Oracle9*i*: Advanced SQL

Student Guide • Volume 2

40058GC11
Production 1.1
November 2001
D34075

Author

Priya Nathan

Technical Contributors and Reviewers

Josephine Turner
Martin Alvarez
Anna Atkinson
Don Bates
Marco Berbeek
Andrew Brannigan
Laszlo Czinkoczki
Michael Gerlach
Sharon Gray
Rosita Hanoman
Mozhe Jalali
Sarah Jones
Charbel Khouri
Christopher Lawless
Diana Lorentz
Nina Minchen
Cuong Nguyen
Daphne Nougier
Patrick Odell
Laura Pezzini
Stacey Procter
Maribel Renau
Bryan Roberts
Helen Robertson
Sunshine Salmon
Casa Sharif
Bernard Soleillant
Craig Spoonemore
Ruediger Steffan
Karla Villasenor
Andree Wheeley
Lachlan Williams

Publisher

Sheryl Domingue

Additional Practices

These exercises can be used for extra practice after you have discussed using SET operators.

1. Find the job that was filled in the first half of 1990 and the same job that was filled during the same period in 1991.

JOB_ID
IT_PROG

2. Write a compound query to produce a list of employees showing raise percentages, employee IDs, and old and new salary increase. Employees in departments 10, 50, and 110 are given a 5% raise, employees in department 60 are given a 10% raise, employees in departments 20 and 80 are given a 15% raise, and employees in department 90 are not given a raise.

RAISE	EMPLOYEE_ID	SALARY	NEW_SALARY
05% raise	124	5800	290
05% raise	141	3500	175
05% raise	142	3100	155
05% raise	143	2600	130
05% raise	144	2500	125
05% raise	200	4400	220
05% raise	205	12000	600
05% raise	206	8300	415
10% raise	103	9000	900
10% raise	104	6000	600
10% raise	107	4200	420
15% raise	149	10500	1575
15% raise	174	11000	1650
15% raise	176	8600	1290
15% raise	201	13000	1950
15% raise	202	6000	900
no raise	100	24000	24000
no raise	101	17000	17000
no raise	102	17000	17000

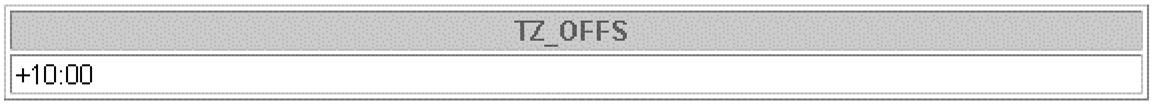
```
19 rows selected.
```

These exercises can be used for extra practice after you have discussed Oracle9*i* single row functions.

Note: The output might be different based on the date when the command is executed.

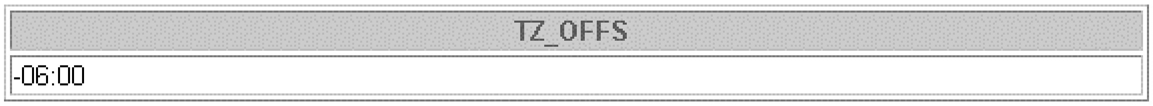
3. Alter the session to set the `NLS_DATE_FORMAT` to `DD-MON-YYYY HH24:MI:SS`.
4. a. Write queries to display the time zone offsets (`TZ_OFFSET`) for the following time zones.

    –Australia/Sydney



TZ_OFFS
+10:00

–Chile/EasterIsland

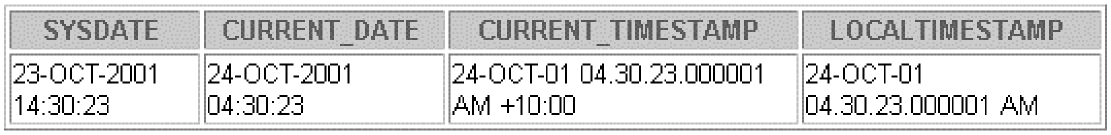


TZ_OFFS
-06:00

b. Alter the session to set the `TIME_ZONE` parameter value to the time zone offset of Australia/Sydney.

c. Display the `SYSDATE`, `CURRENT_DATE`, `CURRENT_TIMESTAMP`, and `LOCALTIMESTAMP` for this session.

Note: The output might be different based on the date when the command is executed.



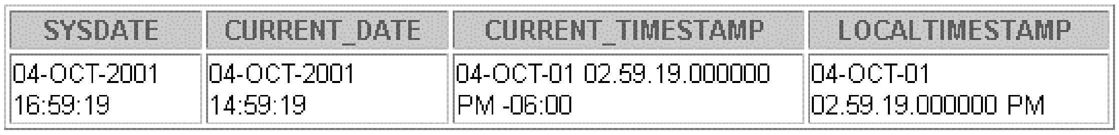
SYSDATE	CURRENT_DATE	CURRENT_TIMESTAMP	LOCALTIMESTAMP
23-OCT-2001 14:30:23	24-OCT-2001 04:30:23	24-OCT-01 04.30.23.000001 AM +10:00	24-OCT-01 04.30.23.000001 AM

d. Alter the session to set the `TIME_ZONE` parameter value to the time zone offset of Chile/EasterIsland.

Note: The results of the preceding question are based on a different date, and in some cases they will not match the actual results that the students get. Also, the time zone offset of the various countries might differ, based on daylight savings time.

e. Display the SYSDATE, CURRENT_DATE, CURRENT_TIMESTAMP, and LOCALTIMESTAMP for this session.

Note: The output might be different based on the date when the command is executed.



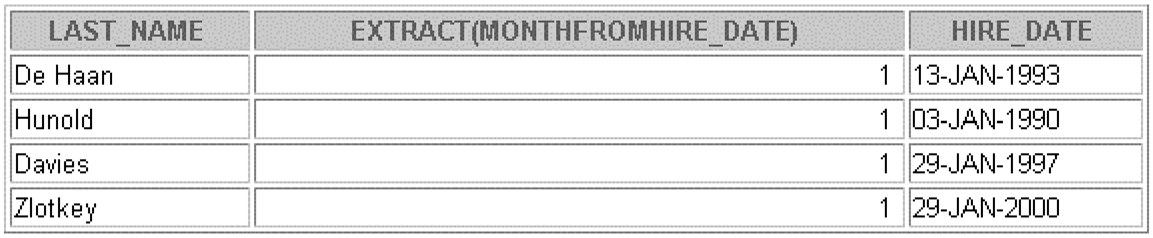
SYSDATE	CURRENT_DATE	CURRENT_TIMESTAMP	LOCALTIMESTAMP
04-OCT-2001 16:59:19	04-OCT-2001 14:59:19	04-OCT-01 02.59.19.000000 PM -06:00	04-OCT-01 02.59.19.000000 PM

Note: Observe in the preceding question that CURRENT_DATE, CURRENT_TIMESTAMP, and LOCALTIMESTAMP are all sensitive to the session time zone. Observe that SYSDATE is not sensitive to the session time zone.

Note: The results of the preceding question are based on a different date, and in some cases they will not match the actual results that the students get. Also the time zone offset of the various countries might differ based on daylight savings time.

f. Alter the session to set the NLS_DATE_FORMAT to DD-MON-YYYY.

