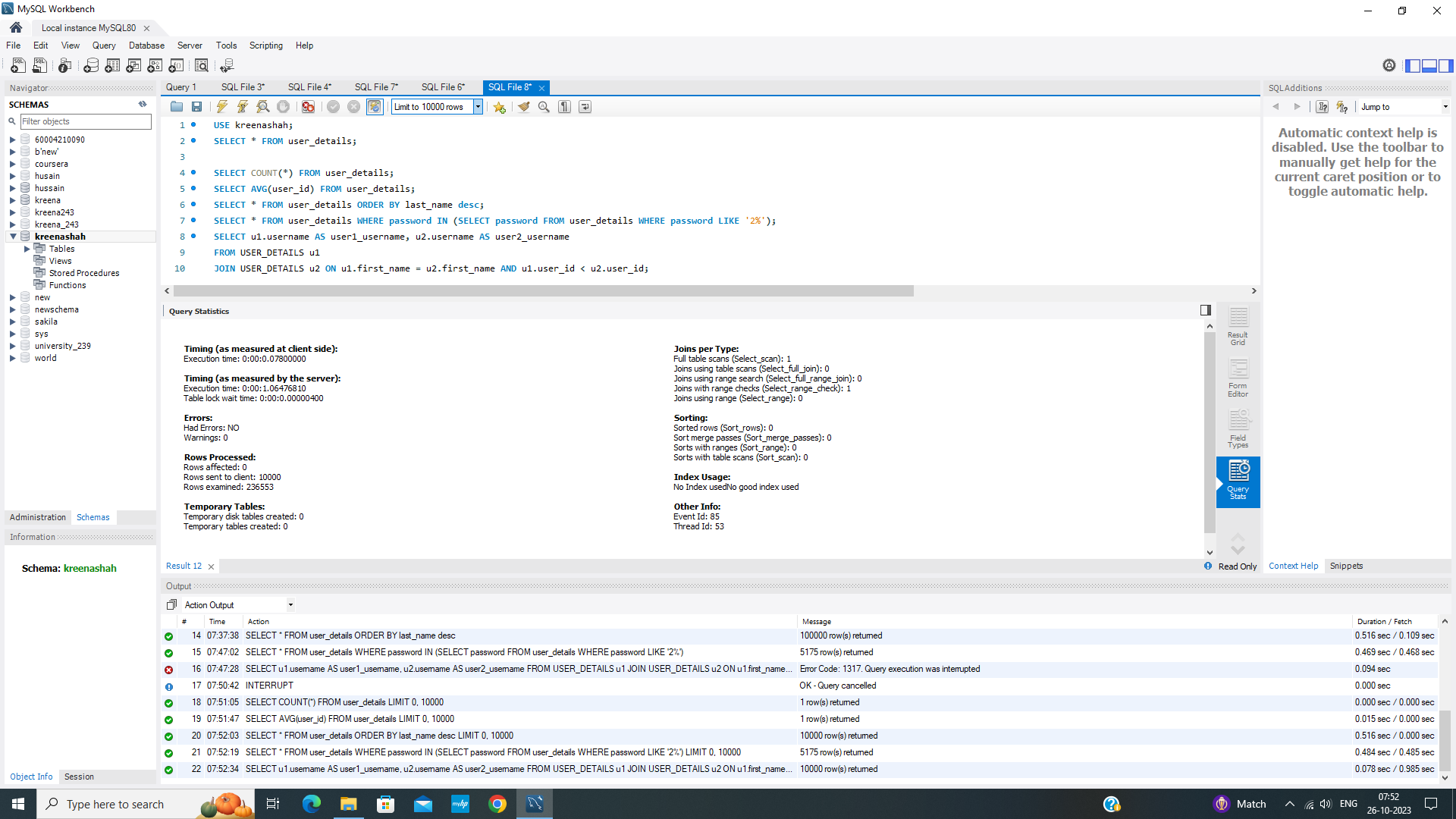
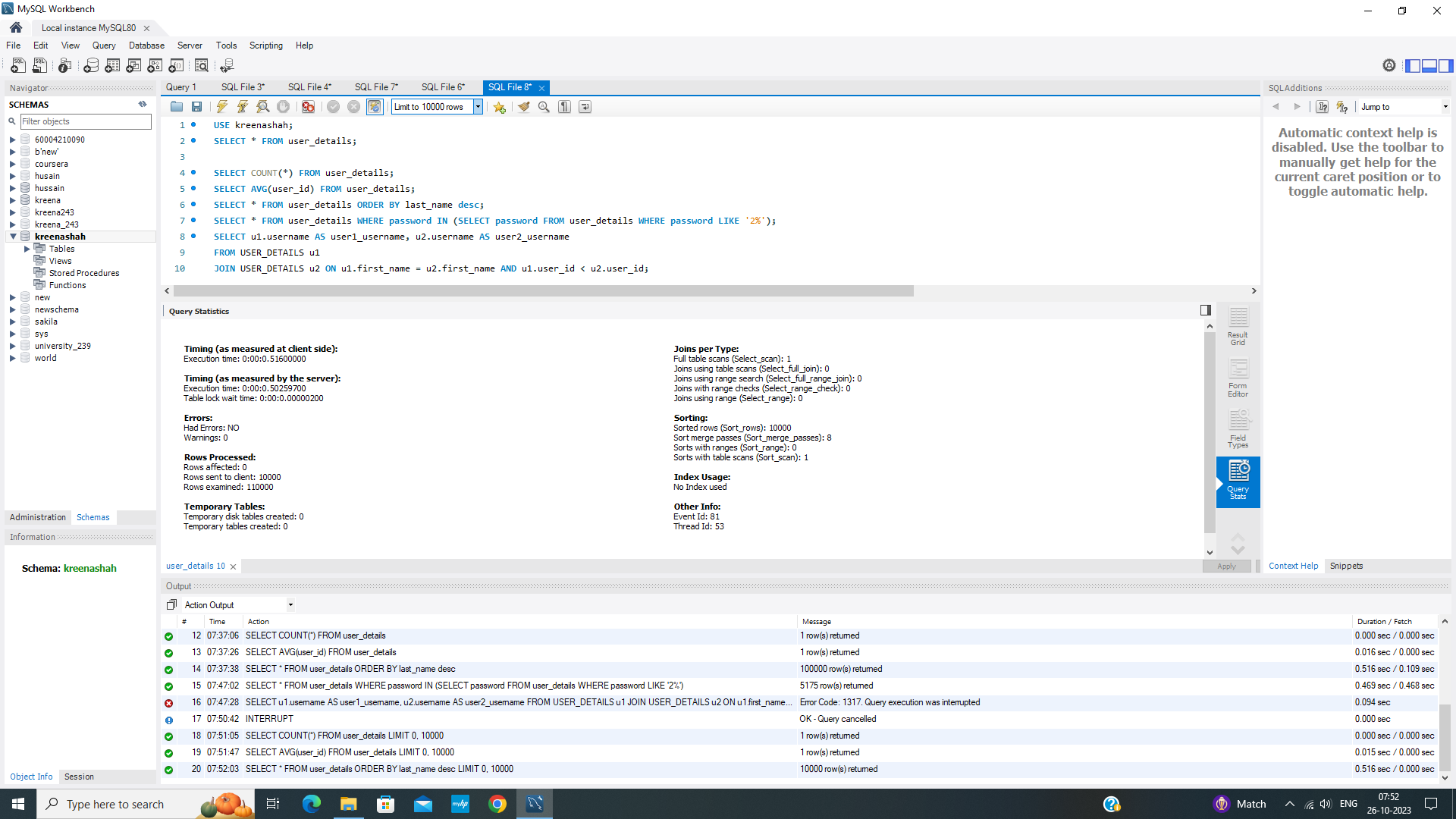
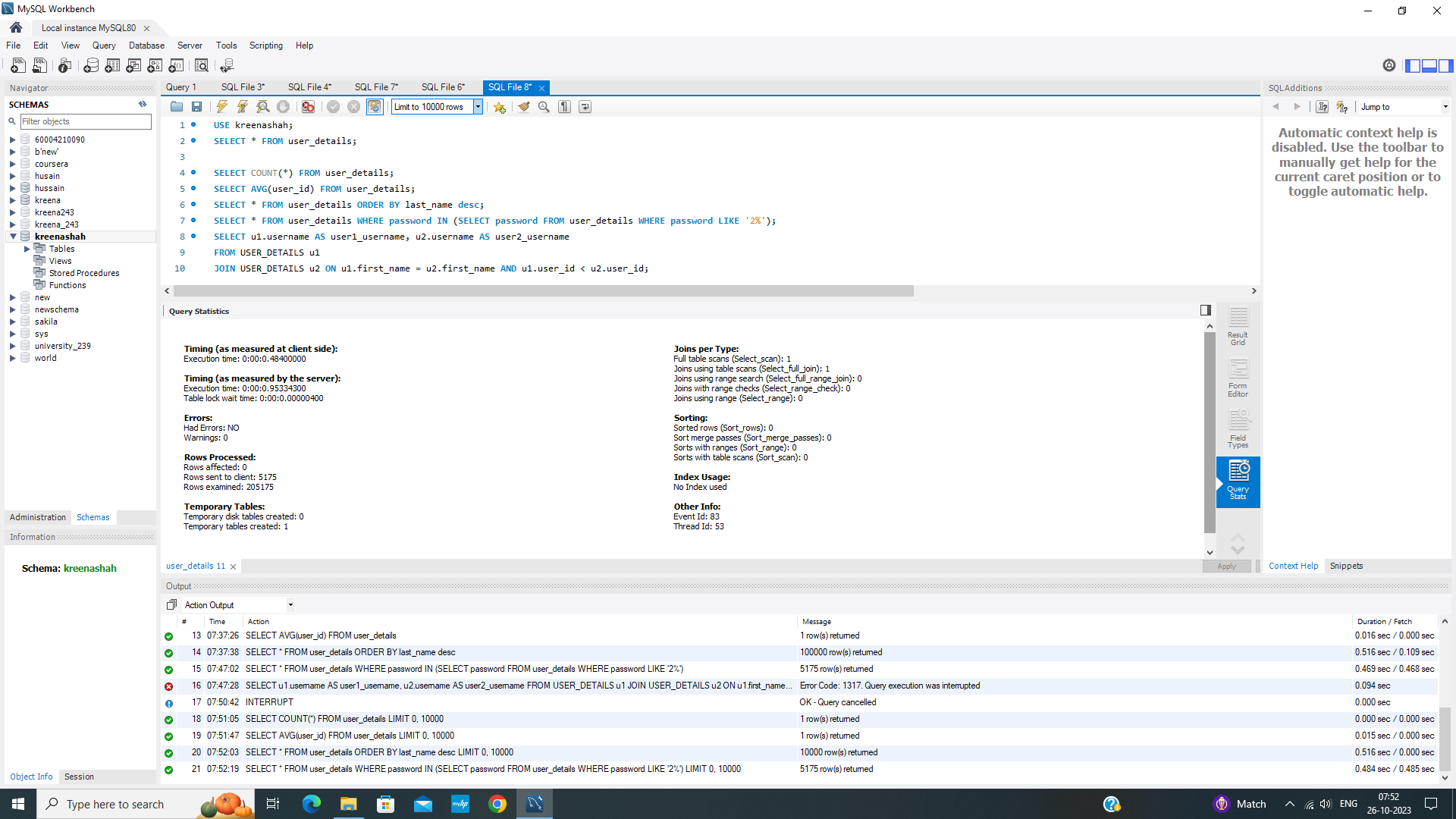
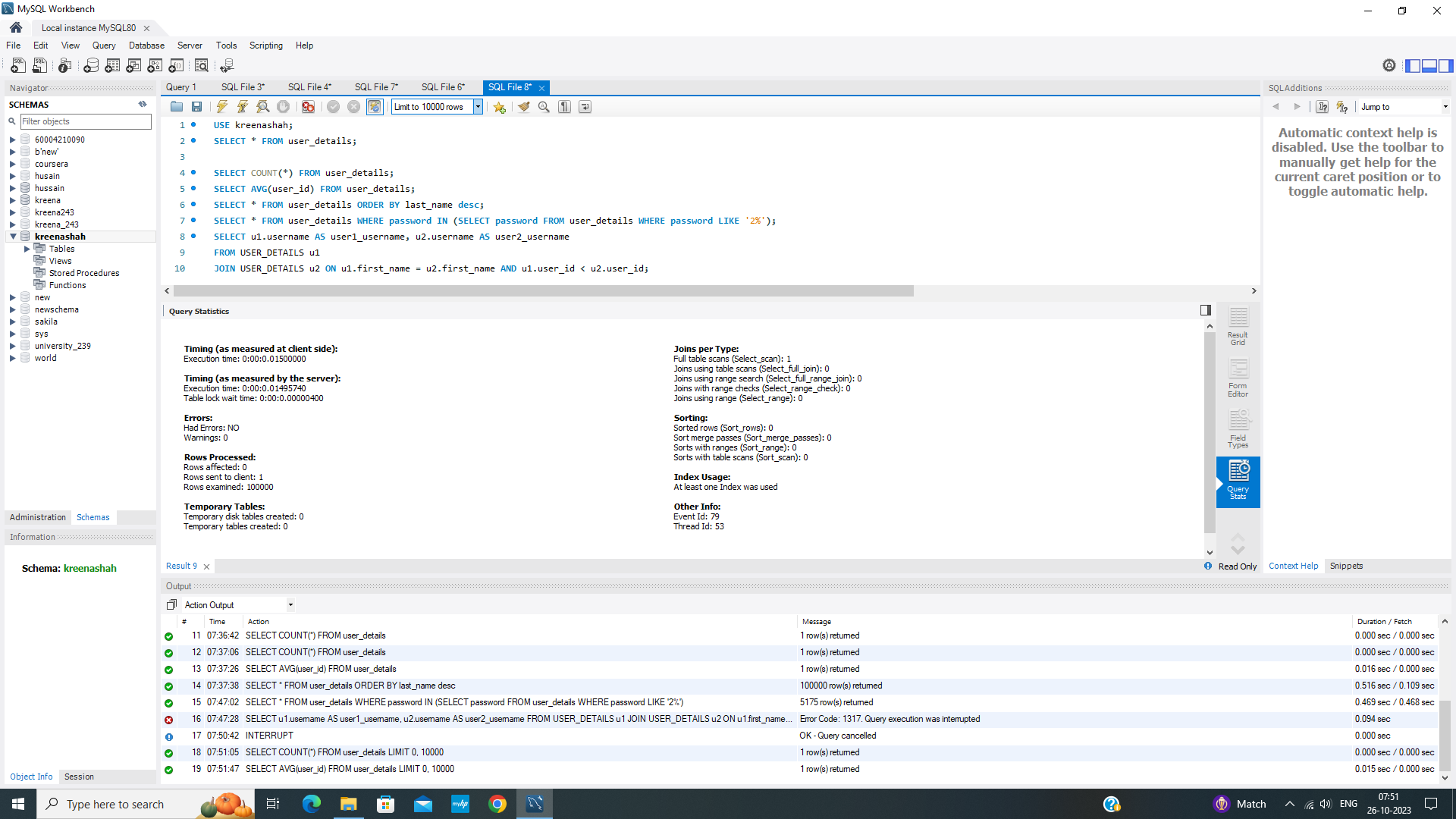
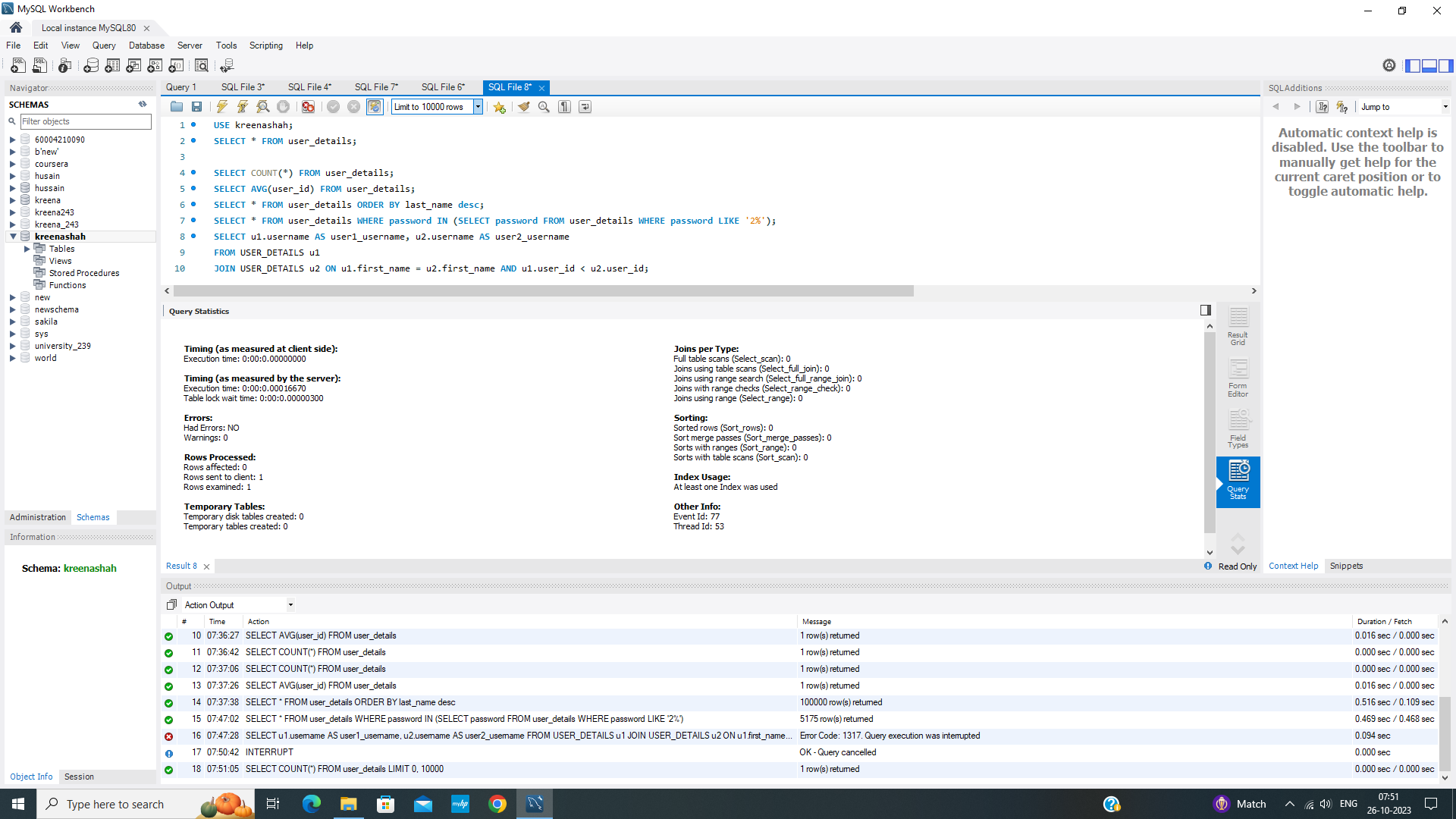
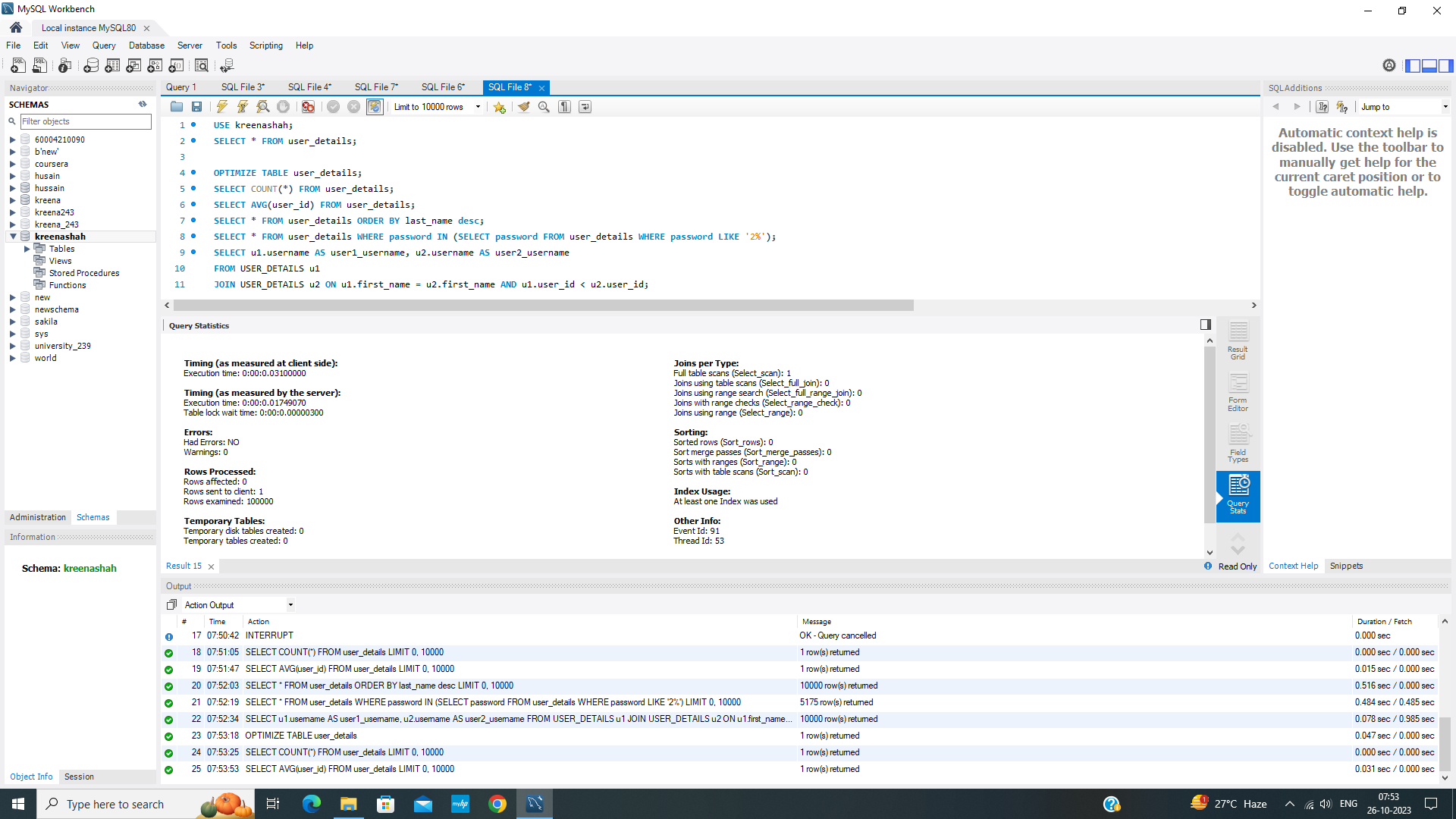
**Experiment No 2 - Query Optimization**

