1. **Group_info.py**

```
import ConfigParser
import os
import re
from lib import get_filepaths, match_parse, match_filepaths
import pprint


# Run squad_info before running this

WORLD_CUP_TABLE_ATTRS = ['Year', 'Host_Country', 'Winner', 'Runner_Up']


def fill_wc_info(file_list):
   year_host_info = {}

   for f in file_list:
      if os.path.split(f)[1] == 'cup.txt':
         year = ((os.path.split(f)[0].split("\\"))[-1].split("--"))[0]
         file_fd = open(f, 'r')
         lines = file_fd.readlines()
         host_country = ''
         for line in lines:
            if "World Cup" in line:
               tokens = line.split()
               name_tokens = []
               for t in tokens[4:]:
                  if not re.search('[a-zA-Z]', t):
                     break
                  else:
                     name_tokens.append(re.sub(r'\W+', '', t))

               host_country = ' '.join(name_tokens)
               break

         year_host_info[year] = host_country

   return year_host_info

# This function generates the insert SQL statements for the WORLD_CUP table
def generate_world_cup_table_sql(wc_sql, year_host_info,match_path):
```

```
    (year_runnerup_winner, match_year_country) = match_parse(match_path)
    sql_file = file(wc_sql, 'w')
    for year in year_host_info:
        winner = year_runnerup_winner[year][0]
        runner_up = year_runnerup_winner[year][1]
        host = year_host_info[year]
        insert_stmt = "INSERT INTO  world_cup ({0}) VALUES ({1}, \"{2}\", \"{3}\",
\"{4}\");\n".format(
            ', '.join(WORLD_CUP_TABLE_ATTRS), year, host, winner, runner_up)
        sql_file.write(insert_stmt)

    sql_file.close()


if __name__ == '__main__':
    config = ConfigParser.RawConfigParser()
    config.read('init.cfg')
    wc_path = config.get('dataset', 'worldcup')
    wc_sql = config.get('dataset', 'worldcup_op')
    match_path = config.get('dataset', 'match')

    file_list = get_filepaths(wc_path)
    year_host_info = fill_wc_info(file_list);
    pprint.pprint(year_host_info)
    generate_world_cup_table_sql(wc_sql, year_host_info,match_path);
```

2. **lib.py**

```
import pickle
import os
from random import randrange, choice
from datetime import datetime
import re
import random
import pprint

# This function loads the country_code_dict from the stored pickle file
def getCountryCodeDict():
    with open('country_code.pickle', 'rb') as f:
        country_code_dict = pickle.load(f)
    return country_code_dict
```

```
def get_filepaths(directory):
    file_paths = []
    for root, directories, files in os.walk(directory):
        for filename in files:
            filepath = os.path.join(root, filename)
            file_paths.append(filepath)
    return file_paths

def get_player_role(tokens):
    return tokens[1]

# We are generating a random date of birth for each player. The players are always in the
# age range 20-30
def generate_random_date(year):
    date_year = choice(range(int(year) - 30, int(year) - 20))
    date_month = choice(range(1, 12))
    date_day = choice(range(1, 28))
    DOB = datetime(date_year, date_month, date_day)
    return DOB


def get_player_jersey(tokens):
    s = tokens[0]
    jersey_number = s[s.find("(") + 1:s.find(")")]
    if jersey_number.isdigit():
        return jersey_number
    else:
        jersey_number = randrange(1, 30)
        return jersey_number
    return None


def get_player_club(tokens):
    hash_index = tokens.index('##')
    club_name = ' '.join(tokens[hash_index + 2:])
    return club_name

# This function parses the squad files and fills the player and country_code_dict, players_dict,
# goal_keepers_dict, player_info_dict and the year_player_dict
def fill_squad_info(player_path):
```

```
    file_list = get_filepaths(player_path)
    country_code_dict = {}
    players_dict = {}  # This dict is a (year, country) to list of players map
    goal_keepers_dict = {}  # This dict is a (year, country) to list of goalkeepers map
    player_info_dict = {}  # This dict is a player name to player details map
    year_player_dict = {}  # This dict is a map from year to a list of (player name, country code)

    for f in file_list:
        if '\\squads\\' in f:
            tokens = os.path.split(f)

            year = ((tokens[0].split("\\"))[-2].split("--"))[0]

            # Split the file name (of the form x-y.txt, where x is the country code and y is the country
name
            file_name = tokens[1]
            file_name = file_name.replace('-', '.')

