



Πολυτεχνική Σχολή
Τμήμα Μηχανικών Η/Υ & Πληροφορικής

«PROJECTΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ»

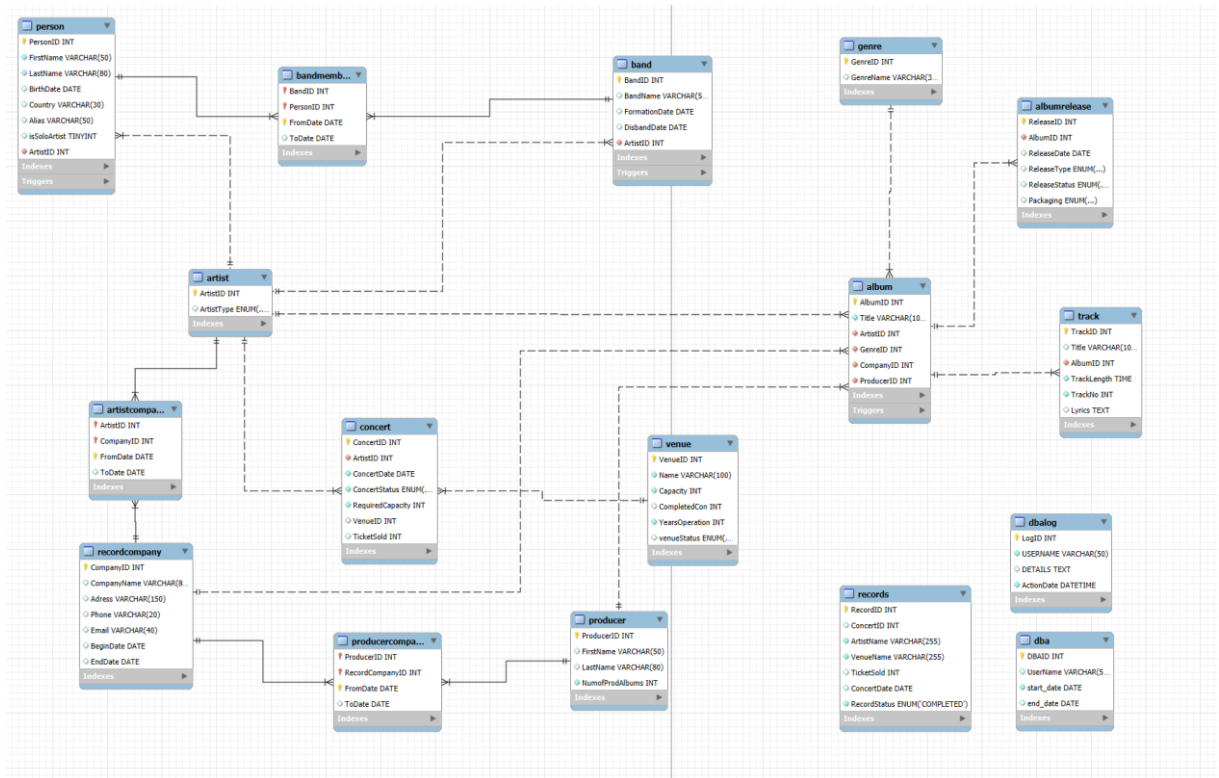
Νικόλαος Σελίμης
Α.Μ. 1100704

Χρήστος Σιάσιος
Α.Μ. 1088618

ΜΕΡΟΣ Α

ΚΕΦΑΛΑΙΟ 1

ΣΧΕΣΙΑΚΟ ΔΙΑΓΡΑΜΜΑ



Τα tables που δημιουργήθηκαν λόγω των νέων απαιτήσεων είναι τα concert, venue, records, dba και dbalog.

Οι εντολές create και insert των παλιών tables υπάρχουν στο τέλος της αναφοράς για ομοιομορφία.

Θα ακολουθήσει εξήγηση των νέων table καθώς και ο κώδικας σε sql που αφορά τη δημιουργία τους.

Ο πίνακας **concert** αφορά τις συναυλίες των καλλιτεχνών άρα το σύστημα θα αποθηκεύει έναν μοναδικό αύξοντα αριθμό για κάθε συναυλία, το ID του καλλιτέχνη που κάνει τη συναυλία, την ημερομηνία διεξαγωγής της συναυλίας και ένα Status που θα δείχνει αν η συναυλία είναι προγραμματισμένη, ακυρωμένη ή ολοκληρωμένη. Στο σύστημα θα δίνεται και η τιμή της απαιτούμενης χωρητικότητας της συναυλίας, το ID του χώρου που θα διεξάγεται καθώς και τα εισιτήρια που πουλήθηκαν.

```
CREATE TABLE IF NOT EXISTS concert (  
    ConcertID INT NOT NULL AUTO_INCREMENT,  
    ArtistID INT NOT NULL,  
    ConcertDate DATE NOT NULL,  
    ConcertStatus ENUM('SCHEDULED', 'COMPLETED', 'CANCELED') NOT NULL DEFAULT 'SCHEDULED',  
    RequiredCapacity INT NOT NULL DEFAULT 0,  
    VenueID INT DEFAULT NULL,  
    TicketSold INT ,  
    PRIMARY KEY (ConcertID),  
    INDEX (ArtistID),  
    INDEX (VenueID),  
    CONSTRAINT concert_venues FOREIGN KEY (VenueID)  
        REFERENCES venue(VenueID)  
        ON DELETE CASCADE ON UPDATE CASCADE,  
    CONSTRAINT concert_artists FOREIGN KEY (ArtistID)  
        REFERENCES artist(ArtistID)  
        ON DELETE CASCADE ON UPDATE CASCADE  
);
```

Ο πίνακας **venue** θα αποθηκεύει στο σύστημα έναν μοναδικό αύξοντα αριθμό για κάθε χώρο, το όνομα του χώρου διεξαγωγής, την χωρητικότητά του, έναν αριθμό των συναυλιών που έχουν διεξαχθεί στον κάθε χώρο και τα έτη λειτουργίας του και τέλος ένα status που θα δείχνει αν ο χώρος είναι ελεύθερος για συναυλία ή όχι.

```
CREATE TABLE IF NOT EXISTS venue(  
    VenueID INT AUTO_INCREMENT ,  
    Name VARCHAR(100)NOT NULL,  
    Capacity INT NOT NULL,  
    CompletedCon INT DEFAULT 0,  
    YearsOperation INT NOT NULL,  
    venueStatus ENUM('AVAILABLE', 'BOOKED') DEFAULT 'AVAILABLE',  
    PRIMARYKEY(VenueID));
```

Ο πίνακας **records** αποτελεί το ιστορικό των συναυλιών, στον οποίο καταχωρούνται οι ολοκληρωμένες συναυλίες, δηλαδή αποθηκεύει στο σύστημα το id κάθε συναυλίας, το όνομα του καλλιτέχνη που την πραγματοποιεί, το όνομα του χώρου στον οποίο λαμβάνει χώρα, το πλήθος των εισιτηρίων που εκδόθηκαν και την ημερομηνία διεξαγωγής της εκάστοτε συναυλίας.

```
CREATE TABLE IF NOT EXISTS records(  
    RecordID INT AUTO_INCREMENT,  
    ConcertID INT,  
    ArtistName VARCHAR(255) NOT NULL DEFAULT 'UNKNOWN',  
    VenueName VARCHAR(255) NOT NULL DEFAULT ' UNKNOWN',  
    TicketSold INT ,  
    ConcertDate DATE DEFAULT NULL,  
    RecordStatus ENUM('COMPLETED') NOT NULL,  
    PRIMARYKEY(RecordID)  
);
```

Το **dbatable** δημιουργήθηκε με βάση την περιγραφή του ερωτήματος 3.1.2.4 και έτσι δεν υπάρχει λόγος περιγραφής.

```
CREATE TABLE IF NOT EXISTS DBA(  
    DBAID INT AUTO_INCREMENT,  
    UserName VARCHAR(55),  
    start_date date NOT NULL,  
    end_date date,  
    PRIMARY KEY (DBAID)  
);
```

Το **dbalog** είναι ο πίνακας που καταγράφονται οι ενέργειες των διαχειριστών της βάσης. Το σύστημα θα αποθηκεύει έναν μοναδικό αύξοντα αριθμό για κάθε καταγεγραμμένη ενέργεια. Το usernametou διαχειριστή που εκτελεί την ενέργεια καταγράφεται καθώς και η περιγραφή της ενέργειας αλλά και η ακριβής ώρα και ημερομηνία που έγινε.

Δεν υπάρχει λόγος σύνδεσης των δύο table καθώς τα triggers που απαιτούνται δημιουργούνται με τρόπο τέτοιο ώστε να δίνεται το usernametou διαχειριστή χωρίς να εμπλακεί το DBA.

```
CREATE TABLE IF NOT EXISTS DBAlog(  
    LogID INT AUTO_INCREMENT,  
    USERNAME VARCHAR(50) NOT NULL,  
    DETAILS TEXT,  
    ActionDate DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,  
    PRIMARY KEY(LogID) );
```

ΚΕΦΑΛΑΙΟ 2

ΟΛΕΣ ΟΙ ΓΡΑΜΜΕΣ ΚΩΔΙΚΑ ΓΙΑ ΚΑΘΕ STORED ΘΑ ΥΠΑΡΧΟΥΝ ΣΤΟ ΤΕΛΟΣ ΤΗΣ ΑΝΑΦΟΡΑΣ.

3.1.3.1.

Αρχικά πρέπει να ελέγξουμε το περιεχόμενο του πίνακα venue με μια εντολή select.

VenueID	Name	Capacity	CompletedCon	YearsOperation	venueStatus
1	Stadium Karaiskakis	50000	100	15	AVAILABLE
2	OPAP Arena	30000	5	10	AVAILABLE
3	Kolokotronis	20000	8	20	AVAILABLE
4	Maracana	40000	7	12	AVAILABLE
5	Open	15000	4	8	AVAILABLE
NULL	NULL	NULL	NULL	NULL	NULL

Έπειτα καλούμε την storedprocedure και πρέπει να επιστρέψει τον βαθμό του χώρου διεξαγωγής όπως περιγράφεται.

call venuegrade(1);(50+3+30)

VenueGrade
83

call venuegrade(2); (30+0+20)

VenueGrade
50

call venuegrade(3); (20+0+40)

VenueGrade
60

call venuegrade(4); (40+0+24)

VenueGrade
64

call venuegrade(5); (15+0+16)

VenueGrade

31

Κώδικας:

DELIMITER \$

create procedure venuegrade (IN vID INT)

BEGIN

DECLARE grade INT;

DECLARE vCapacity int;

DECLARE vCompletedCon int;

DECLARE vYearsOperation int;

SELECT

Capacity, CompletedCon, YearsOperation

INTO

vCapacity, vCompletedCon, vYearsOperation

FROM venue

WHERE vID=VenueID;

SET grade=((vCapacity DIV 1000)*1 + (vCompletedCon DIV 100)*3 + vYearsOperation*2) ;

SELECT grade AS VenueGrade;

END \$

DELIMITER ;

3.1.3.2.

Αρχικά πρέπει να ελέγξουμε το περιεχόμενο του πίνακα concert με μια εντολή select.

ConcertID	ArtistID	ConcertDate	ConcertStatus	RequiredCapacity	VenueID	TicketSold
7	1	2024-06-15	SCHEDULED	25000	NULL	NULL
8	2	2024-07-20	CANCELED	15000	NULL	NULL
9	3	2024-08-10	SCHEDULED	5000	NULL	NULL
10	4	2024-09-05	SCHEDULED	30000	NULL	NULL
11	5	2024-10-01	SCHEDULED	10000	NULL	NULL

Το concertID δεν ξεκινάει από το 1 λόγω παλαιότερων δοκιμών που έχουν γίνει.

Ξεκινώντας τώρα τις νέες δοκιμές καλούμε την stored procedure ελέγχοντας τα 3 σενάρια για κάθε χαρακτήρα(i,c,a).

call CreateCancelRescheduleConcert(1,'2025-07-10','i');

message
I scheduled the concert as you requested.

ConcertID	ArtistID	ConcertDate	ConcertStatus	RequiredCapacity	VenueID	TicketSold
9	3	2024-08-10	SCHEDULED	5000	NULL	NULL
10	4	2024-09-05	SCHEDULED	30000	NULL	NULL
11	5	2024-10-01	SCHEDULED	10000	NULL	NULL
12	1	2025-07-10	SCHEDULED	0	NULL	NULL

Ξανακαλώντας τη με ακριβώς τα ίδια ορίσματα θα εμφανιστεί το μήνυμα:

call CreateCancelRescheduleConcert(1,'2025-07-10','i');

message
A concert is already scheduled

Αν καλέσουμε για canceled συναυλία θα εμφανίσει:

call CreateCancelRescheduleConcert(2,'2024-07-20','i');

message
A concert is canceled for that day.

Τώρα θα ακυρώσουμε την συναυλία που δημιουργήσαμε.

call CreateCancelRescheduleConcert(1,'2025-07-10','c');

message
The concert has been canceled.

ConcertID	ArtistID	ConcertDate	ConcertStatus	RequiredCapacity	VenueID	TicketSold
9	3	2024-08-10	SCHEDULED	5000	NULL	NULL
10	4	2024-09-05	SCHEDULED	30000	NULL	NULL
11	5	2024-10-01	SCHEDULED	10000	NULL	NULL
12	1	2025-07-10	CANCELED	0	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Αν προσπαθήσουμε να το ακυρώσουμε πάλι θα εμφανιστεί το μήνυμα:

```
call CreateCancelRescheduleConcert(1,'2025-07-10','c');
```

message
The concert has already been canceled.

Αν πάμε να ακυρώσουμε συναυλία που δεν υπάρχει καν θα εμφανιστεί το μήνυμα:

```
call CreateCancelRescheduleConcert(1,'2027-10-12','c');
```

message
There is no concert scheduled for that day.

Τώρα θα πάμε να ενεργοποιήσουμε την ακυρωμένη συναυλία.

```
call CreateCancelRescheduleConcert(1,'2025-07-10','a');
```

message
The concert has been rescheduled.

ConcertID	ArtistID	ConcertDate	ConcertStatus	RequiredCapacity	VenueID	TicketSold
9	3	2024-08-10	SCHEDULED	5000	NULL	NULL
10	4	2024-09-05	SCHEDULED	30000	NULL	NULL
11	5	2024-10-01	SCHEDULED	10000	NULL	NULL
12	1	2025-07-10	SCHEDULED	0	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Σε περίπτωση που η συναυλία που πάμε να ενεργοποιήσουμε δεν είναι canceled, θα εμφανίζεται:

```
call CreateCancelRescheduleConcert(1,'2025-07-10','a');
```

message
The concert is scheduled for that date.

Ενώ ενεργοποιώντας συναυλία που δεν υπάρχει καν:

```
call CreateCancelRescheduleConcert(1,'2027-10-12','a');
```

message
There is no canceled concert on that date.

Κώδικας :

DELIMITER \$

```
CREATE PROCEDURE CreateCancelRescheduleConcert(IN aID INT, IN conDATE DATE, IN action_type CHAR(1))
```

```
BEGIN
```



```
DECLARE flag INT;
DECLARE flag0 INT;
DECLARE flag1 INT;
DECLARE flag2 INT;
DECLARE flag3 INT;
DECLARE flag4 INT;
IF action_type = 'i' THEN
```

```
    SELECT COUNT(*) into flag
    FROM CONCERT
    where aID=ArtistID
    AND conDATE=ConcertDate
    AND ConcertStatus ='scheduled';
```

```
    SELECT COUNT(*) into flag0
    FROM CONCERT
    where aID=ArtistID
    AND conDATE=ConcertDate
    AND ConcertStatus ='canceled';
```

```
    IF flag>0 THEN
```

```
        SELECT 'A concert is already scheduled' AS message;
```

```
    ELSEIF flag0>0 THEN
```

```
        SELECT 'A concert is canceled for that day.' AS message;
```

```
    ELSE
```

```
        INSERT INTO concert (ArtistID, ConcertDate, ConcertStatus)
```

```
            VALUES (aID, conDATE, 'scheduled');
```

```
        SELECT 'I scheduled the concert as you requested.' AS message;
```

```
    END IF;
```

```
ELSEIF action_type = 'c' THEN
```

```
SELECT COUNT(*) into flag1
FROM CONCERT
where aID=ArtistID
AND conDATE=ConcertDate
AND ConcertStatus ='scheduled';
```

```
SELECT COUNT(*) into flag2
FROM CONCERT
where aID=ArtistID
AND conDATE=ConcertDate
AND ConcertStatus ='canceled';
```

```
IF flag1>0 THEN
    UPDATE concert
        SET ConcertStatus = 'canceled'
        WHERE ArtistID = aID
        AND ConcertDate = conDATE;
    SELECT 'The concert has been canceled. ' AS message;
```

```
ELSEIF flag2>0 THEN
    SELECT 'The concert has already been canceled. ' AS message;
```

```
ELSE
    SELECT 'There is no concert scheduled for that day. ' AS message;
END IF;
```

```
ELSEIF action_type = 'a' THEN
```

```
    SELECT COUNT(*) into flag3
    FROM CONCERT
    where aID=ArtistID
```

```
AND conDATE=ConcertDate
AND ConcertStatus ='canceled';
```

```
SELECT COUNT(*) into flag4
FROM CONCERT
where aID=ArtistID
AND conDATE=ConcertDate
AND ConcertStatus ='scheduled';
```

```
IF flag3>0 THEN
```

```
UPDATE concert
SET ConcertStatus ='scheduled'
WHERE ArtistID = aID
AND ConcertDate = conDATE;
SELECT 'The concert has been
```

```
rescheduled. ' AS message;
```

```
ELSEIF flag4>0 THEN
```

```
SELECT 'The concert is scheduled for that date.' AS message;
```

```
ELSE SELECT 'There is no canceled concert on that date.' AS
```

```
message;
```

```
END IF;
```

```
ELSE
```

```
SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalid action type provided. Use "i", "c", or "a".';
```

```
END IF;
```

```
END$
```

```
DELIMITER $
```

3.1.3.3.

Σε αυτή την stored procedure πρέπει να ορίσουμε 2 in παραμέτρους και 2 out.

Ας ελέγξουμε το concert table πρώτα

ConcertID	ArtistID	ConcertDate	ConcertStatus	RequiredCapacity	VenueID	TicketSold
7	1	2024-06-15	SCHEDULED	25000	NULL	NULL
8	2	2024-07-20	CANCELED	15000	NULL	NULL
9	3	2024-08-10	SCHEDULED	5000	NULL	NULL
10	4	2024-09-05	SCHEDULED	30000	NULL	NULL
11	5	2024-10-01	SCHEDULED	10000	NULL	NULL
12	1	2025-07-10	SCHEDULED	0	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Πρώτα θα καλέσουμε για μια συναυλία που δεν υπάρχει καν:

call GetConcertVenueDetails(1,2500,@venue,@cap);

VENUEID	CAPACITY
NULL	0

Αν είναι canceled:

call GetConcertVenueDetails(8,15000,@venue,@cap);

VENUEID	CAPACITY
NULL	0

Θα χρειαστεί να αλλάξουμε λίγο τα tables για να ελέγξουμε το ενδεχόμενο που έχει δοθεί ήδη venueID.

update concert

set venueID=1

where concertID=7;

ConcertID	ArtistID	ConcertDate	ConcertStatus	RequiredCapacity	VenueID	TicketSold
7	1	2024-06-15	SCHEDULED	25000	1	NULL
8	2	2024-07-20	CANCELED	15000	NULL	NULL
9	3	2024-08-10	SCHEDULED	5000	NULL	NULL
10	4	2024-09-05	SCHEDULED	30000	NULL	NULL
11	5	2024-10-01	SCHEDULED	10000	NULL	NULL
12	1	2025-07-10	SCHEDULED	0	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL

update venue

set venuestatus='BOOKED'

where venueID=1;

VenueID	Name	Capacity	CompletedCon	YearsOperation	venueStatus
1	Stadium Karaiskakis	50000	100	15	BOOKED
2	OPAP Arena	30000	5	10	AVAILABLE
3	Kolokotronis	20000	8	20	AVAILABLE
4	Maracana	40000	7	12	AVAILABLE
5	Open	15000	4	8	AVAILABLE
NULL	NULL	NULL	NULL	NULL	NULL

Τώρα call GetConcertVenueDetails(7,25000,@venue,@cap);

VENUEID	CAPACITY
1	50000

Και αν δεν έχει βρεθεί ακόμα VenueID:

call GetConcertVenueDetails(10,30000,@venue,@cap);

VENUEID	CAPACITY
4	40000

Κώδικας:

DELIMITER \$

```
CREATE PROCEDURE GetConcertVenueDetails(
```

```
    IN conID INT,
```

```
    IN requiredCapacity INT,
```

```
    OUT venueID INT,
```

```
    OUT venueCapacity INT )
```

```
BEGIN
```

```
DECLARE FLAG INT;
```

```
DECLARE Cstatus VARCHAR(50);
```

```
DECLARE vCapacity INT;
```

```
DECLARE vCompletedCon INT;
```

```
DECLARE vYearsOperation INT;
```

```
    SELECT COUNT(*)
```

```
    INTO FLAG
```

```
    FROM CONCERT
```

```
    WHERE conID=CONCERTID;
```

```
SELECT CONCERT.ConcertStatus
```

```
INTO Cstatus
```

```

FROM CONCERT
WHERE conID=CONCERTID;

IF FLAG=0 THEN
SET venueID = NULL;
    SET venueCapacity = 0;
SELECT venueID AS VENUEID, venueCapacity AS CAPACITY;

ELSEIF Cstatus='CANCELED' THEN
SET venueID = NULL;
    SET venueCapacity = 0;
SELECT venueID AS VENUEID, venueCapacity AS CAPACITY;

ELSEIF FLAG>0 THEN
SELECT concert.VenueID , venue.capacity
INTO venueID, venueCapacity
FROM concert
LEFT JOIN venue ON concert.venueID=venue.venueID
WHERE CONCERTID=conID;

IF venueID IS NOT NULL THEN
SELECT venueID AS VENUEID, venueCapacity AS CAPACITY;

ELSE
SELECT venue.venueID,venue.Capacity INTO venueID,venueCapacity
FROM venue
where capacity >=1.1*requiredCapacity AND venueStatus != 'BOOKED'
order by( (Capacity DIV 1000)*1 + (CompletedCon DIV 100)*3 + YearsOperation*2 ) DESC LIMIT 1;
SELECT venueID AS VENUEID, venueCapacity AS CAPACITY;
end if;
END IF;

END $
DELIMITER ;

```

3.1.3.4.

Σε αυτό το ερώτημα έχουμε δύο stored procedures, όπου η κάθε μία έχει δύο διαφορετικές υλοποιήσεις

α) Στο πρώτο stored procedure ορίζουμε 2 παραμέτρους εισόδου. Αυτή η procedure επιστρέφει τα ονόματα των καλλιτεχνών από τον πίνακα records, οι οποίοι έχουν πραγματοποιήσει συναυλίες, των οποίων το πλήθος των εισιτηρίων που εκδόθηκε βρίσκεται μεταξύ μιας ελάχιστης και μιας μέγιστης τιμής, οι οποίες ορίζονται ως είσοδοι στην κλήση της procedure .

DELIMITER \$

```
CREATE PROCEDURE GetArtistsByTicketRange(IN minTickets INT, IN maxTickets INT)
```

```
BEGIN
```

```
    SELECT DISTINCT ArtistName
```

```
    FROM records
```

```
    WHERE TicketSold BETWEEN minTickets AND maxTickets;
```

```
END $
```

DELIMITER ;

Καλούμε την CALL GetArtistsByTicketRange(20000, 30000);

Ο χρόνος εκτέλεσης αυτής της αναζήτησης αναγράφεται στο δεξιό μέρος της εικόνας. 0.125sec

17	20:16:58	CALL GetArtistsByTicketRange(20000, 30000)	5 row(s) returned	0.125 sec / 0.000 sec
----	----------	--	-------------------	-----------------------

Τώρα δημιουργούμε και ένα ευρετήριο που βασίζεται στο TicketSold και στο ArtistName προκειμένου να μας βοηθήσει στην επιτάχυνση της αναζήτησης.

```
CREATE INDEX idxticketsoldartist ON records (TicketSold, ArtistName);
```

Καλώντας πάλι την CALL GetArtistsByTicketRange(20000, 30000);

Παρατηρούμε ότι ο χρόνος **αναζήτησης έχει μειωθεί σχεδόν στο μισό** από την προηγούμενη υλοποίηση που ήταν χωρίς index. 0,78sec

20	20:23:55	CALL GetArtistsByTicketRange(20000, 30000)	5 row(s) returned	0.078 sec / 0.000 sec
----	----------	--	-------------------	-----------------------

*Το αποτέλεσμα είναι το ίδιο και στις 2 υλοποιήσεις.

β) Στο δεύτερο stored procedure ορίζουμε 1 παράμετρο εισόδου. Αυτή η procedure επιστρέφει τις ημερομηνίες των συναυλιών που έχουν λάβει χώρα σε ένα συγκεκριμένο χώρο, δίνοντας της το όνομα αυτού το χώρου.

DELIMITER \$

```
CREATE PROCEDURE GetConcertDatesByVenue(IN venue VARCHAR(255))
```

```
BEGIN
```

```
SELECT DISTINCT ConcertDate
```

```
FROM records
```

```
WHERE VenueName=venue;
```

```
END $
```

```
DELIMITER ;
```

Καλούμε την CALL GetConcertDatesByVenue('Stadium Karaiskakis');

Ο χρόνος εκτέλεσης αυτής της αναζήτησης αναγράφεται στο δεξιό μέρος της εικόνας. 0,031 sec

27 20:34:52 CALL GetConcertDatesByVenue('Stadium Karaiskakis') 3498 row(s) returned 0.031 sec / 0.016 sec

Τώρα δημιουργούμε και ένα ευρετήριο που βασίζεται στο VenueName προκειμένου να παρατηρήσουμε την διαφορά στο χρόνο ολοκλήρωσης της αναζήτησης.

```
CREATE INDEX idx_venue_name ON records (VenueName);
```

Καλώντας πάλι την CALLGetConcertDatesByVenue('StadiumKaraiskakis');

Παρατηρούμε ότι με την χρήση του ευρετηρίου ο χρόνος αυξάνεται 0,078sec γιατί παρόλο που γενικά το ευρετήριο αυξάνει την ταχύτητα αναζήτησης ειδικότερα σε μεγάλους πίνακες, όταν η τιμές στις οποίες βασίζεται το ευρετήριο είναι πολύ λίγες(στην συγκεκριμένη περίπτωση 1) τότε είναι πιθανό ο χρόνος αναζήτησης αντί να μειώνεται, να αυξάνεται.

29 20:36:13 CALL GetConcertDatesByVenue('Stadium Karaiskakis') 3498 row(s) returned 0.078 sec / 0.000 sec

*Το αποτέλεσμα είναι το ίδιο και στις 2 υλοποιήσεις.

ΚΕΦΑΛΑΙΟ 3

3.1.4.1.

Ξεκινάμε ελέγχοντας το περιεχόμενο του dbalog

LogID	USERNAME	DETAILS	ActionDate
1	admin_user	Created a new table named EmployeeDetails.	2025-01-16 10:51:47
2	db_manager	Updated records in the CustomerOrders table.	2025-01-16 10:51:47
3	sys_admin	Granted SELECT privileges to user analytics_team.	2025-01-16 10:51:47
4	dev_user	Dropped unused table named TempData.	2025-01-16 10:51:47
NULL	NULL	NULL	NULL

Έπειτα θα κάνουμε insert, delete και update σε κάθε table που ζητήθηκε.

