Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology
(Deemed to be University Estd. u/s 3 of UGC Act, 1956)
School of Computing
B.Tech. – Computer Science and Engineering

VTR UGE2021- (CBCS)

Academic Year: 2025–2026

SUMMER SEMESTER - SS2526

Course Code : 10211CS207

Course Name : Database Management Systems

Slot No : S1L12

DBMS TASK - 1 REPORT

Title: Conceptual Design through FTR

TNCA Management System

Abstract: The Tamil Nadu Cricket Association (TNCA), is the governing body for cricket in the Indian state of Tamil Nadu. Affiliated with the Board of Control for Cricket in India (BCCI), the TNCA plays a vital role in promoting, organizing, and overseeing cricket activities within the state. The primary objective of the TNCA is to develop and nurture cricket talent in Tamil Nadu, providing aspiring cricketers with opportunities to showcase their skills at various levels. The association conducts a wide range of domestic cricket tournaments, including first-class matches, one-day competitions, and T20 tournaments, which serve as a platform for players to compete and demonstrate their capabilities. One of the key responsibilities of the TNCA is to manage cricket infrastructure, including maintaining cricket stadiums, grounds, and training facilities to meet international standards. The association strives to create a conducive environment for cricket development, facilitating the growth of the sport in the region. Overall, the Tamil Nadu Cricket Council plays a pivotal role in fostering the spirit of cricket in the state, nurturing talent, and maintaining the rich cricketing heritage of Tamil Nadu. Through its dedicated efforts, the association continues to shape the future of cricket in the region and contribute significantly to the sport on a national and international level. Tamil Nadu Cricket Council Management System is a browser based solution that is designed to store, process, retrieve and analyse information concerned with the administrative, conducting tournaments and match aspects of providing services within a TNCA management system. The 'TNCA Management System' allows us to keep track of matches of tournament and also keeps track of players when requested.

TASK 1: Conceptual Design through FTR

Aim:

Using basic database design methodology and ER modeler, design Entity Relationship Diagram by satisfying the following sub tasks:

1. a Identifying the entities.
1. b Identifying the attributes.
1. c Identification of relationships, cardinality, type of relationship.
1. d Reframing the relations with keys and constraint.
1. e Using create, develop ER/ER diagram

1.a Identifying the entities

1.a.1 CricketBoard

1.a.2 Team

1.a.3 Player

1.a.4 Match

1.a.5 Ground

1.a.6 Umpire

1.b Identifyingtheattributes

1.b.1 CricketBoard(BoardID, Name, Address, Contact_No)

1.b.2 Team(TeamID, Name, Coach, Captain)

1.b.3 Player(PlayerID, FName, LName, Age, DateofBirth, PlayingRole, email, contact_no)

1.b.4 Match(MatchID, Date, Time, Result)

1.b.5 Ground(GroundID, Name, Location, Capacity)

1.b.6 Umpire(UmpireID, FName, LName, Age, DateofBirth, Country, email, contact_no)

1.c Identification of relationships, cardinality, type of relationship.

1.c.1 **Board-Team Relationship:** The Board will have a **one-to-many** relationship with Teams since the board can have multiple teams affiliated with it, but a team can only be associated with one board.

1.c.2 **Team-Player Relationship:** Teams and Players will have a **many-to-many** relationship since a team can have multiple players, and a player can be a part of multiple teams over time.

1.c,3. **Match-Team Relationship:** Matches will have a **many-to-many** relationship with Teams, as a match involves two teams, and a team can participate in multiple matches.

1.c.4. **Match-Ground Relationship:** Matches will have a **one-to-one** relationship with Grounds, as each match takes place in one specific ground.

1.d Reframing the relations with keys and constraint

1.d.1 Create Table CricketBoard:

SQL>create table CricketBoard(BoardID varchar(10) PRIMARY KEY, Name varchar(30), Address varchar(50), Contact_No number);

Table Created

SQL>DESC CricketBoard

Column	NULL	TYPE
BoardID	NOT NULL	VARCHAR(10)
Name	-	VARCHAR(30)
Address	-	VARCHAR(50)
Contact_No	-	NUMBER

1.d.2 Create Table Team:

SQL> create table Team(TeamID varchar(10) PRIMARY KEY, BoardID varchar(10), Name varchar(30), Coach varchar(30), Captain varchar(30), FOREIGN KEY(BoardID) REFERENCES CricketBoard(BoardID));

Table created.

SQL> DESC TEAM

Name	Null?	Type
TEAMID	NOT NULL	VARCHAR2(10)
BOARDID	NOT NULL	VARCHAR2(10)
NAME	-	VARCHAR2(30)
COACH	-	VARCHAR2(30)
CAPTAIN	-	VARCHAR2(30)

1.d.3 Create Table Player:

SQL> CREATE table Player(PlayerID varchar(6) PRIMARY KEY, TeamID varchar(10), FName varchar(30), LName varchar(30), Age number(5,2), DateofBirth date, PlayingRole varchar(25), email varchar(40), contact_no number, FOREIGN KEY(TeamID) REFERENCES Team(TeamID));

Table created.

SQL> DESC PLAYER

Name	Null?	Type
PLAYERID	NOT NULL	VARCHAR2(6)
TEAMID	NOT NULL	VARCHAR2(10)
FNAME		VARCHAR2(30)
LNAME		VARCHAR2(30)
AGE		NUMBER(5,2)
DATEOFBIRTH		DATE
PLAYINGROLE		VARCHAR2(25)
EMAIL		VARCHAR2(40)
CONTACT_NO		NUMBER

1.d.4 Create Table Match:

SQL> create table Match(MatchID varchar(10), TeamID1 varchar(10), TeamID2 varchar(10), Match_Date date, Time1 number, Result varchar(20), PRIMARY KEY(MatchID), FOREIGN KEY(TeamID1) REFERENCES team(TeamID), FOREIGN KEY(TeamID2) REFERENCES team(TeamID));

Table created.

SQL> DESC Match

Name	Null?	Type
MATCHID	NOT NULL	VARCHAR2(10)
TEAMID1	NOT NULL	VARCHAR2(10)
TEAMID2	NOT NULL	VARCHAR2(10)
PLAYERID	NOT NULL	VARCHAR2(6)
MATCH_DATE		DATE
TIME1		NUMBER
RESULT		VARCHAR2(20)

1.d.5 Create Table Ground:

SQL> create table Ground(GroundID varchar(10) PRIMARY KEY, MatchID Varchar(10), Name varchar(30), Location varchar(30), Capacity number, FOREIGN KEY(MatchID) REFERENCES Match(MatchID));

Table created.

SQL> DESC Ground

Name	Null?	Type
GROUNDID	NOT NULL	VARCHAR2(10)
MATCHID	NOT NULL	VARCHAR2(10)
NAME		VARCHAR2(30)
LOCATION		VARCHAR2(30)
CAPACITY		NUMBER

1.d.6 Create Table Umpire:

SQL> Create Table Umpire(UmpireID varchar(10) PRIMARY KEY, FName varchar(30), LName varchar(30), Age number(5,2), DateofBirth date, Country varchar(30), email varchar(40), contact_no number);

SQL> DESC Umpire

Name	Null?	Type
UMPIREID	NOT NULL	VARCHAR2(10)
FNAME		VARCHAR2(30)
LNAME		VARCHAR2(30)
AGE		NUMBER(5,2)
DATEOFBIRTH		DATE
COUNTRY		VARCHAR2(30)

EMAIL	VARCHAR2(40)
CONTACT_NO	NUMBER

1.d.6 Create Table Umpire_Umpired:

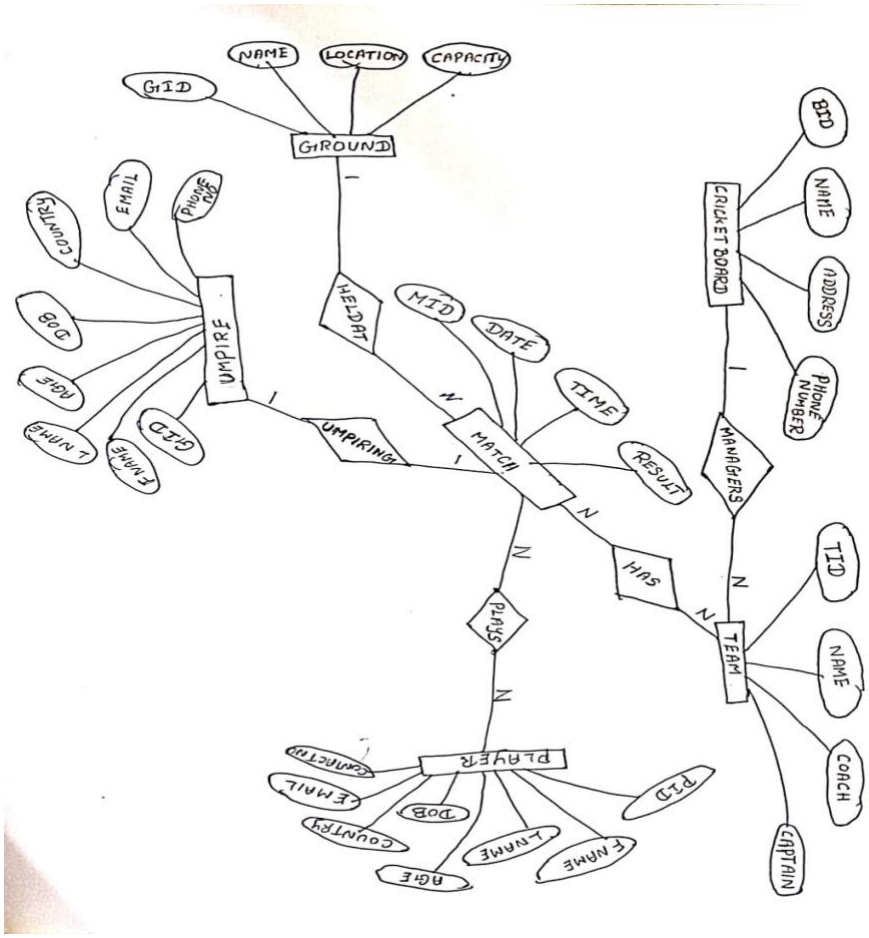
SQL> create table Umpire_Umpired(UmpireID varchar(10), MatchID Varchar(10), GroundID varchar(10), FOREIGN KEY(UmpireID) REFERENCES Umpire(UmpireID), FOREIGN KEY(MatchID) REFERENCES Match(MatchID), FOREIGN KEY(GroundID) REFERENCES Ground(GroundID));

Table created.

SQL> DESC Umpire

Name	Null?	Type
UMPIREID	NOT NULL	VARCHAR2(10)
GROUNDID	NOT NULL	VARCHAR2(10)
MATCHID	NOT NULL	VARCHAR2(10)

1.e. Using creately, develop ER/EER diagram



Result:

Thus the database design methodology and ER Model design diagram has been completed successfully.

Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology
(Deemed to be University Estd. u/s 3 of UGC Act, 1956)
School of Computing
B.Tech. – Computer Science and Engineering

VTR UGE2021- (CBCS)

Academic Year: 2025–2026

SUMMER SEMESTER -SS2526

Course Code : 10211CS207

Course Name : Database Management Systems

Slot No : S1L12

DBMS TASK - 2 REPORT

Title: Generating Design of other traditional database model

TASK 2

Generating Design of other traditional database model

Aim:

Creating Hierarchical/Network model of the database by enhancing the sound abstract data by performing following tasks using forms of inheritance:

a. Identify the specificity of each relationship, find and form surplus relations.

b. Check is-a hierarchy/has-a hierarchy and performs generalization and/or specialization relationship.

c. Find the domain of the attribute and perform check constraint to the applicable.

d. Rename the relations.

e. Perform SQL Relations using DDL, DCL commands.

a. Identify the specificity of each relationship, find and form surplus relations.

Entity Identification:

- CricketBoard has multiple Teams
- Team consists of multiple Players
- Match involves multiple Teams and is played on a Ground
- Umpire supervises the Match

Specificity Analysis:

- CricketBoard ↔ Team → One-to-Many
- Team ↔ Player → Many-to-Many → Team_Player
- Match ↔ Team → Many-to-Many → Match_Team
- Match ↔ Ground → One-to-One

Surplus Relations (Associative Tables):

- Team_Player(TeamID, PlayerID)
- Match_Team(MatchID, TeamID)

b. Check is-a hierarchy/has -a hierarchy and performs generalization and/or specialization relationship.

Generalization

In the ER diagram for the Tamil Nadu Cricket Board (TNCA) described earlier, we can identify potential generalizations based on common attributes or relationships among entities. Here's an example of a possible generalization:

Entities:

Player
Umpire

Attributes:

The above entities have common attributes like First_Name, Last_Name, Date_of_Birth, age, Contact_No, and Email.

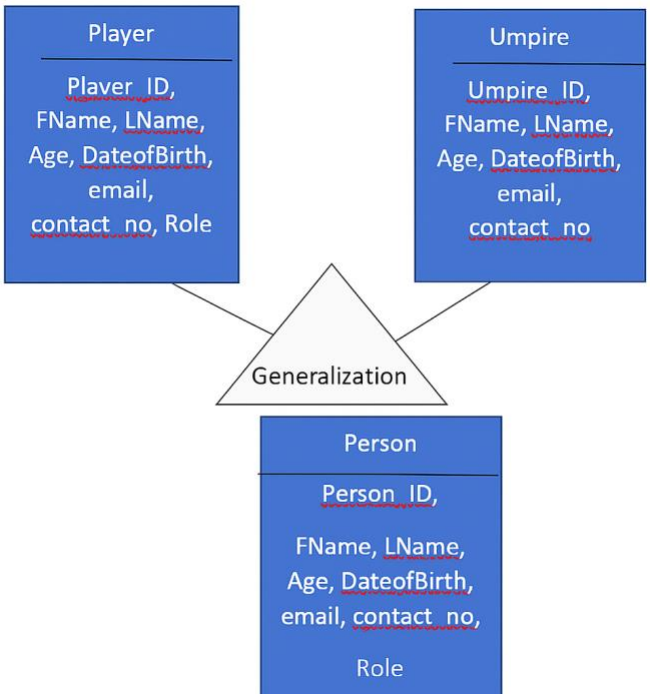
Potential Generalization:

Create a superclass called "Person" to represent the common attributes shared by Player and Umpire. The "Person" entity would have the following attributes:

Person_ID (primary key)
First_Name
Last_Name
Date_of_Birth
Age
Contact_Number
Email

Subclasses:

Player: Inherited attributes from "Person" and add specific attributes like Player_ID.
Umpire: Inherited attributes from "Person" and add specific attributes like Umpire_ID.



By using generalization, we can reduce data redundancy, improve data integrity, and simplify the structure of the ER diagram. This approach also allows for easier maintenance

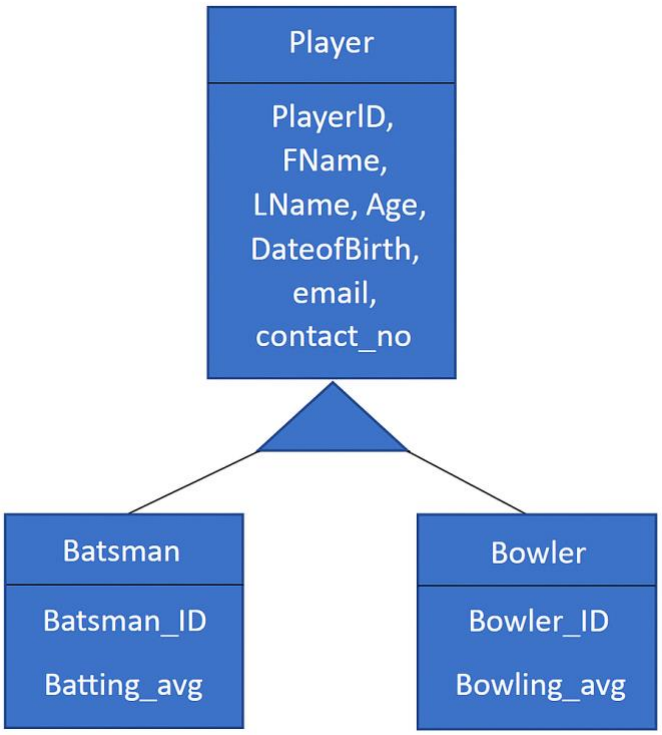
and updates, as changes made to the attributes shared by all "Person" entities will be automatically reflected in the subclasses.

Specialization

In the context of Entity-Relationship (ER) diagrams, specialization refers to the process of defining subtypes within an entity type. It allows, to represent entities that have specific attributes or relationships distinct from the general attributes or relationships of the parent entity.

In the case of the Tamil Nadu Cricket Board Association, let's consider the specialization of the "Player" entity into two subtypes: "Batsman" and "Bowler." This specialization is based on the specific roles that players can have in cricket.

Here's the modified ER diagram with the specialization:



2. c Find the domain of the attribute and perform check constraint to the applicable.

Attribute	Domain	Check Constraint Example
Age	Integer	CHECK (Age >= 18)

Contact_No	VARCHAR(10–15)	CHECK (LENGTH(Contact_No) BETWEEN 10 AND 15)
Email	VARCHAR	CHECK (Email LIKE '%@%.%')
Capacity	Integer	CHECK (Capacity > 0)
PlayingRole	VARCHAR	CHECK (PlayingRole IN ('Batsman', 'Bowler', 'All-Rounder', 'Wicket-Keeper'))

SQL> ALTER TABLE Player ADD CONSTRAINT check_con CHECK (age>= 18);

Table altered.

2.d Rename the relations:

Renaming a table (relation) in SQL can be accomplished using the ALTER TABLE statement with the RENAME TO clause. The specific syntax for renaming tables varies slightly between different database management systems.

Here's the syntax for renaming a column in the Table:

SQL> Alter table Umpire RENAME column contact_no TO phone_no;

Table altered.

SQL> DESC Umpire

Name	Null?	Type
UMPIREID		VARCHAR2(10)
FNAME		VARCHAR2(30)
LNAME		VARCHAR2(30)
AGE		NUMBER(5,2)
DATEOFBIRTH		DATE
COUNTRY		VARCHAR2(30)

EMAIL	VARCHAR2(40)
PHONE_NO	NUMBER

2.e Perform SQL Relations using DDL, DCL commands.

DCL stands for "Data Control Language," which is a subset of SQL (Structured Query Language) used to control access to data in a database. DCL commands are responsible for managing user permissions, granting privileges, and controlling data security within a database system. There are two primary DCL commands:

1. Grant
2. Revoke

GRANT:

The GRANT command is used to provide specific privileges to users or roles, allowing them to perform certain actions on database objects (e.g., tables, views, procedures). Privileges may include SELECT, INSERT, UPDATE, DELETE, EXECUTE, and more.

SQL> create user Raj identified by kumar;

User created.

SQL> grant resource to raj;

Grant succeeded.

SQL> grant create session to raj;

Grant succeeded.

SQL> conn

Enter user-name: raj

Enter password:

Connected.

SQL> create table emp(eno number,ename varchar(10));

Table created.

SQL> conn system/manager

Connected.

SQL> grant all on Umpire to Raj;

Grant succeeded.

Result:

Thus the Hierarchical model and Network model has been successfully created.

Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology
(Deemed to be University Estd. u/s 3 of UGC Act, 1956)
School of Computing
B.Tech. – Computer Science and Engineering

VTR UGE2021- (CBCS)

Academic Year: 2025–2026

SUMMER SEMESTER - SS2526

Course Code : 10211CS207

Course Name : Database Management Systems

Slot No : S1L12

DBMS TASK - 3 REPORT

Title: **Using Clauses, Operator sand Functions in queries**

TASK3

Using Clauses, Operators and Functions in queries

Aim:

To perform the query processing on databases for different retrieval results of queries using DML,DRLoperations usingaggregate,date,string,indentfunctions,setclausesandoperators.

- To retrieve the location of a particular match conducted by its MatchID
- To retrievethePlayersdetailwhose name start with 'A'.
- Addacolumn Batting and BowlinginPlayertable.
- To countthenumberof right-hand batsman in a team.
- To display the CricketBoard details for the BoardIDs 'BID01', 'BID03', and 'BID06'.
- To select the names and IDs of players who are left-hand bowlers.
- To find the UmpireID of umpires who have not umpired any match.

