UNIVERSITY OF TOURS

MASTER OF BIG DATA MANAGEMENT AND ANALYTICS

Decisional Project final report
Data warehousing system for music analysis

Authors:
Akaich Ines
Nascimento Filho, Jessé .F
de Saint Ceran, Louis
Cissé, Ismaila

Tutors :
Verónika Peralta

December 11, 2018

Contents

List of Tables

1 Introduction

As part of our master degree ‘s first year program , we worked on a business intelligence project that aims to verify the learning outcome of several courses of master 1 st year, specifically Data Warehousing and Project Management and Professional Communication.
Two main topics were proposed to students.Each topic describes a set of user needs and an available set of data source. Topics are music analysis and train tickets tracking.
Our group of four chosen to work on the first topic. We designed a data warehouse and implemented a tool that is able to extract data from different sources, transform it and then integrate it into our designed data warehouse system. Our objective is to analyze the current music streaming industry and more precisely the songs and the artists’ popularity. For this purpose, OLAP analysis of warehouse data and reporting tools were used.
Finally,We decided to bring the subject closer to us by focusing on the french market.

The rest of this document is structured as follow:
We start in section 2 by presenting our management methodology . In section 3 , we describe our user requirements. In section 4, we introduce our proposed data warehouse design . In section 5 , we present our data warehouse architecture and describe our ETL process. In section 6 , we present some of our analysis and conclusions.further analysis will be presented in the first semester defense . Finally , we discuss future works and we end with a conclusion section.

2 Management Methodology

Our team is a small multicultural and autonomous group, for this reason we found a common equation to process with our project life cycle ’s tasks . The result from this equation become our management methodology that is based in a lean solid and agile solution called Scrum. For this purpose we decided to follow the best practices of Scrum combined with a Kanban approach.
Other documents are used for managing our project, built with the help of our project management ’s tutor and that are attached with this report.

2.1 Scrum

The Scrum arise as a process framework to manage complex projects since the early 1990s. The essence of Scrum is to provide effective iterations and incremental knowledge transfer to the success of a project [8].

Our Project will be composed of 4 sprints during this semester:

- Sprint 0 — Study : The list of requirements and project planning;
- Sprint 1 — Model : Conception & modeling of data warehouse;
- Sprint 2 — ETL : Data Extraction, Transformation and Loading;
- Sprint 3 — DEMO : B.I system demonstration of an initial version;
- Sprint 4 — DEFENSE : Oral project presentation;

2.1.1 Tasks

Before a project begins,it is necessary to build all possible management plans to allow a safety progress. During the beginning of this part we had isolated ad-hoc personal tasks without any integration or communication between the others. To mitigate it, we chose to use the Trello board [See Online [1] or the appendix A] to solve communication issues. Despite of it, our team did not show progress, neither necessary synergy to adapt to a lean and agile methodology such as Kanban on Trello tools, with it we had developed an implicit work-breakdown structure (WBS) [see table 1], what clearly did not work .

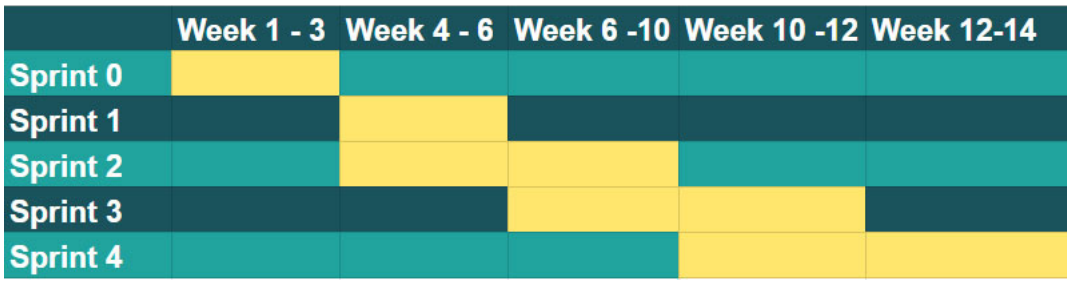
As a task of project management class, we should build a WBS, that is a deliverable-oriented breakdown of a project into smaller components, as the same one the we bent on Trello, but on this time more explicit as mandatory as described in the PMBOK Guide line.

Thereby, we arrived to list of 35 macro tasks with a virtual total duration of 188 hours, that is divided in equals pieces should be of 47 hours per member. However we can see on table 1 it was changing to reach the goals of project.

WBS	Task Title	Task Owner	Start Date	Due Date	Duration	Progress Task
1	Project Conception and Initiation					
1.1	Share a clear vision of the project	Jesse	10/09/2018	14/09/2018	4	100%
1.1.1	Identify users need	Ines	17/09/2018	19/09/2018	2	100%
1.2	Identify the preliminary workload	Ines	19/09/2018	21/09/2018	2	100%
1.3	Prepare the project management plan	Jesse	17/09/2018	21/09/2018	4	100%
1.4	Explore datasources	Ines	17/09/2018	21/09/2018	4	100%
1.5	Researchs and tools	Jesse	17/09/2018	25/09/2018	8	100%
1.6	Write the phase context into the report and submit it	Ines	20/09/2018	25/09/2018	5	100%
1.7	Feedback review and corrections	Ines and Jesse	25/09/2018	28/09/2018	3	100%
2	Project Definition	Planning and Artifact Concepts				
2.1	Create the Conceptual model	Ines	28/09/2018	05/10/2018	7	100%
2.2	Write the phase context into the report and submit it	Jesse	28/09/2018	15/10/2018	14	100%
2.3	Create the Additivity Matrix	Ines	08/10/2018	12/10/2018	4	100%
2.4	Create the Dictionary of datas	Jesse	08/10/2018	12/10/2018	4	100%
2.5	Create the chart sketch	Louis	08/10/2018	12/10/2018	4	100%
2.6	Dimensional Language	Louis	08/10/2018	12/10/2018	4	100%
2.7	PM reporting	Jesse	19/10/2018	21/10/2018	2	100%
2.7.0	Define Scope	Louis	19/10/2018	21/10/2018	2	100%
2.7.1	Time estimation	Ismaila	19/10/2018	21/10/2018	2	100%
2.7.2	Risk Management	Ines	19/10/2018	21/10/2018	2	100%
2.7.3	Reference Planning	Jesse	19/10/2018	21/10/2018	2	100%
2.7.4	Control Scope	Louis	19/10/2018	21/10/2018	2	100%
2.9	Feedback review and corrections	Ines and Jesse	23/10/2018	05/11/2018	12	100%
3	Project Launch	Execution				
3.1	Implemention of model	Jesse	05/11/2018	09/11/2018	4	100%
3.2	Pentaho BI-SERVER Settings	Jesse	05/11/2018	09/11/2018	4	100%
3.3	Saiku Settings	Jesse	05/11/2018	09/11/2018	4	100%
3.4	Power BI Settings	Ines	05/11/2018	09/11/2018	4	100%
3.5	Database Settings and Management	Jesse	05/11/2018	09/11/2018	4	100%
3.6.1	ETL Datamart popularity Song	Ines	05/11/2018	16/11/2018	20	100%
3.6.2	ETL Datamart Songs and poppularity Artist	Jesse	05/11/2018	16/11/2018	25	100%
3.7	Extraction music data it	Ines	12/11/2018	18/11/2018	6	100%
3.8	Write the phase context into the report and submit it	Ines and Jesse	12/11/2018	18/11/2018	6	100%
4	Project Test and Defense					
	Workload Queries test	Ines	19/11/2018	30/11/2018	11	
4.1	Front-end with the final analyses	Jesse	19/11/2018	30/11/2018	11	0%
4.2	Write the phase context into the report and submit it	Ines	19/11/2018	02/12/2018	13	0%
4.3	Slide presentation	Ines	03/12/2018	10/12/2018	7	0%
4.4	Project apresentation	Ines	10/12/2018	10/12/2018	0	0%

Table 1: Work-breakdown Structure (WBS)

2.1.2 Schedule



	Week 1 - 3	Week 4 - 6	Week 6 -10	Week 10 -12	Week 12-14
Sprint 0	🟨				
Sprint 1		🟨			
Sprint 2		🟨	🟨		
Sprint 3			🟨	🟨	
Sprint 4				🟨	🟨

Figure 2.1.2-1: Sprint table on a Gantt Chart.

2.1.3 Sprint 0

The name SPRINT 0 has been learned to describe the preparation phase which precedes the launching of the project. The term SPRINT 0 is being simpler to use than the preparation or inception phase, it is increasingly used in SCRUM projects. Sprint 0 does not diminish the flexibility of our project. On the contrary, it will allow us to anticipate certain actions and have an overview that will facilitate the management of changes that will emerge at the following sprints. [6]

In this Sprint we will be able to :

1. Share a clear vision of the project;
2. Identify users need;
3. Identify the preliminary workload resulting of the users need;
4. Prepare the project management plan;

2.1.4 Sprint 1

The preliminary specification of the workload in sprint 0 will help us in this sprint in modeling our data warehouse,thus the formalization of the entire workload. In addition, in this phase it is essential to maintain an active technological watch to choose our Essential BI tools used in next sprints.

2.1.5 Sprint 2

In this sprint we will be able to define our data warehouse 's architecture, assess the data quality and implement the designed ETL system.

2.1.6 Sprint 3

In this sprint we will be able to visualize our data using the BI restitution tools .

2.1.7 Sprint 4

In this final sprint , We will be able to prepare an oral presentation where we summarize all the steps that we have gone through when developing our project 's data warehouse and present our work to our professors.

2.2 Kanban

In addition of Scrum methodology we choose to use Kanban approach, that means "visual card" in Japanese, to help us simplify the sprints workload. We going to make a visual work-flow using Trello for create and manage all cards with micro tasks, it will result in each sprints deliveries milestones.

2.2.1 Trello

Trello[5] is a project management software that utilizes the concept of boards to represent projects and within boards, cards to represent tasks. Trello supports Team Collaboration enabling members to discuss a project in real-time. It keeps everybody informed through task assignments, activity log, and e-mail notifications.[2]

3 Preliminary Workload

3.1 Data Sources

The main services of stream nowadays are responsible for creating an efficient way to capture data and generate real information about the customers with it, but to obtain the success in this new world where data is like gold, they need to provide a service that is capable to get users loyalty. For this reason companies and communities of independent artists are building continuously distinct forms of technologies to present not only songs, but music with value add on it, such as, variety, quality and shareability. As a result of this frequent process innovation, today It's possible for any person to have access to huge an open-sources databases about music subjects. Hence,to make decisional projects become more simple with the follow assets MusicBrainz encyclopedia[3] and Spotify Web API. We choose to use into our project these datasets and, if necessary, others data sources could be attached as a asset during this project.

MusicBrainz Database[3] includes information about artists, release groups, releases, recordings, works, and labels, as well as the many relationships between them [table 2]. It is a community-maintained open source encyclopedia of music information. This means that anyone can help contribute to the project by adding information about your favorite artists and their related works.The entire dataset is 1.8 GB.

Attributes	type	granularity
released songs	text	by title
location	text	by region, country
genre	text	by binary
date	timestamp	by month,year
artists	text	by name, alias
origins	text	by country
cover	text	by name

Table 2: A brief description of MusicBrainz Database 's attributes.

Spotify Web API [4] Based on simple REST principles, the Web API endpoints return metadata about music artists, albums, and tracks[table 11], directly from the Spotify Data Catalogue.

Attributes	type	granularity
popularity	number	0-100
energy	float	0.0 to 1.0
valence	float	0.0 to 1.0
genre	text	by binary
danceability	float	0.0 to 1.0
duration	number	by by milliseconds
loudness	float	by decibels (dB)
mode	int	by major, minor
tempo	float	by beats per minute (BPM)

Table 3: A brief description of Spotify Web API 's attributes.

3.2 User Needs

Statistics about songs and artists:

1. Number and Average of released songs disaggregating by genre, year , location and artist .
2. The biggest/ lowest number of released songs by location , genre , year and artist .
3. Number of artists disaggregating by origins.
4. Number of artists appeared every year in the music industry.
5. Number of songs or artists that achieved a certain popularity.
6. The average popularity of the songs where artists participated to analyze artist's performance.
7. What is the less popular songs by country and find out why ?
8. What is the impact of cover art on success of an album? Number of recorded covers disaggregating by artist and song .
9. The most covered songs by artist and song .
10. Artists that are most engaged in the last years.
11. What makes a top performer based on songs ‘s technical features?

Statistics about song musical features:

- Average duration, average tempo by artist and/or location and/or year.

- What makes a tube based on culture, market, political time, features of the song or the category of the song.

4 Data Warehouse Modelling

4.1 Conceptual Design

Conceptual modeling is widely recognized to be the necessary foundation for building a database that is well-documented and fully satisfies the user requirements. In particular, from the designer point of view the availability of a conceptual model provides a higher level of abstraction in describing the warehousing process and its architecture in all its aspects.

For modelling our data warehouse schema we used The Dimensional Fact Model or DFM which is a graphical conceptual model, specifically devised for multidimensional design. In subsection 4.2.1 , we present our conceptual models corresponding to our proposed fact schemas .

4.1.1 Conceptual Schemas

In order to represent our DFM models , we used the requirement-based design approach. It is a bottom-up technique that allowed us to do the mapping between our analyzed requirements in subsection 3.2 onto our available data source so that we get to locate the conceptual objects at sources and make sure that the data verifying our user needs effectively exists.

Using this approach , We first start by defining the set of facts , measures and finally dimensions. Figure 1 shows the fact schema for the analysis of songs and artists musical features in which the fact consists of a released song with all its features in the month ,city of release ,the song 's genre and the group artist that released the song .

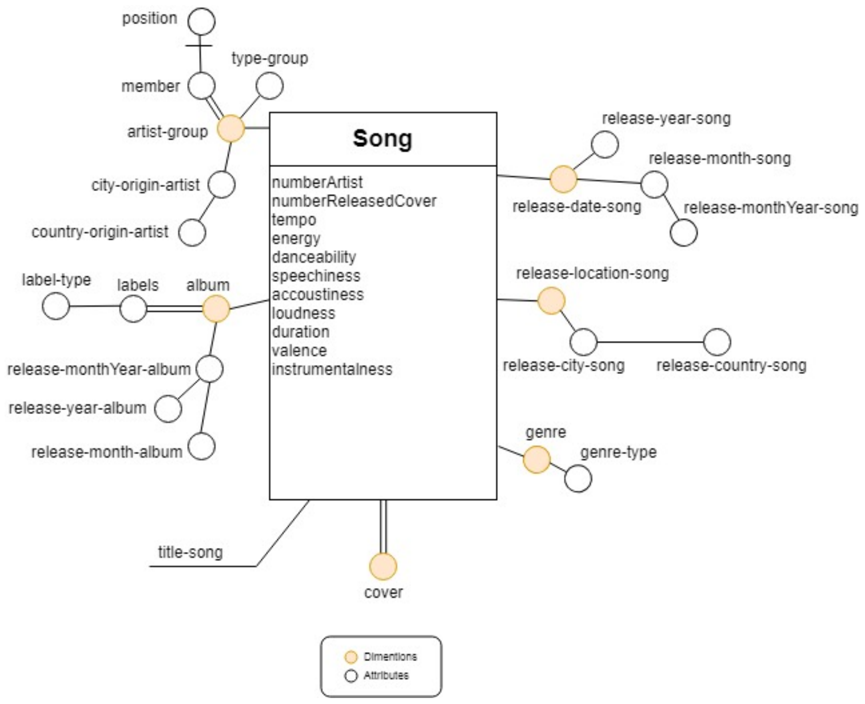


Figure 4.1.1-2: The Dimension Fact model for song.

Figure 2 and 3 shows the fact schema for the analysis of songs and artists popularity in which the fact consists of song's popularity [see figure 2] and the artist 's popularity [see figure3] tracked every month .

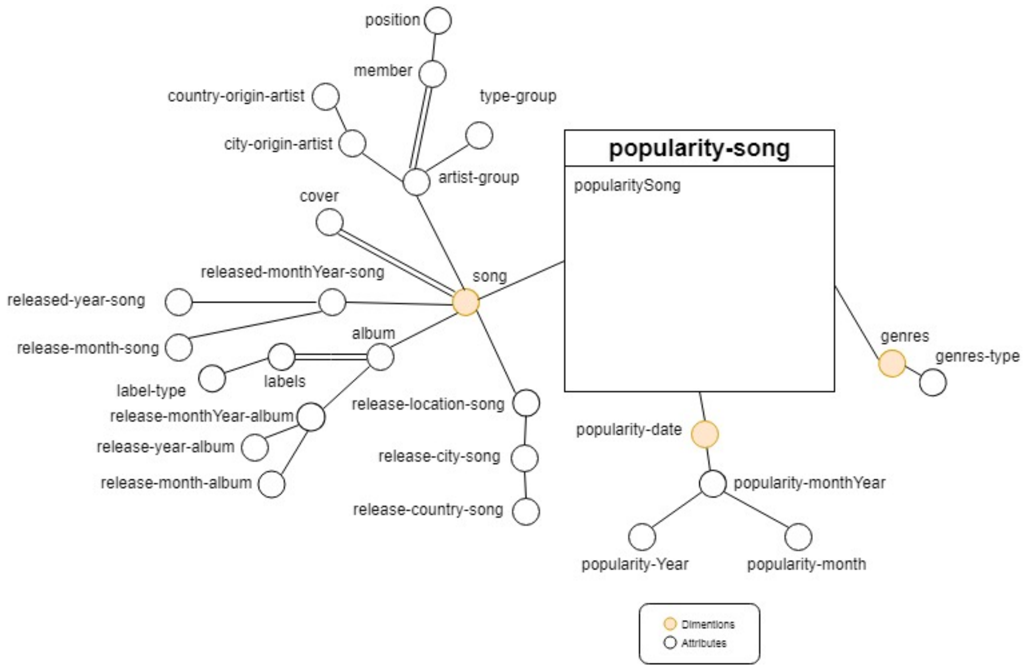


Figure 4.1.1-3: The Dimension Fact model for song popularity.

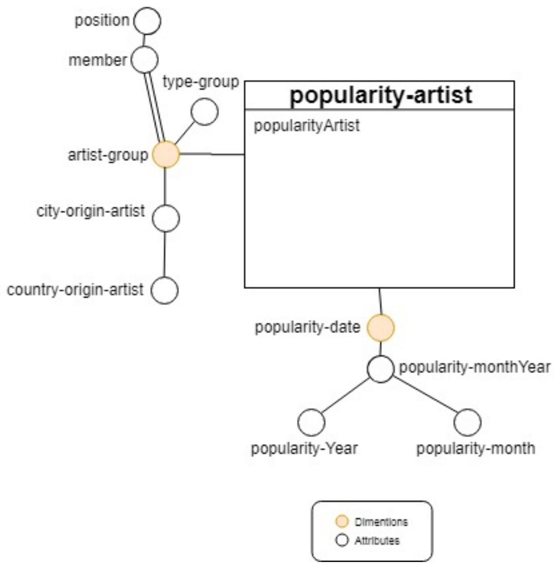


Figure 4.1.1-4: The Dimension Fact model for artist popularity.

4.1.2 Additivity Matrix

In the additivity matrix we identify all the aggregation functions applied on measures and grouped by dimensions .

Table 3 shows the equivalent additivity matrix for the fact schema in figure 1.

	Group-artist	Release-year-song	Release-city-song	genre
numberArtist		Avg	Sum Avg	Sum Avg
numberReleasedCover	Max Avg	Avg	Avg	
numberSong	Max Min	Avg Max Min	Avg Max Min	Avg Max Min
tempo	Avg	Avg	Avg	Avg
energy	Avg	Avg	Avg	Avg
danceability	Avg	Avg	Avg	Avg
speechiness	Avg	Avg	Avg	Avg
accoustiness	Avg	Avg	Avg	Avg
loudness	Avg	Avg	Avg	Avg
duration	Avg	Avg	Avg	Avg
valence	Avg	Avg	Avg	Avg
instrumentalness	Avg	Avg	Avg	Avg

Table 4: Additivity matrix for Fact song.

Table 4 shows the equivalent additivity matrix for the fact schema in figure 2.

	Song	Popularity-month	Group-artist	Popularity-year
popularitySong	Min Max Avg	Min Max Avg	Min Max Avg	Min Max Avg

Table 5: Additivity matrix for Fact Popularity-Song .

Table 5 shows the equivalent additivity matrix for the fact schema in figure 3.

	Group-artist	Popularity-month	popularity-Year
popularityArtist	Min Max Avg	Min Max Avg	Min Max Avg

Table 6: Additivity matrix for Fact Popularity-Artist .

4.1.3 Data Dictionary

An important part of any software project is to be able to provide information in a clear and accessible way. For this purpose, to have, previously, a list with all of data attributes names and description allow an efficient approach to have access to the definition of each metadata. It makes the action of manage different terminologies, formats and contents less painful for the team and users.

For this reason we created a dictionary that contain all of our attributes names, the respective description and if it is a measure (M), dimension (D) or attribute (A). Some descriptions were imported from origin sources, however we made some changes to make more clear each attribute's denotation.

Name	Description	M	D	A
artist-group	it could be a solo singer (e.g. "Drake"), group (e.g."AC DC") or lineup (e.g. "Zedd, Maren Morris and Grey") that the release is primarily credited (string).	no	yes	no
accoustiness	it is a float confidence measure from 0.0 to 1.0 of whether the track is acoustic, e.g. 1.0 represents high confidence the track is acoustic.	yes	no	no
album	is a string with the album name e.g. "Samedi soir sur la Terre"	no	yes	no
city-origin	origin city of an artist or a band e.g. "Liverpool"(string).	no	no	yes

Table 7: Data dictionary for dimensions, attributes and measures.

Name	Description	M	D	A
country-origin	origin country of an artist or a band e.g. "United Kingdom" (string).	no	no	yes
cover	is artist, lineup or band that make a new performance or recording of a song that already exists, e.g. Andrei Zevakin, Ariadne and Martti Hallik.	no	yes	yes
danceability	is a measure that describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A track closer to 0.0 is least danceable than a music near 1.0 that is most danceable, e.g. Cut To The Feeling has a 0.696 of danceability.	yes	no	no
duration	the duration of the track in milliseconds(Float).	yes	no	no
energy	is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity, e.g. Cut To The Feeling has a energy 0.905 that means a high level of activity. Combined with others measure can suggest feelings.	yes	no	no
genre	is the dimension for song genre. The genre of the song usually inherited from the album 's genre.	no	yes	no
genre-type	is the most popular type of a song genre. e.g. "Rock", "Hip-hop", "Blues", Jazz", etc.	no	no	yes
labels	the label which issued the release. There may be more than one for every released song.	no	yes	no
label-type	the label which issued the release. There may be more than one for every released song , e.g. Original Production, Reissue Production, None and Publisher (string).	no	no	yes

Table 8: Data dictionary for dimensions, attributes and measures.

Name	Description	M	D	A
instrumentalness	is a float measure that predicts whether a track contains no vocals. “Ooh” and “aah” sounds are treated as instrumental in this context. Rap or spoken word tracks are clearly “vocal”. The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content. Values above 0.5 are intended to represent instrumental tracks, but confidence is higher as the value approaches 1.0.	yes	no	no
loudness	it is a float measure of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing relative noise into a track. It is the quality of a sound that is the primary psychological correlate of physical strength (amplitude). Values typical range between -60 and 0 db.	yes	no	no
member	it could be a band member e.g. the current members of AC DC are Angus Young, Chris Slade, Stevie Young and W. Axl Rose (string).	no	yes	yes
numberArtist	is a measure to known the number of artists in a group or band, even in a solo arrangement. If artist-group is composed of one artist e.g. ”Lady Gaga”, numberArtist takes 1 as a value.	yes	no	no
numberSong	an integer measure to know how many songs a group artist have released .	yes	no	no
numberReleasedCover	a measure to know how many times a song has covered by an artist or group (Integer).	yes	no	no
popularity-month	the number month that corresponds to the value of the popularity was extracted , e.g. ”11” (string).	no	no	yes
popularity-year	the year that corresponds to the value of the popularity was extracted, e.g. ”2017” (”YYYY” string).	no	no	yes

Table 9: Data dictionary for dimensions, attributes and measures.

Name	Description	M	D	A
popularitySong	is a measure that present popularity of the track. This value will be between 0 and 100, whether closer to 100 being is the most popular. Songs that are being played a lot now will have a higher popularity than songs that were played a lot in the past. Duplicate tracks (e.g. the same track from a single and an album) are rated independently.	yes	no	no
position	is a optional attribute that can have as a string value the role(s) of a member in a band, e.g. Singer, Drummer or if does not exist a position listed "unknown".	no	no	yes
release-date-song	is a dimension with the first date of released songs.	no	yes	no
release-monthYear-song	month and year number of the first song released. e.g. "082018" ("MMYYYY" - string).	no	no	yes
release-month-song	month number of the first song released. e.g. "08" ("MM" - string).	no	no	yes
release-year-song	year number (e.g. "2018") of the first song release ("YYYY" - timestamp).	no	no	yes
release-month-album	month of the first release of the album(e.g. "April", in string)	no	no	yes
release-year-album	year of the first release of the album(e.g. "YYYY", in string).	no	no	yes
release-location-song	is a dimension with a release song locations.	no	yes	no
release-city-song	release city of a song	no	no	yes
release-country-song	release country of a song .	no	no	yes
song	it's a dimension that contains song 's features.	no	yes	no

Table 10: Data dictionary for dimensions, attributes and measures.

Name	Description	M	D	A
speechiness	is a float measure that can detect the presence of spoken words in a track.The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value. Values above 0.66 describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections or layered, including such cases as rap music. Values below 0.33 most likely represent music and other non-speech-like tracks.	yes	no	no
valence	is a float measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry).	yes	no	no
type-group	it represent the type of a group of musicians (e.g. band, orchestra) or an artist (e.g. solo) (string).	no	no	yes
tempo	is a measure of speed or pace of a given piece and derives directly from the average beat per minute (BPM) duration e.g. Billie Jean have 92 BPM (integer).	yes	no	no
title-song	a string that contains a music name or a title of a release e.g. "La corrida" .	no	no	yes

Table 11: Data dictionary for dimensions, attributes ans measures.

4.2 Formalization of Requirements

In this section ,we tried to refine our workload by identifying queries that are expressed directly on the conceptual model in a formal language in order to validate our conceptual models.

- user need 1:Number and Average of released songs disaggregating by genre, year , location and artist.
- user need 2:The biggest/ lowest number of released songs by location , genre , year and artist .
  - SONG[genre,release-year-song,release-city-song,artist-group]. numberSong
- user need 3:Number of artists disaggregating by origins.
  - SONG[city-origin-artist].numberArtist
- user need 4:. Number of artists appeared every year in the music industry.
  - SONG[release-year-song;title-song IN [title-song,release-year-song,artist-group, Min(release-year-song)].title-song].numberArtist
- user need 5:. Number of songs or artists that achieved a certain popularity.
  - POPULARITY-ARTIST[artist-group;popularity > X].numberArtist
  - POPULARITY-SONG[title-song;popularity > X].numberSong
- user need 6:The average popularity of the songs where artists participated to analyze artist's performance.
  - POPULARITY-SONG[artist-group].popularitySong
- user need 7:What is the less popular songs by country and find out why ?
  - POPULARITY-SONG[Song,release-country-song; popularitySong=[POPULARITY-SONG[release-country-song].Min(popuaritySong)]
- user need 8:. What is the impact of cover art on success of an album? Number of recorded covers disaggregating by artist and song
  - SONG[artist-group].numberReleasedCover
- user need 9:. The most covered songs by artist and song.
  - SONG[artist-group,title-song,coverOrNot=1].numberReleasedCover
- user need 10:. Artists that are most engaged in the last years.

- SONG[artist-group,title-song].Max(numberReleasedSongs)

- user need 11:What makes a top performer based on songs ‘s technical features?
  - SONG+POPULARITY-SONG[;popularity > X].tempo
  - SONG+POPULARITY-SONG[;popularity > X].energy
- user need 12:Average duration, average tempo by artist and/or location and/or year.
  - SONG[artist-group;release-year-song >= 2016 and release-year-song <=2018].Max(numberSong)
- user need 13:What makes a tube based on culture, market, political time, features of the song or the category of the song.
  - SONG[artist-group,release-city-song,release-year-song].Avg(Duration)
  - SONG[artist-group,release-city-song,release-year-song].Avg(Tempo)
- additional request 1:
  This query calculates the popularity of group artists in each month in 2016 and its members are six or less:
  - Popularity-artist[month, groupArtist; year='2016', artist-group in SONG[year, artist-group; year = '2016' and numberArtist <= 6].artist-group].popularityArtist

4.3 Prototypes of Charts

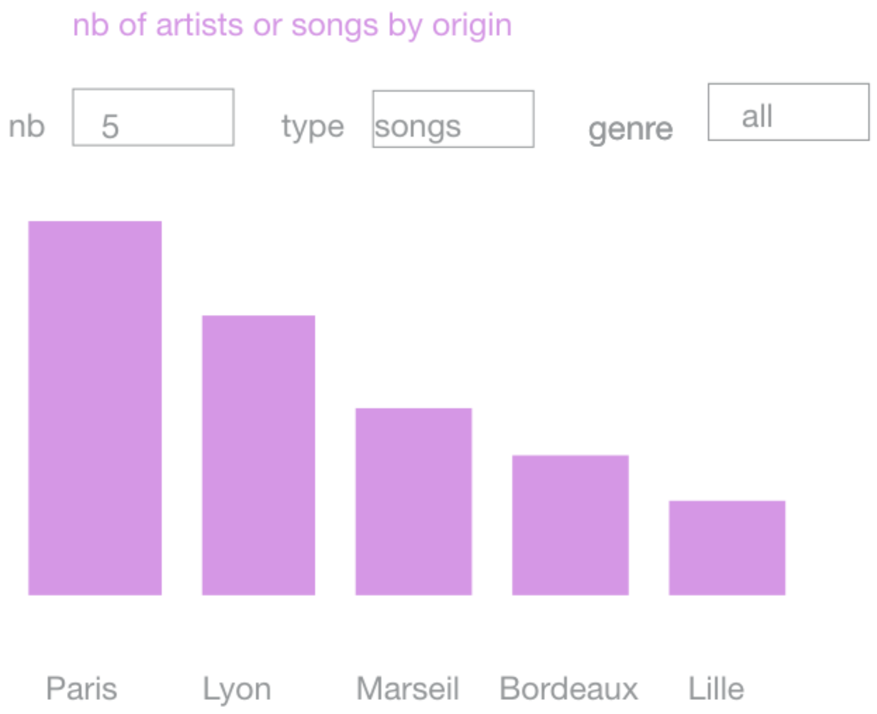


Figure 4.3.0-5: prototype of figure presenting the number of songs or artists by city

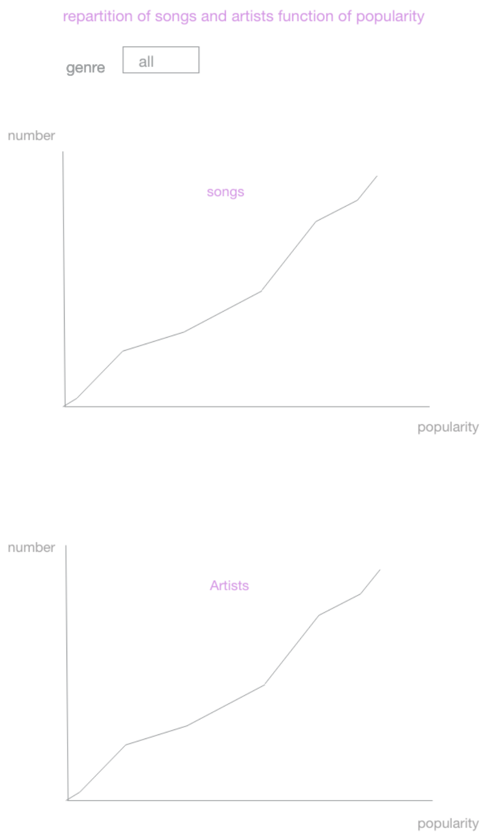
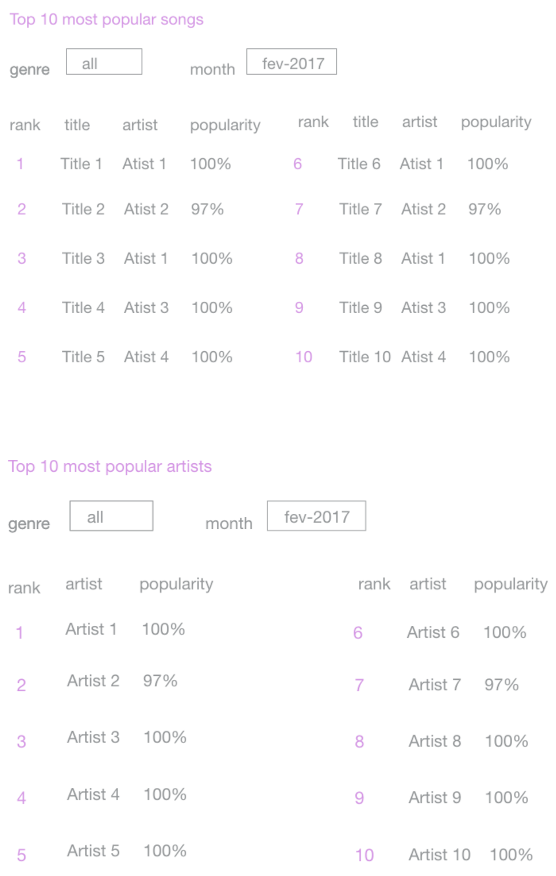


Figure 4.3.0-6: prototype of figure presenting the re-partition of artists and song function of popularity



Top 10 most popular songs

genre: all   month: fev-2017

rank	title	artist	popularity	rank	title	artist	popularity
1	Title 1	Atist 1	100%	6	Title 6	Atist 1	100%
2	Title 2	Atist 2	97%	7	Title 7	Atist 2	97%
3	Title 3	Atist 1	100%	8	Title 8	Atist 1	100%
4	Title 4	Atist 3	100%	9	Title 9	Atist 3	100%
5	Title 5	Atist 4	100%	10	Title 10	Atist 4	100%

Top 10 most popular artists

genre: all   month: fev-2017

rank	artist	popularity	rank	artist	popularity
1	Artist 1	100%	6	Artist 6	100%
2	Artist 2	97%	7	Artist 7	97%
3	Artist 3	100%	8	Artist 8	100%
4	Artist 4	100%	9	Artist 9	100%
5	Artist 5	100%	10	Artist 10	100%

Figure 4.3.0-7: ranking of song and artist by popularity

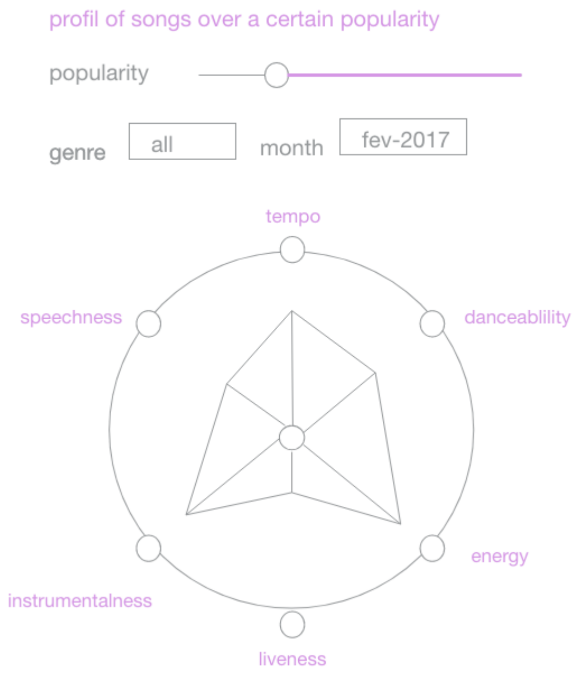


Figure 4.3.0-8: prototype of figure presenting the average technical measures for songs over a certain popularity

5 Data Warehouse Genesis

Data warehousing is a phenomenon that grew from the huge amount of data stored in recent years and from the need to analyse this data. [7].

For this purpose it is necessary to understand data sources, clean it and find the best integration methodology according to the users needs.
In this section, we present:

1. Data warehouse Architecture
2. Extraction process
3. Transformation process

5.1 Data warehouse Architecture

Our data warehouse architecture includes the following layers (see figure 5.1) :

- Data source layer
- Data Staging layer
- Data Storage layer

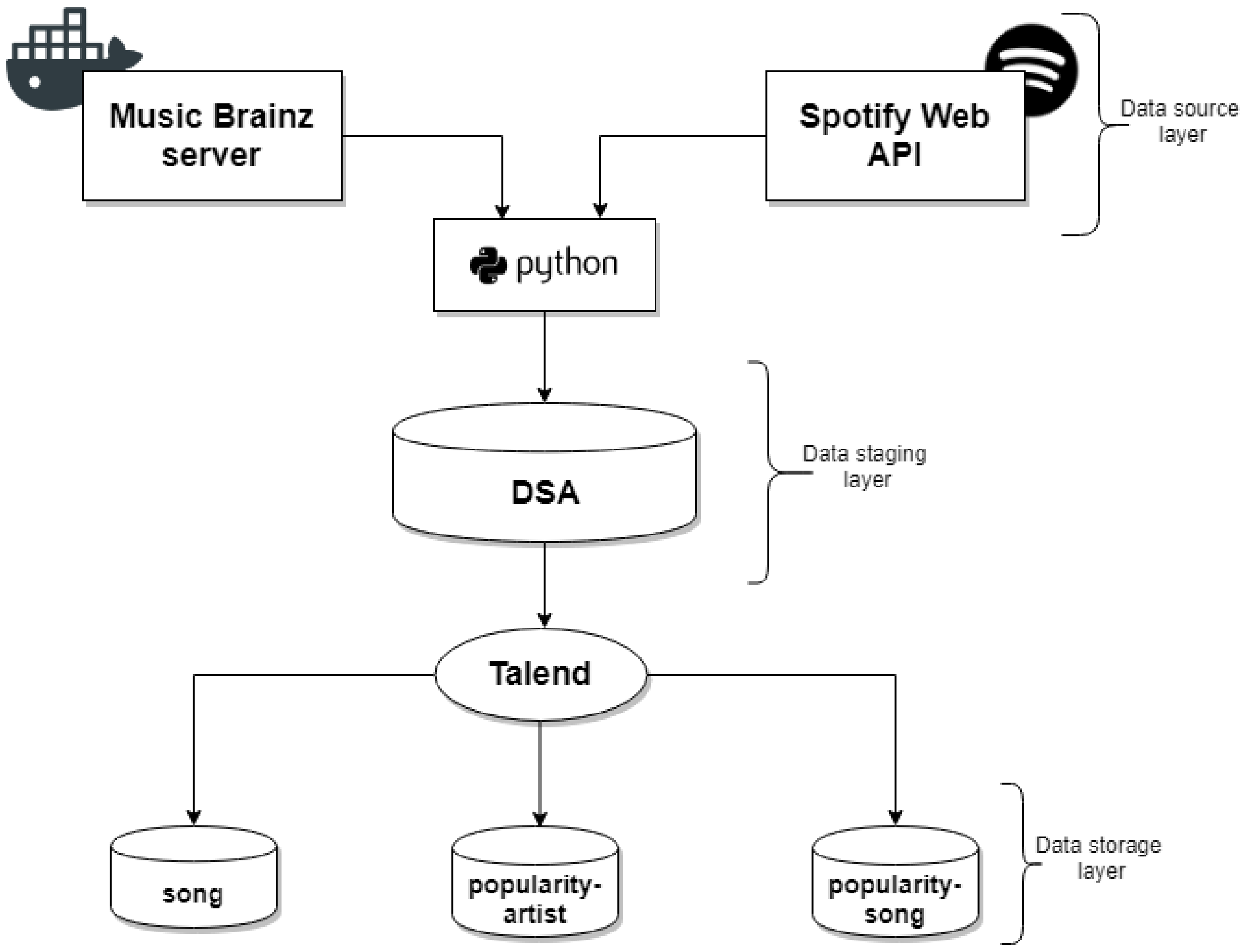


Figure 5.1.0-9: Data warehouse Architecture

5.1.1 Data source layer

The data in our data warehouse is collected from two different sources :

- The music brainz database that is deployed by us in a server running inside a docker container inside our linux virtual machine.
- The spotify web api Based on simple REST principles and that return JSON metadata about music artists, albums, and tracks, directly from the Spotify Data Catalogue.

The two sources are used to extract data related to artists, albums and songs using python scripts and a python library called Spotipy which gave us full access to all of the music data provided by the Spotify platform .

The result sets are directly stored in the staging database.

5.1.2 Data staging layer

We used the data staging area or DSA to store the extracted data in its raw format before transformations.The reason for using the dsa is that the data that comes from spotify is in json format so we need a landing stage where we can find all data from different sources in the same format.
Our DSA resides on a postgres server deployed on our virtual machine.

5.1.3 Data Storage layer

After transformations and cleansing using Talend , data is stored in a ROLAP database .
We find in this layer , three datamarts : the song datamart,the popularity-artist datamart and popularity-song datamart. Our datamarts reside on a postgres server deployed on our virtual machine.

5.2 Extraction, Transformation and Load process

5.2.1 Extraction process

For this, we used three python scripts [See on Appendix B]:
the first one extracts artists from music brainz database and search for every extracted artist in spotify returning their spotify IDs.

The second script, based on the spotify IDs of artists in spotify, we extracted their albums and IDs.
Finally, the third script extracts the tracks and its features for every extracted album.

The reason behind using this strategy is that the spotify api methods need the IDs of artists, albums and tracks to return the requested data.

5.2.2 Transformation and Load process

As a result of our extraction, now we have stages tables that can be easily used to load our data mart. To do this we used Talend studio a French data integration tools that allow multiples forms of data manipulation. However, before taking off to our transformation we need to present our logical data mart model. Through it, we can create the data warehouse that will receive all data.[See Figures 5.2.2-10, 5.2.2-11 and 5.2.2-12]

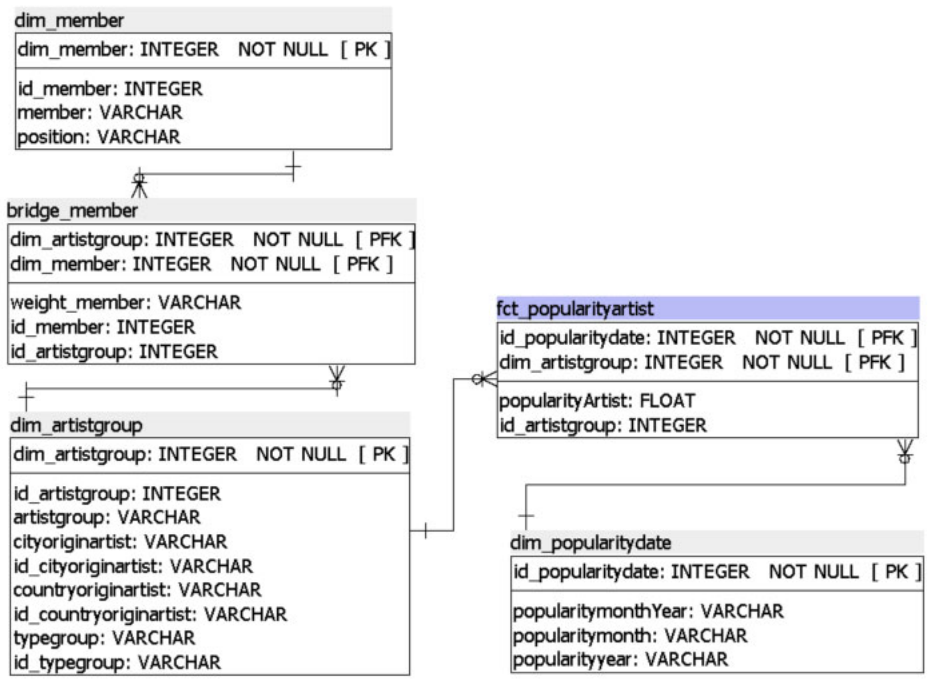

dim_member
dim_member: INTEGER NOT NULL [PK]
id_member: INTEGER
member: VARCHAR
position: VARCHAR
bridge_member
dim_artistgroup: INTEGER NOT NULL [PFK]
dim_member: INTEGER NOT NULL [PFK]
weight_member: VARCHAR
id_member: INTEGER
id_artistgroup: INTEGER
dim_artistgroup
dim_artistgroup: INTEGER NOT NULL [PK]
id_artistgroup: INTEGER
artistgroup: VARCHAR
cityoriginartist: VARCHAR
id_cityoriginartist: VARCHAR
countryoriginartist: VARCHAR
id_countryoriginartist: VARCHAR
typegroup: VARCHAR
id_typegroup: VARCHAR
fct_popularityartist
id_popularitydate: INTEGER NOT NULL [PFK]
dim_artistgroup: INTEGER NOT NULL [PFK]
popularityArtist: FLOAT
id_artistgroup: INTEGER
dim_popularitydate
id_popularitydate: INTEGER NOT NULL [PK]
popularitymonthYear: VARCHAR
popularitymonth: VARCHAR
popularityyear: VARCHAR

Figure 5.2.2-10: The logical model for artist popularity data mart.

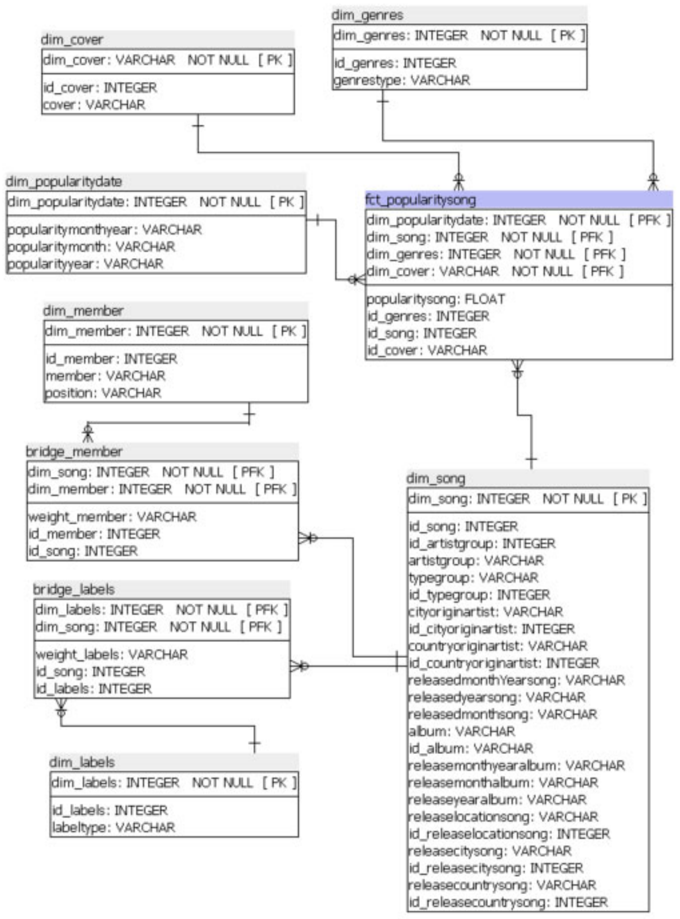

dim_cover
dim_cover: VARCHAR NOT NULL [PK]
id_cover: INTEGER
cover: VARCHAR
dim_genres
dim_genres: INTEGER NOT NULL [PK]
id_genres: INTEGER
genrestype: VARCHAR
dim_popularitydate
dim_popularitydate: INTEGER NOT NULL [PK]
popularitymonthyear: VARCHAR
popularitymonth: VARCHAR
popularityyear: VARCHAR
fct_popularitysong
dim_popularitydate: INTEGER NOT NULL [PFK]
dim_song: INTEGER NOT NULL [PFK]
dim_genres: INTEGER NOT NULL [PFK]
dim_cover: VARCHAR NOT NULL [PFK]
popularitysong: FLOAT
id_genres: INTEGER
id_song: INTEGER
id_cover: VARCHAR
dim_member
dim_member: INTEGER NOT NULL [PK]
id_member: INTEGER
member: VARCHAR
position: VARCHAR
bridge_member
dim_song: INTEGER NOT NULL [PFK]
dim_member: INTEGER NOT NULL [PFK]
weight_member: VARCHAR
id_member: INTEGER
id_song: INTEGER
dim_song
dim_song: INTEGER NOT NULL [PK]
id_song: INTEGER
id_artistgroup: INTEGER
artistgroup: VARCHAR
typegroup: VARCHAR
id_typegroup: INTEGER
cityoriginartist: VARCHAR
id_cityoriginartist: INTEGER
countryoriginartist: VARCHAR
id_countryoriginartist: INTEGER
releasedmonthYearsong: VARCHAR
releasedyearsong: VARCHAR
releasedmonthsong: VARCHAR
album: VARCHAR
id_album: VARCHAR
releasemonthyearalbum: VARCHAR
releasemonthalbum: VARCHAR
releaseyearalbum: VARCHAR
releaselocationsong: VARCHAR
id_releaselocationsong: INTEGER
releasecitysong: VARCHAR
id_releasecitysong: INTEGER
releasecountrysong: VARCHAR
id_releasecountrysong: INTEGER
bridge_labels
dim_labels: INTEGER NOT NULL [PFK]
dim_song: INTEGER NOT NULL [PFK]
weight_labels: VARCHAR
id_song: INTEGER
id_labels: INTEGER
dim_labels
dim_labels: INTEGER NOT NULL [PK]
id_labels: INTEGER
labeltype: VARCHAR

Figure 5.2.2-11: The logical model for song popularity data mart.

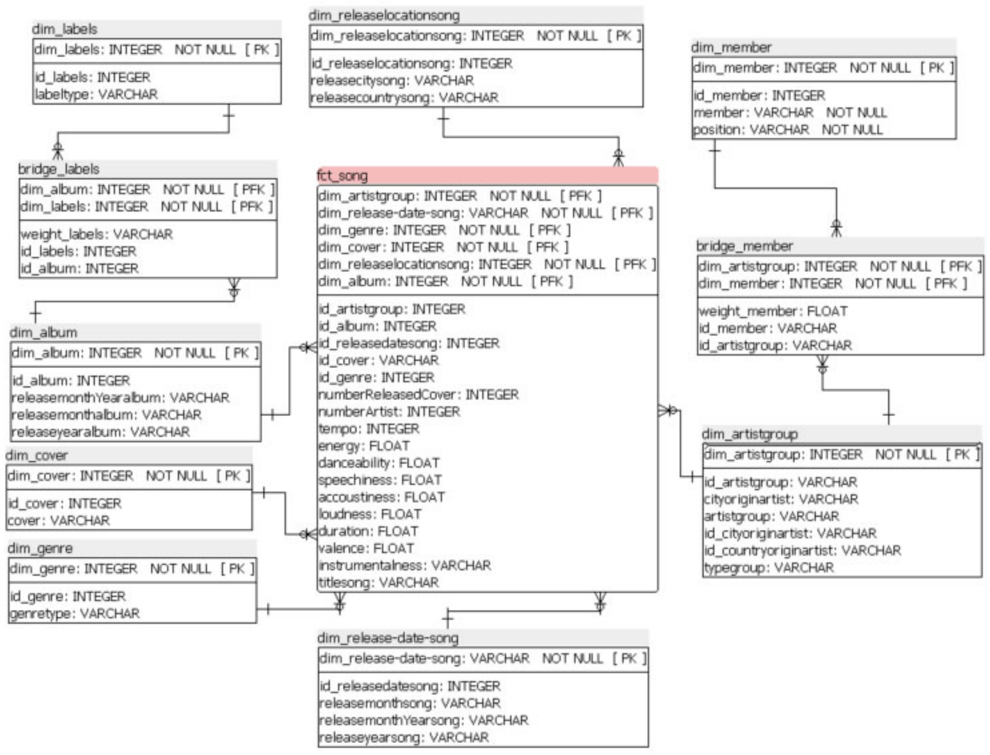

dim_labels
dim_labels: INTEGER NOT NULL [PK]
id_labels: INTEGER
labeltype: VARCHAR
dim_releaselocationsong
dim_releaselocationsong: INTEGER NOT NULL [PK]
id_releaselocationsong: INTEGER
releasecitysong: VARCHAR
releasecountrysong: VARCHAR
dim_member
dim_member: INTEGER NOT NULL [PK]
id_member: INTEGER
member: VARCHAR NOT NULL
position: VARCHAR NOT NULL
bridge_labels
dim_album: INTEGER NOT NULL [PFK]
dim_labels: INTEGER NOT NULL [PFK]
weight_labels: VARCHAR
id_labels: INTEGER
id_album: INTEGER
fct_song
dim_artistgroup: INTEGER NOT NULL [PFK]
dim_release-date-song: VARCHAR NOT NULL [PFK]
dim_genre: INTEGER NOT NULL [PFK]
dim_cover: INTEGER NOT NULL [PFK]
dim_releaselocationsong: INTEGER NOT NULL [PFK]
dim_album: INTEGER NOT NULL [PFK]
id_artistgroup: INTEGER
id_album: INTEGER
id_releasedatesong: INTEGER
id_cover: VARCHAR
id_genre: INTEGER
numberReleasedCover: INTEGER
numberArtist: INTEGER
tempo: INTEGER
energy: FLOAT
danceability: FLOAT
speechiness: FLOAT
accoustiness: FLOAT
loudness: FLOAT
duration: FLOAT
valence: FLOAT
instrumentalness: VARCHAR
titlesong: VARCHAR
bridge_member
dim_artistgroup: INTEGER NOT NULL [PFK]
dim_member: INTEGER NOT NULL [PFK]
weight_member: FLOAT
id_member: VARCHAR
id_artistgroup: VARCHAR
dim_album
dim_album: INTEGER NOT NULL [PK]
id_album: INTEGER
releasemonthYearalbum: VARCHAR
releasemonthalbum: VARCHAR
releaseyearalbum: VARCHAR
dim_cover
dim_cover: INTEGER NOT NULL [PK]
id_cover: INTEGER
cover: VARCHAR
dim_genre
dim_genre: INTEGER NOT NULL [PK]
id_genre: INTEGER
genretype: VARCHAR
dim_artistgroup
dim_artistgroup: INTEGER NOT NULL [PK]
id_artistgroup: VARCHAR
cityoriginartist: VARCHAR
artistgroup: VARCHAR
id_cityoriginartist: VARCHAR
id_countryoriginartist: VARCHAR
typegroup: VARCHAR
dim_release-date-song
dim_release-date-song: VARCHAR NOT NULL [PK]
id_releasedatesong: INTEGER
releasemonthsong: VARCHAR
releasemonthYearsong: VARCHAR
releaseyearsong: VARCHAR

Figure 5.2.2-12: The logical model for song data mart.

We have chosen the data mart approach that gave us more flexibility and impetus during the development process.the reason behind this choice is the technical viability that don't necessary require that the data marts will need exchanges relationships between them, because each data mart can answer the respective user needs independently.

In addition to more details about the logical model, we had exported from our conception model the approach of multiple arcs, however it will not be used, because our front-end selected tool, Saiku, not have the appropriate support for this structure.

Looking at Figure 5.2.2-13, we can see the applicability of our stage strategy described above. With a "db_postgres input" was possible to fetch the data that will feed "dim_popularitydate" and "dim_artistgroup" and then when this process reach the end, the second flow will be started. Consequently with a simple join between dimension and stage table, we can feed our fact "popularity artist" with all data needed for it. The same pattern used to load the other data marts.[see 5.2.2-14, 5.2.2-15 and 5.2.2-16]

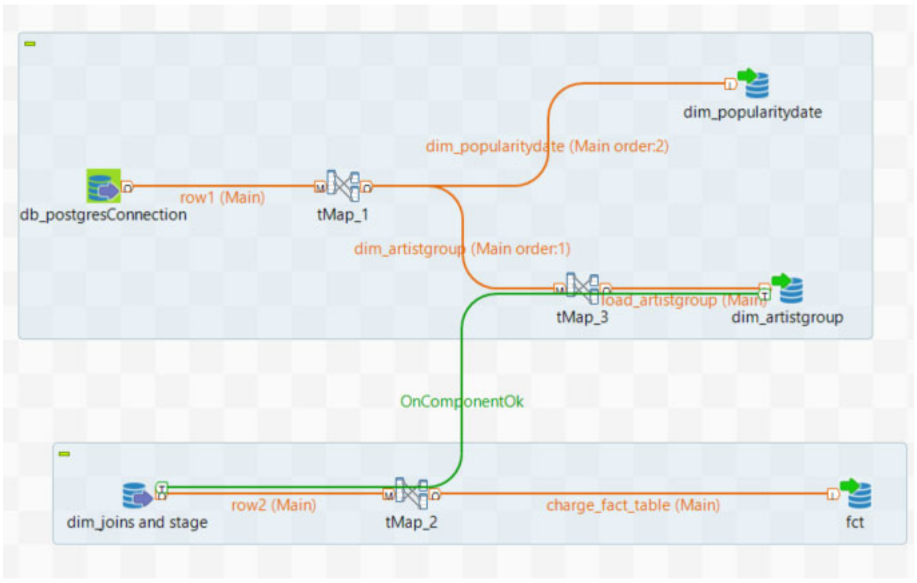


Figure 5.2.2-13: The work-flow for load Popularity Artist data mart.

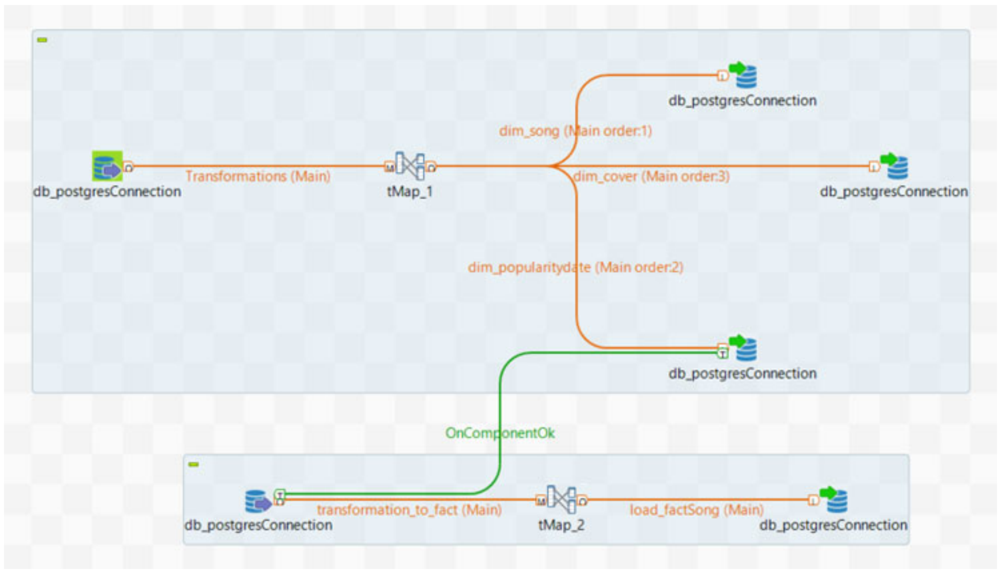


Figure 5.2.2-14: The work-flow for load Popularity Song data mart.

After a lot of practice, developping our skills and devoting our time to troubleshooting Talend , the follow work-flows [tables 5.2.2-15 and 5.2.2-16] clearly show some advanced techniques on ETL process that mixe between the experience that part of the group have with business intelligence tools and the new gained skills .

We chose Saiku to make analyses on our data sets, then in parallel during the ETL process, we studied Saiku limitations. With it, we had in mind this issue (Saiku doesnt support n-n relationships).To try solving this problem, using ”tMap”, we found a way in working on the fact table attributes to register the respective sum of covers for each song. Consequently we had access to the right quantity without the need to implement bridge table.

Additionally in ”dim_genre” [Figure 5.2.2-15] we did some unification over genre type, to find more real matches . When all dimensions tables were loaded, the process of fact song load begin. It push into a stream flow all essential attributes to charge the fact table respecting all constraints. It means that in case of a new wave of data arrival into our stage, this schema is capable of integrating those new ones without any resulting issues with in data mart.

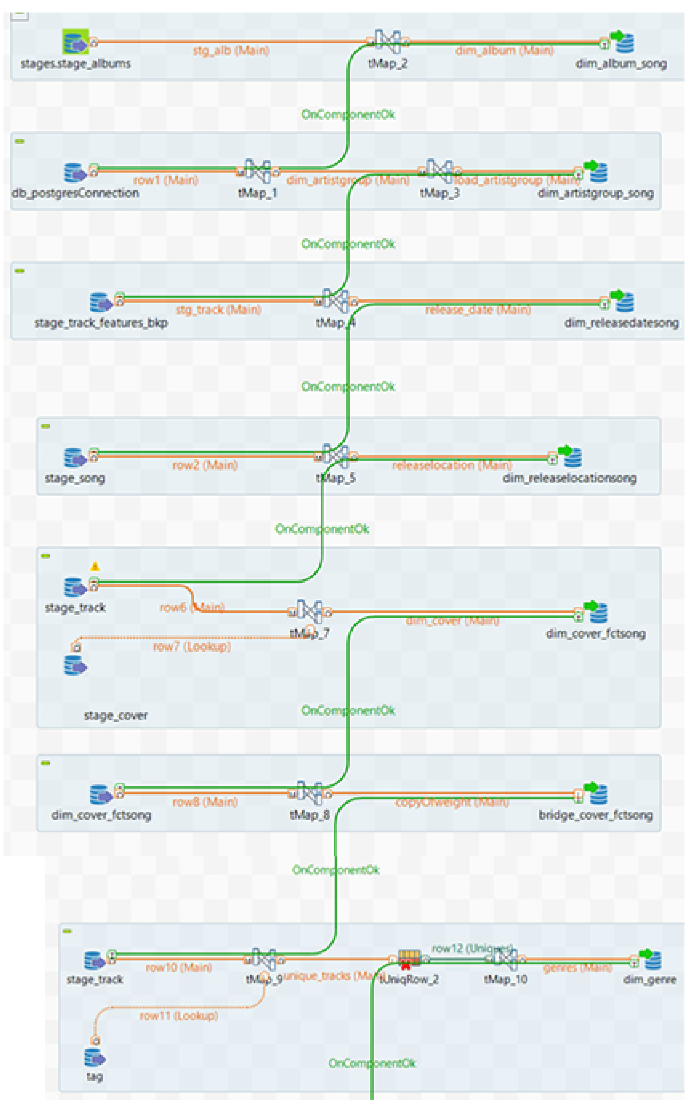


Figure 5.2.2-15: Part 1: The work-flow for load song data marts dimensions.

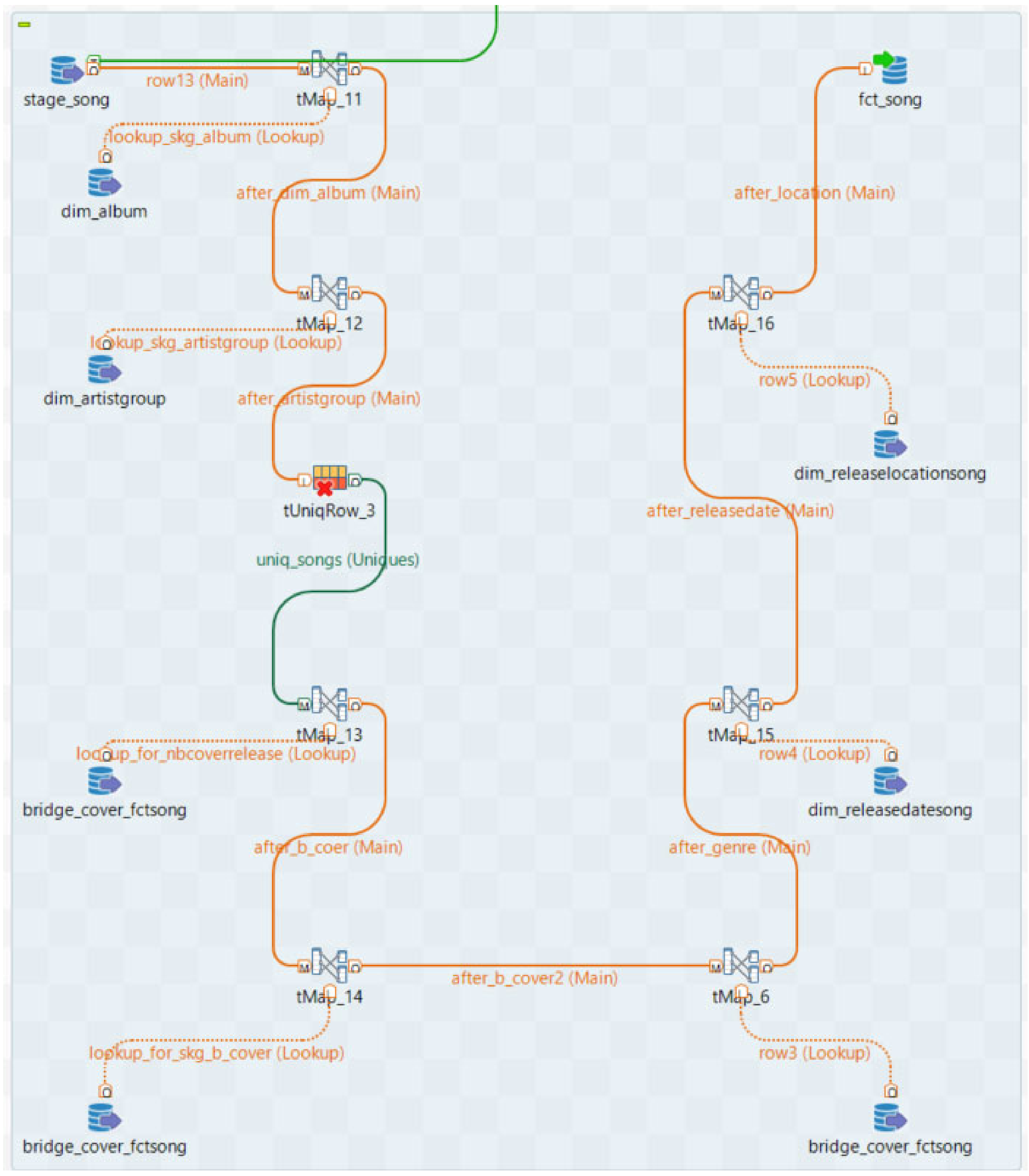


Figure 5.2.2-16: Part 2: The work-flow for load song fact table.

6 Reporting

In this section , we tried to analyze some of the user needs discussed in section 3. Further analysis will be presented in the semester defense.

6.1 Most Popular artist by Country and reasons behind that

All described analysis in this section (6.1) were performed using the OLAP tool Saiku.
As foreign students, We had the curiosity to understand what kind of artists/groups the French market is more conducive to absorb. Obviously this market has a multi-cultural and diversity richness that is available abroad. However, despite of the french market trying selling artist/groups lyrics and melodies to others markets, at home, they are not doing the work. Our analysis let appear a high foreign origins artist/group presence given by quantity and popularity .

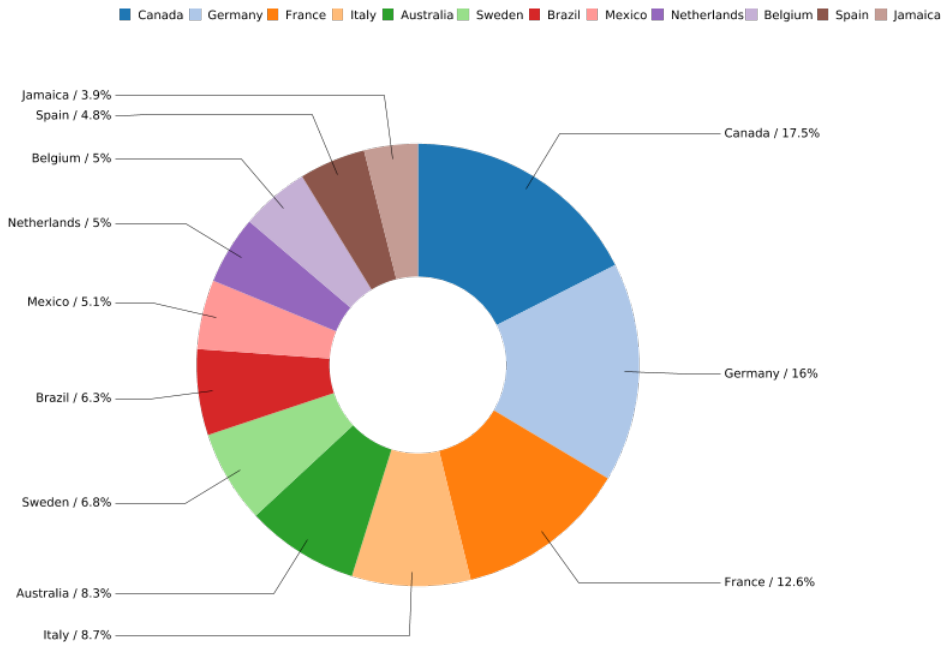
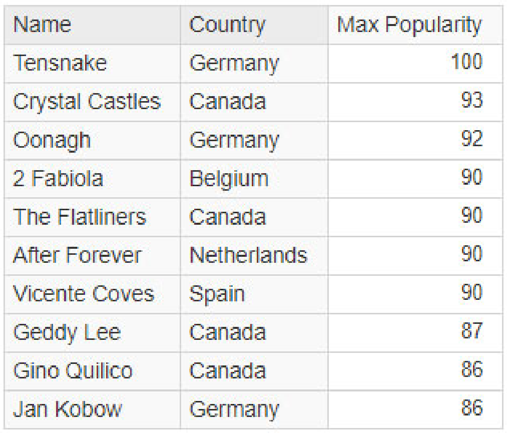


Figure 6.1.0-17: Donuts Chart: Quantity of artist/group by origin country.

This donuts chart [Figure 6.1.0-17] show a large presence of foreign artist/group on french stream music market. With just 12.6% the francophone artists leave a huge place on this market to Canadians and Germans, and others slight pieces to others origins. The main question that may pop up is if it have any correlation with french natives origins, to figure out the answer to this question we have to cross music data with french demographics data.

Comparatively of the market quantity presence, we bring also the maximum popularity by country origin of artist/group [Figure 6.1.0-18]. The result can broke hearts or not, however, maybe your most popular artist does not appear here because of two reason; First one Spotify algorithms may not regard song popularity as a high weight to popularity artist/group, just to avoid disturbances made by sudden song ads.

The Second one is that our analysis bring up only two sources, Music Brainz and Spotify, these two source have characteristic distinct between then, thus it can make our analysis bypass all marketing strategies and social network bubbles data. What is means is that it may indicate an effort that people do to escape from "popular" to find new ones or stay listing them familiar favorite songs. A supplementary reason can be the high quantity of tourist in France, it may be have a side effect on Spotify data.



Name	Country	Max Popularity
Tensnake	Germany	100
Crystal Castles	Canada	93
Oonagh	Germany	92
2 Fabiola	Belgium	90
The Flatliners	Canada	90
After Forever	Netherlands	90
Vicente Coves	Spain	90
Geddy Lee	Canada	87
Gino Quilico	Canada	86
Jan Kobow	Germany	86

Figure 6.1.0-18: Top popularity artist/group on french market by origin country.

Consequently of those reason, we arrived to the conclusion that to clarify some reasons our data warehouse should be fed with others data sources as a demographics, tourism, others musics database, social network text review, Google search data and

YouTube traffic origin. Then will be able to decide what impact on popularity artist and it is may be very useful to allow companies and artist re-target an audience by origins as a marketing strategy to sell not just songs, but cultural articles, language course or simple present ads on native language of target audience.

6.2 How to make a tube based of spotify song features analysis

We maked this analysis with tableau software.

Spotify api give "subjective" song features as :

- instrumentalness (quantity of instrument in the songs)
- valence (song happiness)
- speechiness (quantity of understandable words in the song)
- danceability
- energy
- accousticness

Analyzing this features and the tempo and duration , we tried to find vectors of success of a song.(figure 6.2.0-19)

The more the songs are popular the more they tend to be happy, danceable and with energy. the tempo also tend to grow with the popularity.

On the contrary the remaining features fall with the popularity.

After that we tried to find temporal trends over the years and months but without success (figure 6.2.0-20). the only thing that we find is that autumn and spring are the most active seasons for releases.

Finally we looked at the standard deviation of those variables function of years and popularity. We did that in order to see if the variety of music tend to smooth over the years or with popularity (figure 6.2.0-21). The results are that the variety of music is still the same over the years but that popularity smooth it.

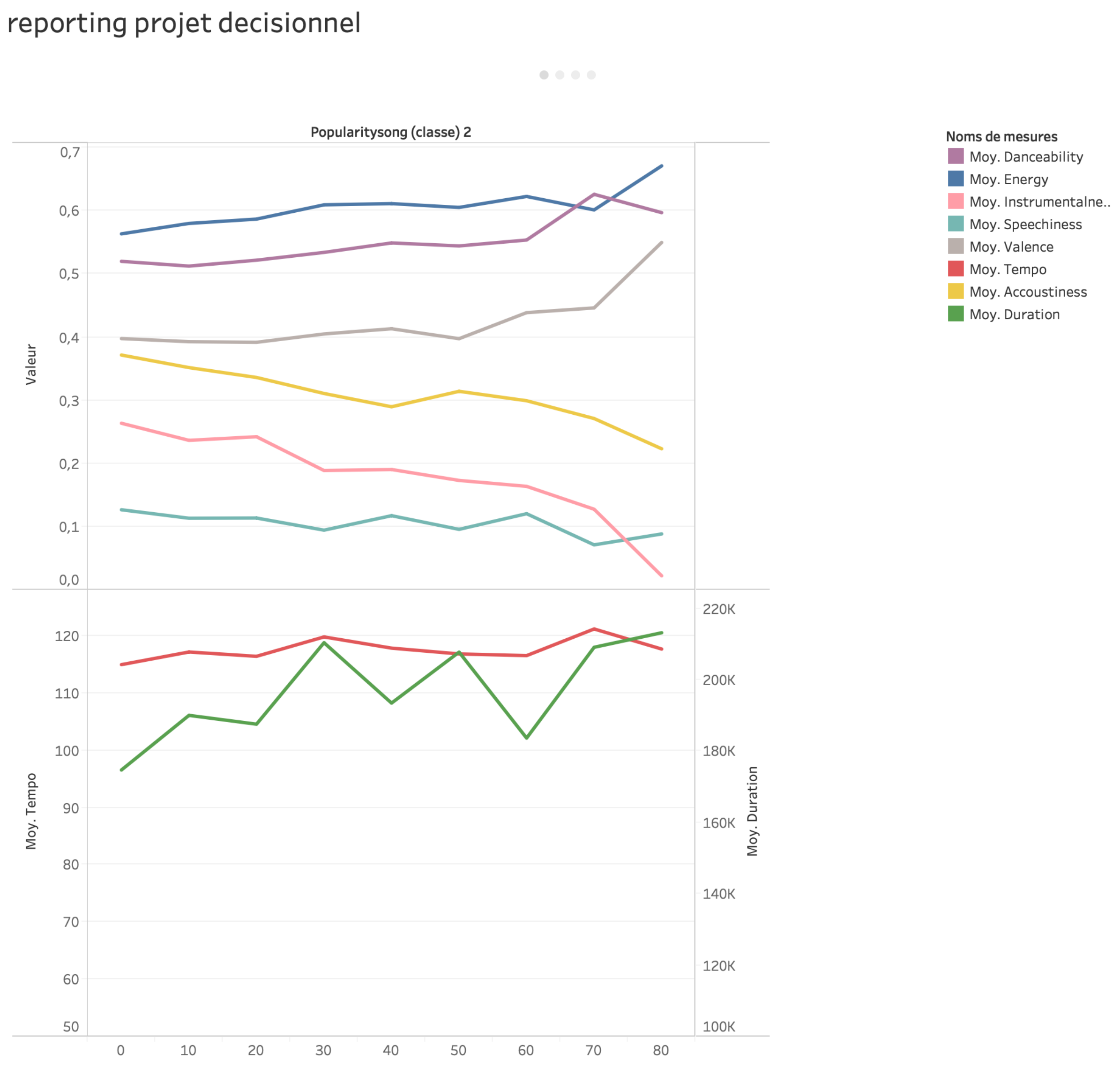


Figure 6.2.0-19: songs features function of popularity of the song

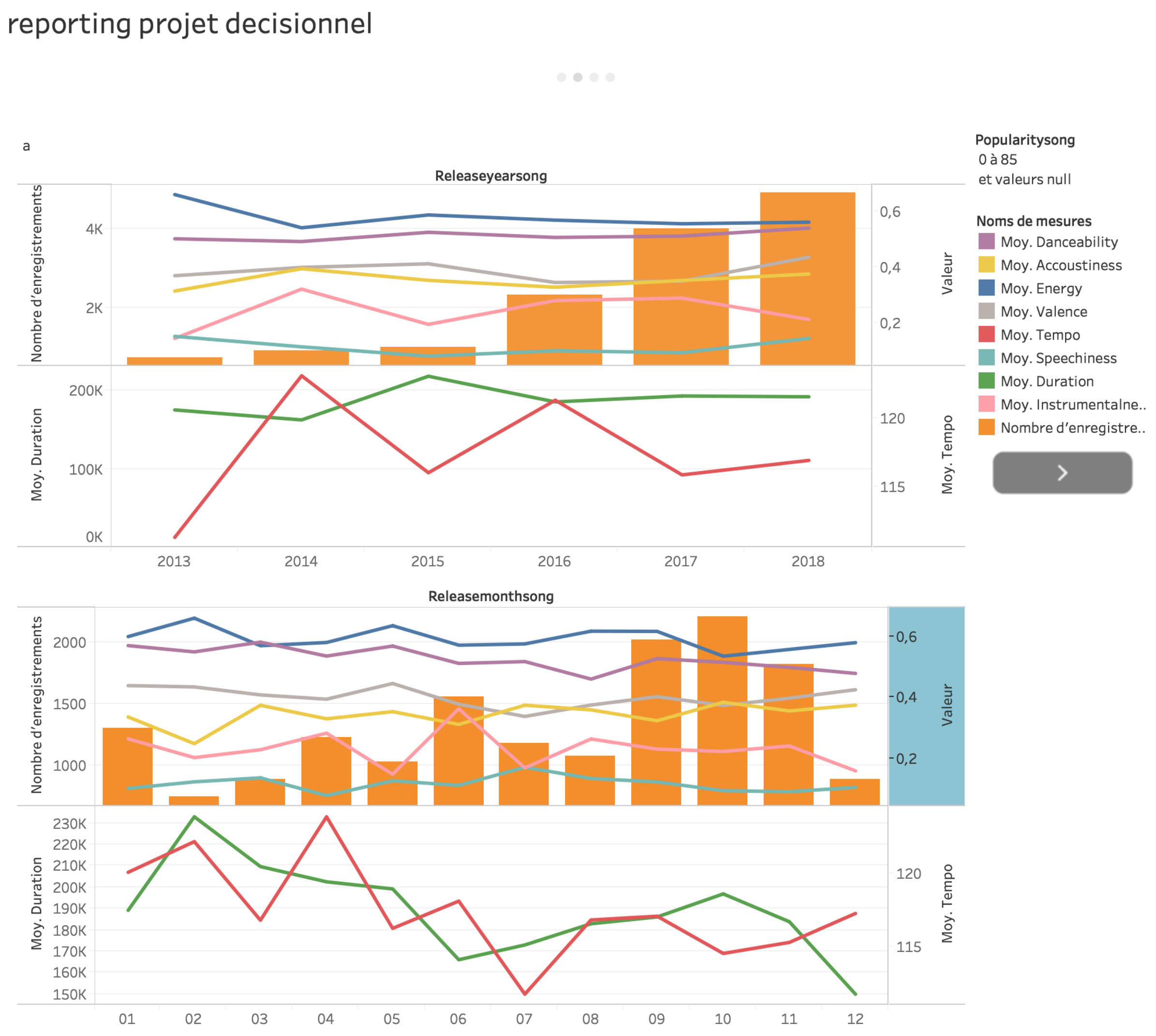


Figure 6.2.0-20: songs features function of years and month

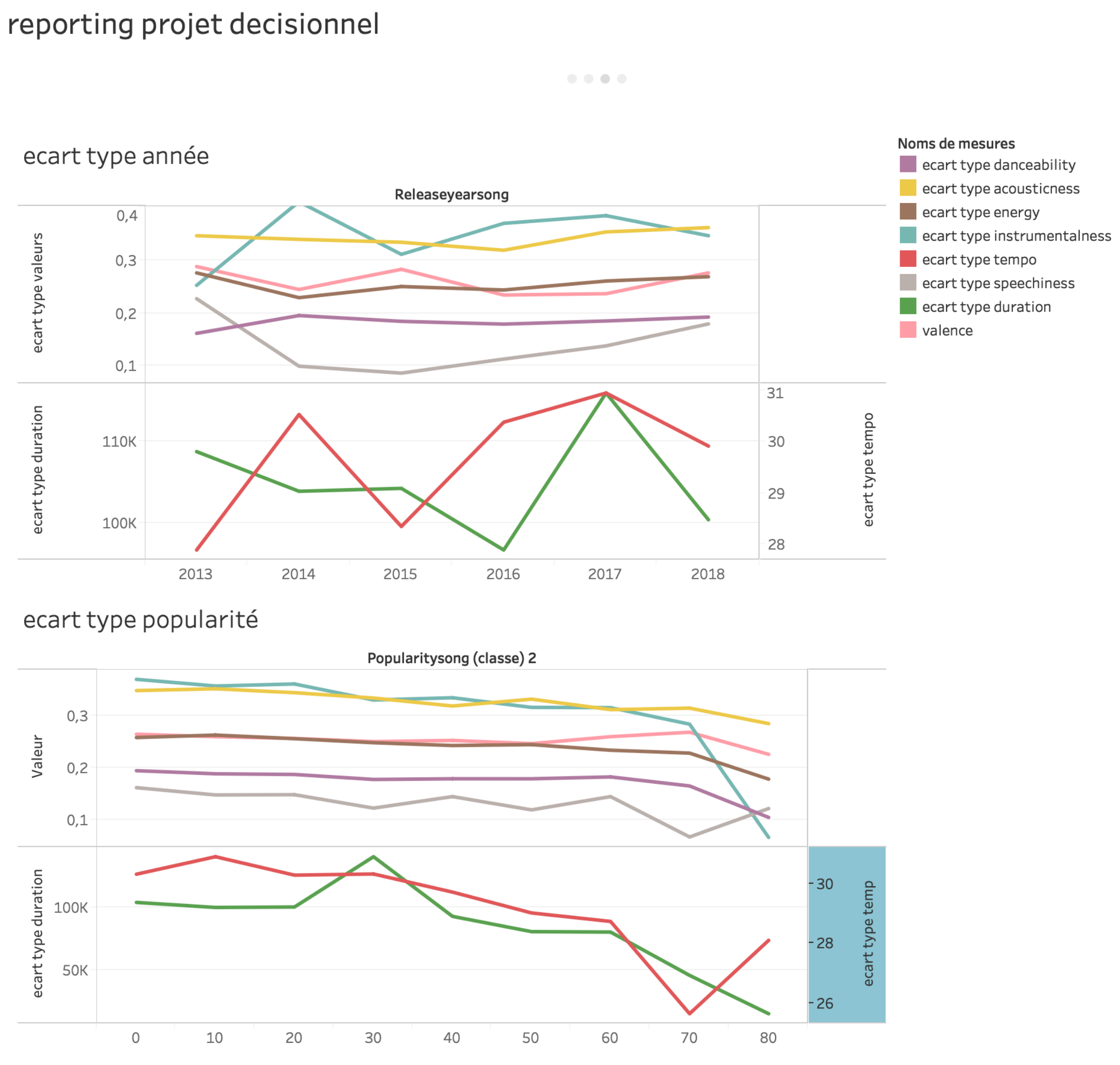


Figure 6.2.0-21: std function of years or popularity

6.2.1 small look at artist popularity rules on spotify

We maked this analysis with tableau.

We looked at the artists mean popularity of theirs songs function of the popularity of the artists. Their is a big correlation between the two. It seems that spotify order the artists by popularty of their songs and then assign them a popularity following a gauss repartition.

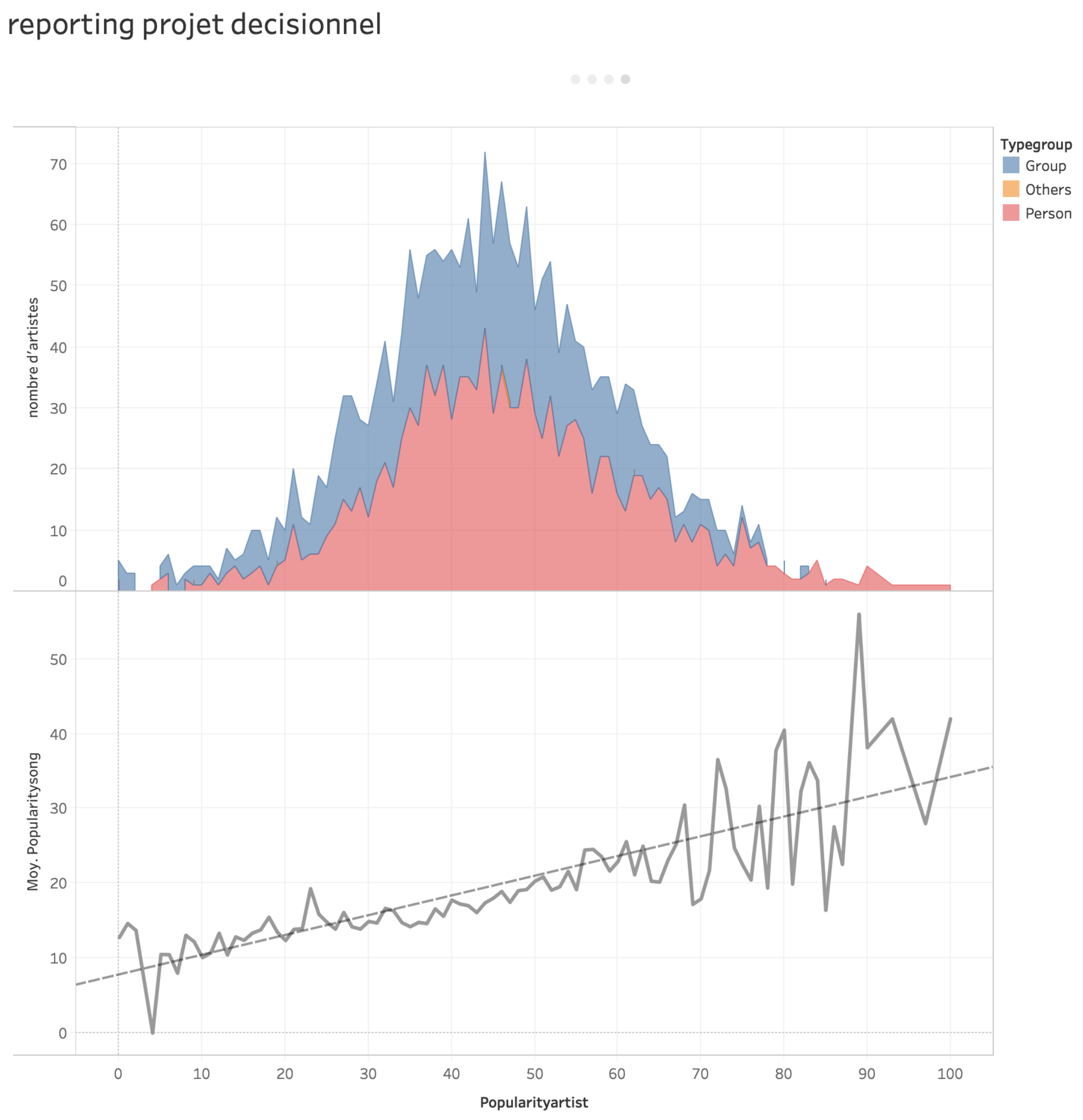


Figure 6.2.1-22: artists popularity

7 Conclusion

In this report, we introduced our proposed design for a data warehouse that aims to analyze music.We presented the architecture of our data warehousing system and the ETL process jobs implemented to extract , prepare and integrate data into the data warehouse. We discussed some performed analysis on data and extracted some interesting insights about the music French market .
As future works , we would love to continue working on this project using sophisticated tools like data mining to extract and predict more complex correlations between our data warehouse entities.

References

[1] Access public to on trello board. `https://trello.com/b/I2OALIiu/decisional-project`. accessed: 07.12.2018.

[2] Kanban. `https://www.tutorialspoint.com/kanban/kanban_tutorial.pdf`. accessed: 21.09.2018.

[3] Musicbrainz database and schema. `https://musicbrainz.org/doc/MusicBrainz_Database/Schema`. accessed: 21.09.2018.

[4] Spotify api. `https://developer.spotify.com/documentation/web-api/`. accessed: 21.09.2018.

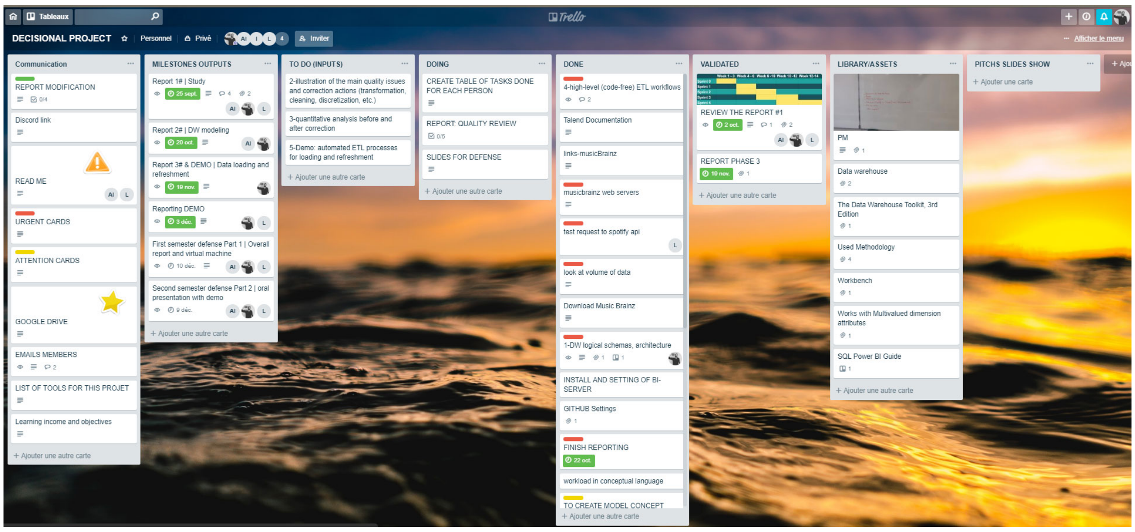
[5] Trello. `https://trello.com/`. accessed: 21.09.2018.

[6] Christian DESTREMAU. Méthode scrum partie 2 : définition de la méthode. `https://www.supinfo.com/articles/single/6054-methode-scrum-partie-2-definition-methode`. accessed: 21.09.2018.

[7] Matteo Golfarelli. Data Warehouse Design: Modern Principles and Methodologies.

[8] Ken Schwaber and Jeff Sutherland. The scrum guide^{TM} : The definitive guide to scrum: The rules of the game. `https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf`. accessed: 21.09.2018.

A Appendix: Trello Board



Tableaux

Trello

DECISIONAL PROJECT · Personnel · Privé · Inviter

Afficher le menu

Communication
- REPORT MODIFICATION 0/4
- Discord link
- READ ME
- URGENT CARDS
- ATTENTION CARDS
- GOOGLE DRIVE
- EMAILS MEMBERS 2
- LIST OF TOOLS FOR THIS PROJET
- Learning income and objectives
- + Ajouter une autre carte

MILESTONES OUTPUTS
- Report 1# | Study 25 sept. 4 2
- Report 2# | DW modeling 20 oct.
- Report 3# & DEMO | Data loading and refreshment 19 nov.
- Reporting DEMO 3 déc.
- First semester defense Part 1 | Overall report and virtual machine 10 déc.
- Second semester defense Part 2 | oral presentation with demo 9 déc.
- + Ajouter une autre carte

TO DO (INPUTS)
- 2-illustration of the main quality issues and correction actions (transformation, cleaning, discretization, etc.)
- 3-quantitative analysis before and after correction
- 5-Demo: automated ETL processes for loading and refreshment
- + Ajouter une autre carte

DOING
- CREATE TABLE OF TASKS DONE FOR EACH PERSON
- REPORT: QUALITY REVIEW 0/5
- SLIDES FOR DEFENSE
- + Ajouter une autre carte

DONE
- 4-high-level (code-free) ETL workflows 2
- Talend Documentation
- links-musicBrainz
- musicbrainz web servers
- test request to spotify api
- look at volume of data
- Download Music Brainz
- 1-DW logical schemas, architecture 1 1
- INSTALL AND SETTING OF BI-SERVER
- GITHUB Settings 1
- FINISH REPORTING 22 oct.
- workload in conceptual language
- TO CREATE MODEL CONCEPT
- + Ajouter une autre carte

VALIDATED
- REVIEW THE REPORT #1 2 oct. 1 2
- REPORT PHASE 3 19 nov. 1
- + Ajouter une autre carte

LIBRARY/ASSETS
- PM 1
- Data warehouse 2
- The Data Warehouse Toolkit, 3rd Edition 1
- Used Methodology 4
- Workbench 1
- Works with Multivalued dimension attributes 1
- SQL Power BI Guide 1
- + Ajouter une autre carte

PITCHS SLIDES SHOW
- + Ajouter une carte

B Appendix: Extraction Python Scripts

B.0.1 Script that extracts artists

```
#!/usr/bin/env python
# coding: utf-8
#python -m pip install xlsxwriter
import spotipy
from spotipy.oauth2 import SpotifyClientCredentials
import pandas as pd
import time
from openpyxl import load_workbook
import os.path
import psycopg2
import xlsxwriter
#import pgdb

hostname = '10.195.25.10'
username = 'userbkp'
password = 'goma123'
database = 'musicbrainz'
port='54587'
```

```
cid ="a154f4842df643a99f9057fb741c86e0"
secret = "543080531fce4f30be3da2c36782ace1"

today_date=(time.strftime("%d_%m_%Y  %M_%S"))
date=(time.strftime("%d/%m/%Y"))
date_monthYear = (time.strftime("%m/%Y"))
date_year = (time.strftime("%Y"))

filepath = 'F:\Documents\dproject_bdma2018\DATA'+today_date+'.xlsx'


checkifexist ="""SELECT id_msbz
                FROM public.stage_artist
                WHERE id_msbz ="""



query = """SELECT distinct artist_table.name_group,
             artist_table.id as id_msbz,
             artist_table.position as id_member_position,
             artist_table.type as id_type_group,
             artist_table.area as id_country_origin,
             area_table.countrycity_name as country_origin,
             artist_table.begin_area as id_city_origin ,
             (SELECT musicbrainz.area.name
            FROM musicbrainz.area
            WHERE musicbrainz.area.id = artist_table.begin_area
            AND musicbrainz.area.type = 3) as city_origin,
             gender_table.name as gender_sex

            FROM (SELECT musicbrainz.artist.name as name_group,*
              FROM musicbrainz.artist
              INNER JOIN musicbrainz.artist_credit_name
            ON musicbrainz.artist.id =
             musicbrainz.artist_credit_name.artist
             where musicbrainz.artist.type = 2) as artist_table
            LEFT JOIN musicbrainz.gender as gender_table
            ON artist_table.gender = gender_table.id
            INNER JOIN (select musicbrainz.area.id as id_area,
             musicbrainz.area.name as countrycity_name,
                musicbrainz.area_type.name as countrycity_type
                      FROM musicbrainz.area
                      INNER JOIN musicbrainz.area_type
                      on musicbrainz.area.type = musicbrainz.area_type.id
                      INNER JOIN (SELECT musicbrainz.area_alias.area as
                      id_area
                      FROM musicbrainz.area_alias
                      WHERE musicbrainz.area_alias.locale = 'en')
                      as area_alias_table
                      on musicbrainz.area.id = area_alias_table.id_area
                      WHERE musicbrainz.area_type.id = 1) as area_table
                ON artist_table.area = area_table.id_area

                WHERE artist_table.area is not null AND
                artist_table.begin_area is not null
                AND artist_table.area != artist_table.begin_area"""

client_credentials_manager = SpotifyClientCredentials(client_id=cid,
                             client_secret=secret)
sp = spotipy.Spotify(client_credentials_manager=
client_credentials_manager)



def doQuery( conn ) :

    cur = conn.cursor()
    cur.execute(query)
    return cur.fetchall()

def existsOnStageQuery( check_id ) :
    connec = psycopg2.connect(host=hostname,
                             user=username,
                             password=password,
                             dbname='stages',
                             port=port )
    cursor = connec.cursor()
    cursor.execute(checkifexist+str(check_id))
    print (cursor.rowcount)
    if cursor.rowcount == 0:
      cursor.close()
      connec.close()
      return False
    else:
      cursor.close()
      connec.close()
      return True




artist_musicbrainz =[]

myConnection = psycopg2.connect( host=hostname,
                                    user=username,
                                    password=password,
                                    dbname=database, port=port )
artist_musicbrainz = doQuery( myConnection )
myConnection.close()
# print size list



#Verify if File in certain date already exists
filename=filepath
if os.path.exists(filename):
    print(filename+' '+'exists')
    print("Extraction is starts...")
else:
    # create empty lists where the results are going to be stored
    artist_name_group = []
    artist_id_msbz = []
    artist_id_member_position = []
    artist_id_type_group = []
    artist_id_country_origin =[]
    artist_country_origin =[]
    artist_id_city_origin = []
    artist_city_origin = []
    artist_gender_sex = []
    #spotify attributes
    artist_popularity = []
    artist_id_spotify = []
    artist_followers = []
    artist_type = []
    artist_genre = []
    artist_gender = []



for i in range(0,len(artist_musicbrainz)-1,1):
for name_group,
id_msbz,id_member_position,
id_type_group,id_country_origin,
country_origin,id_city_origin,
city_origin,gender_sex in artist_musicbrainz :
track_results =
sp.search(q=name_group,
type='artist',market='FR', limit=50)
#print (track_results)
for i, t in enumerate(track_results):
  if  not track_results[t]['items']:
      #print(track_results[t]['items'])
      print('artiste non trouv[U+FFFD]')
  elif (existsOnStageQuery(id_msbz)):
      print "$$$$$ —> I am already here. Rock n' Roll"
  else :
      try:
        print ('Into try')
        stageConnection = psycopg2.connect( host=hostname,
        user=username, password=password,
        dbname='stages',port=port )
        cur = stageConnection.cursor()

        print (track_results[t]['items'][0]['name']+",
        FROM ———>"+country_origin)

        cur.execute("""INSERT INTO
        public.stage_artist(id_msbz,
                 name_group,
                 id_member_position,
                 id_type_group,
                 id_country_origin,
                 country_origin,
                 id_city_origin,
                 city_origin,
                 gender_sex,
                 popularity,
                 id_spotify,
                 followers,
                 genre, date_full,
                 date_monthYear,
                 date_year)
               VALUES (%s,%s,%s,%s,%s,%s,%s,%s,
               %s,%s,%s,%s,%s,%s,%s,%s)""",
               (id_msbz,
                 track_results[t]['items'][0]['name'],
                 id_member_position,
                 id_type_group,
                 id_country_origin,
                 country_origin,
                 id_city_origin,
                 city_origin,
                 gender_sex,
                 track_results[t]['items'][0]['popularity'],
                 track_results[t]['items'][0]['id'],
                 track_results[t]['items'][0]['followers']['total'],
                 track_results[t]['items'][0]['genres'],
                 date,
                 date_monthYear,
                 date_year))

        cur.close()
        stageConnection.commit()
        stageConnection.close()

print ('finish -> connection')
except :
```

```
#    print e
   print ('##IGNORED##')
```

B.0.2 Script that extracts albums

```
#!/usr/bin/env python

import spotipy
from spotipy.oauth2 import SpotifyClientCredentials
import pandas as pd
import psycopg2
import time
from sqlalchemy import create_engine


#musicBrainzCredentials
hostname = '10.195.25.10'
username = 'userbkp'
password = 'goma123'
database = 'stages'
port='54587'

#SpotifyCredentials
cid ="5f0b01a1813a41d1b360ed91e890a93f"
secret ="39877ef7e0a9467db85353f6090b0cbd"

#Declaration
today_date=(time.strftime("%d_%m_%Y"))
date=(time.strftime("%d_%m_%Y"))

client_credentials_manager = SpotifyClientCredentials(client_id=cid,
client_secret=secret)
sp = spotipy.Spotify(client_credentials_manager=
client_credentials_manager)

# Request MusicBrainz For artists using stage_artist
querySelect = """ SELECT id_msbz,id_spotify, name_group
FROM public.stage_artist ; """


def doReturnArtists( conn ) :
    cur = conn.cursor()
    cur.execute(querySelect)
    return cur.fetchall()

# Insert Albums in stage_Album
queryInsert= """INSERT INTO
public.stage_album (artist_name,
artist_id_msbz,
artist_id_spotify ,album_id,
album_name,nb_tracks_album)
values(%s,%s,%s,%s,%s,%s)"""

def doInsertAlbums( artist_id_spotify ,artist_id_msbz,
artist_name,album_id,album_name,nb_tracks_album) :
    myconnection =psycopg2.connect( host=hostname,
    user=username, password=password, dbname=database,
    port=port )
    cur = myconnection.cursor()
    cur.execute(queryInsert,( artist_id_spotify ,
    artist_id_msbz,artist_name,album_id,album_name,
    nb_tracks_album))
    cur.close()
    myconnection.commit()
    myConnection.close()

myConnection = psycopg2.connect( host=hostname,
user=username, password=password, dbname=database,port=port )
artist_musicbrainz = doReturnArtists( myConnection )
myConnection.close()
print ('here')

# create empty lists where the results are going to be stored
artist_namelist = []
artist_id_msbzlist= []
artist_id_spotifylist = []
album_idlist = []
nb_tracks_albumlist=[]
for i in range(0,len(artist_musicbrainz)-1,1):
    print ('*here')
    for id_msbz,id_spotify, name_group in artist_musicbrainz:
        print(id_spotify)
        albums_results=sp.artist_albums(id_spotify,
        album_type='album', limit=20, offset=0)
        for i, t in enumerate(albums_results['items']):
            print ('**here')
            print (t['id'])
            artist_namelist.append(name_group)
            artist_id_msbzlist.append(id_msbz)
            artist_id_spotifylist.append(id_spotify)
            album_idlist.append (t['id'])
            nb_tracks_albumlist.append(t['total_tracks'])
            print (t['total_tracks'])
            doInsertAlbums(name_group,id_msbz,id_spotify,
            t['id'],t['name'],t['total_tracks'])
            print('Inserted**')
```

B.0.3 Script that extracts tracks

```
#!/usr/bin/env python
import spotipy
from spotipy.oauth2 import SpotifyClientCredentials
import pandas as pd
import psycopg2
import time
from sqlalchemy import create_engine

#musicBrainzCredentials
hostname = '10.195.25.10'
username = 'userbkp'
password = 'goma123'
database = 'stages'
port='54587'

date_popularity=(time.strftime("%d/%m/%Y"))

cid ="a154f4842df643a99f9057fb741c86e0"
secret = "543080531fce4f30be3da2c36782ace1"

client_credentials_manager =
SpotifyClientCredentials(client_id=cid,
client_secret=secret)
sp = spotipy.Spotify(client_credentials_manager=
client_credentials_manager)

query = """ SELECT artist_name, artist_id_msbz,
artist_id_spotify, album_id, album_name FROM public.stage_album
; """

queryInsert=""" INSERT INTO
public.stage_track_features(
            artist_id_spotify, artist_id_msbz,
            artist_name, album_id, album_name,
            track_id, track_name, popularity,
            date_popularity, acousticness,
            danceability, duration_ms, energy,
            instrumentalness, key_f, liveness,
            loudness, mode_f, speechiness, tempo,
            time_signature, valence)
    VALUES (%s, %s, %s, %s, %s,
            %s, %s, %s, %s, %s,
            %s, %s, %s, %s, %s, %s, %s,
            %s, %s, %s,%s, %s);

"""

def doQuery( conn ) :

    cur = conn.cursor()
    cur.execute(query)
    return cur.fetchall()

def doInsertTracks(artist_id_spotify, artist_id_msbz,
artist_name, album_id, album_name,
            track_id, track_name, popularity,
            date_popularity, acousticness,
            danceability, duration_ms, energy,
            instrumentalness, key_f, liveness,
            loudness, mode_f, speechiness, tempo,
            time_signature, valence) :
    myconnection =psycopg2.connect( host=hostname,
    user=username, password=password, dbname=database,port=port )
    cur = myconnection.cursor()
    cur.execute(queryInsert,(artist_id_spotify,
    artist_id_msbz, artist_name, album_id, album_name,
            track_id, track_name, popularity,
            date_popularity, acousticness,
            danceability, duration_ms, energy,
            instrumentalness, key_f, liveness,
            loudness, mode_f, speechiness,
            tempo, time_signature, valence))
    cur.close()
    myconnection.commit()
    myConnection.close()

myConnection = psycopg2.connect( host=hostname,
user=username, password=password, dbname=database,port=port )
album_lists = doQuery( myConnection )
myConnection.close()
print ('here')


for i in range(0,len(album_lists)-1,1):
    print ('*here')
    print(i)
    for artist_name, artist_id_msbz, artist_id_spotify,
    album_id, album_name in album_lists :
        track_results=sp.album_tracks(album_id ,
        limit=20, offset=0)
        for i, t in enumerate(track_results['items']):
            print ('**here')
            track_name=t['name'] #track name
            print(t['name']) #track name
            print(artist_name) #artist name
            print(album_name)#album name
            track_id=t['id']#track id
            print(t['id']) #track id
            print('**here')
            track_features=sp.audio_features(t['id'])
            track_populartiy=sp.track(t['id'])
            popularity =track_populartiy['popularity']
```

```
for r in track_features:
    try:
        doInsertTracks(artist_id_spotify ,
        artist_id_msbz ,artist_name , album_id ,
        album_name,track_id ,track_name ,
        popularity ,
                    date_popularity
                    ,r['acousticness'],
                    r['danceability'],
                    r['duration_ms'],
                    r['energy'],
                    r['instrumentalness'],
                    r['key'],r['liveness'],
                    r['loudness'],
                    r['mode'],
                    r['speechiness'],
                    r['tempo'],
                    r['time_signature'],
                    r['valence'])
    except:
        print("Ignore")
```