            country_tokens = file_name.split('.')
            country_code = country_tokens[0].upper()
            country_name_tokens = []
            for t in country_tokens[1:]:
                if t != "txt":
                    country_name_tokens.append(t)
            country_name = ' '.join(country_name_tokens)
            if country_name not in country_code_dict:
                country_code_dict[country_name] = country_code

            # Fill the goal keepers and players dict

            goal_keepers_in_team = []
            players_in_team = []

            file_fd = open(f, 'r')
            lines = file_fd.readlines()
            positions = ['GK', 'MF', 'FW', 'DF']
            for line in lines:
                if any(position in line for position in positions):
                    tokens = line.split()

                    player_country_code = country_code
```

```
            player_role = get_player_role(tokens)
            player_date_of_birth = generate_random_date(year)
            player_jersey_number = get_player_jersey(tokens)
            player_club = get_player_club(tokens)

            pData = [player_country_code, player_role, player_date_of_birth.strftime("%Y-%m-
%d"),
                  player_jersey_number, player_club]

            name_tokens = []
            for t in tokens[2:]:
               if not re.search('[a-zA-Z]', t):
                  break
               else:
                  name_tokens.append(re.sub(r'\W+', '', t))

            player_name = ' '.join(name_tokens)
            #players_in_team.append(player_name)

            if player_name == '':
               continue

            player_info_dict[(player_name, player_country_code)] = pData

            if 'GK' in line:
               goal_keepers_in_team.append(player_name)
            else:
               players_in_team.append(player_name)

            if year not in year_player_dict:
               year_player_dict[year] = [(player_name, player_country_code)]
            else:
               year_player_dict[year].append((player_name, player_country_code))

      if (year, country_code) not in goal_keepers_dict:
         goal_keepers_dict[(year, country_code)] = goal_keepers_in_team

      if (year, country_code) not in players_dict:
         players_dict[(year, country_code)] = players_in_team
```

```
    return country_code_dict, player_info_dict, players_dict, goal_keepers_dict,
year_player_dict


# getting cup.txt and cup_finals.txt
def match_filepaths(path):
    full_file_paths = get_filepaths(path);
    selected_file_paths = []
    for f in full_file_paths:
        if f.split('\\')[-1] == 'cup.txt' or f.split('\\')[-1] == 'cup_finals.txt' :
            selected_file_paths.append(f)
    return selected_file_paths

# filling the match table
def match_parse(match_path):
            # loading country codes for country name into dictionary from pickle dump
    country_country_code_dict = getCountryCodeDict()
            # These are few exception cases
    alternate_names=dict()
    alternate_names['germany']='deutschland'
    alternate_names['trinidad and tobago']='trinidad tobago'
    alternate_names['spain']='espana'
    alternate_names['c te d ivoire']='cote d ivoire'
    alternate_names['c\xc3\xb4te d\'ivoire']='cote d ivoire'
    alternate_names['serbia and montenegro']='serbia'
    alternate_names['bosniaherzegovina'] = 'bosnia herzegovina'



    # create dictionary for year : (winner,runnerup)
    year_runnerup_winner = {}

    # create list of (match_no,year,country_code, country_score,decision)
    match_year_country = []
    for path in match_filepaths(match_path) :
        f=open(path, 'r')
        lines= f.readlines()
        f.close()

        # choosing the starting point in the file different for cup.txt and cup_finals.txt
        start=0;
```

```
c=0;

if path.split('\\')[-1] == 'cup.txt' :
   for line in lines:
      if(line.strip().startswith('(1)')):
         start=c;
         break;
      c=c+1

if path.split('\\')[-1] == 'cup_finals.txt' :
   for line in lines:
      if(line.strip().startswith('(')):
