UNIVERSITI TUNKU ABDUL RAHMAN

Faculty of Information and Communication Technology (FICT)

UCCD 2203 Database Systems

Session: 202001

No.	Name	Student ID	Practical Group	Programme*	Signature**
1	Tai Jia Wei	1806718	P (2)	CS	[illegible]
2	Ling Kheng Yuan	1703264	P (2)	CS	[illegible]
3	Wee Yiiheen	1604297	P (2)	CS	[illegible]
4	Hwang Jia Min	1900242	P (2)	CS	[illegible]
5	Por Eng Joo	1807157	P (2)	CS	

* - IA/IB/CS/CN/CT

**All Students should attach the signed assessment sheet confirming that, the report is not plagiarized

Marking Scheme

PART 1: (Group Assessment - 50%)		Marks
1.	**Scope of Work (5 marks)** Analyse requirements study (briefly explain the requirements/ office / business rules in the system). PLEASE INCLUDE ANY ASSUMPTIONS THAT YOU MAKE.	
2.	**ER model (10 marks)** You are required to design an ER diagram for the case study given, identify entities, identify relationships, identify associate attribute and determine keys. Check your ERD with the transaction requirements stated in the case.	
3.	**Redesign and EER (10 marks)** Redesign your ER diagram with the new requirements and extending the ERD to EER model, if any.	
4.	**Data Dictionary (10 marks)** Based on EER diagram that you created in part 4, create a data dictionary for the solution. (Make sure the data types (Oracle) selected are appropriate)	
5.	**Tables and records (5 marks)** Create all relations in ERD and insert the necessary records (Minimum 5 record for each table)	
6.	**Script (10 marks)** You are required to submit the SQL schema script with proper codes. Should include Integrity and referential integrity constraints. **Softcopy:** *Include the script in CD*	
PART 1: Total Group Assessment - 50%		

PART 2: (Individual Assessment - 50%)
(Filled in all your group members name and ID)

Student Name	**1. TAI JIA WEI**	**2. LING KHENG YUAN**	**3. WEE YIIHEEN**	**4. HWANG JIA MIN**	**5. POR ENG JOO**
Student ID	**1806718**	**1703264**	**1604297**	**1900242**	**1807157**
Queries (30 marks)					
Stored Procedure (10 marks)					
Function (10 marks)					
PART 2: Total Individual Assessment - 50%					
PART 1 + PART 2 = 100%					

***Minus 5 marks for no DVD/CD labelling (ALL members)**
***Minus 5 marks for not stapling the DVD/CD together with the assignment report. (ALL members)**

Table of Contents

1.0 Scope of Work

1.1 Business Rules

1. There is various type of leave that are available to employee when they are enrolled into the company such as Sick Leave, Annual Leave, and more. There is certain leave that is compulsory entitled to employee due to the law such as Annual Leave must be entitled to permanent employee for at least a number of days based on company's policy. There is also additional leave type such as Examination Leave, Prolong Illness Leave that are offered by some companies.

2. There is certain leave that can be used when other leave type has been fully utilized such as employee can take Prolong Illness Leave when they have fully used up Hospitalization Leave. This is a special condition for employee where they have used up their leave quota but still in need of leave due to unavoidable situation such as being hospitalized.

3. There is certain leave which only can be taken by specific gender such as only female employees are allowed to take Maternity Leave. The reason is that there is certain leave that only needed by specific gender and not all.

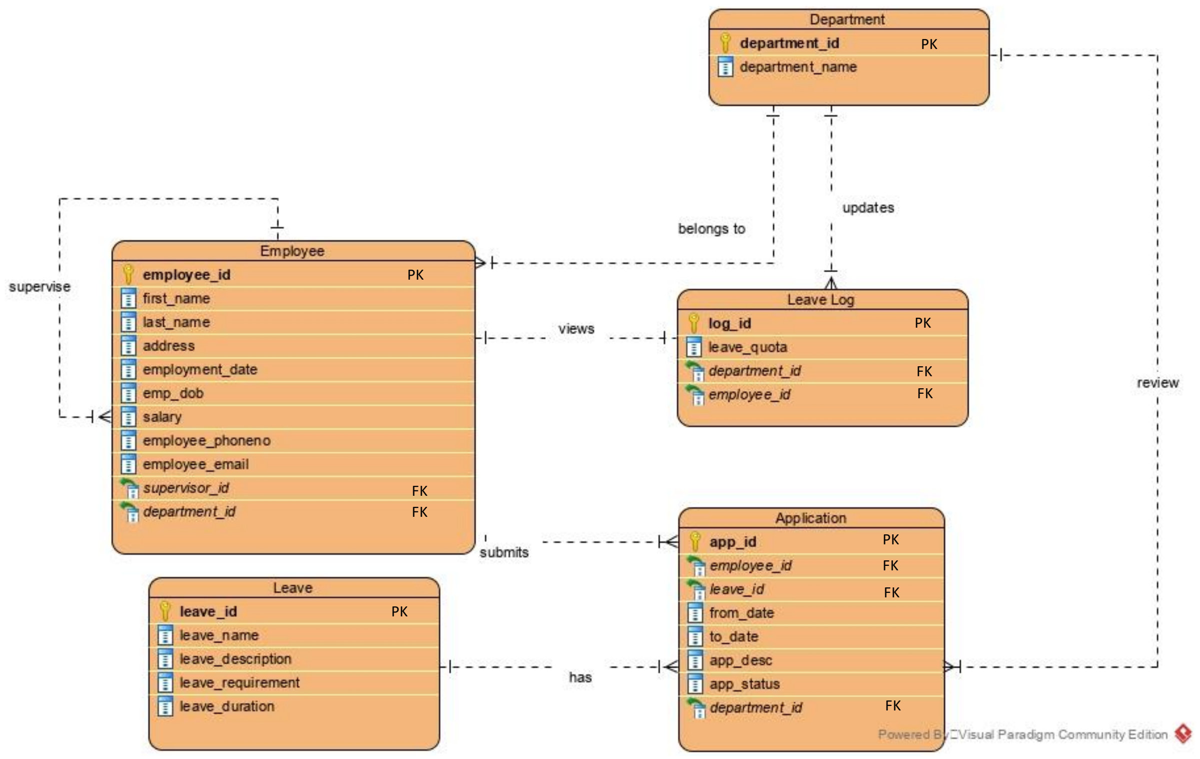
4. There are different number of days of leave that are entitled to an employee based on their year of service in the company. This can be a way of appreciation from the company to its employee for serving such a long period of time. For instance, longer serving employee will have more Annual Leave entitled to them compared to employee that newly joined into the company.

5. Employees are allowed to carry forward their unused leave to next year based on their year of service. Longer serving employees will be able to carry forward more days compared to employees that has a shorter serving time. For example, employees who work for more than 5 years are allowed to carry forward a maximum of 12 days of unused Annual Leave while employees who worked less than that are only allowed to carry forward a maximum of 10 days of unused Annual Leave. This is because longer serving employee will have a better benefit as they have stayed and serve the company for such a long period of time.

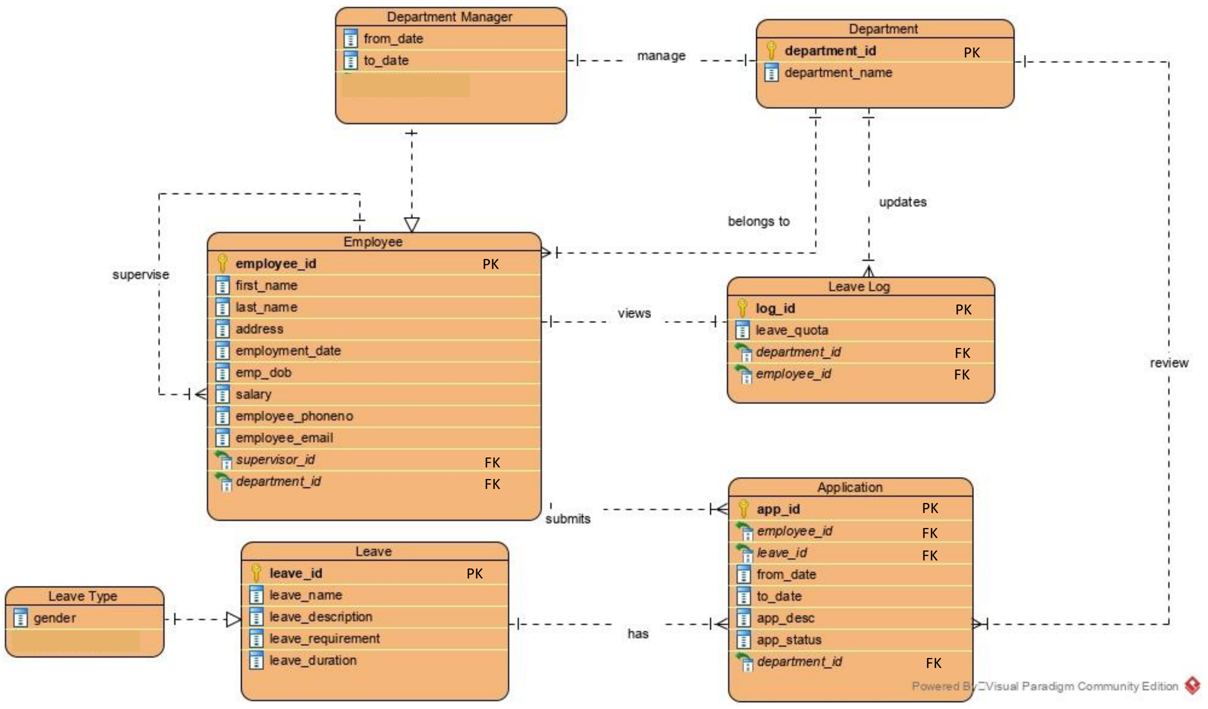
6. Employee would need to present relevant documentation for certain leave. For example, employee would need to present his/her medical certificate as a proof after the individual had taken Sick Leave. This is to ensure that employee do not abuse the system by taking unnecessary leave.

7. Certain leave can be only taken once and cannot be taken anymore for the rest of the service period. For instance, Marriage Leave can be taken only once per employee. This is to prevent any violation that will happen such as the employee divorced and remarried multiple times. This will gradually be unfair to other employee in the company.

2.0 Entity-Relationship Diagram (ERD)



3.0 Enhanced Entity-Relationship Diagram (EERD)



4.0 Data Dictionary

4.1 Employee Table

Field Name	**Field Size**	**Data Type**	**Constraints**	**Example**	**Description**
emp_id	5	NUMBER	Primary Key	18063	Employee number
last_name	30	VARCHAR2	Not Null	Johnny	Employee first name
first_name	30	VARCHAR2	Not Null	Marcus	Employee last name
address	255	VARCHAR2	Not Null	32 Shub Farm Rd. Andover, MA 01810	Employee address
employment_date	-	DATE	Not Null	7/10/2006	Date of employment
emp_dob	-	DATE	Not Null	04/21/1971	Employee Date of Birth
salary	5	NUMBER	-	2800	Employee's Salary
emp_phoneno	15	NUMBER	-	013-5611896	Employee phone number
emp_email	255	VARCHAR2	-	johnnym@hotmail.com	Employee email
supervisor_id	5	NUMBER	Foreign Key	33265	Supervisor number
department_id	5	NUMBER	Foreign Key	44695	Department number

4.2 Leave Table

Field Name	Field Size	Data Type	Constraints	Example	Description
leave_id	5	NUMBER	Primary Key	001	Leave Number
leave_name	30	VARCHAR2	Not Null	Martial Leave	Leave Name
leave_desc	50	VARCHAR2	-	For employee who need to undergo exam	Leave Description
leave_requirement	50	VARCHAR2	-	Medical Certificate	Leave Requirement
leave_duration	2	NUMBER	-	14	Leave Duration

4.3 Application Table

Field Name	Field Size	Data Type	Constraints	Example	Description
app_id	5	NUMBER	Primary Key	19001	Application Number
employee_id	5	NUMBER	Foreign Key	19001	Employee Number
leave_id	5	NUMBER	Foreign Key	1	Leave Number
from_date	-	DATE	Not Null	9/6/2019	Start date of leave
to_date	-	DATE	Not Null	12/5/2019	End date of leave
app_desc	255	VARCHAR2	-	90-day maternity leave	Application Description
app_status	255	VARCHAR2	Not Null	Approved	Status of application
department_id	5	NUMBER	Not Null	1	Department number

4.4 Department Table

Field Name	Field Size	Data Type	Constraints	Example	Description
Department_id	5	NUMBER	Primary Key	18522	Department number
Department_name	30	VARCHAR2	Not Null	Marketing	Department name

4.5 Department Manager

Field Name	Field Size	Data Type	Constraints	Example	Description
from_date	-	DATE	Not Null	3/03/2020	Start date of department manager
to_date	-	DATE	-	1/04/2020	End date of department manager
emp_id	5	NUMBER	Foreign key	54321	Employee number

4.6 Leave Type

Field Name	Field Size	Data Type	Constraints	Example	Description
gender	-	NUMBER	Not Null	1	Employee's gender. For example, 1-male, 2-female.
leave_id	5	NUMBER	Foreign key	101	Leave Number

4.7 Leave log

Field Name	Field Size	Data Type	Constraints	Example	Description
log id	5	NUMBER	Primary Key	2004161	Leave Log Number
leave quota	10	NUMBER	-	20	Balance of Number of Leave
department id	5	NUMBER	Foreign key	18522	Department number
emp id	5	NUMBER	Foreign key	180632	Employee number

5.0 Tables and Records

5.1 Employee

EMP_ ID	LAST_ NAME	FIRST_ NAME	ADDRESS	EMPLOYMENT _DATE	EMP_ DOB	SALARY
19001	McKenzie	Finley	98 Buckingham Rd	09/21/2019	10/24/1994	2000
19002	Morley	Robert	28 Stroude Road	01/28/2015	10/08/1992	3500
19003	Fowler	Summer	87 St James Boulevard	12/03/2011	03/11/1971	4500
19004	Cooke	Harriet	92 Wartnaby Road	05/03/2017	05/01/1972	3000
19005	Ashton	Oscar	47 Ramsgate Rd	11/29/2016	04/28/1977	4200
19006	Coleman	Jordan	28 Bootham Crescent	01/23/2010	01/23/1979	6000
19007	Lamb	Rhys	72 Quay Street	01/25/2013	04/16/1979	5200
19008	Hicks	Isabelle	27 Park Avenue	03/22/2014	03/09/1985	4800
19009	Dickerson	Percy	3857 Hillside Drive	04/02/2018	10/04/1987	3700
19010	Patterson	Joel	20 Crescent Avenue	04/12/2018	03/18/1991	3300

5.2 Employee (Continue)

EMP_PHONENO	EMP_EMAIL	SUPERVISOR_ID	DEPARTMENT_ID
7769436144	finleymc@mail.com	-	58001
7746257082	robert01@mail.com	-	58001
7772907065	sumfow22@mail.com	-	58005
7885399316	harrietc@mail.com	19003	58003
7005444183	oscarman@mail.com	19003	58004
7967887563	coleman97@mail.com	-	58002
7053432547	rhyslamb@mail.com	19003	58004
7847313898	hicksbell@mail.com	-	58002
3395329551	disckerson@mail.com	19003	58005
7715263995	patjoel@mail.com	19003	58002

5.3 Leave

LEAVE_ID	LEAVE_NAME	LEAVE_DESC	LEAVE_ REQUIREMENT	LEAVE_DURATION
001	Unpaid Leave	-	-	-
002	Sick Leave	Paid leave for sick employee	Medical Certificate	-
003	Annual Leave	Annual paid leave for employee	-	-
004	Maternity Leave	Unpaid leave for pregnant employee	Gender	90
005	Examination Leave	For employee who need to undergo exam	Examination Slip	14
006	Marriage Leave	For single employee	Martial Status	14
007	Paternity Leave	For married male employee	Gender	-
008	Prolong Illness Leave	For employee that have long term illness	Medical Report	60
009	Hospitalization Leave	Hospitalized employee eligible to take this leave	Medical Report	21

5.4 Department

DEPARTMENT_ID	DEPARTMENT_NAME
58001	Marketing
58002	Accounting
58003	Production
58004	Financial
58005	Human Resource

5.6 Application

APP_ID	EMPLOYEE_ID	LEAVE_ID	FROM_DATE	TO_DATE	APP_DESC
1	19001	001	02/21/2020	02/22/2020	Emergency Matter
2	19003	007	03/21/2019	03/22/2019	Wife is delivering baby
3	19004	005	05/02/2018	05/16/2018	Oracle Expert Test
4	19007	002	04/26/2014	04/28/2014	Not feeling well
5	19002	009	11/13/2015	12/3/2015	Suspected infection
6	19005	003	05/28/2010	06/3/2010	-

5.7 Application

APP_STATUS	DEPARTMENT_ID
Approved	58001
Approved	58005
Pending	58003
Rejected	58004
Pending	58001
Pending	58004

5.8 Leave Log

LOG_ID	LEAVE_QUOTA	DEPARTMENT_ID	EMP_ID
20001	20	58001	19001
20002	12	58002	19006
20003	21	58005	19003
20004	06	58004	19005
20005	24	58005	19009

6.0 SQL Script

```
CREATE TABLE department
(department_id NUMBER(5),
department_name VARCHAR2(18),
CONSTRAINT department_department_id_pk PRIMARY KEY(department_id),
CONSTRAINT department_department_name_uk UNIQUE(department_name)
);


CREATE TABLE leave
(leave_id NUMBER(5),
leave_name VARCHAR2(30) CONSTRAINT leave_leave_name_nn NOT NULL,
leave_desc VARCHAR2(50),
leave_requirement VARCHAR2(50),
leave_duration NUMBER(2),
CONSTRAINT leave_leave_id_pk PRIMARY KEY(leave_id));


CREATE TABLE employee
(emp_id NUMBER(5),
last_name VARCHAR2(30) CONSTRAINT employee_lastname_nn NOT NULL,
first_name VARCHAR2(30) CONSTRAINT employee_firstname_nn NOT NULL,
emp_address VARCHAR2(255) CONSTRAINT employee_empaddress_nn NOT NULL,
employment_date DATE CONSTRAINT employee_employmentdate_nn NOT NULL,
emp_dob DATE CONSTRAINT employee_empdob_nn NOT NULL,
salary NUMBER(5),
emp_phoneno NUMBER(15),
emp_email VARCHAR2(255) CONSTRAINT employee_empemail_uk UNIQUE,
supervisor_id NUMBER(5),
department_id NUMBER(5),
CONSTRAINT employee_emp_id_pk PRIMARY KEY(emp_id),
CONSTRAINT employee_supervisor_id_fk FOREIGN KEY (supervisor_id) REFERENCES
employee(emp_id) ON DELETE SET NULL,
CONSTRAINT employee_department_id_fk FOREIGN KEY (department_id)
REFERENCES department(department_id));



CREATE TABLE application
(app_id NUMBER(5),
employee_id NUMBER(5) CONSTRAINT application_employee_id_nn NOT NULL,
leave_id NUMBER(5) CONSTRAINT application_leave_id_nn NOT NULL,
from_date DATE CONSTRAINT application_from_date_nn NOT NULL,
to_date DATE CONSTRAINT application_to_date_nn NOT NULL,
app_desc VARCHAR2(255),
app_status VARCHAR2(255) CONSTRAINT application_app_status_nn NOT NULL,
department_id NUMBER(5) CONSTRAINT application_department_id_nn NOT NULL,
CONSTRAINT application_app_id_pk PRIMARY KEY (app_id),
CONSTRAINT application_employee_id_fk FOREIGN KEY (employee_id) REFERENCES
employee(emp_id) ON DELETE CASCADE,
CONSTRAINT application_leave_id_fk FOREIGN KEY (leave_id) REFERENCES
leave(leave_id),
CONSTRAINT application_department_id_fk FOREIGN KEY (department_id)
REFERENCES department(department_id));



CREATE TABLE leavelog
(log_id number(5),
leave_quota number(10),
department_id number(5),
emp_id number(5),
primary key (log_id),
foreign key (department_id) references department(department_id),
foreign key (emp_id) references employee(emp_id) ON DELETE CASCADE);



---- inserting into deparment table
INSERT INTO department VALUES (58001, 'Marketing');
INSERT INTO department VALUES (58002, 'Accounting');

INSERT INTO department VALUES (58003, 'Production');

INSERT INTO department VALUES (58004, 'Financial');

INSERT INTO department VALUES (58005, 'Human Resource');


--- inserting records into LEAVE
INSERT INTO leave VALUES
(001, 'Unpaid Leave', '', '', '');

INSERT INTO leave VALUES
(002, 'Sick Leave', 'Paid leave for sick employee', 'Medical Certificate', '');

INSERT INTO leave VALUES
(003, 'Annual Leave', 'Annual paid leave for employee', '', '');

INSERT INTO leave VALUES
(004, 'Maternity Leave', 'Unpaid leave for pregnant employee', 'Gender', 90);

INSERT INTO leave VALUES
(005, 'Examination Leave', 'For employee who need to undergo exam', 'Examination Slip',
14);

INSERT INTO leave VALUES
(006, 'Marriage Leave', 'For single employee', 'Martial Status', 14);

INSERT INTO leave VALUES
(007, 'Paternity Leave', 'For married male employee', 'Gender', '');
```

```
INSERT INTO leave VALUES

(008, 'Prolong Illness Leave', 'For employee that have long term illness', 'Medical Report',
60);


INSERT INTO leave VALUES

(009, 'Hospitalization Leave', 'Hospitalized employee eligible to take this leave', 'Medical
Report', 21);



--- insterting records into EMPLOYEE

INSERT INTO employee VALUES

(19001, 'McKenzie', 'Finley', '98 Buckingham Rd', TO_DATE('09/21/2019',
'MM/DD/YYYY'), TO_DATE('10/24/1994', 'MM/DD/YYYY'), 2000, 7769436144,
'finleymc@mail.com', '',58001);


INSERT INTO employee VALUES

(19002, 'Morley', 'Robert', '28 Stroude Road', TO_DATE('01/28/2015', 'MM/DD/YYYY'),
TO_DATE('10/08/1992', 'MM/DD/YYYY'), 3500, 7746257082, 'robert01@mail.com', '',
58001);


INSERT INTO employee VALUES

(19003, 'Fowler', 'Summer', '87 St James Boulevard', TO_DATE('12/03/2011',
'MM/DD/YYYY'), TO_DATE('03/11/1971', 'MM/DD/YYYY'), 4500, 7772907065,
'sumfow22@mail.com', '', 58005);


INSERT INTO employee VALUES

(19004, 'Cooke', 'Harriet', '92 Wartnaby Road', TO_DATE('05/03/2017', 'MM/DD/YYYY'),
TO_DATE('05/01/1972', 'MM/DD/YYYY'), 3000, 7885399316, 'harrietc@mail.com', 19003,
58003);


INSERT INTO employee VALUES

(19005, 'Ashton', 'Oscar', '47 Ramsgate Rd', TO_DATE('11/29/2016', 'MM/DD/YYYY'),
TO_DATE('04/28/1977', 'MM/DD/YYYY'), 4200, 7005444183, 'oscarman@mail.com',
19003, 58004);


INSERT INTO employee VALUES
```

```
(19006, 'Coleman', 'Jordan', '28 Bootham Crescent', TO_DATE('01/23/2010',
'MM/DD/YYYY'), TO_DATE('01/23/1979', 'MM/DD/YYYY'), 6000, 7967887563,
'coleman97@mail.com', '', 58002);


INSERT INTO employee VALUES

(19007, 'Lamb', 'Rhys', '72 Quay Street', TO_DATE('02/25/2013', 'MM/DD/YYYY'),
TO_DATE('04/16/1979', 'MM/DD/YYYY'), 5200, 7053432547, 'rhyslamb@mail.com',
19003, 58004);


INSERT INTO employee VALUES

(19008, 'Hicks', 'Isabelle', '27 Park Avenue', TO_DATE('03/22/2014', 'MM/DD/YYYY'),
TO_DATE('03/09/1985', 'MM/DD/YYYY'), 4800, 7847313898, 'hicksbell@mail.com', '',
58002);


INSERT INTO employee VALUES

(19009, 'Dickerson', 'Percy', '3857 Hillside Drive', TO_DATE('04/02/2018',
'MM/DD/YYYY'), TO_DATE('10/04/1987', 'MM/DD/YYYY'), 3700, 3395329551,
'dickerson@mail.com', 19003, 58005);


INSERT INTO employee VALUES

(19010, 'Patterson', 'Joel', '20 Crescent Avenue', TO_DATE('04/12/2018', 'MM/DD/YYYY'),
TO_DATE('03/18/1991', 'MM/DD/YYYY'), 3300, 7715263995, 'patjoel@mail.com', 19003,
58002);



--- insterting records into APPLICATION

INSERT INTO application VALUES

(1, 19001, 001, TO_DATE('02/21/2020', 'MM/DD/YYYY'), TO_DATE('02/22/2020',
'MM/DD/YYYY'), 'Emergency matter', 'Approved', 58001);


INSERT INTO application VALUES

(2, 19003, 007, TO_DATE('03/21/2019', 'MM/DD/YYYY'), TO_DATE('03/22/2019',
'MM/DD/YYYY'), 'Wife is delivering baby', 'Approved', 58005);


INSERT INTO application VALUES

(3, 19004, 005, TO_DATE('05/02/2018', 'MM/DD/YYYY'), TO_DATE('05/16/2018',
'MM/DD/YYYY'), 'Oracle Expert Test', 'Pending', 58003);
```

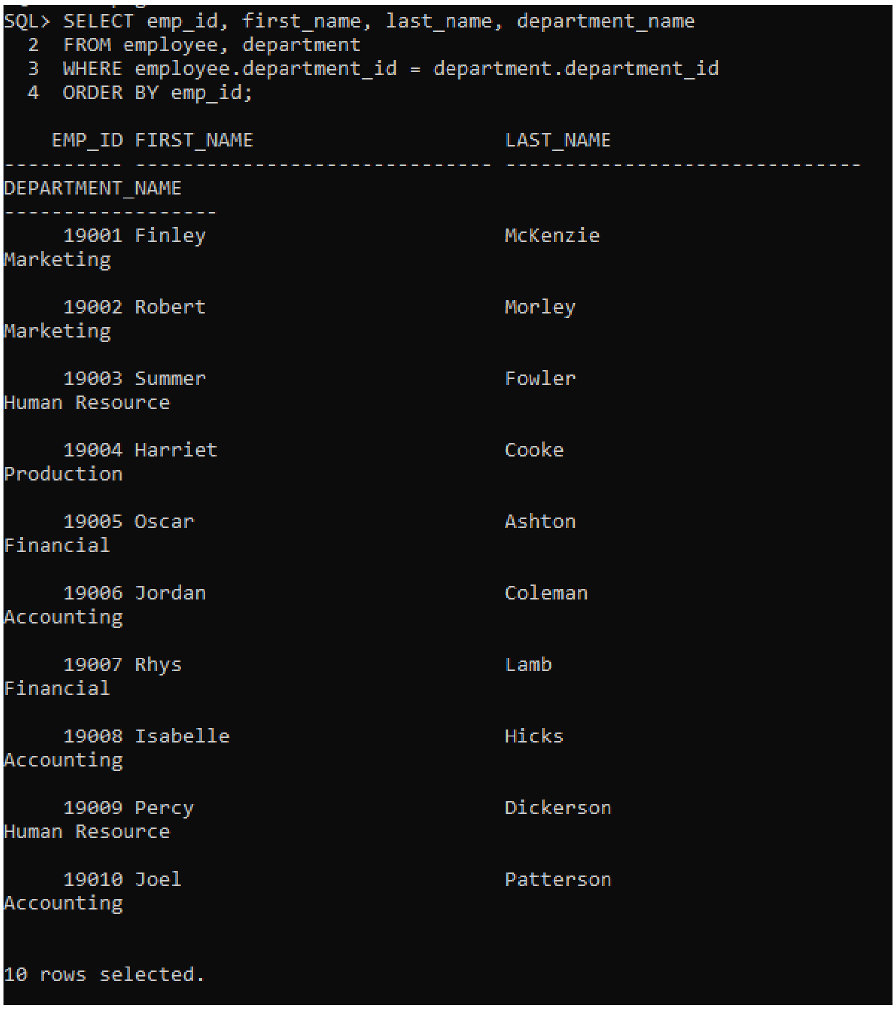
```
INSERT INTO application VALUES

(4, 19007, 002, TO_DATE('04/26/2014', 'MM/DD/YYYY'), TO_DATE('04/28/2014',
'MM/DD/YYYY'), 'Not feeling well', 'Rejected', 58004);


INSERT INTO application VALUES

(5, 19002, 009, TO_DATE('11/13/2015', 'MM/DD/YYYY'), TO_DATE('12/3/2015',
'MM/DD/YYYY'), 'Suspected infection', 'Pending', 58001);


INSERT INTO application VALUES

(6, 19005, 003, TO_DATE('05/28/2010', 'MM/DD/YYYY') , TO_DATE('06/3/2010',
'MM/DD/YYYY'), '', 'Pending', 58004);



--- insterting records into LEAVELOG
INSERT INTO leavelog VALUES
(20001, 20, 58001,19001);


INSERT INTO leavelog VALUES
(20002, 12, 58002, 19006);


INSERT INTO leavelog VALUES
(20003, 21,58005, 19003);


INSERT INTO leavelog VALUES
(20004, 06, 58004, 19005);


INSERT INTO leavelog VALUES
(20005, 24,58005, 19009);



COMMIT;
```

7.0 Individual Assessment (Tai Jia Wei, 1806718)

7.1 Queries

1. The operation displays the list of employees and the department they belong to. This help system user to trace the name of employee and their respective department.

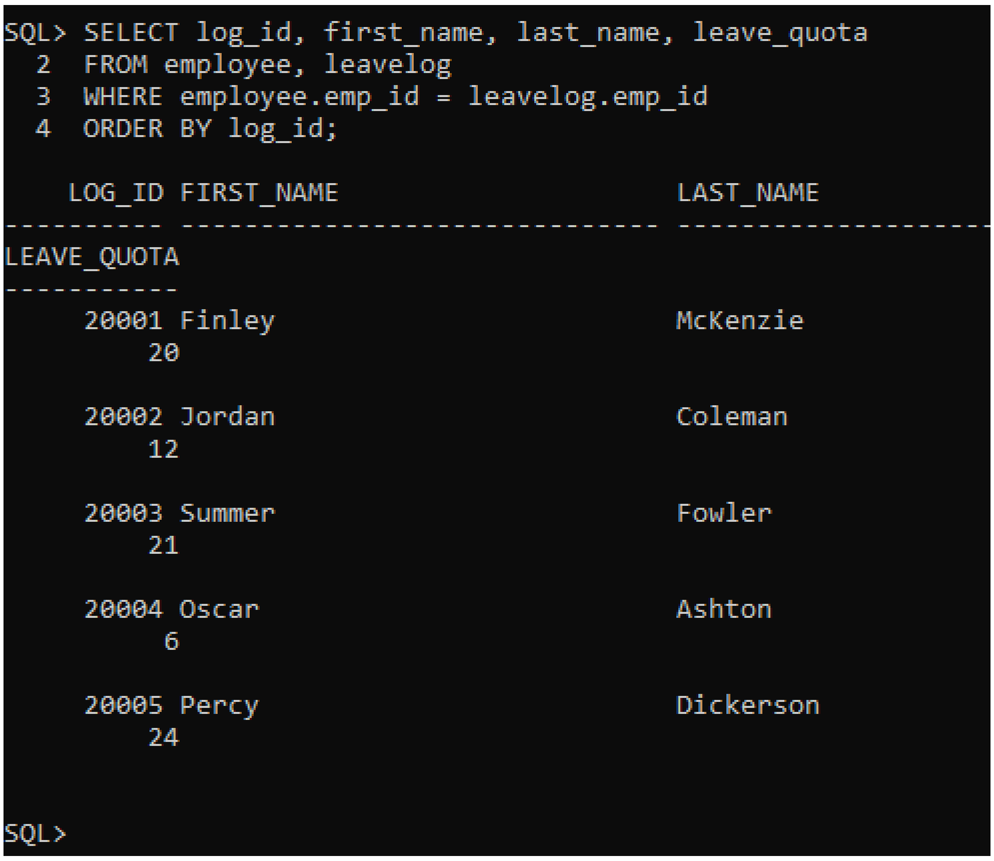
   SELECT emp_id, first_name, last_name, department_name
   FROM employee, department
   WHERE employee.department_id = department.department_id
   ORDER BY emp_id;



```
SQL> SELECT emp_id, first_name, last_name, department_name
  2  FROM employee, department
  3  WHERE employee.department_id = department.department_id
  4  ORDER BY emp_id;

    EMP_ID FIRST_NAME                     LAST_NAME
---------- ------------------------------ ------------------------------
DEPARTMENT_NAME
------------------
     19001 Finley                         McKenzie
Marketing

     19002 Robert                         Morley
Marketing

     19003 Summer                         Fowler
Human Resource

     19004 Harriet                        Cooke
Production

     19005 Oscar                          Ashton
Financial

     19006 Jordan                         Coleman
Accounting

     19007 Rhys                           Lamb
Financial

     19008 Isabelle                       Hicks
Accounting

     19009 Percy                          Dickerson
Human Resource

     19010 Joel                           Patterson
Accounting


10 rows selected.
```

2. The operation shows the amount of leave quota left according to employee. This helps them to keep track of their remaining quota throughout the year.

