**Assignment 1**

**​​1DV503 Database Technology**

**Luna Alfayoumi**

School of Computer Science, Physics

and Mathematics, Linnaeus University, Sweden

la224ua@student.lnu.se

# **Part 1**

## **Task 1 Music Streaming Service Database (15 points)**

***1.1******Identify all entities and their attributes from the description of database requirements using the following Table template:***

The gray text in the table is just an example,please remove it.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity** | **Attribute** | **Attribute Type** | **Is a Key Attribute?** | **The value type of attributes and constraints (data type, NULL/NOT NULL, Unique)** |
| User | User ID | Simple | Yes | String Unique, not null |
| name | Composite | No | String , not null |
| email | simple | No | String, Unique |
|  | Birthday | Simple | No | Date , number |
|  | Profile picture | Simple | No | String, Not null |
|  | Subscription Type | Multivalued | No | String , Not Null |
|  | Password | simple | No | String , Not Null , Unique |
| Artists | Artist ID | simple | Yes | Unique, number |
|  | Name | Composite | No | String , Not null |
|  | Genre | multivalued | No | String, Not Null |
|  | Artist Image | Simple | No | String , Not null |
| Albums | Albums ID | Simple | Yes | Unique , Not null |
|  | Title | Simple | No | String |
|  | Release Date | Simple | No | Number , Not null |
|  | Cover Image | Simple | No | String , Not null |
| Track | Track ID | Simple | Yes | String , unique , not null |
|  | Track Duration | Simple | No | Number interval |
| Playlist | Playlist ID | Simple | No | Unique , string |
|  | Name | Composite | No | String, Not null |
| Subscription | Subscription ID | Simple | Yes | String, uni que, not null (Bollian) |
|  | Status or type | Maltivaluesd | No | String , not null |
|  | Start date | Simple | No | Date , not null |
|  | End date | Simple | No | Date , not null |

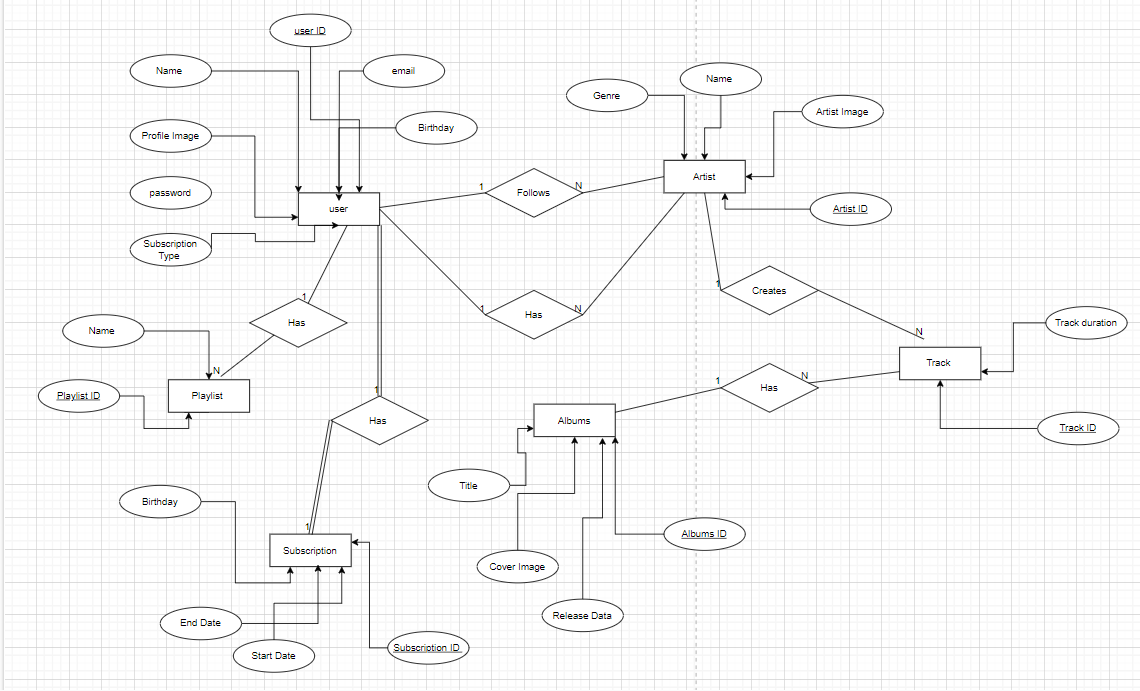
***1.2 Identifying the relationship between entities defined in Task 2.1 using the following table template :***

