Data Engineering Test Assignment

#0 Introduction

Welcome to the technical assignment for the Data Engineering position at Scalable Capital!

- If something is not clear or can be done in multiple ways, just pick one and describe why you chose your approach.
- If something is not feasible for any reason, skip it or replace it with a more straightforward question

Dataset

You will be working with a real-world dataset provided freely by the open source project ListenBrainz:

> The ListenBrainz project serves as an archive where users can store their music listening history. This dataset can be used to create new music recommendation engines. The provided data dumps contains over a 100 million listens in the ListenBrainz database.

As the original dataset is quite large, we will provide you with a subset of this data.

Setup

Please make sure that your code can easily be executed on any operating system. Include instructions on how to execute your code on MacOS.

Task #1 Data Ingestion

You are provided with an export of all listens that happened on the Spotify platform. Each line of the file contains a json document with data about one listen (the song that was listened to, the user who listened to the song, the time of the listen, etc). Please download the dataset in the following Google Drive folder: Test Assignment dataset

Your job is to set up a clean python project that loads this file into a database for easier analysis.

- Set up the database. Create one or more tables, think about how you structure and optimize the database to simplify later analysis.

- Write an ETL function that reads the export-file and writes the data into your database. The function should be idempotent, so try to write it in a way that it can deal with already ingested, duplicate or corrupted data.
- We recommend you to build your solution as a Python project, with a clean setup and instructions to run/deploy your solutions, following the best practices you use to develop.

Use a `duckdb` database for this assignment. Duckdb can be installed for python via pip: `pip install duckdb`. The following code demonstrates how to create a database with duckdb in python, but feel free to use other database drivers if you are more familiar with them.

In [1]:
```
import duckdb
# Connects to an in-file database in the current working directory, or creat
con = duckdb.connect("test.db")

# create a table and load data into it
con.sql('CREATE TABLE IF NOT EXISTS test(i INTEGER)')
con.sql('INSERT INTO test VALUES (42)')
# query the table
con.table('test').show()
```

```
┌─────────┐
│    i    │
│  int32  │
├─────────┤
│      42 │
└─────────┘
```

In [2]:
```
# Copy to a parquet file
duckdb.sql("COPY (SELECT 42) TO 'out.parquet'")
```

In [3]:
```
# Read the parquet file

duckdb.query("SELECT * FROM 'out.parquet'")
```

Out[3]:
```
┌─────────┐
│   42    │
│  int32  │
├─────────┤
│      42 │
└─────────┘
```

Task #2 Data Analysis

In the following, we ask you to run some SQL queries on the database you built in Task #1. The goal is to get more information out of the provided data.

a) To get started, answer the following questions:

- Who are the top 10 users with respect to number of songs listened to?
- How many users did listen to some song on the 1st of March 2019?
- For every user, what was the first song the user listened to?

b) Next, let's do a deep dive into user behaviour next. For each user, we want to know the top 3 days on which they had the most listens, and how many listens they had on each of these days. The result should include the following:

- 3 rows per user
- 3 columns: (user, number_of_listens, date)
- Please sort the result by the `user` and the `number_of_listens` column

c) Finally, we want to understand the development of active users within our userbase. For this, please write a query that calculates, on a daily basis, the absolute number of active users, and the percentage of active users among all users. We define a user to be active one some day X, if the user listened to at least one song in the time interval `[X–6 days, X]`. The result should adhere to the following schema:

- 1 row per day
- 3 columns: (date, number_active_users, percentage_active_users)
- Please sort the result by `date`.

Answer the above questions by running SQL queries against the database that you setup in task #1.

Hand-in

Please provide a link to some private online storage solution (e.g., Google Drive folder) where we can download your solution, as email attachments are filtered out by our system. Thank you!