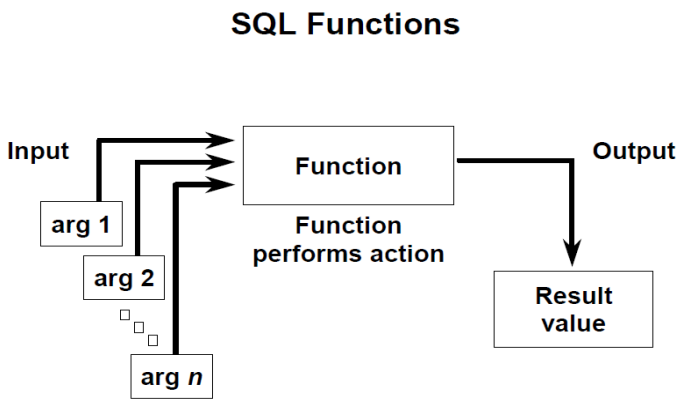
Department of Software Engineering Mehran University of Engineering and Technology, Jamshoro			
Course: SWE324 – Database Management and Administration			
Instructor	Ms. Shafiya Qadeer	**Practical/Lab No.**	06
Date	17/18-02-2021	**CLOs**	CLO-4: P3 & P4
Signature		**Assessment Score**	2 Marks

Topic	**To become familiar with Single row and Multiple row functions.**
Objectives	- SQL functions, types

Lab Discussion: Theoretical concepts and Procedural steps

SQL FUNCTIONS

- It is a set of SQL statements that accepts input parameters, perform actions and always return a value.
- Functions are a very powerful feature of SQL and can be used to do the following:
- Perform calculations on data
- Modify individual data items
- Manipulate output for groups of rows
- Format dates and numbers for display
- Convert column data types



TYPES OF SQL FUNCTIONS

- There are two types of functions:
  1. Single-row functions
  2. Multiple-row functions

1. SINGLE-ROW FUNCTIONS

These functions operate on single rows only and return one result per row.

There are different types of single-row functions:

```
function_name (column|expression, [arg1, arg2,...])
```

