Name: Rashi Girish Chopdekar

Classical music

CONCERT

25

Symphony

sample text

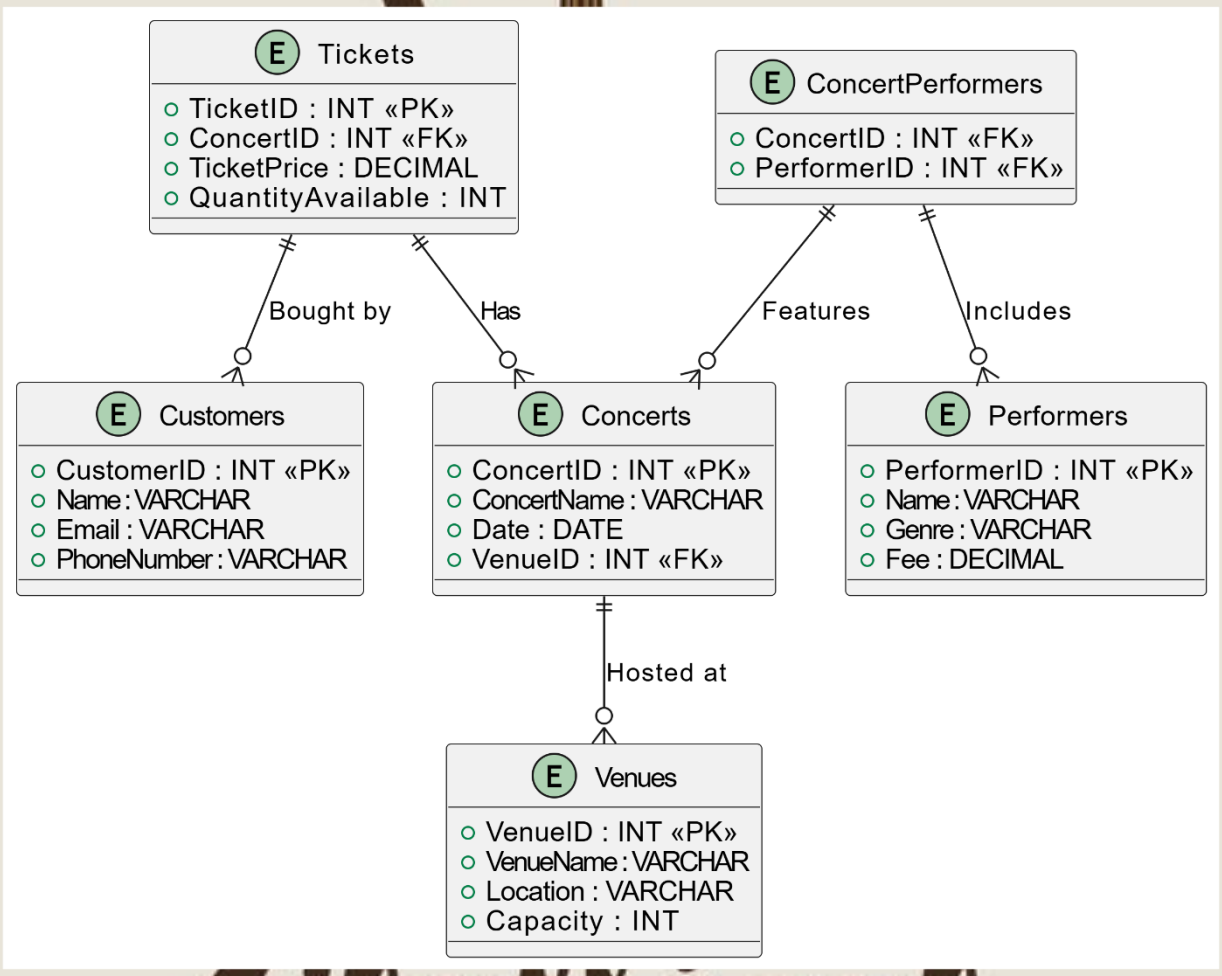
MUSIC CONCERT TICKETING SYSTEM

INTRODUCTION

The Music Concert Ticketing System is a database-driven application designed to streamline and manage the ticketing process for music concerts. It serves as a comprehensive solution for concert organizers, performers, and customers, enabling efficient handling of ticket sales, venue management, and performer schedules. This project showcases the use of relational database concepts to address the real-world needs of a ticketing platform.

The system is built on a structured database schema with interconnected tables to store and manage data about concerts, venues, tickets, customers, performers, and purchases. By leveraging SQL queries, the project facilitates tasks such as retrieving concert details, tracking ticket sales, analyzing customer data, and assigning performers to events.

ER DIAGRAM



sample text

CREATION OF THE DATA

- Create Database

```
show databases;
create database project;
use project;
```

- Create Tables

Used to define the database schema by creating tables like Concerts, Venues, Tickets, Customers, Performers, etc., along with primary and foreign key relationships. Inserted 20+ sample records into each table to populate the database with data about concerts, venues, tickets, customers, and purchases.

- Venues

```
CREATE TABLE Venues (
    VenueID INT PRIMARY KEY,
    VenueName VARCHAR(100),
    Location VARCHAR(100),
    Capacity INT
);

INSERT INTO Venues (VenueID, VenueName, Location, Capacity) VALUES
(1, 'Green Park Arena', 'Downtown, NY', 5000),
(2, 'Sunset Stadium', 'Riverside, NY', 10000),
(3, 'City Hall', 'Central City, NY', 3000),
(4, 'Oceanview Theatre', 'Ocean Drive, NY', 2000),
(5, 'Downtown Plaza', 'Market Square, NY', 1500),
(6, 'Mountain Arena', 'High Peak, NY', 4000);
```


Symphony

sample text

- Concert

```sql
CREATE TABLE Concerts (
    ConcertID INT PRIMARY KEY,
    ConcertName VARCHAR(100),
    Date DATE,
    VenueID INT,
    FOREIGN KEY (VenueID) REFERENCES Venues(VenueID)
);
```

```sql
INSERT INTO Concerts (ConcertID, ConcertName, Date, VenueID) VALUES
(1, 'Summer Music Fest', '2024-07-01', 1),
(2, 'Rock the Night', '2024-07-10', 2),
(3, 'Classical Evening', '2024-07-15', 3),
(4, 'Indie Vibes', '2024-08-01', 1),
(5, 'Jazz Night', '2024-08-05', 4),
(6, 'Pop Hits Concert', '2024-08-10', 2),
(7, 'Electronic Waves', '2024-08-12', 3),
(8, 'Blues & Roots', '2024-08-15', 5),
(9, 'Country Music Bash', '2024-08-18', 6),
(10, 'Hip Hop Legends', '2024-08-22', 2),
(11, 'Dance Party', '2024-09-01', 1),
(12, 'Reggae Fest', '2024-09-05', 4),
(13, 'Orchestra Concert', '2024-09-10', 3),
(14, 'Acoustic Night', '2024-09-12', 5),
(15, 'Metal Fest', '2024-09-15', 6),
(16, 'Latin Beats', '2024-09-20', 2),
(17, 'R&B Under the Stars', '2024-09-25', 4),
(18, 'Alternative Sounds', '2024-09-30', 3),
(19, 'Live Acoustic', '2024-10-05', 5),
(20, 'Pop Rock Show', '2024-10-10', 1);
```

- Tickets

```sql
CREATE TABLE Tickets (
    TicketID INT PRIMARY KEY,
    ConcertID INT,
    TicketPrice DECIMAL(10, 2),
    QuantityAvailable INT,
    FOREIGN KEY (ConcertID) REFERENCES Concerts(ConcertID)
);
```

```sql
INSERT INTO Tickets (TicketID, ConcertID, TicketPrice, QuantityAvailable) VALUES
(1, 1, 50, 200),
(2, 2, 70, 250),
(3, 3, 60, 150),
(4, 4, 40, 300),
(5, 5, 80, 100),
(6, 6, 55, 180),
(7, 7, 45, 220),
(8, 8, 65, 120),
(9, 9, 50, 500),
(10, 10, 75, 150),
(11, 11, 60, 200),
(12, 12, 90, 180),
(13, 13, 100, 120),
(14, 14, 65, 250),
(15, 15, 50, 300),
(16, 16, 60, 220),
(17, 17, 70, 200),
(18, 18, 55, 150),
(19, 19, 40, 350),
(20, 20, 65, 180);
```

- Customers

```sql
CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY,
    Name VARCHAR(100),
    Email VARCHAR(100),
    PhoneNumber VARCHAR(15)
);
```

```sql
INSERT INTO Customers (CustomerID, Name, Email, PhoneNumber) VALUES
(1, 'John Doe', 'john@example.com', '555-1234'),
(2, 'Jane Smith', 'jane@example.com', '555-5678'),
(3, 'Alice Johnson', 'alice@example.com', '555-8765'),
(4, 'Bob Brown', 'bob@example.com', '555-4321'),
(5, 'Charlie Green', 'charlie@example.com', '555-9876'),
(6, 'Dave White', 'dave@example.com', '555-6543'),
(7, 'Emma Blue', 'emma@example.com', '555-1357'),
(8, 'Frank Black', 'frank@example.com', '555-2468'),
(9, 'Grace Pink', 'grace@example.com', '555-3579'),
(10, 'Hannah Red', 'hannah@example.com', '555-4820'),
(11, 'Ian Purple', 'ian@example.com', '555-5921'),
(12, 'Jack Grey', 'jack@example.com', '555-6132'),
(13, 'Kevin White', 'kevin@example.com', '555-7243'),
(14, 'Lily Orange', 'lily@example.com', '555-8354'),
(15, 'Mike Yellow', 'mike@example.com', '555-9465'),
(16, 'Nora Violet', 'nora@example.com', '555-0576'),
(17, 'Oliver Brown', 'oliver@example.com', '555-1687'),
(18, 'Paula Green', 'paula@example.com', '555-2798'),
(19, 'Quinn Black', 'quinn@example.com', '555-3809'),
(20, 'Rachel Silver', 'rachel@example.com', '555-4910');
```

- Purchases

```sql
CREATE TABLE Purchases (
    PurchaseID INT PRIMARY KEY,
    TicketID INT,
    CustomerID INT,
    PurchaseDate DATE,
    Quantity INT,
    FOREIGN KEY (TicketID) REFERENCES Tickets(TicketID),
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);
```

```sql
INSERT INTO Purchases (PurchaseID, TicketID, CustomerID, PurchaseDate, Quantity) VALUES
(1, 1, 1, '2024-06-10', 2),
(2, 2, 2, '2024-06-15', 1),
(3, 3, 3, '2024-06-18', 3),
(4, 4, 4, '2024-06-20', 4),
(5, 5, 5, '2024-06-22', 2),
(6, 6, 6, '2024-06-25', 5),
(7, 7, 7, '2024-06-27', 2),
(8, 8, 8, '2024-06-30', 3),
(9, 9, 9, '2024-07-02', 1),
(10, 10, 10, '2024-07-04', 2),
(11, 11, 11, '2024-07-07', 3),
(12, 12, 12, '2024-07-10', 1),
(13, 13, 13, '2024-07-12', 2),
(14, 14, 14, '2024-07-15', 3),
(15, 15, 15, '2024-07-17', 2),
(16, 16, 16, '2024-07-20', 1),
(17, 17, 17, '2024-07-22', 4),
(18, 18, 18, '2024-07-25', 5),
(19, 19, 19, '2024-07-28', 3),
(20, 20, 20, '2024-07-30', 2);
```

- Performers

```sql
CREATE TABLE Performers (
    PerformerID INT PRIMARY KEY,
    Name VARCHAR(100),
    Genre VARCHAR(50),
    Fee DECIMAL(10, 2)
);
```

```sql
INSERT INTO Performers (PerformerID, Name, Genre, Fee) VALUES
(1, 'The Rockers', 'Rock', 5000),
(2, 'Jazz Masters', 'Jazz', 7000),
(3, 'Electro Beats', 'Electronic', 4500),
(4, 'The Classics', 'Classical', 6000),
(5, 'Indie Vibes', 'Indie', 4000),
(6, 'Pop Stars', 'Pop', 8000);
```

- Concert_Performers

```
CREATE TABLE Concert_Performers (
    ConcertID INT,
    PerformerID INT,
    PRIMARY KEY (ConcertID, PerformerID),
    FOREIGN KEY (ConcertID) REFERENCES Concerts(ConcertID),
    FOREIGN KEY (PerformerID) REFERENCES Performers(PerformerID)
);
```

```
INSERT INTO Concert_Performers (ConcertID, PerformerID) VALUES
(1, 1),
(2, 2),
(3, 4),
(4, 5),
(5, 2),
(6, 6),
(7, 3),
(8, 4),
(9, 1),
(10, 6),
(11, 5),
(12, 2),
(13, 4),
(14, 1),
(15, 6),
(16, 3),
(17, 5),
(18, 6),
(19, 2),
(20, 4);
```

DESCRIBE

```
desc venues;
desc customers;
desc concerts;
desc performers;
desc purchases;
desc concert_performers;
desc tickets;
```

- Desc Venues;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
VenueID	int	NO	PRI	NULL	
VenueName	varchar(100)	YES		NULL	
Location	varchar(100)	YES		NULL	
Capacity	int	YES		NULL	

- Desc customers;

Field	Type	Null	Key	Default	Extra
CustomerID	int	NO	PRI	NULL	
Name	varchar(100)	YES		NULL	
Email	varchar(100)	YES		NULL	
PhoneNumber	varchar(15)	YES		NULL	

- Desc Concerts;

Result Grid | Filter Rows: | Export: | Wrap Ce

Field	Type	Null	Key	Default	Extra
ConcertID	int	NO	PRI	NULL	
ConcertName	varchar(100)	YES		NULL	
Date	date	YES		NULL	
VenueID	int	YES	MUL	NULL	

- Desc Performers;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
PerformerID	int	NO	PRI	NULL	
Name	varchar(100)	YES		NULL	
Genre	varchar(50)	YES		NULL	
Fee	decimal(10,2)	YES		NULL	

- Desc Purchases;

Result Grid | Filter Rows: | Export: | Wrap

Field	Type	Null	Key	Default	Extra
PurchaseID	int	NO	PRI	NULL	
TicketID	int	YES	MUL	NULL	
CustomerID	int	YES	MUL	NULL	
PurchaseDate	date	YES		NULL	
Quantity	int	YES		NULL	

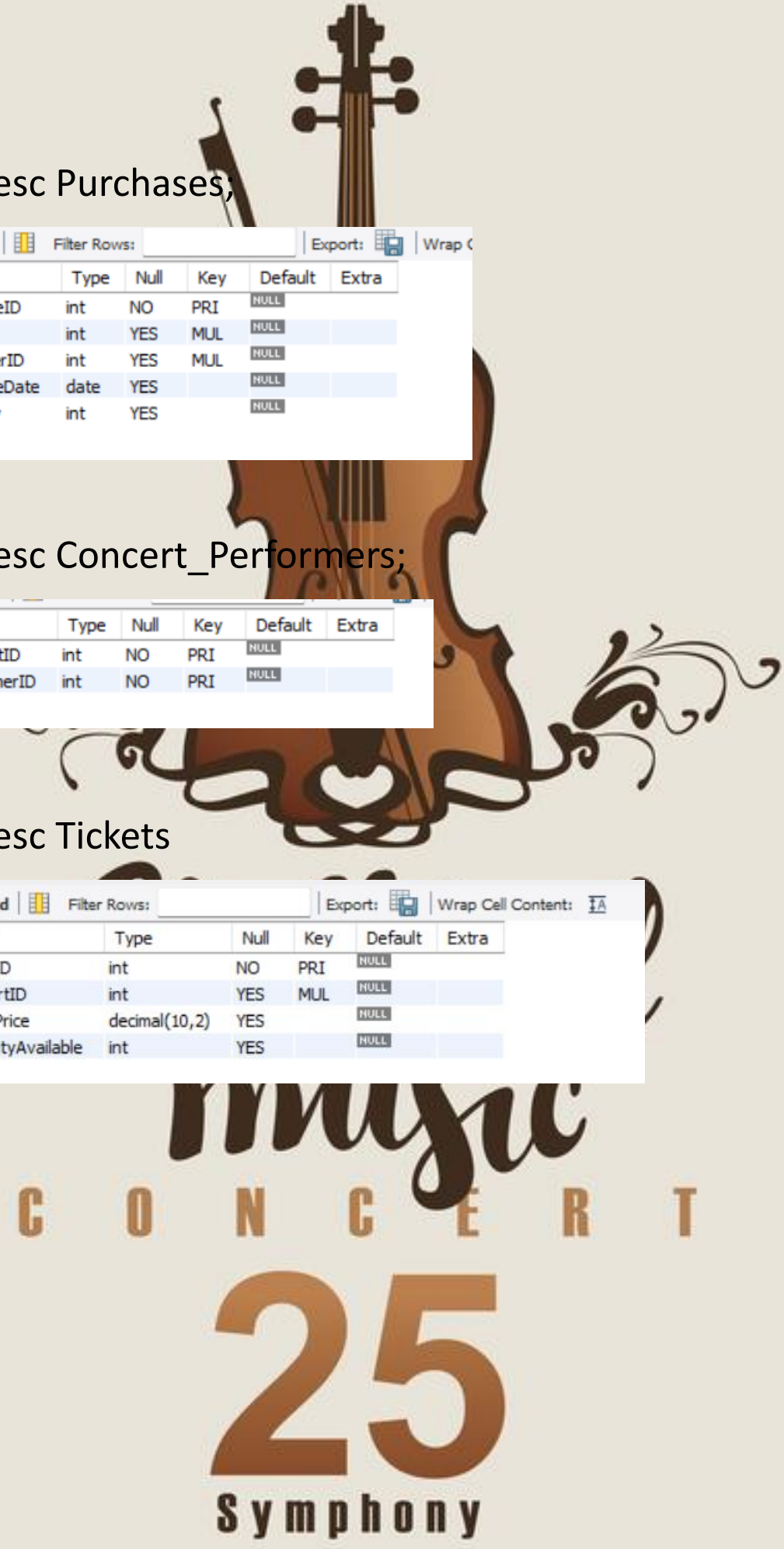
- Desc Concert_Performers;

Field	Type	Null	Key	Default	Extra
ConcertID	int	NO	PRI	NULL	
PerformerID	int	NO	PRI	NULL	

- Desc Tickets

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
TicketID	int	NO	PRI	NULL	
ConcertID	int	YES	MUL	NULL	
TicketPrice	decimal(10,2)	YES		NULL	
QuantityAvailable	int	YES		NULL	



Select Statements

- select * from venues;

VenueID	VenueName	Location	Capacity
1	Green Park Arena	Downtown, NY	5000
2	Sunset Stadium	Riverside, NY	10000
3	City Hall	Central City, NY	3000
4	Oceanview Theatre	Ocean Drive, NY	2000
5	Downtown Plaza	Market Square, NY	1500
6	Mountain Arena	High Peak, NY	4000
NULL	NULL	NULL	NULL

- select * from customers;

CustomerID	Name	Email	PhoneNumber
1	John Doe	john@example.com	555-1234
2	Jane Smith	jane@example.com	555-5678
3	Alice Johnson	alice@example.com	555-8765
4	Bob Brown	bob@example.com	555-4321
5	Charlie Green	charlie@example.com	555-9876
6	Dave White	dave@example.com	555-6543
7	Emma Blue	emma@example.com	555-1357
8	Frank Black	frank@example.com	555-2468
9	Grace Pink	grace@example.com	555-3579
10	Hannah Red	hannah@example.com	555-4820
11	Ian Purple	ian@example.com	555-5921
12	Jack Grey	jack@example.com	555-6132
13	Kevin White	kevin@example.com	555-7243
14	Lily Orange	lily@example.com	555-8354
15	Mike Yellow	mike@example.com	555-9465
15	Mike Yellow	mike@example.com	555-9465
16	Nora Violet	nora@example.com	555-0576
17	Oliver Brown	oliver@example.com	555-1687
18	Paula Green	paula@example.com	555-2798
19	Quinn Black	quinn@example.com	555-3809
20	Rachel Silver	rachel@example.com	555-4910
NULL	NULL	NULL	NULL

- select * from concerts;

ConcertID	ConcertName	Date	VenueID
1	Summer Music Fest	2024-07-01	1
2	Rock the Night	2024-07-10	2
3	Classical Evening	2024-07-15	3
4	Indie Vibes	2024-08-01	1
5	Jazz Night	2024-08-05	4
6	Pop Hits Concert	2024-08-10	2
7	Electronic Waves	2024-08-12	3
8	Blues & Roots	2024-08-15	5
9	Country Music Bash	2024-08-18	6
10	Hip Hop Legends	2024-08-22	2
11	Dance Party	2024-09-01	1
12	Reggae Fest	2024-09-05	4
13	Orchestra Concert	2024-09-10	3
14	Acoustic Night	2024-09-12	5
15	Metal Fest	2024-09-15	6
16	Latin Beats	2024-09-20	2
16	Latin Beats	2024-09-20	2
17	R&B Under the Stars	2024-09-25	4
18	Alternative Sounds	2024-09-30	3
19	Live Acoustic	2024-10-05	5
20	Pop Rock Show	2024-10-10	1
NULL	NULL	NULL	NULL

- select * from performers;

PerformerID	Name	Genre	Fee
1	The Rockers	Rock	5000.00
2	Jazz Masters	Jazz	7000.00
3	Electro Beats	Electronic	4500.00
4	The Classics	Classical	6000.00
5	Indie Vibes	Indie	4000.00
6	Pop Stars	Pop	8000.00
NULL	NULL	NULL	NULL

- Select * from purchases;

PurchaseID	TicketID	CustomerID	PurchaseDate	Quantity
1	1	1	2024-06-10	2
2	2	2	2024-06-15	1
3	3	3	2024-06-18	3
4	4	4	2024-06-20	4
5	5	5	2024-06-22	2
6	6	6	2024-06-25	5
7	7	7	2024-06-27	2
8	8	8	2024-06-30	3
9	9	9	2024-07-02	1
10	10	10	2024-07-04	2
11	11	11	2024-07-07	3
12	12	12	2024-07-10	1
13	13	13	2024-07-12	2
14	14	14	2024-07-15	3
15	15	15	2024-07-17	2
16	16	16	2024-07-20	1
16	16	16	2024-07-20	1
17	17	17	2024-07-22	4
18	18	18	2024-07-25	5
19	19	19	2024-07-28	3
20	20	20	2024-07-30	2
NULL	NULL	NULL	NULL	NULL

- select * from concert_performers;

ConcertID	PerformerID
1	1
9	1
14	1
2	2
5	2
12	2
19	2
7	3
16	3
3	4
8	4
13	4
20	4
4	5
11	5
17	5
17	5
6	6
10	6
15	6
18	6
NULL	NULL

- select * from tickets;

TicketID	ConcertID	TicketPrice	QuantityAvailable
1	1	110.00	200
2	2	70.00	250
3	3	60.00	150
4	4	40.00	300
5	5	80.00	100
6	6	55.00	180
7	7	45.00	220
8	8	65.00	120
9	9	50.00	500
10	10	75.00	150
11	11	60.00	200
12	12	90.00	180
13	13	100.00	120
14	14	65.00	250
15	15	50.00	300
16	16	60.00	220
16	16	60.00	220
17	17	70.00	200
18	18	55.00	150
19	19	40.00	350
20	20	65.00	180
NULL	NULL	NULL	NULL

Show tables;

Result Grid | Filter Rows:

Tables_in_project
concert_performers
concerts
customers
performers
purchases
tickets
venues

QUERIES

1. UPDATE

Modified existing data, e.g., updating ticket prices or venue capacities to reflect changes.

```
/* Update*/
UPDATE Tickets
SET TicketPrice = 110.00
WHERE ConcertID = 1;
```

TicketID	ConcertID	TicketPrice	QuantityAvailable
1	1	110.00	200

2. DELETE

Removed specific records from the database, such as canceling a concert or deleting a customer's data.

```
/* Delete */
DELETE FROM Customers
WHERE CustomerID NOT IN (SELECT DISTINCT CustomerID FROM Purchases);
```

Field	Type	Null	Key	Default	Extra
CustomerID	int	NO	PRI	NULL	
Name	varchar(100)	YES		NULL	
Email	varchar(100)	YES		NULL	
PhoneNumber	varchar(15)	YES		NULL	

3. Select

Extracted specific data, such as listing all concerts, finding performer details, or retrieving ticket prices.

```sql
/*SELECT*/
SELECT * FROM Concerts
WHERE Date BETWEEN '2024-07-01' AND '2024-07-31';
```

Result Grid | Filter Rows: | Edit:

ConcertID	ConcertName	Date	VenueID
1	Summer Music Fest	2024-07-01	1
2	Rock the Night	2024-07-10	2
3	Classical Evening	2024-07-15	3
NULL	NULL	NULL	NULL

4. Where

Filtered records based on conditions, e.g., finding concerts in a specific venue or tickets above a certain price.

```sql
/*Where*/
SELECT VenueName, Location
FROM Venues
WHERE Capacity > 750;
```

VenueName	Location
Green Park Arena	Downtown, NY
Sunset Stadium	Riverside, NY
City Hall	Central City, NY
Oceanview Theatre	Ocean Drive, NY
Downtown Plaza	Market Square, NY
Mountain Arena	High Peak, NY

5. Arithmetic, Comparison, and Logical Operator

Performed calculations and comparisons, e.g., finding concerts with ticket prices greater than $50 or checking combined conditions using AND/OR.

```sql
/*Arithmetic, Comparison, and Logical Operator*/
SELECT ConcertID, TicketPrice * QuantityAvailable AS TotalEarnings
FROM Tickets
WHERE TicketPrice > 100 AND QuantityAvailable > 0;
```

Result Grid | Filter Rows:

ConcertID	TotalEarnings
1	22000.00

6. Range Operator

Queried concerts scheduled within a specific date range or tickets within a price range.

```
/*Range Operator*/
SELECT ConcertName, Date
FROM Concerts
WHERE Date BETWEEN '2024-06-15' AND '2024-07-15';
```

ConcertName	Date
Summer Music Fest	2024-07-01
Rock the Night	2024-07-10
Classical Evening	2024-07-15

7. List Operator

Filtered records with specific values, e.g., retrieving performers belonging to multiple genres.

```
/*List Operator*/
SELECT ConcertName
FROM Concerts
WHERE VenueID IN (1, 3, 5);
```

ConcertName
Summer Music Fest
Classical Evening
Indie Vibes
Electronic Waves
Blues & Roots
Dance Party
Orchestra Concert
Acoustic Night
Alternative Sounds
Live Acoustic
Pop Rock Show

8. LIKE Operator

Searched for patterns, e.g., finding customers whose names start with a particular letter.

```
/*LIKE Operator*/
SELECT Name, Genre
FROM Performers
WHERE Genre LIKE 'J%';
```

Name	Genre
Jazz Masters	Jazz

9. ORDER BY

Sorted data, e.g., listing concerts by date.

```
/*ORDER BY*/
SELECT ConcertName, Date
FROM Concerts
ORDER BY Date DESC;
```

ConcertName	Date
Pop Rock Show	2024-10-10
Live Acoustic	2024-10-05
Alternative Sounds	2024-09-30
R&B Under the Stars	2024-09-25
Latin Beats	2024-09-20
Metal Fest	2024-09-15
Acoustic Night	2024-09-12
Orchestra Concert	2024-09-10
Reggae Fest	2024-09-05
Dance Party	2024-09-01
Hip Hop Legends	2024-08-22
Country Music Bash	2024-08-18

ConcertName	Date
Dance Party	2024-09-01
Hip Hop Legends	2024-08-22
Country Music Bash	2024-08-18
Blues & Roots	2024-08-15
Electronic Waves	2024-08-12
Pop Hits Concert	2024-08-10
Jazz Night	2024-08-05
Indie Vibes	2024-08-01
Classical Evening	2024-07-15
Rock the Night	2024-07-10
Summer Music Fest	2024-07-01

10. DISTINCT

Retrieved unique values, such as unique genres.

```
/*DISTINCT*/
SELECT DISTINCT Genre
FROM Performers;
```

Genre
Rock
Jazz
Electronic
Classical
Indie
Pop

11. TOP/LIMIT

Fetched limited records, e.g., the top 5 most expensive tickets.

```
/*TOP*/
SELECT TicketPrice
FROM Tickets
ORDER BY TicketPrice DESC
LIMIT 3;
```

TicketPrice
110.00
100.00
90.00

12. IS NULL

Identified missing or present values, e.g., concerts without assigned venues.

```
/*IS NULL*/
SELECT ConcertID, ConcertName, Date
FROM Concerts
WHERE VenueID IS NULL;
```

Result Grid | Filter Rows:

ConcertID	ConcertName	Date
NULL	NULL	NULL

13. IS NOT NULL

Identified missing or present values, e.g., concerts without assigned venues.

```
/*IS NOT NULL*/
SELECT CustomerID, Name, Email
FROM Customers
WHERE CustomerID IS NOT NULL;
```

CustomerID	Name	Email
1	John Doe	john@example.com
2	Jane Smith	jane@example.com
3	Alice Johnson	alice@example.com
4	Bob Brown	bob@example.com
5	Charlie Green	charlie@example.com
6	Dave White	dave@example.com
7	Emma Blue	emma@example.com
8	Frank Black	frank@example.com
9	Grace Pink	grace@example.com
10	Hannah Red	hannah@example.com
11	Ian Purple	ian@example.com
12	Jack Grey	jack@example.com

12	Jack Grey	jack@example.com
13	Kevin White	kevin@example.com
14	Lily Orange	lily@example.com
15	Mike Yellow	mike@example.com
16	Nora Violet	nora@example.com
17	Oliver Brown	oliver@example.com
18	Paula Green	paula@example.com
19	Quinn Black	quinn@example.com
20	Rachel Silver	rachel@example.com
NULL	NULL	NULL

14. CASE Statement

```
/*CASE Statement*/
SELECT ConcertID,
       TicketPrice,
       CASE
           WHEN TicketPrice < 50 THEN 'Budget'
           WHEN TicketPrice BETWEEN 50 AND 100 THEN 'Standard'
           ELSE 'Premium'
       END AS PriceCategory
FROM Tickets;
```

ConcertID	TicketPrice	PriceCategory
1	110.00	Premium
2	70.00	Standard
3	60.00	Standard
4	40.00	Budget
5	80.00	Standard
6	55.00	Standard
7	45.00	Budget
8	65.00	Standard
9	50.00	Standard
10	75.00	Standard
11	60.00	Standard
12	90.00	Standard
13	100.00	Standard
14	65.00	Standard
15	50.00	Standard

15	50.00	Standard
16	60.00	Standard
17	70.00	Standard
18	55.00	Standard
19	40.00	Budget
20	65.00	Standard

15. String Function

```
/*String Function*/
SELECT CONCAT(Name, ' - ', Genre) AS PerformerDetails
FROM Performers;
```

PerformerDetails
The Rockers - Rock
Jazz Masters - Jazz
Electro Beats - Electronic
The Classics - Classical
Indie Vibes - Indie
Pop Stars - Pop

16. Math Function

```
/*Math Function*/
SELECT ROUND(AVG(TicketPrice), 2) AS AverageTicketPrice
FROM Tickets;
```

Result Grid | Filter Row

AverageTicketPrice
65.25

17. Date Function

```
/*Date Function*/
SELECT ConcertName, Date
FROM Concerts
WHERE Date >= CURDATE() AND Date <= DATE_ADD(CURDATE(), INTERVAL 30 DAY);
```

Result Grid | Filter Rows:

ConcertName	Date

18. Comparison Functions

```sql
/*Comparison Functions*/
SELECT MAX(TicketPrice) AS HighestPrice, MIN(TicketPrice) AS LowestPrice
FROM Tickets;
```

HighestPrice	LowestPrice
110.00	40.00

19. Aggregate Function

Used SUM, AVG, COUNT, etc., to calculate totals, averages, or counts, e.g., total tickets sold or average ticket price.

- Sum

```sql
/*Aggregate Function*/
SELECT SUM(Quantity) AS TotalTicketsSold
FROM Purchases;
```

TotalTicketsSold
51

- Avg

```sql
/*Aggregate Function*/
SELECT avg(Quantity) AS TotalTicketsSold
FROM Purchases;
```

Result Grid | Filte

TotalTicketsSold
2.5500

20. GROUP BY with HAVING

Grouped data and applied conditions to groups, e.g., finding venues with total capacities exceeding a threshold.

```sql
/*GROUP BY with HAVING*/
SELECT Location, SUM(Capacity) AS TotalCapacity
FROM Venues
GROUP BY Location
HAVING SUM(Capacity) > 1000;
```

Result Grid | Filter Rows:

Location	TotalCapacity
Downtown, NY	5000
Riverside, NY	10000
Central City, NY	3000
Ocean Drive, NY	2000
Market Square, NY	1500
High Peak, NY	4000

21. Joins

Connected data across multiple tables, e.g., listing performers for each concert using JOIN queries.

```sql
/*Joins*/
SELECT c.ConcertName, p.Name AS PerformerName, p.Genre
FROM Concerts c
JOIN Concert_Performers cp ON c.ConcertID = cp.ConcertID
JOIN Performers p ON cp.PerformerID = p.PerformerID;
```

ConcertName	PerformerName	Genre
Summer Music Fest	The Rockers	Rock
Country Music Bash	The Rockers	Rock
Acoustic Night	The Rockers	Rock
Rock the Night	Jazz Masters	Jazz
Jazz Night	Jazz Masters	Jazz
Reggae Fest	Jazz Masters	Jazz
Live Acoustic	Jazz Masters	Jazz
Electronic Waves	Electro Beats	Electronic
Latin Beats	Electro Beats	Electronic
Classical Evening	The Classics	Classical
Blues & Roots	The Classics	Classical
Orchestra Concert	The Classics	Classical

Orchestra Concert	The Classics	Classical
Pop Rock Show	The Classics	Classical
Indie Vibes	Indie Vibes	Indie
Dance Party	Indie Vibes	Indie
R&B Under the Stars	Indie Vibes	Indie
Pop Hits Concert	Pop Stars	Pop
Hip Hop Legends	Pop Stars	Pop
Metal Fest	Pop Stars	Pop
Alternative Sounds	Pop Stars	Pop

22. Sub-Queries

Executed nested queries, e.g., finding concerts with the highest ticket prices or customers who made multiple purchases.

```sql
/*Sub-Queries*/
SELECT VenueName
FROM Venues
WHERE VenueID = (
    SELECT VenueID
    FROM Concerts
    WHERE ConcertID = (
        SELECT ConcertID
        FROM Tickets
        ORDER BY TicketPrice DESC
        LIMIT 1
    )
);
```

VenueName
Green Park Arena

23. Modifying the Structure of a Table

Altered table structures, e.g., adding new columns to store additional information like concert ratings.

```sql
/*Modifying the Structure of a Table*/
ALTER TABLE Tickets
ADD TicketType VARCHAR(20);
```

Result Grid | Filter Rows: | Edit:

TicketID	ConcertID	TicketPrice	QuantityAvailable	TicketType
1	1	110.00	200	NULL
2	2	70.00	250	NULL
3	3	60.00	150	NULL
4	4	40.00	300	NULL
5	5	80.00	100	NULL
6	6	55.00	180	NULL
7	7	45.00	220	NULL
8	8	65.00	120	NULL
9	9	50.00	500	NULL
10	10	75.00	150	NULL
11	11	60.00	200	NULL
12	12	90.00	180	NULL

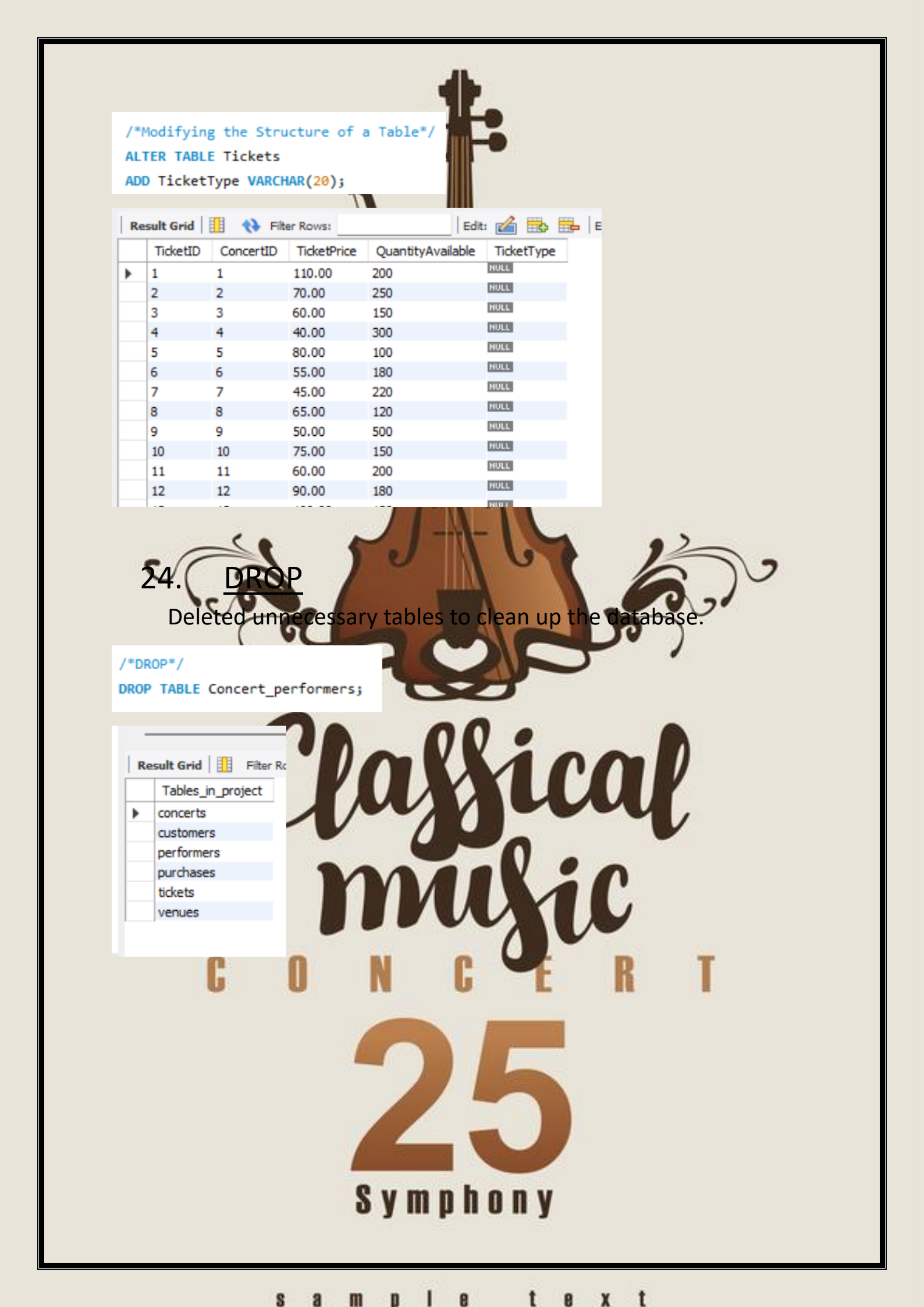
24. DROP

Deleted unnecessary tables to clean up the database.

```sql
/*DROP*/
DROP TABLE Concert_performers;
```

Result Grid | Filter Rc

Tables_in_project
concerts
customers
performers
purchases
tickets
venues



THANK YOU

Classical music

CONCERT

25

Symphony

sample text