Entity 1 can join one or more Entity 2, and one Entity 2 can contain one or more of Entity 1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Entity A** | **Relationship name** | **Entity B** | **Cardinality Ration**  **(1:1,1:N,N:1,M:N)** | **Attribute of Relationship**  **(optional)** | **Justify your decision for the cardinality ratio**  **(mandatory)** |
| User | Follows | Artists | 1:N | Date, duration of hours and frequency of listening to that artist | Not all users follow an artist, but every Artist has a user following them |
| user | Has | Playlist  Not mandatory | 1:N | Duration of listening , how many songs added | Not all users have playlist but every playlist has a user |
| user | Listens | album | 1:1 | duration | User can listen to one or more Album, and on album can contain one or more users |
| user | Listens | tracks | 1:N | Duration | Not all users listen to trackes but all trackes have users |
| User | Has | Subscription  double line(mandatory) | 1:1 | Date  birthday | All users have a subscription and all subscription have a user |
| Artist | Has | users | N:1 | How many streames the artist is getting from the users | One artist can have more than one users but users have a minimum of one artist that they listen to |
| Artist | Create | Album | N:M | Date , number of streams | No limit to the creation of albums artist can create albums as much as they want |
| Artist | Create | Track | N:M | Date , number of streams | Tracks can have more than one artist and artist can create more than one album |
| Track | Has | album | N:1 |  | Many tracks can belong to the same singler album , an albums has tat least one track or more included into it |

***1.3 Design an ER schema for the music streaming service database based on information provided in task 1, and entities defined in 1.1 with relationships defined in 1.2.***

The ER schema image here



## **Task 2 University Social Network Database (15 points)**

***2.1******Identify all entities and their attributes from the description of database requirements using the following Table template:***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity** | **Attribute** | **Attribute Type** | **Is a Key Attribute?** | **The value type of attributes and constrains (data type, NULL/NOT NULL, Unique)** |
| Student | Student ID | Simple | Yes | Unique, String, not null |
| Name | Composite | No | String, not null |
| Email | Simple | No | String,not null |
|  | Password | Simple | No | Number, not null |
|  | Birthday | Composite | No | Date, not null |
|  | Hobbies | Multivalued | No | String, not null |
|  | Gender | Simple | No | String , not null |
|  | Study program | Multivalued | No | String , not null |
|  | Join date | simple | No | Date, not null |
|  | Profile image | Simple | No | String, not null |
|  | Username  (= email) | Simple | No | String , not null |
|  | Registration  statues | Simple  (yes or no depends on age) | No | String , not null |
| Friendship | Status | Simple | No | String, not null |
|  | Date | Simple | No | Date , not null |
| Student groups | Group ID | Simple | Yes | Unique, string, not null |
|  | Group subject | Simple | No | String , not null |
|  | name | composite | No | String , not null |
|  | Created date | Simple | No | Date , not null |
|  | Join date | Simple | No | Date , not null |
| Student events | Event ID | Simple | Yes | Unique ,string, not null |
|  | Name | Composite | No | String , not null |
|  | Description | Multivalued | No | String, not null |
|  | Address | Composite | No | String , not null |
|  | Date | Composite  You create Date + Month | No | Date , not null |
|  | Time | Composite  Hour+Minutes | No | String , not null |
| Student post | Post ID | Simple | Yes | Unique, string , not null |
|  | Text | Maltivaued | No | String , not null |
|  | Image | Simple | No | String , Can be Null (optional) |
|  | Location | Simple | No | String , Can be Null (optional) |
|  | Date | Simple | No | Date , Not null |
| comments | Text  Relation to post | Malitivalued | No | String , Not null |

