



**UNIVERSITY
OF LONDON**

CM3010 Database and Advanced Techniques

Midterms Paper

Student ID: 220516639

Name: Teo Jia En

Date of Submission: 5th January 2025

Table of Contents

DATABASE AND ADVANCED TECHNIQUES	1
1. INTRODUCTION	3
1.1 ABSTRACT	3
1.2 DOMAIN OF CHOICE	3
2. DATASET SELECTION AND CRITIQUE	3
2.1 DATASET PRE-PROCESSING	7
2.1.2 <i>Cutting down on the data</i>	8
2.1.3 <i>Saving the cleaned data</i>	8
2. MODEL OF DATA	8
2.1 ERD DIAGRAM	8
2.2 RELATIONAL DATABASE SCHEMA	10
2.3 BNFC ANALYSIS	11
3. DATABASE CREATION	13
3.1 USER CONFIGURATION	13
3.2 CREATION OF TABLE TO STORE RAW DATA	14
3.3 CREATE TABLES	15
3.4 POPULATE TABLES	17
STAGE 4: CREATE A SIMPLE WEB APPLICATION	18
RESEARCH QUESTION ONE: NUMBER OF SPORTS	20
RESEARCH QUESTION TWO: AVERAGE AGE OF OLYMPIANS	20
RESEARCH QUESTION THREE: AN AGE-OLD QUESTION	21
RESEARCH QUESTION FOUR: NOC AND ACCOLADES	22
RESEARCH QUESTION FIVE: FEMALE ATHLETES AND GOLD MEDALS	23
RESEARCH QUESTION SIX: THE YOUNG OLYMPIAN	24
RESEARCH QUESTION 7: THE AVERAGE OLYMPIAN	26
RESEARCH QUESTION EIGHT: THE CROSS-SEASON OLYMPIAN	27
RESEARCH QUESTION NINE: GOLDIES	28
RESEARCH QUESTION TEN: NOC PARTICIPATION	30
WORKS CITED	33

1. Introduction

1.1 Abstract

This report presents an in-depth analysis of four consecutive Olympic Games held between 2010 and 2016. The research encompassed the development of a robust MySQL database and a dynamic Node.js web application. By structuring the Olympic Games data within a relational database, this project aims to provide valuable insights into the evolving demographics of Olympic athletes during this period. Analysing demographic data can reveal trends in athlete participation, such as the participate rate by gender, or nationality representation over time.

In fulfilment of this coursework, the project demonstrates knowledge and skills learned across the past ten weeks, which includes modelling of relational databases, execution of complex SQL queries for data analysis and development of a web application to allow users to analyse data interactively.

1.2 Domain of choice

From record-breaking feats to historic achievements, the Olympics is regarded as the world's foremost international sports event. [1] With its roots in Ancient Greece and its first modern version in Athens, the Olympics is an age-old event that everyone looks forward to on a biannual basis. This year, Paris Olympics offered unforgettable moments, with Simon Biles winning gold in the women's vault after her two-year break [2], to the introduction of breaking, or break-dancing, as a new sport [3]. What makes the Olympics captivating is that athletes from all around the world congregate together to compete on many different sports. [4] This makes cataloguing the event a complex endeavour, as the games and athletes evolve over time. Hence, a profound fascination with the Olympic Games serves as the impetus for this project.

2. Dataset Selection and Critique

The primary data source for this project is a publicly available historical dataset that documents the past 120 years of Olympic games. It encompasses essential information pertaining to each game, including where it is held, the city it is held at, and various details on the athletes and the medals they won. [5]

The original dataset contains 271,116 entries and 15 columns.

1	ID	Unique identifier for every athlete
2	Name	Full name of the athlete
3	Sex	Denotes whether the athlete is male or female. Since the dataset covers the Games up to 2016 and the first openly non-binary athlete participated in the Tokyo 2020, there are only two possible values for this attribute in this project. [6]

4	Age	Age of the athlete at that game
5	Height	Height of the athlete at that game
6	Weight	Weight of the athlete at that game
7	Team	Name of the team the athlete represents
8	NOC	National Olympic Committee 3-letter code
9	Games	The year and season of the game
10	Year	The year the game is held in
11	Season	Winter or Summer
12	City	The city the game is held in
13	Sport	Sport the event is in (Eg. Swimming)
14	Event	Even the athlete is competing in (Eg. Men's Individual Butterfly)
15	Medal	Medal the athlete achieved

The data was acquired as a comma-separated values (CSV) file from Kaggle. The data source was deemed appropriate as it fulfils the following criteria:

- Quality: The data is reliable, as it uses data from Sports Reference, a website that provides robust sports statistics. [7] Cross-checking of random records with the Olympics website and various other data sources, like Dataful, have also validated that the data is accurate. [8]
- Detail: There are 15 columns in the dataset that provide a holistic view of the athletes, the games they participated in and the medals they won. These attributes can potentially be used to perform analyses, such as identifying participation rate of a particular gender, or comparing performance across games and countries.
- Documentation: The dataset was accompanied by a short blurb and description of each column, which was sufficient to understand the dataset. Each column in the file has a clear and informative column header. While certain attributes like height and weight have missing units of measurement, it can easily be deduced from the range of data.

```
mysql> SELECT ROUND(AVG(height)), ROUND(AVG(weight)) FROM athlete_game;
+-----+-----+
| ROUND(AVG(height)) | ROUND(AVG(weight)) |
+-----+-----+
| 177 | 72 |
+-----+-----+
1 row in set (0.02 sec)
```

An average of 117 suggests that height is in cm, while an average of 72 suggests that weight is in kg.

- Interrelation: A multidisciplinary approach could yield richer insights into the Games. For example, incorporating economic data such as the GDP of each country could reveal correlations between a country's economy and its athletes' performance.

This interdisciplinary approach, however, comes with a set of challenges. Not all countries have reliable economic figures. A severe imbalance in terms of the number of athletes the countries send, as shown below, would impair analysis and create biased results.

```
mysql> SELECT noc_id, COUNT(DISTINCT athlete_id) AS athlete_count
-> FROM athlete_game
-> GROUP BY noc_id
-> ORDER BY athlete_count DESC
-> LIMIT 5;
```

noc_id	athlete_count
18	1176
9	874
31	854
56	808
29	780

5 rows in set (0.02 sec)

```
mysql> SELECT noc_id, COUNT(DISTINCT athlete_id) AS athlete_count
-> FROM athlete_game
-> GROUP BY noc_id
-> ORDER BY athlete_count ASC
-> LIMIT 5;
```

noc_id	athlete_count
232	2
228	2
194	2
91	2
267	2

5 rows in set (0.02 sec)

Hence, this project will focus primarily on the Games itself and the athletes' performance in these Games

- Use: The dataset will be mainly used to answer questions on demographic of the athletes – such as the average physical attributes of the Olympians in a particular sport, or the age of a Olympian. Some data that would offer richer insights in this area but is missing include – details on the athlete's coach, the athlete's performance at other events such as the Commonwealth Games and the athlete's years of experience in that sport.

- Discoverability: It was easy to find open data in the chosen domain of the Olympic Games. However, most data only pertains to one single Game and does not span across multiple Games.

Other data sources that have been considered, but ultimately eliminated, for this project include:

- Olympics official website: This authoritative website contains a substantial volume of information that are dispersed across numerous web pages. [9] Consolidating this information would require the use of web scrapers. However, the site has no clear policy on the use of automated robots. To prevent any potential legal infractions, this project opted to exclude the use of this website.
- Olympic Data Feed (ODF): This repository contains documentation of past Olympic games and serves as the Documentation Web site of the International Olympic Committee (IOC). [10] However, the information is stored as pdf documents and could not be easily consolidated.

2.1 Term of use and licences on data

The owner of the dataset has indicated that it is licenced under CC0 1.0 Universal Deed. [11] The Sports Reference, where the data originates, permits the use of its data for personal and non-commercial purposes, provided that the data is not used to create a database that compete with the platform, or used to train generative machine learning algorithms. The dataset will be used solely for the purpose of building a database as fulfilment of the midterms for this module and will be destroyed after the assessment is over.

