



Faculty of Computing &
Information Technology

Olympics Database System

20.08.2024

Maheen Fatima - BITF22M031

Inam Ul Haq - BITF22M017

Rabia Sadiq - BITF22M037

Instructor: Dr. Asif Sohail

Database Systems CC-215 Project

4th Semester - Spring 2024



Olympics:

The Olympic Games are the world's leading international sporting events. They feature sports competitions in which thousands of athletes from around the world participate in a variety of competitions.

We are constructing an Olympics database for our project. We mainly identified the following entities and built our database around them.

1. **Athlete**
2. **Team**
3. **Country**
4. **Sport**
5. **Game**
6. **Venue**
7. **Medal**
8. **Scores**
9. **Schedule**

These are the most basic things that constitute any Olympics game any year.

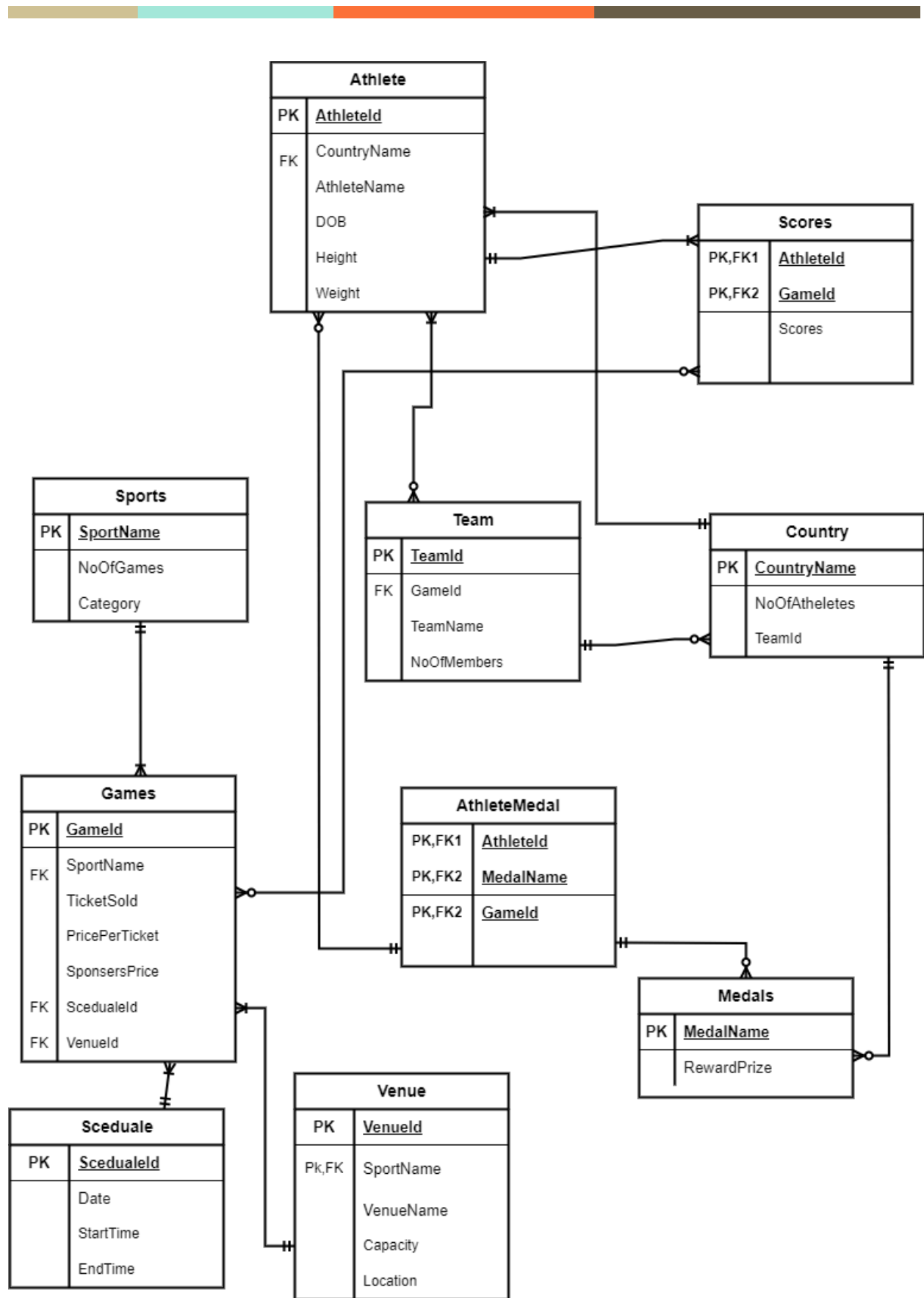
Olympics every year constitute several **sports**, with each sport having several **games** involving either individual **athletes** or **teams**. Athletes or teams belong to **countries** and each athlete or team may or may not win a **medal**. A particular game is played on a particular **venue** according to a particular **Schedule** associated with a particular sport.

This is the crux of our system. We mainly focused on what essentially constitutes any Olympics any year and built a database project around it. All the requirements demanded in the Project Template document being sent to us have been duly fulfilled.

Entity Relationship Diagram:



Below is the entity-relationship diagram for the project.





Below are the key points of the ERD diagram.

- A country may have no athletes (if it only participates in team sports) and may have no teams (if it only participates in individual sports).
- A team is treated as a single entity, and the individual members are not separately identified. Athletes, on the other hand, are represented as a distinct entity for those participating in individual events.
- Both teams and athletes may not win any medals, or at most, can earn a single medal.
- Each team and athlete must be associated with a specific sport, but a sport will only involve either teams or athletes, depending on its category.
- Each game is held at a designated venue, which is an entity representing the location of the event.
- Every game is scheduled according to a specific timeline, with the schedule being an entity that details the timing and sequence of events.

Some entities, such as the Leaderboard, may not be developed yet; they will be created using views to optimize system efficiency.

Construction of the Relational Schema:


The relational schema using the top-down and bottom-up approach is discussed below.

Bottom-Up Approach:

Firstly, let's have a look at the table below:

Unnormalized Form:

We have identified all the attributes. Now, if we take a quick look, we can already see several anomalies e.g. Insert, Update, and Delete anomalies.