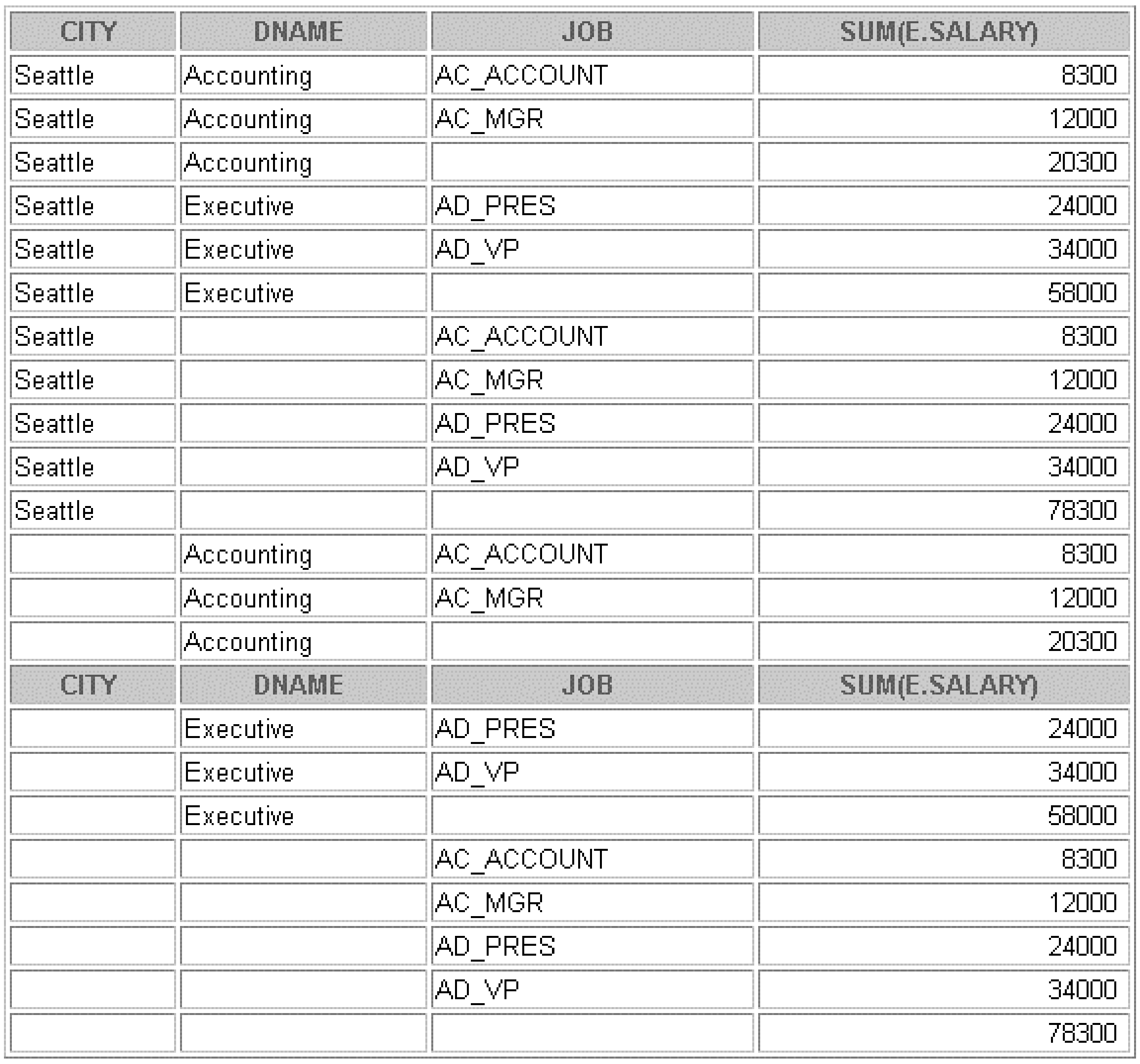
5. Write a query to display the last names, month of the date of join, and hire date of those employees who have joined in the month of January, irrespective of the year of join.



LAST_NAME	EXTRACT(MONTHFROMHIRE_DATE)	HIRE_DATE
De Haan	1	13-JAN-1993
Hunold	1	03-JAN-1990
Davies	1	29-JAN-1997
Zlotkey	1	29-JAN-2000

These exercises can be used for extra practice after you have discussed enhacements to the `GROUP BY` clause.

6. Write a query to display the following for those departments whose department ID is greater than 80:
   – The total salary for every job within a department
   – The total salary
   – The total salary for those cities in which the departments are located
   – The total salary for every job, irrespective of the department
   – The total salary for every department irrespective of the city
   – The total salary of the cities in which the departments are located
   – Total salary for the departments, irrespective of job titles and cities



CITY	DNAME	JOB	SUM(E.SALARY)
Seattle	Accounting	AC_ACCOUNT	8300
Seattle	Accounting	AC_MGR	12000
Seattle	Accounting		20300
Seattle	Executive	AD_PRES	24000
Seattle	Executive	AD_VP	34000
Seattle	Executive		58000
Seattle		AC_ACCOUNT	8300
Seattle		AC_MGR	12000
Seattle		AD_PRES	24000
Seattle		AD_VP	34000
Seattle			78300
	Accounting	AC_ACCOUNT	8300
	Accounting	AC_MGR	12000
	Accounting		20300
	Executive	AD_PRES	24000
	Executive	AD_VP	34000
	Executive		58000
		AC_ACCOUNT	8300
		AC_MGR	12000
		AD_PRES	24000
		AD_VP	34000
			78300

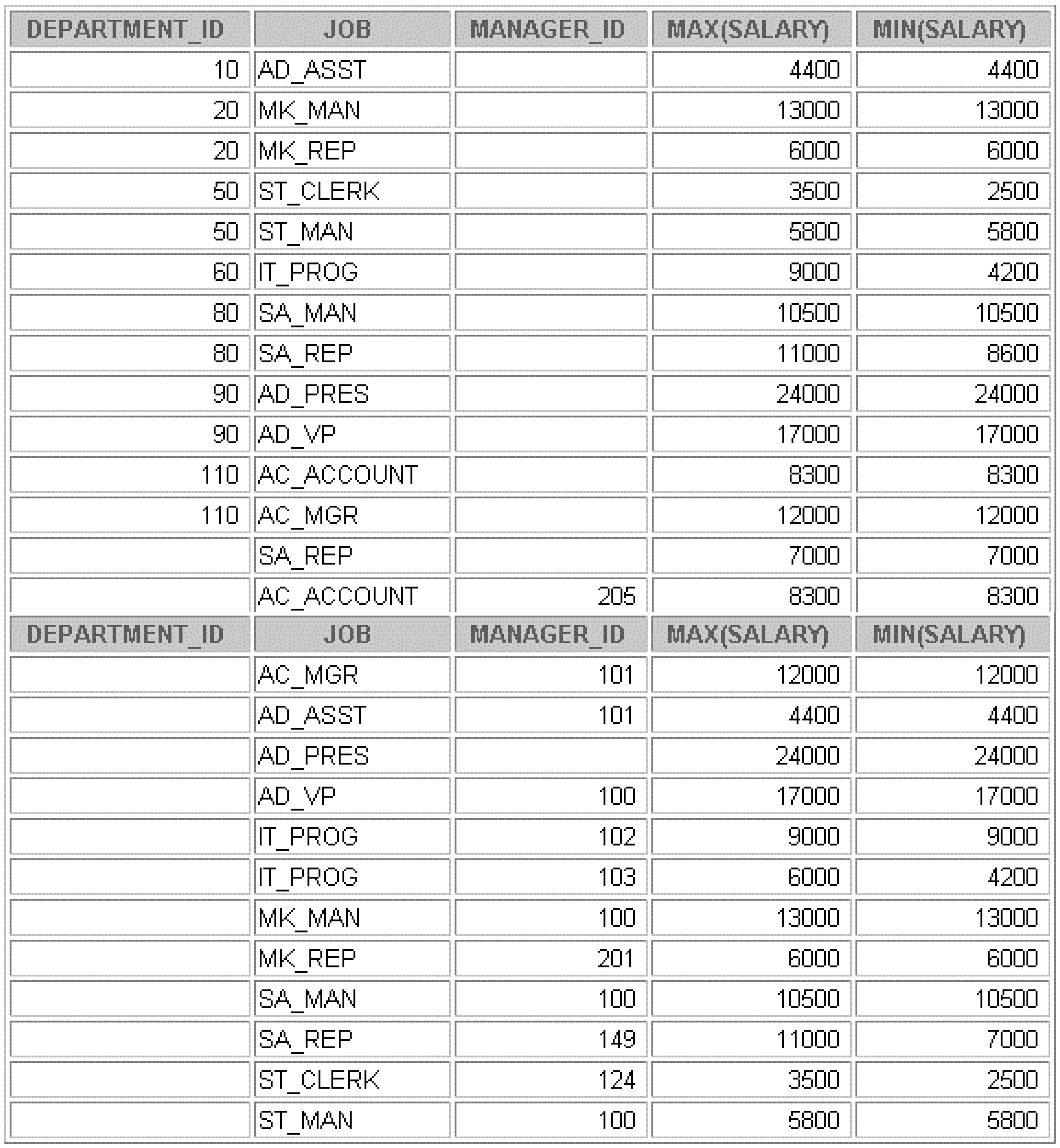
22 rows selected.

7. Write a query to display the following groupings:

   – Department ID, Job ID

   – Job ID, Manager ID

   The query should calculate the maximum and minimum salaries for each of these groups.

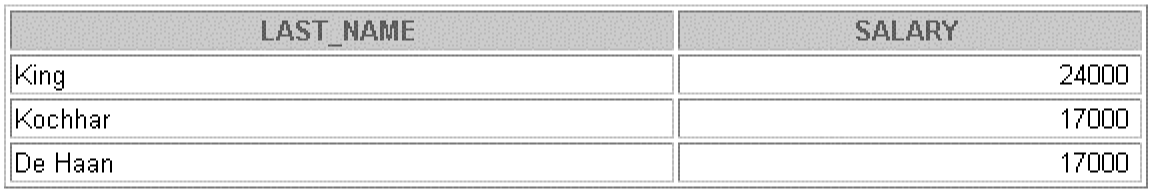


DEPARTMENT_ID	JOB	MANAGER_ID	MAX(SALARY)	MIN(SALARY)
10	AD_ASST		4400	4400
20	MK_MAN		13000	13000
20	MK_REP		6000	6000
50	ST_CLERK		3500	2500
50	ST_MAN		5800	5800
60	IT_PROG		9000	4200
80	SA_MAN		10500	10500
80	SA_REP		11000	8600
90	AD_PRES		24000	24000
90	AD_VP		17000	17000
110	AC_ACCOUNT		8300	8300
110	AC_MGR		12000	12000
	SA_REP		7000	7000
	AC_ACCOUNT	205	8300	8300
	AC_MGR	101	12000	12000
	AD_ASST	101	4400	4400
	AD_PRES		24000	24000
	AD_VP	100	17000	17000
	IT_PROG	102	9000	9000
	IT_PROG	103	6000	4200
	MK_MAN	100	13000	13000
	MK_REP	201	6000	6000
	SA_MAN	100	10500	10500
	SA_REP	149	11000	7000
	ST_CLERK	124	3500	2500
	ST_MAN	100	5800	5800

26 rows selected.

