- Relational schema의 수정사항

1. ‘음원관리자’ relation에 ‘비밀번호’ attribute를 추가하지 않았으므로 이를 새롭게 추가하였다.
2. ‘음원관리자’ relation에 ‘전체사용자목록’, ‘전체음원목록’, ‘전체앨범목록’, ‘전체아티스트목록’은 attribute로 존재하는 것이 아닌 database에서의 select를 이용해 얻을 수 있는 결과이므로 해당 attribute는 삭제하였다.
3. ‘일반사용자’ relation과 ‘스트리밍구독자’ relation의 ‘프로필사진’ relation과 ‘음원’ relation의 ‘음원이미지’ attribute는 data type이 image여야 하므로 외부 파일이 필요하다고 판단하여 이 attribute는 삭제하였다.
4. ‘음원’ relation의 ‘가사’ attribute는 문자열의 길이가 매우 길기 때문에 외부 파일이 필요하다고 판단하여 이 attribute는 삭제하였다.
5. ‘일반_감상가능음질’ relation과 ‘스트리밍_감상가능음질’ relation은 음원을 들을 수 있는 ‘음질’을 나타내기 위해 설계되었으나 해당 내용은 데이터베이스에 들어갈 내용이 아니라는 판단이 들어 해당 relation을 삭제하였다.
6. ‘음질’ relation은 음원의 음질을 나타내기 위해 설계되었으나 해당 내용은 프로그램 상으로 구현하는 것이 적합하다는 판단이 들어 해당 relation을 삭제하였다.
7. ‘음원’ relation의 ‘앨범이름’ attribute는 그 tuple의 ‘albumNum’ attribute와 같은 번호를 ‘앨범번호’로 가지는 ‘앨범’ relation의 tuple의 ‘앨범이름’을 찾으면 되므로 attribute가 굳이 없어도 되기 때문에 이를 삭제하였다.
8. ‘음원’ relation의 ‘아티스트이름’ attribute는 그 tuple의 ‘artistNum’ attribute와 같은 번호를 ‘아티스트번호’로 가지는 ‘아티스트’ relation의 tuple의 ‘아티스트이름’을 찾으면 되므로 attribute가 굳이 없어도 되기 때문에 이를 삭제하였다.
9. ‘앨범’ relation의 ‘아티스트이름’ attribute는 그 tuple의 ‘artistNum’ attribute와 같은 번호를 ‘아티스트번호’로 가지는 ‘아티스트’ relation의 tuple의 ‘아티스트이름’을 찾으면 되므로 attribute가 굳이 없어도 되기 때문에 이를 삭제하였다.
10. ‘수록곡’ relation은 하나의 별도 relation이 필요한 것이 아니라 ‘음원’ relation의 tuple들 중 원하는 앨범의 ‘albumNum’ attribute와 같은 값의 ‘albumNum’ attribute를 가진 tuple들만 골라내면 되므로 ‘수록곡’ relation을 삭제하였다.
11. ‘발매한앨범’ relation은 하나의 별도 relation이 필요한 것이 아니라 ‘앨범’ relation의 tuple들 중 원하는 아티스트의 ‘artistNum’ attribute와 같은 값의 ‘artistNum’ attribute를 가진 tuple들만 골라내면 되므로 ‘발매한앨범’ relation을 삭제하였다.
12. ‘발매한곡’ relation은 하나의 별도 relation이 필요한 것이 아니라 ‘음원’ relation의 tuple들 중 원하는 아티스트의 ‘artistNum’ attribute와 같은 값의 ‘artistNum’ attribute를 가진 tuple들만 골라내면 되므로 ‘발매한곡’ relation을 삭제하였다.
13. ‘일반사용자’ relation의 ‘musicNum’ attribute와 ‘스트리밍구독자’ relation의 ‘musicNum’ attribute는 잘못 설계되었기 때문에 이를 삭제하였다.
14. ‘음원’, ‘앨범’, ‘아티스트’ relation의 ‘소개’attribute는 문자열의 길이가 매우 길기 때문에 외부 파일이 필요하다고 판단하여 삭제하였다.
15. ‘음원’, ‘앨범’, ‘아티스트’ relation의 ‘like(좋아요)’, ‘click(조회수)’ attribute는 프로그램 상 구현이 적합할 것 같아 해당 attribute를 삭제하였다.
16. 스트리밍 서비스를 구현하는 것이 이 프로젝트의 목표이므로 스트리밍권을 구매하지 않는 사람을 나타냈던 ‘일반사용자’ relation을 삭제하였다. 또한 ‘일반사용자’의 플레이리스트를 나타냈던 ‘일반플레이리스트’ relation도 삭제하였다.
17. ‘스트리밍구독자’ relation(suser)의 이름을 ‘사용자(user)’로 변경하고 ‘스트리밍플레이리스트’ relation의 이름을 ‘플레이리스트’로 변경하고, 모든 사용자가 스트리밍권을 가지고 있음을 전제로 하여 ‘스트리밍권 이름’, ‘스트리밍권 구매날짜’ attribute를 삭제하였다.
18. ‘type’, ‘style’, ‘member’, ‘musicgenre’ relation은 현재 database에 크게 의미 있는 내용은 아니기 때문에 해당 table은 삭제하였다.

