



Project. Parts 0-6

Music Platform “Spotify”


Students: Yedil Alinur, Taldybayev Batkyrhan

Tutor: Kozina Lyudmila

Deploying the database on a server

Renting a server on hosting provider

Before we start the project parts, we want to rent a server on which we will deploy the database, because it will be uncomfortable to work on a local database and send data and SQL queries to each other. So, I entered to billing.firstbyte.com to rent a server. The cheapest is good, and that's why this hosting is good for it:

MSK-KVM-SSD
MSK-KVM-SSD
Виртуализация: KVM, Процессор: 1 ядро, Память: 1 GB,
Диск: 20 GB SSD
[Развернуть](#)

Лицензия на панель управления	Дисковое пространство
Без панели управления	20000 МБ
Оперативная память	Количество процессоров
1024 МБ	1 шт.
Публичные IPv4-адреса	IPv6-адреса
1 шт.	1 шт.

[Показать ещё 2](#) ▾

189.00 RUB/месяц ▾ ⓘ

[Заказать](#)

This server is not bad. For this cost we get 1 GB of RAM, 1 Core of CPU, IP-address and 20 GB of SSD.

The next step is to enter into the server using root credentials and setup it for logging by using public and private keys. At first it will solve the security problem, at

second it's easier to log in by using my own password and private key. Thanks to WSL on my system, I can log in to server without other programs like PuTTY, so I do this directly from Linux CLI. But at first, I need to create public and private keys:

```
khan@DESKTOP-6TSCB62: ~  
khan@DESKTOP-6TSCB62:~$ ssh-keygen -t ed25519 -C "34613@iitu.edu.kz" -f ~/.ssh/id_ed25519  
Generating public/private ed25519 key pair.  
Created directory '/home/khan/.ssh'.  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in /home/khan/.ssh/id_ed25519  
Your public key has been saved in /home/khan/.ssh/id_ed25519.pub  
The key fingerprint is:  
SHA256:P80L5IOrgtdei2fnK2z0NBK4j7jKgPCSUCn09+d2mFI 34613@iitu.edu.kz  
The key's randomart image is:  
+--[ED25519 256]--+  
|  
|. .  
|. o. . .  
| o . o .  
|o      oSE.  
|+o      *==o  
|= .. o *o0*oo  
|+. + ooXo++ .  
| oo.++=o+o..  
+-----[SHA256]-----+  
khan@DESKTOP-6TSCB62:~$
```

Keys were created successfully. Next step is to log in to the server, creating user for me and setup the keys:

```
root@vm3613107: ~  
khan@DESKTOP-6TSCB62:~/.ssh$ ssh root@95.81.121.145  
The authenticity of host '95.81.121.145 (95.81.121.145)' can't be established.  
ED25519 key fingerprint is SHA256:5gUJ91wvyFWgDhsh6Fgk+I7s7gVja4SEjGV0us+62JI.  
This key is not known by any other names.  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added '95.81.121.145' (ED25519) to the list of known hosts.  
root@95.81.121.145's password:  
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.4.0-192-generic x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:        https://ubuntu.com/pro  
Last login: Mon Aug 12 15:52:26 2024 from 151.0.51.108  
root@vm3613107:~#
```

```

Выбрать root@vm3613107: ~
khan@DESKTOP-6TSCB62:~/.ssh$ ssh root@95.81.121.145
The authenticity of host '95.81.121.145 (95.81.121.145)' can't be established.
ED25519 key fingerprint is SHA256:5gUJ91wvyFWgDhsh6Fgk+I7s7gVja4SEjGV0us+62JI.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '95.81.121.145' (ED25519) to the list of known hosts.
root@95.81.121.145's password:
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.4.0-192-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro
Last login: Mon Aug 12 15:52:26 2024 from 151.0.51.108
root@vm3613107:~# adduser --disabled-password --gecos "" khan
Adding user `khan' ...
Adding new group `khan' (1000) ...
Adding new user `khan' (1000) with group `khan' ...
Creating home directory `/home/khan' ...
Copying files from `/etc/skel' ...
root@vm3613107:~# usermod -sG sudo khan
root@vm3613107:~#

```

```

root@vm3613107: ~
root@vm3613107:~# mkdir -p /home/khan/.ssh
root@vm3613107:~# echo "ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIH3+DYwUX3HI1+Fi0WRJO1FNCZDrXpH8JBm4VEm81vbZ 34613@iitu.edu.kz" >> /home/khan/.ssh/authorized_keys
root@vm3613107:~# chown -R khan:khan /home/khan/.ssh
root@vm3613107:~# chmod 700 /home/khan/.ssh
root@vm3613107:~# chmod 600 /home/khan/.ssh/authorized_keys
root@vm3613107:~#

```

Public key is implemented to my user, so I can access to the server by using my private key:

```

khan@vm3613107: ~
khan@DESKTOP-6TSCB62:~$ ssh -i .ssh/id_ed25519 khan@95.81.121.145
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.4.0-192-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro
New release '22.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

khan@vm3613107:~$

```

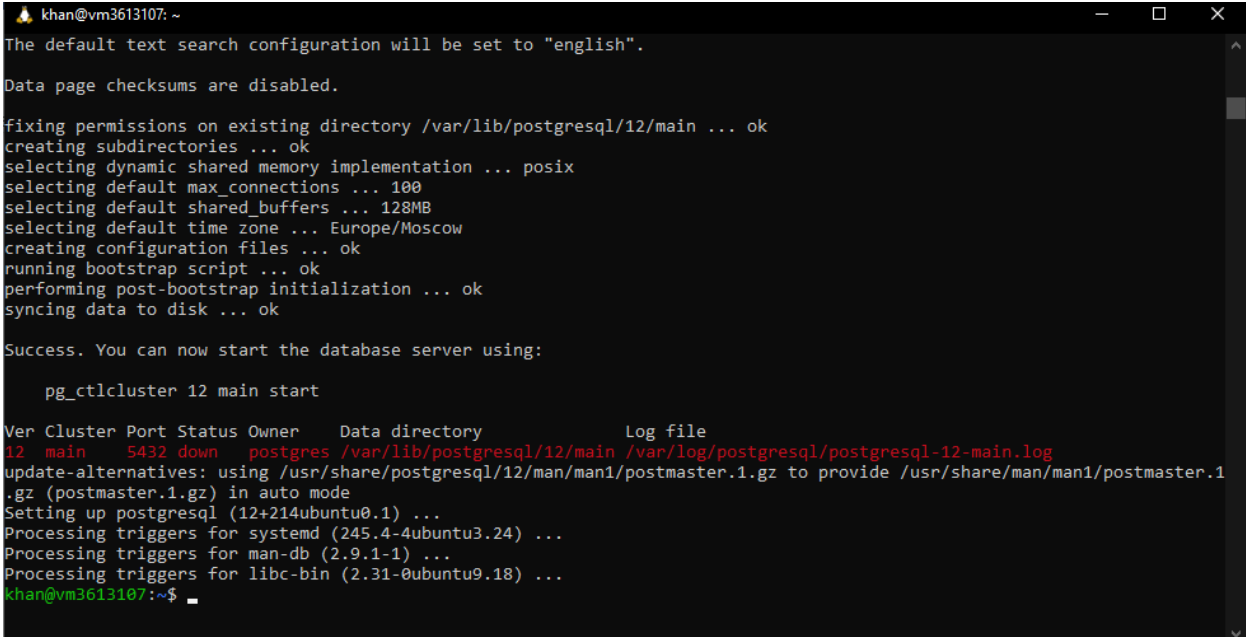
And it worked! So, the next step is to update packages and initialize PostgreSQL for work.

Setup the PostgreSQL

The commands I need to input are:

1. `sudo apt update && apt upgrade -y`
2. `sudo apt install postgresql`

That's enough to work with it.

A terminal window titled 'khan@vm3613107: ~' showing the output of the 'sudo apt install postgresql' command. The output includes messages about text search configuration, data page checksums, and the installation of PostgreSQL 12. It shows the creation of directories, selection of defaults (posix, 100 connections, 128MB buffers, Europe/Moscow time zone), and successful completion of the bootstrap script and initialization. The prompt 'Success. You can now start the database server using:' is followed by the command 'pg_ctlcluster 12 main start'. Below this, a table lists the installed cluster details, and the terminal ends with the prompt 'khan@vm3613107:~\$' and a cursor.

```
khan@vm3613107: ~
The default text search configuration will be set to "english".

Data page checksums are disabled.

fixing permissions on existing directory /var/lib/postgresql/12/main ... ok
creating subdirectories ... ok
selecting dynamic shared memory implementation ... posix
selecting default max_connections ... 100
selecting default shared_buffers ... 128MB
selecting default time zone ... Europe/Moscow
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... ok
syncing data to disk ... ok

Success. You can now start the database server using:

    pg_ctlcluster 12 main start

Ver Cluster Port Status Owner    Data directory          Log file
12  main     5432 down  postgres /var/lib/postgresql/12/main /var/log/postgresql/postgresql-12-main.log
update-alternatives: using /usr/share/postgresql/12/man/man1/postmaster.1.gz to provide /usr/share/man/man1/postmaster.1.gz (postmaster.1.gz) in auto mode
Setting up postgresql (12+214ubuntu0.1) ...
Processing triggers for systemd (245.4-4ubuntu3.24) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for libc-bin (2.31-0ubuntu9.18) ...
khan@vm3613107:~$
```

Screenshot says that PostgreSQL has been installed

But it's already started, because I checked it by systemctl and enabled for auto-start:

```
khan@vm3613107: ~  
khan@vm3613107:~$ sudo systemctl status postgresql  
* postgresql.service - PostgreSQL RDBMS  
   Loaded: loaded (/lib/systemd/system/postgresql.service; enabled; vendor preset: enabled)  
   Active: active (exited) since Wed 2025-10-15 20:00:56 MSK; 2min 5s ago  
     Main PID: 27108 (code=exited, status=0/SUCCESS)  
       Tasks: 0 (limit: 1084)  
      Memory: 0B  
     CGroup: /system.slice/postgresql.service  
  
Oct 15 20:00:56 vm3613107.firstbyte.club systemd[1]: Starting PostgreSQL RDBMS...  
Oct 15 20:00:56 vm3613107.firstbyte.club systemd[1]: Finished PostgreSQL RDBMS.  
khan@vm3613107:~$ sudo systemctl enable postgresql  
Synchronizing state of postgresql.service with SysV service script with /lib/systemd/systemd-sysv-install.  
Executing: /lib/systemd/systemd-sysv-install enable postgresql  
khan@vm3613107:~$ _
```

According to documentation, I need to enable all other computers to connect to database and then create a user.

```
khan@vm3613107: ~  
# The default values of these variables are driven from the -D command-line  
# option or PGDATA environment variable, represented here as ConfigDir.  
  
data_directory = '/var/lib/postgresql/12/main'      # use data in another directory  
# (change requires restart)  
hba_file = '/etc/postgresql/12/main/pg_hba.conf'    # host-based authentication file  
# (change requires restart)  
ident_file = '/etc/postgresql/12/main/pg_ident.conf' # ident configuration file  
# (change requires restart)  
  
# If external_pid_file is not explicitly set, no extra PID file is written.  
external_pid_file = '/var/run/postgresql/12-main.pid' # write an extra PID file  
# (change requires restart)  
  
#-----  
# CONNECTIONS AND AUTHENTICATION  
#-----  
  
# - Connection Settings -  
  
#listen_addresses = 'localhost'      # what IP address(es) to listen on;  
#                                     # comma-separated list of addresses;  
#                                     # defaults to 'localhost'; use '*' for all  
#                                     # (change requires restart)  
port = 5432                          # (change requires restart)  
max_connections = 100                # (change requires restart)  
#superuser_reserved_connections = 3  # (change requires restart)  
-- INSERT -- W10: Warning: Changing a readonly file
```

This line `#listen_addresses = 'localhost'` has to be changed to `*`, so me and Alinur can access to it from our computers.

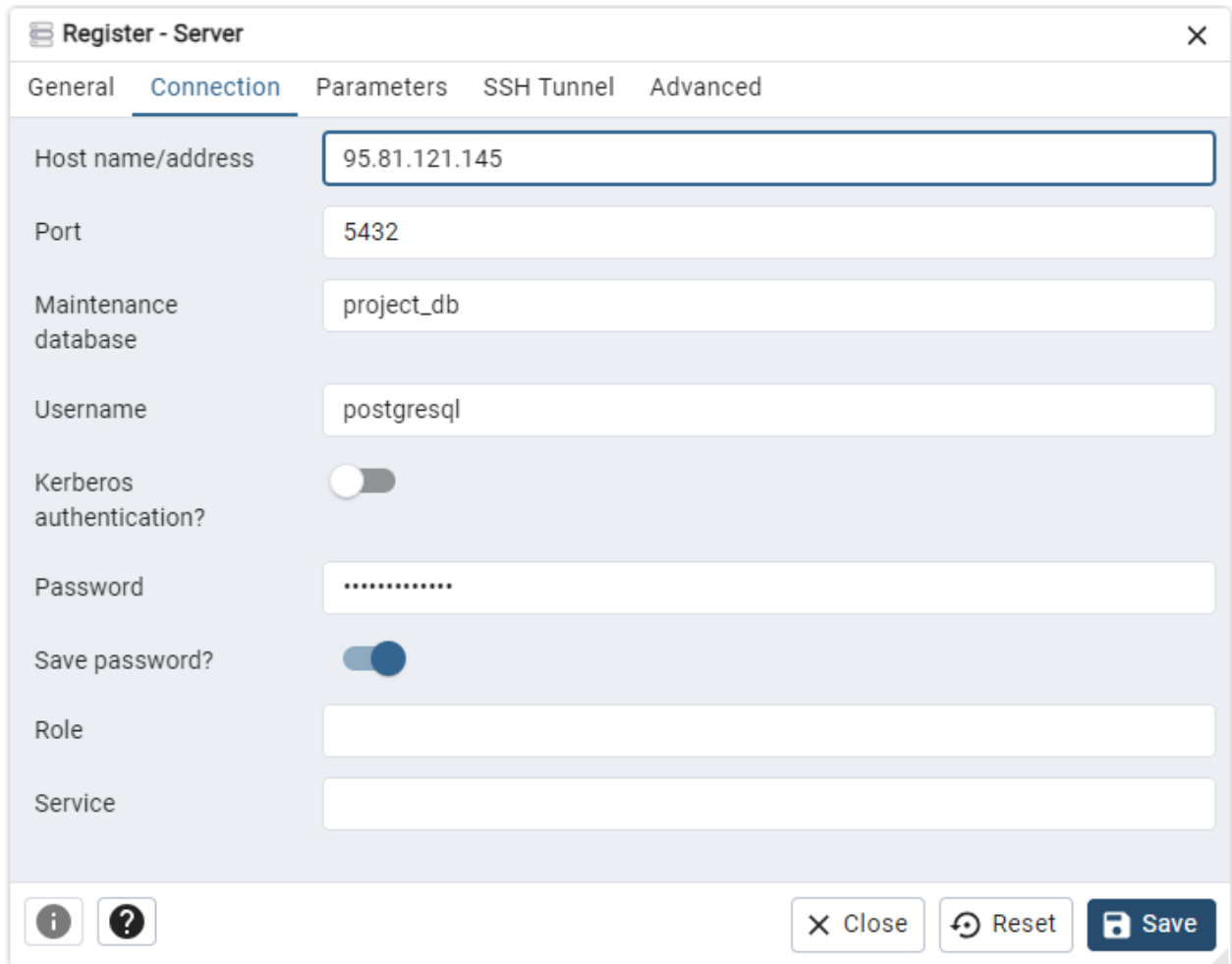
After that, it's time to create user:

```
khan@vm3613107: ~  
khan@vm3613107:~$ sudo vim /etc/postgresql/12/main/postgresql.conf  
khan@vm3613107:~$ sudo -u postgres psql -c "CREATE ROLE postgresql WITH LOGIN PASSWORD 'khanandalinur';"  
CREATE ROLE  
khan@vm3613107:~$ sudo -u postgres psql -c "CREATE DATABASE project_db OWNER postgresql;"  
CREATE DATABASE  
khan@vm3613107:~$ sudo -u postgres psql -c "GRANT ALL PRIVILEGES ON DATABASE project_db TO postgresql;"  
GRANT  
khan@vm3613107:~$ _
```

Thanks to `-c` I can directly in command write the unnecessary query: Created role `postgresql` with our custom password, created database named `project_db` for `postgresql` (our main user) and gave all privileges for `postgresql` to interact with database.

Connecting to the database

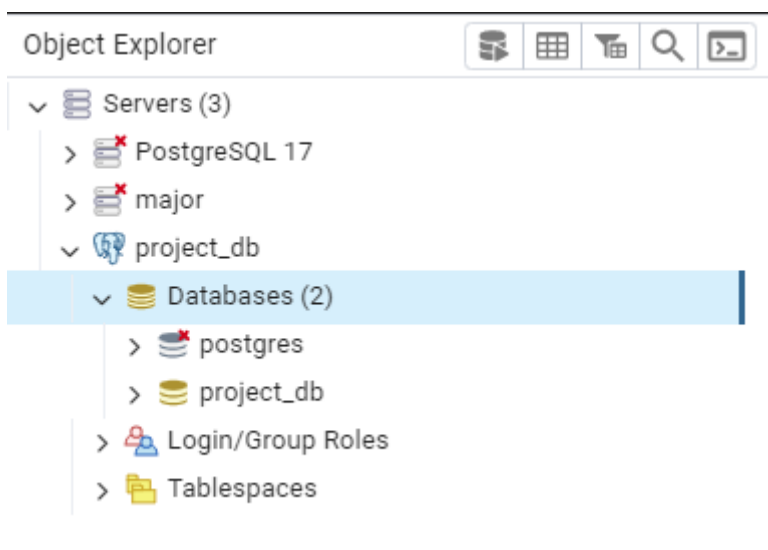
In pgAdmin I created new server “project” and entering credentials:



The image shows the 'Register - Server' dialog box in pgAdmin, specifically the 'Connection' tab. The dialog has a title bar with a close button (X) and a tab bar with 'General', 'Connection' (selected), 'Parameters', 'SSH Tunnel', and 'Advanced'. The 'Connection' tab contains the following fields and controls:

- Host name/address: 95.81.121.145
- Port: 5432
- Maintenance database: project_db
- Username: postgresql
- Kerberos authentication?: ☐
- Password:
- Save password?: ☒
- Role:
- Service:

At the bottom of the dialog, there are three buttons: 'Close' (with an X icon), 'Reset' (with a circular arrow icon), and 'Save' (with a floppy disk icon). There are also information (i) and help (?) icons on the left.



Perfect! Now I and Alinur can connect to the database from our computer via pgAdmin!

Part 0. PostgreSQL Database Implementation

We chose the database for Music Platform. We tried to make similar to *Spotify* database. Of course, it's not so similar and perfect like we didn't use the UUID for IDs, there is no columns such as "danceability", "energy", "loudness", "tempo" and so on, but the main idea exists.

There are tables:

1. Countries

country_id	VARCHAR(50)	PK	Unique id of country
name	VARCHAR(100)		Name of the country

2. Subscriptions

subscription_id	VARCHAR(50)	PK	Unique id
name	VARCHAR(50)		The name of subscription
Price	DECIMAL(6,2)		The cost of subscription

3. Users

user_id	VARCHAR(50)	PK	Unique id
name	VARCHAR(100)		Name of the user
email	VARCHAR(100)		Email (unique)
password	VARCHAR(255)		Hash of the password
subscription_id	VARCHAR(50)	FK	Subscription of the user

country_id	VARCHAR(50)	FK	The country of the user
registration_date	TIMESTAMP		Registration date of the user

4. Artists

artist_id	VARCHAR(50)	PK	Unique id
name	VARCHAR(150)		The name of the artist
bio	TEXT		Bio of the artist

5. Genres

genre_id	VARCHAR(50)	PK	Unique id
name	VARCHAR(100)		The name of the genre

6. Albums

album_id	VARCHAR(50)	PK	Unique id
title	VARCHAR(100)		The name of the album
release_date	DATE		The release date of the album
artist_id	VARCHAR(50)	FK	Author of the album

7. Tracks

track_id	VARCHAR(50)	PK	Unique id
title	VARCHAR(200)		Name of track
duration	INTERVAL		Track duration
artist_id	VARCHAR(50)	FK	Author of track
album_id	VARCHAR(50)	FK	Album of track
genre_id	VARCHAR(50)	FK	Genre of track

8. Playlists

playlist_id	VARCHAR(50)	PK	Unique id
title	VARCHAR(200)		Name of playlist
user_id	VARCHAR(50)	FK	Creator of playlist

9. Playlist_Track

playlist_id	VARCHAR(50)	PK, FK	ID of playlist
track_id	VARCHAR(50)	PK, FK	ID of track
position	INTEGER		Track position in playlist

10. Listening_Histories

history_id	VARCHAR(50)	PK	ID of the listening record
user_id	VARCHAR(50)	FK	ID of user
track_id	VARCHAR(50)	FK	ID of listened track
play_date	TIMESTAMP		Date of listening

11. Ratings

user_id	VARCHAR(50)	PK, FK	ID of user
track_id	VARCHAR(50)	PK, FK	ID of track
rating	SMALLINT		The rating of track (1-5)
review	TEXT		Review

12. Recommendations

recommendation_id	VARCHAR(50)	PK	ID of recommendation
user_id	VARCHAR(50)	FK	ID of user
track_id	VARCHAR(50)	FK	ID of track that recommended

13. Charts

chart_id	VARCHAR(50)	PK	ID of Chart
name	VARCHAR(100)		Name of chart
country_id	VARCHAR(50)	FK	Country of chart

14. ChartTrack

chart_id	VARCHAR(50)	PK, FK	ID of chart
track_id	VARCHAR(50)	PK, FK	ID of track
position	INTEGER		Position in chart

The queries are also represented in [GitHub repository](#)

The query of creating tables:

```
CREATE TABLE Countries (  
    country_id VARCHAR(50) PRIMARY KEY,  
    name VARCHAR(100) NOT NULL UNIQUE  
);  
  
CREATE TABLE Subscriptions (  
    subscription_id VARCHAR(50) PRIMARY KEY,  
    name VARCHAR(50) NOT NULL UNIQUE,  
    price DECIMAL(6,2) NOT NULL  
);  
  
CREATE TABLE "users" (  
    user_id VARCHAR(50) PRIMARY KEY,  
    name VARCHAR(100) NOT NULL,  
    email VARCHAR(100) NOT NULL UNIQUE,  
    password VARCHAR(255) NOT NULL,  
    subscription_id VARCHAR(50) REFERENCES Subscriptions(subscription_id),  
    country_id VARCHAR(50) REFERENCES Countries(country_id),  
    registration_date TIMESTAMP NOT NULL DEFAULT NOW()  
);  
  
CREATE TABLE Artists (  
    artist_id VARCHAR(50) PRIMARY KEY,  
    name VARCHAR(150) NOT NULL,  
    bio TEXT  
);  
  
CREATE TABLE Genres (  
    genre_id VARCHAR(50) PRIMARY KEY,  
    name VARCHAR(100) NOT NULL UNIQUE  
);  
  
CREATE TABLE Albums (  
    album_id VARCHAR(50) PRIMARY KEY,  
    title VARCHAR(200) NOT NULL,  
    release_date DATE,  
    artist_id VARCHAR(50) NOT NULL REFERENCES Artists(artist_id)  
);  
  
CREATE TABLE Tracks (  
    track_id VARCHAR(50) PRIMARY KEY,  
    title VARCHAR(200) NOT NULL,  
    duration INTERVAL MINUTE TO SECOND NOT NULL,  
    artist_id VARCHAR(50) NOT NULL REFERENCES Artists(artist_id),  
    album_id VARCHAR(50) REFERENCES Albums(album_id),  
    genre_id VARCHAR(50) REFERENCES Genres(genre_id)  
);
```

```

CREATE TABLE Playlists (
    playlist_id VARCHAR(50) PRIMARY KEY,
    title VARCHAR(200) NOT NULL,
    user_id VARCHAR(50) NOT NULL REFERENCES "users"(user_id)
);

CREATE TABLE Playlist_Track (
    playlist_id VARCHAR(50) NOT NULL,
    track_id VARCHAR(50) NOT NULL,
    position INT,
    PRIMARY KEY (playlist_id, track_id),
    FOREIGN KEY (playlist_id) REFERENCES Playlists(playlist_id),
    FOREIGN KEY (track_id) REFERENCES Tracks(track_id)
);

CREATE TABLE Listening_Histories (
    history_id VARCHAR(50) PRIMARY KEY,
    user_id VARCHAR(50) NOT NULL REFERENCES "users"(user_id),
    track_id VARCHAR(50) NOT NULL REFERENCES Tracks(track_id),
    play_date TIMESTAMP NOT NULL DEFAULT NOW()
);

CREATE TABLE Ratings (
    user_id VARCHAR(50) NOT NULL,
    track_id VARCHAR(50) NOT NULL,
    rating SMALLINT CHECK (rating BETWEEN 1 AND 5),
    review TEXT,
    PRIMARY KEY (user_id, track_id),
    FOREIGN KEY (user_id) REFERENCES "users"(user_id),
    FOREIGN KEY (track_id) REFERENCES Tracks(track_id)
);

CREATE TABLE Recommendations (
    recommendation_id VARCHAR(50) PRIMARY KEY,
    user_id VARCHAR(50) NOT NULL REFERENCES "users"(user_id),
    track_id VARCHAR(50) NOT NULL REFERENCES Tracks(track_id)
);

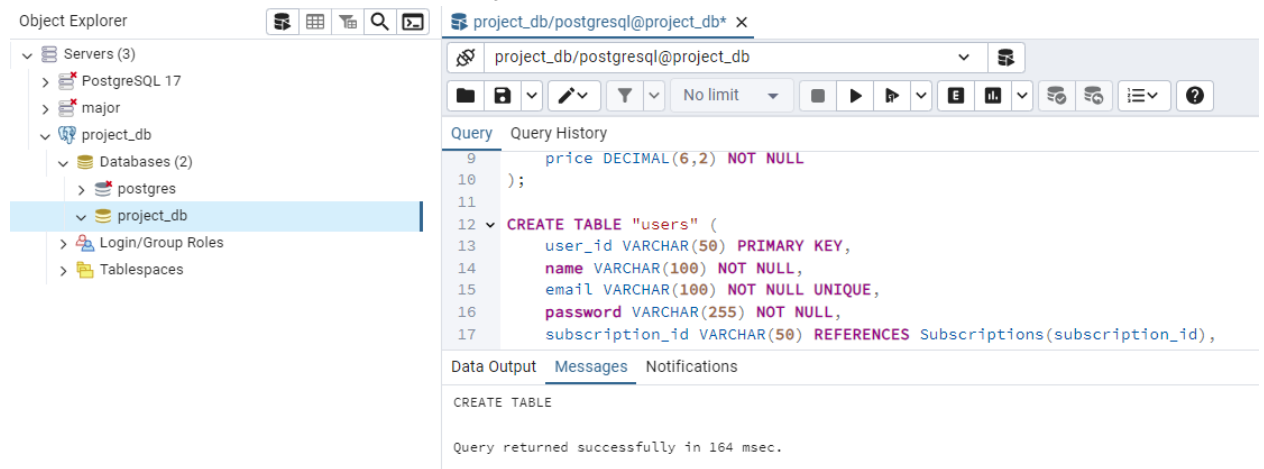
CREATE TABLE Charts (
    chart_id VARCHAR(50) PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    country_id VARCHAR(50) REFERENCES Countries(country_id)
);

CREATE TABLE ChartTrack (
    chart_id VARCHAR(50) NOT NULL,
    track_id VARCHAR(50) NOT NULL,
    position INT NOT NULL,
    PRIMARY KEY (chart_id, track_id),
    FOREIGN KEY (chart_id) REFERENCES Charts(chart_id),

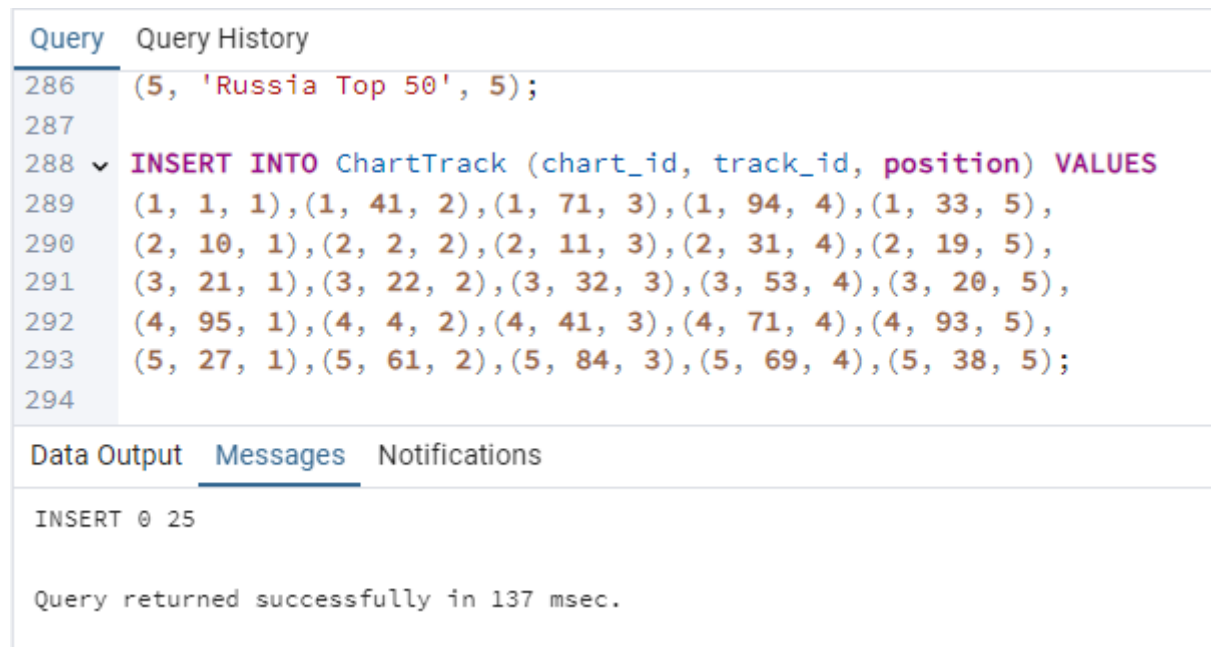
```

```
);  
FOREIGN KEY (track_id) REFERENCES Tracks(track_id)
```

Now if I enter this query in our remote database:



Successfully created tables. And inserting values:



The full query:

```
INSERT INTO Countries (country_id, name) VALUES  
(1, 'United Kingdom'),  
(2, 'United States'),  
(3, 'Canada'),  
(4, 'Kazakhstan'),  
(5, 'Russia'),  
(6, 'Germany'),  
(7, 'France'),  
(8, 'Australia'),  
(9, 'Brazil'),  
(10, 'Japan');  
  
INSERT INTO Subscriptions (subscription_id, name, price) VALUES
```

```
(1, 'Free', 0.00),  
(2, 'Premium', 9.99),  
(3, 'Family', 14.99);
```

```
INSERT INTO "users" (user_id, name, email, password, subscription_id, country_id,  
registration_date) VALUES  
(1, 'Aidar Nurgaliyev', 'aidar.n@local.kz', 'hashed_pass_a1', 2, 4, '2025-01-10  
09:12:00'),  
(2, 'Gulnara Seitova', 'gulnara@example.kz', 'hashed_pass_a2', 1, 4, '2025-02-05  
14:30:00'),  
(3, 'John Miller', 'john.miller@example.com', 'hashed_pass_a3', 2, 2, '2024-11-20  
08:00:00'),  
(4, 'Maria Petrova', 'm.petrov@example.ru', 'hashed_pass_a4', 3, 5, '2023-05-12  
18:45:00'),  
(5, 'Emma Thompson', 'emma.t@example.uk', 'hashed_pass_a5', 2, 1, '2024-08-01  
10:05:00'),  
(6, 'Carlos Silva', 'c.silva@example.br', 'hashed_pass_a6', 1, 9, '2025-03-03  
21:10:00'),  
(7, 'Satoshi Yamamoto', 's.yama@example.jp', 'hashed_pass_a7', 2, 10, '2024-12-12  
12:00:00'),  
(8, 'Lena Müller', 'lena.mueller@example.de', 'hashed_pass_a8', 2, 6, '2025-06-15  
11:23:00'),  
(9, 'Olga Ivanova', 'olga.ivanova@example.ru', 'hashed_pass_a9', 1, 5, '2025-04-  
22 16:40:00'),  
(10, 'Michael Johnson', 'm.johnson@example.com', 'hashed_pass_a10', 3, 2, '2022-  
10-30 09:00:00'),  
(11, 'Daniela Rossi', 'd.rossi@example.it', 'hashed_pass_a11', 1, 7, '2025-07-01  
13:18:00'),  
(12, 'Alex Kim', 'alex.kim@example.ca', 'hashed_pass_a12', 2, 3, '2025-02-27  
07:50:00');
```

```
INSERT INTO Artists (artist_id, name, bio) VALUES  
(1, 'Queen', 'Legendary British rock band formed in 1970.'),  
(2, 'Boney M', 'Disco group created by German record producer Frank Farian.'),  
(3, 'Twenty One Pilots', 'American musical duo from Columbus, Ohio.'),  
(4, 'Travis Scott', 'American rapper and record producer.'),  
(5, 'Kendrick Lamar', 'American rapper and songwriter.'),  
(6, 'Kanye West', 'American rapper, producer and fashion designer.'),  
(7, 'Adele', 'English singer-songwriter.'),  
(8, 'Drake', 'Canadian rapper, singer and actor.'),  
(9, 'The Weeknd', 'Canadian singer, songwriter, and record producer.'),  
(10, 'Imagine Dragons', 'American pop rock band.'),  
(11, 'Coldplay', 'British rock band formed in London.'),  
(12, 'Metallica', 'American heavy metal band.'),  
(13, 'Billie Eilish', 'American singer-songwriter.'),  
(14, 'Post Malone', 'American rapper, singer, songwriter and record producer.'),  
(15, 'David Bowie', 'English singer-songwriter and actor.'),  
(16, 'Arctic Monkeys', 'English rock band formed in Sheffield.'),  
(17, 'Rihanna', 'Barbadian singer and businesswoman.'),  
(18, 'Skrillex', 'American electronic music producer and DJ.'),  
(19, 'Eminem', 'American rapper, songwriter and record producer.'),
```



```
(20, 'Shakira', 'Colombian singer and songwriter.');
```

```
INSERT INTO Genres (genre_id, name) VALUES
```

```
(1, 'Rock'),  
(2, 'Disco'),  
(3, 'Alternative'),  
(4, 'Hip-Hop'),  
(5, 'Pop'),  
(6, 'R&B'),  
(7, 'Electronic'),  
(8, 'Metal'),  
(9, 'Indie'),  
(10, 'Latin');
```

```
INSERT INTO Albums (album_id, title, release_date, artist_id) VALUES
```

```
(1, 'A Night at the Opera', '1975-11-21', 1),  
(2, 'The Miracle', '1989-05-22', 1),  
(3, 'Bobby Farrell Hits', '1978-06-01', 2),  
(4, 'Nightflight to Venus', '1978-05-01', 2),  
(5, 'Blurryface', '2015-05-17', 3),  
(6, 'Vessel', '2013-01-08', 3),  
(7, 'Astroworld', '2018-08-03', 4),  
(8, 'Rodeo', '2015-09-04', 4),  
(9, 'DAMN.', '2017-04-14', 5),  
(10, 'To Pimp a Butterfly', '2015-03-15', 5),  
(11, 'My Beautiful Dark Twisted Fantasy', '2010-11-22', 6),  
(12, 'Graduation', '2007-09-11', 6),  
(13, '25', '2015-11-20', 7),  
(14, '19', '2008-01-28', 7),  
(15, 'Views', '2016-04-29', 8),  
(16, 'Scorpion', '2018-06-29', 8),  
(17, 'After Hours', '2020-03-20', 9),  
(18, 'Beauty and the Beat', '2012-06-04', 9),  
(19, 'Night Visions', '2012-09-04', 10),  
(20, 'Evolve', '2017-06-23', 10),  
(21, 'Parachutes', '2000-07-10', 11),  
(22, 'A Rush of Blood to the Head', '2002-08-26', 11),  
(23, 'Master of Puppets', '1986-03-03', 12),  
(24, 'Ride the Lightning', '1984-07-27', 12),  
(25, 'When We All Fall Asleep, Where Do We Go?', '2019-03-29', 13),  
(26, 'Hollywood's Bleeding', '2019-09-06', 14),  
(27, 'Heroes', '1977-10-14', 15),  
(28, 'Whatever People Say I Am, That's What I'm Not', '2006-01-23', 16),  
(29, 'Anti', '2016-01-28', 17),  
(30, 'Recess', '2014-03-14', 18),  
(31, 'The Marshall Mathers LP', '2000-05-23', 19),  
(32, 'Laundry Service', '2001-11-13', 20),  
(33, 'Greatest Hits Vol.1', '2020-01-01', 14),  
(34, 'Late Registration', '2005-08-30', 6),  
(35, 'King of the Road', '2021-01-01', 4),  
(36, 'Selected Singles', '1999-01-01', 2);
```

```
INSERT INTO Tracks (track_id, title, duration, artist_id, album_id, genre_id)
VALUES
(1, 'Bohemian Rhapsody', INTERVAL '00:05:55', 1, 1, 1),
(2, 'Love of My Life', INTERVAL '00:03:38', 1, 1, 1),
(3, 'Killer Queen', INTERVAL '00:03:01', 1, 2, 1),
(4, 'Rasputin', INTERVAL '00:04:25', 2, 4, 2),
(5, 'Daddy Cool', INTERVAL '00:03:26', 2, 3, 2),
(6, 'Hold the Line', INTERVAL '00:03:58', 2, 36, 2),
(7, 'Stressed Out', INTERVAL '00:03:22', 3, 5, 3),
(8, 'Heavydirtysoul', INTERVAL '00:03:05', 3, 6, 3),
(9, 'Ride', INTERVAL '00:03:45', 4, 8, 4),
(10, 'Sicko Mode', INTERVAL '00:05:13', 4, 7, 4),
(11, 'Goosebumps', INTERVAL '00:04:02', 14, 33, 4),
(12, 'HUMBLE.', INTERVAL '00:02:57', 5, 9, 4),
(13, 'DNA.', INTERVAL '00:03:05', 5, 9, 4),
(14, 'Alright', INTERVAL '00:03:39', 5, 10, 6),
(15, 'Power', INTERVAL '00:04:52', 6, 11, 4),
(16, 'Stronger', INTERVAL '00:05:10', 6, 12, 4),
(17, 'Hello', INTERVAL '00:04:55', 7, 13, 5),
(18, 'Someone Like You', INTERVAL '00:04:45', 7, 13, 5),
(19, 'Hotline Bling', INTERVAL '00:04:27', 8, 15, 6),
(20, 'God''s Plan', INTERVAL '00:03:19', 8, 15, 6),
(21, 'Blinding Lights', INTERVAL '00:03:20', 9, 17, 6),
(22, 'Save Your Tears', INTERVAL '00:03:35', 9, 17, 6),
(23, 'Radioactive', INTERVAL '00:03:08', 10, 19, 5),
(24, 'Demons', INTERVAL '00:02:57', 10, 19, 5),
(25, 'Yellow', INTERVAL '00:04:29', 11, 21, 1),
(26, 'Clocks', INTERVAL '00:05:07', 11, 22, 1),
(27, 'Enter Sandman', INTERVAL '00:05:32', 12, 23, 8),
(28, 'Nothing Else Matters', INTERVAL '00:06:28', 12, 23, 8),
(29, 'Bad Guy', INTERVAL '00:03:14', 13, 25, 5),
(30, 'bury a friend', INTERVAL '00:03:13', 13, 25, 7),
(31, 'Circles', INTERVAL '00:03:35', 14, 26, 5),
(32, 'Sunflower', INTERVAL '00:02:38', 14, 26, 5),
(33, 'Heroes', INTERVAL '00:06:07', 15, 27, 1),
(34, 'Do I Wanna Know?', INTERVAL '00:04:32', 16, 28, 9),
(35, 'Umbrella', INTERVAL '00:04:36', 17, 29, 5),
(36, 'Bangarang', INTERVAL '00:03:35', 18, 30, 7),
(37, 'The Real Slim Shady', INTERVAL '00:04:44', 19, 31, 4),
(38, 'Lose Yourself', INTERVAL '00:05:26', 19, 31, 4),
(39, 'Hips Don''t Lie', INTERVAL '00:03:38', 20, 32, 10),
(40, 'Whenever, Wherever', INTERVAL '00:03:16', 20, 32, 10),
(41, 'Somebody to Love', INTERVAL '00:04:56', 1, 1, 1),
(42, 'I Want to Break Free', INTERVAL '00:04:18', 1, 2, 1),
(43, 'Ma Baker', INTERVAL '00:04:37', 2, 4, 2),
(44, 'Blurryface (Intro)', INTERVAL '00:01:56', 3, 5, 3),
(45, 'Stargazing', INTERVAL '00:04:30', 4, 7, 4),
(46, '90210', INTERVAL '00:05:00', 4, 8, 4),
(47, 'Swimming Pools', INTERVAL '00:03:40', 5, 9, 4),
(48, 'Poetic Justice', INTERVAL '00:05:00', 5, 10, 6),
```

(49, 'Flashing Lights', INTERVAL '00:03:58', 6, 11, 4),
(50, 'Good Life', INTERVAL '00:03:27', 6, 12, 5),
(51, 'Rumour Has It', INTERVAL '00:03:43', 7, 14, 5),
(52, 'Set Fire to the Rain', INTERVAL '00:04:02', 7, 13, 5),
(53, 'In My Feelings', INTERVAL '00:03:38', 8, 15, 6),
(54, 'One Dance', INTERVAL '00:02:54', 8, 15, 6),
(55, 'Heartless', INTERVAL '00:03:53', 9, 18, 6),
(56, 'The Hills', INTERVAL '00:04:02', 9, 17, 6),
(57, 'On Top of the World', INTERVAL '00:03:12', 10, 19, 5),
(58, 'Believer', INTERVAL '00:03:24', 10, 20, 5),
(59, 'The Scientist', INTERVAL '00:05:09', 11, 22, 1),
(60, 'Fix You', INTERVAL '00:04:55', 11, 21, 1),
(61, 'Master of Puppets', INTERVAL '00:08:35', 12, 23, 8),
(62, 'Fade to Black', INTERVAL '00:06:57', 12, 24, 8),
(63, 'Everything I Wanted', INTERVAL '00:04:05', 13, 25, 5),
(64, 'Goodbyes', INTERVAL '00:02:53', 14, 26, 5),
(65, 'Life on Mars?', INTERVAL '00:03:55', 15, 27, 1),
(66, 'Do I Wanna Know? (Live)', INTERVAL '00:04:45', 16, 28, 9),
(67, 'We Found Love', INTERVAL '00:03:35', 17, 29, 5),
(68, 'Scary Monsters and Nice Sprites', INTERVAL '00:04:07', 18, 30, 7),
(69, 'Stan', INTERVAL '00:06:44', 19, 31, 4),
(70, 'Whenever, Wherever (Remix)', INTERVAL '00:03:50', 20, 32, 10),
(71, 'Another One Bites The Dust', INTERVAL '00:03:35', 1, 1, 1),
(72, 'Under Pressure', INTERVAL '00:04:08', 1, 2, 1),
(73, 'Sunshine of My Life', INTERVAL '00:04:00', 2, 36, 2),
(74, 'Chlorine', INTERVAL '00:05:30', 3, 6, 3),
(75, 'Goosebumps (Remix)', INTERVAL '00:04:20', 14, 33, 4),
(76, 'Antidote', INTERVAL '00:03:33', 4, 35, 4),
(77, 'ELEMENT.', INTERVAL '00:03:26', 5, 9, 4),
(78, 'Power (Live)', INTERVAL '00:05:02', 6, 34, 4),
(79, 'Rolling in the Deep', INTERVAL '00:03:49', 7, 13, 5),
(80, 'Hotline Bling (Acoustic)', INTERVAL '00:03:50', 8, 15, 6),
(81, 'Blinding Lights (Remix)', INTERVAL '00:03:40', 9, 17, 6),
(82, 'Thunder', INTERVAL '00:03:07', 10, 20, 5),
(83, 'Viva La Vida', INTERVAL '00:04:02', 11, 21, 1),
(84, 'Nothing Else Matters (Live)', INTERVAL '00:06:22', 12, 23, 8),
(85, 'when the party's over', INTERVAL '00:04:53', 13, 25, 5),
(86, 'White Iverson', INTERVAL '00:04:18', 14, 33, 5),
(87, 'Space Oddity', INTERVAL '00:05:18', 15, 27, 1),
(88, 'R U Mine?', INTERVAL '00:03:22', 16, 28, 9),
(89, 'Diamonds', INTERVAL '00:03:45', 17, 29, 5),
(90, 'Breakn'' a Sweat', INTERVAL '00:04:02', 18, 30, 7),
(91, 'The Real Slim Shady (Live)', INTERVAL '00:05:10', 19, 31, 4),
(92, 'She Wolf', INTERVAL '00:03:06', 20, 32, 10),
(93, 'Somebody to Love (Live)', INTERVAL '00:05:03', 1, 1, 1),
(94, 'Bohemian Rhapsody (Live Aid)', INTERVAL '00:06:30', 1, 2, 1),
(95, 'Rivers of Babylon', INTERVAL '00:03:52', 2, 3, 2),
(96, 'Stressed Out (Live)', INTERVAL '00:03:50', 3, 5, 3),
(97, 'Astrothunder', INTERVAL '00:03:27', 4, 7, 4),
(98, 'LOVE.', INTERVAL '00:03:32', 5, 9, 6),
(99, 'Devil in a New Dress', INTERVAL '00:05:52', 6, 11, 4),

```
(100, 'La Tortura', INTERVAL '00:03:34', 20, 32, 10);
```

```
INSERT INTO Playlists (playlist_id, title, user_id) VALUES
```

```
(1, 'Aidar''s Mix', 1),  
(2, 'Gulnara''s Favorites', 2),  
(3, 'John''s Roadtrip', 3),  
(4, 'Maria''s Chill', 4),  
(5, 'Emma''s Hits', 5),  
(6, 'Carlos Party', 6),  
(7, 'Satoshi Selects', 7),  
(8, 'Lena Workout', 8),  
(9, 'Olga Study', 9),  
(10, 'Michael Classics', 10),  
(11, 'Daniela Dance', 11),  
(12, 'Alex Top 40', 12),  
(13, 'Shared: Party Mix', 1),  
(14, 'Shared: Soft Hits', 3);
```

```
INSERT INTO Playlist_Track (playlist_id, track_id, position) VALUES
```

```
(1, 1, 1),(1, 41, 2),(1, 71, 3),(1, 94, 4),  
(2, 4, 1),(2, 43, 2),(2, 95, 3),  
(3, 10, 1),(3, 45, 2),(3, 76, 3),(3, 97, 4),  
(4, 29, 1),(4, 30, 2),(4, 85, 3),  
(5, 17, 1),(5, 79, 2),(5, 25, 3),  
(6, 11, 1),(6, 31, 2),(6, 75, 3),  
(7, 21, 1),(7, 22, 2),(7, 81, 3),  
(8, 57, 1),(8, 58, 2),(8, 82, 3),  
(9, 63, 1),(9, 64, 2),(9, 86, 3),  
(10, 27, 1),(10, 61, 2),(10, 84, 3),  
(11, 36, 1),(11, 68, 2),(11, 90, 3),  
(12, 19, 1),(12, 20, 2),(12, 53, 3),  
(13, 2, 1),(13, 3, 2),(13, 41, 3),(13, 71, 4),  
(14, 59, 1),(14, 60, 2),(14, 69, 3);
```

```
INSERT INTO Listening_Histories (history_id, user_id, track_id, play_date) VALUES
```

```
(1, 1, 1, '2025-10-10 12:00:00'),  
(2, 1, 41, '2025-10-10 12:05:00'),  
(3, 2, 4, '2025-09-30 09:12:00'),  
(4, 3, 10, '2025-10-01 18:22:00'),  
(5, 4, 29, '2025-09-25 20:00:00'),  
(6, 5, 17, '2025-10-08 14:33:00'),  
(7, 6, 11, '2025-10-09 23:01:00'),  
(8, 7, 21, '2025-10-07 07:44:00'),  
(9, 8, 57, '2025-10-06 06:30:00'),  
(10, 9, 63, '2025-10-05 11:11:00'),  
(11, 10, 27, '2025-09-29 22:22:00'),  
(12, 11, 31, '2025-10-03 16:16:00'),  
(13, 12, 19, '2025-10-02 19:45:00'),  
(14, 2, 95, '2025-09-20 13:10:00'),  
(15, 3, 96, '2025-09-21 14:14:00');
```

```
INSERT INTO Ratings (user_id, track_id, rating, review) VALUES
(1, 1, 5, 'Epic song.'),
(2, 4, 4, 'Great disco vibes.'),
(3, 10, 5, 'Sick production.'),
(4, 29, 4, 'Beautiful melody.'),
(5, 17, 5, 'Powerful voice.'),
(6, 11, 4, 'Nice beat.'),
(7, 21, 5, 'Can''t stop listening.'),
(8, 57, 4, 'Good workout track.'),
(9, 63, 5, 'Very emotional.'),
(10, 27, 5, 'Metal classic.'),
(11, 31, 4, 'Catchy.'),
(12, 19, 3, 'Not my style.');
```

```
INSERT INTO Recommendations (recommendation_id, user_id, track_id) VALUES
(1, 1, 2),
(2, 1, 41),
(3, 2, 43),
(4, 3, 10),
(5, 4, 29),
(6, 5, 17),
(7, 6, 11),
(8, 7, 21),
(9, 8, 57),
(10, 9, 63);
```

```
INSERT INTO Charts (chart_id, name, country_id) VALUES
(1, 'UK Top Hits', 1),
(2, 'US Billboard', 2),
(3, 'Canada Hot 100', 3),
(4, 'Kazakhstan Top', 4),
(5, 'Russia Top 50', 5);
```

```
INSERT INTO ChartTrack (chart_id, track_id, position) VALUES
(1, 1, 1),(1, 41, 2),(1, 71, 3),(1, 94, 4),(1, 33, 5),
(2, 10, 1),(2, 2, 2),(2, 11, 3),(2, 31, 4),(2, 19, 5),
(3, 21, 1),(3, 22, 2),(3, 32, 3),(3, 53, 4),(3, 20, 5),
(4, 95, 1),(4, 4, 2),(4, 41, 3),(4, 71, 4),(4, 93, 5),
(5, 27, 1),(5, 61, 2),(5, 84, 3),(5, 69, 4),(5, 38, 5);
```

User group 1: Listeners (Usual users without subscription)

Access Rights:

Tracks	Read-only
Artists	Read-only
Albums	Read-only
Genres	Read-only
Countries	Read-only
Subscriptions	Read-only
Playlists	Read-only
Playlist_Track	Read-only
Listening_Histories	Read-only
Ratings	Read-write
Recommendations	Read-only

User group 2: Listeners with subscription

Tracks	Read-only
Artists	Read-only
Albums	Read-only
Genres	Read-only
Countries	Read-only
Subscriptions	Read-only
Playlists	Read-write
Playlist_Track	Read-write
Listening_Histories	Read-write
Ratings	Read-write
Recommendations	Read-only

User group 3: Content Managers

Access Rights:

Artists	Read-write
Albums	Read-write
Tracks	Read-write
Genres	Read-write
Charts	Read-write
ChartTrack	Read-write
Countries	Read-only
Subscriptions	Read-only
Users	Read-only
Playlists	Read-only
Ratings	Read-only
Recommendations	Read-only

User group 4: Artists

Artists	Read-only
Albums	Read-write (Only on their profile)
Tracks	Read-write (Only on their profile)
Genres	Read-write (Only on their profile)
Charts	Read-only
ChartTrack	Read-only
Countries	Read-only
Subscriptions	Read-only
Users	Read-only
Playlists	Read-only
Ratings	Read-only
Recommendations	Read-only

Threats for the database:

1. SQL Injection

A: A user is able to inject SQL query in input fields such as “Search bar” or “Login form”

B: Use only ORM like SQLAlchemy, validate input at application layer, use parameterized queries, prepared statements.

2. Illegal entry by hacker











A: A hacker can enter to our database due to the lack of an SSH tunnel for connection. So, hacker needs only login and password to connect to the database.

B: Use of SSH tunnel and connect to the database by using Public and Private keys.

Part 1. PostgreSQL with CSV format

As I, Batyrkhan, love the Queen group, I want to write queries related to them. So, at first, I want to count how many tracks there are in the database. So, there is the SQL query and its result:

1	SELECT	a.name,	COUNT(t.title)
2	FROM	tracks t	
3	JOIN	artists a ON t.artist_id = a.artist_id	
4	WHERE	a.name = 'Queen'	
5	GROUP BY	a.name;	

Data Output		Messages	Notifications
<div></div>			
	name		count
	character varying (150)		bigint
1	Queen		9

And what tracks?

Query

Query History

1

2

3

SELECT

t.title, t.duration, a.name, a.bio

FROM

tracks t

JOIN

artists a

ON

t.artist_id = a.artist_id

WHERE

a.name = 'Queen';

Data Output

Messages

Notifications

Trying to export and...

Query Query History

```
1  COPY (
2      SELECT t.title, t.duration, a.name, a.bio FROM tracks t
3      JOIN artists a ON t.artist_id = a.artist_id
4      WHERE a.name = 'Queen'
5  ) TO '/home/khan/queen_tracks.csv'
6  DELIMITER ','
7  CSV
8  HEADER;
```

Data Output Messages Notifications

ERROR: must be superuser or a member of the pg_write_server_files role to COPY to a file
HINT: Anyone can COPY to stdout or from stdin. psql's \copy command also works for anyone.

SQL state: 42501

That's the problem... I can't export the data in to the server. But it can be fixed if I use \copy from terminal or give the permission to "postgresql" user to change server's data:

```
postgres=# GRANT pg_write_server_files TO postgresql;
GRANT ROLE
```

Try again:

Query Query History

```
1  COPY (
2      SELECT t.title, t.duration, a.name, a.bio FROM tracks t
3      JOIN artists a ON t.artist_id = a.artist_id
4      WHERE a.name = 'Queen'
5  ) TO '/tmp/queen_tracks.csv'
6  DELIMITER ','
7  CSV
8  HEADER;
```

Data Output Messages Notifications

COPY 9

Query returned successfully in 116 msec.

Imported to /tmp directory because khan is my user's folder which can be accessed by my private key.

Anyway, it worked!

The next query is to count how many music in each genre and order it in descending order:

```
1 SELECT g.name, COUNT(*) as total_tracks
2 FROM tracks t
3 JOIN genres g ON t.genre_id = g.genre_id
4 GROUP BY g.name
5 ORDER BY total_tracks DESC;
```

Data Output Messages Notifications



	name character varying (100)	total_tracks bigint
1	Pop	21
2	Hip-Hop	21
3	Rock	17
4	R&B	13
5	Disco	6
6	Alternative	5
7	Latin	5
8	Metal	5
9	Electronic	4
10	Indie	3

Exporting the data:

```
Query  Query History
1  v  COPY (
2      SELECT g.name, COUNT(*) as total_tracks
3      FROM tracks t
4      JOIN genres g ON t.genre_id = g.genre_id
5      GROUP BY g.name
6      ORDER BY total_tracks DESC
7  ) TO '/tmp/genres_tracks.csv'
8  DELIMITER ','
9  CSV
10 HEADER;
11
```

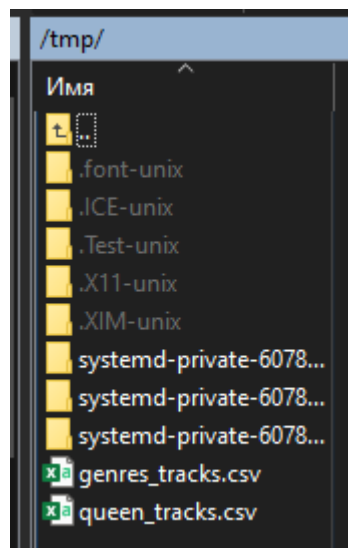
Data Output Messages Notifications

COPY 10

Query returned successfully in 112 msec.

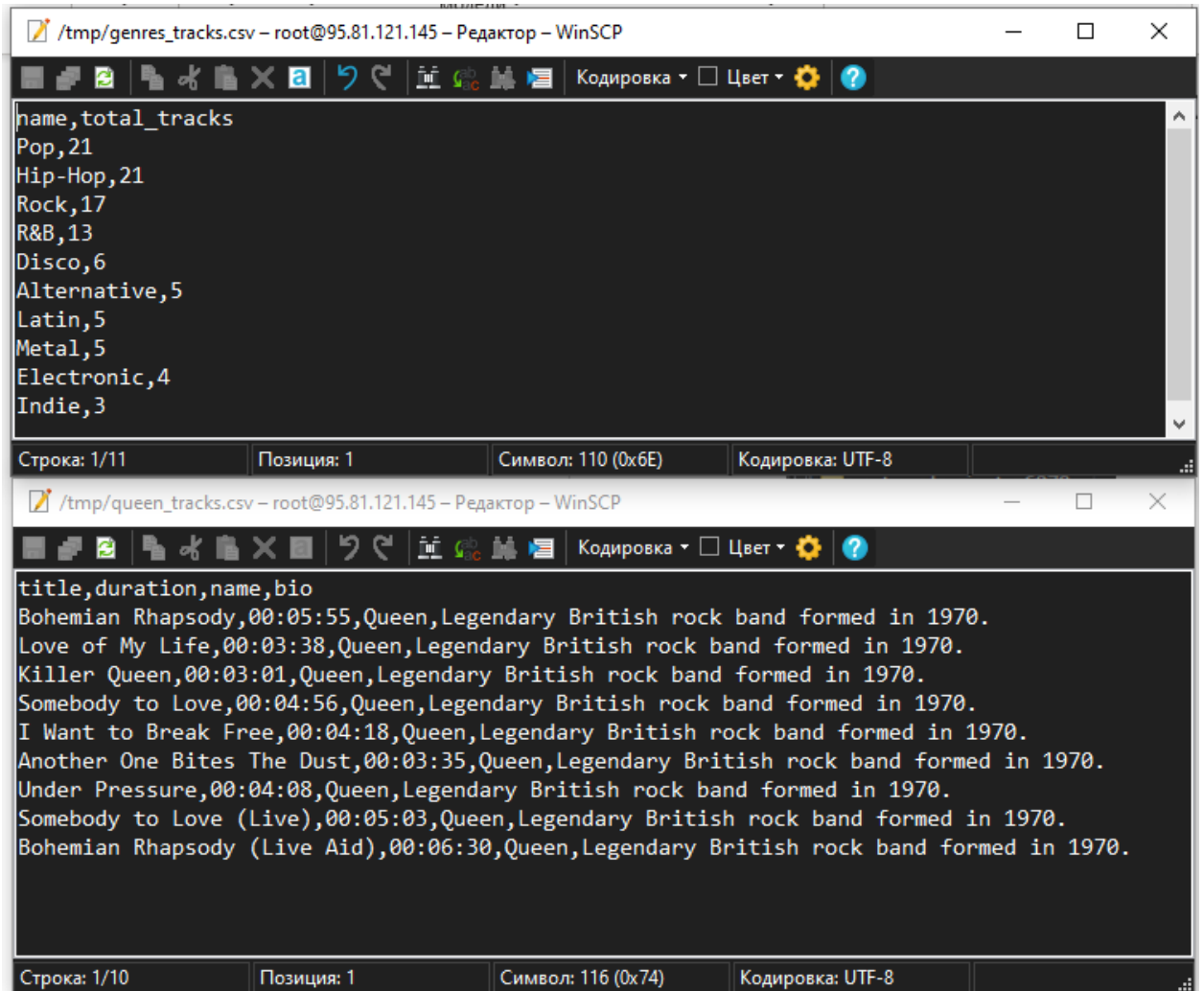
Now, let's connect to the server via FTP and check the CSV files:

WinSCP is needed to enter to the server's file system:



They are right here.

And if I open them:



The image shows two screenshots of the WinSCP editor interface. The top window displays the file `/tmp/genres_tracks.csv` with the following content:

```
name,total_tracks
Pop,21
Hip-Hop,21
Rock,17
R&B,13
Disco,6
Alternative,5
Latin,5
Metal,5
Electronic,4
Indie,3
```

The bottom window displays the file `/tmp/queen_tracks.csv` with the following content:

```
title,duration,name,bio
Bohemian Rhapsody,00:05:55,Queen,Legendary British rock band formed in 1970.
Love of My Life,00:03:38,Queen,Legendary British rock band formed in 1970.
Killer Queen,00:03:01,Queen,Legendary British rock band formed in 1970.
Somebody to Love,00:04:56,Queen,Legendary British rock band formed in 1970.
I Want to Break Free,00:04:18,Queen,Legendary British rock band formed in 1970.
Another One Bites The Dust,00:03:35,Queen,Legendary British rock band formed in 1970.
Under Pressure,00:04:08,Queen,Legendary British rock band formed in 1970.
Somebody to Love (Live),00:05:03,Queen,Legendary British rock band formed in 1970.
Bohemian Rhapsody (Live Aid),00:06:30,Queen,Legendary British rock band formed in 1970.
```

The CSVs were exported successfully. Great!

Part 2. PostgreSQL with JSON format

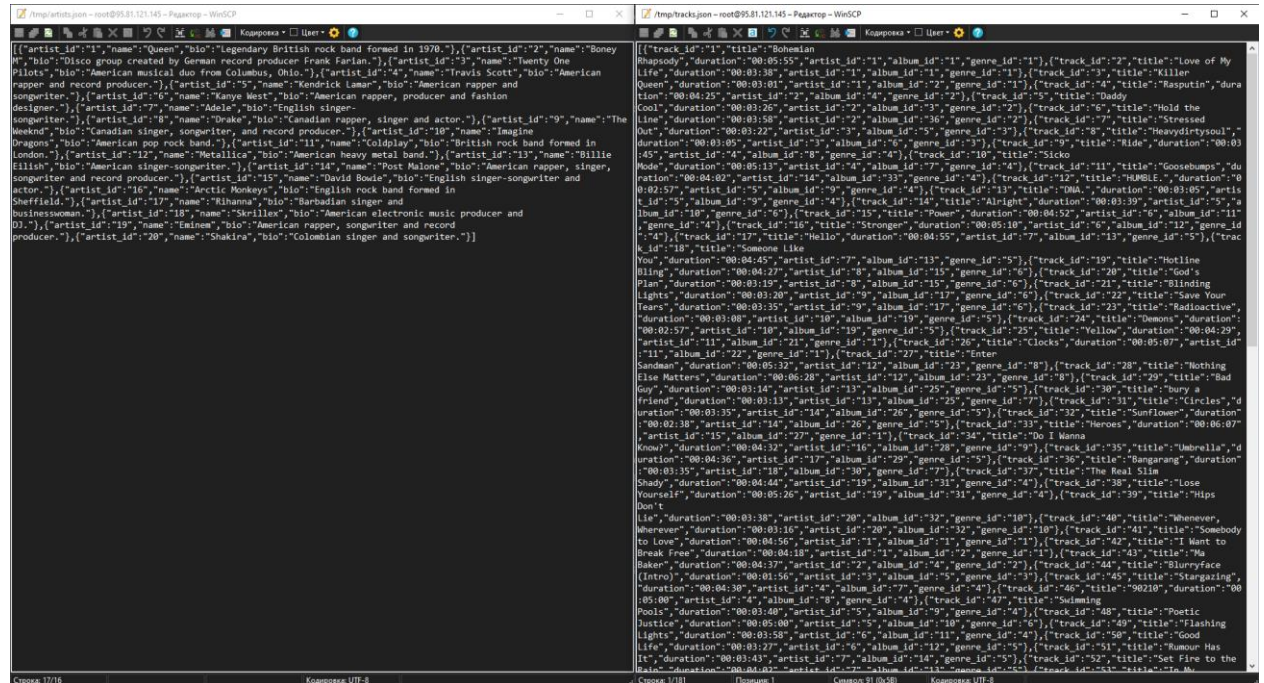
The first query is to copy the tracks table to JSON. There is the query:

Query	Query History	
<pre>1 ▾ COPY (2 SELECT array_to_json(array_agg(row_to_json(t))) 3 FROM (SELECT * FROM tracks) t 4) TO '/tmp/tracks.json';</pre>		
Data Output	Messages	Notifications
COPY 1		
Query returned successfully in 99 msec.		

And the second query is to copy the artists table to JSON. And there is the query for this:

Query	Query History	
<pre>1 ▾ COPY (2 SELECT array_to_json(array_agg(row_to_json(a))) 3 FROM (SELECT * FROM artists) a 4) TO '/tmp/artists.json';</pre>		
Data Output	Messages	Notifications
COPY 1		
Query returned successfully in 104 msec.		

And if I look at the files via FTP:



If I format it to normal JSON:

```
[{"track_id": "1", "title": "Bohemian Rhapsody", "duration": "00:05:55", "artist_id": "1", "album_id": "1", "genre_id": "1"}, {"track_id": "2", "title": "Love of My Life", "duration": "00:03:38", "artist_id": "1", "album_id": "1", "genre_id": "1"}, {"track_id": "3", "title": "Killer Queen", "duration": "00:03:01", "artist_id": "1", "album_id": "1", "genre_id": "1"}, {"track_id": "4", "title": "Rasputin", "duration": "00:04:25", "artist_id": "2", "album_id": "4", "genre_id": "2"}, {"track_id": "5", "title": "Daddy Cool", "duration": "00:03:26", "artist_id": "2", "album_id": "3", "genre_id": "2"}, {"track_id": "6", "title": "Hold the Line", "duration": "00:03:58", "artist_id": "2", "album_id": "36", "genre_id": "2"}, {"track_id": "7",
```

And for artists:

```
[
  {
    "artist_id": "1",
    "name": "Queen",
    "bio": "Legendary British rock band formed in 1970."
  },
  {
    "artist_id": "2",
    "name": "Boney M",
    "bio": "Disco group created by German record producer Frank Farian."
  },
  {
    "artist_id": "3",
    "name": "Twenty One Pilots",
    "bio": "American musical duo from Columbus, Ohio."
  },
  {
    "artist_id": "4",
    "name": "Travis Scott",
    "bio": "American rapper and record producer."
  },
  {
    "artist_id": "5",
    "name": "Kendrick Lamar",
    "bio": "American rapper and songwriter."
  },
  {
    "artist_id": "6",
    "name": "Kanye West",
    "bio": "American rapper, producer and fashion designer."
  },
  {
    "artist_id": "7",
    "name": "Adele",
    "bio": "English singer-songwriter."
  },
  {
    "artist_id": "8",
    "name": "Drake",
    "bio": "Canadian rapper, singer and actor."
  },
  {
    "artist_id": "9",
    "name": "The Weeknd",
    "bio": "Canadian singer, songwriter, and record producer."
  },
  {
    "artist_id": "10",
    "name": "Imagine Dragons",
    "bio": "American pop rock band."
  },
]
```

It worked! Great!

Part 3. Windows functions

It's time to use window functions.

1. The number of tracks of each artist:

Query

Query History

1

2

3

SELECT

a.name,

COUNT(t.track_id)

OVER

(PARTITION BY

a.artist_id)

AS

total_tracks

FROM

artists a

JOIN

tracks t

ON

a.artist_id = t.artist_id;

Data Output

Messages

Notifications

SQL

	name character varying (150)	total_tracks bigint
1	Queen	9
2	Queen	9
3	Queen	9
4	Queen	9
5	Queen	9
6	Queen	9
7	Queen	9
8	Queen	9
9	Queen	9
10	Imagine Dragons	5
11	Imagine Dragons	5
12	Imagine Dragons	5
13	Imagine Dragons	5
14	Imagine Dragons	5
15	Coldplay	5
16	Coldplay	5
17	Coldplay	5
18	Coldplay	5
19	Coldplay	5
20	Metallica	5
21	Metallica	5
22	Metallica	5
23	Metallica	5
24	Metallica	5
25	Billie Eilish	4
26	Billie Eilish	4

Total rows: 100 of 100 Query complete 00:00:00.362 Ln 3, Col 44

2. Average track duration by artist:

Query

Query History

1

SELECT a.name, t.title, t.duration,

2

AVG(t.duration) OVER (PARTITION BY a.artist_id) AS avg_duration_by_artist

3

FROM tracks t

4

JOIN artists a ON t.artist_id = a.artist_id;

Data Output

Messages

Notifications

+

📄

▼

📋

▼

🗑️

🔍

⬇️

📈

SQL

	name character varying (150)	title character varying (200)	duration interval	avg_duration_by_artist interval
1	Queen	Somebody to Love	00:04:56	00:04:33.777778
2	Queen	Killer Queen	00:03:01	00:04:33.777778
3	Queen	Bohemian Rhapsody (Live Aid)	00:06:30	00:04:33.777778
4	Queen	Somebody to Love (Live)	00:05:03	00:04:33.777778
5	Queen	Another One Bites The Dust	00:03:35	00:04:33.777778
6	Queen	Under Pressure	00:04:08	00:04:33.777778
7	Queen	I Want to Break Free	00:04:18	00:04:33.777778
8	Queen	Love of My Life	00:03:38	00:04:33.777778
9	Queen	Bohemian Rhapsody	00:05:55	00:04:33.777778
10	Imagine Dragons	Radioactive	00:03:08	00:03:09.6
11	Imagine Dragons	Demons	00:02:57	00:03:09.6
12	Imagine Dragons	On Top of the World	00:03:12	00:03:09.6
13	Imagine Dragons	Thunder	00:03:07	00:03:09.6
14	Imagine Dragons	Believer	00:03:24	00:03:09.6
15	Coldplay	Clocks	00:05:07	00:04:44.4
16	Coldplay	Fix You	00:04:55	00:04:44.4
17	Coldplay	Yellow	00:04:29	00:04:44.4
18	Coldplay	The Scientist	00:05:09	00:04:44.4
19	Coldplay	Viva La Vida	00:04:02	00:04:44.4
20	Metallica	Nothing Else Matters	00:06:28	00:06:46.8
21	Metallica	Master of Puppets	00:08:35	00:06:46.8
22	Metallica	Fade to Black	00:06:57	00:06:46.8
23	Metallica	Nothing Else Matters (Live)	00:06:22	00:06:46.8
24	Metallica	Enter Sandman	00:05:32	00:06:46.8
25	Billie Eilish	Everything I Wanted	00:04:05	00:03:51.25
26	Billie Eilish	bury a friend	00:03:13	00:03:51.25

Total rows: 100 of 100

Query complete 00:00:00.166

Ln 1, Col 8

3. Rating of Queen tracks by duration:

[Query](#) [Query History](#)

```
1 SELECT a.name, t.title, t.duration,  
2 RANK() OVER (PARTITION BY a.artist_id ORDER BY t.duration DESC)  
3 FROM tracks t  
4 JOIN artists a ON t.artist_id = a.artist_id  
5 WHERE a.name = 'Queen';
```

Data Output Messages Notifications

	name character varying (150) 🔒	title character varying (200) 🔒	duration interval 🔒	rank bigint 🔒
1	Queen	Bohemian Rhapsody (Live Aid)	00:06:30	1
2	Queen	Bohemian Rhapsody	00:05:55	2
3	Queen	Somebody to Love (Live)	00:05:03	3
4	Queen	Somebody to Love	00:04:56	4
5	Queen	I Want to Break Free	00:04:18	5
6	Queen	Under Pressure	00:04:08	6
7	Queen	Love of My Life	00:03:38	7
8	Queen	Another One Bites The Dust	00:03:35	8
9	Queen	Killer Queen	00:03:01	9

4. The user's last played track:

Query

Query History

1

▼

SELECT u.name, t.title, lh.play_date,

2

LAG(t.title) OVER (PARTITION BY u.user_id ORDER BY lh.play_date) AS previous_track

3

FROM listening_histories lh

4

JOIN users u ON lh.user_id = u.user_id

5

JOIN tracks t ON lh.track_id = t.track_id;

Data Output

Messages

Notifications

≡

📄

▼

📋

▼

🗑️

🔍

📥

📶

SQL

	name character varying (100) 🔒	title character varying (200) 🔒	play_date timestamp without time zone 🔒	previous_track character varying 🔒
1	Aidar Nurgaliyev	Bohemian Rhapsody	2025-10-10 12:00:00	[null]
2	Aidar Nurgaliyev	Somebody to Love	2025-10-10 12:05:00	Bohemian Rhapso...
3	Michael Johnson	Enter Sandman	2025-09-29 22:22:00	[null]
4	Daniela Rossi	Circles	2025-10-03 16:16:00	[null]
5	Alex Kim	Hotline Bling	2025-10-02 19:45:00	[null]
6	Gulnara Seitova	Rivers of Babylon	2025-09-20 13:10:00	[null]
7	Gulnara Seitova	Rasputin	2025-09-30 09:12:00	Rivers of Babylon
8	John Miller	Stressed Out (Live)	2025-09-21 14:14:00	[null]
9	John Miller	Sicko Mode	2025-10-01 18:22:00	Stressed Out (Live)
10	Maria Petrova	Bad Guy	2025-09-25 20:00:00	[null]
11	Emma Thompson	Hello	2025-10-08 14:33:00	[null]
12	Carlos Silva	Goosebumps	2025-10-09 23:01:00	[null]
13	Satoshi Yamamoto	Blinking Lights	2025-10-07 07:44:00	[null]
14	Lena Müller	On Top of the World	2025-10-06 06:30:00	[null]
15	Olga Ivanova	Everything I Wanted	2025-10-05 11:11:00	[null]

5. Track rating by popularity in the chart:

Query

Query History

1

2

3

4

5

SELECT c.name, t.title, ct.position,
DENSE_RANK() OVER (PARTITION BY c.chart_id ORDER BY ct.position ASC)
FROM charttrack ct
JOIN charts c ON ct.chart_id = c.chart_id
JOIN tracks t ON ct.track_id = t.track_id;

Data Output

Messages

Notifications

<

6. Track numbering by alphabetic ascending order within each artist

Query

Query History

```

1 SELECT a.name, t.title, ROW_NUMBER() OVER (PARTITION BY a.artist_id ORDER BY t.title ASC)
2 FROM tracks t
3 JOIN artists a ON t.artist_id = a.artist_id;

```

Data Output

Messages

Notifications

+

📄

▼

📋

▼

🗑️

🗄️

⬇️

📈

SQL

	name character varying (150) 🔒	title character varying (200) 🔒	row_number bigint 🔒
1	Queen	Another One Bites The Dust	1
2	Queen	Bohemian Rhapsody	2
3	Queen	Bohemian Rhapsody (Live Aid)	3
4	Queen	I Want to Break Free	4
5	Queen	Killer Queen	5
6	Queen	Love of My Life	6
7	Queen	Somebody to Love	7
8	Queen	Somebody to Love (Live)	8
9	Queen	Under Pressure	9
10	Imagine Dragons	Believer	1
11	Imagine Dragons	Demons	2
12	Imagine Dragons	On Top of the World	3
13	Imagine Dragons	Radioactive	4
14	Imagine Dragons	Thunder	5
15	Coldplay	Clocks	1
16	Coldplay	Fix You	2
17	Coldplay	The Scientist	3
18	Coldplay	Viva La Vida	4
19	Coldplay	Yellow	5
20	Metallica	Enter Sandman	1
21	Metallica	Fade to Black	2
22	Metallica	Master of Puppets	3
23	Metallica	Nothing Else Matters	4
24	Metallica	Nothing Else Matters (Live)	5
25	Billie Eilish	Bad Guy	1
26	Billie Eilish	bury a friend	2

Total rows: 100 of 100

Query complete 00:00:00.170

Ln 3, Col 45

7. Splitting the tracks into 4 groups by duration:

Query

Query History

1

SELECT

a.name, t.title, t.duration, NTILE(4) OVER (ORDER BY t.duration)

2

FROM

tracks t

3

JOIN

artists a ON t.artist_id = a.artist_id;

Data Output

Messages

Notifications

≡

📄

▼

📋

▼

🗑

🗄

⬇

📈

SQL

	name character varying (150)	title character varying (200)	duration interval	ntile integer
1	Twenty One Pilots	Blurryface (Intro)	00:01:56	1
2	Post Malone	Sunflower	00:02:38	1
3	Post Malone	Goodbyes	00:02:53	1
4	Drake	One Dance	00:02:54	1
5	Kendrick Lamar	HUMBLE.	00:02:57	1
6	Imagine Dragons	Demons	00:02:57	1
7	Queen	Killer Queen	00:03:01	1
8	Kendrick Lamar	DNA.	00:03:05	1
9	Twenty One Pilots	Heavydirtysoul	00:03:05	1
10	Shakira	She Wolf	00:03:06	1
11	Imagine Dragons	Thunder	00:03:07	1
12	Imagine Dragons	Radioactive	00:03:08	1
13	Imagine Dragons	On Top of the World	00:03:12	1
14	Billie Eilish	bury a friend	00:03:13	1
15	Billie Eilish	Bad Guy	00:03:14	1
16	Shakira	Whenever, Wherever	00:03:16	1
17	Drake	God's Plan	00:03:19	1
18	The Weeknd	Blinding Lights	00:03:20	1
19	Arctic Monkeys	R U Mine?	00:03:22	1
20	Twenty One Pilots	Stressed Out	00:03:22	1
21	Imagine Dragons	Believer	00:03:24	1
22	Boney M	Daddy Cool	00:03:26	1
23	Kendrick Lamar	ELEMENT.	00:03:26	1
24	Travis Scott	Astrothunder	00:03:27	1
25	Kanye West	Good Life	00:03:27	1
26	Kendrick Lamar	LOVE.	00:03:32	2

Total rows: 100 of 100

Query complete 00:00:00.184

Ln 3, Col 45

Part 4. Window Functions 2

1. The track person listened and the previous track of the listened track:

Query

Query History

1

2

3

4

5

6

7

8

SELECT

u.name, t.title,

LAG(t.title) OVER (

PARTITION BY lh.user_id

ORDER BY lh.play_date

) AS prev_track

FROM listening_histories lh

JOIN users u ON lh.user_id = u.user_id

JOIN tracks t ON lh.track_id = t.track_id;

Data Output

Messages

Notifications

≡+

▼

▼

SQL

	name <div>character varying (100)</div>	title <div>character varying (200)</div>	prev_track <div>character varying</div>
1	Aidar Nurgaliyev	Bohemian Rhapsody	[null]
2	Aidar Nurgaliyev	Somebody to Love	Bohemian Rhapsody
3	Michael Johnson	Enter Sandman	[null]
4	Daniela Rossi	Circles	[null]
5	Alex Kim	Hotline Bling	[null]
6	Gulnara Seitova	Rivers of Babylon	[null]
7	Gulnara Seitova	Rasputin	Rivers of Babylon
8	John Miller	Stressed Out (Live)	[null]
9	John Miller	Sicko Mode	Stressed Out (Live)
10	Maria Petrova	Bad Guy	[null]
11	Emma Thompson	Hello	[null]
12	Carlos Silva	Goosebumps	[null]
13	Satoshi Yamamoto	Blinding Lights	[null]
14	Lena Müller	On Top of the World	[null]
15	Olga Ivanova	Everything I Wanted	[null]

2. The same, but next track after listened track:

Query		Query History	
1	SELECT		
2	u.name,		
3	t.title,		
4	LEAD(t.title) OVER (PARTITION BY lh.user_id ORDER BY lh.play_date) AS next_track		
5	FROM listening_histories lh		
6	JOIN users u ON lh.user_id = u.user_id		
7	JOIN tracks t ON lh.track_id = t.track_id;		
Data Output		Messages	Notifications
<div><div><div>+</div><div>SQL</div></div></div>			
	name character varying (100)	title character varying (200)	next_track character varying
1	Aidar Nurgaliyev	Bohemian Rhapsody	Somebody to Love
2	Aidar Nurgaliyev	Somebody to Love	[null]
3	Michael Johnson	Enter Sandman	[null]
4	Daniela Rossi	Circles	[null]
5	Alex Kim	Hotline Bling	[null]
6	Gulnara Seitova	Rivers of Babylon	Rasputin
7	Gulnara Seitova	Rasputin	[null]
8	John Miller	Stressed Out (Live)	Sicko Mode
9	John Miller	Sicko Mode	[null]
10	Maria Petrova	Bad Guy	[null]
11	Emma Thompson	Hello	[null]
12	Carlos Silva	Goosebumps	[null]
13	Satoshi Yamamoto	Blinding Lights	[null]
14	Lena Müller	On Top of the World	[null]
15	Olga Ivanova	Everything I Wanted	[null]

3. The artist's the shortest track:

Query

Query History

1

▼

SELECT a.name, t.title, t.duration,

2

FIRST_VALUE(t.title) OVER (PARTITION BY a.artist_id ORDER BY t.duration)

3

FROM tracks t

4

JOIN artists a ON t.artist_id = a.artist_id;

Data Output

Messages

Notifications

≡

📄

▼

📋

▼

🗑

🔍

⬇

📈

SQL

	name character varying (150)	title character varying (200)	duration interval	first_value character varying
1	Queen	Killer Queen	00:03:01	Killer Queen
2	Queen	Another One Bites The Dust	00:03:35	Killer Queen
3	Queen	Love of My Life	00:03:38	Killer Queen
4	Queen	Under Pressure	00:04:08	Killer Queen
5	Queen	I Want to Break Free	00:04:18	Killer Queen
6	Queen	Somebody to Love	00:04:56	Killer Queen
7	Queen	Somebody to Love (Live)	00:05:03	Killer Queen
8	Queen	Bohemian Rhapsody	00:05:55	Killer Queen
9	Queen	Bohemian Rhapsody (Live Aid)	00:06:30	Killer Queen
10	Imagine Dragons	Demons	00:02:57	Demons
11	Imagine Dragons	Thunder	00:03:07	Demons
12	Imagine Dragons	Radioactive	00:03:08	Demons
13	Imagine Dragons	On Top of the World	00:03:12	Demons
14	Imagine Dragons	Believer	00:03:24	Demons
15	Coldplay	Viva La Vida	00:04:02	Viva La Vida
16	Coldplay	Yellow	00:04:29	Viva La Vida
17	Coldplay	Fix You	00:04:55	Viva La Vida
18	Coldplay	Clocks	00:05:07	Viva La Vida
19	Coldplay	The Scientist	00:05:09	Viva La Vida
20	Metallica	Enter Sandman	00:05:32	Enter Sandman
21	Metallica	Nothing Else Matters (Live)	00:06:22	Enter Sandman
22	Metallica	Nothing Else Matters	00:06:28	Enter Sandman
23	Metallica	Fade to Black	00:06:57	Enter Sandman
24	Metallica	Master of Puppets	00:08:35	Enter Sandman
25	Billie Eilish	bury a friend	00:03:13	bury a friend

Total rows: 100 of 100 Query complete 00:00:00.348 Ln 4, Col 45

4. The artist's the longest track:

Query

Query History

1

2

3

4

5

6

7

8

SELECT a.name, t.title, t.duration,
LAST_VALUE(t.title) OVER (
PARTITION BY a.artist_id
ORDER BY t.duration
ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING
)
FROM tracks t
JOIN artists a ON t.artist_id = a.artist_id;

Data Output

Messages

Notifications

<

5. The artist's 3rd the longest track

Query
Query History

```

1 SELECT a.name, t.title, t.duration,
2     NTH_VALUE(t.title, 3) OVER(
3         PARTITION BY a.artist_id
4         ORDER BY t.duration
5         ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING
6     )
7 FROM tracks t
8 JOIN artists a ON t.artist_id = a.artist_id;

```

Data Output
Messages
Notifications

≡

▼

▼

SQL

	name character varying (150)	title character varying (200)	duration interval	nth_value character varying
1	Queen	Killer Queen	00:03:01	Love of My Life
2	Queen	Another One Bites The Dust	00:03:35	Love of My Life
3	Queen	Love of My Life	00:03:38	Love of My Life
4	Queen	Under Pressure	00:04:08	Love of My Life
5	Queen	I Want to Break Free	00:04:18	Love of My Life
6	Queen	Somebody to Love	00:04:56	Love of My Life
7	Queen	Somebody to Love (Live)	00:05:03	Love of My Life
8	Queen	Bohemian Rhapsody	00:05:55	Love of My Life
9	Queen	Bohemian Rhapsody (Live Aid)	00:06:30	Love of My Life
10	Imagine Dragons	Demons	00:02:57	Radioactive
11	Imagine Dragons	Thunder	00:03:07	Radioactive
12	Imagine Dragons	Radioactive	00:03:08	Radioactive
13	Imagine Dragons	On Top of the World	00:03:12	Radioactive
14	Imagine Dragons	Believer	00:03:24	Radioactive
15	Coldplay	Viva La Vida	00:04:02	Fix You
16	Coldplay	Yellow	00:04:29	Fix You
17	Coldplay	Fix You	00:04:55	Fix You
18	Coldplay	Clocks	00:05:07	Fix You
19	Coldplay	The Scientist	00:05:09	Fix You
20	Metallica	Enter Sandman	00:05:32	Nothing Else Matters
21	Metallica	Nothing Else Matters (Live)	00:06:22	Nothing Else Matters
22	Metallica	Nothing Else Matters	00:06:28	Nothing Else Matters

Total rows: 100 of 100
Query complete 00:00:00.457
Ln 8, Col 45

6. Artist's track position by duration in descending order:

Query

Query History

1

2

3

4

5

6

7

SELECT a.name, t.title, t.duration,

RANK() OVER (

PARTITION BY a.artist_id

ORDER BY t.duration DESC

)

FROM tracks t

JOIN artists a ON t.artist_id = a.artist_id;

Data Output

Messages

Notifications

7. Amount of tracks that user listened since registration

Query

Query History

1

2

3

4

5

6

7

8

```

SELECT u.name, lh.play_date,
      COUNT(lh.track_id) OVER (
        PARTITION BY u.user_id
        ORDER BY lh.play_date
        ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
      )
FROM listening_histories lh
JOIN users u ON lh.user_id = u.user_id;

```

Data Output

Messages

Notifications

≡+

📄

▼

📋

▼

🗑️

🗄️

⬇️

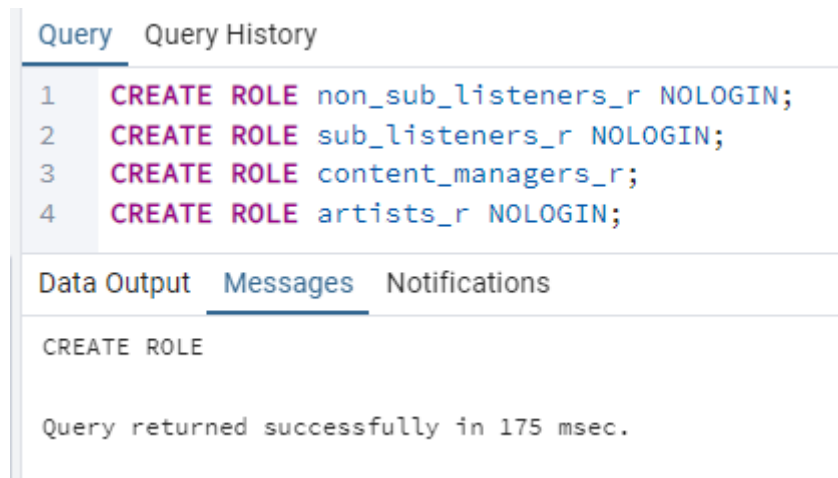
📈

SQL

	name character varying (100) 🔒	play_date timestamp without time zone 🔒	count bigint 🔒
1	Aidar Nurgaliyev	2025-10-10 12:00:00	1
2	Aidar Nurgaliyev	2025-10-10 12:05:00	2
3	Michael Johnson	2025-09-29 22:22:00	1
4	Daniela Rossi	2025-10-03 16:16:00	1
5	Alex Kim	2025-10-02 19:45:00	1
6	Gulnara Seitova	2025-09-20 13:10:00	1
7	Gulnara Seitova	2025-09-30 09:12:00	2
8	John Miller	2025-09-21 14:14:00	1
9	John Miller	2025-10-01 18:22:00	2
10	Maria Petrova	2025-09-25 20:00:00	1
11	Emma Thompson	2025-10-08 14:33:00	1
12	Carlos Silva	2025-10-09 23:01:00	1
13	Satoshi Yamamoto	2025-10-07 07:44:00	1
14	Lena Müller	2025-10-06 06:30:00	1
15	Olga Ivanova	2025-10-05 11:11:00	1

Part 5. Roles

According to Part 0, we need to create user groups:



The screenshot shows a SQL query execution interface. At the top, there are two tabs: 'Query' and 'Query History'. The 'Query' tab is active. Below the tabs, there is a list of four SQL statements, each preceded by a line number (1 to 4). The statements are: 1. CREATE ROLE non_sub_listeners_r NOLOGIN; 2. CREATE ROLE sub_listeners_r NOLOGIN; 3. CREATE ROLE content_managers_r; 4. CREATE ROLE artists_r NOLOGIN;. Below the list of statements, there are three tabs: 'Data Output', 'Messages', and 'Notifications'. The 'Messages' tab is active. Below the 'Messages' tab, the text 'CREATE ROLE' is displayed. At the bottom, a message states 'Query returned successfully in 175 msec.'

```
1 CREATE ROLE non_sub_listeners_r NOLOGIN;
2 CREATE ROLE sub_listeners_r NOLOGIN;
3 CREATE ROLE content_managers_r;
4 CREATE ROLE artists_r NOLOGIN;
```

CREATE ROLE

Query returned successfully in 175 msec.

This query created 4 group roles, and there are:

1. Listeners without subscription
2. Listeners with subscription
3. Content Managers
4. Artists (who only can interact with their album, tracks)

P.S._r at the end means role

And the next step is to create 2 users for each group:

Query	Query History
1	CREATE ROLE listener_nursultan LOGIN PASSWORD 'nirmanych123' INHERIT;
2	CREATE ROLE listener_kamal LOGIN PASSWORD 'kamalgamri321' INHERIT;
3	GRANT non_sub_listeners_r TO listener_nursultan;
4	GRANT non_sub_listeners_r TO listener_kamal;
5	
6	CREATE ROLE subscribed_listener_batyrkhan LOGIN PASSWORD 'qwerty123' INHERIT;
7	CREATE ROLE subscribed_listener_alinur LOGIN PASSWORD 'allurlimur' INHERIT;
8	GRANT sub_listeners_r TO subscribed_listener_batyrkhan;
9	GRANT sub_listeners_r TO subscribed_listener_alinur;
10	
11	CREATE ROLE manager_marat LOGIN PASSWORD 'maratik228';
12	CREATE ROLE manager_sultan LOGIN PASSWORD 'sultikbultik';
13	GRANT content_managers_r TO manager_marat;
14	GRANT content_managers_r TO manager_sultan;
15	
16	CREATE ROLE artist_queen LOGIN PASSWORD 'RestInPeaceFreddieMercury';
17	CREATE ROLE artist_kendricklamar LOGIN PASSWORD 'TheyNotLikeUsKL';
18	GRANT artists_r TO artist_queen;
19	GRANT artists_r TO artist_kendricklamar;


Data Output	Messages	Notifications
GRANT ROLE		
Query returned successfully in 204 msec.		

There are users that have been created are:

1. Nursultan, Kamal (non-subscribed listeners)
2. Batyrkhan, Alinur (subscribed listeners)
3. Marat, Sultan (content managers)
4. Queen, Kendrick Lamar (artists)

Part 6. Privileges

At the previous part we created user groups and 2 users for each group. Now we need to set privileges for each group role such as it's written in the descriptions in part 0:



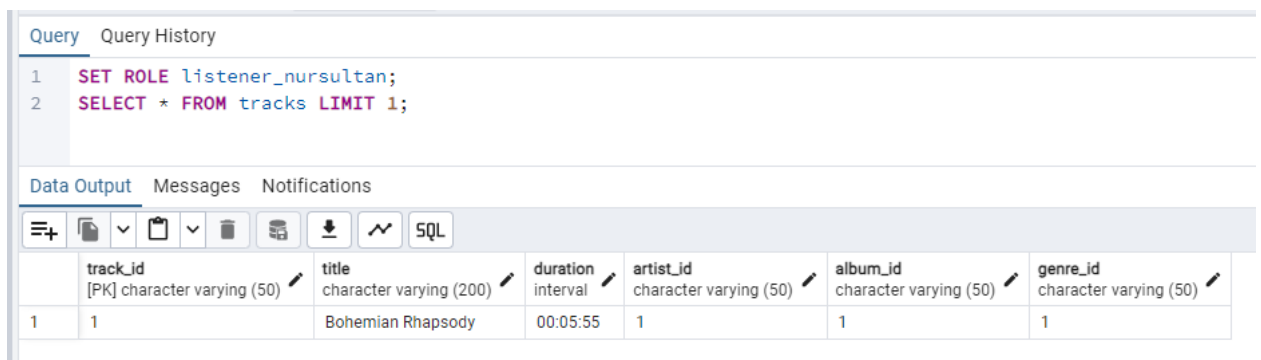
The screenshot shows a SQL query editor with a 'Query' tab. The query is a multi-line GRANT statement. Below the query, the 'Messages' tab is active, showing a 'GRANT' message and a confirmation that the query returned successfully in 340 msec.

```
1 GRANT SELECT ON tracks, artists, albums, genres, countries, subscriptions,  
2     playlists, playlist_track, listening_histories, recommendations TO non_sub_listeners_r;  
3 GRANT INSERT, UPDATE ON ratings TO non_sub_listeners_r;  
4  
5 GRANT SELECT ON tracks, artists, albums, genres, countries, subscriptions, recommendations TO sub_listeners_r;  
6 GRANT SELECT, INSERT, UPDATE ON playlists, playlist_track, listening_histories, ratings TO sub_listeners_r;  
7  
8 GRANT SELECT, INSERT, UPDATE ON artists, albums, tracks, genres, charts, charttrack TO content_managers_r;  
9 GRANT SELECT ON countries, subscriptions, users, playlists, ratings, recommendations TO content_managers_r;  
10  
11 GRANT SELECT ON artists, charts, charttrack, countries, subscriptions, users, playlists, ratings, recommendations TO artists_r;  
12 GRANT SELECT, INSERT, UPDATE ON albums, tracks, genres TO artists_r;
```

GRANT

Query returned successfully in 340 msec.

Now let's check if everything is correct. Let's start with the non-subscribed listeners. The only can insert or update ratings, but can't change anything else:



The screenshot shows a SQL query editor with a 'Query' tab. The query consists of two lines: SET ROLE listener_nursultan; and SELECT * FROM tracks LIMIT 1;. Below the query, the 'Data Output' tab is active, displaying a table with 7 columns and 1 row of data.

```
1 SET ROLE listener_nursultan;  
2 SELECT * FROM tracks LIMIT 1;
```

	track_id [PK] character varying (50)	title character varying (200)	duration interval	artist_id character varying (50)	album_id character varying (50)	genre_id character varying (50)
1	1	Bohemian Rhapsody	00:05:55	1	1	1

Select works!

Now rating insert rating:

Query	Query History
1	<code>SET ROLE listener_nursultan;</code>
2	<code>INSERT INTO ratings(user_id, track_id, rating) VALUES (1, 2, 5)</code>

Data Output	Messages	Notifications
INSERT 0 1		
Query returned successfully in 316 msec.		

And what if I delete the rating?

Query	Query History
1	<code>SET ROLE listener_nursultan;</code>
2	<code>DELETE FROM ratings WHERE user_id = '1' AND track_id = '2'</code>

Data Output	Messages	Notifications
ERROR: permission denied for table ratings		
SQL state: 42501		


Permission denied. That's right, I didn't give DELETE permission for deleting rating. Only insert and update.

The next is subscribed listener and as subscribed listener I want to create my playlist and then try to create new track for artist in database (music platform):

Query	Query History
1 SET ROLE subscribed_listener_batyrkhan; 2 INSERT INTO Playlists(playlist_id, title, user_id) VALUES (999, 'Test', '10');	

Data Output	Messages	Notifications
INSERT 0 1		
Query returned successfully in 253 msec.		

It works, now trying to create a track:

Query	Query History
1  INSERT INTO Tracks (track_id, title, duration, artist_id, album_id, genre_id) 2 VALUES (101, 'Test Track', INTERVAL '3 minutes 30 seconds', 1, 1, 1); 3	

Data Output	Messages	Notifications
ERROR: permission denied for table tracks		
SQL state: 42501		

Works fine. Permission denied.

The next is content manager. Content manager can create artist, but not countries or users:

Query	Query History
1	SET ROLE manager_marat;
2	INSERT INTO artists(artist_id, name, bio) VALUES (999, 'MOLDANAZAR', 'Kazakh pop artist')

Data Output	Messages	Notifications
INSERT 0 1		
Query returned successfully in 195 msec.		

And trying to create country:

Query	Query History
1	SET ROLE manager_marat;
2	INSERT INTO countries(country_id, name) VALUES (999, 'Non-existing Country');

Data Output	Messages	Notifications
ERROR: permission denied for table countries		
SQL state: 42501		

Great!

And the last one is artist. Artist can create track or album, but can't interact with chart.

Query	Query History
1	SET ROLE artist_queen;
2	INSERT INTO albums(album_id, title, release_date, artist_id)
3	VALUES (999, 'Bohemian Rhapsody Movie Soundtrack', '2018-10-19', 1);

Data Output	Messages	Notifications
INSERT 0 1		
Query returned successfully in 585 msec.		

Album has been created. Now I need to create a track for this album, then what if I try to push it to chart?

Query	Query History
1	SET ROLE artist_queen;
2	INSERT INTO tracks(track_id, title, duration, artist_id, album_id, genre_id)
3	VALUES (999, 'Don't Stop Me Now (Revisited)', INTERVAL '3 minutes 29 seconds', 1, 999, 1);

Data Output	Messages	Notifications
INSERT 0 1		
Query returned successfully in 198 msec.		

Query	Query History
1	SET ROLE artist_queen;
2	INSERT INTO ChartTrack (chart_id, track_id, position)
3	VALUES (1, 999, 1);

Data Output	Messages	Notifications
ERROR: permission denied for table charttrack		
SQL state: 42501		

Error: Permission denied. Great! Everything works as expected!