Olympics Data Management

Project Report

Vamsi Krishna Jilla
Akshat Jain

(857) 763-7256
(857) 334-6863

jilla.v@northeastern.edu
jain.akshat2@northeastern.edu

Percentage of Effort Contributed by Vamsi: 50%

Percentage of Effort Contributed by Akshat: 50%

Signature of Student 1: Vamsi Krishna Jilla

Signature of Student 2: Akshat Jain

Submission Date: 12-10-2023

Business Problem

A robust DBMS is required for organizing a humongous event like olympics where multiple parties are involved. It streamlines vast amounts of data available regarding athletes, events, staff and sponsors. It majorly solves three problems as follows

- **Data Organization and Retrieval:** Managing a large amount of information about athletes, events, and results during the Olympics is challenging without a structured system
- Accuracy and Integrity: Ensuring the accuracy and integrity of Olympic data, including athlete details and event results, is prone to errors and inconsistencies without a robust DBMS
- Decision-Making Support: Making informed decisions, such as scheduling events, awarding medals and assigning staff is very hard for the organizing committee without a reliable database system providing real-time and accurate information during the Olympics data

Introduction

- The designed DBMS for Olympic organizers is capable of both storing and retrieving data related to athletes, events, sports, sponsors, and staff representing both the country and the organizers
- Furthermore, the DBMS also keeps track of medal counts (gold, silver, and bronze) as well as the total medal count for each country
- It helps event organizers efficiently handle their staff by easily assigning them specific roles for all events taking place in various stadiums based on the schedule.
- The DBMS supports the efficient management of sponsors by storing detailed information, tracking sponsorships for both events and countries, managing financial aspects, enabling organizers to build successful partnerships

Theory behind our project:

Our project's primary objective is to develop a Database Management System (DBMS) to assist the Olympic organizers in efficiently planning and executing the various events associated with the Games.

The key entities within our project framework are as follows:

- Country: Countries will participate in various Olympic events, with each country required to engage in at least one event. These countries can possess attributes such as a unique country ID and points accrued during the competition.

- Athlete: Athletes, representing their respective countries, will participate in Olympic events. Each athlete will have a distinct ID and will exclusively represent one country. In team sports, athletes may also be part of larger teams.

- Sport: The Olympics will feature a diverse range of sports, each with multiple events in both male and female categories. These sports will be identified by unique sport IDs and will contain at least one associated event.

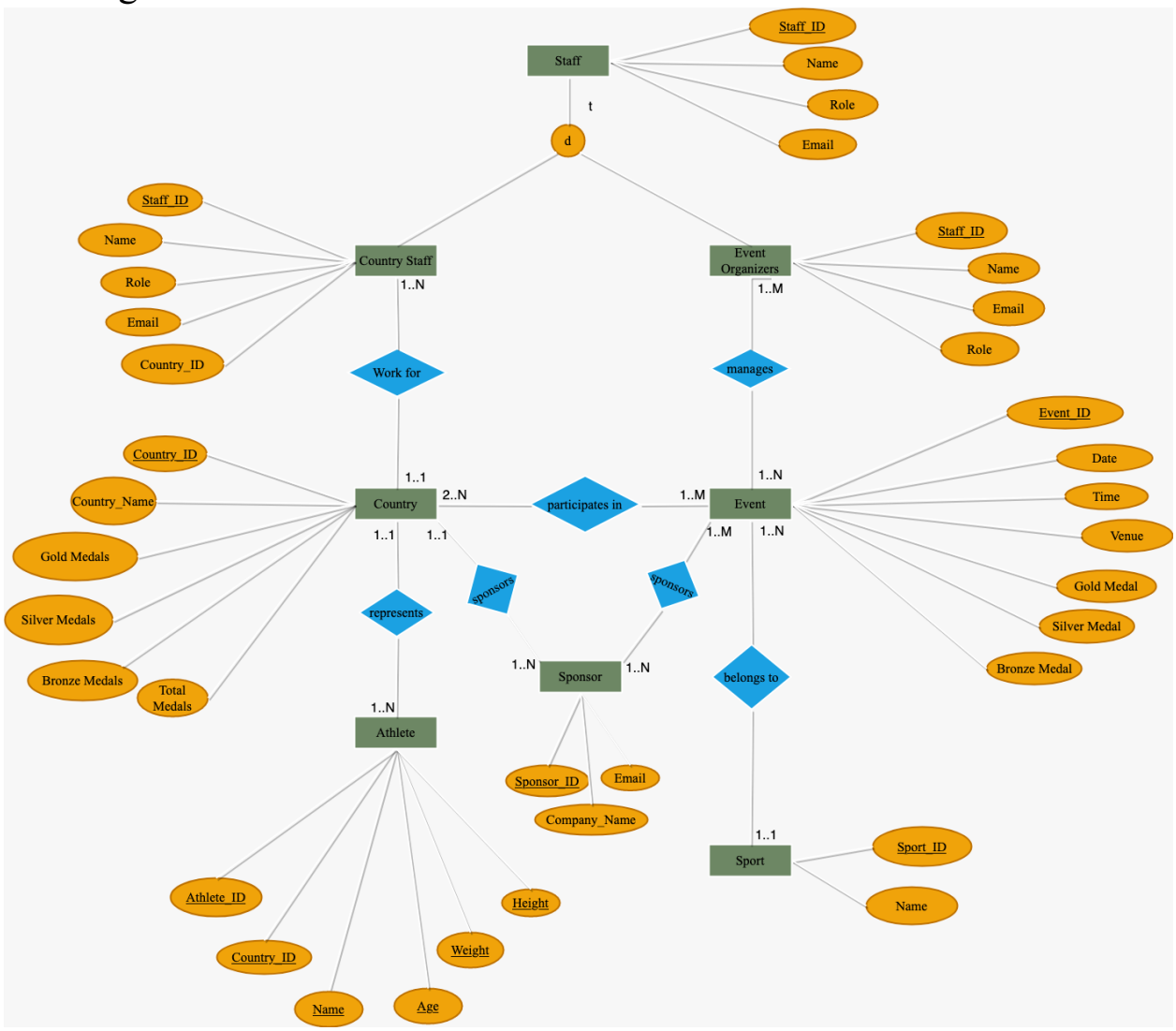
- Event: Each Olympic sport will encompass specific events, each characterized by a unique event ID, a sport ID specifying its parent sport, gender categorization (male or female), and other relevant details.

- Sponsor: The project will also accommodate sponsors, both for specific events and for countries/athletes. Sponsors may choose to support particular events or affiliate themselves with specific countries."

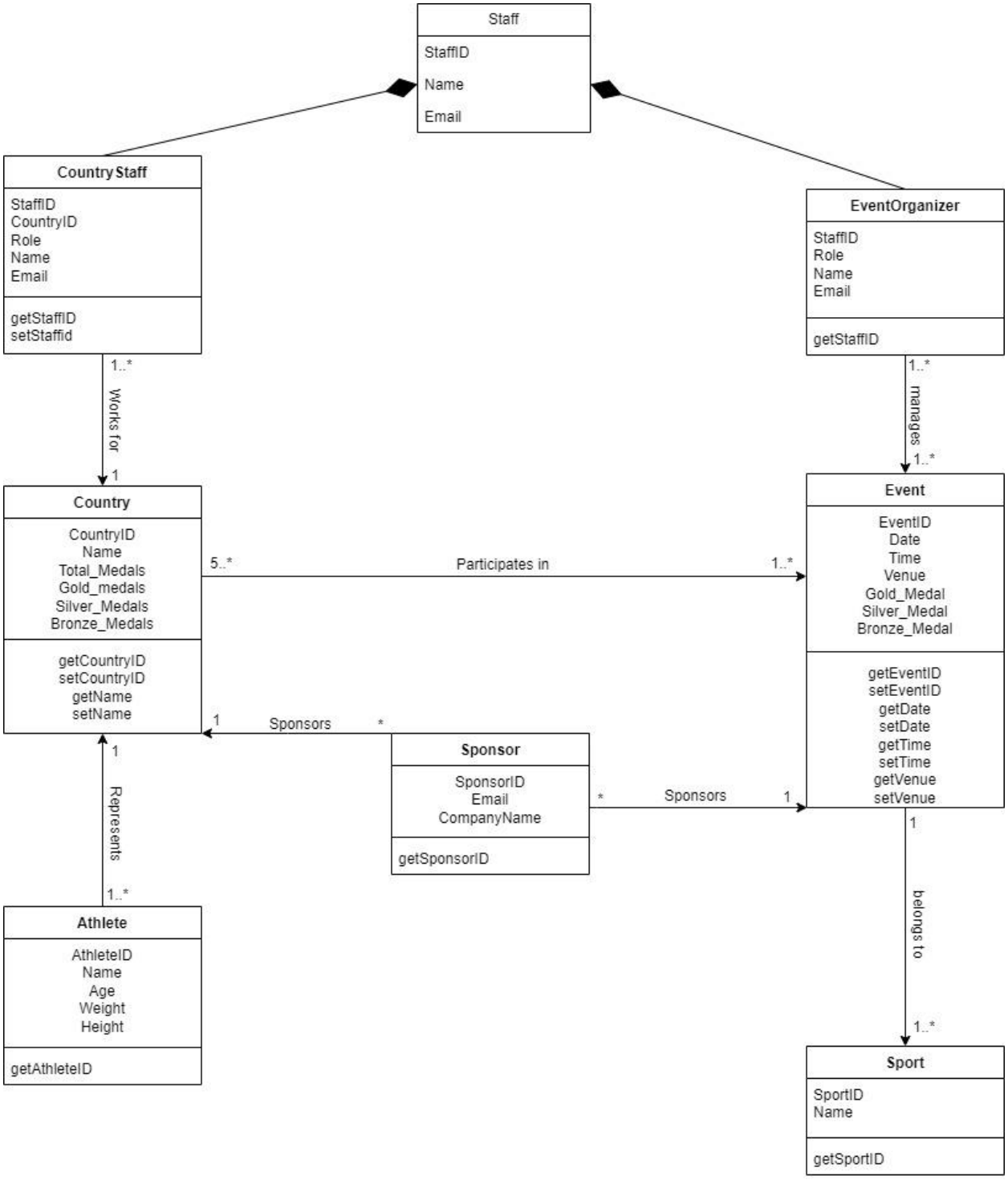
Requirements:

- Each country must have a minimum of one athlete.
- Every country is required to participate in at least one event.
- Sponsorship is optional for countries.
- Each athlete can only represent a single country.
- Each athlete must participate in at least one event.
- Each sport must host at least one event.
- Every event must be associated with a sport.
- Each event must feature a minimum of five participating countries/athletes.
- Every event must secure at least one sponsor.
- Each sponsor must support at least one event or one country.

ER Diagram


Staff
Staff_ID
Name
Role
Email
t
d
Country Staff
Staff_ID
Name
Role
Email
Country_ID
1..N
Work for
Event Organizers
Staff_ID
Name
Email
Role
1..M
manages
Country
Country_ID
Country_Name
Gold Medals
Silver Medals
Bronze Medals
Total Medals
1..1
2..N
participates in
1..M
Event
Event_ID
Date
Time
Venue
Gold Medal
Silver Medal
Bronze Medal
1..N
represents
sponsors
sponsors
belongs to
Athlete
Athlete_ID
Country_ID
Name
Age
Weight
Height
1..N
Sponsor
Sponsor_ID
Email
Company_Name
Sport
Sport_ID
Name
1..1

UML DIAGRAM


Staff
StaffID
Name
Email
Country Staff
StaffID
CountryID
Role
Name
Email
getStaffID
setStaffid
EventOrganizer
StaffID
Role
Name
Email
getStaffID
1..*
Works for
1
1..*
manages
1..*
Country
CountryID
Name
Total_Medals
Gold_medals
Silver_Medals
Bronze_Medals
getCountryID
setCountryID
getName
setName
5..*
Participates in
1..*
Event
EventID
Date
Time
Venue
Gold_Medal
Silver_Medal
Bronze_Medal
getEventID
setEventID
getDate
setDate
getTime
setTime
getVenue
setVenue
1
Sponsors
*
Sponsor
SponsorID
Email
CompanyName
getSponsorID
*
Sponsors
1
1
Represents
1..*
1
belongs to
1..*
Athlete
AthleteID
Name
Age
Weight
Height
getAthleteID
Sport
SportID
Name
getSportID

Relational Model:

i. Country (CountryID, Name, GoldMedals, SilverMedals, BronzeMedals, TotalMedals)

ii. Athlete (AthleteID, Name, Age, Weight, Height, CountryID)

- The relationship type represents is mapped by adding a foreign key CountryID referencing Country(CountryID)

iii. Event (EventID, Date, Time, Venue, GoldMedal, SilverMedal, BronzeMedal, SportID)

- The relationship type belongs to is mapped by adding a foreign key SportID referencing Sport(SportID)
- GoldMedal, SilverMedal, BronzeMedal are foreign keys referencing Country(CountryID)

iv. Sponsor (SponsorID, Name, Email, CountryID)

- The relationship type sponsors is mapped by adding a foreign key CountryID referencing Country(CountryID)

v. Sport (SportID, Name)

vi. Staff (StaffID, Name, Role, Email)

- The specialization is mapped by creating 3 relations, one for the superclass and one for each subclass

vii. Country Staff (StaffID, CountryID) - inherits all the attribute types from the entity type Staff

- The relationship type Work for is mapped by adding foreign key CountryID referencing Country(CountryID)
- The specialization is mapped by adding a foreign key StaffID referencing Staff(StaffID)

viii. Event Organizer (StaffID) - inherits all the attribute types from the entity type Staff

- The specialization is mapped by adding a foreign key StaffID referencing Staff(StaffID)

ix. Participates (CountryID, EventID)

- The relationship type participates is mapped by creating a new relation Participates and further adding two foreign keys CountryID, EventID referencing Country(CountryID) and Event(EventID) respectively which also acts as primary keys for the relation

x. Sponsors (SponsorID, EventID)

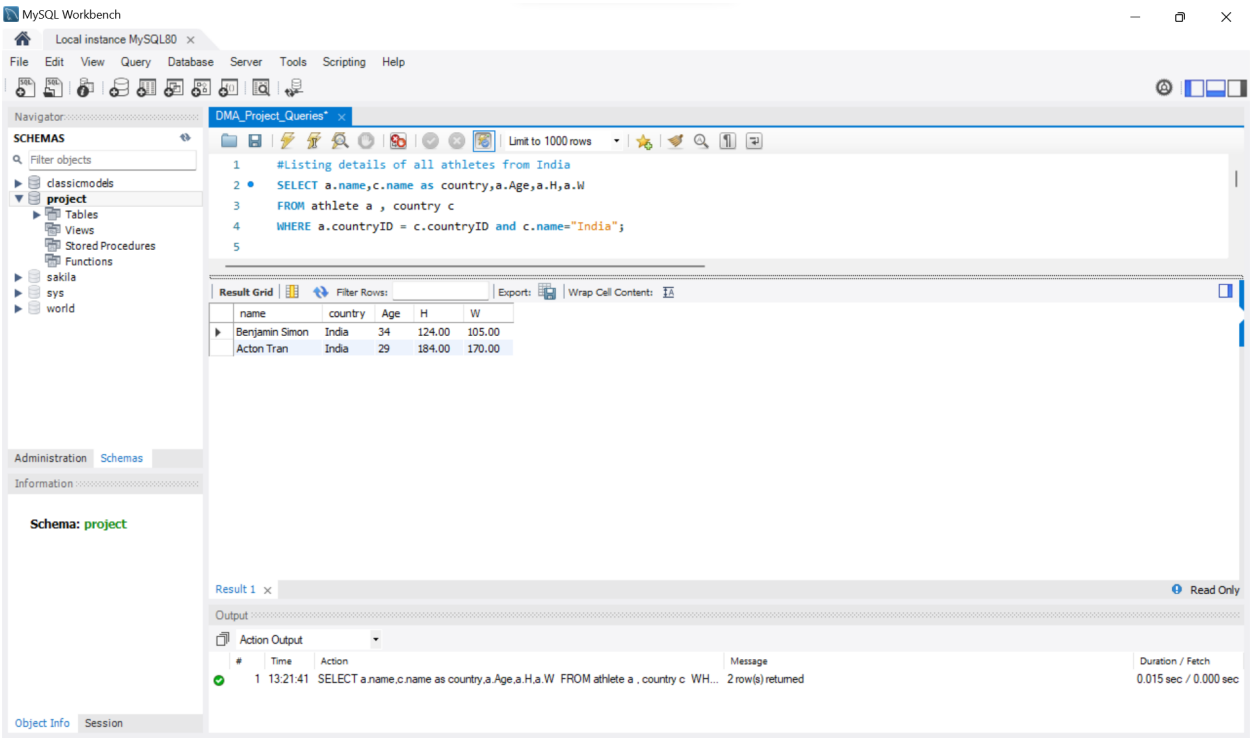
- The relationship type sponsors is mapped by creating a new relation Sponsors and further adding two foreign keys SponsorID, EventID referencing Sponsor(SponsorID) and Event(EventID) respectively which also acts as primary keys for the relation

xi. Manages (StaffID, EventID)