- **프로그램 코드 설명**

1. public int insertAdmin(String id, String pw) : ID와 PW를 입력받아 데이터베이스의 ‘admin’ table에 새로운 관리자를 추가한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
2. public int insertStreamingUser(String id, String pw, int age, String mgr_id) : ID, PW, age, mgr_id를 입력받아 데이터베이스의 ‘user’ table에 새로운 사용자를 추가한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
3. public int insertMusic(int num, String name, int tracknum, Date release, String mgr_id, int albumnum, int artistnum) : num(음악 번호), name, tracknum, release(곡이 발매된 날짜), mgr_id, albumnum, artistnum을 입력받아 데이터베이스의 ‘music’ table에 새로운 음악을 추가한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
4. public int insertAlbum(int num, String name, Date release, String mgr_id, int artistnum) : num(앨범 번호), name(앨범 이름), release(발매일), mgr_id, albumnum, artistnum을 입력받아 데이터베이스의 ‘album’ table에 새로운 앨범을 추가한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
5. public int insertArtist(int num, String name, String debut, String year, String nation, String mgr_id) : num, name, debut(데뷔년도), year(활동연대), nation(국적), mgr_id을 입력받아 데이터베이스의 ‘artist’ table에 새로운 아티스트를 추가한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
6. public int deleteAdmin(String id) : id를 입력받아 데이터베이스의 ‘admin’ table에서 관리자를 삭제한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
7. public int deleteStreamingUser(String id) : id를 입력받아 데이터베이스의 ‘user’ table에서 사용자를 삭제한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
8. public int deleteMusic(int num) : num를 입력받아 데이터베이스의 ‘music’ table에서 음원을 삭제한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
9. public int deleteAlbum(int num) : num를 입력받아 데이터베이스의 ‘album’ table에서 앨범을 삭제한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
10. public int deleteArtist(int num) : num를 입력받아 데이터베이스의 ‘artist’ table에서 아티스트를 삭제한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
11. public int insertStreamingPlaylist(int index, String id, int num) : index(플레이리스트 번호), 아이디, num(플레이리스트 내의 곡 개수)를 입력받아 ‘playlist’ table에 새로운 플레이리스트를 추가한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
12. public int deleteStreamingPlaylist(String id, int index) : id(사용자아이디)와 index(플레이리스트 번호)를 입력받아 ‘playlist’ table에서 입력받은 사용자의 해당 index에 해당하는 플레이리스트를 삭제한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
13. public int updateAdmin(String targetID, String newPW) : targetID와 newPW를 입력받아 ‘admin’ table에서 targetID와 같은 ID를 가진 관리자의 비밀번호를 newPW로 변경한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
14. public int updateStreamingUserPW(String targetID, String newPW) : targetID와 newPW를 입력받아 ‘user’ table에서 targetID와 같은 ID를 가진 스트리밍구독자의 비밀번호를 newPW로 변경한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
15. public int updateStreamingUserAge(String targetID, int age) : targetID와 age를 입력받아 ‘suser’ table에서 targetID와 같은 ID를 가진 스트리밍구독자의 나이를 age로 변경한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
16. public int updateStreamingUserManager(String targetID, String newID) : targetID와 newID를 입력받아 ‘user’ table에서 targetID와 같은 ID를 가진 스트리밍구독자의 매니저의 ID를 newID로 변경한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.

17. public int updateMusicName(int targetNum, String newName) : targetNum와 newName를 입력받아 ‘music’ table에서 targetNum와 같은 번호를 가진 음원의 이름을 newName으로 변경한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
18. public int updateMusicName(int targetNum, int newNum) : targetNum와 newNum를 입력받아 ‘music’ table에서 targetNum와 같은 번호를 가진 음원의 트랙 번호를 newNum으로 변경한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
19. public int updateMusicReleaseDate(int targetNum, Date newDate) : targetNum와 newDate를 입력받아 ‘music’ table에서 targetNum와 같은 번호를 가진 음원의 발매일을 newDate로 변경한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
20. public int updateMusicAlbum(int targetNum, int newNum) : targetNum와 newNum를 입력받아 ‘music’ table에서 targetNum와 같은 번호를 가진 음원의 앨범번호를 newNum로 변경한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
21. public int updateMusicArtist(int targetNum, int newNum) : targetNum와 newNum를 입력받아 ‘music’ table에서 targetNum와 같은 번호를 가진 음원의 아티스트번호를 newNum로 변경한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
22. public int updateMusicManager(int targetNum, String newID) : targetNum와 newID를 입력받아 ‘music’ table에서 targetNum와 같은 번호를 가진 음원의 매니저 ID를 newID로 변경한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
23. public int updateAlbumName(int targetNum, String newName) : targetNum와 newName를 입력받아 ‘album’ table에서 targetNum와 같은 번호를 가진 앨범의 이름을 newName으로 변경한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
24. public int updateAlbumReleaseDate(int targetNum, Date newDate) : targetNum와 newDate를 입력받아 ‘album’ table에서 targetNum와 같은 번호를 가진 앨범의 발매일을 newDate로 변경한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
25. public int updateAlbumArtist(int targetNum, int newNum) : targetNum와 newNum를 입력받아 ‘album’ table에서 targetNum와 같은 번호를 가진 앨범의 발매 가수 번호를 newNum로 변경한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
26. public int updateAlbumManager(int targetNum, String newID) : targetNum와 newID를 입력받아 ‘album’ table에서 targetNum와 같은 번호를 가진 앨범의 매니저의 ID를 newID로 변경한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
27. public int updateArtistName(int targetNum, String newName) : targetNum와 newName를 입력받아 ‘artist’ table에서 targetNum와 같은 번호를 가진 아티스트의 이름을 newName으로 변경한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
28. public int updateArtistDebut(int targetNum, String year) : targetNum와 year를 입력받아 ‘artist’ table에서 targetNum와 같은 번호를 가진 아티스트의 데뷔년도를 year로 변경한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
29. public int updateArtistActive(int targetNum, String year) : targetNum와 year를 입력받아 ‘artist’ table에서 targetNum와 같은 번호를 가진 아티스트의 활동년도를 year로 변경한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
30. public int updateArtistNation(int targetNum, String nation) : targetNum와 nation를 입력받아 ‘artist’ table에서 targetNum와 같은 번호를 가진 아티스트의 국적을 nation으로 변경한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
31. public int updateArtistManager(int targetNum, String newID) : targetNum와 newID를 입력받아 ‘artist’ table에서 targetNum와 같은 번호를 가진 아티스트의 매니저의 ID를 newID로 변경한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
32. public int printAdminAll() : ‘admin’ table에서 모든 관리자의 ID와 비밀번호를 출력한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.