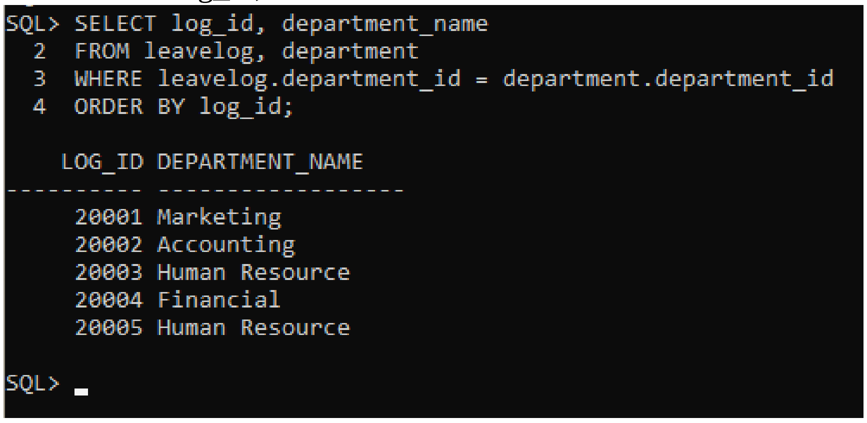
   SELECT log_id, first_name, last_name, leave_quota
   FROM employee, leavelog
   WHERE employee.emp_id = leavelog.emp_id
   ORDER BY log_id;



```
SQL> SELECT log_id, first_name, last_name, leave_quota
  2  FROM employee, leavelog
  3  WHERE employee.emp_id = leavelog.emp_id
  4  ORDER BY log_id;

    LOG_ID FIRST_NAME                     LAST_NAME
---------- ------------------------------ --------------------
LEAVE_QUOTA
-----------
     20001 Finley                         McKenzie
         20

     20002 Jordan                         Coleman
         12

     20003 Summer                         Fowler
         21

     20004 Oscar                          Ashton
          6

     20005 Percy                          Dickerson
         24


SQL>
```

3. The operation show which leavelog is currently handled by which department. This helps the system user to look for departments easily for any occasion.

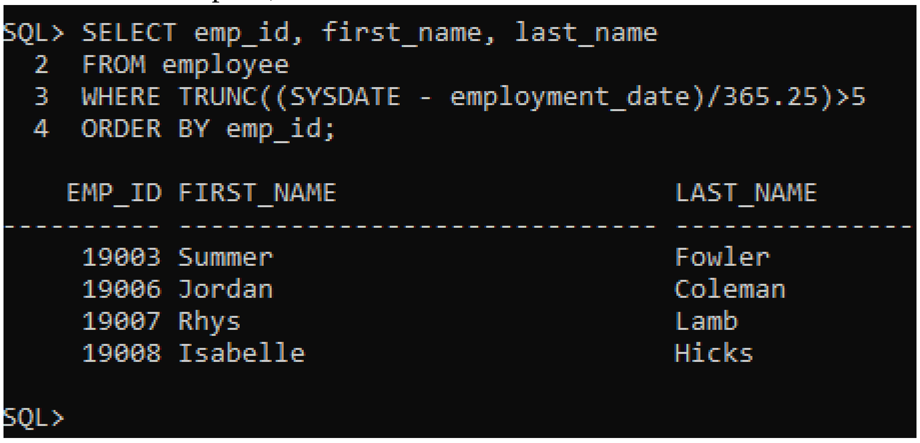
   SELECT log_id, department_name
   FROM leavelog, department
   WHERE leavelog.department_id = department.department_id
   ORDER BY log_id;



```
SQL> SELECT log_id, department_name
  2  FROM leavelog, department
  3  WHERE leavelog.department_id = department.department_id
  4  ORDER BY log_id;

    LOG_ID DEPARTMENT_NAME
---------- ------------------
     20001 Marketing
     20002 Accounting
     20003 Human Resource
     20004 Financial
     20005 Human Resource

SQL>
```

4. The operation show the list of employee who have work for more than 5 years. This is able to help user to find the name of senior employee who have work for more than half a decade.

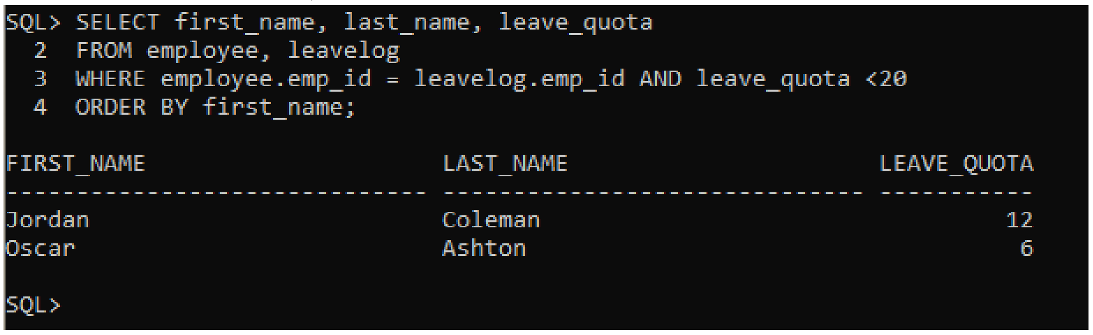
   SELECT emp_id, first_name, last_name
   FROM employee
   WHERE TRUNC((SYSDATE - employment_date)/365.25)>5
   ORDER BY emp_id;



```
SQL> SELECT emp_id, first_name, last_name
  2  FROM employee
  3  WHERE TRUNC((SYSDATE - employment_date)/365.25)>5
  4  ORDER BY emp_id;

    EMP_ID FIRST_NAME                     LAST_NAME
---------- ------------------------------ ---------------
     19003 Summer                         Fowler
     19006 Jordan                         Coleman
     19007 Rhys                           Lamb
     19008 Isabelle                       Hicks

SQL>
```

5. The following query show the list of Leave Log where the quota is less than 20. This helps the user to monitor their application behaviour and apply leave wisely.

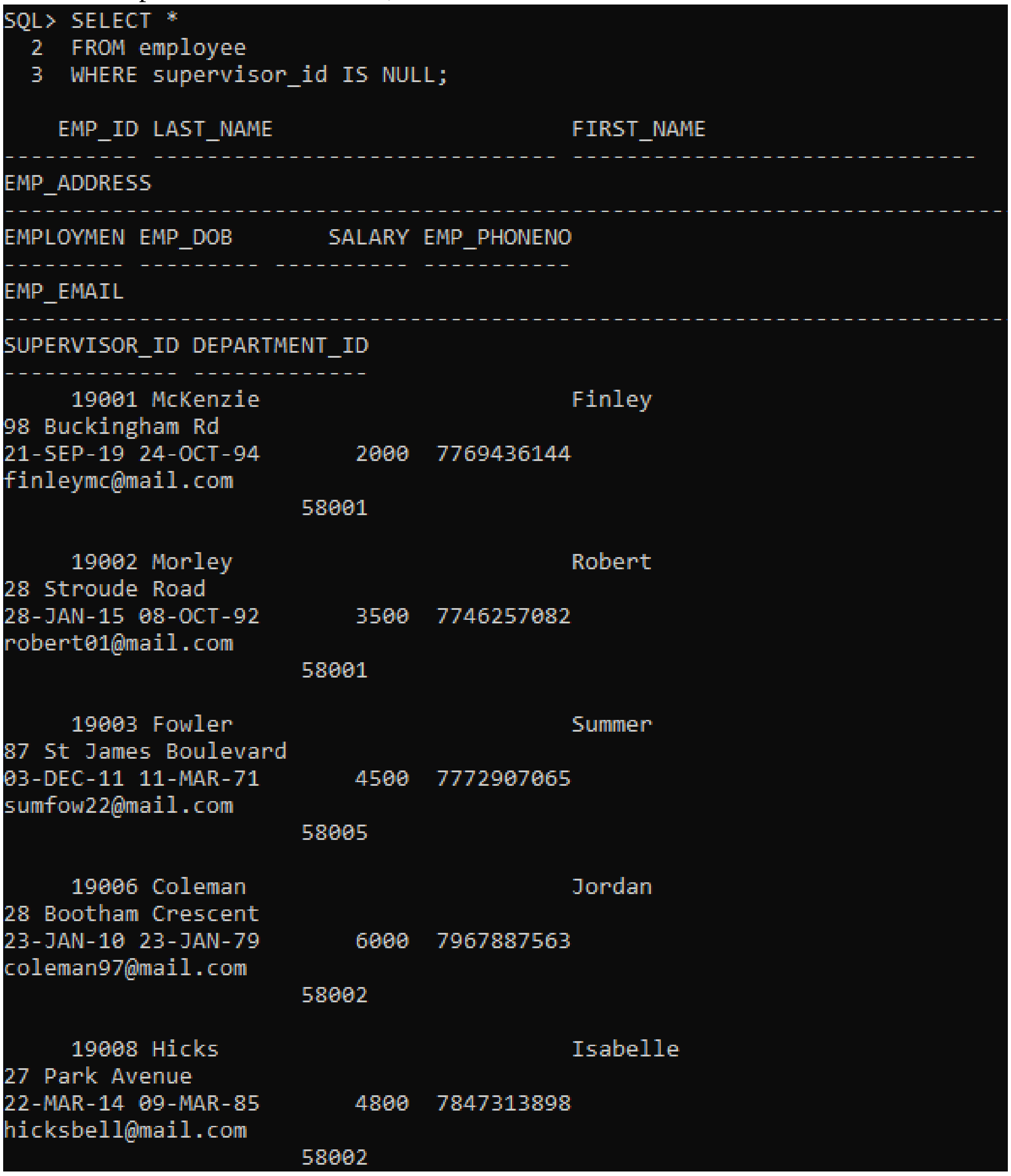
   SELECT first_name, last_name, leave_quota
   FROM employee, leavelog
   WHERE employee.emp_id = leavelog.emp_id AND leave_quota <20
   ORDER BY first_name;



```
SQL> SELECT first_name, last_name, leave_quota
  2  FROM employee, leavelog
  3  WHERE employee.emp_id = leavelog.emp_id AND leave_quota <20
  4  ORDER BY first_name;

FIRST_NAME                     LAST_NAME                      LEAVE_QUOTA
------------------------------ ------------------------------ -----------
Jordan                         Coleman                                 12
Oscar                          Ashton                                   6

SQL>
```

6. The following query show the list of employees who are not under any supervisor. This provide an assist for management or user to conveniently assign them to respective supervisor or even promote them to supervisor.

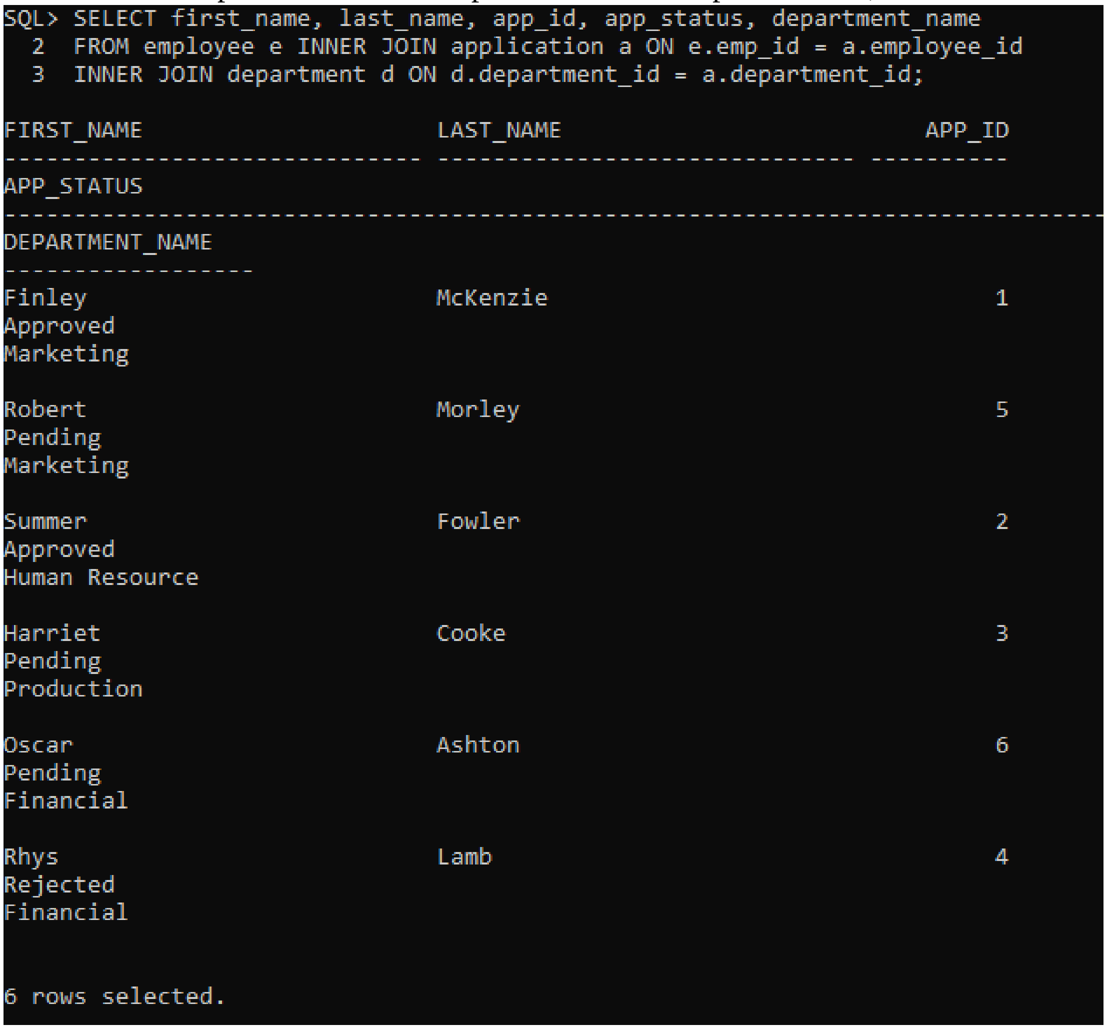
   SELECT *
   FROM employee
   WHERE supervisor_id IS NULL;



```
SQL> SELECT *
  2  FROM employee
  3  WHERE supervisor_id IS NULL;

    EMP_ID LAST_NAME                      FIRST_NAME
---------- ------------------------------ ------------------------------
EMP_ADDRESS
--------------------------------------------------------------------------
EMPLOYMEN EMP_DOB       SALARY EMP_PHONENO
--------- --------- ---------- -----------
EMP_EMAIL
--------------------------------------------------------------------------
SUPERVISOR_ID DEPARTMENT_ID
------------- -------------
     19001 McKenzie                       Finley
98 Buckingham Rd
21-SEP-19 24-OCT-94       2000  7769436144
finleymc@mail.com
                      58001

     19002 Morley                         Robert
28 Stroude Road
28-JAN-15 08-OCT-92       3500  7746257082
robert01@mail.com
                      58001

     19003 Fowler                         Summer
87 St James Boulevard
03-DEC-11 11-MAR-71       4500  7772907065
sumfow22@mail.com
                      58005

     19006 Coleman                        Jordan
28 Bootham Crescent
23-JAN-10 23-JAN-79       6000  7967887563
coleman97@mail.com
                      58002

     19008 Hicks                          Isabelle
27 Park Avenue
22-MAR-14 09-MAR-85       4800  7847313898
hicksbell@mail.com
                      58002
```

7. The query shows the employee name and their leave status which corresponds to their department name. User will able to easily track their application status with ease.

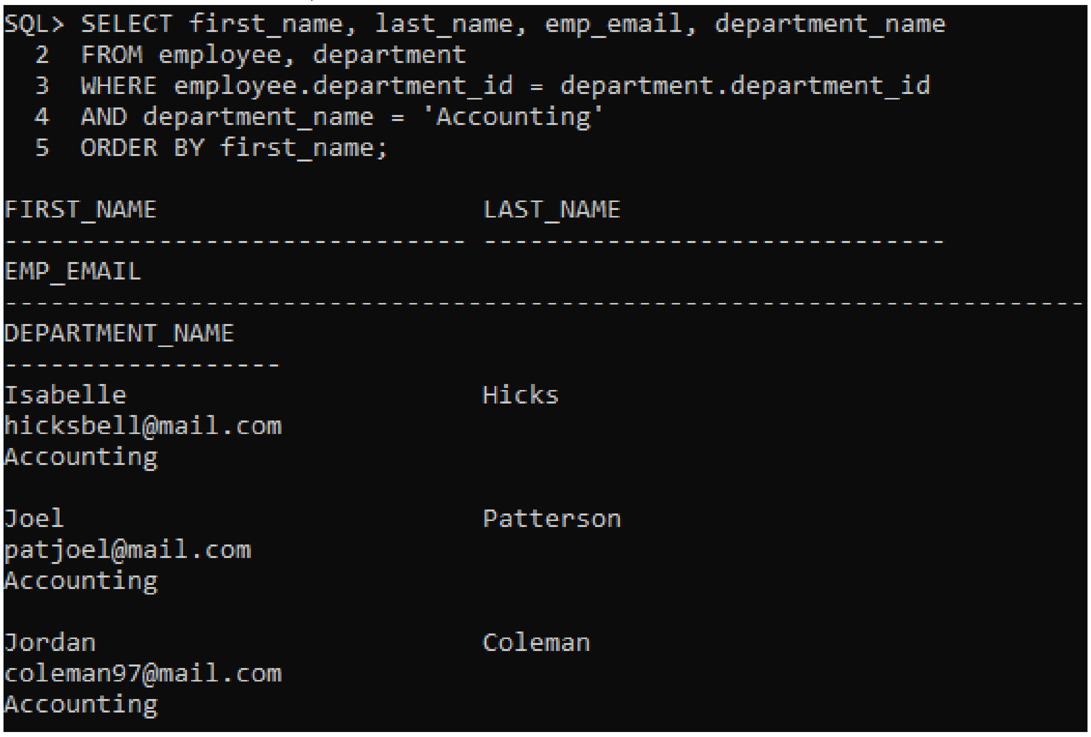
   SELECT first_name, last_name, app_id, app_status, department_name
   FROM employee e INNER JOIN application a ON e.emp_id = a.employee_id
   INNER JOIN department d ON d.department_id = a.department_id;



```
SQL> SELECT first_name, last_name, app_id, app_status, department_name
  2  FROM employee e INNER JOIN application a ON e.emp_id = a.employee_id
  3  INNER JOIN department d ON d.department_id = a.department_id;

FIRST_NAME                     LAST_NAME                          APP_ID
------------------------------ ------------------------------ ----------
APP_STATUS
--------------------------------------------------------------------------------
DEPARTMENT_NAME
------------------
Finley                         McKenzie                                1
Approved
Marketing

Robert                         Morley                                  5
Pending
Marketing

Summer                         Fowler                                  2
Approved
Human Resource

Harriet                        Cooke                                   3
Pending
Production

Oscar                          Ashton                                  6
Pending
Financial

Rhys                           Lamb                                    4
Rejected
Financial


6 rows selected.
```

8. The following query show the list of employees which is under Accounting department. When the user wants to look for employee who is under Accounting department or any other department. They can use this query to look up for them.

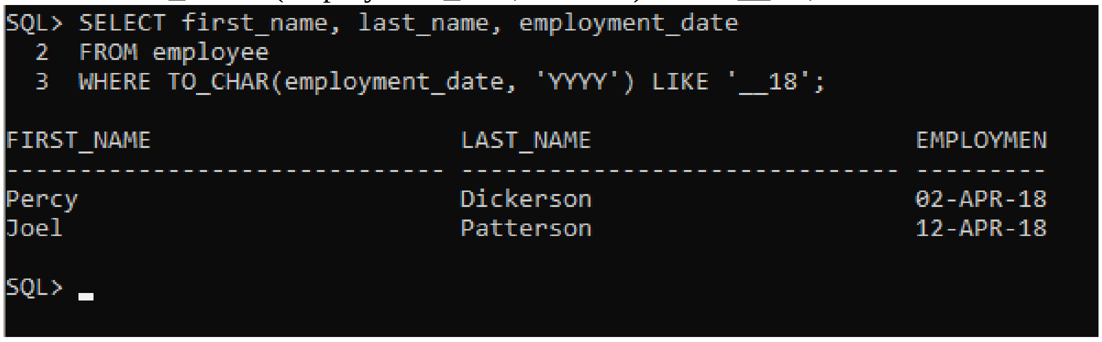
   SELECT first_name, last_name, emp_email, department_name
   FROM employee, department
   WHERE employee.department_id = department.department_id
   AND department_name = 'Accounting'
   ORDER BY first_name;



```
SQL> SELECT first_name, last_name, emp_email, department_name
  2  FROM employee, department
  3  WHERE employee.department_id = department.department_id
  4  AND department_name = 'Accounting'
  5  ORDER BY first_name;

FIRST_NAME                     LAST_NAME
------------------------------ ------------------------------
EMP_EMAIL
----------------------------------------------------------------------
DEPARTMENT_NAME
------------------
Isabelle                       Hicks
hicksbell@mail.com
Accounting

Joel                           Patterson
patjoel@mail.com
Accounting

Jordan                         Coleman
coleman97@mail.com
Accounting
```

9. This query show the list of employee who are hired in the year 2018. User can use this query to look up for employee according to the year they're hired.

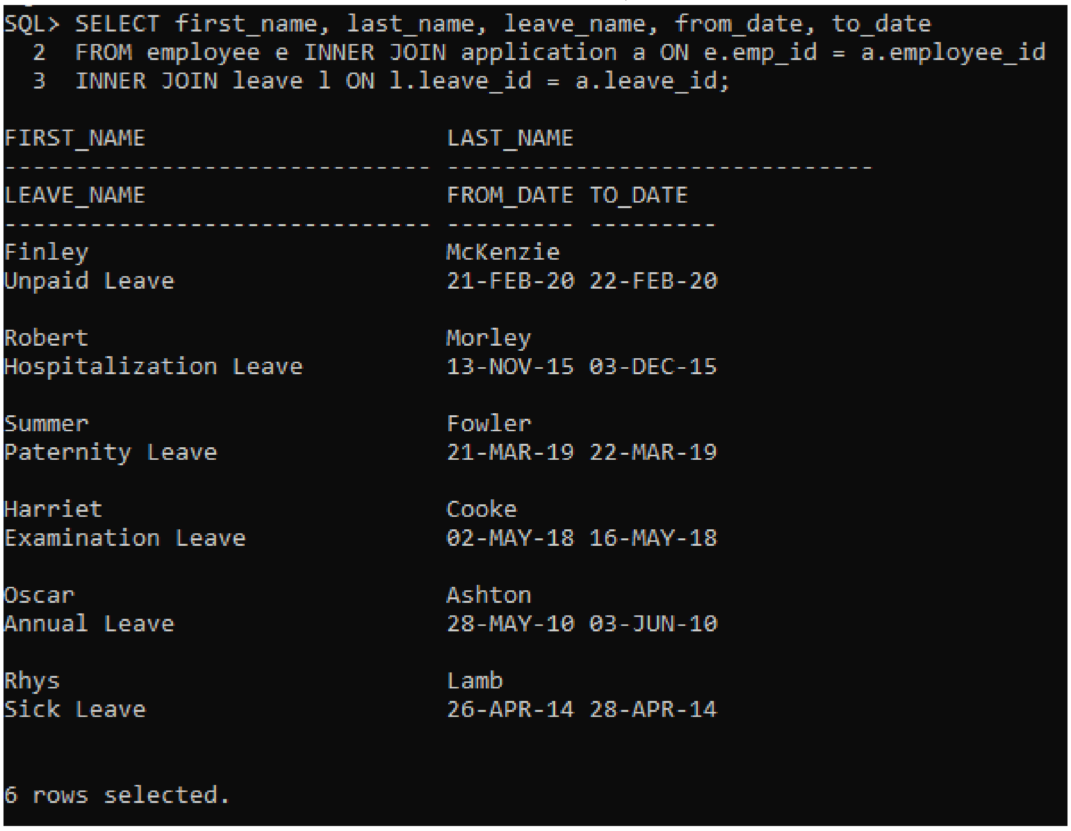
   SELECT first_name, last_name, employment_date
   FROM employee
   WHERE TO_CHAR(employment_date, 'YYYY') LIKE '__18';



```
SQL> SELECT first_name, last_name, employment_date
  2  FROM employee
  3  WHERE TO_CHAR(employment_date, 'YYYY') LIKE '__18';

FIRST_NAME                     LAST_NAME                      EMPLOYMEN
------------------------------ ------------------------------ ---------
Percy                          Dickerson                      02-APR-18
Joel                           Patterson                      12-APR-18

SQL>
```

10. The query shows the list of employees which are not available according to the date listed. Users can use this query to look up for employees which will be absent from date to date and able to schedule work easily.

    SELECT first_name, last_name, leave_name, from_date, to_date
    FROM employee e INNER JOIN application a ON e.emp_id = a.employee_id
    INNER JOIN leave l ON l.leave_id = a.leave_id;

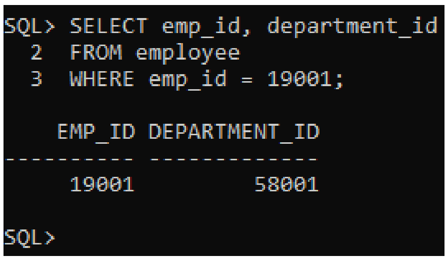


```
SQL> SELECT first_name, last_name, leave_name, from_date, to_date
  2  FROM employee e INNER JOIN application a ON e.emp_id = a.employee_id
  3  INNER JOIN leave l ON l.leave_id = a.leave_id;

FIRST_NAME                     LAST_NAME
------------------------------ ------------------------------
LEAVE_NAME                     FROM_DATE TO_DATE
------------------------------ --------- ---------
Finley                         McKenzie
Unpaid Leave                   21-FEB-20 22-FEB-20

Robert                         Morley
Hospitalization Leave          13-NOV-15 03-DEC-15

Summer                         Fowler
Paternity Leave                21-MAR-19 22-MAR-19

Harriet                        Cooke
Examination Leave              02-MAY-18 16-MAY-18

Oscar                          Ashton
Annual Leave                   28-MAY-10 03-JUN-10

Rhys                           Lamb
Sick Leave                     26-APR-14 28-APR-14


6 rows selected.
```

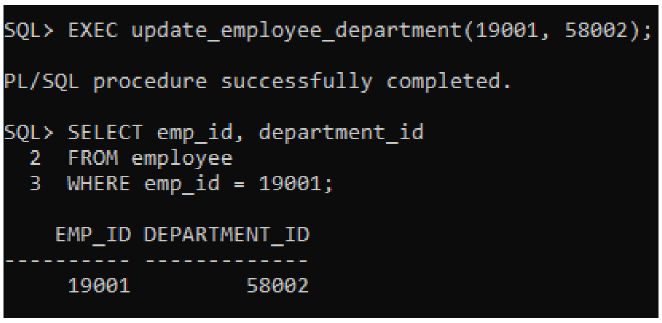
7.2 Stored Procedure

1. This procedure is use to update the department a particular employee belongs to. This procedure can be used when an employee has moved to another department within the company.

   ```
   CREATE OR REPLACE PROCEDURE update_employee_department
   (cur_emp_id IN NUMBER, cur_department_id IN NUMBER)
   IS
   BEGIN
   UPDATE employee
   SET department_id = cur_department_id
   WHERE emp_id = cur_emp_id;
   COMMIT;
   END;
   /
   ```



```
SQL> SELECT emp_id, department_id
  2  FROM employee
  3  WHERE emp_id = 19001;

    EMP_ID DEPARTMENT_ID
---------- -------------
     19001         58001

SQL>
```

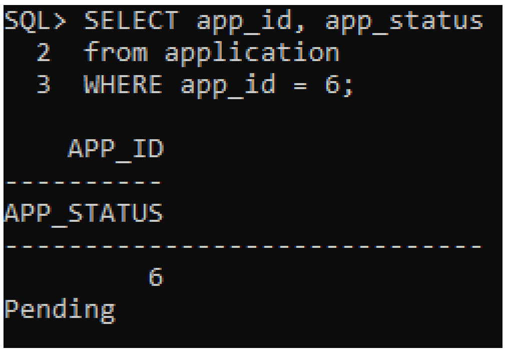


```
SQL> EXEC update_employee_department(19001, 58002);

PL/SQL procedure successfully completed.

SQL> SELECT emp_id, department_id
  2  FROM employee
  3  WHERE emp_id = 19001;

    EMP_ID DEPARTMENT_ID
---------- -------------
     19001         58002
```

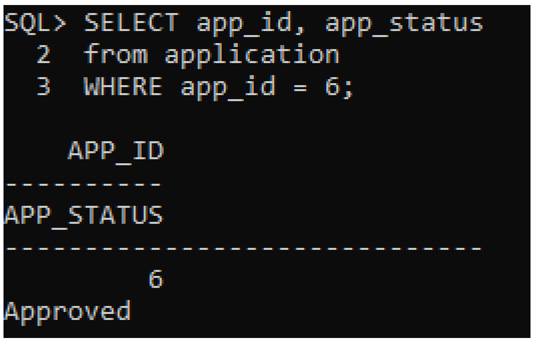
Before After

2. The following procedure is use to change the application status according to the application id. Users can use this procedure to update the status of the application based on the outcome.

   ```
   CREATE OR REPLACE PROCEDURE update_application_status
   (cur_app_id IN NUMBER, cur_app_status IN VARCHAR2)
   IS
   BEGIN
   UPDATE application
   SET app_status = cur_app_status
   WHERE app_id = cur_app_id;
   COMMIT;
   END;
   /
   ```



```
SQL> SELECT app_id, app_status
  2  from application
  3  WHERE app_id = 6;

    APP_ID
----------
APP_STATUS
------------------------------
         6
Pending
```

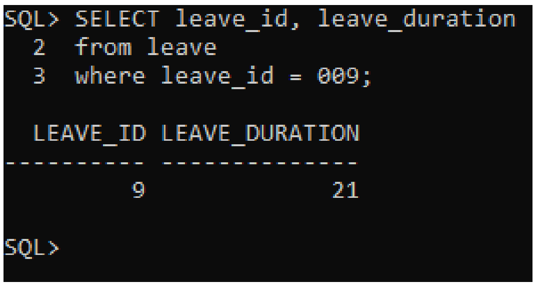


```
SQL> SELECT app_id, app_status
  2  from application
  3  WHERE app_id = 6;

    APP_ID
----------
APP_STATUS
------------------------------
         6
Approved
```

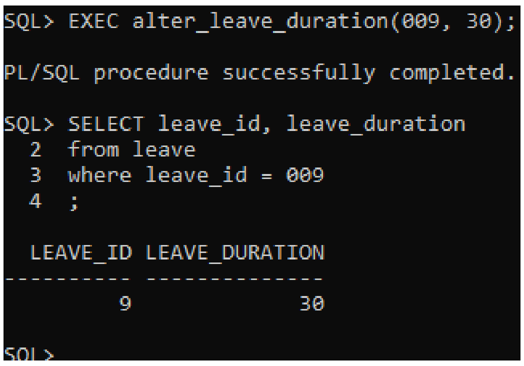
Before After

3. The following procedure is to change the duration of a specific leave type. Users can use this to change the leave duration of their company according to their policy change.

   CREATE OR REPLACE PROCEDURE alter_leave_duration
   (cur_leave_id IN NUMBER, cur_leave_duration IN NUMBER)
   IS
   BEGIN
   UPDATE leave
   SET leave_duration = cur_leave_duration
   WHERE leave_id = cur_leave_id;
   COMMIT;
   END;
   /



```
SQL> SELECT leave_id, leave_duration
  2  from leave
  3  where leave_id = 009;

  LEAVE_ID LEAVE_DURATION
---------- --------------
         9             21

SQL>
```

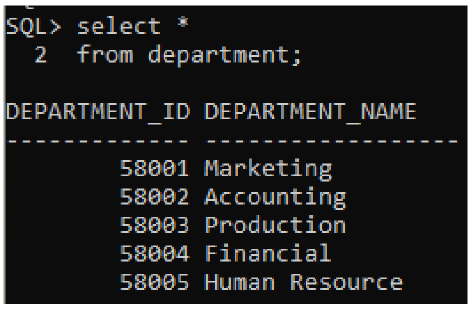


```
SQL> EXEC alter_leave_duration(009, 30);

PL/SQL procedure successfully completed.

SQL> SELECT leave_id, leave_duration
  2  from leave
  3  where leave_id = 009
  4  ;

  LEAVE_ID LEAVE_DURATION
---------- --------------
         9             30

SQL>
```

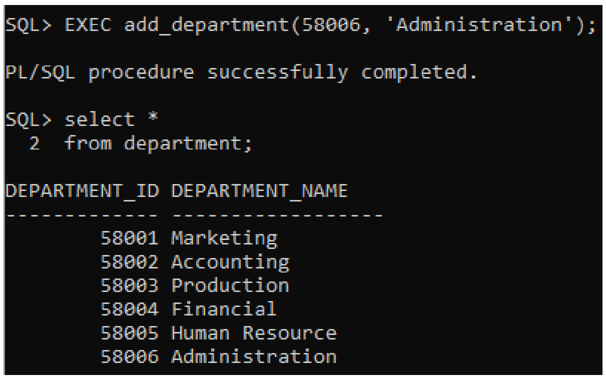
Before After

4. The procedure is the addition of new department into the system. Users can use this to create a new department in the current system when a new department is formed in the company.

   CREATE OR REPLACE PROCEDURE add_department
   (cur_department_id IN NUMBER, cur_department_name IN VARCHAR2)
   IS
   BEGIN
   INSERT INTO department
   VALUES(cur_department_id, cur_department_name);
   COMMIT;
   END;
   /



```
SQL> select *
  2  from department;

DEPARTMENT_ID DEPARTMENT_NAME
------------- ------------------
        58001 Marketing
        58002 Accounting
        58003 Production
        58004 Financial
        58005 Human Resource
```

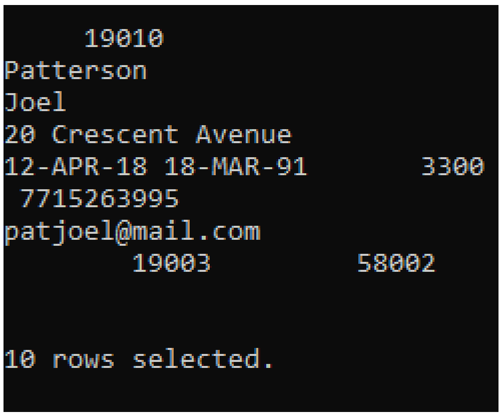


```
SQL> EXEC add_department(58006, 'Administration');

PL/SQL procedure successfully completed.

SQL> select *
  2  from department;

DEPARTMENT_ID DEPARTMENT_NAME
------------- ------------------
        58001 Marketing
        58002 Accounting
        58003 Production
        58004 Financial
        58005 Human Resource
        58006 Administration
```

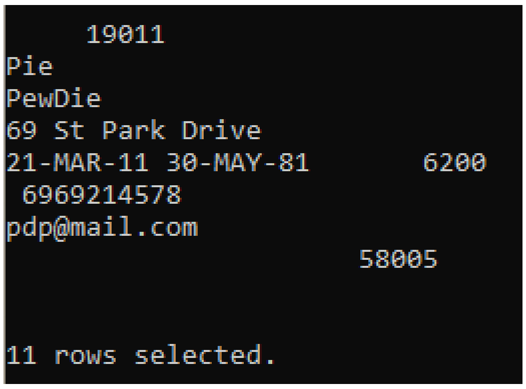
Before After

5. This procedure is for adding new employee to company. Users can use the following procedure to insert relevant employee information into the system.

```
CREATE OR REPLACE PROCEDURE add_new_employee
(cur_emp_id IN NUMBER, cur_last_name IN VARCHAR2, cur_first_name IN
VARCHAR2, cur_emp_address IN VARCHAR2,
cur_employment_date IN DATE, cur_phoneno IN NUMBER, cur_email IN
VARCHAR2, cur_supervisor_id IN NUMBER, cur_department_id IN NUMBER)
IS
BEGIN
INSERT INTO employee
VALUES (cur_emp_id, cur_last_name, cur_first_name, cur_emp_address,
cur_employment_date, cur_phoneno, cur_email, cur_supervisor_id,
cur_department_id);
COMMIT;
END;
/
```



```
     19010
Patterson
Joel
20 Crescent Avenue
12-APR-18 18-MAR-91       3300
 7715263995
patjoel@mail.com
        19003        58002


10 rows selected.
```



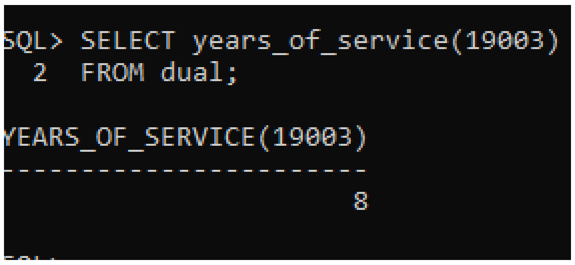
```
     19011
Pie
PewDie
69 St Park Drive
21-MAR-11 30-MAY-81       6200
 6969214578
pdp@mail.com
                      58005


11 rows selected.
```

Before After

7.3 Functions

1. The following function show the number of year an employee has been working in the company. User can use this function to keep track of their service year through their time in the company.

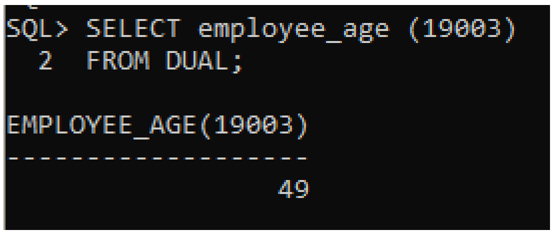
   ```
   CREATE OR REPLACE FUNCTION years_of_service
   (cur_emp_id IN NUMBER) RETURN NUMBER
   IS
   empyear NUMBER;
   BEGIN
   SELECT TRUNC((SYSDATE-employment_date)/365.25) INTO empyear
   FROM employee
   WHERE employee.emp_id = cur_emp_id;
   RETURN empyear;
   END;
   /
   ```



```
SQL> SELECT years_of_service(19003)
  2  FROM dual;

YEARS_OF_SERVICE(19003)
-----------------------
                      8
```

2. The following function shows the age of an employee in the company according to the ID. User can use this to lookup at the age of employee and assign the suitable task to them.

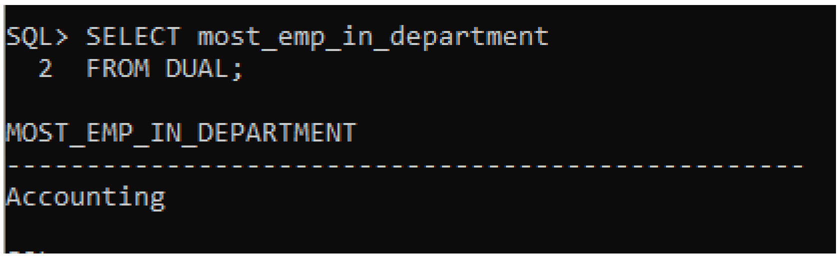
   ```
   CREATE OR REPLACE FUNCTION employee_age
   (cur_emp_id IN NUMBER) RETURN NUMBER
   IS
   empage NUMBER;
   BEGIN
   SELECT TRUNC((SYSDATE-emp_dob)/365.25) INTO empage
   FROM employee
   WHERE employee.emp_id = cur_emp_id;
   RETURN empage;
   END;
   /
   ```



```
SQL> SELECT employee_age (19003)
  2  FROM DUAL;

EMPLOYEE_AGE(19003)
-------------------
                 49
```

3. The following function show the greatest number of employees in a department. User can use this function to lookup for number of employee and evaluate whether they have enough manpower in the respective department.

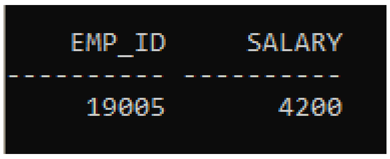
```
CREATE OR REPLACE FUNCTION most_emp_in_department
RETURN VARCHAR2
IS
deptname VARCHAR2(20);
BEGIN
SELECT department_name INTO deptname
FROM department
WHERE department_id IN
(SELECT department_id
FROM employee
GROUP BY department_id
HAVING COUNT(*) IN
(SELECT MAX(mycount)
FROM
(SELECT COUNT(*) mycount
FROM employee
GROUP BY department_id) a));
RETURN deptname;
END;
/
```



```
SQL> SELECT most_emp_in_department
  2  FROM DUAL;

MOST_EMP_IN_DEPARTMENT
--------------------------------------------------
Accounting
```

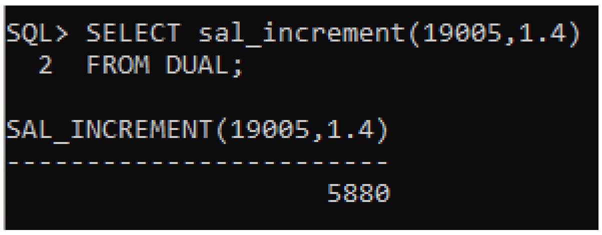
4. The following function show the salary of an employee after their increment. The output of this function will not affect the existing salary in the database. User can use this function to alter the increment percentage based on their salary before actually doing it.

```
CREATE OR REPLACE FUNCTION sal_increment
(cur_emp_id IN NUMBER, increment IN NUMBER) RETURN NUMBER
IS
newsal NUMBER;
tmp NUMBER;
BEGIN
SELECT salary INTO tmp
FROM employee
WHERE emp_id = cur_emp_id;
newsal := (tmp*increment);
RETURN newsal;
END;
/
```



```
    EMP_ID     SALARY
---------- ----------
     19005       4200
```

Before

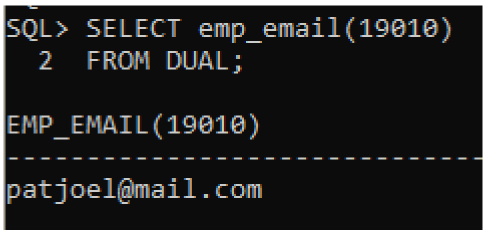


```
SQL> SELECT sal_increment(19005,1.4)
  2  FROM DUAL;

SAL_INCREMENT(19005,1.4)
------------------------
                    5880
```

After

5. The following function can retrieve the email address of an employee based on the input of employee id. User can use this function to lookup for an employee's contact information.

```
CREATE OR REPLACE FUNCTION emp_email
(cur_emp_id IN NUMBER) RETURN VARCHAR2
IS
email VARCHAR2(30);
BEGIN
SELECT emp_email INTO email
FROM employee
WHERE emp_id = cur_emp_id;
RETURN email;
END;
/
```



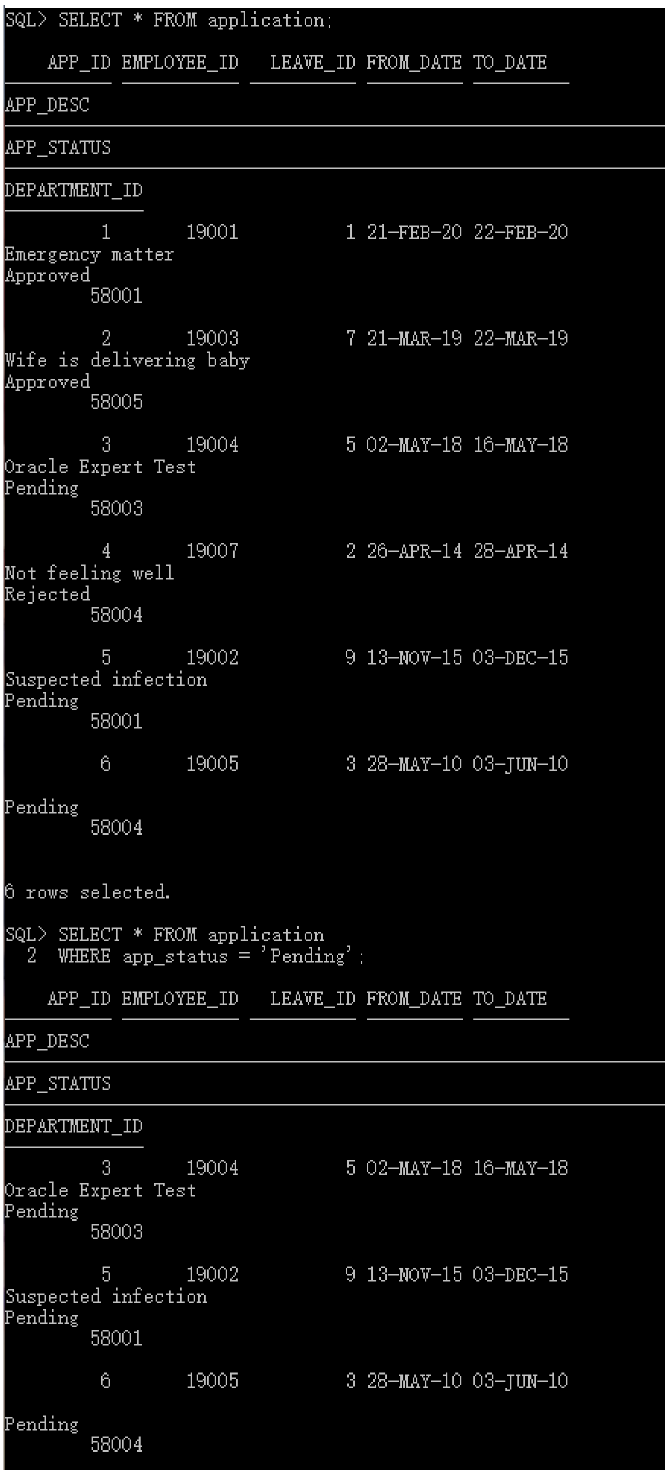
```
SQL> SELECT emp_email(19010)
  2  FROM DUAL;

EMP_EMAIL(19010)
------------------------------
patjoel@mail.com
```

8.0 Individual Assessment (Ling Kheng Yuan, 1703264)

8.1 Queries

1. Check for pending applications
   The manager may use this query to display the pending leave applications.

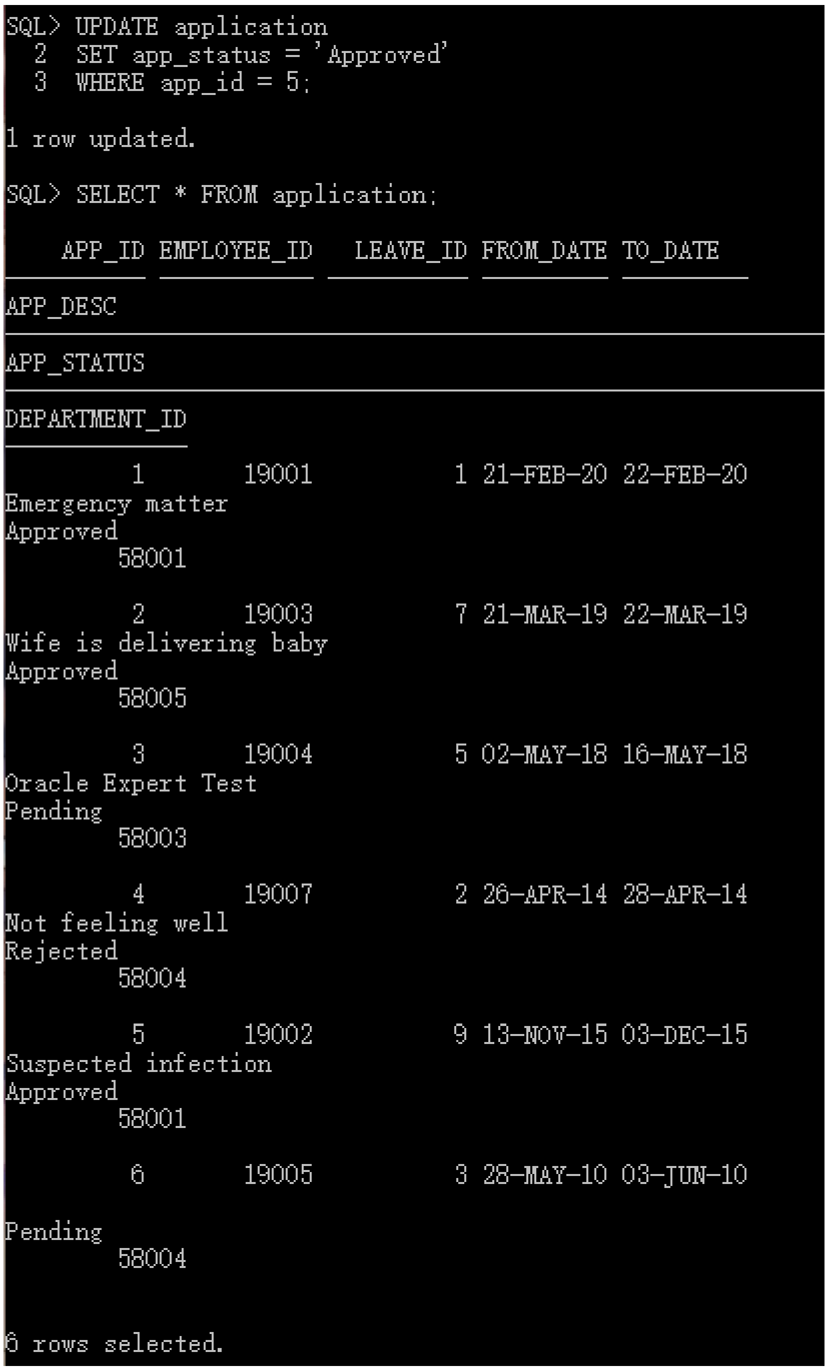
   SELECT * FROM application
   WHERE app_status = 'Pending';



```
SQL> SELECT * FROM application;

    APP_ID EMPLOYEE_ID   LEAVE_ID FROM_DATE TO_DATE
---------- ----------- ---------- --------- ---------
APP_DESC
--------------------------------------------------------------------------------
APP_STATUS
--------------------------------------------------------------------------------
DEPARTMENT_ID
-------------
         1       19001          1 21-FEB-20 22-FEB-20
Emergency matter
Approved
        58001

         2       19003          7 21-MAR-19 22-MAR-19
Wife is delivering baby
Approved
        58005

         3       19004          5 02-MAY-18 16-MAY-18
Oracle Expert Test
Pending
        58003

         4       19007          2 26-APR-14 28-APR-14
Not feeling well
Rejected
        58004

         5       19002          9 13-NOV-15 03-DEC-15
Suspected infection
Pending
        58001

         6       19005          3 28-MAY-10 03-JUN-10

Pending
        58004


6 rows selected.

SQL> SELECT * FROM application
  2  WHERE app_status = 'Pending';

    APP_ID EMPLOYEE_ID   LEAVE_ID FROM_DATE TO_DATE
---------- ----------- ---------- --------- ---------
APP_DESC
--------------------------------------------------------------------------------
APP_STATUS
--------------------------------------------------------------------------------
DEPARTMENT_ID
-------------
         3       19004          5 02-MAY-18 16-MAY-18
Oracle Expert Test
Pending
        58003

         5       19002          9 13-NOV-15 03-DEC-15
Suspected infection
Pending
        58001

         6       19005          3 28-MAY-10 03-JUN-10

Pending
        58004
```

2. Update the pending leave applications
   The manager may use this query to update the pending leave applications.

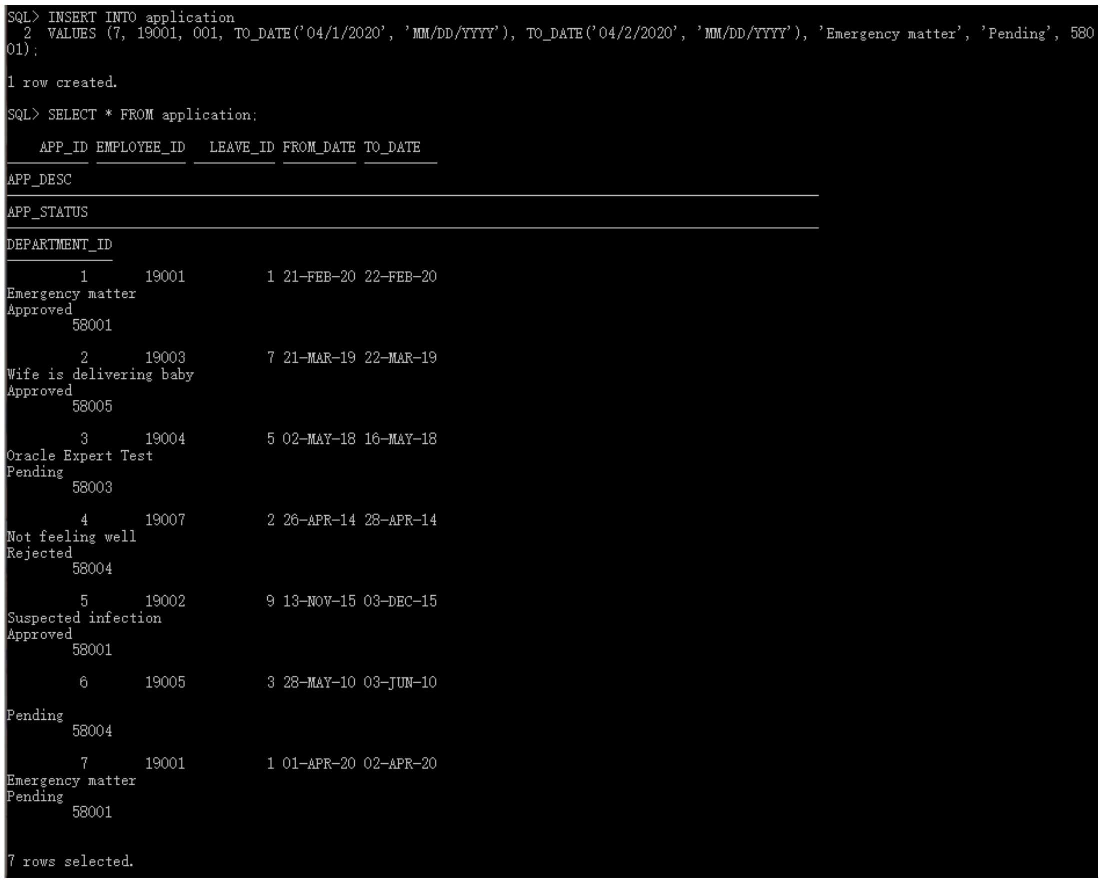
   UPDATE application
   SET app_status = 'Approved'
   WHERE app_id = 5;



```
SQL> UPDATE application
  2  SET app_status = 'Approved'
  3  WHERE app_id = 5;

1 row updated.

SQL> SELECT * FROM application;

    APP_ID EMPLOYEE_ID   LEAVE_ID FROM_DATE TO_DATE
---------- ----------- ---------- --------- ---------
APP_DESC
--------------------------------------------------------------------------------
APP_STATUS
--------------------------------------------------------------------------------
DEPARTMENT_ID
-------------
         1       19001          1 21-FEB-20 22-FEB-20
Emergency matter
Approved
        58001

         2       19003          7 21-MAR-19 22-MAR-19
Wife is delivering baby
Approved
        58005

         3       19004          5 02-MAY-18 16-MAY-18
Oracle Expert Test
Pending
        58003

         4       19007          2 26-APR-14 28-APR-14
Not feeling well
Rejected
        58004

         5       19002          9 13-NOV-15 03-DEC-15
Suspected infection
Approved
        58001

         6       19005          3 28-MAY-10 03-JUN-10

Pending
        58004


6 rows selected.
```

3. Insert into application table
   When the employees want to apply leave, they may use this query to submit their application.

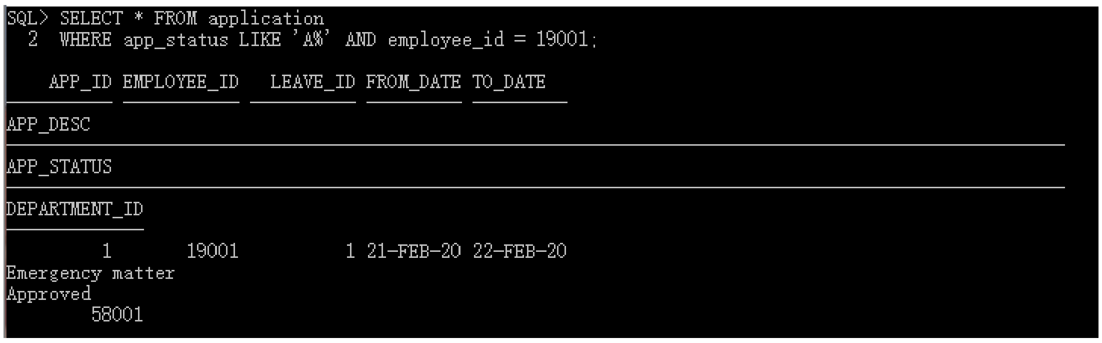
   INSERT INTO application
   VALUES (7, 19001, 001, TO_DATE('04/1/2020', 'MM/DD/YYYY'), TO_DATE('04/2/2020', 'MM/DD/YYYY'), 'Emergency matter', 'Pending', 58001);



```
SQL> INSERT INTO application
  2  VALUES (7, 19001, 001, TO_DATE('04/1/2020', 'MM/DD/YYYY'), TO_DATE('04/2/2020', 'MM/DD/YYYY'), 'Emergency matter', 'Pending', 580
01);

1 row created.

SQL> SELECT * FROM application;

    APP_ID EMPLOYEE_ID   LEAVE_ID FROM_DATE TO_DATE
---------- ----------- ---------- --------- ---------
APP_DESC
--------------------------------------------------------------------------------
APP_STATUS
--------------------------------------------------------------------------------
DEPARTMENT_ID
-------------
         1       19001          1 21-FEB-20 22-FEB-20
Emergency matter
Approved
        58001

         2       19003          7 21-MAR-19 22-MAR-19
Wife is delivering baby
Approved
        58005

         3       19004          5 02-MAY-18 16-MAY-18
Oracle Expert Test
Pending
        58003

         4       19007          2 26-APR-14 28-APR-14
Not feeling well
Rejected
        58004

         5       19002          9 13-NOV-15 03-DEC-15
Suspected infection
Approved
        58001

         6       19005          3 28-MAY-10 03-JUN-10

Pending
        58004

         7       19001          1 01-APR-20 02-APR-20
Emergency matter
Pending
        58001


7 rows selected.
```

4. Check leaves taken by employee_id
   When the manager needs to check leaves taken of a certain employee, he/she may use this query to get the results.

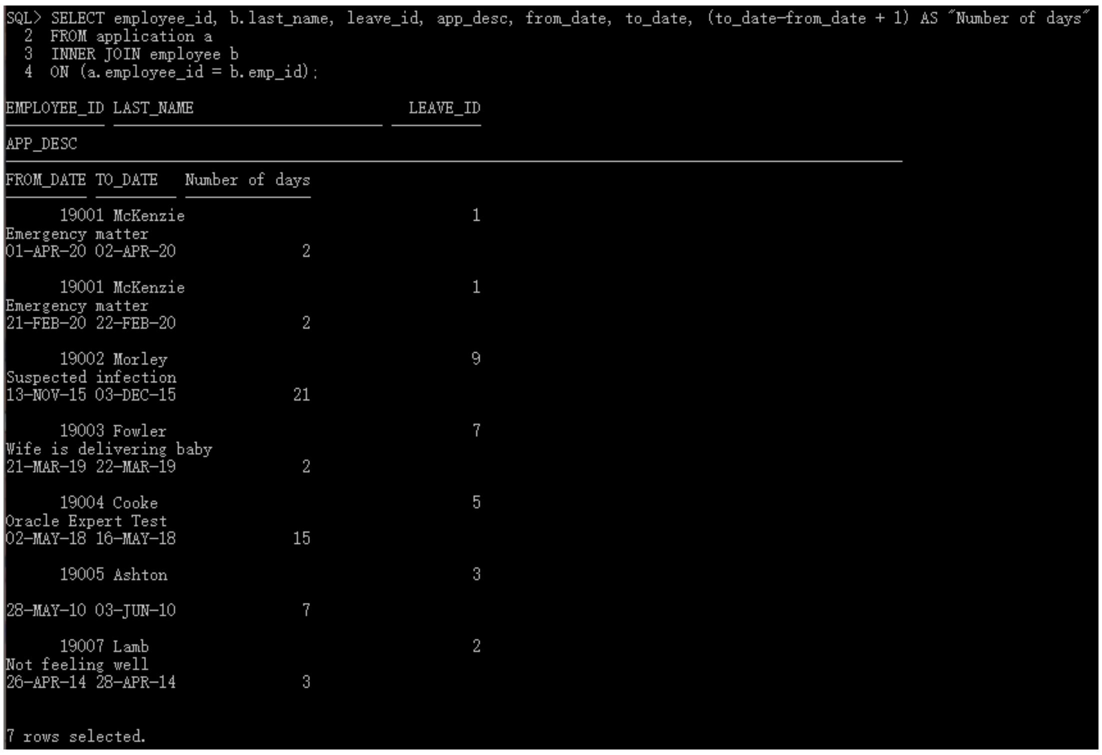
   SELECT * FROM application
   WHERE app_status LIKE 'A%' AND employee_id = 19001;



```
SQL> SELECT * FROM application
  2  WHERE app_status LIKE 'A%' AND employee_id = 19001;

    APP_ID EMPLOYEE_ID   LEAVE_ID FROM_DATE TO_DATE
---------- ----------- ---------- --------- ---------
APP_DESC
--------------------------------------------------------------------------------
APP_STATUS
--------------------------------------------------------------------------------
DEPARTMENT_ID
-------------
         1       19001          1 21-FEB-20 22-FEB-20
Emergency matter
Approved
        58001
```

5. Display number of days of leave
   This query can help manager to get number of days of leave application easily.

   SELECT employee_id, b.last_name, leave_id, app_desc, from_date, to_date,
   (to_date-from_date + 1) AS ”Number of days”
   FROM application a
   INNER JOIN employee b
   ON (a.employee_id = b.emp_id);

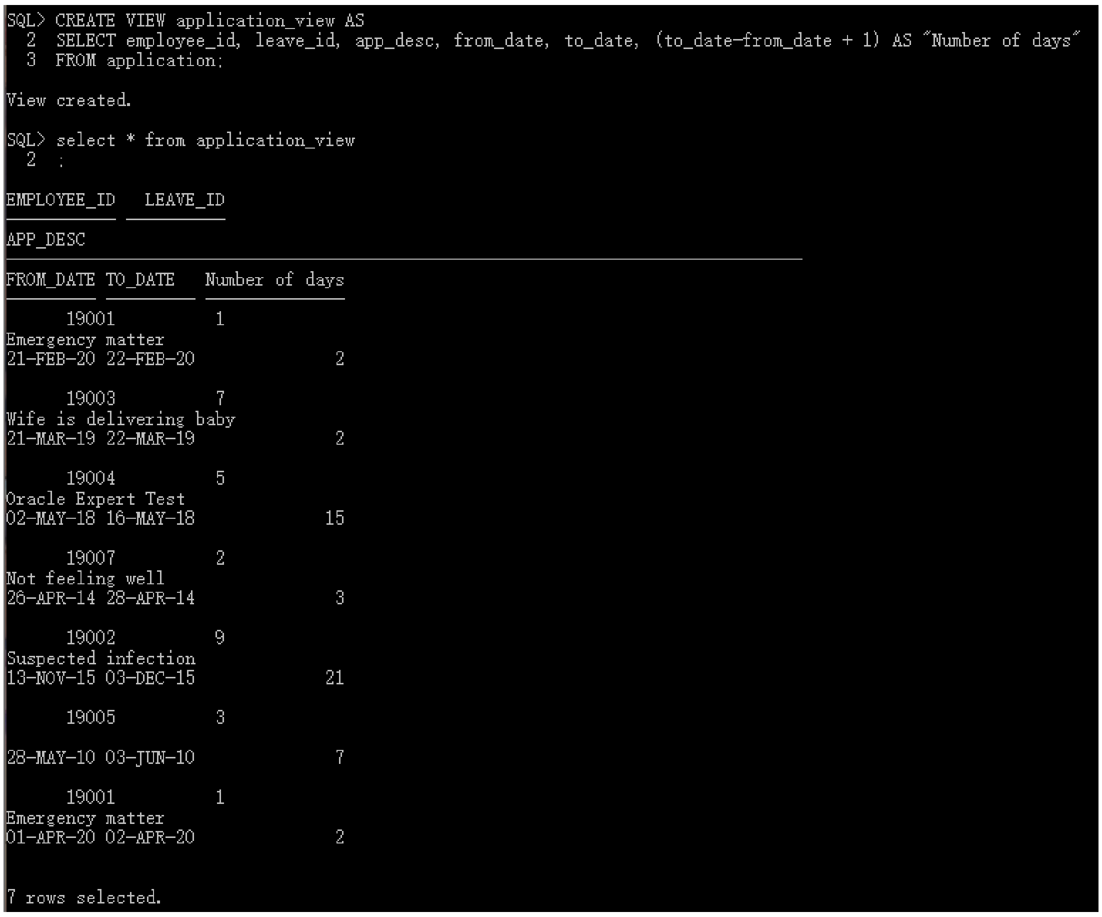


```
SQL> SELECT employee_id, b.last_name, leave_id, app_desc, from_date, to_date, (to_date-from_date + 1) AS "Number of days"
  2  FROM application a
  3  INNER JOIN employee b
  4  ON (a.employee_id = b.emp_id);

EMPLOYEE_ID LAST_NAME                     LEAVE_ID
----------- ------------------------------ ----------
APP_DESC
----------------------------------------------------------------------------------------------------
FROM_DATE TO_DATE   Number of days
--------- --------- --------------
      19001 McKenzie                              1
Emergency matter
01-APR-20 02-APR-20              2

      19001 McKenzie                              1
Emergency matter
21-FEB-20 22-FEB-20              2

      19002 Morley                                9
Suspected infection
13-NOV-15 03-DEC-15             21

      19003 Fowler                                7
Wife is delivering baby
21-MAR-19 22-MAR-19              2

      19004 Cooke                                 5
Oracle Expert Test
02-MAY-18 16-MAY-18             15

      19005 Ashton                                3

28-MAY-10 03-JUN-10              7

      19007 Lamb                                  2
Not feeling well
26-APR-14 28-APR-14              3


7 rows selected.
```

6. Create a view for application table

   This query eases the manager from typing long queries such as [(to_date-from_date + 1) AS ”Number of days”] every time, he/she may create a view and select the view in the future.

