DATA70141 Assignment 1 - Monopolee

SECTION 1: Entity-Relationship Diagram

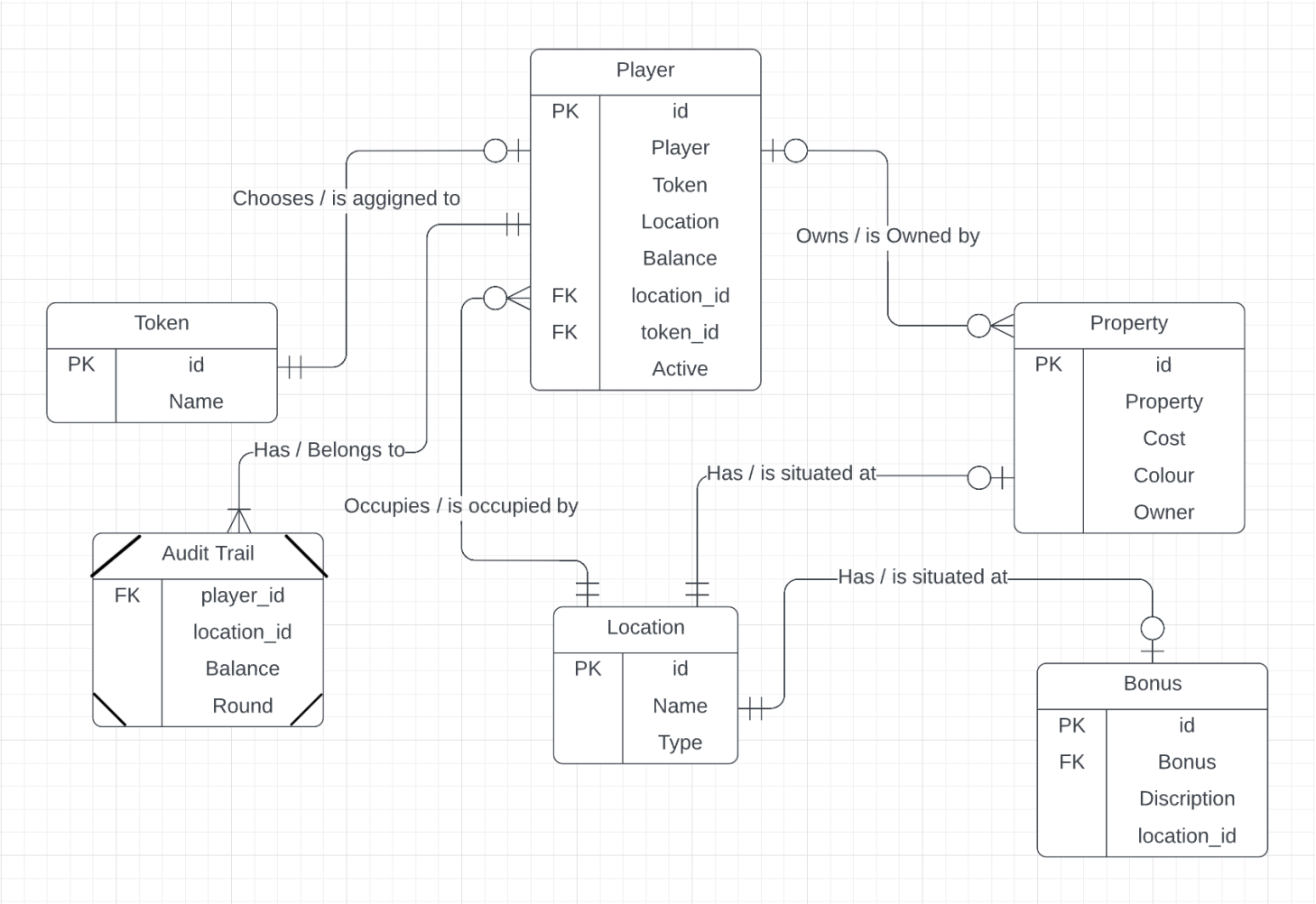


Figure 1: Entity-Relationship Diagram for Monopolee

Name - Samit Shrinivas Uttarkar
Student ID - 11186818

SECTION 2: Entity-Relationship Diagram Description

The Entity Relationship diagram above shows the relationship between different entities in the Monopolee Schema. The ER Diagram contains a total of 6 entities which are:

- **Location Entity** - It stores all the locations id's and Location Names of all the locations in the board
- **Token Entity** - It stores the token names and their id's of which players can choose from
- **Property Entity** - Stores property details and their Owners
- **Bonus Entity** - Stores information about the bonuses on the the board
- **Player Entity** - Stores all the information about the Players playing the game
- **Audit Trail Entity** - Stores the audit trail of Players as game moves forward

In the ER diagram for Monopolee there are a total of 6 Relations which entities share among each other. Some entities have relation with more than one table while some have only one. The Relations between the entities are:

- **Location-Property Relation** - Location and property have One-to-One relations due to the fact that a location can be a property whereas a property has to be a location or at a location. Thus stating that a Location has a property and a property is situated at a location
- **Location-Bonus Relation** - Similar to Location-Property relation, Location and Bonus also have a One to oOne relations because a Property can either be a bonus or not but a bonus is always a location. Bonuses act similar to properties where they are situated at a location and a location has a bonus.
- **Player-Location Relation** - Player and Location have One-to-Many relation since a Player at a given time can only occupy one location but a location can be occupied of zero or many players at a time.
- **Player-Property Relation** - They have Many-to-One relations because a player can own multiple properties at the same time whereas a property can only be owned by a single player.
- **Player-Token Relation** - A player chooses a Token therefore a player is always assigned to one token when the game starts, on the other hand a token can be assigned to zero players or at max one player. Thus Player and Token have One to One relation
- **Player-Audit Trail Relation** - The Player entity and the Audit Trail entity have Many-to-One relation cause a Player has a different location every time a dice is rolled thus consisting have multiple records in the Audit Trail whereas an Audit Trail can only have one player at a time

In the ER diagram the player entity acts as the main entity as it stores all the records and is connected with all the other entities except the bonus entity. The Audit Trail Entity is a weak entity as it does not consist of any Primary key and is fully dependent on the Player entity. This shows that if a player doesn't exist their audit trail also doesn't exist as it fetches most of the records from the Player entity. It was a tough choice whether to add to token entity or not as the only role of token entity is for the tokens to be assigned to the players which can be also done in the player table itself. The reason Token entity was included was because, if more players join the game it would be easy to get their token from the token entity rather than adding manually as the token entity stores Name's of all the tokens. The bonus entity does not have any relation with the player entity due to the fact that bonus is linked with location and location is furthermore linked with player so it's easy to know whether a player has a bonus or not by just knowing their location.

The design is chosen for keeping the diagram simple, readable and to understand information clearly. Keeping the number of entities to a minimum and only including those which are important for the game led to the design of the ER diagram.

