**Project team 9 - Mentor Management System**

### *Team Members: Matt O’Neill, Devika Unnikrishnan, Rakesh Gunti, Hussain Musthaq*

## ***PROJECT PROPOSAL***

**Content, Scope and Objectives**

The objective of this application is to create a communication between three parties. Party 1 is Admin, he maintains the list of all mentors and pupil. He takes care of all attendance and personal information of all people. Party 2 is a Mentor, he is assigned for one or more students. The mentor gives assignments to students and maintains a record of questions and answers. Party 3 is a Student, the student is assigned to one mentor. Given the attendance of the mentor in the system, students can approach the mentor on that day. Overall, both parties can communicate with admin for any information.

## ***PROJECT ENVIRONMENT***

OS: Windows / Mac

Database: MySQL Workbench 8.0.17

Front End: Bootstrap frontend framework

Middleware: NodeJs 10.x /Django 2.2

## ***HIGH LEVEL REQUIREMENTS***

### **Initial user roles**

|  |  |
| --- | --- |
| **User Role** | **Description** |
| Apprentice | Get and seek advice from a mentor |
| Mentor | Guide and assist apprentice in new career |
| Administrators | Has power to set up a mentor/apprentice relationship |

### **Initial user story descriptions**

Green - User story is completed

Cyan - so that statements were added as per suggestion.

Yellow - changed perspective of that user story or updated

Orange - new user story

|  |  |
| --- | --- |
| **Story ID** | **Story description** |
| 1 | As an apprentice, I want to send messages to a mentor |
| 2 | As a mentor, I want to send messages to an apprentice |
| 3 | ~~As a mentor, I want to post questions for the apprentice~~  As a mentor, I want to post a quiz for my apprentices so that i can test their learning.  Note: We assumed instead of posting one or more questions Mentor will post quiz which consists of one or more questions. |
| 4 | ~~As an apprentice, I want to respond to my mentor’s questions~~  As an apprentice I want to respond to the quiz posted by my mentor so that I can test my learning  Note: Instead of responding to all questions individually Apprentice will respond to one quiz. |
| 5 | As a mentor, I want to monitor the attendance of my apprentice at meetings so that i can add these details in the weekly report  Note: Just added “so that statement” to the user story. |
| 6 | ~~As an apprentice, I want to check in to meetings~~ |
| 7 | As a mentor, I want to post/update new homework so that apprentice can prepare for the quiz.  Note: Just added “so that statement” to the user story. |
| 8 | As an apprentice, I want to access homework assignments so that i can prepare for the quiz.  Note: Just added “so that statement” along with user story. |
| 9 | As a mentor, I want to file a weekly report on the apprentices so that I can track their progress.  Note: Just added “so that statement” to the user story. |
| 10 | As an apprentice, I want to Login/Logout to my account so that I can answer quiz and view appointment.  Note: Just added “so that statement” to the user story. |
| 11 | As an Administrator, I want to Create a relationship between mentor and apprentice. |
| 12 | As a mentor I want to schedule/cancel Appointments with my apprentices so that i can discuss their progress.  Note: We assumed only one person should schedule / cancel meeting that should be Mentor. Hence removed User story 6 and added this. |
| 13 | As a mentor, I want to Login/Logout to my account so that I can post quiz n schedule appointments  Note: This was implemented for all user roles, didn’t have user story to each user role. Hence added. |
| 14 | As an admin, I want to Login/Logout to my account, so that I can create mentor and apprentice relationship  Note: This was implemented for all user roles must didn’t have user story to each user role. Hence added. |

## ***HIGH LEVEL CONCEPTUAL DESIGN***

**Entities**:

Apprentice

Mentor

Administrator

Message

Quiz

~~Question~~ [ considered it to be multi-valued attribute of Quiz]

~~Answer~~ [ considered it to be multi-valued attribute of Quiz]

Homework

Appointment

Report

~~Attendance~~ [ considered it to be attribute of Appointment]

**Relationships**:

A mentor guides an Apprentice

Administrator assigns Apprentices to Mentors

Apprentice/Mentor/Administrator send/receives Messages

Mentor generates/files a Report

~~Apprentice schedules an Appointment~~ [user story is updated]

Mentor schedules / cancels Appointment

Mentor monitors / marks Attendance

~~Mentor posts Question~~ [user story is updated]

Mentor posts Quiz

~~Apprentice answers Mentor’s Questions~~ [user story is removed]

Apprentice responds / answers Quiz

***SPRINT-1***

**User Stories for Sprint 1:**

|  |  |
| --- | --- |
| **Story ID** | **Story description** |
| 11 | As an Administrator, I want to Create a relationship between mentor and apprentice. |
| 2 | As a mentor, I want to send messages to an apprentice |
| 1 | As an apprentice, I want to send messages to a mentor |

**CONCEPTUAL DESIGN:**

**Entities:**

Employee

Attributes:

Username

Password

Employee\_Id

First\_Name

Last\_Name

Job\_Title

Is\_Admin

Mobile\_Num [multi-valued]

Email [multi-valued]

Birth\_Date

Join\_Date

Address [multi-valued, composite]

Address\_Line1

Address\_Line2

City

State

Zip\_Code

Note: We assumed that as these mentors, apprentices, admins are employees so create a base employee table then create respective tables for each user.

Apprentice

Attributes:

Apprentice\_Id

Pref\_ProgrammingLang

Pref\_BackendFramework

Pref\_FrontendFramework

Pref\_DatabaseSystem

Mentor

Attributes:

Mentor\_Id

Pref\_ProgrammingLang

Pref\_BackendFramework

Pref\_FrontendFramework

Pref\_DatabaseSystem

Administrator

Attributes:

Admin\_Id

Message

Attributes:

Msg\_Id

Sender\_Id

Receiver\_Id

Msg\_Time

Msg\_Text

**RELATIONSHIPS:**

Employee send/receive message

Cardinality: one-to-many

Participation:Employee has partial participation

Message has partial participation

Mentor coaches Apprentice

Cardinality: one-to-many

Participation: Mentor has partial participation

Apprentice has total participation

~~Apprentice has a Mentor~~

~~Cardinality: Many to One~~

~~Participation: Apprentice has total participation~~

~~Mentor has partial participation~~

Justification: It is the same as the 2nd relation in this relationships tab. Hence Ignoring/removing it.

**LOGICAL DESIGN:**

**Tables:**

Employee

Columns:

Employee\_Id

Username

Password

First\_Name

Last\_Name

Job\_Title

Birth\_Date

Join\_Date

Personal\_Mobile\_Num

Work\_Mobile\_Num

Personal\_Email

Work\_Email

Address

Columns:

Employee\_Id [foreign key; references Employee\_Id in Employee]

Address\_Line1

Address\_Line2

City

State

Zip\_Code

Apprentice

Columns:

Apprentice\_Id [foreign key; references Employee\_Id in Employee]

Mentor\_Id [foreign key; references Mentor\_Id in Mentor]

Pref\_ProgrammingLang

Pref\_BackendFramework

Pref\_FrontendFramework

Pref\_DatabaseSystem

Mentor

Columns:

Mentor\_Id [foreign key; references Employee\_Id in Employee]

Pref\_ProgrammingLang

Pref\_BackendFramework

Pref\_FrontendFramework

Pref\_DatabaseSystem

Administrator

Columns:

Admin\_Id [foreign key; references Employee\_Id in Employee]

Message

Columns:

Msg\_Id

Sender\_Id [foreign key; references Employee\_Id in Employee]

Receiver\_Id [foreign key; references Employee\_Id in Employee]

Msg\_Time

Msg\_Text

**IMPLEMENT & DEPLOY DATABASE**

Database name: MMS

**Sprint 1 - Key SQL Queries for Demonstration**

Query 1:

SELECT A.apprentice\_id as apprentice\_eid,concat(E.first\_name,' ',E.last\_name) as apprentice\_name, A.mentor\_id as mentor\_eid, A.mentor\_name FROM Employee as E, (SELECT A.apprentice\_id, A.mentor\_id, concat(E.first\_name,' ',E.last\_name) as mentor\_name FROM Apprentice AS A, Employee E WHERE A.mentor\_id = E.Employee\_id) A WHERE A.apprentice\_id = E.employee\_id

Query 2:

SELECT A.apprentice\_id AS apprentice\_employee\_id , M.receiver\_id as receiver\_employee\_id, M.msg\_text as message, M.Msg\_time as message\_time FROM Apprentice AS A, Message AS M WHERE M.sender\_id = A.apprentice\_id;

Query3:

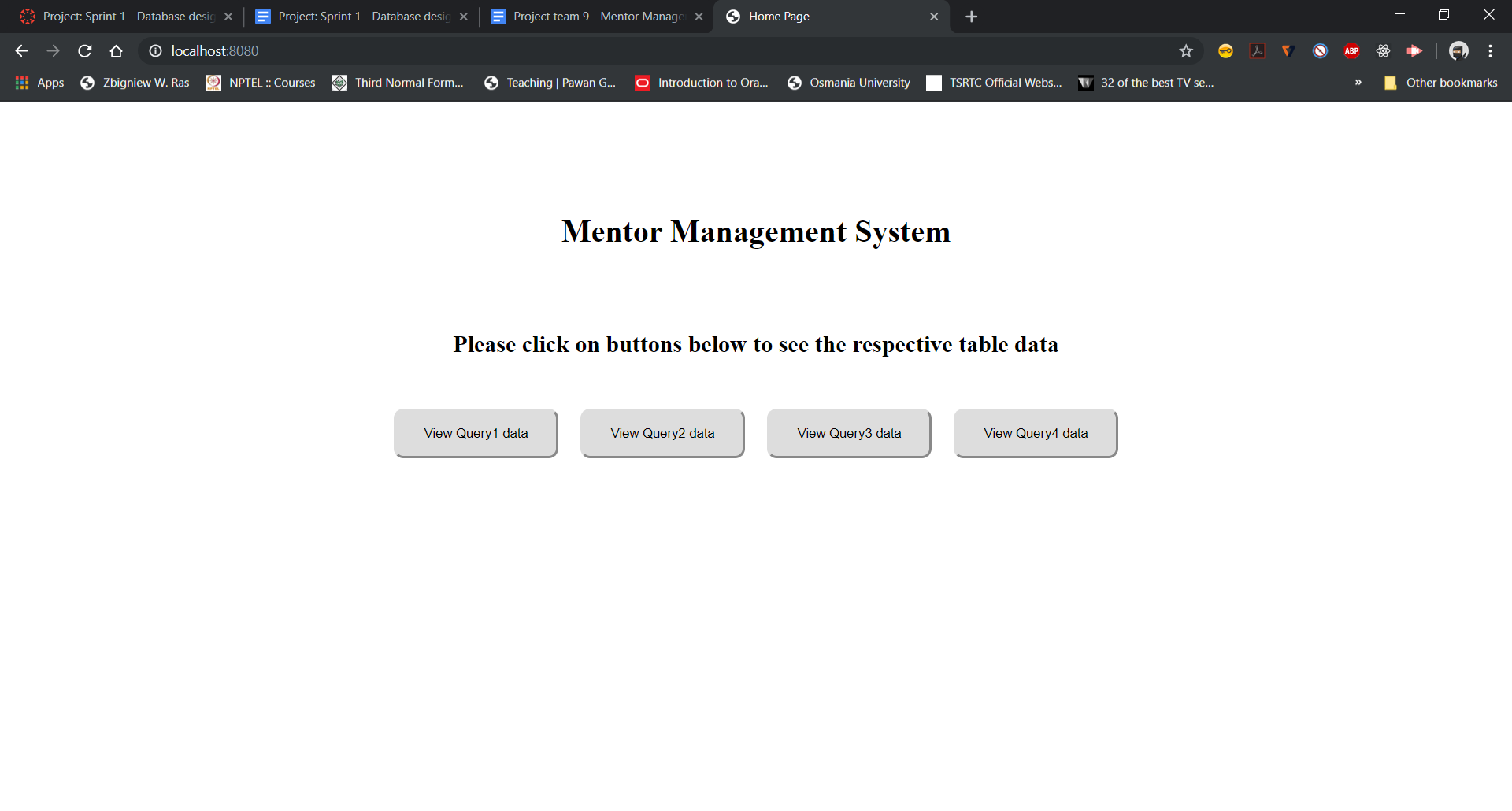
SELECT A.admin\_id AS Employee\_Id, E.first\_name, E.last\_name, E.username, E.birth\_date, E.join\_date FROM Administrator AS A, Employee AS E WHERE A.admin\_id = E.employee\_id;

Query4:

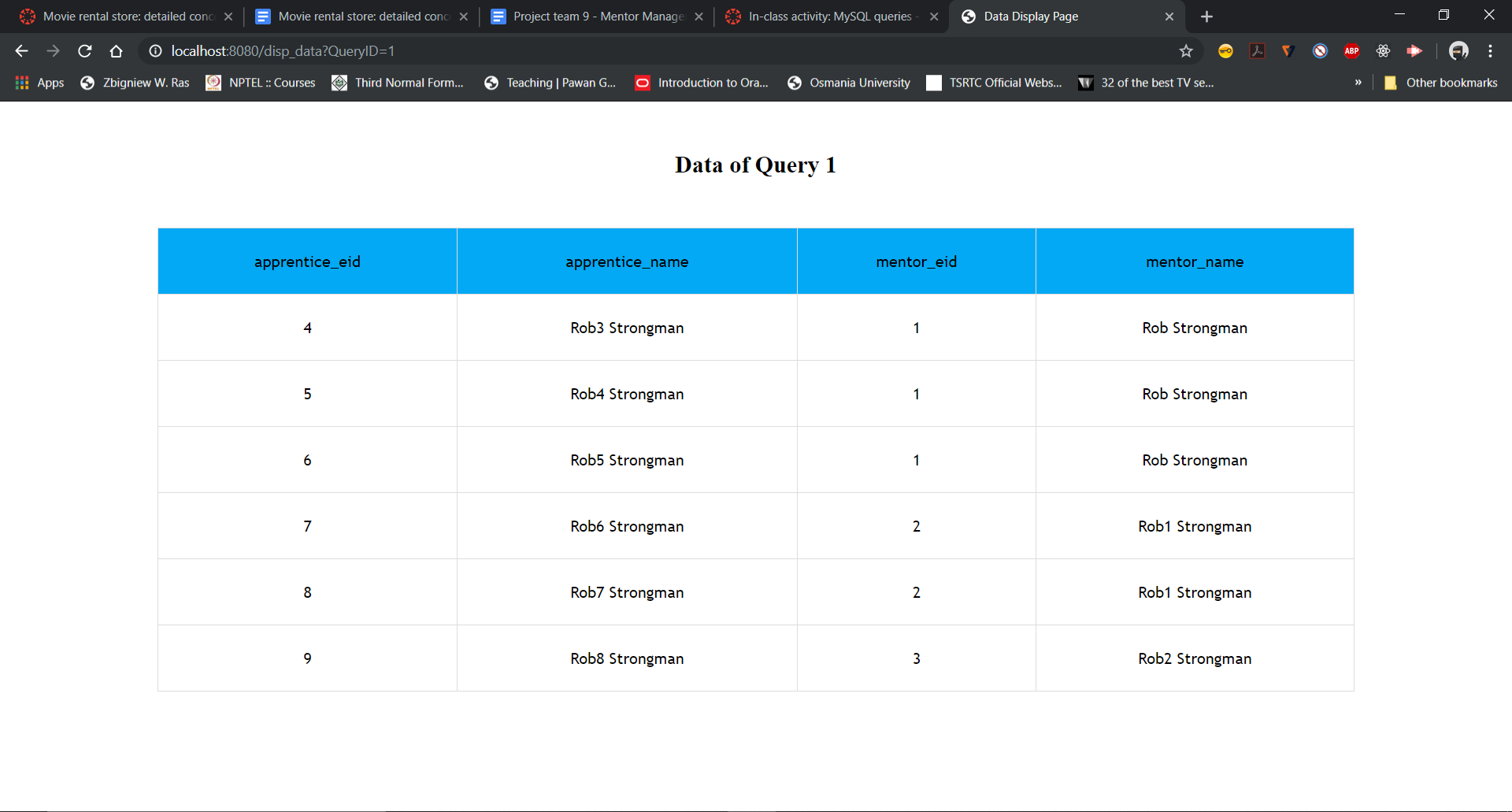
SELECT E.employee\_id, E.username,CONCAT(E.first\_name,' ',E.last\_name) AS employee\_full\_name,E.job\_title,E.join\_date, AD.address\_line1, AD.address\_line2, AD.city, AD.state, AD.zip\_code FROM Employee AS E, Address AS AD WHERE E.employee\_id = AD.employee\_id;

**Demonstration:**

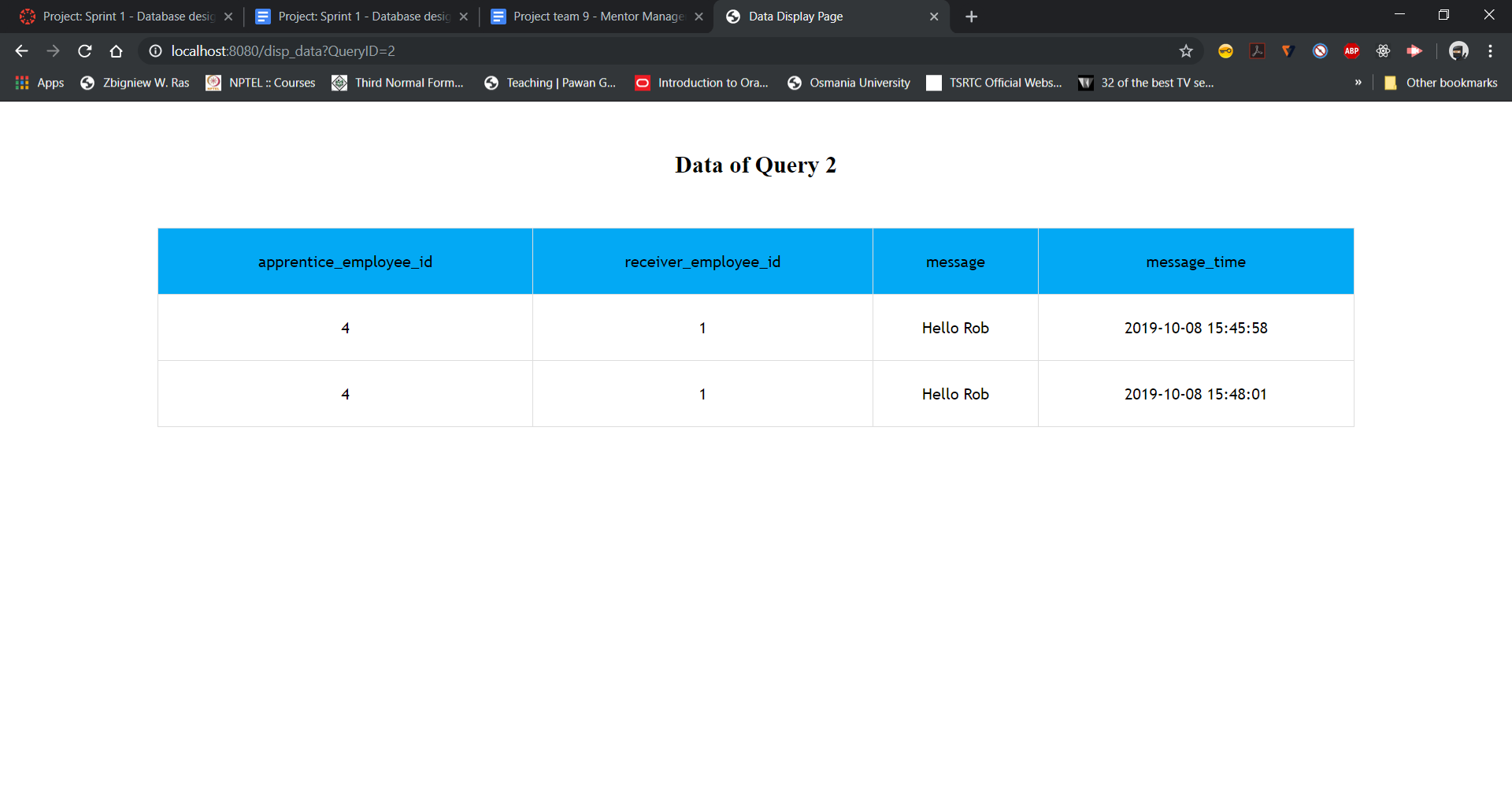
**Home Page:**

****

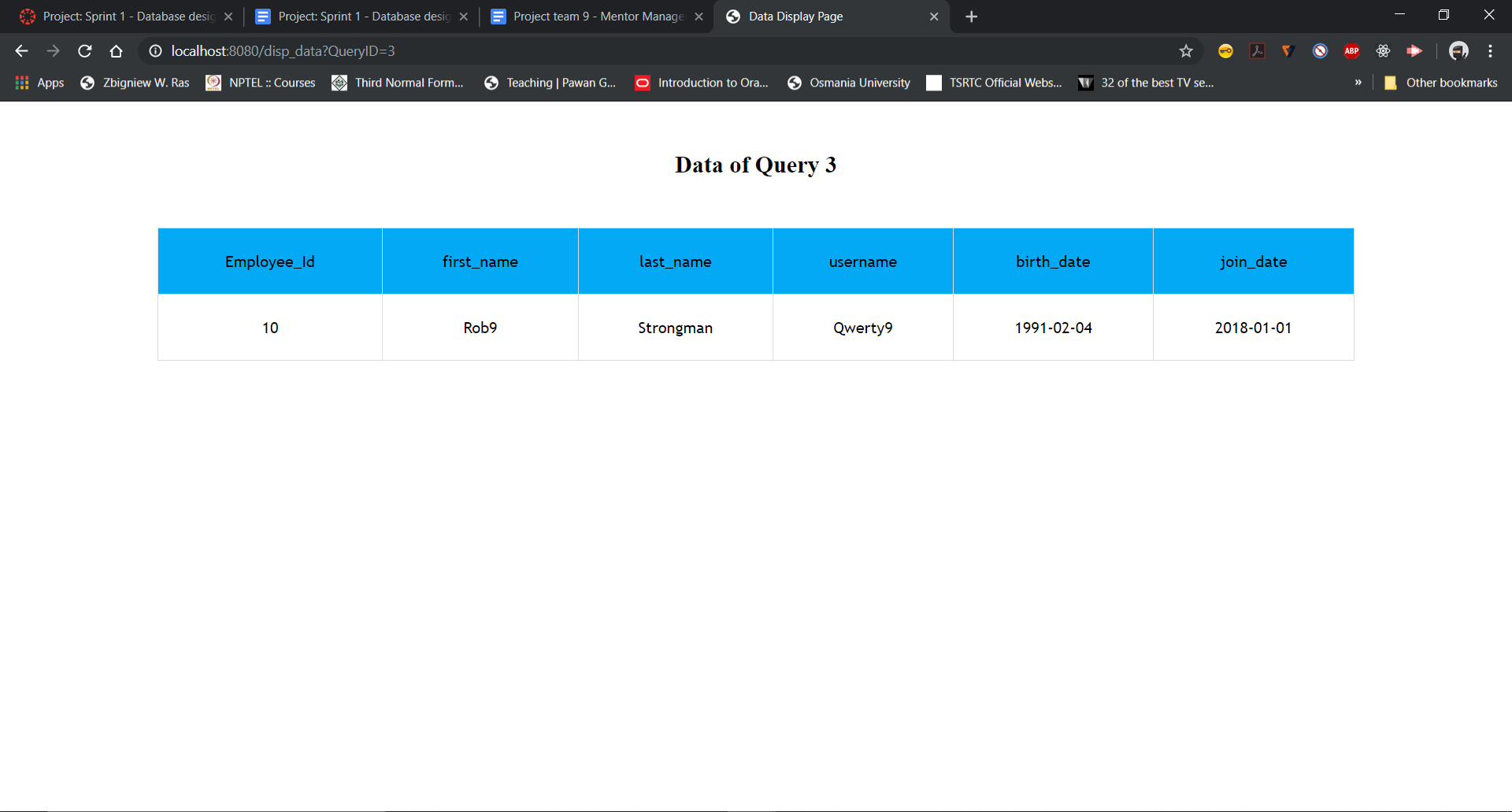
**Query1/Button1 Output:**

****

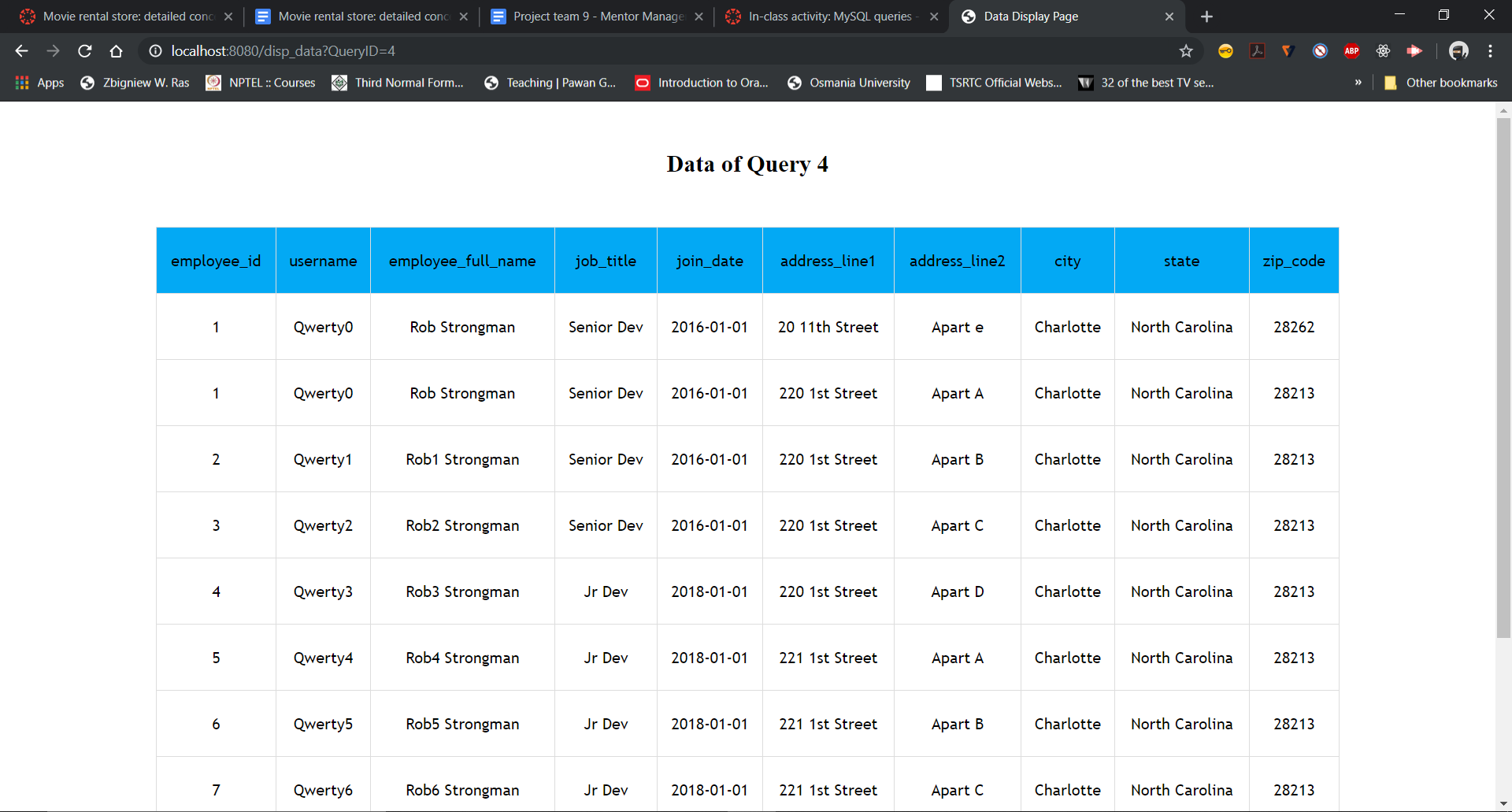
**Query2/ Button2 output:**

****

**Query3/ Button3 output:**

****

**Query4/ Button4 output:**

****

**SPRINT-2**

**User Stories for Sprint 2:**

|  |  |
| --- | --- |
| **Story ID** | **Story description** |
| 14 | As an admin, I want to Login/Logout to my account, so that I can create mentor and apprentice relationship |
| 13 | As a mentor, I want to Login/Logout to my account so that I can post quiz and schedule appointments. |
| 10 | As an apprentice, I want to Login/Logout to my account so that I can answer quiz and view appointments. |
| 12 | As a mentor I want to schedule/cancel Appointments with my apprentices so that i can discuss their progress. |
| 3 | As a mentor, I want to post a quiz for my apprentices so that i can test their learning. |
| 4 | As an apprentice I want to take the quiz posted by my mentor so that I can test my learning |
| 5 | As a mentor, I want to monitor the attendance of my apprentice at meetings so that i can add these details in the weekly report |

**CONCEPTUAL DESIGN:**

**Entities:**

Apprentice

Attributes:

Apprentice\_Id

Pref\_ProgrammingLang

Pref\_BackendFramework

Pref\_FrontendFramework

Pref\_DatabaseSystem

Last\_Login\_Time

Last\_Logout\_Time

Is\_Active

Mentor

Attributes:

Mentor\_Id

Pref\_ProgrammingLang

Pref\_BackendFramework

Pref\_FrontendFramework

Pref\_DatabaseSystem

Last\_Login\_Time

Last\_Logout\_Time

Is\_Active

Administrator

Attributes:

Admin\_Id  
 Last\_Login\_Time

Last\_Logout\_Time

Is\_Active

Appointment

Attributes:

Appointment\_Id

Mentor\_Id [foreign key Mentor\_Id from Mentor]

Apprentice\_Id [foreign key Apprentice\_Id from Apprentice]

Start\_Time

End\_Time

Location(composite)

Building\_Name

Floor\_No

Room\_No

Pre\_Meeting\_Notes

Post\_Meeting\_Notes

Status

Attended

Note:

Status will have only 4 values

“Scheduled” - Meeting is yet to happen,

“Done” - Meeting is completed by mentor and Apprentice,

“Cancelled” - Mentor Cancelled Meeting,

“InComplete” - Apprentice doesn’t show up to the meeting.

Attended Attribute is part of Story ID 5 why we did that is because we assumed that there are no other properties for attendance

Quiz:

Attributes:

Quiz\_Id

Quiz\_Name

Is\_Active

Author

Questions[multi-valued, composite]

Question\_text

Question\_Id

Choice[multi-valued, composite]

Choice

Is\_Right

Response [multi-valued, composite]

Response\_Id

Res\_Time

Attempt\_No

Responder\_Id [foreign key apprentice\_id from apprentice]

Response\_Answer[multi-valued, composite]

Answer\_Id

Answer

Is\_Right

**RELATIONSHIPS:**

Mentor schedules/cancels Appointment

Cardinality: one-to-many

Participation:Mentor has partial participation

Meeting has total participation

Mentor posts Quiz

Cardinality: one-to-many

Participation: Mentor has partial participation

Quiz has total participation

Apprentice responds / answers Quiz

Cardinality: Many to Many

Participation: Apprentice has partial participation

Quiz has partial participation

**LOGICAL DESIGN:**

Tables:

Apprentice

Columns:

Apprentice\_Id [foreign key; references Employee\_Id in Employee]

Mentor\_Id [foreign key; references Mentor\_Id in Mentor]

Pref\_ProgrammingLang

Pref\_BackendFramework

Pref\_FrontendFramework

Pref\_DatabaseSystem

Last\_Login\_time

Last\_Logout\_time

Is\_Active

Justification for change: The login and logout time tracks account active status. If the given user has not logged in 90 days, their profile will be inactive.

Mentor

Columns:

Mentor\_Id [foreign key; references Employee\_Id in Employee]

Pref\_ProgrammingLang

Pref\_BackendFramework

Pref\_FrontendFramework

Pref\_DatabaseSystem

Last\_Login\_time

Last\_Logout\_time

Is\_Active

Justification for change: The login and logout time tracks account active status. If the given user has not logged in 90 days, their profile will be inactive.

Administrator

Columns:

Admin\_Id [foreign key; references Employee\_Id in Employee]

Last\_Login\_time

Last\_Logout\_time

Is\_Active

Justification for change: The login and logout time tracks account active status. If the given user has not logged in 90 days, their profile will be inactive.

Appointment

Columns:

Appointment\_Id

Mentor\_Id [foreign key; references Mentor\_Id from Mentor Table]

Apprentice\_Id [foreign key; references Apprentice\_Id from Apprentice Table]

Start\_Time

End\_Time

Building\_Name

Floor\_No

Room\_No

Pre\_Meeting\_Notes

Post\_Meeting\_Notes

Status

Attended

Quiz

Columns:

Quiz\_Id

Quiz\_Name

Is\_Active

Author [foreign key; references Mentor\_Id in Mentor]

Question

Columns:

Question\_Id

Question\_text

Quiz\_Id [foreign key; references Quiz\_Id in Quiz]

Choices

Columns:

Choice

Question\_Id[foreign key; references Question\_Id from Question]

Quiz\_Id [foreign key; references Quiz\_Id in Quiz]

Is\_Right

Response

Columns:

Attempt\_No

Quiz\_Id [foreign key; references Quiz\_Id in Quiz]

Responder\_Id [foreign key; references Apprentice\_Id from Apprentice]

Res\_time

Response\_Id

Response\_Answer

Columns:

Answer\_id

Answer

Is\_Right

Question\_Id [foreign key; references Question\_Id from Question]

Response\_Id [foreign key; references Response\_Id in Response]

**NORMALIZATION:**

Tables:

Address

Normalized form: 4NF

Administrator

Normalized form: 4NF

Appointment

Normalized form: 4NF

Apprentice

Normalized form: 4NF

Choices

Normalization form:4NF

Employee

Normalized form: 1NF

Justification: Because all other attributes for an employee entry are also functionally dependent on the username field, this table is in 2NF. We think this is justified because unique usernames are important for a simple login system.

Mentor

Normalized form: 4NF

Message

Normalized form: 4NF

Question

Normalized form:4NF

Quiz

Normalized form:4NF

Response

Normalized form:4NF

Response\_Answer

Normalized form: 4NF

**IMPLEMENT & DEPLOY DATABASE:**

Database Name: MMS\_Sprint2

**Sprint2 - Key SQL Queries for Demonstration**

US - User story

us-14:

select E.employee\_id, concat\_ws(' ',E.first\_name, E.last\_name) Admin\_FullName, E.username, E.Password, A.last\_login\_time, A.last\_logout\_time, IF(A.is\_active, 'Active', 'Inactive') Account\_status from Employee as E inner join Administrator as A on E.employee\_id = A.admin\_id

us-13:

select E.employee\_id, concat\_ws(' ',E.first\_name, E.last\_name) Mentor\_FullName, E.username, E.Password, M.last\_login\_time, M.last\_logout\_time, IF(M.is\_active, 'Active', 'Inactive') Account\_status from Employee as E inner join Mentor as M on E.employee\_id = M.mentor\_id

us-10:

select E.employee\_id, concat\_ws(' ',E.first\_name, E.last\_name) Apprentice\_FullName, E.username, E.Password, A.last\_login\_time, A.last\_logout\_time, IF(A.is\_active, 'Active', 'Inactive') Account\_status from Employee as E inner join Apprentice as A on E.employee\_id = A.apprentice\_id

us-12:

SELECT concat\_ws(' ',E1.first\_name,E1.last\_name) Apprentice\_Name, J.\* FROM (SELECT A.apprentice\_id, A.mentor\_id, concat\_ws(' ',E.first\_name,E.last\_name) Mentor\_Name, A.start\_time, A.end\_time, A.status, A.pre\_meeting\_notes, IFNULL(A.post\_meeting\_notes, 'Nothing provided by Mentor') post\_meeting\_notes, concat\_ws(', ', concat('Room No: ',A.room\_no), concat('Floor No: ',A.floor\_no), concat('Building: ',A.building\_name)) Location FROM Appointment as A INNER JOIN Employee as E on E.employee\_id = A.mentor\_id) as J INNER JOIN Employee as E1 ON E1.employee\_id = J.apprentice\_id

us-3:

SELECT Q.quiz\_id, Q.quiz\_name, Q.is\_active, Q.author, Qs.question\_id, Qs.question\_text, C.choice, C.is\_right from Quiz as Q, Question as Qs, Choices as C WHERE Q.quiz\_id = Qs.quiz\_id AND Qs.Question\_id = C.Question\_id AND C.quiz\_id = Q.quiz\_id

us-4:

SELECT R.attempt\_no,R.res\_time, Q.quiz\_id, Q.quiz\_name, Qs.question\_id, Qs.question\_text, RA.is\_right, RA.answer, R.Responder\_id FROM Quiz as Q, Response as R, Question as Qs, Response\_Answer as RA WHERE R.quiz\_id = Q.quiz\_id and Q.quiz\_id = Qs.quiz\_id and RA.response\_id = R.response\_id and Qs.question\_id = RA.question\_id ORDER BY R.Responder\_id, Attempt\_no

us-5:

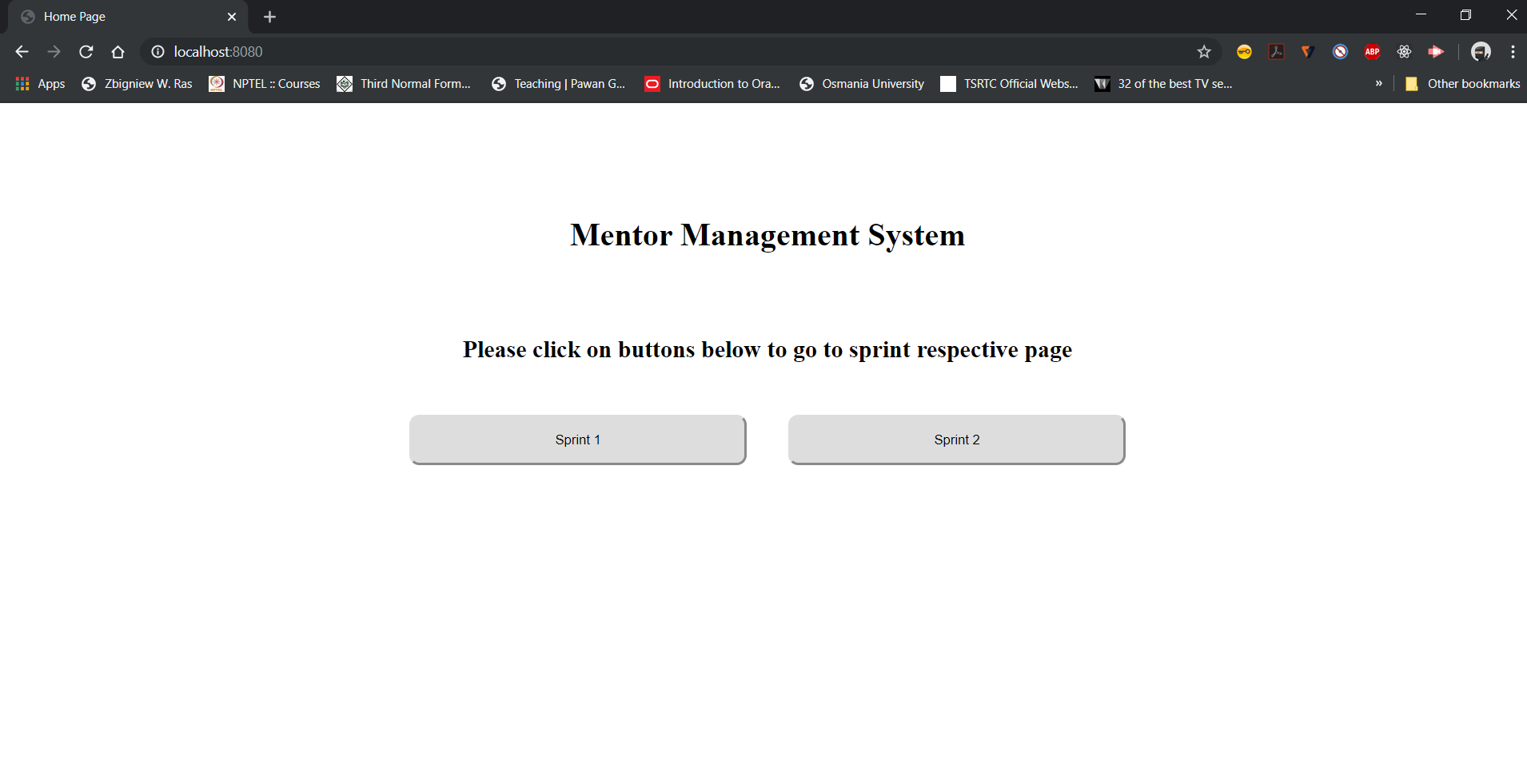
SELECT concat\_ws(' ',E1.first\_name,E1.last\_name) Apprentice\_Name, J.\* FROM (SELECT A.apprentice\_id, A.mentor\_id, concat\_ws(' ',E.first\_name,E.last\_name) Mentor\_Name, A.start\_time, A.end\_time, A.status, IF(A.attended, 'Yes', 'No') Attended, A.pre\_meeting\_notes,IFNULL(A.post\_meeting\_notes, 'Nothing provided by Mentor') post\_meeting\_notes, concat\_ws(', ', concat('Room No: ',A.room\_no), concat('Floor No: ',A.floor\_no), concat('Building: ',A.building\_name)) Location FROM Appointment as A INNER JOIN Employee as E on E.employee\_id = A.mentor\_id) as J INNER JOIN Employee as E1 ON E1.employee\_id = J.apprentice\_id

Merge of US-4 & US-5:

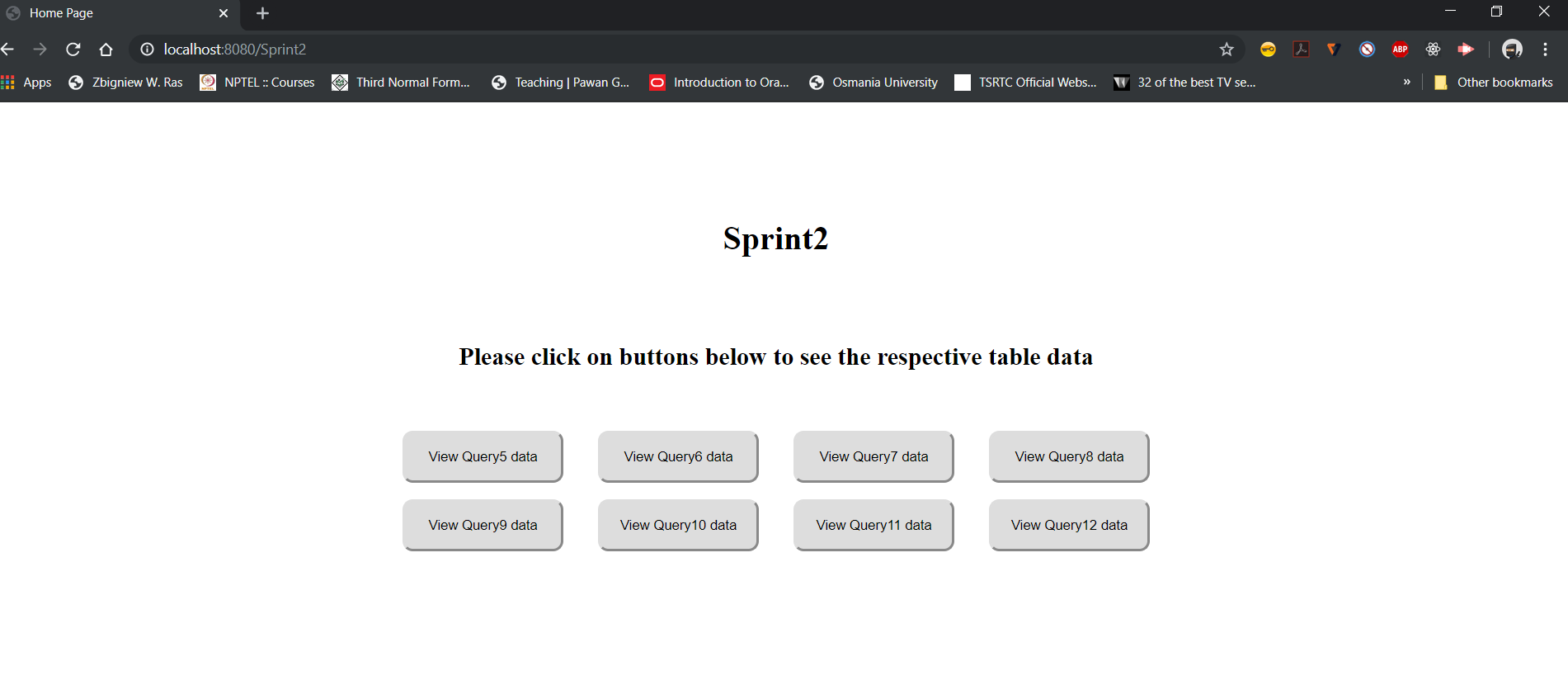
SELECT R.attempt\_no,R.res\_time, Q.quiz\_id, Q.quiz\_name, Qs.question\_id, Qs.question\_text, RA.answer, R.responder\_id , C.choice FROM Quiz as Q, Response as R, Question as Qs, Response\_Answer as RA , (SELECT \* FROM Choices WHERE is\_right = True) as C WHERE R.quiz\_id = Q.quiz\_id and Q.quiz\_id = Qs.quiz\_id and RA.response\_id = R.response\_id and Qs.question\_id = RA.question\_id and C.question\_id = Qs.Question\_id and C.quiz\_id = Qs.quiz\_id ORDER BY R.Responder\_id

**Demonstration:**

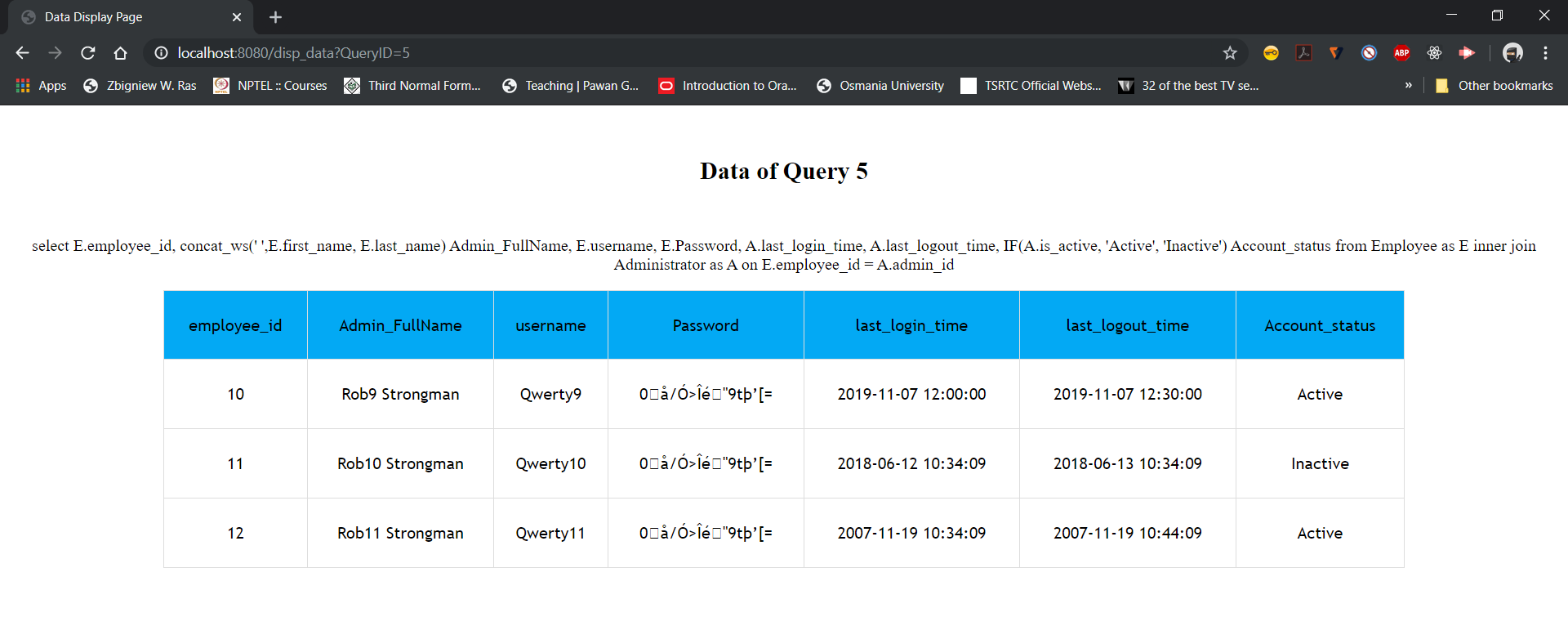
Home Page

****

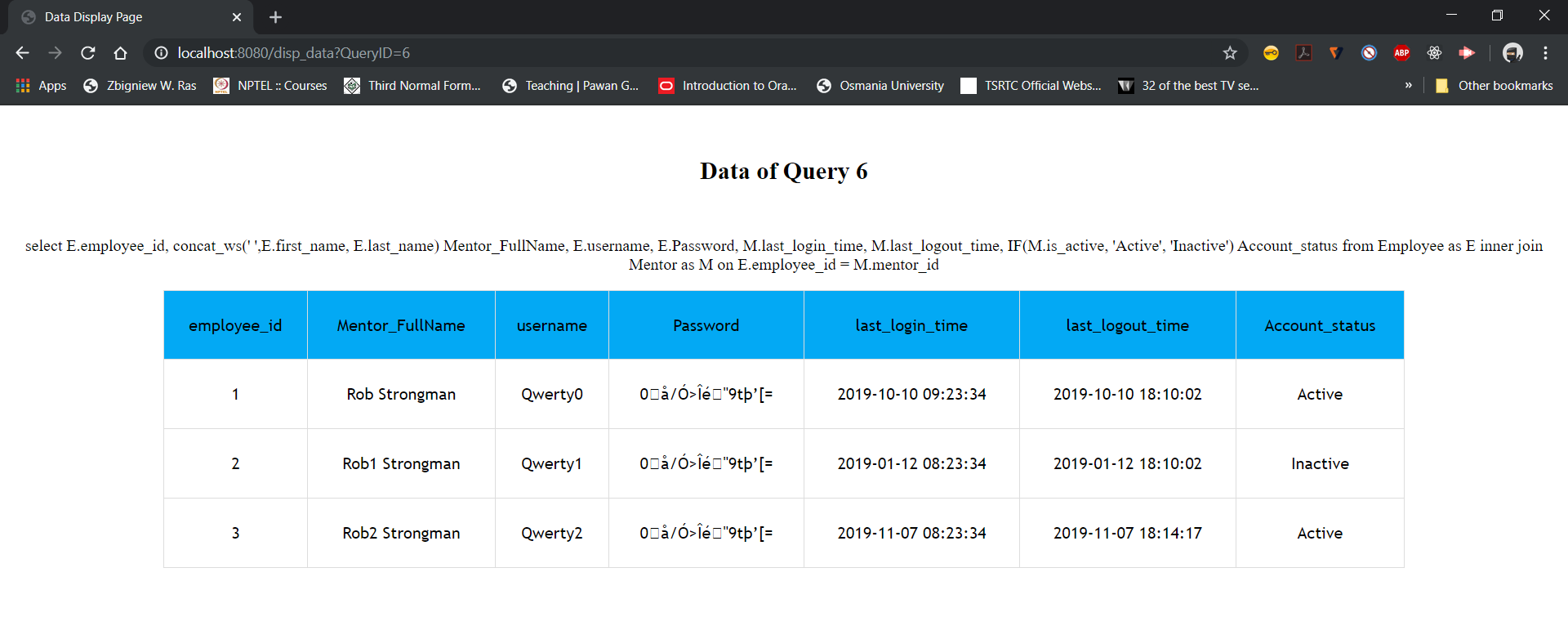
Current Sprint Page:



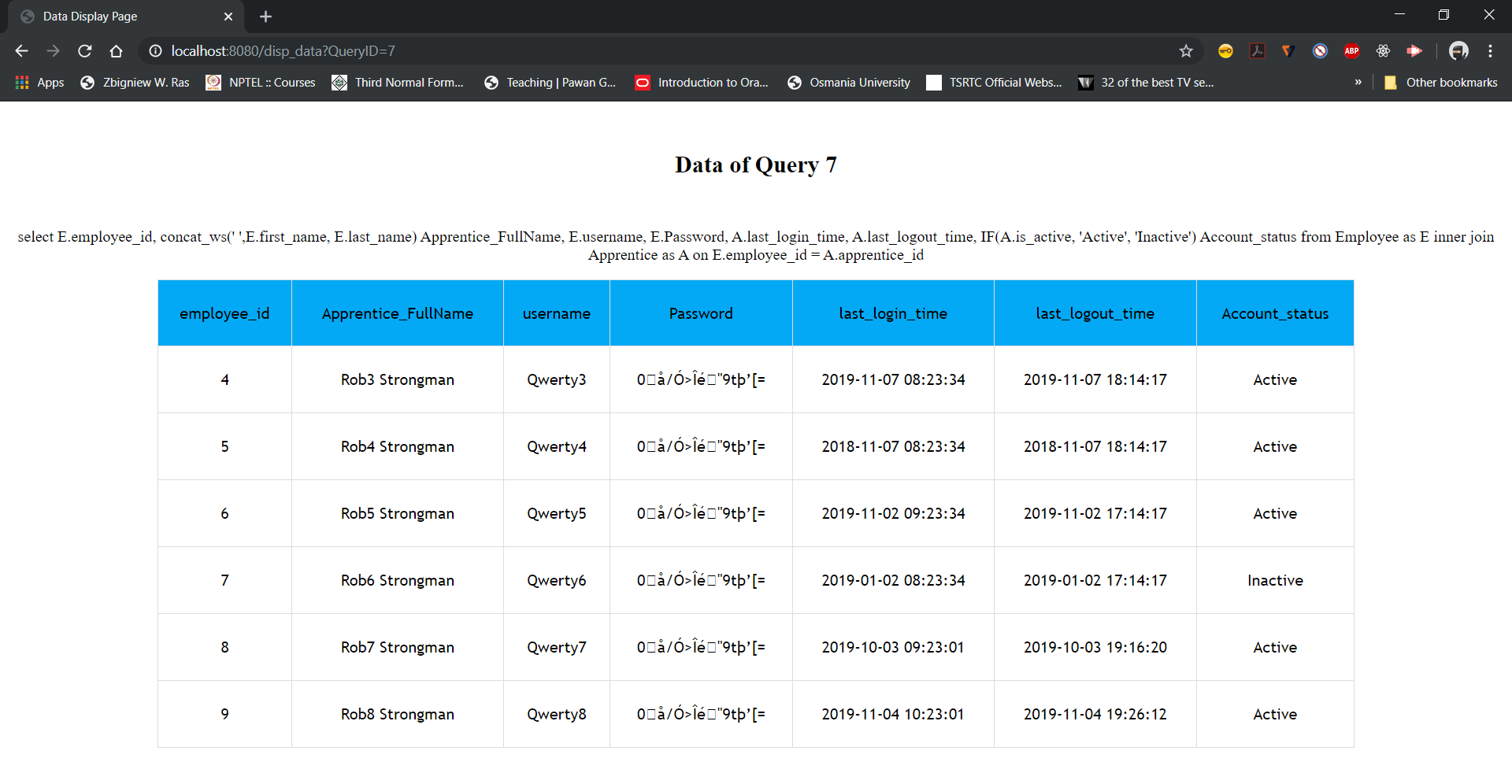
User story- 14:



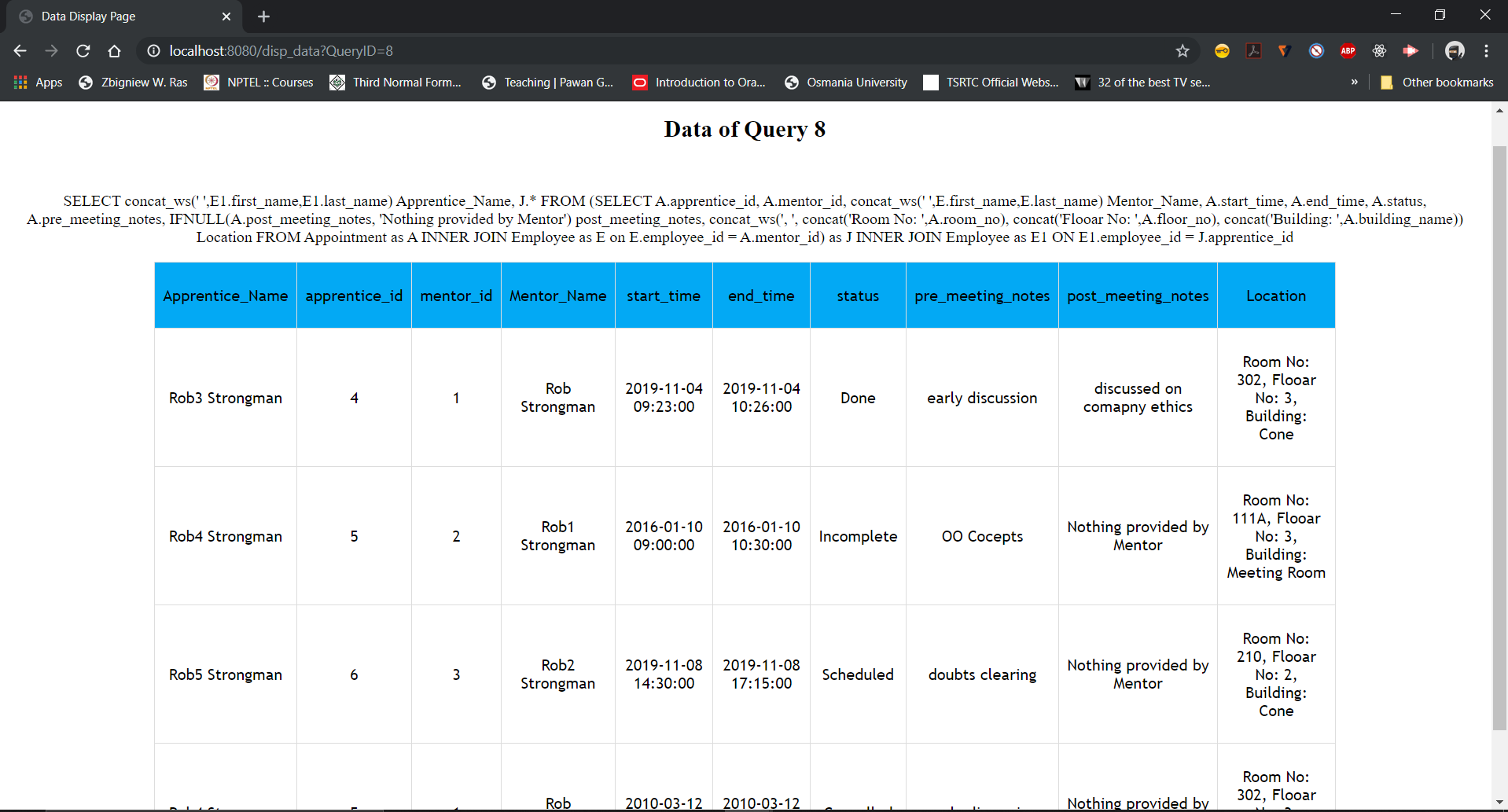
User story-13:



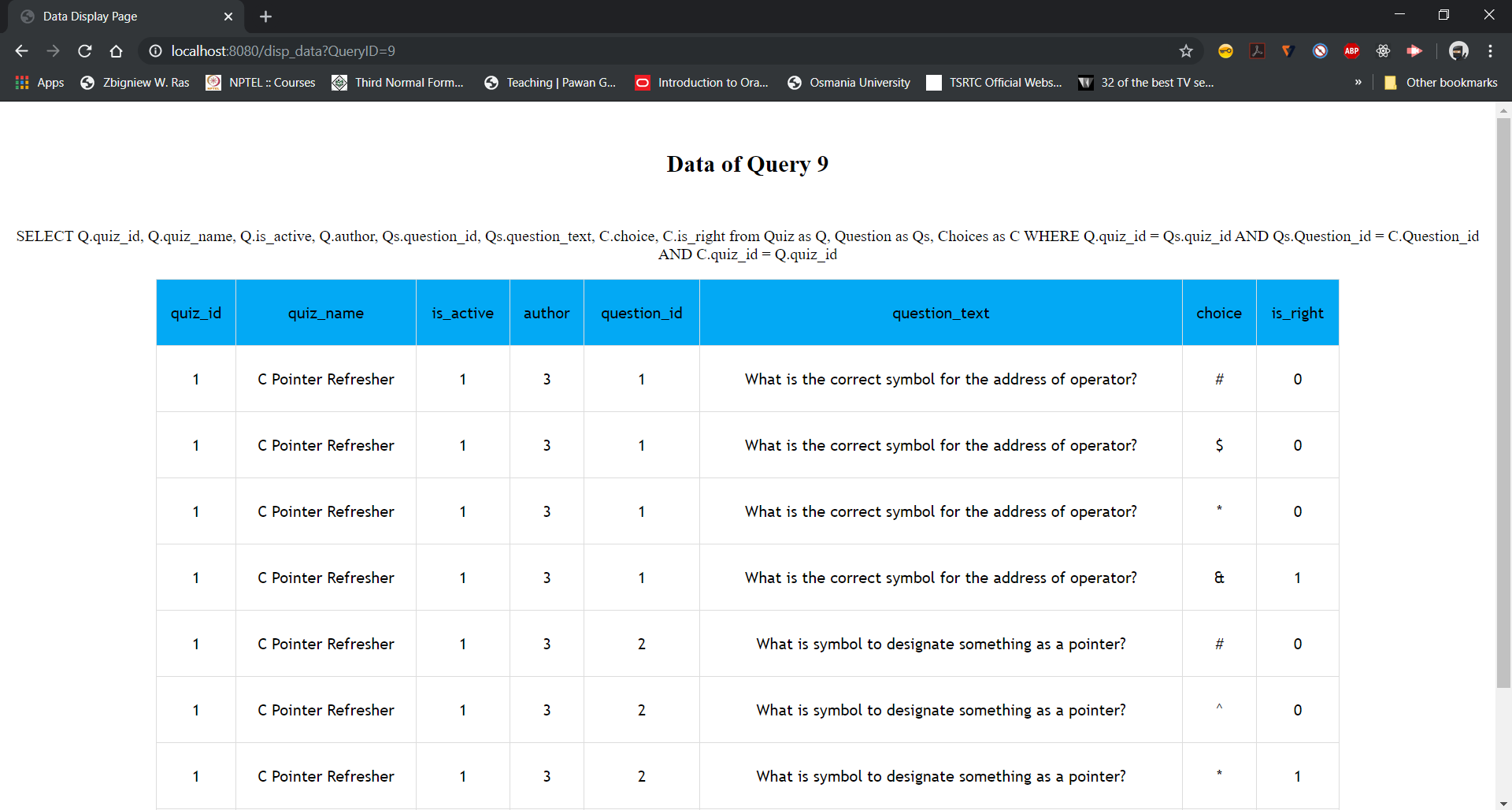
User story - 10:



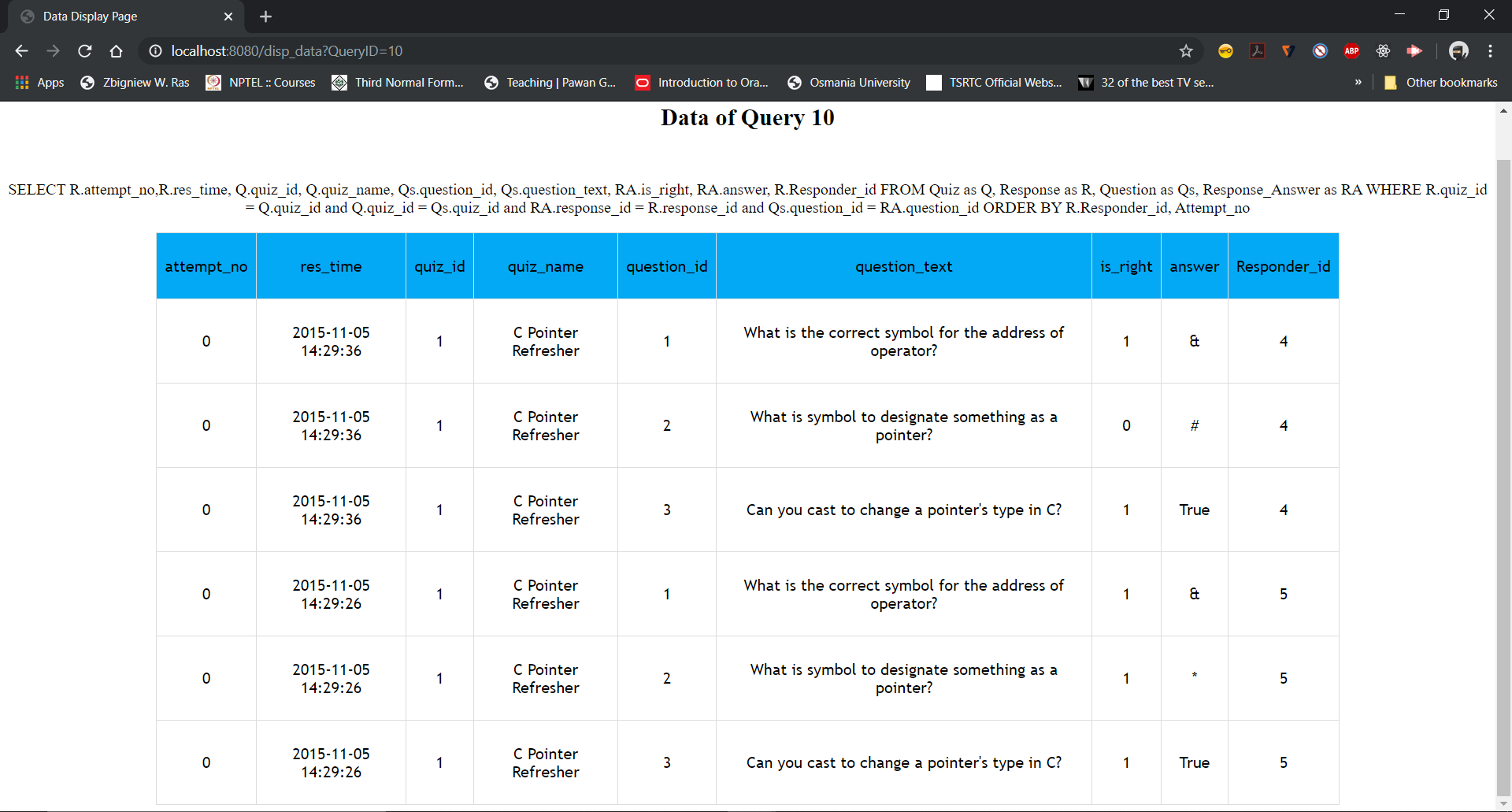
User story - 12:



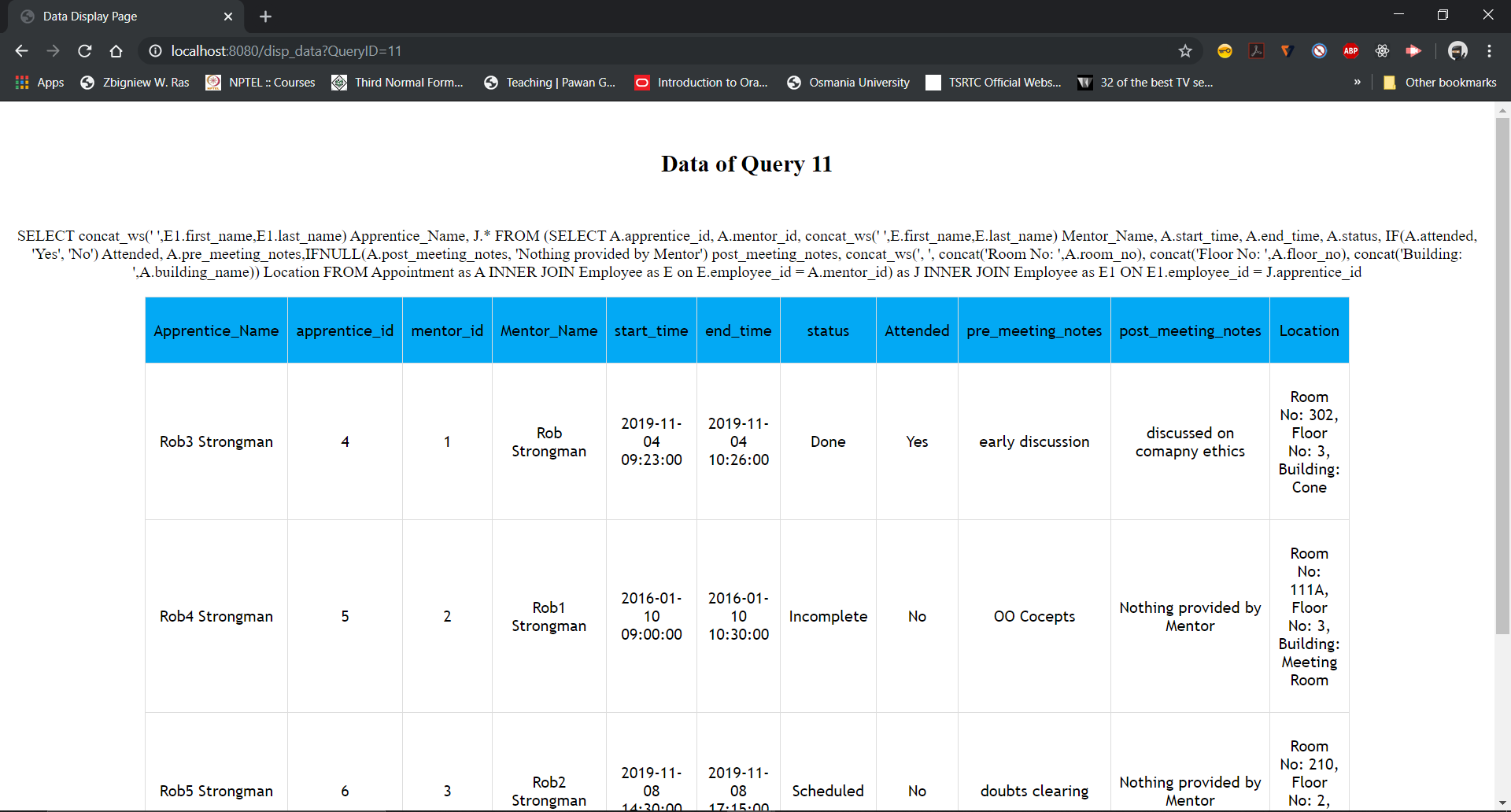
User story - 3:



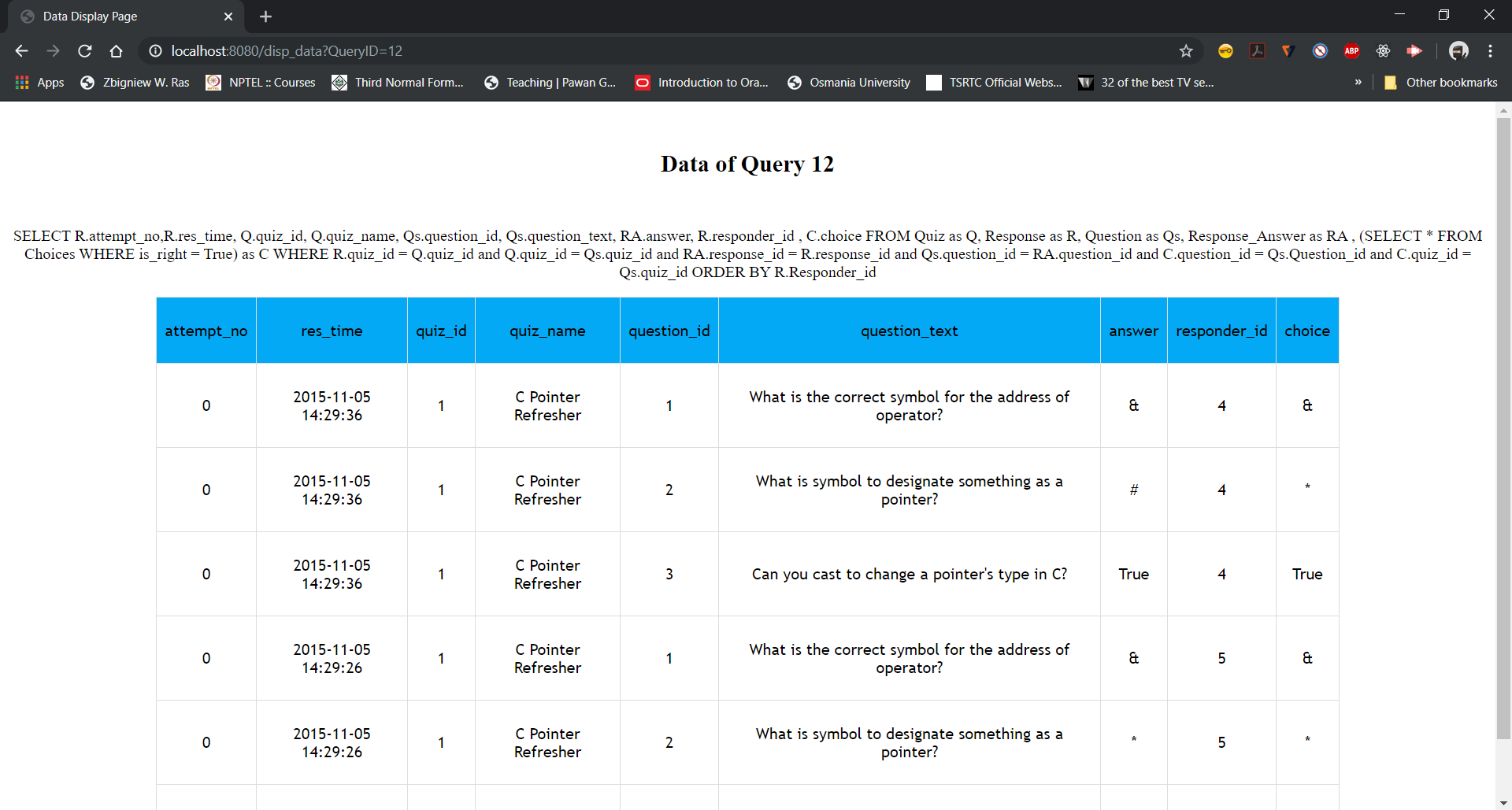
User story-4:



User story-5:



Combination of User story 4 & 5



**SPRINT - 3**

**User Stories for Sprint 3:**

|  |  |
| --- | --- |
| **Story ID** | **Story description** |
| 7 | As a mentor, I want to post/update new homework so that apprentice can prepare for the quiz. |
| 8 | As an apprentice, I want to access homework assignments so that i can prepare for the quiz. |
| 9 | As a mentor, I want to file a weekly report on the apprentices so that I can track their progress. |

**CONCEPTUAL DESIGN:**

**Entities:**

Homework

**Attributes**:

Homework\_Id

Announcement\_time

Mentor\_Id

Apprentice\_Id

Is\_Active

Is\_Completed

Resources [multi-valued, composite]

Resource\_Id

Resource\_text

**Justification for Primary Key**: Because we want to reserve the option for a mentor to post homework to the same apprentice multiple times, we have decided to use a unique Homework\_Id field to capture each unique homework assignment.

**Notes**: “Is\_Active” is to say whether homework is active or not. “Is\_Completed” is to say whether Apprentice completed homework by clicking “Mark as Done” in website.

WeeklyReport

**Attributes**:

Report\_Id

Report\_Time

Mentor\_Id

Apprentice\_Id

Score

Description

**Justification for primary key**: Because a given mentor-apprentice relationship will have multiple WeeklyReports, we have decided to have a unique Report\_Id to uniquely identify reports. We have decided not to use a combination of apprentice\_Id, mentor\_Id, and Report\_Time as a Primary Key because of the possibility of a mentor submitting multiple reports for the same apprentice at the same time. While unlikely, it is theoretically possible.

Quiz:

**Attributes**:

Quiz\_Id

Quiz\_Name

Quiz\_PostedOn

Is\_Active

Author

Questions[multi-valued, composite]

Question\_text

Question\_Id

Choice[multi-valued, composite]

Choice

Is\_Right

Response [multi-valued, composite]

Response\_Id

Res\_Time

Attempt\_No

Responder\_Id [foreign key apprentice\_id from apprentice]

Response\_Answer[multi-valued, composite]

Answer\_Id

Answer

Is\_Right

**Justification for modification**: We need Quiz entity to have the Quiz\_PostedOn attribute to capture time so that we can use it while generating weekly reports for Apprentices.

**RELATIONSHIPS:**

Mentor posts/updates Homework

**Cardinality**: one to Many

**Participation**: Mentor has partial participation

Homework has total participation

Apprentice views Homework

**Cardinality**: one to Many

**Participation**: Apprentice has partial participation

Homework has total participation

Mentor submits WeeklyReport

**Cardinality**: One to Many

**Participation**: Mentor has partial participation

WeeklyReport has total participation

**LOGICAL DESIGN:**

Tables:

Homework

**Columns**:

Homework\_Id

Announcement\_time

Mentor\_Id [foreign key; references Mentor\_Id in Mentor]

Apprentice\_Id [foreign key; references Apprentice\_Id in Apprentice]

Is\_Active

Is\_Completed

Resources

**Columns**:

Resource\_Id

Homework\_Id [foreign key; references Homework\_Id in Homework]

Resource\_text

WeeklyReport

**Columns**:

Report\_Id

Report\_Time

Mentor\_Id [foreign key; references Apprentice\_Id in Apprentice]

Apprentice\_Id [foreign key; references Apprentice\_Id in Apprentice]

Score

Description

Quiz

**Columns**:

Quiz\_Id

Quiz\_Name

Is\_Active

Quiz\_PostedOn

Author [foreign key; references Mentor\_Id in Mentor]

Justification for Change: We need datetime of quiz to be captured so that it can be used in generating weekly report on Apprentice which will be used by Mentor to file a weekly report (For example, Mentor gives rating and feedback on how he performed in that week).

Question

**Columns**:

Question\_Id

Question\_text

Quiz\_Id [foreign key; references Quiz\_Id in Quiz]

Choices

**Columns**:

Choice

Question\_Id[foreign key; references Question\_Id from Question]

Quiz\_Id [foreign key; references Quiz\_Id in Quiz]

Is\_Right

Response

**Columns**:

Attempt\_No

Quiz\_Id [foreign key; references Quiz\_Id in Quiz]

Responder\_Id [foreign key; references Apprentice\_Id from Apprentice]

Res\_time

Response\_Id

Response\_Answer

**Columns**:

Answer\_id

Answer

Is\_Right

Question\_Id [foreign key; references Question\_Id from Question]

Response\_Id [foreign key; references Response\_Id in Response]

**NORMALIZATION:**

Homework

**Normalized form**: 4NF

Resources

**Normalized form**: 4NF

WeeklyReport

**Normalized form:** 3NF

Justification: Because we have transitive dependency other than primary key. Non-key attributes are also functionally dependent on [Mentor\_Id, Apprentice\_Id, Report\_Time] candidate key.

**INDEXES:**

Table: **Address**

**Index Name**: Address\_empid\_index

**Type**: Non-clustered

**Column order forming index**: employeee\_id

**Justification**: In most of the queries we’re running we’re using only the employee\_id column in the Address table, so we created an index for that only.

Table: **Administrator**

**Index Name**: Admin\_id\_index

**Type**: Clustered

**Column order forming index**: admin\_id

**Justification**: In most of the queries we’re running, we’re using only the admin\_id column in Administrator table, so we created an index on that only.

Table: **Appointment**

**Index Name**: Appointment\_id\_index

**Type**: Non-clustered

**Column order forming index**: mentor\_id, apprentice\_id

**Justification**: In most of the queries we’re running, we’ll be using mentor\_id, apprentice\_id columns to join with other tables hence we created an index on them.

Table: **Apprentice**

**Index Name**: Apprentice\_Mentor\_id\_index

**Type**: Non-clustered

**Column order forming index**: mentor\_id, apprentice\_id

**Justification**: In most of the queries we’re running, we’ll be using mentor\_id, apprentice\_id columns to join with other tables hence we created an index on them.

**Type**:Clustered

**Column order forming index**: apprentice\_id

**Justification**: In some of the queries we’re running, we’ll be using apprentice\_id column to join with other tables hence created an index on them.

Table: **Choices**

**Index Name**: Apprentice\_id\_index

**Type**: Non-clustered

**Column order forming index**: quiz\_id, question\_id

**Justification**: In most of the join queries we’re running, we’ll be using quiz\_id, question\_id columns to join with other tables hence we created an index on them.

Table: **Employee**

**Index Name**: employee\_username\_index

**Type**: Non-clustered

**Column order forming index**: username

**Justification**: Need this index while checking user login credentials hence we created it.

**Index Name**: employee\_id\_index

**Type**:Clustered

**Column order forming index**: employee\_id

**Justification**: Need this index while doing join with mentor, apprentice, admin tables to get their employee info.

**Index Name**: employee\_name\_index

**Type**:Non-Clustered

**Column order forming index**: first\_name, last\_name

**Justification**: Need this index while generating report or to search for employees with name fields.

**Index Name**: employee\_title\_date\_index

**Type**:Clustered

**Column order forming index**: job\_title, join\_date

**Justification**: Need this index while generating report or to search for employees with their title or join date.

Table: **Homework**

**Index Name**: homework\_emp\_id\_index

**Type**: Non-clustered

**Column order forming index**: apprentice\_id, mentor\_id

**Justification**: In most of the queries we’re running we’re using only the apprentice\_id, mentor\_id column in the Homework table, so we created an index on them.

Table: **Mentor**

**Index Name**: mentor\_id\_index

**Type**: Clustered

**Column order forming index**: mentor\_id

**Justification**: In most of the queries we’re running we’re using only the mentor\_id column in the Mentor table, so we created an index on them.

Table: **Message**

**Index Name**: message\_id\_index

**Type**: Non-clustered

**Column order forming index**: sender\_id, receiver\_id

**Justification**: In most of the queries we’re running we’ll be using sender\_id, receiver\_id columns in the Message table, so we created an index on them.

Table: **Question**

**Index Name**: question\_quiz\_id\_index

**Type**: Non-clustered  
**Column order forming index**: quiz\_id, question\_id

**Justification**: In most of the queries we’re running we’ll be using question\_id, quiz\_id columns in the Question table, so we created an index on them.

Table: **Quiz**

**Index Name**: quiz\_name\_index

**Type**: Non-Clustered

**Column order forming index**: quiz\_name

**Justification**: we need this so that mentor can search for quizzes with name field.

**Index Name**: quiz\_id\_index

**Type**: Clustered

**Column order forming index**: quiz\_id

**Justification**: In most of the queries we’re running we’ll be using quiz\_id column in the Quiz table, so we created an index on it.

**Index Name**: quiz\_eid\_index

**Type**: Non-clustered

**Column order forming index**: author\_id, quiz\_id

**Justification**: This index will be used when there is join on author and quiz\_id with other tables.

Table: **Resources**

**Index Name**: resources\_hwid\_index

**Type**: Non-clustered

**Column order forming index**: homework\_id

**Justification**: In most of the queries we’re running we’ll be using homework\_id column in the Resources table, so we created an index on them.

Table: **Response**

**Index Name**: response\_id\_index

**Type**: Non-clustered

**Column order forming index**: responder\_id, quiz\_id

**Justification**: In most of the queries we’re running we’ll be using responder\_id, quiz\_id columns in the Response table, so we created an index on them.

Table: **Response\_Answer**

**Index Name**: res\_ans\_id\_index

**Type**: Non-clustered

**Column order forming index**: response\_id, question\_id

**Justification**: In most of the queries we’re running we’ll be using response\_id, question\_id columns in the Response\_Answer table, so we created an index on them.

Table: **Weekly\_Report**

**Indexes**: week\_report\_id\_index

**Type**: Non-clustered

**Column order forming index**: mentor\_id, apprentice\_id

**Justification**: In most of the queries we’re running we’ll be using mentor\_id, apprentice\_id columns in the Weekly\_Report table, so we created an index on them.

**STORED PROCEDURES & VIEWS:**

**VIEWS:**

**View**: EMPLOYEE\_INFO

**Goal**: View consists of basic details of all employees like their id, username, full name, job title, hire date and their address details. It should be accessible to all database users as it consists of basic details about any employee and they cannot update or insert records using this view.

**View**: MENTOR\_PUPILS

**Goal**: View consists of id’s and names of apprentice and their mentors. Using this view we can find if given apprentice is assigned to any mentor or not. It should be accessible only to Administrator and he’ll be creating relationships and needs this info to do his work.

**View**: APPRENTICE\_INFO

**Goal**: View shows the data about apprentice respective info which can be used by management/admin.

**View**: MENTOR\_INFO

**Goal**: View shows the data about meetings respective info which can be used by management/admin.

**View**: ADMIN\_INFO

**Goal**: View shows the data about meetings respective info which can be used by management/admin.

**View**: ALL\_HOMEWORKS

**Goal**: View shows the Homeworks posted and their status respective with publishing and completion status by student.

**View:** ALL\_REPORTS

**Goal**: View shows all the reports along with names of employees involved, score and description. This can be used to see the performance of apprentices.

**TRIGGERS:**

**Trigger**: `MMS\_Sprint3`.`Appointment\_BEFORE\_UPDATE` BEFORE UPDATE ON `Appointment`

**Goal**: This trigger changes the status of attended to true when status of the meeting is marked as done. Update should be done by Mentor as he’ll be using view ``

**Trigger**: `MMS\_Sprint3`.`Appointment\_BEFORE\_INSERT` BEFORE INSERT ON `Appointment`

**Goal**: This trigger insert a row with proper status and attended value before inserting,even though wrong values were given in status and attended while inserting.

**PROCEDURES:**

**Stored procedure**: MAKING\_LOGINS\_INACTIVE

**Goal**: This stored procedure captures all(mentor, apprentice, administrator) accounts or their employee ids where mentor, apprentice & administrator didn’t login for last 90 days.

**Example**: CALL MAKING\_LOGINS\_INACTIVE();

**Stored procedure:** GET\_QUIZ

**Goal**: This stored procedure allows application to get quiz related data like questions and answer choices concatenated when quiz\_id and is\_active values are provided.  
**Example**: CALL GET\_QUIZ(1,1);

**Stored procedure:** LOGIN\_TIME\_UPDATE

**Goal**: This stored procedure allows application to update login time depending on the id and table code we’ve provided during the call.

**Example**: CALL LOGIN\_TIME\_UPDATE(1,'M');

**Stored procedure:** LOGOUT\_TIME\_UPDATE

**Goal**: This stored procedure allows application to update logout time depending on the id and table code we’ve provided during the call.

**Example**:CALL LOGOUT\_TIME\_UPDATE(1,'M');

**Stored function**: SCORE\_CALCULATION

**Goal**: This stored function takes quiz\_id, responder\_id, attempt\_no as input and returns a decimal value(5,3) which is percentage apprentice accumulated in given attempt and quiz.

**Example:**

SELECT SCORE\_CALCULATION(1, 5, 0) AS Score;

SELECT SCORE\_CALCULATION(1, 4, 0) AS Score;

**IMPLEMENT & DEPLOY DATABASE:**

Database Name: MMS\_Sprint3

**Sprint3 - Key SQL Queries for Demonstration**

**Queries:**

Views

-------------------------

SELECT \* FROM EMPLOYEE\_INFO;

SELECT \* FROM MENTOR\_PUPILS;

SELECT \* FROM APPRENTICE\_INFO;

SELECT \* FROM MENTOR\_INFO;

SELECT \* FROM ADMIN\_INFO;

SELECT \* FROM ALL\_HOMEWORKS;

SELECT \* FROM ALL\_REPORTS;

StoredProcedures

----------------------------

CALL MAKING\_LOGINS\_INACTIVE();

CALL GET\_QUIZ(1,1);

CALL LOGOUT\_TIME\_UPDATE(1,'M');

CALL LOGIN\_TIME\_UPDATE(1,'M');

Function

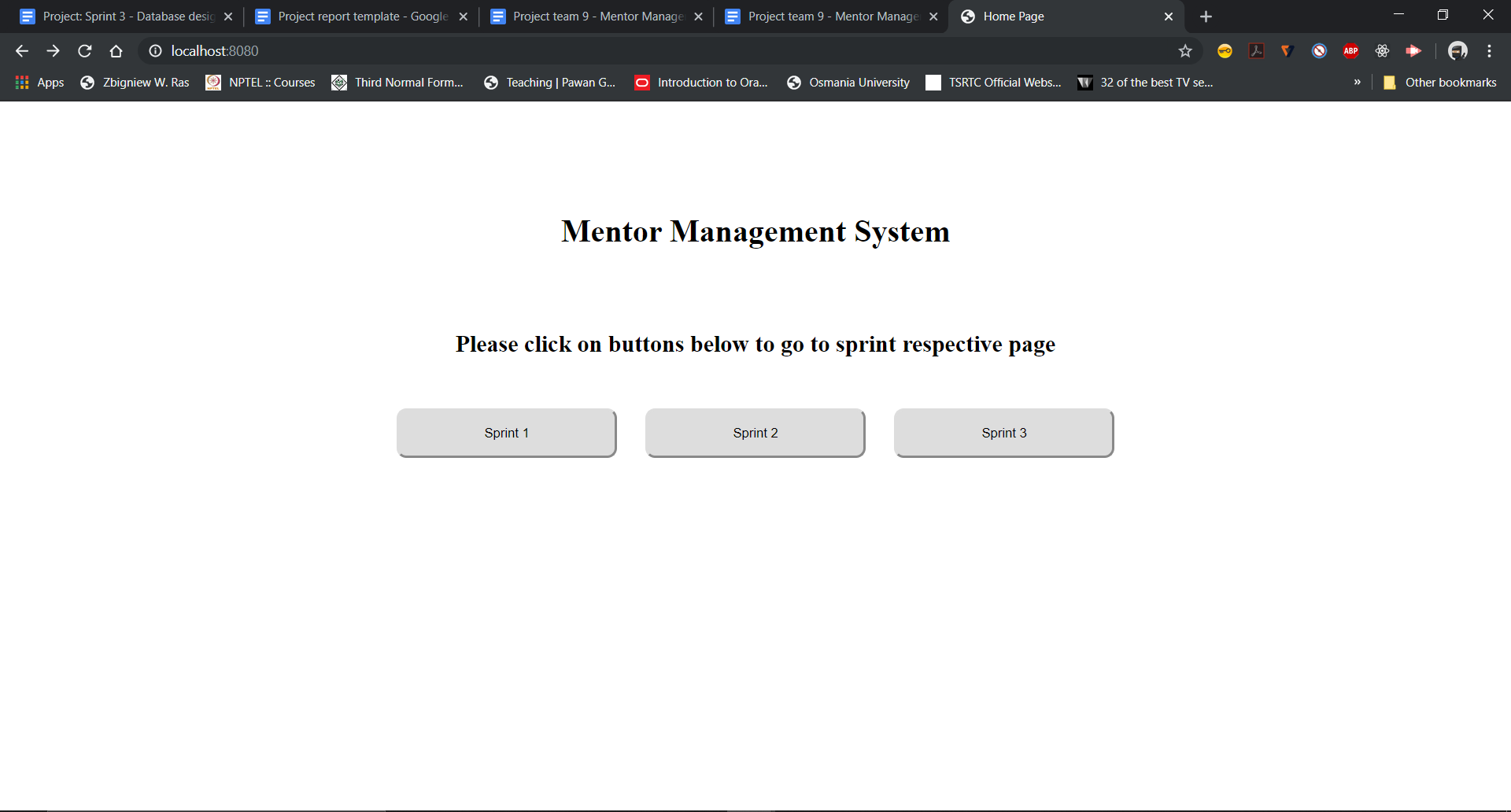
-----------------------

SELECT SCORE\_CALCULATION(1, 5, 0) AS Score;

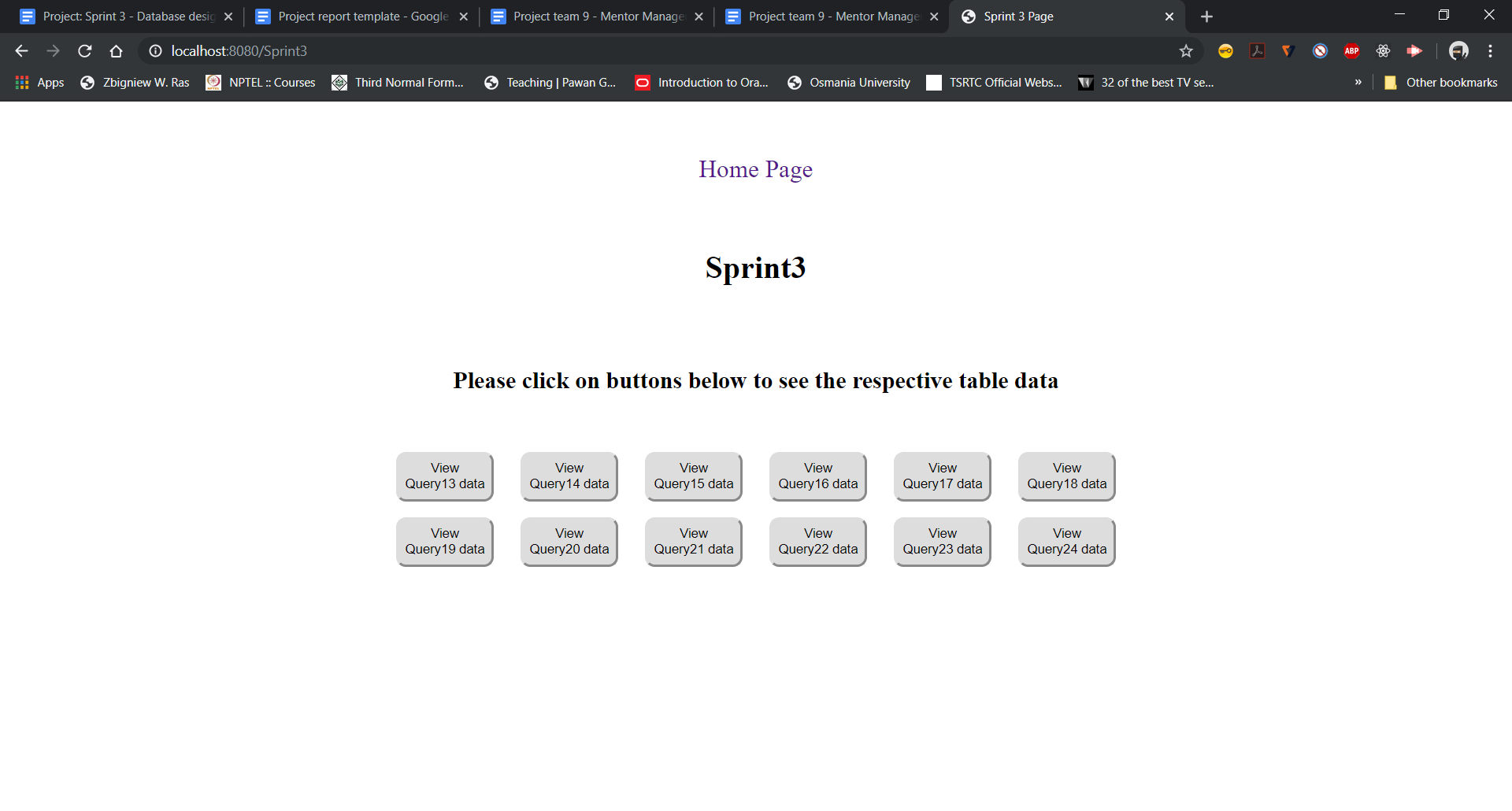
SELECT SCORE\_CALCULATION(1, 4, 0) AS Score;

**Demonstration:**

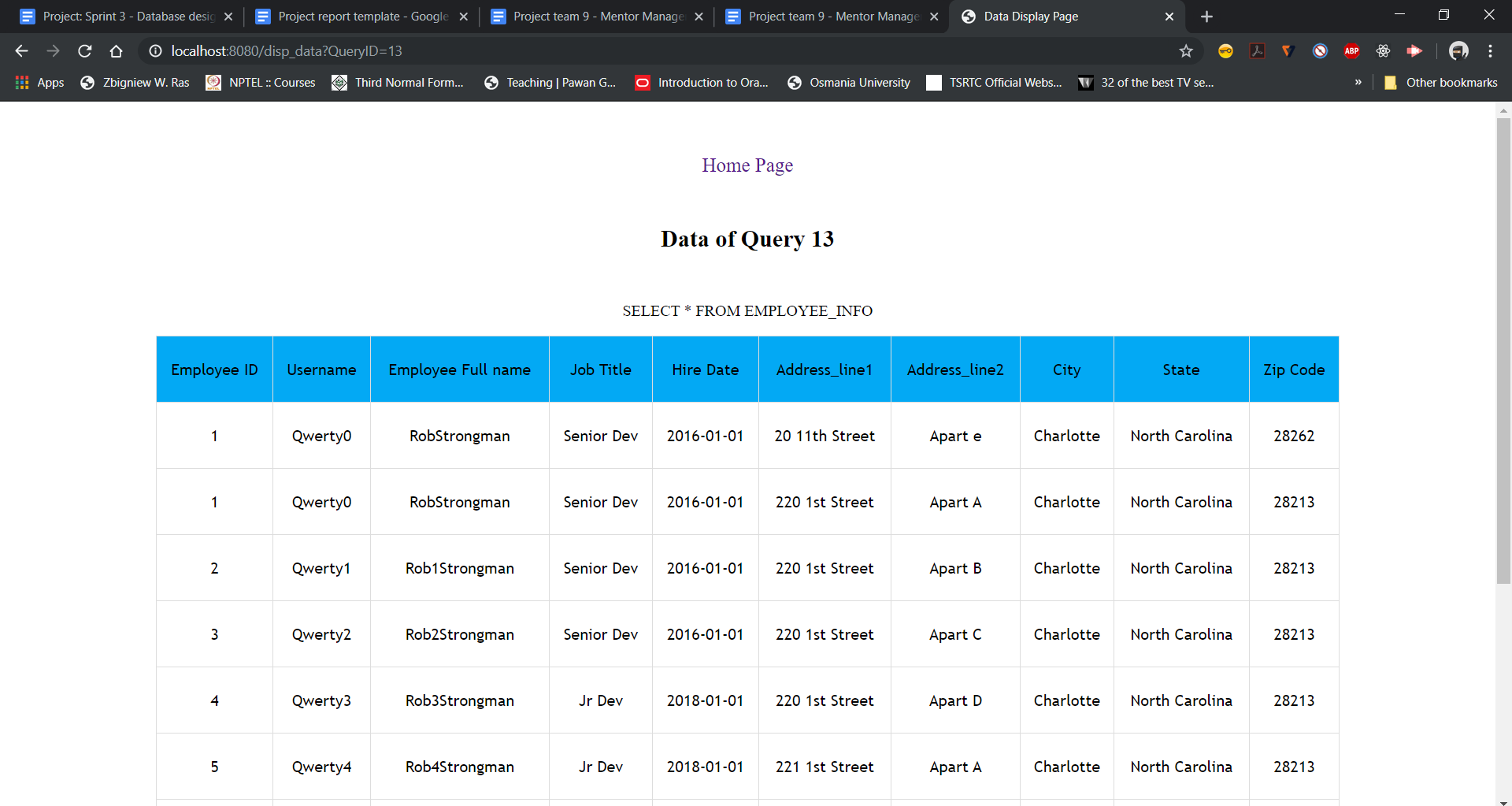
HomePage:



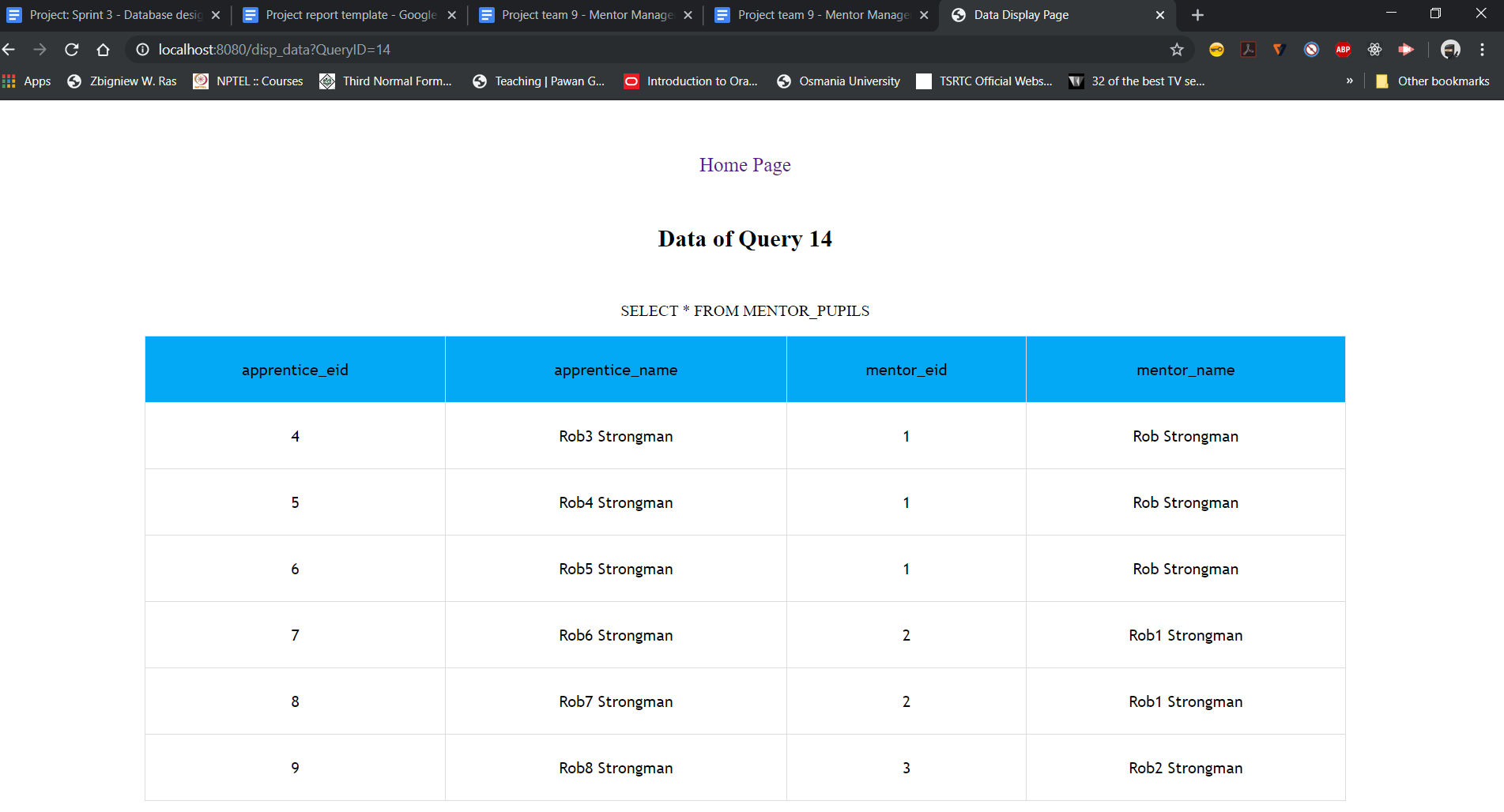
Sprint3 Page:

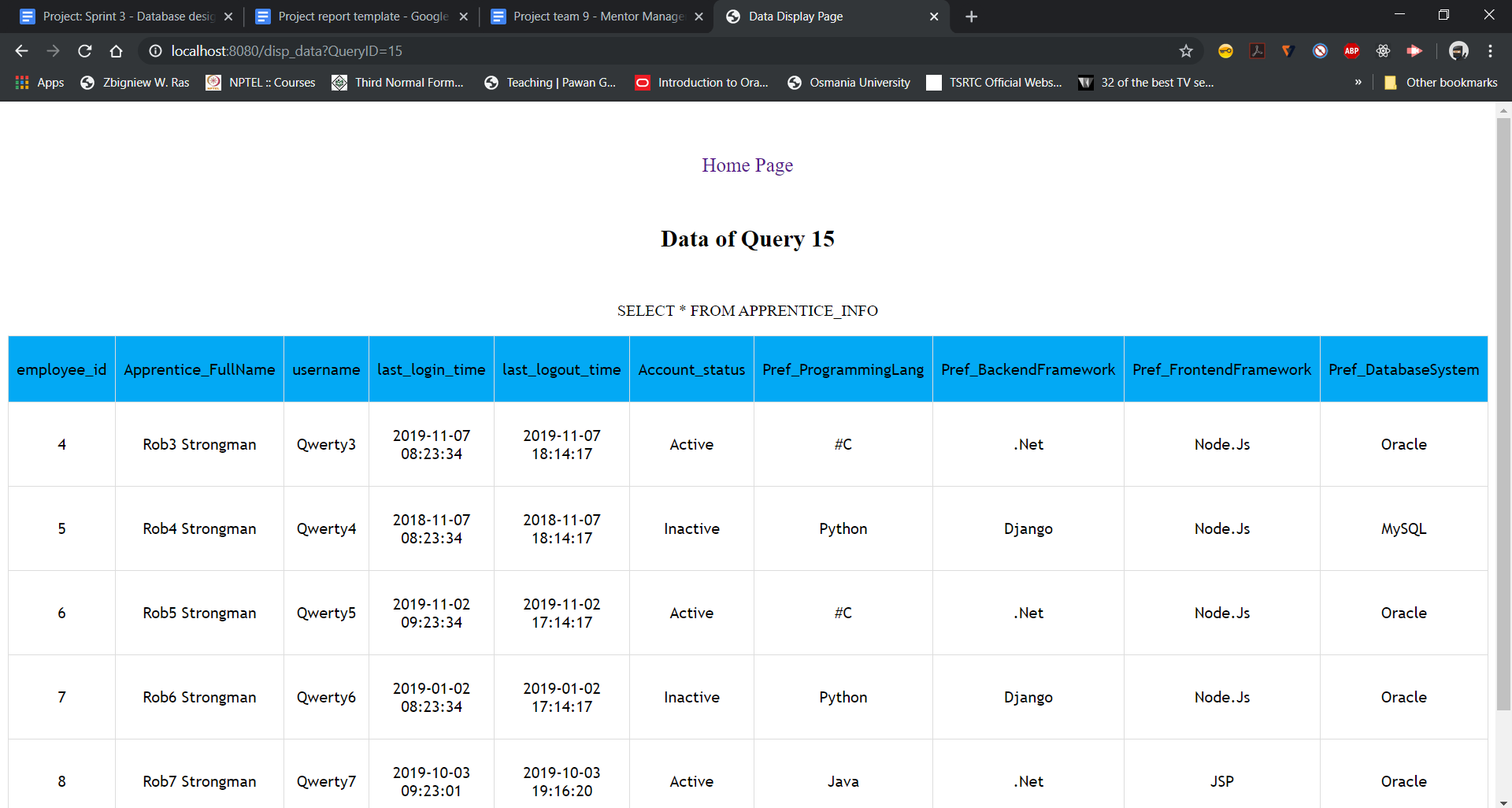


Query-13:

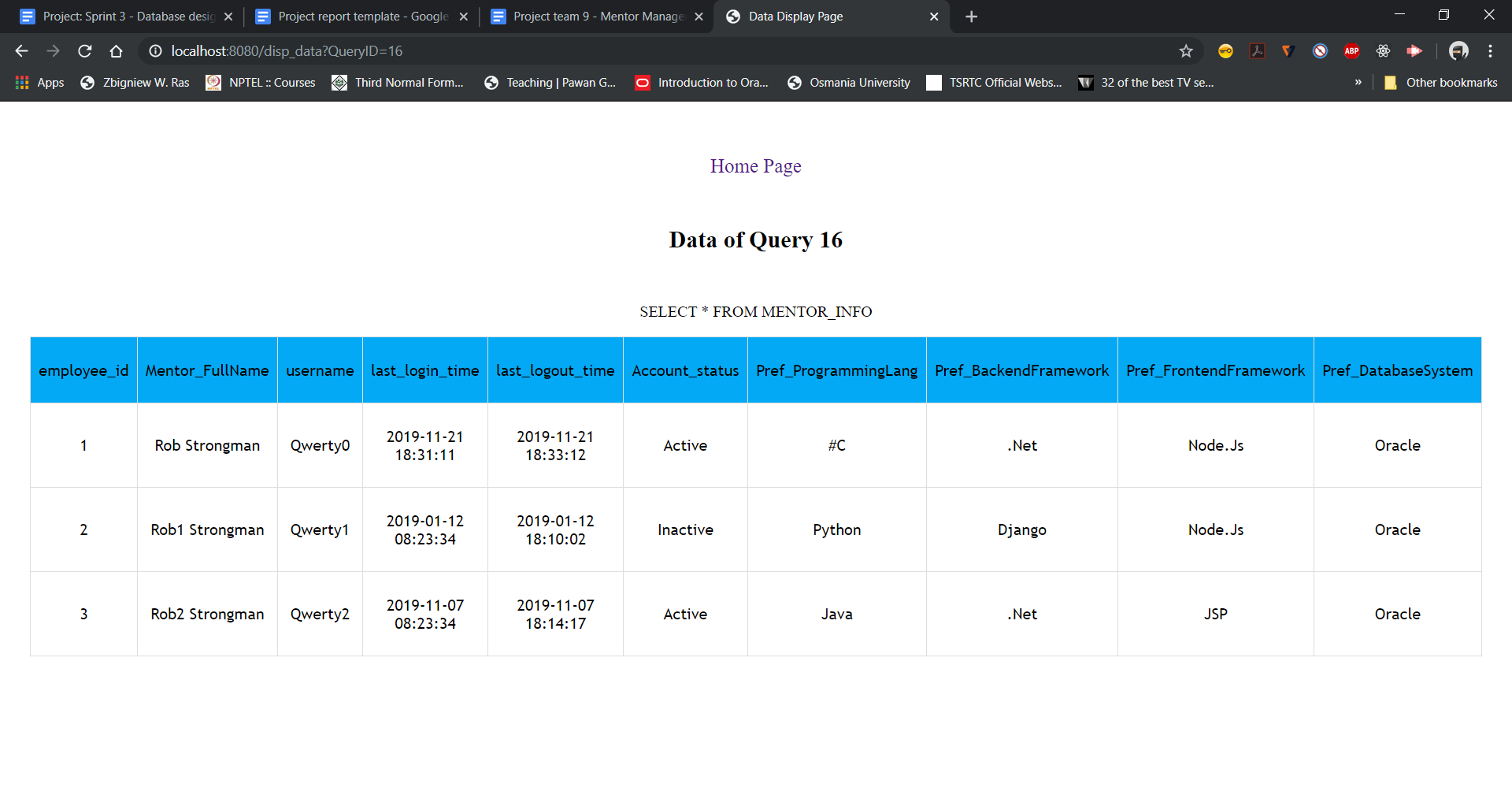


Query-14:

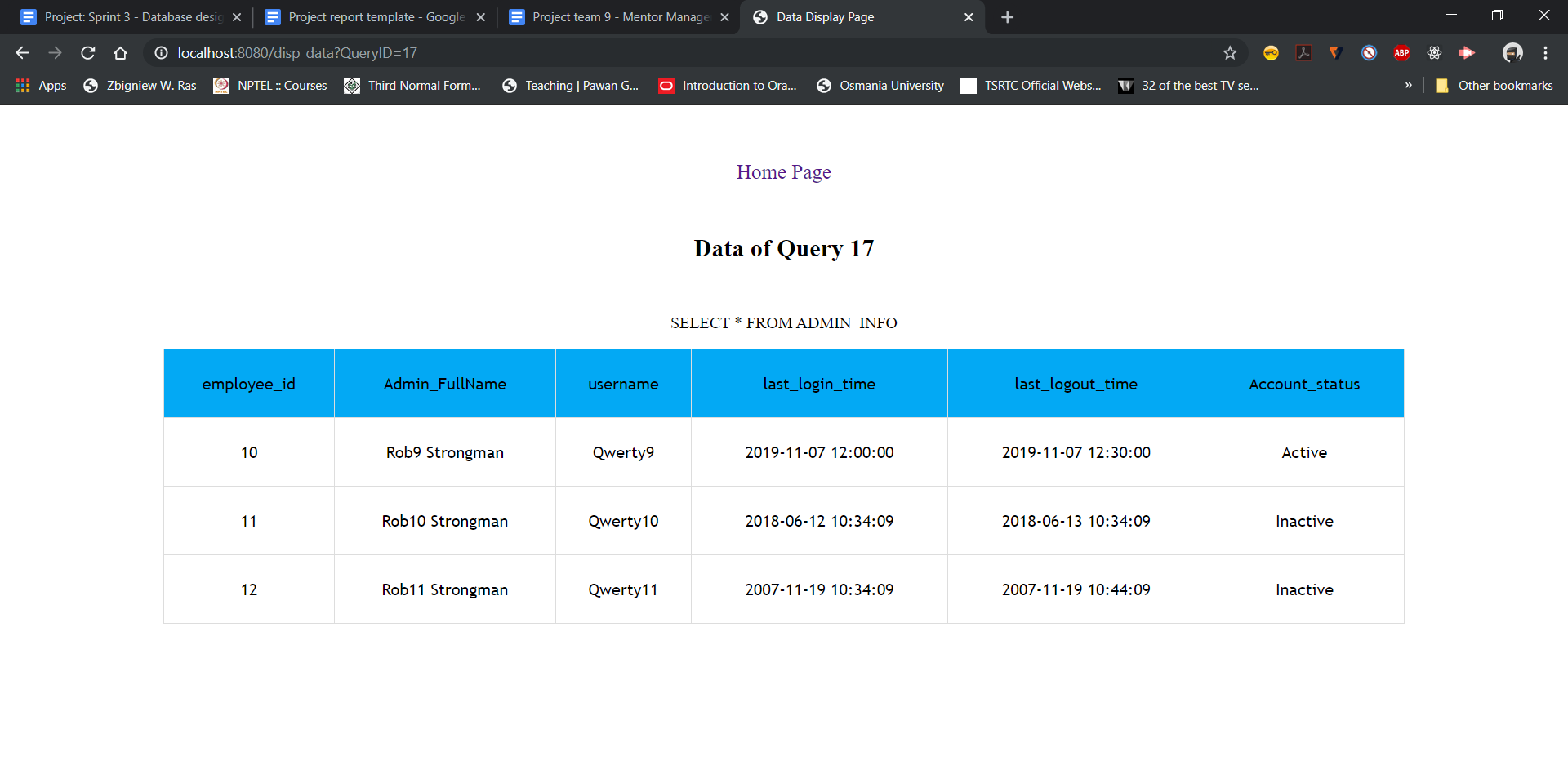


Query-15:

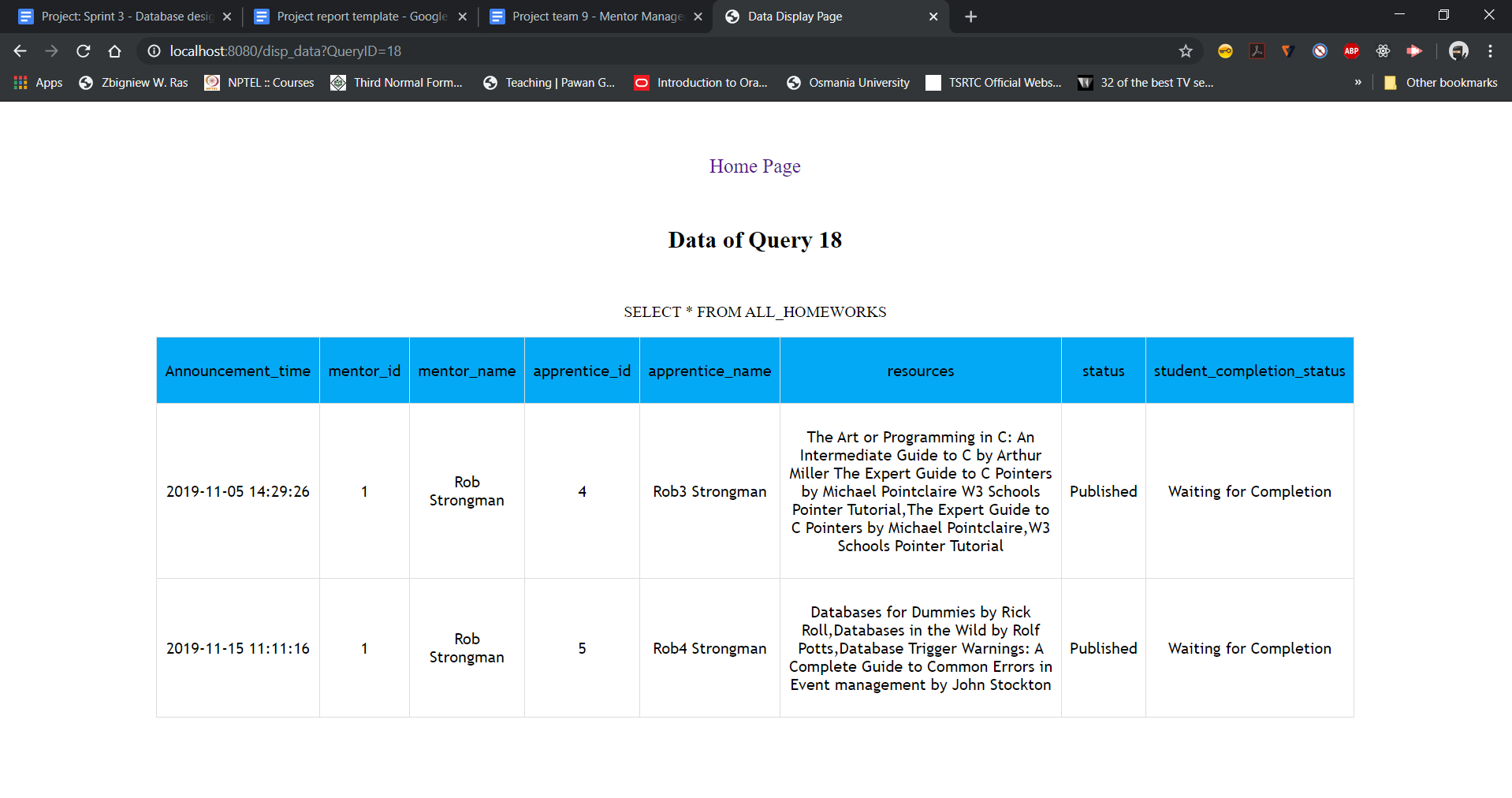
Query-16:



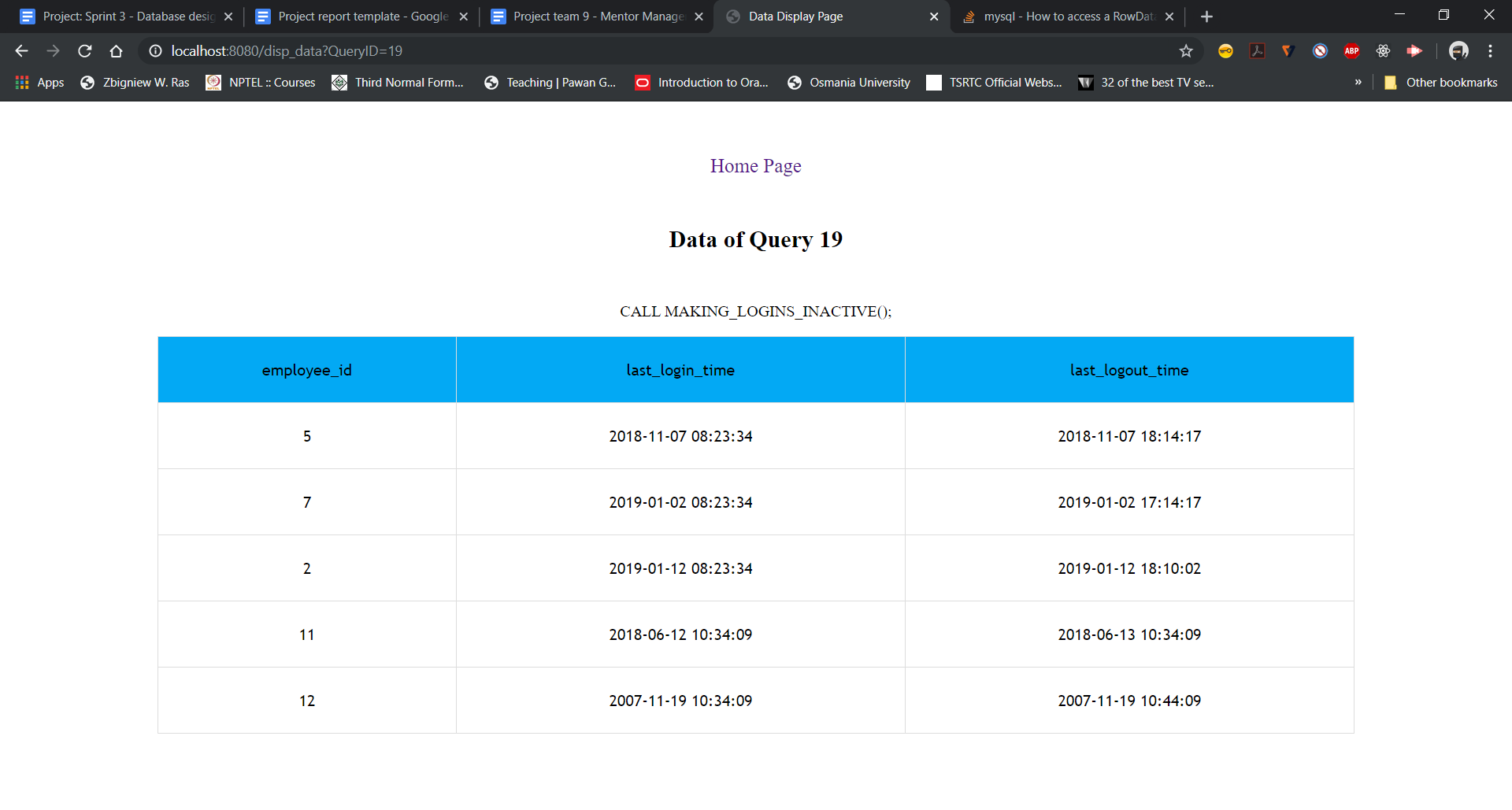
Query-17:



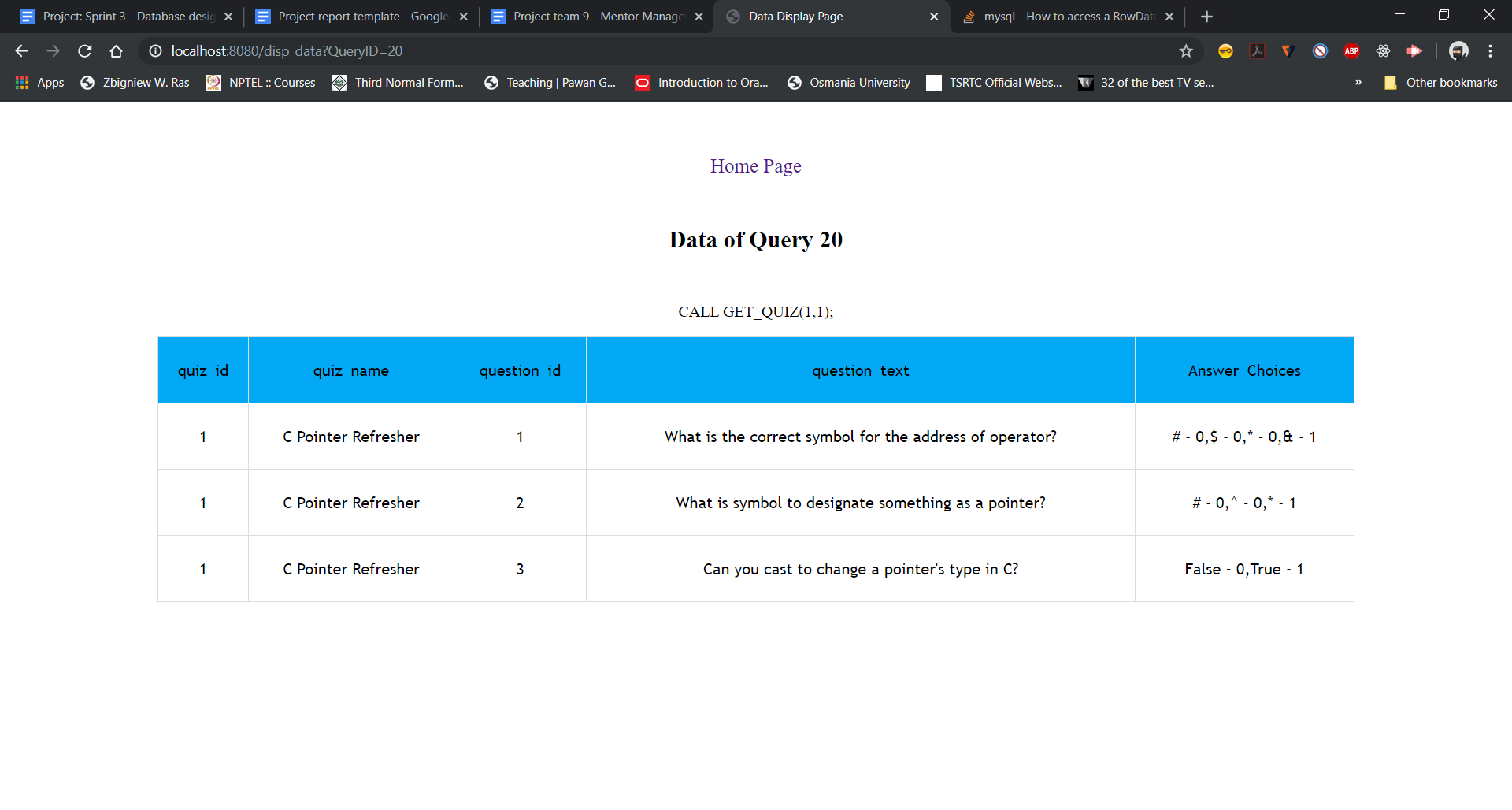
Query-18:



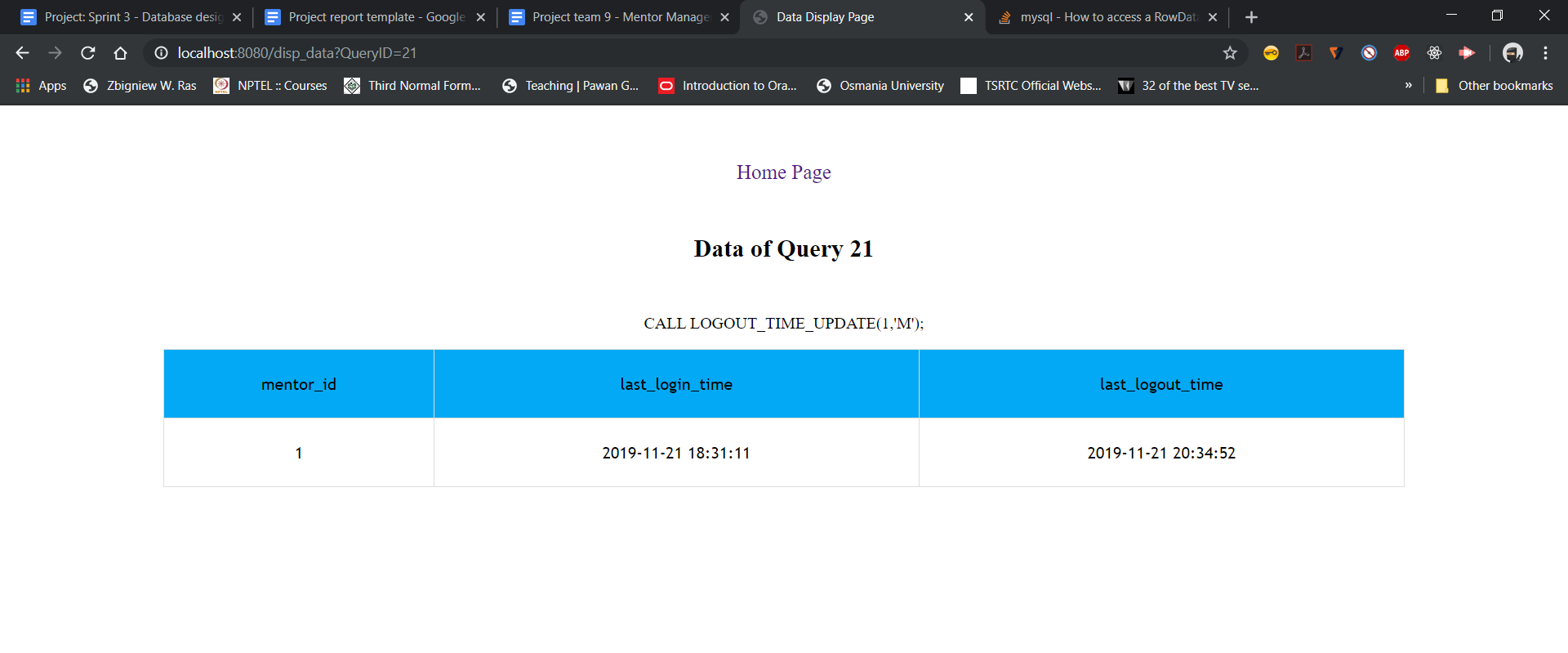
Query-19:



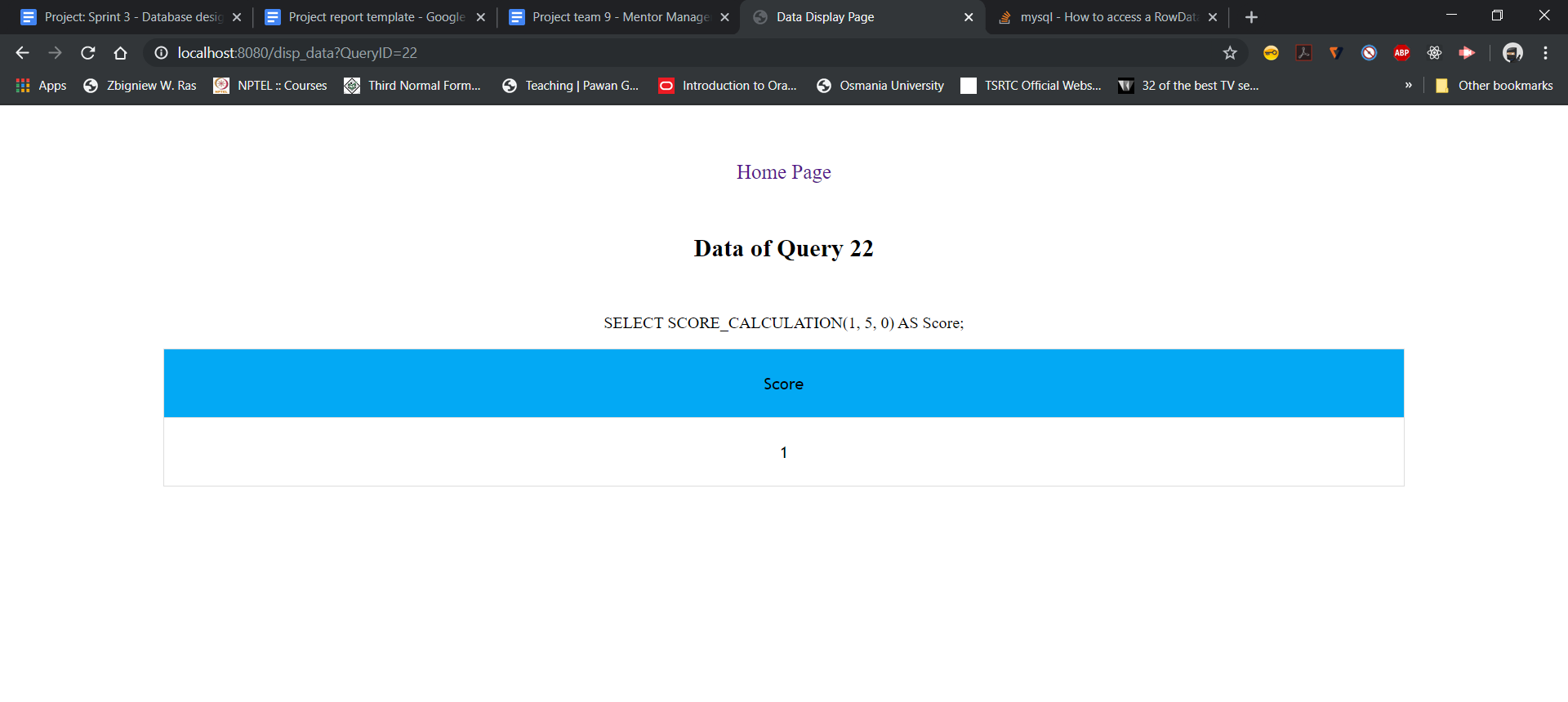
Query-20:



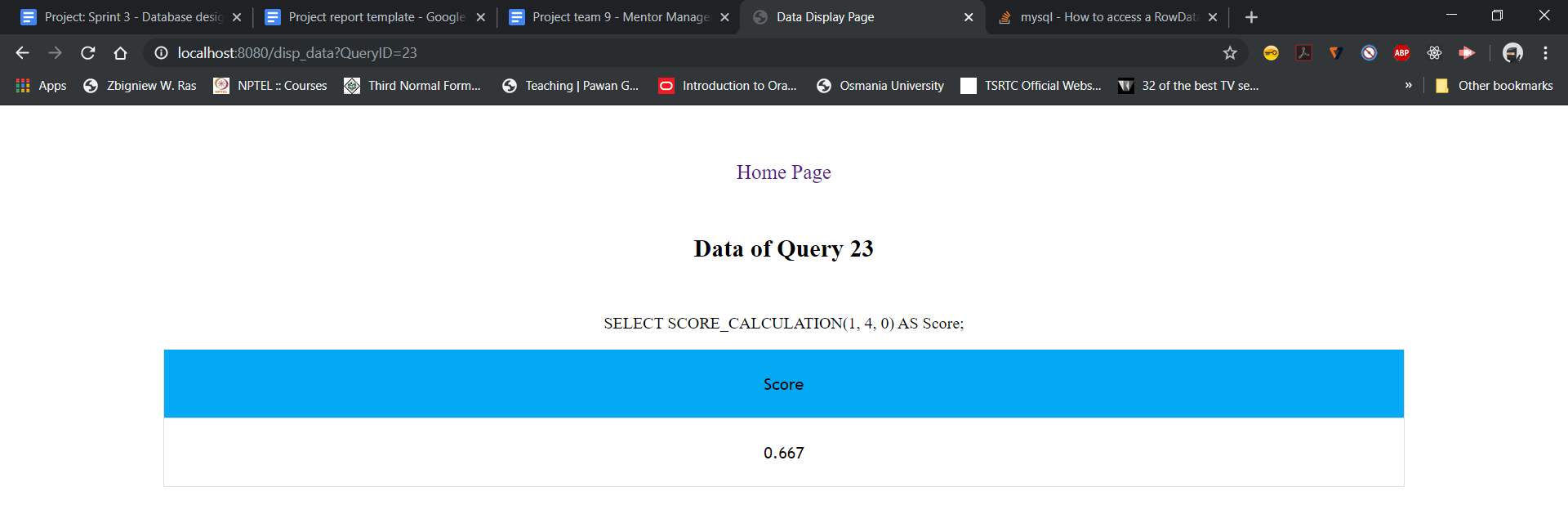
Query-21:



Query - 22:



Query-23:



Query-24:

