



Endterm from DBMS-2

CARS

210103006 Toregul Akzhali

210103206 Baibolat Bekarys

210103264 Tilabek Yerdaulet

210103251 Temirkanat Zhandarbek

210103119 Mussafar Mustafa



Presentation structure:



STEP 1

Introduction.

Explain our project, introduce our project



STEP 2

ER diagram

Explain ER diagram, how they are connected to each other.



STEP 3

Normal Forms

Explanation of why the structure follows normal forms



STEP 4

Coding

Explanation and coding part of each item



introduction

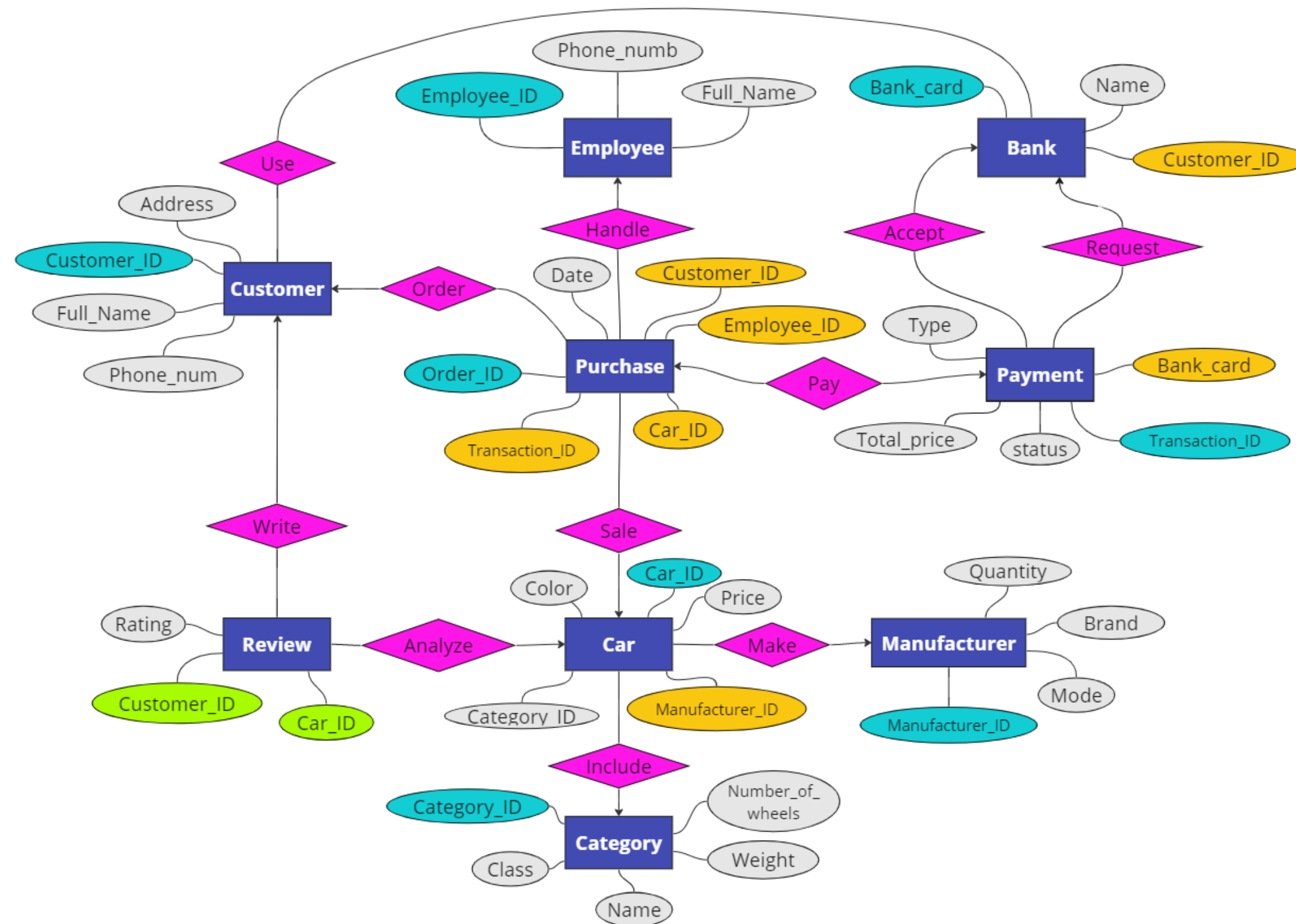
The transport industry is developing at the highest level in the world. And now we have developed a project according to which, with the development of online commerce, you can get transport quickly, easily, and conveniently. In addition, each company in the car interior, manufacturers produce different cars. In this project, customers can get the car they need in the desired category, from the right manufacturer, through a bank or without it and write a review and see a rating of car.

In the project created 9 entity ER diagrams, compiled 1, 2, 3 Normal form tables, recorded 1 trigger, 2 procedures, 1 function, 1 exception by coding



ERD

Blue is Primary key
Orange is Foreign key
Green is Primary and Foreign key
Indigo is Entity
Pink is Relation



Link to ERd (miro)

https://miro.com/welcomeonboard/WjVydK8yQ0xFOtg2T3NwN0dEN0Q4eIVCVVp0THdPRDE1RDhXdmNCRVFKcnpXNUIBd3ZsYVZrQXp0aFlYaXJ1dnwzNDU4NzY0NTQxMTg2MzY4Njk1fDI=?share_link_id=536641103717



Tables Columns

Car (Car_ID, Color, Price, Category_ID, Manufacturer_ID)

Review (Car_ID, Customer_ID, Rating)

Purchase (Order_ID, Customer_ID, Employee_ID, Transaction_ID, Car_ID, Date)

Payment (Transaction_ID, Card_number, Total_price, Type, Status)

Employee (Employee_ID, Phone_number, Full_name)

Customer (Customer_ID, Phone_number, Full_name, Address)

Category (Category_ID, Class, Weight, Number_of_wheels, Name)

Manufacturer (Manufacturer_ID, Model, Quantity, Brand)

Bank (Bank_card, Name, Customer_ID)



Tables Relations

- 1) Relation(handle) between Purchase and Employee (many-to-one)**
- 2) Relation(pay) between Purchase and Payment (one-to-one)**
- 3) Relation(accept,request) between Payment and Bank (many-to-one)**
- 4) Relation(sale) between Purchase and Car (many-to-one)**
- 5) Relation(make) between Car and Manufacturer (many-to-one)**
- 6) Relation(include) between Car and Category (many-to-one)**
- 7) Relation(order) between Purchase and Customer (many-to-one)**
- 8) Relation(write) between Customer and Review (one-to-many)**
- 9) Relation(analyze) between Review and Car (many-to-one)**
- 10) Relation(use) between Bank and Customer (many-to-many)**



Normal Form

Explanation of why the structure follows normal forms

1NF

1NF IS A PROPERTY OF A RELATION IN A RELATIONAL DATABASE . A RELATION IS IN THE FIRST NORMAL FORM IF AND ONLY IF NO ATTRIBUTE DOMAIN HAS RELATIONS AS ELEMENTS. OR, MORE INFORMALLY, NO TABLE COLUMN CAN HAVE TABLES AS VALUES

2NF

FOR A TABLE TO BE IN 2NF, THERE ARE TWO REQUIREMENTS

- THE DATABASE IS IN FIRST NORMAL FORM
- ALL NONKEY ATTRIBUTES IN THE TABLE MUST BE FUNCTIONALLY DEPENDENT ON THE ENTIRE PRIMARY KEY

3NF

A TABLE SATISFIES 3NF, IF AND ONLY IF FOR ANY NON-TRIVIAL $X \rightarrow Y$

- EITHER X IS A SUPERKEY
- OR EACH ATTRIBUTE IN Y IS CONTAINED IN A KEY

Our tables are in 1, 2, 3NF. Because there is one information in each row. And each non-key attribute is functionally dependent on the key. And the non-key attribute is non-transitively dependent on the primary key

1)CAR

{Car_id} -> {color, price, category_id, manufacturer_id}

CAR_ID	COLOR	PRICE	CATEGORY_ID	MANUFACTURER_ID
1	White	500000	2	2
2	Black	650000	1	1
3	Gray	500000	3	3
4	Dark Blue	1500000	5	5

2)CATEGORY

{Category_id} -> {class, weight, number_of_wheels, name}

CATEGORY_ID	CLASS	WEIGHT	NUMBER_OF_WHEELS	NAME
1	A	400	4	Sedan
2	B	600	4	Minivan
3	D	400	4	Bus
4	E	1000	8	Truck

3)CUSTOMER

{Customer_id} -> {phone_number, full_name, address}

CUSTOMER_ID	PHONE_NUMBER	FULL_NAME	ADDRESS
1	87004554545	Onaibaev Dias	Tole bi 12
2	87024945854	Karasayev Dias	Vernye 21
3	87013744957	Tagybai Aset	Samuryk 74
4	87022945930	Kozy-Korpesh Bekzat	Satbaev 22

4)EMPLOYEE

{Employee_id} -> {phone_number, full_name}

EMPLOYEE_ID	PHONE_NUMBER	FULL_NAME
1	87021495824	Temirkanat Zhandarbek
2	87006647896	Toregul Akzhali
3	87078737458	Baibolat Bekarys
4	87026005121	Tilebek Erdaulet

5)MANUFACTURER

{Manufacturer_id} -> {model, quantity, brand}

MANUFACTURER_ID	MODEL	QUANTITY	BRAND
1	Prada-200	130	Toyota
2	Prada-300	145	Toyota
3	E221	150	Mercedes
4	M5	125	BMW

6)PURCHASE

{Oreder_id} -> {customer_id,employee_id, transaction_id, car_id, date}

OREDER_ID	CUSTOMER_ID	EMPLOYEE_ID	TRANSACTION_ID	CAR_ID	DATE
1	2	2	402	1	04/20/2023
2	1	1	401	2	07/07/2022
3	3	3	403	3	11/21/2021
4	4	4	404	4	02/01/2019

7)Review

{Customer_id, Car_id} -> {rating}

CAR_ID	CUSTOMER_ID	RATING
1	1	50
2	2	90
3	3	100
4	4	70

8)PAYMENT

{Transaction_id} -> {bank_card, total_price, type, status}

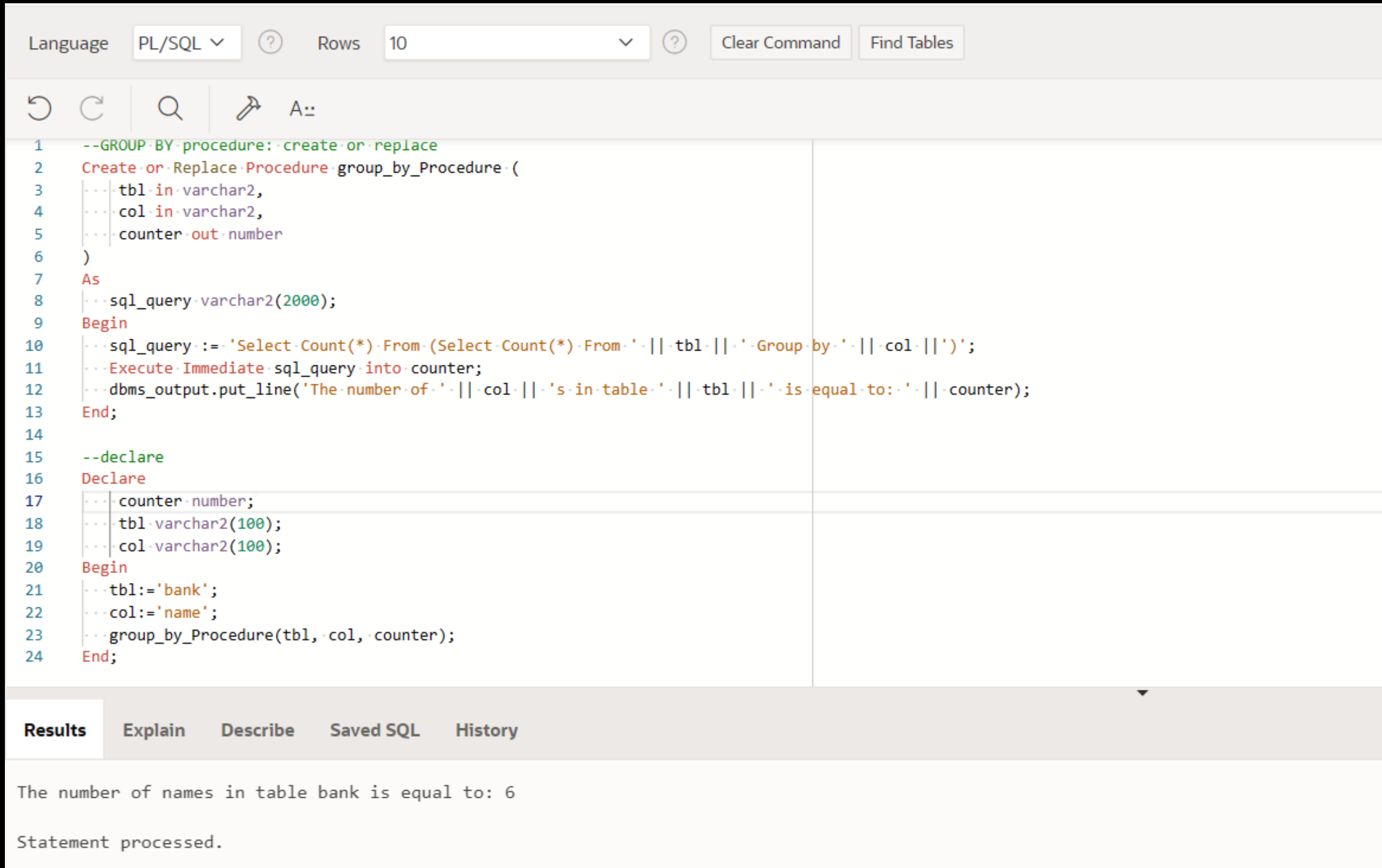
TRANSACTION_ID	BANK_CARD	TOTAL_PRICE	TYPE	STATUS
401	50	1000000	bank_card	accept
402		1200000	cash	
403	52	1500000	bank_card	declained
404	53	900000	bank_card	acceprt

9)BANK

{Bank_card} -> {name}

BANK_CARD	NAME	CUSTOMER_ID
50	halyk	1
51	kaspi	4
52	otbasy	11
53	red	3

--Procedure which does group by information



The screenshot displays the Oracle SQL Developer interface. At the top, the 'Language' is set to 'PL/SQL', 'Rows' are set to 10, and there are buttons for 'Clear Command' and 'Find Tables'. Below the toolbar, a PL/SQL procedure named 'group_by_Procedure' is defined. The procedure takes three parameters: 'tbl' (varchar2), 'col' (varchar2), and 'counter' (out number). It declares a local variable 'sql_query' of type 'varchar2(2000)'. The procedure's logic involves constructing a SQL query to count the number of rows in the specified table grouped by the specified column, executing this query, and then outputting the result. Below the procedure definition, there is a second block of code that declares local variables for 'tbl', 'col', and 'counter', sets 'tbl' to 'bank' and 'col' to 'name', and calls the 'group_by_Procedure' with these values. The 'Results' tab at the bottom shows the output of the procedure: 'The number of names in table bank is equal to: 6' and 'Statement processed.'

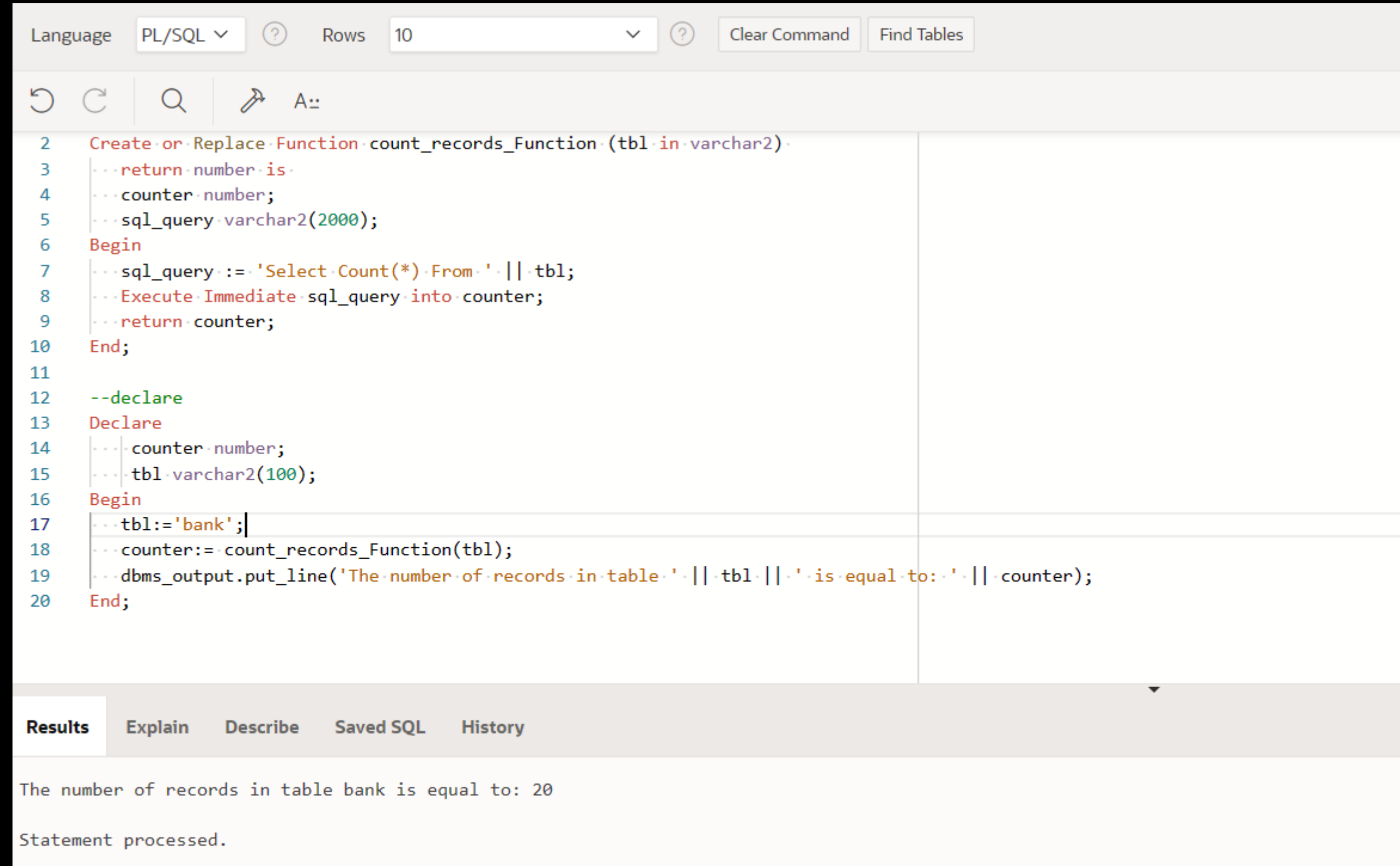
```
1  --GROUP BY procedure: create or replace
2  Create or Replace Procedure group_by_Procedure (
3      tbl in varchar2,
4      col in varchar2,
5      counter out number
6  )
7  As
8      sql_query varchar2(2000);
9  Begin
10     sql_query := 'Select Count(*) From (Select Count(*) From ' || tbl || ' Group by ' || col || ')';
11     Execute Immediate sql_query into counter;
12     dbms_output.put_line('The number of ' || col || 's in table ' || tbl || ' is equal to: ' || counter);
13 End;
14
15 --declare
16 Declare
17     counter number;
18     tbl varchar2(100);
19     col varchar2(100);
20 Begin
21     tbl := 'bank';
22     col := 'name';
23     group_by_Procedure(tbl, col, counter);
24 End;
```

Results Explain Describe Saved SQL History

The number of names in table bank is equal to: 6

Statement processed.

--Function which counts the number of records



The screenshot shows a SQL IDE interface. At the top, there's a toolbar with 'Language' set to 'PL/SQL', 'Rows' set to '10', and buttons for 'Clear Command' and 'Find Tables'. Below the toolbar is a command area with a search icon and a text input field containing 'A:'. The main editor area displays two PL/SQL blocks. The first block is a function definition for 'count_records_Function' that takes a table name as input and returns the number of records. The second block is an anonymous PL/SQL block that declares a variable 'tbl' and calls the function to count records in the 'bank' table, outputting the result. The 'Results' tab at the bottom shows the output of the execution: 'The number of records in table bank is equal to: 20' and 'Statement processed.'

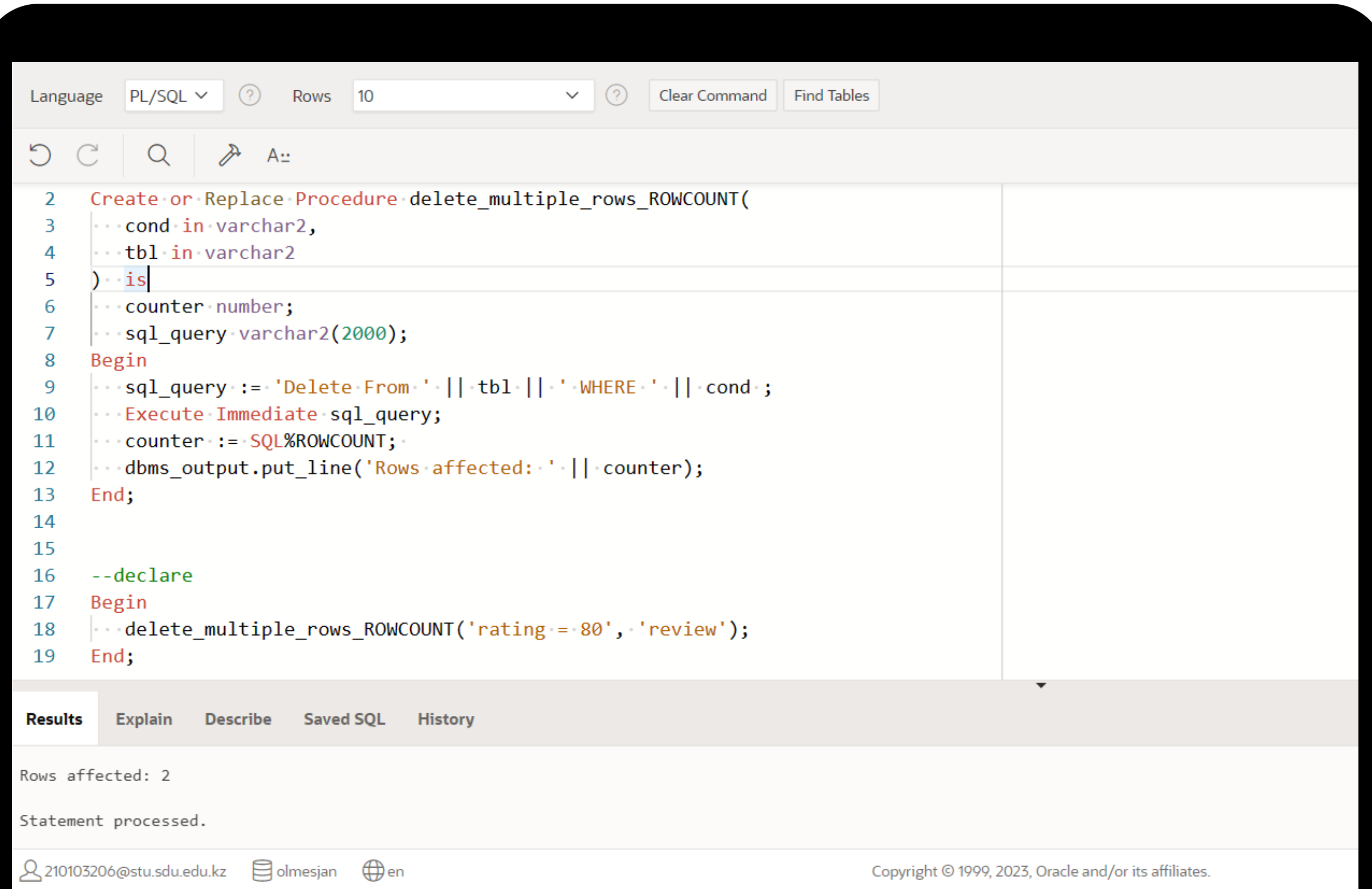
```
2 Create or Replace Function count_records_Function (tbl in varchar2)
3   ..return number is
4   ..counter number;
5   ..sql_query varchar2(2000);
6 Begin
7   ..sql_query := 'Select Count(*) From ' || tbl;
8   ..Execute Immediate sql_query into counter;
9   ..return counter;
10 End;
11
12 --declare
13 Declare
14   ..counter number;
15   ..tbl varchar2(100);
16 Begin
17   ..tbl := 'bank';
18   ..counter := count_records_Function(tbl);
19   ..dbms_output.put_line('The number of records in table ' || tbl || ' is equal to: ' || counter);
20 End;
```

Results Explain Describe Saved SQL History

The number of records in table bank is equal to: 20

Statement processed.

--Procedure which uses SQL%ROWCOUNT to determine the number of rows affected



The screenshot displays the Oracle SQL Developer environment. At the top, the 'Language' is set to 'PL/SQL' and 'Rows' is set to '10'. The main editor shows a PL/SQL procedure named 'delete_multiple_rows_ROWCOUNT' with the following code:

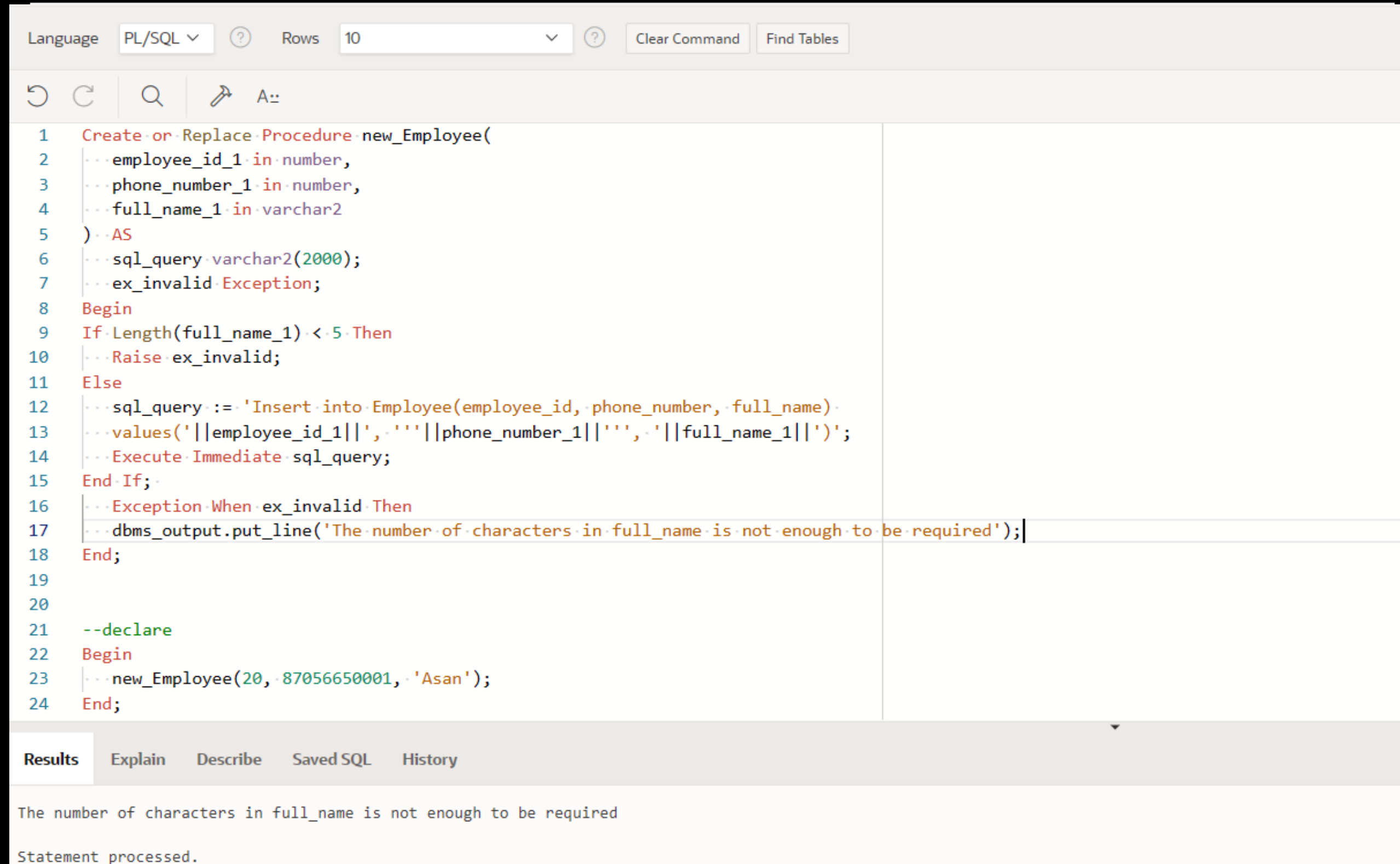
```
2 Create or Replace Procedure delete_multiple_rows_ROWCOUNT(  
3   ...cond in varchar2,  
4   ...tbl in varchar2  
5 ) is  
6   ...counter number;  
7   ...sql_query varchar2(2000);  
8 Begin  
9   ...sql_query := 'Delete From ' || tbl || ' WHERE ' || cond;  
10  ...Execute Immediate sql_query;  
11  ...counter := SQL%ROWCOUNT;  
12  ...dbms_output.put_line('Rows affected: ' || counter);  
13 End;  
14  
15  
16 --declare  
17 Begin  
18   ...delete_multiple_rows_ROWCOUNT('rating=80','review');  
19 End;
```

Below the editor, the 'Results' tab is active, showing the output of the procedure:

```
Rows affected: 2  
  
Statement processed.
```

The footer of the interface includes the user '210103206@stu.sdu.edu.kz', the database 'olmesjan', and the language 'en'. A copyright notice at the bottom right states: 'Copyright © 1999, 2023, Oracle and/or its affiliates.'

--Add user-defined exception which disallows to enter title of item (e.g. book) to be less than 5 characters



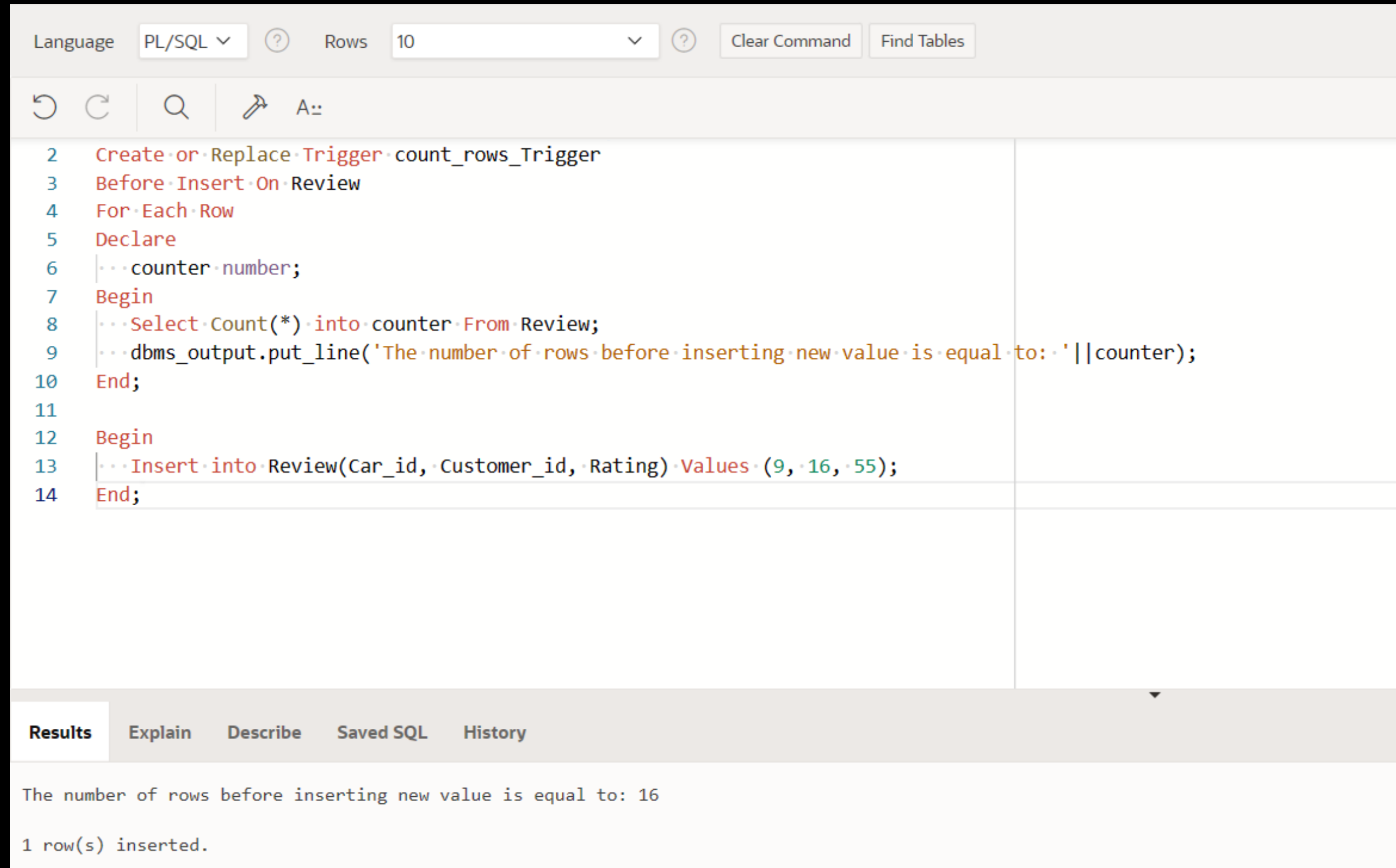
The screenshot shows the Oracle SQL Developer interface. At the top, there's a toolbar with 'Language' set to 'PL/SQL', 'Rows' set to '10', and buttons for 'Clear Command' and 'Find Tables'. Below the toolbar is a command area with a search icon and a text input field containing 'A:'. The main area displays a PL/SQL procedure named 'new_Employee' with the following code:

```
1 Create or Replace Procedure new_Employee(  
2   ...employee_id_1 in number,  
3   ...phone_number_1 in number,  
4   ...full_name_1 in varchar2  
5 ) AS  
6   ...sql_query varchar2(2000);  
7   ...ex_invalid Exception;  
8 Begin  
9   If Length(full_name_1) < 5 Then  
10  ...Raise ex_invalid;  
11 Else  
12  ...sql_query := 'Insert into Employee(employee_id, phone_number, full_name)  
13  ...values('||employee_id_1||', '||phone_number_1||', '||full_name_1||)';  
14  ...Execute Immediate sql_query;  
15 End If;  
16  ...Exception When ex_invalid Then  
17  ...dbms_output.put_line('The number of characters in full_name is not enough to be required');  
18 End;  
19  
20  
21 --declare  
22 Begin  
23  ...new_Employee(20, 87056650001, 'Asan');  
24 End;
```

At the bottom, there's a 'Results' tab with sub-tabs for 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is active, showing the output of the procedure execution:

```
The number of characters in full_name is not enough to be required  
  
Statement processed.
```

--Create a trigger before insert on any entity which will show the current number of rows in the table



The screenshot shows a SQL IDE interface. At the top, there's a toolbar with 'Language' set to 'PL/SQL', a 'Rows' dropdown set to '10', and buttons for 'Clear Command' and 'Find Tables'. Below the toolbar is a command area with a search icon and a text input field containing 'A:'. The main editor area displays the following PL/SQL code:

```
2 Create or Replace Trigger count_rows_Trigger
3 Before Insert On Review
4 For Each Row
5 Declare
6   counter number;
7 Begin
8   select Count(*) into counter From Review;
9   dbms_output.put_line('The number of rows before inserting new value is equal to: '||counter);
10 End;
11
12 Begin
13   Insert into Review(Car_id, Customer_id, Rating) Values (9, 16, 55);
14 End;
```

At the bottom, there's a 'Results' tab selected, showing the output of the execution:

```
The number of rows before inserting new value is equal to: 16

1 row(s) inserted.
```


Conclusion

The transportation management system is an innovative project that offers fast and reliable transportation services. It includes nine entities with unique attributes and is designed using Normal Forms tables to ensure efficient data management. The project uses automated triggers, procedures, functions, and exceptions to provide a streamlined approach to transportation management, setting new standards for the industry.

**THANKS
FOR
YOUR
ATTENTION**