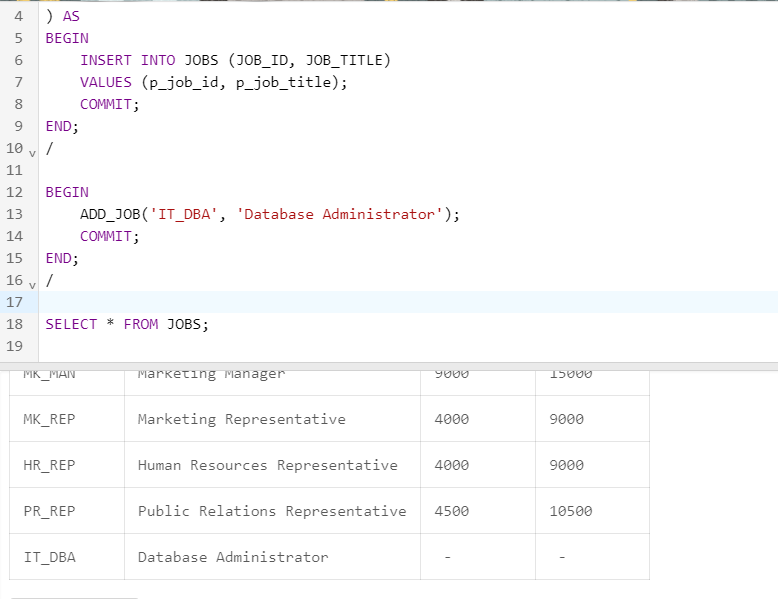
**Practice 6**

1. Create, compile, and invoke the ADD\_JOB procedure and review the results.

* + - 1. Create a procedure called ADD\_JOB to insert a new job into the JOBS table. Provide the ID and job title using two parameters.
      2. Compile the code, and then invoke the procedure with IT\_DBA as the job ID and Database Administrator as the job title. Query the JOBS table and view the results.



CREATE OR REPLACE PROCEDURE ADD\_JOB (

p\_job\_id JOBS.JOB\_ID%TYPE,

p\_job\_title JOBS.JOB\_TITLE%TYPE

) AS

BEGIN

INSERT INTO JOBS (JOB\_ID, JOB\_TITLE)

VALUES (p\_job\_id, p\_job\_title);

COMMIT;

END;

/

BEGIN

ADD\_JOB('IT\_DBA', 'Database Administrator');

COMMIT;

END;

/

SELECT \* FROM JOBS;

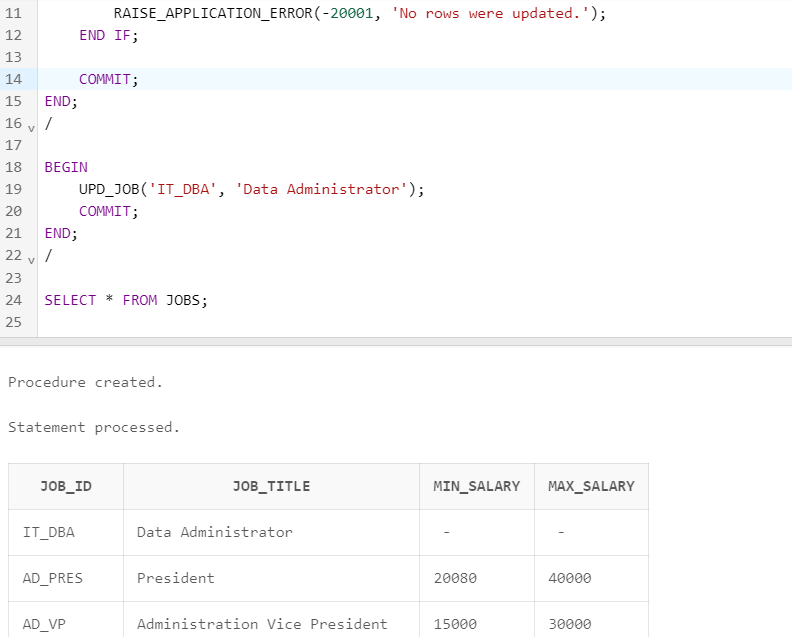
* + - 1. Invoke your procedure again, passing a job ID of ST\_MAN and a job title of Stock Manager. What happens and why?

