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 = "dbuserbdbuser"
```

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

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

Out[3]:
```
'mysql+pymysql://root:dbuserbdbuser@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 [7]:
```
characters_df = pd.read_csv('./charactersFixed.csv')
```

In [8]:
```
characters_df
```

Out[8]:

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

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.

changes to name_basics_all

In [9]:
```
%%sql
use s23_w4111_hw2_zc2670;

drop table if exists name;

create table name as
select actorLink, actorName,primaryName,nconst
from characters
left join name_basics_all
on characters.actorName = name_basics_all.primaryName
where characters.actorName is not NULL
and name_basics_all.primaryName is NULL;

drop table if exists nconst;

create table nconst as
select actorLink, actorName,primaryName,nconst
from characters left join name_basics_all
on characters.actorLink = name_basics_all.nconst
where characters.actorLink is not NULL
and name_basics_all.nconst is NULL;
```

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

Out[9]: []

In [10]:
```
%%sql
select * from name;
```

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

Out[10]:

actorLink	actorName	primaryName	nconst
nm2231505	David Coakley	None	None
nm0057965	Cliff Barry	None	None
nm3226454	Barry John O'Connor	None	None
nm2746504	Dean S. Jagger	None	None
None	Barry O'Connor	None	None
nm8199963	Michael Patrick	None	None

In [11]:
```
# %%sql
# select * from name_basics_all where nconst='nm2231505';
```

In [12]:
```
# %%sql
# select * from characters where actorLink='nm2231505';
```

In [13]:
```
# characters_df.loc[characters_df.actorNconst=='nm2231505']
```

In [14]:
```
%%sql
select * from nconst;
```

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

Out[14]:

actorLink	actorName	primaryName	nconst
nm3226454	Barry John O'Connor	None	None
nm8199963	Michael Patrick	None	None

In [15]:
```
%%sql
select * from name
intersect
select * from nconst;
```

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

Out[15]:

actorLink	actorName	primaryName	nconst
nm3226454	Barry John O'Connor	None	None
nm8199963	Michael Patrick	None	None

In [16]:
```
%%sql
use s23_w4111_hw2_zc2670;

insert into name_basics_all(primaryName,nconst)
values("Barry John O'Connor","nm3226454");

insert into name_basics_all(primaryName,nconst)
values("Michael Patrick","nm8199963");
```

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

Out[16]: []

In [17]:
```
%%sql
use s23_w4111_hw2_zc2670;

update characters
set actorLink = 'nm3226454'
where actorName = "Barry O'Connor";

## since the two names mean the same person
```

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

Out[17]: []

creating and loading charactersFixed

In [18]:
```
%%sql
use s23_w4111_hw2_zc2670;
drop table if exists charactersFixed;
```

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

Out[18]: []

In [19]:
```
%%sql

use s23_w4111_hw2_zc2670;

create table charactersFixed as
with one as (
    select
    characterName, actorName,
    substr(characterLink, 12, length(characterLink) - 12)
    as characterImdbID,
    characterLink,
    actorLink as actorNConst,
    concat('/names/', actorLink) as actorLink,
    characterImageFull,characterImageThumb, kingsguard, royal,
    substr(characterName, 1, locate(' ', concat(characterName,' '))
    as firstName,
    row_number() over() as num
    from characters
),
    two as (
    select
    concat(firstName, num) as characterId,
    characterName,
    characterImdbID,
    characterLink,
    actorNConst,
    actorLink,
    characterImageFull,
    characterImageThumb,
    kingsguard,
    royal
        from one
)

select * from two;
```

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

Out[19]: []

In [20]:
```
%%sql
select * from charactersFixed;
```

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

:

characterId	characterName	characterImdbID	characterLink	actorNConst
Addam1	Addam Marbrand	ch0305333	/character/ch0305333/	nm038969
Aegon2	Aegon Targaryen	None	None	None
Aeron3	Aeron Greyjoy	ch0540081	/character/ch0540081/	nm026992
Aerys4	Aerys II Targaryen	ch0541362	/character/ch0541362/	nm072777
Akho5	Akho	ch0544520	/character/ch0544520/	nm672988
Alliser6	Alliser Thorne	ch0246938	/character/ch0246938/	nm085358
Alton7	Alton Lannister	ch0305012	/character/ch0305012/	nm020380

In [21]:
```
%%sql
use s23_w4111_hw2_zc2670;
ALTER TABLE charactersFixed MODIFY characterName varchar(32);
ALTER TABLE charactersFixed MODIFY kingsguard varchar(4);
ALTER TABLE charactersFixed MODIFY royal varchar(4);
```