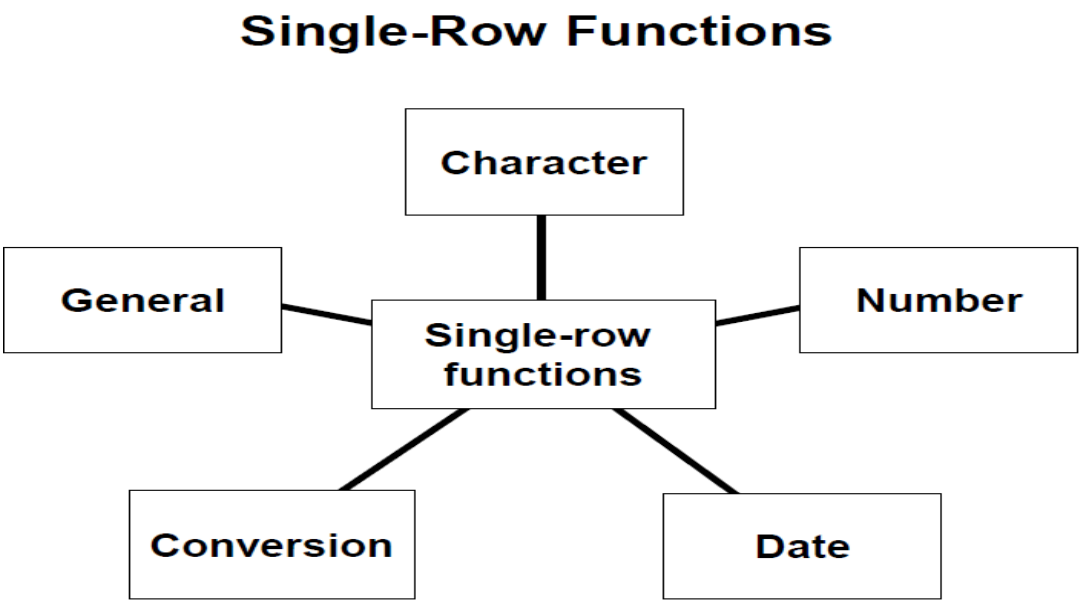
a. Character

b. Number

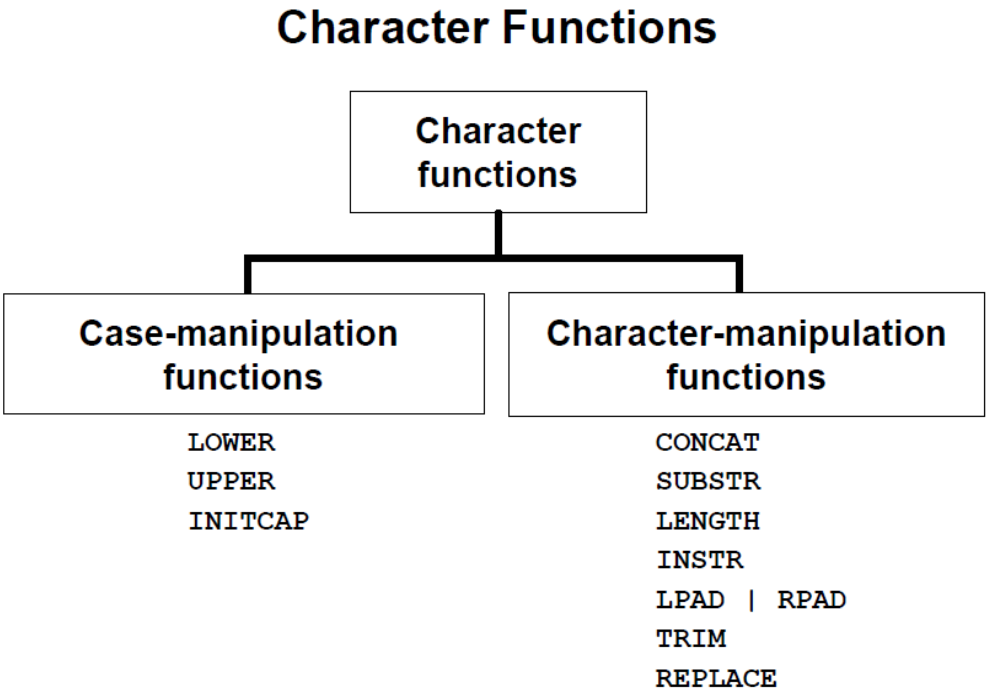
c. Date

d. Conversion

Types of Single-Row Functions



a. CHARACTER FUNCTIONS



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.

```
select UPPER(ename) , LOWER(job), INITCAP(ename) from emp;
```

Results Explain Describe Saved SQL History

UPPER(ENAME)	LOWER(JOB)	INITCAP(ENAME)
LEVI MARIO	tecnico	Levi Mario
CALVINO ANDREA	professore	Calvino Andrea
FUMAROLA LUDOVICO	segretario	Fumarola Ludovico
NERI ELENA	dirigente	Neri Elena

Character Manipulation 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.
- **INSTR** function returns numeric position of a character or a string in a given string.
- **LPAD** (left pad) and **RPAD** (right pad) are used to add padding characters to the left or right side of a string up to a given length.
- **TRIM** function removes all specified characters either from the beginning or the end of a string.

- **REPLACE:** It replaces a sequence of characters in a string with another set of characters.

```
Select ename ,  REPLACE( ename , 'LEVI' , 'John') AS "REplace function" FROM emp;
```

Results Explain Describe Saved SQL History

ENAME	REplace Function
LEVI MARIO	John MARIO
CALVINO ANDREA	CALVINO ANDREA

```
SELECT LENGTH(ename) AS "Name"  FROM emp;
```

Results Explain Describe Saved SQL History

Name
10
14
17
10

```
SELECT ename , substr(ename,1,6) FROM emp;
```

Results Explain Describe Saved SQL History

ENAME	SUBSTR(ENAME,1,6
LEVI MARIO	LEVI M
CALVINO ANDREA	CALVIN
FUMAROLA LUDOVICO	FUMARO
NERI ELENA	NERI E

```
Select LPAD(ename , 20 , '&') AS "LEFT PAD" , RPAD(ename , 20 , '&') AS "RIGHT PAD" from emp;
```

Results Explain Describe Saved SQL History

LEFT PAD	RIGHT PAD
&&&&&&&&&&LEVI MARIO	LEVI MARIO&&&&&&&&&&
&&&&&&CALVINO ANDREA	CALVINO ANDREA&&&&&&
&&&FUMAROLA LUDOVICO	FUMAROLA LUDOVICO&&&
&&&&&&&&&&NERI ELENA	NERI ELENA&&&&&&&&&&
&&&&&&&&SANTE ANDREA	SANTE ANDREA&&&&&&&&
&&&&&ESPOSITO ANDREA	ESPOSITO ANDREA&&&&&

b. General functions

- The general functions work with any data type and are mainly used to handle null values.