Olympics (CountryName, NoOfAthletes, AthleteID, MedalName, AthleteName, SportsName, Height, Weight, DOB, TeamID, TeamName, NoOfMembers, Category, NoOfGames, GameID, VenueName, Capacity, Location, TicketsSold, PricePerTicket, SponsorsPrice, ScheduleID, VenueID, Date, StartTime, EndTime, MedalName, RewardPrize, Scores, ScoreType)

Insert Anomalies:

- We have to insert the country name, as many times as it has athletes and teams which is quite hectic and difficult if we have several countries and athletes.
- Suppose, I add a new game venue. It's a must to add all other redundant values e.g. (NoOfMembers) etc which have no role with the game venue but it's illogically necessary in this relation.

Update Anomalies:

- If an athlete's details (e.g. weight etc) change, we would need to update this information in multiple records. If one record is missed, the data becomes inconsistent.
- If the number of members in a team changes, every occurrence of that team in different rows must be updated. Failing to do so would result in inconsistent information across rows.

Insert Anomalies:

- If we delete an athlete who is the only participant from a particular country and that country doesn't have any other team or athlete, we may lose all information related to that country.
- If we delete a game record, we could lose information about the venue, sport, and possibly even the team or athlete if that game was their only game in the Olympics.

Bottom-Up Approach:

1NF:

To convert a table to 1NF, it must satisfy the following conditions:

- The table must have a primary key.
- Each cell contains only atomic (indivisible) values.
- There are no repeating groups of columns.

Since the table is already in 1NF so the relation will same:

Olympics (**CountryName**, **AthleteID**, **GameID**, SportsName, NoOfAthletes, MedalName, AthleteName, Height, Weight, DOB, TeamID, TeamName, NoOfMembers, Category, NoOfGames, VenueName, Capacity, Location, TicketsSold, PricePerTicket, SponsorsPrice, ScheduleID, VenueID, Date, StartTime, EndTime, MedalName, RewardPrize, Scores, ScoreType)

2NF:

To convert a table to 2NF, it must satisfy the following conditions:

- The table is in 1NF.
- All non-key attributes are fully functionally dependent on the primary key (no partial dependency).

Country(**CountryName**, NoOfAtheletes, TeamID, TeamName, NoOfMembers)



Athlete(**AthleteID**,AthleteName,Height, Weight, DOB,MedalName,rewardPrize)

Game(**GameID**,SportName,Category,NoOfGames,VenueName, Capacity, Location, TicketsSold, PricePerTicket, SponsorsPrice, ScheduleId, VenueId, Date, StartTime, EndTime)

Medal(**CountryName**, **AthleteID**,MedalName, RewardPrize)

Scores(**AthleteID**,**GameID**,Scores)

3NF:

3NF requires:

- No transitive dependencies, where non-key attributes depend on other non-key attributes.

To achieve 3NF:

- Further break down tables to remove transitive dependencies.

Athlete(**AthleteID**,AthleteName,Height, Weight, DOB,MedalName)

Scores (**AthleteID**,**GameID**,Scores)

Country(**CountryName**,NoOfAtheletes,TeamId)

Medal(**MedalName**,RewardPrize)

Team(**TeamID**, TeamName, NoOfMembers)

Game(**GameID**,SportName, TicketsSold, PricePerTicket, SponsorsPrice, ScheduleId, VenueId)



Sport(**SportName**,Category,NoOfGames)

Venue(**VenueId**,VenueName, Capacity, Location)

Schedule(**ScheduleId**,Date, StartTime, EndTime)

Top-Down Approach:

Country Table:

- **Attributes:** CountryName (Primary Key), NoOfAthletes, TeamId
- **Description:** This table holds information about the countries participating in the Olympics.


Athlete Table:

- **Attributes:** AthleteId (Primary Key), CountryName (Foreign Key), AthleteName, Height, Weight, DOB
- **Description:** This table holds information about athletes, including their sport and country.

Sport Table:

- **Attributes:** SportName (Primary Key), Category, NoOfGames
- **Description:** This table holds information about the different sports in the Olympics.

Team Table

- 
- **Attributes:** TeamID (Primary Key), GameID(Foreign Key), TeamName, NoOfMembers
 - **Description:** This table contains team information, including their associated sport and country.

Game Table:

- **Attributes:** GameID (Primary Key), SportName (Foreign Key), VenueID (Foreign Key), TicketSold, PricePerTicket, SponsorPrice, ScedualeID
- **Description:** This table holds information about games, including the sport, venue, and ticket details.

Venue Table:

- **Attributes:** VenueID (Primary Key), SportName (Foreign Key), VenueName, Capacity, Location
- **Description:** This table holds information about venues, including their capacity and the sports they host.


Medal Table:

- **Attributes:** MedalName (Primary Key), RewardPrize
- **Description:** This table holds information about medals awarded to athletes or teams.

Schedule Table:

- **Attributes:** ScedualeID (Primary Key), SportName(Foreign Key), Date, StartTime, EndTime
- **Description:** This table holds information about schedule, including their Date and time

AthleteTeam Table:

- 
- **Attributes:** Athleteld (Primary Key,Foreign Key),TeamId(Primary key,Foreign Key)
 - **Description:**This table holds information about representing the athlete participating in the team and the team the athlete is part of.

Score Table:

- **Attributes:** Gameld (Primary Key,Foreign Key) Athleteld (Primary Key,Foreign Key),Scores
- **Description:**This table holds information about representing a game can have multiple scores (e.g., scores for different athletes), and a score can be associated with multiple games (e.g., if a score is tied).

AthleteMedal Table:

- **Attributes:** Athleteld (Primary Key,Foreign Key),MedalId(Primary key,Foreign Key),Gameld(Primary key,Foreign Key)
- **Description:**This table holds information about representing the athlete participating in the team and the team the athlete is part of.

Normalization:

- **Country Table:** Already normalized as it is in 3NF.
- **Athlete Table:** Each attribute is dependent on the primary key, and there are no transitive dependencies.
- **Sport Table:** No redundant data, all attributes are dependent on the primary key.
- **Team Table:** Properly normalized with no partial or transitive dependencies.
- **Games Table:** All attributes are directly related to the GameID.
- **Venue Table:** All attributes are directly related to the VenueName.