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

Out[21]: []

In [22]:
```
%%sql

use s23_w4111_hw2_zc2670;

drop function if exists compute_next_characterId;

create function compute_next_characterId(characterName varchar(32))
returns varchar(32)
    reads sql data
BEGIN

    declare count int;
    declare result varchar(32);

    set count = (select count(*) from s23_w4111_hw2_zc2670.characte

    set count = count + 1;

    set result = concat(
        substr(characterName, 1, locate(' ', concat(characterName,'
        count
        );


    return result;

end;
```

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

Out[22]: []

In [23]:
```
%%sql

drop trigger if exists compute_characterId;

create trigger compute_characterId
    before insert
    on s23_w4111_hw2_zc2670.charactersFixed
    for each row
begin

    /* I am going to quietly ignore the characterId if provided. */
    set new.characterId = compute_next_characterId(
        new.characterName
        );
end;
```

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

Out[23]: []

clean up

In [24]:
```
%%sql
use s23_w4111_hw2_zc2670;
drop table if exists characters;
```

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

Out[24]: []

test

In [25]:
```
%%sql

/* SQL test to show result. */

select * from name_basics_all where nconst='nm3226454';
```

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

Out[25]:

nconst	primaryName	birthYear	deathYear	primaryProfession	knownForTitles
nm3226454	Barry John O'Connor	None	None	None	None

In [26]:
```
%%sql
select * from charactersFixed limit 10;
```

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

Out[26]:

characterId	characterName	characterImdbID	characterLink	actorNConst	
Addam1	Addam Marbrand	ch0305333	/character/ch0305333/	nm0389698	/names/nr
Aegon2	Aegon Targaryen	None	None	None	
Aeron3	Aeron Greyjoy	ch0540081	/character/ch0540081/	nm0269923	/names/nr
Aerys4	Aerys II Targaryen	ch0541362	/character/ch0541362/	nm0727778	/names/nr
Akho5	Akho	ch0544520	/character/ch0544520/	nm6729880	/names/nr
Alliser6	Alliser Thorne	ch0246938	/character/ch0246938/	nm0853583	/names/nr
Alton7	Alton Lannister	ch0305012	/character/ch0305012/	nm0203801	/names/nr
Alys8	Alys Karstark	ch0576836	/character/ch0576836/	nm8257864	/names/nr
Amory9	Amory Lorch	ch0305002	/character/ch0305002/	nm0571654	/names/nr
Anguy10	Anguy	ch0316930	/character/ch0316930/	nm1528121	/names/nr

test the trigger

In [27]:
```
%%sql

insert into s23_w4111_hw2_zc2670.charactersFixed (
    characterId,
    characterName,
    characterImdbID,
    characterLink,
    actorNConst,
    actorLink,
    characterImageFull,
    characterImageThumb,
    kingsguard,
    royal)
values (
    compute_next_characterId('Robert'),'Robert','null','null','null
    'null','null','null','null'
    );

select * from s23_w4111_hw2_zc2670.charactersFixed;
```

Young382	Young Benjen Stark	ch0153996	/character/ch01539
Young383	Young Cersei Lannister	ch0159526	/character/ch01595
Young384	Young Lyanna Stark	ch0543804	/character/ch05438
Young385	Young Nan	ch0305018	/character/ch03050
Young386	Young Ned	ch0154681	/character/ch01546
Young387	Young Ned Stark	ch0154681	/character/ch01546
Young388	Young Rodrik Cassel	ch0171391	/character/ch01713
Zanrush389	Zanrush	ch0540870	/character/ch05408
Robert390	Robert	null	r

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_zc2670;

select * from name_basics_all limit 10;
```

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

Out[28]:

nconst	primaryName	birthYear	deathYear	primaryProfession	
nm0389698	B.J. Hogg	1955	2020	actor,music_department	tt0970411,tt0944947,
nm0269923	Michael Feast	1946	None	actor,composer	tt0472160,tt0162661,
nm0727778	David Rintoul	1948	None	actor	tt1655420,tt1139328,
nm6729880	Chuku Modu	1990	None	actor,writer,producer	tt4154664,tt2674426,
nm0853583	Owen Teale	1961	None	actor	tt0102797,tt0485301,
nm0203801	Karl Davies	1982	None	actor,producer	tt12879632,tt7366338,
nm8257864	Megan Parkinson	None	None	actress	tt0944947,tt6636246,
nm0571654	Fintan McKeown	None	None	actor	tt0944947,tt0111904,
nm1528121	Philip McGinley	1981	None	actor	tt3922704,tt0053494,
nm0000980	Jim Broadbent	1949	None	actor,writer,soundtrack	tt0203009,tt1431181,

In [29]:
```
%%sql

use s23_w4111_hw2_zc2670;

