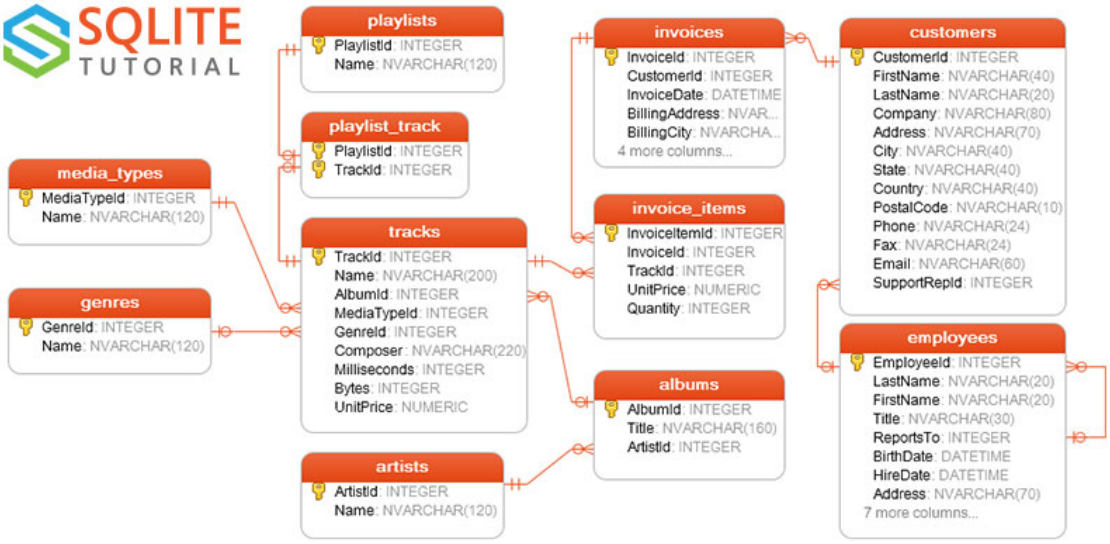
TD-BDD_revision_v1

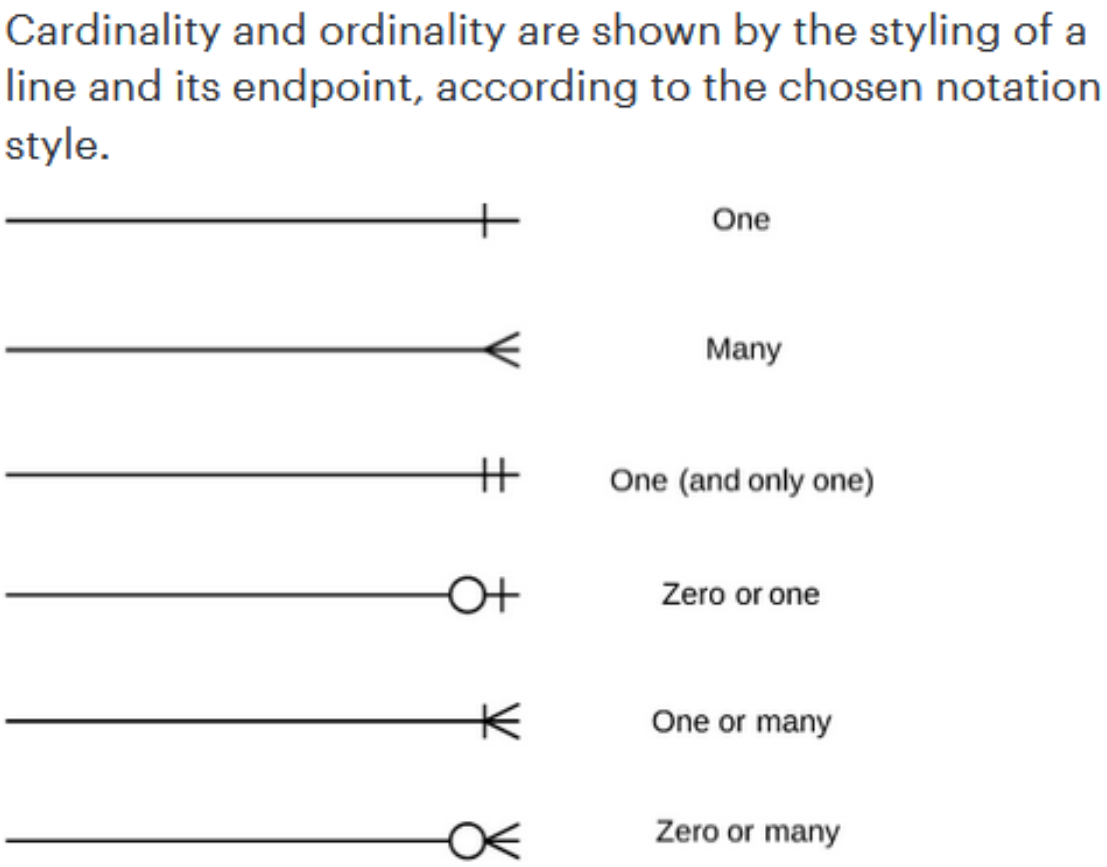
November 21, 2021

[4]:
```
import sqlite3
```

https://www.youtube.com/watch?v=pqoIBiM2AvE

https://www.youtube.com/watch?v=Now9xiWQp78





[5]:
```
## Creation d'une connection
connection = sqlite3.connect("chinook.db")
cursor=connection.cursor()
def sql(od):
    res = cursor.execute(od).fetchall()
    return res
```

[6]:
```
## Affichage la liste de toutes les tables
TABLELIST = cursor.execute("SELECT name FROM sqlite_master WHERE type='table';
 ↪").fetchall()
print(list(map(lambda x: x[0],TABLELIST))) # fonction map(func, iterables)
```

```
['albums', 'sqlite_sequence', 'artists', 'customers', 'employees', 'genres',
'invoices', 'invoice_items', 'media_types', 'playlists', 'playlist_track',
'tracks', 'sqlite_stat1']
```

0.1 Opérations binaires: projections et projections

[4]:
```
# Donner la liste de tous les genres musicaux.
sql('SELECT Name FROM genres') # c'est une projection
