Sports League Database
Objects User Guide

Group number: 13
Seneca College
DBS311 NDD
Professor's Name: Clint MacDonald
November 30, 2023

Table of Contents

1. spPlayersInsert (Q1.a)

- **Required Input Parameters**

  - new_playerID
    - Type: players.playerID%
    - Size: NUMBER (38,0)
    - Meaning: The unique identifier for the new player

  - reg_number
    - Type: players.regNumber%TYPE
    - Size: VARCHAR2 (15)
    - Meaning: The registration number of the new player

  - last_name
    - Type: players.lastName%TYPE
    - Size: VARCHAR2 (25)
    - Meaning: The last name of the new player

  - first_name
    - Type: players.firstName%TYPE
    - Size: VARCHAR2(25)
    - Meaning: The first name of the new player

  - is_active
    - Type: players.isActive%TYPE
    - Size: NUMBER (38,0)
    - Meaning: Indicates whether the player is active or not

  - status OUT
    - Type: NUMBER
    - Meaning: indicates the result of the insertion.

- **Expected Outputs**
  - status = 1 and prints “Insertion successful”

- **Potential Error Codes**
  - -4: Player ID already exists
  - -3: All the other errors

- **Purpose**
  - Insert a new player record into the “players” table

- **Non-Saved Procedural Code**

```
DECLARE
    status NUMBER;
BEGIN
```

```
    spPlayersInsert(9999, 9999, 'Jack', 'Smith', 1,
status);
    IF status < 0 THEN
        DBMS_OUTPUT.PUT_LINE('Insertion failed. Error
code: ' || status);
    ELSE
        DBMS_OUTPUT.PUT_LINE('Insertion successful.');
    END IF;
END;
```

2. spTeamsInsert (Q1.a)

- **Required Input Parameters**

  - new_teamID
    - Type: teams.teamID%TYPE
    - Size: NUMBER (38, 0)
    - Meaning: The unique identifier for the new team

  - team_name
    - Type: teams.teamName%TYPE
    - Size: VARCHAR2 (10)
    - Meaning: The team name of the new team

  - is_active
    - Type: teams.isActive%TYPE
    - Size: NUMBER (38, 0)
    - Meaning: Indicates whether the team is active or not

  - jersey_colour
    - Type: teams.jerseyColour%TYPE
    - Size: VARCHAR2 (10)
    - Meaning: The jersey colour of the new team

  - status OUT
    - Type: NUMBER
    - Meaning: Indicates the result of the insertion

- **Expected Outputs**
  - status = 1 and prints “Insertion successful”

- **Potential Error Codes**
  - -4: Player ID already exists
  - -3: All the other errors

- **Purpose**
  - Insert a new team record into the “teams” table

- **Non-Saved Procedural Code**

```
DECLARE
    status NUMBER;
BEGIN
    spTeamsInsert(999, 'Asdf', 1, 'Blue', status);
    IF status < 0 THEN
        DBMS_OUTPUT.PUT_LINE('Insertion failed. Error
code: ' || status);
    ELSE
        DBMS_OUTPUT.PUT_LINE('Insertion successful.');
    END IF;
END;
```

3. spRostersInsert (Q1.a)

- **Required Input Parameters**

  - player_id
    - Type: rosters.playerID%TYPE
    - Size: NUMBER (38,0)
    - Meaning: The player ID of the new roster

  - team_id
    - Type: rosters.teamID%TYPE
    - Size: NUMBER (38,0)
    - Meaning: The team ID of the new roster

  - is_active
    - Type: rosters.isActive%TYPE
    - Size: NUMBER (38, 0)
    - Meaning: Indicates whether the roster is active or not

  - jersey_number
    - Type: rosters.jerseyNumber%TYPE
    - Size: NUMBER (38,0)
    - Meaning: The jersey number of the new roster

  - new_rosterID OUT
    - Type: rosters.rosterID%TYPE
    - Size: NUMBER(38,0)
    - Meaning: returning the unique identifier for the new roster ID

- **Expected Outputs**
  - status = 1 and prints “Insertion successful”

- **Potential Error Codes**
  - -4: Player ID already exists

- -3: All the other errors

- **Purpose**
  - Insert a new roster record into the “rosters” table

- **Non-Saved Procedural Code**

```
DECLARE
    new_rosterID rosters.rosterID%TYPE;
BEGIN
    spRostersInsert(9999, 999, 1, 18, new_rosterID);
    IF new_rosterID < 0 THEN
        DBMS_OUTPUT.PUT_LINE('Insertion failed. Error
code: ' || new_rosterID);
    ELSE
        DBMS_OUTPUT.PUT_LINE('Insertion successful. New
roster ID: ' || new_rosterID);
    END IF;
END;
```

4. spPlayersUpdate (Q1.b)

- **Required Input Parameters**

- player_id
  - Type: players.playerID%TYPE
  - Size: NUMBER (38,0)
  - Meaning: The unique identifier of the player whose record is to be updated

- reg_number
  - Type: players.regNumber%TYPE
  - Size: VARCHAR2 (15)
  - Meaning: The new registration number to be assigned to the player

- last_name
  - Type: players.lastName%TYPE
  - Size: VARCHAR2 (25)
  - Meaning: The new last name to be updated for the player

- first_name
  - Type: players.firstName%TYPE
  - Size: VARCHAR2 (25)
  - Meaning: The new first name to be updated for the player

- is_active
  - Type: players.isActive%TYPE
  - Size: NUMBER (38,0)

- Meaning: The new active status to be set for the player

- status OUT
  - Type: INT
  - Meaning: Indicates the result of the update

- **Expected Outputs**
  - status = 1 and prints “Update successful”

- **Potential Error Codes**
  - -1: No record found for the given player ID
  - -2: More than 1 row updated.
  - -3: All the other errors

- **Purpose**
  - Update an existing player record with new information in the “players” table

- **Non-Saved Procedural Code**

```
DECLARE
    status INT;
BEGIN
    spPlayersUpdate(9999, 999, 'John', 'Webb', 1,
status);
    IF status = 1 THEN
        DBMS_OUTPUT.PUT_LINE('Update successful');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Update failed. Error code:
' || status);
    END IF;
END;
```

5. spTeamsUpdate (Q1.b)

- **Required Input Parameters**

- Team_id
  - Type: teams.teamID%TYPE
  - Size: NUMBER (38, 0)
  - Meaning: The unique identifier of the team whose record is to be updated

- team_name
  - Type: teams.teamName%TYPE
  - Size: VARCHAR2 (10)
  - Meaning: The new team name to be assigned to the team

- is_active
  - Type: teams.isActive%TYPE
  - Size: NUMBER (38, 0)
  - Meaning: The new active status to be set for the team

- jersey_colour
  - Type: teams.jerseyColour%TYPE
  - Size: VARCHAR2 (10)
  - Meaning: The new jersey colour to be assigned to the team

- status OUT
  - Type: NUMBER
  - Meaning: Indicates the result of the update

- **Expected Outputs**
  - status = 1 and prints “Update successful”

- **Potential Error Codes**
  - -1: No record found for the given team ID
  - -2: More than 1 row updated.
  - -3: All the other errors

- **Purpose**
  - Update an existing team record with new information in the “teams” table

- **Non-Saved Procedural Code**

```
DECLARE
    status INT;
BEGIN
    spTeamsUpdate(999, 'Qwe', 1, 'Yellow', status);
    IF status = 1 THEN
        DBMS_OUTPUT.PUT_LINE('Update successful');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Update failed. Error code:
' || status);
    END IF;
END;
```

6. spRostersUpdate (Q1.b)

- **Required Input Parameters**

- roster_id
  - Type: rosters.rosterID%TYPE
  - Size: NUMBER (38,0)

- Meaning: The unique identifier of the roster whose record is to be updated

- player_id
  - Type: rosters.playerID%TYPE
  - Size: NUMBER (38,0)
  - Meaning: The new player ID to be assigned to the roster

- team_id
  - Type: IN rosters.teamID%TYPE
  - Size: NUMBER (38,0)
  - Meaning: The new team ID to be assigned to the roster

- is_active
  - Type: IN rosters.isActive%TYPE
  - Size: NUMBER (38, 0)
  - Meaning: The new active status to be set for the roster
  -
- jersey_number
  - Type: IN rosters.jerseyNumber%TYPE
  - Size: NUMBER (38,0)
  - Meaning: The new jersey number to be assigned to the roster

- status OUT
  - Type: NUMBER
  - Meaning: indicates the result of the insertion.

- **Expected Outputs**
  - status = 1 and prints “Update Successful”

- **Potential Error Codes**
  - -1: No record found for the given roster ID
  - -2: More than 1 row updated
  - -3: All the other errors

- **Purpose**
  - Update an existing roster record with new information in the “rosters” table

- **Non-Saved Procedural Code**

```
DECLARE
    status INT;
BEGIN
    spRostersUpdate(301, 9999, 999, 1, 12, status);
    IF status = 1 THEN
        DBMS_OUTPUT.PUT_LINE('Update successful');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Update failed. Error code:
' || status);
```

```
    END IF;
END;
```

7. spPlayersDelete (Q1.c)

- **Required Input Parameters**

  - player_id
    - Type: players.playerID%TYPE
    - Size: NUMBER (38,0)
    - Meaning: The unique identifier for the player whose record is to be deleted from the table
  - status OUT
    - Type: INT
    - Meaning: indicates the result of the deletion.

- **Expected Outputs**
  - status = 1 and prints “Delete Successful”

- **Potential Error Codes**
  - -1: No record found for the given player ID
  - -2: More than 1 row deleted.
  - -3: All the other errors

- **Purpose**
  - Delete a player record from the “players” table

- **Non-Saved Procedural Code**

```
DECLARE
    status INT;
BEGIN
    spPlayersDelete(9999, status);
    IF status = 1 THEN
        DBMS_OUTPUT.PUT_LINE('Delete successful');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Delete failed. Error code:
' || status);
    END IF;
END;
```

8. spTeamsDelete (Q1.c)

- **Required Input Parameters**

  - team_id
    - Type: teams.teamID%TYPE

- Size: NUMBER (38,0)
- Meaning: The unique identifier for the team whose record is to be deleted from the table

- status OUT
  - Type: INT
  - Meaning: Indicates the result of the deletion

- **Expected Outputs**
  - status = 1 and prints “Delete Successful”

- **Potential Error Codes**
  - -1: No record found for the given team ID
  - -2: More than 1 row deleted.
  - -3: All the other errors

- **Purpose**
  - Delete a team record from the “teams” table

- **Non-Saved Procedural Code**

```
DECLARE
    status INT;
BEGIN
    spTeamsDelete(999, status);
    IF status = 1 THEN
        DBMS_OUTPUT.PUT_LINE('Delete successful');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Delete failed. Error code:
' || status);
    END IF;
END;
```

9. spRostersDelete (Q1.c)

- **Required Input Parameters**

  - roster_id
    - Type: rosters.roster%TYPE
    - Size: NUMBER (38,0)
    - Meaning: The unique identifier for the roster whose record is to be deleted from the table

  - status OUT
    - Type: INT
    - Meaning: Indicates the result of the deletion

- **Expected Outputs**

- status = 1 and prints “Delete Successful”

- **Potential Error Codes**
  - -1: No record found for the given roster ID
  - -2: More than 1 row deleted.
  - -3: All the other errors

- **Purpose**
  - Delete a roster record from the “rosters” table

- **Non-Saved Procedural Code**

```
DECLARE
    status INT;
BEGIN
    spRostersDelete(301, status);
    IF status = 1 THEN
        DBMS_OUTPUT.PUT_LINE('Delete successful');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Delete failed. Error code:
' || status);
    END IF;
END;
```

10. spPlayersSelect (Q1.d)

- **Required Input Parameters**

  - player_id
    - Type: players.playerID%TYPE
    - Size: NUMBER (38,0)
    - Meaning: The unique identifier of the player whose details user wish to retrieve

  - reg_number OUT
    - Type: players.regNumber%TYPE
    - Size: VARCHAR2 (15)
    - Meaning: The registration number of the player

  - last_name OUT
    - Type: players.lastName%TYPE
    - Size: VARCHAR2 (25)
    - Meaning: The last name of the player

  - first_name OUT
    - Type: players.firstName%TYPE
    - Size: VARCHAR2 (25)
    - Meaning: The first name of the player

- is_active OUT
  - Type: players.isActive%TYPE
  - Size: NUMBER (38,0)
  - Meaning: Indicates whether the player is active or not

- status OUT
  - Type: INT
  - Meaning: indicates the result of the selection

- **Expected Outputs**
  - status = 1 and prints “Select successful” as well as the player ID, reg number, last name, first name and is active of the player

- **Potential Error Codes**
  - -5: No record found for the given player ID
  - -6: More than 1 row selected
  - -3: All the other errors

- **Purpose**
  - Select all fields of the given player ID from the “players” table

- **Non-Saved Procedural Code**

```
DECLARE
    p_playerID players.playerID%TYPE := 1302;
    p_regNum players.regNumber%TYPE;
    p_lastN players.lastName%TYPE;
    p_firstN players.firstName%TYPE;
    p_is_active players.isActive%TYPE;
    status INT;
BEGIN
    spPlayersSelect(p_playerID, p_regNum, p_lastN,
p_firstN, p_is_active, status);
    IF status = 1 THEN
        DBMS_OUTPUT.PUT_LINE('Select successful.');
        DBMS_OUTPUT.PUT_LINE('PLAYERID: '|| p_playerID);
        DBMS_OUTPUT.PUT_LINE('REGNUMBER: '|| p_regNum);
        DBMS_OUTPUT.PUT_LINE('LASTNAME: '|| p_lastN);
        DBMS_OUTPUT.PUT_LINE('FIRSTNAME: '|| p_firstN);
        DBMS_OUTPUT.PUT_LINE('ISACTIVE: '|| p_is_active);
    ELSE
        DBMS_OUTPUT.PUT_LINE('Select failed. Error code:
' || status);
    END IF;
END;
```

11. spTeamsSelect (Q1.d)

- **Required Input Parameters**

  - team_id
    - Type: teams.teamID%TYPE
    - Size: NUMBER (38,0)
    - Meaning: The unique identifier of the team whose details user wish to retrieve

  - team_name OUT
    - Type: teams.teamName%TYPE
    - Size: VARCHAR2 (10)
    - Meaning: The name of the team

  - is_active OUT
    - Type: teams.isActive%TYPE
    - Size: NUMBER (38,0)
    - Meaning: Indicates whether the team is active or not

  - jersey_colour OUT
    - Type: teams.jerseyColour%TYPE
    - Size: VARCHAR2(10)
    - Meaning: The jersey colour of the team

  - status OUT
    - Type: INT
    - Meaning: indicates the result of the selection

- **Expected Outputs**
  - status = 1 and prints “Select successful” as well as the team ID, team name, is active and jersey colour of the team

- **Potential Error Codes**
  - -5: No record found for the given team ID
  - -6: More than 1 row selected
  - -3: All the other errors

- **Purpose**
  - Select all fields of the given team ID from the “teams” table

- **Non-Saved Procedural Code**

```
DECLARE
    t_teamID teams.teamID%TYPE := 210;
    t_regNum teams.teamName%TYPE;
    t_is_active teams.isActive%TYPE;
    t_jersey_colour teams.jerseyColour%TYPE;
    status INT;
BEGIN
    spTeamsSelect(t_teamID, t_regNum, t_is_active,
t_jersey_colour, status);
```

```
    IF status = 1 THEN
        DBMS_OUTPUT.PUT_LINE('Select successful.');
        DBMS_OUTPUT.PUT_LINE('TEAMID: '|| t_teamID);
        DBMS_OUTPUT.PUT_LINE('REGNUMBER: '|| t_regNum);
        DBMS_OUTPUT.PUT_LINE('ISACTIVE: '|| t_is_active);
        DBMS_OUTPUT.PUT_LINE('JERSEYCOLOUR: '||
t_jersey_colour);
    ELSE
        DBMS_OUTPUT.PUT_LINE('Select failed. Error code:
' || status);
    END IF;
END;
```

12. spRostersSelect (Q1.d)

- **Required Input Parameters**

  - roster_id
    - Type: rosters.rosterID%TYPE
    - Size: NUMBER (38,0)
    - Meaning: The unique identifier of the roster whose details user wish to retrieve

  - player_id OUT
    - Type: rosters.playerID%TYPE
    - Size: NUMBER (38,0)
    - Meaning: The player ID of the roster

  - team_id OUT
    - Type: rosters.teamID%TYPE
    - Size: NUMBER (38,0)
    - Meaning: The team ID of the roster

  - is_active OUT
    - Type: rosters.isActive%TYPE
    - Size: NUMBER(38, 0)
    - Meaning: Indicates whether the roster is active or not

  - jersey_number OUT
    - Type: rosters.jerseyNumber%TYPE
    - Size: NUMBER(38,0)
    - Meaning: The jersey number of the roster

  - status OUT
    - Type: INT
    - Meaning: indicates the result of the selection

- **Expected Outputs**

- status = 1 and prints “Select successful” as well as the roster ID, player ID, team ID, is active and jersey number of the roster

- **Potential Error Codes**
  - -5: No record found for the given roster ID
  - -6: More than 1 row selected
  - -3: All the other errors

- **Purpose**
  - Select all fields of the given roster ID from the “rosters” table

- **Non-Saved Procedural Code**

```
DECLARE
    r_rosterID rosters.rosterID%TYPE := 1;
    r_player_id rosters.playerID%TYPE;
    r_team_id rosters.teamID%TYPE;
    r_jersey_colour rosters.isActive%TYPE;
    r_jersey_number rosters.jerseyNumber%TYPE;
    status INT;
BEGIN
    spRostersSelect(r_rosterID, r_player_id, r_team_id,
r_jersey_colour, r_jersey_number, status);
    IF status = 1 THEN
        DBMS_OUTPUT.PUT_LINE('Select successful.');
        DBMS_OUTPUT.PUT_LINE('ROSTERID: '|| r_rosterID);
        DBMS_OUTPUT.PUT_LINE('PLAYERID: '|| r_player_id);
        DBMS_OUTPUT.PUT_LINE('TEAMID: '|| r_team_id);
        DBMS_OUTPUT.PUT_LINE('JERSEYCOLOUR: '||
r_jersey_colour);
        DBMS_OUTPUT.PUT_LINE('JERSEYNUMBER: '||
r_jersey_number);
    ELSE
        DBMS_OUTPUT.PUT_LINE('Select failed. Error code:
' || status);
    END IF;
END;
```

13. spPlayersSelectAll (Q2)

- **Required Input Parameters**
  - No input parameters

- **Expected Outputs**
  - Prints all fields of every player from the “players” table in a tabular format

- **Potential Error Codes**
  - -3: Any errors occurred

- **Purpose**
  - Select and output all players information from the player table by storing players information in the cursor and loop through it to print each player returned by the cursor.

- **Non-Saved Procedural Code**

```
BEGIN
    spPlayersSelectAll();
END;
```

14. spTeamsSelectAll (Q2)

- **Required Input Parameters**
  - No input parameters

- **Expected Outputs**
  - Prints all fields of every team from the “teams” table in a tabular format

- **Potential Error Codes**
  - -3: Any errors occurred

- **Purpose**
  - Select and output all teams information from the teams table by storing teams information in the cursor and loop through it to print each team returned by the cursor.

- **Non-Saved Procedural Code**

```
BEGIN
    spTeamsSelectAll();
END;
```

15. spRostersSelectAll (Q2)

- **Required Input Parameters**
  - No input parameters

- **Expected Outputs**
  - Prints all fields of every roster from the “rosters” table in a tabular format

- **Potential Error Codes**
  - -3: Any errors occurred

- **Purpose**
  - Select and output all rosters information from the roster table by storing rosters information in the cursor and loop through it to print each roster returned by the cursor.

- **Non-Saved Procedural Code**

```
BEGIN
    spRostersSelectAll();
END;
```

16. spPlayersSelectAllOutput (Q3)

- **Required Input Parameters**
  - p_result OUT
    - Type: SYS_REFCURSOR:
    - Meaning: output cursor which will contain the result set after the procedure is executed

- **Expected Outputs**
  - No output from this saved procedure. All player info is printed when this procedure is executed in subsequent non-saved procedure.

- **Potential Error Codes**
  - N/A

- **Purpose**
  - Opens the output cursor and associates it with a select statement that fetches all columns from the “players” table.

- **Non-Saved Procedural Code**

```
DECLARE
    p1_result SYS_REFCURSOR;
    player_id players.playerID%TYPE;
    reg_number players.regNumber%TYPE;
    last_name players.lastName%TYPE;
    first_name players.firstName%TYPE;
    is_active players.isActive%TYPE;
BEGIN
    spPlayersSelectAllOutput(p1_result);
    DBMS_OUTPUT.PUT_LINE(
        RPAD('PLAYERID', 12, ' ') ||
        RPAD('REGNUMBER', 12, ' ') ||
        RPAD('LASTNAME', 20, ' ') ||
        RPAD('FIRSTNAME', 20, ' ') ||
        'ACTIVE');
    LOOP
```

```
        FETCH p1_result INTO player_id, reg_number,
last_name, first_name, is_active;
        EXIT WHEN p1_result%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(
            RPAD(player_id, 12, ' ') ||
            RPAD(reg_number, 12, ' ') ||
            RPAD(last_name, 20, ' ') ||
            RPAD(first_name, 20, ' ') ||
            is_active
        );
    END LOOP;
    CLOSE p1_result;
END;
```

17. spTeamsSelectAllOutput (Q3)

- **Required Input Parameters**
  - p_result OUT
    - Type: SYS_REFCURSOR
    - Meaning: output cursor which will contain the result set after the procedure is executed

- **Expected Outputs**
  - No output from this saved procedure. All teams info is printed when this procedure is executed in subsequent non-saved procedure.

- **Potential Error Codes**
  - N/A

- **Purpose**
  - Opens the output cursor and associates it with a select statement that fetches all columns from the “teams” table.

- **Non-Saved Procedural Code**

```
DECLARE
    p1_result SYS_REFCURSOR;
    team_id teams.teamID%TYPE;
    team_name teams.teamName%TYPE;
    is_active teams.isActive%TYPE;
    jersey_colour teams.jerseyColour%TYPE;
BEGIN
    spTeamsSelectAllOutput(p1_result);
    DBMS_OUTPUT.PUT_LINE(
        RPAD('TEAMID', 9, ' ') ||
        RPAD('TEAMNAME', 12, ' ') ||
        RPAD('ACTIVE', 9, ' ') ||
        'JERSEYNUMBER');
    LOOP
```

```
        FETCH p1_result INTO team_id, team_name,
is_active, jersey_colour;
        EXIT WHEN p1_result%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(
            RPAD(team_id, 9, ' ') ||
            RPAD(team_name, 12, ' ') ||
            RPAD(is_active, 9, ' ') ||
            jersey_colour);
    END LOOP;
    CLOSE p1_result;
END;
```

18. spRostersSelectAllOutput (Q3)

- **Required Input Parameters**
  - p_result OUT
    - Type: SYS_REFCURSOR
    - Meaning: output cursor which will contain the result set after the procedure is executed

- **Expected Outputs**
  - No output from this saved procedure. All rosters info is printed when this procedure is executed in subsequent non-saved procedure.

- **Potential Error Codes**
  - N/A

- **Purpose**
  - Open the output cursor and associate it with a select statement that fetches all columns from the “rosters” table.

- **Non-Saved Procedural Code**

```
DECLARE
    p1_result SYS_REFCURSOR;
    roster_id rosters.rosterID%TYPE;
    player_id rosters.playerID%TYPE;
    team_id rosters.teamID%TYPE;
    is_active rosters.isActive%TYPE;
    jersey_number rosters.jerseyNumber%TYPE;
BEGIN
    spRostersSelectAllOutput(p1_result);
    DBMS_OUTPUT.PUT_LINE(
        RPAD('ROSTERID', 12, ' ') ||
        RPAD('PLAYERID', 12, ' ') ||
        RPAD('TEAMID', 9, ' ') ||
        RPAD('ACTIVE', 9, ' ') ||
        'JERSEY NUMBER');
    LOOP
```

```
        FETCH p1_result INTO roster_id, player_id,
team_id, is_active, jersey_number;
        EXIT WHEN p1_result%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(
            RPAD(roster_id, 12, ' ') ||
            RPAD(player_id, 12, ' ') ||
            RPAD(team_id, 9, ' ') ||
            RPAD(is_active, 9, ' ') ||
            jersey_number);
    END LOOP;
    CLOSE p1_result;
END;
```

19. vwPlayerRosters (Q4)

- **Required Input Parameters**
  - No input parameters

- **Expected Outputs**
  - When select * from vwPlayerRosters is executed, the output will be a table with information about each individual players and their associated teams and rosters.

- **Potential Error Codes**
  - N/A

- **Purpose**
  - Create a view that includes all fields from players, rosters and teams table for players who are currently rostered with a team, making it easier for users to understand the current player-roster information.

- **Execution**

```
SELECT * FROM vwPlayerRosters;
```

20. spTeamRosterByID (Q5)

- **Required Input Parameters**
  - tID
    - Type: vwPlayerRosters.teamID%TYPE
    - Size: NUMBER (38,0)
    - Meaning: the unique identifier for a team

- **Expected Outputs**
  - Print information about all players who are currently rostered with a team given a specific team ID in a tabular form.

- **Potential Error Codes**
  - -7: No data in cursor
  - -8: ROWTYPE_MISMATCH
  - -3: All the other errors

- **Purpose**
  - Use cursor to select and store all records from vwPlayerRosters whose teamID equals the input parameter and use loop to fetch each player roster record and print player ID, name, jersey number and team in a tabular format.

- **Non-Saved Procedural Code**

```
BEGIN
    spTeamRosterByID(212);
END;
```

21. spTeamRosterByName (Q6)

- **Required Input Parameters**
  - tname
    - Type: vwPlayerRosters.teamName%TYPE
    - Size: VARCHAR2(10)
    - Meaning: any part of a team name

- **Expected Outputs**
  - Print information about all players who are currently rostered with a team given a string of any part of the team name.

- **Potential Error Codes**
  - -7: No data in cursor
  - -8: ROWTYPE_MISMATCH
  - -3: All the other errors

- **Purpose**
  - Use cursor to select and store all records from vwPlayerRosters whose team name contains the input parameter and use loop to fetch each player roster record and print player ID, name, jersey number and team in a tabular format.

- **Non-Saved Procedural Code**

```
BEGIN
    spTeamRosterByName('aur');
END;
```

22. vwTeamsNumPlayers (Q7)

- **Required Input Parameters**
  - N/A

- **Expected Outputs**
  - When select * from vwTeamsNumPlayers is executed, the output will be a table with information about number of players on each team

- **Potential Error Codes**
  - N/A

- **Purpose**
  - Create a view to provide a summary of the number of active players on each team

23. fncNumPlayersByTeamID (Q8)

- **Required Input Parameters**
  - tid
    - Type: INT
    - Meaning: the unique identifier of a team

- **Expected Outputs**
  - Return the number of players currently rostered with a team given a specific team ID.

- **Potential Error Codes**
  - -5: NO_DATA_FOUND
  - -6: TOO_MANY_ROW
  - -3: All the other errors

- **Purpose**
  - Retrieve the number of active players on a specific team given the provided teamID, based on the vwTeamsNumPlayers

- **Non-Saved Procedural Code**

```
BEGIN
    DBMS_OUTPUT.PUT_LINE(fncNumPlayersByTeamID(212));
END;
```

24. vwSchedule (Q9)

- **Required Input Parameters**
  - N/A

- **Expected Outputs**
  - When select * from vwSchedule is executed, the output will be a table with information about all games, including game ID, game date and time, home team name and score, visit team name and score, as well as location name.

- **Potential Error Codes**
  - N/A

- **Purpose**
  - Create a view to quickly identify upcoming games, review past results, and access game details such as scores, teams, and locations.

25. spSchedUpcomingGames (Q10)

- **Required Input Parameters**
  - numDay
    - Type: INT
    - Meaning: the games to be played in the next number of days

- **Expected Outputs**
  - Information about all games to be played in the next number of days given a specific number, in a tabular form.

- **Potential Error Codes**
  - -7: No data in cursor
  - -8: ROWTYPE_MISMATCH
  - -3: All the other errors

- **Purpose**
  - Retrieve and display information, including game ID, game date and time, home team name, visit team name and location name about upcoming games scheduled within a specified number of days based on the vwSchedule

- **Non-Saved Procedural Code**

```
BEGIN
    spSchedUpcomingGames(5);
END;
```

26. spSchedPastGames (Q11)

- **Required Input Parameters**
  - numDay
    - Type: INT

- Meaning: the games to be played in the next number of days

- **Expected Outputs**
  - Information about all games have played in the past number of days given a specific number, in a tabular form.

- **Potential Error Codes**
  - -7: No data in cursor
  - -8: ROWTYPE_MISMATCH
  - -3: All the other errors

- **Purpose**
  - Retrieve and display information, including game ID, game date and time, home team name and score, visit team name and score as well as location name about past games played within a specified number of days based on the vwSchedule

- **Non-Saved Procedural Code**

```
BEGIN
    spSchedPastGames(5);
END;
```

27. spRunStandings (Q12)

- **Required Input Parameters**
  - No input parameters

- **Expected Outputs**
  - When spRunStandings is executed, it first deletes all records from the temporary table and then inserts new records into it from the select statement

- **Potential Error Codes**
  - N/A

- **Purpose**
  - Calculate and update the standings of each team based on played games information from the “games” table

- **Non-Saved Procedural Code**

```
BEGIN
    spRunStandings();
END;
```

28. trgRunStandings (Q13)

- **Required Input Parameters**
  - No input parameters

- **Expected Outputs**
  - When trgRunStandings is executed, it automatically executes spRunStandings, which will replace the tempStandings with the output of the select statement

- **Potential Error Codes**
  - N/A

- **Purpose**
  - Automatically update the standings of each team whenever there are changes made to the games table

- **Execution**

```
UPDATE games SET homescore = 2 WHERE gameid = 1;

SELECT * FROM tempStandings;
```

29. vwTeamScorer (Q14)

- **Required Input Parameters**
  - No input parameters

- **Expected Outputs**
  - When select * from vwTeamScorer is executed, the output will be a table with information about all players who ever scored goals in each team.

- **Potential Error Codes**
  - N/A

- **Purpose**
  - Provide a list of players with their total goals scored based on their affiliated team.

- **Execution**

```
SELECT * FROM vwTeamScorer;
```

30. spTeamScorer (Q14)

- **Required Input Parameters**

- tid
    - Type: INT
    - Meaning: the unique identifier of a team

- **Expected Outputs**
    - Print first name, last name and total goals about all players who scored goals in a team given a specific team ID, in a tabular format

- **Potential Error Codes**
    - -7: No data in cursor
    - -8: ROWTYPE_MISMATCH
    - -3: All the other errors

- **Purpose**
    - List the players who scored goals from a specific team

- **Non-Saved Procedural Code**

```
BEGIN
    spTeamScorer(212);
END;
```