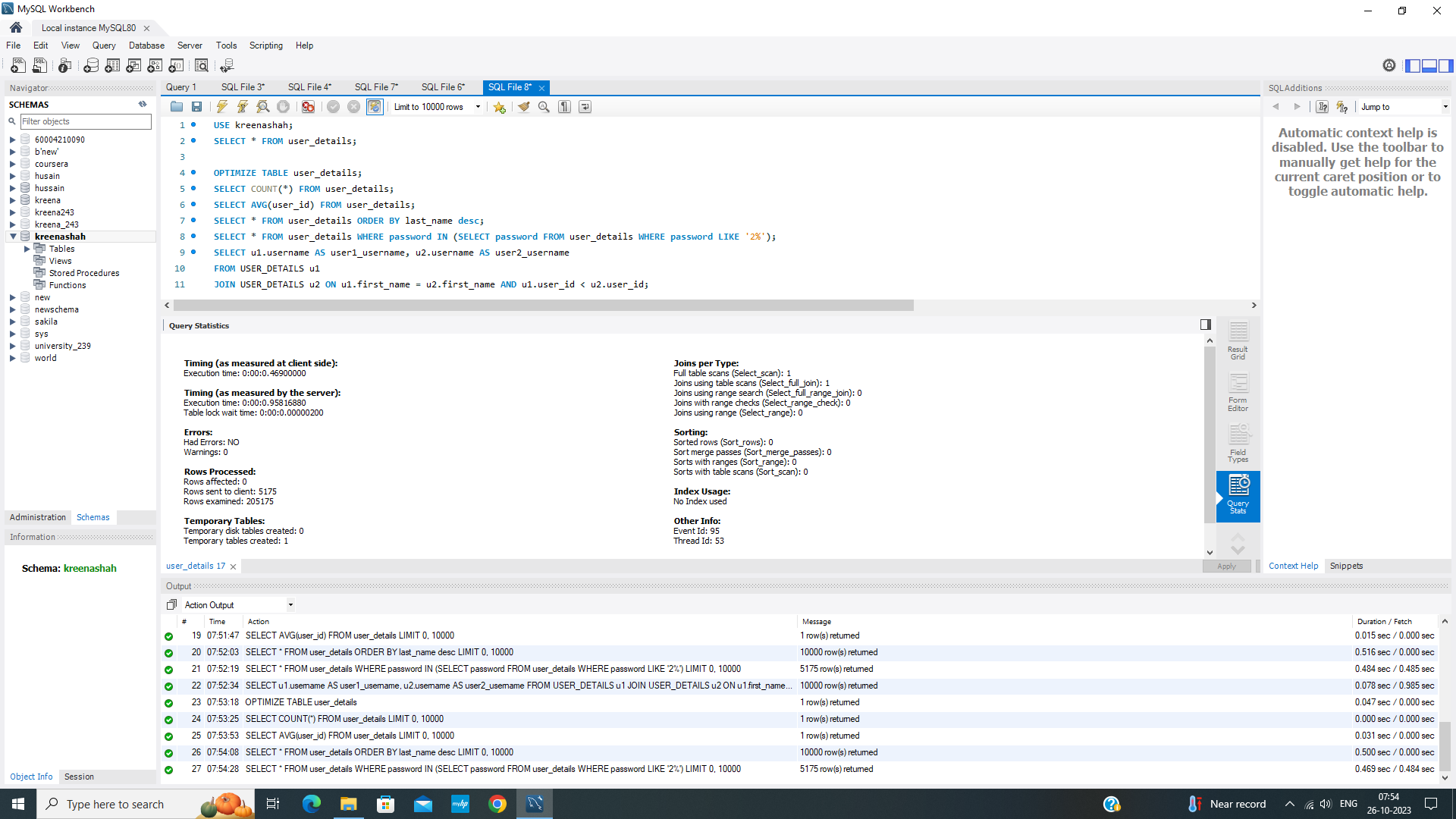
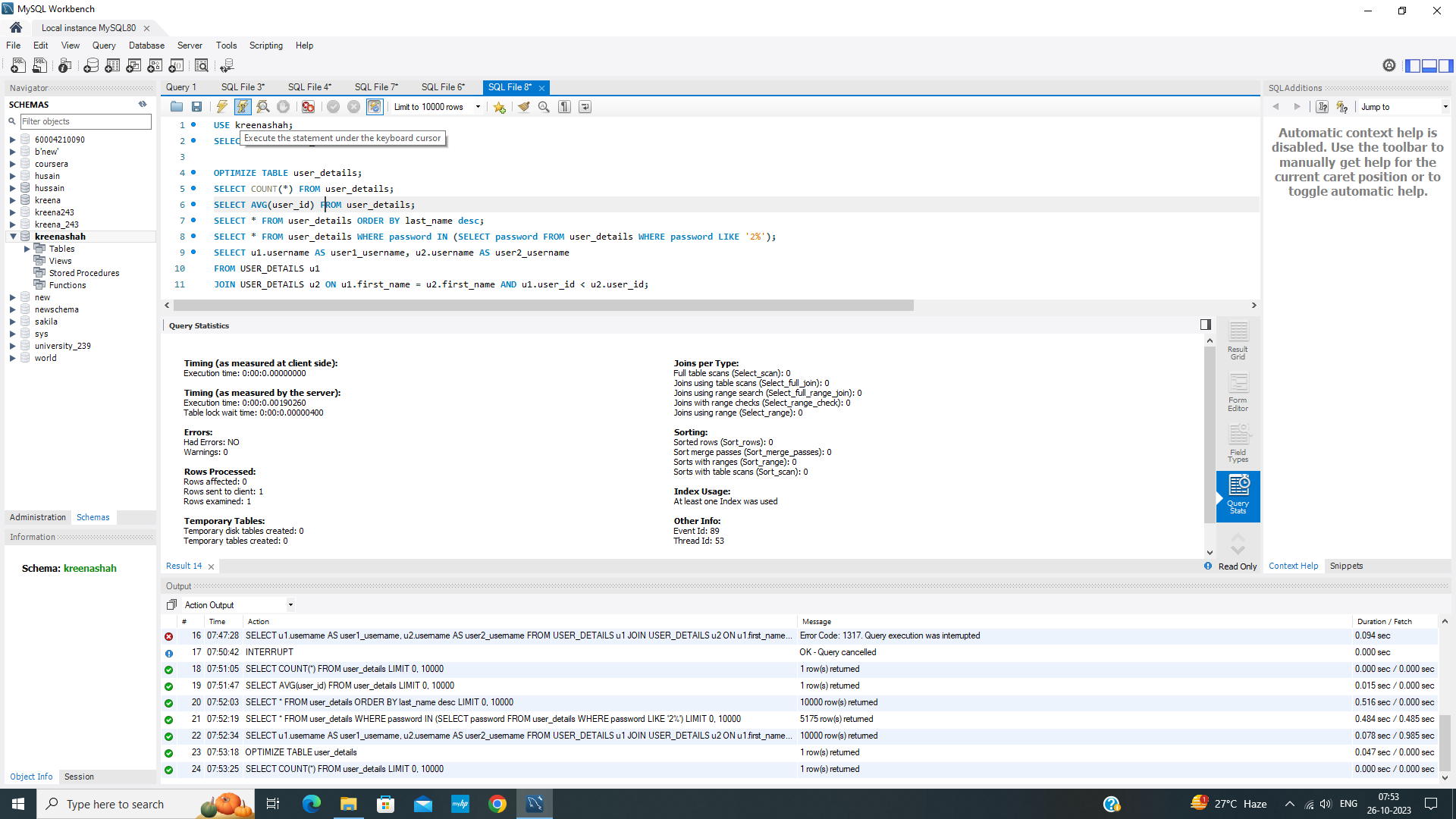
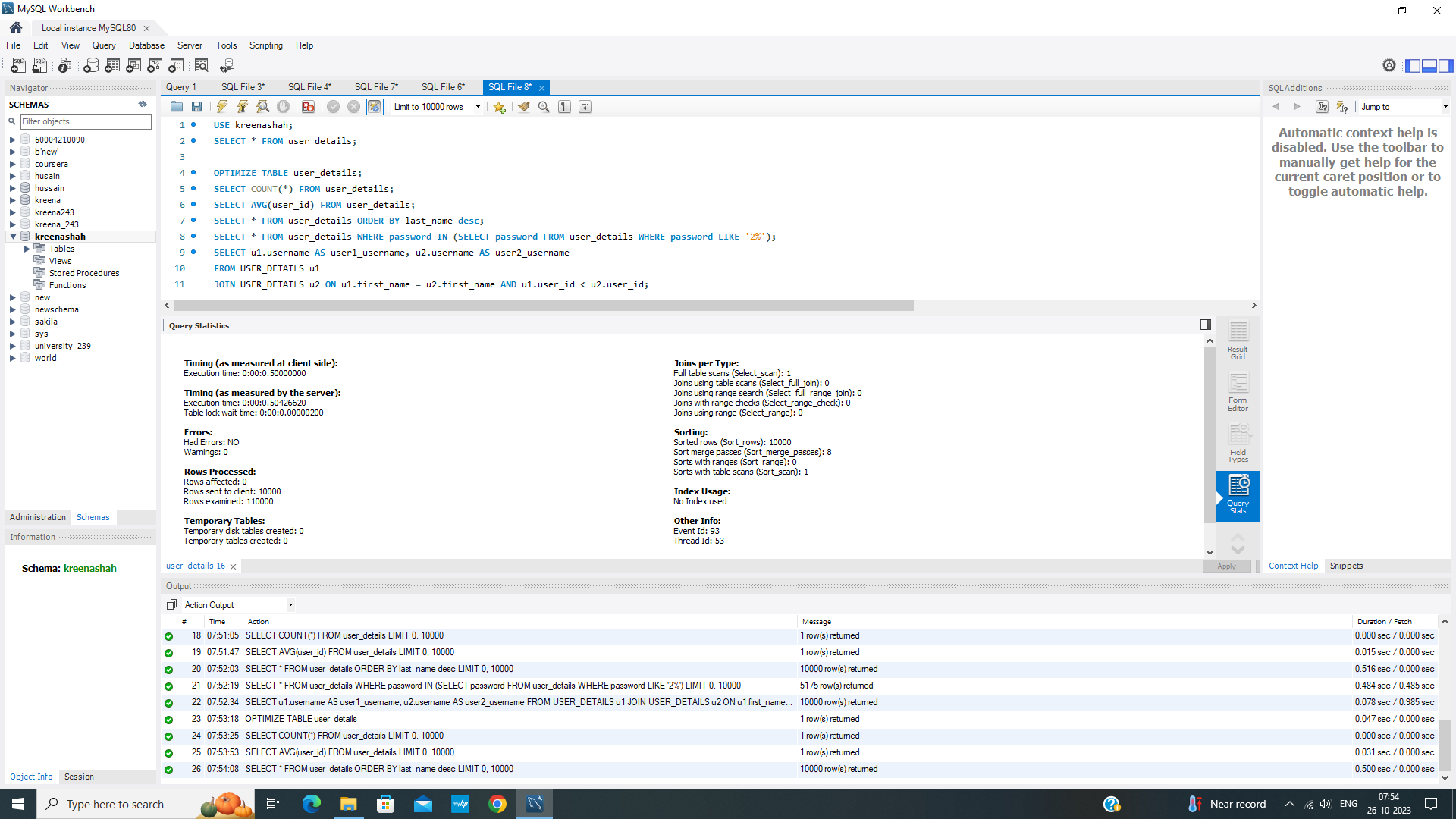
**Simple Query**