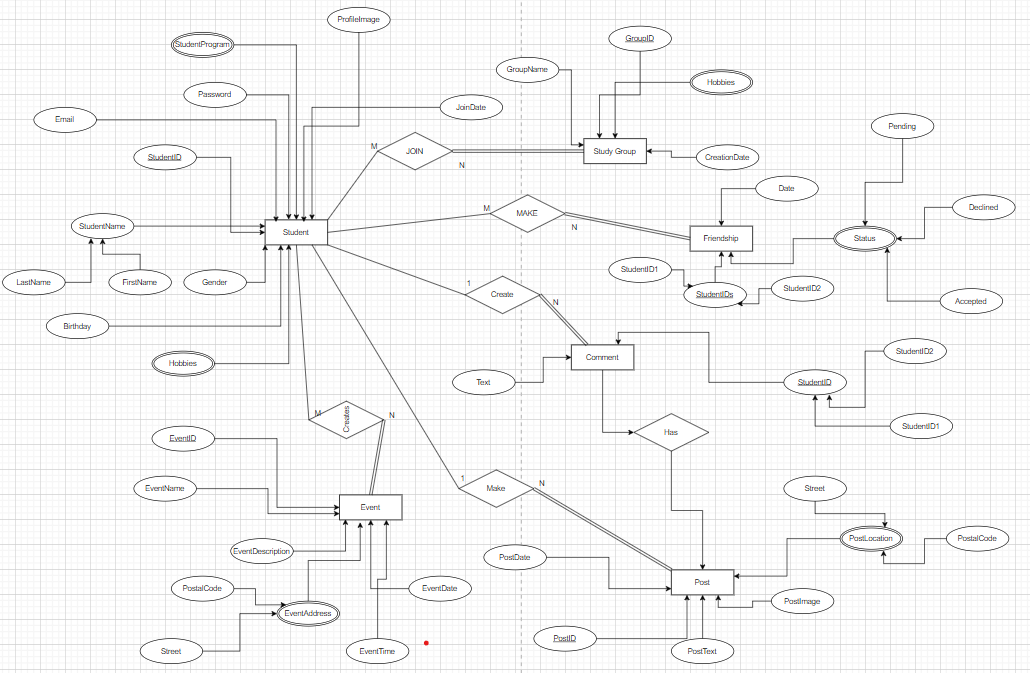
***2.2 Identifying the relationship between entity sets using the following table template:***

Entity 1 can join one or more Entity 2, and one Entity 2 can contain one or more of Entity for N:M

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Entity A** | **Relationship name** | **Entity B** | **Cardinality Ration**  **(1:1,1:N,N:1,M:N)** | **Attribute of Relationship**  **(optional)** | **Justify your decision for the cardinality ratio**  **(mandatory)** |
| Student | Make | Friendship | M:N | Date  Binary relationships between 2 student each with a student ID , hence creating a unique composite attribute to the friendship | Many students can create friendships with many people and can be part of many friendships , that they create or join there is no limit to the friendships student will create |
| Student | Create | Study groups | M:N |  | Student can create many groups and join many study groups at the same time |
| Student | create | events | M:N |  | Many students can create events and events have many students contributing to them |
| students | Make | Student post | 1:N |  | One student can create many student posts or advertisements for such events using those posts , hence its allowed for one student to make many posts |
| Student posts | has | student comments | 1:N | The text they wrote may tell you about there major or hobbies | A singular post can have many comments from different students on it , and student can comment on more than one post |
| student | Create | Comment | 1:N |  | One student can gave many comment but many comments only belong to that particular student |

***2.3 Design an ER schema for the university social network database based on information provided in task 2, and entities defined in 2.1 with relationships defined in 2.2.***

