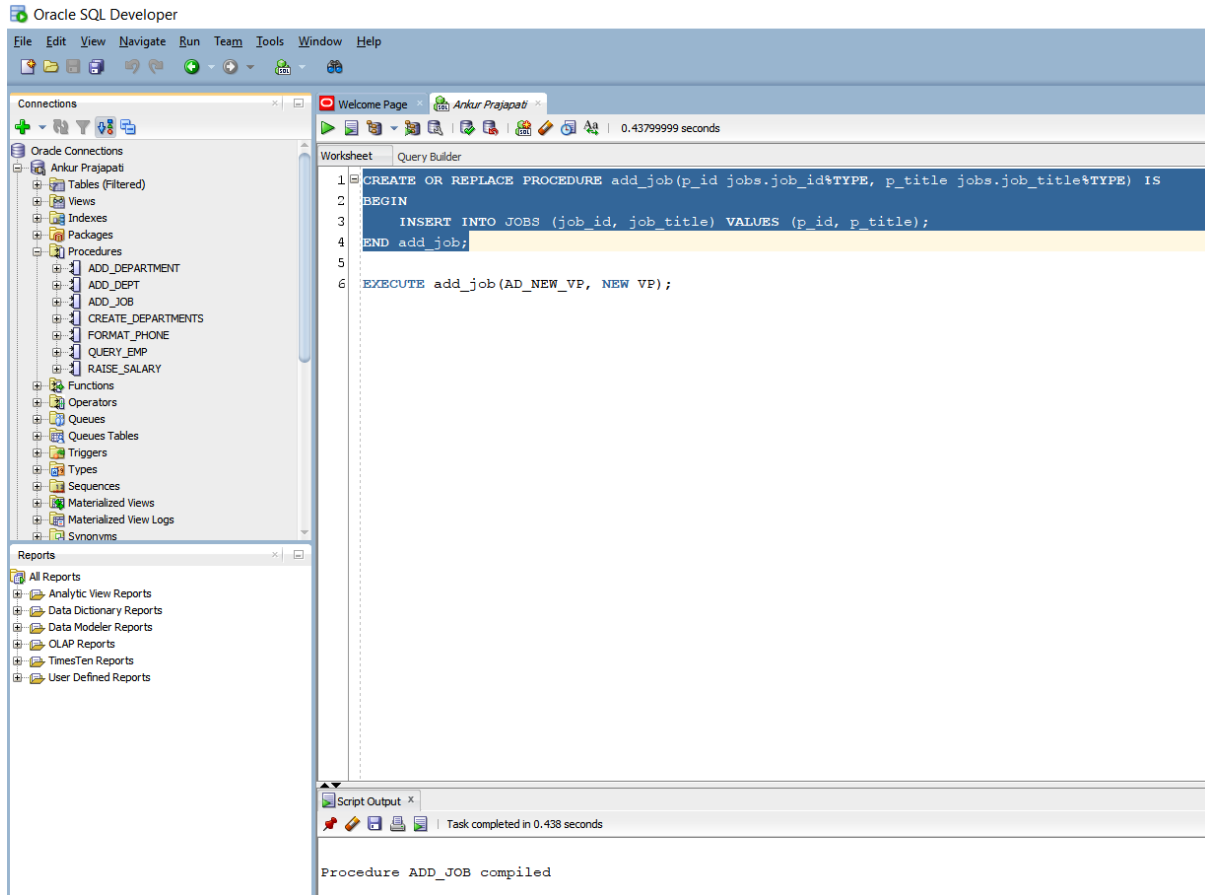
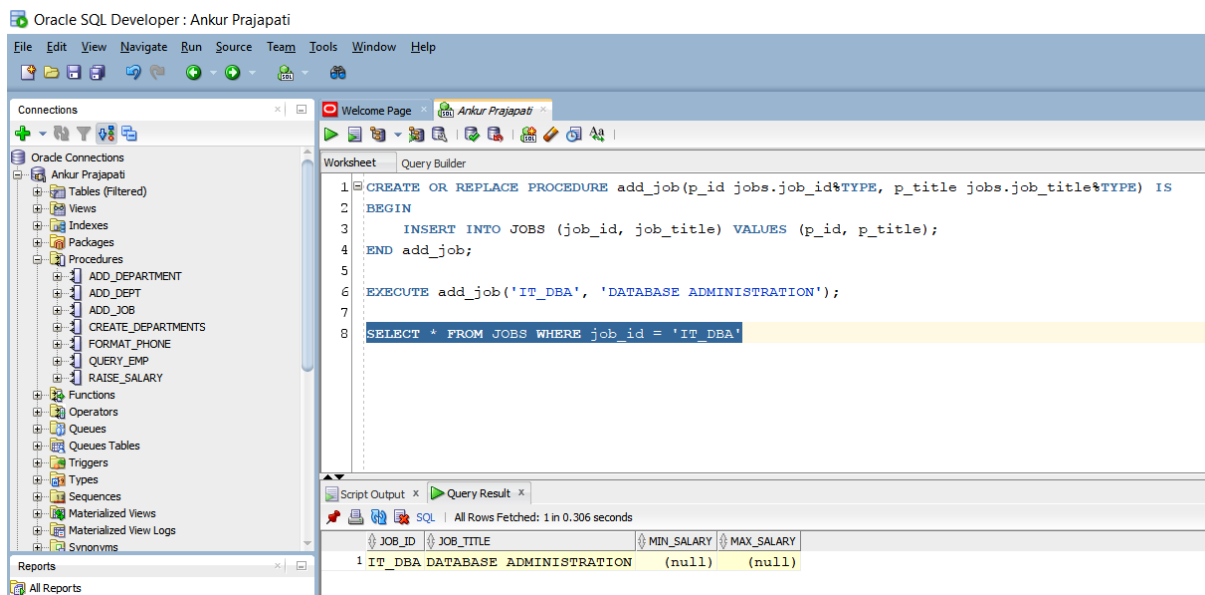


## 1. ADD\_JOB Procedure:



Here we are creating add\_job procedure. As per the statement it will create new add\_job procedure if it doesn't exist otherwise it will drop it and again it will create. It will add job\_id and job\_title with using formal parameters p\_id and p\_title.



Here it will add new job with job\_id having value of 'IT\_DBA'.

The screenshot displays the Oracle SQL Developer interface. The 'Connections' pane on the left shows the 'Ankur Prajapati' connection. The 'Worksheet' pane contains the following PL/SQL code:

```
1 CREATE OR REPLACE PROCEDURE add_job(p_id jobs.job_id%TYPE, p_title jobs.job_title%TYPE) IS
2 BEGIN
3     INSERT INTO JOBS (job_id, job_title) VALUES (p_id, p_title);
4 END add_job;
5
6 EXECUTE add_job('IT_DBA', 'DATABASE ADMINISTRATION');
7
8 SELECT * FROM JOBS WHERE job_id = 'IT_DBA';
9
10 EXECUTE add_job('ST_MAN', 'Stock Manager');
```

