All Queries

Abhishek Shah, as5553

— For loading all tables in this assignment, previous data of assignment 1 was used.

— Question 1

```
-- Title table
CREATE TABLE Title(
	id INTEGER PRIMARY KEY,
	type VARCHAR(15),
	title TEXT,
	originalTitle TEXT,
	startYear INTEGER,
	endYear INTEGER,
	runtime INTEGER,
	avgRating FLOAT,
	numVotes INTEGER
);

-- Inserting into Title
insert into Title
select mo.titleId, mo.titleType, mo.primaryTitle,
mo.originalTitle, mo.startYear, mo.endYear, mo.runtimeMinutes,
mr.averageRating, mr.numVotes
from Title2 as mo
```

```
join Title_Rating as mr
on mo.titleId = mr.titleId
where mo.isAdult = 'false';


-- Temp Genre Table
create table GenreTemp(
        id serial primary key,
        genre varchar(255)
);

-- Inserting into Temp Genre Table
insert into GenreTemp(genre)
select mo.genres
from Title2 as mo;


-- Converting to array
alter table GenreTemp
    alter genre type varchar[] using string_to_array(genre, ',');

-- Creating final Genre table
create table Genre(
        id serial primary key,
        genre varchar(255)
);

-- Inserting into Genre
insert into Genre(genre)
select distinct unnest(genre) from GenreTemp;


-- converting genres from original Title
alter table Title2
alter genres type varchar[] using string_to_array(genres,',');


-- Temp Title_Genre Table
create table Title_Genre_Temp(
        titleId integer,
        genres varchar(250)
);

-- Insert into Temp Title_Genre
insert into Title_Genre_Temp
select titleId, unnest(genres)
from Title2;


-- Title_Genre Table
create table Title_Genre(
        genre integer,
        title integer,
        primary key(genre, title)
);

-- Inserting into Final Title_Genre
insert into Title_Genre
select distinct ge.id, tp.titleId
from Title_Genre_Temp as tp
join Genre as ge on ge.genre = tp.genres;

-- setting foreign key constraints
ALTER TABLE Title_Genre ADD CONSTRAINT fk_titlegenre_tileId FOREIGN KEY(genre) REFERENCES Genre(id);
DELETE FROM Title_Genre WHERE NOT exists ( SELECT NULL FROM Title WHERE Title_Genre.title = Title.id);
```

```
ALTER TABLE Title_Genre ADD CONSTRAINT fk_titlegenre2_tileId FOREIGN KEY(title) REFERENCES Title(id);


-- Member Table
CREATE TABLE Member(
        id INTEGER PRIMARY KEY,
        name VARCHAR(255) NOT NULL,
        birthYear SMALLINT,
        deathYear SMALLINT
);


-- Inserting into Member
insert into Member
select pe.nameId, pe.primaryName, pe.birthYear, pe.deathYear
from Names_ as pe;

-- Title_Actor Table
create table Title_Actor(
        actor integer,
        title integer,
        primary key(actor,title)
);

-- insert into Title_Actor
INSERT INTO Title_Actor
SELECT distinct t2.nameId, t2.titleId
FROM Principals as t2
where t2.category = 'actor';

-- setting up foreign key contraints
ALTER TABLE Title_Actor ADD CONSTRAINT fk_titleactor_tileId FOREIGN KEY (actor) REFERENCES Member(id);
DELETE FROM Title_Actor WHERE NOT exists ( SELECT NULL FROM Title WHERE Title_Actor.title = Title.id);
ALTER TABLE Title_Actor ADD CONSTRAINT fk_titleactor2_tileId FOREIGN KEY (title) REFERENCES Title(id);


--Title_Writer Table
create table Title_Writer(
        writer integer,
        title integer,
        primary key(writer, title)
);

--Inserting into Title_Writer Table
insert into Title_Writer
select me.nameId, me.titleId
from Principals as me
where me.category = 'writer';

-- setting up foreign key contraints
ALTER TABLE Title_Writer ADD CONSTRAINT fk_titlewriter_tileId FOREIGN KEY (writer) REFERENCES Member(id);
DELETE FROM Title_Writer WHERE NOT exists ( SELECT NULL FROM Title WHERE Title_Writer.title = Title.id );
ALTER TABLE Title_Writer ADD CONSTRAINT fk_titlewriter2_tileId FOREIGN KEY (title) REFERENCES Title(id);



-- Title_Director Table
create table Title_Director(
        director integer,
        title integer,
        primary key(director, title)
);


-- Inserting into Title_Director Table
```

```
insert into Title_Director
select me.nameId, me.titleId
from Principals as me
where me.category = 'director';

-- setting up foreign key contraints
ALTER TABLE Title_Director ADD CONSTRAINT fk_titledirector_tileId FOREIGN KEY (director) REFERENCES
