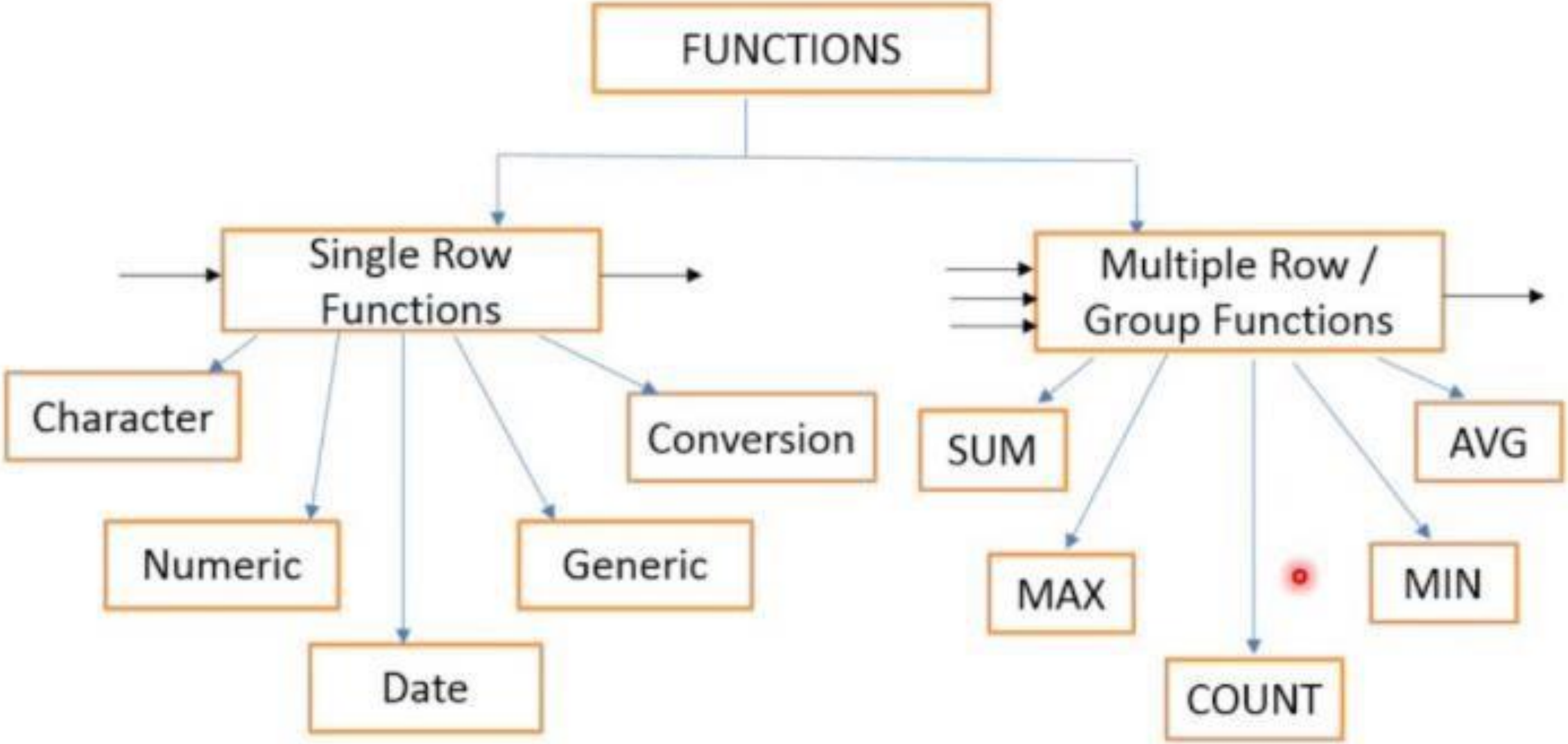
DATABASE SYSTEMS

SQL FUNCTIONS

By Sana Faiz
Sana.faiz.muet83@gmail.com

SQL FUNCTIONS

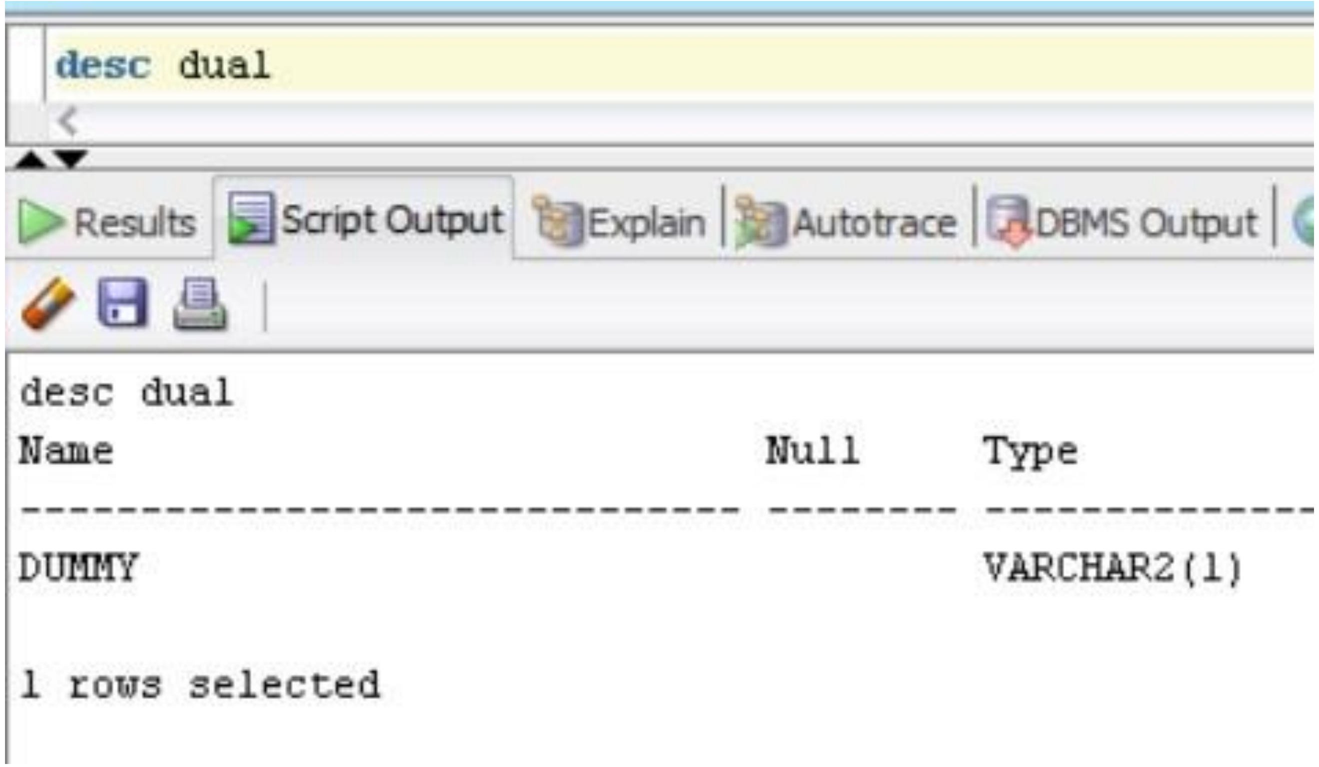


SCALAR FUNCTIONS

- **Scalar Functions** allow you to perform different calculations on data values. These functions operate on single rows only and produce one result per row.
- These are also known as **Single Row Functions**.
- Scalar functions include the following:

1. String / Character Functions – functions that perform operations on character values.
2. Numeric Functions – functions that perform operations on numeric values.
3. Date Functions – functions that perform operations on date values.
4. Conversion Functions – functions that convert data types.
5. NULL-related / Generic Functions – functions for handling null values.

THE DUMMY TABLE

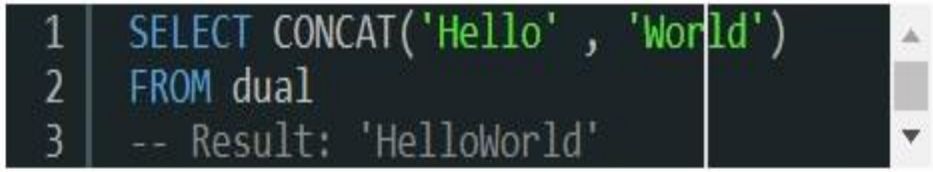


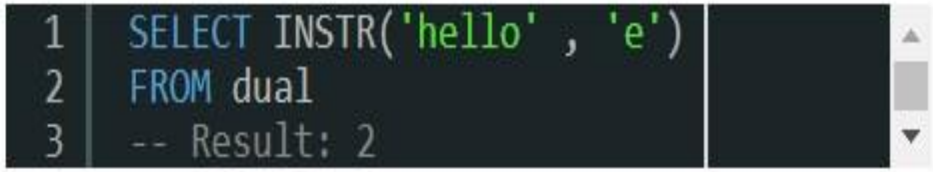
```
desc dual
```

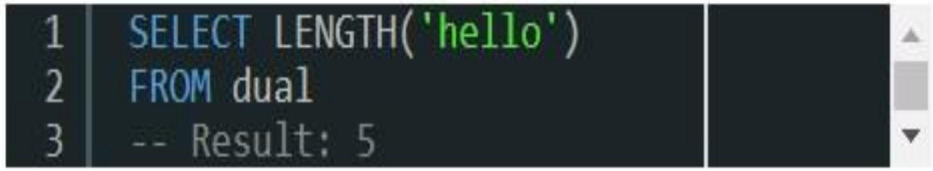
```
desc dual
Name                           Null     Type
------------------------------ -------- -------------
DUMMY                                   VARCHAR2(1)


1 rows selected
```

STRING FUNCTIONS

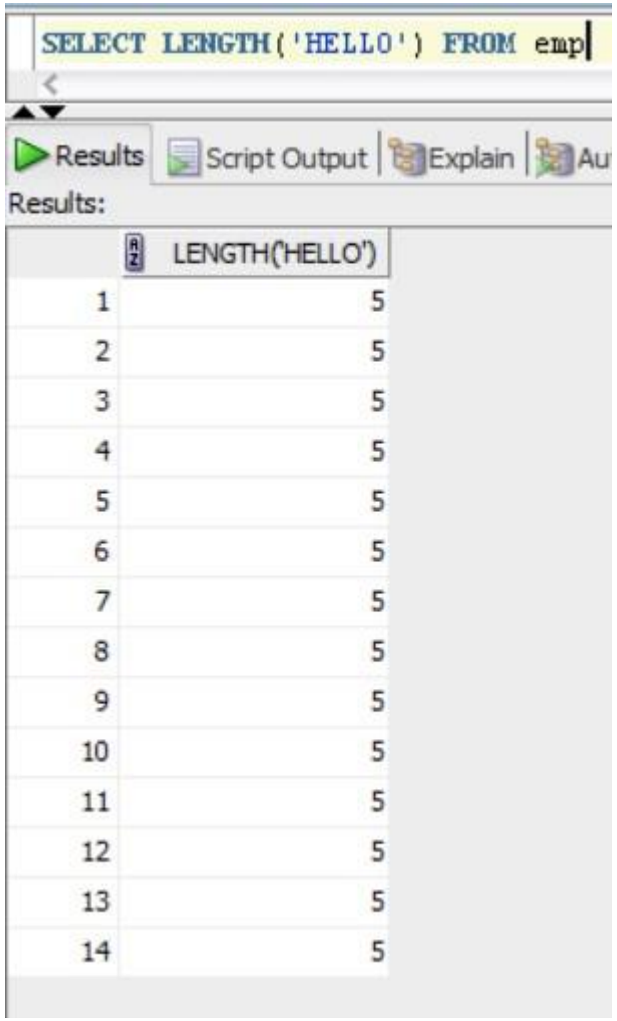
Function	Description	Syntax

CONCAT	Returns text strings concatenated	 `SELECT CONCAT('Hello' , 'World')`
		`FROM dual`
		`-- Result: 'HelloWorld'`

INSTR	Returns the location of a substring in a string	 `SELECT INSTR('hello' , 'e')`
		`FROM dual`
		`-- Result: 2`

LENGTH	Returns the number of characters of the specified .string expression	 `SELECT LENGTH('hello')`
		`FROM dual`
		`-- Result: 5`

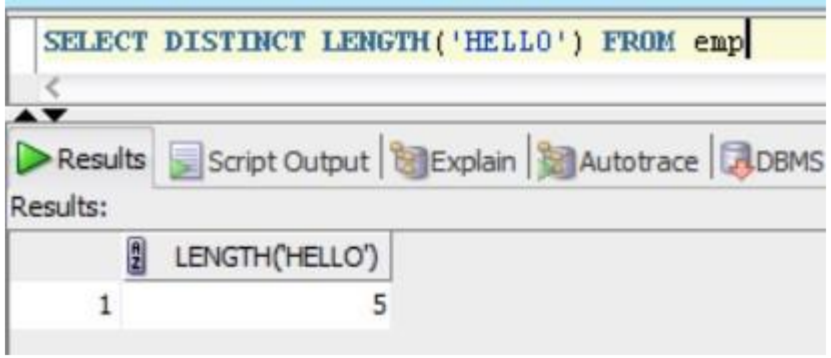
LENGTH



```
SELECT LENGTH('HELLO') FROM emp
```

Results:

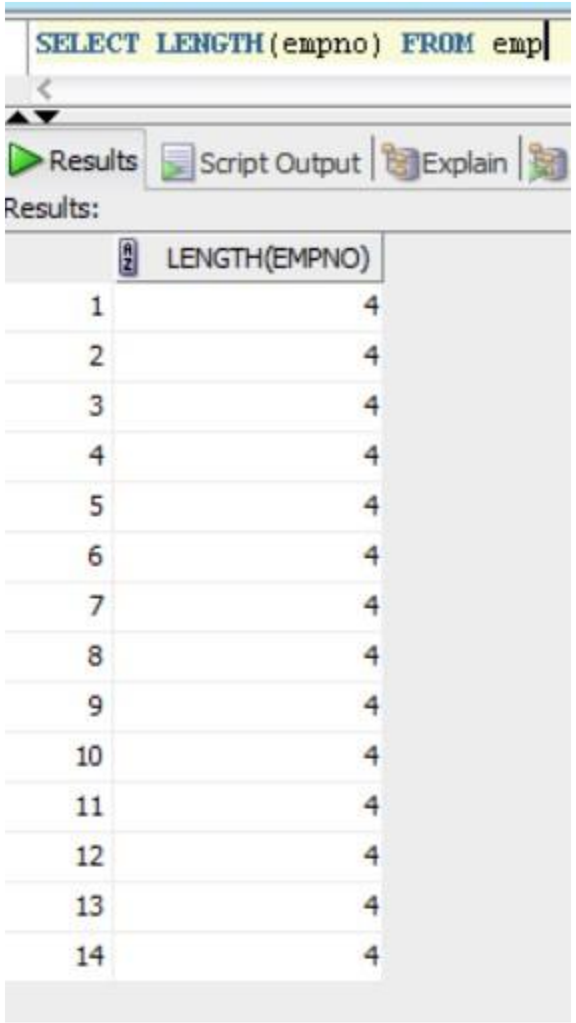
	LENGTH('HELLO')
1	5
2	5
3	5
4	5
5	5
6	5
7	5
8	5
9	5
10	5
11	5
12	5
13	5
14	5



```
SELECT DISTINCT LENGTH('HELLO') FROM emp
```

