

Database Programming with SQL

* 1. : Ensuring Quality Query Results– Advanced Technique Practice Activities

# Objectives

* + - Create a query to produce specific data
    - Modify a query to produce specified data

# Try It / Solve It

1. Create additional tables used in this section by executing the following statements: CREATE TABLE emp AS SELECT \* FROM employees;

CREATE TABLE dept AS SELECT \* FROM departments;

1. Produce a report that lists the constraint name, type, column name, and column position of all the constraints on the JOB\_HISTORY table, apart from the not null constraints.

CREATE TABLE my\_temp\_table AS

(SELECT cons.constraint\_name, cons.constraint\_type,

cols.column\_name, cols.position,

TO\_LOB(cons.search\_condition) search\_condition

FROM user\_constraints cons

INNER JOIN user\_cons\_columns cols

ON cons.constraint\_name = cols.constraint\_name

WHERE cons.table\_name = 'JOB\_HISTORY' );

SELECT \* FROM my\_temp\_table

WHERE NOT (

constraint\_type = 'C'

AND column\_name in

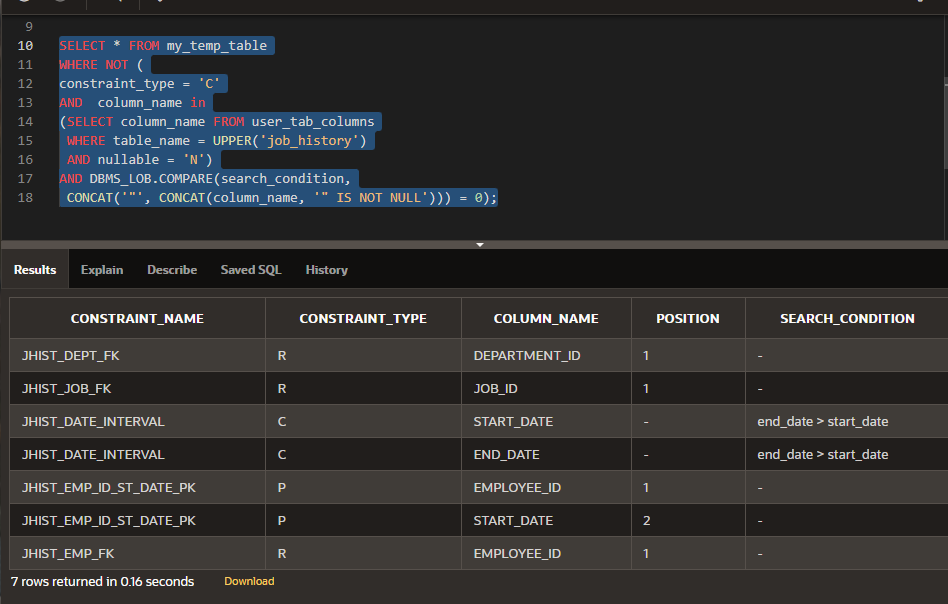
(SELECT column\_name FROM user\_tab\_columns

WHERE table\_name = UPPER('job\_history')

AND nullable = 'N')

AND DBMS\_LOB.COMPARE(search\_condition,

CONCAT('"', CONCAT(column\_name, '" IS NOT NULL'))) = 0);



1. Create a primary key constraint on the emp table’s employee\_id column

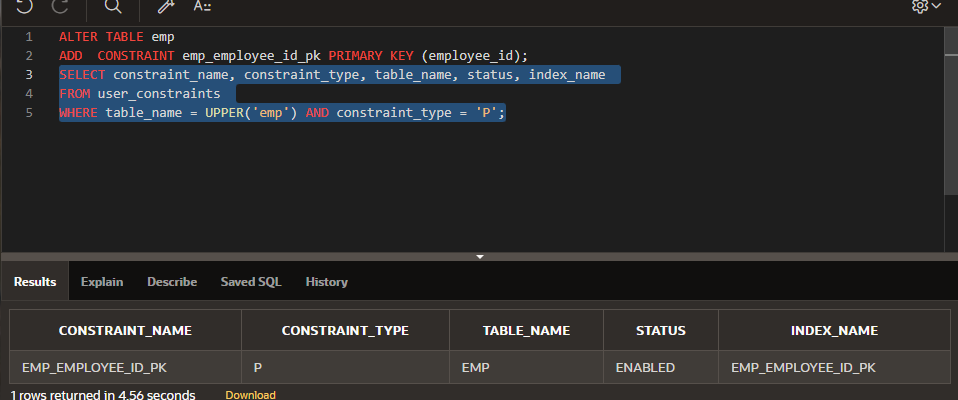
ALTER TABLE emp

ADD CONSTRAINT emp\_employee\_id\_pk PRIMARY KEY (employee\_id);

SELECT constraint\_name, constraint\_type, table\_name, status, index\_name

FROM user\_constraints

WHERE table\_name = UPPER('emp') AND constraint\_type = 'P';



1. Create a primary key on the dept table’s department\_id column

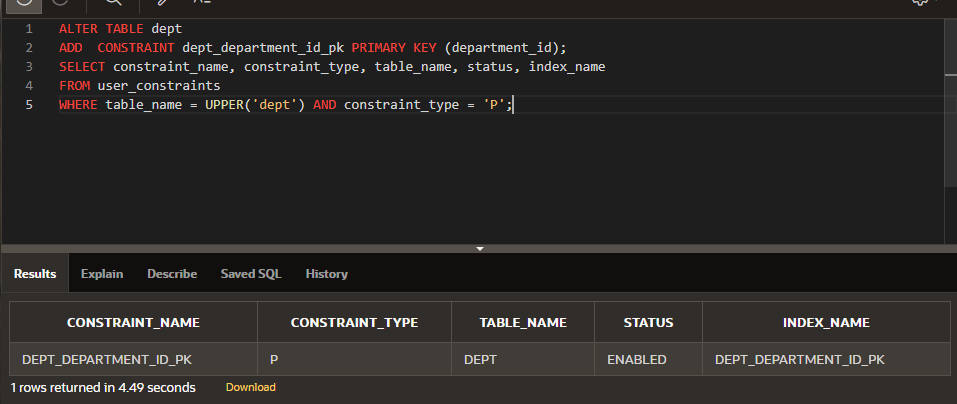
ALTER TABLE dept

ADD CONSTRAINT dept\_department\_id\_pk PRIMARY KEY (department\_id);

SELECT constraint\_name, constraint\_type, table\_name, status, index\_name

FROM user\_constraints

WHERE table\_name = UPPER('dept') AND constraint\_type = 'P';



1. Add a foreign constraint between DEPT and EMP so that only valid departments can be entered in the EMP table. Make sure you can delete any row from the DEPT table, and that referenced rows in the EMP table are deleted.

ALTER TABLE emp ADD CONSTRAINT emp\_dept\_department\_id\_fk FOREIGN KEY (department\_id)

REFERENCES dept (department\_id) ON DELETE CASCADE;

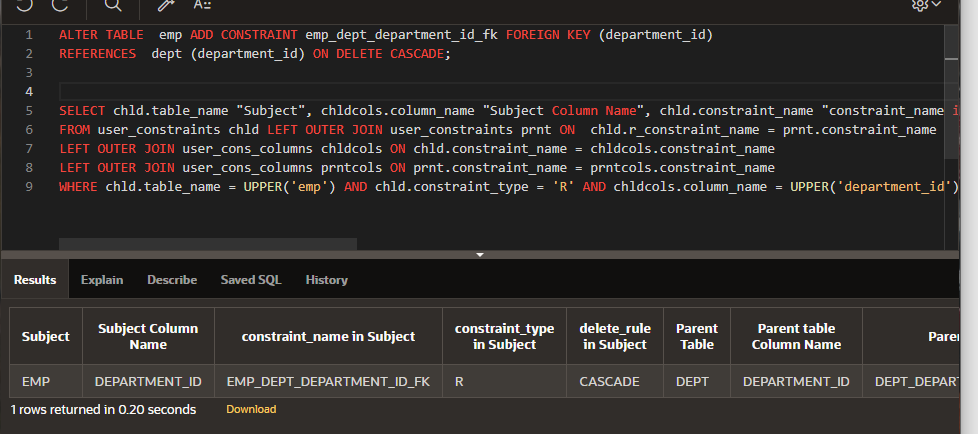
SELECT chld.table\_name "Subject", chldcols.column\_name "Subject Column Name", chld.constraint\_name "constraint\_name in Subject", chld.constraint\_type "constraint\_type in Subject", chld.delete\_rule "delete\_rule in Subject", prnt.table\_name "Parent Table", prntcols.column\_name "Parent table Column Name", prnt.constraint\_name "Parent PK"

FROM user\_constraints chld LEFT OUTER JOIN user\_constraints prnt ON chld.r\_constraint\_name = prnt.constraint\_name

LEFT OUTER JOIN user\_cons\_columns chldcols ON chld.constraint\_name = chldcols.constraint\_name

LEFT OUTER JOIN user\_cons\_columns prntcols ON prnt.constraint\_name = prntcols.constraint\_name

WHERE chld.table\_name = UPPER('emp') AND chld.constraint\_type = 'R' AND chldcols.column\_name = UPPER('department\_id') ;



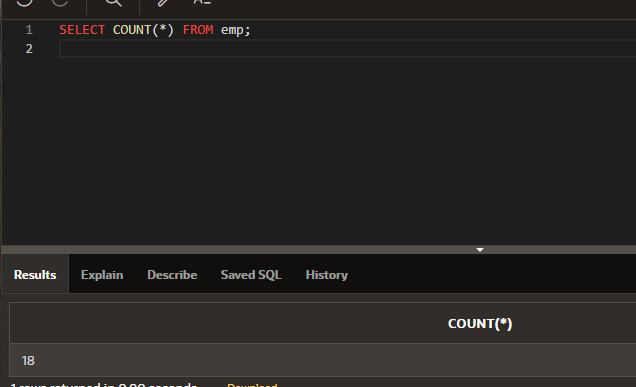
1. Test the foreign key constraint you just created:

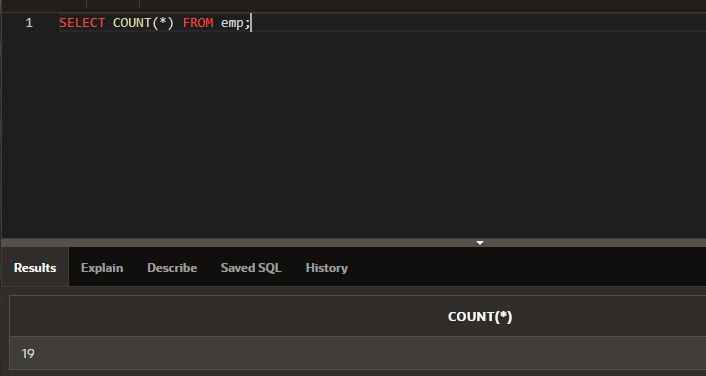
Count the number of rows in the EMP table. Remove department 10 from the dept table.

Now count emps again. There should be fewer employees.

SELECT COUNT(\*) FROM emp;

DELETE FROM dept WHERE department\_id = 10;





1. Produce a report that returns the last name, salary, department number, and average salary of all the departments where salary is greater than the average salary.

WITH avg\_sal\_by\_dept AS

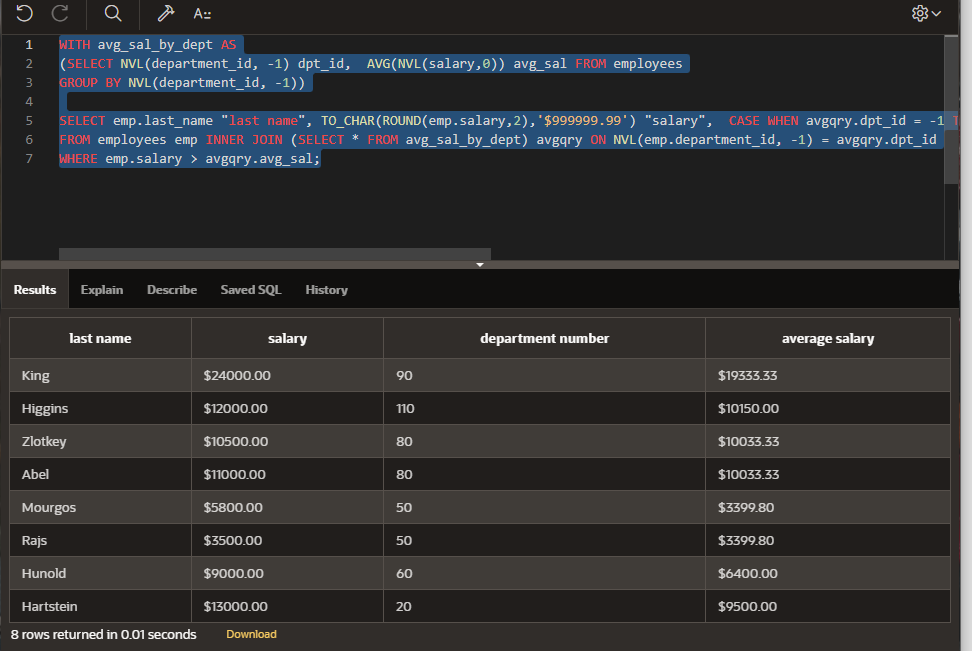
(SELECT NVL(department\_id, -1) dpt\_id, AVG(NVL(salary,0)) avg\_sal FROM employees

GROUP BY NVL(department\_id, -1))

SELECT emp.last\_name "last name", TO\_CHAR(ROUND(emp.salary,2),'$999999.99') "salary", CASE WHEN avgqry.dpt\_id = -1 THEN NULL ELSE avgqry.dpt\_id END "department number", TO\_CHAR(ROUND(avgqry.avg\_sal,2),'$999999.99') "average salary"

FROM employees emp INNER JOIN (SELECT \* FROM avg\_sal\_by\_dept) avgqry ON NVL(emp.department\_id, -1) = avgqry.dpt\_id

WHERE emp.salary > avgqry.avg\_sal;



1. Create a view named V2 that returns the highest salary, lowest salary, average salary and department name.

CREATE OR REPLACE VIEW v2 ("highest salary", "lowest salary", "average salary", "Department Name") AS

SELECT

TO\_CHAR(ROUND(MAX(NVL(emp.salary,0)),2),'$999999.99'),

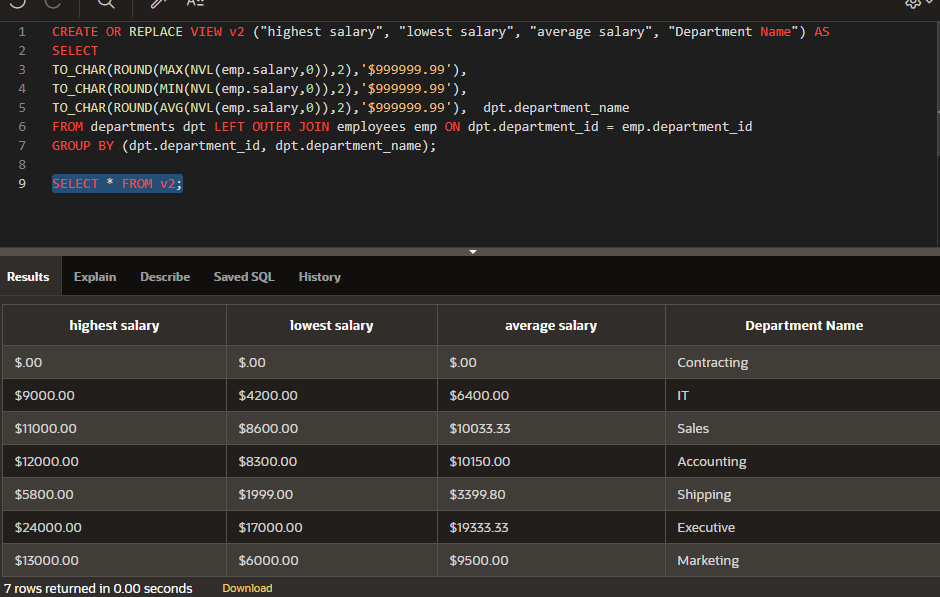
TO\_CHAR(ROUND(MIN(NVL(emp.salary,0)),2),'$999999.99'),