         start=c;
         break;
      c=c+1
#print path
# extracting year
if len(path.split('\\')) > 0 :
   year = path.split('\\')[-2].split('-')[0]

else :
   year = path.split('/')[-2].split('-')[0]

setflag = 0
c = 0
# start processing each line
for line in lines:
   line = line.rstrip('\n')
   # trying to find winner and runner up for an year
   if line.startswith('final') or line.startswith('Final'):
      setflag = 1
   if(c>= start and line.startswith('(')):

      linetemp=line.split();

      # extracting match number
      match_no = linetemp[0]
      match_no = match_no.replace('(', '')
      match_no = match_no.replace(')', '')
      #print match_no
```

```
        # extracting day and month
        if linetemp[2].find('/') != -1 :
           day = linetemp[2].split('/')[1]
           month = linetemp[2].split('/')[0]
        else :
           day = linetemp[1]
           month = linetemp[2]
        #print day
        #print month

        # extracting stadium
        linetemp = line.split('@')[1]
        stadium = linetemp.split(',')[0]
        #print stadium

        # extracting stadium address
        stadiumaddress = line.split('@')[1].split('#')[0]
        #print stadiumaddress

        # extracting team1 country code
        temp1 = line.split(':')
        team1 = []
        team1new = ''

        if (len(temp1) == 2) :
           team1 = line.split(':')[1].split('-')[0].split()
           team1new = team1[1].lower()
           for t in team1[2:-1]:
              team1new = team1new + ' ' + t.lower()
        else :
           team1 = line.split('-')[0].split()
           team1new = team1[3].lower()
           for t in team1[4:-1]:
              team1new = team1new + ' ' + t.lower()

        if(country_country_code_dict.has_key(team1new)):
           team1=country_country_code_dict[team1new]
        elif(alternate_names.has_key(team1new) and
country_country_code_dict.has_key(alternate_names[team1new])):
           team1=country_country_code_dict[alternate_names[team1new]]
```

```
            # extracting team1 score
            team1score = int(line.split('-')[0].split()[-1])
            #print team1score

            # extracting team2 country_code
            team2 = []
            team2 = line.split('@')[0].split('-')[-1].split()
            team2new = team2[1].lower()
            for t in team2[2:]:
               team2new = team2new + ' ' + t.lower()
            if(country_country_code_dict.has_key(team2new)):
               team2=country_country_code_dict[team2new]
            elif(alternate_names.has_key(team2new) and
country_country_code_dict.has_key(alternate_names[team2new])):
               team2=country_country_code_dict[alternate_names[team2new]]
            #print team2

            # extracting team2 score
            team2score = int(line.split('-')[1].split()[0])
            #print team2score

            # winner determination
            if team1score > team2score :
               winner = team1
               if setflag == 1:
                  year_runnerup_winner[year] = (team1,team2)
            if team1score < team2score :
               winner = team2
               if setflag == 1:
                  year_runnerup_winner[year] = (team2,team1)
            if team1score == team2score :
               winner = 'NULL'
               decision = 'DRAW'
            #print winner

            # decision
            if team1score != team2score :
               temp = line.split('-')[1].split()[1]
               if temp[:3] == 'pen' :
                  decision = 'PENALTY'
               else :
```

```
                decision = 'WINNER'
            #print decision

            match_year_country.append((match_no,year,team1,team1score,decision))
            match_year_country.append((match_no,year,team2,team2score,decision))


        c=c+1
    #print year_runnerup_winner
    #print match_year_country
    return (year_runnerup_winner,match_year_country)


def match_played_by_parse(match_path):


    country_code_dict, player_info_dict, players_dict, goal_keepers_dict, year_player_dict =
