INTERNATIONAL HELLENIC UNIVERSITY

MSc Data Science
Advanced Database Systems

Lazaros Panitsidis & Konstantinos Kravaritis

Table of Contents

Part A

Database Design (Table Specifications)

Actors Table

Attribute	Datatype	Constraint	Default	Description
actor_id	NUMBER(9)	PK, pk_actor_id		Unique ID of the actor
a_name	VARCHAR2(30)	CHECK, upper_a_name NOT NULL		First name
a_surname	VARCHAR2(35)	CHECK, upper_a_surname NOT NULL		Last name
a_gender	CHAR(1)	CHECK, check_a_gender		Actor's Gender (Male, Female, Other)
a_email	VARCHAR2(50)	CHECK, upper_a_email UNIQUE, unique_a_email NOT NULL		E-mail address
a_address	VARCHAR2(30)	CHECK, upper_a_address	NULL	Actor's address
a_city	VARCHAR2(30)	CHECK, upper_a_city	NULL	Actor's city
a_country	VARCHAR2(30)	CHECK, upper_a_country	NULL	Actor's country
a_birthdate	DATE	TRIGGER, check_actor_age_trigger	NULL	Date of birth
a_contact_num	VARCHAR2(30)	UNIQUE, unique_a_contact_num NOT NULL		Contact number

Series Table

Attribute	Datatype	Constraint	Default	Description
series_id	NUMBER(8)	PK, pk_series_id		Unique ID of the series
title	VARCHAR2(50)	CHECK , upper_title NOT NULL		Series name
description	VARCHAR2(400)	CHECK , upper_description	NULL	Short synopsis
production_year	DATE	CHECK, check_production_year	NULL	Production year
num_of_seasons	NUMBER(2)			Number of Seasons
genre	VARCHAR2(30)	CHECK, upper_genre	NULL	Type of series

Episodes Table

Attribute	Datatype	Constraint	Default	Description
episode_id	NUMBER(9)	PK, pk_episode_id		Unique ID of the episode
title	VARCHAR2(50)	CHECK, upper_episode_title NOT NULL		Episode name
season_num	NUMBER(2)		NULL	Number of seasons
episode_num	NUMBER(5)			Episode number in the season
duration	NUMBER(5)		NULL	Length of episode in seconds
summary	VARCHAR2(400)	CHECK , upper_summary		Short summary

Viewers Table

Attribute	Datatype	Constraint	Default	Description
viewer_id	NUMBER(10)	PK, pk_viewer_id		Unique ID of the viewer
v_name	VARCHAR2(30)	CHECK, upper_v_name NOT NULL		First name
v_surname	VARCHAR2(35)	CHECK, upper_v_surname NOT NULL		Last name
v_gender	CHAR(1)	CHECK, check_v_gender	NULL	Viewer's gender (Male, Female, Other)
v_email	VARCHAR2(50)	CHECK, upper_v_email UNIQUE, unique_v_email NOT NULL		E-mail address
v_address	VARCHAR2(30)	CHECK, upper_v_address	NULL	Viewer's address
v_city	VARCHAR2(30)	CHECK, upper_v_city	NULL	Viewer's city
v_country	VARCHAR2(30)	CHECK, upper_v_country	NULL	Viewer's country
v_birthdate	DATE	TRIGGER, check_viewer_age_trigger	NULL	Date of birth
v_contact_num	VARCHAR2(30)	UNIQUE, unique_v_contact_num NOT NULL		Contact number

Casting Table

Attribute	Datatype	Constraint	Default	Description
casting_id	NUMBER(9)	PK, pk_casting_id		Unique Casting ID
actor_id	NUMBER(9)	FK, fk_actor_id		Unique ID of the actor
series_id	NUMBER(9)	FK, fk_series_id		Unique Series ID
c_role	VARCHAR2(30)	CHECK, upper_c_role		Role of the actor

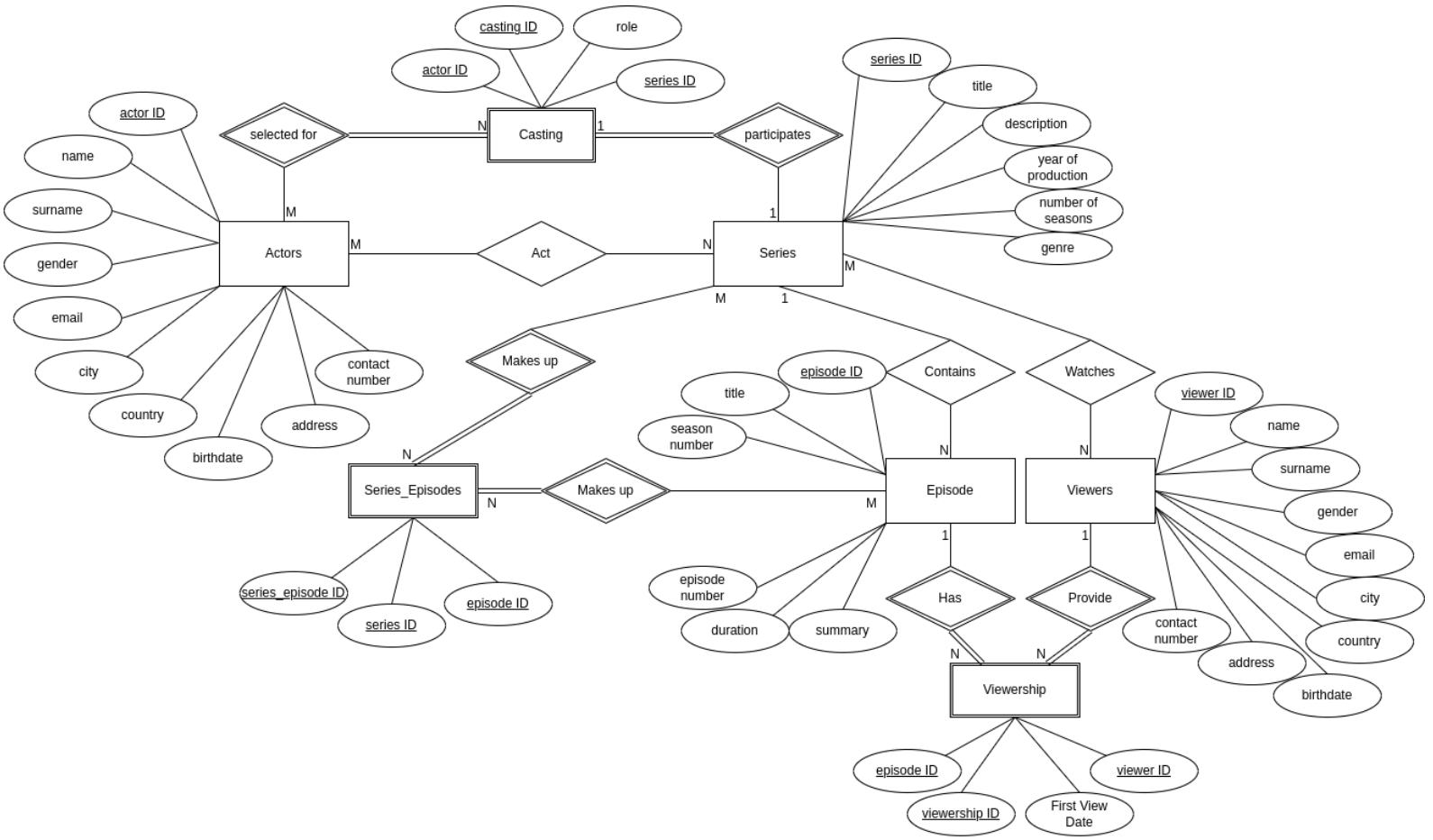
Viewership Table

Attribute	Datatype	Constraint	Default	Description
viewership_id	NUMBER(10)	PK, pk_viewership_id		Unique Viewership ID
episode_id	NUMBER(9)	FK, fk_episode_id		Unique ID of the actor
viewer_id	NUMBER(9)	FK, fk_viewer_id		Unique ID of the viewer
first_view_date	DATE	TRIGGER, check_first_view_trigger		Date that the viewer watched the episode for the first time

