



DATABASE MANAGEMENT SYSTEM

FilmX

**COMPUTER ENGINEERING
DEPARTMENT**

2023-2024 FALL

**YAVUZ SELİM ERDOĞAN –
200315073**

BURAK TALHA MEMİŞ –200315071

İREM DİLŞAT KÖSE – 200315029

SOURCE CODE

1-CREATING TABLES

1.1 Film

Our movie table contains information about the film_id, title, imdb, length, a brief description, year, box office money and the category it is in. Data types are VARCHAR, TEXT for those whose length we do not know, INTEGER and REAL for numbers, and PRIMARY KEY s we used SERIAL, so as a new movie is added, the ID value will automatically increase. We made movie id a PRIMARY KEY because every movie must have a unique id value.

```
CREATE TABLE film (  
    film_id SERIAL PRIMARY KEY,  
    title VARCHAR(50) NOT NULL,  
    imdb REAL NOT NULL,  
    length INTEGER NOT NULL,  
    description TEXT NOT NULL,  
    year INTEGER NOT NULL,  
    box_office INTEGER NOT NULL,  
    category_type VARCHAR(50) NOT NULL  
);
```

Data Output

Messages

Notifications

	film_id	title	imdb	length	description	year	box_office	category_type
	[PK] integer	character varying (50)	real	integer	text	integer	integer	character varying (50)

1.2 Actor

Our actor table contains the actor_id, the actor_name, actor_salary, and the information about the movie the actor played. We took the movie the actor played as a reference from the movie table, so the film_id here is a foreign one. We made actor_id a PRIMARY KEY because every actor must have a unique id value.

```
CREATE TABLE actor (  
  actor_id SERIAL PRIMARY KEY,  
  actor_name VARCHAR(50) NOT NULL,  
  actor_salary INTEGER NOT NULL,  
  film_id INTEGER REFERENCES film(film_id)  
);
```

Data Output

Messages

Notifications

actor_id

[PK] integer

actor_name

character varying (50)

actor_salary

integer

film_id

integer

1.3 Director

Our director table contains information about the director_id, director_name, director_gender, director_salary and which movie directed. To find the movie directed, we took a reference from the film table and used film_id as the foreign key. Since there should be a separate id value for each director, director_id became the PRIMARY KEY.

```
CREATE TABLE director (  
    director_id SERIAL PRIMARY KEY,  
    director_name VARCHAR(50) NOT NULL,  
    director_gender VARCHAR(50) NOT NULL,  
    director_salary INTEGER NOT NULL,  
    film_id INTEGER REFERENCES film(film_id)  
);
```


Data Output


Messages


Notifications


≡


+






















director_id

[PK] integer




director_name

character varying (50)




director_gender

character varying (50)




director_salary

integer



film_id














integer



1.4 Country

Our country table contains country_id, country_name, language, and film_id information of that country. Country names are unique, but there is a possibility that the two movies were shot in the same country, so we made country_id the PRIMARY KEY for the country table.










```
CREATE TABLE country (  
    country_id SERIAL PRIMARY KEY,  
    country_name VARCHAR(50) NOT NULL,  
    language VARCHAR(50) NOT NULL,  
    film_id INTEGER REFERENCES film(film_id)  
);
```

Data Output						Messages	Notifications
							
	director_id [PK] integer 	director_name character varying (50) 	director_gender character varying (50) 	director_salary integer 	film_id integer 		

1.5 Users

In our user table, we received information such as the user_id ,user_ name, user_age, email, and password. We made user_id a PRIMARY KEY because every user must have a unique id value.

```
CREATE TABLE users(  
    user_id SERIAL PRIMARY KEY,  
    user_name VARCHAR(50) NOT NULL,  
    user_age INTEGER NOT NULL,  
    email VARCHAR(50) NOT NULL,  
    password VARCHAR(50) NOT NULL  
);|
```

Data Output Messages Notifications					
<div></div>					
user_id	user_name	user_age	email	password	
[PK] integer	character varying (50)	integer	character varying (50)	character varying (50)	

1.6 Payment

Our payment table contains the payment_id, user_id , amount, payment_date and payment_method. Since each payment must have a different id value, we made the payment_id the PRIMARY KEY. To make the payment table more meaningful, we took the user data from the user table, so user_id became a foreign key here.

```
CREATE TABLE payment(  
    payment_id SERIAL PRIMARY KEY,  
    user_id INTEGER REFERENCES users(user_id),  
    amount INTEGER NOT NULL,  
    payment_date DATE NOT NULL,  
    payment_method VARCHAR(50) NOT NULL  
);
```

Data Output

Messages

Notifications

payment_id

[PK] integer

user_id

integer

amount

integer

payment_date

date
















payment_method

character varying (50)

1.7 Favourite

In our favorite table, there is the id of each favorite, our favorite_category, and a foreign key such as user_id, film_id, d, director_id, actor_id. In this way, we can choose many favorite topics.

```
CREATE TABLE favourite(  
    favourite_id SERIAL PRIMARY KEY,  
    favourite_category VARCHAR(50) NOT NULL,  
    user_id INTEGER REFERENCES users(user_id),  
    film_id INTEGER REFERENCES film(film_id),  
    director_id INTEGER REFERENCES director(director_id),  
    actor_id INTEGER REFERENCES actor(actor_id)  
);
```

