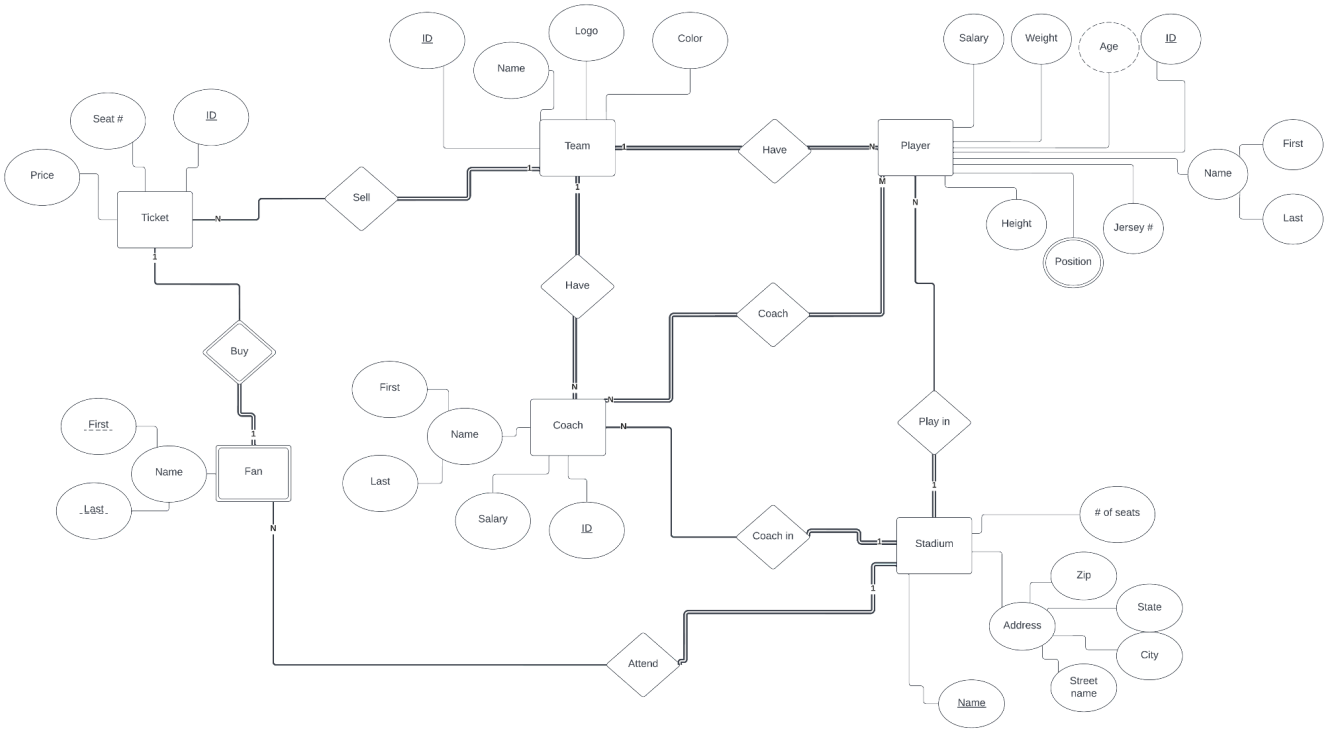
NBA Tickets Database

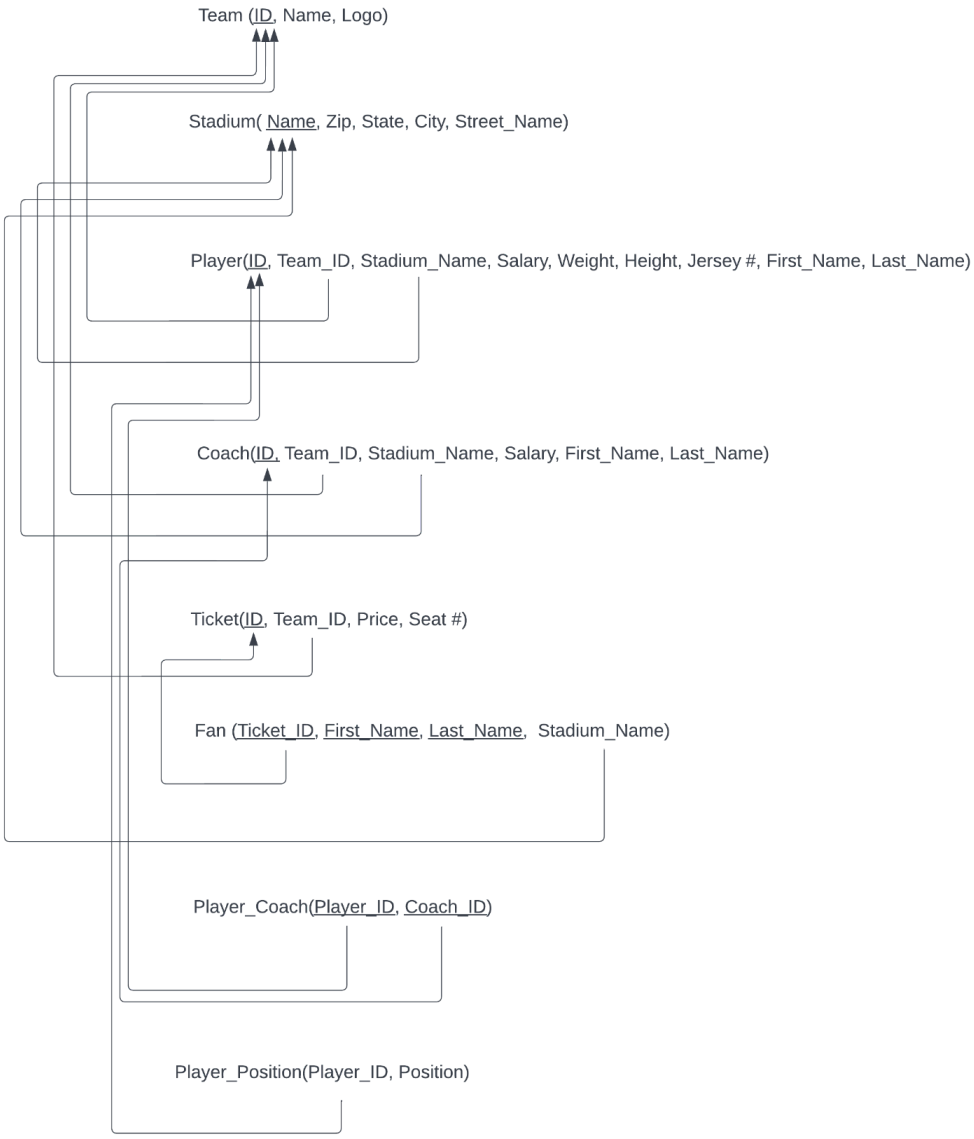
This database was designed to track information about teams, players and most importantly ticket sales in the NBA.Since there are so many players being traded in the NBA and so many fans come and watch NBA games in-person, It is important to keep track of all of that data. This is important for smooth entertainment. Many NBA team general managers would like to keep track of which player is playing in what team and executives of the NBA would also like to keep track of ticket sales to determine if their business is going up or down. Teams would also like to use this database to organize their ticket sales and information to print out their sold tickets. This also includes the type of ticket it is, for when a fan does decide to buy a ticket with a certain price they may get certain benefits while at the game due to the cost of the ticket. Not only that but general managers would also like to see how much they are paying their players as a reflection of their value/skill to the team (positions the players can play). Same goes for coaches in how many people they are coaching to determine their value/skill as a coach.

The business logic for this database is as follows. A team belongs to one stadium, players belong to one team, coaches belong to one team, tickets for sale (sold by that one team) and fans of that team who buy their tickets. Not all tickets will always be sold (depending on the team), to reflect a sold ticket a fan would be added to the database and they would be assigned to a ticket to trace back who bought the ticket and where that fan will be seated. From there you can access the information about the team/players/coaches/stadium according to the team who sold that ticket. Once a team wins a championship their popularity will dramatically increase so with a higher demand to watch the team, their ticket prices will be updated.

