DATABASE MANAGEMENT SYSTEM

Project Report

IIITD SPORTS MANAGEMENT SYSTEM

Naval Kumar Shukla (2019065)
Jishnu Raj Parashar (2019048)
Medhavi Sabherwal (2019371)

INDEX

Problem Statement :

Sports management in a college is an essential task. It is often difficult to coordinate amongst different people with different roles in college. The method of physically keeping records is outdated and a more efficient system is required. There is a lack of facility to reserve slots which when present, can help ease out excess work and coordination. There are a number of existing websites that lack a formal platform with all necessary information integrated for the smooth running of sports in a college. This builds our motivation to create a Sports Management System in IIITD.

Project Description :

Sports is an essential part of college life. Organisation is the most important task and it is necessary to have and showcase, the part of college's identity which is formed through sports, responsibly. The goal of having a Sports Management System in IIITD is to ensure that users of the system are able to carry out sports-related activities in an organised way as well as save time.
Our platform is able to reserve slots and equipment, keep track of events, access information about the various bodies of IIITD involved in sports and its management, notify about tournaments and achievements, schedule training and more. It provides an extensive database system which helps the various bodies of college manage sports in an effective and organised way.

Stakeholders :

- Sports Council
- Student
- Faculty
- Guard
- Instructor

Stakeholders' Roles :

1) Sports Council :

a) Showcase the achievements
b) Notify about upcoming events
c) Schedule & Result (Live updates)
d) Keep track of the availability and condition of sports items
e) Add/Upgrade the present items
f) Add a new sport (and its corresponding items)
g) Reserve (or free reserved) time slots & sports items

2) Student :

a) Check the availability of a particular time slot & sports items
b) Get updates of the upcoming events/tournaments
c) Access results of the past events/tournaments
d) Access information about the sports council
e) Access information about the instructors and class timings
f) Enroll/Leave a class
g) Reserve time slots & sports items
h) Confirm their check-in/out

3) Faculty :

a) Reserve time slots & sports items
b) Notify about upcoming faculty tournaments
c) Schedule & Result

4) Guard :

- **a)** Access information about other guards
- **b)** Keeping track of the sports items
- **c)** Notify sports council about the availability of sport's items
- **d)** Check in/out students/faculty/others

5) Instructor :

- **a)** Notify about the class details
- **b)** Reserve time slots & sports items
- **c)** Enroll students in the class
- **d)** See the information of students who have enrolled in the class

Key Questions The System Will be Answering :

1) **Sports Council:**
   - **a)** How many sports' items are available, damaged & needed currently?
   - **b)** Which time slots are free and available to reserve currently?
   - **c)** Which time slots are reserved and by whom?
   - **d)** Which sport has been the most popular among students and faculty?
   - **e)** Who are the top performers (among students as well as among faculty)?
   - **f)** Who's the on-duty guard and what is his/her point-of-contact?

2) **Student:**
   - **a)** How many sports' items are available currently?
   - **b)** Who are the members and coordinators of the sports council and what are their roles (and achievements) and their point-of-contacts?
   - **c)** Who are the instructors, which sport are they representing and what are the class timings?
   - **d)** What are the updates on the upcoming events/tournaments?
   - **e)** What were the results of the past events/tournaments?
   - **f)** What are the classes that I have enrolled in?
   - **g)** Who's the on-duty guard and what is his/her point-of-contact?

3) **Faculty:**
    - **a)** Which time slots are free and available to reserve currently?
    - **b)** Which time slots are reserved, by whom (with their point-of-contacts)?
    - **c)** What are the updates on the upcoming events/tournaments?
    - **d)** What were the results of the past events/tournaments?
    - **e)** How's my record in the faculty tournaments?
    - **f)** Who's the on-duty guard and what is his/her point-of-contact?

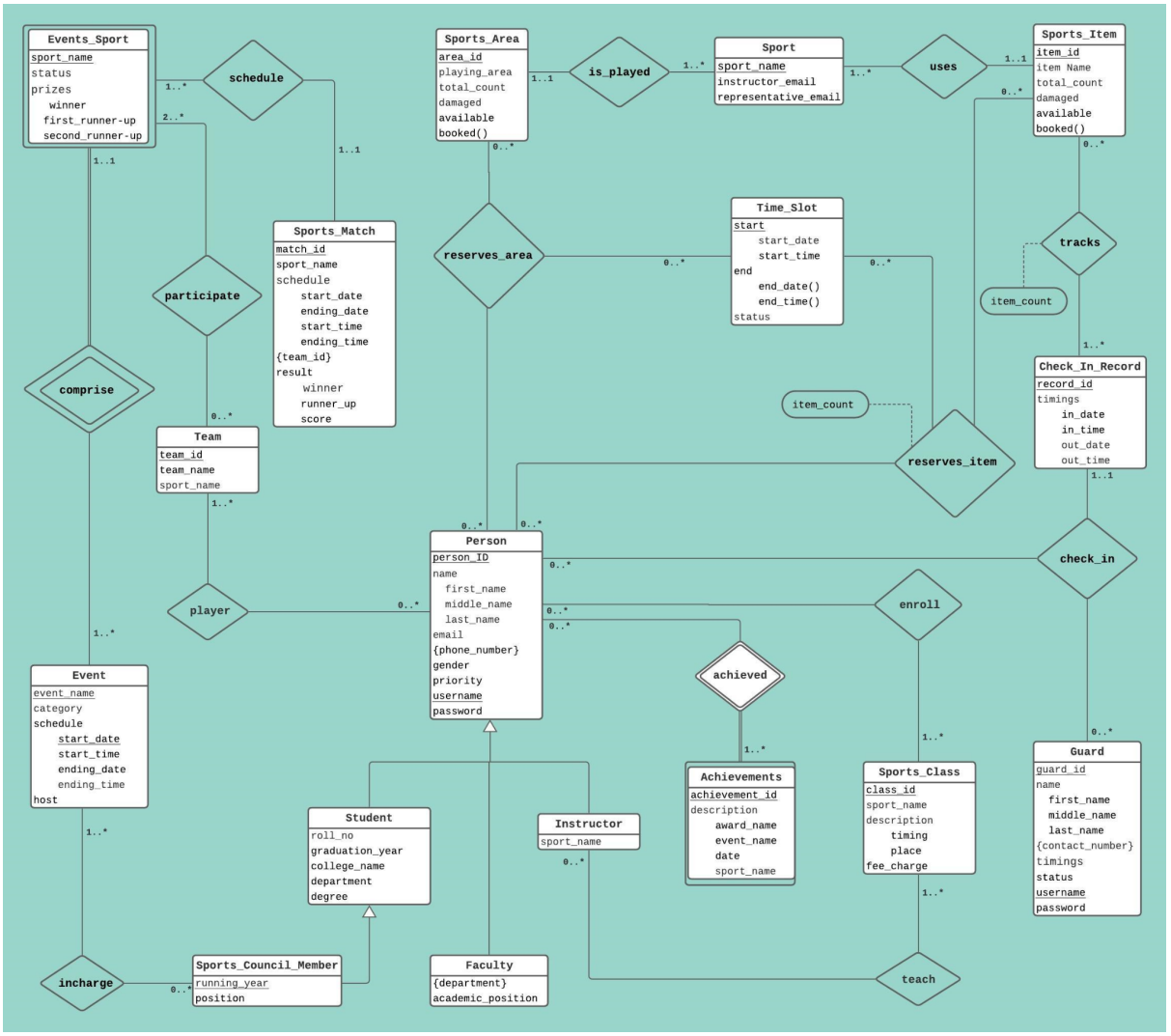
4) **Guard:**
    - **a)** What are the details of people who have checked in?
    - **b)** How many people have checked in but haven't checked out?
    - **c)** What are the details of the guards before and after him/her?
    - **d)** Which time slots are free currently?
    - **e)** Who are the members and coordinators of the current sports council?