33. public int printAdminIDAll() : ‘admin’ table에서 모든 관리자의 ID를 출력한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
34. public int printAdmin(String _id) : ‘admin’ table에서 인자로 전달받은 id와 같은 id를 가지고 있는 관리자의 ID와 비밀번호를 출력한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
35. public int printStreamingUserAll() : ‘user’ table에서 모든 스트리밍 구독자의 ID, 비밀번호, 나이, 관리자 ID를 출력한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
36. public int printStreamingUserIDAll() : ‘user’ table에서 모든 스트리밍 구독자의 ID를 출력한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
37. public int printStreamingUser(String _id) : ‘user’ table에서 인자로 전달받은 id와 같은 id를 가지고 있는 스트리밍 구독자의 ID, 비밀번호, 나이, 관리자 ID를 출력한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
38. public int printMusicAll() : ‘music’ table에서 모든 음악의 고유번호, 이름, 트랙번호, 발매일, 관리자 ID, 앨범번호, 아티스트 번호를 출력한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
39. public int printMusicNameAll() : ‘music’ table에서 모든 음악의 고유번호, 이름을 출력한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
40. public int printMusic(int _num) : ‘music’ table에서 인자로 전달받은 num와 같은 번호를 가지고 있는 음악의 고유번호, 이름, 트랙번호, 발매일, 관리자ID, 앨범번호, 아티스트 번호를 출력한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
41. public int printAlbumAll() : ‘album’ table에서 모든 앨범의 고유번호, 이름, 발매일, 관리자ID, 아티스트 번호를 출력한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
42. public int printAlbumNameAll() : ‘album’ table에서 모든 앨범의 이름을 출력한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
43. public int printAlbum(int _num) : ‘album’ table에서 인자로 전달받은 num과 같은 num을 가지고 있는 앨범의 고유번호, 이름, 발매일, 관리자ID, 아티스트 번호를 출력한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
44. public int printArtistAll() : ‘artist’ table에서 모든 아티스트의 고유번호, 이름, 데뷔년도, 활동년도, 국적, 관리자 ID를 출력한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
45. public int printArtistNameAll() : ‘artist’ table에서 모든 아티스트의 이름을 출력한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
46. public int printArtist(int _num) : ‘artist’ table에서 인자로 전달받은 num과 같은 num을 가지고 있는 아티스트의 고유번호, 이름, 데뷔년도, 활동년도, 국적, 관리자 ID를 출력한다. SQL문이 제대로 실행되었을 경우 1을 반환하고 그렇지 않은 경우 0을 반환한다.
47. public int Music(String musicName, String albumName, String artistName, int trackNum, Date date, String mgr_id) : 음원 이름, 앨범 이름, 아티스트 이름, 트랙넘버, 발매일, 관리자 ID를 인자로 전달받아 ‘music’ table에 새로운 음원을 생성한다. 새로 추가하고자 하는 음원이 이미 table안에 존재하면 2를 return하며 해당 음원이 table에 존재하지 않을 경우 마지막 음원번호+1인 값을 음원 번호로 부여한다. ‘artist’ table과 ‘album’ table에 추가하고자 하는 음원의 아티스트나 앨범이 존재하지 않을 경우 추가적인 정보를 입력받아 새로운 아티스트와 앨범도 함께 추가한다. 음원 추가에 성공하면 1을 return하고 실패하면 0을 return한다.
48. public int Album(String albumName, String artistName, Date date, String mgr_id) : 앨범 이름, 아티스트 이름, 발매일, 관리자 ID를 인자로 전달받아 ‘album’ table에 새로운 앨범을 생성한다. 새로 추가하고자 하는 앨범이 이미 table안에 존재하면 2를 return하며 해당 앨범이 table에 존재하지 않을 경우 마지막 앨범번호+1인 값을 고유 번호로 부여한다. ‘artist’ table에 추가하고자 하는 앨범의 아티스트가 존재하지 않을

경우 추가적인 정보를 입력받아 새로운 아티스트도 함께 추가한다. 앨범 추가에 성공하면 1을 return하고 실패하면 0을 return한다.

49. public int Playlist(String id, String musicName, String artistName, int index) : 사용자 id, 음원 이름, 아티스트 이름, 플레이리스트 번호를 인자로 전달받아 'playlist' table에 새로운 앨범을 생성한다. 새로 추가하고자 하는 곡이 존재하지 않으면 table안에 존재하면 2를 return한다. 플레이리스트에 음원 추가에 성공하면 1을 return하고 실패하면 0을 return한다.
50. public int searchMusic(String musicName) : 음원 이름을 인자로 전달받아 해당 음원의 음원번호, 음원이름, 아티스트이름을 출력한다. 해당 음원이 존재하지 않으면 2를 return하고 음원 출력에 성공하면 1을 return하고 실패하면 0을 return한다.
51. public int searchAlbum(String albumName) : 앨범 이름을 인자로 전달받아 해당 앨범의 앨범번호와 그 앨범에 속한 모든 음원의 음원번호, 음원이름, 아티스트이름을 출력한다. 해당 앨범이 존재하지 않으면 2를 return하고 앨범 출력에 성공하면 1을 return하고 실패하면 0을 return한다.
52. public int searchArtist(String artistName) : 아티스트 이름을 인자로 전달받아 해당 아티스트의 아티스트번호와 그 앨범에 속한 모든 음원의 음원번호, 음원이름을 출력한다. 해당 아티스트가 존재하지 않으면 2를 return하고 앨범 출력에 성공하면 1을 return하고 실패하면 0을 return한다.
53. public int deleteStreamingPlaylist2(String id) : 사용자 id를 인자로 전달받아 해당 id를 userID로 갖고 있는 모든 플레이리스트를 삭제한다. 삭제에 실패하면 0을 return한다.
54. public int deleteStreamingPlaylist2(String name) : 음원 이름을 인자로 전달받아 해당 이름을 musicName으로 갖고 있는 모든 플레이리스트를 삭제한다. 삭제에 실패하면 0을 return한다.
55. public int delMusic(String musicName) : 음원의 이름을 입력받아 'music' table에서 해당 이름에 해당하는 음원 이름을 가진 tuple을 삭제한다. 음원을 삭제하기 전에 먼저 'playlist' table에서 해당 음원을 가지고 있는 모든 tuple을 삭제한다. 이 과정이 실패하면 2를 return한다. 플레이리스트 삭제에 성공하였을 경우 삭제할 음원이름과 같은 음원을 가지고 있는 tuple의 musicNum, albumNum, artistNum을 찾아낸다. 그리고 음원 table에서 같은 albumNum과 artistNum을 가지고 있는 tuple의 개수를 세서 만약 1일 경우 해당 앨범 혹은 아티스트에 속한 음원이 현재 삭제할 음원 한 개 뿐이라는 의미이므로 음원을 삭제한 이후 개수가 0인 앨범 혹은 아티스트를 같이 삭제한다. 이 과정이 성공적으로 수행되었을 경우 1을 return하고 실패하였을 경우 0을 return한다.
56. public int Artist(String artistName, String debut, String year, String nation, String mgr_id) : 아티스트 이름, 데뷔년도, 활동년도, 국적, 관리자 id를 전달받고 select를 통해 artistNum을 정해 insertArtist()를 호출해 artist table에 tuple을 추가한다.

