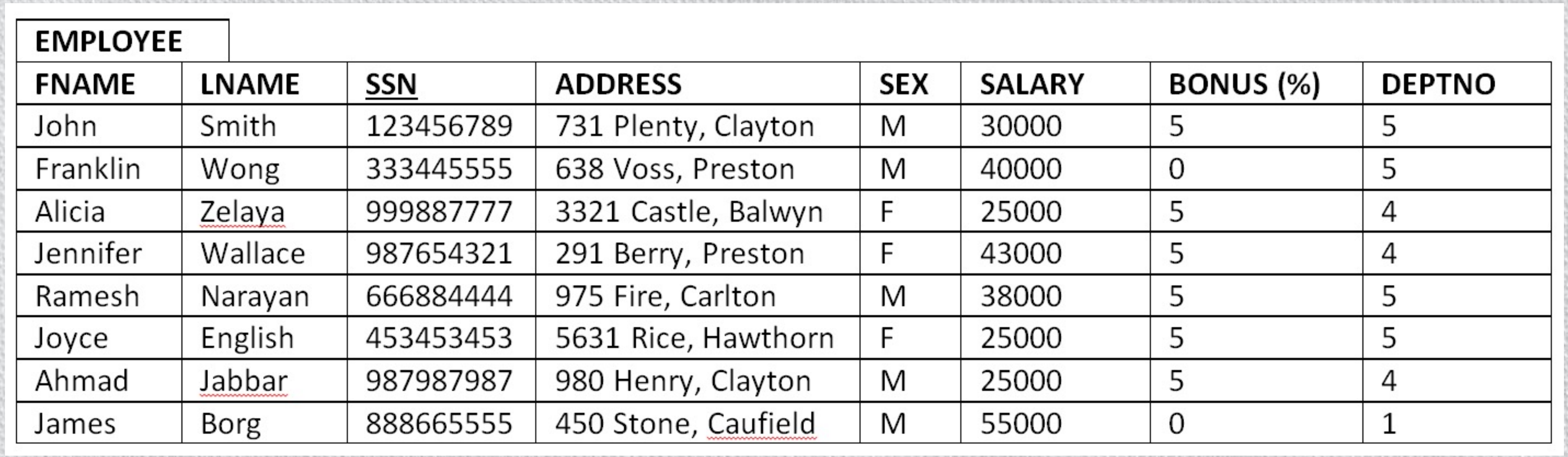
CSE2DBF – CSE4DBF

Stored Procedures

and Stored Functions Exercise

Stored Procedure Exercise 1



EMPLOYEE

FNAME	LNAME	SSN	ADDRESS	SEX	SALARY	BONUS (%)	DEPTNO
John	Smith	123456789	731 Plenty, Clayton	M	30000	5	5
Franklin	Wong	333445555	638 Voss, Preston	M	40000	0	5
Alicia	Zelaya	999887777	3321 Castle, Balwyn	F	25000	5	4
Jennifer	Wallace	987654321	291 Berry, Preston	F	43000	5	4
Ramesh	Narayan	666884444	975 Fire, Carlton	M	38000	5	5
Joyce	English	453453453	5631 Rice, Hawthorn	F	25000	5	5
Ahmad	Jabbar	987987987	980 Henry, Clayton	M	25000	5	4
James	Borg	888665555	450 Stone, Caufield	M	55000	0	1

Based on the above EMPLOYEE table, write a stored procedure which displays employee's total salary. The procedure takes the employee's ***ssn*** as input, and displays the employee's ***full name*** and total salary (***salary + bonus***) to the screen.