Entity Relationship Diagram



Relational Schema



Description of Tables

```
DROP TABLE IF EXISTS Player_Position;
DROP TABLE IF EXISTS Player_Coach;
DROP TABLE IF EXISTS Fan;
DROP TABLE IF EXISTS Ticket;
DROP TABLE IF EXISTS Coach;
DROP TABLE IF EXISTS Player;
DROP TABLE IF EXISTS Stadium;
DROP TABLE IF EXISTS Team;
```

Table 1: **Team.**
Purpose: **Track Teams basic information.**
Attributes: **ID, Name, Logo, Color.**
Keys: **ID is the primary key**
MySql:

```
CREATE TABLE Team
(
   id CHAR(10) PRIMARY KEY,
   name VARCHAR(25) NOT NULL,
   logo VARCHAR(15) NOT NULL,
   color VARCHAR(15) NOT NULL
);
```

id	name	logo	color
0123456789	Boston Celtics	Leprechaun	Green
0986123458	New York Knicks	Basketball	Orange
1029876678	Los Angeles Lakers	Basketball	Yellow
1234987655	Brooklyn Nets	Basketball	Black
2001996030	LA Clippers	Basketball	Blue
3409573490	Milwuakee Bucks	Buck	Green
3456781029	Golden State Warriors	Bridge	Blue
3459871029	Pheonix Suns	Basketball	Purple
NULL	NULL	NULL	NULL

Table 2: **Stadium.**
Purpose: **Track Stadiums location.**
Attributes: **Name, Zip, State, City, Street_Name.**
Keys: **Name is the Primary Key**
MySql:

```
CREATE TABLE Stadium
(
        name VARCHAR(25) PRIMARY KEY,
   zip_code VARCHAR(5) NOT NULL,
   state CHAR(2) NOT NULL,
   city VARCHAR(15) NOT NULL,
   street_name VARCHAR(25) NOT NULL
);
```

name	zip_code	state	city	street_name
Barclays Center	11217	NY	Brooklyn	620 Atlantic Ave
Chase Center	94158	CA	San Francisco	1 Warriors Way
Crypto.com Arena	90015	CA	Los Angeles	1111 S Figueroa St
Fiserv Forum	53203	WI	Milwuakee	1111 Vel R. Phillips Ave
Footprint Center	85004	AZ	Pheonix	201 E Jefferson St
Madison Square Garden	10001	NY	New York City	4 Pennsylvania Plaza
TD Garden	02114	MA	Boston	100 Legends Way
NULL	NULL	NULL	NULL	NULL

Table 3: **Player.**
Purpose: **Track Players basic information in the NBA.**
Attributes: **ID, Team_ID, Stadium_Name, Salary, Weight, Height, Jersey_Number, first_name, last_name, nickname**
Keys: **ID is the primary key. Team_ID is a foreign key. Stadium_Name is a foreign key.**
MySql:

```
CREATE TABLE Player
(
        id CHAR(10) PRIMARY KEY,
   team_id CHAR(10),
   stadium_name VARCHAR(25),
   salary VARCHAR(10) NOT NULL,
   weight_lbs CHAR(3) NOT NULL,
   height VARCHAR(5) NOT NULL,
   jersey_number CHAR(2) NOT NULL,
   first_name VARCHAR(20) NOT NULL,
   last_name VARCHAR(20) NOT NULL,
   nickname VARCHAR(20) Unique DEFAULT NULL,
   FOREIGN KEY (team_id)
                        REFERENCES Team (id)
```

```
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY (stadium_name)
                    REFERENCES Stadium (name)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

id	team_id	stadium_name	salary	weight_lbs	height	jersey_number	first_name	last_name	nickname
1111222220	1029876678	Crypto.com Arena	44470000	220	6'8"	23	Lebron	James	The King
1902346890	3409573490	Fiserv Forum	39340000	243	6'11"	34	Giannis	Antetokumpo	The Greek Freak
2349054784	3459871029	Footprint Center	31650000	206	6'5"	1	Devin	Booker	Kardashian
2550943939	2001996030	Crypto.com Arena	35450000	220	6'8"	13	Paul	George	Pandemic P
3342340909	1234987655	Barclays Center	33330000	193	6'2"	11	Kyrie	Irving	Uncle Drew
3430820948	0123456789	TD Garden	16610000	220	6'3"	36	Marcus	Smart	NULL
3506832454	3409573490	Fiserv Forum	35500000	234	6'7"	22	Khris	Middleton	NULL
3943985220	1029876678	Crypto.com Arena	13300000	185	6'1"	21	Patrick	Beverly	NULL
3958093485	3456781029	Chase Center	16400000	230	6'6"	23	Draymond	Green	NULL
4095340954	3456781029	Chase Center	32740000	216	6'6"	30	Klay	Thompson	Splash Brother
4095368085	3459871029	Footprint Center	38510000	175	6'0"	3	Chris	Paul	CP3
4445463553	3409573490	Fiserv Forum	11610000	227	6'9"	2	Joe	Ingles	NULL
4542929099	0986123458	Madison Square ...	32450000	214	6'6"	9	RJ	Barret	NULL
4597350944	3459871029	Footprint Center	19258000	234	6'6"	99	Jae	Crowder	NULL
4950230927	0123456789	TD Garden	32600000	209	6'8"	0	Jayson	Tatum	Gold Metal Jay
4958393833	0123456789	TD Garden	24830000	220	6'6"	7	Jalen	Brown	NULL
6543789534	1234987655	Barclays Center	42970000	240	6'10"	7	Kevin	Durant	Easy Money Sni...
6656587493	1234987655	Barclays Center	29000000	240	6'10"	10	Ben	Simmons	NULL
6677994430	3456781029	Chase Center	45780000	185	6'2"	30	Stephen	Curry	The Baby Face ...
6766769938	2001996030	Crypto.com Arena	41250000	209	6'3"	11	John	Wall	Dish
9008343929	0986123458	Madison Square ...	13450000	200	6'3"	4	Derick	Rose	NULL
9080984444	0986123458	Madison Square ...	18900000	249	6'8"	30	Julius	Randle	NULL
9573002290	1029876678	Crypto.com Arena	44210000	200	6'3"	0	Russel	Westbrook	Russel Westbrick
9823532343	2001996030	Crypto.com Arena	39340000	225	6'7"	2	Kawahi	Leonard	The Claw
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Table 4: **Coach.**
Purpose: **Track Coaches basic information.**
Attributes: **ID, Team_ID, Stadium_Name, Salary, First_Name, Last_Name.**
Keys: **ID is the primary key. Team_ID is a foreign key. Stadium_Name is a foreign key.**
MySql:

```
CREATE TABLE Coach
(
        id CHAR(10) PRIMARY KEY,
    team_id CHAR(10),
    stadium_name VARCHAR(25),
    salary VARCHAR(10) NOT NULL,
    first_name VARCHAR(20) NOT NULL,
    last_name VARCHAR(20) NOT NULL,
    FOREIGN KEY (team_id)
```

```
                REFERENCES Team (id)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
        FOREIGN KEY (stadium_name)
                REFERENCES Stadium (name)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

id	team_id	stadium_name	salary	first_name	last_name
0022938305	2001996030	Crypto.com Arena	2000000	Brian	Shaw
1124954879	3459871029	Footprint Center	2750000	Monty	Williams
2534338739	3459871029	Footprint Center	250000	Mark	Bryant
3425432434	1029876678	Crypto.com Arena	5000000	Darvin	Ham
3450982097	0986123458	Madison Square Garden	4000000	Tom	Thibodeau
3597459420	1234987655	Barclays Center	430000	Igor	Kokoskov
4537592209	0986123458	Madison Square Garden	390000	Brett	Siegel
4572972029	0123456789	TD Garden	4600000	Joe	Mazzulla
5552929890	3409573490	Fiserv Forum	1000000	Blaine	Mueller
6598937297	3456781029	Chase Center	4000000	Bruce	Fraser
6875949294	0123456789	TD Garden	360000	Damon	Stoudamire
7486379202	3456781029	Chase Center	9500000	Steve	Kerr
7653090280	1029876678	Crypto.com Arena	460000	Chris	Jent
9089749888	3409573490	Fiserv Forum	8000000	Mike	Budenholzer
9385658783	2001996030	Crypto.com Arena	6000000	Tyronn	Lue
9978765954	1234987655	Barclays Center	9300000	Steve	Nash
NULL	NULL	NULL	NULL	NULL	NULL

Table 5: **Ticket.**
Purpose: **Track Tickets basic information.**
Attributes: **ID, Team_ID, Price, Seat_Number.**
Keys: **ID is the primary key. Team_ID is a foreign key.**
MySql:

```
CREATE TABLE Ticket
(
        id CHAR(10) PRIMARY KEY,
   team_id CHAR(10),
   price VARCHAR(6) NOT NULL,
   seat_number VARCHAR(6) NOT NULL UNIQUE,
   FOREIGN KEY (team_id)
                REFERENCES Team (id)
```

```
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

id	team_id	price	seat_number
0000003979	3409573490	5000	111
0024323900	0986123458	12000	45
1111132322	3409573490	400	888
2342320938	3459871029	900	432
2343422322	2001996030	2000	198
2798384080	0123456789	10000	19
2909479073	1029876678	501	340
3098340939	1234987655	11000	14
3200000243	3459871029	700	321
3255423456	0123456789	450	390
3332429528	3456781029	2000	234
3425464223	1234987655	999	400
4059280979	0123456789	900	233
4448394839	1029876678	500	356
4538637978	3409573490	1000	478
5093450982	3456781029	1000	800
5433523434	0986123458	9000	222
5499845798	1234987655	1200	333
5555545222	2001996030	400	829
5666669829	3459871029	3000	578
5746564532	1234987655	10000	20
8834245244	3456781029	15000	1
8844887832	1029876678	3908	123
8984397290	1029876678	300	987
9898745939	0986123458	1200	190
9923423549	3456781029	15000	12
9999932222	2001996030	300	792
NULL	NULL	NULL	NULL

Table 6: **Fan.**
Purpose: **Track Fans name and stadium they are going to go.**
Attributes: **Ticket_ID, First_Name, Last_Name, Stadium_Name.**
Keys: **Ticket_ID, First_Name, Last_Name is the primary key. Stadium_Name is a foreign key.**
MySql:

```
CREATE TABLE Fan
(
        first_name VARCHAR (20) NOT NULL,
   last_name VARCHAR(20) NOT NULL,
   ticket_id CHAR(10),
```

```
    stadium_name VARCHAR(25),
    PRIMARY KEY (first_name, last_name, ticket_id),
    FOREIGN KEY (ticket_id)
                        REFERENCES Ticket (id)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY (stadium_name)
                        REFERENCES Stadium (name)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

first_name	last_name	ticket_id	stadium_name
Dave	Chappele	3425464223	Barclays Center
Dwayne	Johnson	5746564532	Barclays Center
Kanye	West	5499845798	Barclays Center
Homero	Arellano	8834245244	Chase Center
Kendal	Jenner	9923423549	Chase Center
Marshal	Mathers	3332429528	Chase Center
Billie	Eilish	9999932222	Crypto.com Arena
Cardi	B	4448394839	Crypto.com Arena
Kendrick	Lamar	8844887832	Crypto.com Arena
Lady	Gaga	5555545222	Crypto.com Arena
Miley	Cyrus	2343422322	Crypto.com Arena
Travis	Scott	8984397290	Crypto.com Arena
Bruno	Mars	1111132322	Fiserv Forum
Chief	Keef	0000003979	Fiserv Forum
Emma	Stone	4538637978	Fiserv Forum
DeAndre	Cortez	5666669829	Footprint Center
Jack	Harlow	3200000243	Footprint Center
Tory	Lanes	2342320938	Footprint Center
Ayesha	Curry	5433523434	Madison Square ...
Chloe	Kardashian	0024323900	Madison Square ...
Joe	Rogan	9898745939	Madison Square ...
Barthalam...	Concorv	3255423456	TD Garden
Eddie	Murphy	4059280979	TD Garden
Micheal	Jackson	2798384080	TD Garden
NULL	NULL	NULL	NULL

Table 7: **Player_Coach.**
Purpose: **Tracks which players are coached by each coach.**
Attributes: **Player_ID, Coach_ID.**
Keys: **Player_ID and Coach_ID is the primary key. Player_ID is a foreign key. Coach_ID is a foreign key.**
MySql:

```
CREATE TABLE Player_Coach
(
        player_id CHAR(10),
    coach_id CHAR(10),
    PRIMARY KEY (player_id, coach_id),
    FOREIGN KEY (coach_id)
                        REFERENCES Coach(id)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY (player_id)
                        REFERENCES Player(id)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

player_id	coach_id
1111222220	3425432434
1902346890	9089749888
2349054784	1124954879
2550943939	0022938305
2550943939	9385658783
3342340909	9978765954
3430820948	4572972029
3506832454	9089749888
3943985220	7653090280
3958093485	6598937297
3958093485	7486379202
4095340954	6598937297
4095340954	7486379202
4095368085	1124954879
4445463553	5552929890
4542929099	4537592209
4597350944	2534338739
4950230927	6875949294
4958393833	6875949294
6543789534	9978765954
6656587493	3597459420
6677994430	6598937297
6677994430	7486379202
6766769938	9385658783
9008343929	3450982097
9080984444	4537592209
9573002290	3425432434
9573002290	7653090280
9823532343	9385658783
NULL	NULL

Table 8: **Player_Position**
Purpose: **Tracks which players play what positions**
Attributes: **Player_ID, Position**
Keys: **Player_ID is the primary key. Player_ID is a foreign key.**
MySql:

```
CREATE TABLE Player_Position
(
        player_id CHAR(10),
   position VARCHAR(15),
   PRIMARY KEY (player_id, position),
   FOREIGN KEY (player_id)
                        REFERENCES Player(id)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

player_id	position
1111222220	Power Forward
1111222220	Small Forward
1902346890	Center
1902346890	Forward
1902346890	Power Forward
2349054784	Shooting Guard
2349054784	Small Forward
2550943939	Power Forward
2550943939	Small Forward
3342340909	Point Guard
3342340909	Shooting Guard
3506832454	Forward
3506832454	Power Forward
3506832454	Small Forward
3943985220	Point Guard
3943985220	Shooting Guard
3958093485	Center
3958093485	Power Forward
4095340954	Shooting Guard
4095340954	Small Forward
4095368085	Point Guard
4445463553	Forward
4445463553	Guard
4445463553	Power Forward
4445463553	Small Forward
4542929099	Small Forward
4597350944	Power Forward
4597350944	Small Forward
6543789534	Power Forward
6543789534	Small Forward
6656587493	Point Guard
6656587493	Small Forward
6677994430	Point Guard
6766769938	Point Guard
9008343929	Point Guard
9080984444	Power Forward
9573002290	Point Guard
9823532343	Shooting Guard
9823532343	Small Forward
NULL	NULL

Description of VIEW, FUNCTION and PROCEDURE

-- VIEW

-- This view is used to look at the top earning player in each team as long as they have a nickname.

```
CREATE OR REPLACE VIEW TopEarningPlayers
AS (
SELECT team_id, first_name, last_name, salary
FROM Player
WHERE (team_id, salary)
IN (
        SELECT team_id, MAX(Salary)
        FROM Player
        GROUP BY team_id
        HAVING nickname is not null
        )
);
```

SELECT * FROM TopEarningPlayers;

team_id	first_name	last_name	salary
1029876678	Lebron	James	44470000
3409573490	Giannis	Antetokumpo	39340000
3459871029	Chris	Paul	38510000
0123456789	Jayson	Tatum	32600000
1234987655	Kevin	Durant	42970000
3456781029	Stephen	Curry	45780000
2001996030	John	Wall	41250000

-- FUNCTION

-- This function is used to determine the type of ticket based on the price of the ticket.

```
DROP FUNCTION IF EXISTS fn_typeofticket;
DELIMITER //

CREATE FUNCTION fn_typeofticket
(
        ticketprice VARCHAR(6)
)
```

```
RETURNS VARCHAR(25)
BEGIN
        -- declare variables
   DECLARE typeofticket VARCHAR(25);
   -- do things here
   IF ticketprice >= 14000 AND ticketprice <= 20000
                THEN SET typeofticket = 'Court Side Platinum';

        ELSEIF ticketprice < 14000 AND ticketprice >= 10000
                THEN SET typeofticket = 'Front Row Gold';

        ELSEIF ticketprice < 10000 AND ticketprice >= 5000
                THEN SET typeofticket = 'Back Row Premium';

        ELSEIF ticketprice < 5000 AND ticketprice >= 0
                THEN SET typeofticket = 'Normal Seating';
        END IF;
   -- return
   RETURN typeofticket;

END; //

DELIMITER ;
```

SELECT fn_typeofticket(10000);

	fn_typeofticket(10000)
▶	Front Row Gold

<u>-- PROCEDURE</u>

– This procedure is used to neatly organize information for tickets that were bought by fans. Uses fn_typeofticket within the procedure.

```
DROP PROCEDURE IF EXISTS sp_fans;
DELIMITER //

CREATE PROCEDURE sp_fans()
BEGIN
```

```
    SELECT
        CONCAT(Fan.first_name, ' ', Fan.last_name) AS name, Fan.stadium_name,
        Ticket.seat_number, fn_typeofticket(Ticket.price) AS type, Ticket.price
    FROM Fan
    INNER JOIN Ticket ON Ticket.id = Fan.ticket_id;

END; //

DELIMITER ;
```

CALL sp_fans();

name	stadium_name	seat_number	type	price
Dave Chappele	Barclays Center	400	Normal Seating	999
Dwayne Johnson	Barclays Center	20	Front Row Gold	10000
Kanye West	Barclays Center	333	Normal Seating	1200
Homero Arellano	Chase Center	1	Court Side Platinum	15000
Kendal Jenner	Chase Center	12	Court Side Platinum	15000
Marshal Mathers	Chase Center	234	Normal Seating	2000
Billie Eilish	Crypto.com Arena	792	Normal Seating	300
Cardi B	Crypto.com Arena	356	Normal Seating	500
Kendrick Lamar	Crypto.com Arena	123	Normal Seating	3908
Lady Gaga	Crypto.com Arena	829	Normal Seating	400
Miley Cyrus	Crypto.com Arena	198	Normal Seating	2000
Travis Scott	Crypto.com Arena	987	Normal Seating	300
Bruno Mars	Fiserv Forum	888	Normal Seating	400
Chief Keef	Fiserv Forum	111	Back Row Premium	5000
Emma Stone	Fiserv Forum	478	Normal Seating	1000
DeAndre Cortez	Footprint Center	578	Normal Seating	3000
Jack Harlow	Footprint Center	321	Normal Seating	700
Tory Lanes	Footprint Center	432	Normal Seating	900
Ayesha Curry	Madison Square ...	222	Back Row Premium	9000
Chloe Kardashian	Madison Square ...	45	Front Row Gold	12000
Joe Rogan	Madison Square ...	190	Normal Seating	1200
Barthalamule Co...	TD Garden	390	Normal Seating	450
Eddie Murphy	TD Garden	233	Normal Seating	900
Micheal Jackson	TD Garden	19	Front Row Gold	10000

Queries

-- Query #1

– Set Operation (Union, Intersect, OR Except)

-- Produces a scalar result of all tickets never sold (regardless of the team).. All teams unsold tickets in NBA

```
SELECT COUNT(*) AS 'Tickets Never Sold' FROM(
(SELECT id FROM Ticket)
EXCEPT
(SELECT ticket_id FROM Fan)
) AS T;
```

	Tickets Never Sold
▶	3

-- Query #2

– Query joining 3 or more tables

-- Lists out the most important information about a player by joining 3 tables.

```
SELECT
        CONCAT(P.first_name, ' ', P.last_name) AS name,
        T.name as team ,
        P.stadium_name AS stadium,
        CONCAT(S.city, ', ', S.state) AS location,
        COUNT(A.position) AS '# of positions'
FROM Player AS P
JOIN Team AS T
        ON T.id = P.team_id
JOIN Stadium AS S
        ON S.name = P.stadium_name
JOIN Player_Position AS A
         ON A.player_id = P.id GROUP BY A.player_id;
```

name	team	stadium	location	# of positions
Lebron James	Los Angeles Lakers	Crypto.com Arena	Los Angeles, CA	2
Giannis Antetokumpo	Milwuakee Bucks	Fiserv Forum	Milwuakee, WI	3
Devin Booker	Pheonix Suns	Footprint Center	Pheonix, AZ	2
Paul George	LA Clippers	Crypto.com Arena	Los Angeles, CA	2
Kyrie Irving	Brooklyn Nets	Barclays Center	Brooklyn, NY	2
Khris Middleton	Milwuakee Bucks	Fiserv Forum	Milwuakee, WI	3
Patrick Beverly	Los Angeles Lakers	Crypto.com Arena	Los Angeles, CA	2
Draymond Green	Golden State Warriors	Chase Center	San Francisco, CA	2
Klay Thompson	Golden State Warriors	Chase Center	San Francisco, CA	2
Chris Paul	Pheonix Suns	Footprint Center	Pheonix, AZ	1
Joe Ingles	Milwuakee Bucks	Fiserv Forum	Milwuakee, WI	4
RJ Barret	New York Knicks	Madison Square ...	New York City, NY	1
Jae Crowder	Pheonix Suns	Center	Pheonix, AZ	2
Kevin Durant	Brooklyn Nets	Barclays Center	Brooklyn, NY	2
Ben Simmons	Brooklyn Nets	Barclays Center	Brooklyn, NY	2
Stephen Curry	Golden State Warriors	Chase Center	San Francisco, CA	1
John Wall	LA Clippers	Crypto.com Arena	Los Angeles, CA	1
Derick Rose	New York Knicks	Madison Square ...	New York City, NY	1
Julius Randle	New York Knicks	Madison Square ...	New York City, NY	1
Russel Westbrook	Los Angeles Lakers	Crypto.com Arena	Los Angeles, CA	1
Kawahi Leonard	LA Clippers	Crypto.com Arena	Los Angeles, CA	2

Pheonix Suns

-- Query #3

– UPDATE

-- Update Ticket price of the warriors team since they just won a championship

```
UPDATE Ticket
SET price = price + 1000
WHERE team_id IN
        (SELECT id
         FROM Team
        WHERE name = 'Golden State Warriors');
```

BEFORE UPDATE:
SELECT * FROM Ticket;

id	team_id	price	seat_number
0000003979	3409573490	5000	111
0024323900	0986123458	12000	45
1111132322	3409573490	400	888
2342320938	3459871029	900	432
2343422322	2001996030	2000	198
2798384080	0123456789	10000	19
2909479073	1029876678	501	340
3098340939	1234987655	11000	14
3200000243	3459871029	700	321
3255423456	0123456789	450	390
3332429528	3456781029	2000	234
3425464223	1234987655	999	400
4059280979	0123456789	900	233
4448394839	1029876678	500	356
4538637978	3409573490	1000	478
5093450982	3456781029	1000	800
5433523434	0986123458	9000	222
5499845798	1234987655	1200	333
5555545222	2001996030	400	829
5666669829	3459871029	3000	578
5746564532	1234987655	10000	20
8834245244	3456781029	15000	1
8844887832	1029876678	3908	123
8984397290	1029876678	300	987
9898745939	0986123458	1200	190
9923423549	3456781029	15000	12
9999932222	2001996030	300	792
NULL	NULL	NULL	NULL

AFTER UPDATE:
SELECT * FROM Ticket;

id	team_id	price	seat_number
0000003979	3409573490	5000	111
0024323900	0986123458	12000	45
1111132322	3409573490	400	888
2342320938	3459871029	900	432
2343422322	2001996030	2000	198
2798384080	0123456789	10000	19
2909479073	1029876678	501	340
3098340939	1234987655	11000	14
3200000243	3459871029	700	321
3255423456	0123456789	450	390
3332429528	3456781029	3000	234
3425464223	1234987655	999	400
4059280979	0123456789	900	233
4448394839	1029876678	500	356
4538637978	3409573490	1000	478
5093450982	3456781029	2000	800
5433523434	0986123458	9000	222
5499845798	1234987655	1200	333
5555545222	2001996030	400	829
5666669829	3459871029	3000	578
5746564532	1234987655	10000	20
8834245244	3456781029	16000	1
8844887832	1029876678	3908	123
8984397290	1029876678	300	987
9898745939	0986123458	1200	190
9923423549	3456781029	16000	12
9999932222	2001996030	300	792
NULL	NULL	NULL	NULL

-- Query #4
– Query that uses Group By and Having
-- Lists all coaches who coach more than one player

```
SELECT PC.coach_id, CONCAT (C.first_name, ' ', C.last_name) AS name,
            COUNT(PC.player_id) AS coaching
FROM Player_Coach AS PC
JOIN Coach AS C ON C.id = PC.coach_id
GROUP BY PC.coach_id
HAVING COUNT(PC.player_id) > 1;
```

coach_id	name	coaching
1124954879	Monty Williams	2
3425432434	Darvin Ham	2
4537592209	Brett Siegel	2
6598937297	Bruce Fraser	3
6875949294	Damon Stoudamire	2
7486379202	Steve Kerr	3
7653090280	Chris Jent	2
9089749888	Mike Budenholzer	2
9385658783	Tyronn Lue	3
9978765954	Steve Nash	2

-- Query #5
– Subquery using WHERE IN or WHERE NOT IN
-- Lists all teams that sold out their stadium

```
SELECT *, 'SOLD OUT' AS 'sold out?'
FROM Team AS M
WHERE M.id NOT IN (SELECT T.team_id
                   FROM Ticket AS T
                   WHERE T.id NOT IN (
                                      SELECT ticket_id
                                      FROM Fan
                                       )
                   );
```

	id	name	logo	color	sold out?
▶	0123456789	Boston Celtics	Leprechaun	Green	SOLD OUT
	0986123458	New York Knicks	Basketball	Orange	SOLD OUT
	2001996030	LA Clippers	Basketball	Blue	SOLD OUT
	3409573490	Milwuakee Bucks	Buck	Green	SOLD OUT
	3459871029	Pheonix Suns	Basketball	Purple	SOLD OUT