```

[4]:
```
[('Rock',),
 ('Jazz',),
 ('Metal',),
 ('Alternative & Punk',),
 ('Rock And Roll',),
 ('Blues',),
 ('Latin',),
 ('Reggae',),
 ('Pop',),
 ('Soundtrack',),
 ('Bossa Nova',),
 ('Easy Listening',),
 ('Heavy Metal',),
 ('R&B/Soul',),
 ('Electronica/Dance',),
 ('World',),
 ('Hip Hop/Rap',),
 ('Science Fiction',),
 ('TV Shows',),
 ('Sci Fi & Fantasy',),
 ('Drama',),
 ('Comedy',),
 ('Alternative',),
 ('Classical',),
 ('Opera',)]
```

Caractères spéciaux :

- "%" remplace tous les autres caractères,
- "_" remplace un caractère par n'importe lequel

[5]:
```
# Sélectionner les lignes de la table genres dont le Name de Genre comme par la␣
↪lettre E
sql("SELECT * FROM genres WHERE Name LIKE 'E%'")
```

[5]:
```
[(12, 'Easy Listening'), (15, 'Electronica/Dance')]
```

[6]:
```
# Donner le nom des musiques dont le nom du compositeur se termine par la␣
↪lettre x
sql("SELECT Name,Composer FROM tracks WHERE Composer LIKE '%x'")
```

