COMS W4111: Introduction to Databases
Spring 2023, Sections 002, V02

Non-Programming Track, HW2, Part 2

Introduction

Environment

- Test environment.
- Set your MySQL user and password below.

In [1]:
```
mysql_user = "root"
mysql_pw = "dbuserdbuser"
```

In [2]:
```
%load_ext sql
```

In [3]:
```
full_url = f"mysql+pymysql://{mysql_user}:{mysql_pw}@localhost"
full_url
```

Out[3]:
```
'mysql+pymysql://root:dbuserdbuser@localhost'
```

In [4]:
```
%sql $full_url
```

In [5]:
```
%sql select * from db_book.student;
```

```
 * mysql+pymysql://root:***@localhost
13 rows affected.
```

Out[5]:

ID	name	dept_name	tot_cred
00128	Zhang	Comp. Sci.	102
12345	Shankar	Comp. Sci.	32
19991	Brandt	History	80
23121	Chavez	Finance	110
44553	Peltier	Physics	56
45678	Levy	Physics	46
54321	Williams	Comp. Sci.	54
55739	Sanchez	Music	38
70557	Snow	Physics	0
76543	Brown	Comp. Sci.	58
76653	Aoi	Elec. Eng.	60
98765	Bourikas	Elec. Eng.	98
98988	Tanaka	Biology	120

Submission Instructions

- See Ed for instructions.

Data and Scheme Cleanup

characters and name_basics_all

- The task is to "clean up" `characters` and produce a table `charactersFixed`.
- The task will require adding missing rows to `name_basics_all`. There are two row's in `characters` that have an `actorLink` and `actorName` got which there is no matching row in `name_basics_all`.
- `characters` has two actors with actorNames `Barry John O'Connor` and `Barry O'Connor` who are the same actor.
- My `charactersFixed` has the following columns:
  - `characterId` is a generated primary key. See below for an explanation.
  - `characterName`: The value from `characters`.
  - `characterImdbID`: The `characterLink` from `characters` with `/character/` removed.
  - `characterLink`: The `characterLink` from `characters`.
  - `actorNConst`: `actorLink` from `characters`.
  - `actorLink`: A value of the form `/names/` followed by the `actorNConst`.
  - `characterImageFull`: The value from `characters`.
  - `characterImageThumb`: The value from `characters`.
  - `kingsguard`: The value from `characters`.
  - `royal`: The value from `characters`.
- The algorithm for generating the `characterID` on insert is the following:
  - The prefix for the `character` is either:
    - The substring of `characterName` preceeding the first `' '`.
    - The `characterName` is there is no `' '`.
  - If there are `N` rows in the table, the number after the prefix is `N+1`.
  - Implementing this is tricky. Your first attempt might rely on `auto-increment`, but this does not work. You may also be tempted to count rows, but that does not work. A hint is that you will need to use a trigger and some other table/data that you create.
- The directory with this notebook containers data from my version of `charactersFixed`.
- The cells below load the data to allow you to examine. In your SQL table, `NaN` will be `NULL`.

In [6]:
```
import pandas as pd
```

In [10]:
```
characters_df = pd.read_csv('./charactersFixed.csv')
```

In [11]:
```
characters_df
```

Out[11]:

	characterId	characterName	characterImdbID	characterLink	actorNconst	
0	Addam1	Addam Marbrand	ch0305333	/character/ch0305333	nm0389698	/names/n
1	Aegon2	Aegon Targaryen	NaN	NaN	NaN	
2	Aeron3	Aeron Greyjoy	ch0540081	/character/ch0540081	nm0269923	/names/n
3	Aerys4	Aerys II Targaryen	ch0541362	/character/ch0541362	nm0727778	/names/n
4	Akho5	Akho	ch0544520	/character/ch0544520	nm6729880	/names/n
...	...	...	...	...	...	
384	Young385	Young Nan	ch0305018	/character/ch0305018	nm1519719	/names/n
385	Young386	Young Ned	ch0154681	/character/ch0154681	nm7075019	/names/n
386	Young387	Young Ned Stark	ch0154681	/character/ch0154681	nm7509185	/names/n
387	Young388	Young Rodrik Cassel	ch0171391	/character/ch0171391	nm7509186	/names/n
388	Zanrush389	Zanrush	ch0540870	/character/ch0540870	nm0503319	/names/n

389 rows × 10 columns

- Your answer below should show all of your SQL statements, including DDL, for creating and loading `charactersFixed` as well as changes to `name_basics_all`.
- You can use the data in the CSV file to test your work. Show at least one test.

