**1st table**

import requests

import json

url=https://api.sportradar.com/tennis/trial/v3/en/competitions.json?api\_key=3dstNx7HEv0FRTeTImTr8C4CgiohsuJVGUEOzwLV"

headers = {"accept": "application/json"}

response = requests.get(url, headers=headers)

print(response.text)

mycursor.execute("CREATE TABLE Categories (category\_id VARCHAR(50) PRIMARY KEY)")

mycursor.execute("ALTER TABLE Categories ADD category\_name VARCHAR(100) NOT NULL")

mycursor.execute("CREATE TABLE Competitions (competition\_id VARCHAR(50) PRIMARY KEY, competition\_name VARCHAR(100) NOT NULL,parent\_id VARCHAR(50),type VARCHAR(20) NOT NULL,gender VARCHAR(10) NOT NULL,category\_id VARCHAR(50) NOT NULL,FOREIGN KEY (category\_id)REFERENCES Categories(category\_id))")

data=json.loads(response.text)

competitions=data["competitions"]

competitions\_list=[]

for comp in competitions:

competitions\_list.append({

"id":comp["id"],

"name":comp["name"],

"parent\_id":comp.get("parent\_id",None),

"type":comp["type"],

"gender":comp["gender"],

"category\_id":comp["category"]["id"]

})

category\_list=[]

for comp in competitions:

category\_list.append({

"category\_id":comp["category"]["id"],

"category\_name":comp["category"]["name"]

})

category\_df = pd.DataFrame(category\_list)

category\_df

category\_df = category\_df.drop\_duplicates()

category\_df

competition\_df = pd.DataFrame(competitions\_list)

competition\_df

data\_list=category\_df.values.tolist()

query="""INSERT INTO categories(category\_id,category\_name) VALUES(%s,%s);

"""

mycursor.executemany(query,data\_list)

mydb.commit()

print("Data inserted using to\_list()")

data1\_list=competition\_df.values.tolist()

query="""INSERT INTO competitions(competition\_id,competition\_name,type,gender,category\_id) VALUES(%s,%s,%s,%s,%s);

"""

mycursor.executemany(query,data1\_list)

mydb.commit()

print("Data inserted using to\_list()")

from tabulate import tabulate

1. **COLLECT THE COMPETITION DATA FROM THE API ENDPOINTS-**

**Data Analysis:**

       Execute the Following SQL queries:

1. List all competitions along with their category name

mycursor.execute("""SELECT competitions.competition\_name,categories.category\_name

FROM competitions

INNER JOIN categories

ON competitions.category\_id = categories.category\_id;

ORDER BY competitions.competition\_id""")

out=mycursor.fetchall()

print(tabulate(out, headers=[i[0] for i in mycursor.description], tablefmt='psql'))

1. Count the number of competitions in each category

mycursor.execute("""

SELECT ca.category\_name, COUNT(co.competition\_id)

FROM categories AS ca

JOIN competitions AS co

ON co.category\_id = ca.category\_id

GROUP BY ca.category\_id""")

out = mycursor.fetchall()

print(tabulate(out, headers=[i[0] for i in mycursor.description], tablefmt='psql'))

1. Find all competitions of type 'doubles'

mycursor.execute("""SELECT co.competition\_name, co.type

FROM competitions as co

WHERE type='doubles';""")

out = mycursor.fetchall()

print(tabulate(out, headers=[i[0] for i in mycursor.description], tablefmt='psql'))

1. Get competitions that belong to a specific category (e.g., ITF Men)

mycursor.execute("""SELECT competitions.competition\_name, categories.category\_name

FROM competitions

INNER JOIN categories

ON competitions.category\_id = categories.category\_id

WHERE categories.category\_name='ITF Men';""")

out=mycursor.fetchall()

print(tabulate(out, headers=[i[0] for i in mycursor.description], tablefmt='psql'))

1. Identify parent competitions and their sub-competitions

mycursor.execute("""SELECT p.competition\_name AS parent\_competition,

c.competition\_name AS sub\_competition

FROM competitions AS p

JOIN competitions AS c

ON p.competition\_id = c.parent\_id;""")

out=mycursor.fetchall()

print(tabulate(out, headers=[i[0] for i in mycursor.description], tablefmt='psql'))

1. Analyze the distribution of competition types by category

mycursor.execute("""SELECT categories.category\_name, competitions.type

FROM categories

JOIN competitions

ON competitions.category\_id = categories.category\_id;

GROUP BY categories.category\_name""")

out = mycursor.fetchall()

print(tabulate(out, headers=[i[0] for i in mycursor.description], tablefmt='psql'))

1. List all competitions with no parent (top-level competitions)

mycursor.execute("""SELECT competitions.competition\_id, competitions.competition\_name, competitions.parent\_id, categories.category\_name

FROM competitions

JOIN categories

ON competitions.category\_id = categories.category\_id;

WHERE competitions.parent\_competition\_id IS NULL

GROUP BY categories.category\_name""")

out = mycursor.fetchall()

print(tabulate(out, headers=[i[0] for i in mycursor.description], tablefmt='psql'))

import requests

url="https://api.sportradar.com/tennis/trial/v3/en/complexes.json?api\_key=3dstNx7HEv0FRTeTImTr8C4CgiohsuJVGUEOzwLV"

headers = {"accept": "application/json"}

response = requests.get(url, headers=headers)

print(response.text)

data=json.loads(response.text)

complexes=data["complexes"]

mycursor.execute("CREATE TABLE Complexes (complex\_id VARCHAR(50) PRIMARY KEY, complex\_name VARCHAR(100) NOT NULL)")

mycursor.execute("CREATE TABLE Venues (venue\_id VARCHAR(50) PRIMARY KEY, venue\_name VARCHAR(100) NOT NULL, city\_name VARCHAR(100) NOT NULL, country\_name VARCHAR(100) NOT NULL, country\_code CHAR(3) NOT NULL, time\_zone VARCHAR(100) NOT NULL, complex\_id VARCHAR(50) NOT NULL,FOREIGN KEY (complex\_id)REFERENCES Complexes(complex\_id))")

data\_list=complex\_df.values.tolist()

query="""INSERT INTO complexes(complex\_id,complex\_name) VALUES (%s,%s);

"""

mycursor.executemany(query,data\_list)

mydb.commit()

print("data inserted")

complex\_df = complex\_df.drop\_duplicates()

complex\_df

data1\_list=venue\_df.values.tolist()

query="""

INSERT INTO Venues(venue\_id,venue\_name,city\_name,country\_name,country\_code,time\_zone, complex\_id) VALUES (%s,%s,%s,%s,%s,%s,%s);

"""

mycursor.executemany(query,data1\_list)

mydb.commit()

print("data inserted")

from tabulate import tabulate

**2) COLLECT THE COMPLEXES DATA FROM THE API ENDPOINTS-**

Execute the following SQL queries:

1. List all venues along with their associated complex name

mycursor.execute("""SELECT venues.venue\_name,complexes.complex\_name

FROM venues

INNER JOIN complexes

ON venues.complex\_id = complexes.complex\_id;

ORDER BY venues.venue\_id""")

out=mycursor.fetchall()

print(tabulate(out, headers=[i[0] for i in mycursor.description], tablefmt='psql'))

1. Count the number of venues in each complex

mycursor.execute("""SELECT complexes.complex\_name , COUNT(venues.venue\_id)

FROM venues

INNER JOIN complexes

ON complexes.complex\_id=venues.complex\_id GROUP BY complexes.complex\_id;""")

out=mycursor.fetchall()

print(tabulate(out, headers=[i[0] for i in mycursor.description], tablefmt='psql'))

1. Get details of venues in a specific country (e.g., Chile)

mycursor.execute("""SELECT \* from venues WHERE country\_name= 'chile';""")

out=mycursor.fetchall()

print(tabulate(out, headers=[i[0] for i in mycursor.description], tablefmt='psql'))

1. Identify all venues and their timezones

mycursor.execute("SELECT venue\_name, time\_zone from venues;")

out=mycursor.fetchall()

print(tabulate(out, headers=[i[0] for i in mycursor.description], tablefmt='psql'))

1. Find complexes that have more than one venue

mycursor.execute("""SELECT complex\_name, COUNT(venue\_id)

FROM venues

JOIN complexes

ON venues.complex\_id = complexes.complex\_id

GROUP BY complexes.complex\_id

HAVING COUNT(venue\_id) > 1;""")

out=mycursor.fetchall()

print(tabulate(out, headers=[i[0] for i in mycursor.description], tablefmt='psql'))

1. List venues grouped by country

mycursor.execute("""SELECT venue\_name,country\_name FROM venues

GROUP BY country\_name

ORDER BY country\_name;""")

out = mycursor.fetchall()

print(tabulate(out, headers=[i[0] for i in mycursor.description], tablefmt='psql'))

1. Find all venues for a specific complex (e.g., Nacional)

mycursor.execute("""SELECT venues.venue\_name, complexes.complex\_name

FROM complexes

JOIN venues

ON venues.complex\_id = complexes.complex\_id

WHERE complexes.complex\_name='nacional';""")

out = mycursor.fetchall()

print(tabulate(out, headers=[i[0] for i in mycursor.description], tablefmt='psql'))

import requests

url="https://api.sportradar.com/tennis/trial/v3/en/double\_competitors\_rankings.json?api\_key=3dstNx7HEv0FRTeTImTr8C4CgiohsuJVGUEOzwLV"

headers = {"accept": "application/json"}

response = requests.get(url, headers=headers)

print(response.text)

data=json.loads(response.text)

response.text

rankings=data["rankings"]

import pandas as pd

data = response.json()

rankings = data["rankings"]

all\_ranks = []

for group in rankings:

for r in group["competitor\_rankings"]:

all\_ranks.append({

"rank": r["rank"],

"movement": r["movement"],

"points": r["points"],

"competitions\_played": r["competitions\_played"],

"competitor\_id": r["competitor"]["id"]