Series_Episodes Table

Attribute	Datatype	Constraint	Default	Description
series_episode_id	NUMBER(10)	PK, pk_series_episode_id		Unique ID of the series_episodes
series_id	NUMBER(8)	FK, fk_series_id_series_episode		Unique ID of the series
episode_id	NUMBER(9)	FK, fk_episode_id_series_episode		Unique ID of the episode

Entity Relationship Diagram



Test Plan

ID	Test	Element Tested	Expected Result	Actual Result
1	`SELECT * FROM Actors;`	Query 5)a) All columns from the Actors Table	Return 10 columns with 15 rows of data	As expected

```
SQL> SELECT * FROM ACTORS;

  ACTOR_ID A_NAME         A_SURNAME       A_GENDER   A_EMAIL                      A_ADDRESS       A_CITY          A_COUNTRY       A_BIRTHDATE     A_CONTACT_NUM
---------- -------------- --------------- ---------- ---------------------------- --------------- --------------- --------------- --------------- ---------------
         1 BRYAN          CRANSTON        M          BRYAN.CRANSTON@EMAIL.COM     123 MAIN ST     NEW YORK        USA             01-JAN-80       1234567890
         2 KAREN          FUKUHARA        F          KAREN.FUKUHARA@EMAIL.COM     456 OAK ST      TOKYO           JAPAN           15-FEB-92       9876543210
         3 AIDAN          GILLEN          M          AIDAN.GILLEN@EMAIL.COM       789 PINE ST     PARIS           FRANCE          10-APR-85       5551112222
         4 OLIVIA         COLMAN          F          OLIVIA.COLMAN@EMAIL.COM      101 ELM ST      LONDON          UK              22-JUN-77       9998887777
         5 GUY            PIERCE          M          GUY.PIERCE@EMAIL.COM         202 CEDAR ST    SYDNEY          AUSTRALIA       05-AUG-90       3332221111
         6 KATEE          SACKHOFF        F          KATEE.SACKHOFF@EMAIL.COM     303 BIRCH ST    BERLIN          GERMANY         30-SEP-82       7776665555
         7 DAN            CASTELLANETA    M          DAN.CASTELLANETA@EMAIL.COM   404 MAPLE ST    ROME            ITALY           12-NOV-74       4443332222
         8 UNA            STUBBS          F          UNA.STUBBS@EMAIL.COM         505 SPRUCE ST   TORONTO         CANADA          25-DEC-88       1110009999
         9 JAVIER         BARDEM          M          JAVIER.BARDEM@EMAIL.COM      606 WALNUT ST   MADRID          SPAIN           18-JUL-99       6665554444
        10 ANGELA         KINSEY          F          ANGELA.KINSEY@EMAIL.COM      707 PINE ST     SEOUL           SOUTH KOREA     09-MAR-87       2228883333
        11 RAUL           MENDEZ          M          RAUL.MENDEZ@EMAIL.COM        808 CEDAR ST    RIO DE JANEIRO  BRAZIL          03-MAY-95       9990001111
        12 LEONARDO       NAM             M          LEO.NAM@EMAIL.COM            909 OAK ST      BEIJING         CHINA           14-JUL-81       4445556666
        13 HENRY          CAVILLE         M          HENRY.CAVILLE@EMAIL.COM      111 PINE ST     CAPE TOWN       SOUTH AFRICA    27-SEP-73       1119998888
        14 NATALIA        DYER            F          NATALIA.DYER@EMAIL.COM       222 ELM ST      MOSCOW          RUSSIA          08-NOV-76       7773335555
        15 KUNAL          NAYYAR          M          KUNAL.NAYYAR@EMAIL.COM       333 OAK ST      MUMBAI          INDIA           20-JUL-89       3337771111

15 rows selected.
```

2	`SELECT a_name as ACTORS_NAME,`
	`a_surname as ACTORS_SURNAME,`
	`a_gender as ACTORS_GENDER`
	`FROM Actors ORDER BY a_surname DESC;`
		Query 5)b)	Return 3 columns with 15 rows of data	As expected

```
SQL> SELECT a_name as ACTORS_NAME, a_surname as ACTORS_SURNAME, a_gender as ACTORS_GENDER
  2  FROM Actors ORDER BY a_surname DESC;

ACTORS_NAME     ACTORS_SURNAME  ACTORS_GENDER
--------------- --------------- ---------------
UNA             STUBBS          F
KATEE           SACKHOFF        F
GUY             PIERCE          M
KUNAL           NAYYAR          M
LEONARDO        NAM             M
RAUL            MENDEZ          M
ANGELA          KINSEY          F
AIDAN           GILLEN          M
KAREN           FUKUHARA        F
NATALIA         DYER            F
BRYAN           CRANSTON        M
OLIVIA          COLMAN          F
HENRY           CAVILLE         M
DAN             CASTELLANETA    M
JAVIER          BARDEM          M

15 rows selected.
```

3	`SELECT a_name as ACTORS_NAME,`
	`a_surname as ACTORS_SURNAME,`
	`a_country as COUNTRY_OF_RESIDENCE`
	`FROM Actors`
	`WHERE a_country = 'USA';`
		Query 5)c)1)	Return 3 columns with a number of rows equal to actors whose country of origin is the USA.	As expected (1 row returned)


```
SQL> SELECT a_name as ACTORS_NAME, a_surname as ACTORS_SURNAME, a_country as COUNTRY_OF_RESIDENCE
  2  FROM Actors
  3  WHERE a_country = 'USA';

ACTORS_NAME                    ACTORS_SURNAME                     COUNTRY_OF_RESIDENCE
------------------------------ ---------------------------------- ------------------------------
BRYAN                          CRANSTON                           USA
```

4	`SELECT a_name as ACTORS_NAME, a_surname as ACTORS_SURNAME FROM Actors WHERE a_surname LIKE 'C%STON';`	Query 5)c)2)	Return an actor surname that starts with C and ends with STON	As expected

```
SQL> SELECT a_name as ACTORS_NAME, a_surname as ACTORS_SURNAME
  2  FROM Actors
  3  WHERE a_surname LIKE 'C%STON';

ACTORS_NAME                    ACTORS_SURNAME
------------------------------ ----------------------------------
BRYAN                          CRANSTON
```

