USE CASE PROBLEM DEFINITION:

Group: 14
Name: Binghui Lai, Mohith Kota

A) Business Problem and Requirement:

Now-a-days Video games are going digital. Most of the store platforms, whether online or offline, have their own databases to record all the Sales data. Some of them can be accessed by the public, like Steam DB, but all the data on different stores are not linked, which could bring a lot of trouble for Business Investigation and Analytics. Boston Consulting group decided to build a database to store the Sales data, as a consulting tool and open to their customer.

The database is solving practical problems in the gaming industry. For the publisher, it will help them to decide the publishing region while planning for a potential profit game. With this database they will get more data support rather than only relying on experience. When conducting competitive analysis, by searching for region, revenue, cooperated developers and other data of games published by their competitors they will stand a greater chance in the fierce competition.

The database is also an ideal tool for developers. They can find more business opportunities from different regions by catering specific needs of customers. By analyzing customers, they will have a clearer direction in the development.

In this database, the following relationship should be followed while implementing:

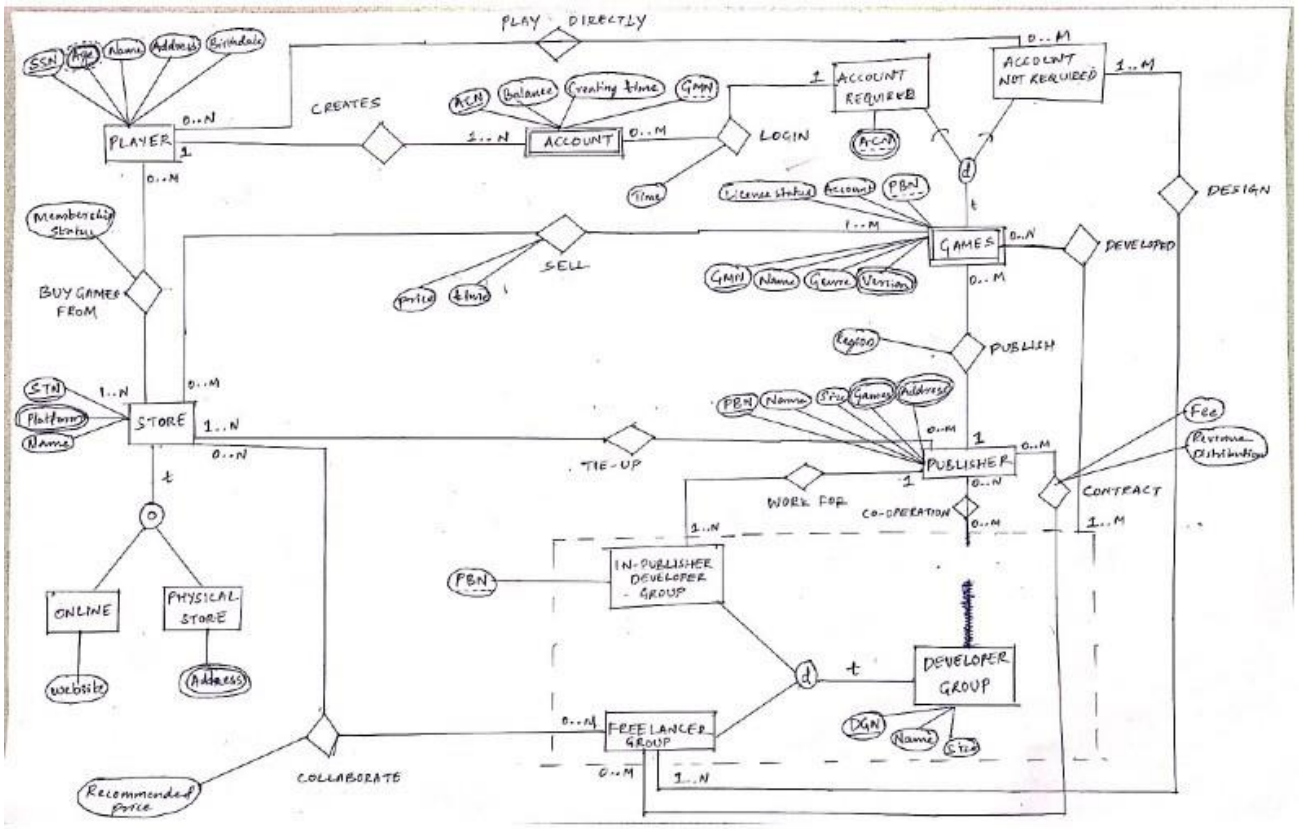
1. Developer group focuses on developing games. They can develop multiple games and a game can be developed by multiple developer groups. And in most circumstances they will have a cooperation relationship with the publisher. Developer group can cooperate with multiple publishers, vice versa. For each developer group, they have a unique group number. The size and name of the developer group are also required for recording.
2. We can divide Developer groups into two types: In-Publisher Developer group and Freelancers. In-Publisher Developer Groups work for specific publishers. For freelancers, they may have contracts with one or multiple publishers, publishing different games developed by them. A developer group can't be freelancers and an In-Publisher Developer group at the same time. Freelancers can negotiate with stores directly, once the store allows, they can sell games directly on the store. In this process free-lancers should give a recommended price for the game.
3. Publishers publish games. They can publish multiple games and have them running at the same time. And publisher will tie up with at least 1 store as a platform to sell their games. For publisher we need to store their unique number (PBN), name, size, address and games they have published. A publisher can have multiple addresses and multiple published game.
4. Store can be divided into online store and offline store. Some stores are online and offline at the same time, like GameStop. A store can cooperate with multiple publishers, and sometimes can be 0. Store sells games to players. At different times, a game might be on discount. Games can have different prices in different stores. Players buy games from the store. They can buy from multiple stores while stores can of course have multiple customers. We should store STN as a unique number for the store, as well as their name. Besides, the

platform they are on should be required. A store can have multiple platforms. (e.g. PlayStation Store is only available on PlayStation while Xbox Store is available on PC and XBOX.). For offline stores we need to store their addresses, which could be more than one. The website of an online store is required too.

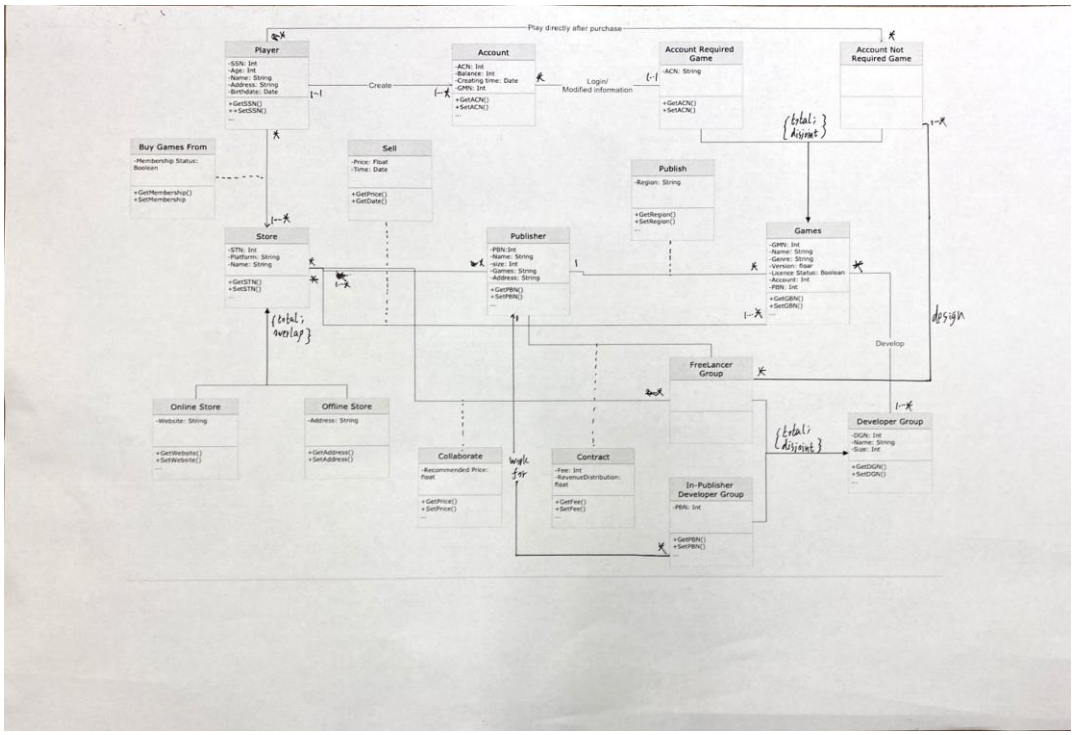
5. Games can be divided into 2 types: Account Required Game and Account Not Required Game. Players must register an account to access the Account Required Game they buy. They can create accounts with the same name in different games. For account the storage of Account number, Balance, Creating time and Game number are required. An account belongs to a specific game. For Account Not Required Game, players can directly enjoy it after purchasing. However, for Free- Lancers, they can only develop Account Not Required Game due to lack of publishers. We need tables to store the unique game number, name, genre, version, and license status and publisher number.
6. Players can buy games from various stores. To do further analysis, we need some information, including SSN, Age, Name, address and birthdate.

B) Conceptual Data Modeling

EER MODEL



UML MODEL



C) Mapping Conceptual Model to Relational Model

PLAYER(SSN, Name, Birthdate, Address, Age)

ACCOUNT(ACN, GMN, SSN, Balance, Creating Time, Login Time)

- GMN foreign key refers to GMN in GAMES, NOT NULL
- SSN Foreign key refers to SSN in PLAYER, NOT NULL

ACCOUNT REQUIRED(AR.GMN, …)

- AR.GMN Foreign key refers to GMN in GAMES, NOT NULL

ACCOUNT NOT REQUIRED(ANR.GMN, …)

- ANR.GMN Foreign key refers to GMN in GAMES, NOT NULL

PLAY DIRECTLY(SSN, GMN)

- SSN Foreign key refers to SSN in PLAYER, NOT NULL
- GMN foreign key refers to GMN in GAMES, NOT NULL

STORE(STN, Name, Platform)

ONLINE STORE(Online.STN, Website, …)

- Online.STN Foreign key refers to STN in STORE, NOT NULL

PHYSICAL STORE(Physical.STN, Address, …)

- Physical.STN Foreign key refers to STN in STORE, NOT NULL

BUY GAMES FROM(SSN, STN, Membership Status)

- STN Foreign key refers to STN in STORE, NOT NULL
- SSN Foreign key refers to SSN in PLAYER, NOT NULL

SELL(GMN, STN, Price, Time)

- STN Foreign key refers to STN in STORE, NOT NULL
- GMN foreign key refers to GMN in GAMES, NOT NULL

TIE-UP(PBN, STN)
- STN Foreign key refers to STN in STORE, NOT NULL
- PBN Foreign key refers to PBN in PUBLISHER, NOT NULL

COLLABORATE(STN, FLG.DGN, Recommended Price)
- STN Foreign key refers to STN in STORE, NOT NULL
- FLG.DGN Foreign key refers to DGN in DEVELOPER GROUP, NOT NULL

GAMES(GMN, Name, Genre, Version, License Status, Account, Region, PBN)
- PBN Foreign key refers to PBN in PUBLISHER, NOT NULL

PUBLISHER(PBN, Name, Size, Games,
Address) DEVELOPER GROUP(DGN,
Name, Size)
IN-PUBLISHER DEVELOPER GROUP(IP.DGN, PBN, …)
- PBN Foreign key refers to PBN in PUBLISHER, NOT NULL

FREELANCER GROUP(FLG.DGN,
…) CO-OPERATION(PBN,
DGN)
- PBN Foreign key refers to PBN in PUBLISHER, NOT NULL
- DGN Foreign key refers to DGN in DEVELOPER GROUP, NOT NULL

CONTRACT(PBN, FLG.DGN, Fee, Revenue Distribution)
- PBN Foreign key refers to PBN in PUBLISHER, NOT NULL
- FLG.DGN Foreign key refers to DGN in DEVELOPER GROUP, NOT NULL

DEVELOPED(GMN, DGN)
- GMN foreignkey refers to GMN in GAMES, NOT NULL
- DGN Foreign key refers to DGN in DEVELOPER GROUP, NOT NULL

DESIGN(FLG.DGN, ANR.GMN)
- ANR.GMN foreign key refers to GMN in GAMES, NOT NULL
- FLG.DGN Foreign key refers to DGN in DEVELOPER GROUP, NOT

NULL DESIGNER GROUP(DGN)

D) Implementation of Relation Model via MySQL & NoSQL

MySQL Implementation:

Query-1: Searching for the player who have visited all the stores in record.

```
select p.name
from buy_games_from bgf inner join player p
on bgf.ssn = p.ssn
```

```
where bgf.MEMBER_SHIP_STATUS = 'TRUE'
group by p.name
having count(distinct bgf.stn) =(select count(distinct stn) from store);
```

name
Mohith
Rohith

Query-2: Searching for all the Developer groups working with Ubisoft.

```
select dg.name, p.name
from co_operation co inner join publisher p on co.pbn = p.pbn
inner join developer_group dg on dg.dgn = co.dgn
where p.name = 'Ubisoft';
```

name	name
Rockstar	Ubisoft
Nexus	Ubisoft
Rebel	Ubisoft

Query-3: Calculating each player's average balance of their accounts with the value other than Zero.

```
select p.name, avg(a.balance) average
from Account a inner join Player p on a.ssn = p.ssn
group by p.ssn
having average!=0;
```

name	average
Mohith	476.5000
TOM	20.0000
TIMMY	392.0000
Kota	59638.0000
Katkam	12.0000
Uppula	7.5000
Dheekonda	8792.0000
Rohith	38.0000

Query-4: Finding the most popular online_store

```
with cte as (
        select count(ssn) num
   from buy_games_from
   group by stn)

select name as 'most popular stores'
from online_store os inner join buy_games_from bgf on os.online_stn = bgf.stn
group by os.online_stn
having count(bgf.ssn) = (select max(num) from cte);
```

most popular stores
playstation store
nintendo store

Query-5: Find all the games published in Taiwan or Hongkong, and their Publisher.

```
select pr.pbn,pr.name publisher,g.name game_name, p.region
from publisher pr inner join publish p
on pr.pbn = p.pbn
inner join games g
on p.gmn = g.gmn
where region in ('HongKong','Taiwan')
order by publisher;
```

pbn	publisher	game_name	region
3	MIcrosoft	FORTNITE	TAIWAN
3	MIcrosoft	Witcher	TAIWAN
1	SONY_INTERACTIVE	FIFA	HongKong
1	SONY_INTERACTIVE	CALL_OF_DUTY	TAIWAN
1	SONY_INTERACTIVE	Divinity	HongKong
2	Ubisoft	APEX	TAIWAN
2	Ubisoft	FINAL_FANTASY	TAIWAN

Query-6: find all the games developed by more than 15 developers.

```
select g.name,  dg.size developer_num
from developer_group dg inner join developed d on dg.dgn = d.dgn
inner join Games g on d.gmn = g.gmn
where dg.size > 15
order by dg.size desc;
```

name	developer_num
FIFA	70
APEX	35
FORTNITE	35
IT_TAKES_TWO	30
FINAL_FANTASY	30
HALO:INFINITE	20

Query-7: Search for the accounts numbers of players born after 1996.

```
select cte1.name, count(acn) having_accounts
from account a inner join
        (select * from player
        where year(Birthday) > 1996) cte1 on a.ssn = cte1.ssn
group by cte1.ssn;
```

	Name	having_accou...
►	Mohith	2
	TOM	1
	TIMMY	1
	Kota	1
	Katkam	1
	Dheekonda	1
	Rohith	1

NoSQL Implementation:

Query-1: Fetch the complete details of the Players and Store who bought the games from 'PLAYSTATION' Platform

```
MATCH (p:PLAYER)-[:TO]-(b:BGF)-[:FROM]-(s:STORE)
WHERE s.Platform='PLAYSTATION'
RETURN p.Name AS Player_Name, p.SSN AS Player_SSN, p.Birthdate AS Player_BirthDate,
p.Address AS Player_Address, p.Age AS Player_Age, s.Name AS Store_Name, s.STN AS
Store_Number, s.Platform AS Store_Platform
```

Query-2: Find the top three oldest players in the order whose are older than 20 years.

```
MATCH (n:PLAYER)
WHERE toInteger(n.Age) > 20
RETURN n.Name AS Player_name, n.SSN AS Player_SSN, n.Birthdate AS Player_Birthdate, n.Address
AS Player_Adress, n.Age AS Player_Age
Order By n.Age DESC
Limit 3
```

Query-3: Find the Name, Age of the Player and Name, Genre, Version, Price and Platform of the game bought by the player from "PLAYSTATION_STOP" store.

```
MATCH (p:PLAYER)-[:TO]-(b:BGF)-[:FROM]-(s:STORE)-[:SELLS]-(sl:SELL)-[:SELLS_GAMES]-
```

(g:GAMES)
WHERE s.Name='PLAYSTATION_STOP'
RETURN p.Name AS Player_Name,p.Age AS Player_Age,g.Name AS Game_Name, s.Platform AS Game_Platform, g.Genre AS Game_Genre, g.Version AS Game_Version, sl.Price AS Game_Price

```
project$ MATCH (p:PLAYER)-[:TO]-(b:BGF)-[:FROM]-(s:STORE)-[:SELLS]-(sl:SELL)-
         [:SELLS_GAMES]-(g:GAMES)
         WHERE s.Name='PLAYSTATION_STOP'
         RETURN p.Name AS Player_Name,p.Age AS Player_Age,g.Name AS Game_Name,
         s.Platform AS Game_Platform, g.Genre AS Game_Genre, g.Version AS
         Game_Version, sl.Price AS Game_Price
```

Table | Text

	Player_Name	Player_Age	Game_Name	Game_Platform	Game_Genre	Game_Version	Game_Price
1	"Katkam"	"22"	"FIFA"	"PLAYSTATION"	"SPORTS"	"1"	"60"

E) Database Access via Python

Connecting Mysql and Python

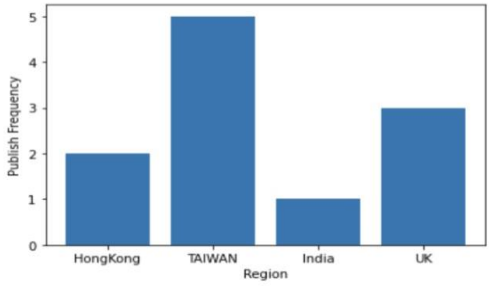
In [128]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [129]:
```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="",
   database= "6700_project"
)
```

GRAPH1: THe number of games published in specific region

In [131]:
```
mycursor = mydb.cursor()
mycursor.execute(
   "select  p.region, count(*) from publisher pr inner join publish p on pr.pbn = p.pbn inne
myresult = mycursor.fetchall()
df = pd.DataFrame(myresult, columns = ['Region','Number'])
fig,ax = plt.subplots()

bar_graph = plt.bar(df['Region'],height = df['Number'])
ax.set_xlabel('Region')
ax.set_ylabel('Publish Frequency')
```

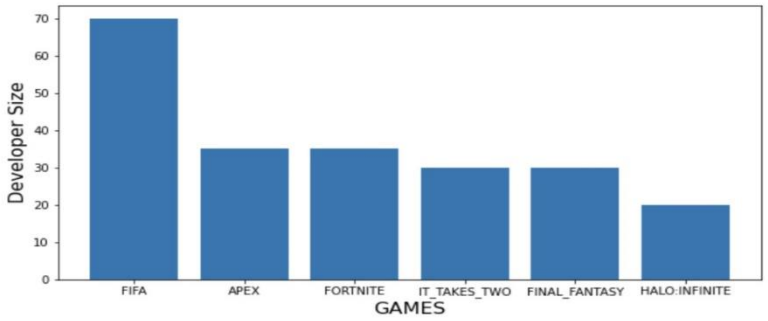
Out[131]: Text(0, 0.5, 'Publish Frequency')



GAMES DEVELOPED BY OVER 15 PEOPLE AND THEIR DEVELOPER GROUP SIZE

```
In [132]: mycursor2 = mydb.cursor()
mycursor2.execute(
   "select g.name, dg.name, dg.size from developer_group dg inner join developed d on dg.dgn
myresult2 = mycursor2.fetchall()
df2 = pd.DataFrame(myresult2, columns = ['Games','Developer','size'])

fig = plt.figure(figsize=(10,5))

bar_graph = plt.bar(df2['Games'],height = df2['size'])
plt.xlabel('GAMES', fontsize = 16)
plt.ylabel('Developer Size', fontsize = 16)
```

```
Out[132]: Text(0, 0.5, 'Developer Size')
```



F) Summary and Recommendation.

The Gaming Development and Publish database designed on MySQL targeting at providing a tool to store, organize, modify gaming publish data. With this database it will be easier to manage data including input and delete operation, searching data for analysis of the gaming industry. We have some sample query in the report as well as a link to python for usage under multiple environments.

While the database is ready for implementation, there are still some improvements that should be considered in the future. Firstly, the database doesn't have any governance measure to ensure the data quality, because of which after a period using the database may not be in a well-organized status. For suggestions, the function built inside of database software or outer programming language like python should contribute to this improvement.

The NoSQL database is also available. But we also have some concerns about it. Firstly, NoSQL is not used as widely as MySQL world-wide. NoSQL hasn't got a formal query language so it may bring difficulties to users of this database to get familiar with that before implementing in real cases. The development kit is not as well-developed as MySQL either. However, in this case the database is not required to keep all-time consistency, periodically modified should be enough. NoSQL can provide a higher efficiency for users in this case.