```
 create database music_s;
mysql> use music_s;
```

1. Employee table.
a. Create all the columns with the required data types
b. Make “employee_id” as the primary key for this table.

```
      mysql> create table employee(employee_id int primary key,last_name
varchar(30),first_name varchar(30),title varchar(60),reports_to
varchar(20),levels varchar(2),birthdate text,hire_date text,address
text,city varchar(50),state varchar(2),country varchar(20),postal_code
varchar(10),phone varchar(18),fax varchar(20), email varchar(50)) ;
mysql> desc employee;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| employee_id | int         | NO   | PRI | NULL    |       |
| last_name   | varchar(30) | YES  |     | NULL    |       |
| first_name  | varchar(30) | YES  |     | NULL    |       |
| title       | varchar(60) | YES  |     | NULL    |       |
| reports_to  | varchar(20) | YES  |     | NULL    |       |
| levels      | varchar(2)  | YES  |     | NULL    |       |
| birthdate   | text        | YES  |     | NULL    |       |
| hire_date   | text        | YES  |     | NULL    |       |
| address     | text        | YES  |     | NULL    |       |
| city        | varchar(50) | YES  |     | NULL    |       |
| state       | varchar(2)  | YES  |     | NULL    |       |
| country     | varchar(20) | YES  |     | NULL    |       |
| postal_code | varchar(10) | YES  |     | NULL    |       |
| phone       | varchar(18) | YES  |     | NULL    |       |
| fax         | varchar(20) | YES  |     | NULL    |       |
| email       | varchar(50) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
```

2. Customer Table.
a.Create all the columns from the customer table
b.Make “customer_id” as the primary key.
c. Make “support_reo_id” as foreign key which is referencing “employee_id” from the employee table and make sure you are using cascade and not null actions while creating foreign keys.

```
mysql> create table customer_s(customer_id int primary key,first_name
varchar(30) not null,last_name varchar(30),company varchar(50),address
text,city varchar(50),state varchar(20),country varchar(20),postal_code
varchar(10),phone varchar(20),fax varchar(20),email varchar(50),
support_rep_id int not null,constraint cus_fk foreign key
(support_rep_id) references employee(employee_id)on update cascade on
delete cascade);

mysql> desc customer_s;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| customer_id   | int         | NO   | PRI | NULL    |       |
| first_name    | varchar(30) | NO   |     | NULL    |       |
| last_name     | varchar(30) | YES  |     | NULL    |       |
| company       | varchar(50) | YES  |     | NULL    |       |
| address       | text        | YES  |     | NULL    |       |
| city          | varchar(50) | YES  |     | NULL    |       |
| state         | varchar(20) | YES  |     | NULL    |       |
```

```
| country        | varchar(20) | YES  |     | NULL    |       |
| postal_code    | varchar(10) | YES  |     | NULL    |       |
| phone          | varchar(20) | YES  |     | NULL    |       |
| fax            | varchar(20) | YES  |     | NULL    |       |
| email          | varchar(50) | YES  |     | NULL    |       |
| support_rep_id | int         | NO   | MUL | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
```

3. Invoice table.
a.Create all the required columns in the invoice table.
b.Make ‘’invoice_id” as the primary key.
c.Make “customer_id” as foreign key referencing the “customer_Id” from the customer table and make sure you are using cascade and not null actions while creating foreign keys.

```
mysql> create table invoice( invoice_id int primary key, customer_id int
not null,constraint inv_fk foreign key (customer_id) references
customer(customer_id)on update cascade on delete cascade,invoice_date
datetime,billing_address text,billing_city varchar(50),billing_state
varchar(20),billing_country varchar(20),billing_postal varchar(10),total
float);
```

```
mysql> desc invoice;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| invoice_id      | int         | NO   | PRI | NULL    |       |
| customer_id     | int         | NO   | MUL | NULL    |       |
| invoice_date    | datetime    | YES  |     | NULL    |       |
| billing_address | text        | YES  |     | NULL    |       |
| billing_city    | varchar(50) | YES  |     | NULL    |       |
| billing_state   | varchar(20) | YES  |     | NULL    |       |
| billing_country | varchar(20) | YES  |     | NULL    |       |
| billing_postal  | varchar(10) | YES  |     | NULL    |       |
| total           | float       | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
```

8.Media_type table.
a.Create all the required columns.
b.Make “media_type_id” as the primary key.

```
mysql> create table media_type(media_type_id varchar(70) primary key,name
varchar(30));
```

```
mysql> desc media_type;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| media_type_id | varchar(70) | NO   | PRI | NULL    |       |
| name          | varchar(30) | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
```

9. Genre table.
a.Create all the required columns.
b.Make “genre_id” as the primary key.

```
mysql> create table genre (genre_id varchar(70) primary key,name
varchar(30));
```

```
mysql> desc genre;
```

```
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| genre_id | varchar(70) | NO   | PRI | NULL    |       |
| name     | varchar(30) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
```

6. Playlist table.
a.Create all the required columns.
b.Make “playlist_id” as the primary key.

```
mysql> create table playlist(playlist_id int primary key,name
varchar(30));
```

```
mysql> desc playlist;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| playlist_id | int         | NO   | PRI | NULL    |       |
| name        | varchar(30) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
```

