**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 | Lam Kean Chin | 17ACB05143 | P (7) | CS | signature.png |
| 2 | Tan Zhi Xuan | 17ACB03376 | P (8) | CS |  |
| 3 | Tan Jia Yong | 17ACB03508 | P (7) | CS |  |
| 4 | Leong Hong Xin | 17ACB04530 | P (7) | CS |  |
| 5 | Eddie H’ng Zheng Yan | 17ACB03421 | P (8) | 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. Eddie H’ng Zheng Yan** | **2.Tan Zhi Xuan** | **3. Leong Hong Xin** | **4. Lam Kean Chin** | **5. Tan Jia Yong** | |
| **Student ID** | | **17ACB03421** | **17ACB03376** | **17ACB04530** | **17ACB05143** | **17ACB03508** | |
| **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)**

**\*Minus 10 marks for not printing out the marking scheme (ALL members)**

**(SAMPLE ONLY)TABLE OF CONTENTS**

1.0 Scope of Work………………………………………...…………….…………………………………………………….………....….1

1.1 Business Rule……………………………………………...………………………………………………………………….1

1.2 Requirements………………………………………………………………………………………………………………….2

2.0 ER Model…………………………….………...………………………………………………………………………………….…………4

3.0 Redesign and EER…………………………………………………………………………………………………………………………6

4.0 Data Dictionary………….…………………………………………………………..…………………………….……….….………...7

5.0 Tables and Records…………………………………………………..……………………………………………………...……….13

6.0 SQL script………………………………………………………………………………………………………….……………....………20

7.0 Individual Assessment (Eddie H’ng Zheng Yan, 17ACB03421)……………….…………………………...……..31

7.1 Queries……………………………………………………….………………………………………………………...……..31

7.2 Stored Procedure…………………………………………………………..……………………………………………..37

7.3 Function…………………………………………………………………………………………………………………..……47

8.0 Individual Assessment (Tan Zhi Xuan, 17ACB03376)………………………………………………………...……… 58

8.1 Queries………………………………………….………………………………………………………………………...…..58

8.2 Stored Procedure………………………………..………………………………………………………………………..68

8.3 Function…………………………………………………..………………………………………………………………...…78

9.0 Individual Assessment (Leong Hong Xin, 17ACB04530)………………………………………….………………….89

9.1 Queries…..………………………………………………….……………………………………………………………...…89

9.2 Stored Procedure…..……………………………………………………………………………………………..………98

9.3 Function…...…………………………………………………………………………………………………………..……109

10.0 Individual Assessment (Lam Kean Chin, 17ACB5143 )…………….…………………………………………......115

10.1 Queries……………………………………………………………………………………………………….…………….115

10.2 Stored Procedure……………………………………………………………………………………………..……….123

10.3 Function………………………………………………………………………………………………………..………….133

11.0 Individual Assessment (Tan Jia Yong, 17ACB03508 )…………………………………………………………......138

11.1 Queries……………………………………………………………………………………………………….…………….138

11.2 Stored Procedure…………………………………………………………………………………..………………….144

11.3 Function………………………………………………………………………………………………………..………….150

**1.0 Scope of Work:**

**1.1 Business rule**

1. Each branch can have one or more employees.

2. Each branch has one or more departments.

3. Each employee is assigned to one branch.

4. Each employee is allocated to one department.

5. Each employee is supervised by one manager.

6. Employees will be further grouped into full time employees and part time employees. Full time employee and part time employee is a subclass of employee; employee is the superclass.

7. Each Employee has an employee type.

8. Each employee has one or more leave balance records.

9. Each employee can submit one or many leave reports.

10. The Employees table record must have name, birthday date, hired date, marriage status, number of children age below 12 years old, identity card number for Malaysian and passport number for foreigner employee.

11. Each department can have at least one or many employees.

12. Each department only can have a branch.

13. Each employee type has one or many employees.

14. Each leave balance will record down each of the employee’s leave balance days.

15. A leave balance record is generated by a leave type table.

16. Each leave type will generate none or many leave balance records.

17. Each leave type can include with none or many leave reports.

18. Each leave report is for one employee.

19. Each leave report is included by one leave type.

20. Only the manager can change the leave status.

21. The number of balance days will be deducted once the leave is approved by the manager.

22. The employees can view back the previous leave history from the leave report.

23. Part time employees and trainees can take unpaid leave only.

**1.2 Requirements:**

First, the requirement of the system of company YY will be providing different types of leaves for example Annual Leave, Unpaid Leave, Sick Leave, Hospitalization Leave, Prolong Illness Leave, Examination Leave, Maternity Leave, Paternity Leave, Marriage Leave and Miscellaneous Leave. Each type of leave is defined as follows:

1. Annual Leave – 12 days of Annual Leave for each employee. Each year of the company starts from the 1st of January to the 31st of December
2. Unpaid leave - Applicable but that they are not paid for
3. Sick Leave – Applicable when the employee is sick but not hospitalised.
4. Hospitalization Leave – Applicable when the employee is hospitalized due to an accident/illness
5. Prolong Illness Leave - Employee suffering from a serious debilitating condition that will impair him/her from returning to work may be granted
6. Examination Leave – Applicable when the employee is sent out for an examination under the employee’s field of work. Examinations that are not under the employee’s scope of work are not applicable for the examination leave and should be registered instead as Unpaid Leave
7. Maternity Leave - Applicable only for a mother's absence from work before and after the birth of her child.
8. Paternity Leave - Applicable when a father wants to spend time with his new born baby.
9. Marriage leave - Applicable when the employee is getting married without loss of wages
10. Childcare leave- Applicable for working parents to look after their sick children which is aged below 12 years old.
11. compassionate leave- Applicable when a family member of an employee has passed away.
12. Miscellaneous leave - Employees who meet the definition of a regular employee may be eligible for leaves of absence, other than holiday, vacation, family and medical, parental, leave without pay, sick, and sick leave pool.

Other than that, each employee should only request for one type of leave in a single instance.

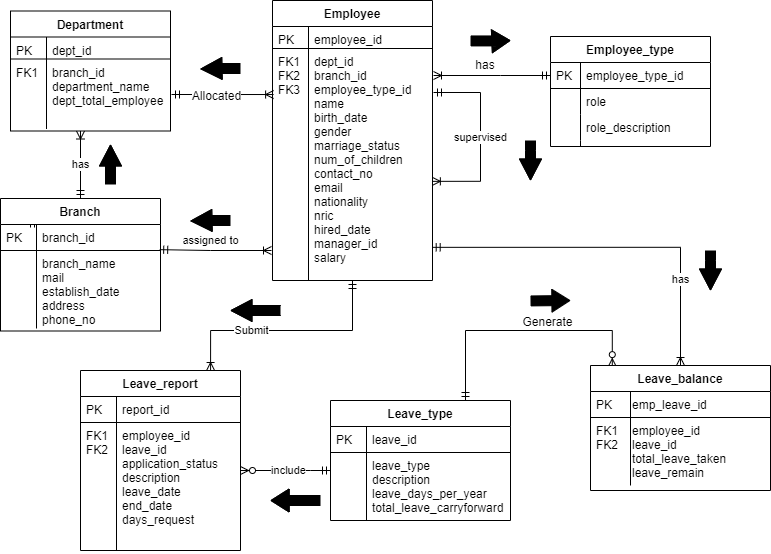
Company YY provides many types of leave. But there are certain types of leave that are only applicable when an employee has fully utilized the other leave type. Such as Unpaid Leave and Prolong Illness Leave. Unpaid Leave is applicable when Annual Leave is fully utilized. Prolong Illness Leave is applicable when Hospitalization Leave was fully utilized. Most of the employees leave type needs to be approved by the manager, but emergency leaves such as sick leave and hospitalization leave can be applied after the leaves are taken. Valid reasons need to be presented to the company when applying any types of leave.

In addition, some of the leave constraints are based on gender and marital status. For instance, only female employees can apply for maternity leave for 2 to 4 months. Hence, marriage leave is applicable for employees having the single status. Next, childcare leave is for employees who have at least one child. Each working parent gets six days of paid childcare leave annually.

All of the employees are given certain days to leave each year. In Company YY, new join employees have a total 12 days of leave each year. Once the new join employee works more than 5 years, the maximum leave of the employees increased to 14 days. All the leave will be unpaid after employees exceed the maximum number of days leave they have.

Employees are allowed to carry forward their days of Annual Leave that were not claimed by them in the previous year for up to 12 days in Annual Leave. Employees that have worked in the company for 5 years or more are only allowed to carry forward a maximum of 10 days of Annual Leave to the next year. Only the Annual Leave can be carried forward to the upcoming year for all employees.

**2.0 ER Model**



Employee

* employee\_id(PK), dept\_id(FK), branch\_id(FK), employee\_type\_id(FK), name, birth\_date, gender, marriage\_status, num\_of\_children, contact\_no, email, nationality, nric, hire\_date, manager\_id, salary

Employee\_type

* employee\_type\_id(PK), role, role\_description

Leave\_balance

* emp\_leave\_id(PK), employee\_id(FK), leave\_id(FK) total\_leave\_taken, leave\_remain

Department

* dept\_id(PK), branch\_id(FK), department\_name, dept\_total\_employee

Branch

* branch\_id(PK), branch\_name, mail, established\_date , address, phone\_no

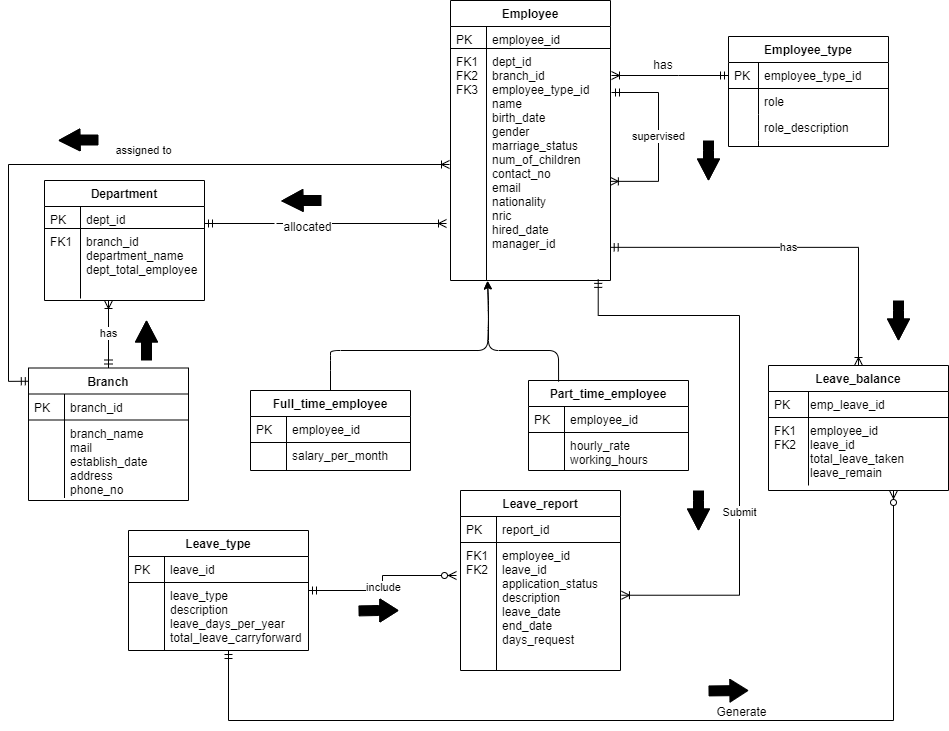
Leave Reports

* report\_id(PK), employee\_id(FK), leave\_id(FK), application\_status, description, leave\_date, end\_date, days\_request

Leave Type

* leave\_id(PK), leave\_type, description, leave\_days\_per\_year, total\_leave\_carryforward

**3.0 Redesign and EER**



**4.0 Data dictionary for entity type**

|  |  |  |  |
| --- | --- | --- | --- |
| Entity Name | Description | Aliases | Occurrence |
| Employee | General term describing employee details of the Company | Employee | Each member of staff works at one particular department |
| Employee\_type | General terms describing the role and description of the employee in the Company. | Role | Each employee works at what position and the details of the position. |
| Department | General term describing the details of the department of the Company | Department | Each department will have one or more employees included in it. |
| Branch | General terms describing the details of the Branch of the Company. | Branch | Each branch is under the same Company that works in a different location. |
| Leave\_report | General term describing the details of the leave request in the report form. | Leave Request Report | Each leave request will be stored in a report form whenever an employee tries to take leave. |
| Leave\_type | General term describing the leave type of the employee chosen to take. | Type of Leave | Each type of leave can be taken once the requirement is fulfilled. |
| Leave\_balance | General term describing the leave balance of the employee. | Leave Balance | Each leave balance recording the number of leaves can be taken by the employee. |

**Data dictionary for relationship type**

|  |  |  |  |
| --- | --- | --- | --- |
| Entity Name | Relationship | Entity Name | Multiplicity |
| Employees | Has | Employee\_type | 1..1 |
|  | Allocated | Department | 1..1 |
|  | Assigned to | Branch | 1..1 |
|  | Supervised | Employees | 1..\* |
|  | Own | Leave\_balance | 1..\* |
|  | Request | Leave\_report | 1..\* |
| Employee\_type | Has | Employees | 1..\* |
| Department | Allocated | Employees | 1..\* |
|  | Has | Branch | 1..1 |
| Branch | Has | Department | 1..\* |
|  | Has | Employees | 1..\* |
| Leave\_report | Request by | Employees | 1..1 |
|  | Include by | Leave\_type | 1..1 |
|  | Has | Leave\_balance | 1..1 |
| Leave\_type | Include | Leave\_report | 0..\* |
|  | Generate | Leave\_balance | 0..\* |
| Leave\_balance | Own by | Employees | 1..1 |
|  | Generate By | Leave\_type | 1..1 |
| Full\_time\_employees | Is A | Employees | - |
| Part\_time\_employees | Is A | Employees | - |

**Data dictionary for attribute**

Employees table attributes

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Attribute | Description | Data Type | Size | Null | Multi  Value | PK/FK | Reference To | Example Data |
| employee\_id | Unique identifies employees | varchar2(8) | 8 | No | No | PK | - | A10001 |
| dept\_id | Unique identifies department | varchar(3) | 3 | No | No | FK | department(department\_id) | A1 |
| branch\_id | Unique identifies branch | varchar(2) | 2 | No | No | FK | branch(branch\_id) | A |
| employee\_type\_id | Unique identifies employee type id | varchar(3) | 3 | No | No | FK | employee\_type(employee\_type\_id) | 1 |
| name | Name of employee | varchar(50) | 50 | No | No | - | - | Lam Kean Chin |
| birth\_date | Birth date of employee | date | 10 | No | No | - | - | 2020-03-29 |
| gender | Gender of employee | varchar(1) | 1 | No | No | - | - | M |
| marriage\_status | Marriage status of employee | varchar(10) | 10 | No | No | - | - | Single |
| num\_of\_children | Number of children which age below 12 years old | int | 1 | No | No | - | - | 0 |
| contact\_no | Contact number of employee | varchar(12) | 12 | Yes | Yes | - | - | 010-2335463 |
| email | Email of employee | varchar(60) | 60 | Yes | No | - | - | keanchin980516@gmail.com |
| nationality | Nationality of employee | varchar(20) | 20 | No | No | - | - | Malaysia |
| nric | Identity number of employee | varchar(20) | 20 | No | No | - | - | 980516-08-6455 |
| hired\_date | Date of employee hired by company | date | 10 | No | No | - | - | 2018-10-10 |
| manager\_id | Show the employee’s manager | varchar(8) | 8 | Yes | No | - | - | A10003 |
| salary | Salary per month of employee | Number(8,2) | 8 | No | No | - | - | 10000 |

Employee type attributes

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Attribute | Description | Data Type | Size | Null | Multi  Value | PK/FK | Reference To | Example Data |
| employee\_type\_id | Unique identifies employee type id | varchar(3) | 3 | No | No | PK | - | 1 |
| role | Employee’s role name in the company | varchar(30) | 30 | No | No | - | - | Software developers |
| role\_description | Employee’s role description in the company | varchar(255) | 255 | Yes | No | - | - | Software developers create applications on database, operating system and software |

Department attributes

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Attribute | Description | Data Type | Size | Null | Multi  Value | PK/FK | Reference To | Example Data |
| dept\_id | Unique identifies department | varchar(3) | 3 | No | No | PK | - | A1 |
| branch\_id | Unique identifies branch | varchar(2) | 2 | No | No | FK | branch(branch\_id) | A |
| dept\_name | Name of the department | varchar(50) | 50 | No | No | - | - | Security Department |
| dept\_total\_employee | Total employees contain in the department | int | 4 | No | No | - | - | 12 |

Branch attributes

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Attribute | Description | Data Type | Size | Null | Multi  Value | PK/FK | Reference To | Example Data |
| branch\_id | Unique identifies branch | varchar(2) | 2 | No | No | PK | - | A |
| name | Name of branch | varchar(50) | 50 | No | No | - | - | Phoenix Ipoh |
| email | Email use for the branch | varchar(60) | 60 | No | No | - | - | phoenixA@gmail.com |
| establish\_date | Date of the branch start working | date | 10 | No | No | - | - | 16/05/2018 |
| address | Exact location of the branch | varchar(255) | 255 | No | No | - | - | 237,Jalan Tai Fi Tong,30010 Ipoh Perak |
| phone\_no | Phone number that use by the branch | varchar(12) | 12 | No | Yes | - | - | 03-2335463 |

Leave Report attributes

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Attribute | Description | Data Type | Size | Null | Multi  Value | PK/FK | Reference To | Example Data |
| report \_id | Unique identifies report | number(5) | 5 | No | No | PK | - | 1234 |
| employee\_id | Unique identifies employees | varchar(6) | 6 | No | No | FK | employees(employee\_id) | A00001 |
| leave\_id | Unique identifies leave\_type | number(3) | 3 | No | No | FK | leave\_type(leave\_id) | 11 |
| application\_status | Show the request pass(approved) or fail(forfeited) | varchar(9) | 9 | No | No | - | - | approved |
| description | Description wrote by the employee for the leave request | varchar(255) | 255 | Yes | No | - | - | Celebrate Chinese Christmas |
| leave\_date | Date that employee start leave | date | 10 | No | No | - | - | 16/05/2020 |
| end\_date | Date that employee need to be come back tomorrow | date | 10 | No | No | - | - | 20/05/2020 |
| days\_request | Number of days request to leave | number(3) | 3 | No | No | - | - | 5 |

Leave Type attributes

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Attribute | Description | Data Type | Size | Null | Multi  Value | PK/FK | Reference To | Example Data |
| leave\_id | Unique identifies leave type | number(3) | 3 | No | No | PK | - | 1 |
| leave\_type | Name of the leave type | varchar(50) | 50 | No | No | - | - | Sick Leave |
| description | Description of the leave type | varchar(255) | 255 | Yes | No | - | - | Guarantee workers [paid time off](https://en.wikipedia.org/wiki/Paid_time_off) to stay home when they are sick |
| leave\_days\_per\_year | Total leave per year of an employee | number(3) | 3 | Yes | No | - | - | 12 |
| max\_days\_carryforward | The max day leave that can carry forward to the next year | number(2) | 2 | No | No | - | - | 10 |

Leave Balance attributes

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Attribute | Description | Data Type | Size | Null | Multi  Value | PK/FK | Reference To | Example Data |
| emp\_leave\_id | Unique identifies leave balance record | number(5) | 5 | No | No | PK | - | 109 |
| employee\_id | Unique identifies employees | varchar2(6) | 6 | No | No | FK | employees(employee\_id) | A00001 |
| leave\_id | Unique identifies leave type | number(3) | 3 | No | No | FK | leave\_type(leave\_id) | 11 |
| total\_leave\_taken | Total leave took by employee | number(2) | 2 | YES | No | - | - | 4 |
| leave\_remain | The remaining leave that can be took by employee | number(3) | 3 | YES | No | - | - | 6 |

**5.0 Tables and Records**

Employee

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| employee\_id | dept\_id | branch\_id |  | employee\_type\_id | name | birth\_date | gender | marriage\_status | num\_of\_children | contact\_no | email | nationality | nric | hired\_  date | manager\_id | Salary |
| A10000 | B1 | B |  | 1 | Lee Jiu Zhe | 1990-03-25 | 'M' | 'Single' | 0 | 012-1122335 | jiuzhe90@gmail.com | Malaysia | 900325-01-1234 | 2014-03-16 | A10004 | 5000 |
| A10001 | B1 | B |  | 2 | 'Ooi hong goh' | 1987-01-11 | M | 'Single' | 0 | 011-3216548 | honggoh87@gmail.com | Malaysia | 870111-02-1122 | 2018-03-16 | A10004 | 4500 |
| A10002 | B1 | B |  | 3 | 'Ng Xiang Wei' | 1991-09-27 | F | 'Single' | 0 | 012-5114489 | xiangwei91@gmail.com | Korea | 910927-05-5566 | 2014-05-24 | A10003 | 10000 |
| A10003 | B1 | B |  | 6 | Lee Ting Wei | 1988-06-24 | F | Single | 0 | 013-5266854 | tingwei88@gmail.com | China | 880624-09-6541 | 2013-03-16 |  | 50000 |
| A10004 | B1 | B |  | 4 | Henry Tan | 1993-12-25 | M | Married | 5 | 010-9879878 | henry1199@ggmai.coc | Malaysia | 931225-01-2645 | 2017-09-10 | A10002 | 8000 |
| A10005 | B1 | B |  | 5 | Lee Jun yan | 1995-01-05 | M | Single | 0 | 016-9523621 | junyan95@gmail.com | Malaysia | 950105-07-7063 | 2018-01-09 | A10004 | 2000 |
| A10006 | B1 | B |  | 1 | Tan Jian Ming | 1985-09-20 | M | Married | 2 | 012-9856362 | jianming85@gmail.com | Malaysia | 850920-07-3087 | 2013-06-21 | A10004 | 5000 |
| A10007 | B1 | B |  | 1 | Ooi Wei Xiang | 1993-12-12 | M | Single | 0 | 019-4825964 | weixiang93@gmail.com | Malaysia | 931212-03-6529 | 2018-04-28 | A10004 | 5000 |
| A20001 | B2 | B |  | 1 | Lim Yan Hui | 1990-08-06 | F | Married | 1 | 012-4862365 | yanhui90@gmail.com | Malaysia | 900806-04-6084 | 2016-11-21 | A10004 | 2000 |
| A20002 | B2 | B |  | 1 | Ng Wei Li | 1992-11-18 | M | Single | 0 | 012-3456854 | Weili92@gmail.com | Malaysia | 921118-03-9063 | 2017-05-06 | A20003 | 5000 |
| A20003 | B2 | B |  | 4 | Ng Pei Feng | 1985-01-20 | F | Single | 0 | 019-6543217 | peifeng85@gmail.com | Malaysia | 850120-05-0458 | 2016-05-27 | A20005 | 8000 |
| A20004 | B2 | B |  | 2 | Tan Zong Guan | 1991-04-19 | M | Married | 0 | 016-5289641 | zongguan91@gmail.com | Thailand | 910419-05-9514 | 2017-05-06 | A20003 | 4500 |
| A20005 | B2 | B |  | 3 | Chuah Fong Jing | 1972-07-25 | F | Single | 0 | 014-9965874 | fongjing72@gmail.com | Malaysia | 720101-07-5236 | 2011-05-13 | A10003 | 10000 |
| A20006 | B2 | B |  | 5 | Goh Zhan Hong | 1984-08-28 | M | Single | 0 | 018-5858641 | zhanhong84@gmail.com | Singapore | 840828-07-9865 | 2015-06-30 | A20003 | 2000 |
| A20007 | B2 | B |  | 1 | Tan Jia Xuan | 1982-02-24 | M | Single | 0 | 017-5450362 | jiaxuan82@gmail.com | Malaysia | 820224-07-8516 | 2019-12-01 | A20003 | 5000 |
| A20008 | B2 | B |  | 1 | Fong Xuan Zi | 1979-09-20 | F | Married | 1 | 014-7539841 | xuanzi79@gmail.com | Malaysia | 790920-08-6152 | 2018-04-01 | A20003 | 5000 |
| A20009 | B2 | B |  | 1 | Ang Quan Xing | 1980-08-31 | M | Single | 0 | 012-5498632 | quanxing80@gmail.com | Malaysia | 800831-03-6621 | 2016-02-09 | A20003 | 5000 |

Employee Type

|  |  |  |
| --- | --- | --- |
| employee\_type\_id | role | role\_description |
| 1 | Full-Time Employee | Employee that works regularly. |
| 2 | Trainee | An employee in training |
| 3 | Manager | The person who manage the department |
| 4 | Team Leader | Team leader that leads a small group of employees to finish the task in time. |
| 5 | Part-Time Employee | Employees that work a few days a week only, no fixed working hours. |
| 6 | CEO | Chief executive in charge of managing of YY company |

Department

|  |  |  |  |
| --- | --- | --- | --- |
| dept\_id | branch\_id | dept\_name | dept\_total\_employee |
| A1 | A | Security Department | 36 |
| B1 | B | IT Department | 40 |
| B2 | B | Marketing Department | 40 |
| C1 | C | Human Resource Department | 35 |
| C2 | C | Sales Department | 110 |
| D1 | D | Design Department | 42 |
| E1 | E | Finance Department | 45 |

Branch

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| branch\_id | name | email | establish\_date | address | phone\_no |
| A | Phoenix Ipoh | phoenixA@gmail.com | 16/05/2018 | 237,Jalan Tai Fi Tong,30010 Ipoh Perak | 03-2335463 |
| B | Phoenix Penang | phoenixB@gmail.com | 26/09/2016 | 136, One Precinct, 11950, Bayan Baru, 11900 Bayan Lepas, Pulau Pinang | 03-2335411 |
| C | Phoenix Kuala Lumpur | phoenixC@gmail.com | 18/03/2014 | 257, Titiwangsa Sentral, 53000 Kuala Lumpur | 03-2335422 |
| D | Phoenix Johor | phoenixD@gmail.com | 07/12/2015 | 415, Jalan Tampoi 7, 81200 Johor Bahru, Johor | 03-2335433 |
| E | Phoenix Taiping | phoenixE@gmail.com | 30/03/2019 | 31, Jalan Medan Taiping 5, Medan Taiping, 34000 Taiping, Perak | 03-2335466 |

Leave Report

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| report \_id | employee\_id | leave\_id | application\_status | description | leave\_date | end\_date | days\_requested |
| 1 | A10007 | 1 | approved | Family Trip | 16/12/2019 | 20/12/2019 | 5 |
| 2 | A10005 | 4 | approved | Fever | 18/12/2019 | 20/12/2019 | 3 |
| 3 | A10001 | 3 | forfeited | Early Christmas celebration | 24/12/2019 | 24/12/2019 | 1 |
| 4 | A20001 | 12 | approved | Family Member Passed away | 25/11/2019 | 27/11/2019 | 3 |
| 5 | A20004 | 10 | approved | Marriage | 06/01/2020 | 07/01/2020 | 2 |
| 6 | A10002 | 2 | approved | Back To Home Country | 08/12/2019 | 12/12/2019 | 5 |
| 7 | A20006 | 3 | approved | Family Trip | 21/01/2020 | 23/01/2020 | 3 |
| 8 | A20007 | 7 | approved | Examination | 02/02/2020 | 08/02/2020 | 7 |
| 9 | A10004 | 11 | approved | Children sick | 25/02/2020 | 26/02/2020 | 2 |
| 10 | A10003 | 5 | approved | CoronaVirus | 18/03/2020 | 06/04/2020 | 20 |

Leave Type

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| leave\_id | leave\_type | description | leave\_days\_per\_year | max\_days\_carryforward  /\*only annual leave can carryforward\*/ |
| 1 | Annual Leave(Junior) | Annual leave for employee work less than 5 years | 12 | 12 |
| 2 | Annual Leave(Senior) | Annual leave for employee work more than 5 years | 14 | 10 |
| 3 | Unpaid Leave | Leave that are not paid | NULL | 0 |
| 4 | Sick Leave | Applicable when sick | NULL | 0 |
| 5 | Hospitalization Leave | Applicable when hospitalized | 20 | 0 |
| 6 | Prolong Illness Leave | Applicable when employee is in a debilitating condition | 40 | 0 |
| 7 | Examination Leave | Applicable when sent out for examination | 15 | 0 |
| 8 | Maternity Leave | Applicable for mother’s absence before and after childbirth | 90 | 0 |
| 9 | Paternity Leave | Applicable for father for time with his newborn | 14 | 0 |
| 10 | Marriage Leave | Applicable for single status employees’ marriage without wage loss | 4 | 0 |
| 11 | Childcare Leave | Applicable for working parents to look after sick child under 12 years old | 6 | 0 |
| 12 | Compassionate Leave | Applicable when a family member of an employee has passed away | 4 | 0 |
| 13 | Miscellaneous Leave | Applicable for leave that were not specified and categorised | 4 | 0 |

Leave Balance

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| emp\_leave\_id | employee\_id | leave\_id | total\_leave\_taken | leave\_remain |
| 1 | A10000 | 2 | 2 | 15 |
| 2 | A10000 | 5 | 0 | 20 |
| 3 | A10000 | 7 | 3 | 15 |
| 4 | A10000 | 10 | 3 | 4 |
| 5 | A10000 | 13 | 5 | 4 |
| 6 | A10001 | 2 | 5 | 13 |

**6.0 SQL Script**

DROP TABLE employee\_type CASCADE CONSTRAINT;

DROP TABLE branch CASCADE CONSTRAINT;

DROP TABLE department CASCADE CONSTRAINT;

DROP TABLE employee CASCADE CONSTRAINT;

DROP TABLE leave\_type CASCADE CONSTRAINT;

DROP TABLE leave\_report CASCADE CONSTRAINT;

DROP TABLE leave\_balance CASCADE CONSTRAINT;

DROP SEQUENCE seq\_emp\_leave\_id;

CREATE TABLE employee\_type(

employee\_type\_id VARCHAR2(3) NOT NULL,

role VARCHAR2(30) NOT NULL,

role\_description VARCHAR2(255) NOT NULL,

CONSTRAINT empt\_employee\_type\_id\_pk PRIMARY KEY(employee\_type\_id)

);

CREATE TABLE branch (

branch\_id VARCHAR(2) NOT NULL,

name VARCHAR(50) NOT NULL,

email VARCHAR(60) NOT NULL,

establish\_date DATE NOT NULL,

address VARCHAR(255) NOT NULL,

phone\_no VARCHAR(12) NOT NULL,

CONSTRAINT branch\_branch\_id\_pk PRIMARY KEY(branch\_id)

);

CREATE TABLE department (

dept\_id VARCHAR(3) NOT NULL,

branch\_id VARCHAR(2) NOT NULL,

dept\_name VARCHAR(50) NOT NULL,

dept\_total\_emp INT NOT NULL,

CONSTRAINT dept\_id PRIMARY KEY (dept\_id),

CONSTRAINT department\_branch\_id\_fk FOREIGN KEY(branch\_id) REFERENCES branch(branch\_id)

);

CREATE TABLE employee (

employee\_id VARCHAR2(8) NOT NULL,

dept\_id VARCHAR2(3) NOT NULL,

branch\_id VARCHAR2(2) NOT NULL,

employee\_type\_id VARCHAR2(3) NOT NULL ,

name VARCHAR2(50) NOT NULL,

birth\_date DATE NOT NULL,

gender VARCHAR2(1) NOT NULL,

marriage\_status VARCHAR2(10) NOT NULL,

num\_of\_children INT NOT NULL,

contact\_no VARCHAR2(12) NOT NULL,

email VARCHAR2(60) NOT NULL,

nationality VARCHAR2(20) NOT NULL,

nric VARCHAR2(20) NOT NULL,

hired\_date DATE NOT NULL,

manager\_id VARCHAR2(8),

salary NUMBER(8,2) NOT NULL,

CONSTRAINT employee\_employee\_id\_pk PRIMARY KEY(employee\_id),

CONSTRAINT employee\_dept\_id\_fk FOREIGN KEY(dept\_id) REFERENCES department(dept\_id),

CONSTRAINT employee\_branch\_id\_fk FOREIGN KEY(branch\_id) REFERENCES branch(branch\_id),

CONSTRAINT employee\_type\_id\_fk FOREIGN KEY(employee\_type\_id) REFERENCES employee\_type(employee\_type\_id)

);

CREATE TABLE leave\_type (

leave\_id NUMBER(3) NOT NULL,

leave\_type VARCHAR2(50) NOT NULL,

description VARCHAR2(255),

leave\_days\_per\_year NUMBER(3),

max\_days\_carryforward NUMBER(3) NOT NULL,

CONSTRAINT leave\_type\_pk PRIMARY KEY(leave\_id)

);

CREATE TABLE leave\_report (

report\_id NUMBER(5) NOT NULL,

employee\_id VARCHAR2(6) NOT NULL,

leave\_id NUMBER(3) NOT NULL,

application\_status VARCHAR2(9) NOT NULL,

description VARCHAR2(255) NOT NULL,

leave\_date DATE NOT NULL,

end\_date DATE NOT NULL,

days\_request NUMBER(3) NOT NULL,

CONSTRAINT leave\_report\_pk PRIMARY KEY(report\_id),

CONSTRAINT leave\_report\_leave\_id\_fk FOREIGN KEY(leave\_id) REFERENCES leave\_type(leave\_id),

CONSTRAINT leave\_report\_employee\_id\_fk FOREIGN KEY(employee\_id) REFERENCES employee(employee\_id)

);

CREATE TABLE leave\_balance

(

emp\_leave\_id NUMBER(5),

employee\_id VARCHAR2(6),

leave\_id NUMBER(3),

total\_leave\_taken NUMBER(2),

leave\_remain NUMBER(3),

CONSTRAINT leave\_balance\_emp\_leave\_id\_pk PRIMARY KEY(emp\_leave\_id),

CONSTRAINT leave\_balance\_employee\_id\_fk FOREIGN KEY(employee\_id) REFERENCES employee(employee\_id),

CONSTRAINT leave\_balance\_leave\_id\_fk FOREIGN KEY(leave\_id) REFERENCES leave\_type(leave\_id)

);

CREATE SEQUENCE seq\_emp\_leave\_id

MINVALUE 1

START WITH 1

INCREMENT BY 1

MAXVALUE 99900

NOCACHE

NOCYCLE;

INSERT INTO employee\_type

VALUES ('1','Full-Time Employee','Employee that works

regularly.');

INSERT INTO employee\_type

VALUES ('2','Trainee','An employee in training.');

INSERT INTO employee\_type

VALUES ('3','Manager','The person who manage the

department.');

INSERT INTO employee\_type

VALUES ('4','Team Leader','Team leader that leads a small group of employees to finish the task in time.');

INSERT INTO employee\_type

VALUES ('5','Part-Time Employee','Employees that work a few

days a week only, no fixed working hours.');

INSERT INTO employee\_type

VALUES('6','CEO','Chief executive in charge of managing of YY company.');

INSERT INTO branch VALUES ('A', 'Phoenix Ipoh', 'phoenixa@gmail.com', '16-MAY-18', '237,Jalan Tai Fi Tong,30010 Ipoh Perak', '03-2335463');

INSERT INTO branch VALUES ('B', 'Phoenix Penang',

'phoenixb@gmail.com', '26-SEP-16', '136, One Precinct, 11950, Bayan Baru, 11900

Bayan Lepas, Pulau Pinang', '03-2335411');

INSERT INTO branch VALUES ('C', 'Phoenix Kuala Lumpur',

'phoenixc@gmail.com', '18-MAR-14', '257, Titiwangsa Sentral, 53000 Kuala

Lumpur', '03-2335422');

INSERT INTO branch VALUES ('D', 'Phoenix Johor',

'phoenixd@gmail.com', '07-DEC-15', '415, Jalan Tampoi 7, 81200 Johor Bahru,

Johor', '03-2335433');

INSERT INTO branch VALUES ('E', 'Phoenix Taiping',

'phoenixe@gmail.com', '30-SEP-19', '31, Jalan Medan Taiping 5, Medan Taiping,

34000 Taiping, Perak', '03-2335466');

INSERT INTO department VALUES ('A1', 'A', 'Security Department', '36');

INSERT INTO department VALUES ('B1', 'B', 'IT Department', '40');

INSERT INTO department VALUES ('B2', 'B', 'Marketing Department', '40');

INSERT INTO department VALUES ('C1', 'C', 'Human Resource Department', '35');

INSERT INTO department VALUES ('C2', 'C', 'Sales Department', '110');

INSERT INTO department VALUES ('D1', 'D', 'Design Department', '42');

INSERT INTO department VALUES ('E1', 'E', 'Finance Department', '45');

INSERT INTO employee

VALUES('A10000','B1','B','1','Lee Jiu Zhe',date

'1990-03-25','M','SINGLE','0','012-1122335','jiuzhe90@gmail.com','Malaysia','900325-01-1234',date

'2014-03-16','A10004','5000');

INSERT INTO employee

VALUES('A10001','B1','B','2','Ooi Hong Goh',date

'2000-01-11','M','MARRIED','2','011-3216548','honggoh87@gmail.com','Malaysia','000111-02-1122',date

'2019-03-16','A10004','4500');

INSERT INTO employee

VALUES('A10002','B1','B','3','Ng Xiang Wei',date

'1991-09-27','F','SINGLE','0','012-5114489','xiangwei91@gmail.com','Korea','910927-05-5566',date

'2014-05-24','A10003','10000');

INSERT INTO employee

VALUES('A10003','B1','B','6','Lee Ting Wei',date

'1988-06-24','F','SINGLE','0','013-5266854','weiting88@gmail.com','China','880624-09-6541',date

'2013-03-16','NULL','50000');

INSERT INTO employee

VALUES('A10004','B1','B','4','Henry Tan',date

'1993-12-25','M','MARRIED','5','010-9879878',

'henry1993@gmail.com','Malaysia','931225-01-2645',date

'2017-09-10','A10002','8000');

INSERT INTO employee

VALUES('A10005','B1','B','5','Lee Jun Yan',date

'2001-01-05','M','SINGLE','0','016-9523621','junyan95@gmail.com','Malaysia','010105-07-7063',date

'2019-01-09','A10004','2000');

INSERT INTO employee

VALUES('A10006','B1','B','1','Tan Jian Ming',date

'1985-09-20','M', 'MARRIED','3','012-9856362','jianming85@1utar.my','Malaysia','850920-06-9514',date

'2013-06-21','A10004','5000');

INSERT INTO employee

VALUES('A10007','B1','B','1','Ooi Wei Xiang',date

'1993-12-12','M','SINGLE','0','019-4825964','weixiang93@gmail.com','Malaysia','931212-03-6529',date '2018-04-28','A10004','5000');

INSERT INTO employee

VALUES('A10008','B1','B','1','Lim Yan Hui',date

'1990-08-06','F','MARRIED','1','012-4862365','yanhui90@gmail.com','Malaysia','900806-04-6084',date

'2020-11-21','A10004','5000');

INSERT INTO employee

VALUES('A20001','B2','B','1','Ng Wei Li',date

'1992-11-18','M','SINGLE','0','012-3456854','weili92@gmail.com','Malaysia','921118-03-9063',date

'2017-05-06','A20002','5000');

INSERT INTO employee

VALUES('A20002','B2','B','4','Ng Pei Feng',date

'1985-01-20','F','SINGLE','0','019-6543217','peifeng85@gmail.com','Malaysia','850120-05-0458',date

'2016-05-27','A10004','8000');

INSERT INTO employee

VALUES('A20003','B2','B','2','Tan Zong Guan',date

'1998-04-19','M','MARRIED','0','016-5289641','zongguan91@gmail.com','Thailand','9810419-05-9514',date

'2018-05-06','A20002','4500');

INSERT INTO employee

VALUES('A20004','B2','B','3','Chuah Fong Jing',date

'1972-07-25','M','SINGLE','0','014-9965874','fongjing72@gmail.com','Malaysia','720101-07-5236',date

'2011-05-13','A10003','10000');

INSERT INTO employee

VALUES('A20005','B2','B','5','Goh Zhan Hong',date

'1999-08-28','M','SINGLE','0','018-5858641','zhanhong84@gmail.com','Singapore','990828-07-9865',date

'2019-06-30','A20002','2000');

INSERT INTO employee

VALUES('A20006','B2','B','1','Tan Jia Xuan',date

'1982-02-24','M','SINGLE','0','017-5450362','jiaxuan82@gmail.com','Malaysia','820224-07-8516',date

'2019-12-01','A20002','5000');

INSERT INTO employee

VALUES('A20007','B2','B','1','Fong Xuan Zi',date

'1979-09-20','F','MARRIED','1','014-7539841','xuanzi79@gmail.com','Malaysia','790920-08-6152',date

'2018-04-01','A20002','5000');

INSERT INTO employee

VALUES('A20008','B2','B','1','Ang Quan Xing',date

'1980-08-31','M','SINGLE','0','012-5498632','quanxing80@gmail.com','Malaysia','800831-03-6621',date

'2016-02-09','A20002','5000');

INSERT INTO leave\_type VALUES('1','Annual leave(Junior)','Annual Leave for employees working less than 5 years','12','12');

INSERT INTO leave\_type VALUES('2','Annual

Leave(Senior)','Annual :Leave for employees working more than 5 years','14','10');

INSERT INTO leave\_type VALUES('3','Unpaid Leave','Leave

that are not paid', NULL, '0');

INSERT INTO leave\_type VALUES('4','Sick Leave','Applicable when sick',null,'0');

INSERT INTO leave\_type VALUES('5','Hospitalization

Leave','Applicable when hospitalized','20','0');

INSERT INTO leave\_type VALUES('6','Prolong Illness

Leave','Applicable when employee is in a debilitating condition','40','0');

INSERT INTO leave\_type VALUES('7','Examination

Leave','Applicable when sent out for examination','15','0');

INSERT INTO leave\_type VALUES('8','Maternity

Leave','Applicable for mother absence before and after childbirth','90','0');

INSERT INTO leave\_type VALUES('9','Paternity

leave','Applicable for father for time with his newborn','14','0');

INSERT INTO leave\_type VALUES('10','Marriage leave','Applicable

for single status employees marriage without wage loss','4','0');

INSERT INTO leave\_type VALUES('11','Childcare

Leave','applicable for working parents to look after sick child under 12 years

old','6','0');

INSERT INTO leave\_type VALUES('12','Compassionate

Leave','Applicable when a family member of an employee has passed

away','4','0');

INSERT INTO leave\_type VALUES('13','Miscellaneous

Leave','Applicable for leave that were not specified and categorised','4','0');

INSERT INTO leave\_report

VALUES('1','A10007','1','APPROVED','Family trip', date '2019-12-16',date '2019-12-20','5');

INSERT INTO leave\_report

VALUES('2','A10005','4','APPROVED','Fever',date '2019-12-18',date '2019-12-20','3');

INSERT INTO leave\_report

VALUES('3','A10001','3','FORFEITED','Early christmas celebration',date '2019-12-24',date '2019-12-24','1');

INSERT INTO leave\_report

VALUES('4','A20001','12','APPROVED','Family member passed away',date '2019-11-25',date '2019-11-27','3');

INSERT INTO leave\_report

VALUES('5','A20002','10','APPROVED','Marriage',date '2020-01-06',date '2020-01-07','2');

INSERT INTO leave\_report

VALUES('6','A10002','2','APPROVED','Back to home country',date '2019-12-08',date

'2019-12-12','5');

INSERT INTO leave\_report

VALUES('7','A20008','3','APPROVED','Family trip',date '2020-05-21',date '2020-05-23','3');

INSERT INTO leave\_report VALUES('8','A20007','7','APPROVED','Examination',date '2020-02-02',date '2020-02-08','7');

INSERT INTO leave\_report

VALUES('9','A10004','11','APPROVED','Children sick',date '2020-02-25',date '2020-02-27','3');

INSERT INTO leave\_report

VALUES('10','A10003','5','APPROVED','Coronavirus',date '2020-03-18',date '2020-04-06','20');

INSERT INTO leave\_report VALUES('11','A20007','7','PENDING','Examination for Master',date '2020-09-02',date '2020-09-06','5');

INSERT INTO leave\_report

VALUES('12','A20008','1','APPROVED', 'Family trip',date '2020-01-10',date '2020-01-21','12');

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A10000', 2, 2, 15);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A10000', 5, 0, 20);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A10000', 7, 0, 15);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A10000', 10, 0, 4);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A10000', 13, 0, 4);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A10002', 2, 5, 13);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A10002', 5, 2, 18);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A10002', 7, 2, 13);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A10002', 8, 0, 90);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A10002', 10, 0, 4);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A10002', 13, 0, 4);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A10003', 2, 2, 13);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A10003', 5, 20, 0);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A10003', 7, 0, 15);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A10003', 8, 0, 90);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A10003', 10, 4, 0);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A10003', 13, 1, 3);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A10004', 1, 4, 10);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A10004', 5, 0, 20);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A10004', 7, 0, 15);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A10004', 9, 14, 0);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A10004', 11, 3, 3);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A10004', 13, 2, 2);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A10005', 2, 1, 13);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A10005', 5, 0, 20);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A10005', 7, 0, 15);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A10005', 9, 0, 14);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A10005', 11, 2, 4);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A10005', 13, 0, 4);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A10007', 1, 1, 11);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A10007', 5, 0, 20);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A10007', 7, 0, 15);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A10007', 10, 0, 4);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A10007', 13, 0, 4);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A20001', 1, 3, 11);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A20001', 5, 0, 20);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A20001', 7, 0, 15);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A20001', 8, 0, 90);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A20001', 11, 2, 4);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A20001', 13, 1, 3);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A20002', 1, 5, 7);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A20002', 5, 0, 20);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A20002', 7, 5, 10);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A20002', 10, 4, 0);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A20002', 13, 0, 4);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A20003', 1, 2, 10);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A20003', 5, 1, 19);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A20003', 7, 0, 14);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A20003', 8, 0, 90);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A20003', 10, 0, 4);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A20003', 13, 1, 3);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A20005', 2, 0, 14);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A20005', 5, 0, 20);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A20005', 7, 0, 14);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A20005', 8, 0, 90);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A20005', 10, 0, 4);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A20005', 13, 0, 4);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A20007', 1, 1, 11);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A20007', 5, 0, 20);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A20007', 7, 7, 8);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A20007', 10, 0, 4);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A20007', 13, 0, 4);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A20008', 1, 12, 0);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A20008', 5, 4, 16);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A20008', 7, 0, 15);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A20008', 8, 90, 0);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A20008', 11, 1, 5);

INSERT INTO leave\_balance VALUES

(seq\_emp\_leave\_id.nextval, 'A20008', 13, 0, 4);

commit;

**7.0 Individual Assessment( Eddie H’ng Zheng Yan, 17ACB03421 )**

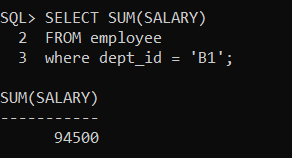
**7.1 QUERIES**

**Query 1:**

SELECT SUM(salary)

FROM employee

WHERE dept\_id = 'B1';



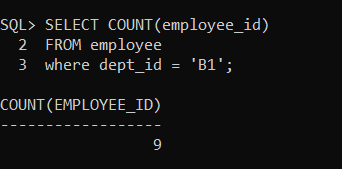
**FUNCTION**: can calculate the total salary for the particular department.

**QUERY 2:**

SELECT COUNT(employee\_id)

FROM employee

WHERE dept\_id = 'B1';

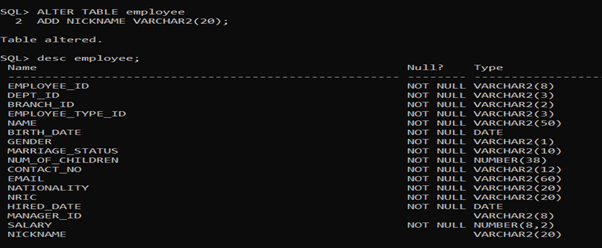


**FUNCTION**: can count whether how many employees allocate in the department.

**QUERY 3:**

ALTER TABLE employee

ADD NICKNAME VARCHAR2(20);



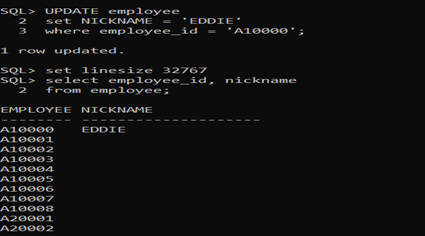
**FUNCTION**: allow CEO/manager to alter the table like add new column.

**QUERY 4:**

UPDATE employee

SET NICKNAME = 'EDDIE'

WHERE employee\_id = 'A10000';



**FUNCTION**: It allow employee to UPDATE the existing records in the table.

**QUERY 5:**

SELECT employee\_id, name

FROM employee

WHERE marriage\_status = 'SINGLE';



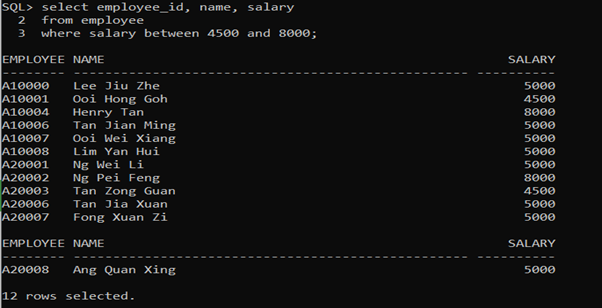
**FUNCTION**: allow the employee to search for something with condition, for example search for the name of employees that are single.

**QUERY 6:**

SELECT employee\_id, name, salary

FROM employee

WHERE salary BETWEEN 4500 AND 8000;



**FUNCTION**: allow employee to search for the employee within the range of the salary for bonus calculation or other reason.

**QUERY 7:**

DESC employee;



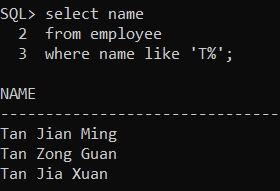
**FUNCTION**: describe the employee table.

**QUERY 8:**

SELECT name

FROM employee

WHERE name LIKE 'T%';



**FUNCTION**: to select some employee with name start from the specific alphabet.

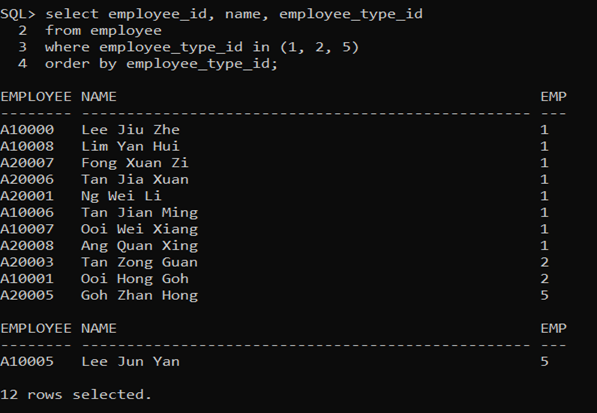
**QUERY 9:**

SELECT employee\_id, name, employee\_type\_id

FROM employee

WHERE employee\_type\_id IN (1, 2, 5)

ORDER BY employee\_type\_id;



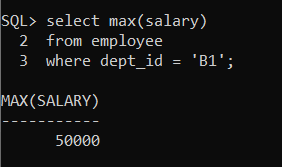
**FUNCTION**: select the employee with employee type id in multiple specific values.

**QUERY 10:**

SELECT MAX(salary)

FROM employee

WHERE dept\_id = 'B1';



**FUNCTION**: find out which employee with the highest salary.

**7.2 STORED PROCEDURES**

**Procedure 1:**

CREATE OR REPLACE PROCEDURE add\_employee

(employee\_id in VARCHAR2,

dept\_id in VARCHAR2,

branch\_id in VARCHAR2,

employee\_type\_id in VARCHAR2,

name in VARCHAR2,

birth\_date in DATE,

gENDer in VARCHAR2,

employee\_id in VARCHAR2,

employee\_id in VARCHAR,

marriage\_status in VARCHAR2, num\_of\_children in NUMBER, contact\_no in VARCHAR2, email in VARCHAR2, nationality in VARCHAR2, nric in VARCHAR2, hired\_date in DATE, manager\_id in VARCHAR2, salary in NUMBER)

IS BEGIN

INSERT INTO employee VALUES

(employee\_id, dept\_id, branch\_id, employee\_type\_id, name, birth\_date, gender, employee\_id ,employee\_id, marriage\_status, num\_of\_children, contact\_no, email, nationality, nric, hired\_date, manager\_id, salary);

COMMIT;

END;

/

EXECUTE add\_employee('A30001 ', 'B1', 'B', '5', 'OWEN' ,date '2001-03-06', 'M', 'SINGLE', '0', '013-5112657', 'owen01@gmail.com', 'Malaysia', '010306-70-5623',date '2020-04-01', 'A10002', '2000');

**EXPLANATION:** This procedure is created for the user to add new recruit employees

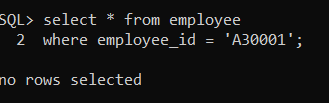


Figure 1: SHOW THE TABLE BEFORE ADDED NEW EMPLOYEE

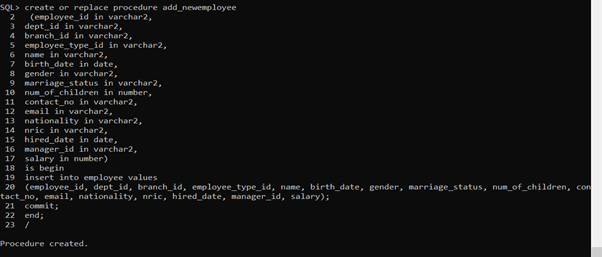


Figure 2: CREATE PROCEDURE add\_employee



Figure3: PERFORMING EXECUTION TO ADD NEW EMPLOYEE



Figure 3: AFTER ADDED NEW EMPLOYEE

**Procedure 2:**

CREATE OR REPLACE PROCEDURE update\_employee

(sel\_employee\_id in VARCHAR2,

new\_name in VARCHAR2)

IS BEGIN

UPDATE employee

SET name = new\_name

WHERE employee\_id = sel\_employee\_id;

COMMIT;

END;

/

EXECUTE update\_employee(‘A30001’, ‘OXWALD’);

WHERE employee\_id = ‘A30001’;

**EXPLANATION:** This procedure is created with the purpose of allowing the users to update the employee’s data for example name.



Figure 1: SHOW THE TABLE BEFORE UPDATE THE EMPLOYEE

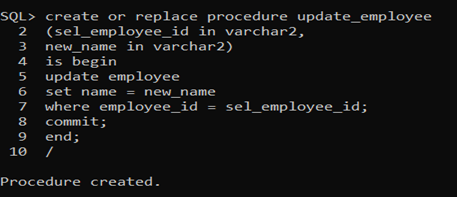


Figure 2: CREATE THE PROCEDURE UPDATE\_EMPLOYEE

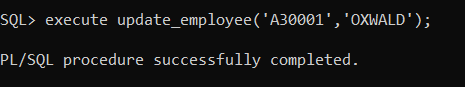


Figure 3: SHOW THE EXECUTE OF THE PROCEDURE SUCCESSFUL

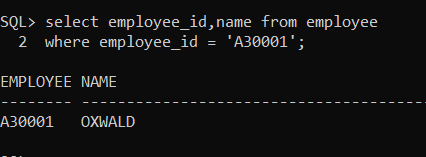


Figure 4: SHOW THE SELECTED EMPLOYEE AFTER CHANGES

**Procedure 3:**

CREATE OR REPLACE PROCEDURE delete\_employee

(del\_employee\_id in VARCHAR2)

IS BEGIN

DELETE FROM employee

WHERE employee\_id = del\_employee\_id;

COMMIT;

END;

/

EXECUTE delete\_employee(‘A30001’);

**EXPLANATION:** This procedure is created for the users to delete any employee that is no longer working for that company.

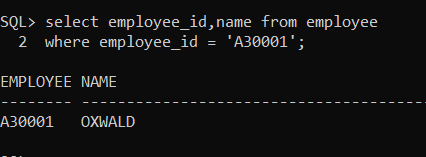


Figure 1: SHOW THE EMPLOYEE BEFORE DELETE

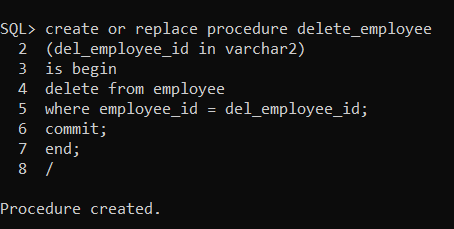


Figure 2: CREATING THE PROCEDURE DELETE\_EMPLOYEE

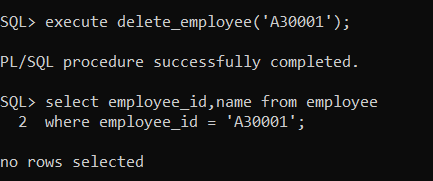


Figure 3: SHOW THE PROCEDURE TO DELETE SUCCESSFUL

**Procedure 4:**

CREATE OR REPLACE PROCEDURE update\_salary

(s\_employee\_id in VARCHAR2,

New\_salary in NUMBER)

IS BEGIN

UPDATE employee

SET salary = new\_salary

WHERE employee\_id = s\_employee\_id;

COMMIT;

END;

/

EXECUTE update\_salary(‘A10000’, ‘8000’);

**EXPLANATION:** This procedure is created for the users to update the employee's salary.

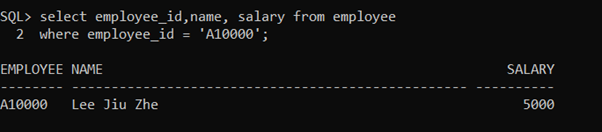


Figure 1 SHOW THE TABLE BEFORE UPDATE THE SALARY



Figure 2 SHOW THE STEP FOR CREATING PROCEDURE UPDATE\_SALARY



Figure 3 AFTER THE SALARY HAD BEEN UPDATED

**Procedure 5:**

CREATE OR REPLACE PROCEDURE update\_employee\_type

(semployee\_id in VARCHAR2,

newemployee\_type in VARCHAR2)

IS BEGIN

UPDATE employee

SET employee\_type\_id = newemployee\_type;

WHERE employee\_id = semployee\_id;

COMMIT;

END;

/

EXECUTE update\_salary(‘A10000’, ‘8000’);

**EXPLANATION:** This procedure is created for the purpose of update the employee’s role

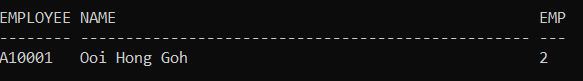


Figure 1 BEFORE UPDATE THE EMPLOYEE'S ROLE

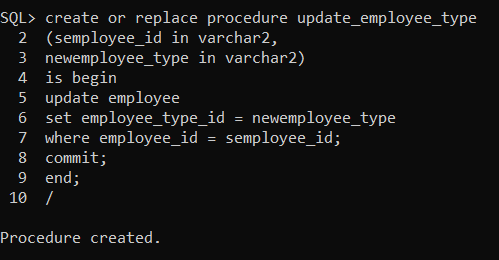


Figure 2 CREATING THE PROCEDURE TO UPDATE THE EMPLOYEE ROLE

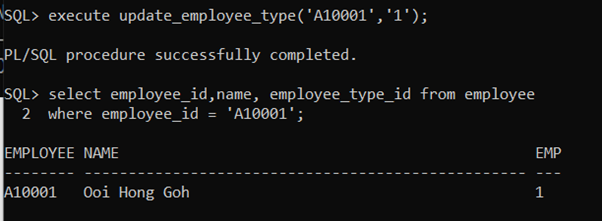


Figure 3 SHOW THE SUCCESS OF UPDATING THE ROLE OF EMPLOYEE

**7.3 FUNCTION**

**Function 1:**

CREATE OR REPLACE FUNCTION totalnum

(d\_id VARCHAR2)

RETURN VARCHAR2

IS

Totale VARCHAR2(50);

BEGIN

SELECT COUNT(employee\_id) INTO totale FROM employee

WHERE dept\_id = d\_id;

RETURN(totale);

END;

/

DECLARE

BEGIN

DBMS\_OUTPUT.PUT\_LINE(‘The total employee in the department is ‘|| totalnum(‘&department\_id’) ||’.’);

END;

/

**EXPLANATION:** This function is created to calculate the total number of the employee in a specific department.

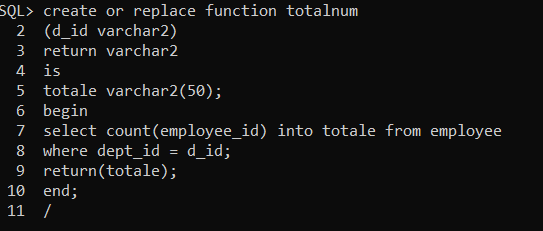


Figure 1 STEP OF CREATING THE FUNCTION TO CALCULATE TOTAL EMPLOYEE

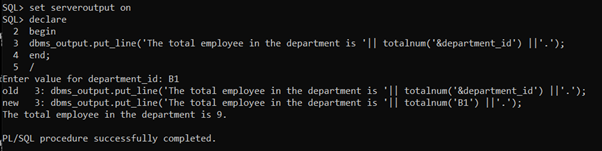


Figure 2 DISPLAY THE TOTAL EMPLOYEE IN DEPARTMENT B1

**Function 2:**

CREATE OR REPLACE FUNCTION total\_income

(d\_id in VARCHAR2)

RETURN VARCHAR2

IS

totalin NUMBER;

BEGIN

SELECT SUM(salary) INTO totalin FROM employee

WHERE dept\_id = d\_id;

RETURN(totalin);

END;

/

DECLARE

BEGIN

DBMS\_OUTPUT.PUT\_LINE(‘The total income of this department is ‘|| total\_income(‘&department\_id’) ||’.’);

END;

/

**EXPLANATION:** This function is created to calculate the total income of a selected department.

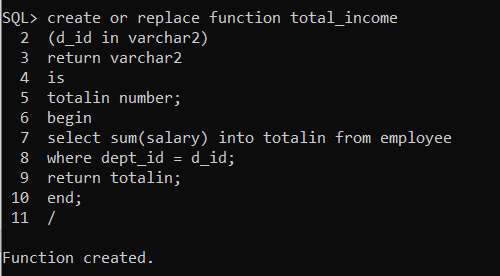


Figure 1 CREATING THE FUNCTION TO CALCULATE THE TOTAL INCOME

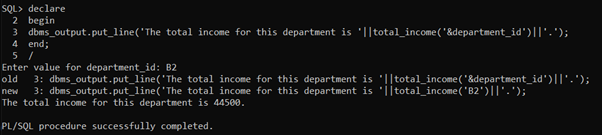


Figure 2 DISPLAY THE OUTPUT(TOTAL INCOME) OF THE DEPARTMENT B2

**Function 3:**

CREATE OR REPLACE FUNCTION marriagestatus1

(d\_id in VARCHAR2)

RETURN VARCHAR2

IS

single VARCHAR2(50);

married VARCHAR2(50);

result VARCHAR2(100);

BEGIN

SELECT COUNT(DISTINCT(employee\_id)) INTO single FROM employee

WHERE dept\_id = d\_id AND marriage\_status = ‘SINGLE’;

SELECT COUNT(DISTINCT(employee\_id)) INTO married FROM employee

WHERE dept\_id = d\_id AND marriage\_status = ‘MARRIED’;

result:=‘There are ‘||single||’ of employees are still single and ‘||married||’ employees are married’;

RETURN(result);

END;

/

DECLARE

BEGIN

DBMS\_OUTPUT.PUT\_LINE(marriagestatus1(‘&department\_id’));

END;

/

**EXPLANATION:** This function is created to find out the exact marriage status of all the employees in a department.

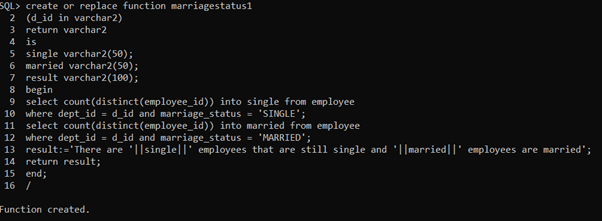


Figure 1 CREATING THE FUNCTION TO CHECK FOR THE MARRIAGE STATUS OF EMPLOYEES IN A DEPARTMENT

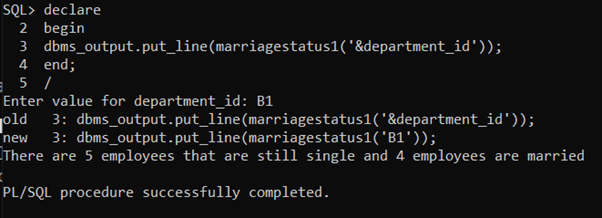


Figure 5 SHOW THE EXACT MARRIAGE STATUS OF EMPLOYEE IN THE DEPARTMENT B1

**Function 4:**

CREATE OR REPLACE FUNCTION manager

(b\_id in VARCHAR2)

RETURN VARCHAR2

IS

totalmanager VARCHAR2(30);

BEGIN

SELECT COUNT(distinct(employee\_type\_id)) INTO totalmanager FROM employee

WHERE branch\_id = b\_id AND employee\_type\_id = ‘3’;

RETURN totalmanager;

END;

/

DECLARE

BEGIN

DBMS\_OUTPUT.PUT\_LINE(‘The total income of this department is ‘|| total\_income(‘&department\_id’) ||’.’);

END;

/

**EXPLANATION:** This function is created to find out how many managers the department had.

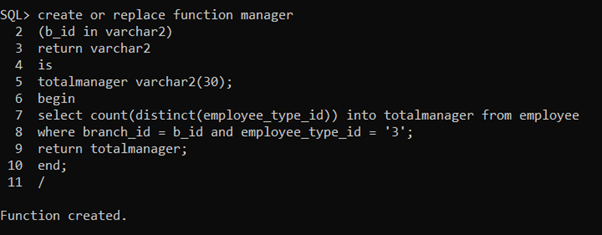


Figure 1 CREATING THE FUNCTION TO FIND THE AMOUNT OF MANAGER IN A DEPARTMENT

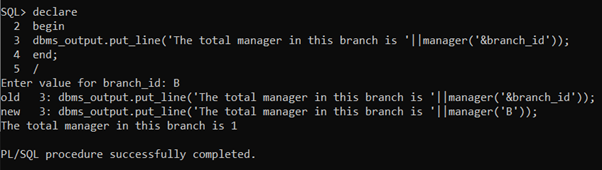


Figure 2 DISPLAY THE EXACT AMOUNT OF EMPLOYEE IN DEPARTMENT B

**Function 5:**

CREATE OR REPLACE FUNCTION subordinate1

(emp\_id in VARCHAR2)

RETURN VARCHAR2

IS

mng\_name VARCHAR2(30);

mng\_id VARCHAR2(20);

result VARCHAR2(100);

BEGIN

SELECT COUNT(employee\_id) INTO totalsub FROM employee

WHERE manager\_id = emp\_id;

SELECT name INTO mng\_name from employee

WHERE employee\_id = emp\_id;

Result:=‘Employee ‘||mng\_name||’ with ‘||mng\_id||’ has a total of ‘||totalsub||’ subORdinates.’;

RETURN result;

END;

/

DECLARE

BEGIN

DBMS\_OUTPUT.PUT\_LINE(subordinate1(‘&manager\_id’));

END;

/

**EXPLANATION:** This function is created to find out subordinates a manager has.



Figure 1 CREATING THE FUNCTION TO FIND OUT THE NUMBER OF SUBORDINATE OF A EMPLOYEE

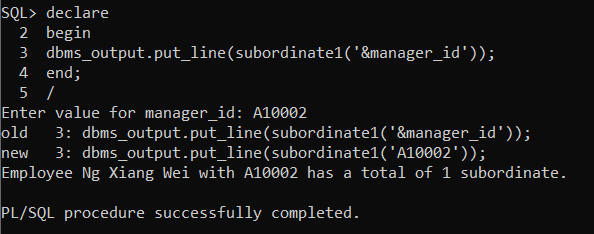


Figure 6 DISPLAY THE EXACT AMOUNT OF SUBORDINATE OF EMPLOYEE A10002

**8.0 Individual Assessment (Tan Zhi Xuan 17ACB03376)**

**8.1 QUERIES**

**Query 1:**

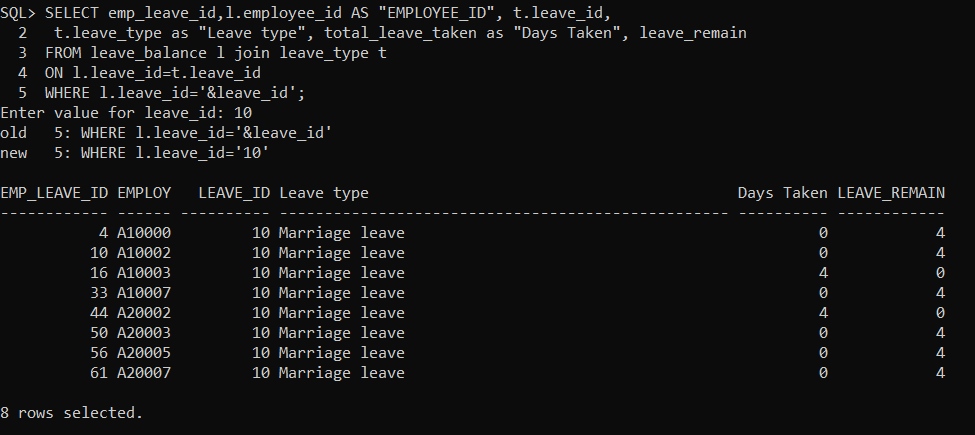
SELECT emp\_leave\_id,l.employee\_id AS "EMPLOYEE\_ID", t.leave\_id,

t.leave\_type as "Leave type", total\_leave\_taken as "Days Taken", leave\_remain

FROM leave\_balance l join leave\_type t

ON l.leave\_id=t.leave\_id

WHERE l.leave\_id='&leave\_id';



This query enables users to be able to get the selected leave id and display all the selected same type of leave from the leave balance table.

**Query 2:**

CREATE OR REPLACE VIEW employee\_working\_years AS

SELECT employee\_id, name, role, trunc((sysdate-hired\_date)/365.25) AS "Total Years Worked"

FROM employee E, employee\_type T

where E.employee\_type\_id=T.employee\_type\_id

ORDER BY "Total Years Worked" DESC

WITH READ only;

SELECT \* FROM employee\_working\_years;



This query enables users to be able to calculate how many years the employee works.

**Query 3:**

CREATE OR REPLACE VIEW only\_approved AS

SELECT e.employee\_id, e.name, r.application\_status as status, t.leave\_type, r.leave\_date, r.days\_request

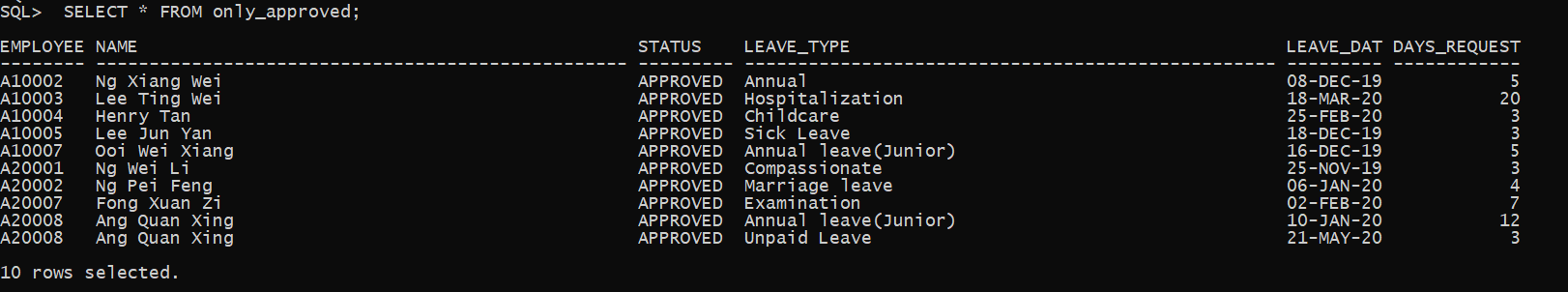
FROM employee e, leave\_report r, leave\_type t

Where ((e.employee\_id=r.employee\_id) AND (r.leave\_id=t.leave\_id))

AND(application\_status='APPROVED');

SELECT \* FROM only\_approved;





This query has been created a view to display all those with approved status in the future or passed.

**Query 4:**

select employee\_id, name, employee\_type\_id as "Role type",

CASE gender

when 'M' THEN 'Male'

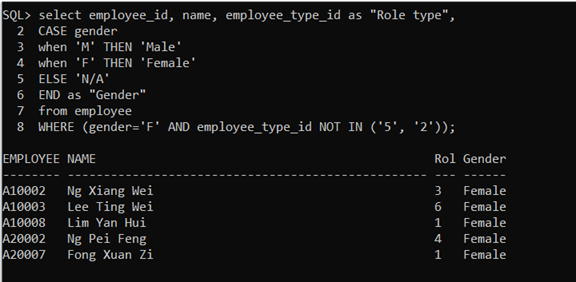
when 'F' THEN 'Female'

ELSE 'N/A'

END as "Gender"

from employee

WHERE (gender='F' AND employee\_type\_id NOT IN ('5', '2'));



This query changes the gender to become more meaningful in terms of words. Hence this select is to display all the female employees except trainee and part time employees. So that only full-time female employees can apply for maternity leave.

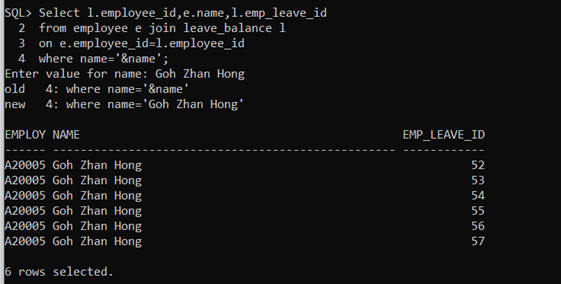
**Query 5:**

SELECT l.employee\_id,e.name,l.emp\_leave\_id

FROM employee e join leave\_balance l

ON e.employee\_id=l.employee\_id

WHERE name='&name';



This query enables the user to input the employee name to retrieve all the leave balance id from the selected employee.

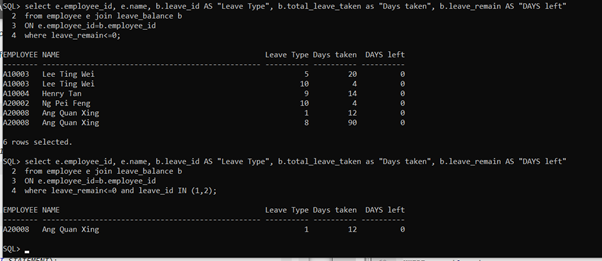
**Query 6:**

SELECT e.employee\_id, e.name, b.leave\_id AS "Leave Type", b.total\_leave\_taken AS "Days taken", b.leave\_remain AS "DAYS Left"

FROM employee e join leave\_balance b

ON e.employee\_id=b.employee\_id

WHERE leave\_remain<=0 AND leave\_id IN (1,2);



This query is to display the employee who has used up all their annual leave this year.

**Query 7:**

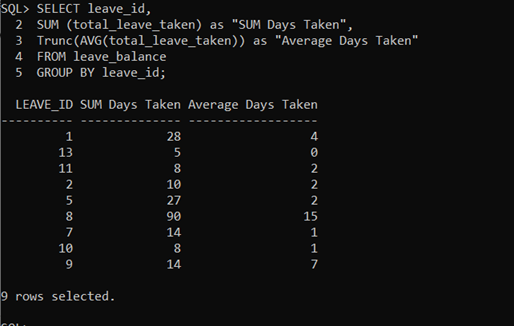
SELECT leave\_id,

SUM (total\_leave\_taken) as "SUM Days Taken",

Trunc(AVG(total\_leave\_taken)) as "Average Days Taken"

FROM leave\_balance

GROUP BY leave\_id;



This query is to calculate the sum and average days taken for each leave\_id.

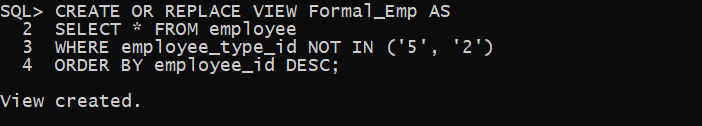
**Query 8:**

CREATE OR REPLACE VIEW Formal\_Emp AS

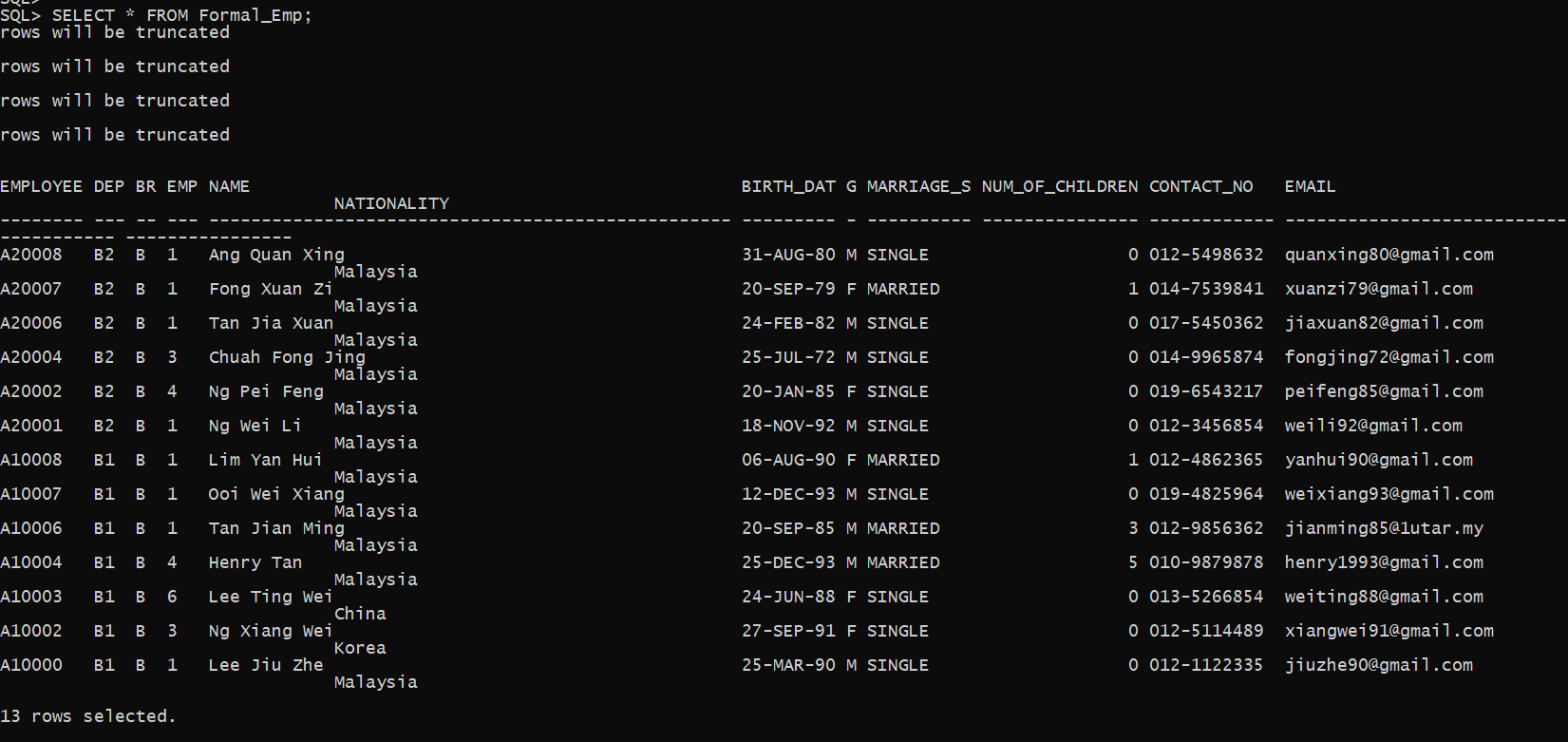
SELECT \* FROM employee

WHERE employee\_type\_id NOT IN ('5', '2')

ORDER BY employee\_id DESC;



SELECT \* FROM Formal\_Emp;



This view is used to enable the user to provide most of the leave to all the formal employees except part-time employees and trainees. Due to part-time employees and trainees can only apply unpaid leave which is listed in business rules.

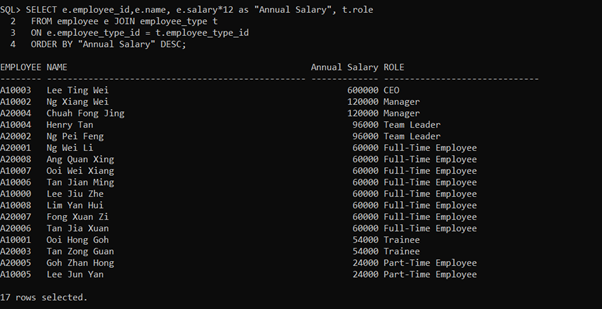
**Query 9:**

SELECT e.employee\_id,e.name, e.salary\*12 AS "Annual Salary", t.role

FROM employee e JOIN employee\_type t

ON e.employee\_type\_id = t.employee\_type\_id

ORDER BY “Annual Salary” DESC;



This query enables users to calculate the annual salary for each of the employees and together with the role position in the whole company.

**Query 10:**

CREATE OR REPLACE VIEW employee\_manager\_view AS

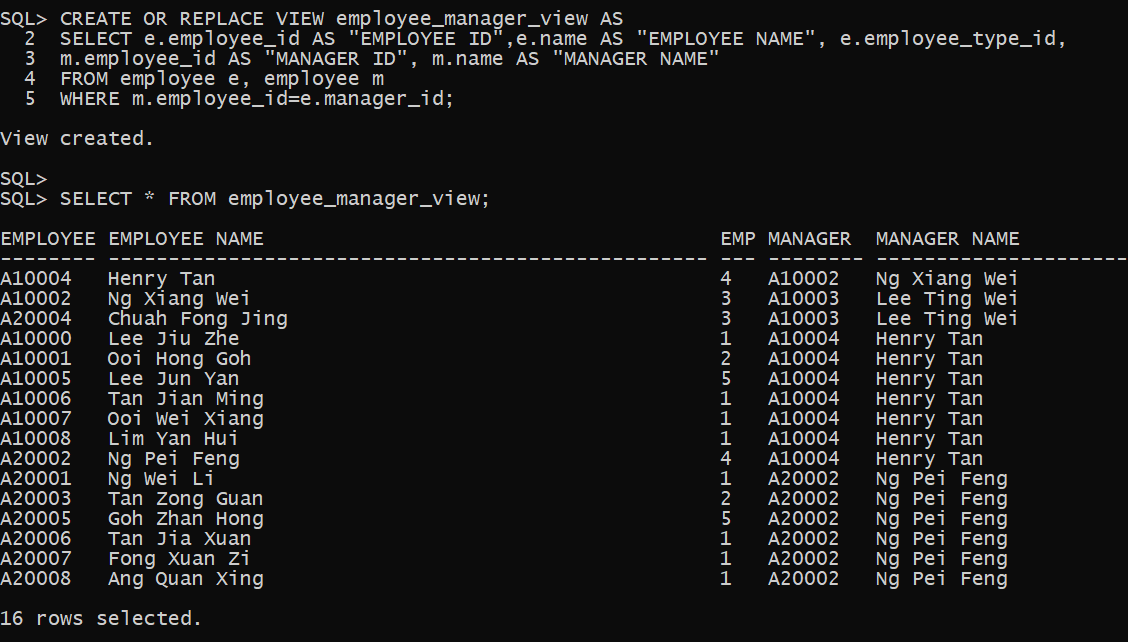
SELECT e.employee\_id AS "EMPLOYEE ID",e.name AS "EMPLOYEE NAME", e.employee\_type\_id,

m.employee\_id AS "MANAGER ID", m.name AS "MANAGER NAME"

FROM employee e, employee m

WHERE m.employee\_id=e.manager\_id;

SELECT \* FROM employee\_manager\_view;



This query enables users to view or find all the employees and listed them with their manager. So that in future this allows the manager to change their subordinate leave status.

**8.2 STORED PROCEDURE**

**Procedure 1:**

CREATE OR REPLACE PROCEDURE add\_branch

(

new\_branch\_id in VARCHAR,

new\_br\_name in VARCHAR,

new\_email in VARCHAR,

new\_est\_date in DATE,

new\_address in VARCHAR,

new\_phone\_no in VARCHAR

)

IS

BEGIN

INSERT INTO branch VALUES ( new\_branch\_id,

new\_br\_name ,

new\_email,

new\_est\_date,

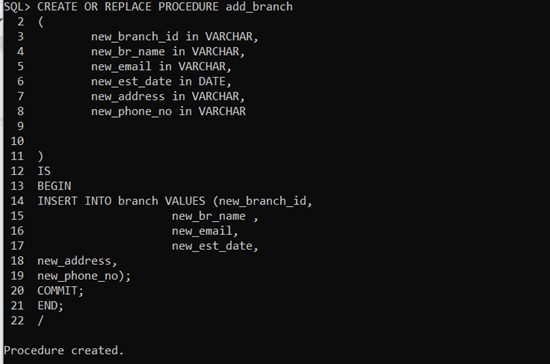
new\_address,

new\_phone\_no);

COMMIT;

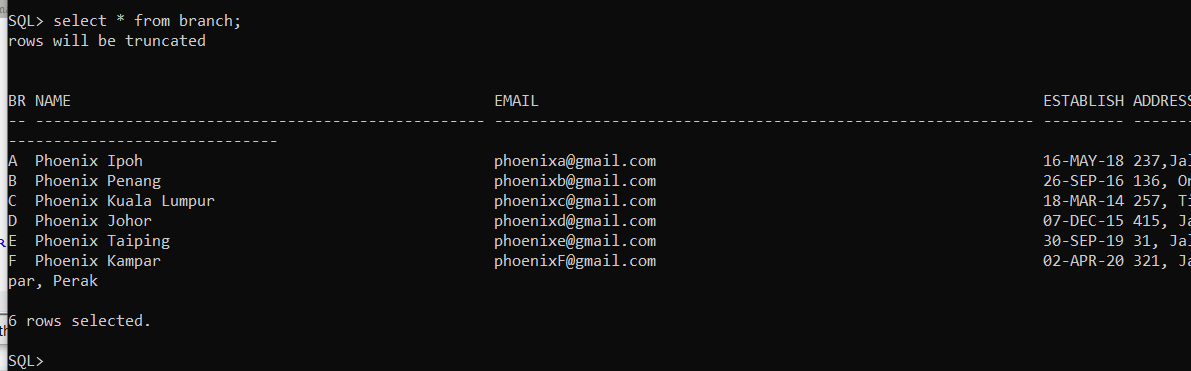
END;

/



Function: Allow user to add a new branch record.

Before the user add\_branch procedure.



A new record with branch\_id with ‘F’ is inserted after add the add\_branch procedure.

EXEC add\_branch('F', 'Phoenix Kampar', 'phoenixF@gmail.com', '2-APR-20', '321, Jalan Kampar 10, Taman Kampar, 31700 Kampar, Perak', '03-1235678');

**Procedure 2:**

CREATE OR REPLACE PROCEDURE add\_leave\_balance

(

new\_emp\_leave\_id in NUMBER,

new\_employee\_id in VARCHAR2,

new\_leave\_id in NUMBER,

new\_total\_leave\_taken in NUMBER,

new\_leave\_remain in NUMBER

)

IS

BEGIN

INSERT INTO leave\_balance VALUES (new\_emp\_leave\_id,

new\_employee\_id ,

new\_leave\_id,

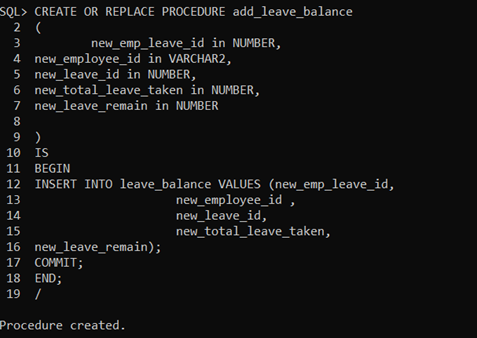
new\_total\_leave\_taken,

new\_leave\_remain);

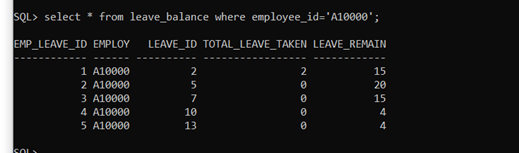
COMMIT;

END;

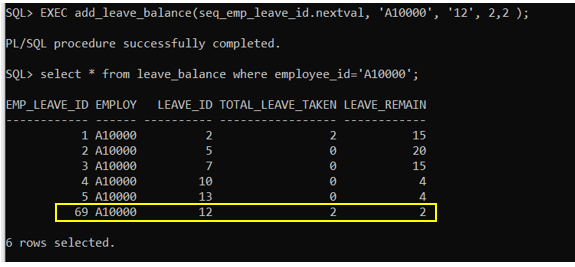
/



Function: Allow user to add a new branch record.



Before the user use add\_leave\_balance procedure.



EXEC add\_leave\_balance(seq\_emp\_leave\_id.nextval, 'A10000', '12', 2,2 );

A new record with emp\_leave\_id with ‘69’ is inserted after adding the add\_leave\_balance procedure.

**Procedure 3:**

CREATE OR REPLACE PROCEDURE update\_leave\_balance

(

selected\_emp\_leave\_id IN NUMBER,

new\_leave\_id IN NUMBER,

new\_total\_leave\_taken IN NUMBER

)

IS

BEGIN

UPDATE leave\_balance

SET leave\_id = new\_leave\_id ,

total\_leave\_taken = total\_leave\_taken + new\_total\_leave\_taken,

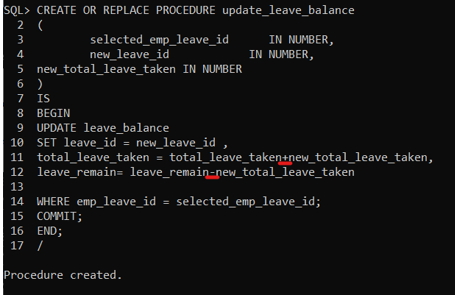
leave\_remain= leave\_remain - new\_total\_leave\_taken

WHERE emp\_leave\_id = selected\_emp\_leave\_id;

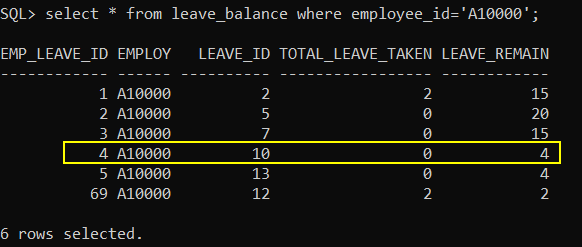
COMMIT;

END;

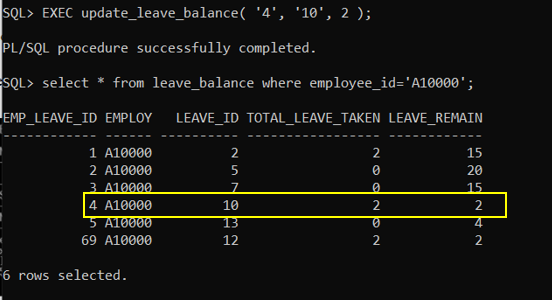
/



Function: Allow the user to modify the leave balance record



Before executing the update\_leave\_balance procedure.



After executing the procedure, the value of leave taken and remaining changed at emp\_leave\_id with ‘4’.

When user enters the leave\_taken value 2, it will automatically minus leave\_remain 2.

EXEC update\_leave\_balance( '4', '10', 2 );

EXEC update\_leave\_balance( emp\_leave\_id, leave\_id, total\_leave\_taken);

**Procedure 4:**

CREATE OR REPLACE PROCEDURE delete\_leave\_balance

(

selected\_emp\_leave\_id IN VARCHAR

)

IS

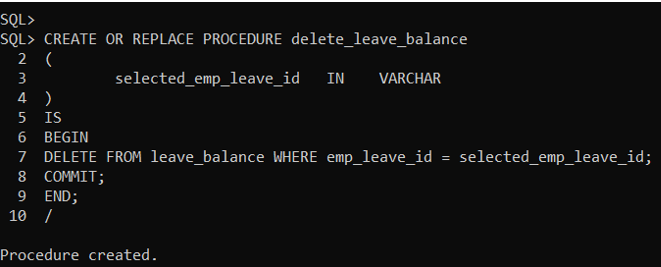
BEGIN

DELETE FROM leave\_balance WHERE emp\_leave\_id = selected\_emp\_leave\_id;

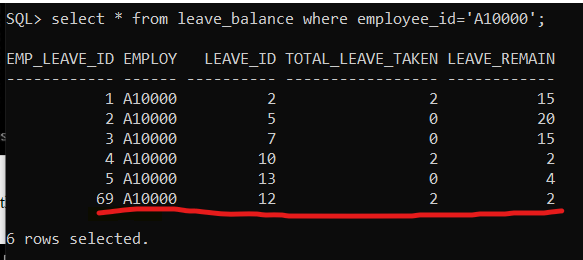
COMMIT;

END;

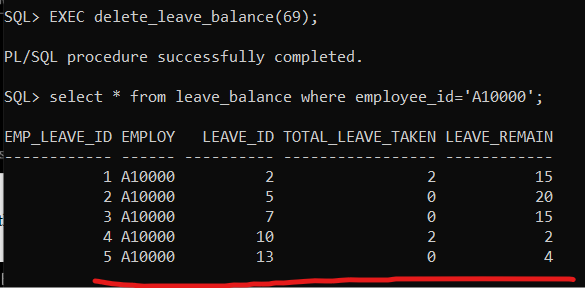
/



Function: To allow the user to delete the unused leave\_balance record faster.



Before executing the delete\_leave\_balance procedure.



After executing the procedure, the emp\_leave\_id with 69 has been removed.

EXEC delete\_leave\_balance(69);

EXEC delete\_leave\_balance(emp\_leave\_id);

**Procedure 5:**

CREATE OR REPLACE PROCEDURE deleteLeaveType

(selected\_leave\_id IN NUMBER)

IS

BEGIN

DELETE FROM leave\_balance where leave\_id =selected\_leave\_id AND exists( select \* from leave\_balance where leave\_id =selected\_leave\_id);

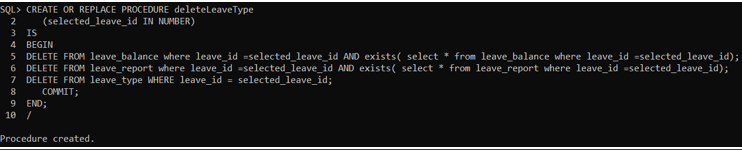
DELETE FROM leave\_report where leave\_id =selected\_leave\_id AND exists( select \* from leave\_report where leave\_id =selected\_leave\_id);

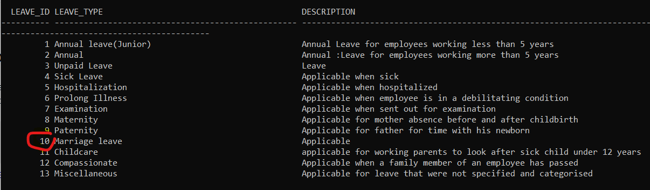
DELETE FROM leave\_type WHERE leave\_id = selected\_leave\_id;

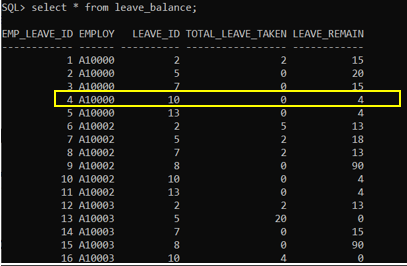
COMMIT;

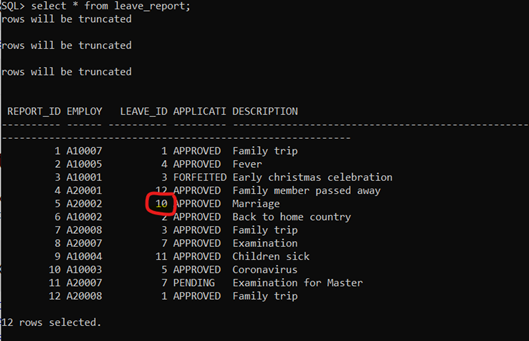
END;

/

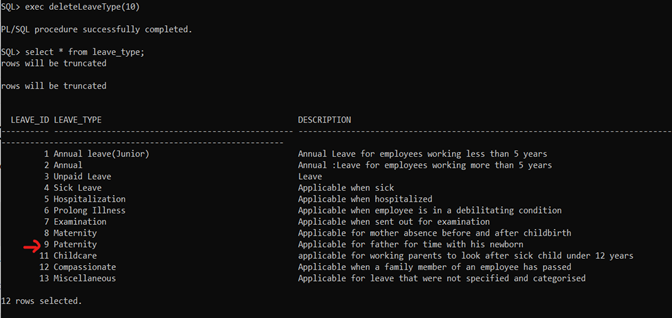


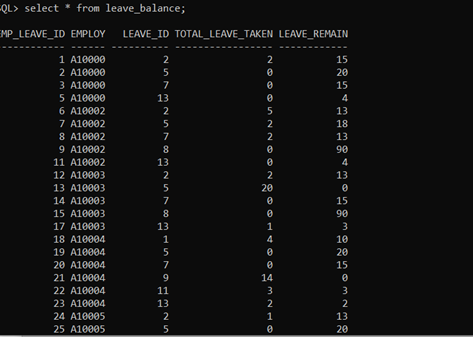


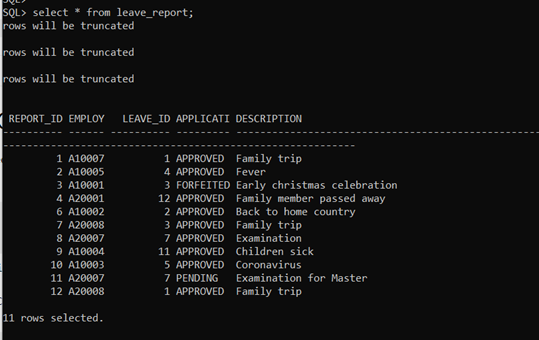




Before executing the deleteLeaveType procedure still contains leave\_id 10.







After executing the procedure, the leave\_id in 10 has been removed in three tables which are leave\_type, leave\_report and leave\_blalance.

exec deleteLeaveType(10)

**8.3 FUNCTIONS**

**Function 1:**

Create function:

CREATE OR REPLACE FUNCTION cal\_numLeave(selected\_employee\_id IN varchar2)

RETURN NUMBER

IS

total NUMBER;

BEGIN

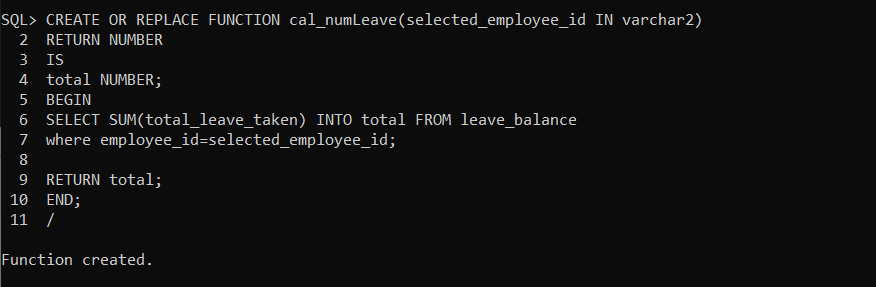
SELECT SUM(total\_leave\_taken) INTO total FROM leave\_balance

where employee\_id=selected\_employee\_id;

RETURN total;

END;

/



**Run function:**

DECLARE

total\_record NUMBER;

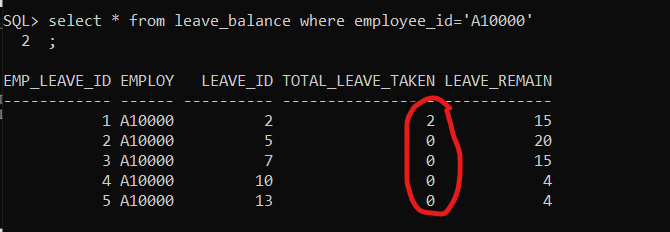
employee\_id varchar2(50);

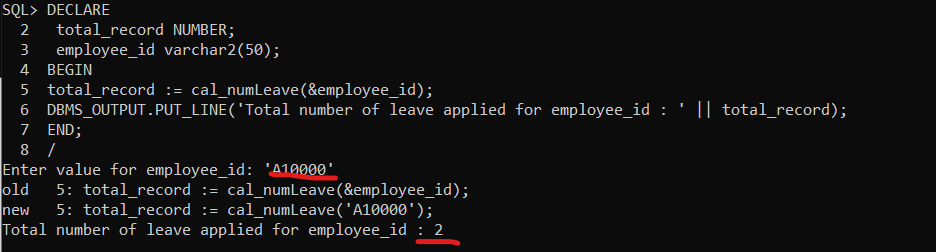
BEGIN

total\_record := cal\_numLeave(&employee\_id);

DBMS\_OUTPUT.PUT\_LINE('Total number of leave applied for employee\_id : ' || total\_record);

END;

/

**Explanation:**

From both the above diagrams we know that the function is performed successfully and displays the output of the total leave applied for employee A10000 as 2. This function is used to calculate the total number of leaves applied for the selected employee with the input of employee\_id as parameter.

**Function 2:**

Create function:

CREATE OR REPLACE FUNCTION GenderPercentage

RETURN VARCHAR2

IS

percentageM NUMBER(5,2);

percentageF NUMBER(5,2);

Total NUMBER;

Male NUMBER;

Female NUMBER;

BEGIN

SELECT COUNT(\*) into Total from employee;

SELECT COUNT(\*) into Male from employee

where gender='M';

SELECT COUNT(\*) into Female from employee

where gender='F';

percentageM:=Male/Total \* 100;

percentageF:=100-percentageM;

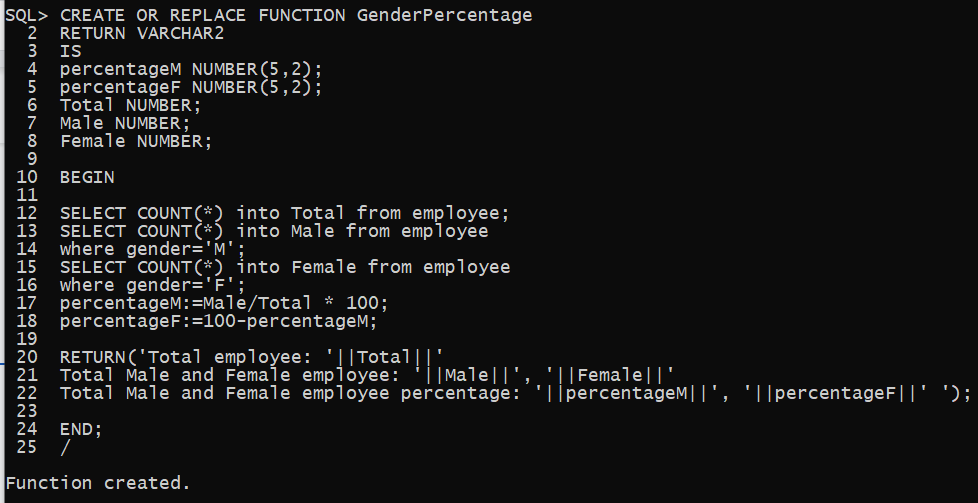
RETURN('Total employee: '||Total||'

Total Male and Female employee: '||Male||', '||Female||'

Total Male and Female employee percentage: '||percentageM||', '||percentageF||' ');

END;

/



Run function:

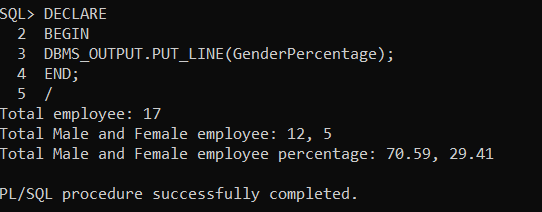
DECLARE

BEGIN

DBMS\_OUTPUT.PUT\_LINE(GenderPercentage);

END;

/



Explanation:

This function is to display and calculate the number of total employees, number of male and female employees and the percentages of both gender in the YY company. This result seems like an IT company.

**Function 3:**

Create function

CREATE OR REPLACE FUNCTION calcLeave(s\_leave\_id IN NUMBER)

RETURN VARCHAR2

IS

percentageT NUMBER(5,2);

percentageR NUMBER(5,2);

taken NUMBER;

remain NUMBER;

BEGIN

SELECT SUM(total\_leave\_taken) INTO taken FROM leave\_balance

WHERE leave\_id=s\_leave\_id;

SELECT SUM(leave\_remain) INTO remain FROM leave\_balance

WHERE leave\_id=s\_leave\_id;

percentageT:=taken/(taken+remain) \* 100;

percentageR:=100-percentageT;

RETURN('

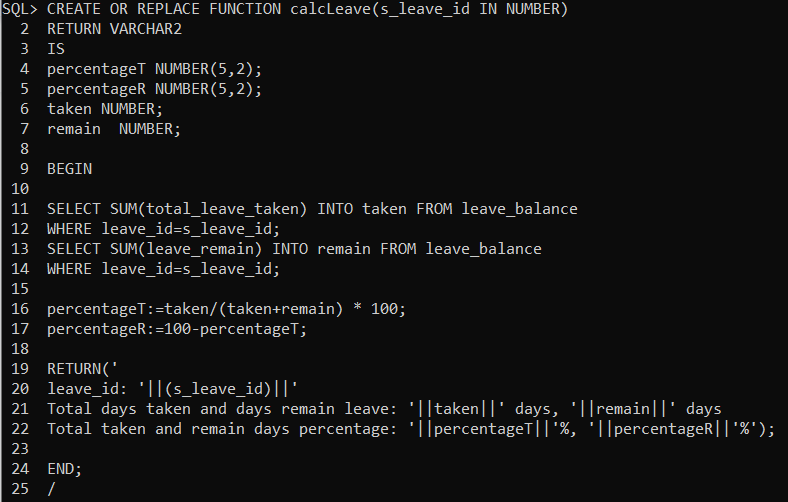
leave\_id: '||(s\_leave\_id)||'

Total days taken and days remain leave: '||taken||' days, '||remain||' days

Total taken and remain days percentage: '||percentageT||'%, '||percentageR||'%');

END;

/



Run function:

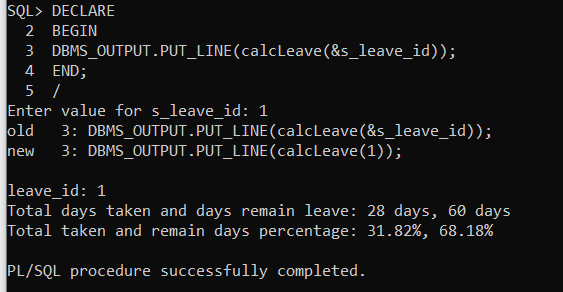
DECLARE

BEGIN

DBMS\_OUTPUT.PUT\_LINE(calcLeave(&s\_leave\_id));

END;

/



Explanation:

This function is used to display and calculate the number of leave days taken and days remaining. The users just need to input the leave id then they can get the percentages of days taken and remaining.