Results:

	LENGTH('HELLO')
1	5

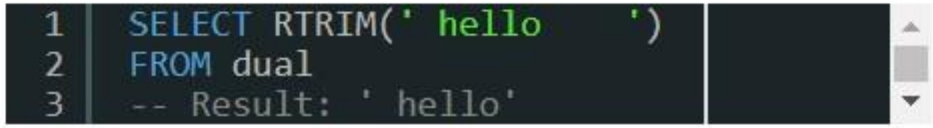


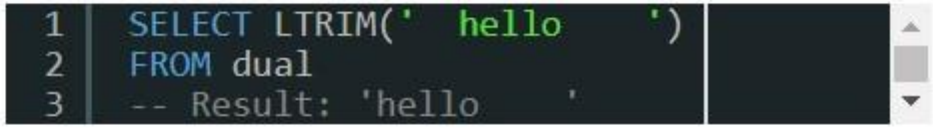
```
SELECT LENGTH(empno) FROM emp
```

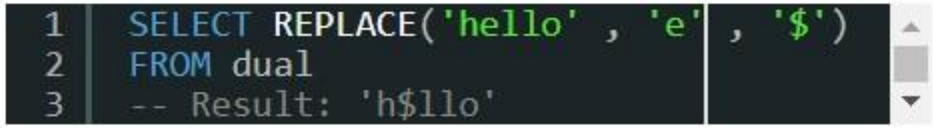
Results:

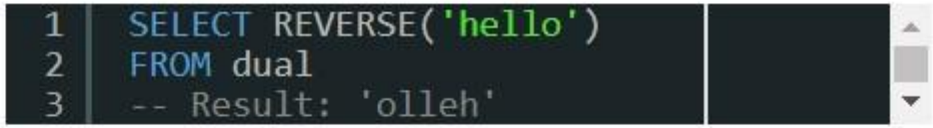
	LENGTH(EMPNO)
1	4
2	4
3	4
4	4
5	4
6	4
7	4
8	4
9	4
10	4
11	4
12	4
13	4
14	4

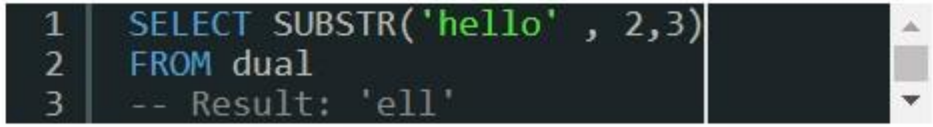
Function	Description	Syntax

RTRIM	Returns a character string after truncating all trailing blanks	 `SELECT RTRIM(' hello    ')`
		`FROM dual`
		`-- Result: ' hello'`

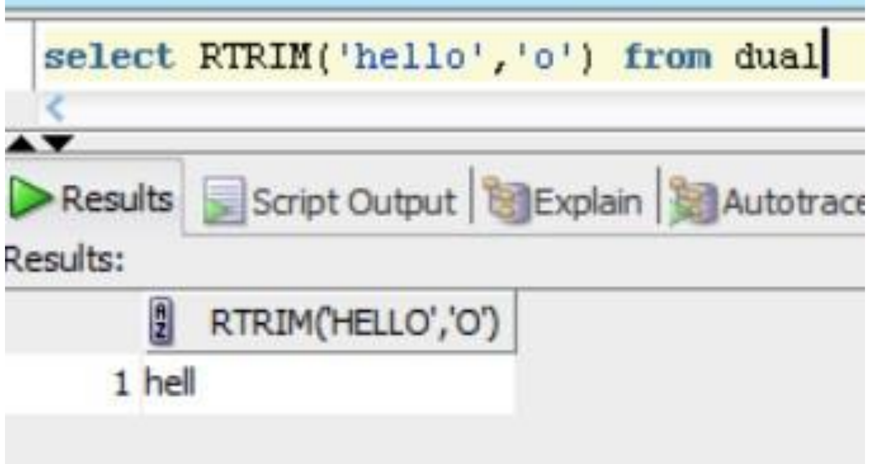
LTRIM	Returns a character expression after it removes leading blanks	 `SELECT LTRIM('  hello    ')`
		`FROM dual`
		`-- Result: 'hello    '`

REPLACE	Replaces all occurrences of a specified string value with another string value	 `SELECT REPLACE('hello' , 'e' , '$')`
		`FROM dual`
		`-- Result: 'h$llo'`

REVERSE	Returns the reverse order of a string value	 `SELECT REVERSE('hello')`
		`FROM dual`
		`-- Result: 'olleh'`

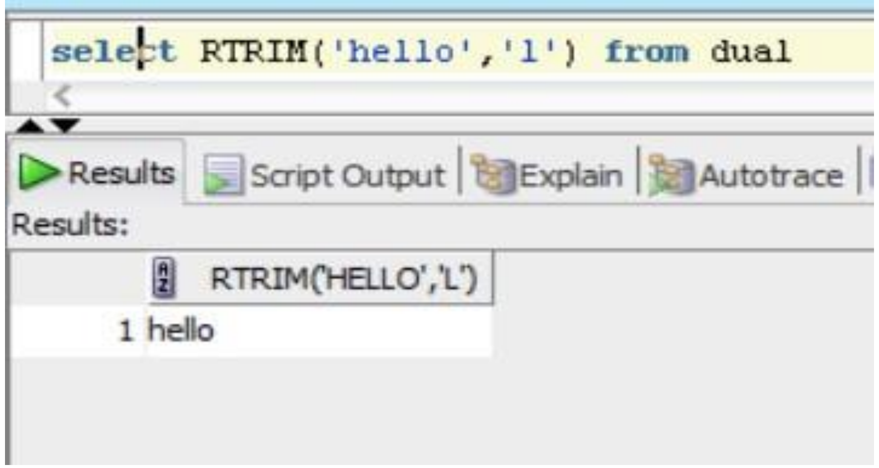
SUBSTR	Returns part of a text	 `SELECT SUBSTR('hello' , 2,3)`
		`FROM dual`
		`-- Result: 'ell'`

RTRIM / LTRIM



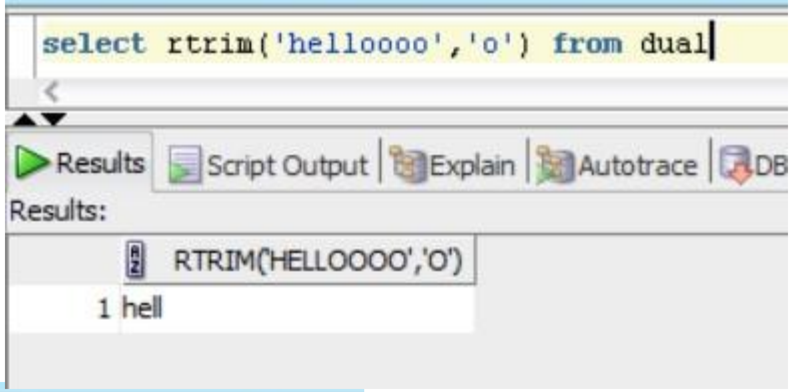
```
select RTRIM('hello','o') from dual
```

```
Results:
     RTRIM('HELLO','O')
   1 hell
```



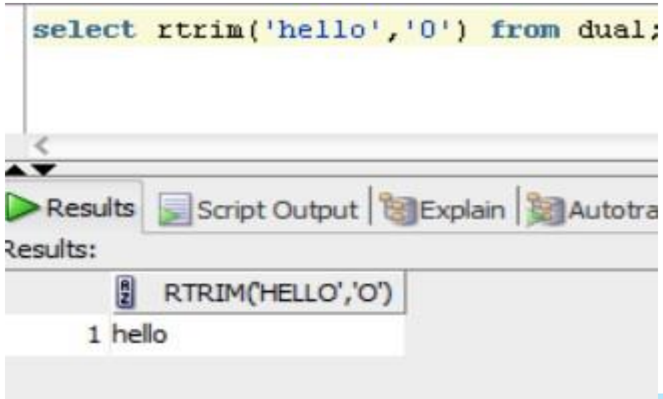
```
select RTRIM('hello','l') from dual
```

```
Results:
     RTRIM('HELLO','L')
   1 hello
```



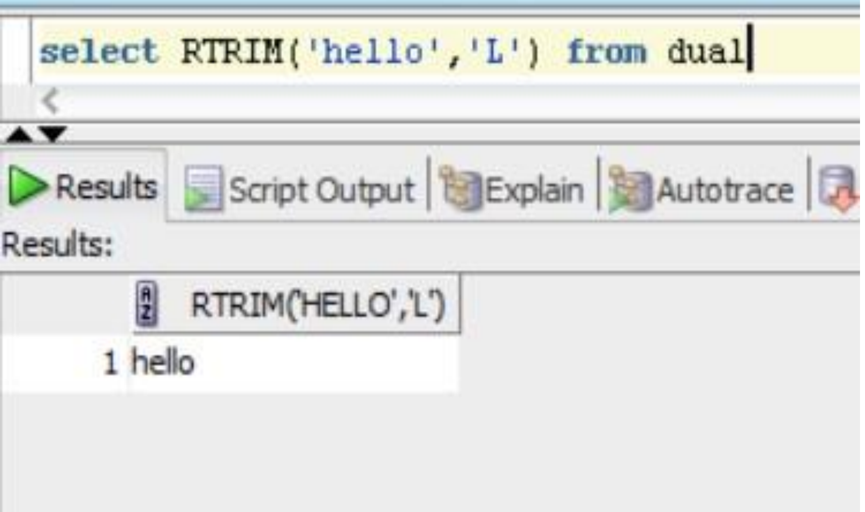
```
select rtrim('helloooo','o') from dual
```

```
Results:
     RTRIM('HELLOOOO','O')
   1 hell
```



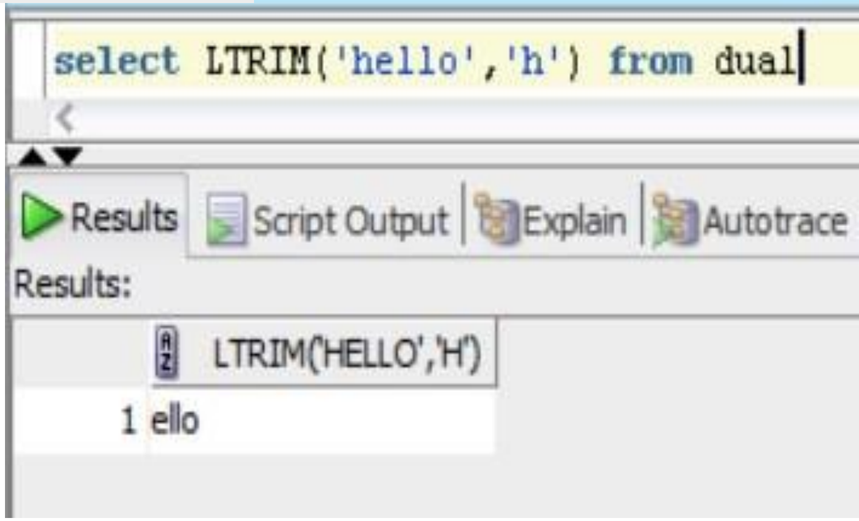
```
select rtrim('hello','O') from dual;
```

```
Results:
     RTRIM('HELLO','O')
   1 hello
```



```
select RTRIM('hello','L') from dual
```

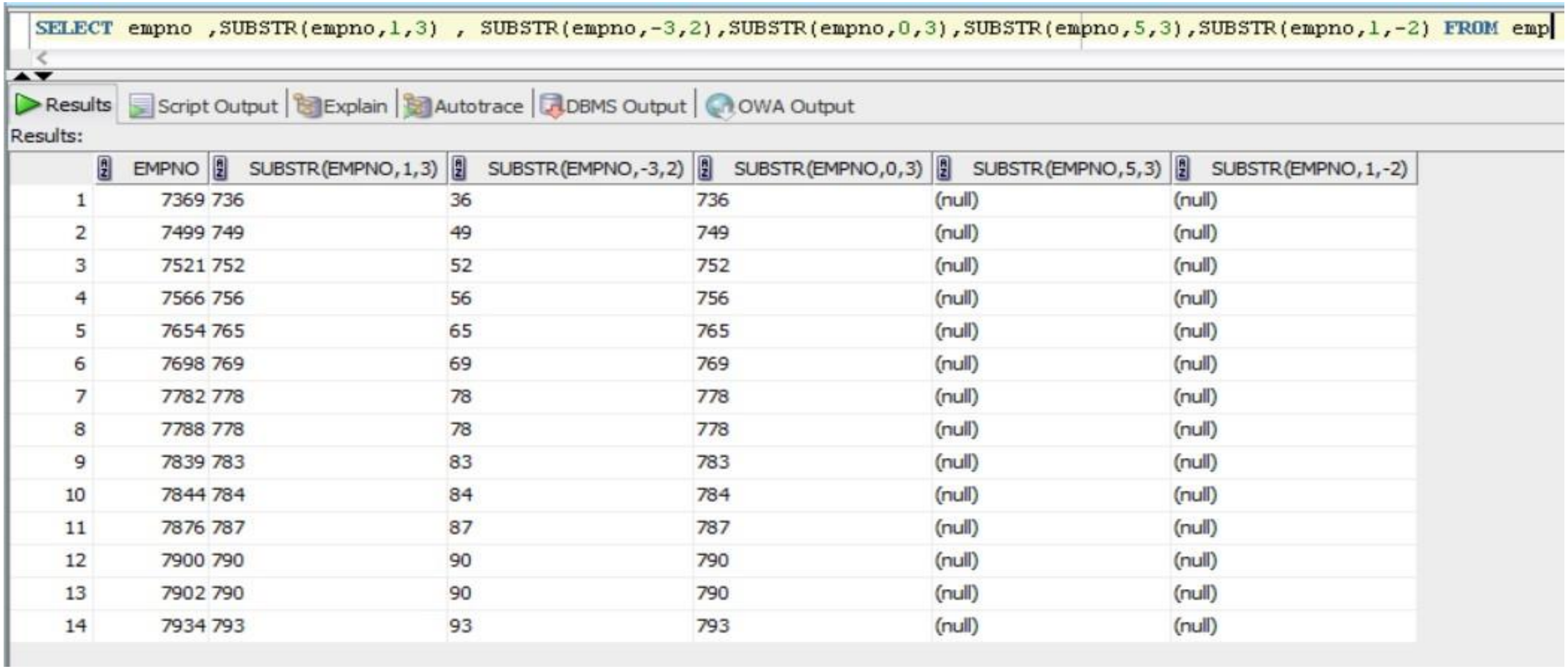
```
Results:
     RTRIM('HELLO','L')
   1 hello
```



```
select LTRIM('hello','h') from dual
```

```
Results:
     LTRIM('HELLO','H')
   1 ello
```

SUBSTR(c,p,l)