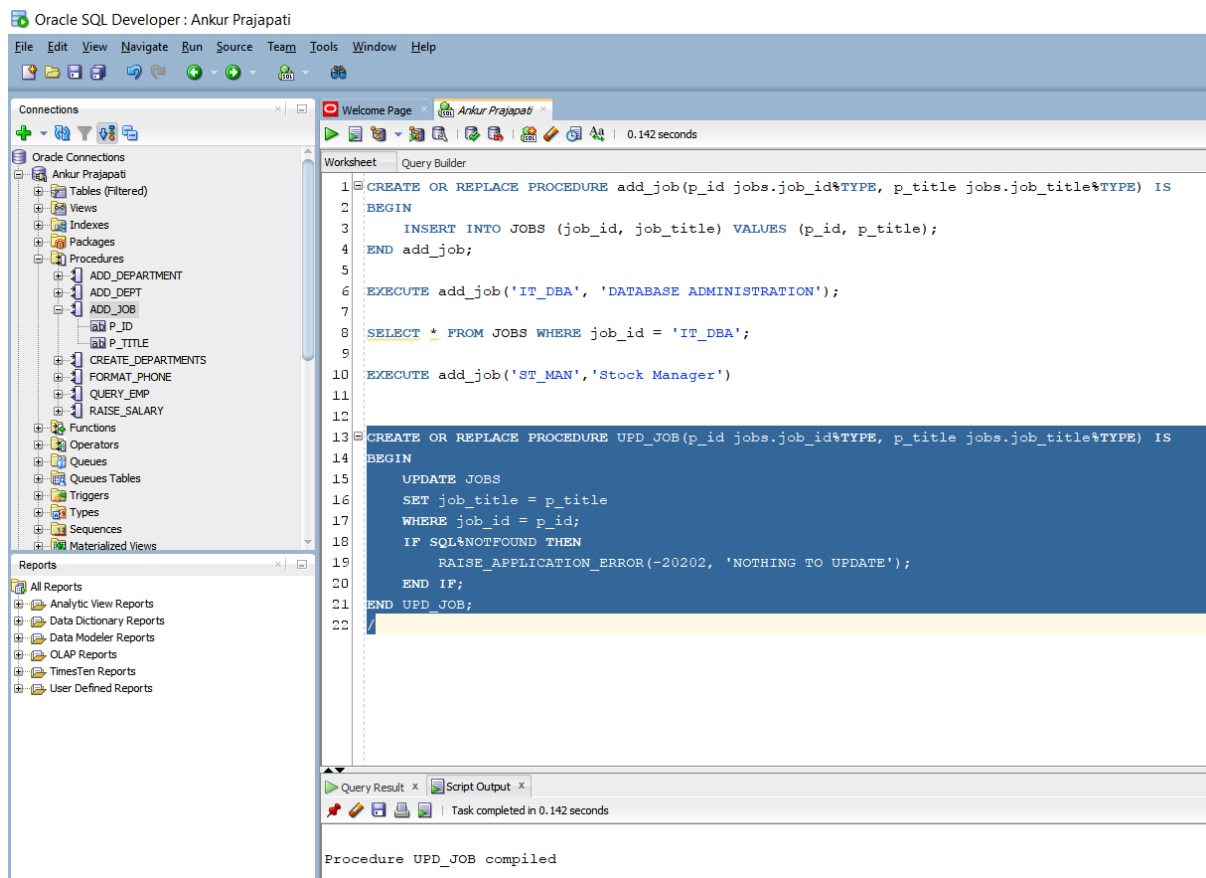
The 'Script Output' pane shows the execution results, including two error messages:

```
Error starting at line : 10 in command -
BEGIN add_job('ST_MAN','Stock Manager'); END;
Error report -
ORA-06550: line 1, column 119:
PLS-00905: object N01324892.ADD_JOB is invalid
ORA-06550: line 1, column 119:
PL/SQL: Statement ignored
06550. 00000 - "line %s, column %s:\n%s"
*Cause:      Usually a PL/SQL compilation error.
*Action:

Error starting at line : 10 in command -
BEGIN add_job('ST_MAN','Stock Manager'); END;
Error report -
ORA-00001: unique constraint (N01324892.JOB_ID_PK) violated
ORA-06512: at "N01324892.ADD_JOB", line 3
ORA-06512: at line 1
00001. 00000 - "unique constraint (%s.%s) violated"
*Cause:      An UPDATE or INSERT statement attempted to insert a duplicate key.
              For Trusted Oracle configured in DBMS MAC mode, you may see
              this message if a duplicate entry exists at a different level.
*Action:      Either remove the unique restriction or do not insert the key.
```

Here I am trying to add new job with job\_id = 'ST\_MAN' and job\_title = 'Stock Manager' but it gives error of unique key constraint violation. It means there is same data with that job\_id is available. It gives solution that either remove the unique restriction or do not insert the key.

## 2. UPD\_JOB Procedure:



Here I am creating a procedure named UPD\_JOB with formal parameters p\_id with type of job\_id, p\_title with type of job\_title. In this it sets job\_title with p\_title. If SQL%NOTFOUND then it will return message showing 'Nothing To UPDATE'.

Basically, if job\_id is not there then it won't update and will return error.

The screenshot shows the Oracle SQL Developer interface. The left pane displays the 'Connections' tree with 'Ankur Prajapati' selected. The 'Reports' pane shows 'All Reports'. The main workspace contains a PL/SQL script with the following code:

```
1 CREATE OR REPLACE PROCEDURE add_job(p_id jobs.job_id%TYPE, p_title jobs.job_title%TYPE) IS
2 BEGIN
3     INSERT INTO JOBS (job_id, job_title) VALUES (p_id, p_title);
4 END add_job;
5
6 EXECUTE add_job('IT_DBA', 'DATABASE ADMINISTRATION');
7
8 SELECT * FROM JOBS WHERE job_id = 'IT_DBA';
9
10 EXECUTE add_job('ST_MAN', 'Stock Manager')
11
12
13 CREATE OR REPLACE PROCEDURE UPD_JOB(p_id jobs.job_id%TYPE, p_title jobs.job_title%TYPE) IS
14 BEGIN
15     UPDATE JOBS
16     SET job_title = p_title
17     WHERE job_id = p_id;
18     IF SQL%NOTFOUND THEN
19         RAISE_APPLICATION_ERROR(-20202, 'NOTHING TO UPDATE');
20     END IF;
21 END UPD_JOB;
22 /
23
24 EXECUTE UPD_JOB('IT_DBA', 'DATA ADMINISTRATION');
25
26 SELECT * FROM JOBS WHERE job_id = 'IT_DBA';
```

The 'Query Result' pane at the bottom shows the output of the final query:

JOB_ID	JOB_TITLE	MIN_SALARY	MAX_SALARY
IT_DBA	DATA ADMINISTRATION	(null)	(null)

Here I am trying to update job with job\_id = 'IT\_DBA'. First it was named as 'DATABASE ADMINISTRATION' to 'DATA ADMINISTRATION'.