**Function 4:**

Create Function:

CREATE OR REPLACE FUNCTION ChildLeave(s\_emp\_id IN varchar)

RETURN VARCHAR2

IS

numC NUMBER;

maxDays NUMBER;

givenDays NUMBER;

BEGIN

SELECT num\_of\_children INTO numC FROM employee

WHERE employee\_id=s\_emp\_id;

SELECT leave\_days\_per\_year INTO maxDays FROM leave\_type

WHERE leave\_id=11;

givenDays:= numC\*2;

IF (givenDays>maxDays) THEN

givenDays:=maxDays;

END IF;

RETURN('

Selected Employee ID: '||(s\_emp\_id)||'

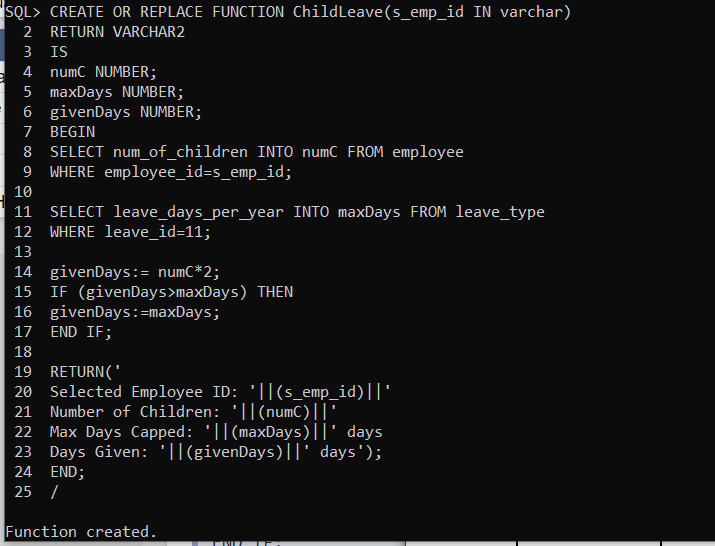
Number of Children: '||(numC)||'

Max Days Capped: '||(maxDays)||' days

Days Given: '||(givenDays)||' days');

END;

/



**Run Function:**

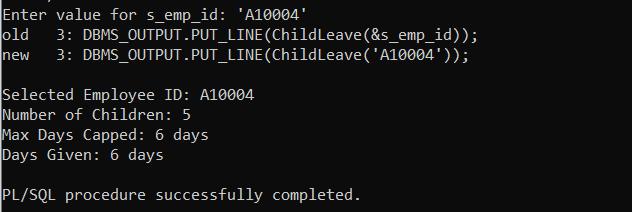
DECLARE

BEGIN

DBMS\_OUTPUT.PUT\_LINE(ChildLeave(&s\_emp\_id));

END;

/



**Explanation:**

If one day, the government comes out with a new policy for all the companies, the requirement is each employee will be given 2 days of childcare leave for each child's age which is below 12 years old. But the maximum capped is 6 days, for example if an employee has 2 children then he/she will get 4 days of childcare leave per year.

This function mainly calculates the days given to each employee who has at least one child.

**Function 5:**

**Create function:**

CREATE OR REPLACE FUNCTION EmptyAnnual

RETURN VARCHAR2

IS

finish NUMBER;

BEGIN

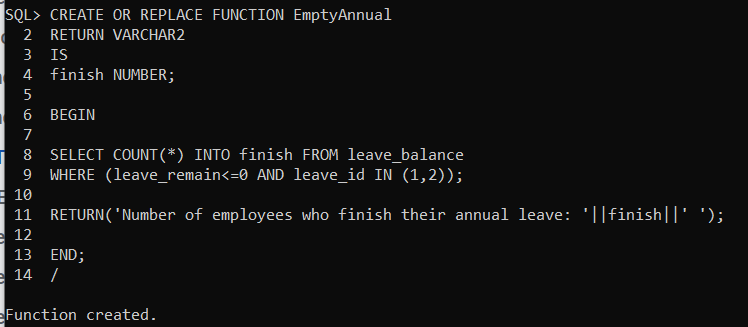
SELECT COUNT(\*) INTO finish FROM leave\_balance

WHERE (leave\_remain<=0 AND leave\_id IN (1,2));

RETURN('Number of employees who finish their annual leave: '||finish||' ');

END;

/



**Run function:**

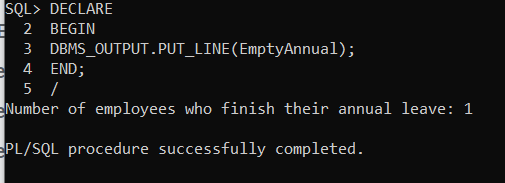
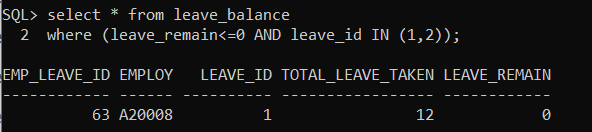
DECLARE

BEGIN

DBMS\_OUTPUT.PUT\_LINE(EmptyAnnual);

END;

/



**Explanation:**

This function is used to display the employee who finished their annual leave, then the employee has to take unpaid leave if he or she still wants to take more leave.

**9.0 Individual Assessment (Leong Hong Xin 17ACB04530)**

**9.1 QUERIES**

**Query 1:**

CREATE OR REPLACE VIEW branch\_view AS

SELECT branch\_id||'-'||branch\_name AS "Branches (ID-Name)",

establish\_date,

ROUND((SYSDATE - establish\_date)/365.25, 1) AS "UPTIME(YEARS)"

FROM branch;

SELECT \* FROM branch\_view;



Function: Allow users to use branch\_view to view all the branches and their established dates in years.

**Query 2:**

SELECT branch.branch\_id||'-'||branch.branch\_name AS "Branch (ID-Name)",

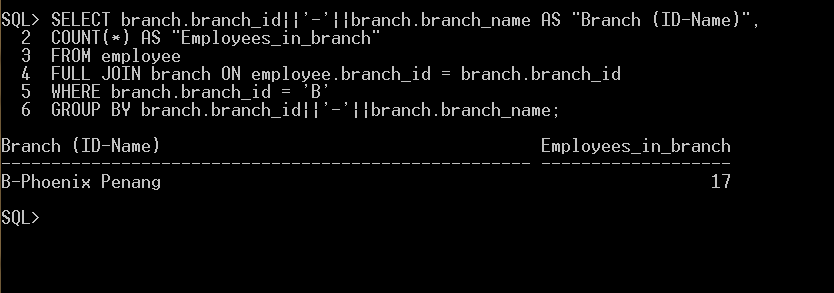
COUNT(\*) AS "Employees\_in\_branch"

FROM employee

FULL JOIN branch ON employee.branch\_id = branch.branch\_id

WHERE branch.branch\_id = 'B'

GROUP BY branch.branch\_id||'-'||branch.branch\_name;



Function: Allow users to check the number of employees within a branch using its ID.

**Query 3:**

UPDATE branch

SET phone\_no = '05324358968',

email = 'phoenixc@pnix.com'

WHERE branch\_id = 'C';



Function: Allows users to update branch contact details such as phone number and email.

**Query 4:**

CREATE OR REPLACE VIEW branch\_details AS

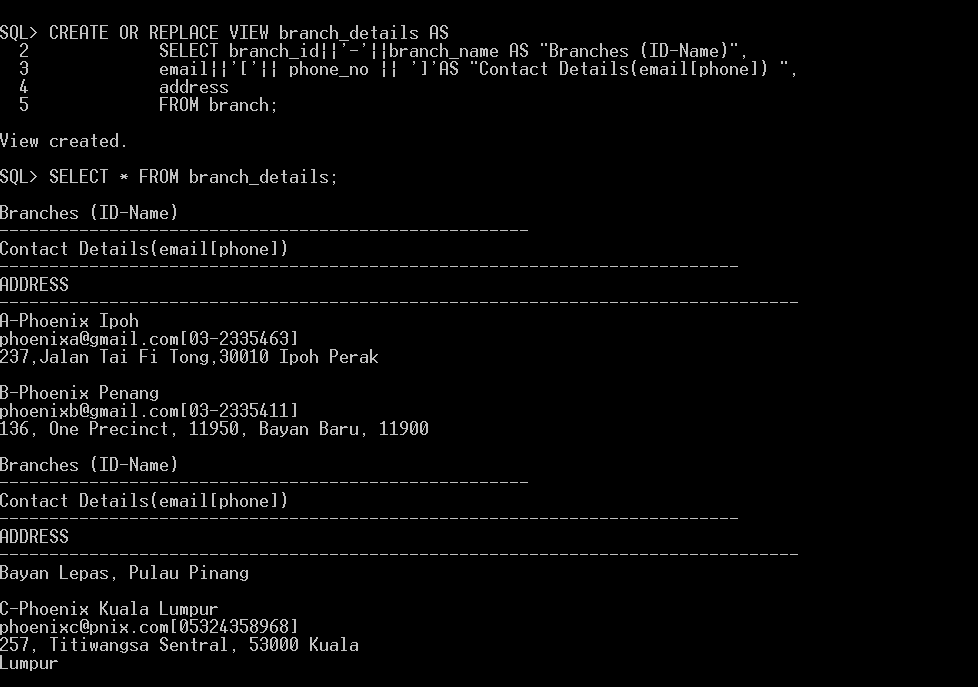
SELECT branch\_id||'-'||branch\_name AS "Branches (ID-Name)",

email||'['|| phone\_no || ']'AS "Contact Details(email[phone]) ",

address

FROM branch;

SELECT \* FROM branch\_details;



Function: Allow users to view all the branches’ contact details.

**Query 5:**

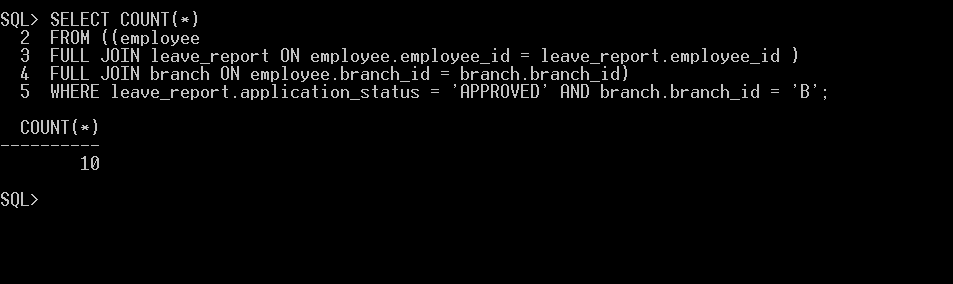
SELECT COUNT(\*)

FROM ((employee

FULL JOIN leave\_report ON employee.employee\_id = leave\_report.employee\_id )

FULL JOIN branch ON employee.branch\_id = branch.branch\_id)

WHERE leave\_report.application\_status = 'APPROVED' AND branch.branch\_id = 'B';



Function: Allow users to check the employees which are on leave in a given branch.

**Query 6:**

CREATE OR REPLACE VIEW dept\_in\_branch AS

SELECT branch.branch\_name,

branch.branch\_id AS "ID",

department.dept\_name AS "Department"

FROM department

FULL JOIN branch ON department.branch\_id = branch.branch\_id

ORDER BY branch.branch\_id;

SELECT \* FROM dept\_in\_branch;



Function: Allow users to view the branches of the company, their associative branch ID and the departments under each branch.

**Query 7:**

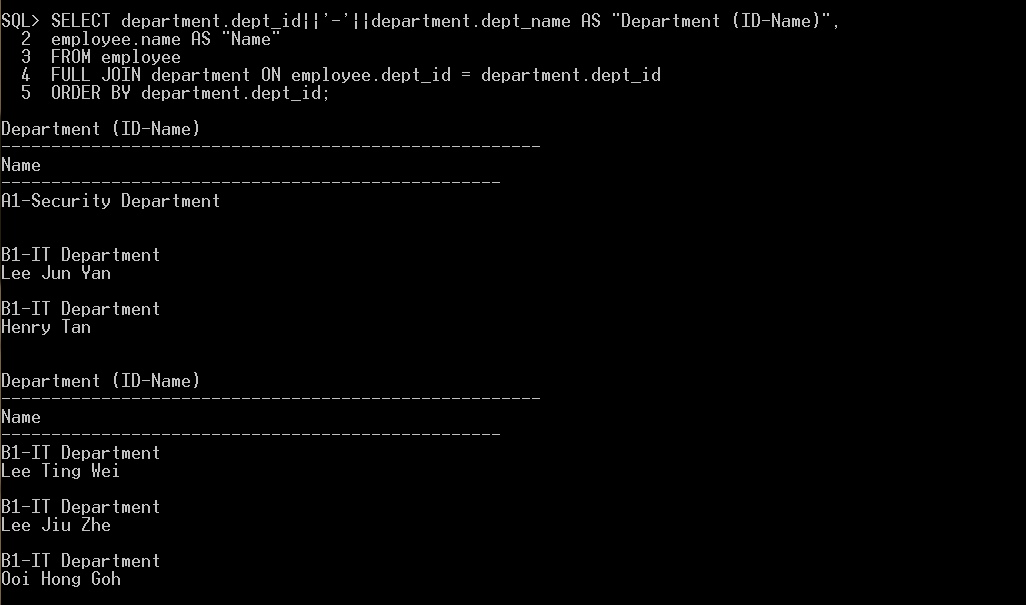
SELECT department.dept\_id||'-'||department.dept\_name AS "Department (ID-Name)",

employee.name AS "Name"

FROM employee

FULL JOIN department ON employee.dept\_id = department.dept\_id

ORDER BY department.dept\_id;



Function: Allow users to view all employees’ names and the department they are under.

**Query 8:**

SELECT department.dept\_name AS "Department",

COUNT(\*) AS "Employee on Leave"

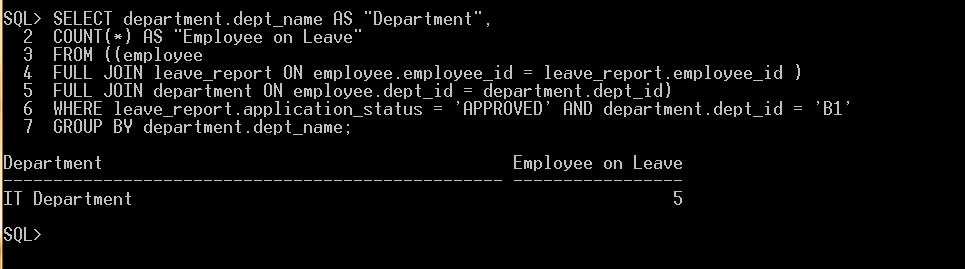
FROM ((employee

FULL JOIN leave\_report ON employee.employee\_id = leave\_report.employee\_id )

FULL JOIN department ON employee.dept\_id = department.dept\_id)

WHERE leave\_report.application\_status = 'APPROVED' AND department.dept\_id = 'B1'

GROUP BY department.dept\_name;



Function: Allow users to check the employees which are on leave in a given department.

**Query 9:**

SELECT branch.branch\_id||'-'||branch.branch\_name AS "Branches (ID-Name)",

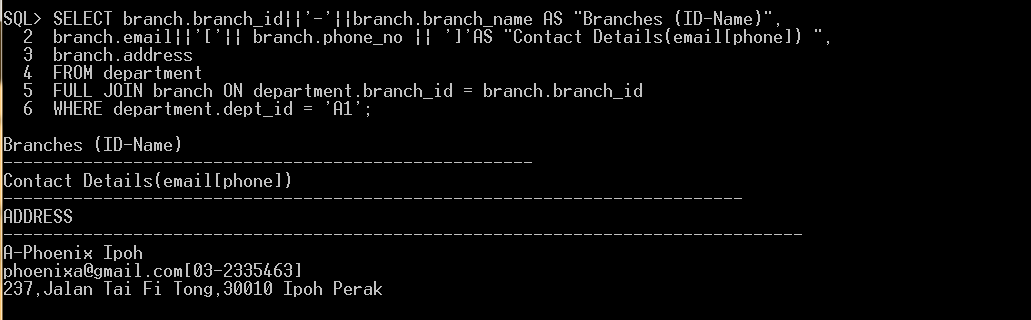
branch.email||'['|| branch.phone\_no || ']'AS "Contact Details(email[phone]) ",

branch.address

FROM department

FULL JOIN branch ON department.branch\_id = branch.branch\_id

WHERE department.dept\_id = 'A1';



Function: Allow users to obtain contact details of their branch using their department ID.

**Query 10:**

SELECT employee.name AS "Name",

department.dept\_name AS "Department",

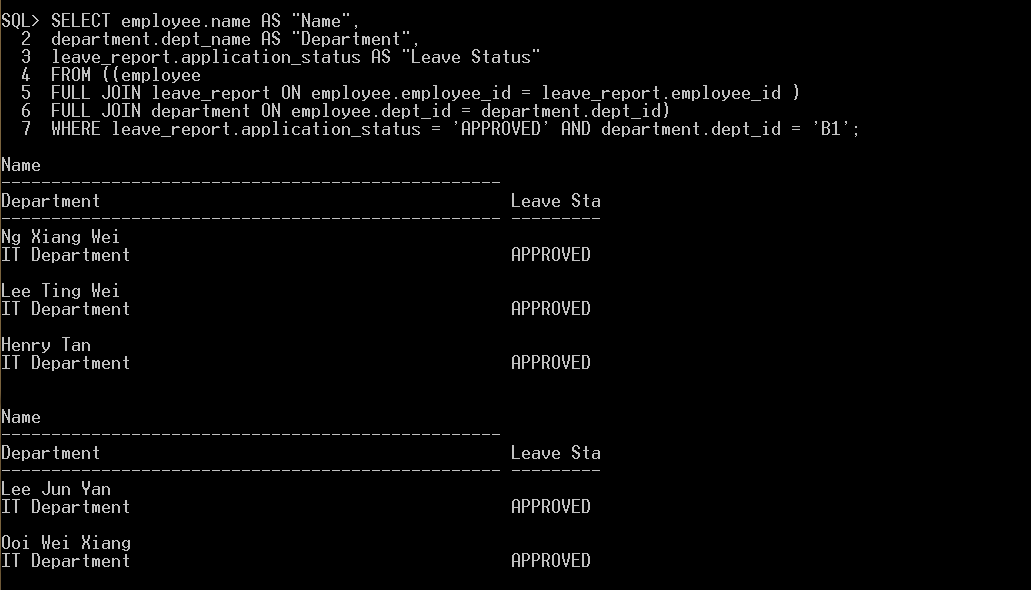
leave\_report.application\_status AS "Leave Status"

FROM ((employee

FULL JOIN leave\_report ON employee.employee\_id = leave\_report.employee\_id )

FULL JOIN department ON employee.dept\_id = department.dept\_id)

WHERE leave\_report.application\_status = 'APPROVED' AND department.dept\_id = 'B1';



Function: Allow users to view employees that are on leave under a given department.

**9.2 STORED PROCEDURES:**

**Procedure 1:**

CREATE OR REPLACE PROCEDURE add\_dept

(

new\_dept\_id in VARCHAR,

new\_branch\_id in VARCHAR,

new\_dept\_name in VARCHAR,

new\_dept\_total\_emp in INT

)

IS

BEGIN

INSERT INTO department VALUES (new\_dept\_id,

new\_branch\_id,

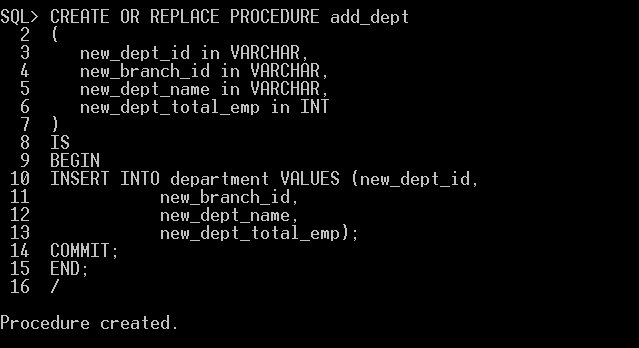
new\_dept\_name,

new\_dept\_total\_emp);

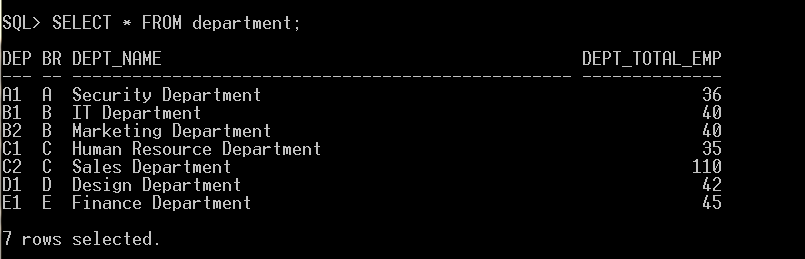
COMMIT;

END;

/

Function: Allows user to add a new department record.

Before the use of add\_dept procedure.

After add\_dept procedure.

**Procedure 2:**

CREATE OR REPLACE PROCEDURE update\_branch\_contacts

(

selected\_branch\_id IN VARCHAR,

new\_email IN VARCHAR,

new\_address IN VARCHAR,

new\_phone\_no IN VARCHAR

)

IS

BEGIN

UPDATE branch

SET email = new\_email,

address = new\_address,

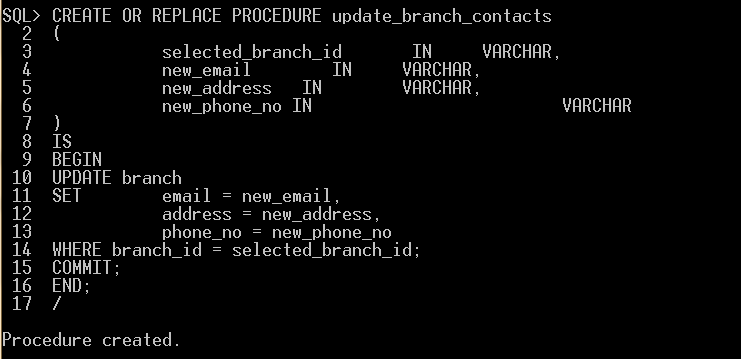
phone\_no = new\_phone\_no

WHERE branch\_id = selected\_branch\_id;

COMMIT;

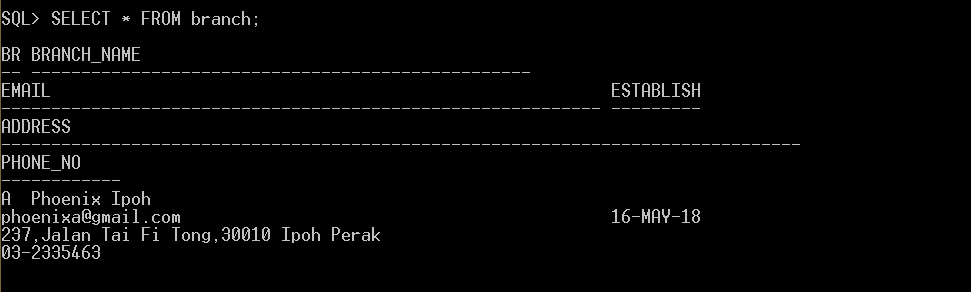
END;

/



Function: Allows user to update contact details of a branch.

Before the use of update\_branch\_contacts procedure.



After update\_branch\_contacts procedure.



**Procedure 3:**

CREATE OR REPLACE PROCEDURE dept\_change\_branch

(

selected\_dept\_id IN VARCHAR,

new\_branch\_id IN VARCHAR,

new\_dept\_id IN VARCHAR

)

IS

BEGIN

UPDATE department

SET branch\_id = new\_branch\_id,

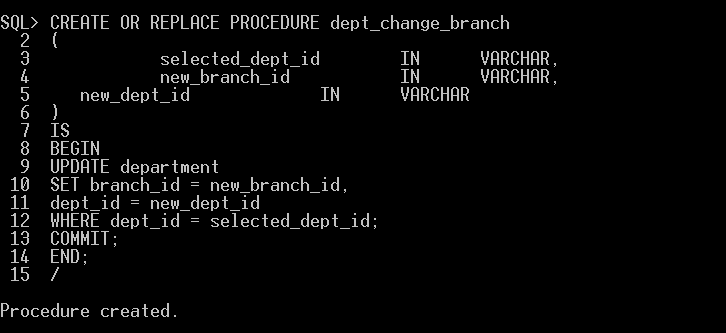
dept\_id = new\_dept\_id

WHERE dept\_id = selected\_dept\_id;

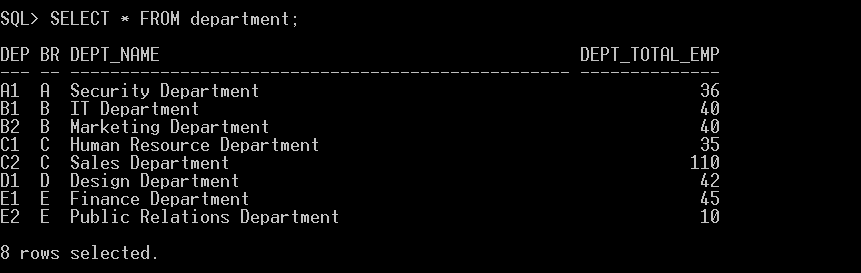
COMMIT;

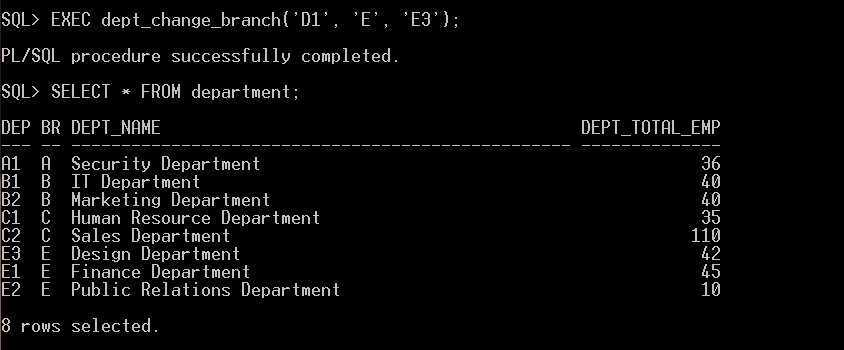
END;

/



Function: Allows user to change the branch of a department when needed.

Before the use of dept\_change\_branch.

After the use of dept\_change\_branch.

**Procedure 4:**

CREATE OR REPLACE PROCEDURE delete\_dept

(

selected\_dept\_id IN VARCHAR

)

IS

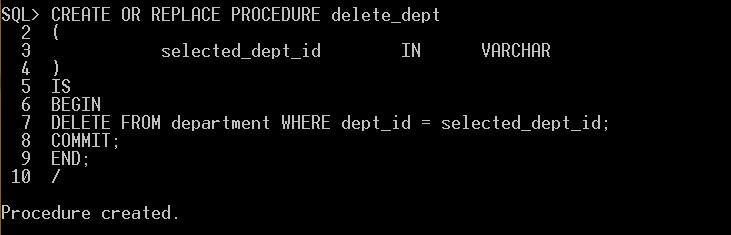
BEGIN

DELETE FROM department WHERE dept\_id = selected\_dept\_id;

COMMIT;

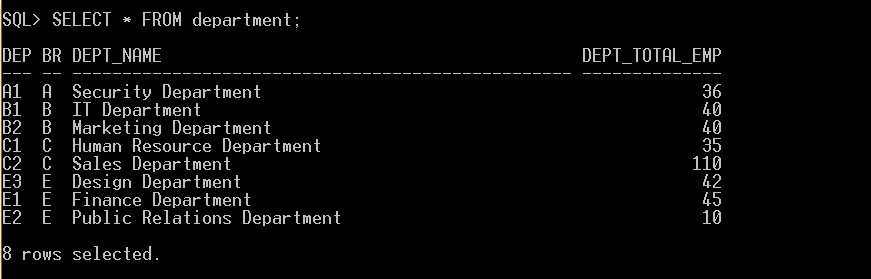
END;

/

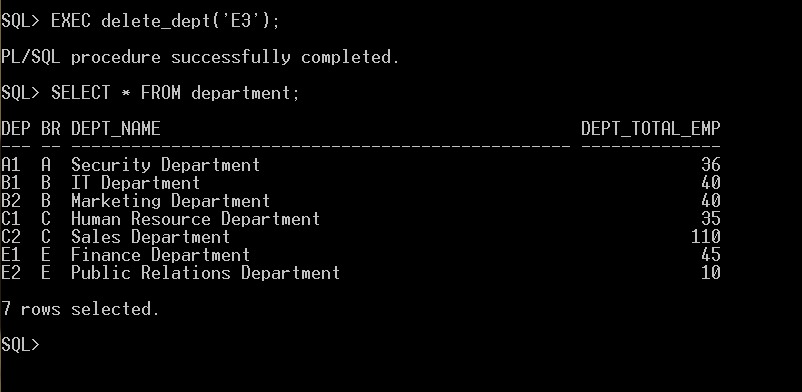


Function: Allows user to delete a department when it is to be dissolved.

Before the use of delete\_dept.



After the use of delete\_dept.



**Procedure 5:**

CREATE OR REPLACE PROCEDURE delete\_branch

(

selected\_branch\_id IN VARCHAR

)

IS

dept\_count NUMBER;

BEGIN

SELECT COUNT(\*) INTO dept\_count from department WHERE branch\_id = selected\_branch\_id;

IF (dept\_count != 0)

THEN

DBMS\_OUTPUT.PUT\_LINE('Branch not empty.');

COMMIT;

ELSE

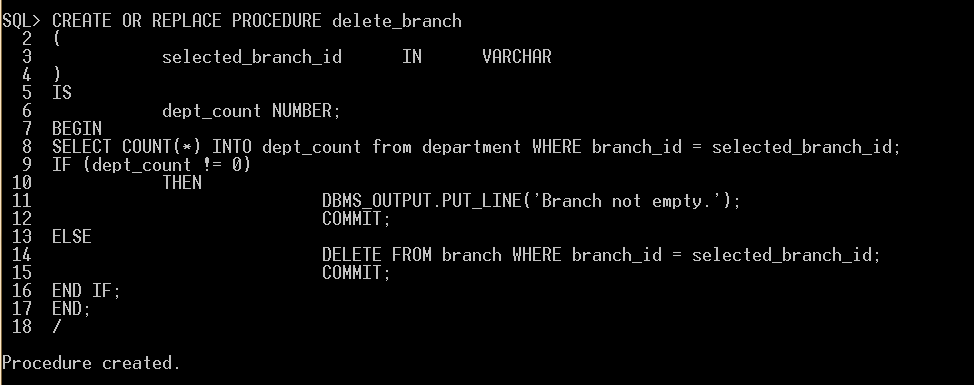
DELETE FROM branch WHERE branch\_id = selected\_branch\_id;

COMMIT;

END IF;

END;

/

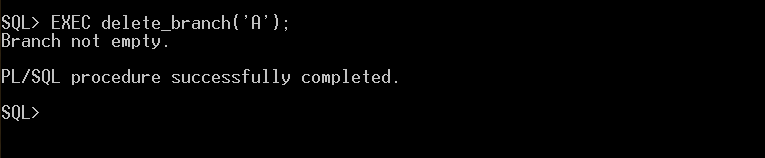


Function: Allows user to delete a branch entry when all departments under it has been dissolved.

Before the use of delete\_branch.

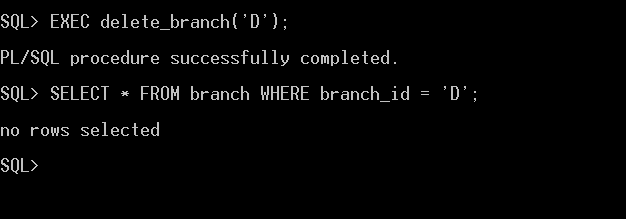


If delete\_branch is used on a non-empty branch. (e.g. Branch A)





If delete\_branch is used on an empty branch. (e.g. Branch D).



**9.3 FUNCTIONS:**

**Function 1:**

CREATE OR REPLACE FUNCTION branch\_emp\_no

(given\_branch\_id IN VARCHAR)

RETURN NUMBER

IS

current\_emp\_no NUMBER;

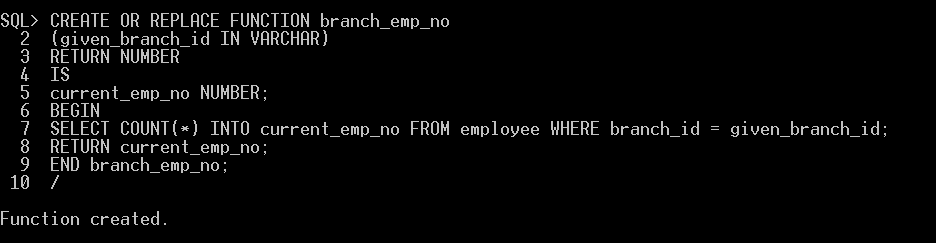
BEGIN

SELECT COUNT(\*) INTO current\_emp\_no FROM employee WHERE branch\_id = given\_branch\_id;

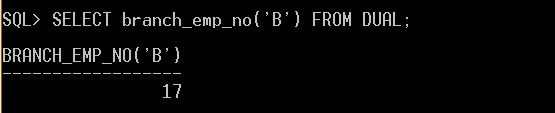
RETURN current\_emp\_no;

END branch\_emp\_no;

/



Function: Takes branch\_id as a parameter and returns the number of employees under the current branch.



**Function 2:**

CREATE OR REPLACE FUNCTION branch\_emp\_on\_leave

(given\_branch\_id IN VARCHAR)

RETURN NUMBER

IS

leave\_emp\_no NUMBER;

BEGIN

SELECT COUNT(\*) INTO leave\_emp\_no

FROM ((employee

FULL JOIN leave\_report ON employee.employee\_id = leave\_report.employee\_id )

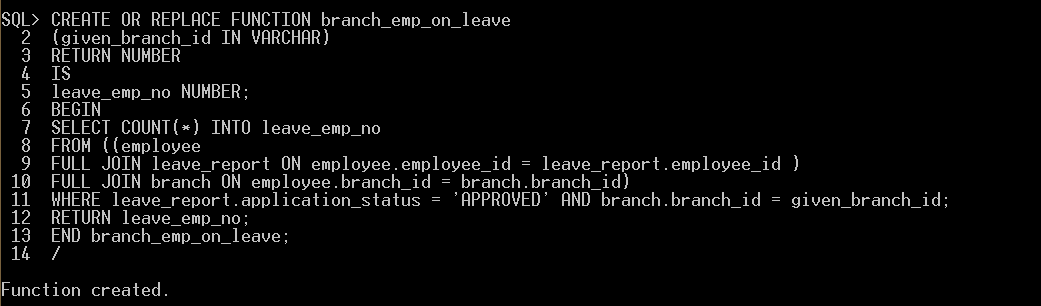
FULL JOIN branch ON employee.branch\_id = branch.branch\_id)

WHERE leave\_report.application\_status = 'APPROVED' AND branch.branch\_id = given\_branch\_id;

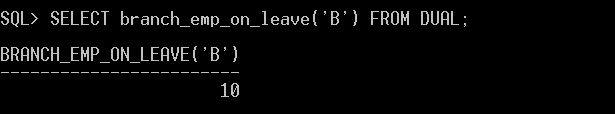
RETURN leave\_emp\_no;

END branch\_emp\_on\_leave;

/



Function: Takes branch\_id as a parameter and returns the number of employees under the current branch that is currently on leave.



**Function 3:**

CREATE OR REPLACE FUNCTION available\_emp\_in\_branch

(given\_branch\_id IN VARCHAR)

RETURN NUMBER

IS

leave\_emp\_no NUMBER;

available\_emp\_no NUMBER;

BEGIN

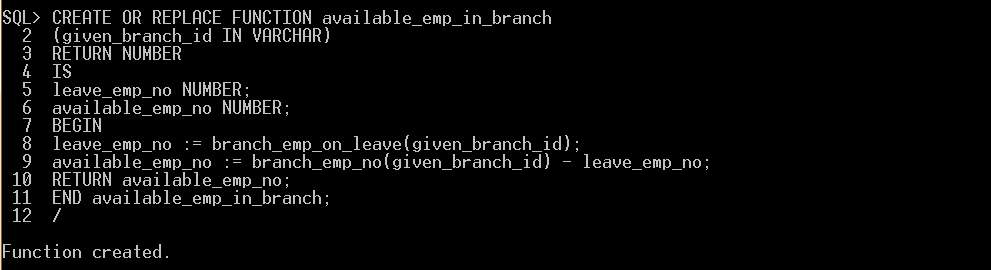
leave\_emp\_no := branch\_emp\_on\_leave(given\_branch\_id);

available\_emp\_no := branch\_emp\_no(given\_branch\_id) - leave\_emp\_no;

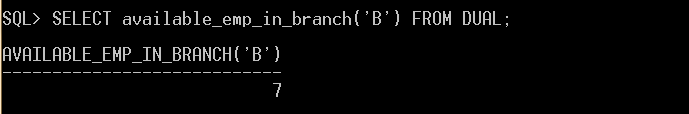
RETURN available\_emp\_no;

END available\_emp\_in\_branch;

/



Function: Takes branch\_id as a parameter and returns the number of employees that are available and not on leave of the current branch.



**Function 4:**

CREATE OR REPLACE FUNCTION dept\_emp\_on\_leave

(given\_dept\_id IN VARCHAR)

RETURN NUMBER

IS

leave\_emp\_no NUMBER;

BEGIN

SELECT COUNT(\*) INTO leave\_emp\_no

FROM ((employee

FULL JOIN leave\_report ON employee.employee\_id = leave\_report.employee\_id )

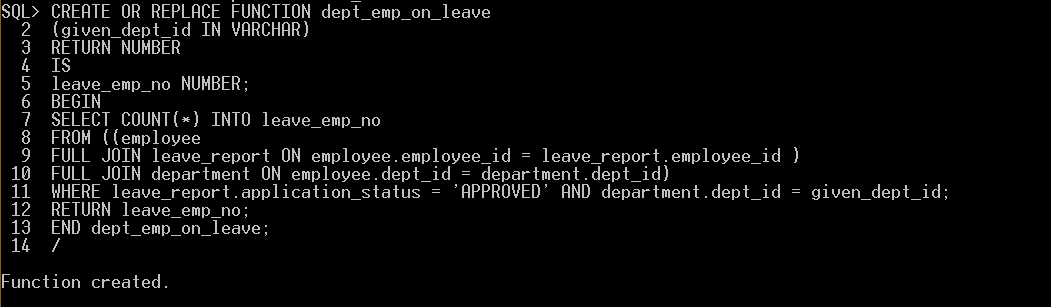
FULL JOIN department ON employee.dept\_id = department.dept\_id)

WHERE leave\_report.application\_status = 'APPROVED' AND department.dept\_id = given\_dept\_id;

RETURN leave\_emp\_no;

END dept\_emp\_on\_leave;

/



Function: Takes dept\_id as a parameter and returns the number of employees on leave under the current department.



**Function 5:**

CREATE OR REPLACE FUNCTION available\_emp\_in\_dept

(given\_dept\_id IN VARCHAR)

RETURN NUMBER

IS

current\_emp\_no NUMBER;

leave\_emp\_no NUMBER;

available\_emp\_no NUMBER;

BEGIN

SELECT COUNT(\*) INTO current\_emp\_no FROM employee WHERE dept\_id = given\_dept\_id;

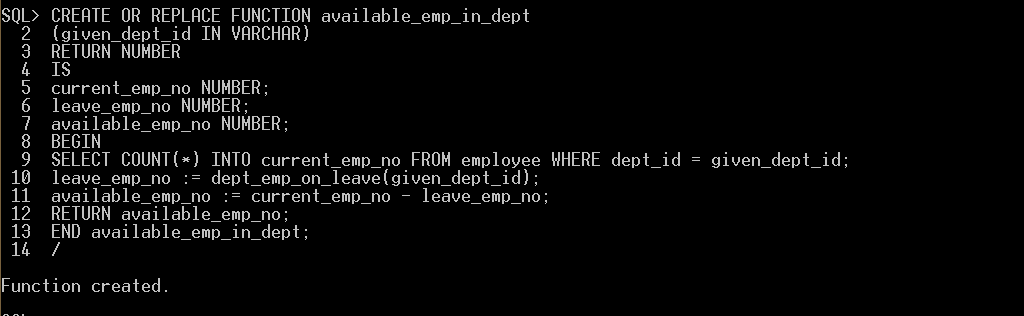
leave\_emp\_no := dept\_emp\_on\_leave(given\_dept\_id);

available\_emp\_no := current\_emp\_no - leave\_emp\_no;

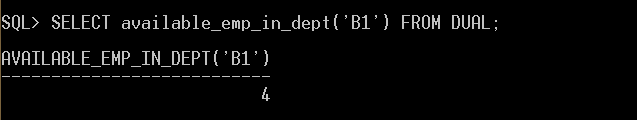
RETURN available\_emp\_no;

END available\_emp\_in\_dept;

/



Function: Takes dept\_id as a parameter and returns the number of employees that are available and not on leave under the current department.



**10.0 Individual Assessment (Lam Kean Chin 17ACB05143)**

**10.1 QUERIES**

**Query 1:**

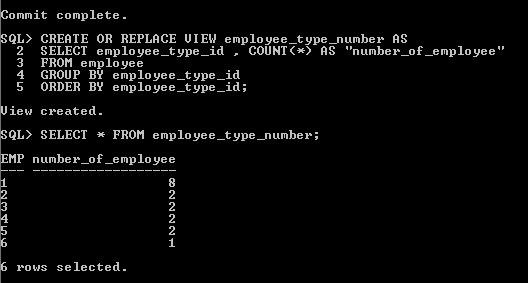
CREATE OR REPLACE VIEW employee\_type\_number AS

SELECT employee\_type\_id , COUNT(\*) AS "number\_of\_employee"

FROM employee

GROUP BY employee\_type\_id

ORDER BY employee\_type\_id;

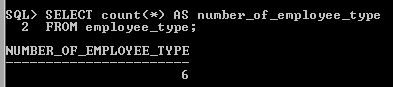


Function: Allow the user to view the number of types of employee existing in the company.

**Query 2:**

SELECT count(\*) AS number\_of\_employee\_type

FROM employee\_type;

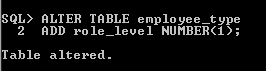


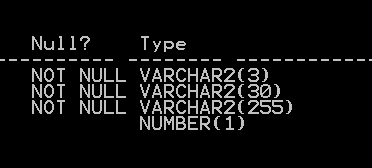
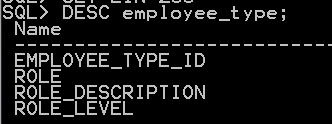
Function: Allow the user to view how many types of employee exist in the company.

**Query 3:**

ALTER TABLE employee\_type

ADD role\_level NUMBER(1);





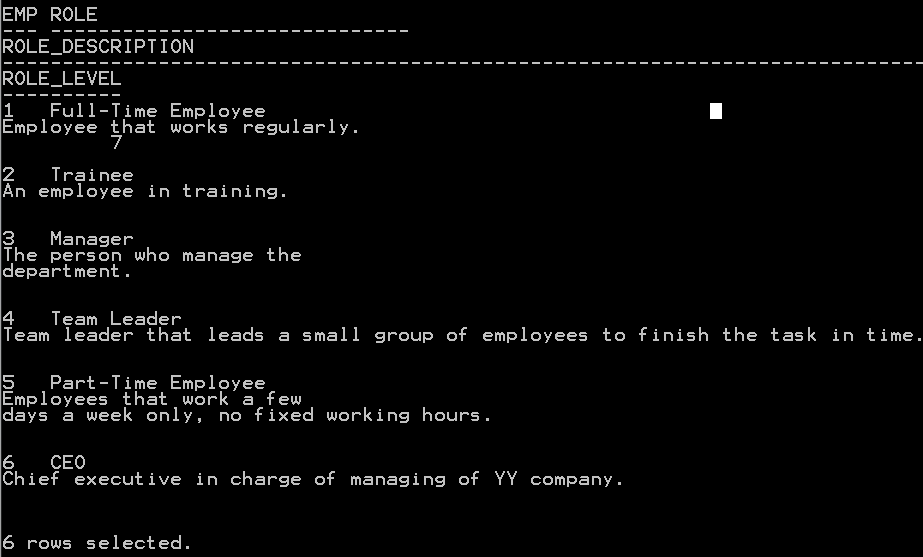
Function: Query that let the user insert a new column that stores the role level of the employee type.

**Query 4:**

UPDATE employee\_type

SET role\_level = 7

WHERE employee\_type\_id = 1;



Function: Query that is used to insert the role level of the employee type.

**Query 5:**

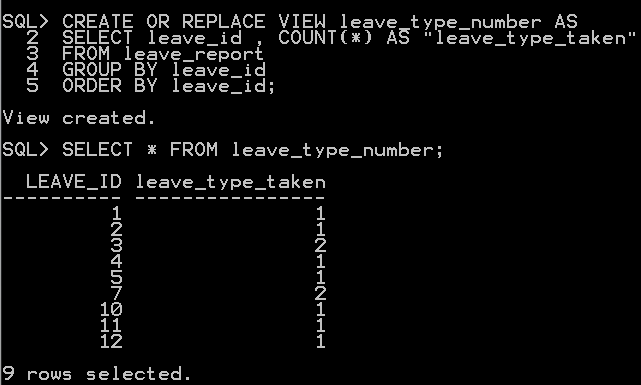
CREATE OR REPLACE VIEW leave\_type\_number AS

SELECT leave\_id , COUNT(\*) AS "leave\_type\_taken"

FROM leave\_report

GROUP BY leave\_id

ORDER BY leave\_id;

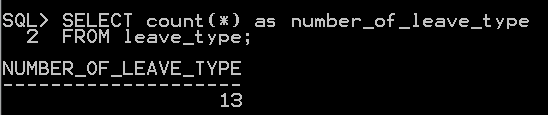


Function: The user will be able to view the number of leave types taken in the company to know what kind of leave is take most often.

**Query 6:**

SELECT count(\*) as number\_of\_leave\_type

FROM leave\_type;

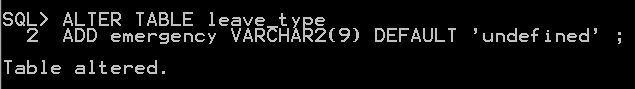


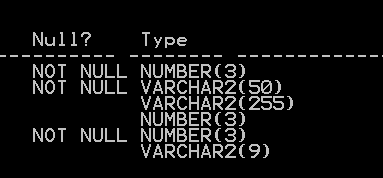
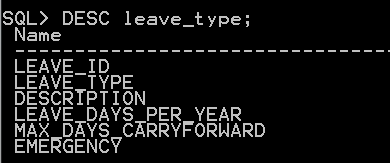
Function: The user will be able to view how many leave types exist in the company.

**Query 7:**

ALTER TABLE leave\_type

ADD emergency VARCHAR2(9) DEFAULT 'undefined' ;





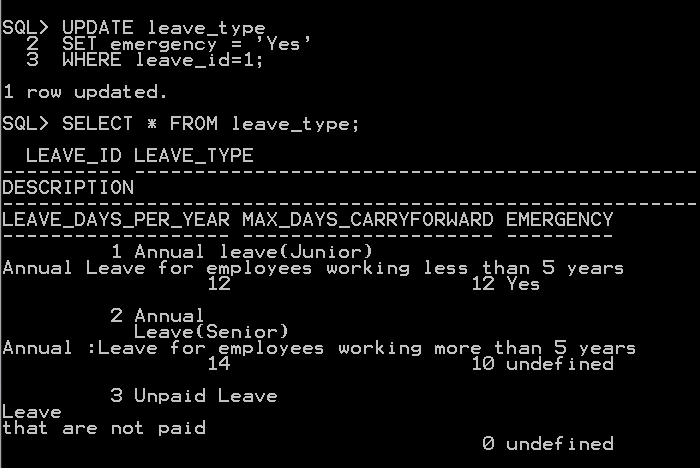
Function: Query that lets the user to insert a new column into emergency with a default value undefined in it.

**Query 8:**

UPDATE leave\_type

SET emergency = 'Yes'

WHERE leave\_id=1;



Function: Query that lets the user to update the emergency status in the leave\_type table.

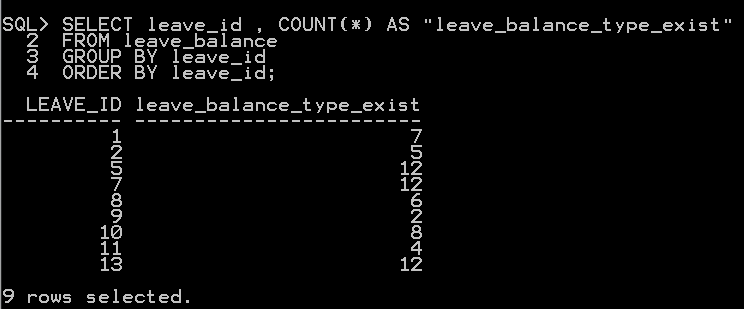
**Query 9:**

SELECT leave\_id , COUNT(\*) AS "leave\_balance\_type\_exist"

FROM leave\_balance

GROUP BY leave\_id

ORDER BY leave\_id;



Function: The user will be able to view how many leave types with balance are used in this company.

**Query 10:**

SELECT employee\_id, name,

SUBSTR(name, 1, instr(name, ' ') - 1) AS first\_name,

SUBSTR (name, instr(name, ' ') + 1) AS last\_name

FROM employee;



Function: The user will be able to separate the employee name with first name and last name.

**10.2 STORED PROCEDURES:**

**Procedure 1:**

CREATE OR REPLACE PROCEDURE add\_employee\_type

(

new\_employee\_type\_id in VARCHAR,

new\_role in VARCHAR,

new\_role\_description in VARCHAR)

IS BEGIN

INSERT INTO employee\_type VALUES

(

new\_employee\_type\_id,

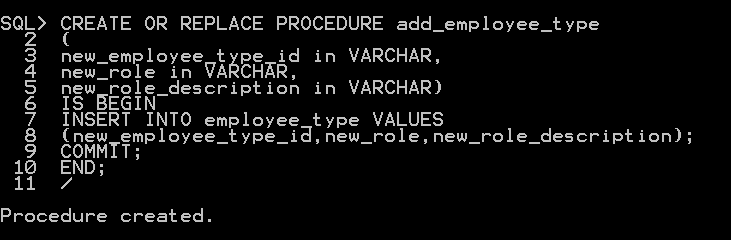
new\_role,

new\_role\_description);

COMMIT;

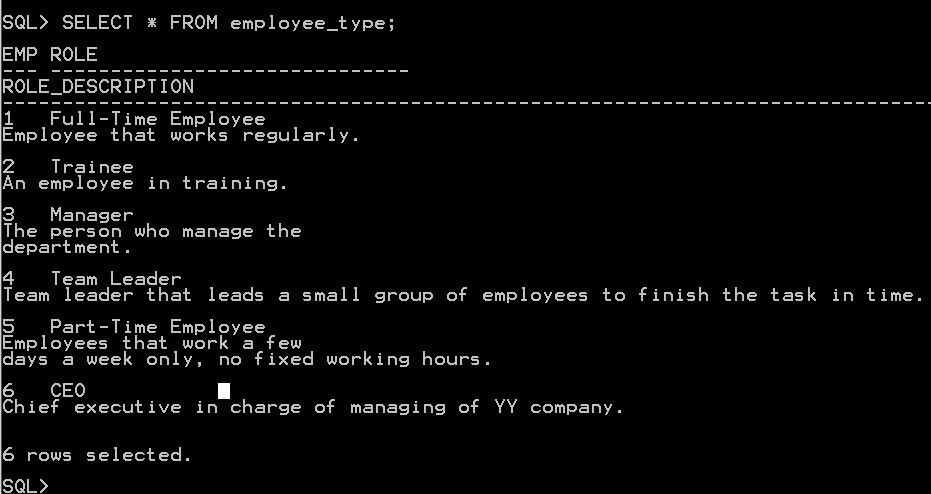
END;

/

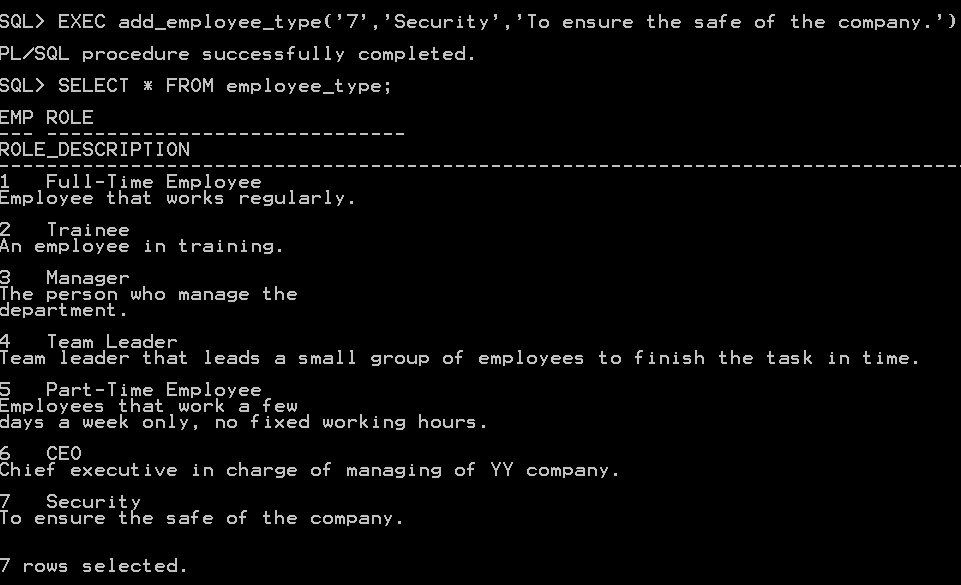


Function: Allow the user to insert new employee type into employee type table using procedure.

Before using the add\_employee\_type procedure.



After the add\_employee\_type procedure.



**Procedure 2:**

CREATE OR REPLACE PROCEDURE update\_employee\_type

(

selected\_employee\_type\_id in VARCHAR,

new\_role in VARCHAR,

new\_role\_description in VARCHAR)

IS BEGIN

UPDATE employee\_type

SET

role=new\_role,

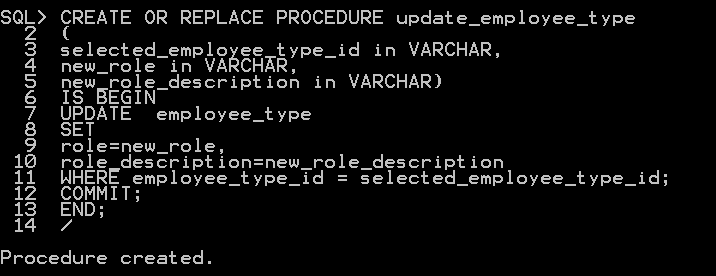
role\_description=new\_role\_description

WHERE employee\_type\_id = selected\_employee\_type\_id;

COMMIT;

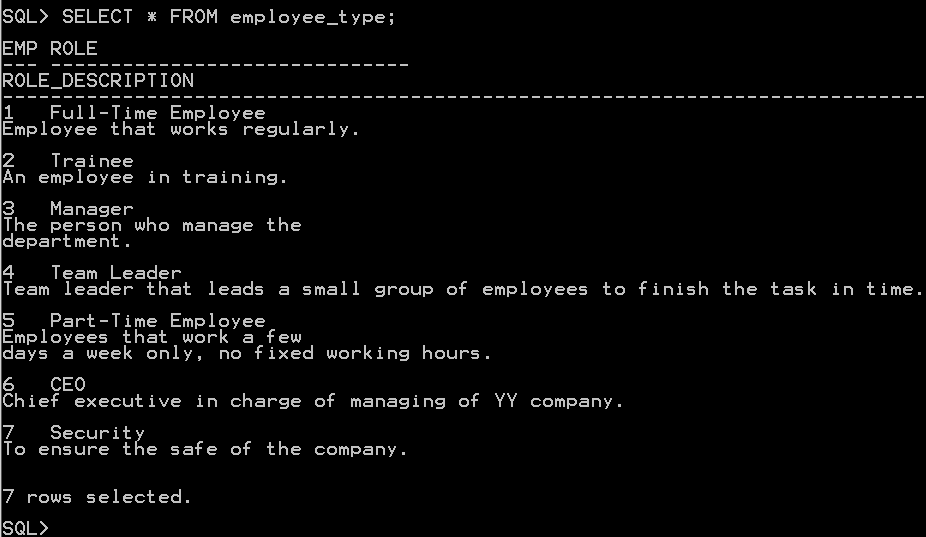
END;

/

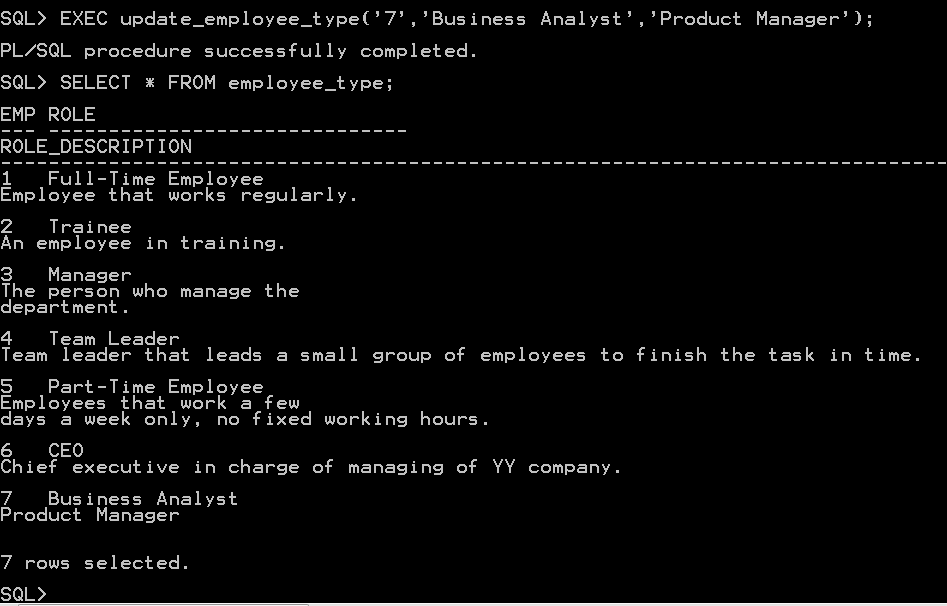


Function: Allow the user to update a selected employee type details with id.

Before using the update\_employee\_type procedure.



After using the update\_employee\_type procedure.



**Procedure 3:**

CREATE OR REPLACE PROCEDURE

delete\_employee\_type

(

selected\_employee\_type\_id in VARCHAR

)

IS BEGIN

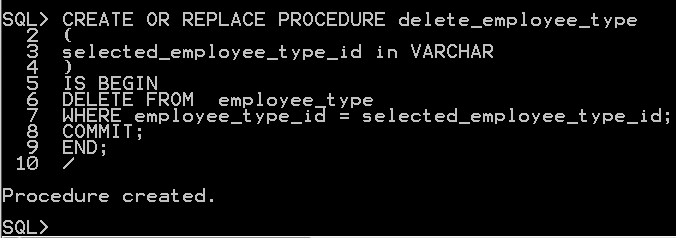
DELETE FROM employee\_type

WHERE employee\_type\_id = selected\_employee\_type\_id;

COMMIT;

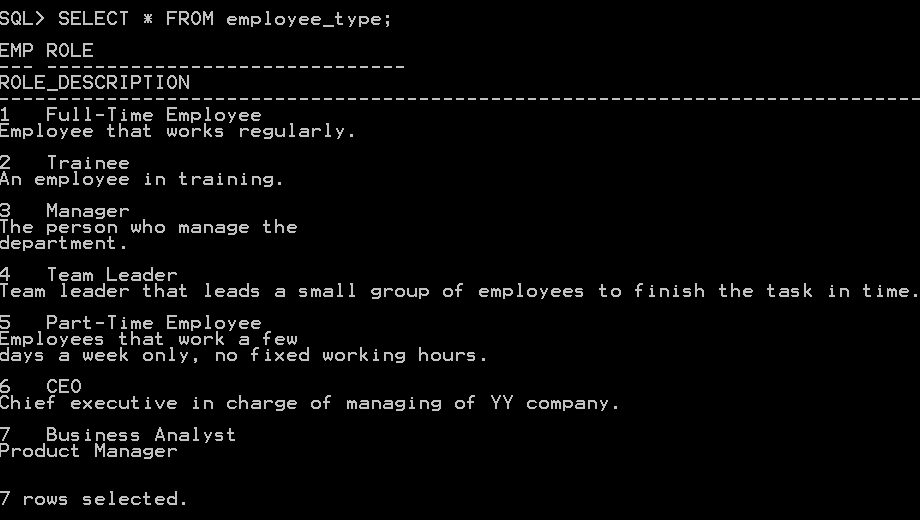
END;

/

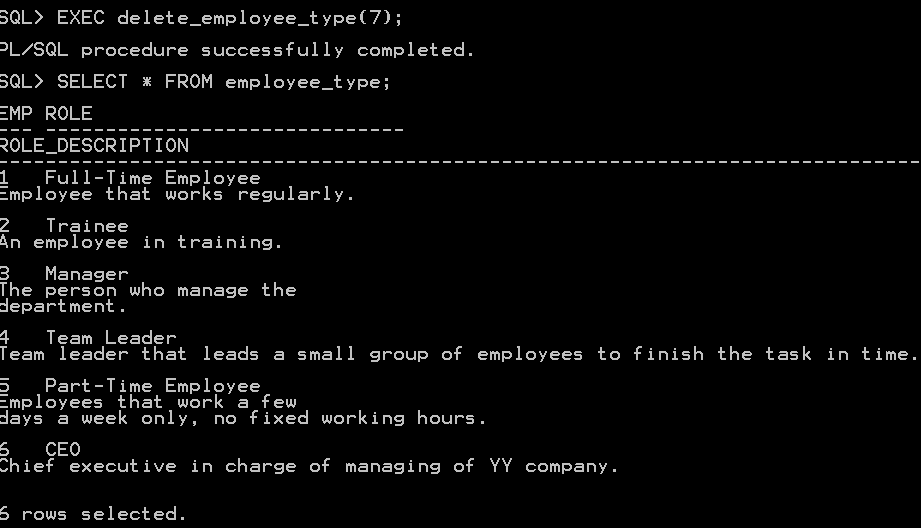


Function: Allow the user to delete a selected employee type details with id.

Before using the delete\_employee\_type procedure.



After using the delete\_employee\_type procedure.



**Procedure 4:**

CREATE OR REPLACE PROCEDURE add\_leave\_type

(

new\_leave\_id in VARCHAR,

new\_leave\_type in VARCHAR,

new\_description in VARCHAR,

new\_leave\_days\_per\_year in INT,

new\_max\_days\_carryforward in INT)

IS BEGIN

INSERT INTO leave\_type VALUES

(

new\_leave\_id,

new\_leave\_type,

new\_description,

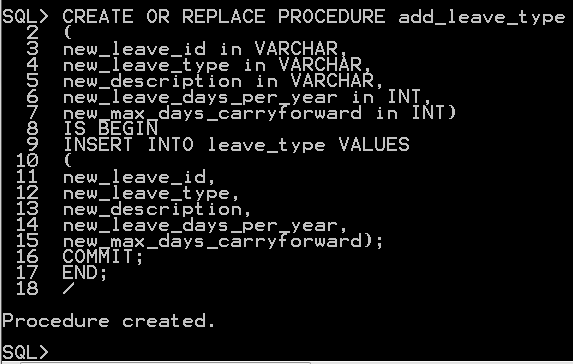
new\_leave\_days\_per\_year,

new\_max\_days\_carryforward);

COMMIT;

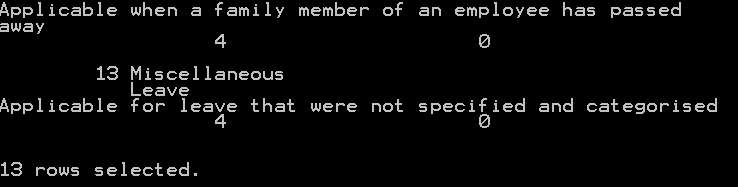
END;

/

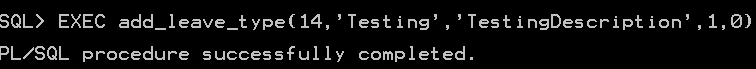


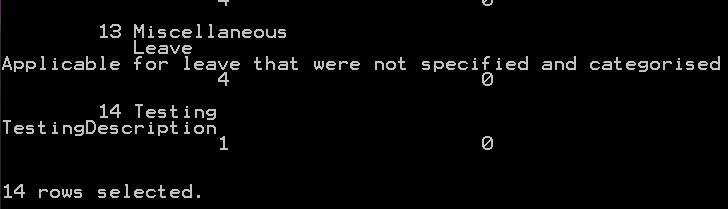
Function: Allow the user to insert new leave type into leave type table using procedure.

Before using the add\_leave\_type procedure.



After using the add\_leave\_type procedure.





**Procedure 5:**

CREATE OR REPLACE PROCEDURE update\_leave\_type

(

selected\_leave\_id in VARCHAR,

new\_leave\_type in VARCHAR,

new\_description in VARCHAR,

new\_leave\_days\_per\_year in INT,

new\_max\_days\_carryforward in INT)

IS BEGIN

UPDATE leave\_type

SET

leave\_type=new\_leave\_type,

description=new\_description,

leave\_days\_per\_year=new\_leave\_days\_per\_year,

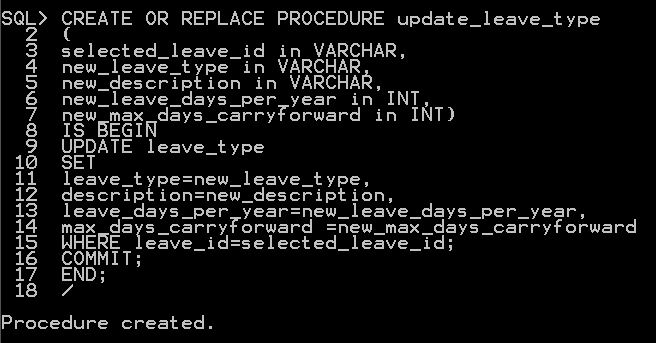
max\_days\_carryforward =new\_max\_days\_carryforward

WHERE leave\_id=selected\_leave\_id;

COMMIT;

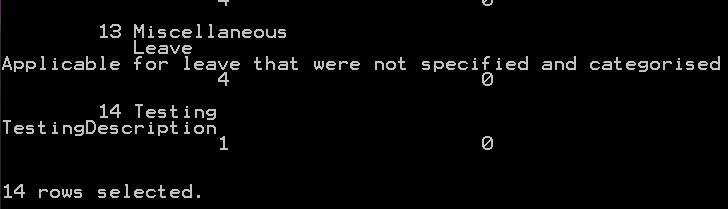
END;

/

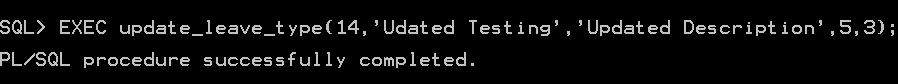


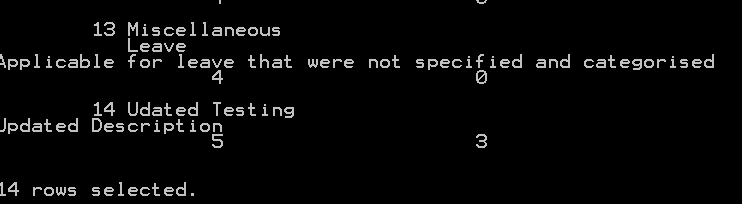
Function: Allow the user to update a selected leave type details with id.

Before using the update\_leave\_type procedure.



After using the update\_leave\_type procedure.





**10.3 FUNCTIONS:**

**Function 1:**

CREATE OR REPLACE FUNCTION leave\_balance\_type(report\_type\_id IN VARCHAR)

RETURN VARCHAR2

IS

type\_count NUMBER;

BEGIN

SELECT COUNT(\*) INTO type\_count

FROM leave\_balance

WHERE leave\_id=report\_type\_id;

RETURN ('There are '||type\_count||' of leave-type-id('||report\_type\_id||') exist in leave balance table. ');

END;

/

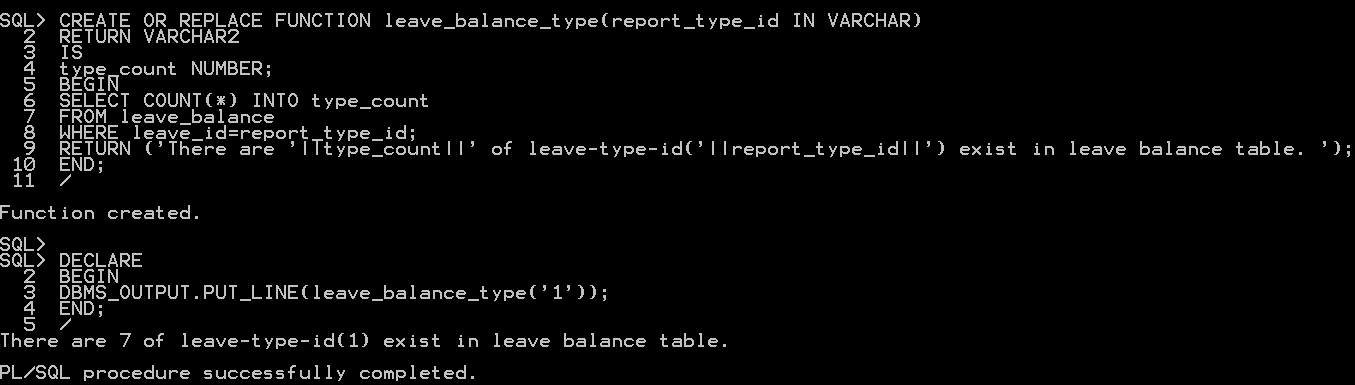
DECLARE

BEGIN

DBMS\_OUTPUT.PUT\_LINE(leave\_balance\_type('1'));

END;

/



Function: Allow the user to search the number of employees by employee type id.

**Function 2:**

CREATE OR REPLACE FUNCTION report\_type\_count(report\_type\_id IN VARCHAR2)

RETURN VARCHAR2

IS

type\_count NUMBER;

BEGIN

SELECT COUNT(\*) INTO type\_count

FROM leave\_report

WHERE leave\_id=report\_type\_id;

RETURN ('There are '||type\_count||' of leave-type-id('||report\_type\_id||') requested in this company. ');

END;

/

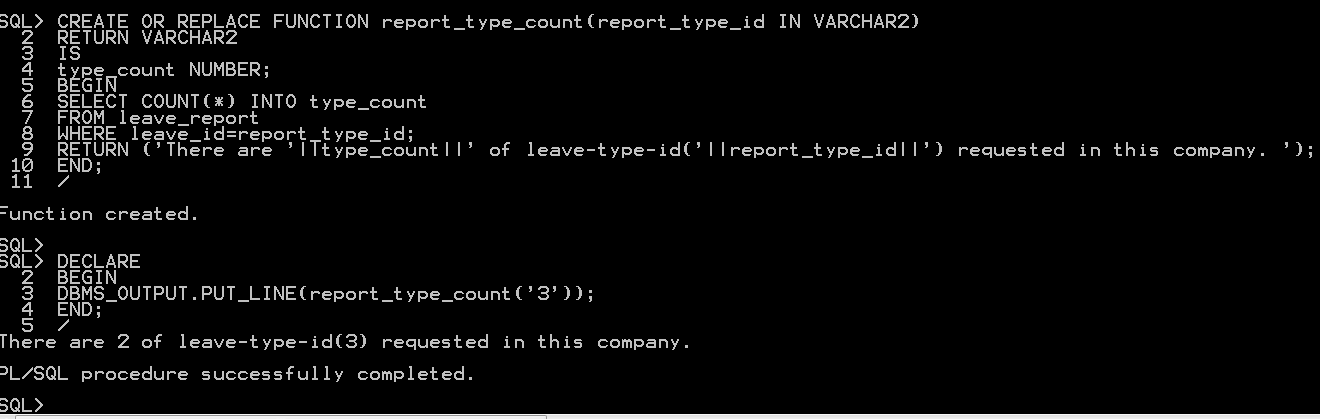
DECLARE

BEGIN

DBMS\_OUTPUT.PUT\_LINE(report\_type\_count('3'));

END;

/



Function: Allow the user to search the times of specific leave\_type requested in the leave report table.

**Function 3:**

CREATE OR REPLACE FUNCTION branch\_salary(input\_branch\_id in VARCHAR2)

RETURN VARCHAR2

IS

total\_salary NUMBER;

BEGIN

SELECT SUM(salary) INTO total\_salary FROM employee

WHERE branch\_id = input\_branch\_id;

RETURN('The monthly paid of this branch is RM'||total\_salary||'. ');

END;

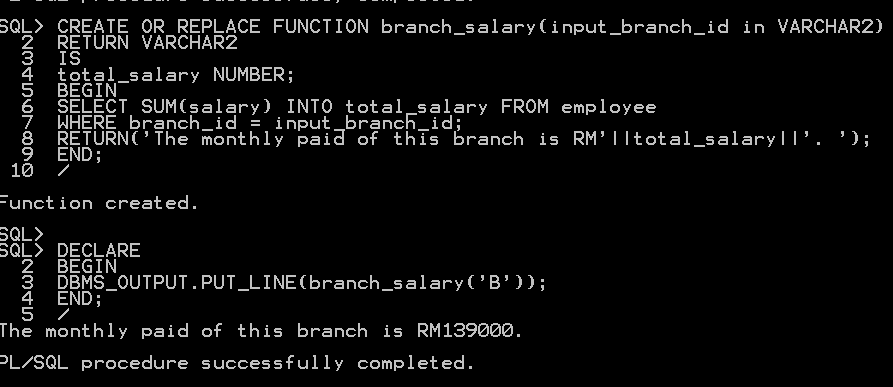
/

DECLARE

BEGIN

DBMS\_OUTPUT.PUT\_LINE(branch\_salary('B'));

END;

/Function: Allow the user to get the total paid to the branch per month.

**Function 4:**

CREATE OR REPLACE FUNCTION employee\_year\_salary(emp\_id in VARCHAR2)

RETURN VARCHAR2

IS

total\_salary NUMBER;

BEGIN

SELECT salary\*12 INTO total\_salary FROM employee

WHERE employee\_id = emp\_id;

RETURN('Employee earn RM'||total\_salary||' per year . ');

END;

/

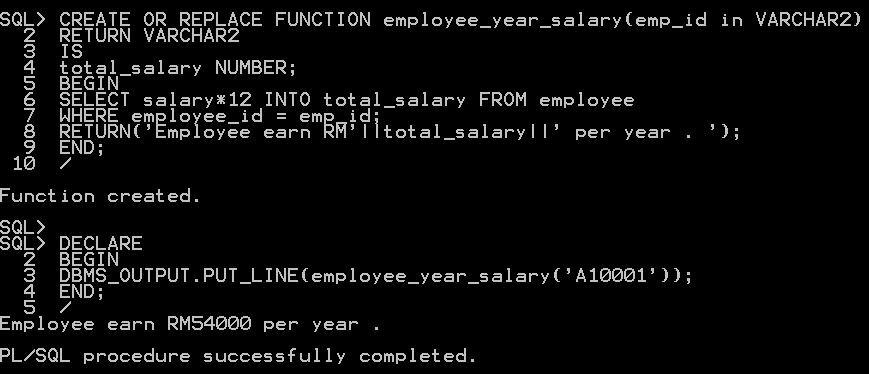
DECLARE

BEGIN

DBMS\_OUTPUT.PUT\_LINE(employee\_year\_salary('A10001'));

END;

/



Function: Allow the user to get the total salary per year of a specified employee.

**Function 5:**

CREATE OR REPLACE FUNCTION emp\_type\_count(emp\_type\_id in VARCHAR2)

RETURN VARCHAR2

IS

emp\_count NUMBER;

BEGIN

SELECT COUNT(\*) INTO emp\_count

FROM employee

WHERE employee\_type\_id=emp\_type\_id;

RETURN('There is '||emp\_count||' of type-id('|| emp\_type\_id ||') employees in this company . ');

END;

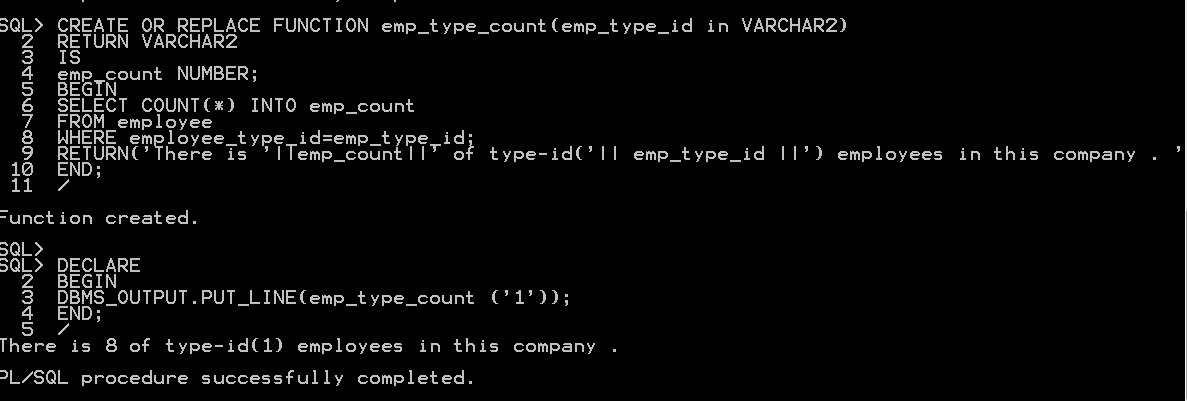
/

DECLARE

BEGIN

DBMS\_OUTPUT.PUT\_LINE(emp\_type\_count ('1'));

END;

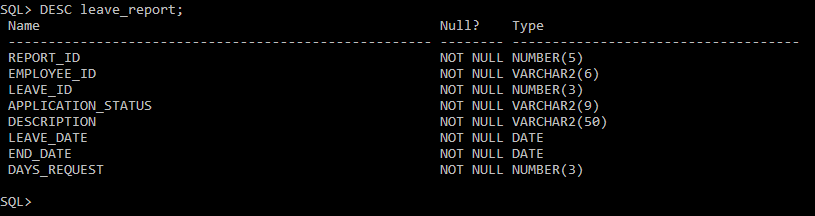
/Function: Allow the user to get the number of employees with specified employee\_type id.

**11.0 Individual Assessment (TAN JIA YONG 17ACB03508)**

**11.1 QUERIES**

**Query 1**

DESC leave\_report;

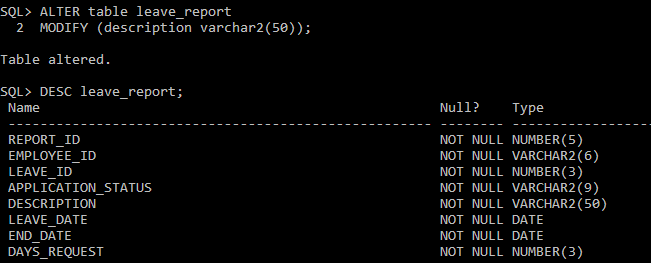


**Function:** View the information of the attribute name and datatype of leave\_report table.

**Query 2**

ALTER table leave\_report

MODIFY (description varchar2(50));

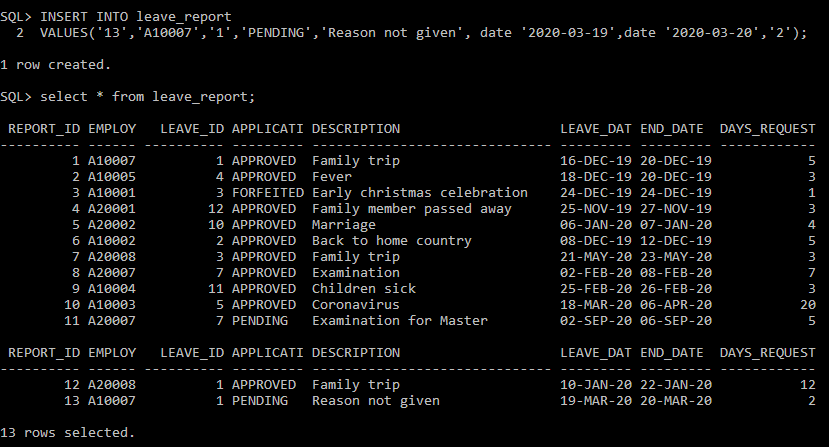


**Function:** The query is able to modify the length of description column so that it can fit the data size.

**Query 3**

INSERT INTO leave\_report

VALUES('13','A10007','1','PENDING','Reason not given', date '2020-03-19',date '2020-03-20','2');



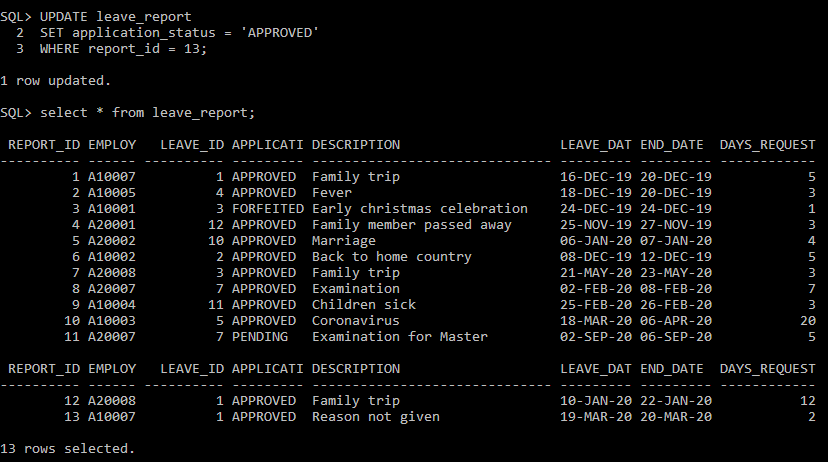
**Function:** Insert new Leave report into the table.

**Query 4**

UPDATE leave\_report

SET application\_status = 'APPROVED'

WHERE report\_id = 13;



**Function:** The query can update the status of application\_status.

**Query 5**

SELECT \* from leave\_report

ORDER BY leave\_date;

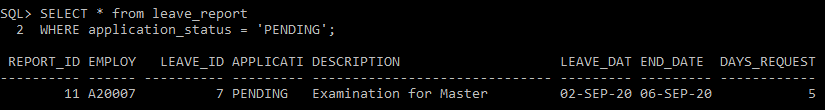


**Function:** The query will sort the records in ascending order based on leave\_date.

**Query 6**

SELECT \* from leave\_report

WHERE application\_status = 'PENDING';



**Function:** The query will display the record with PENDING status on application\_status.

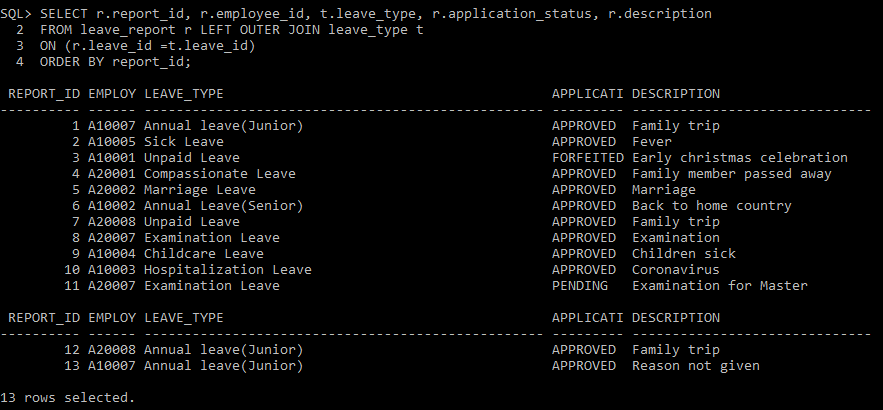
**Query 7**

SELECT r.report\_id, r.employee\_id, t.leave\_type, r.application\_status, r.description

FROM leave\_report r LEFT OUTER JOIN leave\_type t

ON (r.leave\_id =t.leave\_id)

ORDER BY report\_id;



**Function:** The query will replace the leave\_id with leave\_type so that we know exactly which leave is requested.

**Query 8**

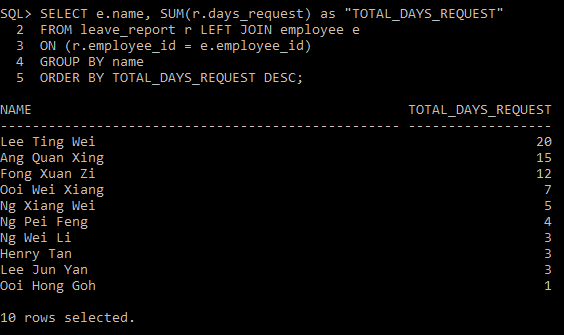
SELECT e.name, SUM(r.days\_request) as "TOTAL\_DAYS\_REQUEST"

FROM leave\_report r LEFT JOIN employee e

ON (r.employee\_id = e.employee\_id)

GROUP BY name

ORDER BY TOTAL\_DAYS\_REQUEST DESC;



**Function:** The query will show the total day requested by each employee based on leave\_report table.

**Query 9**

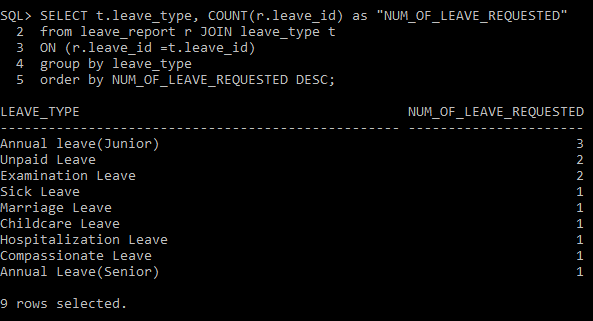
SELECT t.leave\_type, COUNT(r.leave\_id) as "NUM\_OF\_LEAVE\_REQUESTED"

FROM leave\_report r JOIN leave\_type t

ON (r.leave\_id =t.leave\_id)

GROUP BY leave\_type

ORDER BY NUM\_OF\_LEAVE\_REQUESTED DESC;



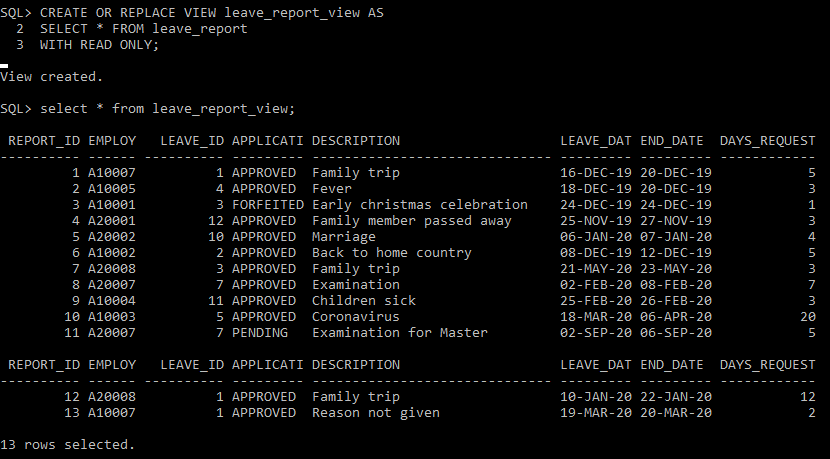
**Function:** The query will show the total number of leave requested and leave\_type based on leave\_report.

**Query 10**

CREATE OR REPLACE VIEW leave\_report\_view AS

SELECT \* FROM leave\_report

WITH READ ONLY;



**Function:** The query will create a view for user to view the table and are not able to edit the record.

**11.2 STORED PROCEDURES**

**Procedure 1:**

CREATE OR REPLACE PROCEDURE add\_report

(new\_report\_id in number, new\_employee\_id in varchar2, new\_leave\_id in number,

new\_application\_status in varchar2, new\_description in varchar2,

new\_leave\_date in date, new\_end\_date in date, new\_days\_request in number)

IS BEGIN

INSERT INTO leave\_report

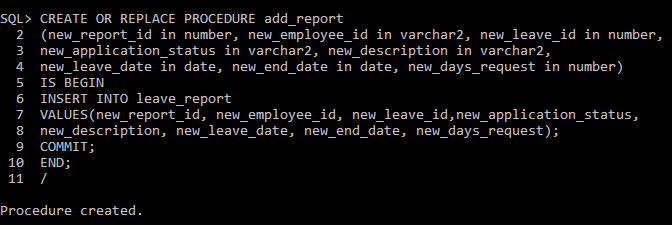
VALUES(new\_report\_id, new\_employee\_id, new\_leave\_id,new\_application\_status,

new\_description, new\_leave\_date, new\_end\_date, new\_days\_request);

COMMIT;

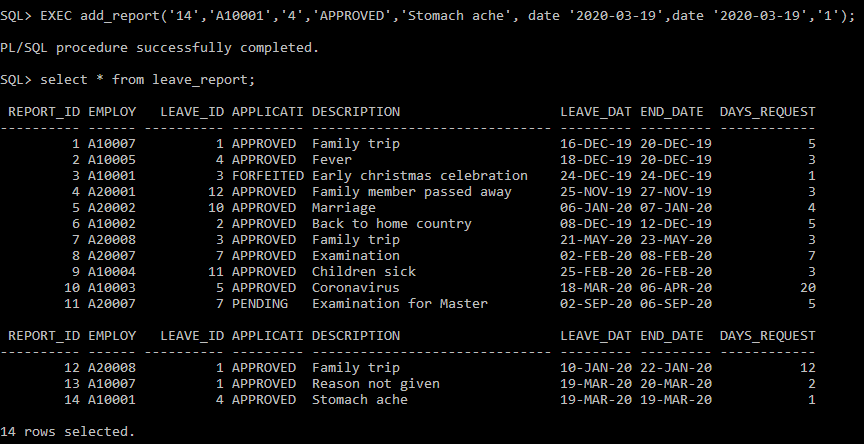
END;

/



**Function:** Allow users to add new leave report into leave\_report using procedure.

After executing the procedure. The record is added into leave\_report.



**Procedure 2:**

CREATE OR REPLACE PROCEDURE update\_application\_status

(selected\_report\_id in number, new\_application\_status in varchar2)

IS BEGIN

UPDATE leave\_report

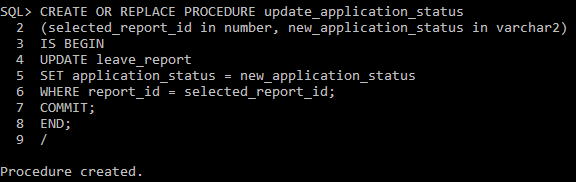
SET application\_status = new\_application\_status

WHERE report\_id = selected\_report\_id;

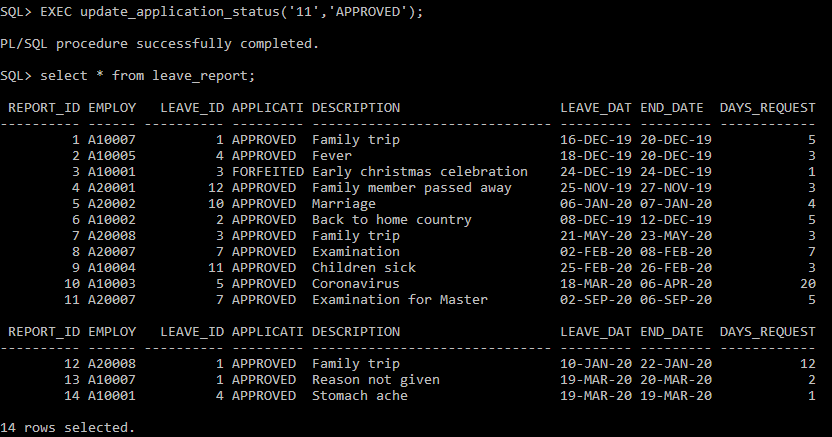
COMMIT;

END;

/



**Function:** Allow users to update the status of application\_status using procedure.

After executing the procedure. The status of report\_id ‘14’ is updated.

**Procedure 3:**

CREATE OR REPLACE PROCEDURE delete\_report

(selected\_report\_id in number)

IS BEGIN

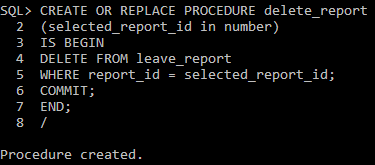
DELETE FROM leave\_report

WHERE report\_id = selected\_report\_id;

COMMIT;

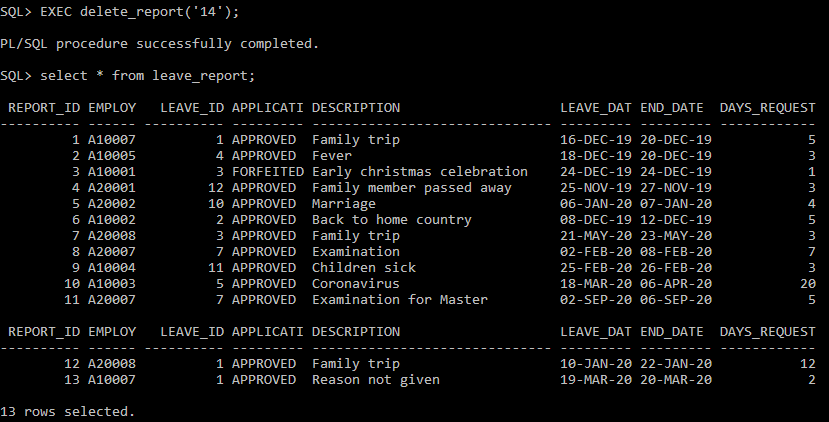
END;

/



**Function:** Allow users to delete the leave report using procedure.

After executing the procedure, the leave report\_id ‘14’ is deleted from the table.



**Procedure 4:**

CREATE OR REPLACE PROCEDURE update\_leave\_date

(selected\_report\_id in number, new\_leave\_date in date,

new\_end\_date in date, new\_days\_request in number)

IS BEGIN

UPDATE leave\_report

SET leave\_date = new\_leave\_date,

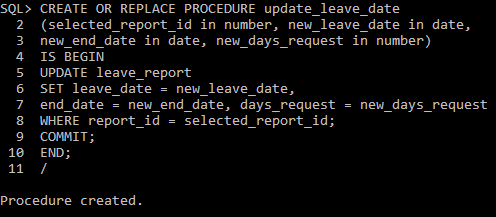
end\_date = new\_end\_date, days\_request = new\_days\_request

WHERE report\_id = selected\_report\_id;

COMMIT;

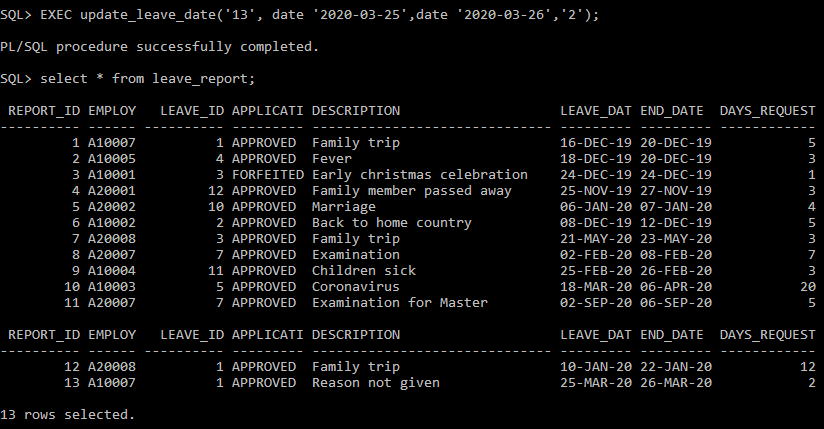
END;

/



**Function:** Allow users to update the leave\_date,end\_date and days\_request using procedure.

After executing the procedure, the leave\_date of report\_id ‘13’ is updated.



**Procedure 5:**

CREATE OR REPLACE PROCEDURE update\_description

(selected\_report\_id in number, new\_description in varchar2)

IS BEGIN

UPDATE leave\_report

SET description = new\_description

WHERE report\_id = selected\_report\_id;

COMMIT;

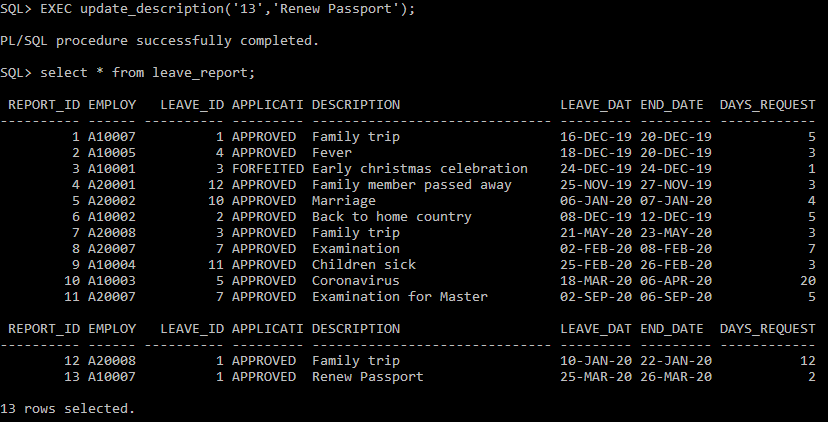
END;

/



**Function:** Allow users to update the description of leave\_report using procedure.

After executing the procedure, the description of report\_id ‘13’ is updated.



**11.3 FUNCTIONS**

**Function 1:**

CREATE OR REPLACE FUNCTION average\_day\_request(input\_leave\_id in VARCHAR2)

RETURN VARCHAR2

IS

average NUMBER;

BEGIN

SELECT ROUND(AVG(days\_request)) INTO average FROM leave\_report

WHERE leave\_id = input\_leave\_id;

RETURN('The average day requested of this leave\_id is '||average||' Day. ');

END;

/

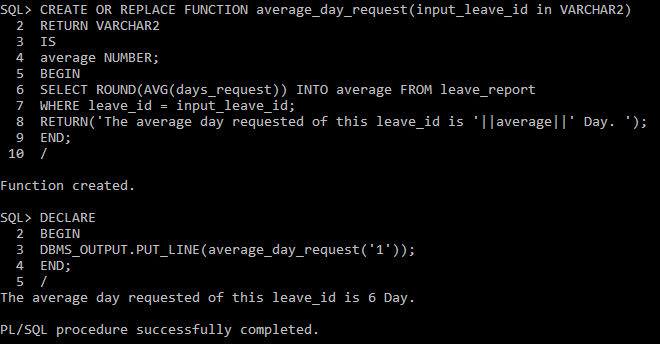
DECLARE

BEGIN

DBMS\_OUTPUT.PUT\_LINE(average\_day\_request('1'));

END;

/



**Function:** The user can find the average day request of a type of leave by executing this function and inputting the leave\_id.

**Function 2:**

CREATE OR REPLACE FUNCTION total\_day\_request(input\_employee\_id in VARCHAR2)

RETURN VARCHAR2

IS

total NUMBER;

BEGIN

SELECT SUM(days\_request) INTO total FROM leave\_report

WHERE employee\_id = input\_employee\_id;

RETURN('The total day requested by '||input\_employee\_id||' is '||total||' Day. ');

END;

/

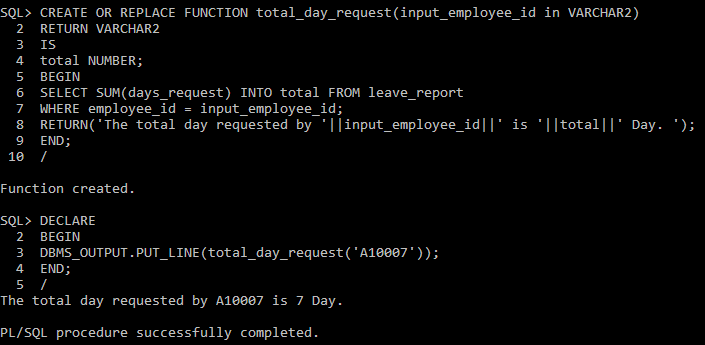
DECLARE

BEGIN

DBMS\_OUTPUT.PUT\_LINE(total\_day\_request('A10007'));

END;

/



**Function:** The user can find the total day requested by a certain employee by executing this function and inputting the employee\_id.

**Function 3:**

CREATE OR REPLACE FUNCTION check\_days

RETURN VARCHAR2

IS

A NUMBER;

B NUMBER;

BEGIN

SELECT SUM(days\_request) INTO A FROM leave\_report;

SELECT SUM(end\_date-leave\_date+1) INTO B FROM leave\_report;

IF B=A THEN

RETURN('The Date and Day Requested is correct.');

ELSE

RETURN('The Date and Day Requested is not correct.');

END IF;

END;

/

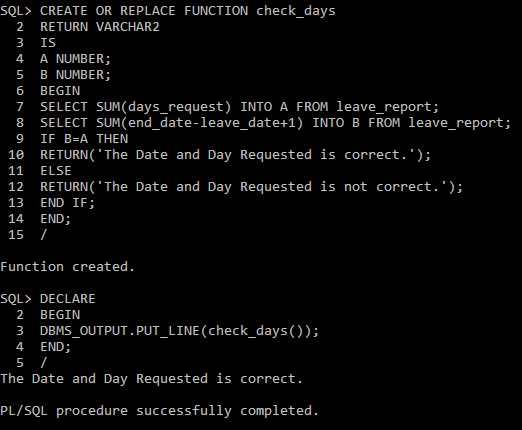
DECLARE

BEGIN

DBMS\_OUTPUT.PUT\_LINE(check\_days());

END;

/



**Function:** The user can check whether the days\_request and the date is tally by executing this function.

**Function 4:**

CREATE OR REPLACE FUNCTION total\_report\_emp(input\_employee\_id in VARCHAR2)

RETURN VARCHAR2

IS

total NUMBER;

BEGIN

SELECT COUNT(report\_id) INTO total FROM leave\_report

WHERE employee\_id = input\_employee\_id;

RETURN('The total number of leave report by '||input\_employee\_id||' is '||total||' .');

END;

/

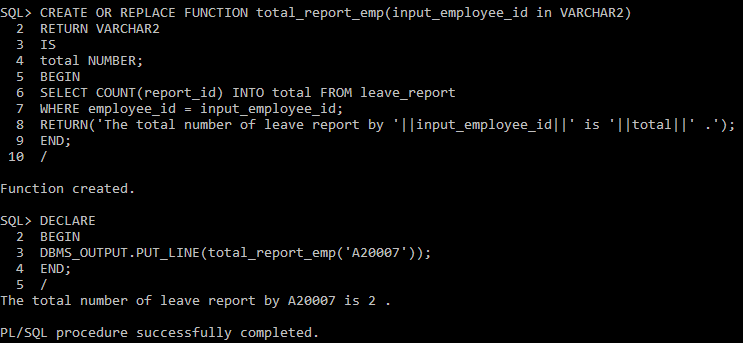
DECLARE

BEGIN

DBMS\_OUTPUT.PUT\_LINE(total\_report\_emp('A20007'));

END;

/



**Function:** The user can find the total number of leave reports made by a certain employee by executing this function.

**Function 5:**

CREATE OR REPLACE FUNCTION check\_total\_status

RETURN VARCHAR2

IS

A NUMBER;

F NUMBER;

P NUMBER;

BEGIN

SELECT COUNT(application\_status) INTO A FROM leave\_report

WHERE application\_status = 'APPROVED';

SELECT COUNT(application\_status) INTO P FROM leave\_report

WHERE application\_status = 'PENDING';

SELECT COUNT(application\_status) INTO F FROM leave\_report

WHERE application\_status = 'FORFEITED';

RETURN('Total APPROVED : '||A||'

Total PENDING : '||P||'

Total FORFEITED : '||F||' ');

END;

/

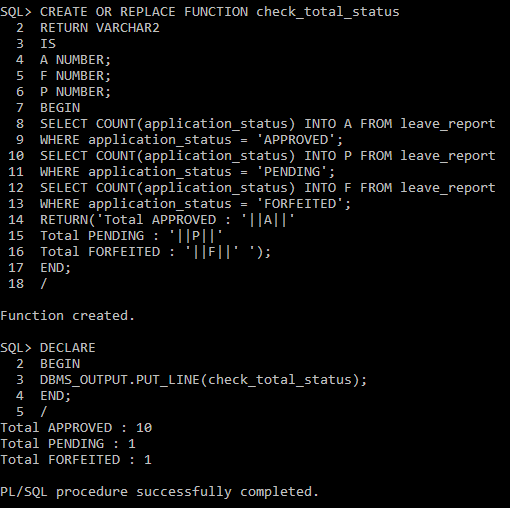
DECLARE

BEGIN

DBMS\_OUTPUT.PUT\_LINE(check\_total\_status);

END;

/



**Function:** The user can check the total number of application status by executing this function.