SQL Code for Monopolee Gameplay – Setup and Sample Game

Code for creating tables, inserting values into them and adding constraints:

```
CREATE PROCEDURE insertTables () BEGIN
-- Creating and inserting values into the board table CREATE TABLE board (
Loc_id int(11) NOT NULL, IsProperty tinyint(1) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
INSERT INTO board (Loc_id, IsProperty) VALUES (0, 0),(1, 1),(2, 0),(3, 1),(4, 0),(5, 1),(6,
0),(7, 1),(8, 0),(9, 1),(10, 0),(11, ),(12, 0),(13, 1),(14, 0),(15, 1);

-- Creating and inserting values into the bonus table
CREATE TABLE bonus (
Name varchar(30) NOT NULL, Id int(11) NOT NULL,
Description varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
INSERT INTO bonus (Name, Id, Description) VALUES
('CHANCE 1', 2, 'Pay each of the other players £50'),
('CHANCE 2', 10, 'Move forward 3 spaces'),
('COMMUNITY CHEST 1', 6, 'For winning a beauty contest, you win £100'),
('COMMUNITY CHEST 2', 14, 'Your library books are overdue. Pay a fine of £30'),
('FREE PARKING', 8, 'No action'),
('GO', 0, 'Collect £200'),
('GO TO JAIL', 12, 'Go to Jail, do not pass GO, do not collect £200'),
('VISITING JAIL', 4, 'Player visiting Jail');

-- Creating the log table
CREATE TABLE log (
LogID int(11) NOT NULL,
PlayerID int(11) NOT NULL, CurLoc int(11) NOT NULL, CurBank int(11) NOT NULL, Round int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- Creating and inserting values into the player table
CREATE TABLE player (
```

```
playerID int(11) NOT NULL, Name varchar(30) NOT NULL, Token int(11) NOT NULL, Amount int(11) NOT NULL,
Bonus varchar(30) DEFAULT NULL, Location int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
INSERT INTO player (playerID, Name, Token, Amount, Bonus, Location) VALUES
(1, 'Mary', 3, 190, 'FREE PARKING', 8),
(2, 'Bill', 1, 500, NULL, 11),
(3, 'Jane', 2, 150, NULL, 13),
(4, 'Norman', 5, 250, NULL, 1),
(5, 'Jack', 6, 320, 'CHANCE 1', 2);

-- Creating and inserting values into the property table
CREATE TABLE property (
Name varchar(30) NOT NULL, Id int(11) NOT NULL,
Owner int(11) DEFAULT NULL,
Value int(11) NOT NULL, Colour varchar(10) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
INSERT INTO property (Name, Id, Owner, Value, Colour) VALUES
 ('AMBS', 13, NULL, 400, 'Blue'),
('Co-Op', 15, 3, 30, 'Blue'),
('Kilburn', 1, NULL, 120, 'Yellow'),
('Oak House', 9, 4, 100, 'Orange'),
('Owens Park', 11, 4, 30, 'Orange'),
('Piccadilly', 7, NULL, 35, 'Green'),
('Uni Place', 3, 1, 100, 'Yellow'),
('Victoria', 5, 2, 75, 'Green');

-- Creating and inserting values into tokens table
CREATE TABLE tokens (
TokenID int(11) NOT NULL, Name varchar(30) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
INSERT INTO tokens (TokenID, Name) VALUES
(1, 'dog'),
(2, 'car'),
(3, 'battleship'),
(4, 'top hat'),
```

(5, 'thimble'),

(6, 'boot');

-- Adding Primary keys and foreign key constraints

ALTER TABLE board ADD PRIMARY KEY (Loc_id);

ALTER TABLE bonus ADD PRIMARY KEY (Name), ADD KEY location_key (Id);

ALTER TABLE log ADD PRIMARY KEY (LogID), ADD KEY player_key (PlayerID);

ALTER TABLE player ADD PRIMARY KEY (playerID), ADD KEY bonus_key (Bonus), ADD KEY token_key (Token), ADD KEY loc_key (Location);

ALTER TABLE property ADD PRIMARY KEY (Name), ADD KEY loc_id_key (Id), ADD KEY owner_key (Owner);

ALTER TABLE tokens ADD PRIMARY KEY (TokenID);

ALTER TABLE log MODIFY LogID int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;

ALTER TABLE bonus ADD CONSTRAINT location_key FOREIGN KEY (Id) REFERENCES board (Loc_id);

ALTER TABLE log ADD CONSTRAINT player_key FOREIGN KEY (PlayerID) REFERENCES player (playerID);

ALTER TABLE player ADD CONSTRAINT bonus_key FOREIGN KEY (Bonus) REFERENCES bonus (Name), ADD CONSTRAINT loc_key FOREIGN KEY (Location) REFERENCES board (Loc_id), ADD CONSTRAINT token_key FOREIGN KEY (Token) REFERENCES tokens (TokenID);

ALTER TABLE property ADD CONSTRAINT loc_id_key FOREIGN KEY (Id) REFERENCES board (Loc_id), ADD CONSTRAINT owner_key FOREIGN KEY (Owner) REFERENCES player (playerID); COMMIT;

END

Procedure for each Turn:

CREATE PROCEDURE takeTurn (IN playerName VARCHAR(30), IN roll1 INT(2), IN roll2 INT(2), IN round INT(2))

sp: BEGIN

-- Declaring the required variables

DECLARE curLoc INT DEFAULT 0;

DECLARE fine INT DEFAULT 0;

DECLARE newLoc INT DEFAULT 0;

DECLARE curBon VARCHAR(30);

DECLARE curOwn INT DEFAULT 0;

DECLARE colour VARCHAR(30);

DECLARE playerID INT DEFAULT 0;

DECLARE bankBal INT DEFAULT 0;

DECLARE propAmt INT DEFAULT 0;

DECLARE ownBank INT DEFAULT 0;

DECLARE players INT DEFAULT 0;

```
SET curLoc = (SELECT player.Location FROM player WHERE player.Name=playerName);

SET curBon = (SELECT player.Bonus FROM player WHERE player.Name=playerName);

SET playerID = (SELECT player.playerID FROM player WHERE player.Name=playerName);


-- Checking if the player is in jail

IF (curLoc=4 AND curBon="GO TO JAIL") THEN

        IF (roll1!=6) THEN

                LEAVE sp; -- Stop the turn if the first roll is not a 6 ELSE

                SET newLoc = ((curLoc + roll2) % 16);

        END IF;

ELSE – Computing the new location of the player

        SET newLoc = ((curLoc + roll1) % 16);

        IF (roll1=6) THEN

        SET newLoc = ((newLoc + roll2) % 16); END IF;

        END IF;

UPDATE player SET Location = newLoc WHERE player.Name = playerName;

UPDATE player SET Bonus = NULL WHERE player.Name = playerName;


-- Check if the player has passed GO. Make required updates if yes.

IF (newLoc = 0 OR (newLoc - curLoc) < 0) THEN

        UPDATE player SET player.Amount = (player.Amount+200) WHERE player.Name = playerName;

END IF;


-- Check whether location is a property.

IF ((SELECT board.IsProperty FROM board WHERE board.Loc_id=newLoc)) THEN

        SET bankBal = (SELECT player.Amount FROM player WHERE player.Name=playerName);

        SET propAmt = (SELECT property.Value FROM property WHERE property.Id=newLoc);


        -- Check if the property has an owner.

        -- If there is no owner, the player is the new owner. Make required updates.

        IF ((SELECT property.Owner FROM property WHERE property.Id=newLoc) IS NULL) THEN UPDATE property SET
        Owner = playerID WHERE property.Id = newLoc;

                UPDATE player SET Amount = (bankBal-propAmt) WHERE player.Name = playerName; SELECT "Updated
                Property Table" AS "Property Table";

                SELECT * FROM propertyTable;
```

```
        ELSE
                -- If there is an owner, then calculate the fine and make required updates.
                SET curOwn = (SELECT property.Owner FROM property WHERE property.Id=newLoc);
                SET ownBank = (SELECT player.Amount FROM player WHERE player.playerID=curOwn);
                SET fine = propAmt;
                SET colour = (SELECT property.Colour FROM property WHERE property.Id=newLoc);
        IF ((SELECT COUNT(*) FROM property p WHERE p.Owner=curOwn AND p.Colour=colour)=2) THEN
                SET fine=(fine*2);
        END IF;
        UPDATE player SET Amount = (bankBal-fine) WHERE player.Name = playerName;
        UPDATE player SET Amount = (ownBank+fine) WHERE player.playerID = curOwn;
END IF; ELSE
-- Check which Bonus location player is at and perform the required updates.
        IF (newLoc=2) THEN
                SET players = (SELECT COUNT(*) FROM player);
                UPDATE player SET Amount = (player.Amount-((players-1)*50)) WHERE player.Name = playerName;
                UPDATE player SET Amount = (player.Amount+50) WHERE player.Name != playerName;
                UPDATE player SET bonus = (SELECT b.Name FROM bonus b WHERE b.Id=newLoc) WHERE player.Name
                = playerName;
        ELSEIF (newLoc=4) THEN
                UPDATE player SET bonus = (SELECT b.Name FROM bonus b WHERE b.Id=newLoc) WHERE player.Name
= playerName;
        ELSEIF (newLoc=6) THEN
                UPDATE player SET Amount = (player.Amount+100) WHERE player.Name = playerName;
                UPDATE player SET bonus = (SELECT b.Name FROM bonus b WHERE b.Id=newLoc) WHERE player.Name
= playerName;
        ELSEIF (newLoc=8) THEN
                UPDATE player SET bonus = (SELECT b.Name FROM bonus b WHERE b.Id=newLoc) WHERE player.Name
= playerName;
        ELSEIF (newLoc=10) THEN
                UPDATE player SET bonus = (SELECT b.Name FROM bonus b WHERE b.Id=newLoc) WHERE player.Name
= playerName;
                CALL takeTurn(playerName,3); ELSEIF (newLoc=12) THEN
                UPDATE player SET player.Location = 4 WHERE player.Name = playerName;
                UPDATE player SET bonus = (SELECT b.Name FROM bonus b WHERE b.Id=newLoc) WHERE player.Name
= playerName;
                SET newLoc = 4; ELSEIF (newLoc=14) THEN
                UPDATE player SET Amount = (player.Amount-30) WHERE player.Name = playerName;
```

```
            UPDATE player SET bonus = (SELECT b.Name FROM bonus b WHERE b.Id=newLoc) WHERE player.Name
            = playerName;

        END IF; END IF;

SET bankBal = (SELECT player.Amount FROM player WHERE player.Name = playerName);

-- insert the details of that turn into the log.

INSERT INTO log (LogID, PlayerID, CurLoc, CurBank, Round) VALUES (NULL,playerID,newLoc,bankBal,round);

END
```

Procedure for simulating sample Game:

```
CREATE PROCEDURE playGame () BEGIN

DECLARE round INT DEFAULT 1;

ALTER TABLE log AUTO_INCREMENT = 1;

CREATE VIEW gameView AS SELECT

log.Round, player.playerID, player.Name, player.Amount, player.Location, (SELECT GROUP_CONCAT(property.Name)
FROM property WHERE player.playerID = property.Owner) AS Properties, player.Bonus FROM player, log WHERE
player.playerID = log.PlayerID AND player.Location = log.CurLoc;

CREATE VIEW playerTable AS SELECT * FROM player; CREATE VIEW propertyTable AS SELECT * FROM property;

SELECT round AS "Round Number";

CALL takeTurn("Jane", 3, NULL, round); CALL viewTables();

CALL takeTurn("Norman", 1, NULL, round); CALL viewTables();

CALL takeTurn("Mary", 4, NULL, round); CALL viewTables();

CALL takeTurn("Bill", 2, NULL, round); CALL viewTables();

SELECT * FROM gameView;

SET round=round+1;

SELECT round AS "Round Number";

CALL takeTurn("Jane", 5, NULL, round); CALL viewTables();

CALL takeTurn("Norman", 4, NULL, round); CALL viewTables();

CALL takeTurn("Mary", 6, 5, round); CALL viewTables();

CALL takeTurn("Bill", 6, 3, round); CALL viewTables();


SELECT * FROM gameView;


END
```