Student Names and Student IDs
Morgan Law #37577153
Emil Aberg #92305986
Naman Bansal #29328144
Thomas Welsh #41953944

DELIVERABLES: EXPLAINED

(Note: anything preceded by a colon in any sql statements is a variable and is replaced with actual correct domains and values in php)

We have a total of over 30 queries, most of which are in the document `queries.sql`. All are integrated within php scripts and run in our application. There was a minor change in the E-R diagram/formal schema, where we added the attribute country to athlete.

Deliverable 1:
Users: Athletes, Coaches, Competition Managers
All users can see ranking, records and competition information.
Athletes are able to:

- search for coaches and sign up for coaching
- sign up for teams
- view their personal drug test results
- modify their own height, weight and country
- view training plans created for them by their coaches
- sign up for competitions

Coaches are able to:

- query for information about the athletes that they coach
- create new training plans for the athletes that they coach
- modify the sports they coach and their location
- sign up their athletes for competitions
- add and delete sports that they coach
- see sports they teach

Competition Managers are able to:

- create new competitions
- update start date, end date and location of their competitions
- update and insert results of their competitions and update status for participants
- add new entries for drug testing results for athletes
- query for athletes who have taken every drug test

Note, some privileges and functionality were added or deleted from our initial formal specifications, mainly due to these not making sense from a logical and data integrity standpoint.

Deliverable 2: There are numerous (many) insert statements in our implementation. Here is an example below:

```
INSERT INTO Competition(competitionID, name, startDate, endDate, inviteOnly,
       country, province_state, city, competitionOrganizerID)
```

```
VALUES (:compID, ':name', ':startDate', ':endDate', :invOnly, ':country', ':prov', ':city',
:managerID)
```

Deliverable 3: There are numerous (many) delete statements in our implementation. Here is an example below:

```
"DELETE FROM teaches
       WHERE coachID = ':coachID' AND sportID = ':sportID'";
```

Deliverable 4: One of the update statement below:

```
UPDATE participates:sportType
SET status = ':status', result = :result
WHERE athleteID = :athleteID AND competitionID = :compID AND sportID = :sportID
```

Deliverable 5 and 7:

This query calculates the records for a given sport grouped by division and filtered by country and/or season. Aggregation is used to find the record. This query joins 3 tables.

```
SELECT CONCAT(d.gender, ' ', CONVERT(d.minAge, CHARACTER), '-', CONVERT(d.maxAge,
CHARACTER)) AS 'Division',
       p.name AS 'Name', a.country AS 'Country', divRecord.record AS 'Record'
FROM Participates:sportType parti
JOIN Competition c
       ON parti.competitionID = c.competitionID AND YEAR(c.endDate) LIKE :season AND
parti.sportID = :sportID
JOIN Athlete a
       ON parti.athleteID = a.athleteID AND a.country LIKE ':country'
JOIN (SELECT divisionID, IF(:sportID IN (SELECT sportID FROM TBSport), MIN(result),
MAX(result)) AS record
       FROM Participates:sportType parti
       JOIN Competition c
              ON parti.competitionID = c.competitionID AND YEAR(c.endDate) LIKE :season
AND parti.sportID = :sportID
       JOIN Athlete a
              ON parti.athleteID = a.athleteID AND a.country LIKE ':country'
       GROUP BY divisionID) AS divRecord
       ON divRecord.divisionID = parti.divisionID
JOIN Division d
       ON parti.divisionID = d.divisionID
JOIN Person p
       ON parti.athleteID = p.personID
WHERE parti.result = divRecord.record
```

Deliverable 6:
Following query calculates the result for a given competition, division and sport. It joins two tables.

```
SELECT p.name AS 'Athlete', result AS 'Result'
FROM Participates:sportType parti
JOIN Person p
	ON parti.athleteID = p.personID
WHERE parti.divisionID = :divID AND parti.competitionID = :compID AND parti.sportID =
:sportID
ORDER BY parti.result :sort
```

Deliverable 8-10: see `queries.sql`

Deliverable 11:

```
CREATE VIEW AthleteRank AS
SELECT p.name AS aName, a.country AS country, r.points AS points, s.name AS sName,
s.sportID AS sportID,
	CONCAT(d.gender, ' ', CONVERT(d.minAge, CHARACTER), '-', CONVERT(d.maxAge,
CHARACTER)) AS divName, d.divisionID AS divisionID
FROM Rank r
JOIN Person p
	ON p.personID = r.athleteID
JOIN Athlete a
	ON a.athleteID = p.personID
JOIN Sport s
	ON s.sportID = r.sportID
JOIN Division d
	ON d.gender = p.gender AND
		YEAR(CURRENT_DATE) - YEAR(p.birthDate) BETWEEN d.minAge AND d.maxAge
```

Extra Stuff
We also have 2 triggers, both to update an athletes rank upon entering of a result. Here is an example below:

```
CREATE TRIGGER UpdateRankSB
AFTER UPDATE ON ParticipatesSB
FOR EACH ROW
	UPDATE Rank r
	SET r.points =
		(
		SELECT ROUND((MAX(result)+AVG(result))/2) AS Points
		FROM participatesSB parti
		JOIN Competition c
			ON parti.competitionID = c.competitionID AND YEAR(c.startDate) =
YEAR(CURRENT_DATE)
		WHERE parti.athleteID = NEW.athleteID AND parti.sportID = NEW.sportID
```

```
            )
        WHERE r.athleteID = NEW.athleteID AND r.sportID = NEW.sportID
```

There are also 2 events, one to automatically delete expired drug tests and one to reset ranks at the end of the season/start of new season. Both are shown below:

```
CREATE EVENT `DrugTestExpiry`
  ON SCHEDULE EVERY 1 MINUTE
    STARTS '2018-11-17 00:00:00.000000' ENDS '2020-03-31 00:00:00.000000'
      ON COMPLETION PRESERVE
      ENABLE
      DO DELETE FROM athletetakestest
        WHERE CURRENT_DATE > expDate;

CREATE EVENT `NewSeason`
  ON SCHEDULE EVERY 1 YEAR
    STARTS '2018-12-31 23:59:59' ENDS '2037-12-31 23:59:59'
      ON COMPLETION PRESERVE
        ENABLE COMMENT 'Reset points of every athlete after season finishes'
          DO UPDATE rank SET points = 0
            WHERE 1=1;
```

Unexpected Challenges:

- None of us knew php when starting the project, which slowed us down greatly in the beginning.
- We also didn't know that much of html, so at times it was hard to obtain the desired layout of tables and other UI elements.
- All the changes we made to the er diagram and formal specifications at various stages of the project. While we eventually settled on something relatively similar to our initial design, most of this chopping and changing was due to us not knowing how to write queries properly when we initially started, as well as the complexity of some of the queries we ended up writing.

Other Features:

All data and initial insert statements were created using a script written in perl (`datageneration.pl`), which allowed for the generation of around 28000 sql queries to populate our tables. Getting all the data to maintain its integrity was a challenge, but also allows us to showcase the full functionality of our database. Some information was created quasi-randomly (e.g. country of athlete, what competition is managed by what competition Manager, etc), allowing for much variation in our data.
`datageneration.pl` outputs `insertStatements.sql`, which creates, loads and populates our database and also creates users.