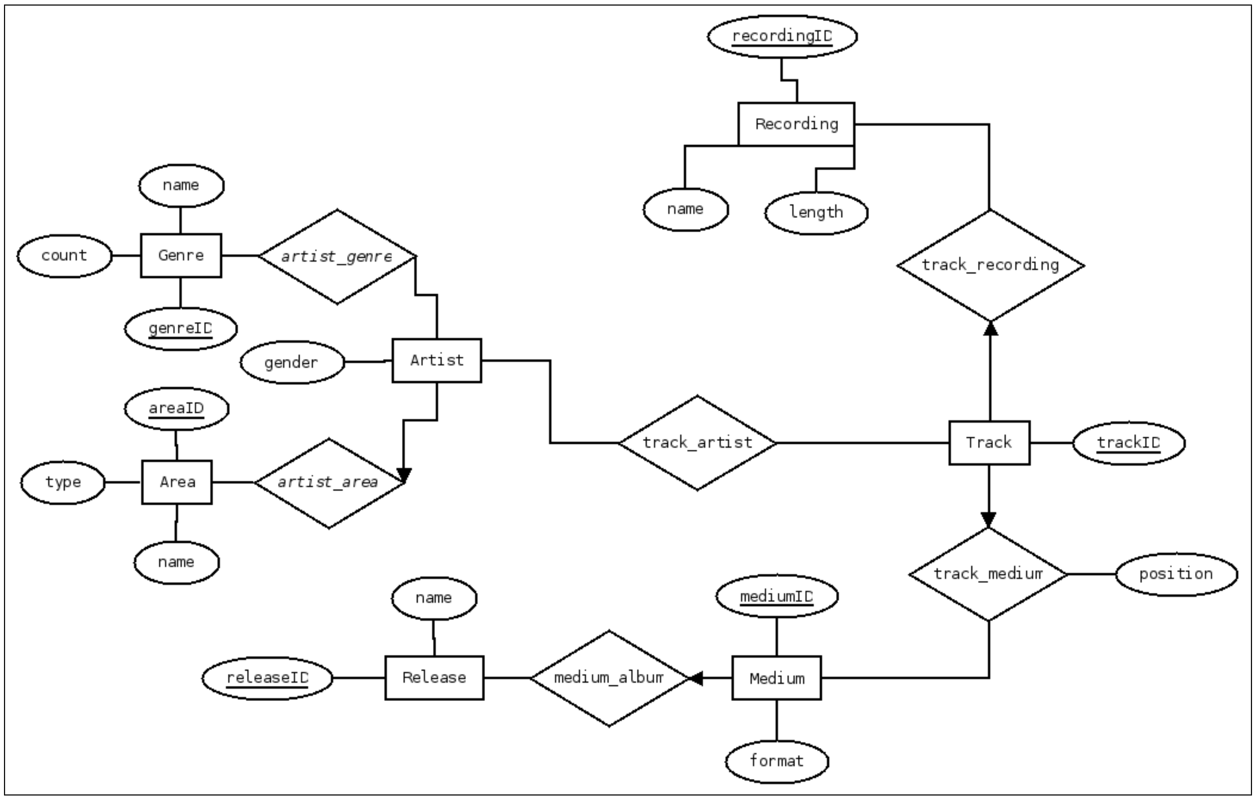
CS-322 Introduction to Database Systems
Project Deliverable #3