The screenshot displays the Oracle SQL Developer interface. The left pane shows the 'Connections' tree with 'Ankur Prajapati' selected, and the 'Reports' tree below it. The main workspace shows a PL/SQL script with the following code:

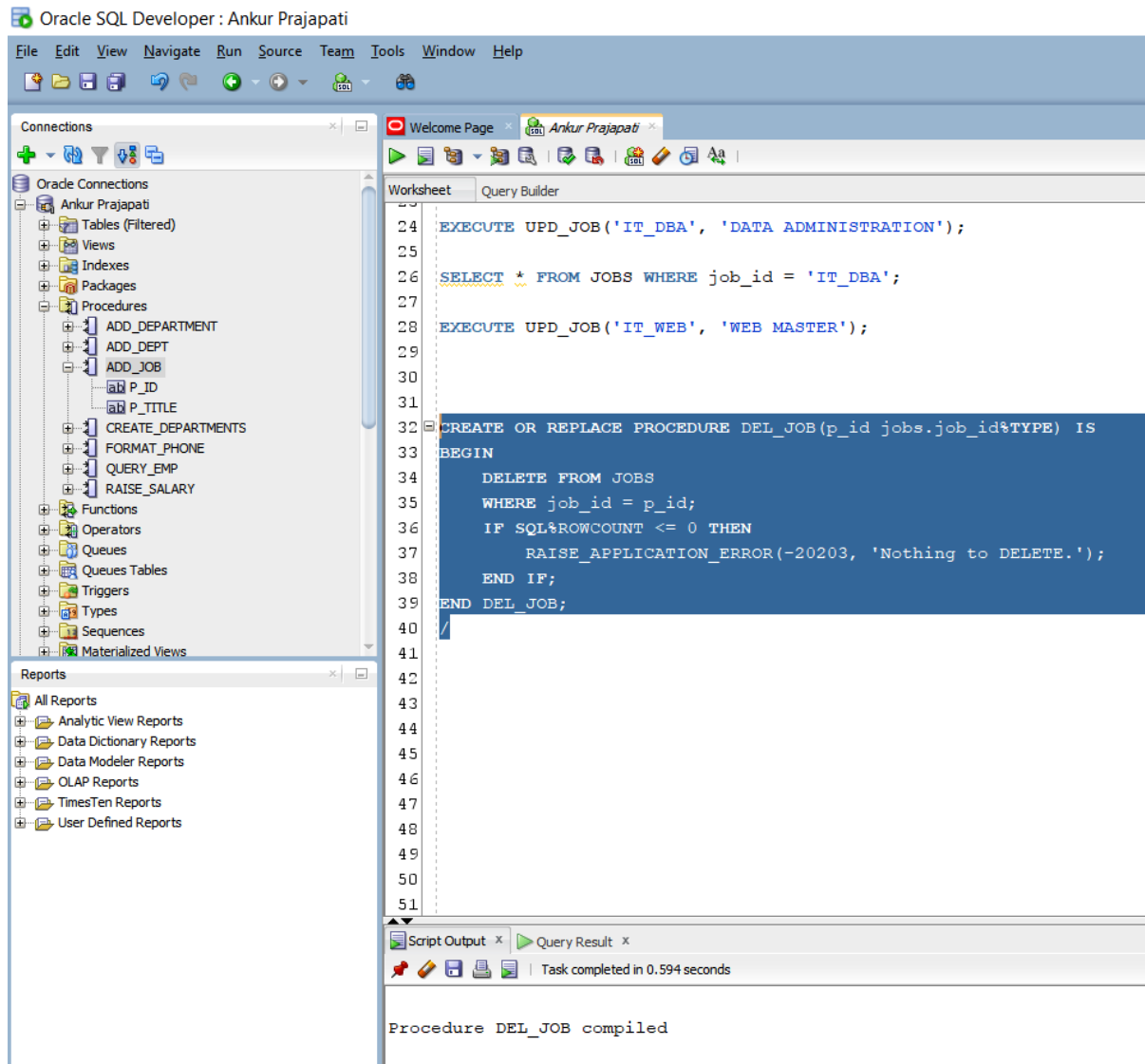
```
1 CREATE OR REPLACE PROCEDURE add_job(p_id jobs.  
2 BEGIN  
3     INSERT INTO JOBS (job_id, job_title) VALUE  
4 END add_job;  
5  
6 EXECUTE add_job('IT_DBA', 'DATABASE ADMINISTRATOR');  
7  
8 SELECT * FROM JOBS WHERE job_id = 'IT_DBA';  
9  
10 EXECUTE add_job('ST_MAN', 'Stock Manager')  
11  
12  
13 CREATE OR REPLACE PROCEDURE UPD_JOB(p_id jobs.  
14 BEGIN  
15     UPDATE JOBS  
16     SET job_title = p_title  
17     WHERE job_id = p_id;  
18     IF SQL%NOTFOUND THEN  
19         RAISE_APPLICATION_ERROR(-20202, 'NOTHING TO UPDATE');  
20     END IF;  
21 END UPD_JOB;  
22 /  
23  
24 EXECUTE UPD_JOB('IT_DBA', 'DATA ADMINISTRATION');  
25  
26 SELECT * FROM JOBS WHERE job_id = 'IT_DBA';  
27  
28 EXECUTE UPD_JOB('IT_WEB', 'WEB MASTER');
```

The bottom pane shows the 'Script Output' window with the following error message:

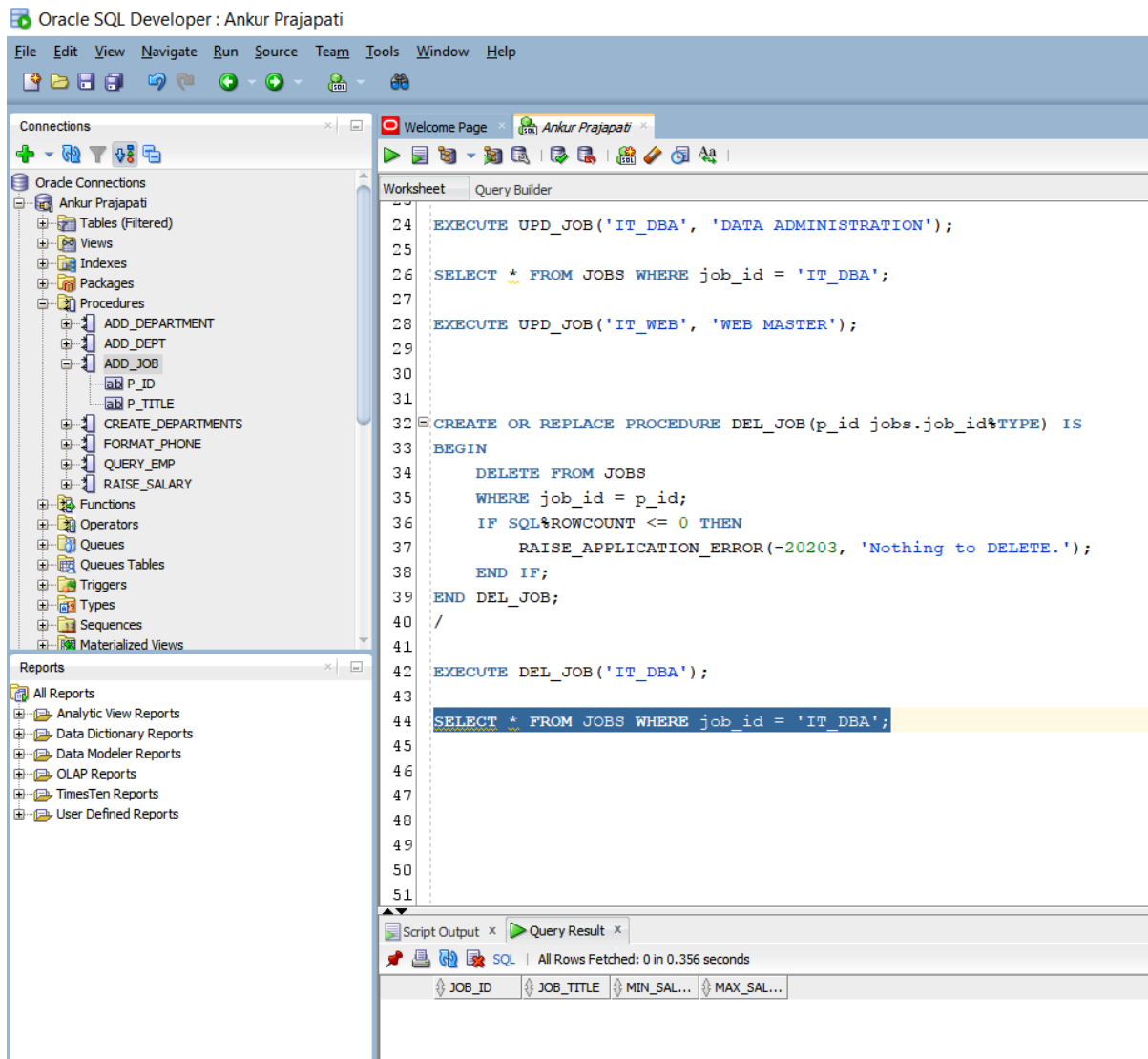
```
Error starting at line : 28 in command -  
BEGIN UPD_JOB('IT_WEB', 'WEB MASTER'); END;  
Error report -  
ORA-20202: NOTHING TO UPDATE  
ORA-06512: at "N01324892.UPD_JOB", line 7  
ORA-06512: at line 1
```