● 수행되는 기능 스크린샷

제공되는 기능은 아래와 같으며, 이 중 하늘색으로 표시된 내용만 스크린샷을 첨부하였다.

0. Exit

1. Admin Menu
    A. Return to Previous Menu
    B. User
        0) Return to Previous Menu
        1) Insert New User
        2) Delete User
        3) Update User
            a) Change Password
            b) Change Age
            c) Change Manager
        4) View User List
            a) View ALL
            b) View ID ALL
            c) View One User ALL
    C. Music
        0) Return to Previous Menu
        1) Insert New Music
        2) Delete Music
        3) Update Music
            a) Change Name
            b) Change Track Number
            c) Change Release Date
            d) Change Album
            f) Change Manager
        4) View Music List
            a) View ALL
            b) View Name ALL
            c) View One Music ALL
    D. Album
        0) Return to Previous Menu
        1) Insert New Album
        2) Delete Album
        3) Update Album
            a) Change Name
            b) Change Release Date
            c) Change Artist
            d) Change Manager
        4) View Album List
            a) View ALL
            b) View Name ALL
            c) View One Album ALL
    E. Artist
        0) Return to Previous Menu
        1) Insert New Artist
        2) Delete Artist
        3) Update Artist
            a) Change Name
            b) Change Debut Year
            c) Change Active Year
            d) Change Nation
            e) Change Manager
        4) View Artist List
            a) View ALL
            b) View Name ALL
            c) View One Artist ALL
    F. Admin
        0) Return to Previous Menu
        1) Insert New Admin
        2) Delete Admin
        3) Change Admin Password
        4) View Admin List
            a) View ALL
            b) View ID ALL
            c) View One Admin ALL

2. User Menu
    A. Return to Previous Menu
    B. Playlist
        0) Return to Previous Menu
        1) Add New Music into PlayList
        2) Delete Music in PlayList
        3) Delete PlayList
        4) View PlayList
            a) View User's Playlist
            b) View User's One Playlist

    C. Search
        0) Return to Previous Menu
        1) Search Music
        2) Search Album
        3) Search Artist

1. User

```
0. Exit
1. Admin Menu
2. User Menu
Input > 1

0. Return to Previous Menu
1. User
2. Music
3. Album
4. Artist
5. Admin
Input > 1

0. Return to Previous Menu
1. Insert New User
2. Delete User
3. Update User
4. View User List
Input > 1

*** New User ***
ID > gg
PW > 1111
Age > 20
Manager ID > coral2cola
Insert New User done!
0. Exit
1. Admin Menu
2. User Menu
Input >
```

insert new user

```
0. Exit
1. Admin Menu
2. User Menu
Input > 1

0. Return to Previous Menu
1. User
2. Music
3. Album
4. Artist
5. Admin
Input > 1

0. Return to Previous Menu
1. Insert New User
2. Delete User
3. Update User
4. View User List
Input > 3

0. Change Password
1. Change Age
2. Change Manager
Input > 0

*** Change Password ***
Target ID > gg
New Password > 0000
Change Password done!
0. Exit
1. Admin Menu
2. User Menu
Input >
```

change user's PW

```
0. Exit
1. Admin Menu
2. User Menu
Input > 1

0. Return to Previous Menu
1. User
2. Music
3. Album
4. Artist
5. Admin
Input > 1

0. Return to Previous Menu
1. Insert New User
2. Delete User
3. Update User
4. View User List
Input > 2

*** Delete User ***
User ID > gg
Delete User done!
```

delete user

```
0. Exit
1. Admin Menu
2. User Menu
Input > 1

0. Return to Previous Menu
1. User
2. Music
3. Album
4. Artist
5. Admin
Input > 1

0. Return to Previous Menu
1. Insert New User
2. Delete User
3. Update User
4. View User List
Input > 4

0. View ALL
1. View ID ALL
2. View One Admin ALL
Input > 0
```

view user list all

ID	PASSWORD	AGE	MANAGER ID
3zu	1234	21	coral2cola
gg	0000	20	coral2cola
moonshort	1111	19	love
raffine	1234	22	love

```
View ALL done!
```

2. Music

```
0. Exit
1. Admin Menu
2. User Menu
Input > 1

0. Return to Previous Menu
1. User
2. Music
3. Album
4. Artist
5. Admin
Input > 2

0. Return to Previous Menu
1. Insert New Music
2. Delete Music
3. Update Music
4. View Music List
Input > 1

*** New Music ***
Music Name > test
Album Name > test
Artist Name > test
Track Number > 1
Release Date  (yyyy-mm-dd)> 2020-12-06
Manager ID > coral2cola
*** Add test(artist) In Database ***
Debut Year > 2020
Active Year > 2020
Nation > Korea
Add test(artist) In Database Done!
*** Add test(album) In Database ***
Add test(album) In Database Done!
Insert New Music done!
```

Insert new music

```
0. Exit
1. Admin Menu
2. User Menu
Input > 1

0. Return to Previous Menu
1. User
2. Music
3. Album
4. Artist
5. Admin
Input > 2

0. Return to Previous Menu
1. Insert New Music
2. Delete Music
3. Update Music
4. View Music List
Input > 3

0. Change Name
1. Change Track Number
2. Change Release Date
3. Change Album
4. Change Manager
Input > 0

*** Change Name ***
Target Music Number > 18
New Name > TEST
Change Name done!
```