Data Output							Messages	Notifications
								
	favourite_id [PK] integer 	favourite_category character varying (50) 	user_id integer 	film_id integer 	director_id integer 	actor_id integer 		

1.8 Watched

In our watched table, film_id, user_id, a short viewer comment and the last change date. Here, we also took the watched_id value as the PRIMARY KEY because you can watch a movie more than once, which causes repetition.


```
CREATE TABLE watched (  
    watched_id SERIAL PRIMARY KEY,  
    film_id INTEGER REFERENCES film(film_id),  
    user_id INTEGER REFERENCES users(user_id),  
    comment TEXT NOT NULL,  
    last_update DATE NOT NULL  
);|
```

Data Output


Messages

Notifications


≡+





▼









▼









	<div><div>watched_id</div><div>[PK] integer</div><div></div></div>	<div><div>film_id</div><div>integer</div><div></div></div>	<div><div>user_id</div><div>integer</div><div></div></div>	<div><div>comment</div><div>text</div><div></div></div>	<div><div>last_update</div><div>date</div><div></div></div>
--	---	---	---	--	--

2. INSERT DATA

2.1 Film

We added the relevant data with the INSERT INTO method .Since we have assigned the PRIMARY KEY AS SERIAL, we do not need to add the film_id here, so the id values will be added automatically.

Since it does not fit in pgAdmin, its code was written in VS code.

```
INSERT INTO film (title,imdb,length,description,year,box_office,category_type)
VALUES
    ('Fight Club', 8.8, 139, 'An insomniac office worker and a devil-may-care soap maker form an underground fight club that evolves into much more.', 1999,100853753, 'drama, thriller'),
    ('Intime',6.7,109,'In a world where people stop aging at the age of 25 and time is more valuable than money. Will Salas, accused of murder, escapes with a hostage and forms an important alliance against the oppressive system.',2011, 173900000,'science fiction, action'),
    ('PK', 8.1, 153 , 'The film follows the extraterrestrial being PK as he navigates humorous and thought-provoking events while attempting to understand human cultures, questioning aspects like religion, language, and society. ', 2014, 115000000,'science fiction, comedy'),
    ('Inception', 8.8, 148 , 'A thief who steals corporate secrets through the use of dream-sharing technology is given the inverse task of planting an idea into the mind of a C.E.O., but his tragic past may doom the project and his team to disaster.', 2010, 836800000 , 'science fiction'),
    ('Scarface', 8.3, 170 , 'In 1980 Miami, a determined Cuban immigrant takes over a drug cartel and succumbs to greed.', 1983,65100000, 'crime, drama'),
```

```
('Joker', 8.4, 122 , 'The film depicts the transformation of Arthur Fleck, an outcast from society, into the crime genius known as Joker in the city of Gotham, showcasing his descent into madness.', 2019,1074000000,'psychological tension'),
('Interstellar', 8.6, 169 , 'The film follows a group of astronauts on a space journey in search of a new habitat for humanity after a climate catastrophe on Earth.', 2014, 773500000 , 'science fiction'),
('Forrest Gump', 8.8, 142 , 'The film depicts the life of Forrest Gump, a man with low IQ who witnesses extraordinary events and significantly influences American history by participating in various important events.', 1994, 677386686,'comedy, drama'),
('Lucy', 6.4, 89 , 'The film follows the struggle of a young woman named Lucy, who develops extraordinary mental and physical abilities after an incident involving drug trafficking.', 2014, 434057600,'science fiction'),
('The Losers Club', 8.1, 130 , 'The film tells the story of Zafer, who leads a mundane life, and his establishment of "The Losers Club" as a means to cope with depression.', 2011,5000000, 'comedy, drama');
```

Data Output Messages Notifications					
	film_id [PK] integer	title character varying (50)	imdb real	length integer	description text
1	1	Fight Club	8.8	139	An insomniac office worker and a devil-may-care soap maker form an underground fight club that evolves into much more.
2	2	Intime	6.7	109	In a world where people stop aging at the age of 25 and time is more valuable than money. Will Salas, accused of murder, escapes with a h
3	3	PK	8.1	153	The film follows the extraterrestrial being PK as he navigates humorous and thought-provoking events while attempting to understand hun
4	4	Inception	8.8	148	A thief who steals corporate secrets through the use of dream-sharing technology is given the inverse task of planting an idea into the min
5	5	Scarface	8.3	170	In 1980 Miami, a determined Cuban immigrant takes over a drug cartel and succumbs to greed.
6	6	Joker	8.4	122	The film depicts the transformation of Arthur Fleck, an outcast from society, into the crime genius known as Joker in the city of Gotham, sl
7	7	Interstellar	8.6	169	The film follows a group of astronauts on a space journey in search of a new habitat for humanity after a climate catastrophe on Earth.
8	8	Forrest Gump	8.8	142	The film depicts the life of Forrest Gump, a man with low IQ who witnesses extraordinary events and significantly influences American hist
9	9	Lucy	6.4	89	The film follows the struggle of a young woman named Lucy, who develops extraordinary mental and physical abilities after an incident inv
10	10	The Losers Club	8.1	130	The film tells the story of Zafer, who leads a mundane life, and his establishment of "The Losers Club" as a means to cope with depression

year integer	box_office integer	category_type character varying (50)
1999	100853753	drama, thriller
2011	173900000	science fiction, action
2014	115000000	science fiction, comedy
2010	836800000	science fiction
1983	65100000	crime, drama
2019	1074000000	psychological tension
2014	773500000	science fiction
1994	677386686	comedy, drama
2014	434057600	science fiction
2011	5000000	comedy, drama

2.2 Actor

We added relevant data. Since we have assigned the PRIMARY KEY AS SERIAL, we do not need to add the user_id here, so the id values will be added automatically.

```
INSERT INTO actor (actor_name,actor_salary,film_id)
VALUES
```

```
    ('Brad Pitt', 20000000,1),
    ('Edward Norton', 15000000,1),
    ('Meat Loaf', 10000000,1),
    ('Justin Timberlake', 9000000,2),
    ('Amanda Seyfried', 5000000,2),
    ('Cillian Murphy', 5000000,2),
    ('Aamir Khan', 10000000,3),
    ('Anushka Sharma', 8000000,3),
    ('Sanjay Dutt', 4000000,3),
    ('Leonardo DiCaprio',20000000,4),
    ('Joseph Gordan-Levitt', 10000000,4),
    ('Elliot Page', 10000000,4),
    ('Al Pacino', 10000000,5),
    ('Michelle Pfeiffer', 5000000,5),
    ('Steven Bauer', 3000000,5),
    ('Joaquin Phoenix', 4500000,6),
```

```
    ('Robert De Niro', 3000000,6),
    ('Zazie Beetz', 1000000,6),
    ('Matthew McConaughey', 9000000 ,7),
    ('Anne Hathaway', 8000000,7),
    ('Jessica Chastain', 6000000,7),
    ('Tom Hanks', 40000000,8),
    ('Robin Wright', 5000000,8),
    ('Gary Sinise', 3000000,8),
    ('Scarlet Johansson', 8000000,9),
    ('Morgan Freeman', 7000000,9),
    ('Choi Min-sik', 2000000,9),
    ('Nejat İşler', 10000000,10),
    ('Yiğit Özşener', 10000000,10),
    ('Ahu Türkpençe', 6000000,10);
```

Data Output Messages Notifications					
	actor_id [PK] integer	actor_name character varying (50)	actor_salary integer	film_id integer	
1	1	Brad Pitt	20000000	1	
2	2	Edward Norton	15000000	1	
3	3	Meat Loaf	10000000	1	
4	4	Justin Timberlake	9000000	2	
5	5	Amanda Seyfried	5000000	2	
6	6	Cillian Murphy	5000000	2	
7	7	Aamir Khan	10000000	3	
8	8	Anushka Sharma	8000000	3	
9	9	Sanjay Dutt	4000000	3	
10	10	Leonardo DiCaprio	20000000	4	
11	11	Joseph Gordan-Levitt	10000000	4	
12	12	Elliot Page	10000000	4	
13	13	Al Pacino	10000000	5	
14	14	Michelle Pfeiffer	5000000	5	
15	15	Steven Bauer	3000000	5	

16	16	Joaquin Phoenix	4500000	6	
17	17	Robert De Niro	3000000	6	
18	18	Zazie Beetz	1000000	6	
19	19	Matthew McConaughey	9000000	7	
20	20	Anne Hathaway	8000000	7	
21	21	Jessica Chastain	6000000	7	
22	22	Tom Hanks	40000000	8	
23	23	Robin Wright	5000000	8	
24	24	Gary Sinise	3000000	8	
25	25	Scarlet Johansson	8000000	9	
26	26	Morgan Freeman	7000000	9	
27	27	Choi Min-sik	2000000	9	
28	28	Nejat İşler	10000000	10	
29	29	Yiğit Özşener	10000000	10	
30	30	Ahu Türkpençe	6000000	10	

2.3 Director

We added the relevant data with the INSERT INTO method. Since we have assigned the PRIMARY KEY AS SERIAL, we do not need to add director_id here, so id values will be added automatically. We added the film_id value that matches the director of the movie.

```
INSERT INTO director (director_name,director_gender,director_salary,film_id)
VALUES
    ('David Fincher', 'male', 8000000,1),
    ('Andrew Niccol', 'male', 4000000,2),
    ('Rajkumar Hirani', 'male', 5000000,3),
    ('Cristopher Nolan', 'male', 15000000,4),
    ('Brian De Palma', 'male', 9000000,5),
    ('Todd Phillips', 'male', 2000000,6),
    ('Cristopher Nolan', 'male', 4000000,7),
    ('Robert Zemeckis', 'male', 15000000,8),
    ('Luc Besson', 'male', 6000000,9),
    ('Tolga Örnek', 'male', 3000000,10);
```

Data Output Messages Notifications						
	director_id [PK] integer	director_name character varying (50)	director_gender character varying (50)	director_salary integer	film_id integer	
1	1	David Fincher	male	8000000	1	
2	2	Andrew Niccol	male	4000000	2	
3	3	Rajkumar Hirani	male	5000000	3	
4	4	Cristopher Nolan	male	15000000	4	
5	5	Brian De Palma	male	9000000	5	
6	6	Todd Phillips	male	2000000	6	
7	7	Cristopher Nolan	male	4000000	7	
8	8	Robert Zemeckis	male	15000000	8	
9	9	Luc Besson	male	6000000	9	
10	10	Tolga Örnek	male	3000000	10	

2.4 Country

We added the relevant data with the INSERT INTO method. Since we have assigned the PRIMARY KEY AS SERIAL, we do not need to add country_id here, so id values will be added automatically. We added the film_id value that matches the country of the movie.

```
INSERT INTO country (country_name,language,film_id)
VALUES
    ('USA', 'English',1),
    ('USA', 'English',2),
    ('India', 'Hindi',3),
    ('USA', 'English',4),
    ('USA', 'English',5),
    ('USA', 'English',6),
    ('USA', 'English',7),
    ('USA', 'English',8),
    ('France', 'English',9),
    ('Turkey', 'Turkish',10);
```

Data Output Messages Notifications				
	country_id [PK] integer	country_name character varying (50)	language character varying (50)	film_id integer
1	1	USA	English	1
2	2	USA	English	2
3	3	India	Hindi	3
4	4	USA	English	4
5	5	USA	English	5
6	6	USA	English	6
7	7	USA	English	7
8	8	USA	English	8
9	9	France	English	9
10	10	Turkey	Turkish	10

2.5 Users

We added the relevant data with the INSERT INTO method. Since we have assigned the PRIMARY KEY AS SERIAL, we do not need to add country_id here, so id values will be added automatically. We added the film_id value that matches the country of the movie.

```
INSERT INTO users (user_name,user_age,email,password)
VALUES
('Feyza Şahan',22,'Feyzasahan@gmail.com','mugla1509'),
('Deniz Kaya',34,'denizKaya06@outlook.com','2343-deniz-2343'),
('Burak Aydın',56,'kralBurak1881@gmail.com','123kralkral123'),
('Emre Demir',19,'arabasevdasi@gmail.com','fordfocus'),
('Caner Karahan',17,'karahanli_polat@hotmail.com','12345678.CK'),
('Gizem Çelik',25,'gizemgizem@gmail.com','gizem123456'),
('Ayşe Aktaş',16,'aysesincan@gmail.com','yoldayuruyenkedi'),
('Melis Tekin',36,'34melis34@hotmail.com','tarantula61'),
('İrem Arslan',17,'izmirliirem@outlook.com','komaistas123'),
('Ömer Polat',23,'vatozpolat@gmail.com','vatozbaligi');
```

Data Output Messages Notifications					
	user_id [PK] integer	user_name character varying (50)	user_age integer	email character varying (50)	password character varying (50)
1	1	Feyza Şahan	22	Feyzasahan@gmail.com	mugla1509
2	2	Deniz Kaya	34	denizKaya06@outlook.com	2343-deniz-2343
3	3	Burak Aydın	56	kralBurak1881@gmail.com	123kralkral123
4	4	Emre Demir	19	arabasevdasi@gmail.com	fordfocus
5	5	Caner Karahan	17	karahanli_polat@hotmail.com	12345678.CK
6	6	Gizem Çelik	25	gizemgizem@gmail.com	gizem123456
7	7	Ayşe Aktaş	16	aysesincan@gmail.com	yoldayuruyenkedi
8	8	Melis Tekin	36	34melis34@hotmail.com	tarantula61
9	9	İrem Arslan	17	izmirliirem@outlook.com	komaistas123
10	10	Ömer Polat	23	vatozpolat@gmail.com	vatozbaligi

2.6 Payment

We added the relevant data with the INSERT INTO method. Since we have assigned the PRIMARY KEY AS SERIAL, we do not need to add payment_id here, so id values will be added automatically. We added the user_id value that matches the payment of the user.

```
INSERT INTO payment(user_id,amount,payment_date,payment_method)
VALUES
(1,30,'01.01.2024','Visa - 1234 5678 9012 3456'),
(2,30,'10.12.2023','Visa - 7234 4124 0574 9401'),
(3,30,'02.01.2024','MasterCard - 9463 9832 1283 6372'),
(4,30,'26.12.2023','Visa - 4210 9876 5432 1098'),
(5,20,'13.12.2023','MasterCard - 011 1122 3344 5566'),
(6,30,'10.01.2024','Visa - 4912 3456 7890 1234'),
(7,20,'26.12.2023','Visa - 6333 1122 3344 5566'),
(8,30,'06.01.2024','MasterCard - 1800 8765 4321 9876'),
(9,20,'10.01.2024','MasterCard - 3802 3456 7890 5678'),
(10,30,'12.12.2023','Visa - 4716 1234 5678 9012');
```

Data Output Messages Notifications						
	payment_id [PK] integer	user_id integer	amount integer	payment_date date	payment_method character varying (50)	
1	1	1	30	2024-01-01	Visa - 1234 5678 9012 3456	
2	2	2	30	2023-12-10	Visa - 7234 4124 0574 9401	
3	3	3	30	2024-01-02	MasterCard - 9463 9832 1283 6372	
4	4	4	30	2023-12-26	Visa - 4210 9876 5432 1098	
5	5	5	20	2023-12-13	MasterCard - 011 1122 3344 5566	
6	6	6	30	2024-01-10	Visa - 4912 3456 7890 1234	
7	7	7	20	2023-12-26	Visa - 6333 1122 3344 5566	
8	8	8	30	2024-01-06	MasterCard - 1800 8765 4321 9876	
9	9	9	20	2024-01-10	MasterCard - 3802 3456 7890 5678	
10	10	10	30	2023-12-12	Visa - 4716 1234 5678 9012	

2.7 Favourite

We added the relevant data with the INSERT INTO method. Since we have assigned the PRIMARY KEY AS SERIAL, we do not need to add favourite_id here, so id values will be added automatically. We added the user_id, film_id, director_id, actor_id values.

```
INSERT INTO favourite(favourite_category,user_id,film_id,director_id,actor_id)
VALUES
('comedy',1,3,7,14),
('science fiction',2,4,7,2),
('comedy',3,7,7,11),
('psychological tension',4,1,6,8),
('action',5,10,2,21),
('drama',6,3,4,12),
('comedy',7,2,4,5),
('science fiction',8,9,9,17),
('drama',9,3,1,19),
('psychological tension',10,6,5,1);
```

Data Output Messages Notifications							
	favourite_id [PK] integer	favourite_category character varying (50)	user_id integer	film_id integer	director_id integer	actor_id integer	
1	1	comedy	1	3	7	14	
2	2	science fiction	2	4	7	2	
3	3	comedy	3	7	7	11	
4	4	psychological tension	4	1	6	8	
5	5	action	5	10	2	21	
6	6	drama	6	3	4	12	
7	7	comedy	7	2	4	5	
8	8	science fiction	8	9	9	17	
9	9	drama	9	3	1	19	
10	10	psychological tension	10	6	5	1	

2.8 Watched

We added the relevant data with the INSERT INTO method. Since we have assigned the PRIMARY KEY AS SERIAL, we do not need to add watched_id here, so id values will be added automatically.

```
INSERT INTO watched (film_id,user_id,comment,last_update)
```

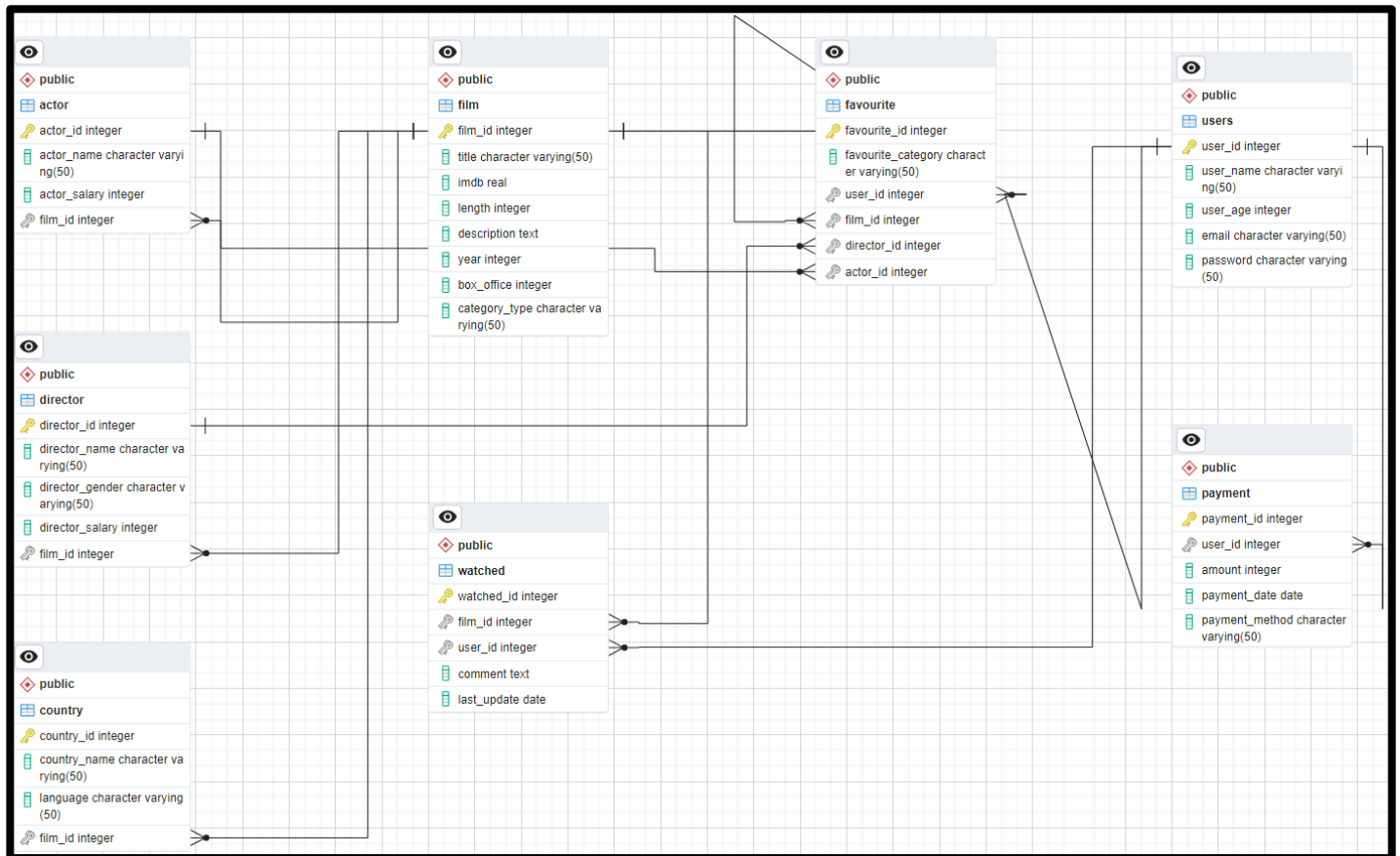
```
VALUES
```

```
(1,1,'It was a cleverly woven psychological thriller. It seems like it takes the audience on an emotional jou
(2,2,'An impressive dystopian film about a struggle against time. A striking production that questions the ef
(3,3,'An entertaining and thought-provoking production of Bollywood. It deals with conflicts of love, faith a
(4,4,'A confusing and gripping science fiction masterpiece. A complex story where the worlds of reality and d
(5,5,'A gritty and striking crime drama. Al Pacino unforgettable performance and the intense atmosphere in th
(6,6,'An intense and compelling character study. . The film draws the audience on a tense journey with a stri
(7,7,'A visual feast and a magnificent example of science fiction. This film, which deals with the theme of t
(8,8,'A heart touching film that tells the unique story of an unforgettable character. Tom Hanks extraordinar
(9,9,'A science fiction movie that attracts attention with Scarlett Johansson remarkable performance and str
(10,10,'Focusing on the lives of losers, the film attracts attention with its character depth and extraordina
```

```
),
```

```
humorous narrative in the film impress the audience.','2016-04-05'),
think deeply.','2020-11-12'),
rise and fall of Tony Montana ensures that you experience a movie full of unforgettable moments.','2019-09-02'),
e portrait in the dark world of Gotham City.','2015-12-30'),
anity.','2022-02-18'),
m make the audience both laugh and emotional.','2016-10-08'),
ng story based on exploring human potential.','2017-07-21'),
ities of life in a humorous way, makes its audience think.','2021-03-14');
```

3-ENTITY RELATION DIAGRAM (ERD)



4. Query Examples

Query 1

```
--Names of actors played in 2014
SELECT actor_name title
From film natural join actor
Where year = 2014
```

Data Output		Messages	Notifications
	title character varying (50)		
1	Aamir Khan		
2	Anushka Sharma		
3	Sanjay Dutt		
4	Matthew McConaughey		
5	Anne Hathaway		
6	Jessica Chastain		
7	Scarlet Johansson		
8	Morgan Freeman		
9	Choi Min-sik		

Explanation

We combined the movie and actor tables to access the year and actor_name data. With the where condition, we limited the actors who played in 2014.

Query 2

```
--user_name with 'a' in the name and over 18 years of age
SELECT user_name
FROM users
WHERE user_name like '%a%' and user_age > 18
```

Data Output		Messages	Notifications
<div> <div>≡</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>▼</div> <div>🗑️</div> <div>🗄️</div> <div>⬇️</div> <div>📈</div> </div>			
	user_name character varying (50) 🔒		
1	Feyza Şahan		
2	Deniz Kaya		
3	Burak Aydın		
4	Ömer Polat		
5	Selim Erdoğan		

Explanation

In the where condition, we checked whether the letter a is present or not with '%a%'. Our other limitation with and is that we chose users over the age of 18.

Query 3

```
--Names of movies in English
SELECT title , language
FROM film natural join country
WHERE language = 'English'
```











Data Output		Messages	Notifications
<div> <div>≡</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>▼</div> <div>🗑️</div> <div>🗄️</div> <div>⬇️</div> <div>📈</div> </div>			
	title character varying (50) 🔒	language character varying (50) 🔒	
1	Fight Club	English	
2	Intime	English	
3	Inception	English	
4	Scarface	English	
5	Joker	English	
6	Interstellar	English	
7	Forrest Gump	English	
8	Lucy	English	

Explanation

We combined the movie and country tables with NATURAL JOIN to access title and language data simultaneously. With the WHERE restriction, we only include those in English.

Query 4

```
--Number of movies with IMDb rating above  
SELECT count(*)  
FROM film  
WHERE imdb > 8
```

Data Output	Messages	Notifications
        		
	count bigint	
1		8

Explanation

With the WHERE restriction, we select the ones with an IMDb number greater than 8 and we find the number of these movies with count(*).

Query 5

```
--Names of directors who earn higher salaries than the average salary  
SELECT director_name , director_salary FROM director  
WHERE director_salary > (SELECT avg(director_salary)  
                        FROM director)
```

Data Output Messages Notifications		
<div> <div> <div>≡</div> <div>+</div> </div> <div> <div>📄</div> <div>▼</div> </div> <div> <div>📋</div> <div>▼</div> </div> <div> <div>🗑️</div> </div> <div> <div>🗄️</div> </div> <div> <div>⬇️</div> </div> <div> <div>📈</div> </div> </div>		
	director_name character varying (50) 🔒	director_salary integer 🔒
1	David Fincher	8000000
2	Cristopher Nolan	15000000
3	Brian De Palma	9000000
4	Robert Zemeckis	15000000

Query 6

```
--Movies in the comedy genre and produced after 2000
SELECT title
FROM film
WHERE year > 2000 and category_type = 'comedy'
```

Data Output

Messages

Notifications

≡+

📄

▼

📋

▼

🗑️

🗄️

⬇️

📈

title


character varying (50) 🔒

Explanation

With the WHERE restriction, we select films with category type = 'comedy' and production years older than 2000.

Query 7

```
--The movie or movies of the highest-earning director
SELECT title , director_name
FROM director, film
WHERE director_salary = (SELECT max(director_salary) FROM director) and film.film_id = director.film_id
```

Data Output Messages Notifications		
		
	title character varying (50) 🔒	director_name character varying (50) 🔒
1	Inception	Cristopher Nolan
2	Forrest Gump	Robert Zemeckis

Explanation

We combine the director and movie tables to access director salary and title data in the same table. In the WHERE condition, we synchronize the data with foreign keys.

Query 8

```
--Number of movies in each category
SELECT category_type , count(*)
FROM film
GROUP BY category_type
```

Data Output		Messages	Notifications
<div> <div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> </div> <div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> </div> </div>			
	category_type character varying	count bigint	
1	science fiction	3	
2	science fiction, come...	1	
3	comedy, drama	2	
4	science fiction, action	1	
5	Drama, Thriller	1	
6	psychological tension	1	
7	crime, drama	1	

Explanation

We separate each category with GROUP BY and keep the number of these data with count(*).

Query 9

```
--Average imdb of each country
SELECT country_name , avg(imdb)
FROM film , country
WHERE film.film_id = country.film_id
GROUP BY country_name
```

Data Output		Messages	Notifications
<div> <div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> </div> <div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> </div> </div>			
	country_name character varying (50)	avg double precision	
1	France	6.400000095367432	
2	Turkey	8.100000381469727	
3	USA	8.342857224600655	
4	India	8.100000381469727	

Explanation

We combine the movie and country tables to access country_name and imdb data in the same table. In the WHERE condition, we write the join condition according to the foreign key. With GROUP BY, we finally separate each data according to its group.

Query 10

```
--Sorting the 5 longest movies from longest to shortest  
SELECT title , length  
FROM film  
ORDER BY length desc LIMIT 5
```

Data Output

Messages

Notifications

☰+

▼

▼

	<div><div>title</div><div>character varying (50)</div><div></div></div>	<div><div>length</div><div>integer</div><div></div></div>
1	Scarface	170
2	Interstellar	169
3	PK	153
4	Inception	148
5	Forrest Gump	142

Explanation

With ORDER BY, we indicate that we will sort by length, and with LIMIT, we specify the number of data we want.

5. Connecting Database to Java

We will now connect the data we wrote in pgadmin to java.

We add the necessary packages for connection, receiving data from the user and creating an ArrayList.

```
import java.sql.Statement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Scanner;
```

Then we get the information required for the connection and send it to pgAdmin.

In case of an error, we ensure that we catch this error with try catch.

```
private static Connection con;

public static Connection Connect() {
    String url = "jdbc:postgresql://localhost:5432/filmX";
    String username = "postgres";
    String password = "admin";
    try {
        con = DriverManager.getConnection(url, username, password);
    } catch (Exception e) {
        System.err.println(e.getMessage());
        System.exit(status:1);
    }
    return con;
}
```

Now let's create a few basic functions for our application.

Our first function is to allow the user to log in to the system with their registered email and password.

In this function, it looks at the e-mails and passwords in our database, tests the accuracy of the entered information and continues to ask until the user enters the information correctly.

```
public static String logIn() {
    ArrayList<String> mailArray = new ArrayList<>();
    ArrayList<String> passArray = new ArrayList<>();
    Statement st = null;
    ResultSet rs1 = null;
    ResultSet rs2 = null;
    try {
        st = con.createStatement();
        String sql1 = "SELECT email FROM users";
        rs1 = st.executeQuery(sql1);
        while (rs1.next()) {
            String mailUser = rs1.getString(columnLabel:"email");
            mailArray.add(mailUser);
        }

        String sql2 = "SELECT password FROM users";
        rs2 = st.executeQuery(sql2);
        while (rs2.next()) {
            String password = rs2.getString(columnLabel:"password");
            passArray.add(password);
        }
    }
```

```
do {
    System.out.println(x:"Enter your email");
    String emailInput = sc.nextLine();
    System.out.println(x:"Enter your password");
    String passInput = sc.nextLine();

    if (mailArray.contains(emailInput) && passArray.contains(passInput)) {
        System.out.println(x:"Login Successful");
        break;
    } else {
        System.out.println(x:"Incorrect mail or password. Try again!");
    }
}
```

```

    } catch (Exception e) {
        System.err.println(e.getMessage());
        System.exit(status:1);
    } finally {
        try {
            if (rs1 != null)
                rs1.close();
            if (rs2 != null)
                rs2.close();
            st.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    return null;
}

```

Calling the function logIn() and Connect()

```

public static void main(String[] args) {

    Connect();
    logIn();

}

```

For incorrectly entered e-mail or password, we need to enter our information again until we enter it correctly.

```

Enter your email
FeyzaSahan@gmail.com
Enter your password
mugla1509
Incorrect mail or password. Try again!
Enter your email
Feyzasahan@gmail.com
Enter your password
mugla1509
Login Successful

```

When a new user wants to log in, he will use the register function. This function checks the database to see if there is such a user before, and if there is no such user, it performs the registration process.

```
public static String register() {
    ArrayList<String> mailArray = new ArrayList<>();
    Statement st = null;
    ResultSet rs = null;
    PreparedStatement ps = null;

    try {
        st = con.createStatement();
        String sql1 = "SELECT email FROM users";
        rs = st.executeQuery(sql1);
        while (rs.next()) {
            String mailUser = rs.getString(columnLabel:"email");
            mailArray.add(mailUser);
        }
        do {
            System.out.println(x:"Enter your email");
            String emailInput = sc.nextLine();
            System.out.println(x:"Enter your password");
            String passInput = sc.nextLine();
            System.out.println(x:"Enter your age");
            Integer ageInput = sc.nextInt();
            System.out.println(x:"Enter your name");
            String nameInput = sc.nextLine();
            sc.nextLine();
```

```
            if (mailArray.contains(emailInput)) {
                System.out.println(x:"Email is already registered");
            } else {
                String sql2 = "INSERT INTO users (user_name, user_age, email, password) VALUES
                (?, ?, ?, ?)";
                ps = con.prepareStatement(sql2);
                ps.setString(parameterIndex:1, nameInput);
                ps.setInt(parameterIndex:2, ageInput);
                ps.setString(parameterIndex:3, emailInput);
                ps.setString(parameterIndex:4, passInput);
                ps.executeUpdate();
                System.out.println(x:"Registration Successful");
                break;
            }
        } while (true);
    } catch (Exception e) {
        System.err.println(e.getMessage());
        System.exit(status:1);
    }
}
```

```

    } finally {
        try {
            if (rs != null)
                rs.close();
            if (st != null)
                st.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    return null;
}

```

Calling the function register() and Connect()

```

public static void main(String[] args) {

    Connect();
    register();

}

```

Enter your information.

```

Enter your email
selim_7@gmail.com
Enter your password
admin123
Enter your age
35
Enter your name
Selim Erdoğan
Registration Successful

```


Adding information to the database

Data Output Messages Notifications					
	user_id [PK] integer	user_name character varying (50)	user_age integer	email character varying (50)	password character varying (50)
1	1	Feyza Şahan	22	Feyzasahan@gmail.com	mugla1509
2	2	Deniz Kaya	34	denizKaya06@outlook.com	2343-deniz-2343
3	3	Burak Aydın	56	kralBurak1881@gmail.com	123kralkral123
4	4	Emre Demir	19	arabasevdasi@gmail.com	fordfocus
5	5	Caner Karahan	17	karahanli_polat@hotmail.com	12345678.CK
6	6	Gizem Çelik	25	gizemgizem@gmail.com	gizem123456
7	7	Ayşe Aktaş	16	aysesincan@gmail.com	yoldayuruyenkedi
8	8	Melis Tekin	36	34melis34@hotmail.com	tarantula61
9	9	İrem Arslan	17	izmirliirem@outlook.com	komaistas123
10	10	Ömer Polat	23	vatozpolat@gmail.com	vatozbaligi
11	16	Selim Erdoğan	35	selim_7@gmail.com	admin123

We also wrote a function to show the e-mail addresses of all users.

```
public static ArrayList<String> getAllUser() {
    ArrayList<String> mailArray = new ArrayList<>();
    Statement st = null;
    ResultSet rs = null;
    try {
        st = con.createStatement();
        String sql1 = "SELECT email FROM users";
        rs = st.executeQuery(sql1);
        while (rs.next()) {
            String mailUser = rs.getString(columnLabel:"email");
            mailArray.add(mailUser);
        }
    } catch (
        Exception e) {
        System.err.println(e.getMessage());
        System.exit(status:1);
    }
}
```

```

    } finally {
        try {
            if (rs != null)
                rs.close();
            if (st != null)
                st.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    return mailArray;
}

```

Calling the function getAllUser() and Connect()

```

public static void main(String[] args) {

    Connect();

    var allUser = getAllUser();

    for (var user : allUser) {
        System.out.println(user);
    }
}

```

Output

```

Feyzasahan@gmail.com
denizKaya06@outlook.com
kralBurak1881@gmail.com
arabasevdasi@gmail.com
karahanli_polat@hotmail.com
gizengizem@gmail.com
aysesincan@gmail.com
34melis34@hotmail.com
izmirliirem@outlook.com
vatozpolat@gmail.com
selim_7@gmail.com

```