Lab 3: Advanced SQL

Part 1: (50 marks). Due on Wednesday (Jan 19th 11.59 pm) on Moodle.

1. Write DDL (create table statements) for the schema shown below. Your DDL file should contain ***both the table definitions as well as the constraints specified at the end of the schema.*** **(15 marks)**

2. With the given set of CSV files that contain data for this schema and the DDL file written in the previous part, write a Python program to install BOTH the DDL you wrote AND upload data into the given DB in Postgres. ***Look at the documentation available at: https://www.psycopg.org/docs/***. Your program will be timed to check for efficiency (while there are no thresholds for performance we urge you to time this program yourselves and play around with how you do the data insertion - there are many web articles on how to improve the performance of Multiple inserts).

   Save the file as **insert.py** and the file would be run using the following command:

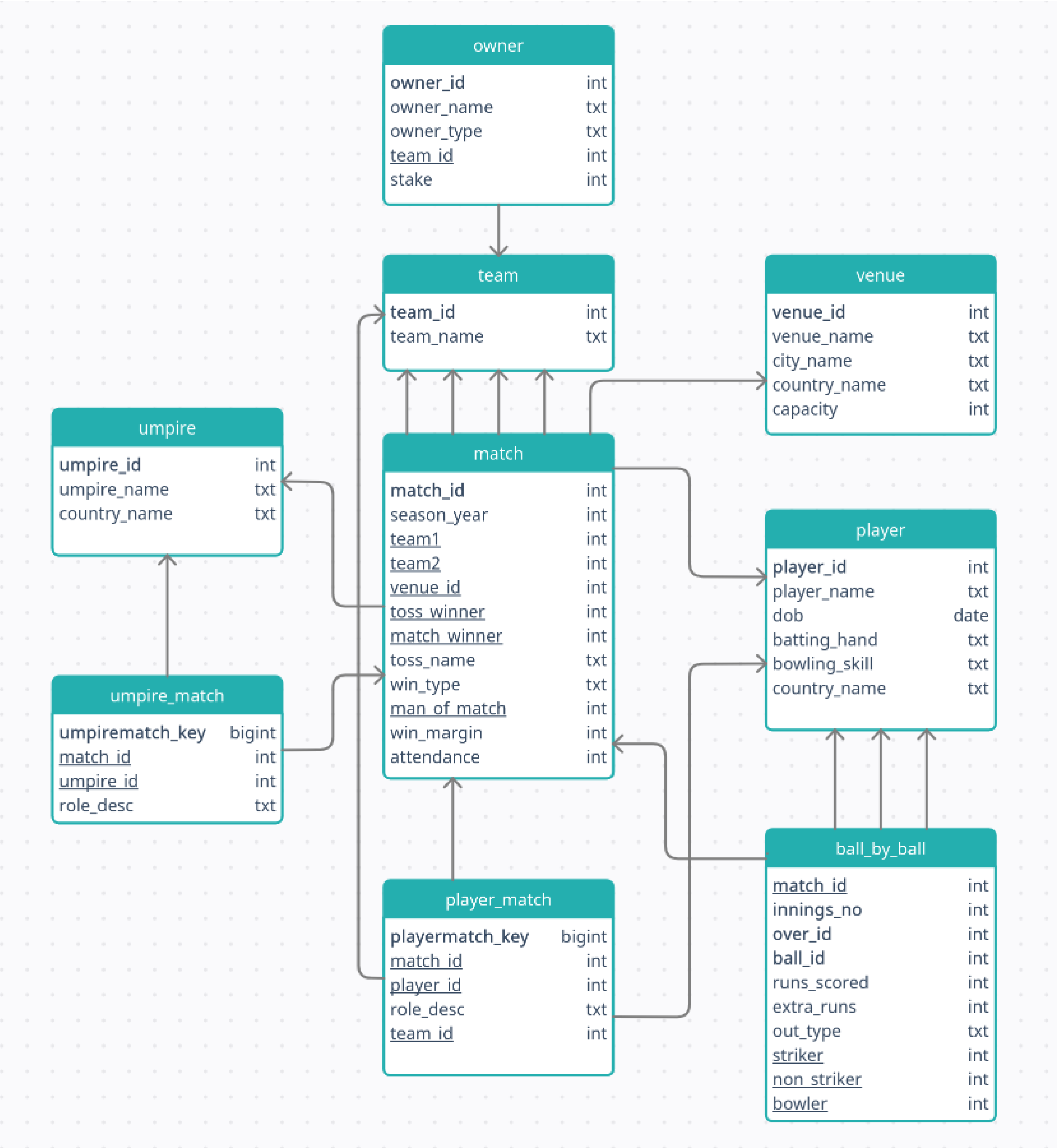
   ```
   python3 insert.py --name <db_name> --user <username> --pswd <password> --host
   <host_address> --port <port> --ddl <path_to_ddl_file> --data
   <path_to_folder_containing_csv_files>
   ```

   The folder path given would contain the files attached to this Moodle assignment.
   (35 marks)

   Notes:

   a. You will need the psycopg2 module for this. Please look up documentation on how to use the API in this module to help with DB connections, cursors, and executing SQL from Python programs.
   b. First, you open a connection, set up a cursor, prepare the SQL insert statement, and then set up a loop to do the inserts with the CSV file open. Hint: **`execute_values`** is faster than **`execute`** API for multiple inserts. I suggest you try both (execute in a loop as well as execute_values) and compare the timings of the two. You may submit the most efficient solution.
   c. **Submit a tarball named <rollnumber>-lab3.tar.gz with two files:**
      i. **insert.py** (we will have our own DDL that we will apply for this). Nothing in your python program should hardcode ANY of the DDL. It MUST read the DDL file given and work.
      ii. **lab3.ddl** (we will just check this manually if needed).

Schema:



owner	
owner_id	int
owner_name	txt
owner_type	txt
team_id	int
stake	int

team	
team_id	int
team_name	txt

venue	
venue_id	int
venue_name	txt
city_name	txt
country_name	txt
capacity	int

umpire	
umpire_id	int
umpire_name	txt
country_name	txt

match	
match_id	int
season_year	int
team1	int
team2	int
venue_id	int
toss_winner	int
match_winner	int
toss_name	txt
win_type	txt
man_of_match	int
win_margin	int
attendance	int

player	
player_id	int
player_name	txt
dob	date
batting_hand	txt
bowling_skill	txt
country_name	txt

umpire_match	
umpirematch_key	bigint
match_id	int
umpire_id	int
role_desc	txt

player_match	
playermatch_key	bigint
match_id	int
player_id	int
role_desc	txt
team_id	int

ball_by_ball	
match_id	int
innings_no	int
over_id	int
ball_id	int
runs_scored	int
extra_runs	int
out_type	txt
striker	int
non_striker	int
bowler	int

Constraints:

1. Primary Key (bold) and Foreign Key (underlined) constraints present in the given schema
2. **out_type** can only take the values 'caught', 'caught and bowled' , 'bowled' , 'stumped' , 'retired hurt', 'keeper catch', 'lbw', 'run out', ‘hit wicket’ or NULL
3. **role_desc** in player_match can only take the values 'Player', 'Keeper', 'CaptainKeeper', or 'Captain'
4. **toss_name** can only take the values 'field' or 'bat'
5. **win_type** can only take the values 'wickets', 'runs' or NULL
6. **runs_scored** in ball_by_ball should be between 0 and 6
7. **innings_no** in ball_by_ball should be either 1 or 2 only.
8. **stake** in owner should be between 1 and 100 (both inclusive). Additionally, the sum of stakes for a team should be 1 <= sum of stakes <= 100.
9. **attendance** in any match should be less than the capacity of the venue at which the match was hosted
10. **role_desc** in umpire_match can only take the values ‘Field’ or ‘Third’
11. There must be at most 2 umpires with the role ‘Field’ and at most 1 umpire with the role ‘Third’ for every match (in umpire_match)

Part2 - Write SQL statements for the following queries (to be submitted in XDATA on Moodle). Due by Thursday, Jan 20th, 11.00 pm on XDATA.

Note/Clarifications:

- In every question(except q.6) don't include extra_runs in the runs scored by a batsman.
- In the wickets taken by a bowler don't include the wickets with out_type as 'run out' and 'retired hurt'.
- Int divided by int gives an int, so make sure to multiply by 1.0 before division.
- Each query is worth 10 marks

1. Find the best player at the venue; the player with the maximum number of man of the match awards at the venue is considered as the best player (break tie with player_name) output in the increasing order of venue_id.

   Output <venue_id, venue_name, player_id, player_name>

2. Find players who scored the maximum runs per match on average; Limit your answer to the top 10 by using sparse rank (you may get more than 10 in case of ties).

   Output <player_id, player_name, avg_runs>

3. Find players who are the top 5 frequent wicket-takers, that is, players who took a wicket in the highest fraction of balls that they bowl. Output the player id, player name, the number of times the player has got a wicket in a ball, the number of balls bowled, and the fraction of wickets*6.

   Output in the decreasing order of frac, break the tie with player_id (lower comes former)

4. Find top 3 (exactly 3) batsmen and top 3 (exactly 3) bowlers' player_ids who got highest no of runs and highest no of wickets respectively in each season.

   Output <season_year, batsman, runs, bowler, wickets>

   Here batsman & bowler are player_ids of the players. In case of ties output the player with lesser player_id first.

   Order by season_year (earlier year comes first) and rank(batsman and bowler with more no of runs and wickets in a particular season come first).

   There will be (no_of_seasons*3) rows.

5. Find the ids of players who got the highest no of partnership runs for each match. There can be multiple rows for a single match.

   Output <match_id, player1_id, player2_id, runs1, runs2, pship_runs>

   player1's contribution i.e. runs1 >= player2's contribution i.e. runs2, in descending order of pship_runs (in case of ties compare match_id in ascending order).

   If runs1=runs2 then player1_id > player2_id.

   Note: extra_runs shouldn't be counted

6. For all the matches with win type as ‘runs’’, find the over ids in which the runs scored are greater than 12 runs.

   Output <match_id, innings_no, over_id>

   Note: Runs scored in an over also include the extra_runs.

7. List top 5 batsmen(exactly 5) by the number of sixes hit in the season 2013. Break ties alphabetically.

   Output <player_name>

8. List the names of players who have taken a 5-wicket haul in any match played. Order the output alphabetically on player_name.

   Output <player_name>

   Note: 5-wicket haul is when a player takes 5 or more wickets in a single innings

9. Find the toss win to match win conversion percentage for all the teams that have won at least one match having won the toss (across all seasons). Order the result alphabetically on team names.

   Output <team_name, conversion_percentage>

   Conversion percentage of a team can be calculated as = (number of matches won by the team, having won the toss/number of tosses won by the team) * 100

   Note: Calculate percentage up to 3 decimal places. Use ROUND (not truncate) to achieve so.

10. For each match, find most and least expensive overs (both runs_scored and extra_runs in that over must be taken into account) break tie with over_id (consider least over_id)

    Output <match_id, over_id>

    There will be (no_of_matches*2) rows.