**# mustard**

-

**=======================================================================**

Python library Imports

**=======================================================================**

-- sudo pip install flatten\_json

-- sudo pip install json

-

**=======================================================================**

--Python script that will parse these files,

--and read them into a database with that schema

**=======================================================================**

-- dataLoad.py

-- db\_config.py

-- db\_info.ini  (This contains connection details to posgreSQL database)

Run time process

1. Clone the git repo or unzip the file in the attached email

<https://github.com/olat-dev-world/mustard.git>

1. install the following python libraries (if required)

-- sudo pip install flatten\_json

-- sudo pip install json

1. create the schema and tables in PostgreSQL

see **DATA MODELLING** section below

1. from (1), change to mustard folder and run the following script

python3 dataLoad.py

1. Run the following script for the assessment reports

**See REPORT/Assessment QUERIES**

**=======================================================================**

-- DATA MODELLING

-- Mustard balls tables creation

-- SQL database schema for representing these balls.

**======================================================================**

create schema cricket;

--Drop tables if they already exists

drop table cricket.matches;

drop table cricket.teams;

drop table cricket.players;

CREATE TABLE cricket.matches (

    row\_id INT GENERATED ALWAYS AS IDENTITY,

    match\_id INT NOT null,

    is\_out VARCHAR(100) not null,

    runs int not null,

    batting\_team\_id INT NOT null ,

    bowling\_team\_id INT NOT null ,

    batter\_id int not null,

    non\_facer\_id INT NOT null ,

    bowler\_id INT NOT null ,

    created\_date timestamp default now(),

    PRIMARY KEY(match\_id,batting\_team\_id,bowling\_team\_id,batter\_id,non\_facer\_id,bowler\_id)

);

CREATE TABLE cricket.teams (

    row\_id INT GENERATED ALWAYS AS IDENTITY,

    team\_id INT NOT null unique,

    name VARCHAR(100) not null,

    created\_date timestamp default now(),

    PRIMARY KEY(team\_id)

);

CREATE TABLE cricket.players (

    row\_id INT GENERATED ALWAYS AS IDENTITY,

    player\_id INT NOT null unique,

    hand VARCHAR(100) not null,

    name VARCHAR(100) not null,

    created\_date timestamp default now(),

    PRIMARY KEY(player\_id)

);

select \* from cricket.matches;

select \* from cricket.teams;

select \* from cricket.players;

**============================================================================**

REPORT/Assessment QUERIES

**============================================================================**

**-- Question 1**

**-- Which team won each match? (i.e. who had the cumulative**

**-- highest number of runs in their innings)**

**-----------------------------------------------------------------**

with winning\_team as

(

    select

        a.\* ,rank() over (partition by a.match\_id order by a.match\_id,a.total desc)  rnk

    from (select  match\_id , batting\_team\_id,bowling\_team\_id , sum(runs) as total

            from cricket.matches cm

            group by match\_id, batting\_team\_id,bowling\_team\_id

        )  as a

)

select wt.match\_id , t1.name as "Batting team" ,t2.name as "Bowling Team" , wt.total ,

    case when rnk = 1 then t1.name else '' end as WINNER

from winning\_team wt

    left join teams t1

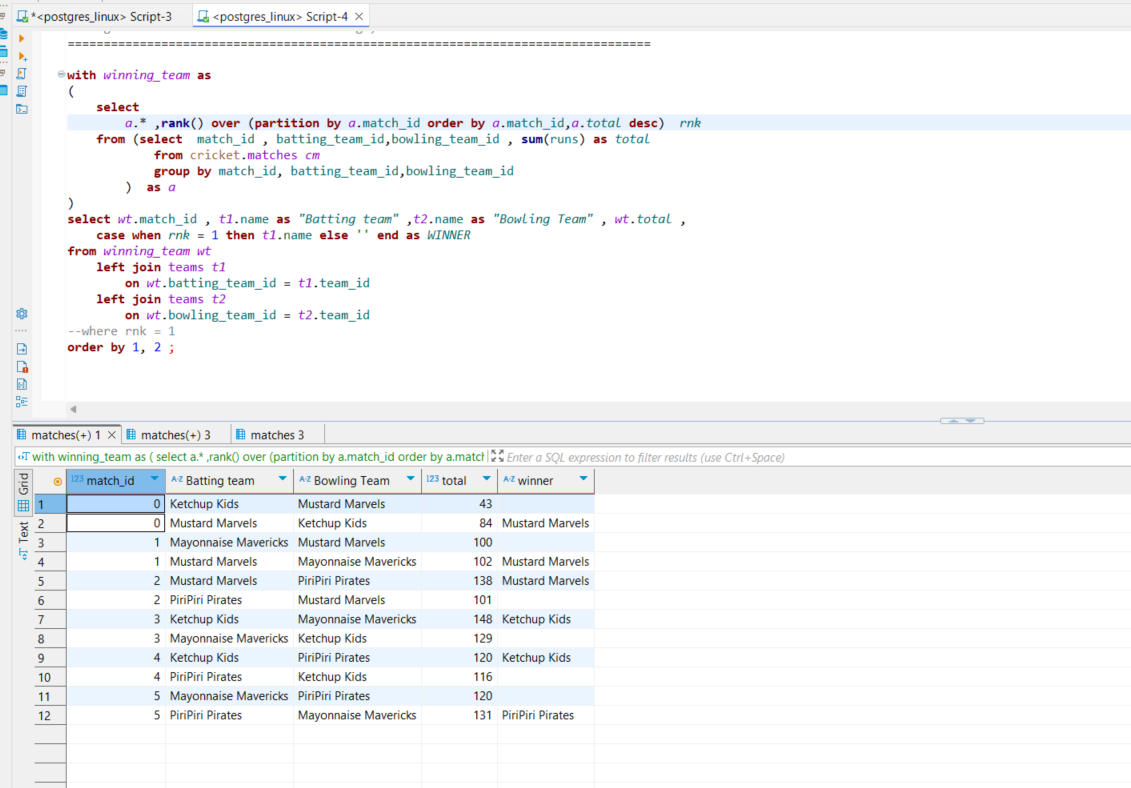
        on wt.batting\_team\_id = t1.team\_id

    left join teams t2

        on wt.bowling\_team\_id = t2.team\_id

--where rnk = 1

order by 1, 2 ;



**-- Question 2**

**-- Which “over” had the highest score for each team for each match?**

**-- (i.e. for balls in this over number, which had the cumulative highest total)**

**--------------------------------------------------------------------------------**

with over\_total as

(

    select  match\_id, bowling\_team\_id, count(1) ball\_total, count(1)/6 over\_cnt

        ,rank() over (partition by match\_id order by count(1) desc)  rnk

    from cricket.matches m

        left join teams t

            on bowling\_team\_id = t.team\_id

    group by match\_id,bowling\_team\_id

    order by 1

)

select

    ot.match\_id , t.name as team, ball\_total, over\_cnt,

    case when rnk = 1 then t.name else '' end as "Highest Total"

from over\_total ot

    left join teams t

        on ot.bowling\_team\_id = t.team\_id

--where rnk = 1

order by ot.match\_id, ball\_total ;

A screenshot of a computer

AI-generated content may be incorrect.

**-- Question 3**

**-- The average number of runs scored by each batter**

**-- across all the matches they played in.**

**---------------------------------------------------------**

select

    batter\_id , p.name , count(1) match\_played\_in, sum(runs) as total\_runs ,

    sum(runs)/count(1) ave\_no\_of\_runs

from cricket.matches cm

    inner join players p

        on cm.batter\_id = p.player\_id

group by  batter\_id , p.name order by 1

A screenshot of a computer

AI-generated content may be incorrect.