VE472 LAB3

Bunyod Suvonov 522370990020

2. SQLite Environment Setup

Used the following bash script to set up the workspace and remove the first line of all tsv files:

```
#!/bin/bash

# define variables
USER_DIR="$HOME"
VAR_DIR="$USER_DIR/var"
mkdir "$VAR_DIR"

# extract .tar file
tar -xvf "$USER_DIR/imdb.tar" -C "$VAR_DIR"

# extract tsv.gz files
find "$VAR_DIR" -name "*.tsv.gz" -exec gzip -d {} \;

# remove the first line of all tsv files:
for file in "$VAR_DIR"/*.tsv; do
        sed -i '' '1d' "$file"
done
```

Created and entered sqlite3 file:

sqlite3 var/imdb.sqlite3

First removed " from all tsv files using the following shell script to avoid data insertion misbehaviour. The file name is changed and script is ran for every tsv file:

```
#!/bin/bash

# Define the input file name
input_file="./title.ratings.tsv"

# Define a temporary file for storing the modified content
temp_file="${input_file}.temp"

# Replace all double quotes with an empty string using sed
sed 's/"//g' "$input_file" > "$temp_file"

# Move the modified content back to the original file
mv "$temp_file" "$input_file"

# Optionally, you can remove the temporary file if no longer needed
# rm "$temp_file"

echo "Double quotes removed from $input_file"
```

Then created tables and imported the files by entering the following into the opened sqlite3 file:

```
create table name
(
    nconst varchar(10) not null,
    primaryName text not null,
    birthYear varchar(4) not null,
    deathYear varchar(4),
    primaryProfession text not null,
    knownForTitles text not null,
    primary key(nconst)
);

.separator "\t"
.import name.basics.tsv name

create table title_akas
(
    tconst varchar(10) not null,
    ordering integer not null,
    title text,
    region text,
    language text,
    types text,
    attributes text,
    isOriginalTitle integer,
    primary key(tconst, ordering)
);

.separator "\t"
.import title.akas.tsv title_akas

create table title_basics
(
    tconst varchar(10) not null,
    titleType varchar(64) not null,
    primaryTitle text not null,
    originalTitle text not null,
    isAdult boolean not null,
    startYear varchar(4) not null,
    endYear varchar(4),
    runtimeMinutes integer not null,
    genres text not null,
    primary key(tconst)
);

.separator "\t"
.import title.basics.tsv title_basics

create table title_crew
(
    tconst varchar(10) not null,
    directors text not null,
    writers text not null,
    primary key(tconst)
);

.separator "\t"
.import title.crew.tsv title_crew

create table title_episode
(
    tconst varchar(10) not null,
    parentTconst varchar(10),
    seasonNumber integer,
    episodeNumber integer
);

.separator "\t"
.import title.episode.tsv title_episode

create table title_principals
(
    tconst varchar(10) not null,
    ordering integer not null,
    ordering integer not null,
    nconst varchar(10) not null,
    category text not null,
    job text,
    characters text,
    primary key(tconst, ordering, nconst)
);

.separator "\t"
.import title.principals.tsv title_principals

create table title_ratings
(
  tconst varchar(10) not null,
  averageRating double not null,
  numVotes integer not null,
    primary key(tconst)
);

.separator "\t"
.import title.ratings.tsv title_ratings
```

3. Verifying the Data

In the imdb.sqlite3 file, I used the following commands to find the required info to verify the data:

Oldest movie

```
 SELECT title_basics.primaryTitle, title_basics.originalTitle, title_basics.startYear
FROM title_basics WHERE title_basics.titleType = 'movie'
ORDER BY title_basics.startYear ASC LIMIT 1;

Birmingham|Birmingham|1896
```

The longest movie in 2009

Not so sure what is meant by "in 2009", but logically movies before 2009 are also available, so queried the movies before 2010

```
 SELECT originalTitle, runtimeMinutes
FROM title_basics
WHERE titleType = "movie" AND runtimeMinutes <>'\N' AND startYear < '2010'
ORDER BY runtimeMinutes DESC
LIMIT 1;

Beijing 2003|9000
```

The year with the most movies

```
 SELECT startYear, COUNT(*) as count FROM title_basics
WHERE startYear <> "\N" AND titleType = "movie"
GROUP BY startYear
ORDER BY count desc
LIMIT 1;

2017|17920
```

The name of the person who contains in the most movies

```
SELECT name.primaryName, COUNT(*) as cnt FROM title_principals
INNER JOIN name on title_principals.nconst = name.nconst
GROUP BY title_principals.nconst
ORDER BY cnt DESC
LIMIT 1;

Shobha Kapoor|14575
```

The principal crew of the movie with highest average ratings and more than 500 votes

```
SELECT * FROM title_crew WHERE tconst IN
(
    SELECT tconst FROM title_ratings WHERE numVotes > 500
    ORDER BY averageRating DESC
    LIMIT 1
);

tt11128054|nm0804080|nm0804080
```

(Advanced) The count of each Pair<BirthYear, DeathYear> of the people

```
SELECT birthYear, deathYear, COUNT(*) as cnt FROM name
WHERE birthYear <> "\N" AND deathYear <> "\N"
GROUP BY birthYear, deathYear
ORDER BY cnt DESC

LIMIT 10;

1931|2020|173
1935|2020|167
1933|2020|164
1929|2015|163
1930|2018|163
1927|2015|162
1930|2020|159
1931|2019|159
1927|2017|157
1928|2012|157
```

Ex 5.

5.1

The top 3 most common professions among these people and also the average life span of these three professions.

Results:

```
Profession: actor, Average Lifespan: 69.55 years
Profession: actress, Average Lifespan: 72.79 years
Profession: producer, Average Lifespan: 71.66 years
```

See com.l2.ex5.Top3Profs for implementation

5.2

The top 3 most popular (received most votes) genres.

Results:

```
Genre: Drama, Total Votes: 567636408
Genre: Action, Total Votes: 340946494
Genre: Comedy, Total Votes: 335927983
```

See com.l2.ex5.TopGenres for implementation

5.3

The average time span (endYear - startYear) of the titles for each person. Most of the titles are movies with the same startYear and endYear, or with Null ("\N") field as endYear, they are all considered 0 time span. List is generated for every person, 10 from the top of the list are shown below:

Results:

```
 Name: Fred Astaire, Average Time Span: 0.0
Name: Lauren Bacall, Average Time Span: 0.0
Name: Brigitte Bardot, Average Time Span: 0.0
Name: John Belushi, Average Time Span: 0.0
Name: Ingmar Bergman, Average Time Span: 0.0
Name: Ingrid Bergman, Average Time Span: 0.0
Name: Humphrey Bogart, Average Time Span: 0.0
Name: Marlon Brando, Average Time Span: 0.0
Name: Richard Burton, Average Time Span: 0.0
Name: James Cagney, Average Time Span: 0.0
Name: Gary Cooper, Average Time Span: 0.0
Name: Bette Davis, Average Time Span: 0.0
Name: Doris Day, Average Time Span: 0.0
Name: Olivia de Havilland, Average Time Span: 0.0
Name: James Dean, Average Time Span: 0.4
Name: Georges Delerue, Average Time Span: 0.0
Name: Marlene Dietrich, Average Time Span: 0.0
Name: Kirk Douglas, Average Time Span: 0.0
Name: Federico Fellini, Average Time Span: 0.0
Name: Henry Fonda, Average Time Span: 0.0
```

See com.l2.ex5.AverageTimeSpan for implementation.