DATA VISUALIZATION PROJECT

Group members :

Nogaye Moundaw DIENG

Nada NADIRE

Marliyatou Touanga DIALLO

Arthy UTHAYARAJAH

DATA VISUALIZATION PROJECT

https://github.com/nada912/Tracks-Record

Start and run the project

Prerequisites:

Have a data bricks account (If not, create an account with your professional email on databricks).

Step 1: Create a cluster on Databricks

On the menu bar, click on “Compute” then “Create cluster” button.

Choose a name then create.

Step 2: Import the data to Databricks

On the menu bar, click on “Catalog”, then “Create table” button.
Name the target directory “data” then upload all the text files at once.

Step 3: Open the notebook

On the menu bar, click “New” -> “Notebook”.

Once the notebook is created, click on “File” -> “Import” to import the notebook available on Github.

Now execute the cells to get and process the data then load it to the database.

Step 4: At this point everything is done, you can visualize the dashboard on PowerBi.

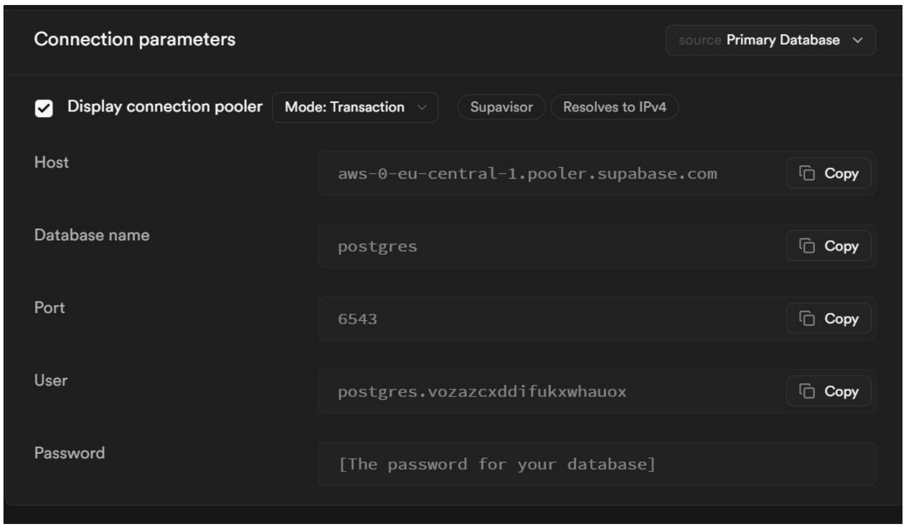
Connect To PowerBi and select “Get Data from other source”

Select on the left sidebar “Database” -> “PostgreSQL Database”

Connect with the following and open the Report Tracks.pbix file:

Host : http://aws-0-eu-central-1.pooler.supabase.com
Database name: postgre
user : postgres.vozazcxddifukxwhauox
password: PasswordTracks2024.

Reconnect to the database in case you have trouble visualizing the dashboard


Connection parameters

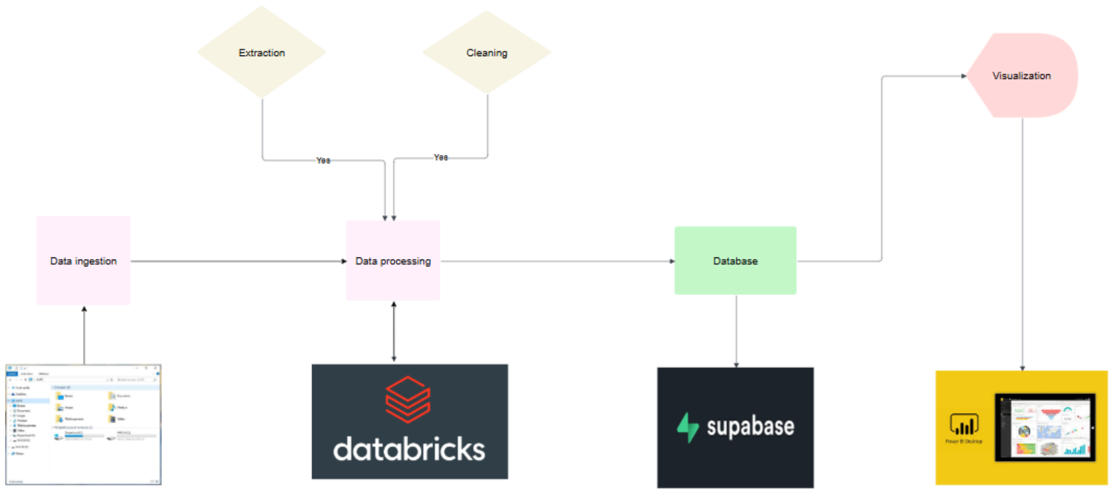
source Primary Database

Display connection pooler [✓] Mode: Transaction · Supavisor · Resolves to IPv4

Host	aws-0-eu-central-1.pooler.supabase.com
Database name	postgres
Port	6543
User	postgres.vozazcxddifukxwhauox
Password	[The password for your database]

Conception and implementation

Process and tools:



To achieve this project, we have gone through several steps and used some tools in order to first retrieve the data, clean it, load it to the database and finally visualize it.

Data ingestion :

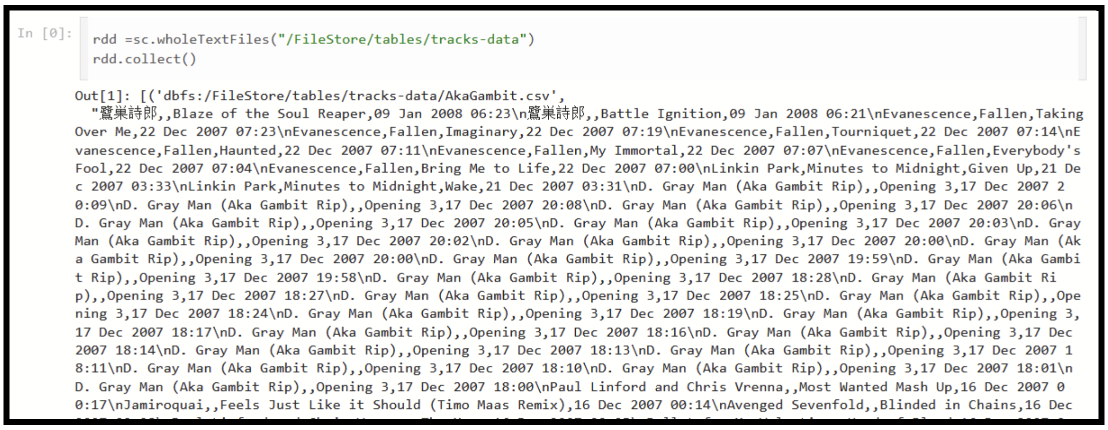
Given the context of the project, the data was available on Teams, so we downloaded the files locally on our computers.