11. Artist table.
a.Create all the required columns.
b.Make “artist_id” as the primary key.

```
mysql> create table artist(artist_id int primary key,name text);
```

```
mysql> desc artist;
+-----------+------+------+-----+---------+-------+
| Field     | Type | Null | Key | Default | Extra |
+-----------+------+------+-----+---------+-------+
| artist_id | int  | NO   | PRI | NULL    |       |
| name      | text | YES  |     | NULL    |       |
+-----------+------+------+-----+---------+-------+
```

10. Album table.
a.Create all the required columns.
b.Make “album_id” as the primary key
c.Make “artist_id” as the foreign key referencing the “artist_id” from the artist table and make sure you are using cascade and not null actions while creating foreign keys.

```
mysql> create table album( album_id varchar(70) primary key,title
text,artist_id int not null,constraint alb_art_fk foreign key (artist_id)
references artist(artist_id)on update cascade on delete cascade);
```

```
mysql> desc album;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| album_id  | varchar(70) | NO   | PRI | NULL    |       |
| title     | text        | YES  |     | NULL    |       |
| artist_id | int         | NO   | MUL | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
```

5.Track table.
a.Create all the required columns from the track table.
b.Make “track_id” as the primary key.

c.Make “media_type_id” as the foreign key referencing the “media_type_id” columns from the table “Media_type” and make sure you are using cascade and not null actions while creating foreign keys.
d.Make “genre_id” as foreign key referencing the “genre_id” from the Genre table and make sure you are using cascade and not null actions while creating foreign keys.
e.Make “album_id” as foreign key referencing the “album_id” from the album table and make sure you are using cascade and not null actions while creating foreign keys.

```
mysql> create table track( track_id int primary key,name text,album_id
varchar(70) not null,constraint tra_alb_fk foreign key (album_id)
references album(album_id)on update cascade on delete
cascade,media_type_id varchar(70) not null,constraint tra_med_fk foreign
key (media_type_id)references media_type(media_type_id)on update cascade
on delete cascade, genre_id varchar(70) not null,constraint tra_gen_fk
foreign key (genre_id) references genre(genre_id)on update cascade on
delete cascade,composer text,milliseconds varchar(50),bytes
varchar(50),unit_price varchar(20));
```

```
mysql> desc track;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| track_id      | int         | NO   | PRI | NULL    |       |
| name          | text        | YES  |     | NULL    |       |
| album_id      | varchar(70) | NO   | MUL | NULL    |       |
| media_type_id | varchar(70) | NO   | MUL | NULL    |       |
| genre_id      | varchar(70) | NO   | MUL | NULL    |       |
| composer      | text        | YES  |     | NULL    |       |
| milliseconds  | varchar(50) | YES  |     | NULL    |       |
| bytes         | varchar(50) | YES  |     | NULL    |       |
| unit_price    | varchar(20) | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
```

7. Playlist_track.
a.Create all the required columns.
b.Make “playlist_id” as foreign key referencing the “playlist_id” from the Playlist table and make sure you are using cascade and not null actions while creating foreign keys.
c.Make “track_id” as foreign key referencing the “track_id” from the track table and make sure you are using cascade and not null actions while creating foreign keys.

```
mysql> create table playlist_track( playlist_id  int null null,constraint
ptra_pl_fk foreign key (playlist_id) references playlist(playlist_id)on
update cascade on delete cascade,track_id int not null,constraint
ptra_tra_fk foreign key (track_id)references track(track_id)on update
cascade on delete cascade);
```

```
mysql> desc playlist_track;
+-------------+------+------+-----+---------+-------+
| Field       | Type | Null | Key | Default | Extra |
+-------------+------+------+-----+---------+-------+
| playlist_id | int  | YES  | MUL | NULL    |       |
| track_id    | int  | NO   | MUL | NULL    |       |
+-------------+------+------+-----+---------+-------+
```

4.Invoice_line.
a.Create all the required columns in the invoice line table.

b.Make “invoice_line_id” as the primary key.
c.Make “invoice_id” as the foreign key which is referencing the “invoice_id” from the invoice table and make sure you are using cascade and not null actions while creating foreign keys.
d.Also all foreign key constraints to the “track_id” referencing the “track_id” from the track table.

```
mysql> create table invoice_line( invoice_line_id int primary
key,invoice_id int not null,constraint invl_inv_fk foreign key
(invoice_id)references invoice(invoice_id)on update cascade on delete
cascade,track_id int not null,constraint inl_tra_fk foreign key
(track_id)references track(track_id)on update cascade on delete cascade,
unit_price float, quantity int);
```

```
mysql> desc invoice_line;
+-----------------+-------+------+-----+---------+-------+
| Field           | Type  | Null | Key | Default | Extra |
+-----------------+-------+------+-----+---------+-------+
| invoice_line_id | int   | NO   | PRI | NULL    |       |
| invoice_id      | int   | NO   | MUL | NULL    |       |
| track_id        | int   | NO   | MUL | NULL    |       |
| unit_price      | float | YES  |     | NULL    |       |
| quantity        | int   | YES  |     | NULL    |       |
+-----------------+-------+------+-----+---------+-------+
```

TASK

1. Who is the senior most employee based on job title?

```
select * from employee;
desc employee;
select * from employee where levels = 'L7';
select * from employee order by levels desc limit 1;
```

2. Which countries have the most Invoices?

```
select * from invoice;
select max(billing_country) from invoice;
select billing_country , count(invoice_id) as count_ from invoice
group by billing_country order by count(invoice_id) desc limit 1;
```

3. What are top 3 values of total invoice?

```
desc invoice;
select total from invoice order by total desc limit 3;
```

4. Which city has the best customers? We would like to throw a promotional Music Festival in the city we made the most money.
Write a query that returns one city that has the highest sum of invoice totals. Return both the city name & sum of all invoice totals

```
select billing_city,
sum(total) as total from invoice
group by billing_city
order by total desc limit 1;
```

5. Who is the best customer? The customer who has spent the most money will be declared the best customer. Write a query that returns the person who has spent the most money

```
select * from invoice;
select * from customer;
```

```
select customer_id as customerId,
sum(total) as total_spend
from invoice
group by customer_id
order by total_spend desc
limit 1;
```

6. Write query to return the email, first name, last name, & Genre of all Rock Music listeners.
Return your list ordered alphabetically by email starting with A

```
select * from customer;
select * from invoice;
select * from genre;
select * from track;
select distinct c.email as email, c.first_name as first_name, c.last_name
as last_name
from customer c join invoice i on c.customer_id = i.customer_id
join invoice_line inl on i.invoice_id = inl.invoice_id
join track t on inl.track_id = t.track_id
join genre g on t.genre_id = g.genre_id
where g.name = 'Rock' order by c.email;
```

7. Let's invite the artists who have written the most rock music in our dataset. Write a query that returns the Artist name and total track count of the top 10 rock bands

```
select * from artist;
select * from genre;
select * from album;
select * from track;

select ar.artist_id as artist_id, ar.name as name, count(t.name) as song
from artist ar join album al on al.artist_id = ar.artist_id
join track t on al.album_id = t.album_id
join genre g on t.genre_id = g.genre_id
where g.name = 'Rock' group by ar.artist_id, ar.name, g.name
order by song desc limit 10;
```

8. Return all the track names that have a song length longer than the average song length. Return the Name and Milliseconds for each track. Order by the song length with the longest songs listed first

```
select * from track;

select t.name , t.milliseconds from track t where t.milliseconds >
(select avg(milliseconds) from track) order by t.milliseconds desc;
```

9. Find how much amount spent by each customer on artists? Write a query to return customer name, artist name and total spent

```
select * from customer;
select * from artist;

select a.name as name, sum(il.unit_price) as spent_amount,
sum(il.quantity) as quantity,
c.customer_id as customer_id, c.first_name as first_name, c. last_name as
last_name
from artist a join album al on a.artist_id =al.artist_id
join track t on t.album_id = al.album_id
join invoice_line il on il.track_id = il.track_id
join invoice i on il.invoice_id = i.invoice_id
```

```
join customer c on c.customer_id = i.customer_id
where a.name = 'Iron Maiden' group by customer_id order by spent_amount
desc;
```

10. We want to find out the most popular music Genre for each country. We determine the most popular genre as the genre with the highest amount of purchases. Write a query that returns each country along with the top Genre. For countries where the maximum number of purchases is shared return all Genres

```
with CountryGenPopularityList as
(select count(*) as Popularity, gen.name as GenreName, i.billing_country
as Country
from invoice_line il
     join track trk on trk.track_id=il.track_id
       join genre gen on gen.genre_id=trk.genre_id
       join invoice i on il.invoice_id = i.invoice_id
group by Country, gen.genre_id)

select cgpl.Country, cgpl.GenreName, cgpl.Popularity
from CountryGenPopularityList cgpl
where      cgpl.Popularity = (select   max(Popularity) from
CountryGenPopularityList
                                                  where
cgpl.Country=Country
                                                  group by Country)
order by Country;
```

11. Write a query that determines the customer that has spent the most on music for each country. Write a query that returns the country along with the top customer and how much they spent. For countries where the top amount spent is shared, provide all customers who spent this amount

```
with TotalsPerCountry as
(
select i.billing_country, cust.first_name || ' ' || cust.last_name as
CustomerId,
sum(i.total) as TotalSpent from invoice i
join customer cust on cust.customer_id=i.customer_id
group by i.billing_country, cust.customer_id
order by i.billing_country
)
select a.billing_country, a.CustomerId, a.TotalSpent
from  TotalsPerCountry a
where a.TotalSpent = ( select max(TotalSpent)
from TotalsPerCountry
where a.billing_country=billing_country
group by billing_country);


select * from customer;
select * from invoice;
SELECT customer.customer_id, customer.first_name, customer.last_name,
invoice.billing_country,SUM(invoice.total) AS total_spending
FROM invoice
JOIN customer ON customer.customer_id = invoice.customer_id
GROUP BY 1,2,3,4
ORDER BY total_spending DESC
LIMIT 5;
```