Change music name

```
0. Exit
1. Admin Menu
2. User Menu
Input > 1

0. Return to Previous Menu
1. User
2. Music
3. Album
4. Artist
5. Admin
Input > 2

0. Return to Previous Menu
1. Insert New Music
2. Delete Music
3. Update Music
4. View Music List
Input > 2

*** Delete Music ***
Music Name > TEST
Delete Music done!
```

delete music

```
0. Exit
1. Admin Menu
2. User Menu
Input > 1

0. Return to Previous Menu
1. User
2. Music
3. Album
4. Artist
5. Admin
Input > 2

0. Return to Previous Menu
1. Insert New Music
2. Delete Music
3. Update Music
4. View Music List
Input > 4

0. View ALL
1. View Name ALL
2. View One Music ALL
Input > 0
```

view music list all

MUSIC NUM	MUSIC NAME	TRACK NUM	RELEASE DATE	MANAGER ID	ALBUM NUM	ARTIST NUM
1	Propose	1	2017-07-08	coral2cola	1	1
2	Purple Daddy	4	2013-04-02	coral2cola	2	1
3	Beap Sea	2	2013-04-02	coral2cola	2	1
8	NAN CHUN	1	2020-05-10	coral2cola	5	2
9	Eternity	1	2018-12-10	coral2cola	6	1
10	Heroine	1	2018-01-18	love	7	3
11	Siren	1	2018-09-04	coral2cola	8	3
12	SAM SAM	1	2019-05-30	love	3	1
13	Superhero	1	2019-05-30	love	3	1
14	Gashina	1	2017-08-22	love	9	3
15	Pallette	1	2017-04-21	coral2cola	10	4
16	Ready	3	2019-05-30	coral2cola	3	1
17	VVS	1	2020-12-05	love	11	6
18	TEST	1	2020-12-06	coral2cola	12	7

```
View ALL done!
```

3. Album

```
0. Exit
1. Admin Menu
2. User Menu
Input > 1

0. Return to Previous Menu
1. User
2. Music
3. Album
4. Artist
5. Admin
Input > 3

0. Return to Previous Menu
1. Insert New Album
2. Delete Album
3. Update Album
4. View Album List
Input > 1

*** New Album ***
Album Name > test
Artist Name > test
Release Date (yyyy-mm-dd) > 2020-12-06
Manager ID > coral2cola
*** Add test(artist) In Database ***
Debut Year > 2020
Active Year > 2020
Nation > Korea
Add test(artist) In Database Done!
Insert New Album done!
```

insert new album

```
0. Exit
1. Admin Menu
2. User Menu
Input > 1

0. Return to Previous Menu
1. User
2. Music
3. Album
4. Artist
5. Admin
Input > 3

0. Return to Previous Menu
1. Insert New Album
2. Delete Album
3. Update Album
4. View Album List
Input > 3

0. Change Name
1. Change Release Date
2. Change Artist
3. Change Manager
Input > 0

*** Change Name ***
Target Album Number > 12
New Name > TEST
Change Name done!
```

change album's name

```
0. Exit
1. Admin Menu
2. User Menu
Input > 1

0. Return to Previous Menu
1. User
2. Music
3. Album
4. Artist
5. Admin
Input > 3

0. Return to Previous Menu
1. Insert New Album
2. Delete Album
3. Update Album
4. View Album List
Input > 2

*** Delete Album ***
Album Number > 12
Delete Album done!
```

delete album

```
0. Exit
1. Admin Menu
2. User Menu
Input > 1

0. Return to Previous Menu
1. User
2. Music
3. Album
4. Artist
5. Admin
Input > 3

0. Return to Previous Menu
1. Insert New Album
2. Delete Album
3. Update Album
4. View Album List
Input > 4

0. View ALL
1. View Name ALL
2. View One Album ALL
Input > 0
```

view album list all

ALBUM NUM	ALBUM NAME	RELEASE DATE	MANAGER ID	ARTIST NUM
1	Propose	2017-07-08	coral2cola	1
2	It's Okay, Dear	2013-04-02	coral2cola	1
3	Stand	2019-05-30	love	1
5	NAN CHUN	2020-05-10	love	2
6	Eternity	2018-12-10	coral2cola	1
7	Heroine	2018-01-18	love	3
8	WARNING	2018-09-04	coral2cola	3
9	Gashina	2017-08-22	love	3
10	Pallette	2017-04-21	coral2cola	4
11	VVS	2020-12-05	love	6
12	TEST	2020-12-06	coral2cola	7

```
View ALL done!
```

4. Artist

```
0. Exit
1. Admin Menu
2. User Menu
Input > 1

0. Return to Previous Menu
1. User
2. Music
3. Album
4. Artist
5. Admin
Input > 3

0. Return to Previous Menu
1. Insert New Album
2. Delete Album
3. Update Album
4. View Album List
Input > 1

*** New Album ***
Album Name > test
Artist Name > omo
Release Date (yyyy-mm-dd) > 2020-12-06
Manager ID > love
*** Add omo(artist) In Database ***
Debut Year > 2020
Active Year > 2020
Nation > Korea
Add omo(artist) In Database Done!
Insert New Album done!
```

insert new artist

```
0. Exit
1. Admin Menu
2. User Menu
Input > 1

0. Return to Previous Menu
1. User
2. Music
3. Album
4. Artist
5. Admin
Input > 4

0. Return to Previous Menu
1. Insert New Artist
2. Delete Artist
3. Update Artist
4. View Artist List
Input > 3

0. Change Name
1. Change Debut Year
2. Change Active Year
3. Change Nation
4. Change Manager
Input > 0

*** Change Name ***
Target Artist Number > 8
New Name > *omo*
Change Name done!
```

Change artist's name

```
0. Exit
1. Admin Menu
2. User Menu
Input > 1

0. Return to Previous Menu
1. User
2. Music
3. Album
4. Artist
5. Admin
Input > 4

0. Return to Previous Menu
1. Insert New Artist
2. Delete Artist
3. Update Artist
4. View Artist List
Input > 2

*** Delete Artist ***
Artist Number > 7
Delete Artist done!
```

delete artist

```
0. Exit
1. Admin Menu
2. User Menu
Input > 1

0. Return to Previous Menu
1. User
2. Music
3. Album
4. Artist
5. Admin
Input > 4

0. Return to Previous Menu
1. Insert New Artist
2. Delete Artist
3. Update Artist
4. View Artist List
Input > 4

0. View ALL
1. View Name ALL
2. View One Album ALL
Input > 0
```

view artist list all

ARTIST NUM	ARTIST NAME	DEBUT YEAR	ACTIVE YEAR	NATION	MANAGER ID
1	sunwoojunga	1985-01-01	2020-01-01	Korea	love
2	sesoneon	2017-01-01	2020-01-01	Korea	coral2cola
3	SUNMI	2007-01-01	2020-01-01	Korea	love
4	IU	2008-01-01	2020-01-01	Korea	coral2cola
6	Mushvenom	2019-01-01	2020-01-01	Korea	coral2cola
7	test	2020-01-01	2020-01-01	Korea	coral2cola

```
View ALL done!
```

