COMPX323 – Project

- Kevin Han 1521885
- Bedir Asici
- Tetsusaburo Kato

Table of Contents

1. Project Milestone 1

1.1 Application Description

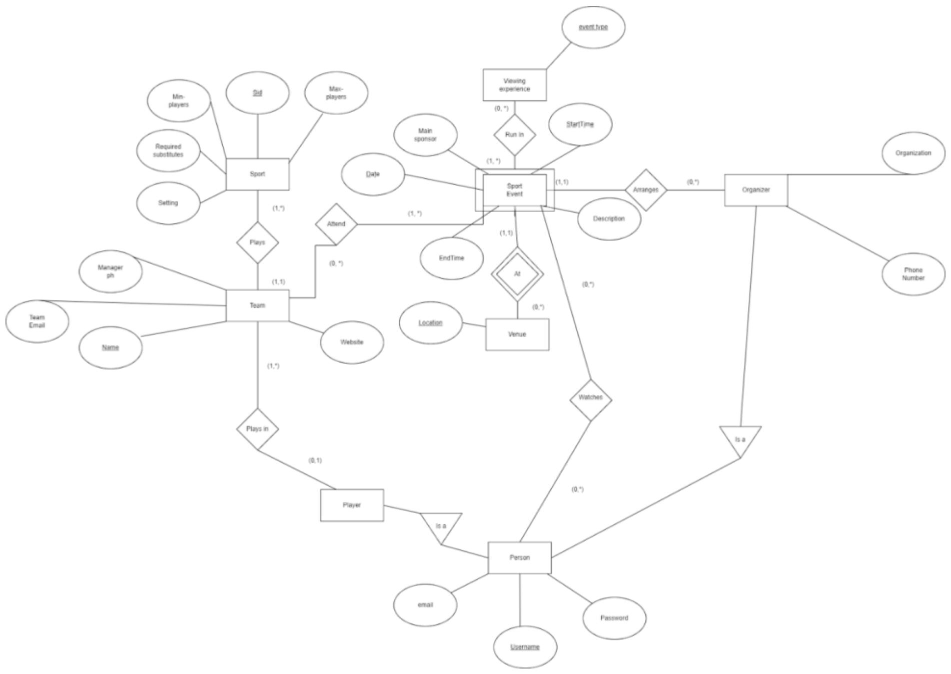
The sport platform wants to have a new database to store information about sport events. The platform stores. Sport events can be hosted in-person, virtual or both. The platform wants to store information about the players, organizers, viewers, the team the players are in and the sport. The team only records the current roster and players can only be assigned to at most 1 team at any given time. Details about the Organization are stored in the organizers. The Sport Event is created when it is organized by an organizer.

Commented [KH1]: To Be Updated

Explanation of relationships:
Team - Plays - Sport: 1 team can play 1 and only 1 sport; A sport will have at least 1 team, to many teams that play it.
Sport - Sport - Event: A sport can take place at no events, to many events; A sport event can hold one to many sports
Viewer - Attends - Sport Event: 1 viewer can attend 0 events or many events; A sport event can have 0 to many members in attendance
Sport-event - arranges - Organizer: A sport event is organized by one organizer and only 1, An organizer can organize 0 to many events.

A sporting event is a generalization between virtual esports and other forms of sports that do not have to be played from a shared location.

1.2 Revised ER Diagram

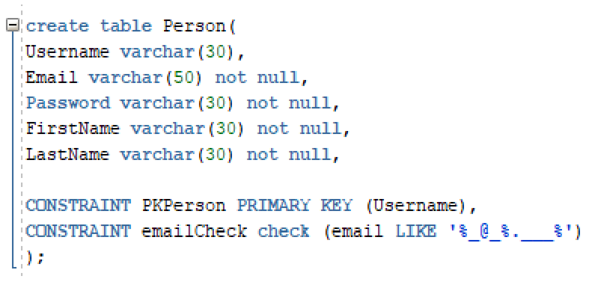


2. Project Milestone 2

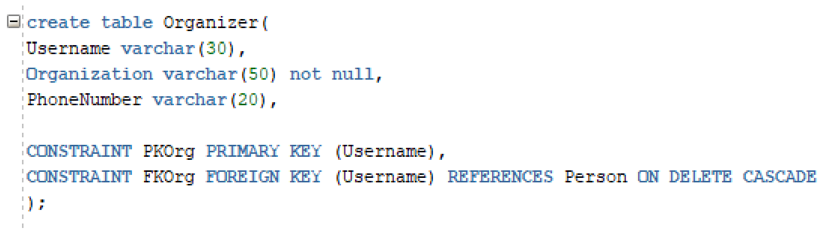
2.1 Relation Schema

Person
Username, Email, Password, First Name, Last Name
Player (Inherit from Person)
Username, TeamID
Organizer (Inherit from Person)
Username, Organization, Phone Number
Team
TeamID, Name, TeamEmail, Phone Number, Website, Sport
Sport
Name, MinPlayer, MaxPlayer, Required Substitute, Setting
Setting
Setting
Sport Event
Date, Start Time, Location, End Time, Main Sponsor, Description, Organizer
Venue
Location
Viewing experience
Event-type
Run In
Date, Start Time, Organizer, Event-type
Attends
TeamID, Date, Start Time, Location
Watches
Person, Date, Start Time, Location