Data processing:

For this part we used *Databricks*, which is a powerful platform for big data processing, allowing us to clean, transform, and process our datasets efficiently.

It supports distributed computing, enabling scalability for complex operations on large amounts of data and it greatly integrates various data sources including databases.

Loading Data:
The data was initially loaded from text files into our Resilient Distributed Dataset (RDD) using the *WholeTextFiles* method. This method reads all files from a specified directory containing the .txt files and returns a pair consisting of the file path and the content of the file.

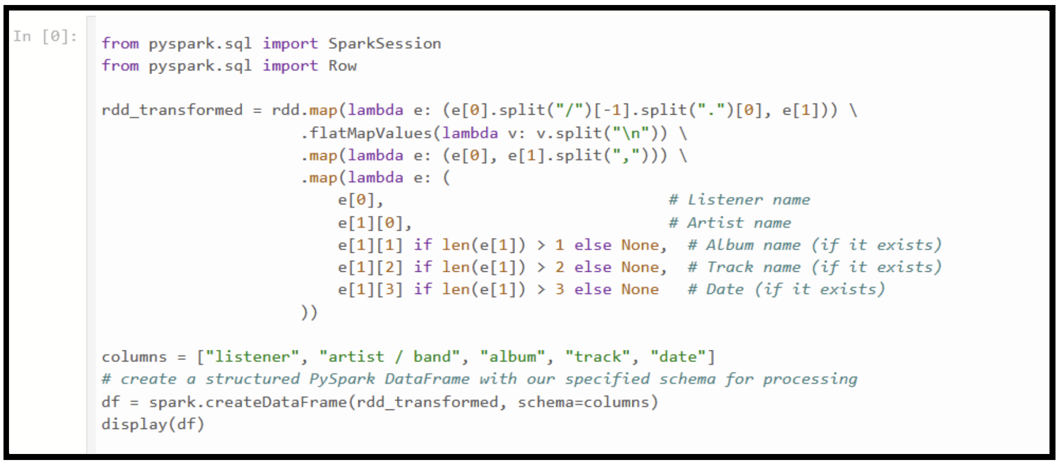


```
In [0]:  rdd =sc.wholeTextFiles("/FileStore/tables/tracks-data")
         rdd.collect()

         Out[1]: [('dbfs:/FileStore/tables/tracks-data/AkaGambit.csv',
           "鷺巣詩郎,,Blaze of the Soul Reaper,09 Jan 2008 06:23\n鷺巣詩郎,,Battle Ignition,09 Jan 2008 06:21\nEvanescence,Fallen,Taking
         Over Me,22 Dec 2007 07:23\nEvanescence,Fallen,Imaginary,22 Dec 2007 07:19\nEvanescence,Fallen,Tourniquet,22 Dec 2007 07:14\nE
         vanescence,Fallen,Haunted,22 Dec 2007 07:11\nEvanescence,Fallen,My Immortal,22 Dec 2007 07:07\nEvanescence,Fallen,Everybody's
         Fool,22 Dec 2007 07:04\nEvanescence,Fallen,Bring Me to Life,22 Dec 2007 07:00\nLinkin Park,Minutes to Midnight,Given Up,21 De
         c 2007 03:33\nLinkin Park,Minutes to Midnight,Wake,21 Dec 2007 03:31\nD. Gray Man (Aka Gambit Rip),,Opening 3,17 Dec 2007 2
         0:09\nD. Gray Man (Aka Gambit Rip),,Opening 3,17 Dec 2007 20:08\nD. Gray Man (Aka Gambit Rip),,Opening 3,17 Dec 2007 20:06\n
         D. Gray Man (Aka Gambit Rip),,Opening 3,17 Dec 2007 20:05\nD. Gray Man (Aka Gambit Rip),,Opening 3,17 Dec 2007 20:03\nD. Gray
         Man (Aka Gambit Rip),,Opening 3,17 Dec 2007 20:02\nD. Gray Man (Aka Gambit Rip),,Opening 3,17 Dec 2007 20:00\nD. Gray Man (Ak
         a Gambit Rip),,Opening 3,17 Dec 2007 20:00\nD. Gray Man (Aka Gambit Rip),,Opening 3,17 Dec 2007 19:59\nD. Gray Man (Aka Gambi
         t Rip),,Opening 3,17 Dec 2007 19:58\nD. Gray Man (Aka Gambit Rip),,Opening 3,17 Dec 2007 18:28\nD. Gray Man (Aka Gambit Ri
         p),,Opening 3,17 Dec 2007 18:27\nD. Gray Man (Aka Gambit Rip),,Opening 3,17 Dec 2007 18:25\nD. Gray Man (Aka Gambit Rip),,Ope
         ning 3,17 Dec 2007 18:24\nD. Gray Man (Aka Gambit Rip),,Opening 3,17 Dec 2007 18:19\nD. Gray Man (Aka Gambit Rip),,Opening 3,
         17 Dec 2007 18:17\nD. Gray Man (Aka Gambit Rip),,Opening 3,17 Dec 2007 18:16\nD. Gray Man (Aka Gambit Rip),,Opening 3,17 Dec
         2007 18:14\nD. Gray Man (Aka Gambit Rip),,Opening 3,17 Dec 2007 18:13\nD. Gray Man (Aka Gambit Rip),,Opening 3,17 Dec 2007 1
         8:11\nD. Gray Man (Aka Gambit Rip),,Opening 3,17 Dec 2007 18:10\nD. Gray Man (Aka Gambit Rip),,Opening 3,17 Dec 2007 18:01\n
         D. Gray Man (Aka Gambit Rip),,Opening 3,17 Dec 2007 18:00\nPaul Linford and Chris Vrenna,,Most Wanted Mash Up,16 Dec 2007 0
         0:17\nJamiroquai,,Feels Just Like it Should (Timo Maas Remix),16 Dec 2007 00:14\nAvenged Sevenfold,,Blinded in Chains,16 Dec
```

Data Extraction:

The raw data within the RDD was processed to extract the listener names from the file paths. Additionally, the content was split into individual records, each representing a unique music listening event. Each record was parsed into five distinct fields: listener name, artist/band, album, track, and date.