Due on Tuesday, June 3^{rd}, 2014

Group 24
Klay, Ameho, Vinh Mau

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

ER model for music database



SQL DDL code for table creations

```
CREATE TABLE Areas (
  areaID INTEGER,
  name VARCHAR(4000) NOT NULL,
  type VARCHAR(255),
  PRIMARY KEY (areaID) ) ;

CREATE TABLE Genres (
  genreID INTEGER,
  name VARCHAR(4000) NOT NULL,
  count INTEGER DEFAULT 0,
  PRIMARY KEY (genreID) ) ;

CREATE TABLE Artists (
  artistID INTEGER,
  name VARCHAR(4000) NOT NULL,
  type VARCHAR(255),
  gender VARCHAR(255),
  areaID INTEGER,
  PRIMARY KEY (artistID),
  FOREIGN KEY (areaID) REFERENCES Areas ) ;


CREATE TABLE Recordings (
```

```sql
  recordingID INTEGER,
  name VARCHAR(4000) ,
  length INTEGER,
  PRIMARY KEY (recordingID) ) ;



CREATE TABLE Releases (
  releaseID INTEGER,
  name VARCHAR(4000) NOT NULL,
  PRIMARY KEY (releaseID) ) ;

CREATE TABLE Mediums (
  mediumID INTEGER,
  releaseID INTEGER,
  format VARCHAR(255),
  PRIMARY KEY (mediumID),
  FOREIGN KEY (releaseID) REFERENCES Releases ) ;


CREATE TABLE Tracks (
  trackID INTEGER,
  recordingID INTEGER,
  mediumID INTEGER,
  position INTEGER,
  PRIMARY KEY (trackID),
  FOREIGN KEY (mediumID) REFERENCES Mediums,
  FOREIGN KEY (recordingID) REFERENCES Recordings ) ;
```

SQL script for entities table creation

```sql
CREATE TABLE Artist_genre (
  artistID INTEGER,
  genreID INTEGER,
  PRIMARY KEY (artistID, genreID),
  FOREIGN KEY (artistID) REFERENCES Artists,
  FOREIGN KEY (genreID) REFERENCES Genres ) ;


CREATE TABLE Track_artist (
  artistID INTEGER,
  trackID INTEGER,
  PRIMARY KEY (artistID, trackID),
  FOREIGN KEY (trackID) REFERENCES Tracks ,
  FOREIGN KEY (artistID) REFERENCES Artists ) ;
```

SQL script for relations table creation

Design choices & data constraints

There are three main concepts in our music database : **Song**, **Artist** and **Album**. Both Song and Album were divided between their descriptive data (**Recording, Release**) and their physical incarnation (**Track, Medium**). Since data is often incomplete, most of the entities 'can be related' but do not have to. We put a NOT NULL constraint on most of the name attributes of the entities, with the exception of **Recording** for the reason just stated. Since they are not required fields to describe music, they should have a valid name when they are in fact used.

- A **Track** is related to:

  Recording: A track can be a physical incarnation of a known recording.

  Artist: A track can exist without known artists, but can also have several artists to describe collaborations.

  Medium: A track can be recorded on some medium. Their relation is characterized by the track position on the medium.

- An **Artist** is defined by a:

  Genre: A genre can regroup multiple artists, whereas an artist can be difficult to define as catering to a specific genre, or crossing boundaries between genres nullifying the need for a constraint. We kept the count attribute, choosing small update costs over on-demand higher computation costs.

  Area: An artist's location can be pinpointed to a specific creation grounds, hence can be expressed by a foreign key constraint. But several artists can be compelled to share their musical feelings in the same studio.

- A **Release** is the logical aggregation of songs, labeled by a title, and can be recorded on multiple mediums. Conversely, a medium identifies a singular recording of an album, enforced by a foreign key constraint.

The integrity of the count attribute in **Genre**, are not guaranteed by the table creation. It will later be enforced later on by the import and delete data commands.