PERSON

INSERT INTO artist(ArtistType)

VALUES('PERSON');

ArtistID	ArtistType
6	PERSON
7	PERSON
8	BAND
9	BAND
10	BAND
11	PERSON
NULL	NULL

INSERT INTO person (FirstName,LastName,BirthDate,Country,Alias,isSoloArtist,ArtistID)VALUES

('Chris','Siasios','2001-06-21','Greece','Pool',1,11);

8	Chris	Siasios	2001-06-21	Greece	Pool	1	11
---	-------	---------	------------	--------	------	---	----

update person

set firstname='John'

where personID=8;

delete from person

where personID=8;

LogID	USERNAME	DETAILS	ActionDate
2	db_manager	Updated records in the CustomerOrders table.	2025-01-16 11:07:45
3	sys_admin	Granted SELECT privileges to user analytics_team.	2025-01-16 11:07:45
4	dev_user	Dropped unused table named TempData.	2025-01-16 11:07:45
5	root@localhost	INSERT PERSON	2025-01-16 11:13:47
6	root@localhost	UPDATE PERSON	2025-01-16 11:17:10
7	root@localhost	DELETE PERSON	2025-01-16 11:17:39
NULL	NULL	NULL	NULL

BAND

```
INSERT INTO artist(ArtistType)
VALUES(BAND);
```

12	BAND
----	------

```
INSERT INTO band(BandName,FormationDate,DisbandDate,ArtistID)VALUES
("Boreia Asteria", '1995-01-01', NULL, 12);
```

4	Boreia Asteria	1995-01-01	NULL	12
---	----------------	------------	------	----

```
update band
set bandname='ZN'
WHERE BANDID=4;
```

```
delete from band
where BANDID=4;
```

LogID	USERNAME	DETAILS	ActionDate
5	root@localhost	INSERT PERSON	2025-01-16 11:13:47
6	root@localhost	UPDATE PERSON	2025-01-16 11:17:10
7	root@localhost	DELETE PERSON	2025-01-16 11:17:39
8	root@localhost	INSERT BAND	2025-01-16 11:22:39
9	root@localhost	UPDATE BAND	2025-01-16 11:25:37
10	root@localhost	DELETE BAND	2025-01-16 11:25:50

ALBUM

```
INSERT INTO album (Title,ArtistID,GenreID,CompanyID,ProducerID)VALUES
('GTK',1,2,1,4);
```

AlbumID	Title	ArtistID	GenreID	CompanyID	ProducerID
6	Silence	6	3	1	2
7	Rolling in The Deep	7	2	1	4
8	Ethereum	3	4	3	1
9	Twilight	6	1	4	5
10	Forza Horizon	5	5	5	6
11	GTK	1	2	1	4
NULL	NULL	NULL	NULL	NULL	NULL

```
update ALBUM
set TITLE='METRO'
WHERE ALBUMID=11;
```

delete from ALBUM
where ALBUMID=11;

LogID	USERNAME	DETAILS	ActionDate
8	root@localhost	INSERT BAND	2025-01-16 11:22:39
9	root@localhost	UPDATE BAND	2025-01-16 11:25:37
10	root@localhost	DELETE BAND	2025-01-16 11:25:50
11	root@localhost	INSERT ALBUM	2025-01-16 11:28:28
12	root@localhost	UPDATE ALBUM	2025-01-16 11:30:07
13	root@localhost	DELETE ALBUM	2025-01-16 11:30:20
NULL	NULL	NULL	NULL

VENUE

INSERT INTO venue (Name, Capacity,CompletedCon,YearsOperation)
VALUES('Lewforos', 17000, 1500, 50);

VenueID	Name	Capacity	CompletedCon	YearsOperation	venueStatus
1	Stadium Karaiskakis	50000	100	15	AVAILABLE
2	OPAP Arena	30000	5	10	AVAILABLE
3	Kolokotronis	20000	8	20	AVAILABLE
4	Maracana	40000	7	12	AVAILABLE
5	Open	15000	4	8	AVAILABLE
6	Lewforos	17000	1500	50	AVAILABLE
NULL	NULL	NULL	NULL	NULL	NULL

update venue
set CAPACITY=17001
WHERE venueID=6;

delete from venue
where venueID=6;

LogID	USERNAME	DETAILS	ActionDate
11	root@localhost	INSERT ALBUM	2025-01-16 11:28:28
12	root@localhost	UPDATE ALBUM	2025-01-16 11:30:07
13	root@localhost	DELETE ALBUM	2025-01-16 11:30:20
14	root@localhost	INSERT VENUE	2025-01-16 11:32:25
15	root@localhost	UPDATE VENUE	2025-01-16 11:34:01
16	root@localhost	DELETE VENUE	2025-01-16 11:34:01
NULL	NULL	NULL	NULL

CONCERT

```
INSERT INTO concert(ArtistID,ConcertDate,ConcertStatus,RequiredCapacity,VenueID)
VALUES(1,'2025-06-18','SCHEDULED',28000,NULL);
```

ConcertID	ArtistID	ConcertDate	ConcertStatus	RequiredCapacity	VenueID	TicketSold
1	1	2024-06-15	SCHEDULED	25000	NULL	NULL
2	2	2024-07-20	SCHEDULED	15000	NULL	NULL
3	3	2024-08-10	SCHEDULED	5000	NULL	NULL
4	4	2024-09-05	SCHEDULED	30000	NULL	NULL
5	5	2024-10-01	SCHEDULED	10000	NULL	NULL
6	1	2025-06-18	SCHEDULED	28000	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
update CONCERT
set RequiredCapacity=17000
WHERE CONCERTID=6;
```

```
delete from CONCERT
where CONCERTID=6;
```

LogID	USERNAME	DETAILS	ActionDate
14	root@localhost	INSERT VENUE	2025-01-16 11:32:25
15	root@localhost	UPDATE VENUE	2025-01-16 11:34:01
16	root@localhost	DELETE VENUE	2025-01-16 11:34:01
17	root@localhost	INSERT CONCERT	2025-01-16 11:37:37
18	root@localhost	UPDATE CONCERT	2025-01-16 11:39:01
19	root@localhost	DELETE CONCERT	2025-01-16 11:39:01
NULL	NULL	NULL	NULL

Κώδικας :

```
DELIMITER $

CREATE TRIGGER person_insert
AFTER INSERT ON person
FOR EACH ROW
BEGIN

INSERT INTO dbalog (USERNAME, DETAILS)
VALUES ( CURRENT_USER(), 'INSERT PERSON');

END$

CREATE TRIGGER person_delete
AFTER DELETE ON person
```

```
FOR EACH ROW
BEGIN

    INSERT INTO dbalog (USERNAME, DETAILS)
    VALUES ( CURRENT_USER(), 'DELETE PERSON');

END$
```

```
CREATE TRIGGER person_update
AFTER UPDATE ON person
FOR EACH ROW
BEGIN

    INSERT INTO dbalog (USERNAME, DETAILS)
    VALUES ( CURRENT_USER(), 'UPDATE PERSON');

END$
DELIMITER ;
```

```
DELIMITER $
CREATE TRIGGER band_insert
AFTER insert ON band
FOR EACH ROW
BEGIN

    INSERT INTO dbalog (USERNAME, DETAILS)
    VALUES ( CURRENT_USER(), 'INSERT BAND');

END$
```

```
CREATE TRIGGER band_delete
AFTER DELETE ON band
FOR EACH ROW
BEGIN

    INSERT INTO dbalog (USERNAME, DETAILS)
    VALUES ( CURRENT_USER(), 'DELETE BAND');

END$
```

```
CREATE TRIGGER band_update
AFTER UPDATE ON band
FOR EACH ROW
BEGIN

    INSERT INTO dbalog (USERNAME, DETAILS)
    VALUES ( CURRENT_USER(), 'UPDATE BAND');

END$
DELIMITER ;
```

```
DELIMITER $
CREATE TRIGGER album_insert
AFTER insert ON album
FOR EACH ROW
BEGIN

    INSERT INTO dbalog (USERNAME, DETAILS)
    VALUES ( CURRENT_USER(), 'INSERT ALBUM');

END$
```

```
CREATE TRIGGER album_delete
AFTER DELETE ON album
FOR EACH ROW
BEGIN

    INSERT INTO dbalog (USERNAME, DETAILS)
    VALUES ( CURRENT_USER(), 'DELETE ALBUM');

END$
```

```
CREATE TRIGGER album_update
AFTER UPDATE ON album
FOR EACH ROW
```

BEGIN

INSERT INTO dbalog (USERNAME, DETAILS)

VALUES (CURRENT_USER(), 'UPDATE ALBUM');

END\$

DELIMITER ;

DELIMITER \$

CREATE TRIGGER concert_insert

AFTER insert ON concert

FOR EACH ROW

BEGIN

INSERT INTO dbalog (USERNAME, DETAILS)

VALUES (CURRENT_USER(), 'INSERT CONCERT');

END\$

CREATE TRIGGER concert_delete

AFTER DELETE ON concert

FOR EACH ROW

BEGIN

INSERT INTO dbalog (USERNAME, DETAILS)

VALUES (CURRENT_USER(), 'DELETE CONCERT');

END\$

CREATE TRIGGER concert_update

AFTER UPDATE ON concert

FOR EACH ROW

BEGIN

INSERT INTO dbalog (USERNAME, DETAILS)

VALUES (CURRENT_USER(), 'UPDATE CONCERT');

END\$

DELIMITER ;

```
DELIMITER $  
  
CREATE TRIGGER venue_insert  
AFTER insert ON venue  
FOR EACH ROW  
BEGIN  
  
    INSERT INTO dbalog (USERNAME, DETAILS)  
    VALUES ( CURRENT_USER(), 'INSERT VENUE');  
  
END$
```

```
CREATE TRIGGER venue_delete  
AFTER DELETE ON venue  
FOR EACH ROW  
BEGIN  
  
    INSERT INTO dbalog (USERNAME, DETAILS)  
    VALUES ( CURRENT_USER(), 'DELETE VENUE');  
  
END$
```

```
CREATE TRIGGER venue_update  
AFTER UPDATE ON venue  
FOR EACH ROW  
BEGIN  
  
    INSERT INTO dbalog (USERNAME, DETAILS)  
    VALUES ( CURRENT_USER(), 'UPDATE VENUE');  
  
END$  
DELIMITER ;
```


3.1.4.2.

Όπως μπορεί κανείς να παρατηρήσει από τα παραπάνω triggers η δοκιμή γίνεται 2025-01-16, κάτι που πρέπει να αναφερθεί για την απόδειξη ομαλής λειτουργίας αυτών.

Το περιεχόμενο τουconcert είναι:

ConcertID	ArtistID	ConcertDate	ConcertStatus	RequiredCapacity	VenueID	TicketSold
1	1	2024-06-15	SCHEDULED	25000	NULL	NULL
2	2	2024-07-20	SCHEDULED	15000	NULL	NULL
3	3	2024-08-10	SCHEDULED	5000	NULL	NULL
4	4	2024-09-05	SCHEDULED	30000	NULL	NULL
5	5	2024-10-01	SCHEDULED	10000	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Αν δοκιμάσουμε να κάνουμε insert σε σωστή ημερομηνία τότε θα περάσει από το σύστημα:


```
INSERT INTO concert(ArtistID,ConcertDate,ConcertStatus,RequiredCapacity,VenueID)
VALUES(1,'2025-06-18','SCHEDULED',28000,NULL);
```

ConcertID	ArtistID	ConcertDate	ConcertStatus	RequiredCapacity	VenueID	TicketSold
1	1	2024-06-15	SCHEDULED	25000	NULL	NULL
2	2	2024-07-20	SCHEDULED	15000	NULL	NULL
3	3	2024-08-10	SCHEDULED	5000	NULL	NULL
4	4	2024-09-05	SCHEDULED	30000	NULL	NULL
5	5	2024-10-01	SCHEDULED	10000	NULL	NULL
7	1	2025-06-18	SCHEDULED	28000	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Το ID πάει 7 αφού το 6 είχε γίνει στη δοκιμή του 3.1.4.1 το έχουμε διαγράψει.

Αν προσπαθήσουμε να προγραμματίσουμε σε διάστημα μικρότερο ή ίσο των 5 ημερών θα εμφανιστεί error

```
INSERT INTO concert(ArtistID,ConcertDate,ConcertStatus,RequiredCapacity,VenueID)
VALUES(1,'2025-01-21','SCHEDULED',28000,NULL);
```

 204 11:49:46 INSERT INTO concert(ArtistID,ConcertDate,ConcertStatus,RequiredCapacity,VenueID) VAL... Error Code: 1644. Concert must be scheduled at least 5 days before ConcertDate.

```
INSERT INTO concert(ArtistID,ConcertDate,ConcertStatus,RequiredCapacity,VenueID)
VALUES(1,'2025-01-22','SCHEDULED',28000,NULL);
```

ConcertID	ArtistID	ConcertDate	ConcertStatus	RequiredCapacity	VenueID	TicketSold
1	1	2024-06-15	SCHEDULED	25000	NULL	NULL
2	2	2024-07-20	SCHEDULED	15000	NULL	NULL
3	3	2024-08-10	SCHEDULED	5000	NULL	NULL
4	4	2024-09-05	SCHEDULED	30000	NULL	NULL
5	5	2024-10-01	SCHEDULED	10000	NULL	NULL
7	1	2025-06-18	SCHEDULED	28000	NULL	NULL
8	1	2025-01-22	SCHEDULED	28000	NULL	NULL

Αν προγραμματίσουμε παραπάνω από 3 συναυλίες για τον artist με ID=1 τότε:

```
INSERT INTO concert(ArtistID,ConcertDate,ConcertStatus,RequiredCapacity,VenueID)
VALUES(1,'2025-01-23','SCHEDULED',28000,NULL);
```

✖ 207 11:52:21 INSERT INTO concert(ArtistID,ConcertDate,ConcertStatus,RequiredCapacity,VenueID) VAL... Error Code: 1644. This artist has 3 scheduled concerts already!

DELIMITER \$

CREATE TRIGGER NewConcert

BEFORE INSERT ON concert

FOR EACH ROW

BEGIN

DECLARE ConcertCount INT;

IF DATEDIFF(NEW.ConcertDATE , CURDATE()) <=5

THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE_TEXT ='Concert must be scheduled at least 5 days before ConsertDate.';

END IF;

SELECT COUNT(*) INTO ConcertCount

FROM concert

WHERE concertstatus='Scheduled'

AND artistID = NEW.artistID;

IF (ConcertCount>=3) THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE_TEXT ='This artist has 3 scheduled concerts already!';

END IF;

END\$

DELIMITER ;

3.1.4.3.

Θα κάνω drop το trigger που δημιουργήσαμε στο παραπάνω ερώτημα για να μην εμποδίσει τις δοκιμές.

Το περιεχόμενο του concert είναι:

ConcertID	ArtistID	ConcertDate	ConcertStatus	RequiredCapacity	VenueID	TicketSold
1	1	2024-06-15	SCHEDULED	25000	NULL	NULL
2	2	2024-07-20	SCHEDULED	15000	NULL	NULL
3	3	2024-08-10	SCHEDULED	5000	NULL	NULL
4	4	2024-09-05	SCHEDULED	30000	NULL	NULL
5	5	2024-10-01	SCHEDULED	10000	NULL	NULL
7	1	2025-06-18	SCHEDULED	28000	NULL	NULL
8	1	2025-01-22	SCHEDULED	28000	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL

UPDATE CONCERT

SET CONCERTSTATUS='CANCELED'

WHERE CONCERTID=8;

8	1	2025-01-22	CANCELED	28000	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Δημιουργούμε μια συναυλία με ημερομηνία 18/01(<3 μέρες από σήμερα):

INSERT INTO concert(ArtistID,ConcertDate,ConcertStatus,RequiredCapacity,VenueID)

VALUES(1,'2025-01-18','SCHEDULED',28000,NULL);

8	1	2025-01-22	CANCELED	28000	NULL	NULL
9	1	2025-01-18	SCHEDULED	28000	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Και την ακυρώνουμε:

UPDATE CONCERT

SET CONCERTSTATUS='CANCELED'

WHERE CONCERTID=9;

219 12:03:58 UPDATE CONCERT SET CONCERTSTATUS='CANCELED' WHERE CONCERTID=9 Error Code: 1644. Concert cancellations must be made at least 3 days prior to the event.

Αυτή τη στιγμή ο artist με id=1 έχει 3 προγραμματισμένες συναυλίες

ConcertID	ArtistID	ConcertDate	ConcertStatus	RequiredCapacity	VenueID	TicketSold
1	1	2024-06-15	SCHEDULED	25000	NULL	NULL
2	2	2024-07-20	SCHEDULED	15000	NULL	NULL
3	3	2024-08-10	SCHEDULED	5000	NULL	NULL
4	4	2024-09-05	SCHEDULED	30000	NULL	NULL
5	5	2024-10-01	SCHEDULED	10000	NULL	NULL
7	1	2025-06-18	SCHEDULED	28000	NULL	NULL
8	1	2025-01-22	CANCELED	28000	NULL	NULL
9	1	2025-01-18	SCHEDULED	28000	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Αν πάμε να κάνουμε reschedule τη συναυλία με id=8 τότε θα εμφανιστεί:

```
UPDATE CONCERT
SET CONCERTSTATUS='SCHEDULED'
WHERE CONCERTID=8;
```

222 12:07:36 UPDATE CONCERT SET CONCERTSTATUS='SCHEDULED' WHERE CONCERTID=8 Error Code: 1644. This artist has 3 scheduled concerts already!

```
DELIMITER $
```

```
CREATE TRIGGER CancelRescheduleConcert
BEFORE UPDATE ON Concert
FOR EACH ROW
BEGIN
DECLARE ConcertCount INT;
IF OLD.ConcertStatus ='Canceled' AND NEW.ConcertStatus ='Scheduled'
THEN
SELECT COUNT(*) INTO ConcertCount
FROM concert
WHERE ConcertStatus ='Scheduled'
AND artistID = NEW.artistID;

IF (ConcertCount>=3) THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT ='This artist has 3 scheduled concerts already!';
END IF;
END IF;

IF OLD.ConcertStatus ='Scheduled' AND NEW.ConcertStatus ='Canceled'
THEN
IF DATEDIFF(OLD.concertDate, CURDATE()) < 3 THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Concert cancellations must be made at least 3 days prior to the event.';
END IF;
END IF;

END$

DELIMITER ;
```

ΚΕΦΑΛΑΙΟ 4

3.2

3.2.1

Περιγραφή του προγράμματος στο Netbeans

Με την χρήση του των εργαλείων του Netbeans έχει σχεδιαστεί το αρχικό interface. Σε αυτό το interface έχει τοποθετηθεί στο πάνω μέρος της σελίδας ένα label, σαν τίτλος που δείχνει στον χρήστη τι είναι τα παρακάτω κουμπιά. Στα παρακάτω κουμπιά έχουν μπει τα ονόματα όλων των πινάκων του database.

- 1) Ο χρήστης μπορεί να πατήσει σε οποιοδήποτε από αυτά τα κουμπιά ανάλογα με τον πίνακα, του οποίου θέλει να εμφανίσει τα περιεχόμενα του.
- 2) Επιλέγοντας τον πίνακα που επιθυμεί και κάνοντας κλικ στο κατάλληλο κουμπί, εμφανίζεται μέσω του κώδικα που έχει γραφτεί σε java ένα νέο παράθυρο με τα περιεχόμενα του εκάστοτε πίνακα.
- 3) Στο κάτω μέρος αυτού του παραθύρου εμφανίζονται 4 επιλογές (Insert, update, delete, ok), τις οποίες μπορεί να επιλέξει ο χρήστης ανάλογα με την ενέργεια που επιθυμεί να κάνει.
- 4) Πατώντας το Insert του εμφανίζεται ένα μικρότερο παράθυρο που του γράφει ποιο στοιχείο πρέπει να εισάγει ανάλογα με τον πίνακα, πατώντας το Ok σε αυτό το παράθυρο εμφανίζεται ένα άλλο που του ζητάει να εισάγει κάποιο άλλο στοιχείο και αυτή η διαδικασία συνεχίζει μέχρι να συμπληρωθούν όλα τα απαραίτητα στοιχεία και μετά εμφανίζεται πάλι ο πίνακας όπως είχε εμφανιστεί με τα στοιχεία του από πριν, πατώντας ok ο χρήστης βλέπει πάλι το αρχικό interface στο οποίο μπορεί να επιλέξει όποιο πίνακα θέλει για να κάνει την ίδια διαδικασία. Αν θέλει να δει την αλλαγή που έχει κάνει στο πίνακα, πατάει ξανά στο κουμπί του πίνακα και του εμφανίζεται ο ανανεωμένος πίνακας. Πατώντας το κουμπί του exit κάτω δεξιά βγαίνει από το interface.
- 5) 6) Αντίστοιχα με το insert γίνεται και το update και delete.

- Αυτό το μέρος του κώδικα κάνει import τις κατάλληλες βιβλιοθήκες και συνδέει το database που χρησιμοποιούμε στην SQL με το κώδικα που θα χρησιμοποιήσουμε για το interface.

```
import java.sql.*;
```

```
import java.awt.*;
```

```
import javax.swing.*;
```

```
public class NewJFrame extends javax.swing.JFrame {
```

```
    private static Connection con;
```

```
    public NewJFrame() {
```

```
        try {
```

```
            con =
```

```
            DriverManager.getConnection("jdbc:mysql://localhost:3306/musicindustry?useSSL=false&allowPublicKeyRetrieval=true",  
            "root", "12345");
```

```
            System.out.println("Database connection successful.");
```

```
        } catch (SQLException e) {
```

```
        System.err.println("Connection failed: " + e.getMessage());
    }
    initComponents();
}
```

1) Κώδικας που έχει φτιάξει το Netbeans με βάση το design του πρώτου παραθύρου

```
// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

    jInternalFrame1 = new javax.swing.JInternalFrame();
    jButton19 = new javax.swing.JButton();
    jLabel1 = new javax.swing.JLabel();
    jButton1 = new javax.swing.JButton();
    jButton2 = new javax.swing.JButton();
    jButton3 = new javax.swing.JButton();
    jButton4 = new javax.swing.JButton();
    jButton5 = new javax.swing.JButton();
    jButton6 = new javax.swing.JButton();
    jButton7 = new javax.swing.JButton();
    jButton8 = new javax.swing.JButton();
    jButton9 = new javax.swing.JButton();
    jButton10 = new javax.swing.JButton();
    jButton11 = new javax.swing.JButton();
    jButton12 = new javax.swing.JButton();
    jButton13 = new javax.swing.JButton();
    jButton14 = new javax.swing.JButton();
    jButton15 = new javax.swing.JButton();
    jButton16 = new javax.swing.JButton();
    jButton17 = new javax.swing.JButton();
    jButton18 = new javax.swing.JButton();

    jInternalFrame1.setVisible(true);

    jButton19.setText("jButton19");

    javax.swing.GroupLayout jInternalFrame1Layout = new javax.swing.GroupLayout(jInternalFrame1.getContentPane());
    jInternalFrame1.getContentPane().setLayout(jInternalFrame1Layout);
    jInternalFrame1Layout.setHorizontalGroup(
        jInternalFrame1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jInternalFrame1Layout.createSequentialGroup()
                .addContainerGap(235, Short.MAX_VALUE)

```

```

        .addComponent(jButton19)
        .addGap(216, 216, 216))
);

jInternalFrame1Layout.setVerticalGroup(
    jInternalFrame1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jInternalFrame1Layout.createSequentialGroup()
            .addGap(54, 54, 54)
            .addComponent(jButton19)
            .addContainerGap(153, Short.MAX_VALUE))
        );

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
setBackground(new java.awt.Color(255, 51, 51));
setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
setForeground(new java.awt.Color(204, 51, 0));
setMaximizedBounds(new java.awt.Rectangle(0, 0, 0, 0));

jLabel1.setBackground(new java.awt.Color(0, 0, 0));
jLabel1.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
jLabel1.setForeground(new java.awt.Color(51, 0, 0));
jLabel1.setText("Πίνακας");
jLabel1.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));
jLabel1.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);

jButton1.setText("album");
jButton1.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        jButton1MouseClicked(evt);
    }
});

jButton2.setText("albumrelease");
jButton2.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        jButton2MouseClicked(evt);
    }
});

jButton3.setText("artist");
jButton3.addMouseListener(new java.awt.event.MouseAdapter() {

```

```

        public void mouseClicked(java.awt.event.MouseEvent evt) {
            jButton3MouseClicked(evt);
        }
    });

    jButton4.setText("artistcompany");
    jButton4.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent evt) {
            jButton4MouseClicked(evt);
        }
    });

    jButton5.setText("band");
    jButton5.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent evt) {
            jButton5MouseClicked(evt);
        }
    });

    jButton6.setText("bandmember");
    jButton6.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent evt) {
            jButton6MouseClicked(evt);
        }
    });

    jButton7.setText("concert");
    jButton7.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent evt) {
            jButton7MouseClicked(evt);
        }
    });

    jButton8.setText("dba");
    jButton8.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent evt) {
            jButton8MouseClicked(evt);
        }
    });

```



```
jButton9.setText("dbalog");
jButton9.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        jButton9MouseClicked(evt);
    }
});

jButton10.setText("genre");
jButton10.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        jButton10MouseClicked(evt);
    }
});

jButton11.setText("person");
jButton11.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        jButton11MouseClicked(evt);
    }
});

jButton12.setText("producer");
jButton12.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton12ActionPerformed(evt);
    }
});

jButton13.setText("producercompany");
jButton13.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        jButton13MouseClicked(evt);
    }
});

jButton14.setText("recordcompany");
jButton14.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        jButton14MouseClicked(evt);
    }
});
```

```
});
```

```
jButton15.setText("records");  
jButton15.addMouseListener(new java.awt.event.MouseAdapter() {  
    public void mouseClicked(java.awt.event.MouseEvent evt) {  
        jButton15MouseClicked(evt);  
    }  
});
```

```
jButton16.setText("track");  
jButton16.addMouseListener(new java.awt.event.MouseAdapter() {  
    public void mouseClicked(java.awt.event.MouseEvent evt) {  
        jButton16MouseClicked(evt);  
    }  
});
```

```
jButton17.setText("venue");  
jButton17.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        jButton17ActionPerformed(evt);  
    }  
});
```

```
jButton18.setText("Exit");  
jButton18.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        jButton18ActionPerformed(evt);  
    }  
});
```

```
javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());  
getContentPane().setLayout(layout);  
layout.setHorizontalGroup(  
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
        .addGroup(layout.createSequentialGroup()  
            .addGap(47, 47, 47)  
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
                .addComponent(jButton3, javax.swing.GroupLayout.PREFERRED_SIZE, 109, javax.swing.GroupLayout.PREFERRED_SIZE)  
                .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 109, javax.swing.GroupLayout.PREFERRED_SIZE)  
                .addComponent(jButton4, javax.swing.GroupLayout.PREFERRED_SIZE, 109, javax.swing.GroupLayout.PREFERRED_SIZE)  
            )  
        )  
);
```

```
.addComponent(jButton5, javax.swing.GroupLayout.PREFERRED_SIZE, 109, javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jButton6, javax.swing.GroupLayout.PREFERRED_SIZE, 109, javax.swing.GroupLayout.PREFERRED_SIZE))

.addGap(80, 80, 80)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

.addComponent(jButton7, javax.swing.GroupLayout.PREFERRED_SIZE, 109, javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jButton8, javax.swing.GroupLayout.PREFERRED_SIZE, 109, javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jButton9, javax.swing.GroupLayout.PREFERRED_SIZE, 109, javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jButton10, javax.swing.GroupLayout.PREFERRED_SIZE, 109,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addComponent(jButton11, javax.swing.GroupLayout.PREFERRED_SIZE, 109,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 109, javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 109, javax.swing.GroupLayout.PREFERRED_SIZE))

.addGap(80, 80, 80)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jButton13, javax.swing.GroupLayout.PREFERRED_SIZE, 109,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addComponent(jButton12, javax.swing.GroupLayout.PREFERRED_SIZE, 109,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addComponent(jButton14, javax.swing.GroupLayout.PREFERRED_SIZE, 109,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addComponent(jButton15, javax.swing.GroupLayout.PREFERRED_SIZE, 109,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addComponent(jButton16, javax.swing.GroupLayout.PREFERRED_SIZE, 109,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addContainerGap(47, Short.MAX_VALUE))

.addGroup(layout.createSequentialGroup()

.addGap(236, 236, 236)

.addComponent(jButton17, javax.swing.GroupLayout.PREFERRED_SIZE, 109, javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)

.addComponent(jButton18, javax.swing.GroupLayout.PREFERRED_SIZE, 72, javax.swing.GroupLayout.PREFERRED_SIZE)

.addGap(19, 19, 19))

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(24, 24, 24)

.addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 33, javax.swing.GroupLayout.PREFERRED_SIZE)

.addGap(26, 26, 26)

.addComponent(jButton1)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
```

```

        .addComponent(jButton2)
        .addComponent(jButton7)
        .addComponent(jButton12))
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
        .addComponent(jButton3)
        .addComponent(jButton8)
        .addComponent(jButton13))
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
        .addComponent(jButton4)
        .addComponent(jButton9)
        .addComponent(jButton14))
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
        .addComponent(jButton5)
        .addComponent(jButton10)
        .addComponent(jButton15))
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
        .addComponent(jButton11)
        .addComponent(jButton6)
        .addComponent(jButton16))
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addComponent(jButton17)
    .addGap(0, 23, Short.MAX_VALUE)
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.TRAILING), layout.createSequentialGroup()
        .addComponent(jButton18)
        .addComponent(jButton1))
    .addContainerGap()

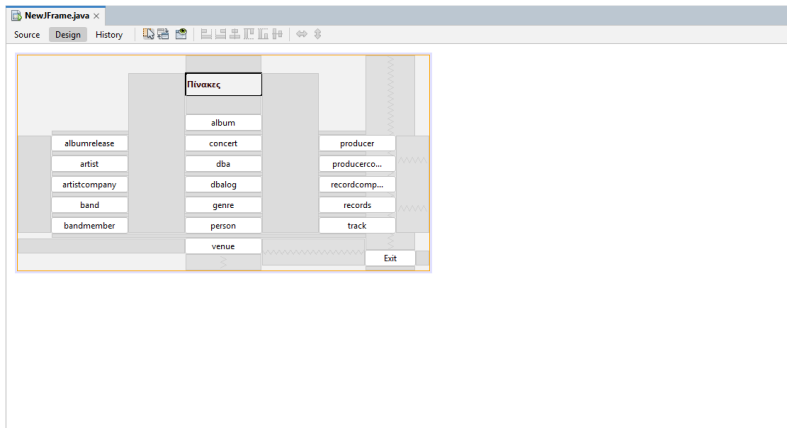
);

pack();
} // </editor-fold>

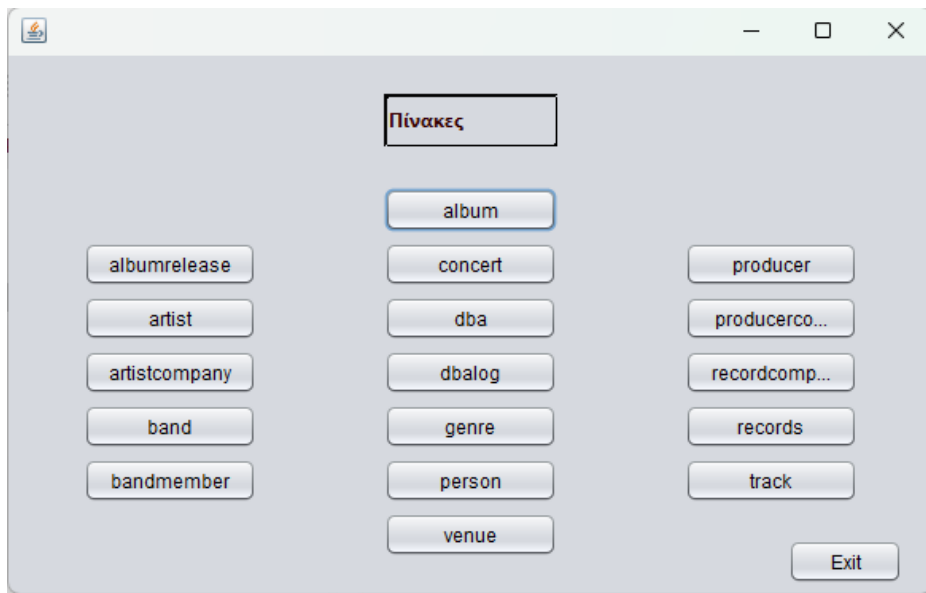
```

***Τέλος κώδικα που έχει φτιάξει το Netbeans

- Αυτό είναι το design του αρχικού interface



Αυτό είναι το αρχικό interface στο οποίο τα κουμπιά δεν έχουν καμία ουσιαστική χρήση χωρίς τον παρακάτω κώδικα :



- Σε αυτό το κομμάτι του κώδικα δίνονται ουσιαστικά οι κατάλληλες εντολές στο πρόγραμμα προκειμένου τα κουμπιά να είναι χρηστικά και να κάνουν τις λειτουργίες που πρέπει (παράδειγμα ενός κουμπιού, με την ίδια λογική δουλεύουν και τα υπόλοιπα).