These exercises can be used for extra practice after you have discussed advanced subqueries.

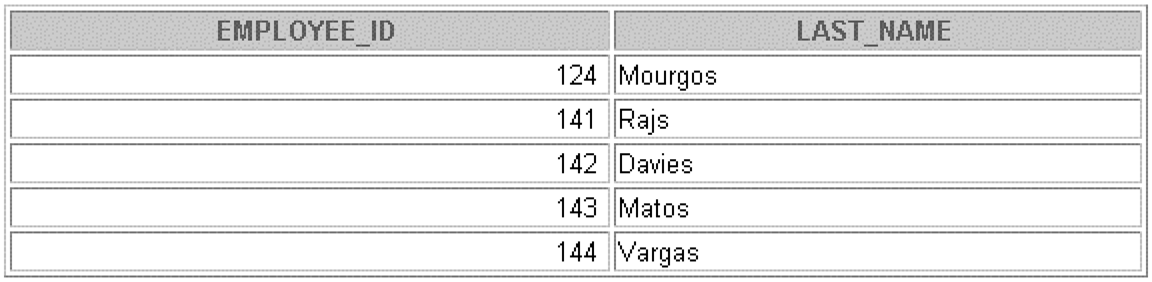
8. Write a query to display the top three earners in the `EMPLOYEES` table. Display their last names and salaries.



LAST_NAME	SALARY
King	24000
Kochhar	17000
De Haan	17000

9. Write a query to display the employee ID and last names of the employees who work in the state of California.

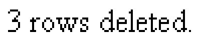
   Hint: Use scalar subqueries.



EMPLOYEE_ID	LAST_NAME
124	Mourgos
141	Rajs
142	Davies
143	Matos
144	Vargas

10. Write a query to delete the oldest `JOB_HISTORY` row of an employee by looking up the `JOB_HISTORY` table for the `MIN(START_DATE)` for the employee. Delete the records of **only** those employees who have changed at least two jobs. If your query executes correctly, you will get the feedback:

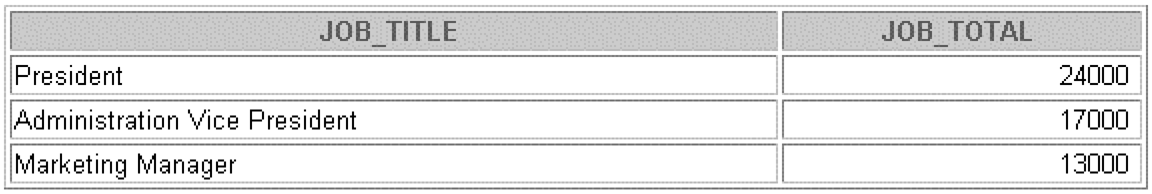
    Hint: Use a correlated `DELETE` command.



```
3 rows deleted.
```

11. Roll back the transaction.

12. Write a query to display the job IDs of those jobs whose maximum salary is above half the maximum salary in the whole company. Use the `WITH` clause to write this query. Name the query `MAX_SAL_CALC`.

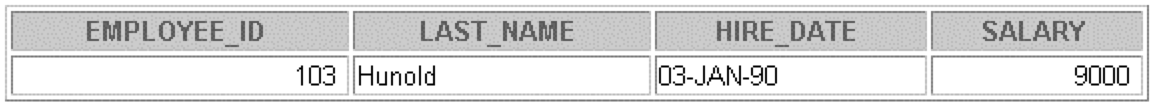


JOB_TITLE	JOB_TOTAL
President	24000
Administration Vice President	17000
Marketing Manager	13000

These exercises can be used for extra practice after you have discussed hierarchical retrieval.

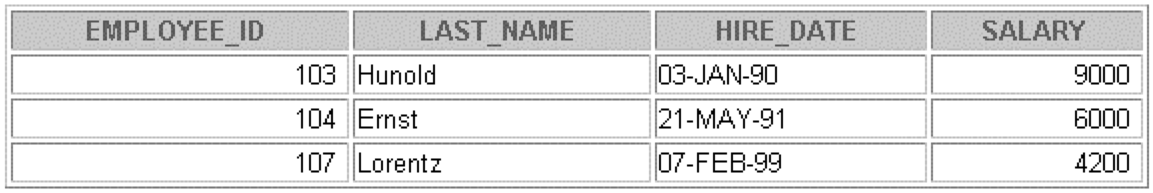
13. Write a SQL statement to display employee number, last name, start date, and salary, showing:

    a. De Haan’s direct reports



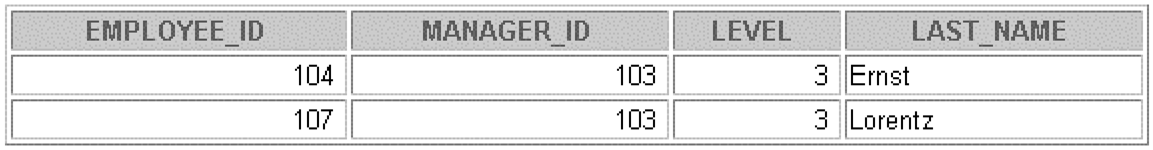
EMPLOYEE_ID	LAST_NAME	HIRE_DATE	SALARY
103	Hunold	03-JAN-90	9000

    b. The organization tree under De Haan (employee number 102)



EMPLOYEE_ID	LAST_NAME	HIRE_DATE	SALARY
103	Hunold	03-JAN-90	9000
104	Ernst	21-MAY-91	6000
107	Lorentz	07-FEB-99	4200

14. Write a hierarchical query to display the employee number, manager number, and employee last name for all employees who are two levels below employee De Haan (employee number 102). Also display the level of the employee.



EMPLOYEE_ID	MANAGER_ID	LEVEL	LAST_NAME
104	103	3	Ernst
107	103	3	Lorentz

15. Produce a hierarchical report to display the employee number, manager number, the `LEVEL` pseudocolumn, and employee last name. For every row in the `EMPLOYEES` table, you should print a tree structure showing the employee, the employee's manager, then the manager's manager, and so on. Use indentations for the `NAME` column.

EMPLOYEE_ID	MANAGER_ID	LEVEL	LAST_NAME
100		1	King
101	100	1	Kochhar
100		2	__King
102	100	1	De Haan
100		2	__King
103	102	1	Hunold
102	100	2	__De Haan
100		3	____King
104	103	1	Ernst
103	102	2	__Hunold
102	100	3	____De Haan
100		4	______King
107	103	1	Lorentz
103	102	2	__Hunold

▪ ▪ ▪

EMPLOYEE_ID	MANAGER_ID	LEVEL	LAST_NAME
101	100	2	__Kochhar
100		3	____King
201	100	1	Hartstein
100		2	__King
202	201	1	Fay
201	100	2	__Hartstein
100		3	____King
205	101	1	Higgins
101	100	2	__Kochhar
100		3	____King
206	205	1	Gietz
205	101	2	__Higgins
101	100	3	____Kochhar
100		4	______King

56 rows selected.

Note: The output shown is only a sample. All the rows from the actual output are not included here.

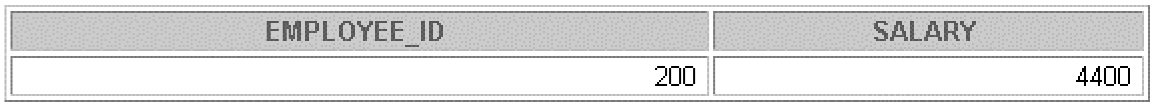
These exercises can be used for extra practice after you have discussed Oracle 9*i* extensions to DML and DDL statements.

Note: Run the `cre_special_sal.sql` , `cre_sal_history.sql` , `cre_mgr_history.sql` scripts in the lab folder to create the `SPECIAL_SAL`, `SAL_HISTORY` and `MGR_HISTORY` tables.

16. Write a query to do the following:
    – Retrieve the details of the employee ID, hire date, salary, and manager ID of those employees whose employee ID is more than or equal to 200 from the `EMPLOYEES` table.
    – If the salary is less than $5,000, insert the details of employee ID and salary into the `SPECIAL_SAL` table.
    – Insert the details of employee ID, hire date, and salary into the `SAL_HISTORY` table.
    – Insert the details of employee ID, manager ID, and salary into the `MGR_HISTORY` table.