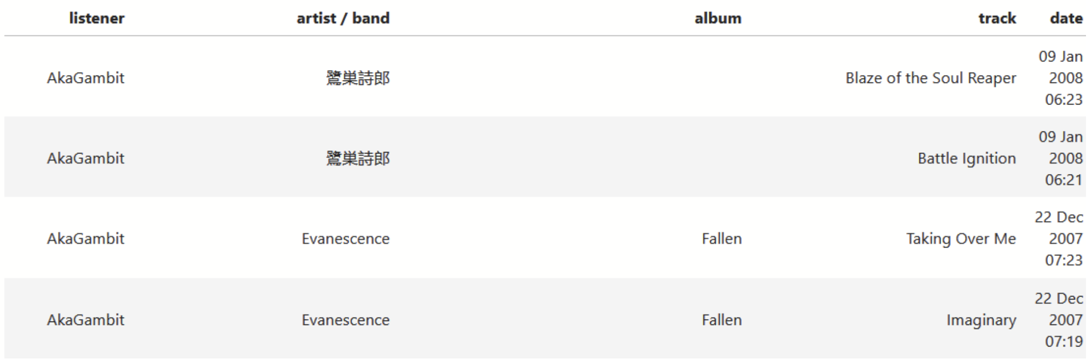


```
In [0]:  from pyspark.sql import SparkSession
         from pyspark.sql import Row

         rdd_transformed = rdd.map(lambda e: (e[0].split("/")[-1].split(".")[0], e[1])) \
                              .flatMapValues(lambda v: v.split("\n")) \
                              .map(lambda e: (e[0], e[1].split(","))) \
                              .map(lambda e: (
                                  e[0],                                # Listener name
                                  e[1][0],                             # Artist name
                                  e[1][1] if len(e[1]) > 1 else None,  # Album name (if it exists)
                                  e[1][2] if len(e[1]) > 2 else None,  # Track name (if it exists)
                                  e[1][3] if len(e[1]) > 3 else None   # Date (if it exists)
                              ))

         columns = ["listener", "artist / band", "album", "track", "date"]
         # create a structured PySpark DataFrame with our specified schema for processing
         df = spark.createDataFrame(rdd_transformed, schema=columns)
         display(df)
```

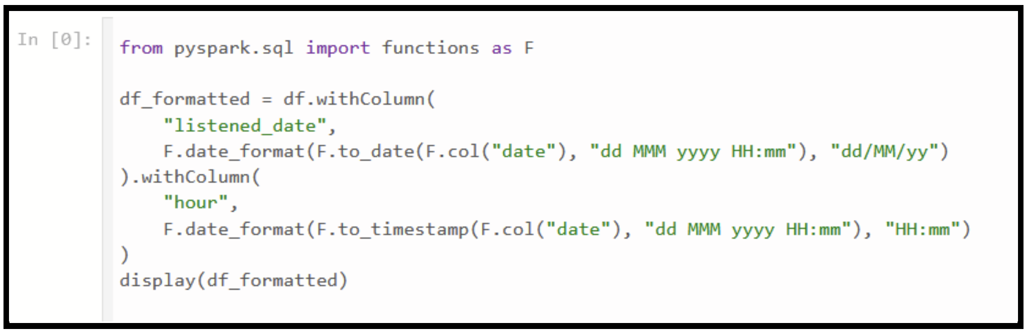
Here is a preview of the result we get after the first transformation:



listener	artist / band	album	track	date
AkaGambit	鷺巣詩郎		Blaze of the Soul Reaper	09 Jan 2008 06:23
AkaGambit	鷺巣詩郎		Battle Ignition	09 Jan 2008 06:21
AkaGambit	Evanescence	Fallen	Taking Over Me	22 Dec 2007 07:23
AkaGambit	Evanescence	Fallen	Imaginary	22 Dec 2007 07:19

Date Column Processing:

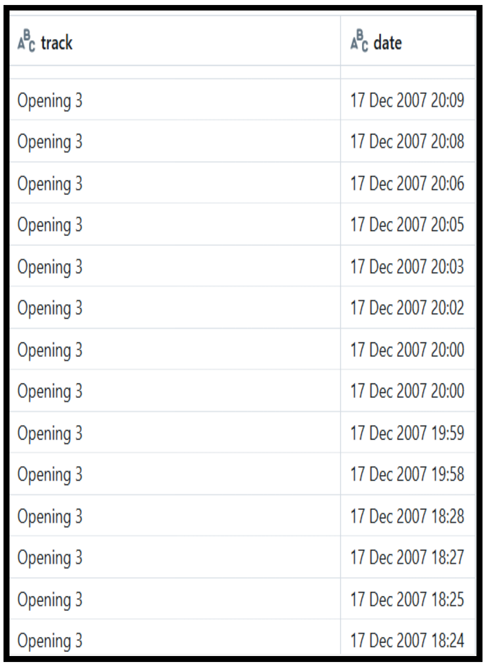
The original "date" column was split into two separate columns: listened_date and hour. This transformation allowed for more granular analysis of the listening behavior, separating the date from the time of the event.