   CREATE VIEW application_view AS
   SELECT employee_id, leave_id, app_desc, from_date, to_date, (to_date-from_date + 1) AS ”Number of days”
   FROM application;

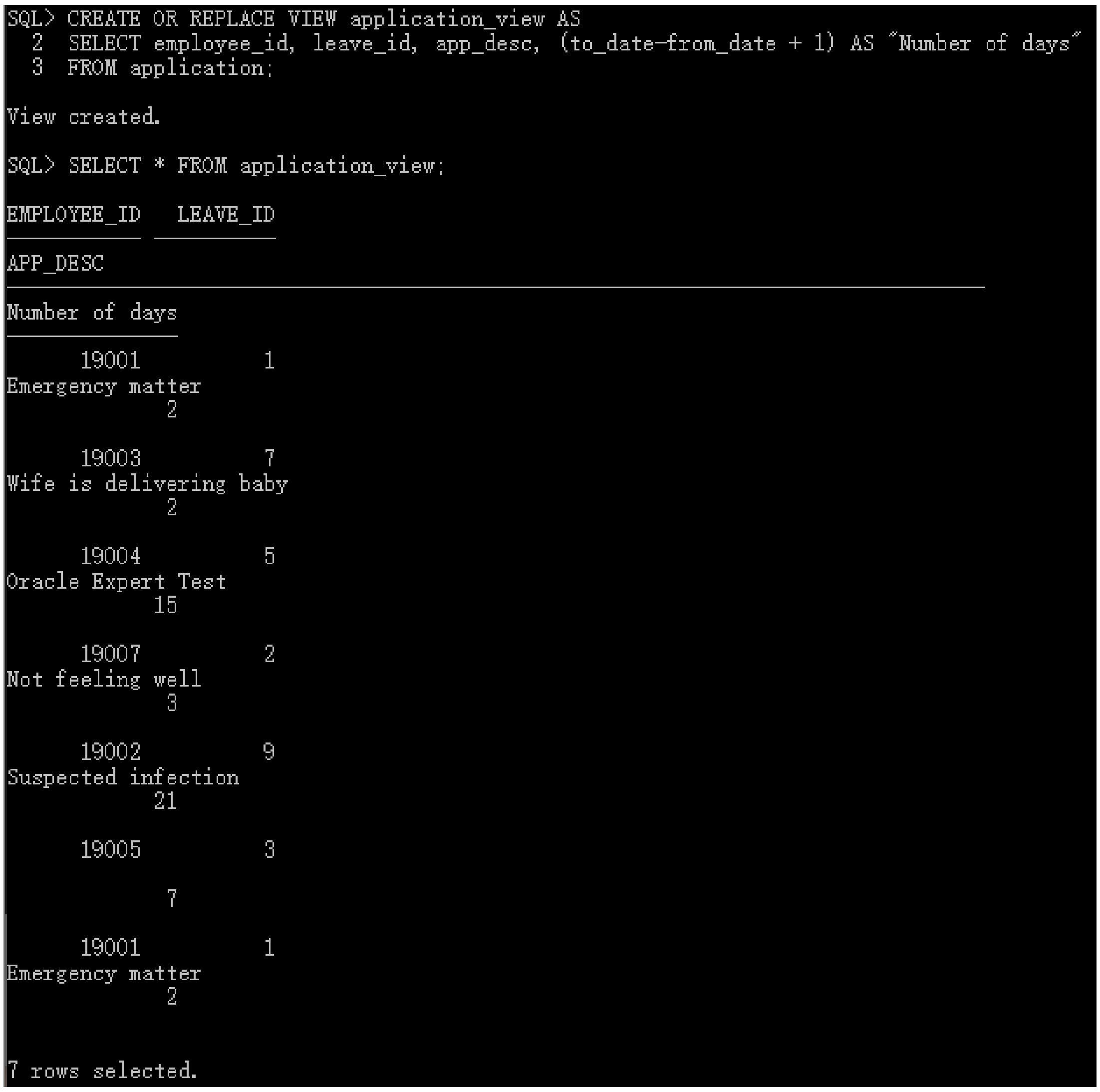


```
SQL> CREATE VIEW application_view AS
  2  SELECT employee_id, leave_id, app_desc, from_date, to_date, (to_date-from_date + 1) AS "Number of days"
  3  FROM application;

View created.

SQL> select * from application_view
  2  ;

EMPLOYEE_ID   LEAVE_ID
----------- ----------
APP_DESC
--------------------------------------------------------------------------------
FROM_DATE TO_DATE   Number of days
--------- --------- --------------
      19001          1
Emergency matter
21-FEB-20 22-FEB-20              2

      19003          7
Wife is delivering baby
21-MAR-19 22-MAR-19              2

      19004          5
Oracle Expert Test
02-MAY-18 16-MAY-18             15

      19007          2
Not feeling well
26-APR-14 28-APR-14              3

      19002          9
Suspected infection
13-NOV-15 03-DEC-15             21

      19005          3

28-MAY-10 03-JUN-10              7

      19001          1
Emergency matter
01-APR-20 02-APR-20              2


7 rows selected.
```

7. Update view of application table

   Sometimes the manager may need to have a new view without creating a new one, he/she may use this query to update existing view without having to create a new view. In this case from_date and to_date is removed from application_view.

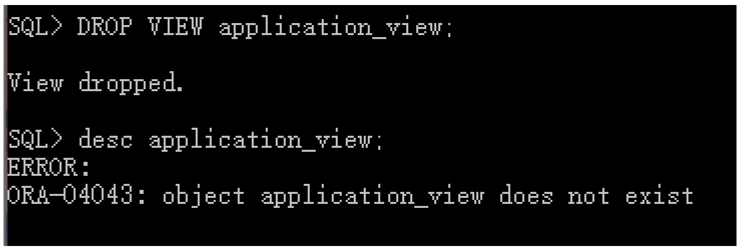
   CREATE OR REPLACE VIEW application_view AS
   SELECT employee_id, leave_id, app_desc, (to_date-from_date + 1) AS ”Number of days”
   FROM application;



```
SQL> CREATE OR REPLACE VIEW application_view AS
  2  SELECT employee_id, leave_id, app_desc, (to_date-from_date + 1) AS "Number of days"
  3  FROM application;

View created.

SQL> SELECT * FROM application_view;

EMPLOYEE_ID   LEAVE_ID
-----------  ----------
APP_DESC
--------------------------------------------------------------------------------
Number of days
--------------
      19001          1
Emergency matter
             2

      19003          7
Wife is delivering baby
             2

      19004          5
Oracle Expert Test
            15

      19007          2
Not feeling well
             3

      19002          9
Suspected infection
            21

      19005          3

             7

      19001          1
Emergency matter
             2


7 rows selected.
```

8. Delete unnecessary view
   When existing views are not needed anymore, the managers can use this query to delete the view.

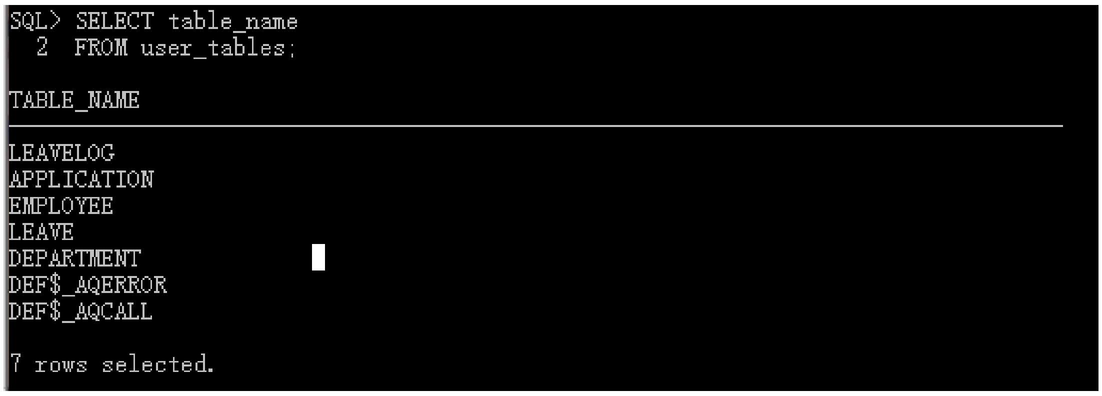
   DROP VIEW application_view;



```
SQL> DROP VIEW application_view;

View dropped.

SQL> desc application_view;
ERROR:
ORA-04043: object application_view does not exist
```

9. View all tables in the database
   This query allows the database managers to view all the tables in the database to know what are they working with.

   SELECT table_name
   FROM user_tables;

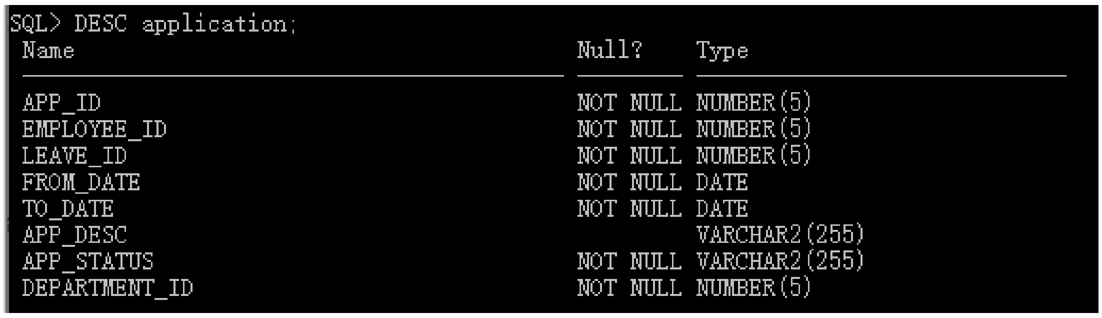


```
SQL> SELECT table_name
  2  FROM user_tables;

TABLE_NAME
------------------------------------------------------------

LEAVELOG
APPLICATION
EMPLOYEE
LEAVE
DEPARTMENT
DEF$_AQERROR
DEF$_AQCALL

7 rows selected.
```

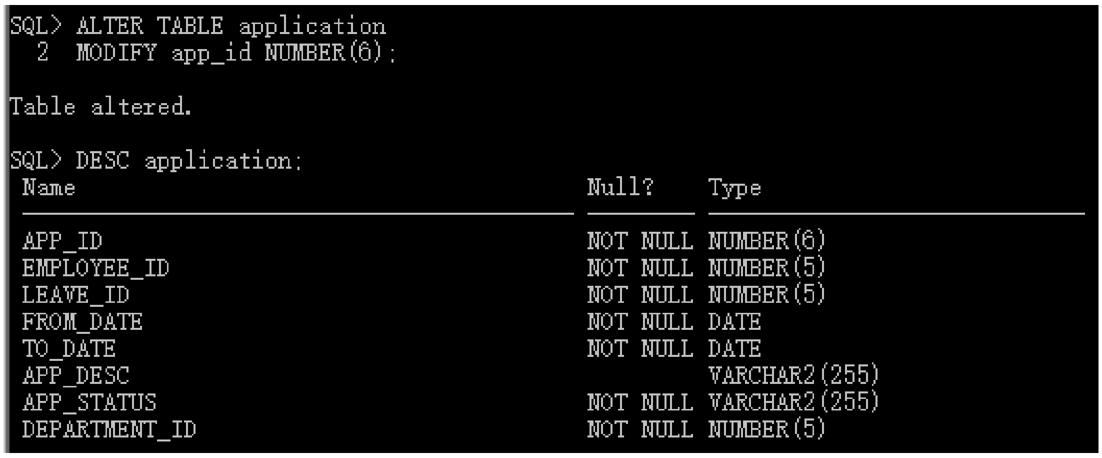
   DEF%_AQERROR and DEF%AQCALL are owned SYSTEM.

10. ALTER table
   As time goes, the id number may be used up and the manager may use this query to modify the type of the id. In this case, application table is used as an example.

   ALTER TABLE application
   MODIFY app_id NUMBER(6);



```
SQL> DESC application;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------

 APP_ID                                    NOT NULL NUMBER(5)
 EMPLOYEE_ID                               NOT NULL NUMBER(5)
 LEAVE_ID                                  NOT NULL NUMBER(5)
 FROM_DATE                                 NOT NULL DATE
 TO_DATE                                   NOT NULL DATE
 APP_DESC                                           VARCHAR2(255)
 APP_STATUS                                NOT NULL VARCHAR2(255)
 DEPARTMENT_ID                             NOT NULL NUMBER(5)
```

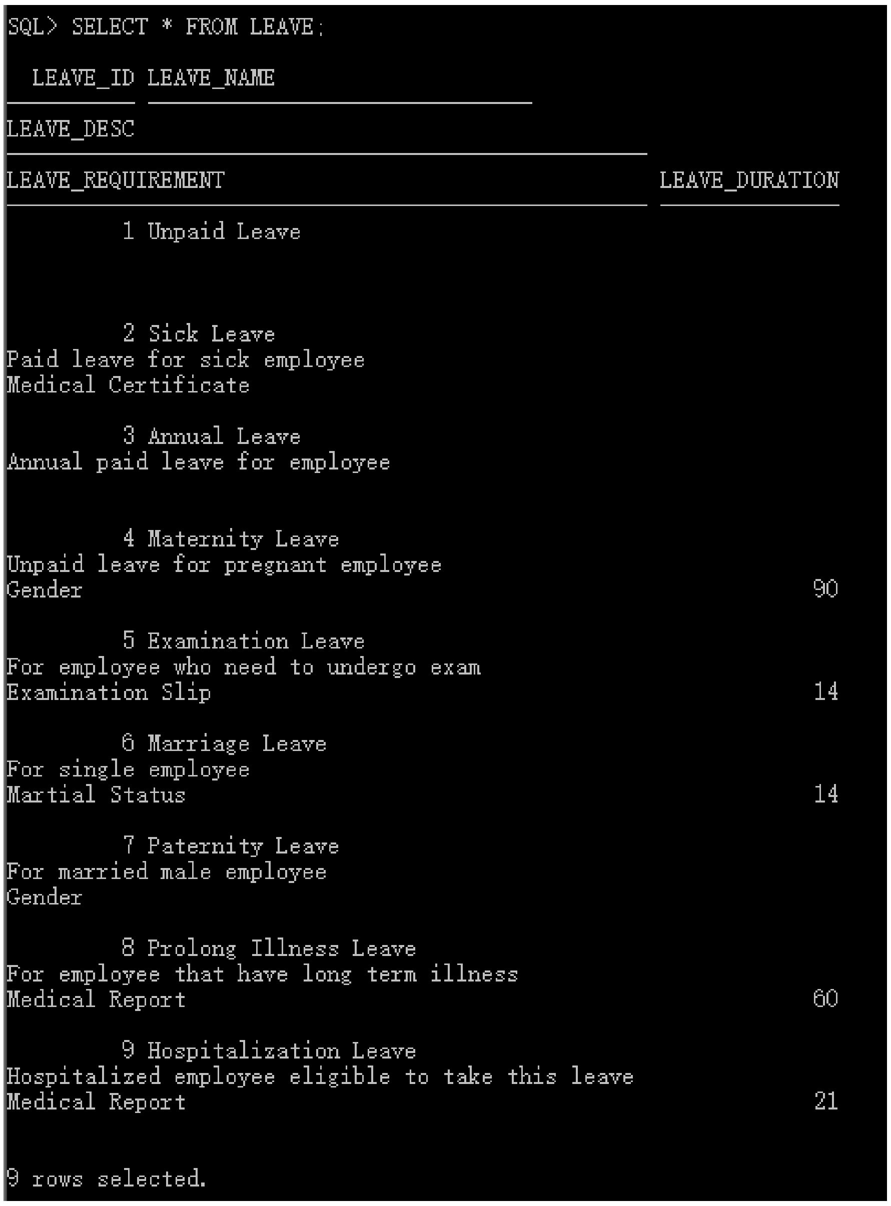


```
SQL> ALTER TABLE application
  2  MODIFY app_id NUMBER(6);

Table altered.

SQL> DESC application;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 APP_ID                                    NOT NULL NUMBER(6)
 EMPLOYEE_ID                               NOT NULL NUMBER(5)
 LEAVE_ID                                  NOT NULL NUMBER(5)
 FROM_DATE                                 NOT NULL DATE
 TO_DATE                                   NOT NULL DATE
 APP_DESC                                           VARCHAR2(255)
 APP_STATUS                                NOT NULL VARCHAR2(255)
 DEPARTMENT_ID                             NOT NULL NUMBER(5)
```

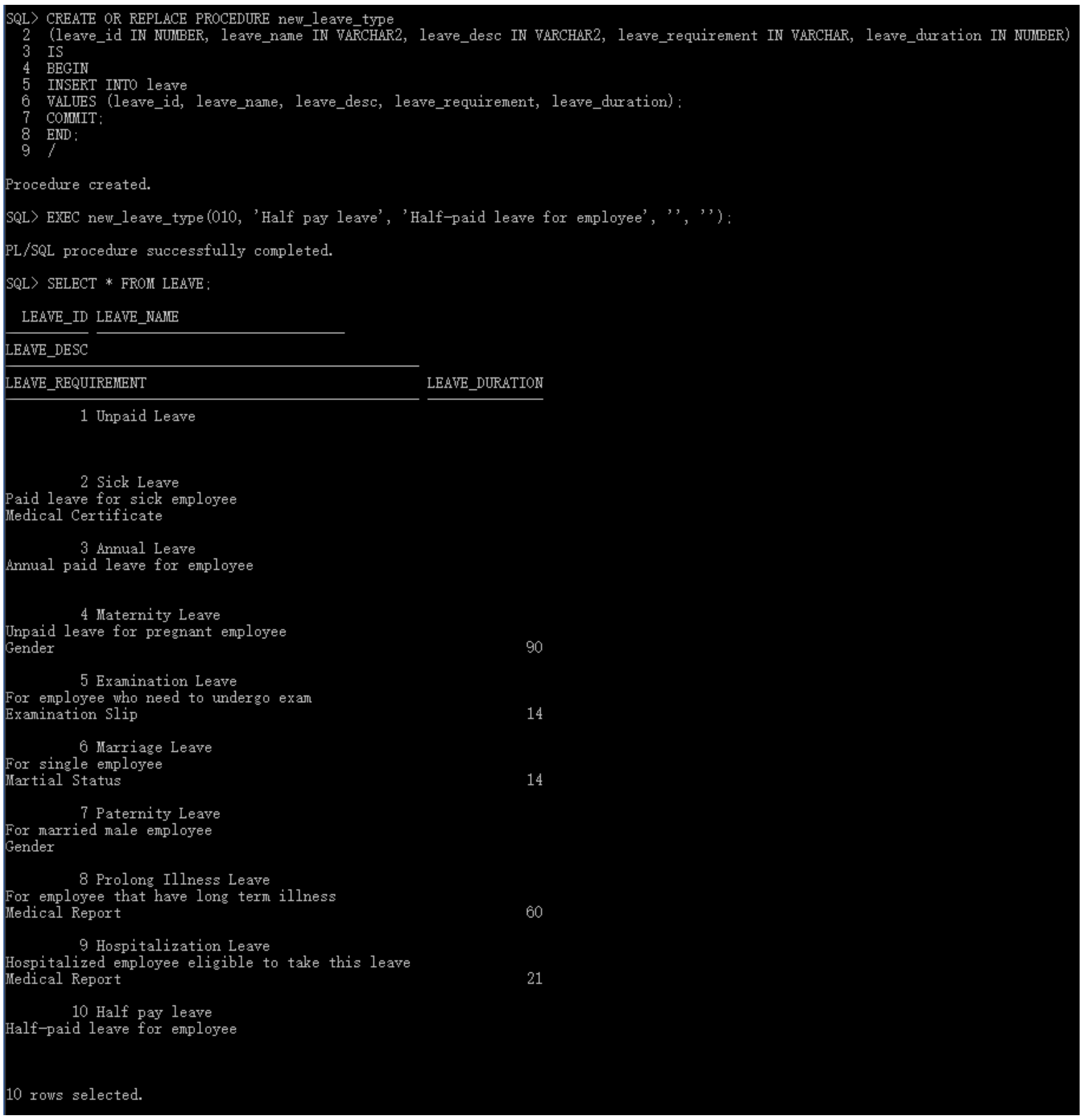
8.2 Stored Procedure

1. Add new leave type
   When the company decide to add a new type of leave, this procedure can be used.

```
CREATE OR REPLACE PROCEDURE new_leave_type
(leave_id IN NUMBER, leave_name IN VARCHAR2, leave_desc IN VARCHAR2,
leave_requirement IN VARCHAR, leave_duration IN NUMBER)
IS
BEGIN
INSERT INTO leave
VALUES (leave_id, leave_name, leave_desc, leave_requirement, leave_duration);
COMMIT;
END;
/

EXEC new_leave_type(010, 'Half pay leave', 'Half-paid leave for employee', '', '');
```



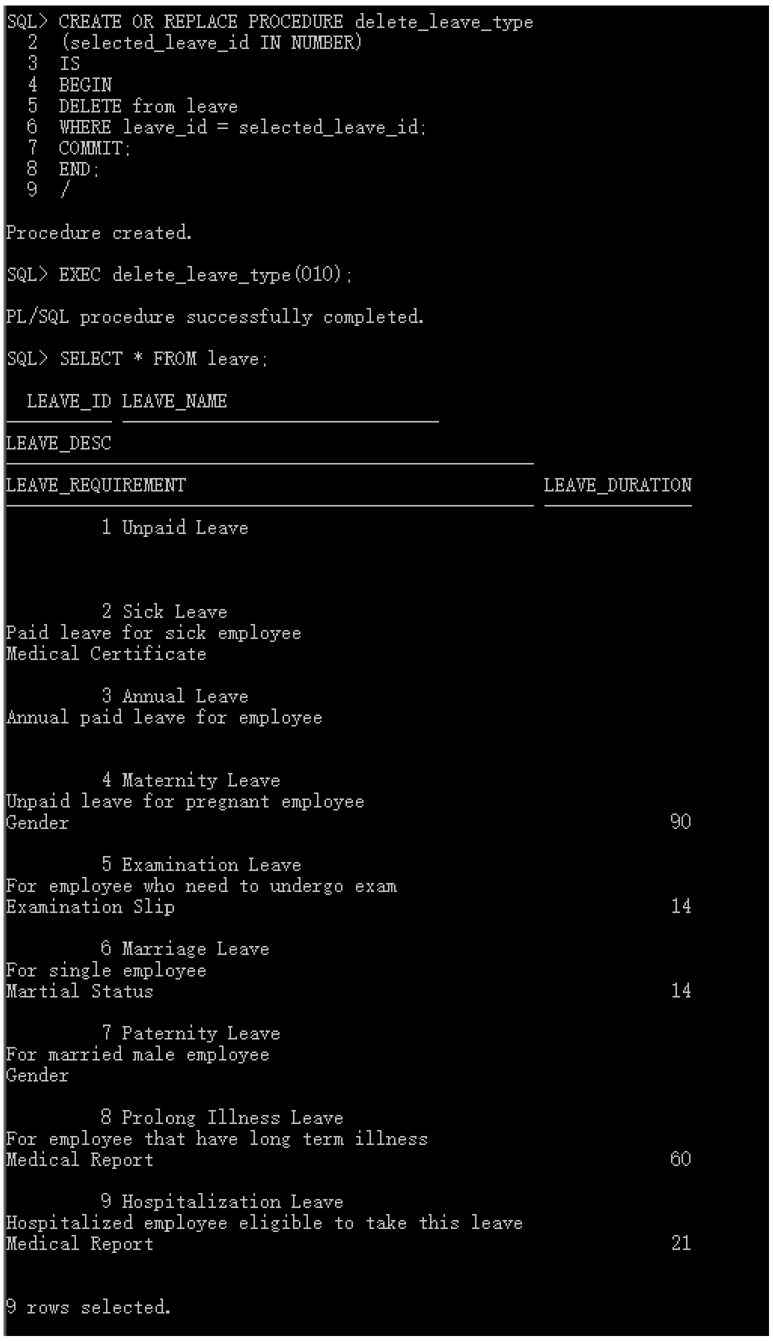
```
SQL> SELECT * FROM LEAVE;

  LEAVE_ID LEAVE_NAME
---------- ------------------------------
LEAVE_DESC
--------------------------------------------------
LEAVE_REQUIREMENT                                  LEAVE_DURATION
-------------------------------------------------- --------------
         1 Unpaid Leave



         2 Sick Leave
Paid leave for sick employee
Medical Certificate

         3 Annual Leave
Annual paid leave for employee


         4 Maternity Leave
Unpaid leave for pregnant employee
Gender                                                         90

         5 Examination Leave
For employee who need to undergo exam
Examination Slip                                               14

         6 Marriage Leave
For single employee
Martial Status                                                 14

         7 Paternity Leave
For married male employee
Gender

         8 Prolong Illness Leave
For employee that have long term illness
Medical Report                                                 60

         9 Hospitalization Leave
Hospitalized employee eligible to take this leave
Medical Report                                                 21


9 rows selected.
```



```
SQL> CREATE OR REPLACE PROCEDURE new_leave_type
     (leave_id IN NUMBER, leave_name IN VARCHAR2, leave_desc IN VARCHAR2, leave_requirement IN VARCHAR, leave_duration IN NUMBER)
     IS
     BEGIN
     INSERT INTO leave
     VALUES (leave_id, leave_name, leave_desc, leave_requirement, leave_duration);
     COMMIT;
     END;
     /

Procedure created.

SQL> EXEC new_leave_type(010, 'Half pay leave', 'Half-paid leave for employee', '', '');

PL/SQL procedure successfully completed.

SQL> SELECT * FROM LEAVE;

  LEAVE_ID LEAVE_NAME
---------- ------------------------------
LEAVE_DESC
--------------------------------------------------
LEAVE_REQUIREMENT                                  LEAVE_DURATION
-------------------------------------------------- --------------
         1 Unpaid Leave



         2 Sick Leave
Paid leave for sick employee
Medical Certificate

         3 Annual Leave
Annual paid leave for employee


         4 Maternity Leave
Unpaid leave for pregnant employee
Gender                                                         90

         5 Examination Leave
For employee who need to undergo exam
Examination Slip                                               14

         6 Marriage Leave
For single employee
Martial Status                                                 14

         7 Paternity Leave
For married male employee
Gender

         8 Prolong Illness Leave
For employee that have long term illness
Medical Report                                                 60

         9 Hospitalization Leave
Hospitalized employee eligible to take this leave
Medical Report                                                 21

        10 Half pay leave
Half-paid leave for employee



10 rows selected.
```

2. Delete leave type
   This procedure can be used to delete existing leave type.

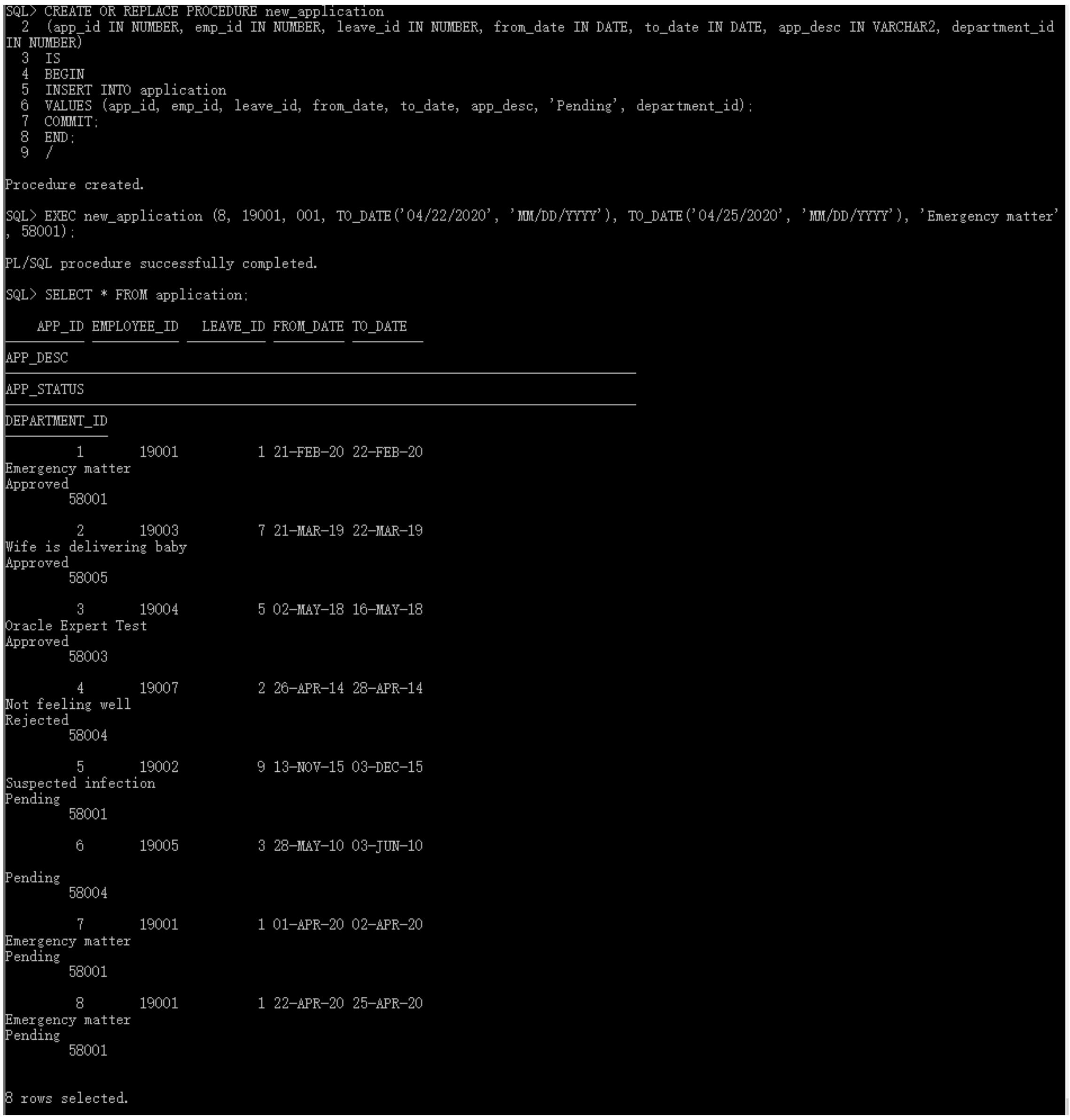
```
CREATE OR REPLACE PROCEDURE delete_leave_type
(selected_leave_id IN NUMBER)
IS
BEGIN
DELETE from leave
WHERE leave_id = selected_leave_id;
COMMIT;
END;
/
EXEC delete_leave_type(010);
```



```
SQL> CREATE OR REPLACE PROCEDURE delete_leave_type
  2  (selected_leave_id IN NUMBER)
  3  IS
  4  BEGIN
  5  DELETE from leave
  6  WHERE leave_id = selected_leave_id;
  7  COMMIT;
  8  END;
  9  /

Procedure created.

SQL> EXEC delete_leave_type(010);

PL/SQL procedure successfully completed.

SQL> SELECT * FROM leave;

  LEAVE_ID LEAVE_NAME
---------- ------------------------------
LEAVE_DESC
--------------------------------------------------
LEAVE_REQUIREMENT                                  LEAVE_DURATION
-------------------------------------------------- --------------
         1 Unpaid Leave



         2 Sick Leave
Paid leave for sick employee
Medical Certificate

         3 Annual Leave
Annual paid leave for employee


         4 Maternity Leave
Unpaid leave for pregnant employee
Gender                                                         90

         5 Examination Leave
For employee who need to undergo exam
Examination Slip                                               14

         6 Marriage Leave
For single employee
Martial Status                                                 14

         7 Paternity Leave
For married male employee
Gender

         8 Prolong Illness Leave
For employee that have long term illness
Medical Report                                                 60

         9 Hospitalization Leave
Hospitalized employee eligible to take this leave
Medical Report                                                 21


9 rows selected.
```

3. Insert new row into application
   Employee may use this procedure to insert new row into application when they are applying leave.

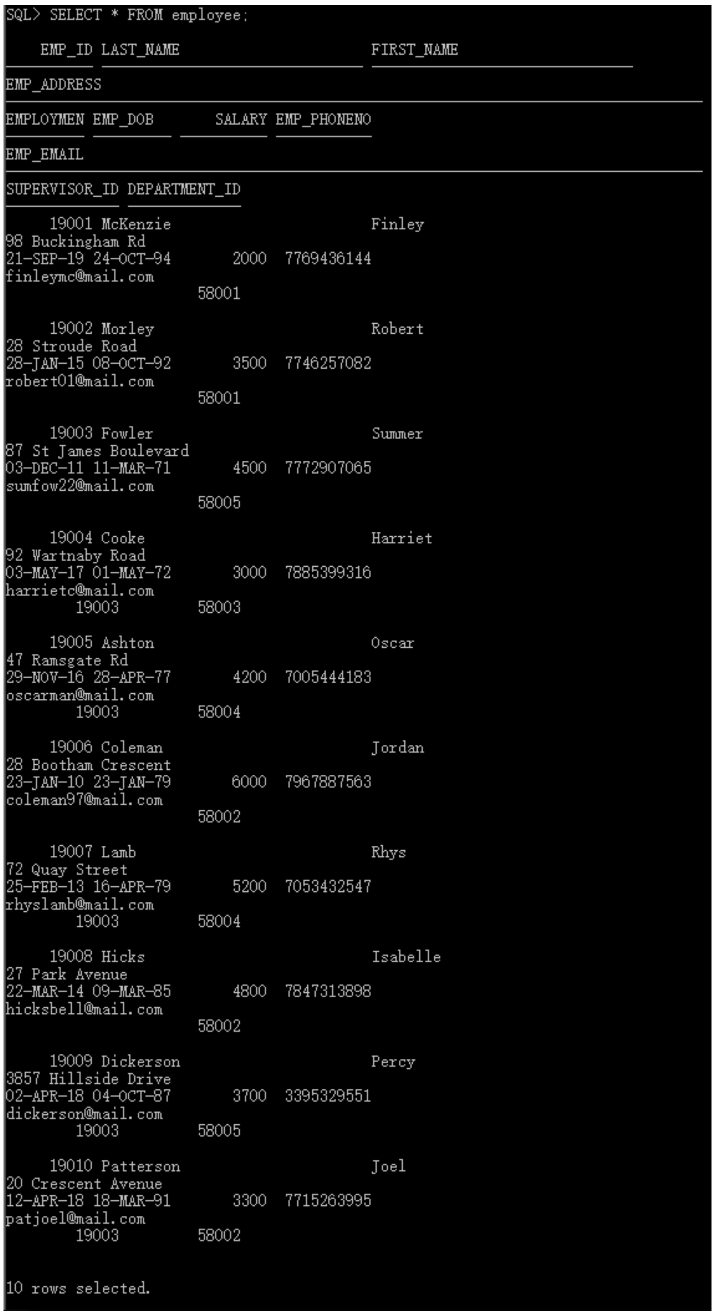
```
CREATE OR REPLACE PROCEDURE new_application
(app_id IN NUMBER, emp_id IN NUMBER, leave_id IN NUMBER, from_date IN
DATE, to_date IN DATE, app_desc IN VARCHAR2, department_id IN NUMBER)
IS
BEGIN
INSERT INTO application
VALUES (app_id, emp_id, leave_id, from_date, to_date, app_desc, 'Pending',
department_id);
COMMIT;
END;
/

EXEC new_application (8, 19001, 001, TO_DATE('04/22/2020', 'MM/DD/YYYY'),
TO_DATE('04/25/2020', 'MM/DD/YYYY'), 'Emergency matter', 58001);
```



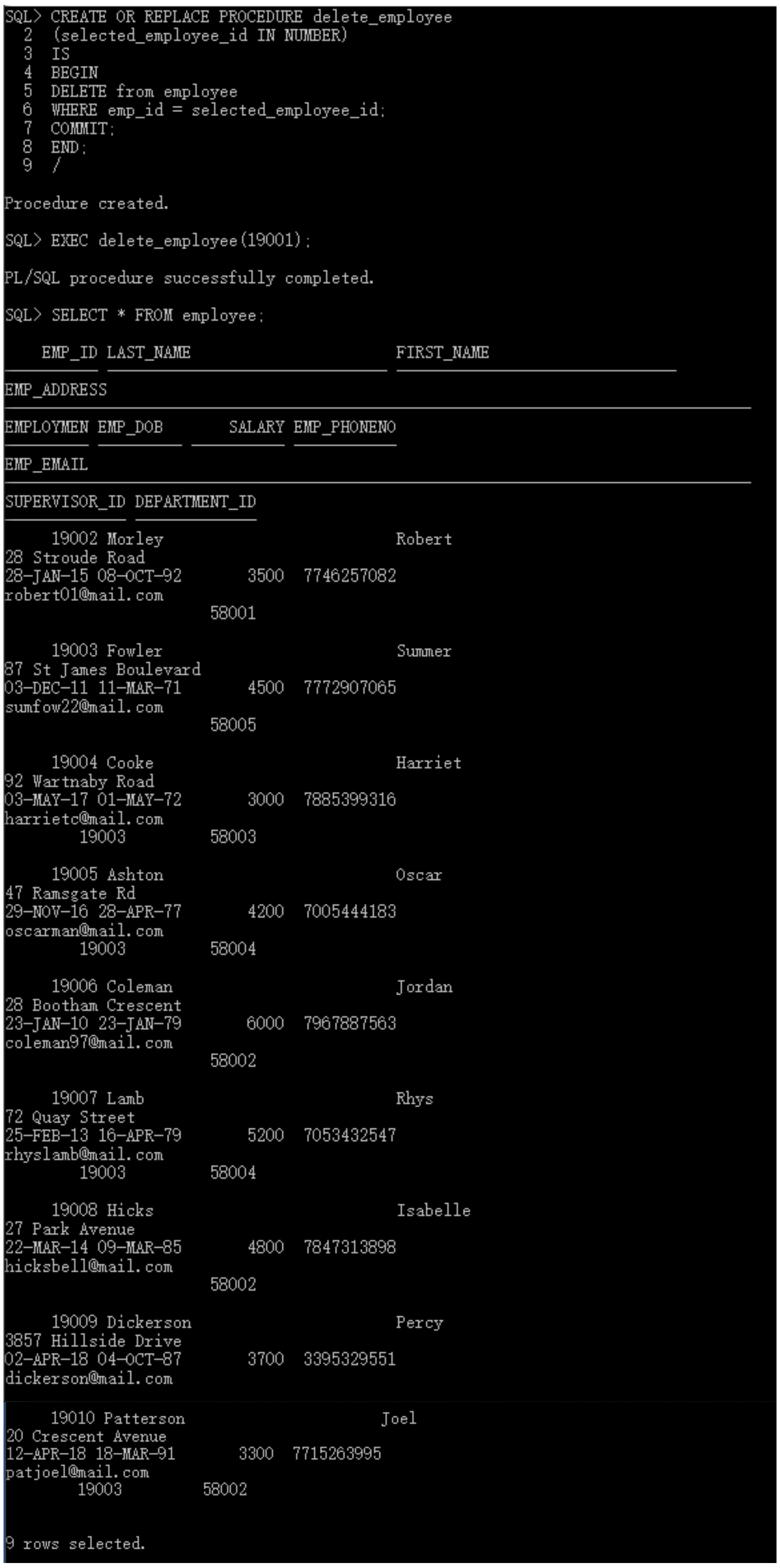
```
SQL> CREATE OR REPLACE PROCEDURE new_application
  2  (app_id IN NUMBER, emp_id IN NUMBER, leave_id IN NUMBER, from_date IN DATE, to_date IN DATE, app_desc IN VARCHAR2, department_id
IN NUMBER)
  3  IS
  4  BEGIN
  5  INSERT INTO application
  6  VALUES (app_id, emp_id, leave_id, from_date, to_date, app_desc, 'Pending', department_id);
  7  COMMIT;
  8  END;
  9  /

Procedure created.

SQL> EXEC new_application (8, 19001, 001, TO_DATE('04/22/2020', 'MM/DD/YYYY'), TO_DATE('04/25/2020', 'MM/DD/YYYY'), 'Emergency matter'
, 58001);

PL/SQL procedure successfully completed.

SQL> SELECT * FROM application;

    APP_ID EMPLOYEE_ID   LEAVE_ID FROM_DATE TO_DATE
---------- ----------- ---------- --------- ---------
APP_DESC
--------------------------------------------------------------------------------
APP_STATUS
--------------------------------------------------------------------------------
DEPARTMENT_ID
-------------
         1       19001          1 21-FEB-20 22-FEB-20
Emergency matter
Approved
        58001

         2       19003          7 21-MAR-19 22-MAR-19
Wife is delivering baby
Approved
        58005

         3       19004          5 02-MAY-18 16-MAY-18
Oracle Expert Test
Approved
        58003

         4       19007          2 26-APR-14 28-APR-14
Not feeling well
Rejected
        58004

         5       19002          9 13-NOV-15 03-DEC-15
Suspected infection
Pending
        58001

         6       19005          3 28-MAY-10 03-JUN-10

Pending
        58004

         7       19001          1 01-APR-20 02-APR-20
Emergency matter
Pending
        58001

         8       19001          1 22-APR-20 25-APR-20
Emergency matter
Pending
        58001


8 rows selected.
```

4. Delete employee
   When the company fires an employee, this procedure can be used to delete data for that employee.

```
CREATE OR REPLACE PROCEDURE delete_employee
(selected_employee_id IN NUMBER)
IS
BEGIN
DELETE from employee
WHERE emp_id = selected_employee_id;
COMMIT;
END;
/

EXEC delete_employee(19001);
```



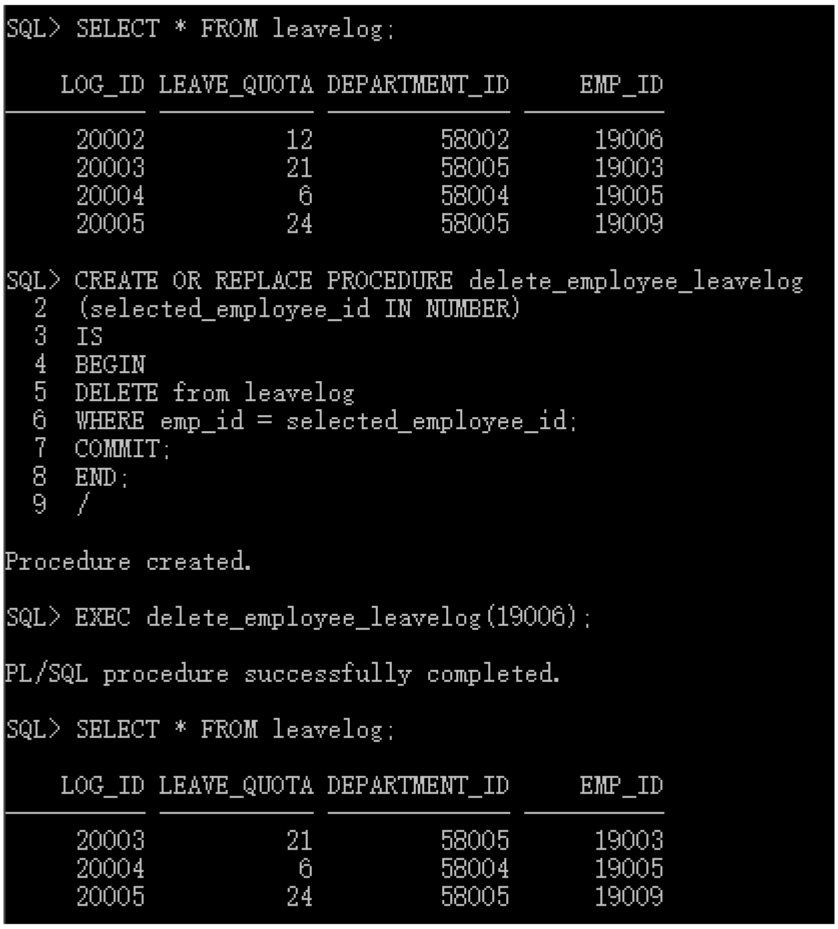
```
SQL> SELECT * FROM employee;

    EMP_ID LAST_NAME                      FIRST_NAME
---------- ------------------------------ ------------------------------
EMP_ADDRESS
--------------------------------------------------------------------------------
EMPLOYMEN EMP_DOB       SALARY EMP_PHONENO
--------- --------- ---------- -----------
EMP_EMAIL
--------------------------------------------------------------------------------
SUPERVISOR_ID DEPARTMENT_ID
------------- -------------
     19001 McKenzie                       Finley
98 Buckingham Rd
21-SEP-19 24-OCT-94       2000  7769436144
finleymc@mail.com
                      58001

     19002 Morley                         Robert
28 Stroude Road
28-JAN-15 08-OCT-92       3500  7746257082
robert01@mail.com
                      58001

     19003 Fowler                         Summer
87 St James Boulevard
03-DEC-11 11-MAR-71       4500  7772907065
sumfow22@mail.com
                      58005

     19004 Cooke                          Harriet
92 Wartnaby Road
03-MAY-17 01-MAY-72       3000  7885399316
harrietc@mail.com
        19003         58003

     19005 Ashton                         Oscar
47 Ramsgate Rd
29-NOV-16 28-APR-77       4200  7005444183
oscarman@mail.com
        19003         58004

     19006 Coleman                        Jordan
28 Bootham Crescent
23-JAN-10 23-JAN-79       6000  7967887563
coleman97@mail.com
                      58002

     19007 Lamb                           Rhys
72 Quay Street
25-FEB-13 16-APR-79       5200  7053432547
rhyslamb@mail.com
        19003         58004

     19008 Hicks                          Isabelle
27 Park Avenue
22-MAR-14 09-MAR-85       4800  7847313898
hicksbell@mail.com
                      58002

     19009 Dickerson                      Percy
3857 Hillside Drive
02-APR-18 04-OCT-87       3700  3395329551
dickerson@mail.com
        19003         58005

     19010 Patterson                      Joel
20 Crescent Avenue
12-APR-18 18-MAR-91       3300  7715263995
patjoel@mail.com
        19003         58002


10 rows selected.
```



```
SQL> CREATE OR REPLACE PROCEDURE delete_employee
     (selected_employee_id IN NUMBER)
     IS
     BEGIN
     DELETE from employee
     WHERE emp_id = selected_employee_id;
     COMMIT;
     END;
     /

Procedure created.

SQL> EXEC delete_employee(19001);

PL/SQL procedure successfully completed.

SQL> SELECT * FROM employee;

    EMP_ID LAST_NAME                      FIRST_NAME
---------- ------------------------------ ------------------------------
EMP_ADDRESS
--------------------------------------------------------------------------------
EMPLOYMEN EMP_DOB       SALARY EMP_PHONENO
--------- --------- ---------- ----------
EMP_EMAIL
--------------------------------------------------------------------------------
SUPERVISOR_ID DEPARTMENT_ID
------------- -------------
     19002 Morley                         Robert
28 Stroude Road
28-JAN-15 08-OCT-92       3500  7746257082
robert01@mail.com
                      58001

     19003 Fowler                         Summer
87 St James Boulevard
03-DEC-11 11-MAR-71       4500  7772907065
sumfow22@mail.com
                      58005

     19004 Cooke                          Harriet
92 Wartnaby Road
03-MAY-17 01-MAY-72       3000  7885399316
harrietc@mail.com
        19003         58003

     19005 Ashton                         Oscar
47 Ramsgate Rd
29-NOV-16 28-APR-77       4200  7005444183
oscarman@mail.com
        19003         58004

     19006 Coleman                        Jordan
28 Bootham Crescent
23-JAN-10 23-JAN-79       6000  7967887563
coleman97@mail.com
                      58002

     19007 Lamb                           Rhys
72 Quay Street
25-FEB-13 16-APR-79       5200  7053432547
rhyslamb@mail.com
        19003         58004

     19008 Hicks                          Isabelle
27 Park Avenue
22-MAR-14 09-MAR-85       4800  7847313898
hicksbell@mail.com
                      58002

     19009 Dickerson                      Percy
3857 Hillside Drive
02-APR-18 04-OCT-87       3700  3395329551
dickerson@mail.com

     19010 Patterson                      Joel
20 Crescent Avenue
12-APR-18 18-MAR-91       3300  7715263995
patjoel@mail.com
        19003         58002


9 rows selected.
```

5. Delete employee in leavelog
   When an employee is fired, this procedure can be used to delete leavelog records for that employee.

```
CREATE OR REPLACE PROCEDURE delete_employee_leavelog
(selected_employee_id IN NUMBER)
IS
BEGIN
DELETE from leavelog
WHERE emp_id = selected_employee_id;
COMMIT;
END;
/

EXEC delete_employee_leavelog(19006);
```

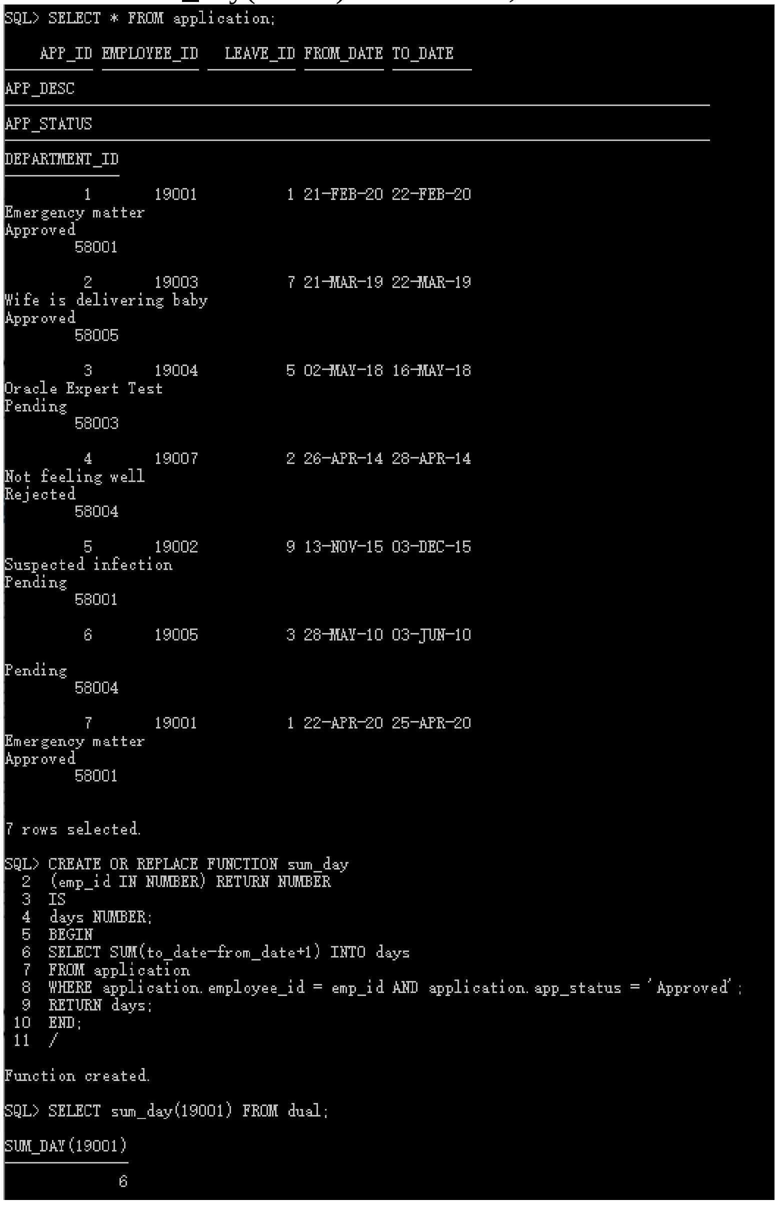


```
SQL> SELECT * FROM leavelog;

    LOG_ID LEAVE_QUOTA DEPARTMENT_ID     EMP_ID
---------- ----------- ------------- ----------
     20002          12         58002      19006
     20003          21         58005      19003
     20004           6         58004      19005
     20005          24         58005      19009

SQL> CREATE OR REPLACE PROCEDURE delete_employee_leavelog
  2  (selected_employee_id IN NUMBER)
  3  IS
  4  BEGIN
  5  DELETE from leavelog
  6  WHERE emp_id = selected_employee_id;
  7  COMMIT;
  8  END;
  9  /

Procedure created.

SQL> EXEC delete_employee_leavelog(19006);

PL/SQL procedure successfully completed.

SQL> SELECT * FROM leavelog;

    LOG_ID LEAVE_QUOTA DEPARTMENT_ID     EMP_ID
---------- ----------- ------------- ----------
     20003          21         58005      19003
     20004           6         58004      19005
     20005          24         58005      19009
```

8.3 Functions

1. Get the sum of days for the leaves taken of selected employee
   This function is to get the sum of days for the leaves taken of selected employee.

```
CREATE OR REPLACE FUNCTION sum_day
(emp_id IN NUMBER) RETURN NUMBER
IS
days NUMBER;
BEGIN
SELECT SUM(to_date-from_date+1) INTO days
FROM application
WHERE application.employee_id = emp_id AND application.app_status =
'Approved';
RETURN days;
END;
/

SELECT sum_day(19001) FROM dual;
```

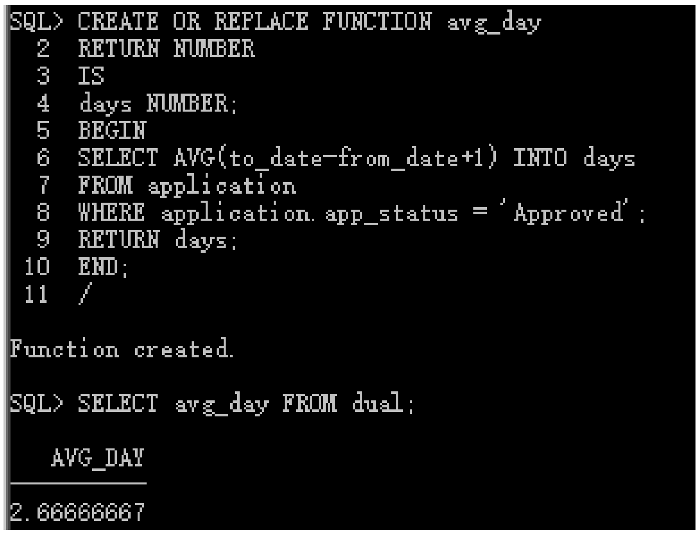


```
SQL> SELECT * FROM application;

    APP_ID EMPLOYEE_ID   LEAVE_ID FROM_DATE TO_DATE
---------- ----------- ---------- --------- ---------
APP_DESC
--------------------------------------------------------------------------------
APP_STATUS
--------------------------------------------------------------------------------
DEPARTMENT_ID
-------------
         1       19001          1 21-FEB-20 22-FEB-20
Emergency matter
Approved
        58001

         2       19003          7 21-MAR-19 22-MAR-19
Wife is delivering baby
Approved
        58005

         3       19004          5 02-MAY-18 16-MAY-18
Oracle Expert Test
Pending
        58003

         4       19007          2 26-APR-14 28-APR-14
Not feeling well
Rejected
        58004

         5       19002          9 13-NOV-15 03-DEC-15
Suspected infection
Pending
        58001

         6       19005          3 28-MAY-10 03-JUN-10

Pending
        58004

         7       19001          1 22-APR-20 25-APR-20
Emergency matter
Approved
        58001


7 rows selected.

SQL> CREATE OR REPLACE FUNCTION sum_day
     (emp_id IN NUMBER) RETURN NUMBER
     IS
     days NUMBER;
     BEGIN
     SELECT SUM(to_date-from_date+1) INTO days
     FROM application
     WHERE application.employee_id = emp_id AND application.app_status = 'Approved';
     RETURN days;
     END;
     /

Function created.

SQL> SELECT sum_day(19001) FROM dual;

SUM_DAY(19001)
--------------
             6
```

2. Get the average days taken by the employees
   This function is to get the average days of leave taken by the employees.

```
CREATE OR REPLACE FUNCTION avg_day
RETURN NUMBER
IS
days NUMBER;
BEGIN
SELECT AVG(to_date-from_date+1) INTO days
FROM application
WHERE application.app_status = 'Approved';
RETURN days;
END;
/
```

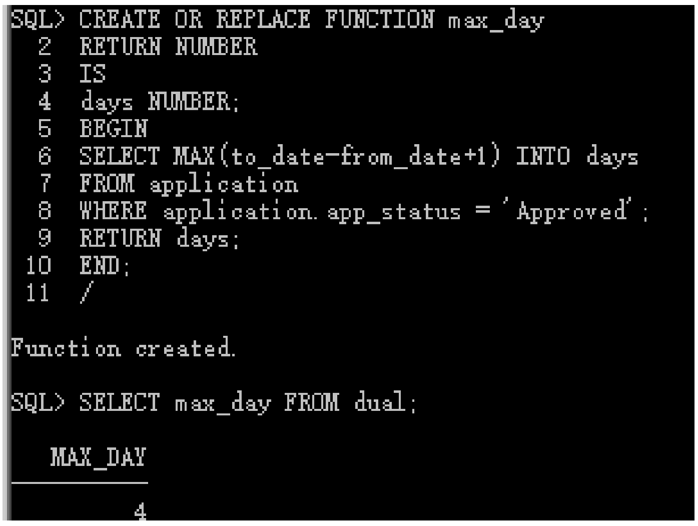
```
SELECT avg_day FROM dual;
```



```
SQL> CREATE OR REPLACE FUNCTION avg_day
  2  RETURN NUMBER
  3  IS
  4  days NUMBER;
  5  BEGIN
  6  SELECT AVG(to_date-from_date+1) INTO days
  7  FROM application
  8  WHERE application.app_status = 'Approved';
  9  RETURN days;
 10  END;
 11  /

Function created.

SQL> SELECT avg_day FROM dual;

   AVG_DAY
----------
2.66666667
```

3. Get the maximum days taken by the employees
   This function is to get the maximum days of leave taken by the employees.

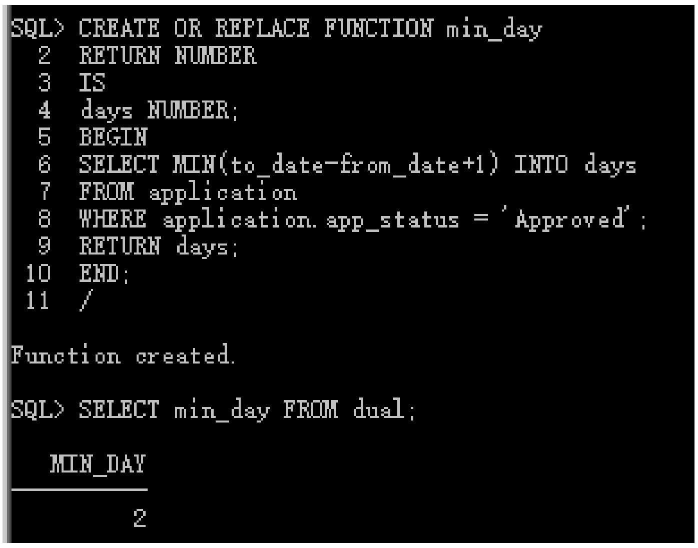
```
CREATE OR REPLACE FUNCTION max_day
RETURN NUMBER
IS
days NUMBER;
BEGIN
SELECT MAX(to_date-from_date+1) INTO days
FROM application
WHERE application.app_status = 'Approved';
RETURN days;
END;
/

SELECT max_day FROM dual;
```



```
SQL> CREATE OR REPLACE FUNCTION max_day
  2  RETURN NUMBER
  3  IS
  4  days NUMBER;
  5  BEGIN
  6  SELECT MAX(to_date-from_date+1) INTO days
  7  FROM application
  8  WHERE application.app_status = 'Approved';
  9  RETURN days;
 10  END;
 11  /

Function created.

SQL> SELECT max_day FROM dual;

   MAX_DAY
----------
         4
```

4. Get the minimum days taken by the employees
   This function is to get the minimum days of leave taken by the employees.

```
CREATE OR REPLACE FUNCTION min_day
RETURN NUMBER
IS
days NUMBER;
BEGIN
SELECT MIN(to_date-from_date+1) INTO days
FROM application
WHERE application.app_status = 'Approved';
RETURN days;
END;
/

SELECT min_day FROM dual;
```



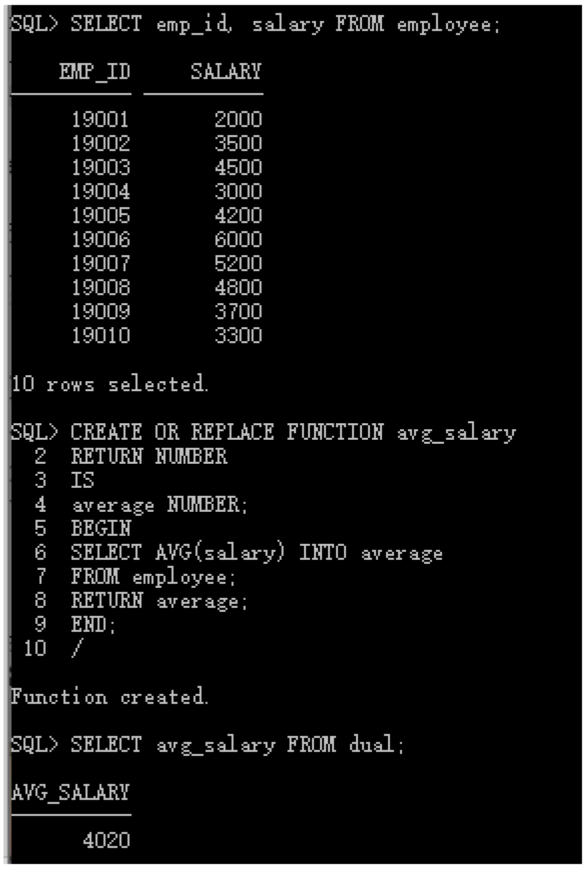
```
SQL> CREATE OR REPLACE FUNCTION min_day
  2  RETURN NUMBER
  3  IS
  4  days NUMBER;
  5  BEGIN
  6  SELECT MIN(to_date-from_date+1) INTO days
  7  FROM application
  8  WHERE application.app_status = 'Approved';
  9  RETURN days;
 10  END;
 11  /

Function created.

SQL> SELECT min_day FROM dual;

   MIN_DAY
----------
         2
```

5. Get the average salary of the employees
   This function is to get the average salary of the employees.

```
CREATE OR REPLACE FUNCTION avg_salary
RETURN NUMBER
IS
average NUMBER;
BEGIN
SELECT AVG(salary) INTO average
FROM employee;
RETURN average;
END;
```

/

SELECT avg_salary FROM dual;



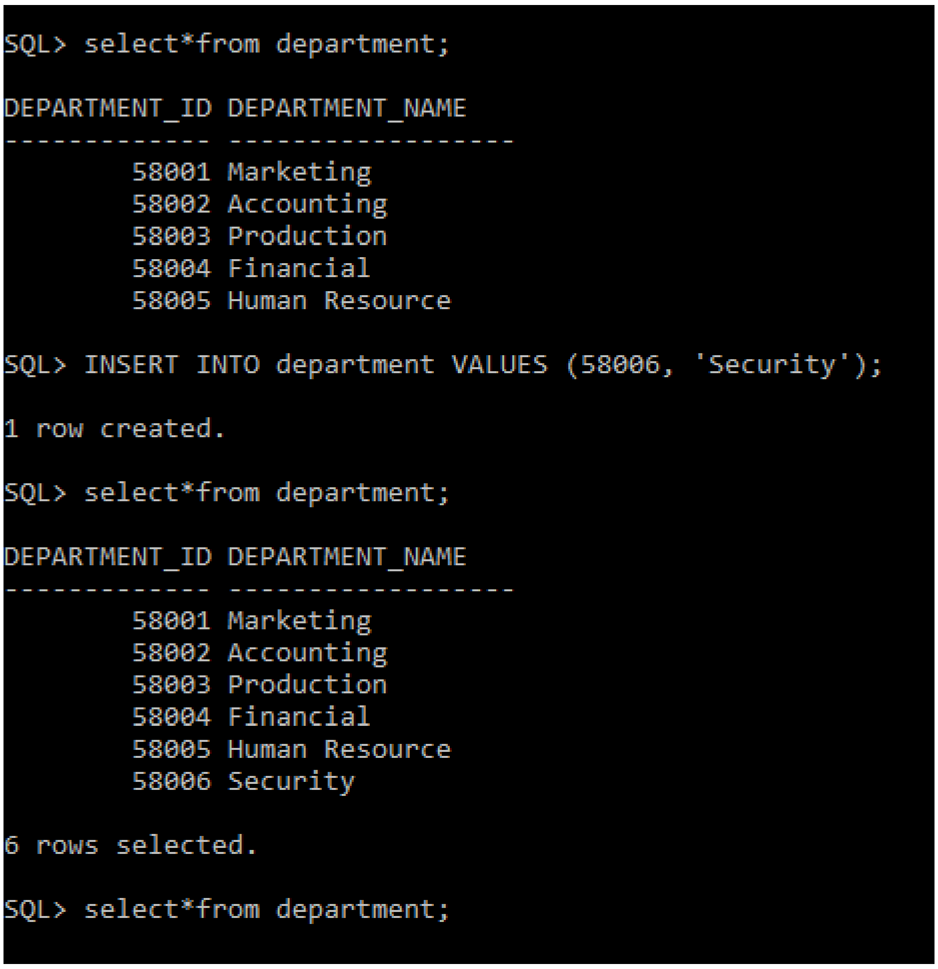
```
SQL> SELECT emp_id, salary FROM employee;

    EMP_ID     SALARY
---------- ----------
     19001       2000
     19002       3500
     19003       4500
     19004       3000
     19005       4200
     19006       6000
     19007       5200
     19008       4800
     19009       3700
     19010       3300

10 rows selected.

SQL> CREATE OR REPLACE FUNCTION avg_salary
  2  RETURN NUMBER
  3  IS
  4  average NUMBER;
  5  BEGIN
  6  SELECT AVG(salary) INTO average
  7  FROM employee;
  8  RETURN average;
  9  END;
 10  /

Function created.

SQL> SELECT avg_salary FROM dual;

AVG_SALARY
----------
      4020
```

9.0 Individual Assessment (Wee Yiiheen, 1604297)

9.1 Queries

1. Create a new department
   If the company would like to add a new department to store the department information of their company, he/she may use this query to create a column using the query below.

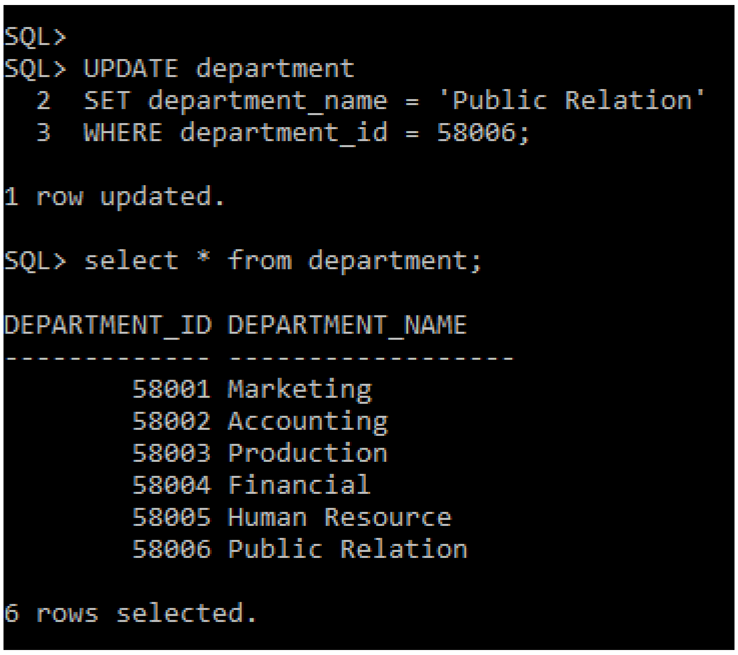
   INSERT INTO department VALUES (58006, 'Security');



```
SQL> select*from department;

DEPARTMENT_ID DEPARTMENT_NAME
------------- -----------------
        58001 Marketing
        58002 Accounting
        58003 Production
        58004 Financial
        58005 Human Resource

SQL> INSERT INTO department VALUES (58006, 'Security');

1 row created.

SQL> select*from department;

DEPARTMENT_ID DEPARTMENT_NAME
------------- -----------------
        58001 Marketing
        58002 Accounting
        58003 Production
        58004 Financial
        58005 Human Resource
        58006 Security

6 rows selected.

SQL> select*from department;
```

2. Update an existing department
   The company owner might rename the department to be more specific depending on the task they being allocated to. The department information stored inside the sql can be updated using the query below.

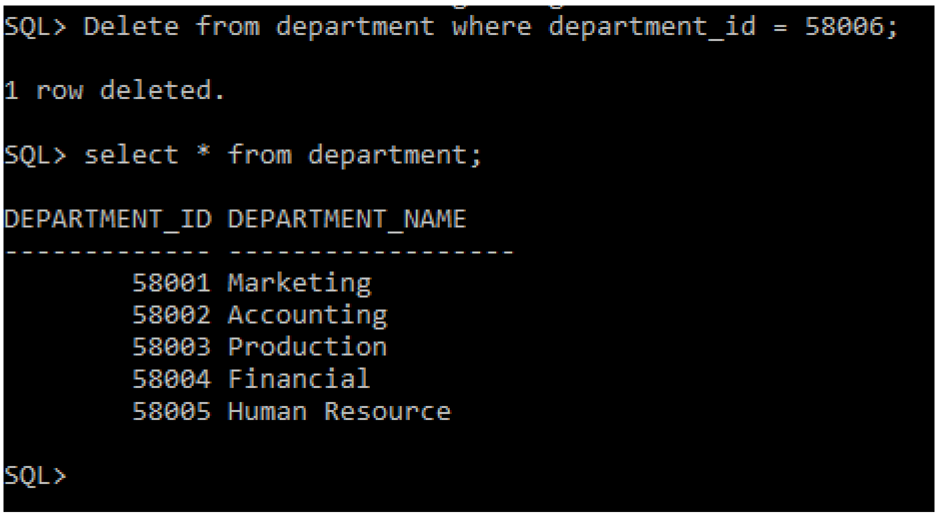
   UPDATE department
   SET department_name = 'Public Relation'
   WHERE department_id = 58006;



```
SQL>
SQL> UPDATE department
  2  SET department_name = 'Public Relation'
  3  WHERE department_id = 58006;

1 row updated.

SQL> select * from department;

DEPARTMENT_ID DEPARTMENT_NAME
------------- ------------------
        58001 Marketing
        58002 Accounting
        58003 Production
        58004 Financial
        58005 Human Resource
        58006 Public Relation

6 rows selected.
```

3. Delete a department
   Sometime the company might undergo low tide, in this case the company might decided to cut expenses by closing or combining some department. To delete a department from sql plus, the query below can be used.

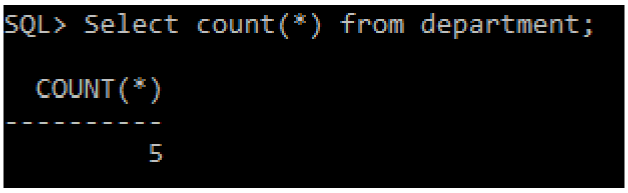
   Delete from department where department_id = 58006;



```
SQL> Delete from department where department_id = 58006;

1 row deleted.

SQL> select * from department;

DEPARTMENT_ID DEPARTMENT_NAME
------------- ------------------
        58001 Marketing
        58002 Accounting
        58003 Production
        58004 Financial
        58005 Human Resource

SQL>
```

4. Display the total number of department.
   In big company, the number of department might be huge, query below allows the user to quickly check the number of department in the company.

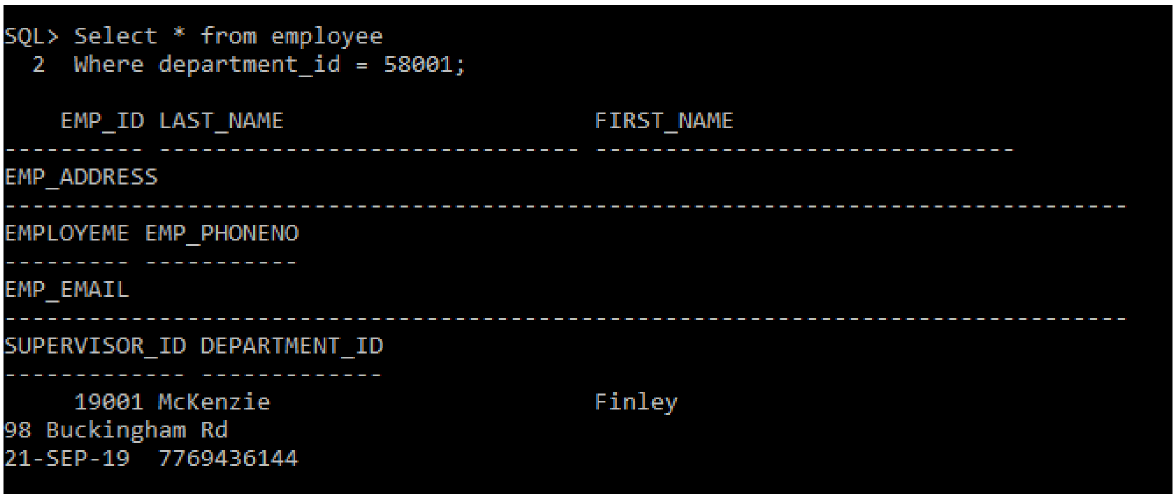
   Select count(*) from department;



```
SQL> Select count(*) from department;

  COUNT(*)
----------
         5
```

5. Checking the employee given department_id
   For example, the manager would like to know who belongs to the marketing department, the manager can check the employee using the query below.

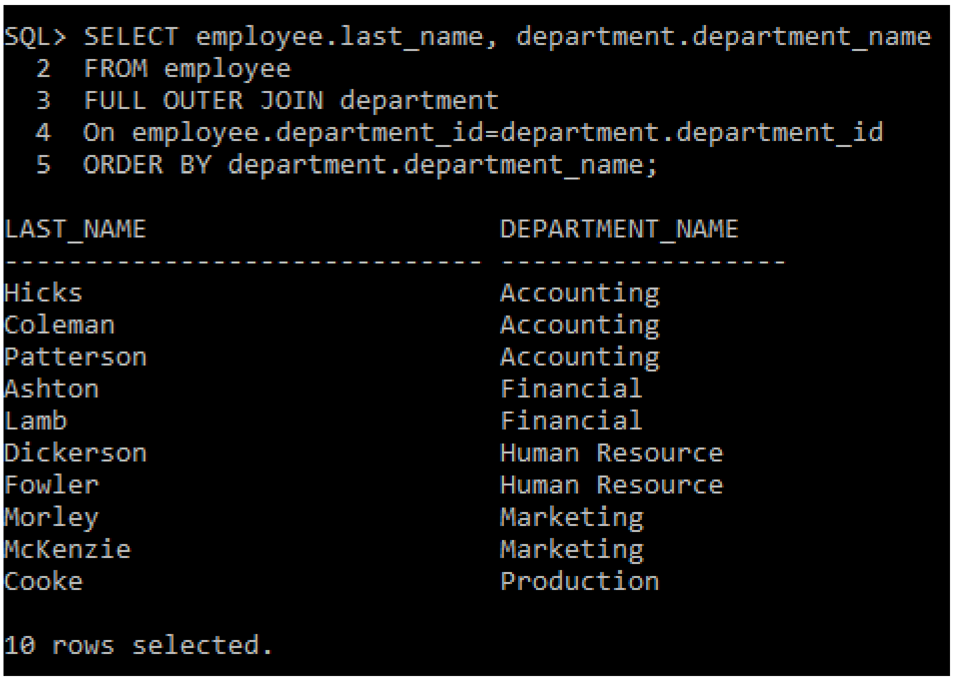
   Select * from employee
   Where department_id = 58001;



```
SQL> Select * from employee
  2  Where department_id = 58001;

    EMP_ID LAST_NAME                      FIRST_NAME
---------- ------------------------------ ------------------------------
EMP_ADDRESS
--------------------------------------------------------------------------------
EMPLOYEME EMP_PHONENO
--------- -----------
EMP_EMAIL
--------------------------------------------------------------------------------
SUPERVISOR_ID DEPARTMENT_ID
------------- -------------
     19001 McKenzie                       Finley
98 Buckingham Rd
21-SEP-19  7769436144
```

6. Listing all employees and their assigned department.
   If the manager would like to check whether all employee are being assigned to department, he/she can use this query to check.

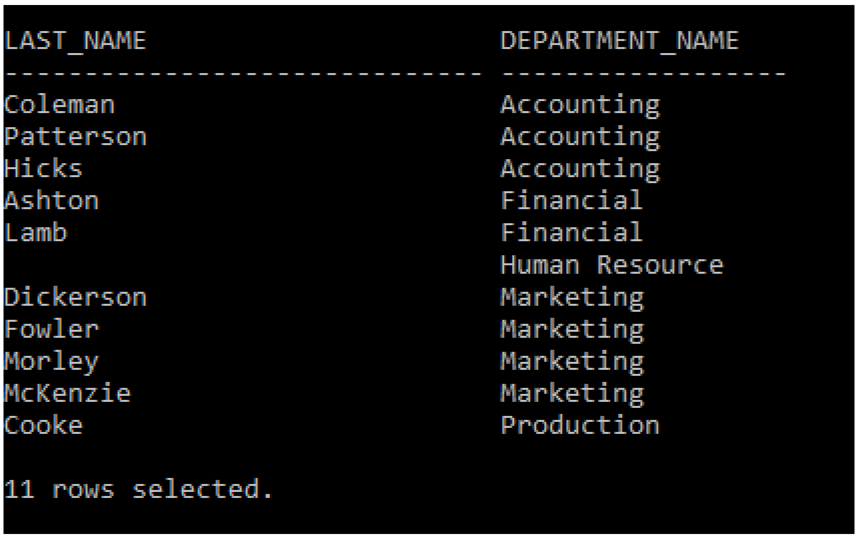
   SELECT employee.last_name, department.department_name
   FROM employee
   FULL OUTER JOIN department
   On employee.department_id=department.department_id
   ORDER BY department.department_name;



```
SQL> SELECT employee.last_name, department.department_name
  2  FROM employee
  3  FULL OUTER JOIN department
  4  On employee.department_id=department.department_id
  5  ORDER BY department.department_name;

LAST_NAME                      DEPARTMENT_NAME
------------------------------ ------------------
Hicks                          Accounting
Coleman                        Accounting
Patterson                      Accounting
Ashton                         Financial
Lamb                           Financial
Dickerson                      Human Resource
Fowler                         Human Resource
Morley                         Marketing
McKenzie                       Marketing
Cooke                          Production

10 rows selected.
```

7. Moving all employee from Human Resource department to Marketing department. After moving the employee to a new department, the Human Resource department is now empty.

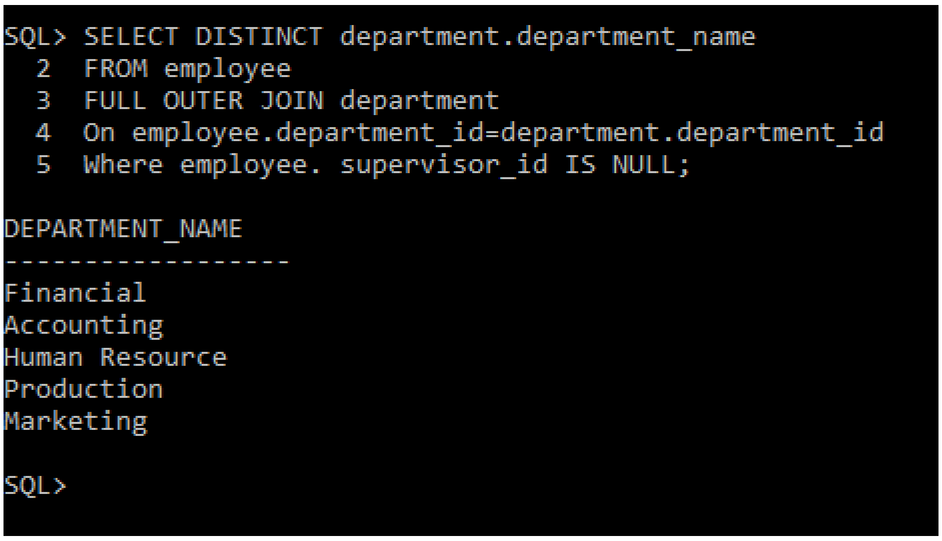
   UPDATE employee
   SET department_id = '58001'
   WHERE department_id = '58005';



```
LAST_NAME                      DEPARTMENT_NAME
------------------------------ ------------------
Coleman                        Accounting
Patterson                      Accounting
Hicks                          Accounting
Ashton                         Financial
Lamb                           Financial
                               Human Resource
Dickerson                      Marketing
Fowler                         Marketing
Morley                         Marketing
McKenzie                       Marketing
Cooke                          Production

11 rows selected.
```

8. Select Department that still does not have a supervisor.

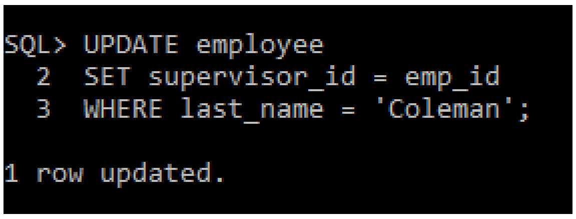
   SELECT DISTINCT department.department_name
   FROM employee
   FULL OUTER JOIN department
   On employee.department_id=department.department_id
   Where employee. supervisor_id IS NULL;



```
SQL> SELECT DISTINCT department.department_name
  2  FROM employee
  3  FULL OUTER JOIN department
  4  On employee.department_id=department.department_id
  5  Where employee. supervisor_id IS NULL;

DEPARTMENT_NAME
------------------
Financial
Accounting
Human Resource
Production
Marketing

SQL>
```

9. Assigning an employee to be the supervisor of a department.
   Assigning the employee called Coleman to be the supervisor of the department he belongs.

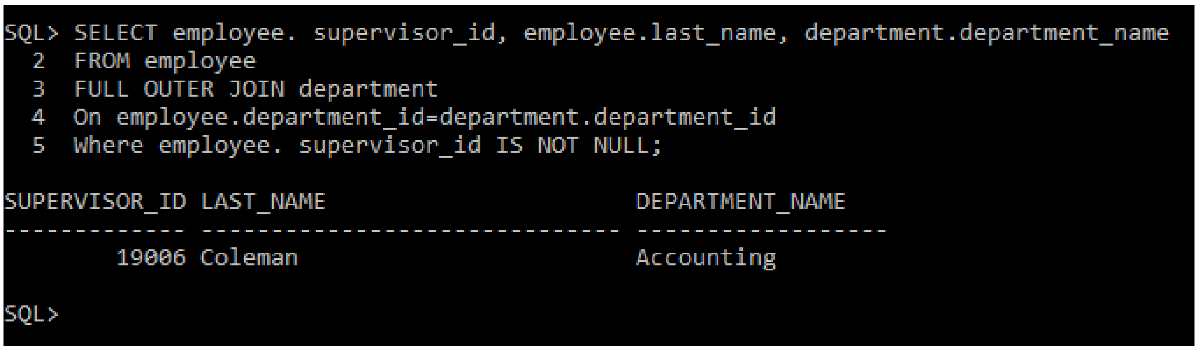
   UPDATE employee
   SET supervisor_id = emp_id
   WHERE last_name = 'Coleman';



```
SQL> UPDATE employee
  2  SET supervisor_id = emp_id
  3  WHERE last_name = 'Coleman';

1 row updated.
```

10. Check the supervisor of each department.

    SELECT employee. supervisor_id, employee.last_name, department.department_name
    FROM employee
    FULL OUTER JOIN department
    On employee.department_id=department.department_id
    Where employee. supervisor_id IS NOT NULL;

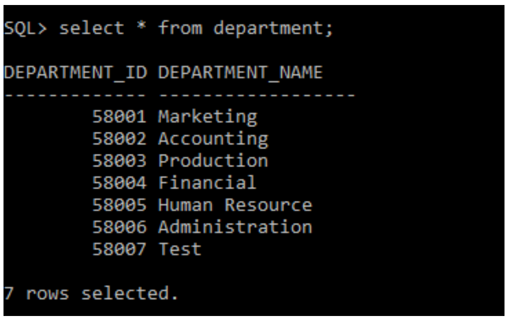


```
SQL> SELECT employee. supervisor_id, employee.last_name, department.department_name
  2  FROM employee
  3  FULL OUTER JOIN department
  4  On employee.department_id=department.department_id
  5  Where employee. supervisor_id IS NOT NULL;

SUPERVISOR_ID LAST_NAME                      DEPARTMENT_NAME
------------- ------------------------------ ------------------
        19006 Coleman                        Accounting

SQL>
```

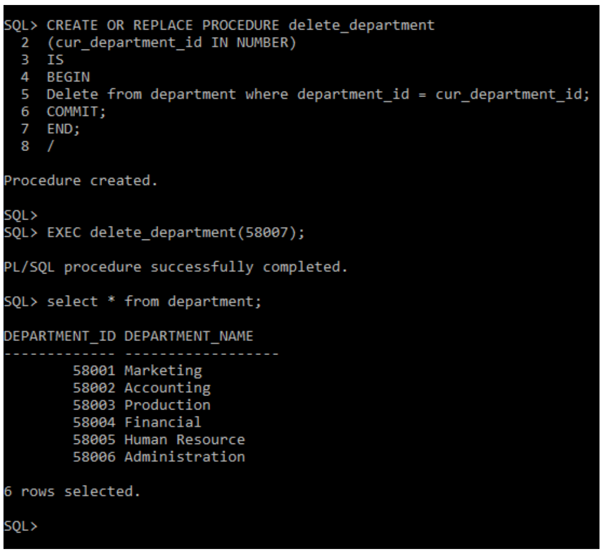
9.2 Stored Procedure

1. Delete Department

```
CREATE OR REPLACE PROCEDURE delete_department
(cur_department_id IN NUMBER)
IS
BEGIN
Delete from department where department_id = cur_department_id;
COMMIT;
END;
/
EXEC delete_department(58007);
```



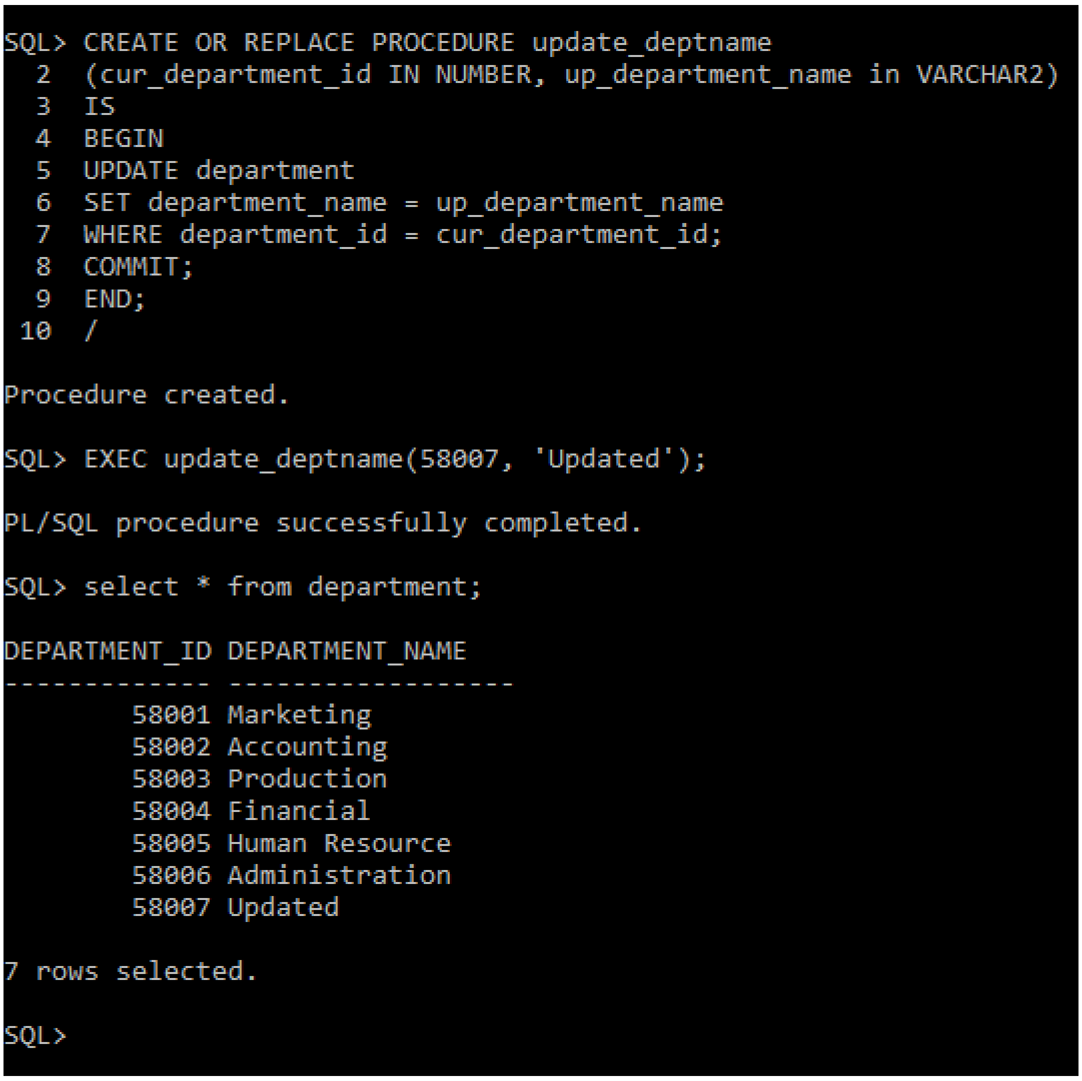
```
SQL> select * from department;

DEPARTMENT_ID DEPARTMENT_NAME
------------- ------------------
        58001 Marketing
        58002 Accounting
        58003 Production
        58004 Financial
        58005 Human Resource
        58006 Administration
        58007 Test

7 rows selected.
```



```
SQL> CREATE OR REPLACE PROCEDURE delete_department
  2  (cur_department_id IN NUMBER)
  3  IS
  4  BEGIN
  5  Delete from department where department_id = cur_department_id;
  6  COMMIT;
  7  END;
  8  /

Procedure created.

SQL>
SQL> EXEC delete_department(58007);

PL/SQL procedure successfully completed.

SQL> select * from department;

DEPARTMENT_ID DEPARTMENT_NAME
------------- -----------------
        58001 Marketing
        58002 Accounting
        58003 Production
        58004 Financial
        58005 Human Resource
        58006 Administration

6 rows selected.

SQL>
```

2. Update Department Name

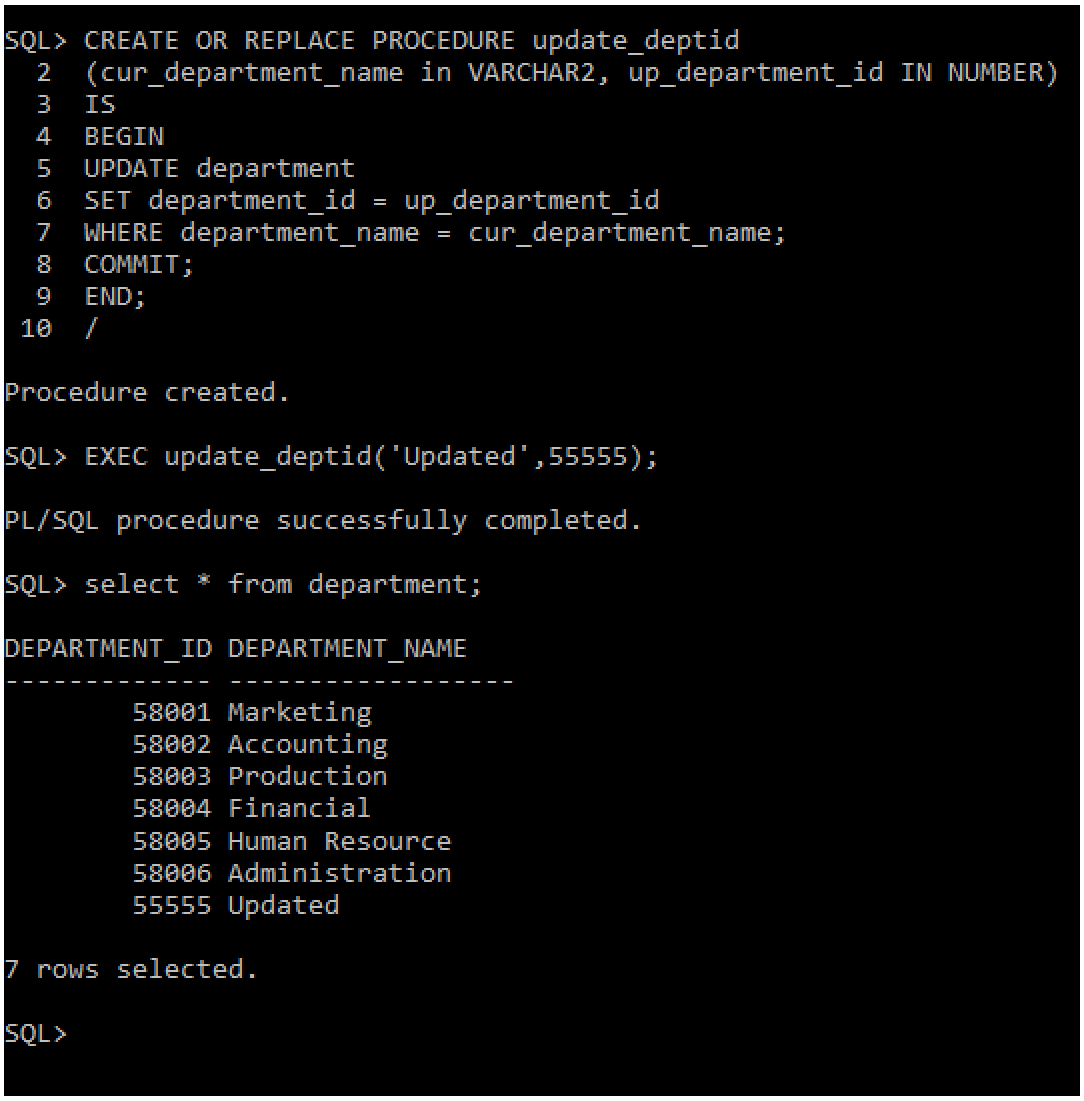
```
CREATE OR REPLACE PROCEDURE update_deptname
(cur_department_id IN NUMBER, up_department_name in VARCHAR2)
IS
BEGIN
UPDATE department
SET department_name = up_department_name
WHERE department_id = cur_department_id;
COMMIT;
END;
/
EXEC update_deptname(58007, 'Updated');
```



```
SQL> CREATE OR REPLACE PROCEDURE update_deptname
  2  (cur_department_id IN NUMBER, up_department_name in VARCHAR2)
  3  IS
  4  BEGIN
  5  UPDATE department
  6  SET department_name = up_department_name
  7  WHERE department_id = cur_department_id;
  8  COMMIT;
  9  END;
 10  /

Procedure created.

SQL> EXEC update_deptname(58007, 'Updated');

PL/SQL procedure successfully completed.

SQL> select * from department;

DEPARTMENT_ID DEPARTMENT_NAME
------------- ------------------
        58001 Marketing
        58002 Accounting
        58003 Production
        58004 Financial
        58005 Human Resource
        58006 Administration
        58007 Updated

7 rows selected.

SQL>
```

3. Update Department ID

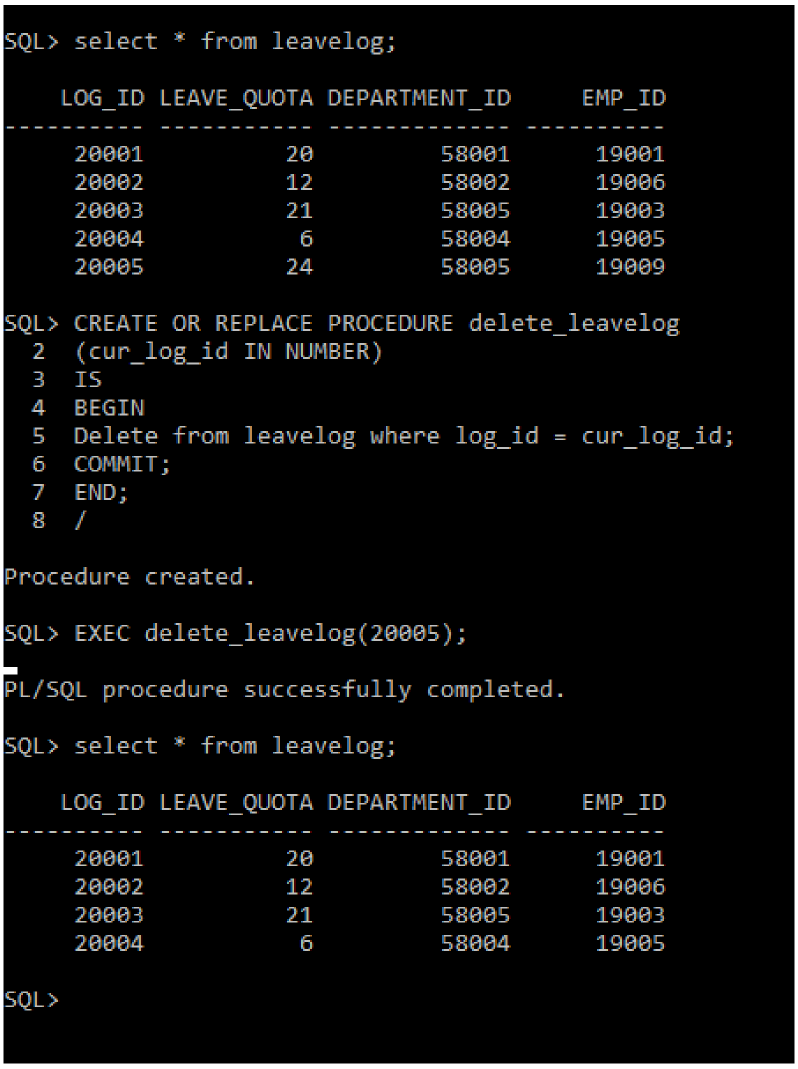
```
CREATE OR REPLACE PROCEDURE update_deptid
(cur_department_name in VARCHAR2, up_department_id IN NUMBER)
IS
BEGIN
UPDATE department
SET department_id = up_department_id
WHERE department_name = cur_department_name;
COMMIT;
END;
/

EXEC update_deptid('Updated',55555);
```



```
SQL> CREATE OR REPLACE PROCEDURE update_deptid
(cur_department_name in VARCHAR2, up_department_id IN NUMBER)
IS
BEGIN
UPDATE department
SET department_id = up_department_id
WHERE department_name = cur_department_name;
COMMIT;
END;
/

Procedure created.

SQL> EXEC update_deptid('Updated',55555);

PL/SQL procedure successfully completed.

SQL> select * from department;

DEPARTMENT_ID DEPARTMENT_NAME
------------- ------------------
        58001 Marketing
        58002 Accounting
        58003 Production
        58004 Financial
        58005 Human Resource
        58006 Administration
        55555 Updated

7 rows selected.

SQL>
```

4. Delete leave log

```
CREATE OR REPLACE PROCEDURE delete_leavelog

(cur_log_id IN NUMBER)

IS

BEGIN

Delete from leavelog where log_id = cur_log_id;

COMMIT;

END;

/

EXEC delete_leavelog(20005);
```

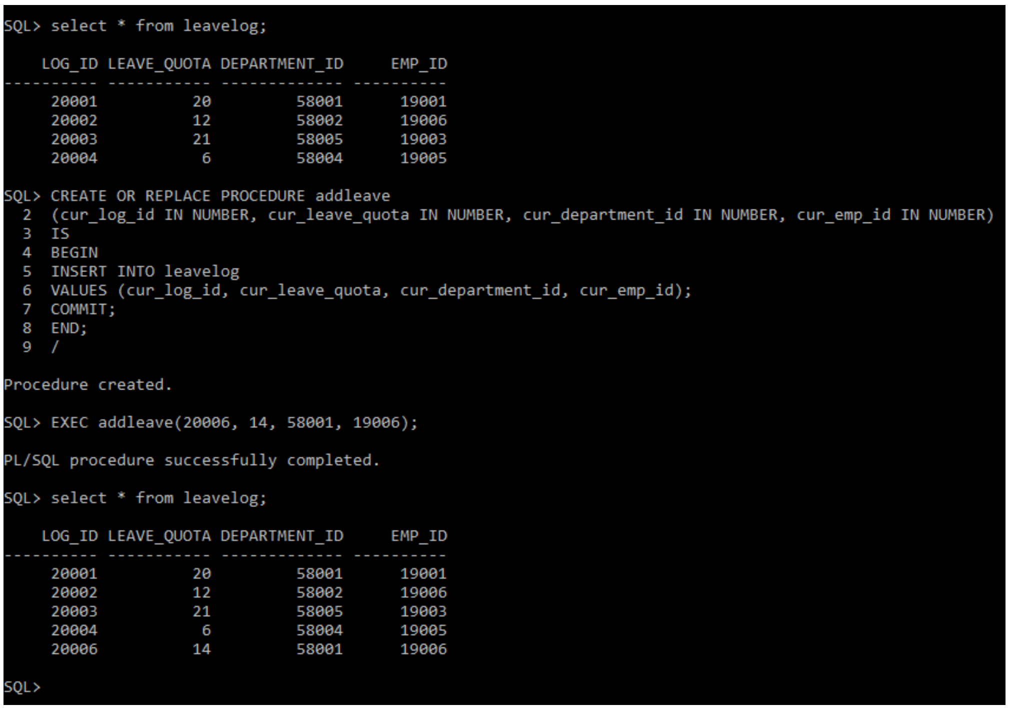


```
SQL> select * from leavelog;

    LOG_ID LEAVE_QUOTA DEPARTMENT_ID     EMP_ID
---------- ----------- ------------- ----------
     20001          20         58001      19001
     20002          12         58002      19006
     20003          21         58005      19003
     20004           6         58004      19005
     20005          24         58005      19009

SQL> CREATE OR REPLACE PROCEDURE delete_leavelog
  2  (cur_log_id IN NUMBER)
  3  IS
  4  BEGIN
  5  Delete from leavelog where log_id = cur_log_id;
  6  COMMIT;
  7  END;
  8  /

Procedure created.

SQL> EXEC delete_leavelog(20005);

PL/SQL procedure successfully completed.

SQL> select * from leavelog;

    LOG_ID LEAVE_QUOTA DEPARTMENT_ID     EMP_ID
---------- ----------- ------------- ----------
     20001          20         58001      19001
     20002          12         58002      19006
     20003          21         58005      19003
     20004           6         58004      19005

SQL>
```

5. Add Leave Log

```
CREATE OR REPLACE PROCEDURE addleave
(cur_log_id IN NUMBER, cur_leave_quota IN NUMBER, cur_department_id IN
NUMBER, cur_emp_id IN NUMBER)
IS
BEGIN
INSERT INTO leavelog
VALUES (cur_log_id, cur_leave_quota, cur_department_id, cur_emp_id);
COMMIT;
END;
/
```

```
EXEC addleave(20006, 14, 58001, 19006);
```

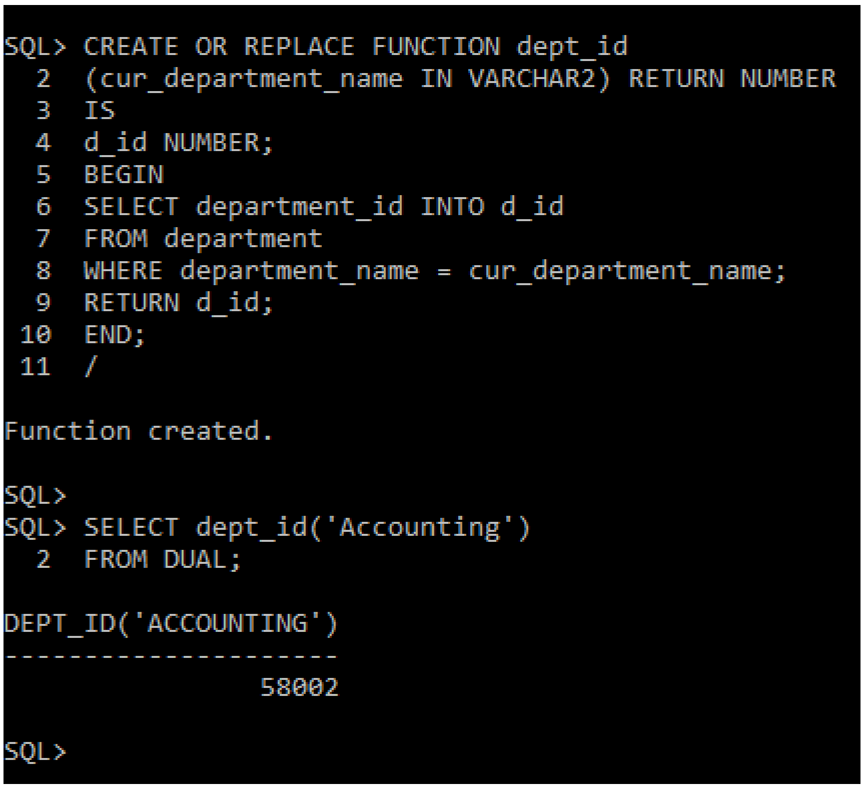


```
SQL> select * from leavelog;

    LOG_ID LEAVE_QUOTA DEPARTMENT_ID     EMP_ID
---------- ----------- ------------- ----------
     20001          20         58001      19001
     20002          12         58002      19006
     20003          21         58005      19003
     20004           6         58004      19005

SQL> CREATE OR REPLACE PROCEDURE addleave
  2  (cur_log_id IN NUMBER, cur_leave_quota IN NUMBER, cur_department_id IN NUMBER, cur_emp_id IN NUMBER)
  3  IS
  4  BEGIN
  5  INSERT INTO leavelog
  6  VALUES (cur_log_id, cur_leave_quota, cur_department_id, cur_emp_id);
  7  COMMIT;
  8  END;
  9  /

Procedure created.

SQL> EXEC addleave(20006, 14, 58001, 19006);

PL/SQL procedure successfully completed.

SQL> select * from leavelog;

    LOG_ID LEAVE_QUOTA DEPARTMENT_ID     EMP_ID
---------- ----------- ------------- ----------
     20001          20         58001      19001
     20002          12         58002      19006
     20003          21         58005      19003
     20004           6         58004      19005
     20006          14         58001      19006

SQL>
```

9.3 Functions

Function 1 : Return Department id with department name.

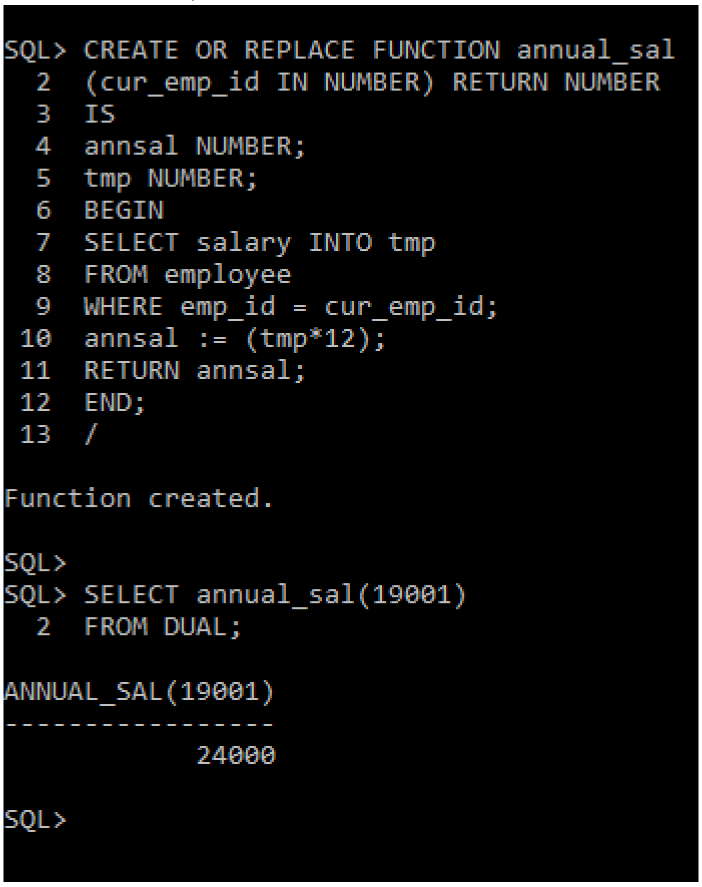
```
CREATE OR REPLACE FUNCTION dept_id
(cur_department_name IN VARCHAR2) RETURN NUMBER
IS
d_id NUMBER;
BEGIN
SELECT department_id INTO d_id
FROM department
WHERE department_name = cur_department_name;
RETURN d_id;
END;
/

SELECT dept_id('Accounting')
FROM DUAL;
```



```
SQL> CREATE OR REPLACE FUNCTION dept_id
  2  (cur_department_name IN VARCHAR2) RETURN NUMBER
  3  IS
  4  d_id NUMBER;
  5  BEGIN
  6  SELECT department_id INTO d_id
  7  FROM department
  8  WHERE department_name = cur_department_name;
  9  RETURN d_id;
 10  END;
 11  /

Function created.

SQL>
SQL> SELECT dept_id('Accounting')
  2  FROM DUAL;

DEPT_ID('ACCOUNTING')
---------------------
                58002

SQL>
```

Function 2 : Calculate Annual Salary

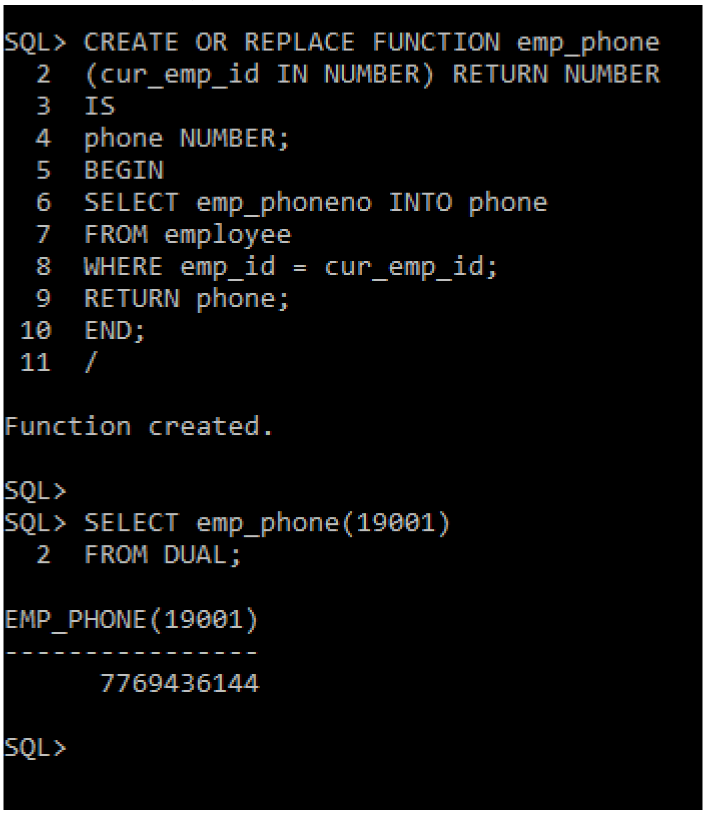
```
CREATE OR REPLACE FUNCTION annual_sal
(cur_emp_id IN NUMBER) RETURN NUMBER
IS
annsal NUMBER;
tmp NUMBER;
BEGIN
SELECT salary INTO tmp
FROM employee
WHERE emp_id = cur_emp_id;
annsal := (tmp*12);
RETURN annsal;
END;
/

SELECT annual_sal(19001)
FROM DUAL;
```



```
SQL> CREATE OR REPLACE FUNCTION annual_sal
  2  (cur_emp_id IN NUMBER) RETURN NUMBER
  3  IS
  4  annsal NUMBER;
  5  tmp NUMBER;
  6  BEGIN
  7  SELECT salary INTO tmp
  8  FROM employee
  9  WHERE emp_id = cur_emp_id;
 10  annsal := (tmp*12);
 11  RETURN annsal;
 12  END;
 13  /

Function created.

SQL>
SQL> SELECT annual_sal(19001)
  2  FROM DUAL;

ANNUAL_SAL(19001)
-----------------
            24000

SQL>
```

Function 3 : Retrieve employee's phone number

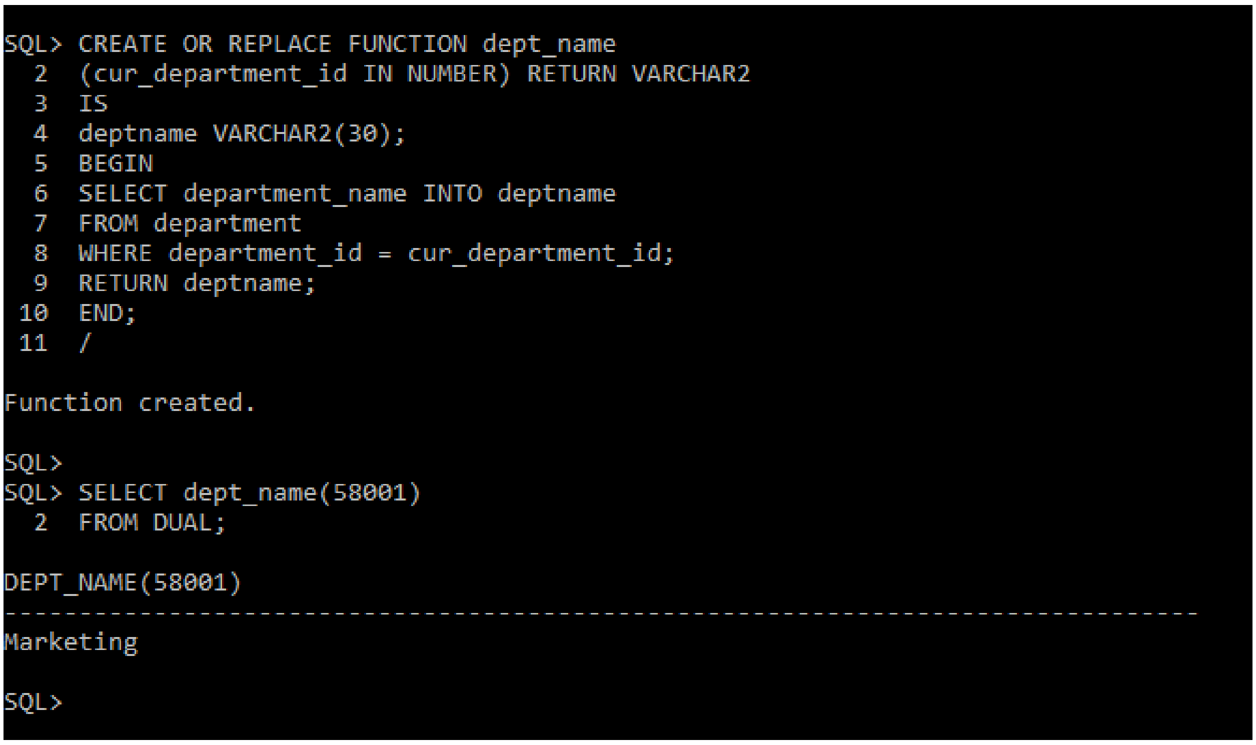
```
CREATE OR REPLACE FUNCTION emp_phone
(cur_emp_id IN NUMBER) RETURN NUMBER
IS
phone NUMBER;
BEGIN
SELECT emp_phoneno INTO phone
FROM employee
WHERE emp_id = cur_emp_id;
RETURN phone;
END;
/

SELECT emp_phone(19001)
FROM DUAL;
```



```
SQL> CREATE OR REPLACE FUNCTION emp_phone
  2  (cur_emp_id IN NUMBER) RETURN NUMBER
  3  IS
  4  phone NUMBER;
  5  BEGIN
  6  SELECT emp_phoneno INTO phone
  7  FROM employee
  8  WHERE emp_id = cur_emp_id;
  9  RETURN phone;
 10  END;
 11  /

Function created.

SQL>
SQL> SELECT emp_phone(19001)
  2  FROM DUAL;

EMP_PHONE(19001)
----------------
      7769436144

SQL>
```

Function 4 : Get Department name with department id

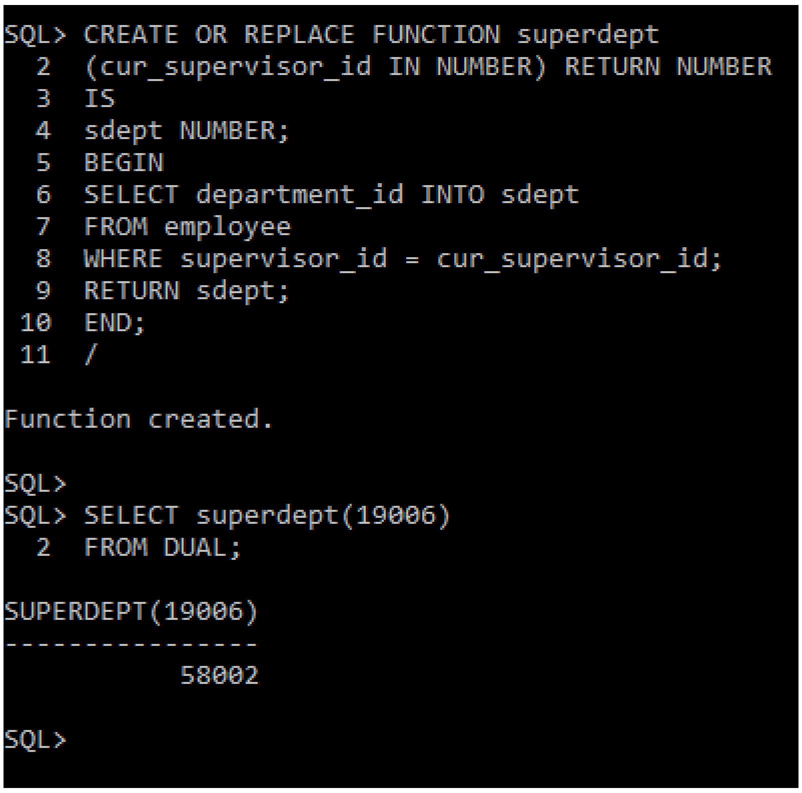
```
CREATE OR REPLACE FUNCTION dept_name
(cur_department_id IN NUMBER) RETURN VARCHAR2
IS
deptname VARCHAR2(30);
BEGIN
SELECT department_name INTO deptname
FROM department
WHERE department_id = cur_department_id;
RETURN deptname;
END;
/

SELECT dept_name(58001)
FROM DUAL;
```



```
SQL> CREATE OR REPLACE FUNCTION dept_name
  2  (cur_department_id IN NUMBER) RETURN VARCHAR2
  3  IS
  4  deptname VARCHAR2(30);
  5  BEGIN
  6  SELECT department_name INTO deptname
  7  FROM department
  8  WHERE department_id = cur_department_id;
  9  RETURN deptname;
 10  END;
 11  /

Function created.

SQL>
SQL> SELECT dept_name(58001)
  2  FROM DUAL;

DEPT_NAME(58001)
--------------------------------------------------------------------------------
Marketing

SQL>
```

Function 5 : Check the department of the supervisor

```
CREATE OR REPLACE FUNCTION superdept
(cur_supervisor_id IN NUMBER) RETURN NUMBER
IS
sdept NUMBER;
BEGIN
SELECT department_id INTO sdept
FROM employee
WHERE supervisor_id = cur_supervisor_id;
RETURN sdept;
END;
/

SELECT superdept(19006)
FROM DUAL;
```



```
SQL> CREATE OR REPLACE FUNCTION superdept
  2  (cur_supervisor_id IN NUMBER) RETURN NUMBER
  3  IS
  4  sdept NUMBER;
  5  BEGIN
  6  SELECT department_id INTO sdept
  7  FROM employee
  8  WHERE supervisor_id = cur_supervisor_id;
  9  RETURN sdept;
 10  END;
 11  /

Function created.

SQL>
SQL> SELECT superdept(19006)
  2  FROM DUAL;

SUPERDEPT(19006)
----------------
           58002

SQL>
```

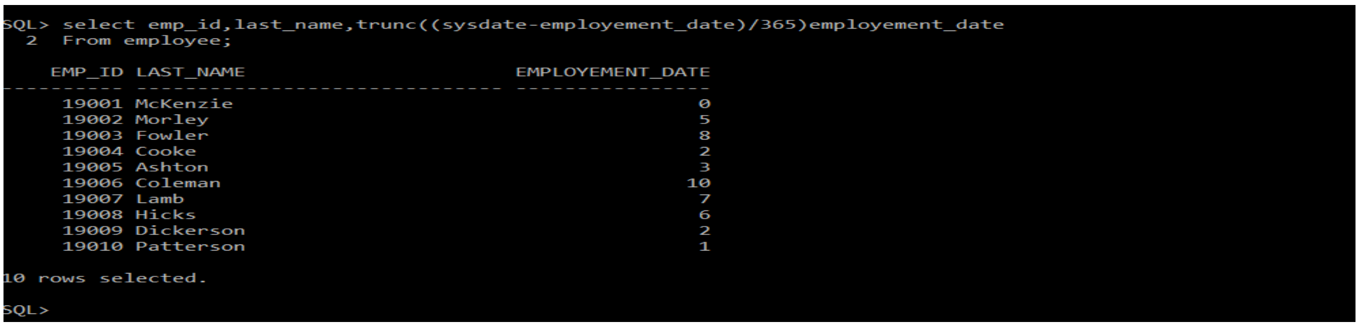
10.0 Individual Assessment (Hwang Jia Min, 1900242)

10.1 Queries

1. Show year of employee worked

   select emp_id,last_name,trunc((sysdate-employement_date)/365)employement_date

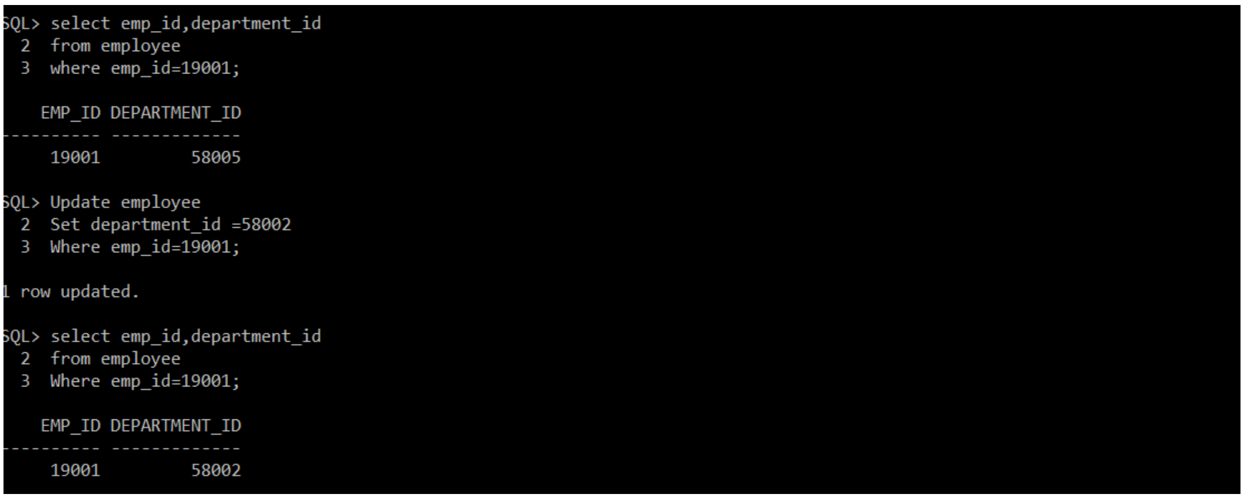
   From employee;



```
SQL> select emp_id,last_name,trunc((sysdate-employement_date)/365)employement_date
  2  From employee;

    EMP_ID LAST_NAME                      EMPLOYEMENT_DATE
---------- ------------------------------ ----------------
     19001 McKenzie                                      0
     19002 Morley                                        5
     19003 Fowler                                        8
     19004 Cooke                                         2
     19005 Ashton                                        3
     19006 Coleman                                      10
     19007 Lamb                                          7
     19008 Hicks                                         6
     19009 Dickerson                                     2
     19010 Patterson                                     1

10 rows selected.

SQL>
```

2. Shift employee to other department

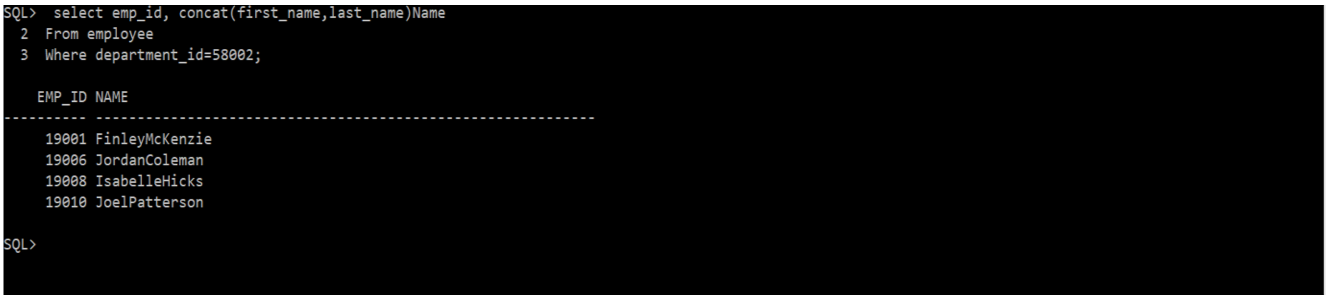
Update employee
Set department_id =58002
Where emp_id=19001;



```
SQL> select emp_id,department_id
  2  from employee
  3  where emp_id=19001;

    EMP_ID DEPARTMENT_ID
---------- -------------
     19001         58005

SQL> Update employee
  2  Set department_id =58002
  3  Where emp_id=19001;

1 row updated.

SQL> select emp_id,department_id
  2  from employee
  3  Where emp_id=19001;

    EMP_ID DEPARTMENT_ID
---------- -------------
     19001         58002
```

3. Show employees in a department

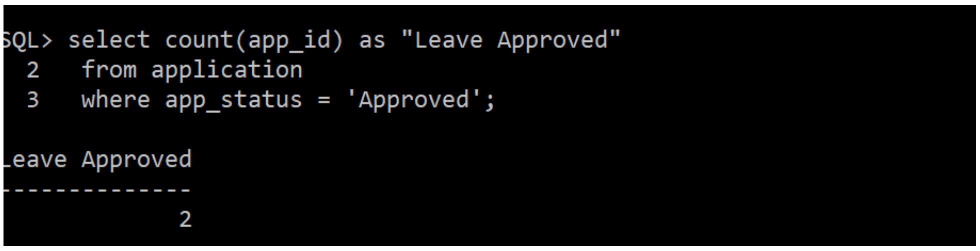
   select emp_id, concat(first_name,last_name)Name
   From employee
   Where department_id=58002;



```
SQL>  select emp_id, concat(first_name,last_name)Name
  2  From employee
  3  Where department_id=58002;

    EMP_ID NAME
---------- --------------------------------------------------------
     19001 FinleyMcKenzie
     19006 JordanColeman
     19008 IsabelleHicks
     19010 JoelPatterson

SQL>
```

4. Show number of leave that are approved

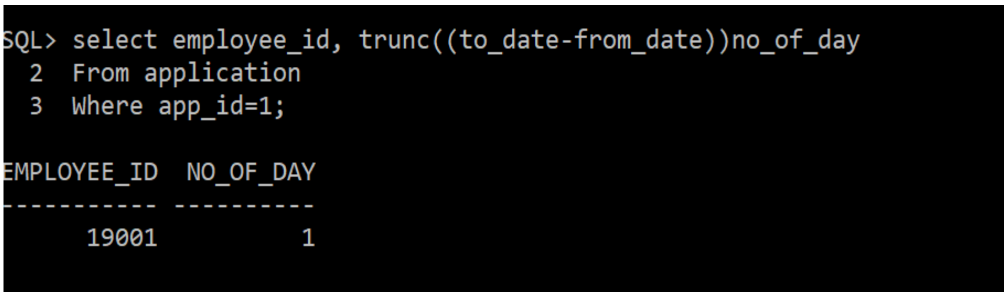
   select count(app_id) as "Leave Approved"
    from application
    where app_status = 'Approved';



```
SQL> select count(app_id) as "Leave Approved"
  2   from application
  3   where app_status = 'Approved';

Leave Approved
--------------
             2
```

5. Show duration of leave taken by an employee

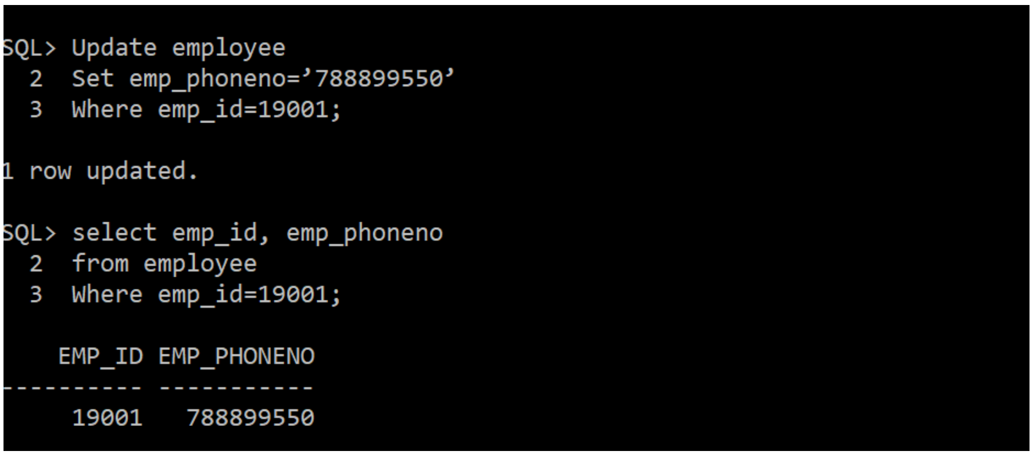
   select employee_id, trunc((to_date-from_date))no_of_day
   From application
   Where app_id=1;



```
SQL> select employee_id, trunc((to_date-from_date))no_of_day
  2  From application
  3  Where app_id=1;

EMPLOYEE_ID  NO_OF_DAY
----------- ----------
      19001          1
```

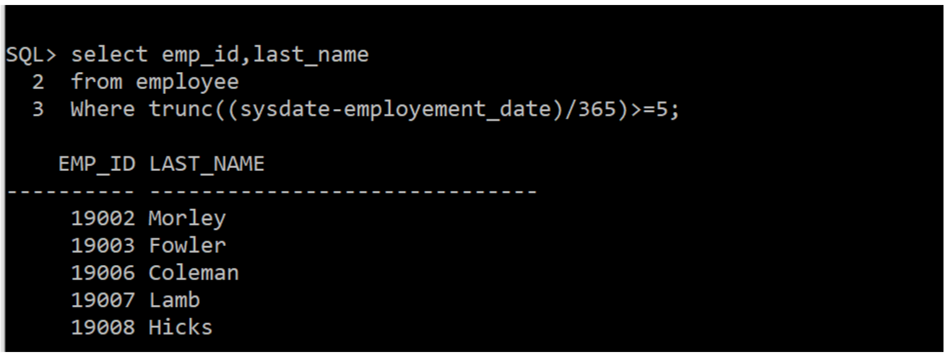
6. Update the information of an employee

Update employee
Set emp_phoneno='788899550'
Where emp_id=19001;


```
SQL> Update employee
  2  Set emp_phoneno='788899550'
  3  Where emp_id=19001;

1 row updated.

SQL> select emp_id, emp_phoneno
  2  from employee
  3  Where emp_id=19001;

    EMP_ID EMP_PHONENO
---------- -----------
     19001   788899550
```

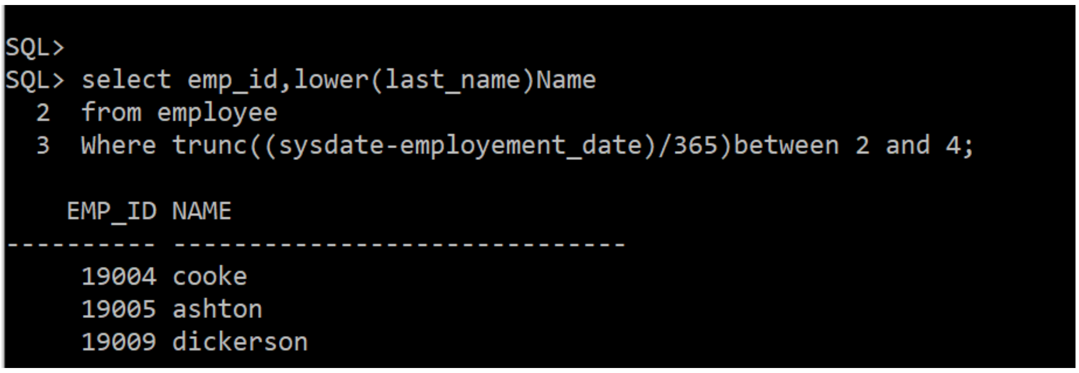
7. Show name of employee that worked more than 5 years

   select emp_id,last_name
   from employee
   Where trunc((sysdate-employement_date)/365)>=5;


```
SQL> select emp_id,last_name
  2  from employee
  3  Where trunc((sysdate-employement_date)/365)>=5;

    EMP_ID LAST_NAME
---------- ------------------------------
     19002 Morley
     19003 Fowler
     19006 Coleman
     19007 Lamb
     19008 Hicks
```

8. Show name of employee that age between 2 years old and 4 years old

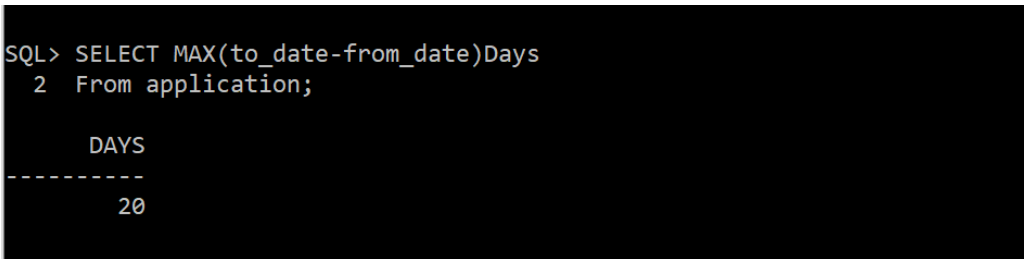
   select emp_id,lower(last_name)Name
   from employee
   Where trunc((sysdate-employement_date)/365)between 2 and 4;


```
SQL>
SQL> select emp_id,lower(last_name)Name
  2  from employee
  3  Where trunc((sysdate-employement_date)/365)between 2 and 4;

    EMP_ID NAME
---------- ------------------------------
     19004 cooke
     19005 ashton
     19009 dickerson
```

9. Show maximum days of leave taken by the employee

SELECT MAX(to_date-from_date)Days

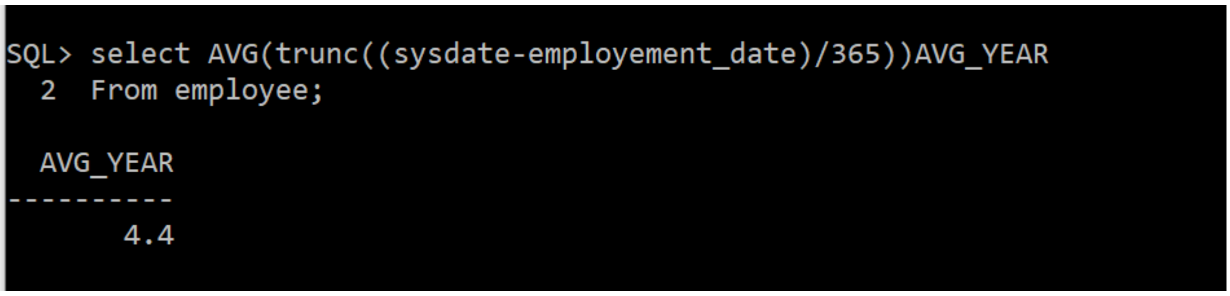
From application;



```
SQL> SELECT MAX(to_date-from_date)Days
  2  From application;

      DAYS
----------
        20
```

10. Show average year of employee worked

select AVG(trunc((sysdate-employement_date)/365))AVG_YEAR
From employee;



```
SQL> select AVG(trunc((sysdate-employement_date)/365))AVG_YEAR
  2  From employee;

  AVG_YEAR
----------
       4.4
```

10.2 Stored Procedure

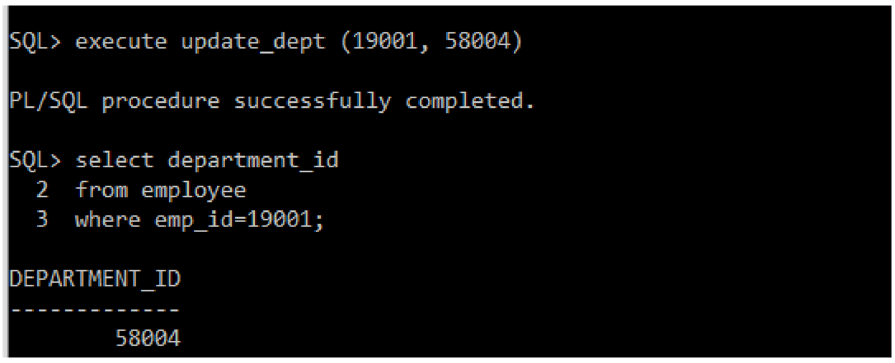
1. Update_Employee Department
   A. Explanation

      Execute this procedure to update the employee address of a specific record in table Employees. This parameter only accepts NUMBER.

   B. Code

```
CREATE  OR REPLACE PROCEDURE update_dept
   (selected_emp_id in NUMBER, new_department_id in NUMBER)
   AS
   BEGIN
  UPDATE EMPLOYEE
  set department_id = new_department_id
 where emp_id = selected_emp_id;
  COMMIT;
  END;
 /
```

   C. Result



```
SQL> execute update_dept (19001, 58004)

PL/SQL procedure successfully completed.

SQL> select department_id
  2  from employee
  3  where emp_id=19001;

DEPARTMENT_ID
-------------
        58004
```

2. Update_Employee address
   A. Explanation

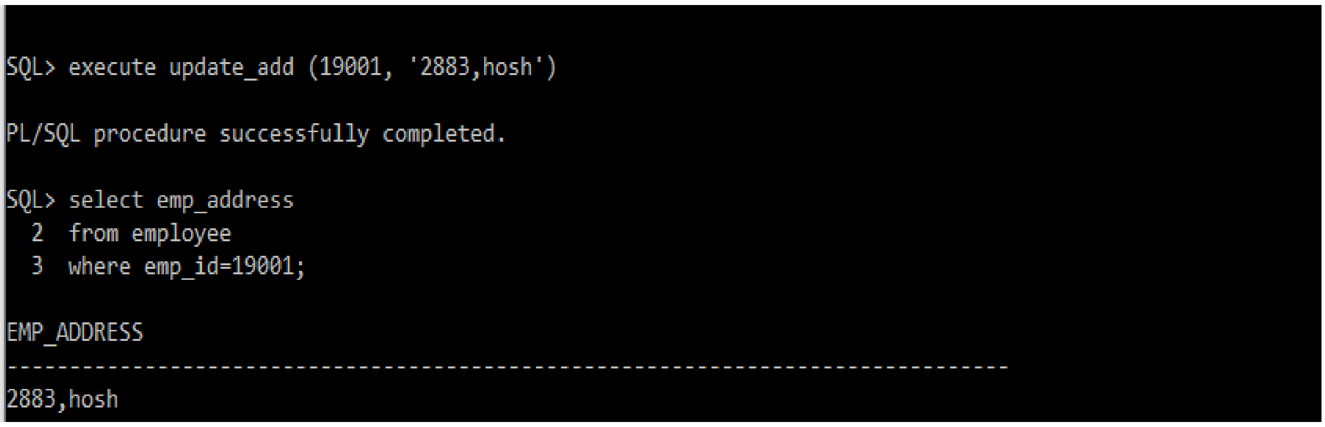
      Execute this procedure to update position of employees of a specific record in table Employee. This parameter only accepts VARCHAR2.

   B. Code

```
 CREATE OR REPLACE PROCEDURE update_add
 ( selected_emp_id IN NUMBER, new_emp_address IN VARCHAR2)
 AS
```

```
BEGIN
UPDATE EMPLOYEE
 SET emp_address = new_emp_address
 where emp_id = selected_emp_id;
  COMMIT;
  END;
  /
```

C. Result



```
SQL> execute update_add (19001, '2883,hosh')

PL/SQL procedure successfully completed.

SQL> select emp_address
  2  from employee
  3  where emp_id=19001;

EMP_ADDRESS
--------------------------------------------------------------------------------
2883,hosh
```

3. Update_Employee Phone Number

A. Explanation

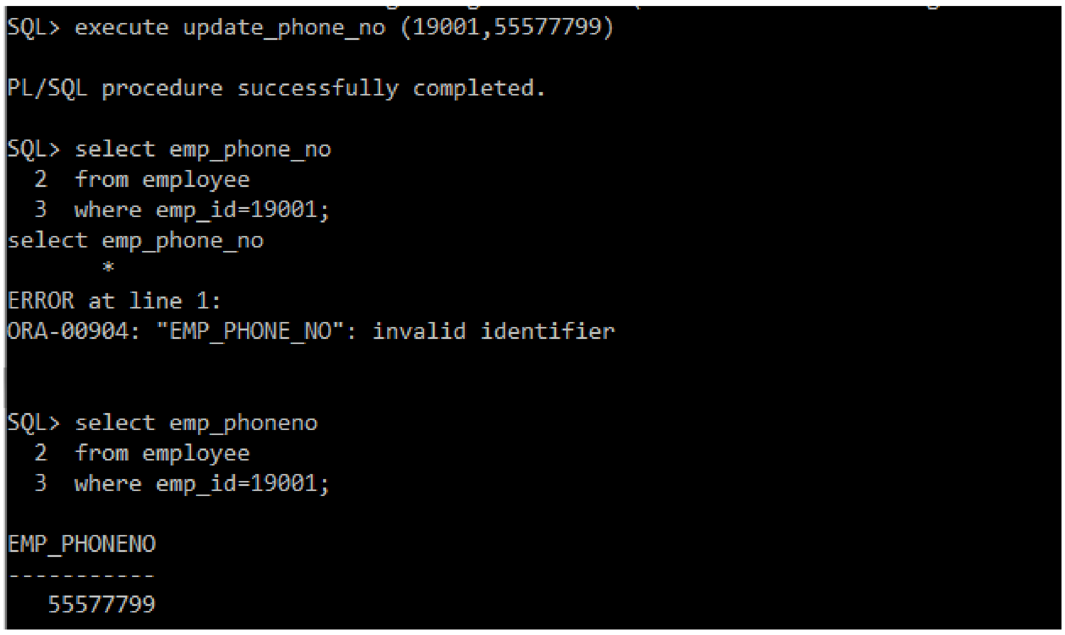
This procedure is used to update the phone number of employee . This parameter only accepts NUMBER.

B. Code

```
CREATE OR REPLACE PROCEDURE update_phone_no
( selected_emp_id IN NUMBER, new_emp_phoneno IN NUMBER)
AS
BEGIN
UPDATE EMPLOYEE
```

```
SET emp_phoneno = new_emp_phoneno
where emp_id = selected_emp_id;
 COMMIT;
 END;
 /
```

C. Result



```
SQL> execute update_phone_no (19001,55577799)

PL/SQL procedure successfully completed.

SQL> select emp_phone_no
  2  from employee
  3  where emp_id=19001;
select emp_phone_no
       *
ERROR at line 1:
ORA-00904: "EMP_PHONE_NO": invalid identifier


SQL> select emp_phoneno
  2  from employee
  3  where emp_id=19001;

EMP_PHONENO
-----------
   55577799
```

4. <u>update_salary</u>

A. Explantion

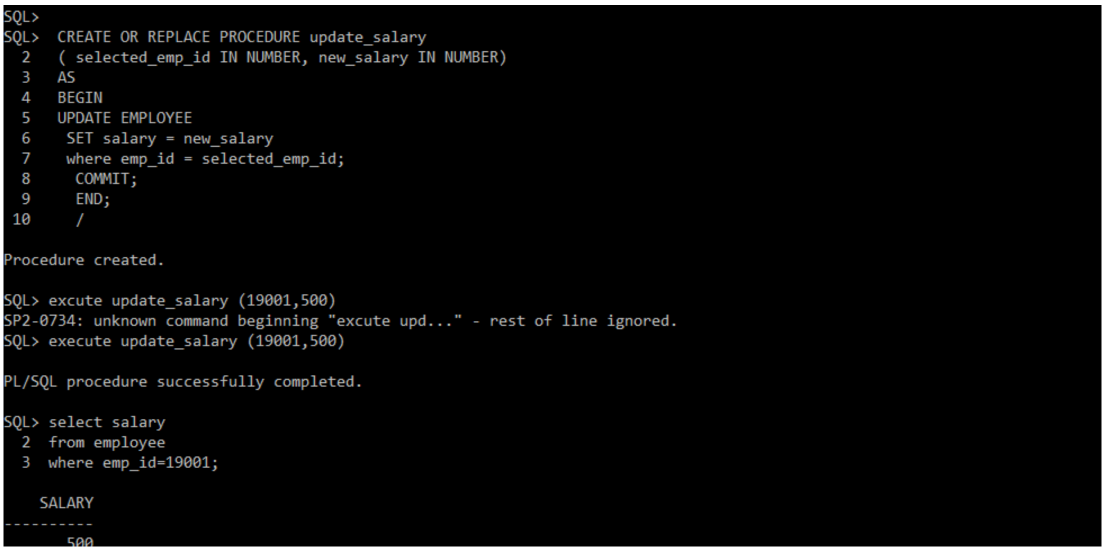
This procedure is used to update the salary of the employee and this parameter only accept NUMBER.

B. Code

```
CREATE OR REPLACE PROCEDURE update_salary
( selected_emp_id IN NUMBER, new_salary IN NUMBER)
AS
BEGIN
UPDATE EMPLOYEE
 SET salary = new_salary
 where emp_id = selected_emp_id;
  COMMIT;
  END;
```

```
/
```

C. Result



```
SQL>
SQL>  CREATE OR REPLACE PROCEDURE update_salary
      ( selected_emp_id IN NUMBER, new_salary IN NUMBER)
      AS
      BEGIN
      UPDATE EMPLOYEE
       SET salary = new_salary
       where emp_id = selected_emp_id;
        COMMIT;
        END;
        /

Procedure created.

SQL> excute update_salary (19001,500)
SP2-0734: unknown command beginning "excute upd..." - rest of line ignored.
SQL> execute update_salary (19001,500)

PL/SQL procedure successfully completed.

SQL> select salary
     from employee
     where emp_id=19001;

    SALARY
----------
       500
```

5. Update employee email

A. Explanation

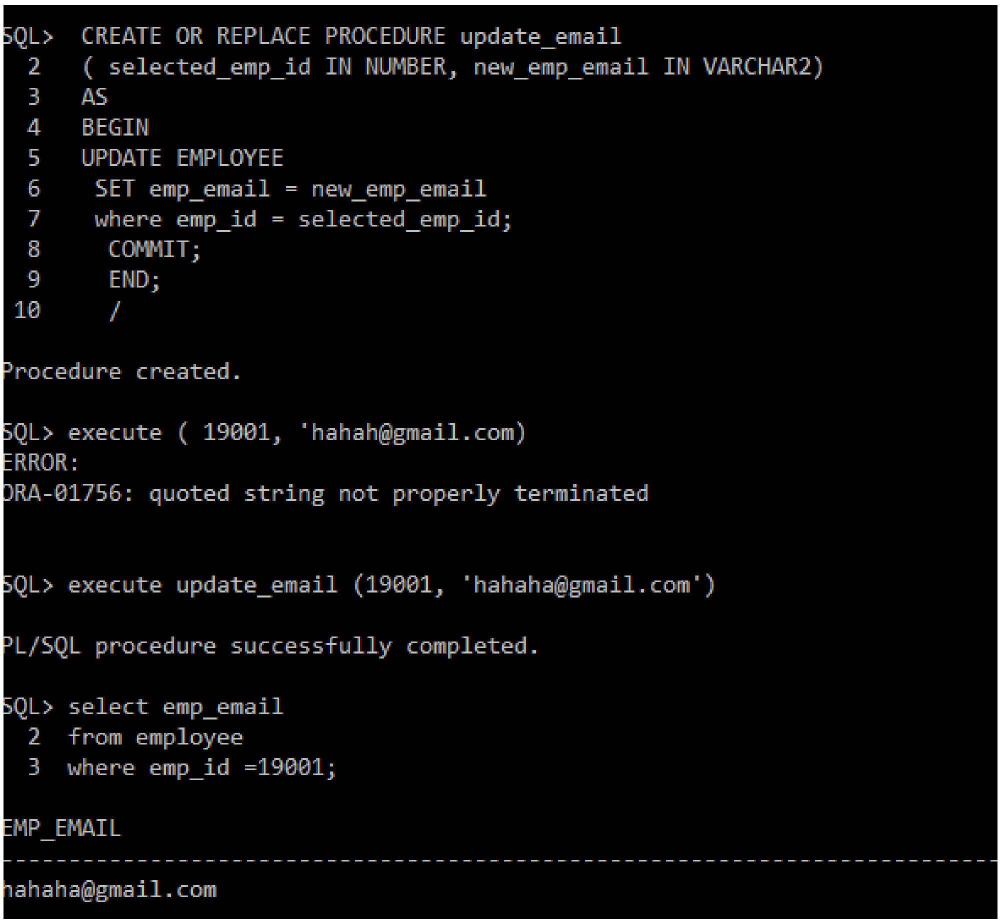
Execute this procedure to update employee’s new department of a specific record in table Employees. This parameter only accepts VARCHAR2.

B. Code

```
CREATE OR REPLACE PROCEDURE update_email
( selected_emp_id IN NUMBER, new_emp_email IN VARCHAR2)
AS
BEGIN
UPDATE EMPLOYEE
 SET emp_email = new_emp_email
 where emp_id = selected_emp_id;
  COMMIT;
  END;
  /
```

Execute update_email (19001,’hahaha@gmail.com)

C. Result



```
SQL>  CREATE OR REPLACE PROCEDURE update_email
  ( selected_emp_id IN NUMBER, new_emp_email IN VARCHAR2)
  AS
  BEGIN
  UPDATE EMPLOYEE
   SET emp_email = new_emp_email
   where emp_id = selected_emp_id;
    COMMIT;
    END;
    /

Procedure created.

SQL> execute ( 19001, 'hahah@gmail.com)
ERROR:
ORA-01756: quoted string not properly terminated


SQL> execute update_email (19001, 'hahaha@gmail.com')

PL/SQL procedure successfully completed.

SQL> select emp_email
  2  from employee
  3  where emp_id =19001;

EMP_EMAIL
--------------------------------------------------------------------------------
hahaha@gmail.com
```

10.3 Functions

1. Total department Number

A. Explantaion

User can this function yo count the total number of department in a company.

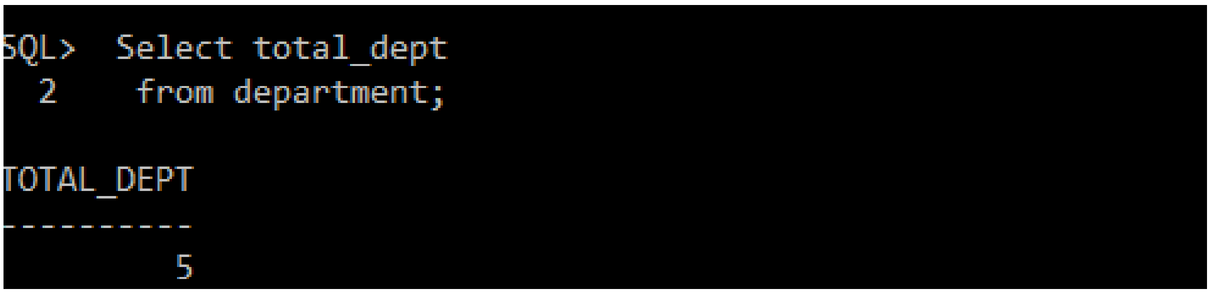
B. Code

```
CREATE OR REPLACE FUNCTION total_dept
 RETURN NUMBER
 IS
 Total NUMBER (2) :=0;
 BEGIN
SELECT count(*) into total
 FROM department;
 return total;
 END;
 /
```

C. Run

```
Select total_dept
 from dual;
```

D. Result



```
SQL>  Select total_dept
  2    from department;

TOTAL_DEPT
----------
         5
```

2. Show duration of leave taken by employee

   A. Explantaion

      User can use this function to determine the day of leave taken by employee

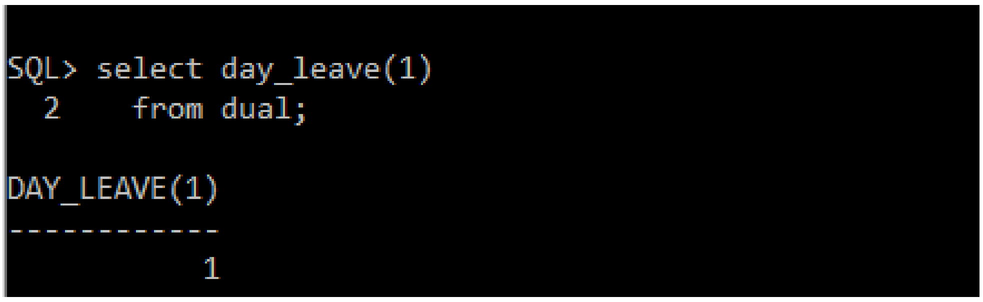
   B. Code

```
CREATE OR REPLACE FUNCTION day_leave
(cur_app_id IN NUMBER) RETURN NUMBER
IS
leave NUMBER;
BEGIN
SELECT TRUNC((to_date-from_date)) INTO leave
FROM application
WHERE application.app_id = cur_app_id;
RETURN leave;
END;
/
```

   C. Run

```
SELECT day_leave(1)
FROM dual;
```

   D. Result



```
SQL> select day_leave(1)
  2    from dual;

DAY_LEAVE(1)
------------
           1
```

3. Total number employee

   A. Explanation

      This function is used to determine to total number of employee

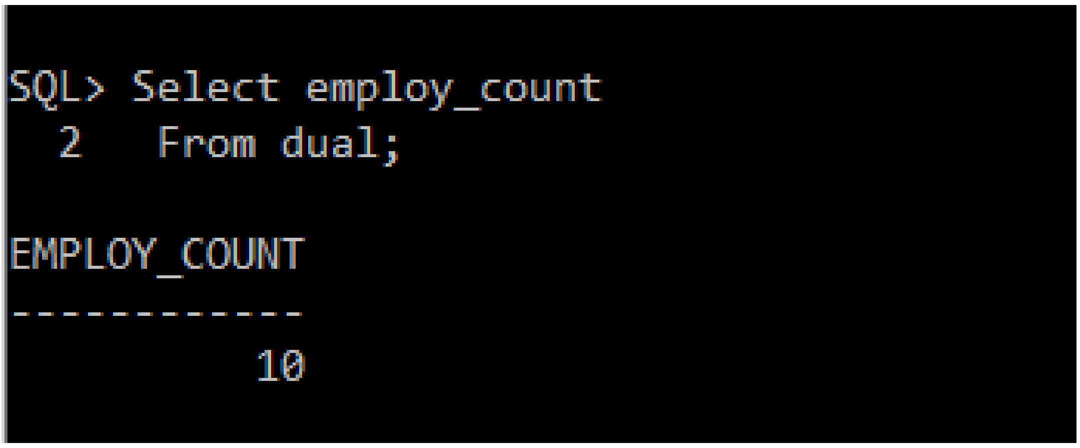
   B. Code

```
CREATE OR REPLACE FUNCTION employ_count
  RETURN NUMBER
  IS
 Total NUMBER (2) :=0;
  BEGIN
  SELECT count(*) into total
  FROM employee;
  return total;
  END;
  /
```

   C. Run

```
Select employee_count
 From dual;
```

   D. Result



```
SQL> Select employ_count
  2   From dual;

EMPLOY_COUNT
------------
          10
```

4. Total Number Of Application

   A. Explanation

      User can use this function to get the total numbers of application took by employee.

   B. Code

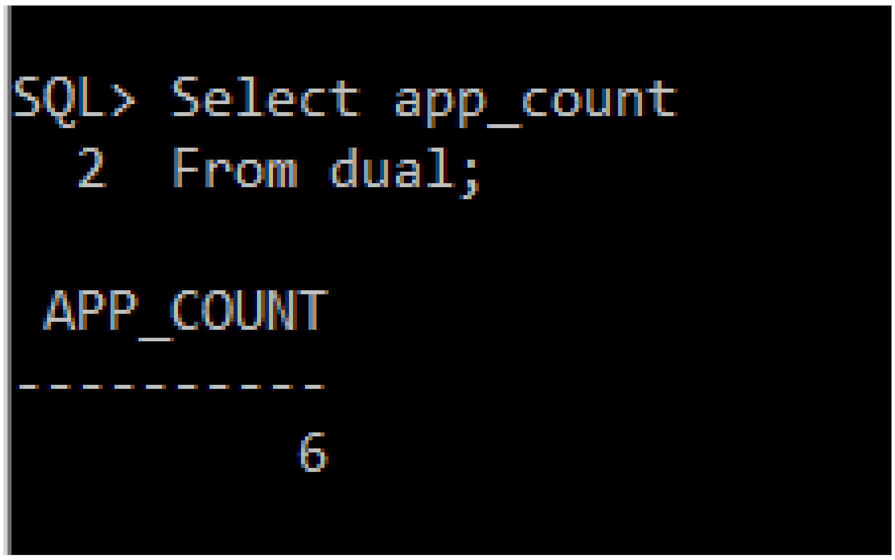
```
CREATE OR REPLACE FUNCTION App_count
```

```
RETURN NUMBER
 IS
 Total NUMBER (2) :=0;
BEGIN
 SELECT count(*) into total
 FROM application;
  return total;
 END;
  /
```

C. Run

```
Select app_count
From dual;
```

D. Result



```
SQL> Select app_count
  2  From dual;

 APP_COUNT
----------
         6
```

5. Employee Name

   A. Explanation

      User can use this function to get the name of employee by key in the id of employee in the database.

   B. Code

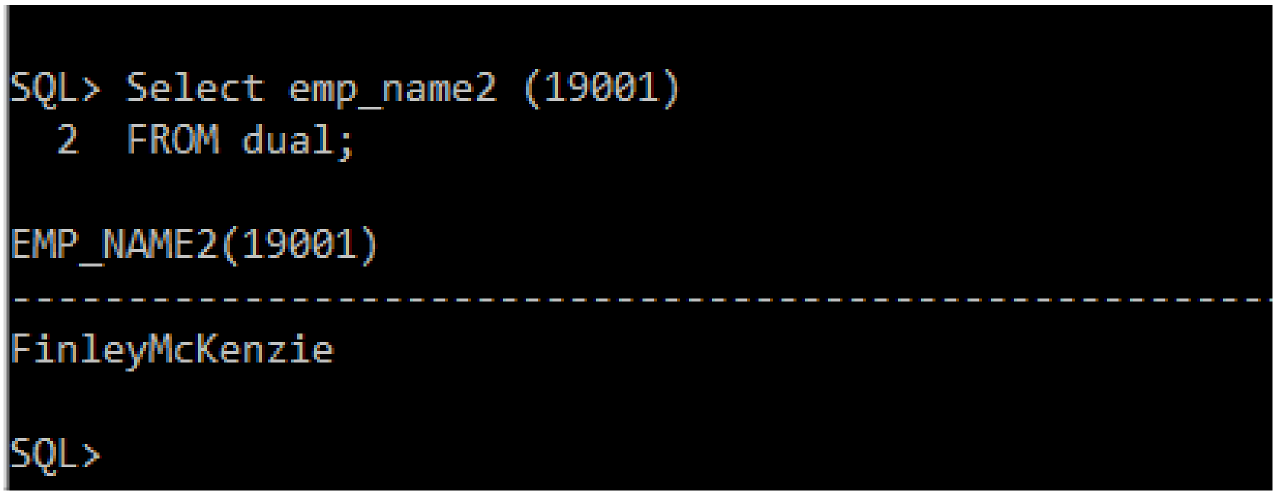
```
 CREATE OR REPLACE FUNCTION emp_name2
(cur_emp_id IN NUMBER) RETURN VARCHAR2
```

```
IS
Emp VARCHAR2 (30) ;
BEGIN
SELECT concat(first_name,last_name)INTO emp
FROM employee
WHERE emp_id = cur_emp_id;
RETURN emp;
END;
/
```

C. Run

```
Select emp_name2(19001)
 From dual;
```

D. Result



```
SQL> Select emp_name2 (19001)
  2  FROM dual;

EMP_NAME2(19001)
-------------------------------------------------------
FinleyMcKenzie

SQL>
```