TO\_CHAR(ROUND(AVG(NVL(emp.salary,0)),2),'$999999.99'), dpt.department\_name

FROM departments dpt LEFT OUTER JOIN employees emp ON dpt.department\_id = emp.department\_id

GROUP BY (dpt.department\_id, dpt.department\_name);

SELECT \* FROM v2;



1. Create a view named Dept\_Managers\_view that returns a listing of department names long with the manager initial and surname for that department. Test the view by returning all the rows from it. Make sure no rows can be updated through the view. Try to run an UPDATE statement against the view.

CREATE OR REPLACE VIEW dept\_managers\_view AS

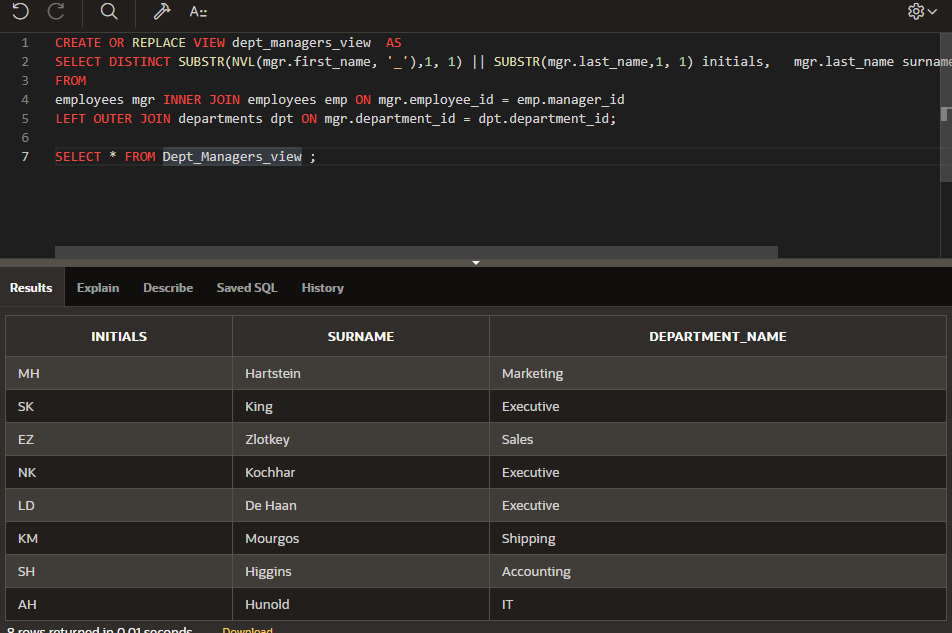
SELECT DISTINCT SUBSTR(NVL(mgr.first\_name, '\_'),1, 1) || SUBSTR(mgr.last\_name,1, 1) initials, mgr.last\_name surname, dpt.department\_name

FROM

employees mgr INNER JOIN employees emp ON mgr.employee\_id = emp.manager\_id

LEFT OUTER JOIN departments dpt ON mgr.department\_id = dpt.department\_id;

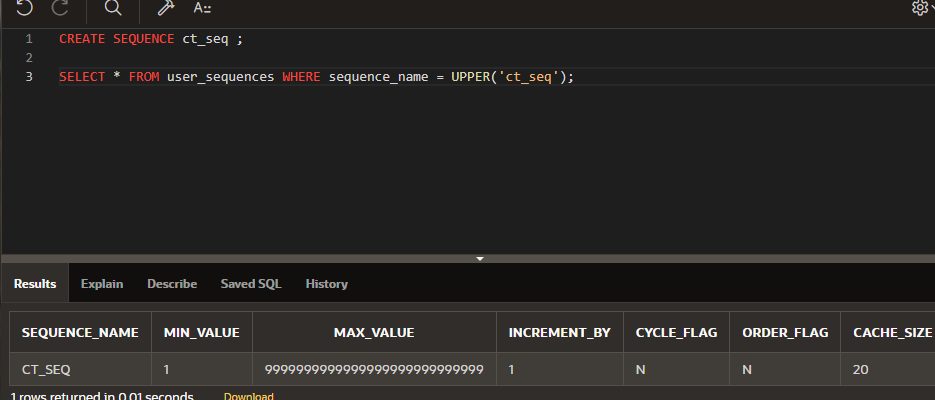
SELECT \* FROM Dept\_Managers\_view ;



1. Create a sequence named ct\_seq using all the default values.

CREATE SEQUENCE ct\_seq ;

SELECT \* FROM user\_sequences WHERE sequence\_name = UPPER('ct\_seq');



1. Examine the following insert statement and fix the errors.

INSERT INTO emp

(employee\_id, first\_name, last\_name, email, phone\_number, hire\_date, job\_id, salary, commission\_pct, manager\_id, department\_id)

VALUES

(ct\_seq.nextvalue, "Kaare", 'Hansen', 'KHANSEN', '44965 832123',

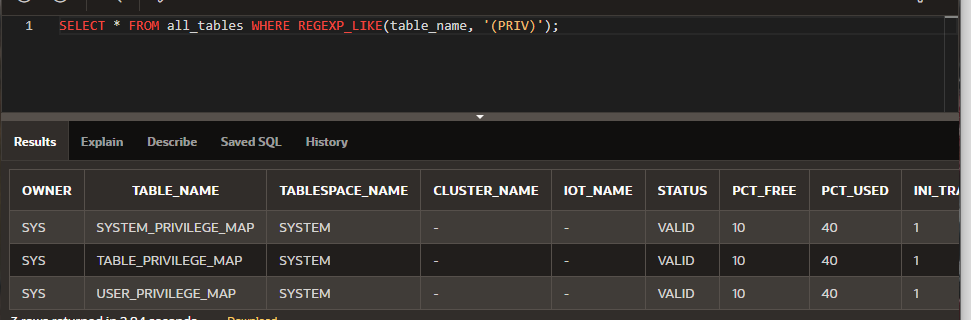
sysdate, 'SA\_REP', $6500, null, 100, 20);

(ct\_seq.NEXTVAL, 'Kaare', 'Hansen', 'KHANSEN', '44965 832123',

sysdate, 'SA\_REP', 6500, null, 100, 20);

1. Write the SQL statement to list all the user tables which contains the name PRIV.

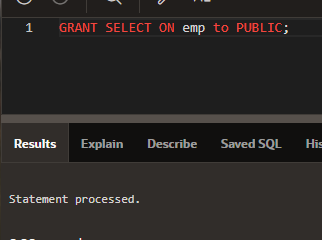
SELECT \* FROM all\_tables WHERE REGEXP\_LIKE(table\_name, '(PRIV)');



1. Give select access to public on the EMP table, and verify the grant by running this query.

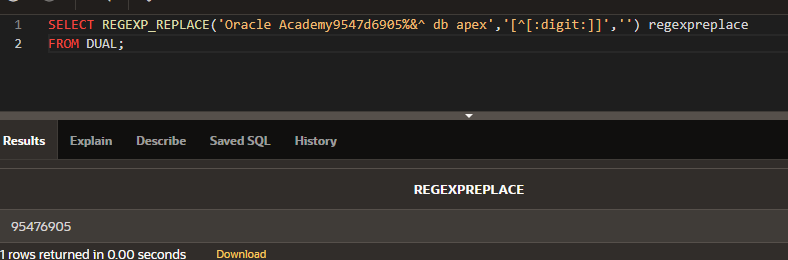
SELECT \*

FROM user\_tab\_privs WHERE table\_name = 'EMP';



1. Replace the ?? in the following query using regular expressions to return only the numbers from the following string: 'Oracle Academy9547d6905%&^ db apex'.

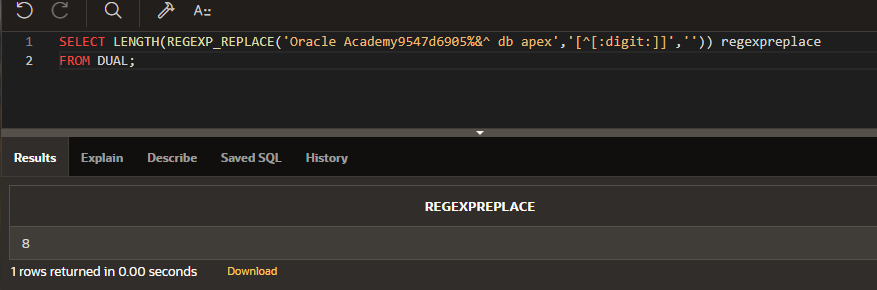
SELECT REGEXP\_REPLACE('Oracle Academy9547d6905%&^ db apex',??,'') regexpreplace FROM DUAL;



1. Amend the previous query using regular expressions to return the number of digits from the following string: 'Oracle Academy9547d6905%&^ db’

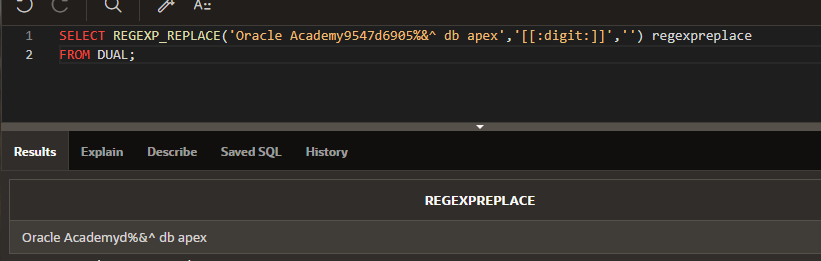
SELECT LENGTH(REGEXP\_REPLACE('Oracle Academy9547d6905%&^ db apex','??',''))

regexpreplace FROM DUAL;

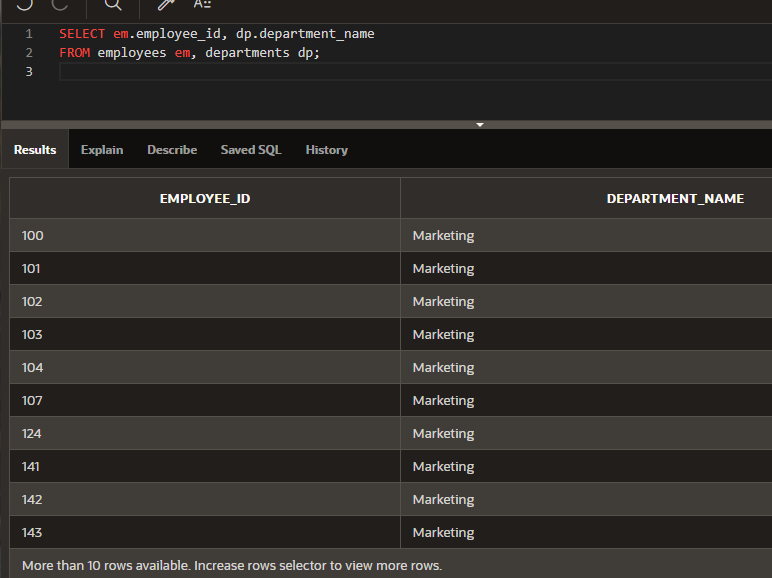


1. Amend the query again to return only the non-numeric characters.

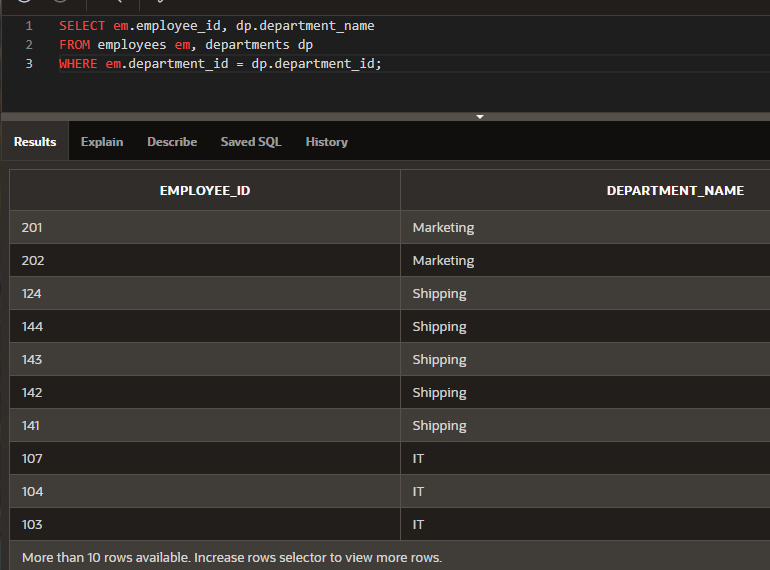
SELECT REGEXP\_REPLACE('Oracle Academy9547d6905%&^ db apex','??','') regexpreplace FROM DUAL;



1. Using Oracle proprietary joins, construct a statement that returns all the employee\_ids joined to all the department\_names.



1. Still using Oracle Joins, correct the previous statement so that it returns only the name of the department that the employee actually works in.



1. Still using Oracle Joins, construct a query that lists the employees last name, the department name, the salary, and the country name of all employees.

SELECT em.last\_name "last name", dp.department\_name "department name",em.salary, con.country\_name "country name"

FROM employees em, departments dp, locations loc, countries con

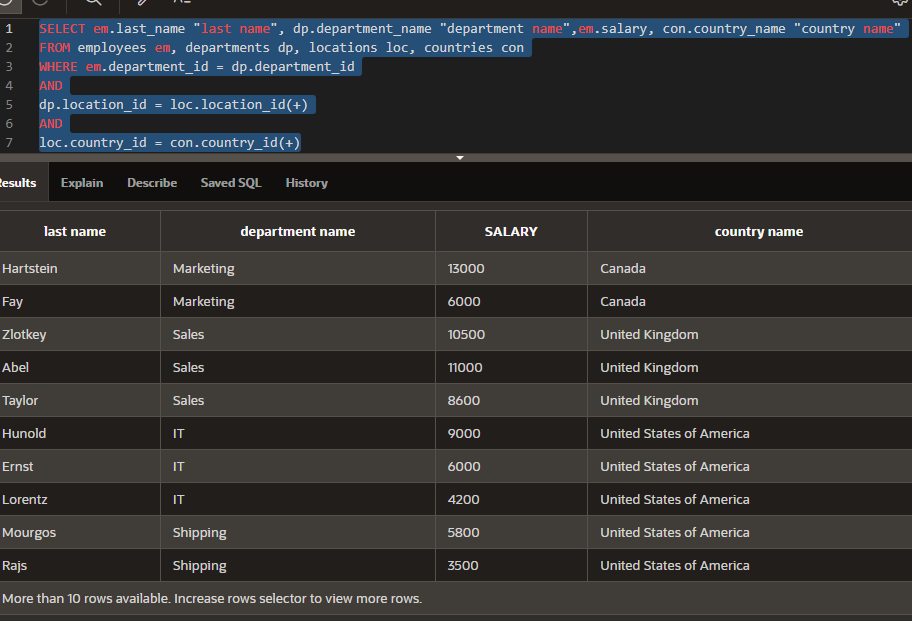
WHERE em.department\_id = dp.department\_id

AND

dp.location\_id = loc.location\_id(+)

AND

loc.country\_id = con.country\_id(+)



1. Still using Oracle join syntax, alter the previous query so that it also includes the employee record of the employee with no department\_id, ‘Grant’.

SELECT em.last\_name "last name", dp.department\_name "department name",em.salary, con.country\_name "country name"

FROM employees em, departments dp, locations loc, countries con

WHERE em.department\_id = dp.department\_id(+)

AND

dp.location\_id = loc.location\_id(+)

AND

loc.country\_id = con.country\_id(+);