fill_squad_info(match_path)
    year_runnerup_winner,match_year_country=match_parse(match_path)

    match_year_country_players=dict()

    for x in match_year_country:

        match_no=x[0]
        year=x[1]
        team1=x[2]
        if(players_dict.has_key((year,team1)) and goal_keepers_dict.has_key((year,team1))):
            players=players_dict[(year,team1)]
            goalkeepers=goal_keepers_dict[(year,team1)]
            ten_players=random.sample(players,10)

            match_year_country_players[match_no,year,team1]=ten_players
            one_goalkeeper=random.sample(goalkeepers,1)

    return match_year_country_players
```

3. match_info.py

```
import ConfigParser
import random
from lib import match_parse, match_played_by_parse, match_filepaths, getCountryCodeDict
```

```
# filling the match table
def match_stadium_parse_into_table(match_path,match_op_path,stadium_op_path):
	# loading country codes for country name into dictionary from pickle dump
	country_country_code_dict = getCountryCodeDict()
	# these are few exception cases
	alternate_names=dict()
	alternate_names['germany']='deutschland'
	alternate_names['trinidad and tobago']='trinidad tobago'
	alternate_names['spain']='espana'
	alternate_names['c te d ivoire']='cote d ivoire'
	alternate_names['c\xc3\xb4te d\'ivoire']='cote d ivoire'
	alternate_names['serbia and montenegro']='serbia'
	alternate_names['bosniaherzegovina'] = 'bosnia herzegovina'

	f2=open(match_op_path, 'w')
	f2.truncate()
	f2.close()

	f3=open(stadium_op_path, 'w')
	f3.truncate()
	f3.close()

	# create dictionary for year : (winner,runnerup)
	year_runnerup_winner = {}

	# create list of (match_no,year,country_code,country_score,decision)
	match_year_country = []
	distinct_stadium=[]
	for path in match_filepaths(match_path) :
		f=open(path, 'r')
		lines= f.readlines()
		f.close()

		# choosing the starting point in the file different for cup.txt and cup_finals.txt
		start=0;
		c=0;

		if path.split('\\')[-1] == 'cup.txt' :
			for line in lines:
				if(line.strip().startswith('(1)')):
```

```
                        start=c;
                        break;
                c=c+1

if path.split('\\')[-1] == 'cup_finals.txt' :
        for line in lines:
                if(line.strip().startswith('(')):
                        start=c;
                        break;
                c=c+1
#print path
# year
if len(path.split('\\')) > 0 :
        year = path.split('\\')[-2].split('-')[0]

else :
        year = path.split('/')[-2].split('-')[0]

setflag = 0
c = 0
# start processing each line
for line in lines:
        line = line.rstrip('\n')
        # trying to find winner and runner up for an year
        if line.startswith('final') or line.startswith('Final'):
                setflag = 1
        if(c>= start and line.startswith('(')):

                linetemp=line.split();

                # extracting match number
                match_no = linetemp[0]
                match_no = match_no.replace('(', '')
                match_no = match_no.replace(')', '')
                #print match_no

                # extracting day and month
                if linetemp[2].find('/') != -1 :
                        day = linetemp[2].split('/')[1]
                        month = linetemp[2].split('/')[0]
                else :
```

```
                day = linetemp[1]
                month = linetemp[2]
        #print day
        #print month

        # extracting stadium
        linetemp = line.split('@')[1]
        stadium = linetemp.split(',')[0].lower()
        stadium =stadium.replace("'","")


        #print stadium

        # extracting stadium address
        stadiumaddress = line.split('@')[1].split('#')[0].lower()
        stadiumaddress=stadiumaddress.replace("'","")

        #print stadiumaddress

        # extracting team1 country code
        temp1 = line.split(':')
        team1 = []
        team1new = ''

        if (len(temp1) == 2) :
                team1 = line.split(':')[1].split('-')[0].split()
                team1new = team1[1].lower()
                for t in team1[2:-1]:
                        team1new = team1new + ' ' + t.lower()
        else :
                team1 = line.split('-')[0].split()
                team1new = team1[3].lower()
                for t in team1[4:-1]:
                        team1new = team1new + ' ' + t.lower()

        if(country_country_code_dict.has_key(team1new)):
                team1=country_country_code_dict[team1new]
        elif(alternate_names.has_key(team1new) and
country_country_code_dict.has_key(alternate_names[team1new])):

        team1=country_country_code_dict[alternate_names[team1new]]
```

```
                # extracting team1 score
                team1score = int(line.split('-')[0].split()[-1])
                #print team1score

                # extracting team2
                team2 = []
                team2 = line.split('@')[0].split('-')[-1].split()
                team2new = team2[1].lower()
                for t in team2[2:]:
                        team2new = team2new + ' ' + t.lower()

                if(country_country_code_dict.has_key(team2new)):
                        team2=country_country_code_dict[team2new]
                elif(alternate_names.has_key(team2new) and
country_country_code_dict.has_key(alternate_names[team2new])):

        team2=country_country_code_dict[alternate_names[team2new]]
                #print team2

                # team2 score
                team2score = int(line.split('-')[1].split()[0])
                #print team2score

                # winner determination
                if team1score > team2score :
                        winner = team1
                        if setflag == 1:
                                year_runnerup_winner[year] = (team1,team2)
                if team1score < team2score :
                        winner = team2
                        if setflag == 1:
                                year_runnerup_winner[year] = (team2,team1)
                if team1score == team2score :
                        winner = 'NULL'
                        decision = 'DRAW'
                #print winner

                # decision
                if team1score != team2score :
                        temp = line.split('-')[1].split()[1]
```

```
                                        if temp[:3] == 'pen' :
                                                decision = 'PENALTY'
                                        else :
                                                decision = 'WINNER'
                                #print decision


        match_year_country.append((match_no,year,team1,team1score,decision))

        match_year_country.append((match_no,year,team2,team2score,decision))

                                #print match_no, year

                                stmt= 'insert into matches (Stadium, Match_Number, Winner,
Decision, Team_1, Team_2, Team_1_Score, Team_2_Score, Date_Day, Date_Month,
Date_Year) values
(\'{0}\',{1},\'{2}\',\'{3}\',\'{4}\',\'{5}\',{6},{7},{8},\'{9}\',{10});\n'.format(stadium, match_no,
winner, decision, team1, team2, team1score, team2score, day, month, year)
                                print stmt

                                f2=open(match_op_path, 'a')
                                f2.write(stmt)
                                f2.close()

                                if(stadium not in distinct_stadium):
                                        stmt= 'insert into stadium (Stadium, Stadium_Address)
values (\'{0}\',\'{1}\');\n'.format(stadium,stadiumaddress)
                                        #stmt= '\'{0}\',\'{1}\''.format(stadium,stadiumaddress)
                                        print stmt

                                        f3=open(stadium_op_path, 'a')
                                        f3.write(stmt)
                                        stmt=''
                                        f3.close()
                                        distinct_stadium.append(stadium)



                        c=c+1
        #print year_runnerup_winner
        #print match_year_country
```

```
	return (year_runnerup_winner,match_year_country)



# This function fills goal_and_player_scores_relation
def goal_parse(match_path,goal_op_path):

	f2=open(goal_op_path, 'w')
	f2.truncate()
	f2.close()
	# match_year_country_list_of_players  (It gives the list of 10 randomely selected players who are potential candidates for scoring goals)
	match_year_country_players = match_played_by_parse(match_path)
	#print match_year_country_players
	year_runnerup_winner,match_year_country = match_parse(match_path)
	matchno_year_time = list()
	for entry in match_year_country:
		#print entry
		year = entry[1]
		match_no = entry[0]
		country = entry[2]
		list_of_players = match_year_country_players[(match_no,year,country)]
		#print entry[3]

		# randomly chose goal scoring times in minutes based on the number of goals scored by each team. In case of penalty the times were chosen between 120 and 130 mins
		chosen_time=list()
		for x in range(0,entry[3]) :
			if entry[4] != 'PENALTY' :
				time = random.randint(1,120)
				while(time in chosen_time or (match_no,year,time) in matchno_year_time):
					time = random.randint(1,120)
				chosen_time.append(time)
				#print time
			else:
				time = random.randint(121,130)
				while(time in chosen_time or (match_no,year,time) in matchno_year_time):
					time = random.randint(121,130)
```

```
                    chosen_time.append(time)
                    #print time

                matchno_year_time.append((match_no,year,time))

                num = random.randint(0,len(list_of_players)-1)
                player = list_of_players[num]

                stmt= 'insert into goal_and_player_scores_goals
(Match_Number,Date_Year,Recorded_Time,Player_Name,Country_Code) values
({0},{1},{2},\'{3}\',\'{4}\');\n'.format(match_no,year,time,player,country)
                print stmt

                f2=open(goal_op_path, 'a')
                f2.write(stmt)
                f2.close()

config = ConfigParser.RawConfigParser()
config.read('init.cfg')
match_path = config.get('dataset', 'match')
match_op_path=config.get('dataset', 'match_op')
stadium_op_path=config.get('dataset', 'stadium_op')
match_played_by_op = config.get('dataset', 'match_played_by_op')
goal_op_path=config.get('dataset', 'goal_op')

match_stadium_parse_into_table(match_path, match_op_path, stadium_op_path)
goal_parse(match_path,goal_op_path);
```

4. match_played_by_info.py

```
import ConfigParser
import random
from lib import match_parse, fill_squad_info
import pprint

#correlated data of players and matches each year
def match_played_by_parse_into_table(match_path, match_op_path):
   #get player info for each year
   country_code_dict, player_info_dict, players_dict, goal_keepers_dict, year_player_dict =
fill_squad_info(match_path)
   #get matches played in each year
   (year_runnerup_winner,match_year_country)=match_parse(match_path)
```

```
   match_year_country_players=dict()

   f2=open(match_played_by_op, 'w')
   f2.truncate()
   f2.close()

   for x in match_year_country:
      match_no=x[0]
      year=x[1]
      team1=x[2]
      if(players_dict.has_key((year,team1)) and goal_keepers_dict.has_key((year,team1))):
         players=players_dict[(year,team1)]
         goalkeepers=goal_keepers_dict[(year,team1)]
         #extracting 10 random players among squad that match
         ten_players=random.sample(players,10)

         #extracting 1 random goal keeper among squad that match
         match_year_country_players[match_no,year,team1]=ten_players
         one_goalkeeper=random.sample(goalkeepers,1)


         for player in ten_players:
            stmt= 'insert into match_played_by (Match_Number, Date_Year, Country_Code,
Player_Name) values ({0},{1},\'{2}\',\'{3}\');\n'.format(match_no, year, team1, player)

            f2=open(match_played_by_op, 'a')
            f2.write(stmt)
            f2.close()

         for keeper in one_goalkeeper:
            stmt= 'insert into match_played_by (Match_Number, Date_Year, Country_Code,
Player_Name) values ({0},{1},\'{2}\',\'{3}\');\n'.format(match_no, year, team1, keeper)

            f2=open(match_played_by_op, 'a')
            f2.write(stmt)
            f2.close()