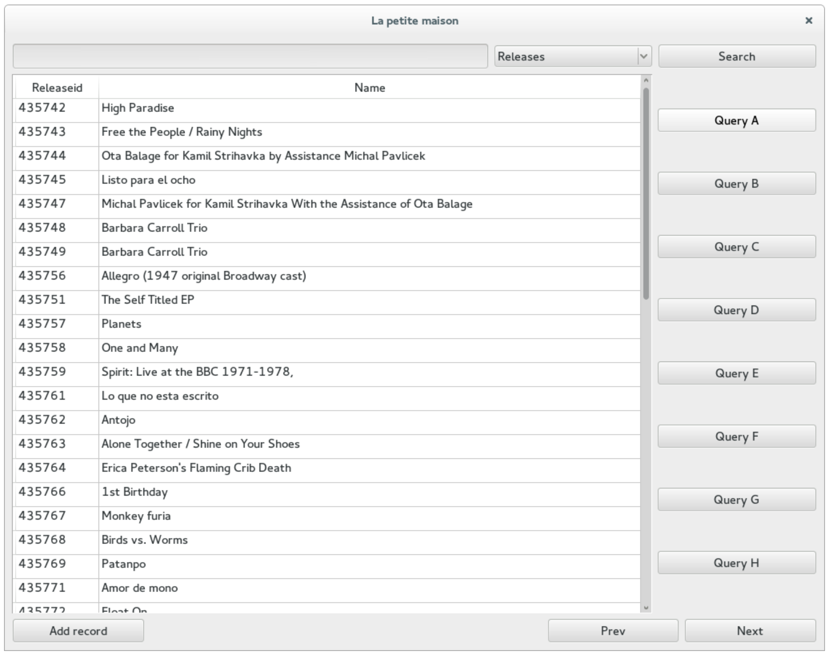
Design changes from deliverable 1

We removed all of our 'at least one' constraints to facilitate the import of new data. For the same reason, we also changed our model to be closer to the given data, so that one can easily add any information into the database and not just track-related information like we initially had in mind. That is, in our first model, everything had to be related to a track to be relevant but now every entities are independent.

Data import

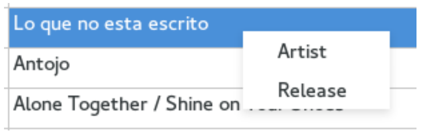
After having quite a lot of trouble trying to import the data on the oracle server, we decided to use a local database instead. We used SQLite to handle this database.

Interface



Releaseid	Name
435742	High Paradise
435743	Free the People / Rainy Nights
435744	Ota Balage for Kamil Strihavka by Assistance Michal Pavlicek
435745	Listo para el ocho
435747	Michal Pavlicek for Kamil Strihavka With the Assistance of Ota Balage
435748	Barbara Carroll Trio
435749	Barbara Carroll Trio
435756	Allegro (1947 original Broadway cast)
435751	The Self Titled EP
435757	Planets
435758	One and Many
435759	Spirit: Live at the BBC 1971-1978,
435761	Lo que no esta escrito
435762	Antojo
435763	Alone Together / Shine on Your Shoes
435764	Erica Peterson's Flaming Crib Death
435766	1st Birthday
435767	Monkey furia
435768	Birds vs. Worms
435769	Patanpo
435771	Amor de mono
435772	Float On

We chose to use Python to design the software to harness the powerful SQLAlchemy API to interface with the database. Therefore we used PyQt5 to design the GUI itself providing clean and practical tools. SQLAlchemy offers a very useful model as it reflects accurately the actual database schema and makes it easier to map the data to the Model/View offered by Qt. Most of the window is taken over by the table representation presenting the data. It offers contextual research such as getting all records for an artist or for a release.



At the top of the window is a search field. The entry is searched in the field selected in the ComboBox. On the right is a list of buttons to perform the queries requested in the deliverable. The last function is the import of data in the database which is handled quite straightforwardly by a dialog which presents the fields to fill in and generates dynamically the INSERT statement and the needed dynamic parameters, namely the id and the updated count per genre value.

SQL Queries

```sql
-- Print the names of artists from Switzerland, i.e.,
-- artists whose area is Switzerland.
-- You should not include the names of the artists associated
-- with individual cantons and towns in Switzerland.
SELECT    arti.name
FROM      artists arti, areas area
WHERE     arti.areaID=area.areaID AND area.name='Switzerland' ;
```