ORA-00001: unique constraint

(SQL\_TTPAYIEHJWTLFJILYAFCOFWDI.JOB\_ID\_PK) violated

2. Create a procedure called UPD\_JOB to modify a job in the JOBS table.

* + - 1. Create a procedure called UPD\_JOB to update the job title. Provide the job ID and a new title using two parameters. Include the necessary exception handling if no update occurs.
      2. Compile the procedure. Invoke the procedure to change the job title of the job ID IT\_DBA to Data Administrator. Query the JOBS table and view the results.

a

CREATE OR REPLACE PROCEDURE UPD\_JOB (

p\_job\_id JOBS.JOB\_ID%TYPE,

p\_new\_job\_title JOBS.JOB\_TITLE%TYPE

) AS

BEGIN

UPDATE JOBS

SET JOB\_TITLE = p\_new\_job\_title

WHERE JOB\_ID = p\_job\_id;

IF SQL%ROWCOUNT = 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'No rows were updated.');

END IF;

COMMIT;

END;

/

BEGIN

UPD\_JOB('IT\_DBA', 'Data Administrator');

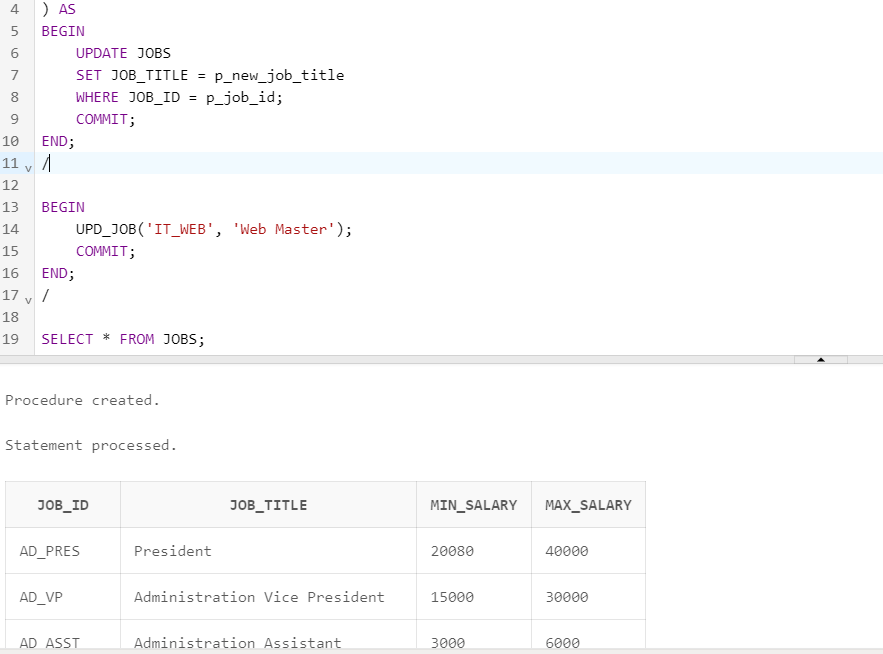
COMMIT;

END;

/

SELECT \* FROM JOBS;

* + - 1. Test the exception handling section of the procedure by trying to update a job that does not exist. You can use the job ID IT\_WEB and the job title Web Master.



CREATE OR REPLACE PROCEDURE UPD\_JOB (

p\_job\_id JOBS.JOB\_ID%TYPE,

p\_new\_job\_title JOBS.JOB\_TITLE%TYPE

) AS

BEGIN

UPDATE JOBS

SET JOB\_TITLE = p\_new\_job\_title

WHERE JOB\_ID = p\_job\_id;

COMMIT;

END;

/

BEGIN

UPD\_JOB('IT\_DBA', 'Data Administrator');

COMMIT;

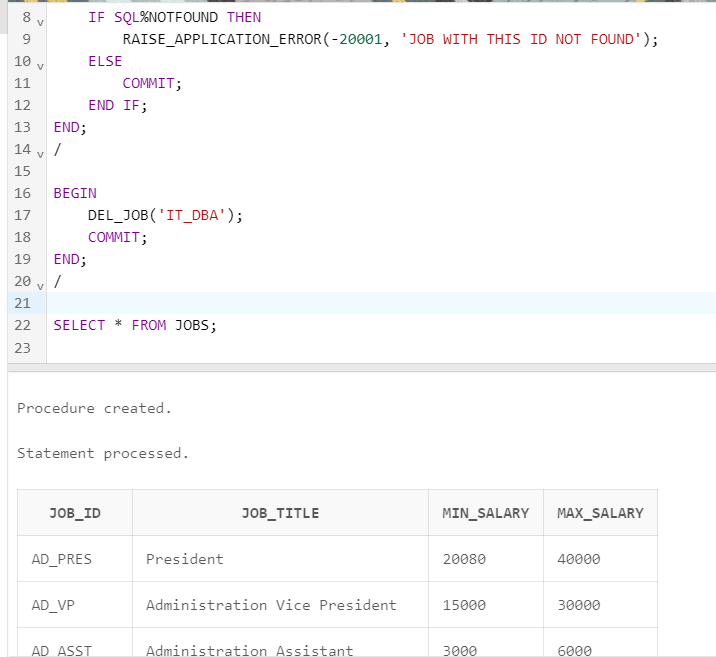
END;