   return match_year_country_players
```

```
config = ConfigParser.RawConfigParser()
config.read('init.cfg')
match_path=config.get('dataset', 'match')
match_played_by_op = config.get('dataset', 'match_played_by_op')


match_played_by_parse_into_table(match_path, match_played_by_op)
```

5. player_info.py

```
import ConfigParser
import os
import re
from random import randrange, choice
from datetime import datetime
import pickle
from lib import get_filepaths, fill_squad_info


PLAYER_TABLE_TUPLES = ['Country_Code', 'Player_Role', 'Player_Name', 'DOB',
'Jersey_Number', 'Club']
WORLD_CUP_PLAYED_BY_TABLE_TUPLES = ['Year', 'Player_Name', 'Country_Code']

# This function generates the insert sql statements for the Player table and the
# world_cup_played_by_player table
def generate_player_table_sql(player_info_dict, year_played_dict):
   sql_file_1 = file(player_sql, 'w')
   sql_file_2 = file(worldcup_played_sql, 'w')

   for player_name, country_code in player_info_dict:
      player_role = player_info_dict[(player_name,country_code)][1]
      DOB = player_info_dict[(player_name,country_code)][2]
      jersey_no = player_info_dict[(player_name,country_code)][3]
      club = player_info_dict[(player_name,country_code)][4]

      insert_stmt = "INSERT INTO player ({0}) VALUES (\"{1}\", \"{2}\", \"{3}\", {4}, {5},
\"{6}\");\n".format(
         ', '.join(PLAYER_TABLE_TUPLES), country_code, player_role, player_name, DOB,
jersey_no, club)
      #print insert_stmt

      sql_file_1.write(insert_stmt)
```

```
    for year in year_played_dict:
        players_country_list = year_played_dict[year]
        for pl_name, country_code in players_country_list:
            insert_stmt = "INSERT INTO world_cup_played_by_player ({0}) VALUES ({1}, \"{2}\",
\"{3}\");\n".format(
            ', '.join(WORLD_CUP_PLAYED_BY_TABLE_TUPLES), year, pl_name, country_code)
            sql_file_2.write(insert_stmt)
            #print insert_stmt

    sql_file_1.close()
    sql_file_2.close()


if __name__ == '__main__':
    config = ConfigParser.RawConfigParser()
    config.read('init.cfg')
    player_path = config.get('dataset', 'player')

    player_sql = config.get('dataset', 'player_op')
    worldcup_played_sql = config.get('dataset', 'worldcup_played_by_op')


    country_code_dict, player_info_dict, players_dict, goal_keepers_dict, year_played_dict =
fill_squad_info(player_path);
    generate_player_table_sql(player_info_dict, year_played_dict);

    # Dump the country code dict to a pickle file
    with open('country_code.pickle', 'wb') as handle:
        pickle.dump(country_code_dict, handle)
```

6. team_info.py

```
import ConfigParser
import re
import lib

#Add countries that do not have ranking and point info
def fillAdditonalInfo(dynamic_country_list,country_code_dict,country_assoc):

    for (country_name, country_code) in country_code_dict.iteritems():
        if(dynamic_country_list.count(country_name)==0):
```

```
        association=''
        if country_assoc.has_key(country_name):
           association=country_assoc[country_name]

        stmt= 'insert into team (Country_Code, Country_Name, Association) values
(\'{0}\',\'{1}\',\'{2}\');\n'.format(country_code, country_name, association)

        f2=open(team_op_path, 'a')
        f2.write(stmt)
        f2.close()

#parse country statistics and also extract association and country code
def world_cup_parse(team_path,team_op_path, country_assoc_path):

   #fixing some country names spelled differently
   alternate_names=dict()
   alternate_names['germany']='deutschland'
   alternate_names['trinidad and tobago']='trinidad tobago'
   alternate_names['spain']='espana'
   alternate_names['c te d ivoire']='cote d ivoire'

   #get country code
   country_code_dict = lib.getCountryCodeDict()

   dynamic_country_list = list()

   f=open(team_path, 'r')
   lines= f.readlines()
   f.close()

   f3=open(country_assoc_path, 'r')
   a_lines= f3.readlines()
   f3.close()

   f2=open(team_op_path, 'w')
   f2.truncate()
   f2.close()


   country_assoc=dict()
```

```
#extract country name and country association info
