CS 260 - Database Systems

Lab 9

This lab will focus on creating stored programs written in PL/SQL. Our stored programs will use the MLB database from assignment 1. If you've dropped or modified those tables, you'll need to rerun the installation scripts present with assignment 1. You will create a single script file: "mlb_stats_pkg.sql". You can use the provided "assignment_4_tests.sql" document to test your scripts by comparing its output to the provided output generated by my solution.

mlb_stats_pkg.sql

This script is responsible for creating a package (and its body) named "mlb_stats_pkg" that has the procedures and functions specified below. Be sure to SHOW ERRORS after the package declaration and body.

procedure calc_team_batting_stats(team_name)

This procedure takes in a team name (as present in mlb_team.name) and prints the following information for each player on the team with batting stats (sorted by last name) exactly as present in the provided "assignment 4 tests" output:

- First and last name
- At bats per home run (at_bats / home_runs)
    - Rounded to 2 decimal places using the ROUND function
- Stolen base percentage (stolen_bases / (stolen_bases + caught_stealing))
    - Displayed as a percentage as seen in the provided output

If a ZERO_DIVIDE exception occurs, the same information should be printed, but the player's at bats per home run or stolen base percentage value should be "undefined".

After each player's batting information has been printed, the team's average at bats per home run and stolen base percentage should be printed exactly as present in the provided output. The describe_team_ab_per_hr(ab_per_hr) function should be called to print a description of that statistic. You should only use a single cursor in this procedure.

procedure calc_all_team_batting_stats

This procedure will call calc_all_team_batting_stats(team_name) once for each team (sorted by team name).

function describe_team_ab_per_hr(ab_per_hr)

This function should return a varchar2 according to the following conditions:

- ab_per_hr < 28: return "great"
- 28 <= ab_per_hr < 33: return "good"

- 33 <= ab_per_hr < 38: return “okay”
- ab_per hr >= 38: return “poor”

Tips

- Suggested approach
  - Write functions outside of a package and test
  - Write team procedures outside of a package and test
  - Write all-team procedures outside of a package and test
  - Add all procedures and functions to a package and test
- In your functions, the data type capacity needs to be specified for any declared variables, but not for the specified return type