Name - Samit Shrinivas Uttarkar
Student ID - 11186818

SECTION 3: Relational Database Schema

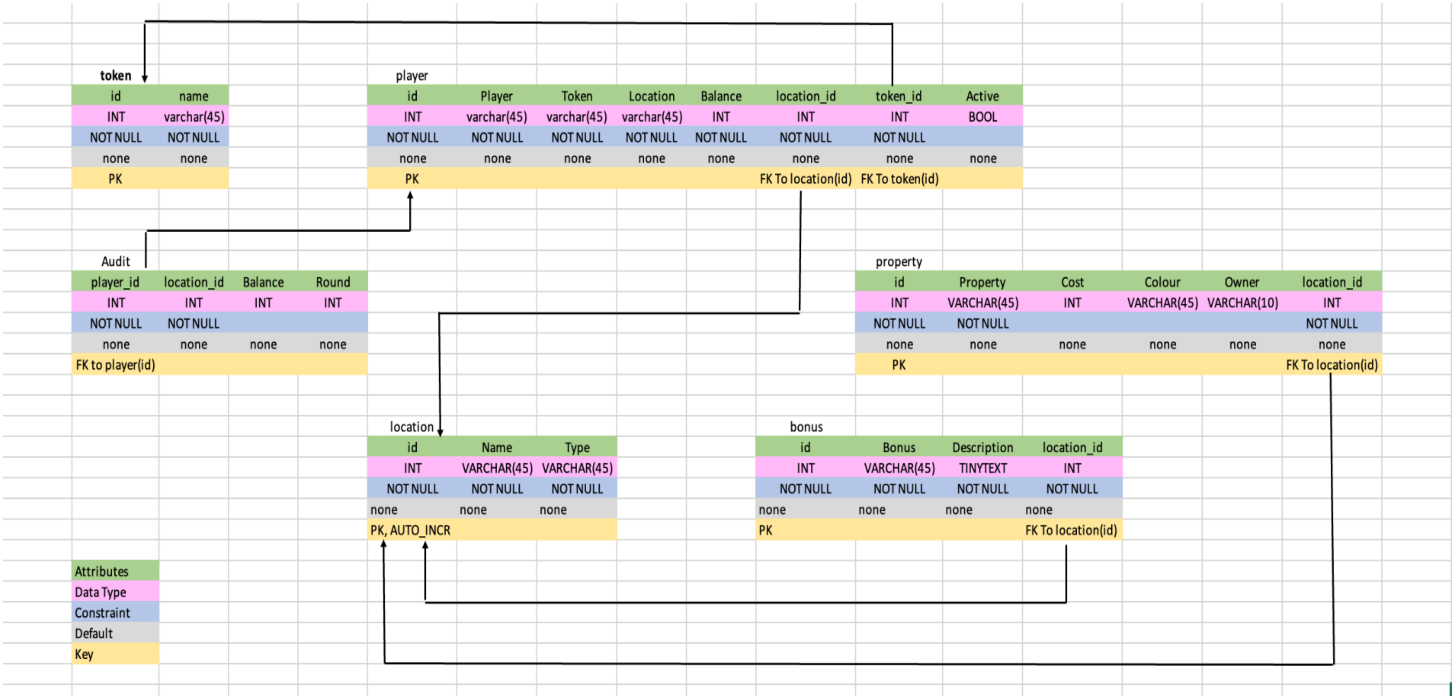
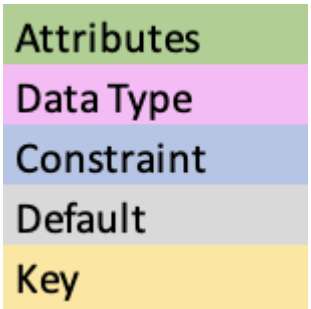


Figure 2: Relational Database Schema diagram

The following color code was used for able to easily distinguish between rows and better readability



The Relational Database Schema for Monopolee consists of 6 tables each with different attributes. They are comprised of follow tables:

1. The **Location** entity in our ER diagram is mapped to the **location** table which consists of 3 attributes:
    - The 'id' which is the primary key of the table
    - The 'Name' which stores the names of all the locations and is assigned to a location id

- 'Type' attribute stores what type of location it is. A location can be of: a Corner, a Chance, a Community Chest or a Property Type

location		
id	Name	Type
INT	VARCHAR(45)	VARCHAR(45)
NOT NULL	NOT NULL	NOT NULL
none	none	none
PK, AUTO_INCR		

Figure 3: location table

2. The **Token** entity in the ER diagram is mapped to the **token** table which consists of 2 attributes:
   - The 'id' is the primary key for the tables as each token is unique
   - 'Name' stores all the names of the token

token	
id	name
INT	varchar(45)
NOT NULL	NOT NULL
none	none
PK	

Figure 4: token table

3. The **Property** entity in the ER diagram is mapped to the **property** table which consists of 6 attributes:
   - The 'id' which is the primary key used to assign to all the properties
   - The 'Property' which consists of names of all the properties
   - The 'Cost' which stores the costs of all the properties
   - The 'Colour' which stores the color of each location
   - The 'Owner' which stores the owner of the property
   - The location_id which is the foreign key reference to the location tables id. It is used to let us know which locations are properties

property					
id	Property	Cost	Colour	Owner	location_id
INT	VARCHAR(45)	INT	VARCHAR(45)	VARCHAR(10)	INT
NOT NULL	NOT NULL				NOT NULL
none	none	none	none	none	none
PK					FK To location(id)

Figure 5: property table

4. The **Bonus** entity in the ER diagram is mapped to the **bonus** table which comprises of 4 attributes

- The ‘id’ which is the primary key used to assign to all the bonuses
- The ‘Bonus’ which stores the names of the bonuses
- The ‘Description’ which stores the description of the bonus
- The location_id which is the foreign key reference to the location tables id. It is used to let us know which locations are bonuses

bonus			
id	Bonus	Description	location_id
INT	VARCHAR(45)	TINYTEXT	INT
NOT NULL	NOT NULL	NOT NULL	NOT NULL
none	none	none	none
PK			FK To location(id)

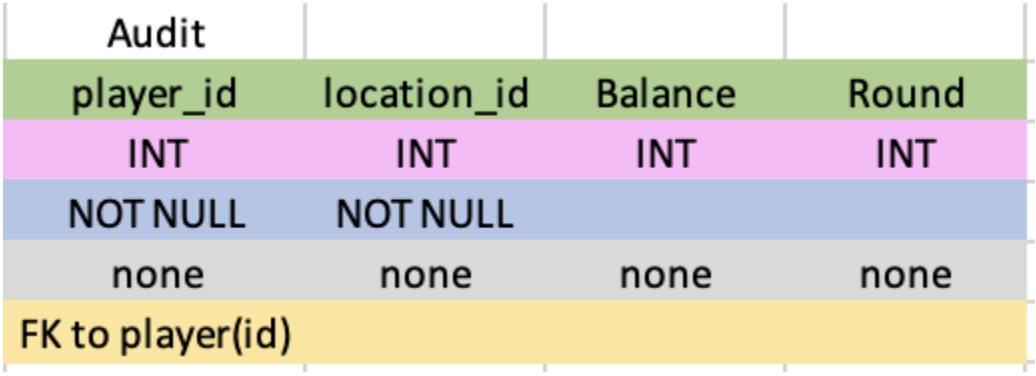
Figure 6: bonus table

5. The **Player** entity in the ER diagram is mapped to the **player** table which comprises of 8 Attributes:
    - The ‘id’ which is the primary key used to assign to all the player
    - The ‘Player’ which stores the names of the players playing the game
    - The ‘Token’ stores the token name assigned to the player
    - The ‘Location’ stores the Location name of the location on which player is on
    - The ‘Balance’ stores the balance of the players
    - The ‘token_id’ which is the foreign key reference to Token tables id.
    - The ‘location_id’ which is the foreign key reference to Locationtables id. From this attribute we can fetch values from the property table as well as the bonus table and get to know which player owns a property and which player is on a bonus.
    - The ‘Active’ attribute is of a boolean type where it shows whether the player is active i.e. playing or not

player							
id	Player	Token	Location	Balance	location_id	token_id	Active
INT	varchar(45)	varchar(45)	varchar(45)	INT	INT	INT	BOOL
NOT NULL	NOT NULL	NOT NULL	NOT NULL	NOT NULL	NOT NULL	NOT NULL	
none	none	none	none	none	none	none	none
PK					FK To location(id)	FK To token(id)	

Figure 7: player table

6. The **Audit Trail** entity in the ER is mapped to the audit **table** which consists of 4 attributes:
    - The ‘player_id’ which is the foreign key references to the player table id. It is used to fetch all the useful records from the players such as player name, player balance etc.
    - The ‘location_id’ which tells us where to player are on the board
    - The ‘Balance’ which stores the balance of the player at a particular round
    - The ‘Round’ which stores the round played



Audit			
player_id	location_id	Balance	Round
INT	INT	INT	INT
NOT NULL	NOT NULL		
none	none	none	none
FK to player(id)			

Figure 8: audit table

The following queries are for creating the schema `Monopolee` and table

```
CREATE SCHEMA `Monopolee` ;
```


```
CREATE TABLE Monopolee.location (
  id INT NOT NULL AUTO_INCREMENT,
  Name VARCHAR(45) NOT NULL,
  Type VARCHAR(45) NULL,
  PRIMARY KEY (id));

CREATE TABLE Monopolee.token(
  id INT NOT NULL,
  Name Varchar(45),
  PRIMARY KEY(id));

CREATE TABLE Monopolee.bonus(
  id INT NOT NULL,
  Bonus VARCHAR(45) NOT NULL,
  Description TINYTEXT ,
  location_id INT NOT NULL,
  PRIMARY KEY(id),
  FOREIGN KEY (location_id) REFERENCES location(id));

CREATE TABLE Monopolee.property(
  id INT NOT NULL,
  Property VARCHAR(45) NOT NULL,
  Cost INT,
  Colour varchar(45),
  Owner varchar(10),
  location_id INT NOT NULL,
  PRIMARY KEY(id),
```

```
  FOREIGN KEY (location_id) REFERENCES location(id));

CREATE TABLE Monopolee.player(
  id INT NOT NULL AUTO_INCREMENT ,
  Player VARCHAR(45) NOT NULL,
  Token VARCHAR(45) NOT NULL,
  Location VARCHAR(45) NOT NULL,
  Balance INT NOT NULL,
  location_id INT NOT NULL,
  token_id INT NOT NULL,
  Active Bool,
  PRIMARY KEY(id),
  FOREIGN KEY (location_id) REFERENCES location(id),
  FOREIGN KEY (token_id) REFERENCES token(id));


CREATE TABLE Monopolee.audit(
  player_id INT NOT NULL,
  location_id INT NOT NULL,
  Balance INT,
  Round INT,
  FOREIGN KEY (player_id) REFERENCES player(id));
```

```
599  00:09:19  CREATE TABLE Monopolee.location (  id INT NOT NULL...  0 row(s) affected  0.067 sec
600  00:09:19  CREATE TABLE Monopolee.token( id INT NOT NULL, Na...  0 row(s) affected  0.0048 sec
601  00:09:19  CREATE TABLE Monopolee.bonus(  id INT NOT NULL,...  0 row(s) affected  0.0079 sec
602  00:09:19  CREATE TABLE Monopolee.property(  id INT NOT NULL,...  0 row(s) affected  0.0077 sec
603  00:09:19  CREATE TABLE Monopolee.player(  id INT NOT NULL A...  0 row(s) affected  0.0078 sec
604  00:09:19  CREATE TABLE Monopolee.audit( player_id INT NOT NU...  0 row(s) affected  0.0062 sec
```

The following queries are for populating the tables in our schema

```
INSERT INTO location (Name,Type)
VALUES ('GO','Corner'),
('Kilburn','Property'),
('CHANCE 1','Chance'),
('Uni Place','Property'),
('IN JAIL','Corner'),
('Victoria','Property'),
('COMMUNITY CHEST 1','Community chest'),
('Piccadilly','Property'),
('FREE PARKING','Corner'),
('Oak House','Property'),
('CHANCE 2','Chance'),
('Owens Park','Property'),
('GO TO JAIL','Corner'),
('AMBS','Property'),
('COMMUNITY CHEST 2','Community chest'),
('Co-op','Property');
```

```
620  00:20:16  INSERT INTO location (Name,Type) VALUES ('GO','Corne...  16 row(s) affected Records: 16  Duplicates: 0  Warnin...  0.016 sec
```

Name - Samit Shrinivas Uttarkar
Student ID - 11186818

```
INSERT INTO property
VALUES (1,"Oak house", 100, "Orange","Norman",10),
(2,"Owens Park",30,"Orange","Norman",12),
(3,"AMBS",400,"Blue",DEFAULT,14),
(4,"Co-Op",30,"Blue","Jane",16),
(5,"Kilburn",120,"Yellow",DEFAULT,2),
(6,"Uni Place",100,"Yellow",DEFAULT,4),
(8,"Victoria",75,"Green","Bill",6),
(9,"Piccadilly",35,"Green",DEFAULT,8);
```

```
622  00:23:23  INSERT INTO property VALUES (1,"Oak house", 100, "Or...  8 row(s) affected Records: 8  Duplicates: 0  Warnings...  0.079 sec
```

```
INSERT INTO bonus
VALUES (1,'Chance 1', 'Pay each of the other players £50',2),
(2,'Chance 2','Move forward 3 spaces',11),
(3,'Community Chest 1','For winning a Beauty Contest, you win £100',7),
(4,'Community Chest 2','Your library books are overdue. Play a fine of £30',15),
(5,'Free Parking','No action',9),
(6,'Go to Jail','Go to Jail, do not pass GO, do not collect £200',13),
(7,'GO','Collect £200',1);
```

```
625  00:26:01  INSERT INTO bonus VALUES (1,'Chance 1', 'Pay each of t...  7 row(s) affected Records: 7  Duplicates: 0  Warnings:...  0.030 sec
```

```
INSERT INTO token
values
(1,'Dog'),
(2,'Car'),
(3,'Battleship'),
(4,'Top Hat'),
(5,'Thimble'),
(6,'Boot')
```

```
631  00:34:06  INSERT INTO token values (1,'Dog'), (2,'Car'), (3,'Battles...  6 row(s) affected Records: 6  Duplicates: 0  Warnings...  0.0074 sec
```

```
INSERT INTO player(Player,Token,Location,Balance,location_id,token_id,Active)
VALUES ('Mary', 'Battleship','FREE PARKING',190,9,3,False),
('Bill','Dog','Owens Park',500,12,1,False),
('Jane','Car','AMBS',150,14,2,False),
('Norman','Thimble','Kilburn', 250,2,5,False)
```

```
634  00:38:27  INSERT INTO player(Player,Token,Location,Balance,locat...  4 row(s) affected Records: 4  Duplicates: 0  Warnings...  0.016 sec
```

```
INSERT INTO audit
VALUES (1, 9,190,0),
(2,12,500,0),
(3,14,150,0),
(4,2,250,0);
```

```
637  00:41:44  INSERT INTO audit VALUES (1, 9,190,0), (2,12,500,0), (3,...  4 row(s) affected Records: 4  Duplicates: 0  Warnings...  0.0051 sec
```

Name - Samit Shrinivas Uttarkar
Student ID - 11186818

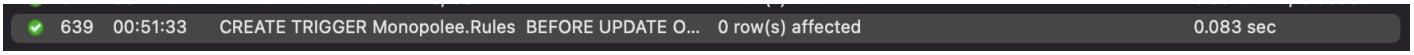
SECTION 4: Queries and outputs for gameplay

4.1. Creating Trigger

The Trigger `Rules` is applied to Monopolee using the following code. This trigger basically updates the player's location name when landed on the location and also updates the balance of the player when the player lands on a location which is a Bonus automatically. The trigger also set's location to 'IN JAIL' when the player lands on 'GO TO JAIL'.

```
DELIMITER /
CREATE TRIGGER Monopolee.Rules
BEFORE UPDATE ON player
FOR EACH ROW
BEGIN
DECLARE location_id INT;
IF NEW.location_id =13 THEN
SET NEW.location_id = 5;
END IF;
IF NEW.location_id = 5 THEN
SET New.Location = 'IN JAIL';
END IF;
IF New.location_id = 3 AND New.Active = True THEN
SET New.Balance = New.Balance - 150, New.Location = 'CHANCE 1';
END IF;
IF New.location_id = 7 AND New.Active = True THEN
SET New.Balance = New.Balance + 100, New.Location = 'COMMUNITY CHEST 1';
END IF;
IF New.location_id = 11 AND New.Active = True THEN
SET New.location_id = New.location_id + 3, New.Location = 'AMBS';
END IF;
IF New.location_id = 15 AND New.Active = True THEN
SET New.Balance = New.Balance - 30,New.Location = 'COMMUNITY CHEST 2';
END IF;
IF NEW.location_id = 1 THEN
set New.Location = 'GO';
END IF;
IF NEW.location_id = 2 THEN
SET New.Location = 'Kilburn';
END IF;
IF NEW.location_id = 4 THEN
SET New.Location = 'Uni Place';
END IF;
IF NEW.location_id = 6 THEN
SET New.Location = 'Victoria';
END IF;
IF NEW.location_id = 8 THEN
SET New.Location = 'Piccadilly';
END IF;
IF NEW.location_id = 9 THEN
SET New.Location = 'FREE PARKING';
END IF;
```

```
IF NEW.location_id = 10 THEN
SET New.Location = 'Oak House';
END IF;
IF NEW.location_id = 12 THEN
SET New.Location = 'Owens Park';
END IF;
IF NEW.location_id = 14 THEN
SET New.Location = 'AMBS';
END IF;
IF NEW.location_id = 16 THEN
SET New.Location = 'Co-op';
END IF;
END /
DELIMITER ;
```



4.2 Creating Stored Procedures

4.2.1 Creating Procedure for updating player and property table

In the following code we have created a stored procedure named Update_players which as the name states updates the location of the player with its inputs as the Name of the player playing and the Dice rolled when called.

This stored procedure contains three update statements. First we update the player table and set it to Active as False indicating all the players are inactive before the start.

Second it updates the player table again where it will add 200 to the player balance if the location id and the dice number adds up to more than 16. This basically means the player passes from 'GO', next it will set

Lastly it updates the property table where if a player lands on the property and if there is no owner, the player will buy the property, setting the player as the owner and the property cost will be subtracted from the players balance when called.

```
DELIMITER //
CREATE PROCEDURE Update_player(IN Name varchar(45),IN dice INT)
BEGIN
    UPDATE player
      set Active = False;
    UPDATE player
    SET Balance = IF(location_id+dice>16,balance + 200,Balance), player.Active = True,
      location_id =
IF(IF(MOD((location_id+dice),16)=0,16,MOD((location_id+dice),16))=13,5,IF(MOD((location_id+dic
e),16)=0,16,MOD((location_id+dice),16)))
      WHERE Player = Name;

    UPDATE property
    join location on location_id = property.location_id
    join player on player.location_id = location.id
    SET property.Owner = Name, player.Balance = player.Balance - property.Cost
    where property.location_id = player.location_id and property.Owner is Null;
```

```
END //
DELIMITER ;
```

4.2.2 Creating Procedure to update Audit Table

The following code creates a stored procedure called Update_balance where it will take the Name of the player and amount of money to be added to the balance. Further will it add the amount to be added to the Balance of the player.

```
DELIMITER //
CREATE PROCEDURE Update_balance(IN Name varchar(45), IN Add_Balance INT)
BEGIN
      UPDATE player
    set Balance = Balance + Add_Balance
      WHERE Player = Name;
END //

DELIMITER ;
```

4.2.3 Creating Procedure to update audit

Here we are creating an update_audit procedure which takes Round as an input. This procedure will fetch id, location id and Balance from the payer table and set Round as the input where Round is Null.

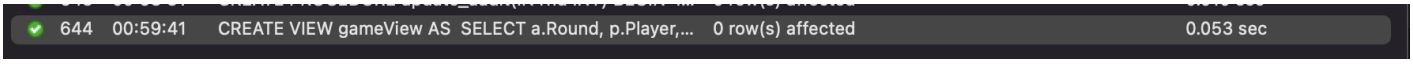
```
DELIMITER //
CREATE PROCEDURE update_audit(IN rnd INT)
BEGIN
      INSERT INTO audit(`player_id`,`location_id`,`Balance`)
      SELECT id,location_id,Balance FROM player;

    UPDATE audit
    set Round = rnd
    where Round is Null;
END //

DELIMITER ;
```

4.3 Creating gameView

4.3.1 creating a view, “gameView”

creating a view, “gameView”, of the database which will be displayed on a giant screen and will give the current status of each player, the round number, the player’s name, their current balance and their current board location .

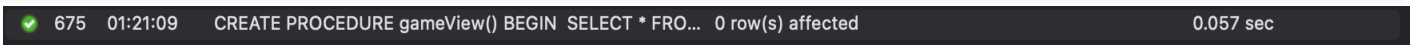
```
CREATE VIEW gameView AS
 SELECT a.Round, p.Player, p.Token, p.Balance, p.Location
 FROM audit as a
 JOIN player p on p.id = a.player_id
 WHERE Round = (SELECT MAX(Round) from Audit);
```



4.3.2 Creating Stored Procedure for gameView

The following code creates a procedure for the View “gameView” and returns everything from gameView when called

```
DELIMITER //
CREATE PROCEDURE gameView()
BEGIN
	SELECT * FROM gameView;
END //

DELIMITER ;
```



gameView before the game starts

```
CALL gameView
```

	Round	Player	Token	Balance	Location
▶	0	Mary	Battleship	190	FREE PARKING
	0	Bill	Dog	500	Owens Park
	0	Jane	Car	150	AMBS
	0	Norman	Thimble	250	Kilburn

Figure 9 : gameView before the game starts

4.4 Round 1

4.4.1 Updating Players

The game run's as follows:

1. First Jane rolle's a 3 for which we will call Update_player procedure to update the table.
2. Norman rolle's a 1 which makes him land on 'CHANCE 1' for which 150 will be automatically deducted from his balance but we still have to add 50 to other players. Here we call Update_balance to add 50 to other 3 players
3. Next Mary rolle's a 4
4. Bill rolle's a 5

```
call Update_player('Jane',3)

call Update_player('Norman',1)
call Update_balance('Jane',50)
call Update_balance('Bill',50)
call Update_balance('Mary',50)

call Update_player('Mary',4)

call Update_player('Bill',2)
```

646	01:04:57	call Update_player('Jane',3)	2 row(s) affected	0.045 sec
647	01:05:00	call Update_player('Norman',1)	0 row(s) affected	0.0044 sec
648	01:05:02	call Update_balance('Jane',50)	1 row(s) affected	0.0032 sec
649	01:05:05	call Update_balance('Bill',50)	1 row(s) affected	0.0021 sec
650	01:05:09	call Update_balance('Mary',50)	1 row(s) affected	0.0019 sec
651	01:05:12	call Update_player('Mary',4)	0 row(s) affected	0.0036 sec
652	01:05:15	call Update_player('Bill',2)	2 row(s) affected	0.0036 sec

```
SELECT * FROM player
```

	id	Player	Token	Location	Balance	location_id	token_id	Active
▶	1	Mary	Battleship	IN JAIL	240	5	3	0
	2	Bill	Dog	AMBS	150	14	1	1
	3	Jane	Car	GO	400	1	2	0
	4	Norman	Thimble	CHANCE 1	100	3	5	0

Figure 10: Updated player table after round 1

4.4.2 Updating Audit

Now we will update the Audit table

```
call update_audit(1)
```

Name - Samit Shrinivas Uttarkar
Student ID - 11186818

```
672  01:10:53  call update_audit(1)  4 row(s) affected  0.043 sec
```

```
SELECT * FROM audit
```

	player_id	location_id	Balance	Round
▶	1	9	190	0
	2	12	500	0
	3	14	150	0
	4	2	250	0
	1	5	240	1
	2	14	150	1
	3	1	400	1
	4	3	100	1

Figure 11: Updated Audit table after round 1

4.4.3 gameView for Round 1

```
CALL gameView
```

	Round	Player	Token	Balance	Location
▶	1	Mary	Battleship	240	IN JAIL
	1	Bill	Dog	150	AMBS
	1	Jane	Car	400	GO
	1	Norman	Thimble	100	CHANCE 1

Figure 12: gameView for Round 1

4.5 Round 2

4.5.1 Updating players

1. Jane rolle's a 6 which makes her get out of jail and then she rolle's 5 which lands her on a property, but the property is already owned by someone. So we will subtract property cost from jane and add it to the property owner by calling Update_balance
2. Norman rolle's a 4
3. Mary rolle's a 5 which makes her land on property, but this property is owned by Norman which owns two properties of the same color. Thus we need to subtract double the property cost from mary and add it to Norman
4. Bill rolle's a 6 and a 3

```
call Update_player('Jane',5)
call Update_balance('Jane',-75)
call Update_balance('Bill',75)
```

```
call Update_player('Norman',4)

call Update_player('Mary',5)
call Update_balance('Mary',-200)
call Update_balance('Norman',+200)

call Update_player('Bill',9)
```

684	01:25:10	call Update_player('Jane',5)	0 row(s) affected	0.038 sec
685	01:25:13	call Update_balance('Jane',-75)	1 row(s) affected	0.0044 sec
686	01:25:17	call Update_balance('Bill',75)	1 row(s) affected	0.0024 sec
687	01:25:20	call Update_player('Norman',4)	0 row(s) affected	0.0056 sec
688	01:25:24	call Update_player('Mary',5)	0 row(s) affected	0.0091 sec
689	01:25:38	call Update_balance('Mary',-200)	1 row(s) affected	0.0042 sec
690	01:25:41	call Update_balance('Norman',+200)	1 row(s) affected	0.0012 sec
691	01:25:44	call Update_player('Bill',9)	0 row(s) affected	0.0033 sec

```
SELECT * FROM player
```

id	Player	Token	Location	Balance	location_id	token_id	Active
1	Mary	Battleship	Oak House	40	10	3	0
2	Bill	Dog	COMMUNITY CHEST 1	525	7	1	1
3	Jane	Car	Victoria	325	6	2	0
4	Norman	Thimble	COMMUNITY CHEST 1	400	7	5	0

Figure 13: Player table after Round 2

4.5.2 Updating Audit

Now we will update the Audit table

```
call update_audit(2)
```

693	01:34:09	call update_audit(2)	4 row(s) affected	0.058 sec

```
SELECT * FROM audit
```

	player_id	location_id	Balance	Round
▶	1	9	190	0
	2	12	500	0
	3	14	150	0
	4	2	250	0
	1	5	240	1
	2	14	150	1
	3	1	400	1
	4	3	100	1
	1	10	40	2
	2	7	525	2
	3	6	325	2
	4	7	400	2

Figure 14: Audit Table after Round 2

4.5.3 gameView for Round 2

```
call gameView
```

	Round	Player	Token	Balance	Location
▶	2	Mary	Battleship	40	Oak House
	2	Bill	Dog	525	COMMUNITY CHEST 1
	2	Jane	Car	325	Victoria
	2	Norman	Thimble	400	COMMUNITY CHEST 1

Figure 15: gameView after Round 2