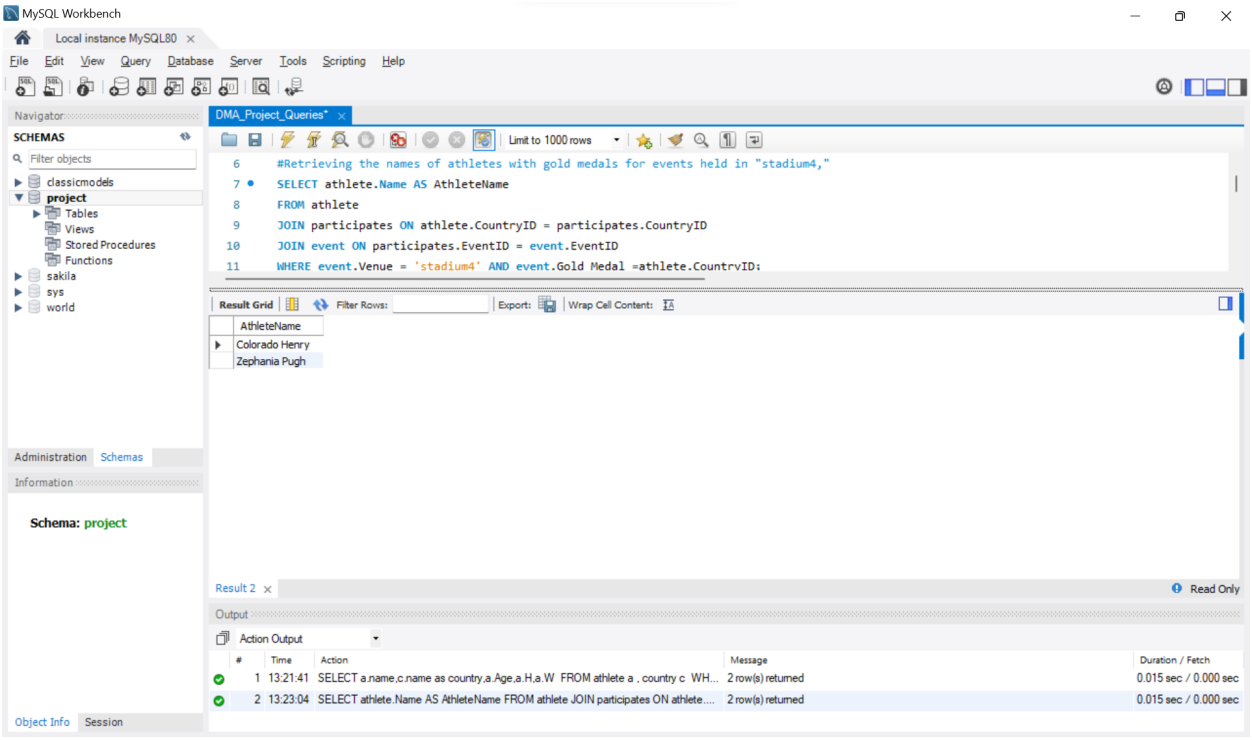
- The relationship type manages is mapped by creating a new relation Manages and further adding two foreign keys StaffID, EventID referencing Staff(StaffID) and Event(EventID) respectively which also acts as primary keys for the relation

MySQL Implementation:

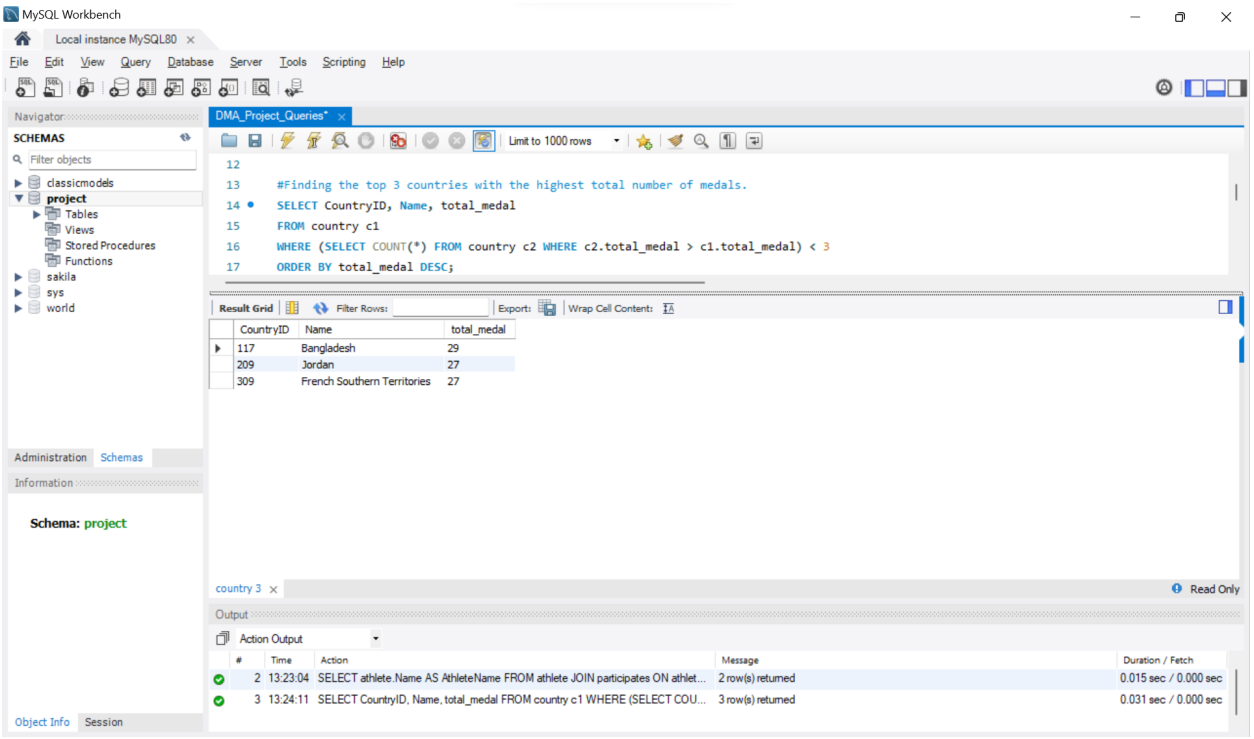
Query 1: #Listing details of all athletes from India



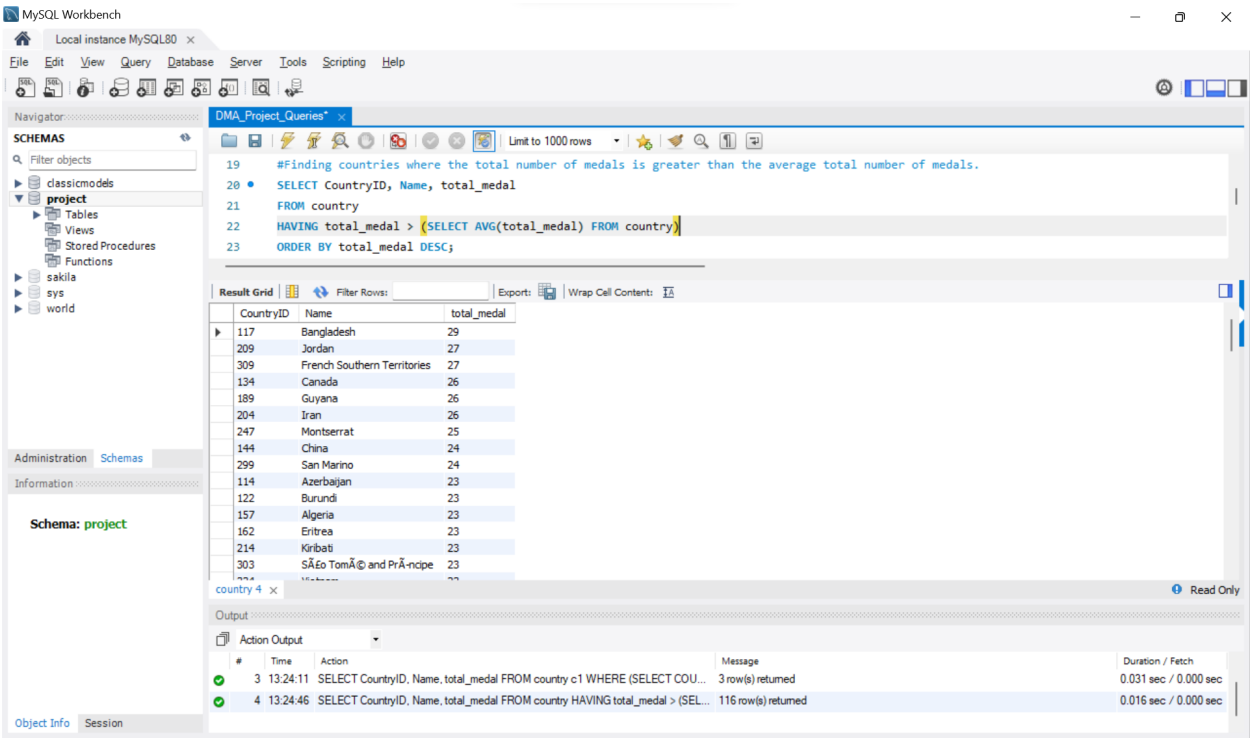
Query 2: #Retrieving the names of athletes with gold medals for events held in "stadium4,"