[15 marks]

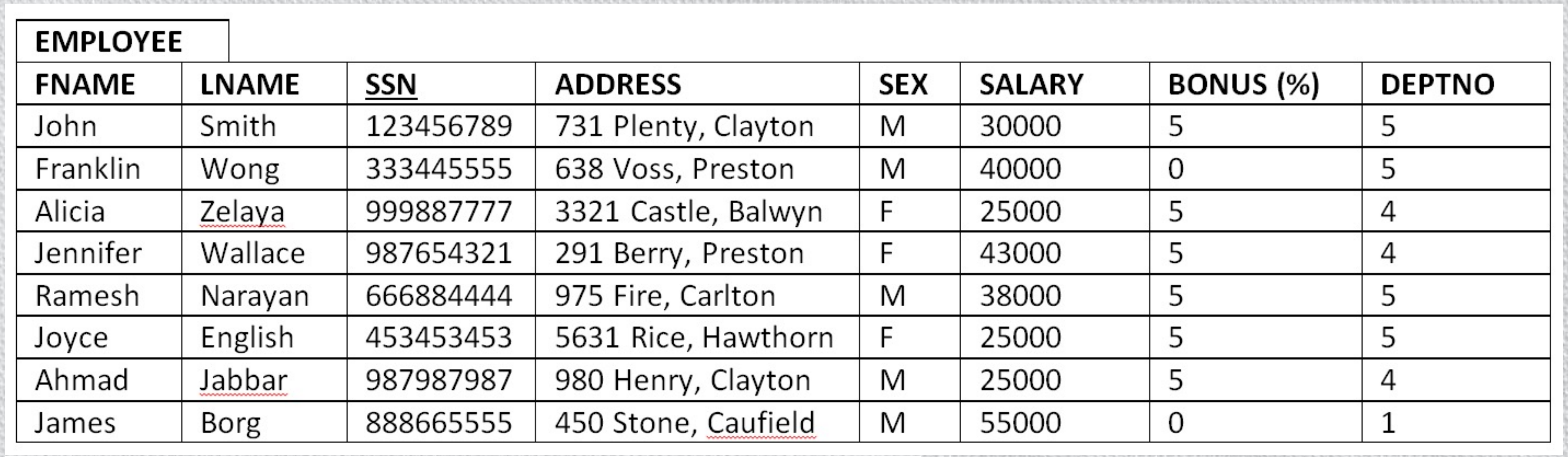
Stored Procedure Exercise 1

```
CREATE OR REPLACE PROCEDURE TotalSalary
(P_SSN EMPLOYEE.SSN%type) AS

V_FName EMPLOYEE.FName%Type;
V_LName EMPLOYEE.LName%Type;
V_TotalSalary EMPLOYEE.Salary%Type;

BEGIN

  SELECT FName, LName, Salary * ((100+Bonus)/100)
  INTO V_FName, V_LName, V_TotalSalary
  FROM EMPLOYEE
  WHERE SSN = P_SSN;

  DBMS_OUTPUT.PUT_LINE
        (v_FName || ' ' || v_LName || ' ' || V_TotalSalary);

END TotalSalary;
/
```

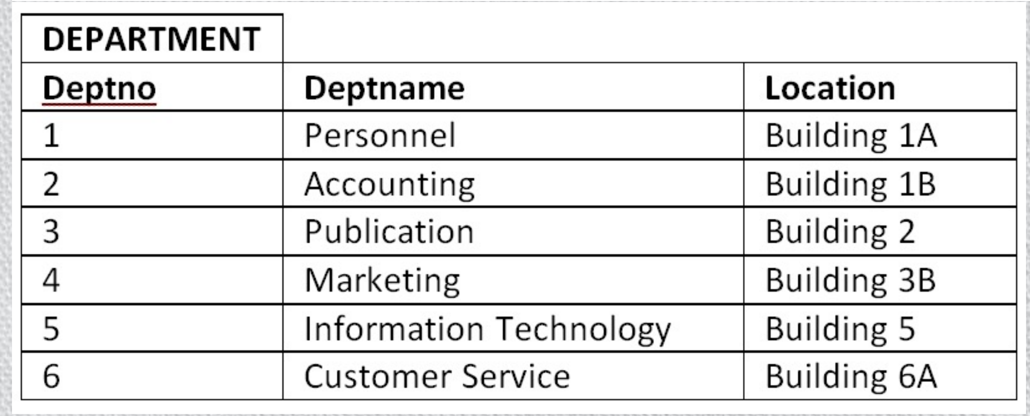
Stored Procedure Exercise 1

```
SQL> EXECUTE TotalSalary(‘123456789’);

John Smith 31500

PL/SQL procedure successfully completed.


SQL> EXECUTE TotalSalary(‘333445555’);

Franklin Wong 40000

PL/SQL procedure successfully completed.
```

Stored Function Exercise 1



EMPLOYEE							
FNAME	LNAME	SSN	ADDRESS	SEX	SALARY	BONUS (%)	DEPTNO
John	Smith	123456789	731 Plenty, Clayton	M	30000	5	5
Franklin	Wong	333445555	638 Voss, Preston	M	40000	0	5
Alicia	Zelaya	999887777	3321 Castle, Balwyn	F	25000	5	4
Jennifer	Wallace	987654321	291 Berry, Preston	F	43000	5	4
Ramesh	Narayan	666884444	975 Fire, Carlton	M	38000	5	5
Joyce	English	453453453	5631 Rice, Hawthorn	F	25000	5	5
Ahmad	Jabbar	987987987	980 Henry, Clayton	M	25000	5	4
James	Borg	888665555	450 Stone, Caufield	M	55000	0	1



DEPARTMENT		
Deptno	Deptname	Location
1	Personnel	Building 1A
2	Accounting	Building 1B
3	Publication	Building 2
4	Marketing	Building 3B
5	Information Technology	Building 5
6	Customer Service	Building 6A

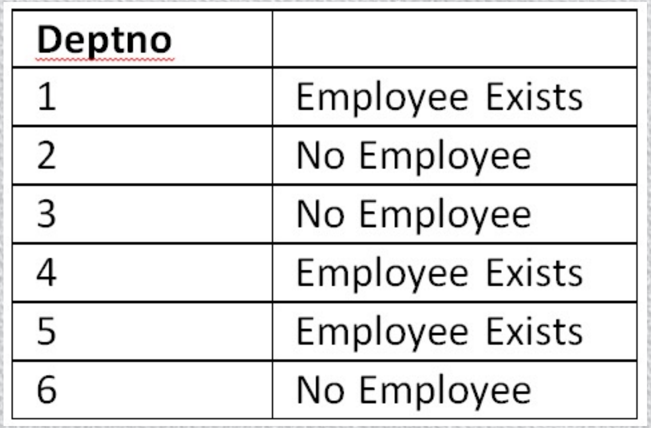
Write a stored function which takes ***Deptno*** as a parameter input. If the EMPLOYEE table does not contain that *Deptno*, return “***No Employee***”, otherwise return a “***Employee Exists***” value. [10 marks]

Stored Function Exercise 1

```
CREATE OR REPLACE FUNCTION EmpInfo
(P_Department EMPLOYEE.DeptNo%TYPE)
RETURN VARCHAR2 IS

  V_EmpCount NUMBER;

BEGIN

  SELECT Count(*)
  INTO V_EmpCount
  FROM EMPLOYEE
  WHERE DeptNo = P_Department;

  IF V_EmpCount > 0 THEN
    RETURN 'Employee Exists';
  ELSE
    RETURN 'No Employee';
  END IF;

END EmpInfo;
/
```

Stored Function Exercise 1

Write an appropriate SQL statement on the previous tables which uses stored function ***EmpInfo*** to display the following information



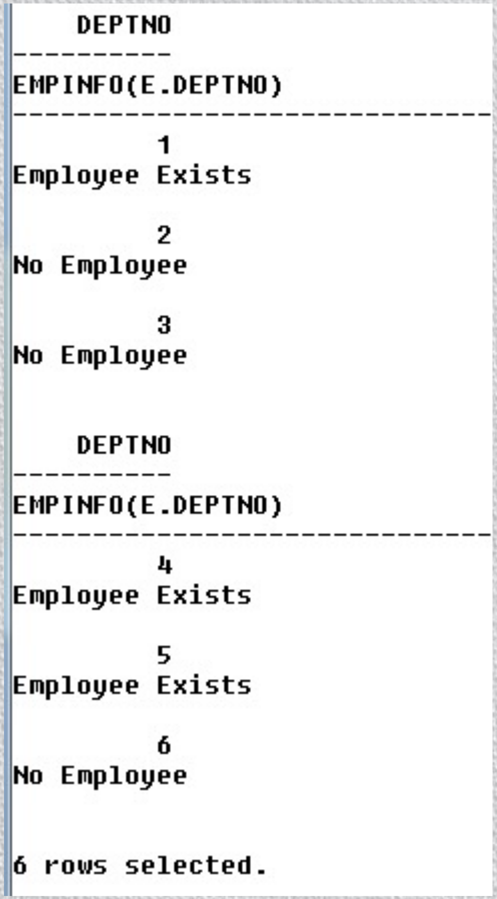
Deptno	
1	Employee Exists
2	No Employee
3	No Employee
4	Employee Exists
5	Employee Exists
6	No Employee

Note:

Whenever the *Deptno* from DEPARTMENT table exists within the EMPLOYEE table, it will have “Employee Exists” value associated with it. When the *Deptno* does not exist within the EMPLOYEE table, it will have a “No Employee” value associated with it. [5 marks]

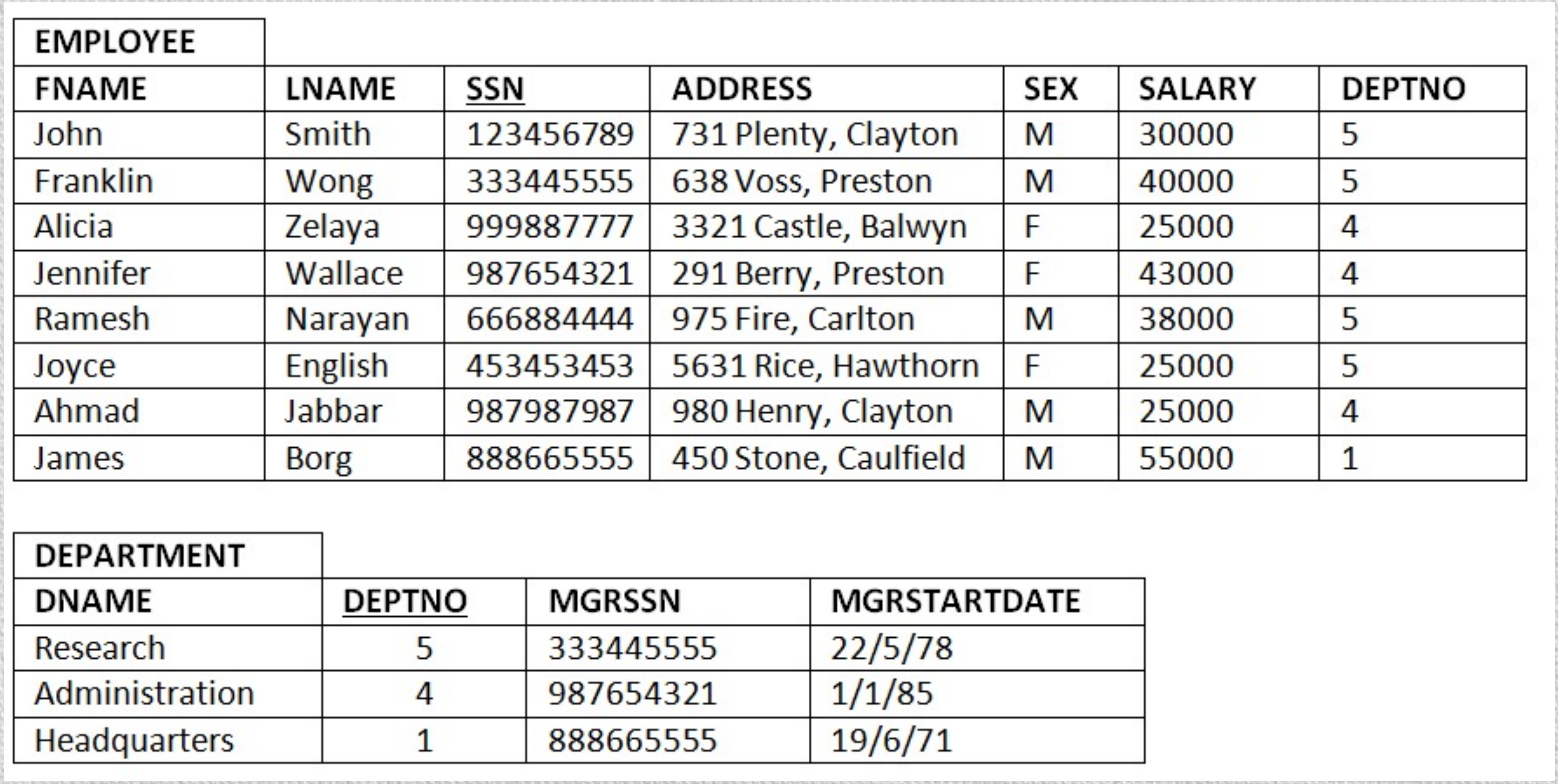
Stored Function Exercise 1

```
SELECT DeptNo, EmpInfo(DeptNo)
FROM Department
ORDER BY DeptNo;
```



```
    DEPTNO
----------
EMPINFO(E.DEPTNO)
------------------------------
         1
Employee Exists

         2
No Employee

         3
No Employee


    DEPTNO
----------
EMPINFO(E.DEPTNO)
------------------------------
         4
Employee Exists

         5
Employee Exists

         6
No Employee


6 rows selected.
```

Stored Procedure Exercise 2



EMPLOYEE						
FNAME	LNAME	SSN	ADDRESS	SEX	SALARY	DEPTNO
John	Smith	123456789	731 Plenty, Clayton	M	30000	5
Franklin	Wong	333445555	638 Voss, Preston	M	40000	5
Alicia	Zelaya	999887777	3321 Castle, Balwyn	F	25000	4
Jennifer	Wallace	987654321	291 Berry, Preston	F	43000	4
Ramesh	Narayan	666884444	975 Fire, Carlton	M	38000	5
Joyce	English	453453453	5631 Rice, Hawthorn	F	25000	5
Ahmad	Jabbar	987987987	980 Henry, Clayton	M	25000	4
James	Borg	888665555	450 Stone, Caulfield	M	55000	1

DEPARTMENT			
DNAME	DEPTNO	MGRSSN	MGRSTARTDATE
Research	5	333445555	22/5/78
Administration	4	987654321	1/1/85
Headquarters	1	888665555	19/6/71

Based on the above EMPLOYEE table, write a stored procedure which performs an update of the salary column. The procedure takes the **percentage of salary increment** as input, and changes all salary values within the table by adding the increment. The procedure will display to the screen employee names and new salaries as output.

[15 marks]

Stored Procedure Exercise 2

```
CREATE OR REPLACE PROCEDURE UpdateSalary
(Increment NUMBER) AS

CURSOR C_UpdateSalary IS
  SELECT Fname, (Salary * ((100+Increment)/100)) AS NewSalary
  FROM EMPLOYEE;

BEGIN

  FOR V_UpdateSalary IN C_UpdateSalary LOOP
    dbms_output.put_line
         (v_UpdateSalary.FName||' '|| v_UpdateSalary.NewSalary);
   END LOOP;

  UPDATE EMPLOYEE
  SET SALARY = Salary * ((100 + Increment)/100);

END UpdateSalary;
/
```