**DATABASE MANAGEMENT SYSTEMS**

**Name: Kabeleswar P E**

**Roll.no: CB.SCI5das21032**

**Github URL of the project page:** [KB1629/Database-Design (github.com)](https://github.com/KB1629/Database-Design)

**Kaggle URL of the dataset page where the dataset is hosted):**[database design full-stack project - food delivery (kaggle.com)](https://www.kaggle.com/datasets/kabeleswarpe/database-design-full-stack-project-food-delivery/data)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Section-1**

**Application Name:**  Hungry

**Description:** Hungry is basically a food delivery app, where people can order different types of food by seeing the menu on their phones and choose any one from the nearby restaurants and a delivery partner will deliver the food after the payment.

**[About the application]**

**Report:**

**List the report names with its purpose**

|  |  |
| --- | --- |
| **Report Name** | **Purpose** |
| **Orders list** | Top/least selling items can be found. |
| **Customer feedback** | To improve our business , and give customer satisfaction. |
| **Sales report** | Weekly/monthly sales report can be used to compare or create financial reports.(profit/loss) |
| **Driver report** | Driver performance . |

**Technologies**

**SQL Based Application:**

|  |  |
| --- | --- |
| **Front End** | Html (ver-5), css(css-3) |
| **Back End** | SQL Developer – (Ver 20.4) |
| **Editor** | Visual Studio Code |
| **Language** | Java Script |
| **Framework** | Node js |

**Why? What?**

**[[Write why this application is required]]**

"Hungry" is essential as it simplifies food delivery, offering diverse restaurant choices with a user-friendly interface. Hungry aims to be a reliable and innovative solution for those seeking convenience and variety in their food ordering process.

List of similar applications :

|  |  |
| --- | --- |
| Application Name | URL |
| Zomato | [Best Restaurants in India - Zomato](https://www.zomato.com/india) |
| Swiggy | [Order food online from India's best food delivery service. Order from restaurants near you (swiggy.com)](https://www.swiggy.com/) |

**2.Section-2**

**DDL,DML,TCL operations**

**Table Details**

**Other than user Table there should be five master and six transaction tables**

**Master Table**

|  |  |
| --- | --- |
| **Table Name** | **Purpose** |
| **Restaurants** | To have the various restaurants in our database. |
| **Menuitems** | Menu items of various restaurants are here . |
| **Address** | Addresses of all the restaurants. |
| **Delivery drivers** | Information of all the delivery drivers are listed here |
| **Payment methods** | payment methods of users saved here. |

**For Each master table:<<Five table other than user table>>**

**Table Name:** Restaurants

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Datatype** | **Constraint** |
| RestaurantID | int | Primary key |
| Name | varchar | not null |
| Location | varchar |  |
| Cuisine | Varchar |  |
| Rating | decimal |  |
| Opening Time | Varchar |  |
| Closing time | Varchar |  |

**Table Name:** MenuItems

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Datatype** | **Constraint** |
| ItemID | Int | Primary key |
| RestaurantID | Int | foreignkey RestaurantID ref Restaurants(RestaurantID) |
| Name | Varchar | Notnull |
| Price | Varchar | Notnull |
| Description | text |  |

**Table Name:** DeliveryDrivers

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Datatype** | **Constraint** |
| DriverID | int | Primary key |
| name | varchar | Not null |
| phone | Varchar | Unique |
| Vehicle type | varchar |  |

**Table Name:** Addresess

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Datatype** | **Constraint** |
| **AddressID** | **Int** | **Primary key** |
| **Userid** | **Int** | **Foreignkey key ref users(userid)** |
| **Street** | **Varchar** |  |
| **City** | **Varchar** |  |
| **State** | **Varchar** |  |
| **Postal code** | **varchar** |  |

**Table Name:** PaymentMethods

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Datatype** | **Constraint** |
| PAymentmethodId | Int | primarykey |
| Userid | Int | FK(userid) references users(userid) |
| Cardnumber | Varchar |  |
| Expiryddate | Date |  |
| Cvv | varchar |  |

**For Each transaction table:<<Six table other than user table>>**

**Table Name:** Orders

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Datatype** | **Constraint** |
| OrderID | Int | Primary key |
| UserID | Int | FK(userID) ref Users(UserID) |
| RestaurantID | Int |  |
| TotalAmount | Decimal |  |
| Status | varchar | FK(restaurantID) ref Users(RestaurantID) |

**Table Name:** OrderItems

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Datatype** | **Constraint** |
| OrderITemID | int | Primary key |
| ORderid | Int | FOREIGN KEY (OrderID) REFERENCES Orders(OrderID) |
| Itemid | Int | FOREIGN KEY (ItemID) REF MenuItems(ItemID) |
| Quantity | Int |  |
| Subtotal | Decimal |  |

**Table Name: Payments**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Datatype** | **Constraint** |
| PaymentID | Int | Primary key |
| OrderID | Int | FOREIGN KEY (OrderID) REFERENCES Orders(OrderID) |
| PaymentMethod | Int | FOREIGN KEY (PaymentMethodID) REFERENCES PaymentMethods(PaymentMethodID) |
| Amount | Decimal |  |

**Table Name: Reviews**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Datatype** | **Constraint** |
| ReviewID | int | Primary key |
| Userid | Int | FOREIGN KEY (UserID) REFERENCES Users(UserID) |
| REstaurantID | Int | FOREIGN KEY (RestaurantID) REFERENCES Restaurants(RestaurantID) |
| Rating | Decimal |  |
| Comment | Text |  |

**Table Name: Delivery**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Datatype** | **Constraint** |
| DeliveryID | int | Primary key |
| OrderID | Int | FOREIGN KEY (UserID) REFERENCES Users(UserID) |
| DriverID | Int | FOREIGN KEY (DriverID) REFERENCES DeliveryDrivers(DriverID) |
| DeliveryDate | Timestamp |  |
| Status | varchar |  |

**Table Name: Favorites**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Datatype** | **Constraint** |
| FavoriteID | Int | Primary key |
| USerID | Int | FOREIGN KEY (UserID) REFERENCES Users(UserID) |
| RestaurantID | int | FOREIGN KEY (RestaurantID) REFERENCES Restaurants(RestaurantID) |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Operations (DDL,DML,TCL):**

**1.Create**

**2.insert**

**3.Delete**

**4.Update**

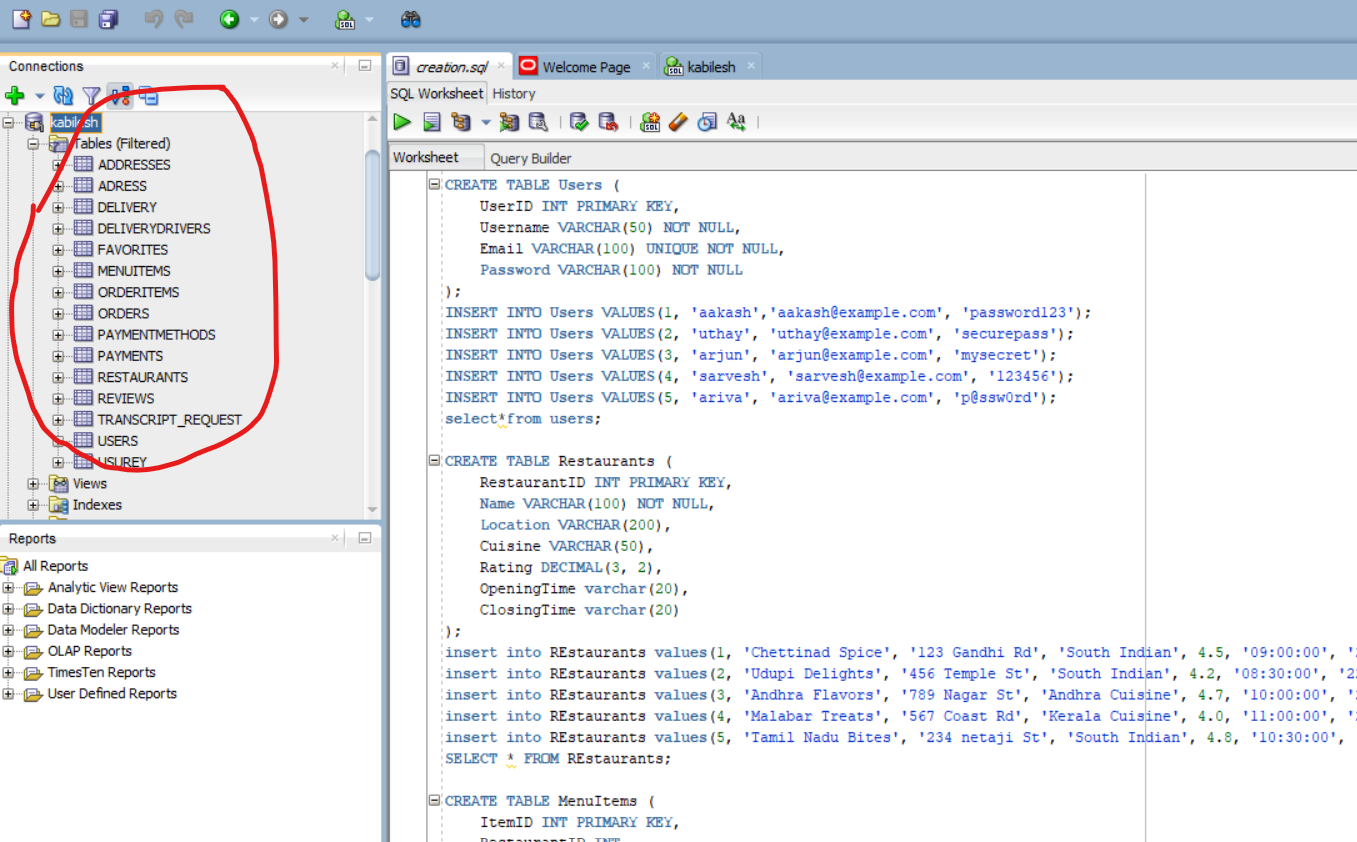
Operations with creation insertion deletion and updation is done in sql and saved in the file named – creation.sql

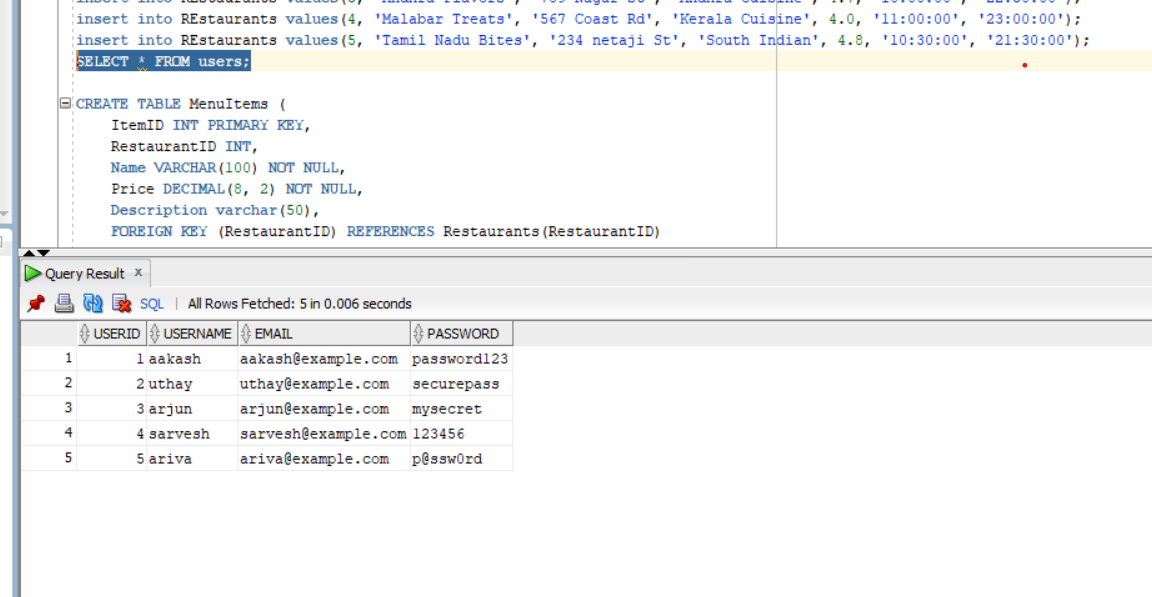
**-- Sql query file name<<creation.sql>>**

**<<Place outputs of execution>>**

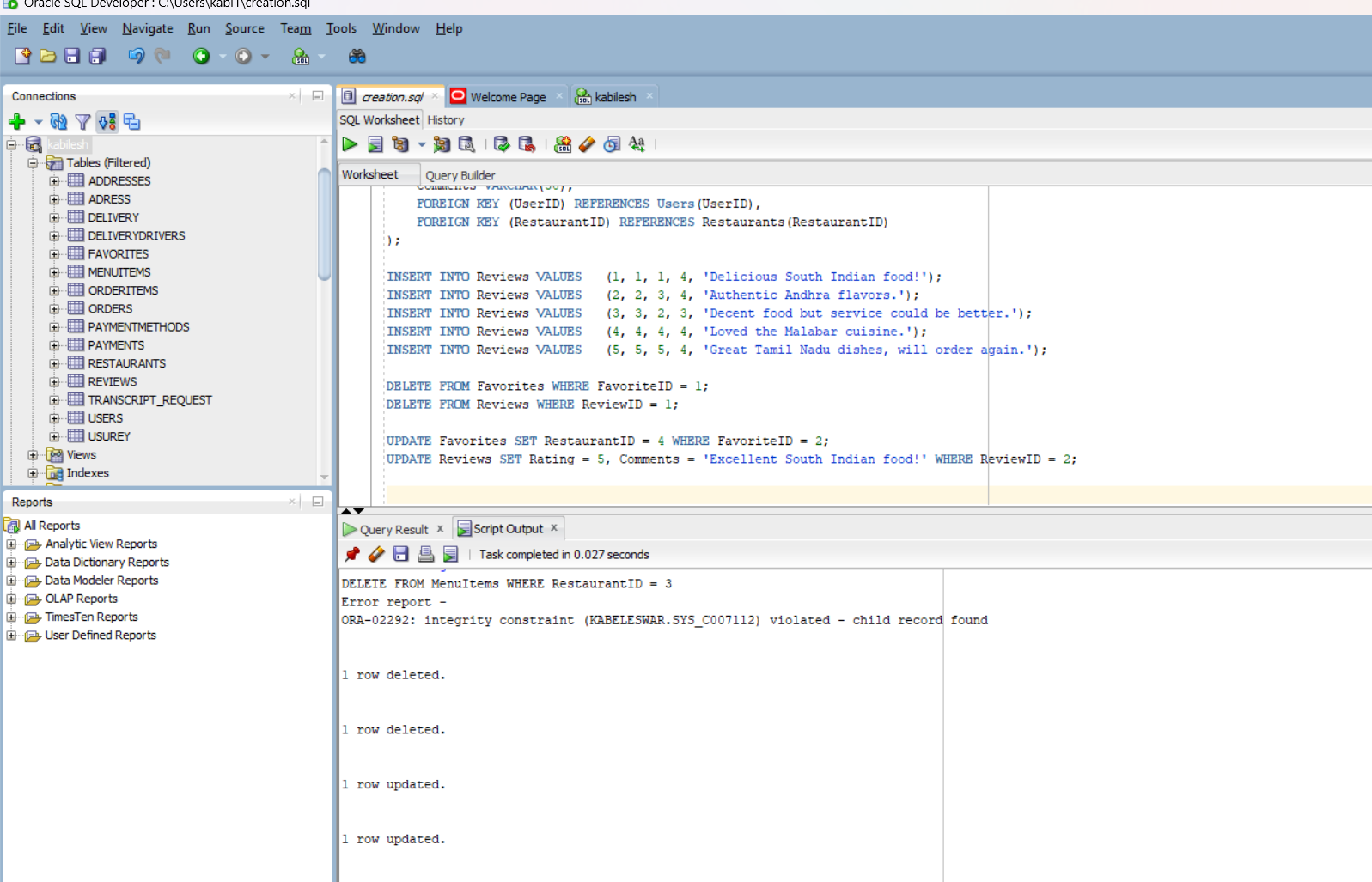
**Screenshot:** Examples of create, insert, delete and update ( 2 in each ) taken screenshot and uploaded here; remaining are under SQL file name “**creation.sql”.**

1.Tables created and values inserted their examples:



****

**Delete and update examples:**

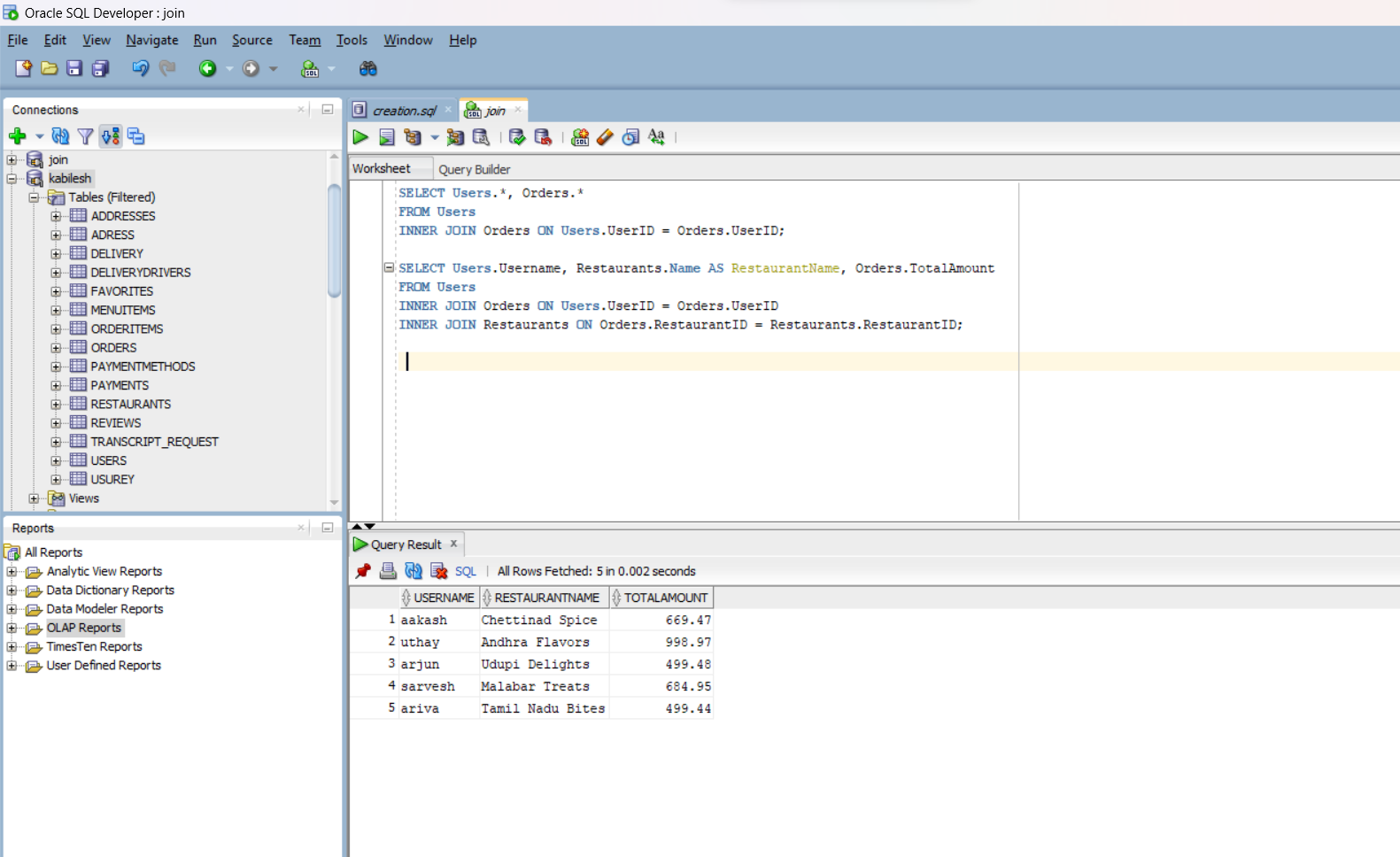
****

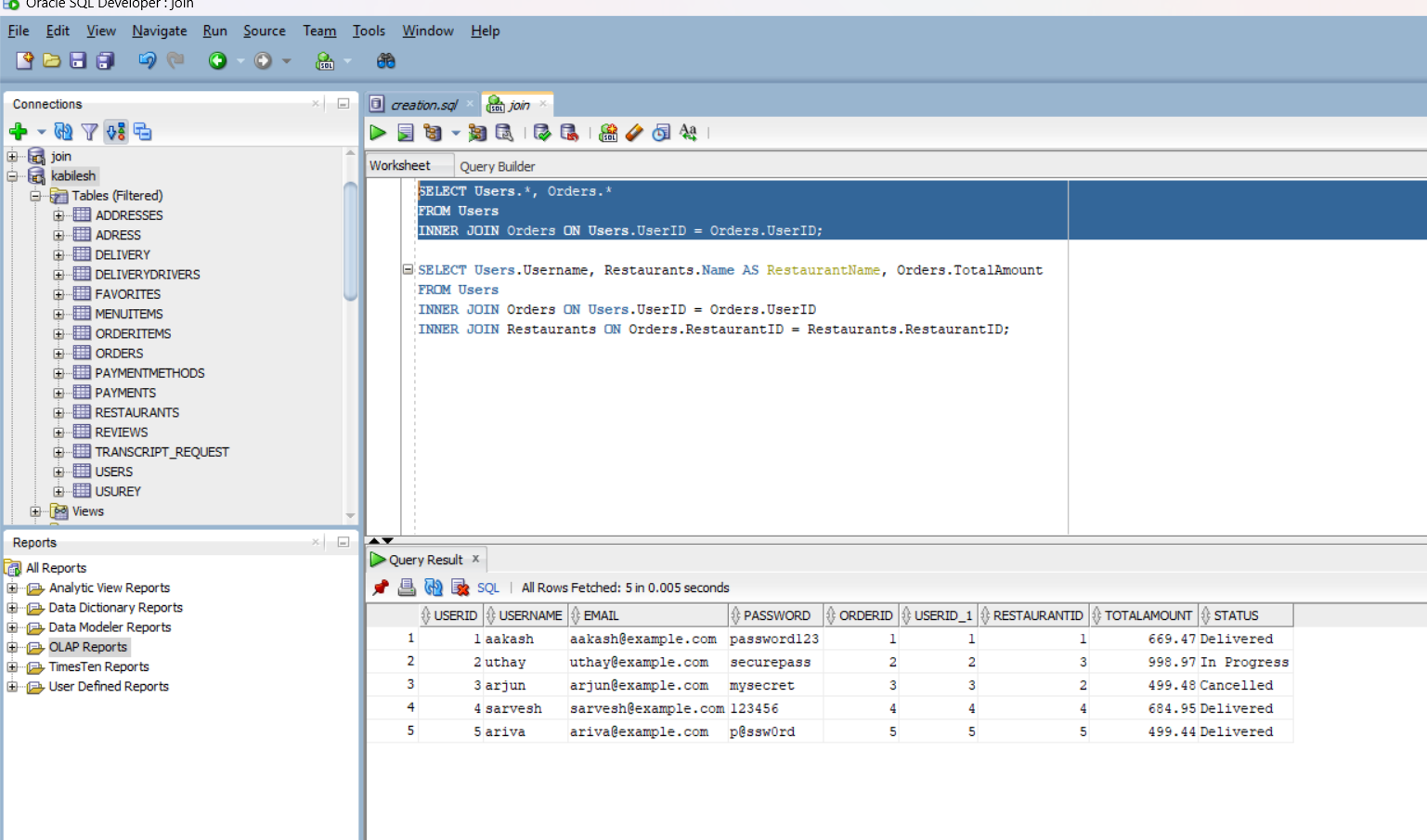
**3.Section-3(Join)**

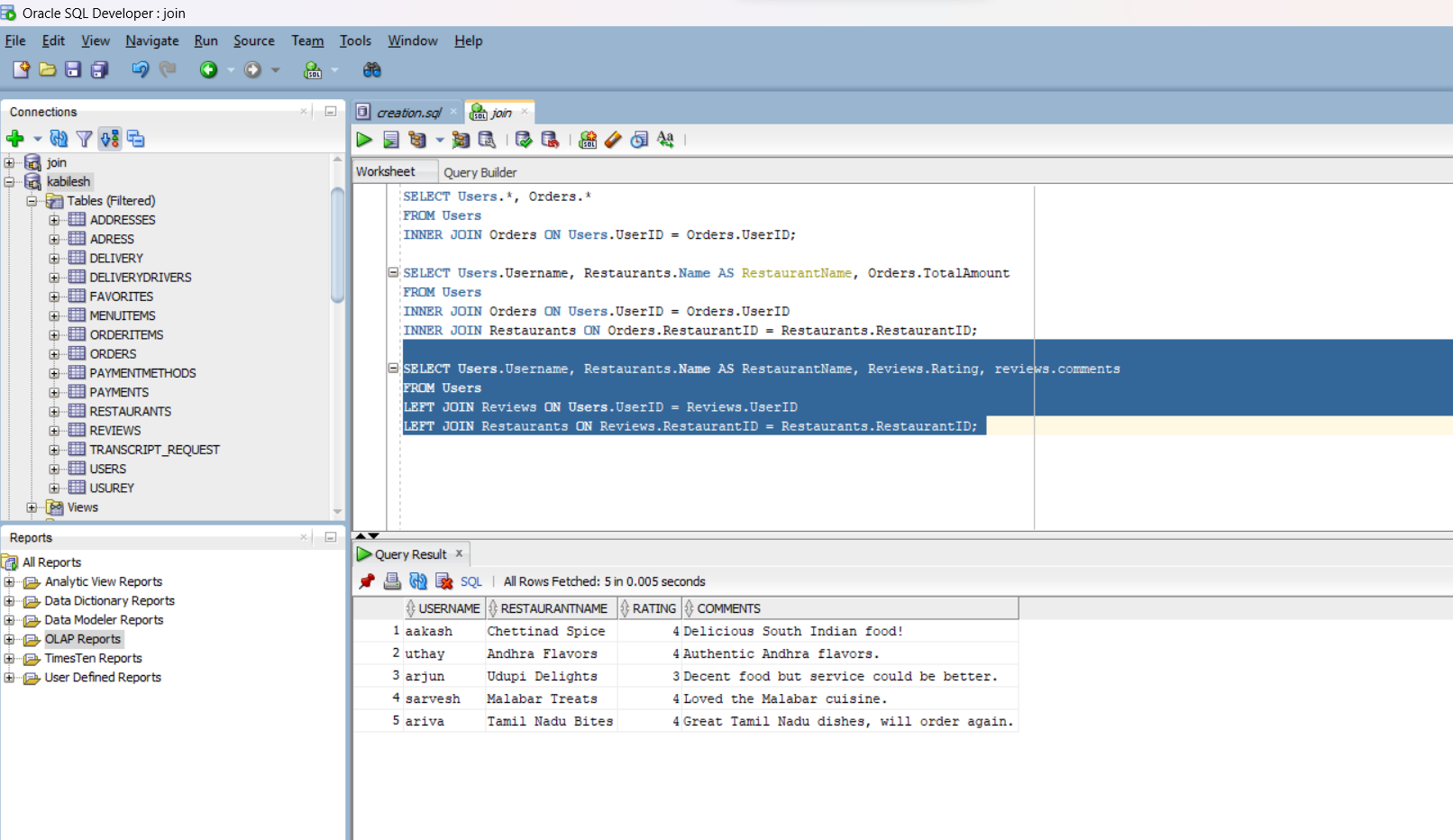
**-- Sql query file name<<joins.sql>>**

**Screenshot:**

**<<Place outputs of execution>>**

****

****

**Inference:**

**<<Explain what you would like to explain about the output>>**

**Query 1:** **Inference:**

This query retrieves all columns from both the Users and Orders tables for rows where the UserID in the Users table matches the UserID in the Orders table.

**Query 2: Inference:**

This query retrieves the Username from the Users table, the Name column from the Restaurants table (aliased as RestaurantName), and the TotalAmount from the Orders table. It fetches rows where the UserID in the Users table matches the UserID in the Orders table, and the RestaurantID in the Orders table matches the RestaurantID in the Restaurants table.

**Query 3: Inference:**

This query retrieves the Username from the Users table, the Name column from the Restaurants table (aliased as RestaurantName), Rating, and comments from the Reviews table. It uses a LEFT JOIN to include all rows from the Users table and matching rows from the Reviews and Restaurants tables based on UserID and RestaurantID respectively. If there is no match, the result will still include the row from the Users table with NULL values for the columns from the other tables.

**In summary,** these queries are useful for fetching information related to user orders, including details about the restaurant and any reviews they may have given. The JOIN operations help link data from multiple tables based on common keys.

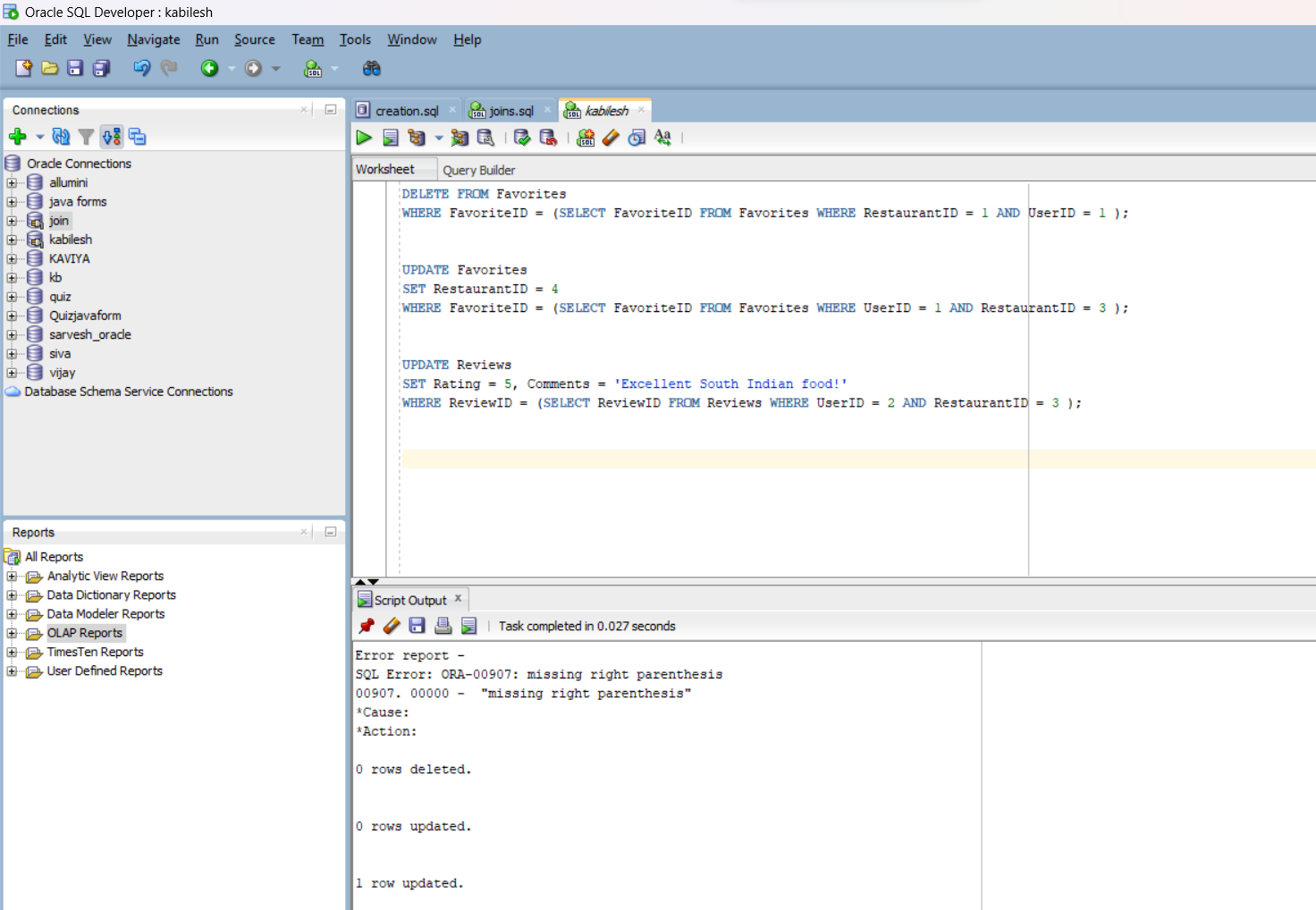
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**4.Section-4(Nested Queries)**

**-- Sql query file name<<nestedqueries.sql>>**

**Screenshot:**

**<<Place outputs of execution>>**

****

**Inference:**

**<<Explain what you would like to explain about the output>>**

**Query 1 Inference**: This subquery updates a record in the Favorites table, setting the RestaurantID to 4, where the UserID is 1 and the RestaurantID is initially 3. It uses a subquery to find the FavoriteID corresponding to the specified conditions.

**Query 2 Inference:** This subquery updates a record in the Reviews table, setting the Rating to 5 and updating the Comments to 'Excellent South Indian food!'. It targets the record where the UserID is 2 and the RestaurantID is 3, using a subquery to find the corresponding ReviewID.

**QUERY 3 Inference:** This subquery updates a record in the Reviews table, setting the Rating to 5 and updating the Comments to 'Excellent South Indian food!'. It targets the record where the UserID is 2 and the RestaurantID is 3, using a subquery to find the corresponding ReviewID.

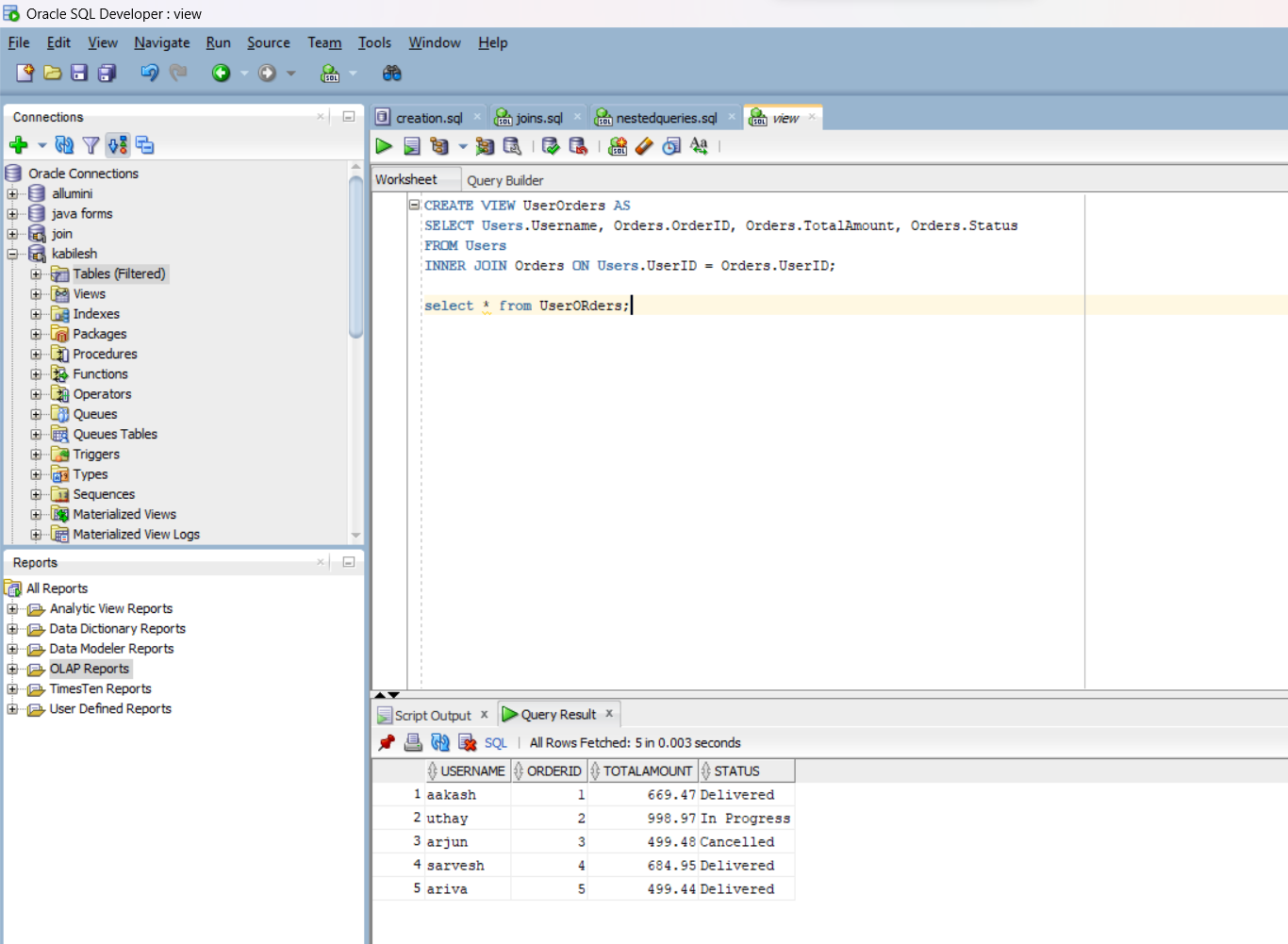
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**5.Section-5(view)**

**-- Sql query file name<<view.sql>>**

**Screenshot:**

**<<Place outputs of execution>>**

****

**Inference:**

**<<Explain what you would like to explain about the output>>**

This view retrieves information about user orders, including the username, order ID, total amount, and status.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**6. Section-6(PLSQL file)**

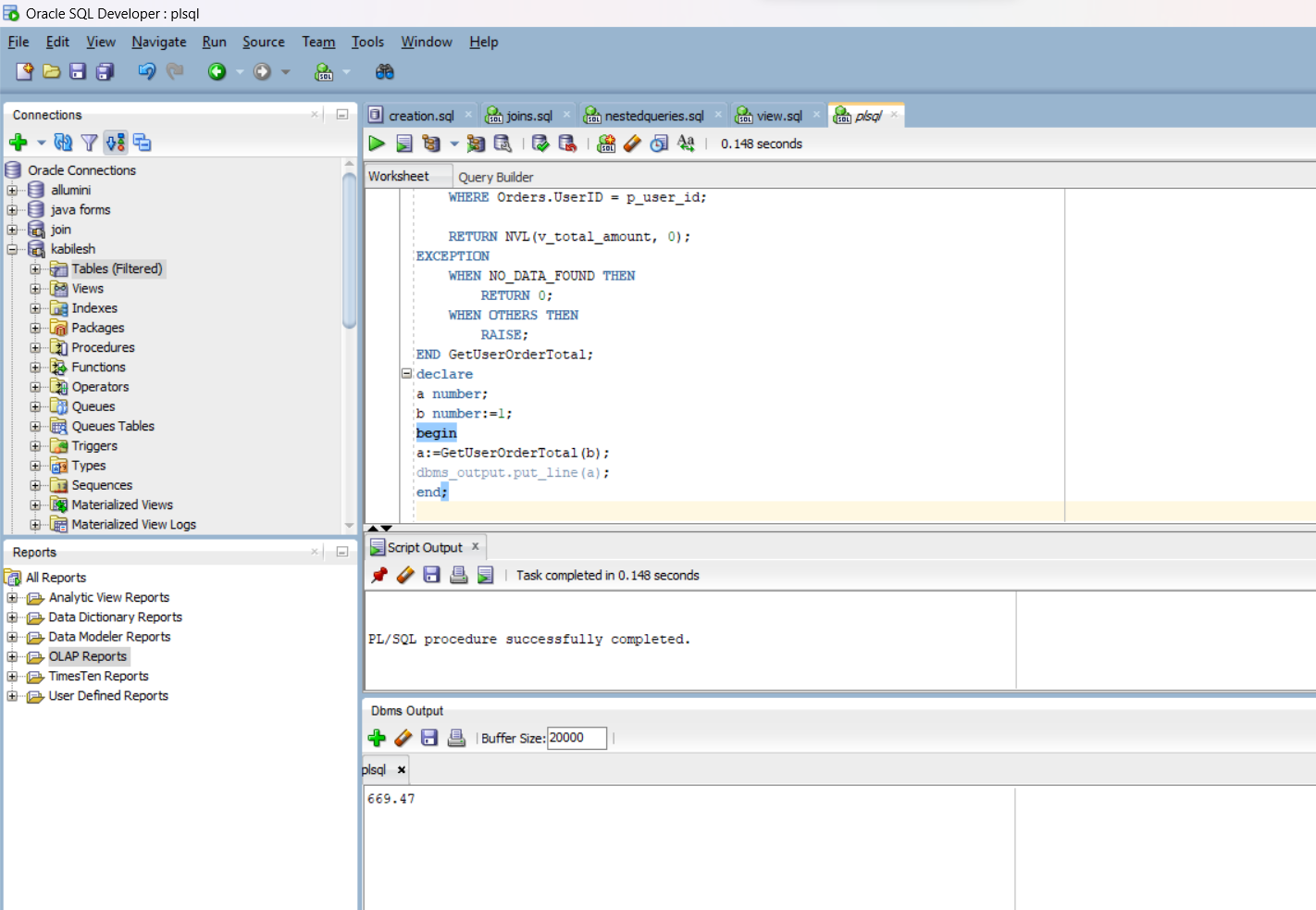
**<<Can be more than one PL/SQL but should have function>>**

|  |  |
| --- | --- |
| **Function Name** | GetUserOrderTotal |
| **Procedure Name** | getuserordertotal |
| **Expected Output** | Total order amount for the specified user. |
| **Tables operated** | Orders table |

**-- Sql query file name<<plsql.sql>>**

**Screenshot:**

**<<Place outputs of execution>>**

****

**Inference:**

**<<Explain what you would like to explain about the output>>**

You can call this function to get the total order amount for a specific user.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

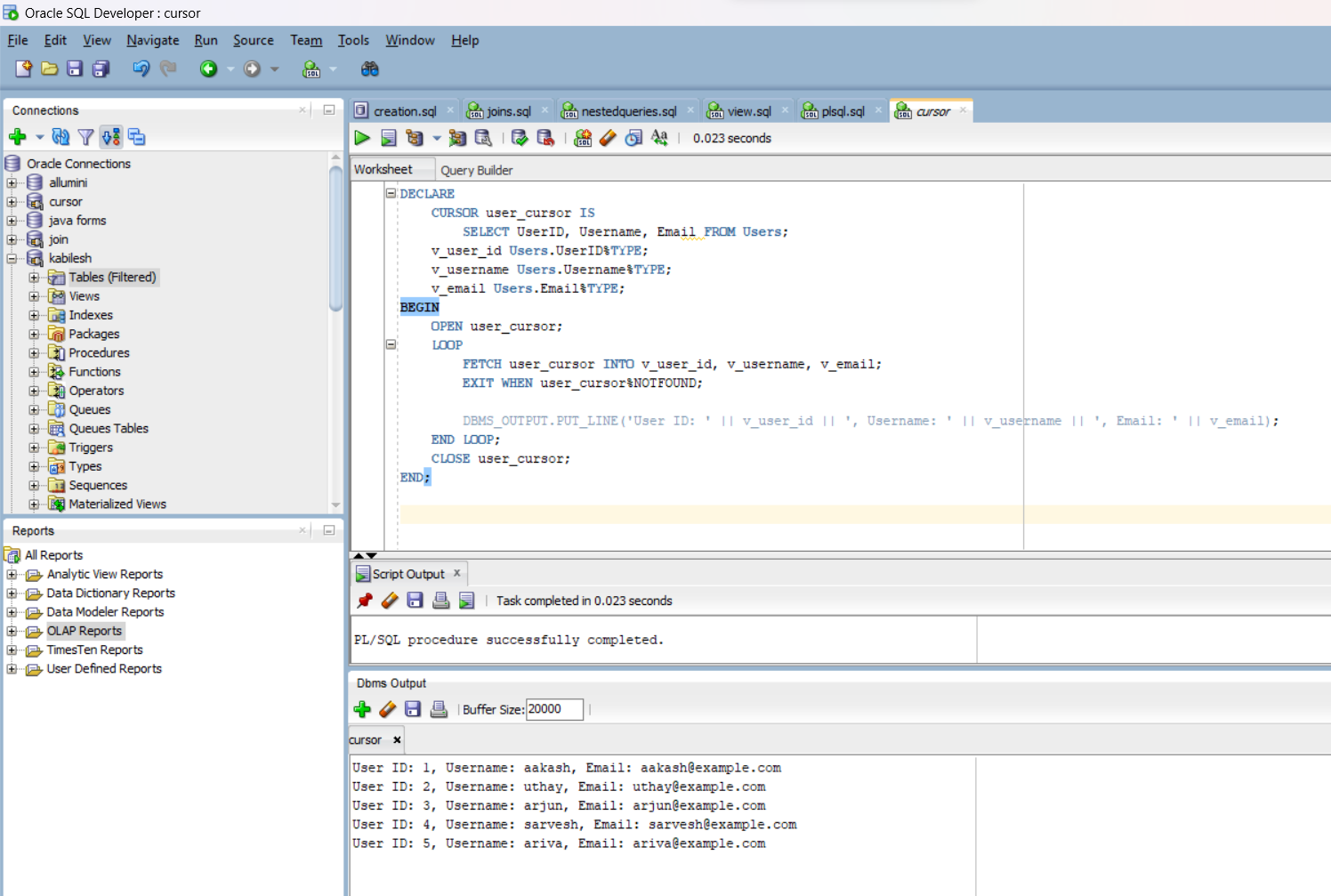
**7. Section-7(Cursor file)**

**<<Can be more than one cursor file>>**

|  |  |
| --- | --- |
| **Cursor Name** | **User\_cursor** |
| **…** | **…** |
| **Expected Output** | Value for each row in the user\_table table. |

**-- Sql query file name<<cursor.sql>>**

**Screenshot:<<Place outputs of execution>>**

****

**Inference:**

**<<Explain what you would like to explain about the output>>**

Value for each row in the user\_table table.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**8. Section-8(Trigger file)**

**<<Can be more than one trigger operation>>**

|  |  |
| --- | --- |
| **Cursor Name** | **Before insert review** |
| **Expected Output** | The next id for review will be given |

**-- Sql query file name<<Trigger.sql>>**

**Screenshot:<<Place outputs of execution>>**

****

**Inference:<<Explain what you would like to explain about the output:**

When a new row is inserted into the Reviews table, the trigger automatically assigns a unique ReviewID to that row using the sequence.

(i.e) when a reviewID is not given while inserting it automatically assigns.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Section -9**

**<< Web Application>>**

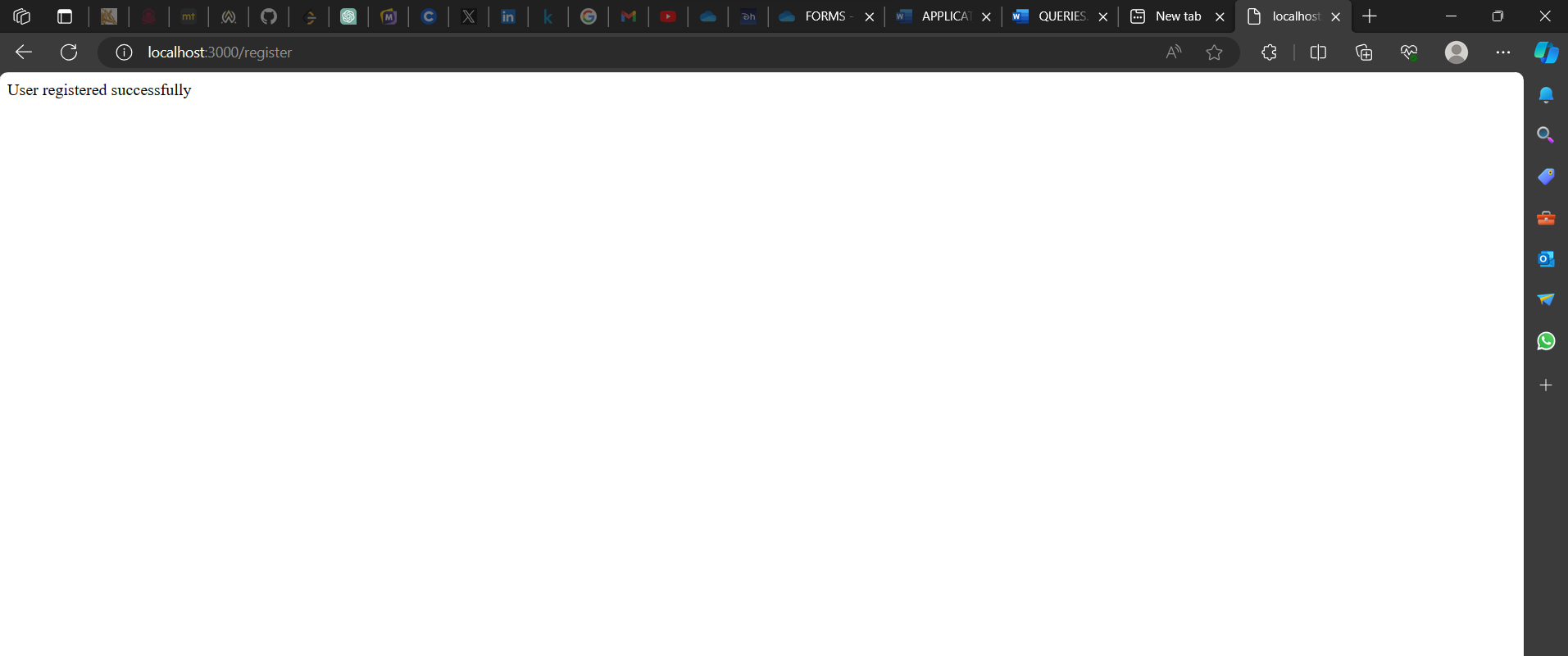
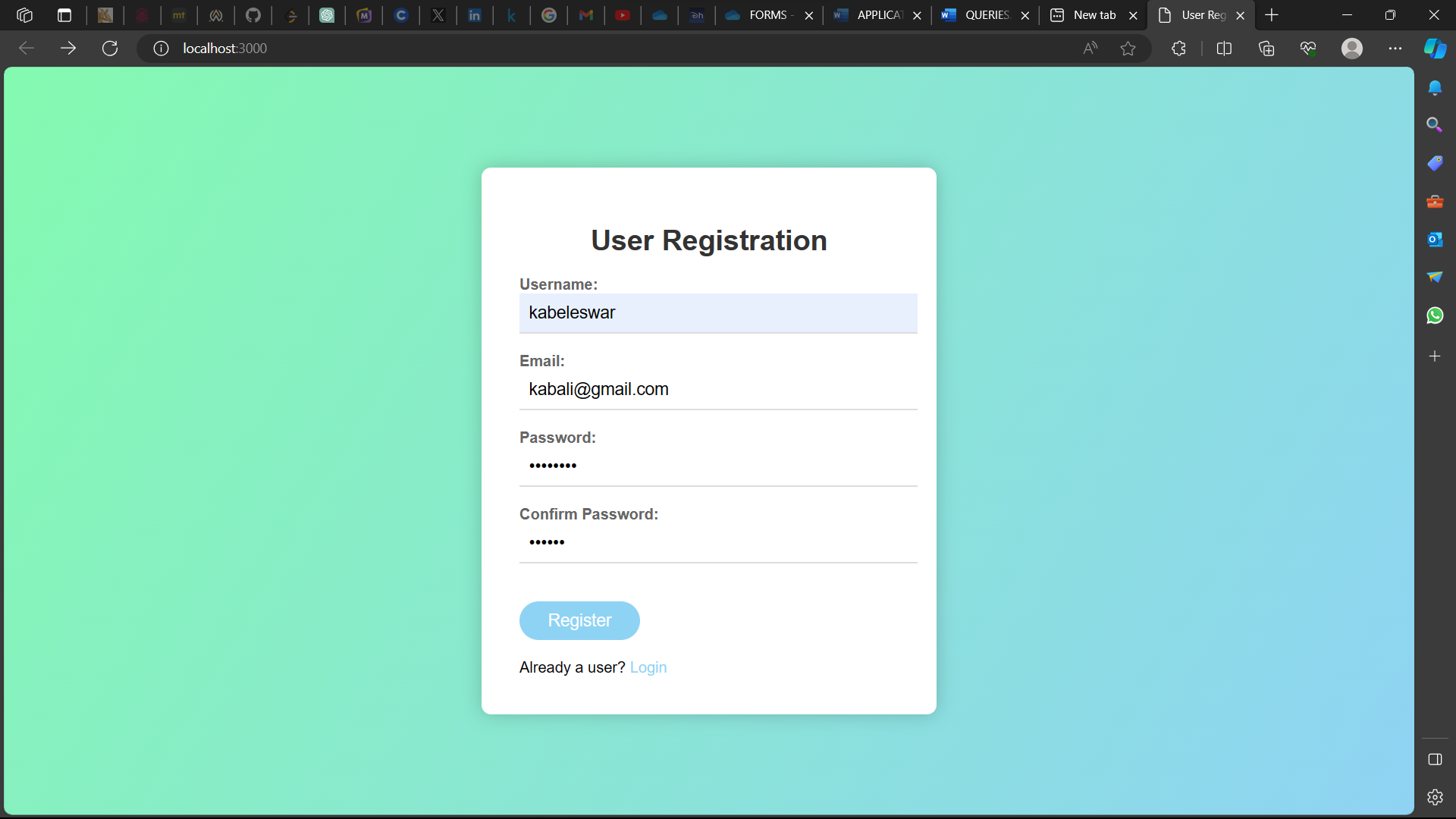
* **5 Master Tables**
* **6 Transaction table**

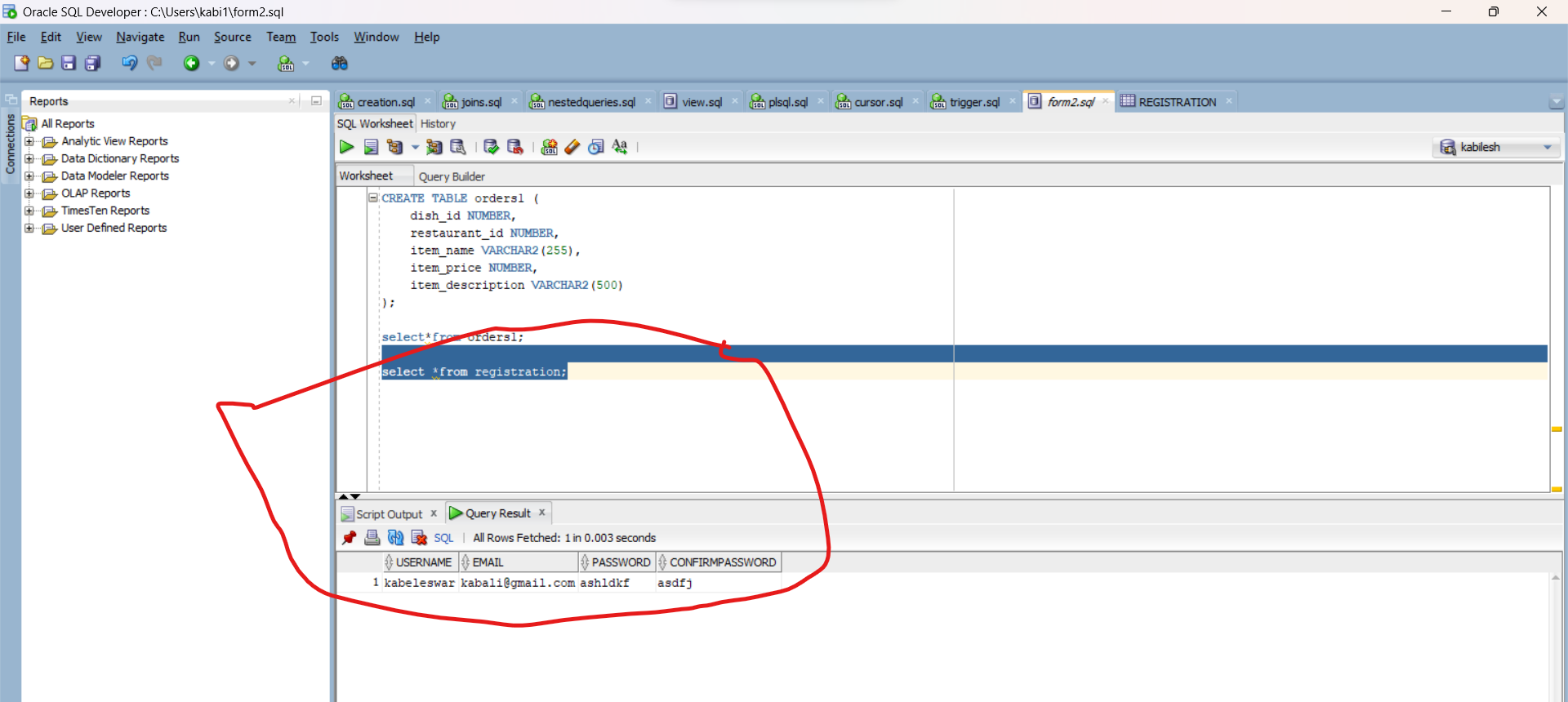
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Form Name** | **Category**  **(Master**  **/Transaction)** | **Table**  **Name associated**  **with**  **the form** | **Form Name/css**  **For ex: Emp\_master.html**  **Empmaster.css** | **File Name for reference in the shared drive** | **Type of operations**  **(Insert/Update/Delete**  **/Search/Display)** |
| Login | Master | Users | User\_  registration.html | User\_  registration.html | Insert |
| Home | Master | Reviews,orders,  menuItems,  delivery,addreses  menu,delivery  address,restaurants | Home.html | Home.html | Update , insert ,display,search |
| profile | Master,transaction | Users,payments,favorites | Profile.html | Profile.html | Display , update |
| restaurants | Master,transaction | Menu,reviews,restaurants | Restaurants.html | Restaurants.html | Insert,update,display |
| order | transaction | Delivery details,orders | Order.html | Order.html | Update , insert |
| Outfor  delivery | transaction | Delivery drivers,  adresses | Delivery.html | Delivery.html | Display, update ,search |

**Operations :**

**1.Insert :** Insertion of user details

**form 1: <<keep the screenshot for the form operations for all the forms>>**





**Inference: -Write 1-2 sentence explaining the operation shown:**

When I create a new user with registration form, the page shows “user registered successfully” and when we check it in sql developer the data is inserted automatically.

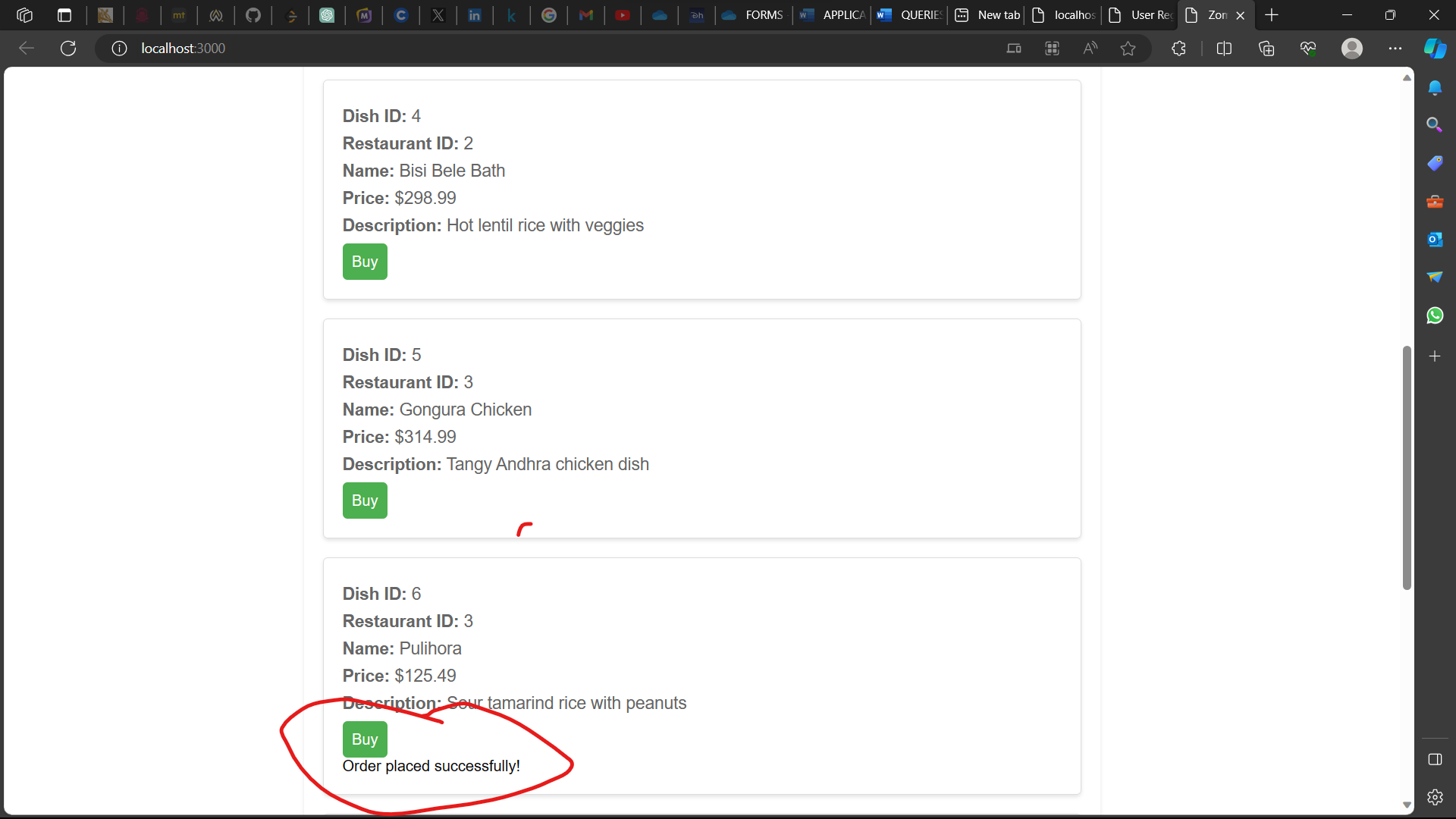
“video file is uploaded for new updation named **“form1.mp4”**

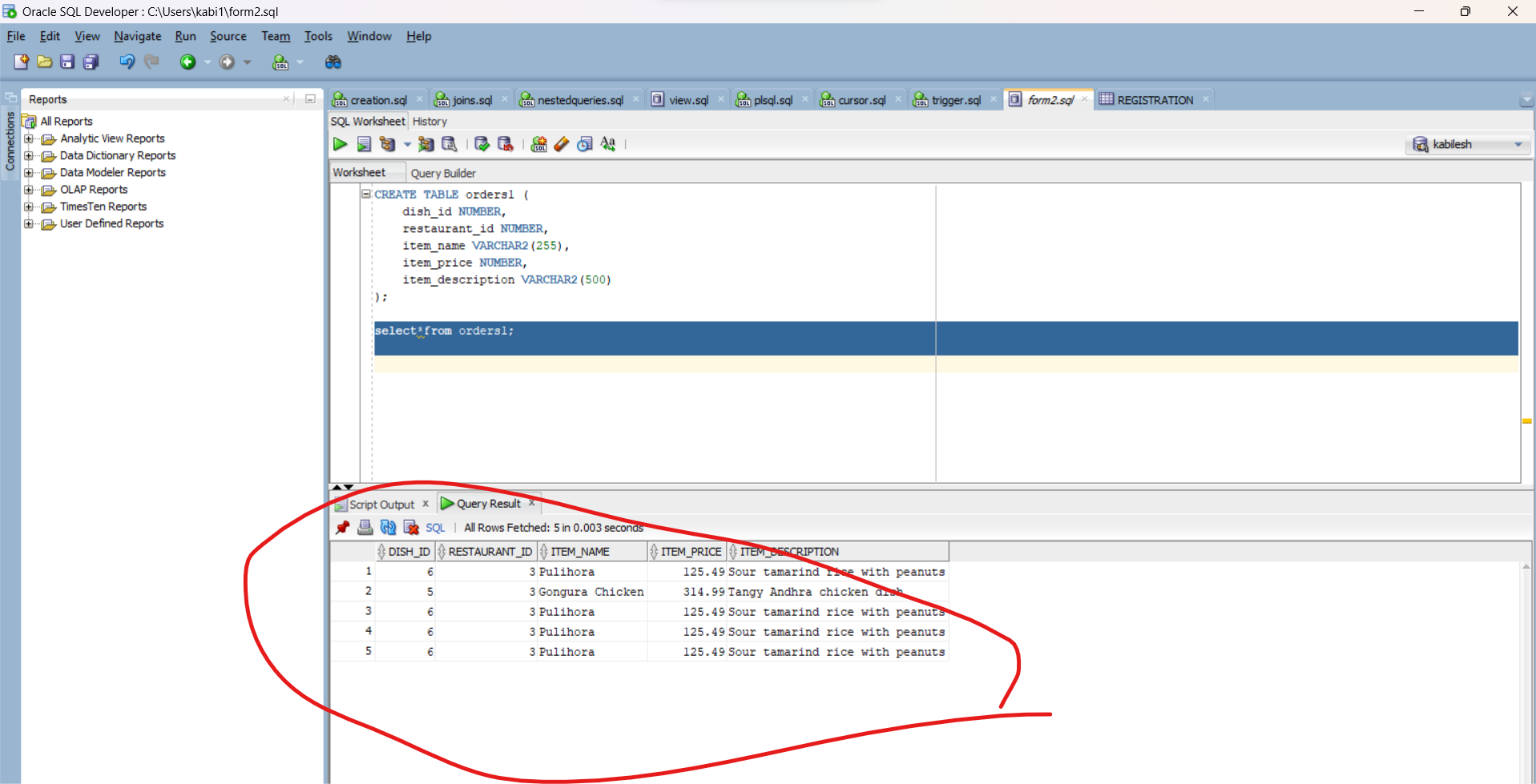
**-- record the video of the working output and keep in the drive**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Form 2: inserting ordered dish “buy” in backend.**

**Connecting second form:**

****

****

**Inference: --Write 1-2 sentence explaining the operation shown:**

When I click “buy” order is placed and data is stored in backend.

**<<video recording file :>>**

Video file is uploaded for new updation named **“form2.mp4**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**10.Section-10(No-SQL Application)**

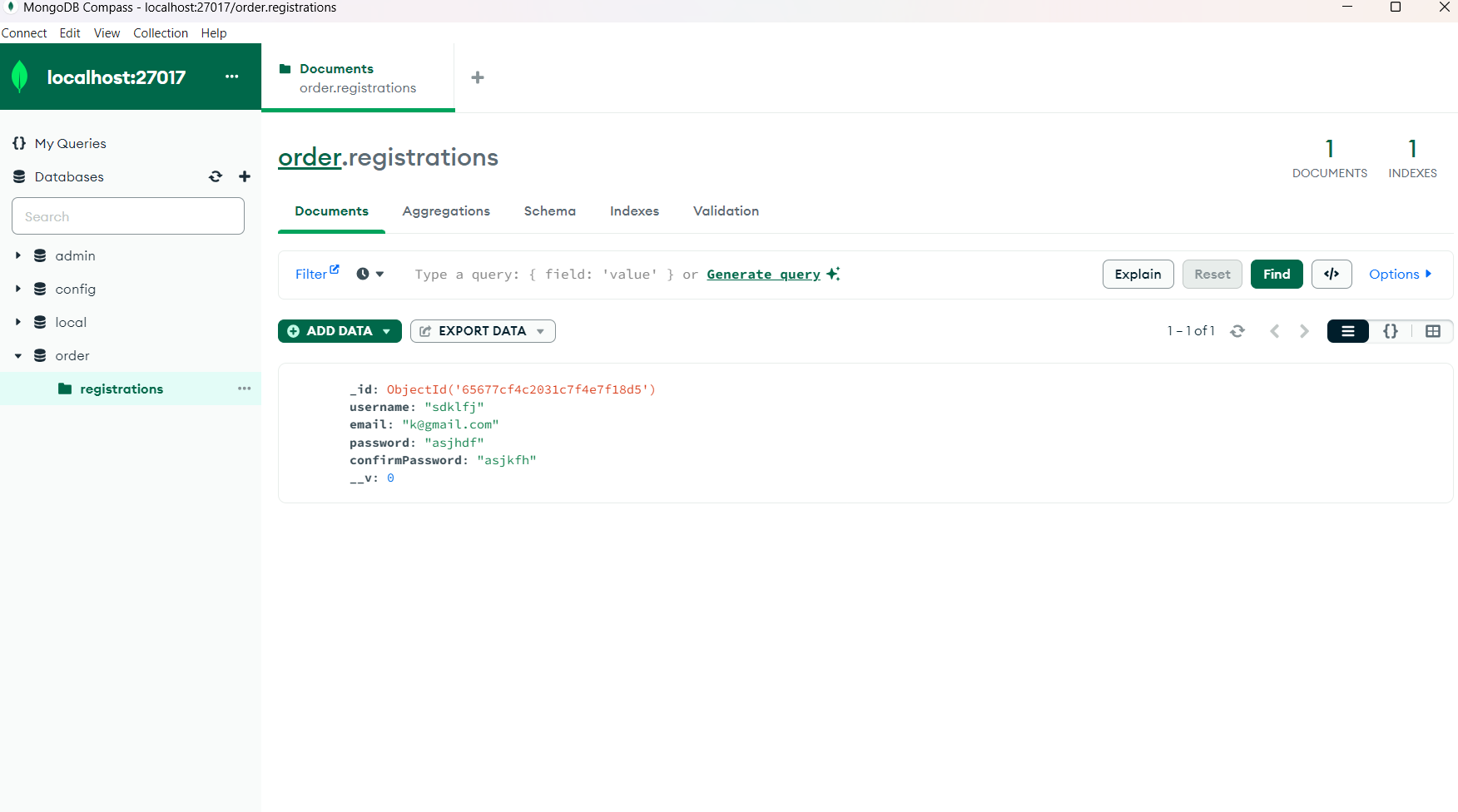
**-- Can show for one Table alone**

**Technologies:** mongodb, mongoDBcompass

|  |  |
| --- | --- |
| **Front End** | **Html 5 , css** |
| **Back End** | **Javascript,mongoDB** |
| **Editor** | **Visual studio code** |
| **Language** | **Javascript** |
| **Framework** | **Node js** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Form Name** | **Category**  **(Master/**  **Transaction)** | **Table Name associated with the form** | **Form Name /css**  **For ex: Emp\_master.html**  **Emp\_master.css** | **File Name for reference in the shared drive** | **Type of operations**  **(Insert/Update/Delete**  **/Search/Display)** |
| **user** | **master** | **User\_registration** | **User\_reg.html**  **User\_reg.js** | **Mongodb\_files** | **Insertion,updation** |

**Screenshot:**

****

**-- record the video of the working output and keep in the drive**

**Video file named: “mongo.mp4”**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

|  |  |  |
| --- | --- | --- |
| **Section No** | **Mark** | **Marks Awarded** |
| **1** | **5** |  |
| **2** | **10** |  |
| **3** | **5** |  |
| **4** | **5** |  |
| **5** | **5** |  |
| **6** | **10** |  |
| **7** | **10** |  |
| **8** | **10** |  |
| **9** | **30** |  |
| **10** | **10** |  |