8 June 2022

Practical Assignment 5

COS 221

Mr. Quintin de Villiers - u21513768
Mr. Jason Dutton - u21477452
Mr. Justin Eckard - u21528332
Mr. Jesse Naidoo - u21433102
Mr. Phillip Malan - u20547791

Task 1:

Our chosen sport to include in the SportsDB was swimming. Swimming is an individual sport where each swimmer competes for the fastest time in a race of a certain stroke over a certain distance. While swimming is an individual sport a swimmer can enter an event as part of a team where they will then earn points for their team based on their individual performance in the races of their choice. Swimming for a team in an event does not change any aspects of an event other than that they will earn points for team.

Every Event will offer many individual races, each race has an associated distance (50m, 100m, 200m, …) and stroke type (Freestyle, Backstroke, Breaststroke, Butterfly, …). A race is won by the swimmer who finishes the race in the quickest time, while each swimmer will earn points for their team, if they are part of one, based on their position in the race.

If there are more swimmers participating in a race than lanes in the pool then the race will be divided into heats, however only the time in each heat matters and coming first in your heat makes no difference as positions in races are based on time for the whole race not individual heats

A swimmer will be disqualified from a race if they:

- Perform a false start – when a swimmer dives/starts swimming before the start of the race is signalled
- Do not touch the wall when performing a turn, if the race is longer than 1 lap in the pool
- Do not finish the race
- Finish incorrectly – some strokes require the swimmer to touch the wall with both hands simultaneously or to touch the wall on their back.

If a swimmer is disqualified from a race, then they will not have a recorded time or position and in turn will earn no points for their team if they are part of one.

A swimming event can take place at multiple venues over multiple days. In addition to every race being timed in order to measure the time each swimmer takes to complete the race; each race has a start time which is the time during the day the event will start and a particular date on which the race will happen.

Task 2

Assumptions:

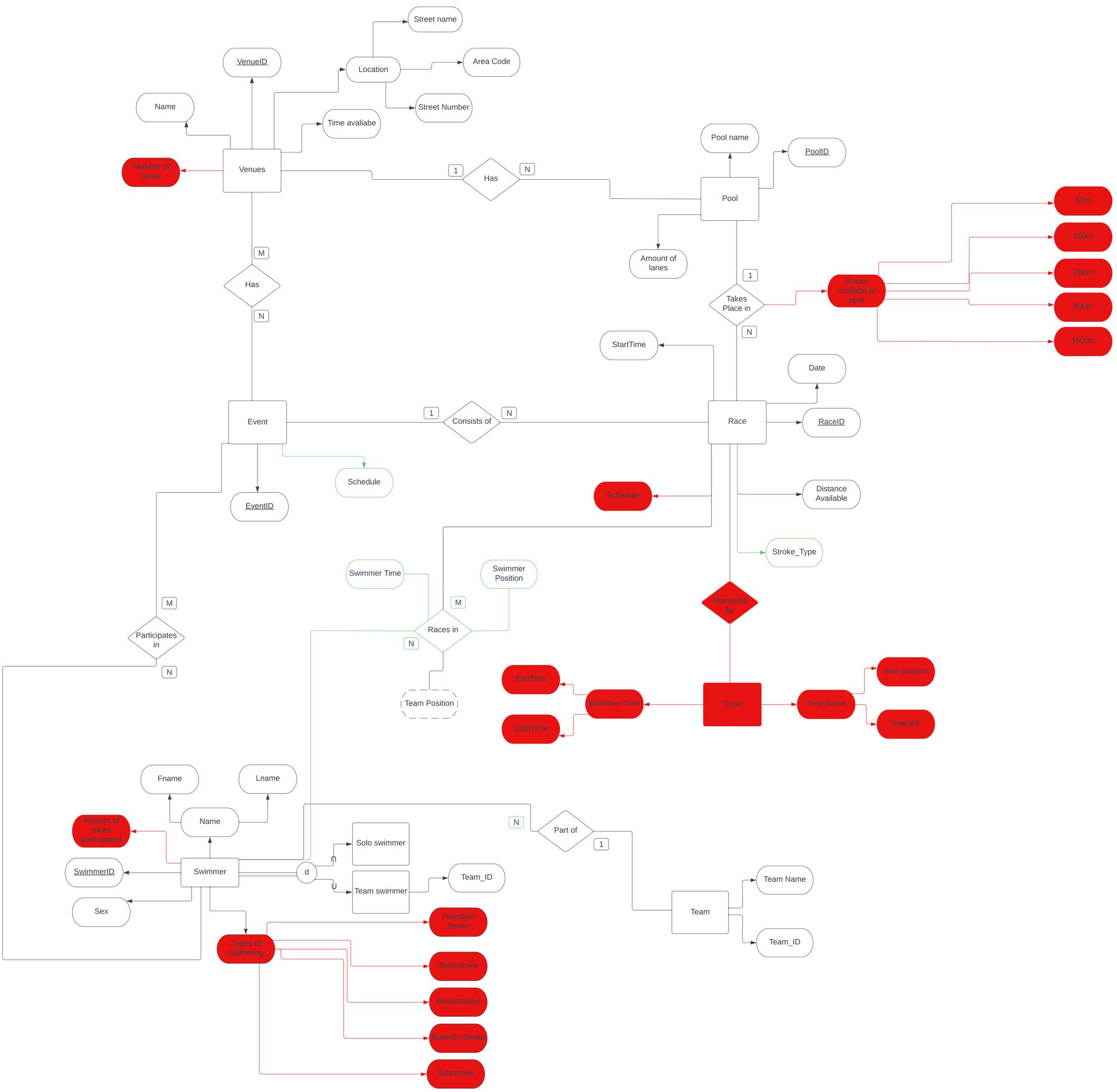
- Event-Venue Relationship – Events can have many venues, while venues are also able to host multiple events
- Solo/team swimmer – a swimmer can enter an event part of a team or as an individual, a swimmer can only be part of 1 team
- There will be no winning team since this is an individual sport there will be individual winners for events and races