- **Medals Table:** Normalized to ensure that each attribute depends on the primary key
- **Schedule Table:** Normalized to ensure that each attribute depends on the primary key
- **AthleteMedal Table:** Normalized to ensure that there is not any non key attribute.
- **GameScore Table:** Normalized to ensure that there is not any non key attribute.

Country

<u>CountryName</u>	NoOfAthletes	TeamId
--------------------	--------------	--------

Athlete

<u>AthleteID</u>	CountryName	AthleteName	DOB	Height	Weight
------------------	-------------	-------------	-----	--------	--------

Sport

<u>SportName</u>	Category	NoOfGames
------------------	----------	-----------

Team



<u>TeamID</u>	GameId	TeamName	NoOfMembers
---------------	--------	----------	-------------

Games

<u>GameID</u>	SportName	Venueld	TicketsSold	PricePerTicket	SponsorsPrice	ScedualeId
---------------	-----------	---------	-------------	----------------	---------------	------------

Venue

<u>Venueld</u>	SportName	VenueName	Capacity	Location
----------------	-----------	-----------	----------	----------

Medal

<u>MedalName</u>	RewardPrize
------------------	-------------

Schedule

<u>ScheduleId</u>	SportName	Date	StartTime	EndTime
-------------------	-----------	------	-----------	---------

AthleteTeam

<u>Athleteld</u>	<u>TeamId</u>
------------------	---------------

AthleteMedal

<u>Athleteld</u>	<u>MedalName</u>	<u>Gameld</u>
------------------	------------------	---------------

Score

<u>Gameld</u>	<u>Athleteld</u>	Scores
---------------	------------------	--------


DESCRIPTION OF the Relations:

COUNTRY:

Attributes	DataType	Size	Constraints
CountryName	VARCHAR	(100)	PRIMARY KEY
NoOfAthletes	number		
TeamId	number		FOREIGN KEY REFERENCES Team

Sport:

Attributes	DataType	Size	Constraints
------------	----------	------	-------------



SportName	VARCHAR	(50)	PRIMARY KEY
Category	VARCHAR2	(50)	
NoOfGames	number		CHECK(Category IN ('Individual', 'Team'))

Medal:

Attributes	Data Type	Size	Constraints
MedalName	VARCHAR2(50)	(50)	PRIMARY KEY CHECK(MedalName IN ('Gold', 'Silver', 'Bronze'))
NoOfAthletes	number		
RewardPrize	NUMBER	(10, 2)	FOREIGN KEY REFERENCES Team

Team:

Attributes	Data Type	Size	Constraints
TeamID	NUMBER		PRIMARY KEY
GameID	number		FOREIGN KEY REFERENCES Game(GameID) ON DELETE SET NULL
TeamName	VARCHAR2(100)		NOT NULL
NoOfMembers	NUMBER		

AthleteID INT PRIMARY KEY,
 AthleteName VARCHAR(100),
 Height DECIMAL(5, 2),
 Weight DECIMAL(5, 2),
 DOB DATE,
 CountryName VARCHAR(100),
 FOREIGN KEY (CountryName) REFERENCES Country(CountryName),

Athlete:

Attributes	Data Type	Size	Constraints
AthleteID	INT		PRIMARY KEY
Height	DECIMAL(5, 2),		




Weight	DECIMAL(5, 2),		
DOB	Date		
CountryName	VARCHAR	(100)	FOREIGN KEY (CountryName) REFERENCES Country(CountryName) ,

Venue:

Attributes	Data Type	Size	Constraints
VenueID	INT		PRIMARY KEY
VenueName	VARCHAR	(100),	
Capacity	INT		
Location	VARCHAR	(100)	

Schedule:



Attributes	Data Type	Size	Constraints
ScheduleId	INT		PRIMARY KEY
SDate	DATE		
Start_Time	VARCHAR2	(8)	not null
End_Time	VARCHAR2	(8)	not null

Game:

Attributes	Data Type	Size	Constraints
GameID	NUMBER		PRIMARY KEY
SportName	VARCHAR2	(50)	NOT NULL, FOREIGN KEY (SportName) REFERENCES Sport(SportName) ON DELETE CASCADE

Venueld	Number	NOT NULL,	not null,FOREIGN KEY (Venueld) REFERENCES Venue(Venueld) ON DELETE CASCADE,
TicketsSold	NUMBER		
SponsorsPrice	NUMBER(10, 2),		
PricePerTicket	NUMBER(10, 2)		
ScheduleId	NUMBER		FOREIGN KEY (ScheduleId) REFERENCES schedule(ScheduleId) ON DELETE CASCADE

Athleteld NUMBER,
 GameNo NUMBER,
 Score NUMBER,
 PRIMARY KEY (Athleteld, GameNo),
 FOREIGN KEY (Athleteld) REFERENCES Athlete(Athleteld),
 FOREIGN KEY (GameNo) REFERENCES Game(GameNo)

Scores:



Attributes	Data Type	Size	Constraints
Athleteld	NUMBER		PRIMARY KEY FOREIGN KEY (Athleteld) REFERENCES Athlete(Athleteld),
GameNo	NUMBER	(50)	PRIMARY KEY FOREIGN KEY (GameNo) REFERENCES Game(GameNo)
Score	Number		

AthleteTeam:

Attributes	Data Type	Size	Constraints
Athleteld	NUMBER		PRIMARY KEY FOREIGN KEY (Athleteld) REFERENCES Athlete(Athleteld),
TeamId	NUMBER		PRIMARY KEY



			FOREIGN KEY (TeamId) REFERENCES Team(TeamId)
--	--	--	--

AthleteMedal:

Attributes	DataType	Size	Constraints
Athleteld	NUMBER		PRIMARY KEY FOREIGN KEY (Athleteld) REFERENCES Athlete(Athleteld),
Gameld	NUMBER		PRIMARY KEY FOREIGN KEY (Gameld) REFERENCES Game(Gameld)
MedalName	varchar		PRIMARY KEY FOREIGN KEY (MedalName) REFERENCES MEdal(MedalName)

Team:

Attributes	Data Type	Size	Constraints
TeamID	NUMBER		PRIMARY KEY
GameID	NUMBER		PRIMARY KEY FOREIGN KEY (GameID) REFERENCES Team(GameID) ON DELETE SET NULL
TeamName	VARCHAR2(100)	(100)	NOT NULL
NoOfMembers	NUMBER		

CREATE and DESCRIBE the Relations:

Below are the “CREATE TABLE ...” statements and the “DESCRIBE ...” statements describing different tables.


