Operators & SQL Functions & Views

Prepared By Dr. R. B. Shinde

Operators

- An operator is a reserved word or a character used primarily in an SQL statement's WHERE clause to perform operation(s), such as comparisons and arithmetic operations. These Operators are used to specify conditions in an SQL statement and to serve as conjunctions for multiple conditions in a statement.
- Arithmetic operators
- Comparison operators
- Logical operators
- Operators used to negate conditions

SQL Arithmetic Operators

Assume **'variable a'** holds 10 and **'variable b'** holds 20, then

Operator	Description	Example
+ (Addition)	Adds values on either side of the operator.	a + b will give 30
- (Subtraction)	Subtracts right hand operand from left hand operand.	a - b will give -10
* (Multiplication)	Multiplies values on either side of the operator.	a * b will give 200
/ (Division)	Divides left hand operand by right hand operand.	b / a will give 2
% (Modulus)	Divides left hand operand by right hand operand and returns remainder.	b % a will give 0

SQL Comparison Operators

Assume **'variable a'** holds 10 and **'variable b'** holds 20, then

Operator	Description	Example
=	Checks if the values of two operands are equal or not, if yes then condition becomes true.	(a = b) is not true.
!=	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(a != b) is true.
<>	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(a <> b) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(a > b) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(a < b) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(a >= b) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(a <= b) is true.
!<	Checks if the value of left operand is not less than the value of right operand, if yes then condition becomes true.	(a !< b) is false.
!>	Checks if the value of left operand is not greater than the value of right operand, if yes then condition becomes true.	(a !> b) is true.

SQL Logical Operators

- The following are logical operators used in SQL
- 1) AND Operator
- 2) OR Operator
- 3) NOT Operator

AND Operator

- AND operator allows creating an SQL statement using two or more condition. This operators can be used with SELECT, INSERT, UPDATE or DELETE.
- The AND operator requires that each condition must be met for the record to be included in the result set.
- If all of the condition specified using the AND operator are satisfied then the result will be displayed.

EXAMPLE

- Consider employee table having column **empid, name** and **salary**. You want to retrive the data of those employees having salary grater than 10000 and less than 20000. Then the command can be used as

```
SQL>  Select empid, name, salary
      Where
      Salary >10000 AND salary<20000;
```

OR operator

- The OR operator is used to compare multiple conditions. It can be used in any valid SQL statement such as SELECT, INSERT, UPDATE or DELETE

- The oracle engine process and displays all the rows in a table and display the result only when any of the condition specified using the OR operator is satisfied.

EXAMPLE

- Consider table **STUDENT** having column ROLLNO, NAME, ADDRESS and you want to display the records of those student who are from ‘LATUR’ OR ‘Pune’ city.

```
SQL> Select * from STUDENT
        where
        ADDRESS=‘Latur’OR
        ADDRESS=‘Pune’;
```

NOT Operator

- The oracle engine will process all rows in a table and display only those records that do not satisfy the condition specified.

Example

- Consider table STUDENT having ROLLNO, NAME and ADDRESS column and you don’t want to display the records of students those who having address PUNE & Latur.

```
SQL> select *FROM STUDENT
        WHERE
        NOT (ADDRESS=‘Latur’AND ADDRESS=‘Pune’);
```

Comparison Operators
BETWEEN

- When You want to search a data between range of value then we can use **BETWEEN** operator. It allows the selection of rows that contains value within a specified **LOWER** and **UPPER** limit.
- The two values in between the range must be linked with the keyword AND

Eg.

The employee having salary in between 10000 to 20000;

➢Select *from employee where

Salary between 10000 AND 20000;

Patter Matching

- Use of LIKE OPERATOR
  - The like predicate allows comparison of one string value with another string value, which is not identical. This can be used with some character like.

  For character data type:

  %- allows to match any string of any length(including zero length).

  _- allows to match on a single character.

  Eg. Display the records of those student whose name is starting with character ‘S’from STUD table.

```
>SELECT ROLLNO, NAME, ADDRESS
   FROM STUD
   WHERE
   NAME like ‘5%’;
```

- Eg. Select records of those students whose names have the second character as ‘a’ or ‘s’.

```
>select ROLLNO, NAME, ADDRESS from  STUD
          WHERE
          NAME like ‘_a%’OR name like ‘_s%’;
```

The IN and NOT IN Predicates:

IN:

The arithmetic operator(=) compares a single value to another single value. In case a value needs to be compared to a list of values Then the IN predicate is used. The IN predicate helps reduce the need to use multiple OR conditions.

Eg. List out the students whose address is LATUR , NANDED, PUNE.

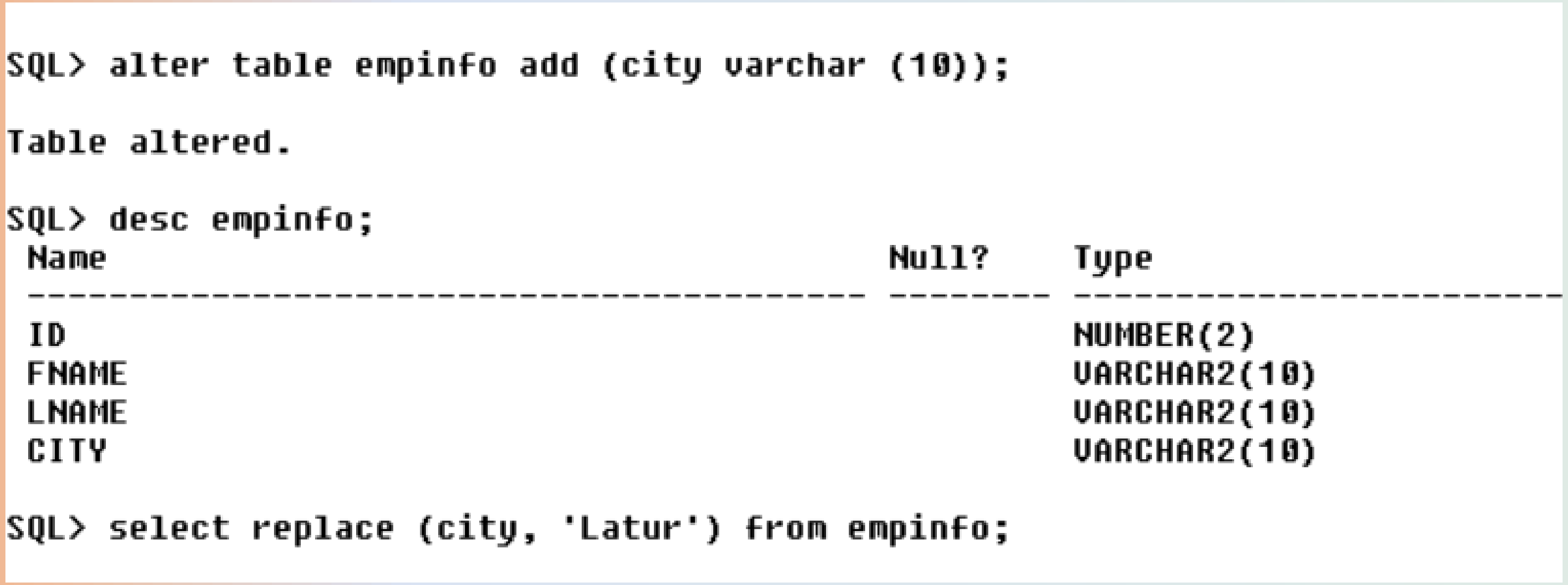
```
SQL> SELECT ROLLNO, NAME, ADDRESS from STUD
        WHERE
        ADDRESS IN(‘LATUR’, ‘PUNE’, ‘NANDED’);
```

NOT IN

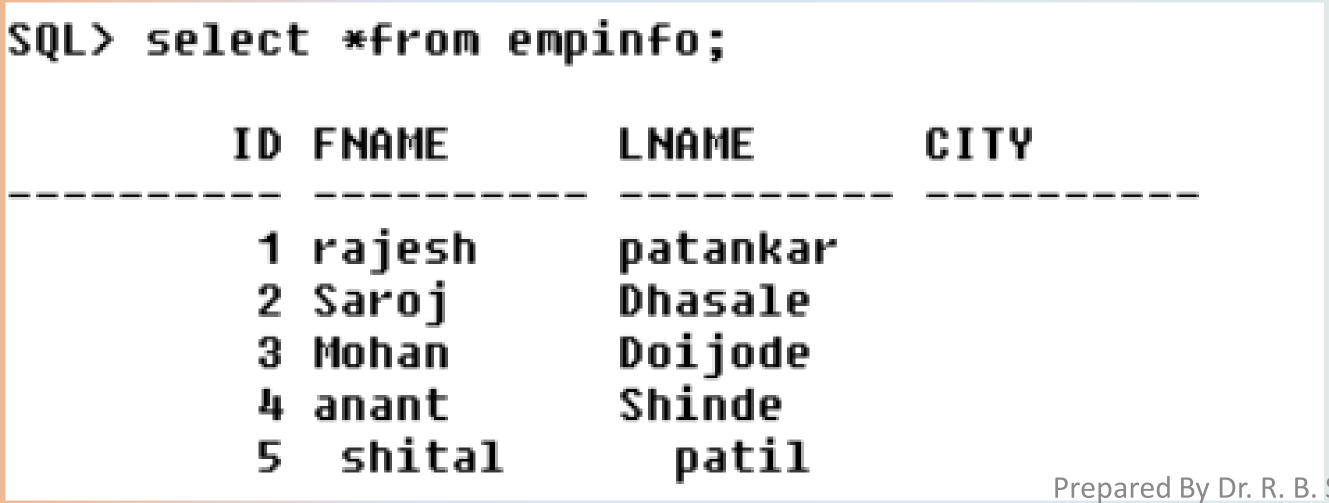
- The NOT IN predicate is the opposite of the IN predicate. This will select all the rows where values do not match the values in the list.
- Eg. List out the students details who are not belongs to PUNE, NANDED and LATUR. Consider the tablel STUD.

```
SQL> Select ROLLNO, NAME, ADDRESS
        FROM STUD
        WHERE ADDRESS NOT IN
        (‘Latur’, ‘Nanded’, ‘Pune’);
```