1. NVL (exp1,exp2)
2. NVL2 (exp1,exp2,exp3)
3. NULLIF (exp1,exp2)
4. COALESCE (exp1,exp2,exp3,.....,expn)

NVL Function

- The NVL function converts a null value to an actual value.
- Data types that can be used are date, character and number.

The syntax of NVL function is

```
NVL(expr1, expr2)
```

```
SELECT ename, comm, job , NVL(comm , 10) FROM emp;
```

Results Explain Describe Saved SQL History

ENAME	COMM	JOB	NVL(COMM,10)
LEVI MARIO	-	TECNICO	10
CALVINO ANDREA	-	PROFESSORE	10
FUMAROLA LUDOVICO	-	SEGRETARIO	10
NERI ELENA	-	DIRIGENTE	10

expr1 is the source value that contain null.

expr2 is the target value for converting null

NVL2 FUNCTION

- The NVL2 function takes three arguments as its input.

  If the expr1 is NOT NULL, NVL2 function returns expr2. If expr1 is NULL, then NVL2 returns expr3

```
SELECT ename, comm, job , NVL2(comm , sal * 12 , sal ) AS "NVL2 FUNCTION" FROM emp;
```

The syntax of NVL2 function is

```
NVL2(expr1,expr2,expr3)
```

Results Explain Describe Saved SQL History

ENAME	COMM	JOB	NVL2 FUNCTION
LEVI MARIO	-	TECNICO	1009
CALVINO ANDREA	-	PROFESSORE	1095
FUMAROLA LUDOVICO	-	SEGRETARIO	1501
NERI ELENA	-	DIRIGENTE	521

expr1: value that may contain null.

expr2: value returned if expr1 is not null.

expr3: value returned if expr1 is null.

Autocommit Display 10

```
select ename, sal, comm, nvl2(comm, sal+comm, sal) from emp
```

Results Explain Describe Saved SQL History

ENAME	SAL	COMM	NVL2(COMM,SAL+COMM,SAL)
KING	5000	-	5000
BLAKE	2850	-	2850
CLARK	2450	-	2450
JONES	2975	-	2975
JONES	2975	-	2975
SCOTT	3000	-	3000
FORD	3000	-	3000
SMITH	800	-	800
ALLEN	1600	300	1900
WARD	1250	500	1750

More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.01 seconds CSV Export

NULLIF FUNCTION

- The NULLIF function compares expression1 and expression2. If expression1 and expression2 are equal, the NULLIF function returns NULL. Otherwise, it returns the first expression which is expression1.

```
NULLIF( expression1, expression2 )
```

Parameters or Arguments

- expression1, expression2

  The expressions that will be compared. Values must be of the same datatype.

EXAMPLES:

- SELECT NULLIF('TechOnTheNet.com', 'TechOnTheNet.com')

  FROM DUAL;

  (returns NULL because values are the same)

SELECT NULLIF(12, 45) FROM DUAL;

(returns first value because values are different)

```
☑ Autocommit  Display 10
select empno, deptno, NULLIF(empno, deptno) FROM emp
```

EMPNO	DEPTNO	NULLIF(EMPNO,DEPTNO)
1	10	1
2	30	2
3	10	3
4	20	4
5	20	5
6	20	6
7	20	7
8	20	8
9	30	9
10	30	10
More than 10 rows available. Increase rows selector to view more rows.		

COALESCE FUNCTION

- COALESCE returns the first non-null expr in the expression list. You must specify at least two expressions. If all occurrences of expr evaluate to null, then the function returns null.

