INFS1200 / INFS7900 Assignment 2

Code Due: 6 October 2023 @ 4:00 PM AEST
(Oral Assessment to be held in Week 12, 17-20 October 2023)

Weighting: 25%   Version 2.0

Full Name	Student ID (8 digits)
Le Na Ngo	47615427

Overview

The purpose of this assignment is to test your ability to use and apply SQL concepts to complete tasks in a real-world scenario. Specifically, this assessment will examine your ability to use SQL Data Manipulation Language to return specific subsets of information which exist in a database and Data Definition Language to create new relational schema. The assignment is to be completed **individually.**

Submission

Assignment 2 is made up of two parts. **Part 1** will be submitted through an electronic marking tool called Gradescope, which will also be used for providing feedback. **Part 2** is an oral assessment that will be completed during an in-person meeting with a tutor during a tutorial or practical session in Week 12 (after your Gradescope submission). Details below:

Part 1: Answer the questions on this task sheet and submit them through an electronic marking tool called Gradescope. For this assignment, you will need to submit two types of files to the portal:

- **Query Files**:
  - For each question in Sections A, B and C, you are required to submit a separate *.sql* or *.txt* file which contains your SQL query solution for that question (submit only one of these files; if you submit both, the *.sql* file will be graded).
  - Each file should only contain the SQL query(s) and no additional text.
  - Each file should be named as per the *Filename* description in the question.
  - The total number of queries allowed to be run per question is also specified in each question's description.
  - When submitting files to the autograder, select all of your *.txt* or *.sql* files as well as your *.pdf* file.

- **Assignment PDF**:
  - Insert your answers for all Sections A-D into the template boxes on this assignment task sheet where appropriate, then export this document to a PDF and also upload it to the Gradescope autograder portal.
  - Only Section D will be hand-marked from your PDF submission, however this is also a backup for Sections A, B and C in case of autograder failure.
  - **UPDATES:**
    - For Sections A, B and C, include a screenshot of the output of your query for each question in the space provided. Use your zones to generate the output. For queries with a returning relation of more than 10 tuples, you can use the LIMIT 10 clause to only capture the first 10 tuples of the table.
    - **Please name your file 'Assignment_2.pdf'.** Please do not alter the format or layout of this document in anyway and ensure the name and SID boxes are completed.

Part 2 is an oral assessment, to verify your understanding of the code you submitted in Part 1 Sections A, B and C.

- This will be an oral critique of your submitted code. In a short meeting with a tutor during Week 12, you will explain the work you have submitted in Part 1 and discuss your choices.
- All oral assessments must be given live and will be recorded by the teaching team for archiving purposes.

Marking

Assignment 2 is worth **25 course marks**, and marking is made up of two parts.

First, the marks available per section of Part 1 are as follows (INFS1200 differs from INFS7900):

	INFS1200	**INFS7900**
Section A – SQL DML (SELECT)	15 marks	14 marks
Section B – SQL DML (UPDATE, INSERT, DELETE)	4 marks	4 marks
Section C – SQL DDL	4 marks	4 marks
Section D – Critical thinking	2 marks	3 marks

Given these available marks, students must also achieve a pass (+/-) in Part 2, the oral critique, to be eligible to pass Assignment 2. Failure in Part 2 will result in your mark being capped at 12.5%.

Grading and autograder feedback

Sections A, B and C of this assignment will be graded via an autograder deployed on Gradescope; however we reserve the right to revert to hand marking using the pdf submission should the need arise. Specifically, your assignment may be graded against several data instances, which may include a simple (and small) data instance, a large data instance or an instance containing curated edge cases. The correctness of your queries will be judged by comparing your queries' return values to those of our solutions, because there is usually more than one equivalent way to execute a given query.

Note that solutions to each question will be limited to contain a maximum of 3 queries.

When you submit your code, the autograder will provide you with two forms of immediate feedback:

- **File existence and compilation tests**: Your code will be checked to see if it compiles correctly. If it fails one or more compilation test, the errors returned by the autograder will help you debug. Note that code that fails to compile will receive 0 marks. No marks are given for passing the compilation tests.
- **Simple instance data tests:** The autograder will return your degree of success on the simple data instance, so that you can judge your progress (i.e. 90% of simple instance tests passed). Individual test results will not be revealed, and your submission's performance on the more difficult instances will remain hidden until grades are released. Final weightings on the different test instances will also remain hidden until grades are released.

More details will be provided regarding how you can interpret the results of these tests and what it means for your assignment grade during practicals.