5) **Instructor:**
    - **a)** Which time slots are free and available to reserve currently?
    - **b)** Which time slots are reserved and by whom (with their point-of-contacts) ?
    - **c)** How many people have enrolled in (and unenrolled out of) the class, and what are their details?
    - **d)** Who are the top performers of the class?
    - **e)** Who is the on-duty guard and what is his/her point-of-contact?

Mini World:

1) Details of every student and faculty of IIITD, as well as those who have participated from different colleges in any event, and/or have used the IIITD sports facility, will be available.
2) Details of a student include his/her name, roll no (Unique id), email-id, contact details, branch, year and college.
3) Some students form the Sports Council and hold various positions in it (sports coordinator, co-sports coordinator etc). Their positions along with the sport they represent and the experience they have, will be available for all.
4) Details of a faculty include his/her name, email-id, contact details, department and academic rank.
5) A student/faculty can also have some sports achievements which will be available along with their levels (state, national, international etc).
6) The statistics (games/tournaments played and won along with the sport's name) associated with a player (student/faculty) can also be seen.
7) The college can be a part of many sports events (intra-college as well inter-college), and each event can have multiple sports in it.
8) Each event needs to have schedule details (timing, venue, host etc), and each sport in it needs to have details such as venue, timings, participating teams and winner.
9) Each event-sport can have multiple participating teams in it, and each team consists of either a single member or multiple members in it.
10) The Time Slot showing booked and available time slots of the current day, where the sports council, faculty members and student body can book a particular time slot. (Sports Council and Faculty will be given the priority to book them).
11) The contact details along with the name of the on-duty guard will be available.
12) Guards are able to confirm a check-in or a check-out by a student (or other visitors/workers as well) by entering the time of exchange.
13) All the sports items (along with information regarding the damaged ones) will be available for each sport.
14) The details of all the sports instructors (including his/her specialty and contact details) will be available. Each instructor will provide the class details, and those interested can join the class.

E-R Diagram


Events_Sport
sport_name
status
prizes
winner
first_runner-up
second_runner-up
schedule
Sports_Area
area_id
playing_area
total_count
damaged
available
booked()
is_played
Sport
sport_name
instructor_email
representative_email
uses
Sports_Item
item_id
item Name
total_count
damaged
available
booked()
Sports_Match
match_id
sport_name
schedule
start_date
ending_date
start_time
ending_time
{team_id}
result
winner
runner_up
score
reserves_area
Time_Slot
start
start_date
start_time
end
end_date()
end_time()
status
tracks
item_count
participate
comprise
Team
team_id
team_name
sport_name
item_count
reserves_item
Check_In_Record
record_id
timings
in_date
in_time
out_date
out_time
check_in
Person
person_ID
name
first_name
middle_name
last_name
email
{phone_number}
gender
priority
username
password
player
enroll
achieved
Event
event_name
category
schedule
start_date
start_time
ending_date
ending_time
host
Achievements
achievement_id
description
award_name
event_name
date
sport_name
Sports_Class
class_id
sport_name
description
timing
place
fee_charge
Guard
guard_id
name
first_name
middle_name
last_name
{contact_number}
timings
status
username
password
Student
roll_no
graduation_year
college_name
department
degree
Instructor
sport_name
incharge
Sports_Council_Member
running_year
position
Faculty
{department}
academic_position
teach

Database Schemas

person

Attribute	Data Type
person_ID	int NOT NULL UNIQUE
email	varchar(50) NOT NULL
first_name	varchar(30) NOT NULL
middle_name	varchar(30) DEFAULT NULL
last_name	varchar(30) DEFAULT IIITD
gender	varchar(10) NOT NULL CHECK (gender in (“Male”, “Female”))
priority	int DEFAULT 0 CHECK (priority >= 0 and priority <= 2)
username	varchar(45) UNIQUE NOT NULL
password	varchar(45) NOT NULL

PRIMARY KEY : person_ID
FOREIGN KEY : NONE

contact

Attribute	Data Type
person_ID	int NOT NULL
phone_number	varchar(30) NOT NULL

PRIMARY KEY: (person_ID, phone_number)
FOREIGN KEY: person_ID from *person*

student

Attribute	Data Type
person_ID	int NOT NULL UNIQUE
roll_no	varchar(20) NOT NULL
graduation_year	int NOT NULL
college_name	varchar(50) NOT NULL
degree	varchar(30) NOT NULL

PRIMARY KEY: person_ID
FOREIGN KEY: person_ID from *person*

faculty

Attribute	Data Type
person_ID	int NOT NULL UNIQUE
academic_position	varchar(25) NOT NULL

PRIMARY KEY: person_ID
FOREIGN KEY: person_ID from *person*

department

Attribute	Data Type
person_ID	int NOT NULL
department	varchar(30) NOT NULL

PRIMARY KEY: (person_ID, department)
FOREIGN KEY: person_ID from person

sports_council_member

Attribute	Data Type
person_ID	int NOT NULL
running_year	YEAR NOT NULL
position	varchar(20) NOT NULL

PRIMARY KEY: (person_ID, running_year)
FOREIGN KEY: person_ID from student

achievements

Attribute	Data Type
achievement_ID	int NOT NULL UNIQUE
award_name	varchar(50) NOT NULL
achievement_date	date NOT NULL CHECK (achievement_date >= event_start_date)
event_name	varchar(40) NOT NULL
event_start_date	date NOT NULL
sport_name	varchar(30) NOT NULL

PRIMARY KEY: achievement_ID
FOREIGN KEY: (event_name, event_start_date, sport_name) from *events_sport*

achieved

Attribute	Data Type
person_ID	int NOT NULL
achievement_ID	int NOT NULL

PRIMARY KEY: (person_ID, achievement_ID)
FOREIGN KEY: person_ID from *person,* achievement_ID from *achievement*

time_slot

Attribute	Data type
start_date	DATE NOT NULL
start_time	TIME NOT NULL
status	varchar(30) DEFAULT “Available” CHECK (status in ("Available", "Booked"))

PRIMARY KEY: (start_date, start_time)
FOREIGN KEY: None

reserves_item

Attribute	Data type
person_ID	int NOT NULL
start_date	DATE NOT NULL
start_time	TIME NOT NULL
item_ID	int NOT NULL
item_count	int DEFAULT 1 CHECK (item_count ≤ item_id.available)

PRIMARY KEY: (person_ID, start_date, start_time, item_ID)
FOREIGN KEY: person_ID from *person,* (start_date, start_time) from time_slot, item_ID from *sports_item*

reserves_area

Attribute	Data type
person_ID	int NOT NULL
start_date	DATE NOT NULL
start_time	TIME NOT NULL
area_ID	varchar(10) NOT NULL

PRIMARY KEY: (person_ID, start_date, start_time, area_ID)
FOREIGN KEY: person_ID from *person*, (start_date, start_time) from *time_slot*, area_ID from *sports_area*

sport

Attribute	Data Type
sport_name	varchar(40) NOT NULL UNIQUE
instructor_ID	int DEFAULT NULL
student_rep_ID	int DEFAULT NULL

PRIMARY KEY: sport_name
FOREIGN KEY: instructor_ID from *instructor*, sudent_rep_ID from *sports_council_member*

instructor

Attribute	Data Type
person_ID	int NOT NULL UNIQUE
sport_name	varchar(30) NOT NULL

PRIMARY KEY: person_ID
FOREIGN KEY: person_ID from *person*, sport_name from *sport*

teach

Attribute	Data Type
class_ID	varchar(30) NOT NULL
person_ID	int NOT NULL

PRIMARY KEY: (class_ID, person_ID)
FOREIGN KEY: class_ID from *sports_class*, person_ID from *instructor*

guard

Attribute	Data Type
guard_ID	int NOT NULL UNIQUE
first_name	varchar(30) NOT NULL
middle_name	varchar(30) DEFAULT NULL
last_name	varchar(30) NOT NULL

contact_number	varchar(30) NOT NULL UNIQUE
alternate_number	varchar(30) DEFAULT NULL
timings	varchar(50) NOT NULL
status	varchar(30) NOT NULL DEFAULT “Off” CHECK (status in ("Off", "On"))
username	varchar(45) UNIQUE NOT NULL
password	varchar(45) NOT NULL

PRIMARY KEY: guard_ID
FOREIGN KEY: None

check_in_record

Attribute	Data Type
record_ID	int NOT NULL AUTO INCREMENT
person_ID	int NOT NULL
guard_ID	int NOT NULL
in_date	DATE NOT NULL
in_time	TIME NOT NULL
out_date	DATE DEFAULT NULL
out_time	TIME DEFAULT NULL

PRIMARY KEY: record_ID
FOREIGN KEY: person_ID from *person*, guard_ID from *guard*

tracks

Attribute	Data Type
item_ID	int NOT NULL
record_ID	int NOT NULL
item_count	int DEFAULT 1 CHECK (item_count ≤ item_id.avaialble)

PRIMARY KEY: record_ID
FOREIGN KEY: item_ID from *sports_item*, record_ID from *check_in_record*

sports_item

Attribute	Data Type
item_ID	int NOT NULL UNIQUE
sport_name	varchar(40) NOT NULL
item_name	varchar(40) NOT NULL
total_count	int NOT NULL CHECK (total_count > 0)
damaged	int NOT NULL DEFAULT 0 CHECK (damaged >= 0 and damaged ≤ total_count)

available	int NOT NULL CHECK (available >= 0 and available ≤ total_count) CHECK (available + damaged ≤ total_count)

PRIMARY KEY: item_ID
FOREIGN KEY: sport_name from *sport*

sports_area

Attribute	Data Type
area_ID	varchar(10) NOT NULL UNIQUE
sport_name	varchar(40) NOT NULL
playing_area	varchar(40) NOT NULL
total_count	int NOT NULL CHECK (total_count > 0)
damaged	int NOT NULL DEFAULT 0 CHECK (damaged >= 0)
available	int NOT NULL CHECK (available >= 0) CHECK (available + damaged ≤ total_count)

PRIMARY KEY: area_ID
FOREIGN KEY: sport_name from *sport*

sports_class

Attribute	Data Type
class_ID	varchar(30) NOT NULL UNIQUE
sport_name	varchar(30) NOT NULL
timing	varchar(50) NOT NULL
place	varchar(50) NOT NULL
fee_charge	int DEFAULT 0

PRIMARY KEY: class_ID
FOREIGN KEY: sport_name from *sport*

enroll

Attribute	Data Type
class_ID	varchar(30) NOT NULL
person_ID	int NOT NULL

PRIMARY KEY: (class_ID, person_ID)
FOREIGN KEY: class_ID from *sports_Class*, person_ID from *person*

event

Attribute	Data Type
event_name	varchar(40) NOT NULL
start_date	DATE NOT NULL
end_date	DATE NOT NULL CHECK (end_date >= start_date)
start_time	TIME NOT NULL
end_time	TIME NOT NULL
host	varchar(50) NOT NULL

PRIMARY KEY: (event_name, start_date)
FOREIGN KEY: NONE

event_categories

Attribute	Data Type
event_name	varchar(40) NOT NULL
category	varchar(30) NOT NULL

PRIMARY KEY: event_name
FOREIGN KEY: event_name from *event*

incharge

Attribute	Data Type
event_name	varchar(40) NOT NULL
start_date	DATE NOT NULL
person_ID	int NOT NULL
running_year	YEAR NOT NULL

PRIMARY KEY: (event_name, start_date, person_ID, running_year)
FOREIGN KEY: (event_name, start_date) from *event*, (person_ID, running_year) from *sports_council_member*

team

Attribute	Data Type
team_ID	int NOT NULL UNIQUE
team_name	varchar(30) DEFAULT NULL
sport_name	varchar(30) NOT NULL

