Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

ПСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет информатики и вычислительной техники

Кафедра вычислительной техники

ПРЕДМЕТ «Базы Данных»

# *КУРСОВОЙ ПРОЕКТ*

***«Вариант 18»***

РАЗРАБОТКА БАЗЫ ДАННЫХ В СУБД POSTGRESQL

Выполнил студент группы 0024-01

Невзоров Л.Е.

Проверил преподаватель

Вертешев А.С.

Псков

2022

**Задание**

Тема: сеть кинотеатров по прокату фильмов.

Объекты: кинотеатры, фильмы, актеры, режиссеры, жанры.

Для моделирования задачи необходимо хранить следующую информацию:

\* Название фильма

\* Год выпуска

\* Режиссер

\* Жанр

\* Актеры, снимавшиеся в фильме

\* Кинотеатры, в которых идет прокат данного фильма

\* Сроки проката фильма (с... по...)

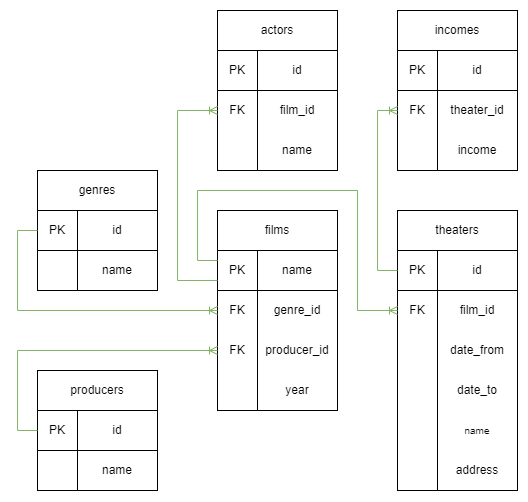
\* Адрес кинотеатра

Дополнительные условия:

\* в одном фильме может сниматься несколько актеров;

\* каждый фильм показывается более чем в одном кинотеатре.

**Схема**



**Создание таблиц**

CREATE TABLE "producers" (

  id SERIAL PRIMARY KEY,

  name VARCHAR(50) NOT NULL

);

INSERT INTO producers (name)

VALUES

  ('Shay Walters'),

  ('Inga Hendricks'),

  ('Paul Sutton'),

  ('Anjolie Moore'),

  ('Tanner Burton');

CREATE TABLE "genres" (

  id SERIAL PRIMARY KEY,

  name VARCHAR(30) UNIQUE NOT NULL

);

INSERT INTO genres (name)

VALUES

  ('Drama'),

  ('Comedy'),

  ('Thriller'),

  ('Biography'),

  ('Documentary'),

  ('Horror'),

  ('Anime');

CREATE TABLE "films" (

  id SERIAL PRIMARY KEY,

  name VARCHAR(255) UNIQUE NOT NULL,

  genre\_id INTEGER NOT NULL REFERENCES genres(id) ON DELETE CASCADE,

  producer\_id INTEGER NOT NULL REFERENCES producers(id) ON DELETE CASCADE,

  year VARCHAR(4)

);

INSERT INTO films (name,genre\_id,producer\_id,year)

VALUES

  ('The Green Mile',4,2,'2021'),

  ('Schindlers List',4,12,'2020'),

  ('Escape from Shawshank',3,8,'2004'),

  ('Forrest Gump',3,9,'2005'),

  ('The Lord of the Rings',1,6,'2016');

CREATE TABLE "actors" (

  id SERIAL PRIMARY KEY,

  film\_id VARCHAR(50) NOT NULL REFERENCES films(name) ON DELETE CASCADE,

  name VARCHAR(50) NOT NULL

);

INSERT INTO actors (film\_id,name)

VALUES

  ('The Green Mile','Xenos Wallace'),

  ('The Green Mile','Felix Maynard'),

  ('The Green Mile','Reese Banks'),

  ('The Green Mile','Ashely Meyer'),

  ('The Green Mile','Nita Workman');

CREATE TABLE "theaters" (

  id SERIAL PRIMARY KEY,

  film\_id VARCHAR NOT NULL REFERENCES films(name) ON DELETE CASCADE,

  date\_from VARCHAR(20) NOT NULL,

  date\_to VARCHAR(20) NOT NULL,

  name VARCHAR(50) NOT NULL,

  address VARCHAR(50) NOT NULL

);

INSERT INTO theaters (film\_id,date\_from,date\_to,name,address)

VALUES

  ('The Green Mile','Sep 25, 2022','Dec 18, 2022','Mirage','476-3720 Curae St.'),

  ('Schindlers List','Sep 19, 2022','Dec 11, 2022','Mirage','476-3720 Curae St.'),

  ('Escape from Shawshank','Sep 23, 2022','Dec 1, 2022','Mirage','476-3720 Curae St.'),

  ('Forrest Gump','Sep 10, 2022','Dec 2, 2022','Mirage','476-3720 Curae St.'),

  ('The Lord of the Rings','Sep 22, 2022','Dec 16, 2022','Mirage','476-3720 Curae St.');

CREATE TABLE "incomes" (

  id SERIAL PRIMARY KEY,

  theater\_id INTEGER NOT NULL REFERENCES theaters(id) ON DELETE CASCADE,

  income INTEGER NOT NULL

);

INSERT INTO incomes (theater\_id,income)

VALUES

  (1,161347),

  (2,871968),

  (3,400827),

  (4,563866),

  (5,703793);

**Запросы**

1)Три запроса на выборку с использованием join, использовать не мене 3 таблиц

SELECT fi.name,

       ge.name,

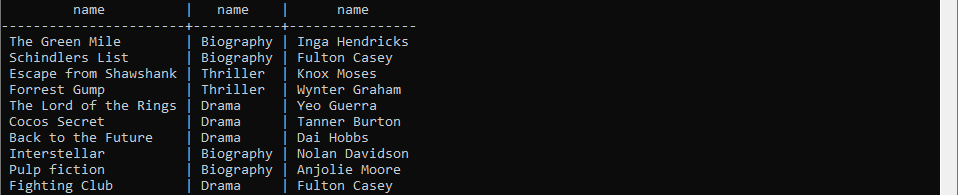
       pr.name

FROM films AS fi

JOIN genres AS ge ON ge.id=fi.genre\_id

JOIN producers AS pr ON pr.id=fi.producer\_id

LIMIT 10;



SELECT fi.name,

       ac.name,

       th.name

FROM films AS fi

JOIN actors AS ac ON fi.name=ac.film\_id

JOIN theaters AS th ON fi.name=th.film\_id

LIMIT 10;



SELECT fi.name,

       th.name,

       ic.income

FROM films AS fi

JOIN theaters AS th ON fi.name=th.film\_id

JOIN incomes AS ic ON th.id=ic.theater\_id

LIMIT 10;



[2) Запросы на объединение, пересечение, разность, произведение, проекцию, деление](#_Toc91305837)

SELECT name,

       year

FROM films

WHERE name LIKE '%S%'

UNION

SELECT name,

       year

FROM films

WHERE name LIKE '%M%';



SELECT fi.name,

       th.name,

       ic.income

FROM films AS fi

JOIN theaters AS th ON fi.name=th.film\_id

JOIN incomes AS ic ON th.id=ic.theater\_id

WHERE ic.income <= 600000

  AND ic.income >= 200000

INTERSECT

SELECT fi.name,

       th.name,

       ic.income

FROM films AS fi

JOIN theaters AS th ON fi.name=th.film\_id

JOIN incomes AS ic ON th.id=ic.theater\_id

WHERE ic.income > 500000;



SELECT fi.name,

       th.name,

       ic.income

FROM films AS fi

JOIN theaters AS th ON fi.name=th.film\_id

JOIN incomes AS ic ON th.id=ic.theater\_id

EXCEPT

SELECT fi.name,

       th.name,

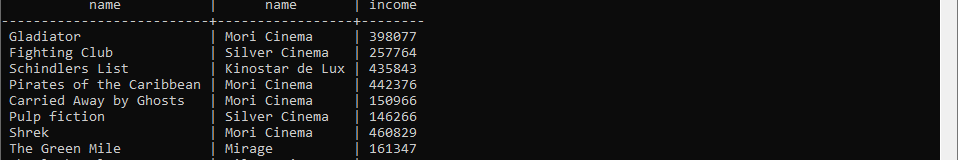
       ic.income

FROM films AS fi

JOIN theaters AS th ON fi.name=th.film\_id

JOIN incomes AS ic ON th.id=ic.theater\_id

WHERE ic.income > 500000;



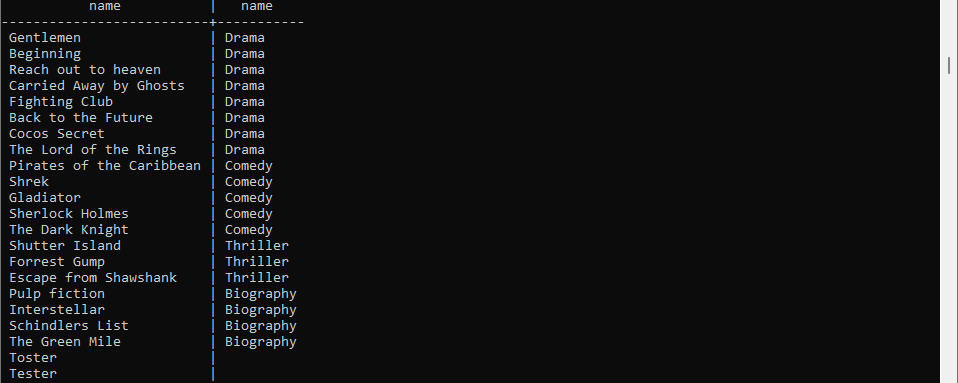
[3) Запросы на соединение: left join, right join, full outer join](#_Toc91305838)

SELECT fi.name,

       ge.name

FROM films AS fi

LEFT OUTER JOIN genres AS ge ON ge.id=fi.genre\_id;

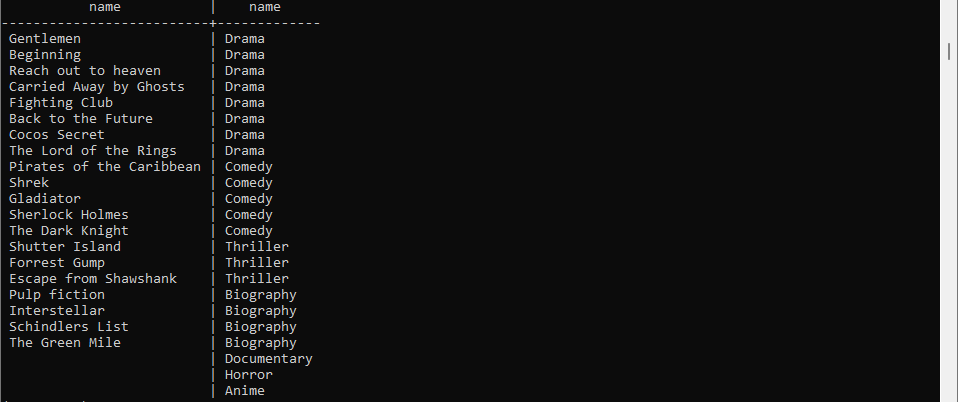


SELECT fi.name,

       ge.name

FROM films AS fi

RIGHT OUTER JOIN genres AS ge ON ge.id=fi.genre\_id;

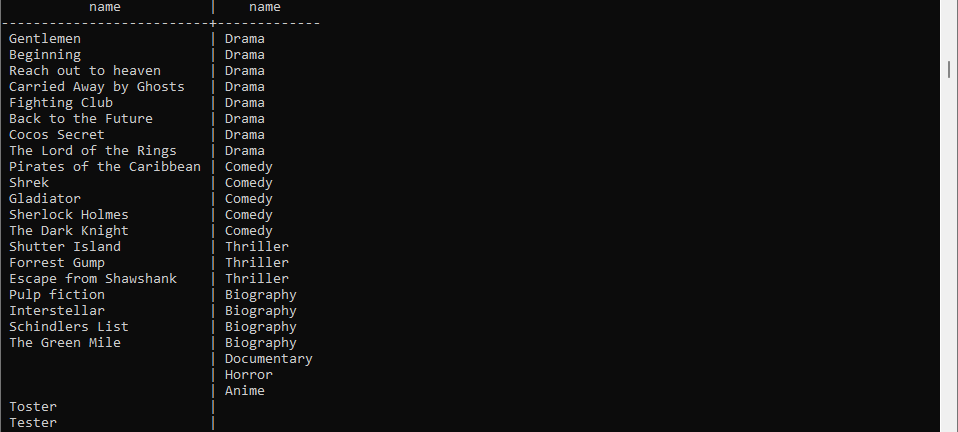


SELECT fi.name,

       ge.name

FROM films AS fi

FULL OUTER JOIN genres AS ge ON ge.id=fi.genre\_id;



4) 2 Подзапроса: Простой, Сложный

SELECT fi.name,

       th.name,

       ic.income

FROM films AS fi

LEFT OUTER JOIN theaters AS th ON fi.name=th.film\_id

LEFT OUTER JOIN incomes AS ic ON th.id=ic.theater\_id

WHERE ic.income > 900000

LIMIT 10;



SELECT fi.name,

       th.name,

       ic.income

FROM films AS fi

LEFT OUTER JOIN theaters AS th ON fi.name=th.film\_id

LEFT OUTER JOIN incomes AS ic ON th.id=ic.theater\_id

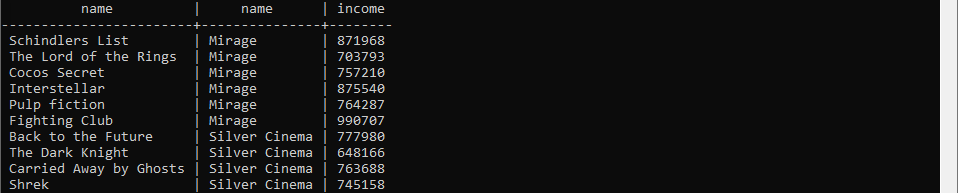
WHERE ic.income > (SELECT AVG(ic.income)

                   FROM incomes AS ic

                   JOIN theaters AS th ON th.id=ic.theater\_id

                   WHERE th.date\_from BETWEEN 'Sep 1, 2022' AND 'Sep 20, 2022')

LIMIT 10;



5) Запросы с различными уровнями вложенности подзапросов в частях SELECT, FROM, WHERE, HAVING

SELECT fi.name,

       th.name,

       ic.income

FROM films AS fi

LEFT OUTER JOIN theaters AS th ON fi.name=th.film\_id

LEFT OUTER JOIN incomes AS ic ON th.id=ic.theater\_id

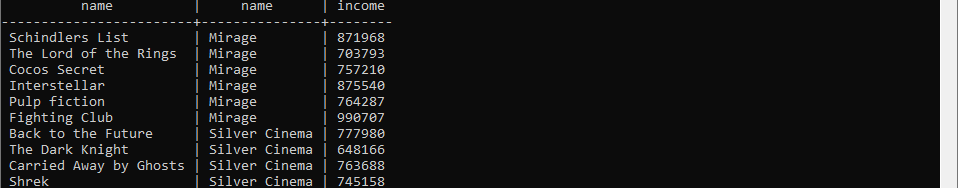
WHERE ic.income > (SELECT AVG(ic.income)

                   FROM incomes AS ic

                   JOIN theaters AS th ON th.id=ic.theater\_id

                   WHERE th.date\_from BETWEEN 'Sep 1, 2022' AND 'Sep 10, 2022')

LIMIT 10;



SELECT tbl.film\_name,

       tbl.theater\_name,

       tbl.income

FROM (SELECT fi.name AS film\_name,

             th.name AS theater\_name,

             ic.income

      FROM films AS fi

      LEFT OUTER JOIN theaters AS th ON fi.name=th.film\_id

      LEFT OUTER JOIN incomes AS ic ON th.id=ic.theater\_id) AS tbl

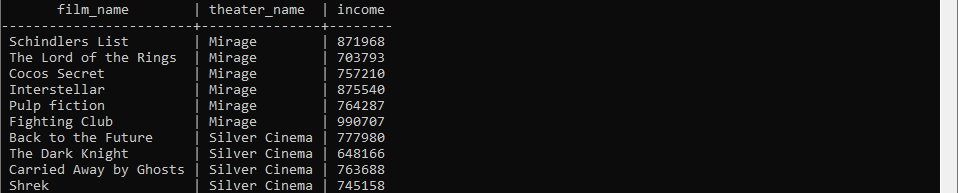
WHERE tbl.income > (SELECT AVG(ic.income)

                   FROM incomes AS ic

                   JOIN theaters AS th ON th.id=ic.theater\_id

                   WHERE th.date\_from BETWEEN 'Sep 1, 2022' AND 'Sep 10, 2022')

LIMIT 10;



SELECT tbl.theater\_name,

       SUM(tbl.income)

FROM (SELECT fi.name AS film\_name,

             th.name AS theater\_name,

             ic.income

      FROM films AS fi

      LEFT OUTER JOIN theaters AS th ON fi.name=th.film\_id

      LEFT OUTER JOIN incomes AS ic ON th.id=ic.theater\_id) AS tbl

WHERE tbl.income > (SELECT AVG(ic.income)

                    FROM incomes AS ic

                    JOIN theaters AS th ON th.id=ic.theater\_id

                    WHERE th.date\_from BETWEEN 'Sep 1, 2022' AND 'Sep 10, 2022')

GROUP BY tbl.theater\_name

HAVING SUM(tbl.income) > (SELECT AVG(ic.income)

                          FROM incomes AS ic

                          JOIN theaters AS th ON th.id=ic.theater\_id

                          WHERE th.date\_to BETWEEN 'Dec 1, 2022' AND 'Dec 10, 2022')

LIMIT 10;



[6) Создание ролей пользователей user, guest](#_Toc91305840)

DROP ROLE IF EXISTS test;

-- создание пользователя

CREATE ROLE test WITH LOGIN PASSWORD 'passworduser';

-- выдаем привелегии

GRANT ALL ON DATABASE grade TO test;

-- просмотр зарегистрированных пользователей

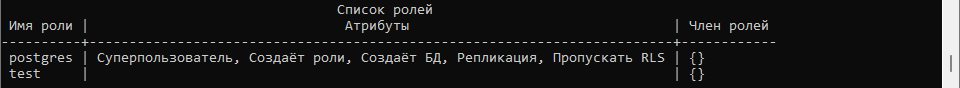
\du

-- забираем привелегии

REVOKE ALL ON DATABASE grade FROM test;

-- удаляем пользователя

DROP ROLE test;



DROP ROLE IF EXISTS guest;

-- создание пользователя

CREATE ROLE guest WITH LOGIN PASSWORD 'password';

-- выдаем привелегии

GRANT ALL ON students TO guest;

-- просмотр зарегистрированных пользователей

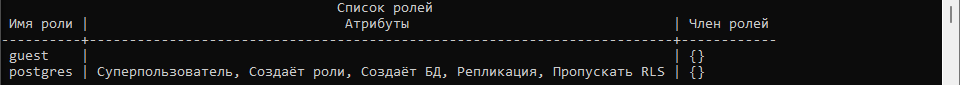
\du

-- забираем привелегии

REVOKE ALL ON students FROM guest;

-- удаляем пользователя

DROP ROLE guest;



Проверка, может ли GUEST (гость) зайти в недоступные ему таблицы в базе данных:





[7) Создание двух транзакций с тремя точками восстановления на удаление, создание транзакции на вставку](#_Toc91305841)





[8) Создание двух представлений: изменяемое, неизменяемое](#_Toc91305842)

-- создадим изменяемое представление

CREATE OR REPLACE VIEW film\_view AS

    SELECT \*

    FROM films

    WHERE year = '2012'

       OR year = '2014';

-- изменим представление

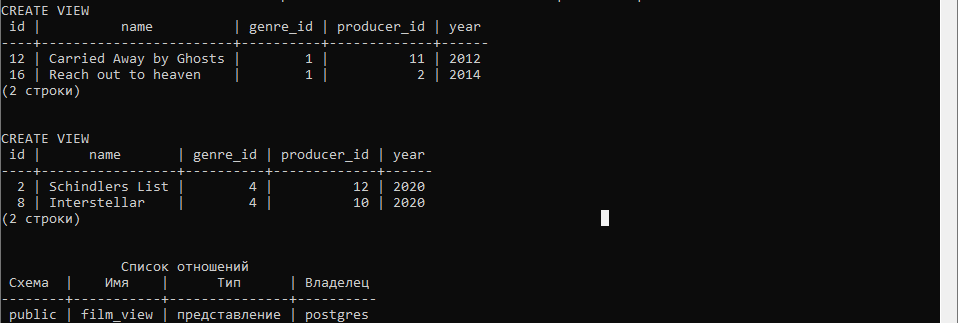
CREATE OR REPLACE VIEW film\_view AS

    SELECT \*

    FROM films

    WHERE year = '2018'

       OR year = '2020';



CREATE VIEW film\_view AS

    SELECT \*

    FROM films

    WHERE year = '2012'

       OR year = '2024';

-- попробуем его изменить

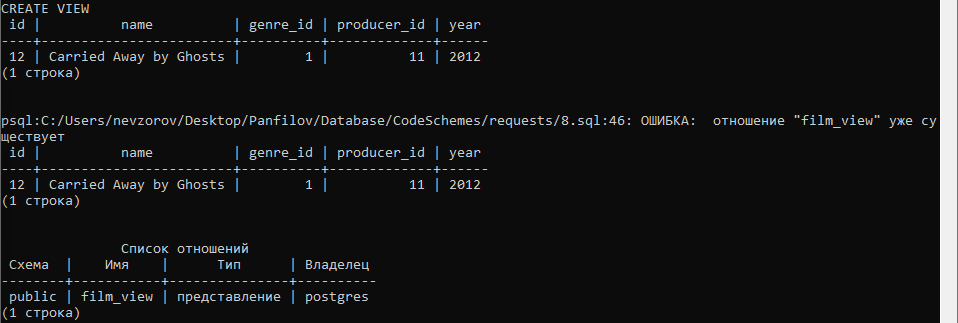
CREATE VIEW film\_view AS

    SELECT \*

    FROM films

    WHERE year = '2018'

       OR year = '2020';



[9) Создание функции](#_Toc91305843)

CREATE FUNCTION mean(x float, y float) RETURNS INTEGER AS $$

     SELECT x / y;

$$ LANGUAGE SQL;

SELECT mean((SELECT SUM(income) FROM incomes),

            (SELECT COUNT(income) FROM incomes)) AS mean\_mark;



[10) Создание триггера](#_Toc91305844)

-- создание функции

CREATE FUNCTION films\_stamp() RETURNS trigger AS $films\_stamp$

    BEGIN

        IF NEW.name IS NULL THEN

            RAISE EXCEPTION 'name cannot be null';

        END IF;

        IF NEW.genre\_id IS NULL THEN

            RAISE EXCEPTION '% genre\_id cannot be null', NEW.genre\_id;

        END IF;

        IF NEW.producer\_id IS NULL THEN

            RAISE EXCEPTION '% producer\_id cannot be null', NEW.producer\_id;

        END IF;

        IF NEW.year IS NULL THEN

            RAISE EXCEPTION '% year cannot be null', NEW.year;

        END IF;

        RETURN NEW;

    END;

$films\_stamp$ LANGUAGE plpgsql;

-- создание триггера

CREATE TRIGGER films\_stamp BEFORE INSERT OR UPDATE ON films

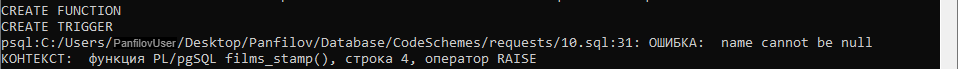
    FOR EACH ROW EXECUTE FUNCTION films\_stamp();

-- проверка неправильными данными

INSERT INTO films (name, genre\_id, producer\_id, year)

VALUES

  (NULL, 4, 2, '1950');



[11). Создание процедуры](#_Toc91305845)

-- удалние через процедуру

CREATE PROCEDURE clean\_emp() AS '

    DELETE FROM incomes

           WHERE income < 200000;

' LANGUAGE SQL;

CALL clean\_emp();

DROP PROCEDURE clean\_emp;



[12) Шифрование столбцов](#_Toc91305846)

CREATE EXTENSION IF NOT EXISTS pgcrypto;

CREATE TABLE income\_table (

  id SERIAL PRIMARY KEY,

  theater\_id VARCHAR(150) NOT NULL,

  income VARCHAR(150)  NOT NULL

);

INSERT INTO income\_table (theater\_id,income)

VALUES

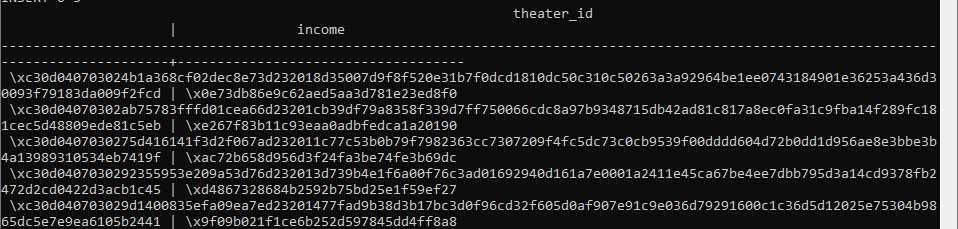
  (pgp\_sym\_encrypt('1','incomeshifr'), encrypt('161347','incomeshifr', 'aes')),

  (pgp\_sym\_encrypt('2','incomeshifr'), encrypt('871968','incomeshifr', 'aes')),

  (pgp\_sym\_encrypt('3','incomeshifr'), encrypt('400827','incomeshifr', 'aes')),

  (pgp\_sym\_encrypt('4','incomeshifr'), encrypt('563866','incomeshifr', 'aes')),

  (pgp\_sym\_encrypt('5','incomeshifr'), encrypt('703793','incomeshifr', 'aes'));



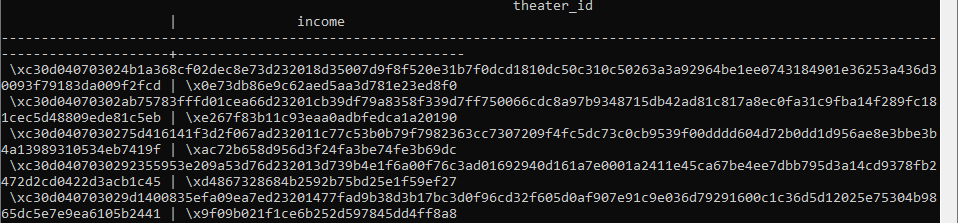
13) Расшифровка столбцов

-- введем неправильый пароль

SELECT  pgp\_sym\_decrypt(theater\_id::bytea, 'dsfgsdfg'),

        convert\_from(decrypt(income::bytea, 'aserthaef', 'aes'), 'SQL\_ASCII')

FROM income\_table;



-- прочитаем расшифрованные пароли

SELECT  pgp\_sym\_decrypt(theater\_id::bytea, 'incomeshifr'),

        convert\_from(decrypt(income::bytea, 'incomeshifr', 'aes'), 'SQL\_ASCII')

FROM income\_table;



14) Создание объектных типов данных

CREATE TYPE estimation AS ENUM ('Horror', 'Drama', 'History');

CREATE TABLE estimation\_table (

    id SERIAL,

    name TEXT,

    est estimation

);

INSERT INTO estimation\_table (id, name, est)

VALUES

    (1, 'Vyasov Street', 'Horror'),

    (2, 'Green Mile', 'Drama'),

    (3, 'Wofl from Street', 'History');

    (4, 'Seavales on earth', 'Documental');



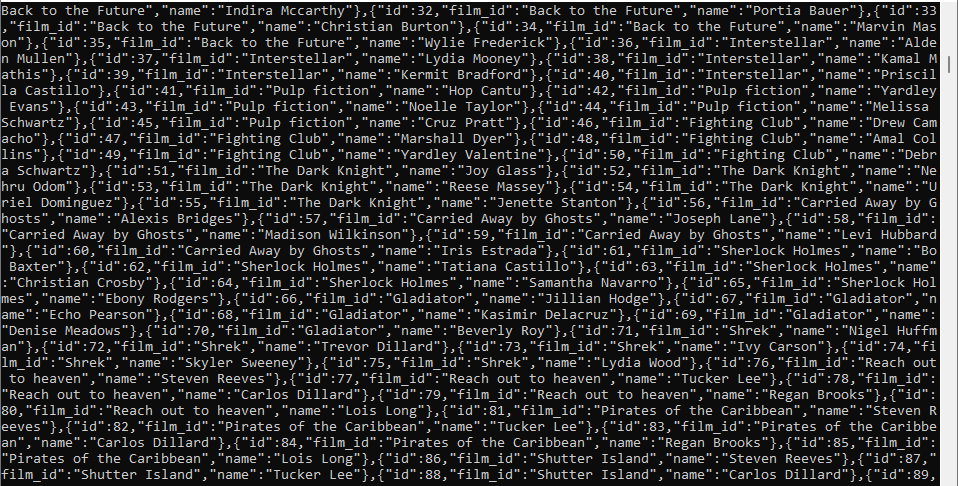
16) Запрос на перевод в формат JSON

SELECT array\_to\_json(array\_agg(row\_to\_json (ac))) FROM (

    SELECT \*

    FROM actors

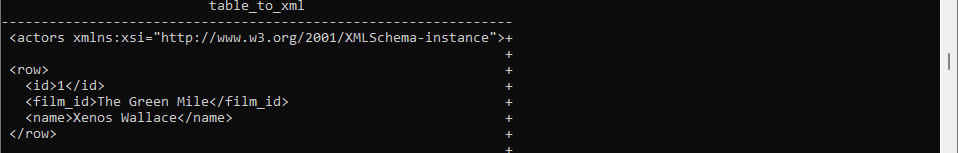
) ac;



17) Запрос на перевод в формат XML

-- таблица в XML

SELECT table\_to\_xml('actors', true, false, '');



-- query запрос в XML

SELECT query\_to\_xml('SELECT \* FROM actors', true, false, '');