COUNTRY:

Below is the CREATE table statement for the COUNTRY table.

```
CREATE TABLE Country (
  CountryName VARCHAR(100) PRIMARY KEY,
  NoOfAthletes number,
  TeamId number,
  FOREIGN KEY (TeamId) REFERENCES Team(TeamId) ON DELETE SET NULL
)
```

SPORT:

Below is the CREATE table statement for the SPORT table.



```
CREATE TABLE Sport (  
    SportName VARCHAR2(50) PRIMARY KEY,  
    Category VARCHAR2(50),  
    NoOfGames NUMBER,  
    CONSTRAINT SPT_CHECK CHECK(Category IN ('Individual', 'Team'))  
)
```

MEDAL:

Below is the CREATE table statement for the MEDAL table.

```
CREATE TABLE Medal (  
    MedalName VARCHAR2(50) PRIMARY KEY,  
    RewardPrize NUMBER(10, 2),  
    CONSTRAINT MDL_CHECK CHECK(MedalName IN ('Gold', 'Silver', 'Bronze'))  
)
```

TEAM:

Below is the CREATE table statement for the TEAM table.

```
CREATE TABLE Teams (  
    TeamID NUMBER PRIMARY KEY,  
    GameID NUMBER,  
    TeamName VARCHAR2(100) NOT NULL,  
    NoOfMembers NUMBER,  
    FOREIGN KEY (GameID) REFERENCES Game(GameID) ON DELETE SET NULL,
```



)

ATHLETE:

Below is the CREATE table statement for the ATHLETE table.

```
CREATE TABLE Athlete (  
    AthleteID INT PRIMARY KEY,  
    AthleteName VARCHAR(100),  
    Height DECIMAL(5, 2),  
    Weight DECIMAL(5, 2),  
    DOB DATE,  
    CountryName VARCHAR(100),  
    FOREIGN KEY (CountryName) REFERENCES Country(CountryName),  
);
```

VENUE:


Below is the CREATE table statement for the VENUE table.

```
CREATE TABLE Venue (  
    VenueId INT PRIMARY KEY,  
    VenueName VARCHAR(100),  
    Capacity INT,  
    Location VARCHAR(100),  
);
```

Schedule:

Below is the CREATE table statement for the schedule table.

```
CREATE TABLE Schedule (  
    ScheduleId NUMBER PRIMARY KEY,  
    SDate DATE,
```

```
Start_Time VARCHAR2(8) not null,  
End_Time VARCHAR2(8) not null,  
);
```

GAME:


Below is the CREATE table statement for the GAME table.

```
CREATE TABLE Game (  
    GameID NUMBER PRIMARY KEY,  
    SportName VARCHAR2(50) NOT NULL,  
    VenuelD Number NOT NULL,  
    TicketsSold NUMBER,  
    PricePerTicket NUMBER(10, 2),  
    SponsorsPrice NUMBER(10, 2),  
    ScheduleId NUMBER ,  
    FOREIGN KEY (SportName) REFERENCES Sport(SportName) ON DELETE CASCADE,  
    FOREIGN KEY (VenuelD) REFERENCES Venue(VenuelD) ON DELETE CASCADE,  
    FOREIGN KEY (ScheduleId) REFERENCES schedule(ScheduleId) ON DELETE  
    CASCADE  
)
```

Scores:

Below is the CREATE table statement for the scores table.

```
CREATE TABLE Scores (  
    AthletelD NUMBER,  
    SportName VARCHAR2(100),  
    GameNo NUMBER,  
    Score NUMBER,  
    ScoreType VARCHAR2(50),
```



```
PRIMARY KEY (Athleteld, SportName, GameNo),  
FOREIGN KEY (Athleteld) REFERENCES Athlete(Athleteld),  
FOREIGN KEY (SportName) REFERENCES Sport(SportName)  
);
```

AthleteTeam Table:

Below is the CREATE table statement for the Athlete table.

```
CREATE TABLE AthleteTeam (  
    Athleteld NUMBER,  
    TeamId NUMBER,  
    PRIMARY KEY (Athleteld, TeamId),  
    FOREIGN KEY (Athleteld) REFERENCES Athlete(AthleteID),  
    FOREIGN KEY (TeamId) REFERENCES Team(TeamID)  
);
```

AthleteMedal Table:

Below is the CREATE table statement for the Athlete table.

```
CREATE TABLE AthleteMedal (  
    Athleteld NUMBER,  
    MedalName varchar2(50),  
    Gameld NUMBER,  
    PRIMARY KEY (Athleteld, Medalname, Gameld),  
  
    FOREIGN KEY (Athleteld) REFERENCES Athlete(AthleteID),  
    FOREIGN KEY (Gameld) REFERENCES Game(Gameld)  
    FOREIGN KEY (MedalName) REFERENCES Medal(MedalName),  
  
);
```

Entering data to our system:

Below are the DML commands to insert data into the system. For simplicity, here only INSERT commands are displayed. Values were inserted after verifying locally on the notebook.

SPORT:

Below is the INSERT command to input data to SPORT table.

INSERT ALL

INTO Sport (SportName, Category, NoOfGames) VALUES ('Basketball', 'Team', 4)

INTO Sport (SportName, Category, NoOfGames) VALUES ('Swimming', 'Individual', 1)

INTO Sport (SportName, Category, NoOfGames) VALUES ('Running', 'Individual', 1)

SELECT * FROM DUAL

SPORTNAME	CATEGORY	NOOFGAMES
Shooting	Individual	30
Athletics	Individual	40
Football	Team	15
Basketball	Team	4
Swimming	Individual	1
Running	Individual	1

6 rows returned in 0.00 seconds

[Download](#)

MEDAL:

Below is the INSERT command to input data to MEDAL table.

