Submitted Date – 04/15/2018

BALLER INDEX

(INFO 6210, Spring 2018, Chaiyaporn Mutsalklisana)

Team Star

Jayanth Chava

Ramit Jaal

Sreya Pedamella

Siddhi Kulkarni

Soumya Zacharia

TABLE OF CONTENT

ABSTRACT

Sports are the biggest recreation in the world, and the current rate of commercialization in sports is growing exponentially. Gambling is a part of the human condition, and winning creates an uncontrollable frenzy which unknowingly becomes an addiction. The whole sports entertainment industry is based on the human tendencies to gamble. This database project is centered on the concept of an athlete stock exchange. This platform allows sports fanatics to invest and trade stocks of their beloved players. Each athlete's stock is subjected to change in price with respect to his performance in real game-time scenarios. The dynamic pricing of athlete's stocks creates a good opportunity for the ardent sports followers to make them financially independent. This project is aimed at building the backend and frontend for this athlete stock exchange. Each Athlete is given a preset value based on his career averages and current real-time form. The prices of the athletes are updated after every game the athlete takes part in, if the player performs better than his career averages there will be a bump in his initial preset price, likewise his price will drop in case he performs below his career averages. The core mathematical algorithm that is the main centerpiece of the project increases the stock of the players accurately with improvement in form and averages. This mathematical model also considers the different positions in the game. The user interface gives us an interactive platform for trading athlete stocks. The interface also includes many features such as watchlist and portfolio. The aim of this project was to build a website for the implementation of our athlete stock exchange database. Using principles of PHP and SQL we have managed to create a prototype of a real-time athlete stock exchange called Baller Index.

PROBLEM STATEMENT

America's sports industry is worth a whopping $422 billion, employing 1% of the population with almost 3 million sports-related jobs. Fans go insane when it comes to collecting sports memorabilia and attending games, and both are only becoming more expensive, but fans aren't becoming any less interested. Both the demand and the reckless enthusiasm to buy tickets at any cost are reasons for this needless inflation, and those printing price tags know this well enough to continue pumping money out of the herd of sports enthusiasts who are willing to dole it out. For far too long the fans have been exploited to increase the market value of the sports industry. Sport team owners are pumping more fund into their teams, athletes are stepping up their fitness levels. Overall the sports industry is on an upward growth curve. The only downside to this growing trend of sports industry is the growing prices of tickets and sports merchandise. It may sound cynical but an analysis of the industry will give you a broader picture of how much revenue is being generated. An argument could be made that this is how the sports industry has been functioning ever since the dawn of its creation. As true as that is, the fans need to still make a living out of their day jobs so they could continue watching their favorite athletes and teams perform. The only proposed solution to this problem would be that the industry provides for their fans in a manner which would help increase awareness of the sport and bring the team and fans closer. Baller Index is an interactive interface built for trading athlete stocks. Through this project, we are developing an alternate career path for the public and sports fanatics.

OBJECTIVE & SCOPE

We have built a database management system and a user interface which allows you to invest in the stock of players. Each player will be given a preset value based on his career averages and current real-time form. The prices of the players are subject to dynamic pricing and will be updated daily, if the player performs better than his career averages we will see a bump in his initial preset price, likewise his price will drop in case he performs below his career averages. So, if you are an avid sports fanatic you probably have an eye on the next breakout star. It's time to put your research passion and instincts in the right direction and make some money. Baller Index gives the viewers and supporters a chance to invest in their players. The objective of this project is to "bring the game to the viewers" in actual terms. Through this project we are enabling viewers and supporters the chance to not only support the players but also invest in them and have the chance to make some money of their idols. We intend to give all the sports fanatics a chance to find an alternative career path. By introducing Baller Index to the investment market, we intend to start a viral trend for all the sports fans out there, and maybe at some point, it will turn into a real-time stock market. The scope of this project is immense we may be creating the next big thing like the cryptocurrency market, with the right application and good marketing. The success of this project will only mean that other unexploited sports domains can be utilized with the same model to create a stock market.

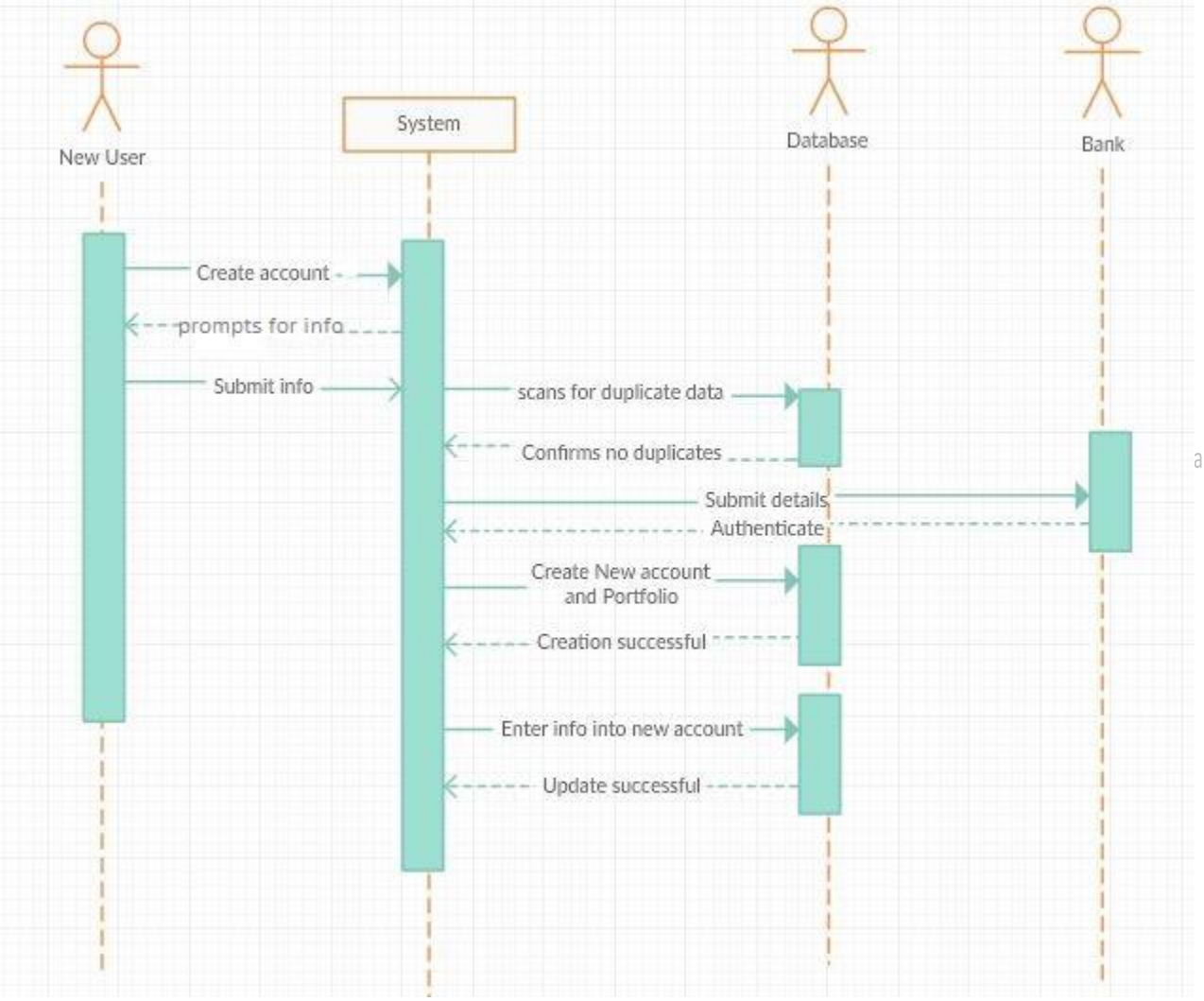
ASSUMPTION AND CONSTRAINTS

- Buyers and Sellers are available in the Baller Index
- Stats factor taken into consideration only if the player plays the game
- Initial price of the player is decided by the player rater (Top 10 = $10, Top 11-20 = $7, rest of the players = $5)
- Price of an individual share cannot go below $0.01
- When the player retires the shares will be void, and cannot be traded in the market.
- Offseason the market will not be affected by stats.

SEQUENCE DIAGRAMS

1. Create Account:

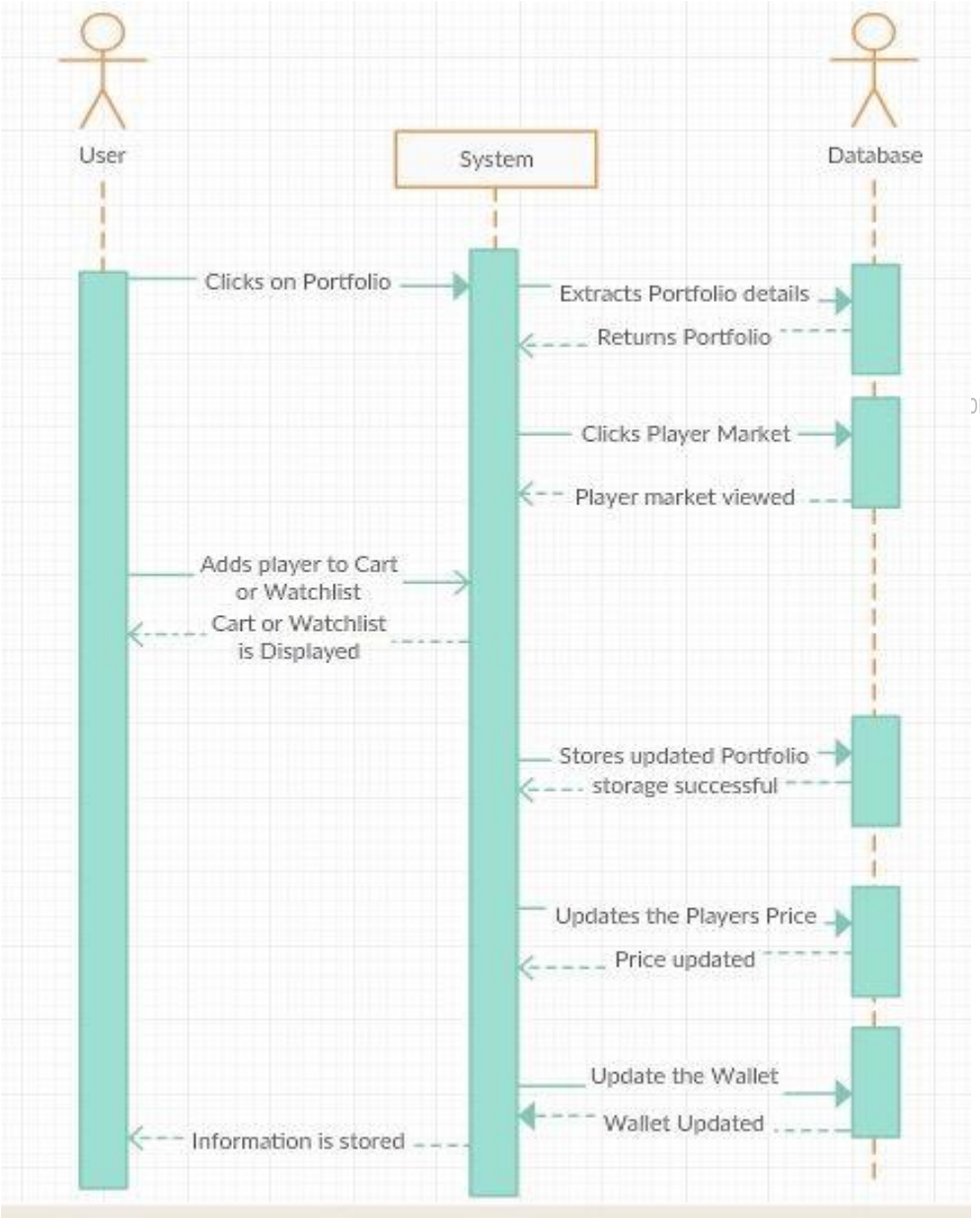
This sequence diagram shows the interaction of a **new user** with our system. The new user enters his credentials into our sign-up form. On entering his bank details our system requires an authentication from the bank to confirm the new user's bank details. After this step the customer has completed the sign-up procedure and now holds an account in Baller Index.



Reference: www.uml-diagrams .com

2. Buy Stock:

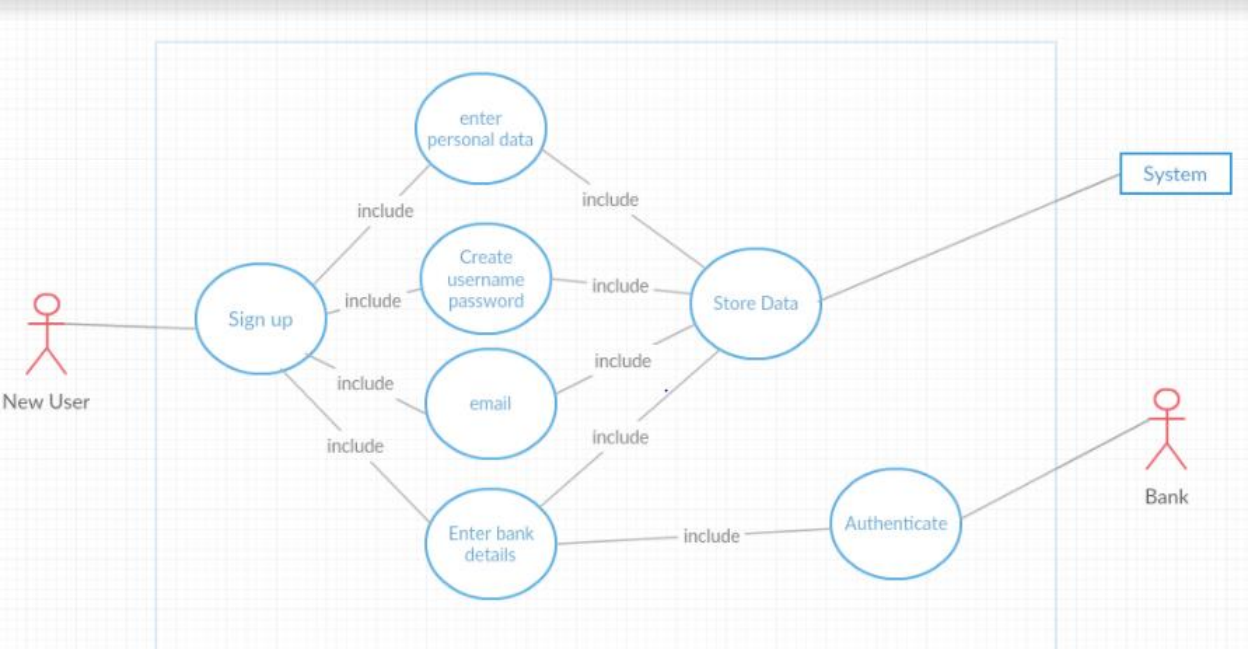
To buy stocks of a player, the user selects individual player from the Player Market. The user has an option to add the player either to the Watchlist or to the Cart. Watchlist gives him an option to keep close look at the player he wishes to Buy in future and Cart stores the player for check out. Either one action is performed on a Player If a share of a player is bought the user Portfolio updates and the stocks of the player are updated and this information is stored.



Reference: www.uml-diagrams .com

USE CASE DIAGRAMS

1. NEW USER

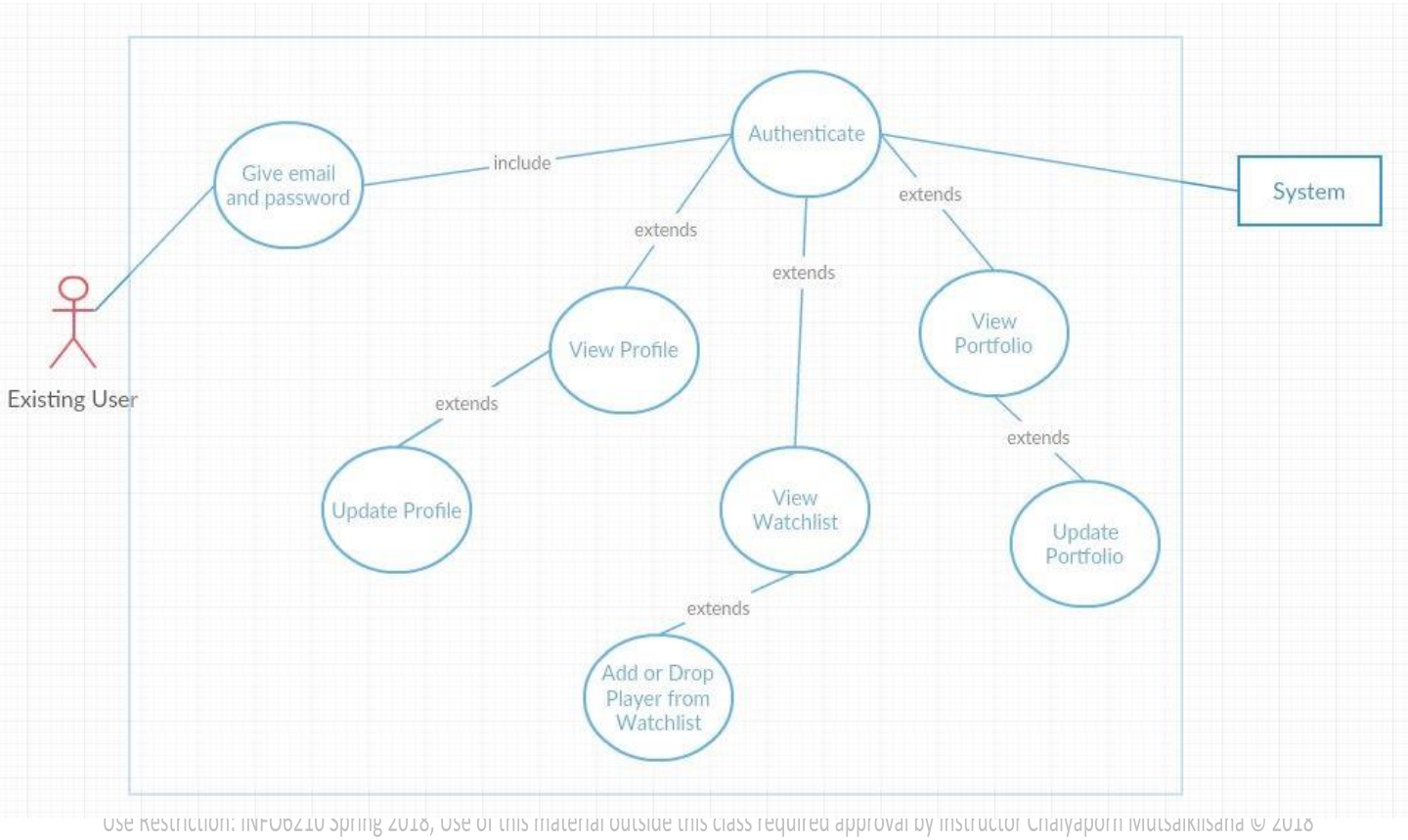


For Signing Up, the User enters the email, password, his personal data and bank details from the System User Interface of Baller Index. The entered details are stored in the data base. The Bank details entered are validated by the bank

2. Existing Customer

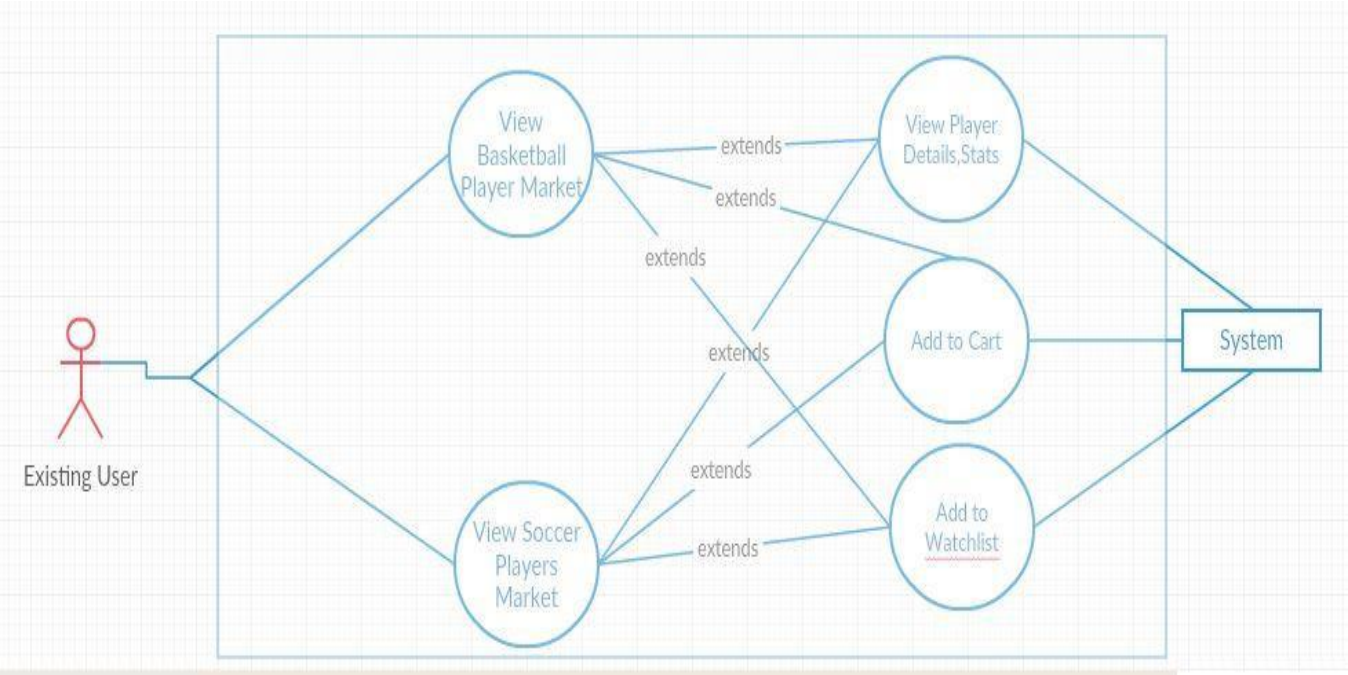
An existing user enters his email ID and password to login, these are authenticated by the system by comparing with the data in the database .After the customer logs into the system, he can navigate into his profile, watchlist , Portfolio. Customer can also add or remove players from the Watch list and Update his profile.

Reference: www.uml-diagrams .com



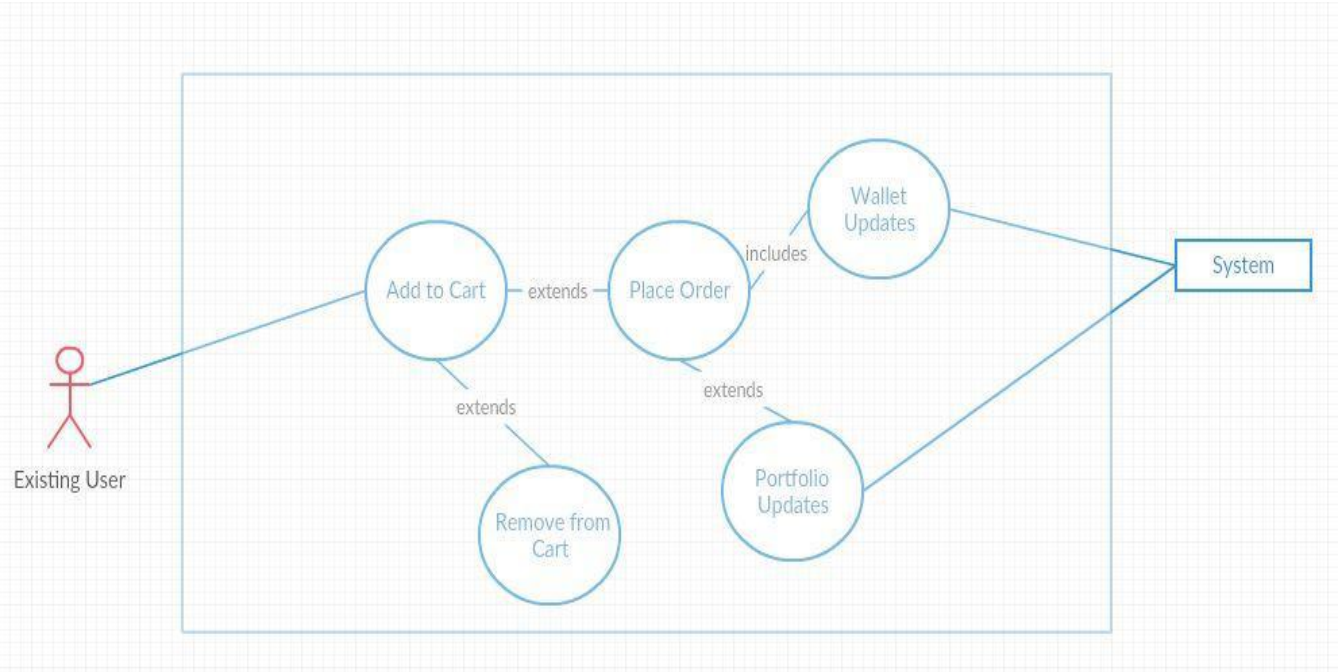
3. Actions on Player Market:

This is a use case where the user interacts with the dynamic player market. The BallerIndex has players from two sports:- NBA(National Basketball Association) and MLS(Major League Soccer). When the customer logs in he can see two player markets- Basketball player market and the Soccer player market. Customer clicks on any of these ,which gives him the option to View the Player details ,Stats, Add Players to the cart, Add Players to the watch list.