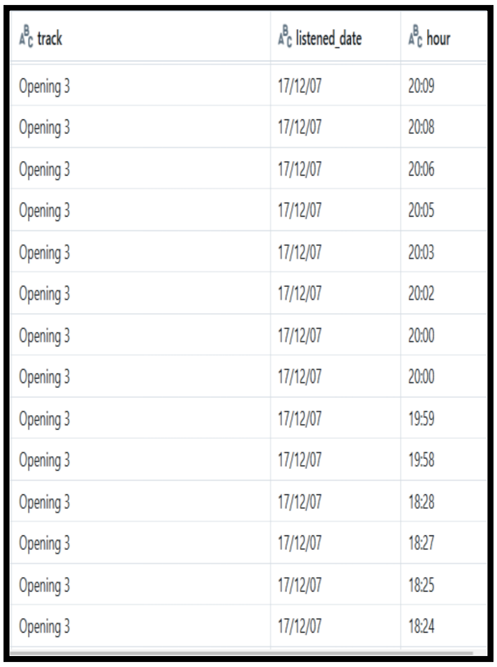
```
In [0]:  from pyspark.sql import functions as F

         df_formatted = df.withColumn(
             "listened_date",
             F.date_format(F.to_date(F.col("date"), "dd MMM yyyy HH:mm"), "dd/MM/yy")
         ).withColumn(
             "hour",
             F.date_format(F.to_timestamp(F.col("date"), "dd MMM yyyy HH:mm"), "HH:mm")
         )
         display(df_formatted)
```

This is what we get:



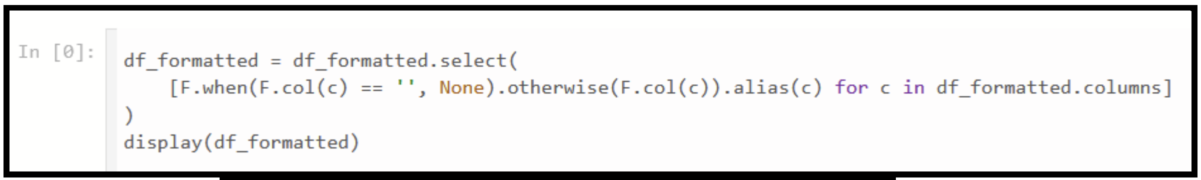
track	date
Opening 3	17 Dec 2007 20:09
Opening 3	17 Dec 2007 20:08
Opening 3	17 Dec 2007 20:06
Opening 3	17 Dec 2007 20:05
Opening 3	17 Dec 2007 20:03
Opening 3	17 Dec 2007 20:02
Opening 3	17 Dec 2007 20:00
Opening 3	17 Dec 2007 20:00
Opening 3	17 Dec 2007 19:59
Opening 3	17 Dec 2007 19:58
Opening 3	17 Dec 2007 18:28
Opening 3	17 Dec 2007 18:27
Opening 3	17 Dec 2007 18:25
Opening 3	17 Dec 2007 18:24



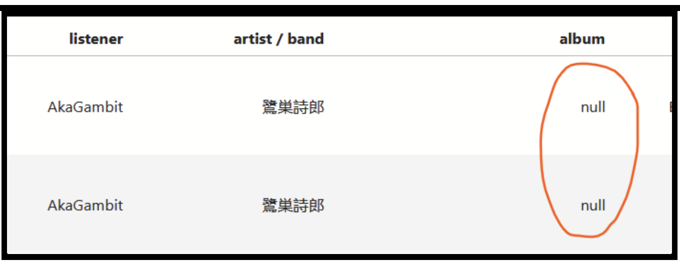
track	listened_date	hour
Opening 3	17/12/07	20:09
Opening 3	17/12/07	20:08
Opening 3	17/12/07	20:06
Opening 3	17/12/07	20:05
Opening 3	17/12/07	20:03
Opening 3	17/12/07	20:02
Opening 3	17/12/07	20:00
Opening 3	17/12/07	20:00
Opening 3	17/12/07	19:59
Opening 3	17/12/07	19:58
Opening 3	17/12/07	18:28
Opening 3	17/12/07	18:27
Opening 3	17/12/07	18:25
Opening 3	17/12/07	18:24

Handling Null Values:

Empty cells in the dataset were set to null, ensuring that missing values were appropriately handled. The original date column was then dropped as its information had been separated into the new listened_date and hour columns.



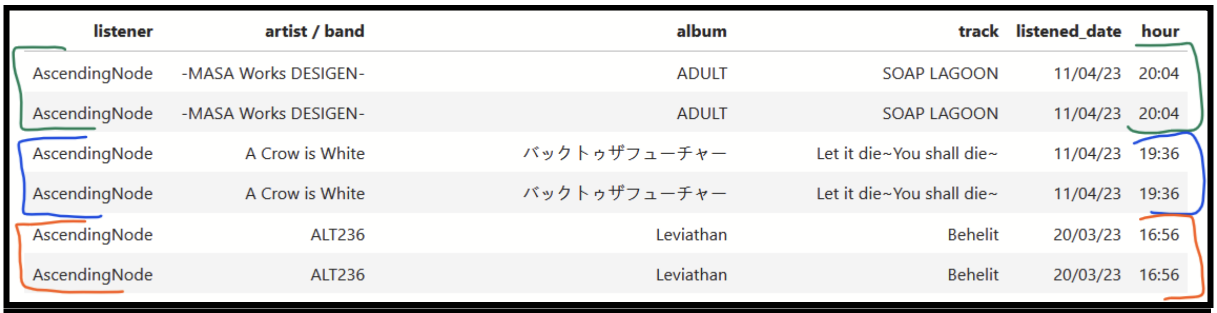
```
In [0]:  df_formatted = df_formatted.select(
             [F.when(F.col(c) == '', None).otherwise(F.col(c)).alias(c) for c in df_formatted.columns]
         )
         display(df_formatted)
```



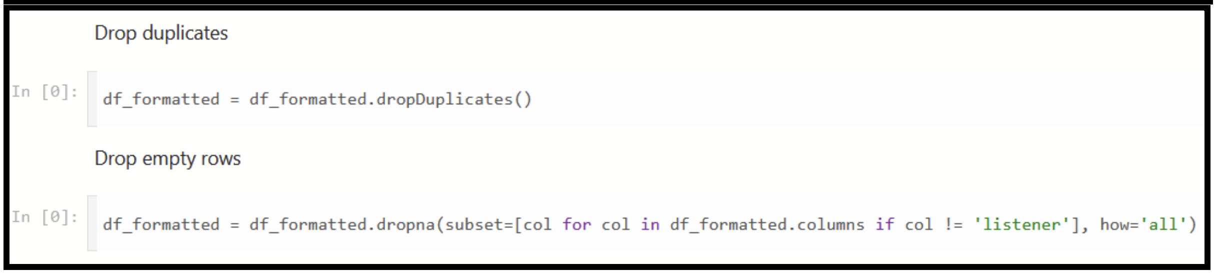
listener	artist / band	album
AkaGambit	鷺巣詩郎	null
AkaGambit	鷺巣詩郎	null

Removing Duplicates and Empty Rows:

As can be seen below, there were quite a bit of duplicates in the dataset. We then had to make sure that any rows that were redundant or completely empty were removed from the dataset to ensure data integrity and reduce potential noise in the analysis.



listener	artist / band	album	track	listened_date	hour
AscendingNode	-MASA Works DESIGEN-	ADULT	SOAP LAGOON	11/04/23	20:04
AscendingNode	-MASA Works DESIGEN-	ADULT	SOAP LAGOON	11/04/23	20:04
AscendingNode	A Crow is White	バックトゥザフューチャー	Let it die~You shall die~	11/04/23	19:36
AscendingNode	A Crow is White	バックトゥザフューチャー	Let it die~You shall die~	11/04/23	19:36
AscendingNode	ALT236	Leviathan	Behelit	20/03/23	16:56
AscendingNode	ALT236	Leviathan	Behelit	20/03/23	16:56



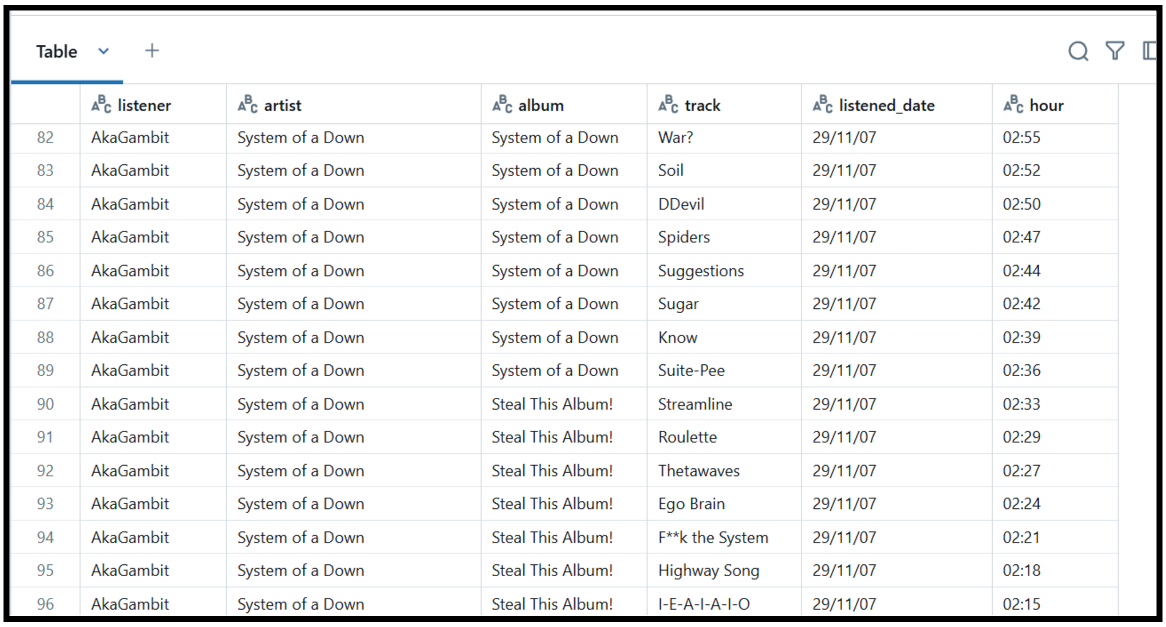
Drop duplicates

```
In [0]:  df_formatted = df_formatted.dropDuplicates()
```

Drop empty rows

```
In [0]:  df_formatted = df_formatted.dropna(subset=[col for col in df_formatted.columns if col != 'listener'], how='all')
```

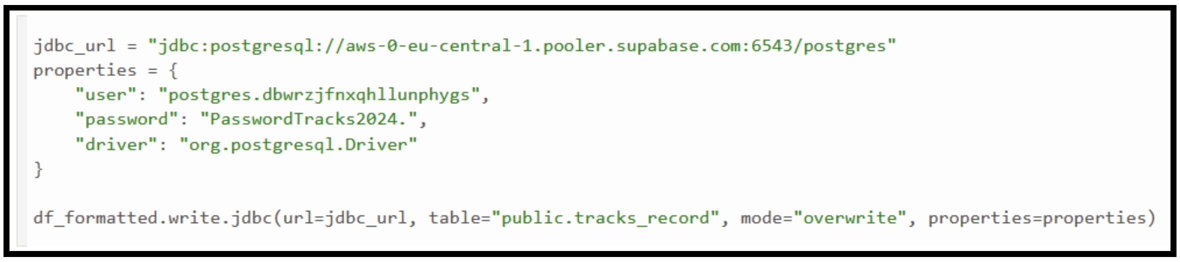
Final processed database



Table

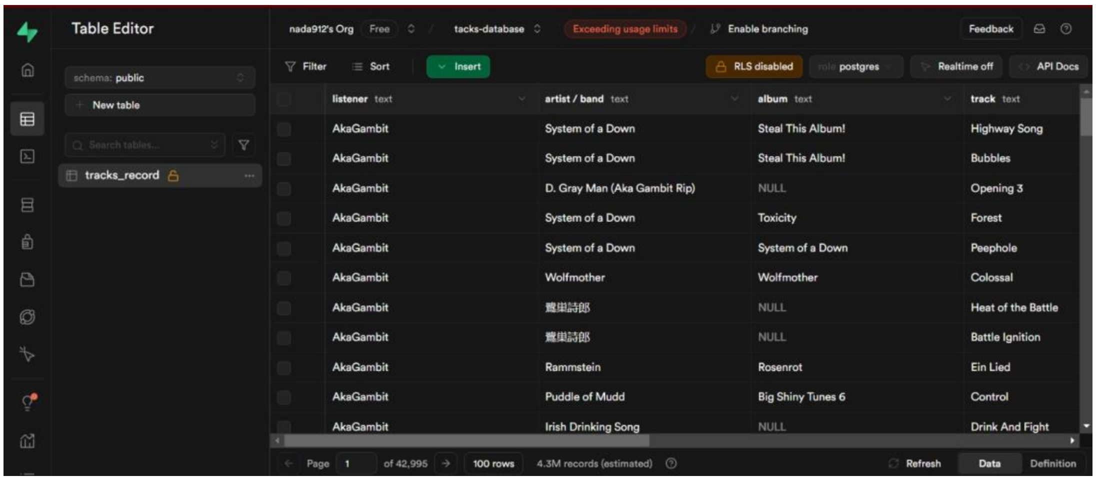
	listener	artist	album	track	listened_date	hour
82	AkaGambit	System of a Down	System of a Down	War?	29/11/07	02:55
83	AkaGambit	System of a Down	System of a Down	Soil	29/11/07	02:52
84	AkaGambit	System of a Down	System of a Down	DDevil	29/11/07	02:50
85	AkaGambit	System of a Down	System of a Down	Spiders	29/11/07	02:47
86	AkaGambit	System of a Down	System of a Down	Suggestions	29/11/07	02:44
87	AkaGambit	System of a Down	System of a Down	Sugar	29/11/07	02:42
88	AkaGambit	System of a Down	System of a Down	Know	29/11/07	02:39
89	AkaGambit	System of a Down	System of a Down	Suite-Pee	29/11/07	02:36
90	AkaGambit	System of a Down	Steal This Album!	Streamline	29/11/07	02:33
91	AkaGambit	System of a Down	Steal This Album!	Roulette	29/11/07	02:29
92	AkaGambit	System of a Down	Steal This Album!	Thetawaves	29/11/07	02:27
93	AkaGambit	System of a Down	Steal This Album!	Ego Brain	29/11/07	02:24
94	AkaGambit	System of a Down	Steal This Album!	F**k the System	29/11/07	02:21
95	AkaGambit	System of a Down	Steal This Album!	Highway Song	29/11/07	02:18
96	AkaGambit	System of a Down	Steal This Album!	I-E-A-I-A-I-O	29/11/07	02:15

Database Integration:
The cleaned and processed data was then connected to a PostgreSQL database hosted on Supabase. Here's the Identifiers to connect to our postgre database in supabase and populate it.



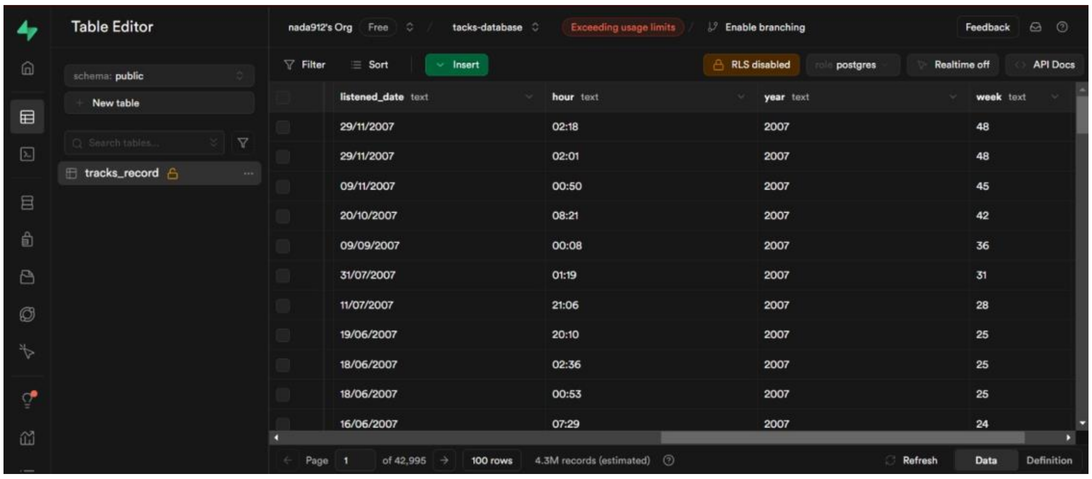
```
jdbc_url = "jdbc:postgresql://aws-0-eu-central-1.pooler.supabase.com:6543/postgres"
properties = {
    "user": "postgres.dbwrzjfnxqhllunphygs",
    "password": "PasswordTracks2024.",
    "driver": "org.postgresql.Driver"
}

df_formatted.write.jdbc(url=jdbc_url, table="public.tracks_record", mode="overwrite", properties=properties)
```



listener text	artist / band text	album text	track text
AkaGambit	System of a Down	Steal This Album!	Highway Song
AkaGambit	System of a Down	Steal This Album!	Bubbles
AkaGambit	D. Gray Man (Aka Gambit Rip)	NULL	Opening 3
AkaGambit	System of a Down	Toxicity	Forest
AkaGambit	System of a Down	System of a Down	Peephole
AkaGambit	Wolfmother	Wolfmother	Colossal
AkaGambit	鷺巣詩郎	NULL	Heat of the Battle
AkaGambit	鷺巣詩郎	NULL	Battle Ignition
AkaGambit	Rammstein	Rosenrot	Ein Lied
AkaGambit	Puddle of Mudd	Big Shiny Tunes 6	Control
AkaGambit	Irish Drinking Song	NULL	Drink And Fight

Page 1 of 42,995 100 rows 4.3M records (estimated) Refresh Data Definition



listened_date text	hour text	year text	week text
29/11/2007	02:18	2007	48
29/11/2007	02:01	2007	48
09/11/2007	00:50	2007	45
20/10/2007	08:21	2007	42
09/09/2007	00:08	2007	36
31/07/2007	01:19	2007	31
11/07/2007	21:06	2007	28
19/06/2007	20:10	2007	25
18/06/2007	02:36	2007	25
18/06/2007	00:53	2007	25
16/06/2007	07:29	2007	24

Page 1 of 42,995 100 rows 4.3M records (estimated)

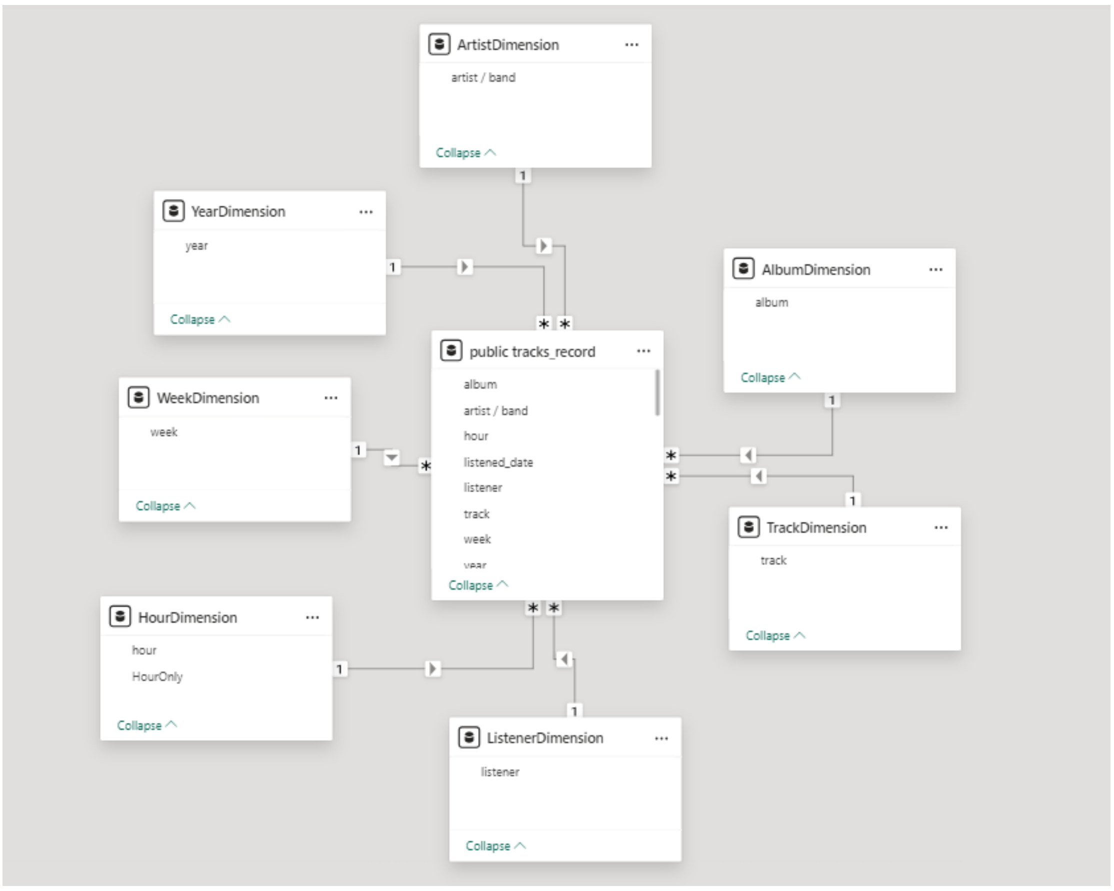
PowerBi Dashboard conception

The star model

In this project, we used a star schema to structure our data model, enhancing query performance and enabling efficient reporting. The star schema consists of a central fact table here called "**public tracks_record**" surrounded by multiple dimension tables, which simplifies data relationships and improves readability. Each table is smaller and easier to join, which makes it easier to filter and query our data.

Therefore, the dimensions table here are:

- **Listener dimension**
- **Artist dimension**
- **Album dimension**
- **Track dimension**
- **Hour dimension**
- **Week dimension**
- **Year dimension**



The requested KPIs

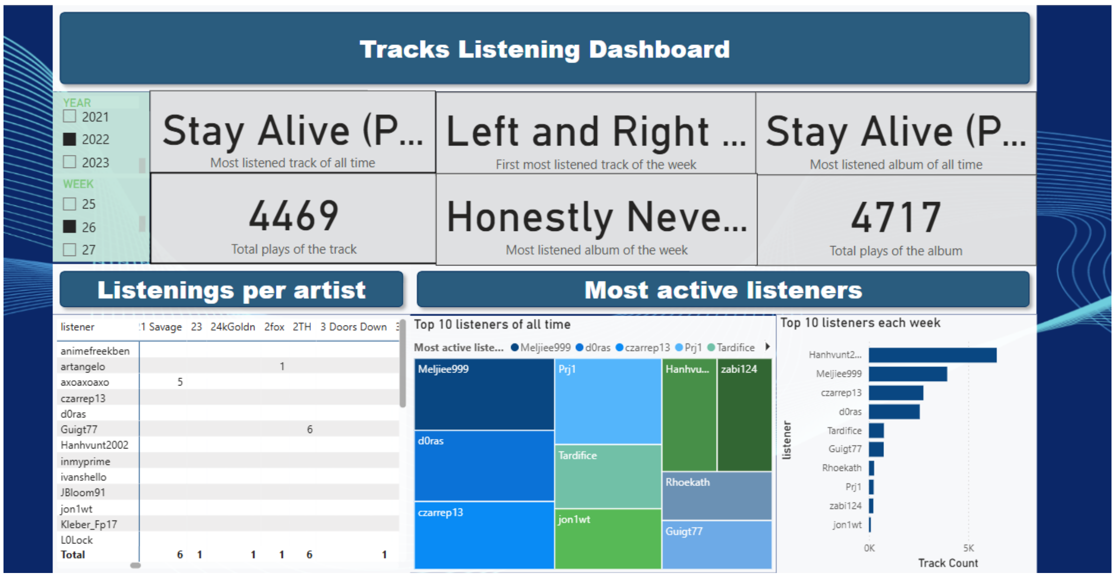
Most listened track of all time
Most listened track for each week
Most listened album of all time
Most listened album for each week
Cross tabulation of the number of tracks listened to by listener and by artist
Ranking the 10 biggest listeners of all time
Ranking the 10 biggest listeners for each week

Additional KPIs

For additional KPIs, we added:
- Top 10 most listened artists overall & per listener
- Top 10 most listened artists per year
- Top 10 most listened artists per hour
- Total of plays of the most listened track
- Total of plays of the most listened album
- Top 10 most active hours overall & per listener

Dashboard:
https://app.powerbi.com/groups/me/reports/02389710-c320-490c-a63a-f2f4d7b99d63/aaec84d08186c3f7547d?experience=power-bi