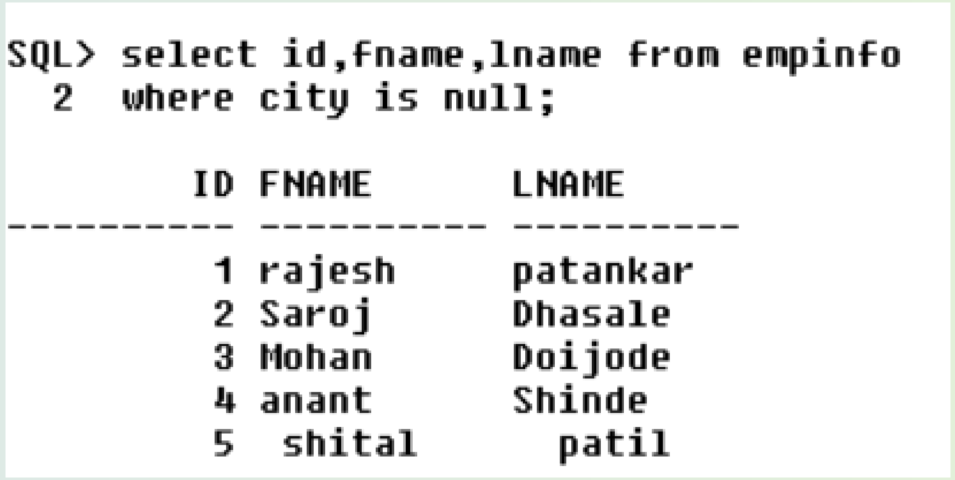
Is null



```
SQL> alter table empinfo add (city varchar (10));

Table altered.

SQL> desc empinfo;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                                 NUMBER(2)
 FNAME                                              VARCHAR2(10)
 LNAME                                              VARCHAR2(10)
 CITY                                               VARCHAR2(10)

SQL> select replace (city, 'Latur') from empinfo;
```



```
SQL> select *from empinfo;

        ID FNAME      LNAME      CITY
---------- ---------- ---------- ----------
         1 rajesh     patankar
         2 Saroj      Dhasale
         3 Mohan      Doijode
         4 anant      Shinde
         5  shital      patil
```



```
SQL> select id,fname,lname from empinfo
  2  where city is null;

        ID FNAME      LNAME
---------- ---------- ----------
         1 rajesh     patankar
         2 Saroj      Dhasale
         3 Mohan      Doijode
         4 anant      Shinde
         5  shital      patil
```

SQL FUNCTIONS

- Oracle SQL supplies a rich library of in-built functions which can be employed for various tasks. The essential capabilities of a functions can be the case conversion of strings, in-string or substring operations, mathematical computations on numeric data, and date operations on date type values. SQL Functions optionally take arguments from the user and mandatorily return a value.
- On a broader category, there are two types of functions :-

On a broader category, there are two types of functions :-

- **Single Row functions** - Single row functions are the one who work on single row and return one output per row. For example, length and case conversion functions are single row functions.
- **Multiple Row functions** - Multiple row functions work upon group of rows and return one result for the complete set of rows. They are also known as Group Functions.

Single Row functions -

- Single row functions are the one who work on single row and return one output per row. For example, length and case conversion functions are single row functions.

Multiple Row functions -

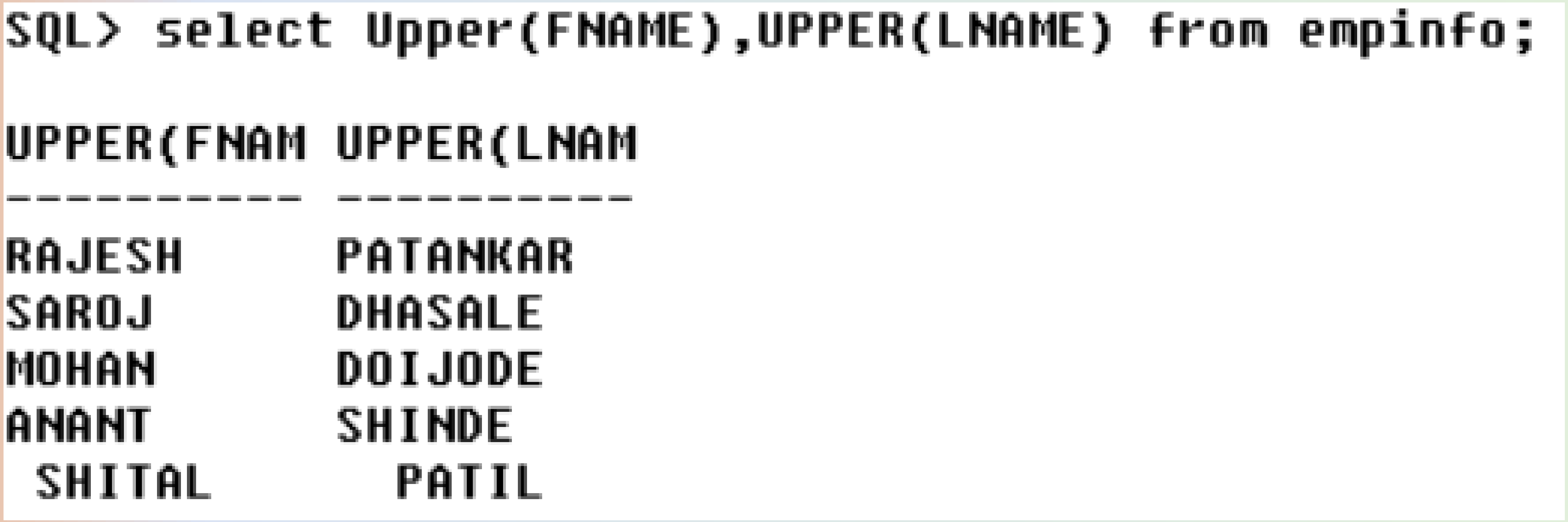
- Multiple row functions work upon group of rows and return one result for the complete set of rows. They are also known as Group Functions.

Single row functions

- Single row functions can be character functions, numeric functions, date functions, and conversion functions. Note that these functions are used to manipulate data items. These functions require one or more input arguments and operate on each row, thereby returning one output value for each row. Argument can be a column, literal or an expression. Single row functions can be used in SELECT statement, WHERE and ORDER BY clause. Single row functions can be -

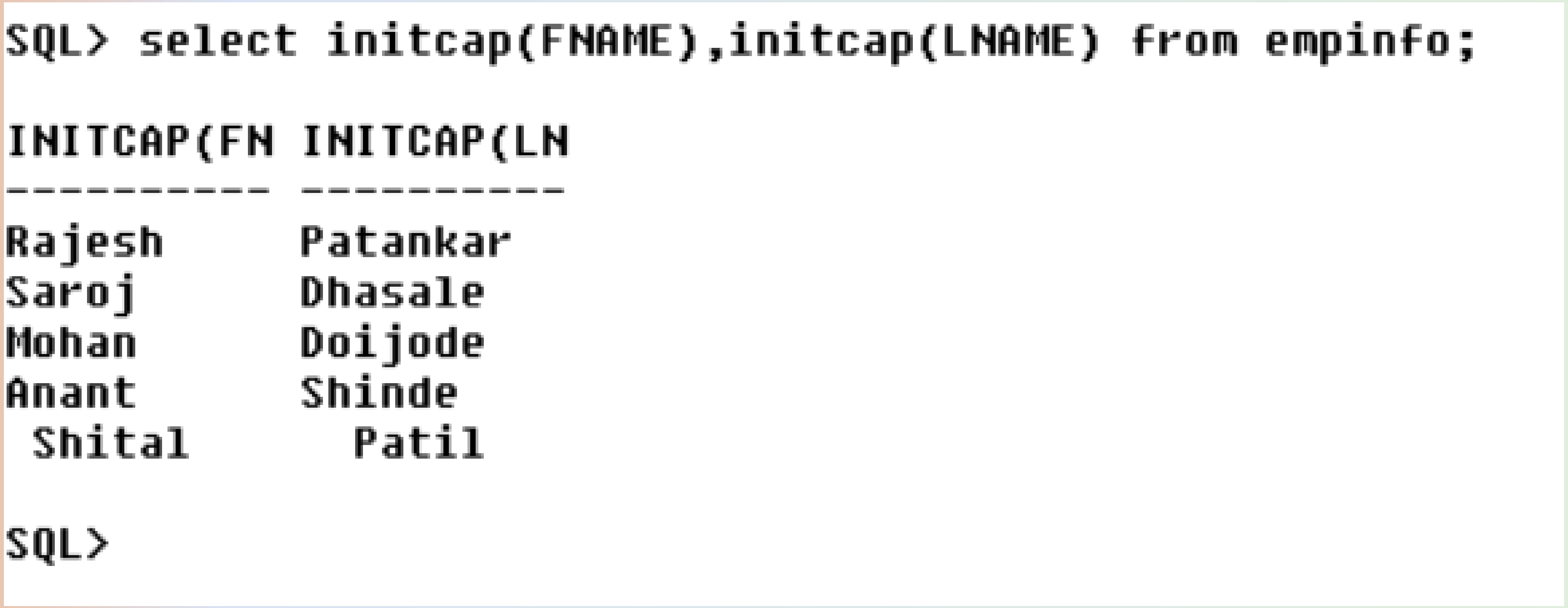
- **General functions** - Usually contains NULL handling functions. The functions under the category are NVL, NVL2, NULLIF, COALESCE, CASE, DECODE.
- **Case Conversion functions** - Accepts character input and returns a character value. Functions under the category are UPPER, LOWER and INITCAP.
  - UPPER function converts a string to upper case.
  - LOWER function converts a string to lower case.
  - INITCAP function converts only the initial alphabets of a string to upper case.
- **Character functions** - Accepts character input and returns number or character value. Functions under the category are CONCAT, LENGTH, SUBSTR, INSTR, LPAD, RPAD, TRIM and REPLACE.
  - CONCAT function concatenates two string values.
  - LENGTH function returns the length of the input string.
  - SUBSTR function returns a portion of a string from a given start point to an end point.
  - LPAD and RPAD functions pad the given string upto a specific length with a given character.
  - TRIM function trims the string input from the start or end.