for a_line in a_lines:
   country=a_line.split(',')[0].strip()
   assoc=a_line.split(',')[1].strip()
   country_assoc[country]=assoc

start=0;
c=0;
for line in lines:
   if(line.strip().startswith('1 ')):
      start=c;
      break;
   c=c+1

c=0;
#extract country name, ranking and points info
for line in lines:
   if(c>= start):
      line=line.strip();

      t=re.split('[^a-zA-Z0-9_()]',line)
      t= filter(None, t)

      i=0
      team=' '

      for token in t:
         if(i==0):
            i=i+1
            continue
         if(token.startswith('(')):
            break
         team= team+" "+token
         i=i+1

      ranking=t[3+(i-2)]
      points=t[9+(i-2)]

      team=team.lstrip().lower()

      if(country_code_dict.has_key(team)):
```

```
            teamcode=country_code_dict[team]
        elif(alternate_names.has_key(team) and
country_code_dict.has_key(alternate_names[team])):
            teamcode=country_code_dict[alternate_names[team]]
            team=alternate_names[team]

        else:
            continue

        dynamic_country_list.append(team)

        if country_assoc.has_key(team):
            association=country_assoc[team]
            stmt= 'insert into team (Country_Code, Country_Name,Association, Points, Ranking)
values (\'{0}\',\'{1}\',\'{2}\',{3},{4});\n'.format(teamcode, team,association,points, ranking)
        else:
            stmt= 'insert into team (Country_Code, Country_Name, Points, Ranking) values
(\'{0}\',\'{1}\',{2},{3});\n'.format(teamcode, team,points, ranking)

        f2=open(team_op_path, 'a')
        f2.write(stmt)
        f2.close()


    c=c+1

  fillAdditonalInfo(dynamic_country_list,country_code_dict,country_assoc)


config = ConfigParser.RawConfigParser()
config.read('init.cfg')
team_path = config.get('dataset', 'team')
team_op_path=config.get('dataset', 'team_op')
country_assoc_path= config.get('dataset', 'country_assoc')

world_cup_parse(team_path, team_op_path,country_assoc_path)
```

7. world_cup_info.py

```
import ConfigParser
import os
import re
```

```
from lib import get_filepaths, match_parse, match_filepaths
import pprint


# Run squad_info before running this

WORLD_CUP_TABLE_ATTRS = ['Year', 'Host_Country', 'Winner', 'Runner_Up']


def fill_wc_info(file_list):
   year_host_info = {}

   for f in file_list:
      if os.path.split(f)[1] == 'cup.txt':
         year = ((os.path.split(f)[0].split("\\"))[-1].split("--"))[0]
         file_fd = open(f, 'r')
         lines = file_fd.readlines()
         host_country = ''
         for line in lines:
            if "World Cup" in line:
               tokens = line.split()
               name_tokens = []
               for t in tokens[4:]:
                  if not re.search('[a-zA-Z]', t):
                     break
                  else:
                     name_tokens.append(re.sub(r'\W+', '', t))

               host_country = ' '.join(name_tokens)
               break

         year_host_info[year] = host_country

   return year_host_info

# This function generates the insert SQL statements for the WORLD_CUP table
def generate_world_cup_table_sql(wc_sql, year_host_info,match_path):
   (year_runnerup_winner, match_year_country) = match_parse(match_path)
   sql_file = file(wc_sql, 'w')
   for year in year_host_info:
      winner = year_runnerup_winner[year][0]
```

```
        runner_up = year_runnerup_winner[year][1]
        host = year_host_info[year]
        insert_stmt = "INSERT INTO  world_cup ({0}) VALUES ({1}, \"{2}\", \"{3}\",
\"{4}\");\n".format(
            ', '.join(WORLD_CUP_TABLE_ATTRS), year, host, winner, runner_up)
        sql_file.write(insert_stmt)

    sql_file.close()


if __name__ == '__main__':
    config = ConfigParser.RawConfigParser()
    config.read('init.cfg')
    wc_path = config.get('dataset', 'worldcup')
    wc_sql = config.get('dataset', 'worldcup_op')
    match_path = config.get('dataset', 'match')

    file_list = get_filepaths(wc_path)
    year_host_info = fill_wc_info(file_list);
    pprint.pprint(year_host_info)
    generate_world_cup_table_sql(wc_sql, year_host_info,match_path);
```