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

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

Out[29]:

space_count	names_with_space_count
None	2
1	68
2	97
0	183

In [30]:
```
%%sql
use s23_w4111_hw2_zc2670;
drop table if exists name_basics_all_fixed;
```

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

Out[30]: []

In [31]:
```
%%sql

use s23_w4111_hw2_zc2670;

create table name_basics_all_fixed as
with one as (
select
nconst,
primaryName,
birthYear,
deathYear,
knownForTitles,
substr(primaryProfession, 1,
locate(',', concat(primaryProfession,', , ,'), 1)-1)
    as first_profession,
substr(primaryProfession,
locate(',', concat(primaryProfession,', , ,'), 1)+1,
locate(',', concat(primaryProfession,', , ,'),
locate(',', concat(primaryProfession,', , ,'), 1)+1)-1
- locate(',', concat(primaryProfession,', , ,'), 1))
    as second_profession,
substr(primaryProfession, locate(',', concat(primaryProfession,', ,
locate(',', concat(primaryProfession,', , ,'), 1)+1)+1,
locate(',', concat(primaryProfession,', , ,'),
locate(',', concat(primaryProfession,', , ,'),
locate(',', concat(primaryProfession,', , ,'), 1)+1)+1)-1)
    as third_profession
from name_basics_all
)
select * from one;
```

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

Out[31]: []

In [32]:
```
%%sql

use s23_w4111_hw2_zc2670;
select * from name_basics_all_fixed limit 10;
```

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

Out[32]:

nconst	primaryName	birthYear	deathYear	knownForTitles	firs
nm0389698	B.J. Hogg	1955	2020	tt0970411,tt0944947,tt0986233,tt1240982	
nm0269923	Michael Feast	1946	None	tt0472160,tt0162661,tt0120879,tt0362192	
nm0727778	David Rintoul	1948	None	tt1655420,tt1139328,tt4786824,tt6079772	
nm6729880	Chuku Modu	1990	None	tt4154664,tt2674426,tt6470478,tt0944947	
nm0853583	Owen Teale	1961	None	tt0102797,tt0485301,tt0462396,tt0944947	
nm0203801	Karl Davies	1982	None	tt12879632,tt7366338,tt3428912,tt0944947	
nm8257864	Megan Parkinson	None	None	tt0944947,tt6636246,tt5761478,tt4276618	
nm0571654	Fintan McKeown	None	None	tt0944947,tt0111904,tt0166396,tt0112178	
nm1528121	Philip McGinley	1981	None	tt3922704,tt0053494,tt0944947,tt1446714	
nm0000980	Jim Broadbent	1949	None	tt0203009,tt1431181,tt1007029,tt0217505	

In [33]:
```
%%sql

use s23_w4111_hw2_zc2670;
create or replace view name_basics_all_fixed_view as
    select
        nconst, primaryName, birthYear, deathYear, knownForTitles,
        concat(first_profession,' ',second_profession,' ',third_pro
        as professions
    from
        name_basics_all_fixed
```

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

Out[33]: []

In [34]:
```
%%sql

/* Write a query that uses your view to reproduce name_basics_all *
select * from name_basics_all_fixed_view limit 10;
```

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

Out[34]:

nconst	primaryName	birthYear	deathYear	knownForTitles	
nm0389698	B.J. Hogg	1955	2020	tt0970411,tt0944947,tt0986233,tt1240982	mu
nm0269923	Michael Feast	1946	None	tt0472160,tt0162661,tt0120879,tt0362192	
nm0727778	David Rintoul	1948	None	tt1655420,tt1139328,tt4786824,tt6079772	
nm6729880	Chuku Modu	1990	None	tt4154664,tt2674426,tt6470478,tt0944947	
nm0853583	Owen Teale	1961	None	tt0102797,tt0485301,tt0462396,tt0944947	
nm0203801	Karl Davies	1982	None	tt12879632,tt7366338,tt3428912,tt0944947	
nm8257864	Megan Parkinson	None	None	tt0944947,tt6636246,tt5761478,tt4276618	
nm0571654	Fintan McKeown	None	None	tt0944947,tt0111904,tt0166396,tt0112178	
nm1528121	Philip McGinley	1981	None	tt3922704,tt0053494,tt0944947,tt1446714	
nm0000980	Jim Broadbent	1949	None	tt0203009,tt1431181,tt1007029,tt0217505	

In [ ]:

In [ ]: