TASK1

Netflix is ​​an American entertainment company, one of the main goal of this company is to provide access to watch films and series in the network. In the business rule, goes Production Company, Producer, Film, Director, Scenarist, Operator and Actor .By help SQL we created database, the aim of the database is to show up data of the cinema. It is needed to provide information about the cinema, by whom this cinema was directed, recorded, edited, who took participation as an actor and etc. The users are ordinary people who are very keen on cinematography, they can look at the scenery or rating of the cinema to understand how this cinema is interesting. The system should solve ignorance pieces of knowledge information about the cinema. All information that is related to the cinema can be available to the database, e.g. Cast team, Genre, Country and etc.

TASK2

It was decided to create ERD diagram by using Crows Foot Notation, there are seven tables each of them has several attributes and they have a relation between them, Movie has data regarding cinema, for instance, name of a movie, country of movie and genre of movie. The next table is a creative team, this table consists of data of the developers of the exact cinema. In addition, there are a couple of tables that have relation to the finance they are Investor table and Finance table, in these tables consist the name of Production Company and Budget of the cinema. The fifth table is age limit which has data regarding starting at what age is it the film is granted for watching. Also, there is a Release table, in this table you are able to identify the release date of the cinema. And the last table is Running Time table, Running Time table has a responsibility to provide an answer to the question as to how long is this movie?

TASK3

In this task, it is required to create tables and create attributes according to crows foot notation.

CREATE TABLE creative\_team(

creative\_id INT,

producer VARCHAR(255),

director VARCHAR(255),

scenarist VARCHAR(255),

operator VARCHAR(255),

editor VARCHAR(255),

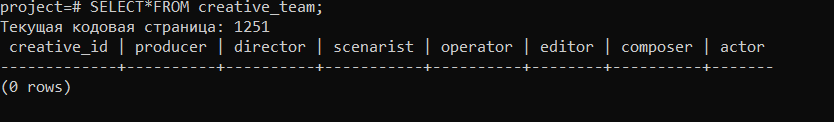
composer VARCHAR(255),

actor VARCHAR(255),

PRIMARY KEY (creative\_id)

);

According to this code, it was created creative team table, in this table were created seven attributes with char datatype as illustrated in crows foot notation and also there is ID attribute which have integer and which was created for these rest of the seven attributes.



CREATE TABLE investor(

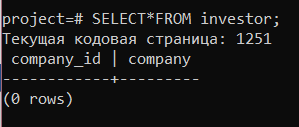
company\_id INT,

company VARCHAR(255),

PRIMARY KEY (company\_id)

);

This code written for creating investor table, which have company id and name of the production company.



CREATE TABLE running\_time(

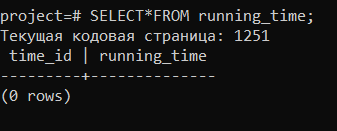
time\_id INT,

running\_time TIME,

PRIMARY KEY (time\_id)

);

For creating running time table, we write code which was mentioned above, in this code we append time ID and running time attribute with Time datatype.



CREATE TABLE release(

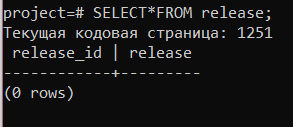
release\_id INT,

release DATE,

PRIMARY KEY (release\_id)

);

For this table, it is necessary to include attribute as known as date of release with date datatype.



CREATE TABLE finance(

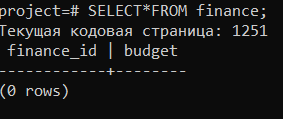
finance\_id INT,

budget INT,

PRIMARY KEY (finance\_id)

);

For finance table, it should be included budget of the cinema and ID for this attribute.



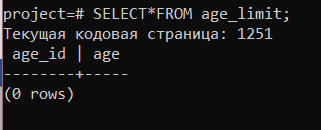
CREATE TABLE age\_limit(

age\_id INT,

age INT,

PRIMARY KEY (age\_id)

As claimed by this code, it was designed to age limit table which will have data regarding age limitation.



);

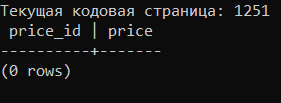
CREATE TABLE price(

price\_id INT,

price INT,

PRIMARY KEY (price\_id)

);



CREATE TABLE movie(

id INT NOT NULL,

name VARCHAR(255),

country VARCHAR(255),

genre VARCHAR(255),

PRIMARY KEY (id),

team\_id INT REFERENCES creative\_team(creative\_id),

time\_id INT REFERENCES running\_time(time\_id),

release\_id INT REFERENCES release(release\_id),

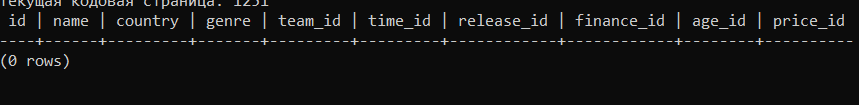
finance\_id INT REFERENCES finance(finance\_id),

age\_id INT REFERENCES age\_limit(age\_id),

price\_id INT REFERENCES price(price\_id)

);

As stated by this code, it was created table as known as the movie which has four attributes and references to the ID columns of the previous tables.

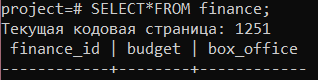


TASK4

In the fourth task we are responsible to write at least five alter table statements.

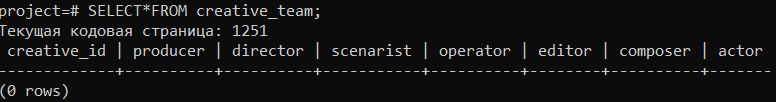
In this statement in the finance table we add column as entitled box office with integer datatype.

ALTER TABLE finance ADD COLUMN box\_office INT;



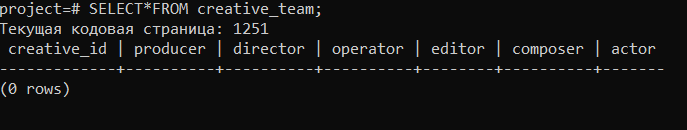
In the second statement, we understand that any movie has producer, therefore we wrote Not Null statement to the creative team table.

ALTER TABLE creative\_team ALTER producer SET NOT NULL;



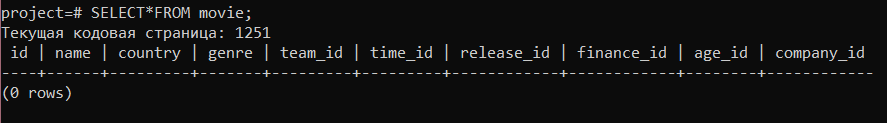
By this, alter table statement we dropped column as known as scenarist.

ALTER TABLE creative\_team DROP COLUMN scenarist;



The fourth alter table statement add column company ID with the integer datatype.

ALTER TABLE movie ADD COLUMN company\_id INT;



The last statement is responsible to update company id column in the movie and investor tables to the foreign key.

ALTER TABLE movie ADD FOREIGN KEY (company\_id) REFERENCES investor (company\_id);

TASK5

In the fifth task, we decided to input values to the tables, it is needed for filling in tables and database. In addition, in this task values should be affected by a statement with the condition for updating or deleting.

INSERT INTO creative\_team

VALUES

(1, 'Frank Darabont','Frank Darabont', 'Roger Deakins', 'Richard Bruse','Thomas Newman', 'Tim Robbins'),

(2, 'Peter Jackson', 'Peter Jackson', 'Andrew Lesnie', 'Jhon Gilbert', 'Ghovard Shor', 'Elijah Wood'),

(3, 'Christopher Nolan', 'Christopher Nolan', 'Hoyte Van', 'Li Smith', 'Hans Zimmer', 'Matthew McConaughey'),

(4, 'Branko Lustig', 'Steven Spielberg', 'Janusz Kaminski', 'Michael Kan', 'Jhon Williams', 'Liam Neeson'),

(5, 'Wendy Finerman', 'Robert Zemeckis', 'Don Burgess', 'Arthur Schmidt','Alan Silvestri', 'Tom Hanks'),

(6, 'Nicolas Duval', 'Olivier Nakache', 'Mathieu Vadepied', 'Dorian Rigal-Ansous', 'Ludovico Einaudi', 'Francois Cluzet'),

(7, 'Don Hahn', 'Roger Allers', 'Chris Sanders','Ivan Bilancio','Hans Zimmer','Matthew Broderick'),

(8, 'Lawrence Bender', 'Quentin Tarantino', 'Andrzej Sekula','Sally Menke','David Wasco','John Travolta'),

(9, 'Matthew Anderson', 'Guy Ritchie','Guy Ritchie','Alan Stewart','Christopher Benstead','Matthew McConaughey'),

(10, 'Ross Grayson Bell', 'David Fincher', 'Jeff Cronenweth','James Haygood','Dust Brothers','Edward Norton'),

(11,'Frank Darabont','Frank Darabont','David Tattersall','Richard Francis','Thomas Newman','Tom Hanks');



INSERT INTO investor

VALUES

(1, 'Castle Rock Entertainment'),

(2, 'New Line Cinema'),

(3, 'Paramount Pictures'),

(4, 'Universal Pictures'),

(5, Sony Pictures’),

(6, 'Garmount'),

(7, 'Walt Disney Pictures'),

(8, 'A Band a Part'),

(9, 'Miramax Films'),

(10, 'Fox 2000 Pictures'),

(11, 'Warner Bros');



INSERT INTO running\_time

VALUES

(1, '2:20'),

(2, '2:58'),

(3, '2:49'),

(4, '3:15'),

(5, '2:22'),

(6, '1:52'),

(7, '1:28'),

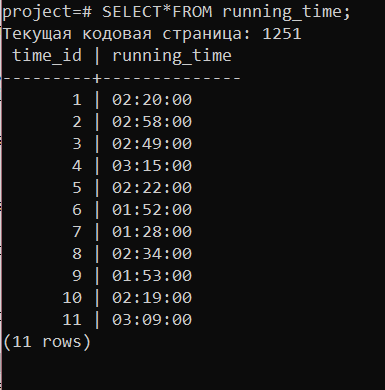
(8, '2:34'),

(9, '1:53'),

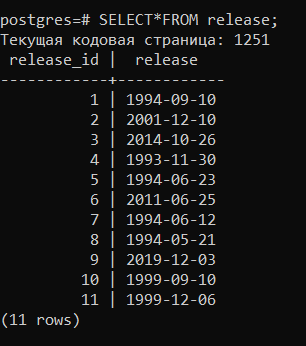
(10, '2:19'),

(11, '3:09')

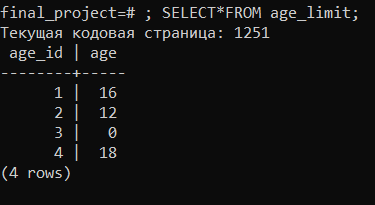
;



INSERT INTO release  
VALUES  
(1,'1994-09-10'),  
(2,'2001-12-10'),  
(3,'2014-10-26'),  
(4,'1993-11-30'),  
(5,'1994-06-23'),  
(6,'2011-06-25'),  
(7,'1994-06-12'),  
(8,'1994-05-21'),  
(9,'2019-12-3'),  
(10,'1999-09-10'),  
(11,'1999-12-06');



INSERT INTO age\_limit  
VALUES  
(1, 16),  
(2, 12),  
(3, 0),  
(4, 18);



INSERT INTO FINANCE

VALUES

(1,25000000,28418687),

(2,93000000,868385360),

(3,165000000,677463813),

(4,22000000,96065768),

(5,55000000,677387716),

(6,9500000,426588510),

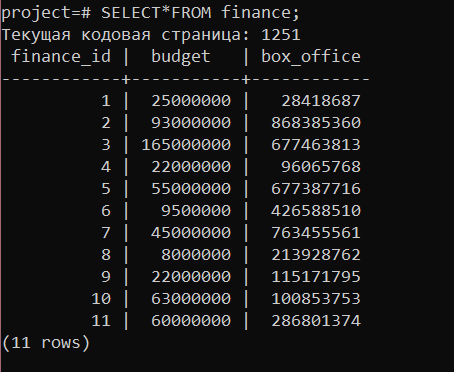
(7,45000000,763455561),

(8,8000000,213928762),

(9,22000000,115171795),

(10,63000000,100853753),

(11,60000000,286801374);



INSERT INTO price

VALUES

(1, 3500),

(2, 5000),

(3, 4250),

(4, 2750),

(5, 3750),

(6, 1890),

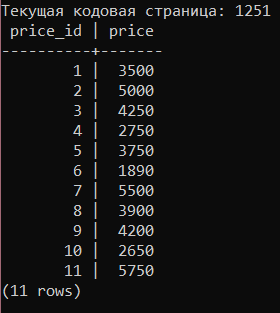
(7, 5500),

(8, 3900),

(9, 4200),

(10, 2650),

(11, 5750);



INSERT INTO movie

VALUES

(1, 'The Swashank Redemption', 'The USA', 'Drama', '1','1','1','1','1','1','1'),

(2, 'The Lord of the Rings', 'The USA', 'Fantasy','2','2','2','2','2','2','2'),

(3, 'Interstellar', 'The USA', 'Fantasy','3','3','3','3','2','3','3'),

(4, 'Schindlers List', 'The USA', 'Fantasy','4','4','4','4','1','4','4'),

(5, 'Forrest Gump', 'The USA', 'Drama','5','5','5','5','2','5','5'),

(6, '1+1', 'France', 'Drama','6','6','6','6','1','6','6'),

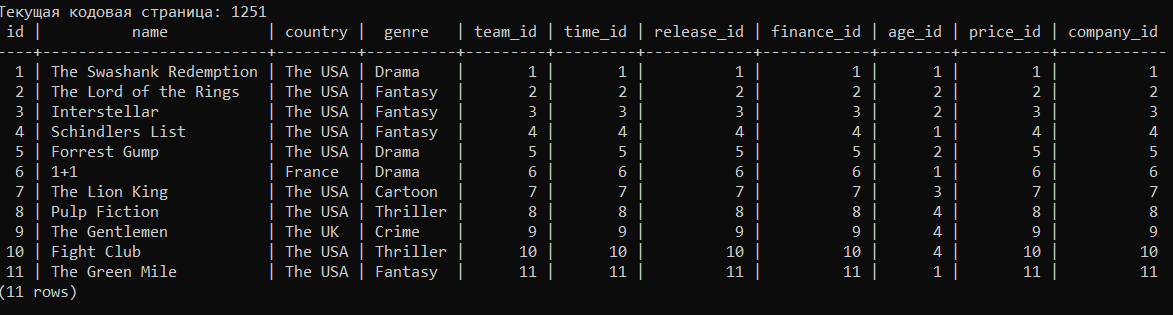
(7, 'The Lion King', 'The USA', 'Cartoon','7','7','7','7','3','7','7'),

(8, 'Pulp Fiction', 'The USA', 'Thriller','8','8','8','8','4','8','8'),

(9, 'The Gentlemen', 'The UK', 'Crime','9','9','9','9','4','9','9'),

(10, 'Fight Club', 'The USA', 'Thriller','10','10','10','10','4','10','10'),

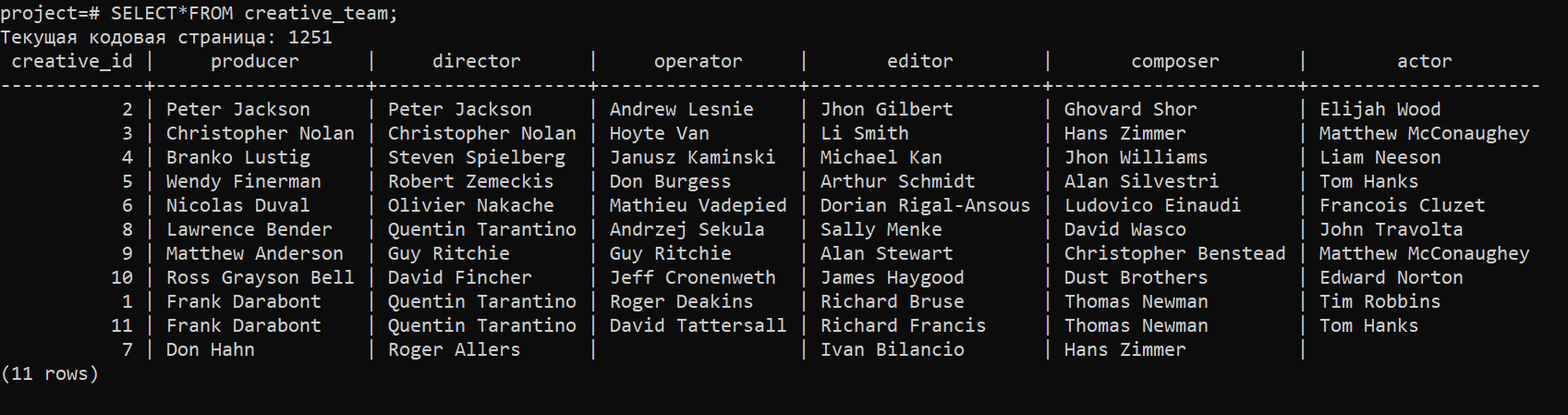
(11, 'The Green Mile', 'The USA', 'Fantasy','11','11','11','11','1','11','11');

; 

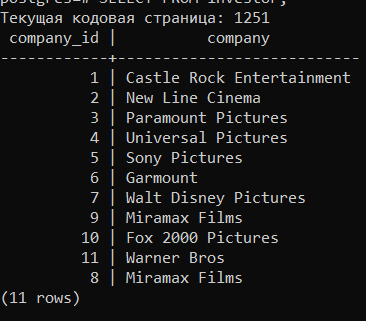
UPDATE creative\_team SET director = 'Quentin Tarantino' WHERE director = 'Frank Darabont';

UPDATE creative\_team SET operator = null WHERE creative\_id = 7;

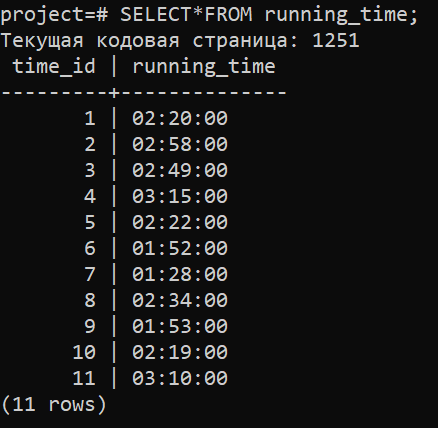
UPDATE creative\_team SET actor = null WHERE creative\_id = 7;



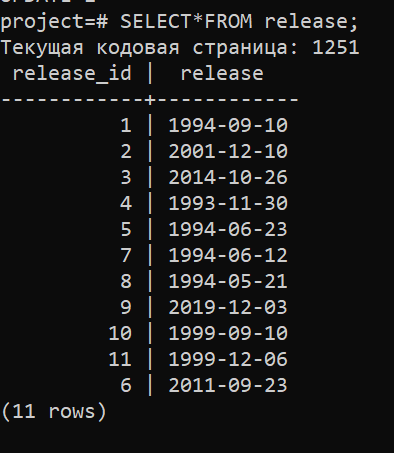
UPDATE investor SET company = 'Miramax Films' WHERE company\_id = 8;



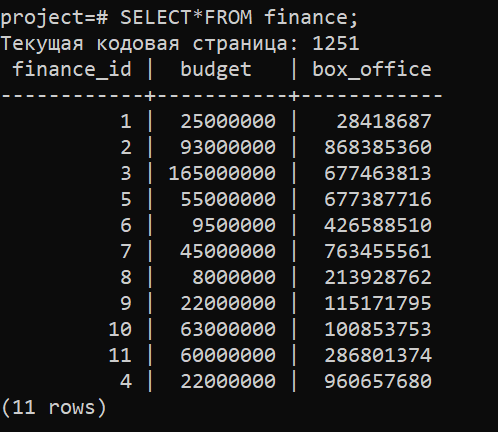
UPDATE running\_time SET running\_time = '3:10:00' WHERE time\_id = 11;



UPDATE release SET release = '2011-09-23' WHERE release\_id = 6;

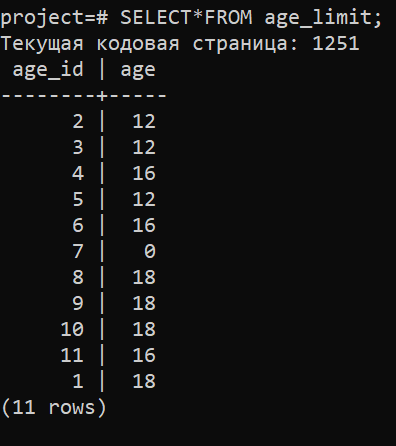


UPDATE finance SET box\_office = 960657680 WHERE finance\_id = 4;

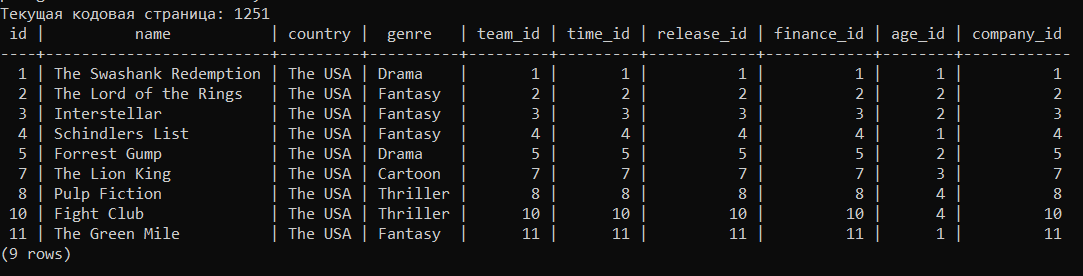


UPDATE age\_limit SET age = 18 WHERE age\_id = 1;

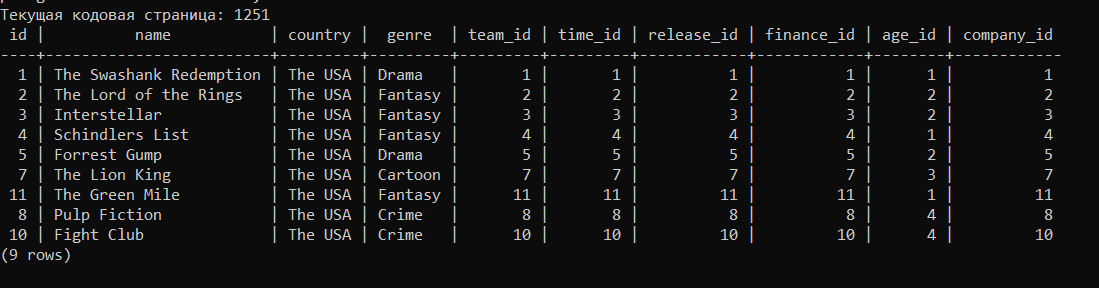
UPDATE movie SET age\_id = 4 WHERE id = 1;



DELETE FROM movie WHERE country != 'The USA';



UPDATE movie SET genre = 'Crime' WHERE genre = 'Thriller';



TASK 6

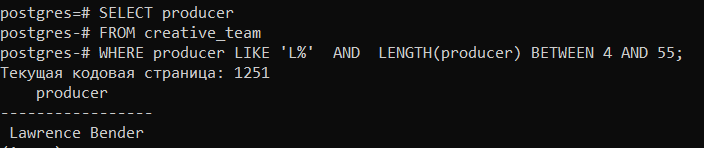
In this task we need to write at least ten queries, and it is necessary to use distinct conditions, joiner operators and aggregate functions.

1) SELECT producer

FROM creative\_team

WHERE producer LIKE 'L%' AND LENGTH(producer) BETWEEN 4 AND 50;

This statement serves us to identify producer with the name on L and this name need to have length between 4 and 55 characters.



2) SELECT operator, composer FROM creative\_team ORDER BY operator ASC, composer DESC;

This code supposed for sorting composer column for descending order and ascending order for operator column.



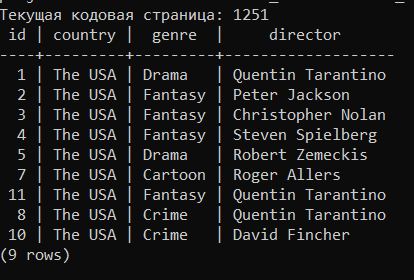
3) SELECT id , country, movie.genre, creative\_team.director

FROM movie

LEFT OUTER JOIN creative\_team

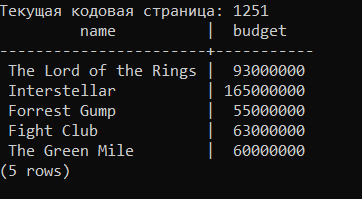
ON movie.id=creative\_team.creative\_id;

In this statement we used left outer join operator for mixing country, genre and director columns, by help this code it is available to identify in what country this cinema was filmed, what kind of genre film, and by who this cinema was directed.



4) SELECT [movie.name](http://movie.name/), finance.budget  
FROM movie  
INNER JOIN finance  
ON movie.finance\_id=finance.finance\_id  
WHERE budget> (SELECT AVG(budget) FROM finance);

This code works to output common Inner Join data of the name of the cinema and budget in the finance. It helps to identify what budget did exact cinema had.



5)

SELECT movie.id, movie.country, investor.company

FROM movie

JOIN investor

ON movie.company\_id=investor.company\_id

WHERE investor.company\_id<> '01';

This query have responsibility to output cinema ID, country of the cinema and production company name where investor id more or lesser than 1;



6)

SELECT AVG(budget),COUNT(\*)

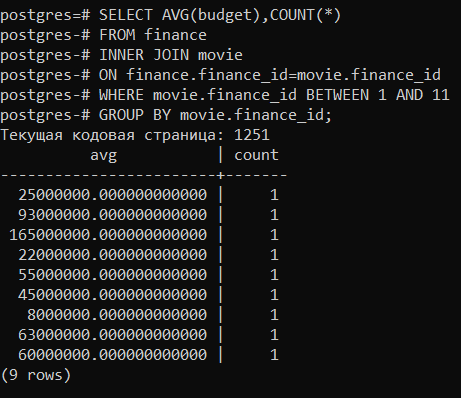
FROM finance

INNER JOIN movie

ON finance.finance\_id=movie.finance\_id

WHERE movie.finance\_id BETWEEN 1 AND 11

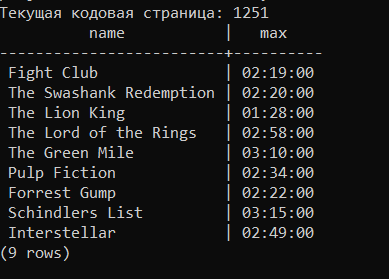
GROUP BY movie.finance\_id;

This code should output average of the budget and amount of the count from finance table by inner join operator, on finance id in the finance table and finance id in the movie table where finance id of the movie table is between 1 and 11, it also should be sorted by order finance id of the movie table.

7)

SELECT [movie.name](http://movie.name/),MAX(running\_time)  
FROM movie  
JOIN running\_time  
ON movie.time\_id=running\_time.time\_id  
GROUP BY [movie.name](http://movie.name/);

This code was written for outputting name of the cinema and running time by group name of the cinema.

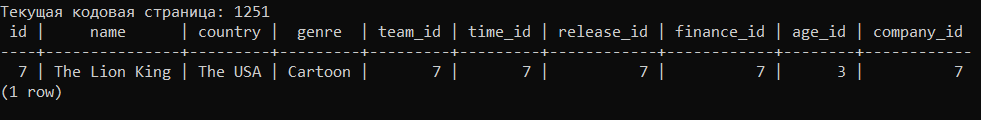


8) SELECT \*

FROM movie

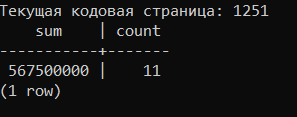
WHERE age\_id=(SELECT age\_id FROM age\_limit WHERE age=0);

In this case code outputs data of the cinema where age limit is equal to 0;



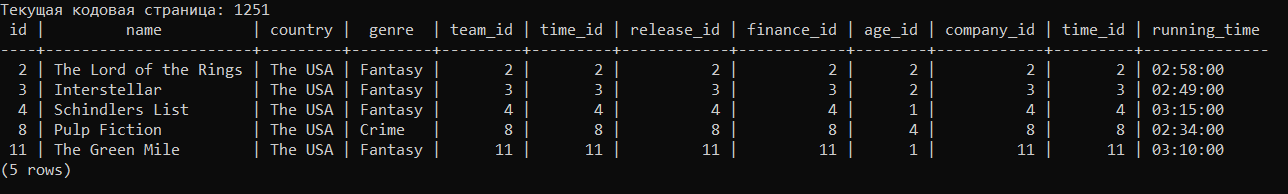
9) SELECT SUM(budget),COUNT(\*) FROM finance;

The ninth code of this task should output the sum of the budget and count of the rows from the finance table.



10) SELECT \* FROM movie FULL OUTER JOIN running\_time ON movie.time\_id=running\_time.time\_id WHERE running\_time> (Select AVG(running\_time) FROM running\_time);

In this code we used subquerie to output running times which are longer than average running time.



TASK 7

In the seventh task of the project asked to us to write at least five subqueries, with the single row, multiple row and multiple-column subqueries.

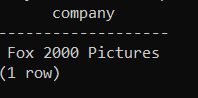
1)

SELECT company

FROM investor

WHERE company\_id = (SELECT company\_id FROM movie WHERE company\_id = 10);

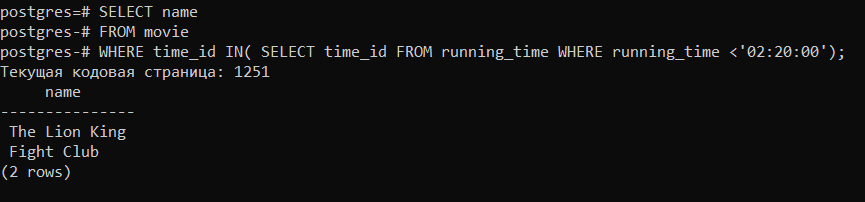
By helping this subquery we are able to identify the tenth company.



2) SELECT name

FROM movie

WHERE time\_id IN( SELECT time\_id FROM running\_time WHERE running\_time <'02:20:00');

In this code, we output the name of the cinema which longs longer than 2 hours and 20 minutes.

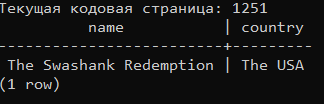
3)

SELECT name, country

FROM movie

WHERE movie.company\_id IN (SELECT company\_id FROM investor WHERE company\_id = 1);

According to this code, we can output data of the cinema of the first production company.



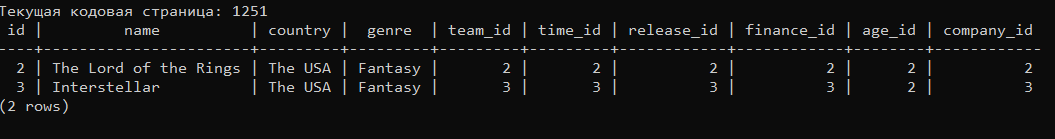
4)

SELECT\*

FROM movie

WHERE finance\_id IN (SELECT finance\_id FROM finance WHERE budget > 63000000);

This subquery is able to output data list of the cinemas which have budget more than 63000000.



5) SELECT\*

FROM movie,release

WHERE movie.release\_id IN (SELECT release\_id FROM release WHERE release >= '1999-09-10');

This last subquery serves us to output data of the cinemas which is not older than 1999-09-10.