INSERT ALL

INTO Medal (MedalName, RewardPrize) VALUES ('Gold', 50000)

```
INTO Medal (MedalName, RewardPrize) VALUES ('Silver', 30000)
INTO Medal (MedalName, RewardPrize) VALUES ('Bronze', 10000)
SELECT * FROM DUAL
```

MEDALNAME	REWARDPRIZE
Gold	50000
Silver	30000
Bronze	10000

3 rows returned in 0.00 seconds

[Download](#)

TEAM:

Below is the INSERT command to input data to TEAM table.

```
INSERT ALL
```

```
  INTO TEAM (TEAMID, TEAMNAME, NOOFMEMBERS,GameId) VALUES (1, 'USA
Dream Team', 10,1)
```

```
  INTO TEAM (TEAMID, TEAMNAME, NOOFMEMBERS,GameId) VALUES (2, 'Canada
Hoopers', 10,2)
```

```
  INTO TEAM (TEAMID, TEAMNAME, NOOFMEMBERS,GameId) VALUES (3, 'Japan
Hoopers', 10,3)
```

```
SELECT * FROM DUAL;
```

Results Explain Describe SQL History

TEAMID	TEAMNAME	NOOFMEMBER	History	AMEID
1	USA Dream Team	10		1
2	Canada Hoopers	10		2
3	Japan Hoopers	10		3

3 rows returned in 0.01 seconds

[Download](#)

COUNTRY:

Below is the INSERT command to input data to COUNTRY table.

INSERT ALL

INTO COUNTRY (COUNTRYNAME, NOOFATHLETES, TEAMID) VALUES ('United States', 2, 1)

INTO COUNTRY (COUNTRYNAME, NOOFATHLETES, TEAMID) VALUES ('Canada', 1, 1)

INTO COUNTRY (COUNTRYNAME, NOOFATHLETES, TEAMID) VALUES ('Japan', 3, 1)

SELECT * FROM DUAL;

COUNTRYNAME	NOOFATHLETES	TEAMID
United States	2	1
Canada	1	1
Japan	3	1

3 rows returned in 0.00 seconds

[Download](#)

ATHLETE:

Below is the INSERT command to input data to ATHLETE table.

INSERT ALL

INTO ATHLETE (ATHLETEID, ATHLETENAME, HEIGHT, WEIGHT, DOB,
COUNTRYNAME, SPORTNAME, MEDALNAME) VALUES (1, 'Michael Phelps', 193, 91,
NULL, 'United States', 'Swimming', 'Gold')

INTO ATHLETE (ATHLETEID, ATHLETENAME, HEIGHT, WEIGHT, DOB,
COUNTRYNAME, SPORTNAME, MEDALNAME) VALUES (2, 'Allyson Felix', 168, 55,
NULL, 'United States', 'Running', 'Bronze')

INTO ATHLETE (ATHLETEID, ATHLETENAME, HEIGHT, WEIGHT, DOB,
COUNTRYNAME, SPORTNAME, MEDALNAME) VALUES (4, 'Penny Oleksiak', 186, 68,
NULL, 'Canada', 'Swimming', 'Bronze')

INTO ATHLETE (ATHLETEID, ATHLETENAME, HEIGHT, WEIGHT, DOB,
COUNTRYNAME, SPORTNAME, MEDALNAME) VALUES (6, 'Kosuke Hagino', 177, 71,
NULL, 'Japan', 'Swimming', 'Silver')

INTO ATHLETE (ATHLETEID, ATHLETENAME, HEIGHT, WEIGHT, DOB,
COUNTRYNAME, SPORTNAME, MEDALNAME) VALUES (7, 'Naoko Takahashi', 158, 45,
NULL, 'Japan', 'Running', 'Silver')

INTO ATHLETE (ATHLETEID, ATHLETENAME, HEIGHT, WEIGHT, DOB,
COUNTRYNAME, SPORTNAME, MEDALNAME) VALUES (8, 'Yuki Kawauchi', 170, 62,
NULL, 'Japan', 'Running', 'Gold')

SELECT * FROM dual;

ATHLETEID	ATHLETENAME	HEIGHT	WEIGHT	DOB	COUNTRYNAME
1	Michael Phelps	193	91	-	United States
2	Allyson Felix	168	55	-	United States
4	Penny Oleksiak	186	68	-	Canada
6	Kosuke Hagino	177	71	-	Japan
7	Naoko Takahashi	158	45	-	Japan
8	Yuki Kawauchi	170	62	-	Japan

6 rows returned in 0.00 seconds

[Download](#)

VENUE:

Below is the INSERT command to input data to VENUE table.

INSERT ALL

INTO VENUE (VENUEID, VENUENAME, CAPACITY, LOCATION, SPORTNAME) VALUES
(1, 'Olympic Arena', 20000, 'Los Angeles, USA', 'Basketball')

INTO VENUE (VENUEID, VENUENAME, CAPACITY, LOCATION, SPORTNAME) VALUES
(2, 'Aquatic Center', 15000, 'Toronto, Canada', 'Swimming')

INTO VENUE (VENUEID, VENUENAME, CAPACITY, LOCATION, SPORTNAME) VALUES
(3, 'National Stadium', 10000, 'Tokyo, Japan', 'Running')

SELECT * FROM DUAL;

Results	Explain	Describe	Saved SQL	History
VENUEID	VENUENAME	CAPACITY	LOCATION	SPORTNAME
1	Olympic Arena	20000	Los Angeles, USA	Basketball
2	Aquatic Center	15000	Toronto, Canada	Swimming
3	National Stadium	10000	Tokyo, Japan	Running

3 rows returned in 0.01 seconds [Download](#)

Schedule:

Below is the INSERT command to input data to schedule table.

