# Group Project Team 12482_1

The concert management company needs to effectively track all aspects of their operations. The company owns several venues with different names, capacities, locations, and different main points of contact for rental inquiries. The venues grant spaces for multiple vendors of different names, types, and permits to sell items in their respective stalls that range in flat rental charges and sizes. The company also has a designated team of staff members that the company wants to store information about; the company wants to store information about their names, emails, phone numbers, addresses, and pay rates. The concert management team would also like to analyze the shifts of each staff member and retain data about the start and end time of the shifts.

Next, the concert management team would also like to save information about agents who manage the performers. An agent can have many performers under them. The company wants to save the names and phone numbers of the agents and the name of the performers they manage. Design a data model to accurately depict the process of concert management. With each performer, they will fall in a specific genre category. With each concert that will be performed, the company wants to track the name, concert date, time of the concert, and opener for the concert.

Finally, the company wants to track attendee name, contact, and credit card numbers. The attendees can purchase one or many tickets from different platforms, with different costs, and pick their seat numbers. The concert will feature multiple performers and can take place at multiple venues. Create a data model that represents the entire process of planning the concert, from purchasing tickets and managing logistics to booking performers and managing staff.

# Team Members
- [@Grace Fazzone](https://github.com/gracefazzone/MIST-4610-Group-Project)
- [@Annie Li](https://github.com/anniewli/annieli)
- [@Young Kim](https://github.com/Youngiyoung/Group-Project)
- [@William McBride](https://github.com/WilliamMcB23/William-McBride-Project-1---MIST-4610)
- [@Gustav Bringle](https://github.com/gustavbringle/Sky)

# Data Model

![Logo](https://github.com/gracefazzone/MIST-4610-Group-Project/blob/main/dm1.png?raw=true
)

## Data Dictionary

```sql

```

Table: Agent

Column Name	Description	Data Type	Size	Format	Key
agentID	Unique Agent Identifier	INT			PK
fName	The agent's first name	Text	45		
lName	The agent's last name	Text	45		
phone	The agent's phone number	Text	20	999-999-9999	

Table: Attendee

Column Name	Description	Data Type	Size	Format	Key
attendeeID	Unique Attendee Identifier	INT			PK
firstName	The attendee's first name	Text	45		
lastName	The attendee's last name	Text	45		
email	The attendee's email	Text	45		
phone	The attendee's phone number	Text	20	999-999-9999	
creditCardNumber	The attendee's credit card information	Text	16		

Table: Company

Column Name	Description	Data Type	Size	Format	Key
plannerID	Unique Planner Identifier	INT			PK
companyName	The name of the company the planner is with	Text	45		

Table: Concert

Column Name	Description	Data Type	Size	Format	Key
concertID	Unique Concert Identifier	INT			PK
concertDate	The date of the concert	Date	10	YYYY-MM-DD	
concertTime	The concert start time	Time		HH-MM-SS	
opener	The opening performer	Text	45		
performerID	Unique Performer Identifier	INT			FK
venueID	Unique Venue Identifier	INT			FK

Table: Genre

Column Name	Description	Data Type	Size	Format	Key
genreID	Unique Genre Identifier	INT			PK
genreName	The music genre name	Text	45		

Table: Performer

Column Name	Description	Data Type	Size	Format	Key
performerID	Unique Performer Identifier	INT			PK

performerName	The name of the performer	Text	45		
agentID	Unique Agent Identifier	INT			FK
genreID	Unique Genre Identifier	INT			FK

Table: Shift

Column Name	Description	Data Type	Size	Format	Key
shiftID	Unique Shift Identifier	INT			PK
shiftBegin	The shift start time	Time	8	HH:MM:SS	
shiftEnd	The shift end time	Time	8	HH:MM:SS	
staffID	Unique Staff Identifier	INT			FK
venueID	Unique Venue Identifier	INT			FK