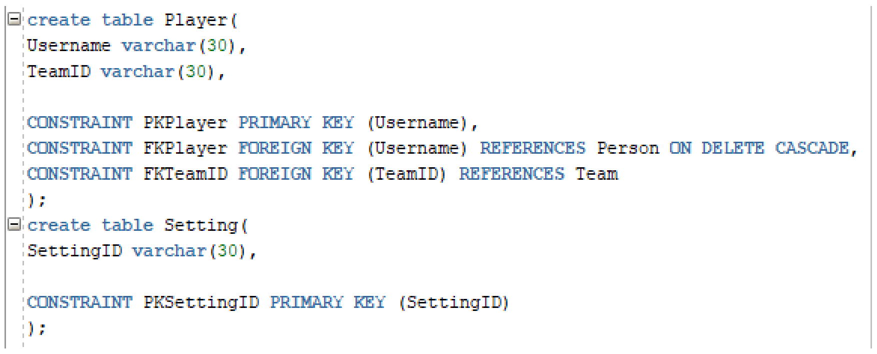
2.2 Table Definitions



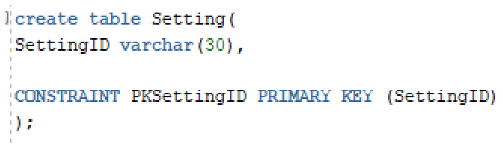
```
create table Person(
Username varchar(30),
Email varchar(50) not null,
Password varchar(30) not null,
FirstName varchar(30) not null,
LastName varchar(30) not null,

CONSTRAINT PKPerson PRIMARY KEY (Username),
CONSTRAINT emailCheck check (email LIKE '%_@_%.___%')
);
```



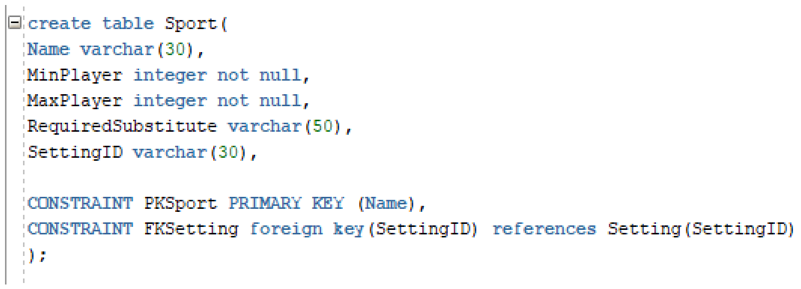
```sql
create table Organizer(
Username varchar(30),
Organization varchar(50) not null,
PhoneNumber varchar(20),

CONSTRAINT PKOrg PRIMARY KEY (Username),
CONSTRAINT FKOrg FOREIGN KEY (Username) REFERENCES Person ON DELETE CASCADE
);
```



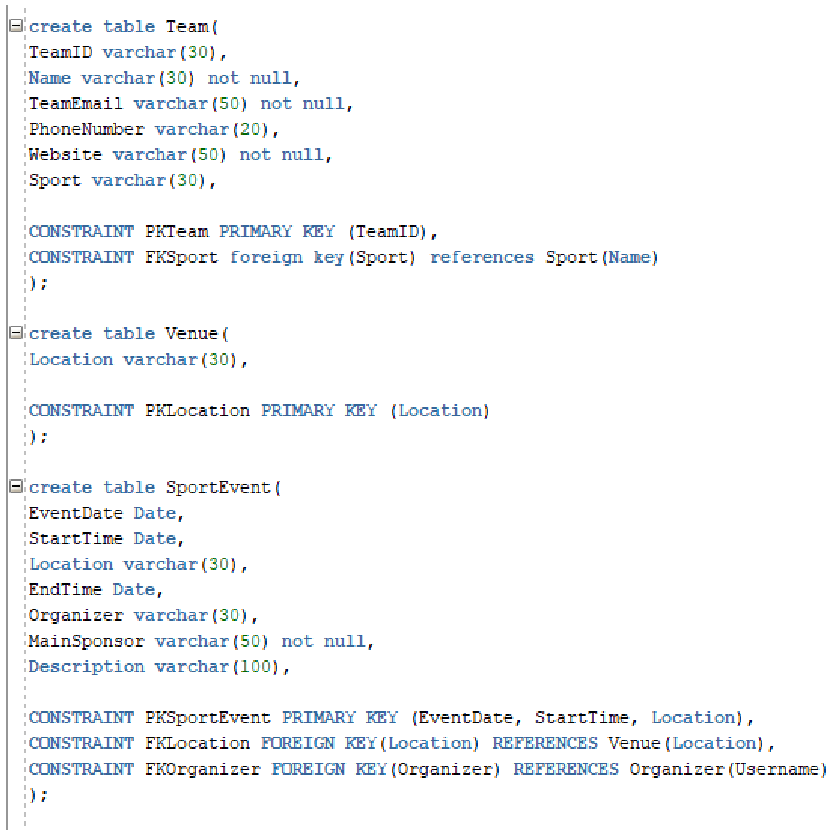
```sql
create table Player(
Username varchar(30),
TeamID varchar(30),

CONSTRAINT PKPlayer PRIMARY KEY (Username),
CONSTRAINT FKPlayer FOREIGN KEY (Username) REFERENCES Person ON DELETE CASCADE,
CONSTRAINT FKTeamID FOREIGN KEY (TeamID) REFERENCES Team
);
create table Setting(
SettingID varchar(30),

CONSTRAINT PKSettingID PRIMARY KEY (SettingID)
);
```



```sql
create table Setting(
SettingID varchar(30),

CONSTRAINT PKSettingID PRIMARY KEY (SettingID)
);
```