[6]:
```
[("Nobody Knows You When You're Down & Out", 'Jimmy Cox'),
 ('Foxy Lady', 'Jimi Hendrix'),
```

```
('Manic Depression', 'Jimi Hendrix'),
('Red House', 'Jimi Hendrix'),
('Can You See Me', 'Jimi Hendrix'),
('Love Or Confusion', 'Jimi Hendrix'),
("I Don't Live Today", 'Jimi Hendrix'),
('May This Be Love', 'Jimi Hendrix'),
('Fire', 'Jimi Hendrix'),
('Third Stone From The Sun', 'Jimi Hendrix'),
('Remember', 'Jimi Hendrix'),
('Are You Experienced?', 'Jimi Hendrix'),
('Stone Free', 'Jimi Hendrix'),
('Purple Haze', 'Jimi Hendrix'),
('51st Anniversary', 'Jimi Hendrix'),
('The Wind Cries Mary', 'Jimi Hendrix'),
('Highway Chile', 'Jimi Hendrix'),
('Kickstart My Heart', 'Nikki Sixx'),
('Dr. Feelgood', 'Mick Mars/Nikki Sixx'),
('Afraid', 'Nikki Sixx'),
("Don't Go Away Mad (Just Go Away)", 'Mick Mars/Nikki Sixx'),
('Without You', 'Mick Mars/Nikki Sixx'),
('Too Fast For Love', 'Nikki Sixx'),
('Looks That Kill', 'Nikki Sixx'),
('Shout At The Devil', 'Nikki Sixx')]
```

[ ]:
```
# Donner pour chaque musique : son nom et sa durée en minutes, affiché avec des␣
↪virgules
sql('SELECT Name, milliseconds/60000. FROM tracks')
```

0.2 ORDER BY

Tri des champs

Tri en ordre inverse ORDER BY champ DESC

[ ]:
```
# Donner pour la liste des musiques et de leur durées en minute par ordre␣
↪croissant de durée
sql('SELECT Name, milliseconds/60000. dureemin FROM tracks ORDER BY dureemin')
```

[9]:
```
# Afficher tous les genres musicaux triés par ordre alphabétique inverse
sql('SELECT Name FROM genres ORDER BY Name DESC')
```

[9]:
```
[('World',),
 ('TV Shows',),
 ('Soundtrack',),
 ('Science Fiction',),
 ('Sci Fi & Fantasy',),
 ('Rock And Roll',),
```

```
('Rock',),
('Reggae',),
('R&B/Soul',),
('Pop',),
('Opera',),
('Metal',),
('Latin',),
('Jazz',),
('Hip Hop/Rap',),
('Heavy Metal',),
('Electronica/Dance',),
('Easy Listening',),
('Drama',),
('Comedy',),
('Classical',),
('Bossa Nova',),
('Blues',),
('Alternative & Punk',),
('Alternative',)]
```

0.3 SELECT DISTINCT : suppression de doublons

[ ]:
```
## Donner la liste de tous les compositeurs
sql('SELECT DISTINCT composer FROM tracks')
```

Pour s'entrainer...

[11]:
```
## Donner le Prenom, Nom et date d'embauche des employés dont le prénom␣
↪contient la lettre 'm'
sql('SELECT FirstName, LastName, HireDate FROM employees WHERE FirstName LIKE␣
↪"%m%" ;')
```

[11]:
```
[('Margaret', 'Park', '2003-05-03 00:00:00'),
 ('Michael', 'Mitchell', '2003-10-17 00:00:00')]
```

[12]:
```
## Donner la liste des pays dont font partie les clients
sql('SELECT DISTINCT country FROM customers ')
```

[12]:
```
[('Brazil',),
 ('Germany',),
 ('Canada',),
 ('Norway',),
 ('Czech Republic',),
 ('Austria',),
 ('Belgium',),
 ('Denmark',),
 ('USA',),
```

```
('Portugal',),
('France',),
('Finland',),
('Hungary',),
('Ireland',),
('Italy',),
('Netherlands',),
('Poland',),
('Spain',),
('Sweden',),
('United Kingdom',),
('Australia',),
('Argentina',),
('Chile',),
('India',)]
```

0.4 Jointures

syntaxe :

Première possibilité: on fait une sélection à partir du produit cartésien de tables.