SQL script for query A

```sql
-- Print the names of areas with the highest number male artists,
-- female artists and groups.
-- For each of these 3 areas, print the number of artists of
-- each of the three types in the area.

-- Area with the most male Artists
Select  artistarea.areaname, artistarea.Type, count(*) "number" from
     (Select * from Artists arti INNER JOIN (
     SELECT    Area.name areaname, area.areaId
     FROM      Areas area
     WHERE     area.areaid = (
                         SELECT    AreaId areafemale
                         FROM (
                              SELECT    AreaId , count(*) c
                              FROM      Artists
                              WHERE     (gender = 'Male')
                              GROUP BY  AreaId
                              ORDER BY  c DESC
                         )
                         WHERE     ROWNUM <=1
     )
     ) toparea
     ON arti.areaid = toparea.areaid) artistarea
GROUP BY artistarea.Type , artistarea.areaname
;
-- Area with the most female  Artists
Select  artistarea.areaname, artistarea.Type, count(*) "number" from
     (Select * from Artists arti INNER JOIN (
     SELECT    Area.name areaname, area.areaId
     FROM      Areas area
     WHERE     area.areaid = (
                         SELECT    AreaId areafemale
                         FROM (
                              SELECT    AreaId , count(*) c
                              FROM      Artists
                              WHERE     (gender = 'Female')
                              GROUP BY  AreaId
                              ORDER BY  c DESC
                         )
                         WHERE     ROWNUM <=1
     )
     ) toparea
     ON arti.areaid = toparea.areaid) artistarea
GROUP BY artistarea.Type , artistarea.areaname
;
-- Area with the most female Groups
Select  artistarea.areaname, artistarea.Type, count(*) "number" from
     (Select * from Artists arti INNER JOIN (
     SELECT    Area.name areaname, area.areaId
     FROM      Areas area
     WHERE     area.areaid = (
                         SELECT    AreaId areafemale
                         FROM (
                                   SELECT    AreaId , count(*) c
                                   FROM      Artists
                                   WHERE     (gender = 'Female') and (type = 'Group')
                                   GROUP BY  AreaId
                                   ORDER BY  c DESC
                              )
                              WHERE     ROWNUM <=1
      )
      ) toparea
      ON arti.areaid = toparea.areaid) artistarea
GROUP BY artistarea.Type , artistarea.areaname
;
```

SQL script for query B

```
-- List the names of 10 groups with the most recorded tracks.
SELECT    *
FROM (
      SELECT    Name
      FROM      Artists arti
      INNER JOIN (
                  SELECT    ArtistId
                  FROM (
                              SELECT ArtistId   , count(*) numb
                              FROM      TRACK_ARTIST
                              GROUP BY  ArtistId
                              ORDER BY  numb DESC )
            ) artiId
      ON        arti.ArtistId = artiId.ArtistId
      WHERE     arti.Type = "GROUP"
)
WHERE     ROWNUM <=10  ;
```

SQL script for query C

```
-- List the names of 10 groups with the most releases.
SELECT *
FROM
 (
SELECT    arti.name
FROM      Artists arti
INNER JOIN (
            SELECT         ArtistId, COUNT(DISTINCT ReleaseID) num
            FROM           Track_Artist trackarti
            INNER JOIN (
                        SELECT    *
                        FROM      Tracks track
                        INNER JOIN (
                                    SELECT         mediums.MediumId, Mediums.AlbumID
                                    FROM           Mediums mediums
                                    INNER JOIN     Albums albums
                                    ON             mediums.AlbumID = albums.AlbumID
                                ) media
```

```
                    ON    track.MediumID = media.Medium
                ) track
          ON        trackarti.TrackID = track.TrackID
          GROUP BY  ArtistID
          ORDER BY  num DESC
      ) artiId
ON        arti.ArtistId = artiId.ArtistId
WHERE     arti.Type = "Group"
)
WHERE     ROWNUM <=10  ;
```

SQL script for query D

```
-- Print the name of a female artist associated with the most genres.
SELECT    name
FROM      Artists arti
INNER JOIN (
          SELECT         arti.ArtistID, COUNT(DISTINCT genre.GenreID) numb
          FROM           Artists arti
          INNER JOIN     Artist_Genre genre
          ON             arti.ArtistID = genre.ArtistID
          WHERE          arti.Gender = 'Female'
          GROUP BY       arti.ArtistId
          ORDER BY       numb DESC
     ) artigenre
ON        arti.artistid = artigenre.artistid
WHERE     ROWNUM <=1  ;
```