5	`SELECT a_name as ACTORS_NAME, a_surname as ACTORS_SURNAME, a_city as CITY_OF_RESIDENCE FROM Actors WHERE a_city != 'LONDON';`	Query 5)c)3)	Return all actors names and surnames whose city of residence is not London	As expected

```
SQL> SELECT a_name as ACTORS_NAME, a_surname as ACTORS_SURNAME, a_city as CITY_OF_RESIDENCE
  2  FROM Actors
  3  WHERE a_city != 'LONDON';

ACTORS_NAME                    ACTORS_SURNAME                     CITY_OF_RESIDENCE
------------------------------ ---------------------------------- ------------------------------
BRYAN                          CRANSTON                           NEW YORK
KAREN                          FUKUHARA                           TOKYO
AIDAN                          GILLEN                             PARIS
GUY                            PIERCE                             SYDNEY
KATEE                          SACKHOFF                           BERLIN
DAN                            CASTELLANETA                       ROME
UNA                            STUBBS                             TORONTO
JAVIER                         BARDEM                             MADRID
ANGELA                         KINSEY                             SEOUL
RAUL                           MENDEZ                             RIO DE JANEIRO
LEONARDO                       NAM                                BEIJING
HENRY                          CAVILLE                            CAPE TOWN
NATALIA                        DYER                               MOSCOW
KUNAL                          NAYYAR                             MUMBAI

14 rows selected.
```

6	`SELECT title, production_year`
	`FROM Series`
	`WHERE production_year BETWEEN`
	`'1-JAN-2000' AND '31-DEC-2010';`
		Query 5)c)4)	Return the titles and production year of series produced between 1-1-2000 and 31-12-2010	As expected

```
SQL> SELECT title, production_year
  2  FROM Series
  3  WHERE production_year BETWEEN '1-JAN-2000' AND '31-DEC-2010';

TITLE                     PRODUCTION_YEAR
------------------------- ---------------
BREAKING BAD              20-JAN-08
SHERLOCK                  25-JUL-10
THE OFFICE                24-MAR-05
THE BIG BANG THEORY       24-SEP-07
```

7	`SELECT a_name as ACTORS_NAME,`
	`a_surname as ACTORS_SURNAME, c_role`
	`as ROLE`
	`FROM Actors a`
	`JOIN Casting c ON a.actor_id =`
	`c.casting_id;`
		Query 5)d)1)	Return a new table with 3 columns by joining the tables Actors and Casting (15 rows returned)	As expected

```
SQL> SELECT a_name as ACTORS_NAME, a_surname as ACTORS_SURNAME, c_role as ROLE
  2  FROM Actors a
  3  JOIN Casting c ON a.actor_id = c.casting_id;

ACTORS_NAME               ACTORS_SURNAME                    ROLE
------------------------- --------------------------------- ------------------------------
BRYAN                     CRANSTON                          LEAD ACTOR
KAREN                     FUKUHARA                          SUPPORTING ACTOR
AIDAN                     GILLEN                            MAIN CHARACTER
OLIVIA                    COLMAN                            MAIN CHARACTER
GUY                       PIERCE                            GUEST ACTRESS
KATEE                     SACKHOFF                          MAIN CHARACTER
DAN                       CASTELLANETA                      SUPPORTING ACTOR
UNA                       STUBBS                            LEAD ACTRESS
JAVIER                    BARDEM                            GUEST ACTRESS
ANGELA                    KINSEY                            GUEST ACTOR
RAUL                      MENDEZ                            MAIN CHARACTER
LEONARDO                  NAM                               SUPPORTING ACTRESS
HENRY                     CAVILLE
NATALIA                   DYER                              SUPPORTING ACTRESS
KUNAL                     NAYYAR                            LEAD ACTOR

15 rows selected.
```

8	`SELECT a_name as ACTORS_NAME,`
	`a_surname as ACTORS_SURNAME, title`
	`FROM Actors a`
	`JOIN Casting c ON a.actor_id =`
	`c.actor_id`
	`JOIN Series s ON s.series_id =`
	`c.series_id;`
		Query 5)d)2)	Return the actor's names and surnames and the title of the series in which they participated.	As expected

```
SQL> SELECT a_name as ACTORS_NAME, a_surname as ACTORS_SURNAME, title
  2  FROM Actors a
  3  JOIN Casting c ON a.actor_id = c.actor_id
  4  JOIN Series s ON s.series_id = c.series_id;

ACTORS_NAME                 ACTORS_SURNAME                   TITLE
--------------------------- -------------------------------- ------------------------
BRYAN                       CRANSTON                         BREAKING BAD
KAREN                       FUKUHARA                         THE BOYS
AIDAN                       GILLEN                           GAME OF THRONES
AIDAN                       GILLEN                           THE CROWN
OLIVIA                      COLMAN                           THE CROWN
BRYAN                       CRANSTON                         FRIENDS
GUY                         PIERCE                           FRIENDS
KATEE                       SACKHOFF                         THE MANDALORIAN
JAVIER                      BARDEM                           THE SIMPSONS
UNA                         STUBBS                           SHERLOCK
BRYAN                       CRANSTON                         BLACK MIRROR
KATEE                       SACKHOFF                         BLACK MIRROR
LEONARDO                    NAM                              BLACK MIRROR
ANGELA                      KINSEY                           THE OFFICE
JAVIER                      BARDEM                           NARCOS
RAUL                        MENDEZ                           NARCOS
OLIVIA                      COLMAN                           WESTWORLD
NATALIA                     DYER                             STRANGER THINGS
KAREN                       FUKUHARA                         THE BIG BANG THEORY

19 rows selected.
```

9	`SELECT COUNT(*) as VIEWER_COUNT, v_country AS VIEWER_COUNTRY FROM Viewers GROUP BY v_country;`	Query 5)e)2)	Return the total number of viewers per country (All viewers live in Greece)	As expected

```
SQL> SELECT COUNT(*) as VIEWER_COUNT, v_country AS VIEWER_COUNTRY
  2  FROM Viewers
  3  GROUP BY v_country;

VIEWER_COUNT VIEWER_COUNTRY
------------ ----------------------------
          15 GREECE
```

10	`SELECT e.title AS episode_title, COUNT(*) AS view_count FROM Viewership v JOIN Episodes e ON v.episode_id = e.episode_id GROUP BY e.title ORDER BY view_count DESC FETCH FIRST 1 ROW ONLY;`	Query 5)e)3)	Return the episode titles watched by the most viewers (but only the top row)	As expected

```
SQL> SELECT e.title AS episode_title, COUNT(*) AS view_count
  2  FROM Viewership v
  3  JOIN Episodes e ON v.episode_id = e.episode_id
  4  GROUP BY e.title
  5  ORDER BY view_count DESC
  6  FETCH FIRST 1 ROW ONLY;

EPISODE_TITLE                  VIEW_COUNT
------------------------------ ----------
WINTER IS COMING                        4
```

11	`SELECT s.title AS series_title, COUNT(*) AS view_count FROM Viewership v JOIN Series_Episodes se ON v.episode_id = se.episode_id JOIN Series s ON se.series_id = s.series_id GROUP BY s.title ORDER BY view_count DESC FETCH FIRST 1 ROW ONLY;`	Query 5)e)4)	Return the series titles watched by the most viewers (but only the top row)	As expected

```
SQL> SELECT s.title AS series_title, COUNT(*) AS view_count
  2  FROM Viewership v
  3  JOIN Series_Episodes se ON v.episode_id = se.episode_id
  4  JOIN Series s ON se.series_id = s.series_id
  5  GROUP BY s.title
  6  ORDER BY view_count DESC
  7  FETCH FIRST 1 ROW ONLY;

SERIES_TITLE                                       VIEW_COUNT
-------------------------------------------------- ----------
GAME OF THRONES                                             4
```

12	`SELECT a_name as ACTORS_NAME, a_surname as ACTORS_SURNAME, title AS series_title FROM Actors a JOIN Series s ON a.actor_id = s.series_id ORDER BY actor_id ASC;`	Query 5)e)5)	Return 3 columns with the actor's names, surnames in ascending order by the actors ID and series title they participated in.	As expected

```
SQL> SELECT a_name as ACTORS_NAME, a_surname as ACTORS_SURNAME, title AS series_title
  2  FROM Actors a
  3  JOIN Series s ON a.actor_id = s.series_id
  4  ORDER BY actor_id ASC;

ACTORS_NAME                  ACTORS_SURNAME                   SERIES_TITLE
---------------------------- -------------------------------- --------------------------------------------------
BRYAN                        CRANSTON                         BREAKING BAD
KAREN                        FUKUHARA                         THE BOYS
AIDAN                        GILLEN                           GAME OF THRONES
OLIVIA                       COLMAN                           THE CROWN
GUY                          PIERCE                           FRIENDS
KATEE                        SACKHOFF                         THE MANDALORIAN
DAN                          CASTELLANETA                     THE SIMPSONS
UNA                          STUBBS                           SHERLOCK
JAVIER                       BARDEM                           BLACK MIRROR
ANGELA                       KINSEY                           THE OFFICE
RAUL                         MENDEZ                           NARCOS
LEONARDO                     NAM                              WESTWORLD
HENRY                        CAVILLE                          THE WITCHER
NATALIA                      DYER                             STRANGER THINGS
KUNAL                        NAYYAR                           THE BIG BANG THEORY

15 rows selected.
```

13	`WITH ActorSeriesCounts AS (`
	`  SELECT a.actor_id, a.a_name, a.a_surname, COUNT(DISTINCT c.series_id) AS series_count`
	`  FROM Casting c`
	`  JOIN Actors a ON c.actor_id = a.actor_id`
	`  GROUP BY a.actor_id, a.a_name, a.a_surname`
	`)`
	`SELECT actor_id, a_name, a_surname, series_count`
	`FROM ActorSeriesCounts`
	`WHERE series_count = (SELECT MAX(series_count) FROM ActorSeriesCounts);`
		Query 5)e)6)	Return the actor's ID, name and surname who has participated in the most series.	As expected

```
SQL> WITH ActorSeriesCounts AS (
  2      SELECT a.actor_id, a.a_name, a.a_surname, COUNT(DISTINCT c.series_id) AS series_count
  3      FROM Casting c
  4      JOIN Actors a ON c.actor_id = a.actor_id
  5      GROUP BY a.actor_id, a.a_name, a.a_surname
  6  )
  7  SELECT actor_id, a_name, a_surname, series_count
  8  FROM ActorSeriesCounts
  9  WHERE series_count = (SELECT MAX(series_count) FROM ActorSeriesCounts);

  ACTOR_ID A_NAME          A_SURNAME       SERIES_COUNT
---------- --------------- --------------- ------------
         1 BRYAN           CRANSTON                   3
```

14	WITH ActorEpisodeCounts AS (SELECT a.actor_id, a.a_name, a.a_surname, COUNT(DISTINCT e.episode_id) AS episode_count FROM Casting c JOIN Series_Episodes se ON c.series_id = se.series_id JOIN Episodes e ON se.episode_id = e.episode_id JOIN Actors a ON c.actor_id = a.actor_id GROUP BY a.actor_id, a.a_name, a.a_surname) SELECT actor_id, a_name, a_surname, episode_count FROM ActorEpisodeCounts WHERE episode_count = (SELECT MAX(episode_count) FROM ActorEpisodeCounts);	Query 5)e)7)	Return the actor's ID, name and surname who has participated in the most episodes.	As expected

```
SQL> WITH ActorEpisodeCounts AS (
  2      SELECT a.actor_id, a.a_name, a.a_surname, COUNT(DISTINCT e.episode_id) AS episode_count
  3      FROM Casting c
  4      JOIN Series_Episodes se ON c.series_id = se.series_id
  5      JOIN Episodes e ON se.episode_id = e.episode_id
  6      JOIN Actors a ON c.actor_id = a.actor_id
  7      GROUP BY a.actor_id, a.a_name, a.a_surname
  8  )
  9  SELECT actor_id, a_name, a_surname, episode_count
 10  FROM ActorEpisodeCounts
 11  WHERE episode_count = (SELECT MAX(episode_count) FROM ActorEpisodeCounts);

  ACTOR_ID A_NAME          A_SURNAME       EPISODE_COUNT
---------- --------------- --------------- -------------
         1 BRYAN           CRANSTON                  3
```

15	SELECT title, num_of_seasons FROM Series ORDER BY num_of_seasons DESC FETCH FIRST ROW ONLY;	Query 5)e)8)	Return the series title with the highest number of seasons	As expected

```
SQL> SELECT title, num_of_seasons
  2  FROM Series
  3  ORDER BY num_of_seasons DESC
  4  FETCH FIRST ROW ONLY;

TITLE                                              NUM_OF_SEASONS
-------------------------------------------------- --------------
THE SIMPSONS                                               33
```

16	WITH SeriesEpisodesCount AS (SELECT s.title, COUNT(e.episode_id) as number_of_episodes FROM Series s JOIN Series_Episodes se ON s.series_id = se.series_id JOIN Episodes e ON se.episode_id = e.episode_id GROUP BY s.title) SELECT sec.title, sec.number_of_episodes FROM SeriesEpisodesCount sec WHERE sec.number_of_episodes = (SELECT MAX(number_of_episodes) FROM SeriesEpisodesCount);	Query 5)e)9)	Return the series's titles with the highest number of episodes	As expected

```
SQL> WITH SeriesEpisodesCount AS (
  2      SELECT s.title, COUNT(e.episode_id) as number_of_episodes
  3      FROM Series s
  4      JOIN Series_Episodes se ON s.series_id = se.series_id
  5      JOIN Episodes e ON se.episode_id = e.episode_id
  6      GROUP BY s.title
  7  )
  8  SELECT sec.title, sec.number_of_episodes
  9  FROM SeriesEpisodesCount sec
 10  WHERE sec.number_of_episodes = (SELECT MAX(number_of_episodes) FROM SeriesEpisodesCount);

TITLE                                              NUMBER_OF_EPISODES
-------------------------------------------------- ------------------
SHERLOCK                                                            2
THE WITCHER                                                         2
```

17	SELECT title, production_year FROM Series WHERE TO_CHAR(production_year, 'YYYY') BETWEEN '2000' AND '2009';	Query 5)e)10)	Return the titles of the series produced between 2000 and 2009	As expected

```
SQL> SELECT title, production_year
  2  FROM Series
  3  WHERE TO_CHAR(production_year, 'YYYY')
  4  BETWEEN '2000' AND '2009';

TITLE                     PRODUCTION_YEAR
------------------------- ---------------
BREAKING BAD              20-JAN-08
THE OFFICE                24-MAR-05
THE BIG BANG THEORY       24-SEP-07
```

18	SELECT a_name as ACTORS_NAME, a_surname as ACTORS_SURNAME, a_birthdate as ACTORS_BIRTHDATE FROM Actors WHERE TO_CHAR(a_birthdate, 'MON') = 'JUL';	Query 5)e)11)	Return the actor's names, surnames that were born the month July	As expected

```
SQL> SELECT a_name as ACTORS_NAME, a_surname as ACTORS_SURNAME, a_birthdate as ACTORS_BIRTHDATE
  2  FROM Actors
  3  WHERE TO_CHAR(a_birthdate, 'MON') = 'JUL';

ACTORS_NAME                    ACTORS_SURNAME                     ACTORS_BIRTHDATE
------------------------------ ---------------------------------- --------------------
JAVIER                         BARDEM                             18-JUL-99
LEONARDO                       NAM                                14-JUL-81
KUNAL                          NAYYAR                             20-JUL-89
```

19	SELECT a_name as ACTORS_NAME, a_surname as ACTORS_SURNAME, c_role as ROLE FROM Actors a JOIN Casting c ON a.actor_id = c.actor_id WHERE c.series_id IS NULL or c_role IS NULL;	Query 5)e)12)	Return the actor's names and surnames that haven't participated in a series	As expected

```
SQL> SELECT a_name as ACTORS_NAME, a_surname as ACTORS_SURNAME, c_role as ROLE
  2  FROM Actors a JOIN Casting c ON a.actor_id = c.actor_id
  3  WHERE c.series_id IS NULL or c_role IS NULL;

ACTORS_NAME                    ACTORS_SURNAME                     ROLE
------------------------------ ---------------------------------- ------------------------------
DAN                            CASTELLANETA
HENRY                          CAVILLE
```

20	SELECT v_name as VIEWER_NAME, v_surname as VIEWER_SURNAME, v_birthdate as VIEWER_BIRTHDATE, v_gender as VIEWER_GENDER FROM Viewers WHERE v_gender = 'F' ORDER BY v_birthdate FETCH FIRST ROW ONLY;	Query 5)e)13)	Return the name, surname, birthdate (and gender) of the oldest female viewer.	As expected

```
SQL> SELECT v_name as VIEWER_NAME, v_surname as VIEWER_SURNAME, v_birthdate as VIEWER_BIRTHDATE, v_gender as VIEWER_GENDER
  2  FROM Viewers
  3  WHERE v_gender = 'F'
  4  ORDER BY v_birthdate
  5  FETCH FIRST ROW ONLY;

VIEWER_NAME          VIEWER_SURNAME       VIEWER_BIRTHDATE     VIEWER_GENDER
-------------------- -------------------- -------------------- --------------
SOFIA                GEORGIOU             30-SEP-87            F
```

21	SELECT v_name as VIEWER_NAME, v_surname as VIEWER_SURNAME, v_address as VIEWER_ADDRESS FROM Viewers WHERE v_address LIKE '%ODOS%';	Query 5)e)14)	Return the rows with the viewer's name, surname and address which contains the word ODOS in it.	As expected

```
SQL> SELECT v_name as VIEWER_NAME, v_surname as VIEWER_SURNAME, v_address as VIEWER_ADDRESS
  2  FROM Viewers
  3  WHERE v_address LIKE '%ODOS%';

VIEWER_NAME          VIEWER_SURNAME       VIEWER_ADDRESS
-------------------- -------------------- ------------------------------
ELENI                KONSTANTINIDOU       ODOS THESSALONIKIS 456
STAVROS              AVGERIS              ODOS IRAKLIOU 808
ATHINA               PAPOUTSOGLOU         ODOS LARISAS 222
```

22	SELECT s.title, AVG(e.duration) as AVERAGE_EPISODE_LENGTH FROM Series s JOIN Series_Episodes se ON s.series_id = se.series_id JOIN Episodes e ON se.episode_id = e.episode_id GROUP BY s.title;	Query 5)e)15)	Return all the series titles in descending order according to their average episode length.	As expected

```
SQL> SELECT s.title, AVG(e.duration) as AVERAGE_EPISODE_LENGTH
  2  FROM Series s
  3  JOIN Series_Episodes se ON s.series_id = se.series_id
  4  JOIN Episodes e ON se.episode_id = e.episode_id
  5  GROUP BY s.title;

TITLE                                          AVERAGE_EPISODE_LENGTH
---------------------------------------------- ----------------------
BREAKING BAD                                                     3600
THE BOYS                                                         3600
GAME OF THRONES                                                  3720
FRIENDS                                                          1320
THE MANDALORIAN                                                  1800
THE SIMPSONS                                                     1320
SHERLOCK                                                         5400
BLACK MIRROR                                                     3600
THE OFFICE                                                       1320
NARCOS                                                           2700
WESTWORLD                                                        3600
THE WITCHER                                                      3600
THE BIG BANG THEORY                                              1320

13 rows selected.
```

23	WITH ViewerEpisodeCounts AS (SELECT vi.viewer_id, COUNT(vi.episode_id) AS episode_count FROM Viewership vi	Query 5)e)16)	Return the viewer's details (ID, name, surname) who has watched the most episodes.	As expected. In our data's case we have 2 viewers who have watched

	GROUP BY vi.viewer_id) SELECT ve.viewer_id, v.v_name, v.v_surname, ve.episode_count FROM ViewerEpisodeCounts ve JOIN Viewers v ON ve.viewer_id = v.viewer_id WHERE ve.episode_count = (SELECT MAX(episode_count) FROM ViewerEpisodeCounts) ORDER BY ve.viewer_id;			the same number of episodes.

```
SQL> WITH ViewerEpisodeCounts AS (
  2      SELECT vi.viewer_id, COUNT(vi.episode_id) AS episode_count
  3      FROM Viewership vi
  4      GROUP BY vi.viewer_id
  5  )
  6  SELECT ve.viewer_id, v.v_name, v.v_surname, ve.episode_count
  7  FROM ViewerEpisodeCounts ve
  8  JOIN Viewers v ON ve.viewer_id = v.viewer_id
  9  WHERE ve.episode_count = (SELECT MAX(episode_count) FROM ViewerEpisodeCounts)
 10  ORDER BY ve.viewer_id;

VIEWER_ID V_NAME                       V_SURNAME                          EPISODE_COUNT
--------- ---------------------------- ---------------------------------- -------------
        1 DIMITRIS                     PAPADOPOULOS                               3
        5 VASILIS                      PAPANIKOLAOU                               3
```

24	WITH ViewerEpisodeCounts AS (SELECT vi.viewer_id, COUNT(vi.episode_id) AS episode_count FROM Viewership vi JOIN Viewers v ON vi.viewer_id = v.viewer_id WHERE v.v_city = 'THESSALONIKI' GROUP BY vi.viewer_id) SELECT ve.viewer_id, v.v_name, v.v_surname, ve.episode_count FROM ViewerEpisodeCounts ve JOIN Viewers v ON ve.viewer_id = v.viewer_id WHERE ve.episode_count = (SELECT MAX(episode_count) FROM ViewerEpisodeCounts);	Query 5)e)17)	Return the viewer's details (ID, name, surname) who has watched the most episodes and lives in Thessaloniki.	As expected. In our data's case we have several viewers who have watched the same number of episodes (1) and live in Thessaloniki.

```
SQL> WITH ViewerEpisodeCounts AS (
  2      SELECT vi.viewer_id, COUNT(vi.episode_id) AS episode_count
  3      FROM Viewership vi
  4      JOIN Viewers v ON vi.viewer_id = v.viewer_id
  5      WHERE v.v_city = 'THESSALONIKI'
  6      GROUP BY vi.viewer_id
  7  )
  8  SELECT ve.viewer_id, v.v_name, v.v_surname, ve.episode_count
  9  FROM ViewerEpisodeCounts ve
 10  JOIN Viewers v ON ve.viewer_id = v.viewer_id
 11  WHERE ve.episode_count = (SELECT MAX(episode_count) FROM ViewerEpisodeCounts);

VIEWER_ID V_NAME                       V_SURNAME                          EPISODE_COUNT
--------- ---------------------------- ---------------------------------- -------------
        6 SOFIA                        GEORGIOU                                   1
        9 GIANNIS                      ANASTASIOU                                 1
       12 KONSTANTINA                  DIMITRIOU                                  1
       13 CHRISTOS                     KARALIS                                    1
       15 PETROS                       PAPADIMITRIOU                              1
        2 ELENI                        KONSTANTINIDOU                             1

6 rows selected.
```

25	WITH GenderSeriesViewerCounts AS (SELECT v.v_gender as viewer_gender, se.series_id, s.title AS series_title, COUNT(vi.viewer_id) AS viewer_count FROM Viewership vi JOIN Viewers v ON vi.viewer_id = v.viewer_id JOIN Series_Episodes se ON vi.episode_id = se.episode_id JOIN Series s ON se.series_id = s.series_id GROUP BY v.v_gender, se.series_id, s.title) SELECT gsvc.viewer_gender, gsvc.series_id, gsvc.series_title, gsvc.viewer_count FROM GenderSeriesViewerCounts gsvc WHERE gsvc.viewer_count = (SELECT MAX(viewer_count) FROM GenderSeriesViewerCounts WHERE viewer_gender = gsvc.viewer_gender);	Query 5)e)18)	Return the series' titles whose episodes were watched the most times per gender.	As expected we get as a result 2 rows of data. 1 row for <u>males</u> and the most popular series title for that gender. 1 row for <u>females</u> and the most popular series title for that gender. No rows for the <u>Other</u> gender option since no such data were inserted.

```
SQL> WITH GenderSeriesViewerCounts AS (
  2      SELECT
  3          v.v_gender as viewer_gender,
  4          se.series_id,
  5          s.title AS series_title,
  6          COUNT(vi.viewer_id) AS viewer_count
  7      FROM
  8          Viewership vi
  9      JOIN Viewers v ON vi.viewer_id = v.viewer_id
 10      JOIN Series_Episodes se ON vi.episode_id = se.episode_id
 11      JOIN Series s ON se.series_id = s.series_id
 12      GROUP BY v.v_gender, se.series_id, s.title
 13  )
 14  SELECT
 15      gsvc.viewer_gender,
 16      gsvc.series_id,
 17      gsvc.series_title,
 18      gsvc.viewer_count
 19  FROM
 20      GenderSeriesViewerCounts gsvc
 21  WHERE
 22      gsvc.viewer_count = (
 23          SELECT MAX(viewer_count)
 24          FROM GenderSeriesViewerCounts
 25          WHERE viewer_gender = gsvc.viewer_gender
 26      );

V SERIES_ID SERIES_TITLE                                      VIEWER_COUNT
- --------- -------------------------------------------------- ------------
M         3 GAME OF THRONES                                             3
F         7 THE SIMPSONS                                                2
```

26	`SELECT`
	`  viewer_id,`
	`  v_name AS first_name,`
	`  v_surname AS last_name,`
	`  CONCAT(SUBSTR(v_name, 1, 4),`
	`SUBSTR(v_surname, 1, 4)) AS username`
	`FROM`
	`  Viewers;`
		Query 5)e)19)	Return the usernames of the viewers which were created by concatenating the first 4 letters of a viewer's name and the first 4 letters of the viewer's surname.	As expected

```
SQL> SELECT
  2      viewer_id,
  3      v_name AS first_name,
  4      v_surname AS last_name,
  5      CONCAT(SUBSTR(v_name, 1, 4), SUBSTR(v_surname, 1, 4)) AS username
  6  FROM
  7      Viewers;

VIEWER_ID FIRST_NAME                    LAST_NAME                           USERNAME
--------- ----------------------------- ----------------------------------- ------------------------------
        1 DIMITRIS                      PAPADOPOULOS                        DIMIPAPA
        2 ELENI                         KONSTANTINIDOU                      ELENKONS
        3 GEORGIOS                      IOANNOU                             GEORIOAN
        4 MARIA                         NIKOLAOU                            MARINIKO
        5 VASILIS                       PAPANIKOLAOU                        VASIPAPA
        6 SOFIA                         GEORGIOU                            SOFIGEOR
        7 NIKOS                         KARAGIANNIS                         NIKOKARA
        8 KATERINA                      PAPAGEORGIOU                        KATEPAPA
        9 GIANNIS                       ANASTASIOU                          GIANANAS
       10 DESPINA                       ANTONIOU                            DESPANTO
       11 STAVROS                       AVGERIS                             STAVAVGE
       12 KONSTANTINA                   DIMITRIOU                           KONSDIMI
       13 CHRISTOS                      KARALIS                             CHRIKARA
       14 ATHINA                        PAPOUTSOGLOU                        ATHIPAPO
       15 PETROS                        PAPADIMITRIOU                       PETRPAPA

15 rows selected.
```

27	SELECT
	    s.title,
	    TO_CHAR(s.production_year, 'YYYY') AS production_year,
	    SUM(e.duration) AS total_series_length
	FROM
	    Series s
	JOIN
	    Series_Episodes se ON s.series_id = se.series_id
	JOIN
	    Episodes e ON se.episode_id = e.episode_id
	GROUP BY
	    TO_CHAR(s.production_year, 'YYYY'), s.title
	ORDER BY
	    TO_CHAR(s.production_year, 'YYYY');
		Query 5)e)20)	Return the total series length (total length of all episodes of each series) per year it was produced.	As expected

```
SQL> SELECT
  2      s.title,
  3      TO_CHAR(s.production_year, 'YYYY') AS production_year,
  4      SUM(e.duration) AS total_series_length
  5  FROM
  6      Series s
  7  JOIN
  8      Series_Episodes se ON s.series_id = se.series_id
  9  JOIN
 10      Episodes e ON se.episode_id = e.episode_id
 11  GROUP BY
 12      TO_CHAR(s.production_year, 'YYYY'), s.title
 13  ORDER BY
 14      TO_CHAR(s.production_year, 'YYYY');

TITLE                                              PROD TOTAL_SERIES_LENGTH
-------------------------------------------------- ---- ------------------
THE SIMPSONS                                       1989                1320
FRIENDS                                            1994                1320
THE OFFICE                                         2005                1320
THE BIG BANG THEORY                                2007                1320
BREAKING BAD                                       2008                3600
SHERLOCK                                           2010               10800
BLACK MIRROR                                       2011                3600
GAME OF THRONES                                    2011                3720
NARCOS                                             2015                2700
WESTWORLD                                          2016                3600
THE BOYS                                           2018                3600
THE MANDALORIAN                                    2019                1800
THE WITCHER                                        2019                7200

13 rows selected.
```

Part B

1. Could a star or snowflake schema be used for the database you created in Part A? What changes should you introduce? Provide an implementation.