SELECT attribut1, attribut2, … FROM tableA, tableB WHERE conditions de jointure

Seconde possibilité: on utilise JOIN ON

SELECT attribut1, attribut2, … FROM tableA JOIN tableB ON conditions de jointure

[ ]:
```
## Donner la liste des titres d'album et du nom d'artiste.
sql('SELECT al.Title,ar.Name FROM albums al, artists ar WHERE ar.artistid = al.
 ↪artistid')
# Noter le renommage des tables avec des alias
```

[ ]:
```
## Donner la liste des titres d'album et du nom d'artiste.
# Avec JOIN ON
sql('SELECT al.Title, ar.Name FROM albums al JOIN artists ar ON ar.artistid =␣
 ↪al.artistid')

# Noter le renommage des tables avec des alias
```

0.4.1 JOINTURES MULTIPLES

Syntaxe:

SELECT attribut1, attribut2, … FROM tableA JOIN tableB JOIN tableC ON conditions de jointure

[ ]:
```
## Donner la table (Nom de la playlist, Nom de la musique (tracks))
sql('SELECT pl.Name, t.Name FROM playlists pl JOIN playlist_track pt JOIN␣
 ↪tracks t ON pt.playlistid=pl.playlistid AND t.trackid=pt.trackid')
```

[16]:
```
##Donner l'ensemble des artistes de la playlists dont l'id vaut 3
sql('''SELECT DISTINCT ar.Name
FROM artists ar JOIN albums al JOIN tracks t JOIN playlist_track pt JOIN␣
 ↪playlists pl
ON ar.artistid = al.artistid AND  al.albumid = t.albumid
AND t.trackid=pt.trackid AND pt.playlistid = pl.playlistid
WHERE pl.playlistid=3''')
```

[16]:
```
[('Battlestar Galactica',),
 ('Heroes',),
 ('Lost',),
 ('The Office',),
 ('Battlestar Galactica (Classic)',),
 ('Aquaman',)]
```

0.5 Fonctions d'agrégats

COUNT (nombre de tuples), SUM(somme), AVG (average), MAX, MIN.

[17]:
```
## Combien y a-t-il de playlists?
sql('SELECT COUNT(*) FROM playlists')
```

[17]:
```
[(18,)]
```

[18]:
```
## autre possibilité
sql('SELECT COUNT(playlistid) FROM playlists')
```

[18]:
```
[(18,)]
```

[19]:
```
## Quelles est, en heures, la durée de la musique la plus longue?
sql('SELECT MAX(milliseconds/3600000.) heures FROM tracks')
```

[19]:
```
[(1.4685980555555556,)]
```

[20]:
```
## Quelle est la durée moyenne, en minutes, d'un morçeau ?
sql('SELECT AVG(milliseconds/60000) minutes FROM tracks')
```

[20]:
```
[(6.057664858692549,)]
```

[21]:
```
## Quelle est la durée moyenne d'un morçeau de genre 'Blues' ?
sql('SELECT AVG(t.milliseconds/60000) duréeeMin FROM tracks t JOIN genres g ON␣
 ↪g.genreid=t.genreid WHERE g.Name="Blues"')
```

[21]:
```
[(4.012345679012346,)]
```

[22]:
```
```

```
## Combien y a-t-il de morçeaux de genres 'Blues'?
sql('SELECT COUNT(*) FROM tracks t JOIN genres g ON g.genreid=t.genreid WHERE g.
 ↪Name="Blues"')
```

[22]: `[(81,)]`

/! ATTENTION

[23]:
```
## Combien y a-t-il de compositeurs différents parmi toutes les musiques de␣
 ↪tracks?
sql('SELECT COUNT(DISTINCT composer) FROM tracks')
```

[23]: `[(852,)]`

[24]:
```
## Combien y a-t-il de genres différents dans la playlist dont l'id vaut 3
sql('SELECT count(DISTINCT g.Name) FROM genres g JOIN tracks t JOIN␣
 ↪playlist_track pt ON g.genreid=t.genreid AND t.trackid=pt.trackid AND pt.
 ↪playlistid=3')
```

[24]: `[(5,)]`

1 Sous requêtes

On peut effectuer une requête dans une requête.