Note: Your queries must compile using **MySQL version 8.0**. This is the same DBMS software as is used on your zones. You may use any MySQL function that have been used in class in addition to those specified in the questions. You may also use other MySQL functions not covered in this course to assist with manipulating the data if needed, however please ensure you read the MySQL documentation page first to ensure the functions works as intended.

The final details of the Gradescope autograder will be released closer to the assignment deadline. Note that you will be able to resubmit to the autograder an unlimited number of times before the deadline.

Materials provided:

You will be provided with the database schema and the simple data instance. Because the autograder uses the same DBMS as your zones, you are encouraged to use your zones to develop your assignment answers.

Plagiarism

The University has strict policies regarding plagiarism. Penalties for engaging in unacceptable behaviour range from loss of grades in a course through to expulsion from UQ. You are required to read and understand the policies on academic integrity and plagiarism in the course profile (Section 6.1). If you have any questions regarding acceptable level of collaboration with your peers, please see either the lecturer or your tutor for guidance. Remember that ignorance is not a defence!

In particular, you are permitted to use generative AI tools to help you complete this assessment task. However, if you do, please provide complete copies of your interactions with the AI tool in the space provided at the end of your submission. Please note that if you use generative AI but fail to acknowledge this by attaching your interaction to the end of the assignment, it will be considered misconduct as you are claiming credit for work that is not your own.

Task

For this assignment you will be presented with the simplified schema of an event management application for the Olympics. It captures essential details about countries, sports, athletes, events, and the medals won in these events.

1. **Countries** table lists all the participating countries in the Olympic games. Each country has a unique identifier (CountryID), a name (CountryName), and is associated with a specific geographical region (Region).

2. **Sports** table contains information about various sports played in the Olympics. Each sport has a unique identifier (SportID) and a name (SportName).

3. **Athletes** table records all athletes participating in the games. Each athlete is uniquely identified by an AthleteID, and they also have a name (AthleteName). They are associated with a specific country, represented by the CountryID, which links to the Countries table. Their age is also stored in the database.

4. **Events** table represents the events held in different sports. Each event is uniquely identified by an EventID and is associated with a specific sport, represented by the SportID, which refers to the Sports table and an associated date and ticket price for the event.

5. **Medals** table archives the allocation of medals. Each entry, denoting a medal, is uniquely associated with a particular athlete (AthleteID) and a specific event (EventID), and it documents the category of the medal (MedalType - Gold, Silver, or Bronze) won by the athlete. The AthleteID references the Athletes table, and the EventID refers to the Events table.

6. **Contestants** table contains all contestants in all events. Each entry is uniquely identified by an event (EventID) and an athlete (AthleteID) participating in that event. The EventID refers to the Events table, and the AthleteID references the Athletes table.

Relational Schema:

Countries [<u>CountryID</u>, CountryName, Region]

Sports [<u>SportID</u>, SportName]

Athletes [<u>AthleteID</u>, AthleteName, CountryID, Age]

Events [<u>EventID</u>, SportID, Date, TicketPrice]

Medals [<u>AthleteID, EventID,</u> MedalType]

Contestants [<u>EventID, AthleteID</u>]

Foreign Keys:

Athletes.CountryID references Countries.CountryID

Events.SportID references Sports.SportID

Medals.AthleteID references Athletes.AthleteID

Medals.EventID references Events.EventID

Contestants.EventID references Events.EventID

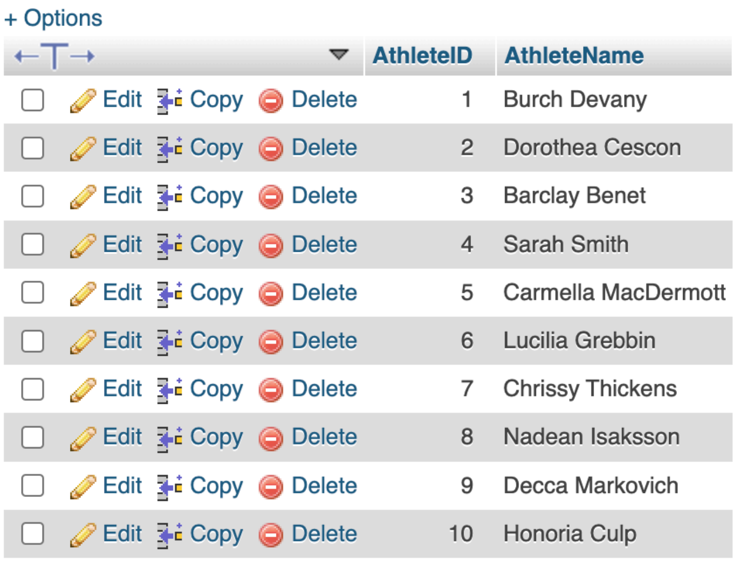
Contestants.AthleteID references Athletes.AthleteID

For this assignment you will be required to write SQL queries to answer to complete the following tasks. Please use the submission boxes provided to record your answers. An example is given below.