Here I am trying to update job having job\_id = 'IT\_WEB' but it's not stored in database so it gives an error message notifying that 'Nothing to Update' associated to SQLERRORCode.

### 3. DEL\_JOB Procedure:



Here I am creating procedure named DEL\_JOB with formal parameter p\_id having type of job\_id. It deletes job having same id in database. If there is no row available to delete (SQL%ROWCOUNT is less than 0) then it will give error message that 'Nothing To DELETE' associated with SQLERRORCODE -20203.



There is data stored in database with job\_id = 'IT\_DBA' and I have deleted it from here. That's why it is showing us the blank record.

The screenshot displays the Oracle SQL Developer interface. On the left, the 'Connections' pane shows the 'Ankur Prajapati' connection. Below it, the 'Reports' pane lists various report types. The main workspace is divided into a 'Worksheet' and a 'Query Builder'. The 'Worksheet' contains a PL/SQL script with the following code:

```
24 EXECUTE UPD_JOB('IT_DBA', 'DATA ADMINISTRATION');
25
26 SELECT * FROM JOBS WHERE job_id = 'IT_DBA';
27
28 EXECUTE UPD_JOB('IT_WEB', 'WEB MASTER');
29
30
31
32 CREATE OR REPLACE PROCEDURE DEL_JOB(p_id jobs.job_id%TYPE) IS
33 BEGIN
34     DELETE FROM JOBS
35     WHERE job_id = p_id;
36     IF SQL%ROWCOUNT <= 0 THEN
37         RAISE_APPLICATION_ERROR(-20203, 'Nothing to DELETE.');
```

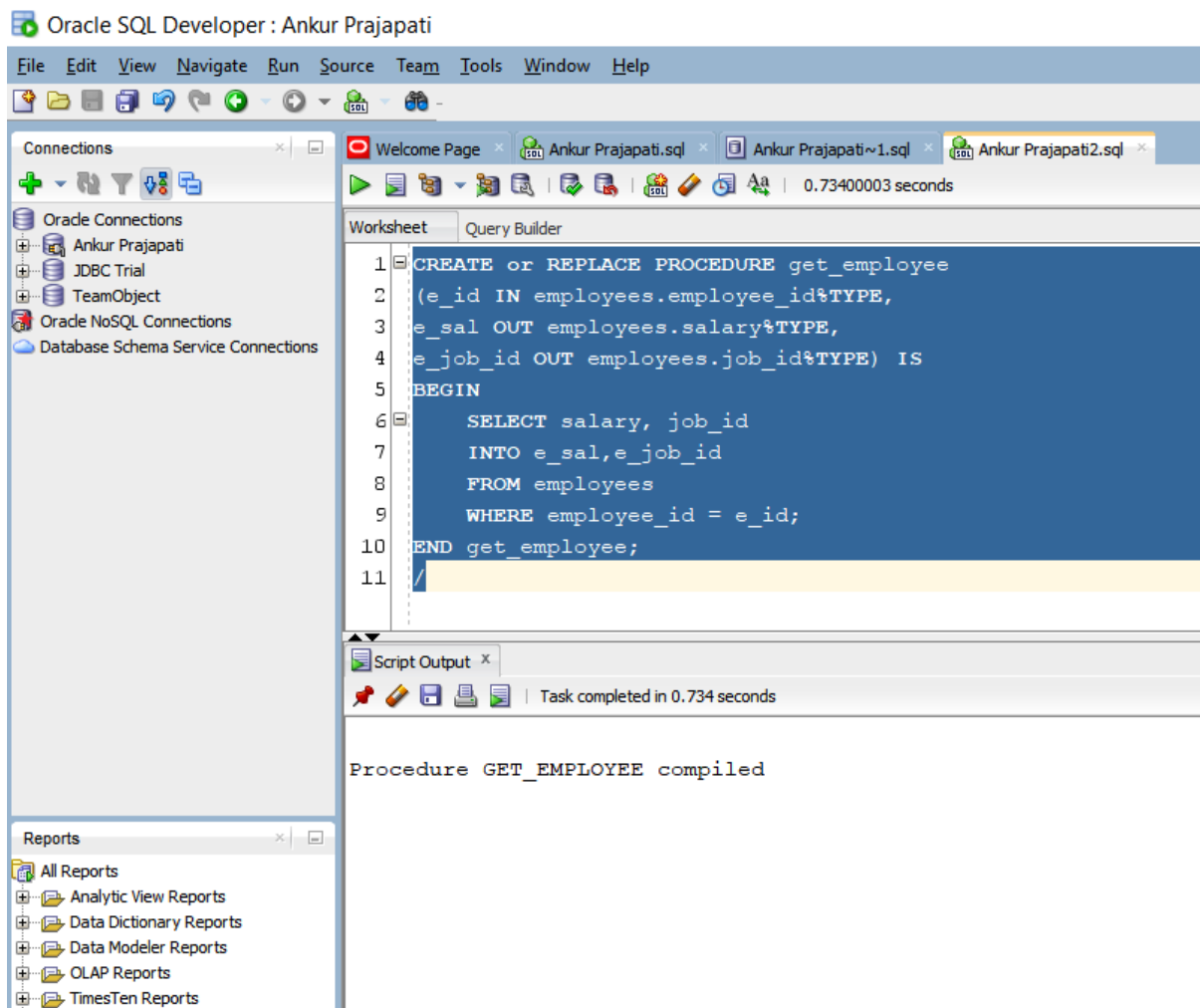
The script is executed, and the 'Script Output' pane at the bottom shows the following error message:

```
Error starting at line : 46 in command -
BEGIN DEL_JOB('IT_WEB'); END;
Error report -
ORA-20203: Nothing to DELETE.
ORA-06512: at "N01324892.DEL_JOB", line 6
ORA-06512: at line 1
```

Here it will give error that there is nothing to delete because there is not record of job\_id = 'IT\_WEB'.



## 4. GET\_EMPLOYEE Procedure:



Here I am creating get\_employee procedure with formal parameters e\_Id with type of employees.employee\_id, e\_sal with type of employees.salary and e\_job\_id with type of employees.job\_id where employee\_id(actual parameter) is same as e\_id(formal parameters).

