USMAN INSTITUTE OF TECHNOLOGY

Department of Computer Science

CS311 Introduction to Database Systems

Lab#3

Objective:

- **Single-row and multiple-row functions in SQL**

Name of Student: Muhammad Waleed

Roll No: 20B-115-SE **Sec.** B

Date of Experiment:

....

Marks Obtained/Remarks: ______________________________

Signature: ______________________________

THEORY

SQL functions

Functions are a very powerful feature of SQL and can be used to do the following tasks:

- Perform calculations on data
- Modify individual data items
- Manipulate output for groups of rows
- Format dates and numbers for display
- Convert column datatypes

SQL functions may accept arguments and always return a value as illustrated in figure 4.1.

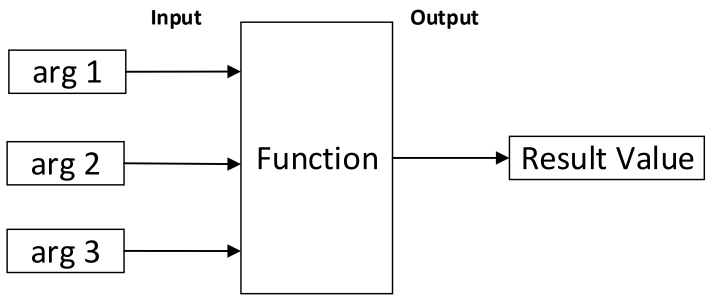


Figure 4.1: Functions accept arguments and return a value

Types of SQL functions

There are two distinct types of functions:

- Single-row
- Multiple-row

Single-Row functions

These functions operate on single rows only and return one result per row. There are different types of single-row functions. This session covers the following ones:

- Character
- Number
- Date
- Conversion

Multiple-Row functions

These functions manipulate groups of rows to give one result per group of rows.

Single-Row Functions

Character Functions

Single-row character functions accept character data as input and can return both character and number values. Character functions can be divided into the following: -

- Case conversion functions
- Character manipulation functions

Case Conversion Functions
Convert case for character strings

Table 3.1

Function	Result
LOWER(‘SQL Course’)	sql course
UPPER(‘SQL Course’)	SQL COURSE
INITCAP(‘SQL Course’)	Sql Course

Examples

i. To print an employee name (first letter capital) and job title (lower case)

```
SELECT ‘The job title for ‘ || INITCAP(ename) || ‘ is ‘ || LOWER(job) AS “EMPLOYEE
DETAILS” FROM emp;
```

ii. To display the employee number, name (in upper case) and department number for employee *Blake*.

```
SELECT empno, UPPER(ename), deptno
FROM emp
WHERE LOWER(ename) = ‘blake’;
```

Note: Since the actual case of the letters in the employee name column may not be known, so it is necessary for comparison to convert the name to either uppercase or lowercase.

Character manipulation functions
Manipulate character strings

Table 3.2

Function	Result
CONCAT(‘Good’, ‘String’)	GoodString
SUBSTR(‘String’, 2, 4)	trin
LENGTH(‘String’)	6
INSTR(‘String’, ‘r’)	3
LPAD(sal, 10, ‘*’)	******5000

Example

To display employee name and job joined together, length of employee name, and the numeric position of letter A in the employee name, for all employees who are in *sales*.

```
SELECT empno, CONCAT(ename, job), LENGTH(ename), INSTR(ename, ‘A’)
FROM emp
WHERE SUBSTR(job, 1, 5) = ‘SALES’;
```

Number Functions

Number functions accept numeric input and return numeric values.

ROUND(column|expression, n) Rounds the column, expression or value to **n** decimal places or if **n** is omitted, no decimal places (If n is negative, numbers to left of decimal point are rounded)

TRUNC(column|expression, n) Truncates the column, expression or value to *n* decimal places or if *n* is omitted, no decimal places (If *n* is negative, numbers to left of decimal point are truncated)

MOD(m, n) Returns the remainder of *m* divided by *n*

Table 3.3

Function	Result
ROUND(45.927, 2)	45.93
ROUND(45.927)	46
ROUND(45.927, -1)	50
TRUNC(45.927, 2)	45.92
TRUNC(45.927)	45
MOD(20, 3)	2

Examples

i. SELECT ROUND(45.923, 2), ROUND(45.923, 0), ROUND(45.923, -1)
FROM DUAL;

The DUAL is a dummy table with one column and one row.

ii. SELECT TRUNC(45.923, 2), TRUNC(45.923), TRUNC(45.923, -1)
FROM DUAL;

iii. To calculate the remainder of the ratio of salary to commission for all employees whose job title is salesman.

```
SELECT ename, sal, comm., MOD(sal, comm.)
FROM emp
WHERE UPPER(job) = ‘SALESMAN’;
```

Date Functions

SYSDATE is a date function that returns the current date and time. The current date can be displayed by selecting SYSDATE from a table. It is customary to select SYSDATE from a dummy table called DUAL. The DUAL table is owned by the user SYS and can be accessed by all users. It contains one column, DUMMY, and one row with the value X. It is useful for returning a value once only – for instance, the value of a constant, pseudo column, or expression that is not derived from a table with user data.

