**Write down the SQL sentences that give the following results. If the result is a table, draw it and put the results inside.**

1. SELECT a.title, COUNT(DISTINCT as.id\_song) FROM Album AS a JOIN Album\_Songs AS as ON a.id = as.id\_album GROUP BY a.id

|  |  |
| --- | --- |
| a.title | COUNT(DISTINCT as.id\_song) |
| MIX | 2 |
| ESTOPA | 2 |
| MIX VARIOS | 3 |

1. SELECT s.name, a.name FROM Songs AS s JOIN Artist AS a ON s.artist = a.id ORDER BY a.name ASC

|  |  |
| --- | --- |
| s.name | a.name |
| HIGHWAY TO HELL | AC/DC |
| NUBES | ESTOPA |
| ROPAJA | ESTOPA |
| CALORRO | ESTOPA |
| FLOR DE LOTO | HEROES DEL SILENCIO |
| AZUL | SERGIO DALMA |

1. SELECT FROM a1.name, COUNT(a2.id) FROM Album AS a1 JOIN Album\_Songs AS as ON a1.id = as.id\_album JOIN Songs AS s ON s.id = as.id\_song JOIN Artist AS a2 ON a2.id = s.artist GROUP BY a1.id

|  |  |
| --- | --- |
| a1.name | COUNT(a2.id) |
| MIX | 2 |
| ESTOPA | 1 |
| MIX VARIOS | 3 |

1. DELETE FROM Songs WHERE artist = (SELECT id FROM Artists WHERE name LIKE '%A%E%' OR name LIKE '%E%A%')

|  |  |  |
| --- | --- | --- |
| Songs | | |
| id | name | artist |

This table is empty

1. UPDATE Artist SET birthdate = DATEADD(year, 10, birthdate) WHERE birthdate <= (SELECT MIN(birthdate) FROM Artist)

|  |  |  |  |
| --- | --- | --- | --- |
| Artist | | | |
| id | email | birthdate | name |
| 1 | s@dalma.com | 01/01/1960 | SERGIO DALMA |
| 2 | est@pa.com | 01/02/1955 | ESTOPA |
| 3 | AC@dc.com | 30/04/1950 | AC/DC |
| 4 | HE@es.com | 02/02/1970 | HEROES DEL SILENCIO |

1. CREATE VIEW SongsFromEstopa AS SELECT s.id, s.name FROM Songs AS s JOIN Artists AS a ON a.id = s.artist WHERE a.name LIKE 'ESTOPA'

|  |  |
| --- | --- |
| SongsFromEstopa | |
| s.id | s.name |
| 2 | NUBES |
| 3 | ROPAJA |
| 4 | CALORRO |

**Write down the Java code that you need to represent this database (1 point). You don't need to implement the methods.**

**public** **class** Song **implements** Serializable {

**private static final long** ***serialVerisonUID*** = -64567603420449597L;

**private** Integer id;

**private** String name;

**private** Artist artist;

**private** List<Album> albums;

**public** Song(Integer id, String name, Artist artist, List<Album> albums){

**super**();

**this**.id = id;

**this**.name = name;

**this**.artist = artist;

**this**.albums = albums;

}

**public** Song(Integer id, String name, Artist artist){

**super**();

**this**.id = id;

**this**.name = name;

**this**.artist = artist;

**this**.albums = **new** ArrayList<Album>();

}

**public** Integer getId(){}

**public** **void** setId(Integer id){}

**public** String getName(){}

**public** **void** setName(String name){}

**public** Artist getArtist(){}

**public** **void** setArtist(Artist artist){}

**public** List<Album> getAlbums(){}

**public** **void** setAlbums(List<Album> albums){}

@Override

**public** String toString(){}

@Override

**public** **int** hashCode(){}

@Override

**public** **boolean** equals(Object o){}

}

**public** **class** Artist **implements** Serializable {

**private static final long** ***serialVerisonUID*** = -64567603420449597L;

**private** Integer id;

**private** String email;

**private** Date birthdate;

**private** String name;

**public** Artist(Integer id, String email, Date birthdate, String name){

**super**();

**this**.id = id;

**this**.email = email;

**this**.birthdate = birthdate;

**this**.name = name;

}

**public** Integer getId(){}

**public** **void** setId(Integer id){}

**public** String getEmail(){}

**public** **void** setEmail(String email){}

**public** Date getBirthdate(){}

**public** **void** setBirthdate(Date birthdate){}

**public** String getName(){}

**public** **void** setName(String name){}

@Override

**public** String toString(){}

@Override

**public** **int** hashCode(){}

@Override

**public** **boolean** equals(Object o){}

}

**public** **class** Album **implements** Serializable {

**private static final long** ***serialVerisonUID*** = -64567603420449597L;

**private** Integer id;

**private** String title;

**private** List<Song> songs;

**public** Album(Integer id, String title, List<Song> songs){

**super**();

**this**.id = id;

**this**.title = title;

**this**.songs = songs;

}

**public** Album(Integer id, String title){

**super**();

**this**.id = id;

**this**.title = title;

**this**.songs = **new** ArrayList<Song>();

}

**public** Integer getId(){}

**public** **void** setId(Integer id){}

**public** String getTitle(){}

**public** **void** setTitle(String title){}

**public** List<Song> getSongs(){}

**public** **void** setSongs(List<Song> songs){}

@Override

**public** String toString(){}

@Override

**public** **int** hashCode(){}

@Override

**public** **boolean** equals(Object o){}

}

**Write down the following Java methods, part of a DatabaseManager class which provides you with already initialized Connection c object, that you can freely use.**

**public** List<Album> searchAlbumBySong(Song s) {

List<Album> albums = **new** ArrayList<Album>();

**try** {

String sql = "SELECT a.\* FROM Album AS a JOIN Album\_Songs AS "

+ "as ON a.id = as.id\_song JOIN Songs AS s ON "

+ "s.id = as.id\_song WHERE s.id = ?";

PreparedStatement prep = c.prepareStatement(sql);

prep.setInt(1, s.getId());

ResultSet rs = prep.executeQuery();

**while**(rs.next()){

int id = rs.getInt("id");

String title = rs.getString("title");

albums.add(**new** Album(id, title));

}

rs.close();

prep.close();

**return** albums;

} **catch**(SQLException e) {

e.printStackTrace();

}

**return null**;

}

**public void** createAlbum(String title, List<Songs> songs) {

**try** {

String sql = "INSERT INTO Album title VALUES ?";

PreparedStatement prep = c.prepareStatement(sql);

prep.setString(1, title);

ResultSet rs = prep.executeUpdate();

int id = rs.getInt("id");

**for**(int i = 0; i < songs.size(); i++) {

sql = "INSERT INTO Album\_Songs (id\_album, id\_song) "

+ "VALUES (?, ?)";

prep = c.prepareStatement(sql);

prep.setInt(1, id);

prep.setInt(2, songs.get(i).getId());

prep.executeUpdate();

}

prep.close();

} **catch**(SQLException e) {

e.printStackTrace();

}

}