Principal KPIs


Tracks Listening Dashboard

YEAR	
☐	2021
■	2022
☐	2023

WEEK	
☐	25
■	26
☐	27

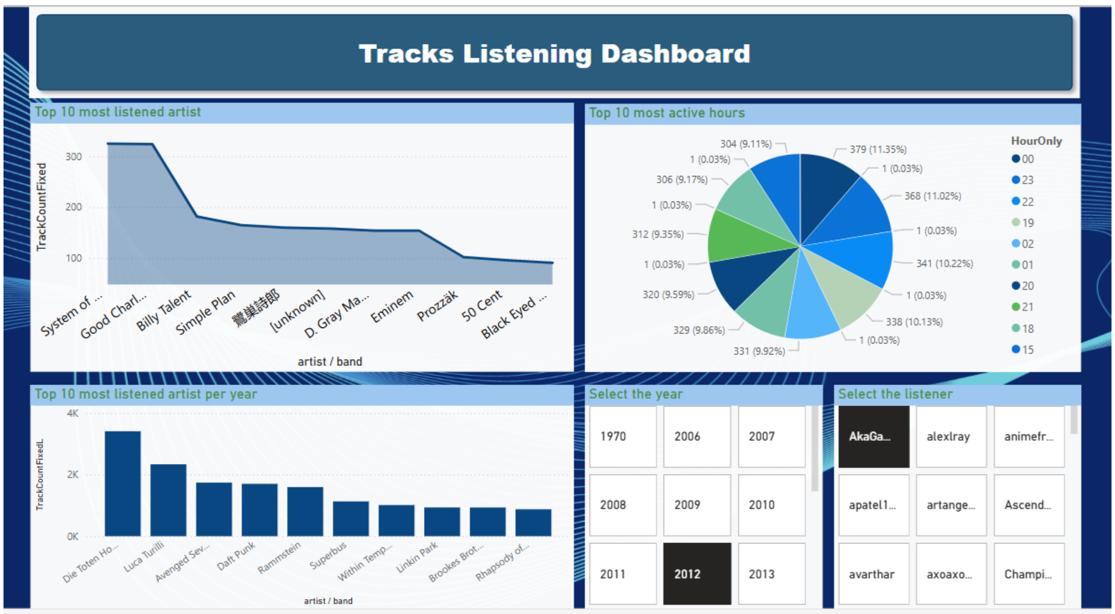
Stay Alive (P...	Left and Right ...	Stay Alive (P...
Most listened track of all time	First most listened track of the week	Most listened album of all time
4469	Honestly Neve...	4717
Total plays of the track	Most listened album of the week	Total plays of the album

Listenings per artist

listener	1 Savage	23	24kGoldn	2fox	2TH	3 Doors Down
animefreekben						
artangelo				1		
axoaxoaxo	5					
czarrep13						
d0ras						
Guigt77					6	
Hanhvunt2002						
inmyprime						
ivanshello						
JBloom91						
jon1wt						
Kleber_Fp17						
L0Lock						
Total	**6**	**1**	**1**	**1**	**6**	**1**

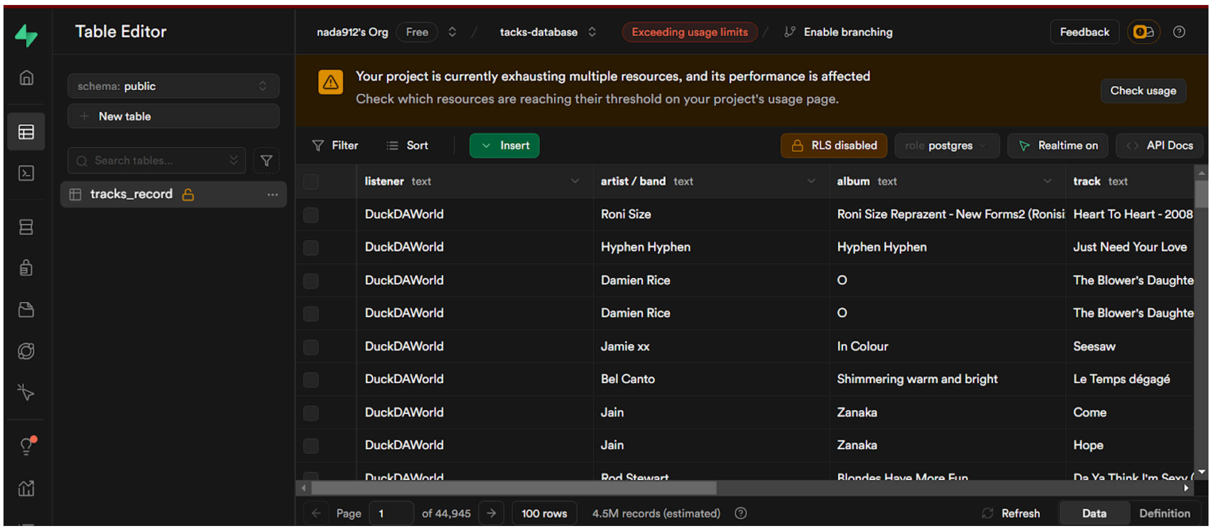
Most active listeners

Additional KPIs



Troubles:

We opted for a direct connection to the database stored in supabase but in the free version we had some problems with the available resources so that we could keep our data updated.



Your project is currently exhausting multiple resources, and its performance is affected
Check which resources are reaching their threshold on your project's usage page.

listener text	artist / band text	album text	track text
DuckDAWorld	Roni Size	Roni Size Reprazent - New Forms2 (Ronisi	Heart To Heart - 2008
DuckDAWorld	Hyphen Hyphen	Hyphen Hyphen	Just Need Your Love
DuckDAWorld	Damien Rice	O	The Blower's Daughte
DuckDAWorld	Damien Rice	O	The Blower's Daughte
DuckDAWorld	Jamie xx	In Colour	Seesaw
DuckDAWorld	Bel Canto	Shimmering warm and bright	Le Temps dégagé
DuckDAWorld	Jain	Zanaka	Come
DuckDAWorld	Jain	Zanaka	Hope
DuckDAWorld	Rod Stewart	Blondes Have More Fun	Da Ya Think I'm Sexy (

Page 1 of 44,945 100 rows 4.5M records (estimated)