UPPER



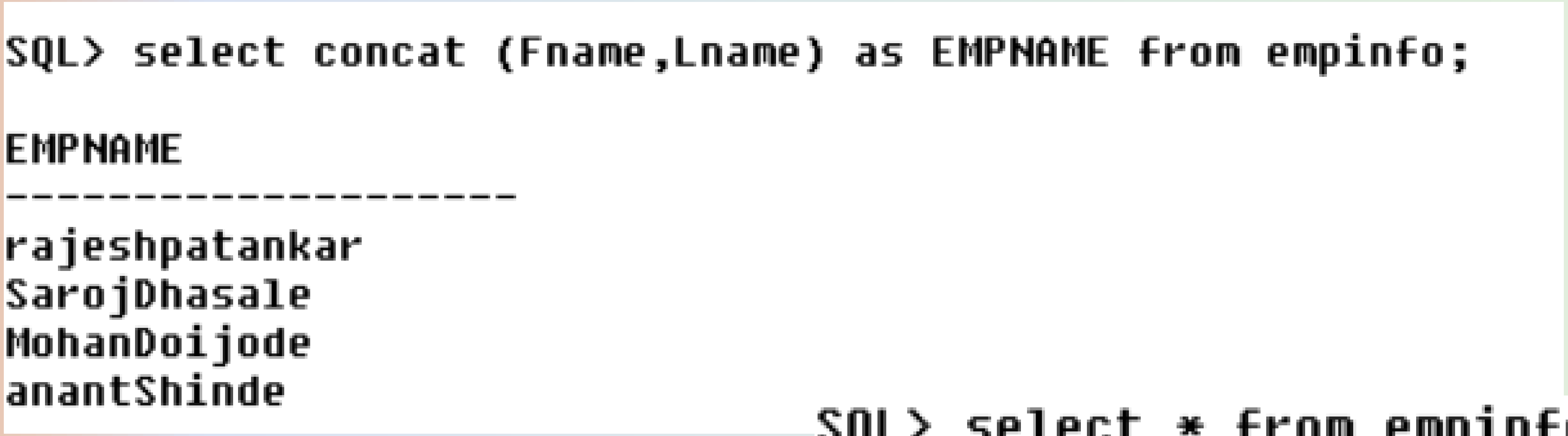
```
SQL> select Upper(FNAME),UPPER(LNAME) from empinfo;

UPPER(FNAM UPPER(LNAM
---------- ----------
RAJESH     PATANKAR
SAROJ      DHASALE
MOHAN      DOIJODE
ANANT      SHINDE
 SHITAL      PATIL
```

INITCAP

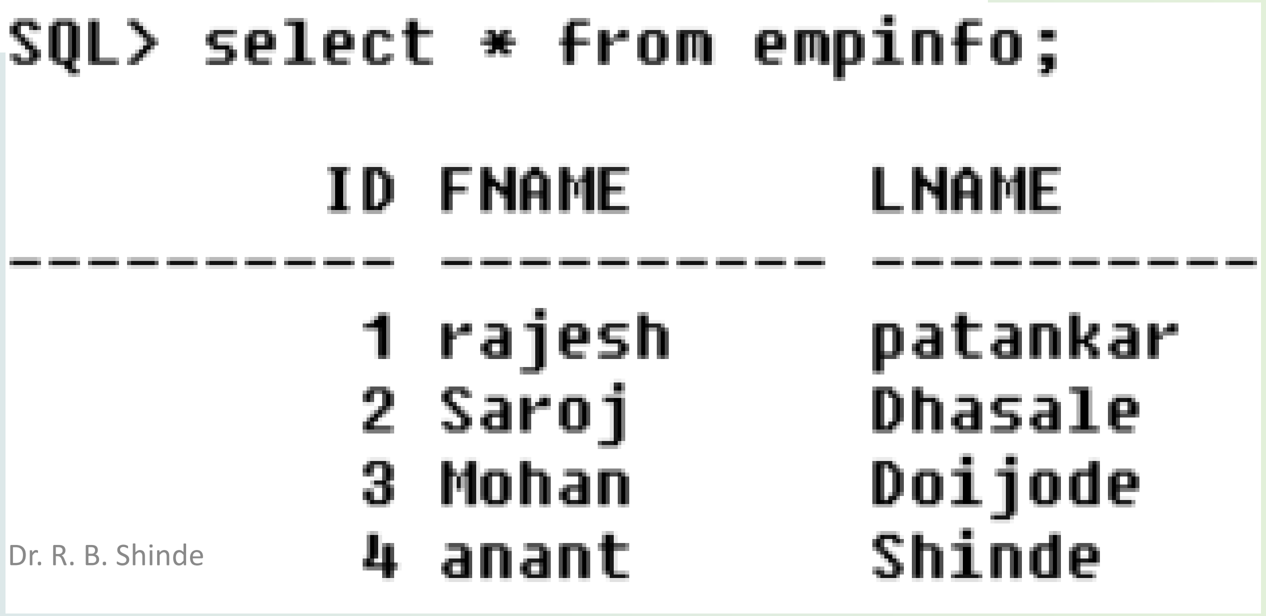


```
SQL> select initcap(FNAME),initcap(LNAME) from empinfo;

INITCAP(FN INITCAP(LN
---------- ----------
Rajesh     Patankar
Saroj      Dhasale
Mohan      Doijode
Anant      Shinde
 Shital      Patil

SQL>
```