In [13]:
```
%sql drop database if exists W4111_HW2_p2_cs4185;
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
```

Out[13]: []

In [ ]:
```
```

In [23]:
```
%%sql
use s23_w4111_hw2_cs4185;

insert into name_basics_all (nconst, primaryName)
with left_join_table as (
    select * from characters left join name_basics_all
    on characters.actorLink=name_basics_all.nconst
),
missing_rows as (
 select * from left_join_table
    where nconst is null and primaryName is null
    and actorLink is not null and actorName is not null
)

select actorLink, actorName from missing_rows;


drop table if exists charactersFixed;
create table charactersFixed
(
    characterId         varchar(25)  not null,
    characterName       varchar(50)  null,
    characterImdbID     varchar(25)  null,
    characterLink       varchar(50)  null,
    actorNConst         varchar(50)  null,
    actorLink           varchar(50)  null,
    characterImageFull  varchar(250) null,
    characterImageThumb varchar(250) null,
    kingsguard          float        null,
    royal               float        null,
    constraint charactersFixed_pk
        primary key (characterId)
);

drop table if exists charactersTemp;
create table charactersTemp (
    id int auto_increment primary key)
select * from characters order by characterName;
```

```
insert into charactersFixed
(characterId, characterName, characterImdbID,
 characterLink, actorNConst,
 actorLink, characterImageFull,
 characterImageThumb, kingsguard, royal)
select case
    when locate(' ', characterName) = 0
    then concat(characterName, id)
    else concat(substring(
        characterName, 1,  locate(
            ' ', characterName) - 1), id) end
as characterId, characterName,  SUBSTRING(characterLink, 12,
                                        length(
                                            characterLink) - 12) ,
    SUBSTRING(characterLink, 1, length(
        characterLink) - 1) , actorLink as actorNConst,
    concat('/names/', actorLink),
    characterImageFull, characterImageThumb,
    kingsguard, royal from charactersTemp;
/* SQL statements in this cell. You may use multiple cells. */
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
0 rows affected.
0 rows affected.
0 rows affected.
0 rows affected.
389 rows affected.
389 rows affected.
0 rows affected.
```

Out[23]: []

In [26]:
```
%%sql

use s23_w4111_hw2_cs4185;
select * from name_basics_all

/* SQL test to show result. */
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
350 rows affected.
```

Out[26]:

nconst	primaryName	birthYear	deathYear	primaryProfession
nm0389698	B.J. Hogg	1955	2020	actor,music_department
nm0269923	Michael Feast	1946	None	actor,composer
nm0727778	David Rintoul	1948	None	actor
nm6729880	Chuku Modu	1990	None	actor,writer,producer
nm0853583	Owen Teale	1961	None	actor
nm0203801	Karl Davies	1982	None	actor,producer
nm8257864	Megan Parkinson	None	None	actress
nm0571654	Fintan McKeown	None	None	actor

In [27]:
```
%%sql

use s23_w4111_hw2_cs4185;
select * from charactersFixed

/* SQL test to show result. */
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
389 rows affected.
```

Out[27]:

characterId	characterName	characterImdbID	characterLink
Addam1	Addam Marbrand	ch0305333	/character/ch0305333
Aegon2	Aegon Targaryen	None	None
Aeron3	Aeron Greyjoy	ch0540081	/character/ch0540081
Aerys4	Aerys II Targaryen	ch0541362	/character/ch0541362
Akho5	Akho	ch0544520	/character/ch0544520
Alliser6	Alliser Thorne	ch0246938	/character/ch0246938
Alton7	Alton Lannister	ch0305012	/character/ch0305012

name_basics_all

- The column `primaryProfessions` is multi-valued and non-atomic. This violates good relational design principle.
- Create a new table `name_basics_all_fixed` which does not have the column `primaryProfessions`.
- You will need to use SQL to create and load other tables with information from `name_basics_all` to enable you to create a view `name_basics_all_fixed_view` that recreates the data in `name_basics_all`. The tables you create should have atomic columns, primary keys and foreign keys, etc.