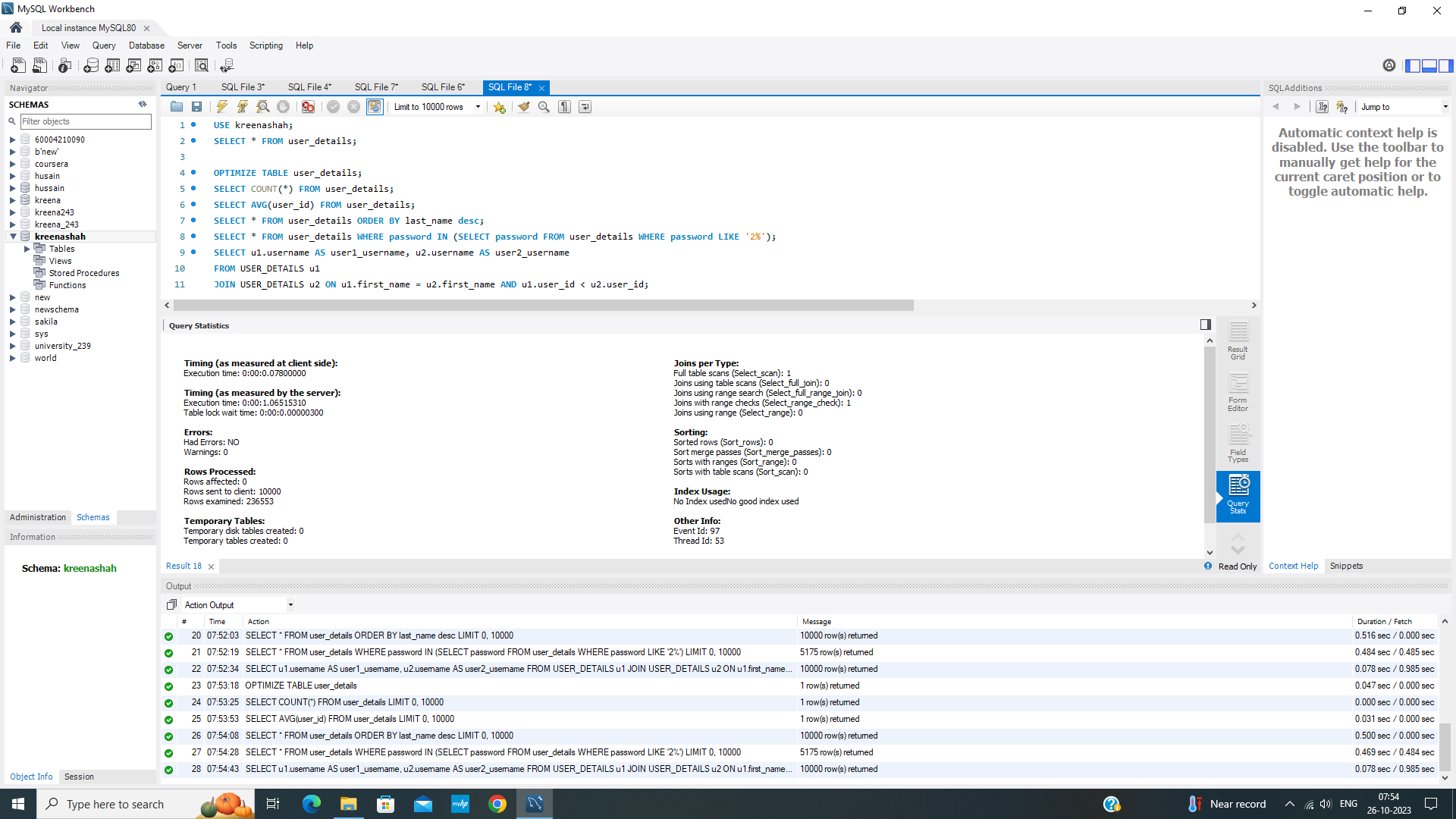




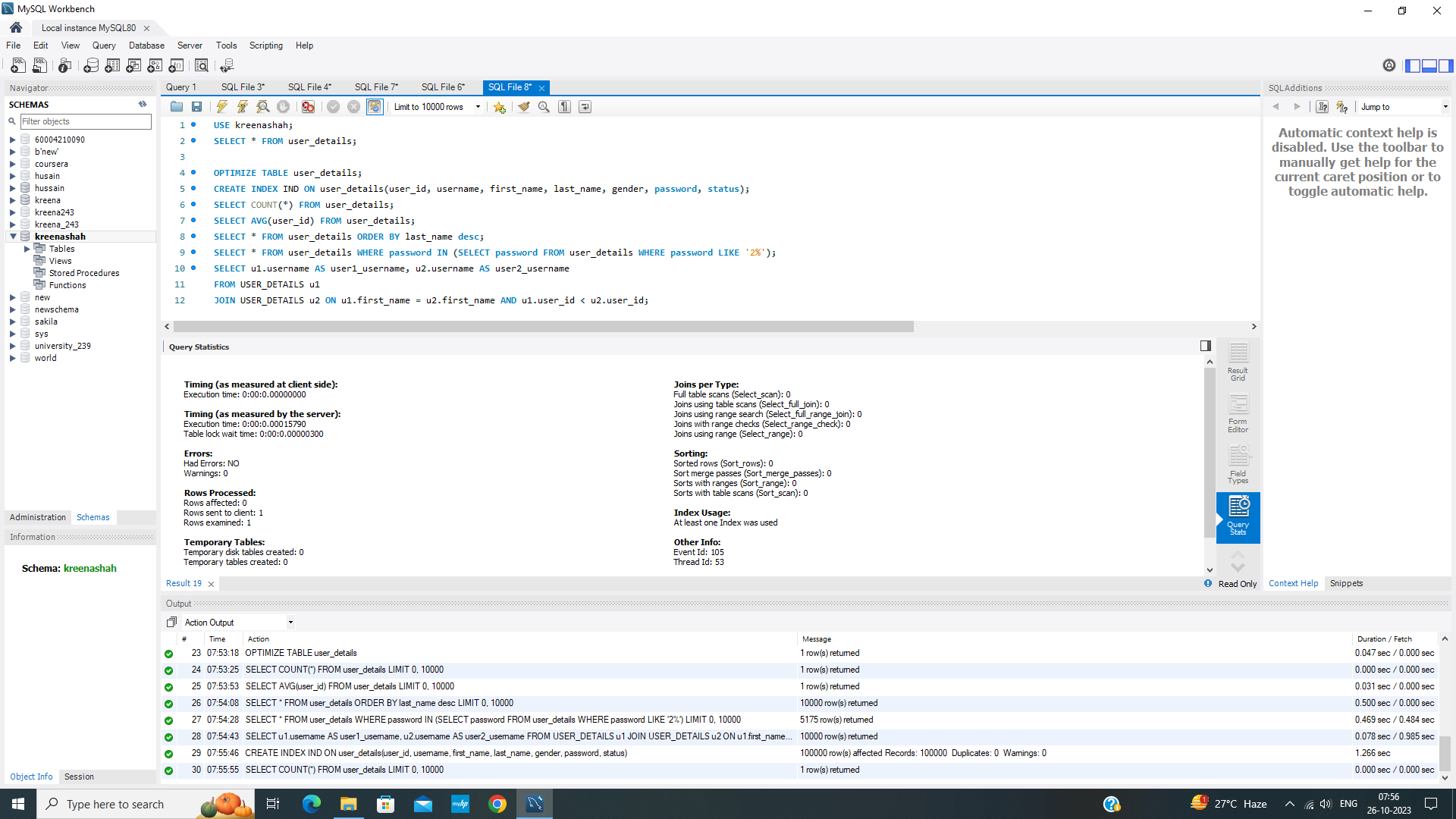
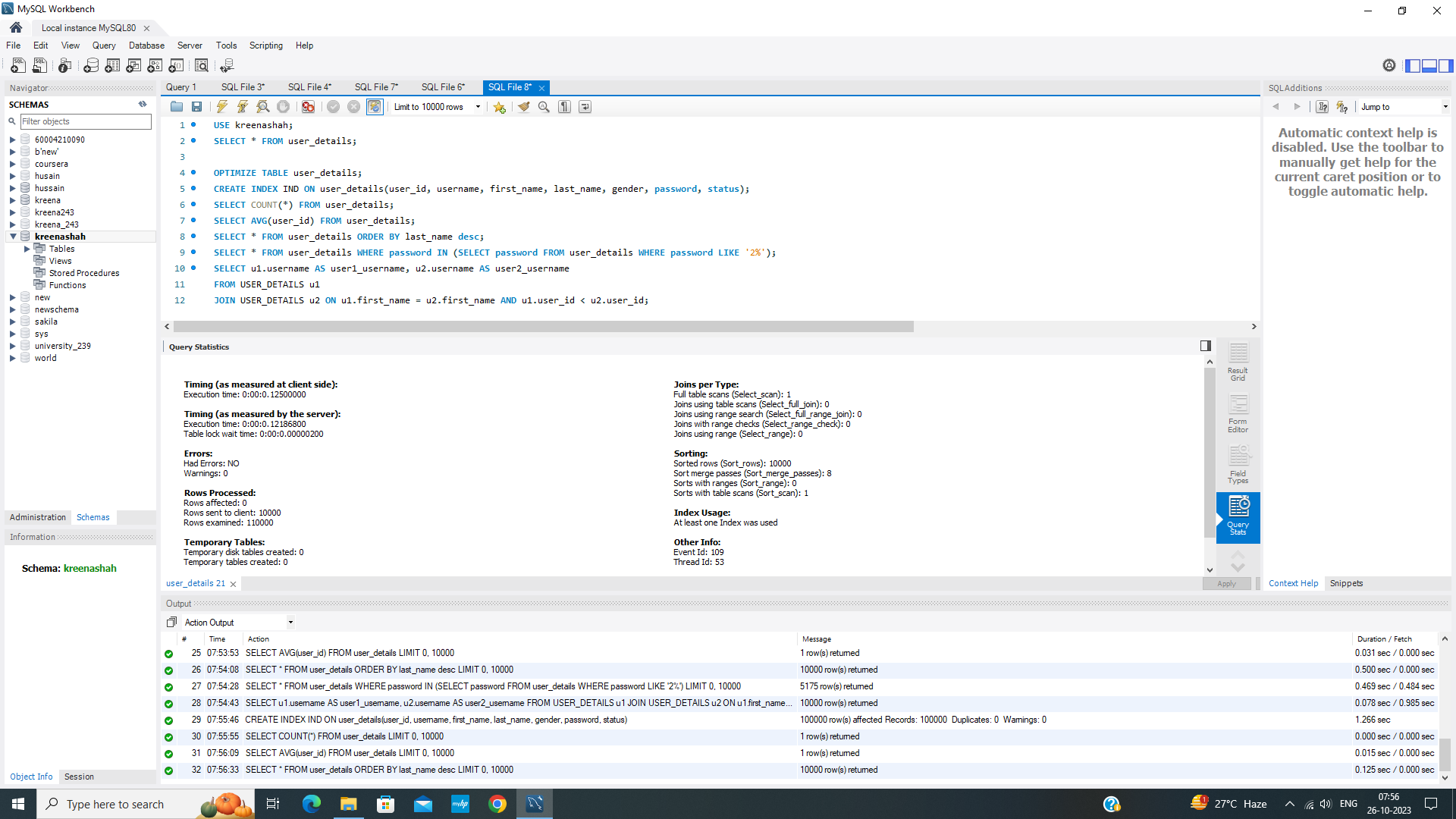
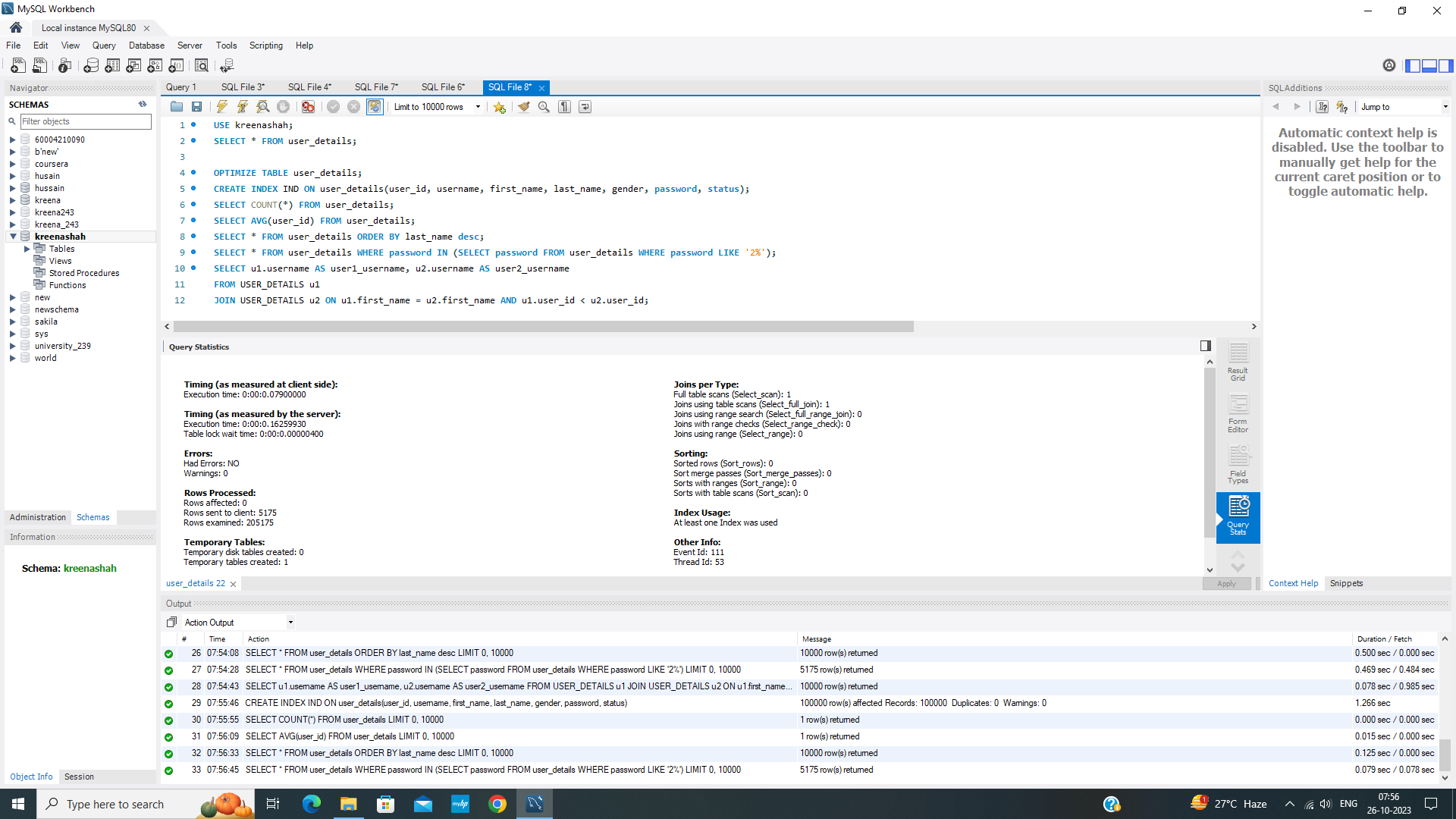
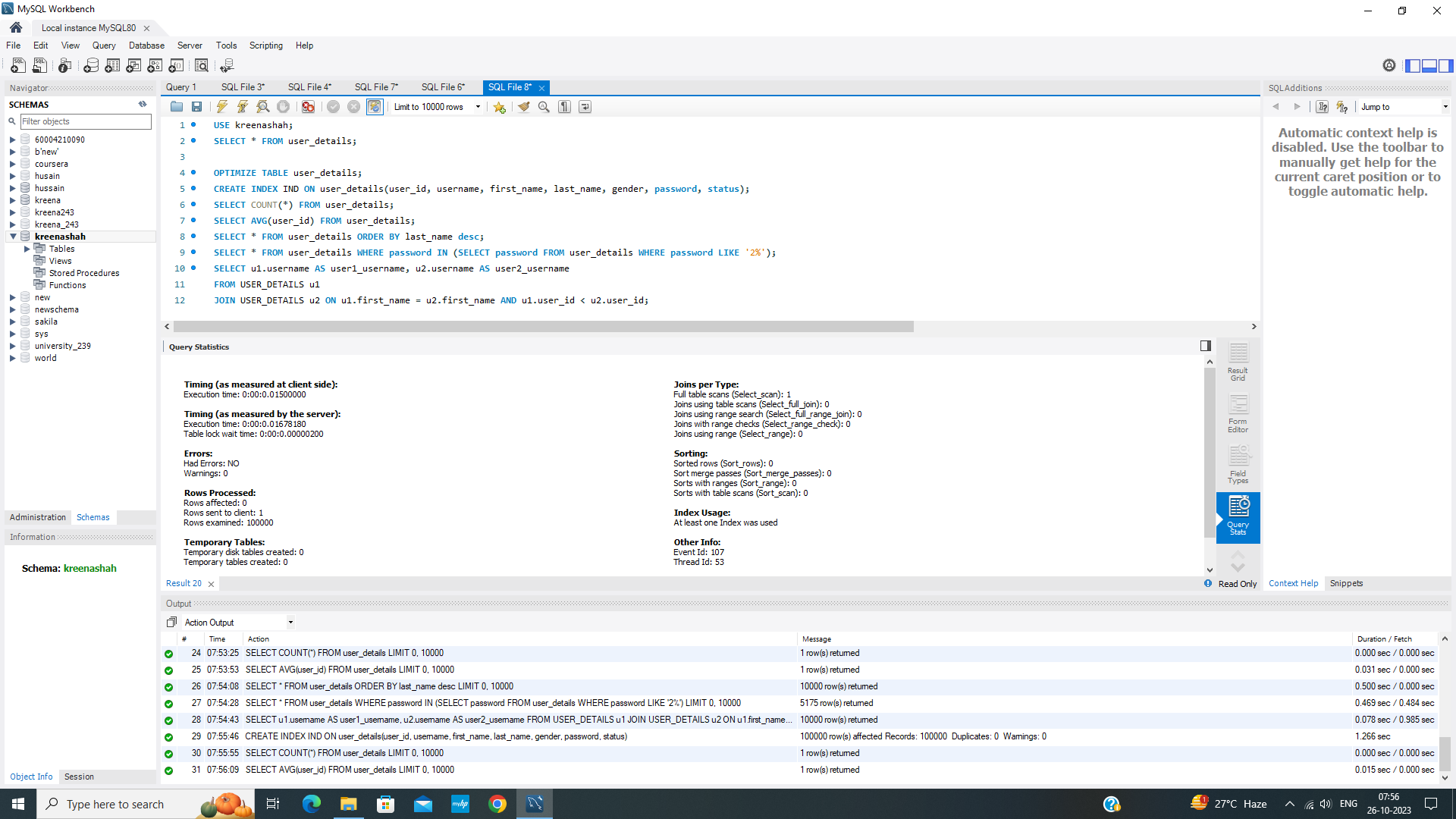
**Optimized Query**