Syntax

```
COALESCE (expr1, expr2, ... exprn)
```

In the syntax:

expr1	returns this expression if it is not null
expr2	returns this expression if the first expression is null and this expression is not null
exprn	returns this expression if the preceding expressions are null

```
SELECT ename, COALESCE(sal, comm)
FROM emp;
```

Results Explain Describe Saved SQL History

ENAME	COALESCE(SAL,COMM)
LEVI MARIO	1009
CALVINO ANDREA	1095
FUMAROLA LUDOVICO	1501
NERI ELENA	521
SANTE ANDREA	1663

DECODE Function

- The DECODE function decodes an expression in a way similar to the IF-THEN-ELSE logic used in various languages.
- DECODE compares expression or column to each search value one by one. If expression or column is equal to a search, then Oracle Database returns the corresponding result.
- If no match is found, then Oracle returns default. If default is omitted, then Oracle returns null.
    - **SYNTAX:**

```
DECODE( expression , search , result [, search , result]... [, default] )
```

- **Parameters or Arguments**
- **expression/column:** The value to compare.
- **search:** The value that is compared against expression.
- **result:** The value returned, if expression is equal to search.
- **default:** Optional. If no matches are found, the DECODE function will return default. If default is omitted, then the DECODE function will return null (if no matches are found).

```
SELECT
DECODE('Superman', 'Superman', 'True, Strings are equal', 'False, Strings are not equal')
FROM dual;
```

```
SELECT ename, job, sal,
DECODE(job, 'TECNICO', 1.10*sal,
'PROFESSORE', 1.15*sal,
'INGEGNERE', 1.20*sal,
sal) AS "SALARY"
FROM emp;
```

Results Explain Describe Saved SQL History

ENAME	JOB	SAL	SALARY
LEVI MARIO	TECNICO	1009	1109.9
CALVINO ANDREA	PROFESSORE	1095	1259.25
FUMAROLA LUDOVICO	SEGRETARIO	1501	1501
NERI ELENA	DIRIGENTE	521	521
SANTE ANDREA	DIRIGENTE	1663	1663

c. NUMBER FUNCTION

- Accepts numeric input and returns numeric values.
- Functions under the category are

  1. ROUND
  2. TRUNC
  3. 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.

ROUND FUNCTION:

- Rounds value to a specified decimal.
- To round off decimals:
- Find the place value you want (the "rounding digit") and look at the digit just to the right of it.
- If that digit is less than 5, do not change the rounding digit but drop all digits to the right of it.
- If that digit is greater than or equal to five, add one to the rounding digit and drop all digits to the right of it.
- **Syntax:**
  - ROUND(number, places)

```
SELECT ROUND(1.3456, 2) FROM DUAL;
```

Results Explain Describe Saved SQL

ROUND(1.3456,2)
1.35

- The **DUAL table** is a special one-row, one-column table present by default in Oracle and other database installations. In Oracle, the table has a single column called DUMMY that has a value of 'X'.
- **TRUNC FUNCTION:**
  - Truncates value to a specified decimal.
  - **Syntax:**

    ROUND(number, places)

```
SELECT TRUNC(1.3456, 2) FROM DUAL;
```

Results Explain Describe Saved SQL His

TRUNC(1.3456,2)
1.34

MOD FUNCTION

- It returns remainder of division.
- **Syntax:**

  MOD(dividend, divisor)

```
SELECT MOD(1700, 3) FROM DUAL;
```

Results Explain Describe Saved SQ

MOD(1700,3)
2

d. DATE FUNCTION

- Date functions operate on date data type and returns a date value or numeric value.
- Functions under the category are:
- **SYSDATE** returns the current oracle database server date.
- **MONTHS_BETWEEN** function returns the number of months between the two dates.

```
SELECT SYSDATE FROM DUAL;
```

Results Explain Describe Save

SYSDATE
20-MAY-17

```
Syntax: months_between(date1,date2)
SELECT months_between(sysdate,hire_date) FROM EMPLOYEES:
SELECT months_between('01-JUL-2000', '23-JAN-2000') FROM DUAL;
```

- **ADD_MONTHS** function is used to add or subtract the number of calendar months to the given date.

```
SELECT ADD_MONTHS(SYSDATE, 2) FROM DUAL;
```

```
Syntax: add_months(date,n)
SELECT add_months(sysdate,3) FROM DUAL;
SELECT add_months(sysdate,-3) FROM DUAL;
SELECT add_months('01-JUL-2000', 3) FROM DUAL;
```