SQL script for query E

```
-- List all cities which have more female than male artists.
SELECT     areamalefemale.topname
FROM (
     SELECT         city."NAME" topname , city.AreaId,
                    count(CASE WHEN arti.gender = 'Female' THEN 1 END) AS females,
                    count(CASE WHEN arti.gender = 'Male' THEN 1 END) AS males
     FROM           Artists arti
     INNER JOIN     Areas city
     ON             city.areaid = arti.areaId
     WHERE          city.type = 'City'
     GROUP BY       city.areaId , city."NAME"
) areamalefemale
WHERE     areamalefemale.Females > areamalefemale.Males
```

SQL script for query F

```
-- List the mediums with the highest number of tracks.
SELECT    track.mediumid
FROM      Tracks track
GROUP BY  track.mediumid
HAVING    COUNT (*) >= ALL (
                          SELECT     COUNT(*)
```

```
                                FROM      Tracks track
                                GROUP BY  track.mediumid
                      )
```

SQL script for query G

```
-- For each area that has more than 30 artists, list the male artist,
-- the female artist and the group with the most tracks recorded.

SELECT art.artistId, MAX ( trackCount )
FROM (
     SELECT artistId, COUNT(*) trackCount
```

SQL script for query H

```
```

SQL script for query I

```
-- For each of the 10 genres with the most artists, list the most popular female artist. Most popula
Select ArtistId, "NAME", AreaId Gender ,"TYPE", GenreId
  from (
  -- Seqnum is used to keep only ONE artist per genre
  Select r.* , row_number() over (partition by genreId order by artistId) as seqnum
  from (
  -- Max counter is used to keep the artists with the most tracks for every genreId
    Select t.* , max(counter) over (partition by genreId) as maxcounter
    from (
    -- Return , for each selected genre, all artists with their number of recorded tracks ("counter"
         Select artilist.ArtistId, artilist."NAME", artilist.AreaId, artilist.gender, artilist."TYPE'
         from Track_Artist
         INNER JOIN (
          Select Artists.*  , genreid
          from Artists
          INNER JOIN (
                Select ArtistId, genreids.genreid
                from Artist_Genre
                INNER JOIN (
                     Select GenreId
                     from (
                          Select *
                          From (
                            Select GenreId, count(*) counter
                            from Artist_GENRE
                            GROUP BY GenreId
                            ORDER BY counter DESC  )
                          WHERE      ROWNUM <=10)
                ) genreids
                ON genreids.genreid = Artist_Genre.genreid ) artigenre
          ON artigenre.artistId = Artists.ArtistId
          where Artists.gender = 'Female' ) artilist
         ON Track_Artist.artistid = artilist.artistid
```

```
        GROUP BY  artilist.ArtistId, artilist."NAME", artilist.AreaId, artilist.gender, artilist."TY
        ORDER BY genreid, counter DESC ) t
    ) r
    where counter = maxcounter
  )
where seqnum = 1
```

SQL script for query J

```
-- List all genre with no female artist, all genre that have no males artists and all genres that hav
--- <=> List all genre with no female artist OR no male artists OR no groups (?)


SELECT *
FROM Genres g
WHERE g.genreId NOT IN (
     SELECT GenreId FROM (
          SELECT  GenreId,
               count(CASE WHEN arty.gender = 'Female' THEN 1 END) AS females,
               count(CASE WHEN arty.gender = 'Male' THEN 1 END) AS males,
                  count(CASE WHEN arty.type = 'Group' THEN 1 END) AS grps
          FROM Artist_Genre
          INNER JOIN Artists arty
          ON  Artist_Genre.ArtistId = arty.ArtistId
          GROUP BY GenreId )
     WHERE males > 0 AND females > 0 AND grps > 0)
```