5. Admin

```
0. Exit
1. Admin Menu
2. User Menu
Input > 1

0. Return to Previous Menu
1. User
2. Music
3. Album
4. Artist
5. Admin
Input > 5

0. Return to Previous Menu
1. Insert New Admin
2. Delete Admin
3. Change Admin Password
4. View Admin List
Input > 1

*** New Admin ***
ID > omo
PW > 1111
Insert New Admin done!
```

insert new admin

```
0. Exit
1. Admin Menu
2. User Menu
Input > 1

0. Return to Previous Menu
1. User
2. Music
3. Album
4. Artist
5. Admin
Input > 5

0. Return to Previous Menu
1. Insert New Admin
2. Delete Admin
3. Change Admin Password
4. View Admin List
Input > 3

*** Change Admin Password ***
Target Admin ID > omo
New Password > 0000
Change Admin Password done!
```

Change admin password

```
0. Exit
1. Admin Menu
2. User Menu
Input > 1

0. Return to Previous Menu
1. User
2. Music
3. Album
4. Artist
5. Admin
Input > 5

0. Return to Previous Menu
1. Insert New Admin
2. Delete Admin
3. Change Admin Password
4. View Admin List
Input > 4

0. View ALL
1. View ID ALL
2. View One Admin ALL
Input > 0

-----------------------------------------
ID                      PASSWORD
-----------------------------------------
coral2cola              1234
love                    1234
omo                     0000
-----------------------------------------
View ALL done!
```

view admin list all

```
0. Exit
1. Admin Menu
2. User Menu
Input > 1

0. Return to Previous Menu
1. User
2. Music
3. Album
4. Artist
5. Admin
Input > 5

0. Return to Previous Menu
1. Insert New Admin
2. Delete Admin
3. Change Admin Password
4. View Admin List
Input > 2

*** Delete Admin ***
Admin ID > omo
Delete Admin done!
```

delete admin

6. Playlist

```
0. Exit
1. Admin Menu
2. User Menu
Input > 2

0. Return to Previous Menu
1. Playlist
2. Search
Input > 1

0. Return to Previous Menu
1. Add New Music into PlayList
2. Delete Music in PlayList
3. Delete PlayList
4. View PlayList
Input > 1

*** Add New Music into PlayList ***
Index > 1
User ID > 3zu
Music Name > superhero
Artist Name > sunwoojunga
Add New Music into PlayList done!
```

add new music into playlist

```
0. Exit
1. Admin Menu
2. User Menu
Input > 2

0. Return to Previous Menu
1. Playlist
2. Search
Input > 1

0. Return to Previous Menu
1. Add New Music into PlayList
2. Delete Music in PlayList
3. Delete PlayList
4. View PlayList
Input > 2

*** Delete Music in PlayList ***
User ID > 3zu
Music Name > superhero
Delete Music in PlayList done!
```

delete music in playlist

```
0. Exit
1. Admin Menu
2. User Menu
Input > 2

0. Return to Previous Menu
1. Playlist
2. Search
Input > 1

0. Return to Previous Menu
1. Add New Music into PlayList
2. Delete Music in PlayList
3. Delete PlayList
4. View PlayList
Input > 4

0. View User's Playlist
1. View User's One Playlist
Input > 0
Target User ID > 3zu
-------------------------------------------------
INDEX           MUSIC NUM       MUSIC NAME
-------------------------------------------------
1               1               Propose
2               3               Beap Sea
1               10              Heroine
1               12              SAM SAM
1               13              Superhero
-------------------------------------------------
View User's Playlist done!
```

view user's playlist

● 사용되는 SQL문 명세

**** INSERT ****

1. INSERT INTO admin VALUES(AdminID,AdminPW)

admin(관리자) table에 AdminID(관리자ID), AdminPw(관리자비밀번호)를 가진 tuple을 insert한다.

2. INSERT INTO user VALUES(userID,userPW,userAge,mgr_ID)

user(사용자) table에 userID(사용자ID), userPW(사용자비밀번호), userAge(사용자나이), mgr_ID(관리자아이디)를 가진 tuple을 insert한다.

3. INSERT INTO music VALUES(musicNum,musicName,trackNum,releaseDate,mgr_ID,albumNum,artistNum)

music(음악) table에 musicNum(음원번호), musicName(음원이름), trackNum(트랙넘버), releaseDate(음원발매일), mgr_ID(관리자아이디), albumNum(음원이속한앨범번호), artistNum(음원을발매한아티스트번호)를 가진 tuple을 insert한다.

4. INSERT INTO album VALUES(albumNum,albumName,releaseDate,mgr_ID,artistNum)

album(앨범) table에 albumNum(앨범번호), albumName(앨범이름), releaseDate(앨범발매일), mgr_ID(관리자아이디), artistNum(앨범을발매한아티스트번호)를 가진 tuple을 insert한다.

5. INSERT INTO artist VALUES(artistNum,artistName,debut,year,nation,mgr_ID)

artist(아티스트) table에 artistNum(아티스트번호), artistName(아티스트이름), debut(데뷔년도), year(활동연대), nation(국적), mgr_ID(관리자ID)를 가진 tuple을 insert한다.

6. INSERT INTO playlist VALUES(index,userID,musicNum)

playlist(플레이리스트) table에 index(플레이리스트번호), userID(사용자ID), musicNum(음원번호)를 가진 tuple을 insert한다.

**** UPDATE ****

7. UPDATE admin SET AdminPW='newPW' WHERE AdminID='targetID'

admin(관리자) table에서 AdminID(관리자아이디)의 value가 targetID와 같은 tuple의 AdminPW(관리자비밀번호)의 값을 newPW로 변경한다.