[28]:
```
### Renvoyer le nom de la musique la plus longue
sql('''SELECT Name
    FROM tracks
    WHERE milliseconds =
        (SELECT MAX(milliseconds)
        FROM tracks)''')
```

[28]: `[('Occupation / Precipice',)]`

[41]:
```
### Renvoyer le nom de la musique la plus longue de la playlist 'Classical
sql('''SELECT Name
    FROM tracks
    WHERE milliseconds =
        (SELECT MAX(t.milliseconds)
        FROM tracks t JOIN playlist_track pt JOIN playlists pl
        ON t.trackId=pt.TrackId AND pt.playlistId= pl.playlistId
        WHERE pl.Name="Classical")''')
```

[41]: `[('Adagio for Strings from the String Quartet, Op. 11',)]`

2 PARTITIONNEMENT

GROUP BY

L'attribut de groupage doit toujours être dans la clause select.

[ ]:
```
## Donner pour chaque nom de playlist sa durée totale en heure
sql('''SELECT pl.Name, SUM(t.milliseconds/3600000.)
FROM playlists pl JOIN playlist_track pt JOIN tracks t
ON pl.playlistid=pt.playlistid AND t.trackid= pt.trackid
GROUP BY pl.Name''')
```

[26]:
```
## On peut évidemment trier les résultats par durée totale
## Donner pour chaque nom de playlist sa durée totale en heure
sql('''SELECT pl.Name, SUM(t.milliseconds/3600000.) dureeTotale
FROM playlists pl JOIN playlist_track pt JOIN tracks t
ON pl.playlistid=pt.playlistid AND t.trackid= pt.trackid
GROUP BY pl.Name ORDER BY dureeTotale''')
```

[26]:
```
[('On-The-Go 1', 0.05484972222222222),
 ('Music Videos', 0.08174833333333334),
 ('Grunge', 1.1450049999999998),
 ('Classical 101 - Deep Cuts', 1.8765916666666667),
 ('Classical 101 - The Basics', 2.0666141666666666),
 ('Classical 101 - Next Steps', 2.1041808333333334),
 ('Heavy Metal Classic', 2.279531111111111),
 ('Brazilian Music', 2.6351552777777782),
 ('Classical', 6.047386666666667),
 ('90's Music', 110.75143138888888),
 ('TV Shows', 278.3860872222224),
 ('Music', 487.6017127777833)]
```

[27]:
```
# On peut demander d'afficher en plus le nombre de morçeaux dans chacune de ces␣
 ↪playlists
sql('''SELECT pl.Name, SUM(t.milliseconds/3600000.) dureeTotale, COUNT(t.
 ↪trackid)
FROM playlists pl JOIN playlist_track pt JOIN tracks t
ON pl.playlistid=pt.playlistid AND t.trackid= pt.trackid
GROUP BY pl.Name ORDER BY dureeTotale''')
```

[27]:
```
[('On-The-Go 1', 0.05484972222222222, 1),
 ('Music Videos', 0.08174833333333334, 1),
 ('Grunge', 1.1450049999999998, 15),
 ('Classical 101 - Deep Cuts', 1.8765916666666667, 25),
 ('Classical 101 - The Basics', 2.0666141666666666, 25),
 ('Classical 101 - Next Steps', 2.1041808333333334, 25),
 ('Heavy Metal Classic', 2.279531111111111, 26),
 ('Brazilian Music', 2.6351552777777782, 39),
```

```
('Classical', 6.047386666666667, 75),
('90's Music', 110.75143138888888, 1477),
('TV Shows', 278.3860872222224, 426),
('Music', 487.6017127777833, 6580)]
```

On peut ensuite sélectionner certains tuplpes de cette nouvelle table à l'aide de la clause HAVING.

