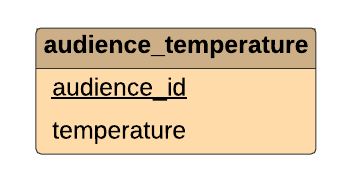
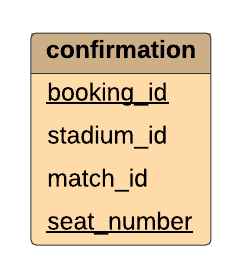
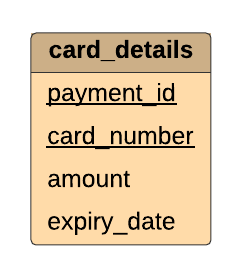
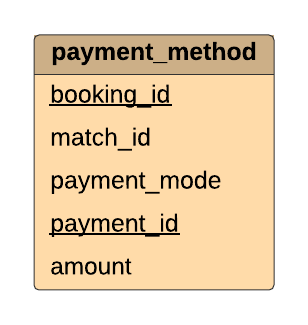
**Diary Writing**

**Meet Sheladiya – 19BIT076**

**Assignment – 2**

**Relational Model**

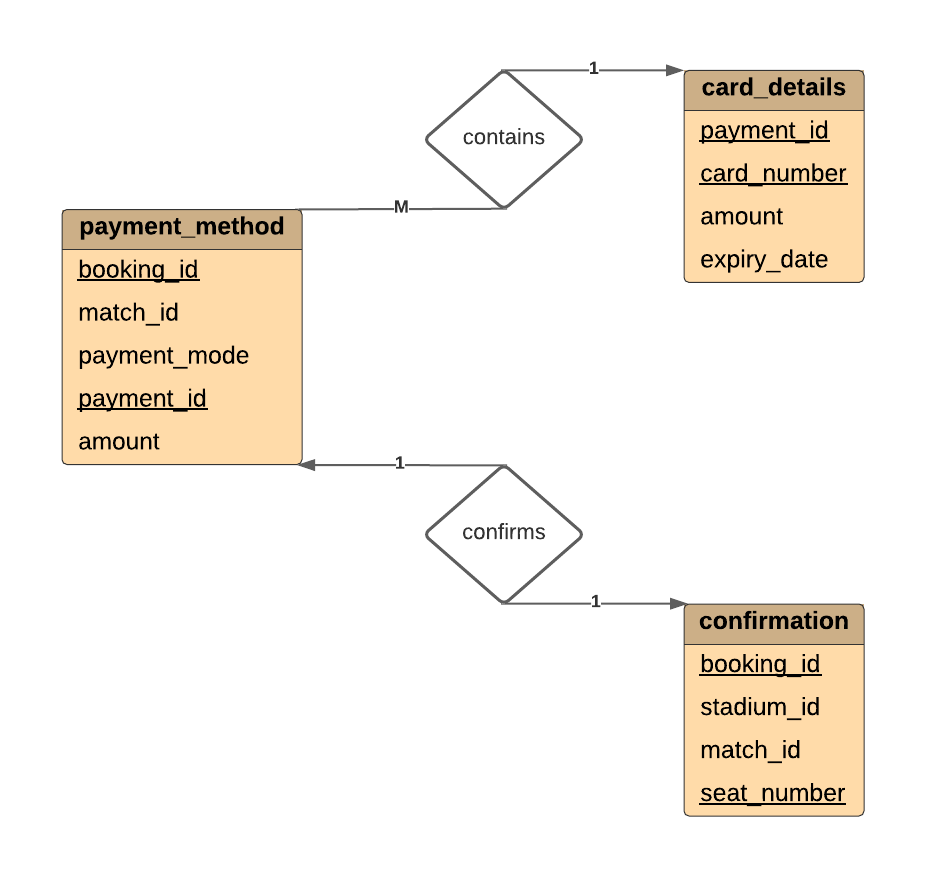
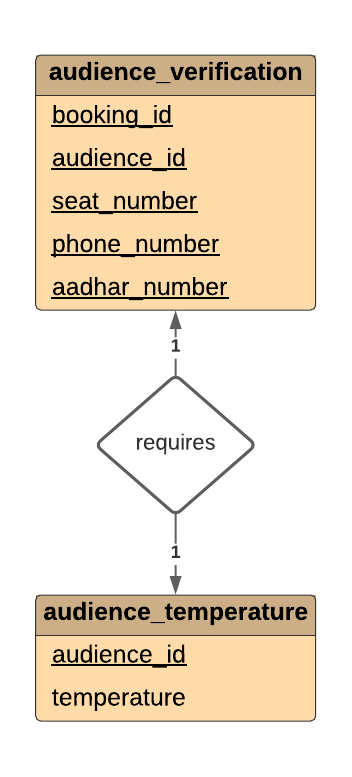
Well, the concept of this assignment i.e., to make a relational model of our data set was new to us so we were quite excited to perform this with the participation of whole group. Again, as a team we divided our work and started working on it. We were asked to give 4\*N tables therefore every member came up with 4 entities and their attributes, before finalising we had a group discussion on the entities provided by all the members to make the model unique and efficient. I worked upon **Payment method, card details, confirmation and audience temperature.** I made the tables with these entities and attributes and highlighted the respective Primary Key. We had a final discussion and the compilation was done along with the discussion and we also had a discussion over Foreign Key to relate the tables with each other. Finally, our Assignment 2 was completed and it was as it was supposed to be.



**Assignment – 3**

**ER Model**

This assignment was related to the previous one it was a complete team work as we were supposed to make the relations between the tables, like in my case the card details table was related to payment method and it was further related to confirmation after successful payment. The audience temperature was not related to above three therefore we made the relations of all the leftover tables through group discussion with all the members. And then a complete E-R model was made with the help of an online software – [Lucid chart](https://lucid.app/lucidchart/21c4d692-d021-4418-a7e8-7b0c9c5d97c6/edit?page=0_0)



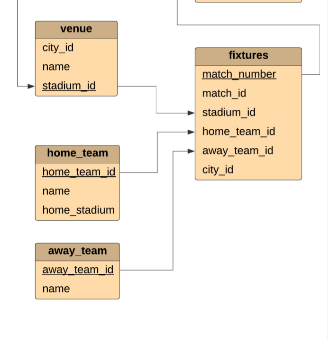
**Renish Jagani – 19BIT108**

**Assignment -2**

**Relational Model**

After our first assignment we got the second one and this time it was something which we never heard of before and everyone was little confused regarding the assignment 2, So we decided to have a discussion and then we discussed a lot amongst ourselves after taking the idea from our study material, Google and watched some YouTube Videos. After few discussions we got clarity of this particular Project, the task was to make a relational model of the data base system and there needs to be 4\*N tables so the distribution work took place and everyone got their fields.

I made the table’s **fixtures, venue, home\_ team, away\_team.** Th**e** task was to make the table using these entities and attributes and underlined the Primary key of each table. The last task was to decide the foreign key to relate all the tables to each other. At the end after so much of discussion and hard work our second assignment was ready to upload.



**Assignment – 3**

**ER Model**

After the second assignment the third one was co related to the second one, the task was to make out the relations between all the tables. Two out of four entities that I had were related the other two were not related so we had an online meet to relate the tables with the tables of other group members. We did a lot of discussions and worked out the things for a perfect model. We had a lot of contradictions but then after working out on them finally, the relations were made for all the tables and the task was done in an online platform.

Every group member played their role of completing the task on time, everyone helped each other in their doubts and confusions and this eventually increased our understanding and team spirit.

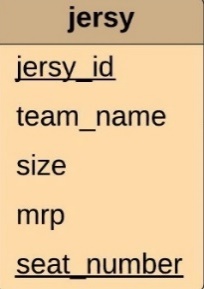
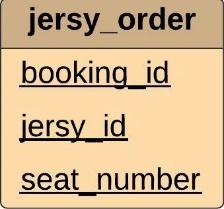
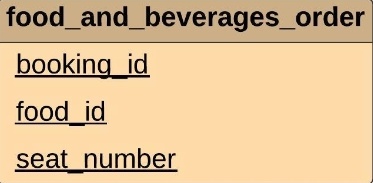
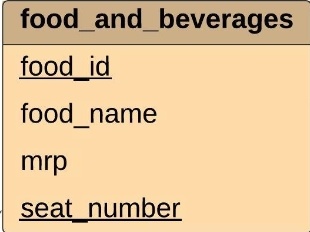


**Rishit Samariya – 19BIT109**

**Assignment-2**

**Relational Model**

This assignment was also a group assignment in which we have also develop group spirit and bonding. In this assignment every student have given idea of entity and there attribute we have called meeting in which every student have given 4 entity and there attribute to. If by chance it lead that there is common entity then we have solved that problem in group and it help us to figure it out. I am given the area to find entity and its attribute related food and beverage and jersey of team. So I have made the 4 table with their entity and attribute in which Primary key was underline.

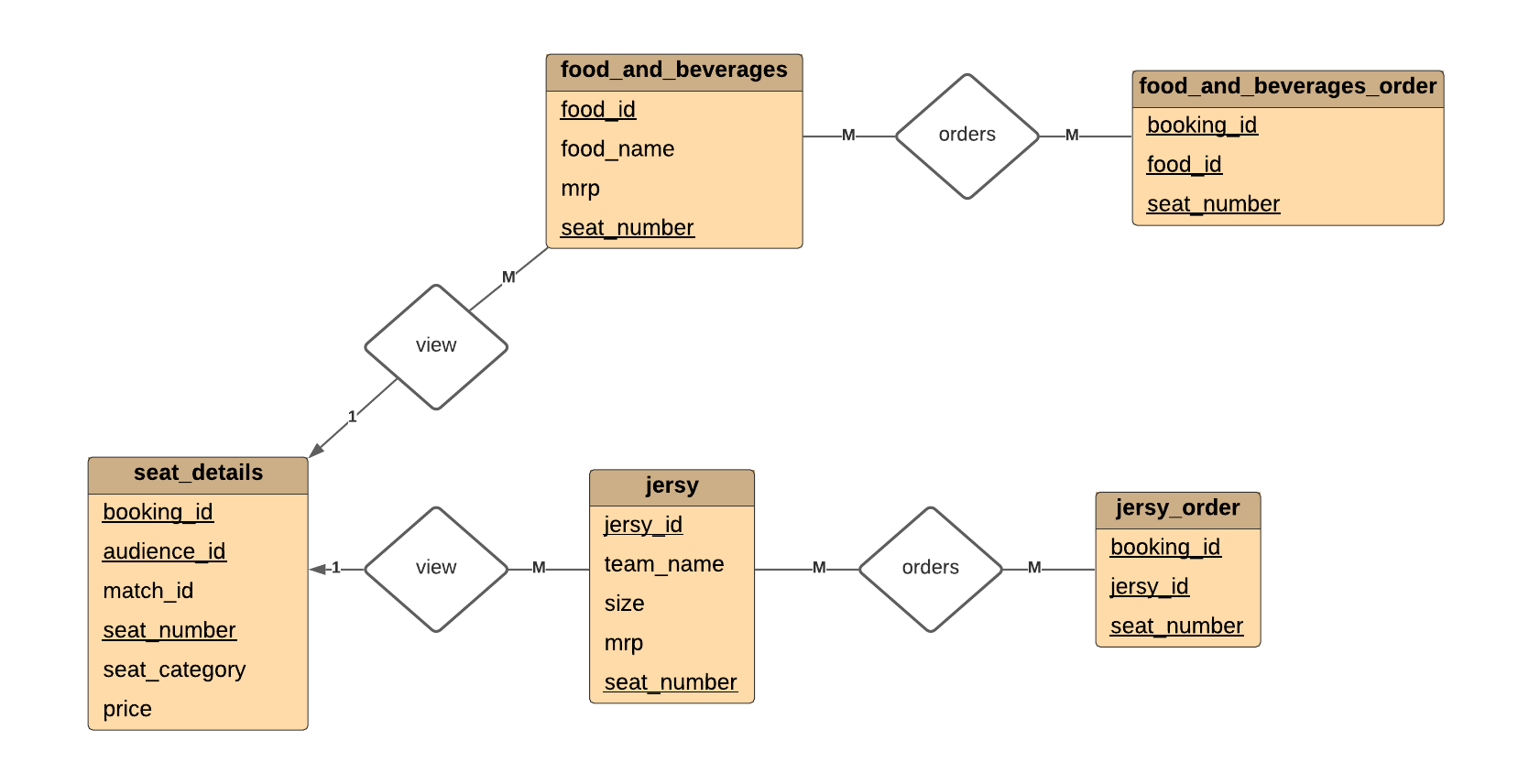
   

Then by gathering all detail by group member we have made the Relational model in group by giving the idea of foreign key To relate the table with each other. And we have finalise the Relatinal model in online platform.

**Assignment-3**

**E-R model**

This Assignment was based on assignment 2 which was Relational model after making of Relational model I have make relation between the tables which I have made to other. Between the jersey and jersey order there will be relation of order which will be denoted by diamond and entity will be denoted by rectangle. Same in Food and beverages and Food and its Food and Beverages order there will be same relation of order. And also given the relation between Seat Details Entity with Food and Beverages entity and Jersey entity the relation I have given was of view. Then we have given idea in Group and then we have made the diagram through online platform ([https://lucid.app/lucidchart/21c4d692-d021-4418-a7e8-7b0c9c5d97c6/edit?page=0\_0#).](https://lucid.app/lucidchart/21c4d692-d021-4418-a7e8-7b0c9c5d97c6/edit?page=0_0%23).%20) Then I have also given 1 feature of Stadium which is listed in the PPT that we have made. This all work has been finalised in the PPT.

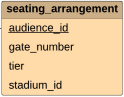
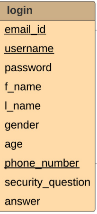
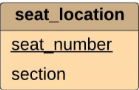
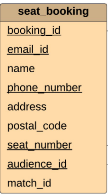
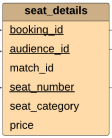


**Rishabh Patel - 19BIT141**

**Assignment – 2**

**Relational Model**

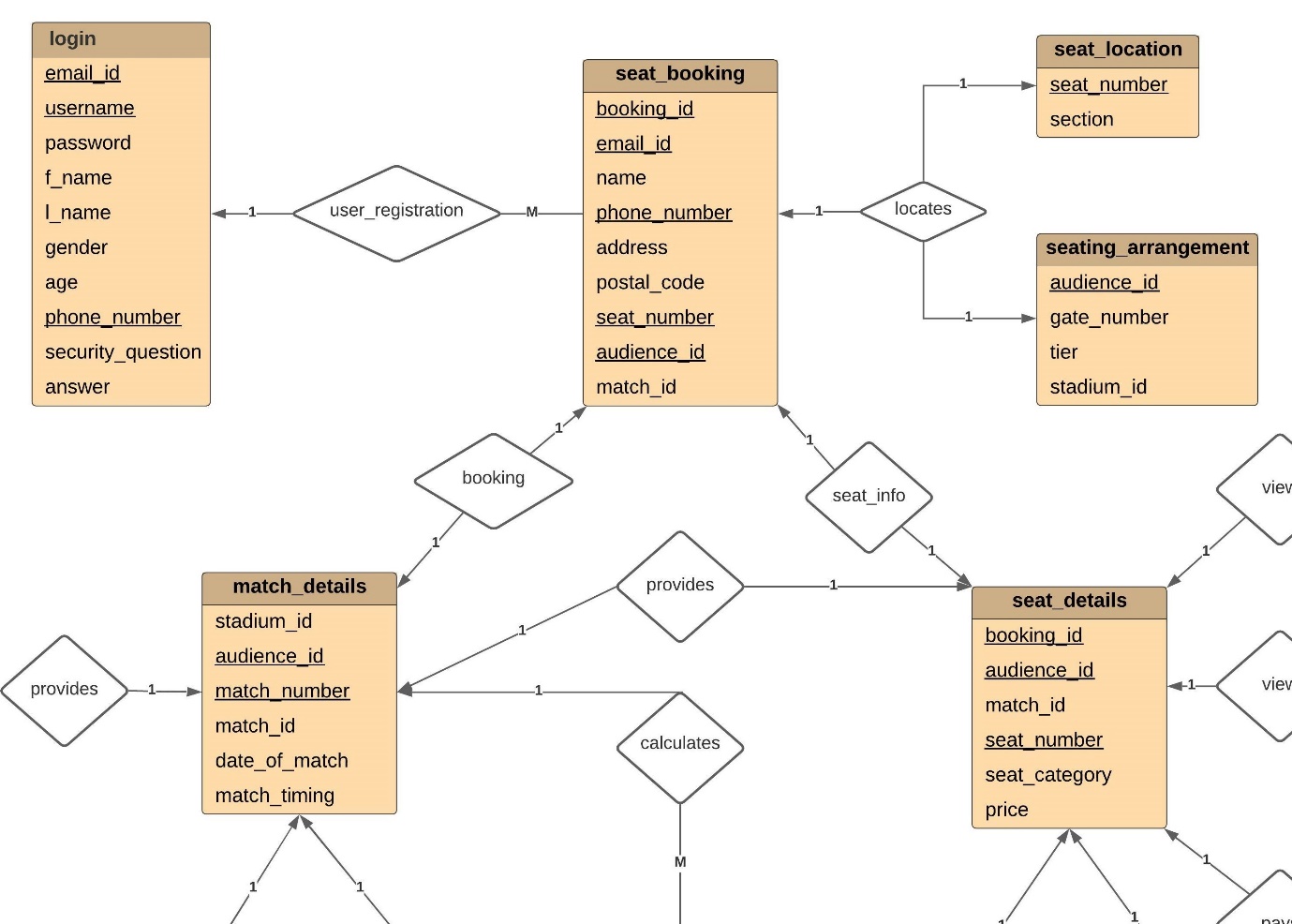
In the assignment – 2 we work on relational model where first I create my 4 input tables that are login table, seat booking table, seat location table, seat detail table and then we started making diagram in lucid chart where we first made rectangle tables for attributes and then we find super keys, primary key and foreign key. I underlined the attributes for primary key. Then I find relations between each of the four tables. By calling a meeting we solved the problems which occurs between each other input tables and came up with more relevant solution. Finally we evaluate each other work and merge the final work done by all team members.

**Assignment – 3**

**ER Model**

The Assignment – 3 is Entity Relational modeI and because Relational model is simpler model of ER model and so it is very convenient of us. I made the relation sets between the assigned input tables and created the entity sets for the attributes and shown the symbols used in ER model and represents the aggregation through schema. For Instance login and seat booking has relation set of user registration shown that one user can book many seats for each match. Similarly for seat booking and seat detail there one-to-one relation because for each seat there can only be single data in seat detail. We initially created ER model in Visual paradigm but then we moved to lucid chart because it provides more feature and it is more user interactive. Then we finally evaluate our work and merge final output of all team members.

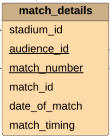
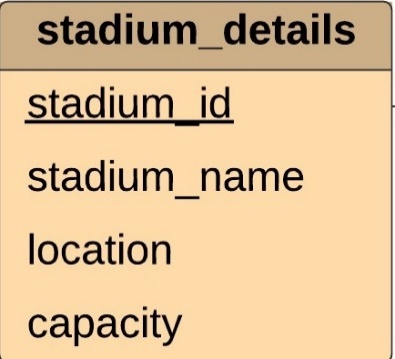


**Ujash Thakkar – 19BIT142**

**Assignment-2**

**Relational Model**

In This assignment every member creates 4 input table. And I create stadium\_details, match\_details, audience\_verification and tickets\_cancelled. In this relational model underlined attributes for primary key and every relational model has minimum at least one primary key. After creating 4 table I have to think about how all table are interrelated after the group discussion all doubt will be clear. After completing all the work by a group member that’s the time merge the final work.

**Assignment-3**

**E-R model**

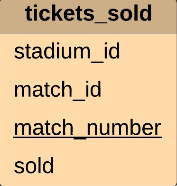
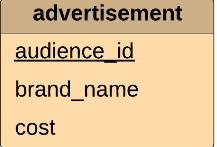
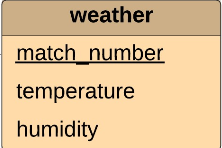
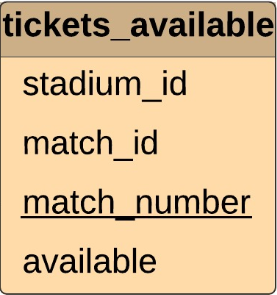
This Assignment was entity relationship model it’s quite easy because it based on assignment 2 which was Relational model after making of Relational model. After that I have to create relation sets between the assigned input tables and created the entity sets for the attributes. Between the stadium details and match details there will be relation of order which will be denoted by diamond that called by (provides) and entity will be denoted by rectangle. At the time when person was seat booking that time person see name of stadium and location of that and etc. The relationship between audience\_verification and seat\_details is verifies and the relationship between tickets\_cancelled and seat\_details is shows. Then we have given idea in Group and then we have made the diagram through online platform.

**Ananya Khandelwal – 19BIT144**

**Assignment – 2**

**Relational Model**

Our second Assignment was to make a relational model of our data base. This thing was new to us so we took some time to properly understand what a relational model is and how it is made we had a lot of group discussions about the relational model and when everyone was familiar with the model we began working in our respective provided field. We were supposed to make 4\*N tables, so the four I worked upon were **tickets sold, advertisement, weather, tickets available.**  I was not clear about how to make the tables so I discussed with my group and we concluded that the table is to be made with these entities and attributes and the primary key is to be mentioned or highlighted so I made the tables accordingly. Then at last we had a discussion on foreign key to relate the tables with each other. After a lot of discussion and improvements the Assignment 2 came to an end and was made perfectly. Our understanding of the subject was quite better than before.

**Assignment – 3**

**E-R Model**

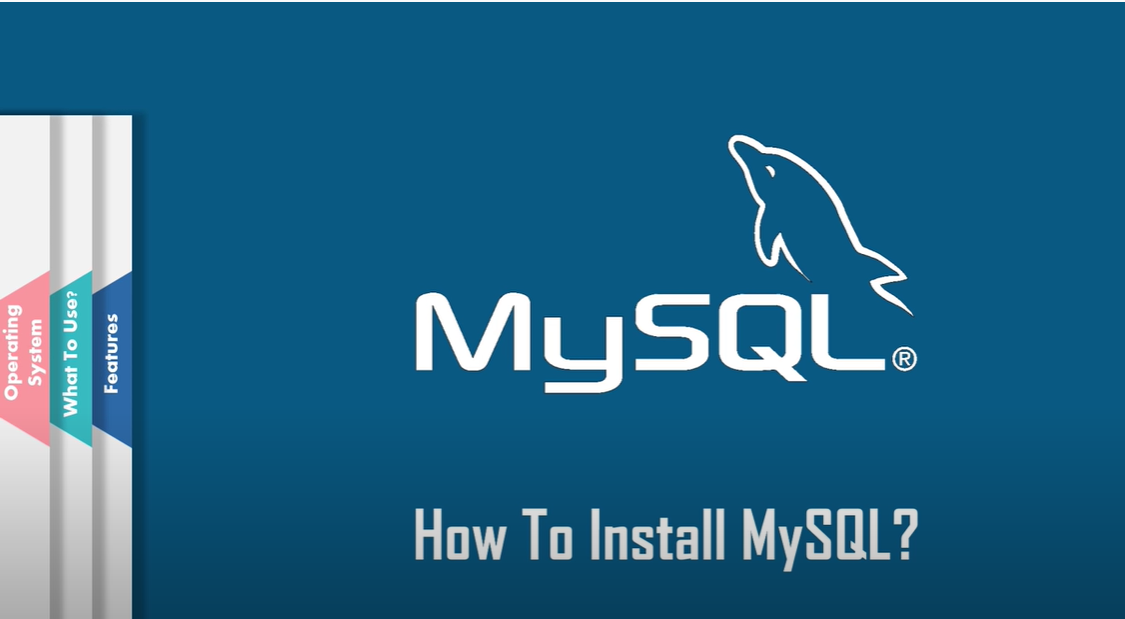
This assignment was just an upgrade of the previous one and it was completely a team work as we need to relate the tables with each other so we had arranged meetings accordingly and began the work of making the relations. Talking about my tables all the four were not related to each other therefore in our meetings we related each other’s table and came up with a proper ER Model and the model was created with the help of an online platform - [https://lucid.app/lucidchart/21c4d692-d021-4418-a7e8-7b0c9c5d97c6/edit?page=0\_0#](https://lucid.app/lucidchart/21c4d692-d021-4418-a7e8-7b0c9c5d97c6/edit?page=0_0%23)

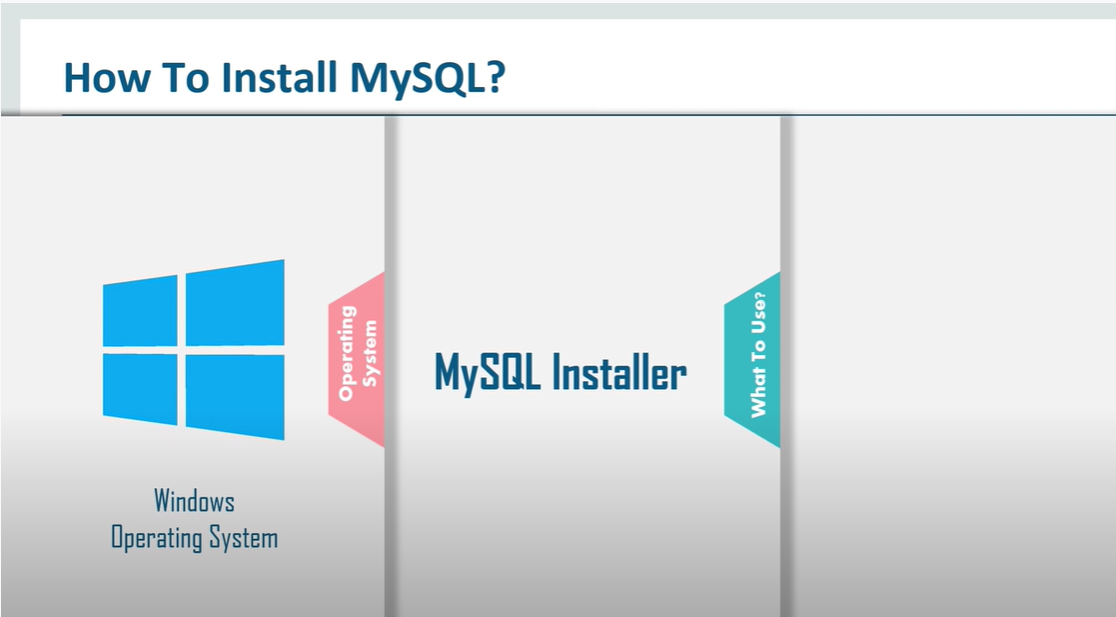


**Assignment – 5**

**Topic:- Installation of MySQL Relational Database Management System**

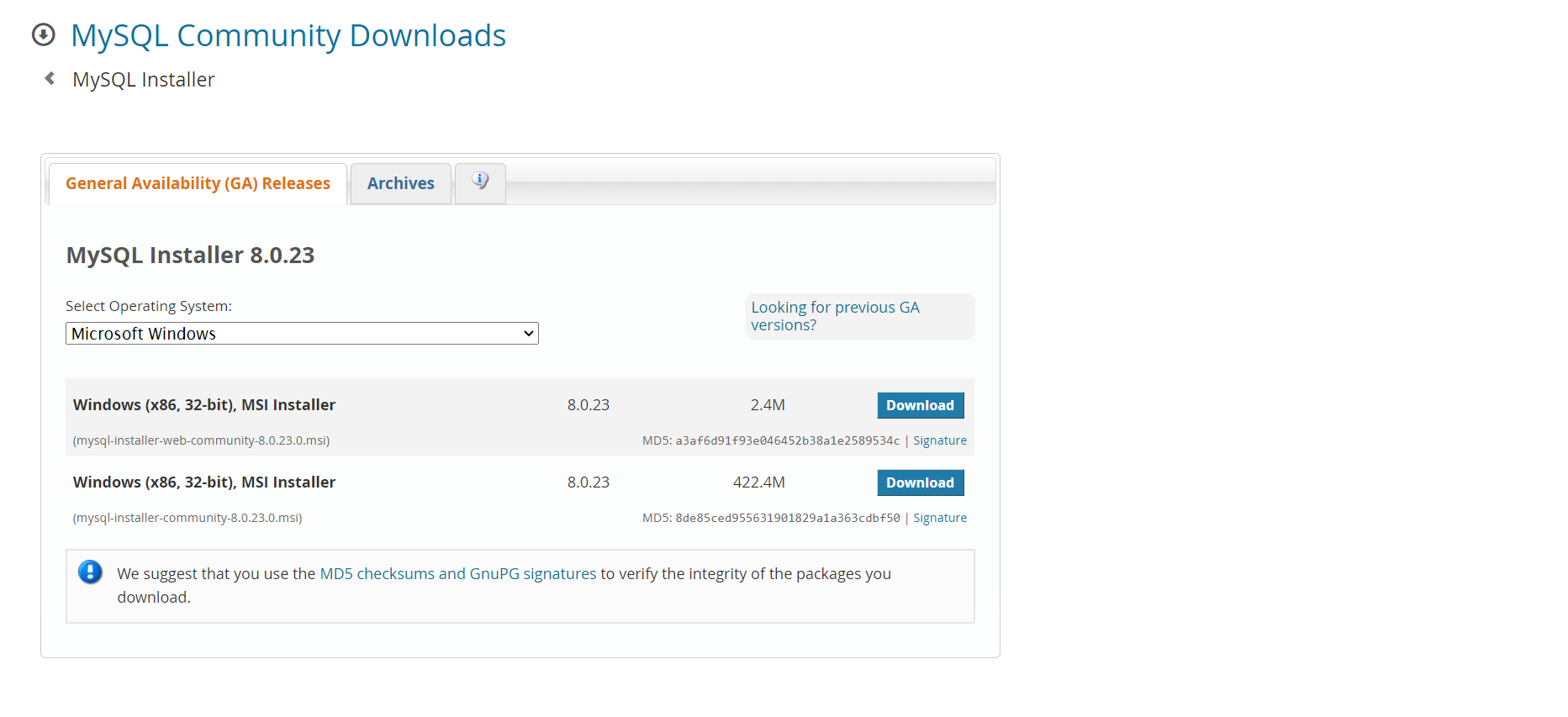
**Objective:-** To learn basics of MySQL software that can be used for SQL Query evaluations.



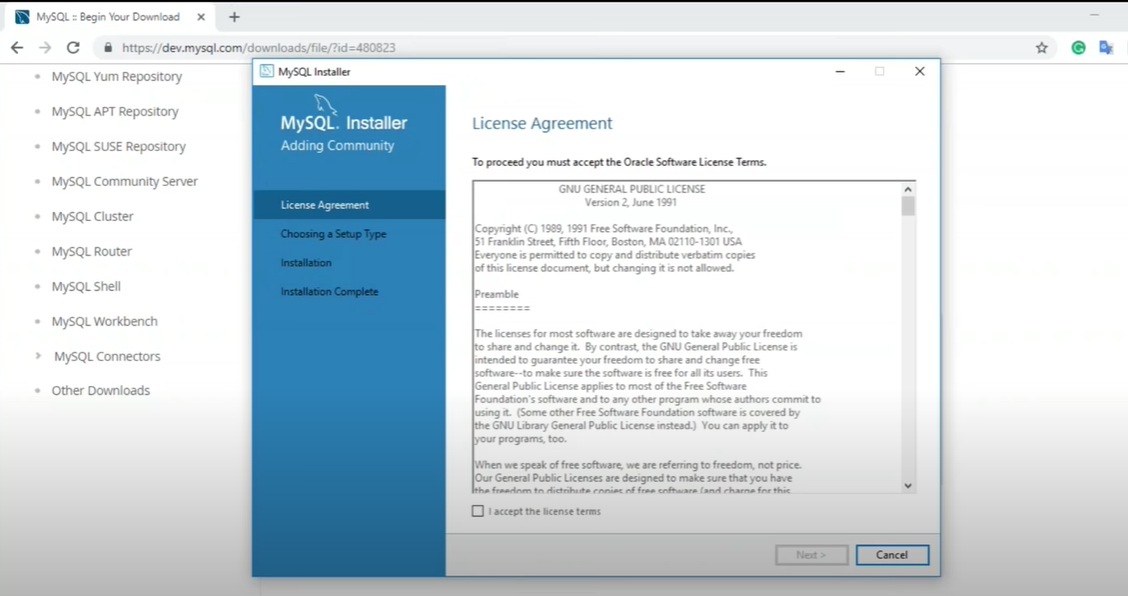


**Follow the Steps Below:-**

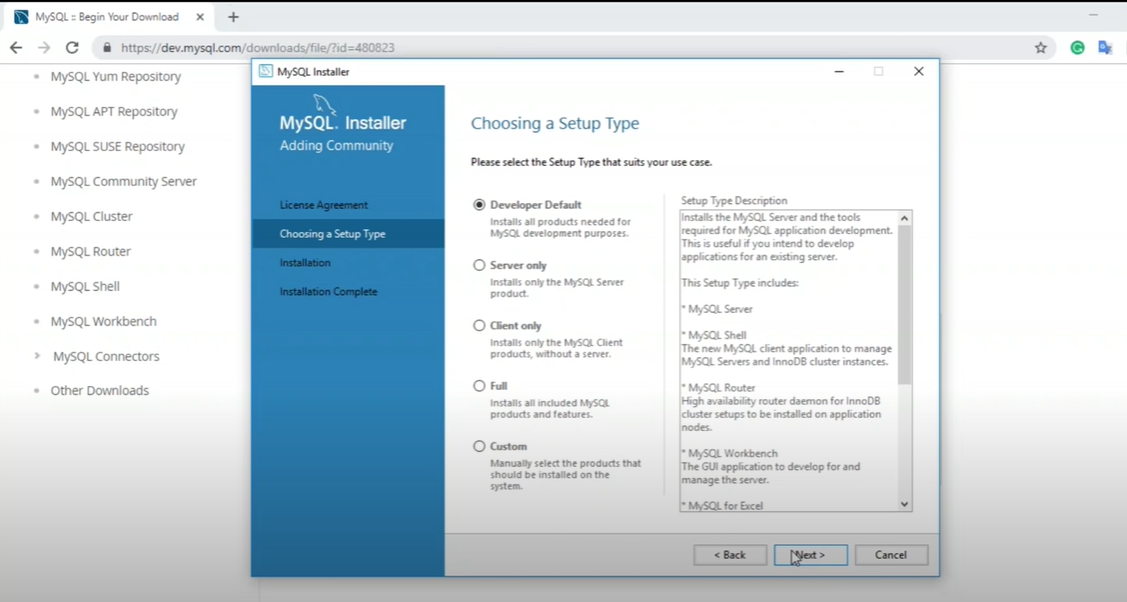
**STEP: - 1**



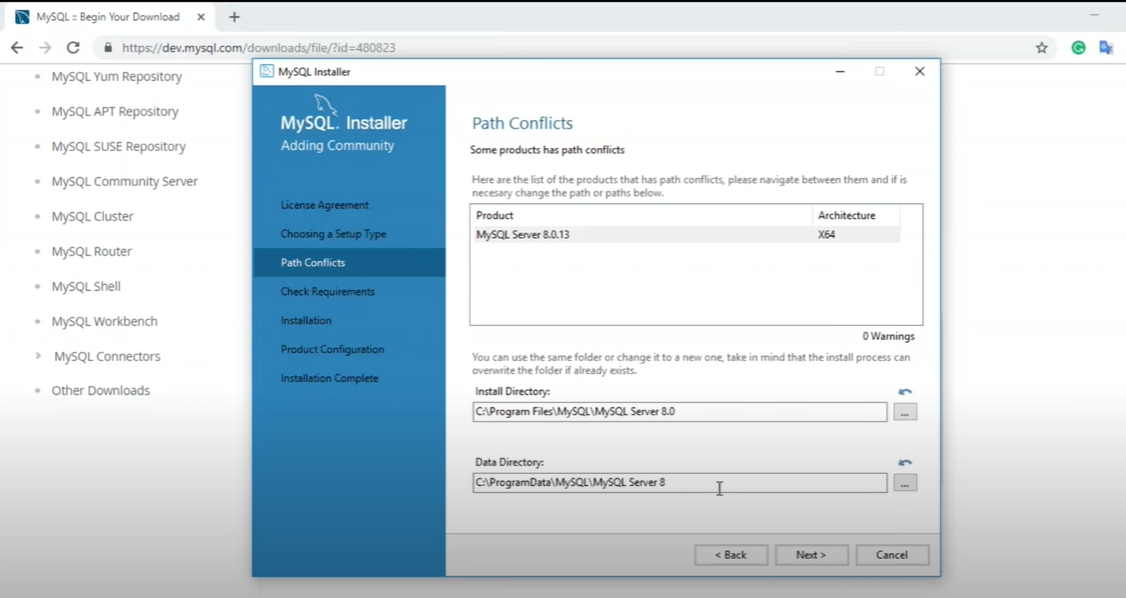
**STEP: - 2**



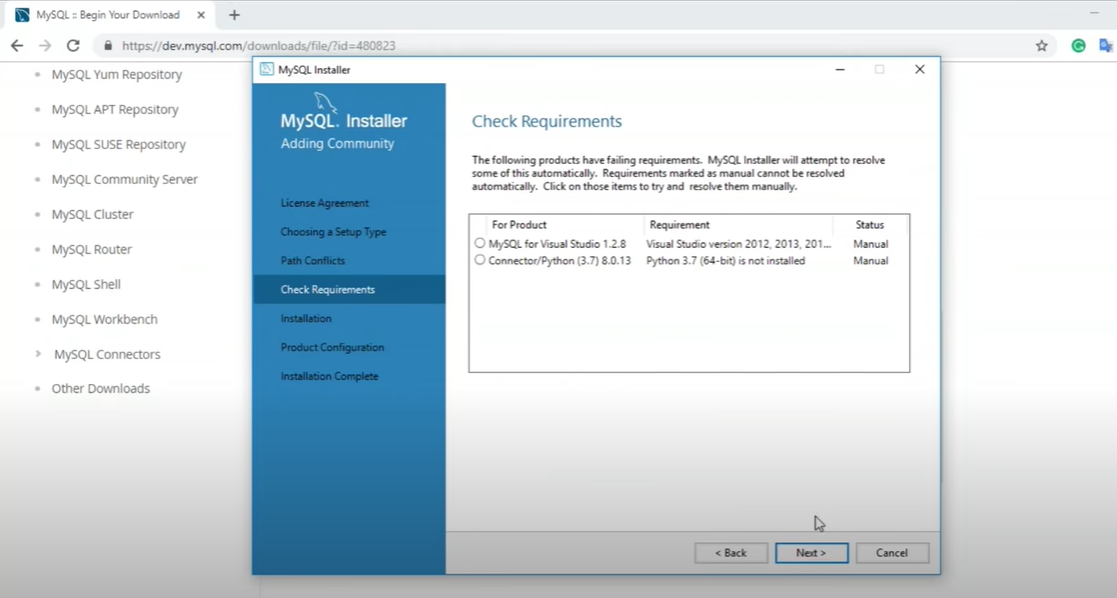
**STEP: - 3**



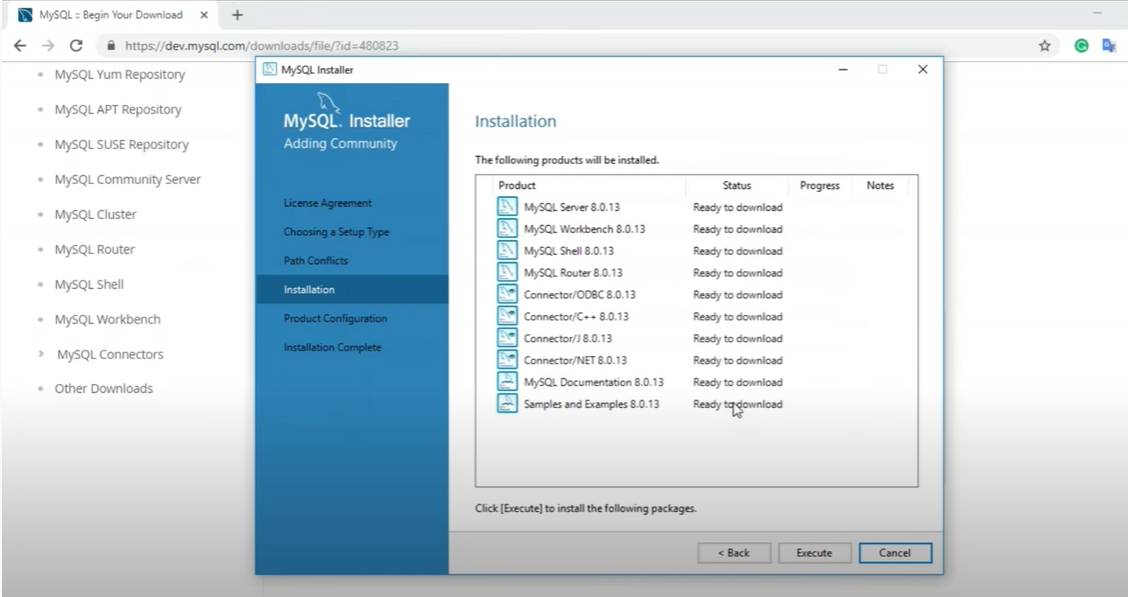
**STEP: - 4**



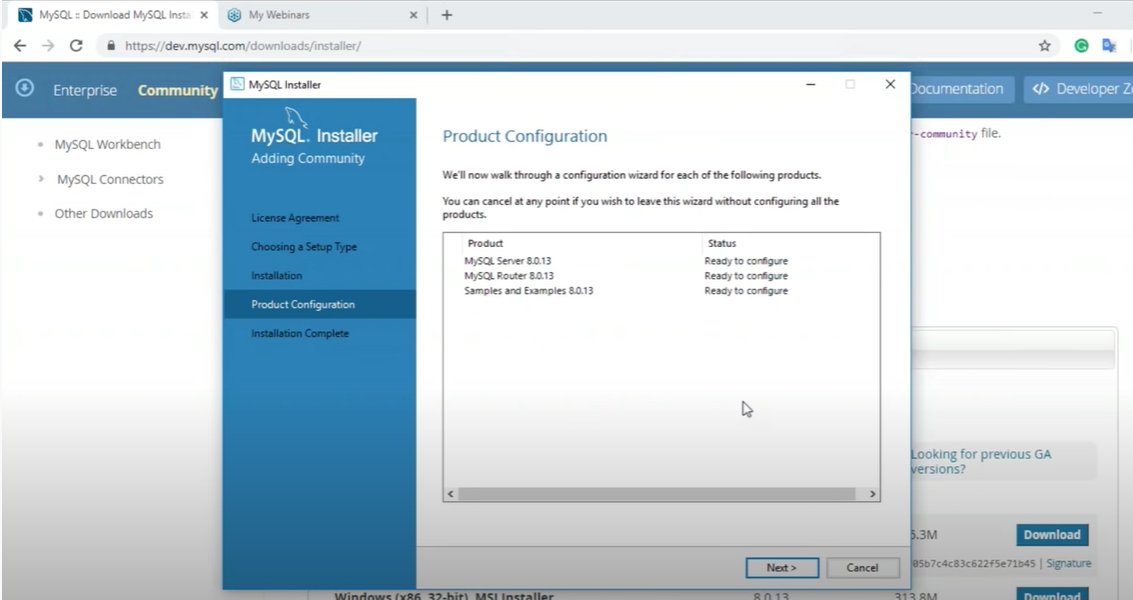
**STEP: - 5**



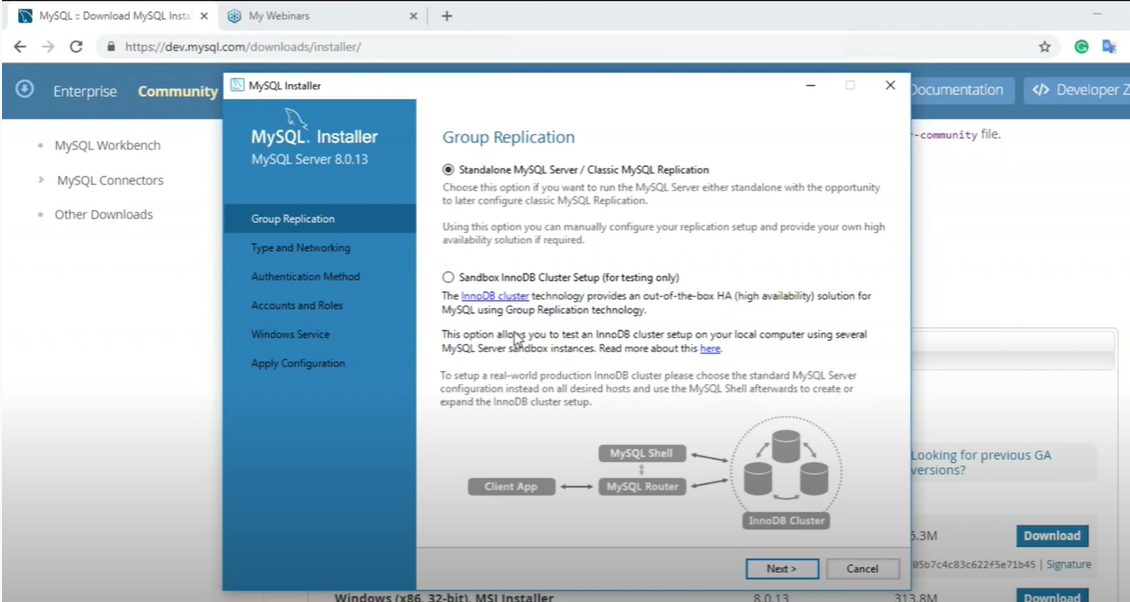
**STEP: - 6**



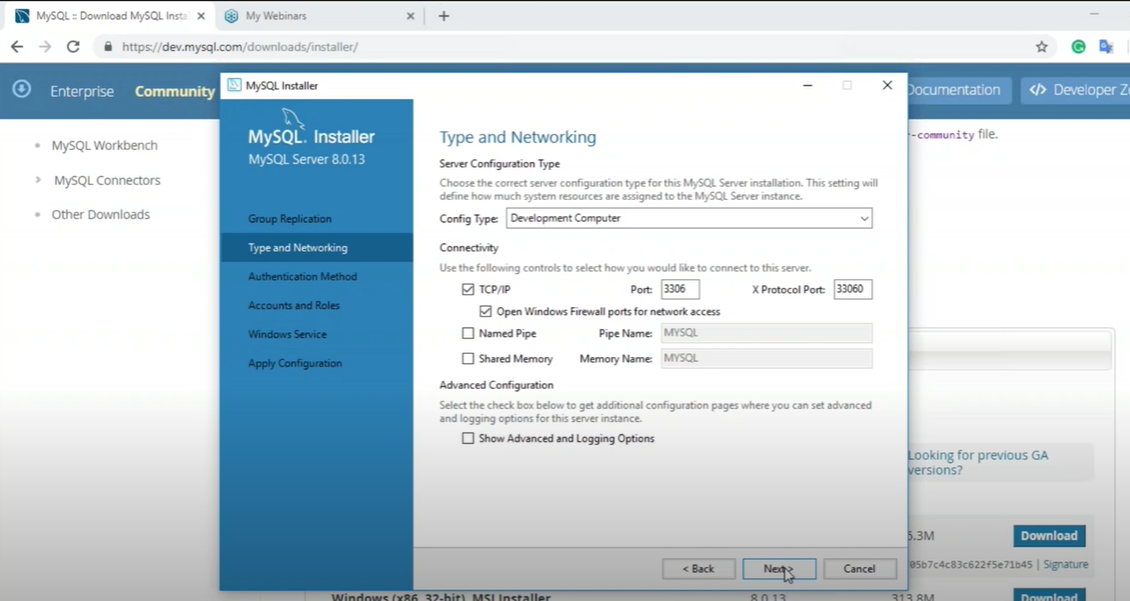
**STEP: - 7**



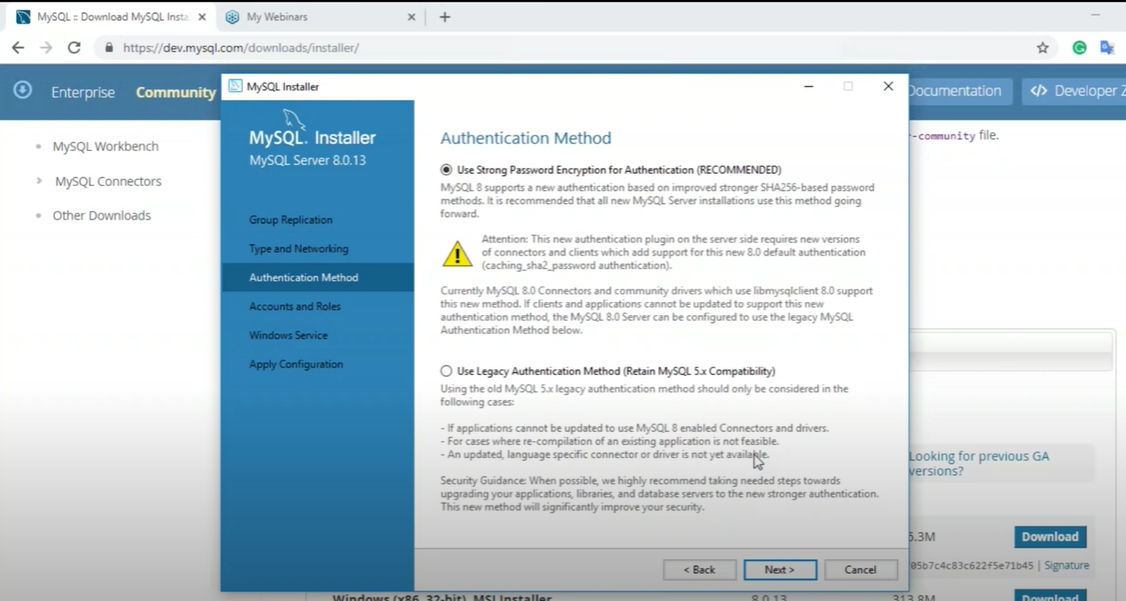
**STEP: - 8**



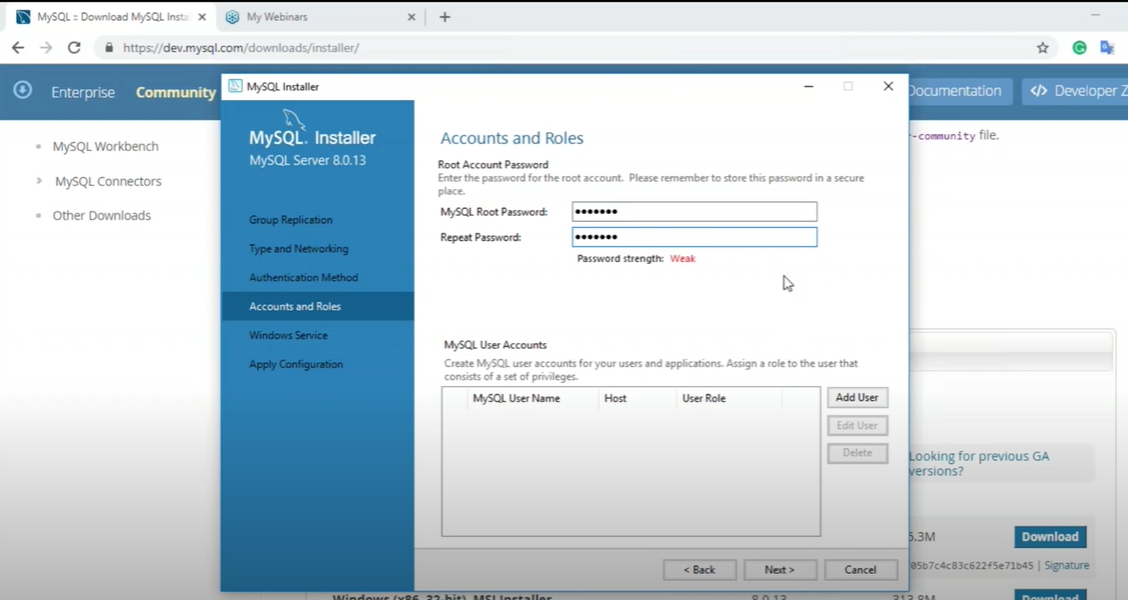
**STEP: - 9**



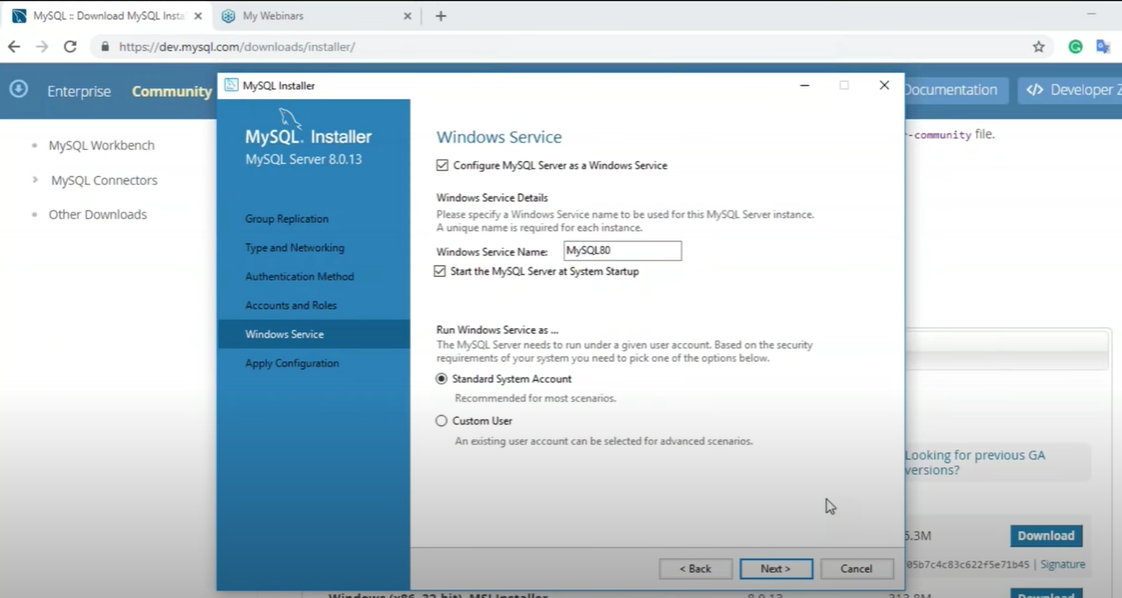
**STEP: - 10**



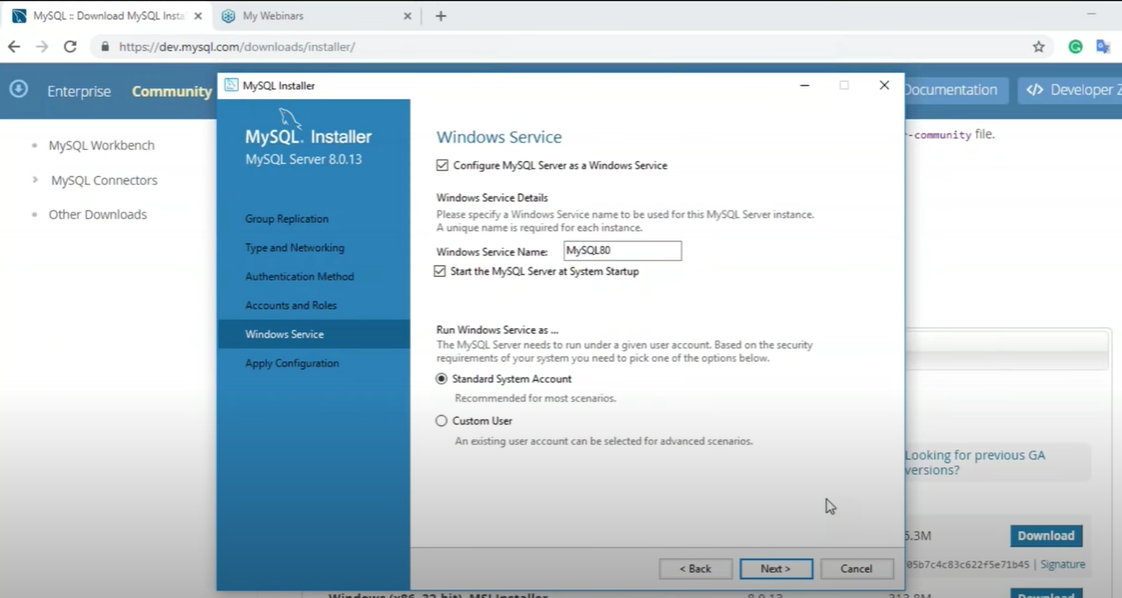
**STEP: - 11**



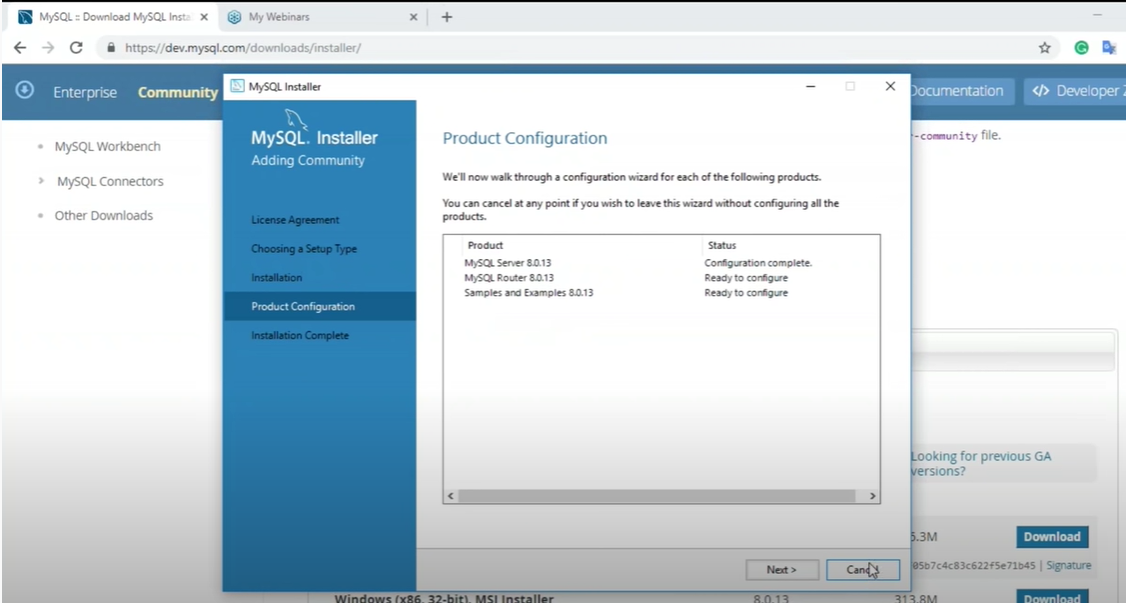
**STEP: - 12**



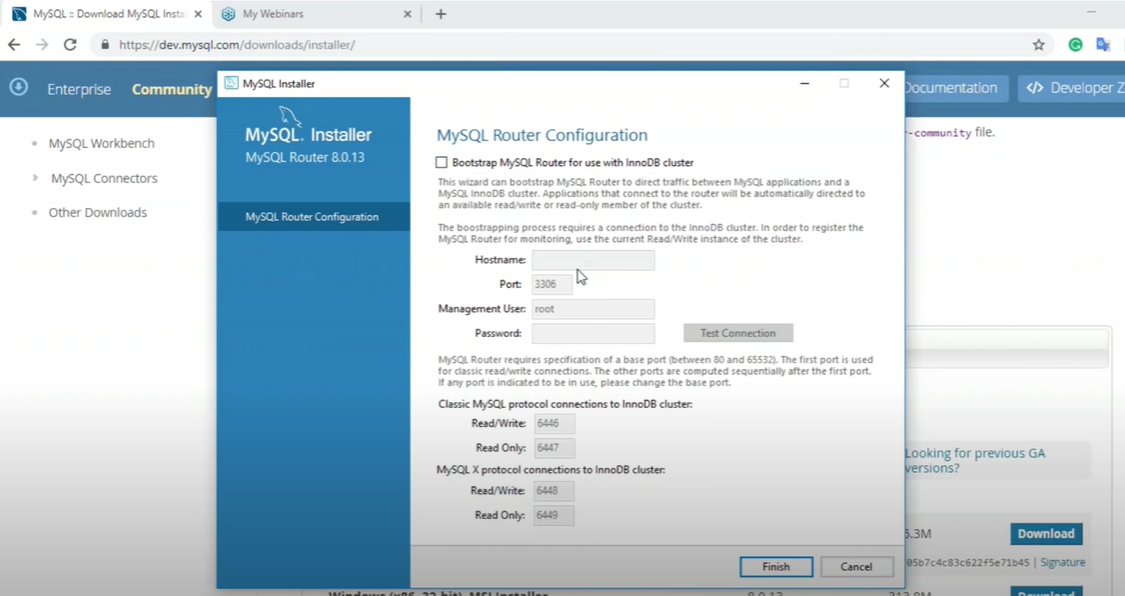
**STEP: - 13**



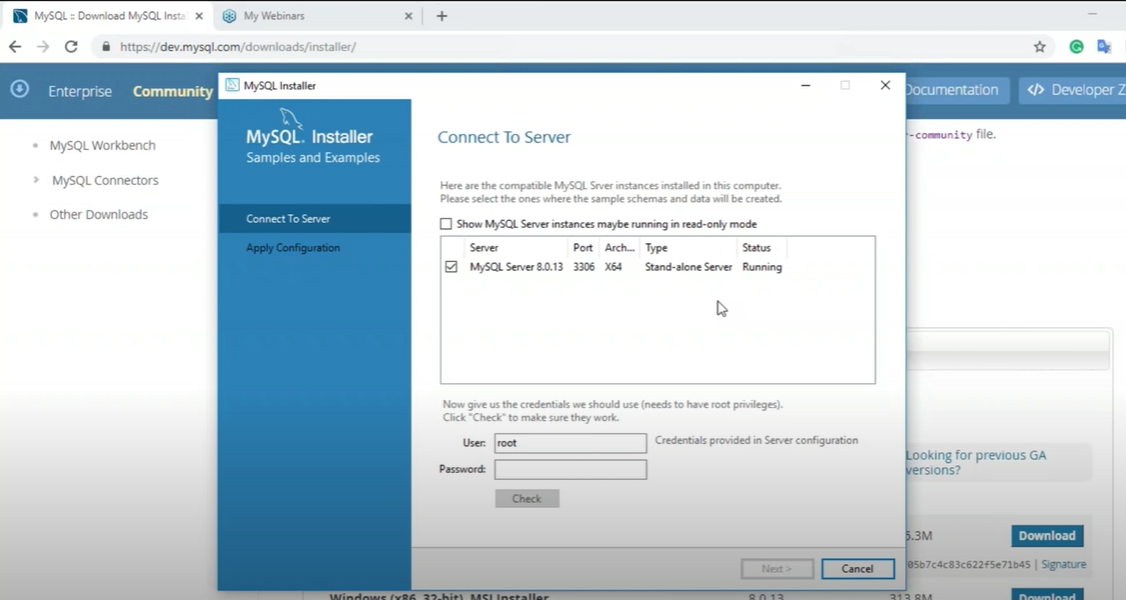
**STEP: - 14**



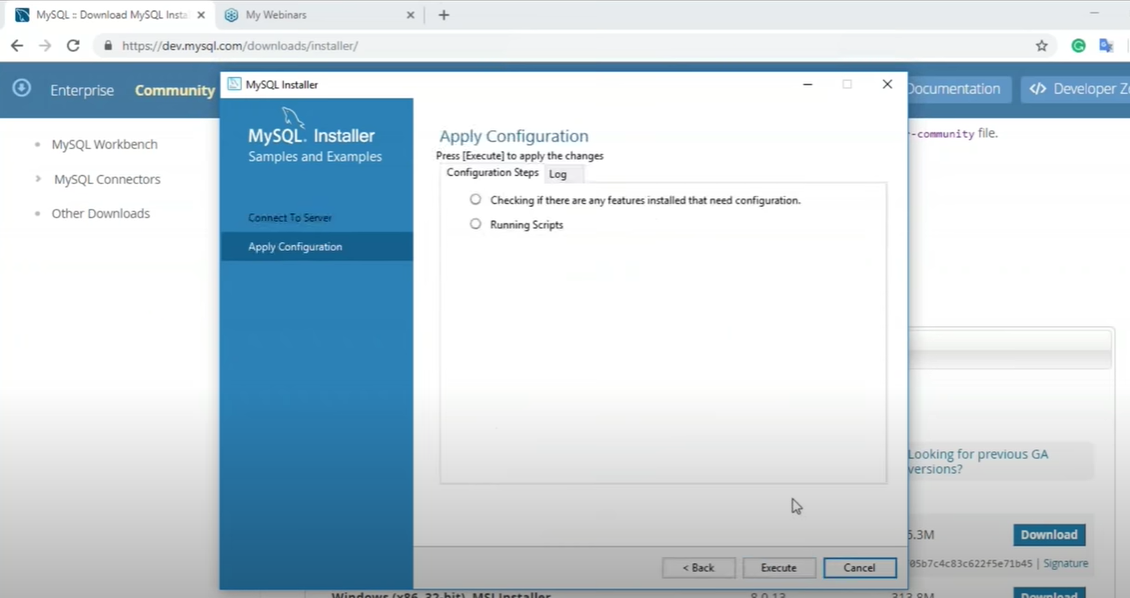
**STEP: - 15**



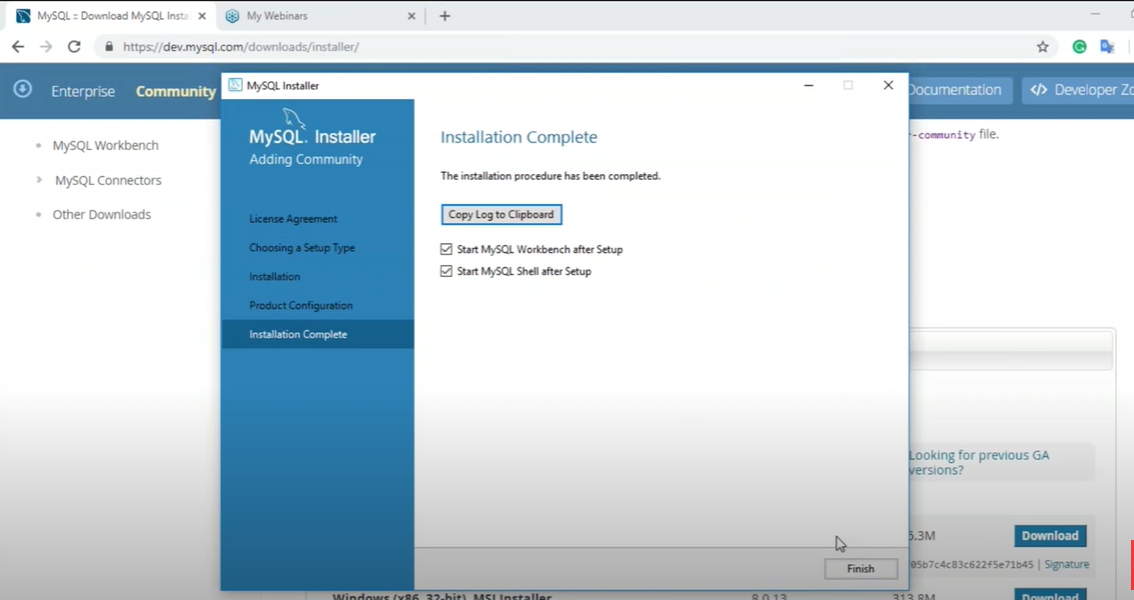
**STEP: - 16**



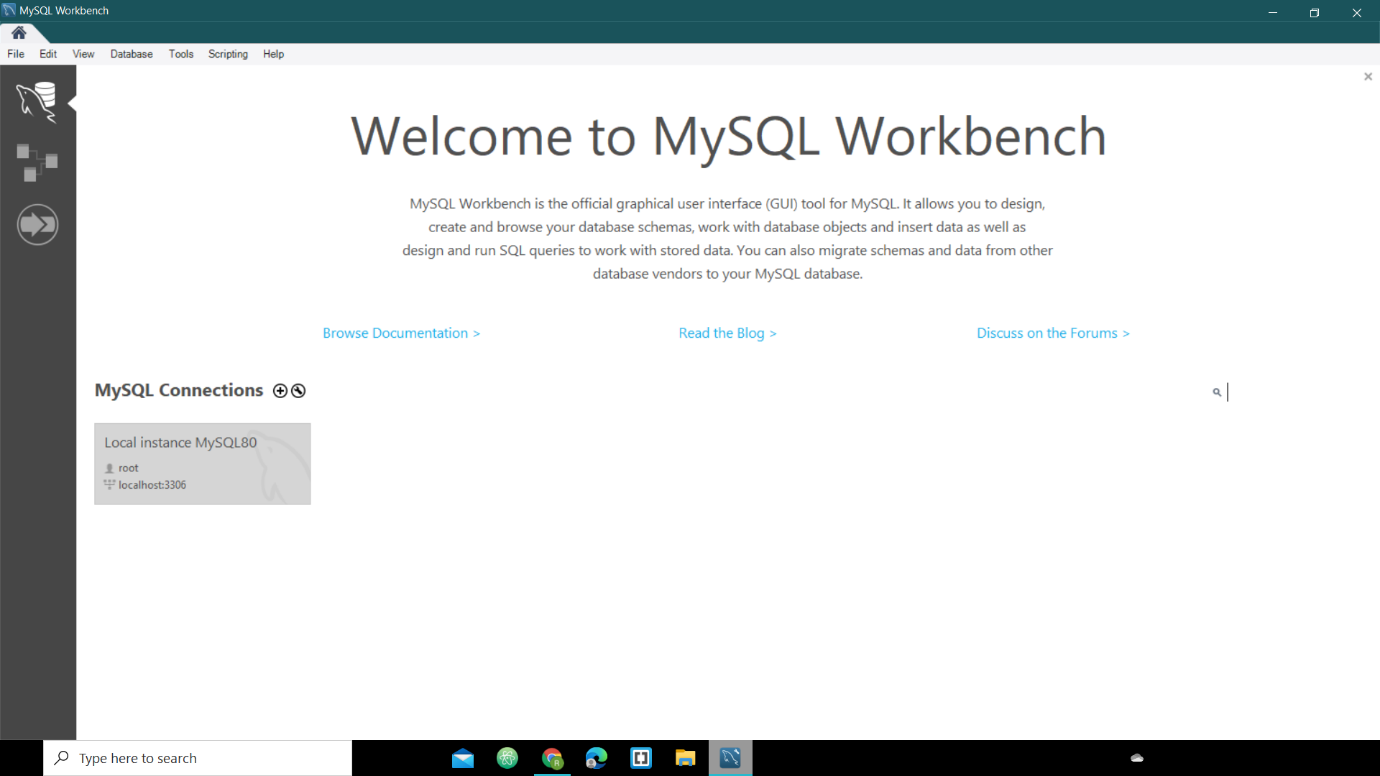
**STEP: - 17**



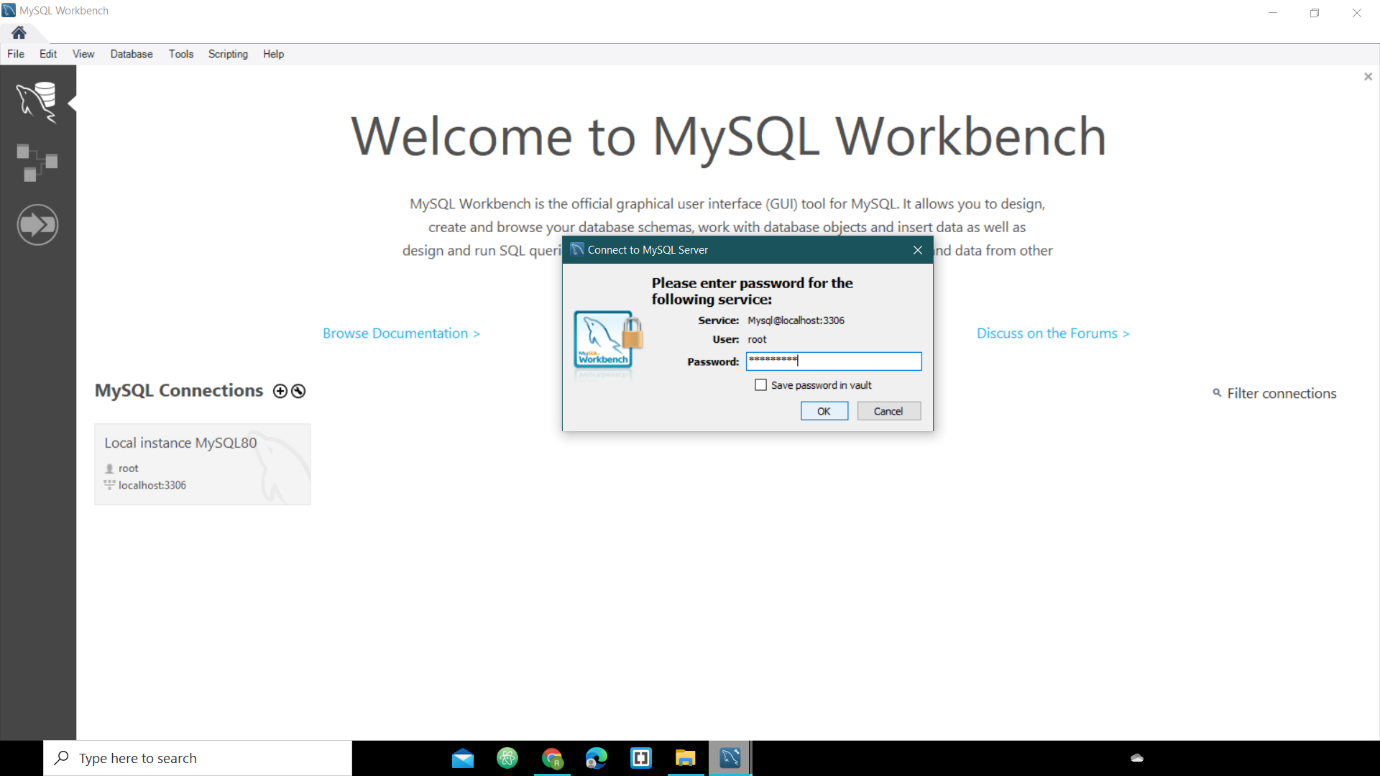
**STEP: - 18**



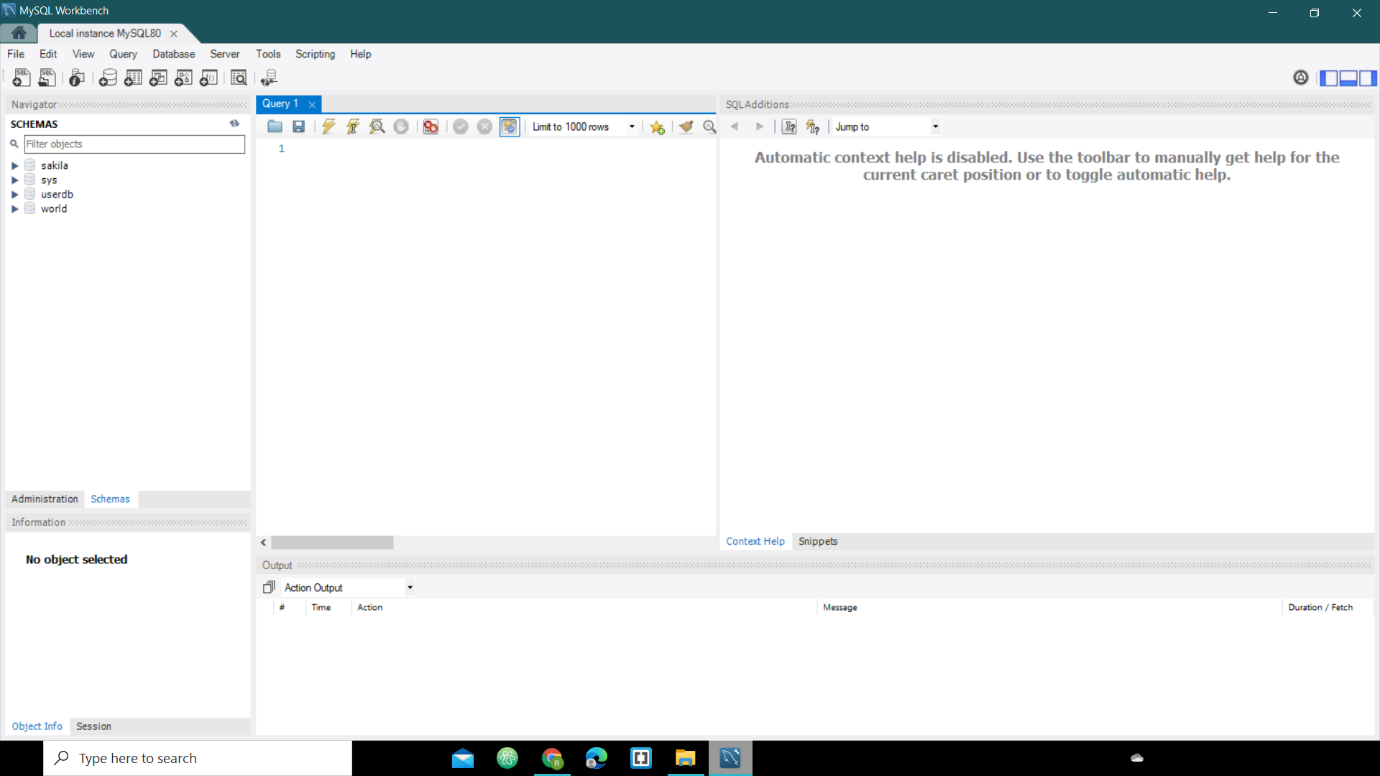
**STEP: - 19**



**STEP: - 20**



**STEP: - 21**



**Assignment – 4**

**19BIT076 (Meet Sheladiya)**

1. **Selection**

Query:

SELECT \* FROM card\_details WHERE amount>1500

Description:

The above query will select rows from card\_details table where amount is greater than 1500

1. **Projection**

Query:

SELECT booking\_id, seat\_number FROM confirmation

Description:

The above query will select columns booking\_id and seat\_number from confirmation table

1. **Cartesian Product**

Query:

SELECT payment\_method.booking\_id, card\_details.amount

FROM

payment\_method, card\_details

Description:

The above query will select columns booking\_id and amount from the tables payment\_method and card\_details respectively and join them.

1. **Union**

Query:

SELECT audience\_id, seat\_number FROM seat\_details

UNION

SELECT audience\_id, ROUND (temperature) FROM audience\_temperature

Description:

The above query will select columns from respective tables and display unique data and display the common data present in both tables only once.

1. **Set Difference**

Query:

SELECT audience\_id, seat\_number FROM seat\_details

MINUS

SELECT audience\_id, ROUND (temperature) FROM audience\_temperature

Description:

The above query will select columns from respective tables and display unique data from columns of seat\_details table which is not present in columns of audience\_temperature table.

1. **Natural Join**

Query:

SELECT \* FROM payment\_method

NATURAL JOIN

card\_details

Description:

The above query will display the common columns from the tables only once.

1. **Composition of any two form (1-6) operators**

Query:

SELECT \* FROM payment\_method

NATURAL JOIN

card\_details

Description:

The above query uses the composition of two operators SELECT and NATURAL JOIN.

1. **Composition of any three form (1-6) operators**

Query:

SELECT \* FROM payment\_method

NATURAL JOIN

card\_details

UNION

confirmation

Description:

The above query uses the composition of three operators SELECT, NATURAL JOIN and UNION. Firstly, the columns of the two tables payment\_method and card\_details will be Natural Joined and then we will Union them with confirmation table.

**19BIT108(Renish Jagani)**

* **Selection:-**

**Query:-**

SELECT \* FROM venue WHERE city\_id =21;

**Description**

The above query will select rows from table venue where city\_id is 21.

* **Projection:-**

**Query:-**

SELECT match\_number, match\_id from fixtures;

**Description**

The above query will select columns match\_number and match\_id from fixtures.

* **Cartesian Product:-**

**Query:-**

SELECT home\_team. home\_team\_ID, away\_team. name, away\_team. away\_team\_id

FROM home\_team, away\_team;

**Description:**  The above query will display all the selected columns from tables home\_team, away\_team

* **Union**

**Query:-**

SELECT match\_number,stadium\_id FROM fixtures

UNION

seat\_number, stadium\_id FROM conformation;

**Description: -** The above query will display only unique data from the selected attributes from respected tables.

* **Set Difference:-**

**Query:-**

SELECT match\_number, stadium\_id FROM fixtures

MINUS

SELECT seat\_number, stadium\_id FROM conformation;

**Description: -** The above query will subtract the common data and display the attributes of fixtures which are not present in conformation.

* **Natural Joint:-**

**Query:-**

SELECT \* FROM

venue NATURAL JOIN fixtures;

**Description:** The columns with the same name of associated tables will appear once only.

* **Composition of any two from (1-6) operators**

**Query:-**

SELECT \* FROM

fixtures NATURAL JOIN conformation

**Description:** The above query uses the composition of two operators SELECT AND NATURAL JOIN.

* **Composition of any three of above from (1-6) operators**

SELECT \* FROM

fixtures NATURAL JOIN conformation

UNION

venue

**Description**:The above query uses the composition of three operators SELECT, NATURAL JOIN and union. Firstly, the attributes of two tables fixtures and conformation will be Natural Joined and all the duplicate values of common attribute will be eliminated and then we will union it with venue table.

**19BIT109(Risht Samariya)**

* **Selection:-**

**Query:**

SELECT \* FROM food name WHERE MRP=Rs 100

**Description-** Selection will select tuples from table food\_and\_beverages where the food name mrp is Rs100

* **Projection:-**

**Query:**

SELECT food\_id, seat\_number from food\_and\_beverages\_order

**Description**- Projection will select the column of food\_id and seat\_number from table food\_and \_beverages\_order

* **Cartesian Product:-**

**Query:**

SELECT jersy. jersy\_id, jersy\_order. booking\_id FROM jersy, jersy\_order

**Description**- It will merge jersy\_id and seat\_number from jersy table to booking\_id from jersy\_order table

* **Union:-**

**Query:**

SELECT seat\_number from food\_and\_beverages

UNION

SELECT seat\_number from jersy

**Description**- Union will merge the seat\_number of table food\_and\_beverages with seat\_number of table jersy which will show only unique data which is not common in both column.

* **Set difference:-**

**Query:**

SELECT booking\_id from food\_and\_beverages\_order

MINUS

SELECT booking\_id from jersy\_order

**Description**- It will subtract the common data from booking\_id of both table food\_and\_beverages\_order and jersy\_order and it will display it.

* **Natural Join:--**

**Query:-**

SELECT\*FROM food\_and\_beverages NATURAL JOIN jersy

**Description**- The columns of seat\_number and MRP in both table will only appear once in the output because it is common in both the table.

* **Composition of any two from (1-6) operators Query:-**

**Query:-**

SELECT booking\_id from food\_and\_beverages

UNION

SELECT booking\_id from jersy

**Description**- there is two operation composition projection and union

* **Composition of any three of above from (1-6) operators Query:-**

**Query:-**

SELECT \* FROM

food\_and\_beverages NATURAL JOIN jersy

UNION

Food\_and\_beverages\_order

**Description**- there is three operation composition projection (select), Natural Join and Union

**19BIT141(Rishabh Patel)**

1. **Select Operation**

Query: - SELECT \* FROM seat\_booking WHERE seat\_number >= 45

Description – In Select operation, we are selecting seat\_number attribute from seat\_booking and having constraint that seat\_number must be greater than greater 45.

1. **Projection**

Query: - SELECT username, password FROM login

Description – In above projection operation we select username and password from login table and output both table.

1. **Cartesian Product**

Query: - SELECT login. email\_id, login. username, seat\_booking. seat\_number

FROM login, seat\_booking

Description – In Cartesian product is selecting email\_id and seat\_number from login and seat\_number from seat\_booking and displaying all three attributes as an output.

1. **Union**

Query: - SELECT email\_id, age FROM login

UNION

SELECT email\_id, seat\_number FROM seat\_booking

Description – In Union Operation it includes email\_id, age from login and email\_id, seat\_number from seat\_booking and it eliminates duplicate tuples.

1. **Set Difference**

Query: - SELECT email\_id, age FROM login

MINUS

SELECT email\_id, seat\_number FROM seat\_booking

Description – The output of set difference (A-B), is removing common values of age and seat\_number and display only unique data from the login table.

1. **Natural Join**

Query: - SELECT \* FROM login NATURAL JOIN seat\_booking

Description – In Natural Join, displaying the attributes only once which are present in both

The tables and uncommon as it is.

1. **Composition of any two from (1-6)operations**

Query: - SELECT login.email\_id, login.username, seat\_booking.audience\_id

FROM login, seat\_booking

UNION

SELECT audience\_id, gate\_number, tier FROM seat\_arrangement

Description: - In this above query, there are composition of two operations **Cartesian product** and **Union** operator firstly selecting name, phone\_number seat\_number attributes from seat\_booking table and section from seat\_location and cross joining two tables and finally talking union with gate\_number which results in displaying all values of respective attributes only once.

1. **Composition of any three from (1-6)operations**

Query: - ­SELECT email\_id, booking\_id, audience\_id FROM seat\_booking

NATURAL JOIN seat\_location

UNION

SELECT email\_id, phone\_number, username FROM login

MINUS

SELECT audience\_id FROM seat\_arrangement

Description:- In the above query, the Composition of three operators are **Natural Join, Union** and **Set Difference** first we select all attributes from seat\_booking and seat\_location tables and perform natural join so we get all unique columns and common columns only once then taking union with email\_id and phone\_number attribute from login input table so we include both attribute in our output then finally set difference with audience\_id attribute from seat\_arrangement so final output is that we will get all the attributes except of audience\_id attribute because we minus it from previous output table.

**19BIT142(Ujash Thakkar)**

1. **Selection Operation**

**QUERY:**

SELECT tuples FORM match\_details WHERE match\_number =15.

**Description:**

The above query will select rows form table match\_details where match\_number is 15.

1. **Projection**

**QUERY:**

SELECT seat\_number and booking\_id FORM audience\_verification.

**Description:**

In above query will select seat\_number and booking\_id from audience\_verification.

1. **Cartesian Product**

**QUERY:**

SELECT booking\_id, match\_number, audience\_id FORM audience\_verification, tickets\_cancelled.

**Description:**

In above query Cartesian product is selecting booking\_id, audience\_id form audience\_verification and match \_number from tickets\_cancelled.

1. **Union**

**QUERY:**

SELECT stadium\_id, audience\_id, match\_number FORM match\_details.

UNION

SELECT stadium\_id, stadium\_number FORM stadium\_details.

**Description:**

In Union Operation in includes stadium\_id, audience\_id, match\_number FORM match\_details and stadium\_id, stadium\_number FORM stadium\_details and it eliminates supplicate tuples.

1. **Set Difference**

**QUERY:**

SELECT stadium\_id, audience\_id, match\_number FORM match\_details.

MINUS

SELECT stadium\_id, stadium\_number FORM stadium\_details.

**Description:**

The above query will subtract the common data and display the unique data.

1. **Natural Join**

**QUERY:**

SELECT \*FROM tickets\_cancelled NATURAL JOIN audience\_verification.

**Description:**

In above query natural join, print the attributes only once which are not present in both tables and uncommon as it is.

1. **Composition of any two from (1-6) operators**

**QUERY:**

SELECT seat\_number

FROM audience\_verification

NATURAL JOIN tickets\_cancelled

**Description:** The above query uses the composition of two operators SELECT AND NATURAL JOIN.

1. **Composition of any three of above from (1-6) operators**

**QUERY:**

SELECT seat\_number

FROM audience\_verification.

NATURAL JOIN tickets\_cancelled

UNION match\_detail

**Description:** The above query uses the composition of 3 operators SELECT, UNION AND NATURAL JOIN.

**19BIT144(Ananya Khandelwal)**

1. **Selection** – SELECT \* FROM tickets\_sold where sold >= 500.

**Description** – Selection will select tuples from table tickets sold where the tickets sold are more than 500.

1. **Projection** – SELECT brand\_name, cost from advertisement.

**Description** – Projection will select the column of brand\_name and cost from table advertisement.

1. **Cartesian product** - SELECT tickets\_sold. stadium\_id, tickets\_sold. sold, tickets\_available. available

FROM tickets\_sold, tickets\_available.

**Description** – Cartesian product will merge stadium\_id, sold and available from tables ticekts\_sold and tickets\_available respectively.

1. **Union** – Select match\_number, sold from tickets sold

UNION

select match\_number, available from tickets available.

**Description** – Union will display only unique data from the selected attributes which is not common in both the tables.

1. **Set Difference** - Select match\_number, sold from tickets sold

MINUS

select match\_number, available from tickets\_available.

**Description** – Set difference will subtract the common data and display the attributes of tickets sold which are not present in tickets available.

1. **Natural Join** – SELECT \* FROM

tickets\_sold

NATURAL JOIN

weather.

**Description** – The columns with same name in both the tables will appear only once in the output.

1. **Composition of Two Operators** – SELECT \* FROM

Weather NATURAL JOIN tickets\_available

**Description** – This query will composite two individual operators i.e., SELECT & NATURAL JOIN.

1. **Composition of Three Operators** – SELECT \* FROM

Weather NATURAL JOIN tickets\_available.

UNION

tickets\_sold.

**Description** – This query will composite three individual operators i.e., SELECT, NATURAL JOIN, UNION.

**Assignment-6**

**19BIT076 (Meet Sheladiya)**

1. **payment\_method**

CREATE TABLE `payment\_method` (

`booking\_id` varchar (255),

`match\_id` varchar (255),

`payment\_mode` varchar (255),

`payment\_id` varchar (255) PRIMARY KEY AUTO\_INCREMENT,

`amount` int DEFAULT NULL

);

1. **card\_details**

CREATE TABLE `card\_details` (

`payment\_id` varchar (255) PRIMARY KEY AUTO\_INCREMENT,

`card\_number` varchar (255),

`amount` int DEFAULT NULL,

`expiry\_date` varchar (255)

);

1. **confirmation**

CREATE TABLE `confirmation` (

`booking\_id` varchar (255),

`stadium\_id` varchar (255),

`match\_id` varchar (255),

`seat\_number` int DEFAULT NULL PRIMARY KEY AUTO\_INCREMENT

);

1. **audience\_temperature**

CREATE TABLE `audience\_temperature` (

`audience\_id` varchar (255) PRIMARY KEY AUTO\_INCREMENT,

`temperature` double DEFAULT NULL

)

**19BIT108(Renish Jagani)**

* CREATE TABLE ‘venue(

‘city\_id’ int,

‘name’ varchar (255),

‘stadium\_id’ varchar (255) PRIMARY KEY AUTO\_INCREMENT,

);

* CREATE TABLE ‘fixtures’ (

‘match\_number’ int PRIMARY KEY AUTO\_INCREMENT,

‘match\_id’ varchar (255),

‘stadium\_id’ varchar (255),

‘home\_team\_id’ varchar (255),

‘away\_team\_id’ varchar (255),

‘city\_id’ int,

);

* CREATE TABLE ‘home\_team\_id’ (

‘home\_team\_id’ varchar (255) PRIMARY KEY AUTO\_INCREMENT,

‘name’ varchar (255),

‘home\_stadium’ varchar (255),

);

* CREATE TABLE ‘away\_team\_id’ (

‘away\_team\_id’ varchar (255) PRIMARY KEY AUTO\_INCREMENT,

‘name’ varchar(255),

)

**19BIT109(Rishit Samariya)**

* CREATE TABLE ‘food\_and\_beverages’(

‘seat\_number’ int PRIMARY KEY AUTO\_INCREMENT,

‘food\_id’ varchar (255),

‘food\_name’ varchar (255),

‘MRP’ int,

);

* CREATE TABLE ‘food\_and\_beverages\_order’(

‘seat\_number’ int PRIMARY KEY AUTO\_INCREMENT,

‘food\_id’ varchar (255),

‘booking\_id’ varchar (255),

);

* CREATE TABLE ‘jersy’(

‘seat\_number’ int PRIMARY KEY AUTO\_INCREMENT

‘jersy\_id’ varchar (255),

‘team\_name’ varchar (255),

‘size’ int,

‘MRP’ int,

);

* CREATE TABLE ‘jersy\_order’(

‘seat\_number’ int PRIMARY KEY AUTO\_INCREMENT

‘jersy\_id’ varchar (255),

‘booking\_id’ varchar (255),

);

**19BIT141(Rishabh Patel)**

* **Login:-**

CREATE TABLE `stadium\_seat\_booking\_database`. `login` (

`email\_id` VARCHAR (45) NOT NULL,

`username` VARCHAR (245) NOT NULL,

`password` VARCHAR (245) NOT NULL,

`f\_name` VARCHAR (245) NOT NULL,

`l\_name` VARCHAR (245) NOT NULL,

`gender` VARCHAR (245) NOT NULL,

`age` INT NOT NULL,

`phone\_number` INT (10) NOT NULL,

`security\_question` VARCHAR (245) NOT NULL,

`answer` VARCHAR (245) NOT NULL,

PRIMARY KEY (`email\_id`, `username`, `password`));

* **Seat\_booking:-**

CREATE TABLE `stadium\_seat\_booking\_database`.`seat\_booking` (

`booking\_id` VARCHAR (245) NOT NULL,

`email\_id` VARCHAR (245) NOT NULL,

`name` VARCHAR (245) NOT NULL,

`phone\_number` INT (10) NOT NULL,

`address` VARCHAR (245) NOT NULL,

`postal\_code` INT (10) NOT NULL,

`seat\_number` INT (10) NOT NULL,

`audience\_id` VARCHAR (245) NOT NULL,

`match\_id` VARCHAR (245) NOT NULL,

PRIMARY KEY (`booking\_id`, `phone\_number`, `seat\_number`, `audience\_id`, `match\_id`));

* **Seat arrangement:-**

CREATE TABLE `stadium\_seat\_booking\_database`.`seat\_arrangement` (

`audience\_id` VARCHAR (245) NOT NULL,

`gate\_number` INT (10) NOT NULL,

`tier` INT (10) NOT NULL,

`stadium\_id` VARCHAR (245) NOT NULL,

PRIMARY KEY (`audience\_id`));

* **Seat Location:-**

CREATE TABLE `stadium\_seat\_booking\_database`.`seat\_location` (

`seat\_number` INT (10) NOT NULL,

`section` VARCHAR (245) NOT NULL,

PRIMARY KEY (`seat\_number`));

* **Seat Details:-**

CREATE TABLE `stadium\_seat\_booking\_database`.`seat\_details` (

`booking\_id` VARCHAR (245) NOT NULL,

`audience\_id` VARCHAR (245) NOT NULL,

`match\_id` VARCHAR (245) NOT NULL,

`seat\_number` INT (10) NOT NULL,

`seat\_category` VARCHAR (245) NOT NULL,

`price` DOUBLE (10) NOT NULL,

PRIMARY KEY (`booking\_id`, `audience\_id`, `seat\_number`));

**19BIT142(Ujash Thakkar)**

1. audience\_verification

CREATE TABLE’ audience\_verification` (

‘booking\_id’ varchar (255),

‘audience\_id’ varchar (255), PRIMARY KEY AUTO\_INCREMENT

‘phone\_number’ int (10),

‘seat\_number’ int (10), PRIMARY KEY AUTO\_INCREMENT

‘aadhar\_number’ int (12),

PRIMARY KEY (`booking\_id’, ‘phone\_number’, ‘aadhar\_number’));

1. ticket\_cancelled

CREATE TABLE` ticket\_cancelled ‘(

‘stadium\_id` varchar (255),

‘match\_id` varchar (255),

‘match\_number’ int DEFAULT NULL, PRIMARY KEY AUTO\_INCREMENT

‘audience\_id’ varchar (255), PRIMARY KEY AUTO\_INCREMENT

1. stadium\_details

CREATE TABLE ‘stadium\_details’ (

‘stadium\_id’ varchar (255), PRIMARY KEY AUTO\_INCREMENT

‘stadium\_name’ varchar (255),

‘address’ varchar (255),

‘capacity’ int,

1. match\_details

CREATE TABLE ‘match\_details’ (

‘stadium\_id’ varchar (255),

‘audience\_id’ varchar (255), PRIMARY KEY AUTO\_INCREMENT

‘match\_number’ INT DEFAULT NULL, PRIMARY KEY AUTO\_INCREMENT

‘match\_id` varchar (255),

‘date\_of\_match’ varchar (255),

‘match\_timing’ varchar (255),

**19BIT144(Ananya Khandelwal)**

1. CREATE TABLE `tickets\_sold` (

`Stadium\_id` varchar (255),

`Match\_id` varchar (255),

`Match\_number` int DEFAULT NULL, PRIMARY KEY AUTO\_INCREMENT

`Sold` int DEFAULT NULL

);

1. CREATE TABLE `advertisement` (

`Audience\_id` varchar (255), PRIMARY KEY AUTO\_INCREMENT

`Brand\_name` varchar (255),

`Cost (In Rs) ` double DEFAULT NULL

);

1. TABLE `weather` (

`Match\_Number` int DEFAULT NULL, PRIMARY KEY AUTO\_INCREMENT

`Temperature (Celsius) ` double DEFAULT NULL,

`Humidity (%) ` int DEFAULT NULL

;

1. CREATE TABLE `tickets\_available` (

`Stadium\_id` varchar (255),

`Match\_id` varchar (255),

`Match\_number` int DEFAULT NULL, PRIMARY KEY AUTO\_INCREMENT

`Available` int DEFAULT NULL

)

**Assignment – 7**

**19BIT076 (Meet Sheladiya)**

1. **Aggregate functions**

**Max**

Query:

SELECT MAX (temperature) FROM audience\_temperature

1. **Numeric**

Query:

SELECT ROUND (temperature) FROM audience\_temperature

1. **Date**

Query:

SELECT \* FROM card\_details WHERE expiry\_date=’2023-05-27’

1. **String function**

Query:

SELECT LENGTH (payment\_mode) FROM payment\_method

**19BIT108 (Renish Jagani)**

* **Aggregate function:-**

SELECT COUNT (city\_id) FROM venue;

* **Built-in Numeric Function:-**

SELECT MAX (humidity) FROM weather;

* **String Functions:-**

SELECT name, ASCII (name)

FROM home\_team;

* **Date Function:-**

SELECT YEAR ('2017/08/25');

**19BIT109 (Rishit Samariya)**

* **Aggregate function-**

**Query:-**

SELECT MAX (MRP) FROM food\_and\_beverages;

* **Built-in Numeric Function-**

**Query:-**

SELECT ABS (Size) FROM Jersy;

* **String Function-**

**Query:-**

Select food\_name, UPPER (food\_name) from food\_and\_beverages

* **Date Function-**

**Query:-**

SELECT \* FROM card\_details WHERE expiry\_date=’2025-01-02’

**19BIT141 (Rishabh Patel)**

1. **Aggregate function:-**

SELECT SUM (age) FROM login;

1. **Built-in Numeric Function:-**

SELECT ROUND (price) FROM seat\_details;

1. **String Function:-**

SELECT CONCAT (f\_name, l\_name) FROM login;

1. **Date Function:-**

SELECT \* FROM card\_details WHERE expiry\_date = ‘2023-24-11’

**19BIT142(Ujash Thakkar)**

1. **Aggregate function**

**Query:**

SELECT COUNT (audience\_id) FROM ticket\_cancelled

1. **Numeric function**

**Query:**

SELECT ABS (seat\_number) FROM audience\_verification

1. **Date function**

**Query:**

SELECT date\_of\_match FROM match\_details WHERE

date\_of\_match=’2021-04-03’;

1. **String function**

**Query:**

SELECT stadium\_name, LOWER (stadium\_name) FROM stadium\_details

**19BIT144 (Ananya Khandelwal)**

1. **Aggregate Function:-**

SELECT avg (cost) FROM advertisement.

1. **Built in Numeric function:-**

SELECT ROUND (temperature) FROM weather where match number = 2.

1. **Date:-**

SELECT YEAR (‘2019/0510’)

1. **String:-**

SELECT length (brand\_name) FROM advertisement where audience id = \_\_\_\_\_\_\_\_\_.

**Assignment-8**

**19BIT076 (Meet Sheladiya)**

1. **Set Operations**
2. **Union**

Query:

SELECT audience\_id, seat\_number FROM seat\_details

UNION

SELECT audience\_id, ROUND (temperature) FROM audience\_temperature

1. **Union All**

Query:

SELECT audience\_id, seat\_number FROM seat\_details

UNION ALL

SELECT audience\_id, ROUND (temperature) FROM audience\_temperature

1. **Intersect**

Query:

SELECT audience\_id, seat\_number FROM seat\_details

INTERSECT

SELECT audience\_id, ROUND (temperature) FROM audience\_temperature

1. **Minus**

Query:

SELECT audience\_id, seat\_number FROM seat\_details

MINUS

SELECT audience\_id, ROUND (temperature) FROM audience\_temperature

1. **Sub-queries**

Query:

SELECT \* FROM audience\_temperature WHERE temperature>37

1. **Correlated sub-queries**

Query:

SELECT booking\_id, match\_id, amount

FROM payment\_method

WHERE payment\_id = (SELECT payment\_id FROM card\_details WHERE amount>4900)

**19BIT108 (Renish Jagani)**

* **Set operations:-**

SELECT match\_number, stadium\_id FROM fixtures

UNION

seat\_number, stadium\_id FROM conformation;

* **Sub queries:-**

SELECT \*

FROM fixtures

WHERE match\_id IN (SELECT match\_id

FROM fixtures

WHERE city\_id > 20);

* **Co-related Sub queries:-**

SELECT match\_id from fixtures where stadium\_id= (SELECT STADIUM\_ID FROM VENUE WHERE city\_id>6)

**19BIT109 (Rishit Samariya)**

1. **Set Operations:-**

* **Union-**

**Query:-**

SELECT seat\_number from food\_and\_beverages

UNION

SELECT seat\_number from jersy

* **Union All-**

**Query:-**

SELECT seat\_number from food\_and\_beverages

UNION All

SELECT seat\_number from jersy

* **Intersect-**

**Query:-**

SELECT seat\_number from food\_and\_beverages

INTERSECT

SELECT seat\_number from jersy

* **Minus-**

**Query:-**

SELECT booking\_id from food\_and\_beverages\_order

MINUS

SELECT booking\_id from jersy\_order

1. **Sub-queries:-**

**Query:-**

SELECT \* FROM food name WHERE MRP=Rs 100

1. **Correlated sub-queries:-**

**Query:-**

SELECT food\_name, seat\_number,

FROM food\_and\_beverages

WHERE food\_id = (SELECT food\_id FROM food\_and\_beverages\_order WHERE seat\_number = 10)

**19BIT141 (Rishabh Patel)**

1. **Set Operations:-**

* **INTERSECT:-**

**Query:-**

SELECT email\_id, username, password FROM login

INTERSECT

SELECT email\_id, name, phone\_number FROM seat\_booking;

* **UNION:-**

**Query:-**

SELECT email\_id, age FROM login

UNION

SELECT email\_id, seat\_number FROM seat\_booking;

* **UNION ALL:-**

**Query:-**

SELECT \* FROM seat\_booking

UNION ALL

SELECT \* FROM seat\_arrangement;

* **MINUS:-**

**Query:-**

SELECT audience\_id, name, seat\_number FROM seat\_booking

MINUS

SELECT audience\_id, gate\_number, tier;

1. **Sub-Queries:-**

* **UPDATE:-**

**Query:-**

UPDATE seat\_details

SET price = price \* 0.5

WHERE 4 >= (SELECT COUNT (match\_id) FROM seat\_booking

WHERE seat\_booking.seat\_number = seat\_details.seat\_number)

1. **Correlated Sub-Queries:-**

**Query:-**

SELECT name, email\_id, seat\_number

FROM seat\_booking

WHERE booking\_id = (SELECT booking\_id FROM seat\_details

WHERE audience\_id = ‘A-011’);

**19BIT142(Ujash Thakkar)**

1. **Set Operations**

**Union**

**QUERY:**

SELECT stadium\_id, audience\_id, match\_number FORM match\_details.

UNION

SELECT stadium\_id, stadium\_number FORM stadium\_details.

1. **Sub-queries**

**QUERY:** SELECT \* FROM audience\_temperature WHERE temperature>37

1. **Correlated sub-queries**

**QUERY:**

SELECT stadium\_id, stadium\_name, FROM stadium\_details

WHERE match\_id = (SELECT match\_id FROM match\_details

WHERE match\_number>23)

**19BIT144 (Ananya Khandelwal)**

1. **Set Operations:-**

SELECT \* FROM weather

INTERSECT

SELECT \* FROM tickets\_available.

1. **Sub Queries:-**

SELECT \* FROM advertisement WHERE audience\_id IN (SELECT audience\_id FROM advertisement WHERE cost > 100000)

1. **Correlated Sub Queries:-**

SELECT stadium\_id,match\_id FROM tickets\_available

WHERE match\_number=(SELECT match\_number FROM tickets\_sold WHERE sold>4900)

**Assignment-9**

**19BIT076 (Meet Sheladiya)**

1. **Group By;-**

Query:-

SELECT payment\_mode, SUM (amount) FROM payment\_method

GROUP BY payment\_mode

1. **Having;-**

Query:-

SELECT stadium\_id, COUNT (booking\_id) FROM confirmation

GROUP BY stadium\_id

HAVING COUNT (booking\_id)>1

1. **Order By;-**

Query:-

SELECT \* FROM card\_details

ORDER BY amount

**19BIT108 (Renish Jagani)**

* **Group By:-**

SELECT COUNT (stadium\_id), name  
FROM venue  
GROUP BY name;

* **Having:-**

SELECT COUNT (stadium\_id), name  
FROM venue  
GROUP BY name

HAVING COUNT (stadium\_id)> 1

(RETURNS ONLY CITY THAT HAS MORE THAN ONE STADIUM)

* **Order By:-**

SELECT \* FROM fixtures ORDER BY city\_id;

**19BIT109 (Rishit Samariya)**

* **Group By-**

**Query:-**

SELECT food\_name, AVG (MRP) FROM food\_and\_beverages

GROUP BY food\_name;

* **ORDER BY-**

**Query:-**

SELECT \* FROM food\_and\_beverages ORDER BY MRP DESC;

* **HAVING-**

**Query:-**

SELECT COUNT (MRP), food\_name FROM food\_and\_beverages

GROUP BY food\_name

HAVING COUNT (MRP)>=50;

**19BIT141 (Rishabh Patel)**

1. **Group By:-**

**Query:-**

SELECT security\_question, COUNT (email\_id) FROM login

GROUP BY security\_question;

1. **Having:-**

**Query:-**

SELECT COUNT (email\_id), age FROM login

GROUP BY age

HAVING COUNT (email\_id) >= 8;

1. **ORDER BY:-**

**Query:-**

SELECT COUNT (name), address FROM seat\_booking

GROUP BY address

ORDER BY COUNT (name) DESC;

**19BIT142(Ujash Thakkar)**

1. **Group By**

Query:

SELECT stadium\_name, stadium\_id

FROM stadium\_details

GROUP BY stadium\_name;

1. **Having**

Query:

SELECT stadium\_name, stadium\_id

FROM stadium\_details

GROUP BY stadium\_name

HAVING COUNT (\*) < 8;

1. **Order By**

Query:

SELECT \*

FROM match\_details

ORDER BY date\_of\_match;

**19BIT144 (Ananya Khandelwal)**

1. **Group by:-**

SELECT humidity, count (match\_number) FROM weather

GROUP BY humidity.

1. **Having:-**

SELECT stadium\_id, match\_id, match\_number FROM tickets\_available

GROUP BY available

HAVING available>500.

1. **Order by:-**

SELECT \* FROM weather

ORDER by temperature

**Assignment – 10**

**19BIT076 (Meet Sheladiya)**

1. **Join**
2. **Inner Join**

Query:-

SELECT card\_details.amount, payment\_method.booking\_id, payment\_method.match\_id FROM payment\_method

INNER JOIN card\_details

ON payment\_method. payment\_id=card\_details.payment\_id

1. **Left Join**

Query:-

SELECT card\_details.amount, payment\_method.booking\_id, payment\_method.match\_id FROM payment\_method

LEFT JOIN card\_details

ON payment\_method.payment\_id=card\_details.payment\_id

1. **Right Join**

Query:-

SELECT card\_details.amount, payment\_method.booking\_id, payment\_method.match\_id FROM payment\_method

RIGHT JOIN card\_details

ON payment\_method.payment\_id=card\_details.payment\_id

1. **Full Join**

Query:-

SELECT card\_details.amount, payment\_method.booking\_id, payment\_method.match\_id FROM payment\_method

FULL JOIN card\_details

ON payment\_method.payment\_id=card\_details.payment\_id

1. **Exists**

Query:-

SELECT booking\_id, match\_id

FROM payment\_method

WHERE EXISTS (SELECT \* FROM card\_details WHERE payment\_method.payment\_id=card\_details.payment\_id AND amount>2500)

1. **Any**

Query:-

SELECT booking\_id

FROM payment\_method

WHERE payment\_id = ANY (SELECT payment\_id FROM card\_details WHERE amount<2000)

1. **All**

Query:-

SELECT booking\_id

FROM payment\_method

WHERE payment\_id = ALL (SELECT payment\_id FROM card\_details WHERE amount>1000)

**19BIT108 (Renish Jagani)**

* **Join operations inner join:-**

SELECT fixtures.match\_number, fixtures.match\_id,home\_team.name

from fixtures

INNER JOIN home\_team

ON home\_team.home\_team\_id = fixtures.home\_team\_id

* **Left join:-**

SELECT fixtures. match number, fixtures.match\_id,home\_team.name

from fixtures

LEFT JOIN home\_team

ON home\_team.home\_team\_id = fixtures.home\_team\_id

* **Right join:-**

SELECT fixtures.match\_number, fixtures.match\_id,home\_team.name

from fixtures

RIGHT JOIN home\_team

ON home\_team.home\_team\_id = fixtures.home\_team\_id

* **Full join:-**

SELECT fixtures.match\_number, fixtures.match\_id,home\_team.name

from fixtures

FULL JOIN home\_team

ON home\_team.home\_team\_id = fixtures.home\_team\_id

* **Exist:-**

SELECT match\_id, match\_number from fixtures

WHERE EXISITS (SELECT \* FROM away\_team WHERE away\_team.away\_team\_id = fixtures.home\_team\_id)

* **All:-**

SELECT match\_id from fixtures where stadium\_id= All (SELECT STADIUM\_ID FROM VENUE WHERE CITY\_ID>6)

* **Any:-**

SELECT match\_id from fixtures where stadium\_id= Any (SELECT STADIUM\_ID FROM VENUE WHERE CITY\_ID>6)

**19BIT109 (Rishit Samariya)**

* **INNER JOIN-**

**Query:-**

SELECT Food\_and\_beverages\_order.booking\_id, Food\_and\_beverages.MRP, Food\_and\_beverages\_order.booking\_id

FROM food\_and\_bevearges

INNER JOIN Food\_and\_beverages\_order.

ON food\_and\_beverages.seat\_number = food\_and\_beverages\_order.seat\_number;

* **Left Join-**

**Query:-**

SELECT jersy\_order.booking\_id, jersy.team\_name

FROM jersy\_order

LEFT JOIN jersy

ON jersy.seat\_number =jersy\_order.seat\_number;

* **Right Join-**

**Query:-**

SELECT Food\_and\_beverages\_order.booking\_id, Food\_and\_beverages.booking\_id

FROM food\_and\_bevearges

RIGHT JOIN Food\_and\_beverages\_order.

ON food\_and\_beverages.seat\_number = food\_and\_beverages\_order.seat\_number;

* **FULL JOIN-**

**Query:-**

SELECT Food\_and\_beverages\_order.booking\_id, Food\_and\_beverages.booking\_id

FROM food\_and\_bevearges

FULL JOIN Food\_and\_beverages\_order.

ON food\_and\_beverages.seat\_number = food\_and\_beverages\_order.seat\_number;

* **EXISTS-**

**Query:-**

SELECT food\_id from food\_and\_beverages\_order

WHERE EXISTS (SELECT \* FROM jersy\_order

WHERE food\_and\_beverages\_order.seat\_number = jersy\_order.seat\_number)

* **All-**

**Query:-**

SELECT food\_name

FROM food\_and\_beverages

WHERE food\_id = ALL (SELECT food\_id FROM food\_and\_beverages\_order WHERE seat\_number = 10)

* **ANY-**

**Query:-**

SELECT food\_name

FROM food\_and\_beverages

WHERE food\_id = ANY (SELECT food\_id FROM food\_and\_beverages\_order WHERE seat\_number = 10)

**19BIT141 (Rishabh Patel)**

* **Join Operations:-**

1. **INNER JOIN:-**

**Query:-**

SELECT seat\_arrangement. gate\_number, seat\_details. match\_id, seat\_details. seat\_number FROM seat\_details

INNER JOIN seat\_arrangement

ON seat\_details. audience\_id = seat\_arrangement. audience\_id;

1. **LEFT JOIN:-**

**Query:-**

SELECT seat\_arrangement. gate\_number, seat\_details. match\_id, seat\_details. seat\_number FROM seat\_details

LEFT JOIN seat\_arrangement

ON seat\_details. audience\_id = seat\_arrangement. audience\_id;

1. **RIGHT JOIN:-**

**Query:-**

SELECT seat\_arrangement. gate\_number, seat\_details. match\_id, seat\_details. seat\_number FROM seat\_details

RIGHT JOIN seat\_arrangement

ON seat\_details. audience\_id = seat\_arrangement. audience\_id;

1. **FULL JOIN:-**

**Query:-**

SELECT seat\_arrangement. gate\_number, seat\_details. match\_id, seat\_details. seat\_number FROM seat\_details

FULL JOIN seat\_arrangement

ON seat\_details. audience\_id = seat\_arrangement. audience\_id;

* **Exists:-**

**Query:-**

SELECT f\_name, l\_name FROM login

WHERE EXISTS (SELECT \* FROM seat\_booking

WHERE login. email\_id = seat\_booking. Email\_id AND age>=25)

* **Any:-**

**Query:-**

SELECT name

FROM seat\_booking

WHERE audience\_id = ANY (SELECT audience\_id FROM seat\_arrangement

WHERE gate\_number = 2 OR gate\_number = 4)

* **ALL:-**

**Query:-**

SELECT name

FROM seat\_booking

WHERE audience\_id = ALL (SELECT audience\_id FROM seat\_arrangement

WHERE gate\_number = 2 OR gate\_number = 4);

**19BIT142(Ujash Thakkar)**

1. **Inner Join**

**Query:**

**SELECT** audience\_verification. phone number, match\_details. match\_number, match\_details. match\_timing FROM match\_details

**INNER JOIN** audience\_verification

**ON** match\_details. audience\_id = audience\_verification. audience\_id;

1. **Left Join**

**Query:**

**SELECT** audience\_verification. phone number, match\_details. match\_number, match\_details. match\_timing FROM match\_details

**LEFT JOIN** audience\_verification

**ON** match\_details. audience\_id = audience\_verification. audience\_id;

1. **Right join**

**Query:**

**SELECT** audience\_verification. phone number, match\_details. match\_number, match\_details. match\_timing FROM match\_details

**RIGHT JOIN** audience\_verification

**ON** match\_details. audience\_id = audience\_verification. audience\_id;

1. **Full join**

**Query:**

**SELECT** audience\_verification. phone number, match\_details. match\_number, match\_details. match\_timing FROM match\_details

**FULL JOIN** audience\_verification

**ON** match\_details. audience\_id = audience\_verification. audience\_id;

1. **Exists**

**Query:**

**SELECT** booking\_id, seat\_number

FROM audience\_verification

WHERE **EXISTS** (SELECT \* FROM match\_details WHERE

audience\_verification.audience\_id=match\_details.audience\_id)

1. **Any**

**Query:**

**SELECT** booking\_id

FROM audience\_verification

WHERE audience\_id = (SELECT audience\_id FROM ticket\_cancelled WHERE seat\_number is=564)

1. **All**

**Query:**

**SELECT** booking\_id

FROM audience\_verification

WHERE audience\_id = (SELECT audience\_id FROM ticket\_cancelled WHERE seat\_number is=465)

**19BIT144 (Ananya Khandelwal)**

1. **INNER JOIN**

Query -

SELECT weather.temperature, tickets\_available.stadium\_id, tickets\_available.match\_id FROM tickets\_available

INNER JOIN weather

ON tickets\_available.match\_number = weather.match\_number

1. **LEFT JOIN**

Query

SELECT weather.temperature, tickets\_available.stadium\_id, tickets\_available.match\_id FROM tickets\_available

LEFT JOIN weather

ON tickets\_available.match\_number = weather.match\_number

1. **RIGHT JOIN**

Query

SELECT weather.temperature, tickets\_available.stadium\_id, tickets\_available.match\_id FROM tickets\_available

RIGHT JOIN weather

ON tickets\_available.match\_number = weather.match\_number

1. **FULL JOIN**

Query

SELECT weather.temperature, tickets\_available.stadium\_id, tickets\_available.match\_id FROM tickets\_available

FULL JOIN weather

ON tickets\_available.match\_number = weather.match\_number

1. **EXIST**

SELECT temperature, humidity FROM weather

WHERE EXISTS (SELECT \* FROM tickets\_available WHERE weather.match\_number=tickets\_available.match\_number

1. **ALL**

SELECT brand\_name FROM advertisement

WHERE match\_number=ALL (SELECT match\_number FROM tickets\_sold WHERE available>500)

1. **ANY**

SELECT sold FROM tickets\_sold

WHERE match\_number=ANY (SELECT match\_number FROM tickets\_available WHERE available>4900)