8. UPDATE user SET userPW='newPW' WHERE userID='targetID'

user(사용자) table에서 userID(사용자아이디)의 value가 targetID와 같은 tuple의 userPW(사용자비밀번호)의 값을 newPW로 변경한다.

9. UPDATE user SET userAge=age WHERE userID='targetID'

user(사용자) table에서 userID(사용자아이디)의 value가 targetID와 같은 tuple의 userAge(사용자나이)의 값을 age로 변경한다.

10. UPDATE user SET mgr_ID='newID' WHERE userID='targetID'

user(사용자) table에서 userID(사용자아이디)의 value가 targetID와 같은 tuple의 mgr_ID(관리자아이디)의 값을 newID로 변경한다.

11. UPDATE music SET musicName='newName' WHERE musicNum=targetNum

music(음원) table에서 musicNum(음원번호)의 value가 targetNum와 같은 tuple의 musicName(음원이름)의 값을 newName로 변경한다.

12. UPDATE music SET trackNum=newNum WHERE musicNum=targetNum

music(음원) table에서 musicNum(음원번호)의 value가 targetNum와 같은 tuple의 trackNum(트랙넘버)의 값을 newNum로 변경한다.

13. UPDATE music SET releaseDate='newDate' WHERE musicNum=targetNum

music(음원) table에서 musicNum(음원번호)의 value가 targetNum와 같은 tuple의 releaseDate(음원발매일)의 값을 newDate로 변경한다.

14. UPDATE music SET albumNum=newNum WHERE musicNum=targetNum

music(음원) table에서 musicNum(음원번호)의 value가 targetNum와 같은 tuple의 albumNum(음원이속한앨범번호)의 값을 newNum로 변경한다.

15. UPDATE music SET artistNum=newNum WHERE musicNum=targetNum

music(음원) table에서 musicNum(음원번호)의 value가 targetNum와 같은 tuple의 artistNum(음원을발매한아티스트번호)의 값을 newNum로 변경한다.

16. UPDATE music SET mgr_ID='newID' WHERE musicNum=targetNum

music(음원) table에서 musicNum(음원번호)의 value가 targetNum와 같은 tuple의 mgr_ID(관리자ID)의 값을 newID로 변경한다.

17. UPDATE album SET albumName='newName' WHERE albumNum=targetNum

album(앨범) table에서 albumNum(앨범번호)의 value가 targetNum와 같은 tuple의 albumName(앨범이름)의 값을 newName로 변경한다.

18. UPDATE album SET releaseDate='newDate' WHERE albumNum=targetNum

album(앨범) table에서 albumNum(앨범번호)의 value가 targetNum와 같은 tuple의 releaseDate(앨범발매일)의 값을 newDate로 변경한다.

19. UPDATE album SET artistNum=newNum WHERE albumNum=targetNum

album(앨범) table에서 albumNum(앨범번호)의 value가 targetNum와 같은 tuple의 artistNum(앨범을발매한아티스트번호)의 값을 newNum로 변경한다.

20. UPDATE album SET mgr_ID='newID' WHERE albumNum=targetNum

album(앨범) table에서 albumNum(앨범번호)의 value가 targetNum와 같은 tuple의 mgr_ID(관리자ID)의 값을 newID로 변경한다.

21. UPDATE artist SET artistName='newName' WHERE artistNum=targetNum

artist(아티스트) table에서 artistNum(아티스트번호)의 value가 targetNum와 같은 tuple의 artistName(아티스트이름)의 값을 newName로 변경한다.

22. UPDATE artist SET debut='newDebut' WHERE artistNum=targetNum

artist(아티스트) table에서 artistNum(아티스트번호)의 value가 targetNum와 같은 tuple의 debut(데뷔년도)의 값을 newDebut로 변경한다.

23. UPDATE artist SET year='newYear' WHERE artistNum=targetNum

artist(아티스트) table에서 artistNum(아티스트번호)의 value가 targetNum와 같은 tuple의 year(활동년도)의 값을 newYear로 변경한다.

24. UPDATE artist SET nation='newNation' WHERE artistNum=targetNum

artist(아티스트) table에서 artistNum(아티스트번호)의 value가 targetNum와 같은 tuple의 nation(국적)의 값을 newNation로 변경한다.

25. UPDATE artist SET mgr_ID='newID' WHERE artistNum=targetNum

artist(아티스트) table에서 artistNum(아티스트번호)의 value가 targetNum와 같은 tuple의 mgr_ID(관리자ID)의 값을 newID로 변경한다.

**** DELETE ****

26. DELETE FROM admin where AdminID='id'

admin(관리자) table에서 AdminID(관리자아이디)의 value가 targetID와 같은 tuple을 삭제한다.

27. DELETE FROM user where userID='id'

user(사용자) table에서 userID(사용자아이디)의 value가 targetID와 같은 tuple을 삭제한다.

28. DELETE FROM music where musicNum=num

music(음원) table에서 musicNum(음원번호)의 value가 targetNum와 같은 tuple을 삭제한다.

29. DELETE FROM album where albumNum=num

album(앨범) table에서 albumNum(앨범번호)의 value가 targetNum와 같은 tuple을 삭제한다.

30. DELETE FROM artist where artistNum=num

artist(아티스트) table에서 artistNum(아티스트번호)의 value가 targetNum와 같은 tuple을 삭제한다.

31. DELETE FROM playlist where index=num

playlist(플레이리스트) table에서 index(플레이리스트번호)의 value가 num와 같은 tuple을 삭제한다.

32. DELETE FROM playlist where userID='id'

playlist(플레이리스트) table에서 userID(사용자ID)의 값이 id인 tuple들을 전부 삭제한다.

33. DELETE FROM playlist WHERE musicNum=num

playlist(플레이리스트) table에서 musicNum(음원번호)의 value가 num와 같은 tuple을 삭제한다.

34. DELETE FROM playlist WHERE playlist.index=_index and userID='id'

playlist(플레이리스트) table에서 index의 값이 _index이고 userID의 값이 id인 tuple을 삭제한다.

**** SELECT ****

35. SELECT * FROM admin

admin(관리자) table의 모든 attribute의 value를 select한다.