[28]:
```
## Donner les noms des playlists dont la durée totale dépasse 3 heures
sql('''SELECT pl.Name, SUM(t.milliseconds/3600000.) dureeT
FROM playlists pl JOIN playlist_track pt JOIN tracks t
ON pl.playlistid=pt.playlistid AND t.trackid= pt.trackid
GROUP BY pl.Name HAVING dureeT >3. ''')
```

[28]:
```
[('90's Music', 110.75143138888888),
 ('Classical', 6.047386666666667),
 ('Music', 487.6017127777833),
 ('TV Shows', 278.3860872222224)]
```

[29]:
```
#On peut demander de classer les résultats par ordre croissant de durée
## Donner les noms des playlists dont la durée totale dépasse 3 heures
sql('''SELECT pl.Name, SUM(t.milliseconds/3600000.) dureeT
FROM playlists pl JOIN playlist_track pt JOIN tracks t
ON pl.playlistid=pt.playlistid AND t.trackid= pt.trackid
GROUP BY pl.Name HAVING dureeT >3. ORDER BY dureeT''')
```

[29]:
```
[('Classical', 6.047386666666667),
 ('90's Music', 110.75143138888888),
 ('TV Shows', 278.3860872222224),
 ('Music', 487.6017127777833)]
```

[30]:
```
# NOTER QUE HAVING doit être AVANT le ORDER BY
#On peut demander de classer les résultats par ordre croissant de durée
## Donner les noms des playlists dont la durée totale dépasse 3 heures
sql('''SELECT pl.Name, SUM(t.milliseconds/3600000.) dureeT
FROM playlists pl JOIN playlist_track pt JOIN tracks t
ON pl.playlistid=pt.playlistid AND t.trackid= pt.trackid
GROUP BY pl.Name ORDER BY dureeT HAVING dureeT >3. ''')
```

```
␣
↪---------------------------------------------------------------------------

OperationalError                          Traceback (most recent call␣
↪last)

<ipython-input-30-c25ac322cc51> in <module>
  5 FROM playlists pl JOIN playlist_track pt JOIN tracks t
  6 ON pl.playlistid=pt.playlistid AND t.trackid= pt.trackid
```

```
----> 7 GROUP BY pl.Name ORDER BY dureeT HAVING dureeT >3. ''')


    <ipython-input-2-1c73ebf1968c> in sql(od)
      3 cursor=connection.cursor()
      4 def sql(od):
----> 5     res = cursor.execute(od).fetchall()
      6     return res


    OperationalError: near "HAVING": syntax error
```

Clause LIMIT

On peut limiter le nombre de lignes à l'aide de la clause

`LIMIT nombre_max`

[56]:
```
## Donner les dix premiers de noms de musiques commençant par la lettre B et
## triés par ordre alphabétique, sans doublon
sql('''SELECT DISTINCT name
FROM tracks WHERE name LIKE "B%"
ORDER BY name
LIMIT 10''')
```

[56]:
```
[('B.Y.O.B.',),
 ("Baba O'Riley",),
 ("Babe I'm Gonna Leave You",),
 ('Baby',),
 ('Baby Break It Down',),
 ('Babyface',),
 ('Babylon',),
 ('Back Door Man',),
 ('Back from Vacation',),
 ('Back in the Village',)]
```

[71]:
```
## Comment avoir les dix suivants par ordre alphabétique?
# SOLUTION : --> On utilise une requête imbriquée
# avec la clause WHERE  name NOT IN (...)
sql('''SELECT DISTINCT name
FROM tracks
WHERE name LIKE "B%" AND name NOT IN
    (SELECT DISTINCT name
    FROM tracks WHERE name LIKE "B%"
    ORDER BY name
    LIMIT 10)
ORDER BY name
LIMIT 10''')
```

[71]:
```
[('Back off Bitch',),
 ('Back to Black',),
 ('Bad',),
 ('Bad Apples',),
 ('Bad Attitude',),
 ('Bad Boy',),
 ('Bad Boy Boogie',),
 ('Bad Moon Rising',),
 ('Bad Obsession',),
 ('Bad Seed',)]
```

2.1 Entrainement

3 Niveau I

[ ]:
```
# Donner la liste des noms de playlists
```

[ ]:
```
# Donner le nombre de genres de musiques de noms différents
```

[ ]:
```
# Donner la liste des fonctions (title) des employees
```

[ ]:
```
# Donner, par ordre alphabétique, la liste des Pays dont viennent les clients,
```

Niveau II

[ ]:
```
# Donner le nom des artistes commençant par la lettre R triée par ordre␣
↪alphabétique
```