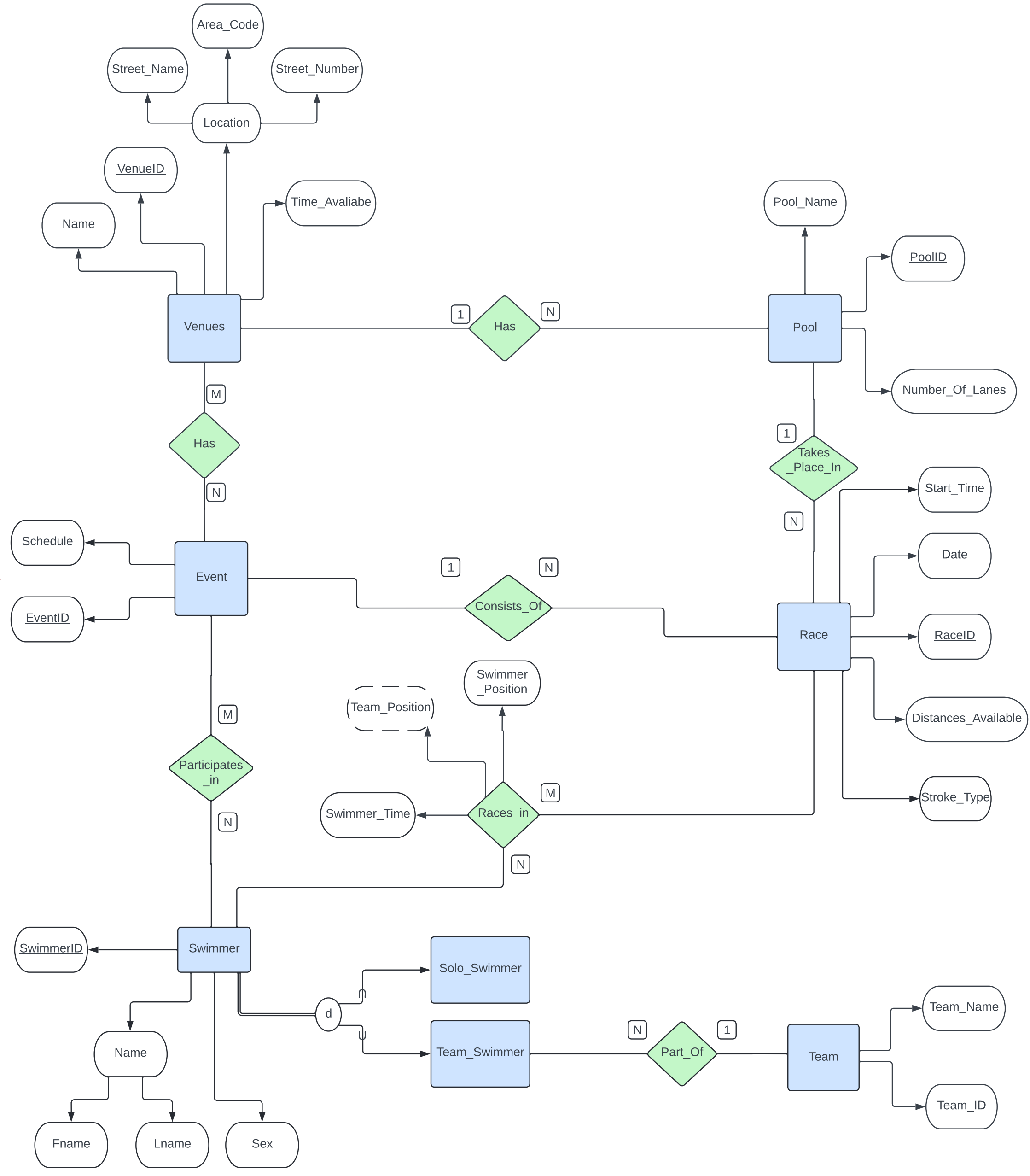
Improvements from the Draft:

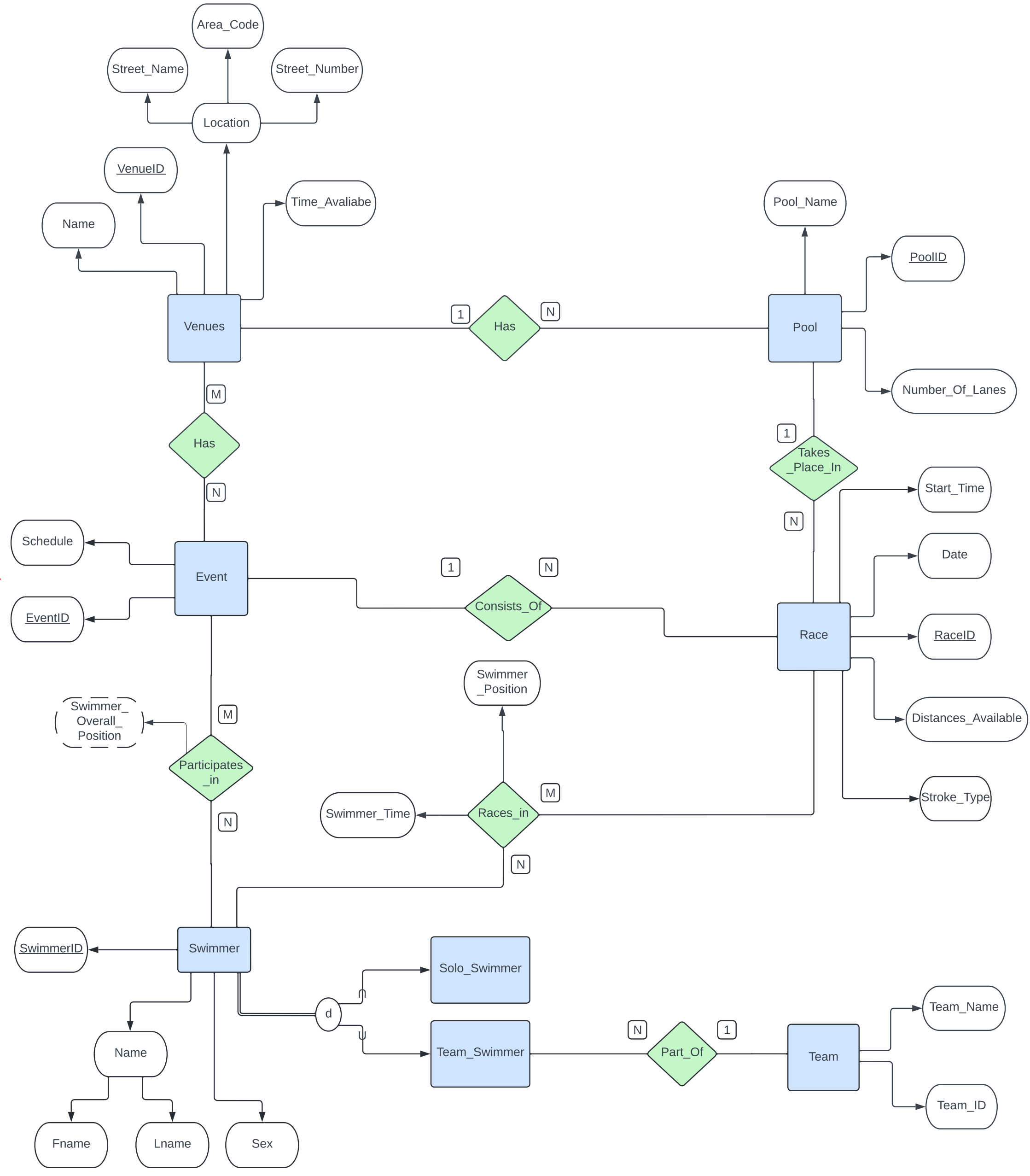
- Added a relationship between swimmer and race which stores the swimmers time and position in a particular race and the team that the swimmer represents' positions
- Removes the composite attributes of race distance and stroke
- Added a stroke type and distance available for each race
- Removed the timer entity and added Start_Time in race instead
- Added a team entity which stores the team name and id of the team a swimmer is part of

Improvements from the 2nd draft:

- Removed the derived attribute Team_Position from the Races_In relationship
- Added the derived attribute Swimmer_Overall_Position to the Participates_In relationship


Street name
VenueID
Location
Area Code
Name
Street Number
Time avaliabe
Pool name
PoolID
Number of lanes
Venues
1
Has
N
Pool
50m
100m
200m
800m
1500m
M
Has
Amount of lanes
1
Races availabe at pool
Takes Place in
N
StartTime
Date
Event
1
Consists of
N
Race
RaceID
Schedule
Schedule
Distance Available
EventID
Stroke_Type
Swimmer Time
Swimmer Position
M
Monitored by
M
Participates in
Races in
N
N
Team Position
EndTime
Swimmer Time
Timer
Time frame
time passed
StartTime
time left
Fname
Lname
Name
N
Part of
Amount of races participated
Solo swimmer
SwimmerID
Swimmer
d
Team swimmer
Team_ID
1
Team
Team Name
Sex
Freestyle Stroke
Types of swimming
Backstroke
Team_ID
Breaststroke
Butterfly Stroke
Sidestroke


Area_Code
Street_Name
Street_Number
Location
VenueID
Name
Time_Avaliabe
Venues
1
Has
N
Pool_Name
PoolID
Pool
Number_Of_Lanes
M
Has
N
1
Takes
_Place_In
N
Start_Time
Schedule
Event
1
Consists_Of
N
Date
EventID
Race
RaceID
Swimmer
_Position
Team_Position
Distances_Available
M
Participates
_in
N
Swimmer_Time
Races_in
M
Stroke_Type
N
SwimmerID
Swimmer
Solo_Swimmer
d
Team_Name
N
Part_Of
1
Team_Swimmer
Team
Name
Team_ID
Fname
Lname
Sex


Area_Code
Street_Name
Street_Number
Location
VenueID
Name
Time_Avaliabe
Venues
1
Has
N
Pool_Name
PoolID
Pool
Number_Of_Lanes
M
Has
N
1
Takes
_Place_In
N
Start_Time
Schedule
Event
1
Consists_Of
N
Date
EventID
Race
RaceID
Swimmer
_Position
Distances_Available
Swimmer_
Overall_
Position
M
Participates
_in
N
Swimmer_Time
Races_in
M
Stroke_Type
N
SwimmerID
Swimmer
Solo_Swimmer
d
Team_Name
Name
Team_Swimmer
N
Part_Of
1
Team
Team_ID
Fname
Lname
Sex

Task 3:

EER diagram to Relational Mapping steps.

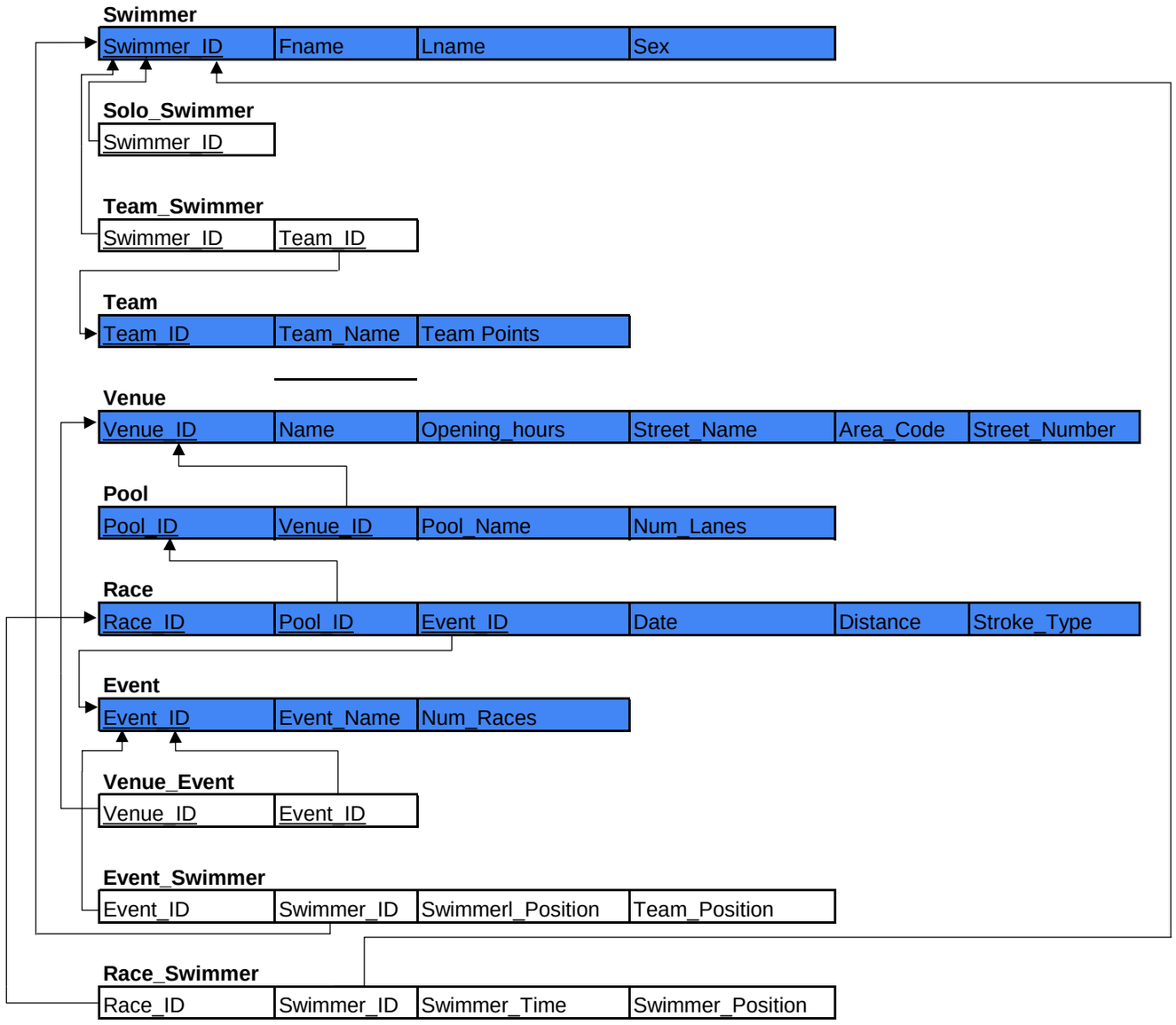
1. Mapping of strong entity types: Strong entities include
    - Swimmer
    - Team
    - Venue
    - Event
    - Race
    - pool

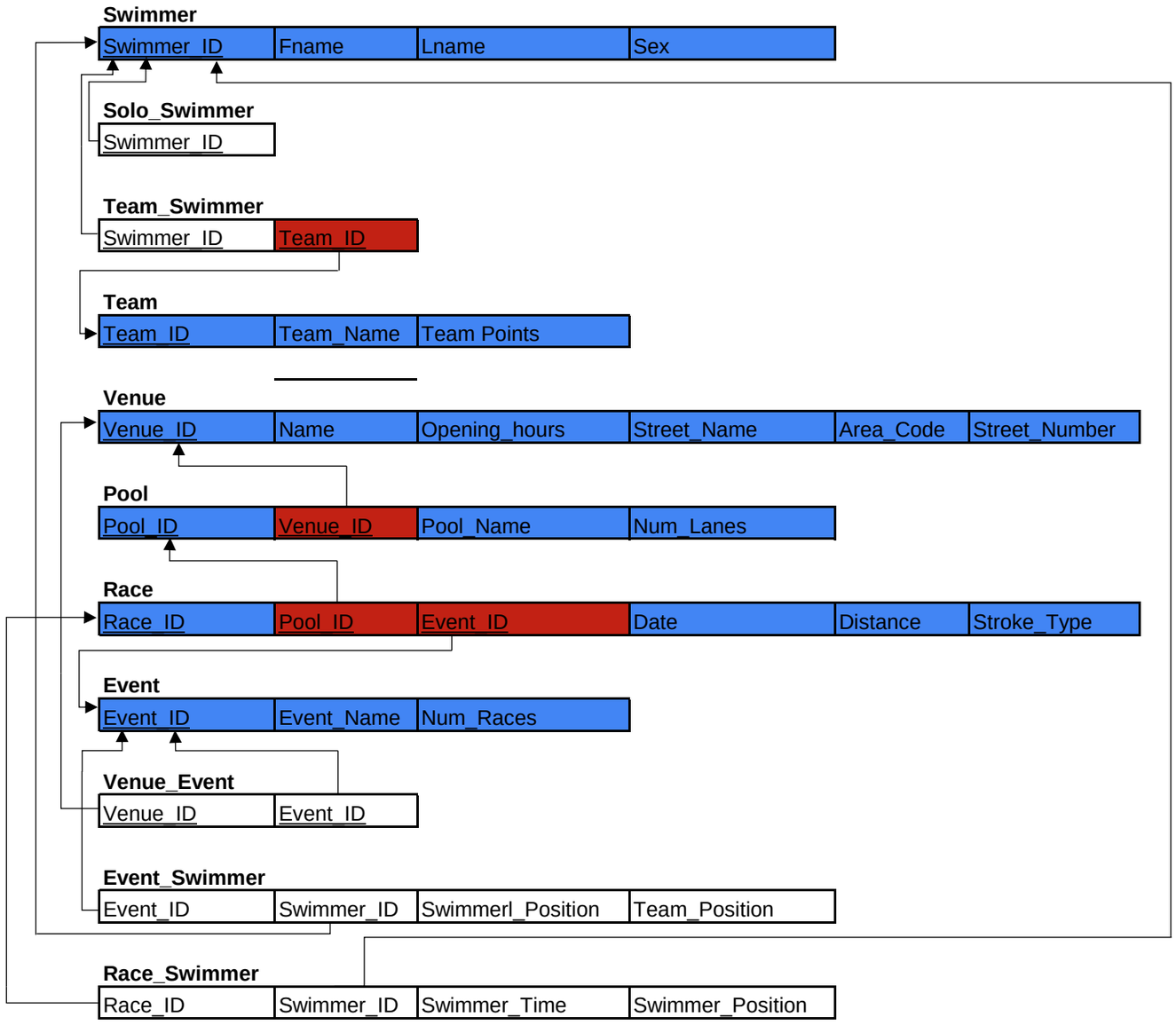
    The simple component attributes for each are then added. This is shown in blue.
2. Mapping of weak entity types: No weak entity types present in the E-ER diagram.
3. Mapping of binary 1:1 relations: no binary 1:1 relations present in the E-ER diagram
4. Mapping of binary 1:N relationships: These include
    - Team Swimmer - Team
    - Race - Pool
    - Race - Event
    - Pool - Venue

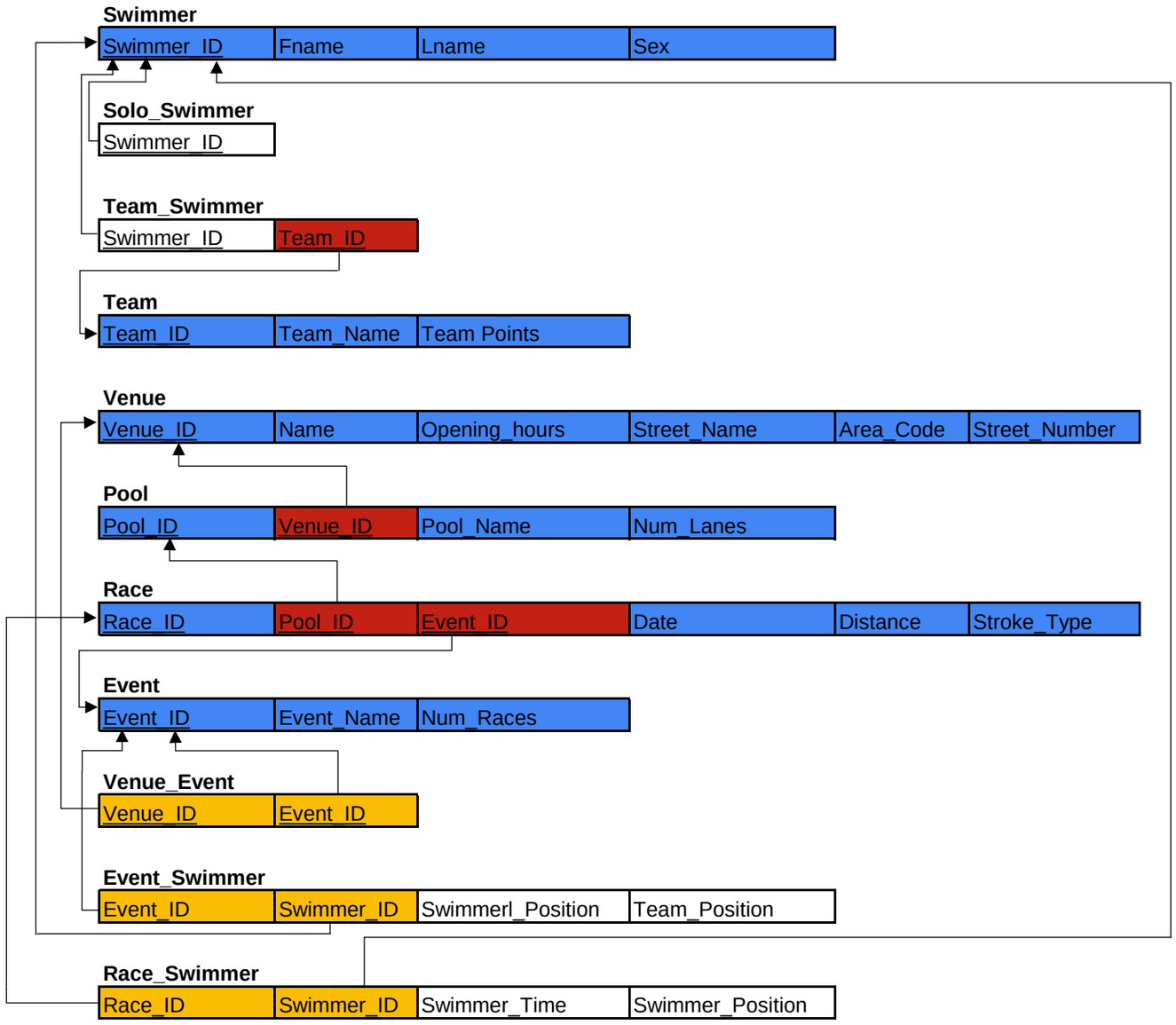
    The primary keys of the 1 relation are added to relation N as shown in red.
5. Mapping of binary M:N relationships: These relations are as follows
    - Event – Swimmer
    - Race – Swimmer
    - Event – Venue

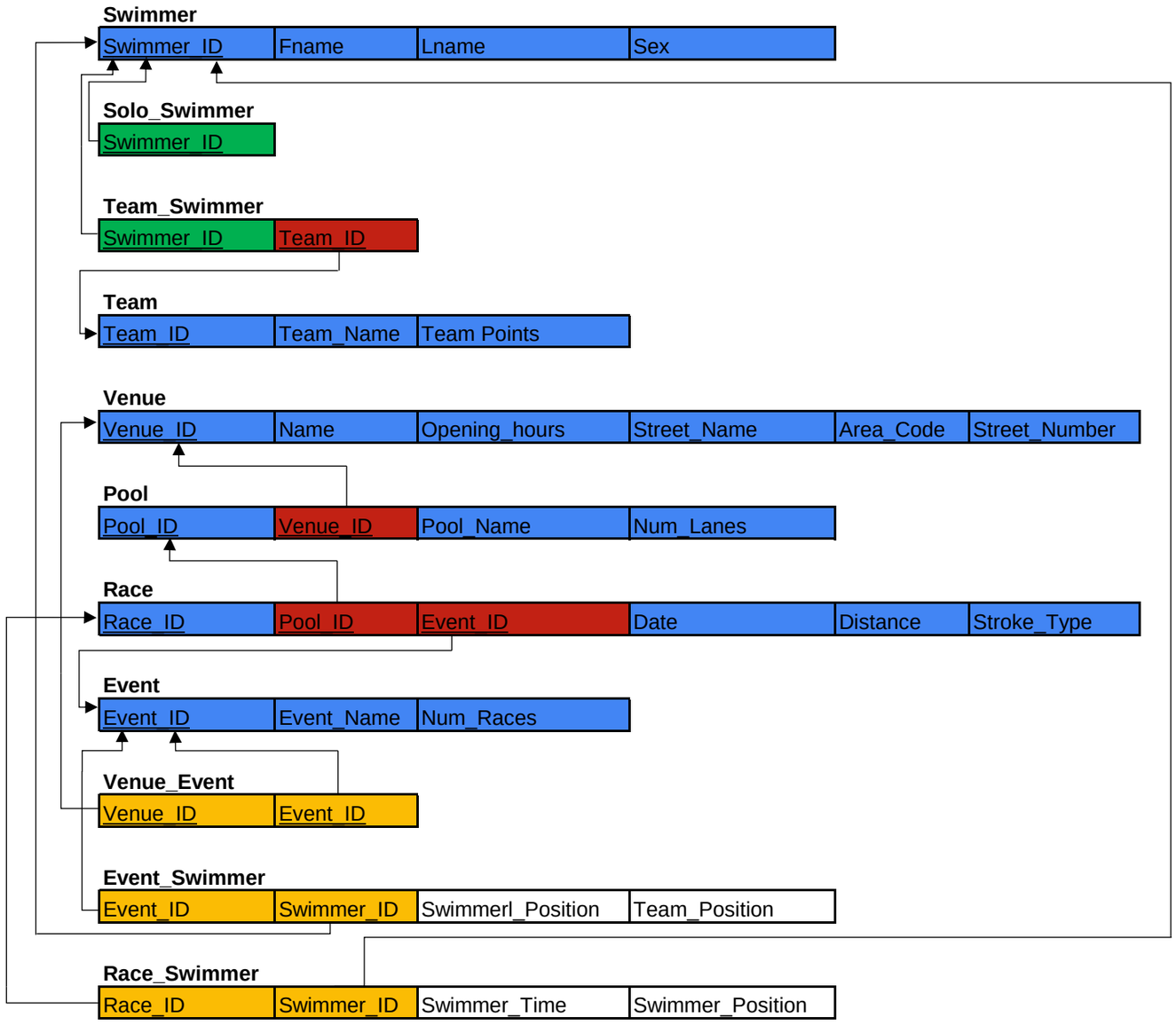
    New relations have been created for the relationships and the primary keys of the relations are foreign keys in the new relation. This is shown in Yellow. The simple attributes have also been added.
6. Mapping of multivalued attributes: No multivalued attributes in the E-ER diagram.
7. Mapping of N-ary relationships: No N-ary relationships.
8. Mapping of specialisation: Specialisation includes
    - Solo Swimmer
    - Team Swimmer

    Option 8a. created a superclass Swimmer that contains all common attributes of Solo and team swimmer. Each subclass contains the primary key of Swimmer as a foreign key.
    This is shown in green.
9. Mapping of unions: No unions in the E-ER diagram.


Swimmer
Swimmer_ID
Fname
Lname
Sex
Solo_Swimmer
Swimmer_ID
Team_Swimmer
Swimmer_ID
Team_ID
Team
Team_ID
Team_Name
Team Points
Venue
Venue_ID
Name
Opening_hours
Street_Name
Area_Code
Street_Number
Pool
Pool_ID
Venue_ID
Pool_Name
Num_Lanes
Race
Race_ID
Pool_ID
Event_ID
Date
Distance
Stroke_Type
Event
Event_ID
Event_Name
Num_Races
Venue_Event
Venue_ID
Event_ID
Event_Swimmer
Event_ID
Swimmer_ID
Swimmerl_Position
Team_Position
Race_Swimmer
Race_ID
Swimmer_ID
Swimmer_Time
Swimmer_Position


Swimmer
Swimmer_ID
Fname
Lname
Sex
Solo_Swimmer
Swimmer_ID
Team_Swimmer
Swimmer_ID
Team_ID
Team
Team_ID
Team_Name
Team Points
Venue
Venue_ID
Name
Opening_hours
Street_Name
Area_Code
Street_Number
Pool
Pool_ID
Venue_ID
Pool_Name
Num_Lanes
Race
Race_ID
Pool_ID
Event_ID
Date
Distance
Stroke_Type
Event
Event_ID
Event_Name
Num_Races
Venue_Event
Venue_ID
Event_ID
Event_Swimmer
Event_ID
Swimmer_ID
Swimmerl_Position
Team_Position
Race_Swimmer
Race_ID
Swimmer_ID
Swimmer_Time
Swimmer_Position


Swimmer
Swimmer_ID
Fname
Lname
Sex
Solo_Swimmer
Swimmer_ID
Team_Swimmer
Swimmer_ID
Team_ID
Team
Team_ID
Team_Name
Team Points
Venue
Venue_ID
Name
Opening_hours
Street_Name
Area_Code
Street_Number
Pool
Pool_ID
Venue_ID
Pool_Name
Num_Lanes
Race
Race_ID
Pool_ID
Event_ID
Date
Distance
Stroke_Type
Event
Event_ID
Event_Name
Num_Races
Venue_Event
Venue_ID
Event_ID
Event_Swimmer
Event_ID
Swimmer_ID
Swimmerl_Position
Team_Position
Race_Swimmer
Race_ID
Swimmer_ID
Swimmer_Time
Swimmer_Position


Swimmer
Swimmer_ID
Fname
Lname
Sex
Solo_Swimmer
Swimmer_ID
Team_Swimmer
Swimmer_ID
Team_ID
Team
Team_ID
Team_Name
Team Points
Venue
Venue_ID
Name
Opening_hours
Street_Name
Area_Code
Street_Number
Pool
Pool_ID
Venue_ID
Pool_Name
Num_Lanes
Race
Race_ID
Pool_ID
Event_ID
Date
Distance
Stroke_Type
Event
Event_ID
Event_Name
Num_Races
Venue_Event
Venue_ID
Event_ID
Event_Swimmer
Event_ID
Swimmer_ID
Swimmerl_Position
Team_Position
Race_Swimmer
Race_ID
Swimmer_ID
Swimmer_Time
Swimmer_Position

Task 4 : SQL for database

```
CREATE TABLE Swimmer(
Swimmer_ID INT NOT NULL auto_increment, primary key(Swimmer_ID), Fname
VARCHAR(100) NOT NULL, Lname VARCHAR(100) NOT NULL,Sex VARCHAR(50));

CREATE TABLE Solo_Swimmer(Swimmer_ID INT auto_increment NOT NULL, primary
key(Swimmer_ID), foreign key (Swimmer_ID) references swimmer(Swimmer_ID));

CREATE TABLE Team_Swimmer(Swimmer_ID INT auto_increment NOT NULL, Team_ID
INT NOT NULL, primary key(Swimmer_ID)-- Remember to add forigen key);

create TABLE Venue(Venue_ID INT auto_increment NOT NULL, primary key(Venue_ID),
Name VARCHAR(100) NOT NULL, Opening_hours TIME, Street_Name VARCHAR(100),
Area_Code VARCHAR(20), Street_Number varchar(10));

CREATE TABLE Pool(Pool_ID INT auto_increment NOT NULL, primary key(Pool_ID),
Venue_ID INT, foreign key (Venue_ID) references Venue(Venue_ID), Pool_Name
varchar(100) NOT NULL, Num_Lanes INT);

CREATE TABLE Event(Event_ID INT auto_increment NOT NULL,Event_Namevarchar(100),
Num_Races INT, primary key(Event_ID));

CREATE TABLE Team(Team_ID INT auto_increment NOT NULL,primary
key(Team_ID),Team_Name VARCHAR(100));

alter TABLE team_swimmerADD FOREIGN KEY (Team_ID) REFERENCES
Team(Team_ID);

CREATE TABLE Venue_Event(Venue_ID INT auto_increment NOT NULL,
primarykey(Venue_ID),Event_ID INT, foreign key (Venue_ID) referencesEvent(Event_ID));

CREATE TABLE Event_Swimmer(Event_ID INT auto_increment NOT NULL,
primarykey(Event_ID), Swimmer_ID INT, foreign key (Event_ID) references
Swimmer(Swimmer_ID));

create table Race_Swimmer(Race_ID INT auto_increment NOT NULL,
primarykey(Race_ID), Swimmer_ID INT, foreign key (Swimmer_ID)
referencesSwimmer(Swimmer_ID), Swimmer_Time TIME, Swimmer_Position
INT,Team_Position INT);

ALTER TAble Race_Swimmer MODIFY column Swimmer_Time TIME(3);


CREATE TABLE Race(Race_ID INT auto_increment NOT NULL,primary key(Race_ID),
Pool_ID INT NOT NULL, foreign key (Pool_ID) references pool(Pool_ID),Event_ID INT NOT
NULL, foreign key (Event_ID) references event(Event_ID), Date DATE NOT NULL, Distance
INT NOT NULL, Stroke_TypeVARCHAR(20));
```

Task 6:

Reason for manual data entry : Simpler and quicker than using PHP to enter the data to the respective tables , more familiarity with mysql workbench also made it easier

```
INSERT INTO swimmer (Fname , Lname , Sex)
VALUES ('Susan','Penn','Female');

INSERT INTO swimmer (Fname , Lname , Sex)
VALUES ('Joel','Osteen','Male');

INSERT INTO swimmer (Fname , Lname , Sex)
VALUES ('Quintin','vd Merwe','Male');

INSERT INTO swimmer (Fname , Lname , Sex)
VALUES ('Ian','de Witt','Male');

INSERT INTO swimmer (Fname , Lname , Sex)
VALUES ('Russel','Orhii','Male');

INSERT INTO swimmer (Fname , Lname , Sex)
VALUES ('Taylor','Atwood','Male');

INSERT INTO swimmer (Fname , Lname , Sex)
VALUES ('Nicole','Bosch','Female');

INSERT INTO swimmer (Fname , Lname , Sex)
VALUES ('Barbara','Wolff','Female');

INSERT INTO swimmer (Fname , Lname , Sex)
VALUES ('Shannon','Wolff','Female');

INSERT INTO swimmer (Fname , Lname , Sex)
VALUES ('Brandon','Duncan','Male');
```

```
INSERT INTO solo_swimmer (Swimmer_ID)
VALUES ('5');

INSERT INTO solo_swimmer (Swimmer_ID)
VALUES ('3');
```

```
INSERT INTO solo_swimmer (Swimmer_ID)
VALUES ('7');

INSERT INTO solo_swimmer (Swimmer_ID)
VALUES ('10');



INSERT INTO team (Team_Name)
VALUES ('Randburg High');

INSERT INTO team (Team_Name)
VALUES ('Wet boys swim team');

INSERT INTO team (Team_Name)
VALUES ('Gauteng water sports');



INSERT INTO team_swimmer(Swimmer_ID,Team_ID)
VALUES ('1','1');

INSERT INTO team_swimmer(Swimmer_ID,Team_ID)
VALUES ('2','1');

INSERT INTO team_swimmer(Swimmer_ID,Team_ID)
VALUES ('4','2');

INSERT INTO team_swimmer(Swimmer_ID,Team_ID)
VALUES ('6','2');

INSERT INTO team_swimmer(Swimmer_ID,Team_ID)
VALUES ('8','3');

INSERT INTO team_swimmer(Swimmer_ID,Team_ID)
VALUES ('9','3');



INSERT INTO venue (Name, Opening_hours, Street_Name , Area_Code , Street_Number)
VALUES ('Pretoria Swim club','06:30','Prickett','2169','12');

INSERT INTO venue (Name, Opening_hours, Street_Name , Area_Code , Street_Number)
```

VALUES ('Planet Fitness Randburg','05:30','Malibongwe','2144','546');

INSERT INTO event (Event_Name,Num_Races)
VALUES ('Gauteng qualifying',12);

INSERT INTO event (Event_Name,Num_Races)
VALUES ('Wet boys friendly meet',3);

INSERT INTO event (Event_Name,Num_Races)
VALUES ('Semi-Finals',4);

INSERT INTO venue_event (Venue_ID,Event_ID)
VALUES (1,1);

INSERT INTO venue_event (Venue_ID,Event_ID)
VALUES (2,2);

INSERT INTO venue_event (Venue_ID,Event_ID)
VALUES (1,3);

INSERT INTO Event_swimmer (Event_ID , Swimmer_ID)
VALUES (1,2);

INSERT INTO Event_swimmer (Event_ID , Swimmer_ID)
VALUES (1,3);

INSERT INTO Event_swimmer (Event_ID , Swimmer_ID)
VALUES (1,4);

INSERT INTO Event_swimmer (Event_ID , Swimmer_ID)
VALUES (1,5);

INSERT INTO Event_swimmer (Event_ID , Swimmer_ID)
VALUES (1,6);

INSERT INTO Event_swimmer (Event_ID , Swimmer_ID)

```
VALUES (1,7);

INSERT INTO Event_swimmer (Event_ID , Swimmer_ID)
VALUES (1,8);

INSERT INTO Event_swimmer (Event_ID , Swimmer_ID)
VALUES (1,9);

INSERT INTO Event_swimmer (Event_ID , Swimmer_ID)
VALUES (1,10);

INSERT INTO Event_swimmer (Event_ID , Swimmer_ID)
VALUES (2,2);

INSERT INTO Event_swimmer (Event_ID , Swimmer_ID)
VALUES (2,3);

INSERT INTO Event_swimmer (Event_ID , Swimmer_ID)
VALUES (2,5);

INSERT INTO Event_swimmer (Event_ID , Swimmer_ID)
VALUES (3,2);

INSERT INTO Event_swimmer (Event_ID , Swimmer_ID)
VALUES (3,4);

INSERT INTO Event_swimmer (Event_ID , Swimmer_ID)
VALUES (3,6);

INSERT INTO Event_swimmer (Event_ID , Swimmer_ID)
VALUES (3,7);

INSERT INTO Event_swimmer (Event_ID , Swimmer_ID)
VALUES (3,8);

INSERT INTO Event_swimmer (Event_ID , Swimmer_ID)
VALUES (3,9);



INSERT INTO pool (Venue_ID,Pool_name,Num_lanes)
VALUES (1,'Pool A',4);

INSERT INTO pool (Venue_ID,Pool_name,Num_lanes)
```

```
VALUES (1,'Pool B',4);

INSERT INTO pool (Venue_ID,Pool_name,Num_lanes)
VALUES (1,'Pool c',8);

INSERT INTO pool (Venue_ID,Pool_name,Num_lanes)
VALUES (1,'Practice pool',2);

INSERT INTO pool (Venue_ID,Pool_name,Num_lanes)
VALUES (2,'Main',6);



INSERT INTO race (Pool_ID,Event_ID,Date,Distance,Stroke_Type)
VALUES (1,1,'2022-02-14',50,'Butterfly');

INSERT INTO race (Pool_ID,Event_ID,Date,Distance,Stroke_Type)
VALUES (1,1,'2022-02-14',50,'Back');

INSERT INTO race (Pool_ID,Event_ID,Date,Distance,Stroke_Type)
VALUES (2,1,'2022-02-14',50,'Freestyle');

INSERT INTO race (Pool_ID,Event_ID,Date,Distance,Stroke_Type)
VALUES (2,1,'2022-02-14',100,'Breast');

INSERT INTO race (Pool_ID,Event_ID,Date,Distance,Stroke_Type)
VALUES (2,1,'2022-02-14',100,'Back');

INSERT INTO race (Pool_ID,Event_ID,Date,Distance,Stroke_Type)
VALUES (3,1,'2022-02-14',100,'Freestyle');

INSERT INTO race (Pool_ID,Event_ID,Date,Distance,Stroke_Type)
VALUES (3,1,'2022-02-14',50,'Breast');

INSERT INTO race (Pool_ID,Event_ID,Date,Distance,Stroke_Type)
VALUES (3,1,'2022-02-14',50,'Butterfly');

INSERT INTO race (Pool_ID,Event_ID,Date,Distance,Stroke_Type)
VALUES (1,1,'2022-02-14',50,'Back');

INSERT INTO race (Pool_ID,Event_ID,Date,Distance,Stroke_Type)
VALUES (1,1,'2022-02-14',50,'Freestyle');

INSERT INTO race (Pool_ID,Event_ID,Date,Distance,Stroke_Type)
```

```
VALUES (1,1,'2022-02-14',100,'Breast');

INSERT INTO race (Pool_ID,Event_ID,Date,Distance,Stroke_Type)
VALUES (5,2,'2022-02-22',50,'Practice relay');

INSERT INTO race (Pool_ID,Event_ID,Date,Distance,Stroke_Type)
VALUES (5,2,'2022-02-22',100,'Practice relay');

INSERT INTO race (Pool_ID,Event_ID,Date,Distance,Stroke_Type)
VALUES (5,3,'2022-02-22',50,'Freestyle');

INSERT INTO race (Pool_ID,Event_ID,Date,Distance,Stroke_Type)
VALUES (5,3,'2022-02-22',50,'Breast');

INSERT INTO race (Pool_ID,Event_ID,Date,Distance,Stroke_Type)
VALUES (5,3,'2022-02-22',100,'Freestyle');

INSERT INTO race (Pool_ID,Event_ID,Date,Distance,Stroke_Type)
VALUES (5,3,'2022-02-22',100,'Breast');


INSERT INTO race_swimmer
(Race_ID,Swimmer_ID,Swimmer_Time,Swimmer_Position,Team_Position)
VALUES (8,10,'00:00:55',3,null);

INSERT INTO race_swimmer
(Race_ID,Swimmer_ID,Swimmer_Time,Swimmer_Position,Team_Position)
VALUES (7,7,'00:03:22',1,3);

INSERT INTO race_swimmer
(Race_ID,Swimmer_ID,Swimmer_Time,Swimmer_Position,Team_Position)
VALUES (3,4,'00:00:55',3,1);


INSERT INTO race_swimmer
(Race_ID,Swimmer_ID,Swimmer_Time,Swimmer_Position,Team_Position)
VALUES (5,4,'00:01:55',2,1);


INSERT INTO race_swimmer
(Race_ID,Swimmer_ID,Swimmer_Time,Swimmer_Position,Team_Position)
VALUES (9,3,'00:01:23',1,1)
```

Task 7:

Query 1: Show all the races a specific swimmer participates in.

```
SELECT race.Event_ID, race_swimmer.Swimmer_ID, race.Race_ID, race.Pool_ID,
race.Distance, race.Stroke_Type, race.date

FROM race_swimmer, race

WHERE race_swimmer.Race_ID = race.Race_ID AND Swimmer_ID = 4;
```

A where clause join is called a cartesian join, with this join all-possible combinations of tuples are created when the query is run. In the example above all possible combinations of a race and the swimmer will be created. This is extremely inefficient and wastes computational resources.

The same result but more efficient query, which will only output the needed data.

```
SELECT race.Event_ID, race_swimmer.Swimmer_ID, race.Race_ID, race.Pool_ID,
race.Distance, race.Stroke_Type, race.date

FROM race_swimmer INNER JOIN race ON race_swimmer.Race_ID = race.Race_ID

WHERE race_swimmer.Swimmer_ID = 4;
```

Query 2: Selects all the swimmers participate in an event. Inefficient query:

```
SELECT *

FROM event_swimmer INNER JOIN swimmer ON event_swimmer.Swimmer_ID =
swimmer.Swimmer_ID

WHERE     event_swimmer.Event_ID = 1;
```

Selecting all the columns (SELECT *) in the result set will waste system resources.

Select the specific columns in the result set instead of selecting all of them.

Efficient query:

```
SELECT event_swimmer.Event_ID, swimmer.Swimmer_ID, swimmer.Fname,
swimmer.Lname, swimmer.sex

FROM event_swimmer INNER JOIN swimmer ON event_swimmer.Swimmer_ID =
swimmer.Swimmer_ID

WHERE     event_swimmer.Event_ID = 1;
```

Thus, selecting specific attributes in the result set will use less system resources for the same result, making it more efficient.

Query 3: Select all the swimmers that are in teams.

Inefficient query:

```
SELECT Fname, Lname, Sex
```

```
FROM swimmer
WHERE exists (
SELECT *
FROM team_swimmer
WHERE team_swimmer.Swimmer_ID = swimmer.Swimmer_ID
);
```

In this nested query it takes the team which the swimmer is in as a parameter and returns true if the swimmer is a team. This is inefficient because it will evaluate every tuple of the outside query.

Nesting queries is inefficient.

Efficient query:

```
SELECT Fname, Lname, Sex
FROM swimmer, team_swimmer
WHERE team_swimmer.Swimmer_ID = swimmer.Swimmer_ID;
```

This new query will make sure that there is a tuple from the tale swimmer will match with at most 1 tuple from the relation department.