SQL script for query K

```
--For each area with more than 10 groups, list the 5 male artists that have recorded the highest numb

Select ArtistId, "NAME", gender, "TYPE", areaId
From (
  Select r.* ,  row_number() over (partition by areaId order by counter) as seqnum
  From (
    -- Count the number of tracks per areaId/Artists entry
    Select t.artistId , t."NAME", t.gender , t."TYPE", t.areaId, count(*) counter
    From Track_Artist tr
    INNER JOIN (
            Select *
            From Artists
            where AreaId in (
               -- Return the Areas with the more than 10 Artists
              Select AreaId
              From (
                          Select AreaId, count(*) counter
                          from Artists
                          where "TYPE" = 'Group'
                         GROUP BY AreaId
                         ORDER BY counter DESC

                          )
              where NOT  areaId IS NULL AND  counter > 0
            )
    )t
    ON tr.artistid = t.artistid
    Group by t.artistId , t."NAME", t.gender , t."TYPE", t.areaId
    ORDER BY t.areaId, counter DESC
    ) r
)
where seqnum <= 5
```

SQL script for query L

```
-- Select the 10 groups with the highest number of tracks that appears in a compilation
Select art.*
FROM Artists art
INNER JOIN  (
  Select ArtistId, count(*) counter
  From Track_Artist tracky
  INNER JOIN (
    -- Select all the tracks that appears in a compilation
    Select tr.trackId
    From Tracks tr
    INNER JOIN (
    -- Select all the mediumid with at least one collaboration (compilations)
      Select DISTINCT tr.mediumId mediId
      From Tracks tr
      INNER JOIN (
        -- Select all the collaboration (TrackId with at least two artists)
        Select TrackId
        From(
          SELECT TrackId, count(*) artistnumber
          FROM Track_Artist
          GROUP BY TrackId)
        where artistnumber > 1) trid
      ON trid.trackId = tr.trackId) medi
    ON medi.mediId = tr.mediumId) compilId
  ON tracky.trackId = compilId.trackId
  Group By ArtistId
  ORDER BY counter DESC) arti
ON art.artistId = arti.artistId
WHERE ROWNUM <=10 AND art."TYPE" = 'Group'
```

SQL script for query M

```
-- List the top 10 releases with the most collaborations, i.e., releases where one artist is perform
-- songs and the highest number of different guest artists contribute to the album.

Select *
      FROM (
       -- Filter the compilation (where an artist is credited for ALL the tracks of the release), the
       Select DISTINCT ReleaseId, guestsnumbers
      FROM (
             Select DISTINCT ReleaseId, tra.trackId, ArtistId , trackperRelease , tracksperArtist, g
             From  (
              -- Used to Return the number of distincts tracks per release, useful to determine if a
              Select ReleaseId, TrackId, COUNT(DISTINCT trackId) OVER ( PARTITION BY mediums.release
) trackperRelease
              From Tracks , Mediums
               where tracks.mediumId = mediums.mediumId) trrl ,
             -- Used to return, for each tuple release/artist the number of tracks in wich this arti
             -- Return also the number of differents artists credited in each release
             (Select DISTINCT med.releaseId rlId , tr.trackId , trart.artistId ,
             COUNT(DISTINCT tr.trackId) OVER ( PARTITION BY med.releaseId, trart.artistId
) trackspeArtist ,
             COUNT(DISTINCT trart.artistId) OVER ( PARTITION BY med.releaseId) guestsnumbers
             from mediums med, releases rel , tracks tr , Track_Artist trart
             where med.releaseId = rel.releaseId AND med.mediumId = tr.mediumId AND  tr.trackId = tr
             ORDER by med.releaseId, trart.artistId DESC ) tra

             where trrl.releaseId = tra.rlId
             )
       Where trackspeArtist = trackperRelease
       ORDER BY guestsnumbers DESC )
where ROWNUM <= 10
```

SQL script for query N

```
--List the release which is associated with the most mediums. If there are more than one such release
-- Problem with max in oracle, need to be combined with a GROUP BY
Select ReleaseId
From (
     Select ReleaseId, dense_rank() over (order by medperrel desc) r
     From (
            Select DISTINCT ReleaseId, COUNT(DISTINCT MediumId) OVER ( PARTITION BY releaseId
) medperrel
            From Mediums
            Order By medperrel DESC
     )tr )
where r = 1
```