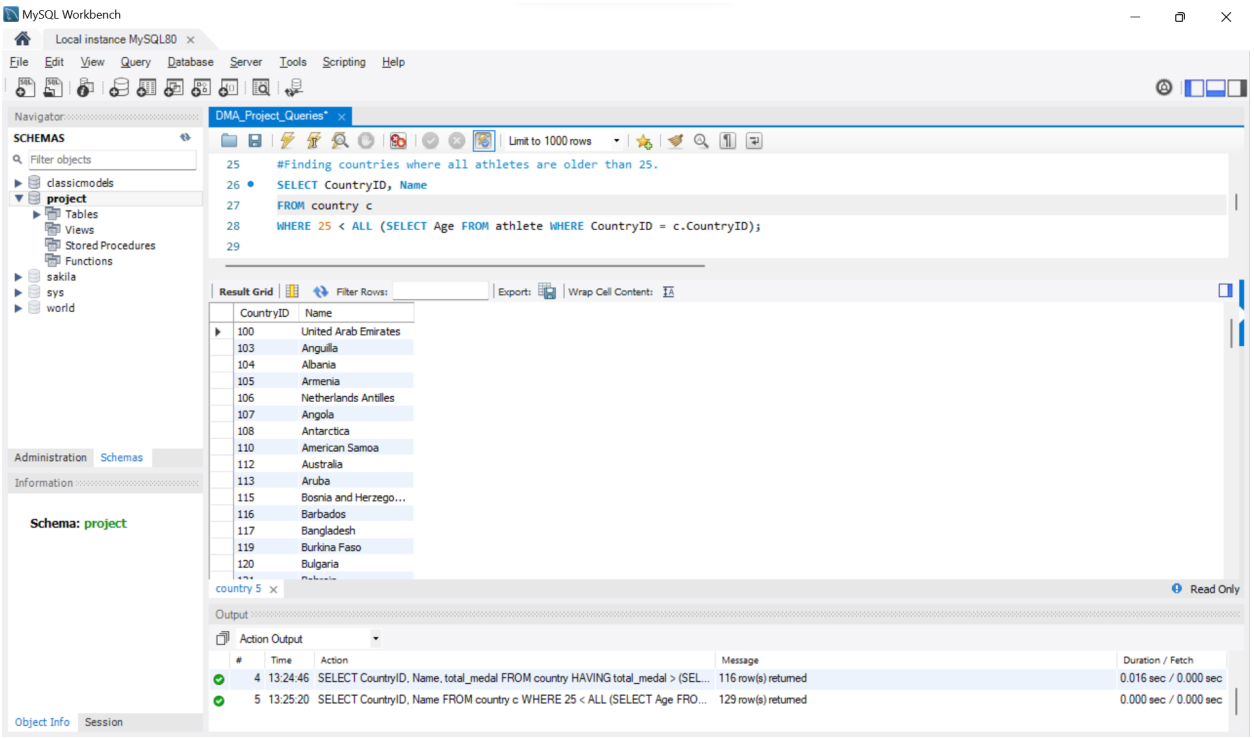
Query 3: #Finding the top 3 countries with the highest total number of medals.



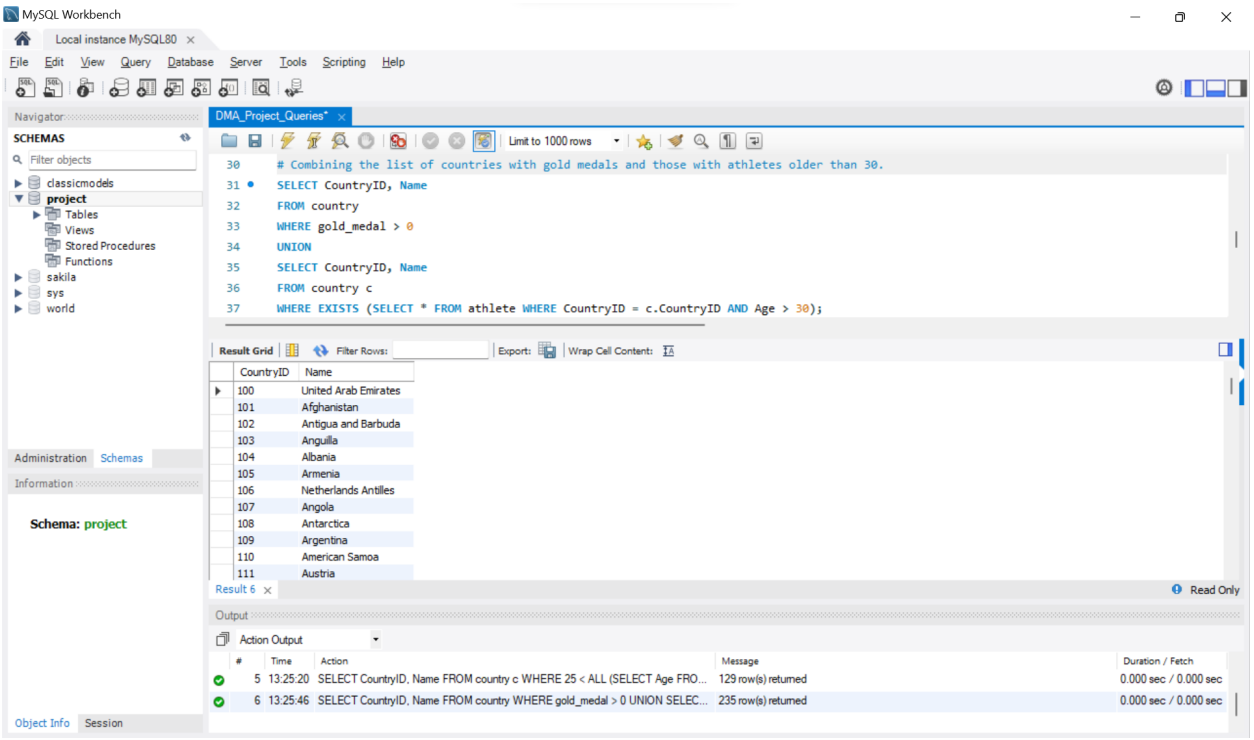
Query 4: #Finding countries where the total number of medals is greater than the average total number of medals.



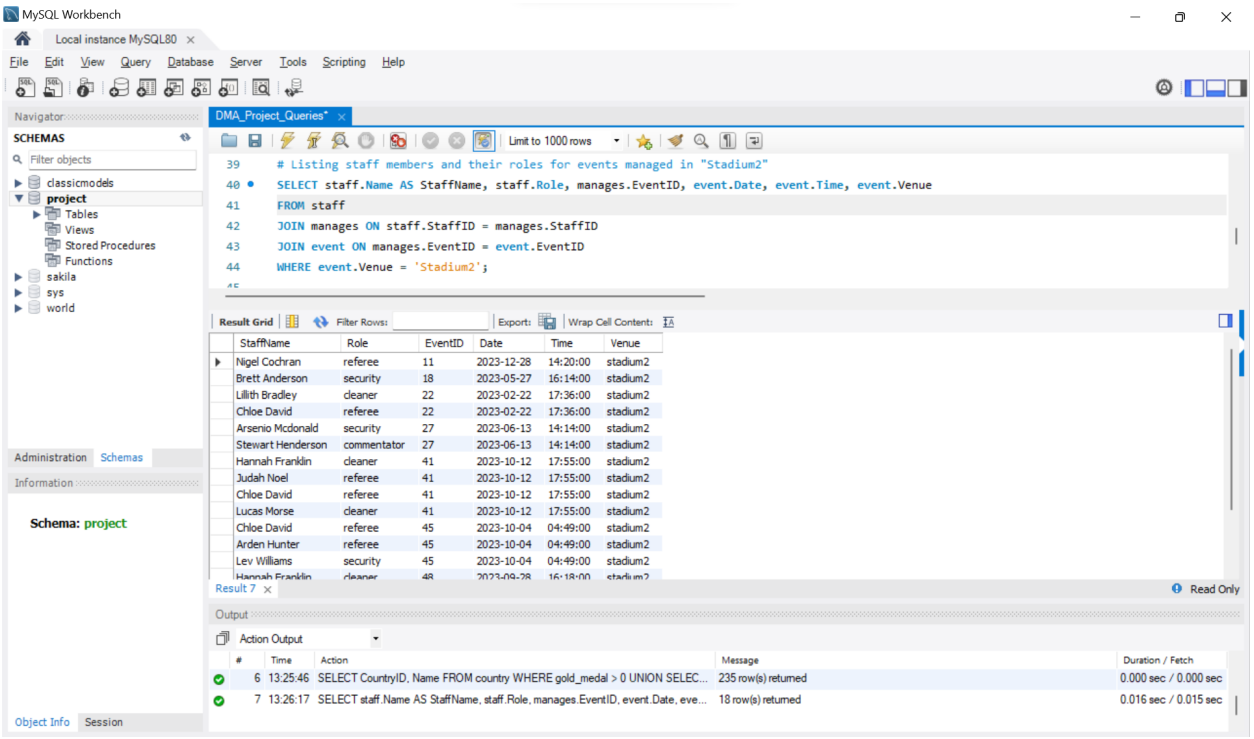
Query 5: #Finding countries where all athletes are older than 25.



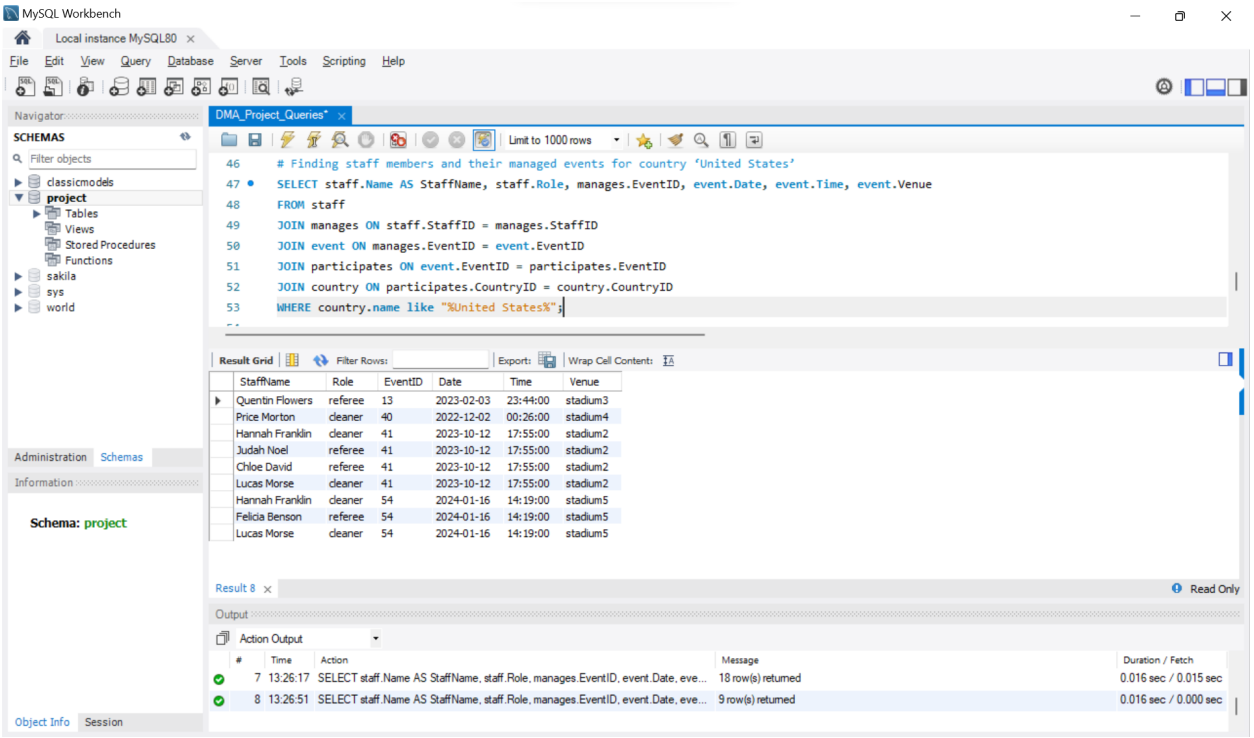
Query 6: # Combining the list of countries with gold medals and those with athletes older than 30.



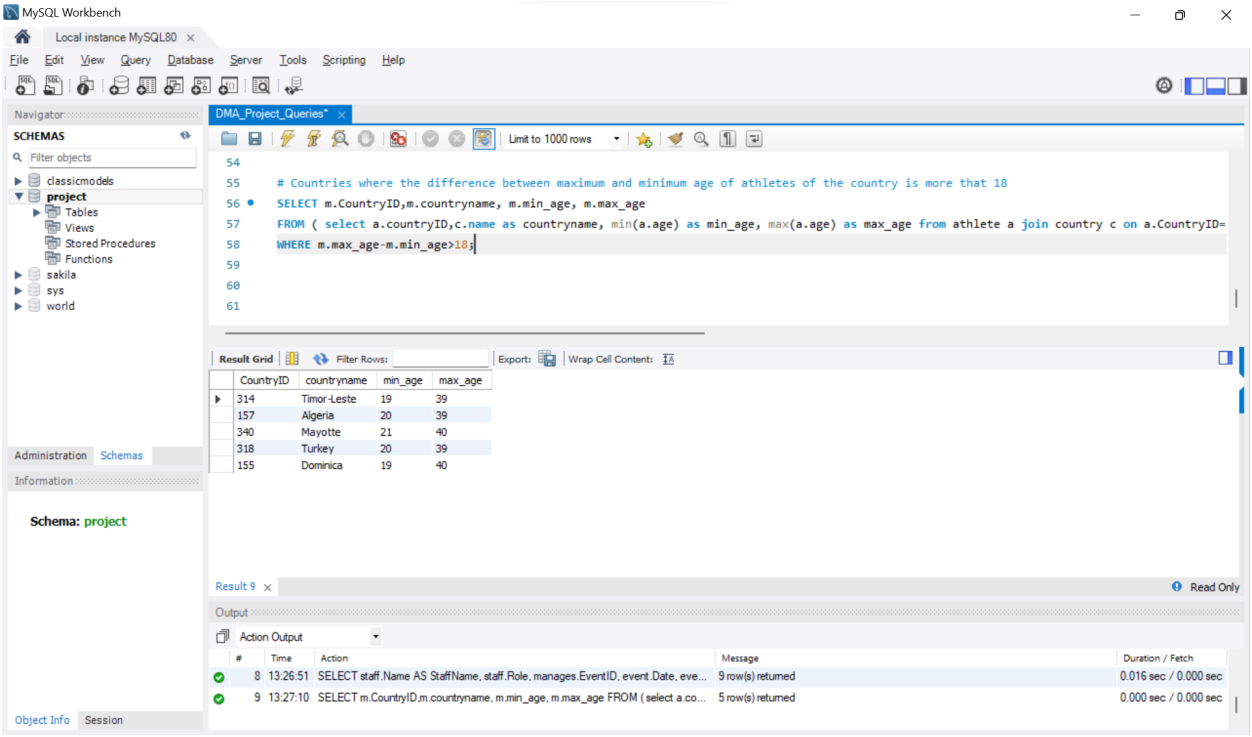
Query 7: # Listing staff members and their roles for events managed in "Stadium2"



Query 8: # Finding staff members and their managed events for country ‘United States’

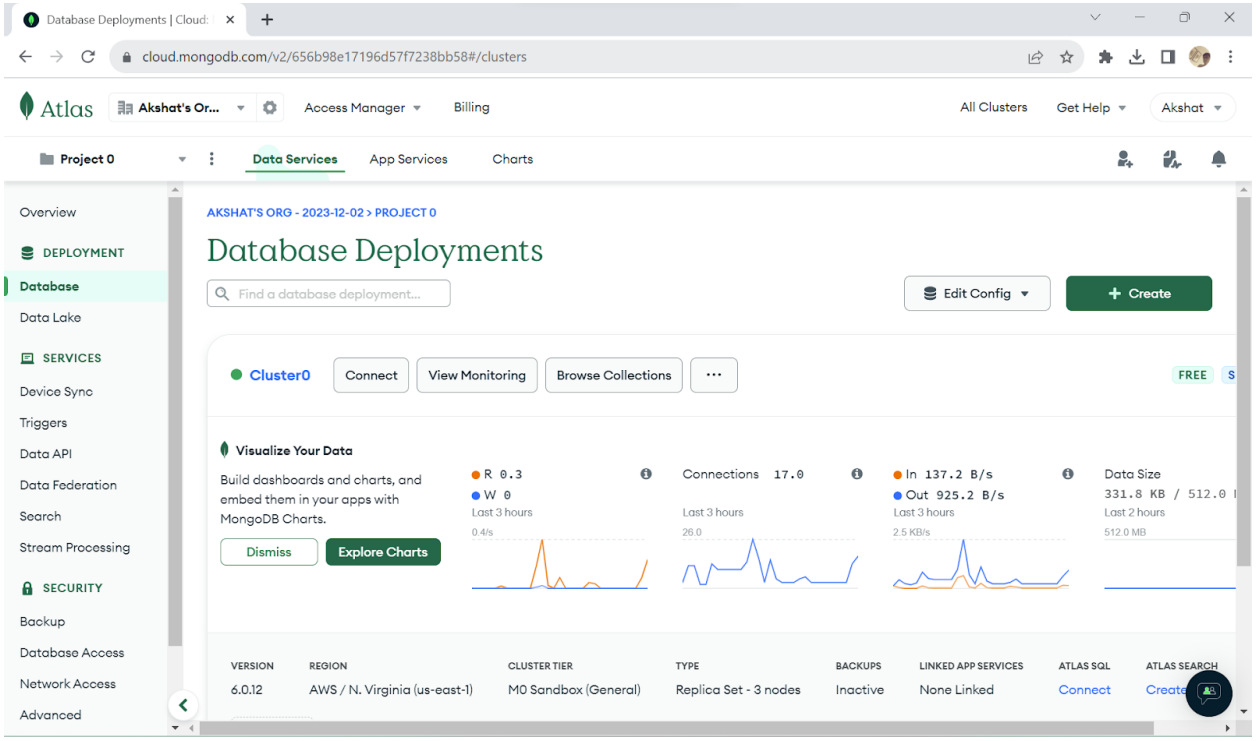


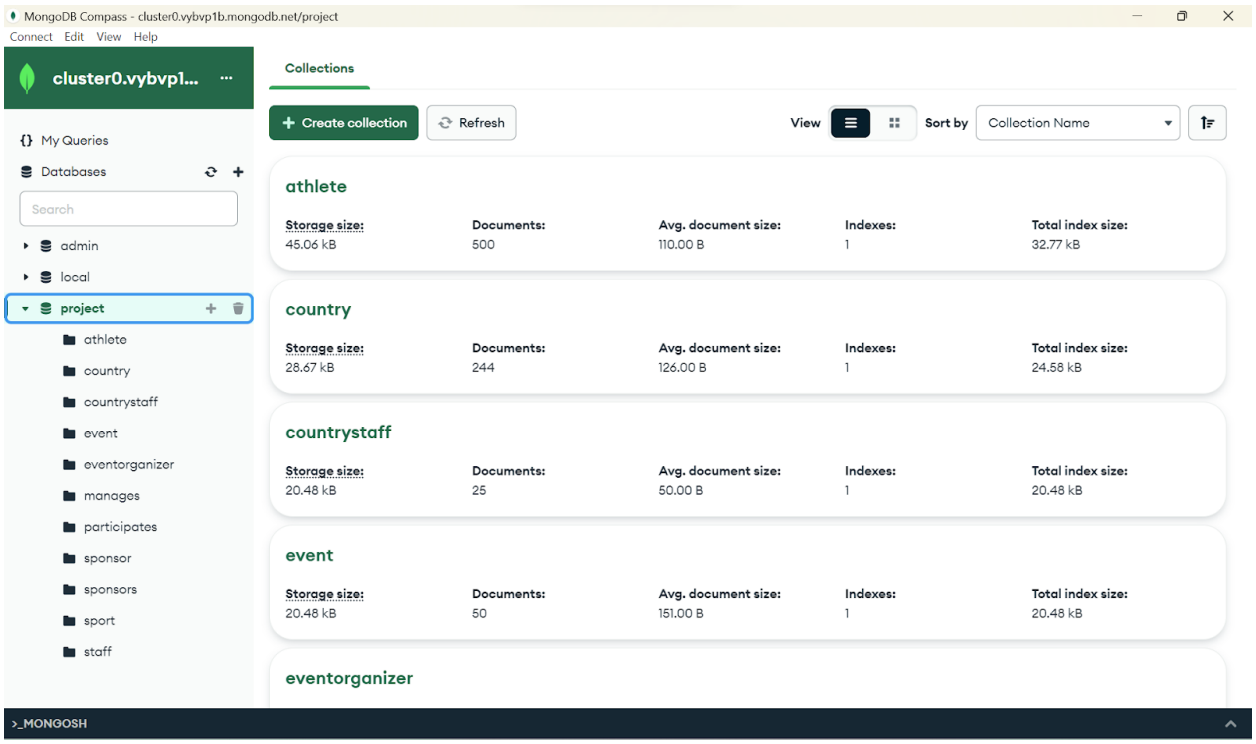
Query 9: # Countries where the difference between maximum and minimum age of athletes of the country is more that 18



NoSQL Implementation

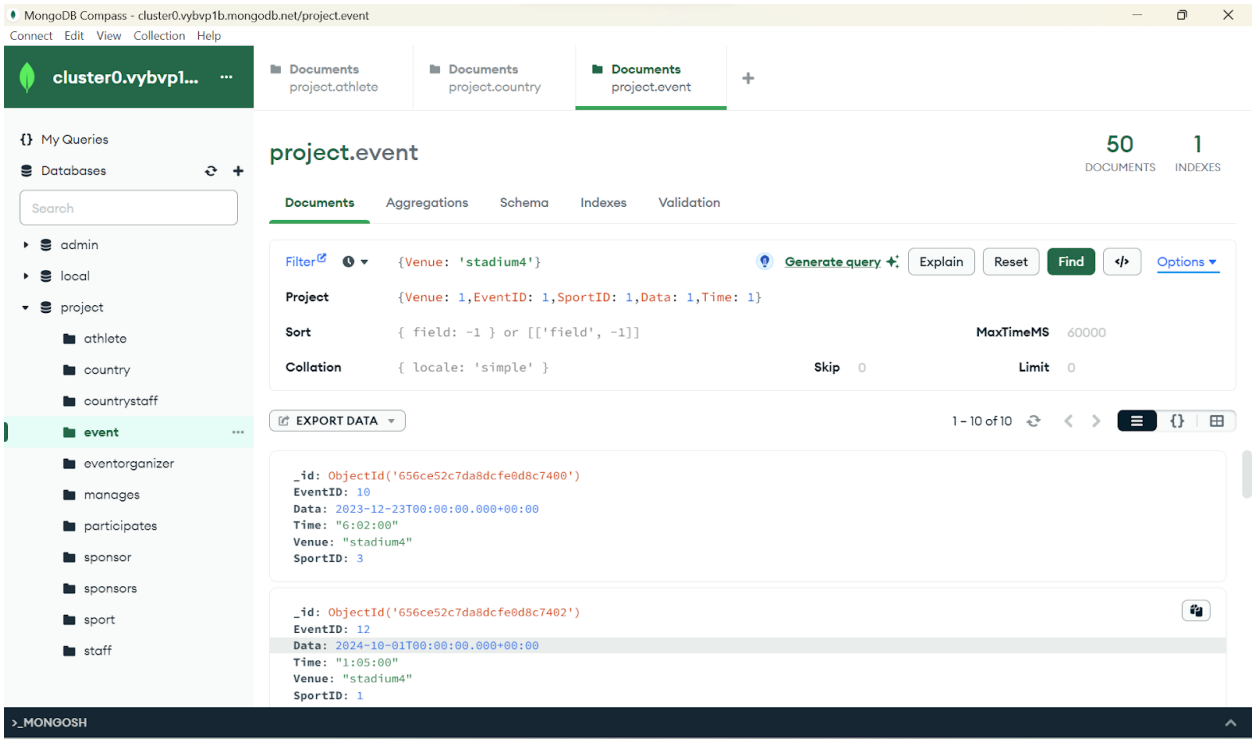
For NoSQL Implementation, we have implemented the database in MongoDB using the Atlas cloud service and used the MongoDB compass to host it on the local device


Database Deployments | Cloud:
cloud.mongodb.com/v2/656b98e17196d57f7238bb58#/clusters
Atlas
Akshat's Or...
Access Manager
Billing
All Clusters
Get Help
Akshat
Project 0
Data Services
App Services
Charts
Overview
DEPLOYMENT
Database
Data Lake
SERVICES
Device Sync
Triggers
Data API
Data Federation
Search
Stream Processing
SECURITY
Backup
Database Access
Network Access
Advanced
AKSHAT'S ORG - 2023-12-02 > PROJECT 0
Database Deployments
Find a database deployment...
Edit Config
Create
Cluster0
Connect
View Monitoring
Browse Collections
FREE
Visualize Your Data
Build dashboards and charts, and embed them in your apps with MongoDB Charts.
Dismiss
Explore Charts
R 0.3
W 0
Last 3 hours
Connections 17.0
Last 3 hours
In 137.2 B/s
Out 925.2 B/s
Last 3 hours
Data Size
331.8 KB / 512.0
Last 2 hours
VERSION
REGION
CLUSTER TIER
TYPE
BACKUPS
LINKED APP SERVICES
ATLAS SQL
ATLAS SEARCH
6.0.12
AWS / N. Virginia (us-east-1)
M0 Sandbox (General)
Replica Set - 3 nodes
Inactive
None Linked
Connect

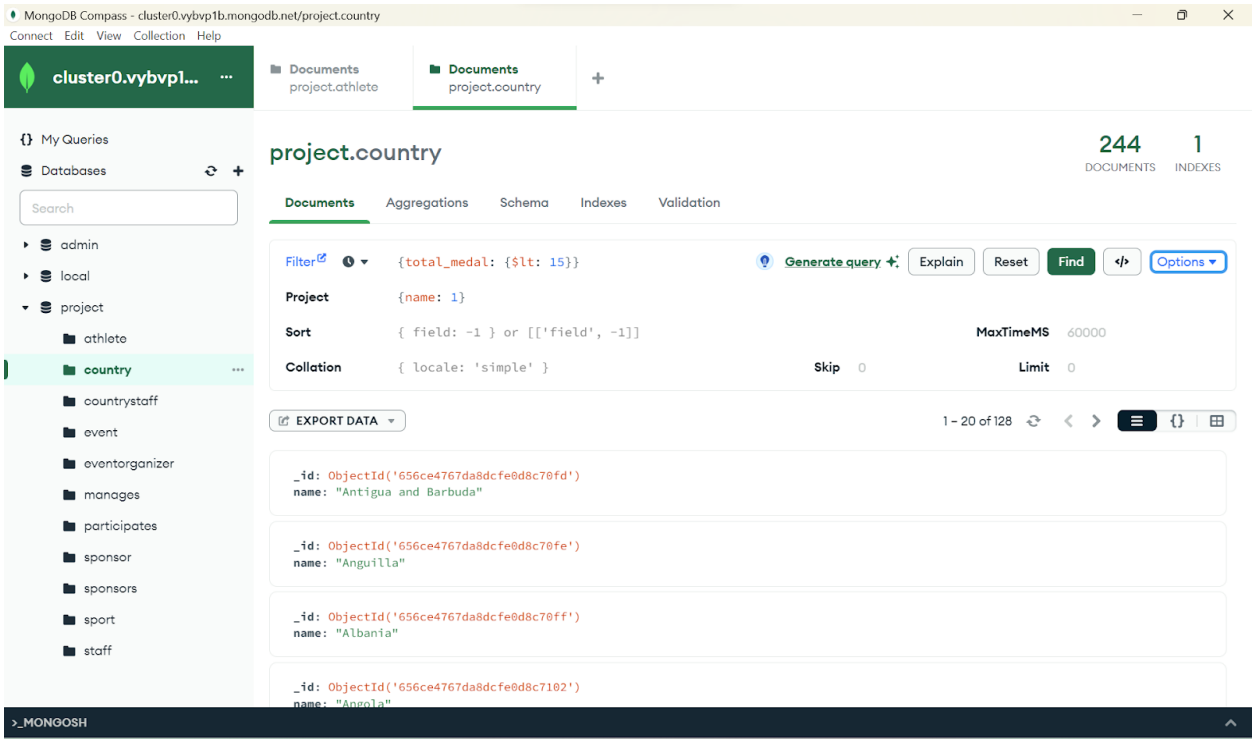

MongoDB Compass - cluster0.vybvp1b.mongodb.net/project
Connect Edit View Help
cluster0.vybvp1...
My Queries
Databases
Search
admin
local
project
athlete
country
countrystaff
event
eventorganizer
manages
participates
sponsor
sponsors
sport
staff
Collections
Create collection
Refresh
View
Sort by
Collection Name
athlete
Storage size: 45.06 kB
Documents: 500
Avg. document size: 110.00 B
Indexes: 1
Total index size: 32.77 kB
country
Storage size: 28.67 kB
Documents: 244
Avg. document size: 126.00 B
Indexes: 1
Total index size: 24.58 kB
countrystaff
Storage size: 20.48 kB
Documents: 25
Avg. document size: 50.00 B
Indexes: 1
Total index size: 20.48 kB
event
Storage size: 20.48 kB
Documents: 50
Avg. document size: 151.00 B
Indexes: 1
Total index size: 20.48 kB
eventorganizer
>_MONGOSH

We have run 3 queries on the database after adding all the appropriate data:

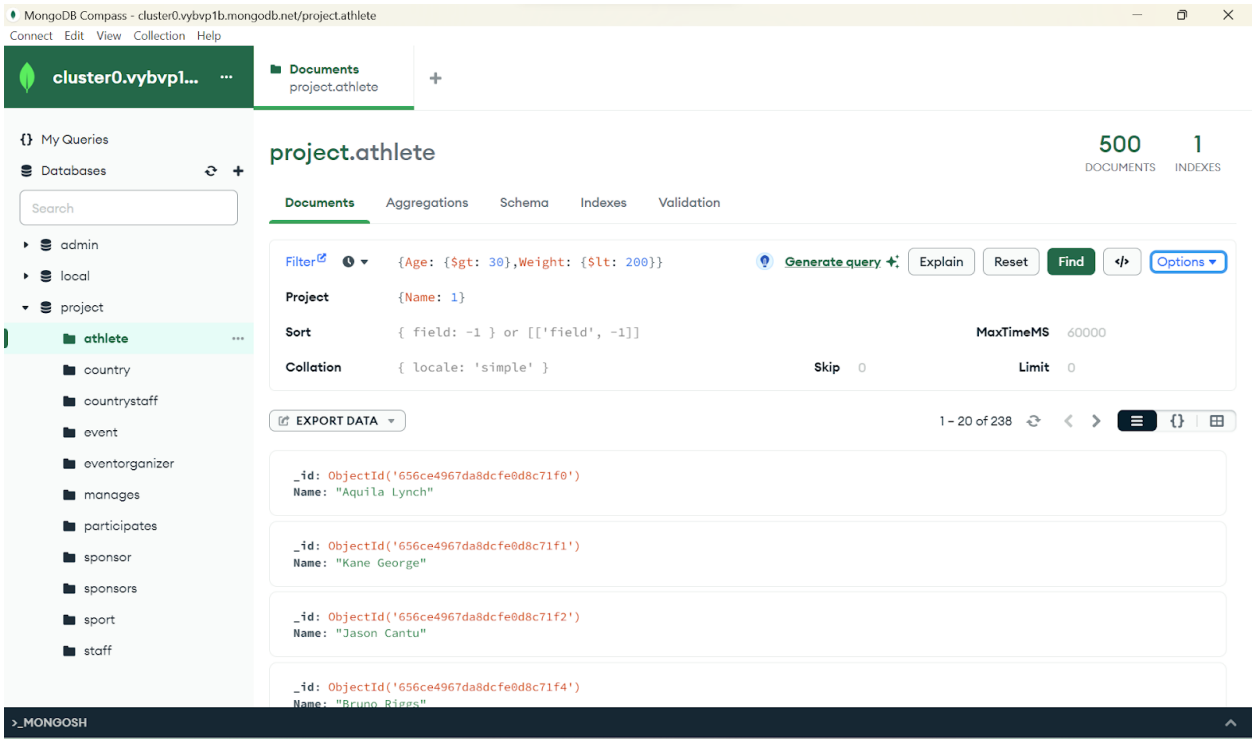
1) Query to find all events with the venue as stadium4



2) Query to find all countries with total no of medals less than 15



3) Query to find all athletes with age greater than 30 and weight less than 200



Python Implementation:

For the Python Application, we have opened the connection to the MySQL Server in Jupyter Notebook-(For easy Implementation)

We have run 3 queries to retrieve data and have plotted the data into Bar Graph and Pie chart

In [1]:
```
import mysql.connector
import os
import matplotlib.pyplot as plt
```

In [2]:
```
mydb=mysql.connector.connect(host='localhost',user='root',password='Mysqlkapas
```

In [3]:
```
mycursor=mydb.cursor()
```

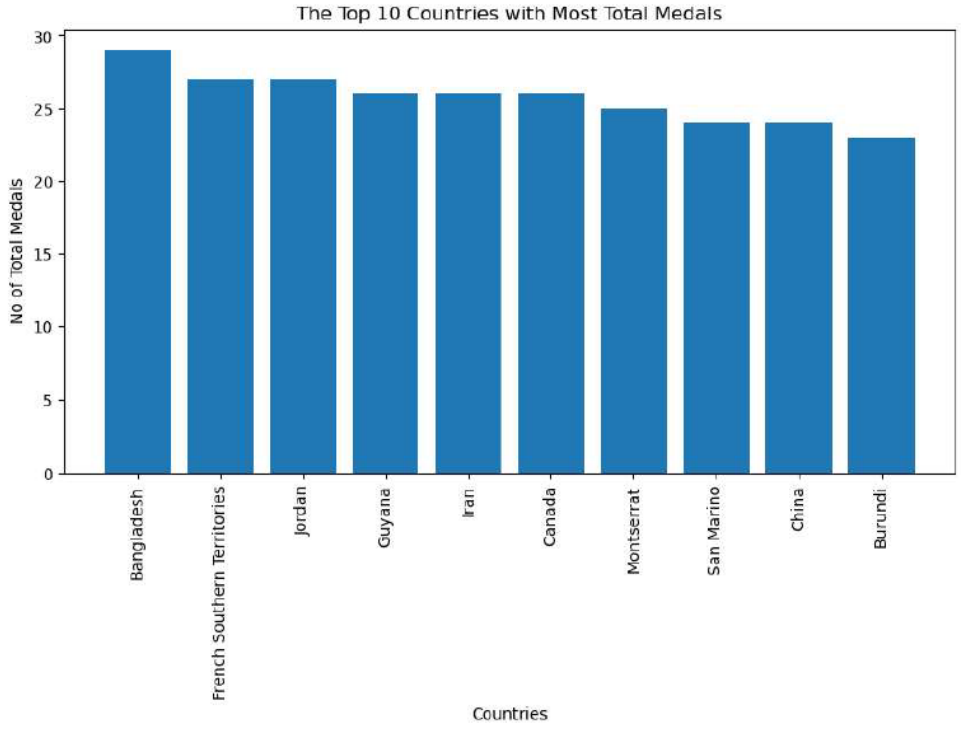
Query retrieves the top ten countries that have the highest no of total medals

In [4]:
```
mycursor.execute('SELECT CountryID, Name, total_medal FROM country ORDER BY to
```

In [5]:
```
r=mycursor.fetchall()
```

In [6]:
```
country=[]
total_medals=[]
for x in r:
    country.append(x[1])
    total_medals.append(x[2])
```

In [7]:
```
fig=plt.figure(figsize=(10,5))
plt.bar(country,total_medals)
plt.xlabel("Countries")
plt.ylabel("No of Total Medals")
plt.title("The Top 10 Countries with Most Total Medals")
plt.xticks(rotation=90)
plt.show()
```



In [8]:
```
mycursor.close()
```

Out[8]: True

In [9]:
```
mycursor=mydb.cursor()
```

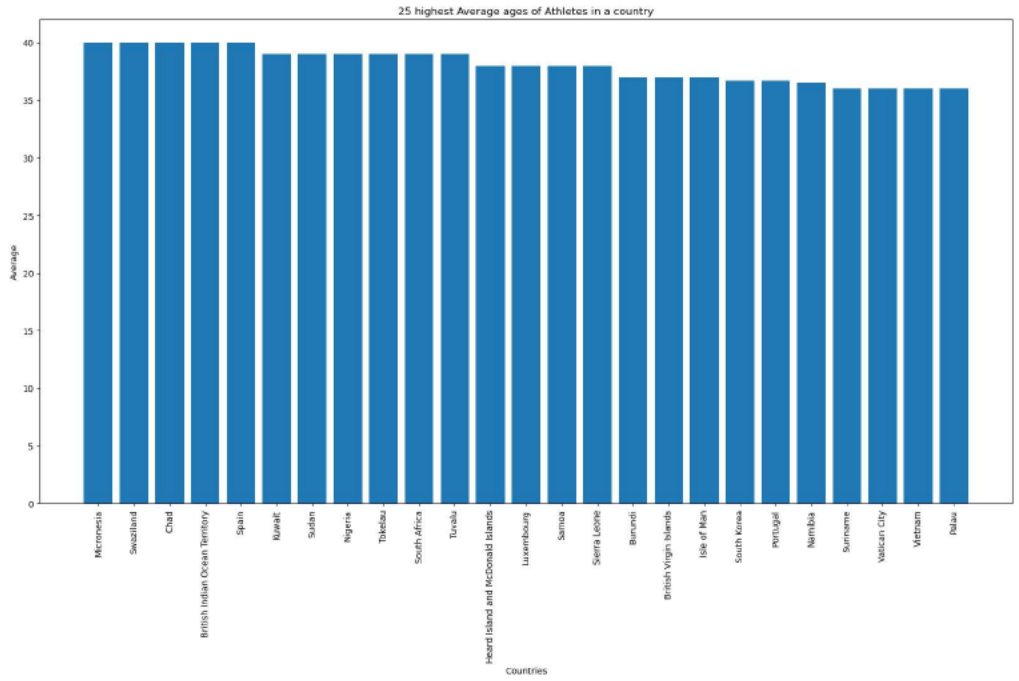
Query retries the Average age of the athletes of the countries with the 25 highest average ages

In [10]:
```
mycursor.execute('SELECT c.Name AS CountryName, AVG(a.Age) FROM athlete a,cour
```

In [11]:
```
r=mycursor.fetchall()
```

In [12]:
```
country=[]
avg_age=[]
for x in r:
    country.append(x[0])
    avg_age.append(x[1])
```

In [13]:
```
fig=plt.figure(figsize=(20,10))
plt.bar(country,avg_age)
plt.xlabel("Countries")
plt.ylabel("Average")
plt.title("25 highest Average ages of Athletes in a country")
plt.xticks(rotation=90)
plt.show()
```



In [14]:
```
mycursor.close()
```

Out[14]: True

In [15]:
```
mycursor=mydb.cursor()
```

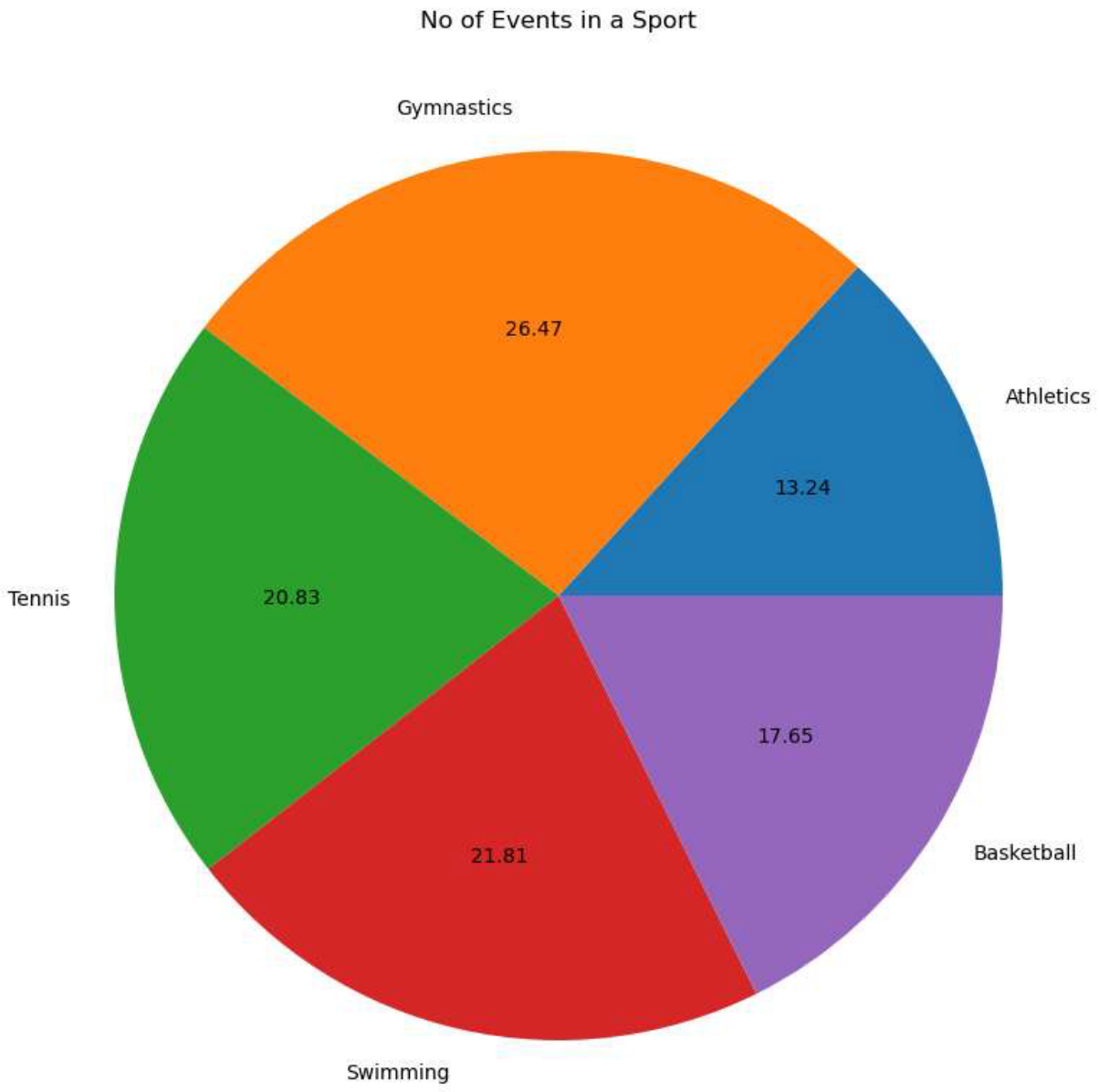
Query retrieves the Percentage of Events that belong to a particular sport

In [24]:
```
, Participates WHERE Sport.SportID = Event.SportID AND Event.EventID = Participates.EventID GROUP BY Sport.SportID, Sport.Name;')
```

In [25]:
```
r=mycursor.fetchall()
```

In [26]:
```
sport=[]
no_event=[]
for x in r:
    sport.append(x[0])
    no_event.append(x[1])
```

In [27]:
```
fig=plt.figure(figsize=(20,10))
plt.pie(no_event,labels=sport,autopct="%.2f")
plt.title("No of Events in a Sport")
plt.show()
```



```
In [28]: mycursor.close()

Out[28]: True

In [29]: mydb.close()
```

As the queries are not clear in the Screenshots, we have included them below

1) SELECT CountryID, Name, total_medal FROM country ORDER BY total_medal DESC LIMIT 10;

2) SELECT c.Name AS CountryName, AVG(a.Age) FROM athlete a,country c WHERE a.CountryID = c.CountryID GROUP BY c.CountryID, c.Name order by avg(a.age) desc limit 25;

3) SELECT Sport.Name AS SportName, COUNT(DISTINCT Participates.CountryID) AS NumberOfCountries FROM Sport, Event, Participates WHERE Sport.SportID = Event.SportID AND Event.EventID = Participates.EventID GROUP BY Sport.SportID, Sport.Name;