INSERT ALL

INTO SCHEDULE (SCHEDULEID, SPORTNAME, SDATE, START_TIME, END_TIME)
VALUES (1, 'Basketball', TO_DATE('2024-08-20', 'YYYY-MM-DD'), '14:00:00',
'16:00:00')

INTO SCHEDULE (SCHEDULEID, SPORTNAME, SDATE, START_TIME, END_TIME)
VALUES (2, 'Swimming', TO_DATE('2024-08-21', 'YYYY-MM-DD'), '10:00:00',
'12:00:00')

```

    INTO SCHEDULE (SCHEDULEID, SPORTNAME, SDATE, START_TIME, END_TIME)
VALUES (3, 'Running', TO_DATE('2024-08-22', 'YYYY-MM-DD'), '09:00:00', '11:00:00')
    INTO SCHEDULE (SCHEDULEID, SPORTNAME, SDATE, START_TIME, END_TIME)
VALUES (4, 'Basketball', TO_DATE('2024-08-23', 'YYYY-MM-DD'), '16:00:00',
'18:00:00')
    INTO SCHEDULE (SCHEDULEID, SPORTNAME, SDATE, START_TIME, END_TIME)
VALUES (5, 'Swimming', TO_DATE('2024-08-24', 'YYYY-MM-DD'), '12:00:00',
'14:00:00')
    INTO SCHEDULE (SCHEDULEID, SPORTNAME, SDATE, START_TIME, END_TIME)
VALUES (6, 'Running', TO_DATE('2024-08-25', 'YYYY-MM-DD'), '11:00:00', '13:00:00')
SELECT * FROM dual;

```

results Explain Describe Saved SQL History

SCHEDULEID	SPORTNAME	SDATE	START_TIME	END_TIME
1	Basketball	08/20/2024	14:00:00	16:00:00
2	Swimming	08/21/2024	10:00:00	12:00:00
3	Running	08/22/2024	09:00:00	11:00:00
4	Basketball	08/23/2024	16:00:00	18:00:00
5	Swimming	08/24/2024	12:00:00	14:00:00
6	Running	08/25/2024	11:00:00	13:00:00

Games:

Below is the INSERT command to input data to games table.

INSERT ALL

```

    INTO GAME (GAMEID, SPORTNAME, VENUEID, TICKETSSOLD, PRICEPERTICKET,
SPONSORSPRICE, SCHEDULEID) VALUES (1, 'Basketball', 1, 15000, 85, 20000, 1)
    INTO GAME (GAMEID, SPORTNAME, VENUEID, TICKETSSOLD, PRICEPERTICKET,
SPONSORSPRICE, SCHEDULEID) VALUES (2, 'Basketball', 1, 12000, 80, 15000, 2)

```



```

    INTO GAME (GAMEID, SPORTNAME, VENUEID, TICKETSSOLD, PRICEPERTICKET, SPONSORSPRICE, SCHEDULEID) VALUES (3, 'Basketball', 1, 13000, 95, 18000, 3)
    INTO GAME (GAMEID, SPORTNAME, VENUEID, TICKETSSOLD, PRICEPERTICKET, SPONSORSPRICE, SCHEDULEID) VALUES (4, 'Basketball', 1, 13000, 95, 18000, 4)
    INTO GAME (GAMEID, SPORTNAME, VENUEID, TICKETSSOLD, PRICEPERTICKET, SPONSORSPRICE, SCHEDULEID) VALUES (5, 'Swimming', 2, 10000, 75, 15000, 5)
    INTO GAME (GAMEID, SPORTNAME, VENUEID, TICKETSSOLD, PRICEPERTICKET, SPONSORSPRICE, SCHEDULEID) VALUES (6, 'Running', 3, 9000, 60, 14000, 6)
    SELECT * FROM DUAL;

```

GAMEID	SPORTNAME	VENUEID	TICKETSSOLD	PRICEPERTICKET	SPONSORSPRICE	SCHEDULEID
1	Basketball	1	15000	85	20000	1
2	Basketball	1	12000	80	15000	2
3	Basketball	1	13000	95	18000	3
4	Basketball	1	13000	95	18000	4
5	Swimming	2	10000	75	15000	5
6	Running	3	9000	60	14000	6

6 rows returned in 0.00 seconds

[Download](#)

Scores:

Below is the INSERT command to input data to Scores table.

INSERT ALL

```

    INTO SCORES (ATHLETEID, SPORTNAME, GAMENO, SCORE, SCORETYPE) VALUES (1, 'Swimming', 1, 50.34, 'Time')

```

```

    INTO SCORES (ATHLETEID, SPORTNAME, GAMENO, SCORE, SCORETYPE) VALUES (2, 'Running', 2, 9.58, 'Time')

```

```

    INTO SCORES (ATHLETEID, SPORTNAME, GAMENO, SCORE, SCORETYPE) VALUES (8, 'Basketball', 3, 102, 'Points')

```

```

    INTO SCORES (ATHLETEID, SPORTNAME, GAMENO, SCORE, SCORETYPE) VALUES (4, 'Swimming', 4, 45.67, 'Time')

```

```

    INTO SCORES (ATHLETEID, SPORTNAME, GAMENO, SCORE, SCORETYPE) VALUES (7, 'Running', 5, 3.45, 'Time')

```

INTO SCORES (ATHLETEID, SPORTNAME, GAMENO, SCORE, SCORETYPE) VALUES (6,
'Basketball', 6, 88, 'Points')
SELECT * FROM DUAL;

ATHLETEID	GAMENO	SCORE
1	1	50.34
2	2	9.58
8	3	102
4	4	45.67
7	5	3.45
6	6	88

6 rows returned in 0.01 seconds

[Download](#)

AthleteTeam:

Below is the INSERT command to input data to AthleteTeam table.

INSERT all

INTO AthleteTeam (AthletelId, TeamId) VALUES

(1, 1)

INTO AthleteTeam (AthletelId, TeamId) VALUES

(2, 2)

INTO AthleteTeam (AthletelId, TeamId) VALUES

(4, 3)

INTO AthleteTeam (AthletelId, TeamId) VALUES

(6, 2)

select * from dual

ATHLETEID	TEAMID
1	1
2	2
4	3
6	2

4 rows returned in 0.00 seconds

AthleteMedal:

Below is the INSERT command to input data to AthleteMedal table.

```
INSERT all
```

```
  INTO AthleteMedal (AthleteId, MedalName, GameId) VALUES
```

```
(1, 'Gold', 2)
```

```
  INTO AthleteMedal (AthleteId, MedalName, GameId) VALUES
```

```
(2, 'Gold', 3)
```

```
  INTO AthleteMedal (AthleteId, MedalName, GameId) VALUES
```

```
(7, 'Bronze', 4)
```

```
  INTO AthleteMedal (AthleteId, MedalName, GameId) VALUES
```

```
(4, 'Silver', 5)
```

```
select * from dual
```

ATHLETEID	MEDALNAME	GAMEID
1	Gold	2
2	Gold	3
4	Silver	5
7	Bronze	4

4 rows returned in 0.00 seconds

[Download](#)

VIEWS:

Below are some important views created related to our database system.