For Example: To display the current date using the DUAL table as

```
SELECT SYSDATE
FROM DUAL;
```

Arithmetic with Dates

We can add or subtract a number to or from a date for a resultant date value. For example, to display the name and the number of weeks employed for all employees in department 10.

```
SELECT ename, (SYSDATE – HIREDATE) / 7 “Number of Weeks”
FROM emp
WHERE deptno = 10;
```

Date functions operate on Oracle dates. All date functions return a value of DATE datatype except MONTHS_BETWEEN, which returns a numeric value.

Table 3.4

Function	Result	Description
MONTHS_BETWEEN(’01-SEP-95’, ’11-JAN-94’)	19.6774194	Number of months between two dates
ADD_MONTHS(’11-JAN-94’, 6)	’11-JUL-94’	Add calendar months to dates
NEXT_DAY(’01-SEP-95’, ‘FRIDAY’)	’08-SEP-95’	Next day of the date specified
LAST_DAY(’01-SEP-95’)	’30-SEP-95’	Last day of the month
ROUND(TO_DATE(’25-JUL-95’, ‘DD-MON-YY’), ‘MONTH’)	01-AUG-95	Round date
ROUND(TO_DATE(’25-JUL-95’, ‘DD-MON-YY’), ‘YEAR’)	01-JAN-96	Round date
TRUNC(TO_DATE(’25-JUL-95’, ‘DD-MON-YY’), ‘MONTH’)	01-JUL-95	Truncate date
TRUNC(TO_DATE(’25-JUL-95’, ‘DD-MON-YY’), ‘YEAR’)	01-JAN-95	Truncate date

Examples

i. For all employees employed for fewer than 200 months, display the employee number, hiredate, number of months employed, six-month review date, first Friday after hiredate and last day of the month hired.

```
SELECT empno, hiredate, MONTHS_BETWEEN(SYSDATE, hiredate) TENURE,
ADD_MONTHS(hiredate, 6) REVIEW, NEXT_DAY(hiredate, ‘FRIDAY’),
LAST_DAY(hiredate) FROM emp
WHERE MONTHS_BETWEEN(SYSDATE, hiredate) < 200;
```

ii. Comparing the hire dates for all employees who started in 1982, display the employee number, hiredate, and month started using the ROUND and TRUNC functions.

```
SELECT empno, hiredate, ROUND(hiredate, ‘MONTH’),TRUNC(hiredate, ‘MONTH’)
FROM emp
WHERE hiredate like ‘%82’;
```

Conversion Functions

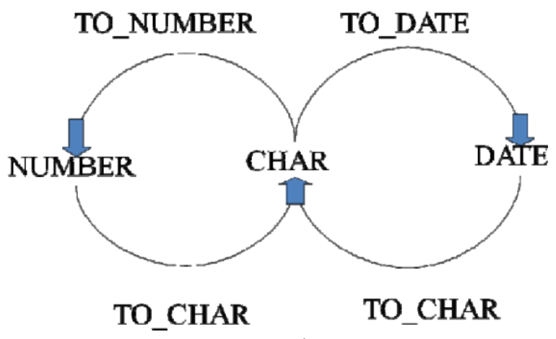


Figure 4.1

SQL provides three functions to convert a value from one data type to another.

```
TO_CHAR
TO_NUMBER
TO_DATE
```

TO_CHAR function with Dates

i. To display the employee number, the month number and year of hiring

```
SELECT empno, TO_CHAR(hiredate, ‘MM/YY’) Month_Hired
FROM emp
WHERE ename = ‘BLAKE’;
```

The second argument of TO_CHAR is called *format model*, is in single quotation marks and is case sensitive.

ii. To display the employee name and hiredate for all employees. The hiredate appears as 17 November, 1981.

```
SELECT ename, TO_CHAR(hiredate, 'fmDD Month YYYY') HIREDATE
FROM emp;
```

The *fm* element is used to remove padded blanks or suppress leading zeros.

iii. To print the employee name and time of joining in format HH:MI:SS (Assuming that hiredate column were used for storing joining time)

```
SELECT ename, TO_CHAR(hiredate, ‘HH:MI:SS’) HIREDATE FROM emp;
```

TO_CHAR function with Numbers

It is used to display a number value as a character string. This technique is especially useful for concatenating a numeric value to a character string.

i. To display the salary of employee SCOTT with $ sign preceded

```
SELECT TO_CHAR(sal, ‘$99,999’) SALARY
FROM emp
WHERE ename = ‘SCOTT’;
```

The oracle server displays a string of pound signs (#) in place of a whole number whose digits exceed the number of digits provided in the format model. The oracle server rounds the stored decimal value to the number of decimal places provided in the format model.