2.2 Interest in the data

The comprehensive dataset offers a granular perspective on an individual athlete's participation in each Olympic Games. The questions below facilitates a more nuanced understanding of the factors that influence athlete success and broader trends that shape the modern Olympic landscape.

Questions:

1. How many sports were there at each of the four games?
2. What is the average age of athletes that won medals?
3. How old is the oldest Olympian?
4. Which NOC won the most bronze, silver and gold medals across all four games?
5. Which female athletes accumulated the most gold medals?
6. How old is the youngest Olympian and what country do they represent?

7. What is the average height and weight of athletes in a sport?
8. Were there athletes that participated in both Summer and Winter games and won medals?
9. Were there athletes that always ended up on the podium at all their events in London 2012 and won Gold medals at both London and Rio 2016?
10. Were there any NOCs that participated in the 2010 Vancouver games but not the 2014 Sochi games?

This project aims to answer these questions using a relational model and MySQL queries.

2.1 Dataset pre-processing

Data pre-processing for this project was made with the use of Pandas and NumPy. Pandas is a Python data manipulation and analysis tool while NumPy is a memory-efficient tool to work with numerical data. [12]

2.1.1 Handling missing values

The medals column uses null values to represent athletes who did not receive a medal. To better represent the data and accord more meaning, these fields are replaced with a string 'No medal'.

```
✓ 0s df['Medal'].fillna('No medal', inplace = True)
```

The dataset contains missing values in 'age' and 'weight' columns, initially represented by NumPy's NaN. Since SQL does not natively support this data type, replacing them with Python's NoneType object, which is more compatible with SQL's NULL handling, will facilitate data integration and processing within the SQL environment.

```
✓ 0s [15] import numpy as np
      df = df.replace(np.nan, None)
```

```
mysql> SELECT COUNT(*)
      -> FROM athlete_game
      -> WHERE weight IS NULL;
+-----+
| COUNT(*) |
+-----+
|      652 |
+-----+
1 row in set (0.00 sec)
```

df.head(3)

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Season	City	Sport	Event	Medal
0	1	A Dijiang	M	24.0	180.0	80.0	China	CHN	1992 Summer	1992	Summer	Barcelona	Basketball	Basketball Men's Basketball	No medal
1	2	A Lamusi	M	23.0	170.0	60.0	China	CHN	2012 Summer	2012	Summer	London	Judo	Judo Men's Extra-Lightweight	No medal
2	3	Gunnar Nielsen Aaby	M	24.0	None	None	Denmark	DEN	1920 Summer	1920	Summer	Antwerpen	Football	Football Men's Football	No medal

2.1.2 Cutting down on the data

The original data contains 271,116 entries. To cater for data processing on Coursera Labs, the data has been limited to include only Games that took place after the year 2010. This reduced the dataset to around 36,000 entries.

```
✓ 0s df2 = df[df['Year'] >= 2010]
```

2.1.3 Saving the cleaned data

```
✓ 0s # save the clean data to a csv file for data loading and ingestion
df2.to_csv(f"{DATA_PATH}/athletes_events_cleaned.csv", index=False)
```

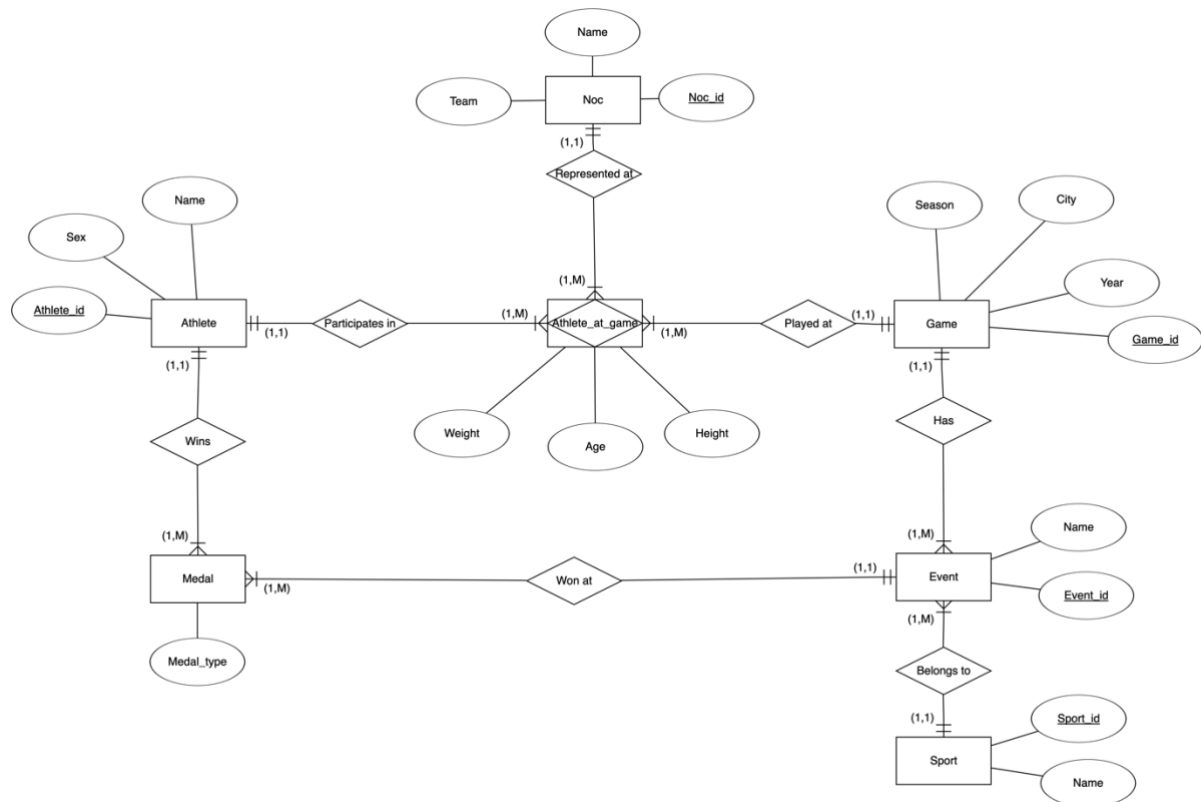
The data is then saved as a new CSV file with the Pandas “to_csv” method.

2. Model of data

2.1 ERD Diagram

To effectively model the relationship between the various entities within the dataset, it is important to construct an accurate Entity-Relationship Diagram (ERD). The diagram

provides a good representation of the various relations between the entities.



The ERD above uses the conceptual Chen notation. It is created with the ERD Plus tool online that provides standard ERD components such as entities, attributes and relationships. The ERD models a database that captures key entities in the dataset – such as Athlete, Game and Event.

Athlete	<ul style="list-style-type: none"> - Athlete_id: Unique identifier for the athlete. - Name of the athlete. - Sex of the athlete.
Game	<ul style="list-style-type: none"> - Game_id: Unique identifier for the Game - Season: Summer or Winter - City: City the game is held (Eg. London) - Year (Eg. 2012)
Noc	<ul style="list-style-type: none"> - Noc_id: Unique identifier for each National Olympic Committee. - Name: The NOC code. (eg. USA) - Team: The name of the team (eg. United States 1)
Sport	<ul style="list-style-type: none"> - Sport_id: Unique identifier for each sport. - Name: The name of the sport (eg. “Swimming”)
Event	<ul style="list-style-type: none"> - Event_id: Unique identifier for each event. - Name: Name of the event (eg. “Women’s 100M Butterfly Event)

Medal	Medal_type: The type of medal won (eg. “Gold, Silver, Bronze, No medal”)
Athlete_at_game	<p>This entity is an associative entity that resolves many-to-many relationships present in the database. [13]</p> <ul style="list-style-type: none"> - The first many-to-many relationship exists between Athletes and Games – one athlete can participate in multiple Games and one Game is participated by many athletes. - The second many-to-many relationship exists between NOC and Games – one NOC can participate in multiple Games and one Game is participated by many athletes. <p>The Athlete_at_game converts each many-to-many relationship into two one-to-many relationships. With this table, only one-to-many relationships exist between the entities.</p>