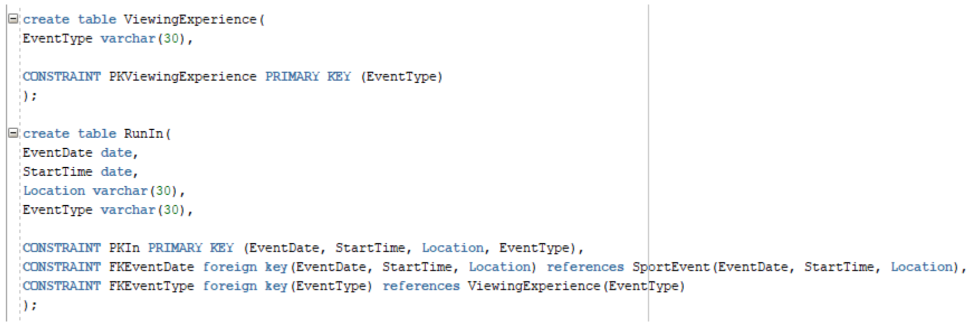


```sql
create table Sport(
Name varchar(30),
MinPlayer integer not null,
MaxPlayer integer not null,
RequiredSubstitute varchar(50),
SettingID varchar(30),

CONSTRAINT PKSport PRIMARY KEY (Name),
CONSTRAINT FKSetting foreign key(SettingID) references Setting(SettingID)
);
```

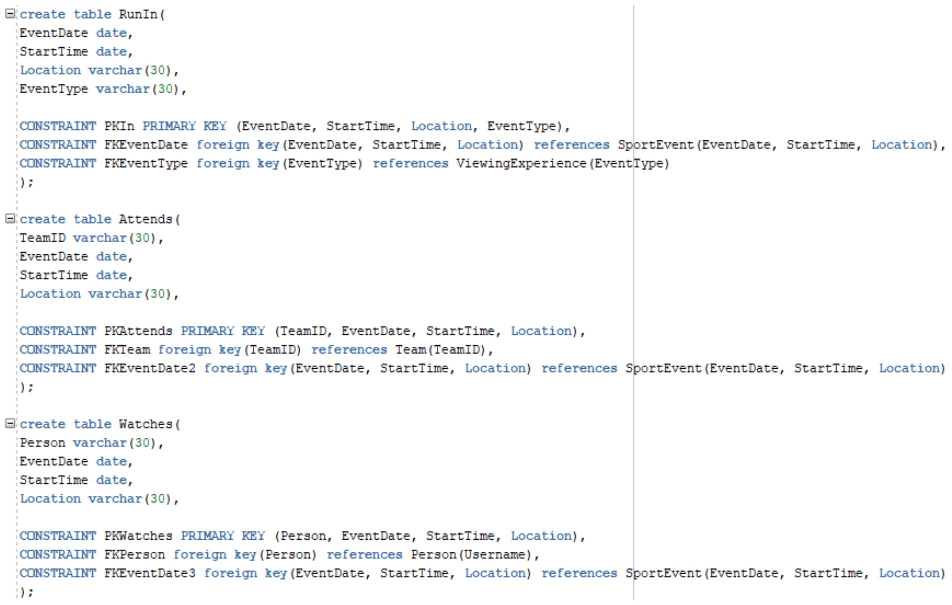


```sql
create table Team(
TeamID varchar(30),
Name varchar(30) not null,
TeamEmail varchar(50) not null,
PhoneNumber varchar(20),
Website varchar(50) not null,
Sport varchar(30),

CONSTRAINT PKTeam PRIMARY KEY (TeamID),
CONSTRAINT FKSport foreign key(Sport) references Sport(Name)
);

create table Venue(
Location varchar(30),

CONSTRAINT PKLocation PRIMARY KEY (Location)
);

create table SportEvent(
EventDate Date,
StartTime Date,
Location varchar(30),
EndTime Date,
Organizer varchar(30),
MainSponsor varchar(50) not null,
Description varchar(100),

CONSTRAINT PKSportEvent PRIMARY KEY (EventDate, StartTime, Location),
CONSTRAINT FKLocation FOREIGN KEY(Location) REFERENCES Venue(Location),
CONSTRAINT FKOrganizer FOREIGN KEY(Organizer) REFERENCES Organizer(Username)
);
```

s



```sql
create table ViewingExperience(
EventType varchar(30),

CONSTRAINT PKViewingExperience PRIMARY KEY (EventType)
);

create table RunIn(
EventDate date,
StartTime date,
Location varchar(30),
EventType varchar(30),

CONSTRAINT PKIn PRIMARY KEY (EventDate, StartTime, Location, EventType),
CONSTRAINT FKEventDate foreign key(EventDate, StartTime, Location) references SportEvent(EventDate, StartTime, Location),
CONSTRAINT FKEventType foreign key(EventType) references ViewingExperience(EventType)
);
```