Member(id);
DELETE FROM Title_Director WHERE NOT exists ( SELECT NULL FROM Title WHERE Title_Director.title = Title.id );
ALTER TABLE Title_Director ADD CONSTRAINT fk_titledirector2_tileId FOREIGN KEY (title) REFERENCES Title(id);


-- Title_Producer Table
create table Title_Producer(
        producer integer,
        title integer,
        primary key(producer, title)
);


-- Inserting into Title_Producer Table
insert into Title_Producer
select me.nameId, me.titleId
from Principals as me
where me.category = 'producer';

-- setting up foreign key constraints
ALTER TABLE Title_Producer ADD CONSTRAINT fk_titleproducer_tileId FOREIGN KEY (producer) REFERENCES
Member(id);
DELETE FROM Title_Producer WHERE NOT exists ( SELECT NULL FROM Title WHERE Title_Producer.title = Title.id );
ALTER TABLE Title_Producer ADD CONSTRAINT fk_titleproducer2_tileId FOREIGN KEY (title) REFERENCES Title(id);


-- create temporary Character table
create table TempCharacter_(
        characterId serial primary key,
        characters_ text
);

-- inserting into temporary Character table
insert into TempCharacter_(characters_)
select me.characters_
from Principals as me;

-- preprocessing
update TempCharacter_ set characters_ = replace(characters_, '[', '');
update TempCharacter_ set characters_ = replace(characters_, ']', '');
update TempCharacter_ set characters_ = replace(characters_, '"', '');



-- converting characters_ to array
alter table TempCharacter_
    alter characters_ type text[] using string_to_array(characters_, ',');

-- create table Character
create table Character(
        id serial primary key,
        character text
);

-- insert into Character
insert into Character(character)
select distinct unnest(characters_) from TempCharacter_;
```

```
-- create 1st temp Actor_Title_Character table
Create Table Actor_Title_CharacterTemp1(actor integer,
                                                            titleId integer,
                                                            character_ text);

-- insert into 1st temp Actor_Title_Character table
insert into Actor_Title_CharacterTemp1
select pe.nameId,pe.titleId,pe.characters_
from principals as pe;


-- preprocessing
update Actor_Title_CharacterTemp1 set character_ = replace(character_, '[', '');
update Actor_Title_CharacterTemp1 set character_ = replace(character_, ']', '');
update Actor_Title_CharacterTemp1 set character_ = replace(character_, '"', '');

-- converting character_ to array
alter table Actor_Title_CharacterTemp1
alter character_ type varchar[] using string_to_array(character_,',');

-- create 2nd temp Actor_Title_Character table
Create Table Actor_Title_CharacterTemp2(actor integer,
                                                                titleId integer,
                                                                character_ varchar);

-- insert into 2nd temp Actor_Title_Character table
insert into Actor_Title_CharacterTemp2
select actor,titleId,unnest(character_) from Actor_Title_CharacterTemp1;

-- create table Actor_Title_Character
Create Table Actor_Title_Character(actor integer,
                                                       title integer,
                                                       character integer,
                                                       primary key (actor, title, character));

-- insert into Actor_Title_Character
insert into Actor_Title_Character
select distinct te.actor,te.titleId,c.characterId
from Actor_Title_CharacterTemp2 as te
join Character as c
on c.character = te.character_;


-- setting foreign constraints
ALTER TABLE Actor_Title_Character ADD CONSTRAINT fk_atc2_tileId FOREIGN KEY (character) REFERENCES
Character(id);
DELETE FROM Actor_Title_Character WHERE NOT exists ( SELECT NULL FROM Title_Actor WHERE
Actor_Title_Character.actor = Title_Actor.actor and Actor_Title_Character.title = Title_Actor.title );
ALTER TABLE Actor_Title_Character ADD CONSTRAINT fk_atc_tileId FOREIGN KEY (actor, title) REFERENCES
Title_Actor(actor, title);


— Question 2


-- Question 2.1
-- 139359 rows, Total query runtime: 1 secs 312 msec.