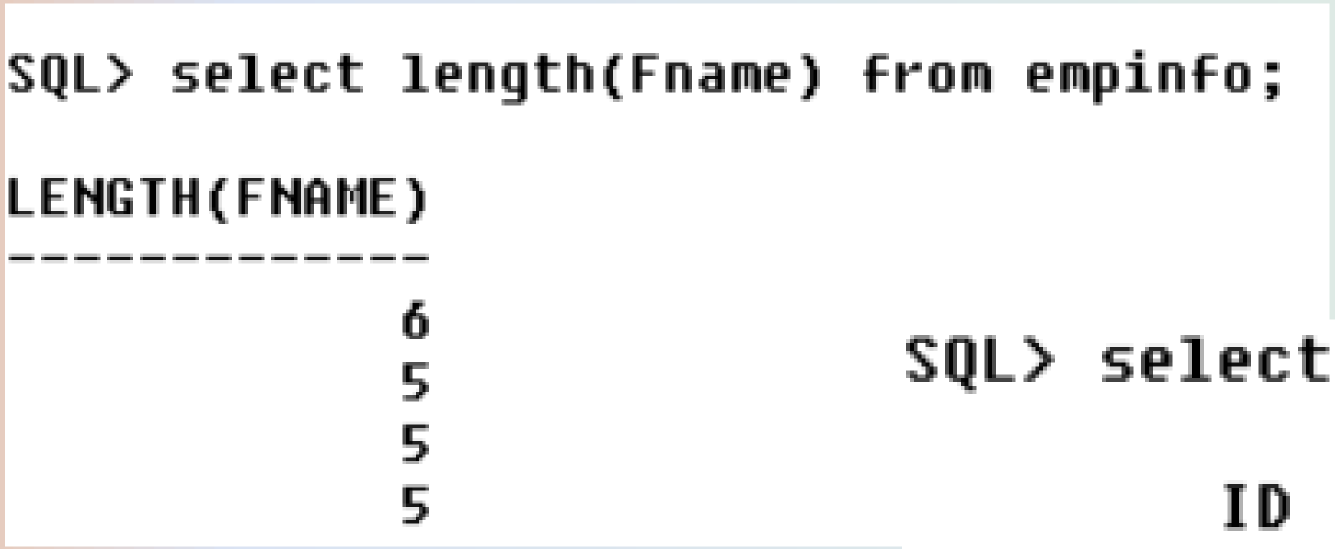
CONCATE FUNCTION



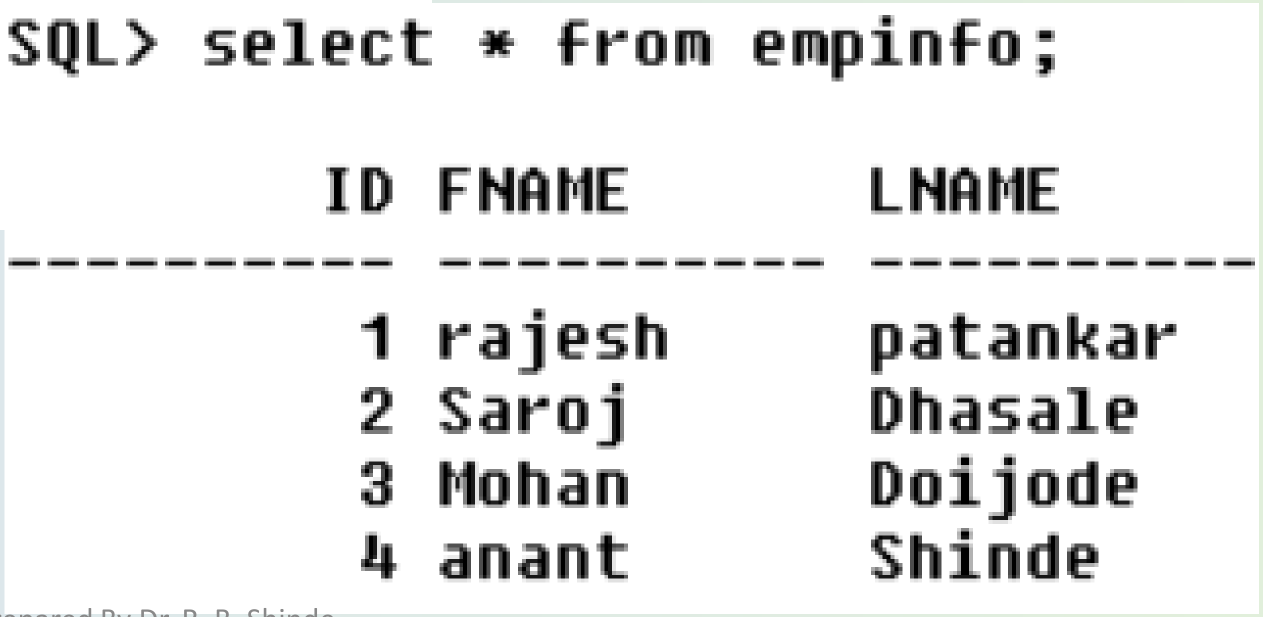
```
SQL> select concat (Fname,Lname) as EMPNAME from empinfo;

EMPNAME
--------------------
rajeshpatankar
SarojDhasale
MohanDoijode
anantShinde
```



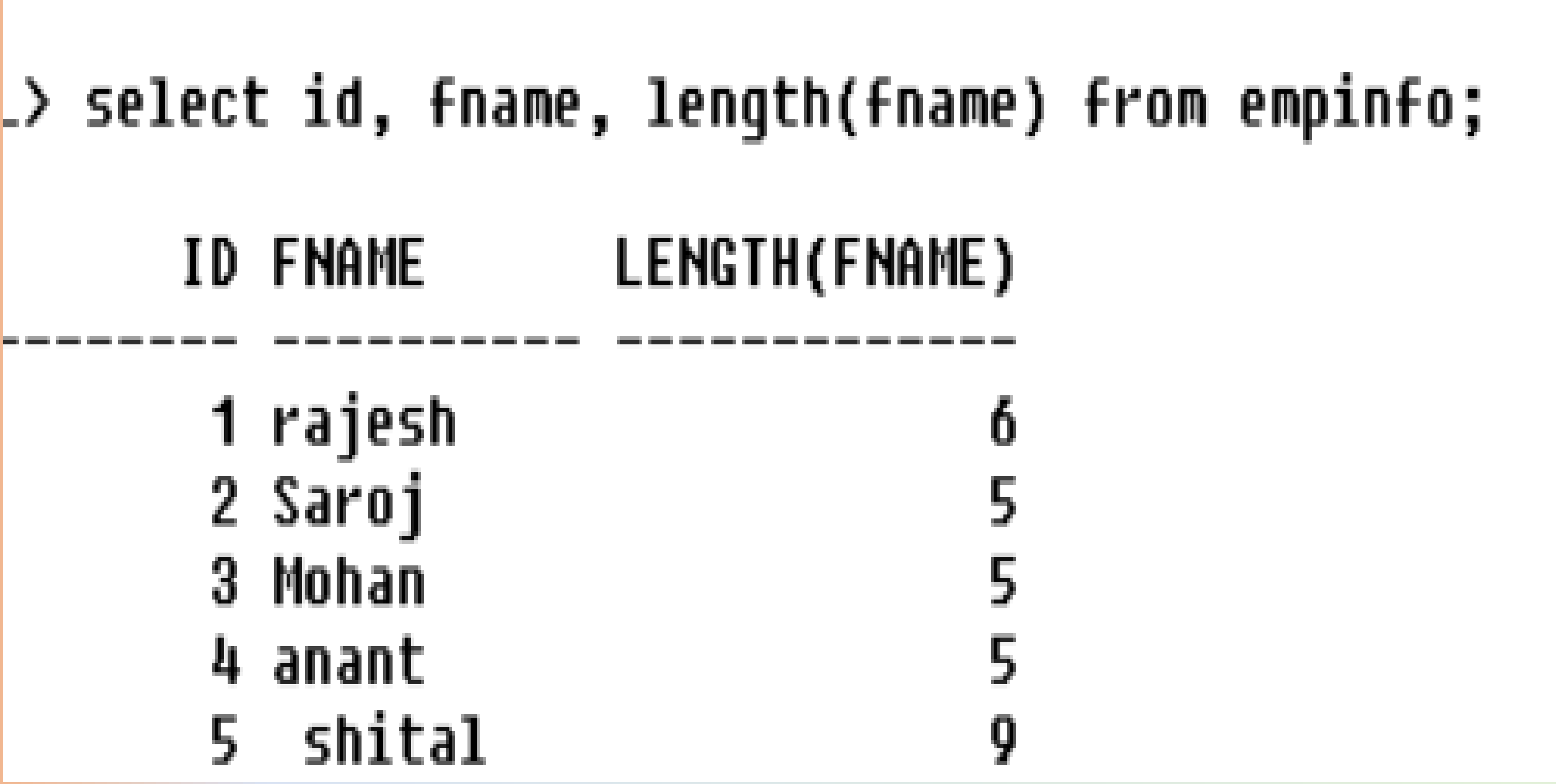
```
SQL> select * from empinfo;

        ID FNAME      LNAME
---------- ---------- ----------
         1 rajesh     patankar
         2 Saroj      Dhasale
         3 Mohan      Doijode
         4 anant      Shinde
```



```
SQL> select length(Fname) from empinfo;

LENGTH(FNAME)
-------------
            6
            5
            5
            5
```

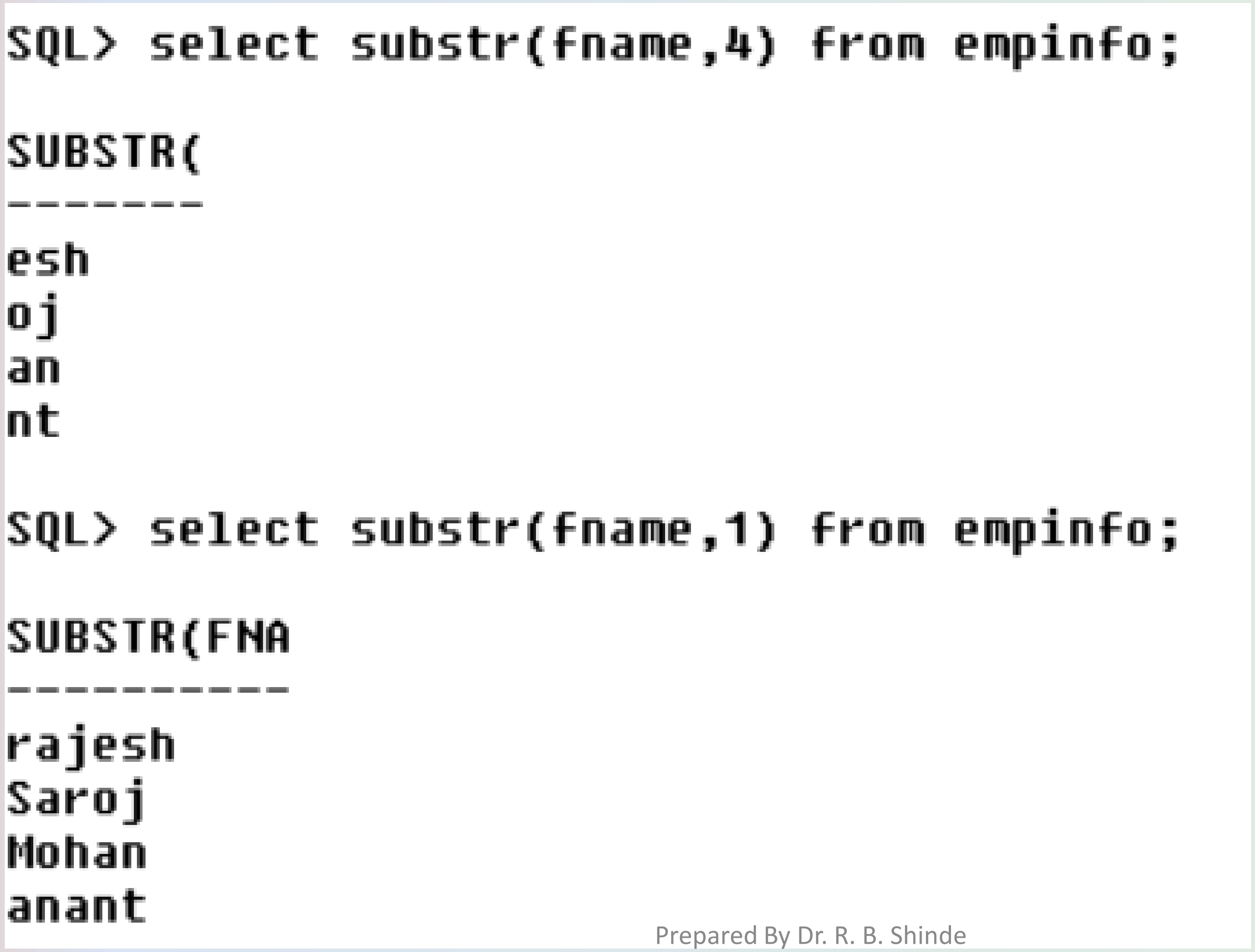


```
SQL> select * from empinfo;

        ID FNAME      LNAME
---------- ---------- ----------
         1 rajesh     patankar
         2 Saroj      Dhasale
         3 Mohan      Doijode
         4 anant      Shinde
```



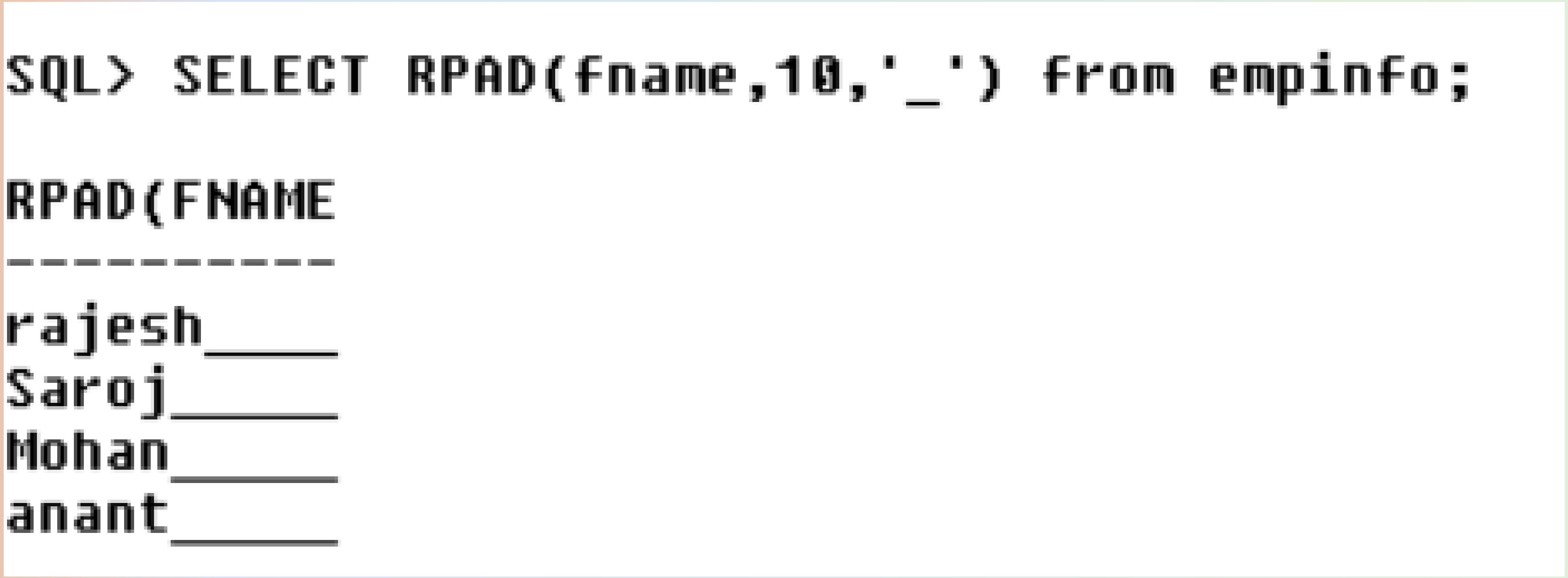
```
> select id, fname, length(fname) from empinfo;

        ID FNAME      LENGTH(FNAME)
---------- ---------- -------------
         1 rajesh                 6
         2 Saroj                  5
         3 Mohan                  5
         4 anant                  5
         5    shital              9
```

SUBSTR



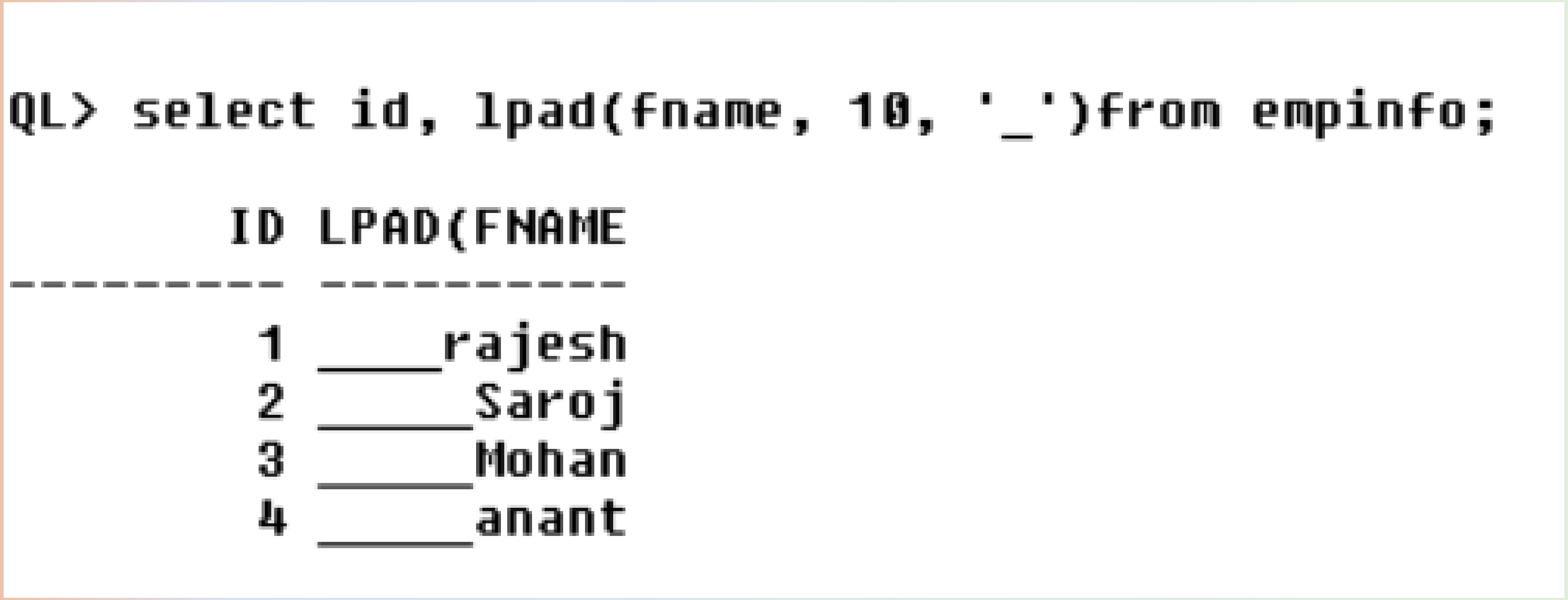
```
SQL> select substr(fname,4) from empinfo;

SUBSTR(
-------
esh
oj
an
nt

SQL> select substr(fname,1) from empinfo;

SUBSTR(FNA
----------
rajesh
Saroj
Mohan
anant
```

RPAD



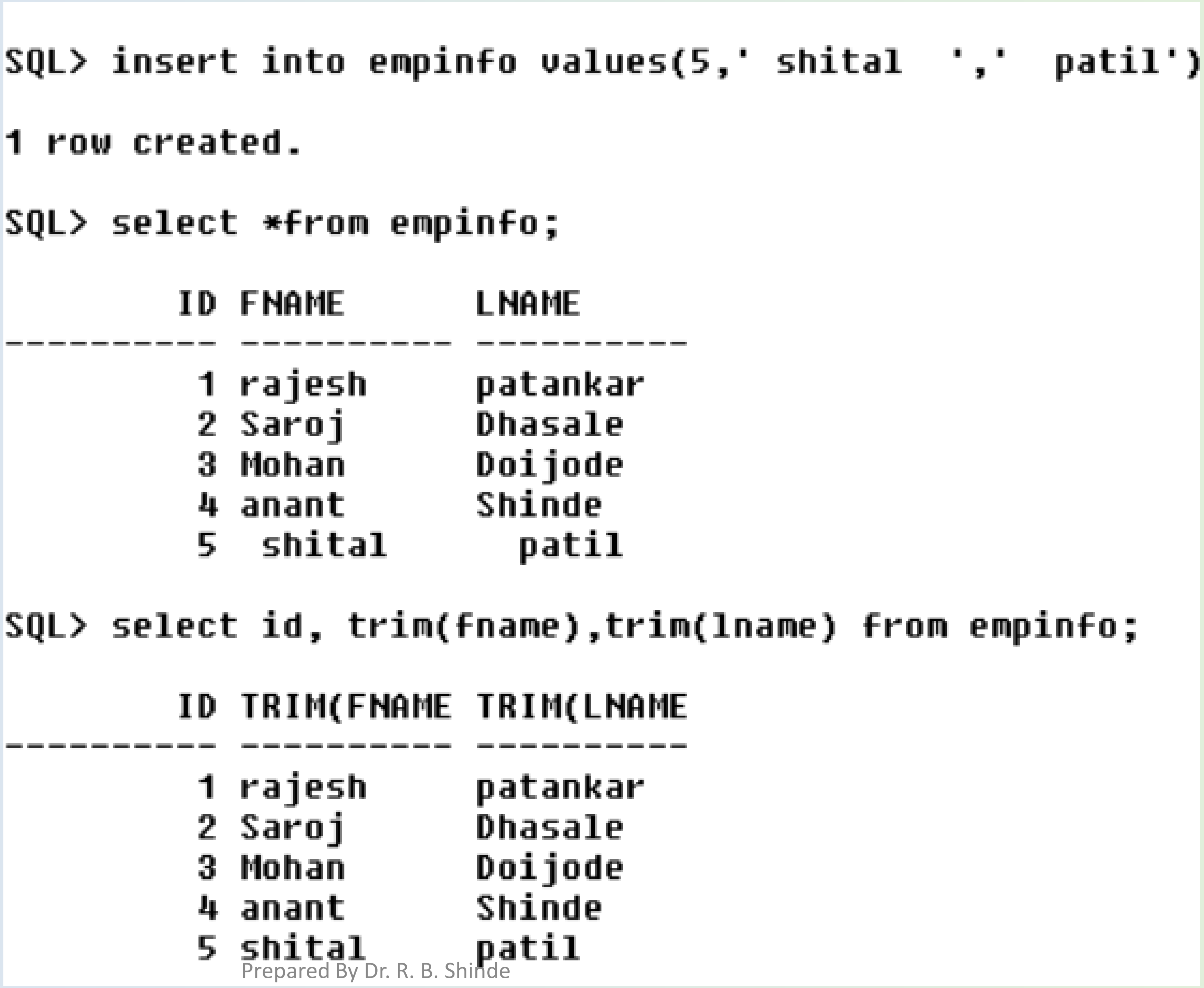
```
SQL> SELECT RPAD(fname,10,'_') from empinfo;

RPAD(FNAME
----------
rajesh____
Saroj_____
Mohan_____
anant_____
```

LPAD



```
QL> select id, lpad(fname, 10, '_')from empinfo;

        ID LPAD(FNAME
---------- ----------
         1 ____rajesh
         2 _____Saroj
         3 _____Mohan
         4 _____anant
```

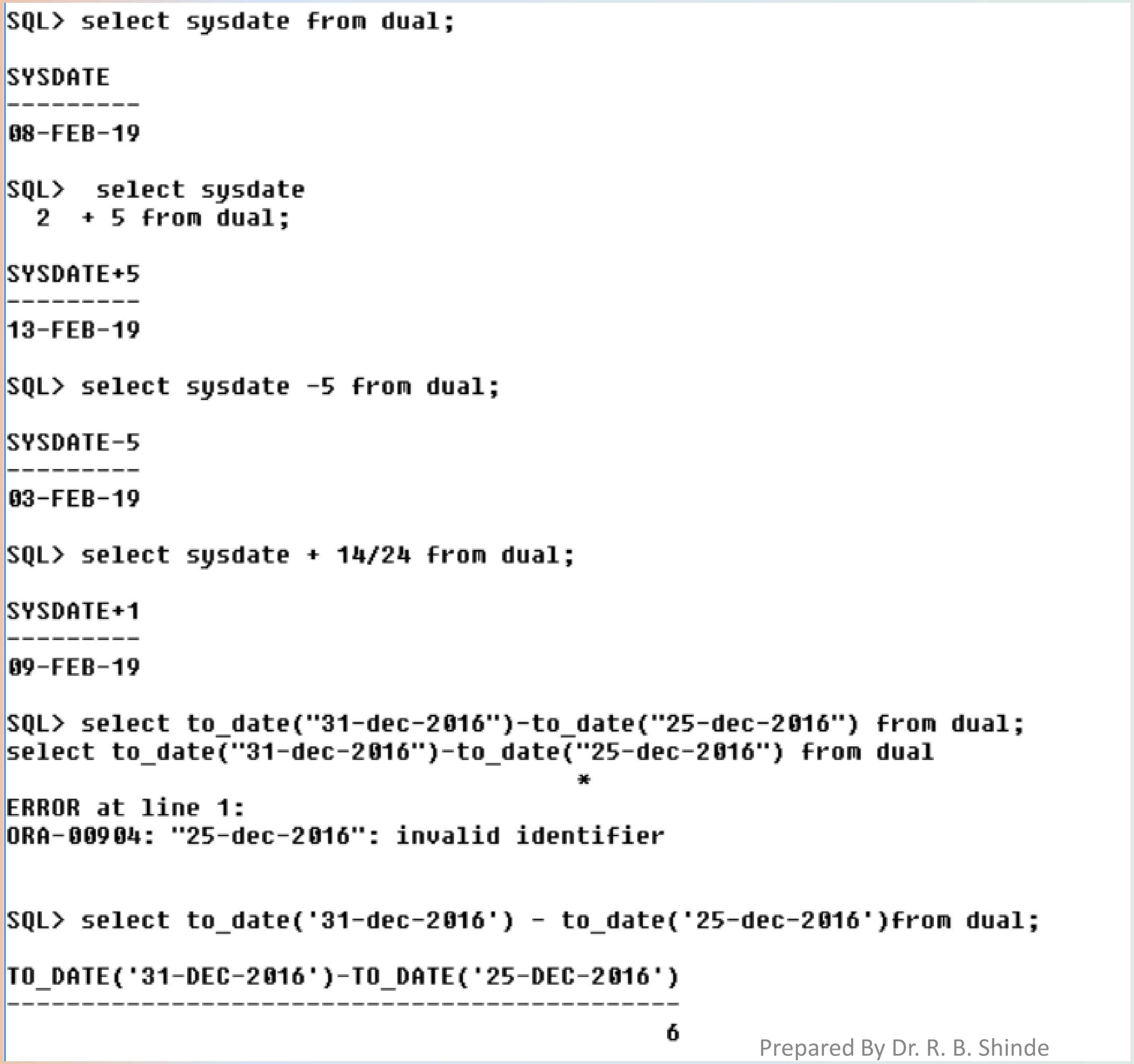
TRIM

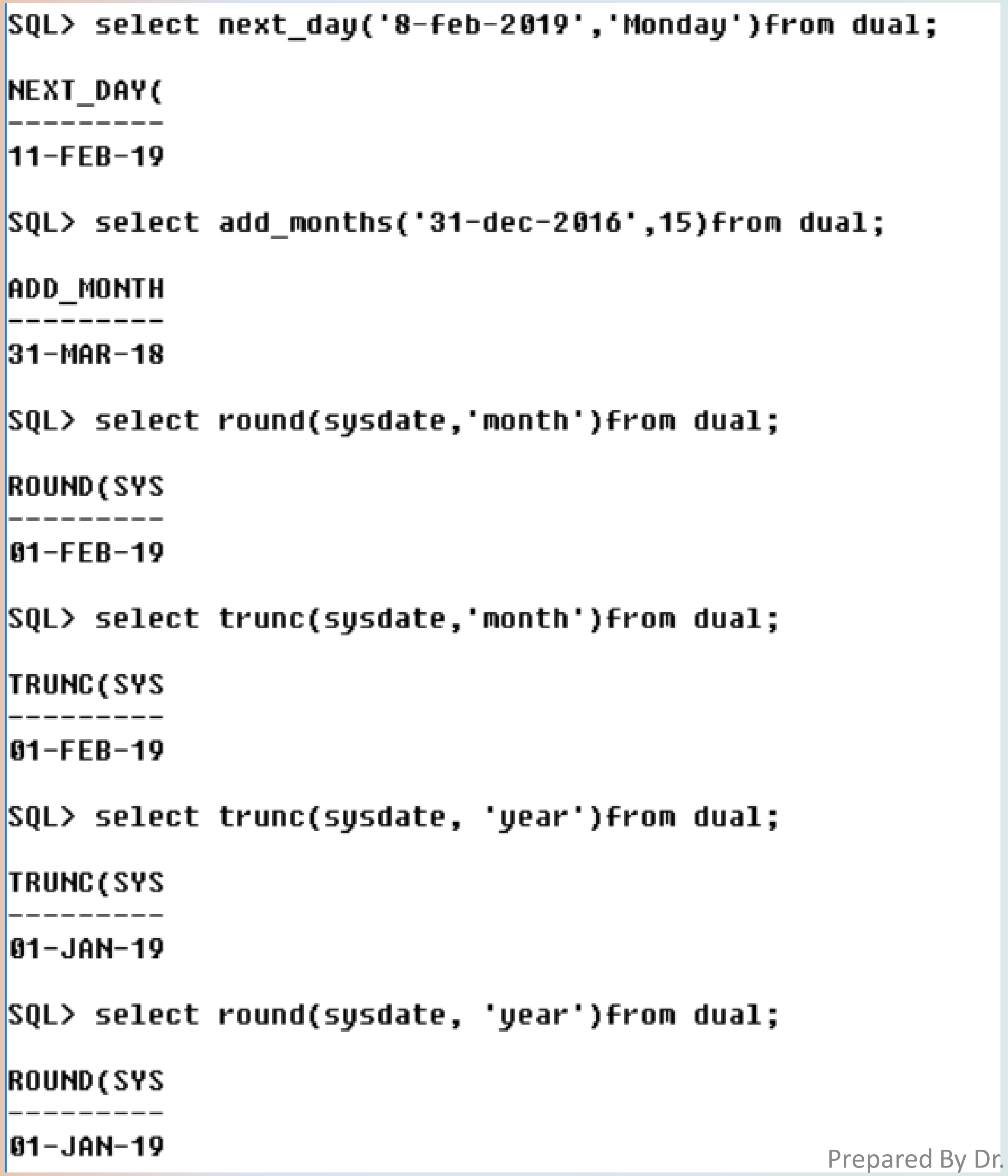


```
SQL> insert into empinfo values(5,' shital  ','  patil')

1 row created.

SQL> select *from empinfo;

        ID FNAME      LNAME
---------- ---------- ----------
         1 rajesh     patankar
         2 Saroj      Dhasale
         3 Mohan      Doijode
         4 anant      Shinde
         5  shital      patil

SQL> select id, trim(fname),trim(lname) from empinfo;

        ID TRIM(FNAME TRIM(LNAME
---------- ---------- ----------
         1 rajesh     patankar
         2 Saroj      Dhasale
         3 Mohan      Doijode
         4 anant      Shinde
         5 shital     patil
```

Date functions

- Date arithmetic operations return date or numeric values. Functions under the category are MONTHS_BETWEEN, ADD_MONTHS, NEXT_DAY, LAST_DAY, ROUND and TRUNC.
- MONTHS_BETWEEN function returns the count of months between the two dates.
- ADD_MONTHS function add 'n' number of months to an input date.
- NEXT_DAY function returns the next day of the date specified.
- LAST_DAY function returns last day of the month of the input date.
- ROUND and TRUNC functions are used to round and truncates the date value.


```
SQL> select sysdate from dual;

SYSDATE
---------
08-FEB-19

SQL>  select sysdate
  2  + 5 from dual;

SYSDATE+5
---------
13-FEB-19

SQL> select sysdate -5 from dual;

SYSDATE-5
---------
03-FEB-19

SQL> select sysdate + 14/24 from dual;

SYSDATE+1
---------
09-FEB-19

SQL> select to_date("31-dec-2016")-to_date("25-dec-2016") from dual;
select to_date("31-dec-2016")-to_date("25-dec-2016") from dual
                                       *
ERROR at line 1:
ORA-00904: "25-dec-2016": invalid identifier


SQL> select to_date('31-dec-2016') - to_date('25-dec-2016')from dual;

TO_DATE('31-DEC-2016')-TO_DATE('25-DEC-2016')
---------------------------------------------
                                            6
```



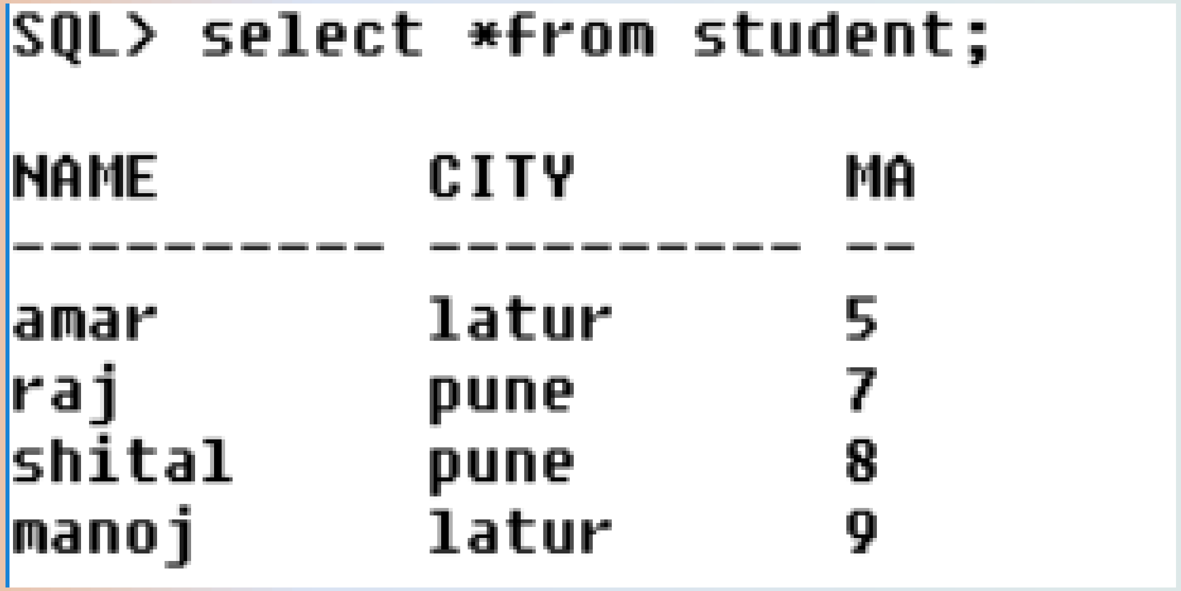
```
SQL> select next_day('8-feb-2019','Monday')from dual;

NEXT_DAY(
---------
11-FEB-19

SQL> select add_months('31-dec-2016',15)from dual;

ADD_MONTH
---------
31-MAR-18

SQL> select round(sysdate,'month')from dual;

ROUND(SYS
---------
01-FEB-19

SQL> select trunc(sysdate,'month')from dual;

TRUNC(SYS
---------
01-FEB-19

SQL> select trunc(sysdate, 'year')from dual;

TRUNC(SYS
---------
01-JAN-19

SQL> select round(sysdate, 'year')from dual;

ROUND(SYS
---------
01-JAN-19
```

- **Number functions** - Accepts numeric input and returns numeric values. Functions under the category are ROUND, TRUNC, and MOD.
- ROUND and TRUNC functions are used to round and truncate the number value.
- MOD is used to return the remainder of the division operation between two numbers.

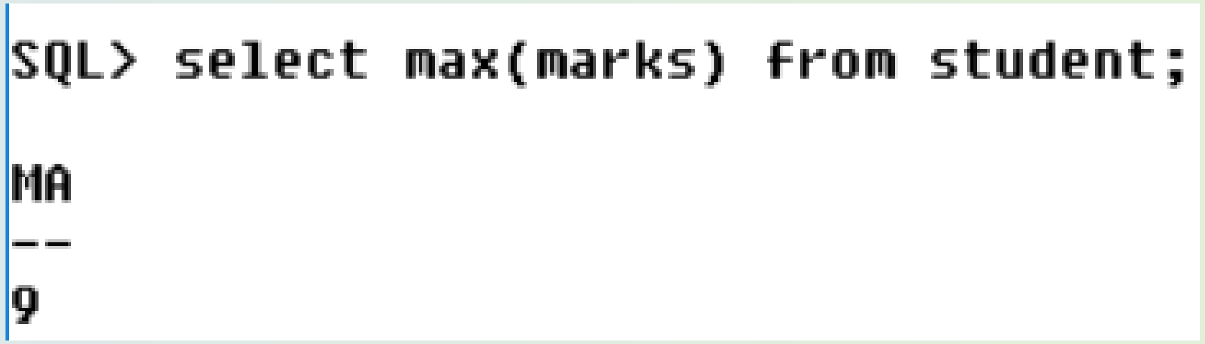
SQL Multi Row functions

- These functions are also calldded “Aggregate functions” (Or) Group functions.
- All these multi row functions will process multiple records.
- Applied on group of records and returns one value from the entire group.

- **MAX ( ) Function in SQL:-**
- Returns maximum value of a given expression.
- **Syntax** :
- Max(expr)



```
SQL> select *from student;

NAME       CITY       MA
---------- ---------- --
amar       latur      5
raj        pune       7
shital     pune       8
manoj      latur      9
```

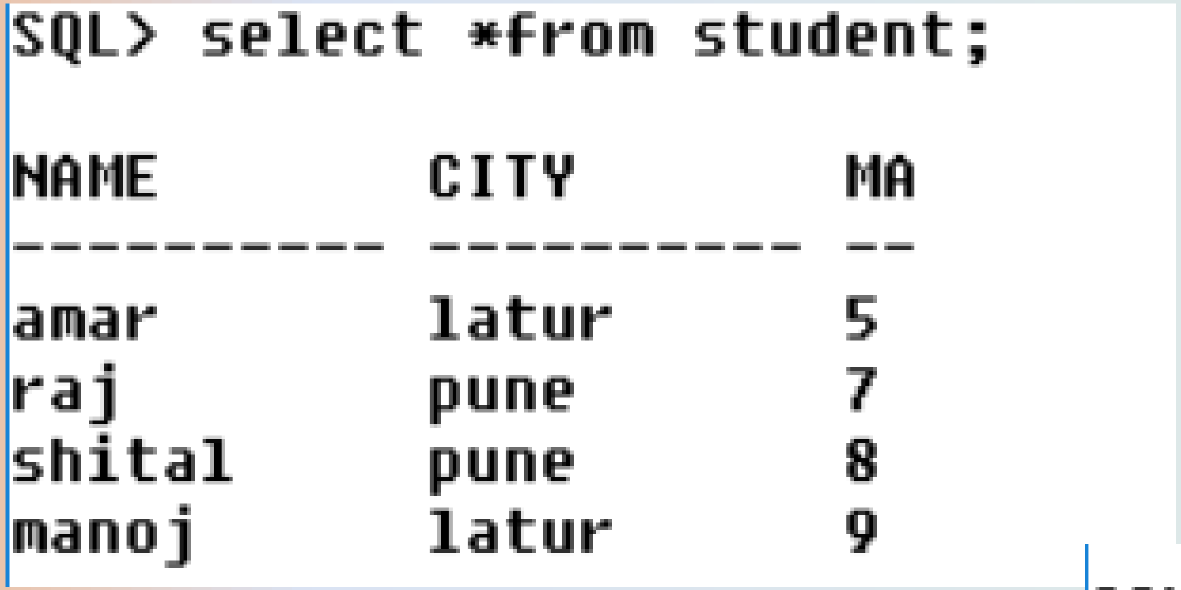


```
SQL> select max(marks) from student;

MA
--
9
```