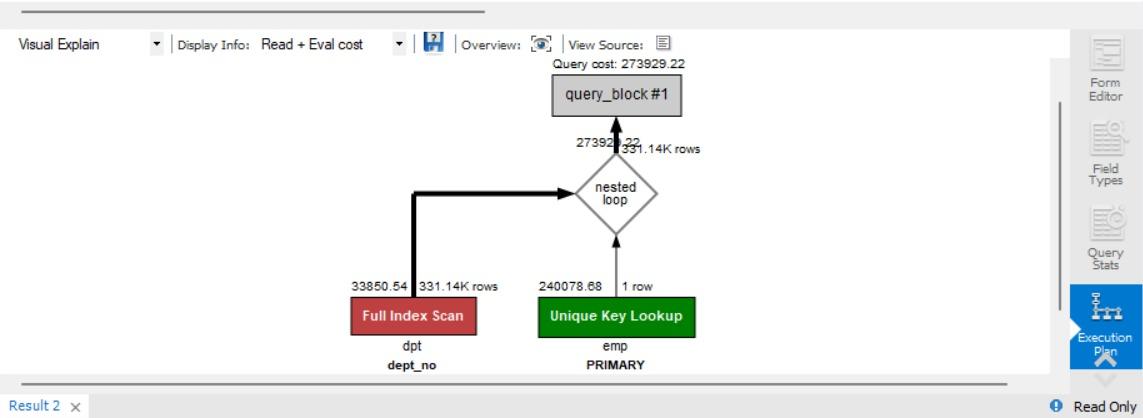




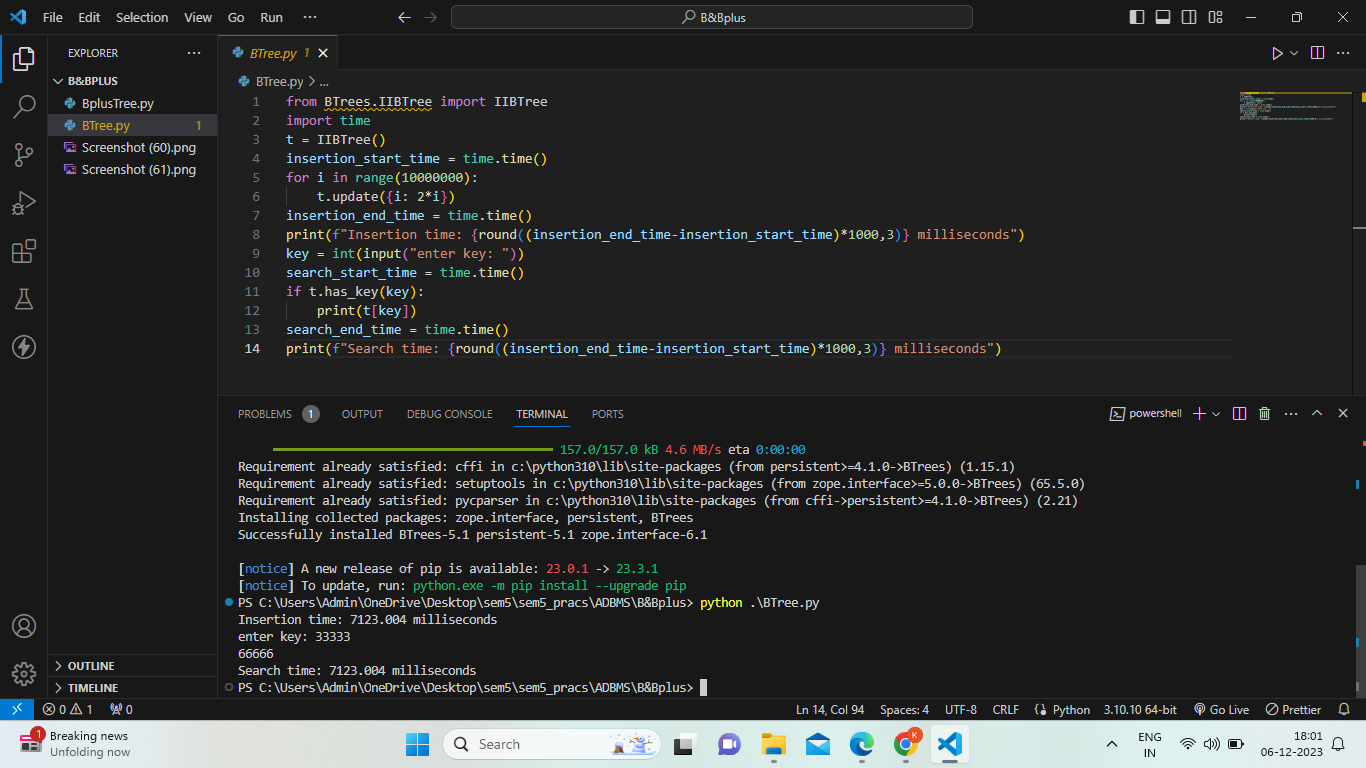
**Indexed Query**

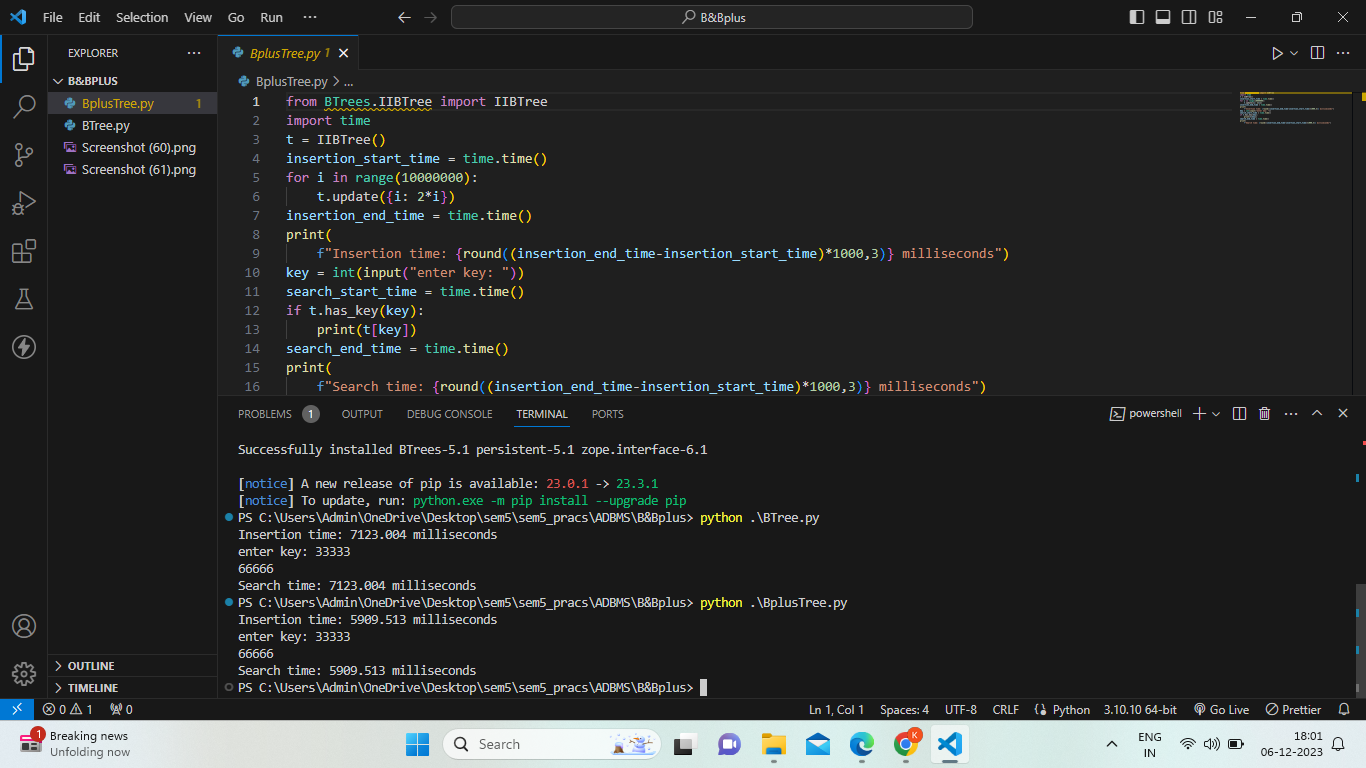


**Experiment No 3 - Query