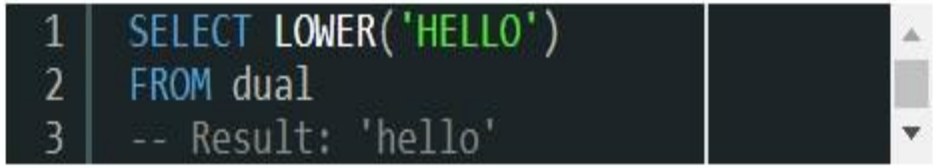


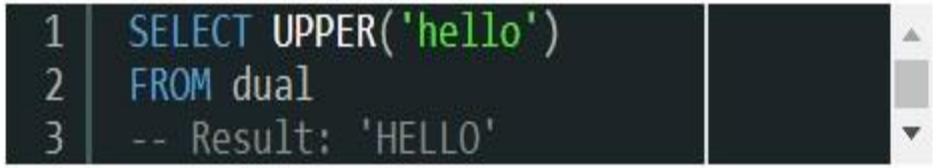
```
SELECT empno ,SUBSTR(empno,1,3) , SUBSTR(empno,-3,2),SUBSTR(empno,0,3),SUBSTR(empno,5,3),SUBSTR(empno,1,-2) FROM emp
```

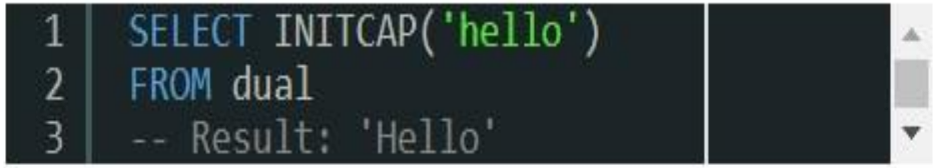
Results:

	EMPNO	SUBSTR(EMPNO,1,3)	SUBSTR(EMPNO,-3,2)	SUBSTR(EMPNO,0,3)	SUBSTR(EMPNO,5,3)	SUBSTR(EMPNO,1,-2)
1	7369	736	36	736	(null)	(null)
2	7499	749	49	749	(null)	(null)
3	7521	752	52	752	(null)	(null)
4	7566	756	56	756	(null)	(null)
5	7654	765	65	765	(null)	(null)
6	7698	769	69	769	(null)	(null)
7	7782	778	78	778	(null)	(null)
8	7788	778	78	778	(null)	(null)
9	7839	783	83	783	(null)	(null)
10	7844	784	84	784	(null)	(null)
11	7876	787	87	787	(null)	(null)
12	7900	790	90	790	(null)	(null)
13	7902	790	90	790	(null)	(null)
14	7934	793	93	793	(null)	(null)

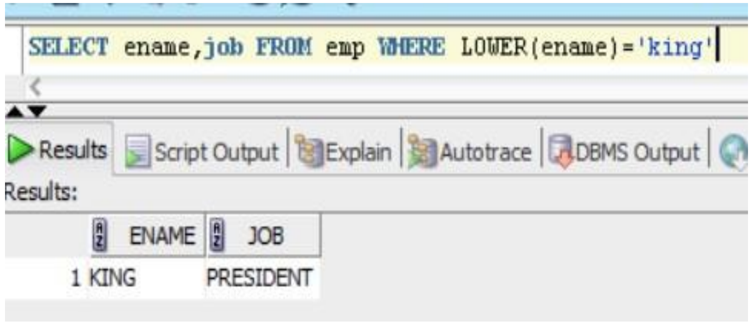
Function	Description	Syntax

LOWER	Returns a character expression after converting uppercase character data to lowercase	
		SELECT LOWER('HELLO')
		FROM dual
		-- Result: 'hello'

UPPER	Returns a character expression with lowercase character data converted to uppercase	
		SELECT UPPER('hello')
		FROM dual
		-- Result: 'HELLO'

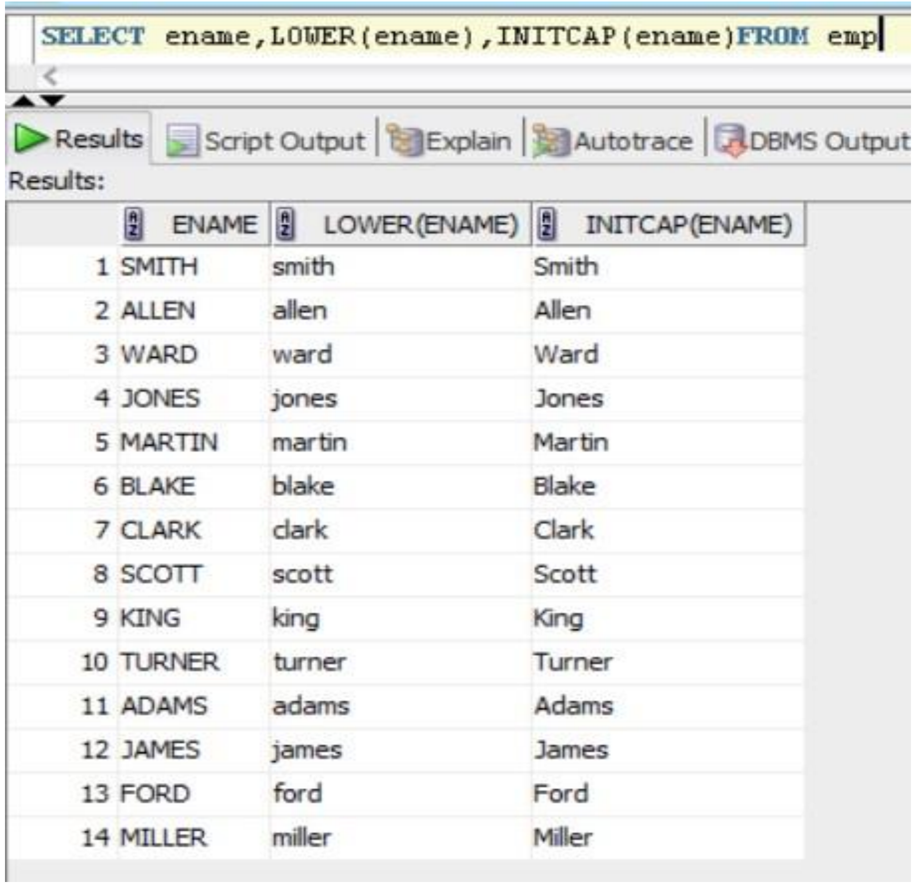
INITCAP	Returns a character expression, with the first letter of each word in uppercase, all other letters in lowercase	
		SELECT INITCAP('hello')
		FROM dual
		-- Result: 'Hello'

LOWER,UPPER,INITCAP



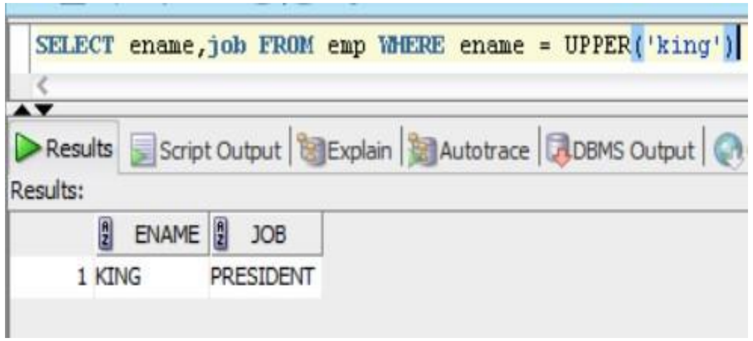
```sql
SELECT ename,job FROM emp WHERE LOWER(ename)='king'
```

	ENAME	JOB
1	KING	PRESIDENT



```sql
SELECT ename,LOWER(ename),INITCAP(ename)FROM emp
```

	ENAME	LOWER(ENAME)	INITCAP(ENAME)
1	SMITH	smith	Smith
2	ALLEN	allen	Allen
3	WARD	ward	Ward
4	JONES	jones	Jones
5	MARTIN	martin	Martin
6	BLAKE	blake	Blake
7	CLARK	clark	Clark
8	SCOTT	scott	Scott
9	KING	king	King
10	TURNER	turner	Turner
11	ADAMS	adams	Adams
12	JAMES	james	James
13	FORD	ford	Ford
14	MILLER	miller	Miller



```sql
SELECT ename,job FROM emp WHERE ename = UPPER('king')
```

	ENAME	JOB
1	KING	PRESIDENT

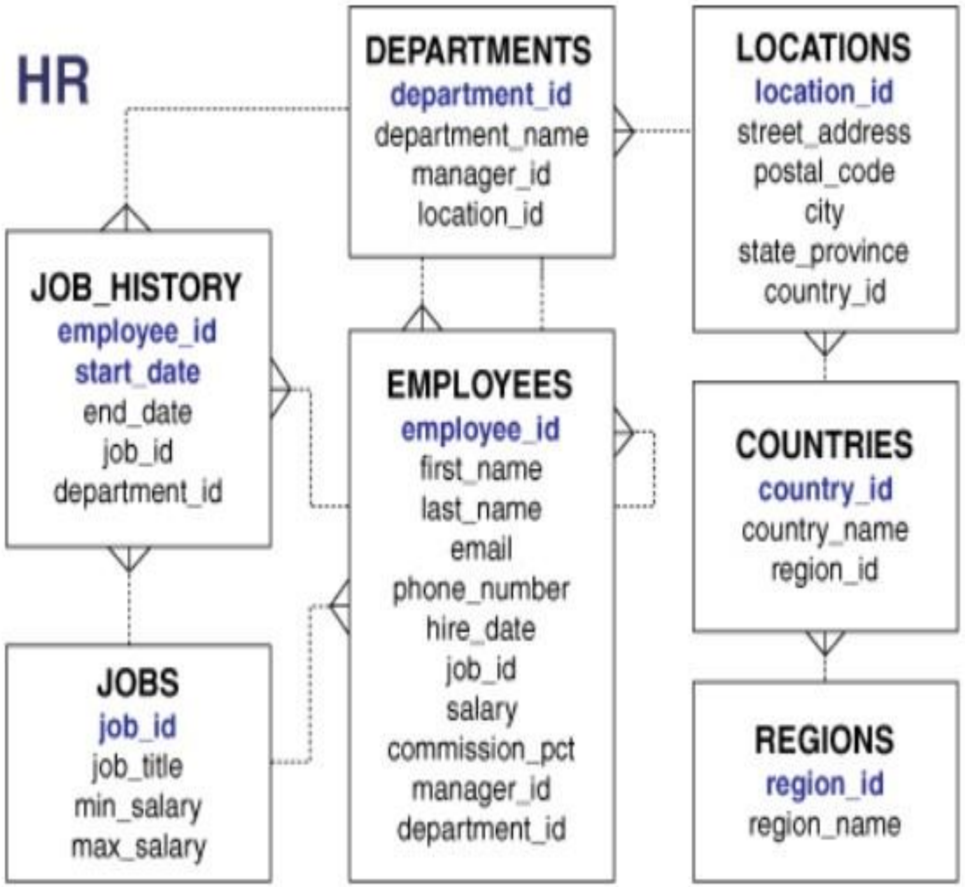
Case-insensitive search (regardless of the capitalization used for the values within *ename* column).

TASK A

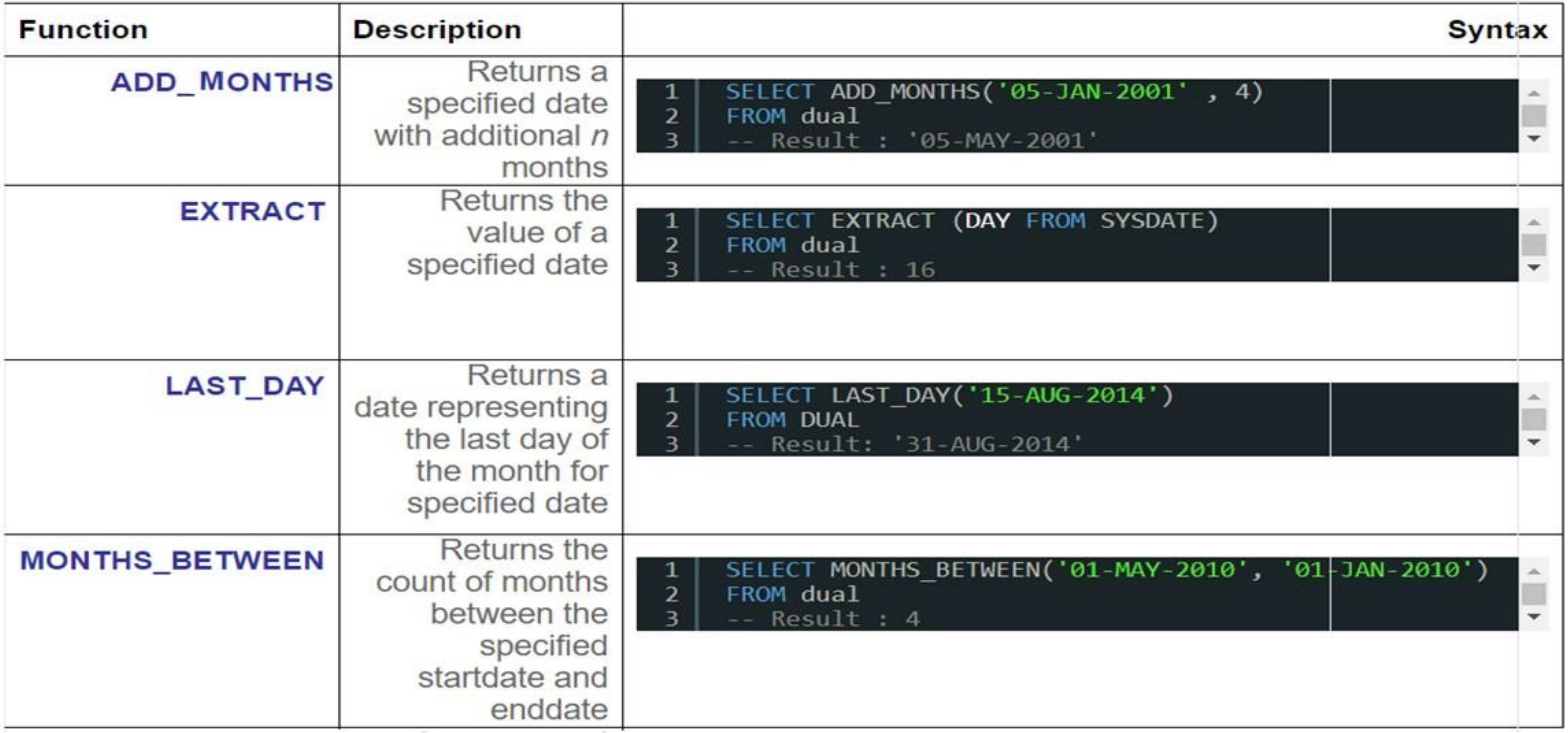
1.Generating new email address

- For each employee, display the first name, last name, and email address. The email address will be composed from the first letter of first name, concatenated with the three first letters of last name, concatenated with *@abc.com*.

- For each employee, display the first name, last name, and email address. The email address will be composed from the first letter of first name, concatenated with the three last letters of last name, concatenated with *@abc.com*.



DATE FUNCTIONS



Function	Description	Syntax

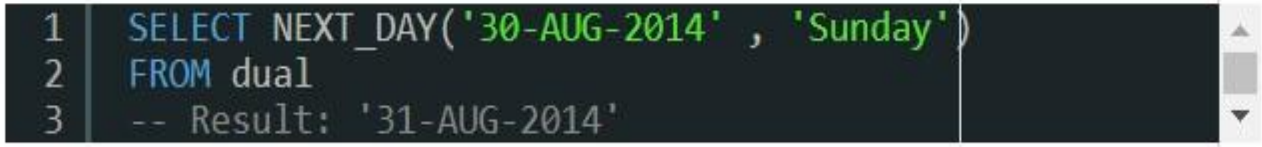
ADD_MONTHS	Returns a specified date with additional *n* months	`SELECT ADD_MONTHS('05-JAN-2001' , 4)`
		`FROM dual`
		`-- Result : '05-MAY-2001'`

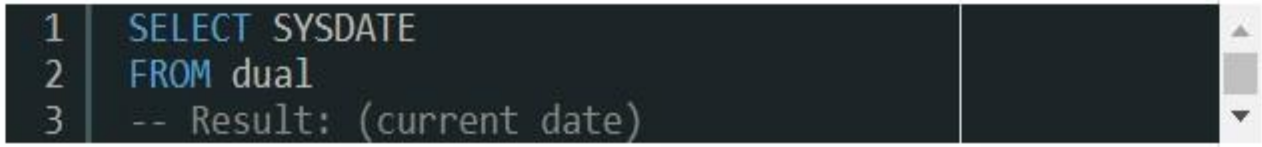
EXTRACT	Returns the value of a specified date	`SELECT EXTRACT (DAY FROM SYSDATE)`
		`FROM dual`
		`-- Result : 16`

LAST_DAY	Returns a date representing the last day of the month for specified date	`SELECT LAST_DAY('15-AUG-2014')`
		`FROM DUAL`
		`-- Result: '31-AUG-2014'`

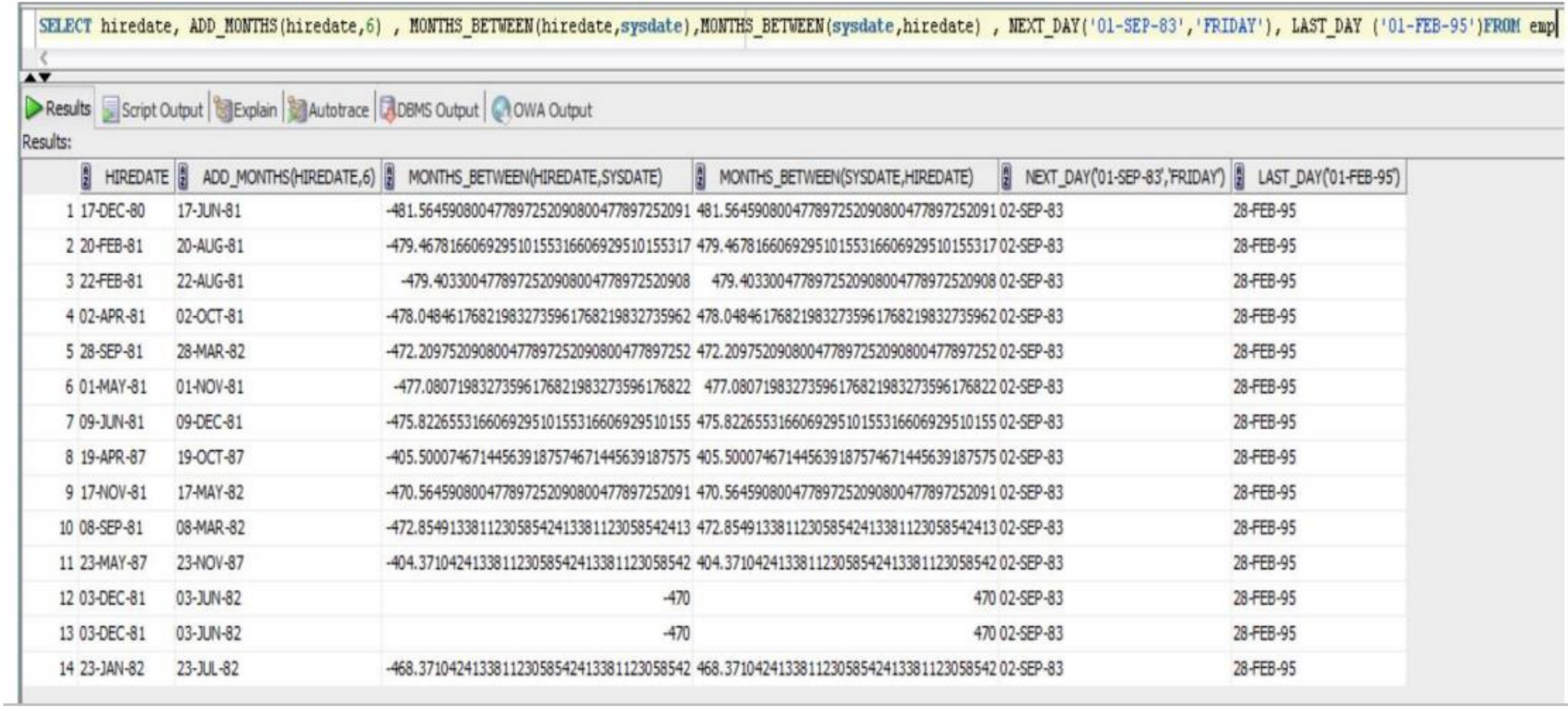
MONTHS_BETWEEN	Returns the count of months between the specified startdate and enddate	`SELECT MONTHS_BETWEEN('01-MAY-2010', '01-JAN-2010')`
		`FROM dual`
		`-- Result : 4`

Function	Description	Syntax

NEXT_DAY	returns the first weekday that is greater than the specified date	
		`SELECT NEXT_DAY('30-AUG-2014' , 'Sunday')`
		`FROM dual`
		`-- Result: '31-AUG-2014'`

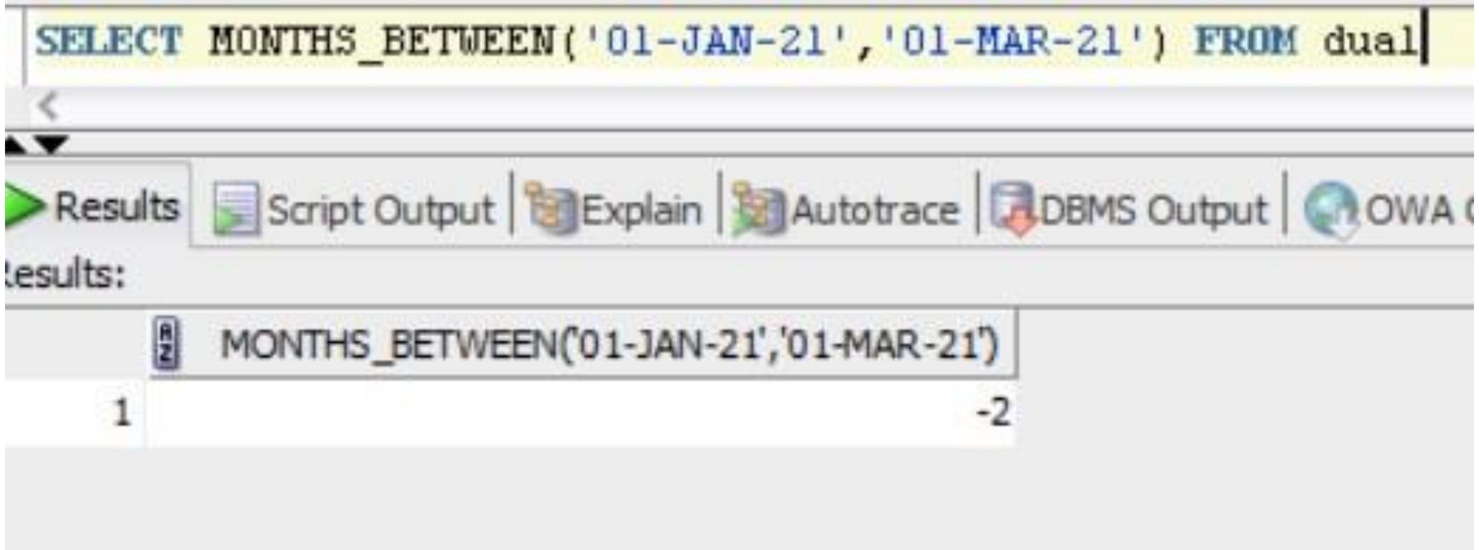
SYSDATE()	Returns the current database system date. This value is derived from the operating system of the computer on which the instance of Oracle is running	
		`SELECT SYSDATE`
		`FROM dual`
		`-- Result: (current date)`

ADD_MONTHS(d,m) MONTHS_BETWEEN(d1,d2)
NEXT_DAY(d,'character') LAST_DAY(d)



```
SELECT hiredate, ADD_MONTHS(hiredate,6) , MONTHS_BETWEEN(hiredate,sysdate),MONTHS_BETWEEN(sysdate,hiredate) , NEXT_DAY('01-SEP-83','FRIDAY'), LAST_DAY ('01-FEB-95')FROM emp
```

	HIREDATE	ADD_MONTHS(HIREDATE,6)	MONTHS_BETWEEN(HIREDATE,SYSDATE)	MONTHS_BETWEEN(SYSDATE,HIREDATE)	NEXT_DAY('01-SEP-83','FRIDAY')	LAST_DAY('01-FEB-95')
1	17-DEC-80	17-JUN-81	-481.5645908004778972520908004778972520 91	481.564590800477897252090800477897252091	02-SEP-83	28-FEB-95
2	20-FEB-81	20-AUG-81	-479.467816606929510155316606929510155317	479.467816606929510155316606929510155317	02-SEP-83	28-FEB-95
3	22-FEB-81	22-AUG-81	-479.40330047789725209080047789725 20908	479.4033004778972520908004778972520908	02-SEP-83	28-FEB-95
4	02-APR-81	02-OCT-81	-478.048461768219832735961768219832735962	478.048461768219832735961768219832735962	02-SEP-83	28-FEB-95
5	28-SEP-81	28-MAR-82	-472.209752090800477897252090800477897252	472.209752090800477897252090800477897252	02-SEP-83	28-FEB-95
6	01-MAY-81	01-NOV-81	-477.08071983273596176821983273596176822	477.08071983273596176821983273596176822	02-SEP-83	28-FEB-95
7	09-JUN-81	09-DEC-81	-475.822655316606929510155316606929510155	475.822655316606929510155316606929510155	02-SEP-83	28-FEB-95
8	19-APR-87	19-OCT-87	-405.500074671445639187574671445639187575	405.500074671445639187574671445639187575	02-SEP-83	28-FEB-95
9	17-NOV-81	17-MAY-82	-470.564590800477897252090800477897252091	470.564590800477897252090800477897252091	02-SEP-83	28-FEB-95
10	08-SEP-81	08-MAR-82	-472.854913381123058542413381123058542413	472.854913381123058542413381123058542413	02-SEP-83	28-FEB-95
11	23-MAY-87	23-NOV-87	-404.371042413381123058542413381123058542	404.371042413381123058542413381123058542	02-SEP-83	28-FEB-95
12	03-DEC-81	03-JUN-82	-470	470	02-SEP-83	28-FEB-95
13	03-DEC-81	03-JUN-82	-470	470	02-SEP-83	28-FEB-95
14	23-JAN-82	23-JUL-82	-468.371042413381123058542413381123058542	468.371042413381123058542413381123058542	02-SEP-83	28-FEB-95



```
SELECT MONTHS_BETWEEN('01-JAN-21','01-MAR-21') FROM dual
```

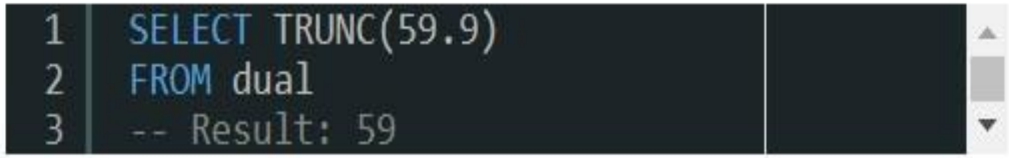
```
Results:

     MONTHS_BETWEEN('01-JAN-21','01-MAR-21')
   1                                      -2
```

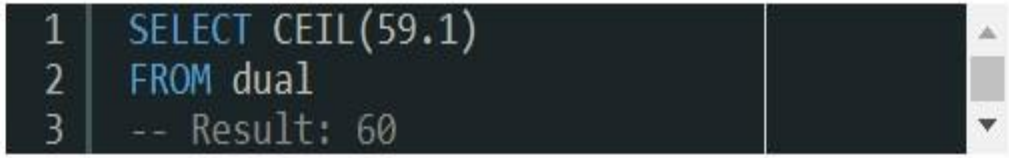
d2-d1

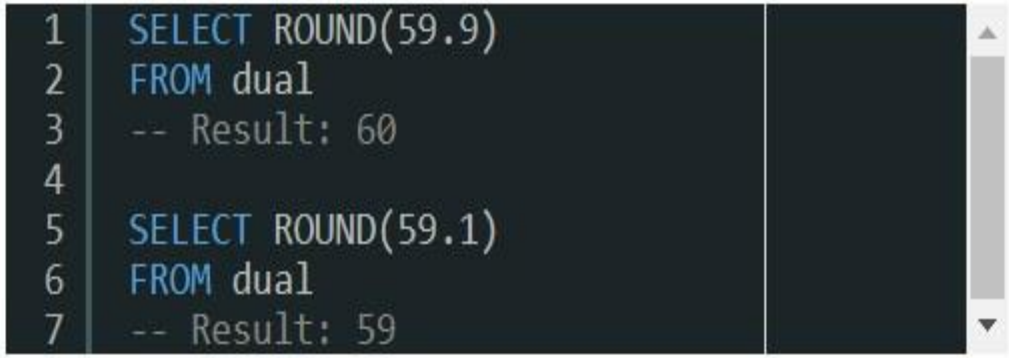
If we subtract two months from second date, then we will have the first date.

NUMBER FUNCTIONS

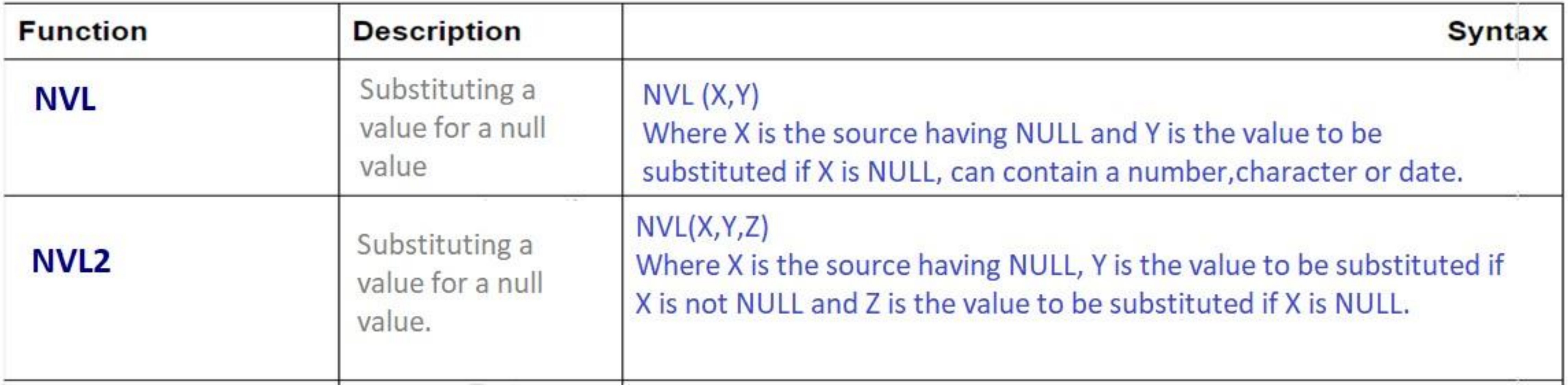
Function	Description	Syntax

TRUNC	Returns an integer that is less than or equal to the specified numeric expression	
		SELECT TRUNC(59.9)
		FROM dual
		-- Result: 59

CEIL	Returns an integer that is greater than, or equal to, the specified numeric expression	
		SELECT CEIL(59.1)
		FROM dual
		-- Result: 60

ROUND	Returns a numeric value, rounded to the specified length or precision	
		SELECT ROUND(59.9)
		FROM dual
		-- Result: 60
		
		SELECT ROUND(59.1)
		FROM dual
		-- Result: 59

NULL-RELATED FUNCTIONS



Function	Description	Syntax
NVL	Substituting a value for a null value	NVL (X,Y) Where X is the source having NULL and Y is the value to be substituted if X is NULL, can contain a number,character or date.
NVL2	Substituting a value for a null value.	NVL(X,Y,Z) Where X is the source having NULL, Y is the value to be substituted if X is not NULL and Z is the value to be substituted if X is NULL.

NVL FUNCTION

EXAMPLE A:

Calculate the gross pay of each employee.

SELECT ename , sal+ NVL(comm,0)

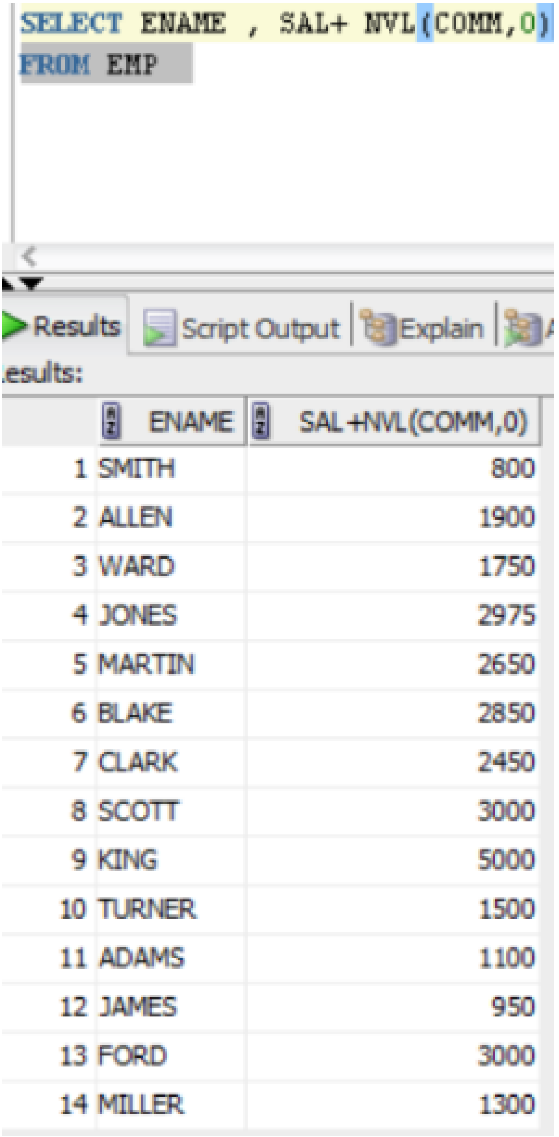
FROM emp ;

EXAMPLE B:

SELECT ename , sal+ NVL(NULL,0)

FROM emp ;

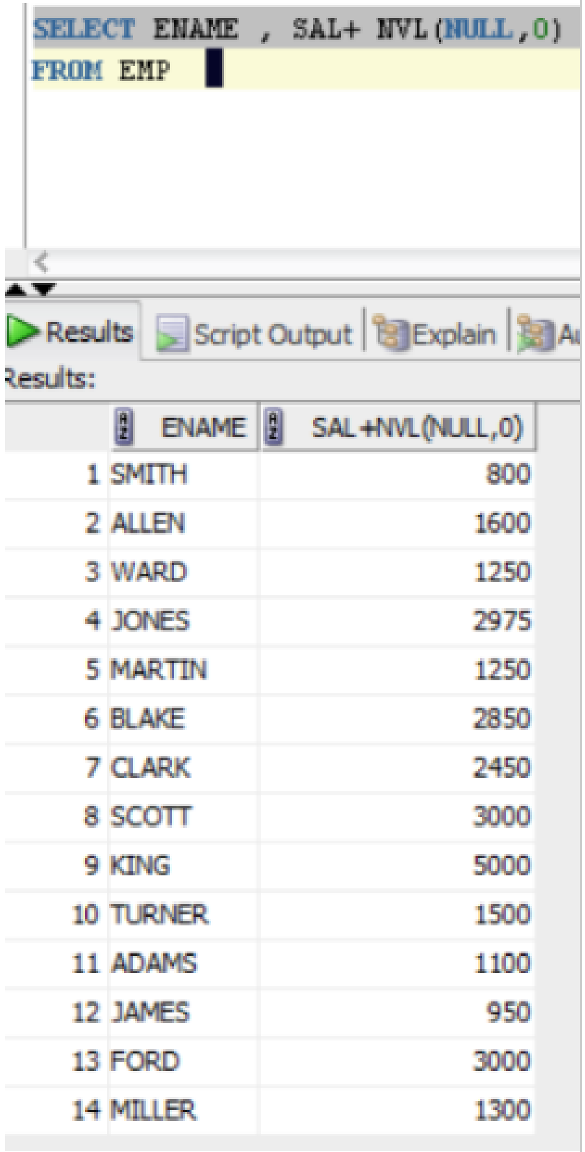
WHY IS THE SALARY OF ALLEN DIFFERENT IN BOTH THE CASES ?



```
SELECT ENAME , SAL+ NVL(COMM,0)
FROM EMP
```

esults:

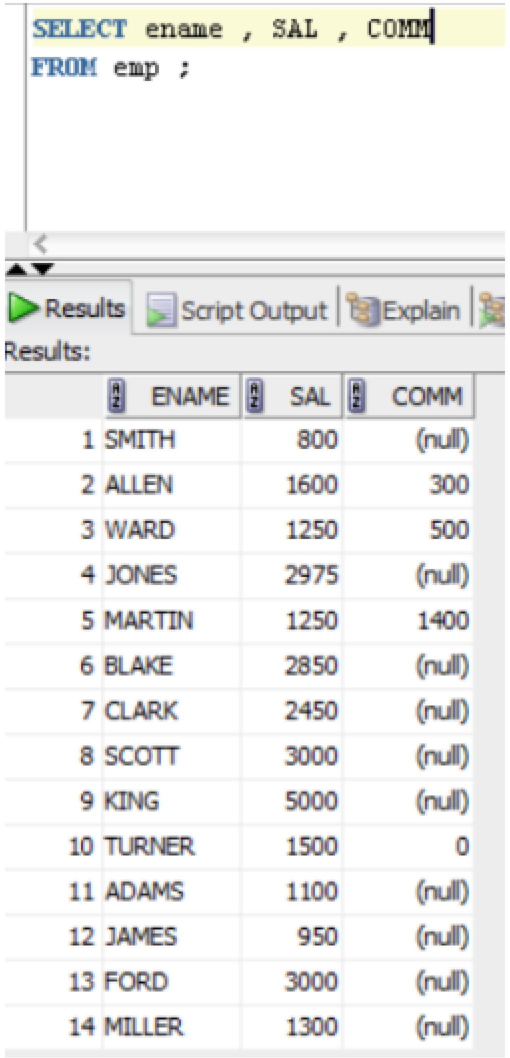
	ENAME	SAL+NVL(COMM,0)
1	SMITH	800
2	ALLEN	1900
3	WARD	1750
4	JONES	2975
5	MARTIN	2650
6	BLAKE	2850
7	CLARK	2450
8	SCOTT	3000
9	KING	5000
10	TURNER	1500
11	ADAMS	1100
12	JAMES	950
13	FORD	3000
14	MILLER	1300



```
SELECT ENAME , SAL+ NVL(NULL,0)
FROM EMP
```

Results:

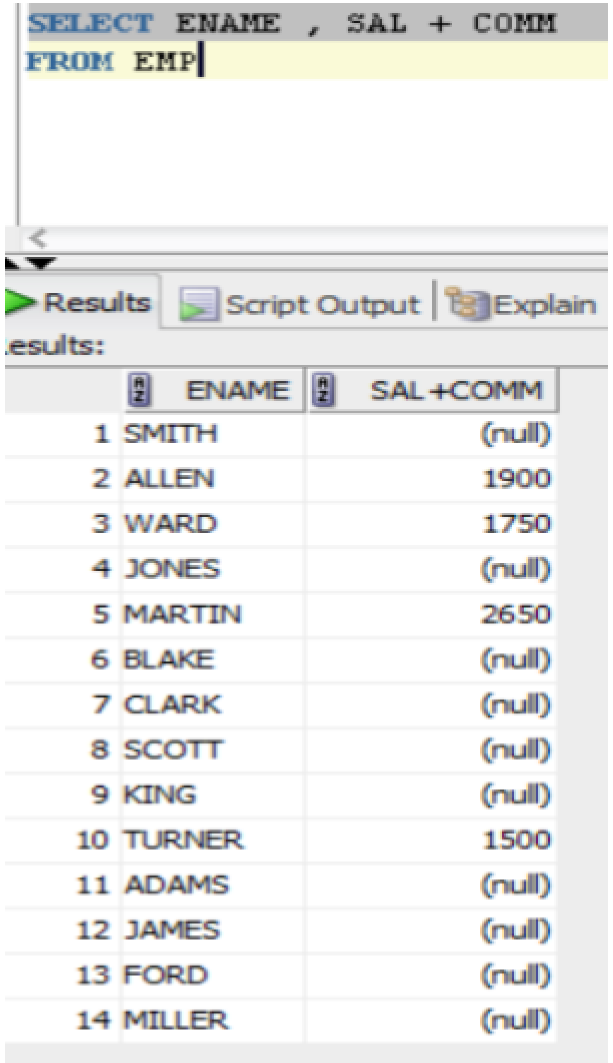
	ENAME	SAL+NVL(NULL,0)
1	SMITH	800
2	ALLEN	1600
3	WARD	1250
4	JONES	2975
5	MARTIN	1250
6	BLAKE	2850
7	CLARK	2450
8	SCOTT	3000
9	KING	5000
10	TURNER	1500
11	ADAMS	1100
12	JAMES	950
13	FORD	3000
14	MILLER	1300



```
SELECT ename , SAL , COMM
FROM emp ;
```

	ENAME	SAL	COMM
1	SMITH	800	(null)
2	ALLEN	1600	300
3	WARD	1250	500
4	JONES	2975	(null)
5	MARTIN	1250	1400
6	BLAKE	2850	(null)
7	CLARK	2450	(null)
8	SCOTT	3000	(null)
9	KING	5000	(null)
10	TURNER	1500	0
11	ADAMS	1100	(null)
12	JAMES	950	(null)
13	FORD	3000	(null)
14	MILLER	1300	(null)

ARITHMETIC EXPRESSIONS ARE EVALUATED TO NULL IF THEY INVOLVE A NULL VALUE IN THE OPERATION.



```
SELECT ENAME , SAL + COMM
FROM EMP
```

	ENAME	SAL +COMM
1	SMITH	(null)
2	ALLEN	1900
3	WARD	1750
4	JONES	(null)
5	MARTIN	2650
6	BLAKE	(null)
7	CLARK	(null)
8	SCOTT	(null)
9	KING	(null)
10	TURNER	1500
11	ADAMS	(null)
12	JAMES	(null)
13	FORD	(null)
14	MILLER	(null)

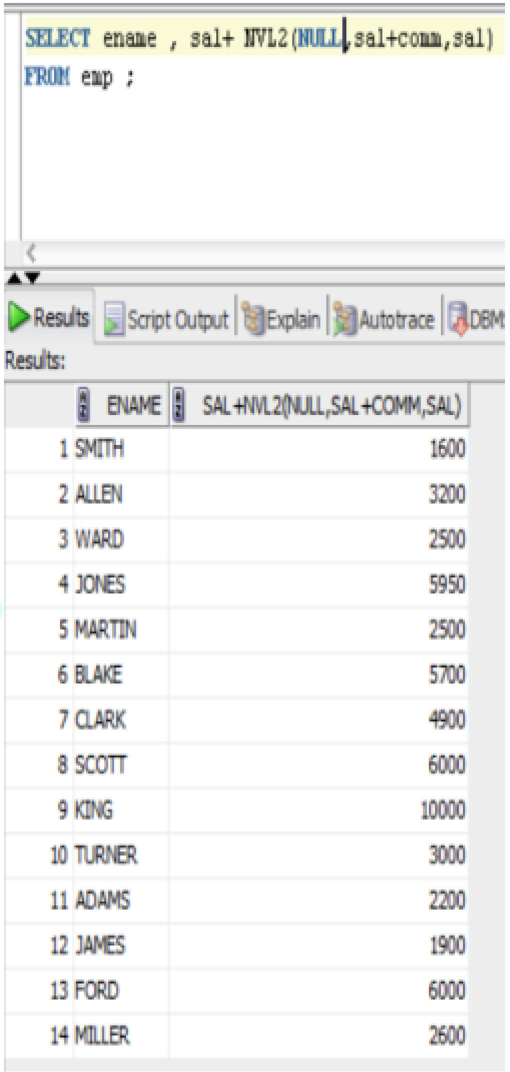
NVL2 FUNCTION

EXAMPLE C:

```
SELECT ename , sal+ NVL2(NULL,sal+comm,sal)
FROM emp ;
```

EXAMPLE D:

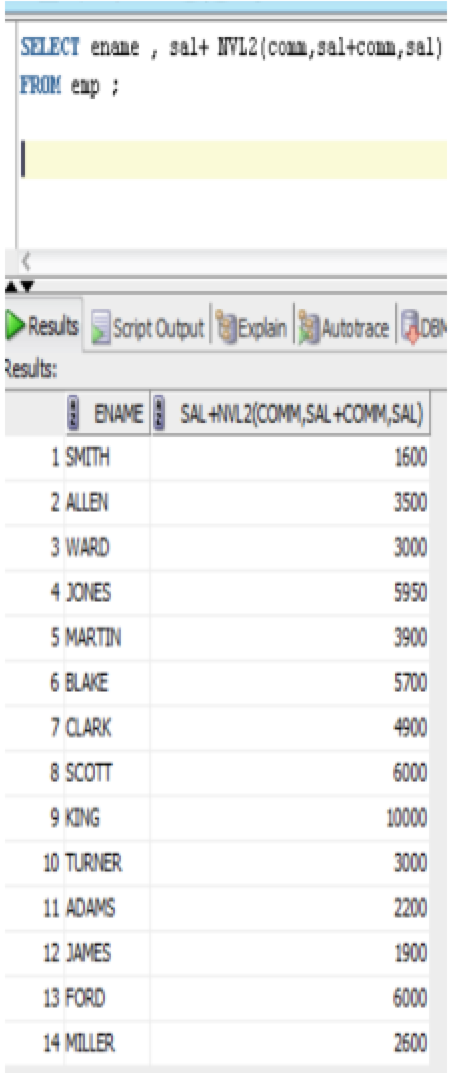
```
SELECT ename , sal+ NVL2(comm,sal+comm,sal)
FROM emp ;
```



```
SELECT ename , sal+ NVL2(NULL,sal+comm,sal)
FROM emp ;
```

Results:

	ENAME	SAL+NVL2(NULL,SAL+COMM,SAL)
1	SMITH	1600
2	ALLEN	3200
3	WARD	2500
4	JONES	5950
5	MARTIN	2500
6	BLAKE	5700
7	CLARK	4900
8	SCOTT	6000
9	KING	10000
10	TURNER	3000
11	ADAMS	2200
12	JAMES	1900
13	FORD	6000
14	MILLER	2600



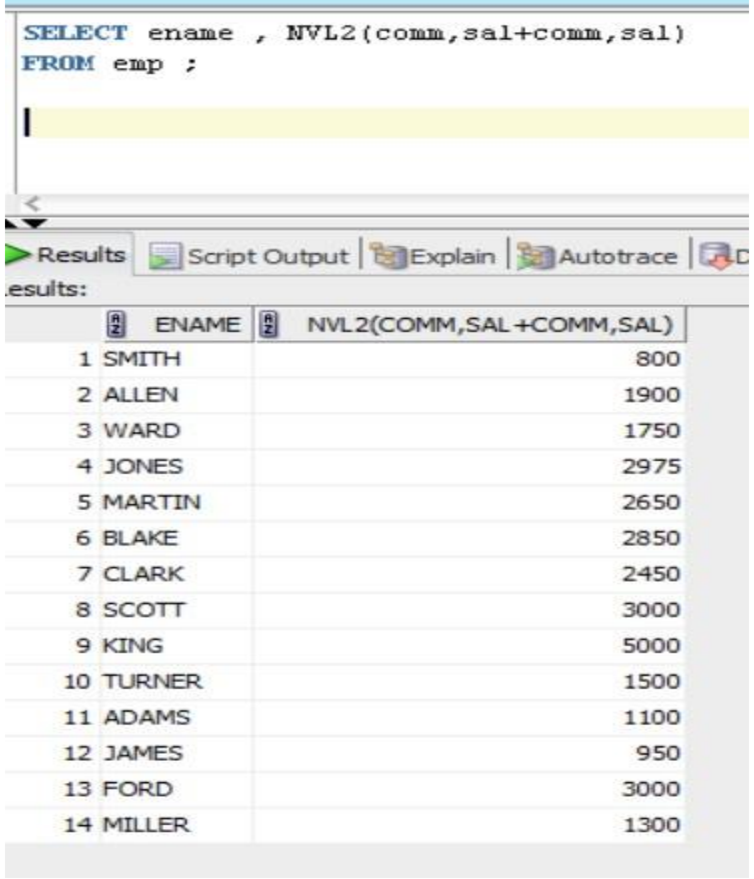
```
SELECT ename , sal+ NVL2(comm,sal+comm,sal)
FROM emp ;
```

Results:

	ENAME	SAL+NVL2(COMM,SAL+COMM,SAL)
1	SMITH	1600
2	ALLEN	3500
3	WARD	3000
4	JONES	5950
5	MARTIN	3900
6	BLAKE	5700
7	CLARK	4900
8	SCOTT	6000
9	KING	10000
10	TURNER	3000
11	ADAMS	2200
12	JAMES	1900
13	FORD	6000
14	MILLER	2600

TASK B

Find the GROSS PAY of all employees using NVL2 function.



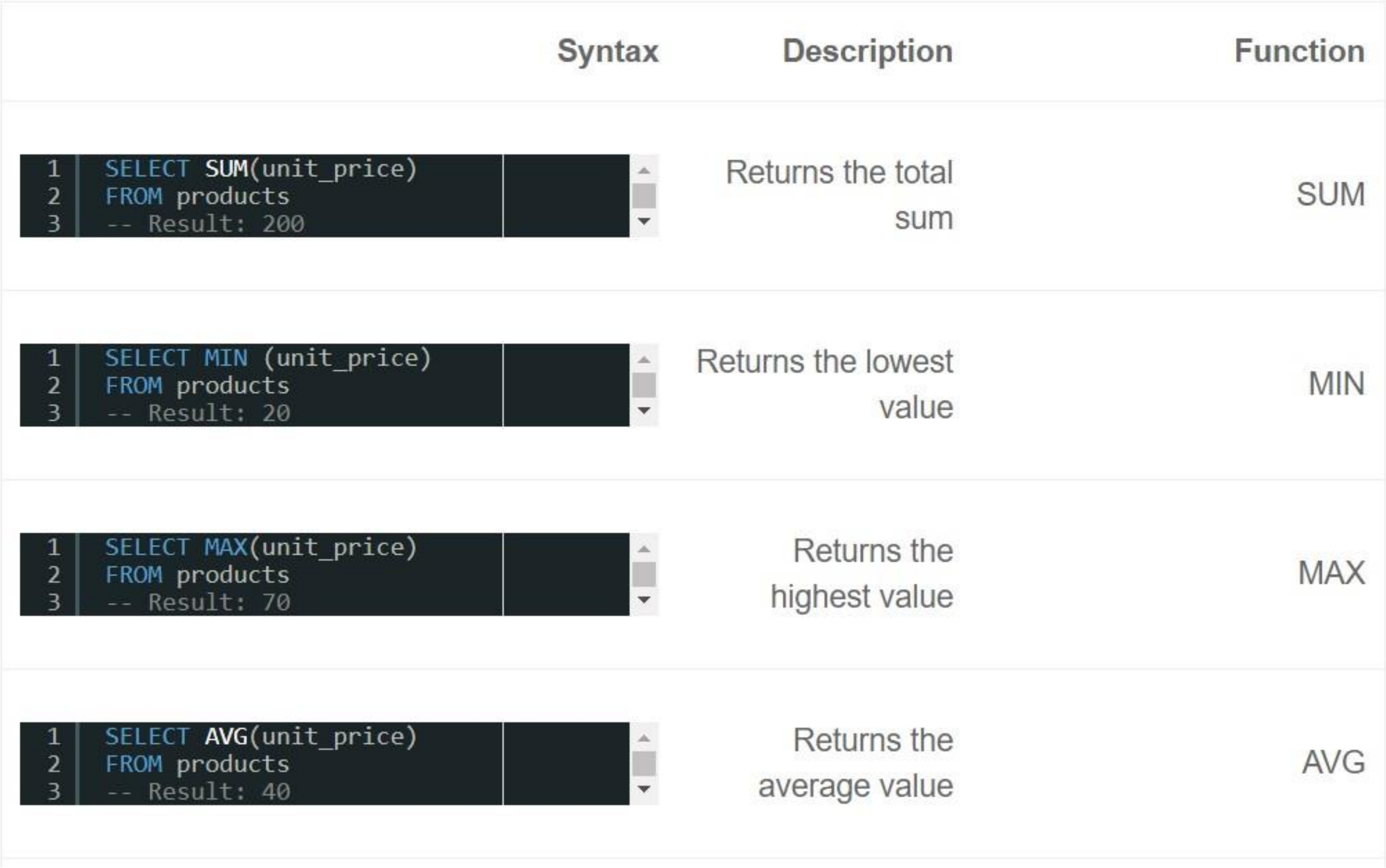
```sql
SELECT ename , NVL2(comm,sal+comm,sal)
FROM emp ;
```

esults:

	ENAME	NVL2(COMM,SAL+COMM,SAL)
1	SMITH	800
2	ALLEN	1900
3	WARD	1750
4	JONES	2975
5	MARTIN	2650
6	BLAKE	2850
7	CLARK	2450
8	SCOTT	3000
9	KING	5000
10	TURNER	1500
11	ADAMS	1100
12	JAMES	950
13	FORD	3000
14	MILLER	1300

GROUP FUNCTIONS

- **Group Functions** process the values of multiple rows to give one result per group.
- Unlike Scalar Functions, Group Functions process the values of multiple rows to give one result per group.
- They are also known as **Multiple Row** or **Aggregate Functions**.
- All Group functions ignore NULL values.
- You can use the NVL function to force group functions to include **NULL** values.
- Groups are formed using the **GROUP BY** clause, incase its not used then the whole table is considered as one group.



Syntax	Description	Function

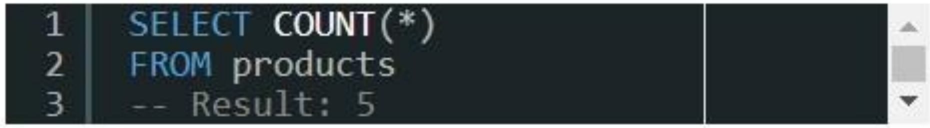
`SELECT SUM(unit_price)`
`FROM products`
`-- Result: 200`
	Returns the total sum	SUM

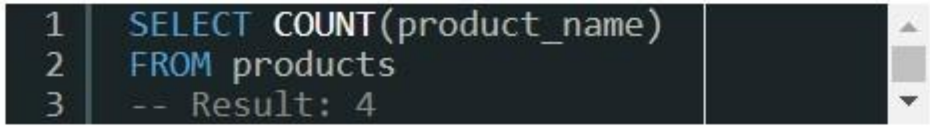
`SELECT MIN (unit_price)`
`FROM products`
`-- Result: 20`
	Returns the lowest value	MIN

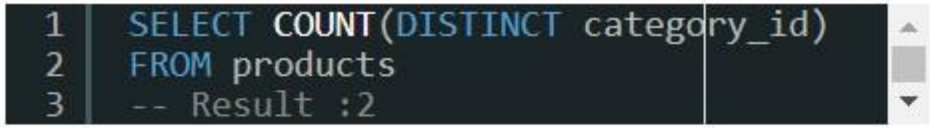
`SELECT MAX(unit_price)`
`FROM products`
`-- Result: 70`
	Returns the highest value	MAX

`SELECT AVG(unit_price)`
`FROM products`
`-- Result: 40`
	Returns the average value	AVG

Syntax	Description	Function

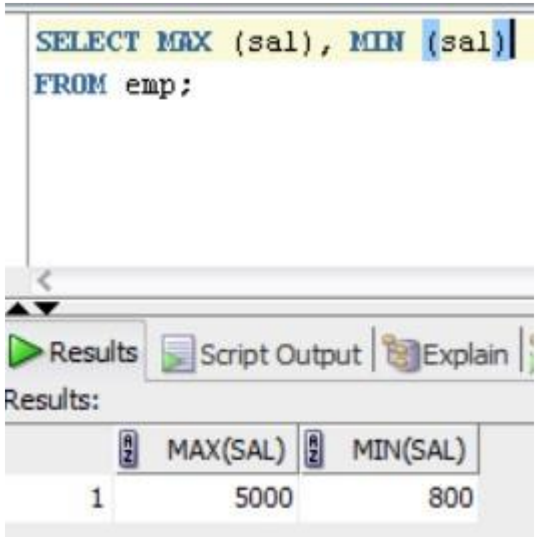

SELECT COUNT(*)
FROM products
-- Result: 5
	Returns the number of records in a table	(*) COUNT


SELECT COUNT(product_name)
FROM products
-- Result: 4
	Returns the number of values (NULL values will not be counted) of the specified column	COUNT (column)


SELECT COUNT(DISTINCT category_id)
FROM products
-- Result :2
	Returns the number of distinct values	COUNT (DISTINCT column)

EXAMPLE E:

SELECT MAX (sal), MIN (sal) FROM emp;



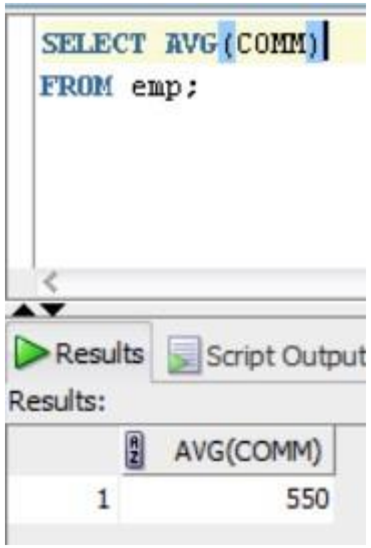
```
SELECT MAX (sal), MIN (sal)
FROM emp;
```

```
Results:
     MAX(SAL)  MIN(SAL)
1        5000       800
```

EXAMPLE F:

SELECT AVG(comm)

FROM emp;



```
SELECT AVG(COMM)
FROM emp;
```

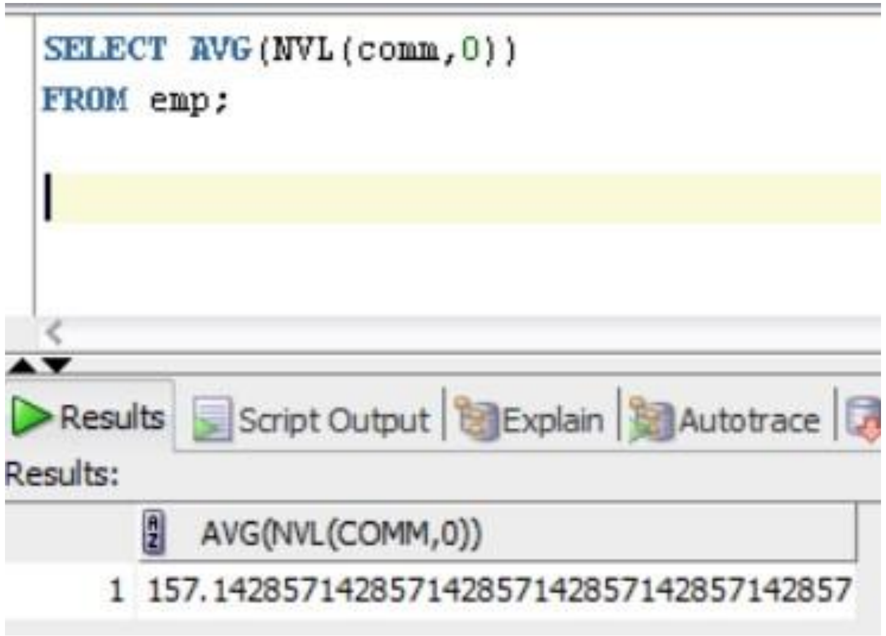
```
Results:
     AVG(COMM)
1          550
```

EXAMPLE G:

SELECT

AVG(NVL(comm,0))

FROM emp;



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

```
Results:
     AVG(NVL(COMM,0))
1  157.142857142857142857142857142857142857
```

WHY IS THE AVERAGE DIFFERENT IN EXAMPLE F AND EXAMPLE G ?

GROUP BY CLAUSE

It is used to form groups of data within a table.

Guidelines:

1. Column alias cannot be used.
2. Results returned from a select statement that includes the clause are by default in descending order.
3. If a group function is used in the Select clause then any individual column listed in the **SELECT** clause must also be listed in the **GROUP BY** clause.
4. Every Column used in **GROUP BY** clause does not need to be listed in the **SELECT** clause i.e., one is good.

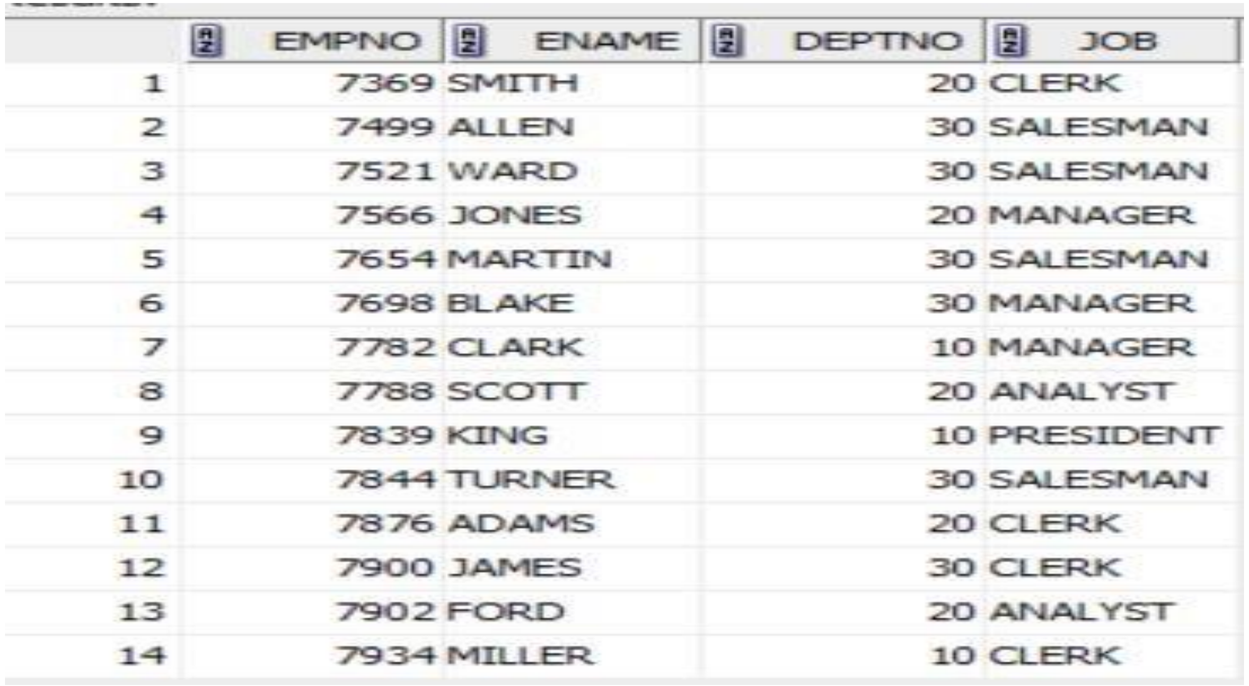
SYNTAX:

GROUP BY column_ name [,………….column_ name]

EXAMPLE H:

SELECT EMPNO,ENAME,DEPTNO,JOB

FROM EMP ;



	EMPNO	ENAME	DEPTNO	JOB
1	7369	SMITH	20	CLERK
2	7499	ALLEN	30	SALESMAN
3	7521	WARD	30	SALESMAN
4	7566	JONES	20	MANAGER
5	7654	MARTIN	30	SALESMAN
6	7698	BLAKE	30	MANAGER
7	7782	CLARK	10	MANAGER
8	7788	SCOTT	20	ANALYST
9	7839	KING	10	PRESIDENT
10	7844	TURNER	30	SALESMAN
11	7876	ADAMS	20	CLERK
12	7900	JAMES	30	CLERK
13	7902	FORD	20	ANALYST
14	7934	MILLER	10	CLERK

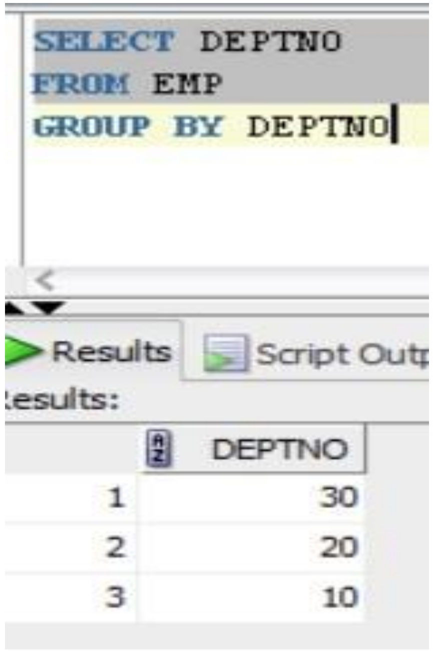
EXAMPLE I:

SELECT DEPTNO

FROM EMP

GROUP BY DEPTNO ;

Results returned from a select statement that includes the clause are by default in descending order.



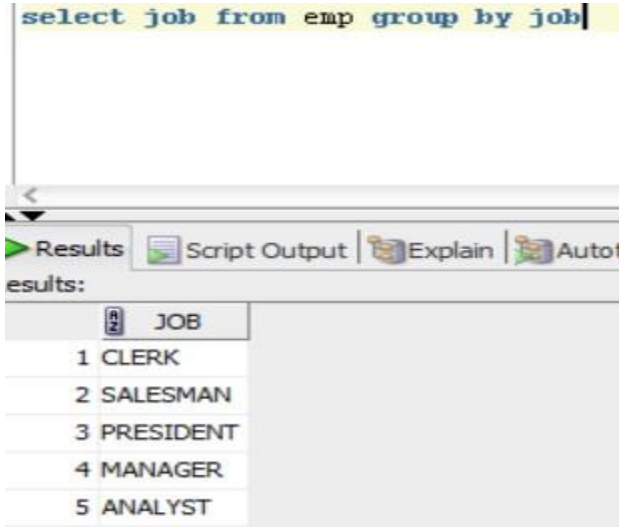
```
SELECT DEPTNO
FROM EMP
GROUP BY DEPTNO
```

esults:

	DEPTNO
1	30
2	20
3	10

EXAMPLE J:

```
SELECT job
FROM emp
GROUP BY job ;
```

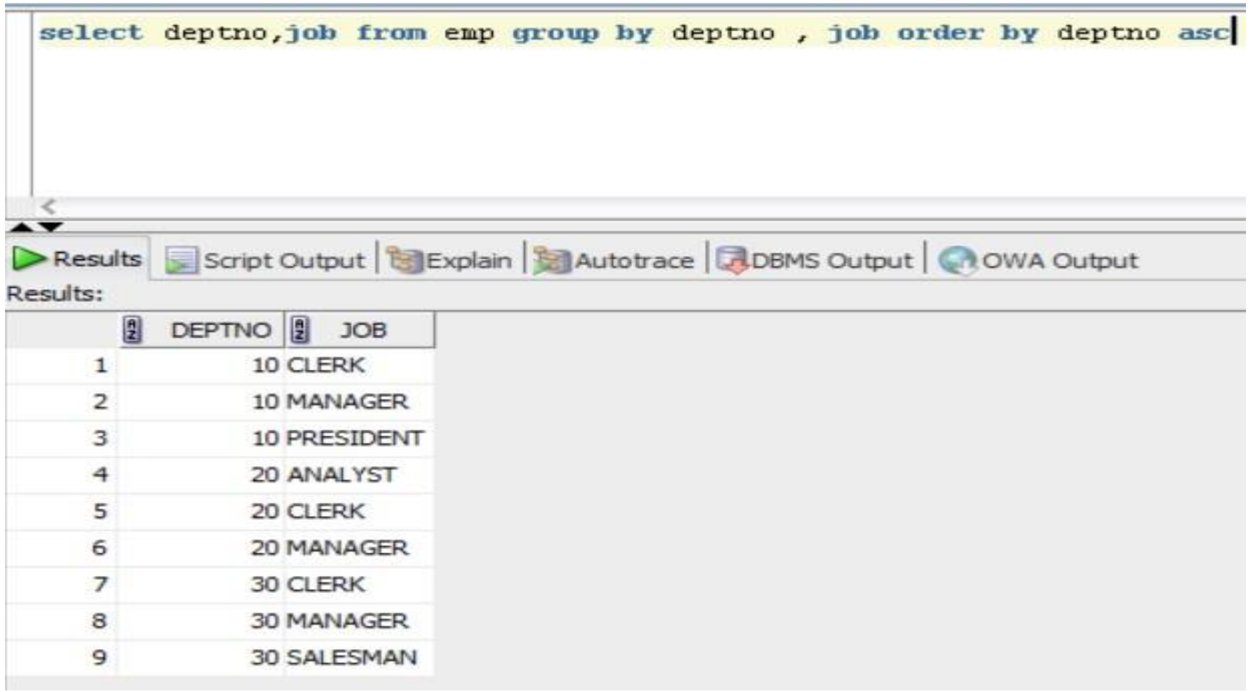


```
select job from emp group by job
```

```
esults:
     JOB
   1 CLERK
   2 SALESMAN
   3 PRESIDENT
   4 MANAGER
   5 ANALYST
```

EXAMPLE K:

```
SELECT deptno , job
FROM emp
GROUP BY deptno , job
ORDER BY deptno ASC ;
```



```
select deptno,job from emp group by deptno , job order by deptno asc
```

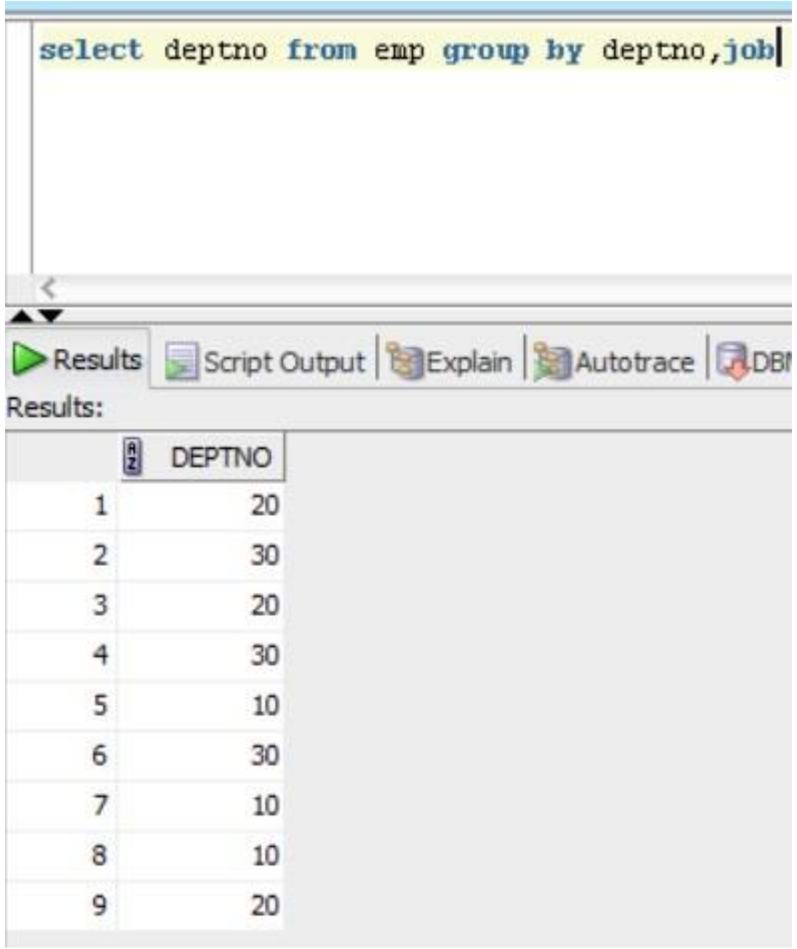
```
Results:
        DEPTNO  JOB
   1        10 CLERK
   2        10 MANAGER
   3        10 PRESIDENT
   4        20 ANALYST
   5        20 CLERK
   6        20 MANAGER
   7        30 CLERK
   8        30 MANAGER
   9        30 SALESMAN
```

Every Column used in **GROUP BY** clause does not need to be listed in the **SELECT** clause i.e., one is good

EXAMPLE L:

```
SELECT deptno
FROM emp
GROUP BY deptno , job ;
```

Column alias cannot be used in the **GROUP BY** clause.



```
select deptno from emp group by deptno,job
```

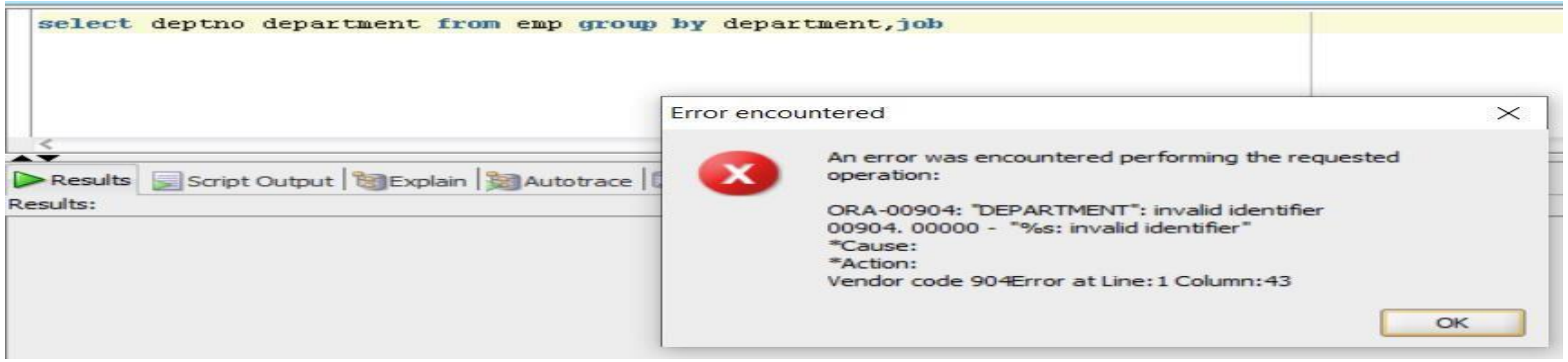
Results:

	DEPTNO
1	20
2	30
3	20
4	30
5	10
6	30
7	10
8	10
9	20

EXAMPLE M:

```
SELECT deptno department
FROM emp
GROUP BY department,job
```

If a group function is used in the Select clause then any individual column listed in the **SELECT** clause must also be listed in the **GROUP BY** clause.



```
select deptno department from emp group by department,job
```

Results:

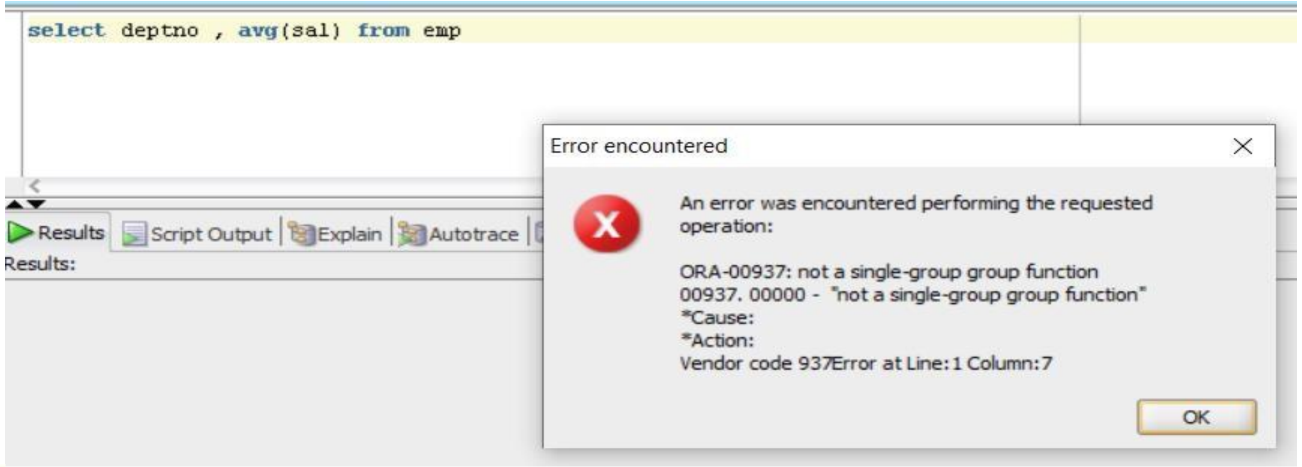
Error encountered

```
An error was encountered performing the requested
operation:

ORA-00904: "DEPARTMENT": invalid identifier
00904. 00000 -  "%s: invalid identifier"
*Cause:
*Action:
Vendor code 904Error at Line:1 Column:43
```

EXAMPLE N:

SELECT deptno, AVG(sal)

FROM emp ;



```
select deptno , avg(sal) from emp
```

An error was encountered performing the requested operation:

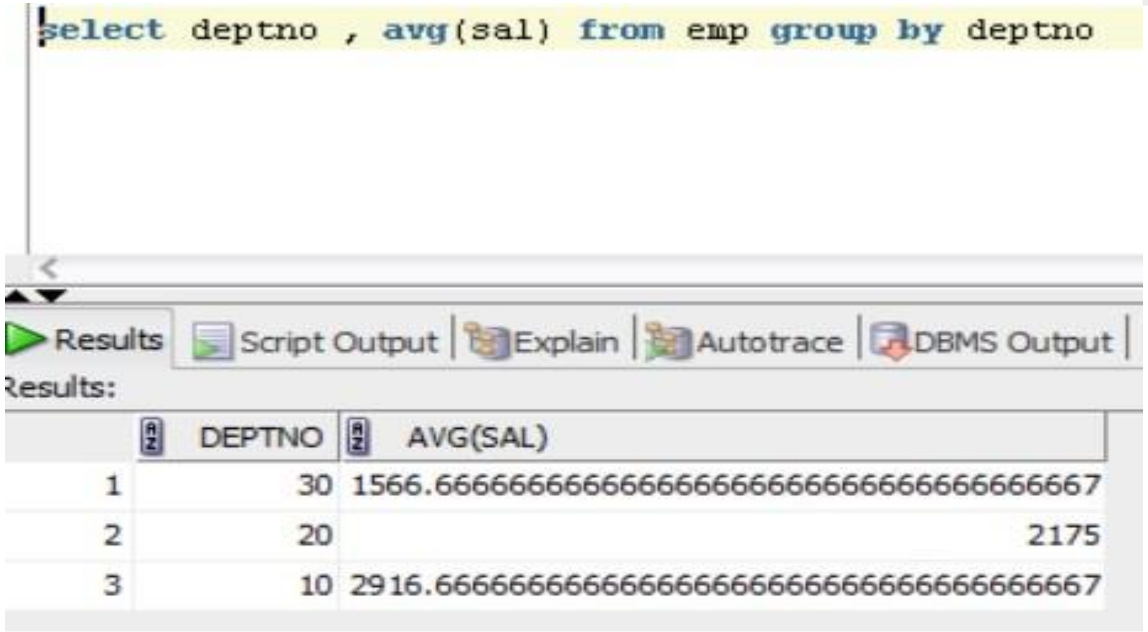
ORA-00937: not a single-group group function
00937. 00000 - "not a single-group group function"
*Cause:
*Action:
Vendor code 937Error at Line:1 Column:7

EXAMPLE O:

SELECT deptno, AVG(sal)

FROM emp

GROUP BY deptno;

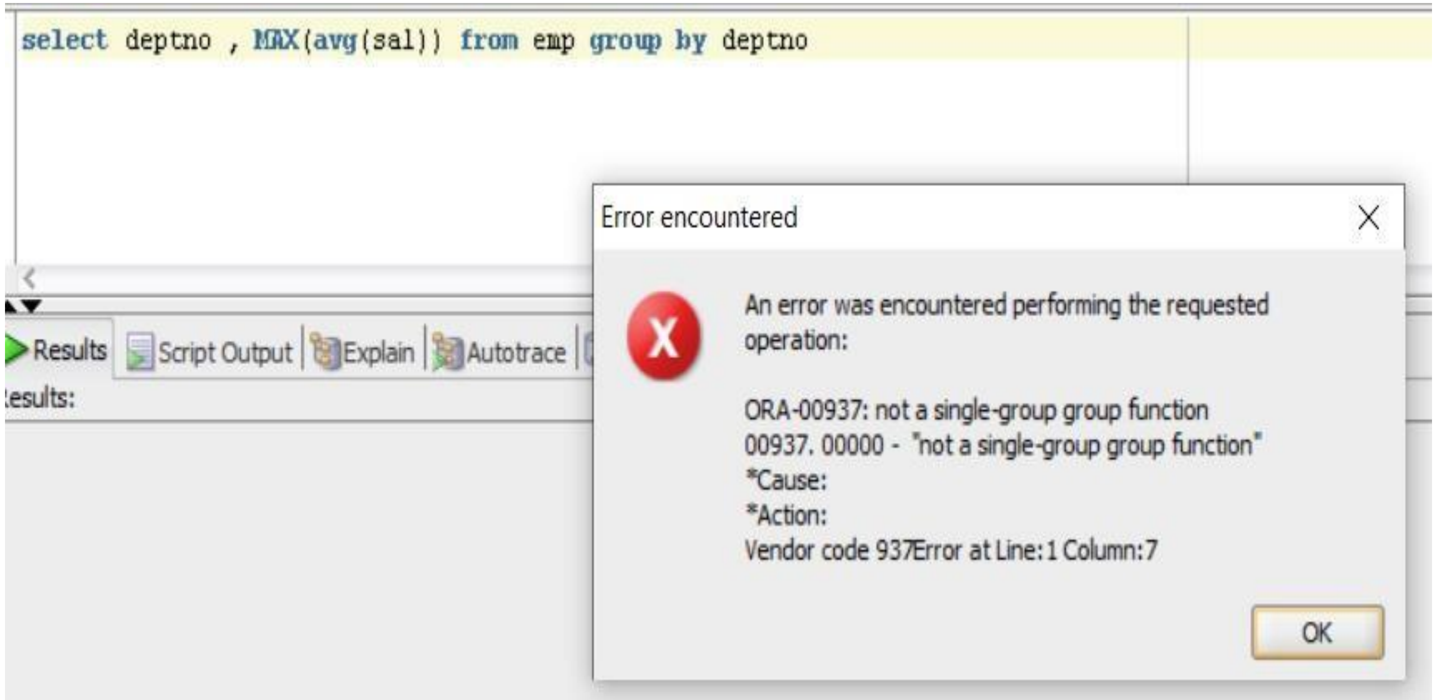


```
select deptno , avg(sal) from emp group by deptno
```

	DEPTNO	AVG(SAL)
1	30	1566.6666666666666666666666666666666667
2	20	2175
3	10	2916.6666666666666666666666666666666667

EXAMPLE P:

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



```
select deptno , MAX(avg(sal)) from emp group by deptno
```

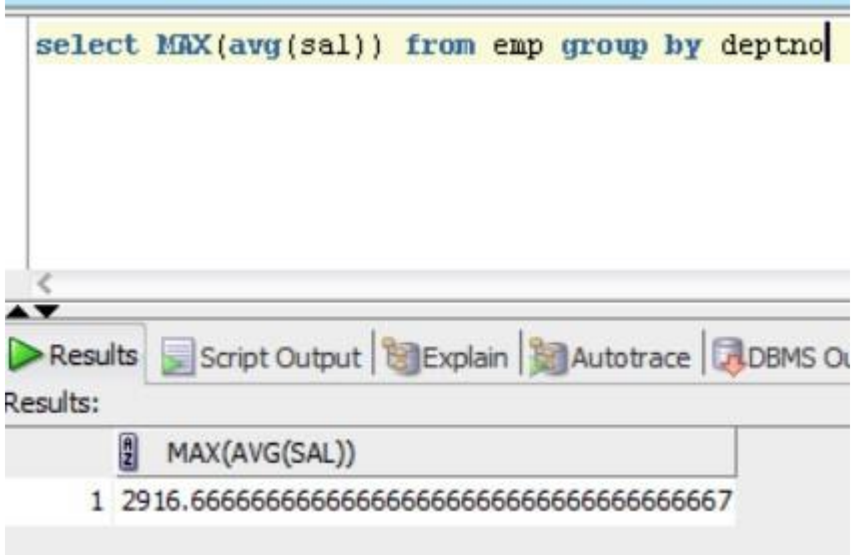
Error encountered

An error was encountered performing the requested operation:

```
ORA-00937: not a single-group group function
00937. 00000 -  "not a single-group group function"
*Cause:
*Action:
Vendor code 937Error at Line:1 Column:7
```

EXAMPLE Q:

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



```
select MAX(avg(sal)) from emp group by deptno
```

```
     MAX(AVG(SAL))
1  2916.6666666666666666666666666666666667
```

TASK C

1. **Display manger id and the salary of the lowest paid employee for that manger, exclude any those whose manger is unknown and sort the result in descending order of the lowest salary.**
2. **Display the total salary being paid to each job title within each department.**
3. **Find the total annual salary distributed job wise in the year 81.**
4. **List the Manager ids & number of employees working for those managers in the ascending order.**
5. **Find the number of employees who are serving as CLERK?**
6. **Find the total salary given to the MANAGERS?**

HAVING CLAUSE

- The HAVING clause is like WHERE but operates on grouped records returned by a GROUP BY clause.
- HAVING applies to summarized group records, whereas WHERE applies to individual records.
- Only the groups that meet the HAVING criteria are returned.
- To restrict group results we use the HAVING clause.
- It's used only for group conditions.
- WHERE clause is not used for applying conditions on group functions.
- HAVING requires that a GROUP BY clause is present.
- Both WHERE and HAVING can be used in the same query at the same time.

SYNTAX:

1.SELECT column-names

2.FROM table-name

3.WHERE condition

4.GROUP BY column-names

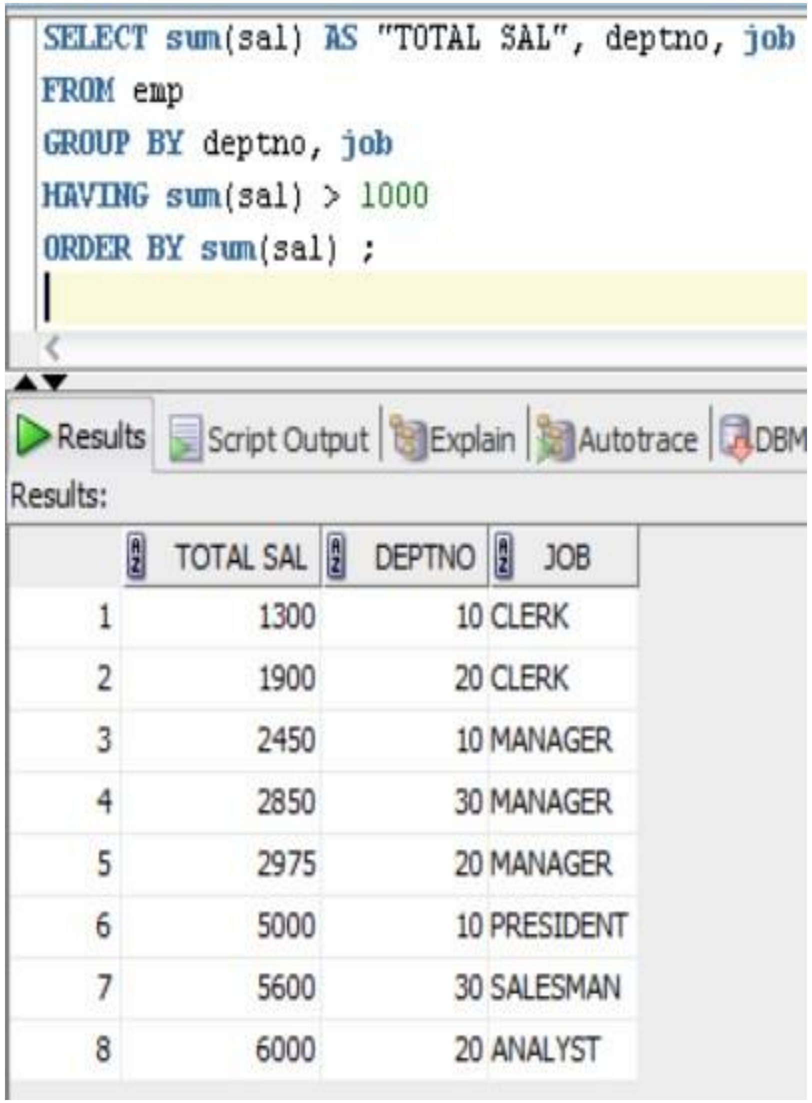
5.HAVING condition

6.ORDER BY column-names

HAVING group_function comparison_operator value EXAMPLE R:

Display the jobs, department no: and the total monthly salary for each job title within each department, with a total payroll exceeding 1000. Sort the list by total monthly salary.

```
SELECT sum(sal) AS "TOTAL SAL", deptno, job

FROM emp

GROUP BY deptno, job

HAVING sum(sal) > 1000

ORDER BY sum(sal) ;
```



```
SELECT sum(sal) AS "TOTAL SAL", deptno, job
FROM emp
GROUP BY deptno, job
HAVING sum(sal) > 1000
ORDER BY sum(sal) ;
```

Results:

	TOTAL SAL	DEPTNO	JOB
1	1300	10	CLERK
2	1900	20	CLERK
3	2450	10	MANAGER
4	2850	30	MANAGER
5	2975	20	MANAGER
6	5000	10	PRESIDENT
7	5600	30	SALESMAN
8	6000	20	ANALYST

TASK D

1. **List the departments where at least two employees are working.**
2. **List the number of employees in each department where the number of employees exceeds 3.**
3. **Find out the least 5 earners of the emp table.**