SQL script for query O

```
-- List the most popular genre among the groups wich are associated with at least 3 genres
Select Genres.genreId, Genres.name
from Genres
     Inner Join (
          Select GenreId, count(*) genrecounter
          from Artist_Genre artigenre
          Inner Join (
               Select ArtistId
               From (
                    Select arti.ArtistId, count(*) numbgenre
                    from Artist_genre
                    INNER JOIN (
                         Select artistId
                         from Artists
                         where type = 'Group') arti
                    on Artist_genre.artistId = arti.artistId
                    GROUP BY arti.ArtistId) artistgenrecount
               where numbgenre > 2 ) artithree
          ON artithree.artistId = artigenre.artistId
          Group By GenreId
          Order by genrecounter Desc) genreordered
     On genreordered.genreid = Genres.genreId
WHERE     ROWNUM <=1  ;
```

SQL script for query P

```
-- List the 5 titles that are associated with the most different songs (recordings) along with the nu

Select *
From (
     Select DISTINCT "NAME" , COUNT(DISTINCT RecordingId) OVER ( PARTITION BY "NAME"  ) numberofsongs
     From Recordings
     ORDER BY numberofsongs DESC
)
where ROWNUM <= 5
```

SQL script for query Q

```
-- List the top 10 artists according to their track-to-release ratio. This ratio is computed by divi


Select ArtistId
From (
       Select ArtistId , tracknumber/releasenumber ratio
       FROM (
         Select DISTINCT ArtistId , COUNT(DISTINCT tra.TrackId) OVER ( PARTITION BY ArtistId
) tracknumber ,
            COUNT(DISTINCT rel.releaseId) OVER ( PARTITION BY ArtistId  ) releasenumber
         From Track_Artist tra , Tracks tr,  Mediums med, Releases rel
         where tra.trackId = tr.trackId AND tr.mediumId = med.mediumId AND rel.releaseId = med.releas
         )
       ORDER BY ratio DESC
)
where ROWNUM <= 10
```

SQL script for query R

```
--List the release which is associated with the most mediums. If there are more than one such release
-- Problem with max in oracle, need to be combined with a GROUP BY



  Select ArtistId, totalrelease/numberoftopsong hitability
  From (
    Select ArtistId, SUM(ReleaseperSong) totalrelease, count(*) numberoftopsong
    from (
      -- now, filter all the song that aren't in the "TOP10" of an "hit artist", in case of tie (ie,
      Select ArtistId,RecordingId,ReleaseperSong , songrank
      FROM (
        -- Rank the top song per artist, and filter all the non "hit artist"
        Select  ArtistId,RecordingId,ReleaseperSong ,rank() over (PARTITION BY ArtistId order by
ReleaseperSong desc) songrank
        From(
          Select DISTINCT ArtistId,RecordingId,ReleaseperSong ,
          COUNT(DISTINCT RecordingId) OVER ( PARTITION BY ArtistId  ) numberoftopsong
          from  (
            -- Return all the song that appears in at least 10 recordings
            Select ArtistId,  tra.trackId ,tr.mediumId,  med.releaseId, tr.recordingId, COUNT(DISTINC
) releasepersong
            From Track_Artist tra , Tracks tr,  Mediums med
            where tra.trackId = tr.trackId AND tr.mediumId = med.mediumId
            )
          where releasepersong >= 2
        )
        where numberoftopsong >= 1
      )
      where songrank <= 2
  ) GROUP BY ArtistId
)
ORDER BY hitability DESC
```

SQL script for query S

Performance analysis

On the necessity of indexes

Run time

Query	Run time (ms)
A	0
B	0
C	0
D	0
E	0
F	0
G	0
H	0
I	0
J	0
K	0
L	0
M	0
N	0
O	0
P	0
Q	0
R	0
S	0