Monitoring**

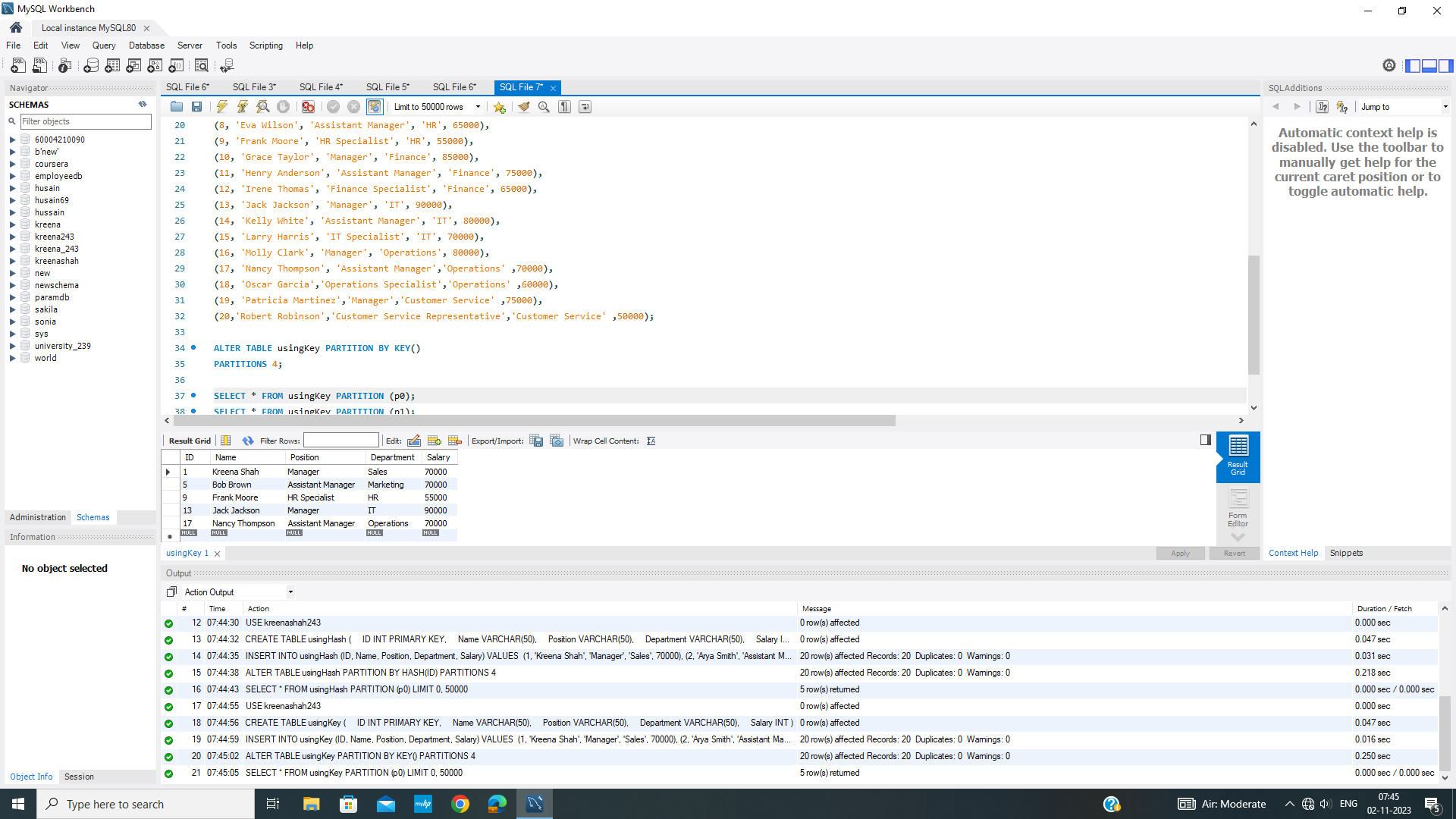
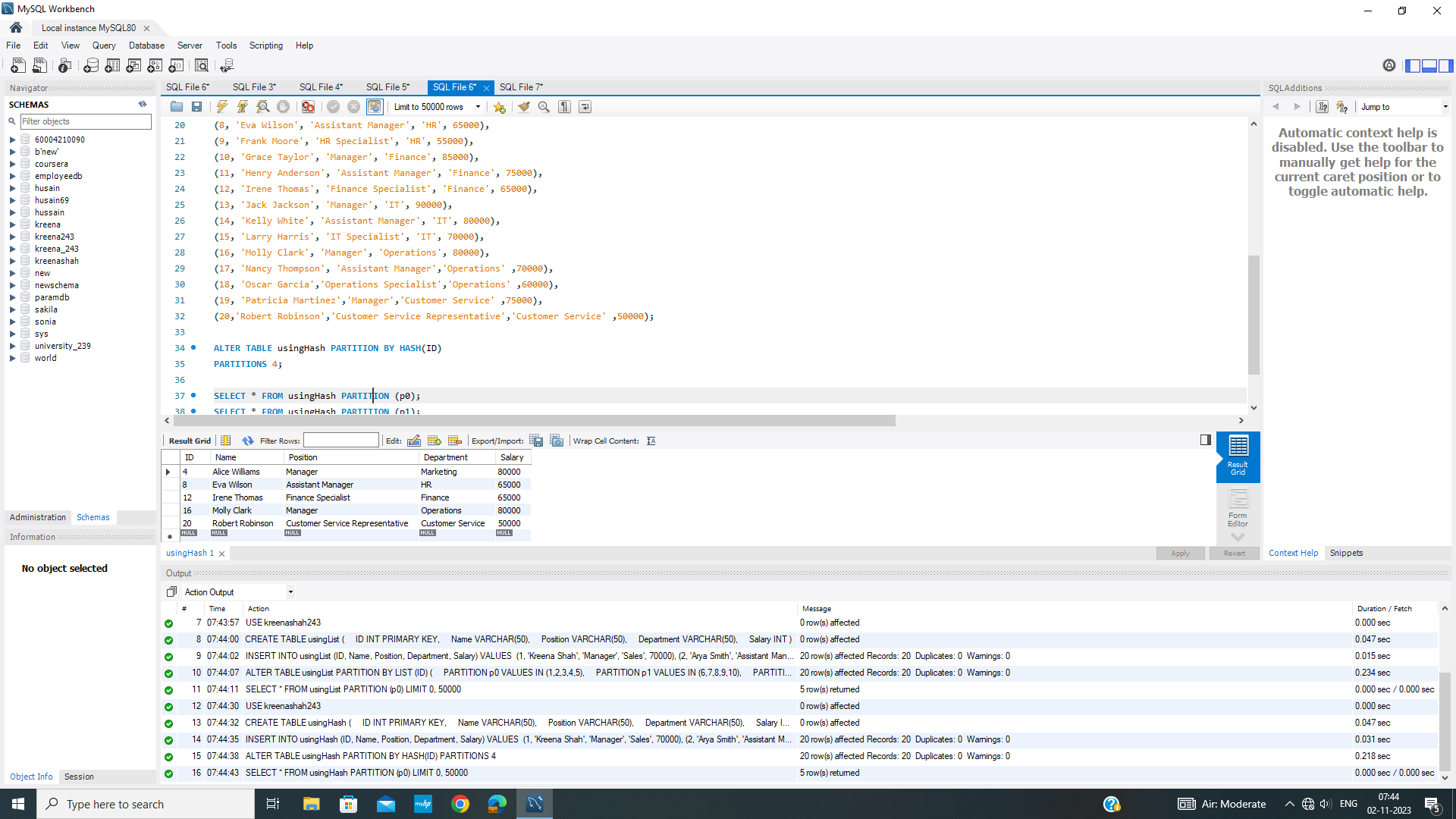
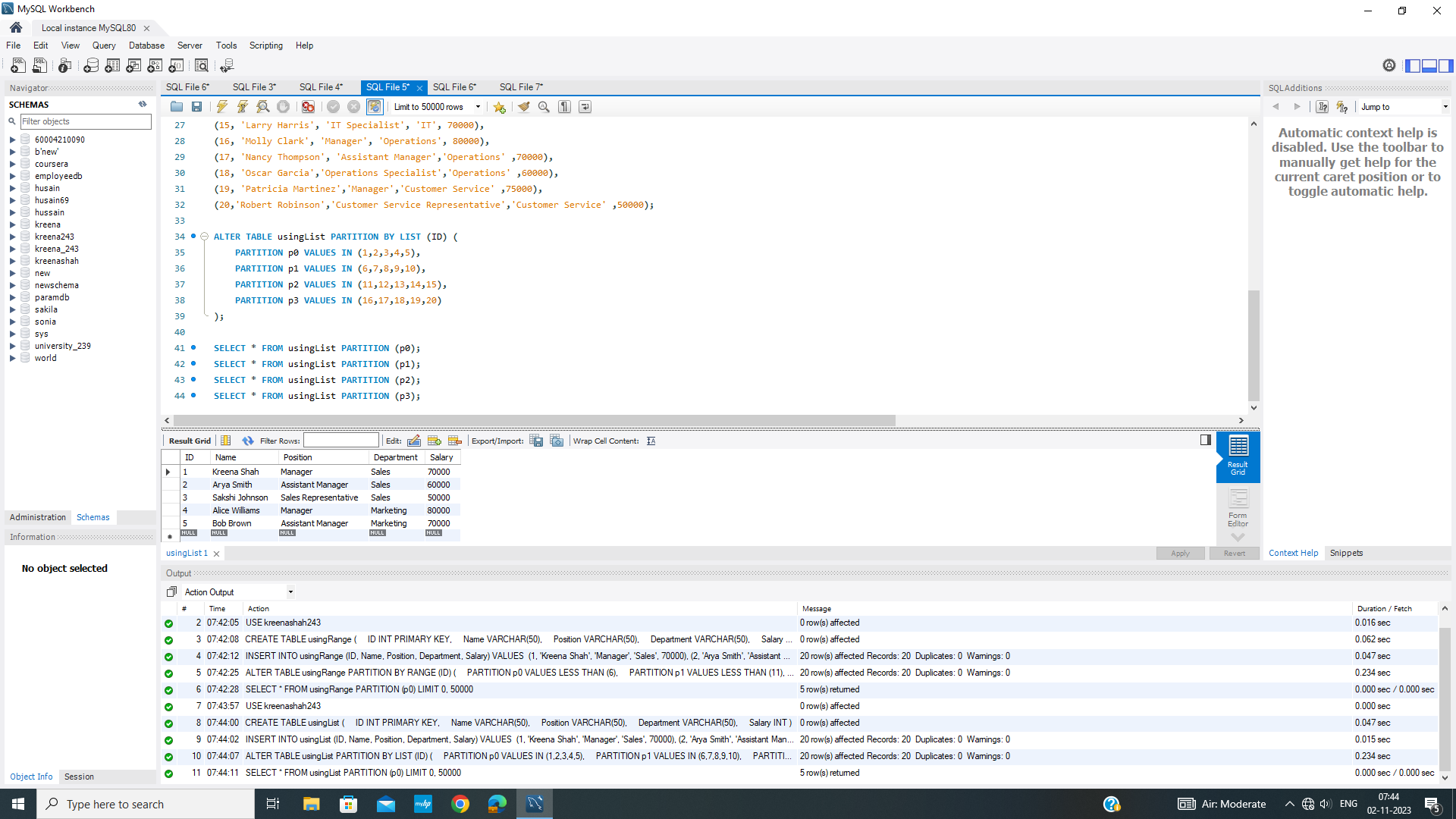
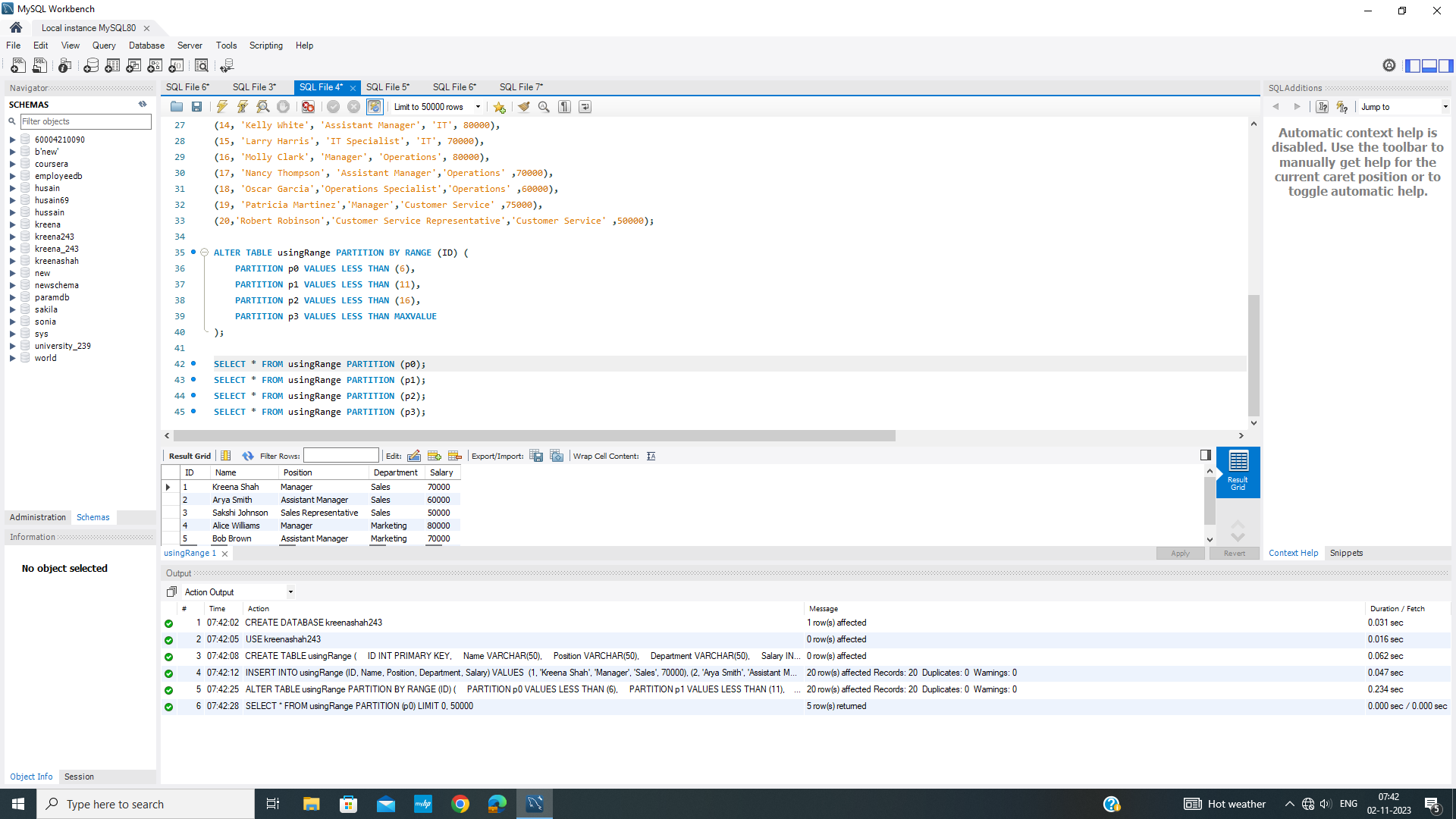


**Experiment No 4 - B/B+ Trees**





**Experiment No 5 - Distributed Database Design**



**Experiment No 6 - Two Phase Protocol**

**Server Code**

import socket def ServerSoc():

host = "127.0.0.1"

port = 8000

print("Server is running!") msg = "PREPARE"

log = msg over = 0

s\_soc = socket.socket()

s\_soc.bind((host,port))

s\_soc.listen(2) while(1):

replies = []

print(f"Coordinator : {msg.upper()}") for i in range(3):

conn,add = s\_soc.accept() conn.send(msg.encode())

data = conn.recv(1024).decode() replies.append(data.upper()) print(f"Subordinator {i} {add} : {data.upper()}")

if over == 1: break

if ("ABORT" in replies) or (len(replies)<3) :

print(f'at the abort stage the replies are {replies},and the length is

{len(replies)}')

msg = "ABORT"

print("TRANSACTION ABORTED!\nThe final log is: ", log+" "+msg) over = 1

elif "SUCCESS" in replies: msg = "COMPLETE"

print("TRANSACTION COMPLETED!\nThe final log is: ", log+" "+msg) over = 1

else:

msg = "COMMIT"

log += " "+msg ServerSoc()

**Client Code**

import socket def ClientSoc():

host = "127.0.0.1"

port = 8000 log = "" over = 0 while(1):

try:

s\_soc = socket.socket() s\_soc.connect((host,port))

rec\_data = s\_soc.recv(1024).decode() print("Coordinator: ", rec\_data.upper()) if rec\_data.upper() == "ABORT":

msg = "OK" print("TRANSACTION ABORTED!")

over = 1

elif rec\_data.upper() == "SUCCESS": msg = "OK"

print("TRANSACTION COMPLETED!")

over = 1

else:

msg = input("System Status: ").upper() log += " "+msg s\_soc.send(msg.encode())

if over == 1: break s\_soc.close()

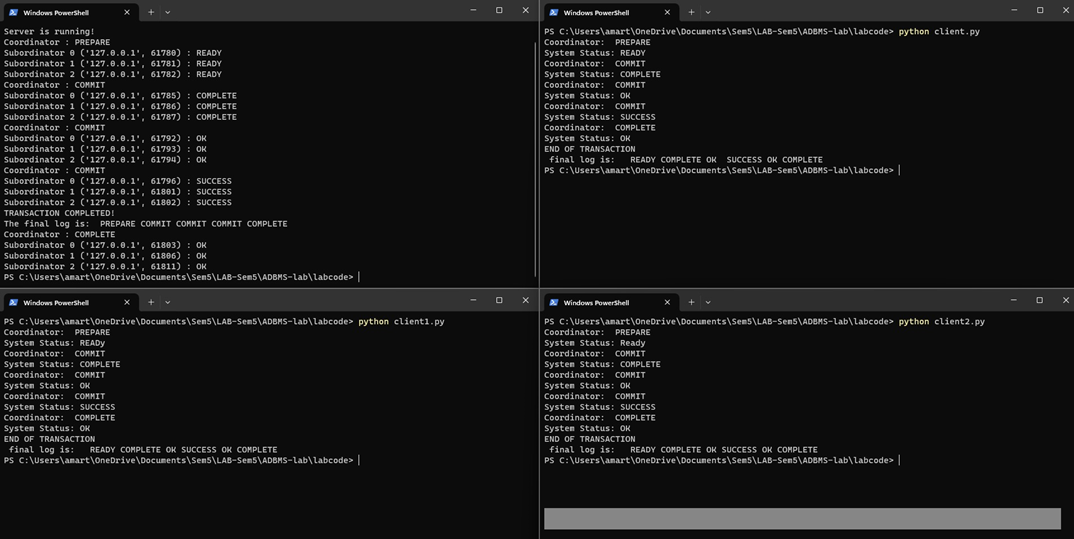
except:

print("END OF TRANSACTION\n final log is: ",log+" "+rec\_data.upper()) break

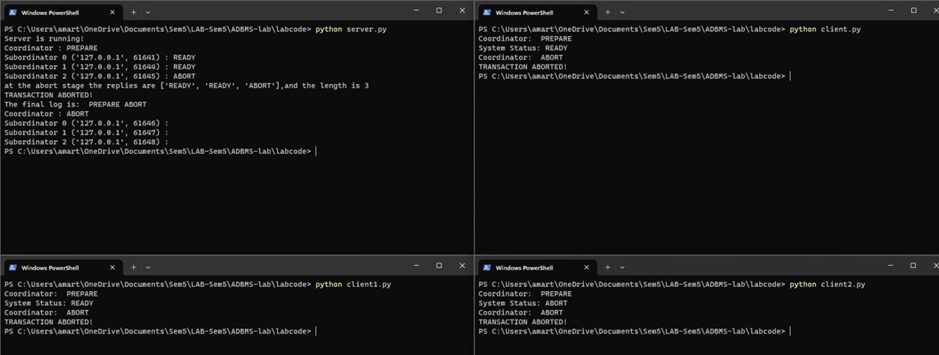
ClientSoc()

Output:

All in the ready phase:



One of the subordinators aborts:



**Experiment No 7 - XML Programming**

**Code :**

<!DOCTYPE html>

<html>

<head>

<title>XML Catalog</title>

</head>

<body>

<h1>Book Catalog</h1>

<input type="text" id="titleInput" placeholder="Search by Title">

<input type="text" id="authorInput" placeholder="Search by Author">

<input type="text" id="genreInput" placeholder="Search by Genre">

<input type="number" id="priceInput" placeholder="Search by Price">

<input type="date" id="dateInput" placeholder="Search by Publish Date">

<button onclick="searchBooks()">Search</button>

<ul id="bookList"></ul>

<script>

var catalogData; // Store the XML data for searching

// Function to load and parse the XML file

function loadXML() {

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {

if (this.readyState === 4 && this.status === 200) {

catalogData = this.responseXML; // Store the XML data for searching

displayCatalog(catalogData);

}

};

xhttp.open("GET", "book.xml", true);

xhttp.send();

}

// Function to display the catalog data

function displayCatalog(xml) {

var books = xml.getElementsByTagName("book");

var list = document.getElementById("bookList");

for (var i = 0; i < books.length; i++) {

var book = books[i];

var author = book.getElementsByTagName("author")[0].textContent;

var title = book.getElementsByTagName("title")[0].textContent;

var genre = book.getElementsByTagName("genre")[0].textContent;

var price = parseFloat(book.getElementsByTagName("price")[0].textContent);

var publishDate = new Date(book.getElementsByTagName("publish\_date")[0].textContent);

var description = book.getElementsByTagName("description")[0].textContent;

var listItem = document.createElement("li");

listItem.innerHTML = "<strong>Title:</strong> " + title + "<br>" +

"<strong>Author:</strong> " + author + "<br>" +

"<strong>Genre:</strong> " + genre + "<br>" +

"<strong>Price:</strong> $" + price.toFixed(2) + "<br>" +

"<strong>Publish Date:</strong> " + publishDate.toDateString() + "<br>" +

"<strong>Description:</strong> " + description + "<br>";

list.appendChild(listItem);

}

}

// Function to search books based on user input

function searchBooks() {

var title = document.getElementById("titleInput").value.toLowerCase();

var author = document.getElementById("authorInput").value.toLowerCase();

var genre = document.getElementById("genreInput").value.toLowerCase();

var price = parseFloat(document.getElementById("priceInput").value);

var date = new Date(document.getElementById("dateInput").value);

var list = document.getElementById("bookList");

list.innerHTML = ""; // Clear the previous list

var books = catalogData.getElementsByTagName("book");

for (var i = 0; i < books.length; i++) {

var book = books[i];

var bookTitle = book.getElementsByTagName("title")[0].textContent.toLowerCase();

var bookAuthor = book.getElementsByTagName("author")[0].textContent.toLowerCase();

var bookGenre = book.getElementsByTagName("genre")[0].textContent.toLowerCase();

var bookPrice = parseFloat(book.getElementsByTagName("price")[0].textContent);

var bookDate = new Date(book.getElementsByTagName("publish\_date")[0].textContent);

if (

(title === "" || bookTitle.includes(title)) &&

(author === "" || bookAuthor.includes(author)) &&

(genre === "" || bookGenre.includes(genre)) &&

(isNaN(price) || price === bookPrice) &&

(isNaN(date) || date.toDateString() === bookDate.toDateString())

) {

var description = book.getElementsByTagName("description")[0].textContent;

var listItem = document.createElement("li");

listItem.innerHTML = "<strong>Title:</strong> " + bookTitle + "<br>" +

"<strong>Author:</strong> " + bookAuthor + "<br>" +

"<strong>Genre:</strong> " + bookGenre + "<br>" +

"<strong>Price:</strong> $" + bookPrice.toFixed(2) + "<br>" +

"<strong>Publish Date:</strong> " + bookDate.toDateString() + "<br>" +

"<strong>Description:</strong> " + description + "<br>";

list.appendChild(listItem);

}

}

}

// Load and display the XML data when the page loads

window.onload = function() {

loadXML();

};

</script>

</body>

</html>

**Database (sample book added here)**

<?xml version="1.0"?>

<catalog>

<book id="bk101">

<author>Gambardella, Matthew</author>

<title>XML Developer's Guide</title>

<genre>Computer</genre>

<price>44.95</price>

<publish\_date>2000-10-01</publish\_date>

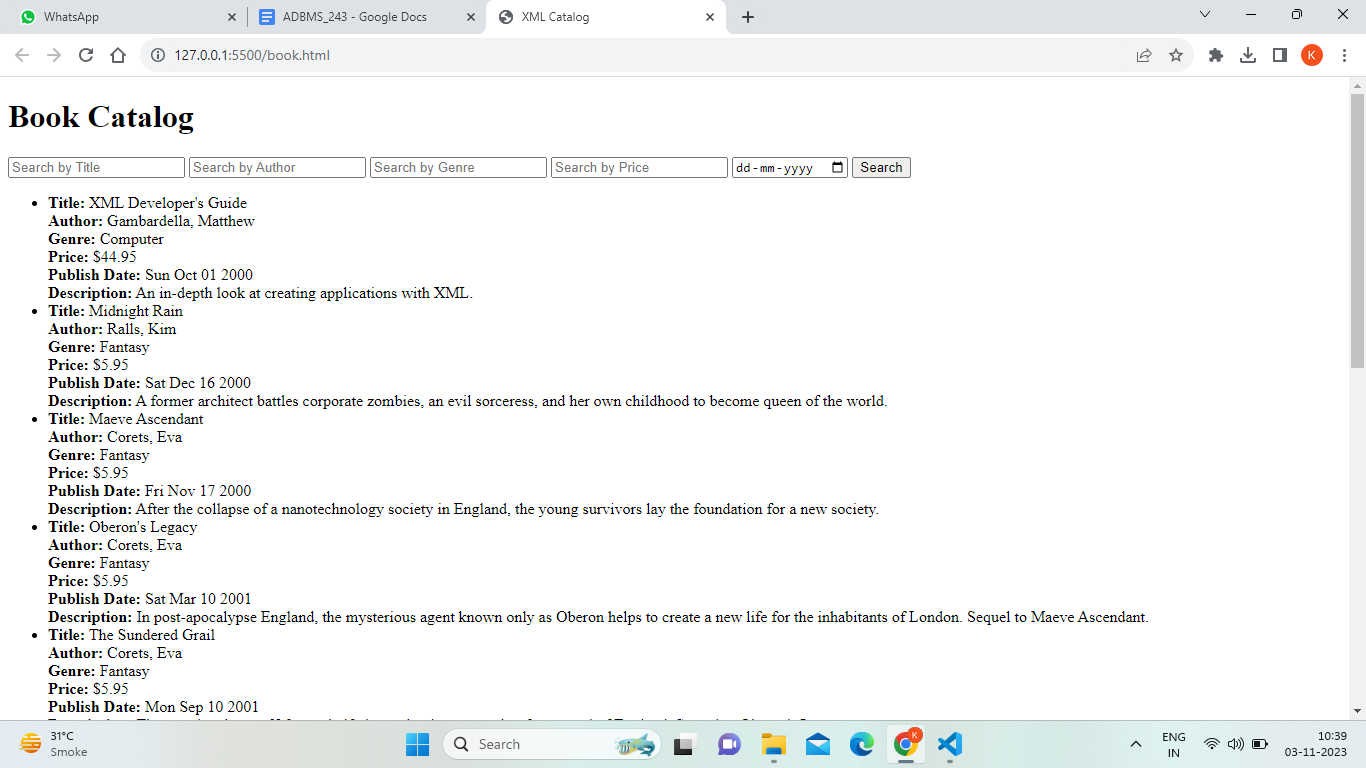
<description>An in-depth look at creating applications

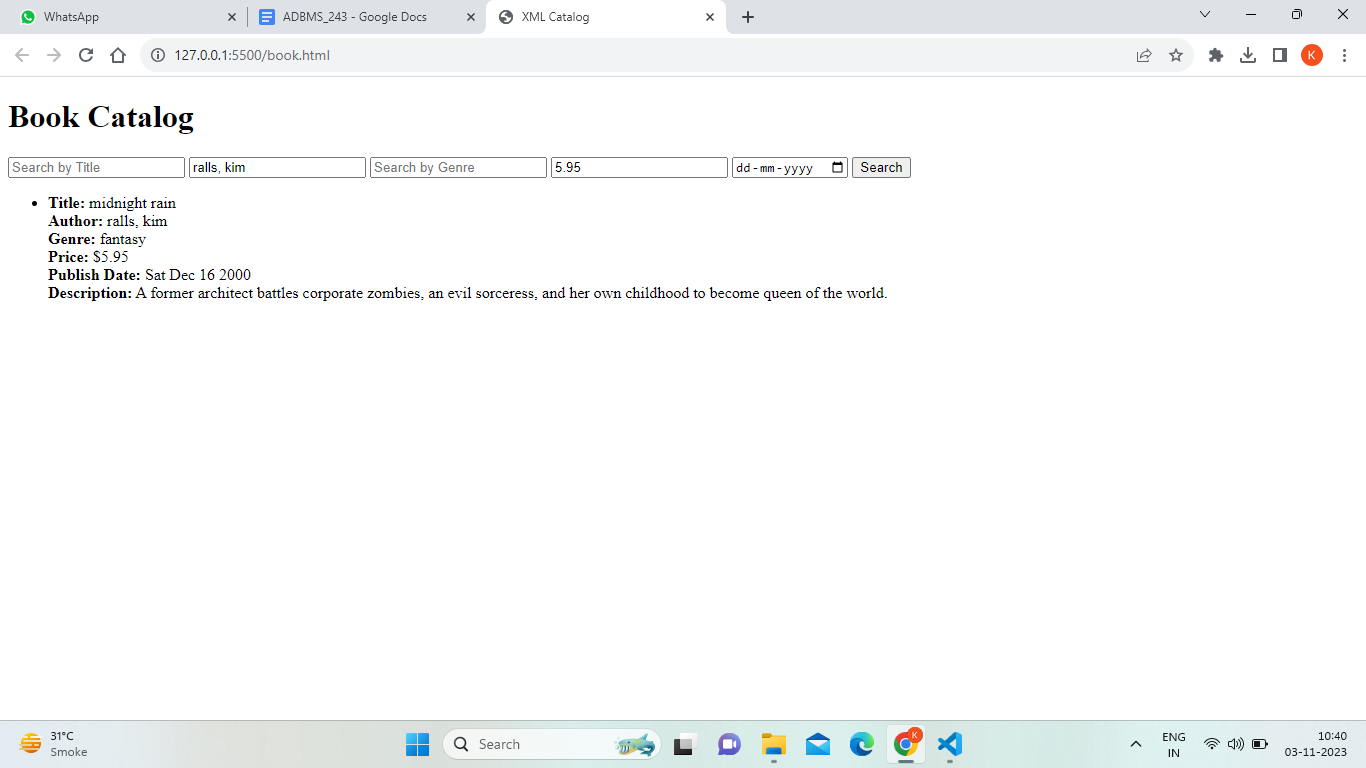
with XML.</description>

</book>

</catalog>

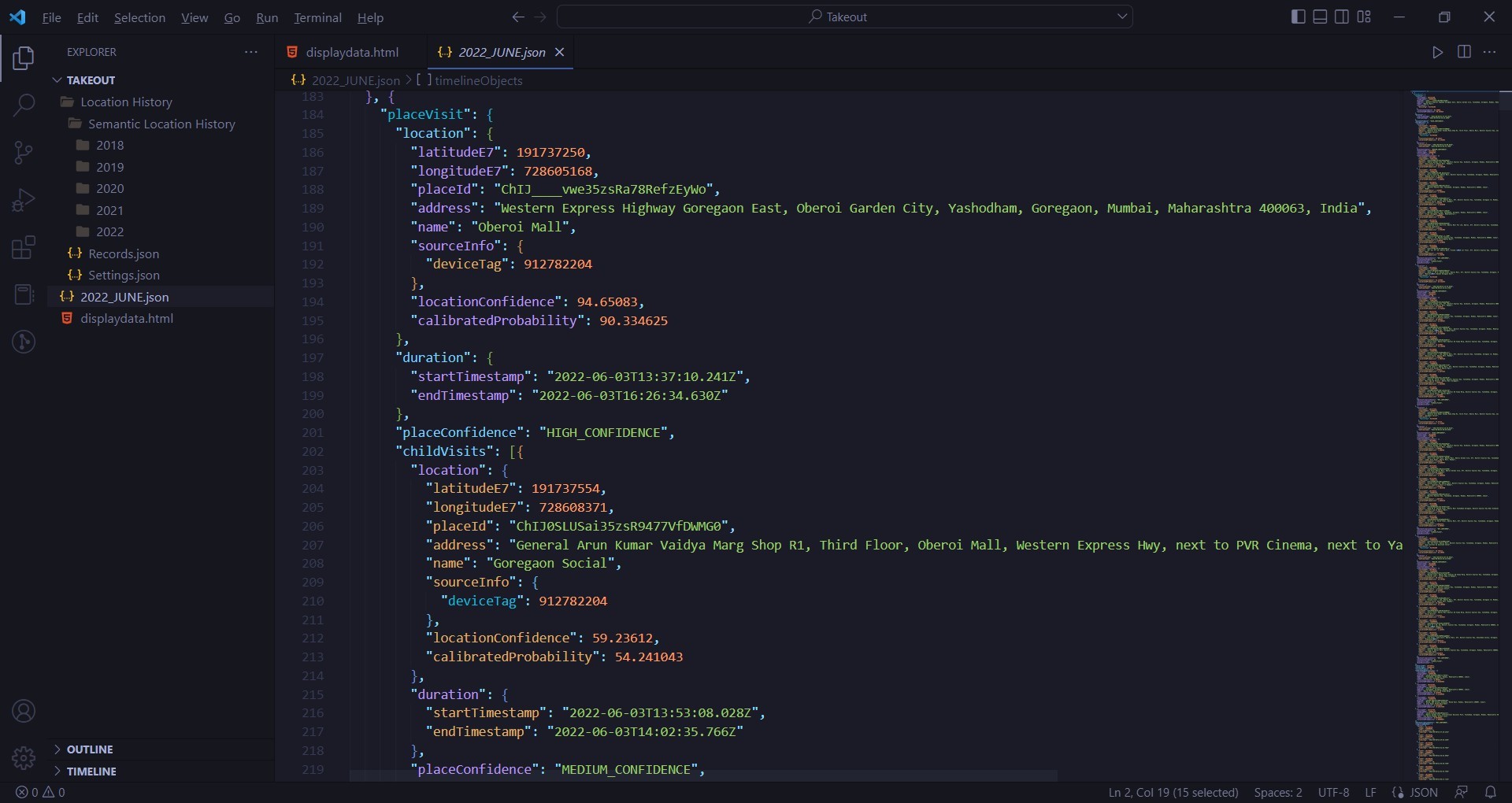
Screenshots :

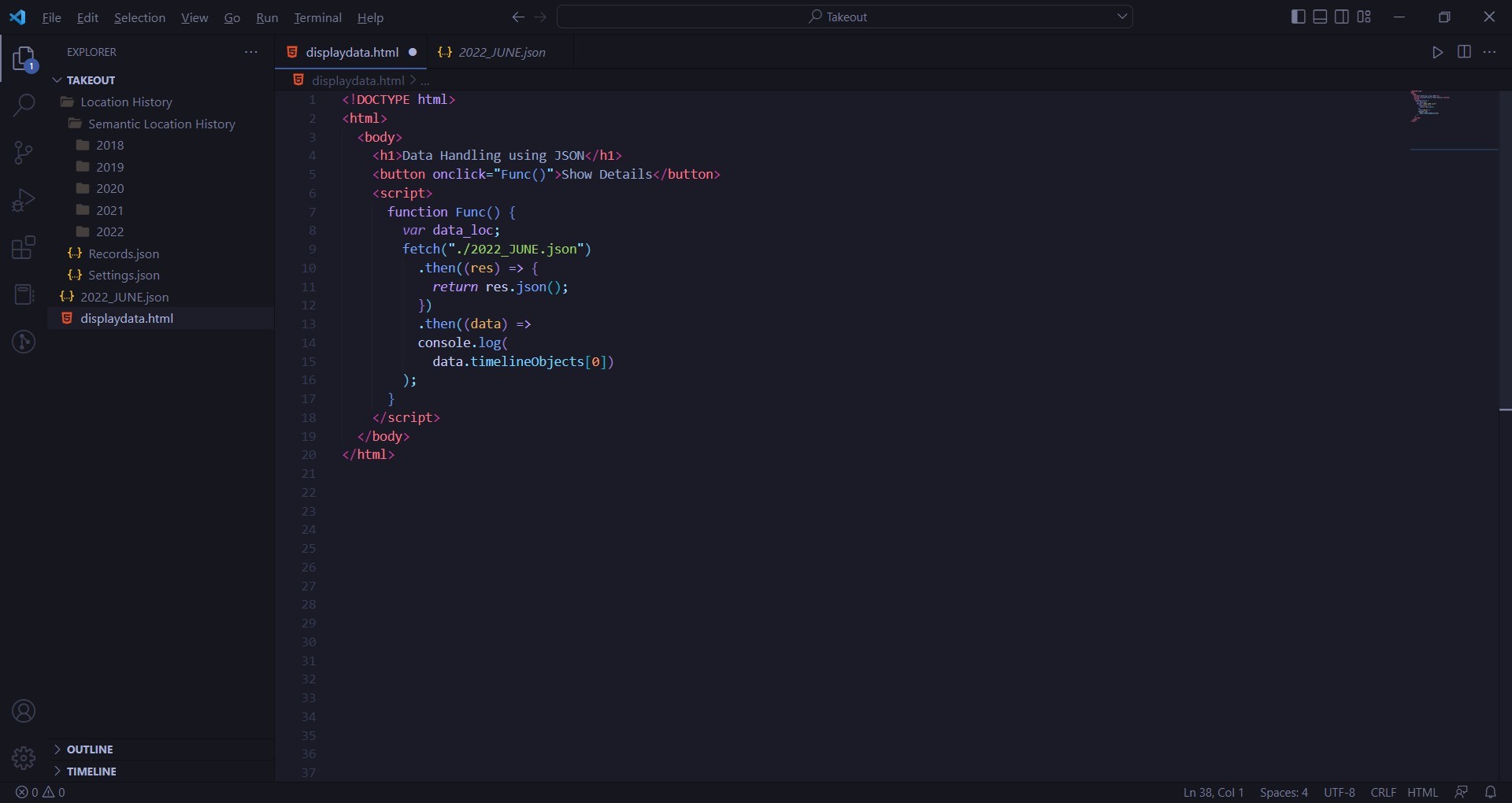


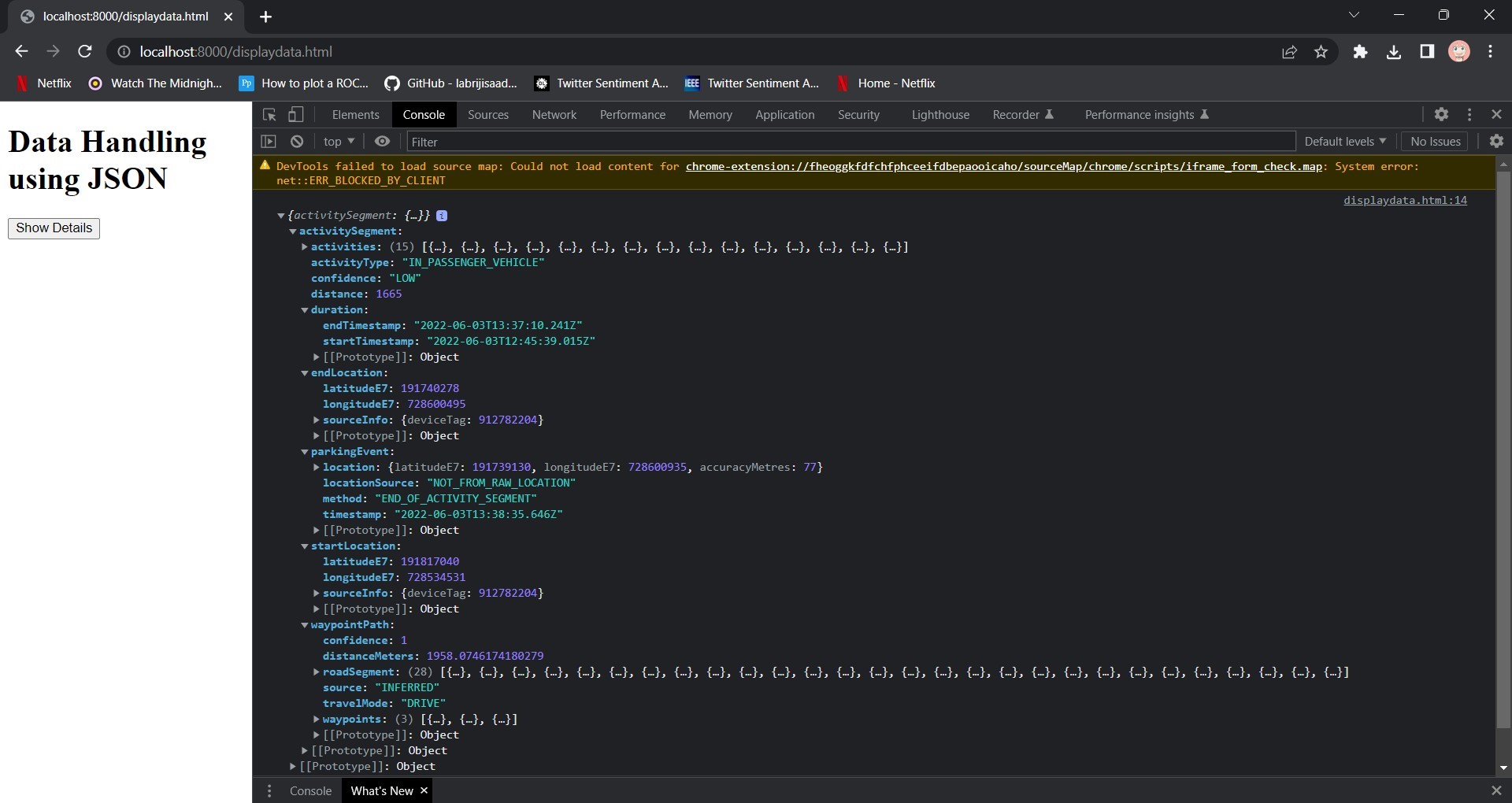


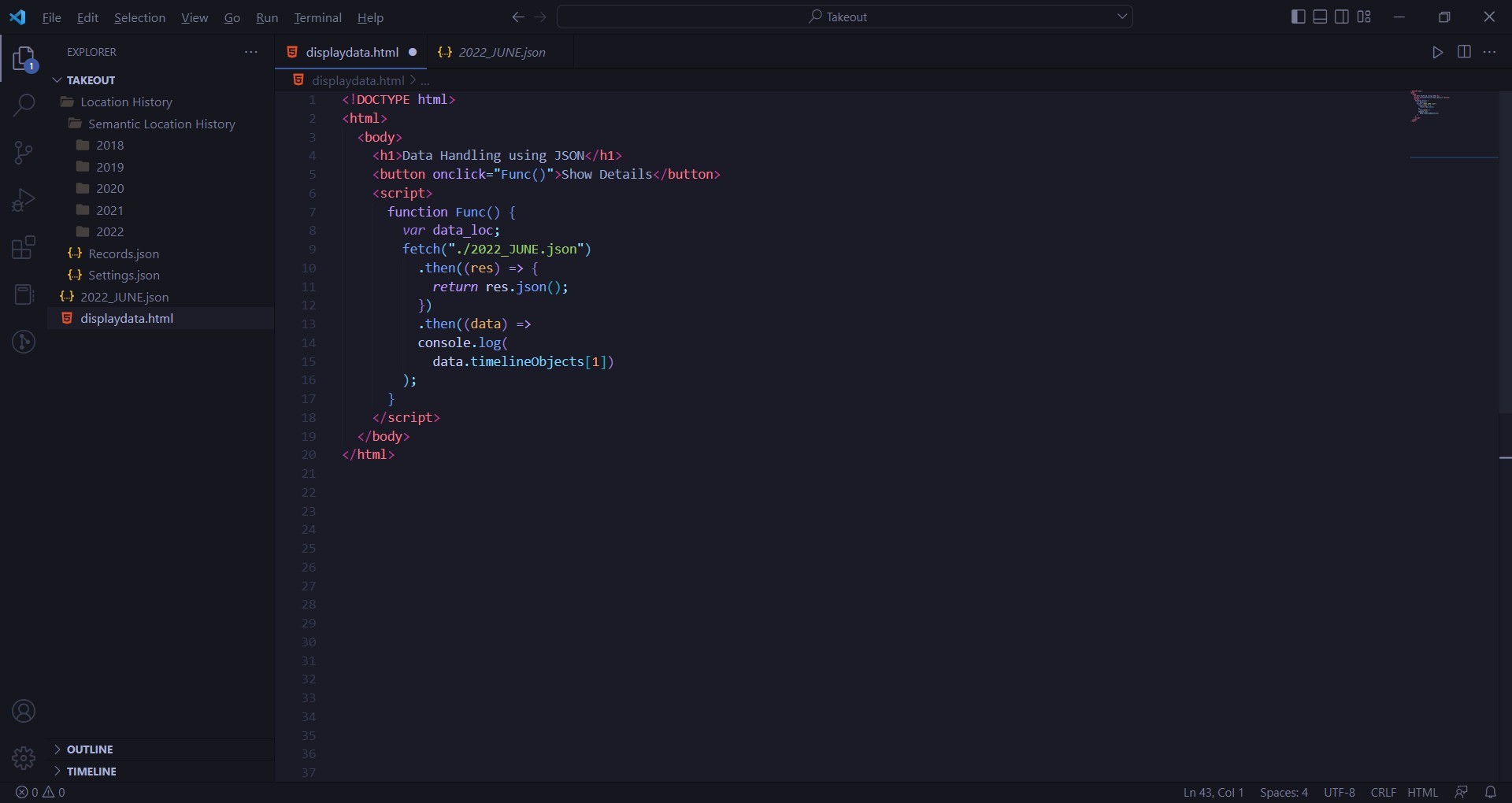
**Experiment No 8 - Document Database**

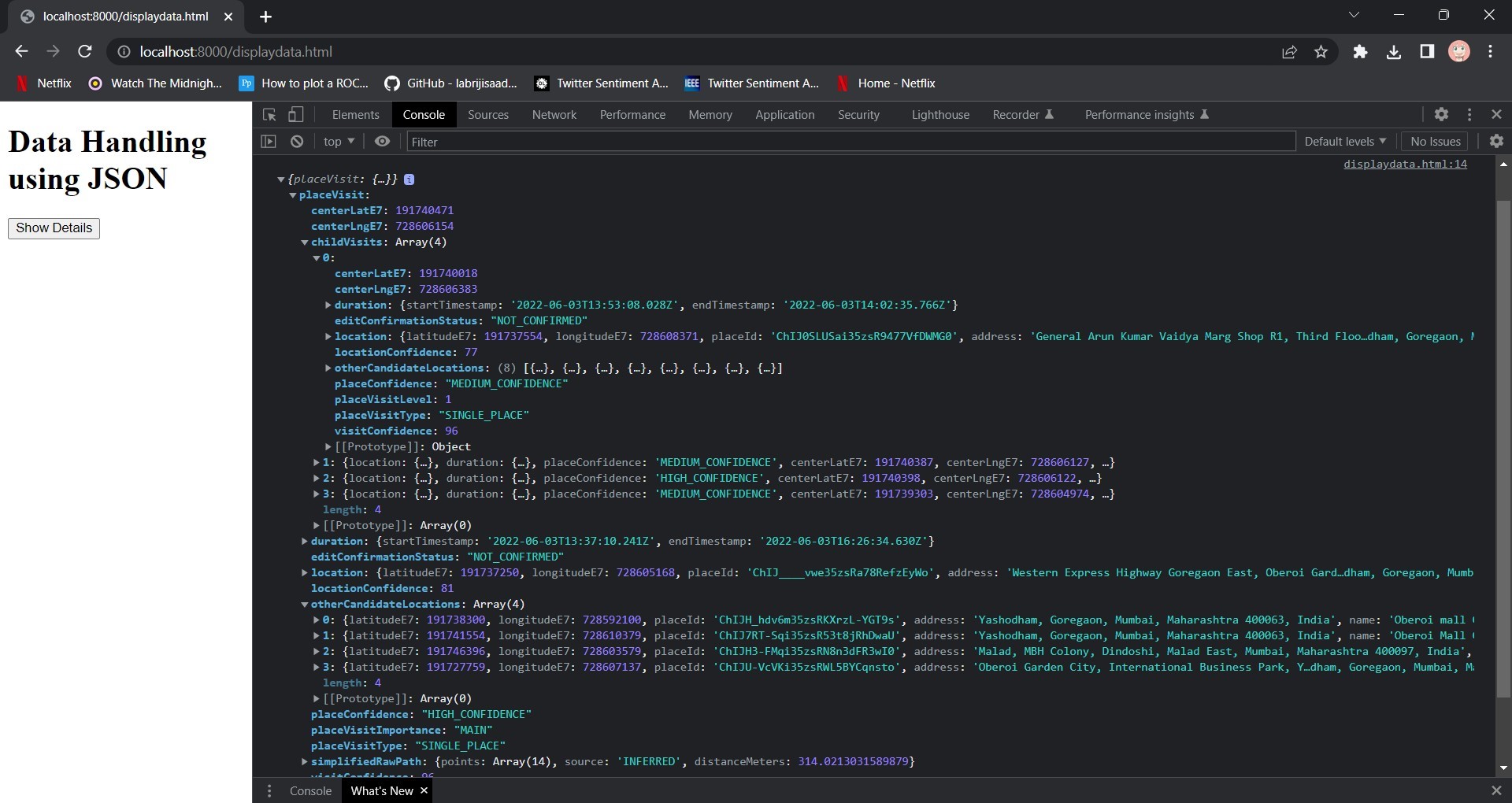












**Experiment No. 9 - Spatial and Temporal Data Handling**