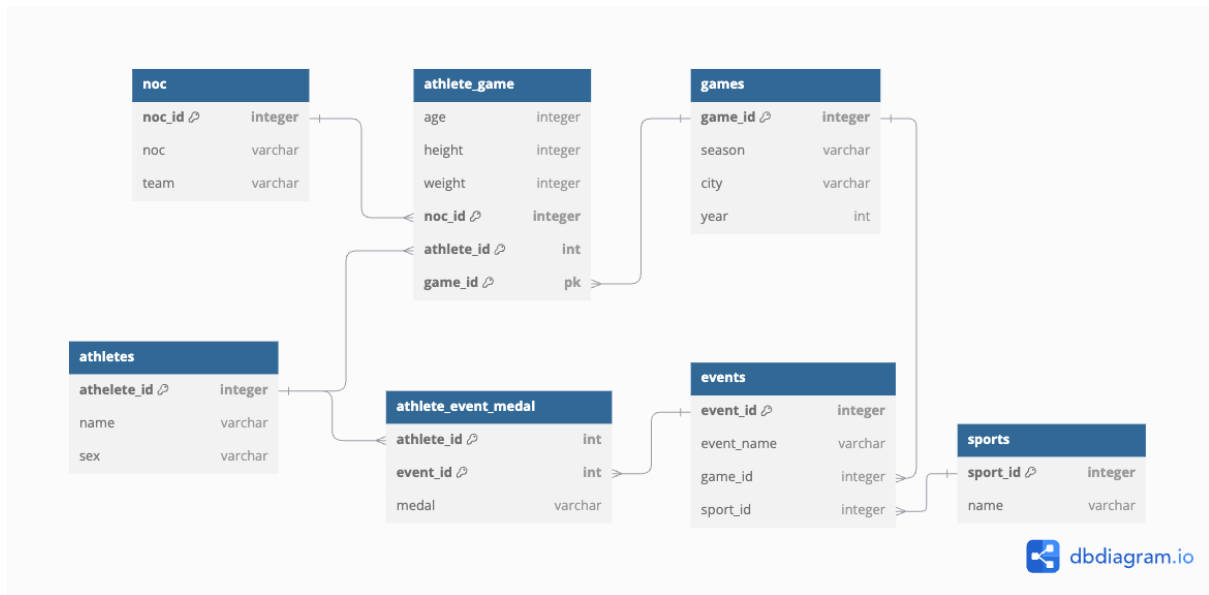
Relationships as depicted by the ERD

The cardinality seen in the E/R diagram above is elaborated in the table below.

Noc to Athlete_game	1:M	One NOC can be represented by many Athlete games but one Athlete game can only have one NOC.
Game to Athlete_game	1:M	One Athlete Game can only belong to one Game but one Game can have many Athlete Games.
Game to Event	1:M	One Game have many Events.
Sport to Event	1:M	One Sport can have many Events but one Event can only belong to one Sport.
Athlete to Athlete_game	1:M	One Athlete can appear at many Athlete_games but one Athlete_game can only refer to one Athlete.
Athlete to Medal	1:M	One Athlete can have many Medals but one Medal only belongs to one Athlete.
Medal to Event	1:M	One Medal belongs to one Event but one Event can have multiple medals.

2.2 Relational Database Schema

A relational database schema is created to serve as a blueprint of how the entities above and their relationships will be structured within the database.



The schema consists of seven tables, as translated from the seven entities in the ER diagram. The schema defines primary and foreign keys in bold, which facilitates efficient data retrieval.

2.3 BNFC Analysis

Boyce-Codd Normal Form (BCNF) is a normal form used in database normalisation that satisfies all conditions of the Third Normal Form (3NF). In BCNF, for any functional dependency defined as $(A \rightarrow B)$, A should be either the super key or the candidate key. [14] Identification of the primary key and functional dependencies is necessary to check if a table is in this form.

Analysis of dependencies as justification for BCNF

Table	Functional Dependency	Superkey
NOC	$\text{noc_id} \rightarrow \text{noc}$ $\text{noc_id} \rightarrow \text{team}$	Primary key NOC is a superkey.
Athletes	$\text{athlete_id} \rightarrow \text{name}$ $\text{athlete_id} \rightarrow \text{sex}$	Primary key athlete_id is a superkey.
Athlete_Game	$\text{athlete_id} + \text{game_id} + \text{noc_id} \rightarrow \text{age}$ $\text{athlete_id} + \text{game_id} + \text{noc_id} \rightarrow \text{height}$ $\text{athlete_id} + \text{game_id} + \text{noc_id} \rightarrow \text{weight}$ $\text{athlete_id} + \text{game_id} + \text{noc_id} \rightarrow \text{noc_id}$	Composite keys (athlete_id, game_id, noc_id) is a superkey.
Athlete_Event_Medal	$\text{athlete_id} + \text{event_id} \rightarrow \text{medal}$	Composite key (athlete_id, event_id) is a superkey.
Games	$\text{game_id} \rightarrow \text{season}$ $\text{game_id} \rightarrow \text{city}$ $\text{game_id} \rightarrow \text{year}$	Game_id is a superkey.
Events	$\text{event_id} \rightarrow \text{event_id}$	Event_id is a superkey.

	$\text{event_id} \rightarrow \text{event_name}$ $\text{event_id} \rightarrow \text{game_id}$ $\text{event_id} \rightarrow \text{sport_id}$	
Sports	$\text{sport_id} \rightarrow \text{name}$	Sport_id is a superkey.

The BCNF principle mandates that every non-trivial functional dependency defined as ($A \rightarrow B$), must have A as a superkey. Evaluation of the functional dependencies and the superkeys revealed that all tables satisfy this condition.

1. 'noc' table: The 'noc_id' is the primary key and uniquely identifies each Olympic committee. This also prevents duplicate entries, as the primary key ensures data integrity and uniqueness.
2. 'athletes' table: The 'athlete_id' is the primary key and determines the name and sex of the athlete.
3. 'sports' table: The 'sport_id' field uniquely identifies each sport. This streamlines data management.
4. 'events' table: The 'event_id' is a primary key that uniquely identifies each event, which includes the event name, game it is held in and sport that is played at that event.
5. 'games' table: The 'games_id' is a primary key that uniquely identifies each game. The season, city and year fields are dependent on this id.
6. 'athlete_game' table: The composite key of 'athlete_id', 'game_id' and 'noc_id' uniquely identifies each athlete's participation in a specific game representing a particular noc, and to associate their age, height, weight, and NOC with that particular game.
7. 'athlete_event_medal': The composite key of 'athlete_id' and 'event_id' uniquely identifies each athlete's participation at a specific event.

BCNF ensures that business rules expressed in functional dependencies using keys are correctly followed. [15] It also enforces data integrity and reduces risk of anomalies and inconsistencies by removing partial and transitive dependencies. [16]

The transformation of the table from wide format to long format ensures that the database is scalable and flexible. For example, the 'sports' table implemented as a long-table format ensures that the database can scale up and adapt to new sports without a redesign of the database. [17]

However, performing analysis of the data, as seen in Section 4, requires the tables to be denormalised and joined together.