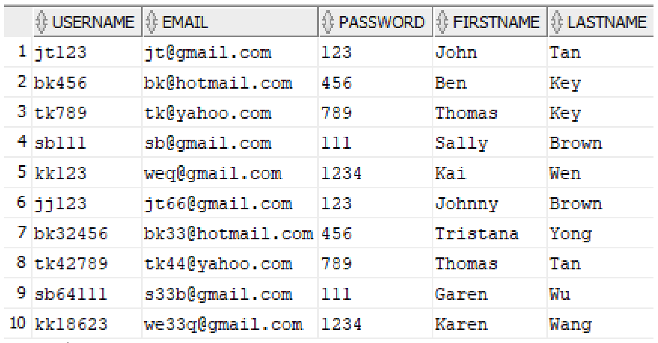


```
create table RunIn(
EventDate date,
StartTime date,
Location varchar(30),
EventType varchar(30),

CONSTRAINT PKIn PRIMARY KEY (EventDate, StartTime, Location, EventType),
CONSTRAINT FKEventDate foreign key(EventDate, StartTime, Location) references SportEvent(EventDate, StartTime, Location),
CONSTRAINT FKEventType foreign key(EventType) references ViewingExperience(EventType)
);

create table Attends(
TeamID varchar(30),
EventDate date,
StartTime date,
Location varchar(30),

CONSTRAINT PKAttends PRIMARY KEY (TeamID, EventDate, StartTime, Location),
CONSTRAINT FKTeam foreign key(TeamID) references Team(TeamID),
CONSTRAINT FKEventDate2 foreign key(EventDate, StartTime, Location) references SportEvent(EventDate, StartTime, Location)
);

create table Watches(
Person varchar(30),
EventDate date,
StartTime date,
Location varchar(30),

CONSTRAINT PKWatches PRIMARY KEY (Person, EventDate, StartTime, Location),
CONSTRAINT FKPerson foreign key(Person) references Person(Username),
CONSTRAINT FKEventDate3 foreign key(EventDate, StartTime, Location) references SportEvent(EventDate, StartTime, Location)
);
```

2.3 Dataset

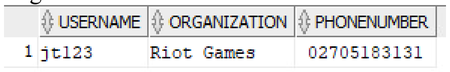
2.3.1 Small

Person



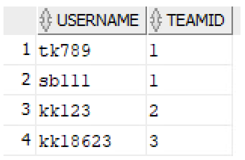
	USERNAME	EMAIL	PASSWORD	FIRSTNAME	LASTNAME
1	jt123	jt@gmail.com	123	John	Tan
2	bk456	bk@hotmail.com	456	Ben	Key
3	tk789	tk@yahoo.com	789	Thomas	Key
4	sb111	sb@gmail.com	111	Sally	Brown
5	kk123	weq@gmail.com	1234	Kai	Wen
6	jj123	jt66@gmail.com	123	Johnny	Brown
7	bk32456	bk33@hotmail.com	456	Tristana	Yong
8	tk42789	tk44@yahoo.com	789	Thomas	Tan
9	sb64111	s33b@gmail.com	111	Garen	Wu
10	kk18623	we33q@gmail.com	1234	Karen	Wang

Organizer



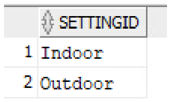
	USERNAME	ORGANIZATION	PHONENUMBER
1	jt123	Riot Games	02705183131

Player



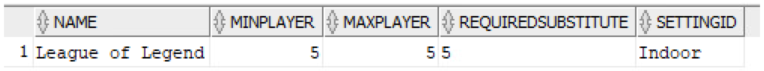
	USERNAME	TEAMID
1	tk789	1
2	sb111	1
3	kk123	2
4	kk18623	3

Setting



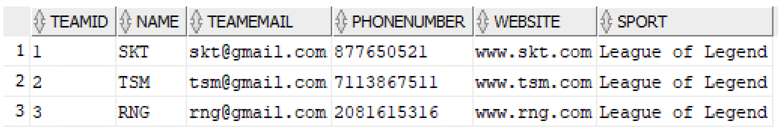
	SETTINGID
1	Indoor
2	Outdoor

Sport



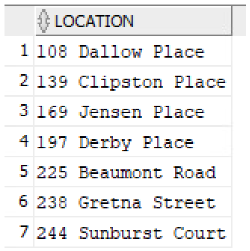
	NAME	MINPLAYER	MAXPLAYER	REQUIREDSUBSTITUTE	SETTINGID
1	League of Legend	5	5	5	Indoor

Team



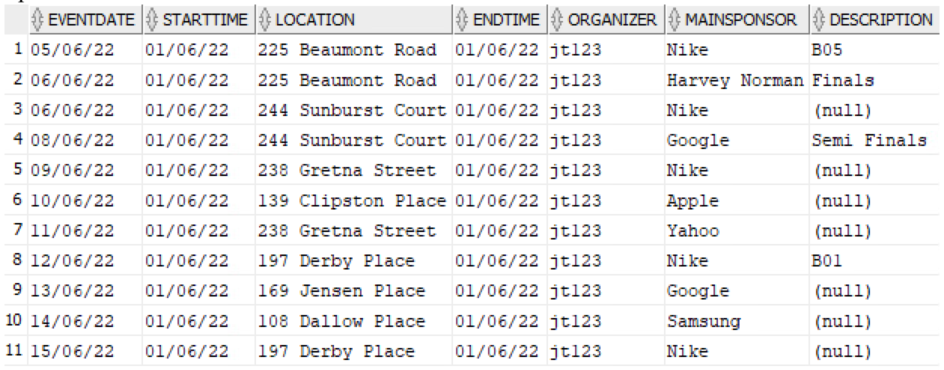
	TEAMID	NAME	TEAMEMAIL	PHONENUMBER	WEBSITE	SPORT
1	1	SKT	skt@gmail.com	877650521	www.skt.com	League of Legend
2	2	TSM	tsm@gmail.com	7113867511	www.tsm.com	League of Legend
3	3	RNG	rng@gmail.com	2081615316	www.rng.com	League of Legend

Venue



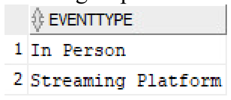
	LOCATION
1	108 Dallow Place
2	139 Clipston Place
3	169 Jensen Place
4	197 Derby Place
5	225 Beaumont Road
6	238 Gretna Street
7	244 Sunburst Court

Sport Event



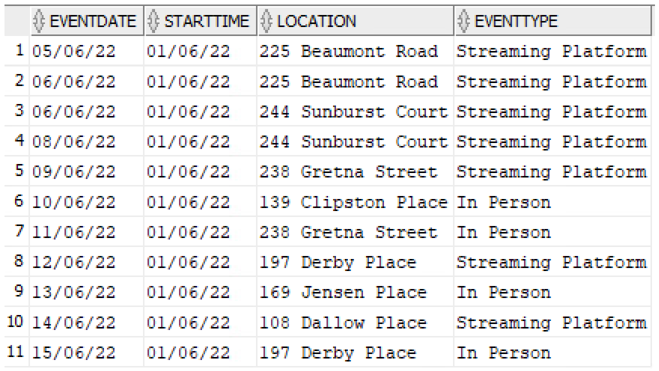
	EVENTDATE	STARTTIME	LOCATION	ENDTIME	ORGANIZER	MAINSPONSOR	DESCRIPTION
1	05/06/22	01/06/22	225 Beaumont Road	01/06/22	jtl23	Nike	B05
2	06/06/22	01/06/22	225 Beaumont Road	01/06/22	jtl23	Harvey Norman	Finals
3	06/06/22	01/06/22	244 Sunburst Court	01/06/22	jtl23	Nike	(null)
4	08/06/22	01/06/22	244 Sunburst Court	01/06/22	jtl23	Google	Semi Finals
5	09/06/22	01/06/22	238 Gretna Street	01/06/22	jtl23	Nike	(null)
6	10/06/22	01/06/22	139 Clipston Place	01/06/22	jtl23	Apple	(null)
7	11/06/22	01/06/22	238 Gretna Street	01/06/22	jtl23	Yahoo	(null)
8	12/06/22	01/06/22	197 Derby Place	01/06/22	jtl23	Nike	B01
9	13/06/22	01/06/22	169 Jensen Place	01/06/22	jtl23	Google	(null)
10	14/06/22	01/06/22	108 Dallow Place	01/06/22	jtl23	Samsung	(null)
11	15/06/22	01/06/22	197 Derby Place	01/06/22	jtl23	Nike	(null)

Viewing Experience



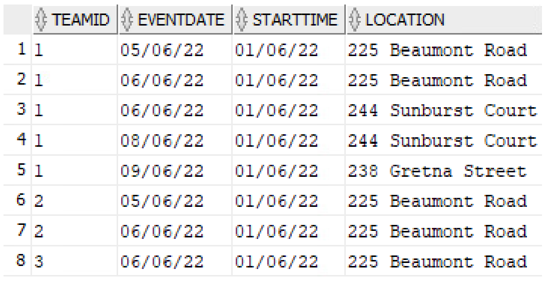
	EVENTTYPE
1	In Person
2	Streaming Platform

Run In



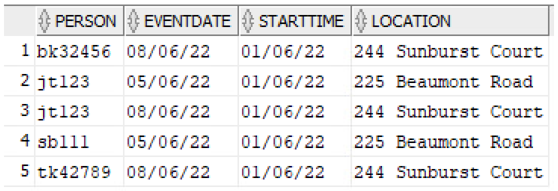
	EVENTDATE	STARTTIME	LOCATION	EVENTTYPE
1	05/06/22	01/06/22	225 Beaumont Road	Streaming Platform
2	06/06/22	01/06/22	225 Beaumont Road	Streaming Platform
3	06/06/22	01/06/22	244 Sunburst Court	Streaming Platform
4	08/06/22	01/06/22	244 Sunburst Court	Streaming Platform
5	09/06/22	01/06/22	238 Gretna Street	Streaming Platform
6	10/06/22	01/06/22	139 Clipston Place	In Person
7	11/06/22	01/06/22	238 Gretna Street	In Person
8	12/06/22	01/06/22	197 Derby Place	Streaming Platform
9	13/06/22	01/06/22	169 Jensen Place	In Person
10	14/06/22	01/06/22	108 Dallow Place	Streaming Platform
11	15/06/22	01/06/22	197 Derby Place	In Person

Attends



	TEAMID	EVENTDATE	STARTTIME	LOCATION
1	1	05/06/22	01/06/22	225 Beaumont Road
2	1	06/06/22	01/06/22	225 Beaumont Road
3	1	06/06/22	01/06/22	244 Sunburst Court
4	1	08/06/22	01/06/22	244 Sunburst Court
5	1	09/06/22	01/06/22	238 Gretna Street
6	2	05/06/22	01/06/22	225 Beaumont Road
7	2	06/06/22	01/06/22	225 Beaumont Road
8	3	06/06/22	01/06/22	225 Beaumont Road

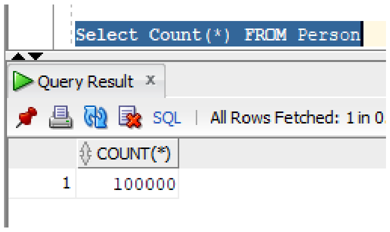
Watches



	PERSON	EVENTDATE	STARTTIME	LOCATION
1	bk32456	08/06/22	01/06/22	244 Sunburst Court
2	jt123	05/06/22	01/06/22	225 Beaumont Road
3	jt123	08/06/22	01/06/22	244 Sunburst Court
4	sb111	05/06/22	01/06/22	225 Beaumont Road
5	tk42789	08/06/22	01/06/22	244 Sunburst Court

2.3.2 Large

Person Table



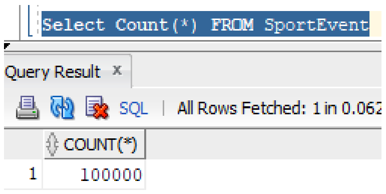
```
Select Count(*) FROM Person
```

Query Result ×

SQL | All Rows Fetched: 1 in 0

	COUNT(*)
1	100000

Sport Event Tables



```
Select Count(*) FROM SportEvent
```

Query Result ×

SQL | All Rows Fetched: 1 in 0.062

	COUNT(*)
1	100000

The screenshot above are the two main table.

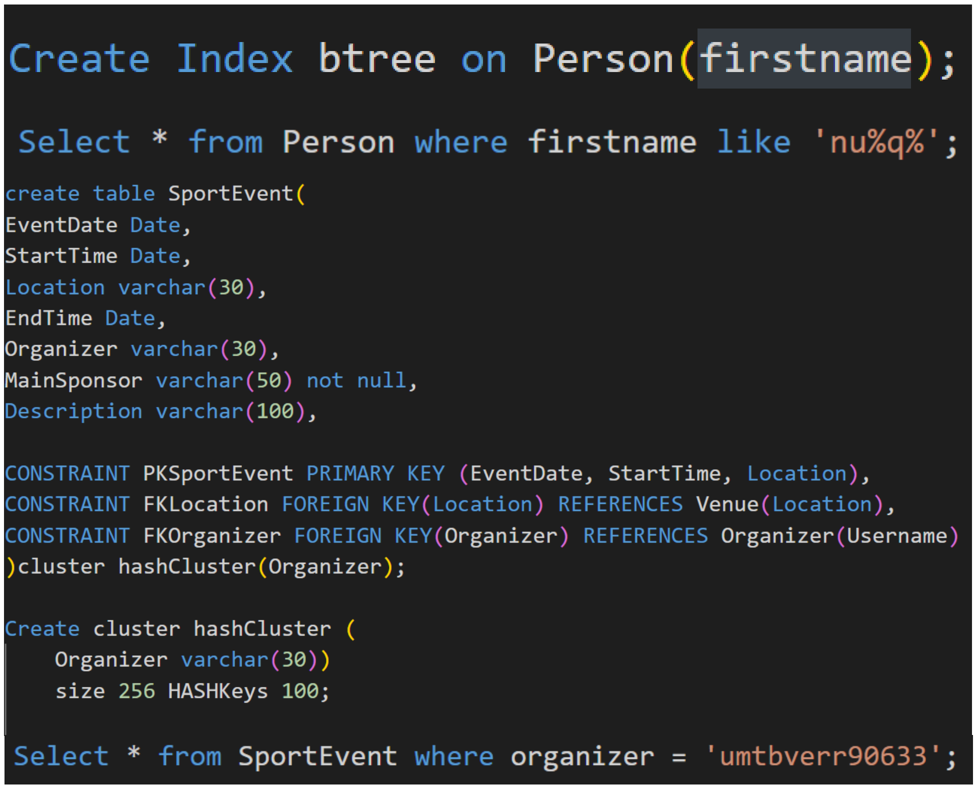
The dataset is randomly generated using a java program generating random string of chars/integer.

2.4 Application

3. Project Milestone 3

3.1 Indexing and Querying Optimization

3.1.1 Queries and SQL script to create index



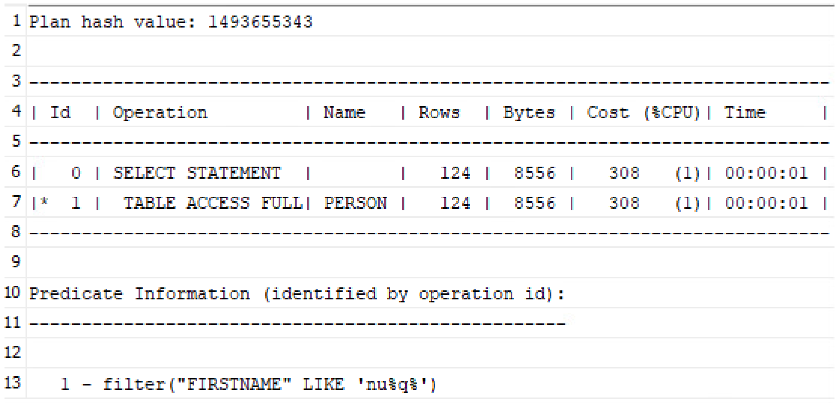
```
Create Index btree on Person(firstname);

 Select * from Person where firstname like 'nu%q%';
create table SportEvent(
EventDate Date,
StartTime Date,
Location varchar(30),
EndTime Date,
Organizer varchar(30),
MainSponsor varchar(50) not null,
Description varchar(100),

CONSTRAINT PKSportEvent PRIMARY KEY (EventDate, StartTime, Location),
CONSTRAINT FKLocation FOREIGN KEY(Location) REFERENCES Venue(Location),
CONSTRAINT FKOrganizer FOREIGN KEY(Organizer) REFERENCES Organizer(Username)
)cluster hashCluster(Organizer);

Create cluster hashCluster (
    Organizer varchar(30))
    size 256 HASHKeys 100;

 Select * from SportEvent where organizer = 'umtbverr90633';
```

3.1.2 Why was it chosen?

B-Tree:
We decided to choose first name on Person table. It allows us to search for different group of people with some similarities with their first name.
Hash-Cluster:
We chose organizer as the hash for the Sport Event table. We felt that there will be lot of queries trying to find the events that are organized by certain organizer. Using hash allow for quick direct to all event organized by the one organizer.

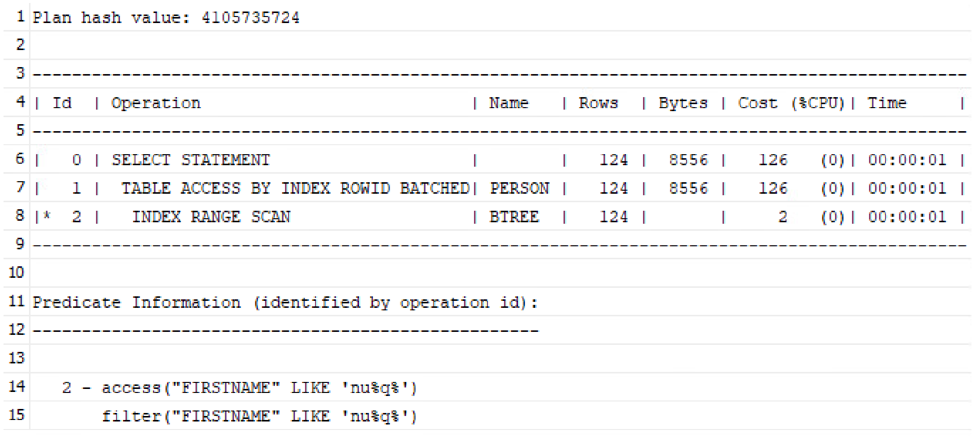
3.1.3 Performance Measurement

B-Tree Index Before Implement



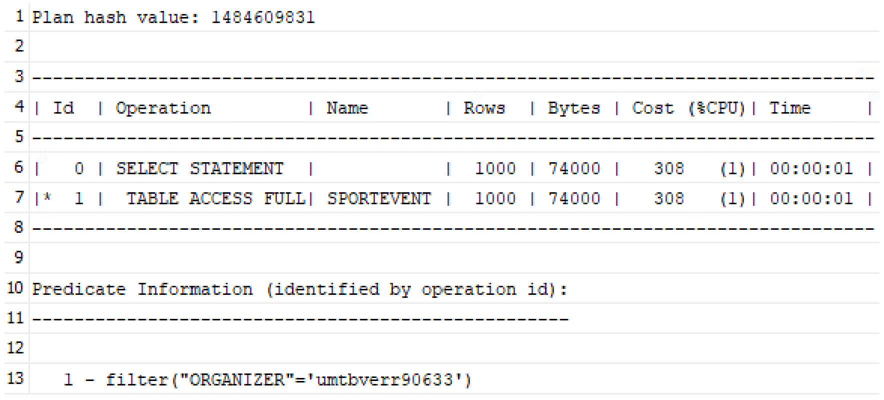
```
Plan hash value: 1493655343

----------------------------------------------------------------------------
| Id  | Operation         | Name   | Rows  | Bytes | Cost (%CPU)| Time     |
----------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |        |   124 |  8556 |   308   (1)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| PERSON |   124 |  8556 |   308   (1)| 00:00:01 |
----------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("FIRSTNAME" LIKE 'nu%q%')
```

B-Tree Index After Implement



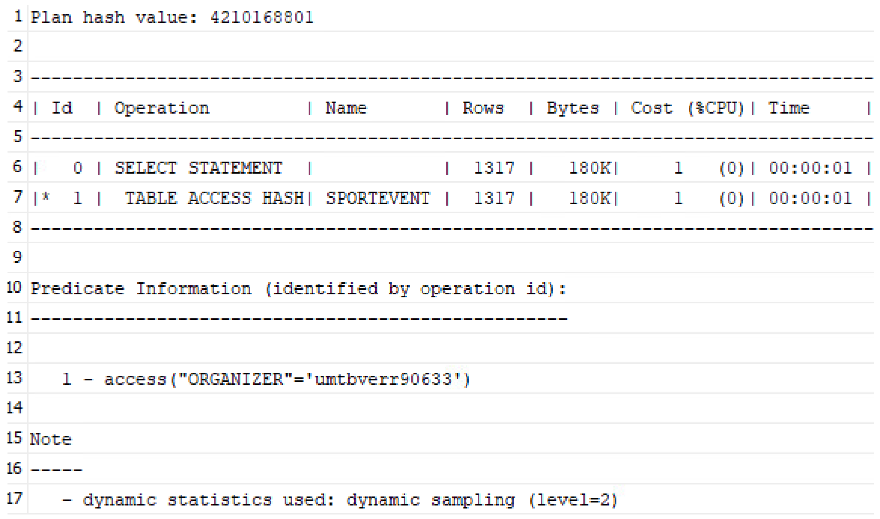
```
Plan hash value: 4105735724

-----------------------------------------------------------------------------------------------
| Id  | Operation                            | Name    | Rows  | Bytes | Cost  (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                     |         |   124 |  8556 |   126    (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID BATCHED| PERSON  |   124 |  8556 |   126    (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN                   | BTREE   |   124 |       |     2    (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("FIRSTNAME" LIKE 'nu%q%')
       filter("FIRSTNAME" LIKE 'nu%q%')
```

Hash-Cluster Index Before Implement



```
Plan hash value: 1484609831

-------------------------------------------------------------------------------------
| Id  | Operation         | Name        | Rows  | Bytes | Cost  (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |             |  1000 | 74000 |   308    (1)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| SPORTEVENT  |  1000 | 74000 |   308    (1)| 00:00:01 |
-------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("ORGANIZER"='umtbverr90633')
```

Hash-Cluster Index After Implement



```
Plan hash value: 4210168801

--------------------------------------------------------------------------------
| Id  | Operation          | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |            |  1317 |   180K|     1   (0)| 00:00:01 |
|*  1 |  TABLE ACCESS HASH| SPORTEVENT |  1317 |   180K|     1   (0)| 00:00:01 |
--------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("ORGANIZER"='umtbverr90633')

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)
```

3.1.4 Discussion of Performance Measurement

4. COMPX323-22A Project Checklist

Kevin Han:	(Contributions %)
Bedir Asici:	(Contributions %)
Tetsusaburo Kato:	(Contributions %)

Project Milestone 1		
1	1a. Clear structure of milestone material, including headings, sections, readable screenshots with captions. This checklist should be included and filled in.	
2	1b. Database application description.	
6	1c. Revised ER Diagram.	✓
Project Milestone 2		
4	2. Relational schema for your ER Diagram.	✓
6	3. Table definitions in Oracle, include SQL script which creates relevant tables etc.	✓
3	4a. Dataset: small (screenshots of dataset successfully loaded).	✓
4	4b. Dataset: large. Description of how data was created (incl code if relevant) Screenshot of large dataset successfully loaded (use count).	✓
	5. Application: Functionality to display and modify the database. System should be error proof with appropriate user messages. Screenshots showing functionality, with appropriate descriptions.	
Project Milestone 3		
	6. Indexing and Query Optimization: Show queries used and SQL script that creates the indexes. Discussion of why these indexes were chosen to optimize the queries. Performance measurements with and without indexes (with query plan). Discussion of performance measurements.	
	7. Application: extend with MongoDB. MongoDB version of database (show structure) + small dataset (screenshot). Explanation of the data structures you have chosen and comparison to your SQL version. Core functionality of application in second tab/area, using MongoDB (screenshots).	

5. Code, SQL and Data

https://github.com/Lecreator-KH/COMPX323App.git