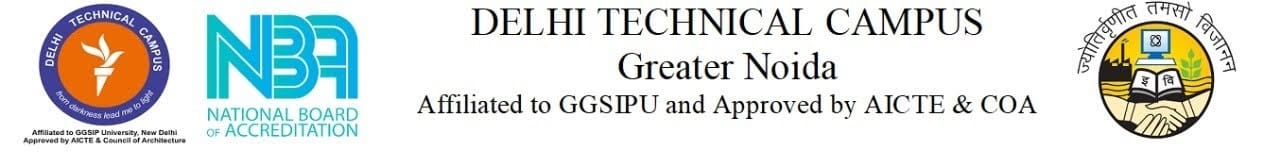
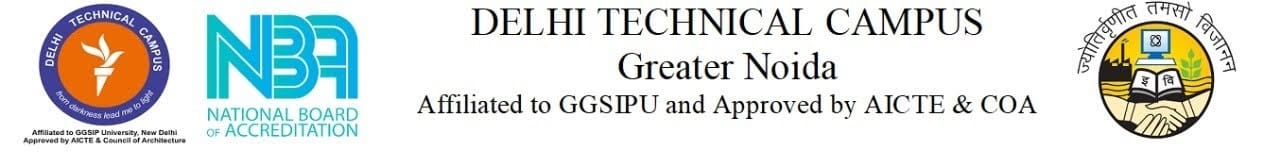
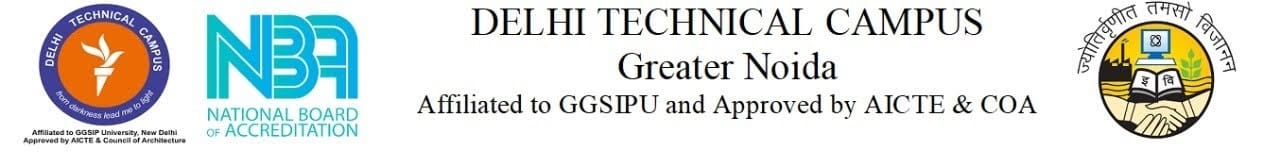
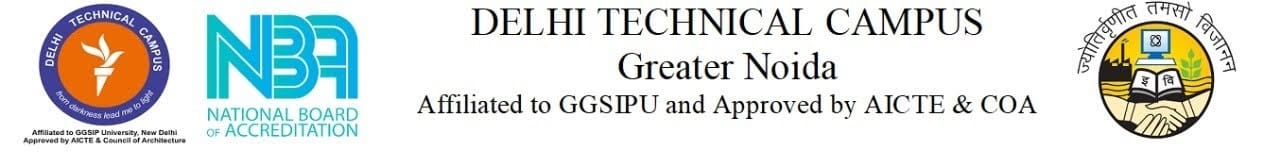
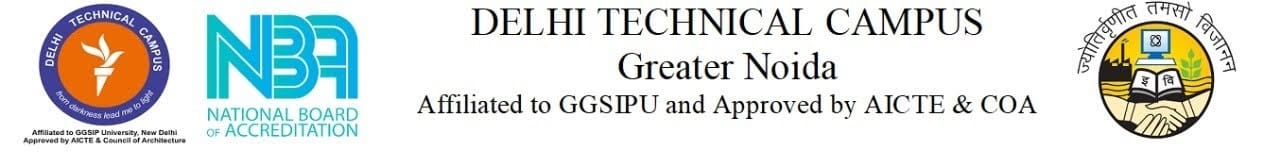
****PRACTICAL FILE

**SESSION:2024-25**

**DBMS Lab (CIC‐256)**

**II Year, 4th Sem**

****

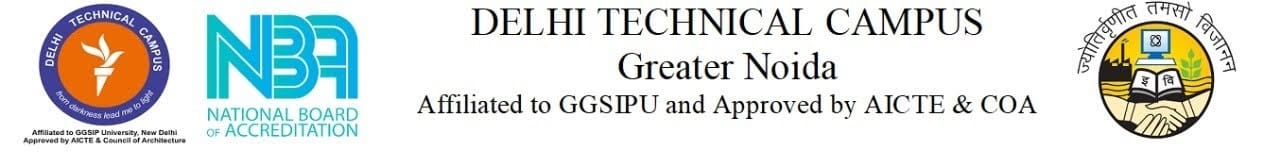
**Submitted to: Submitted by:**

**Name:** Dr. Upasna Joshi  **Name:** Saksham Mendiratta

**Designation:** Assistant Professor **Enrollment No:** 04118002723

Department of Computer Science and Engineering

Delhi Technical Campus, Greater Noida

**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No** | **Experiment** | **Date of Conduction** | **Date of Submission** | **Signature** |
| 1. | Introduction to DBMS, RDBMS and SQL. |  |  |  |
| 2. | Experiments based on DDL commands – CREATE, ALTER, DROP and TRUNCATE. |  |  |  |
| 3. | Apply the integrity constraints like Primary Key, Foreign key, Check, NOT NULL, etc. to the tables. |  |  |  |
| 4. | Experiments based on basic DML commands – SELECT, INSERT, UPDATE and DELETE. |  |  |  |
| 5. | Write the queries for implementing Built-in functions, GROUP BY, HAVING and ORDER BY. |  |  |  |
| 6. | Write the queries to implement the joins and subqueries |  |  |  |
| 7. | Write the queries to implement the set operations and also create a views based on views. |  |  |  |
| 8. | Demonstrate the concept of Control Structures also demonstrate the concept of Exception Handling. |  |  |  |
| 9. | Create employee management system using MongoDB. |  |  |  |
| 10. | Connect employee management system using JDBC. |  |  |  |
| 11. | Executing Queries using NoSQL(document-based) |  |  |  |

**Experiment-1**

**Title:-** Introduction to DBMS, RDBMS and SQL.

**Objective:-**

* To understand the concepts of Database Management Systems (DBMS) and Relational Database Management Systems (RDBMS).
* To learn the basics of Structured Query Language (SQL).
* To differentiate between DBMS and RDBMS.

**Theory:-**

**DBMS (Database Management System)** is a software system designed to manage databases. It allows users to store, retrieve, and manipulate data in an organized way. A DBMS provides tools for creating, reading, updating, and deleting data (CRUD operations), as well as managing security, access control, and backups.

**Key features of DBMS:**

* Data storage and management.
* Data retrieval.
* Data integrity and security.
* Data concurrency control (handling multiple users).

Examples: Microsoft Access, Oracle, MySQL, SQLite (when not using relational features).

**RDBMS (Relational Database Management System)** is a type of DBMS that stores data in a structured format using rows and columns (tables). It follows the relational model, where data is stored in tables, and relationships can be established between these tables using keys (primary and foreign keys).

**Key features of RDBMS:**

* Tables (relations) with rows (tuples) and columns (attributes).
* Data is stored in a tabular form.
* Supports SQL (Structured Query Language) for querying and managing data.
* Allows data integrity and relationships using primary and foreign keys.
* Ensures ACID properties (Atomicity, Consistency, Isolation, Durability).

Examples: MySQL, PostgreSQL, Oracle Database, Microsoft SQL Server.

|  |  |  |
| --- | --- | --- |
| Feature | DBMS | RDBMS |
| Data Storage | Data is stored in files (not necessarily structured). | Data is stored in tables (structured format). |
| Data Model | Supports various models (hierarchical, network, etc.). | Follows the relational model (tables). |
| Normalization | Not necessarily supported. | Data normalization is supported to reduce redundancy. |
| Relationships | Does not support relationships between tables. | Supports relationships using primary and foreign keys. |
| Query Language | Supports simple query languages (e.g., File-based queries). | Uses SQL to query and manage data. |
| Data Integrity | Basic checks (optional). | Stronger integrity constraints (primary keys, foreign keys, etc.). |
| ACID Properties | Not strictly implemented. | Supports ACID properties for transaction management. |
| Multi-user Support | May or may not support multiple users. | Supports multiple users with concurrent access. |

In an **RDBMS (Relational Database Management System)**, a **table** is a collection of related data organized in a structured format using **rows** and **columns**. Each table represents a specific entity or concept within the database, such as customers, orders, products, etc. The table is the fundamental building block where all the data is stored.

**Key Characteristics of a Table in RDBMS:-**

1. **Rows (Records/tuples)**:
   * Each row in a table represents a single record or instance of the entity.
   * For example, in a table of "Employees," each row could represent a single employee with all their details (ID, name, salary, etc.).
2. **Columns (Attributes/fields)**:
   * Columns represent the characteristics or attributes of the entity.
   * For example, in the "Employees" table, columns could include **EmployeeID**, **Name**, **Department**, **Salary**, etc.
   * Each column has a defined **data type** (e.g., INTEGER, VARCHAR, DATE) that determines what kind of data it can store.
3. **Primary Key**:
   * A primary key is a unique identifier for each row in the table.
   * For instance, in the "Employees" table, **EmployeeID** could be the primary key because each employee will have a unique ID.
4. **Foreign Key**:
   * A foreign key is a column that creates a relationship between two tables. It links to the primary key of another table.
   * For example, in an "Orders" table, a **CustomerID** column may act as a foreign key linking back to the **CustomerID** primary key in the "Customers" table.
5. **Constraints**:
   * **Not Null**: Ensures that a column cannot have a NULL value.
   * **Unique**: Ensures that all values in a column are distinct.
   * **Check**: Enforces domain integrity by limiting the values a column can hold (e.g., age cannot be negative).
6. **Indexes**:
   * Indexes can be created on columns to speed up data retrieval and query performance. For example, an index on the **EmployeeID** column can make searches faster.

**Introduction to SQL (Structured Query Language):-**

SQL (Structured Query Language) is a standard programming language used to manage and manipulate relational databases. It provides a powerful way to communicate with a database, allowing users to create, retrieve, update, and

delete data. SQL is widely used by developers, database administrators, and data analysts for tasks ranging from simple data retrieval to complex reporting and data analysis.

**Key Features of SQL:**

1. **Data Querying**: SQL is primarily used for querying and retrieving data from databases using the SELECT statement.
2. **Data Manipulation**: SQL allows for inserting new data (INSERT), updating existing data (UPDATE), and removing data (DELETE).
3. **Database Management**: SQL can be used to define and manage database schemas using CREATE, ALTER, and DROP commands.
4. **Data Integrity**: It supports enforcing constraints like PRIMARY KEY, FOREIGN KEY, UNIQUE, and NOT NULL to maintain data consistency and validity.
5. **Data Security**: SQL allows for controlling access to data using GRANT and REVOKE permissions, ensuring data security.

**Types of SQL Languages**

SQL is divided into five categories based on functionality:

1**. Data Query Language (DQL)** – Used for retrieving data.

* Command: SELECT

2. **Data Definition Language (DDL)** – Defines the database structure.

* Commands: CREATE, ALTER, DROP, TRUNCATE

3. **Data Manipulation Language (DML)** – Modifies existing data.

* Commands: INSERT, UPDATE, DELETE

4. **Data Control Language (DCL)** – Controls user access to data.

* Commands: GRANT, REVOKE

5. **Transaction Control Language (TCL)** – Manages database transactions.

* Commands: COMMIT, ROLLBACK, SAVEPOINT

**SQL Process (Query Execution Process)**

When an SQL query is executed, it follows these steps:

1 .Query Parsing – SQL syntax is checked.

2 .Query Optimization – The best execution plan is selected.

3 .Query Execution – The database processes the query.

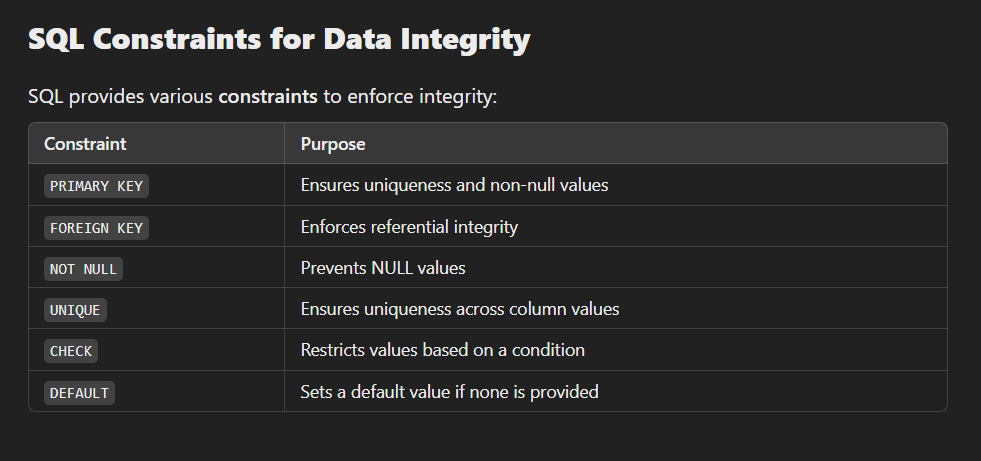
4 .Result Fetching – The final output is displayed to the user.

**Data Integrity in SQL:-**

Data integrity ensures accuracy, consistency, and reliability in a database. It is enforced using **constraints, transactions, and normalization**.

**Types of Data Integrity:**

1. **Entity Integrity** → Uses **PRIMARY KEY** to ensure uniqueness.
2. **Referential Integrity** → Uses **FOREIGN KEY** to maintain relationships.
3. **Domain Integrity** → Uses **NOT NULL, CHECK, UNIQUE** to enforce valid values.
4. **User-Defined Integrity** → Enforces business rules via **triggers & procedures**.

****

**Transactions for Consistency:**

* Use **COMMIT** to save changes.
* Use **ROLLBACK** to undo errors.

**Experiment-2**

**Title:-** Experiment based on DDL Commands- CREATE , ALTER , DROP And TRUNCATE.

**Objective:-**

* To understand and implement Data Definition Language (DDL) commands in SQL.
* To learn how to create, modify, delete, and reset database tables using DDL commands

**Theory:-**

DDL is an abbreviation of Data Definition Language.The DDL Commands in Structured Query Language are used to create and modify the schema of the database and its objects. The syntax of DDL commands is predefined for describing the data. The

commands of Data Definition Language deal with how the data should exist in the database. Following are the five DDL commands in SQL:

1. CREATE Command
2. DROP Command
3. ALTER Command
4. TRUNCATE Command
5. RENAME Command

* **CREATE Command:-** CREATE is a DDL command used to create databases, tables, triggers and other database objects.

SYNTAX:-

* 1. For database creation:- CREATE Database Database\_Name;
  2. For table creation:-

CREATE TABLE table\_name (

column\_Name1 data\_type (size of the column ) , column\_Name2 data\_type (size of the column) , column\_Name3 data\_type (size of the column) ,

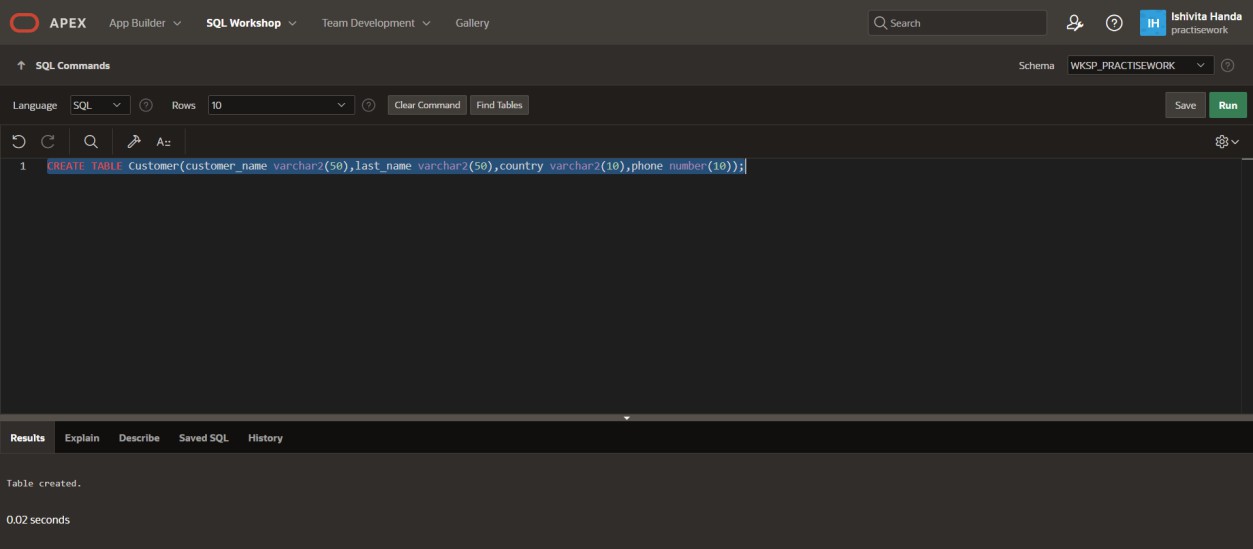
...

column\_NameN data\_type ( size of the column )

) ;

# EXAMPLE:-

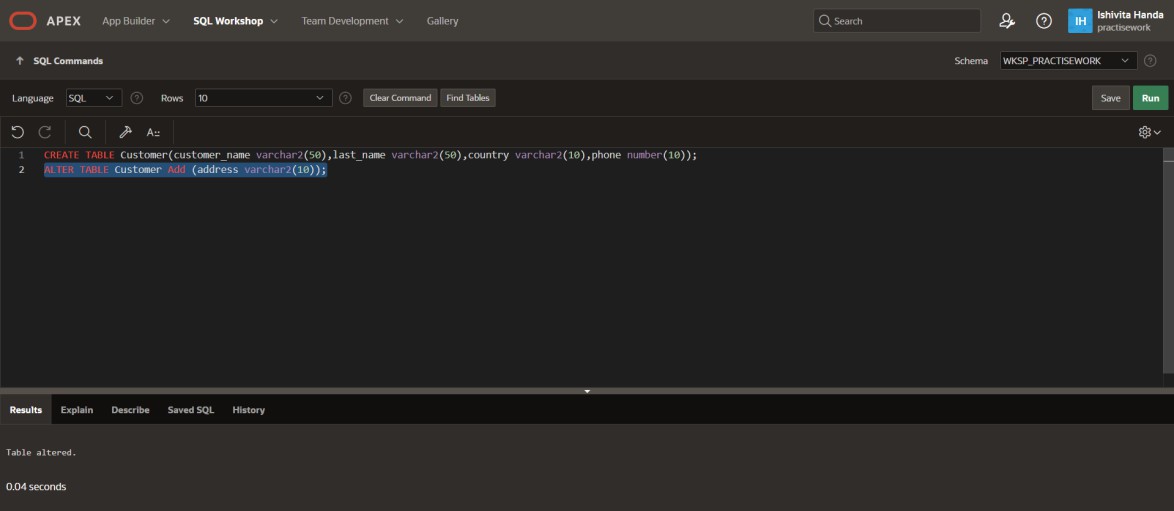
CREATE TABLE Customer(customer\_name varchar2(50),last\_name varchar2(50),country varchar2(10),phone number(10));



* ALTER Command:- ALTER is a DDL command which changes or modifies the existing structure of the database, and it also changes the schema of database objects.
* SYNTAX:-

ALTER TABLE name\_of\_table ADD column\_name column\_definition;

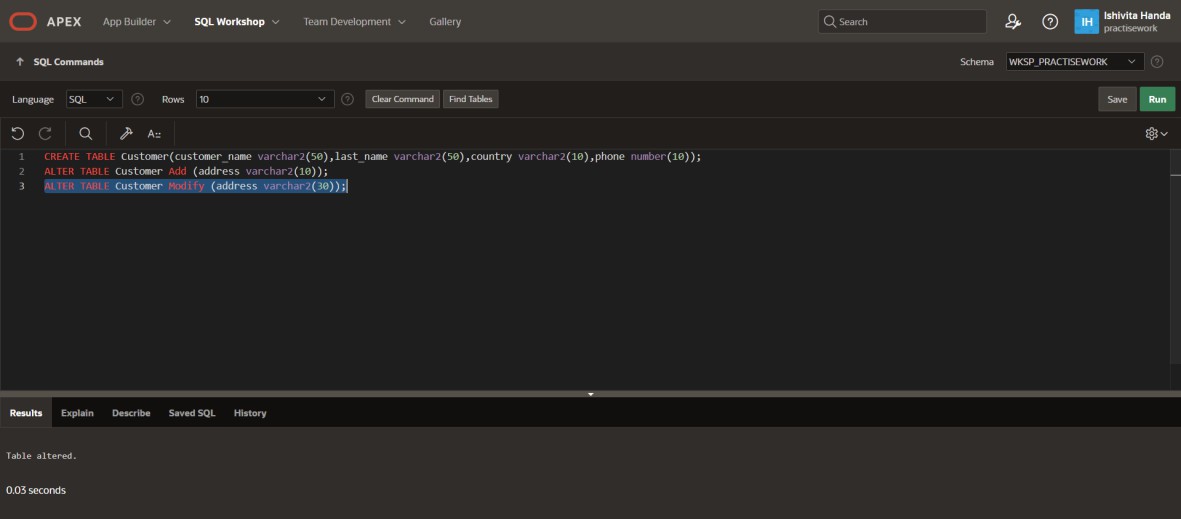
EXAMPLE:- ALTER TABLE Customer Add (address varchar2(10));



# SYNTAX:-

# ALTER TABLE table\_name MODIFY ( column\_name column\_datatype(size));

EXAMPLE:-

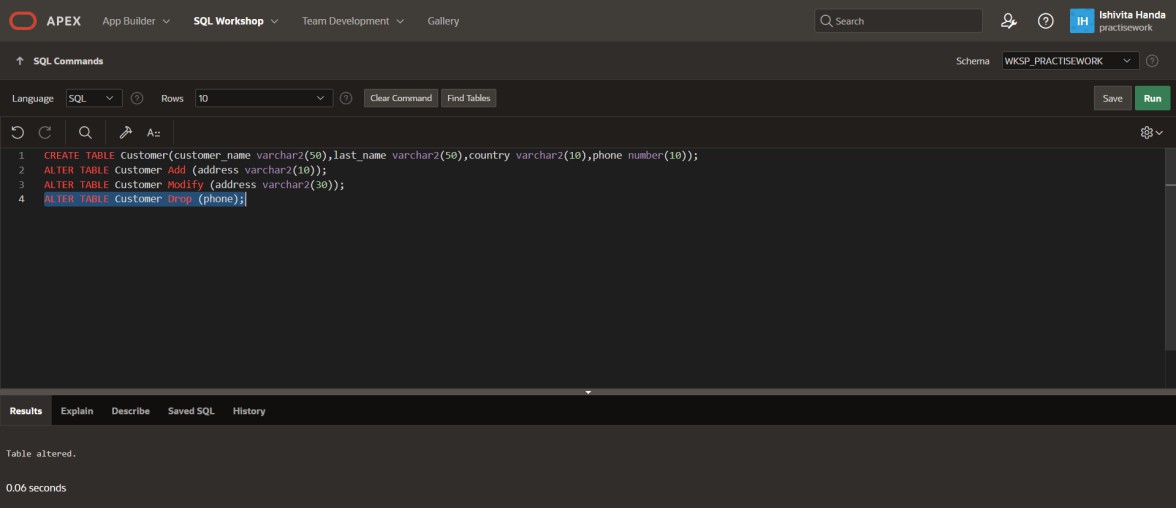
ALTER TABLE Customer Modify (address varchar2(30));

# SYNTAX:-

ALTER TABLE name\_of\_table DROP Column\_Name1 , …..,column\_Name\_N;

EXAMPLE:-

ALTER TABLE Customer Drop (phone);

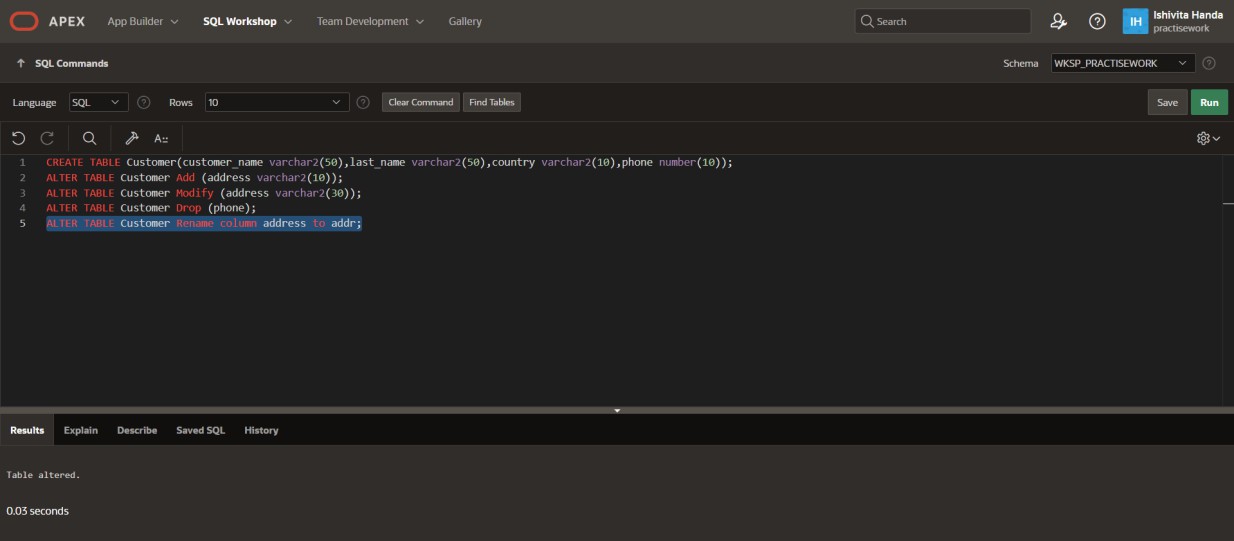


# SYNTAX:-

ALTER TABLE table\_name RENAME column prev\_name to new\_name;

# EXAMPLE:-

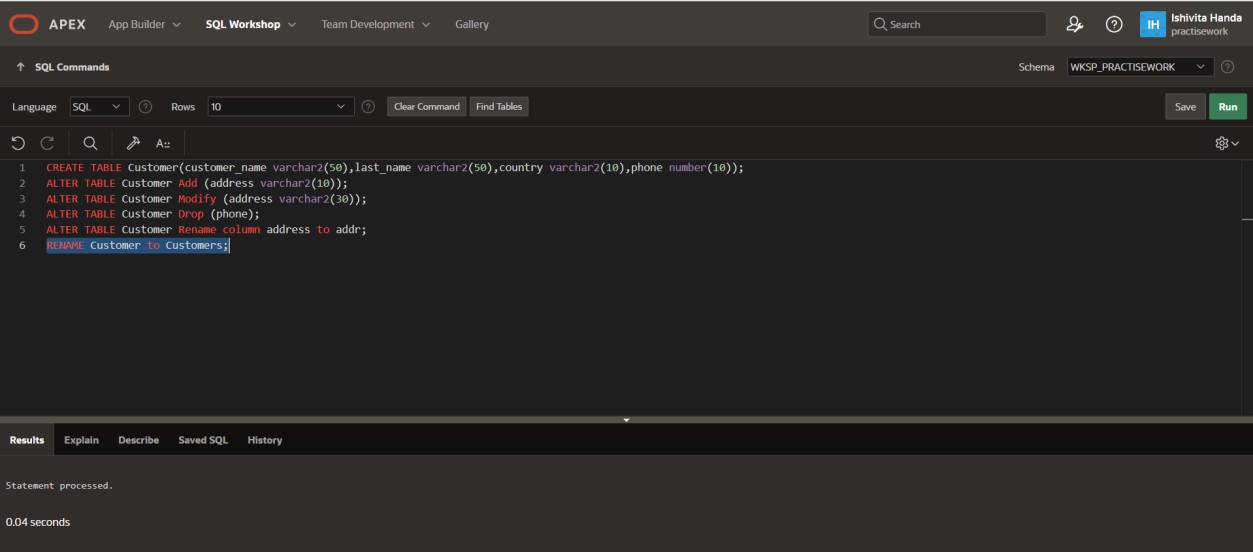
ALTER TABLE Customer Rename column address to addr;



* RENAME Command:- RENAME is a DDL command which is used to change the name of the database table.

SYNTAX:- RENAME TABLE Old\_Table\_Name TO New\_Table\_Name;

EXAMPLE:-

RENAME Customer to Customers;

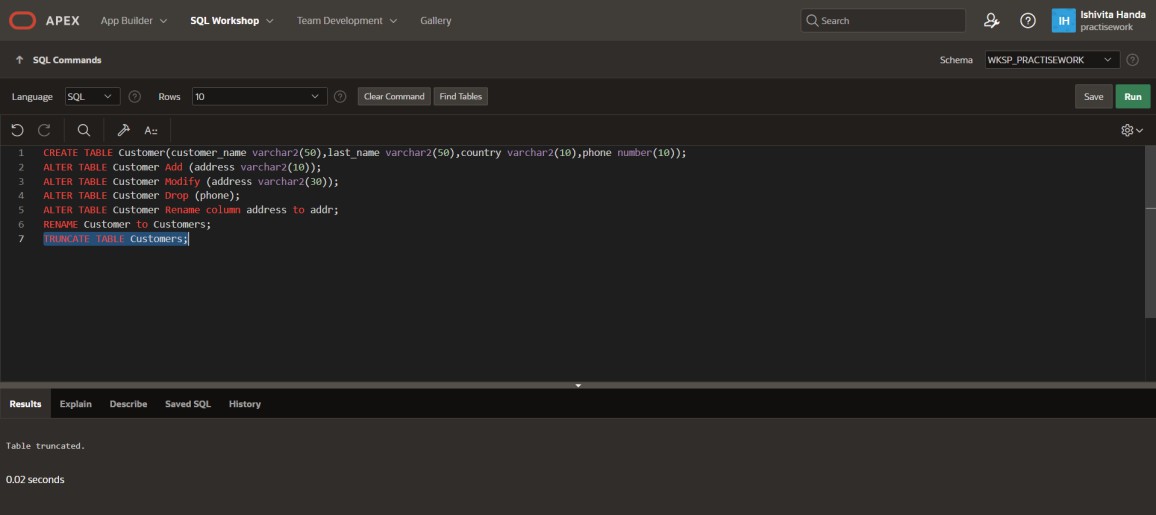
* TRUNCATE Command:- TRUNCATE is another DDL command which deletes or removes all the records from the table.This command also removes the space

allocated for storing the table records.

SYNTAX:- TRUNCATE TABLE Table\_Name;

EXAMPLE:-

TRUNCATE TABLE Customers;



* DROP Command:- DROP is a DDL command used to delete/remove the database objects from the SQL database. We can easily remove the entire table, view, or index from the database using this DDL command.

SYNTAX:-

* 1. DROP DATABASE Database\_Name;
  2. DROP TABLE Table\_Name;

EXAMPLE:-DROP TABLE Customers;



**Experiment-3**

**Title:-** Apply the integrity constraints like Primary Key, Foreign key, Check, NOT NULL, etc. to the tables.

**Objective:-**

* To understand and apply integrity constraints like Primary Key, Foreign Key, Check, NOT NULL, etc., to database tables.
* To ensure data consistency and enforce relationships between tables in a database.

**Theory:-**

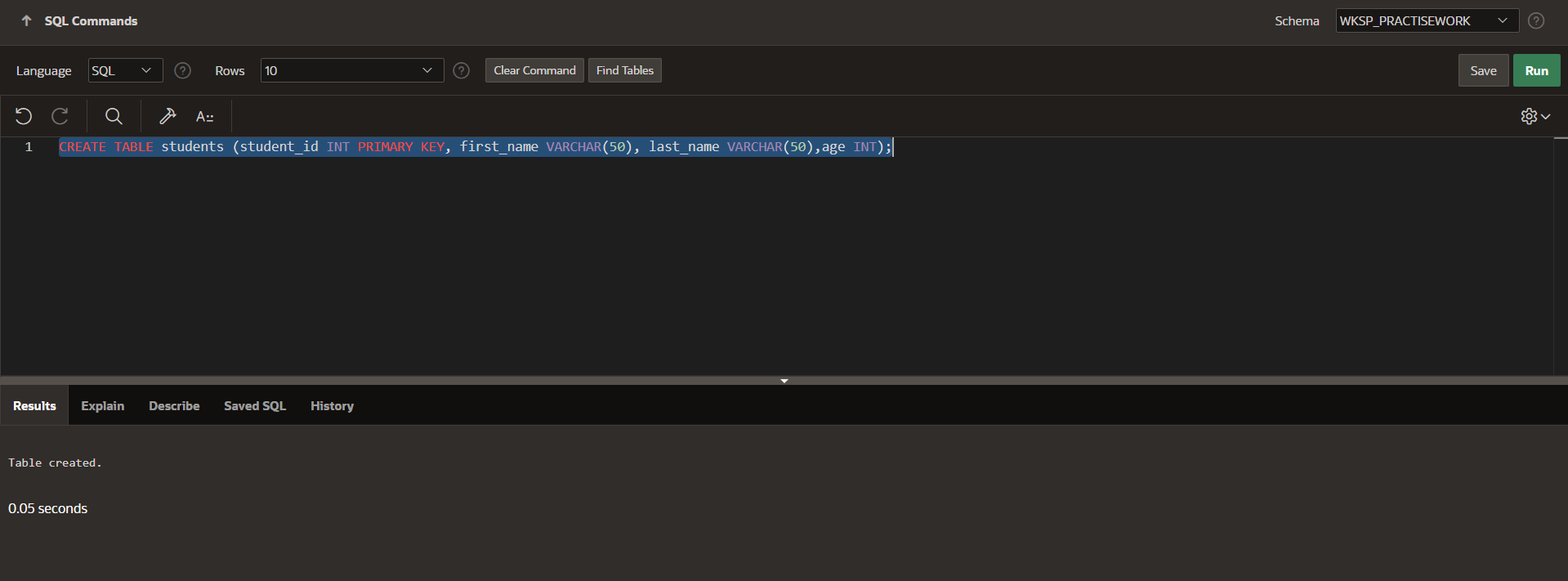
Structured Query Language (SQL) is a fundamental tool in the world of data, used to create, manipulate, and manage data stored in relational databases. One of the pivotal components in SQL are constraints, which are rules applied to columns or tables to prevent unwanted or inaccurate data from being stored. Constraints are essential in maintaining the integrity and reliability of the data in your database. This blog post is designed to provide a comprehensive, yet accessible understanding of SQL constraints, namely PRIMARY KEY, FOREIGN KEY, NOT NULL, UNIQUE, and CHECK. With practical examples and explanations, beginners and professionals alike will find this guide helpful in enhancing their SQL knowledge.

* **PRIMARY KEY Constraint:-**

The PRIMARY KEY constraint uniquely identifies each record in a table. Primary keys must contain unique values and cannot contain null values. A table can have only one primary key, which may consist of single or multiple fields**.**

EXAMPLE:-

CREATE TABLE students (student\_id INT PRIMARY KEY, first\_name VARCHAR(50), last\_name VARCHAR(50),age INT);

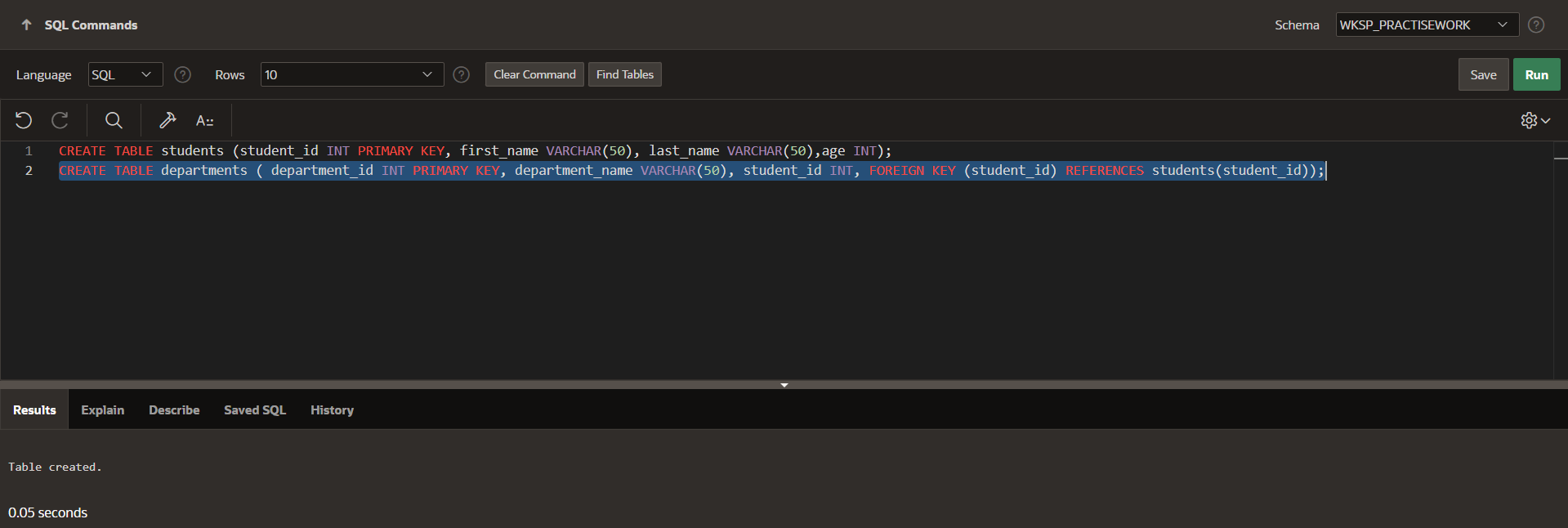


* **FOREIGN KEY Constraint:-**

A FOREIGN KEY is a field (or collection of fields) in a table, that is the primary key of another table. The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables.

EXAMPLE:-

CREATE TABLE departments ( department\_id INT PRIMARY KEY, department\_name VARCHAR(50), student\_id INT, FOREIGN KEY (student\_id) REFERENCES students(student\_id));

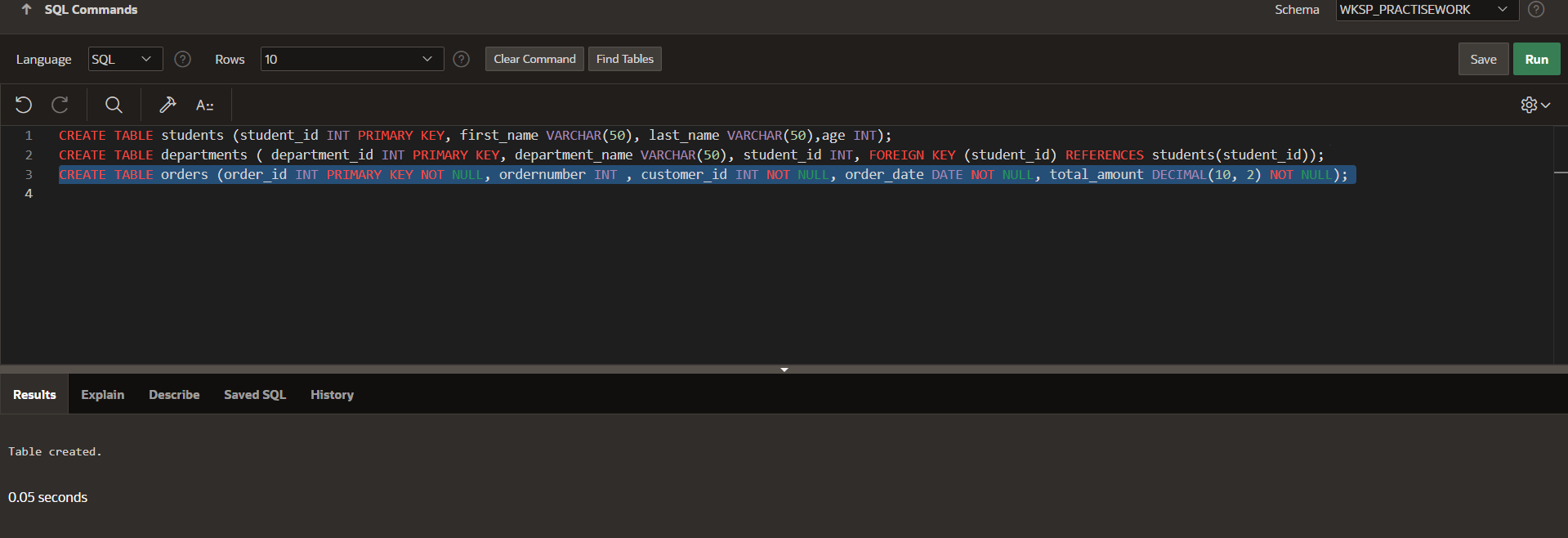


* **NOT NULL Constraint:-**

By default, a column in a SQL table can hold null values. The NOT NULL constraint enforces a column to not accept null values, ensuring that a column always has a value.

EXAMPLE:-

CREATE TABLE orders (order\_id INT PRIMARY KEY NOT NULL, ordernumber INT , customer\_id INT NOT NULL, order\_date DATE NOT NULL, total\_amount DECIMAL (10, 2) NOT NULL);

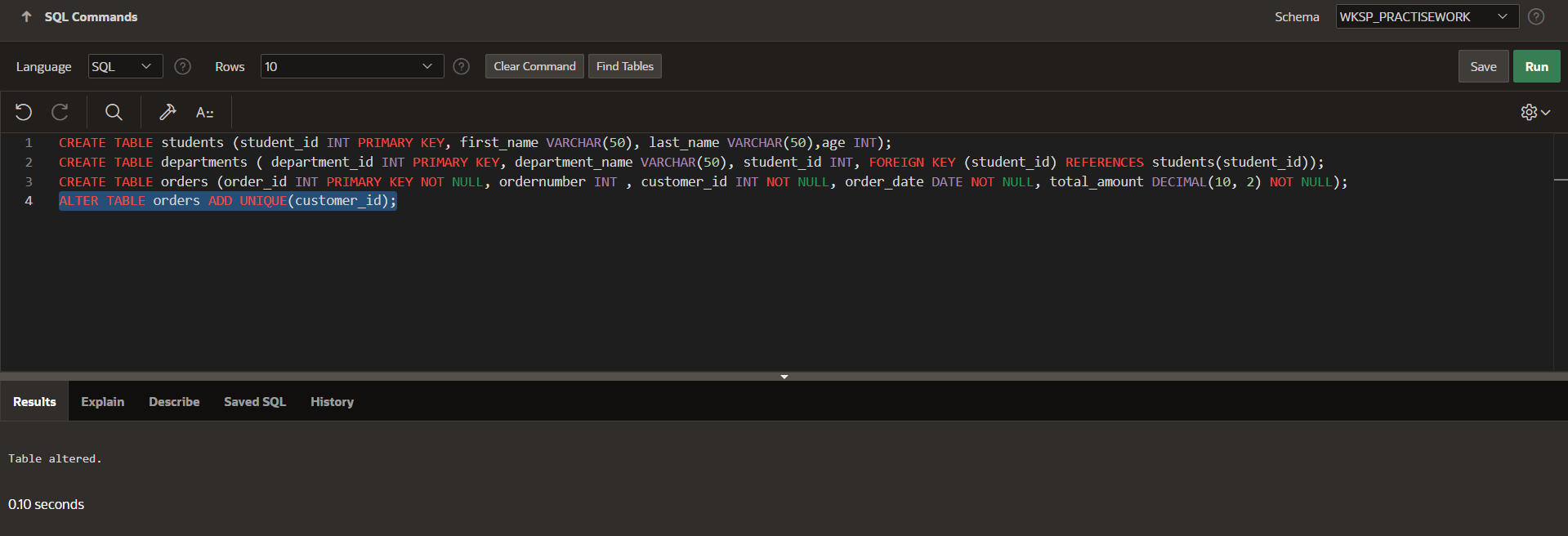


* **UNIQUE Constraint:-**

The UNIQUE constraint ensures that all values in a column are different. Unlike the PRIMARY KEY constraint, you can have multiple UNIQUE constraints in a table, with the ability to accept null values.

EXAMPLE:-

ALTER TABLE orders ADD UNIQUE(customer\_id);

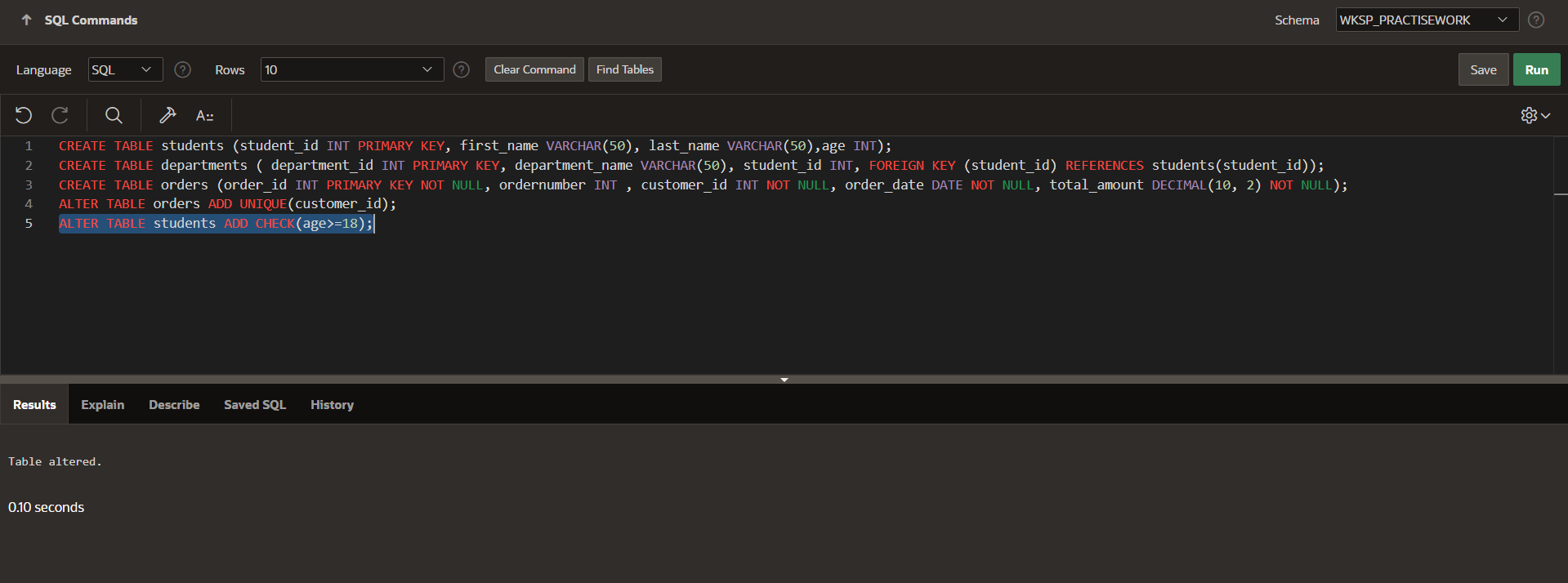


* **CHECK Constraint:-**

The CHECK constraint enforces that all values in a field satisfy a certain condition. This is used to ensure the validity of the data in a column.

EXAMPLE:-

ALTER TABLE students ADD CHECK(age>=18);



**Experiment-4**

**Title:-** Experiments based on basic DML commands – SELECT, INSERT, UPDATE and DELETE.

**Objective:-**

* To understand and implement the basic Data Manipulation Language (DML) commands in SQL.
* To learn how to retrieve, insert, modify, and delete data in tables using DML commands.

**Theory:-**

The SQL commands that deal with the manipulation of data present in the database belong to DMLor Data Manipulation Language and this includes most of the SQL statements. It is the component of the SQL statement that controls access to data and to the database.

The examples of DML in the Database Management System (DBMS) are as follows −

* SELECT − Retrieve data from the database.
* INSERT − Insert data into a table.
* UPDATE − Update existing data within a table.
* DELETE − Delete records from a database table.

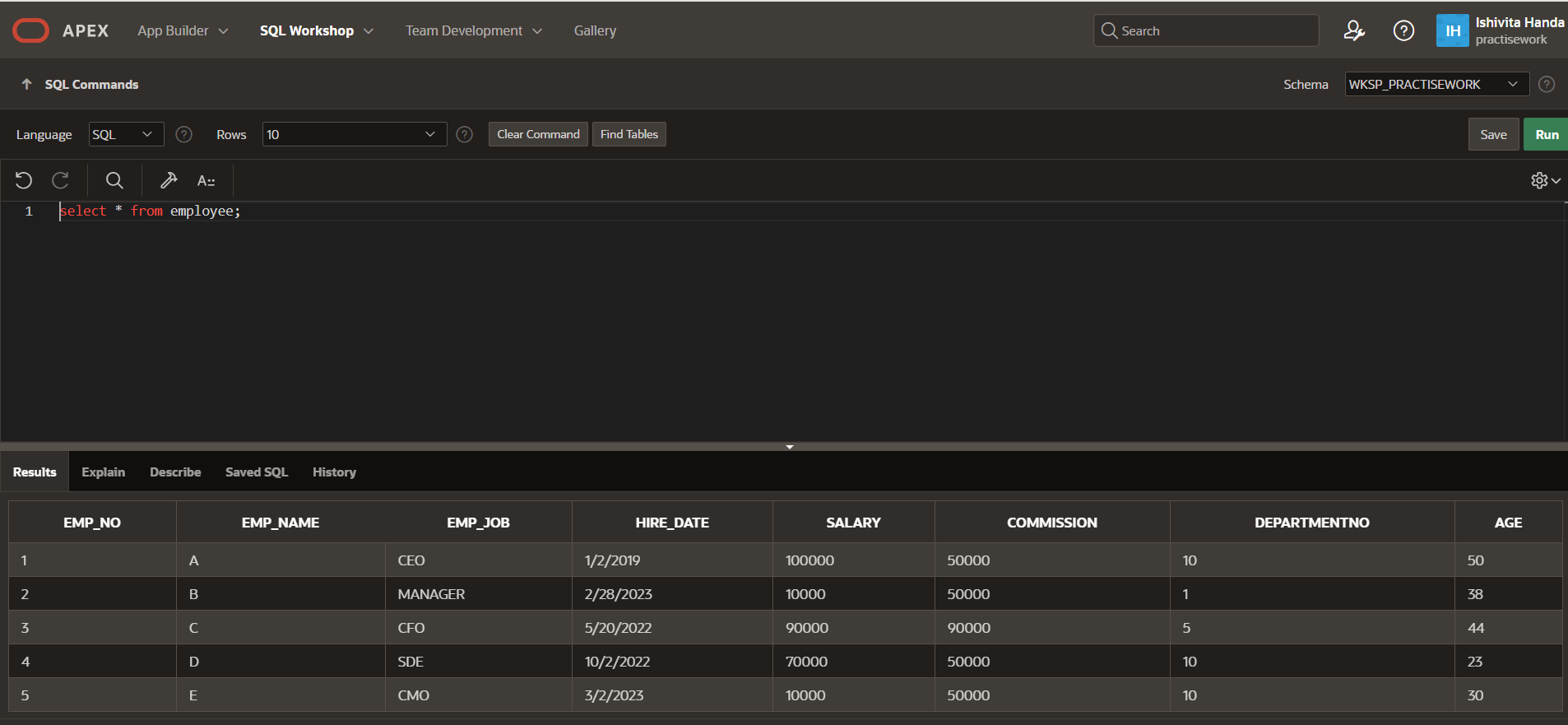
1. **SELECT:-** Select command is used to retrieve data from the database. In the SELECT statement, the table name and column names are specified. It returns data in the form of a result table. These result tables are called “result sets”.

SYNTAX:-

SELECT \* from <table\_name>;

EXAMPLE:-

SELECT \* from Employee;



1. **INSERT:-** It is a command to insert data into the table’s columns. The user must check the table to see whether it has any integrity constraints, such as PRIMARY KEY, UNIQUE, NOT NULL, etc., and enter records accordingly.

SYNTAX:-

Insert into <table\_name> (column list) values (column values);

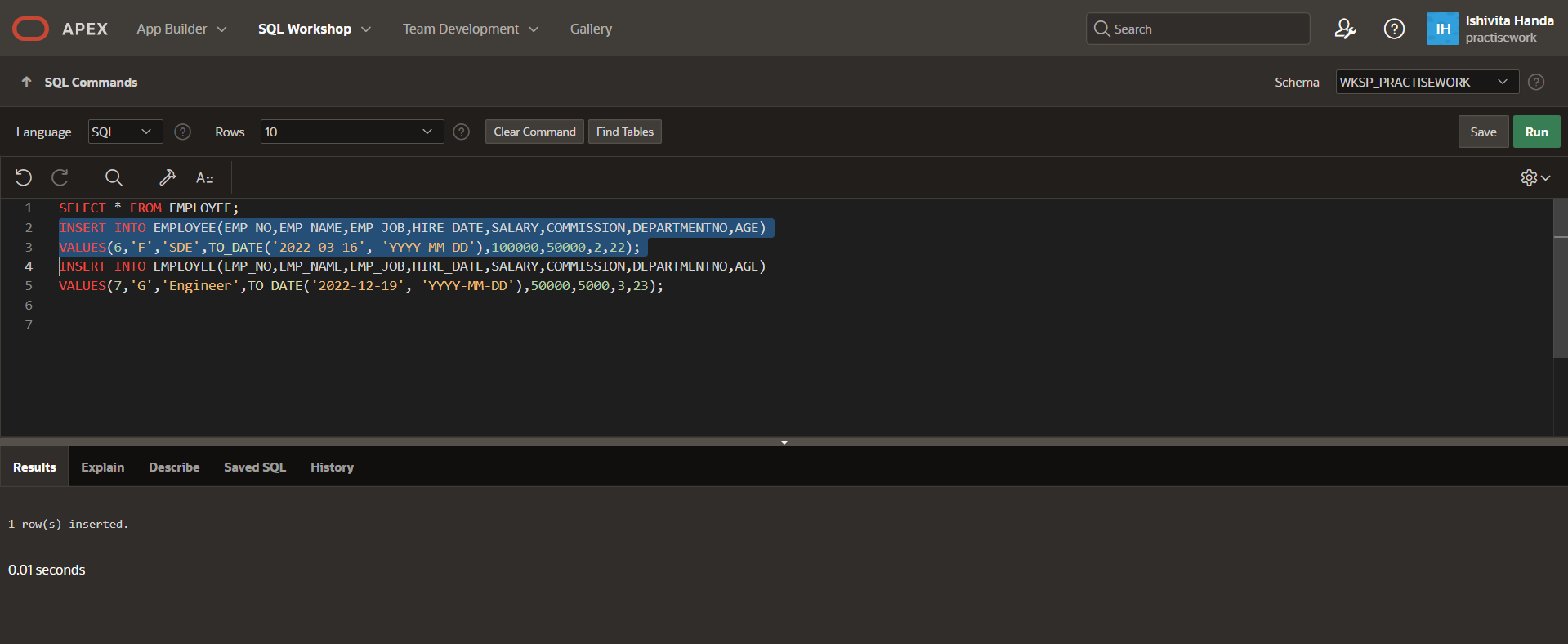
EXAMPLE:-

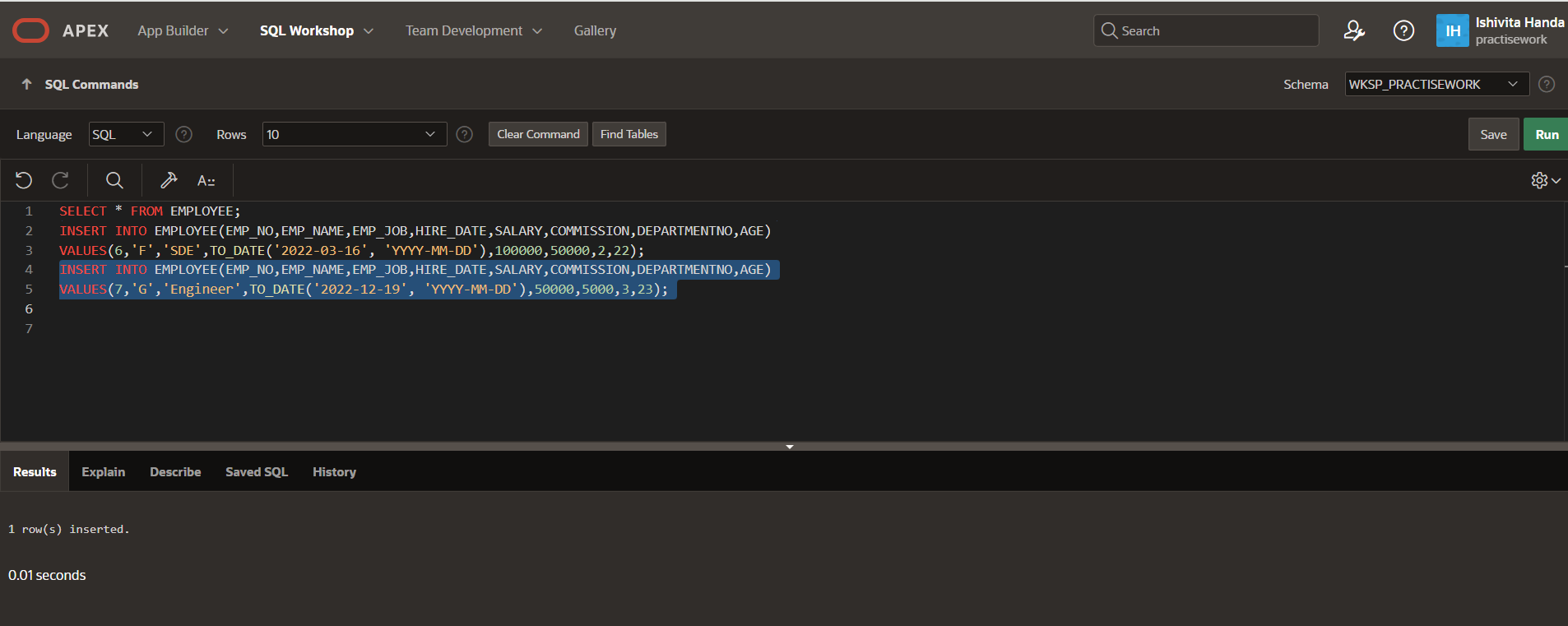
INSERT INTO EMPLOYEE (EMP\_NO , EMP\_NAME , EMP\_JOB , HIRE\_DATE , SALARY , COMMISSION , DEPARTMENTNO , AGE)

VALUES(6,'F','SDE',TO\_DATE('2022-03-16', 'YYYY-MM-DD'),100000,50000,2,22);

INSERT INTO EMPLOYEE (EMP\_NO , EMP\_NAME , EMP\_JOB , HIRE\_DATE , SALARY , COMMISSION , DEPARTMENTNO , AGE)

VALUES(7,'G','Engineer',TO\_DATE('2022-12-19', 'YYYY-MM-DD'),50000,5000,3,23);



****

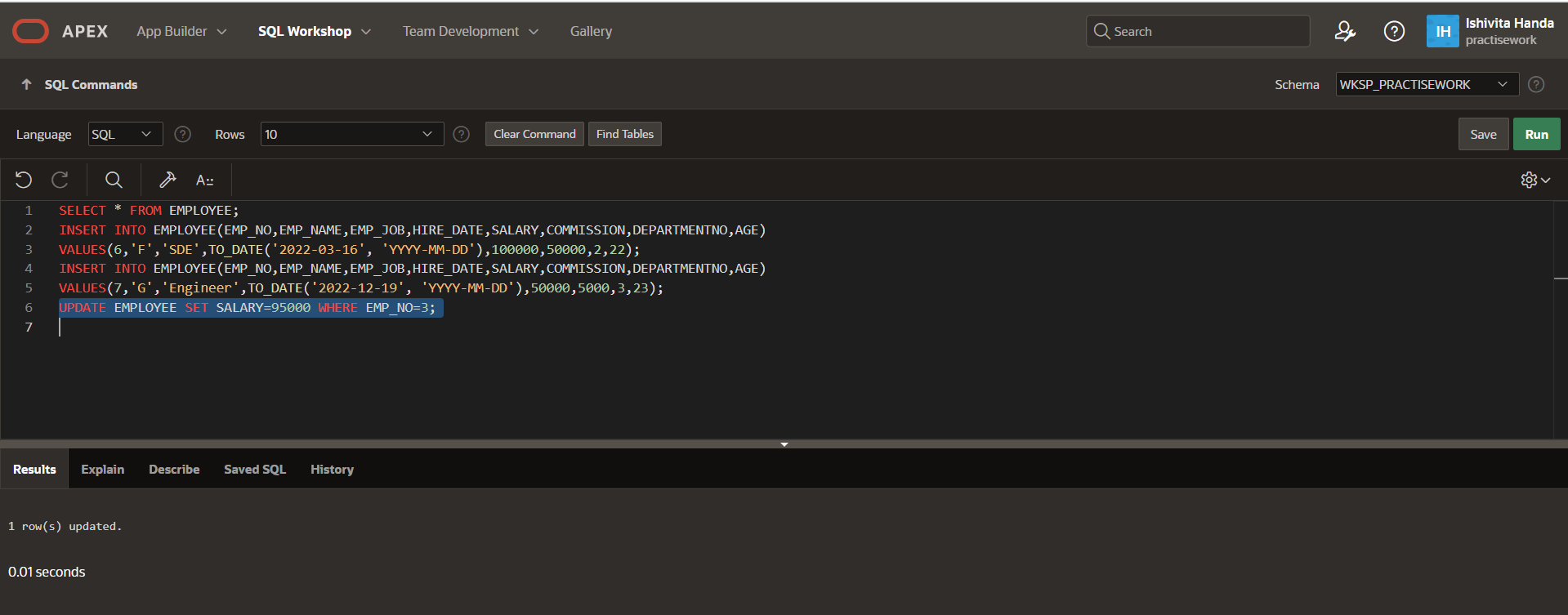
1. **UPDATE:-** This DML command in SQL modifies the record present in the existing table. It updates the records by using a WHERE clause that specifies the condition with an UPDATE statement. You need to mention the condition, otherwise, all the rows will be affected.

SYNTAX:-

UPDATE <table\_name> SET column\_number =value\_number WHERE condition;

EXAMPLE:-

UPDATE EMPLOYEE SET SALARY=95000 WHERE EMP\_NO=3;



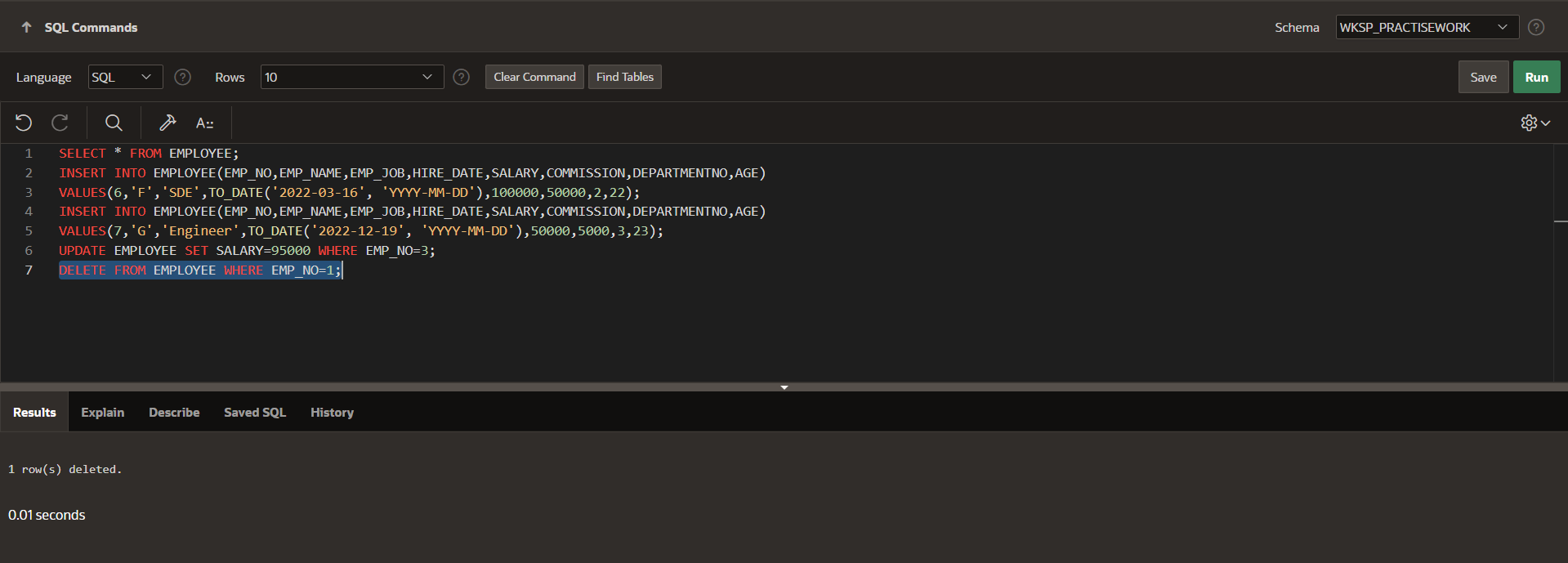
1. **DELETE:-** It is used to delete some or all of the records from the existing table. It should contain the WHERE clause to give the condition. It is necessary to mention the WHERE condition to delete the selected rows; otherwise, it will delete all the data.

SYNTAX:-

Delete from <table\_name>WHERE condition;

EXAMPLE:-

DELETE FROM EMPLOYEE WHERE EMP\_NO=1;



**Experiment-5**

**Title:-** Write the queries for implementing Built-in functions, GROUP BY, HAVING and ORDER BY.

**Objective:-**

* To understand and apply SQL built-in functions.
* To learn how to use GROUP BY, HAVING, and ORDER BY clauses for data aggregation and sorting in SQL queries.

**Theory:-**

* In SQL, built-in data functions are functions that operate on data values to perform calculations and return results. They are categorized into:
  1. Aggregate Functions (Operate on multiple rows and return a single value).
  2. String Functions (Operate on text values).
  3. Date & Time Functions (Work with date/time values).
  4. Mathematical functions (perform numeric operations).
  5. Conversion functions (Convert data types).

Examples of built-in functions:

COUNT()

SUM()

AVG()

MAX() and MIN()

UPPER() and LOWER()



* **GROUP BY:-**

Used to group rows based on one or more columns.

Often used with aggregate functions like SUM(), COUNT(), etc.

SYNTAX:-

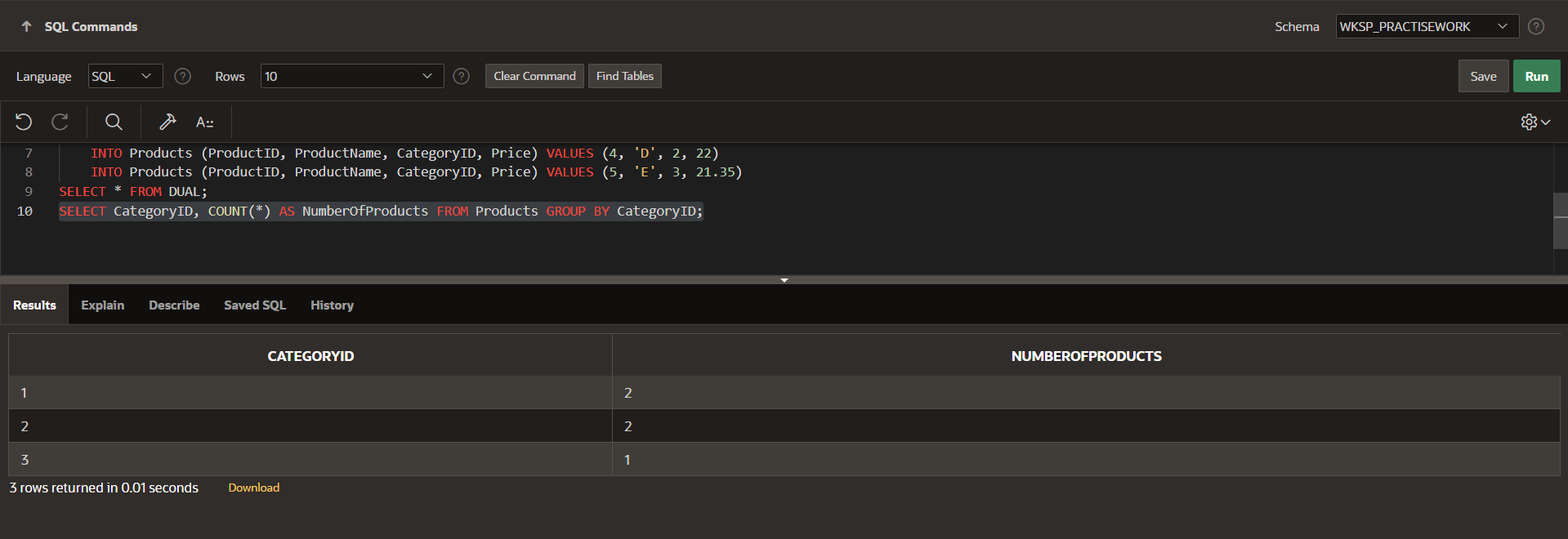
SELECT column\_name, aggregate\_function(column\_name) FROM table\_name

GROUP BY column\_name;

EXAMPLE:-

SELECT CategoryID, COUNT(\*) AS NumberOfProducts FROM Products GROUP

BY CategoryID;



* **HAVING:-**

Filters grouped records (works like WHERE but for aggregated results).

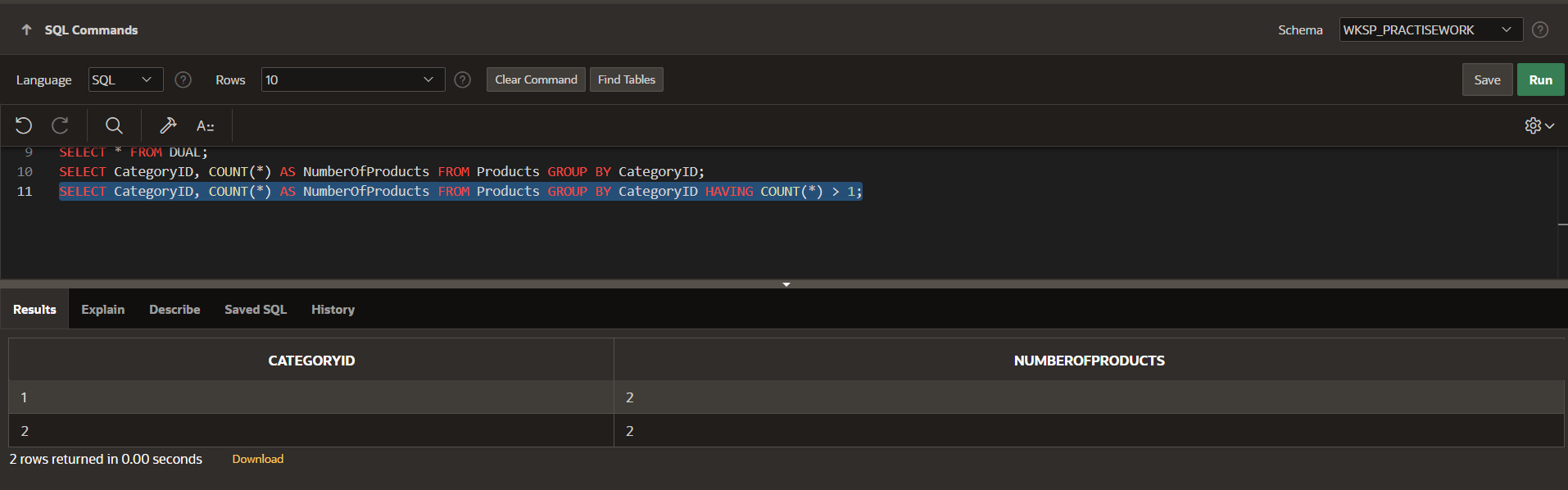
SYNTAX:-

SELECT column\_name, aggregate\_function(column\_name) FROM table\_name

GROUP BY column\_name HAVING condition;

EXAMPLE:-

SELECT CategoryID, COUNT(\*) AS NumberOfProducts FROM Products GROUP BY CategoryID HAVING COUNT(\*) > 1;



* **ORDER BY:-**

Sorts results in ascending (ASC, default) or descending (DESC) order.

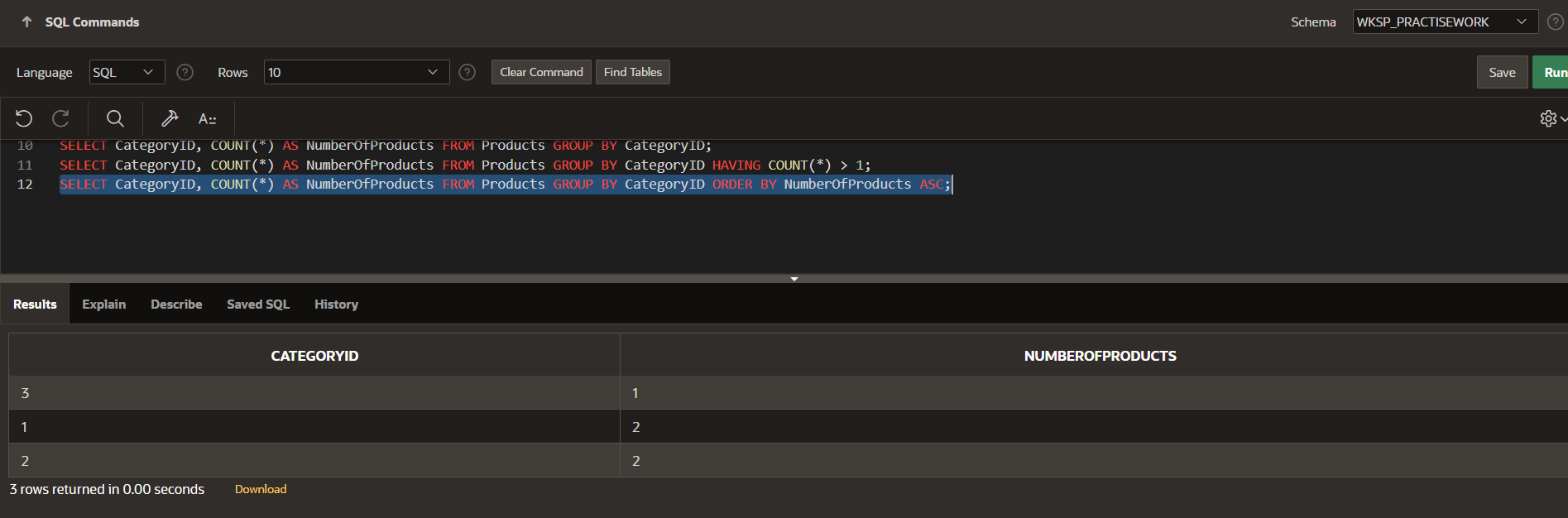
SYNTAX:-

SELECT column\_name, aggregate\_function(column\_name) FROM table\_name

GROUP BY column\_name ORDER BY column\_name [ASC|DESC];

EXAMPLE:-

SELECT CategoryID, COUNT(\*) AS NumberOfProducts FROM Products GROUP BY CategoryID ORDER BY NumberOfProducts ASC;



**Experiment-6**

**Title:-** Write the queries to implement the joins and subqueries.

**Objective:-**

* To understand and implement different types of joins and subqueries in SQL.
* To learn how to retrieve data from multiple tables using joins and subqueries.

**Theory:-**

SQL JOIN clause is used to query and access data from multiple tables by establishing logical relationships between them. It can access data from multiple tables simultaneously using common key values shared across different tables. We can use SQL JOIN with multiple tables. It can also be paired with other clauses, the most popular use will be using JOIN with WHERE clause to filter data retrieval.There are many types of Joins. Depending on the use case, we can use different type of SQL JOIN clause. Below, we explain the most commonly used join types with syntax and examples:

* INNER JOIN
* LEFT JOIN
* RIGHT JOIN
* FULL JOIN
* NATURAL JOIN

1. **INNER JOIN:-** The INNER JOIN keyword selects all rows from both the tables as long as the condition is satisfied. This keyword will create the result-set by combining all rows from both the tables where the condition satisfies i.e value of the common field will be the same.

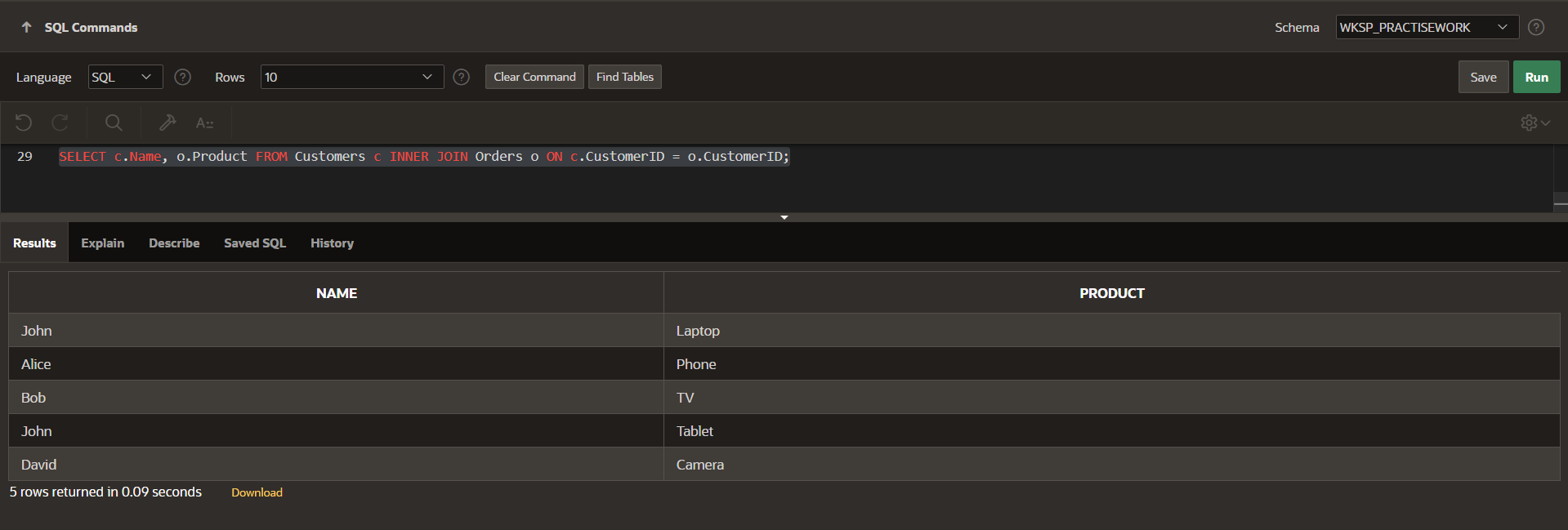
SYNTAX:-

SELECT table1.column1,table1.column2,table2.column1,.... FROM table1   
INNER JOIN table2 ON table1.matching\_column = table2.matching\_column;

EXAMPLE:-

SELECT c.Name, o.Product FROM Customers c INNER JOIN Orders o ON

c.CustomerID = o.CustomerID;



1. **LEFT JOIN:-**

LEFT JOIN returns all the rows of the table on the left side of the join and matches rows for the table on the right side of the join.LEFT JOIN is also known as LEFT OUTER JOIN.

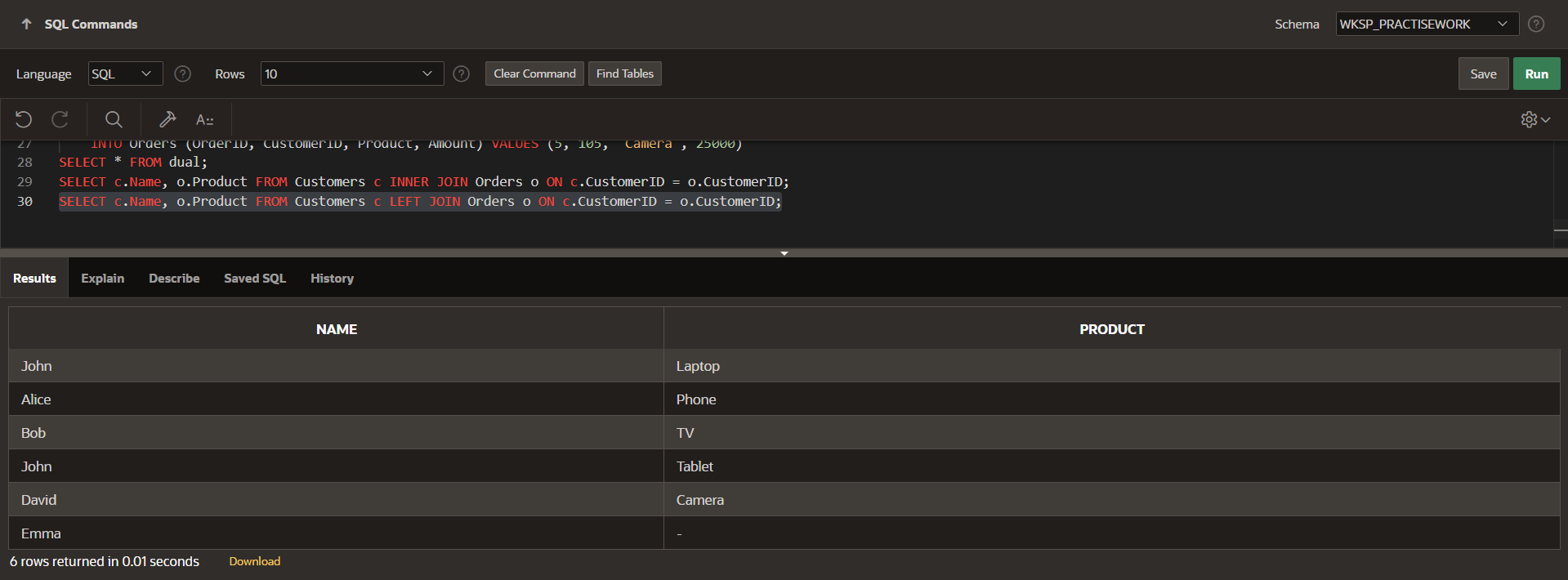
SYNTAX:-

SELECT table1.column1,table1.column2,table2.column1,.... FROM table1   
LEFT JOIN table2 ON table1.matching\_column = table2.matching\_column;

EXAMPLE:-

SELECT c.Name, o.Product FROM Customers c LEFT JOIN Orders o ON

c.CustomerID = o.CustomerID;



1. **RIGHT JOIN:-**

RIGHT JOIN returns all the rows of the table on the right side of the join and matching rows for the table on the left side of the join.RIGHT JOIN is also known as RIGHT OUTER JOIN.

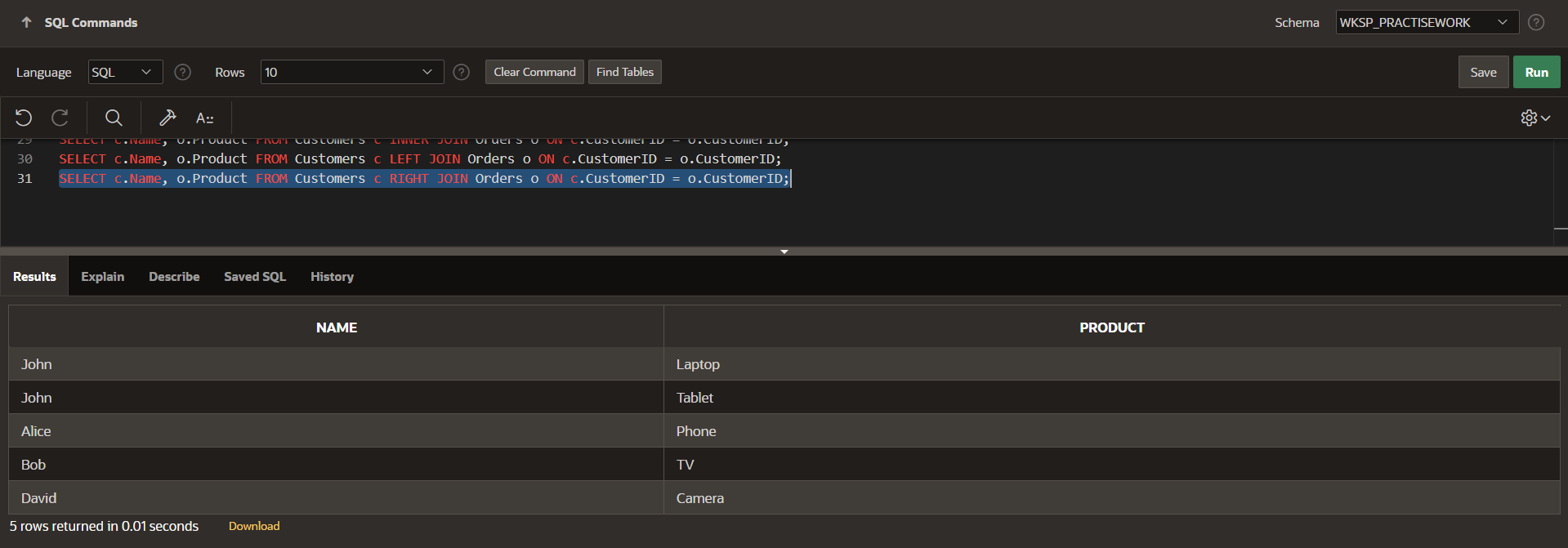
SYNTAX:-

SELECT table1.column1,table1.column2,table2.column1,.... FROM table1   
RIGHT JOIN table2 ON table1.matching\_column = table2.matching\_column;

EXAMPLE:-

SELECT c.Name, o.Product FROM Customers c RIGHT JOIN Orders o ON

c.CustomerID = o.CustomerID;



1. **FULL JOIN:-**

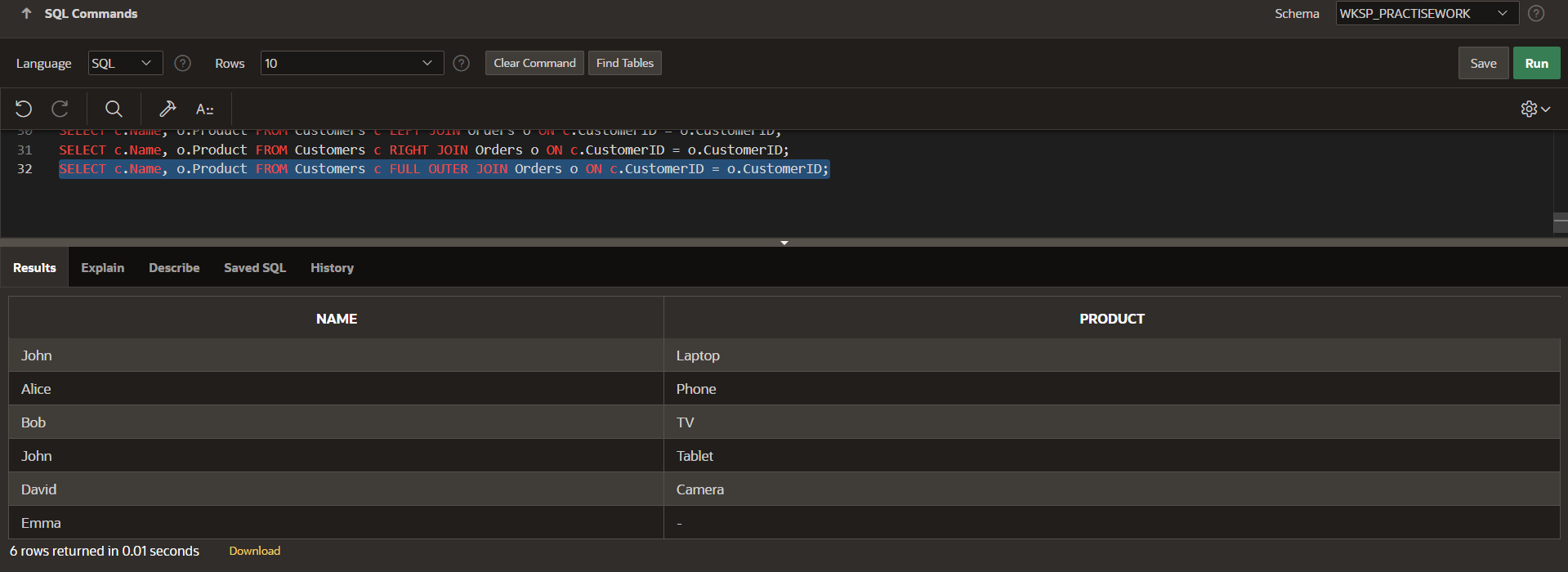
FULL JOIN creates the result-set by combining results of both LEFT JOIN and RIGHT JOIN. The result-set will contain all the rows from both tables. For the rows for which there is no matching, the result-set will contain NULL values.

SYNTAX:-

SELECT table1.column1,table1.column2,table2.column1,.... FROM table1   
FULL JOIN table2 ON table1.matching\_column = table2.matching\_column;

EXAMPLE:-

SELECT c.Name, o.Product FROM Customers c FULL OUTER JOIN Orders o ON c.CustomerID = o.CustomerID;



* **SUBQUERIES IN SQL:-**

In SQL, a subquery can be defined as a query embedded within another query. It is often used in the WHERE, HAVING, or FROM clauses of a statement. Subqueries are commonly used with SELECT, UPDATE, INSERT, and DELETE statements to achieve complex filtering and data manipulation.

SUBQUERY can be located in :

* + - A SELECT clause
  + - A FROM clause
  + - A WHERE clause
  + - A HAVING clause

1. **WHERE CLAUSE:-** Subqueries in the WHERE clause help filter data based on the results of another query.

SYNTAX:-

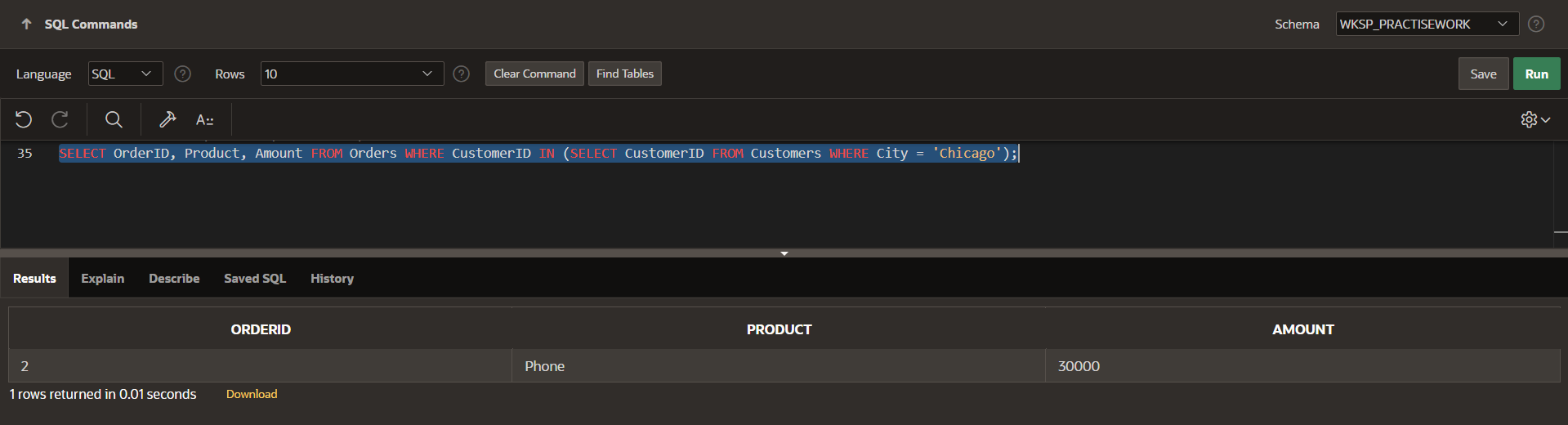
SELECT column\_name FROM table\_name WHERE column\_name

expression operator (SELECT column\_name FROM table\_name WHERE ...);

EXAMPLE:-

SELECT OrderID, Product, Amount FROM Orders WHERE CustomerID IN

(SELECT CustomerID FROM Customers WHERE City = 'Chicago');

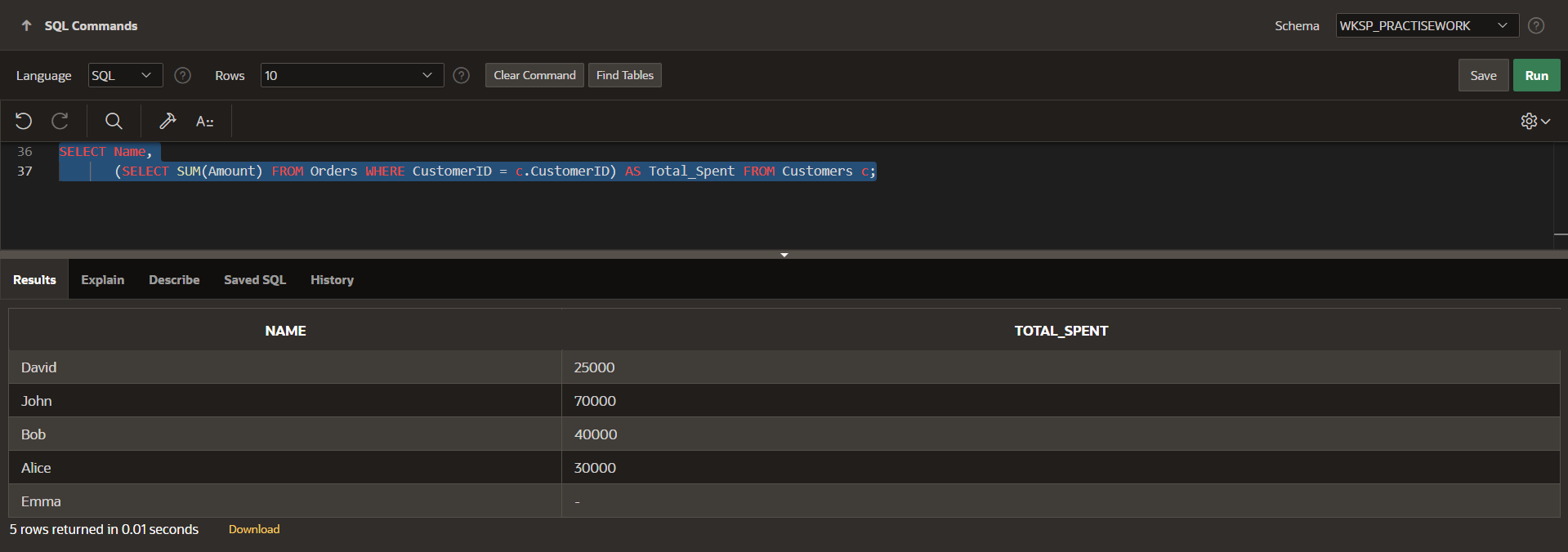
****

1. **SELECT CLAUSE:-**

EXAMPLE:- SELECT Name,

(SELECT SUM(Amount) FROM Orders WHERE CustomerID = c.CustomerID)

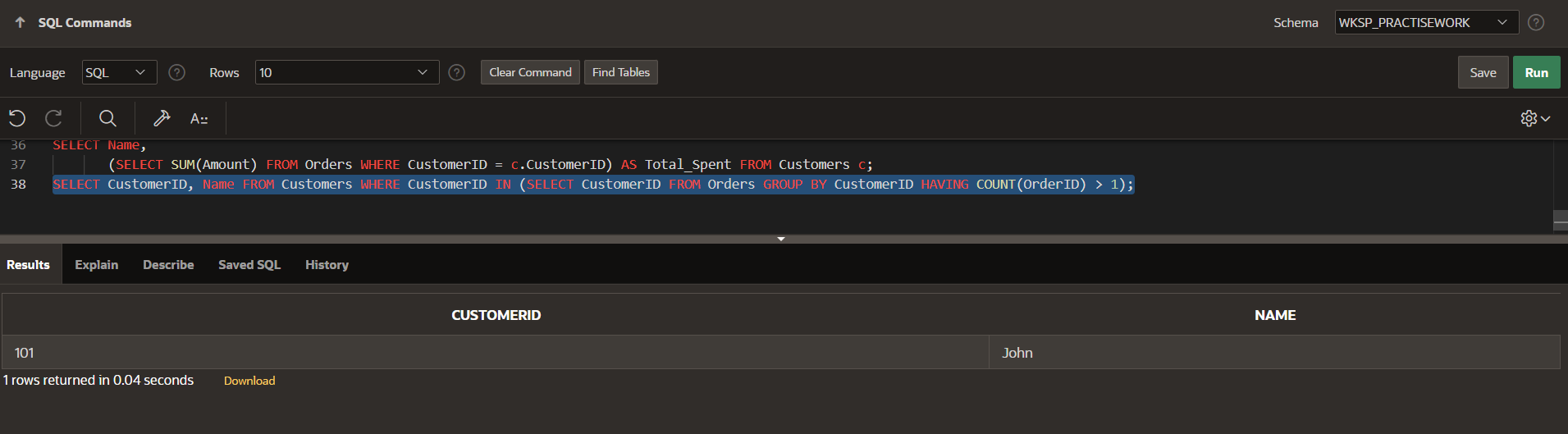
AS Total\_Spent FROM Customers c;

****

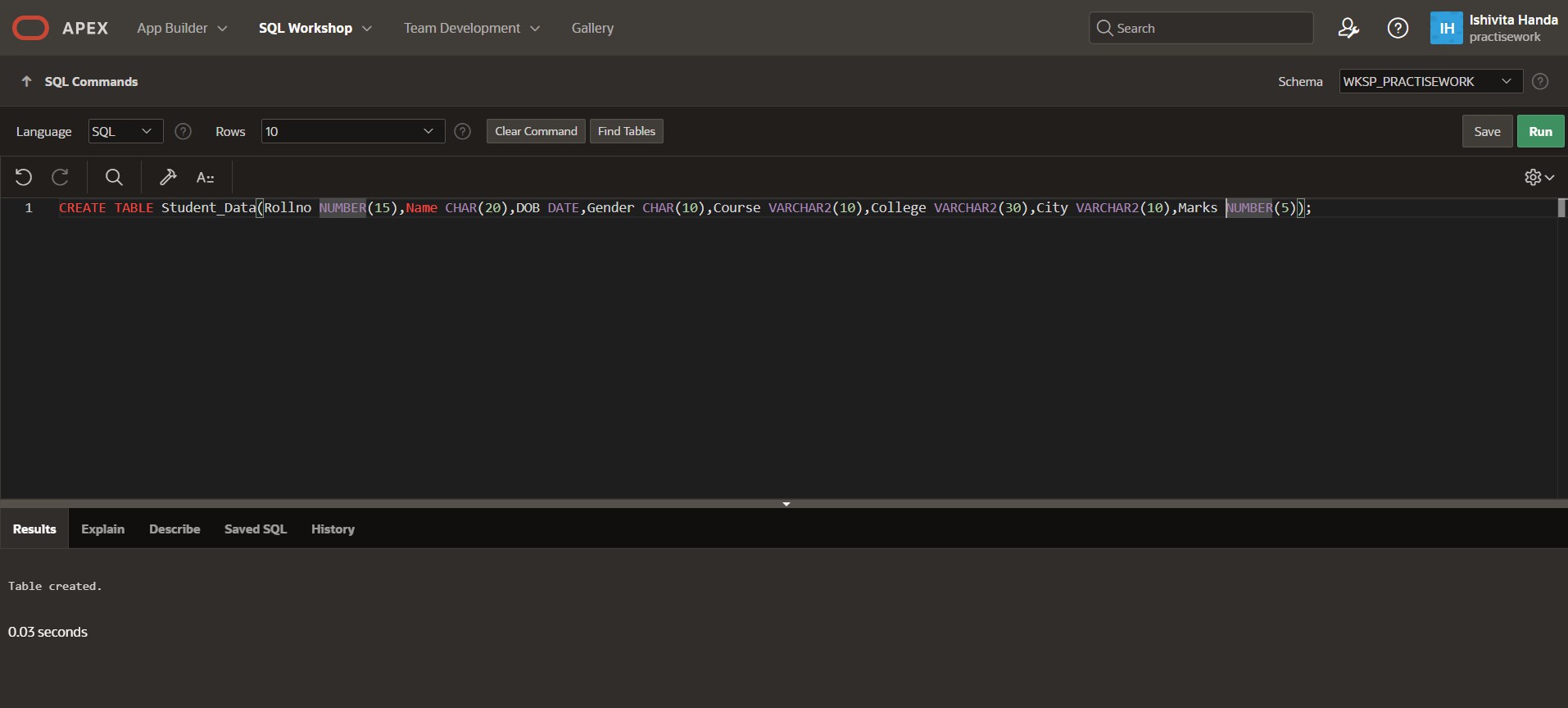
1. **HAVING CLAUSE:-**

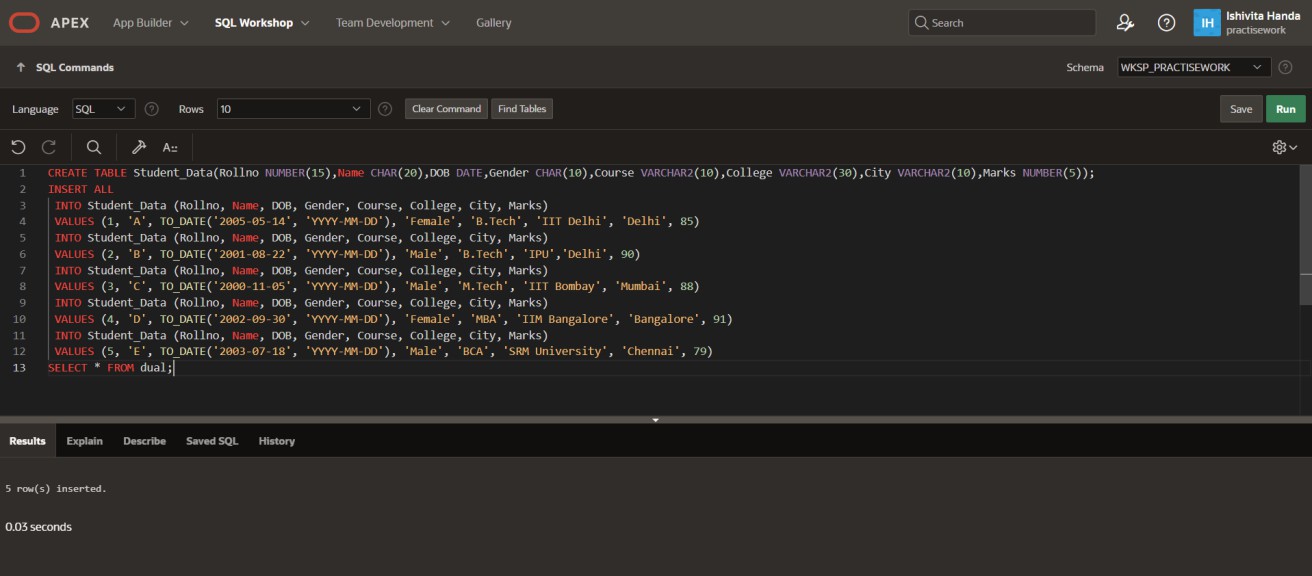
EXAMPLE:-

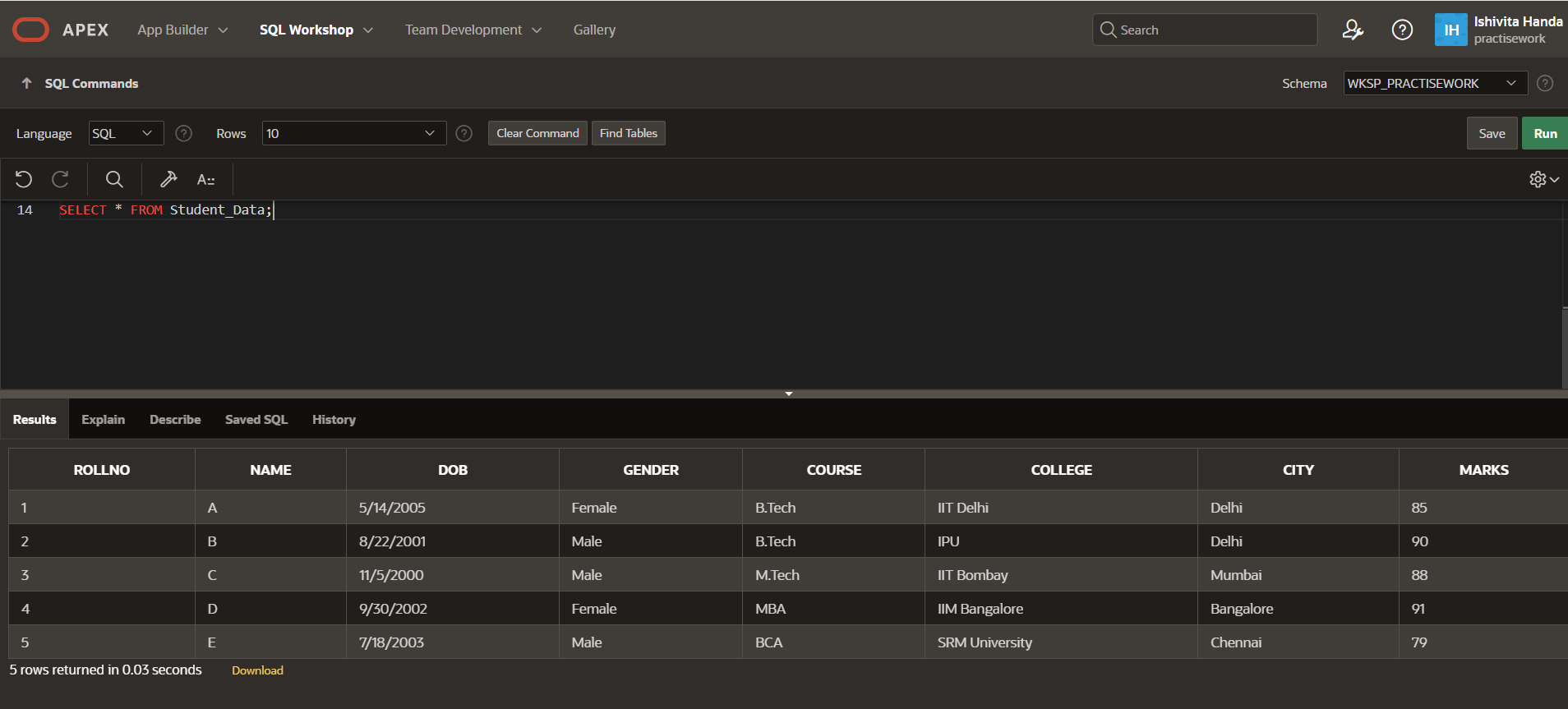
SELECT CustomerID, Name FROM Customers WHERE CustomerID IN (SELECT CustomerID FROM Orders GROUP BY CustomerID HAVING COUNT(OrderID) > 1);



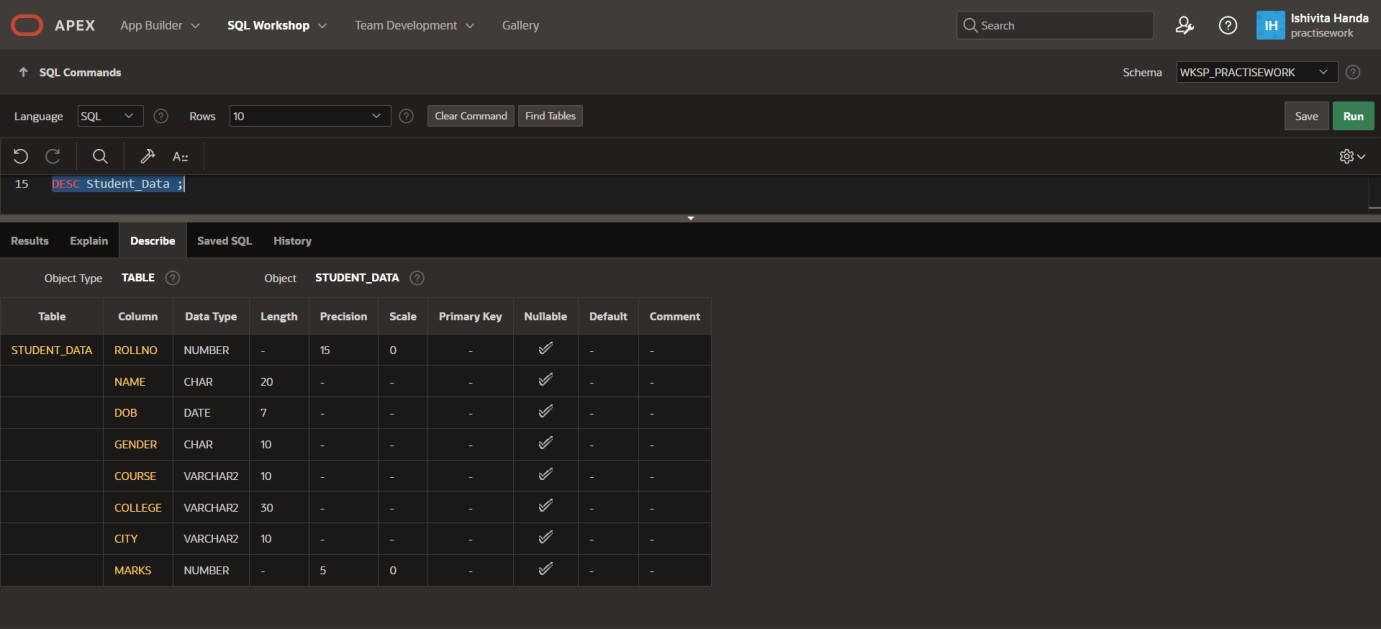
**Lab Assignment 1**

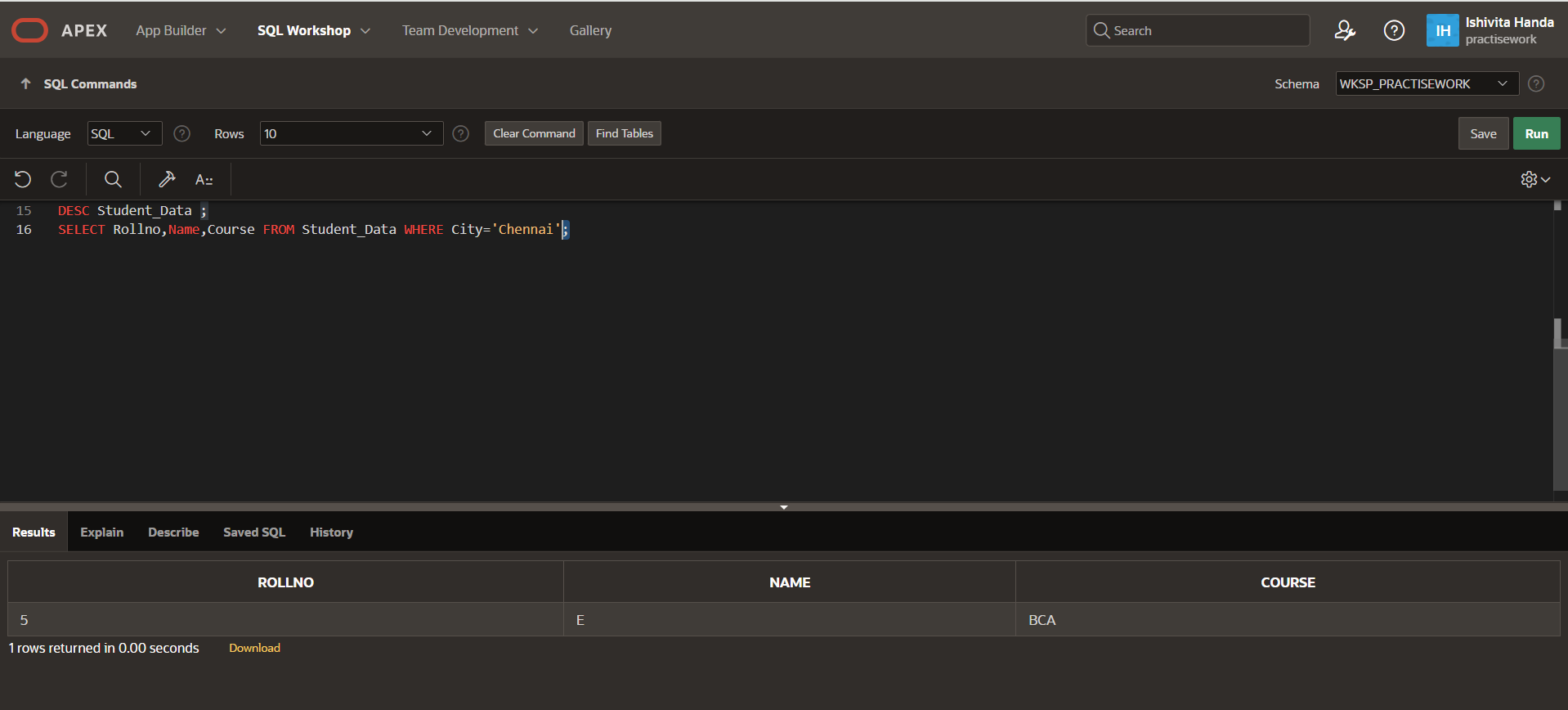
Q.1 Create a table student which contain attributes like roll no.,dob,gender,class,college,city and marks.

  
Q.2 Insert any 5 records

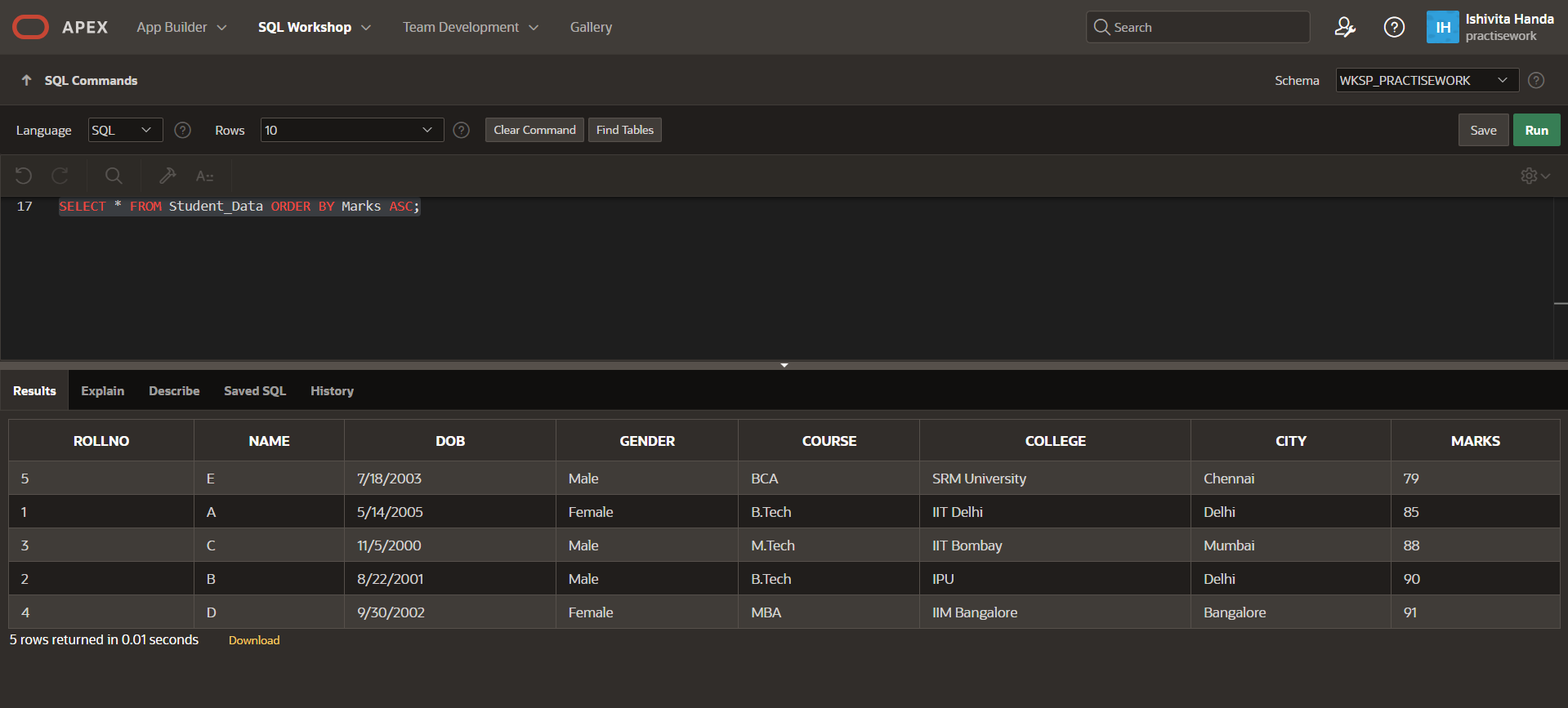
  
Q.3 Display the information of all the students

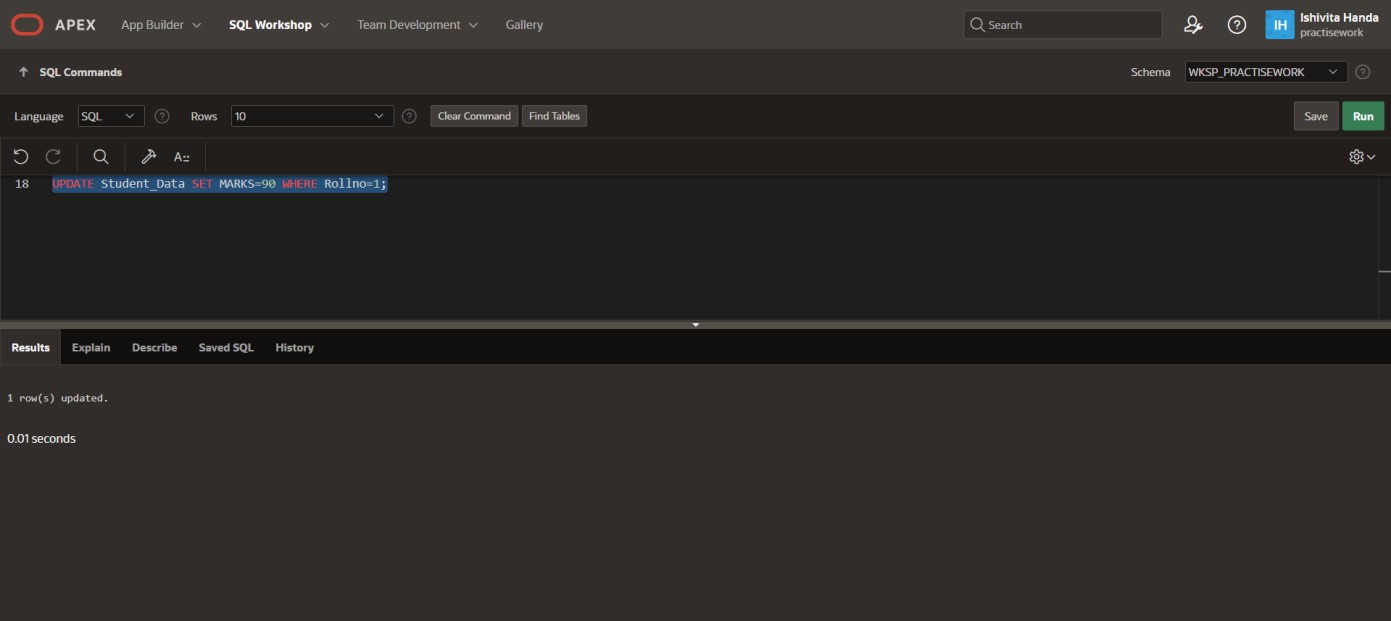
Q.4 Display detailed structure of student table

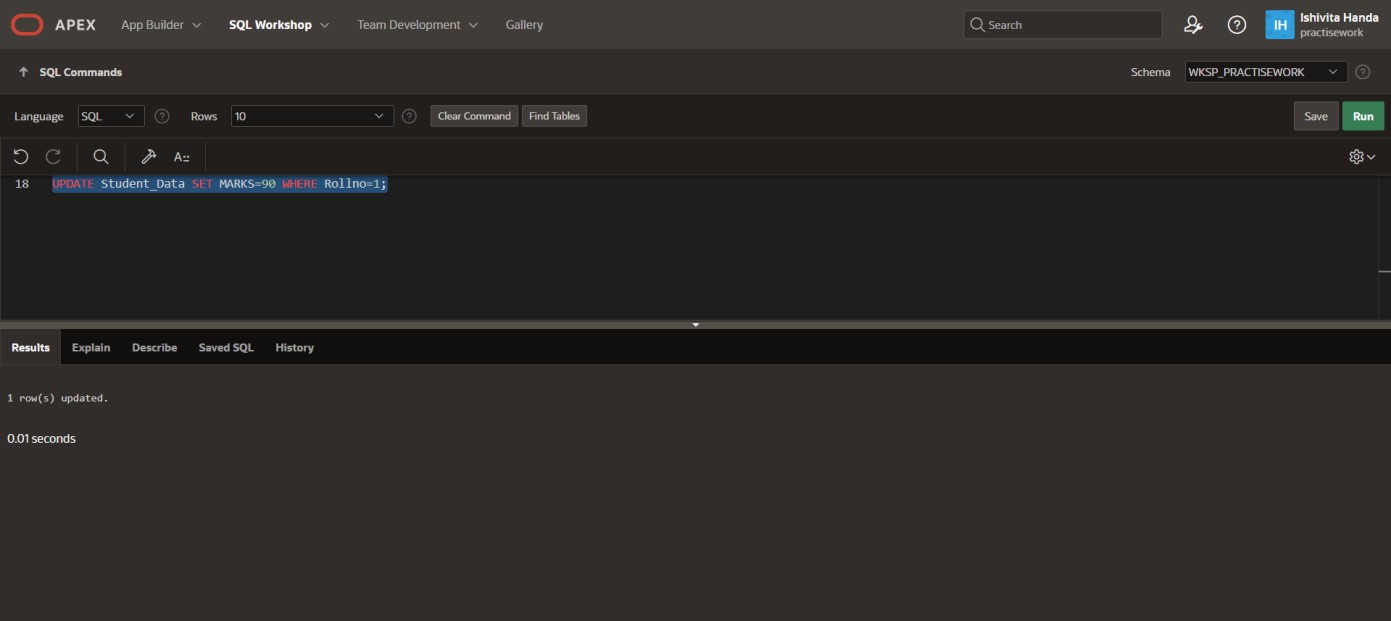


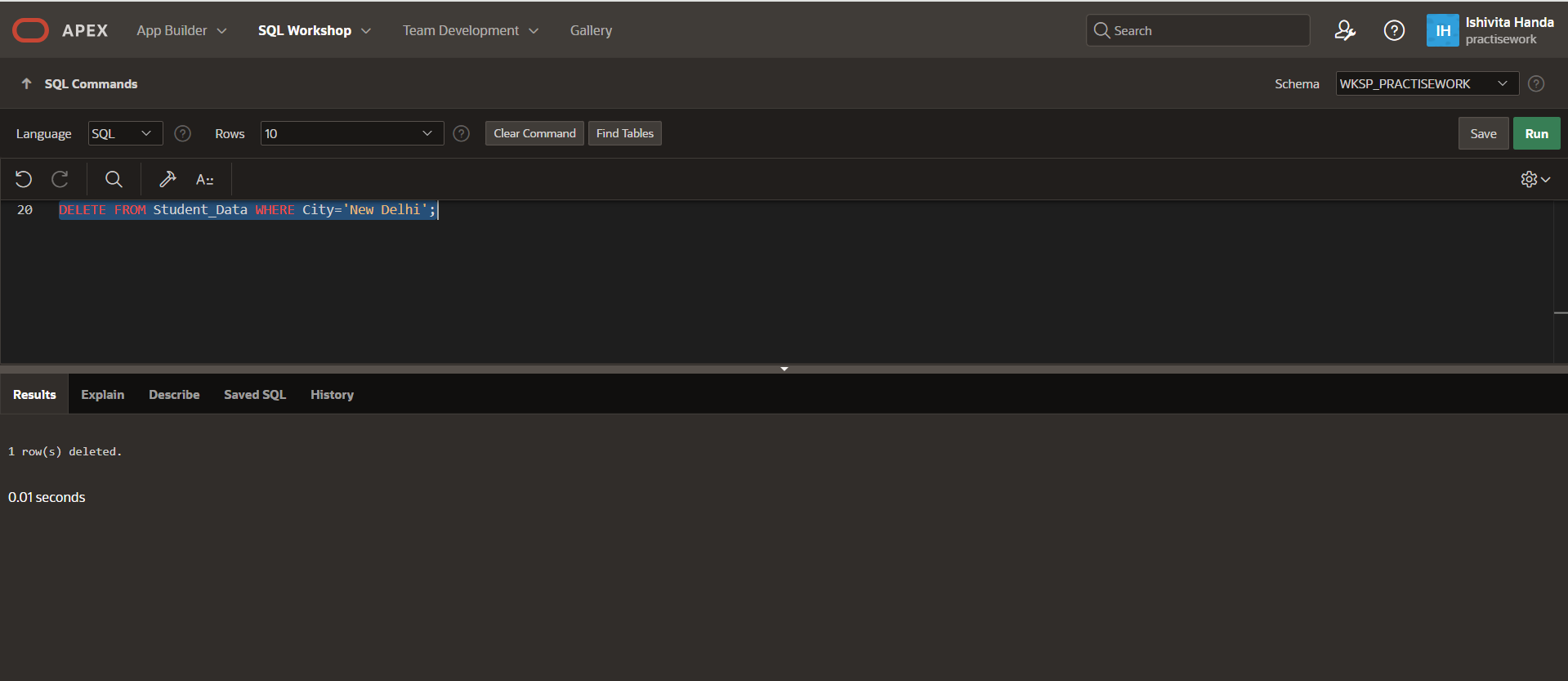
  
Q.5 Display rollno,name and class of Noida students

Q.6 Display all the information in descending of marks

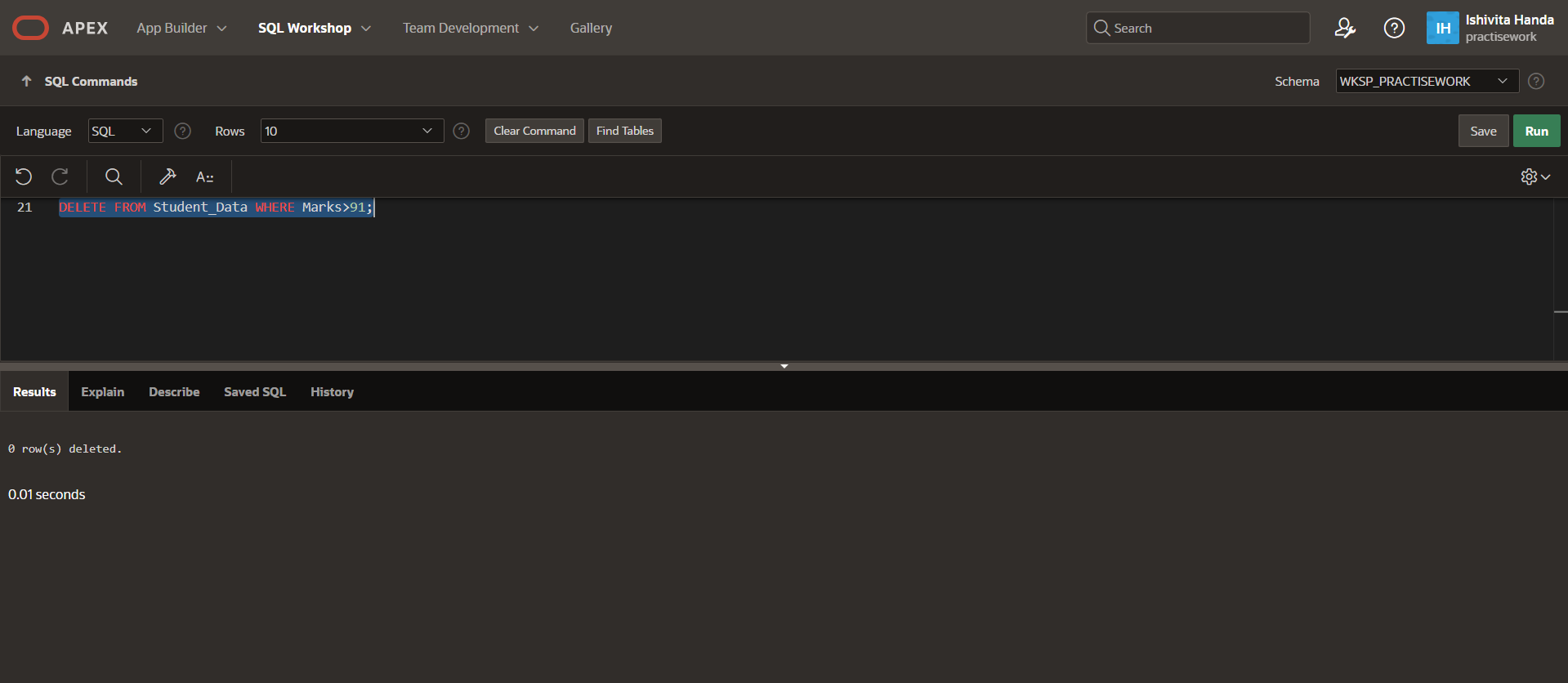


  
Q.7 Change the marks of roll no 5 to 89

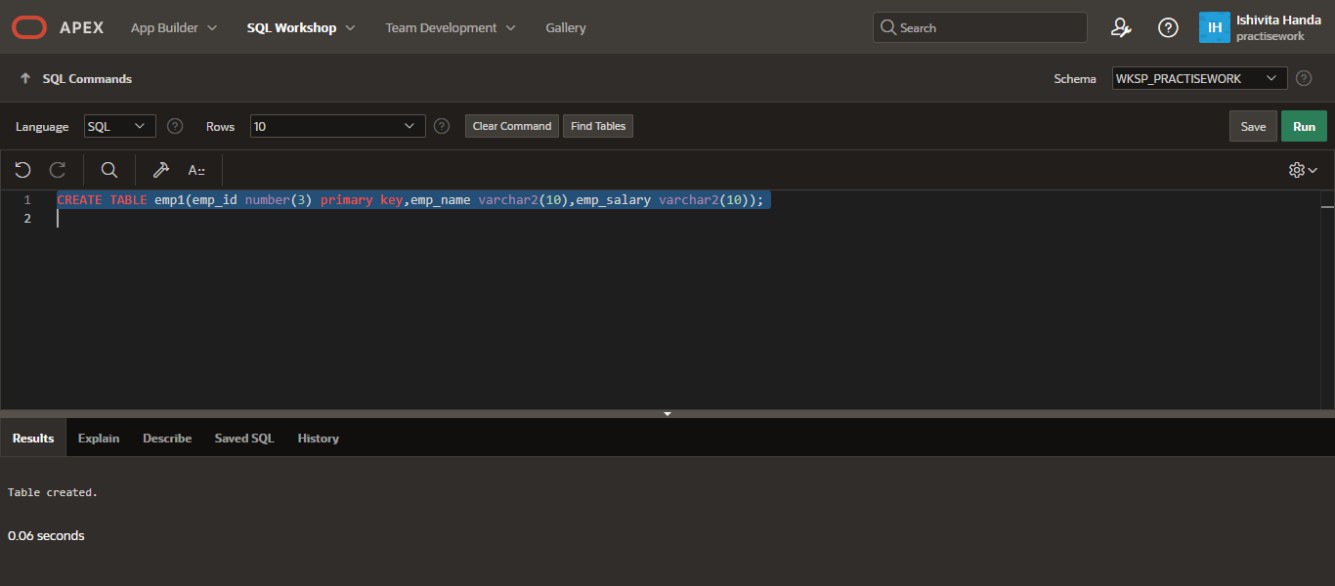
  
Q.8 Change the name and city of roll no 3

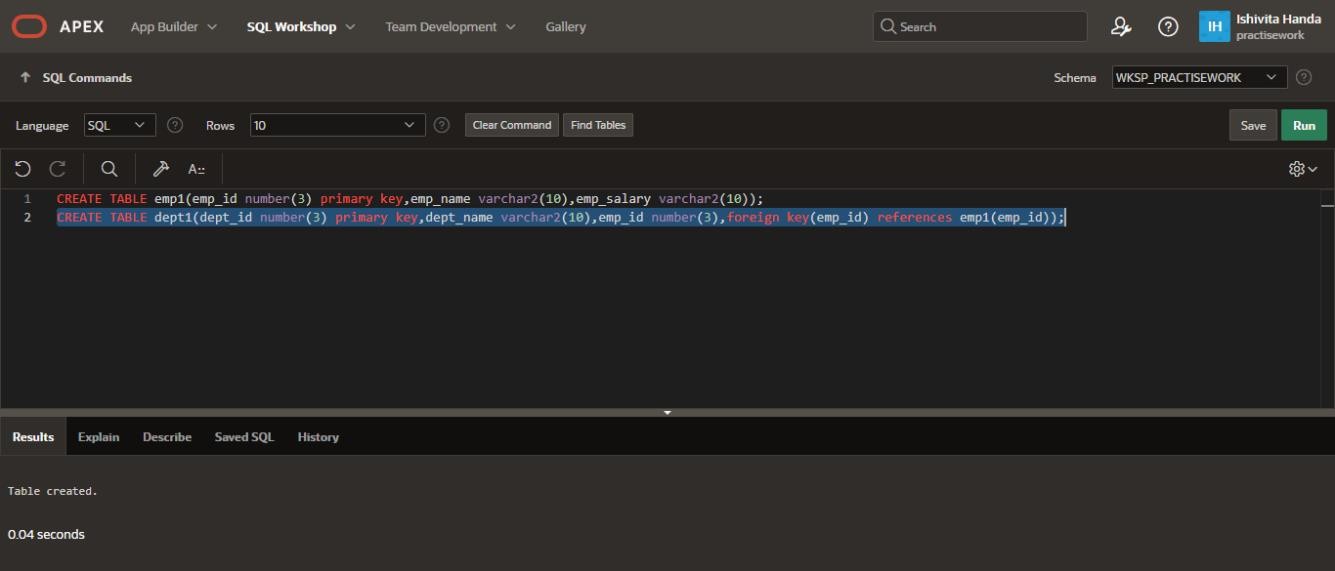
  
Q.9 Delete the names of Noida students

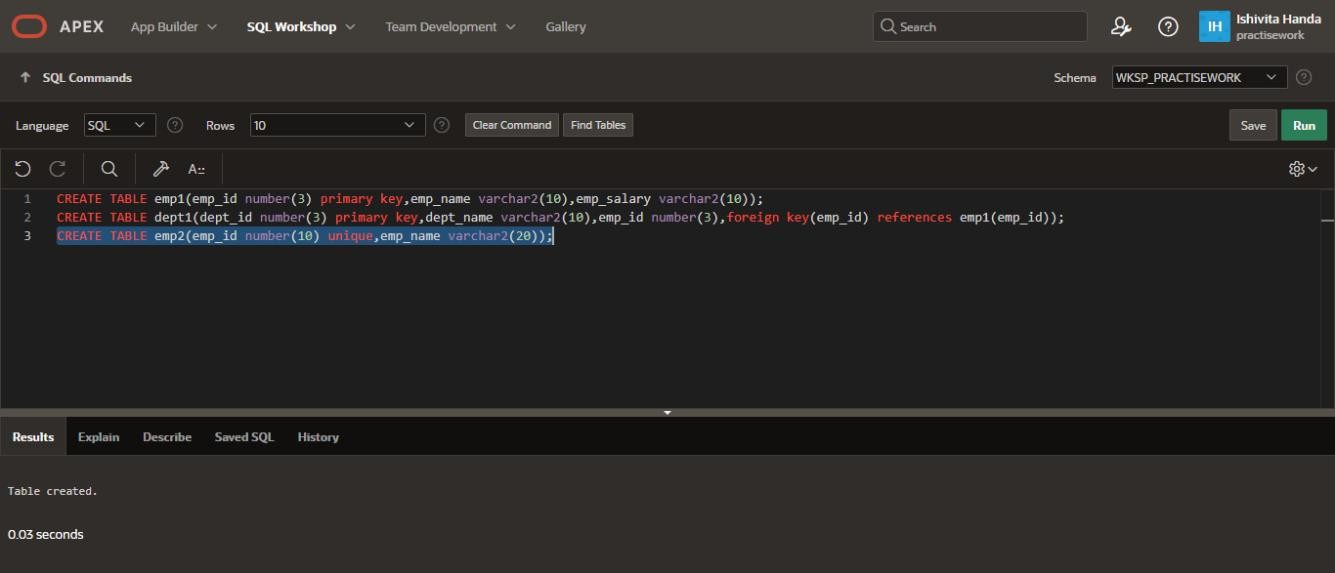
Q.10 Delete the records of students where marks is less than 80.



**Lab Assignment 2**

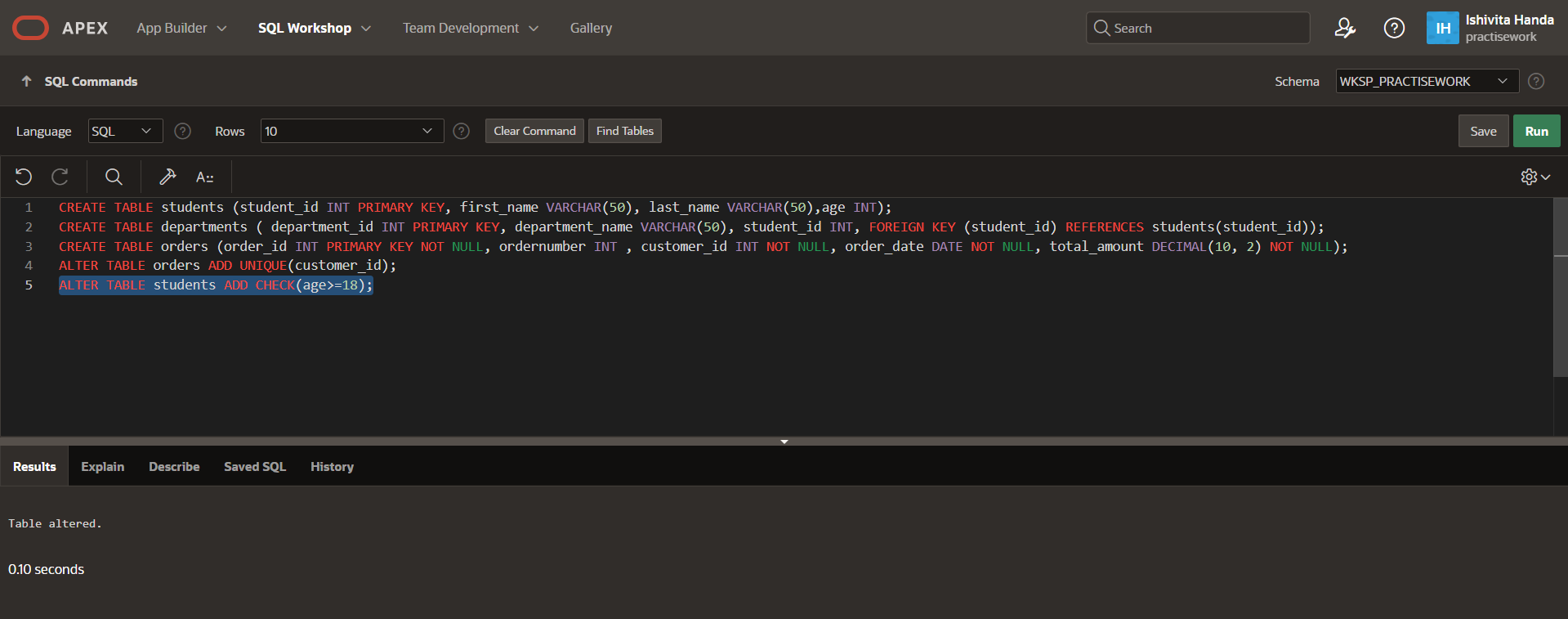
Q1.Create table emp1 having primary key as emp1(emp\_id primary key, emp\_name , emp\_salary )

Q2. Create table dept1with foreign key (dept\_id number(3) primary key,dept\_name varchar2(10),emp\_id foreign key(emp\_id) references emp1(emp\_id));

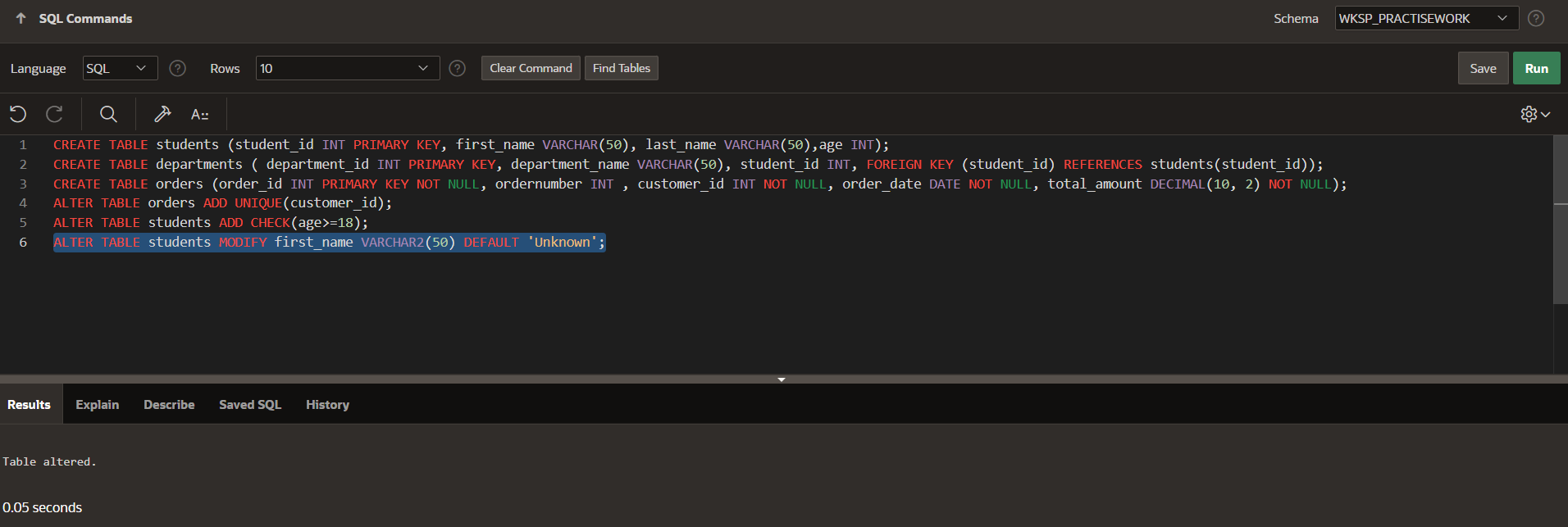
Q3. Create table emp having one unique column as (emp\_id unique , emp\_name)

Q4. Create table student with NOT NULL constraint

Q5. Apply CHECK constrant in student table.

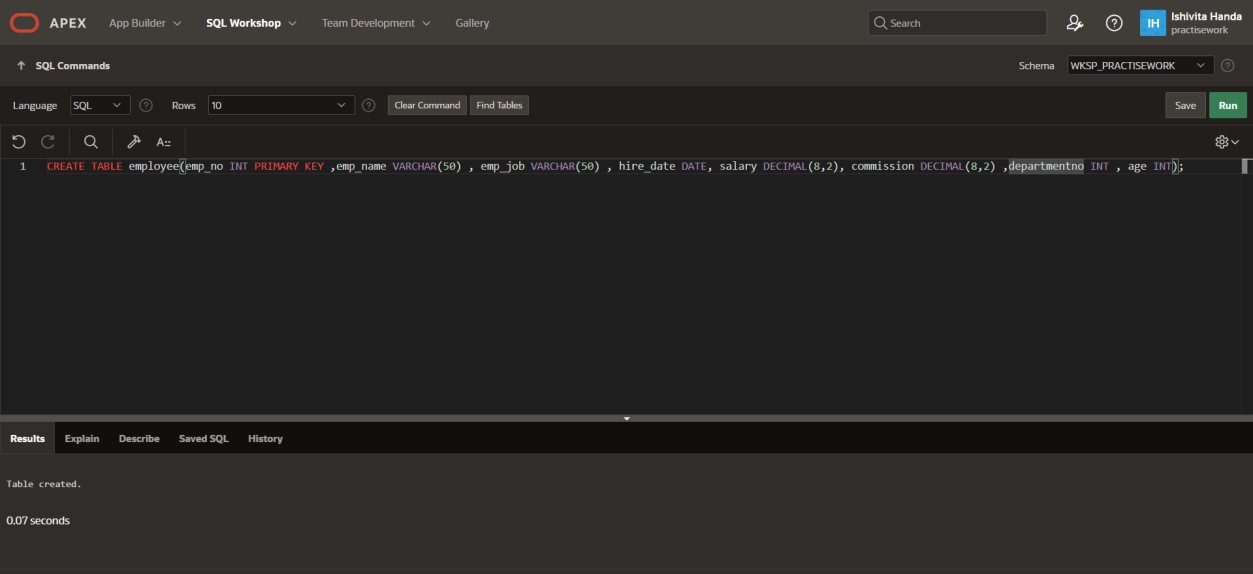


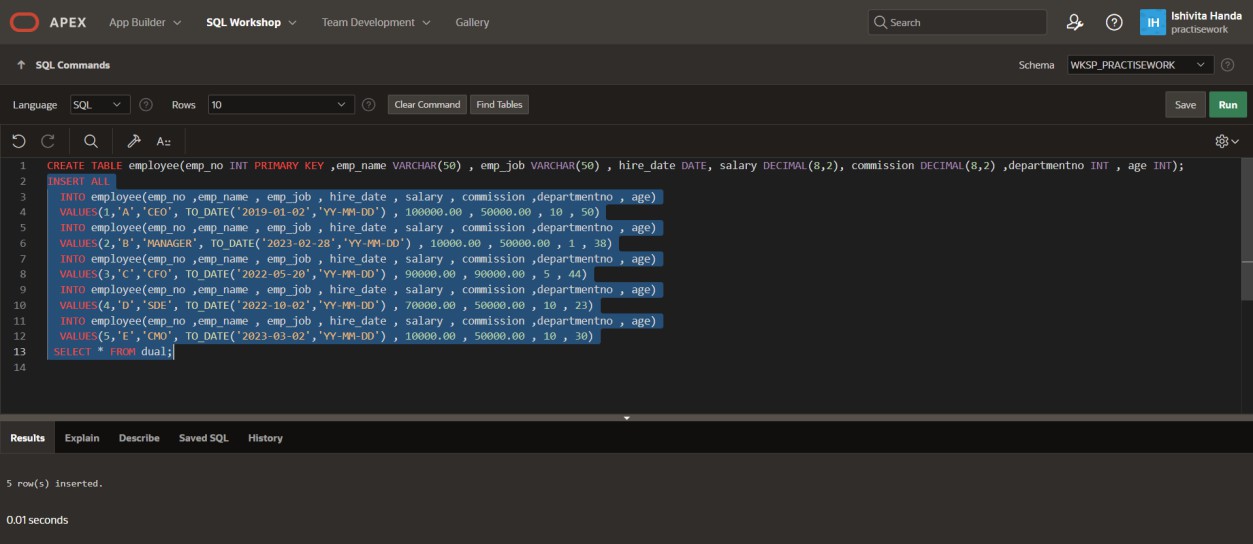
Q6. Apply DEFAULT constraint in student table.



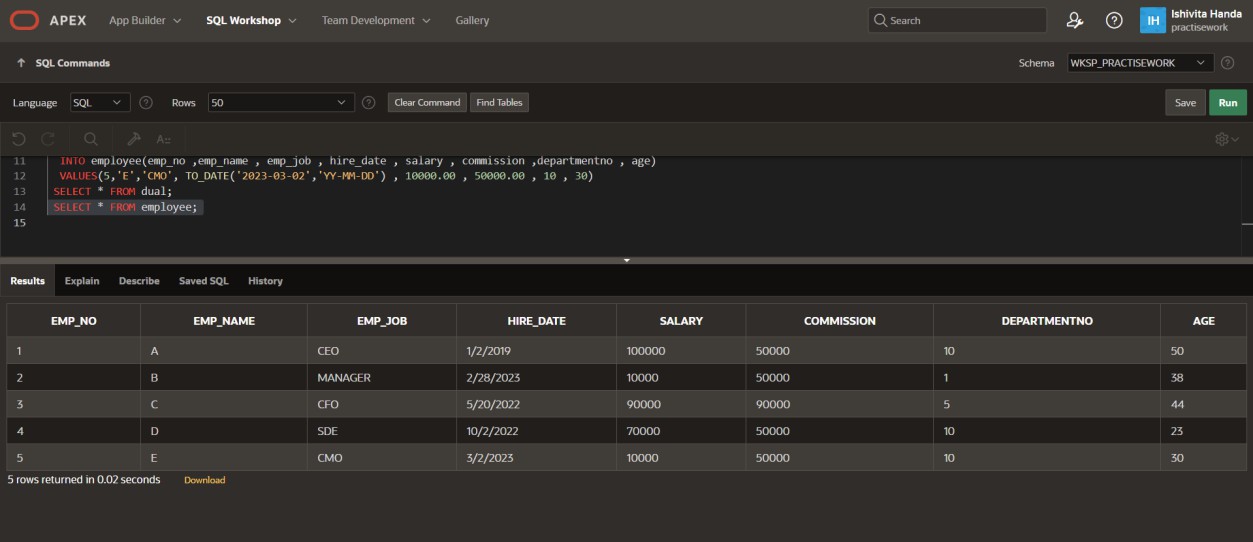
**Lab Assignment 3**

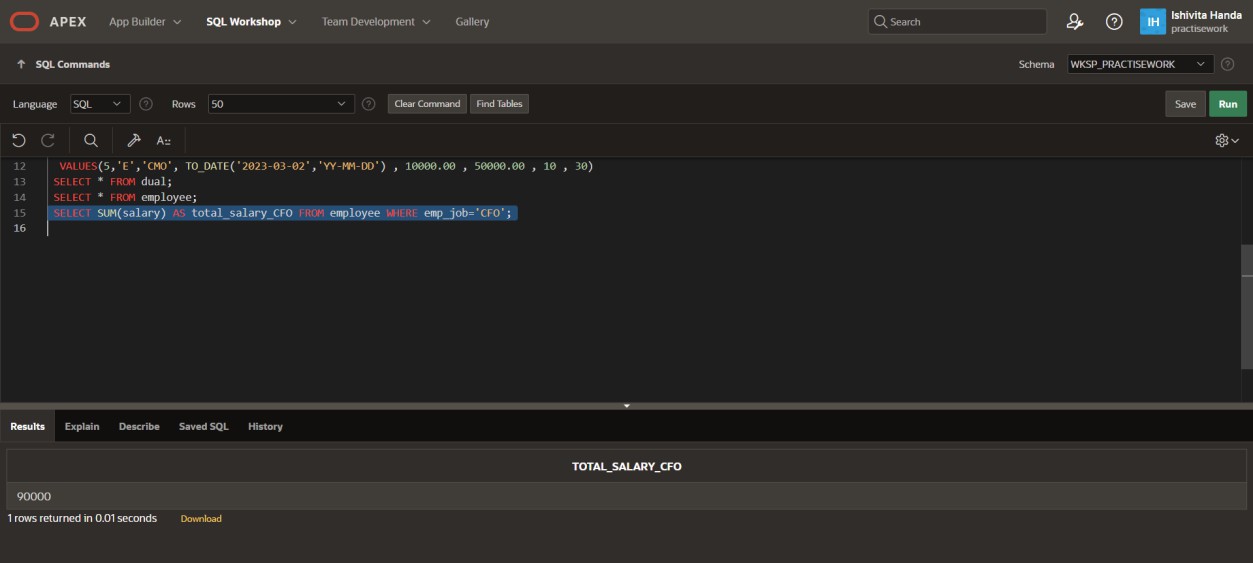
Q1.Create a table employee (empno., empname, job, hire\_date, salary , commission , department no., age).



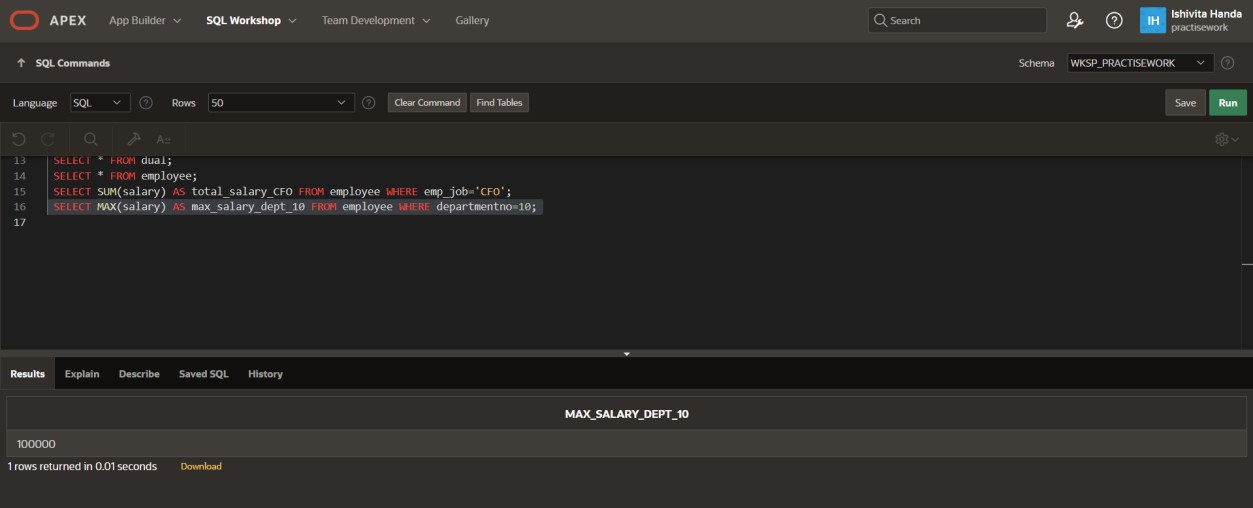
Q2.Insert any 5 records

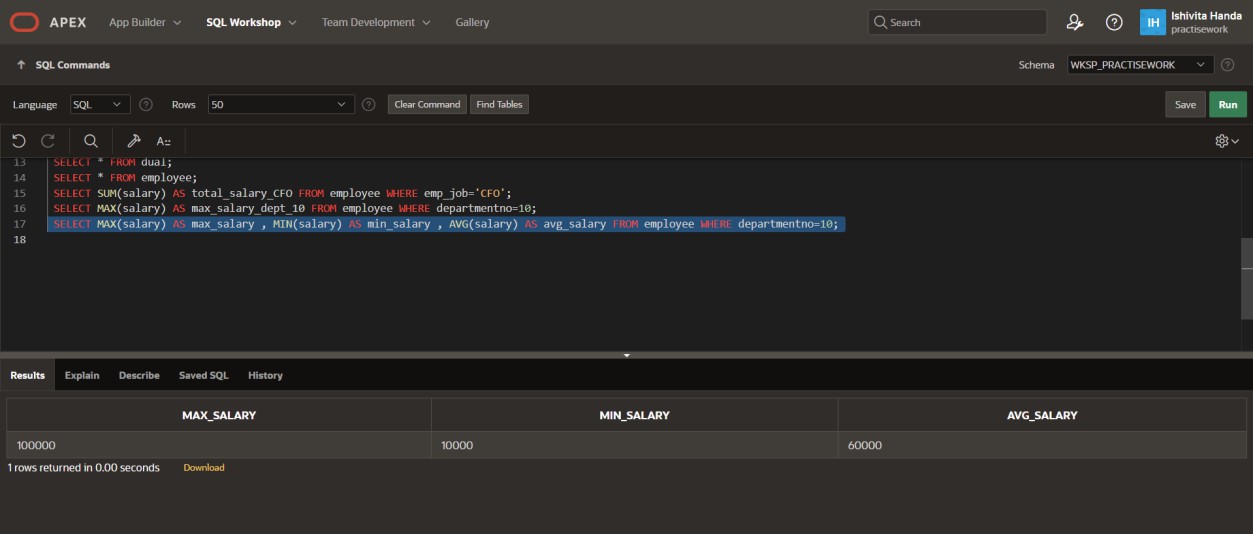
Q3.Display the info of all employee

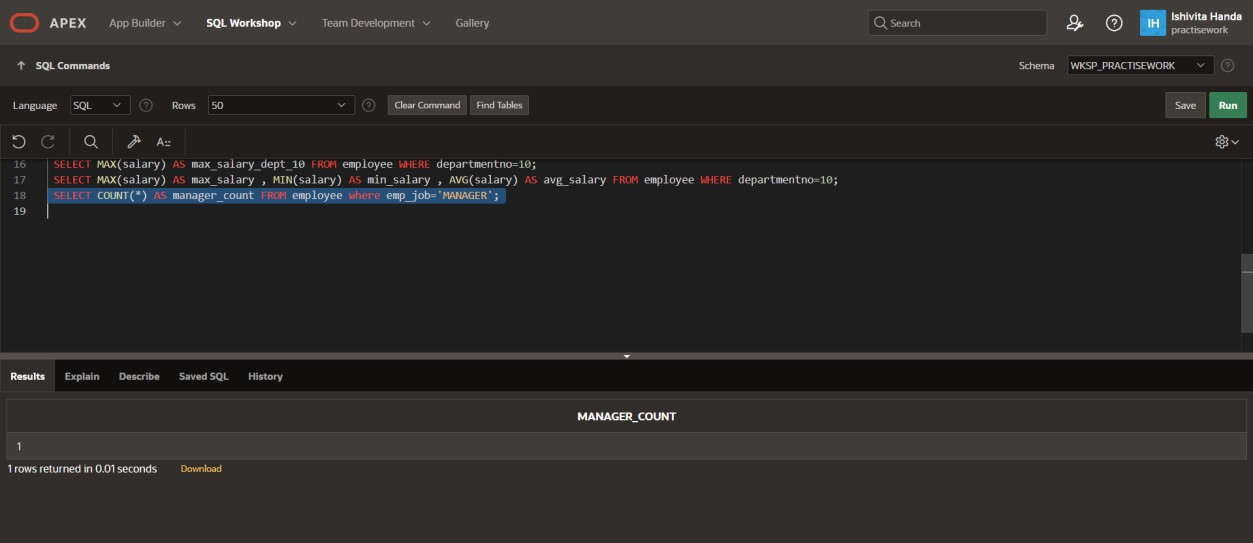


Q4.Display total salary paid to CFO

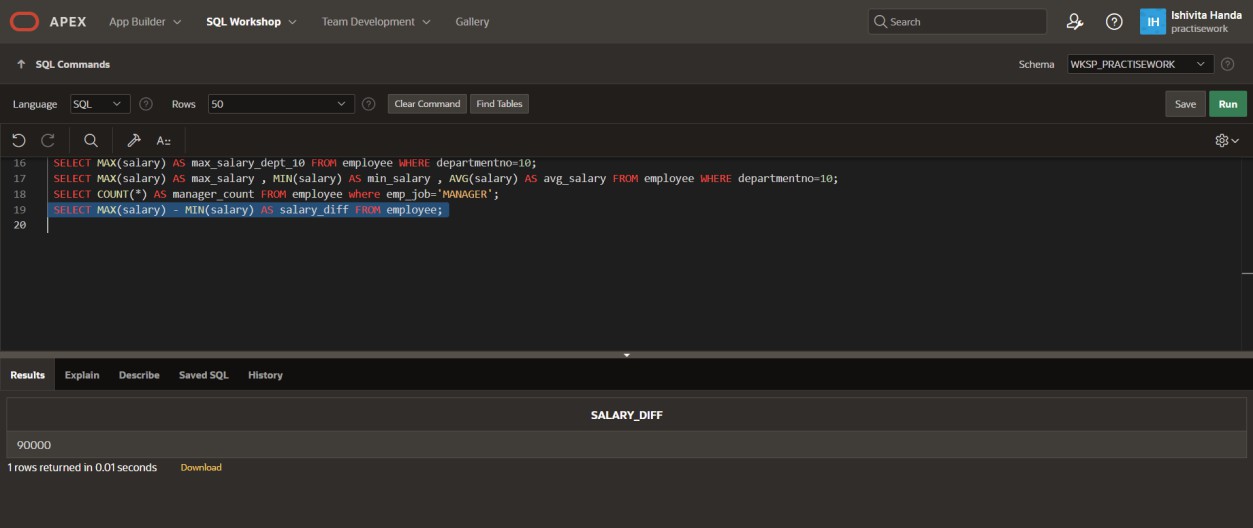
Q5.Display max salary of employee who belongs to department no.=10

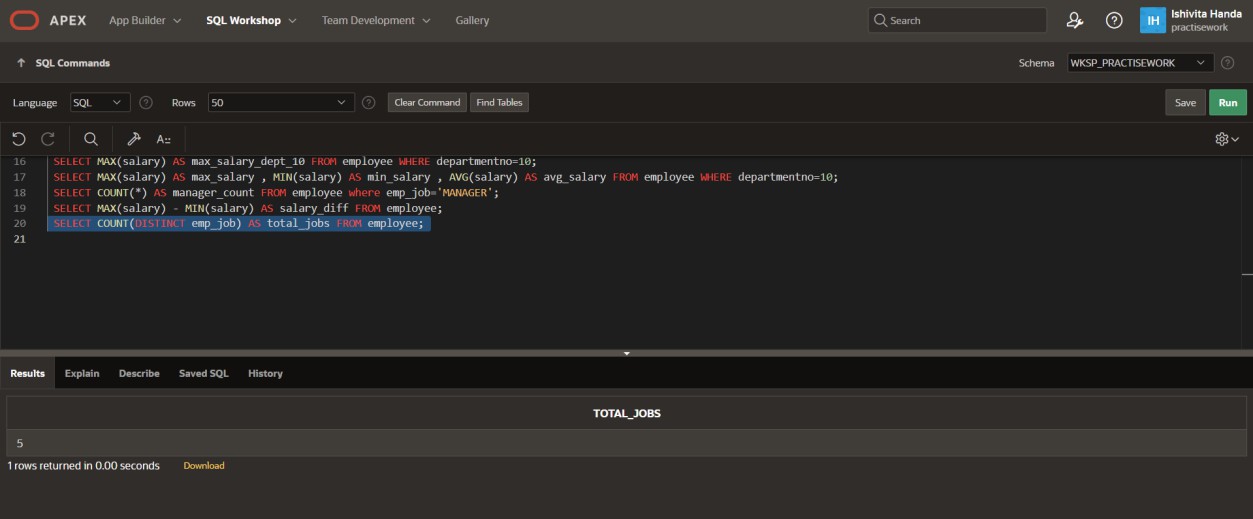


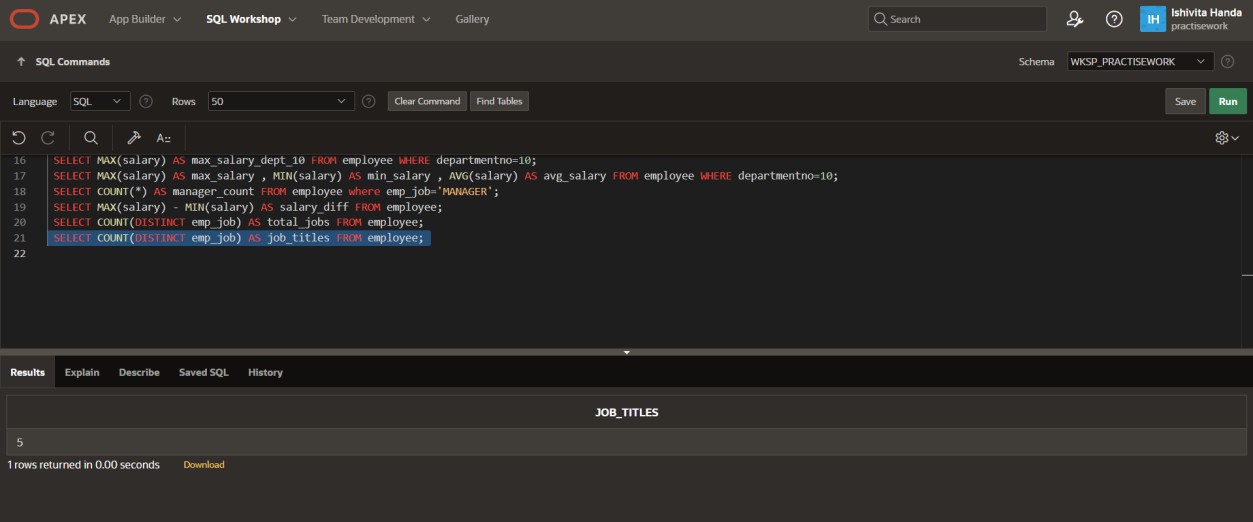
Q6.Display max,min,average salary of department no.=10

Q7.Count no. of employee where job is manager

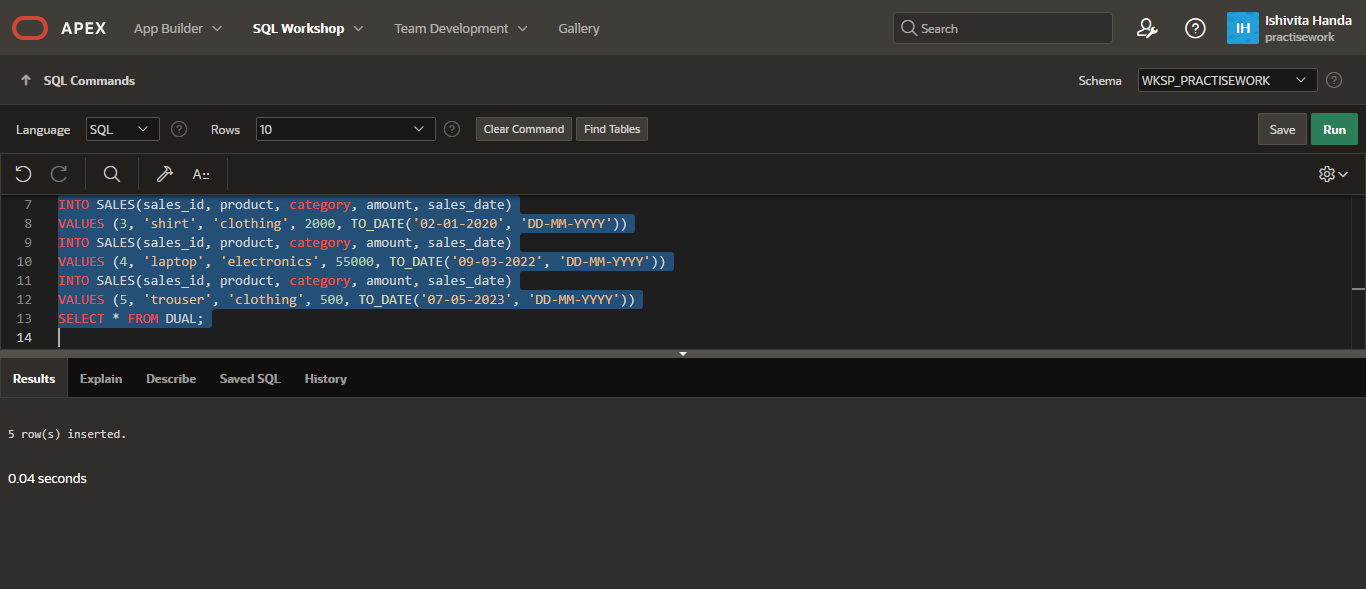
Q8. What is the difference between maximum and minimum salary of employee in organisation?



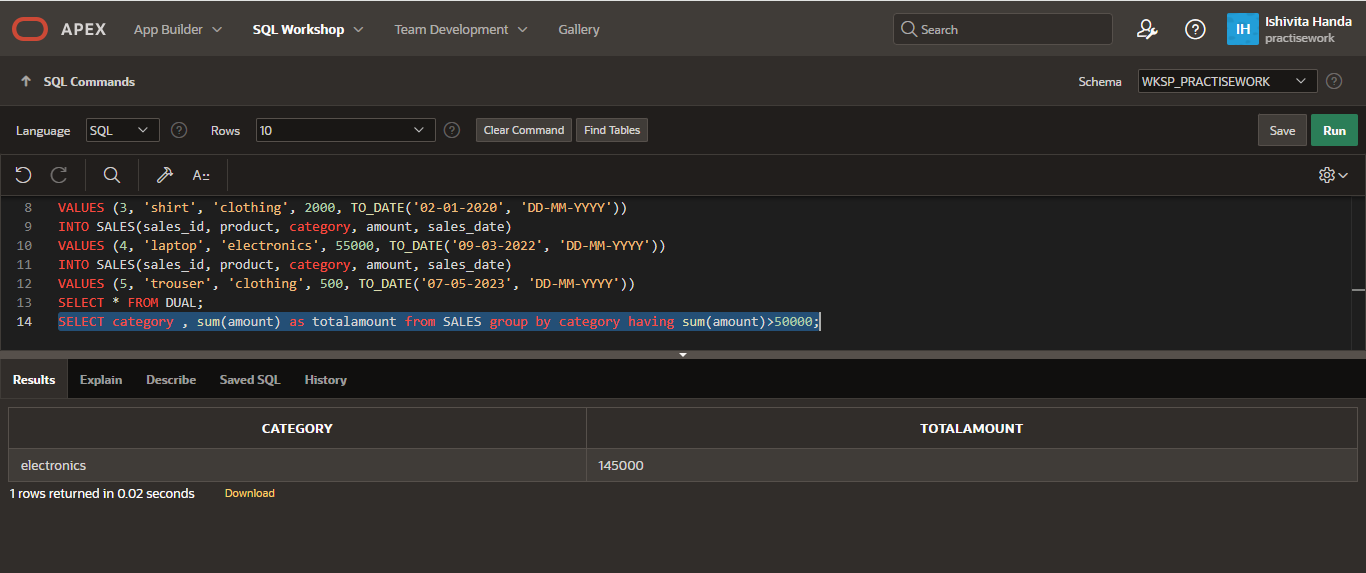
Q9. How many jobs are available in organisation?

Q10. How many job titles are available?

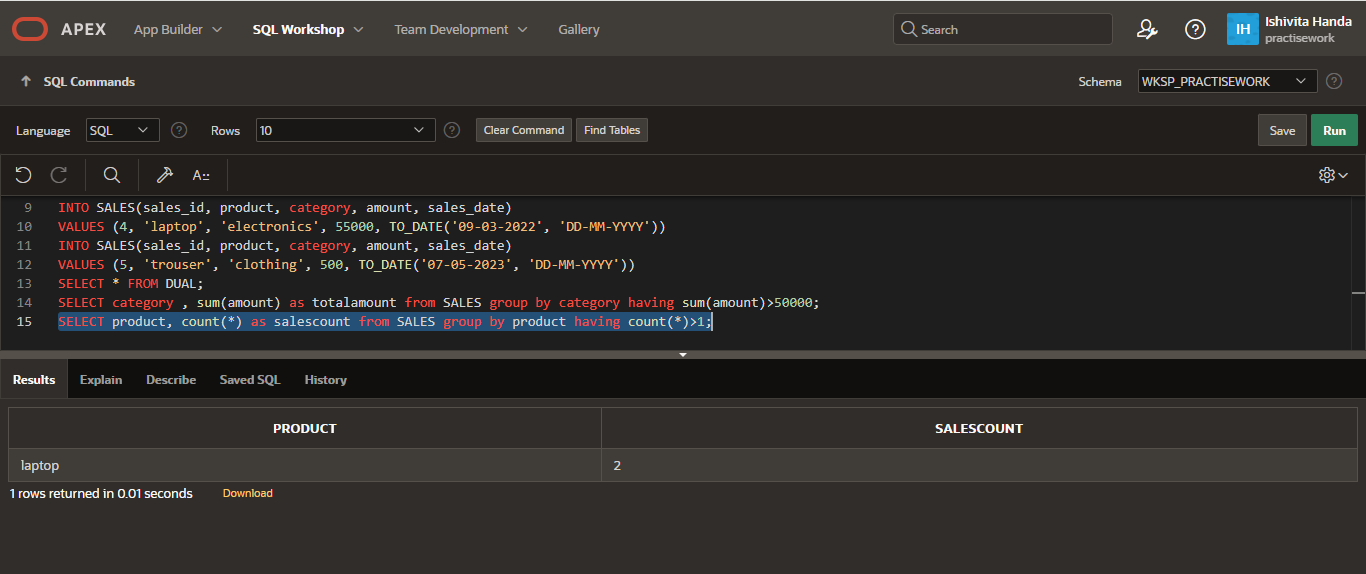
**Lab Assignment 4**



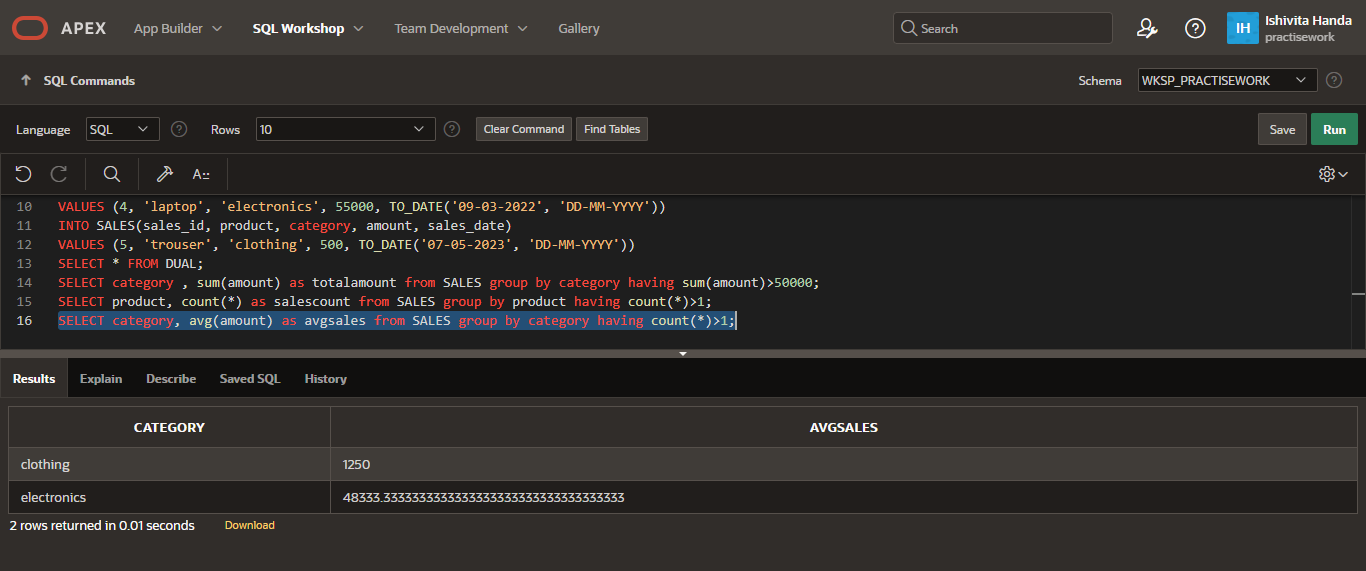
Q1. You have a table called with the following structure:   
(SaleID,Product,Category,Amount,SaleDate) , (1,Laptop,Electronics,50000,2024-01-15) ,  
(2,TV,Electronics,40000,2024-01-18) , (3,Shirt,Clothing,2000,2024-01-20) ,   
(4,Laptop,Electronics,55000,2024-02-05) , (5,Shirt,Clothing,2500,2024-02-10)



Q2. Find products that have been sold more than once.

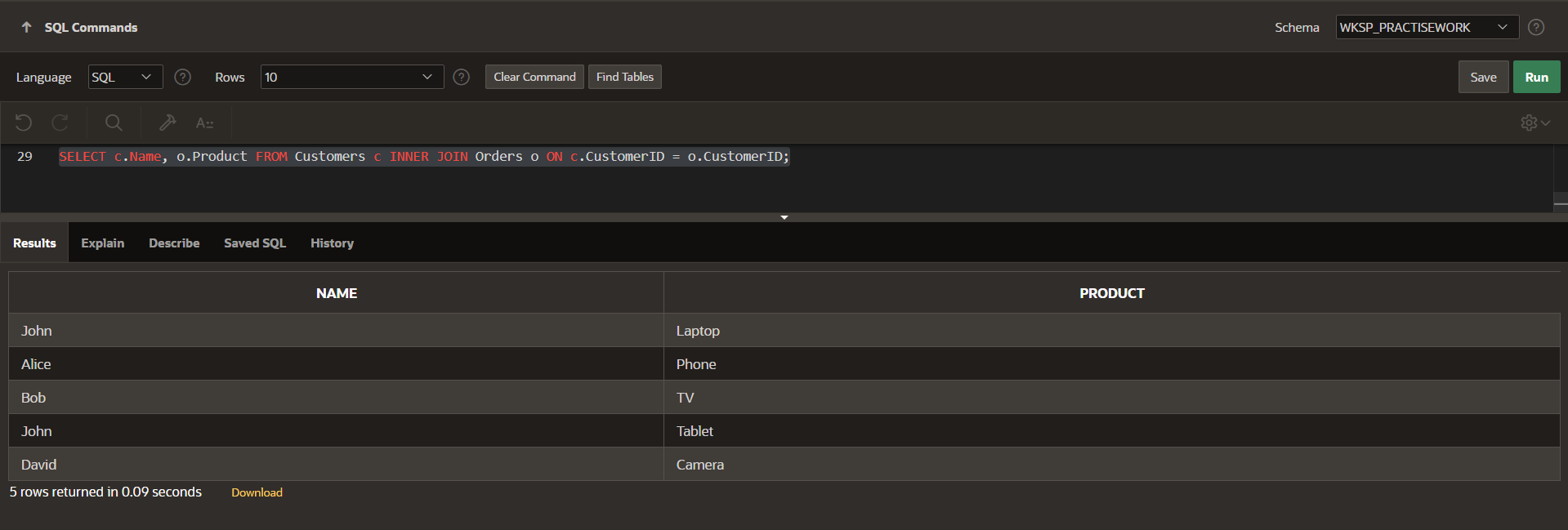


Q3. Find the average sale amount per category, but only for categories with more than one sale.

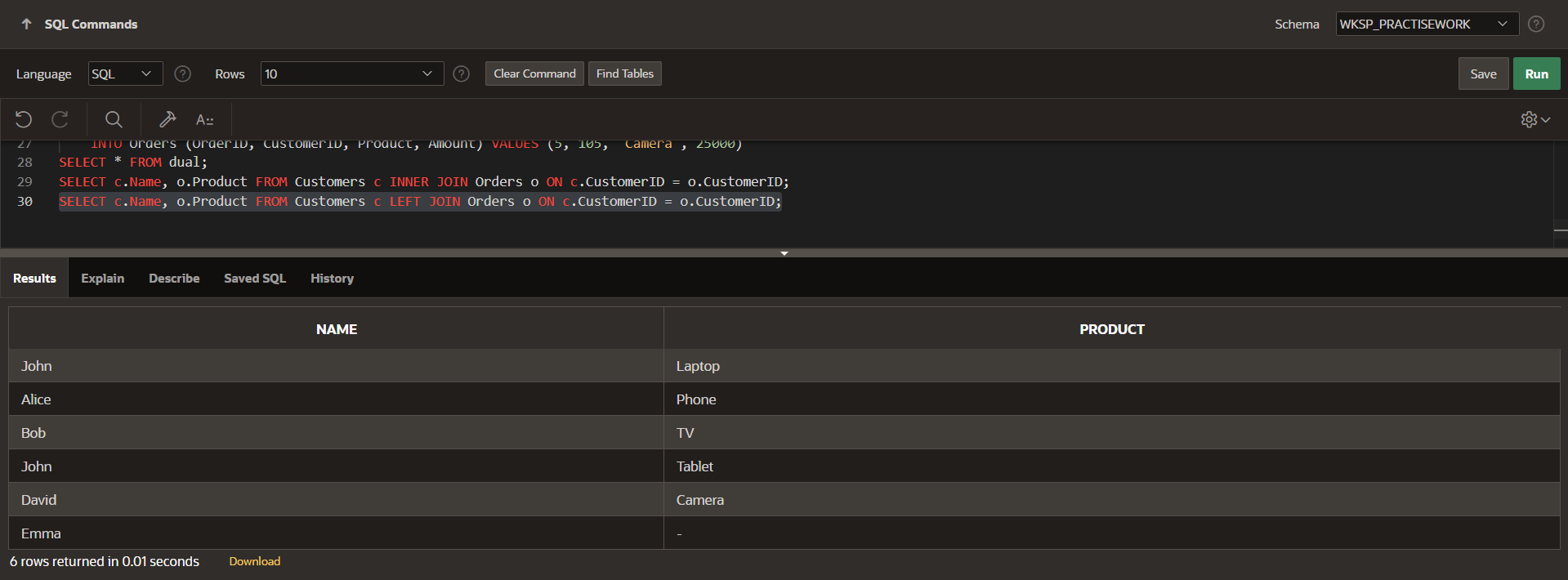


**Lab Assignment 5**

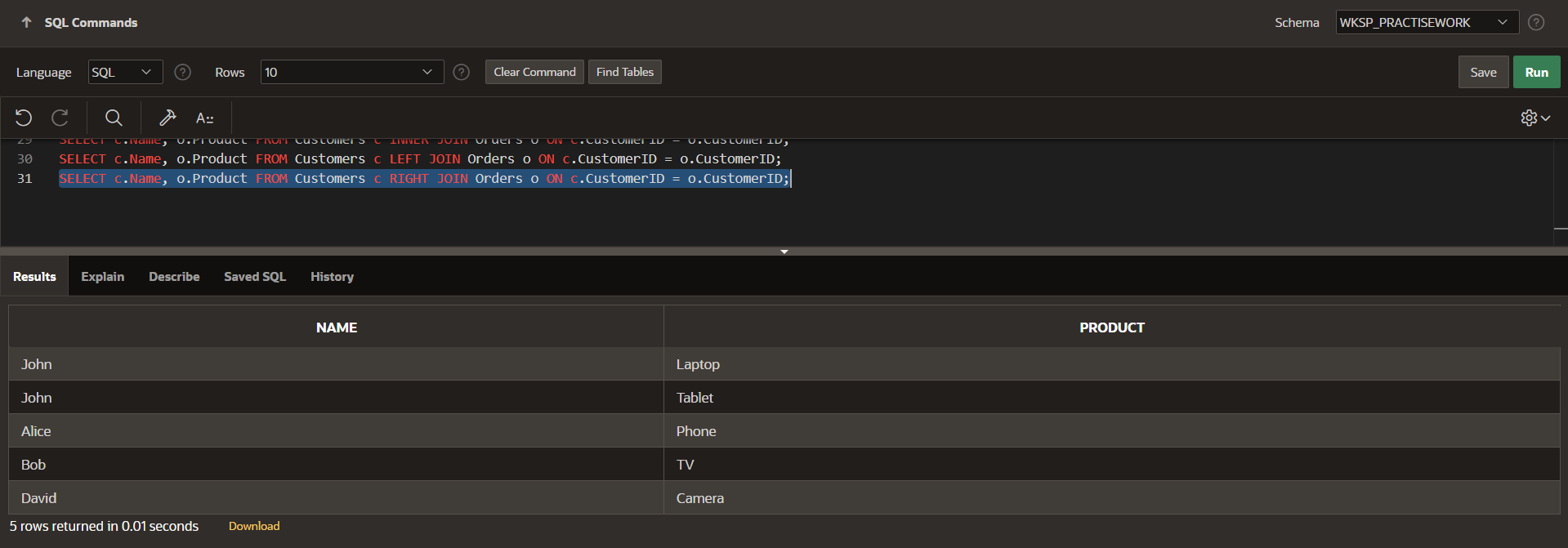
Q1. Retrieve the names of customers who have placed an order along with the product they bought.



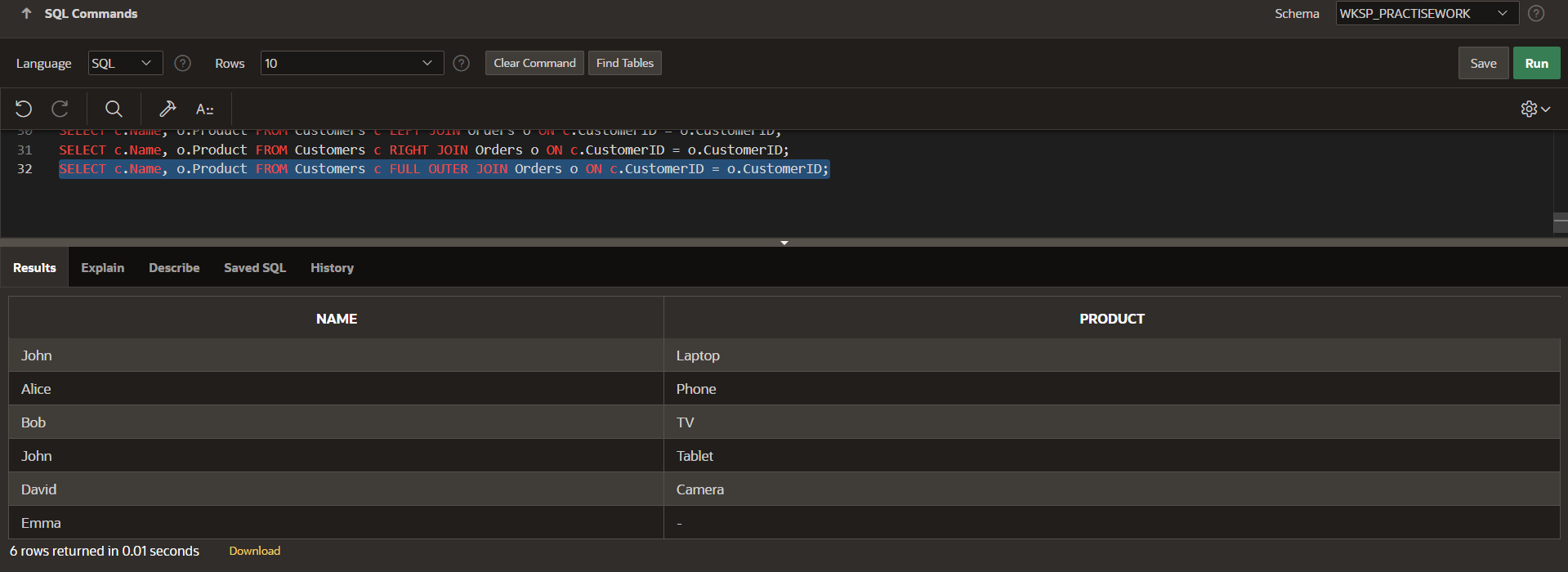
Q2. Retrieve all customers and their orders, even if they haven’t placed any order.



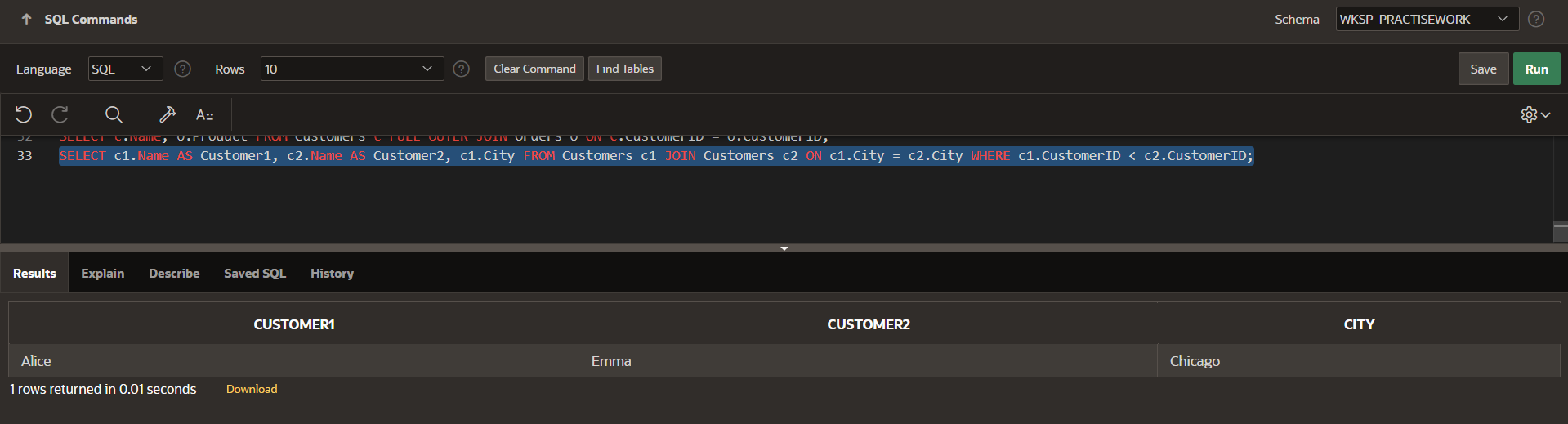
Q3. Retrieve all orders and their customers, including orders placed by unknown customers.



Q4. Retrieve all customers and all orders, even if there is no match.



Q5. Retrieve pairs of customers from the same city.



Q6. Find the total amount spent by each customer.