In [28]:
```
%%sql
use s23_w4111_hw2_cs4185;

with one as (
    select primaryProfession,
        replace(primaryProfession, ',', '') as no_comma
    from
        name_basics_all),
    two as (select primaryProfession, length(
        primaryProfession) - length(no_comma) as comma_count
            from one)
select comma_count, count(*) as profession_comma_space_count
from two group by comma_count order
by profession_comma_space_count asc;

/* Use this cell and others to create tables, load data, etc. */
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
4 rows affected.
0 rows affected.
```

Out[28]: []

In [41]:
```
%%sql
use s23_w4111_hw2_cs4185;

drop table if exists Professions;
create table Professions
(
  nconst varchar(200) not null,
  primaryProfession varchar(200),
  Profession1 varchar(200),
  Profession2 varchar(200),
  Profession3 varchar(200),
  primary key (nconst)
);

/* Write a query that uses your view to reproduce name_basics_all */
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
0 rows affected.
0 rows affected.
0 rows affected.
0 rows affected.
```

Out[41]: []

In [42]:
```
%%sql
use s23_w4111_hw2_cs4185;

insert into Professions(
nconst, primaryProfession, Profession1, Profession2, Profession3)
select
    nconst, primaryProfession,
    substring_index(substring_index(
        concat(primaryProfession, ','), ',', 1), ',', -1),
    substring_index(substring_index(
        concat(primaryProfession, ','), ',', 2), ',', -1),
    substring_index(substring_index(
        concat(primaryProfession, ','), ',', 3), ',', -1)
from name_basics_all
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
350 rows affected.
```

Out[42]: []

In [43]:
```
%%sql
use s23_w4111_hw2_cs4185;

drop table if exists name_basics_all_fixed;
create table name_basics_all_fixed
select * from name_basics_all
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
0 rows affected.
350 rows affected.
```

Out[43]: []

In [45]:
```
%%sql
use s23_w4111_hw2_cs4185;

alter table name_basics_all_fixed
drop column primaryProfession;
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
0 rows affected.
```

Out[45]: []

In [47]:
```
%%sql
use s23_w4111_hw2_cs4185;

#alter table name_basics_all
#add primary key(nconst);

#alter table name_basics_all_fixed
#add constraint fk
#foreign key (nconst)
#references name_basics_all (nconst);

alter table Professions
add constraint fk_profession
foreign key (nconst)
references name_basics_all (nconst);
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
0 rows affected.
0 rows affected.
350 rows affected.
```

Out[47]: []

In [50]:
```
%%sql
use s23_w4111_hw2_cs4185;

drop view if exists name_basics_all_fixed_view;
create view name_basics_all_fixed_view as
select
    name_basics_all_fixed.nconst,
    primaryName,
    birthYear,
    deathYear,
    knownForTitles,
    profession1,
    profession2,
    profession3
from
    name_basics_all_fixed
join
    Professions
on
    name_basics_all_fixed.nconst = Professions.nconst;
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
0 rows affected.
0 rows affected.
```

Out[50]:
```
[]
```

In [51]:
```
%%sql
use s23_w4111_hw2_cs4185;

select * from name_basics_all_fixed_view;
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
350 rows affected.
```

Out[51]:

nconst	primaryName	birthYear	deathYear	knownForTitles
nm0000293	Sean Bean	1959	None	tt0120737,tt0167261,tt0944947,tt1181791
nm0000596	Jonathan Pryce	1947	None	tt0104348,tt8404614,tt0120347,tt3750872
nm0000980	Jim Broadbent	1949	None	tt0203009,tt1431181,tt1007029,tt0217505
nm0001097	Charles Dance	1946	None	tt0944947,tt0107362,tt2084970,tt0280707
nm0001290	Richard E. Grant	1957	None	tt4595882,tt0280707,tt0102070,tt0094336
nm0001354	Ciarán Hinds	1953	None	tt1340800,tt1596365,tt1201607,tt12789558
nm0001671	Diana Rigg	1938	2020	tt0054518,tt9639470,tt0064757,tt0944947

In [ ]: