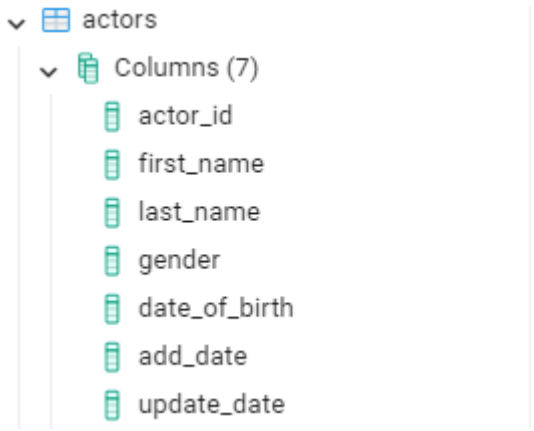
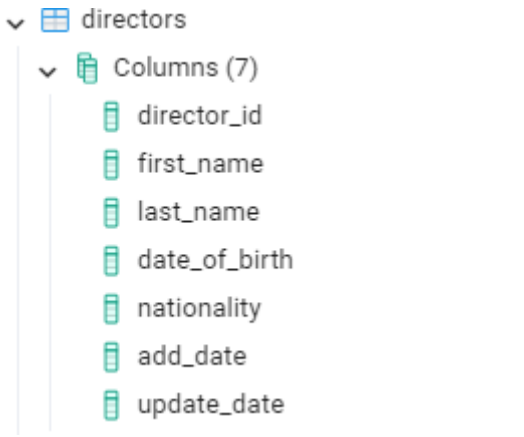
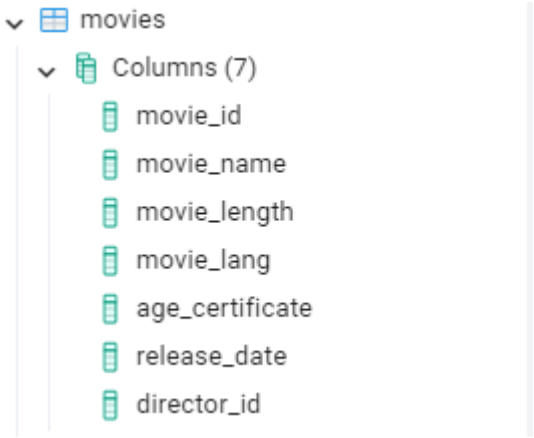
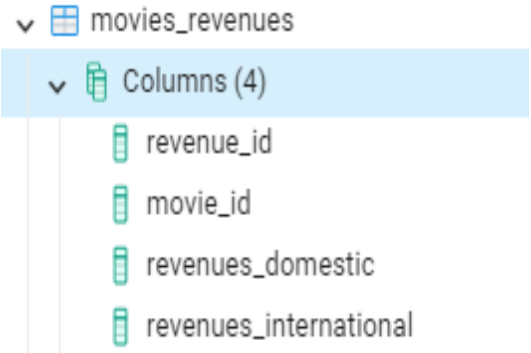
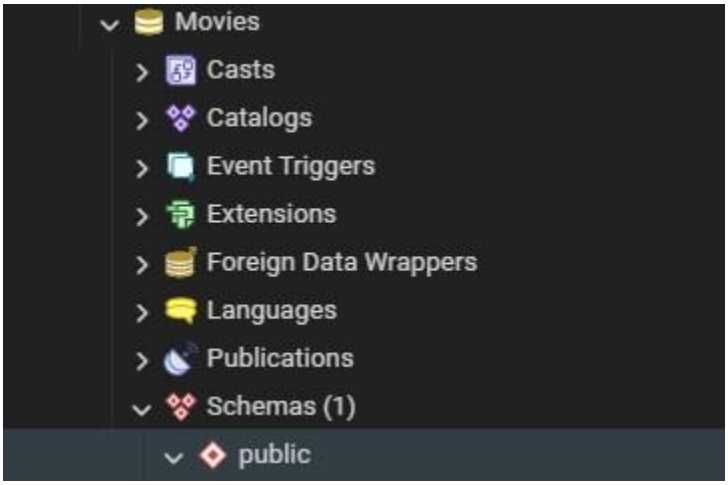

actors
Columns (7)
actor_id
first_name
last_name
gender
date_of_birth
add_date
update_date


directors
Columns (7)
director_id
first_name
last_name
date_of_birth
nationality
add_date
update_date


movies
Columns (7)
movie_id
movie_name
movie_length
movie_lang
age_certificate
release_date
director_id


movies_revenues
Columns (4)
revenue_id
movie_id
revenues_domestic
revenues_international

1) So, your first step is to go ahead and create this database called “movies”. As you can see, a sample screenshot from the output is given below. After creating the database, please attach your own output screenshot .



2) Write a SQL query to create the “**actors**” table as well as the related 7 variables. The features of these variables (that should be considered while creating these variables) are provided below. As can be seen, a sample screenshot is given (below) from the “actors” table. (Note that, I have provided couple of rows for you to see what this table will eventually look like. However, you should not insert any data values at this step. You should only create the table with the following variables and that is it.

- **actor_id** has the “serial” type. Note that this column is the primary key of the table.
- **first_ name** and the **last_name** are strings that have the “limited varying character (150 characters)” type.

- Note that the **last_name** cannot take "null" values (constraint)
- **gender** is string that have the "fixed length character (1character)" type.
- **Date_of_birth, add_date** and the **update_date** are date columns that have the "date" type.

	actor_id [PK] integer	first_name character varying (150)	last_name character varying (150)	gender character	date_of_birth date	add_date date	update_date date
1	1	Malin	Akerman	F	1978-05-12	[null]	[null]
2	2	Tim	Allen	M	1953-06-13	[null]	[null]
3	3	Julie	Andrews	F	1935-10-01	[null]	[null]
4	4	Ivana	Baquero	F	1994-06-11	[null]	[null]

```
CREATE TABLE actors (
actor_id serial primary key,
first_name varchar(150),
last_name varchar(150) not null,
gender char(1),
date_of_birth date,
add_date date,
update_date date
);
```

3) Write a SQL query to create the "**directors**" table as well as the related 7 variables. The features of these variables (that should be considered while creating these variables) are provided below. As can be seen, a sample screenshot is given (below) from the "directors" table. (Note that, I have provided couple of rows for you to see what this table will eventually look like. However, you should not insert any data values at this step.

- **director_id** has the "serial" type. Note that this column is the primary key of the table.
- **first_name** and the **last_name** are strings that have the "limited varying character (150 characters)" type.
- **nationality** is string that has the "limited varying character (20 characters)" type.
- **Date_of_birth, add_date** and the **update_date** are date columns that have the "date" type.

	director_id [PK] integer	first_name character varying (150)	last_name character varying (150)	nationality character varying (20)	date_of_birth date	add_date date	update_date date
1	1	Tomas	Alfredson	Swedish	1965-04-01	[null]	[null]
2	2	Paul	Anderson	American	1970-06-26	[null]	[null]
3	3	Wes	Anderson	American	1969-05-01	[null]	[null]
4	4	Richard	Ayoade	British	1977-06-12	[null]	[null]

```
CREATE TABLE directors (
director_id serial primary key,
first_name varchar(150),
last_name varchar(150),
nationality varchar(20),
date_of_birth date,
add_date date,
update_date date
);
```

4) Write a SQL query to create the "**movies**" table as well as the related 7 variables. The features of these variables (that should be considered while creating these variables) are provided below. As can be seen, a sample screenshot is given (below) from the "movies" table. (Note that, I have provided couple of rows for you to see what this table will eventually look like. However, you should not insert any data values at this step.

- **movie_id** has the "serial" type. Note that this column is the primary key of the table.
- **movie_name** is string that has the "limited varying character (150 characters)" type.
- Note that the **movie_name** cannot take "null" values (constraint)
- **Movie_length** has the "integer" type.
- **movie_lang** is string that has the "limited varying character (20 characters)" type.
- **age_certificate** is string that has the "limited varying character (10 characters)" type.
- **release_date** is a date column that have the "date" type.
- **director_id** has the "integer" type.
- **director_id** is a foreign key that should be referenced to the director_id from the directors table.

	movie_id [PK] integer	movie_name character varying (150)	movie_length integer	movie_lang character varying (20)	age_certificate character varying (10)	release_date date	director_id integer
1	1	A Clockwork Orange	112	English	18	1972-02-02	13
2	2	Apocalypse Now	168	English	15	1979-08-15	9
3	3	Battle Royale	111	Japanese	18	2001-01-04	10
4	4	Blade Runner	121	English	15	1982-06-25	27

```
CREATE TABLE movies (
movie_id serial primary key,
movie_name varchar(150) not null,
movie_length int,
movie_lang varchar(20),
age_certificate varchar(10),
release_date date,
director_id int,
foreign key(director_id) references directors(director_id)
);
```

5) Write a SQL query to create a table called "**movies_revenues**" table as well as the related 4 variables. The features of these variables (that should be considered while creating these variables) are provided below. As can be seen, a sample screenshot is given (below) from the "**movies_revenues**" table. (Note that, I have provided couple of rows for you to see what this table will eventually look like. However, you should not insert any data values at this step.

- **revenue_id** has the "serial" type. Note that this column is the primary key of the table.
- **movie_id** has the "integer" type
- **movie_id** is a foreign key that references to the movie_id from the movies table.
- **Revenues_domestic** and the **revenues_international** are numeric columns that have the "numeric" type that have precision of "10", and the scale of "2"

	revenue_id [PK] integer	movie_id integer	revenues_domestic numeric (10,2)	revenues_international numeric (10,2)
1	1	45	22.20	1.30
2	2	13	199.40	201.20
3	3	23	102.10	[null]
4	4	44	158.70	[null]

```
CREATE TABLE movies_revenue (
revenue_id serial primary key,
movie_id int,
revenue_domestic numeric(10,2),
revenue_international numeric(10,2),
foreign key(movie_id) references movies (movie_id)
);
```

6) Go ahead and open the “directors.sql” file in your PgAdmin environment and insert the values given in that file into “Directors” table. Then go ahead and make sure that your data is inserted by running;

SELECT * FROM directors
LIMIT 4;

and provide the screenshot of your output below.

	director_id [PK] integer	first_name character varying (150)	last_name character varying (150)	nationality character varying (20)	date_of_birth date	add_date date	update_date date
1	1	Tomas	Alfredson	Swedish	1965-04-01	[null]	[null]
2	2	Paul	Anderson	American	1970-06-26	[null]	[null]
3	3	Wes	Anderson	American	1969-05-01	[null]	[null]
4	4	Richard	Ayoade	British	1977-06-12	[null]	[null]

Total rows: 4 of 4 Query complete 00:00:00.128 Ln 43, Col 1

7) Do the same thing for “actors” table.

	actor_id [PK] integer	first_name character varying (150)	last_name character varying (150)	gender character (1)	date_of_birth date	add_date date	update_date date
1	1	Malin	Akerman	F	1978-05-12	[null]	[null]
2	2	Tim	Allen	M	1953-06-13	[null]	[null]
3	3	Julie	Andrews	F	1935-10-01	[null]	[null]
4	4	Ivana	Baquero	F	1994-06-11	[null]	[null]

Total rows: 4 of 4 Query complete 00:00:00.207 Ln 143, Col 38

8) Do the same thing for “movies” table.

	movie_id [PK] integer	movie_name character varying (150)	movie_length integer	movie_lang character varying (20)	age_certificate character varying (10)	release_date date	director_id integer
1	1	A Clockwork Orange	112	English	18	1972-02-02	13
2	2	Apocalypse Now	168	English	15	1979-08-15	9
3	3	Battle Royale	111	Japanese	18	2001-01-04	10
4	4	Blade Runner	121	English	15	1982-06-25	27

Total rows: 4 of 4 Query complete 00:00:00.121 Ln 57, Col 9

9) Do the same thing for “movies_revenues” table.

	revenue_id [PK] integer	movie_id integer	revenue_domestic numeric (10,2)	revenue_international numeric (10,2)
1	1	45	22.20	1.30
2	2	13	199.40	201.20
3	3	23	102.10	[null]
4	4	44	158.70	[null]

Total rows: 4 of 4 Query complete 00:00:00.201 Ln 58, Col 9

10) By using inner join, retrieve movie_id, movie_name, movie_lang, release_date, revenues_domestic and revenues_international. Save the output of this question as “movies_total” table.

Use movies_total table for the questions 11, 12, 13 and 14.

```
CREATE TABLE movies_total AS

SELECT movie_id, movie_name, movie_lang,release_date, revenue_domestic, revenue_international

from movies

inner join movies_revenue

using(movie_id);
```

11) Find the total domestic revenue amount for movies with “English” language.

```
select sum(revenue_domestic) as total_revenue_domestic
from movies_total
where movie_lang = 'English';
```

12) Replace the NULL values in the revenues_domestic column with **30**.

```
update movies_total
set revenue_domestic = 30
where revenue_domestic is null;
```

13) Add a NOT NULL constraint for revenues_domestic column.

```
alter table movies_total
alter column revenue_domestic set not null;
```

14) Add a DEFAULT value of **50 for** revenues_international column and show that default value is working by inserting a record to the movies_total table.

alter table movies_total

alter column revenue_international set default 50;

INSERT INTO movies_total (movie_id,

movie_name,movie_lang,release_date,revenue_domestic) VALUES

(67,'Leo', 'Tamil', '2023-10-23', 100);

select * from movies_total

where movie_id = 67;

```
80  alter table movies_total
81  alter column revenue_international set default 50;
82
83
84  INSERT INTO movies_total (movie_id, movie_name,movie_lang,release_date,revenue_domestic) VALUES
85  (67,'Leo', 'Tamil', '2023-10-23', 100);
86
87  select * from movies_total
88  where movie_id = 67;
```

Data Output | Messages | Notifications

SQL

	movie_id integer	movie_name character varying (150)	movie_lang character varying (20)	release_date date	revenue_domestic numeric (10,2)	revenue_international numeric (10,2)
1	67	Leo	Tamil	2023-10-23	100.00	50.00

15) Attempt to update the **director_id** from 27 to 44 in the **directors** table. Describe the type of output you receive upon trying to update the **director_id** in the **directors** table. What should have been done to avoid the error encountered? (Note: You don't have to recreate the **directors** table. Just explain the solution that allows us to update the **director_id** in the **directors** table.)

```
91
92  update directors
93  set director_id = 44
94  where director_id = 27;
```

Data Output | Messages | Notifications

```
ERROR:  Key (director_id)=(27) is still referenced from table "movies".update or delete on table "directors" violates foreign key constraint "movies_director_id_fkey" on table "movies"

ERROR:  update or delete on table "directors" violates foreign key constraint "movies_director_id_fkey" on table "movies"
SQL state: 23503
Detail: Key (director_id)=(27) is still referenced from table "movies".
```

We got this error because director_id in directors table is referenced as a foreign key in movies table.

This update violates foreign key constraint on movies table.

To solve this issue we have to define the foreign key with ‘On update cascade’ constraint. This constraint will automatically update the values in the foreign key of the child table when the primary key of the parent table is updated.