CricketBoard:

BoardID	Name	Address	Contact_No
BID01	Chennai Cricket Board	Chennai	9988776699
BID02	Tiruvallur Cricket Board	Chennai	9977886699
BID03	Viluppuram Cricket Board	Viluppuram	9966886699
BID04	Trichy Cricket Board	Trichy	9955886699
BID05	Madurai Cricket Board	Madurai	9944886699
BID06	Tuticorin Cricket Board	Tuticorin	9933886699
BID07	Selam Cricket Board	Selam	9922886699
BID08	Tiruppur Cricket Board	Tiruppur	9911886699

Team:

TeamID	BoardID	Name	Coach	Captain
CCB01	BID01	ABS EXPRESS	G.D.RAMESH	SAMPATH KUMAR
CCB02	BID01	AVG EXPRESS	T.KARTHIK	Y.JOHN

TCB01	BID02	ANGRY BARD	TOM BABU	CINIL JOHN
TCB02	BID02	TIGER ROCK	S.KANNAN	BEN GEORGE
TRICB01	BID04	ROCK	K.PAUL	K.MUTHU
VCB01	BID03	RAINBOW	S.RAJESHKUMAR	MANIMARAN
MCB01	BID05	PANTHER	SARAVANAN	R.SUNILKUMAR
TUCB01	BID06	THUNDER	D ALEX	BARATHI
SCB01	BID07	EAGLE	SOMU	SRI HARI
TICB01	BID08	KINGS	D ANAND	MATHAN

Player:

PlayerID	TeamID	FName	LName	Age	DateofBirth	PlayingRole	email	contact_no	Batting	Bowling
1	CCB01	Raj	N	27	29-Jun-96	Bowler	rajn@gmail.com	9191910101	null	left-hand
33	CCB01	Balaji	D	23	2-Jan-99	Batsman	balajid@gmail.com	9191910031	right-hand	*Null*
2	CCB02	Krishna	R	23	2-Jan-99	Bowler	krishnar@gmail.com	9191930103	*null*	right-hand
18	CCB02	Kishore	K	24	2-Sep-98	ALL ROUNDER	kishorek@gmail.com	9291930105	left-hand	left-hand
19	TCB01	Karthick	K	24	14-Sep-98	Batsman	karthickk@gmail.com	9292930107	right-hand	*Null*
62	TCB01	Amar	J	22	21-Sep-98	Batsman	Amarj@gmail.com	9292930508	right-hand	*Null*
102	TCB02	Akash	G	21	26-Sep-99	Batsman	Amarj@gmail.com	9292930510	right-hand	*Null*
12	TCB02	Premkumar	S	21	13-Oct-99	Bowler	Premkumars@gmail.com	9592930517	*null*	right-hand
1	VCB01	Prem	V	23	13-Apr-97	Bowler	Premkumars@gmail.com	9592950517	*null*	left-hand
21	VCB01	Kali	J	21	11-Apr-02	Batsman	Kalij@gmail.com	9592950630	right-hand	*Null*
61	VCB01	Kamalesh	A	22	21-Jun-01	Batsman	Kamalesha@gmail.com	9592958730	right-hand	*Null*
66	VCB01	Ganesh	V	24	21-Jun-98	Batsman	Ganeshv@gmail.com	9592958790	right-hand	*Null*
303	TRICB01	Arun	T	24	21-Oct-98	Batsman	Ganeshv@gmail.com	9592958450	right-hand	*Null*
313	TRICB01	Srinivasan	N	24	21-Oct-98	Batsman	srinivasann@gmail.com	9992958450	right-hand	*Null*

Match:

MatchID	TeamID	TeamID	Match_Date	Time1	Result
M01	CCB01	TCB01	22-Jun-22	1.3	TCB01 - WIN
M02	CCB02	TCB02	22-Jun-22	8.3	CCB01 - WIN
M03	TRICB01	TCB01	24-Jun-22	8.3	TCB01 - WIN
M04	TRICB01	TCB02	25-Jun-22	8.3	TRICB01 - WIN

Ground:

GroundID	MatchID	Name	Location	Capacity
GID01	M01	Nehru	Chennai	10000
GID02	M02	GK	Coimbatore	10000
GID03	M03	Sankar	Nellai	6000

Umpire:

UmpireID	FName	LName	Age	DateofBirth	Country	Email	contact_no
UID01	Venkatesh	T	45	21-Jun-78	INDIA	venkatesh@gmail.com	9665571435
UID02	Muthukumar	R	46	1-Jun-79	INDIA	mutukumarr@gmail.com	9665571460
UID03	Somu	K	42	1-Jun-83	INDIA	somuk@gmail.com	9664471460

Umpire_Umpired:

UmpireID	MatchID	GroundID
UID01	M01	GID01

UID02	M03	GID02

3.1: To retrieve the location of a particular match conducted by its MatchID

SQL> SELECT LOCATION FROM GROUND WHERE MatchID=’M03’;

Resul:

Location
Nellai

3.2: To retrievethePlayersdetailwhose name start with ‘A’.

SQL> Select * from Player where FName like 'A%';

PlayerID	TeamID	FName	LName	Age	DateofBirth	PlayingRole	Email	contact_no
62	TCB01	Amar	J	22	21-Sep-98	Batsman	Amarj@gmail.com	9292930508
102	TCB02	Akash	G	21	26-Sep-99	Batsman	Amarj@gmail.com	9292930510
303	TRICB01	Arun	T	24	21-Oct-98	Batsman	Ganeshv@gmail.com	9592958450

3.3: Addacolumn Batting and BowlinginPlayertable.

SQL> Alter table player add Batting varchar(10);

Table Altered

SQL> Alter table player add Bowling varchar(10);

Table Altered

3.4: To countthenumberof right-hand batsman in a team.

SQL> Select count(*) from Player where Batting='right-hand';

Result:

count(*)
9

3.5: To display the CricketBoard details for the BoardIDs 'BID01', 'BID03', and 'BID06'.

SQL> Select * from CricketBoard where BoardID in('BID01','BID03','BID06');

BoardID	Name	Address	Contact_No
BID01	Chennai Cricket Board	Chennai	9988776699
BID03	Viluppuram Cricket Board	Viluppuram	9966886699
BID06	Tuticorin Cricket Board	Tuticorin	9933886699

3.6: To select the names and IDs of players who are left-hand bowlers.

SQL> Select playerID, FName, LName from player where Bowling='left-hand';

PlayerID	FName	LName
1	Raj	N
18	Kishore	K
1	Prem	V

3.7: To find the UmpireID of umpires who have not umpired any match.

SQL> select a.UmpireID from Umpire a where UmpireID NOT IN(select UmpireID from Umpire_Umpired);

Result:

UmpireID
UID03

Result:

Thus the query processing on database for different retrieval result of query using Clauses, Operators and Functions in queries has been performed successfully.

Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology
(Deemed to be University Estd. u/s 3 of UGC Act, 1956)
School of Computing
B.Tech. – Computer Science and Engineering

VTR UGE2021- (CBCS)

Academic Year: 2025–2026

SUMMER SEMESTER - SS2526

Course Code : 10211CS207

Course Name : Database Management Systems

Slot No : S1L12

DBMS TASK - 4 REPORT

Title: Using Functions in Queries and Writing Sub Queries

TASK 4

Using Functions in Queries and Writing Sub Queries

Aim:

To perform the advanced query processing and test its heuristics using designing of optimalcorrelated and nested sub queries such as finding summary statistics.

1. To retrieve all team details, including the count of winning matches for each team
2. To retrieve the total number of 'Tie' matches in a team-wise manner.
3. To retrieve the team details who won the matches.
4. To retrieve players and match details of players who are above 25 years old.
5. To retrieve the details of Team who have not played any matches.
6. To retrieve the teamid, boardid, teamname, and playername for a particular playerid given.

4.1 To retrieve all team details, including the count of winning matches for each team.

SQL> SELECT t.TeamID, t.Name AS TeamName, t.Coach, t.Captain,(SELECT COUNT(m.MatchID)FROM Match m WHERE t.TeamID = SUBSTR(m.result, 1, 5)) AS WinningMatchCount FROM Team t;

Output:

TEAMID	TEAMNAME	COACH	CAPTAIN	WINNINGMATCHCOUNT
TRICB01	ROCK	K.PAUL	K.MUTHU	0
CCB02	AVG EXPRESS	T.KARTHIK	Y.JOHN	0
SCB01	EAGLE	SOMU	SRI HARI	0
MCB01	PANTHER	SARAVANAN	R.SUNILKUMAR	0
TUCB01	THUNDER	D ALEX	BARATHI	0
TCB01	ANGRY BARD	TOM BABU	CINIL JOHN	2
VCB01	RAINBOW	S.RAJESHKUMAR	MANIMARAN	0

TCB02	**TIGER ROCK**	**S.KANNAN**	**BEN GEORGE**	**0**
CCB01	**ABS EXPRESS**	**G.D.RAMESH**	**SAMPATH KUMAR**	**1**
TICB01	**KINGS**	**D ANAND**	**MATHAN**	**0**

10 rows selected.

4.2 To retrieve the total number of 'Tie' matches in a team-wise manner.

SQL> SELECT t.TeamID, t.Name AS TeamName,(SELECT COUNT(m.MatchID) FROM Match m WHERE(t.TeamID = m.TeamID1 OR t.TeamID = m.TeamID2) AND INSTR(m.Result, 'Tie') > 0) AS TieCount FROM Team t group by t.teamid, t.name;

TEAMID	TEAMNAME	TIECOUNT
TUCB01	THUNDER	0
TRICB01	ROCK	1
VCB01	RAINBOW	0
CCB02	AVG EXPRESS	0
MCB01	PANTHER	0
CCB01	ABS EXPRESS	0
TCB01	ANGRY BARD	0
TCB02	TIGER ROCK	1
SCB01	EAGLE	0
TICB01	KINGS	0

10 rows selected.

4.3 To retrieve the team details who won the matches.

SQL> SELECT * FROM team t WHERE t.teamid IN (SELECT teamid1 FROM match WHERE INSTR(result, teamid1) > 0 UNION SELECT teamid2 FROM match WHERE INSTR(result, teamid2) >)

TEAMID BOARDID NAME COACH CAPTAIN

---------- ---------- ------------------------------ ------------------------- -------------------------
---

TCB01 BID02 ANGRY BARD TOM BABU CINIL JOHN

TRICB01 BID04 ROCK K.PAUL K.MUTHU

2 rows selected.

4.4 To retrieve players and match details of players who are above 25 years old.

SQL> SELECT p.playerID,p.FName,p.LName,p.age,m.matchid,m.match_Date, m.Time1,m.result FROM player p, match m WHERE p.age > 25 AND (p.teamid = m.teamid1 OR p.teamid = m.teamid2);

PLAYER PLAYERNAME AGE MATCHID MATCH_DAT TIME1 RESULT

------ ------------------------- ---------- ---------- --------- ---------- --------------------

1 Raj 27 M01 22-JUN-22 1.3 TCB01 - WIN

4.5 To retrieve the details of Team who have not played any matches.

SQL> SELECT t.TeamID, t.Name AS TeamName, t.Coach, t.Captain FROM Team t WHERE t.TeamID NOT IN (SELECT TeamID1 FROM Match UNION SELECT TeamID2 FROM Match);

TEAMID TEAMNAME COACH CAPTAIN

---------- ------------------------------ ------------------------------ ------------------------------

VCB01 RAINBOW S.RAJESHKUMAR MANIMARAN

MCB01 PANTHER SARAVANAN R.SUNILKUMAR

TUCB01 THUNDER D ALEX BARATHI

SCB01	EAGLE	SOMU	SRI HARI
TICB01	KINGS	D ANAND	MATHAN

4.6 To retrieve the teamid, boardid, teamname, and playername for a particular playerid given.

SQL> SELECT t.teamid, t.boardid, t.name, p.fname FROM team t JOIN player p ON t.teamid = p.teamid WHERE p.playerid = '66';

TEAMID	BOARDID	NAME	FNAME
VCB01	BID03	RAINBOW	Ganesh

Result:

Thus the query using joins and writing subqueries has been done successfully.

Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology
(Deemed to be University Estd. u/s 3 of UGC Act, 1956)
School of Computing
B.Tech. – Computer Science and Engineering

VTR UGE2021- (CBCS)

Academic Year: 2025–2026

SUMMER SEMESTER - SS2526

Course Code : 10211CS207

Course Name : Database Management Systems

Slot No : S1L12

DBMS TASK- 5 REPORT

Title: **Writing Join Queries, equivalent, and/or recursive queries:**

TASK 5

Writing Join Queries, equivalent, and/or recursive queries:

(Tool: SQL/ Oracle, ALM: Flipped Classroom)

Aim: To Perform the advanced query processing and test its heuristics using designing of optimal correlated and nested sub queries such as finding summary statistics.

5.1 To retrieve all cricket boards and their teams.
5.2 To list all matches along with the teams and their captains.
5.3 To count the number of matches played by each team.
5.4 To find all the players who are part of the team named " TIGER ROCK ".
5.5 To retrieve all team details, including the count of winning matches for each team.
5.6 To retrieve the total number of 'Tie' matches in a team-wise manner.
5.7 To retrieve the team details who won the matches.
5.8 To retrieve players and match details of players who are above 25 years old.
5.9 To retrieve the details of Team who have not played any matches.
5.10 To retrieve the teamid, boardid, teamname, and playername for a particular playerid given.

5.1 To retrieve all cricket boards and their teams.

SQL> SELECT cb.Name AS CricketBoard, t.Name AS Team FROM CricketBoard cb JOIN Team t ON cb.BoardID = t.BoardID;

CricketBoard	Team
Chennai Cricket Board	ABS EXPRESS
Chennai Cricket Board	AVG EXPRESS

Tiruvallur Cricket Board	ANGRY BARD
Tiruvallur Cricket Board	TIGER ROCK
Trichy Cricket Board	ROCK
Viluppuram Cricket Board	RAINBOW
Madurai Cricket Board	PANTHER
Tuticorin Cricket Board	THUNDER
Selam Cricket Board	EAGLE
Tiruppur Cricket Board	KINGS

5.2 List all matches along with the teams and their captains.

SQL>SELECT m.matchid, m.match_date, m.result, t1.teamid AS team1_id, t1.name AS team1_name, t1.captain AS team1_captain, t2.teamid AS team2_id, t2.name AS team2_name, t2.captain AS team2_captain FROM match m JOIN team t1 ON m.teamid1 = t1.teamid JOIN team t2 ON m.teamid2 = t2.teamid ORDER BY m.matchid;

MATCHID	MATCH_DAT	RESULT	TEAM1_ID	TEAM1_NAME	TEAM1_CAPTAIN	TEAM2_ID	TEAM2_NAME	TEAM2_CAPTAIN
M01	22-JUN-22	TCB01 - WIN	CCB01	ABS EXPRESS	SAMPATH KUMAR	TCB01	ANGRY BARD	CINIL JOHN
M02	22-JUN-22	CCB01 - WIN	CCB02	AVG EXPRESS	Y.JOHN	TCB02	TIGER ROCK	BEN GEORGE

M03 24-JUN-22 TCB01 - WIN TRICB01 ROCK K.MUTHU TCB01 ANGRY BARD CINIL JOHN

M04 25-JUN-22 TRICB01 - WIN TRICB01 ROCK K.MUTHU TCB02 TIGER ROCK BEN GEORGE

M05 27-JUN-25 TRICB01 - Tie TRICB01 ROCK K.MUTHU TCB02 TIGER ROCK BEN GEORGE

5.3 Count the number of matches played each team.

SQL>SELECT t.teamid, t.name AS team_name, COUNT(m.matchid) AS matches_played FROM team t JOIN match m ON t.teamid = m.teamid1 OR t.teamid = m.teamid2 GROUP BY t.teamid, t.name ORDER BY matches_played DESC

TEAMID	TEAM_NAME	MATCHES_PLAYED
TRICB01	ROCK	3
TCB02	TIGER ROCK	3
TCB01	ANGRY BARD	2
CCB02	AVG EXPRESS	1
CCB01	ABS EXPRESS	1

SQL>

5.4 To find all the players who are part of the team named "TIGER ROCK".

SQL> SELECT p.playerID, p.fname, p.teamID, t.coach, t.captain FROM player p JOIN team t ON p.teamID = t.teamID WHERE t.name = 'TIGER ROCK';

PLAYER	FNAME	TEAMID	COACH	CAPTAIN
102	Akash	TCB02	S.KANNAN	BEN GEORGE
12	Premkumar	TCB02	S.KANNAN	BEN GEORGE

5.5 To retrieve all team details, including the count of winning matches for each team.

SQL> SELECT t.TeamID, t.Name AS TeamName, t.Coach, t.Captain, COUNT(m.MatchID) AS WinningMatchCount FROM Team t LEFT JOIN Match m ON t.TeamID = substr(m.result,1,5) GROUP BY t.TeamID, t.Name, t.Coach, t.Captain;

Output:

TEAMID	TEAMNAME	COACH	CAPTAIN	WINNINGMATCHCOUNT
TRICB01	ROCK	K.PAUL	K.MUTHU	0
CCB02	AVG EXPRESS	T.KARTHIK	Y.JOHN	0
SCB01	EAGLE	SOMU	SRI HARI	0
MCB01	PANTHER	SARAVANAN	R.SUNILKUMAR	0

TUCB01	THUNDER	D ALEX	BARATHI	0
TCB01	ANGRY BARD	TOM BABU	CINIL JOHN	2
VCB01	RAINBOW	S.RAJESHKUMAR	MANIMARAN	0
TCB02	TIGER ROCK	S.KANNAN	BEN GEORGE	0
CCB01	ABS EXPRESS	G.D.RAMESH	SAMPATH KUMAR	1
TICB01	KINGS	D ANAND	MATHAN	0

10 rows selected.

5.6 To retrieve the total number of 'Tie' matches in a team-wise manner.

SQL>SELECT t.teamid,t.name AS team_name,COUNT(m.matchid) AS tie_matchesFROM team tJOIN match mON (t.teamid = m.teamid1 OR t.teamid = m.teamid2)WHEREUPPER(m.result) LIKE '%TIE%'GROUP BY t.teamid, t.nameORDER BY tie_matches DESC;

TEAMID	TEAM_NAME	TIE_MATCHES
TCB02	TIGER ROCK	1
TRICB01	ROCK	1

5.7. To retrieve the team details who won the matches.

SQL>SELECT DISTINCT t.teamid, t.boardid, t.name AS team_name, t.coach, t.captainFROM team tJOIN match mON ((INSTR(m.result, m.teamid1) > 0 AND t.teamid = m.teamid1) OR (INSTR(m.result, m.teamid2) > 0 AND t.teamid = m.teamid2));

TEAMID	BOARDID	TEAM_NAME	COACH	CAPTAIN
TCB01	BID02	ANGRY BARD	TOM BABU	CINIL JOHN
TRICB01	BID04	ROCK	K.PAUL	K.MUTHU

5.8 To retrieve players and match details of players who are above 25 years old.

SQL>SELECT p.playerid,p.fname, p.lname, p.age, p.playingrole,t.teamid,t.name AS team_name,m.matchid,m.match_date,m.resultFROM player pJOIN team t ON p.teamid = t.teamidJOIN match mON (t.teamid = m.teamid1 OR t.teamid = m.teamid2)WHERE p.age > 25 ORDER BY p.playerid, m.matchid;

PLAYER	FNAME	LNAME	AGE	PLAYINGROLE	TEAMID	TEAM_NAME	MATCHID	MATCH_DAT	RESULT
1	Raj	N	27	Bowler	CCB01	ABS EXPRESS	M01	22-JUN-22	TCB01 - WIN

5.9 To retrieve the details of Team who have not played any matches.

SQL>SELECT t.teamid,t.boardid,t.name AS team_name, t.coach,t.captainFROM team tWHERE t.teamid NOT IN (SELECT teamid1 FROM matchUNIONSELECT teamid2 FROM match);

TEAMID	BOARDID	TEAM_NAME	COACH	CAPTAIN
VCB01	BID03	RAINBOW	S.RAJESHKUMAR	MANIMARAN
MCB01	BID05	PANTHER	SARAVANAN	R.SUNILKUMAR
TUCB01	BID06	THUNDER	D ALEX	BARATHI
SCB01	BID07	EAGLE	SOMU	SRI HARI
TICB01	BID08	KINGS	D ANAND	MATHAN

5.10 To retrieve the teamid, boardid, teamname, and playername for a particular playerid given.

SQL> SELECT t.teamid, t.boardid, t.name, p.fname FROM team t JOIN player p ON t.teamid = p.teamid WHERE p.playerid = '66';

TEAMID	BOARDID	NAME	FNAME
VCB01	BID03	RAINBOW	Ganesh

Result:

Thus the query using Join Queries, equivalent, and/or recursive queries has been done successfully.

Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology
(Deemed to be University Estd. u/s 3 of UGC Act, 1956)
School of Computing
B.Tech. – Computer Science and Engineering

VTR UGE2021- (CBCS)

Academic Year: 2025–2026

SUMMER SEMESTER -SS2526

Course Code : 10211CS207

Course Name : Database Management Systems

Slot No : S1L12

DBMS TASK - 6 REPORT

Title: Procedures, Functions And Loops

TASK 6: Procedures, Function and Loops

Aim: To write a programming using PL/SQL Procedures, Functions and loops on Number theory and businessscenarios like.

1. Write a PL/SQL block that calculates the average age of players and displays the result.
2. Write a PL/SQL block that inserts a new player record into the Player table.
3. To create a function that returns the total number of teams in a particular Cricket Board.
4. To write a non-recursive PL/SQL procedure to retrieve even-numbered PlayerIDs registered for any tournament.