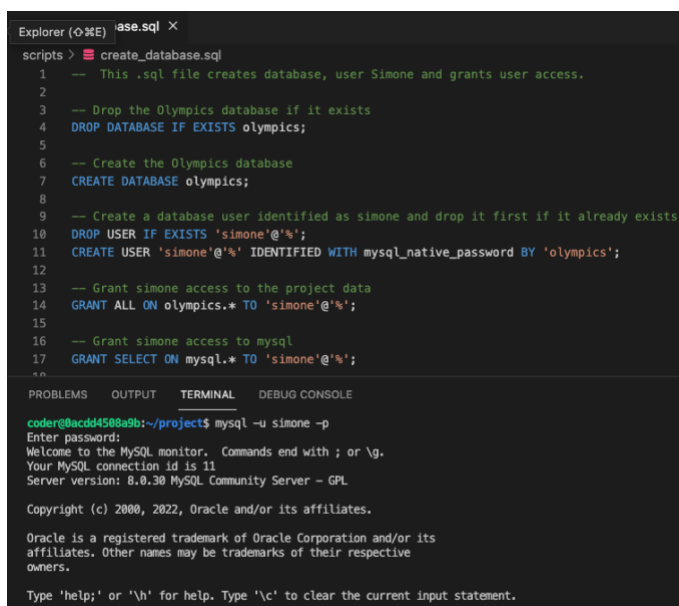
3. Database creation

The database is constructed on a local machine, before it is uploaded onto Coursera Labs. To populate the database with data, a series of SQL scripts was implemented while adhering to best practices in database management.

Extract, transform and load (ETL) is the process of extracting data, cleaning and converting it into a single standard format, and loading the prepped data into a target database. The ETL pipeline is used in this project to improve its quality by performing data cleansing before data loading. [18] Four SQL files are used in the ETL pipeline for this project.

3.1 User configuration

The first SQL file is written to create the Olympics database and add a new user. User access to a fictional user 'Simone' with the password 'olympics' is configured using the SQL CREATE USER statement, as seen in line 11 in the screen capture below. This user is then granted all permissions to the database, as seen in line 14.



The screenshot shows a code editor with a file named 'ase.sql'. The script contains SQL commands to create a database, a user, and grant permissions. Below the script, a terminal window shows the execution of the script using the 'mysql -u simone -p' command. The terminal output shows the MySQL prompt, the password prompt, and the successful execution of the script.

```
Explorer (🔍) ase.sql ✕
scripts > create_database.sql
1  -- This .sql file creates database, user Simone and grants user access.
2
3  -- Drop the Olympics database if it exists
4  DROP DATABASE IF EXISTS olympics;
5
6  -- Create the Olympics database
7  CREATE DATABASE olympics;
8
9  -- Create a database user identified as simone and drop it first if it already exists
10 DROP USER IF EXISTS 'simone'@'%';
11 CREATE USER 'simone'@'%' IDENTIFIED WITH mysql_native_password BY 'olympics';
12
13 -- Grant simone access to the project data
14 GRANT ALL ON olympics.* TO 'simone'@'%';
15
16 -- Grant simone access to mysql
17 GRANT SELECT ON mysql.* TO 'simone'@'%';
18
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
coder@0acdd4508a9b:~/project$ mysql -u simone -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 8.0.30 MySQL Community Server - GPL


Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

The SQL file is executed with the *MySQL Command*: “*SOURCE create_database SQL*”, which executes the SQL file and creates the user and the database.

As seen in the Coursera Lab terminal above, the user is successfully configured. The username and password is saved as a .env file and loaded into the process using the DotEnv module, as seen below.



```
.env
1 HOST=localhost
2 USER_NAME=simone
3 PASSWORD=olympics
4 DATABASE=olympics

JS index.js
24 global.db = mysql.createConnection({
25   host: env.parsed.HOST,
26   user: env.parsed.USER_NAME,
27   password: env.parsed.PASSWORD,
28   database: env.parsed.DATABASE,
29 });
```

Storing the credentials as environment variables and using the DotEnv library ensures the security of the database connections, as sensitive information are kept secure and not included in the source code served to the client. Preventing unauthorised access and keeping credentials confidential also complies with best practices for database management and application development.

3.2 Creation of table to store raw data

The first SQL file stores the raw data as a wide format into a table called 'denormalised'. All create commands are written into a SQL file, which is then called by the MySQL console in Coursera Lab's terminal.

```
CREATE TABLE denormalised (
  id INT,
  name VARCHAR(128),
  sex VARCHAR(64),
  age INT,
  height INT NULL,
  weight INT NULL,
  team VARCHAR(64),
  noc VARCHAR(64),
  games VARCHAR(128),
  year INT,
  season VARCHAR(128),
  city VARCHAR(64),
  sport VARCHAR(64),
  event VARCHAR(128),
  medal VARCHAR(64)
);
```

Each attribute in the table corresponds to one row in the original CSV file.

Using SQL, the data is loaded into the 'denormalised' table.

```
LOAD DATA INFILE '/home/coder/project/scripts/athletes_events_cleaned.csv'
INTO TABLE denormalised
FIELDS TERMINATED BY ','
ENCLOSED BY ''
```