})

for i, row in enumerate(all\_ranks):

row["rank\_id"] = i + 1

competitor\_rankings\_df = pd.DataFrame(all\_ranks)[["rank\_id", "rank", "movement", "points", "competitions\_played", "competitor\_id"]]

print(competitor\_rankings\_df)

competitors = []

for item in rankings:

for c in item["competitor\_rankings"]:

comp = c["competitor"]

competitors.append({

"competitor\_id": comp.get("id"),

"name": comp.get("name"),

"country": comp.get("country"),

"country\_code": comp.get("country\_code"),

"abbreviation": comp.get("abbreviation")

})

competitors\_df = pd.DataFrame(competitors)

print(competitors\_df.head())

mycursor.execute("DESCRIBE competitor\_rankings")

out = mycursor.fetchall()

print(tabulate(out, headers=[i[0] for i in mycursor.description], tablefmt='psql'))

mycursor.execute("CREATE TABLE Competitors(competitor\_id VARCHAR(50) PRIMARY KEY, name VARCHAR(10) NOT NULL, country VARCHAR(10) not null, country\_code CHAR(3) not null, abbreviations VARCHAR(10) not null)")

mycursor.execute("CREATE TABLE Competitor\_Rankings(rank\_id INT PRIMARY KEY, rank INT NOT NULL, movement INT not null, points INT not null, competitions\_played INT not null, competitor\_id VARCHAR(50), FOREIGN KEY (competitor\_id) References competitors(competitor\_id))")

data\_list1=competitors\_df.values.tolist()

query="""

INSERT INTO Competitors(competitor\_id,name,country,country\_code,abbreviations) VALUES (%s,%s,%s,%s,%s);

"""

mycursor.executemany(query,data\_list1)

mydb.commit()

print("data inserted")

data\_list=competitor\_rankings\_df.values.tolist()

query="""INSERT INTO competitor\_rankings(rank\_id,rank,movement,points,competitions\_played,competitor\_id) VALUES (%s,%s,%s,%s,%s,%s); """

mycursor.executemany(query,data\_list)

mydb.commit()

print("data inserted")

from tabulate import tabulate

**3) COLLECT THE DOUBLES COMPETITOR RANKINGS DATA FROM THE API ENDPOINTS-**

Execute the following SQL queries:

1. Get all competitors with their rank and points.

mycursor.execute("""SELECT competitors.name, competitor\_rankings.rank, competitor\_rankings.points

FROM competitors

JOIN competitor\_rankings

WHERE competitors.competitor\_id=competitor\_rankings.competitor\_id;""")

out=mycursor.fetchall()

print(tabulate(out, headers=[i[0] for i in mycursor.description], tablefmt='psql'))

1. Find competitors ranked in the top 5

mycursor.execute("""SELECT competitors.name, competitor\_rankings.rank

FROM competitors

JOIN competitor\_rankings

ON competitors.competitor\_id=competitor\_rankings.competitor\_id

WHERE competitor\_rankings.rank BETWEEN 1 AND 5 ORDER BY competitor\_rankings.rank ASC;""")

out=mycursor.fetchall()

print(tabulate(out, headers=[i[0] for i in mycursor.description], tablefmt='psql'))

1. List competitors with no rank movement (stable rank)

mycursor.execute("""SELECT competitors.name, competitor\_rankings.movement

FROM competitors

JOIN competitor\_rankings

ON competitors.competitor\_id=competitor\_rankings.competitor\_id

WHERE competitor\_rankings.movement=0;""")

out=mycursor.fetchall()

print(tabulate(out, headers=[i[0] for i in mycursor.description], tablefmt='psql'))

1. Get the total points of competitors from a specific country (e.g., Croatia)

mycursor.execute("""SELECT competitors.country, SUM(competitor\_rankings.points) AS total\_points

FROM competitors

JOIN competitor\_rankings

ON competitors.competitor\_id=competitor\_rankings.competitor\_id

WHERE competitors.country='Croatia'

GROUP BY competitors.country;""")

out=mycursor.fetchall()

print(tabulate(out, headers=[i[0] for i in mycursor.description], tablefmt='psql'))

1. Count the number of competitors per country

mycursor.execute("""SELECT competitors.country, COUNT(competitors.competitor\_id)

FROM competitors

GROUP BY competitors.country;""")

out = mycursor.fetchall()

print(tabulate(out, headers=[i[0] for i in mycursor.description], tablefmt='psql'))

1. Find competitors with the highest points in the current week

mycursor.execute("""SELECT competitors.name, competitor\_rankings.points

FROM competitors

JOIN competitor\_rankings

ON competitors.competitor\_id=competitor\_rankings.competitor\_id

WHERE competitor\_rankings.points = (SELECT MAX(points) FROM competitor\_rankings);""")

out=mycursor.fetchall()

print(tabulate(out, headers=[i[0] for i in mycursor.description], tablefmt='psql'))

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*