-', 8.3, 7.100000095367432, 1)
## ('Salah', 'Abouseif', "Shabab imra'a", 7.9, 7.100000095367432, 2)
## ('Salah', 'Abouseif', 'Bidaya wa nihaya', 7.8, 7.100000095367432, 3)
## ('Jim', 'Abrahams', 'Airplane!', 7.7, 6.112499833106995, 1)
## ('Derwin', 'Abrahams', 'Secrets of the Wasteland', 7.4, 5.599999904632568, 1)
```

(b)

```
con = pymysql.connect(
  host = "relational.fel.cvut.cz",
  port = 3306,
  user = "guest",
```

```
  password = "ctu-relational",
  database = "imdb_ijs"
)

with con.cursor() as cur:
    query = """
        WITH director_stats AS (
          SELECT
            md.director_id,
            FLOOR(m.year / 10) * 10 AS decade,
            COUNT(*) AS movie_count
          FROM movies m
          JOIN movies_directors md ON m.id = md.movie_id
          WHERE m.year >= 1950
          GROUP BY md.director_id, decade
        ), max_count AS (
          SELECT
            decade,
            MAX(movie_count) AS max_count
          FROM director_stats
          GROUP BY decade
        ), top_directors AS (
          SELECT
            ds.director_id,
            ds.decade,
            ds.movie_count
          FROM director_stats ds
          JOIN max_count mc ON ds.decade = mc.decade AND ds.movie_count = mc.max_count
        )
        SELECT
          d.first_name,
          d.last_name,
          td.decade,
          td.movie_count
        FROM top_directors td
        JOIN directors d ON d.id = td.director_id
        ORDER BY td.decade, d.last_name;

    """
    cur.execute(query)
    results = cur.fetchall()
```

7

```
for row in results:
    print(row)
```

```
## ('Seymour', 'Kneitel', 1950, 130)
## ('Wui', 'Ng', 1950, 130)
## ('Hal (I)', 'Seeger', 1960, 179)
## ('K.N.', 'Sasidharan', 1970, 63)
## ('Narayana Rao', 'Dasari', 1980, 74)
## ('Kevin (III)', 'Dunn', 1990, 68)
## ('Kevin (III)', 'Dunn', 2000, 70)
```

Question 5)

```sql
SELECT movie_id,
  year,
  SUM(views) AS cumulative_views
FROM (
  SELECT
  m1.movie_id,
  m1.year,
  (
    SELECT SUM(m2.views)
    FROM m2.movie_id = m1.movie_id
      AND m2.year BETWEEN m1.year - 2 AND m2.year
  ) AS views
  FROM movie_views m1
  WHERE m1.year < (SELECT MAX(year) FROM movie_views)
) AS subquery
ORDER BY movie_id, year;
```