```

LINES TERMINATED BY '\n'
IGNORE 1 ROWS
(id, name, sex, @age, @height, @weight, team, noc, games, @year, season, city,
sport, event, medal)
SET age = NULLIF(@age,"),
    height = CASE
        WHEN @height = " OR @height = 'N' THEN NULL
        ELSE @height
    END,
    weight = NULLIF(@weight, ""),
    year = NULLIF(@year, "");

```

The LOAD DATA statement reads the rows from the CSV file, [19] terminating the rows by ‘\n’ and separating each field by a comma. The first row, which is the header, is ignored. As NULL values are signified by empty strings in CSV files, a CASE statement must be made to handle potential invalid statements. The ‘@’ operator in front of potential invalid field declares these parameters as variables. If the input parameters are empty strings, it will be replaced with NULL using the NULLIF function. [20]

This file is ran with the *MySQL Command*: “*SOURCE load_denormalised.sql*”.

3.3 Create Tables

The third SQL file creates the seven tables defined in Section 2 of this report and defines the primary and foreign keys of each table.

SQL commands	Description
CREATE TABLE noc (noc_id int PRIMARY KEY AUTO_INCREMENT, noc varchar(16), team varchar(65));	Creates a table “noc” with attributes noc, team and noc_id (primary key).
CREATE TABLE athletes (athlete_id int PRIMARY KEY, name varchar(255), sex varchar(16));	Creates a table “athletes” with attributes name and sex and athlete_id (primary key).
CREATE TABLE games (game_id int PRIMARY KEY AUTO_INCREMENT, season varchar(16), city varchar(64), year int);	Creates a table “games” with attributes season, city, year and game_id (primary key).

CREATE TABLE sports (sport_id int PRIMARY KEY AUTO_INCREMENT, name varchar(255));	Creates a table “sports” with attributes name and sport_id (primary key).
CREATE TABLE events (event_id int PRIMARY KEY AUTO_INCREMENT, event_name varchar(255), sport_id int, game_id int, FOREIGN KEY (sport_id) REFERENCES sports(sport_id), FOREIGN KEY (game_id) REFERENCES games(game_id));	Creates a table “events” with attributes event_name, sport_id, game_id and event_id (Primary key). Sport_id and game_id are foreign keys that reference the Sports and Games table respectively.
CREATE TABLE athlete_game (athlete_id int, game_id int, age int, height int, weight int, noc_id int, PRIMARY KEY (athlete_id, game_id, noc_id), FOREIGN KEY (noc_id) REFERENCES noc(noc_id), FOREIGN KEY (athlete_id) REFERENCES athletes(athlete_id), FOREIGN KEY (game_id) REFERENCES games(game_id));	Creates a junction table “athlete_game” with attributes athlete_id, game_id, age, height, weight and noc_id. The composite key athlete_id, game_id and noc_id uniquely identifies an athlete’s participation in a game. The keys noc_id, athlete_id and game_id references the Noc, Athletes and Games tables respectively.
CREATE TABLE athlete_event_medal (athlete_id int, event_id int, medal varchar(16), PRIMARY KEY (athlete_id, event_id), FOREIGN KEY (athlete_id) REFERENCES athletes(athlete_id), FOREIGN KEY (event_id) REFERENCES events(event_id));	Creates a junction table “athlete_event_medal” with attributes athlete_id, event_id, medal. The composite key of athlete_id and event_id uniquely identifies an athlete’s participation at a single event. The athlete_id and event_id references the Athlete and Events table respectively.

Data types

- INT: An integer is used for unique identifiers and for numerical columns such as age, weight and height.
- VARCHAR(16): Selected for fields with shorter strings as this length allows for a smaller space allocation
- VARCHAR(64): Selected for fields with strings of medium length
- VARCHAR(255): Used for fields with long strings such as event names.

This file is ran with the *MySQL Command*: “*SOURCE create_tables.sql*”.

3.4 Populate tables

In the fourth SQL file, we load the data from the ‘denormalised’ table into the main tables defined in the previous section.

Table	Command
Noc	INSERT INTO noc (noc, team) SELECT DISTINCT noc, team FROM denormalised;
Athletes	INSERT INTO athletes (athlete_id, name, sex) SELECT DISTINCT id, name, sex FROM denormalised;
Games	INSERT INTO games (season, city, year) SELECT DISTINCT season, city, year FROM denormalised;
Sports	INSERT INTO sports (name) SELECT DISTINCT sport FROM denormalised;
Events	INSERT INTO events (event_name, sport_id, game_id) SELECT DISTINCT d.event, s.sport_id, g.game_id FROM denormalised d INNER JOIN sports s ON d.sport = s.name INNER JOIN games g ON d.year = g.year AND d.season = g.season AND d.city = g.city;
Athlete_game	INSERT INTO athlete_game (athlete_id, game_id, age, height, weight, noc_id) SELECT DISTINCT a.athlete_id, g.game_id, d.age, d.height, d.weight, n.noc_id FROM denormalised d INNER JOIN athletes a ON d.id = a.athlete_id INNER JOIN games g ON d.year = g.year AND d.season = g.season AND d.city = g.city INNER JOIN noc n ON d.noc = n.noc AND d.team = n.team;
Athlete_event_medal	INSERT INTO athlete_event_medal (athlete_id, event_id, medal) SELECT DISTINCT a.athlete_id, e.event_id, d.medal FROM denormalised d

	<pre>INNER JOIN athletes a ON d.id = a.athlete_id INNER JOIN events e ON e.sport_id = (SELECT sport_id FROM sports s WHERE s.name = d.sport) AND e.game_id = (SELECT game_id FROM games g WHERE g.year = d.year AND g.season = d.season AND g.city = d.city) AND e.event_name = d.event;</pre>
--	--

This file is run against the MySQL database with the *MySQL Command*: “*SOURCE populate_data.sql*”.

Stage 4: Create a Simple Web Application

This section covers how a simple web application is created to provide an interface with which users can interact with the data. The application is designed with a simple interface using the Bootstrap frontend toolkit. [21]

Index Page

The main page serves as the gateway to the different endpoints. The endpoints are displayed in a table, with each row dedicated to a endpoint that answers a specific research question, as defined in Section 1.

CM3010: Database and Advanced Data Techniques [Midterms]

Exploring the Olympics

The table below presents a series of research questions addressed within this project. The questions are ordered in increasing complexity, with more sophisticated SQL statements that delve deeper into the Olympics dataset.

#	Question	Summary	Advanced SQL used	Link
Simple queries				
1	How many sports were there at each game?	The recent 2024 Paris Olympics featured a few new sports - among which are breaking (street dance), skateboarding and sport climbing. This research question aims to investigate how the number of sports featured has changed over time.	Inner Join, Group by and Having clause	Link
2	What is the average age of athletes that won medals?	Since the Olympics test physical endurance, speed and skill, it is plausible that Olympians peak when they are in their youth. This research question tests the hypothesis that an average athlete that places on the podium is below the age of 30.	Aggregation with Avg(), Inner Join and filtering with the WHERE clause	Link
3	How old is the oldest Olympian?	The second query revealed that the average age of a Olympian with a medal is 26.5 years old. This prompts the subsequent research question - who was the oldest Olympian to compete between 2010 and 2016, and which sport did they participate in? This investigation sheds light on the extent to which age may be a less significant factor in certain Olympic sports.	Left Joins, Aggregation with Max() and filtering with Where clause	Link
Queries that combine multiple advanced SQL techniques				
4	Which NOC won the most bronze, silver and gold medals across all four games?	This query finds the countries with the most accolates across all four games. Identifying these countries would aid in a study into the infrastructure that a country must provide in order to have a thriving sports scene.	Aggregation with Sum, Ordering with Case keyword	Link

Each row includes:

- **Summary:** A brief background of what inspires this query that gives users an understanding of the scope and relevance of the query before initiating it.
- **Advanced SQL used:** A list of the advanced SQL techniques that are used to construct this query
- **Link:** A button that executes the query and brings the user to a page that displays the data. The page will contain information that answers the query.

The web application uses a three-tier architecture. The presentation tier is developed with HTML and the Bootstrap framework. The application tier is built on Node.JS and Express, which processes the requests the clients make and performs business logic. The data tier is implemented with MySQL, where the data is stored across seven tables.

User interaction

1. The user initiates the session with the command “node index.js”. A connection to the MySQL database is made and a GET request is sent to retrieve the index page. The server processes the request and serves the index page. The html page is rendered with the EJS templating engine. [22]
2. Query: The user selects a research question to investigate and clicks the link, which sends the relevant GET request to the server. (Eg. /female_most_medals)
3. Data retrieval: The server makes translates the interaction into a SQL query, which is executed against the MySQL database. The code snippet below shows how a GET request to the ‘/games_no’ endpoint is handled. The SQL query is ran against the database and the result is rendered by a html file with the help of the EJS templating engine.

```

77 // Question One: How many sports were there at each game?
78 app.get("/games_no", (req, res) => {
79     const sql = `SELECT g.season, g.city, g.year, count(sport_id) AS sport_count
80     FROM games g
81     INNER JOIN events e
82     ON g.game_id = e.game_id
83     GROUP BY g.game_id
84     ORDER BY g.year`;
85     db.query(sql, function (err, result) {
86         if (err) throw err;
87         res.render("games_no", {
88             data: result,
89         });
90     });
91 });

```

4. Display of results: After the query is processed successfully, the results is returned to the server. The data is then rendered on a new HTML page with EJS and served back to the client.

Research Questions, SQL Statements and Results

Each research question is addressed through an SQL query. The questions are ordered in increasing complexity, accompanied by more sophisticated SQL statements.

Research Question One: Number of sports

Question: How many sports were there at each of the four games?

Background: The recent 2024 Paris Olympics featured a few new sports – such as breaking (street dance), skateboarding and sport climbing. This research question aims to investigate how the number of sports featured has changed over time and whether it is a steady increase.

SQL Query:

```
SELECT g.season, g.city, g.year, count(sport_id) AS sport_count
FROM games g
INNER JOIN events e
ON g.game_id = e.game_id
GROUP BY g.game_id
ORDER BY g.year;
```

Advanced SQL used: Inner Join, Group by and Having clause

This query extracts information about the number of sports at each Olympic Games by joining the Games and Events tables and grouping the resultant table by Game. The results are then ordered chronologically.

Result page: The result page shows a table with the count of sports at each game.

Game	Sport Count
2010 Vancouver (Winter)	86
2012 London (Summer)	302
2014 Sochi (Winter)	98
2016 Rio de Janeiro (Summer)	306

Analysis: From the table below, it appears that there is an increase of sports from 2010 to 2014 (86 to 98) and 2012 to 2016 (302 to 306). Evidently, games have been added to newer renditions. This collaborates initial research into this domain, which showed that new sports are introduced and featured as special editions at each Game. [23]

Research Question Two: Average Age of Olympians

Question: What is the average age of athletes that won medals?

Background: Since the Olympics test physical endurance, speed and skill, it is plausible that Olympians achieve peak performance in their youth. This research question tests the hypothesis that the average age of athletes who achieve a podium finish is below 30 years.

SQL Query:

```
SELECT AVG(ag.age) AS age
FROM athlete_game ag
INNER JOIN athlete_event_medal aem
ON ag.athlete_id = aem.athlete_id
WHERE aem.medal != 'No medal';
```

Advanced SQL used: Aggregation with Avg(), Inner Join and filtering with the WHERE clause

This query joins the athlete_event_model and athlete_game tables, filters for medal-winning athletes, and then calculates the average.

Result page: The result page shows the results of the calculation.

The average age of Olympians that won medals is : **26.5** years old.

Analysis: Based on the results, it is clear that Olympians who won medals had an average age of 26.5 years. This finding implies that younger Olympians may possess a higher probability of securing medals.

Research Question Three: An age-old question

Question: How old is the oldest Olympian?

Background: The second query revealed that the average age of an Olympian with a medal is 26.5 years old. This prompts the subsequent research question - who was the oldest Olympian to compete between 2010 and 2016, and which sport did they participate in? This investigation sheds light on the extent to which age may be a less significant factor in certain Olympic sports.

SQL Query:

```
SELECT a.athlete_id, a.name, a.sex, ag.age, ag.height, e.event_name
FROM athletes a
INNER JOIN athlete_game ag
ON a.athlete_id = ag.athlete_id
LEFT JOIN athlete_event_medal aem
ON a.athlete_id = aem.athlete_id
LEFT JOIN events e
```

```
ON aem.event_id = e.event_id
WHERE age = (SELECT MAX(age) from athlete_game);
```

Advanced SQL used: Left Joins, Aggregation with Max() and filtering with Where clause

The query joins three tables – athletes, athlete_game and athlete_event_medal and returns the record that holds the maximum age from the athlete_game table.

Result page:

Name	Sex	Age	Height (cm)	Event Name	Link to Games
Hiroshi Hoketsu	M	71	168	Equestrianism Mixed Dressage, Individual	Athlete details

Analysis:

The results showed that Mr Hiroshi Hoketsu was the oldest participant across the four games. He competed in the Equestrianism Mixed Dressage event at 71-years-old. This webpage also includes a link, which brings the user to all the events that the athlete has participated in.

The Olympic Career of Hiroshi Hoketsu

The table below shows all the events that the athlete participated in, in the 2010 to 2016 Olympic games.

Game competed in:	Age	Height (cm)	Weight (kg)	Team	Event	Medal
London 2012 (Summer)	71	168	62	Japan	Equestrianism Mixed Dressage, Individual	No medal

The link brings the user to a second webpage, which discloses the singular event Mr Hokuetsu participated in. He represented Japan at the London Games but did not win a medal.

This finding suggests that Japan might be a country that supports and funds older athletes, and that Equestrianism is a sport that is age friendly.

Research Question Four: NOC and accolades

Question: Which NOC won the most bronze, silver and gold medals across all four games?

Background: The objective of this research question is to ascertain the countries with the most accolades across all four games. Identifying these nations will be instrumental in conducting a study into the essential infrastructure that a country must establish to foster and nurture a flourishing sports environment.

SQL Query:

```
SELECT n.noc, g.city, g.year, g.season,  
       SUM(CASE WHEN aem.medal = 'Gold' THEN 1 ELSE 0 END) AS gold,  
       SUM(CASE WHEN aem.medal = 'Silver' THEN 1 ELSE 0 END) AS silver,  
       SUM(CASE WHEN aem.medal = 'Bronze' THEN 1 ELSE 0 END) AS bronze,  
       COUNT(aem.medal) AS total  
FROM noc n  
JOIN athlete_game ag ON n.noc_id = ag.noc_id  
JOIN games g ON ag.game_id = g.game_id  
LEFT JOIN athlete_event_medal aem ON ag.athlete_id = aem.athlete_id  
GROUP BY n.noc, g.game_id, g.year, g.season  
ORDER BY total DESC;
```

Advanced SQL used: Aggregation with Sum, Ordering with Case keyword.

The query performs a summation of the medals and group the results by the NOC and the game.

Result page:

NOC	Game	Gold	Silver	Bronze	Total medals
USA	Rio de Janeiro 2016 (Summer)	211	79	96	1003
USA	London 2012 (Summer)	204	88	78	973
GBR	London 2012 (Summer)	95	58	59	913
BRA	Rio de Janeiro 2016 (Summer)	46	15	14	756
GER	Rio de Janeiro 2016 (Summer)	72	59	80	740
GER	London 2012 (Summer)	63	42	42	735
GBR	Rio de Janeiro 2016 (Summer)	98	72	54	713
AUS	Rio de Janeiro 2016 (Summer)	33	54	55	712
RUS	London 2012 (Summer)	73	54	69	708
AUS	London 2012 (Summer)	23	54	71	704
FRA	Rio de Janeiro 2016 (Summer)	33	70	30	684
CHN	Rio de Janeiro 2016 (Summer)	80	47	47	656
CHN	London 2012 (Summer)	76	56	43	636

Analysis: This table shows the medals of each NOC aggregated over the four Olympic games. This shows us which country boasts the highest number of medallists and will facilitate an investigation into how a country can best support its athletes.

Research Question Five: Female athletes and Gold medals

Question: Which female athletes accumulated the most gold medals?

Background: This query aims to identify the most successful female Olympians by finding those that accumulated the most gold medals. This query highlights the individuals that excelled at their sports in the four games and lays the groundwork for further investigation in any commonalities that these athletes share that aided them in their success.

SQL Query:

```
SELECT a.athlete_id, a.name, count(m.athlete_id) AS gold_medal_count
FROM athlete_event_medal m
INNER JOIN athletes a
ON a.athlete_id = m.athlete_id
WHERE (m.medal = 'Gold' AND a.sex = 'F')
GROUP BY a.athlete_id
ORDER BY gold_medal_count DESC
LIMIT 10;
```

Advanced SQL used: Inner Joins, filtering with Where Clause, Group by and Order by

This query joins two tables, athletes and athlete_event_medal to calculate the number of gold medals each female athlete won. The top 10 athletes are then returned.

Result page:

Name	Medal Count
Marit Bjrgen	6
Kathleen Genevieve "Katie" Ledecky	5
Allyson Michelle Felix	5
Melissa Jeanette "Missy" Franklin	5
Danuta Kozk	5
Allison Rodgers Schmitt	4
Svetlana Alekseyevna Romashina	4
Nataliya Sergeevna Ishchenko	4
Simone Arianne Biles	4
Laura Rebecca Trott (-Kenny)	4

Analysis:

It is evident that Ms Marit Bjrgen won the most medals across the four games.

Research Question Six: The Young Olympian

Question: How old is the youngest Olympian and what country do they represent?

Background: The third query established the oldest age of the Olympian as 71 years-old. This research question shifts the focus to the opposite end of the spectrum - to identify the youngest Olympian and the nations they represent. This information can illuminate evolving demographics in participation of youths, as well as provide insight into any traits that these young athletes share.

SQL Query:

```
SELECT a.athlete_id, a.name, a.sex, ag.age, ag.height, n.team
FROM athletes a
INNER JOIN athlete_game ag
ON a.athlete_id = ag.athlete_id
LEFT JOIN noc n
ON ag.noc_id = n.noc_id
WHERE age = (SELECT MIN(age) from athlete_game);
```

Advanced SQL used: Inner Join, Left Join, Aggregation with Min() and filtering with Where clause.

This query combines three tables – athletes, athlete_game and noc. The record with the lowest age is returned.

Result page:

Name	Sex	Age	Height	Country	Link to games
Ana Iulia Dascl	F	13	183	Romania	Athlete details
Adzo Rebecca Kpossi	F	13	158	Togo	Athlete details
Gaurika Singh	F	13	155	Nepal	Athlete details

Analysis:

The table below shows the youngest Olympians across the Games, who are thirteen year olds. When delving into their sport (by clicking on the rightmost column), it is evident that all three athletes are swimmers.

The Olympic Career of Ana Iulia Dascl

The table below shows all the events that the athlete participated in, in the 2010 to 2016 Olympic games.

Game competed in:	Age	Height (cm)	Weight (kg)	Team	Event	Medal
Rio de Janeiro 2016 (Summer)	13	183	60	Romania	Swimming Women's 100 metres Freestyle	No medal

This is interesting as it shows that swimmers typically start competing when they are young.

Research Question 7: The Average Olympian

Question: What is the average height and weight of athletes in a sport?

Background: From the previous query, it is evident that the youngest athletes are all 13 years old. One athlete, Ana Inulia Dascl, has a height of 183cm, which is peculiar for her age. This research question asks for the average height and weight of athletes in a sport. Clicking on the query from the index page directs to a page with the sports listed. This page was rendered with the SQL query 'SELECT * FROM SPORTS'.

All sports

The table below shows the list of sports that were played across all four games. Click on the link to get the average average statistics of athletes participating in that particular sport.

ID	Name	Link
1	Judo	Average athlete details
2	Ice Hockey	Average athlete details
3	Weightlifting	Average athlete details
4	Athletics	Average athlete details
5	Boxing	Average athlete details

Each sport has a different link that directs to a second page with the appropriate statistic, this is given by the query below.

SQL Query:

```
SELECT s.name, AVG(ag.height), AVG(ag.weight), AVG(ag.age)
FROM athlete_game ag
INNER JOIN events e ON ag.game_id = e.game_id
INNER JOIN sports s ON e.sport_id = s.sport_id
WHERE s.sport_id = ?;
```

Advanced SQL used: Left Join, Right Join, Aggregation with Avg() and filtering with Where clause with different values on the same query.

This query involves a combination of three tables: athlete_game, events and sports. The placeholder, represented by '?' within the where clause is substituted with the actual sport unique identifier upon the execution of the query. This query then proceeds to compute the average values for height, weight and age.

Result page:

Average Olympian in Judo

The table below shows the statistics of the average athlete in Judo. This statistics elucidate what a typical athlete in this sport would look like.

Name of Sport	Height (cm)	Weight (kg)	Age
Judo	176.9	72.1	26.3

Analysis:

Each page describes the typical physical traits of an athlete in that sport.

Research Question Eight: The Cross-season Olympian

Question: Were there athletes that participated in both Summer and Winter games and won medals?

Background: Given the distinct nature of the Summer and Winter Games, the conventional assumption is that medalling in both iterations of the Olympic Games is an unattainable feat. This research question challenges this assumption.

SQL Query:

```
WITH pro_athletes AS (SELECT a.athlete_id, a.name, a.sex
FROM athletes a
JOIN athlete_game ag ON a.athlete_id = ag.athlete_id
JOIN games g ON ag.game_id = g.game_id
WHERE g.season IN ('Summer', 'Winter')
GROUP BY a.athlete_id, a.name, a.sex
HAVING COUNT(DISTINCT g.season) = 2)

SELECT p.name, p.sex, e.event_name, g.season, g.city, g.year, aem.medal,
p.athlete_id
FROM athlete_event_medal aem
INNER JOIN pro_athletes p
ON aem.athlete_id = p.athlete_id
LEFT JOIN events e
ON aem.event_id = e.event_id
LEFT JOIN games g
ON e.game_id = g.game_id
WHERE aem.medal != 'No medal';
```

Advanced SQL used: Decorrelated Common Table Expression (CTE) query, Aggregation with COUNT(), Group By, Distinct and Having clause

This query retrieves information about the professional athletes that participated in both Summer and Winter Olympics.

The 'WITH' clause defines a Common Table Expression (CTE) named `pro_athletes`, that is a temporary set that contains the id, name and sex of the athletes who have participated in both Summer and Winter Olympics. Then, it retrieves medal information for these athletes, as well as the game and event details.

Result page:

Name	Sex	Event	Game	Medal	Link to Games
Clara Hughes	F	Speed Skating Women's 5,000 metres	Vancouver 2010 (Winter)	Bronze	Athlete details
Georgia Simmerling	F	Cycling Women's Team Pursuit	Rio de Janeiro 2016 (Summer)	Bronze	Athlete details
Laurine van Riessen	F	Speed Skating Women's 1,000 metres	Vancouver 2010 (Winter)	Bronze	Athlete details
Lauryn Chenet Williams	F	Athletics Women's 4 x 100 metres Relay	London 2012 (Summer)	Gold	Athlete details
Lauryn Chenet Williams	F	Bobsleigh Women's Two	Sochi 2014 (Winter)	Silver	Athlete details

Analysis:

Given the distinct nature of the Summer and Winter Games, the conventional assumption is that medalling in both iterations of the Olympic Games is an unattainable feat. This research question overturns this assumption.

Research Question Nine: Goldies

Question: Were there athletes that always ended up on the podium at all their events in London 2012 and won gold medals at both London and Rio 2016?

Background: Athletes improve across games as they get faster, stronger or more skilled. This research question seeks to identify a subset of athletes who placed at every event they participated in, in London, and earned a Gold at the Rio Games. Since the Olympics test physical prowess, it is a testament to their hard work that their skill remain unmatched across games. In the next page, there will be a link for

each athlete that leads to a table of all events they have participated in across the two games.

SQL Query:

```
WITH london_athletes AS (  
  SELECT DISTINCT a.name, ag.athlete_id, ag.game_id  
  FROM athlete_game ag  
  INNER JOIN athletes a ON ag.athlete_id = a.athlete_id  
  INNER JOIN athlete_event_medal aem ON ag.athlete_id = aem.athlete_id  
  WHERE ag.game_id = (  
    SELECT game_id  
    FROM games  
    WHERE year = 2012 AND city = 'London'  
  )  
  AND aem.medal = 'Gold'  
)  
  
SELECT la.athlete_id, la.name  
FROM london_athletes la  
WHERE NOT EXISTS (  
  SELECT 1  
  FROM athlete_game ag2  
  INNER JOIN athlete_event_medal aem2  
  ON ag2.athlete_id = aem2.athlete_id  
  WHERE ag2.athlete_id = la.athlete_id AND ag2.game_id = la.game_id  
  AND aem2.medal IN ('No medal')  
)  
AND EXISTS (  
  SELECT 1  
  FROM athlete_game ag3  
  INNER JOIN athlete_event_medal aem3  
  ON ag3.athlete_id = aem3.athlete_id  
  WHERE ag3.athlete_id = la.athlete_id  
  AND aem3.medal IN ('Gold')  
  AND ag3.game_id = (  
    SELECT game_id  
    FROM games  
    WHERE year = 2016 AND city = 'Rio de Janeiro'  
  )  
)  
);
```

Advanced SQL used: Correlated queries, Exists and Not Exists clause, filtering for more data with different ids

The query identifies athletes who won gold medals in London 2012. Then, the list is filtered to include only those who did not participate in any event without a medal in

Rio 2016 and achieved at least one gold medal in Rio. The 'EXISTS' condition is used to check the existence returned by a subquery. [24]

Result page:

Athlete ID	Name	Link to their games
73	Luc Abalo	Athlete details
832	Nicola Virginia Adams	Athlete details
846	Valerie Kasanita Adams-Vili (-Price)	Athlete details
1017	Nathan Ghar-Jun Adrian	Athlete details
2464	Jo Qesem Ayela Aleh	Athlete details
2497	Artur Aleksanyan	Athlete details
2517	Milan Aleksi	Athlete details
2758	Alison Conte Cerutti	Athlete details
3077	Ida Alstad	Athlete details

Analysis:

Given the distinct nature of the Summer and Winter Games, the conventional assumption is that medalling in both iterations of the Olympic Games is an unattainable feat. This research question overturns this assumption.

Research Question Ten: NOC Participation

Question: Were there any NOCs that participated in the 2010 Vancouver games but not the 2014 Sochi games?

Background: While the Olympics Games endeavour to showcase and celebrate athletes from across the globe, it is undeniable that socio-political factors can impact a nation's decision to send its athletes abroad. [25] Consequently, this research question seeks to determine which countries attended the 2010 Olympic Games in Canada but did not participate in the 2014 Olympic Games in Russia.

SQL Query:

```
SELECT n1.noc_id, n1.noc, n1.team
FROM noc n1
LEFT JOIN athlete_game ag1
  ON n1.noc_id = ag1.noc_id
WHERE ag1.game_id = (SELECT game_id FROM games WHERE year = '2010')
AND n1.noc_id NOT IN (
  SELECT n2.noc_id
  FROM noc n2
  LEFT JOIN athlete_game ag2
    ON n2.noc_id = ag2.noc_id
  WHERE ag2.game_id = (SELECT game_id FROM games WHERE year = '2014')
);
```

Advanced SQL used: Aggregation with Avg(), Inner Join and filtering with the WHERE clause

The query identifies all the NOCs and their participation records for the 2014 Olympics. Then, it filters out the NOCs that were present in the 2014 Olympics. This results in a list of NOCs that were absent from the 2014 Games but were present in 2010. Originally, an Except clause was used. However, Coursera Labs did not support the use of Except clause. Hence, the 'NOT IN' clause is used to indicate the absence of the NOC instead.

Result page:

ID	NOC	Team			
143	ROU	Romania-2	82	RSA	South Africa
198	CZE	Czech Republic-2	162	SEN	Senegal
187	ROU	Romania-1	256	UKR	Ukraine-2
226	CZE	Czech Republic-1	139	PRK	North Korea
50	COL	Colombia	183	AUS	Australia-1
243	UKR	Ukraine-1	226	CZE	Czech Republic-1
183	AUS	Australia-1	226	CZE	Czech Republic-1
143	ROU	Romania-2	256	UKR	Ukraine-2
11	ALG	Algeria	187	ROU	Romania-1
243	UKR	Ukraine-1	33	ETH	Ethiopia
139	PRK	North Korea	198	CZE	Czech Republic-2
226	CZE	Czech Republic-1	243	UKR	Ukraine-1
256	UKR	Ukraine-2	198	CZE	Czech Republic-2
198	CZE	Czech Republic-2	256	UKR	Ukraine-2
82	RSA	South Africa	267	CHN	China-3
243	UKR	Ukraine-1	267	CHN	China-3
57	GHA	Ghana			
152	AUS	Australia-2			
152	AUS	Australia-2			
183	AUS	Australia-1			

Analysis:

The table above lists the NOCs absent from the Sochi Games in 2014. While there are no discernible commonalities, further investigation into these countries may shed light on why they were absent from the 2014 Games, despite their participation in the games four years prior.

Works Cited

- [1 D. Ganesan, "Why are the Olympics so special?," 11 July 2024. [Online]. Available:
] <https://www.straitstimes.com/sport/why-are-the-olympics-so-special>. [Accessed 25 December 2024].

- [2 Channel News Asia, "Gymnastics-Biles soars to third gold medal of Paris Olympics
] with vault triumph," 3 August 2024. [Online]. Available:
<https://www.channelnewsasia.com/sport/gymnastics-biles-soars-third-gold-medal-paris-olympics-vault-triumph-4524821>. [Accessed 25 December 2024].

- [3 F. Karimi, "This sport is making its Olympics debut in Paris. Just don't call it
] breakdancing," 6 August 2024. [Online]. Available:
<https://edition.cnn.com/2024/08/06/sport/breaking-olympic-debut-paris-cec/index.html>. [Accessed 25 December 2024].

- [4 A. Moore, "5 Reasons Why the Olympics Are Important," 1 August 2024. [Online].
] Available: <https://cnr.ncsu.edu/news/2024/08/why-the-olympics-are-important/#:~:text=%E2%80%9CWhat%20makes%20the%20Olympics%20unique,and%20Tourism%20Management%20at%20NC>. [Accessed 25 December 2024].

- [5 R. Griffin, "120 years of Olympic history: athletes and results," 2018. [Online].
] Available: https://www.kaggle.com/datasets/heesoo37/120-years-of-olympic-history-athletes-and-results?select=athlete_events.csv. [Accessed 26 December 2024].

- [6 C. Bumbaca, "Nikki Hiltz, US track Olympian, embraces 'superpower' of being queer
] and running 'free,'" 9 August 2024. [Online]. Available:
<https://www.usatoday.com/story/sports/olympics/2024/08/09/usa-track-athlete-nikki-hiltz-superpower-queer/74733630007/>. [Accessed 26 December 2024].

- [7 Sports Reference, "Sports Reference | Sports Stats, fast, easy, and up-to-date,"
] 2024. [Online]. Available: <https://www.sports-reference.com/>. [Accessed 26 December 2024].

- [8 Dataful, "Sports - Olympic Games: Year-, Country-, Sport- and Athlete-wise Gold,
] Silver, and Bronze Medals Won, since 1896," 2024. [Online]. Available:
<https://dataful.in/datasets/19672/>. [Accessed 26 December 2024].

- [9 International Olympic Committee, "Athletes," 2024. [Online]. Available:
] <https://olympics.com/en/athletes/>. [Accessed 26 December 2024].

[1 International Olympic Committee, "Archived Documents," 2024. [Online]. Available:
0] <https://odf.olympictech.org/project.htm>. [Accessed 26 December 2024].

[1 Creative Commons, "CC0 1.0 Universal," 2024. [Online]. Available:
1] <https://creativecommons.org/publicdomain/zero/1.0/>. [Accessed 26 December
2024].

[1 Geek for Geeks, "Difference between Pandas VS NumPy," 22 July 2024. [Online].
2] Available: <https://www.geeksforgeeks.org/difference-between-pandas-vs-numpy/>.
[Accessed 26 December 2024].

[1 Greg, "What is an associative entity in an ERD?," 28 August 2024. [Online].
3] Available: <https://www.gleek.io/blog/associative-entity-erd>. [Accessed 26
December 2024].

[1 M. Jain, "BCNF in DBMS," 30 Aoruk 2024. [Online]. Available:
4] <https://www.scaler.com/topics/bcnf-in-dbms/>. [Accessed 26 December 2024].

[1 P. Pedamkar, "BCNF," 24 March 2023. [Online]. Available:
5] <https://www.educba.com/bcnf/>. [Accessed 26 December 2024].

[1 LinkedIn community, "What is the Boyce-Codd normal form and how is it used?,"
6] 2024. [Online]. Available: [https://www.linkedin.com/advice/0/what-boyce-codd-
normal-form-how-used-skills-database-engineering](https://www.linkedin.com/advice/0/what-boyce-codd-normal-form-how-used-skills-database-engineering). [Accessed 27 December
2024].

[1 L. Bennett, "Long vs Wide Data Tables," 2024. [Online]. Available:
7] <https://www.thedataschool.co.uk/luke-bennett/long-vs-wide-data-tables/>.
[Accessed 27 December 2024].

[1 IBM, "What is ETL (extract, transform, load)?," 2024. [Online]. Available:
8] <https://www.ibm.com/think/topics/etl>. [Accessed 27 December 2025].

[1 Oracle, "15.2.9 LOAD DATA Statement," 2024. [Online]. Available:
9] <https://dev.mysql.com/doc/refman/8.4/en/load-data.html>. [Accessed 28 December
2024].

[2 Geeks for Geeks, "MySQL NULLIF() Function," 2024. [Online]. Available:
0] https://www.w3schools.com/sql/func_mysql_nullif.asp. [Accessed 28 December
2024].

[2 Bootstrap, "Build fast, responsive sites with Bootstrap," 2024. [Online]. Available:
1] <https://getbootstrap.com/>. [Accessed 29 December 2024].

[2] M. Eernisse, "Embedded JavaScript templating," 2024. [Online]. Available:
2] <https://ejs.co/>. [Accessed 29 December 2024].

[2] S. Darbhamulla, "Looking back, looking forward: The Olympics of adding new
3] sports," 12 August 2024. [Online]. Available:
<https://www.thehindu.com/sport/olympics/new-olympic-sports-and-events-additions/article68505265.ece>. [Accessed 30 December 2024].

[2] Geek for Geeks, "SQL Exists," 10 December 2024. [Online]. Available:
4] <https://www.geeksforgeeks.org/sql-exists/>. [Accessed 30 December 2024].

[2] G. Goorno, "Should the Olympics ignore international conflict?," 30 September
5] 2024. [Online]. Available: <https://www.tuftsdaily.com/article/2024/09/should-the-olympics-ignore-international-conflict>. [Accessed 29 December 2024].