Example	
Task	Return the id and name of all athletes.
Explanation	This query should return a table with two columns, one for the id and one for the name of the athletes.

SQL Solution	`SELECT AthleteID, AthleteName`
	`FROM  Athletes`
	`LIMIT 10;`

Output screenshot:



+ Options

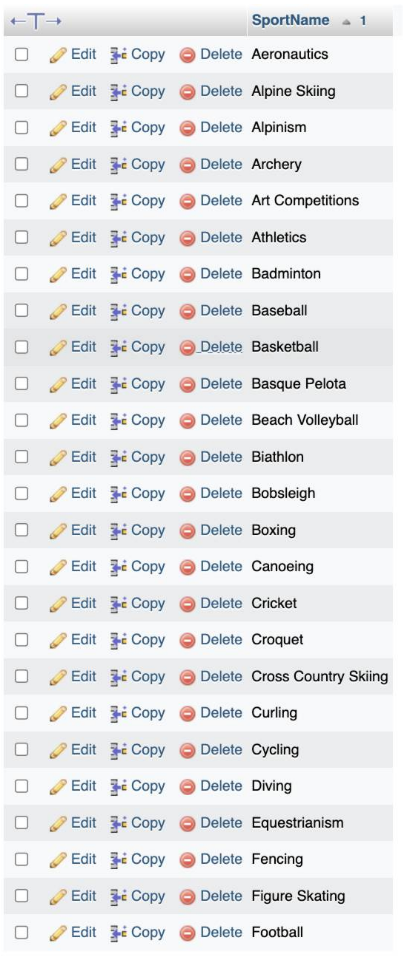
	AthleteID	AthleteName
Edit Copy Delete	1	Burch Devany
Edit Copy Delete	2	Dorothea Cescon
Edit Copy Delete	3	Barclay Benet
Edit Copy Delete	4	Sarah Smith
Edit Copy Delete	5	Carmella MacDermott
Edit Copy Delete	6	Lucilia Grebbin
Edit Copy Delete	7	Chrissy Thickens
Edit Copy Delete	8	Nadean Isaksson
Edit Copy Delete	9	Decca Markovich
Edit Copy Delete	10	Honoria Culp

Section A – SQL DML (SELECT)

Question 1	
Task	Return the names of all sports played at the Olympics (duplicates should not be included), ordered by SportName in alphabetical order.
Filename	*a1.sql* or *a1.txt*

SQL Solution	SELECT DISTINCT SportName
	FROM Sports
	ORDER BY SportName ASC;

Output screenshot:

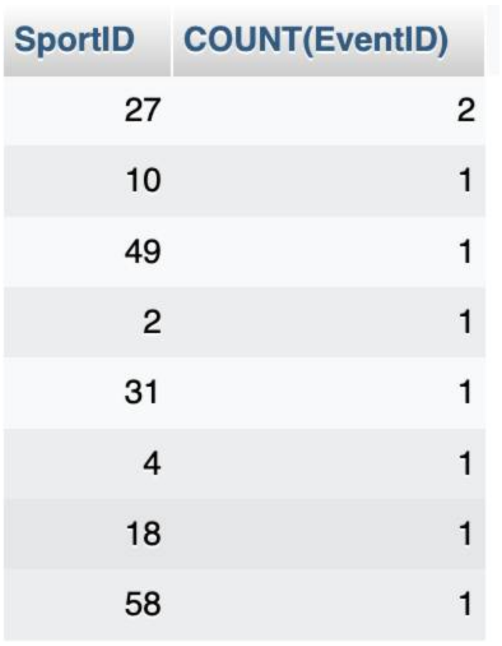


	SportName ▲ 1
Edit Copy Delete	Aeronautics
Edit Copy Delete	Alpine Skiing
Edit Copy Delete	Alpinism
Edit Copy Delete	Archery
Edit Copy Delete	Art Competitions
Edit Copy Delete	Athletics
Edit Copy Delete	Badminton
Edit Copy Delete	Baseball
Edit Copy Delete	Basketball
Edit Copy Delete	Basque Pelota
Edit Copy Delete	Beach Volleyball
Edit Copy Delete	Biathlon
Edit Copy Delete	Bobsleigh
Edit Copy Delete	Boxing
Edit Copy Delete	Canoeing
Edit Copy Delete	Cricket
Edit Copy Delete	Croquet
Edit Copy Delete	Cross Country Skiing
Edit Copy Delete	Curling
Edit Copy Delete	Cycling
Edit Copy Delete	Diving
Edit Copy Delete	Equestrianism
Edit Copy Delete	Fencing
Edit Copy Delete	Figure Skating
Edit Copy Delete	Football

Question 2	
Task	Return the number of events that occurred for each sport during the month of July 2023.
Explanation	This query should return two columns, one for the SportID and one for the number of events that occurred for each sport.
Filename	*a2.sql* or *a2.txt*

SQL Solution	SELECT SportID, COUNT(EventID)
	FROM Events
	WHERE Date >= '2023-07-01' AND Date <= '2023-07-31'
	GROUP BY SportID;

Output screenshot:

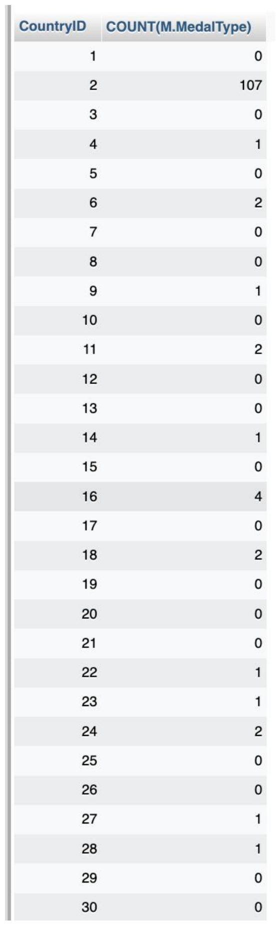


SportID	COUNT(EventID)
27	2
10	1
49	1
2	1
31	1
4	1
18	1
58	1

Question 3	
Task	Return the number of medals won for each country.
Explanation	This query should return two columns, one for the CountryID, and one for the number of medals won (if the country has won 0 medals, it should still be included).
Filename	*a3.sql* or *a3.txt*

SQL Solution	SELECT C.CountryID, COUNT(M.MedalType)
	FROM Medals M
	RIGHT JOIN Athletes A ON M.AthleteID = A.AthleteID
	RIGHT JOIN Countries C ON A.CountryID = C.CountryID
	GROUP BY C.CountryID;

Output screenshot:

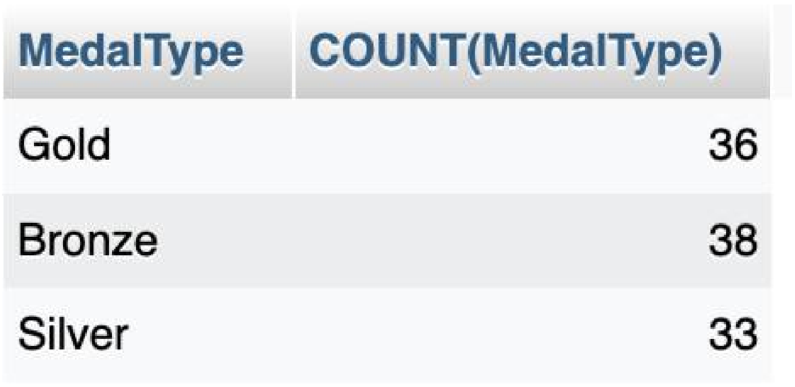


CountryID	COUNT(M.MedalType)
1	0
2	107
3	0
4	1
5	0
6	2
7	0
8	0
9	1
10	0
11	2
12	0
13	0
14	1
15	0
16	4
17	0
18	2
19	0
20	0
21	0
22	1
23	1
24	2
25	0
26	0
27	1
28	1
29	0
30	0

