Правительство Российской Федерации
Федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский университет «Высшая школа экономики»

Факультет компьютерных наук
Образовательная программа бакалавриата 01.03.02 «Прикладная математика и информатика»

ОТЧЕТ
Финальная работа : проектирование базы данных

на факультете компьютерных наук НИУ ВШЭ

Выполнил:
Студент группы БПМИ191 _____ А.А. Городилова
Подпись И.О.Фамилия

26.12.2021
Дата

Преподаватель курса
Департамент больших данных и информационного поиска, кандидат наук
(подразделение ФКН, должность)
Моргунов Евгений Павлович
(ФИО руководителя практики)

Дата _____ _____ _____
Оценка Подпись

Москва 2021

Содержание

1 Описание предметной области

Предметной областью моей базы данных являются музыкальные исполнители.

В современном мире существует огромное число артистов, каждый из которых записывает собственные музыкальные альбомы и видеоклипы, дает концерты в разных странах и, может даже, снимается в кино.

Все имеющиеся популярные ресурсы, например, стриминговые площадки (spotify или apple music, где в основном информация лишь про треки) или википедия, странички которой может модифицировать любой пользователь, собирают информацию лишь частично. Поэтому было принято решение создать единую базу данных, которая бы собрала в единое целое разнообразную информацию и позволила бы делать запросы для получения данных, которые невозможно также быстро и удобно получить ни на одном из имеющихся популярных ресурсов.

Информация, которая будет храниться в базе данных:

1. музыкальные исполнители
2. лейблы звукозаписи
3. треки
4. награды и музыкальные премии
5. видеоклипы
6. музыкальные альбомы
7. фильмография (опционально)
8. мировые туры

Вся эта хранящаяся информация позволит находить треки конкретных исполнителей, смотреть, какие туры и в каких странах были в прошлом у артиста, сравнивать полученные награды на различных премиях. Все данные в базе можно будет модифицировать, а также добавлять новые.

2 Требования к базе данных

2.1 Требования к данным и транзакциям

2.1.1 Требования к данным:

1. музыкальные исполнители

   должна содержаться следующая информация: имя (*text*), дата рождения (*date*), страна (*text*), тэги (жанры исполняемой музыки) (*text[]*), лейбл звукозаписи (*text*)

2. лейблы звукозаписи

   название (*text*), компания (*text*), страна (*text*), год образования (*int*)

3. треки и видеоклипы

   название (*text*), исполнитель (*text*), год создания (*int*), жанр (*text*), альбом (*text*)

4. награды

   исполнитель (*text*), название награды (*text*), год (*int*)

5. музыкальные альбомы

   исполнитель (*text*), название альбома (*text*), число треков (*int*), дата (*date*)

6. фильмография

   исполнитель (*text*), тип (фильм / сериал), название (*text*), год (*int*)

7. мировые туры

   исполнитель (*text*), название тура (*text*), год (*int*), страны-города (*json*)

2.1.2 Требования к транзакциям:

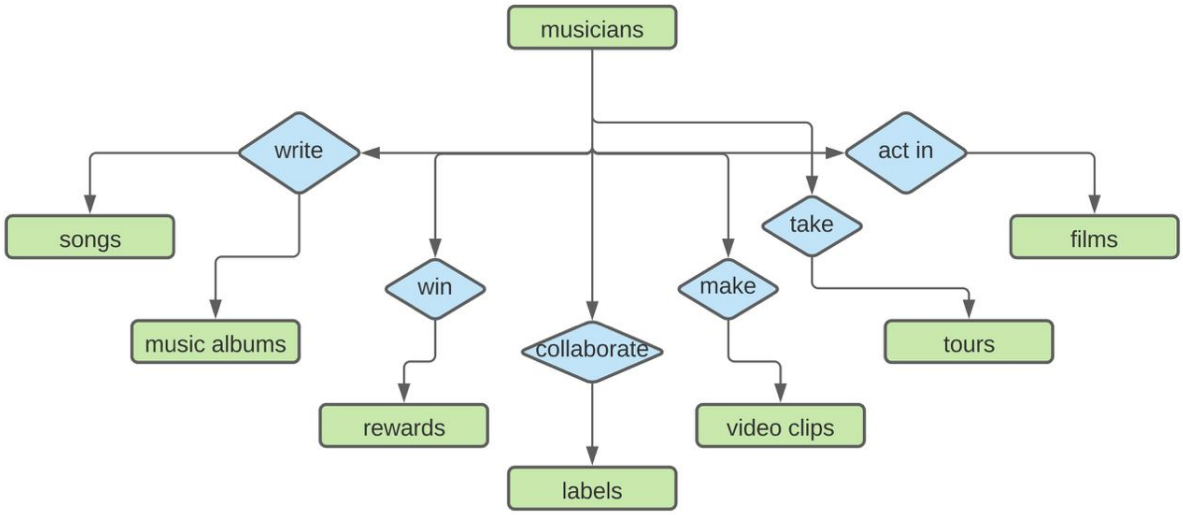
1. ввод данных
2. обновление / удаление данных
3. запросы к данным

2.2 Типичные запросы к базе данных

1. вывести список музыкальных артистов, которые также снимались в кино / сериалах
2. перечислить артистов, у которых 5 и более музыкальных альбомов
3. вывести лейбл звукозаписи с самой большой клиентской базой
4. перечислить исполнителей, которые получили заданную награду / премию

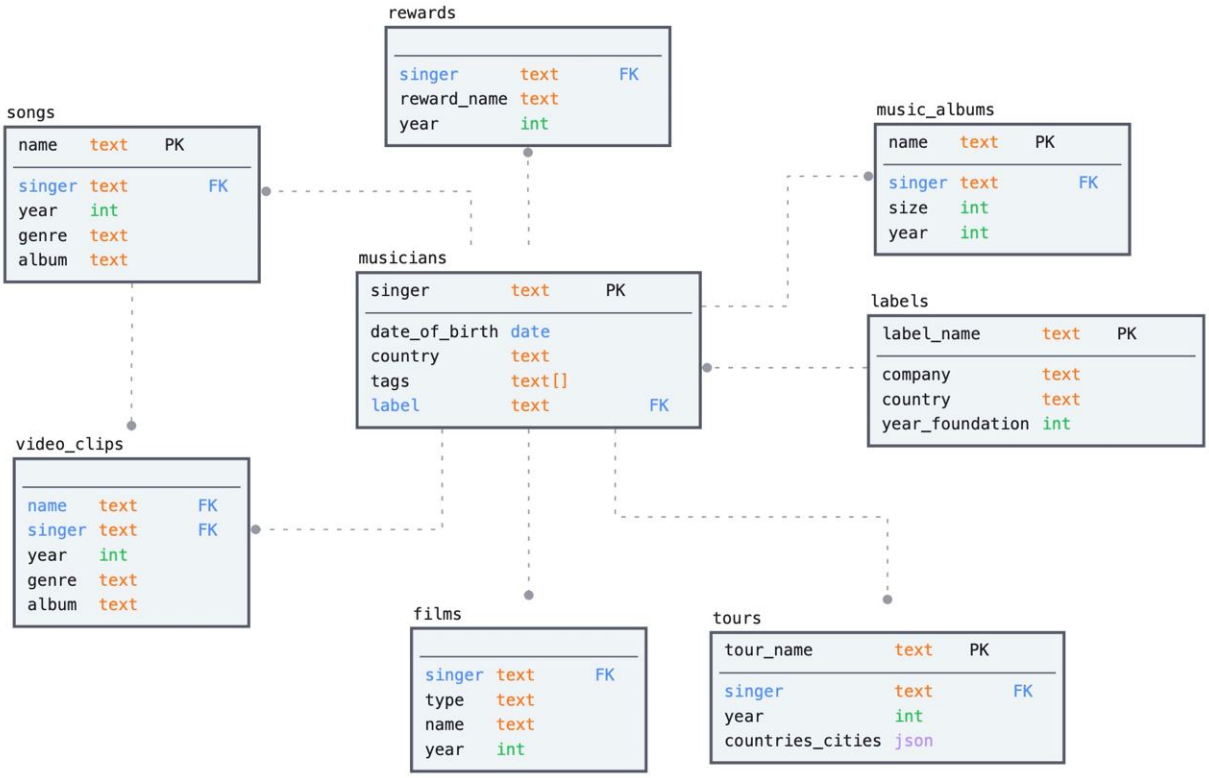
3 Концептуальная модель

Создадим концептуальную модель:



4 Логическая модель

Создадим логическую модель базы данных:



5 Физическая схема

5.1 Создание базы и таблиц

Создадим базу данных под названием musicians:

```
anastasiagorodilova=# create database musicians;
CREATE DATABASE
anastasiagorodilova=# \c musicians
You are now connected to database "musicians" as user "anastasiagorodilova".
```

Создадим необходимые таблицы и наложим нужные нам ограничения:

```
musicians=# CREATE TABLE labels
musicians-# ( label_name text  NOT NULL,
musicians(#   company text,
musicians(#   country text,
musicians(#   year_foundation int,
musicians(#   PRIMARY KEY ( label_name )
musicians(# );
CREATE TABLE
musicians=# CREATE TABLE musicians
musicians-# ( singer text  NOT NULL,
musicians(#   date_of_birth date,
musicians(#   country text,
musicians(#   tags text[],
musicians(#   label text DEFAULT '',
musicians(#   PRIMARY KEY ( singer ),
musicians(#   FOREIGN KEY ( label )
musicians(#     REFERENCES labels ( label_name )
musicians(#     ON DELETE SET DEFAULT
musicians(#
musicians(# );
CREATE TABLE
musicians=# CREATE TABLE songs
musicians-# ( name text  NOT NULL,
musicians(#   singer text NOT NULL,
musicians(#   year int,
musicians(#   genre text,
musicians(#   album text,
musicians(#   PRIMARY KEY ( name ),
musicians(#   FOREIGN KEY ( singer )
musicians(#     REFERENCES musicians ( singer )
musicians(#     ON DELETE CASCADE
musicians(# );
CREATE TABLE
musicians=# CREATE TABLE video_clips
musicians-# ( name text  NOT NULL,
musicians(#   singer text NOT NULL,
musicians(#   year int,
musicians(#   genre text,
musicians(#   album text,
musicians(#   FOREIGN KEY ( name )
musicians(#     REFERENCES songs ( name )
musicians(#     ON DELETE CASCADE,
musicians(#   FOREIGN KEY ( singer )
musicians(#     REFERENCES musicians ( singer )
musicians(#     ON DELETE CASCADE
musicians(# );
CREATE TABLE
musicians=#
```

```
musicians=# CREATE TABLE films
musicians-# ( singer text NOT NULL,
musicians(#   type text CHECK ( type = 'serial' OR type = 'film'),
musicians(#   name text NOT NULL,
musicians(#   year int,
musicians(#   FOREIGN KEY ( singer )
musicians(#     REFERENCES musicians ( singer )
musicians(#     ON DELETE CASCADE
musicians(# );
CREATE TABLE
musicians=# CREATE TABLE tours
musicians-# ( tour_name text  NOT NULL,
musicians(#   singer text NOT NULL,
musicians(#   year int,
musicians(#   countries_cities JSON,
musicians(#   PRIMARY KEY ( tour_name ),
musicians(#   FOREIGN KEY ( singer )
musicians(#     REFERENCES musicians ( singer )
musicians(#     ON DELETE CASCADE
musicians(# );
CREATE TABLE
musicians=# CREATE TABLE rewards
musicians-# ( singer text NOT NULL,
musicians(#   reward_name text,
musicians(#   year int,
musicians(#   FOREIGN KEY ( singer )
musicians(#     REFERENCES musicians ( singer )
musicians(#     ON DELETE CASCADE
musicians(# );
CREATE TABLE
musicians=# CREATE TABLE music_albums
musicians-# ( name text NOT NULL,
musicians(#   singer text NOT NULL,
musicians(#   size int,
musicians(#   year int,
musicians(#   PRIMARY KEY ( name ),
musicians(#   FOREIGN KEY ( singer )
musicians(#     REFERENCES musicians ( singer )
musicians(#     ON DELETE CASCADE
musicians(# );
CREATE TABLE
musicians=#
```

5.2 Триггеры

- Триггер, который добавляет музыкального исполнителя в табличку musicians, если добавляем что-то в таблицу songs:

```
musicians=# CREATE OR REPLACE FUNCTION upd_musicians()
musicians-#     RETURNS TRIGGER
musicians-#     AS
musicians-# $$
musicians$# BEGIN INSERT INTO musicians ( singer )
musicians$# VALUES ( NEW.singer )
musicians$# ON CONFLICT ( singer ) DO NOTHING;
musicians$# RETURN NEW;
musicians$# END;
musicians$# $$
musicians-# LANGUAGE plpgsql;
CREATE FUNCTION
musicians=# CREATE TRIGGER add_musicians
musicians-# BEFORE INSERT ON songs FOR EACH ROW
musicians-# EXECUTE PROCEDURE upd_musicians();
CREATE TRIGGER
```

- Триггер, который добавляет лейбл звукозаписи в табличку labels, если добавляем нового музыканта в таблицу musicians:

```
musicians=# CREATE OR REPLACE FUNCTION upd_labels()
musicians-#     RETURNS TRIGGER
musicians-#     AS
musicians-# $$
musicians$# BEGIN INSERT INTO labels ( label_name )
musicians$# VALUES ( NEW.label )
musicians$# ON CONFLICT ( label_name ) DO NOTHING;
musicians$# RETURN NEW;
musicians$# END;
musicians$# $$
musicians-# LANGUAGE plpgsql;
CREATE FUNCTION
musicians=# CREATE TRIGGER add_labels
musicians-# BEFORE INSERT ON musicians FOR EACH ROW
musicians-# EXECUTE PROCEDURE upd_labels();
CREATE TRIGGER
musicians=#
```

5.3 Хранимые функции

- Функция, которая говорит сколько треков заданного исполнителя есть в таблице songs:

```
musicians=# CREATE OR REPLACE FUNCTION count_songs( singer_ text )
musicians-# RETURNS INTEGER AS
musicians-# $$
musicians$# DECLARE count_ int;
musicians$# BEGIN SELECT SUM(1) INTO count_ FROM songs
musicians$# WHERE singer = singer_;
musicians$# RETURN count_;
musicians$# END;
musicians$# $$
musicians-# LANGUAGE plpgsql;
CREATE FUNCTION
```

- Функция, которая показывает, сколько человек сотрудничает с заданным лейблом звукозаписи:

```
musicians=# CREATE OR REPLACE FUNCTION customer_base( label_ text )
musicians-# RETURNS INTEGER AS
musicians-# $$
musicians$# DECLARE count_ int;
musicians$# BEGIN SELECT SUM(1) INTO count_ FROM musicians
musicians$# WHERE label = label_;
musicians$# RETURN count_;
musicians$# END;
musicians$# $$
musicians-# LANGUAGE plpgsql;
CREATE FUNCTION
```

5.4 Ввод данных

Введем немного записей в базу данных, чтобы продемонстрировать типичные запросы, а также проверить уже созданные триггеры и хранимые функции:

Изначально все таблички пустые.

Добавим трех исполнителей в таблицу musicians и проверим, работает ли триггер, добавляющий лейблы по добавленным артистам в соответствующую таблицу labels:

```
musicians=# INSERT INTO musicians ( singer, country, label )
musicians-# VALUES ( 'Miley Cyrus', 'USA', 'Columbia Records' ),
musicians-#        ( 'Danna Paola', 'Spain', 'Universal Music Group'),
musicians-#        ( 'Adele', 'UK', 'Columbia Records');
INSERT 0 3
musicians=# select * from musicians;
   singer    | date_of_birth | country | tags |         label
-------------+---------------+---------+------+-----------------------
 Miley Cyrus |               | USA     |      | Columbia Records
 Danna Paola |               | Spain   |      | Universal Music Group
 Adele       |               | UK      |      | Columbia Records
(3 rows)

musicians=# select * from labels;
      label_name       | company | country | year_foundation
-----------------------+---------+---------+-----------------
 Columbia Records      |         |         |
 Universal Music Group |         |         |
(2 rows)

musicians=#
```

Теперь добавим несколько песен в таблицу songs и проверим работу триггера по добавлению исполнителей:

```
musicians=# INSERT INTO songs ( name, singer, year )
musicians-# VALUES ( 'Robot', 'Miley Cyrus', 2010 ),
musicians-#        ( 'Contigo', 'Danna Paola', 2020 ),
musicians-#        ( 'CORALINE', 'Maneskin', 2021);
INSERT 0 3
musicians=# select * from songs;
   name   |   singer    | year | genre | album
----------+-------------+------+-------+-------
 Robot    | Miley Cyrus | 2010 |       |
 Contigo  | Danna Paola | 2020 |       |
 CORALINE | Maneskin    | 2021 |       |
(3 rows)

musicians=# select * from musicians;
   singer    | date_of_birth | country | tags |         label
-------------+---------------+---------+------+-----------------------
 Miley Cyrus |               | USA     |      | Columbia Records
 Danna Paola |               | Spain   |      | Universal Music Group
 Adele       |               | UK      |      | Columbia Records
 Maneskin    |               |         |      |
(4 rows)

musicians=#
```

Теперь добавим еще данных в таблицы:

```
musicians=# INSERT INTO songs ( name, singer, year )
musicians-# VALUES ( 'Malibu', 'Miley Cyrus', 2017 ),
musicians-#        ( 'Polo a Tierra', 'Danna Paola', 2020 ),
musicians-#        ( 'Hallucinate', 'Dua Lipa', 2020 ),
musicians-#        ( 'Drive', 'Miley Cyrus', 2013 ),
musicians-#        ( 'Liberty Walk', 'Miley Cyrus', 2010 ),
musicians-#        ( '1 Sun', 'Miley Cyrus', 2015 ),
musicians-#        ( 'Plastic Hearts', 'Miley Cyrus', 2020 ),
musicians-#        ( 'Unstoppable', 'Sia', 2016 ),
musicians-#        ( 'Big Girls Cry', 'Sia', 2014 ),
musicians-#        ( 'Levitating', 'Dua Lipa', 2020 ),
musicians-#        ( 'Final Feliz', 'Danna Paola', 2020 ),
musicians-#        ( 'Beggin', 'Maneskin', 2017 ),
musicians-#        ( 'Future Nostalgia', 'Dua Lipa', 2020 ),
musicians-#        ( 'Someone Like You', 'Adele', 2011 );
INSERT 0 14
```

```
musicians=# INSERT INTO video_clips ( name, singer, year )
musicians-# VALUES ( 'Malibu', 'Miley Cyrus', 2017 ),
musicians-#        ( 'Contigo', 'Danna Paola', 2020 );
INSERT 0 2
musicians=#
```

```
musicians=# INSERT INTO rewards ( singer, reward_name, year )
musicians-# VALUES ( 'Miley Cyrus', 'Billboard', 2014 ),
musicians-#        ( 'Miley Cyrus', 'iHeartRadio Music Awards', 2014 ),
musicians-#        ( 'Dua Lipa', 'Grammy Awards', 2021 ),
musicians-#        ( 'Adele', 'American Music Awards', 2016 ),
musicians-#        ( 'Maneskin', 'Wind Music Awards',  2018),
musicians-#        ( 'Miley Cyrus', 'MTV Europe Music Awards', 2013 );
INSERT 0 6
musicians=#
```

```
musicians=# INSERT INTO music_albums ( name, singer )
musicians-# VALUES ( '21', 'Adele' ),
musicians-#        ( 'SIE7E', 'Danna Paola' ),
musicians-#        ( 'Future Nostalgia', 'Dua Lipa' ),
musicians-#        ( 'Bangerz', 'Miley Cyrus' ),
musicians-#        ( 'SHE IS COMING', 'Miley Cyrus' ),
musicians-#        ( 'This Is Acting', 'Sia' ),
musicians-#        ( 'Chosen', 'Maneskin' ),
musicians-#        ( 'Younger Now', 'Miley Cyrus' ),
musicians-#        ( '1000 Forms of Fear', 'Sia' );
INSERT 0 9
musicians=#
```

```
musicians=# INSERT INTO films ( singer, type, name, year )
musicians-# VALUES ( 'Miley Cyrus', 'film', 'LOL', 2012 ),
musicians-#        ( 'Miley Cyrus', 'serial', 'Crisis in Six Scenes', 2016 ),
musicians-#        ( 'Danna Paola', 'serial', 'Elite', 2019 ),
musicians-#        ( 'Danna Paola', 'serial', 'La Doña', 2016 ),
musicians-#        ( 'Danna Paola', 'film', 'Home', 2015 );
INSERT 0 5
musicians=#
```

```
musicians=# INSERT INTO tours ( tour_name, singer, year, countries_cities )
musicians-# VALUES ( 'Bangerz Tour', 'Miley Cyrus', 2013, '{ "USA": { "city" : "New-York" } }
'::json ),
musicians-#        ( 'Maneskin Tour', 'Maneskin', 2021, '{ "Russia": { "city" : "Moscow" } }']
::json );
INSERT 0 2
musicians=#
```

Проверим, как работают наши хранимые функции:

```
musicians=# select * from count_songs( 'Miley Cyrus' );
 count_songs
-------------
           6
(1 row)

musicians=# select * from songs;
       name       |   singer    | year | genre | album
------------------+-------------+------+-------+-------
 Robot            | Miley Cyrus | 2010 |       |
 Contigo          | Danna Paola | 2020 |       |
 CORALINE         | Maneskin    | 2021 |       |
 Malibu           | Miley Cyrus | 2017 |       |
 Polo a Tierra    | Danna Paola | 2020 |       |
 Hallucinate      | Dua Lipa    | 2020 |       |
 Drive            | Miley Cyrus | 2013 |       |
 Liberty Walk     | Miley Cyrus | 2010 |       |
 1 Sun            | Miley Cyrus | 2015 |       |
 Plastic Hearts   | Miley Cyrus | 2020 |       |
 Unstoppable      | Sia         | 2016 |       |
 Big Girls Cry    | Sia         | 2014 |       |
 Levitating       | Dua Lipa    | 2020 |       |
 Final Feliz      | Danna Paola | 2020 |       |
 Beggin           | Maneskin    | 2017 |       |
 Future Nostalgia | Dua Lipa    | 2020 |       |
 Someone Like You | Adele       | 2011 |       |
(17 rows)

musicians=#
```

```
musicians=# select * from customer_base( 'Columbia Records' );
 customer_base
---------------
             2
(1 row)

musicians=# select * from musicians;
   singer    | date_of_birth | country | tags |         label
-------------+---------------+---------+------+-----------------------
 Miley Cyrus |               | USA     |      | Columbia Records
 Danna Paola |               | Spain   |      | Universal Music Group
 Adele       |               | UK      |      | Columbia Records
 Maneskin    |               |         |      |
 Dua Lipa    |               |         |      |
 Sia         |               |         |      |
(6 rows)

musicians=#
```

5.5 Демонстрация типичных запросов к базе данных

5.5.1 Подзапросы

Лейбл звукозаписи с самой большой клиентской базой:

```
musicians=# SELECT customer_base( label_name ),
musicians-#        label_name
musicians-# FROM labels
musicians-# WHERE label_name <> ''
musicians-# ORDER BY customer_base DESC
musicians-# LIMIT 1;
 customer_base |    label_name
---------------+------------------
             2 | Columbia Records
(1 row)

musicians=#
```

Список музыкантов, которые также снимались в кино / сериалах:

```
musicians=# SELECT singer
musicians-#     FROM musicians
musicians-#     INTERSECT
musicians-# SELECT singer
musicians-#     FROM films
musicians-# ORDER BY singer;
   singer
-------------
 Danna Paola
 Miley Cyrus
(2 rows)

musicians=#
```

5.5.2 CTE

Сколько суммарно треков в таблице songs у каждого из исполнителей:

```
musicians=# WITH singer_ AS
musicians-# ( SELECT singer, country FROM musicians ORDER BY singer ),
musicians-# count_songs_ AS
musicians-# ( SELECT count_songs( singer ), singer FROM musicians)
musicians-# SELECT singer_.singer, singer_.country, count_songs_.count_songs
musicians-# FROM singer_ JOIN count_songs_
musicians-# ON singer_.singer = count_songs_.singer;
   singer    | country | count_songs
-------------+---------+-------------
 Adele       | UK      |           1
 Danna Paola | Spain   |           3
 Dua Lipa    |         |           3
 Maneskin    |         |           2
 Miley Cyrus | USA     |           6
 Sia         |         |           2
(6 rows)

musicians=#
```

5.5.3 Window Functions

Оконная функция, которая позволяет сравнить год выпуска песни со значениями различных статистик от величины (минимальный год выпуска (дебютный), в среднем выпускал песни в окрестности этого года, год последней песни) :

```
musicians=# SELECT singer, year, min(year) OVER w,
musicians-# avg(year) OVER w, max(year) OVER w
musicians-# FROM songs
musicians-# WINDOW w AS (PARTITION BY singer);
   singer    | year | min  |          avg          | max
-------------+------+------+-----------------------+------
 Adele       | 2011 | 2011 | 2011.0000000000000000 | 2011
 Danna Paola | 2020 | 2020 | 2020.0000000000000000 | 2020
 Danna Paola | 2020 | 2020 | 2020.0000000000000000 | 2020
 Danna Paola | 2020 | 2020 | 2020.0000000000000000 | 2020
 Dua Lipa    | 2020 | 2020 | 2020.0000000000000000 | 2020
 Dua Lipa    | 2020 | 2020 | 2020.0000000000000000 | 2020
 Dua Lipa    | 2020 | 2020 | 2020.0000000000000000 | 2020
 Maneskin    | 2017 | 2017 | 2019.0000000000000000 | 2021
 Maneskin    | 2021 | 2017 | 2019.0000000000000000 | 2021
 Miley Cyrus | 2015 | 2010 | 2014.1666666666666667 | 2020
 Miley Cyrus | 2017 | 2010 | 2014.1666666666666667 | 2020
 Miley Cyrus | 2013 | 2010 | 2014.1666666666666667 | 2020
 Miley Cyrus | 2010 | 2010 | 2014.1666666666666667 | 2020
 Miley Cyrus | 2010 | 2010 | 2014.1666666666666667 | 2020
 Miley Cyrus | 2020 | 2010 | 2014.1666666666666667 | 2020
 Sia         | 2014 | 2014 | 2015.0000000000000000 | 2016
 Sia         | 2016 | 2014 | 2015.0000000000000000 | 2016
(17 rows)

musicians=#
```

6 Создание резервной копии базы данных

Создадим резервную копию с помощью утилиты pg_dump:

```
MacBook-Pro-determinant:Desktop anastasiagorodilova$ pg_dump -O musicians > myDataBase.sql
MacBook-Pro-determinant:Desktop anastasiagorodilova$
```