-- Number of invalid Title_Actor relationships with respect to characters.

SELECT count(*) as Number_of_invalid_relationships
FROM Title_Actor as t1
LEFT JOIN Actor_Title_Character as t2 ON t2.title = t1.title
Where t2.character is null;
```

```
-- Question 2.2
-- 8425 rows, Total query runtime: 1 secs 774 msec.
-- Alive actors whose name starts with “Phi” and did not participate in any movie in
2014.

select name as Actors
from Title as tt
join Title_Actor as ta on tt.id = ta.title
join Member as me on me.id = ta.actor
where name like 'Phi%' and deathYear is null and startyear <> 2014;
```

```
-- Question 2.3
-- 8 rows (count = 8), Total query runtime: 450 msec.
-- Producers who have produced the most talk shows in 2017 and whose name contains
“Gill”.

select name, count(id) as Number_of_Talk_Shows
from Member as me
join Title_Producer as tp on tp.producer = me.id
join Title as t on t.id = tp.title
join Title_Genre as tg on tg.title = t.id
join Genre as g on g.id = tg.genre
where me.deathYear is null and t.startYear = 2017 and g.genre = 'Talk-Show' and
me.name like '%Gill%'
group by name
order by count(t.id) DESC;
group by name
order by count(t.id) DESC;
```

```
-- Question 2.4
-- 24331 rows, Total query runtime: 520 msec.
-- Alive producers ordered by the greatest number of long-run titles produced
(runtime greater than 120 minutes)

select name, runtime as Number_of_long_run_titles
from Title as tt
join Title_Producer as tp on tt.id = tp.title
join Member as me on tp.producer = me.id
where me.deathYear is null and tt.runtime > 120
order by runtime desc;
```

```
-- Question 2.5
-- 87 rows, Total query runtime: 428 msec.
-- Alive actors who have portrayed Jesus Christ (simply look for a character with
this specific name)

select name as Actors from Member as me
join Actor_Title_Character as atc on
atc.actor = me.id
join Character as c on
```

```
c.id = atc.character
where c.character = 'Jesus Christ'
and deathYear is null;




— Question 5


-- Question 5
-- Abhishek Shah, as5553


-- Indexes
-- Title table index
create index TitleIndex
on Title (id);
create index TitleIndex2
on Title (id, runtime);
create index TitleIndex3
on Title (startYear);

-- Member table index
create index MemberIndex
on Member (id);
create index MemberIndex2
on Member (deathYear);
create index MemberIndex3
on Member (name);


-- Title_Actor table index
CREATE INDEX TitleActorIndex
ON Title_Actor (title);


-- Title_Genre table index
create index TitleGenreIndex
on Title_Genre(title);

-- Actor_Title_Character table index
CREATE INDEX ActorTitleCharacterIndex
ON Actor_Title_Character (title);
create index ActorTitleCharacterIndex2
on Actor_Title_Character (actor, character);


-- Title_Producer table index
create index ProducerIndex
on Title_Producer(producer);
create index ProducerIndex2
on Title_Producer(title);



-- Queries

-- Normal Runtime: Total query runtime: 846 msec.