4.Cart

If the User wishes to buy a share from the player market, he adds the Players and the number of shares of that player to the cart. Later he can remove it from cart as well. After adding to cart ,customer can place Order which contains all the items added to his cart. Making an order reduces the Wallet balance and updates the Portfolio.



Reference: www.uml-diagrams .com

ENTITY RELATIONSHIP DIAGRAM

Before reaching the final ER diagram we had to go through multiple ER diagrams, rectifying each ER diagram step by step. Through this whole process of building an ER diagram we understood the importance of having a correct ER diagram for proper implementation of our project. Our original ER diagram had 6 tables. After consulting our professor and TA, they suggested that we design sequence and use case diagrams to have better understanding of our project. This helped us design our final ER diagram which we then normalized.

Normalization

1NF

Definition:

- Each table cell should contain a single value.
- Each record needs to be unique.
- Tables should have a primary key.

1.Customer
{UserID, First Name, Last Name, Nationality, Age, SSN, Email, Password, Address, Zip,City, State, BankID, Name on card, Card number, Expiry Date}

2.PlayerMarket
{PlayerID, GameType, Gametype, StatstypeID, Statstype, Playername, Height, Weight, Nationality, Age, Buy Rate, Sell rate,}

3.Bstats
{BstatsID, StatsTypeID, PlayerID, Points, Rebounds, Assists, Steals, Blocks, 3pointersmade, FG%, FT%, 3PFG%, Turnovers, Personal fouls}

4. Fstats
(FstatID, StatsTypeID, PlayerID, Goals scored, Assists, Saves, Interceptions, Clearances, Blocks, Goals Conceded, Yellow Card, Red Card, Penalty Missed, Own goal, Minutes Played, Pass Percentage, Clean sheet)
5.Cart
{CartID, UserID, PlayerID, No. of shares, Amount}
6.Portfolio
{PortfolioID, PlayerID, UserID, No. of shares, Buy rate, Sell Rate, Total stock Price}

7. Wallet {WalletID, UserID, Balance}
8.Watchlist {WatchlistID, PlayerID, UserID}
9. Selling (Selling orderID, UserID, PlayerID, Sellrate, No. of shares, Amount earned, WalletID)
10. Orders (OrderID, CartID, UserID, Total Amount, Wallet ID)

2NF:

Definition:

- A table is in 2nd Normal Form if:
- The table is in 1st normal form, and
- All the non-key columns are dependent on the table's primary key

1.Customer {UserID, First Name, Last Name, Nationality, Age, SSN, Email, Password,}
2.Address {Address, Zip}
3.State {Zip, City, State}
4.Bank {Bank ID, Name on card, Card number, Expiry Date}
5.Bstats

{BstatsID, StatsTypeID, PlayerID, Points, Rebounds, Assists, Steals, Blocks, 3pointersmade, FG%, FT%, 3PFG%, Turnovers, Personal fouls}

4. Fstats
{FstatID, StatsTypeID, PlayerID, Goals scored, Assists, Saves, Interceptions, Clearances, Blocks, Goals Conceded, Yellow Card, Red Card, Penalty Missed, Own goal, Minutes Played, Pass Percentage, Clean sheet}
5.PlayerMarket {PlayerID, Playername, Height, Weight, Nationality, Age}
6.Gametype (GametypeID, Gametype)
7.StatsType (StatstypeID, Statstype)
8. Price (PriceID, Buy Rate, Sell rate)
5.Cart {CartID, UserID, PlayerID, No. of shares, Amount}
6.Portfolio {PortfolioID, PlayerID, UserID, No. of shares, Buy rate, Sell Rate, Total stock Price}
7. Wallet {WalletID, UserID, Balance}
8.Watchlist {WatchlistID, PlayerID, UserID}
9. Selling {SellingorderID, UserID, PlayerID, Sellrate, No. of shares, Amount earned, WalletID}
10. Orders {OrderID, CartID, UserID, Total Amount, Wallet ID}

Reference: www.stackoverflow.com

3NF

Definition:

- A table is in third normal form if:
- A table is in 2nd normal form.
- It contains only columns that are non-transitively dependent on the primary key

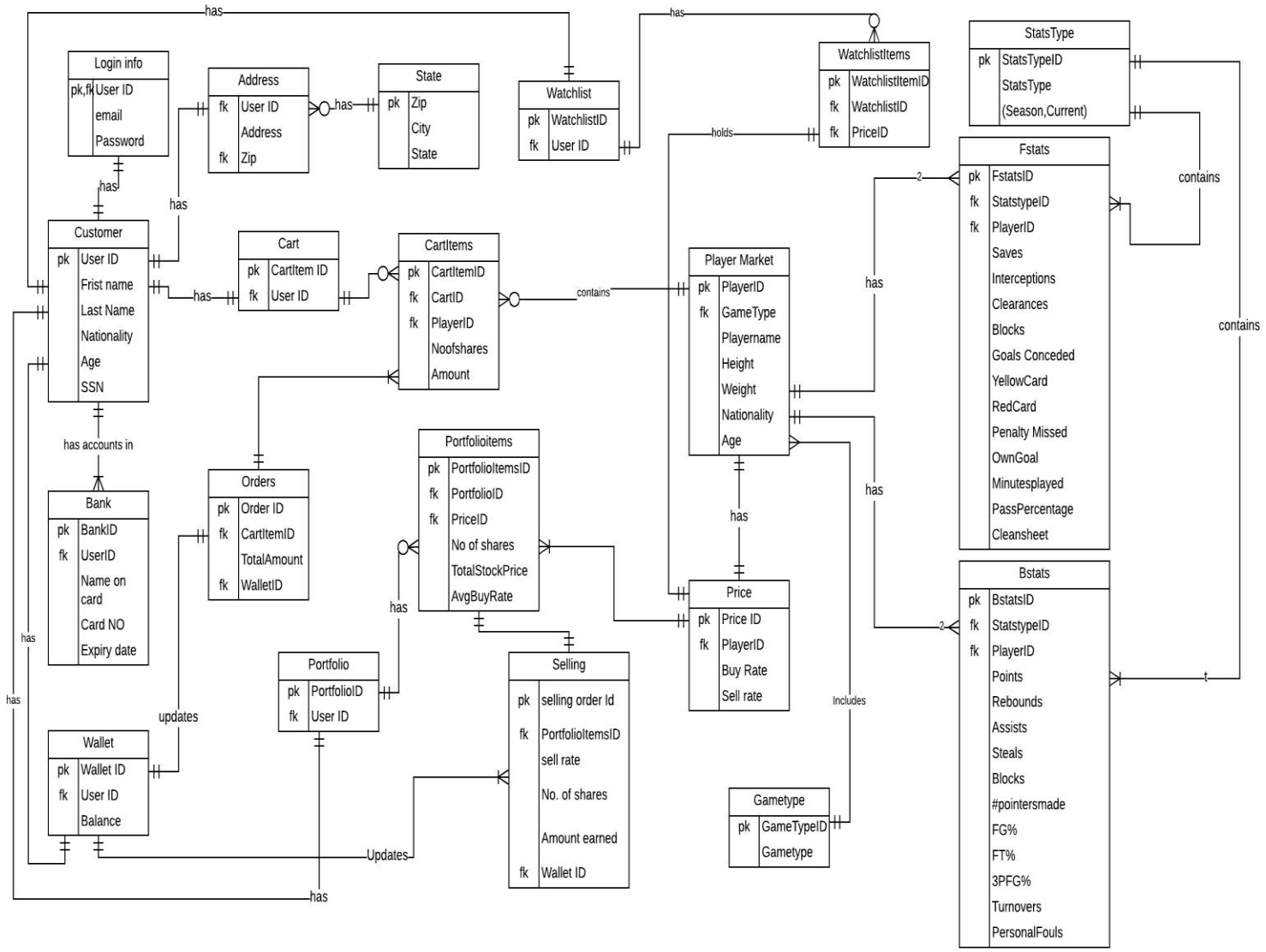
FINAL ENTITY RELATIONSHIP DIAGRAM

We have 20 tables in our entity relationship diagram

- Login Info (UserID, email, Password)
- Customer (UserID, First Name, Last Name, Nationality, Age, SSN)
- Bank (BankID, UserID, Name on card, Card number, Expiry Date)
- Address (UserID, Address, Zip)
- State (Zip, City, State)
- Cart (CartitemID, UserID)
- Cartitems (CartitemID, CartID, UserID, PlayerID, No. of shares, Amount)
- Wallet (WalletID, UserID, Balance)

- Orders (OrderID, CartitemsID, UserID, Total Amount, Wallet ID)
- Portfolio (PortfolioId, UserID)
- PortfolioitemsID (PortfolioitemsID, PortfolioID, PlayerID, UserID, No. of shares, Intial Buy rate, Total stock Price)
- Watchlist (watchlist Id, UserId)
- WatchlistItems (WatchlistitemsID, WatchlistID, PlayerID, UserID, PriceID)
- PlayerMarket (PlayerID, GameType, Playername, Height, Weight, Nationality, Age)
- Selling (Selling orderID, UserID, PlayerID, Sellrate, No. of shares, Amount earned, WalletID)
- Price (PriceID, Buy Rate, Sell rate)
- Gametype (GametypeID, Gametype)
- StatsType (StatstypeID, Statstype)
- Bstats (BstatsID, StatsTypeID, PlayerID,
    - Points, Rebounds, Assists,
    - Steals, Blocks, 3pointersmade,
    - FG%, FT%, 3PFG%,
    - Turnovers, Personal fouls)
- Fstats (FstatID, StatsTypeID, PlayerID,
    - Goals scored, Assists, Saves,

- Interceptions, Clearances, Blocks,
- Goals Conceded, Yellow Card, Red Card, Penalty Missed, Own goal,
- Minutes Played, Pass Percentage, Clean sheet)



References: www.visual-paradigm.com

CARDINALITIES:

- A CUSTOMER has LOGIN INFO. (one to One)

  Each customer will have one login credentials. Each row in login info table belongs to only one Customer.

- A CUSTOMER has BANK details. (one to Many)

  Each customer can have multiple bank account details. Each row in Customer table corresponds to one or many rows in bank table.

- A CUSTOMER has a CART. (One to One)

  Every customer has a cart. Each cart belongs to one Customer

- A CART has CARTITEMS. (One to Many)

  Each customer will have only one CART which can have multiple CARTITEM

- CARTITEMS will contain players from PLAYERMARKET. (Many to One)

  One player can be present in one or more Cartitems.

- A CUSTOMER has ADDRESS. (Many to One)

  Each customer will have one address. Multiple Customers can have the same address.

- ADDRESS table has STATE. (Optional many to One)

  Each address will have corresponding Zip, City and State from STATE table. Each row in the STATE table may be linked to multiple ADDRESS.

- A CUSTOMER has WALLET. (One to One)

  Each customer will have only one wallet.

- ORDER has CARTITEMS. (One to Many)

  A customer can place one order which can have multiple cartitems (player stocks). One cart item is present in only one order.

- ORDER will update WALLET. (One to One)

  When an Order is placed by a customer balance in his Wallet will get updated.

- A CUSTOMER has PORTFOLIO. (One to One)

  A customer will have only one Portfolio. Each portfolio belongs to one customer.

- PORTFOLIO has PORTFOLIOITEMS. (One to Optional Many)

  A portfolio may or may not have multiple Portfolioitems.

- SELLING updates WALLET. (One to One)

  Wallet will get updated once a player stock is sold.

- A CUSTOMER has WATCHLIST. (One to One)

  Each customer will have only one watchlist.

- WATCHLIST has WATCHLISTITEMS.(One to Optional Many)

"Use Restr One watchlist may or may not have multiple watchlistitems.

- WATCHLISTITEMS has PlayerMarket. (One to Many)

  A user can add one or more players to the watchlist. One player might be present in one or more watchlistitems.

- PLAYERMARKET has FSTATS. (One to Many)

  Soccer Players from playermarket will have multiple stats in Fstats.

- PLAYERMARKET has BSTATS. (One to Many)

  Basketball Players from playermarket will have multiple stats in Bstats

- PLAYERMARKET has GAMETYPE. (Many to One)

  Players from playemarket will have GametypeID2 for Basketball, Gametype1 for Soccer

- PLAYERMARKET has PRICE (One to One)

Players from playermarket will have price details in Price. Every row in price table corresponds to one player

- PORTFOLIOITEMS has PRICE (One to One)

Portfolioitems has details of player stock prices.

- FSTATS contains STATSTYPE

Fstats contains current and season stats. Multiple players. Statstype table has two entries- Career Stats, Current game stats. The Stats Type table is linked to multiple players

- STATSTYPE contain BSTATS - Bstats contains current and season stats.

References: www.visual-paradigm.com

NATIONAL BASKETBALL ASSOCIATION STATISTICS GRADING & PRICE ALGORITHM

- Primary Stats (Players accumulate these stats in abundant)

  • Points – Every field goal made is accumulated as points, if the ball is shot from inside the 3-point line it is 2 points and if its shot from outside the 3-point line it is 3 points, each free throw is one point.

  • Rebounds – Every time a basketball is shot and it misses the basket, the ball bounces off the board or rim, and the player who collects it accumulates a rebound.

  • Assists – When the basketball player makes the pass to the player before the points are scored he records an assist.

- Secondary Stats (Important stats but are usually with high variance)

  • 3 Pointers made – it's the accumulation of the number of 3-point field goals made by the athlete.

  • Steals – When a player pressurizes an opponent player and causes him to lose possession and thereby helps his team gain possession it's called a steal.

  • Blocks – when a player protects the rim and swats the basketball away on an attempt from an opposing player to make a field goal it's called a block.

- Percentages (Calculated in ratios and Percentages)

  • Field Goal Percentage – This is a percentage calculated with respect to the number of baskets made from inside the 3point line to the number of attempts to make those numbers of baskets.

  • Free Throw Percentage – This is a percentage calculated as the number of free throws made to the number of free throws attempted.

  • 3 Point Shooting Percentage – This is a percentage calculated as the number of 3-point field goals made as compared to the number of attempts made to make those baskets.

- Negative Stats (These stats have a negative impact on the game)

  • Turnovers – It's when a player having the ball possession loses the ball under pressure of the opposition or makes a bad pass.

  • Personal fouls – fouls committed by the athlete in a game.

Price Determination Algorithm For National Basketball Association

(Primary stats of the game(PSG) ---- Season Average primary stats(CAPS)) X 0.1

+

(Secondary stats of the game (SSG) ---- Season Average secondary stats(CASS)) X 0.2

+

(Percentages stats of the game(PESG) ---- Season Average Percentages(CAPE))/100

+

(Negative stats of the game(NSG) ---- Season Negatives Average stats(CNAS)) X 0.3

Total (can be positive or negative) + Old Price = New updated Price

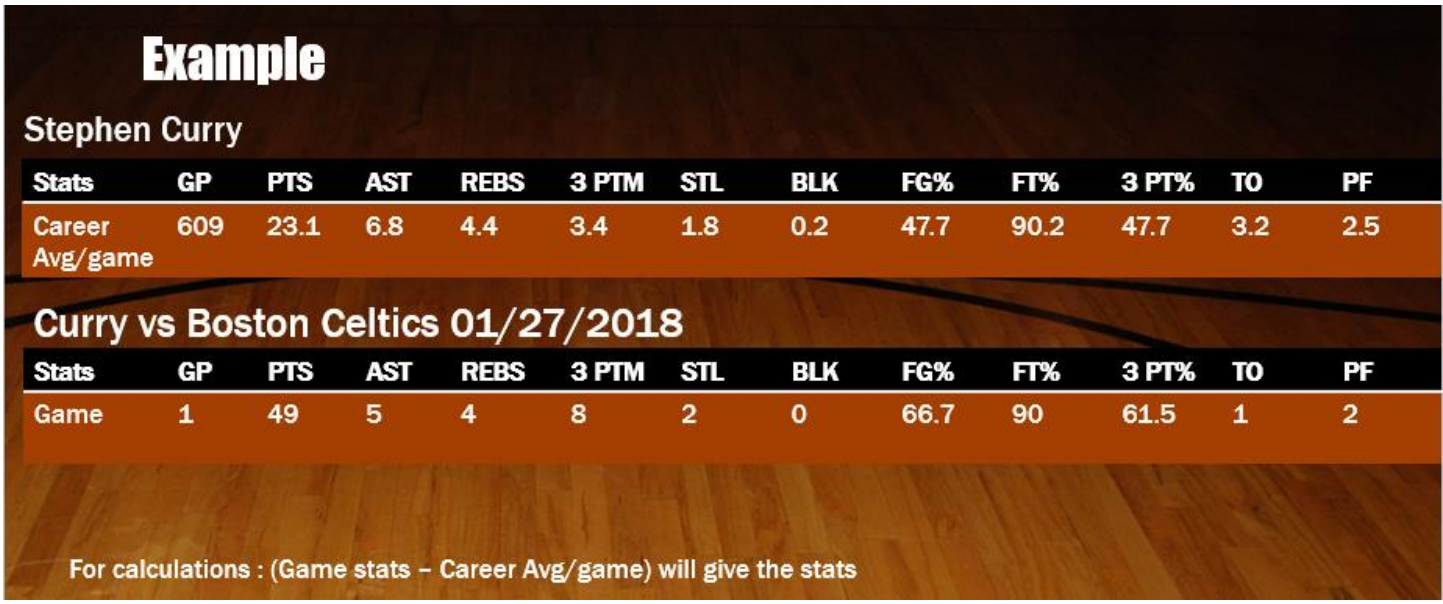
This algorithm compares each athlete's stats to his career's average to determine his initial value. The increase or decrease of the athlete's price is calculated by comparing the athlete's season averages to his current games stats.

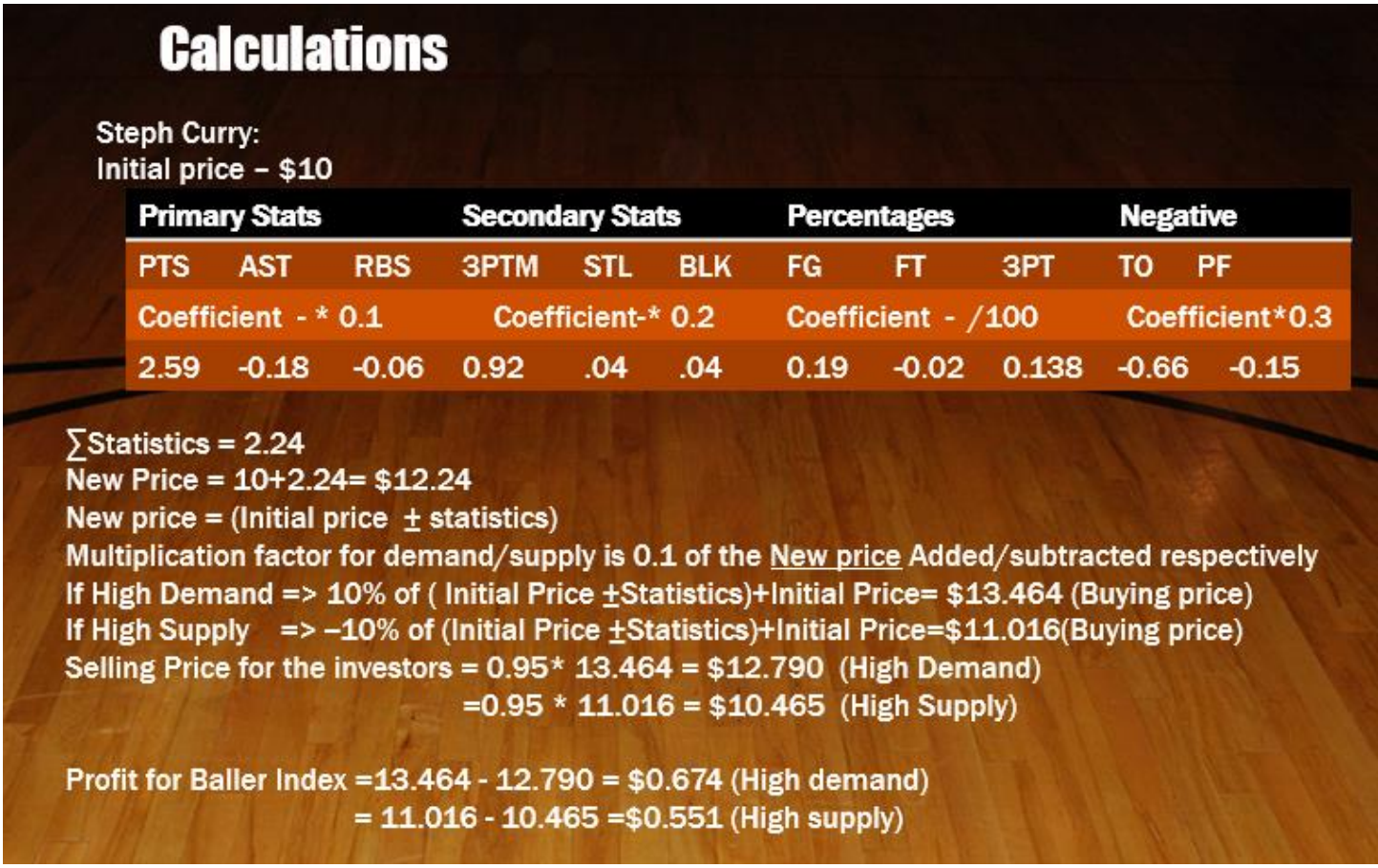
http://www.nba.com/players/stephen/curry/201939

https://www.basketball-reference.com/

http://www.rotoworld.com/sports/nba/basketball?ls=roto:nba:gnav

Example of National Basketball Association Price algorithm





MAJOR LEAGUE SOCCER STATISTICS GRADING & PRICE ALGORITHM

- Primary Stats (Players accumulate these stats in abundant)

  • Goals – When a player manages to put the football in the opponent's goal, its accumulated as a goal scored

  • Assists – The player who makes the pass to the goal scorer records an assist.

- Secondary Stats (Important stats but are usually with high variance)

  • Save - This stat only for the goalkeepers, every time a keeper stops the football from going into his goal it's recorded as a save.

  • Interceptions – when a player interferes in a pass and manages to get possession to his team it's recorded as an interception.

  • Clearances – When a defender clears the ball when the opposite team is applying pressure it's recorded as a clearance.

  • Blocks- when a player puts himself legally in between the player who is shooting the ball and his goal it's recorded as a block.

  • Goals conceded – This is a stat which is accumulated on basis of the number of goals conceded in a match.

- Percentages (Calculated in ratios and percentages)

  • Minutes played – This is the number of minutes played by the player out of 90.
  • Pass percentage – This is the number of passes made successfully as compared to the number of passes attempted.

- Negative Stats (These stats have a negative impact on the game)

  - Yellow card – This is a warning given to a player when he has committed 5 fouls or one rash foul.
  - Red card – When this is given the player is ejected from the game instantly, it can be given as an accumulation of two red cards or for a rash tackle.
  - Penalty missed – This is a stat recorded when the player misses the penalty.
  - Own goal – when a player scores a goal in his own goal this stat is recorded.

- Bonus stats

  • Clean sheet – When the team does not concede a goal the team will record a clean sheet.

Price Determination Algorithm Major league soccer

(Primary stats of the game(PSG) ---- Career Average primary stats(CAPS)) X 0.2

+

(Secondary stats of the game (SSG) ---- Career Average secondary stats(CASS)) X 0.3

+

(Percentages stats of the game(PESG) ---- Career Average Percentages(CAPE))/100 + Minutes played/90

+

(Negative stats of the game(NSG) ---- Career Negative Average stats(CNAS)) X 0.3

+

(Bonus stats) X 0.3

Total (can be positive or negative) + Old Price = New updated Price

This algorithm compares each athlete's stats to his career's average to determine his initial value. The increase or decrease of the athlete's price is calculated by comparing the athlete's season averages to his current games stats.