PRIMARY KEY: team_ID
FOREIGN KEY: sport_name from *sport*

player

Attribute	Data Type
team_ID	int NOT NULL
peson_ID	int NOT NULL

PRIMARY KEY: (team_ID, person_ID)
FOREIGN KEY: team_ID from *team*, person_ID from *person*

sports_match

Attribute	Data Type
match_ID	int NOT NULL UNIQUE
event_name	varchar(30) NOT NULL
event_start_date	DATE NOT NULL
sport_name	varchar(30) NOT NULL
start_date	DATE NOT NULL
end_date	DATE NOT NULL CHECK end_date >= start_date
start_time	TIME NOT NULL
end_time	TIME DEFAULT NULL

area	varchar(30) DEFAULT NULL
winner	int DEFAULT NULL
runner_up	int DEFAULT NULL

PRIMARY KEY: match_ID
FOREIGN KEY: (event_name, event_start_date, sport_name) from *Events_Sport*, winner from *participate*, runner_up from *participate*

match_team

Attribute	Data Type
team_ID	int NOT NULL
match_ID	int NOT NULL
score	varchar(30) DEFAULT NULL

PRIMARY KEY: (team_ID, match_ID)
FOREIGN KEY: team_ID from *participate*,match_ID from *sports_match*

events_sport

Attribute	Data Type
event_name	varchar(30) NOT NULL
start_date	DATE NOT NULL
sport_name	varchar(30) NOT NULL

status	varchar(100) NOT NULL
winner	int DEFAULT NULL
first_runner_up	int DEFAULT NULL
second_runner_up	int DEFAULT NULL

PRIMARY KEY: (event_name, start_date, sport_name)
FOREIGN KEY: (event_name, start_date) from *event*, sport_name from *sport*

participate

Attribute	**Data Type**
team_ID	int NOT NULL
event_name	varchar(30) NOT NULL
start_date	DATE NOT NULL
sport_name	varchar(30) NOT NULL

PRIMARY KEY: (team_ID, event_name, start_date, sport_name)
FOREIGN KEY: team_ID from *team*, (event_name, start_date, sport_name) from *events_sport*

Views, Indices and Triggers:

Views

1) **person_names**

```
CREATE VIEW person_names AS
SELECT person_ID, CONCAT(first_name, " ", COALESCE(CONCAT(middle_name, " "), ""),
last_name) as name
FROM person;
```

2) **person_contacts**

```
CREATE VIEW person_contacts AS
SELECT p.person_ID, CONCAT(p.first_name, " ", COALESCE(CONCAT(p.middle_name, " "),
""), p.last_name) as name, p.email, c.phone_number
FROM person p
LEFT JOIN
        (SELECT person_ID, phone_number
        FROM contact
        GROUP BY person_ID) as c
ON p.person_ID = c.person_ID;
```

3) **guard_info**

```
CREATE VIEW guard_info AS
SELECT guard_ID, CONCAT(first_name, " ", COALESCE(CONCAT(middle_name, " "), ""),
last_name) as name, contact_number, status
FROM guard;
```

4) **person_achievements**

```
CREATE VIEW person_achievements AS
SELECT p.person_ID, p.name, a.achievement_ID, a.award_name, a.achievement_date,
a.event_name, a.event_start_date, a.sport_name
FROM
        (SELECT ad.person_ID, am.achievement_ID, am.award_name, am.achievement_date,
am.event_name, am.event_start_date, am.sport_name
        FROM achievements as am
        JOIN
                achieved as ad
        ON am.achievement_ID = ad.achievement_ID) as a
JOIN
        person_names p
ON a.person_ID = p.person_ID;
```

5) **cricket_matches**

```
CREATE VIEW cricket_matches AS
SELECT team_ID, match_ID, SUBSTRING_INDEX(score, '/', 1) AS runs,
SUBSTRING_INDEX(score, '/', -1) AS wickets
FROM match_team
WHERE match_ID IN(
        SELECT match_ID
```

```
        FROM sports_match
        WHERE sport_name = 'Cricket');
```

6) **football_matches**

```
CREATE VIEW football_matches AS
SELECT team_ID, match_ID, score as goals
FROM match_team
WHERE match_ID IN(
        SELECT match_ID
        FROM sports_match
        WHERE sport_name = 'Football');
```

7) faculty_names

```
CREATE VIEW faculty_names AS
SELECT person_ID, name, academic_position
FROM person_names
NATURAL JOIN
faculty;
```

8) IIITD_student_info

```
CREATE VIEW IIITD_student_info AS
SELECT p.person_ID, p.name, p.email, s.roll_no, s.graduation_year, s.degree
FROM
        (SELECT person_ID, roll_no, graduation_year, degree
   FROM student
        WHERE college_name = "IIITD") as s
NATURAL JOIN
        person_contacts as p;
```

9) guest_info

```
CREATE VIEW guest_info AS
SELECT p.person_ID, p.name, p.email, s.roll_no, s.graduation_year, s.degree, s.college_name
FROM
        (SELECT *
   FROM student
        WHERE college_name <> "IIITD") as s
NATURAL JOIN
        person_contacts as p;
```

10) reserved_items_info

```
CREATE VIEW reserved_items_info AS
SELECT p.person_ID, p.name, r.start_date, r.start_time, r.item_ID, s.item_name, r.item_count
FROM
        sports_item as s
NATURAL JOIN
        person_names as p
NATURAL JOIN
        reserves_item as r;
```

11) reserved_area_info

```
CREATE VIEW reserved_area_info AS
SELECT p.person_ID, p.name, r.start_date, r.start_time, r.area_ID, s.playing_area
FROM
sports_area as s
NATURAL JOIN
person_names as p
NATURAL JOIN
reserves_area as r;
```

12) check_in_info

```
CREATE VIEW check_in_info AS
SELECT c.record_ID, p.person_ID, p.name as person_name, g.guard_ID, g.name as
guard_name, c.in_date, c.in_time, c.out_date, c.out_time
FROM
        check_in_record as c
NATURAL JOIN
        person_names as p
JOIN
        guard_info as g
ON g.guard_ID = c.guard_ID;
```

13) instructor_info

```
CREATE VIEW instructor_info AS
SELECT p.person_ID, p.name, i.sport_name, p.email, p.phone_number
FROM
        instructor i
NATURAL JOIN
        person_contacts p;
```

14) sc_member_info

```
CREATE VIEW sc_member_info AS
SELECT p.person_ID, p.name, p.email, p.phone_number, s.running_year
FROM sports_council_member s
NATURAL JOIN
        person_contacts p;
```

15) event_incharge

```
CREATE VIEW event_incharge AS
SELECT s.person_ID, s.name, s.email, s.phone_number, i.event_name, i.start_date
FROM incharge i
NATURAL JOIN
        sc_member_info s;
```

16) class_info

```
CREATE VIEW class_info AS
SELECT i.person_ID, i.name, i.email, i.phone_number, s.class_ID, s.sport_name, s.timings,
s.place, s.fee_charge
FROM instructor_info i
NATURAL JOIN
        teach t
```

```
NATURAL JOIN
        sports_class s;
```