Write a PL/SQL block that calculates the average age of players and displays the result.

```
DECLARE
  total_age NUMBER := 0;
  num_players NUMBER := 0;
  avg_age NUMBER := 0;
BEGIN
  -- Using a cursor to loop through all players
  FOR player_rec IN (SELECT Age FROM Player) LOOP
    total_age := total_age + player_rec.Age; -- Summing up the ages
    num_players := num_players + 1; -- Counting the number of players
  END LOOP;

  -- Calculating the average age
  IF num_players > 0 THEN
    avg_age := total_age / num_players;
  END IF;

  -- Displaying the result
  DBMS_OUTPUT.PUT_LINE('Total Players: ' || num_players);
  DBMS_OUTPUT.PUT_LINE('Total Age: ' || total_age);
```

```
        DBMS_OUTPUT.PUT_LINE('Average Age: ' || ceil(avg_age));
    END;
```

Output:

Total Players: 14

Total Age: 342

Average Age: 24

Write a PL/SQL block that inserts a new player record into the Player table.

```
DECLARE
   v_PlayerID VARCHAR(6) := '&Playerid'; -- You can generate a unique PlayerID as needed
   v_TeamID VARCHAR(10) := '&TEAMID'; -- Replace with the actual TeamID
   v_FName VARCHAR(30) := '&Fname';
   v_LName VARCHAR(30) := '&Lname';
   v_Age NUMBER(5,2) := &age;
   v_DateofBirth DATE := TO_DATE('&DOB', 'DD-Mon-YYYY'); -- Replace with the actual DateofBirth
   v_PlayingRole VARCHAR(25) := '&PlayingRole';
   v_email VARCHAR(40) := '&email';
   v_contact_no NUMBER := &phone; -- Replace with the actual contact number
   v_batting varchar(10) := '&batting';
   v_bowling varchar(10) := '&bowling';
BEGIN
   INSERT INTO Player (PlayerID, TeamID, FName, LName, Age, DateofBirth, PlayingRole, email, contact_no, batting, bowling)
   VALUES (v_PlayerID, v_TeamID, v_FName, v_LName, v_Age, v_DateofBirth, v_PlayingRole, v_email, v_contact_no, v_batting, v_bowling);
   COMMIT;
```

```
    DBMS_OUTPUT.PUT_LINE('Player record inserted successfully.');
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
        ROLLBACK;
END;
/
```

```
Enter the PlayerID: 676
Enter the TeamID: CCB01
Enter the FName: Rahul
Enter the LName: Sharma
Enter the Age: 23
Enter the DateofBirth: 17-07-1999
Enter the PlayingRole: AllRounder
Enter the email: rahulsharma@gmail.com
Enter the Contact_no: 9797181815

Player record inserted successfully.
PL/SQL procedure successfully completed.
```

To create a function that returns the total number of teams in a particular Cricket Board.

```
CREATE OR REPLACE FUNCTION GetTotalTeamsInBoard(BoardID1
VARCHAR2) RETURN NUMBER IS
    v_TotalTeams NUMBER := 0;
BEGIN
    SELECT COUNT(*) INTO v_TotalTeams FROM Team WHERE BoardID =
BoardID1;
    RETURN v_TotalTeams;
EXCEPTION
```

```
    WHEN NO_DATA_FOUND THEN
        -- Handle the case when the board doesn't exist or has no teams
        RETURN 0;
    WHEN OTHERS THEN
        -- Handle other exceptions as needed
        RETURN -1; -- Return a negative value to indicate an error
END GetTotalTeamsInBoard;
/
```

Function successfully created.

SQL>

```
Declare
Res number;
Begin
res:= GetTotalTeamsInBoard('BID01');
DBMS_OUTPUT.PUT_LINE('No of teams:  '||res);
END;
/
```

No of teams: 2

To write a non-recursive PL/SQL procedure to retrieve even-numbered PlayerIDs registered for any tournament.

```
CREATE OR REPLACE PROCEDURE GetEvenNumberedPlayerIDs IS
BEGIN
    FOR player_rec IN ( SELECT PlayerID FROM Player WHERE
MOD(TO_NUMBER(PlayerID), 2) = 0)
    LOOP
        DBMS_OUTPUT.PUT_LINE('Even-Numbered PlayerID: ' || player_rec.PlayerID);
```

```
    END LOOP;
END GetEvenNumberedPlayerIDs;
/
```

```
SQL> EXEC GetEvenNumberedPlayerIDs;
Even-Numbered PlayerID: 102
Even-Numbered PlayerID: 12
Even-Numbered PlayerID: 18
Even-Numbered PlayerID: 2
Even-Numbered PlayerID: 62
Even-Numbered PlayerID: 66
Even-Numbered PlayerID: 676

PL/SQL procedure successfully completed.
```

Result:

Thus the PL/SQL Procedures, Functions and loops on Number theory and business scenarios experiment was successfully completed and results are verified.

Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology
(Deemed to be University Estd. u/s 3 of UGC Act, 1956)
School of Computing
B.Tech. – Computer Science and Engineering

VTR UGE2021- (CBCS)

Academic Year: 2025–2026

SUMMER SEMESTER - SS2526

Course Code : 10211CS207

Course Name : Database Management Systems

Slot No : S1L12

DBMS TASK- 7 REPORT

Title: **Triggers, Views and Exceptions**

TASK 7:

Triggers, Views and Exceptions

Aim:

To Conduct events, views, exceptions on CRUD operations for restricting phenomenon.
a) To create a trigger in PL/SQL that automatically inserts a new record in the match_result table when a new record is inserted into the match table.
b) To create a view that displays the details of players along with their team details.
c) To write a non-recursive PL/SQL procedure to retrieve even-numbered PlayerIDs registered for any Match with exception handling.

a. To create a trigger in PL/SQL that automatically inserts a new record in the match_result table when a new record is inserted into the match table.

SQL>CREATE TABLE Match_Result (ResultIDvarchar(10) PRIMARY KEY, MatchIDvarchar(10), TeamIDvarchar(6), Result varchar(20), FOREIGN KEY(MatchID) REFERENCES Match(MatchID), FOREIGN KEY(TeamID) REFERENCES Team(TeamID));

```
SQL>CREATE OR REPLACE TRIGGER trg_insert_match_result AFTER INSERT ON Match
FOR EACH ROW
DECLARE
   N NUMBER;
BEGIN
N:=INSTR(UPPER(:NEW.RESULT),'WIN');
      IF N>0 AND INSTR(UPPER(:NEW.RESULT),UPPER(:NEW.TEAMID1))>0 THEN
           INSERT INTO Match_Result(ResultID, MatchID, TeamID, Result)
          VALUES ('R'||:NEW.MatchID||'1', :NEW.MatchID, :NEW.TeamID1, 'WIN');
          INSERT INTO Match_Result(ResultID, MatchID, TeamID, Result)
          VALUES ('R'||:NEW.MatchID||'2', :NEW.MatchID, :NEW.TeamID2, 'FAIL');
     ELSE
          INSERT INTO Match_Result(ResultID, MatchID, TeamID, Result)
          VALUES ('R'||:NEW.MatchID||'1', :NEW.MatchID, :NEW.TeamID1, 'FAIL');
          INSERT INTO Match_Result(ResultID, MatchID, TeamID, Result)
          VALUES ('R'||:NEW.MatchID||'2', :NEW.MatchID, :NEW.TeamID2, 'WIN');
     END IF;
END;
```

```
SQL>insert into  matchvalues('M01','CCB01','TCB01','22-JUN-2022',1.3,'TCB01 - WIN');
SQL>insert into  matchvalues('M02','CCB02','TCB02','22-JUN-2022',8.3,'CCB01 - WIN');
SQL>insert into  matchvalues('M03','TRIB01','TCB01','24-JUN-2022',8.3,'TCB01 - WIN');
SQL>insert into  matchvalues('M04','TRIB01','TCB02','25-JUN-2022',8.3,'TRICB01 - WIN');
```

SQL>insert into match values('M05','TCB01','TCB02',DATE '2025-09-20',16.00,'TCB02 - WIN');

SQL> Select * from Match_Result;

```
SQL> Select * from Match_Result;

RESULTID   MATCHID    TEAMID RESULT
---------- ---------- ------ --------------------
RM011      M01        CCB01  FAIL
RM012      M01        TCB01  WIN
RM021      M02        CCB02  FAIL
RM022      M02        TCB02  WIN
RM031      M03        TRIB01 FAIL
RM032      M03        TCB01  WIN
RM041      M04        TRIB01 FAIL
RM042      M04        TCB02  WIN
RM051      M05        TCB01  FAIL
RM052      M05        TCB02  WIN

10 rows selected.

SQL>
```

b. **To create a view that displays the details of players along with their team details.**

SQL>CREATE OR REPLACE VIEW Player_Team_View AS SELECT p.PlayerID, p.FName AS FirstName, p.PlayingRole AS PlayingRole, p.Batting AS BattingStyle, p.Bowling AS BowlingStyle, t.TeamID AS TeamID, t.Name AS TeamName, t.Coach AS Coach, t.Captain AS Captain FROM Player p JOIN Team t ON p.TeamID = t.TeamID;

SQL> Select * from Player_Team_View;

```
SQL> Select * from Player_Team_View;

PLAYER FIRSTNAME                      PLAYINGROLE              BATTINGSTY BOWLINGSTY TEAMID TEAMNAME
       COACH                               CAPTAIN
------ ------------------------------ ------------------------ ---------- ---------- ------ ----------
------------
1      Raj                            Bowler                              left-hand  CCB01  ABS EXPRESS
       G.D.RAMESH                          SAMPATH KUMAR
33     Balaji                         Batsman                  right-hand            CCB01  ABS EXPRESS
       G.D.RAMESH                          SAMPATH KUMAR
2      Krishna                        Bowler                              right-hand CCB02  AVG EXPRESS
       T.KARTHIK                           Y.JOHN
18     Kishore                        ALL ROUNDER              left-hand  left-hand  CCB02  AVG EXPRESS
       T.KARTHIK                           Y.JOHN
19     Karthick                       Batsman                  right-hand            TCB01  ANGRY BARD
       TOM BABU                            CINIL JOHN
62     Amar                           Batsman                  right-hand            TCB01  ANGRY BARD
       TOM BABU                            CINIL JOHN
102    Akash                          Batsman                  right-hand            TCB02  TIGER ROCK
       S.KANNAN                            BEN GEORGE
12     Premkumar                      Bowler                              right-hand TCB02  TIGER ROCK
       S.KANNAN                            BEN GEORGE
11     Prem                           Bowler                              left-hand  VCB01  RAINBOW
       S.RAJESHKUMAR                       MANIMARAN
21     Kali                           Batsman                  right-hand            VCB01  RAINBOW
       S.RAJESHKUMAR                       MANIMARAN
61     Kamalesh                       Batsman                  right-hand            VCB01  RAINBOW
       S.RAJESHKUMAR                       MANIMARAN

PLAYER FIRSTNAME                      PLAYINGROLE              BATTINGSTY BOWLINGSTY TEAMID TEAMNAME
       COACH                               CAPTAIN
------ ------------------------------ ------------------------ ---------- ---------- ------ ----------
------------
66     Ganesh                         Batsman                  right-hand            VCB01  RAINBOW
       S.RAJESHKUMAR                       MANIMARAN
303    Arun                           Batsman                  right-hand            TRIB01 ROCK
       K.PAUL                              K.MUTHU
313    Srinivasan                     Batsman                  right-hand            TRIB01 ROCK
       K.PAUL                              K.MUTHU

14 rows selected.
```

c. To write a non-recursive PL/SQL procedure to retrieve even-numbered PlayerIDs registered for any Matchwith exception handling.

```
CREATE OR REPLACE PROCEDURE Get_Even_PlayerIDs_For_Matches IS
  CURSOR even_players IS
    SELECT DISTINCT p.PlayerID, p.FName, p.LName, m.MatchID
    FROM Player p
    JOIN Team t ON p.TeamID = t.TeamID
    JOIN Match m ON (m.TeamID1 = t.TeamID OR m.TeamID2 = t.TeamID)
    WHERE MOD(TO_NUMBER(REGEXP_SUBSTR(p.PlayerID, '\d+$')), 2) = 0;

v_countNUMBER := 0;
BEGIN
  DBMS_OUTPUT.PUT_LINE('Even-numbered PlayerIDs registered for Matches:');

  FOR rec IN even_players LOOP
v_count := v_count + 1;
    DBMS_OUTPUT.PUT_LINE('MatchID: ' || rec.MatchID ||
                         ' | PlayerID: ' || rec.PlayerID ||
                         ' | Name: ' || rec.FName || ' ' || rec.LName);
  END LOOP;

  -- Handle case where no records found
  IF v_count = 0 THEN
    RAISE NO_DATA_FOUND;
```

```
    END IF;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No even-numbered PlayerIDs found for any match.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
/
```

```
SQL> set serveroutput on
SQL> /
```

Procedure created.

```
SQL>SET SERVEROUTPUT ON;
```

```
SQL>EXEC Get_Even_PlayerIDs_For_Matches;
```

```
SQL> EXEC Get_Even_PlayerIDs_For_Matches;
Even-numbered PlayerIDs registered for Matches:
MatchID: M02 | PlayerID: 2 | Name: Krishna R
MatchID: M02 | PlayerID: 18 | Name: Kishore K
MatchID: M01 | PlayerID: 62 | Name: Amar J
MatchID: M03 | PlayerID: 62 | Name: Amar J
MatchID: M05 | PlayerID: 62 | Name: Amar J
MatchID: M02 | PlayerID: 102 | Name: Akash G
MatchID: M04 | PlayerID: 102 | Name: Akash G
MatchID: M05 | PlayerID: 102 | Name: Akash G
MatchID: M02 | PlayerID: 12 | Name: Premkumar S
MatchID: M04 | PlayerID: 12 | Name: Premkumar S
MatchID: M05 | PlayerID: 12 | Name: Premkumar S

PL/SQL procedure successfully completed.

SQL>
```

Result:
Thus the Triggers,Views andExceptions experiment was successfully completed results are verified.

Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology
(Deemed to be University Estd. u/s 3 of UGC Act, 1956)
School of Computing
B.Tech. – Computer Science and Engineering

VTR UGE2021- (CBCS)

Academic Year: 2025–2026

SUMMER SEMESTER -SS2526

Course Code : 10211CS207

Course Name : Database Management Systems

Slot No : S1L12

DBMS TASK - 8 REPORT

Title: Normalizing databases using functional dependencies upto Third Normal Form

Task 8

Normalizing databases using functional dependencies upto Third Normal Form

Aim: To normalize the below relation and create the simplified table with suitable constraint.

CricketBoard(BoardID, Name, Address, Contact_No, TeamID, TName, Coach, Captain, PlayerID, PFName, PLName, Age, PDateofBirth, PlayingRole, email, contact_no, Batting, Bowling, MatchID, Match_Date, Time1, Result, GroundID, GName, Location, Capacity, UmpireID, UFName, ULName, UAge, UDateofBirth, Country, Uemail, Ucontact_no).

a) Apply the functional dependency, normalize to 1NF
b) Normalize the relations using FD+ and α+.
c) Find the minimal cover, canonical cover.
d) Normalize to 2NF, add/alter constraints if necessary.
e) Normalize to 3NF, add/alter constraints if necessary.

Procedure:

Normalize the given relation and create simplified tables with suitable constraints, we need to identify the functional dependencies and separate them into different tables. Normalization involves breaking down the data into smaller, related tables to minimize data redundancy and maintain data integrity. Let's identify the functional dependencies:

Functional Dependency:

BoardID → Name, Address, Contact_No
TeamID → TName, Coach, Captain
PlayerID → PFName, PLName, Age, PDateofBirth, PlayingRole, email, contact_no, Batting, Bowling
MatchID → Match_Date, Time1, Result, GroundID
GroundID → GName, Location, Capacity
UmpireID → UFName, ULName, UAge, UDateofBirth, Country, Uemail, Ucontact_no

Now, we can create simplified tables:

CricketBoard (BoardID [PK], Name, Address, Contact_No)

CricketTeam (TeamID [PK], TName, Coach, Captain)
CricketPlayer (PlayerID [PK], TeamID [FK], PFName, PLName, Age, PDateofBirth, PlayingRole, email, contact_no, Batting, Bowling)
CricketMatch (MatchID [PK], TeamID [FK], Match_Date, Time1, Result, GroundID [FK])
CricketGround (GroundID [PK], GName, Location, Capacity)
CricketUmpire (UmpireID [PK], UFName, ULName, UAge, UDateofBirth, Country, Uemail, Ucontact_no)

In these tables, [PK] denotes the primary key, [FK] denotes the foreign key, and suitable constraints should be added to maintain data integrity.

Create tables for all non-prime attributes using α+

α+ (Alpha Plus) allows to group attributes based on their functional dependencies and candidate keys. And create tables for each set of attributes that functionally depend on a candidate key. The candidate keys in this case are BoardID, TeamID, PlayerID, MatchID, and UmpireID.

CricketBoard Table: BoardID (PK), Name, Address, Contact_No
Team Table: TeamID (PK), TName, Coach, Captain
Player Table: PlayerID (PK), TeamID (FK), PFName, PLName, Age, PDateofBirth, PlayingRole,
Email, contact_no, Batting, Bowling
Match Table: MatchID (PK), TeamID (FK), Match_Date, Time1, Result
Ground Table: GroundID (PK), GName, Location, Capacity
Umpire Table: UmpireID (PK), UFName, ULName, UAge, UDateofBirth, Country, Uemail, Ucontact_no

Create additional tables to represent transitive dependencies.

Already addressed transitive dependencies in previous normalization steps by introducing the MatchVenue table for the transitive dependency between MatchID and GroundID through the Result attribute.

MatchVenue Table: MatchID (PK, FK), GroundID (FK)

First Normal Form:

The given relation into the First Normal Form (1NF), to need to ensure that each attribute (column) contains atomic (indivisible) values, and there are no repeating groups or arrays.

Based on the provided relation, it appears that each attribute already contains atomic values, so there are no repeating groups to eliminate.

Second Normal Form:

To determine whether the given relation is in the Second Normal Form (2NF), we need to check two conditions:

The relation must already be in 1NF (First Normal Form).

All non-prime attributes (attributes not part of any candidate key) must be fully functionally dependent on the entire primary key.

First, let's identify the potential candidate key(s) from the given relation based on functional dependencies:

It appears that the potential candidate keys could be:

1. BoardID
2. TeamID
3. PlayerID
4. MatchID
5. UmpireID

Next, we need to check if all non-prime attributes are fully functionally dependent on their respective candidate key(s).

Third Normal Form:

To determine whether the given relation is in the Third Normal Form (3NF), need to check two conditions:

1. The relation must already be in the Second Normal Form (2NF).
2. There should be no transitive dependencies between non-prime attributes and candidate keys.

The given relation satisfies the conditions of the Second Normal Form (2NF). Now, let's check for transitive dependencies:

Now, let's analyze each functional dependency and check for transitive dependencies:

BoardID → Name, Address, Contact_No

There are no transitive dependencies in this case, as Name, Address, and Contact_No are directly dependent on BoardID.

TeamID → TName, Coach, Captain

There are no transitive dependencies here either, as TName, Coach, and Captain are directly dependent on TeamID.

PlayerID → PFName, PLName, Age, PDateofBirth, PlayingRole, email, contact_no, Batting, Bowling

There are no transitive dependencies for PlayerID, as all the mentioned attributes are directly dependent on PlayerID.

MatchID → Match_Date, Time1, Result, GroundID

There is a transitive dependency between MatchID and GroundID through the Result attribute. To resolve this, we create a new table called MatchVenue:
MatchVenue (MatchID [PK], GroundID [FK])

GroundID → GName, Location, Capacity

There are no transitive dependencies for GroundID, as GName, Location, and Capacity are directly dependent on GroundID.

UmpireID → UFName, ULName, UAge, UDateofBirth, Country, Uemail, Ucontact_no

There are no transitive dependencies for UmpireID, as UFName, ULName, UAge, UDateofBirth, Country, Uemail, and Ucontact_no are directly dependent on UmpireID.

With the introduction of the MatchVenue table to resolve the transitive dependency, the relation now satisfies the conditions of the Third Normal Form (3NF).

Result:

Thus the normalization of the given relation is created the simplified tables with suitable constraint successfully.

Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology
(Deemed to be University Estd. u/s 3 of UGC Act, 1956)
School of Computing
B.Tech. – Computer Science and Engineering

VTR UGE2021- (CBCS)

Academic Year: 2025–2026

SUMMER SEMESTER - SS2526

Course Code : 10211CS207

Course Name : Database Management Systems

Slot No :S1L12

DBMS TASK - 10 REPORT

Title: CRUD operations in Document databases

TASK 10

CRUD operations in Document databases

AIM:

To Perform Mongoose using NPM design on MongoDB designing document database and performing CRUD operations like creating, inserting, querying, finding, removing operations

STEPS:

Step 1: Install Mongo db using following link

https://www.mongodb.com/try/download/community

Step 2: install Mongosh using the below link

https://www.mongodb.com/docs/mongodb-shell/#download-and-install-mongosh

Step 3: To add the MongoDB Shell binary's location to your PATH environment variable:

3.1 Open the Control Panel.

3.2 In the System and Security category, click System.

3,3 Click Advanced system settings. The System Properties modal displays.

3.4 Click Environment Variables.

3.5 In the System variables section, select path and click Edit. The Edit environment

variable modal displays.

3.6 Click New and add the filepath to your mongosh binary.

3.7 Click OK to confirm your changes. On each other modal, click OK to confirm

your changes.

Step 4: To confirm that your PATH environment variable is correctly configured to find

mongosh, open a command prompt and enter the mongosh --help command. If your

PATH is configured correctly, a list of valid commands displays.

Step 5: Open mongo shell 4.0 from
c:\programfiles\mongoDB\server\bin\mongod.exe

Step 6: Type the CRUD(CREATE READ UPDATE DELETE) COMMANDS GIVEN IN TEXT FILE.

CRUD OPERATIONS:

```
db.createCollection("CricketBorad")
{ "ok" : 1 }
> db.CricketBoard.insertOne({BoardID: "BID01", Name:"Chennai Cricket Board",
Address:"Chennai",Phone:9988776699});
{
	"acknowledged" : true,
	"insertedId" : ObjectId("651cf1726ebbfe7993adf909")
}
db.mylab.find({BoardID:"BID01"})
{ "_id" : ObjectId("651cf1726ebbfe7993adf909"), "BoardID" : "BID01", "Name" :
"Chennai Cricket Board", "Address" : "Chennai", "Phone" : 9988776699 }

db.CricketBoard.insertMany([{BoardID:"BID02",Name:"Tiruvallur Cricket
Board",Address:"Chennai",Phone:9977886699},{BoardID:"BID03",Name:"Viluppuram
Cricket
Board",Address:"Viluppuram",Phone:9966886699},{BoardID:"BID04",Name:"Trichy
Cricket
Board",Address:"Trichy",Phone:9955886699},{BoardID:"BID05",Name:"Madurai
Cricket Board",Address:"Madurai",Phone:9944886699}]);

{
	"acknowledged" : true,
	"insertedIds" : [
		ObjectId("651ceee36ebbfe7993adf904"),
```

```
            ObjectId("651ceee36ebbfe7993adf905"),
            ObjectId("651ceee36ebbfe7993adf906"),
            ObjectId("651ceee36ebbfe7993adf907")
        ]
}
```

```
db.CricketBoard.find()
```

```
{ "_id" : ObjectId("651ceee36ebbfe7993adf904"), "BoardID" : "BID02", "Name" :
"Tiruvallur Cricket Board", "Address" : "Chennai", "Phone" : 9977886699 }

{ "_id" : ObjectId("651ceee36ebbfe7993adf905"), "BoardID" : "BID03", "Name" :
"Viluppuram Cricket Board", "Address" : "Viluppuram", "Phone" : 9966886699 }

{ "_id" : ObjectId("651ceee36ebbfe7993adf906"), "BoardID" : "BID04", "Name" :
"Trichy Cricket Board", "Address" : "Trichy", "Phone" : 9955886699 }

{ "_id" : ObjectId("651ceee36ebbfe7993adf907"), "BoardID" : "BID05", "Name" :
"Madurai Cricket Board", "Address" : "Madurai", "Phone" : 9944886699 }

{ "_id" : ObjectId("651cf1726ebbfe7993adf909"), "BoardID" : "BID01", "Name" :
"Chennai Cricket Board", "Address" : "Chennai", "Phone" : 9988776699 }
```

```
db.CricketBoard.find().pretty()
{
        "_id" : ObjectId("651ceee36ebbfe7993adf904"),
        "BoardID" : "BID02",
        "Name" : "Tiruvallur Cricket Board",
        "Address" : "Chennai",
        "Phone" : 9977886699
}
{
        "_id" : ObjectId("651ceee36ebbfe7993adf905"),
        "BoardID" : "BID03",
        "Name" : "Viluppuram Cricket Board",
        "Address" : "Viluppuram",
        "Phone" : 9966886699
}
{
        "_id" : ObjectId("651ceee36ebbfe7993adf906"),
```

```
        "BoardID" : "BID04",
        "Name" : "Trichy Cricket Board",
        "Address" : "Trichy",
        "Phone" : 9955886699
}
{
        "_id" : ObjectId("651ceee36ebbfe7993adf907"),
        "BoardID" : "BID05",
        "Name" : "Madurai Cricket Board",
        "Address" : "Madurai",
        "Phone" : 9944886699
}
{
        "_id" : ObjectId("651cf1726ebbfe7993adf909"),
        "BoardID" : "BID01",
        "Name" : "Chennai Cricket Board",
        "Address" : "Chennai",
        "Phone" : 9988776699
}
```

```
db.CricketBoard.deleteOne({BoardID:"BID03"})
{ "acknowledged" : true, "deletedCount" : 1 }
db.CricketBoard.find().pretty()
{
        "_id" : ObjectId("651ceee36ebbfe7993adf904"),
        "BoardID" : "BID02",
        "Name" : "Tiruvallur Cricket Board",
        "Address" : "Chennai",
        "Phone" : 9977886699
}
{
        "_id" : ObjectId("651ceee36ebbfe7993adf906"),
```

"BoardID" : "BID04",

"Name" : "Trichy Cricket Board",

"Address" : "Trichy",

"Phone" : 9955886699

}

{

"_id" : ObjectId("651ceee36ebbfe7993adf907"),

"BoardID" : "BID05",

"Name" : "Madurai Cricket Board",

"Address" : "Madurai",

"Phone" : 9944886699

}

{

"_id" : ObjectId("651cf1726ebbfe7993adf909"),

"BoardID" : "BID01",

"Name" : "Chennai Cricket Board",

"Address" : "Chennai",

"Phone" : 9988776699

}

Result:

Thus CRUD using NPM design on MongoDB designing document database and performing CRUD operations like creating, inserting, querying, finding, removing operations are performed.

Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology
(Deemed to be University Estd. u/s 3 of UGC Act, 1956)
School of Computing
B.Tech. – Computer Science and Engineering

VTR UGE2021- (CBCS)

Academic Year: 2025–2026

SUMMER SEMESTER - SS2526

Course Code : 10211CS207

Course Name : Database Management Systems

Slot No : S1L12

DBMS TASK - 11 REPORT

Title: **CRUD operations in Graph databases**

TASK 11

CRUD operations in Graph databases

AIM:

To perform CRUD operations like creating, inserting, querying, finding, deleting operations on graph spaces.

The steps to get started with Neo4j's Aura Graph Database:

Step1: Copy and paste the following link into your web browser: https://neo4j.com/cloud/platform/aura-graph-database/?ref=docs-get-started-dropdown

Step2: Click on "Start Free."

Step3: Choose the option to "Continue with Google."

Step4: Click the "Open" button.

Step5: After clicking "Open," a text file will be automatically downloaded. This file contains your user ID and password details.

Step6: Copy the password from the downloaded text file and paste it where required.

Step7: Close the "Get started with Neo4j with beginner guides" if it's open.

Sep8: You're now ready to begin practicing with the Graph Database.

Create Node with Properties

Properties are the key-value pairs using which a node stores data. Create a node with properties using the CREATE clause and need to specify these properties separated by commas within the flower braces "{ }".

Syntax

```
CREATE (node:label{ key1: value, key2: value, . . . . . }) return node
```

To verify the creation of the node, type and execute the following query in the dollar prompt.

Syntax:

```
MATCH (n) RETURN n
```

Creating Relationships

To create a relationship using the CREATE clause and specify relationship within the square braces “[ ]” depending on the direction of the relationship it is placed between hyphen “ - ” and arrow “ → ” as shown in the following syntax.

Syntax:

```
CREATE (node1)-[:RelationshipType]->(node2)
```

Syntax:

```
MATCH (a:LabeofNode1), (b:LabeofNode2)
  WHERE a.name = "nameofnode1" AND b.name = " nameofnode2"
CREATE (a)-[: Relation]->(b) RETURN a,b
```

Deleting a Particular Node

To delete a particular node and need to specify the details of the node in the place of “n” in the above query.

Syntax:

```
MATCH (node:label {properties . . . . . }) DELETE node
```

Create a graph database for student course registration, create student and dept node and insert values of properties.

Create a CrickerBoard Node:

```
create(cb:CricketBoard{BoardID:'BID01',Name:'Chennai Cricket Board', Address:'Chennai',
Phone:9988776699})  returncb
```

Create Team Nodes:

```
create(t1:Team{teamID:'CCB01',BoardID:'BID01',name:'ABS EXPRESS',
Coach:'G.D.RAMESH', Captain:'SAMPATH KUMAR'}) return t1

create(t2:Team{teamID:'CCB02',BoardID:'BID01',name:'AVG EXPRESS',Coach:
'T.KARTHIKH', Captain:'Y.JOHN'}) return t2
```

Create Player Nodes:

```
create(p1:Player{PlayerID:'1',TeamID:'CCB01',Name:'Raj',Age:23,DateofBirth:'29-JUN-1996',
PlayingRole:'Bowler',email:'rajn@gmail.com'}) return p1

create(p2:Player{PlayerID:'33',TeamID:'CCB01',Name:'Anand',Age:23,DateofBirth:'02-JAN-
1999', PlayingRole:'Batsman',email:'balajid@gmail.comm'}) return p2
```

create(p3:Player{PlayerID:'65',TeamID:'CCB02',Name:'Suresh',Age:27,DateofBirth:'02-JUN-1996', PlayingRole:'Batsman',email:'sureshd@gmail.comm'}) return p3

create(p4:Player{PlayerID:'75',TeamID:'CCB02',Name:'Rohit',Age:33,DateofBirth:'02-JUN-1991', PlayingRole:'Batsman',email:'srohit@gmail.comm'}) return p4

Creating Relationship among CricketBoard and Teams:

match(cb:CricketBoard{BoardID:'BID01'}),(t1:Team{teamID:'CCB01'}) create(cb)-[r:has]->(t1) return cb,r,t1

match(cb:CricketBoard{BoardID:'BID01'}),(t2:Team{teamID:'CCB02'}) create(cb)-[r:has]->(t2) return cb,r,t2

Creating Relationship among Players and Teams:

match(p1:Player{PlayerID:'1'}),(t1:Team{teamID:'CCB01'}) create(p1)-[r1:playfor]->(t1) return p1,r1,t1

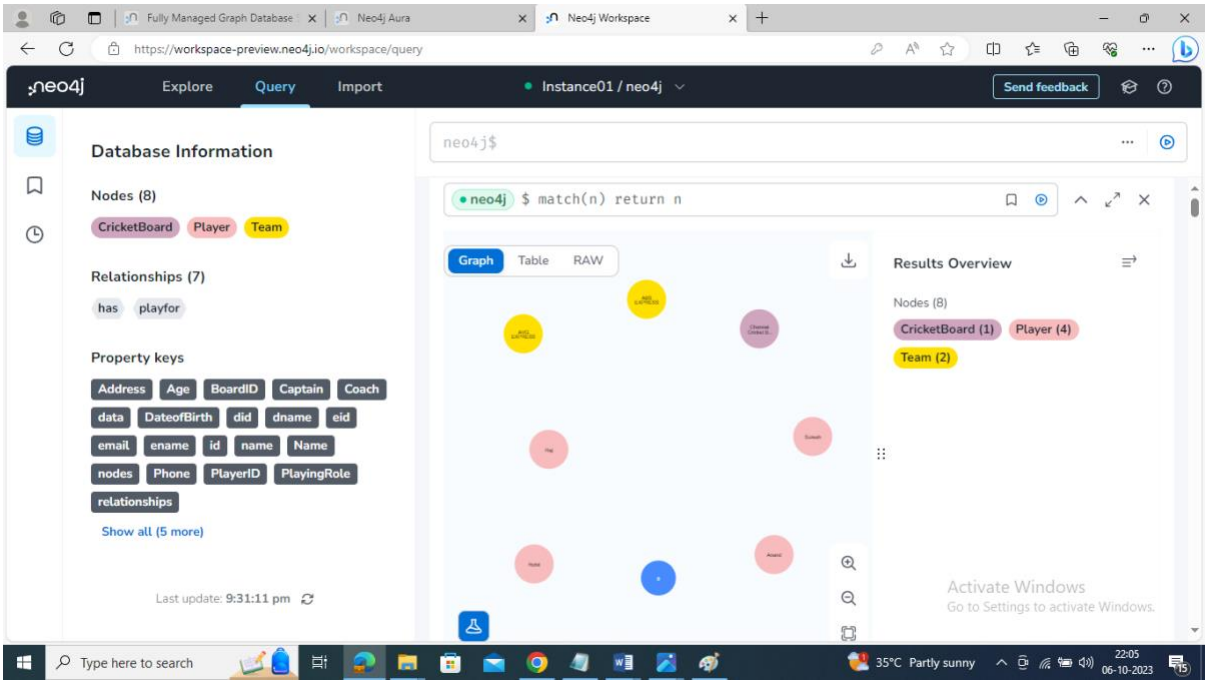
match(p2:Player{PlayerID:'33'}),(t1:Team{teamID:'CCB01'}) create(p2)-[r2:playfor]->(t1) return p2,r2,t1

match(p3:Player{PlayerID:'65'}),(t2:Team{teamID:'CCB02'}) create(p3)-[r3:playfor]->(t2) return p3,r3,t2

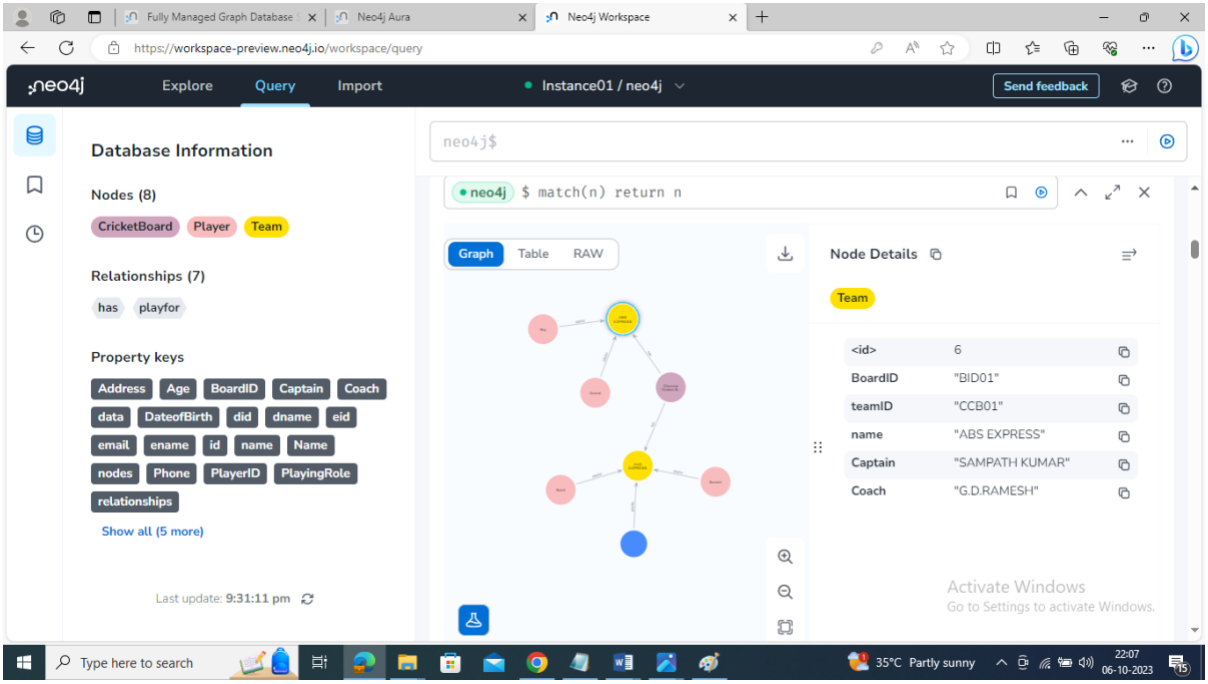
match(p4:Player{PlayerID:'75'}),(t2:Team{teamID:'CCB02'}) create(p3)-[r4:playfor]->(t2) return p4,r4,t2

Display All nodes: match(n) return n

Output:

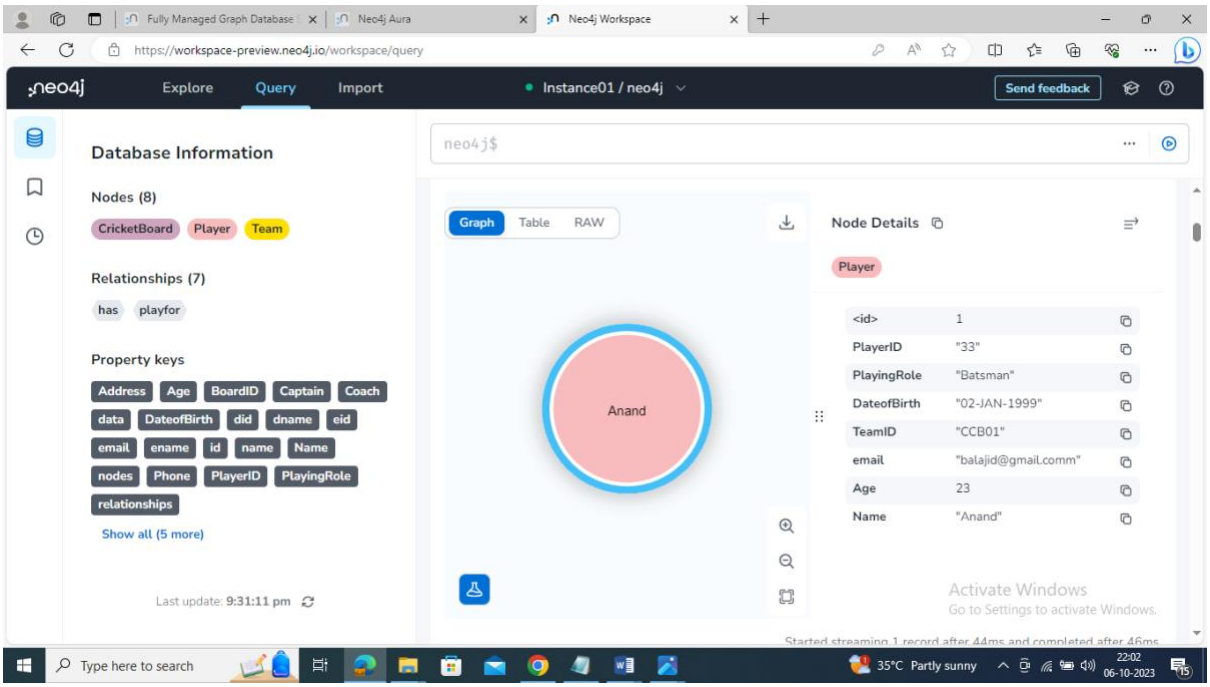


OUTPUT:



Retrieve particular player details:

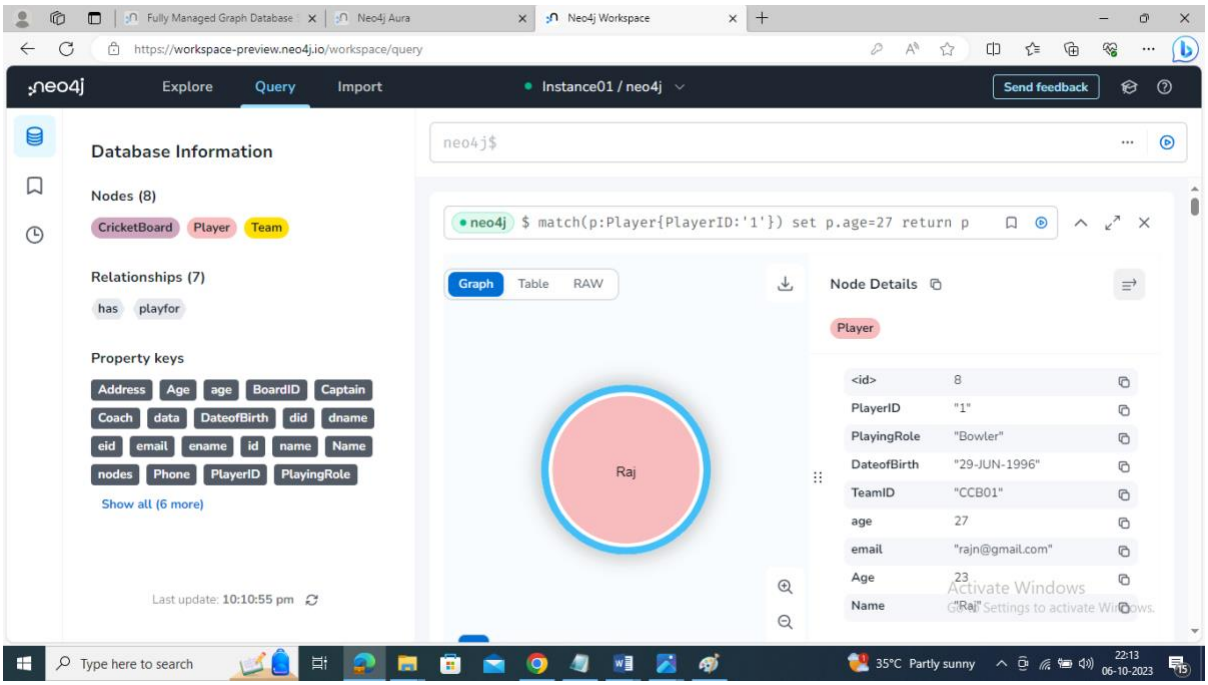
match(p:Player{PlayerID:'33'}) return p



Update particular player details:

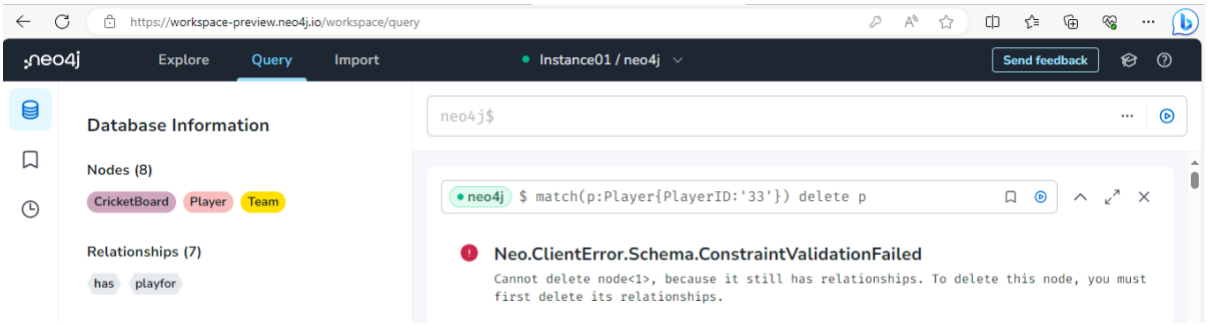
match(p:Player{PlayerID:'1'}) set p.age=27 return p

Output:



Delete particular player from the team:

match(p:Player{PlayerID:'33'}) delete p



Result:

Thus the CRUD operations like creating, inserting, querying, finding, deleting operations on graph spaces were executed successfully.