

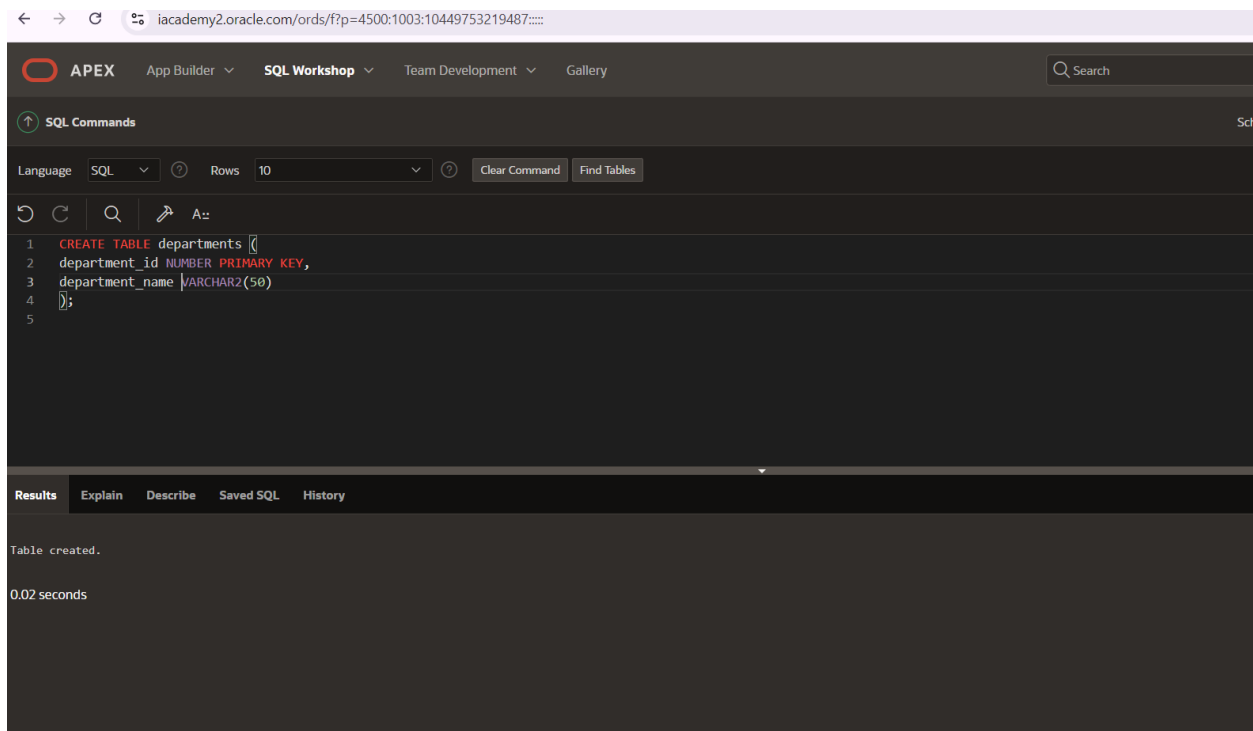
# PL\_SQL APEX PRACTICE TRIGGERS

- Shaik sumiya

192372090

Write a code in PL/SQL to develop a trigger that enforces referential integrity by preventing the deletion of a parent record if child records exist.

Create departments table:

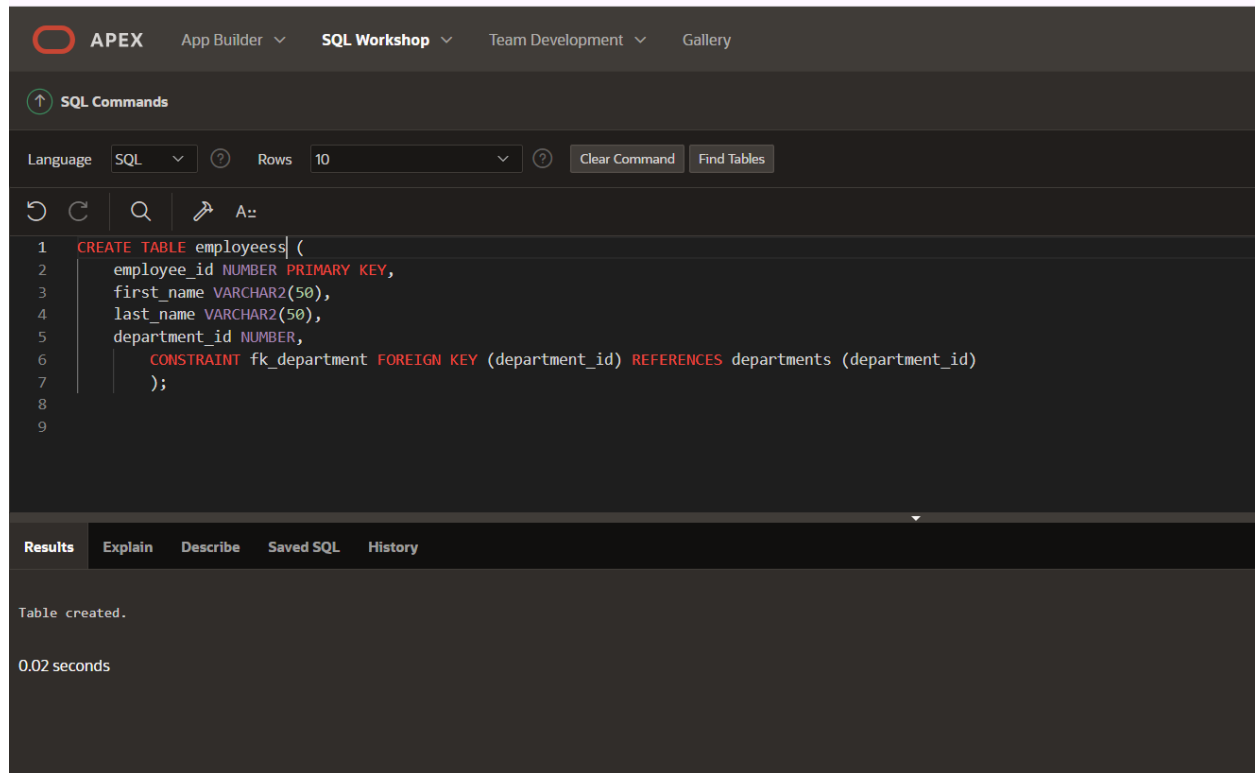


The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. A search bar is on the right. Below the navigation bar, the 'SQL Commands' tab is active. The 'Language' is set to 'SQL' and 'Rows' is set to '10'. The SQL command entered is:

```
1 CREATE TABLE departments (  
2   department_id NUMBER PRIMARY KEY,  
3   department_name VARCHAR2(50)  
4 );  
5
```

The 'Results' tab is selected at the bottom, showing the message 'Table created.' and the execution time '0.02 seconds'.

## Create the 'employeeess' table with a foreign key reference

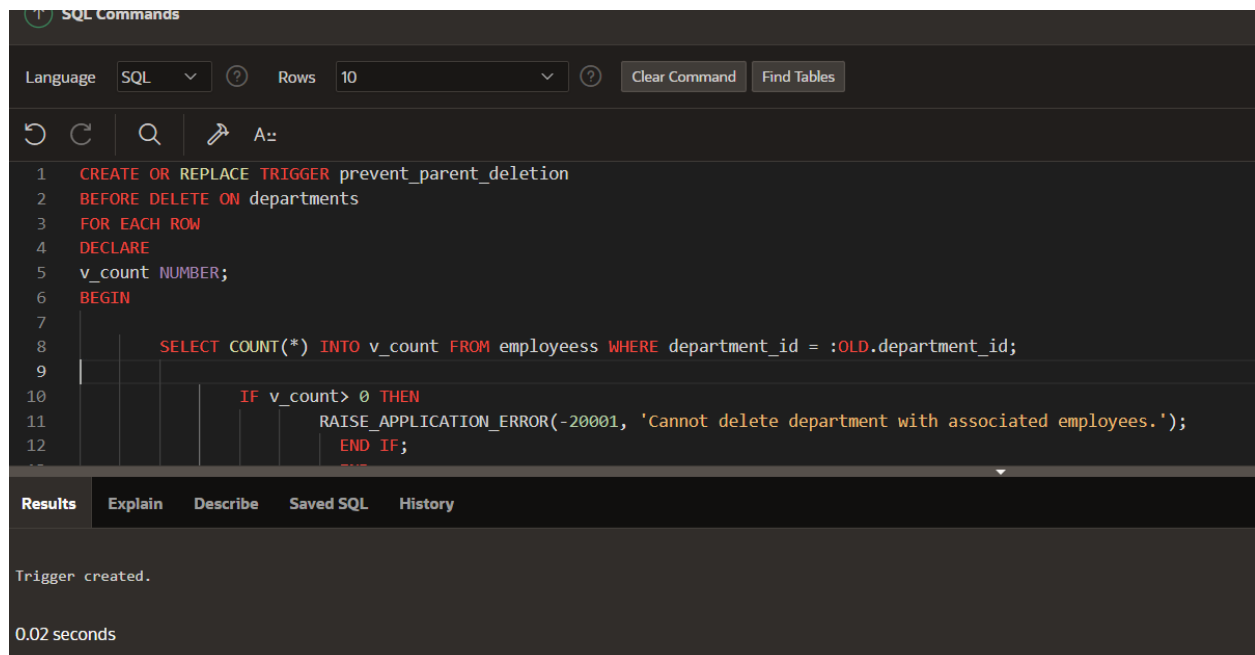


The screenshot shows the APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The 'SQL Commands' tab is active, showing a list of commands with '1' selected. The 'Language' is set to 'SQL' and 'Rows' is set to '10'. The 'Clear Command' and 'Find Tables' buttons are visible. The SQL command being executed is:

```
1 CREATE TABLE employeeess (  
2     employee_id NUMBER PRIMARY KEY,  
3     first_name VARCHAR2(50),  
4     last_name VARCHAR2(50),  
5     department_id NUMBER,  
6     CONSTRAINT fk_department FOREIGN KEY (department_id) REFERENCES departments (department_id)  
7 );  
8  
9
```

The 'Results' tab is selected, showing the message 'Table created.' and the execution time '0.02 seconds'.

## Create a trigger to enforce referential integrity



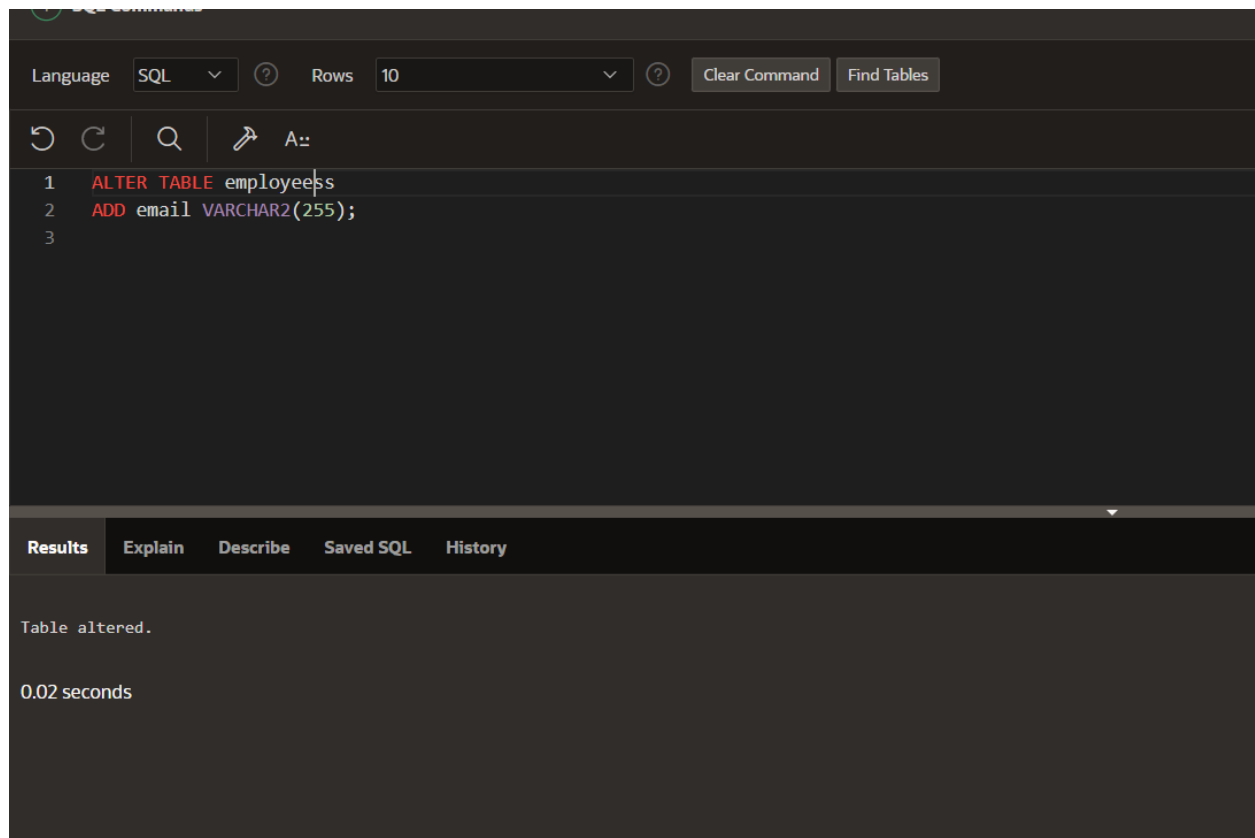
The screenshot shows the APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The 'SQL Commands' tab is active, showing a list of commands with '1' selected. The 'Language' is set to 'SQL' and 'Rows' is set to '10'. The 'Clear Command' and 'Find Tables' buttons are visible. The SQL command being executed is:

```
1 CREATE OR REPLACE TRIGGER prevent_parent_deletion  
2 BEFORE DELETE ON departments  
3 FOR EACH ROW  
4 DECLARE  
5     v_count NUMBER;  
6 BEGIN  
7  
8     SELECT COUNT(*) INTO v_count FROM employeeess WHERE department_id = :OLD.department_id;  
9  
10    IF v_count > 0 THEN  
11        RAISE_APPLICATION_ERROR(-20001, 'Cannot delete department with associated employees.');
```

The 'Results' tab is selected, showing the message 'Trigger created.' and the execution time '0.02 seconds'.

Write a code in PL/SQL to create a trigger that checks for duplicate values in a specific column and raises an exception if found.

Add a column email to the employeeess table



The screenshot shows a SQL IDE interface. At the top, there's a header bar with 'Language' set to 'SQL', 'Rows' set to '10', and buttons for 'Clear Command' and 'Find Tables'. Below this is a toolbar with icons for undo, redo, search, and a keyboard shortcut 'A:'. The main editor area contains the following SQL code:

```
1 ALTER TABLE employeeess
2 ADD email VARCHAR2(255);
3
```

At the bottom, there's a 'Results' tab selected, showing the message 'Table altered.' and the execution time '0.02 seconds'.

SQL Commands

Language SQL Rows 10 Clear Command Find Tables

A::

1

CREATE OR REPLACE TRIGGER check\_for\_duplicate\_email

2

BEFORE INSERT OR UPDATE ON employeeess

3

FOR EACH ROW

4

DECLARE

5

v\_count NUMBER;

6

BEGIN

7

8

SELECT COUNT(\*)

9

INTO v\_count

10

FROM employeeess

11

WHERE email = :NEW.email

12

AND employee\_id != :NEW.employee\_id;

13

Results

Explain

Describe

Saved SQL

History

Trigger created.

0.03 seconds

3<sup>rd</sup>

↑ SQL Commands

Language SQL ? Rows 10 ? Clear Command Find Tables

↶ ↷ 🔍 📌 A::

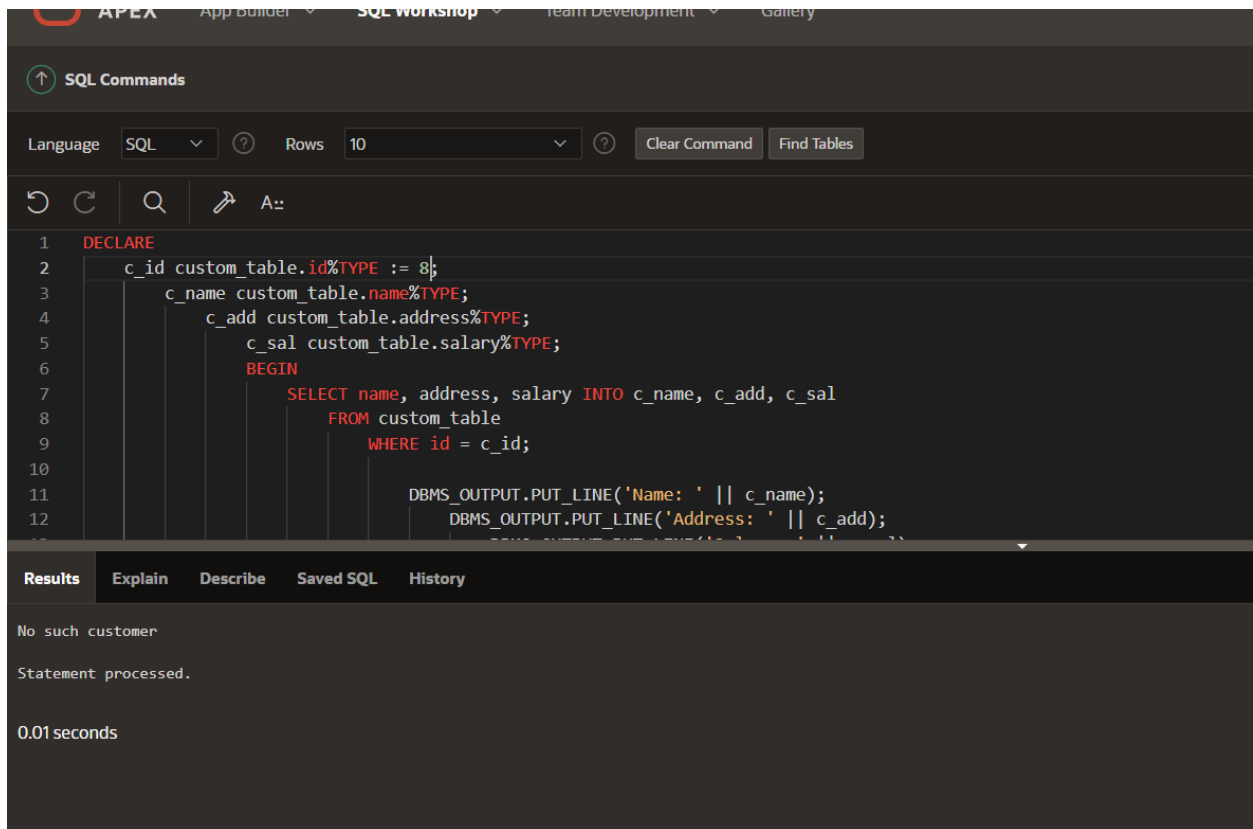
1 CREATE TABLE custom\_table (  
2       id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,  
3       name VARCHAR2(100),  
4       age NUMBER,  
5       address VARCHAR2(255),  
6       salary NUMBER(10, 2)  
7     );  
8  
9

Results Explain Describe Saved SQL History

Table created.

0.03 seconds

# Exception handling



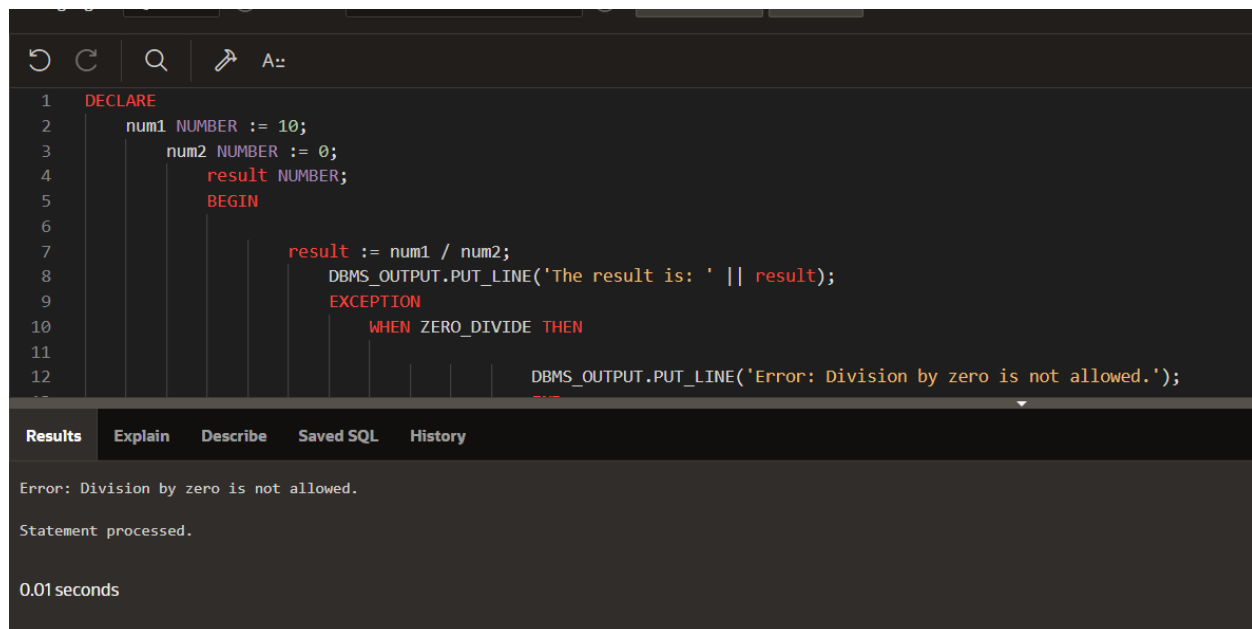
The screenshot displays the APEX SQL Workshop interface. At the top, there are navigation links: APEX, App Builder, SQL Workshop, Team Development, and Gallery. Below these, the 'SQL Commands' section is active, showing a command editor with the following SQL code:

```
1 DECLARE
2   c_id custom_table.id%TYPE := 8;
3   c_name custom_table.name%TYPE;
4   c_add custom_table.address%TYPE;
5   c_sal custom_table.salary%TYPE;
6   BEGIN
7     SELECT name, address, salary INTO c_name, c_add, c_sal
8     FROM custom_table
9     WHERE id = c_id;
10
11     DBMS_OUTPUT.PUT_LINE('Name: ' || c_name);
12     DBMS_OUTPUT.PUT_LINE('Address: ' || c_add);
```

Below the command editor, the 'Results' tab is selected, showing the output of the SQL command:

```
No such customer
Statement processed.
0.01seconds
```

Write a PL/SQL block to handle the exception when a division by zero occurs



```
1 DECLARE
2     num1 NUMBER := 10;
3     num2 NUMBER := 0;
4     result NUMBER;
5 BEGIN
6
7     result := num1 / num2;
8     DBMS_OUTPUT.PUT_LINE('The result is: ' || result);
9     EXCEPTION
10    WHEN ZERO_DIVIDE THEN
11
12        DBMS_OUTPUT.PUT_LINE('Error: Division by zero is not allowed.');
```

**Results** Explain Describe Saved SQL History

Error: Division by zero is not allowed.

Statement processed.

0.01seconds

Handle the INVALID\_NUMBER exception when converting a non-numeric value to a number.

```
12      WHEN VALUE_ERROR THEN  
13  
14  
15  
16  
17  
18
```

**Results** Explain Describe Saved SQL History

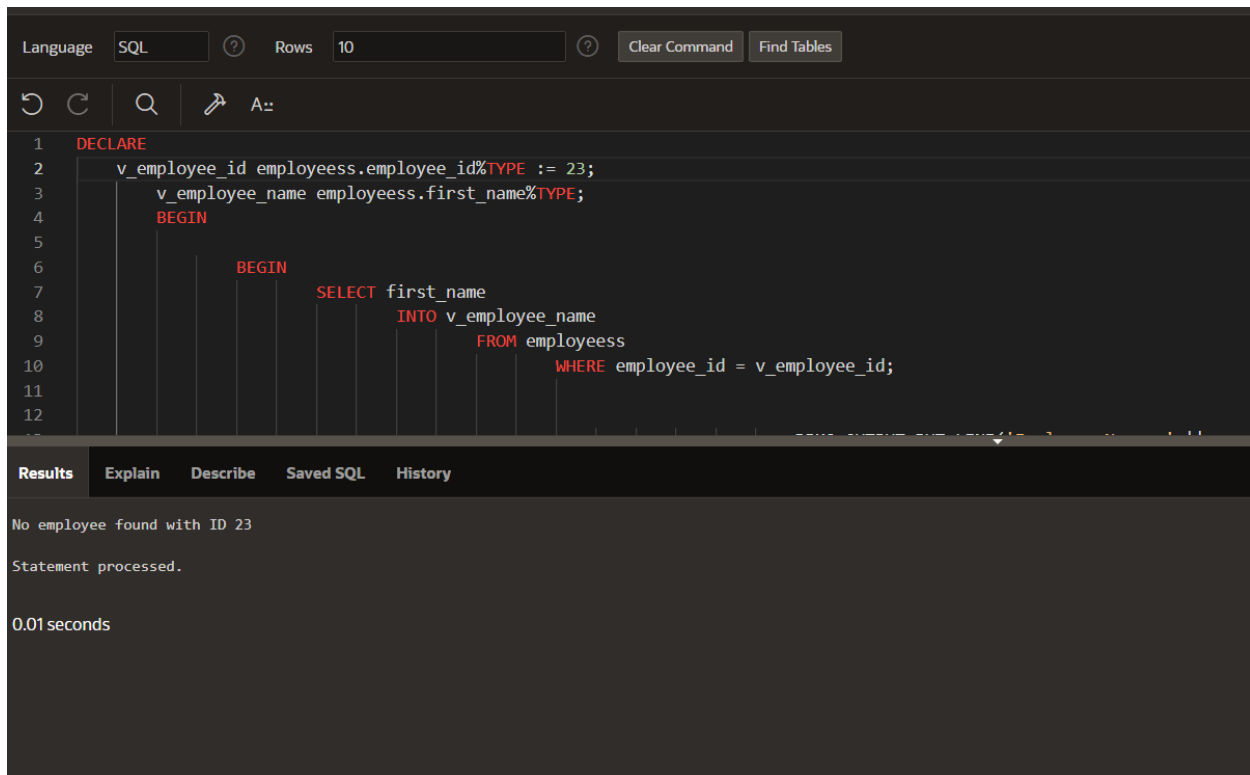
error: Invalid number. The value "abc" cannot be converted to a number.

statement processed.

.01 seconds



Handle the NO\_DATA\_FOUND exception when retrieving a row from a table and no matching record is found.



```
1 DECLARE
2   v_employee_id employees.employee_id%TYPE := 23;
3   v_employee_name employees.first_name%TYPE;
4 BEGIN
5
6   BEGIN
7     SELECT first_name
8     INTO v_employee_name
9     FROM employees
10    WHERE employee_id = v_employee_id;
11
12  
```

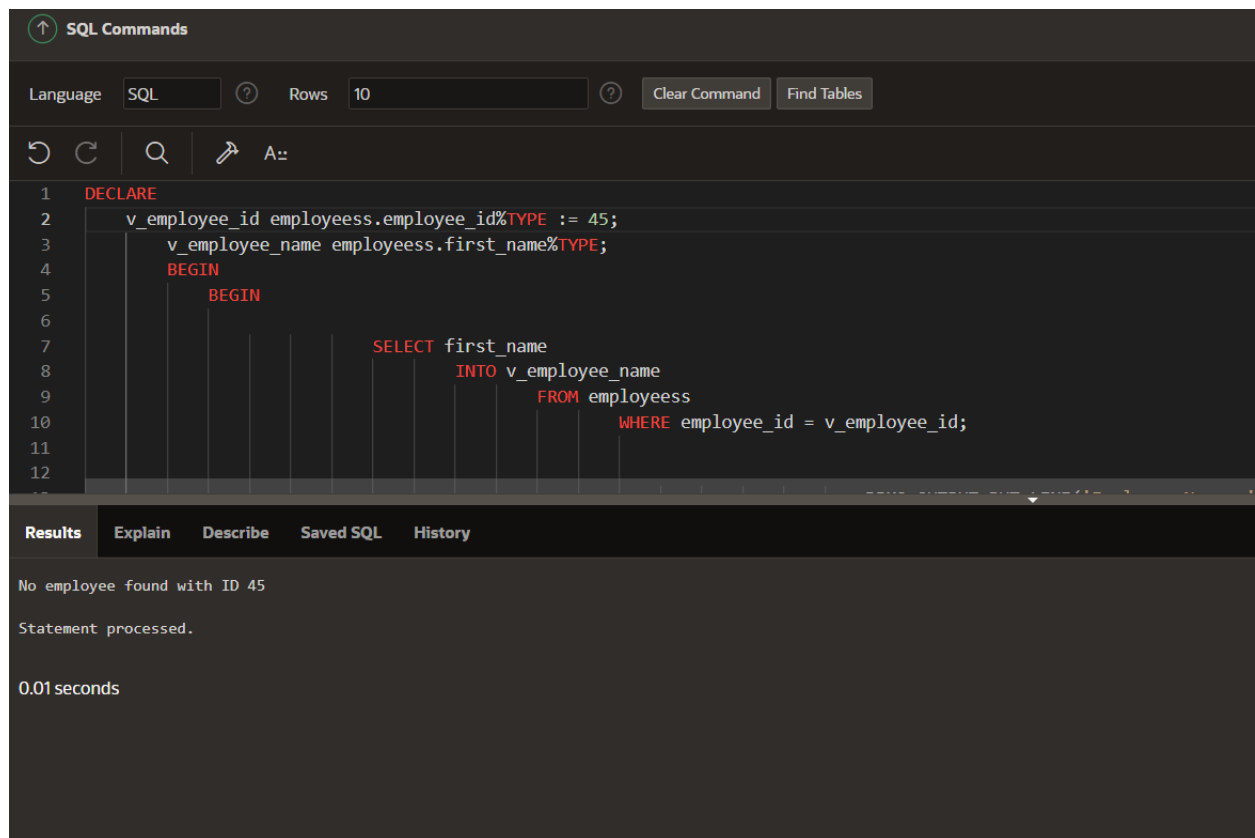
**Results** Explain Describe Saved SQL History

No employee found with ID 23

Statement processed.

0.01 seconds

Handle the `TOO_MANY_ROWS` exception when retrieving multiple rows instead of a single row from a table.



The screenshot shows an SQL IDE interface. At the top, there's a header 'SQL Commands' with an upward arrow icon. Below it, a toolbar contains 'Language' (set to 'SQL'), 'Rows' (set to '10'), 'Clear Command', and 'Find Tables'. A secondary toolbar has icons for undo, redo, search, and a format icon labeled 'A++'. The main editor area contains a PL/SQL block with line numbers 1 through 12. The code declares two variables, assigns a value to one, and performs a SELECT query. Below the editor, a tabbed interface shows 'Results' selected, displaying the output of the query: 'No employee found with ID 45', 'Statement processed.', and '0.01seconds'.

```
1 DECLARE
2   v_employee_id employees.employee_id%TYPE := 45;
3   v_employee_name employees.first_name%TYPE;
4 BEGIN
5   BEGIN
6
7     SELECT first_name
8     INTO v_employee_name
9     FROM employees
10    WHERE employee_id = v_employee_id;
11
12
```

**Results**   Explain   Describe   Saved SQL   History

No employee found with ID 45

Statement processed.

0.01seconds