VE472 LAB3

Wu Jiayao 517370910257

3. Verifying the data

Schema

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

create table title
(
    tconst varchar(10) not null primary key,
    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
);

create table principal
(
    tconst varchar(10) not null,
    ordering integer not null,
    nconst varchar(10) not null,
    category text not null,
    job text,
    characters text
);
create table rating
(
    tconst varchar(10) not null primary key,
    averageRating double not null,
    numVotes integer not null
);

.separator "\t"
.import name.basics.tsv name
.import title.basics.tsv title
.import title.principals.tsv principal
.import title.ratings.tsv rating
```

the oldest movie

```
 select tconst, primaryTitle, startYear from title
 where startYear <> "\N" and titleType = "movie"
 order by startYear
 limit 1;
```

```
tconst      primaryTitle  startYear
----------  ------------  ----------
tt2210499   Birmingham    1896
```

the longest in 2019

```
select tconst, primaryTitle, startYear,runtimeMinutes from title
where startYear = "2009" and runtimeMinutes <> "\N" and titleType = "movie"
order by runtimeMinutes desc
limit 1;
```

```
tconst      primaryTitle       startYear   runtimeMinutes
----------  -----------------  ----------  --------------
tt1458948   Native of Owhyhee  2009        390
```

The year with the most movies

```
select startYear, count(*) as count from title
where startYear <> "\N" and titleType = "movie"
group by startYear
order by count desc
limit 1;
```

```
startYear   count
----------  ----------
2019        14205
```

The name of the person who contains in the most movies

```
 select principal.nconst, name.primaryName, count(*) as contained from
principal
 inner join title on principal.tconst = title.tconst
 inner join name on principal.nconst = name.nconst
 where title.titleType = "movie"
 group by principal.nconst
 order by contained desc
 limit 1;
```

```
nconst      primaryName  contained
----------  -----------  ----------
nm0006137   Ilaiyaraaja  952
```

the principal crew

```
select tconst,nconst, name.primaryName, category from principal
left outer join name using (nconst)
where principal.tconst in
(
    select rating.tconst from rating
    where rating.numVotes > 500
    order by rating.averageRating
    desc
    limit 1
);
```

```
tconst      nconst      primaryName     category
----------  ----------  --------------  -------------------
tt11128054  nm1249167   Kiril Spaseski  production_designer
tt11128054  nm8262223                   actor
tt11128054  nm9819225                   actor
tt11128054  nm1426138                   actor
tt11128054  nm0953471   Ivan Zaric      actor
tt11128054  nm0804080   Slobodan Skerl  director
tt11128054  nm5173970   Ivana Mikovic   producer
tt11128054  nm0559105   Mate Matisic    composer
tt11128054  nm3409959   Maja Radosevic  cinematographer
tt11128054  nm5275099                   production_designer
```

the count of each pair

```
select birthYear, deathYear, count(*) as pairCount from name
where birthYear <> "\N" and deathYear <> "\N"
group by birthYear, deathYear
order by pairCount desc;
```

5. Advanced

```
create table name_profession
(
    nconst varchar(10) not null,
    profession text not null,
    foreign key(nconst) references name(nconst)
);

create table IF NOT EXISTS title_genre
(
    tconst varchar(10) not null,
    genre text not null,
    foreign key(tconst) references title(tconst)
);
```

```
import sqlite3
conn = sqlite3.connect('var/imdb.sqlite3')

c = conn.cursor()
```

```
names = c.execute("select * from name")
insert_list = []
for name in names:
    if name[4] != '\\N' and len(name[4])>0:
        nconst = name[0]
        professions = name[4].split(',')
        for _profession in professions:
            insert_list.append((nconst,_profession))
c.executemany("insert into name_profession values (?,?)",insert_list)

titles = c.execute("select * from title")
insert_list = []
for title in titles:
    if title[8] != '\\N' and len(title[8])>0:
        tconst = title[0]
        genres = title[8].split(',')
        for _genre in genres:
            insert_list.append((tconst,_genre))
c.executemany("insert into title_genre values (?,?)",insert_list)
conn.commit()
conn.close()
```

top 3 common profession

```
select profession, count(*) as count, avg(deathYear - birthYear) as
avgLifeSpan from name_profession
inner join name using (nconst)
where deathYear <> "\N" and birthYear <> "\N"
group by profession
order by count desc
limit 3;
```

```
profession  count       avgLifeSpan
----------  ----------  ----------------
actor       60186       70.0475525869804
writer      30931       71.8120655652905
actress     26260       73.4029702970297
```

The top 3 most popular (received most votes) genres

```
select genre, sum(numVotes) as votes from title_genre
inner join rating using (tconst)
group by genre
order by votes desc
limit 3;
```

```
genre       votes
----------  ----------
Drama       466974607
Action      302900340
Comedy      286806624
```

The average time span (endYear - startYear) of the titles for each person

```
select principal.nconst, name.primaryName, avg(title.endYear -
title.startYear) as avgTimeSpan from principal
inner join title using (tconst)
inner join name using (nconst)
where title.startYear <> "\N" and title.endYear <> "\N"
group by principal.nconst
order by avgTimeSpan desc;
```