36. SELECT AdminID FROM admin

admin(관리자) table의 AdminID(관리자ID) attribute의 value를 select한다.

37. SELECT * FROM admin WHERE AdminID='_id'

admin(관리자) table의 AdminID(관리자ID)의 값이 _id인 tuple들의 모든 attribute의 value를 select한다.

38. SELECT * FROM user

user(사용자) table의 모든 attribute의 value를 select한다.

39. SELECT userID FROM user

user(사용자) table의 userID(사용자ID) attribute의 value를 select한다.

40. SELECT * FROM user WHERE userID='_id'

user(사용자) table의 userID(사용자ID)의 값이 _id인 tuple들의 모든 attribute의 value를 select한다.

41. SELECT * FROM music

music(음원) table의 모든 attribute의 value를 select한다.

42. SELECT musicName FROM music

music(음원) table의 musicName(음원이름) attribute의 value를 select한다.

43. SELECT * FROM music WHERE musicNum=_num

music(음원) table의 musicName(음원번호)의 값이 _num인 tuple들의 모든 attribute의 value를 select한다.

44. SELECT * FROM album

album(앨범) table의 모든 attribute의 value를 select한다.

45. SELECT albumName FROM album

album(앨범) table의 albumName(앨범이름) attribute의 value를 select한다.

46. SELECT * FROM album WHERE albumNum=_num

album(앨범) table의 albumNum(앨범번호)의 값이 _num인 tuple들의 모든 attribute의 value를 select한다.

47. SELECT * FROM artist

artist(아티스트) table의 모든 attribute의 value를 select한다.

48. SELECT artistName FROM artist

artist(아티스트) table의 artistName(아티스트이름) attribute의 value를 select한다.

49. SELECT * FROM artist WHERE artistNum=_num

artist(아티스트) table의 artistNum(아티스트번호)의 값이 _num인 tuple들의 모든 attribute의 value를 select한다.

50. SELECT playlist.index,playlist.musicNum,musicName FROM playlist,music WHERE userID='_id' and playlist.musicNum= music.musicNum

playlist(플레이리스트), music(음원) table에서 userID(사용자ID)의 값이 _id이고 playlist table의 musicNum(음원번호)과 music table의 musicNum이 같은 tuple들에 대해 index(플레이리스트번호), musicNum, musicName(음원이름) attribute의 값을 select한다.

51. SELECT playlist.index,playlist.musicNum,musicName FROM playlist,music WHERE userID='_id' and playlist.index=_index and playlist.musicNum=music.musicNum

playlist(플레이리스트), music(음원) table에서 userID(사용자ID)의 값이 _id이고, index(플레이리스트번호)의

값이 _index이고, playlist table의 musicNum(음원번호)과 music table의 musicNum이 같은 tuple들에 대해 index, musicNum, musicName(음원이름) attribute의 값을 select한다.

52. SELECT musicNum,musicName FROM music

music(음원) table의 musicNum(음원번호), musicName(음원이름) attribute의 value를 select한다.

53. SELECT artistNum,artistName FROM artist

artist(아티스트) table의 artistNum(아티스트번호), artistName(아티스트이름) attribute의 value를 select한다.

54. SELECT albumNum,albumName FROM album

album(앨범) table의 albumNum(앨범번호), albumName(앨범이름) attribute의 value를 select한다.

55. SELECT musicNum,musicName,artistNum FROM music

music(음원) table의 musicNum(음원번호), musicName(음원이름),artistNum(아티스트번호) attribute의 value를 select한다.

56. SELECT musicNum,albumName,artistName FROM music,album,artist WHERE musicName='musicName' and music.albumNum=album.albumNum and music.artistNum=artist.artistNum

music(음원), album(앨범), artist(아티스트) table에서 musicName(음원이름)의 값이 musicName이고, music table의 albumNum(앨범번호)과 album table의 albumNum이 같고, music table의 artistNum(아티스트번호)과 artist table의 artistNum이 같은 tuple들에 대해 musicNum(음원번호), albumName(앨범이름), artistName(아티스트이름) attribute의 값을 select한다.

57. SELECT musicNum,album.albumNum,musicName,artistName FROM music,album,artist WHERE albumName='albumName' and music.albumNum=album.albumNum and album.artistNum=artist.artistNum

music(음원), album(앨범), artist(아티스트) table에서 albumName(앨범이름)의 값이 albumName이고, music table의 albumNum(앨범번호)과 album table의 albumNum이 같고, album table의 artistNum(아티스트번호)과 artist table의 artistNum이 같은 tuple들에 대해 musicNum(음원번호), albumNum, musicName(음원이름), artistName(아티스트이름) attribute의 값을 select한다.

58. SELECT musicNum,artist.artistNum,musicName FROM music,artist WHERE artistName='artistName' and artist.artistNum=music.artistNum

music(음원), artist(아티스트) table에서 artistName(아티스트이름)의 값이 artistName이고, music table의 artistNum(아티스트번호)과 artist table의 artistNum이 같은 tuple들에 대해 musicNum(음원번호), artistNum, musicName(음원이름) attribute의 값을 select한다.

59. SELECT musicNum from music WHERE musicName='musicName'

music(음원) table에서 musicName(음원이름)의 값이 musicName인 musicNum(음원번호) attribute의 value를 select한다.

60. SELECT musicNum from playlist

playlist(플레이리스트) table에서 musicNum(음원번호) attribute의 value를 select한다.

61. SELECT musicNum,albumNum,artistNum FROM music WHERE musicName='musicName'

music(음원) table에서 musicName(음원이름)의 값이 musicName인 musicNum(음원번호), albumNum(앨범번호), artistNum(아티스트번호) attribute의 value를 select한다.

62. SELECT albumNum FROM music WHERE albumNum=num

music(음원) table에서 musicNum(음원번호)의 값이 num인 albumNum(앨범번호) attribute의 value를 select한다.

63. SELECT artistNum FROM music WHERE artistNum=num

music(음원) table에서 artistNum(아티스트번호)의 값이 num인 artistNum(아티스트번호) attribute의 value를 select한다.

64. SELECT artistNum FROM artist

artist(아티스트) table에서 artistNum(아티스트번호) attribute의 value를 select한다.