Question 4	
Task	Return the medal tally for ‘Australia’ across all events.
Explanation	This query should return a table with two columns, one with the type of medal (Gold, Silver, or Bronze) and the other with the number of medals won for Australia.
Filename	*a4.sql* or *a4.txt*
SQL Solution	`SELECT MedalType, COUNT(MedalType) FROM Medals WHERE AthleteID IN ( SELECT AthleteID FROM Athletes WHERE CountryID = ( SELECT CountryID FROM Countries WHERE CountryName= 'Australia' ) ) GROUP BY MedalType;`

```sql
SELECT MedalType, COUNT(MedalType)
FROM Medals
WHERE AthleteID IN (
      SELECT AthleteID
   FROM Athletes
   WHERE CountryID = (
      SELECT CountryID
      FROM Countries
      WHERE CountryName= 'Australia'
   )
)
GROUP BY MedalType;
```

Output screenshot:

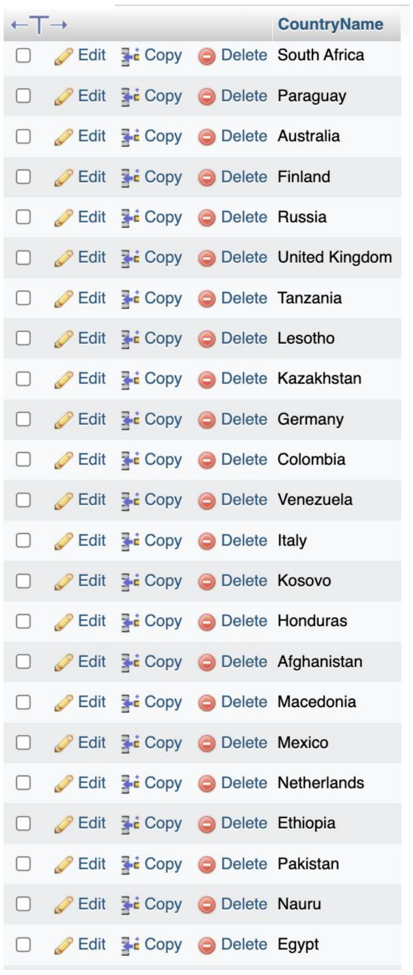


MedalType	COUNT(MedalType)
Gold	36
Bronze	38
Silver	33

Question 5	
Task	Return the country names of countries who have at least one participating athlete over the age of 30.
Explanation	This query should use at least one sub-query.
Filename	*a5.sql* or *a5.txt*

SQL Solution	SELECT CountryName
	FROM Countries
	WHERE CountryID IN (
	    SELECT DISTINCT CountryID
	    FROM Athletes
	    WHERE Age > 30
	);

Output screenshot:

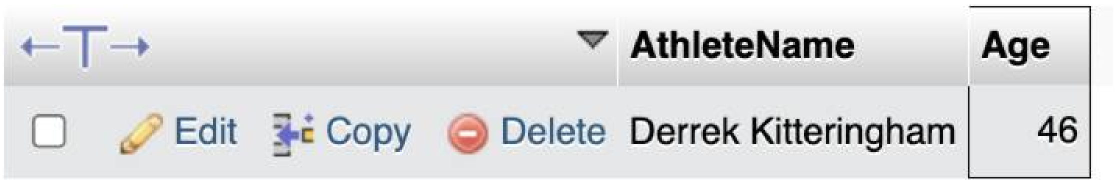


```
CountryName
South Africa
Paraguay
Australia
Finland
Russia
United Kingdom
Tanzania
Lesotho
Kazakhstan
Germany
Colombia
Venezuela
Italy
Kosovo
Honduras
Afghanistan
Macedonia
Mexico
Netherlands
Ethiopia
Pakistan
Nauru
Egypt
```

Question 6	
Task	Return the name, age of the youngest Australian athlete(s) participating in the Olympics.
Explanation	
Filename	*a6.sql* or *a6.txt*

SQL Solution	
	```
	SELECT AthleteName, Age
	FROM Athletes
	WHERE CountryID = (
	      SELECT CountryID
	   FROM Countries
	   WHERE CountryName= 'Australia'
	) AND Age <= ALL (
	   SELECT Age
	   FROM Athletes
	   WHERE CountryID = (
	      SELECT CountryID
	   FROM Countries
	   WHERE CountryName= 'Australia'
	      )
	);
	```

Output screenshot:

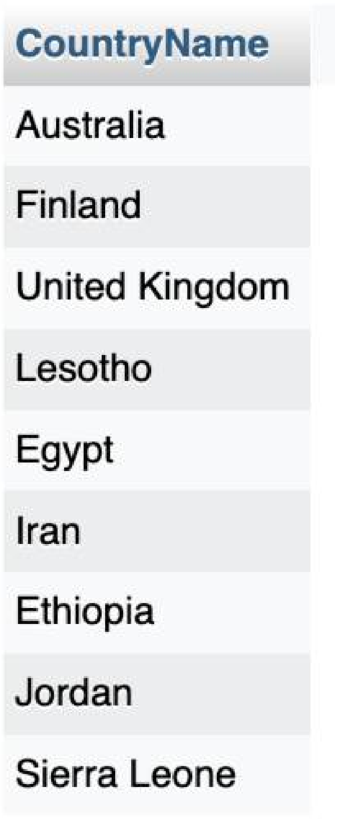


```
AthleteName          Age
Derrek Kitteringham   46
```

Question 7	
Task	Return the country names of countries that won more than one gold medal in the Olympics.
Explanation	
Filename	*a7.sql* or *a7.txt*

SQL Solution	`SELECT C.CountryName`
	 `FROM Countries C`
	 `JOIN Athletes A ON C.CountryID = A.CountryID`
	 `JOIN Medals M ON A.AthleteID = M.AthleteID`
	 `WHERE M.MedalType = 'Gold'`
	 `GROUP BY C.CountryName`
	 `HAVING COUNT(M.MedalType) > 1;`

Output screenshot:

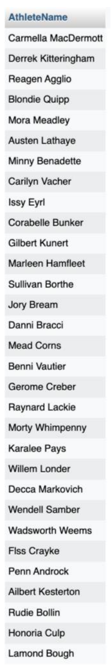


CountryName
Australia
Finland
United Kingdom
Lesotho
Egypt
Iran
Ethiopia
Jordan
Sierra Leone

Question 8	
Task	Return the names of athletes that medalled in expensive sports (i.e., sports that had at least 3 events with a ticket price over $100).
Explanation	**Hint**. You may want to use one or more views in your answer.
Filename	*a8.sql* or *a8.txt*
SQL Solution	`CREATE VIEW a8 AS SELECT SportID, Count(EventID) AS eventNumber FROM Events WHERE TicketPrice > 100 GROUP BY SportID; SELECT DISTINCT A.AthleteName FROM Athletes A JOIN Medals M ON A.AthleteID = M.AthleteID JOIN Events E ON M.EventID = E.EventID WHERE E.SportID IN (SELECT SportID FROM a8 WHERE eventNumber >= 3);`

```
CREATE VIEW a8 AS
   SELECT SportID, Count(EventID) AS eventNumber
   FROM Events
   WHERE TicketPrice > 100
   GROUP BY SportID;

SELECT DISTINCT A.AthleteName
FROM Athletes A
JOIN Medals M ON A.AthleteID = M.AthleteID
JOIN Events E ON M.EventID = E.EventID
WHERE E.SportID IN (
      SELECT SportID
   FROM a8
   WHERE eventNumber >= 3
);
```

Output screenshot:



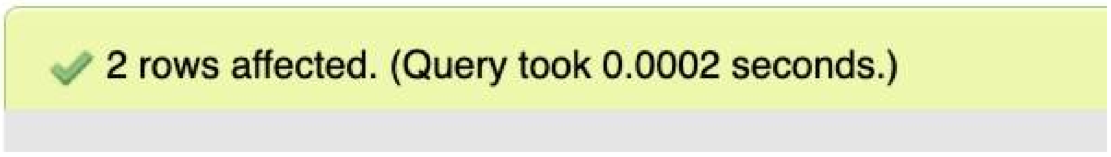
AthleteName
Carmella MacDermott
Derrek Kitteringham
Reagen Agglio
Blondie Quipp
Mora Meadley
Austen Lathaye
Minny Benadette
Carilyn Vacher
Issy Eyrl
Corabelle Bunker
Gilbert Kunert
Marleen Hamfleet
Sullivan Borthe
Jory Bream
Danni Bracci
Mead Corns
Benni Vautier
Gerome Creber
Raynard Lackie
Morty Whimpenny
Karalee Pays
Willem Londer
Decca Markovich
Wendell Samber
Wadsworth Weems
Flss Crayke
Penn Androck
Ailbert Kesterton
Rudie Bollin
Honoria Culp
Lamond Bough

Section B – SQL DML (UPDATE, DELETE, INSERT)

Question 1	
Task	Sarah Smith has had a positive performance-enhancing drugs test, so her medals (if any) need to be removed from the database.
Explanation	
Filename	*b1.sql* or *b1.txt*

SQL Solution	
	```
	DELETE FROM Medals
	WHERE AthleteID = (
	        SELECT AthleteID
	    FROM Athletes
	    WHERE AthleteName = 'Sarah Smith'
	);
	```

Output screenshot:

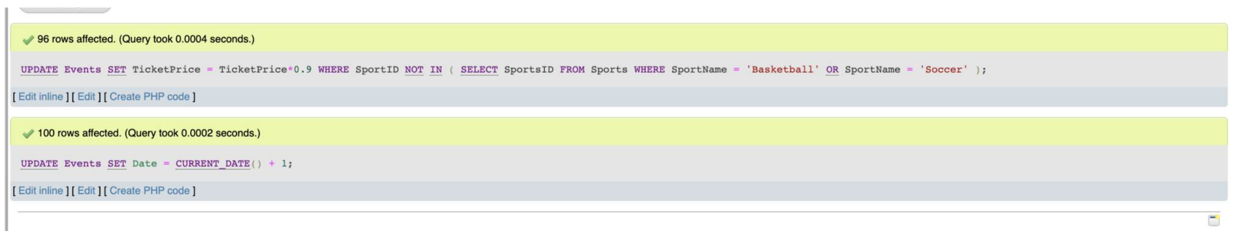


```
2 rows affected. (Query took 0.0002 seconds.)
```

Question 2	
Task	The ticket price for all sports and games except Basketball and Soccer are to be reduced by 10% due to a lack of demand. Issue this update in the database.
Explanation	This query should update the price of all other events in the future (i.e., the Date is later than the current date), to be 10% less than the existing price in the database.
Filename	*b2.sql* or *b2.txt*

SQL Solution	`UPDATE Events`
	`SET TicketPrice = TicketPrice*0.9`
	`WHERE SportID NOT IN (`
	`        SELECT SportID`
	`    FROM Sports`
	`    WHERE SportName = 'Basketball' OR SportName = 'Soccer'`
	`);`
	
	`UPDATE Events`
	`SET Date = CURRENT_DATE() + 1;`

Output screenshot:



```
✓ 96 rows affected. (Query took 0.0004 seconds.)
UPDATE Events SET TicketPrice = TicketPrice*0.9 WHERE SportID NOT IN ( SELECT SportsID FROM Sports WHERE SportName = 'Basketball' OR SportName = 'Soccer' );
[ Edit inline ] [ Edit ] [ Create PHP code ]
✓ 100 rows affected. (Query took 0.0002 seconds.)
UPDATE Events SET Date = CURRENT_DATE() + 1;
[ Edit inline ] [ Edit ] [ Create PHP code ]
```

Section C – SQL DDL

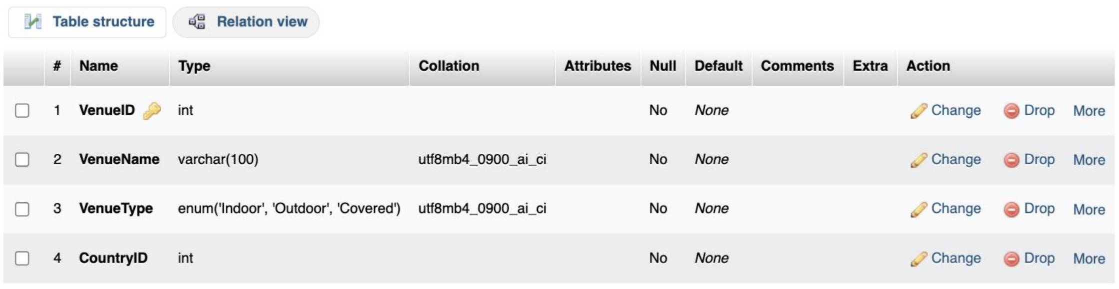
Question 1	
Task	Write a SQL DDL query to implement the following relational schema and associated foreign keys.
Explanation	The relational schema for this the table is as follows:

Table: Venues			
Column	**Data Type**	**Allow Nulls?**	**Primary Key?**
VenueID	**INT**	**N**	**Y**
VenueName	**VARCHAR**	**N**	**N**
VenueType	**(‘Indoor’, ‘Outdoor’, ‘Covered’)**	**N**	**N**
CountryID	**INT**	**N**	**N**

Filename	*c1.sql* or *c1.txt*
SQL Solution	

```
CREATE TABLE Venues

        ( VenueID INT NOT NULL,

        VenueName VARCHAR(100) NOT NULL,

        VenueType ENUM ('Indoor','Outdoor','Covered') NOT NULL,

        CountryID INT NOT NULL,

        PRIMARY KEY (VenueID) );

SELECT * FROM Venues;
```

Output screenshot:



	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	**VenueID**	int			No	*None*			Change Drop More
☐	2	**VenueName**	varchar(100)	utf8mb4_0900_ai_ci		No	*None*			Change Drop More
☐	3	**VenueType**	enum('Indoor', 'Outdoor', 'Covered')	utf8mb4_0900_ai_ci		No	*None*			Change Drop More
☐	4	**CountryID**	int			No	*None*			Change Drop More

Question 2	
Task	To ensure that all events are reasonably priced, add a constraint that ensures that no ticket is priced under $10 and over $1000.
Explanation	The following resources may be useful when answering this question: Check constraints
Filename	*c2.sql* or c2*.txt*

SQL Solution	`ALTER TABLE `` `Events` `` ADD CONSTRAINT `` `Chk_TicketPrice` `` CHECK (`` `TicketPrice` `` > 10.00 AND `` `TicketPrice` `` < 1000.00);`
	
	`SELECT * FROM Events;`

```
ALTER TABLE `Events`
ADD CONSTRAINT `Chk_TicketPrice`
CHECK (`TicketPrice` > 10.00  AND `TicketPrice` < 1000.00);

SELECT *
FROM Events;
```

Output screenshot:

Section D – Critical Thinking

In this section, you will receive theoretical situations related to the UoD mentioned in the task description. Your task is to offer strategies to tackle the situation and write SQL queries to execute the approaches.

- INFS1200 students answer Question 1 only.
- INFS7900 students answer both Question 1 and Question 2.

Question 1	
Task	Olympics games planners want to know what to expect during the any of the most busy weeks in the Olympics (i.e., how many athletes will be participating, how many different sports, how many different countries and so on). Propose a strategy for the given task and write an SQL query to implement that strategy. **Hint**: The SQL WEEK() function may be useful.
Strategies	Create 2 tables: - d1[Weeks, NumbEvents, NumbSports] -- Number of Events and Sports for each week - d2[Weeks, NumbAthletes] -- Number of Athletes for each week (only select unique)
SQL Solution	`WITH d1 AS (SELECT Week(Date) AS `Weeks`, COUNT(EventID) AS `NumbEvents`, COUNT(SportID) AS `NumbSports` FROM Events GROUP BY Weeks), d2 AS (SELECT Week(Date) AS `Weeks`, COUNT(DISTINCT C.AthleteID) AS `NumbAthletes` FROM Events E JOIN Contestants C ON E.EventID = C.EventID GROUP BY Weeks) SELECT d1.Weeks, d1.NumbEvents, d1.NumbSports, d2.NumbAthletes FROM d1, d2 WHERE d1.Weeks = d2.Weeks ORDER BY d1.Weeks`

```
WITH d1 AS
  (
    SELECT
      Week(Date) AS `Weeks`,
      COUNT(EventID) AS `NumbEvents`,
      COUNT(SportID) AS `NumbSports`
    FROM Events
    GROUP BY Weeks
  ),
  d2 AS
  (
    SELECT
      Week(Date) AS `Weeks`,
      COUNT(DISTINCT C.AthleteID) AS `NumbAthletes`
    FROM Events E
    JOIN Contestants C ON E.EventID = C.EventID
    GROUP BY Weeks
  )
SELECT d1.Weeks, d1.NumbEvents, d1.NumbSports, d2.NumbAthletes
FROM d1, d2
WHERE d1.Weeks = d2.Weeks
ORDER BY d1.Weeks
```

Question 2 – INFS7900 ONLY	
Task	Olympics games planners want to know how many athletes to expect in the Olympic Village on different days (so that they can provide enough accommodation, catering and support services, etc). Propose a strategy for the given task and write an SQL query to implement that strategy.
Strategies	Count number of athletes on the day that event happens (only count unique athletes). It means that if one person participates 2 events on the same day, it will be count just one.

SQL Solution	SELECT E.Date, COUNT(DISTINCT C.AthleteID)
	FROM Events E
	JOIN Contestants C ON E.EventID = C.EventID
	GROUP BY E.Date
	ORDER BY E.Date;

Documenting the use of Generative AI

Please note that if you have used generative AI in any manner, you are required to provide a transcript of your engagement with the system in this section. You can simply copy and paste your discussion with the generative AI system below. It is fine if it goes across multiple pages.

A reminder that a failure to reference AI use may constitute student misconduct under the Student Code of Conduct.