```
private void jButton1MouseClicked(java.awt.event.MouseEvent evt) {
    JFrame frame=new JFrame(" Album Table Management");
    frame.setSize(600,400);
    frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    frame.setLayout(new BorderLayout());

    JTextArea textArea=new JTextArea();
    textArea.setEditable(false);
```

```

try{ String query="SELECT * FROM album";

    Statement stmt=connection.createStatement();

    ResultSet rs=stmt.executeQuery(query);

    StringBuilder res=new StringBuilder(" Albums:\n");

    while(rs.next()){

        res.append("ID: ").append(rs.getInt("AlbumID"))

        .append(", Title: ").append(rs.getString("Title"))

        .append(", ArtistID: ").append(rs.getString("ArtistID"))

        .append(", GenreID: ").append(rs.getString("GenreID"))

        .append(", CompanyID: ").append(rs.getString("CompanyID"))

        .append(", ProducerID: ").append(rs.getString("ProducerID"))

        .append("\n");

    }

    textArea.setText(res.toString());

    rs.close();

    stmt.close();

}

catch(SQLException exc){

    JOptionPane.showMessageDialog(this,"Error executing query: "+exc.getMessage(),"Database Error",JOptionPane.ERROR_MESSAGE);

    return;

}

JScrollPane scrollPane=new JScrollPane(textArea);

frame.add(scrollPane, BorderLayout.CENTER);


JPanel buttpanel=new JPanel();

JButton insertbutton=new JButton("Insert");

JButton updatebutton=new JButton("Update");

JButton deletebutton=new JButton("Delete");

JButton okbutton=new JButton("OK");


buttpanel.add(insertbutton);

buttpanel.add(updatebutton);

buttpanel.add(deletebutton);

buttpanel.add(okbutton);

frame.add(buttpanel, BorderLayout.SOUTH);


insertbutton.addActionListener(exc->insertAlbum());

```

```
updatebutton.addActionListener(exc->updateAlbum());
deletebutton.addActionListener(exc->deleteAlbum());
```

```
okbutton.addActionListener(exc->{ frame.dispose();
    new JFrame().setVisible(true);
});
frame.setVisible(true);
}
```

```
private boolean isValidForeignKey(String tableName,String columnName,String value) throws SQLException{
    String query="SELECT COUNT(*) FROM "+tableName+" WHERE "+columnName+" = ?";
    PreparedStatement pstmt=connection.prepareStatement(query);
    pstmt.setInt(1,Integer.parseInt(value));
    ResultSet rs=pstmt.executeQuery();
    rs.next();
    return rs.getInt(1)>0;
}
```

```
private void insertAlbum(){
    String title=JOptionPane.showInputDialog(this,"Enter Title:");
    String artistID=JOptionPane.showInputDialog(this,"Enter ArtistID:");
    String genreID=JOptionPane.showInputDialog(this,"Enter GenreID:");
    String companyID=JOptionPane.showInputDialog(this,"Enter CompanyID:");
    String producerID=JOptionPane.showInputDialog(this,"Enter ProducerID:");

    if(title==null||artistID==null||genreID==null||companyID==null||producerID==null||
        title.trim().isEmpty()||artistID.trim().isEmpty()||genreID.trim().isEmpty()||companyID.trim().isEmpty()||producerID.trim().isEmpty()){
        JOptionPane.showMessageDialog(this,"All fields are required!","Input Error",JOptionPane.ERROR_MESSAGE);
        return;
    }

    try{ String query="INSERT INTO album (Title, ArtistID, GenreID, CompanyID, ProducerID) VALUES (?, ?, ?, ?, ?)";
        PreparedStatement pstmt=connection.prepareStatement(query);
        pstmt.setString(1,title);
        pstmt.setInt(2,Integer.parseInt(artistID));
        pstmt.setInt(3,Integer.parseInt(genreID));
        pstmt.setInt(4,Integer.parseInt(companyID));
        pstmt.setInt(5,Integer.parseInt(producerID));
        pstmt.executeUpdate();

        JOptionPane.showMessageDialog(this,"Album inserted successfully!");
    }
```

```

    }

    catch(SQLException exc){

        JOptionPane.showMessageDialog(this,"Error inserting album: "+exc.getMessage(),"Database
        Error",JOptionPane.ERROR_MESSAGE);

    }

}

private void updateAlbum(){

    String albumID=JOptionPane.showInputDialog(this,"Enter AlbumID to update:");

    String newTitle=JOptionPane.showInputDialog(this,"Enter new Title:");

    if(albumID==null||newTitle==null||albumID.trim().isEmpty()||newTitle.trim().isEmpty()){

        JOptionPane.showMessageDialog(this,"All fields are required!", "Input Error",JOptionPane.ERROR_MESSAGE);

        return;

    }

    try{ String query="UPDATE album SET Title = ? WHERE AlbumID = ?";

        PreparedStatement pstmt=connection.prepareStatement(query);

        pstmt.setString(1,newTitle);

        pstmt.setInt(2,Integer.parseInt(albumID));

        pstmt.executeUpdate();

        JOptionPane.showMessageDialog(this,"Album updated successfully!");

    } catch(SQLException exc){

        JOptionPane.showMessageDialog(this,"Error updating album: "+exc.getMessage(),"Database
        Error",JOptionPane.ERROR_MESSAGE);

    }

}

private void deleteAlbum(){

    String albumID=JOptionPane.showInputDialog(null,"Enter AlbumID to delete:");

    if(albumID==null||albumID.trim().isEmpty()){

        JOptionPane.showMessageDialog(null,"AlbumID is required to delete!", "Input Error",JOptionPane.ERROR_MESSAGE);

        return;

    }

    try{ String query="DELETE FROM album WHERE AlbumID = ?";

        PreparedStatement pstmt=connection.prepareStatement(query);

        pstmt.setInt(1,Integer.parseInt(albumID));

        pstmt.executeUpdate();

        JOptionPane.showMessageDialog(null,"Album deleted successfully!");

    } catch(SQLException exc){

        JOptionPane.showMessageDialog(null,"Error deleting album: "+exc.getMessage(),"Database
        Error",JOptionPane.ERROR_MESSAGE);

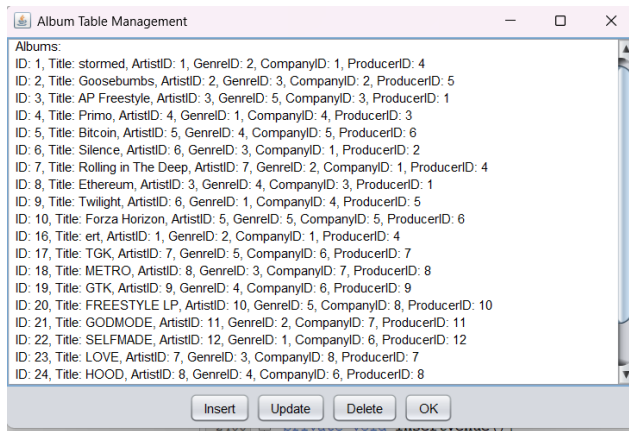
    }

}

```

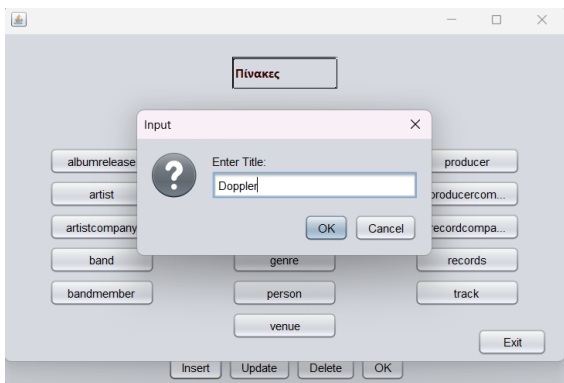

}

- Αφού έχουμε ενεργοποιήσει τα κουμπιά, θα κάνουμε κλικ στο κουμπί του album και εμφανίζεται το interface που έχουμε περιγράψει στην αρχή του κεφαλαίου.

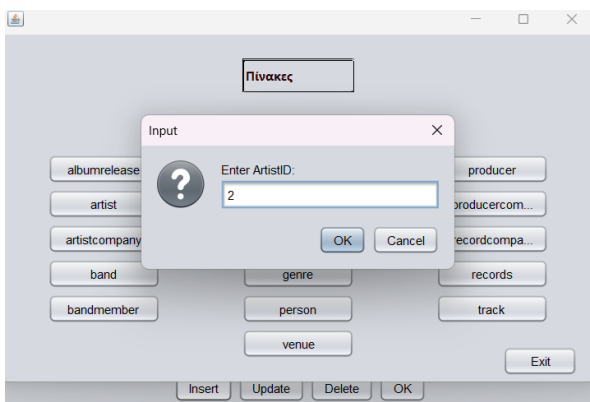


- Κάνουμε ένα **Insert** στο album.

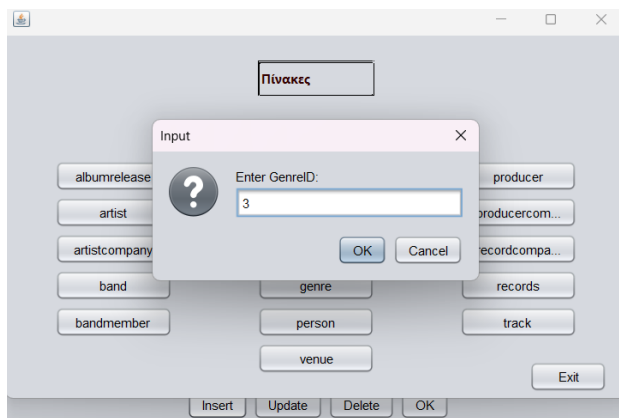
Πρώτα δίνουμε τον τίτλο του album.



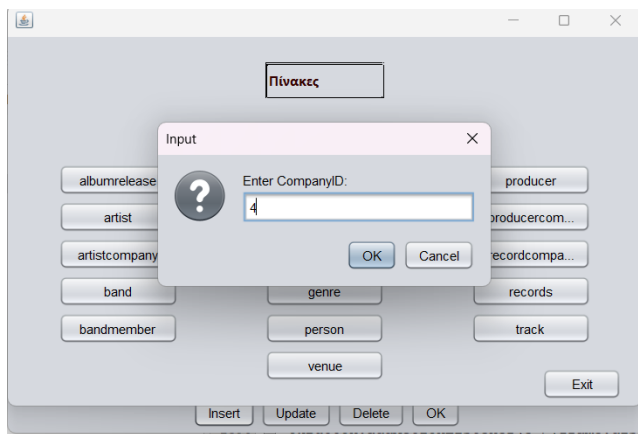
- Δίνουμε το ArtistID



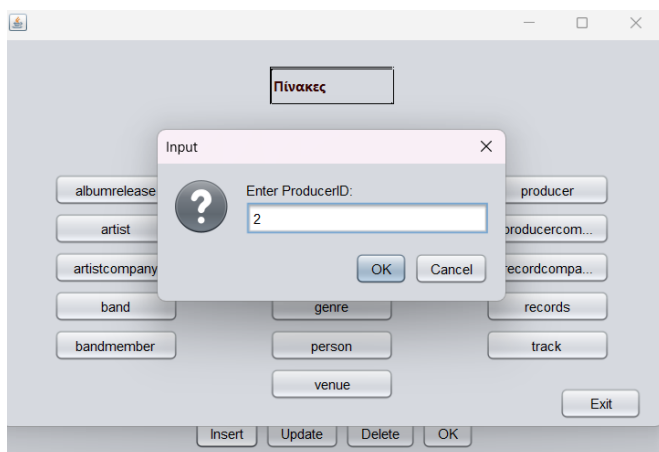
- Δίνουμε το GenreID



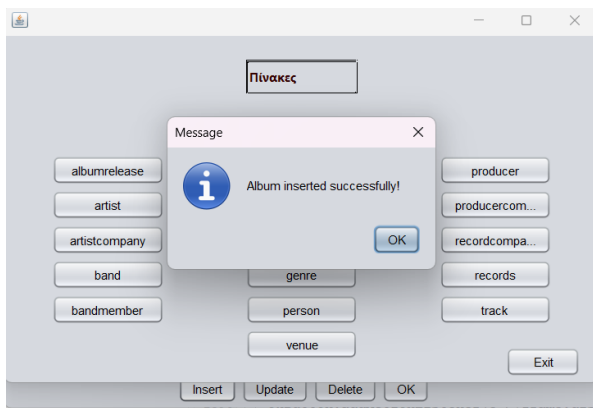
- Δίνουμε το CompanyID



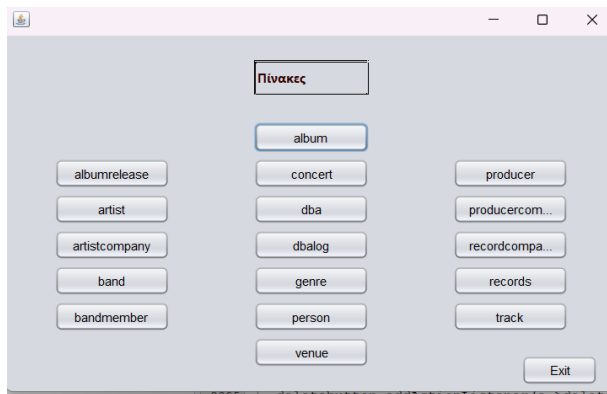
- Δίνουμε το ProducerID



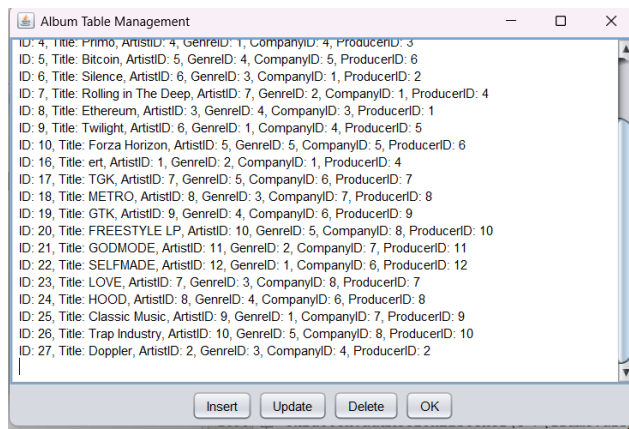
- Μόλις συμπληρωθούν όλα τα απαραίτητα στοιχεία εμφανίζεται μήνυμα ότι έχει γίνει το Insert



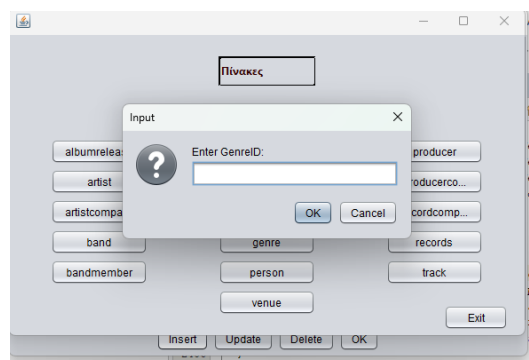
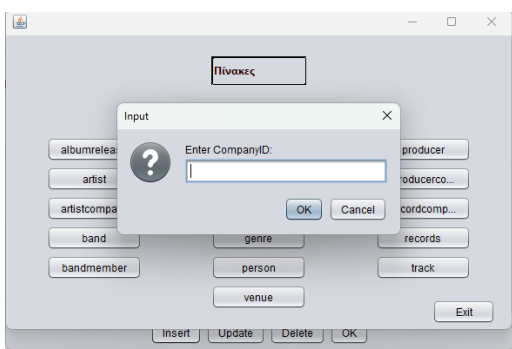
- Εμφανίζεται πάλι το αρχικό μενού και για να δούμε την αλλαγή στον πίνακα ξανά πατάμε στο κουμπί του.

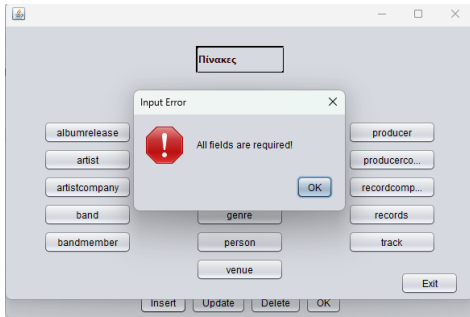


- Βλέπουμε την νέα μορφή του πίνακα με την νέα εγγραφή.

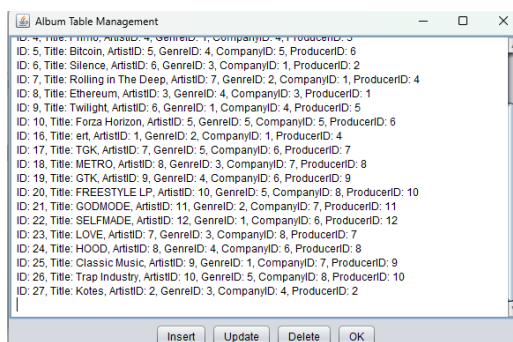
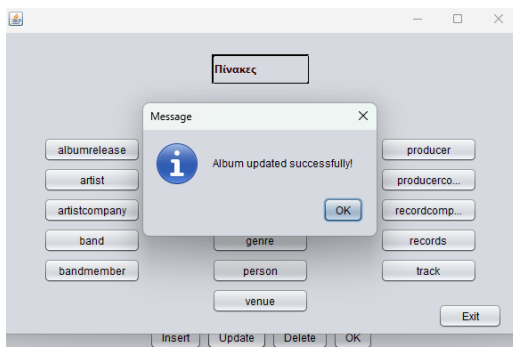
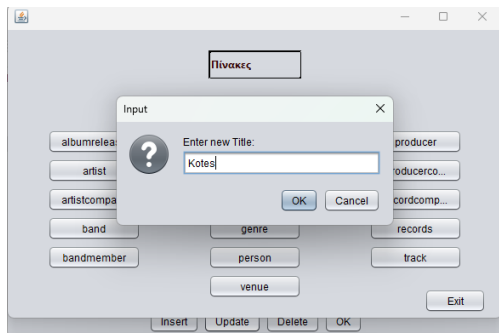
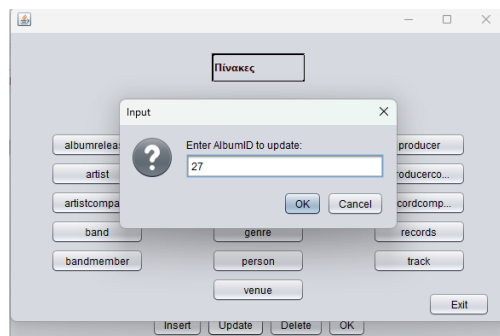


- Σε περίπτωση δεν συμπληρώσουμε όλα τα απαραίτητα στοιχεία εμφανίζεται το κατάλληλο μήνυμα. (όπως και στο update και στο delete).

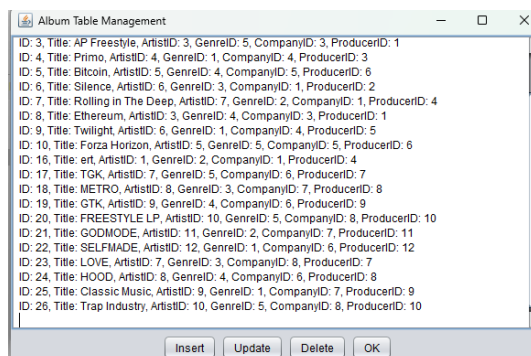
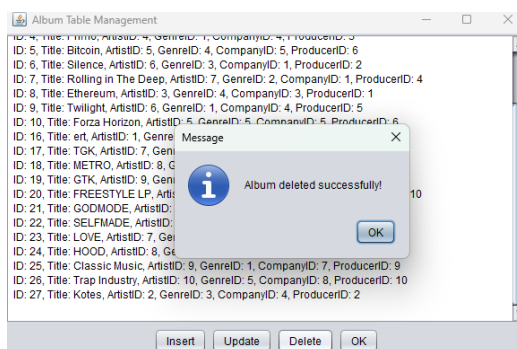
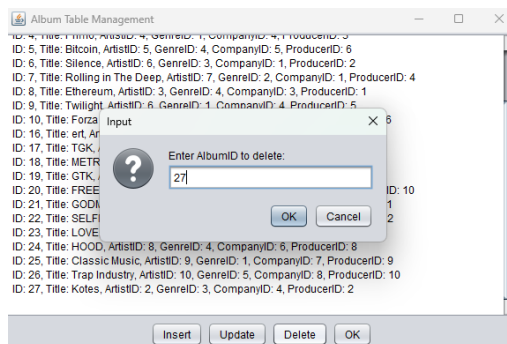




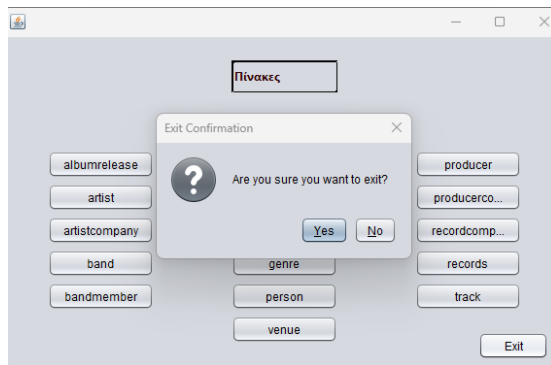
- Για το **update** γίνεται κάτι αντίστοιχο(θα χρησιμοποιήσουμε την εγγραφή που εισαγάγαμε).



- Για το **delete** γίνεται κάτι αντίστοιχο(θα χρησιμοποιήσουμε την εγγραφή που εισαγάγαμε).



- Για έξοδο από το interface πατάει ο χρήστης το exit στο αρχικό παράθυρο και εμφανίζεται ένα μικρό παράθυρο που τον ρωτάει αν θέλει σίγουρα να βγει .



ΚΩΔΙΚΑΣ SQL

--CREATES

```
CREATE TABLE IF NOT EXISTS artist(  
    ArtistID INT NOT NULL AUTO_INCREMENT,  
    ArtistType ENUM('PERSON', 'BAND') NULL DEFAULT NULL,  
    PRIMARY KEY (ArtistID));
```

```
CREATE TABLE IF NOT EXISTS band(  
    BandID INT NOT NULL AUTO_INCREMENT,  
    BandName VARCHAR(50) NULL DEFAULT "unknown",  
    FormationDate DATE NULL DEFAULT NULL,  
    DisbandDate DATE NULL DEFAULT NULL,  
    ArtistID INT NOT NULL,  
    PRIMARY KEY (BandID),  
    CONSTRAINT ArtistID1  
    FOREIGN KEY(ArtistID)  
    REFERENCES artist(ArtistID)  
    ON DELETE CASCADE  
    ON UPDATE CASCADE  
);
```

```

CREATE TABLE IF NOT EXISTS person(
    PersonID INT NOT NULL AUTO_INCREMENT,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(80) NOT NULL,
    BirthDate DATE ,
    Country VARCHAR(30) NULL DEFAULT "unknown",
    Alias VARCHAR(50) NULL DEFAULT "unknown",
    isSoloArtist TINYINT NULL ,
    ArtistID INT NOT NULL ,
    PRIMARY KEY (PersonID),
    CONSTRAINT ArtistID2
    FOREIGN KEY (ArtistID)
    REFERENCES artist(ArtistID)
    ON DELETE CASCADE
    ON UPDATE CASCADE);

```

```

CREATE TABLE IF NOT EXISTS bandmember(
    BandID INT NOT NULL,
    PersonID INT NOT NULL ,
    FromDate DATE NOT NULL,
    ToDate DATE ,
    PRIMARY KEY (PersonID,BandID,FromDate,ToDate),
    CONSTRAINT BandID
    FOREIGN KEY (BandID)
    REFERENCES band(BandID)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
    CONSTRAINT PersonID
    FOREIGN KEY (PersonID)
    REFERENCES person(PersonID)
    ON DELETE CASCADE
    ON UPDATE CASCADE);

```

```

CREATE TABLE IF NOT EXISTS genre(
    GenreID INT NOT NULL AUTO_INCREMENT,
    GenreName VARCHAR(30) NULL DEFAULT "unknown",
    PRIMARY KEY (GenreID)
);

```

```

CREATE TABLE IF NOT EXISTS producer(

```

```
ProducerID INT NOT NULL AUTO_INCREMENT,  
FirstName VARCHAR(50) NULL DEFAULT "unknown",  
LastName VARCHAR(80) NULL DEFAULT "unknown",  
NumofProdAlbums INT NOT NULL,  
PRIMARY KEY (ProducerID)  
);
```

```
CREATE TABLE IF NOT EXISTS recordcompany(  
CompanyID INT NOT NULL AUTO_INCREMENT,  
CompanyName VARCHAR(80) NULL DEFAULT "unknown",  
Adress VARCHAR(150) NULL DEFAULT "unknown",  
Phone VARCHAR(20) NULL DEFAULT "unknown",  
Email VARCHAR(40) NULL DEFAULT "unknown",  
BeginDate DATE,  
EndDate DATE,  
PRIMARY KEY (CompanyID)  
);
```

```
CREATE TABLE IF NOT EXISTS producercompany(  
ProducerID INT NOT NULL ,  
RecordCompanyID INT NOT NULL,  
FromDate DATE,  
ToDate DATE,  
PRIMARY KEY(ProducerID,RecordCompanyID,FromDate),  
CONSTRAINT CompanyID1  
    FOREIGN KEY (RecordCompanyID)  
REFERENCES recordcompany(CompanyID)  
ON DELETE CASCADE  
ON UPDATE CASCADE,  
CONSTRAINT ProducerID1  
    FOREIGN KEY (ProducerID)  
REFERENCES producer(ProducerID)  
ON DELETE CASCADE  
ON UPDATE CASCADE  
);
```

```
CREATE TABLE IF NOT EXISTS artistcompany(  
ArtistID INT NOT NULL AUTO_INCREMENT,  
CompanyID INT NOT NULL,  
FromDate DATE NOT NULL,
```



```
ToDate DATE NOT NULL,  
PRIMARY KEY (ArtistID, CompanyID, FromDate),  
CONSTRAINT ArtistID3  
FOREIGN KEY (ArtistID)  
REFERENCES artist(ArtistID)  
ON DELETE CASCADE  
ON UPDATE CASCADE,  
CONSTRAINT CompanyID2  
FOREIGN KEY (CompanyID)  
REFERENCES recordcompany(CompanyID)  
ON DELETE CASCADE  
ON UPDATE CASCADE);
```

```
CREATE TABLE IF NOT EXISTS album(  
AlbumID INT NOT NULL AUTO_INCREMENT,  
Title VARCHAR(100) NOT NULL,  
ArtistID INT NOT NULL,  
GenreID INT NOT NULL,  
CompanyID INT NOT NULL,  
ProducerID INT NOT NULL,  
PRIMARY KEY (AlbumID),  
CONSTRAINT GenreID  
FOREIGN KEY (GenreID)  
REFERENCES genre(GenreID)  
ON DELETE CASCADE  
ON UPDATE CASCADE,  
CONSTRAINT ArtistID  
FOREIGN KEY (ArtistID)  
REFERENCES artist(ArtistID)  
ON DELETE CASCADE  
ON UPDATE CASCADE,  
CONSTRAINT ProducerID  
FOREIGN KEY (ProducerID)  
REFERENCES producer(ProducerID)  
ON DELETE CASCADE  
ON UPDATE CASCADE,  
CONSTRAINT CompanyID  
FOREIGN KEY (CompanyID)  
REFERENCES recordcompany(CompanyID)  
ON DELETE CASCADE
```

ON UPDATE CASCADE

);

CREATE TABLE IF NOT EXISTS albumrelease(

ReleaseID INT NOT NULL AUTO_INCREMENT,

AlbumID INT NOT NULL,

ReleaseDate DATE NULL DEFAULT NULL,

ReleaseType ENUM('LP', 'CD', 'MP3') NULL DEFAULT NULL,

ReleaseStatus ENUM('OFFICIAL', 'PROMOTION', 'BOOTLEG', 'WITHDRAWN', 'CANCELED') NULL DEFAULT NULL,

Packaging ENUM('BOOK', 'CARDBOARD SLEEVE', 'DIGIPAK', 'JEWEL CASE', 'NA') NULL DEFAULT NULL,

PRIMARY KEY (ReleaseID),

CONSTRAINT AlbumID

FOREIGN KEY (AlbumID)

REFERENCES album(AlbumID)

ON DELETE CASCADE

ON UPDATE CASCADE);

CREATE TABLE If NOT EXISTS track(

TrackID INT NOT NULL AUTO_INCREMENT,

Title VARCHAR(100) NULL DEFAULT "unknown",

AlbumID INT NOT NULL ,

TrackLength TIME NOT NULL,

TrackNo INT NOT NULL ,

Lyrics TEXT NULL,

PRIMARY KEY(TrackID),

CONSTRAINT AlbumID1

FOREIGN KEY (AlbumID)

REFERENCES album(AlbumID)

ON DELETE CASCADE

ON UPDATE CASCADE

);

CREATE TABLE IF NOT EXISTS venue(

VenueID INT AUTO_INCREMENT ,

Name VARCHAR(100)NOT NULL,

Capacity INT NOT NULL,

CompletedCon INT DEFAULT 0,

YearsOperation INT NOT NULL,

venueStatus ENUM('AVAILABLE', 'BOOKED') DEFAULT 'AVAILABLE',

PRIMARY KEY(VenueID));

```

CREATE TABLE IF NOT EXISTS concert (
    ConcertID INT NOT NULL AUTO_INCREMENT,
    ArtistID INT NOT NULL,
    ConcertDate DATE NOT NULL,
    ConcertStatus ENUM('SCHEDULED', 'COMPLETED', 'CANCELED') NOT NULL DEFAULT 'SCHEDULED',
    RequiredCapacity INT NOT NULL DEFAULT 0,
    VenueID INT DEFAULT NULL,
    TicketSold INT ,
    PRIMARY KEY (ConcertID),
    INDEX (ArtistID),
    INDEX (VenueID),
    CONSTRAINT concert_venues FOREIGN KEY (VenueID)
        REFERENCES venue(VenueID)
        ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT concert_artists FOREIGN KEY (ArtistID)
        REFERENCES artist(ArtistID)
        ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE IF NOT EXISTS records(
    RecordID INT AUTO_INCREMENT,
    ConcertID INT,
    ArtistName VARCHAR(255) NOT NULL DEFAULT 'UNKNOWN',
    VenueName VARCHAR(255) NOT NULL DEFAULT 'UNKNOWN',
    TicketSold INT ,
    ConcertDate DATE DEFAULT NULL,
    RecordStatus ENUM('COMPLETED') NOT NULL,
    PRIMARY KEY(RecordID)
);

CREATE TABLE IF NOT EXISTS DBA(
    DBAID INT AUTO_INCREMENT,
    UserName VARCHAR(55),
    start_date date NOT NULL,
    end_date date,
    PRIMARY KEY (DBAID)
);

CREATE TABLE IF NOT EXISTS DBAlog(
    LogID INT AUTO_INCREMENT,

```

```
USERNAME VARCHAR(50) NOT NULL,  
DETAILS TEXT,  
actionDate DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,  
PRIMARY KEY(LogID)  
);
```

--INSERTS

```
INSERT INTO artist(ArtistType)
```

```
VALUES('PERSON'),
```

```
('PERSON'),
```

```
('PERSON'),
```

```
('PERSON'),
```

```
('PERSON'),
```

```
('PERSON'),
```

```
('PERSON'),
```

```
('BAND'),
```

```
('BAND'),
```

```
('BAND');
```

```
INSERT INTO band(BandName,FormationDate,DisbandDate,ArtistID)VALUES
```

```
("Ragland", '2005-01-01', NULL, 1),
```

```
("Strakofski", '2010-03-15', '2018-05-01', 2),
```

```
("Dropping stars", '2015-06-20', NULL, 3);
```

```
INSERT INTO person (FirstName,LastName,BirthDate,Country,Alias,isSoloArtist,ArtistID)VALUES
```

```
('George','Papadopoulos','1990-05-14','Greece','Giorgos',1,1),
```

```
('Maria','Kostantinidh', '2000-02-20','Greece','Marianna',0,2),
```

```
('John','Smith', '1988-11-03','United Kingdom','Johnny',1,3),
```

```
('Sophia','Nikolopoulou','1992-07-25', 'Greece', 'Sofia',1,4),
```

```
('Ethan','Taylor','1995-01-11','United States','EthanT',0,5),
```

```
('Katerina','Alexiou','1990-09-30','Greece','Kat',1,6),
```

```
('Michael','Anderson','2004-04-17','United States','Mike',0,7);
```

```
INSERT INTO bandmember(BandID,PersonID,FromDate,ToDate)VALUES
```

```
(1,1,'2005-01-01', '2010-12-31'),
```

```
(2,3,'2010-03-15', '2018-05-01'),
```

```
(2,4,'2011-01-01', '2016-06-01'),  
(3,2,'2015-06-20', '2020-12-31');
```

```
INSERT INTO genre(GenreName)VALUES  
('Rock'),('Jazz'),('Pop'),('Classical'),('Hip-Hop');
```

```
INSERT INTO producer(FirstName,LastName,NumofProdAlbums)VALUES  
('Elon',"Musk",15),  
('Giorgos',"Arsenakos",7),  
('John',"Doe",10),  
('Alice',"Smith",20),  
('George',"Ginwall",12),  
('George',"Sorros",8);
```

```
INSERT INTO recordcompany(CompanyName,Adress,Phone,Email,BeginDate,EndDate)VALUES  
('Tesla',"123 Electric Avenue,Silicon Valley,CA","123-456-7890","contact@tesla.com","2010-07-01",NULL),  
('PanikRecords',"45 Music St,Athens,Greece","210-123-4567","info@panikrecords.gr","2005-04-15", "2020-12-31"),  
('Omni',"789 Innovation Blvd,New York,NY","212-987-6543","support@omnimusic.com","2012-09-10",NULL),  
('Tutorial',"101 Creative Lane,Los Angeles,CA","310-555-1234","contact@tutorialrecords.com","2018-03-20", NULL),  
('OneWeb',"505 Tech Road,London,UK","020-7946-5120","info@onewebrecords.co.uk","2016-05-15","2023-06-30");
```

```
INSERT INTO producercompany (ProducerID, RecordCompanyID, FromDate, ToDate) VALUES  
(1, 3,'2011-01-01','2015-12-31'),  
(2, 2,'2005-04-15','2020-12-31'),  
(3, 4,'2010-09-10','2015-06-30'),  
(4, 5,'2018-03-20','2023-01-01'),  
(5, 1,'2016-05-15','2021-06-30'),  
(6, 3,'2016-07-01',NULL),  
(3, 5,'2015-07-01','2020-12-31'),  
(2, 4,'2021-01-01',NULL);
```

```
INSERT INTO artistcompany (ArtistID, CompanyID, FromDate, ToDate) VALUES  
(1, 1, '2008-05-01', '2023-05-01'),  
(2, 2, '2015-03-10', '2020-12-31'),  
(3, 3, '2008-11-15', '2023-11-15'),  
(4, 4, '2010-05-01', '2025-05-01'),  
(5, 5, '2014-07-01', '2023-06-30'),  
(6, 1, '2008-10-05', '2023-10-05'),
```

(7, 2, '2021-06-10', '2025-06-10');

```
INSERT INTO album (Title,ArtistID,GenreID,CompanyID,ProducerID)VALUES
('stormed',1,2,1,4),
('Goosebumps',2,3,2,5),
('AP Freestyle',3, 5,3,1),
('Primo',4,1,4,3),
('Bitcoin',5,4,5,6),
('Silence',6,3,1,2),
('Rolling in The Deep',7,2,1,4),
('Ethereum',3,4,3,1),
('Twilight',6,1,4,5),
('Forza Horizon',5,5,5,6);
```

```
INSERT INTO albumrelease(AlbumID,ReleaseDate,ReleaseType,ReleaseStatus,Packaging)
VALUES(1,'2015-01-01','CD','OFFICIAL','JEWEL CASE'),
(1,'2016-02-15','MP3','PROMOTION','DIGIPAK'),
(2,'2017-06-20','LP','OFFICIAL','CARDBOARD SLEEVE'),
(2,'2018-08-01','CD','PROMOTION','NA'),
(3,'2019-12-10','MP3','OFFICIAL','JEWEL CASE'),
(3,'2020-01-15','LP','PROMOTION','DIGIPAK'),
(4,'2015-05-15','CD','PROMOTION','DIGIPAK'),
(5,'2021-03-25','LP','CANCELED','JEWEL CASE'),
(6,'2019-07-01','MP3','OFFICIAL','JEWEL CASE'),
(7,'2020-09-10','CD','OFFICIAL','DIGIPAK'),
(8,'2021-05-20','MP3','PROMOTION','DIGIPAK'),
(9,'2021-11-11','LP','OFFICIAL','JEWEL CASE'),
(10,'2022-02-01','MP3','OFFICIAL','NA');
```

```
INSERT INTO track (Title, AlbumID, TrackLength, TrackNo, Lyrics) VALUES
("Story",1,'00:03:45',1,"vgazei story to vrady"),
("Booth",1,'00:04:15', 2,"mesa sto booth tragoydao"),
("Troll",2,'00:03:50', 1,"trollarei kai moy xalaei to paixnidi"),
("Manifest",2,'00:04:00',2,"manifest manifest manifest"),
("Break", 3, '00:04:05',1,"spao ta desma troo mpaklava"),
```

("Speed",3,'00:03:35',2,"trexo grhgora grhgoraa"),
 ("End",4,'00:04:20',1,"the end is coming for you"),
 ("Day",4,'00:03:50',2,"the day is today"),
 ("King ",5,'00:04:30',1,"vasilias ths kardias mas"),
 ("Lost",5,'00:03:55',2,"xamenos sthn avyso"),
 ("Revolution",6,'00:04:10',1,"the revolution begins now"),
 ("Tripoli",6,'00:03:40',2,"Catch the wave, the vibes of tomorrow"),
 ("Run", 7,'00:03:30',1,"Run free, nothing can hold us down"),
 ("Fire", 7,'00:04:00',2,"We're burning up, but it's all worth it"),
 ("Fade",8,'00:04:00',1,"The light shines through the darkness"),
 ("Past",8,'00:03:55',2,"Whispers of what was, haunting the future"),
 ("Hour",9,'00:04:20',1,"The clock is ticking, but we will not fall"),
 ("Win",9,'00:03:50',2,"We are born to win, no matter the cost"),
 ("Redemption",10,'00:04:10',1,"We seek redemption in the light"),
 ("Edge", 10,'00:03:45',2,"Living on the edge, nothing is safe"),
 ("Wild", 1,'00:03:35',3,"No rules, just freedom, we run wild"),
 ("Dreams", 2,'00:04:00',3,"Never stop chasing the dreams that drive you"),
 ("The Horizon",3,'00:03:50',3,"We look to the horizon, the future's ours"),
 ("Fight",4,'00:04:15',3,"Fight for tomorrow, never back down"),
 ("Starlight",5,'00:03:55',3,"Under the starlight, we find our way");

INSERT INTO artist (ArtistType)

VALUES

('PERSON'),
 ('PERSON'),
 ('PERSON'),
 ('PERSON'),
 ('PERSON'),
 ('PERSON'),
 ('PERSON'),
 ('PERSON'),
 ('BAND'),
 ('BAND'),
 ('BAND');

INSERT INTO person (FirstName, LastName, BirthDate, Country, Alias, isSoloArtist, ArtistID)

VALUES

('Chris', 'Rose', '2001-07-05', 'USA', 'CR', 1, 8),
 ('John', 'Snow', '2000-08-06', 'CANADA', 'KING', 1, 9),
 ('Stephen', 'Curry', '1995-12-05', 'USA', 'STEPH', 1, 10),

```
('Dimitris', 'Diamantidis', '1988-11-01', 'GREECE', '3D', 1, 11),
('Kendrick', 'Nunn', '1992-08-03', 'USA', 'NULL', 1, 12),
('Marinos', 'Argyros', '2002-03-27', 'GREECE', 'Laser', 1, 13),
('Peter', 'Johnshon', '1975-10-02', 'UK', 'NULL', 1, 14);
```

```
INSERT INTO person (FirstName, LastName, BirthDate, Country, Alias, isSoloArtist, ArtistID)
VALUES
('Alex', 'Johnson', '1995-04-12', 'USA', 'AJ', 0, 15), -- Assuming ArtistID 15
('Sarah', 'Smith', '1992-11-22', 'Canada', 'Sassy', 0, 16), -- Assuming ArtistID 16
('Mark', 'Davis', '1990-02-09', 'Australia', 'M-D', 0, 17), -- Assuming ArtistID 17
('Emily', 'Brown', '1998-05-30', 'UK', 'Ems', 0, 18); -- Assuming ArtistID 18
```

```
INSERT INTO band (BandName, FormationDate, DisbandDate, ArtistID)
VALUES
('GOATS', '2023-05-10', 'NULL', 15),
('GREENS', '2021-03-02', 'NULL', 16),
('HIVE', '2020-10-10', '2023-01-01', 17);
```

```
INSERT INTO person (FirstName, LastName, BirthDate, Country, Alias, isSoloArtist, ArtistID)
VALUES
('Alex', 'Johnson', '1995-04-12', 'USA', 'AJ', 0, 15),
('Sarah', 'Smith', '1992-11-22', 'Canada', 'Sassy', 0, 16),
('Mark', 'Davis', '1990-02-09', 'Australia', 'M-D', 0, 17),
('Emily', 'Brown', '1998-05-30', 'UK', 'Ems', 0, 18);
```

```
INSERT INTO bandmember (BandID, PersonID, FromDate, ToDate)
VALUES
(1, 15, '2023-05-10', '2025-12-31'),
(2, 16, '2021-03-02', '2025-12-31'),
(3, 17, '2020-10-10', '2023-01-01'),
(1, 18, '2023-05-10', '2025-12-31');
```

```
INSERT INTO genre (GenreName)
VALUES
('Metal'),
('Blues'),
('Reggae'),
```


('Electronic'),
('Indie');

INSERT INTO recordcompany (CompanyName, Address, Phone, Email, BeginDate, EndDate)

VALUES

('Soundwave Records', '123 Melody Lane, Nashville, TN', '555-1234-5678', 'info@soundwave.com', '2015-08-01', NULL),
('Echo Music Group', '456 Rhythm St, Los Angeles, CA', '555-2345-6789', 'contact@echomusic.com', '2010-03-15', NULL),
('Infinity Sounds', '789 Infinity Ave, Chicago, IL', '555-3456-7890', 'support@infinitysounds.com', '2012-11-01', '2020-12-31'),
('Neon Lights Records', '321 Glow Blvd, Miami, FL', '555-4567-8901', 'info@neonlights.com', '2018-05-10', NULL),
('Starry Night Music', '654 Dream St, New York, NY', '555-5678-9012', 'contact@starrynightmusic.com', '2020-01-15', NULL);

INSERT INTO artistcompany (ArtistID, CompanyID, FromDate, ToDate)

VALUES

(1, 6, '2022-06-01', '2025-06-01'),
(2, 7, '2020-08-10', '2025-08-10'),
(3, 8, '2021-09-15', '2025-09-15'),
(4, 9, '2019-04-22', '2024-04-22'),
(5, 10, '2020-12-01', '2025-12-01'),
(6, 6, '2018-11-05', '2023-11-05'),
(7, 7, '2021-05-15', '2026-05-15'),
(8, 8, '2017-07-30', '2022-07-30'),
(9, 9, '2022-03-05', '2027-03-05'),
(10, 10, '2023-01-10', '2028-01-10'),
(11, 6, '2021-11-25', '2026-11-25');

INSERT INTO producer (FirstName, LastName, NumofProdAlbums)

VALUES

('Alice', 'Johnson', 18),
('David', 'Bowie', 22),
('Rachel', 'Green', 13),
('Thomas', 'Anderson', 25),
('Olivia', 'Williams', 8),
('Chris', 'Brown', 30);

INSERT INTO producercompany (ProducerID, RecordCompanyID, FromDate, ToDate)

VALUES

(7, 6, '2015-07-01', '2020-12-31'),
(8, 7, '2020-05-01', NULL),
(9, 6, '2019-10-01', NULL),
(10, 8, '2020-03-15', '2023-06-30'),
(11, 7, '2021-01-01', NULL),
(12, 6, '2022-01-01', '2023-12-31'),
(7, 8, '2017-06-01', '2021-06-30'),
(8, 6, '2016-09-01', NULL);

INSERT INTO album (Title, ArtistID, GenreID, CompanyID, ProducerID)

VALUES

('TGK', 7, 5, 6, 7),
('METRO', 8, 3, 7, 8),
('GTK', 9, 4, 6, 9),
('FREESTYLE LP', 10, 5, 8, 10),
('GODMODE', 11, 2, 7, 11),
('SELFMADE', 12, 1, 6, 12),
('LOVE', 7, 3, 8, 7),
('HOOD', 8, 4, 6, 8),
('Classic Music', 9, 1, 7, 9),
('Trap Industry', 10, 5, 8, 10);

INSERT INTO track (Title, AlbumID, TrackLength, TrackNo, Lyrics) VALUES

("Xeiroterh Genia", 3, '00:05:17', 1, 'Lyrics'),
("Mov VroXH", 3, '00:03:27', 1, 'Lyrics'),
("Alhtikh Agaph", 3, '00:04:47', 1, 'Lyrics'),
("Fraxths", 1, '00:01:57', 1, 'Lyrics'),
("LovotomimenoI", 1, '00:04:37', 1, 'Lyrics'),
("TGK", 1, '00:05:47', 1, 'Lyrics'),
("Holdem", 2, '00:06:17', 1, 'Lyrics'),
("M2S", 2, '00:03:57', 1, 'Lyrics'),
("Point Bank", 2, '00:05:34', 1, 'Lyrics'),
("PsyXh", 4, '00:03:33', 1, 'Lyrics'),
("Oxi shmera", 5, '00:03:22', 1, 'Lyrics'),
("Diagnwsh", 6, '00:04:47', 1, 'Lyrics'),
("Airmax", 7, '00:00:57', 1, 'Lyrics'),
("Boban", 8, '00:02:32', 1, 'Lyrics'),
("Diasthmoploia", 9, '00:07:27', 1, 'Lyrics'),

("Polikatikies", 10, '00:01:11', 1, 'Lyrics'),
 ("Spike Lee", 4, '00:02:22', 1, 'Lyrics'),
 ("Ta skylia", 5, '00:04:44', 1, 'Lyrics'),
 ("Selini", 6, '00:05:55', 1, 'Lyrics'),
 ("Pou hsoun?", 7, '00:03:55', 1, 'Lyrics'),
 ("Aderfos gi aderfo", 8, '00:02:44', 1, 'Lyrics'),
 ("Paranoid", 9, '00:05:00', 1, 'Lyrics'),
 ("Lithium", 10, '00:03:04', 1, 'Lyrics'),
 ("Teleiwnoyme", 6, '00:04:05', 1, 'Lyrics'),
 ("Agaphmenoi", 8, '00:05:06', 1, 'Lyrics');

INSERT INTO albumrelease (AlbumID, ReleaseDate, ReleaseType, ReleaseStatus, Packaging) VALUES

(1, '2023-01-10', 'CD', 'OFFICIAL', 'JEWEL CASE'),
 (2, '2023-02-20', 'LP', 'OFFICIAL', 'CARDBOARD SLEEVE'),
 (3, '2023-03-15', 'MP3', 'PROMOTION', 'DIGIPAK'),
 (4, '2023-04-05', 'CD', 'PROMOTION', 'NA'),
 (5, '2023-05-10', 'LP', 'OFFICIAL', 'JEWEL CASE'),
 (6, '2023-06-01', 'MP3', 'OFFICIAL', 'DIGIPAK'),
 (7, '2023-07-12', 'LP', 'CANCELED', 'DIGIPAK'),
 (8, '2023-08-25', 'CD', 'OFFICIAL', 'NA'),
 (9, '2023-09-30', 'MP3', 'PROMOTION', 'CARDBOARD SLEEVE'),
 (10, '2023-10-15', 'LP', 'OFFICIAL', 'DIGIPAK'),
 (11, '2023-11-05', 'CD', 'PROMOTION', 'JEWEL CASE'),
 (12, '2023-12-01', 'MP3', 'CANCELED', 'NA');

INSERT INTO venue (Name, Capacity, CompletedCon, YearsOperation)

VALUES('Stadium Karaiskakis', 50000, 100, 15),
 ('OPAP Arena', 30000, 5, 10),
 ('Kolokotronis', 20000, 8, 20),
 ('Maracana', 40000, 7, 12),
 ('Open ', 15000, 4, 8);

INSERT INTO venue (Name, Capacity, CompletedCon, YearsOperation)

VALUES('Stadium Karaiskakis', 50000, 100, 15),
 ('OPAP Arena', 30000, 5, 10),
 ('Kolokotronis', 20000, 8, 20),
 ('Maracana', 40000, 7, 12),
 ('Open ', 15000, 4, 8);

INSERT INTO concert (ArtistID, ConcertDate, ConcertStatus, RequiredCapacity, VenueID)

```
VALUES(1,'2024-06-15','SCHEDULED',25000,NULL),
(2,'2024-07-20','SCHEDULED',15000,NULL),
(3,'2024-08-10','SCHEDULED',5000,NULL),
(4,'2024-09-05','SCHEDULED',30000,NULL),
(5,'2024-10-01','SCHEDULED',10000,NULL);
```

```
INSERT INTO DBA(UserName,start_date,end_date)
VALUES ('admin_user', '2023-01-01',NULL),
      ('db_manager', '2020-05-15',NULL);
INSERT INTO DBA(UserName,start_date,end_date)
VALUES ('sys_admin', '2023-05-24',NULL),
      ('dev_user', '2021-05-20',NULL);
```

```
INSERT INTO DBAlog (USERNAME, DETAILS) VALUES
('admin_user', 'Created a new table named EmployeeDetails.'),
('db_manager', 'Updated records in the CustomerOrders table.'),
('sys_admin', 'Granted SELECT privileges to user analytics_team.'),
('dev_user', 'Dropped unused table named TempData.');
```

--3.1.2.1.

DELIMITER \$

```
CREATE TRIGGER NewConcert
BEFORE INSERT ON concert
FOR EACH ROW
BEGIN

DECLARE ConcertCount INT;

IF DATEDIFF(NEW.ConcertDATE , CURDATE()) <=5
THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT ='Concert must be scheduled at least 5 days before ConsertDate.';
END IF;

SELECT COUNT(*) INTO ConcertCount
FROM concert
WHERE concertstatus='Scheduled'
AND artistID = NEW.artistID;
```

```

        IF (ConcertCount>=3) THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'This artist has 3 scheduled concerts already!';
END IF;
END$
DELIMITER ;

```

```

DELIMITER $

```

```

CREATE TRIGGER updateConcert
BEFORE UPDATE ON Concert
FOR EACH ROW
BEGIN
DECLARE ConcertCount INT;
IF OLD.ConcertStatus = 'Canceled' AND NEW.ConcertStatus = 'Scheduled'
THEN
SELECT COUNT(*) INTO ConcertCount
FROM concert
WHERE ConcertStatus = 'Scheduled'
AND artistID = NEW.artistID;

```

```

        IF (ConcertCount>=3) THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'This artist has 3 scheduled concerts already!';
END IF;
END IF;

```

```

IF OLD.ConcertStatus = 'Scheduled' AND NEW.ConcertStatus = 'Canceled'
THEN
        IF DATEDIFF(OLD.concertDate, CURDATE()) < 3 THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Concert cancellations must be made at least 3 days prior to the event.';
END IF;
END IF;

```

END\$

DELIMITER ;

DELIMITER \$

CREATE PROCEDURE ScheduleConcert(

IN artist_id INT,

IN concert_date DATE,

IN requiredcap INT

)

BEGIN

INSERT INTO concert (ArtistID, ConcertDate, RequiredCapacity, ConcertStatus)

VALUES (artist_id, concert_date, requiredcap, 'SCHEDULED');

END \$

DELIMITER ;

--3.1.2.2.

DELIMITER \$

create procedure venueforconcerts (IN cID INT)

BEGIN

DECLARE cap int;

DECLARE venID INT;

Select RequiredCapacity INTO cap

FROM concert

WHERE ConcertID=cID AND ConcertStatus!='Canceled';

SELECT venueID INTO venID

FROM venue

where capacity >=1.1*cap AND venueStatus != 'BOOKED'

order by((Capacity DIV 1000)*1 + (CompletedCon DIV 100)*3 + YearsOperation*2) DESC LIMIT 1;

IF venID IS NOT NULL THEN

UPDATE Concert

SET Concert.VenueID=venID, ConcertStatus ='Scheduled'

WHERE concertID=cID;

```

        UPDATE venue

        SET venueStatus = 'BOOKED'

        WHERE VenueID = venID;

        SELECT VenueID,Name FROM VENUE WHERE VenueID = venID;
ELSE SIGNAL SQLSTATE '45000'

        SET MESSAGE_TEXT ='THERE IS NO VENUE AVAILABLE';
END IF;

END $

DELIMITER ;

DELIMITER $
CREATE TRIGGER venueUpdate
AFTER UPDATE ON concert
FOR EACH ROW
BEGIN
    IF NEW.ConcertStatus !='Scheduled'
    THEN
        UPDATE VENUE
        SET venueStatus='AVAILABLE'
        where VenueID=NEW.VenueID;

    END IF;
END $
DELIMITER ;

DELIMITER $
CREATE PROCEDURE MoveConcertToRecord(IN inputConcertID INT)
BEGIN
    DECLARE rstatus varchar(100);

    SELECT ConcertStatus INTO rstatus
    from concert

```

```
where ConcertID=inputConcertID;
```

```
If rstatus!='SCHEDULED' THEN
```

```
    INSERT INTO records (
```

```
        ConcertID,
```

```
        ArtistName,
```

```
        VenueName,
```

```
        TicketSold,
```

```
        ConcertDate,
```

```
        RecordStatus
```

```
    )
```

```
SELECT
```

```
    c.ConcertID,
```

```
    CASE
```

```
        WHEN a.ArtistType = 'PERSON' THEN p.LastName
```

```
        ELSE b.BandName
```

```
    END AS ArtistName,
```

```
    v.Name AS VenueName,
```

```
    c.TicketSold AS TicketSold,
```

```
    CASE
```

```
        WHEN c.ConcertStatus = 'CANCELED' THEN NULL
```

```
        ELSE c.ConcertDate
```

```
    END AS ConcertDate,
```

```
    'COMPLETED' AS RecordStatus
```

```
FROM concert c
```

```
LEFT JOIN artist a ON c.ArtistID = a.ArtistID
```

```
LEFT JOIN venue v ON c.VenueID = v.VenueID
```

```
LEFT JOIN person p ON c.ArtistID = p.ArtistID
```

```
LEFT JOIN band b ON c.ArtistID = b.ArtistID
```

```
WHERE c.ConcertID = inputConcertID;
```

```
DELETE FROM concert WHERE ConcertID = inputConcertID;
```

```
ELSE
```

```
SIGNAL SQLSTATE '45000'
```

```
SET MESSAGE_TEXT = 'The concert is still scheduled.';
```

```
END IF;
```

```
END $
```

```
DELIMITER ;
```


--3.1.2.3.

```
CREATE TABLE IF NOT EXISTS records(  
    RecordID INT AUTO_INCREMENT,  
    ConcertID INT,  
    ArtistName VARCHAR(255) NOT NULL DEFAULT 'UNKNOWN',  
    VenueName VARCHAR(255) NOT NULL DEFAULT ' UNKNOWN',  
    TicketSold INT ,  
    ConcertDate DATE DEFAULT NULL,  
    RecordStatus ENUM('COMPLETED') NOT NULL,  
    PRIMARY KEY(RecordID)  
);
```

--3.1.2.4.

```
CREATE TABLE IF NOT EXISTS DBA(  
    DBAID INT AUTO_INCREMENT,  
    UserName VARCHAR(55),  
    start_date date NOT NULL,  
    end_date date,  
    PRIMARY KEY (DBAID)  
);
```

```
CREATE TABLE IF NOT EXISTS DBAlog(  
    LogID INT AUTO_INCREMENT,  
    USERNAME VARCHAR(50) NOT NULL,  
    DETAILS TEXT,  
    ActionDate DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,  
    PRIMARY KEY(LogID)  
);
```

--3.1.3.1.

```
DELIMITER $  
  
create procedure venuegrade (IN vID INT)  
  
BEGIN  
  
    DECLARE grade INT;  
        DECLARE vCapacity int;  
    DECLARE vCompletedCon int;  
    DECLARE vYearsOperation int;  
  
    SELECT
```

```

Capacity, CompletedCon, YearsOperation
INTO
vCapacity, vCompletedCon, vYearsOperation
FROM venue
WHERE vID=VenueID;

SET grade=((vCapacity DIV 1000)*1 + (vCompletedCon DIV 100)*3 + vYearsOperation*2) ;
SELECT grade AS VenueGrade;
END $

```

```

DELIMITER ;

```

--3.1.3.2.

```

DELIMITER $

CREATE PROCEDURE CreateCancelRescheduleConcert(IN aID INT, IN conDATE DATE, IN action_type CHAR(1))
BEGIN
DECLARE flag INT;
DECLARE flag0 INT;
DECLARE flag1 INT;
DECLARE flag2 INT;
DECLARE flag3 INT;
DECLARE flag4 INT;
IF action_type = 'i' THEN

    SELECT COUNT(*) into flag
    FROM CONCERT
    where aID=ArtistID
    AND conDATE=ConcertDate
    AND ConcertStatus ='scheduled';

    SELECT COUNT(*) into flag0
    FROM CONCERT
    where aID=ArtistID
    AND conDATE=ConcertDate
    AND ConcertStatus ='canceled';

    IF flag>0 THEN

```

```

        SELECT 'A concert is already scheduled' AS message;

ELSEIF flag0>0 THEN
    SELECT 'A concert is canceled for that day.' AS message;
ELSE
    INSERT INTO concert (ArtistID, ConcertDate, ConcertStatus)
        VALUES (aID, conDATE, 'scheduled');
    SELECT 'I scheduled the concert as you requested.' AS message;
    END IF;

ELSEIF action_type = 'c' THEN

    SELECT COUNT(*) into flag1
        FROM CONCERT
        where aID=ArtistID
        AND conDATE=ConcertDate
        AND ConcertStatus='scheduled';

    SELECT COUNT(*) into flag2
        FROM CONCERT
        where aID=ArtistID
        AND conDATE=ConcertDate
        AND ConcertStatus='canceled';

    IF flag1>0 THEN
        UPDATE concert
            SET ConcertStatus = 'canceled'
            WHERE ArtistID = aID
            AND ConcertDate = conDATE;
        SELECT 'The concert has been canceled.' AS message;

    ELSEIF flag2>0 THEN
        SELECT 'The concert has already been canceled.' AS message;

    ELSE
        SELECT 'There is no concert scheduled for that day.' AS message;
    END IF;

```

ELSEIF action_type = 'a' THEN

```
SELECT COUNT(*) into flag3
FROM CONCERT
where aID=ArtistID
AND conDATE=ConcertDate
AND ConcertStatus ='canceled';
```

```
SELECT COUNT(*) into flag4
FROM CONCERT
where aID=ArtistID
AND conDATE=ConcertDate
AND ConcertStatus ='scheduled';
```

IF flag3>0 THEN

```
UPDATE concert
SET ConcertStatus = 'scheduled'
WHERE ArtistID = aID
AND ConcertDate = conDATE;
SELECT 'The concert has been
```

rescheduled. ' AS message;

ELSEIF flag4>0 THEN

SELECT 'The concert is scheduled for that date.' AS message;

ELSE SELECT 'There is no canceled concert on that date.' AS message;

END IF;

ELSE

SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalid action type provided. Use "i", "c", or "a".';

END IF;

END\$

DELIMITER \$

--3.1.3.3.

DELIMITER \$

```
CREATE PROCEDURE GetConcertVenueDetails(
    IN conID INT,
    IN requiredCapacity INT,
    OUT venueID INT,
    OUT venueCapacity INT )
BEGIN
    DECLARE FLAG INT;
    DECLARE Cstatus VARCHAR(50);
    DECLARE vCapacity INT;
    DECLARE vCompletedCon INT;
    DECLARE vYearsOperation INT;

    SELECT COUNT(*)
    INTO FLAG
    FROM CONCERT
    WHERE conID=CONCERTID;

    SELECT CONCERT.ConcertStatus
    INTO Cstatus
    FROM CONCERT
    WHERE conID=CONCERTID;

    IF FLAG=0 THEN
        SET venueID = NULL;
        SET venueCapacity = 0;
        SELECT venueID AS VENUEID, venueCapacity AS CAPACITY;

    ELSEIF Cstatus='CANCELED' THEN
        SET venueID = NULL;
        SET venueCapacity = 0;
        SELECT venueID AS VENUEID, venueCapacity AS CAPACITY;

    ELSEIF FLAG>0 THEN
```

```

SELECT concert.VenueID , venue.capacity
INTO venueID, venueCapacity
FROM concert
LEFT JOIN venue ON concert.venueID=venue.venueID
WHERE CONCERTID=conID;

IF venueID IS NOT NULL THEN
SELECT venueID AS VENUEID, venueCapacity AS CAPACITY;

ELSE
SELECT venue.venueID,venue.Capacity INTO venueID,venueCapacity
FROM venue
where capacity >=1.1*requiredCapacity AND venueStatus != 'BOOKED'
order by( (Capacity DIV 1000)*1 + (CompletedCon DIV 100)*3 + YearsOperation*2 ) DESC LIMIT 1;

SELECT venueID AS VENUEID, venueCapacity AS CAPACITY;
end if;

END IF;
END $

```

DELIMITER ;

--3.1.3.4.

--A)

DELIMITER \$

```

CREATE PROCEDURE GetConcertDatesByVenue(IN venue VARCHAR(255))
BEGIN
    SELECT DISTINCT ConcertDate
    FROM records
    WHERE VenueName=venue;
END $
DELIMITER ;

```

CALL GetConcertDatesByVenue('Stadium Karaiskakis');

-- To index

CREATE INDEX idx_venue_name ON records (VenueName);

--B)

DELIMITER \$

CREATE PROCEDURE GetArtistsByTicketRange(IN minTickets INT, IN maxTickets INT)

BEGIN

 SELECT DISTINCT ArtistName

 FROM records

 WHERE TicketSold BETWEEN minTickets AND maxTickets;

END \$

DELIMITER ;

CALL GetArtistsByTicketRange(20000, 30000);

-- µε index

CREATE INDEX idxticketsoldartist ON records (TicketSold, ArtistName);

--3.1.4.1.

DELIMITER \$

CREATE TRIGGER person_insert

AFTER INSERT ON person

FOR EACH ROW

BEGIN

 INSERT INTO dbalog (USERNAME, DETAILS)

 VALUES (CURRENT_USER(), 'INSERT PERSON');

END\$

CREATE TRIGGER person_delete

AFTER DELETE ON person

FOR EACH ROW

BEGIN

 INSERT INTO dbalog (USERNAME, DETAILS)

 VALUES (CURRENT_USER(), 'DELETE PERSON');

END\$

```
CREATE TRIGGER person_update
AFTER UPDATE ON person
FOR EACH ROW
BEGIN

    INSERT INTO dbalog (USERNAME, DETAILS)
    VALUES ( CURRENT_USER(), 'UPDATE PERSON');

END$

DELIMITER ;
```

```
DELIMITER $

CREATE TRIGGER band_insert
AFTER insert ON band
FOR EACH ROW
BEGIN

    INSERT INTO dbalog (USERNAME, DETAILS)
    VALUES ( CURRENT_USER(), 'INSERT BAND');

END$
```

```
CREATE TRIGGER band_delete
AFTER DELETE ON band
FOR EACH ROW
BEGIN

    INSERT INTO dbalog (USERNAME, DETAILS)
    VALUES ( CURRENT_USER(), 'DELETE BAND');

END$
```

```
CREATE TRIGGER band_update
AFTER UPDATE ON band
FOR EACH ROW
BEGIN

    INSERT INTO dbalog (USERNAME, DETAILS)
    VALUES ( CURRENT_USER(), 'UPDATE BAND');
```


END\$

DELIMITER ;

DELIMITER \$

CREATE TRIGGER album_insert

AFTER insert ON album

FOR EACH ROW

BEGIN

INSERT INTO dbalog (USERNAME, DETAILS)

VALUES (CURRENT_USER(), 'INSERT ALBUM');

END\$

CREATE TRIGGER album_delete

AFTER DELETE ON album

FOR EACH ROW

BEGIN

INSERT INTO dbalog (USERNAME, DETAILS)

VALUES (CURRENT_USER(), 'DELETE ALBUM');

END\$

CREATE TRIGGER album_update

AFTER UPDATE ON album

FOR EACH ROW

BEGIN

INSERT INTO dbalog (USERNAME, DETAILS)

VALUES (CURRENT_USER(), 'UPDATE ALBUM');

END\$

DELIMITER ;

DELIMITER \$

CREATE TRIGGER concert_insert

AFTER insert ON concert

FOR EACH ROW

BEGIN

```
INSERT INTO dbalog (USERNAME, DETAILS)
VALUES ( CURRENT_USER(), 'INSERT CONCERT');
END$
```

```
CREATE TRIGGER concert_delete
AFTER DELETE ON concert
FOR EACH ROW
BEGIN
```

```
INSERT INTO dbalog (USERNAME, DETAILS)
VALUES ( CURRENT_USER(), 'DELETE CONCERT');
END$
```

```
CREATE TRIGGER concert_update
AFTER UPDATE ON concert
FOR EACH ROW
BEGIN
```

```
INSERT INTO dbalog (USERNAME, DETAILS)
VALUES ( CURRENT_USER(), 'UPDATE CONCERT');
END$
DELIMITER ;
```

DELIMITER \$

```
CREATE TRIGGER venue_insert
AFTER insert ON venue
FOR EACH ROW
BEGIN
```

```
INSERT INTO dbalog (USERNAME, DETAILS)
VALUES ( CURRENT_USER(), 'INSERT VENUE');
END$
```

```
CREATE TRIGGER venue_delete
AFTER DELETE ON venue
```

```

FOR EACH ROW
BEGIN

        INSERT INTO dbalog (USERNAME, DETAILS)
        VALUES ( CURRENT_USER(), 'DELETE VENUE');

END$

CREATE TRIGGER venue_update
AFTER UPDATE ON venue
FOR EACH ROW
BEGIN

        INSERT INTO dbalog (USERNAME, DETAILS)
        VALUES ( CURRENT_USER(), 'UPDATE VENUE');

END$
DELIMITER ;

```

--3.1.4.2.

```
DELIMITER $
```

```

CREATE TRIGGER NewConcert
BEFORE INSERT ON concert
FOR EACH ROW
BEGIN

DECLARE ConcertCount INT;

IF DATEDIFF(NEW.ConcertDATE , CURDATE()) <=5
    THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT ='Concert must be scheduled at least 5 days before ConsertDate.';
END IF;

SELECT COUNT(*) INTO ConcertCount
FROM concert
WHERE concertstatus='Scheduled'
AND artistID = NEW.artistID;

```

```

        IF (ConcertCount>=3) THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT ='This artist has 3 scheduled concerts already!';
END IF;
END$
DELIMITER ;

```

--3.1.4.3.

```

DELIMITER $

CREATE TRIGGER CancelRescheduleConcert
BEFORE UPDATE ON Concert
FOR EACH ROW
BEGIN
DECLARE ConcertCount INT;
IF OLD.ConcertStatus ='Canceled' AND NEW.ConcertStatus ='Scheduled'
THEN
SELECT COUNT(*) INTO ConcertCount
FROM concert
WHERE ConcertStatus ='Scheduled'
AND artistID = NEW.artistID;

```

```

        IF (ConcertCount>=3) THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT ='This artist has 3 scheduled concerts already!';
END IF;
END IF;

IF OLD.ConcertStatus ='Scheduled' AND NEW.ConcertStatus ='Canceled'
THEN
        IF DATEDIFF(OLD.concertDate, CURDATE()) < 3 THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Concert cancellations must be made at least 3 days prior to the event.';
END IF;
END IF;

END$

```

DELIMITER ;

Κώδικας java (GUI)

```
import java.sql.*;
import java.awt.*;
import javax.swing.*;
import java.sql.Date;

public class NewJFrame extends javax.swing.JFrame {
    private static Connection connection;

    /**
     * Creates new form NewJFrame
     */
    public NewJFrame(){
        try{ connection=DriverManager.
getConnection("jdbc:mysql://localhost:3306/musicindustry?useSSL=false&allowPublicKeyRetrieval=true", "root", "12345");
        System. out. println("Database connected!" ); }
        catch(SQLException e){ System. err. println("Not Connected: "+e.getMessage()); }
        initComponents();
    }

    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jInternalFrame1 = new javax.swing.JInternalFrame();
        jButton19 = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();
        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();
        jButton3 = new javax.swing.JButton();
        jButton4 = new javax.swing.JButton();
        jButton5 = new javax.swing.JButton();
        jButton6 = new javax.swing.JButton();
        jButton7 = new javax.swing.JButton();
    }
}
```

```

jButton8 = new javax.swing.JButton();
jButton9 = new javax.swing.JButton();
jButton10 = new javax.swing.JButton();
jButton11 = new javax.swing.JButton();
jButton12 = new javax.swing.JButton();
jButton13 = new javax.swing.JButton();
jButton14 = new javax.swing.JButton();
jButton15 = new javax.swing.JButton();
jButton16 = new javax.swing.JButton();
jButton17 = new javax.swing.JButton();
jButton18 = new javax.swing.JButton();

jInternalFrame1.setVisible(true);

jButton19.setText("jButton19");

javax.swing.GroupLayout jInternalFrame1Layout = new javax.swing.GroupLayout(jInternalFrame1.getContentPane());
jInternalFrame1.getContentPane().setLayout(jInternalFrame1Layout);
jInternalFrame1Layout.setHorizontalGroup(
    jInternalFrame1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jInternalFrame1Layout.createSequentialGroup()
            .add(jButton19, 235, Short.MAX_VALUE)
            .addGap(216, 216, 216))
);
jInternalFrame1Layout.setVerticalGroup(
    jInternalFrame1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jInternalFrame1Layout.createSequentialGroup()
            .add(jButton19, 54, 54)
            .addGap(153, Short.MAX_VALUE))
);

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
setBackground(new java.awt.Color(255, 51, 51));
setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
setForeground(new java.awt.Color(204, 51, 0));
setMaximizedBounds(new java.awt.Rectangle(0, 0, 0, 0));

jLabel1.setBackground(new java.awt.Color(0, 0, 0));

```

```
jLabel1.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
jLabel1.setForeground(new java.awt.Color(51, 0, 0));
jLabel1.setText("Τίτλος");
jLabel1.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));
jLabel1.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);


jButton1.setText("album");
jButton1.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        jButton1MouseClicked(evt);
    }
});

jButton2.setText("albumrelease");
jButton2.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        jButton2MouseClicked(evt);
    }
});

jButton3.setText("artist");
jButton3.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        jButton3MouseClicked(evt);
    }
});

jButton4.setText("artistcompany");
jButton4.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        jButton4MouseClicked(evt);
    }
});

jButton5.setText("band");
jButton5.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        jButton5MouseClicked(evt);
    }
});
```

```
jButton6.setText("bandmember");

jButton6.addMouseListener(new java.awt.event.MouseAdapter() {

    public void mouseClicked(java.awt.event.MouseEvent evt) {

        jButton6MouseClicked(evt);

    }

});
```

```
jButton7.setText("concert");

jButton7.addMouseListener(new java.awt.event.MouseAdapter() {

    public void mouseClicked(java.awt.event.MouseEvent evt) {

        jButton7MouseClicked(evt);

    }

});
```

```
jButton8.setText("dba");

jButton8.addMouseListener(new java.awt.event.MouseAdapter() {

    public void mouseClicked(java.awt.event.MouseEvent evt) {

        jButton8MouseClicked(evt);

    }

});
```

```
jButton9.setText("dbalog");

jButton9.addMouseListener(new java.awt.event.MouseAdapter() {

    public void mouseClicked(java.awt.event.MouseEvent evt) {

        jButton9MouseClicked(evt);

    }

});
```

```
jButton10.setText("genre");

jButton10.addMouseListener(new java.awt.event.MouseAdapter() {

    public void mouseClicked(java.awt.event.MouseEvent evt) {

        jButton10MouseClicked(evt);

    }

});
```

```
jButton11.setText("person");

jButton11.addMouseListener(new java.awt.event.MouseAdapter() {

    public void mouseClicked(java.awt.event.MouseEvent evt) {

        jButton11MouseClicked(evt);

    }

});
```



```

    }
});

jButton12.setText("producer");
jButton12.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton12ActionPerformed(evt);
    }
});

jButton13.setText("producercompany");
jButton13.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        jButton13MouseClicked(evt);
    }
});

jButton14.setText("recordcompany");
jButton14.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        jButton14MouseClicked(evt);
    }
});

jButton15.setText("records");
jButton15.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        jButton15MouseClicked(evt);
    }
});

jButton16.setText("track");
jButton16.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        jButton16MouseClicked(evt);
    }
});

jButton17.setText("venue");
jButton17.addActionListener(new java.awt.event.ActionListener() {

```

```

    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton17ActionPerformed(evt);
    }
});

jButton18.setText("Exit");
jButton18.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton18ActionPerformed(evt);
    }
});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(47, 47, 47)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jButton3, javax.swing.GroupLayout.PREFERRED_SIZE, 109, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 109, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jButton4, javax.swing.GroupLayout.PREFERRED_SIZE, 109, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jButton5, javax.swing.GroupLayout.PREFERRED_SIZE, 109, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jButton6, javax.swing.GroupLayout.PREFERRED_SIZE, 109, javax.swing.GroupLayout.PREFERRED_SIZE)
            )
            .addGap(80, 80, 80)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                .addComponent(jButton7, javax.swing.GroupLayout.PREFERRED_SIZE, 109, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jButton8, javax.swing.GroupLayout.PREFERRED_SIZE, 109, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jButton9, javax.swing.GroupLayout.PREFERRED_SIZE, 109, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jButton10, javax.swing.GroupLayout.PREFERRED_SIZE, 109, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jButton11, javax.swing.GroupLayout.PREFERRED_SIZE, 109, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 109, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 109, javax.swing.GroupLayout.PREFERRED_SIZE)
            )
            .addGap(80, 80, 80)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jButton13, javax.swing.GroupLayout.PREFERRED_SIZE, 109, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jButton12, javax.swing.GroupLayout.PREFERRED_SIZE, 109, javax.swing.GroupLayout.PREFERRED_SIZE)
            )
        )
);

```

```

        .addComponent(jButton14, javax.swing.GroupLayout.PREFERRED_SIZE, 109,
javax.swing.GroupLayout.PREFERRED_SIZE)

        .addComponent(jButton15, javax.swing.GroupLayout.PREFERRED_SIZE, 109,
javax.swing.GroupLayout.PREFERRED_SIZE)

        .addComponent(jButton16, javax.swing.GroupLayout.PREFERRED_SIZE, 109,
javax.swing.GroupLayout.PREFERRED_SIZE))

        .addContainerGap(47, Short.MAX_VALUE))

    .addGroup(layout.createSequentialGroup())

    .addGap(236, 236, 236)

    .addComponent(jButton17, javax.swing.GroupLayout.PREFERRED_SIZE, 109, javax.swing.GroupLayout.PREFERRED_SIZE)

    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)

    .addComponent(jButton18, javax.swing.GroupLayout.PREFERRED_SIZE, 72, javax.swing.GroupLayout.PREFERRED_SIZE)

    .addGap(19, 19, 19))

);

layout.setVerticalGroup(

    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

    .addGroup(layout.createSequentialGroup())

    .addGap(24, 24, 24)

    .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 33, javax.swing.GroupLayout.PREFERRED_SIZE)

    .addGap(26, 26, 26)

    .addComponent(jButton1)

    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

        .addComponent(jButton2)

        .addComponent(jButton7)

        .addComponent(jButton12))

    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

        .addComponent(jButton3)

        .addComponent(jButton8)

        .addComponent(jButton13))

    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

        .addComponent(jButton4)

        .addComponent(jButton9)

        .addComponent(jButton14))

    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

        .addComponent(jButton5)

        .addComponent(jButton10)

        .addComponent(jButton15))

```

```

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jButton11)
            .addComponent(jButton6)
            .addComponent(jButton16))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jButton17)
        .addGap(0, 23, Short.MAX_VALUE))
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(jButton18)
            .addContainerGap())
    );

    pack();
} // </editor-fold>

```

```

private void jButton2MouseClicked(java.awt.event.MouseEvent evt) {
    JFrame frame = new JFrame("AlbumRelease Table Management");
    frame.setSize(900,600);
    frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    frame.setLayout(new BorderLayout());
    JTextArea textArea=new JTextArea();
    textArea.setEditable(false);

```

```

    try{ String query="SELECT * FROM albumrelease";
        Statement stmt=connection.createStatement();
        ResultSet rs=stmt. executeQuery(query);

```

```

        StringBuilder res=new StringBuilder(" Album Releases:\n");//res=result
        while (rs.next()) {
            res.append("ReleaseID: ").append(rs.getInt("ReleaseID"))
                .append(", AlbumID: ").append(rs.getInt("AlbumID"))
                .append(", ReleaseDate: ").append(rs.getDate("ReleaseDate"))
                .append(", ReleaseType: ").append(rs.getString("ReleaseType"))
                .append(", ReleaseStatus: ").append(rs.getString("ReleaseStatus"))
                .append(", Packaging: ").append(rs.getString("Packaging"))
                .append("\n");}
        textArea.setText(res.toString());
        rs.close();

```

```

stmt.close();

    } catch(SQLException exc) {

        JOptionPane.showMessageDialog(this, "Cannot execute the query, there is an error: "+exc.getMessage(),"Database
Error",JOptionPane.ERROR_MESSAGE);

        return;
    }

    JScrollPane scrollPane=new JScrollPane(textArea);
    frame.add(scrollPane, BorderLayout.CENTER);


    JPanel buttpanel = new JPanel();
    JButton insertbutton = new JButton("Insert");
    JButton updatebutton = new JButton("Update");
    JButton deletebutton = new JButton("Delete");
    JButton okButton = new JButton("OK");


    buttpanel.add(insertbutton);
    buttpanel.add(updatebutton);
    buttpanel.add(deletebutton);
    buttpanel.add(okButton);
    frame.add(buttpanel, BorderLayout.SOUTH);


    insertbutton.addActionListener(exc->insertAlbumRelease());
    updatebutton.addActionListener(exc->updateAlbumRelease());
    deletebutton.addActionListener(exc->deleteAlbumRelease());


    okButton.addActionListener(exc->{ frame.dispose();new JFrame().setVisible(true);});


    frame.setVisible(true);
}

private void insertAlbumRelease() {

    String albumID=JOptionPane.showInputDialog(this,"Enter the AlbumID:");

    String releaseDate=JOptionPane.showInputDialog(this,"Enter the ReleaseDate(YYYY-MM-DD):");

    String releaseType=JOptionPane.showInputDialog(this,"Enter the ReleaseType(LP/CD/MP3):");

    String releaseStatus=JOptionPane.showInputDialog(this,"Enter the
ReleaseStatus(OFFICIAL/PROMOTION/BOOTLEG/WITHDRAWN/CANCELED):");

    String packaging=JOptionPane.showInputDialog(this,"Enter the Packaging(BOOK/CARDBOARD SLEEVE/DIGIPAK/JEWEL
CASE/NA):");


    if(albumID==null||albumID.trim().isEmpty()||releaseType==null||releaseStatus==null||packaging==null){

        JOptionPane.showMessageDialog(this,"ReleaseDate is not required!", "Input Error",JOptionPane.ERROR_MESSAGE);

        return;}
}

```

```

try{ String query="INSERT INTO albumrelease(AlbumID,ReleaseDate,ReleaseType,ReleaseStatus,Packaging)VALUES(?, ?, ?, ?, ?)";

    PreparedStatement pstmt=connection.prepareStatement(query);

    pstmt.setInt(1,Integer.parseInt(albumID));

    if(releaseDate==null||releaseDate.trim().isEmpty()){

        pstmt.setNull(2,java.sql.Types.DATE);

    }

    else{

        pstmt.setDate(2,java.sql.Date.valueOf(releaseDate));}

    pstmt.setString(3,releaseType.toUpperCase());

    pstmt.setString(4,releaseStatus.toUpperCase());

    pstmt.setString(5,packaging.toUpperCase());

    pstmt.executeUpdate();

    JOptionPane.showMessageDialog(this,"The AlbumRelease inserted successfully!");

}

catch(SQLException exc){

    JOptionPane.showMessageDialog(this,"Error inserting album release: "+exc.getMessage(),"Database
Error",JOptionPane.ERROR_MESSAGE);

}}

private void updateAlbumRelease(){

    String releaseID=JOptionPane.showInputDialog(this, "Enter ReleaseID:");

    String newReleaseDate=JOptionPane.showInputDialog(this, "Enter new ReleaseDate(YYYY-MM-DD):");

    String newReleaseType=JOptionPane.showInputDialog(this, "Enter new ReleaseType(LP/CD/MP3):");

    String newReleaseStatus=JOptionPane.showInputDialog(this, "Enter new
ReleaseStatus(OFFICIAL/PROMOTION/BOOTLEG/WITHDRAWN/CANCELED):");

    String newPackaging=JOptionPane.showInputDialog(this, "Enter new Packaging (BOOK/CARDBOARD SLEEVE/DIGIPAK/JEWEL
CASE/NA):");

    if (releaseID==null||releaseID.trim().isEmpty()||newReleaseType==null||newReleaseStatus==null||newPackaging==null) {

        JOptionPane.showMessageDialog(this,"ReleaseDate is not required!", "Input Error", JOptionPane.ERROR_MESSAGE);

        return;

    }

    try {

        String query = "UPDATE albumrelease SET ReleaseDate = ?, ReleaseType = ?, ReleaseStatus = ?, Packaging = ? WHERE ReleaseID =
?";

        PreparedStatement pstmt=connection.prepareStatement(query);

        if (newReleaseDate==null||newReleaseDate.trim().isEmpty()){

```

```

        pstmt.setNull(1,java.sql.Types.DATE);
    }
    else{
        pstmt.setDate(1,java.sql.Date.valueOf(newReleaseDate));
    }

    pstmt.setString(2,newReleaseType.toUpperCase());
    pstmt.setString(3,newReleaseStatus.toUpperCase());
    pstmt.setString(4,newPackaging.toUpperCase());
    pstmt.setInt(5,Integer.parseInt(releaseID));
    pstmt.executeUpdate();

    JOptionPane.showMessageDialog(this, "Album Release updated successfully!");
} catch(SQLException e){
    JOptionPane.showMessageDialog(this,"Error updating album release: " + e.getMessage(),"Database
Error",JOptionPane.ERROR_MESSAGE);
}
}

private void deleteAlbumRelease(){
    String releaseIDStr=JOptionPane.showInputDialog(this,"Enter the ReleaseID to delete:");

    if(releaseIDStr==null||releaseIDStr.trim().isEmpty()){
        JOptionPane.showMessageDialog(this,"The ReleaseID is required!", "Input Error",JOptionPane.ERROR_MESSAGE);
        return;}

    try{String query="DELETE FROM albumrelease WHERE ReleaseID = ?";
        PreparedStatement pstmt=connection.prepareStatement(query);
        pstmt.setInt(1,Integer.parseInt(releaseIDStr));
        pstmt.executeUpdate();

        JOptionPane.showMessageDialog(this,"The AlbumRelease deleted successfully!");
    }
    catch(SQLException exc)
    {
        JOptionPane.showMessageDialog(this,"Error deleting albumrelease: "+exc.getMessage(),"Database
Error",JOptionPane.ERROR_MESSAGE);
    }

}

private void jButton3MouseClicked(java.awt.event.MouseEvent evt) {
    JFrame frame=new JFrame("Artist Table Management");
    frame.setSize(800,500);

```

```

frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

frame.setLayout(new BorderLayout());

JTextArea textArea = new JTextArea();
textArea.setEditable(false);

try{ String query = "SELECT * FROM artist";
    Statement stmt=connection.createStatement();
    ResultSet rs=stmt.executeQuery(query);

    StringBuilder res=new StringBuilder("Artists:\n"); //res=result
    while (rs.next()) {
        res.append("ArtistID: ").append(rs.getInt("ArtistID"))
            .append(", ArtistType: ").append(rs.getString("ArtistType"))
            .append("\n"); }
    textArea.setText(res.toString());

    rs.close();
    stmt.close();
}
catch(SQLException exc){
    JOptionPane.showMessageDialog(this,"Error executing query: "+exc.getMessage(),"Database
Error",JOptionPane.ERROR_MESSAGE);
    return;
}

JScrollPane scrollPane = new JScrollPane(textArea);
frame.add(scrollPane, BorderLayout.CENTER);

JPanel buttpanel = new JPanel();
JButton insertButton = new JButton("Insert");
JButton updateButton = new JButton("Update");
JButton deleteButton = new JButton("Delete");
JButton okButton = new JButton("OK");

buttpanel.add(insertButton);
buttpanel.add(updateButton);
buttpanel.add(deleteButton);
buttpanel.add(okButton);
frame.add(buttpanel, BorderLayout.SOUTH);

```



```

insertButton.addActionListener(e->insertArtist());
updateButton.addActionListener(e->updateArtist());
deleteButton.addActionListener(e->deleteArtist());

okButton.addActionListener(e -> { frame.dispose();new JFrame().setVisible(true);});
frame.setVisible(true);
}

private void insertArtist() {
    String artistType=JOptionPane.showInputDialog(this,"Enter the ArtistType (PERSON/BAND):");

    if (artistType==null||artistType.trim().isEmpty()){
        JOptionPane.showMessageDialog(this,"ArtistType is required!", "Input Error",JOptionPane.ERROR_MESSAGE);
        return;}

    try{ String query = "INSERT INTO artist (ArtistType) VALUES (?)";
        PreparedStatement pstmt = connection.prepareStatement(query);
        pstmt.setString(1,artistType.toUpperCase());
        pstmt.executeUpdate();
        JOptionPane.showMessageDialog(this,"The Artist inserted successfully!");
    }
    catch(SQLException e){
        JOptionPane.showMessageDialog(this,"Error inserting artist: "+e.getMessage(),"Database Error",JOptionPane.ERROR_MESSAGE);
    }
}

private void updateArtist() {
    String artistIDStr = JOptionPane.showInputDialog(this, "Enter ArtistID:");
    String newArtistType = JOptionPane.showInputDialog(this, "Enter new ArtistType (PERSON/BAND):");

    if(artistIDStr==null||newArtistType==null||artistIDStr.trim().isEmpty()||newArtistType.trim().isEmpty()){
        JOptionPane.showMessageDialog(this, "All fields must be filled!", "Input Error",JOptionPane.ERROR_MESSAGE);
        return;}

    try{ int artistID=Integer.parseInt(artistIDStr);

        String query="UPDATE artist SET ArtistType = ? WHERE ArtistID = ?";
        PreparedStatement pstmt=connection.prepareStatement(query);
        pstmt.setString(1,newArtistType.toUpperCase());
        pstmt.setInt(2,artistID);
        pstmt.executeUpdate();
    }
}

```

```

        JOptionPane.showMessageDialog(this,"Artist updated successfully!");
    } catch (SQLException exc){
        JOptionPane.showMessageDialog(this,"Error updating artist:"+exc.getMessage(),"Database Error",JOptionPane.ERROR_MESSAGE);
    }
}

private void deleteArtist(){
    String artistIDStr=JOptionPane.showInputDialog(this,"Enter the ArtistID to delete:");

    if(artistIDStr==null||artistIDStr.trim().isEmpty()){
        JOptionPane.showMessageDialog(this,"ArtistID is required!", "Input Error",JOptionPane.ERROR_MESSAGE);
        return; }

    try{ int artistID=Integer.parseInt(artistIDStr);

        String query="DELETE FROM artist WHERE ArtistID = ?";
        PreparedStatement pstmt = connection.prepareStatement(query);
        pstmt.setInt(1,artistID);
        pstmt.executeUpdate();
        JOptionPane.showMessageDialog(this,"Artist deleted successfully!");
    }
    catch(SQLException e){
        JOptionPane.showMessageDialog(this,"Error deleting artist: "+e.getMessage(),"Database Error",JOptionPane.ERROR_MESSAGE);
    }

}

private void jButton4MouseClicked(java.awt.event.MouseEvent evt) {
    JFrame frame=new JFrame("ArtistCompany Table Management");
    frame.setSize(800,500);
    frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    frame.setLayout(new BorderLayout());

    JTextArea textArea = new JTextArea();
    textArea.setEditable(false);

    try{ String query="SELECT * FROM artistcompany";
        Statement stmt=connection.createStatement();
        ResultSet rs=stmt.executeQuery(query);

```

```

        StringBuilder res=new StringBuilder(" Artist Company Associations:\n");
        while(rs.next()){
            res.append("ArtistID: ").append(rs.getInt("ArtistID"))
                .append(", CompanyID: ").append(rs.getInt("CompanyID"))
                .append(", FromDate: ").append(rs.getDate("FromDate"))
                .append(", ToDate: ").append(rs.getDate("ToDate"))
                .append("\n");}
        textArea.setText(res.toString());
        rs.close();
        stmt.close();}
    catch(SQLException e){
        JOptionPane.showMessageDialog(this,"Error executing query: "+e.getMessage(),"Database Error",JOptionPane.ERROR_MESSAGE);
        return;}

    JScrollPane scrollPane=new JScrollPane(textArea);
    frame.add(scrollPane,BorderLayout.CENTER);

    JPanel buttpanel=new JPanel();
    JButton insertbutton = new JButton("Insert");
    JButton updatebutton = new JButton("Update");
    JButton deletebutton = new JButton("Delete");
    JButton okButton = new JButton("OK");

    buttpanel.add(insertbutton);
    buttpanel.add(updatebutton);
    buttpanel.add(deletebutton);
    buttpanel.add(okButton);
    frame.add(buttpanel, BorderLayout.SOUTH);

    insertbutton.addActionListener(e->insertArtistCompany());
    updatebutton.addActionListener(e->updateArtistCompany());
    deletebutton.addActionListener(e->deleteArtistCompany());

    okButton.addActionListener(e->{ frame.dispose();new JFrame().setVisible(true);});
    frame.setVisible(true);
}

private void insertArtistCompany(){
    String artistIDStr=JOptionPane.showInputDialog(this,"Enter the ArtistID:");
    String companyIDStr=JOptionPane.showInputDialog(this,"Enter the CompanyID:");

```

```

String fromDate=JOptionPane.showInputDialog(this,"Enter From Date(YYYY-MM-DD):");
String toDate=JOptionPane.showInputDialog(this,"Enter To Date(YYYY-MM-DD):");

if(artistIDStr==null||companyIDStr==null||fromDate==null||toDate==null||
    artistIDStr.trim().isEmpty()||companyIDStr.trim().isEmpty()||fromDate.trim().isEmpty() || toDate.trim().isEmpty()) {
    JOptionPane.showMessageDialog(this, "All fields are required!", "Input Error", JOptionPane.ERROR_MESSAGE);
    return;
}

try {
    int artistID=Integer.parseInt(artistIDStr);
    int companyID=Integer.parseInt(companyIDStr);

    String query ="INSERT INTO artistcompany (ArtistID, CompanyID, FromDate, ToDate) VALUES (?, ?, ?, ?)";
    PreparedStatement pstmt=connection.prepareStatement(query);
    pstmt.setInt(1,artistID);
    pstmt.setInt(2,companyID);
    pstmt.setDate(3,Date.valueOf(fromDate));
    pstmt.setDate(4,Date.valueOf(toDate));
    pstmt.executeUpdate();
    JOptionPane.showMessageDialog(this,"The insert of ArtistCompany is successfull");
}
catch(SQLException exc){
    JOptionPane.showMessageDialog(this,"Error inserting ArtistCompany: "+exc.getMessage(),"Database
    Error",JOptionPane.ERROR_MESSAGE);
}

private void updateArtistCompany(){
    String artistIDStr=JOptionPane.showInputDialog(this,"Enter the ArtistID:");
    String companyIDStr=JOptionPane.showInputDialog(this,"Enter the CompanyID:");
    String fromDate=JOptionPane.showInputDialog(this,"Enter From Date(YYYY-MM-DD):");
    String newToDate=JOptionPane.showInputDialog(this,"Enter new To Date(YYYY-MM-DD):");

    if(artistIDStr==null||companyIDStr==null||fromDate==null||newToDate==null||
        artistIDStr.trim().isEmpty()||companyIDStr.trim().isEmpty()||fromDate.trim().isEmpty()||newToDate.trim().isEmpty()){
        JOptionPane.showMessageDialog(this, "All fields are required!", "Input Error", JOptionPane.ERROR_MESSAGE);
        return;
    }
    try{ int artistID=Integer.parseInt(artistIDStr);
        int companyID=Integer.parseInt(companyIDStr);

        String query="UPDATE artistcompany SET ToDate = ? WHERE ArtistID = ? AND CompanyID = ? AND FromDate = ?";

```

```

        PreparedStatement pstmt = connection.prepareStatement(query);

        pstmt.setDate(1, Date.valueOf(newToDate));

        pstmt.setInt(2, artistID);

        pstmt.setInt(3, companyID);

        pstmt.setDate(4, Date.valueOf(fromDate));

        pstmt.executeUpdate();

        JOptionPane.showMessageDialog(this, "The insert of ArtistCompany is successfull");

    }

    catch(SQLException exc){

        JOptionPane.showMessageDialog(this, "Error updating ArtistCompany: " + exc.getMessage(), "Database
        Error", JOptionPane.ERROR_MESSAGE);

    }

}

private void deleteArtistCompany(){

    String artistIDStr=JOptionPane.showInputDialog(this, "Enter ArtistID:");

    String companyIDStr=JOptionPane.showInputDialog(this, "Enter CompanyID:");

    String fromDate = JOptionPane.showInputDialog(this, "Enter From Date (YYYY-MM-DD):");

    if(artistIDStr==null||companyIDStr==null||fromDate==null||
        artistIDStr.trim().isEmpty()||companyIDStr.trim().isEmpty()||fromDate.trim().isEmpty()){

        JOptionPane.showMessageDialog(this, "All fields must e filled", "Input Error", JOptionPane.ERROR_MESSAGE);

        return;

    }

    try{int artistID=Integer.parseInt(artistIDStr);

        int companyID=Integer.parseInt(companyIDStr);

        String query="DELETE FROM artistcompany WHERE ArtistID = ? AND CompanyID = ? AND FromDate = ?";

        PreparedStatement pstmt = connection.prepareStatement(query);

        pstmt.setInt(1, artistID);

        pstmt.setInt(2, companyID);

        pstmt.setDate(3, Date.valueOf(fromDate));

        pstmt.executeUpdate();

        JOptionPane.showMessageDialog(this, "ArtistCompany deleted successfully!");

    }

    catch(SQLException exc){

        JOptionPane.showMessageDialog(this, "Error deleting ArtistCompany : " + exc.getMessage(), "Database
        Error", JOptionPane.ERROR_MESSAGE);

    }

}

private void jButton5MouseClicked(java.awt.event.MouseEvent evt) {

```

```

JFrame frame=new JFrame("Band Table Management");
frame.setSize(800,500);
frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
frame.setLayout(new BorderLayout());

JTextArea textArea=new JTextArea();
textArea.setEditable(false);

try{ String query="SELECT * FROM band";
    Statement stmt=connection.createStatement();
    ResultSet rs=stmt.executeQuery(query);

    StringBuilder res=new StringBuilder("Bands:\n");
    while(rs.next()){
        res.append("BandID: ").append(rs.getInt("BandID"))
            .append(", BandName: ").append(rs.getString("BandName"))
            .append(", FormationDate: ").append(rs.getDate("FormationDate"))
            .append(", DisbandDate: ").append(rs.getDate("DisbandDate"))
            .append(", ArtistID: ").append(rs.getInt("ArtistID"))
            .append("\n");}
    textArea.setText(res.toString());
    rs.close();
    stmt.close();}
catch(SQLException exc){
    JOptionPane.showMessageDialog(this,"Error executing query: "+exc.getMessage(),"Database Error",JOptionPane.ERROR_MESSAGE);
    return;}

JScrollPane scrollPane=new JScrollPane(textArea);
frame.add(scrollPane,BorderLayout.CENTER);

JPanel buttpanel=new JPanel();
JButton insertbutton=new JButton("Insert");
JButton updatebutton=new JButton("Update");
JButton deletebutton=new JButton("Delete");
JButton okButton=new JButton("OK");

buttpanel.add(insertbutton);
buttpanel.add(updatebutton);
buttpanel.add(deletebutton);
buttpanel.add(okButton);

```

```
frame.add(buttonpanel, BorderLayout.SOUTH);
```

```
insertbutton.addActionListener(e->insertBand());
```

```
updatebutton.addActionListener(e->updateBand());
```

```
deletebutton.addActionListener(e->deleteBand());
```

```
okButton.addActionListener(e->{ frame.dispose();new JFrame().setVisible(true);});
```

```
frame.setVisible(true);
```

```
}
```

```
private void insertBand(){
```

```
    String bandName=JOptionPane.showInputDialog(this,"Enter the BandName:");
```

```
    String formationDate=JOptionPane.showInputDialog(this,"Enter Formation Date(YYYY-MM-DD):");
```

```
    String disbandDate=JOptionPane.showInputDialog(this,"Enter Disband Date(YYYY-MM-DD):");
```

```
    String artistIDStr=JOptionPane.showInputDialog(this,"Enter the ArtistID:");
```

```
    if(bandName==null||artistIDStr==null||bandName.trim().isEmpty()||artistIDStr.trim().isEmpty()){
```

```
        JOptionPane.showMessageDialog(this,"Band Name and ArtistID are missing!", "Input Error",JOptionPane.ERROR_MESSAGE);
```

```
        return;}
```

```
    try{ int artistID=Integer.parseInt(artistIDStr);
```

```
        String query="INSERT INTO band (BandName, FormationDate, DisbandDate, ArtistID) VALUES (?, ?, ?, ?)";
```

```
        PreparedStatement pstmt=connection.prepareStatement(query);
```

```
        pstmt.setString(1,bandName);
```

```
        pstmt.setDate(2,formationDate.isEmpty()?null:Date.valueOf(formationDate));
```

```
        pstmt.setDate(3,disbandDate.isEmpty()?null:Date.valueOf(disbandDate));
```

```
        pstmt.setInt(4,artistID);
```

```
        pstmt.executeUpdate();
```

```
        JOptionPane.showMessageDialog(this,"The Band inserted successfully!");}
```

```
    catch(SQLException exc){
```

```
        JOptionPane.showMessageDialog(this,"Error inserting band: "+exc.getMessage(),"Database Error",JOptionPane.ERROR_MESSAGE);
```

```
    }}
```

```
private void updateBand(){
```

```
    String bandIDStr=JOptionPane.showInputDialog(this,"Enter the BandID to update:");
```

```
    String newBandName=JOptionPane.showInputDialog(this,"Enter the new BandName:");
```

```
    if(bandIDStr==null||newBandName==null||bandIDStr.trim().isEmpty()||newBandName.trim().isEmpty()){
```

```
        JOptionPane.showMessageDialog(this,"The BandID and the BandName are required!", "Input Error",JOptionPane.ERROR_MESSAGE);
```

```

        return;}

try{ int bandID=Integer.parseInt(bandIDStr);

    String query="UPDATE band SET BandName = ? WHERE BandID = ?";
    PreparedStatement pstmt=connection.prepareStatement(query);
    pstmt.setString(1,newBandName);
    pstmt.setInt(2,bandID);
    pstmt.executeUpdate();

    JOptionPane.showMessageDialog(this,"The Band updated successfully!");}
catch(SQLException exc){
    JOptionPane.showMessageDialog(this,"Error updating band: "+exc.getMessage(),"Database Error",JOptionPane.ERROR_MESSAGE);
}}

private void deleteBand(){
    String bandIDStr=JOptionPane.showInputDialog(null,"Enter the BandID to delete:");

    if(bandIDStr==null||bandIDStr.trim().isEmpty()){
        JOptionPane.showMessageDialog(null,"BandID cannot be empty!", "Input Error",JOptionPane.ERROR_MESSAGE);
        return;}

    try{ int bandID=Integer.parseInt(bandIDStr);

        String query="DELETE FROM band WHERE BandID = ?";
        PreparedStatement pstmt=connection.prepareStatement(query);
        pstmt.setInt(1,bandID);
        pstmt.executeUpdate();

        JOptionPane.showMessageDialog(null,"Band deleted successfully!");}
    catch(SQLException e){
        JOptionPane.showMessageDialog(null,"Error deleting band: "+e.getMessage(),"Database Error",JOptionPane.ERROR_MESSAGE);
    }

}

private void jButton6MouseClicked(java.awt.event.MouseEvent evt) {
    JFrame frame=new JFrame("Band Member Table Management");
    frame.setSize(800,500);
    frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    frame.setLayout(new BorderLayout());

```



```
JTextArea textArea=new JTextArea();

textArea.setEditable(false);

try{ String query="SELECT * FROM bandmember";

    Statement stmt=connection.createStatement();

    ResultSet rs=stmt.executeQuery(query);

    StringBuilder res=new StringBuilder("Band Members:\n");

    while(rs.next()){

        res.append("BandID: ").append(rs.getInt("BandID"))

        .append(", PersonID: ").append(rs.getInt("PersonID"))

        .append(", FromDate: ").append(rs.getDate("FromDate"))

        .append(", ToDate: ").append(rs.getDate("ToDate"))

        .append("\n");

    }

    textArea.setText(res.toString());

    rs.close();

    stmt.close();

}

catch(SQLException exc){

    JOptionPane.showMessageDialog(this,"Error executing query: "+exc.getMessage(),"Database Error",JOptionPane.ERROR_MESSAGE);

    return;

}

JScrollPane scrollPane=new JScrollPane(textArea);

frame.add(scrollPane,BorderLayout.CENTER);

JPanel buttpanel=new JPanel();

JButton insertbutton=new JButton("Insert");

JButton updatebutton=new JButton("Update");

JButton deletebutton=new JButton("Delete");

JButton okbutton=new JButton("OK");

buttpanel.add(insertbutton);

buttpanel.add(updatebutton);

buttpanel.add(deletebutton);

buttpanel.add(okbutton);

frame.add(buttpanel,BorderLayout.SOUTH);
```

```

insertbutton.addActionListener(exc->insertBandMember());
updatebutton.addActionListener(exc->updateBandMember());
deletebutton.addActionListener(exc->deleteBandMember());

```

```

okbutton.addActionListener(exc->{ frame.dispose();
    new JFrame().setVisible(true);
});

```

```

frame.setVisible(true);
}

```

```

private void insertBandMember() {
String bandIDStr=JOptionPane.showInputDialog(this, "Enter the BandID:");
String personIDStr=JOptionPane.showInputDialog(this, "Enter the PersonID:");
String fromDate=JOptionPane.showInputDialog(this, "Enter From Date(YYYY-MM-DD):");
String toDate=JOptionPane.showInputDialog(this, "Enter To Date(YYYY-MM-DD):");

if(bandIDStr==null||personIDStr==null||fromDate==null||toDate==null||
    bandIDStr.trim().isEmpty()||personIDStr.trim().isEmpty()||fromDate.trim().isEmpty()||toDate.trim().isEmpty()){
    JOptionPane.showMessageDialog(this, "All fields are required!", "Input Error", JOptionPane.ERROR_MESSAGE);
    return;}

try{ int bandID=Integer.parseInt(bandIDStr);
    int personID=Integer.parseInt(personIDStr);

    String query="INSERT INTO bandmember (BandID, PersonID, FromDate, ToDate) VALUES (?, ?, ?, ?)";
    PreparedStatement pstmt=connection.prepareStatement(query);
    pstmt.setInt(1,bandID);
    pstmt.setInt(2,personID);
    pstmt.setDate(3,Date.valueOf(fromDate));
    pstmt.setDate(4,Date.valueOf(toDate));
    pstmt.executeUpdate();

    JOptionPane.showMessageDialog(this,"The Insert of BandMember is successfull!");
}

catch (SQLException exc) {
    JOptionPane.showMessageDialog(this,"Error inserting band member: "+exc.getMessage(),"Database
    Error",JOptionPane.ERROR_MESSAGE);
}
}

```

```

private void updateBandMember(){

    String bandIDStr=JOptionPane.showInputDialog(this, "Enter BandID:");

    String personIDStr=JOptionPane.showInputDialog(this, "Enter PersonID:");

    String fromDate=JOptionPane.showInputDialog(this, "Enter From Date (YYYY-MM-DD):");

    String newToDate=JOptionPane.showInputDialog(this, "Enter new To Date (YYYY-MM-DD):");


    if(bandIDStr==null||personIDStr==null||fromDate==null||newToDate==null||

        bandIDStr.trim().isEmpty()||personIDStr.trim().isEmpty()||fromDate.trim().isEmpty()||newToDate.trim().isEmpty()){

        JOptionPane.showMessageDialog(this,"All fields are required!", "Input Error",JOptionPane.ERROR_MESSAGE);

        return;

    }

    try{int bandID=Integer.parseInt(bandIDStr);

        int personID=Integer.parseInt(personIDStr);

String query="UPDATE bandmember SET ToDate = ? WHERE BandID = ? AND PersonID = ? AND FromDate = ?";

        PreparedStatement pstmt=connection.prepareStatement(query);

        pstmt.setDate(1,Date.valueOf(newToDate));

        pstmt.setInt(2,bandID);

        pstmt.setInt(3,personID);

        pstmt.setDate(4,Date.valueOf(fromDate));

        pstmt.executeUpdate();


        JOptionPane.showMessageDialog(this,"The update of BandMeber is successfull!");

    }

    catch(SQLException exc){

        JOptionPane.showMessageDialog(this,"Error updating band member: " +exc.getMessage(),"Database Error",JOptionPane.ERROR_MESSAGE);

    }

}

private void deleteBandMember(){

    String bandIDStr=JOptionPane.showInputDialog(this,"Enter the BandID:");

    String personIDStr=JOptionPane.showInputDialog(this, "Enter the PersonID:");

    String fromDate=JOptionPane.showInputDialog(this,"Enter From Date(YYYY-MM-DD):");

    String toDate=JOptionPane.showInputDialog(this,"Enter To Date(YYYY-MM-DD):");


    if(bandIDStr==null||personIDStr==null||fromDate==null||toDate==null||

        bandIDStr.trim().isEmpty()||personIDStr.trim().isEmpty()||fromDate.trim().isEmpty()||toDate.trim().isEmpty()) {

        JOptionPane.showMessageDialog(this,"All fields must be filled!", "Input Error", JOptionPane.ERROR_MESSAGE);

        return; }

    try{int bandID=Integer.parseInt(bandIDStr);

```

```

int personID=Integer.parseInt(personIDStr);

String query = "DELETE FROM bandmember WHERE BandID = ? AND PersonID = ? AND FromDate = ? AND ToDate = ?";
PreparedStatement pstmt=connection.prepareStatement(query);

pstmt.setInt(1,bandID);
pstmt.setInt(2,personID);
pstmt.setDate(3,Date.valueOf(fromDate));
pstmt.setDate(4,Date.valueOf(toDate));
pstmt.executeUpdate();

JOptionPane.showMessageDialog(this,"The Bandmember deleted successfully!");
}
catch(SQLException e){
    JOptionPane.showMessageDialog(this,"Error deleting band member: " +e.getMessage(),"Database
Error",JOptionPane.ERROR_MESSAGE);
}
}

private void jButton1MouseClicked(java.awt.event.MouseEvent evt) {
    JFrame frame=new JFrame("Album Table Management");
    frame.setSize(600,400);
    frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    frame.setLayout(new BorderLayout());

    JTextArea textArea=new JTextArea();
    textArea.setEditable(false);

    try{ String query="SELECT * FROM album";

        Statement stmt=connection.createStatement();
        ResultSet rs=stmt.executeQuery(query);

        StringBuilder res=new StringBuilder("Albums:\n");
        while(rs.next()){
            res.append("ID: ").append(rs.getInt("AlbumID"))
                .append(", Title: ").append(rs.getString("Title"))
                .append(", ArtistID: ").append(rs.getString("ArtistID"))
                .append(", GenreID: ").append(rs.getString("GenreID"))
                .append(", CompanyID: ").append(rs.getString("CompanyID"))
                .append(", ProducerID: ").append(rs.getString("ProducerID"))
                .append("\n");
        }
    }
}

```

```

        textArea.setText(res.toString());

        rs.close();

        stmt.close();

    }

    catch(SQLException exc){
        JOptionPane.showMessageDialog(this,"Error executing query: "+exc.getMessage(),"Database Error",JOptionPane.ERROR_MESSAGE);
        return;
    }

    JScrollPane scrollPane=new JScrollPane(textArea);
    frame.add(scrollPane,BorderLayout.CENTER);

    JPanel buttpanel=new JPanel();
    JButton insertbutton=new JButton("Insert");
    JButton updatebutton=new JButton("Update");
    JButton deletebutton=new JButton("Delete");
    JButton okbutton=new JButton("OK");

    buttpanel.add(insertbutton);
    buttpanel.add(updatebutton);
    buttpanel.add(deletebutton);
    buttpanel.add(okbutton);
    frame.add(buttpanel,BorderLayout.SOUTH);

    insertbutton.addActionListener(exc->insertAlbum());
    updatebutton.addActionListener(exc->updateAlbum());
    deletebutton.addActionListener(exc->deleteAlbum());

    okbutton.addActionListener(exc->{ frame.dispose();
        new JFrame().setVisible(true);
    });
    frame.setVisible(true);
}

private boolean isValidForeignKey(String tableName,String columnName,String value) throws SQLException{
    String query="SELECT COUNT(*) FROM "+tableName+" WHERE "+columnName+" = ?";
    PreparedStatement pstmt=connection.prepareStatement(query);
    pstmt.setInt(1,Integer.parseInt(value));
    ResultSet rs=pstmt.executeQuery();

```

```

        rs.next();
        return rs.getInt(1)>0;
    }

    private void insertAlbum(){
        String title=JOptionPane.showInputDialog(this,"Enter Title:");
        String artistID=JOptionPane.showInputDialog(this,"Enter ArtistID:");
        String genreID=JOptionPane.showInputDialog(this,"Enter GenreID:");
        String companyID=JOptionPane.showInputDialog(this,"Enter CompanyID:");
        String producerID=JOptionPane.showInputDialog(this,"Enter ProducerID:");

        if(title==null||artistID==null||genreID==null||companyID==null||producerID==null||
            title.trim().isEmpty()||artistID.trim().isEmpty()||genreID.trim().isEmpty()||companyID.trim().isEmpty()||producerID.trim().isEmpty()){
            JOptionPane.showMessageDialog(this,"All fields are required!", "Input Error",JOptionPane.ERROR_MESSAGE);
            return;
        }
        try{ String query="INSERT INTO album (Title, ArtistID, GenreID, CompanyID, ProducerID) VALUES (?, ?, ?, ?, ?)";
            PreparedStatement pstmt=connection.prepareStatement(query);
            pstmt.setString(1,title);
            pstmt.setInt(2,Integer.parseInt(artistID));
            pstmt.setInt(3,Integer.parseInt(genreID));
            pstmt.setInt(4,Integer.parseInt(companyID));
            pstmt.setInt(5,Integer.parseInt(producerID));
            pstmt.executeUpdate();

            JOptionPane.showMessageDialog(this,"Album inserted successfully!");
        }
        catch(SQLException exc){
            JOptionPane.showMessageDialog(this,"Error inserting album: "+exc.getMessage(),"Database
            Error",JOptionPane.ERROR_MESSAGE);
        }
    }

    private void updateAlbum(){
        String albumID=JOptionPane.showInputDialog(this,"Enter AlbumID to update:");
        String newTitle=JOptionPane.showInputDialog(this,"Enter new Title:");

        if(albumID==null||newTitle==null||albumID.trim().isEmpty()||newTitle.trim().isEmpty()){
            JOptionPane.showMessageDialog(this,"All fields are required!", "Input Error",JOptionPane.ERROR_MESSAGE);
            return;
        }
    }

```

```

try{ String query="UPDATE album SET Title = ? WHERE AlbumID = ?";

    PreparedStatement pstmt=connection.prepareStatement(query);

    pstmt.setString(1,newTitle);

    pstmt.setInt(2,Integer.parseInt(albumID));

    pstmt.executeUpdate();

    JOptionPane.showMessageDialog(this,"Album updated successfully!");

} catch(SQLException exc){

    JOptionPane.showMessageDialog(this,"Error updating album: "+exc.getMessage(),"Database
Error",JOptionPane.ERROR_MESSAGE);

}

}

private void deleteAlbum(){

    String albumID=JOptionPane.showInputDialog(null,"Enter AlbumID to delete:");

    if(albumID==null||albumID.trim().isEmpty()){

        JOptionPane.showMessageDialog(null,"AlbumID is required to delete!", "Input Error",JOptionPane.ERROR_MESSAGE);

        return;

    }

    try{ String query="DELETE FROM album WHERE AlbumID = ?";

        PreparedStatement pstmt=connection.prepareStatement(query);

        pstmt.setInt(1,Integer.parseInt(albumID));

        pstmt.executeUpdate();

        JOptionPane.showMessageDialog(null,"Album deleted successfully!");

    } catch(SQLException exc){

        JOptionPane.showMessageDialog(null,"Error deleting album: "+exc.getMessage(),"Database
Error",JOptionPane.ERROR_MESSAGE);

    }

}

private void jButton7MouseClicked(java.awt.event.MouseEvent evt) {

JFrame frame=new JFrame("Concert Table Management");

    frame.setSize(900,600);

    frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

    frame.setLayout(new BorderLayout());

    JTextArea textArea=new JTextArea();

    textArea.setEditable(false);

```

```

try{ String query="SELECT * FROM concert";

    Statement stmt=connection.createStatement();

    ResultSet rs=stmt.executeQuery(query);


    StringBuilder res=new StringBuilder("Concerts:\n");

    while(rs.next()){

        res.append("ConcertID: ").append(rs.getInt("ConcertID"))

        .append(", ArtistID: ").append(rs.getInt("ArtistID"))

        .append(", ConcertDate: ").append(rs.getDate("ConcertDate"))

        .append(", ConcertStatus: ").append(rs.getString("ConcertStatus"))

        .append(", RequiredCapacity: ").append(rs.getInt("RequiredCapacity"))

        .append(", VenueID: ").append(rs.getInt("VenueID"))

        .append("\n");

    }

    textArea.setText(res.toString());


    rs.close();

    stmt.close();}

catch(SQLException exc){

    JOptionPane.showMessageDialog(this,"Error executing query: "+exc.getMessage(),"Database Error",JOptionPane.ERROR_MESSAGE);

    return;

}


JScrollPane scrollPane=new JScrollPane(textArea);

frame.add(scrollPane,BorderLayout.CENTER);


JPanel buttpanel=new JPanel();

JButton insertbutton=new JButton("Insert");

JButton updatebutton=new JButton("Update");

JButton deletebutton=new JButton("Delete");

JButton okbutton=new JButton("OK");


buttpanel.add(insertbutton);

buttpanel.add(updatebutton);

buttpanel.add(deletebutton);

buttpanel.add(okbutton);

frame.add(buttpanel,BorderLayout.SOUTH);


insertbutton.addActionListener(e->insertConcert());

updatebutton.addActionListener(e->updateConcert());

```



```
deletebutton.addActionListener(e->deleteConcert());
```

```
okbutton.addActionListener(exc->{ frame.dispose();  
    new JFrame().setVisible(true);});
```

```
frame.setVisible(true);  
}
```

```
private void insertConcert(){
```

```
    String artistIDStr=JOptionPane.showInputDialog(this,"Enter the ArtistID:");
```

```
    String concertDateStr=JOptionPane.showInputDialog(this,"Enter the Concert Date(YYYY-MM-DD):");
```

```
    String concertStatus=JOptionPane.showInputDialog(this,"Enter the Concert Status(SCHEDULED, COMPLETED, CANCELED):");
```

```
    String requiredCapacityStr=JOptionPane.showInputDialog(this,"Enter the Required Capacity:");
```

```
    String venueIDStr=JOptionPane.showInputDialog(this,"Enter the VenueID (optional):");
```

```
    if(artistIDStr==null||concertDateStr==null||concertStatus==null||requiredCapacityStr==null){
```

```
        JOptionPane.showMessageDialog(this,"Operation canceled.", "Cancel",JOptionPane.INFORMATION_MESSAGE);
```

```
        return;}
```

```
    try{ String query="INSERT INTO concert (ArtistID, ConcertDate, ConcertStatus, RequiredCapacity, VenueID) VALUES (?, ?, ?, ?, ?)";
```

```
        PreparedStatement pstmt=connection.prepareStatement(query);
```

```
        pstmt.setInt(1,Integer.parseInt(artistIDStr));
```

```
        pstmt.setDate(2,Date.valueOf(concertDateStr));
```

```
        pstmt.setString(3,concertStatus);
```

```
        pstmt.setInt(4,Integer.parseInt(requiredCapacityStr));
```

```
        if(venueIDStr!=null&&!venueIDStr.trim().isEmpty()){
```

```
            pstmt.setInt(5,Integer.parseInt(venueIDStr));
```

```
        }
```

```
    else{
```

```
        pstmt.setNull(5,java.sql.Types.INTEGER);
```

```
    }
```

```
    pstmt.executeUpdate();
```

```
    JOptionPane.showMessageDialog(this,"Concert inserted successfully!");
```

```
    }
```

```
    catch(SQLException exc){
```

```
        JOptionPane.showMessageDialog(this,"Error inserting concert: "+exc.getMessage(),"Database  
Error",JOptionPane.ERROR_MESSAGE);
```

```
    }
```

```
}
```

```
private void updateConcert(){
```

```

String concertIDStr=JOptionPane.showInputDialog(this,"Enter ConcertID to update:");
String artistIDStr=JOptionPane.showInputDialog(this,"Enter new ArtistID:");
String concertDateStr=JOptionPane.showInputDialog(this,"Enter new Concert Date (YYYY-MM-DD):");
String concertStatus=JOptionPane.showInputDialog(this,"Enter new Concert Status (SCHEDULED, COMPLETED, CANCELED):");
String requiredCapacityStr=JOptionPane.showInputDialog(this,"Enter new Required Capacity:");
String venueIDStr=JOptionPane.showInputDialog(this,"Enter new VenueID (optional):");

if((concertIDStr==null||artistIDStr==null||concertDateStr==null||concertStatus==null||requiredCapacityStr==null){
    JOptionPane.showMessageDialog(this,"Operation canceled.", "Cancel",JOptionPane.INFORMATION_MESSAGE);
    return;
}

try{String query="UPDATE concert SET ArtistID = ?, ConcertDate = ?, ConcertStatus = ?, RequiredCapacity = ?, VenueID = ? WHERE
ConcertID = ?";

    PreparedStatement pstmt=connection.prepareStatement(query);

    pstmt.setInt(1,Integer.parseInt(artistIDStr));
    pstmt.setDate(2,Date.valueOf(concertDateStr));
    pstmt.setString(3,concertStatus);
    pstmt.setInt(4,Integer.parseInt(requiredCapacityStr));

    if(venueIDStr!=null&&!venueIDStr.trim().isEmpty()){
        pstmt.setInt(5,Integer.parseInt(venueIDStr));
    }
else{
    pstmt.setNull(5,java.sql.Types.INTEGER); // For NULL VenueID
}

    pstmt.setInt(6,Integer.parseInt(concertIDStr));
    pstmt.executeUpdate();

    JOptionPane.showMessageDialog(this,"Concert updated successfully!");
}
catch(SQLException exc){
    JOptionPane.showMessageDialog(this,"Error updating concert: "+exc.getMessage(),"Database
Error",JOptionPane.ERROR_MESSAGE);
}
}

private void deleteConcert(){
    String concertIDStr=JOptionPane.showInputDialog(this,"Enter ConcertID to delete:");

    if((concertIDStr==null||concertIDStr.trim().isEmpty()){

```

```

JOptionPane.showMessageDialog(this,"Operation canceled.", "Cancel",JOptionPane.INFORMATION_MESSAGE);
return;
}

try{ String query="DELETE FROM concert WHERE ConcertID = ?";
    PreparedStatement pstmt=connection.prepareStatement(query);
    pstmt.setInt(1,Integer.parseInt(concertIDStr));
    pstmt.executeUpdate();

    JOptionPane.showMessageDialog(this,"Concert deleted successfully!");
}

catch(SQLException exc){
    JOptionPane.showMessageDialog(this,"Error deleting concert: "+exc.getMessage(),"Database
    Error",JOptionPane.ERROR_MESSAGE);
}

}

private void jButton8MouseClicked(java.awt.event.MouseEvent evt) {
    JFrame frame=new JFrame("DBA Table Management");
    frame.setSize(900,600);
    frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    frame.setLayout(new BorderLayout());

    JTextArea textArea=new JTextArea();
    textArea.setEditable(false);

    try{
        String query="SELECT * FROM DBA";
        Statement stmt=connection.createStatement();
        ResultSet rs=stmt.executeQuery(query);

        StringBuilder res=new StringBuilder("DBA Records:\n");
        while(rs.next()){
            res.append("DBAID: ").append(rs.getInt("DBAID"))
                .append(", UserName: ").append(rs.getString("UserName"))
                .append(", Start Date: ").append(rs.getDate("start_date"))
                .append(", End Date: ").append(rs.getDate("end_date"))
                .append("\n");
        }
    }
}

```

```

        textArea.setText(res.toString());

        rs.close();

        stmt.close();
    }
    catch(SQLException exc){
        JOptionPane.showMessageDialog(frame,"Error executing query: "+exc.getMessage(),"Database
        Error",JOptionPane.ERROR_MESSAGE);

        return;
    }

    JScrollPane scrollPane=new JScrollPane(textArea);
    frame.add(scrollPane,BorderLayout.CENTER);

    JPanel buttpanel=new JPanel();
    JButton insertbutton=new JButton("Insert");
    JButton updatebutton=new JButton("Update");
    JButton deletebutton=new JButton("Delete");
    JButton okbutton=new JButton("OK");

    buttpanel.add(insertbutton);
    buttpanel.add(updatebutton);
    buttpanel.add(deletebutton);
    buttpanel.add(okbutton);
    frame.add(buttpanel,BorderLayout.SOUTH);

    insertbutton.addActionListener(e->insertDBA());
    updatebutton.addActionListener(e->updateDBA());
    deletebutton.addActionListener(e->deleteDBA());

    okbutton.addActionListener(e->{ frame.dispose();new NewJFrame().setVisible(true);});
    frame.setVisible(true);
}

private void insertDBA(){
    String userName=JOptionPane.showInputDialog(null,"Enter the UserName:");
    String startDate=JOptionPane.showInputDialog(null,"Enter the Start Date(YYYY-MM-DD):");
    String endDate=JOptionPane.showInputDialog(null,"Enter the End Date(YYYY-MM-DD):");

    if(userName==null||userName.trim().isEmpty()||startDate==null||startDate.trim().isEmpty()){
        JOptionPane.showMessageDialog(null,"UserName and Start Date are required!", "Input Error",JOptionPane.ERROR_MESSAGE);
    }
}

```

```

        return;
    }

    try{String query="INSERT INTO DBA (UserName, start_date, end_date) VALUES (?, ?, ?)";

        PreparedStatement pstmt=connection.prepareStatement(query);

        pstmt.setString(1,userName);

        pstmt.setDate(2,Date.valueOf(startDate));

        pstmt.setDate(3,Date.valueOf(endDate));

        pstmt.executeUpdate();

        JOptionPane.showMessageDialog(null,"DBA inserted successfully!");

    } catch(SQLException exc){

        JOptionPane.showMessageDialog(null,"Error inserting DBA: "+exc.getMessage(),"Database
Error",JOptionPane.ERROR_MESSAGE);

    }

}

private void updateDBA(){

    String dbaID=JOptionPane.showInputDialog(null,"Enter the DBAID to update:");

    String userName=JOptionPane.showInputDialog(null,"Enter the new UserName:");

    String startDate=JOptionPane.showInputDialog(null,"Enter the new Start Date(YYYY-MM-DD):");

    String endDate=JOptionPane.showInputDialog(null,"Enter the new End Date(YYYY-MM-DD):");

    if(dbaID==null||dbaID.trim().isEmpty()||userName==null||startDate==null||startDate.trim().isEmpty()){

        JOptionPane.showMessageDialog(null,"DBAID, UserName, and Start Date are required!", "Input
Error",JOptionPane.ERROR_MESSAGE);

        return;

    }

    try{ String query="UPDATE DBA SET UserName=?, start_date=?, end_date=? WHERE DBAID=?";

        PreparedStatement pstmt=connection.prepareStatement(query);

        pstmt.setString(1,userName);

        pstmt.setDate(2,Date.valueOf(startDate));

        pstmt.setDate(3,Date.valueOf(endDate));

        pstmt.setInt(4,Integer.parseInt(dbaID));

        pstmt.executeUpdate();

        JOptionPane.showMessageDialog(null,"DBA updated successfully!");

    } catch(SQLException exc){

        JOptionPane.showMessageDialog(null,"Error updating DBA: "+exc.getMessage(),"Database
Error",JOptionPane.ERROR_MESSAGE);

    }

}

private void deleteDBA(){

    String dbaID=JOptionPane.showInputDialog(null,"Enter the DBAID to delete:");

```

```

        if(dbaID==null||dbaID.trim().isEmpty()){
            JOptionPane.showMessageDialog(null,"DBAID is required!", "Input Error",JOptionPane.ERROR_MESSAGE);
            return;
        }
        try{ String query="DELETE FROM DBA WHERE DBAID=?";
            PreparedStatement pstmt=connection.prepareStatement(query);
            pstmt.setInt(1,Integer.parseInt(dbaID));
            pstmt.executeUpdate();
            JOptionPane.showMessageDialog(null,"DBA deleted successfully!");
        }
        catch(SQLException exc){
            JOptionPane.showMessageDialog(null,"Error deleting DBA: "+exc.getMessage(),"Database Error",JOptionPane.ERROR_MESSAGE);
        }
    }

    private void jButton9MouseClicked(java.awt.event.MouseEvent evt) {
JFrame frame=new JFrame("DBAlog Table Management");
        frame.setSize(800,500);
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        frame.setLayout(new BorderLayout());

        JTextArea textArea=new JTextArea();
        textArea.setEditable(false);

        try{
            String query="SELECT * FROM DBAlog";
            Statement stmt=connection.createStatement();
            ResultSet rs=stmt.executeQuery(query);

            StringBuilder res=new StringBuilder("DBA Logs:\n");
            while(rs.next()){
                res.append("LogID: ").append(rs.getInt("LogID"))
                    .append(", USERNAME: ").append(rs.getString("USERNAME"))
                    .append(", DETAILS: ").append(rs.getString("DETAILS"))
                    .append(", ActionDate: ").append(rs.getTimestamp("ActionDate"))
                    .append("\n");
            }textArea.setText(res.toString());
        }
    }

```

```

        rs.close();
        stmt.close();
    }
catch(SQLException e){
    JOptionPane.showMessageDialog(this,"Error executing query: "+e.getMessage(),"Database
Error",JOptionPane.ERROR_MESSAGE);
    return;
}

JScrollPane scrollPane=new JScrollPane(textArea);
frame.add(scrollPane,BorderLayout.CENTER);

JPanel buttpanel=new JPanel();
JButton insertbutton=new JButton("Insert");
JButton updatebutton=new JButton("Update");
JButton deletebutton=new JButton("Delete");
JButton okbutton=new JButton("OK");

buttpanel.add(insertbutton);
buttpanel.add(updatebutton);
buttpanel.add(deletebutton);
buttpanel.add(okbutton);
frame.add(buttpanel,BorderLayout.SOUTH);

insertbutton.addActionListener(e->insertDBAlog());
updatebutton.addActionListener(e->updateDBAlog());
deletebutton.addActionListener(e->deleteDBAlog());

okbutton.addActionListener(e->{ frame.dispose();
    new NewJFrame().setVisible(true);});
frame.setVisible(true);
}

private void insertDBAlog(){
    String username=JOptionPane.showInputDialog(this,"Enter USERNAME:");
    String details=JOptionPane.showInputDialog(this,"Enter DETAILS:");

    if(username==null||username.trim().isEmpty()){
        JOptionPane.showMessageDialog(this,"USERNAME is required!","Input Error",JOptionPane.ERROR_MESSAGE);
        return;
    }
}

```

```

try{String query="INSERT INTO DBAlog (USERNAME, DETAILS) VALUES (?, ?)";

    PreparedStatement pstmt=connection.prepareStatement(query);

    pstmt.setString(1,username);

    pstmt.setString(2,details);

    pstmt.executeUpdate();

    JOptionPane.showMessageDialog(this,"DBA log inserted successfully!");

}

catch(SQLException e){

    JOptionPane.showMessageDialog(this,"Error inserting the DBA log: "+e.getMessage(),"Database
Error",JOptionPane.ERROR_MESSAGE);

}

}

private void updateDBAlog(){

    String logIDStr=JOptionPane.showInputDialog(this,"Enter the LogID to update:");

    String newDetails=JOptionPane.showInputDialog(this,"Enter the new DETAILS:");

    if(logIDStr==null||newDetails==null||logIDStr.trim().isEmpty()||newDetails.trim().isEmpty()){

        JOptionPane.showMessageDialog(this,"Inputs cannot be empty!", "Input Error",JOptionPane.ERROR_MESSAGE);

        return;

    }

    try{int logID=Integer.parseInt(logIDStr);

        String query="UPDATE DBAlog SET DETAILS = ? WHERE LogID = ?";

        PreparedStatement pstmt=connection.prepareStatement(query);

        pstmt.setString(1,newDetails);

        pstmt.setInt(2,logID);

        pstmt.executeUpdate();

        JOptionPane.showMessageDialog(this,"The DBA log updated successfully!");

    }

    catch(SQLException exc){

        JOptionPane.showMessageDialog(this,"Error updating DBA log: "+exc.getMessage(),"Database
Error",JOptionPane.ERROR_MESSAGE);

    }

}

private void deleteDBAlog(){

    String logIDStr=JOptionPane.showInputDialog(null,"Enter the LogID to delete:");

    if(logIDStr==null||logIDStr.trim().isEmpty()){

        JOptionPane.showMessageDialog(null,"LogID cannot be empty!", "Input Error",JOptionPane.ERROR_MESSAGE);

    }

}

```



```

        return;
    }

    try{ int logID=Integer.parseInt(logIDStr);

        String query="DELETE FROM DBAlog WHERE LogID = ?";
        PreparedStatement pstmt=connection.prepareStatement(query);
        pstmt.setInt(1,logID);
        pstmt.executeUpdate();

        JOptionPane.showMessageDialog(null,"DBA log deleted successfully!");
    }

    catch(SQLException exc){
        JOptionPane.showMessageDialog(null,"Error deleting DBA log: "+exc.getMessage(),"Database
        Error",JOptionPane.ERROR_MESSAGE);
    }

}

private void jButton1MouseClicked(java.awt.event.MouseEvent evt) {
JFrame frame=new JFrame("Genre Table Management");
    frame.setSize(600,400);
    frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    frame.setLayout(new BorderLayout());

    JTextArea textArea=new JTextArea();
    textArea.setEditable(false);

    try{String query="SELECT * FROM genre";
        Statement stmt=connection.createStatement();
        ResultSet rs=stmt.executeQuery(query);

        StringBuilder res=new StringBuilder("Genres:\n");
        while(rs.next()){
            res.append("Genre ID: ").append(rs.getInt("GenreID"))
                .append(", Genre Name: ").append(rs.getString("GenreName"))
                .append("\n");
        }textArea.setText(res.toString());
        rs.close();
        stmt.close();
    }
}

```

```

    }

    catch(SQLException exc){

        JOptionPane.showMessageDialog(this,"Error executing query: "+exc.getMessage(),"Database
        Error",JOptionPane.ERROR_MESSAGE);

        return;}

    JScrollPane scrollPane=new JScrollPane(textArea);

    frame.add(scrollPane,BorderLayout.CENTER);

    JPanel buttpanel=new JPanel();

    JButton insertbutton=new JButton("Insert");

    JButton updatebutton=new JButton("Update");

    JButton deletebutton=new JButton("Delete");

    JButton okbutton=new JButton("OK");

    buttpanel.add(insertbutton);

    buttpanel.add(updatebutton);

    buttpanel.add(deletebutton);

    buttpanel.add(okbutton);

    frame.add(buttpanel,BorderLayout.SOUTH);

    insertbutton.addActionListener(e->insertGenre());

    updatebutton.addActionListener(e->updateGenre());

    deletebutton.addActionListener(e->deleteGenre());

    okbutton.addActionListener(e->{ frame.dispose();new JFrame().setVisible(true);});

    frame.setVisible(true);

    }

    private void insertGenre(){

        String genreName=JOptionPane.showInputDialog(this,"Enter Genre Name:");

        if(genreName==null||genreName.trim().isEmpty()){

            JOptionPane.showMessageDialog(this,"Genre Name cannot be empty!", "Input Error",JOptionPane.ERROR_MESSAGE);

            return;

        }

        try{ String query="INSERT INTO genre (GenreName) VALUES (?)";

            PreparedStatement pstmt=connection.prepareStatement(query);

            pstmt.setString(1,genreName);

            pstmt.executeUpdate();

```

```

        JOptionPane.showMessageDialog(this, "The Genre inserted successfully!");
    }
catch(SQLException exc){
    JOptionPane.showMessageDialog(this, "Error inserting genre: "+exc.getMessage(), "Database
Error", JOptionPane.ERROR_MESSAGE);
}
}

private void updateGenre(){
    String genreIDStr=JOptionPane.showInputDialog(this, "Enter the GenreID to update:");
    String newGenreName=JOptionPane.showInputDialog(this, "Enter the new Genre Name:");

    if(genreIDStr==null||newGenreName==null||genreIDStr.trim().isEmpty()||newGenreName.trim().isEmpty()){
        JOptionPane.showMessageDialog(this, "Inputs cannot be empty!", "Input Error", JOptionPane.ERROR_MESSAGE);
        return;
    }

    try{ int genreID=Integer.parseInt(genreIDStr);

        String query="UPDATE genre SET GenreName = ? WHERE GenreID = ?";
        PreparedStatement pstmt=connection.prepareStatement(query);
        pstmt.setString(1, newGenreName);
        pstmt.setInt(2, genreID);
        pstmt.executeUpdate();

        JOptionPane.showMessageDialog(this, "The Genre updated successfully!");
    }
catch(SQLException exc){
    JOptionPane.showMessageDialog(this, "Error updating genre: "+exc.getMessage(), "Database
Error", JOptionPane.ERROR_MESSAGE);
}
}

private void deleteGenre(){
    String genreIDStr=JOptionPane.showInputDialog(null, "Enter the GenreID to delete:");

    if(genreIDStr==null||genreIDStr.trim().isEmpty()){
        JOptionPane.showMessageDialog(null, "GenreID cannot be empty!", "Input Error", JOptionPane.ERROR_MESSAGE);
        return;
    }
    try{ int genreID=Integer.parseInt(genreIDStr);

        String query="DELETE FROM genre WHERE GenreID = ?";

```

```

        PreparedStatement pstmt=connection.prepareStatement(query);

        pstmt.setInt(1,genreID);

        pstmt.executeUpdate();

        JOptionPane.showMessageDialog(null,"Genre deleted successfully!");
    }

    catch(SQLException eXC){

        JOptionPane.showMessageDialog(null,"Error deleting genre: "+eXC.getMessage(),"Database
        Error",JOptionPane.ERROR_MESSAGE);

    }

}

private void jButton1MouseClicked(java.awt.event.MouseEvent evt) {
JFrame frame=new JFrame("Person Table Management");

    frame.setSize(800,500);

    frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

    frame.setLayout(new BorderLayout());

    JTextArea textArea=new JTextArea();

    textArea.setEditable(false);

    try{ String query="SELECT * FROM person";

        Statement stmt=connection.createStatement();

        ResultSet rs=stmt.executeQuery(query);

        StringBuilder res=new StringBuilder("Persons:\n");

        while(rs.next()){res.append("PersonID: ").append(rs.getInt("PersonID"))

            .append(", First Name: ").append(rs.getString("FirstName"))

            .append(", Last Name: ").append(rs.getString("LastName"))

            .append(", BirthDate: ").append(rs.getDate("BirthDate"))

            .append(", Country: ").append(rs.getString("Country"))

            .append(", Alias: ").append(rs.getString("Alias"))

            .append(", Is Solo Artist: ").append(rs.getBoolean("isSoloArtist"))

            .append(", ArtistID: ").append(rs.getInt("ArtistID"))

            .append("\n");}textArea.setText(res.toString());

        rs.close();

        stmt.close();

    }

    catch(SQLException exc){

```

```
JOptionPane.showMessageDialog(this, "Error executing query: "+exc.getMessage(), "Database Error", JOptionPane.ERROR_MESSAGE);
```

```
return;}


```

```
JScrollPane scrollPane=new JScrollPane(textArea);
```

```
frame.add(scrollPane, BorderLayout.CENTER);


```

```
JPanel buttpanel=new JPanel();
```

```
JButton insertbutton=new JButton("Insert");
```

```
JButton updatebutton=new JButton("Update");
```

```
JButton deletebutton=new JButton("Delete");
```

```
JButton okbutton=new JButton("OK");


```

```
buttpanel.add(insertbutton);
```

```
buttpanel.add(updatebutton);
```

```
buttpanel.add(deletebutton);
```

```
buttpanel.add(okbutton);
```

```
frame.add(buttpanel, BorderLayout.SOUTH);


```

```
insertbutton.addActionListener(e->insertPerson());
```

```
updatebutton.addActionListener(e->updatePerson());
```

```
deletebutton.addActionListener(e->deletePerson());


```

```
okbutton.addActionListener(e->{ frame.dispose();
```

```
    new JFrame().setVisible(true);});
```

```
frame.setVisible(true);
```

```
}
```

```
private void insertPerson(){
```

```
    String firstName=JOptionPane.showInputDialog(this, "Enter the First Name:");
```

```
    String lastName=JOptionPane.showInputDialog(this, "Enter the Last Name:");
```

```
    String birthDate=JOptionPane.showInputDialog(this, "Enter the Birth Date(YYYY-MM-DD):");
```

```
    String country=JOptionPane.showInputDialog(this, "Enter the Country:");
```

```
    String alias=JOptionPane.showInputDialog(this, "Enter the Alias:");
```

```
    String isSoloArtist=JOptionPane.showInputDialog(this, "Is Solo Artist(1 for Yes, 0 for No):");
```

```
    String artistID=JOptionPane.showInputDialog(this, "Enter the ArtistID:");


```

```
if(firstName==null||lastName==null||artistID==null||firstName.trim().isEmpty()||lastName.trim().isEmpty()||artistID.trim().isEmpty()){
```

```
    JOptionPane.showMessageDialog(this, "Required fields cannot be empty!", "Input Error", JOptionPane.ERROR_MESSAGE);
```

```
    return;
```

```
}
```

```

try{String query="INSERT INTO person (FirstName, LastName, BirthDate, Country, Alias, isSoloArtist, ArtistID) VALUES (?, ?, ?, ?, ?, ?, ?)";

    PreparedStatement pstmt=connection.prepareStatement(query);

    pstmt.setString(1,firstName);
    pstmt.setString(2,lastName);
    pstmt.setString(3,birthDate.isEmpty()?null:birthDate);
    pstmt.setString(4,country.isEmpty()? "unknown":country);
    pstmt.setString(5,alias.isEmpty()? "unknown":alias);
    pstmt.setInt(6,Integer.parseInt(isSoloArtist));
    pstmt.setInt(7,Integer.parseInt(artistID));
    pstmt.executeUpdate();

JOptionPane.showMessageDialog(this,"Person inserted successfully!");
}
catch(SQLException e){
    JOptionPane.showMessageDialog(this,"Error inserting person: "+e.getMessage(),"Database Error",JOptionPane.ERROR_MESSAGE);
}

private void updatePerson(){
    String personIDStr=JOptionPane.showInputDialog(this,"Enter PersonID to update:");
    String newFirstName=JOptionPane.showInputDialog(this,"Enter new First Name:");
    String newLastName=JOptionPane.showInputDialog(this,"Enter new Last Name:");

    if(personIDStr==null||newFirstName==null||newLastName==null||personIDStr.trim().isEmpty()||newFirstName.trim().isEmpty()||newLastName.trim().isEmpty()){
        JOptionPane.showMessageDialog(this,"Inputs cannot be empty!", "Input Error",JOptionPane.ERROR_MESSAGE);
        return;}

try{int personID=Integer.parseInt(personIDStr);

    String query="UPDATE person SET FirstName = ?, LastName = ? WHERE PersonID = ?";
    PreparedStatement pstmt=connection.prepareStatement(query);
    pstmt.setString(1,newFirstName);
    pstmt.setString(2,newLastName);
    pstmt.setInt(3,personID);
    pstmt.executeUpdate();

    JOptionPane.showMessageDialog(this,"Person updated successfully!");
}
catch(SQLException exc){

```

```

        JOptionPane.showMessageDialog(this, "Error updating person: "+exc.getMessage(), "Database
Error", JOptionPane.ERROR_MESSAGE);
    }}

    private void deletePerson(){
        String personIDStr=JOptionPane.showInputDialog(null, "Enter the PersonID to delete:");

        if(personIDStr==null||personIDStr.trim().isEmpty()){
            JOptionPane.showMessageDialog(null, "PersonID cannot be empty!", "Input Error", JOptionPane.ERROR_MESSAGE);
            return;
        }
        try{int personID=Integer.parseInt(personIDStr);

            String query="DELETE FROM person WHERE PersonID = ?";
            PreparedStatement pstmt=connection.prepareStatement(query);
            pstmt.setInt(1, personID);
            pstmt.executeUpdate();

            JOptionPane.showMessageDialog(null, "Person deleted successfully!");
        }
        catch(SQLException exc){
            JOptionPane.showMessageDialog(null, "Error deleting person: "+exc.getMessage(), "Database
Error", JOptionPane.ERROR_MESSAGE);
        }
    }

    private void jButton16MouseClicked(java.awt.event.MouseEvent evt) {

        JFrame frame=new JFrame("Track Table Management");
        frame.setSize(900,600);
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        frame.setLayout(new BorderLayout());

        JTextArea textArea=new JTextArea();
        textArea.setEditable(false);

        try {
            String query="SELECT * FROM track";
            Statement stmt=connection.createStatement();
            ResultSet rs=stmt.executeQuery(query);

```

```

        StringBuilder res=new StringBuilder("Tracks:\n");
        while(rs.next()){
            res.append("TrackID: ").append(rs.getInt("TrackID"))
                .append(", Title: ").append(rs.getString("Title"))
                .append(", AlbumID: ").append(rs.getInt("AlbumID"))
                .append(", TrackLength: ").append(rs.getTime("TrackLength"))
                .append(", TrackNo: ").append(rs.getInt("TrackNo"))
                .append("\n");} textArea.setText(res.toString());
rs.close();
        stmt.close();
    }
    catch(SQLException exc){
        JOptionPane.showMessageDialog(null,"Error executing query: "+exc.getMessage(),"Database
        Error",JOptionPane.ERROR_MESSAGE);
        return;
    }

    JScrollPane scrollPane=new JScrollPane(textArea);
    frame.add(scrollPane,BorderLayout.CENTER);

    JPanel buttpanel=new JPanel();
    JButton insertbutton=new JButton("Insert");
    JButton updatebutton=new JButton("Update");
    JButton deletebutton=new JButton("Delete");
    JButton okButton=new JButton("OK");

    buttpanel.add(insertbutton);
    buttpanel.add(updatebutton);
    buttpanel.add(deletebutton);
    buttpanel.add(okButton);
    frame.add(buttpanel,BorderLayout.SOUTH);

    insertbutton.addActionListener(e->insertTrack());
    updatebutton.addActionListener(e->updateTrack());
    deletebutton.addActionListener(e->deleteTrack());

    okButton.addActionListener(e->{ frame.dispose();
        new JFrame().setVisible(true); });
frame.setVisible(true);
    }

    private void insertTrack() {

```



```

String title=JOptionPane.showInputDialog(null,"Enter the Track Title:");
String albumIDStr=JOptionPane.showInputDialog(null,"Enter the AlbumID:");
String trackLengthStr=JOptionPane.showInputDialog(null,"Enter the Track Length (HH:MM:SS):");
String trackNoStr=JOptionPane.showInputDialog(null,"Enter the Track Number:");
String lyrics=JOptionPane.showInputDialog(null,"Enter the Lyrics:");

if(title==null||albumIDStr==null||trackLengthStr==null||trackNoStr==null) {
    JOptionPane.showMessageDialog(null,"Operation canceled.", "Cancel",JOptionPane.INFORMATION_MESSAGE);
    return;}

try{ String query="INSERT INTO track (Title, AlbumID, TrackLength, TrackNo, Lyrics) VALUES (?, ?, ?, ?, ?)";

    PreparedStatement pstmt=connection.prepareStatement(query);

    pstmt.setString(1,title);
    pstmt.setInt(2,Integer.parseInt(albumIDStr));
    pstmt.setTime(3,Time.valueOf(trackLengthStr));
    pstmt.setInt(4,Integer.parseInt(trackNoStr));
    pstmt.setString(5,lyrics);
    pstmt.executeUpdate();

    JOptionPane.showMessageDialog(null,"Track inserted successfully!");
}

catch(SQLException exc){
    JOptionPane.showMessageDialog(null,"Error inserting track: "+exc.getMessage(),"Database
    Error",JOptionPane.ERROR_MESSAGE);
    }}

private void updateTrack(){
    String trackIDStr=JOptionPane.showInputDialog(null,"Enter TrackID to update:");
    String title=JOptionPane.showInputDialog(null,"Enter new Track Title:");
    String albumIDStr=JOptionPane.showInputDialog(null,"Enter new AlbumID:");
    String trackLengthStr=JOptionPane.showInputDialog(null,"Enter new Track Length (HH:MM:SS):");
    String trackNoStr=JOptionPane.showInputDialog(null,"Enter new Track Number:");
    String lyrics=JOptionPane.showInputDialog(null,"Enter new Lyrics:");

    if(trackIDStr==null||trackIDStr.trim().isEmpty()||title==null||albumIDStr==null) {
        JOptionPane.showMessageDialog(null,"Operation canceled.", "Cancel",JOptionPane.INFORMATION_MESSAGE);
        return;
    }

    try{ String query="UPDATE track SET Title = ?, AlbumID = ?, TrackLength = ?, TrackNo = ?, Lyrics = ? WHERE TrackID = ?";

        PreparedStatement pstmt=connection.prepareStatement(query);

```

```

        pstmt.setString(1,title);
        pstmt.setInt(2,Integer.parseInt(albumIDStr));
        pstmt.setTime(3,Time.valueOf(trackLengthStr));
        pstmt.setInt(4,Integer.parseInt(trackNoStr));
        pstmt.setString(5,lyrics);
        pstmt.setInt(6,Integer.parseInt(trackIDStr));
        pstmt.executeUpdate();

        JOptionPane.showMessageDialog(null,"Track updated successfully!");
    }
    catch (SQLException e) {
        JOptionPane.showMessageDialog(null,"Error updating track: "+e.getMessage(),"Database
        Error",JOptionPane.ERROR_MESSAGE);
    }
}

private void deleteTrack(){
    String trackIDStr=JOptionPane.showInputDialog(null,"Enter TrackID to delete:");

    if((trackIDStr==null||trackIDStr.trim().isEmpty())){
        JOptionPane.showMessageDialog(null,"Operation canceled.", "Cancel",JOptionPane.INFORMATION_MESSAGE);
        return;
    }

    try{ String query="DELETE FROM track WHERE TrackID = ?";

        PreparedStatement pstmt=connection.prepareStatement(query);

        pstmt.setInt(1,Integer.parseInt(trackIDStr));

        pstmt.executeUpdate();

        JOptionPane.showMessageDialog(null,"Track deleted successfully!");}

    catch (SQLException exc){

        JOptionPane.showMessageDialog(null,"Error deleting track: "+exc.getMessage(),"Database
        Error",JOptionPane.ERROR_MESSAGE);

    }

}

private void jButton13MouseClicked(java.awt.event.MouseEvent evt) {
JFrame frame=new JFrame("Producer Company Table Management");

    frame.setSize(900,600);

    frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

    frame.setLayout(new BorderLayout());

```

```

JTextArea textArea=new JTextArea();

textArea.setEditable(false);

try{ String query="SELECT * FROM producercompany";

    Statement stmt=connection.createStatement();

    ResultSet rs=stmt.executeQuery(query);

    StringBuilder res=new StringBuilder("Producer Companies:\n");

    while(rs.next()){

        res.append("ProducerID: ").append(rs.getInt("ProducerID"))

        .append(", RecordCompanyID: ").append(rs.getInt("RecordCompanyID"))

        .append(", FromDate: ").append(rs.getDate("FromDate"))

        .append(", ToDate: ").append(rs.getDate("ToDate"))

        .append("\n");}textArea.setText(res.toString());

rs.close();

    stmt.close();

    }

catch(SQLException exc){

    JOptionPane.showMessageDialog(null,"Error executing query: "+exc.getMessage(),"Database
Error",JOptionPane.ERROR_MESSAGE);

    return;}

JScrollPane scrollPane=new JScrollPane(textArea);

frame.add(scrollPane,BorderLayout.CENTER);

JPanel buttpanel=new JPanel();

JButton insertbutton=new JButton("Insert");

JButton updatebutton=new JButton("Update");

JButton deletebutton=new JButton("Delete");

JButton okButton=new JButton("OK");

buttpanel.add(insertbutton);

buttpanel.add(updatebutton);

buttpanel.add(deletebutton);

buttpanel.add(okButton);

frame.add(buttpanel,BorderLayout.SOUTH);

insertbutton.addActionListener(e->insertProducerCompany());

updatebutton.addActionListener(e->updateProducerCompany());

deletebutton.addActionListener(e->deleteProducerCompany());

```

```

        okButton.addActionListener(e->{ frame.dispose();

            new JFrame().setVisible(true);});

frame.setVisible(true);

    }

    private void insertProducerCompany(){

        String producerIDStr=JOptionPane.showInputDialog(null,"Enter the ProducerID:");

        String recordCompanyIDStr=JOptionPane.showInputDialog(null,"Enter the RecordCompanyID:");

        String fromDateStr=JOptionPane.showInputDialog(null,"Enter FromDate(YYYY-MM-DD):");

        String toDateStr=JOptionPane.showInputDialog(null,"Enter ToDate (YYYY-MM-DD):");

        if(producerIDStr==null||producerIDStr.trim().isEmpty()||recordCompanyIDStr==null||recordCompanyIDStr.trim().isEmpty()) {

            JOptionPane.showMessageDialog(null,"ProducerID and RecordCompanyID are required!", "Input
Error",JOptionPane.ERROR_MESSAGE);

            return;}

        try{ String query="INSERT INTO producercompany (ProducerID, RecordCompanyID, FromDate, ToDate) VALUES (?, ?, ?, ?)";

            PreparedStatement pstmt=connection.prepareStatement(query);

            pstmt.setInt(1,Integer.parseInt(producerIDStr));

            pstmt.setInt(2,Integer.parseInt(recordCompanyIDStr));

            pstmt.setDate(3,java.sql.Date.valueOf(fromDateStr));

            if(toDateStr==null||toDateStr.trim().isEmpty()) {

                pstmt.setNull(4,java.sql.Types.DATE); // Set NULL for ToDate if not provided

            } else {

                pstmt.setDate(4,java.sql.Date.valueOf(toDateStr));

            }

            pstmt.executeUpdate();

            JOptionPane.showMessageDialog(null,"Producer Company inserted successfully!");

        }

        catch(SQLException exc){

            JOptionPane.showMessageDialog(null,"Error inserting producer company: "+exc.getMessage(),"Database
Error",JOptionPane.ERROR_MESSAGE);

            }}

        private void updateProducerCompany(){

            String producerIDStr=JOptionPane.showInputDialog(null,"Enter the ProducerID:");

            String recordCompanyIDStr=JOptionPane.showInputDialog(null,"Enter the RecordCompanyID:");

            String fromDateStr=JOptionPane.showInputDialog(null,"Enter new FromDate (YYYY-MM-DD):");

            String toDateStr=JOptionPane.showInputDialog(null,"Enter new ToDate (YYYY-MM-DD):");

```

```

if(producerIDStr==null||producerIDStr.trim().isEmpty()||recordCompanyIDStr==null||recordCompanyIDStr.trim().isEmpty()) {
    JOptionPane.showMessageDialog(null,"ProducerID and RecordCompanyID are required!", "Input
Error",JOptionPane.ERROR_MESSAGE);
    return; }

try{ String query="UPDATE producercompany SET FromDate = ?, ToDate = ? WHERE ProducerID = ? AND RecordCompanyID = ?";
    PreparedStatement pstmt=connection.prepareStatement(query);
    pstmt.setDate(1,java.sql.Date.valueOf(fromDateStr));
    if(toDateStr==null||toDateStr.trim().isEmpty()) {
        pstmt.setNull(2,java.sql.Types.DATE); // Set NULL for ToDate if not provided
    }
    else{
        pstmt.setDate(2,java.sql.Date.valueOf(toDateStr));}

    pstmt.setInt(3,Integer.parseInt(producerIDStr));
    pstmt.setInt(4,Integer.parseInt(recordCompanyIDStr));
    pstmt.executeUpdate();

    JOptionPane.showMessageDialog(null,"Producer Company updated successfully!");
}
catch(SQLException exc){
    JOptionPane.showMessageDialog(null,"Error updating producer company: "+exc.getMessage(),"Database
Error",JOptionPane.ERROR_MESSAGE);
}
}

private void deleteProducerCompany() {
    String producerIDStr=JOptionPane.showInputDialog(null,"Enter the ProducerID:");
    String recordCompanyIDStr=JOptionPane.showInputDialog(null,"Enter the RecordCompanyID:");

    if(producerIDStr==null||producerIDStr.trim().isEmpty()||recordCompanyIDStr==null||recordCompanyIDStr.trim().isEmpty()) {
        JOptionPane.showMessageDialog(null,"ProducerID and RecordCompanyID are required!", "Input
Error",JOptionPane.ERROR_MESSAGE);
        return;}

    try{ String query="DELETE FROM producercompany WHERE ProducerID = ? AND RecordCompanyID = ?";
        PreparedStatement pstmt=connection.prepareStatement(query);
        pstmt.setInt(1,Integer.parseInt(producerIDStr));
        pstmt.setInt(2,Integer.parseInt(recordCompanyIDStr));
        pstmt.executeUpdate();

        JOptionPane.showMessageDialog(null,"Producer Company deleted successfully!");
    }
    catch(SQLException exc){

```

```

        JOptionPane.showMessageDialog(null,"Error deleting producer company: "+exc.getMessage(),"Database
        Error",JOptionPane.ERROR_MESSAGE);
    }
}

```

```

private void jButton14MouseClicked(java.awt.event.MouseEvent evt) {
JFrame frame=new JFrame("Record Company Table Management");
frame.setSize(900,600);
frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
frame.setLayout(new BorderLayout());

```

```

JTextArea textArea=new JTextArea();
textArea.setEditable(false);

```

```

try{ String query="SELECT * FROM recordcompany";

```

```

    Statement stmt=connection.createStatement();
    ResultSet rs=stmt.executeQuery(query);

```

```

    StringBuilder res=new StringBuilder("Record Companies:\n");
    while(rs.next()){
        res.append("CompanyID: ").append(rs.getInt("CompanyID"))
            .append(", CompanyName: ").append(rs.getString("CompanyName"))
            .append(", Address: ").append(rs.getString("Adress"))
            .append(", Phone: ").append(rs.getString("Phone"))
            .append(", Email: ").append(rs.getString("Email"))
            .append(", BeginDate: ").append(rs.getDate("BeginDate"))
            .append(", endDate: ").append(rs.getDate("EndDate"))
            .append("\n"); } textArea.setText(res.toString());

```

```

    rs.close();
    stmt.close();}

```

```

catch(SQLException exc){
    JOptionPane.showMessageDialog(null,"Error executing query: "+exc.getMessage(),"Database
    Error",JOptionPane.ERROR_MESSAGE);
    return;}

```

```

JScrollPane scrollPane=new JScrollPane(textArea);
frame.add(scrollPane,BorderLayout.CENTER);

```

```

JPanel buttpanel=new JPanel();

```

```

JButton insertbutton=new JButton("Insert");
JButton updatebutton=new JButton("Update");
JButton deletebutton=new JButton("Delete");
JButton okButton=new JButton("OK");

buttpanel.add(insertbutton);
buttpanel.add(updatebutton);
buttpanel.add(deletebutton);
buttpanel.add(okButton);
frame.add(buttpanel, BorderLayout.SOUTH);

insertbutton.addActionListener(e->insertRecordCompany());
updatebutton.addActionListener(e->updateRecordCompany());
deletebutton.addActionListener(e->deleteRecordCompany());

okButton.addActionListener(e->{
    frame.dispose();
    new JFrame().setVisible(true);
});

frame.setVisible(true);
}

private void insertRecordCompany() {
    String companyName=JOptionPane.showInputDialog(null,"Enter the CompanyName:");
    String address=JOptionPane.showInputDialog(null,"Enter thee Address:");
    String phone=JOptionPane.showInputDialog(null,"Enter the Phone:");
    String email=JOptionPane.showInputDialog(null,"Enter the Email:");
    String beginDateStr=JOptionPane.showInputDialog(null,"Enter BeginDate (YYYY-MM-DD):");
    String endDateStr=JOptionPane.showInputDialog(null,"Enter EndDate (YYYY-MM-DD):");

    if(companyName==null||companyName.trim().isEmpty()||address==null||address.trim().isEmpty()||
        phone==null||phone.trim().isEmpty()||email==null||email.trim().isEmpty()) {
        JOptionPane.showMessageDialog(null,"CompanyName, Address, Phone, and Email are required!", "Input
Error",JOptionPane.ERROR_MESSAGE);
        return;}

    try{ String query="INSERT INTO recordcompany (CompanyName, Adress, Phone, Email, BeginDate, EndDate) VALUES (?, ?, ?, ?, ?)";

        PreparedStatement pstmt=connection.prepareStatement(query);

        pstmt.setString(1,companyName);

        pstmt.setString(2,address);

        pstmt.setString(3,phone);

```

```

        pstmt.setString(4,email);
        if(beginDateStr==null||beginDateStr.trim().isEmpty()) {
            pstmt.setNull(5,java.sql.Types.DATE);
        }
        else{
            pstmt.setDate(5,java.sql.Date.valueOf(beginDateStr));
        }
        if(endDateStr==null||endDateStr.trim().isEmpty()) {
            pstmt.setNull(6,java.sql.Types.DATE);
        }
        else{
            pstmt.setDate(6,java.sql.Date.valueOf(endDateStr));
        }

        pstmt.executeUpdate();

        JOptionPane.showMessageDialog(null,"The Record Company inserted successfully!");
    }
    catch(SQLException exc){
        JOptionPane.showMessageDialog(null,"Error inserting record company: "+exc.getMessage(),"Database
        Error",JOptionPane.ERROR_MESSAGE);
    }
}

private void updateRecordCompany(){
    String companyIDStr=JOptionPane.showInputDialog(null,"Enter the CompanyID:");
    String newCompanyName=JOptionPane.showInputDialog(null,"Enter the new CompanyName:");
    String newAddress=JOptionPane.showInputDialog(null,"Enter new Address:");
    String newPhone=JOptionPane.showInputDialog(null,"Enter the new Phone:");
    String newEmail=JOptionPane.showInputDialog(null,"Enter the new Email:");
    String newBeginDateStr=JOptionPane.showInputDialog(null,"Enter new BeginDate(YYYY-MM-DD):");
    String newEndDateStr=JOptionPane.showInputDialog(null,"Enter new EndDate(YYYY-MM-DD):");

    if(companyIDStr==null||companyIDStr.trim().isEmpty()||newCompanyName==null||newCompanyName.trim().isEmpty()||
        newAddress==null||newAddress.trim().isEmpty()||newPhone==null||newPhone.trim().isEmpty()||
        newEmail==null||newEmail.trim().isEmpty()){
        JOptionPane.showMessageDialog(null,"All fields except BeginDate and EndDate must be filled!", "Input
        Error",JOptionPane.ERROR_MESSAGE);
        return;}

    try{String query="UPDATE recordcompany SET CompanyName = ?, Adress = ?, Phone = ?, Email = ?, BeginDate = ?, EndDate = ?
    WHERE CompanyID = ?";

        PreparedStatement pstmt=connection.prepareStatement(query);

        pstmt.setString(1,newCompanyName);
        pstmt.setString(2,newAddress);

```



```

        pstmt.setString(3,newPhone);
        pstmt.setString(4,newEmail);
        if(newBeginDateStr==null||newBeginDateStr.trim().isEmpty()) {
            pstmt.setNull(5,java.sql.Types.DATE);
        }
        else{
            pstmt.setDate(5,java.sql.Date.valueOf(newBeginDateStr));
        }
        if(newEndDateStr==null||newEndDateStr.trim().isEmpty()) {
            pstmt.setNull(6,java.sql.Types.DATE);
        }
        else{
            pstmt.setDate(6,java.sql.Date.valueOf(newEndDateStr));
        }
        pstmt.setInt(7,Integer.parseInt(companyIDStr));
        pstmt.executeUpdate();

        JOptionPane.showMessageDialog(null,"The RecordCompany updated successfully!");
    }

    catch(SQLException exc){
        JOptionPane.showMessageDialog(null,"Error updating record company: "+exc.getMessage(),"Database
        Error",JOptionPane.ERROR_MESSAGE);
    }}

    private void deleteRecordCompany(){
        String companyIDStr=JOptionPane.showInputDialog(null,"Enter CompanyID to delete:");

        if(companyIDStr==null||companyIDStr.trim().isEmpty()) {
            JOptionPane.showMessageDialog(null,"CompanyID is required!", "Input Error",JOptionPane.ERROR_MESSAGE);
            return;}

        try{String query="DELETE FROM recordcompany WHERE CompanyID = ?";

            PreparedStatement pstmt=connection.prepareStatement(query);

            pstmt.setInt(1,Integer.parseInt(companyIDStr));

            pstmt.executeUpdate();

            JOptionPane.showMessageDialog(null,"Record Company deleted successfully!");
        }

        catch(SQLException ex){
            JOptionPane.showMessageDialog(null,"Error deleting record company: "+ex.getMessage(),"Database
            Error",JOptionPane.ERROR_MESSAGE);
        }
    }
}

```

```

private void jButton15MouseClicked(java.awt.event.MouseEvent evt) {
JFrame frame=new JFrame("Record Table Management");

frame.setSize(600,400);

frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

frame.setLayout(new BorderLayout());


JTextArea textArea=new JTextArea();

textArea.setEditable(false);


try{ String query="SELECT * FROM records"; // Change to records table

Statement stmt=connection.createStatement();

ResultSet rs=stmt.executeQuery(query);


StringBuffer result=new StringBuffer("Records:\n");
while(rs.next()){

result.append("Record ID: ").append(rs.getInt("RecordID"))

.append(", Concert ID: ").append(rs.getInt("ConcertID"))

.append(", Artist Name: ").append(rs.getString("ArtistName"))

.append(", Venue Name: ").append(rs.getString("VenueName"))

.append(", Ticket Sold: ").append(rs.getInt("TicketSold"))

.append(", Concert Date: ").append(rs.getDate("ConcertDate"))

.append(", Record Status: ").append(rs.getString("RecordStatus"))

.append("\n");}

textArea.setText(result.toString());


rs.close();

stmt.close();

}

catch(SQLException exc){

JOptionPane.showMessageDialog(null,"Error executing query: "+exc.getMessage(),"Database
Error",JOptionPane.ERROR_MESSAGE);

return;}


JScrollPane scrollPane=new JScrollPane(textArea);

frame.add(scrollPane,BorderLayout.CENTER);


JPanel buttonPanel=new JPanel();

JButton insertbutton=new JButton("Insert");

JButton updatebutton=new JButton("Update");

JButton deletebutton=new JButton("Delete");

```

```

JButton okButton=new JButton("OK");

buttonPanel.add(insertbutton);
buttonPanel.add(updatebutton);
buttonPanel.add(deletebutton);
buttonPanel.add(okButton);
frame.add(buttonPanel, BorderLayout.SOUTH);

insertbutton.addActionListener(e->insertRecord());
updatebutton.addActionListener(e->updateRecord());
deletebutton.addActionListener(e->deleteRecord());

okButton.addActionListener(e->{ frame.dispose();
    new JFrame().setVisible(true);});
frame.setVisible(true);}

private void insertRecord(){
    String concertIDStr=JOptionPane.showInputDialog(null,"Enter the ConcertID:");
    String artistName=JOptionPane.showInputDialog(null,"Enter the Artist Name:");
    String venueName=JOptionPane.showInputDialog(null,"Enter the Venue Name:");
    String ticketSoldStr=JOptionPane.showInputDialog(null,"Enter the Number of Tickets Sold:");
    String concertDateStr=JOptionPane.showInputDialog(null,"Enter Concert Date(yyyy-mm-dd):");
    String recordStatus="COMPLETED";
    if(concertIDStr==null||artistName==null||venueName==null||ticketSoldStr==null||concertDateStr==null) {
        JOptionPane.showMessageDialog(null,"Operation canceled.", "Cancel",JOptionPane.INFORMATION_MESSAGE);
        return;}

    try{int concertID=Integer.parseInt(concertIDStr);
        int ticketSold=Integer.parseInt(ticketSoldStr);
        java.sql.Date concertDate=java.sql.Date.valueOf(concertDateStr);

        String query="INSERT INTO records (ConcertID, ArtistName, VenueName, TicketSold, ConcertDate, RecordStatus) VALUES (?,
        ?, ?, ?, ?, ?)";

        PreparedStatement pstmt=connection.prepareStatement(query);
        pstmt.setInt(1,concertID);
        pstmt.setString(2,artistName);
        pstmt.setString(3,venueName);
        pstmt.setInt(4,ticketSold);
        pstmt.setDate(5,concertDate);
        pstmt.setString(6,recordStatus);
        pstmt.executeUpdate();
    }
}

```

```

        JOptionPane.showMessageDialog(null,"Record inserted successfully!");
    }
    catch(NumberFormatException exc){
        JOptionPane.showMessageDialog(null,"Ticket Sold must be a valid number!","Input Error",JOptionPane.ERROR_MESSAGE);
    }
    catch(SQLException exc){
        JOptionPane.showMessageDialog(null,"Error inserting record: "+exc.getMessage(),"Database
        Error",JOptionPane.ERROR_MESSAGE);
    }
}

private void updateRecord(){
    String recordIDStr=JOptionPane.showInputDialog(null,"Enter RecordID to update:");
    String newConcertIDStr=JOptionPane.showInputDialog(null,"Enter new ConcertID:");
    String newArtistName=JOptionPane.showInputDialog(null,"Enter new Artist Name:");
    String newVenueName=JOptionPane.showInputDialog(null,"Enter new Venue Name:");
    String newTicketSoldStr=JOptionPane.showInputDialog(null,"Enter new Number of Tickets Sold:");
    String newConcertDateStr=JOptionPane.showInputDialog(null,"Enter new Concert Date (yyyy-mm-dd):");

    if(recordIDStr==null||newConcertIDStr==null||newArtistName==null||newVenueName==null||newTicketSoldStr==null||newConcertDateStr
    ==null) {
        JOptionPane.showMessageDialog(null,"Operation canceled.", "Cancel",JOptionPane.INFORMATION_MESSAGE);
        return; }

    try{int recordID=Integer.parseInt(recordIDStr);
        int newConcertID=Integer.parseInt(newConcertIDStr);
        int newTicketSold=Integer.parseInt(newTicketSoldStr);
        java.sql.Date newConcertDate=java.sql.Date.valueOf(newConcertDateStr);

        String query="UPDATE records SET ConcertID = ?, ArtistName = ?, VenueName = ?, TicketSold = ?, ConcertDate = ? WHERE
        RecordID = ?";

        PreparedStatement pstmt=connection.prepareStatement(query);

        pstmt.setInt(1,newConcertID);
        pstmt.setString(2,newArtistName);
        pstmt.setString(3,newVenueName);
        pstmt.setInt(4,newTicketSold);
        pstmt.setDate(5,newConcertDate);
        pstmt.setInt(6,recordID);
        pstmt.executeUpdate();

        JOptionPane.showMessageDialog(null,"Record updated successfully!");
    }
}

```

```

        catch(NumberFormatException exc){
            JOptionPane.showMessageDialog(null,"Ticket Sold must be a valid number!", "Input Error",JOptionPane.ERROR_MESSAGE);
        }
        catch(SQLException exc){
            JOptionPane.showMessageDialog(null,"Error updating record: "+exc.getMessage(),"Database
Error",JOptionPane.ERROR_MESSAGE);
        }
    }

    private void deleteRecord(){
        String recordIDStr=JOptionPane.showInputDialog(null,"Enter the RecordID to delete:");

        if(recordIDStr==null){
            JOptionPane.showMessageDialog(null,"Operation canceled.", "Cancel",JOptionPane.INFORMATION_MESSAGE);
            return;
        }

        try{String query="DELETE FROM records WHERE RecordID = ?";
            PreparedStatement pstmt=connection.prepareStatement(query);
            pstmt.setInt(1,Integer.parseInt(recordIDStr));
            pstmt.executeUpdate();

            JOptionPane.showMessageDialog(null,"Record deleted successfully!");
        }
        catch(SQLException exc){
            JOptionPane.showMessageDialog(null,"Error deleting record: "+exc.getMessage(),"Database
Error",JOptionPane.ERROR_MESSAGE);
        }
    }

}

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
JFrame frame=new JFrame("Producer Table Management");
        frame.setSize(600,400);
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        frame.setLayout(new BorderLayout());

        JTextArea textArea=new JTextArea();
        textArea.setEditable(false);
        try{String query="SELECT * FROM producer";
            Statement stmt=connection.createStatement();
            ResultSet rs=stmt.executeQuery(query);
            StringBuilder result=new StringBuilder("Producers:\n");

```

```

while(rs.next()){
    result.append("ID: ").append(rs.getInt("ProducerID"))
        .append(", First Name: ").append(rs.getString("FirstName"))
        .append(", Last Name: ").append(rs.getString("LastName"))
        .append(", Number of Produced Albums: ").append(rs.getInt("NumofProdAlbums"))
        .append("\n");}

    textArea.setText(result.toString());
    rs.close();
    stmt.close();}
catch(SQLException exc){
    JOptionPane.showMessageDialog(this,"Error executing query: "+exc.getMessage(),"Database Error",JOptionPane.ERROR_MESSAGE);
    return;
}

JScrollPane scrollPane=new JScrollPane(textArea);
frame.add(scrollPane,BorderLayout.CENTER);

JPanel buttpanel=new JPanel();
JButton insertbutton=new JButton("Insert");
JButton updatebutton=new JButton("Update");
JButton deletebutton=new JButton("Delete");
JButton okButton=new JButton("OK");
buttpanel.add(insertbutton);
buttpanel.add(updatebutton);
buttpanel.add(deletebutton);
buttpanel.add(okButton);
frame.add(buttpanel,BorderLayout.SOUTH);
insertbutton.addActionListener(e->insertProducer());
updatebutton.addActionListener(e->updateProducer());
deletebutton.addActionListener(e->deleteProducer());
okButton.addActionListener(e->{
    frame.dispose();

    new JFrame().setVisible(true);
});
frame.setVisible(true);
}

private void insertProducer(){
    String firstName=JOptionPane.showInputDialog(this,"Enter Producer First Name:");
    String lastName=JOptionPane.showInputDialog(this,"Enter Producer Last Name:");

```

```

String numofProdAlbumsStr=JOptionPane.showInputDialog(this,"Enter Number of Produced Albums:");
if(firstName==null||lastName==null||numofProdAlbumsStr==null){
    JOptionPane.showMessageDialog(this,"Operation canceled.", "Cancel",JOptionPane.INFORMATION_MESSAGE);
    return;
}
try{ int numofProdAlbums=Integer.parseInt(numofProdAlbumsStr);
    String query="INSERT INTO producer (FirstName, LastName, NumofProdAlbums) VALUES (?, ?, ?)";
    PreparedStatement pstmt=connection.prepareStatement(query);
    pstmt.setString(1,firstName);
    pstmt.setString(2,lastName);
    pstmt.setInt(3,numofProdAlbums);
    pstmt.executeUpdate();
    JOptionPane.showMessageDialog(this,"The Producer inserted successfully!");
}
catch(NumberFormatException exc){
    JOptionPane.showMessageDialog(this,"Number of Produced Albums must be a valid number!", "Input
Error",JOptionPane.ERROR_MESSAGE);
}
catch(SQLException exc){
    JOptionPane.showMessageDialog(this,"Error inserting producer: "+exc.getMessage(), "Database
Error",JOptionPane.ERROR_MESSAGE);
}
}}
private void updateProducer(){
    String producerIDStr=JOptionPane.showInputDialog(this,"Enter ProducerID to update:");
    String newFirstName=JOptionPane.showInputDialog(this,"Enter new First Name:");
    String newLastName=JOptionPane.showInputDialog(this,"Enter new Last Name:");
    String newNumofProdAlbumsStr=JOptionPane.showInputDialog(this,"Enter new Number of Produced Albums:");
    if(producerIDStr==null||newFirstName==null||newLastName==null||newNumofProdAlbumsStr==null){
        JOptionPane.showMessageDialog(this,"Operation canceled.", "Cancel",JOptionPane.INFORMATION_MESSAGE);
        return;
    }
    try{ int producerID=Integer.parseInt(producerIDStr);
        int newNumofProdAlbums=Integer.parseInt(newNumofProdAlbumsStr);
        String query="UPDATE producer SET FirstName=?, LastName=?, NumofProdAlbums=? WHERE ProducerID=?";
        PreparedStatement pstmt=connection.prepareStatement(query);
        pstmt.setString(1,newFirstName);
        pstmt.setString(2,newLastName);
        pstmt.setInt(3,newNumofProdAlbums);
        pstmt.setInt(4,producerID);
        pstmt.executeUpdate();
        JOptionPane.showMessageDialog(this,"Producer updated successfully!");
    }
}

```

```

    }

    catch(NumberFormatException exc){

        JOptionPane.showMessageDialog(this,"Number of Produced Albums must be a valid number!", "Input
        Error",JOptionPane.ERROR_MESSAGE);

    }

    catch(SQLException exc){

        JOptionPane.showMessageDialog(this,"Error updating producer: "+exc.getMessage(),"Database
        Error",JOptionPane.ERROR_MESSAGE);

    }
}

private void deleteProducer(){

    String producerIDStr=JOptionPane.showInputDialog(null,"Enter ProducerID to delete:");

    if(producerIDStr==null){

        JOptionPane.showMessageDialog(null,"Operation canceled.", "Cancel",JOptionPane.INFORMATION_MESSAGE);

        return;

    }

    try{ String query="DELETE FROM producer WHERE ProducerID=?";

        PreparedStatement pstmt=connection.prepareStatement(query);

        pstmt.setInt(1,Integer.parseInt(producerIDStr));

        pstmt.executeUpdate();

        JOptionPane.showMessageDialog(null,"Producer deleted successfully!");

    }

    catch(SQLException exc){

        JOptionPane.showMessageDialog(null,"Error deleting producer: "+exc.getMessage(),"Database
        Error",JOptionPane.ERROR_MESSAGE);

    }

}

}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

JFrame frame=new JFrame("Venue Table Management");

frame.setSize(900,600);

frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

frame.setLayout(new BorderLayout());

JTextArea textArea=new JTextArea();

textArea.setEditable(false);

try{ String query="SELECT * FROM venue";

    Statement stmt=connection.createStatement();

    ResultSet rs=stmt.executeQuery(query);

    StringBuilder res=new StringBuilder("Venues:\n");

    while(rs.next()){

```



```

        res.append("VenueID: ").append(rs.getInt("VenueID"))
        .append(", Name: ").append(rs.getString("Name"))
        .append(", Capacity: ").append(rs.getInt("Capacity"))
        .append(", CompletedCon: ").append(rs.getInt("CompletedCon"))
        .append(", YearsOperation: ").append(rs.getInt("YearsOperation"))
        .append("\n"); }textArea.setText(res.toString());

rs.close();

stmt.close();

}

catch(SQLException exc){

    JOptionPane.showMessageDialog(this,"Error executing query: "+exc.getMessage(),"Database Error",JOptionPane.ERROR_MESSAGE);

    return;}

JScrollPane scrollPane=new JScrollPane(textArea);

frame.add(scrollPane,BorderLayout.CENTER);

JPanel buttpanel=new JPanel();

JButton insertbutton=new JButton("Insert");

JButton updatebutton=new JButton("Update");

JButton deletebutton=new JButton("Delete");

JButton okButton=new JButton("OK");

buttpanel.add(insertbutton);

buttpanel.add(updatebutton);

buttpanel.add(deletebutton);

buttpanel.add(okButton);

frame.add(buttpanel,BorderLayout.SOUTH);

insertbutton.addActionListener(e->insertVenue());

updatebutton.addActionListener(e->updateVenue());

deletebutton.addActionListener(e->deleteVenue());

okButton.addActionListener(e->{ frame.dispose();

    new JFrame().setVisible(true);});

frame.setVisible(true);

}

private void insertVenue(){

    String name=JOptionPane.showInputDialog(this,"Enter the Venue Name:");

    String capacityStr=JOptionPane.showInputDialog(this,"Enter the Venue Capacity:");

    String yearsOperationStr=JOptionPane.showInputDialog(this,"Enter the Years of Operation:");

    if(name==null||capacityStr==null||yearsOperationStr==null){

```

```

        JOptionPane.showMessageDialog(this,"Operation canceled.", "Cancel",JOptionPane.INFORMATION_MESSAGE);
    }

    try{
        String query="INSERT INTO venue (Name, Capacity, YearsOperation) VALUES (?, ?, ?)";
        PreparedStatement pstmt=connection.prepareStatement(query);

        pstmt.setString(1,name);
        pstmt.setInt(2,Integer.parseInt(capacityStr));
        pstmt.setInt(3,Integer.parseInt(yearsOperationStr));
        pstmt.executeUpdate();

        JOptionPane.showMessageDialog(this,"Venue inserted successfully!");
    }

    catch(SQLException exc){

        JOptionPane.showMessageDialog(this,"Error inserting venue: "+exc.getMessage(),"Database
        Error",JOptionPane.ERROR_MESSAGE);
    }
}

private void updateVenue(){
    String venueIDStr=JOptionPane.showInputDialog(this,"Enter the VenueID to update:");
    String name=JOptionPane.showInputDialog(this,"Enter the new Venue Name:");
    String capacityStr=JOptionPane.showInputDialog(this,"Enter the new Venue Capacity:");
    String yearsOperationStr=JOptionPane.showInputDialog(this,"Enter new Years of Operation:");

    if(venueIDStr==null||venueIDStr.trim().isEmpty()||name==null||capacityStr==null||yearsOperationStr==null){
        JOptionPane.showMessageDialog(this,"Operation canceled.", "Cancel",JOptionPane.INFORMATION_MESSAGE);
        return;
    }

    try{ String query="UPDATE venue SET Name=?, Capacity=?, YearsOperation=? WHERE VenueID=?";

        PreparedStatement pstmt=connection.prepareStatement(query);

        pstmt.setString(1,name);
        pstmt.setInt(2,Integer.parseInt(capacityStr));
        pstmt.setInt(3,Integer.parseInt(yearsOperationStr));
        pstmt.setInt(4,Integer.parseInt(venueIDStr));
        pstmt.executeUpdate();

        JOptionPane.showMessageDialog(this,"Venue updated successfully!");
    }

    catch(SQLException exc){

        JOptionPane.showMessageDialog(this,"Error updating venue: "+exc.getMessage(),"Database
        Error",JOptionPane.ERROR_MESSAGE);
    }
}

private void deleteVenue(){
    String venueIDStr=JOptionPane.showInputDialog(this,"Enter the VenueID to delete:");

    if(venueIDStr==null||venueIDStr.trim().isEmpty()){
        JOptionPane.showMessageDialog(this,"Operation canceled.", "Cancel",JOptionPane.INFORMATION_MESSAGE);
        return;
    }
}

```

```

try{ String query="DELETE FROM venue WHERE VenueID=?";

    PreparedStatement pstmt=connection.prepareStatement(query);

    pstmt.setInt(1,Integer.parseInt(venueIDStr));

    pstmt.executeUpdate();

    JOptionPane.showMessageDialog(this,"Venue deleted successfully!");

}

catch(SQLException exc){

    JOptionPane.showMessageDialog(this,"Error deleting venue: "+exc.getMessage(),"Database Error",JOptionPane.ERROR_MESSAGE);

}

}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

int response = JOptionPane.showConfirmDialog(this, "Are you sure you want to exit?", "Exit Confirmation",
JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE);

if (response==JOptionPane.YES_OPTION) {

    System.exit(0);

}

}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {

    /* Set the Nimbus look and feel */

    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">

    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
     * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
     */

    try {

        for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

            if ("Nimbus".equals(info.getName())) {

                javax.swing.UIManager.setLookAndFeel(info.getClassName());

                break;

            }

        }

    } catch (ClassNotFoundException ex) {

        java.util.logging.Logger.getLogger(NewJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    }
}

```

```

    } catch (InstantiationException ex) {
        java.util.logging.Logger.getLogger(NewJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {
        java.util.logging.Logger.getLogger(NewJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
        java.util.logging.Logger.getLogger(NewJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    }
}

//</editor-fold>

//java.awt.EventQueue.invokeLater(() -> new NewJFrame().setVisible(true));

/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
        new NewJFrame().setVisible(true);
    }
});

}

// Variables declaration - do not modify
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton10;
private javax.swing.JButton jButton11;
private javax.swing.JButton jButton12;
private javax.swing.JButton jButton13;
private javax.swing.JButton jButton14;
private javax.swing.JButton jButton15;
private javax.swing.JButton jButton16;
private javax.swing.JButton jButton17;
private javax.swing.JButton jButton18;
private javax.swing.JButton jButton19;
private javax.swing.JButton jButton2;
private javax.swing.JButton jButton3;
private javax.swing.JButton jButton4;
private javax.swing.JButton jButton5;
private javax.swing.JButton jButton6;
private javax.swing.JButton jButton7;
private javax.swing.JButton jButton8;
private javax.swing.JButton jButton9;
private javax.swing.JInternalFrame jInternalFrame1;
private javax.swing.JLabel jLabel1;

// End of variables declaration

// Declare UI components

```

```
private javax.swing.JTextField titleField;  
private javax.swing.JTextField artistIDField;  
private javax.swing.JTextField genreIDField;  
private javax.swing.JTextField companyIDField;  
private javax.swing.JTextField producerIDField;  
private javax.swing.JButton insertAlbumButton;  
private javax.swing.JButton exitButton;  
}
```