The ER schema image here



## **Task 3 Airport Management Database (10 points)**

***3.1******Given the constraints shown in the ER schema below, respond to the following statements with True, False, or Maybe.***

|  |  |  |  |
| --- | --- | --- | --- |
| ***N*** | ***Statement*** | ***True/False/Maybe*** | ***Justify your answer (mandatory)*** |
| ***1*** | ***Every pilot has been a passenger on at least one flight.*** | ***False*** | Not mandatory relationship, because it is possible to be a pilot without being a passenger on at least one flight , one does not determine the other |
| ***2*** | ***Every flight has at least one deadheading pilot*** | ***False*** | Not mandatory relationship, because it is possible to have a flight without a deadheading pilot |
| ***3*** | ***Every flight has 2 pilots.*** | ***True*** | It is a mandatory relationship, because yes its necessary to have 2 pilots on every flight |
| ***4*** | ***Each flight must have an assigned departure and arrival airport.*** | ***True*** | It is a mandatory relationship, because yes, every flight must have a departure time assigned and in,as at the arrival airport |
| ***5*** | ***Every pilot has flown at least 2 times*** | ***False*** | Could be seen as true logically but,There is no minimum requirements stated for pilots , hence it can be assumed that no not every pilot has flown at least 2 times |
| ***6*** | ***Some tickets that do not belong to any flight*** | ***False*** | It is false because tickets have a mandatory relationship with a flight , that means there is no way there is a ticket that do not belongs to a flight |
| ***7*** | ***Some airlines do not have flights.*** | ***False*** | Airlines have a mandatory relationship with flight , which every airlines has a flight |
| ***8*** | ***A passenger can be a pilot.*** | ***True*** | There is not a mandatory relationship between the passenger with a pilot, it is possible that a passenger can be a pilot but not every passenger is a pilot  They can exit without relying on each other |
| ***9*** | ***Some tickets do not belong to any class type (Economy, Business, etc.)*** | ***False*** | Tickets have a mandatory relationship with a class , every ticket has to belong to a class |
| ***10*** | ***Some tickets are without payment*** | ***True*** | Tickets have a existing mandatory relationships with payment , meaning a ticket den not exit without a payment to purchase it , every ticket requires a payment |

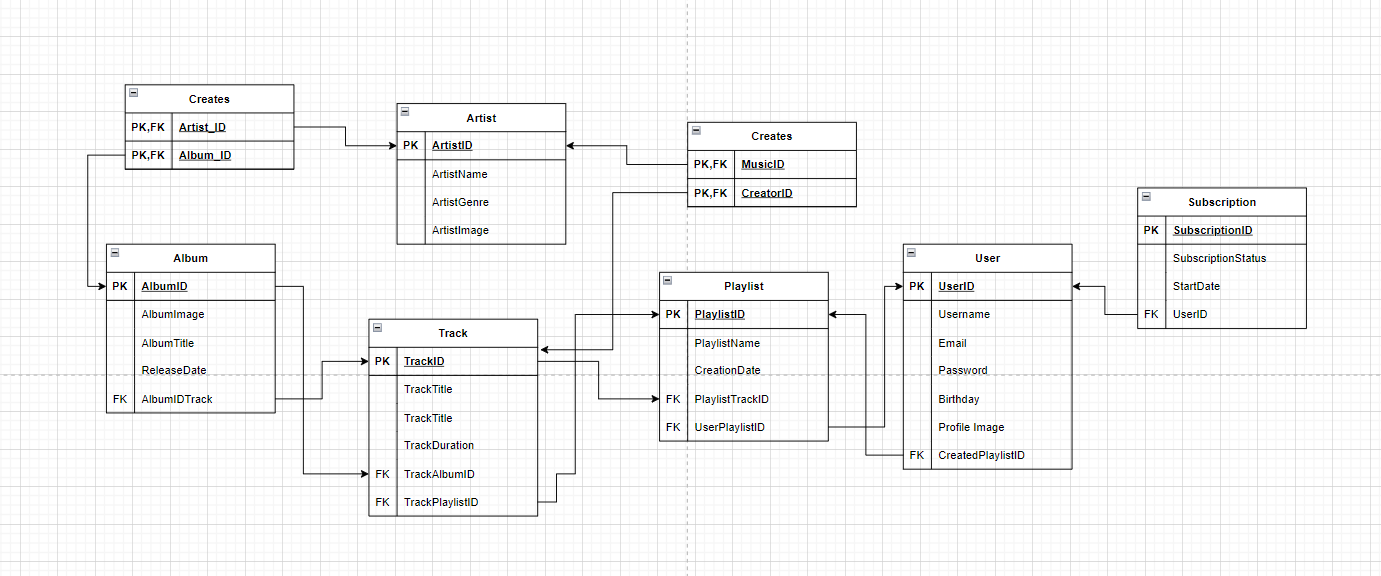
***3.2 List the (min, max) constraint using the total/partial participation for the Ticket entity. Justify your answer.***

|  |  |  |  |
| --- | --- | --- | --- |
| *Entity* | *Relation* | *(min,max)* | *Explanation* |
| *Ticket* | *Has* | *(1,1)* | Because one Ticket can only have one class but one class can have many tickets |
| *Ticket* | *Receives* | *(0,1)* | *The entity Ticket and the relation “receives” will have the minimum 0 and the maximum 1 in the ratio because you can be given a ticket that you did not pay for or someone else has paid for it , but behind every ticket a payment must have been made by someone to purchase the ticket* |
| *Ticket* | *Belongs to* | *(1,N)* | *The entity Ticket and the relation “belongs to” have the minimum of 1 and the max of N , cardinality ratio , because you can have multiple tickets that belong to a single flight, but you need at least one ticket to belong to a flight , it can not be 0* |
| *Ticket* | *Book* | *(1,N)* | *The entity tickets has the relation “Book” with a min of 1 and a max of N , because you can have many tickets per passenger , but for you to be a passenger you must have at least one ticket , can not be 0* |

Part 2

## **Task 4. Convert ER model to Relational Model (15 points)**

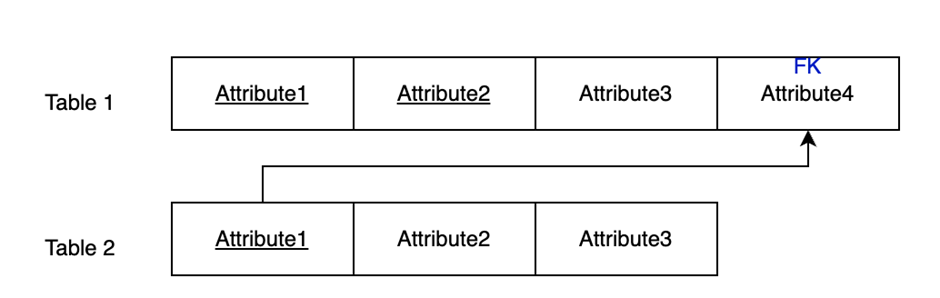
The Relational Model diagram image here



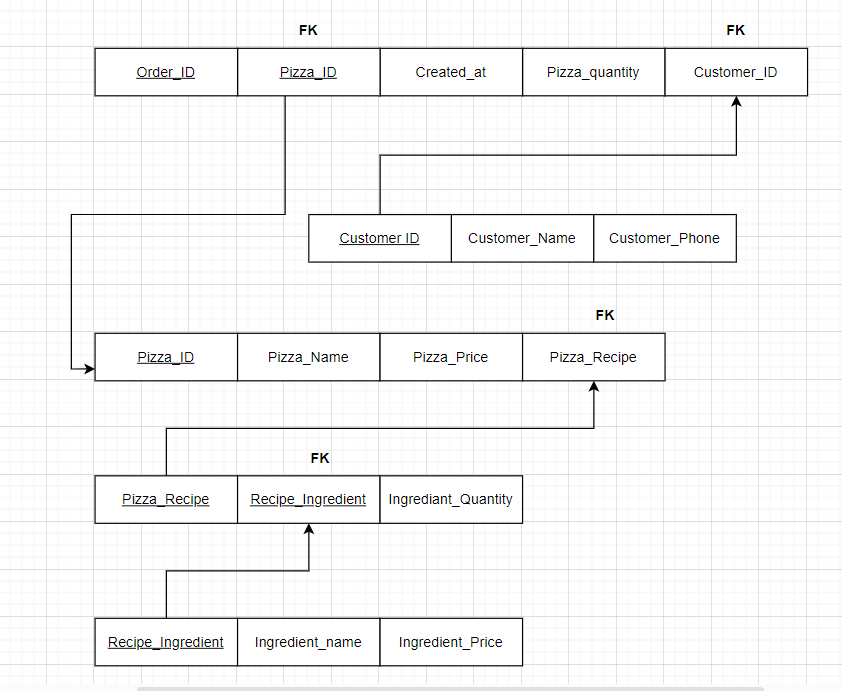
## **Task 5 Normalisation (15 points)**

**5.1** ***Considering unnormalized Pizzeria database with one table is shown below on image bellow, perform normalisation to achieve 3NF for all tables***.

Present the results in a similar format as shown as the example bellow. Make sure that each table has a key, a foreign key (if any), and references from the primary key to the foreign key

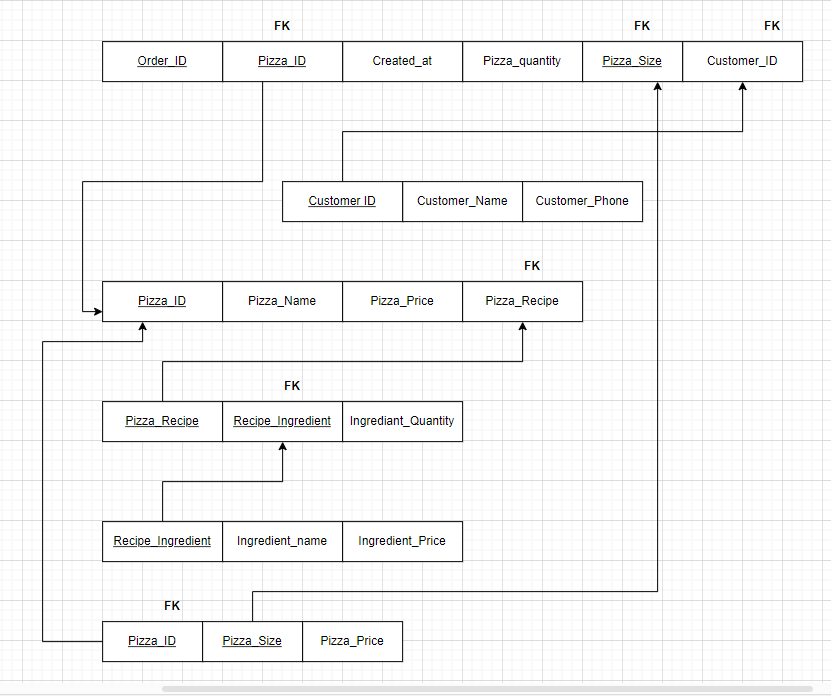


5.1



***5.2 Consider adding pizza size (small, medium, large) attribute to the Order table, and the pizza price will be different for each pizza size. Make sure that changes and new tables satisfy the 3NF.***

Use the same format as above and provide the final solution only for changed (and new ) tables in 3NF.



# **Part 3**

## **Task 6. Basic SQL queries (20 points)**

|  |  |
| --- | --- |
| **Query** | **Answer** |
| 1.1 retrieves all artists whose names start with the same letter as the first letter of your first name. | select Artist.name  FROM Artist  WHERE Artist.name Like 'L%';    the total number of records the query returns : 13 rows |
| 1.2 retrieves all tracks by “BackBeat” artists. | Select Track.Name, Album.AlbumId , Artist.Name  From Track  JOIN Album on Track.AlbumID = Album.AlbumId  Join Artist on Artist.Artistid = Album.Artistid  where Artist.name = 'BackBeat';  3 column , 12 rows |
| 1.3 lists all album titles along with their respective artist name. | Select Album.Title , Artist.Name  FROM Album  JOIN Artist on Album.Artistid = Artist.Artistid;  2 columns , 347 rows |
| 1.4 Lists the total number of tracks in each album, sorted by total in descending order. | Select Album.Title , Count(Track.name)  From Album  JOIN Track on Track.Albumid = Album.AlbumId  GROUP By Album.Title ; --- this seperates the track number under each album title  2 columns , 347 rows |
| 1.5 finds tracks with “Protected AAC audio file” media type | Select Track.name , Mediatype.Name  From Track  Join MediaType On Mediatype.MediaTypeId = Track.MediaTypeId  WHERE MediaType.name = 'Protected AAC audio file';  2 columns, 237 rows |
| 1.6 Lists all tracks from “Big Ones” album | select Track.Name , Album.Title  From Album  Join Track On Album.AlbumId = Track.AlbumId  WHERE Album.Title = 'Big Ones';  15 rows |
| 1.7 finds the total duration of tracks in each playlist. You'll need to use the *Playlist*, *PlaylistTrack*, and *Track* tables for this. | select sum(Track.Milliseconds) , playlist.Name  From Playlist  join PlaylistTrack on Playlist.playlistid = PlaylistTrack.playlistid  Join Track On PlaylistTrack.TrackId = Track.TrackId  Group BY PLaylist.Name;  2 columns , 12 rows |
| 1.8 finds the 10 most expensive tracks. | SELECT Track.name, Track.UnitPrice  From Track  Order BY Track.UnitPrice Desc  Limit 10; --taking in the first 10 values its by default ranked in descending order , hence being the top 10  2 columns , 10 rows |
| 1.9 finds the list of artists who do not have any albums. | SELECT Artist.Name  From Artist  LEft JOIN Album On Album.ArtistId = Artist.ArtistId  WHERE Album.ArtistId IS NULL;  71 rows |
| 1.10 lists all playlists and the number of tracks in each. | SELECT Playlist.name, COUNT(Track.TrackId) As TrackCount  frOM Playlist  Join PlaylistTrack ON PlaylistTrack.playlistId = Playlist.PlaylistId  JOIN Track On PlaylistTrack.TrackId = Track.TrackId  Group By Playlist.PlaylistId;  2 columns , 14 rows |

## 

## 

## **Task 7. Advanced SQL queries (10 points)**

The table was weird with the lay out so placed the for part task 7

-- 2.1

Select Album.Title

From Album

Join Track On Track.AlbumId = Album.AlbumId

Join Genre ON Genre.GenreId = Track.GenreId

GROUP BY Album.AlbumId

Having COUNT(DISTINCT Genre.GenreId) > 1;

**11 rows**

-- 2.2

Select Customer.FirstName, Customer.Lastname, AVG(Invoice.Total) As InvoiceAverage

FRom Customer

JOIN Invoice ON Customer.CustomerId = Invoice.CustomerId

GROUP By Customer.CustomerId

Order BY InvoiceAverage DESC;

**59 rows**

-- 2.3

Select Playlist.Name, SUM(Track.Milliseconds) AS TotalTime

From Playlist

Join PlaylistTrack On PlaylistTrack.PlaylistId = Playlist.PlaylistId

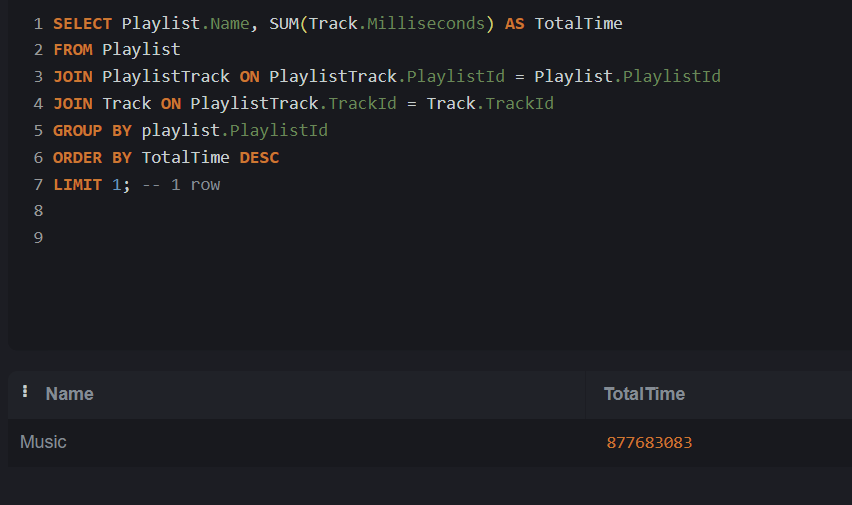
Join Track On PlaylistTrack.TrackId = Track.TrackId

Group BY playlist.PlaylistId

Order By TotalTime DESC

LIMIT 1;

**1 row**



-- 2.4

With TotalPurchasesPerCustomer AS (

Select Customer.CustomerId, Customer.FirstName, Customer.LastName, Sum(InvoiceLine.Quantity) AS totalpurchases, Customer.Country

frOM Customer

Join invoice On Customer.CustomerId = Invoice.CustomerId

jOIN InvoiceLine On Invoice.InvoiceId = InvoiceLine.InvoiceId

GroUP BY customer.CustomerId, Customer.Country

)

SELECT TotalPurchasesPerCustomer.FirstName, TotalPurchasesPerCustomer.LastName, TotalPurchasesPerCustomer.TotalPurchases, TotalPurchasesPerCustomer.Country

From TotalPurchasesPerCustomer

WHERE TotalPurchasesPerCustomer.TotalPurchases = (

SELECT MAX(TotalPurchases)

From TotalPurchasesPerCustomer AS CountryIndividualMaxPurchase

where CountryIndividualMaxPurchase.Country = TotalPurchasesPerCustomer.Country

)

ORDER By TotalPurchasesPerCustomer.Country;

**58 rows**

-- 2.5

With TotalAlbumPerCustomer AS (

Select Album.Title, SUM(InvoiceLine.Quantity) As TotalQuantity, Customer.Country

FRom Album

Join Track On Album.AlbumId = Track.AlbumId

join InvoiceLine On InvoiceLine.TrackId = Track.TrackId

Join Invoice ON Invoice.InvoiceId = InvoiceLine.InvoiceId

Join Customer On Invoice.CustomerId = Customer.CustomerId

Group by Album.Title, Customer.Country

)

SELECT TotalAlbumPerCustomer.Title, TotalAlbumPerCustomer.Country, TotalAlbumPerCustomer.TotalQuantity

FRom TotalAlbumPerCustomer

WHERE TotalAlbumPerCustomer.TotalQuantity = (

SELECT MAX(TotalQuantity)

From TotalAlbumPerCustomer As T2

WHERE T2.Country = TotalAlbumPerCustomer.Country

)

Order By TotalAlbumPerCustomer.TotalQuantity DESC;

**34 rows**

|  |  |
| --- | --- |
| **Query** | **Answer** |
| 2.1 finds albums with multiple genres. | SQL CO |
| 2.2 identifies customers with the highest average invoice value. |  |
| 2.3 Find the Longest Playlist by Duration. |  |
| 2.4 finds customers with Maximum Purchase in Each Country. |  |
| 2.5 find the most popular album purchased by each country. Order the results by number of Sales in descending order. Print the country, album name, and maximum sales |  |

OBS! DO NOT FORGET ALSO TO SUBMIT THE .SQL SCRIPT FILE CONTAINING ALL SQL QUERRIES