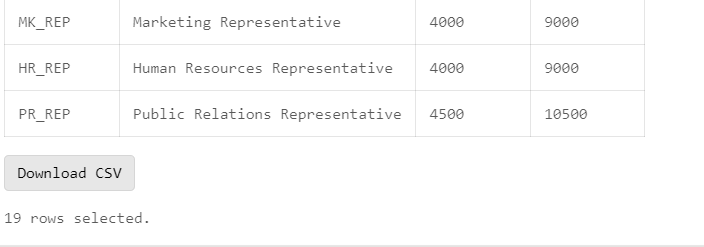
/

SELECT \* FROM JOBS;

3. Create a procedure called DEL\_JOB to delete a job from the JOBS table.

* + - 1. Create a procedure called DEL\_JOB to delete a job. Include the necessary exception handling code if no job is deleted.
      2. Compile the code; invoke the procedure using the job ID IT\_DBA. Query the JOBS table and view the results.





CREATE OR REPLACE PROCEDURE DEL\_JOB (

p\_job\_id JOBS.JOB\_ID%TYPE

) AS

BEGIN

DELETE FROM JOBS

WHERE JOB\_ID = p\_job\_id;

IF SQL%NOTFOUND THEN

RAISE\_APPLICATION\_ERROR(-20001, 'No job found with the specified job ID');

ELSE

COMMIT;

END IF;

END;

/

BEGIN

DEL\_JOB('IT\_DBA');

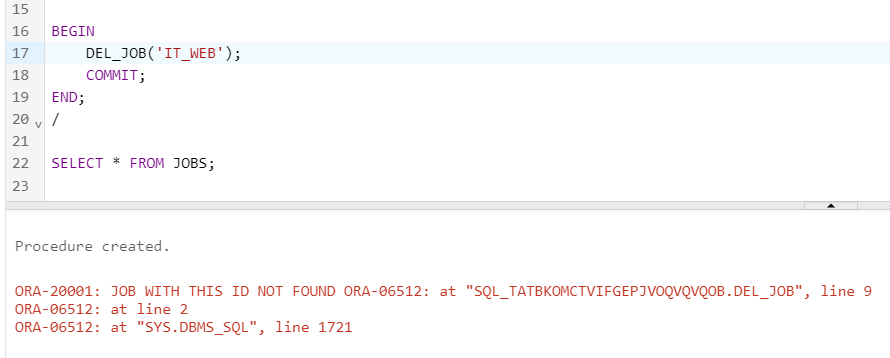
COMMIT;

END;

/

SELECT \* FROM JOBS;

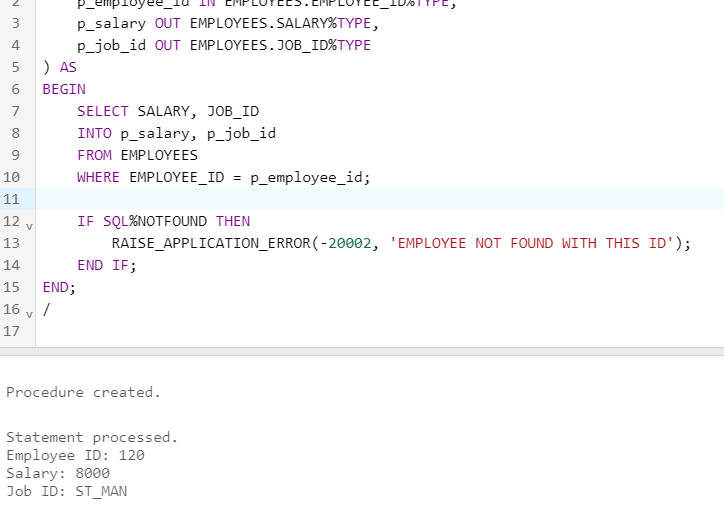
* + - 1. Test the exception handling section of the procedure by trying to delete a job that does not exist. Use the IT\_WEB as the job ID. You should get the message that you included in the exception handling section of the procedure as the output.



4. Create a procedure called GET\_EMPLOYEE to query the EMPLOYEES table, retrieving the salary and job ID for an employee when provided with the employee ID.

a. Create a procedure that returns a value from the SALARY and JOB\_ID columns for a specified employee ID. Compile the code and remove syntax errors, if any.

b. Create an anonymous block and call the procedure using variables for the two OUT parameters – one for the salary and the other for the job ID. Display the salary and job ID for employee ID 120.



1. Invoke the procedure again, passing an EMPLOYEE\_ID of 300. What happens and why?