- www.statista.com
- www.mlssoccer.com

Example of MLS price Algorithm

	A	B	C	D	E	F	G	H	I	J	K	L
1		Career stat 2016-17				Game stats				Price change		
2	Player Name	Andre Blake	Matt Hedges	Sebastian Gio		Andre	Matt	Sebastian		Andre	Matt	Sebastian
3	Position	Goalkeeper	Defender	Attacker								
4	Games	30.0	24.0	27.0								
5												
6	Primary Stats *0.2											
7	Goals Scored	0.0	1.0	0.6		0	0	1		0.0	0.2	0.0
8	Assists	0.0	0.0	0.5		0	0	1		0.0	0.0	-0.1
9	Secondary Stats *0.3											
10	Saves	2.6	0.0	0.0		3	0	0		-0.1	0.0	0.0
11	Interceptions	0.1	3.8	0.3		0	3	1		0.0	0.3	-0.2
12	Chances Created	0.0	0.2	1.8		0	1	2		0.0	-0.2	-0.1
13	Key Passes	0.0	0.2	1.3		0	1	1		0.0	-0.2	0.1
14	Clearaces	0.4	4.2	0.0		2	4	0		-0.5	0.1	0.0
15	Blocks	0.0	0.5	0.0		0	2	0		0.0	-0.1	0.0
16	Goals Conceded	1.5	0.0	0.0		2	0	0		-0.1	0.0	0.0
17	Negative stats * -0.4											
18	Yellow Cards	0.0	0.2	0.1		0	0	1		0.0	0.0	0.4
19	Red Cards	0.0	0.0	0.0		0	0	0		0.0	0.0	0.0
20	Bonus Stats *0.4											
21	Goals From Penalty	0.0	0.0	0.1		0	0	0		0.0	0.0	0.1
22	Clean Sheets	0.2	0.0	0.0		0	0	0		0.1	0.0	0.0
23	Percentage stats											
24	Pass Completion	61.0	81.0	73.0		75	79	63		-0.1	0.0	0.1
25	Minutes played	30.0	23.9	25.8						0.3	0.2	0.3
26										**-0.4**	**0.1**	**0.5**

Code Implementation

Front End: HTML, CSS

Back End: Php, MySQL

Software Requirement: WAMP SERVER, Notepad++.

Executed MYSQL Queries

- 20 Tables
- 12 Stored Procedure
- 3 Joins
- 2 triggers
- 4 Views

- Users and Privileges
    - New user Investor – not given access to the database
    - System admin – select, insert, update, create tables, but he can't create a new user
    - Analyst - select, insert, update but he can't alter tables and create tables.

www.docs.microsoft.com

www.dba.stackexchange.com

www.webdeveloper.com

TABLES

1. **Login Info**

```
CREATE TABLE `Logininfo` (
  `UserID` integer(5) NOT NULL,
  `Email` varchar(100) NOT NULL,
  `Password` varchar(15) NOT NULL,
  primary key (`UserID`,`Email`),
  foreign key (`UserID`) references `Customer` (`UserID`)
);
```

2. **State**

```
CREATE TABLE `State`(
  `Zip` integer(7) NOT NULL,
`City` Varchar(20) NOT NULL,
`State` varchar(20) NOT NULL,
primary key (`Zip`)
);
```

3. **Address**


```
CREATE TABLE `Address`(
  `UserID` integer(5) NOT NULL,
  `Address` varchar(20) NOT NULL,
  `Zip` integer(7) NOT NULL,
  foreign key (`UserID`) references Customer(`UserID`),
  foreign key (`Zip`) references State(`Zip`)
);
```

4. **Customer**

```
CREATE TABLE `Customer` (
 `UserID` integer(5) NOT NULL auto_increment,
`first name` varchar(20) NOT NULL,
`Last Name` varchar(20)NOT NULL,
`Nationality` varchar(20) NOT NULL,
`Age` integer NOT NULL,
  `SSN` integer(50) NOT NULL,
  primary key (`UserID`));
```

5. **Bank**

```
CREATE TABLE `Bank` (
`BankID` integer(5) NOT NULL auto_increment,
```

```
  `UserID` integer(5) NOT NULL,
  `Name on Card` varchar(50) NOT NULL,
  `Card NO` double NOT NULL,
  `Expiry Date` date NOT NULL,
   primary key (`BankID`),
   foreign key (`UserID`) references `Customer`(`UserID`));
```

6.GameType

```
 create table `GameType` (`GameTypeID` integer(5) NOT NULL,
`GameType` nvarchar(20),
primary key (`GameTypeID`));
```

7.StatsType

```
 create table `StatsType` (`StatsTypeID` integer(5) NOT NULL,
`StatsType` nvarchar(20),
primary key (`StatsTypeID`));
select*from statstype;
```


8. Player Market

```
  CREATE TABLE `PlayerMarket1` (
  `PlayerID` integer(5) NOT NULL auto_increment,
  `GameTypeID` integer(5) NOT NULL,
 `PlayerName` varchar(50) NOT NULL,
  `Height` numeric(4,2) NOT NULL,
  `Weight` numeric(4,2) NOT NULL,
  `Age` integer(5) NOT NULL,
  `Nationality` varchar(20) NOT NULL,
  primary key (`PlayerID`),
  foreign key (`GameTypeID`) references `GameType`(`GameTypeID`)
);
```

9.Price

```
CREATE TABLE `Price` (
  `PriceID` integer(5) NOT NULL auto_increment,
  `PlayerID` integer(5) NOT NULL,
  `BuyRate` numeric(4,2) NOT NULL,
  `SellRate` numeric(4,2) NOT NULL,
  primary key (`PriceID`),
  foreign key (`PlayerID`) references `PlayerMarket`(`PlayerID`));
```

10. Portfolio

```
 create table `portfolio`(
`PortfolioID` integer(5) NOT NULL auto_increment,
`UserID` integer(5) NOT NULL,primary key (`PortfolioID`),
 foreign key (`UserID`) references customer(`UserID`));
```

11. PortfolioItems

```
CREATE TABLE  `PortfolioItems` (
  `PortfolioItemID` integer(5) NOT NULL auto_increment,
  `PortfolioID` integer(5) NOT NULL,
  `PlayerID` integer(5) NOT NULL,
  `No of shares` integer(5) NOT NULL,
  `PriceID` integer(5) NOT NULL,
  `Profits` numeric(10,2),
  `UserID` integer(5),
  primary key(`PortfolioItemID`),
  foreign key (`UserID`) references `Customer` (`UserID`),
  foreign key (`PlayerID`) references `PlayerMarket` (`PlayerID`),
  foreign key (`PriceID`) references `Price` (`PriceID`),
  foreign key(`PortfolioID`) references portfolio(`PortfolioID`)
);
```


12. Cart

```
create table `Cart`(
    `CartID` integer(5) not null auto_increment,
    `UserID` integer(5) not null,
    primary key(`CartID`),  foreign key(`UserID`) references `Customer` (`UserID`));
```

13. Cartitems

```
CREATE TABLE `CartItems` (
  `CartItemsID` integer(5) NOT NULL auto_increment,
  `CartID` integer(5) not null,
  `UserID` integer(5) NOT NULL,
  `PlayerID` integer(5) NOT NULL,
   `Buy Rate` numeric(4,2) NOT NULL,
  `NoofShares` integer(5) NOT NULL,
  `Amount` integer(20) NOT NULL,
  primary key (`CartItemsID`),
   foreign key(`CartID`) references `Cart` (`CartID`),
  foreign key(`UserID`) references `Customer` (`UserID`),
  foreign key (`PlayerID`) references `PlayerMarket` (`PlayerID`)
);
```

14. Orders

```
  CREATE TABLE `Orders` (
  `OrderID` integer(5) NOT NULL auto_increment,
  `CartID` integer(5) NOT NULL,
  `UserID` integer(5) NOT NULL,
  `TotalAmount` integer(20) NOT NULL,
  `WalletID` integer(5) NOT NULL,
  primary key (`OrderID`),
  foreign key (`UserID`) references `Customer` (`UserID`),
   foreign key (`CartID`) references `Cart` (`CartID`) ,
   foreign key (`WalletID`) references `Wallet` (`WalletID`)
);
```


15. Wallet

```
CREATE TABLE `Wallet` (
  `WalletID` integer(5) NOT NULL auto_increment,
  `UserID` integer(5) NOT NULL,
  `Balance` integer(5) NOT NULL,
  primary key (`WalletID`),
  foreign key (`UserID`) references `Customer` (`UserID`)
);
```

16. Watchlist

```
CREATE TABLE `Watchlist` (
  `WatchlistID` integer(5) NOT NULL auto_increment,
  `UserID` integer(5) NOT NULL,
  primary key (`WatchlistID`),
  foreign key (`UserID`) references `Customer` (`UserID`)
  );
```

17. Watchlistitems

```
create table `WatchlistItems` (
`WatchlistItemID` integer(5) NOT NULL auto_increment,
`WatchlistID` integer(5) NOT NULL,
`PlayerID` integer(5) NOT NULL,
`UserID` integer(5) NOT NULL,
`PriceID` integer(5) NOT NULL,
primary key(`WatchlistItemID`),
foreign key(`UserID`) references `Customer` (`UserID`),
foreign key (`PlayerID`) references `PlayerMarket` (`PlayerID`),
  foreign key (`PriceID`) references `Price` (`PriceID`),
  foreign key(`WatchlistID`) references `watchlist`(`WatchlistID`)
);
```


18. Selling

```
CREATE TABLE `Selling` (
  `sellingorderId` integer(5) NOT NULL auto_increment,
  `UserID` integer(5) NOT NULL,
  `PlayerID` integer(5) NOT NULL,
  -- `Sell rate` integer(5) NOT NULL ,
  `Noofshares` integer(5) NOT NULL,
  `Amountearned` integer(10) NOT NULL,
  `WalletID` integer(10) NOT NULL,
  primary key (`sellingorderId`),
foreign key (`WalletID`) references `Wallet` (`WalletID`),
foreign key (`UserID`) references `Customer` (`UserID`),
foreign key (`PlayerID`) references `PlayerMarket` (`PlayerID`));
```

19. Fstats

```
CREATE TABLE `fstats` ( `StatsTypeID` integer(5) ,
```

```
`PlayerID` integer(5),`FstatsID` integer(5) not null auto_increment ,
`goals` numeric(2,2) not null, `assists` numeric(2,2) not null,
 Saves` numeric(2,2) NOT NULL, `Interceptions` numeric(2,2) NOT NULL,
`Clearances` numeric(2,2) NOT NULL, `Blocks` numeric(2,2) NOT NULL,
`GoalsConceded` numeric(2,2) NOT NULL,`YellowCard` numeric(2,2) NOT NULL,
`Redcard` numeric(2,2) NOT NULL,`PenaltyMissed` numeric(2,2) NOT NULL,
`OwnGoal` numeric(2,2) NOT NULL, `Minutesplayed/90` numeric(4,2) NOT NULL,
`PassPercentage` numeric(4,2) NOT NULL,`CleanSheet` integer(5) NOT NULL,
 primary key (`BstatsID`),foreign key (`PlayerID`) references `playermarket` (`PlayerID`),
 foreign key (`StatsTypeID`) references `StatsType` (`StatsTypeID`));
```

20. Bstats

```
CREATE TABLE `Bstats` (
  `StatsTypeID` integer(5) ,
  `PlayerID` integer(5),
  `BstatsID` integer(5) not null auto_increment ,
  `Points` numeric(3,2) NOT NULL,
  `Rebounds` numeric(3,2) NOT NULL,
  `Assists` numeric(3,2) NOT NULL,
  `Steals` numeric(3,2) NOT NULL,
  `Blocks` numeric(3,2) NOT NULL,
  `3pointersmade` numeric(3,2) NOT NULL,
  `FG%` numeric(3,2) NOT NULL,
  `FT%` numeric(3,2) NOT NULL,
  `3PFG%`numeric(3,2) NOT NULL,
  `Turnovers` numeric(3,2) NOT NULL,
  `PersonalFouls` numeric(3,2) NOT NULL,
  primary key (`BstatsID`),
  foreign key (`PlayerID`) references `playermarket` (`PlayerID`),
   foreign key (`StatsTypeID`) references `StatsType` (`StatsTypeID`)
 );
```

Errors faced during table creation

1364 Field doesn't have a default value: when we are inserting values into the table and we don't pass a value in the insert command for a attribute that is declared NOT NULL. To rectify this error, we make sure that we give all necessary values in table without leaving anything NULL

1215 cannot add foreign key constraint: The foreign key constraint is incorrectly formed when that key is not declared as primary key in any table. To rectify this error, we made sure that every foreign key in every table is already a primary key in another table.

1217 Cannot delete or update a parent row a foreign key constraint fails: This error arises when we try to delete a table which has a primary key that is used as a foreign key in another table. To rectify this error,

'Set foreign_key_checks=0;

Drop table Table_name;

Set foreign_key_checks=1'

.

STORED PROCEDURES

SPnewpriceB

This is a price determining procedure for basketball players and is one of the core procedures for our database project. This procedure helps us alter the price with respect to each game, thereby creating a dynamic pricing scenario that we original promised in our objectives. In this store procedure the players gameday stats are subtracted from his season stats to determine his new price. As earlier mentioned we divide the stats in to 4 different categories based on the accumulation of these stats in each game. Once the stats are updated we call the store procedure with an input parameter of PlayerID.

```
delimiter $$
create procedure `SPnewpriceB` (in iplayerID int(10))
begin
declare a integer(2); declare vpts decimal(5,2);
declare vast decimal(5,2); declare vrebs decimal(5,2);
declare v3pm decimal(5,2); declare vstl decimal(5,2);
declare vblk decimal(5,2); declare vfg decimal(5,2);
declare vft decimal(5,2); declare v3fg decimal(5,2);
declare vtov decimal(5,2); declare vpf decimal(5,2);
declare vpts1 decimal(5,2); declare vast1 decimal(5,2);
declare vrebs1 decimal(5,2); declare v3pm1 decimal(5,2);
declare vstl1 decimal(5,2); declare vblk1 decimal(5,2);
declare vfg1 decimal(5,2); declare vft1 decimal(5,2);
declare vtov1 decimal(5,2); declare vpf1 decimal(5,2);
declare v3fg1 decimal(5,2); declare vpri decimal(5,2);
declare vsec decimal(5,2); declare vper decimal(5,2);
declare vneg decimal(5,2); declare vpri1 decimal(5,2);
declare vsec1 decimal(5,2); declare vper1 decimal(5,2);
declare vneg1 decimal(5,2); declare changeinprice decimal(5,2);
declare currentprice decimal(5,2); declare newsellprice decimal(5,2);
```


/*In this store procedure many variables are declared to query and call the stat values and store them in the declared variables. Variables have been declared to store the calculations of each category of stats such as primary, secondary, negative, percentage. The table 'StatsType' is used to retrieve the stats, if the StatsTypeID= 1 it refers to current game stats and if the StatsTypeID=2 then it refers to season averages. */

```
set a=(select playerMarket.GametypeID from playermarket where playerID=iPlayerID);
if a=2 then /*GametypeID = 2 for Basketball & 1 for football(soccer)*/
set vpts = (select bstats.points from bstats where playerID=iplayerID and statstypeid = 1);
set vast = (select bstats.assists from bstats where playerID=iplayerID and statstypeid = 1);
set vrebs = (select bstats.rebounds from bstats where playerID=iplayerID and statstypeid = 1);
set v3pm = (select bstats.`3pointersmade` from bstats where playerID=iplayerID and statstypeid
= 1);
set vfg = (select bstats.`FG%` from bstats where playerID=iplayerID and statstypeid = 1);
set vft = (select bstats.`FT%` from bstats where playerID=iplayerID and statstypeid = 1);
set v3fg = (select bstats.`3PFG%` from bstats where playerID=iplayerID and statstypeid = 1);
set vtov = (select bstats.Turnovers from bstats where playerID=iplayerID and statstypeid = 1);
set vpf = (select bstats.PersonalFouls from bstats where playerID=iplayerID and statstypeid =
1);
set vstl = (select bstats.Steals from bstats where playerID=iplayerID and statstypeid = 1);
set vblk = (select bstats.Blocks from bstats where playerID=iplayerID and statstypeid = 1);
set vpts1 = (select bstats.points from bstats where playerID=iplayerID and statstypeid = 2);
set vast1 = (select bstats.assists from bstats where playerID=iplayerID and statstypeid = 2);
set vrebs1 = (select bstats.rebounds from bstats where playerID=iplayerID and statstypeid = 2);
set v3pm1 = (select bstats.`3pointersmade` from bstats where playerID=iplayerID and
statstypeid = 2);
set vfg1 = (select bstats.`FG%` from bstats where playerID=iplayerID and statstypeid = 2);
set vft1 = (select bstats.`FT%` from bstats where playerID=iplayerID and statstypeid = 2);
set v3fg1 = (select bstats.`3PFG%` from bstats where playerID=iplayerID and statstypeid = 2);
set vtov1 = (select bstats.Turnovers from bstats where playerID=iplayerID and statstypeid = 2);
set vpf1 = (select bstats.PersonalFouls from bstats where playerID=iplayerID and statstypeid =
2);
set vstl1 = (select bstats.Steals from bstats where playerID=iplayerID and statstypeid = 2);
set vblk1 = (select bstats.Blocks from bstats where playerID=iplayerID and statstypeid = 2);
set vpri = (vpts+vast+vrebs);
set vsec = (v3pm+vstl+vblk);
set vper = (vfg+vft+v3fg);
set vneg = (vtov+vpf);
set vpri1 = (vpts1+vast1+vrebs1);
set vsec1 = (v3pm1+vstl1+vblk1);
set vper1 = (vfg1+vft1+v3fg1);
set vneg1 = (vtov1+vpf1);
set vpri = (vpri-vpri1)*0.2;  /* The price algorithm begins here */
set vsec = (vsec-vsec1)*0.2;
set vper = (vper-vper1)/100;
set vneg = (vneg-vneg1)*0.2;
set changeinprice = (vpri+vsec+vper+vneg);
set currentprice = (select price.buyrate from price where playerID=iplayerID);
```

```
set currentprice = (currentprice+changeinprice); /*The price algorithm ends here */
update price set buyrate=currentprice where playerID=iplayerID; /* the Price table is updated */
set newsellprice = (select price.SellRate from price where playerID=iplayerID);
set newsellprice = (newsellprice+changeinprice);
update price set sellrate=newsellprice where playerID=iplayerID; /* Sell rate is updated */
select currentprice;
/* the change in price and current price is given as an output after the store procedure is
completed */
end if;
end$$
delimiter ;
```

SPnewpriceF

This is a price determining procedure for football (soccer) players. Every time a soccer player plays a game his stats will be updated to the database. After the stats are updated the SPnewpriceF store procedure is used to generate the new buy rate for the soccer athlete's stock, the procedure also develops the new sell rate simultaneously. The procedure requires an input parameter of the PlayerID, this is the PlayerID of the player whose stats are being updated.

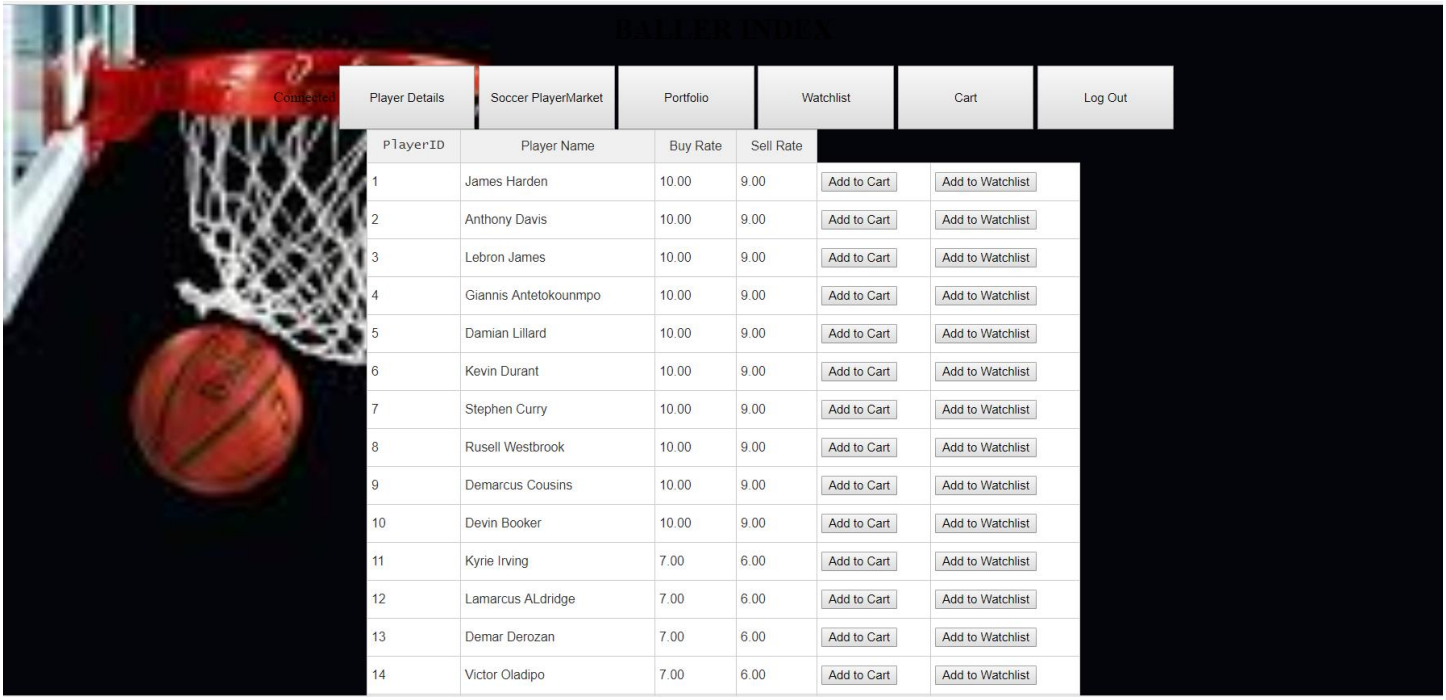

```
delimiter $$
create procedure `SPnewpriceF` (in iplayerID int(10))
begin
declare a integer(2); declare vgls decimal(5,2);
declare vast decimal(5,2);declare vsav decimal(5,2);
declare vint decimal(5,2);declare vclr decimal(5,2);
declare vblk decimal(5,2); declare vgc decimal(5,2);
declare vyc decimal(5,2); declare vrc decimal(5,2);
declare vpm decimal(5,2); declare vog decimal(5,2);
declare vmp decimal(5,2); declare vppe decimal(5,2);
declare vcs decimal(5,2); declare vgls1 decimal(5,2);
declare vast1 decimal(5,2); declare vsav1 decimal(5,2);
declare vint1 decimal(5,2); declare vclr1 decimal(5,2);
declare vblk1 decimal(5,2); declare vgc1 decimal(5,2);
declare vyc1 decimal(5,2); declare vrc1 decimal(5,2);
declare vpm1 decimal(5,2); declare vog1 decimal(5,2);
declare vmp1 decimal(5,2); declare vppe1 decimal(5,2);
declare vcs1 decimal(5,2); declare vpri decimal(5,2);
declare vsec decimal(5,2); declare vper decimal(5,2);
declare vneg decimal(5,2); declare vbon decimal(5,2);
declare vpri1 decimal(5,2); declare vsec1 decimal(5,2);
declare vper1 decimal(5,2); declare vneg1 decimal(5,2);
```

```
declare vbon1 decimal (5,2); declare changeinprice decimal(5,2);
declare currentprice decimal(5,2); declare newsellprice decimal(5,2);
```

Multiples Variables are declared to store the retrieved stats for current game and season averages. The table 'StatsType' is used to retrieve the stats, if the StatsTypeID= 1 it refers to current game stats and if the StatsTypeID=2 then it refers to season averages. Variables have been declared to store the calculations of each category of stats such as primary, secondary, negative, percentage, bonus.

```
set a=(select playerMarket.GametypeID from playermarket where playerID=iPlayerID);
if a=1 then /*GametypeID = 1 for soccer & 2 for basketball*/
set vgls = (select fstats.goals from fstats where playerID=iplayerID and statstypeid = 1);
set vast = (select fstats.assists from fstats where playerID=iplayerID and statstypeid = 1);
set vsav = (select fstats.saves from fstats where playerID=iplayerID and statstypeid = 1);
set vint = (select fstats.Interceptions from fstats where playerID=iplayerID and statstypeid = 1);
set vclr = (select fstats.Clearances from fstats where playerID=iplayerID and statstypeid = 1);
set vblk = (select fstats.Blocks from fstats where playerID=iplayerID and statstypeid = 1);
set vgc = (select fstats.GoalsConceded from fstats where playerID=iplayerID and statstypeid =
1);
set vyc = (select fstats.YellowCard from fstats where playerID=iplayerID and statstypeid = 1);
set vrc = (select fstats.Redcard from fstats where playerID=iplayerID and statstypeid = 1);
set vpm = (select fstats.PenaltyMissed from fstats where playerID=iplayerID and statstypeid =
1);
set vog = (select fstats.OwnGoal from fstats where playerID=iplayerID and statstypeid = 1);
set vmp = (select fstats.`Minutesplayed/90` from fstats where playerID=iplayerID and
statstypeid = 1);
set vppe = (select fstats.PassPercentage from fstats where playerID=iplayerID and statstypeid =
1);
set vcs = (select fstats.CleanSheet from fstats where playerID=iplayerID and statstypeid = 1);
set vgls1 = (select fstats.goals from fstats where playerID=iplayerID and statstypeid = 2);
set vast1 = (select fstats.assists from fstats where playerID=iplayerID and statstypeid = 2);
set vsav1 = (select fstats.saves from fstats where playerID=iplayerID and statstypeid = 2);
set vint1 = (select fstats.Interceptions from fstats where playerID=iplayerID and statstypeid =
2);
set vclr1 = (select fstats.Clearances from fstats where playerID=iplayerID and statstypeid = 2);
set vblk1 = (select fstats.Blocks from fstats where playerID=iplayerID and statstypeid = 2);
set vgc1 = (select fstats.GoalsConceded from fstats where playerID=iplayerID and statstypeid =
2);
set vyc1 = (select fstats.YellowCard from fstats where playerID=iplayerID and statstypeid = 2);
set vrc1 = (select fstats.Redcard from fstats where playerID=iplayerID and statstypeid = 2);
```

```
set vpm1 = (select fstats.PenaltyMissed from fstats where playerID=iplayerID and statstypeid =
2);
set vog1 = (select fstats.OwnGoal from fstats where playerID=iplayerID and statstypeid = 2);
set vmp1 = (select fstats.`Minutesplayed/90` from fstats where playerID=iplayerID and
statstypeid = 2);
set vppe1 = (select fstats.PassPercentage from fstats where playerID=iplayerID and statstypeid
= 2);
set vcs1 = (select fstats.CleanSheet from fstats where playerID=iplayerID and statstypeid = 2);
set vpri = (vgls+vast); /* The price algorithm begins here */
set vsec = (vsav+vint+vclr+vblk+vgc);
set vper = (vmp+vppe);
set vneg = (vyc+vrc+vpm+vog);
set vbon = (vcs);
set vpri1 = (vgls1+vast1);
set vsec1 = (vsav1+vint1+vclr1+vblk1+vgc1);
set vper1 = (vmp1+vppe1);
set vneg1 = (vyc1+vrc1+vpm1+vog1);
set vbon1 = (vcs1);
set vpri = (vpri-vpri1)*0.5;
set vsec = (vsec-vsec1)*0.3;
set vper = (vper-vper1)/100;
set vneg = (vneg-vneg1)*0.2;
set vbon = (vbon-vbon1)*0.1;
set changeinprice = (vpri+vsec+vper+vneg);
set currentprice = (select price.buyrate from price where playerID=iplayerID);
set currentprice = (currentprice+changeinprice); /*The price algorithm ends here */
select changeinprice;
update price set buyrate=currentprice where playerID=iplayerID; /* updates price table */
set newsellprice = (select price.SellRate from price where playerID=iplayerID);
set newsellprice = (newsellprice+changeinprice);
update price set sellrate=newsellprice where playerID=iplayerID; /* sell rate has been updated*/
select currentprice;
/* the change in price and current price is given as an output after the store procedure is
completed */
end if;
end$$
delimiter ;
/* please refer to the video in the folder to get a demo of the working model of the dynamic
pricing of shares */
```



PlayerID	Player Name	Buy Rate	Sell Rate		
1	James Harden	10.00	9.00	Add to Cart	Add to Watchlist
2	Anthony Davis	10.00	9.00	Add to Cart	Add to Watchlist
3	Lebron James	10.00	9.00	Add to Cart	Add to Watchlist
4	Giannis Antetokounmpo	10.00	9.00	Add to Cart	Add to Watchlist
5	Damian Lillard	10.00	9.00	Add to Cart	Add to Watchlist
6	Kevin Durant	10.00	9.00	Add to Cart	Add to Watchlist
7	Stephen Curry	10.00	9.00	Add to Cart	Add to Watchlist
8	Rusell Westbrook	10.00	9.00	Add to Cart	Add to Watchlist
9	Demarcus Cousins	10.00	9.00	Add to Cart	Add to Watchlist
10	Devin Booker	10.00	9.00	Add to Cart	Add to Watchlist
11	Kyrie Irving	7.00	6.00	Add to Cart	Add to Watchlist
12	Lamarcus ALdridge	7.00	6.00	Add to Cart	Add to Watchlist
13	Demar Derozan	7.00	6.00	Add to Cart	Add to Watchlist
14	Victor Oladipo	7.00	6.00	Add to Cart	Add to Watchlist

spADDWatchlist

spAddWatchlist enables the customer of Baller Index to add many players to his watchlist. A watchlist is a tool which helps each user to monitor individual athlete's stocks with a potential trading option. If used accurately it could give the customer an upper hand when the desired price of the stock is reached. This procedure requires us to input these following parameters to call the procedure playerID, "add or remove", UserID.

When the customer clicks on the button `Add to watchlist` that player is added to his watchlist. This Procedure is called on clicking the `Add to watchlist` button.

```
delimiter $$
create procedure spAddWatchlist (in iplayerid integer(5), iaddremove varchar(10), iuserid
integer(5))
begin
declare addremove varchar(10);
declare currentprice decimal(5);
declare watchlistidv integer(5);
```

```
declare p integer(5);
set watchlistidv = (select watchlistid from watchlist where watchlist.UserID = iuserid);
set p = (select priceID from price where price.PlayerID=iplayerID);
If  iaddremove = 'Add' or 'add' or 'ADD' then
insert into watchlistitems
values (null,watchlistidv, iplayerid, iuserid, p); /* The procedure updates the watchlist */
elseif iAddRemove = 'Remove' or 'remove' or 'REMOVE' then
delete from watchlistitems where watchlistitems.PlayerID = iplayerid and
watchlistitems.UserID = iuserid;

end if;
end$$
 delimiter ;
```

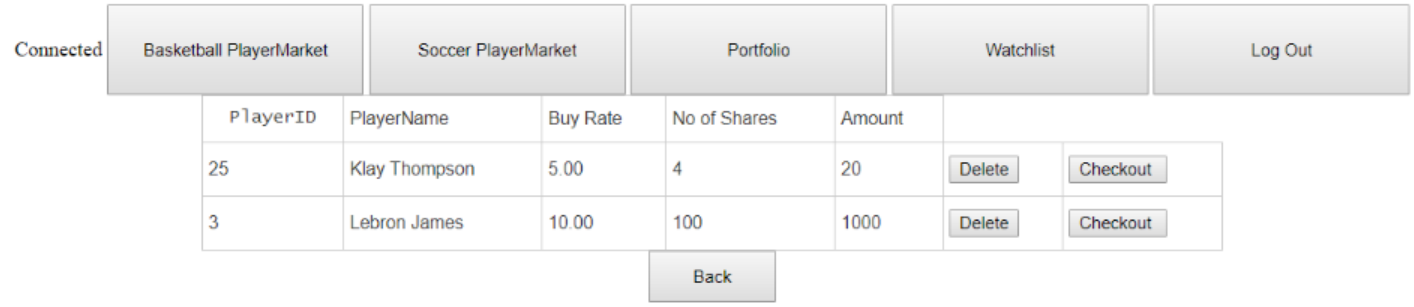
spAddtocart

The spAddtocart procedure is used for buying shares. The cart feature is used as a pre-confirmation which is beneficial for the customer. Customer can add individual player either from watchlist or directly from the player market. This procedure can only be executed when the input parameters (userID, PlayerID, no. of shares) are given.


```
delimiter @@
create procedure spAddtocart(`iUserID` integer(5),`iPlayerID` integer(5),`inoofshares`
integer(5))
Begin
declare cid integer(5);
declare pric integer(5);
set cid=(select CartID from Cart where cart.UserID=iUserID);
set pric= (select price.BuyRate from price where PriceID=(
select PriceID from price where price.playerID=iplayerID));
insert into CartItems values(null,cid,iUserID,iPlayerID,inoofshares,pric*inoofshares,null);
end@@    /* cartitems table is updated */

delimiter ;
```

User Interface for add to cart, this is a display of the cart items table.

- The user has an option of ‘checkout’ which means placing an order for the shares.
- The user has an option to ‘delete’ , which deletes the stock item from the cart.



Customer_details

`Customer_details` view- joins Customer table, Address table and State table. The key UserID is used to connect Customer and Address tables. Zip is used to connect Address and State tables. This view displays all customer details that have been stored in multiple tables.

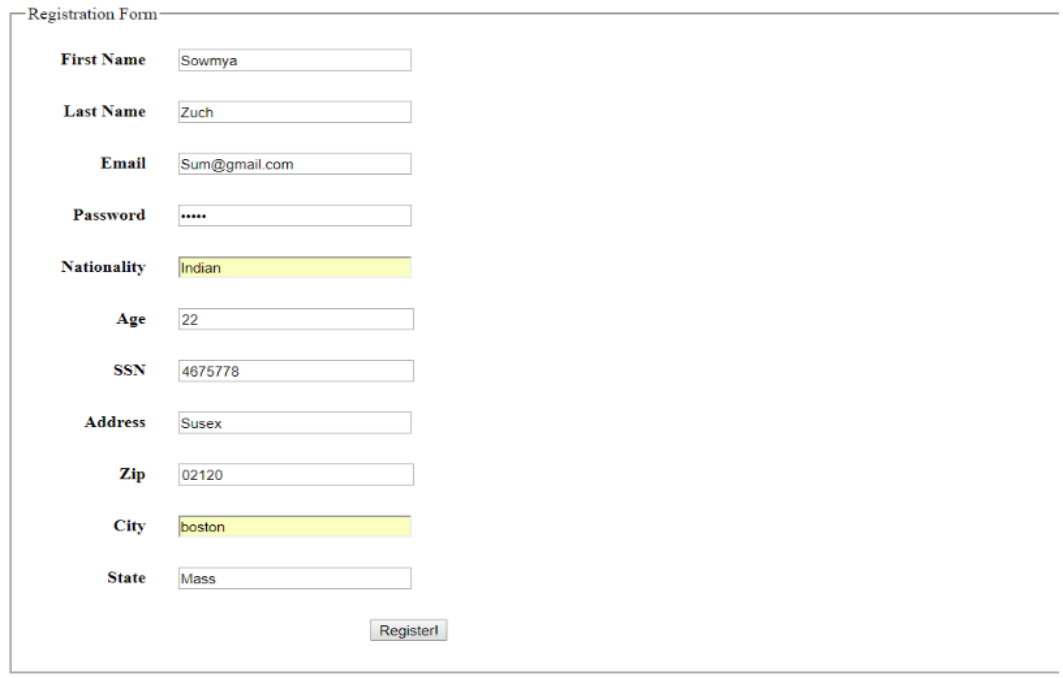

```
Create VIEW ballerindex1.`CUSTOMER_DETAILS` as
select
customer.UserID,
customer.`First Name`,
customer.`Last Name`,
customer.Nationality,
customer.age,
Address.Address,
state.zip,
state.state
from ballerindex1.Customer inner join ballerindex1.address
on customer.UserID=address.UserID
inner join ballerindex1.state on state.zip=address.Zip; /* we used inner join here to connect state and address table with the help of the common column  ZIP , the reason we used an inner join is cause it's the most common kind of join , and this procedure returns only one column, theoretically any join could be used for developing this view. Since the User cannot create an account without entering all these information, there won't be any null values in any of the columns. So inner join or full outer join will work.  */
```

```
DELIMITER $$
create procedure ballerindex1.spviewcustomer(in ipuser integer(5))
BEGIN
select * from ballerindex1.`customer_details` where `customer_details`.UserID=ipuser;
end$$
Delimiter;
```

```
315 •  select*from customer_details;
```

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

UserID	First Name	Last Name	Nationality	age	Address	zip	state
2	Jorie	Leile	Ukraine	38	778 Orin Circle	97240	Oregon
3	Danielle	Sutherden	Philippines	46	375 Johnson Parkway	52809	Iowa
4	Christiano	Liversage	Russia	41	44 Fair Oaks Terrace	50335	Iowa
5	Nelly	Reace	China	51	43 Little Fleur Road	44329	Ohio
6	Sean	Molian	Kazakhstan	33	120 Sullivan Terrace	29605	South Carolina
7	Cleavland	Gooderson	Finland	32	364 Utah Avenue	37245	Tennessee
8	Madison	Bernet	Philippines	31	0 Westerfield Alley	49518	Michigan
9	Charlotta	Presslie	Ukraine	64	31775 Eagle Crest Pass	80045	Colorado
10	Averil	Brasted	Mexico	46	3 Carey Avenue	60624	Illinois



SpSignUp

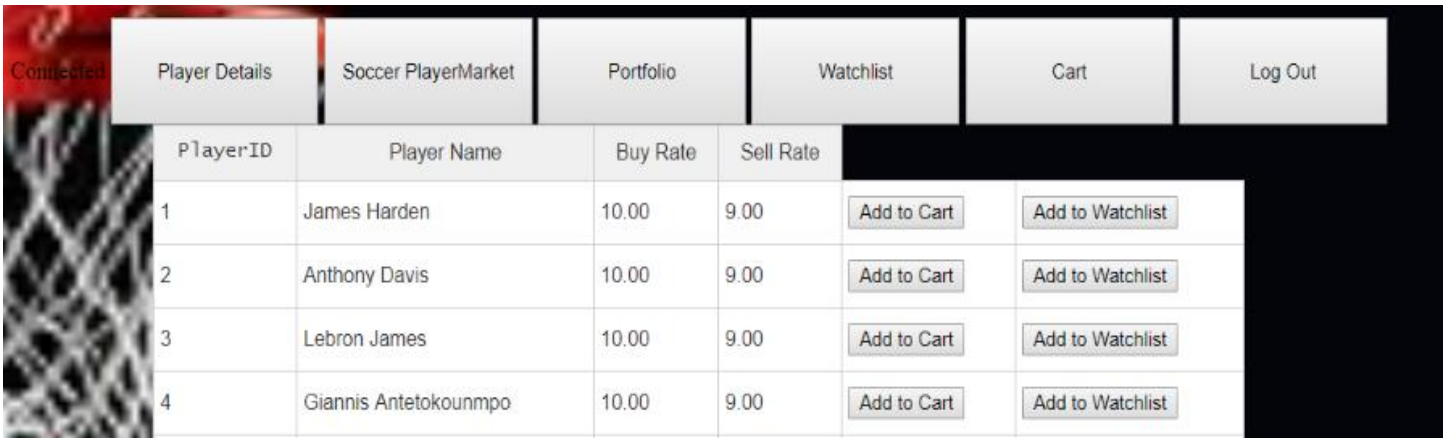
Customer enters his personal details from the User Interface. All these inputs are passed as parameters to the stores procedure SpSignUp.

Here a stored procedure is used because the entries are inserted into multiple tables, connected by foreign keys. Since multiple insertion commands are used, they are combined inside a single stored procedure.

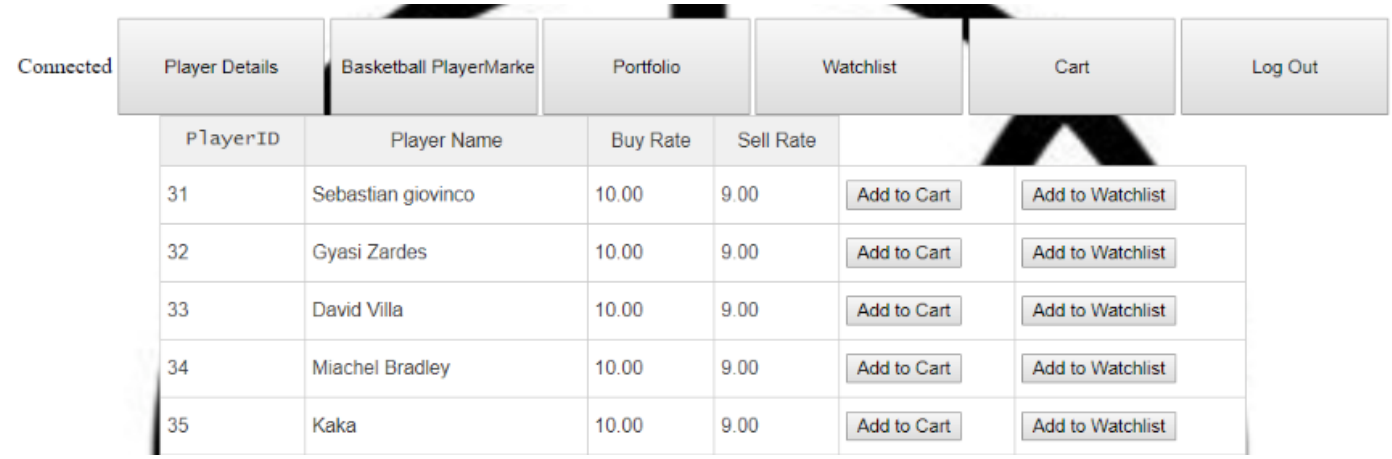
```
delimiter $$

CREATE procedure SpSignUp(
 in`iemail` varchar(100),
 `ipassword` varchar(15),
 `iname on card` varchar(20),
`icard no` integer(16),
`iexpiry date` date,
`iaddress` varchar(20),
`izip` integer(7),
`ifirst name` varchar(20),
`ilast name` varchar(20),
`inationality` varchar(20),
`iage` integer(5),
`iSSN` integer(15)) ))   /* All these parameters are entered from the User Interface in  the Sign Up form.They are passed as input parameters to the Stored procedure.*/
```

```
BEGIN
DECLARE msg VARCHAR(25);
declare data1 integer;  /* If the entered email id exists in the logininfo table, new sign up doesn't occur and message "User Already Exists is shown"  */


IF exists( select email from logininfo where logininfo.email = iemail)
   then set msg = 'USER ALREADY EXISTS';
   select msg;
else
   insert into `customer` values (null,`iFirst Name`, `iLast Name`, `inationality`, `iage`,`issn`);
     /* Since the UserID is an auto increment field null value is passed. Every time a new entry is inserted in the row, the UserID is auto incremented starting from 1. */


 set data1=last_insert_id();
  /* This statement is used to get the primary key value of the last row inserted.  This ID is used as foreign key in other tables Bank, Address and logininfo.

   insert into `address`values (data1,`iaddress`, `izip`);
   INSERT into `bank`values (null,data1, `iname on card`, `icard no`, `iexpiry date`);
  /* Since the bankid is an auto increment field,  null  value is passed. Every time a new entry is    inserted in the row, the Bank ID is auto incremented. Data1 has the value of the UserID of the corresponding row in Customer table*/

   Insert into `logininfo`values (data1,`iemail`, `ipassword`);
   insert into `portfolio` values(null,data1);
   insert into `cart` values(null,data1);
   insert into `watchlist` values(null,data1);
   insert into `wallet` values(null,data1,0);
End if;
End $$
Delimiter ;
```

BasketBall Player Market



Soccer Player Market



---

spViewPlayerMarket

This procedure is used for displaying the details and Prices for all the players in the player market. Since there are football and Basketball player markets, GametypeID is passed as input parameter depending on the selection made by the User from the Baller Index User Interface. If User clicks Basketball Player market, Gametype ID 1 is passed and for Football player market gametype ID 2 is passed.

```
Delimiter $$
create procedure spViewPlayerMarket (in igametypeID nvarchar(20))
begin
select
PlayerMarket.PlayerName,
PlayerMarket.Height,
PlayerMarket.Weight,
PlayerMarket.Age,
```

```
Price.BuyRate,
Price.SellRate
from Ballerindex1.PlayerMarket
inner join Ballerindex1.Price
on PlayerMarket.PlayerID=Price.PlayerID
inner join Ballerindex1.gametype on gametype.GameTypeid=playermarket.GameTypeID
where playermarket.GameTypeID=igametypeID;
end$$
delimiter ;
```

spPlayerName

This procedure is used to display the player details depending on the player name entered by the User from the User Interface. The procedure joins Player market, Price and Stats tables. The input parameters for this procedure is Player name.

```
delimiter $$
Create PROCEDURE ballerindex1.spPlayerName(in ipPlayerName varchar(20))
BEGIN
declare a varchar(50);
set  a = (select gametype from playermarket inner join
 gametype on GameType.GameTypeid=playermarket.GameTypeID where
playername=ipPlayername);
```


/*Here if else condition is used based on the GameType (Basketball or Football) because the stats list is different for the football and basketball players*/

```
 If a = 'basketball' then
 select
playermarket.playername,
playermarket.height,
playermarket.weight,
playermarket.age,
gametype.gametype,
bstats.points,
bstats.rebounds,
bstats.assists,
bstats.steals,
bstats.blocks,
bstats.`3pointersmade`,
bstats.`FG%`,
bstats.`3PFG%`,
bstats.turnovers,
```

```
bstats.personalfouls
from playermarket inner join gametype on
playermarket.GameTypeID=gametype.GameTypeID
inner join bstats on playermarket.playerID=bstats.playerID
where playermarket.playername = ipPlayerName;
else
select
playermarket.playername,
playermarket.height,
playermarket.weight,
playermarket.age,
gametype.gametype,
fstats.`goals`,
fstats.`assists`,
fstats.`saves`,
fsats.`interceptions`,
fstats.`clearances`,
fstats.`blocks`,
fstats.`goalsconceded`,
fstats.`yellowcard`,
fstats.`redcard`,
fstats.`penaltymissed`,
fstats.`owngoal`,
fstats.`minutesplayed/90`,
fstats.`passpercentage`,
fstats.`cleansheet`
from playermarket inner join gametype on
playermarket.GameTypeID=gametype.GameTypeID
inner join fstats on playermarket.playerID=fstats.playerID
where playermarket.playername = ipPlayerName;
 end if;
 end$$
```



```
4 •  call spPlayerName('James harden');
```

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

playername	height	weight	age	gametype	points	rebounds	assists	steals	blocks	3pointersmade	FG%	3PFG%	turnovers	personalfouls
James Harden	197.00	100.00	28	basketball	35.00	6.00	10.00	3.00	1.00	4.00	50.00	35.00	2.00	2.00
James Harden	197.00	100.00	28	basketball	29.00	8.00	11.20	1.50	0.50	3.20	44.00	34.00	5.70	2.70

spAddMoney

This stored procedure is called when the user wants to add or remove money from wallet. Using if else condition money is added or removed from the wallet. Here we used a stored procedure to combine all the if else queries and execute them on the procedure call. If the balance goes to a negative amount, deduction doesn't take place. Instead user gets a message.

```
delimiter $$
create procedure `spAddMoney` (in ipuserid integer(5), ipAddRemove varchar(6), ipAmount integer(5))
/* If the user wants to add money to wallet `Add` is passed as 'ipAddRemove ' and  to remove money 'remove' is passed. */
BEGIN
declare a integer(50);
declare msg varchar(50);
declare bal decimal(7,2);
set  a = (select walletID from wallet where userid = ipuserid);
If  ipAddRemove = 'Add' or 'add' or 'ADD' then
update wallet set wallet.balance = wallet.balance + ipAmount where wallet.UserID = ipuserid;
elseif ipAddRemove = 'Remove' or 'remove' or 'REMOVE' then
set bal=(select wallet.balance from wallet where wallet.UserID=ipUserID);
```



```
  if (bal-ipAmount<0) / *If  statement checks if the balance goes negative after money is removed from wallet.*/
  then
  set msg="Negative balance";
  select msg;
  else
  update wallet set wallet.balance = wallet.balance - ipAmount where userid = ipuserid;
end if;
else
set msg="Amount added";
select msg;
end if;
end$$
Delimiter ;
```

1# checking wallet balance

```
317  select*from wallet where userid=1;
```

Result Grid | Filter Rows: | Edit:

WalletID	UserID	Balance
1	1	545
NULL	NULL	NULL

#2 Adding money to wallet and checking balance.

```
319  call  spAddMoney(1,'add',500);
320  select*from wallet where userid=1;
```

Result Grid | Filter Rows: | Edit:

WalletID	UserID	Balance
1	1	1045
NULL	NULL	NULL

#3 Removing money and checking balance.

```
319  call  spAddMoney(1,'remove',1000);
320  select*from wallet where userid=1;
```


Result Grid | Filter Rows: | Edit:

WalletID	UserID	Balance
1	1	45
NULL	NULL	NULL

SPupdateorder

After the User Adds items to the cart and proceeds to Check out, Order is made and Order table is updated. Each order entry is connected to one or more items in CartItems table. The procedure consists of calculating the total amount of all the shares in the cart, updating the Wallet and inserting a new row into the Orders table with details of the Order. After the insertion is done , trigger 'triggerAddtoportfolio' is automatically called which updates the Portfolio of the Customer.

```
delimiter $$
create procedure SPupdateorder(`iUserID` integer(5)) /* only User ID is passed as input parameter. When the user clicks check out, all the items in his cart are added to the portfolio*/

begin
declare c integer(10);
```

```
declare a decimal(7,2);
declare w integer(10);
declare b decimal(7,2);
set c=(select distinct(CartID) from cartitems where cartitems.UserID=iUserID);
 /*CartID for the user is retrieved.  */

set a=(select sum(Amount) from  cartitems where cartitems.UserID=iUserID);
  / * To calculate the total price of all the  shares added in the cart  */

set w=(select WalletID from wallet where wallet.UserID=iUserID);
insert into orders values(null,c,iUserID,a,w);
update cartitems set orderplaced = 'yes' where cartitems.UserID=iUserID;
set b=(select balance from wallet where wallet.UserID=iUserID);
set a= (b-a);
update wallet set balance=a where wallet.UserID=UserID;
 /* Amount is deducted from the wallet*/

delimiter ;
```

/*please refer to the demo video for a better understanding of the stored procedure*/

BALLER INDEX

Connected | Basketball PlayerMarket | Soccer PlayerMarket | Portfolio | Watchlist | Cart | Log Out

PlayerID	Player Name	sell Rate	No of shares	TotalStockPrice	
1	James Harden	9.00	82	738.000	Sell
5	Damian Lillard	9.00	76	684.000	Sell
6	Kevin Durant	9.00	56	504.000	Sell
2	Anthony Davis	9.00	10	90.000	Sell
4	Giannis Antetokounmpo	9.00	100	900.000	Sell
13	Demar Derozan	6.00	67	402.000	Sell
32	Gyasi Zardes	9.00	17	153.000	Sell

Spselling

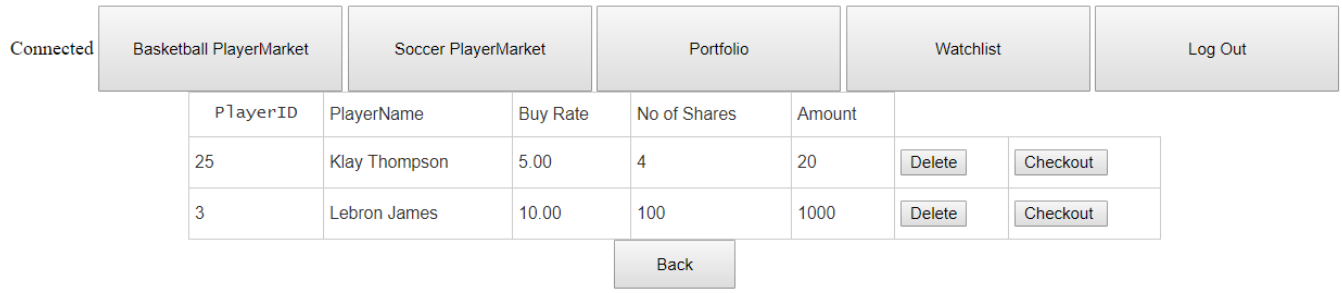
The stored procedure takes userid of the customer, number of shares and Player that the user wishes to sell as the input parameters. Once a customer decides to sell his shares multiples tables (Portfolio,Wallet,Selling) in the database have to be updated.
To make the process easier Spselling is called on clicking the button `Sell`.

```
 DELIMITER $$
CREATE PROCEDURE `SpSelling`(`iuserid`integer (5), `iplayerid` integer (5), `inoofshares`
integer (5))
    BEGIN
    declare profitloss decimal (7,3);
    declare uid integer(5);
    declare pid integer (5);
    declare nos integer(5);
    declare avgrate decimal(7,3);
    declare sllrate decimal(7,3);
    declare diff decimal (7,3);
    declare wallID integer(5);
    declare samount decimal(7,3);
    declare tprice decimal(7,3);
    declare msg nvarchar(50);
    set nos = (select `No of shares`-inoofshares from portfolioitems
    where portfolioitems.UserID=iUserID and portfolioitems.playerID=iplayerID);
    if (nos>0)
 /* The number of shares owned by the customer is compared with the number of shares he
wishes to sell. If he enters a larger number, transaction will not proceed.
```



```
 then set avgrate= (select AvgBuyRate from portfolioitems
     where  portfolioitems.UserID=iUserID and portfolioitems.playerID=iplayerID);
     /*  AvgBuyRate stores the average of all the buy rates at which the shares were bought.*/

   set wallID=(select walletID from wallet where userID=iuserID);
     set sllrate = (select SellRate from price where playerid= iplayerid);
     set diff= inoofshares*(sllrate-avgrate);
 /*variable diff  stores the profit he makes from the transaction. */

     set samount=sllrate*inoofshares;
     set tprice=nos*sllrate;
/* tprice stores the total price of the stocks the customer holds after the selling*/

     insert into selling values(null,iuserID,iplayerID,inoofshares,diff,wallID);
     update wallet set balance = `balance`+ samount where userid= iuserid;
     update portfolioitems set `no of shares` = nos ,totalstockprice=tprice
     where playerid= iplayerid and userID=iUserID;
 else
    set msg="Cannot proceed the transaction select msg” ;   END if;
    END$$
```

DELIMITER ;



spAddtoPortfolio

When an order is made, the order table is updated and next step is Portfolio updating. First a check is made if the added player exists in the portfolio. If 'yes' the number of shares is updated, else a new line is added to the portfolio. This stored procedure is created for combining all the update and insert operations in Portfolio, retrieving the price from the price table.
On clicking the checkout the spAddtoPortfolio is called.


```
delimiter $$
create procedure spAddtoPortfolio(`iUserID` integer(5),`iPlayerID` integer(5),`inoofshares`
integer(5))
begin
declare msg varchar(20);
declare pid integer(5);
declare pric decimal(5,2);
declare pricid integer(5);
declare tprice decimal(7,3);
declare newnoofshares integer(5);
declare ibuyrate decimal(7,3);
declare spric decimal(7,3);
set pid=(select portfolioID from portfolio where portfolio.UserID=iUserID);
set pric= (select price.BuyRate from price where PriceID=(
select PriceID from price where price.playerID=iplayerID));
set pricid=(select PriceID from price where price.playerID=iplayerID);

set spric= (select price.SellRate from price where PriceID=(
select PriceID from price where price.playerID=iplayerID));
```

```
if
exists(select portfolioItems.playerID from portfolioitems where
portfolioItems.playerID=iplayerID and userID=iUserID)
 /* to check if the newly added player exists in the portfolio

then
set newnoofshares=(select `no of shares`+inoofshares from portfolioitems
where  PlayerID=iplayerID and UserID=iUserID);
set tprice=spric*newnoofshares;
set ibuyrate=(select AvgBuyRate from portfolioitems where  PlayerID=iplayerID and
UserID=iUserID);
set ibuyrate=(ibuyrate+pric)/2;
 /* ibuyrate stores the previous buy rate of the player initially. After another purchase is made
the variable is updated with the average of
present and previous  buy rate.*/

update portfolioitems set `No of shares`=`newnoofshares`,`totalstockprice`=
tprice,`AvgBuyRate`=ibuyrate
 where  PlayerID=iplayerID and UserID=iUserID; /*newly bought shares are added */
else
```

```
set newnoofshares=`inoofshares`;
set tprice= spric*newnoofshares;
/* Updating the total stock price after adding new items to the portfolio*/

insert into portfolioitems values(null,pid,iplayerID,newnoofshares,pric,tprice,pricid,`iUserID`);

end if;
end$$
delimiter ;


delimiter @@
create trigger trigger_selling
after insert on selling
for each row
begin
declare sid integer(5);
declare uid integer(5);
set sid=(select max(sellingorderId) from selling);
set uid=(select UserID from selling where sellingorderID=sid);
```

```
delete from portfolioitems where  UserID=uid and portfolioitems.`No of shares`=0;
end@@    delimiter ;
```

triggeraddportfolio

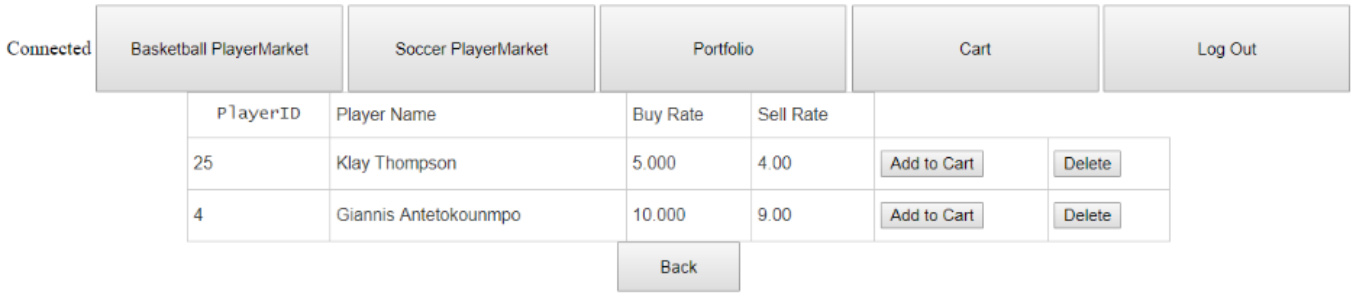
The trigger automatically executes the stored procedure spAddtoPortfolio when a new row is inserted in Orders table. ie after customer makes a check out from the cart, the purchased stocks are automatically updated in the portfolio.

```
delimiter !!
create trigger triggeraddportfolio
after insert on orders
for each row
begin
declare cid integer(5);
declare uid integer(5);
declare pid integer(5);
declare nos integer(5);
declare oid integer(5);
declare notfound integer default 0;
DECLARE cur1 CURSOR FOR SELECT cartitemsID FROM cartItems WHERE userID=
(select UserID from orders  where OrderID=max(OrderID));     /*  max(OrderID) is used to get
```


the last inserted OrderID in the orders table . Here a cursor is declared to point to each row returned from the CartItems table corresponding to a particular user.

```
 */

DECLARE CONTINUE HANDLER
      FOR NOT FOUND SET notfound = 0/* After all the CartItemIDs of a particular User are
returned, notfound  is set  to 1.
open cur1;                  /* for each row corresponding to the value in cur1(CartItems ID ), below
operations are done.*/
get_details: loop
fetch cur1 into cid;
set uid=(select UserID from cartitems where cartitemsID=cid);
set pid=(select playerID from cartitems where cartitemsID=cid);
set nos=(select noofshares from cartitems where cartitemsID=cid);
call spaddtoportfolio(uid,pid,nos);
if notfound=1
then
leave get_details;
end if;
end loop get_details;
close cur1;
```

```
end !!
delimiter ;
```



spViewWatchlist

This a stored procedure for viewing the players stored in an individual watchlist. spViewWatchlist requires an input of the unique userid(iuserid) provided to the customer just after the signing up.

```
delimiter $$
create procedure spViewWatchlist (iuserid integer(5))
begin
SELECT
PlayerMarket.PlayerID,PlayerMarket.PlayerName,watchlistitems.Currentprice,price.`Sell rate`
from PlayerMarket inner join Price on PlayerMarket.PlayerID=Price.PriceID
INNER JOIN watchlistitems ON watchlistitems.PriceID=price.PriceID Where
watchlistitems.UserID=iuserid;
end$$
delimiter ;
```

spviewcart

This is a stored procedure to view cart, this procedure is used to display in the UI. The cart is primarily a tool used for buying shares from the player market. The input parameter for this store procedure is userID.

```
delimiter %%
create procedure spviewcart(in `iuserID` isnteger(5))
begin
select cartItems.PlayerID,PlayerMarket.Playername,Price.`Buy
Rate`,CartItems.noofshares,CartItems.Amount from CartItems inner join PlayerMarket on
CartItems.PlayerID=PlayerMarket.PlayerID inner join Price on
PlayerMarket.PlayerId=Price.PlayerID where cartitems.userID=iuserID and
cartitems.OrderPlaced='no';
end%%
delimiter ;
```

Errors faced while creating store procedures

1107 incorrect parameter count to procedure: This error arises when incorrect number parameters are defined or passed in the call statement. To rectify this error, we need to make sure we are passing the correct number of parameters.

We faced this error in procedures for signup, addtocart etc . Sign up procedure takes so many input parameters. When the order or parameter count goes wrong this error appears. We rectified these errors by calling the procedures by giving correct input parameters

.

1452 cannot add or update a child row, foreign key constraint: This error does not allow the user to insert or update data into the columns of the tables which are under a foreign key constraint. The user must make sure the data he is inserting or updating is contained in the table where it is a primary key.

For procedures like sign up, we faced this error since the zip code entered was not there in State table. To rectify this we made sure every foreign key inserted is present in the other table.

1193 Unknown system variable: This error arises when we do not declare the system variable, but we use it in the following code lines.

We faced this error in almost all the stored procedure especially price calculation were we used temporary variables. This was rectified when all the variables are declared correctly with correct data type.

1406 Data too long for column: This error arises when the range of the defined variable is too short for the data inserted. This error can be rectified by increasing the range of the defined variable.
When procedures like spSelling, SpAddToPortfolio, spSelling etc are called, the totalPrice was too large to be stored in the database. So we increased the attribute size from decimal (5,2) to decimal(7,3).

1242 Sub query returns more than one row: This error arises when the sub query returns more than one value. This error can be prevented by carefully analyzing the sub query and making sure that we are not returning more than one value.
We faced this error in trigger for updating portfolio since there are multiple cart items for same user. We used distinct (UserID) to return single row from cartitems.

1052 column and where clause is ambiguous: This is an error that we faced when the server cannot identify the table we wont to query. To fix this error we simply add the table name of the table you want to work with
We faced this error when we created views or procedures that uses many joins. This error was rectified by adding table name before the column name
Eg playermarket.PlayerID

USERS AND PRIVILEGES

```
create user investor identified by 'baller'; /* syntax for creating a new user */
revoke all privileges, grant option from investor@localhost;

create user systemadmin identified by 'qwerty';
revoke all privileges, grant option from systemadmin@localhost;
grant select, insert, update on ballerindex1.* to systemadmin; /* granting privileges to user */

create user Mainadmin@localhost identified by 'ballerindex';
revoke all privileges, grant option from systemadmin@localhost;
grant all on *.* to Mainadmin@localhost with grant option;


create user Analyst identified by 'ballerindex';
revoke all privileges, grant option from Analyst@localhost;
grant select on ballerindex1.* to Analyst with grant option;
show grants for Mainadmin@localhost;
```

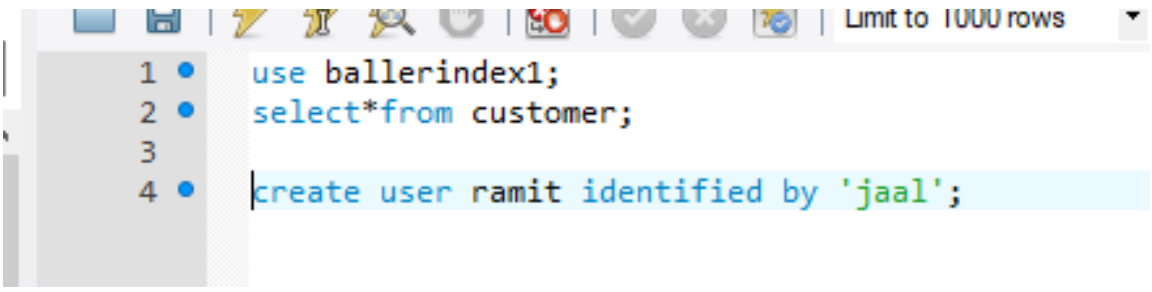
The above SQL codes we make 3 different users

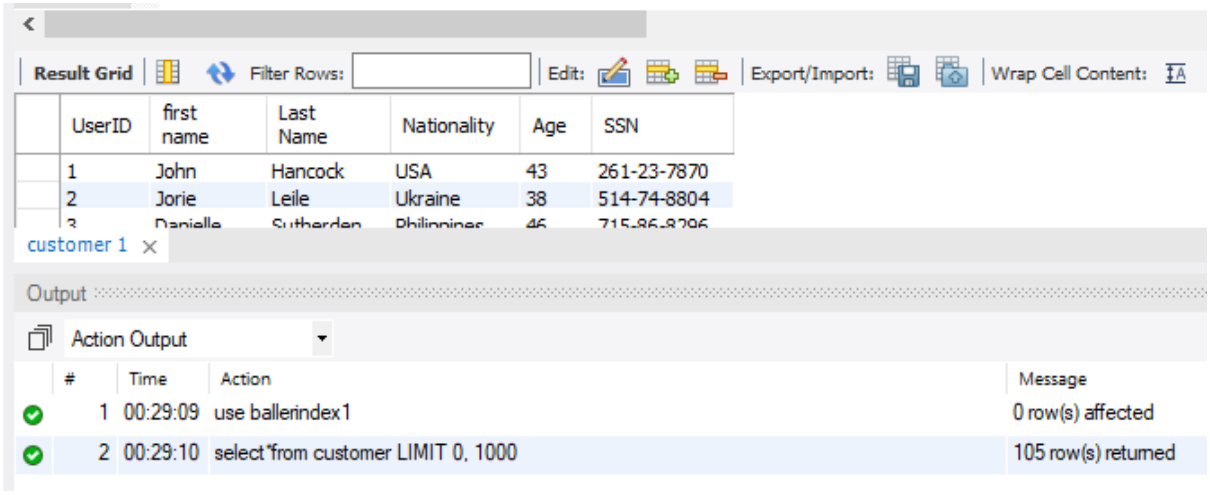
- Investor
- systemadmin
- Analyst

Investor is a customer in this context, and she is not allowed any privileges. The customer is not allowed to view the database as it is confidential and can only be viewed by members of the team.

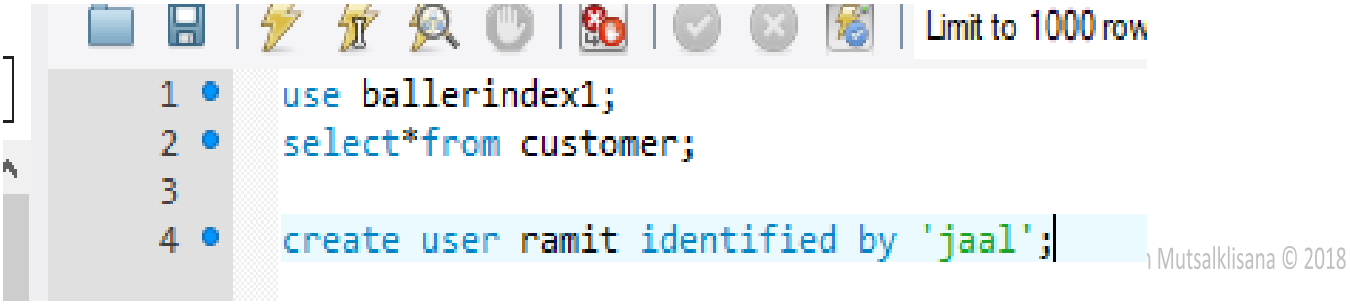
#	Time	Action	Message
1	20:11:35	use ballerindex1	Error Code: 1044. Access denied for user 'investor'@'%' to database 'ballerindex1'

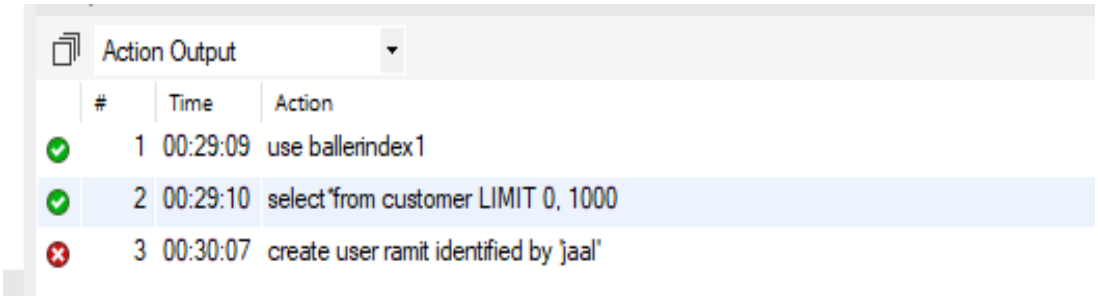
Systemadmin user is given a few privileges such as inserting data and selecting data to view tables and updating the tables and stats.



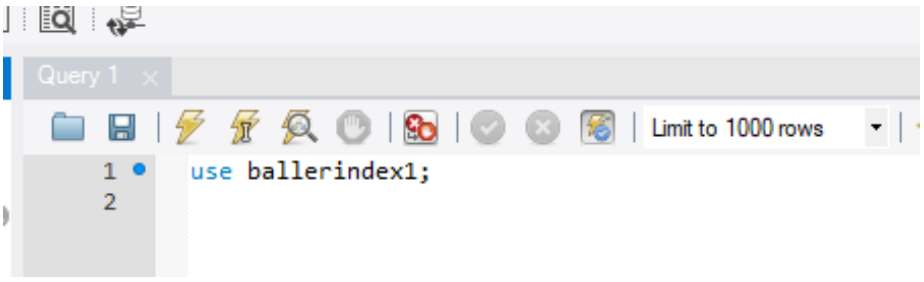


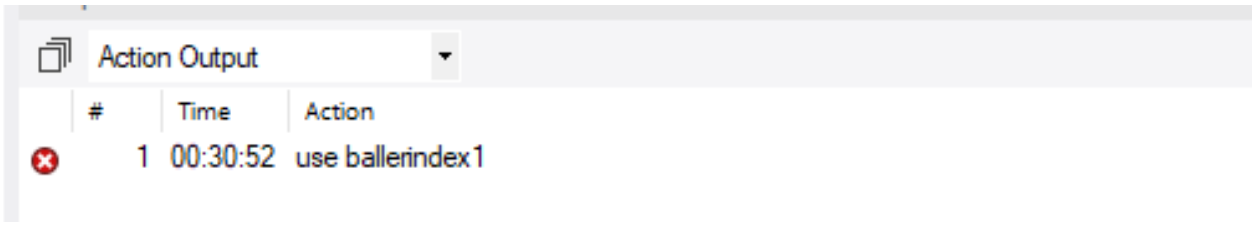
The system admin cannot create a new user.





Analyst is granted privileges to only select and view tables from the database.





BACKUP

- Number of hours full backup = 5
- Number of hours incremental backup = 1
- Number of hours full incremental backup = 3

Peak hours

during weekdays (3PM to 2AM)

During weekends (1PM to 2AM)

Backup Plan

Incremental backup – Monday, Tuesday, Wednesday, Friday (6-7AM)

Full Incremental backup – Thursday (6-9AM)

Full Backup – Monday(6-11********AM)

Monday ---- Tuesday ---- Wednesday ---- Thursday ---- Friday ---- Saturday ---- Sunday

Full Backup (6-11AM)	Incremental Backup (6-7AM)	Incremental Backup (6-7AM)	Full Incremental Backup (6-9AM)	Incremental Backup (6-7AM)	Incremental Backup (6-7AM)	Full incremental Backup (6-9AM)

RESULT AND CONCLUSION

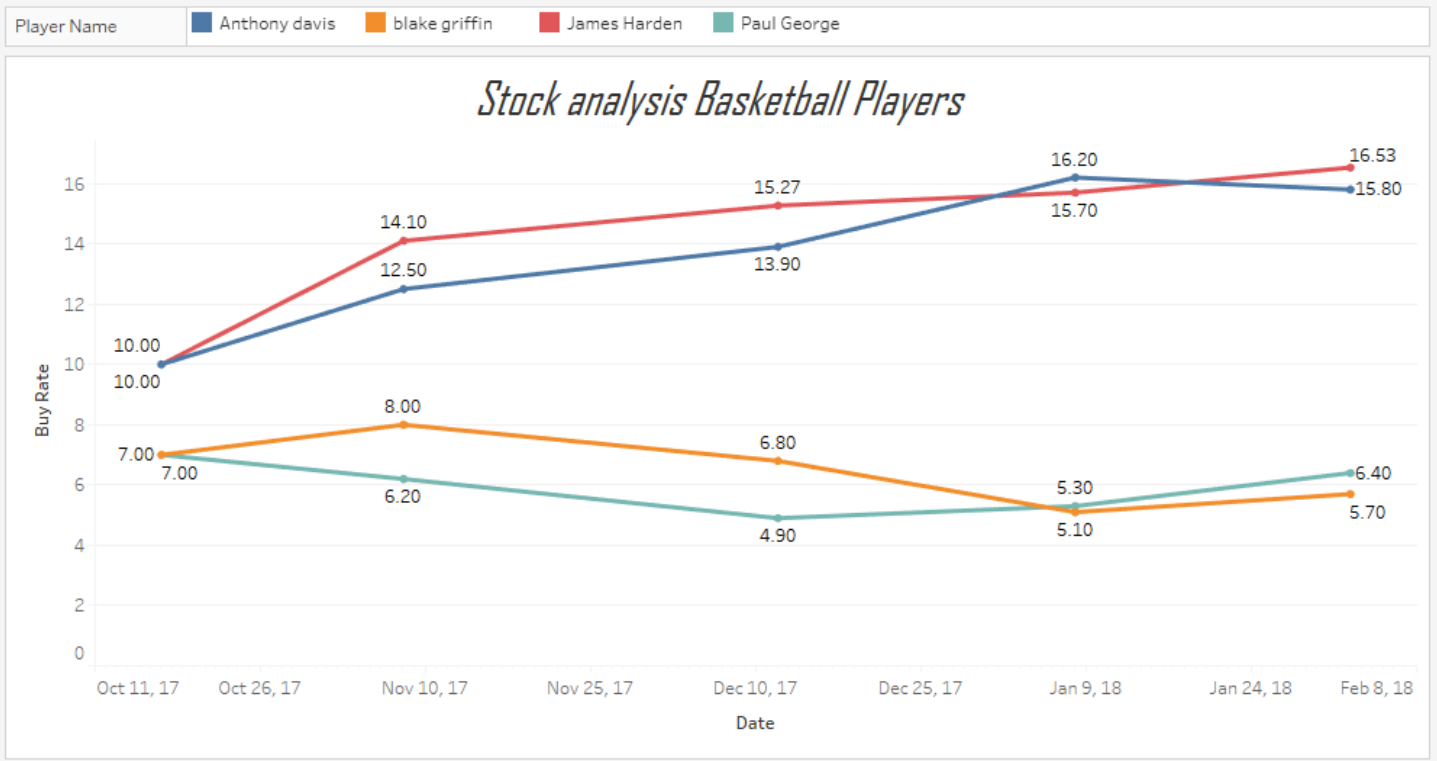
To display the result these players have been put through our price algorithm over 5 game days and their change in prices and new buy rate have been recorded for these games.

The collected Data has then been visualized on the tableau tool creating these line graphs.

This Visualization has been createfrom our database project, to show the change in stock price over a period.

Basketball

- James Harden
- Anthony Davis
- Paul George
- Blake Griffin



The likes of James Harden and Anthony Davis have started at a base price of 10$ as they are ranked in the top 10 in the player rater. The players Paul George and Blake Griffin start at a base price of 7$ as they are ranked between 11-20 in player rater.

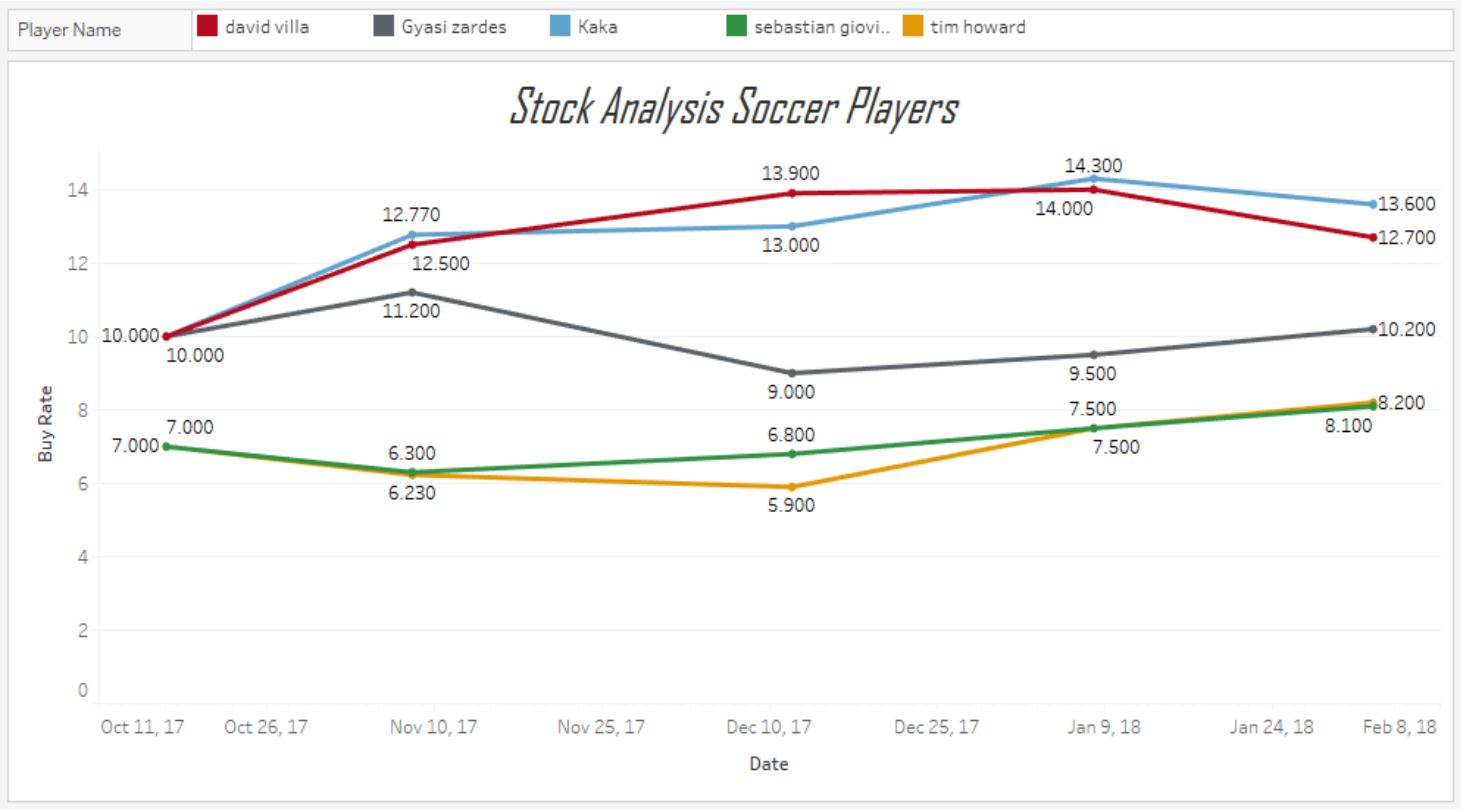
As you can see players James Harden and Anthony Davis have endured a good run for the 5 sample games

The steady increase in stock price complements their form in those 5 games.

Athletes Blake Griffin and Paul George have shown an irregular form in these 5 sample games the visualization shows the fluctuation in price.

Soccer

- Kaka
- Sebastian Giovinco
- Gyasi Zardes
- David Villa
- Tim Howard



This Visualization shows us that David Villa and Kaka are in sublime form and their price has been surging in these 5 sample games. The same cannot be said about Gyasi Zardes who starts at the same price as David Villa and Kaka but endures a dip in form thereby seeing his stock price plummet. The likes of Sebastian Giovinco and Tim Howard start at a lower price as compared to the previous players as they are ranked higher in the player rater as compared to those players. Their line graphs show us that they have been putting in average performances, as a result, we see very little shift in their price over the 5 games.

Taking our problem statement into consideration we have developed a database and a front end for an athlete stock exchange. Using our price determination algorithm which determines the price of a player dynamically we have created two separate markets

- National Basketball Association PlayerMarket
- Major League Soccer PlayerMarket

Users can navigate between both markets and buy shares of players they are interested in. We can proudly say that we have reached our objectives that we have set for ourselves during project proposal stages.

We have Managed to design a fully functional athlete stock exchange from scratch with a user interface.

APPENDIX PROJECT CODE

MYSQL CODE :

```
create database ballerindex1;
use ballerindex1;


CREATE TABLE `State`(
  `Zip` integer(7) NOT NULL,
  `City` Varchar(20) NOT NULL,
  `State` varchar(20) NOT NULL,
  primary key (`Zip`)
);


CREATE TABLE `Address` (
  `UserID` integer(5) NOT NULL auto_increment,
  `Address` varchar(50) NOT NULL,
  `Zip` integer(7) NOT NULL,
  foreign key (`UserID`) references Customer(`UserID`),
  foreign key (`Zip`) references State(`Zip`)
);


CREATE TABLE `Customer` (
  `UserID` integer(5) NOT NULL auto_increment,
`first name` varchar(20) NOT NULL,
  `Last Name` varchar(20)NOT NULL,
  `Nationality` varchar(20) NOT NULL,
  `Age` integer NOT NULL,
  `SSN` varchar(50) NOT NULL,
  primary key (`UserID`)
);

SET foreign_key_checks = 0;

SET foreign_key_checks = 1;
```

```
CREATE TABLE `Logininfo` (
  `UserID` integer(5) NOT NULL,
  `Email` varchar(100) NOT NULL,
  `Password` varchar(15) NOT NULL,
  primary key (`UserID`,`Email`),
  foreign key (`UserID`) references `Customer` (`UserID`)
);


CREATE TABLE `Bank` (
  `BankID` integer(5) NOT NULL auto_increment,
  `UserID` integer(5) NOT NULL,
  `Name on Card` varchar(50) NOT NULL,
  `Card NO` double NOT NULL,
  `Expiry Date` date NOT NULL,
  primary key (`BankID`),
  foreign key (`UserID`) references `Customer`(`UserID`)
);
```



```
create table `GameType` (
`GameTypeID` integer(5) NOT NULL,
`GameType` nvarchar(20),
primary key (`GameTypeID`));


create table `StatsType` (
`StatsTypeID` integer(5) NOT NULL,
`StatsType` nvarchar(20),
primary key (`StatsTypeID`));


CREATE TABLE `PlayerMarket` (
  `PlayerID` integer(5) NOT NULL auto_increment,
  `GameTypeID` integer(5) NOT NULL,
 `PlayerName` varchar(50) NOT NULL,
  `Height` numeric(4,2) NOT NULL,
  `Weight` numeric(4,2) NOT NULL,
  `Age` integer(5) NOT NULL,
  `Nationality` varchar(20) NOT NULL,
  primary key (`PlayerID`),
```

```
  foreign key (`GameTypeID`) references `GameType`(`GameTypeID`)
);



CREATE TABLE `Price` (
  `PriceID` integer(5) NOT NULL auto_increment,
  `PlayerID` integer(5) NOT NULL,
  `BuyRate` numeric(7,2) NOT NULL,
  `SellRate` numeric(7,2) NOT NULL,
  primary key (`PriceID`),
  foreign key (`PlayerID`) references `PlayerMarket`(`PlayerID`));


SET foreign_key_checks = 0;

SET foreign_key_checks = 1;


create table `portfolio`(
`PortfolioID` integer(5) NOT NULL auto_increment,
`UserID` integer(5) NOT NULL,
primary key (`PortfolioID`),
foreign key (`UserID`) references customer(`UserID`));


 CREATE TABLE  `PortfolioItems` (
  `PortfolioItemID` integer(5) NOT NULL auto_increment,
  `PortfolioID` integer(5) NOT NULL,
  `PlayerID` integer(5) NOT NULL,
  `No of shares` integer(5) NOT NULL,
  `PriceID` integer(5) NOT NULL,
  `Profits` numeric(10,2),
  `UserID` integer(5),
  primary key(`PortfolioItemID`),
  foreign key (`UserID`) references `Customer` (`UserID`),
  foreign key (`PlayerID`) references `PlayerMarket` (`PlayerID`),
  foreign key (`PriceID`) references `Price` (`PriceID`),
  foreign key(`PortfolioID`) references portfolio(`PortfolioID`)
);

create table `Cart`(
```

```
    `CartID` integer(5) not null auto_increment,
    `UserID` integer(5) not null,
    primary key(`CartID`),
    foreign key(`UserID`) references `Customer` (`UserID`));

CREATE TABLE `CartItems` (
  `CartItemsID` integer(5) NOT NULL auto_increment,
  `CartID` integer(5) not null,
  `UserID` integer(5) NOT NULL,
  `PlayerID` integer(5) NOT NULL,
  `Buy Rate` numeric(7,2) NOT NULL,
  `NoofShares` integer(5) NOT NULL,
  `Amount` integer(20) NOT NULL,
  `orderplaced` varchar(50) NOT NULL,
  primary key (`CartItemsID`),
   foreign key(`CartID`) references `Cart` (`CartID`),
  foreign key(`UserID`) references `Customer` (`UserID`),
  foreign key (`PlayerID`) references `PlayerMarket` (`PlayerID`)
);
```



```
CREATE TABLE `Orders` (
  `OrderID` integer(5) NOT NULL auto_increment,
  `CartID` integer(5) NOT NULL,
  `UserID` integer(5) NOT NULL,
  `TotalAmount` integer(20) NOT NULL,
  `WalletID` integer(5) NOT NULL,
  primary key (`OrderID`),
  foreign key (`UserID`) references `Customer` (`UserID`),
   foreign key (`CartID`) references `Cart` (`CartID`) ,
   foreign key (`WalletID`) references `Wallet` (`WalletID`)
);


CREATE TABLE `Wallet` (
  `WalletID` integer(5) NOT NULL auto_increment,
  `UserID` integer(5) NOT NULL,
  `Balance` integer(5) NOT NULL,
  primary key (`WalletID`),
  foreign key (`UserID`) references `Customer` (`UserID`)
);
```

```
CREATE TABLE `Watchlist` (
  `WatchlistID` integer(5) NOT NULL auto_increment,
  `UserID` integer(5) NOT NULL,
  primary key (`WatchlistID`),
  foreign key (`UserID`) references `Customer` (`UserID`)
  );

create table `WatchlistItems` (
`WatchlistItemID` integer(5) NOT NULL auto_increment,
`WatchlistID` integer(5) NOT NULL,
`PlayerID` integer(5) NOT NULL,
`UserID` integer(5) NOT NULL,
`PriceID` integer(5) NOT NULL,
primary key(`WatchlistItemID`),
foreign key(`UserID`) references `Customer` (`UserID`),
foreign key (`PlayerID`) references `PlayerMarket` (`PlayerID`),
  foreign key (`PriceID`) references `Price` (`PriceID`),
  foreign key(`WatchlistID`) references `watchlist`(`WatchlistID`)
);
```



```
CREATE TABLE `Selling` (
  `sellingorderId` integer(5) NOT NULL auto_increment,
  `UserID` integer(5) NOT NULL,
  `PlayerID` integer(5) NOT NULL,
  `Sell rate` integer(5) NOT NULL ,
  `Noofshares` integer(5) NOT NULL,
  `Amountearned` integer(10) NOT NULL,
  `WalletID` integer(10) NOT NULL,
  primary key (`sellingorderId`),
foreign key (`WalletID`) references `Wallet` (`WalletID`),
foreign key (`UserID`) references `Customer` (`UserID`),
foreign key (`PlayerID`) references `PlayerMarket` (`PlayerID`)
);

 CREATE TABLE `fstats` (
  `StatsTypeID` integer(5) ,
  `PlayerID` integer(5),
  `FstatsID` integer(5) not null auto_increment ,
  `goals` numeric(7,2) not null,
  `assists` numeric(7,2) not null,
```

```
 `Saves` numeric(7,2) NOT NULL,
 `Interceptions` numeric(7,2) NOT NULL,
 `Clearances` numeric(7,2) NOT NULL,
 `Blocks` numeric(7,2) NOT NULL,
 `GoalsConceded` numeric(7,2) NOT NULL,
 `YellowCard` numeric(7,2) NOT NULL,
 `Redcard` numeric(7,2) NOT NULL,
 `PenaltyMissed` numeric(7,2) NOT NULL,
 `OwnGoal` numeric(7,2) NOT NULL,
 `Minutesplayed/90` numeric(7,2) NOT NULL,
 `PassPercentage` numeric(7,2) NOT NULL,
 `CleanSheet` integer(5) NOT NULL,
  primary key (`BstatsID`),
  foreign key (`PlayerID`) references `playermarket` (`PlayerID`),
  foreign key (`StatsTypeID`) references `StatsType` (`StatsTypeID`)
 );


 CREATE TABLE `Bstats` (
 `StatsTypeID` integer(5) ,
 `PlayerID` integer(5),
```

```
 `BstatsID` integer(5) not null auto_increment ,
 `Points` numeric(7,2) NOT NULL,
 `Rebounds` numeric(7,2) NOT NULL,
 `Assists` numeric(7,2) NOT NULL,
 `Steals` numeric(7,2) NOT NULL,
 `Blocks` numeric(7,2) NOT NULL,
 `3pointersmade` numeric(7,2) NOT NULL,
 `FG%` numeric(7,2) NOT NULL,
 `FT%` numeric(7,2) NOT NULL,
 `3PFG%`numeric(7,2) NOT NULL,
 `Turnovers` numeric(7,2) NOT NULL,
 `PersonalFouls` numeric(7,2) NOT NULL,
  primary key (`BstatsID`),
  foreign key (`PlayerID`) references `playermarket` (`PlayerID`),
   foreign key (`StatsTypeID`) references `StatsType` (`StatsTypeID`)
 );

/*customer details view*/

Create VIEW ballerindex1.`CUSTOMER_DETAILS` as
select
```

```
customer.UserID,
customer.`First Name`,
customer.`Last Name`,
customer.Nationality,
customer.age,
Address.Address,
state.zip,
state.state
from ballerindex1.Customer inner join ballerindex1.address
on customer.UserID=address.UserID
inner join ballerindex1.state on state.zip=address.Zip;

DELIMITER $$
create procedure ballerindex1.spviewcustomer(in ipuser integer(5))
BEGIN
select * from ballerindex1.`customer_details` where `customer_details`.UserID=ipuser;
end$$
Delimiter :

select*from customer_details;
```



```
/*stored procedure for signup*/

delimiter $$
CREATE procedure ballerindex1.SpSignUp(in
`iemail` varchar(100),
`ipassword` varchar(15),
`iname on card` varchar(20),
`icard no` integer(16),
`iexpiry date` date,
iaddress varchar(20),
izip integer(7),
icity varchar(20),
istate varchar(20),
`ifirst name` varchar(20),
`ilast name` varchar(20),
 inationality varchar(20),
 iage integer(5),
 iSSN integer(15))

BEGIN
DECLARE msg VARCHAR(25);
```

```
IF exists( select email from logininfo where logininfo.email = iemail)
	then set msg = 'USER ALREADY EXISTS';
else
	INSERT into `bank`(`bankid`, `name on card`, `card no`, `expiry date`)
		values (null, `iname on card`, `icard no`, `iexpiry date`);
	insert into `address`(address, zip)
		values (iaddress, izip);
		insert into `customer`  (`userid`, `first name`, `last name`, `nationality`,`age`, `ssn`)
		values (null,`iFirst Name`, `iLast Namesp`, `inationality`, `iDOB`,`issn`);
	insert into `state` (state, city)
		values (`istate`, `icity`);
	Insert into `logininfo`(`email`, `password`)
		values (`iemail`, `ipassword`);


End if;

End $$
Delimiter ;
```



```
/*store procedure for playermarket view*/

Delimiter $$
create procedure spViewPlayerName (in igametypeID nvarchar(20))
begin
select
PlayerMarket.PlayerName,
PlayerMarket.Height,
PlayerMarket.Weight,
PlayerMarket.Age,
Price.BuyRate,
Price.SellRate
from Ballerindex1.PlayerMarket
inner join Ballerindex1.Price
on PlayerMarket.PlayerID=Price.PlayerID
inner join Ballerindex1.gametype on gametype.GameTypeid=playermarket.GameTypeID
where playermarket.GameTypeID=igametypeID;
end$$
delimiter ;

/* stored procedure for player name and his details */
```

```
delimiter $$

Create PROCEDURE ballerindex1.spPlayerName(in ipPlayerName varchar(20))
BEGIN
declare a varchar(50);
set  a = (select gametype from playermarket inner join
 gametype on GameType.GameTypeid=playermarket.GameTypeID where
playername=ipPlayername);
If  a = 'basketball' then

select
playermarket.playername,
playermarket.height,
playermarket.weight,
playermarket.age,
gametype.gametype,
bstats.points,
bstats.rebounds,
bstats.assists,
bstats.steals,
bstats.blocks,
bstats.`3pointersmade`,
bstats.`FG%`,
bstats.`3PFG%`,
bstats.turnovers,
bstats.personalfouls
from playermarket inner join gametype on playermarket.GameTypeID=gametype.GameTypeID
inner join bstats on playermarket.playerID=bstats.playerID
where playermarket.playername = ipPlayerName;
else

select
playermarket.playername,
playermarket.height,
playermarket.weight,
playermarket.age,
gametype.gametype,
fstats.`goals`,
fstats.`assists`,
fstats.`saves`,
fstats.`interceptions`,
```

```
fstats.`clearances`,
fstats.`blocks`,
fstats.`goalsconceded`,
fstats.`yellowcard`,
fstats.`redcard`,
fstats.`penaltymissed`,
fstats.`owngoal`,
fstats.`minutesplayed/90`,
fstats.`passpercentage`,
fstats.`cleansheet`
from playermarket inner join gametype on playermarket.GameTypeID=gametype.GameTypeID
inner join fstats on playermarket.playerID=fstats.playerID
where playermarket.playername = ipPlayerName;

end if;

end$$
Delimiter ;

/* creating users and privileges*/
```



```
create user soumya identified by 'baller';
grant all privileges on ballerindex1 to soumya;
grant select, insert , update on ballerindex1.* to systemadmin;
grant select,insert,delete on ballerindex.customer to soumya;
grant select,insert on ballerindex to soumya;
create user user_account identified by 'qwerty';
create user admin@localhost identified by 'ballerindex';
grant all on *.* to admin@localhost with grant option;
show grants for admin@localhost;
revoke all privileges, grant option from admin@localhost;
show grants for admin@localhost;
grant select, update , delete on ballerindex1.* to admin@localhost;
create user Mainadmin@localhost identified by 'root';
grant all on *.* to Mainadmin@localhost with grant option;
show grants for user_account;
grant all on Ballerindex.* to Mainadmin@localhost with grant option;
grant insert , update on ballerindex1.Customer to user_account;
grant insert , update on ballerindex1.PortfolioItems to user_account;
grant insert , update on ballerindex1.CartItems to user_account;
revoke all privileges , grant option from user_account;
grant select on ballerindex1.Customer to user_account;
```

```
grant select on ballerindex1.PortfolioItems to user_account;
grant select on ballerindex1.CartItems to user_account;
grant select on ballerindex1.WatchlistItems to user_account;
grant select on ballerindex1.playermarket to user_account;


/* stored procedure for adding money to wallet */

delimiter $$
create procedure `spAddMoney` (in ipuserid integer(5), ipAddRemove varchar(6), ipAmount
integer(5))
BEGIN
declare a integer(50);
declare msg varchar(50);
set  a = (select walletID from wallet where userid = ipuserid);
If  ipAddRemove = 'Add' or 'add' or 'ADD' then
update ballerindex1.wallet set balance = balance + ipAmount where userid = ipuserid;
elseif ipAddRemove = 'Remove' or 'remove' or 'REMOVE' then
update ballerindex1.wallet set balance = balance - ipAmount where userid = ipuserid;
else
select msg;
end if;

end$$

Delimiter ;

/*price determining procedure for basketball*/


delimiter $$
create procedure `SPnewpriceB` (in iplayerID int(10))
begin

declare a integer(2);
declare vpts decimal(5,2);
declare vast decimal(5,2);
declare vrebs decimal(5,2);
declare v3pm decimal(5,2);
declare vstl decimal(5,2);
declare vblk decimal(5,2);
declare vfg decimal(5,2);
```

```
declare vft decimal(5,2);
declare v3fg decimal(5,2);
declare vtov decimal(5,2);
declare vpf decimal(5,2);
declare vpts1 decimal(5,2);
declare vast1 decimal(5,2);
declare vrebs1 decimal(5,2);
declare v3pm1 decimal(5,2);
declare vstl1 decimal(5,2);
declare vblk1 decimal(5,2);
declare vfg1 decimal(5,2);
declare vft1 decimal(5,2);
declare vtov1 decimal(5,2);
declare vpf1 decimal(5,2);
declare v3fg1 decimal(5,2);
declare vpri decimal(5,2);
declare vsec decimal(5,2);
declare vper decimal(5,2);
declare vneg decimal(5,2);
declare vpri1 decimal(5,2);
declare vsec1 decimal(5,2);
```

```
declare vper1 decimal(5,2);
declare vneg1 decimal(5,2);
declare changeinprice decimal(5,2);
declare currentprice decimal(5,2);
declare newsellprice decimal(5,2);
set a=(select playerMarket.GametypeID from playermarket where playerID=iPlayerID);
if a=2 then
set vpts = (select bstats.points from bstats where playerID=iplayerID and statstypeid = 1);
set vast = (select bstats.assists from bstats where playerID=iplayerID and statstypeid = 1);
set vrebs = (select bstats.rebounds from bstats where playerID=iplayerID and statstypeid = 1);
set v3pm = (select bstats.`3pointersmade` from bstats where playerID=iplayerID and statstypeid
= 1);
set vfg = (select bstats.`FG%` from bstats where playerID=iplayerID and statstypeid = 1);
set vft = (select bstats.`FT%` from bstats where playerID=iplayerID and statstypeid = 1);
set v3fg = (select bstats.`3PFG%` from bstats where playerID=iplayerID and statstypeid = 1);
set vtov = (select bstats.Turnovers from bstats where playerID=iplayerID and statstypeid = 1);
set vpf = (select bstats.PersonalFouls from bstats where playerID=iplayerID and statstypeid =
1);
set vstl = (select bstats.Steals from bstats where playerID=iplayerID and statstypeid = 1);
set vblk = (select bstats.Blocks from bstats where playerID=iplayerID and statstypeid = 1);
set vpts1 = (select bstats.points from bstats where playerID=iplayerID and statstypeid = 2);
```

```
set vast1 = (select bstats.assists from bstats where playerID=iplayerID and statstypeid = 2);
set vrebs1 = (select bstats.rebounds from bstats where playerID=iplayerID and statstypeid = 2);
set v3pm1 = (select bstats.`3pointersmade` from bstats where playerID=iplayerID and
statstypeid = 2);
set vfg1 = (select bstats.`FG%` from bstats where playerID=iplayerID and statstypeid = 2);
set vft1 = (select bstats.`FT%` from bstats where playerID=iplayerID and statstypeid = 2);
set v3fg1 = (select bstats.`3PFG%` from bstats where playerID=iplayerID and statstypeid = 2);
set vtov1 = (select bstats.Turnovers from bstats where playerID=iplayerID and statstypeid = 2);
set vpf1 = (select bstats.PersonalFouls from bstats where playerID=iplayerID and statstypeid =
2);
set vstl1 = (select bstats.Steals from bstats where playerID=iplayerID and statstypeid = 2);
set vblk1 = (select bstats.Blocks from bstats where playerID=iplayerID and statstypeid = 2);
set vpri = (vpts+vast+vrebs);
set vsec = (v3pm+vstl+vblk);
set vper = (vfg+vft+v3fg);
set vneg = (vtov+vpf);
set vpri1 = (vpts1+vast1+vrebs1);
set vsec1 = (v3pm1+vstl1+vblk1);
set vper1 = (vfg1+vft1+v3fg1);
set vneg1 = (vtov1+vpf1);
set vpri = (vpri-vpri1)*0.2;
```



```
set vsec = (vsec-vsec1)*0.2;
set vper = (vper-vper1)/100;
set vneg = (vneg-vneg1)*0.2;
set changeinprice = (vpri+vsec+vper+vneg);
set currentprice = (select price.buyrate from price where playerID=iplayerID);
set currentprice = (currentprice+changeinprice);
update price set buyrate=currentprice where playerID=iplayerID;
set newsellprice = (select price.SellRate from price where playerID=iplayerID);
set newsellprice = (newsellprice+changeinprice);
update price set sellrate=newsellprice where playerID=iplayerID;
select currentprice;

end if;
end$$
delimiter ;


/* price determining algorithm for soccer*/

delimiter $$
create procedure `SPnewpriceF` (in iplayerID int(10))
```

```
begin
declare a integer(2);
declare vgls decimal(5,2);
declare vast decimal(5,2);
declare vsav decimal(5,2);
declare vint decimal(5,2);
declare vclr decimal(5,2);
declare vblk decimal(5,2);
declare vgc decimal(5,2);
declare vyc decimal(5,2);
declare vrc decimal(5,2);
declare vpm decimal(5,2);
declare vog decimal(5,2);
declare vmp decimal(5,2);
declare vppe decimal(5,2);
declare vcs decimal(5,2);
declare vgls1 decimal(5,2);
declare vast1 decimal(5,2);
declare vsav1 decimal(5,2);
declare vint1 decimal(5,2);
declare vclr1 decimal(5,2);
```

```
declare vblk1 decimal(5,2);
declare vgc1 decimal(5,2);
declare vyc1 decimal(5,2);
declare vrc1 decimal(5,2);
declare vpm1 decimal(5,2);
declare vog1 decimal(5,2);
declare vmp1 decimal(5,2);
declare vppe1 decimal(5,2);
declare vcs1 decimal(5,2);
declare vpri decimal(5,2);
declare vsec decimal(5,2);
declare vper decimal(5,2);
declare vneg decimal(5,2);
declare vbon decimal(5,2);
declare vpri1 decimal(5,2);
declare vsec1 decimal(5,2);
declare vper1 decimal(5,2);
declare vneg1 decimal(5,2);
declare vbon1 decimal (5,2);
declare changeinprice decimal(5,2);
declare currentprice decimal(5,2);
```

```
declare newsellprice decimal(5,2);
set a=(select playerMarket.GametypeID from playermarket where playerID=iPlayerID);
if a=1 then
set vgls = (select fstats.goals from fstats where playerID=iplayerID and statstypeid = 1);
set vast = (select fstats.assists from fstats where playerID=iplayerID and statstypeid = 1);
set vsav = (select fstats.saves from fstats where playerID=iplayerID and statstypeid = 1);
set vint = (select fstats.Interceptions from fstats where playerID=iplayerID and statstypeid = 1);
set vclr = (select fstats.Clearances from fstats where playerID=iplayerID and statstypeid = 1);
set vblk = (select fstats.Blocks from fstats where playerID=iplayerID and statstypeid = 1);
set vgc = (select fstats.GoalsConceded from fstats where playerID=iplayerID and statstypeid =
1);
set vyc = (select fstats.YellowCard from fstats where playerID=iplayerID and statstypeid = 1);
set vrc = (select fstats.Redcard from fstats where playerID=iplayerID and statstypeid = 1);
set vpm = (select fstats.PenaltyMissed from fstats where playerID=iplayerID and statstypeid =
1);
set vog = (select fstats.OwnGoal from fstats where playerID=iplayerID and statstypeid = 1);
set vmp = (select fstats.`Minutesplayed/90` from fstats where playerID=iplayerID and
statstypeid = 1);
set vppe = (select fstats.PassPercentage from fstats where playerID=iplayerID and statstypeid =
1);
set vcs = (select fstats.CleanSheet from fstats where playerID=iplayerID and statstypeid = 1);
```

```
set vgls1 = (select fstats.goals from fstats where playerID=iplayerID and statstypeid = 2);
set vast1 = (select fstats.assists from fstats where playerID=iplayerID and statstypeid = 2);
set vsav1 = (select fstats.saves from fstats where playerID=iplayerID and statstypeid = 2);
set vint1 = (select fstats.Interceptions from fstats where playerID=iplayerID and statstypeid =
2);
set vclr1 = (select fstats.Clearances from fstats where playerID=iplayerID and statstypeid = 2);
set vblk1 = (select fstats.Blocks from fstats where playerID=iplayerID and statstypeid = 2);
set vgc1 = (select fstats.GoalsConceded from fstats where playerID=iplayerID and statstypeid =
2);
set vyc1 = (select fstats.YellowCard from fstats where playerID=iplayerID and statstypeid = 2);
set vrc1 = (select fstats.Redcard from fstats where playerID=iplayerID and statstypeid = 2);
set vpm1 = (select fstats.PenaltyMissed from fstats where playerID=iplayerID and statstypeid =
2);
set vog1 = (select fstats.OwnGoal from fstats where playerID=iplayerID and statstypeid = 2);
set vmp1 = (select fstats.`Minutesplayed/90` from fstats where playerID=iplayerID and
statstypeid = 2);
set vppe1 = (select fstats.PassPercentage from fstats where playerID=iplayerID and statstypeid
= 2);
set vcs1 = (select fstats.CleanSheet from fstats where playerID=iplayerID and statstypeid = 2);
set vpri = (vgls+vast);
set vsec = (vsav+vint+vclr+vblk+vgc);
```

```
set vper = (vmp+vppe);
set vneg = (vyc+vrc+vpm+vog);
set vbon = (vcs);
set vpri1 = (vgls1+vast1);
set vsec1 = (vsav1+vint1+vclr1+vblk1+vgc1);
set vper1 = (vmp1+vppe1);
set vneg1 = (vyc1+vrc1+vpm1+vog1);
set vbon1 = (vcs1);
set vpri = (vpri-vpri1)*0.5;
set vsec = (vsec-vsec1)*0.3;
set vper = (vper-vper1)/100;
set vneg = (vneg-vneg1)*0.2;
set vbon = (vbon-vbon1)*0.1;
set changeinprice = (vpri+vsec+vper+vneg);
set currentprice = (select price.buyrate from price where playerID=iplayerID);
set currentprice = (currentprice+changeinprice);
select changeinprice;
update price set buyrate=currentprice where playerID=iplayerID;
set newsellprice = (select price.SellRate from price where playerID=iplayerID);
set newsellprice = (newsellprice+changeinprice);
update price set sellrate=newsellprice where playerID=iplayerID;
```

```
select currentprice;

end if;
end$$
delimiter ;


/* sprocedure for adding stock to cart*/

delimiter @@
create procedure spAddtocart(`iUserID` integer(5),`iPlayerID` integer(5),`inoofshares`
integer(5))
Begin
declare cid integer(5);
declare pric integer(5);
set cid=(select CartID from Cart where cart.UserID=iUserID);
set pric= (select price.BuyRate from price where PriceID=(
select PriceID from price where price.playerID=iplayerID));
insert into CartItems values(null,cid,iUserID,iPlayerID,inoofshares,pric*inoofshares,null);
end@@
```

```
delimiter ;


/* updating order store procedure*/

delimiter $$
create procedure SPupdateorder(`iUserID` integer(5))
begin
declare c integer(10);
declare a decimal(7,2);
declare w integer(10);
declare b decimal(7,2);
set c=(select distinct(CartID) from cartitems where cartitems.UserID=iUserID);
set a=(select sum(Amount) from  cartitems where cartitems.UserID=iUserID);
set w=(select WalletID from wallet where wallet.UserID=iUserID);
insert into orders values(null,c,iUserID,a,w);
update cartitems set orderplaced = 'yes' where cartitems.UserID=iUserID;
set b=(select balance from wallet where wallet.UserID=iUserID);
set a= (b-a);
update wallet set balance=a where wallet.UserID=UserID;
end$$
```

```
delimiter ;

/*add to watchlist store procedure */


delimiter $$
create procedure spAddWatchlist (in iplayerid integer(5), iaddremove varchar(10), iuserid
integer(5))
begin
declare addremove varchar(10);
declare currentprice decimal(5);
declare watchlistidv integer(5);
declare p integer(5);
set watchlistidv = (select watchlistid from watchlist where watchlist.UserID = iuserid);
set p = (select priceID from price where price.PlayerID=iplayerID);
If  iaddremove = 'Add' or 'add' or 'ADD' then
insert into watchlistitems
values (null,watchlistidv, iplayerid, iuserid, p);
elseif iAddRemove = 'Remove' or 'remove' or 'REMOVE' then
delete from watchlistitems where watchlistitems.PlayerID = iplayerid and
watchlistitems.UserID = iuserid;
```

```
end if;
end$$
 delimiter ;

/* signup store procedure */

delimiter $$
CREATE procedure SpSignUp(
 in`iemail` varchar(100),
 `ipassword` varchar(15),
 `iname on card` varchar(20),
`icard no` integer(16),
`iexpiry date` date,
`iaddress` varchar(20),
`izip` integer(7),
`ifirst name` varchar(20),
`ilast name` varchar(20),
`inationality` varchar(20),
`iage` integer(5),
`iSSN` integer(15))
```



```
BEGIN
DECLARE msg VARCHAR(25);
declare data1 integer;
IF exists( select email from logininfo where logininfo.email = iemail)
        then set msg = 'USER ALREADY EXISTS';
   select msg;
else
   insert into `customer` values (null,`iFirst Name`, `iLast Name`, `inationality`, `iage`,`issn`);
      set data1=last_insert_id();
      insert into `address`values (data1,`iaddress`, `izip`);
        INSERT into `bank`values (null,data1, `iname on card`, `icard no`, `iexpiry date`);
   Insert into `logininfo`values (data1,`iemail`, `ipassword`);
   insert into `portfolio` values(null,data1);
   insert into `cart` values(null,data1);
   insert into `watchlist` values(null,data1);
   insert into `wallet` values(null,data1,0);

 End if;
End $$
Delimiter ;
```

/*store procedure for selling */

```
 DELIMITER $$
CREATE PROCEDURE `SpSelling`(`iuserid`integer (5), `iplayerid` integer (5), `inoofshares` integer (5))
   BEGIN
        declare profitloss decimal (7,3);
   declare uid integer(5);
   declare pid integer (5);
   declare nos integer(5);
   declare avgrate decimal(7,3);
   declare sllrate decimal(7,3);
   declare diff decimal (7,3);
   declare wallID integer(5);
   declare samount decimal(7,3);
   declare tprice decimal(7,3);
   set nos = (select `No of shares`-inoofshares from portfolioitems
   where portfolioitems.UserID=iUserID and portfolioitems.playerID=iplayerID);
        set avgrate= (select AvgBuyRate from portfolioitems
   where  portfolioitems.UserID=iUserID and portfolioitems.playerID=iplayerID);
   set wallID=(select walletID from wallet where userID=iuserID);
   set sllrate = (select SellRate from price where playerid= iplayerid);
   set diff= inoofshares*(sllrate-avgrate);
   set samount=sllrate*inoofshares;
   set tprice=nos*sllrate;
   insert into selling values(null,iuserID,iplayerID,inoofshares,diff,wallID);

   update wallet set balance = `balance`+ samount where userid= iuserid;
   update portfolioitems set `no of shares` = nos ,totalstockprice=tprice
   where playerid= iplayerid and userID=iUserID;
   END $$
DELIMITER ;
```


/* store procedure for adding to portfolio */

```
delimiter $$
create procedure spAddtoPortfolio(`iUserID` integer(5),`iPlayerID` integer(5),`inoofshares` integer(5))
begin
declare msg varchar(20);
declare pid integer(5);
```

```
declare pric decimal(5,2);
declare pricid integer(5);
declare tprice decimal(7,3);
declare newnoofshares integer(5);
declare ibuyrate decimal(7,3);
declare spric decimal(7,3);
set pid=(select portfolioID from portfolio where portfolio.UserID=iUserID);
set pric= (select price.BuyRate from price where PriceID=(
select PriceID from price where price.playerID=iplayerID));
set pricid=(select PriceID from price where price.playerID=iplayerID);

set spric= (select price.SellRate from price where PriceID=(
select PriceID from price where price.playerID=iplayerID));


if
exists(select portfolioItems.playerID from portfolioitems where
portfolioItems.playerID=iplayerID and userID=iUserID)
then
set newnoofshares=(select `no of shares`+inoofshares from portfolioitems where
PlayerID=iplayerID and UserID=iUserID);
```

```
set tprice=spric*newnoofshares;
set ibuyrate=(select AvgBuyRate from portfolioitems where  PlayerID=iplayerID and
UserID=iUserID);

set ibuyrate=(ibuyrate+pric)/2;

update portfolioitems set `No of shares`=`newnoofshares`,`totalstockprice`=
tprice,`AvgBuyRate`=ibuyrate
 where  PlayerID=iplayerID and UserID=iUserID;
else
set newnoofshares=`inoofshares`;
set tprice= pric*newnoofshares;
insert into portfolioitems values(null,pid,iplayerID,newnoofshares,pric,tprice,pricid,`iUserID`);

end if;
end$$
delimiter ;

/*trigger for selling */

delimiter @@
```

```
create trigger trigger_selling
after insert on selling
for each row
begin
declare sid integer(5);
declare uid integer(5);
set sid=(select max(sellingorderId) from selling);
set uid=(select UserID from selling where sellingorderID=sid);

delete from portfolioitems where  UserID=uid and portfolioitems.`No of shares`=0;
end@@

delimiter ;

/*trigger for adding to portfolio */

delimiter !!
create trigger triggeraddportfolio
after insert on orders
for each row
begin
```

```
declare cid integer(5);
declare uid integer(5);
declare pid integer(5);
declare nos integer(5);
declare oid integer(5);
declare notfound integer default 0;
DECLARE cur1 CURSOR FOR SELECT cartitemsID FROM cartItems WHERE userID=3;

DECLARE CONTINUE HANDLER
      FOR NOT FOUND SET notfound = 1;
open cur1;
get_details:loop
fetch cur1 into cid;
set uid=(select UserID from cartitems where cartitemsID=cid);
set pid=(select playerID from cartitems where cartitemsID=cid);
set nos=(select noofshares from cartitems where cartitemsID=cid);
call spaddtoportfolio(uid,pid,nos);
if notfound=1
then
leave get_details;
end if;
```

```
end loop get_details;
close cur1;
end !!
delimiter ;
```

PHP CODE :

1.Php Code to connect to the mysql database.

```
<?php
$host = "localhost";
$userName = "root";
$password = "SreyaReddy6@";
$dbName = "BallerIndex";
 // Create database connection
$conn = mysqli_connect ($host, $userName, $password, $dbName);
 // Check connection
if ($conn->connect_error) {
die("Connection failed: " . $conn->connect_error);
}
else {
        echo("Connected");
}
?>
```


2.HTML code to create a New User Registration form

```
<html>
<h1 align='center'> BALLER INDEX </h1>
<body>
<form method="post" action="Conn.php"> //*Form begins here*//
<fieldset>
<legend>Registration Form</legend>
<table width="400" border="0" cellpadding="10" cellspacing="10">
<tr>
<td style="font-weight: bold"><div align="right"><label for="FirstName">First
Name</label></div></td>
<td><input name="FirstName" type="text" class="input" size="25" required /></td>
</tr>
<tr>
```

```
<td style="font-weight: bold"><div align="right"><label for="LastName">Last
Name</label></div></td>
<td><input name="LastName" type="text" class="input" size="25" required /></td>
</tr>
<tr>
<td style="font-weight: bold"><div align="right"><label
for="email">Email</label></div></td>
<td><input name="Email" type="email" class="input" size="25" required /></td>
</tr>
<tr>
<td height="23" style="font-weight: bold"><div align="right"><label
for="password">Password</label></div></td>
<td><input name="Password" type="password" class="input" size="25" required /></td>
</tr>
<tdstyle="font-weight:bold"><div align="right"><label
for="Nationality">Nationality</label></div></td>
<td><input name="Nationality" type="text" class="input" size="25" required /></td>
</tr>
<tr>
<td style="font-weight: bold"><div align="right"><label for="age">Age</label></div></td>
<td><input name="Age" type="integer" class="input" size="25" required /></td>
</tr>
<tr>
<td style="font-weight: bold"><div align="right"><label for="SSN">SSN</label></div></td>
<td><input name="SSN" type="integer" class="input" size="25" required /></td>
</tr>
<tr>
<td style="font-weight: bold"><div align="right"><label
for="Address">Address</label></div></td>
<td><input name="Address" type="text" class="input" size="25" required /></td>
</tr>
<tr>
<td style="font-weight: bold"><div align="right"><label for="Zip">Zip</label></div></td>
<td><input name="Zip" type="integer" class="input" size="25" required /></td>
</tr>
<tr>
<td style="font-weight: bold"><div align="right"><label for="City">City</label></div></td>
<td><input name="City" type="text" class="input" size="25" required /></td>
</tr>
<tr>
<td style="font-weight: bold"><div align="right"><label for="State">State</label></div></td>
<td><input name="State" type="text" class="input" size="25" required /></td>
```

```
</tr>
<tr>
<td height="23"></td>
<td><div align="right">
  <input type="submit" name="submit" value="Register!"  /> </div></td>
//* On clicking submit the details given in the form are passed as parameters to the next page*//
</tr>
</table>
</fieldset>
</form>
</body>
</html>
```

Php code to store the New User details in the database.


```
html>
<body>
<?php
require_once("Connect.php");

 //* Getting  all the details from the form and assigning them to variables *//
$FirstName = $_POST['FirstName'];
$LastName = $_POST['LastName'];
$Email =$_POST['Email'];
$Password =$_POST['Password'];
$Nationality = $_POST['Nationality'];
$Age = $_POST['Age'];
$SSN = $_POST['SSN'];
$Address = $_POST['Address'];
$Zip = $_POST['Zip'];
$City = $_POST['City'];
$State= $_POST['State'];

$sql1="INSERT INTO Customer (`First Name`,`Last Name`,`Nationality`,`Age`,`SSN`)
VALUES ('$FirstName','$LastName','$Nationality','$Age','$SSN')";
echo $sql1;
if ($conn->query($sql1) === TRUE) {
$sql2="INSERT INTO Logininfo (`UserID`,`Email`,`Password`) VALUES
('$lastid','$Email','$Password')";
$sql3="INSERT INTO Address (`UserID`,`Address`,`Zip`) VALUES
('$lastid','$Address','$Zip')";
echo $sql2;
```

```
echo $sql3;
if($conn->query($sql2) === TRUE&&$conn->query($sql3) === TRUE){

        echo "New record created successfully";
}
        }

else
{
    echo "Error: " . $sql . "<br>" . $conn->error;
}
?>
</body>
</html>
```

HTML code for login page.

```
<html>
<body align='center'>
<style type="text/css">
```

```
body
{
   background-image:url('ballerindex.png');
   height: 100%;
   background-position: center;
   background-repeat: repeat;
   background-size: cover;
}
table {
        align:center;
   margin: 0 auto;
        font size="20"
}
</style>
<table width="300" border="0" cellpadding="0" cellspacing="1" bgcolor="#CCCCCC"
align="center"  >
<tr>
<form name="form1" method="post" action="CheckLogin.php">
<td>
<table width="100%" border="0" cellpadding="3" cellspacing="1" bgcolor="#FFFFFF">
<tr>
```

```
<td colspan="3"><strong>Member Login </strong></td>
</tr>
<tr>
<td width="78">Email</td>
<td width="6">:</td>
<td width="294"><input name="Email" type="text" id="Email"></td>
</tr>
<tr>
<td>Password</td>
<td>:</td>
<td><input name="Password" type="password" id="Password"></td>
</tr>
<tr>
<td> </td>
<td> </td>
<td><input type="submit" name="Submit" value="Login"></td>
</tr>
</table>
</td>
</form>
</tr>
</table>
<input type="button" value="New User?"
onclick="window.location.href='http://localhost/BallerIndex/hello.php'" />
</body>
</html>
```


5.Php code to authenticate Login

```
<?php

require_once("connect.php");

// username and password sent from form
$Email=$_POST['Email'];
$Password=$_POST['Password'];

// To protect MySQL injection (more detail about MySQL injection)
$Email = stripslashes($Email);
$Password = stripslashes($Password);
$Email = mysqli_real_escape_string($conn,$Email);
$Password = mysqli_real_escape_string($conn,$Password);
```

```
$sql="SELECT Email,Password FROM Logininfo WHERE Email='$Email' and
Password='$Password'";


$result=mysqli_query($conn,$sql);


$count=mysqli_num_rows($result);

$sql1="SELECT UserID FROM Logininfo WHERE Email='$Email' and
Password='$Password'";
echo $sql1;
$row=mysqli_query($conn,$sql1);
$id=mysqli_fetch_assoc($row);
echo $id["UserID"];
$uid=$id["UserID"];
echo $uid;
if($count==1){

  header("location:HomePage.php?id=".$uid);
  exit();
}
else {
echo "Wrong Username or Password";
}

?>
```


6.Php code to View HomePage

```
<html>
<h1 align='center'>Baller Index</h1>
<?php
session_start();
$uid=$_GET['id'];
$url = 'ballerindex.png';

?>


<style type="text/css">
body
```

```
{

        background-image:url('ballerindex.png');
    height: 100%;
    background-position: center;
    background-repeat: repeat;
    background-size: cover;
        font-size: 24px;

}

</style>
<body align='center'>

<input type="button" value="Basketball PlayerMarket"
onclick="window.location.href='http://localhost/BallerIndex/PlayerMarket.php?id=<?php echo
$uid ?>'" style="width:200px; height:70px;"/>
<input type="button" value="Soccer PlayerMarket"
onclick="window.location.href='http://localhost/BallerIndex/SPlayerMarket.php?id=<?php
echo $uid ?>'" style="width:200px; height:70px;"/>
<input type="button" value="Portfolio"
onclick="window.location.href='http://localhost/BallerIndex/ViewPortfolio.php?id=<?php echo
$uid ?>'" style="width:200px; height:70px;"/>
<input type="button" value="Watchlist"
onclick="window.location.href='http://localhost/BallerIndex/ViewWatchlist.php?id=<?php
echo $uid ?>'" style="width:200px; height:70px;"/>
<input type="button" value="Cart"
onclick="window.location.href='http://localhost/BallerIndex/CartView.php?id=<?php echo
$uid ?>'" style="width:200px; height:70px;"/>
<input type="button" value="Log Out"
onclick="window.location.href='http://localhost/BallerIndex/Login.html?id=<?php echo $uid
?>'" style="width:200px; height:70px;"/>

</body>
</html>
```


7. Php code to View Basketball PlayerMarket

```
<html>
<h1 align='center'>Baller Index</h1>
<?php
session_start();
$uid=$_GET['id'];
```

```
$url = 'ballerindex.png';
?>
<style type="text/css">
body
{

        background-image:url('ballerindex.png');
    height: 100%;
    background-position: center;
    background-repeat: repeat;
    background-size: cover;
        font-size: 24px;

}
</style>
<body align='center'>
<input type="button" value="Basketball PlayerMarket"
onclick="window.location.href='http://localhost/BallerIndex/PlayerMarket.php?id=<?php echo
$uid ?>'" style="width:200px; height:70px;"/>
<input type="button" value="Soccer PlayerMarket"
onclick="window.location.href='http://localhost/BallerIndex/SPlayerMarket.php?id=<?php
echo $uid ?>'" style="width:200px; height:70px;"/>
<input type="button" value="Portfolio"
onclick="window.location.href='http://localhost/BallerIndex/ViewPortfolio.php?id=<?php echo
$uid ?>'" style="width:200px; height:70px;"/>
<input type="button" value="Watchlist"
onclick="window.location.href='http://localhost/BallerIndex/ViewWatchlist.php?id=<?php
echo $uid ?>'" style="width:200px; height:70px;"/>
<input type="button" value="Cart"
onclick="window.location.href='http://localhost/BallerIndex/CartView.php?id=<?php echo
$uid ?>'" style="width:200px; height:70px;"/>
<input type="button" value="Log Out"
onclick="window.location.href='http://localhost/BallerIndex/Login.html?id=<?php echo $uid
?>'" style="width:200px; height:70px;"/>
</body>
</html>
```


8. PHP Code to View Basketball Player details.

```
<html>
<h1 align='center'> BALLER INDEX </h1>
<body>
<?php
```

```
$url = 'ballerindex.png';
?>
<style type="text/css">
.tg  {border-collapse:collapse;border-spacing:0;border-color:#ccc;}
.tg td{font-family:Arial, sans-serif;font-size:14px;padding:10px 5px;border-style:solid;border-
width:1px;overflow:hidden;word-break:normal;border-color:#ccc;color:#333;background-
color:#fff;}
.tg th{font-family:Arial, sans-serif;font-size:14px;font-weight:normal;padding:10px
5px;border-style:solid;border-width:1px;overflow:hidden;word-break:normal;border-
color:#ccc;color:#333;background-color:#f0f0f0;}
.tg .tg-nleg{font-size:14px;font-family:"Lucida Console", Monaco, monospace !important;;text-
align:center;vertical-align:top}
</style>

<style type="text/css">
body
{
        background-image:url('<?php echo $url ?>');
   height: 100%;
   background-position: center;
   background-repeat: repeat;
   background-size: cover;
}
</style>
<?php
require_once("connect.php");
$uid=$_GET['id'];
?>
<input type="button" value="Back"
onclick="window.location.href='http://localhost/BallerIndex/Playermarket.php?id=<?php echo
$uid ?>'" style="width:150px; height:70px;"/>
<table width=50% class="tg" align="center">
<tr>
<th class="tg-nleg"> PlayerId </th>
<th> Player Name </th>
<th> Height </th>
<th> Weight </th>
<th> Age </th>
</tr>
<?php
session_start();
require_once("connect.php");
```

```
$sql="SELECT PlayerID,PlayerName,Hieght ,Weight ,Age FROM  PlayerMarket Where
PlayerMarket.GameTypeID = 2";
$result = mysqli_query($conn,$sql);
if( mysqli_num_rows($result) > 0)
{
        while($row = mysqli_fetch_array($result))

        {

                echo "<tr>";
                echo"<td>".$row['PlayerID']."</td>";
                echo"<td>".$row['PlayerName']."</td>";
                echo"<td>".$row['Hieght']."</td>";
                echo"<td>".$row['Weight']."</td>";
                echo"<td>".$row['Age']."</td>";
                echo"</tr>";
        }
}
        else
        {
                echo "0rows";
        }
?>
</table>
</body>
 </html>
```


9.Php code to view Soccer Player Market

```
<html>
<h1 align='center' color='white'> BALLER INDEX </h1>
<body align='center'>
<style type="text/css">
.tg  {border-collapse:collapse;border-spacing:0;border-color:#ccc;}
.tg td{font-family:Arial, sans-serif;font-size:14px;padding:10px 5px;border-style:solid;border-
width:1px;overflow:hidden;word-break:normal;border-color:#ccc;color:#333;background-
color:#fff;}
.tg th{font-family:Arial, sans-serif;font-size:14px;font-weight:normal;padding:10px
5px;border-style:solid;border-width:1px;overflow:hidden;word-break:normal;border-
color:#ccc;color:#333;background-color:#f0f0f0;}
.tg .tg-nleg{font-size:14px;font-family:"Lucida Console", Monaco, monospace !important;;text-
align:center;vertical-align:top}
</style>
```

```
<?php

$url = 'ballerindex.png';

?>

<?php
session_start();
require_once("connect.php");
$uid=$_GET['id'];

?>


<input type="button" value="Player Details"
onclick="window.location.href='http://localhost/BallerIndex/SPlayerDetails.php?id=<?php
echo $uid ?>'" style="width:150px; height:70px;" />
<input type="button" value="Basketball PlayerMarket"
onclick="window.location.href='http://localhost/BallerIndex/SPlayerMarket.php?id=<?php
echo $uid ?>'" style="width:150px; height:70px;"/>
<input type="button" value="Portfolio"
onclick="window.location.href='http://localhost/BallerIndex/ViewPortfolio.php?id=<?php echo
$uid ?>'" style="width:150px; height:70px;"/>
<input type="button" value="Watchlist"
onclick="window.location.href='http://localhost/BallerIndex/ViewWatchlist.php?id=<?php
echo $uid ?>'" style="width:150px; height:70px;"/>
<input type="button" value="Cart"
onclick="window.location.href='http://localhost/BallerIndex/CartView.php?id=<?php echo
$uid ?>'" style="width:150px; height:70px;"/>
<input type="button" value="Log Out"
onclick="window.location.href='http://localhost/BallerIndex/Login.html?id=<?php echo $uid
?>'" style="width:150px; height:70px;"/>
<style type="text/css">
body
{
        background-image:url('<?php echo $url ?>');
    height: 100%;
    background-position: center;
    background-repeat: repeat;
    background-size: cover;
```

```
}

</style>

<table width=50% class="tg" align='center'>
<tr>
<th class="tg-nleg"> PlayerID</th>
<th>Player Name </th>
<th> Buy Rate</th>
<th> Sell Rate</th>

<?php

$sql = "SELECT  PlayerMarket.PlayerID,PlayerName,`Buy Rate`,`Sell rate` FROM
PlayerMarket INNER JOIN Price ON PlayerMarket.PlayerID=Price.PriceID where
PlayerMarket.GameTypeID=1";
$result= mysqli_query($conn,$sql);
if( mysqli_num_rows($result) > 0)
{

while($row = mysqli_fetch_array($result) )
```

```
{
        echo"<tr>";
        $pid=$row['PlayerID'];
        echo "<td>".$row['PlayerID']."</td>";
        echo "<td>".$row['PlayerName']."</td>";
        echo "<td>".$row['Buy Rate']."</td>";
        echo "<td>".$row['Sell rate']."</td>";?>
       <td> <input type="button" value="Add to Cart"
onclick="window.location.href='http://localhost/BallerIndex/Shares.php?id=<?php echo $uid
?>&pid=<?php echo $pid ?>'"/></td>
       <td> <input type="button" value="Add to Watchlist"
onclick="window.location.href='http://localhost/BallerIndex/AddWAtchlist.php?id=<?php echo
$uid ?>&pid=<?php echo $pid ?>'"/></td>
  <?php echo "</tr>";

 }
 }
else

 {
        echo "0 rows";
```

```
 }
if (!$result) {
        die ('SQL Error: ' . mysqli_error($conn));
}
?>
</table>
 </body>
</html>
```

10.Php code to view PlayerDetails.

```
<html>
<h1 align='center'> BALLER INDEX </h1>
<body>
<?php

$url = 'ballerindex.png';

?>

<style type="text/css">
.tg  {border-collapse:collapse;border-spacing:0;border-color:#ccc;}
.tg td{font-family:Arial, sans-serif;font-size:14px;padding:10px 5px;border-style:solid;border-width:1px;overflow:hidden;word-break:normal;border-color:#ccc;color:#333;background-color:#fff;}
.tg th{font-family:Arial, sans-serif;font-size:14px;font-weight:normal;padding:10px 5px;border-style:solid;border-width:1px;overflow:hidden;word-break:normal;border-color:#ccc;color:#333;background-color:#f0f0f0;}
.tg .tg-nleg{font-size:14px;font-family:"Lucida Console", Monaco, monospace !important;;text-align:center;vertical-align:top}
</style>

<style type="text/css">
body
{
        background-image:url('<?php echo $url ?>');
   height: 100%;
   background-position: center;
   background-repeat: repeat;
   background-size: cover;
}
```

```
</style>


<?php
require_once("connect.php");
$uid=$_GET['id'];
?>
<input type="button" value="Back"
onclick="window.location.href='http://localhost/BallerIndex/Playermarket.php?id=<?php echo
$uid ?>'" style="width:150px; height:70px;"/>
<table width=50% class="tg" align="center">
<tr>
<th class="tg-nleg"> PlayerId </th>
<th> Player Name </th>
<th> Hieght </th>
<th> Weight </th>
<th> Age </th>
</tr>

<?php
session_start();
```

```
$sql="SELECT PlayerID,PlayerName,Hieght ,Weight ,Age FROM  PlayerMarket Where
PlayerMarket.GameTypeID = 1";
$result = mysqli_query($conn,$sql);
if( mysqli_num_rows($result) > 0)
{
	while($row = mysqli_fetch_array($result))

	{

		echo "<tr>";
		echo"<td>".$row['PlayerID']."</td>";
		echo"<td>".$row['PlayerName']."</td>";
		echo"<td>".$row['Hieght']."</td>";
		echo"<td>".$row['Weight']."</td>";
		echo"<td>".$row['Age']."</td>";
		echo"</tr>";
	}
}
	else
	{
		echo "0rows";
```

```
        }
?>
</table>
</body>
 </html>
```

11.Php code Add to portfolio

```
<html><h1 align='center' color='white'> BALLER INDEX </h1>
<style type="text/css">
.tg  {border-collapse:collapse;border-spacing:0;border-color:#ccc;}
.tg td{font-family:Arial, sans-serif;font-size:14px;padding:10px 5px;border-style:solid;border-
width:1px;overflow:hidden;word-break:normal;border-color:#ccc;color:#333;background-
color:#fff;}


.tg .tg-nleg{font-size:14px;font-family:"Lucida Console", Monaco, monospace !important;;text-
align:center;vertical-align:top}
</style>
<body align='center'>
```



```
<?php
require_once("connect.php");
session_start();
$uid=$_GET['id'];
$pid=$_GET['pid'];
$shares=$_GET['shares'];
$sql="call SPupdateorder($uid)";
$sql="call spAddtoPortfolio($uid,$pid,$shares)";
$result=mysqli_query($conn,$sql);
header("location:ViewPortfolio.php?id=".$uid);
?>
```

12.Php code to View Portfolio

```
<html>
<h1 align='center' color='white'> BALLER INDEX </h1>
<style type="text/css">
.tg  {border-collapse:collapse;border-spacing:0;border-color:#ccc;}
.tg td{font-family:Arial, sans-serif;font-size:14px;padding:10px 5px;border-style:solid;border-
width:1px;overflow:hidden;word-break:normal;border-color:#ccc;color:#333;background-
color:#fff;}
```

```
.tg th{font-family:Arial, sans-serif;font-size:14px;font-weight:normal;padding:10px 5px;border-style:solid;border-width:1px;overflow:hidden;word-break:normal;border-color:#ccc;color:#333;background-color:#f0f0f0;}
.tg .tg-nleg{font-size:14px;font-family:"Lucida Console", Monaco, monospace !important;;text-align:center;vertical-align:top}
</style>
<body align='center'>
<?php
require_once("connect.php");
session_start();
$uid=$_GET['id'];
$sql="call spViewPortfolio($uid)";
$result = mysqli_query($conn,$sql); ?>
<input type="button" value="Basketball PlayerMarket"
onclick="window.location.href='http://localhost/BallerIndex/PlayerMarket.php?id=<?php echo $uid ?>'" style="width:200px; height:70px;"/>
<input type="button" value="Soccer PlayerMarket"
onclick="window.location.href='http://localhost/BallerIndex/SPlayerMarket.php?id=<?php echo $uid ?>'" style="width:200px; height:70px;"/>
<input type="button" value="Portfolio"
onclick="window.location.href='http://localhost/BallerIndex/ViewPortfolio.php?id=<?php echo $uid ?>'" style="width:200px; height:70px;"/>
<input type="button" value="Watchlist"
onclick="window.location.href='http://localhost/BallerIndex/ViewWatchlist.php?id=<?php echo $uid ?>'" style="width:200px; height:70px;"/>
<input type="button" value="Cart"
onclick="window.location.href='http://localhost/BallerIndex/CartView.php?id=<?php echo $uid ?>'" style="width:200px; height:70px;"/>
<input type="button" value="Log Out"
onclick="window.location.href='http://localhost/BallerIndex/Login.html?id=<?php echo $uid ?>'" style="width:200px; height:70px;"/>
<table width="50%" class="tg" align="center">
<tr>
<td class="tg-nleg"> PlayerID </td>
<td> Player Name </td>
<td>sell Rate </td>
<td> No of shares</td>
<td> TotalStockPrice</td>
</tr>
<?php

if( mysqli_num_rows($result) > 0)
```

```
        {
 while($row = mysqli_fetch_array($result))

        {
                echo "<tr>";
                $pid=$row['PlayerID'];
                echo"<td>".$row['PlayerID']."</td>";
                echo"<td>".$row['Playername']."</td>";
                echo"<td>".$row['Sell rate']."</td>";
                echo"<td>".$row['No of shares']."</td>";
                echo"<td>".$row['TotalStockPrice']."</td>";?>
                <td> <input type="button" value="Sell"
onclick="window.location.href='http://localhost/BallerIndex/SellShare.php?id=<?php echo $uid
?>&pid=<?php echo $pid  ?>'"/></td>
                <?php echo"</tr>";
        }
}
        else
        {
                echo "0rows";
        }
        ?>
        </body>
        </html>
```


13.Php code to Sell Shares

```
<?php
require_once("connect.php");
session_start();
$uid=$_GET['id'];
$pid=$_GET['pid'];
$Shares=$_POST['Shares'];
$sql= "call SpSelling($uid,$pid,$Shares)";
$result= mysqli_query($conn,$sql);
header("location:ViewPortfolio.php?id=". $uid);
?>
```

14. Php code to give the no of shares to sell

```
<html>
<body>
```

```
<?phprequire_once("connect.php");
session_start();
$uid = $_GET['id'];
$pid = $_GET['pid'];
 ?>
<form method="post" action="Sell.php?id=<?php echo $uid ?>&pid=<?php echo $pid ?>">
<table width="400" border="0" cellpadding="10" cellspacing="10">
<tr>
<td style="font-weight: bold"><div align="right"><label for="Shares">No of
Shares</label></div></td>
<td><input name="Shares" type="integer" class="input" size="25" required /></td>
</tr>
<tr>
<td height="23"></td>
<td><div align="right">
  <input type="submit" name="submit" value="Next" />
</div></td>
</tr>
</form>
</body>
</html>
```


15. Php code to Add to watchlist

```
<?php
require_once("connect.php");
session_start();
$uid=$_GET['id'];
$pid=$_GET['pid'];
$sql="call spAddWatchlist($pid,'add',$uid)";
$result=mysqli_query($conn,$sql);
header("location:ViewWatchlist.php?id=".$uid);
?>
```

16. Php code to Delete from watchlist

```
<?php
require_once("connect.php");
session_start();
$uid=$_GET['id'];
$pid=$_GET['pid'];
$sql="call spAddWatchlist($pid,'remove',$uid)";
```

```
$result=mysqli_query($conn,$sql);
header("location:ViewWatchlist.php?id=".$uid);
?>
```

17. Php code to view Watchlist

```
<?php
require_once("connect.php");
session_start();
$uid=$_GET['id'];
$sql="call spViewWatchlist($uid)";
$result=mysqli_query($conn,$sql);?>
<html>
<body align='center'>
<input type="button" value="Basketball PlayerMarket"
onclick="window.location.href='http://localhost/BallerIndex/PlayerMarket.php?id=<?php echo
$uid ?>'" style="width:200px; height:70px;"/>
<input type="button" value="Soccer PlayerMarket"
onclick="window.location.href='http://localhost/BallerIndex/SPlayerMarket.php?id=<?php
echo $uid ?>'" style="width:200px; height:70px;"/>
<input type="button" value="Portfolio"
onclick="window.location.href='http://localhost/BallerIndex/ViewPortfolio.php?id=<?php echo
$uid ?>'" style="width:200px; height:70px;"/>
<input type="button" value="Cart"
onclick="window.location.href='http://localhost/BallerIndex/CartView.php?id=<?php echo
$uid ?>'" style="width:200px; height:70px;"/>
<input type="button" value="Log Out"
onclick="window.location.href='http://localhost/BallerIndex/Login.html?id=<?php echo $uid
?>'" style="width:200px; height:70px;"/>
<style type="text/css">
.tg  {border-collapse:collapse;border-spacing:0;border-color:#ccc;}
.tg td{font-family:Arial, sans-serif;font-size:14px;padding:10px 5px;border-style:solid;border-
width:1px;overflow:hidden;word-break:normal;border-color:#ccc;color:#333;background-
color:#fff;}
.tg th{font-family:Arial, sans-serif;font-size:14px;font-weight:normal;padding:10px
5px;border-style:solid;border-width:1px;overflow:hidden;word-break:normal;border-
color:#ccc;color:#333;background-color:#f0f0f0;}
.tg .tg-nleg{font-size:14px;font-family:"Lucida Console", Monaco, monospace !important;;text-
align:center;vertical-align:top}
</style>
<table width="50%" class="tg" align="center">
<tr>
<td class="tg-nleg"> PlayerID </td>
```

```
<td> Player Name </td>
<td> Buy Rate </td>
<td> Sell Rate</td>
<tr>
<?php
if( mysqli_num_rows($result) > 0)
{
  while($row = mysqli_fetch_array($result))
        {

                echo "<tr>";
                $pid=$row['PlayerID'];
                echo"<td>".$row['PlayerID']."</td>";
                echo"<td>".$row['PlayerName']."</td>";
                echo"<td>".$row['Currentprice']."</td>";
                echo"<td>".$row['Sell rate']."</td>";?>
                <td> <input type="button" value="Add to Cart"
onclick="window.location.href='http://localhost/BallerIndex/Shares.php?id=<?php echo $uid
?>&pid=<?php echo $pid ?>'"/></td>
                <td> <input type="button" value="Delete"
onclick="window.location.href='http://localhost/BallerIndex/DeleteWatchlist.php?id=<?php
echo $uid ?>&pid=<?php echo $pid  ?>'" /></td>
                <?php
                echo"</tr>";

        }
}
        else
        {
                echo "0rows";
        }
?>
</table>
<input type="button" value="Back"
onclick="window.location.href='http://localhost/BallerIndex/Playermarket.php?id=<?php echo
$uid ?>'" style="width:100px; height:40px;"/>
</body>
</html>
```


18. php code to add to cart

```
<html>
```

```
<h1 align='center' color='white'> BALLER INDEX </h1>
<body align='center'>
<style type="text/css">
.tg  {border-collapse:collapse;border-spacing:0;border-color:#ccc;}
.tg td{font-family:Arial, sans-serif;font-size:14px;padding:10px 5px;border-style:solid;border-
width:1px;overflow:hidden;word-break:normal;border-color:#ccc;color:#333;background-
color:#fff;}
.tg th{font-family:Arial, sans-serif;font-size:14px;font-weight:normal;padding:10px
5px;border-style:solid;border-width:1px;overflow:hidden;word-break:normal;border-
color:#ccc;color:#333;background-color:#f0f0f0;}
.tg .tg-nleg{font-size:14px;font-family:"Lucida Console", Monaco, monospace !important;;text-
align:center;vertical-align:top}
</style>
<?php

require_once("connect.php");
session_start();
$uid=$_GET['id'];
$pid=$_GET['pid'];
$Shares=$_POST['Shares'];
$sql="call spAddtocart($uid,$pid,$Shares)";
$result=mysqli_query($conn,$sql);
$sql1="call spviewcart($uid)";
$view = mysqli_query($conn,$sql1);

?>
<input type="button" value="Basketball PlayerMarket"
onclick="window.location.href='http://localhost/BallerIndex/PlayerMarket.php?id=<?php echo
$uid ?>'" style="width:200px; height:70px;"/>
<input type="button" value="Soccer PlayerMarket"
onclick="window.location.href='http://localhost/BallerIndex/SPlayerMarket.php?id=<?php
echo $uid ?>'" style="width:200px; height:70px;"/>
<input type="button" value="Portfolio"
onclick="window.location.href='http://localhost/BallerIndex/ViewPortfolio.php?id=<?php echo
$uid ?>'" style="width:200px; height:70px;"/>
<input type="button" value="Cart"
onclick="window.location.href='http://localhost/BallerIndex/CartView.php?id=<?php echo
$uid ?>'" style="width:200px; height:70px;"/>
<input type="button" value="Log Out"
onclick="window.location.href='http://localhost/BallerIndex/Login.html?id=<?php echo $uid
?>'" style="width:200px; height:70px;"/>
<table width=50% class="tg" align='center'>
```

```
<tr>
<td class="tg-nleg"> PlayerID </td>
<td> PlayerName </td>
<td> Buy Rate </td>
<td> No of Shares</td>
<td> Amount </td>
<tr>
<?php
if( mysqli_num_rows($view) > 0)
{

	while($row = mysqli_fetch_array($view))

	{

		echo "<tr>";
		$pid=$row['PlayerID'];
		echo"<td>".$row['PlayerID']."</td>";
		echo"<td>".$row['Playername']."</td>";
		echo"<td>".$row['Buy Rate']."</td>";
		$shares=$row['noofshares'];
		echo"<td>".$row['noofshares']."</td>";
		echo"<td>".$row['Amount']."</td>";?>
		<td> <input type="button" value="Delete"
onclick="window.location.href='http://localhost/BallerIndex/DeleteCart.php?id=<?php echo
$uid ?>&pid=<?php echo $pid  ?>&shares=<?php echo $shares ?>'"/></td>
		<td><input type="button" value="Checkout "
onclick="window.location.href='http://localhost/BallerIndex/Portfolio.php?id=<?php echo $uid
?>&pid=<?php echo $pid ?>&shares=<?php echo $shares ?>'"/></td>

		<?php
		echo"</tr>";

	}
}
	else
	{
		echo "0rows";
	}
?>
</body>
</html>
```

19. php code to give no of shares to add to to cart

```
<html>
<body>
<?php
require_once("connect.php");
session_start();
$uid = $_GET['id'];
$pid = $_GET['pid'];
 ?>
<form method="post" action="Cart.php?id=<?php echo $uid ?>&pid=<?php echo $pid ?>">
<table width="400" border="0" cellpadding="10" cellspacing="10">
<tr>
<td style="font-weight: bold"><div align="right"><label for="Shares">No of
Shares</label></div></td>
<td><input name="Shares" type="integer" class="input" size="25" required /></td>
</tr>
<tr>
<td height="23"></td>
<td><div align="right">
  <input type="submit" name="submit" value="Next"  />
</div></td>
</tr>
</form>
</body>
</html>
```


20. php code to delete from cart

```
<?php
require_once("Connect.php");
session_start();
$uid=$_GET['id'];
$pid=$_GET['pid'];
$sql="DELETE FROM CartItems Where CartItems.UserId=$uid AND
CartItems.PlayerID=$pid";
$result= mysqli_query($conn,$sql);
header("location:CartView.php?id=".$uid);
?>
```

21. php code to view Cart

```
<?php
require_once("Connect.php");
session_start();
$uid=$_GET['id'];
$sql1="call spviewcart($uid)";
$view = mysqli_query($conn,$sql1);
?>
<html>
<body align='center'>
<style type="text/css">
.tg  {border-collapse:collapse;border-spacing:0;border-color:#ccc;}
.tg td{font-family:Arial, sans-serif;font-size:14px;padding:10px 5px;border-style:solid;border-
width:1px;overflow:hidden;word-break:normal;border-color:#ccc;color:#333;background-
color:#fff;}
.tg th{font-family:Arial, sans-serif;font-size:14px;font-weight:normal;padding:10px
5px;border-style:solid;border-width:1px;overflow:hidden;word-break:normal;border-
color:#ccc;color:#333;background-color:#f0f0f0;}
.tg .tg-nleg{font-size:14px;font-family:"Lucida Console", Monaco, monospace !important;;text-
align:center;vertical-align:top}
</style>
<input type="button" value="Basketball PlayerMarket"
onclick="window.location.href='http://localhost/BallerIndex/PlayerMarket.php?id=<?php echo
$uid ?>'" style="width:200px; height:70px;"/>
<input type="button" value="Soccer PlayerMarket"
onclick="window.location.href='http://localhost/BallerIndex/SPlayerMarket.php?id=<?php
echo $uid ?>'" style="width:200px; height:70px;"/>
<input type="button" value="Portfolio"
onclick="window.location.href='http://localhost/BallerIndex/ViewPortfolio.php?id=<?php echo
$uid ?>'" style="width:200px; height:70px;"/>
<input type="button" value="Watchlist"
onclick="window.location.href='http://localhost/BallerIndex/ViewWatchlist.php?id=<?php
echo $uid ?>'" style="width:200px; height:70px;"/>
<input type="button" value="Log Out"
onclick="window.location.href='http://localhost/BallerIndex/Login.html?id=<?php echo $uid
?>'" style="width:200px; height:70px;"/>
<table width="50%" class="tg" align="center">
<tr>
<td class="tg-nleg"> PlayerID </td>
<td> PlayerName </td>
<td> Buy Rate </td>
<td> No of Shares</td>
<td> Amount </td>
```

```
<tr>
<?php
if( mysqli_num_rows($view) > 0)

{
while($row = mysqli_fetch_array($view))
                {
                echo "<tr>";
                $pid=$row['PlayerID'];
                echo"<td>".$row['PlayerID']."</td>";
                echo"<td>".$row['Playername']."</td>";
                echo"<td>".$row['Buy Rate']."</td>";
                $shares=$row['noofshares'];
                echo"<td>".$row['noofshares']."</td>";
                echo"<td>".$row['Amount']."</td>";?>
                <td> <input type="button" value="Delete"
onclick="window.location.href='http://localhost/BallerIndex/DeleteCart.php?id=<?php echo
$uid ?>&pid=<?php echo $pid  ?>'" /></td>
                <td><input type="button" value="Checkout "
onclick="window.location.href='http://localhost/BallerIndex/Portfolio.php?id=<?php echo $uid
?>&pid=<?php echo $pid ?>&shares=<?php echo $shares ?>'"/></td>
                <?php
                echo"</tr>";

        }?>

<?}
        else
        {
                echo "0rows";?>
<?php }
?>
</table>
<input type="button" value="Back"
onclick="window.location.href='http://localhost/BallerIndex/HomePage.php?id=<?php echo
$uid ?>'" style="width:100px; height:40px;"/>

</html>
```

REFERENCES

- http://www.nba.com/players/stephen/curry/201939
- https://www.basketball-reference.com/
- http://www.rotoworld.com/sports/nba/basketball?ls=roto:nba:gnav
- www.stackoverflow.com
- www.mysqltutorial.org
- https://www.phpmyadmin.net
- www.dba.stackexchange.com

- www.webdeveloper.com
- www.mysql.com
- www.statista.com
- www.mlssoccer.com