Table: Staff

Column Name	Description	Data Type	Size	Format	Key
staffID	Unique Staff Identifier	INT			PK
firstName	The staff's first name	Text	45		
lastName	The staff's last name	Text	45		
email	The staff's email	Text	45		
phone	The staff's phone number	Text	20		
streetAddress	The staff's street address	Text	45		
city	The staff's city of residence	Text	45		
state	The staff's state of residence	Text	45		
zipCode	The staff's residence zip code	Text	45		
payRate	The staff's pay rate per hour	Double	5		XX.XX
plannerID	Unique Planner Identifier	INT			FK

Table: Stall

Column Name	Description	Data Type	Size	Format	Key
stallID	Unique Stall Identifier	INT			PK
flatRentalCharge	The stall's flat rental charge	Double		XXXXX.XX	
squareMeters	The stall's size in square meters	Double		XX.XX	
venueID	Unique Venue Identifier	INT			FK
vendorID	Unique Vendor Identifier	INT			FK

Table: Ticket

Column Name	Description	Data Type	Size	Format	Key
ticketID	Unique Ticket Identifier	INT			PK
ticketPlatform	The platform that sells tickets	Text	45		
ticketCost	The cost of the ticket	Double		XXX.XX	
seatNumber	The seat number on ticket	Text	5		

attendeeID	Unique Attendee Identifier	INT			FK
concertID	Unique Concert Identifier	INT			FK

Table: Vendor

Column Name	Description	Data Type	Size	Format	Key
vendorID	Unique Vendor Identifier	INT			PK
vendorName	The vendor's name	Text	45		
vendorPermit	The permit vendor requires	Text	45		
vendorType	The vendor's type of service or goods	Text	45		

Table: Venue

Column Name	Description	Data Type	Size	Format	Key
venueID	Unique Venue Identifier	INT			PK
venueName	The venue's name	Text	45		
location	The venue's location	Text	45		
capacity	The venue's maximum capacity	INT			
mainContact	The main contact's name that helps book the venue	Text	45		
plannerID	Unique Planner Identifier	INT			FK

# SQL Queries

```sql
Query 1
Description: Write a query to list out each venue and the most expensive ticket offered that was
sold on Ticketmaster.

SELECT venueName, CONCAT(" $ ",ROUND(MAX(ticketCost),2)) AS 'Most Expensive Venue
Ticket'
FROM Venue
JOIN Concert ON Venue.venueID = Concert.venueID
JOIN Ticket ON Concert.concertID = Ticket.concertID
WHERE ticketPlatform = "Ticketmaster"
GROUP BY venueName
ORDER BY MAX(ticketCost) DESC;

Justification: The company managing the different venues would want to know the most
expensive ticket price for each venue to help their ticket pricing strategy
since they can assess demand for further revenue optimization analysis

Query 2
```

Description: Write a query to list out the concert genres and the amount of business each has generated that is over $500.

```
SELECT genreName, CONCAT(" $ ",ROUND(SUM(ticketCost),2)) AS 'Business Generated'
FROM Genre
JOIN Performer ON Genre.genreID = Performer.genreID
JOIN Concert ON Performer.PerformerID = Concert.PerformerID
JOIN Ticket ON Concert.concertID = Ticket.concertID
GROUP BY genreName
HAVING SUM(ticketCost) > 500
ORDER BY SUM(ticketCost) DESC;
```

Justification: Management would want to know data about the amount of business generated by each genre to assess which to
focus on as revenue aligns with consumer demands. The 500 filter gives better data of top genres without outliers for the purpose of simple data analysis.

Query 3
Description: Write a query to list the first and last name of the attendees that purchased more than one ticket, the total cost of all of their tickets, and order by highest to lowest total cost.

```
SELECT firstName, lastName, COUNT(ticketID) AS "Number of Tickets", SUM(ticketCost) AS
"Total Cost"
FROM Attendee
JOIN Ticket ON Attendee.attendeeID = Ticket.attendeeID
GROUP BY firstName, lastName
HAVING COUNT(ticketID) > 1
ORDER BY SUM(ticketCost) DESC;
```

Justification: This query identifies customers who have bought more than one ticket for concerts, which is good for management to know for customer segmentation.
These customers may frequently go to many concerts or buy multiple tickets at a time and require a different approach to marketing and communication.
This query also lets us know the total money they have spent on all of their tickets, which provides a clear understanding of how much revenue they are generating from each customer.
This provides an opportunity to identify high-value customers and maybe reward these customers or provide incentives for them to keep buying tickets.

Query 4
Description: Write a query to list out first and last name of attendee and ticket cost if the ticket cost is greater than the average ticket cost for that attendee.

```
SELECT firstName, lastName, ticketCost
FROM Attendee
```

```
JOIN Ticket ON Attendee.attendeeID = Ticket.attendeeID
WHERE ticketCost > (SELECT AVG(ticketCost) FROM Ticket WHERE Attendee.attendeeID =
Ticket.attendeeID);
```

Justification: This query identifies customers that are paying more for tickets than their average ticket cost.
This can also help management identify those customers that are willing to pay more for tickets, see what concerts they are willing to pay more for, and personalize experiences and incentives for these customers.
This would hopefully drive higher engagement for these ticket platforms and increase sales. Along with that, knowing what concerts people are willing to pay more for can help them evaluate their pricing strategy and see if there are opportunities to adjust pricing to maximize revenue.

Query 5
Description: Write a query to display the average ticket price for each venue and order by highest to lowest average ticket cost.

```
SELECT venueName, ROUND(AVG(ticketCost), 2) AS 'Average Ticket Price'
FROM Venue
JOIN Concert ON Venue.venueID = Concert.venueID
JOIN Ticket ON Concert.concertID = Ticket.concertID
GROUP BY venueName
ORDER BY AVG(ticketCost) DESC;
```

Justification: This query selects the name of each venue along with the average ticket price for all concerts held at that venue. To accomplish this, it joins three tables - Venue, Concert, and Ticket - based on their respective ID fields.
By using the AVG function, it calculates the average ticket price for each venue and orders the results in descending order based on that average.
This query can be useful for concert promoters, venue managers, and other stakeholders to get an overview of the average ticket prices for different venues, helping them make informed decisions about where to hold future concerts or events.

Query 6
Description: Write a query to display the names of all the bands that have performed at a venue with a capacity greater than 20,000 people.

```
SELECT DISTINCT Performer.performerName
FROM Performer
JOIN Concert ON Performer.performerID = Concert.performerID
JOIN Venue ON Concert.venueID = Venue.venueID
WHERE Venue.capacity > 20000;
```

Justification: The given query selects the distinct names of performers who have performed at venues with a capacity greater than 20000 people by joining the Performer, Concert, and Venue tables based on their respective IDs. This query can be useful for event organizers and talent agencies who are looking for performers that have a history of performing at larger venues. By filtering data based on venue capacity, organizers and agencies can identify performers who are more likely to be suitable for their upcoming events or future bookings.

Query 7
Description: Write a query that states which tickets have a value greater than $50 that were purchased for the Red Rocks Theatre venue, and order by ascending ticket cost.

```
SELECT ticketID, venueName, ticketCost
FROM Venue
JOIN Concert ON Concert.venueID = Venue.venueID
JOIN Ticket ON Concert.concertID = Ticket.concertID
WHERE venueName = 'Red Rocks Amphitheater' AND ticketCost > 50
ORDER BY ticketCost;
```

JUSTIFICATION: This query lists which tickets were purchased at Red Rocks Amphitheatre which had a price of over $50. To accomplish this, it joins three tables - Venue, Concert, and Ticket - based on their respective ID fields. By using the having function, it helped show the respective tickets which had a price greater than $50. This query can be useful for ticket companies, Red Rocks management team, and customers to see how much people are paying for tickets at this specific venue.

Query 8
Description: Write a query that lists the name of the agents and the date of the concerts associated with the performer who performs Country music between the dates January 1st 2022 to January 1st 2023.

```
SELECT CONCAT(fName,'  ', lName) AS 'Agent Name', concertDate
FROM Agent
JOIN Performer ON Agent.agentID = Performer.agentID
JOIN Genre ON Performer.genreID = Genre.genreID
JOIN Concert ON Performer.performerID = Concert.performerID
WHERE genreName = 'Country' AND (concertDate BETWEEN '2022-01-01' AND '2023-01-01');
```

Justification: This query lists the first and last names of the agents who are associated with a performer who performs country music between 01/01/22 and 01/01/23.
To accomplish this, it joins four tables - Agent, Performer, Genre and Concert - based on their respective ID fields.
By using the Between function, it helped show the country concerts associated with the agent between the listed date range.

This query can be useful for concert performers, venue managers, and other people invested to get an overview of the agents who are working with country artists within the span of a year.

Query 9
Description: Write a query to list out the names and the pay rate of the staff if the pay rate is greater than the average pay rate of all staff from Las Vegas.

```
SELECT firstName, lastName, CONCAT(“$”, payRate) AS “Pay Rate”
FROM Staff
WHERE city = "Las Vegas" AND payRate > (SELECT AVG(payRate) FROM Staff);
```

Justification: This query retrieves the first and last names of staff members who work in Las Vegas and have a pay rate higher than the average pay rate of all staff members.
Additionally, it concatenates the pay rate with a dollar sign for presentation purposes. This query is useful for identifying high-performing staff members in a specific location and determining if they are being compensated appropriately.
By comparing their pay rate to the average, it helps identify if they are being paid above or below market rate.

Query 10
Description: Write a query to display which ticket platform offered tickets under $30 for concerts in 2022 and list out the ticket prices to those concerts and the venue names.

```
SELECT Ticket.ticketPlatform, CONCAT(“$”,Ticket.ticketCost) AS “Ticket Cost”, concertDate,
venueName
FROM Ticket
JOIN Concert ON Concert.concertID = Ticket.concertID
JOIN Venue ON Venue.venueID = Concert.venueID
WHERE ticketCost < 30
AND concertDate REGEXP "^2022";
```

Justification: This query selects data from three tables, Ticket, Concert, and Venue, by joining them based on their IDs.
The query filters the data by only selecting tickets with a cost of less than 30 and concerts that have a date in the year 2022.
The selected columns from the tables include the ticket platform, ticket cost, concert date, and venue name.
This query will provide a list of all concerts in 2022 that have tickets costing less than $30, along with their venue information.

```

## Query Results
```

Query 1:

![App Screenshot](https://github.com/gracefazzone/MIST-4610-Group-Project/blob/main/TP_Q1().png?raw=true)

Query2:

![App Screenshot](https://github.com/gracefazzone/MIST-4610-Group-Project/blob/main/TP_Q2().png?raw=true)

Query3:

![App Screenshot](https://github.com/gracefazzone/MIST-4610-Group-Project/blob/main/TP_Q3().png?raw=true)

Query 4:

![App Screenshot](https://github.com/gracefazzone/MIST-4610-Group-Project/blob/main/TP_Q4().png?raw=true)

Query 5:

![App Screenshot](https://github.com/gracefazzone/MIST-4610-Group-Project/blob/main/TP_Q5().png?raw=true)

Query 6:

![App Screenshot](https://github.com/gracefazzone/MIST-4610-Group-Project/blob/main/TP_Q6().png?raw=true)

Query 7:

![App Screenshot](https://github.com/gracefazzone/MIST-4610-Group-Project/blob/main/TP_Q7().png?raw=true)

Query 8:

![App Screenshot](https://github.com/gracefazzone/MIST-4610-Group-Project/blob/main/TP_Q8().png?raw=true)

Query 9:

![App Screenshot](https://github.com/gracefazzone/MIST-4610-Group-Project/blob/main/TP_Q9().png?raw=true)

Query 10:

![App Screenshot](https://github.com/gracefazzone/MIST-4610-Group-Project/blob/main/TP_Q10().png?raw=true)

## SQL Query Matrix

![App Screenshot](https://github.com/gracefazzone/MIST-4610-Group-Project/blob/main/Screenshot%20(144).png?raw=true)