[ ]:
```
## Donner le nom de musiques en face du nom de chaque playlists
```

[ ]:
```
# Donnner le nombre de musiques dont la durée dépasse 15 min
```

[ ]:
```
# Donner la liste des noms de compositeurs de la playlist "Movies"
# dont le nom contient le groupe de lettres "ichael", la liste
# devant être triée par ordre alphabétique
```

[124]:
```
# lister tous les genres musicaux différents contenus dans la playlist appelée␣
↪"Classical"
```

[124]:
```
[('Classical',), ('Opera',), ('Soundtrack',)]
```

[ ]:
```
# Lister toutes les noms de musiques du genre "Reggae" précédées du nom de␣
↪l'album
# Les résultats doivent être triés par ordre alphabétique d'album
```

4 Niveau III

[ ]:
```
# Combien y a-t-il d'artistes différents dans la playlist "Brazilian Music"
```

[ ]:
```
# Donner, pour chaque playlist sa durée totale en heures
```

[ ]:
```
# Donner le nombre de musiques de chaque genres parmi les musiques de la table␣
↪tracks
# On triera les données par ordre alphabétique de genre
```

[ ]:
```
# Donner, pour chaque genre musical, le nombre d'artistes parmi les musiques de␣
↪la table tracks
```

[ ]:
```
## Donner les noms des cinq morceaux les plus longs en durée de la playlist␣
↪'Music'
```

[ ]:
```
# Quel est, pour chaque playlist, la durée du morceau le plus long?
```

5 Niveau IV

[58]:
```
## Donner l'artiste le plus représenté dans chaque genre musical
## parmi les musiques de la liste tracks
```

[7]:
```
# Quel est, pour chaque playlist, le nom du morceau le plus long et sa durée en␣
↪minutes
```

[7]:
```
[('90's Music', 'Dazed And Confused', 18),
 ('Brazilian Music', 'Vai Passar', 6),
 ('Classical', 'Adagio for Strings from the String Quartet, Op. 11', 9),
 ('Classical 101 - Deep Cuts',
  'Symphony No. 3 Op. 36 for Orchestra and Soprano "Symfonia Piesni Zalosnych"
\\ Lento E Largo - Tranquillissimo',
  9),
 ('Classical 101 - Next Steps',
  "Symphonie Fantastique, Op. 14: V. Songe d'une nuit du sabbat",
  9),
 ('Classical 101 - The Basics',
  'Adagio for Strings from the String Quartet, Op. 11',
  9),
 ('Grunge', 'Alive', 5),
 ('Heavy Metal Classic', 'Master Of Puppets', 8),
 ('Music', 'Dazed And Confused', 26),
 ('Music', 'Dazed And Confused', 26),
 ('Music Videos', 'Band Members Discuss Tracks from "Revelations"', 4),
 ('On-The-Go 1', "Now's The Time", 3),
 ('TV Shows', 'Occupation / Precipice', 88),
```

```
 ('TV Shows', 'Occupation / Precipice', 88)]
```

[ ]:
```
## Donner le compositeur le plus présent pour chaque playlist,
## en cas d'égalité on souhaite que ce soit le compositeur dont le
## est le premier dans l'ordre alphabétique qui soit utilisé
```

[13]:
```
## Question subsidiare: afficher en plus la fréquence
## maxi de ce compositeur
# on fait une table avec la fréquence maxi du composer
```

[13]:
```
[('90's Music', 'U2', 34),
 ('Brazilian Music', 'Djavan', 3),
 ('Classical', 'Johann Sebastian Bach', 7),
 ('Classical 101 - Deep Cuts', 'Henry Purcell', 2),
 ('Classical 101 - Next Steps', 'Johann Sebastian Bach', 2),
 ('Classical 101 - The Basics', 'Johann Sebastian Bach', 3),
 ('Grunge', 'Kurt Cobain', 6),
 ('Heavy Metal Classic', 'Steve Harris', 4),
 ('Music', 'Steve Harris', 160),
 ('On-The-Go 1', 'Miles Davis', 1)]
```

[ ]: