Chapter 4
Structure Query Language

SQL - JOINS

Department: Computer
Course: DBMS
Faculty: Sana Shaikh

Types of Joins

Inner Join

- Equi Join
- Self Join
- Natural Join

Outer Joins

- Left Outer Join
- Right Outer Join
- Full Outer Join

Cross Join

Displaying data from Multiple Tables

- Write `SELECT` statements to access data from more than one table using equijoins and nonequijoins
- Join a table to itself by using a self-join
- View data that generally does not meet a join condition by using outer joins
- Generate a Cartesian product of all rows from two or more tables

Obtaining Data from Multiple Tables

EMPLOYEES

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	200	Whalen	10
2	201	Hartstein	20
3	202	Fay	20
4	205	Higgins	110
...			
18	174	Abel	80
19	176	Taylor	80
20	178	Grant	(null)

DEPARTMENTS

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
1	10	Administration	1700
2	20	Marketing	1800
3	50	Shipping	1500
4	60	IT	1400
5	80	Sales	2500
6	90	Executive	1700
7	110	Accounting	1700
8	190	Contracting	1700

	EMPLOYEE_ID	DEPARTMENT_ID	DEPARTMENT_NAME
1	100	90	Executive
2	101	90	Executive
...			
17	202	20	Marketing
18	205	110	Accounting
19	206	110	Accounting

Creating Natural Joins

- The `NATURAL JOIN` clause is based on all columns in the two tables that have the same name.
- It selects rows from the two tables that have equal values in all matched columns.
- If the columns having the same names have different data types, an error is returned.

Retrieving Records with Natural Joins

```
SELECT department_id, department_name,
       location_id, city
FROM   departments
NATURAL JOIN locations ;
```

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID	CITY
1	60	IT	1400	Southlake
2	50	Shipping	1500	South San Francisco
3	10	Administration	1700	Seattle
4	90	Executive	1700	Seattle
5	110	Accounting	1700	Seattle
6	190	Contracting	1700	Seattle
7	20	Marketing	1800	Toronto
8	80	Sales	2500	Oxford

SQL> select * from dept;

DEPTNO	DNAME	LOC
22	IT	Mum
11	com	mumbai
12		
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

7 rows selected.

SQL> select * from emp1;

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	20
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7500	SANA	Prof	7499	18-JUN-08	1000	100	11

```
SQL> select empno, ename, deptno, dname
  2  from emp1
  3  natural join dept;
```

```
     EMPNO ENAME                DEPTNO DNAME
---------- ---------- ---------- --------------
      7500 SANA                   11 com
      7499 ALLEN                  20 RESEARCH
      7844 TURNER                 30 SALES
      7521 WARD                   30 SALES
      7654 MARTIN                 30 SALES
```

Joining Column Names

EMPLOYEES

	EMPLOYEE_ID	DEPARTMENT_ID
1	200	10
2	201	20
3	202	20
4	205	110
5	206	110
6	100	90
7	101	90
8	102	90
9	103	60
10	104	60
11	107	60
12	124	50
13	141	50
14	142	50
15	143	50
16	144	50
17	149	80
18	174	80
19	176	80
20	178	(null)

DEPARTMENTS

	DEPARTMENT_ID	DEPARTMENT_NAME
1	10	Administration
2	20	Marketing
3	20	Marketing
4	50	Shipping
5	50	Shipping
6	50	Shipping
7	50	Shipping
8	50	Shipping
9	60	IT
10	60	IT
11	60	IT
12	80	Sales
13	80	Sales
14	80	Sales
15	90	Executive
16	90	Executive
17	90	Executive
18	110	Accounting
19	110	Accounting

Foreign key **Primary key**

Creating Joins with the ON Clause

- The join condition for the natural join is basically an equijoin of all columns with the same name.
- Use the ON clause to specify arbitrary conditions or specify columns to join.
- The join condition is separated from other search conditions.
- The ON clause makes code easy to understand.

Retrieving Records with the ON Clause

```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_id, d.location_id
FROM   employees e JOIN departments d
ON     (e.department_id = d.department_id);
```

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID_1	LOCATION_ID
1	200	Whalen	10	10	1700
2	201	Hartstein	20	20	1800
3	202	Fay	20	20	1800
4	205	Higgins	110	110	1700
5	206	Gietz	110	110	1700
6	100	King	90	90	1700
7	101	Kochhar	90	90	1700
8	102	De Haan	90	90	1700
9	103	Hunold	60	60	1400
10	104	Ernst	60	60	1400

...

```
SQL> select * from dept;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        22 IT             Mum
        11 com            mumbai
        12
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON

7 rows selected.
```

```
SQL> select * from empl;

     EMPNO ENAME      JOB              MGR HIREDATE       SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7499 ALLEN      SALESMAN         7698 20-FEB-81       1600        300         20
      7521 WARD       SALESMAN         7698 22-FEB-81       1250        500         30
      7654 MARTIN     SALESMAN         7698 28-SEP-81       1250       1400         30
      7844 TURNER     SALESMAN         7698 08-SEP-81       1500          0         30
      7500 SANA       Prof             7499 18-JUN-08       1000        100         11
```

```
SQL> select e.empno, e.ename, e.deptno, d.dname
  2  from emp1 e join dept d
  3  on (e.deptno=d.deptno);

     EMPNO ENAME          DEPTNO DNAME
---------- ---------- ---------- --------------
      7500 SANA               11 com
      7499 ALLEN              20 RESEARCH
      7844 TURNER             30 SALES
      7521 WARD               30 SALES
      7654 MARTIN             30 SALES
```

Self-Joins Using the ON Clause

EMPLOYEES (WORKER)

	EMPLOYEE_ID	LAST_NAME	MANAGER_ID
1	100	King	(null)
2	101	Kochhar	100
3	102	De Haan	100
4	103	Hunold	102
5	104	Ernst	103
6	107	Lorentz	103
7	124	Mourgos	100
8	141	Rajs	124
9	142	Davies	124
10	143	Matos	124

...

EMPLOYEES (MANAGER)

	EMPLOYEE_ID	LAST_NAME
1	100	King
2	101	Kochhar
3	102	De Haan
4	103	Hunold
5	104	Ernst
6	107	Lorentz
7	124	Mourgos
8	141	Rajs
9	142	Davies
10	143	Matos

...

MANAGER_ID in the WORKER table is equal to EMPLOYEE_ID in the MANAGER table.

	EMP	MGR
1	Abel	Zlotkey
2	Davies	Mourgos
3	De Haan	King
4	Ernst	Hunold
5	Fay	Hartstein
6	Gietz	Higgins
7	Grant	Zlotkey
8	Hartstein	King

...

```
SELECT e.last_name emp, m.last_name mgr
FROM   employees e JOIN employees m
ON     (e.manager_id = m.employee_id);
```

```
SQL> create table worker_N as select empno,ename,mgr from emp1;

Table created.

SQL> select empno,ename,mgr from worker_N;

EMPNO      ENAME       MGR
---------- ---------- ----------
    7499       ALLEN       7698
    7521       WARD        7698
    7654       MARTIN      7698
    7844       TURNER      7698
    7500       SANA        7499
```

```
SQL>  create table manager_N as select empno, ename from emp1;

Table created.

SQL> select empno,ename from manager_N;

     EMPNO ENAME
---------- ----------
      7499 ALLEN
      7521 WARD
      7654 MARTIN
      7844 TURNER
      7500 SANA
```

Only for understanding the purpose that how output will look like , created two different tables.

```
SQL>  select w.ename emp_name,  m.ename manager_name
  2   from worker_N w join manager_N m
  3   on (w.mgr=m.empno);

EMP_NAME   MANAGER_NA
---------- ----------
SANA       ALLEN
```

Self-Joins Using the ON Clause

```
SQL> select * from empl;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	20
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7500	SANA	Prof	7499	18-JUN-08	1000	100	11

```
SQL> Select w.ename emp_name, m.ename manager_name
  2  From emp1 w join emp1 m
  3  On (w.mgr=m.empno);
```

EMP_NAME	MANAGER_NA
SANA	ALLEN

Applying Additional Conditions to a Join

```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_id, d.location_id
FROM   employees e JOIN departments d
ON     (e.department_id = d.department_id)
AND    e.manager_id = 149 ;
```

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID_1	LOCATION_ID
1	174	Abel	80	80	2500
2	176	Taylor	80	80	2500

Applying Additional Conditions to a Join

Display SANA is working in which department and name of the department.

Applying Additional Conditions to a Join

Display SANA is working in which department and name of the department.