-- Index  Runtime: Total query runtime: 593 msec.
-- Number of invalid Title_Actor relationships with respect to characters.
```

```
SELECT count(*) as Number_of_invalid_relationships
FROM Title_Actor as t1
LEFT JOIN Actor_Title_Character as t2 ON t2.title = t1.title
Where t2.character is null;

-----------------------------------------------------------------------------------

-- Normal Runtime: Total query runtime: 1 secs 312 msec
-- Index  Runtime: Total query runtime: 621 msec
-- Alive actors whose name starts with “Phi” and did not participate in any movie in
2014.

select name as Actors
from Title as tt
join Title_Actor as ta on tt.id = ta.title
join Member as me on me.id = ta.actor
where name like 'Phi%' and deathYear is null and startyear <> 2014;


-----------------------------------------------------------------------------------

-- Normal Runtime: Total query runtime: 676 msec.
-- Index  Runtime: Total query runtime: 450 msec.
-- Producers who have produced the most talk shows in 2017 and whose name contains
“Gill”.

select name, count(id) as Number_of_Talk_Shows
from Member as me
join Title_Producer as tp on tp.producer = me.id
join Title as t on t.id = tp.title
join Title_Genre as tg on tg.title = t.id
join Genre as g on g.id = tg.genre
where me.deathYear is null and t.startYear = 2017 and g.genre = 'Talk-Show' and
me.name like '%Gill%'
group by name
order by count(t.id) DESC;



-----------------------------------------------------------------------------------

-- Normal Runtime: Total query runtime: 519 msec.
-- Index  Runtime: Total query runtime: 368 msec.
-- Alive producers ordered by the greatest number of long-run titles produced
(runtime greater than 120 minutes)

select name, runtime as Number_of_long_run_titles
from Title as tt
join Title_Producer as tp on tt.id = tp.title
join Member as me on tp.producer = me.id
where me.deathYear is null and tt.runtime > 120
order by runtime desc;

------------------------------------------------------------------------------------

-- Normal Runtime: Total query runtime: 665 msec.
-- Index  Runtime: Total query runtime: 253 msec.
-- Alive actors who have portrayed Jesus Christ (simply look for a character with
this specific name)
```

```
select name as Actors from Member as me
join Actor_Title_Character as atc on
atc.actor = me.id
join Character as c on
c.id = atc.character
where c.character = 'Jesus Christ'
and deathYear is null;
```

```
-- Full text Indexing
-- Producers who have produced the most talk shows in 2017 and whose name contains “Gill”.

-- We use this for finding the text that contains the given string. 'Gill' in our case.

-- Let's also say that we want to carry out a full-text search on the data on the name column in the Member table.
-- We could add a new column to the table to store the list of lexemes.

-- add a new column to the table to store the preprocessed search document
ALTER TABLE Member ADD COLUMN ts tsvector
    GENERATED ALWAYS AS (to_tsvector('english', name)) STORED;

-- create a GIN index on ts:
CREATE INDEX ts_idx ON Member USING GIN (ts);


-- Normal Runtime: Total query runtime: 676 msec.
-- Index  Runtime: Total query runtime: 114 msec.
-- Producers who have produced the most talk shows in 2017 and whose name contains “Gill”.

select name, count(id) as Number_of_Talk_Shows
from Member as me
join Title_Producer as tp on tp.producer = me.id
join Title as t on t.id = tp.title
join Title_Genre as tg on tg.title = t.id
join Genre as g on g.id = tg.genre
where me.deathYear is null and t.startYear = 2017 and g.genre = 'Talk-Show' and ts @@ to_tsquery('english', 'Gill')
group by name
order by count(t.id) DESC;


-- As compared to searching without indexing(400 msec) or searching with normal indexing using Like '%Gill%' (450 msec)
-- Using full text indexing is way faster than both the methods (114 msec).
-- There is a huge performance boost due to the use of full text indexing.
```