Results Explain Describe Saved SQL History

ADD_MONTHS(SYSDATE,2)
20-JUL-17

```
SELECT NEXT_DAY('20-May-2017', 'FRIDAY') FROM DUAL;
```

Results Explain Describe Saved SQL History

NEXT_DAY('20-MAY-2017','FRIDAY')
26-MAY-17

LAST_DAY function returns the last date of the specified month.

```
SELECT LAST_DAY(SYSDATE) FROM DUAL;
```

Results Explain Describe Saved SQL Hist

LAST_DAY(SYSDATE)
31-MAY-17

- **NEXT_DAY** function returns the date of the next day after specified date.
  - **<u>SYNTAX:</u>**
    - NEXT_DAY(date,'char')
    - The argument char must be a day of the week in the date language, either the full name or the abbreviation.

- **ROUND** function returns the date rounded to the specified format.
  - **<u>Syntax:</u>**

    Round(date [,'fmt']) where, fmt = format like 'MONTH', 'YEAR'

    If the fmt is omitted, date is rounded to the nearest day.

```
SELECT ROUND(To_DATE('5-MAY-2017'), 'MONTH') FROM DUAL;
```

Results Explain Describe Saved SQL History

ROUND(TO_DATE('5-MAY-2017','MONTH')
01-MAY-17

```
SELECT ROUND(To_DATE('20-MAY-2017'), 'MONTH') FROM DUAL;
```

Results Explain Describe Saved SQL History

ROUND(TO_DATE('20-MAY-2017','MONTH')
01-JUN-17

```
SELECT ROUND(To_DATE('20-MAY-2017'), 'YEAR') FROM DUAL;
```

Results Explain Describe Saved SQL History

ROUND(TO_DATE('20-MAY-2017','YEAR')
01-JAN-17

```
SELECT ROUND(To_DATE('20-OCT-2017'), 'YEAR') FROM DUAL;
```

Results Explain Describe Saved SQL History

ROUND(TO_DATE('20-OCT-2017','YEAR')
01-JAN-18

- **TRUNC** function returns the date truncated to the specified format.
  - **Syntax:**
    - Trunc(date [,'fmt'])

```
SELECT TRUNC(To_DATE('25-MAY-2017'), 'Month') FROM DUAL;
```

Results Explain Describe Saved SQL History

TRUNC(TO_DATE('1-MAY-2017'),'MONTH')
01-MAY-17

```
SELECT TRUNC(To_DATE('25-OCT-2017'), 'year') FROM DUAL;
```

Results Explain Describe Saved SQL History

TRUNC(TO_DATE('25-OCT-2017'),'YEAR')
01-JAN-17

Arithmetic with Dates

- Add or subtract a number to or from a date for a resultant date value.
  - date + number = date
  - date - number = date
- Subtract two dates to find the number of days between those dates.
  - date1 - date2 = number of days in between those dates
- Add hours to a date by dividing the number of hours by 24.
  - date + number/24 = date

```
SELECT To_DATE('20-MAY-2017') + 2 FROM DUAL;
```

Results Explain Describe Saved SQL History

TO_DATE('20-MAY-2017')+2
22-MAY-17

```
SELECT To_DATE('20-May-2017') + 36/24  FROM DUAL;
```

Results Explain Describe Saved SQL History

TO_DATE('20-MAY-2017')+36/24
21-MAY-17

```
SELECT To_DATE('30-MAY-2017')  - To_DATE('25-MAY-2017') AS "NUMBER OF DAYS" FROM DUAL;
```

Results Explain Describe Saved SQL History

NUMBER OF DAYS
5

Conversion Functions

- Conversion functions convert a value from one datatype to another.

1. Implicit datatype conversion

2. Explicit datatype conversion

1. Implicit Datatype Conversion

Oracle can automatically convert the following:

From	To
VARCHAR2 or CHAR	NUMBER
VARCHAR2 or CHAR	DATE
NUMBER	VARCHAR2
DATE	VARCHAR2

Explicit Data Type Conversion

Explicit datatype conversions are done by using the conversion functions. There are 3 types of Explicit Data Type Conversion

1. TO_CHAR

2. TO_NUMBER

3. TO_DATE

1. TO_CHAR FUNCTION:

Converts a number or a date value to a VARCHAR2 character string with format model fmt.