TO_NUMBER function

Convert a character string to a number format using the TO_NUMBER function.
TO_DATE function converts a character string to a date format.
i. To display the names and hire dates of all the employees who joined on February 22, 1981

```
SELECT ename, hiredate
FROM emp
WHERE hiredate = TO_DATE('February 22, 1981', 'Month dd, YYYY');
```

Nesting Functions

Single-row functions can be nested to any level. For example, the following query displays the head of a company who has no manager.

```
SELECT ENAME, NVL(TO_CHAR(MGR), 'No Manager')
FROM EMP
WHERE MGR IS NULL;
```

Multiple_Row Functions

Group functions operate on sets of rows to give one result per group. The following table identifies the options that can be used in the syntax. Following is the syntax of using group functions: -

```
SELECT      [column, ] group_function(column)
FROM  table
[WHERE condition]
[GROUP BY column]
[ORDER BY column];
```

Applying multiple-row functions to all rows in a table

Examples

i. To show the average salary, minimum salary, maximum salary and count of employees in the organization

```
SELECT AVG (SAL), MIN(SAL), MAX(SAL), COUNT(*)
FROM EMP;
```

ii. To show the minimum and maximum hiredate for employees

```
SELECT MIN (hiredate), MAX(hiredate)
FROM emp;
```

iii. To return the number of rows in a table

```
SELECT COUNT(*)
FROM emp
WHERE deptno = 30;
```

iv. To return the number of nonnull rows in a table

```
SELECT COUNT(comm)
FROM emp
WHERE deptno = 30;
```

v. The group function like AVG do not include null rows. The NVL function forces group functions to include null values.

```
SELECT AVG(NVL(comm, 0))
FROM emp;
```

Applying Multiple-row functions to groups of rows in a table

Examples

i. To show the department-wise average salary,

```
SELECT deptno, AVG(sal) AVERAGE_SALARY
FROM emp
GROUP BY deptno;
```

Note that all columns in the SELECT list that are not in group functions must be in the GROUP BY clause.

ii. To show the job-wise total salary for each department

```
SELECT deptno, job, sum(sal)
FROM emp
GROUP BY deptno, job;
```

Excluding groups result

In the same way that we use the WHERE clause to restrict the rows that we select, the HAVING clause is used to restrict groups. First the group function is applied and the groups matching the HAVING clause are displayed. The syntax of the SELECT statement showing the HAVING clause along with the GROUP BY clause is shown below:-

```
SELECT  column, group_function
FROM  table
[WHERE  condition]
[GROUP BY  group_by_expression]
[HAVING group_condition]
[ORDER BY column];
```

The HAVING clause can precede the GROUP BY clause but it is recommended that the GROUP BY clause come first because it is more logical.

Examples

i. To show the department-wise average and maximum salary, in the descending order of average salary, for all departments having average salary higher than 4500.

```
SELECT DEPTNO, AVG(SAL), MAX(SAL)
FROM EMP
GROUP BY DEPTNO
HAVING AVG(SAL) > 2000
ORDER BY AVG(SAL)
```

ii. To display the job title and total monthly salary for each job title with a total payroll exceeding 5000.

```
SELECT JOB, SUM(SAL) PAYROLL
FROM EMP
WHERE JOB NOT LIKE 'SALES%'
GROUP BY JOB
HAVING SUM(SAL) > 5000
ORDER BY SUM(SAL);
```

Nesting Group Functions

i. To display the maximum average salary by nesting group functions

```
SELECT max(avg(sal))
FROM emp
GROUP BY deptno;
```

Lab No. 3

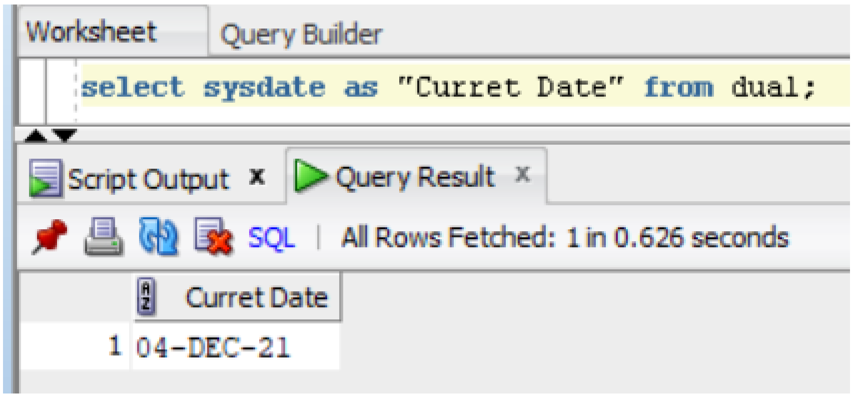
EXERCISES

1. Write down the result of the function calls in the first column to the second column.

Function Call	Result
SUBSTR(CONCAT('HIGH', 'SALARY'), 4, 6)	HSALAR
CONCAT(SUBSTR('INFORMATION', 3, 4), 'TECH')	FORMTECH
INSTR(CONCAT('GET', 'ING'), 'TIN')	3
ROUND(69.476, 1)	69.5
TRUNC('13-MAR-90', 'MONTH')	Invalid because truck takes number
TRUNC('13-MAR-90', 'YEAR')	Invalid because truck takes number
MOD(90, LENGTH('SEVENTY'))	6
MONTHS_BETWEEN('14-AUG-96', '23-MAR-95')	16.709677

2. Write down SQL queries to perform following functions:-

i. To show the current date and label the column *Current Date*.

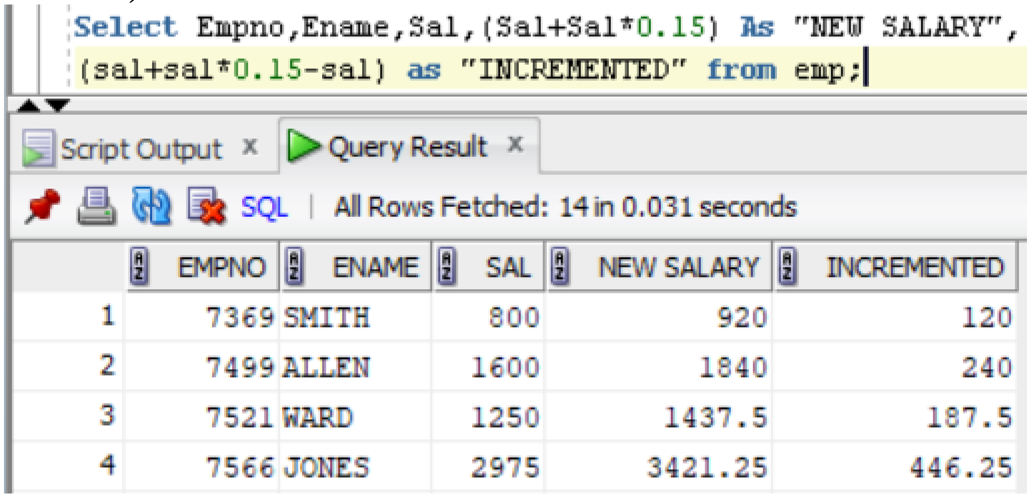


```
select sysdate as "Curret Date" from dual;
```

```
All Rows Fetched: 1 in 0.626 seconds

    Curret Date
  1 04-DEC-21
```

ii. To display the employee number, name, salary, salary increase by 15% expressed as a whole number (labeled as *New Salary*), the difference between old salary and new salary (labeled as *Increment*).

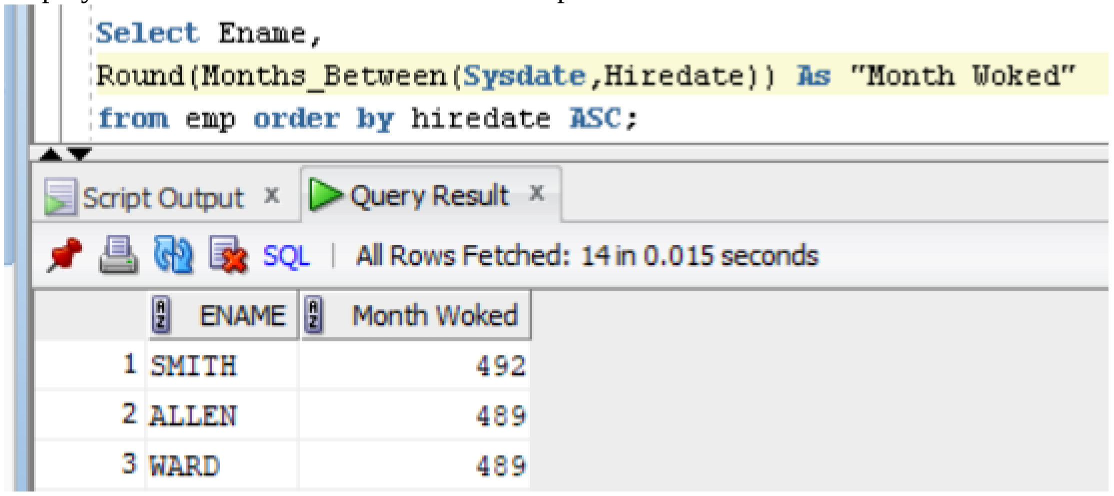


```
Select Empno,Ename,Sal,(Sal+Sal*0.15) As "NEW SALARY",
(sal+sal*0.15-sal) as "INCREMENTED" from emp;
```

```
All Rows Fetched: 14 in 0.031 seconds

    EMPNO ENAME   SAL  NEW SALARY  INCREMENTED
  1  7369 SMITH   800         920          120
  2  7499 ALLEN  1600        1840          240
  3  7521 WARD   1250      1437.5        187.5
  4  7566 JONES  2975     3421.25       446.25
```

iii. To display the employee name and calculate the number of months between today and the date the employee was hired (Labeled as *Months_Worked*). Order the results by the number of months employed and round the number of months up to the closest whole number.



```
Select Ename,
Round(Months_Between(Sysdate,Hiredate)) As "Month Woked"
from emp order by hiredate ASC;
```

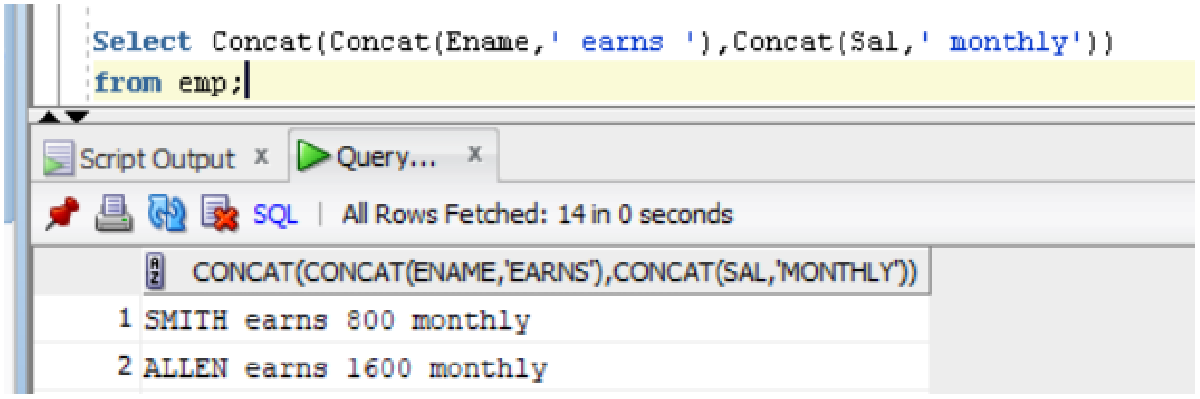
Script Output × | Query Result ×

SQL | All Rows Fetched: 14 in 0.015 seconds

```
     ENAME   Month Woked
   1 SMITH           492
   2 ALLEN           489
   3 WARD            489
```

iv. Write a query that produces the following for each employee:

<*employee name*> earns <*salary*> monthly



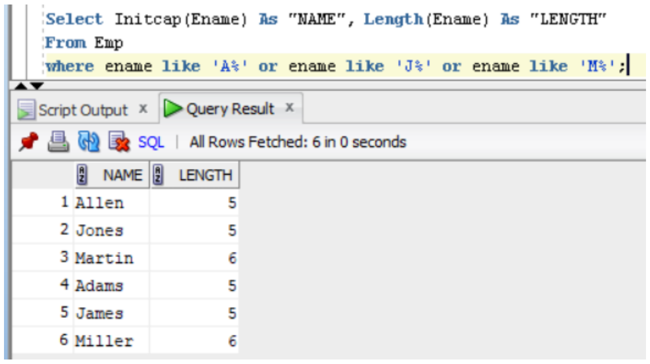
```
Select Concat(Concat(Ename,' earns '),Concat(Sal,' monthly'))
from emp;
```

Script Output × | Query... ×

SQL | All Rows Fetched: 14 in 0 seconds

```
     CONCAT(CONCAT(ENAME,'EARNS'),CONCAT(SAL,'MONTHLY'))
   1 SMITH earns 800 monthly
   2 ALLEN earns 1600 monthly
```

v. To display the employee's name (labeled *name*) with the first letter capitalized and all other letters lowercase and the length of their name (labeled *length*), for all employees whose name starts with J, A or M.



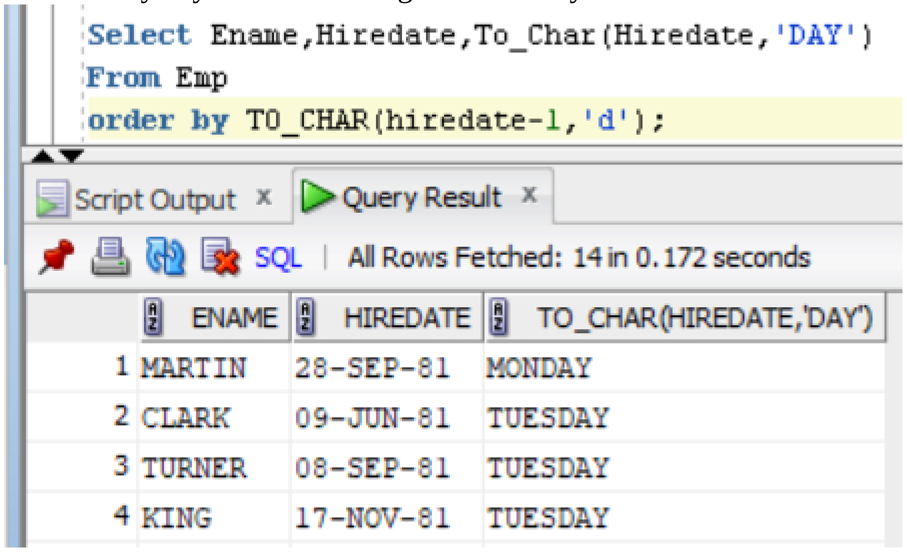
```
Select Initcap(Ename) As "NAME", Length(Ename) As "LENGTH"
From Emp
where ename like 'A%' or ename like 'J%' or ename like 'M%';
```

Script Output × | Query Result ×

SQL | All Rows Fetched: 6 in 0 seconds

```
     NAME    LENGTH
   1 Allen        5
   2 Jones        5
   3 Martin       6
   4 Adams        5
   5 James        5
   6 Miller       6
```

vi. To list the name, hiredate, and day of the week (labeled *DAY*) on which job was started. Order the result by day of week starting with Monday.



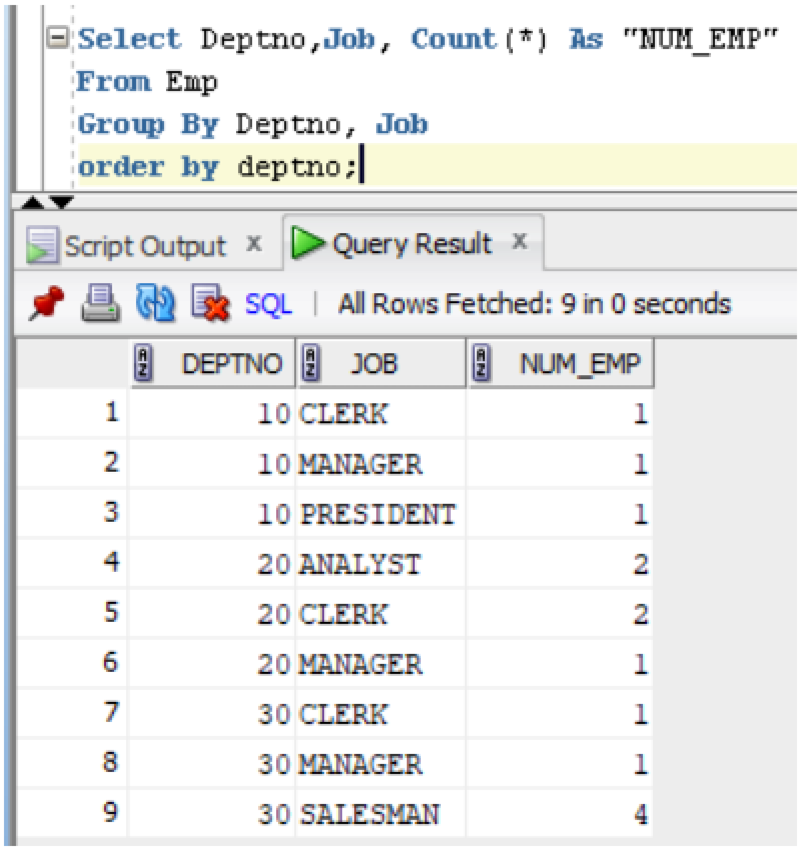
```
Select Ename,Hiredate,To_Char(Hiredate,'DAY')
From Emp
order by TO_CHAR(hiredate-1,'d');
```

All Rows Fetched: 14 in 0.172 seconds

	ENAME	HIREDATE	TO_CHAR(HIREDATE,'DAY')
1	MARTIN	28-SEP-81	MONDAY
2	CLARK	09-JUN-81	TUESDAY
3	TURNER	08-SEP-81	TUESDAY
4	KING	17-NOV-81	TUESDAY

vii. To display the job-wise count of employees in each department as follows:-

DEPTNO JOB NUM_EMP

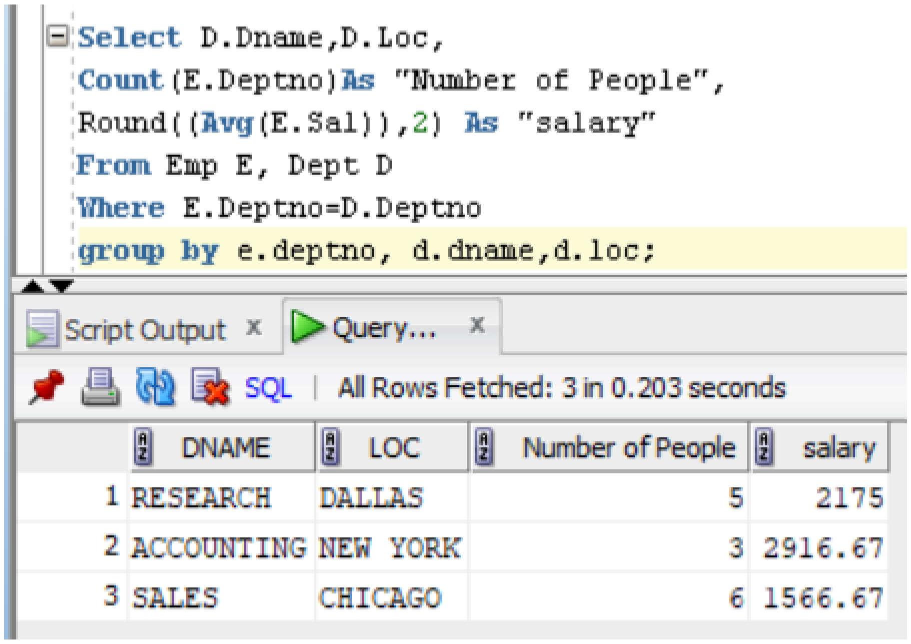


```
Select Deptno,Job, Count(*) As "NUM_EMP"
From Emp
Group By Deptno, Job
order by deptno;
```

All Rows Fetched: 9 in 0 seconds

	DEPTNO	JOB	NUM_EMP
1	10	CLERK	1
2	10	MANAGER	1
3	10	PRESIDENT	1
4	20	ANALYST	2
5	20	CLERK	2
6	20	MANAGER	1
7	30	CLERK	1
8	30	MANAGER	1
9	30	SALESMAN	4

viii. To display the department name, location name, number of employees and the average salary for all employees in that department. Label the columns DNAME, LOC, NUMBER OF PEOPLE and SALARY, respectively. Round the average salary to two decimal places.

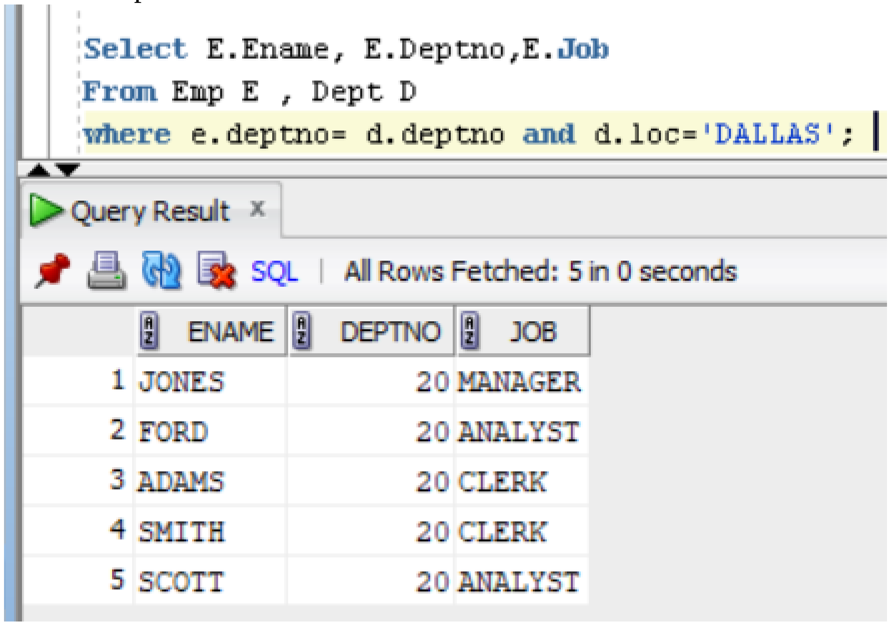


```
Select D.Dname,D.Loc,
Count(E.Deptno)As "Number of People",
Round((Avg(E.Sal)),2) As "salary"
From Emp E, Dept D
Where E.Deptno=D.Deptno
group by e.deptno, d.dname,d.loc;
```

All Rows Fetched: 3 in 0.203 seconds

```
  DNAME      LOC       Number of People  salary
1 RESEARCH   DALLAS                   5     2175
2 ACCOUNTING NEW YORK                 3  2916.67
3 SALES      CHICAGO                  6  1566.67
```

ix. To display the employee name, department number and job title for all employees whose department location is *Dallas*.

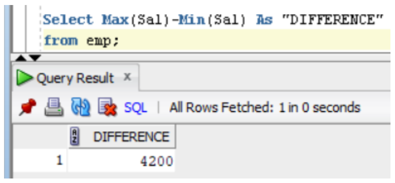


```
Select E.Ename, E.Deptno,E.Job
From Emp E , Dept D
where e.deptno= d.deptno and d.loc='DALLAS';
```

All Rows Fetched: 5 in 0 seconds

```
  ENAME DEPTNO JOB
1 JONES     20 MANAGER
2 FORD      20 ANALYST
3 ADAMS     20 CLERK
4 SMITH     20 CLERK
5 SCOTT     20 ANALYST
```

x. To display the difference between the highest and lowest salaries (Labeled as *DIFFERENCE*)

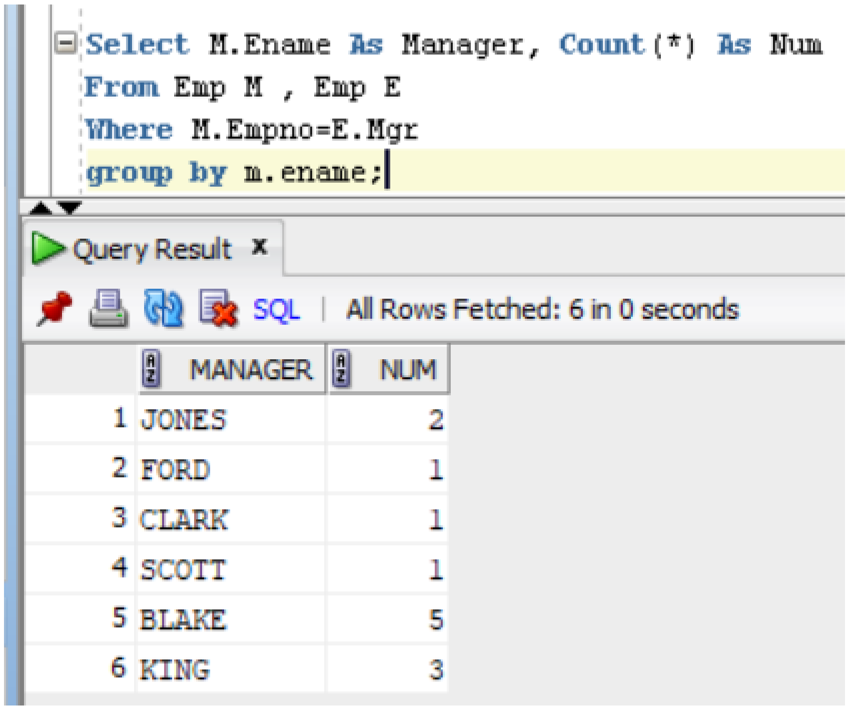


```
Select Max(Sal)-Min(Sal) As "DIFFERENCE"
from emp;
```

All Rows Fetched: 1 in 0 seconds

```
  DIFFERENCE
1       4200
```

xi. To show the manager name, MANAGER, and the number of employees, NUM, working under him.



```
Select M.Ename As Manager, Count(*) As Num
From Emp M , Emp E
Where M.Empno=E.Mgr
group by m.ename;
```

Query Result

SQL | All Rows Fetched: 6 in 0 seconds

	MANAGER	NUM
1	JONES	2
2	FORD	1
3	CLARK	1
4	SCOTT	1
5	BLAKE	5
6	KING	3