1. AthleteMedalsLeaderboard:

This is the leaderboard that show the number of medals won by countries.

```
CREATE or replace VIEW AthleteMedalsLeaderboard AS
SELECT
    a.AthleteID,
    a.AthleteName,
    COUNT(CASE WHEN am.MedalName = 'Gold' THEN 1 END) AS GoldMedals,
    COUNT(CASE WHEN am.MedalName = 'Silver' THEN 1 END) AS
SilverMedals,
    COUNT(CASE WHEN am.MedalName = 'Bronze' THEN 1 END) AS
BronzeMedals
FROM
    Athlete a
```

JOIN

AthleteMedal am ON a.AthleteID = am.AthleteID

GROUP BY

a.AthleteID, a.AthleteName

ORDER BY

GoldMedals DESC, SilverMedals DESC, BronzeMedals DESC;

ATHLETEID	ATHLETENAME	GOLDMEDALS	SILVERMEDALS	BRONZEMEDALS
2	Allyson Felix	1	0	0
1	Michael Phelps	1	0	0
4	Penny Oleksiak	0	1	0
7	Naoko Takahashi	0	0	1

4 rows returned in 0.01 seconds [Download](#)

2. CountriesLeaderboard:

This view will display each country and the total number of Gold, Silver, and Bronze medals won by athletes from that country.

CREATE VIEW CountriesLeaderboard AS

SELECT

c.CountryName,

COUNT(CASE WHEN am.MedalName = 'Gold' THEN 1 END) AS GoldMedals,

COUNT(CASE WHEN am.MedalName = 'Silver' THEN 1 END) AS

SilverMedals,

COUNT(CASE WHEN am.MedalName = 'Bronze' THEN 1 END) AS

BronzeMedals,

COUNT(am.MedalName) AS TotalMedals

FROM

Country c

JOIN

AthleteMedal am ON g.GameID = am.GameID
 GROUP BY
 g.SportName
 ORDER BY
 GoldMedals DESC, SilverMedals DESC, BronzeMedals DESC;

SPORTNAME	GOLDMEDALS	SILVERMEDALS	BRONZEMEDALS	TOTALMEDALS
Basketball	2	0	1	3
Swimming	0	1	0	1

2 rows returned in 0.01 seconds [Download](#)

4. TopPerformingAthletes:

This view lists athletes along with their scores in various games, sorted by highest scores for each sport.

```
CREATE VIEW TopPerformingAthletes AS
SELECT
    a.AthleteID,
    a.AthleteName,
    g.SportName,
    s.Score

FROM
    Athlete a
JOIN
    Scores s ON a.AthleteID = s.AthleteID
JOIN
    Game g ON s.GameNo = g.GameID
ORDER BY
    g.SportName, s.Score DESC;
```

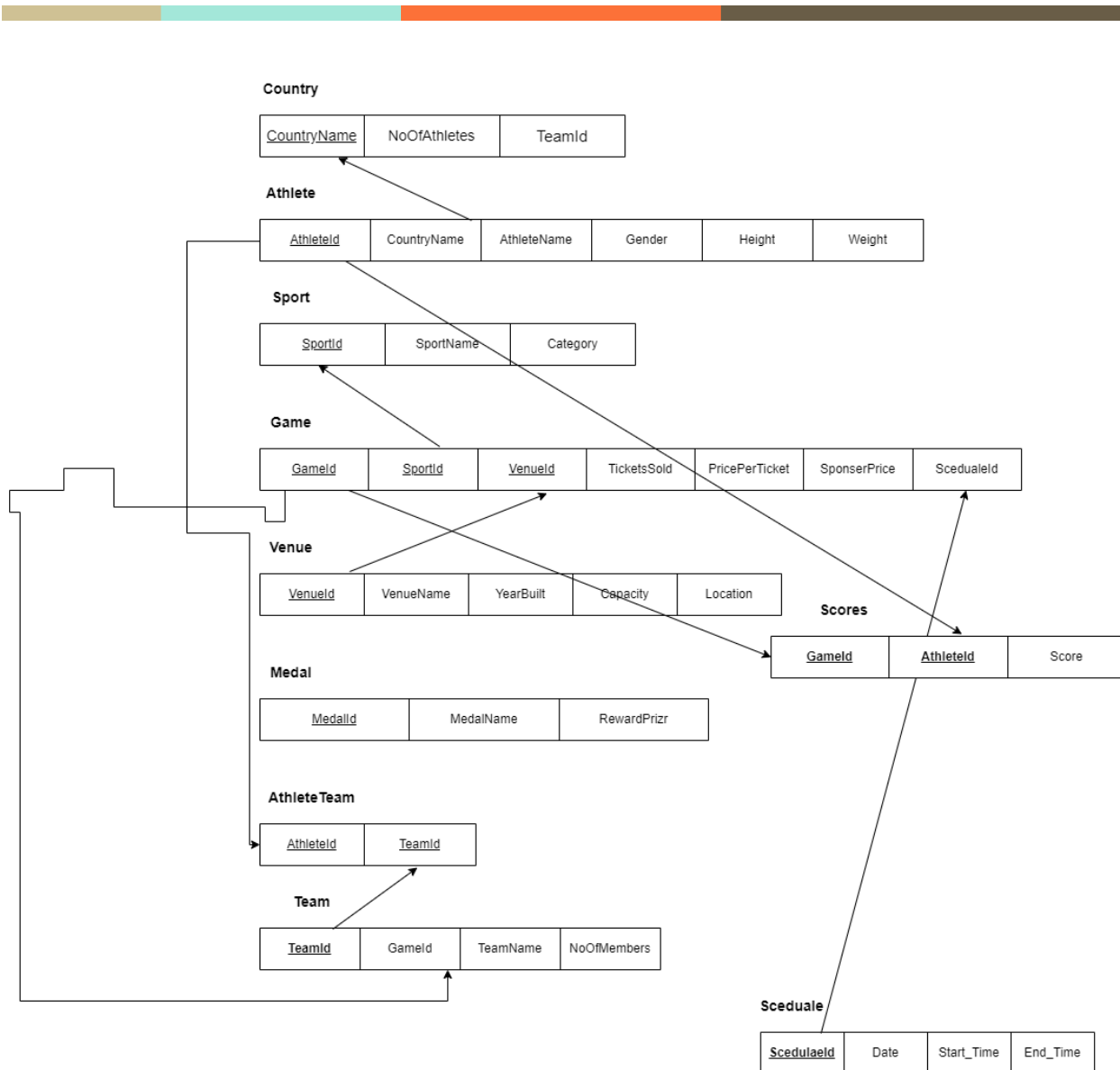
ATHLETEID	ATHLETENAME	SPORTNAME	SCORE
8	Yuki Kawauchi	Basketball	102
1	Michael Phelps	Basketball	50.34
4	Penny Oleksiak	Basketball	45.67
2	Allyson Felix	Basketball	9.58
6	Kosuke Hagino	Running	88
7	Naoko Takahashi	Swimming	3.45

6 rows returned in 0.00 seconds

[Download](#)

Association between different Relations:

Below are the relations and the association between them shown using arrows.



SELECT Statements:

Below are some SELECT statements that retrieve important data from the system. All these queries use either JOINS or Subqueries.

1. Total Revenue Generated by Each Sport

```

SELECT
    g.SportName,
    SUM(g.TicketsSold * g.PricePerTicket) AS TotalRevenue
FROM

```

Game g
GROUP BY
g.SportName

ORDER BY
TotalRevenue DESC;

Results Columns Rows Data Shape Memory

SPORTNAME	TOTALREVENUE
Basketball	4705000
Swimming	750000
Running	540000

3 rows returned in 0.00 seconds

[Download](#)

2.Athletes Who Didn't Win a Medal

```
SELECT
  a.AthleteID,
  a.AthleteName,
  a.CountryName
FROM
  Athlete a
LEFT JOIN
  AthleteMedal am ON a.AthleteID = am.AthleteID
WHERE
  am.MedalName IS NULL;
```

Results Explain Describe Saved SQL History

ATHLETEID	ATHLETENAME	COUNTRYNAME
6	Kosuke Hagino	Japan
8	Yuki Kawauchi	Japan

2 rows returned in 0.01 seconds [Download](#)

3. Countries Which Never Won Any Medal

```
SELECT
  c.CountryName
FROM
  Country c
LEFT JOIN
  Athlete a ON c.CountryName = a.CountryName
LEFT JOIN
  AthleteMedal am ON a.AthleteID = am.AthleteID
WHERE
  am.MedalName IS NULL
GROUP BY
  c.CountryName;
```

Results Explain Describe Saved SQL History

COUNTRYNAME
Japan

1 rows returned in 0.01 seconds [Download](#)

4. Countries with Total Medals Count

```
SELECT
  c.CountryName,
  COUNT(am.MedalName) AS TotalMedals
FROM
  Country c
JOIN
  Athlete a ON c.CountryName = a.CountryName
JOIN
  AthleteMedal am ON a.AthleteID = am.AthleteID
GROUP BY
  c.CountryName
ORDER BY
  TotalMedals DESC;
```

Results Explain Describe Saved SQL History

COUNTRYNAME	TOTALMEDALS
United States	2
Japan	1
Canada	1

3 rows returned in 0.01 seconds

[Download](#)

5. Athletes in Order of Their Medal Count

```
SELECT
  a.AthleteID,
  a.AthleteName,
  COUNT(am.MedalName) AS TotalMedals
```

```

FROM
  Athlete a
JOIN
  AthleteMedal am ON a.AthleteID = am.Athletelid
GROUP BY
  a.AthleteID, a.AthleteName
ORDER BY
  TotalMedals DESC;

```

[Results](#)
[Expand](#)
[Describe](#)
[Saved SQL](#)
[History](#)

ATHLETEID	ATHLETENAME	TOTALMEDALS
2	Allyson Felix	1
7	Naoko Takahashi	1
1	Michael Phelps	1
4	Penny Oleksiak	1

4 rows returned in 0.01 seconds

[Download](#)

6. Players Count by Sport

```

SELECT
  g.SportName,
  COUNT(DISTINCT s.Athletelid) AS PlayerCount
FROM
  Game g
JOIN
  Scores s ON g.GameID = s.GameNo

```

GROUP BY

g.SportName

ORDER BY

PlayerCount DESC;

SPORTNAME	PLAYERCOUNT
Basketball	4
Swimming	1
Running	1

3 rows returned in 0.01 seconds

[Download](#)

7. Top 5 Highest Scoring Athletes by Sport

SELECT

g.SportName,

a.AthleteName,

s.Score

FROM

Game g

JOIN

Scores s ON g.GameID = s.GameNo

JOIN

Athlete a ON s.AthleteID = a.AthleteID

ORDER BY

g.SportName, s.Score DESC

Results Explain Describe Saved SQL History

SPORTNAME	ATHLETENAME	SCORE
Basketball	Yuki Kawauchi	102
Basketball	Michael Phelps	50.34
Basketball	Penny Oleksiak	45.67
Basketball	Allyson Felix	9.58
Running	Kosuke Hagino	88
Swimming	Naoko Takahashi	3.45

6 rows returned in 0.01 seconds [Download](#)

8. Games with Most Tickets Sold

```
SELECT
  g.GameID,
  g.SportName,
  g.TicketsSold,
  g.TicketsSold * g.PricePerTicket AS TotalRevenue
FROM
  Game g
ORDER BY
  g.TicketsSold DESC;
```

GAMEID	SPORTNAME	TICKETSSOLD	TOTALREVENUE
1	Basketball	15000	1275000
3	Basketball	13000	1235000
4	Basketball	13000	1235000
2	Basketball	12000	960000
5	Swimming	10000	750000
6	Running	9000	540000

6 rows returned in 0.00 seconds

[Download](#)