Star Schema Implementation

A star schema would require a central fact table and several dimension tables.

- Fact Table: The Viewership table could be used as the central fact table, as it records the key measurable event (viewers watching episodes).
- Dimension Tables: The other tables (Series, Episodes and Viewers) would act as dimension tables.

Changes Needed:

- Modify Viewership Table: Include foreign keys to dimension tables and measurable metrics.
- Remove Actors Table: It doesn't contribute to the calculation of key measures and metrics included in the Fact Table.
- Simplify Dimension Tables: Remove redundant attributes and ensure they provide descriptive information about the facts.
- In star schema, Dimension tables are not connected to each other directly. Therefore, the Casting table can not exist since it is made up from foreign keys of other tables.

- Fact Table - Viewership:
    - Primary Key: ViewershipID
    - Foreign Keys: ActorID (references Actors), SeriesID (references Series), ViewerID (references Viewers), EpisodeID (references Episodes)
    - Metrics:
        - Series View Count: The number of times a series is viewed by all viewers.
        - Series Watch Time: Total time spent watching a series by all viewers.
        - Episode View Count: The number of times an episode is viewed by all viewers.
        - Episode Watch Time: Total time spent watching an episode by all viewers.
        - Series Unique Viewers: The number of unique viewers that have watched a particular series.
        - Episode Unique Viewers: The number of unique viewers that have watched a particular episode.
        - Average View Duration: The average amount of time spent watching an episode in a single session.

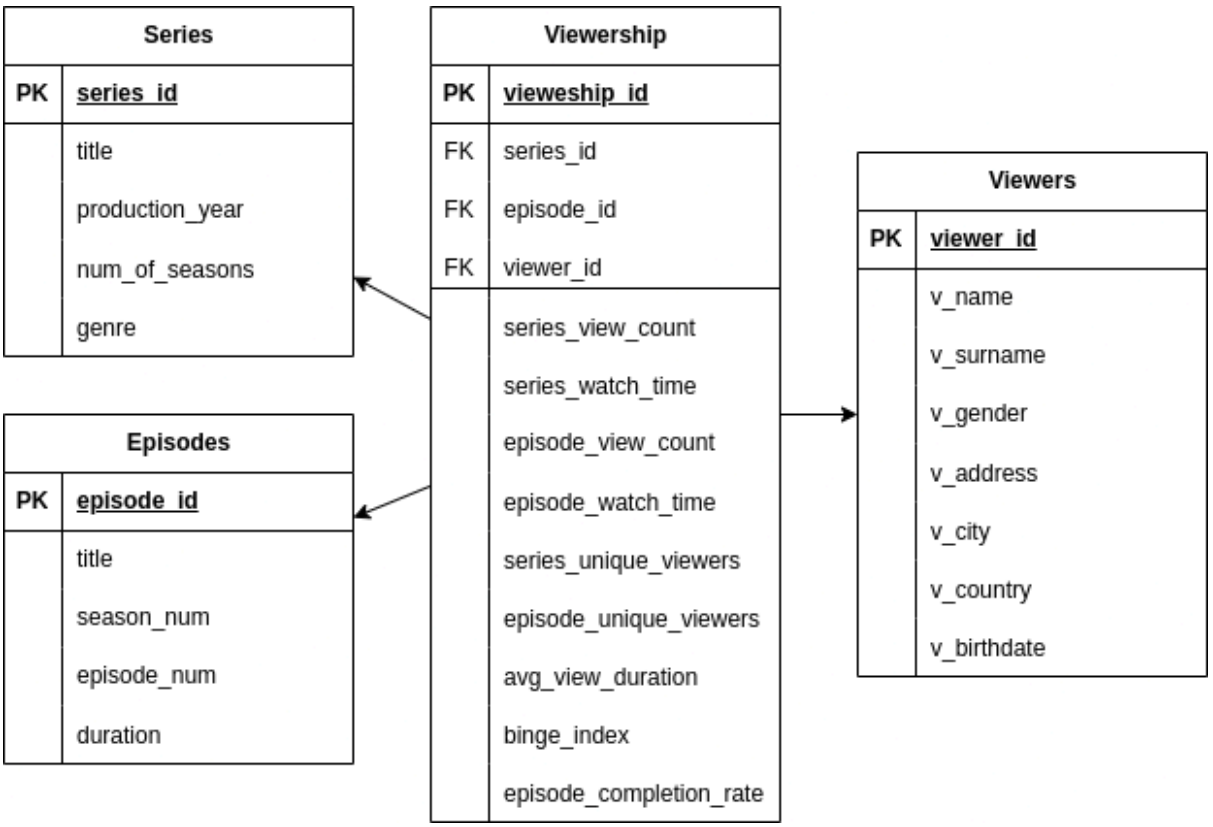
- Binge Index: A measure of how many episodes of the same series are watched in one sitting.
- Episode Completion Rate: Percentage of an episode watched on average (100% would mean the entire episode was watched).

Metrics that could be included if certain attributes were available:

- Peak View Time: The most common time of day or day of the week an episode is watched.

- Dimension Tables:
  - Series: SeriesID (PK), Title, Production Year, Number of Seasons, etc.
  - Episodes: EpisodeID (PK), Title, Season Number, SeriesID (FK), Duration, etc.
  - Viewers: ViewerID (PK), Name, Surname, Gender, etc.

In the star schema, each dimension table is directly linked to the fact table. The dimension tables are denormalized, meaning they might contain redundant data.



Snowflake Schema Implementation

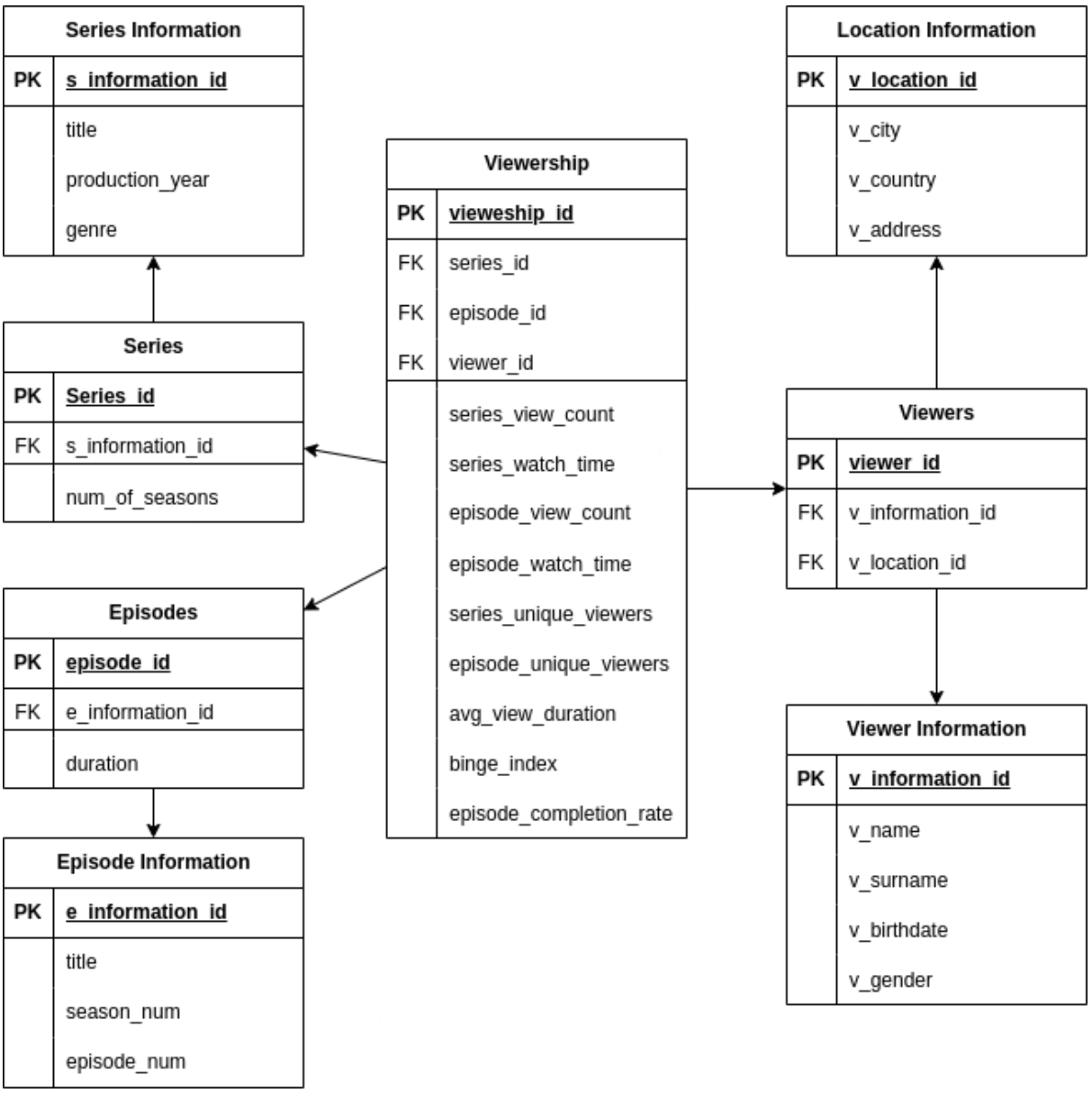
The snowflake schema is a more normalized version of the star schema. It would involve further decomposing the dimension tables.

Changes Needed:

- Similar to the star schema, but with additional steps to normalize the dimension tables.
- For instance the Viewers table could be divided into three separate tables.
    - The main one consists of each viewer's unique ID number and two foreign keys of the following normalized tables:
        - Viewer Information: each viewer's personal details and
        - Location Information: pertaining to their origin (country, city, address).
- Additionally, the Series table could be split into two tables. The main table consists of the unique ID, foreign key and number of seasons. The normalized table, called Series Information, contains the series title, production year and genre.
- Finally, the Episode table could be divided into two tables. The unique ID, foreign key, and duration are the only columns in the primary table. The title of the episode, season number, and episode number are listed in the normalized table named Episode Information.

- Fact Table - Viewership: Same as in the star schema.
- Dimension Tables:
    - Series: SeriesID (PK), SeriesInformationID (FK), NumberOfSeasons
    - Episodes: EpisodeID (PK), EpisodeInformationID (FK), Duration, EpisodeNumber
    - Viewers: ViewerID (PK), ViewerInformationID (FK), ViewerLocationID (FK)
- Additional Normalized Tables:
    - Series Information: SeriesInformationID (PK), Title, ProductionYear
    - Episode Information: EpisodeInformationID (PK), Title, SeasonNumber, EpisodeNumber
    - Viewer Information: ViewerInformationID (PK), ViewerName, ViewerSurname, ViewerGender
    - Location Information: ViewerLocationID (PK), Address, City, Country

In the above implementation of the snowflake schema the inclusion of the necessary attributes further reduces redundancy and improves computational efficiency. At the same time this can lead into more complex queries when trying to retrieve more specific information, due to the increased number of joins.


Series Information
PK
s_information_id
title
production_year
genre
Location Information
PK
v_location_id
v_city
v_country
v_address
Viewership
PK
vieweship_id
FK
series_id
FK
episode_id
FK
viewer_id
series_view_count
series_watch_time
episode_view_count
episode_watch_time
series_unique_viewers
episode_unique_viewers
avg_view_duration
binge_index
episode_completion_rate
Series
PK
Series_id
FK
s_information_id
num_of_seasons
Viewers
PK
viewer_id
FK
v_information_id
FK
v_location_id
Episodes
PK
episode_id
FK
e_information_id
duration
Viewer Information
PK
v_information_id
v_name
v_surname
v_birthdate
v_gender
Episode Information
PK
e_information_id
title
season_num
episode_num

Term Frequency - Inverse Document Frequency

2. Using a simple definition of Term Frequency (TF) as the number of occurrences of the term in a document, give the TF-IDF scores for the 10 most frequent terms in the following set of 2 documents:

A) “A star schema model can be depicted as a simple star: a central table contains fact data and multiple tables radiate out from it, connected by the primary and foreign keys of the database. In a star schema implementation, Warehouse Builder stores the dimension data in a single table or view for all the dimension levels. For example, if you implement the Product dimension using a star schema, Warehouse Builder uses a single table to implement all the levels in the dimension, as shown in the screenshot. The attributes in all the levels are mapped to different columns in a single table called PRODUCT”.

B) “The snowflake schema represents a dimensional model which is also composed of a central fact table and a set of constituent dimension tables which are further normalized into subdimension tables. In a snowflake schema implementation, Warehouse Builder uses more than one table or view to store the dimension data. Separate database tables or views store data pertaining to each level in the dimension. The screenshot displays the snowflake implementation of the Product dimension. Each level in the dimension is mapped to a different table”.

Websites for relevant material:

https://tfidf.com/
https://github.com/gearmonkey/tfidf-python/tree/master
https://code.google.com/archive/p/tfidf/
https://en.wikipedia.org/wiki/Tf%E2%80%93idf

Very common words (called stop words) such as “a”, “an”, “the”, “it” etc. are eliminated

TF x IDF Calculation

$$w_{ik} = tf_{ik} * \log(N / n_k)$$

T_k = term k in document D_i

tf_{ik} = frequency of term T_k in document D_i

idf_k = inverse document frequency of term T_k in C

N = total number of documents in the collection C

n_k = the number of documents in C that contain T_k

$$idf_k = \log\left(\frac{N}{n_k}\right)$$

T_k \ D_i	Document A	Document B	TF (raw count)	IDF $\log(N/n_k)$	TF*IDF
table	5	6	11	0	0
dimension	4	7	11	0	0
schema	3	2	5	0	0
levels	3	2	5	0	0
implement	3	2	5	0	0
data	2	2	4	0	0
star	4	0	4	0.301	1.204
warehouse	2	1	3	0	0
builder	2	1	3	0	0
single	3	0	3	0.301	0.903
snowflake	0	3	3	0.301	0.903
all	3	0	3	0.301	0.903
product	2	1	3	0	0
view	1	2	3	0	0
store	1	2	3	0	0

N = 2, because our corpus contains 2 documents in total

Similar words that counted as 1:

Table = [table, tables]
Dimension = [dimension, dimensional, subdimension]
Implement = [implement, implementation]
Level = [level, levels]
View = [view, views]
Store = [store, stores]