```
SQL> select e.empno, e.ename, e.deptno, d.dname
  2  from emp1 e join dept d
  3  on (e.deptno=d.deptno)
  4  where ename='SANA';

     EMPNO ENAME          DEPTNO DNAME
---------- ---------- ---------- --------------
      7500 SANA               11 com
```

Outer Joins

DEPARTMENTS

	DEPARTMENT_NAME	DEPARTMENT_ID
1	Administration	10
2	Marketing	20
3	Shipping	50
4	IT	60
5	Sales	80
6	Executive	90
7	Accounting	110
8	Contracting	190

EMPLOYEES

	DEPARTMENT_ID	LAST_NAME
1	10	Whalen
2	20	Hartstein
3	20	Fay
4	110	Higgins
5	110	Gietz
6	90	King
7	90	Kochhar
8	90	De Haan
9	60	Hunold
10	60	Ernst

...

Where we are loosing information for not matching departments.

There are no employees in department 190.

INNER Versus OUTER Joins

- In SQL:1999, the join of two tables returning only matched rows is called an inner join.
- A join between two tables that returns the results of the inner join as well as the unmatched rows from the left (or right) tables is called a left (or right) outer join.
- A join between two tables that returns the results of an inner join as well as the results of a left and right join is a full outer join.

LEFT OUTER JOIN

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e LEFT OUTER JOIN departments d
ON    (e.department_id = d.department_id) ;
```

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Hartstein	20	Marketing
3	Fay	20	Marketing
4	Higgins	110	Accounting

...

18	Abel	80	Sales
19	Taylor	80	Sales
20	Grant	(null)	(null)

LEFT OUTER JOIN

SQL> select * from dept;

DEPTNO	DNAME	LOC
22	IT	Mum
11	com	mumbai
12		
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

7 rows selected.

SQL> select * from empl;

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	20
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7500	SANA	Prof	7499	18-JUN-08	1000	100	11

LEFT OUTER JOIN

```
SQL> select * from dept;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        22 IT             Mum
        11 com            mumbai
        12
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON

7 rows selected.
```

```
SQL> select * from empl;

     EMPNO ENAME      JOB              MGR HIREDATE       SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7499 ALLEN      SALESMAN         7698 20-FEB-81       1600        300         20
      7521 WARD       SALESMAN         7698 22-FEB-81       1250        500         30
      7654 MARTIN     SALESMAN         7698 28-SEP-81       1250       1400         30
      7844 TURNER     SALESMAN         7698 08-SEP-81       1500          0         30
      7500 SANA       Prof             7499 18-JUN-08       1000        100         11
```

```
SQL> select e.empno, e.ename, d.deptno, d.dname
  2   from dept d left outer join emp1 e
  3   on (d.deptno=e.deptno);
```

LEFT OUTER JOIN

```
SQL> select * from dept;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        22 IT             Mum
        11 com            mumbai
        12
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON

7 rows selected.
```

```
SQL> select * from empl;

     EMPNO ENAME      JOB              MGR HIREDATE       SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7499 ALLEN      SALESMAN         7698 20-FEB-81       1600        300         20
      7521 WARD       SALESMAN         7698 22-FEB-81       1250        500         30
      7654 MARTIN     SALESMAN         7698 28-SEP-81       1250       1400         30
      7844 TURNER     SALESMAN         7698 08-SEP-81       1500          0         30
      7500 SANA       Prof             7499 18-JUN-08       1000        100         11
```

```
SQL> select e.empno, e.ename, d.deptno, d.dname
  2   from dept d left outer join emp1 e
  3   on (d.deptno=e.deptno);

     EMPNO ENAME          DEPTNO DNAME
---------- ---------- ---------- --------------
                               10 ACCOUNTING
      7500 SANA                11 com
                               12
      7499 ALLEN               20 RESEARCH
                               22 IT
      7844 TURNER              30 SALES
      7521 WARD                30 SALES
      7654 MARTIN              30 SALES
                               40 OPERATIONS

9 rows selected.
```

RIGHT OUTER JOIN

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e RIGHT OUTER JOIN departments d
ON    (e.department_id = d.department_id) ;
```

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Hartstein	20	Marketing
3	Fay	20	Marketing

...

18	Higgins	110	Accounting
19	Gietz	110	Accounting
20	(null)	(null)	Contracting

FULL OUTER JOIN

```
SELECT e.last_name, d.department_id, d.department_name
FROM   employees e FULL OUTER JOIN departments d
ON    (e.department_id = d.department_id) ;
```

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Hartstein	20	Marketing
3	Fay	20	Marketing

...

18	Abel	80	Sales
19	Taylor	80	Sales
20	Grant	(null)	(null)
21	(null)	190	Contracting

Cartesian Products

- A Cartesian product is formed when:
  - A join condition is omitted
  - A join condition is invalid
  - All rows in the first table are joined to all rows in the second table
- To avoid a Cartesian product, always include a valid join condition.

Generating a Cartesian Product

EMPLOYEES (20 rows)

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	200	Whalen	10
2	201	Hartstein	20
...			
19	176	Taylor	80
20	178	Grant	(null)

DEPARTMENTS (8 rows)

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
1	10	Administration	1700
2	20	Marketing	1800
3	50	Shipping	1500
4	60	IT	1400
5	80	Sales	2500
6	90	Executive	1700
7	110	Accounting	1700
8	190	Contracting	1700

Cartesian product:
20 x 8 = 160 rows

	EMPLOYEE_ID	DEPARTMENT_ID	LOCATION_ID
1	100	10	1700
2	101	10	1700
...			
156	200	190	1700
157	201	190	1700
158	202	190	1700
159	205	190	1700
160	206	190	1700

Creating Cross Joins

- The `CROSS JOIN` clause produces the cross-product of two tables.
- This is also called a Cartesian product between the two tables.

```
SELECT last_name, department_name
FROM   employees
CROSS JOIN departments ;
```

	LAST_NAME	DEPARTMENT_NAME
1	Abel	Administration
2	Davies	Administration
3	De Haan	Administration
4	Ernst	Administration

...

159	Whalen	Contracting
160	Zlotkey	Contracting

```
SQL> select * from dept;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        22 IT             Mum
        11 com            mumbai
        12
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON

7 rows selected.
```

Creating Cross Joins

Select empno, deptno, dname
from dept cross join emp;

```
SQL> select * from empl;

     EMPNO ENAME      JOB              MGR HIREDATE            SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7499 ALLEN      SALESMAN         7698 20-FEB-81       1600        300         20
      7521 WARD       SALESMAN         7698 22-FEB-81       1250        500         30
      7654 MARTIN     SALESMAN         7698 28-SEP-81       1250       1400         30
      7844 TURNER     SALESMAN         7698 08-SEP-81       1500          0         30
      7500 SANA       Prof             7499 18-JUN-08       1000        100         11
```

Discussion Questions

Exercise Questions

Human Resources (HR) Data Set

The Human Resources (HR) schema is part of the Oracle Common Schema that can be installed in an Oracle Database. The practices in this course use the data from the HR schema.

Table Descriptions

REGIONS contains rows representing a region (such as Americas, Asia, and so on).

COUNTRIES contains rows for countries, each of which are associated with a region.

LOCATIONS contains the addresses of specific offices, warehouses, and/or production sites of a company in a particular country.

DEPARTMENTS shows details of the departments in which employees work. Each department can have a relationship representing the department manager in the EMPLOYEES table.

EMPLOYEES contains details about each employee who works for a department. Some employees may not be assigned to any department.

JOBS contains the job types that can be held by each employee.

JOB_HISTORY contains the job history of the employees. If an employee changes departments within the job or changes jobs within the department, a new row is inserted in this table with the old job information of the employee.

JOB_GRADES identifies a salary range per job grade. The salary ranges do not overlap.

COUNTRIES Table

DESCRIBE countries

Name	Null	Type
COUNTRY_ID	NOT NULL	CHAR(2)
COUNTRY_NAME		VARCHAR2(40)
REGION_ID		NUMBER

3 rows selected

EMPLOYEES Table

DESCRIBE employees

Name	Null	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

11 rows selected

JOBS Table

DESCRIBE jobs

Name	Null	Type
JOB_ID	NOT NULL	VARCHAR2(10)
JOB_TITLE	NOT NULL	VARCHAR2(35)
MIN_SALARY		NUMBER(6)
MAX_SALARY		NUMBER(6)

4 rows selected

LOCATIONS Table

DESCRIBE locations

Name	Null	Type
LOCATION_ID	NOT NULL	NUMBER(4)
STREET_ADDRESS		VARCHAR2(40)
POSTAL_CODE		VARCHAR2(12)
CITY	NOT NULL	VARCHAR2(30)
STATE_PROVINCE		VARCHAR2(25)
COUNTRY_ID		CHAR(2)

6 rows selected

JOB_HISTORY Table

DESCRIBE job_history

Name	Null	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
START_DATE	NOT NULL	DATE
END_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
DEPARTMENT_ID		NUMBER(4)

5 rows selected

DEPARTMENTS Table

DESCRIBE departments

Name	Null	Type
DEPARTMENT_ID	NOT NULL	NUMBER(4)
DEPARTMENT_NAME	NOT NULL	VARCHAR2(30)
MANAGER_ID		NUMBER(6)
LOCATION_ID		NUMBER(4)

REGIONS Table

DESCRIBE regions

Name	Null	Type
REGION_ID	NOT NULL	NUMBER
REGION_NAME		VARCHAR2(25)

2 rows selected

JOB_GRADES Table

DESCRIBE job_grades

Name	Null	Type
GRADE_LEVEL		VARCHAR2(3)
LOWEST_SAL		NUMBER
HIGHEST_SAL		NUMBER

3 rows selected

COUNTRIES Table

```
DESCRIBE countries

Name                          Null     Type
----------------------------- -------- ------------
COUNTRY_ID                    NOT NULL CHAR(2)
COUNTRY_NAME                           VARCHAR2(40)
REGION_ID                              NUMBER

3 rows selected
```

LOCATIONS Table

```
DESCRIBE locations

Name                          Null     Type
----------------------------- -------- -------------
LOCATION_ID                   NOT NULL NUMBER(4)
STREET_ADDRESS                         VARCHAR2(40)
POSTAL_CODE                            VARCHAR2(12)
CITY                          NOT NULL VARCHAR2(30)
STATE_PROVINCE                         VARCHAR2(25)
COUNTRY_ID                             CHAR(2)

6 rows selected
```

1. Write a query for the HR department to produce the addresses of all the departments. Use the LOCATIONS and COUNTRIES tables. Show the location ID, street address, city, state or province, and country in the output. Use a NATURAL JOIN to produce the results.

COUNTRIES Table

```
DESCRIBE countries
```

Name	Null	Type
COUNTRY_ID	NOT NULL	CHAR(2)
COUNTRY_NAME		VARCHAR2(40)
REGION_ID		NUMBER

3 rows selected

LOCATIONS Table

```
DESCRIBE locations
```

Name	Null	Type
LOCATION_ID	NOT NULL	NUMBER(4)
STREET_ADDRESS		VARCHAR2(40)
POSTAL_CODE		VARCHAR2(12)
CITY	NOT NULL	VARCHAR2(30)
STATE_PROVINCE		VARCHAR2(25)
COUNTRY_ID		CHAR(2)

6 rows selected

1. Write a query for the HR department to produce the addresses of all the departments. Use the LOCATIONS and COUNTRIES tables. Show the location ID, street address, city, state or province, and country in the output. Use a NATURAL JOIN to produce the results.

```
SELECT location_id, street_address, city, state_province, country_name
FROM   locations
NATURAL JOIN  countries;
```

DEPARTMENTS Table

```
DESCRIBE departments

Name                          Null     Type
----------------------------- -------- ------------
DEPARTMENT_ID                 NOT NULL NUMBER(4)
DEPARTMENT_NAME               NOT NULL VARCHAR2(30)
MANAGER_ID                             NUMBER(6)
LOCATION_ID                            NUMBER(4)

4 rows selected
```

EMPLOYEES Table

```
DESCRIBE employees

Name                          Null     Type
----------------------------- -------- ------------
EMPLOYEE_ID                   NOT NULL NUMBER(6)
FIRST_NAME                             VARCHAR2(20)
LAST_NAME                     NOT NULL VARCHAR2(25)
EMAIL                         NOT NULL VARCHAR2(25)
PHONE_NUMBER                           VARCHAR2(20)
HIRE_DATE                     NOT NULL DATE
JOB_ID                        NOT NULL VARCHAR2(10)
SALARY                                 NUMBER(8,2)
COMMISSION_PCT                         NUMBER(2,2)
MANAGER_ID                             NUMBER(6)
DEPARTMENT_ID                          NUMBER(4)

11 rows selected
```

2. The HR department needs a report of all employees. Write a query to display the last name, department number, and department name for all employees.

DEPARTMENTS Table

DESCRIBE departments

Name	Null	Type
DEPARTMENT_ID	NOT NULL	NUMBER(4)
DEPARTMENT_NAME	NOT NULL	VARCHAR2(30)
MANAGER_ID		NUMBER(6)
LOCATION_ID		NUMBER(4)

4 rows selected

EMPLOYEES Table

DESCRIBE employees

Name	Null	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

11 rows selected

2. The HR department needs a report of all employees. Write a query to display the last name, department number, and department name for all employees.

```
Select e.last_name, e.department_id , d.department_name
from employees e
Join departments d
On (e.department_id = d.department_id);
```

EMPLOYEES Table

```
DESCRIBE employees
```

```
Name                            Null     Type
------------------------------- -------- --------------
EMPLOYEE_ID                     NOT NULL NUMBER(6)
FIRST_NAME                               VARCHAR2(20)
LAST_NAME                       NOT NULL VARCHAR2(25)
EMAIL                           NOT NULL VARCHAR2(25)
PHONE_NUMBER                             VARCHAR2(20)
HIRE_DATE                       NOT NULL DATE
JOB_ID                          NOT NULL VARCHAR2(10)
SALARY                                   NUMBER(8,2)
COMMISSION_PCT                           NUMBER(2,2)
MANAGER_ID                               NUMBER(6)
DEPARTMENT_ID                            NUMBER(4)

11 rows selected
```

4. Create a report to display the last name and employee number of employees along with their manager's last name and manager number. Label the columns `Employee`, `Emp#`, `Manager`, and `Mgr#`, respectively. Place your SQL statement in a text file named `lab_05_04.sql`.

EMPLOYEES Table

```
DESCRIBE employees
```

```
Name                            Null     Type
------------------------------- -------- --------------
EMPLOYEE_ID                     NOT NULL NUMBER(6)
FIRST_NAME                               VARCHAR2(20)
LAST_NAME                       NOT NULL VARCHAR2(25)
EMAIL                           NOT NULL VARCHAR2(25)
PHONE_NUMBER                             VARCHAR2(20)
HIRE_DATE                       NOT NULL DATE
JOB_ID                          NOT NULL VARCHAR2(10)
SALARY                                   NUMBER(8,2)
COMMISSION_PCT                           NUMBER(2,2)
MANAGER_ID                               NUMBER(6)
DEPARTMENT_ID                            NUMBER(4)

11 rows selected
```

4. Create a report to display the last name and employee number of employees along with their manager's last name and manager number. Label the columns `Employee`, `Emp#`, `Manager`, and `Mgr#`, respectively. Place your SQL statement in a text file named `lab_05_04.sql`.

```
SELECT w.last_name "Employee", w.employee_id "EMP#",
       m.last_name "Manager", m.employee_id  "Mgr#"
FROM   employees w join employees m
ON     (w.manager_id = m.employee_id);
```

EMPLOYEES Table

```
DESCRIBE employees
```

```
Name                           Null     Type
------------------------------ -------- -------------
EMPLOYEE_ID                    NOT NULL NUMBER(6)
FIRST_NAME                              VARCHAR2(20)
LAST_NAME                      NOT NULL VARCHAR2(25)
EMAIL                          NOT NULL VARCHAR2(25)
PHONE_NUMBER                            VARCHAR2(20)
HIRE_DATE                      NOT NULL DATE
JOB_ID                         NOT NULL VARCHAR2(10)
SALARY                                  NUMBER(8,2)
COMMISSION_PCT                          NUMBER(2,2)
MANAGER_ID                              NUMBER(6)
DEPARTMENT_ID                           NUMBER(4)

11 rows selected
```

5. Modify `lab_05_04.sql` to display all employees, including King, who has no manager.

EMPLOYEES Table

```
DESCRIBE employees
```

```
Name                            Null     Type
------------------------------- -------- -------------
EMPLOYEE_ID                     NOT NULL NUMBER(6)
FIRST_NAME                               VARCHAR2(20)
LAST_NAME                       NOT NULL VARCHAR2(25)
EMAIL                           NOT NULL VARCHAR2(25)
PHONE_NUMBER                             VARCHAR2(20)
HIRE_DATE                       NOT NULL DATE
JOB_ID                          NOT NULL VARCHAR2(10)
SALARY                                   NUMBER(8,2)
COMMISSION_PCT                           NUMBER(2,2)
MANAGER_ID                               NUMBER(6)
DEPARTMENT_ID                            NUMBER(4)

11 rows selected
```

5. Modify `lab_05_04.sql` to display all employees, including King, who has no manager.

```
SELECT w.last_name "Employee", w.employee_id "EMP#",
       m.last_name "Manager", m.employee_id  "Mgr#"
FROM   employees w
LEFT   OUTER JOIN employees m
ON     (w.manager_id = m.employee_id);
```