The screenshot displays the Oracle SQL Developer interface. The main window shows a PL/SQL script in the 'Worksheet' tab. The script defines a procedure named 'get\_employee' that takes an input parameter 'e\_id' and returns two output parameters, 'e\_sal' and 'e\_job\_id'. The procedure uses a SELECT statement to retrieve salary and job\_id for the specified employee\_id. Below the procedure definition, there is a block of code that declares two variables, 'v\_salary' and 'v\_job', and then executes the 'get\_employee' procedure with '120' as the input. Finally, it prints the values of 'v\_salary' and 'v\_job'.

```
1 CREATE or REPLACE PROCEDURE get_employee
2 (e_id IN employees.employee_id%TYPE,
3 e_sal OUT employees.salary%TYPE,
4 e_job_id OUT employees.job_id%TYPE) IS
5 BEGIN
6     SELECT salary, job_id
7     INTO e_sal, e_job_id
8     FROM employees
9     WHERE employee_id = e_id;
10 END get_employee;
11 /
12
13 VARIABLE v_salary NUMBER
14 VARIABLE v_job VARCHAR2(15)
15 EXECUTE get_employee(120, :v_salary, :v_job)
16 PRINT v_salary v_job
```

The 'Script Output' window at the bottom shows the results of the execution. It indicates that the procedure 'GET\_EMPLOYEE' was compiled successfully and that the PL/SQL procedure completed successfully. The output shows the values of the bind variables 'v\_salary' and 'v\_job'.

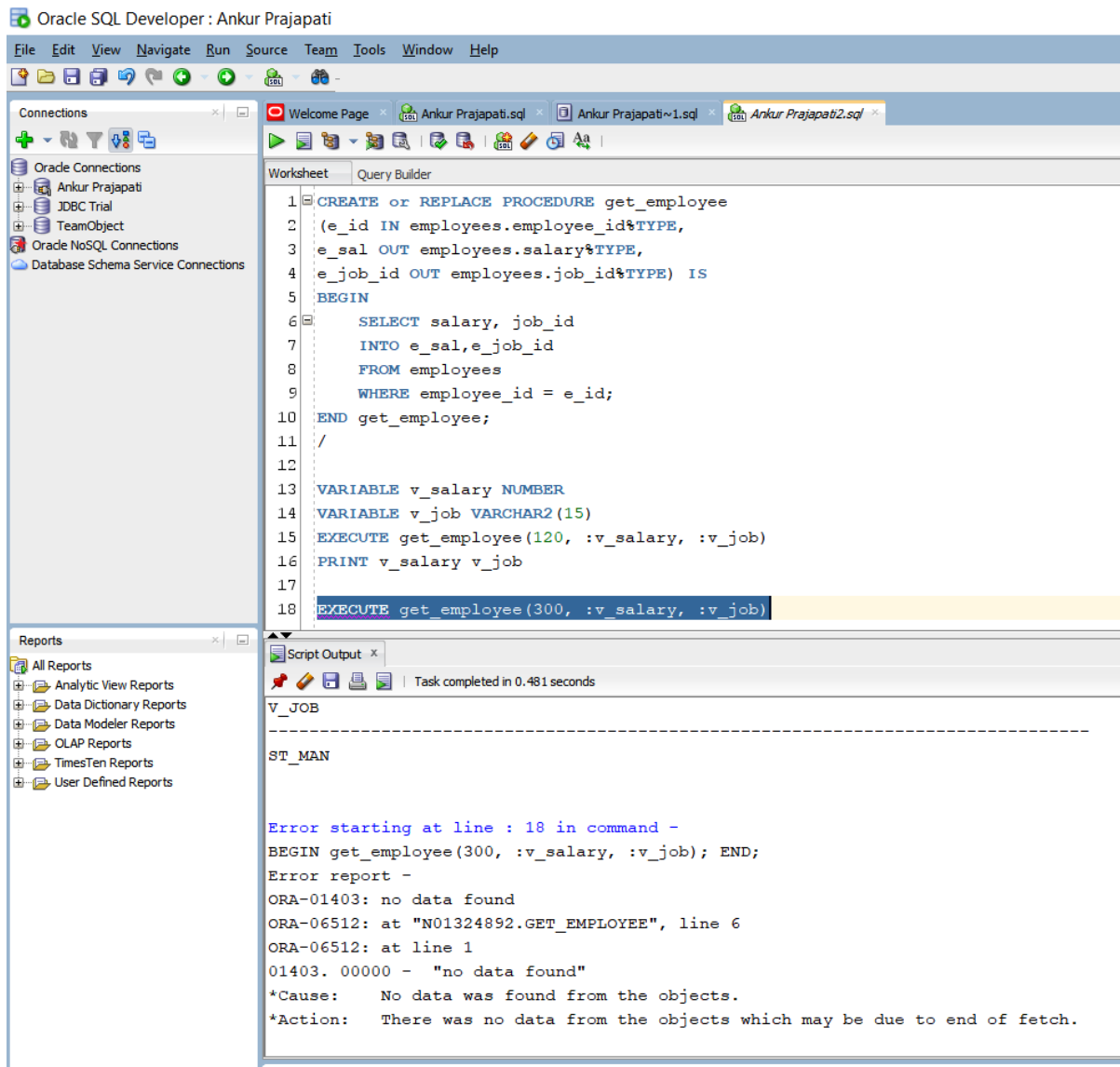
```
PL/SQL procedure successfully completed.

V_SALARY
-----
      8000

V_JOB
-----
ST_MAN
```

Here v\_salary of NUMBER type and v\_job of VARCHAR2 type is declared.

Here employee\_id is 120 and based on that salary and job\_id associated with it will be printed using bind variable. Note that here e\_id is of TYPE IN which means we have to give it, where v\_salary and v\_job is of TYPE OUT that means it will give data to us.



The screenshot displays the Oracle SQL Developer interface. The main window shows a PL/SQL script in the Worksheet tab. The script defines a procedure named `get_employee` that takes an employee ID as input and returns the salary and job ID. The procedure is then executed twice: first with ID 120, and then with ID 300. The execution results are shown in the Script Output window, indicating that no data was found for the second execution.

```
1 CREATE or REPLACE PROCEDURE get_employee
2 (e_id IN employees.employee_id%TYPE,
3 e_sal OUT employees.salary%TYPE,
4 e_job_id OUT employees.job_id%TYPE) IS
5 BEGIN
6     SELECT salary, job_id
7     INTO e_sal, e_job_id
8     FROM employees
9     WHERE employee_id = e_id;
10 END get_employee;
11 /
12
13 VARIABLE v_salary NUMBER
14 VARIABLE v_job VARCHAR2(15)
15 EXECUTE get_employee(120, :v_salary, :v_job)
16 PRINT v_salary v_job
17
18 EXECUTE get_employee(300, :v_salary, :v_job)
```

Script Output x

Task completed in 0.481 seconds

V\_JOB

ST\_MAN

Error starting at line : 18 in command -

BEGIN get\_employee(300, :v\_salary, :v\_job); END;

Error report -

ORA-01403: no data found

ORA-06512: at "N01324892.GET\_EMPLOYEE", line 6

ORA-06512: at line 1

01403. 00000 - "no data found"

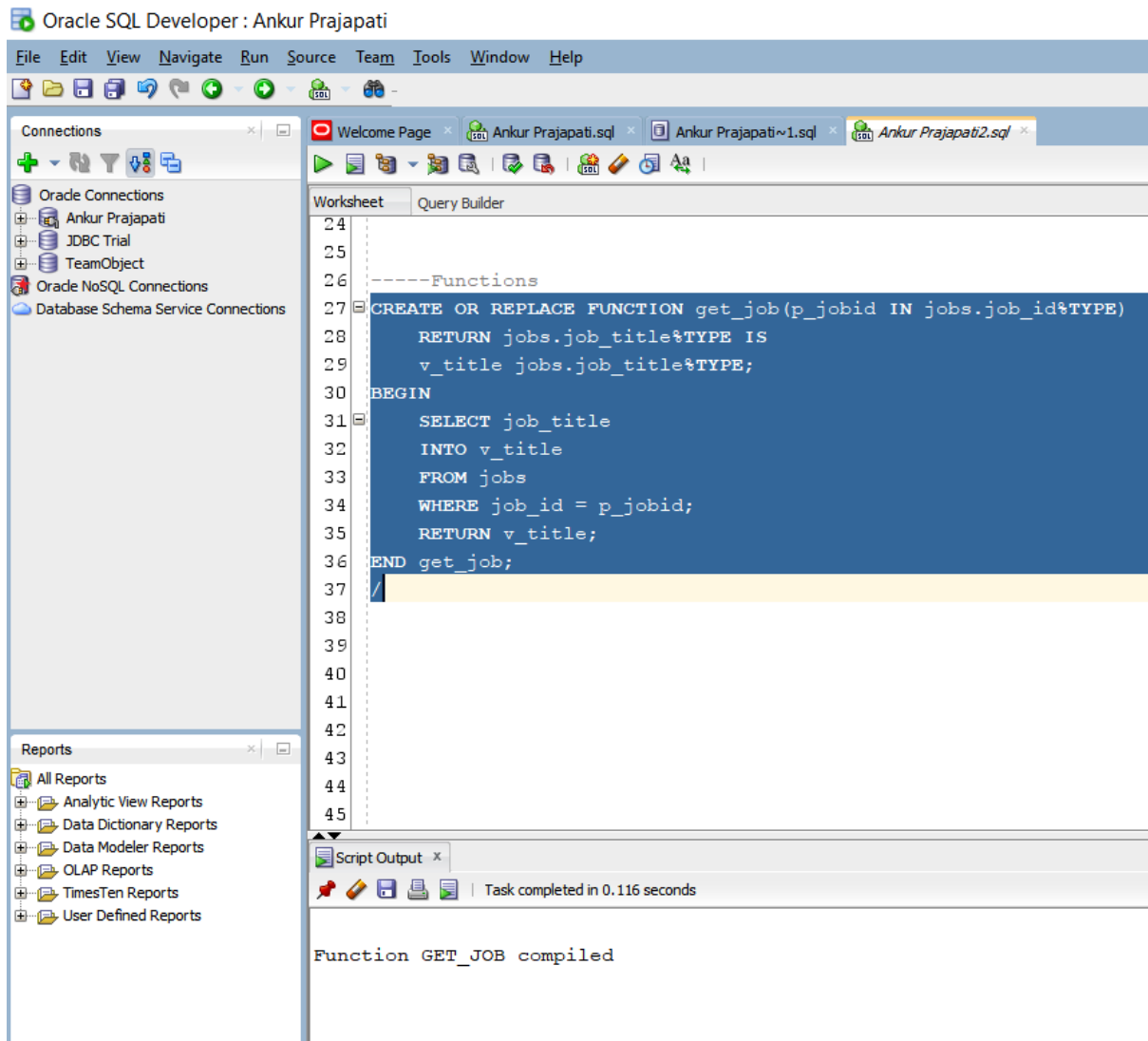
\*Cause: No data was found from the objects.

\*Action: There was no data from the objects which may be due to end of fetch.

Here we tried to fetch data from employees table having employee id 300. But as you can see, we have no data for this. So that it gives an error no data found.

## Practice 3: Creating Functions:

## 1. GET\_JOB Function:



GET\_JOB is compiled

The screenshot displays the Oracle SQL Developer interface. The top menu bar includes File, Edit, View, Navigate, Run, Source, Team, Tools, Window, and Help. The left sidebar contains a 'Connections' panel with a tree view showing 'Oracle Connections', 'Ankur Prajapati', 'JDBC Trial', 'TeamObject', 'Oracle NoSQL Connections', and 'Database Schema Service Connections'. Below it is a 'Reports' panel with a tree view showing 'All Reports', 'Analytic View Reports', 'Data Dictionary Reports', 'Data Modeler Reports', 'OLAP Reports', 'TimesTen Reports', and 'User Defined Reports'. The main workspace is divided into a 'Worksheet' and a 'Query Builder' tab. The 'Worksheet' tab is active, showing a PL/SQL script. The script defines a function 'get\_job' that takes a job ID and returns the job title. It then declares a variable 'b\_title', calls the function with 'SA\_REP', and prints the result. The 'Script Output' panel at the bottom shows the execution results: 'Function GET\_JOB compiled', 'PL/SQL procedure successfully completed.', and the output 'Sales Representative'.

```
24  
25  
26 -----Functions  
27 CREATE OR REPLACE FUNCTION get_job(p_jobid IN jobs.job_id%TYPE)  
28   RETURN jobs.job_title%TYPE IS  
29   v_title jobs.job_title%TYPE;  
30 BEGIN  
31   SELECT job_title  
32   INTO v_title  
33   FROM jobs  
34   WHERE job_id = p_jobid;  
35   RETURN v_title;  
36 END get_job;  
37 /  
38  
39 VARIABLE b_title VARCHAR2(35)  
40 EXECUTE :b_title := get_job('SA_REP');  
41 PRINT b_title  
42  
43  
44  
45
```

Script Output x

Task completed in 0.16 seconds

Function GET\_JOB compiled

PL/SQL procedure successfully completed.

B\_TITLE

-----

Sales Representative

Here it gives job title from given job\_id.  
For example, SA\_REP to SALES REPRESENTATIVE.

## 2. GET\_ANNUAL\_COMP Function:

The screenshot displays the Oracle SQL Developer interface. The left pane shows the 'Connections' tree with 'Ankur Prajapati' selected. The main workspace is divided into a 'Worksheet' and a 'Script Output' pane. The 'Worksheet' contains the following SQL code:

```
33 FROM jobs
34 WHERE job_id = p_jobid;
35 RETURN v_title;
36 END get_job;
37 /
38
39 VARIABLE b_title VARCHAR2(35)
40 EXECUTE :b_title := get_job('SA_REP');
41 PRINT b_title
42
43
44 CREATE OR REPLACE FUNCTION get_annual_comp(
45 p_sal IN employees.salary%TYPE,
46 p_comm IN employees.commission_pct%TYPE)
47 RETURN NUMBER IS
48 BEGIN
49 RETURN (NVL(p_sal,0) * 12 + (NVL(p_comm,0) * NVL(p_sal,0) * 12));
50 END get_annual_comp;
51 /
52
53
54
```

The 'Script Output' pane shows the results of the execution:

```
PL/SQL procedure successfully completed.

B_TITLE
-----
Sales Representative

Function GET_ANNUAL_COMP compiled
```

Get\_annual\_comp is compiled. And it returns annual compound salary.

Note that here NVL will ignore null values and it converts it to 0.

The screenshot displays the Oracle SQL Developer interface. The main window shows a PL/SQL script with the following code:

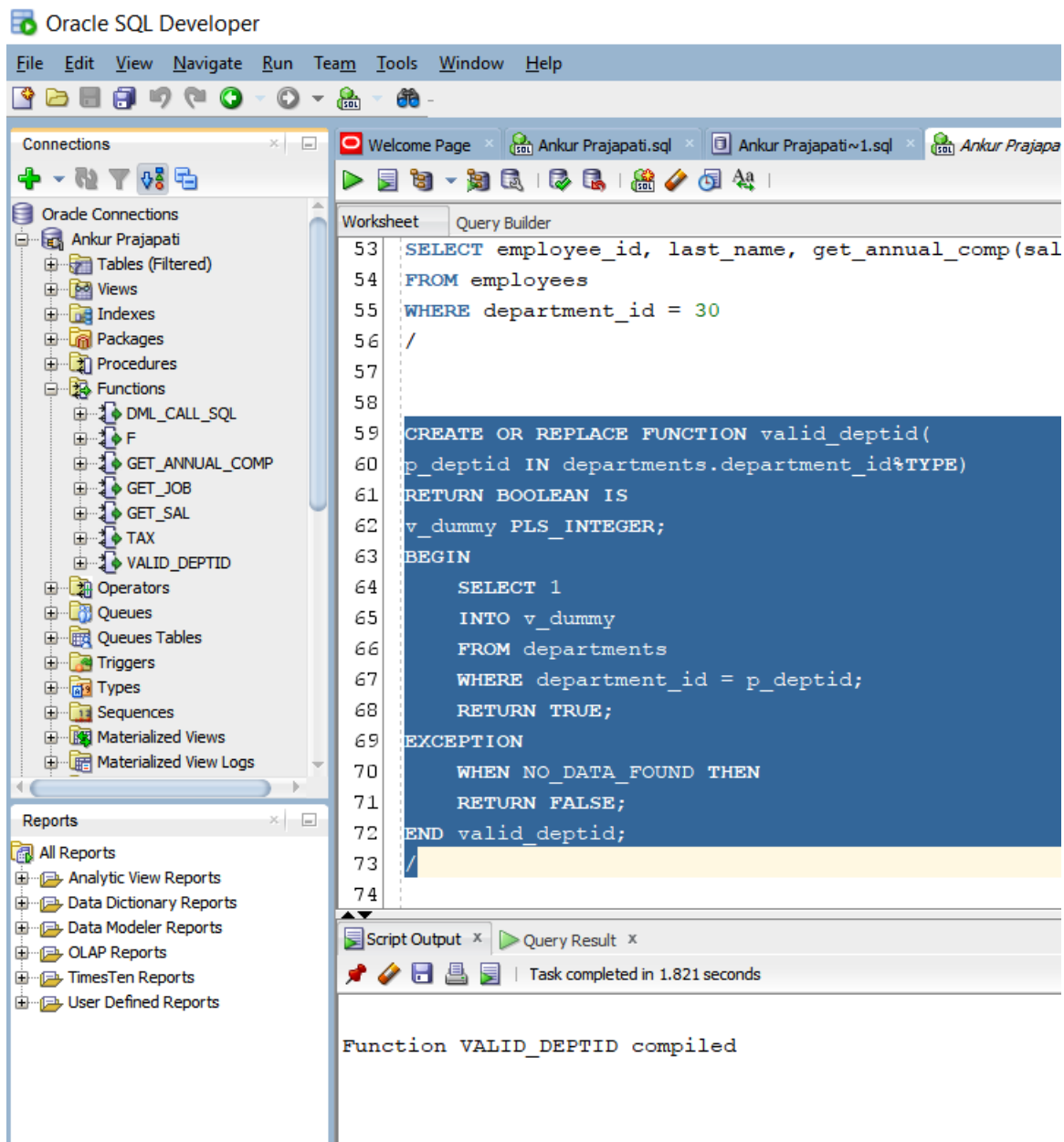
```
37 /
38
39 VARIABLE b_title VARCHAR2(35)
40 EXECUTE :b_title := get_job('SA_REP');
41 PRINT b_title
42
43
44 CREATE OR REPLACE FUNCTION get_annual_comp(
45   p_sal IN employees.salary%TYPE,
46   p_comm IN employees.commission_pct%TYPE)
47   RETURN NUMBER IS
48   BEGIN
49     RETURN (NVL(p_sal,0) * 12 + (NVL(p_comm,0) * NVL(p_sal,0) * 12));
50   END get_annual_comp;
51 /
52
53 SELECT employee_id, last_name, get_annual_comp(salary, commission_pct) "Annual Compensation"
54 FROM employees
55 WHERE department_id = 30
56 /
57
58
```

The script is executed, and the results are displayed in the Query Result window. The results show the employee ID, last name, and annual compensation for employees in department 30.

EMPLOYEE_ID	LAST_NAME	Annual Compensation
114	Raphaely	132000
115	Khoo	37200
116	Baida	34800
117	Tobias	33600
118	Himuro	31200
119	Colmenares	30000

Fetching data of employees with id, last name and annual compensation where department id is 30.

## 3. ADD\_EMPLOYEE:

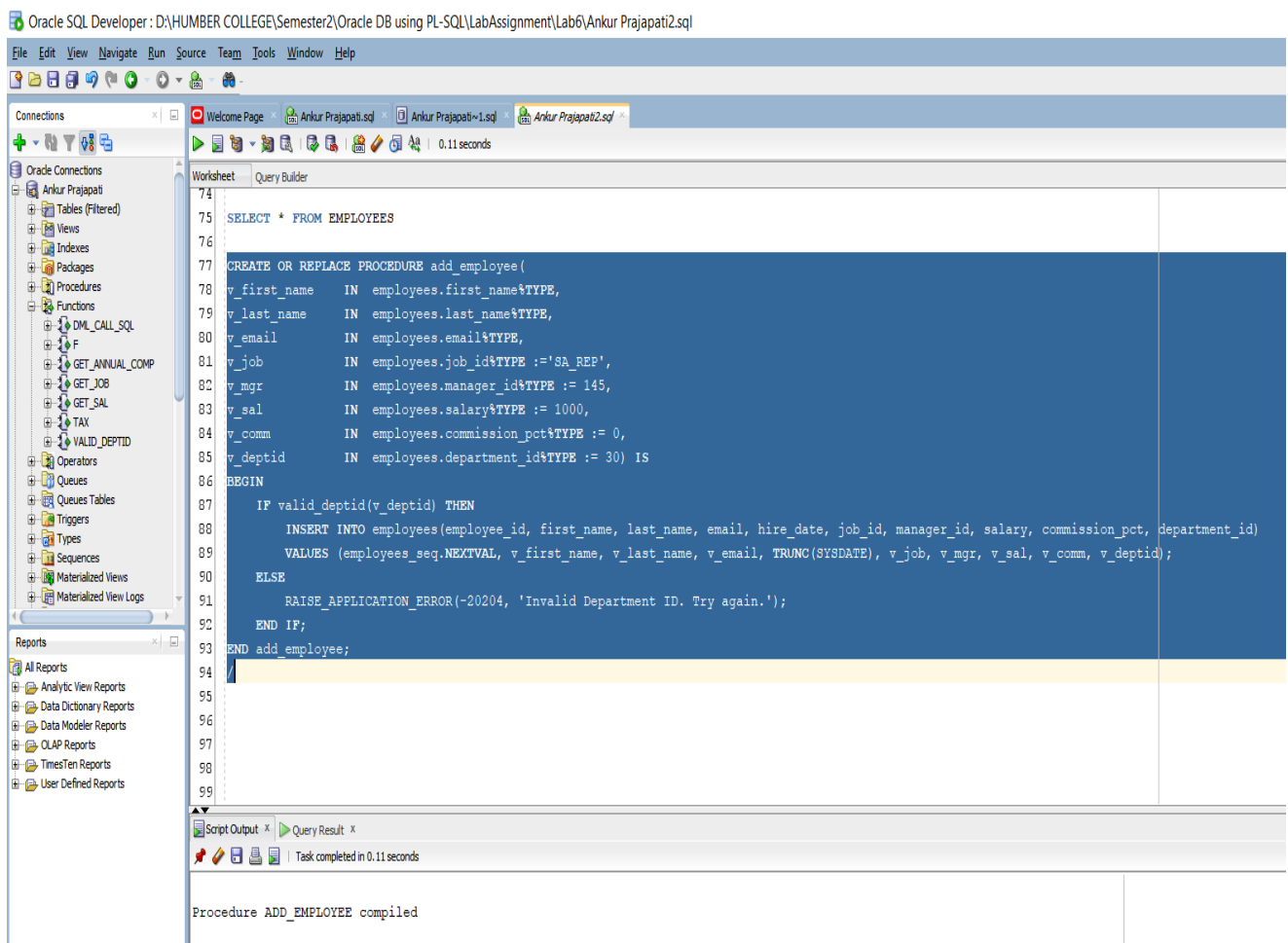


Creation of valid\_deptid function.

We are selecting 1 into v-dummy which is PLS\_INTEGER from departments table if the department id is there with given department id(p\_deptid).

There is an Exception block is also included if we don't have data then it will return false.





The screenshot displays the Oracle SQL Developer interface. The title bar indicates the connection to 'D:\HUMBER COLLEGE\Semester2\Oracle DB using PL-SQL\LabAssignment\Lab6\Ankur Prajapati2.sql'. The 'Connections' pane on the left shows the 'Ankur Prajapati' connection with a tree view of database objects including Tables, Views, Indexes, Packages, Procedures, Functions, and various SQL scripts. The main 'Worksheet' area contains the following SQL code:

```
74  
75 SELECT * FROM EMPLOYEES  
76  
77 CREATE OR REPLACE PROCEDURE add_employee(  
78   v_first_name  IN employees.first_name%TYPE,  
79   v_last_name   IN employees.last_name%TYPE,  
80   v_email       IN employees.email%TYPE,  
81   v_job         IN employees.job_id%TYPE := 'SA_REP',  
82   v_mgr         IN employees.manager_id%TYPE := 145,  
83   v_sal         IN employees.salary%TYPE := 1000,  
84   v_comm        IN employees.commission_pct%TYPE := 0,  
85   v_deptid      IN employees.department_id%TYPE := 30) IS  
86 BEGIN  
87   IF valid_deptid(v_deptid) THEN  
88     INSERT INTO employees(employee_id, first_name, last_name, email, hire_date, job_id, manager_id, salary, commission_pct, department_id)  
89       VALUES (employees_seq.NEXTVAL, v_first_name, v_last_name, v_email, TRUNC(SYSDATE), v_job, v_mgr, v_sal, v_comm, v_deptid);  
90   ELSE  
91     RAISE_APPLICATION_ERROR(-20204, 'Invalid Department ID. Try again.');92   END IF;  
93 END add_employee;  
94  
95  
96  
97  
98  
99
```

Below the code editor, the 'Script Output' pane shows the message: 'Procedure ADD\_EMPLOYEE compiled'. The 'Query Result' pane is empty.

Creation of add\_employee procedure. First, we declared all the formal parameters using data type from employees table.

In executable section, there is one IF condition which checks if the valid\_deptid(v\_deptid) is true or not. If it is true it will insert values into employees table. If it is not true then it will raise an application error notifying Invalid Department ID. Try again.

Oracle SQL Developer: D:\HUMBER COLLEGE\Semester2\Oracle DB using PL-SQL\LabAssignment\Lab6\Ankur Prajapati2.sql

File Edit View Navigate Run Source Team Tools Window Help

Connections

Oracle Connections

- Ankur Prajapati
  - Tables (Filtered)
  - Views
  - Indexes
  - Packages
  - Procedures
  - Functions
    - DML\_CALL\_SQL
    - F
    - GET\_ANNUAL\_COMP
    - GET\_JOB
    - GET\_SAL
    - TAX
    - VALID\_DEPTID
  - Operators
  - Queues
  - Queues Tables
  - Triggers
  - Types
  - Sequences
  - Materialized Views
  - Materialized View Logs

Worksheet Query Builder

```
83 v_sal      IN employees.salary%TYPE := 1000,
84 v_comm     IN employees.commission_pct%TYPE := 0,
85 v_deptid   IN employees.department_id%TYPE := 30) IS
86 BEGIN
87     IF valid_deptid(v_deptid) THEN
88         INSERT INTO employees(employee_id, first_name, last_name, email, hire_date, job_id, manager_id, salary, commission_pct, department_id)
89         VALUES (employees_seq.NEXTVAL, v_first_name, v_last_name, v_email, TRUNC(SYSDATE), v_job, v_mgr, v_sal, v_comm, v_deptid);
90     ELSE
91         RAISE_APPLICATION_ERROR(-20204, 'Invalid Department ID. Try again.');
```

82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100

EXECUTE add\_employee('JANE', 'Harris', 'JAHARRIS', v\_deptid => 15)

Reports

All Reports

- Analytic View Reports
- Data Dictionary Reports
- Data Modeler Reports
- OLAP Reports
- TimesTen Reports
- User Defined Reports

Script Output x Query Result x

Task completed in 0.712 seconds

Error starting at line : 96 in command -

```
BEGIN add_employee('JANE', 'Harris', 'JAHARRIS', v_deptid => 15); END;
```

Error report -

```
ORA-20204: Invalid Department ID. Try again.
ORA-06512: at "N01324892.ADD_EMPLOYEE", line 15
ORA-06512: at line 1
```

Here add\_employee is executed and due to department id miss matching it gives invalid department id.

Oracle SQL Developer: D:\HUMBER COLLEGE\Semester2\Oracle DB using PL-SQL\LabAssignment\Lab6\Ankur Prajapati2.sql

File Edit View Navigate Run Source Team Tools Window Help

Connections

Oracle Connections

Ankur Prajapati

- Tables (Filtered)
- Views
- Indexes
- Packages
- Procedures
- Functions
  - DML\_CALL\_SQL
  - F
  - GET\_ANNUAL\_COMP
  - GET\_JOB
  - GET\_SAL
  - TAX
  - VALID\_DEPTID
- Operators
- Queues
- Queues Tables
- Triggers
- Types
- Sequences
- Materialized Views
- Materialized View Logs

Worksheet

```
84 v_comm IN employees.commission_pct%TYPE := 0;
85 v_deptid IN employees.department_id%TYPE := 30) IS
86 BEGIN
87     IF valid_deptid(v_deptid) THEN
88         INSERT INTO employees(employee_id, first_name, last_name, email, hire_date, job_id, manager_id, salary, commission_pct, department_id)
89         VALUES (employees_seq.NEXTVAL, v_first_name, v_last_name, v_email, TRUNC(SYSDATE), v_job, v_mgr, v_sal, v_comm, v_deptid);
90     ELSE
91         RAISE_APPLICATION_ERROR(-20204, 'Invalid Department ID. Try again.');
```

Script Output x Query Result x

SQL | All Rows Fetched: 1 in 0.007 seconds

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
1	212 Joe	Harris	Jaharris	(null)	27-06-19	SA_REP	1000	0	145	80

Adding new record here in table.

It prints the inserted record here.