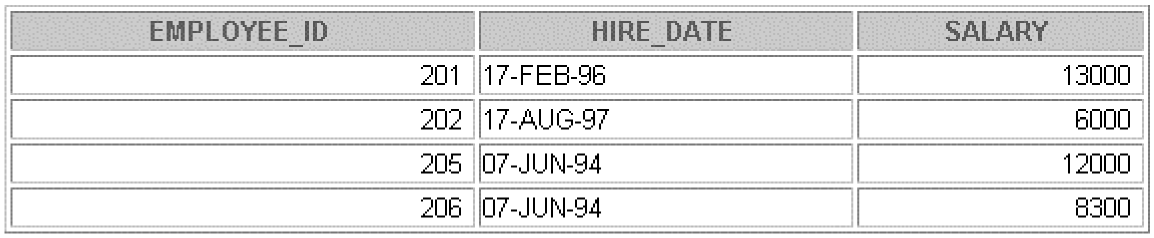
17. Query the `SPECIAL_SAL`, `SAL_HISTORY` and the `MGR_HISTORY` tables to view the inserted records.

`SPECIAL_SAL` Table



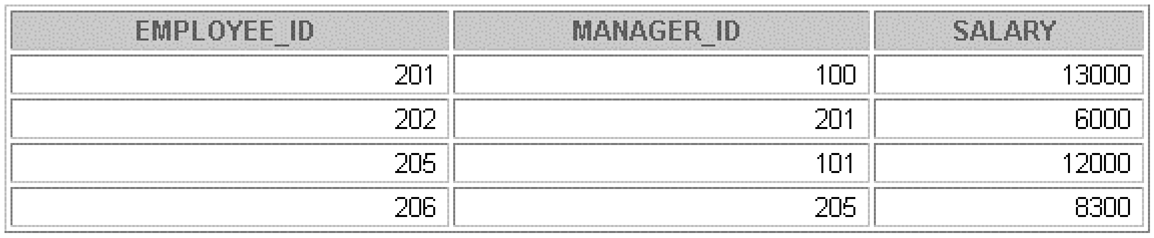
EMPLOYEE_ID	SALARY
200	4400

`SAL_HISTORY` Table



EMPLOYEE_ID	HIRE_DATE	SALARY
201	17-FEB-96	13000
202	17-AUG-97	6000
205	07-JUN-94	12000
206	07-JUN-94	8300

`MGR_HISTORY` Table

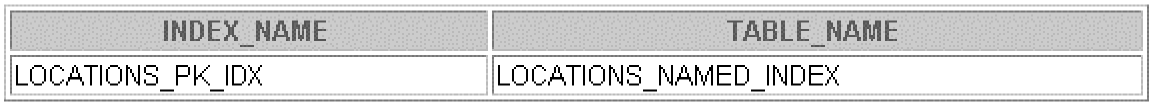


EMPLOYEE_ID	MANAGER_ID	SALARY
201	100	13000
202	201	6000
205	101	12000
206	205	8300

18. Create the `LOCATIONS_NAMED_INDEX` table based on the following table instance chart. Name the index for the `PRIMARY KEY` column as `LOCATIONS_PK_IDX`.

COLUMN Name	**Deptno**	**Dname**
Primary Key	Yes	
Datatype	Number	VARCHAR2
Length	4	30

19. Query the `USER_INDEXES` table to display the `INDEX_NAME` for the `LOCATIONS_NAMED_INDEX` table.

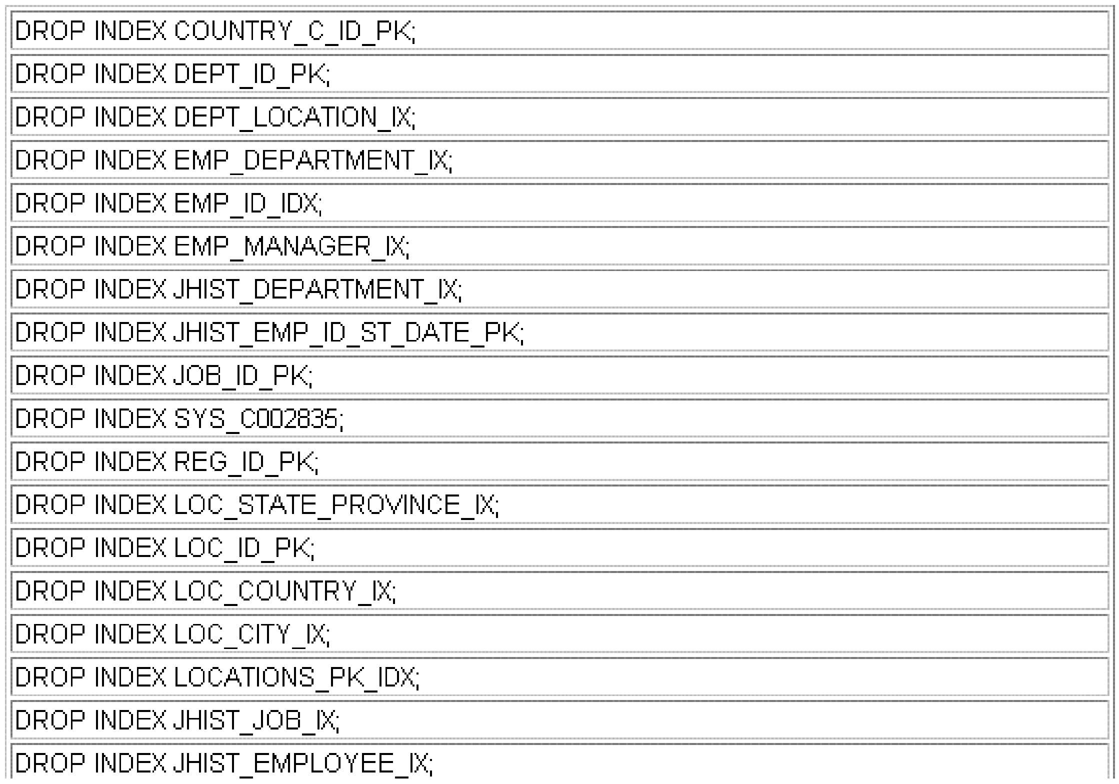


INDEX_NAME	TABLE_NAME
LOCATIONS_PK_IDX	LOCATIONS_NAMED_INDEX

This exercise can be used for extra practice after you have discussed writing advanced scripts.

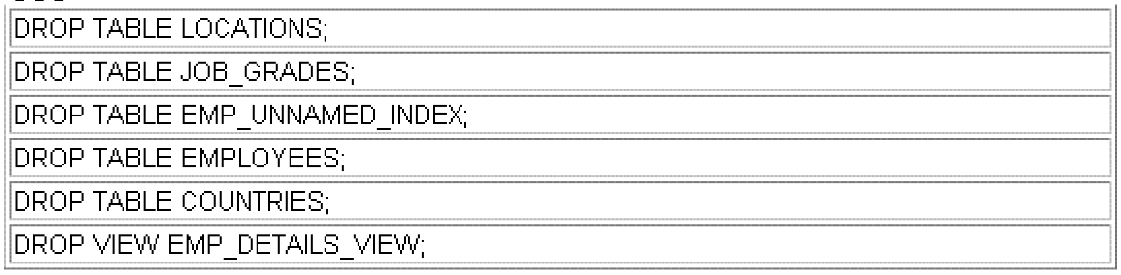
20. Write a SQL script file to drop all objects (tables, views, indexes, sequences, synonyms, and so on) that you own.

Note: The output shown is only a guideline.



```
DROP INDEX COUNTRY_C_ID_PK;
DROP INDEX DEPT_ID_PK;
DROP INDEX DEPT_LOCATION_IX;
DROP INDEX EMP_DEPARTMENT_IX;
DROP INDEX EMP_ID_IDX;
DROP INDEX EMP_MANAGER_IX;
DROP INDEX JHIST_DEPARTMENT_IX;
DROP INDEX JHIST_EMP_ID_ST_DATE_PK;
DROP INDEX JOB_ID_PK;
DROP INDEX SYS_C002835;
DROP INDEX REG_ID_PK;
DROP INDEX LOC_STATE_PROVINCE_IX;
DROP INDEX LOC_ID_PK;
DROP INDEX LOC_COUNTRY_IX;
DROP INDEX LOC_CITY_IX;
DROP INDEX LOCATIONS_PK_IDX;
DROP INDEX JHIST_JOB_IX;
DROP INDEX JHIST_EMPLOYEE_IX;
```

▪ ▪ ▪



```
DROP TABLE LOCATIONS;
DROP TABLE JOB_GRADES;
DROP TABLE EMP_UNNAMED_INDEX;
DROP TABLE EMPLOYEES;
DROP TABLE COUNTRIES;
DROP VIEW EMP_DETAILS_VIEW;
```

Additional Practice Solutions

These exercises can be used for extra practice after you have discussed SET operators.

1. Find the job that was filled in the first half of 1990 and the same job that was filled during the same period in 1991.

```
SELECT job_id
FROM   employees
WHERE  hire_date
BETWEEN '01-JAN-1990' AND '30-JUN-1990'
INTERSECT
SELECT job_id
FROM   employees
WHERE  hire_date BETWEEN '01-JAN-1991'
AND '30-JUN-1991';
```

2. Write a compound query to produce a list of employees showing raise percentages, employee IDs, and old and new salaries. Employees in departments 10, 50, and 110 are given a 5% raise, employees in department 60 are given a 10% raise, employees in departments 20 and 80 are given a 15% raise, and employees in department 90 are not given a raise.

```
SELECT '05% raise' raise, employee_id, salary,
       salary *.05 new_salary
FROM   employees
WHERE  department_id IN (10,50, 110)
UNION
SELECT '10% raise', employee_id, salary, salary * .10
FROM   employees
WHERE  department_id = 60
UNION
SELECT '15% raise', employee_id, salary, salary * .15
FROM   employees
WHERE  department_id IN (20, 80)
UNION
SELECT 'no raise', employee_id, salary, salary
FROM   employees
WHERE  department_id = 90;
```

These exercises can be used for extra practice after you have discussed Oracle9*i* single row functions.

3. Alter the session to set the `NLS_DATE_FORMAT` to `DD-MON-YYYY HH24:MI:SS`.

```
ALTER SESSION
SET NLS_DATE_FORMAT = 'DD-MON-YYYY HH24:MI:SS';
```

4. a. Write queries to display the time zone offsets (`TZ_OFFSET`), for the following time zones.

   – Australia/Sydney

```
SELECT TZ_OFFSET ('Australia/Sydney') from dual;
```

   – Chile/EasterIsland

```
SELECT TZ_OFFSET ('Chile/EasterIsland') from dual;
```

   b. Alter the session to set the `TIME_ZONE` parameter value to the time zone offset of Australia/Sydney.

```
ALTER SESSION SET TIME_ZONE = '+10:00';
```

   c. Display the `SYSDATE`, `CURRENT_DATE`, `CURRENT_TIMESTAMP`, and `LOCALTIMESTAMP` for this session. **Note**: The output might be different based on the date when the command is executed.

```
SELECT SYSDATE,CURRENT_DATE,
CURRENT_TIMESTAMP, LOCALTIMESTAMP
FROM DUAL;
```

   d. Alter the session to set the `TIME_ZONE` parameter value to the time zone offset of Chile/EasterIsland.

```
ALTER SESSION SET TIME_ZONE = '-06:00';
```

e. Display the `SYSDATE, CURRENT_DATE`, `CURRENT_TIMESTAMP`, and `LOCALTIMESTAMP` for this session.

Note: The output might be different based on the date when the command is executed.

```
SELECT SYSDATE,CURRENT_DATE, CURRENT_TIMESTAMP, LOCALTIMESTAMP
FROM   DUAL;
```

f. Alter the session to set the `NLS_DATE_FORMAT` to `DD-MON-YYYY`.

```
ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MON-YYYY';
```

Note: Observe in the preceding question that `CURRENT_DATE`, `CURRENT_TIMESTAMP`, and `LOCALTIMESTAMP` are all sensitive to the session time zone. Observe that `SYSDATE` is not sensitive to the session time zone.

Note: The results of the preceding question are based on a different date, and in some cases they will not match the actual results that the students get. Also the time zone offset of the various countries might differ, based on daylight savings time.

5. Write a query to display the last names, month of the date of join, and hire date of those employees who have joined in the month of January, irrespective of the year of join.

```
SELECT last_name, EXTRACT (MONTH FROM HIRE_DATE),HIRE_DATE
FROM employees
WHERE EXTRACT (MONTH FROM HIRE_DATE) = 1;
```

These exercises can be used for extra practice after you have discussed enhacements to the `GROUP BY` clause

6. Write a query to display the following for those departments whose department ID is greater than 80:
   - The total salary for every job within a department
   - The total salary
   - The total salary for those cities in which the departments are located
   - The total salary for every job, irrespective of the department
   - The total salary for every department irrespective of the city
   - The total salary of the cities in which the departments are located
   - Total salary for the departments, irrespective of job titles and cities

```
COLUMN city FORMAT A25 Heading CITY
COLUMN department_name FORMAT A15 Heading DNAME
COLUMN job_id  FORMAT A10 Heading JOB
COLUMN SUM(salary)  FORMAT $99,99,999.00 Heading SUM(SALARY)

SELECT   l.city,d.department_name, e.job_id, SUM(e.salary)
FROM     locations l,employees e,departments d
WHERE    d.location_id = l.location_id
AND      e.department_id = d.department_id
AND      e.department_id > 80
GROUP    BY CUBE( l.city,d.department_name, e.job_id);
```

7. Write a query to display the following groupings:

   – Department ID, Job ID

   – Job ID, Manager ID

The query should calculate the maximum and minimum salaries for each of these groups.

```
SELECT department_id, job_id, manager_id,max(salary),min(salary)
FROM   employees
GROUP BY GROUPING SETS
((department_id,job_id), (job_id,manager_id));
```

These exercises can be used for extra practice after you have discussed advanced subqueries.

8. Write a query to display the top three earners in the `EMPLOYEES` table. Display their last names and salaries.

```
SELECT last_name, salary
FROM   employees e
WHERE  3  > (SELECT COUNT(*)
             FROM   employees
             WHERE  e.salary < salary);
```

9. Write a query to display the employee ID and last names of the employees who work in the state of California.

   Hint: Use scalar subqueries.

```
SELECT employee_id, last_name
FROM employees e
WHERE ((SELECT location_id
        FROM departments d
        WHERE e.department_id = d.department_id )
              IN   (SELECT location_id
                    FROM locations l
                    WHERE STATE_province = 'California'));
```

10. Write a query to delete the oldest `JOB_HISTORY` row of an employee by looking up the `JOB_HISTORY` table for the `MIN(START_DATE)` for the employee. Delete the records of *only* those employees who have changed at least two jobs. If your query executes correctly, you will get the following feedback:

    Hint: Use a correlated `DELETE` command.

```
DELETE FROM job_history JH
WHERE employee_id =
                  (SELECT employee_id
                   FROM employees E
                   WHERE JH.employee_id = E.employee_id
                   AND START_DATE = (SELECT MIN(start_date)
                            FROM job_history JH
                             WHERE JH.employee_id = E.employee_id)
                            AND 3 >  (SELECT COUNT(*)
                            FROM job_history JH
                             WHERE JH.employee_id = E.employee_id
                             GROUP BY EMPLOYEE_ID
                             HAVING COUNT(*) >= 2));
```

11. Roll back the transaction.

```
ROLLBACK;
```

12. Write a query to display the job IDs of those jobs whose maximum salary is above half the maximum salary in the whole company. Use the `WITH` clause to write this query. Name the query `MAX_SAL_CALC`.

```
WITH
MAX_SAL_CALC AS (
SELECT job_title, MAX(salary) AS job_total
FROM employees, jobs
WHERE employees.job_id = jobs.job_id
GROUP BY job_title)
SELECT job_title, job_total
FROM MAX_SAL_CALC
WHERE job_total > (SELECT MAX(job_total) * 1/2
                   FROM MAX_SAL_CALC)
ORDER BY job_total DESC;
```

These exercises can be used for extra practice after you have discussed hierarchical retrieval.

13. Write a SQL statement to display employee number, last name, start date, and salary, showing:

a. De Haan's direct reports

```
SELECT employee_id, last_name, hire_date, salary
FROM   employees
WHERE  manager_id = (SELECT employee_id
                     FROM   employees
                     WHERE last_name = 'De Haan');
```

b. The organization tree under De Haan (employee number 102)

```
SELECT employee_id, last_name, hire_date, salary
FROM   employees
WHERE  employee_id != 102
CONNECT BY manager_id = PRIOR employee_id
START WITH employee_id = 102;
```

14. Write a hierarchical query to display the employee number, manager number, and employee last name for all employees who are two levels below employee De Haan (employee number 102). Also display the level of the employee.

```
SELECT employee_id, manager_id, level, last_name
FROM   employees
WHERE LEVEL = 3
CONNECT BY manager_id = PRIOR employee_id
START WITH employee_id= 102;
```

15. Produce a hierarchical report to display employee number, manager number, the `LEVEL` pseudocolumn, and employee last name. For every row in the `EMPLOYEES` table, you should print a tree structure showing the employee, the employee's manager, then the manager's manager, and so on. Use indentations for the `NAME` column.

```
COLUMN name FORMAT A25
SELECT  employee_id, manager_id, LEVEL,
LPAD(last_name, LENGTH(last_name)+(LEVEL*2)-2,'_') LAST_NAME
FROM    employees
CONNECT BY employee_id = PRIOR manager_id;
COLUMN name CLEAR
```

These exercises can be used for extra practice after you have discussed Oracle 9*i* extensions to DML and DDL.

16. Write a query to do the following:
    - Retrieve the details of the employee ID, hire date, salary, and manager ID of those employees whose employee ID is more than or equal to 200 from the `EMPLOYEES` table.
    - If the salary is less than $5,000, insert the details of employee ID and salary into the `SPECIAL_SAL` table.
    - Insert the details of employee ID, hire date, and salary into the `SAL_HISTORY` table.
    - Insert the details of employee ID, manager ID, and salary into the `MGR_HISTORY` table.

```
INSERT ALL
WHEN SAL < 5000 THEN
INTO  special_sal VALUES (EMPID, SAL)
ELSE
INTO sal_history VALUES(EMPID,HIREDATE,SAL)
INTO mgr_history VALUES(EMPID,MGR,SAL)
SELECT employee_id EMPID, hire_date HIREDATE,
       salary SAL, manager_id MGR
FROM employees
WHERE employee_id >=200;
```

17. Query the `SPECIAL_SAL`, `SAL_HISTORY` and the `MGR_HISTORY` tables to view the inserted records.

```
SELECT * FROM special_sal;
SELECT * FROM sal_history;
SELECT * FROM mgr_history;
```

18. Create the `LOCATIONS_NAMED_INDEX` table based on the following table instance chart. Name the index for the `PRIMARY KEY` column as `LOCATIONS_PK_IDX`.

```
CREATE TABLE LOCATIONS_NAMED_INDEX
(location_id NUMBER(4) PRIMARY KEY USING INDEX
(CREATE INDEX locations_pk_idx ON
LOCATIONS_NAMED_INDEX(location_id)),
location_name VARCHAR2(20));
```

19. Query the `USER_INDEXES` table to display the `INDEX_NAME` for the `LOCATIONS_NAMED_INDEX` table.

```
SELECT INDEX_NAME, TABLE_NAME
FROM USER_INDEXES
WHERE TABLE_NAME = 'LOCATIONS_NAMED_INDEX';
```

This exercise can be used for extra practice after you have discussed writing advanced scripts.

20. Write a SQL script file to drop all objects (tables, views, indexes, sequences, synonyms, and so on) that you own. **Note:** The output shown is only a guideline.

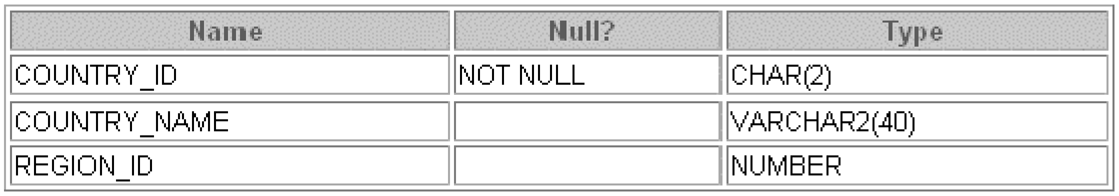
```
SET HEADING OFF ECHO OFF FEEDBACK OFF
SET PAGESIZE 0


SELECT   'DROP ' || object_type || ' ' || object_name || ';'
FROM     user_objects
ORDER BY object_type
/


SET HEADING ON ECHO ON FEEDBACK ON
SET PAGESIZE 24
```

Additional Practices

Table Descriptions and Data

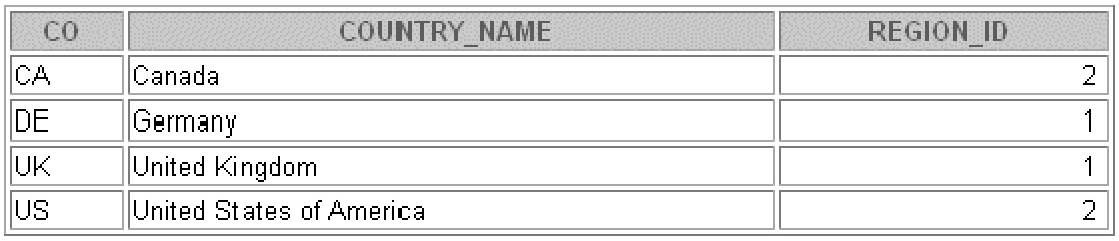
COUNTRIES Table

```
DESCRIBE countries
```



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

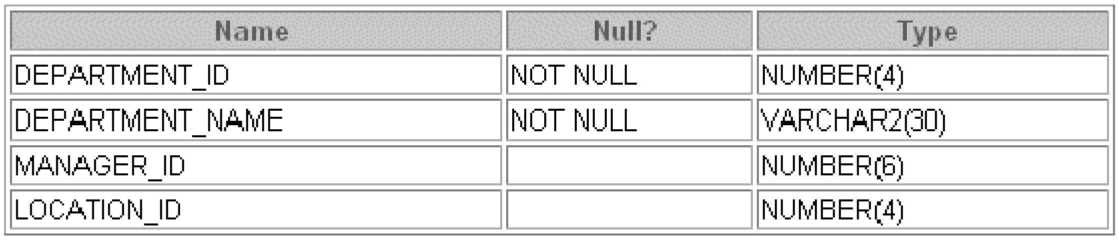
```
SELECT * FROM countries;
```



CO	COUNTRY_NAME	REGION_ID
CA	Canada	2
DE	Germany	1
UK	United Kingdom	1
US	United States of America	2

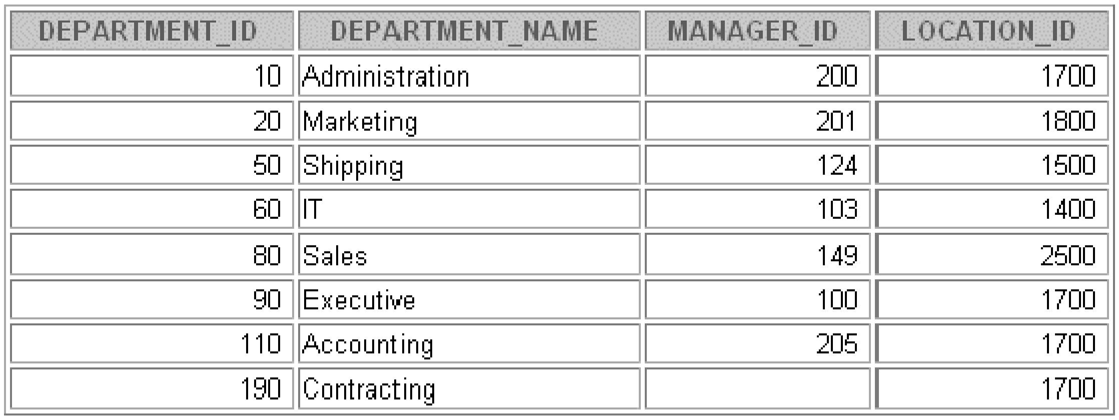
DEPARTMENTS Table

```
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)

```
SELECT * FROM departments;
```

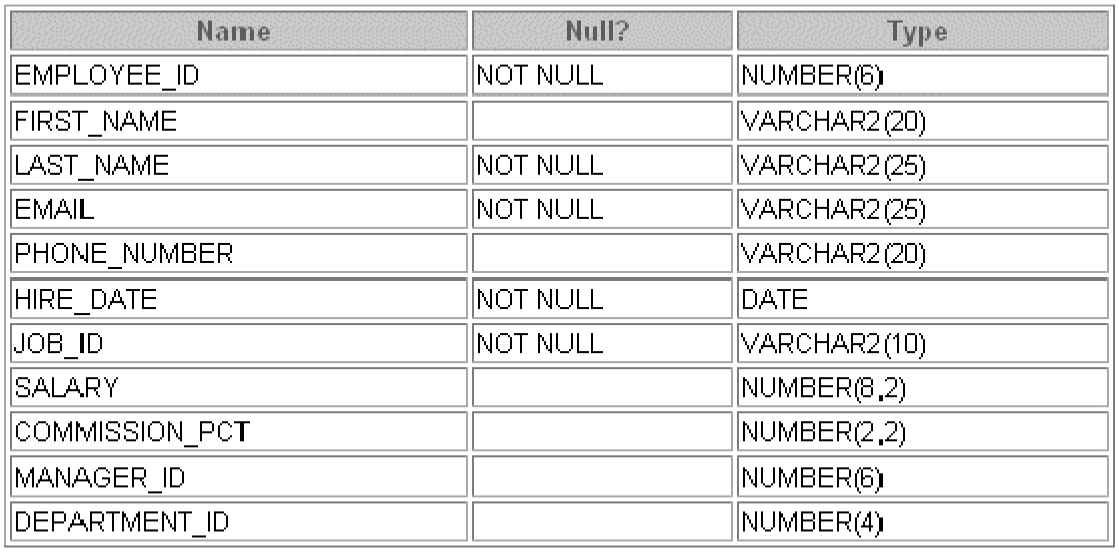


DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700

8 rows selected.

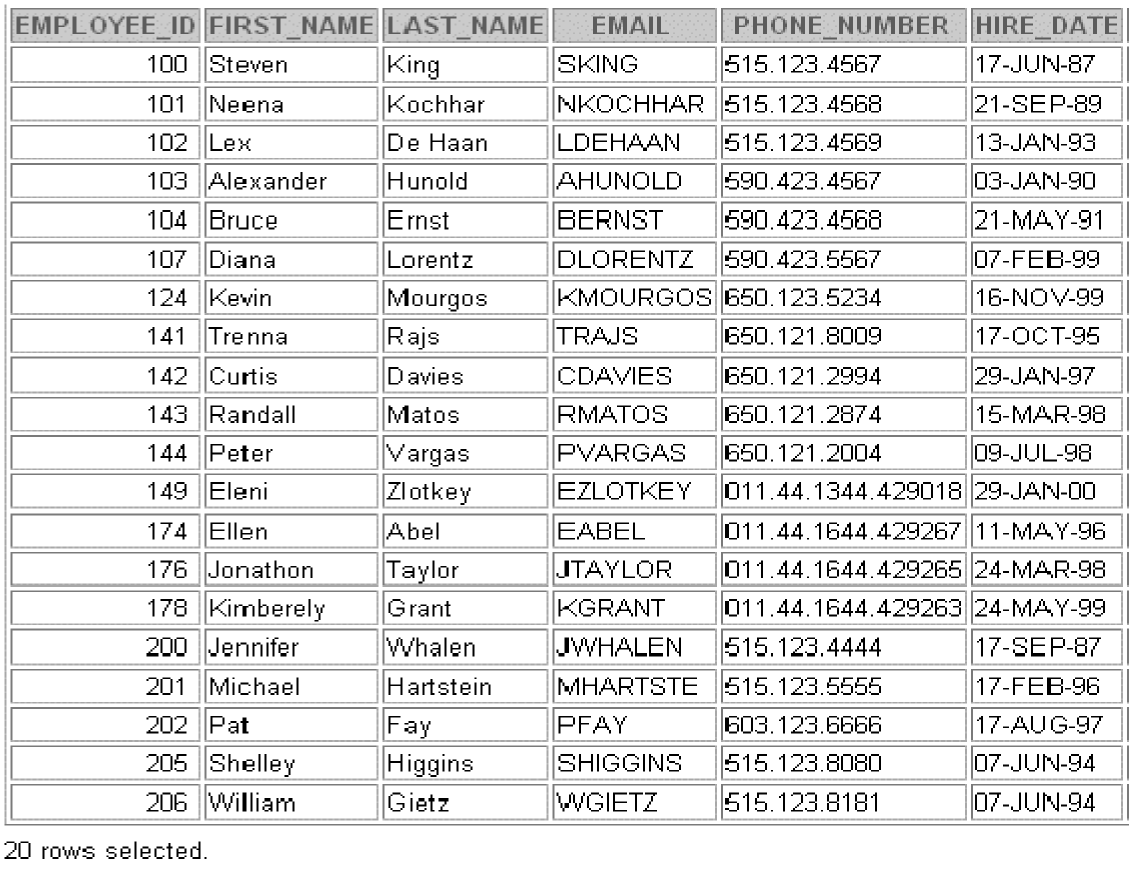
EMPLOYEES Table

```
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)

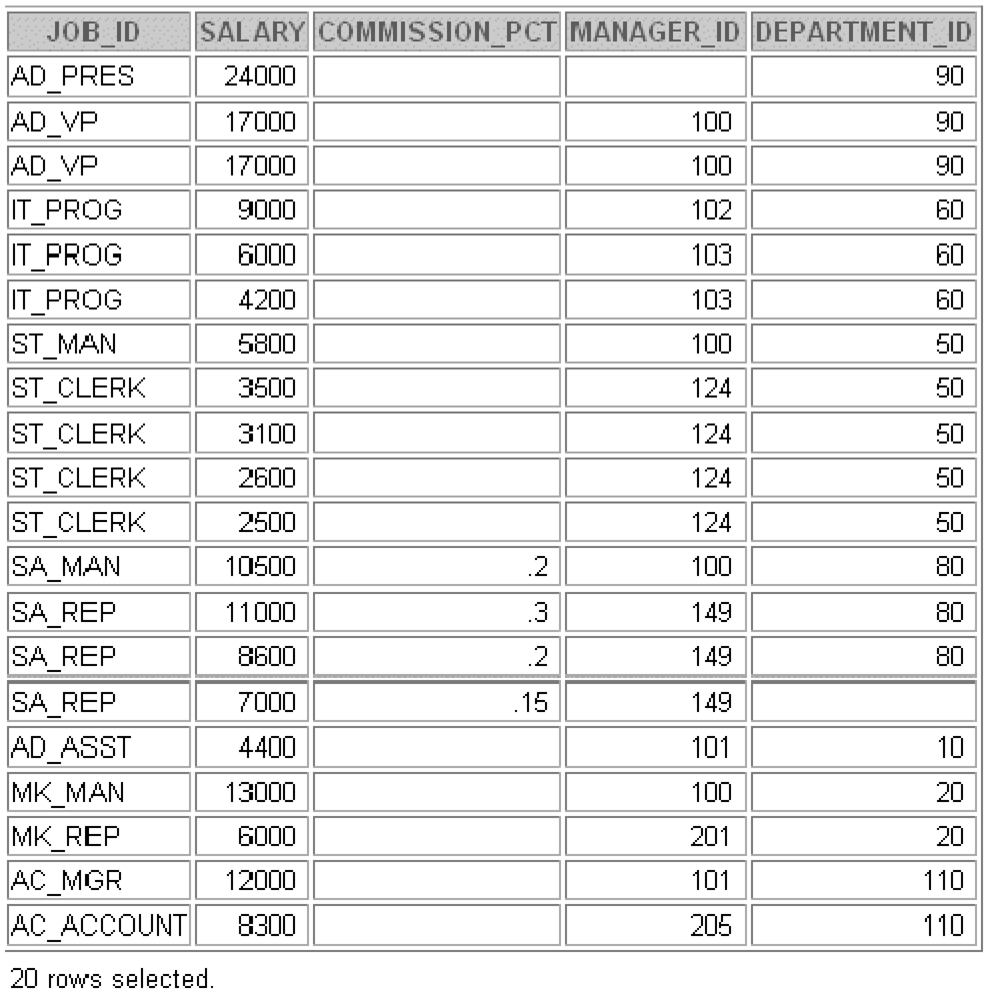
```
SELECT * FROM employees;
```



EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE
100	Steven	King	SKING	515.123.4567	17-JUN-87
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-93
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-90
104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-91
107	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-99
124	Kevin	Mourgos	KMOURGOS	650.123.5234	16-NOV-99
141	Trenna	Rajs	TRAJS	650.121.8009	17-OCT-95
142	Curtis	Davies	CDAVIES	650.121.2994	29-JAN-97
143	Randall	Matos	RMATOS	650.121.2874	15-MAR-98
144	Peter	Vargas	PVARGAS	650.121.2004	09-JUL-98
149	Eleni	Zlotkey	EZLOTKEY	011.44.1344.429018	29-JAN-00
174	Ellen	Abel	EABEL	011.44.1644.429267	11-MAY-96
176	Jonathon	Taylor	JTAYLOR	011.44.1644.429265	24-MAR-98
178	Kimberely	Grant	KGRANT	011.44.1644.429263	24-MAY-99
200	Jennifer	Whalen	JWHALEN	515.123.4444	17-SEP-87
201	Michael	Hartstein	MHARTSTE	515.123.5555	17-FEB-96
202	Pat	Fay	PFAY	603.123.6666	17-AUG-97
205	Shelley	Higgins	SHIGGINS	515.123.8080	07-JUN-94
206	William	Gietz	WGIETZ	515.123.8181	07-JUN-94

20 rows selected.

`EMPLOYEES` Table (continued)

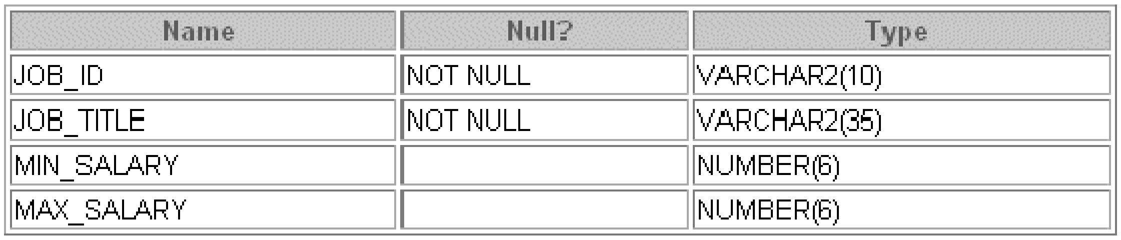


JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
AD_PRES	24000			90
AD_VP	17000		100	90
AD_VP	17000		100	90
IT_PROG	9000		102	60
IT_PROG	6000		103	60
IT_PROG	4200		103	60
ST_MAN	5800		100	50
ST_CLERK	3500		124	50
ST_CLERK	3100		124	50
ST_CLERK	2600		124	50
ST_CLERK	2500		124	50
SA_MAN	10500	.2	100	80
SA_REP	11000	.3	149	80
SA_REP	8600	.2	149	80
SA_REP	7000	.15	149	
AD_ASST	4400		101	10
MK_MAN	13000		100	20
MK_REP	6000		201	20
AC_MGR	12000		101	110
AC_ACCOUNT	8300		205	110

20 rows selected.

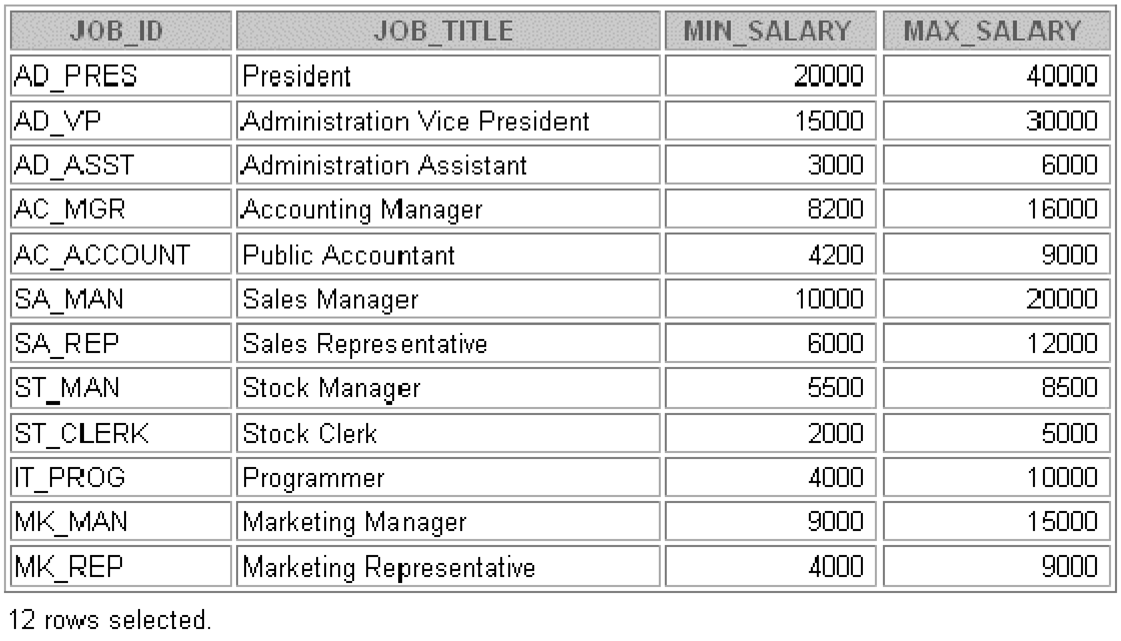
JOBS Table

```
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)

```
SELECT * FROM jobs;
```

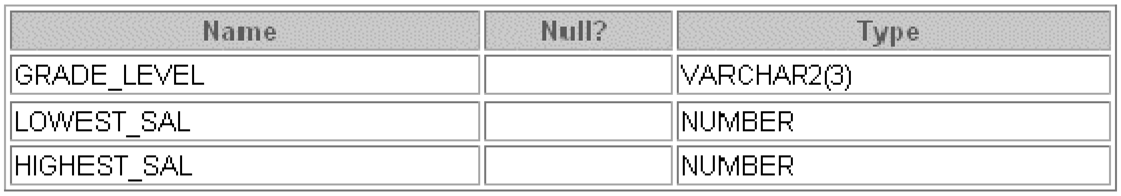


JOB_ID	JOB_TITLE	MIN_SALARY	MAX_SALARY
AD_PRES	President	20000	40000
AD_VP	Administration Vice President	15000	30000
AD_ASST	Administration Assistant	3000	6000
AC_MGR	Accounting Manager	8200	16000
AC_ACCOUNT	Public Accountant	4200	9000
SA_MAN	Sales Manager	10000	20000
SA_REP	Sales Representative	6000	12000
ST_MAN	Stock Manager	5500	8500
ST_CLERK	Stock Clerk	2000	5000
IT_PROG	Programmer	4000	10000
MK_MAN	Marketing Manager	9000	15000
MK_REP	Marketing Representative	4000	9000

12 rows selected.

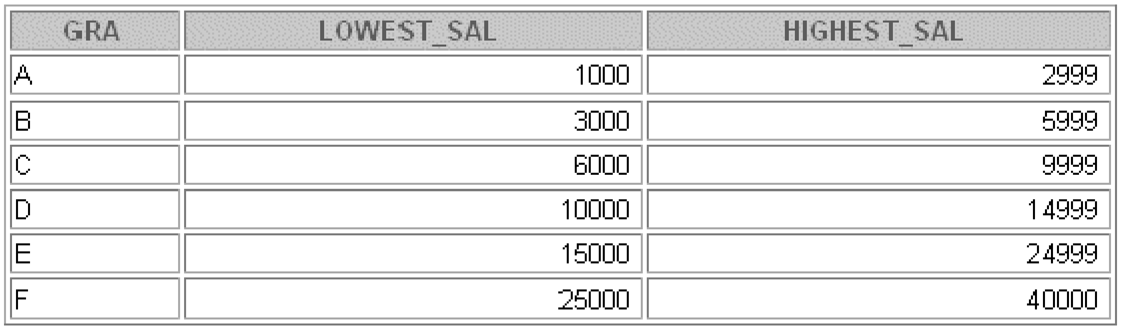
JOB_GRADES Table

```
DESCRIBE job_grades
```



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

```
SELECT * FROM job_grades;
```



GRA	LOWEST_SAL	HIGHEST_SAL
A	1000	2999
B	3000	5999
C	6000	9999
D	10000	14999
E	15000	24999
F	25000	40000

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)

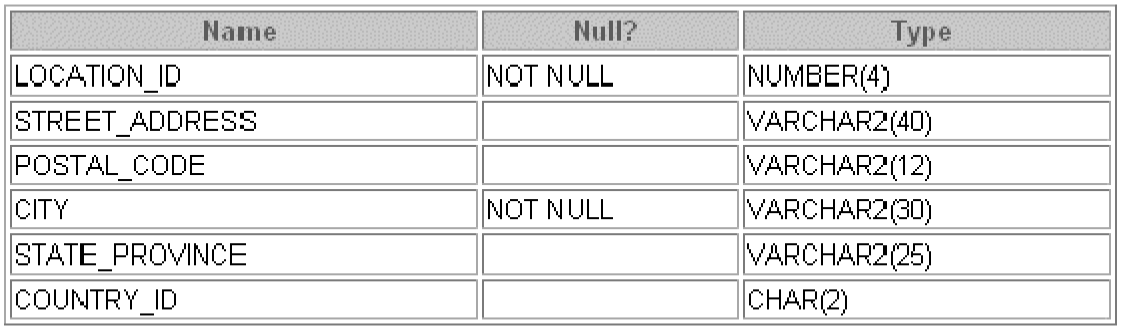
```
SELECT * FROM job_history;
```

EMPLOYEE_ID	START_DAT	END_DATE	JOB_ID	DEPARTMENT_ID
102	13-JAN-93	24-JUL-98	IT_PROG	60
101	21-SEP-89	27-OCT-93	AC_ACCOUNT	110
101	28-OCT-93	15-MAR-97	AC_MGR	110
201	17-FEB-96	19-DEC-99	MK_REP	20
114	24-MAR-98	31-DEC-99	ST_CLERK	50
122	01-JAN-99	31-DEC-99	ST_CLERK	50
200	17-SEP-87	17-JUN-93	AD_ASST	90
176	24-MAR-98	31-DEC-98	SA_REP	80
176	01-JAN-99	31-DEC-99	SA_MAN	80
200	01-JUL-94	31-DEC-98	AC_ACCOUNT	90

10 rows selected.

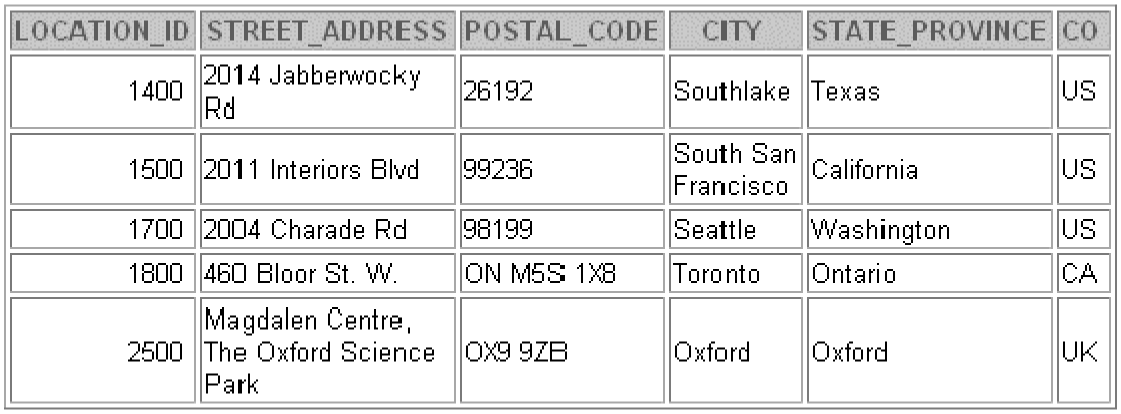
LOCATIONS Table

```
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)

```
SELECT * FROM locations;
```



LOCATION_ID	STREET_ADDRESS	POSTAL_CODE	CITY	STATE_PROVINCE	CO
1400	2014 Jabberwocky Rd	26192	Southlake	Texas	US
1500	2011 Interiors Blvd	99236	South San Francisco	California	US
1700	2004 Charade Rd	98199	Seattle	Washington	US
1800	460 Bloor St. W.	ON M5S 1X8	Toronto	Ontario	CA
2500	Magdalen Centre, The Oxford Science Park	OX9 9ZB	Oxford	Oxford	UK

REGIONS Table

```
DESCRIBE regions
```

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

```
SELECT * FROM regions;
```

REGION_ID	REGION_NAME
1	Europe
2	Americas
3	Asia
4	Middle East and Africa