ICT285 Databases
Assignment

Name: Styverson Ng Yu Hao

Table of Contents

Question 1: Relational Algebra

ATHLETE (<u>AthleteNo</u>, AthleteName, **CountryName**)
COUNTRY (<u>CountryName</u>, NumberOfCompetitors)
EVENT (<u>EventName</u>, ScheduledStart, **VenueName**)
VENUE (<u>VenueName</u>, Location, Capacity)
FINAL (**<u>AthleteNo, EventName</u>**, Place, Medal)

Q1: $\pi_{\text{AthleteName, CountryName}}$ (Athlete)

Q2: $\pi_{\text{EventName, ScheduledStart}}$ ($\sigma_{\text{VenueName = 'Velodrome'}}$ (Event))

Q3: $\pi_{\text{AthleteName}}$ (Athlete $\bowtie_{\text{Athlete.AthleteNo = Final.AthleteNo}}$ Final (Final $\bowtie_{\text{FINAL.EventName = EVENT.EventName}}$ Event ($\sigma_{\text{VenueName = 'Rio de Janeiro' OR VenueName = 'Sao Paolo'}}$ (Event))))

Q4: $\pi_{\text{AthleteName}}$ ($\sigma_{\text{CountryName = 'Brazil' AND Medal = 'Gold'}}$ (Athlete $\bowtie_{\text{Athlete.AthleteNo = Final.AthleteNo}}$ Final))

Q5: $\pi_{\text{EventName, ScheduledStart, VenueName}}$ ($\sigma_{\text{Capacity > 50,000}}$ (Event $\bowtie_{\text{Event.VenueName = Venue.VenueName}}$ Venue))

Q6: $\pi_{\text{EventName, VenueName, Location}}$ (Venue LEFT OUTER JOIN $_{\text{Venue.VenueName = Event.VenueName}}$ Event)

Q7: $\pi_{\text{AthleteName}}$ ($\sigma_{\text{Medal = 'Gold' AND VenueName = 'Aquatic Stadium' AND CountryName = 'Australia'}}$ (Athlete $\bowtie_{\text{Athlete.AthleteNo = Final.AthleteNo}}$ Final (Final $\bowtie_{\text{FINAL.EventName = EVENT.EventName}}$ Event)

Q8: $\pi_{\text{AthleteName}}$ ($\sigma_{\text{EventName = 'Men's 100m'}}$ ($\sigma_{\text{Place = '1'}}$ (Athlete $\bowtie_{\text{Athlete.AthleteNo = Final.AthleteNo}}$ Final (Final $\bowtie_{\text{FINAL.EventName = EVENT.EventName}}$ Event))))
$\bowtie$
$\pi_{\text{AthleteName}}$ ($\sigma_{\text{EventName = 'Men's 200m'}}$ ($\sigma_{\text{Place = '1'}}$ (Athlete $\bowtie_{\text{Athlete.AthleteNo = Final.AthleteNo}}$ Final (Final $\bowtie_{\text{FINAL.EventName = EVENT.EventName}}$ Event))))

Q9: $\pi_{\text{AthleteName}}$ ($\sigma_{\text{EventName = 'Women's 1500m Freestyle'}}$ (Athlete $\bowtie_{\text{Athlete.AthleteNo = Final.AthleteNo}}$ Final (Final $\bowtie_{\text{FINAL.EventName = EVENT.EventName}}$ Event)))

MINUS

$\pi_{\text{AthleteName}}$ ($\sigma_{\text{Medal = 'Gold' OR Medal = 'Silver' OR Medal = 'Bronze'}}$ ($\sigma_{\text{EventName = 'Women's 1500m Freestyle'}}$ (Athlete $\bowtie_{\text{Athlete.AthleteNo = Final.AthleteNo}}$ Final (Final $\bowtie_{\text{FINAL.EventName = EVENT.EventName}}$ Event))))

Q10: $\pi_{\text{AthleteName}}$ (Athlete $\bowtie$ Athlete.AthleteNo = Final.AthleteNo (Final $\div$ ($\sigma_{\text{Medal = 'Gold'}}$ Final))

Question 2: SQL – Select Queries

1. List the details of any work of art (including the name of the artist who created the work) that has ‘Signed’ in their description.

STUDENT.sql

SQL Worksheet | History

Worksheet | Query Builder

```sql
-- Select all work details along with the artist's name
SELECT Work.WorkID, Work.Title, Work.Medium, Work.Description, Artist.FirstName, Artist.LastName

-- Combines the work and artist table together using the common attribute "ArtistID"
FROM tutorials.Work INNER JOIN tutorials.ARTIST ON Work.ArtistID = Artist.ArtistID

-- Filter works that have "Signed" in the description
WHERE Work.Description LIKE '%Signed%';
```

Query Result

SQL | All Rows Fetched: 8 in 0.284 seconds

	WORKID	TITLE	MEDIUM	DESCRIPTION	FIRSTNAME	LASTNAME
1	537	The Woven World	Color lithograph	Signed	Mark	Tobey
2	570	Untitled Number 1	Monotype with tempera	4.3 x 6.1 in. Signed	Mark	Tobey
3	580	Forms in Progress I	Color aquatint	19.3 x 24.4 in. - Signed	Mark	Tobey
4	581	Forms in Progress II	Color aquatint	19.3 x 24.4 in. - Signed	Mark	Tobey
5	571	Yellow Covers Blue	Oil and collage	71 x 78 in. - Signed	Paul	Horiuchi
6	548	Night Bird	Watercolor on Paper	50 x 72.5 cm. - Signed	Morris	Graves
7	561	Sunflower	Watercolor and ink	33.3 x 16.1 cm. - Signed	Morris	Graves
8	593	Surf and Bird	Gouache	26.5 x 29.75 in. - Signed	Morris	Graves

2. List all the nationalities with more than one artist represented in the database, and the number of artists of that nationality.

STUDENT.sql

SQL Worksheet | History

Worksheet | Query Builder

```sql
-- Select the nationality and count the number of artists for each nationality
SELECT Artist.Nationality, COUNT(Artist.ArtistID) AS NumberOfArtists
FROM tutorials.ARTIST
-- Group by nationality
GROUP BY Artist.Nationality
-- Filter to show only nationalities with more than one artist
HAVING COUNT(Artist.ArtistID) > 1;
```

Query Result

SQL | All Rows Fetched: 3 in 0.282 seconds

	NATIONALITY	NUMBEROFARTISTS
1	Spanish	3
2	French	2
3	United States	4

3. List the number of works in each medium, ordered from highest to lowest number.

STUDENT.sql

SQL Worksheet History

Worksheet Query Builder

```sql
-- Select the medium and count the number of works in each
SELECT Work.Medium, COUNT(Work.WorkID) AS NumberOfWorks
FROM tutorials.WORK

-- Group the results by medium
GROUP BY Work.Medium

-- Order the results from highest to lowest count
ORDER BY NumberOfWorks DESC;
```

Query Result

SQL | All Rows Fetched: 10 in 0.281 seconds

	MEDIUM	NUMBEROFWORKS
1	High Quality Limited Print	26
2	Color aquatint	2
3	Watercolor and ink	1
4	Gouache	1
5	Tempera on card	1
6	Watercolor on Paper	1
7	Monotype with tempera	1
8	Oil and collage	1
9	Color lithograph	1
10	Casein rice paper collage	1

4. List the names of all the customers and the names of the artists each customer has an interest in, in alphabetical order of artist last name within customer last name.

STUDENT.sql

SQL Worksheet | History

Worksheet | Query Builder

```sql
-- Select customer and artist details for customers with interest in artists
SELECT Artist.LastName AS ArtistLastName, Artist.FirstName AS ArtistFirstName,
Customer.LastName AS CustomerLastName, Customer.FirstName AS CustomerFirstName
FROM tutorials.CUSTOMER
-- Relate customer and customer_artist_int table together using "CustomerID"
JOIN tutorials.CUSTOMER_ARTIST_INT ON CUSTOMER.CustomerID = CUSTOMER_ARTIST_INT.CustomerID
-- Relate customer and customer_artist_int table together using "ArtistID"
JOIN tutorials.ARTIST ON CUSTOMER_ARTIST_INT.ArtistID = ARTIST.ArtistID
-- Arrange table by alphabetical order based on the artist's last name than by customer last name
ORDER BY ArtistLastName, CustomerLastName;
```

Query Result

SQL | All Rows Fetched: 35 in 0.285 seconds

	ARTISTLASTNAME	ARTISTFIRSTNAME	CUSTOMERLASTNAME	CUSTOMERFIRSTNAME
1	Basher	Bigmouth	Smith	David
2	Chagall	Marc	Frederickson	Mary Beth
3	Chagall	Marc	Smith	David
4	Chagall	Marc	Warning	Selma
5	Graves	Morris	Gray	Donald
6	Graves	Morris	Janes	Jeffrey
7	Graves	Morris	Smathers	Fred
8	Graves	Morris	Smith	David
9	Graves	Morris	Twilight	Tiffany
10	Graves	Morris	Warning	Selma
11	Graves	Morris	Wilkens	Chris
12	Horiuchi	Paul	Gray	Donald
13	Horiuchi	Paul	Janes	Jeffrey
14	Horiuchi	Paul	Smathers	Fred
15	Horiuchi	Paul	Smith	David
16	Horiuchi	Paul	Twilight	Tiffany
17	Horiuchi	Paul	Wilkens	Chris
18	Julio	Bloxham Smythe	Smith	David
19	Kandinsky	Wassily	Frederickson	Mary Beth
20	Kandinsky	Wassily	Smith	David
21	Klee	Paul	Smith	David
22	Matisse	Henri	Frederickson	Mary Beth
23	Matisse	Henri	Smith	David
24	Miro	Joan	Frederickson	Mary Beth
25	Miro	Joan	Smith	David
26	Sargent	John Singer	Gliddens	Melinda
27	Sargent	John Singer	Smith	David
28	Sargent	John Singer	Twilight	Tiffany
29	Sargent	John Singer	Warning	Selma
30	Tobey	Mark	Gray	Donald
31	Tobey	Mark	Janes	Jeffrey
32	Tobey	Mark	Smathers	Fred
33	Tobey	Mark	Smith	David
34	Tobey	Mark	Twilight	Tiffany
35	Tobey	Mark	Wilkens	Chris

5. List the full name and email of any customers who have no address recorded.

STUDENT.sql

SQL Worksheet | History

Worksheet | Query Builder

```
-- Select customer names and email to display
SELECT Customer.LastName, Customer.FirstName, Customer.Email
FROM tutorials.CUSTOMER

-- Create filter condition to find the customer with NULL address
WHERE Customer.Street IS NULL AND Customer.City IS NULL
AND Customer.State IS NULL AND Customer.ZipPostalCode IS NULL;
```

Query Result

SQL | All Rows Fetched: 1 in 0.282 seconds

	LASTNAME	FIRSTNAME	EMAIL
1	Gliddens	Melinda	Melinda.Gliddens@somewhere.com

6. List the work ID, title and artist name of all the works of art that sold for more than the average sales price, and the price they sold for.

STUDENT.sql

SQL Worksheet | History

Worksheet | Query Builder

```
-- Select work and artist details for works that sold for more than the average sales price
SELECT Work.WorkID, Work.Title, Artist.FirstName, Artist.LastName, Trans.SalesPrice
FROM tutorials.Work
-- Combines artist and work table using "ArtistID"
JOIN tutorials.ARTIST ON Work.ArtistID = ARTIST.ArtistID
-- Combines trans and work table using "WorkID"
JOIN tutorials.TRANS ON Work.WorkID = TRANS.WorkID
-- Filter out works that sold for more than the average sales price
WHERE Trans.SalesPrice > (SELECT AVG(SalesPrice) FROM tutorials.TRANS);
```

Query Result

SQL | All Rows Fetched: 6 in 0.284 seconds

	WORKID	TITLE	FIRSTNAME	LASTNAME	SALESPRICE
1	570	Untitled Number 1	Mark	Tobey	13750
2	500	Memories IV	Paul	Horiuchi	42500
3	500	Memories IV	Paul	Horiuchi	72500
4	571	Yellow Covers Blue	Paul	Horiuchi	55000
5	548	Night Bird	Morris	Graves	27500
6	561	Sunflower	Morris	Graves	17500

7. List the full name of any customers who haven't bought any works of art.

STUDENT.sql

SQL Worksheet | History

Worksheet | Query Builder

```sql
-- Select and display full name of customers
SELECT Customer.LastName, Customer.FirstName
FROM tutorials.CUSTOMER
-- Use LEFT OUTER JOIN, reutrning all rows from tutorials.CUSTOMER and all matching rows from tutorials.TRANS,
-- based on "CustomerID"
LEFT OUTER JOIN tutorials.TRANS ON CUSTOMER.CustomerID = TRANS.CustomerID
-- Filter out all the customers with no transactions
WHERE Trans.CustomerID IS NULL;
```

Query Result

SQL | All Rows Fetched: 2 in 0.283 seconds

	LASTNAME	FIRSTNAME
1	Wu	Susan
2	Johnson	Lynda

8. Which artist (give his/her full name) has the most customers interested in him or her, and how many customers are interested in them?

STUDENT.sql

SQL Worksheet | History

Worksheet | Query Builder

```sql
-- Select artist details and the count of customers interested in each artist
SELECT Artist.FirstName, Artist.LastName, COUNT(Customer_Artist_Int.CustomerID) AS NumberOfCustomers
FROM tutorials.ARTIST
-- Join customer_artist_int and artist table together using "ArtistID"
JOIN tutorials.CUSTOMER_ARTIST_INT ON ARTIST.ArtistID = CUSTOMER_ARTIST_INT.ArtistID
-- Group by artist to
GROUP BY Artist.FirstName, Artist.LastName
-- Create condition to only display the artist with the most number of customers
-- Using the operator of MAX to locate the maximum value of COUNT(Customer_Artist_Int.CustomerID)
-- I equate maximum value of COUNT(Customer_Artist_Int.CustomerID) and the displayed COUNT(Customer_Artist_Int.CustomerID)
-- Hence the condition where only the maximum number of customers from a single artist is displayed on the table
HAVING COUNT(Customer_Artist_Int.CustomerID)
= (SELECT MAX(NumberOfCustomers)
   FROM (SELECT COUNT(Customer_Artist_Int.CustomerID) AS NumberOfCustomers
         FROM tutorials.ARTIST
         JOIN tutorials.CUSTOMER_ARTIST_INT ON ARTIST.ArtistID = CUSTOMER_ARTIST_INT.ArtistID
         GROUP BY Artist.ArtistID)
   );
```

Query Result

SQL | All Rows Fetched: 1 in 0.286 seconds

	FIRSTNAME	LASTNAME	NUMBEROFCUSTOMERS
1	Morris	Graves	7

9. List the total dollar amount of sales each artist (give his/her full name) has made on their works, in descending order of total.

STUDENT.sql

SQL Worksheet | History

Worksheet | Query Builder

```sql
-- Select artist details and the total sales amount for their works
SELECT Artist.FirstName, Artist.LastName, SUM(Trans.SalesPrice) AS TotalSales
FROM tutorials.ARTIST
-- Join work and artist table using "ArtistID"
JOIN tutorials.WORK ON ARTIST.ArtistID = WORK.ArtistID
-- Join work and trans table using "WorkID"
JOIN tutorials.TRANS ON WORK.WorkID = TRANS.WorkID
-- Group by artist
GROUP BY Artist.ArtistID, Artist.FirstName, Artist.LastName
-- Eliminate artist that did not sell any works
HAVING SUM(Trans.SalesPrice) IS NOT NULL
-- Arrange the table by descending total sales
ORDER BY SUM(Trans.SalesPrice) DESC;
```

Script Output | Query Result

SQL | All Rows Fetched: 8 in 0.282 seconds

	FIRSTNAME	LASTNAME	TOTALSALES
1	Paul	Horiuchi	170875
2	Morris	Graves	45500
3	Mark	Tobey	28625
4	John Singer	Sargent	925
5	Henri	Matisse	800
6	Marc	Chagall	675
7	Joan	Miro	600
8	Wassily	Kandinsky	400

10. List the name of any customers who have an interest in all the artists from the United States.

```sql
-- Select and display the Customer's full name
SELECT Customer.FirstName, Customer.LastName -- COUNT(Artist.ArtistID) AS Number_Of_Interested_Artist_From_US -> Used to verify the number of interested artist of each customer
FROM tutorials.CUSTOMER_ARTIST_INT
-- Join the artist and customer_artist_int table using "ArtistID"
JOIN tutorials.ARTIST ON CUSTOMER_ARTIST_INT.ArtistID = Artist.ArtistID
-- Join the customer and customer_artist_int table using "CustomerID"
JOIN tutorials.CUSTOMER ON CUSTOMER_ARTIST_INT.CustomerID = Customer.CustomerID
-- Create a restriction condition where the artist's nationality has to be United States
WHERE Artist.Nationality LIKE '%United States%'
-- Group the table by the customers
GROUP BY Customer.FirstName, Customer.LastName
-- Create another restriction condition where only customers that have showed interest in ALL the artists from United States
-- Count the number of artist each customers are interest that are from United States
-- With the total number of artists from United States, by comparing the number of artist each customers are interest that are from United States,
-- We are able to filter out the customers that have shown interest to ALL the artists from United States
HAVING COUNT(Artist.ArtistID) =
       -- Count the total number of artists from United States to compare against the customer's number
      (SELECT COUNT(Artist.ArtistID)
       FROM tutorials.ARTIST
       WHERE Artist.Nationality = 'United States');
```

All Rows Fetched: 2 in 0.286 seconds

	FIRSTNAME	LASTNAME
1	David	Smith
2	Tiffany	Twilight

Question 3: Further SQL

1. Create a VENUE table. Choose appropriate data types. None of the attributes should be allowed to be null. Include the primary key constraint.

STUDENT.sql

SQL Worksheet History

Worksheet Query Builder

```sql
-- Create a table with the name "VENUE"
CREATE TABLE VENUE (
    --  VenueName and Location is defined as a VARCHAR data types with 100 and 200 characters respectively
    -- Capacity is deinfed to be an INT as it holds a numerical value
    -- All datas are NOT NULL
    -- VenueName is assigned to be the PRIMARY KEY
    VenueName VARCHAR(100) NOT NULL,
    Location VARCHAR(200) NOT NULL,
    Capacity INT NOT NULL,
    PRIMARY KEY (VenueName)
);
```

Script Output Query Result

Task completed in 0.341 seconds

```
Table VENUE created.
```

STUDENT.sql

SQL Worksheet History

Worksheet Query Builder

```sql
-- Create a table with the name "VENUE"
CREATE TABLE VENUE (
    --  VenueName and Location is defined as a VARCHAR data types with 100 and 200 characters respectively
    -- Capacity is deinfed to be an INT as it holds a numerical value
    -- All datas are NOT NULL
    -- VenueName is assigned to be the PRIMARY KEY
    VenueName VARCHAR(100) NOT NULL,
    Location VARCHAR(200) NOT NULL,
    Capacity INT NOT NULL,
    PRIMARY KEY (VenueName)
);

SELECT *
FROM VENUE;
```

Script Output Query Result

SQL All Rows Fetched: 0 in 0.543 seconds

VENUENAME	LOCATION	CAPACITY

2. Create an EVENT table. Use appropriate data types and include the primary key and foreign key constraints. A venue may not be deleted from the database if there is an event recorded in it.

STUDENT.sql

SQL Worksheet | History

Worksheet | Query Builder

```sql
-- Create a table named "EVENT"
CREATE TABLE EVENT (
-- EventName and VenueName are both VARCHAR data types that hold up 100 characters
-- ScheduledStart has a TIMESTEMP data type containing both date and time
-- All datas are not allowed to be NULL
-- EventName is assigned to be the PRIMARY KEY
-- VenueName is assigned to be the FOREIGN KEY from VENUE table
-- Since VenueName is a FOREIGN KEY, Oracle's default behaviour has implicitly restricted its deletions from this table
    EventName VARCHAR(150) NOT NULL,
    ScheduledStart TIMESTAMP NOT NULL,
    VenueName VARCHAR(100) NOT NULL,
    PRIMARY KEY (EventName),
    FOREIGN KEY (VenueName) REFERENCES VENUE(VenueName)
);
```

Script Output | Task completed in 0.304 seconds

```
Table EVENT created.
```

STUDENT.sql

SQL Worksheet | History

Worksheet | Query Builder

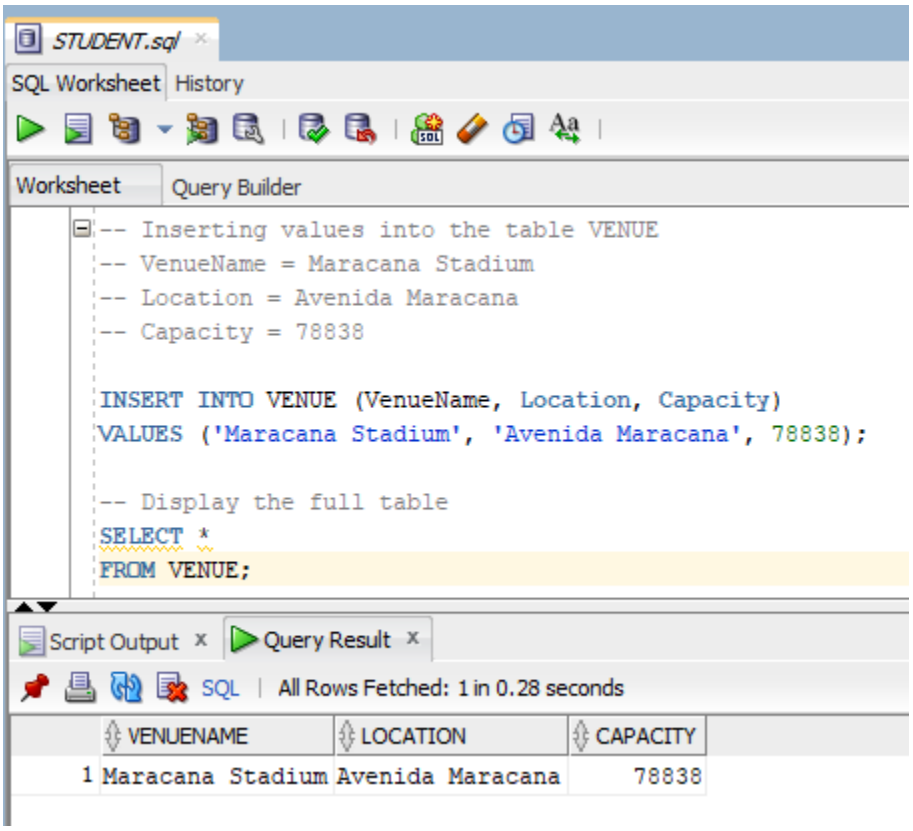
```sql
-- Create a table named "EVENT"
CREATE TABLE EVENT (
-- EventName and VenueName are both VARCHAR data types that hold up 100 characters
-- ScheduledStart has a TIMESTEMP data type containing both date and time
-- All datas are not allowed to be NULL
-- EventName is assigned to be the PRIMARY KEY
-- VenueName is assigned to be the FOREIGN KEY from VENUE table
-- Since VenueName is a FOREIGN KEY, Oracle's default behaviour has implicitly restricted its deletions from this table
    EventName VARCHAR(150) NOT NULL,
    ScheduledStart TIMESTAMP NOT NULL,
    VenueName VARCHAR(100) NOT NULL,
    PRIMARY KEY (EventName),
    FOREIGN KEY (VenueName) REFERENCES VENUE(VenueName)
);

-- Display the full table
SELECT *
FROM EVENT;
```

Script Output | Query Result

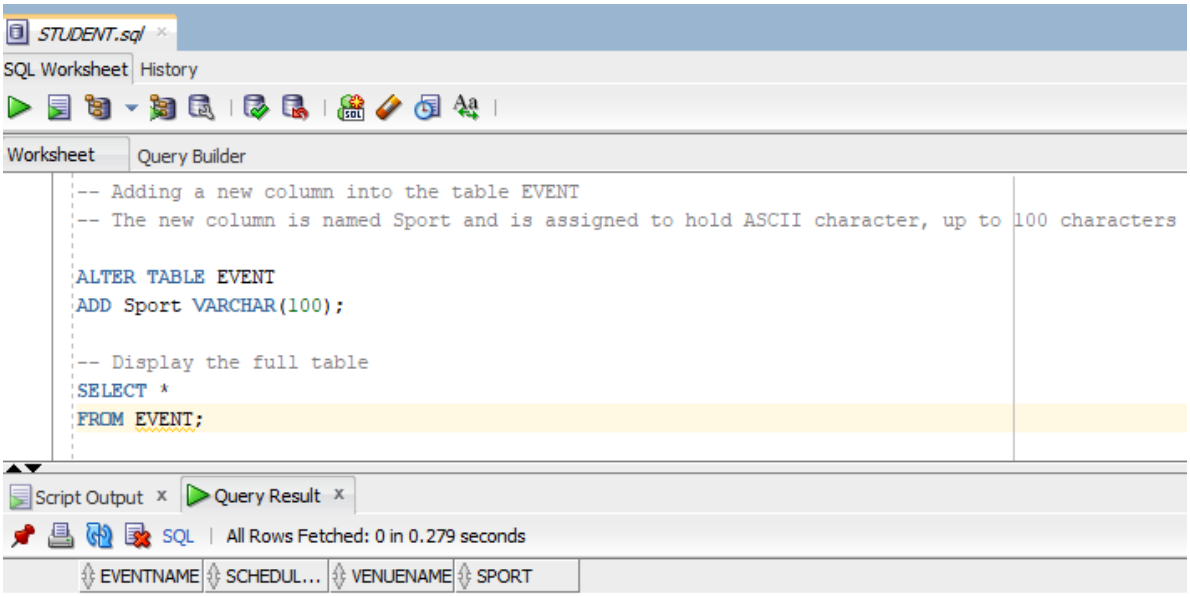
SQL | All Rows Fetched: 0 in 0.306 seconds

EVENTNAME	SCHEDULEDSTART	VENUENAME

3. Adding Maracana Stadium to the VENUE table, located in Avenida Maracana and has a capacity of 78,838.



4. Adding an attribute Sport to EVENT. (Possible values include Athletics, Swimming, Tennis, etc).



5. Updating the Maracana Stadium's capacity to 80,000.

STUDENT.sql

SQL Worksheet History

Worksheet Query Builder

```sql
-- Updating a value in the table VENUE
-- Previously the capacity was set to only 78838
-- The new value for capacity is 80000 for Maracana Stadium
UPDATE VENUE
SET Capacity = 80000
WHERE VenueName = 'Maracana Stadium';

-- Display the full table
SELECT *
FROM VENUE;
```

Script Output Query Result

SQL | All Rows Fetched: 1 in 0.279 seconds

	VENUENAME	LOCATION	CAPACITY
1	Maracana Stadium	Avenida Maracana	80000

Question 4: Normalization

The current design of the Car-Service relation table presents several significant problems that affect both its current functionality and future scalability. These issues primarily include data redundancy, data inconsistency, and a lack of normalization.

Firstly, information such as CustName, CustDOB, and Staff Name are repeated throughout the table. For instance, when a customer has multiple service appointments, their details are duplicated across several records within the same table. This redundancy increases storage needs and also introduces the risk of data inconsistency, as updating one record requires updating all instances of that customer's information. Failing to do so can result in contradictory information, where the same customer is recorded with different birthdates, names, or addresses.

Second, there is inconsistency introduced due to repeated entries of customer and vehicle information. Errors could be made if, for instance, the same customer data is entered with incorrect information or various spellings. Additionally, updating one item requires manually updating all related entries, which further raises the possibility of errors.

Third, the table violates key principles of database normalization, such as Second Normal Form (2NF) and Third Normal Form (3NF). There are partial dependencies where non-key attributes like CustDOB depend on non-primary attributes such as CustName and transitive dependencies where attributes like CarRego, CarMake, and CarModel depend on customer information rather than being stored in their own separate entities. These problems result in data management that is ineffective and prone to errors.

Furthermore, the Service Fee column's use of both numeric and alphabetic representations makes accurate calculation difficult. These inconsistencies complicate financial reporting and data analysis.

By separating the data into several related tables, the new design reduces storage needs and avoids data duplication by eliminating the repetition of client and vehicle information. Each piece of information is kept in one spot, making it easier to insert and update data. This ensures consistency and lowers the possibility of errors. Moreover, it is simpler to manage the smaller, more focused tables, which enables more effective updates and scalable growth when the database introduces new clients, vehicles, and services. Finally, financial calculations such as total service costs become precise and effective by standardizing the Service Fee column as a numeric value, enabling better reporting and analysis.

The structure of the new tables should be as such:

Table Name: Customer

Column Names and Data Types:

CustomerID INT PRIMARY KEY	CustName VARCHAR(50)	CustDOB DATE

Table Name: Car

Column Names and Data Types:

CarID INT PRIMARY KEY	CarRego VARCHAR(20)	CarMake VARCHAR(30)	CarModel VARCHAR(30)	CustomerID INT FOREIGN KEY REFERENCES Customer(CustomerID)

Table Name: Staff

Column Names and Data Types:

StaffID INT PRIMARY KEY	StaffName VARCHAR(50)	StaffSpeciality VARCHAR(50)

Table Name: Service

Column Names and Data Types:

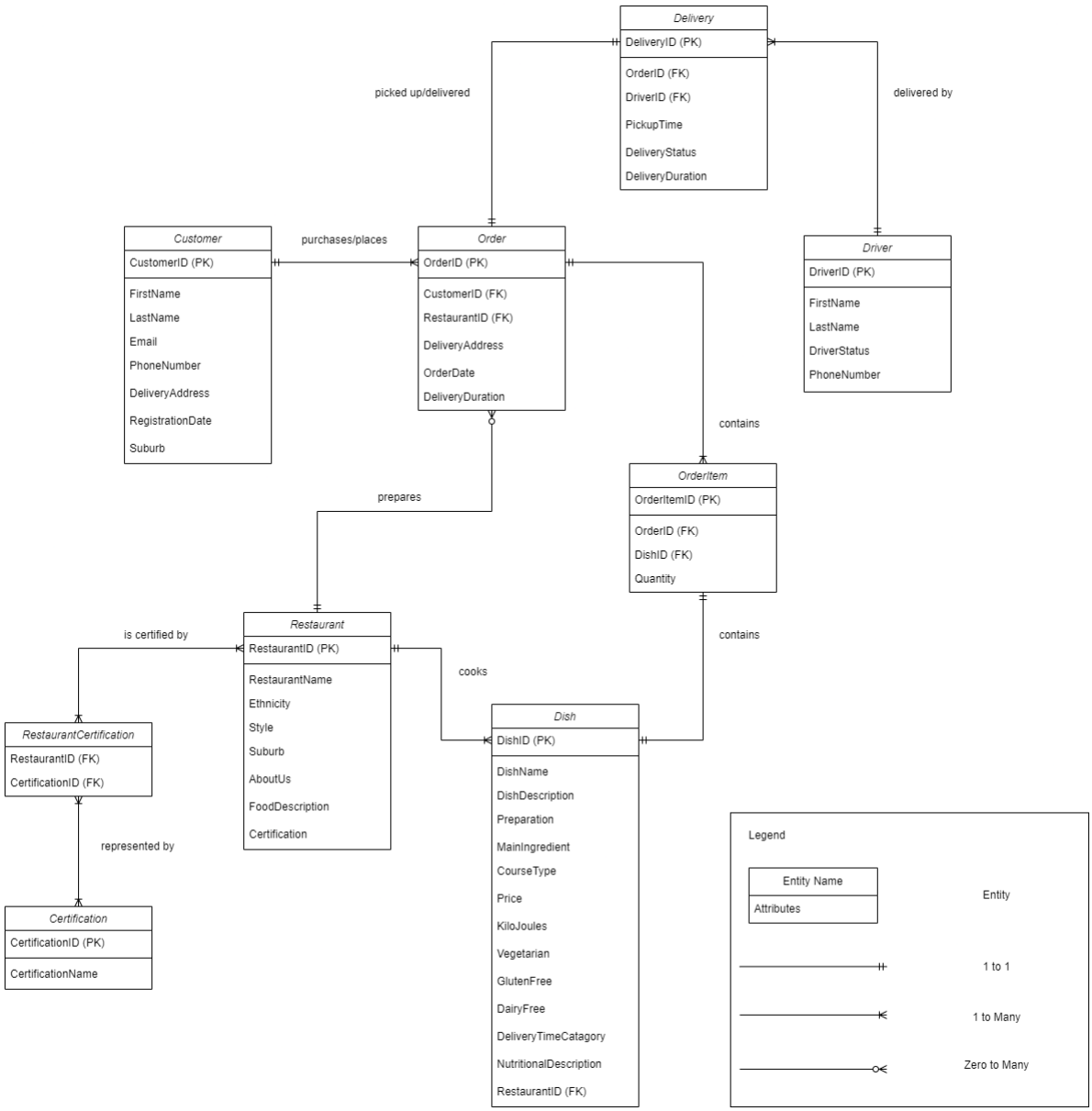
ServiceID INT PRIMARY KEY	ServiceDate DATE	ServiceFee DECIMAL (10, 2),	CustomerID INT FOREIGN KEY REFERENCES Customer(CustomerID)	CarID INT FOREIGN KEY REFERENCES Car(CarID)	StaffID INT FOREIGN KEY (StaffID) REFERENCES Staff(StaffID)

Table Name: Car-Service Relation

Column Names and Data Types:

CustomerID INT PRIMARY KEY	ServiceID INT PRIMARY KEY	StaffID INT FOREIGN KEY REFERENCES Staff(StaffID)	CarID INT FOREIGN KEY REFERENCES Car(CarID)

Question 5: Conceptual Design



Assumptions:

- Customers, or at least the delivery address, must located in the same suburb as the restaurant for delivery to adhere to the 10 minutes delivery timing slogan.
- Only a single order can be placed from each restaurant at one time, with an order containing multiple dishes.
- The driver's status, "Available" or "On delivery", is assumed to be automatically updated once an order has been completed.
- "DeliveryTimeCategory" is assigned a status of "Fast", "Regular" or "Worth the Wait" based on the dish's preparation time, since delivery duration has to be at most 10 minutes.
- Restaurants are able to be certified with multiple certificates at a single time.
- Drivers are automatically assigned delivery pickups once their status is "Available" and when there is a pickup that is available.
- Customers are required to sign up to utilize the platform, registering their personal information before they can place an order.
- Payment methods are currently not needed, but is speculated to be associated with PayPal.