17) check_in_slot

```
CREATE VIEW check_in_slot AS
SELECT record_ID, person_ID, guard_ID, in_date, out_date,
        CASE WHEN (in_time >= '07:00:00' AND in_time < '12:00:00') THEN 'Morning'
        ELSE
        CASE WHEN (in_time >= '12:00:00' AND in_time < '16:00:00') THEN 'Afternoon'
        ELSE
        CASE WHEN (in_time >= '16:00:00' AND in_time < '20:00:00') THEN 'Evening'
        ELSE
         CASE WHEN (in_time < '07:00:00' OR in_time >= '20:00:00') THEN 'Night'
        END
        END
        END
        END AS 'slot'
FROM check_in_record;
```

Database Indices

1) CREATE INDEX guard_status on **guard(status);**
2) CREATE INDEX class_idx on **teach(class_ID);**
3) CREATE INDEX event_name_start on **incharge(event_name,start_date);**
4) CREATE INDEX out_date_idx on **check_in_record(out_date);**
5) CREATE INDEX class_idx on **enroll(class_ID);**
6) CREATE INDEX running_yrx on **sports_council_member(running_year);**
7) CREATE INDEX date_and_status on **time_slot(start_date,status);**
8) CREATE INDEX host_and_date on **event(host,start_date);**
9) CREATE INDEX team_idx on **match_team(team_ID);**
10) CREATE INDEX person_idx on **achieved(person_ID);**

Triggers

1) ***verify_IIITD_Student***

```
DELIMITER //
CREATE TRIGGER verify_IIITD_Student
AFTER INSERT ON sports_council_member
FOR EACH ROW
        BEGIN
                IF NEW.person_ID NOT IN
                        (SELECT person_ID
                        FROM student
                        WHERE college_name = "IIITD")
                THEN
                        DELETE
                        FROM sports_council_member s
                        WHERE NEW.person_ID = s.person_ID
                        AND NEW.running_year = s.running_year;
                END IF;
        END;
```

2) ***reserve_item_availability***

```
DELIMITER //
CREATE TRIGGER reserve_item_availability
AFTER INSERT ON reserves_item
FOR EACH ROW
        BEGIN
                IF EXISTS (
        SELECT available
        FROM sports_item
        WHERE item_ID = NEW.item_ID AND NEW.item_count <= available)
                THEN
        UPDATE sports_item
        SET available = available - NEW.item_count
        WHERE item_ID = NEW.item_ID;
                ELSE
                        DELETE
        FROM reserves_item r
        WHERE r.person_ID = NEW.person_ID AND r.start_date = NEW.start_date AND
r.start_time = NEW.start_time AND r.item_ID = NEW.item_ID;
      END IF;
  END;
```

3) ***checked_item_availability***

```
DELIMITER //
CREATE TRIGGER checked_item_availability
AFTER INSERT ON tracks
FOR EACH ROW
        BEGIN
                IF EXISTS (
        SELECT available
```

```
        FROM sports_item
        WHERE item_ID = NEW.item_ID AND NEW.item_count <= available)
                THEN
        UPDATE sports_item
        SET available = available - NEW.item_count
        WHERE item_ID = NEW.item_ID;
                ELSE
                        DELETE
        FROM tracks t
        WHERE t.item_ID = NEW.item_ID AND t.record_ID = NEW.record_ID;
    END IF;
  END;
```

4) *check_winner_runnerUps*

```
DELIMITER //
CREATE TRIGGER check_winner_runnerUps
BEFORE UPDATE ON events_sport
FOR EACH ROW
                IF (NEW.winner is NOT NULL AND NEW.first_runner_up IS NOT NULL)
THEN
                        IF (NEW.second_runner_up is NOT NULL) THEN
                                IF (NEW.second_runner_up = NEW.winner OR
NEW.second_runner_up = NEW.first_runner_up) THEN
                                        SET NEW.second_runner_up = NULL;
                                END IF;
                        END IF;
        IF (NEW.winner = NEW.first_runner_up) THEN
                                SET NEW.winner = NULL;
          SET NEW.first_runner_up = NULL;
                        END IF;
                END IF;
         END;
```

5) *priority_check*

```
DELIMITER //
CREATE TRIGGER priority_check
AFTER INSERT ON sports_council_member
FOR EACH ROW
          BEGIN
                DECLARE latest_yr YEAR;
                SET latest_yr = (
                                SELECT max(running_year)
           FROM sports_council_member);

      SET SQL_SAFE_UPDATES = 0;
      UPDATE person
      SET priority = 0
      WHERE person_ID IN (
      SELECT person_ID
      FROM sports_council_member
      WHERE running_year = latest_yr - 1);
```

```
    UPDATE person
    SET priority = 2
    WHERE person_ID IN (
    SELECT person_ID
    FROM sports_council_member
    WHERE running_year = latest_yr);

    SET SQL_SAFE_UPDATES = 1;
  END;
```

6) unique_guard_username_insert

```
DELIMITER //
CREATE TRIGGER unique_guard_username_insert
AFTER INSERT ON guard
FOR EACH ROW
        BEGIN
                IF NEW.username IN
                        (SELECT username
                        FROM person)
                THEN
                        DELETE
                        FROM guard g
                        WHERE NEW.guard_ID = g.guard_ID ;
                END IF;
   END;
```

7) unique_guard_username_update

```
DELIMITER //
CREATE TRIGGER unique_guard_username_update
AFTER UPDATE ON guard
FOR EACH ROW
        BEGIN
                IF NEW.username IN
                        (SELECT username
                        FROM person)
                THEN
                        DELETE
                        FROM guard g
                        WHERE NEW.guard_ID = g.guard_ID ;
                END IF;
   END;
```

8) unique_person_username_insert

```
DELIMITER //
CREATE TRIGGER unique_person_username_insert
AFTER INSERT ON person
FOR EACH ROW
        BEGIN
                IF NEW.username IN
                        (SELECT username
```

```
                    FROM guard)
                THEN
                    DELETE
                    FROM person p
                    WHERE NEW.person_ID = p.person_ID ;
                END IF;
    END;
```

9) unique_person_username_update

```
DELIMITER //
CREATE TRIGGER unique_person_username_update
AFTER UPDATE ON person
FOR EACH ROW
        BEGIN
                IF NEW.username IN
                    (SELECT username
                    FROM guard)
                THEN
                    DELETE
                    FROM person p
                    WHERE NEW.person_ID = p.person_ID ;
                END IF;
    END;
```

Roles, Users And Permissions:

1) **Role:** IIITD Student
   Users: IIITD Students
   Permissions:
   SELECT, INSERT Permission - IIITD_student_info
   SELECT Permission - person_achievements
   SELECT Permission - guard_info
   SELECT Permission - sports_item
   SELECT Permission - sports_area
   SELECT Permission - event
   SELECT Permission - events_sport
   SELECT Permission - reserved_area_info
   SELECT Permission - reserved_items_info
   SELECT Permission - instructor_info
   SELECT Permission - sc_member_info
   SELECT Permission - event_incharge
   SELECT Permission - sports_match
   SELECT Permission - class_info

2) **Role:** Guest
   Users: Non-IIITians
   Permissions:
   SELECT, INSERT Permission - guest_info
   SELECT Permission - person_achievements
   SELECT Permission - guard_info
   SELECT Permission - event_incharge
   SELECT Permission - event
   SELECT Permission - events_sport
   SELECT Permission - sports_match

3) **Role:** Faculty
   Users: Faculty
   Permissions:
   SELECT, INSERT Permission - faculty_names
   SELECT Permission - person_achievements
   SELECT Permission - guard_info
   SELECT Permission - sports_item
   SELECT Permission - sports_area
   SELECT Permission - event
   SELECT Permission - events_sport
   SELECT Permission - reserved_area_info
   SELECT Permission - reserved_items_info
   SELECT Permission - instructor_info

SELECT Permission - sc_member_info
SELECT Permission - event_incharge
SELECT Permission - sports_match
SELECT Permission - class_info

4) **Role:** Sports Council Member
Users: Current Council Members
Permissions:
SELECT, UPDATE Permission - sc_member_info
SELECT, UPDATE Permission - person_achievements
SELECT, UPDATE Permission - guard_info
SELECT, UPDATE Permission - guest_info
SELECT, UPDATE Permission - IIITD_student_info
SELECT, UPDATE Permission - faculty_names
SELECT, UPDATE Permission - sports_item
SELECT, UPDATE Permission - sports_area
SELECT, UPDATE Permission - event
SELECT, UPDATE Permission - events_sport
SELECT, UPDATE Permission - reserved_area_info
SELECT, UPDATE Permission - reserved_items_info
SELECT, UPDATE Permission - instructor_info
SELECT, UPDATE Permission - event_incharge
SELECT, UPDATE Permission - class_info
SELECT, UPDATE, INSERT Permission - sports_match
SELECT, UPDATE Permission - time_slot
SELECT, UPDATE, INSERT Permission - team
SELECT, UPDATE, INSERT Permission - player
SELECT, UPDATE, INSERT Permission - participate
SELECT, UPDATE, INSERT Permission - match_team

5) **Role:** Sports Council Coordinator (Admin)
Users: Current Council Coordinator
Permissions:
ALL PRIVILEGES Permission - sc_member_info
ALL PRIVILEGES Permission - person_achievements
ALL PRIVILEGES Permission - guard_info
ALL PRIVILEGES Permission - guest_info
ALL PRIVILEGES Permission - IIITD_student_info
ALL PRIVILEGES Permission - faculty_names
ALL PRIVILEGES Permission - sports_item
ALL PRIVILEGES Permission - sports_area
ALL PRIVILEGES Permission - event
ALL PRIVILEGES Permission - events_sport
ALL PRIVILEGES Permission - reserved_area_info
ALL PRIVILEGES Permission - reserved_items_info
ALL PRIVILEGES Permission - instructor_info

ALL PRIVILEGES Permission - event_incharge
ALL PRIVILEGES Permission - sports_match
ALL PRIVILEGES Permission - class_info
ALL PRIVILEGES Permission - team
ALL PRIVILEGES Permission - player
ALL PRIVILEGES Permission - participate
ALL PRIVILEGES Permission - match_team
SELECT, UPDATE Permission - time_slot

6) **Role:** Instructor
Users: Instructors
Permissions:
SELECT, INSERT Permission - instructor_info
SELECT, INSERT Permission - class_info
SELECT Permission - person_achievements
SELECT Permission - guard_info
SELECT Permission - sports_item
SELECT Permission - sports_area
SELECT Permission - reserved_area_info
SELECT Permission - reserved_items_info

7) **Role:** Guard
Users: Guards
Permissions:
INSERT Permission - guard_info
SELECT, INSERT, UPDATE Permission - check_in_record
SELECT, INSERT, UPDATE Permission - reserved_area_info
SELECT, INSERT, UPDATE Permission - reserved_items_info
SELECT, INSERT, UPDATE Permission - check_in_info
SELECT, INSERT, UPDATE Permission - tracks
SELECT, UPDATE Permission - sports_item
SELECT, UPDATE Permission - sports_area
SELECT, UPDATE Permission - time_slot
SELECT Permission - sc_member_info
SELECT Permission - event_incharge
SELECT Permission - sports_match

Queries:

1) IIITD Student

a) What are the details (including point of contact) of the current sports council ?

```
SELECT p.name, sc.position, p.email, p.phone_number
FROM
        (SELECT person_ID, position
         FROM sports_council_member
        WHERE running_year='2020') as sc
INNER JOIN person_contacts as p
ON sc.person_ID = p.person_ID;
```

b) Who has the most achievements ?

```
SELECT name, COUNT(*) as 'number_of_achievements'
FROM person_achievements
GROUP BY person_ID
ORDER BY number_of_achievements DESC LIMIT 1;
```

c) Which time slots are available on '2021-04-20' ?

```
SELECT start_date, start_time, status
FROM time_slot
WHERE start_date='2021-04-20' AND status='Available';
```

d) On which time slots, is the 'Athletics Practice Field' booked ?

```
SELECT start_date, start_time
FROM
   (SELECT area_ID
   FROM sports_area
   WHERE playing_area = 'Athletics Practice Field') as a
JOIN
```

```
reserves_area as r

WHERE r.area_ID = a.area_ID;
```

e) Which all sports have an instructor ?

```
SELECT DISTINCT sport_name

FROM instructor;
```

f) What are the details of all swimming instructors who are teaching a sports class, along with the class timings and fee information ?

```
SELECT name, email, phone_number, timings, fee_charge
FROM
        (SELECT class_ID, timings, fee_charge
        FROM sports_class
        WHERE sport_name = 'Swimming') AS sp
INNER JOIN
        teach AS t
ON t.class_ID = sp.class_ID
INNER JOIN person_contacts AS p
ON t.person_ID=p.person_ID;
```

g) What are the names and contact numbers of all the guards on duty ?

```
SELECT name, contact_number
FROM guard_info
WHERE status = 'On';
```

h) Who all have achieved in more than two sports ?

```
SELECT DISTINCT a1.name
FROM person_achievements a1, person_achievements a2
WHERE a1.person_ID = a2.person_ID AND a1.sport_name != a2.sport_name;
```

i) How many intramural events are hosted by IIITD in the year 2020 ?

Select host, count(*) as No_of_events from Event as e inner join Event_categories as ec on e.event_name = ec.event_name where host = "IIITD" and YEAR(date) = 2020;

```
SELECT count(*) as 'No_of_events'
FROM
        (SELECT event_name
         FROM event_categories
        WHERE category = 'Intramural') as ec
INNER JOIN
        (SELECT event_name
        FROM event
        WHERE host = "IIITD" and YEAR(start_date) = 2020) as e
on e.event_name = ec.event_name;
```

j) How many goals per match on an average are scored by each team in football?

```
SELECT team_name, ROUND(AVG(goals), 2) AS 'Goals per match'
FROM football_matches m
NATURAL JOIN
        (SELECT team_ID, team_name
        FROM team
        WHERE sport_name = 'Football') as t
GROUP BY m.team_ID;
```

k) How many runs are scored by each team who has participated in cricket matches ?

```
SELECT team_name, SUM(runs) AS 'Total runs'
FROM cricket_matches m
NATURAL JOIN
        (SELECT team_ID, team_name
        FROM team
        WHERE sport_name = 'Cricket') as t
GROUP BY m.team_ID;
```

l) Who is in incharge of 'Triquetra' held on '17-01-2020' (with their point of contact) ?

```
SELECT name, email, phone_number
FROM
        (SELECT person_ID
        FROM incharge
        WHERE event_name = 'Triquetra' AND start_date = '2020-01-17') as i
JOIN
        person_contacts p
ON p.person_ID = i.person_ID;
```

2) Faculty

a) Which faculty has achieved the most in faculty tournaments so far ?

```
SELECT name
FROM person_achievements
WHERE event_name = "Faculty Tournament"
GROUP BY person_ID
ORDER BY COUNT(*) DESC LIMIT 1;
```

b) Who won the 'Badminton sport' in the faculty tournament held on '2019-06-01'?

```
SELECT name
FROM person_names
WHERE person_ID IN
        (SELECT person_ID
          FROM player
          WHERE team_ID IN
```

```
                (SELECT winner
                FROM events_sport
                WHERE event_name = "Faculty Tournament" and sport_name =
"Badminton" and start_date = "2019-06-01"));
```

c) Which faculty members have participated in only one sport ?

```
SELECT name
FROM
        (SELECT r1.person_ID, count(sport_name) as cnt
        FROM
                (SELECT person_ID, sport_name
                FROM participate
                NATURAL JOIN
                        player
                GROUP BY person_ID, sport_name) as r1
        GROUP BY r1.person_ID
        HAVING cnt = 1) as r2
NATURAL JOIN
        faculty_names;
```

d) What are the details of the earliest and latest faculty tournament ?

```
(SELECT *
FROM event
WHERE event_name = "Faculty Tournament"
ORDER BY start_date ASC LIMIT 1)

UNION

(SELECT *
FROM event
WHERE event_name = "Faculty Tournament"
ORDER BY  start_date DESC LIMIT 1);
```

3) Guard

a) Who has reserved ‘TT Table’ on ‘2021-04-22’ (and his/her contact number) ?

```
SELECT name
FROM
        (SELECT person_ID
   FROM reserves_area
   WHERE area_ID = (
                SELECT area_ID
      FROM sports_area
      WHERE playing_area = "TT Table")) as r
INNER JOIN
        person_names p
```

```
ON p.person_ID = r.person_ID;
```

b) How many people check out from the sports block daily (from 2020-04-20 to 2020-04-22)?

```
SELECT ROUND(AVG(c.check_ins),2) as average_check_ins_daily
FROM
        (SELECT out_date, COUNT(out_date) as check_ins
          FROM check_in_record
          WHERE out_date = "2020-04-20"
                UNION
          SELECT out_date, count(out_date) as check_ins
          FROM check_in_record
          WHERE out_date = "2020-04-21"
                UNION
        SELECT out_date, count(out_date) as check_ins
          FROM check_in_record
          WHERE out_date = "2020-04-22"
        GROUP BY out_date) as c;
```

4) Sports Council Member

a) Which is the least popular sports class ?

```
SELECT class_ID
FROM
        (SELECT class_ID, count(*) as cnt
        FROM enroll
        GROUP BY(class_ID)
        HAVING cnt =
                (SELECT min(r1.cnt)
                FROM
                        (SELECT class_ID, count(class_ID) as cnt
                        FROM enroll
                        GROUP BY(class_id)) as r1)) as r;
```

b) Which sport has been the most popular among students (based on no of check-ins) ?

```
SELECT sport_name
FROM sports_item as sp
NATURAL JOIN
  (SELECT item_ID, count(*) as cnt
  FROM tracks
  GROUP BY(item_ID)
  HAVING cnt =
                (SELECT MAX(r1.cnt)
```

```
        FROM
                (SELECT item_ID, count(*) as cnt
    FROM tracks
    GROUP BY(item_ID)) as r1)) as r;
```

c) Who all have enrolled in only one sports class ?

```
SELECT name
FROM person_names
WHERE person_ID IN
    (SELECT person_ID
    FROM enroll
    WHERE person_ID NOT IN
        (SELECT DISTINCT e1.person_ID
        FROM enroll e1, enroll e2
        WHERE e1.person_ID = e2.person_ID and e1.class_ID !=
e2.class_ID));
```

5) Instructor

a) What's the average number of wins per team in which there is at least one student of class 'GM01'?

```
SELECT ROUND(SUM(m.wins)/count(*),2) as 'Avg. Wins'
FROM
    (SELECT winner, count(*) as wins
    FROM sports_match
    WHERE winner in (
        SELECT team_ID
        FROM team
        WHERE sport_name IN (
            SELECT sport_name
            FROM sports_class
            WHERE class_ID = 'GM01')
        AND team_ID in (
        SELECT team_ID
        FROM player
        WHERE person_ID in (
            SELECT person_ID
            FROM enroll
            WHERE class_ID = 'GM01')))
    GROUP BY winner) as m;
```

b) What are the achievements of the students of class 'GM01' ?

```
SELECT name, award_name, achievement_date, event_name, event_start_date,
sport_name
FROM person_achievements
WHERE person_ID IN (
    SELECT person_ID
    FROM enroll
    WHERE class_ID = 'GM01');
```

Embedded Queries:

1) Editing and updating user information:

```
let save= function save(params){
    return new Promise(function (resolve, reject) {
        const response= {};
            connection.query('update person set middle_name = ? ,
last_name = ? , gender = ? where person_ID =
?',[params['mname'],params['lname'],params['gender'],params['id']
], function (err, result) {
                if (err) {
                    console.log(err);
                    response['code']=-1;
                    response['info']= "Oops! looks like we have
some problem with our database. Please try again";
                    reject(JSON.stringify(response));
                } else {
                    response['code']=1;
                    response['info']="updated database!";
                    console.log("updated");
                    resolve(JSON.stringify(response));
                }
            });
    });
};
```

2) Fetching guard information:

```
let GuardInfo= function GuardInfo(params){
    return new Promise(function (resolve, reject) {
        const response= {};
            connection.query(`select * from guard_info where
guard_ID in
            (select guard_ID from guard where
timings='`+params['timings']+`');`, function (err, result) {
                if (err) {
                    connection.end();
                    response['code']=-1;
                    response['info']= "Oops! looks like we have
some problem with our database. Please try again";
                    reject(JSON.stringify(response));
```

```
                } else {
                    response['guard']=[];
                    for (let i=0;i<result.length; i++) {
                        response['code']=1;
                        response['info']= "Account Details";
                        let obj= {};
                        obj['id']=result[i].guard_ID;
                        obj['name']=result[i].name;

obj['contact_number']=result[i].contact_number;
                        response['guard'].push(obj);

                    }

                    resolve(JSON.stringify(response));
                }
            })
    });
};
```

Function for filtering guard information on the basis of user input:

```
function filterResults() {
            let day_sel=document.getElementById('day_select');
            let
day=day_sel.options[day_sel.selectedIndex].value.trim();
            let
time_sel=document.getElementById('timing_select');
            let
time=time_sel.options[time_sel.selectedIndex].value.trim();
            let
shift_sel=document.getElementById('shift_select');
            let
shift=shift_sel.options[shift_sel.selectedIndex].value.trim();
            let
days=["Mon","Tues","Wed","Thu","Fri","Sat","Sun"];
            let times=["12AM-6AM", "6AM-12PM", "12PM-6PM",
"6PM-12AM"];
            let dayindex=0;
            for(let i=0;i<7;i++){
                if(days[i]==day){
                    dayindex=i;
```

```
                    break;
                }
            }
            for(let i=0;i<4;i++){
                if(times[i]==time){
                    if(i==3 && shift=="Next"){
                        time=times[0];
                        day=days[(dayindex+1)%7];
                    }else if(i==0 && shift=="Previous"){
                        time=times[3];
                        day=days[(dayindex-1+7)%7];
                    }else{
                        if(shift=="Next"){
                            time=times[(i+1)%4];
                        }else if(shift=="Previous"){
                            time=times[(i-1+4)%4];
                        }
                    }
                    break;
                }
            }
            var xhr = new XMLHttpRequest();
            xhr.open('POST', '/guardInfo');
            xhr.setRequestHeader('Content-Type',
'application/x-www-form-urlencoded');
            xhr.onload = function() {
                guardDetails=JSON.parse(xhr.responseText);
                console.log(guardDetails.guard[0]);
                showGuards(guardDetails);
            };
            xhr.send("timings=" + day +" "+ time);
        }
```

3) Fetching football matches related information:

```
let footballInfo= function footballInfo(params){
    return new Promise(function (resolve, reject) {
        const response= {};
            connection.query(`SELECT team_name, ROUND(AVG(goals),
2) AS 'Goals_per_match'
            FROM
            (
            SELECT team_ID, match_ID, score as goals
```

```
            FROM match_team
            WHERE match_ID IN(
                SELECT match_ID
                FROM sports_match
                WHERE sport_name = ? and event_name= ? )
            ) as m
            NATURAL JOIN
                (SELECT team_ID, team_name
                    FROM team
                    WHERE sport_name = ? ) as t
            GROUP BY
m.team_ID;`,[params['sport'],params['event'],params['sport']],
function (err, result) {
                if (err) {
                    connection.end();
                    response['code']=-1;
                    response['info']= "Oops! looks like we have
some problem with our database. Please try again";
                    reject(JSON.stringify(response));

                } else {
                    response['records']=[];
                    for (let i=0;i<result.length; i++) {
                        response['code']=1;
                        response['info']= "Team Details";
                        let obj= {};
                        obj['name']=result[i].team_name;

obj['av_goals']=result[i].Goals_per_match;
                        response['records'].push(obj);

                    }

                    resolve(JSON.stringify(response));
                }
            })
    });
};
```

4) Fetching cricket matches related information:

```
let cricketInfo= function cricketInfo(params){
    return new Promise(function (resolve, reject) {
        const response= {};
            connection.query(`SELECT team_name, SUM(runs) AS
'Total_runs'
            FROM (
                SELECT team_ID, match_ID, SUBSTRING_INDEX(score,
'/', 1) AS runs, SUBSTRING_INDEX(score, '/', -1) AS wickets
                FROM match_team
                WHERE match_ID IN(
                        SELECT match_ID
                        FROM sports_match
                        WHERE sport_name = ? and event_name = ? )
            )
            as m
            NATURAL JOIN
                (SELECT team_ID, team_name
                    FROM team
                    WHERE sport_name = ?) as t
            GROUP BY
m.team_ID;`,[params['sport'],params['event'],params['sport']],
function (err, result) {
                if (err) {
                    connection.end();
                    response['code']=-1;
                    response['info']= "Oops! looks like we have
some problem with our database. Please try again";
                    reject(JSON.stringify(response));

                } else {
                    response['records']=[];
                    for (let i=0;i<result.length; i++) {
                        response['code']=1;
                        response['info']= "Account Details";
                        let obj= {};
                        obj['name']=result[i].team_name;
                        obj['runs']=result[i].Total_runs;
                        response['records'].push(obj);

                    }

                    resolve(JSON.stringify(response));
                }
```

```
            })
    });
};
```

5) Fetching *Check-in* vs *time of the day* information:

```
let checkInEntries= function checkInEntries(){
    return new Promise(function (resolve, reject) {
        const response= {};
            connection.query(`select slot, sport_name,
count(slot) as entries from
            sports_item
            natural join (check_in_slot natural join tracks)
            group by slot,sport_name with rollup`, function (err,
result) {
                if (err) {
                    connection.end();
                    response['code']=-1;
                    response['info']= "Oops! looks like we have
some problem with our database. Please try again";
                    reject(JSON.stringify(response));

                } else {
                    response['entry']=[];
                    for (let i=0;i<result.length; i++) {
                        response['code']=1;
                        response['info']= "entry details";
                        let obj= {};
                        obj['slot']=result[i].slot;
                        obj['sport']=result[i].sport_name;
                        obj['count']=result[i].entries;
                        response['entry'].push(obj);

                    }

                    resolve(JSON.stringify(response));
                }
            })
    });
};
```