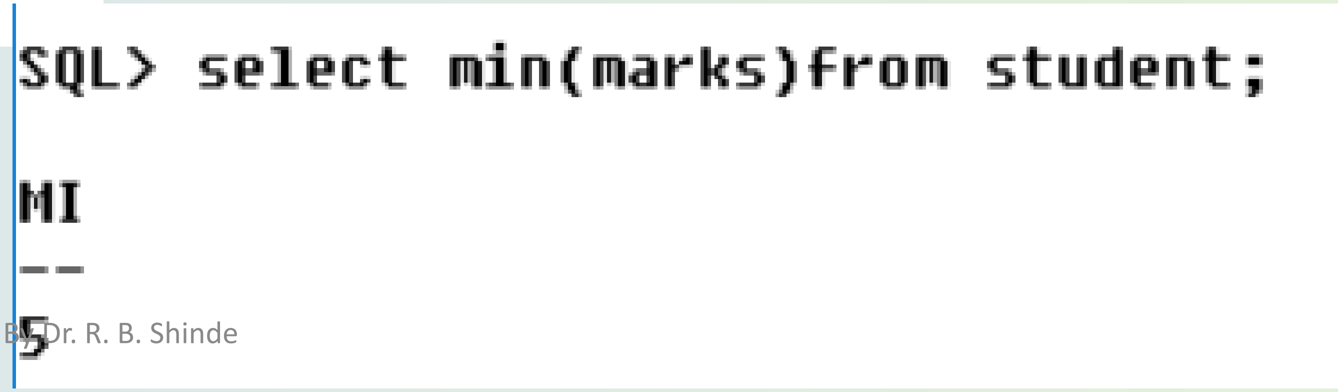
Min ( ) Function in SQL:-

Returns minimum value of a given expression

Syntax :-min(expr).



```
SQL> select *from student;

NAME       CITY       MA
---------- ---------- --
amar       latur      5
raj        pune       7
shital     pune       8
manoj      latur      9
```

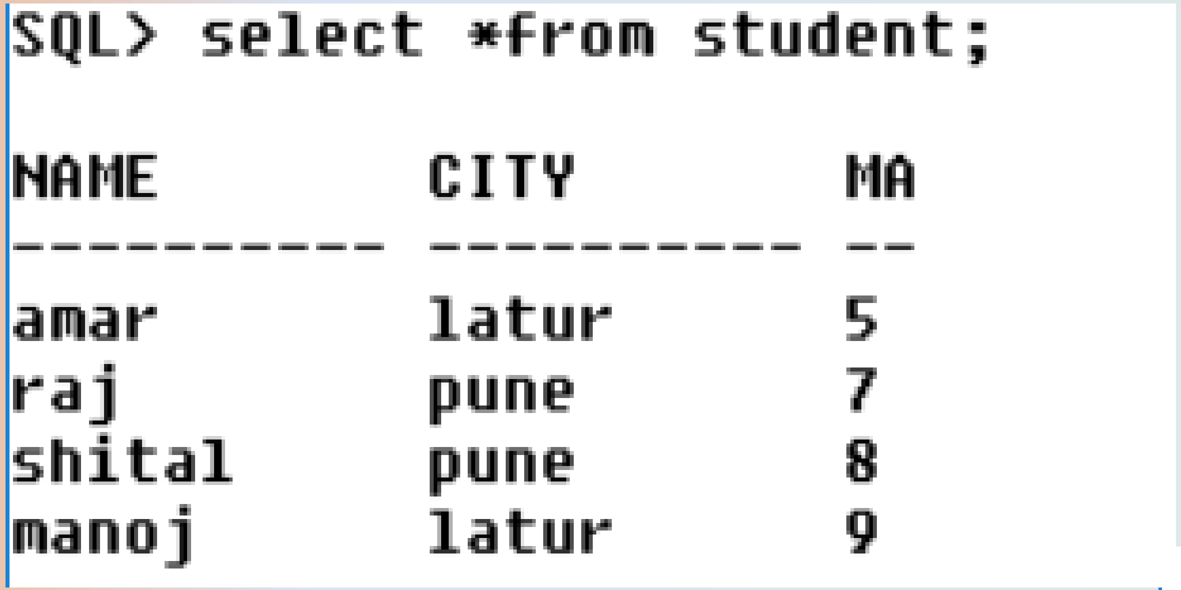


```
SQL> select min(marks)from student;

MI
--
5
```

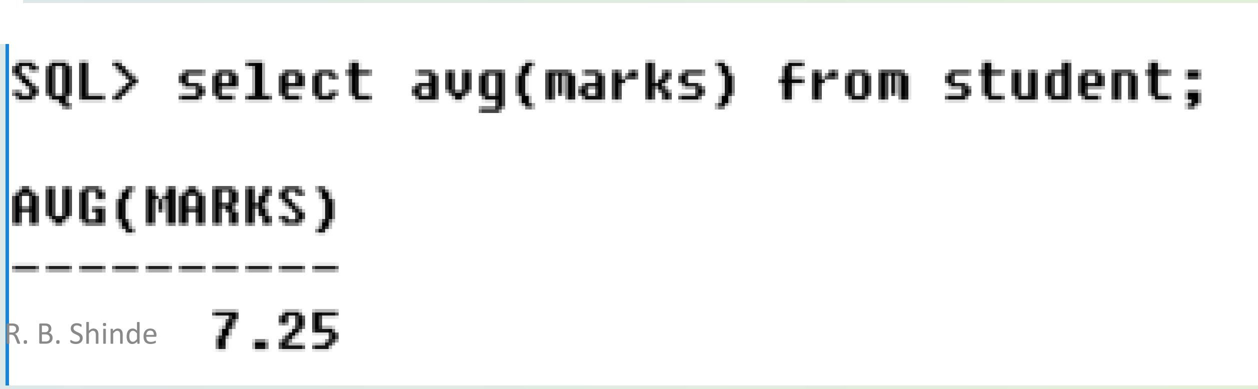
avg ( ) Function in SQL:-

It returns avg value of given expression

Select avg (sal) from emp

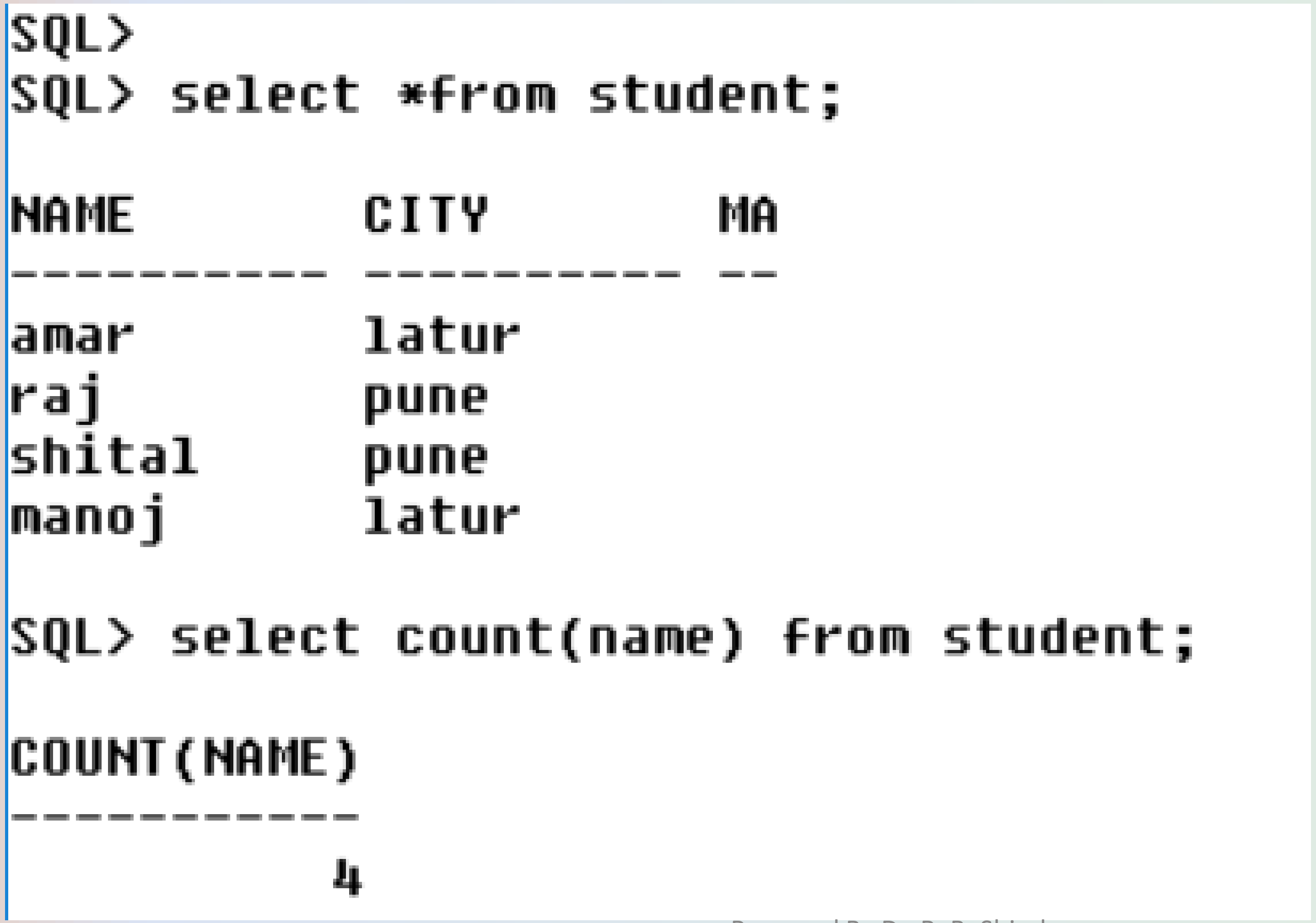


```
SQL> select *from student;

NAME        CITY        MA
----------- ----------- --
amar        latur       5
raj         pune        7
shital      pune        8
manoj       latur       9
```



```
SQL> select avg(marks) from student;

AVG(MARKS)
----------
      7.25
```

Count()


```
SQL>
SQL> select *from student;

NAME       CITY       MA
---------- ---------- --
amar       latur
raj        pune
shital     pune
manoj      latur

SQL> select count(name) from student;

COUNT(NAME)
-----------
          4
```