SYNTAX:

TO_CHAR(value, [format], [nls_language])

- **Parameters or Arguments:**

- Value can either be a number or date that will be converted to a string.
- Format is optional. This is the format that will be used to convert value to a string.
- nls_language is optional. This is the nls language used to convert value to a string.

```
SELECT ename, job ,
       TO_CHAR (hiredate, 'MONTH DD, YYYY') AS "HIRE_DATE"
         FROM emp;
```

Results Explain Describe Saved SQL History

ENAME	JOB	HIRE_DATE
LEVI MARIO	TECNICO	JUNE 09, 2081
CALVINO ANDREA	PROFESSORE	JUNE 09, 2081
FUMAROLA LUDOVICO	SEGRETARIO	JUNE 09, 2081
NERI ELENA	DIRIGENTE	JUNE 09, 2081
SANTE ANDREA	DIRIGENTE	JUNE 09, 2081
ESPOSITO ANDREA	ISPETTORE	JUNE 09, 2081

TO_NUMBER

- Converts a character string containing digits to a number with the optional format model fmt.
  - **<u>SYNTAX:</u>**

    TO_NUMBER(char, [fmt])

```
SELECT  To_NUMBER('97.13') + 25.5
FROM DUAL;
```

Results Explain Describe Saved SQL Histo

TO_NUMBER('97.13')+25.5
122.63

TO_DATE

- The function takes character values as input and returns formatted date equivalent of the same.

TO_DATE(string1, [format])

<u>Parameters or Arguments</u>

- string1 is the string that will be converted to a date.
- format is optional. This is the format that will be used to convert string1 to a date.

```
SELECT TO_DATE('January 15, 1989, 11:00 A.M.',  'Month dd, YYYY, HH:MI A.M.')
FROM DUAL;
```

Results Explain Describe Saved SQL History

TO_DATE('JANUARY15,1989,11:00A.M.','MONTHDD,YYYY,HH:MIA.M.')
15-JAN-89

Elements of the Date Format Model

YYYY	Full year in numbers
YEAR	Year spelled out
MM	Two-digit value for month
MONTH	Full name of the month
MON	Three-letter abbreviation of the month
DY	Three-letter abbreviation of the day of the week
DAY	Full name of the day of the week
DD	Numeric day of the month

DD “of” MONTH = 12 of JUNE

DDD = Day of year

HH = hour of day, MI = Minute , SS = Second

HH:MI:SS AM = 5:45:32 AM

Lab Tasks

- **Q1. Execute the following queries and display their outputs:**
- i. Display the emp name, salary, and commission for all employees who earn commissions. Sort data in descending order of salary and commissions.
- ii) Show empno and ename having deptno 20 or 30 in descending order
- iii. Write a query that displays the employee name (with the first letter uppercase and all other letters lowercase) and the length of the name for all employees whose name starts with the letters J, A, or M. Give each column an appropriate label. Sort the results descending order of the employee name.
- iv.Display the emp name and calculate the number of months between today and the date on which the employee was hired. Label the column MONTHS_WORKED. Order your results in descending order by the number of months employed.
- iii. Display the emp name and salary for all employees. Format the salary to be 15 characters long, left-padded with the "$" symbol. Label the column SALARY.
- iv. Write a query to display the current date. Label the column Date.
- v. Display the employee number, hire date, number of months employed, first Friday after hire date, and last day of the month.
- vi. Create a query that displays the employees names and commission amounts. If an employee does not earn commission, show "No Commission." Label the column COMM. (Hint: Use NVL Function)
- **Q3. Run the following queries and display their outputs:**
- i. Display ename, hiredate and print the date in DD Month YYYY Format. Label the coulmn as "DATE FORMAT". (Hint: Use To_CHAR Function)
- ii. Write a query that generates the following output:
- **(OPTIONAL QUERY)**
- 1. Display the emp name, and email address. The email address will be composed from the four first letters of emp name concatenated with the string "@gmail.com"
- (For example : LEVI MARIO levi@gmail.com)