Task 1:

**Let’s assume we’re building an employee management system. The relationships can be modeled as follows:**

**- \*Tables:**

- Employees

- Departments

- Projects

- Employee\_Project\_Assignments (This table shows which employee is working on which project).

**Diagram:**

Here's a quick description of how we structured our diagram:

**- Employees Table:** Includes EmployeeID (Primary Key), FirstName, LastName, DepartmentID (Foreign Key).

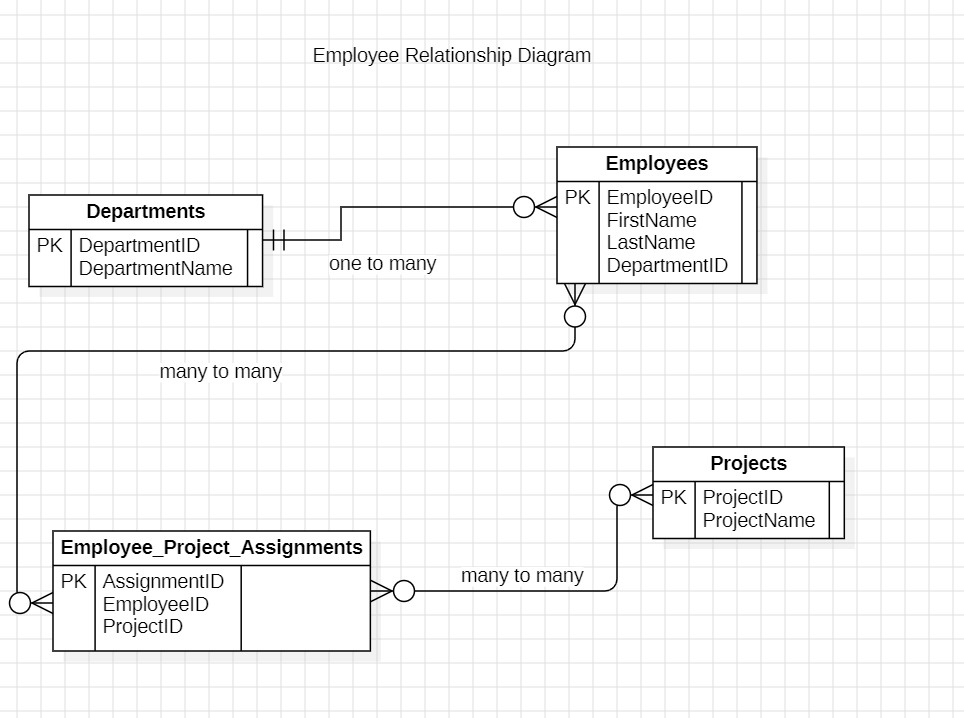
- **Departments Table**: Includes DepartmentID (Primary Key), DepartmentName.

- **Projects Table:** Includes ProjectID (Primary Key), ProjectName.

- **Employee\_Project\_Assignments Table**: Includes AssignmentID (Primary Key), EmployeeID (Foreign Key), ProjectID (Foreign Key).

**- One-to-many:** Departments → Employees

- **Many-to-many:** Employees ↔ Projects (via Employee\_Project\_Assignments)



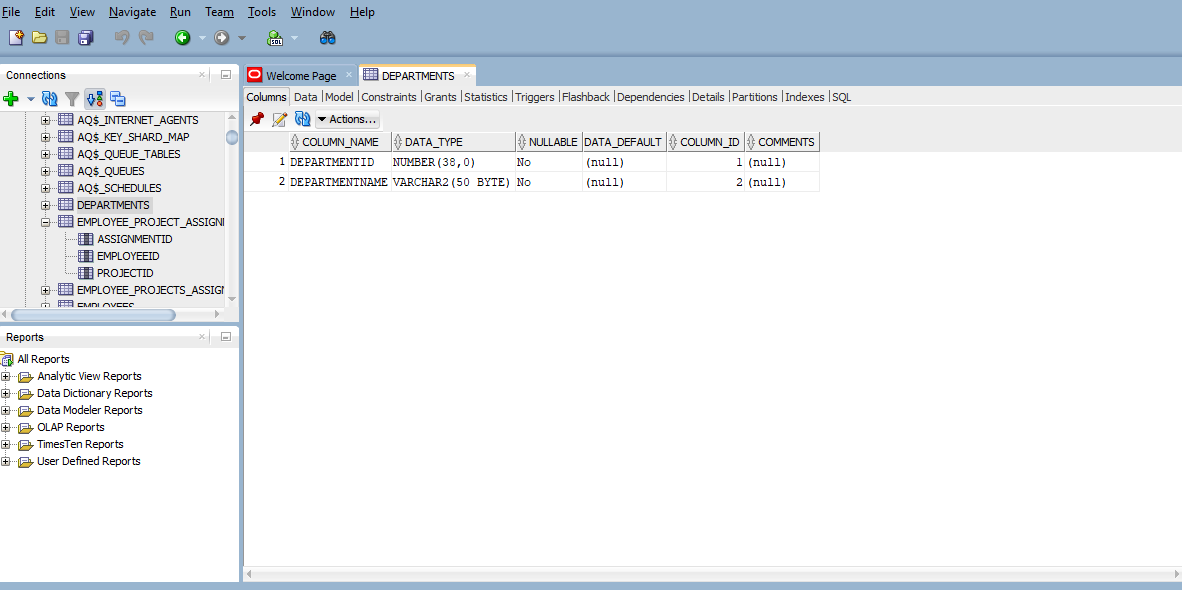
**Task 2:**

CREATE TABLE Departments (

DepartmentID INT PRIMARY KEY,

DepartmentName VARCHAR(50) NOT NULL

);



CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

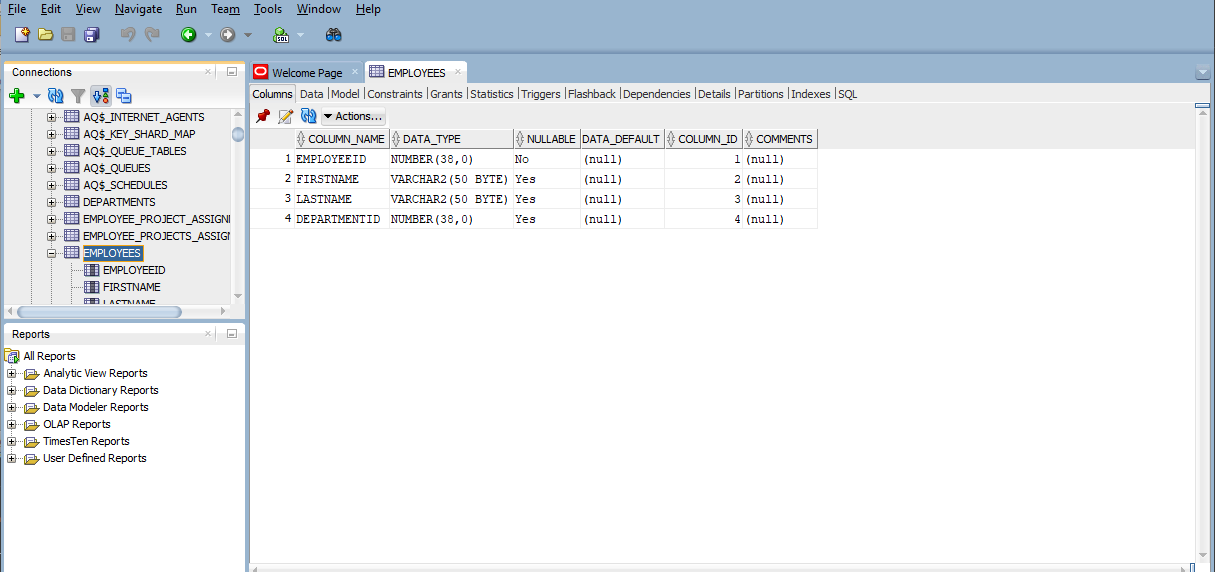
FirstName VARCHAR(50),

LastName VARCHAR(50),

DepartmentID INT,

FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID)

);



-- Create Employee\_Project\_Assignments table (to link employees with projects)

CREATE TABLE Employee\_Projects\_Assignments (

AssignmentID INT PRIMARY KEY,

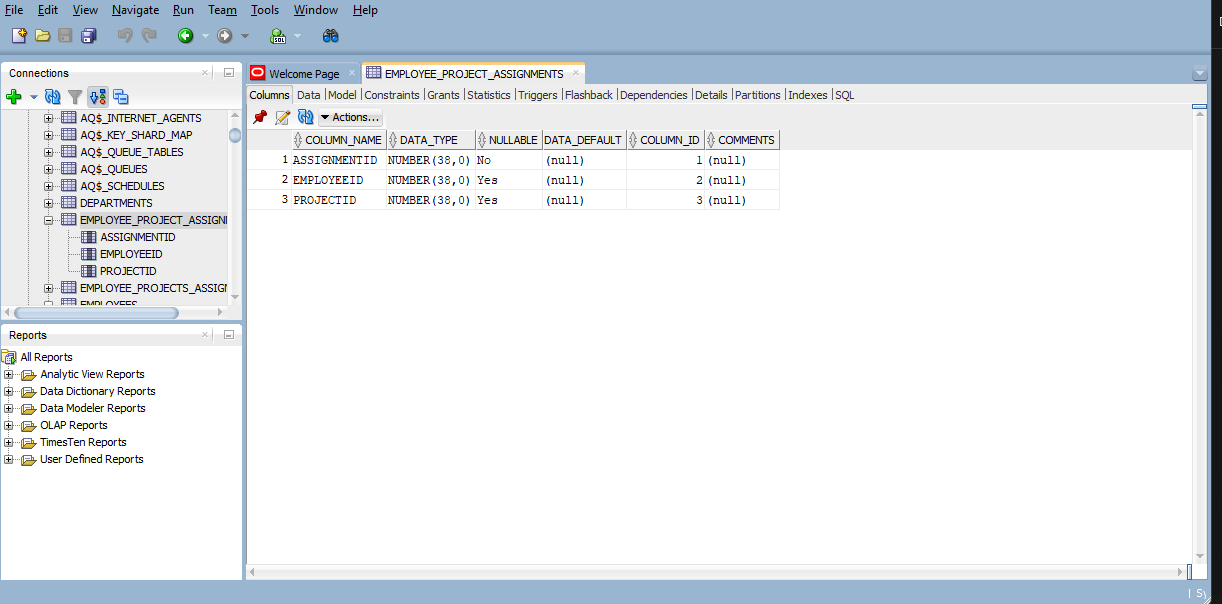
EmployeeID INT,

ProjectID INT,

FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID),

FOREIGN KEY (ProjectID) REFERENCES Projects(ProjectID)

);



INSERT INTO Departments (DepartmentID, DepartmentName)

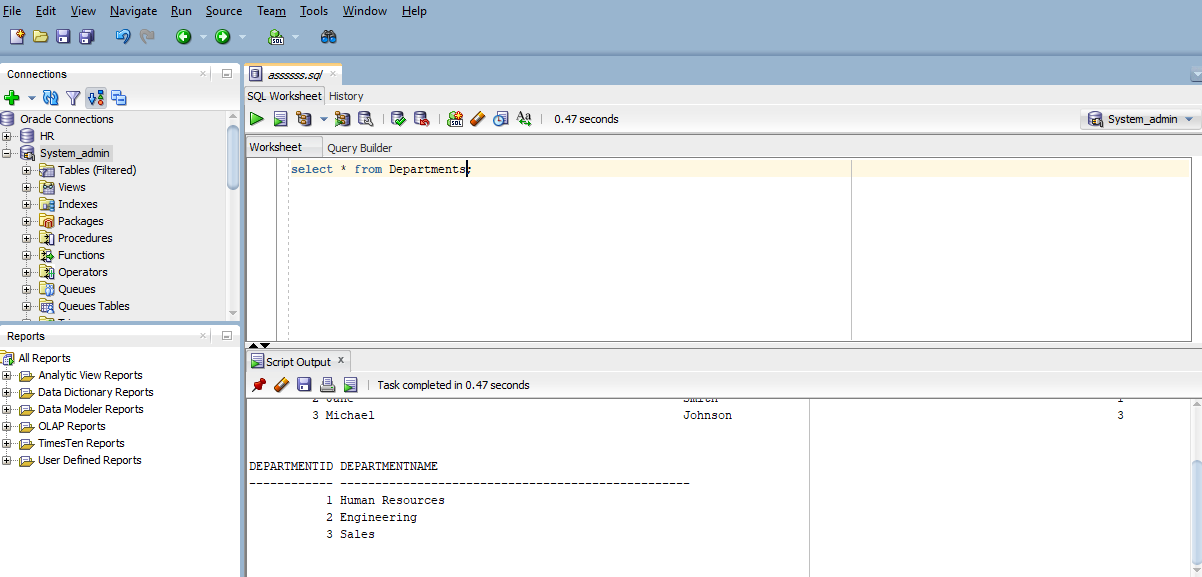
VALUES (1, 'Human Resources');

INSERT INTO Departments (DepartmentID, DepartmentName)

VALUES (2, 'Engineering');

INSERT INTO Departments (DepartmentID, DepartmentName)

VALUES (3, 'Sales');



INSERT INTO Employees (EmployeeID, FirstName, LastName, DepartmentID)

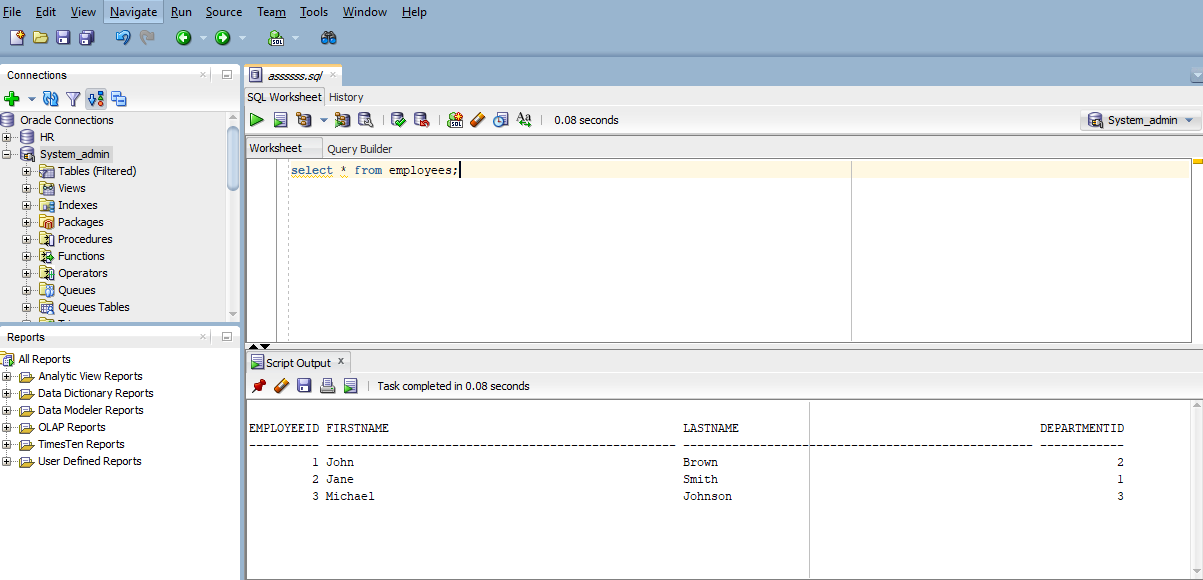
VALUES (1, 'John', 'Doe', 2);

INSERT INTO Employees (EmployeeID, FirstName, LastName, DepartmentID)

VALUES (2, 'Jane', 'Smith', 1);

INSERT INTO Employees (EmployeeID, FirstName, LastName, DepartmentID)

VALUES (3, 'Michael', 'Johnson', 3);



CREATE TABLE Projects (

ProjectID INT PRIMARY KEY,

ProjectName VARCHAR(100)

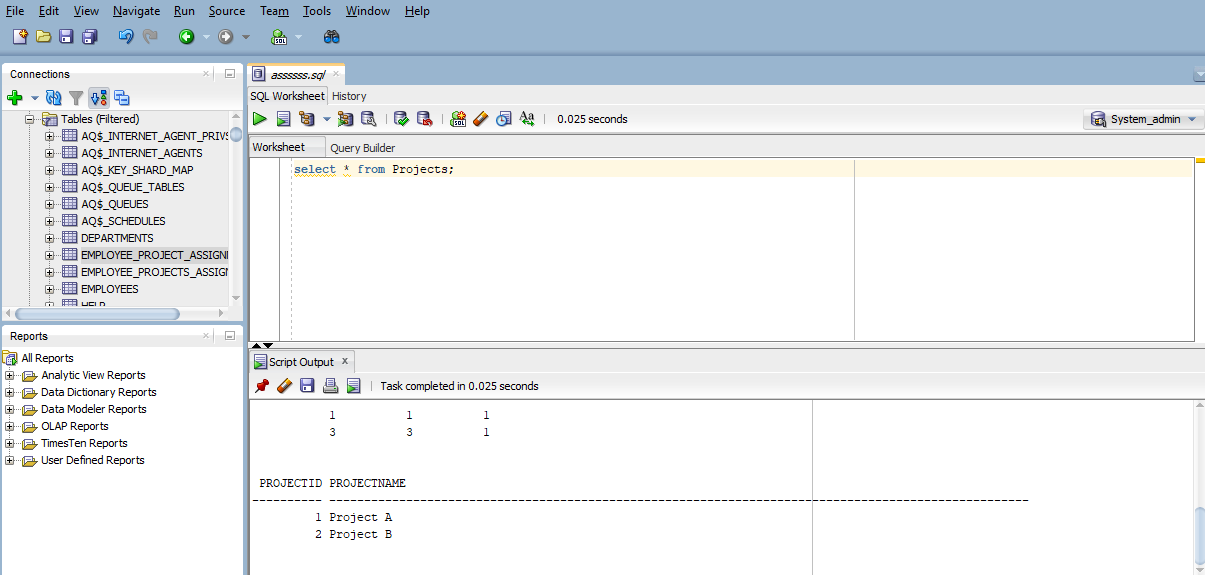
);

INSERT INTO Projects (ProjectID, ProjectName)

VALUES (1, 'Project A');

INSERT INTO Projects (ProjectID, ProjectName)

VALUES (2, 'Project B');



INSERT INTO Employee\_Project\_Assignments (AssignmentID, EmployeeID, ProjectID)

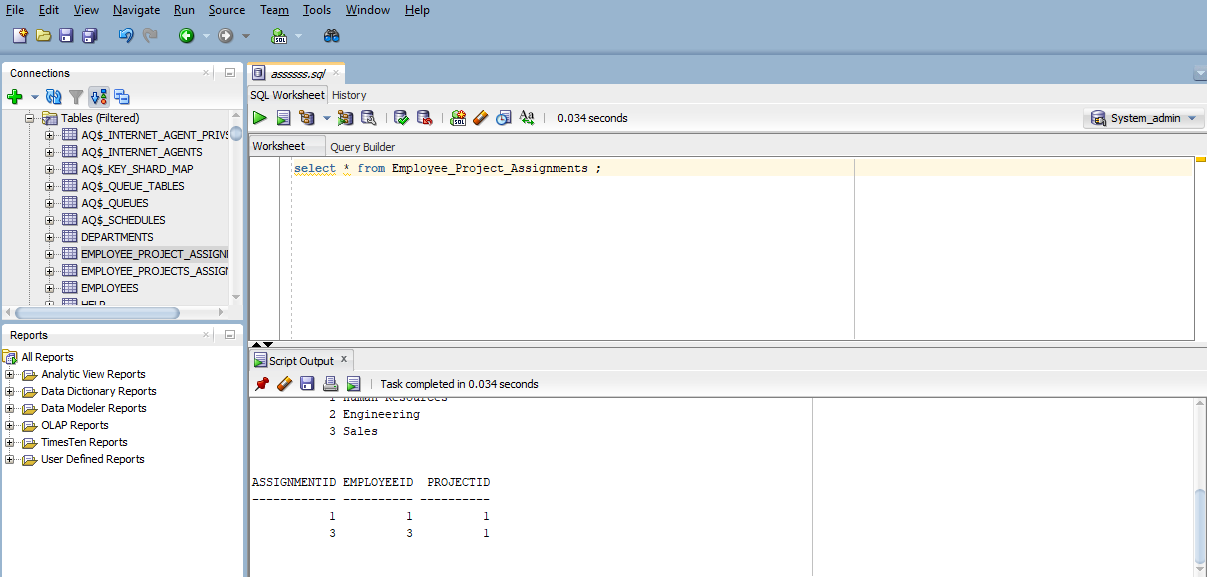
VALUES (1, 1, 1);

INSERT INTO Employee\_Project\_Assignments (AssignmentID, EmployeeID, ProjectID)

VALUES (2, 2, 2);

INSERT INTO Employee\_Project\_Assignments (AssignmentID, EmployeeID, ProjectID)

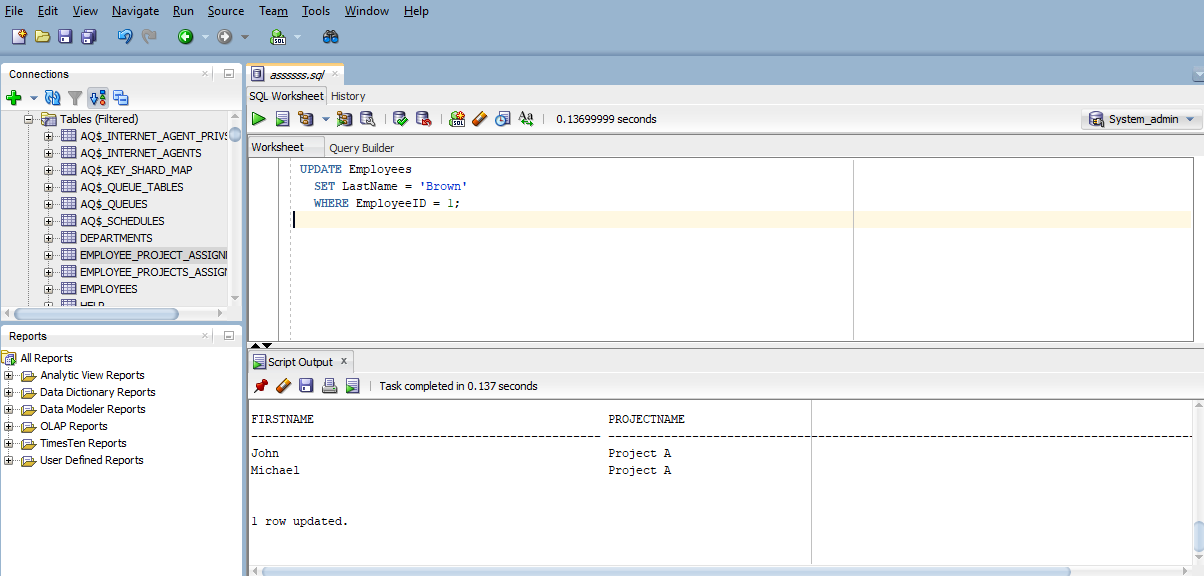
VALUES (3, 3, 1);



UPDATE Employees

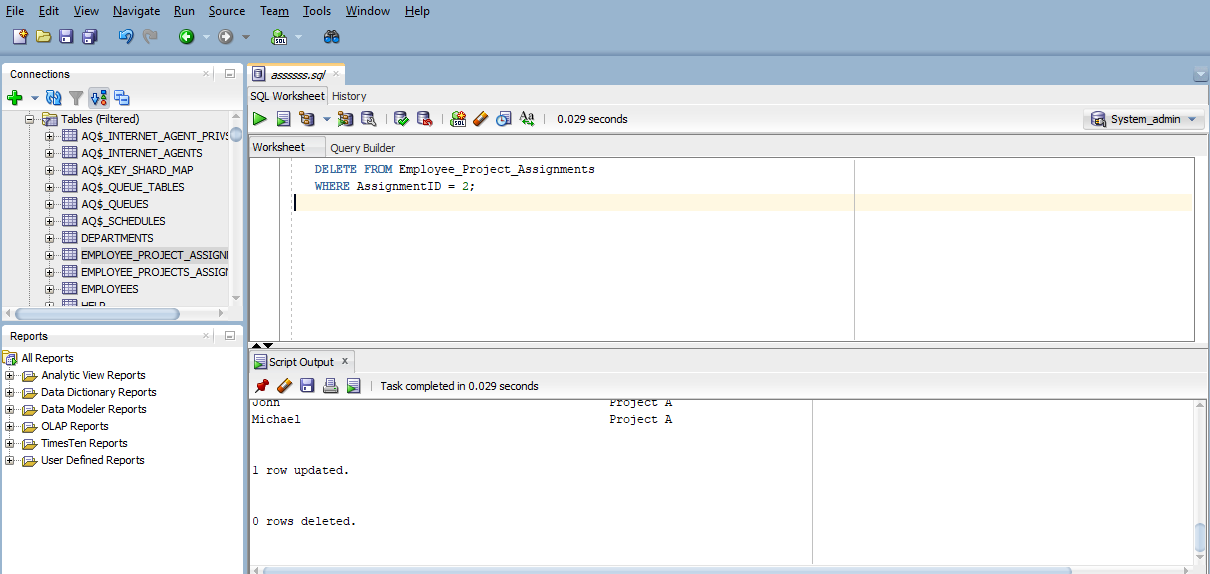
SET LastName = 'Brown'

WHERE EmployeeID = 1;



DELETE FROM Employee\_Project\_Assignments

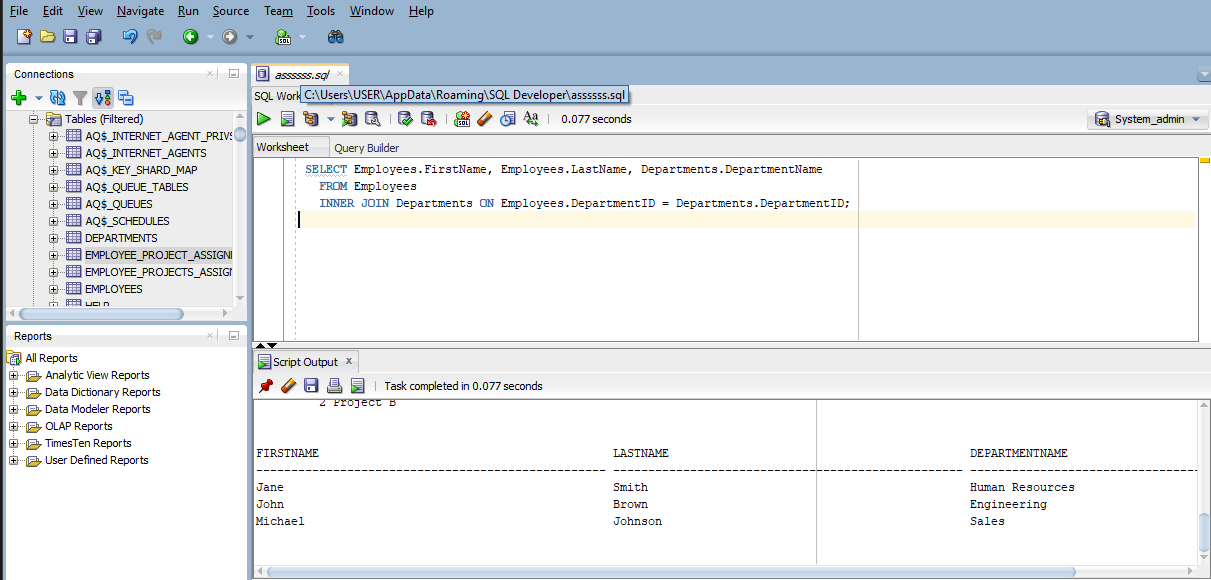
WHERE AssignmentID = 2;



SELECT Employees.FirstName, Employees.LastName, Departments.DepartmentName

FROM Employees

INNER JOIN Departments ON Employees.DepartmentID = Departments.DepartmentID;

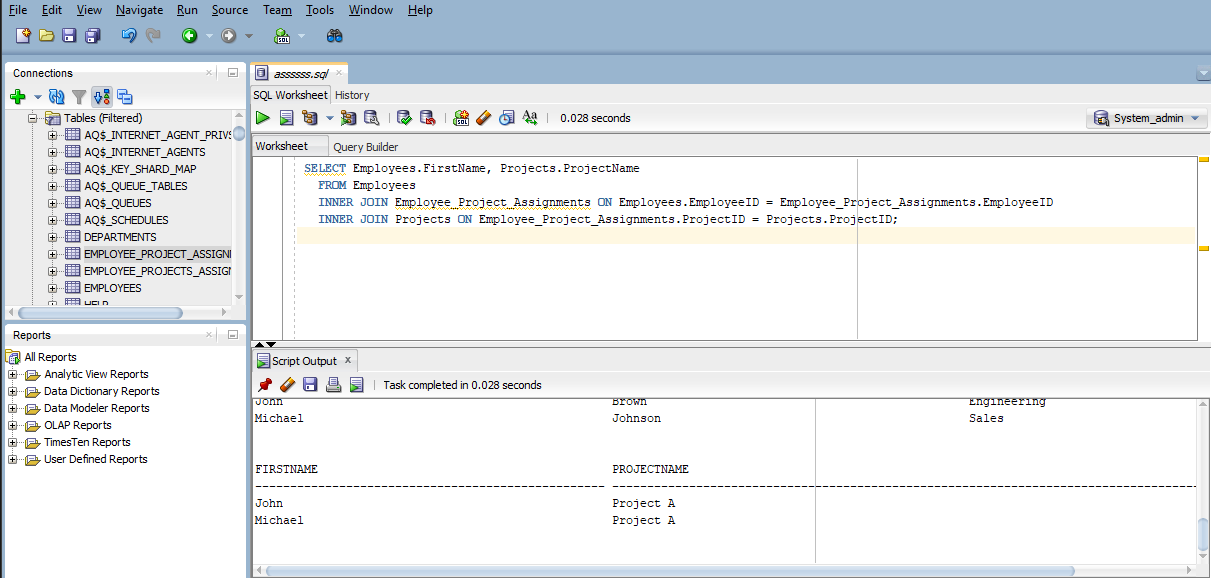


SELECT Employees.FirstName, Projects.ProjectName

FROM Employees

INNER JOIN Employee\_Project\_Assignments ON Employees.EmployeeID = Employee\_Project\_Assignments.EmployeeID

INNER JOIN Projects ON Employee\_Project\_Assignments.ProjectID = Projects.ProjectID;



**1. DDL (Data Definition Language)**

The DDL queries we already provided define the structure of the tables (Departments, Employees, Projects, and Employee\_Project\_Assignments) by creating them

**2. DML (Data Manipulation Language)**

DML commands are used for managing data within database objects. They allow you to retrieve, insert, update, and delete data as we did above. Common DML operations include:

SELECT: Retrieves data from one or more tables.

INSERT: Adds new rows to a table.

UPDATE: Modifies existing data in a table.

DELETE: Removes rows from a table.

**3. DCL (Data Control Language)**

DCL commands are used to control access to data within the database. They define permissions and security levels for database users. Common DCL operations include:

GRANT: Gives users access privileges to database objects.

REVOKE: Removes access privileges from users. By using these codes

-- Grant select and insert permissions on the Employees table to user 'admin'

**GRANT SELECT, INSERT ON Employees TO 'admin';**

-- Revoke insert permission from user 'admin' on the Employees table

**REVOKE INSERT ON Employees FROM 'admin';**

**4. TCL (Transaction Control Language)**

TCL commands are used to manage transactions within a database. They ensure data integrity by controlling how transactions are processed. Common TCL operations include:

**COMMIT:** Saves all changes made in the current transaction.

**ROLLBACK:** Undoes changes made in the current transaction if there was an error.

**SAVEPOINT:** Sets a point in a transaction to which you can later roll back. By using these codes

-- Start a transaction

**BEGIN TRANSACTION;**

-- Insert new department data

**INSERT INTO Departments (DepartmentID, DepartmentName)**

**VALUES (4, 'Marketing');**

-- Commit the transaction to save changes

**COMMIT;**

-- Rollback in case of error

**ROLLBACK;**

-- Use a savepoint to mark a point in the transaction

**SAVEPOINT BeforeInsert;**

-- Insert more data

**INSERT INTO Departments (DepartmentID, DepartmentName)**

**VALUES (5, 'Legal');**

-- Rollback to savepoint if necessary

**ROLLBACK TO BeforeInsert;**