**EXPERIMENT-3**

**AIM:**To perform inbuilt functions using DML command.

**FACILITIES REQUIRED :**

|  |  |  |
| --- | --- | --- |
| **Serial No** | **Facilities required** | **Quantity** |
| 1 | system | 1 |
| 2 | Operating system | windows |
| 3 | frontend |  |
| 4 | backend | Oracle apex |

**PROCEDURE:**

|  |  |
| --- | --- |
| **Step no** | **Details of the step** |
| 1 | Function is a group of code that accepts zero or more arguments and both return one or more results. Both are used to manipulate individual data items. Operators differ from functional in that they follow the format of function name (arg..). An argument is a user defined variables or constants. Most operators accept at most 2 arguments while the structure of functions permit to accept 3 or more arguments. Function can be classifies **into single row function and group functions** |
| 2 | **Single Row functions**  A single row function or scalar function returns only one value for every row  queries in table. Single row function can appear in a select command and can also  be included in a where clause. The single row function can be broadly classified as,  o Date Function o Numeric Function  o Character Function o Conversion Function  o Miscellaneous Function  The example that follows mostly uses the symbol table “dual”. It is a table, which is  automatically created by oracle along with the data dictionary. |
| 3 | **Date Function**  they operate on date values and produce outputs, which also belong to date data  type except for months, between, date function returns a numbe |
| 4 | **Group Functions**  A group function returns a result based on group of rows |

**SQL Commands:**

**DATE FUNCTION**

**1. Add\_month:** This function returns a date after adding a specified date with specified number of months.

Syntax: Add\_months(d,n); where d-date n-number of months

Example: Select add\_months(sysdate,2) from dual;

**2. last\_day:** It displays the last date of that month.

Syntax: last\_day (d); where d-date

Example: Select last\_day (‘1-jun-2009’) from dual;

**3. Months\_between:** It gives the difference in number of months between d1 & d2.

Syntax: month\_between (d1,d2); where d1 & d2 -dates

Example: Select month\_between (‘1-jun-2009’,’1-aug-2009’) from dual;

**4. next\_day:**It returns a day followed the specified date.

Syntax: next\_day (d,day);

Example: Select next\_day (sysdate,’wednesday’) from dual

**5. round:**This function returns the date, which is rounded to the unit specified by the format model.

Syntax : round (d,[fmt]);

where d- date, [fmt] – optional. By default date will be rounded to the nearest day

Example: Select round (to\_date(‘1-jun-2009’,’dd-mm-yy’),’year’) from dual;

Select round (‘1-jun-2009’,’year’) from dual;

**NUMERICAL FUNCTIONS**

|  |  |  |
| --- | --- | --- |
| **Command** | **Query** | **Output** |
| Abs(n)  Ceil(n)  Exp(n)  Floor(n)  Power(m,n)  Mod(m,n)  Round(m,n)  Trunc(m,n)  Sqrt(m,n) | Select abs(-15) from dual;  Select ceil(55.67) from dual;  Select exp(4) from dual;  Select floor(100.2) from dual;  Select power(4,2) from dual;  Select mod(10,3) from dual;  Select round(100.256,2) from dual;  Select trunc(100.256,2) from dual;  Select sqrt(16) from dual; | 15  56  54.59  100  16  1  100.26  100.23  4 |

**CHARACTER FUNCTIONS**

|  |  |  |
| --- | --- | --- |
| **Command** | **Query** | **Output** |
| initcap(char);  lower (char);  upper (char);  ltrim (char,[set]);  rtrim (char,[set]);  replace (char,search  string, replace string);  substr (char,m,n); | select initcap(“hello”) from dual; select lower (‘HELLO’) from dual; select upper (‘hello’) from dual; select ltrim (‘cseit’, ‘cse’) from dual; select rtrim (‘cseit’, ‘it’) from dual; select replace(‘jack and jue’,‘j’,‘bl’) from dual; select substr (‘information’, 3, 4) from dual; | Hello  hello  HELLO  it  cse  black and  blue  Form |

**CONVERSION FUNCTION**

**1. to\_char():**This function converts date to a value of varchar type in a form specified by date format.

If format is negelected then it converts date to varchar2 in the default date format.

Syntax: to\_char(d,[format]);

Example: select to\_char (sysdate, ’dd-mm-yy’) from dual;

**2. to\_date():**This function converts character to date data format specified in the form character.

Syntax: to\_date(d,[format]);

Example: select to\_date(‘aug 15 2009’,’mm-dd-yy’) from dual;

**Miscellaneous Functions**

**1. uid** – This function returns the integer value (id) corresponding to the user currently logged in.

Example: select uid from dual;

**2. user** – This function returns the logins user name.

Example: select user from dual;

**3. nvl** – The null value function is mainly used in the case where we want to consider null values as zero.

Syntax; nvl(exp1, exp2)

If exp1 is null, return exp2. If exp1 is not null, return exp1.

Example: select custid, shipdate, nvl(total,0) from order;

**4. vsize**: It returns the number of bytes in expression.

Example: select size(‘tech’) from dual;

**GROUP FUNCTIONS**

A group function returns a result based on group of rows.

**1. avg** - Example: select avg (total) from student;

**2. max** - Example: select max (percentagel) from student;

**3.min** - Example: select min (marksl) from student;

**4. sum** - Example: select sum(price) from product;

**COUNT FUNCTION**

In order to count the number of rows, count function is used.

**1. count(\*)** – It counts all, inclusive of duplicates and nulls.

Example: select count(\*) from student;

**2. count(col\_name)–** It avoids null value.

Example: select count(total) from order;

**3. count(distinct col\_name**) – It avoids the repeated and null values.

Example: select count(distinct ordid) from order;

**GROUP BY CLAUSE**

This allows us to use simultaneous column name and group functions.

Example: Select max(percentage), deptname from student group by deptname;

**HAVING CLAUSE**

This is used to specify conditions on rows retrieved by using group by clause.

Example: Select max(percentage), deptname from student group by deptname having

count(\*)>=50;

**SPECIAL OPERATORS**:

**In / not in** – used to select a equi from a specific set of values

**Any** - used to compare with a specific set of values

Between / not between – used to find between the ranges

**Like / not like** – used to do the pattern matching

**Queries**

**Q1: Display all the details of the records whose employee name starts with ‘A’.**

Ans: SQL> select \* from EMP\_DISHA\_171 where ename like 'D%';

**Q2: Display all the details of the records whose employee name does not starts with ‘A’.**

Ans: SQL> select \* from EMP\_DISHA\_171 where ename not like 'D%';

**Q3: Display the rows whose salary ranges from 12000 to 20000.**

Ans: SQL> select \* from EMP\_DISHA\_171 where sal between 12000 and 20000;

**Q4: Calculate the total and average salary amount of the emp table.**

Ans: SQL> select sum(sal),avg(sal) from EMP\_DISHA\_171;

**Q5: Count the total records in the emp table.**

Ans: SQL>select \* from EMP\_DISHA\_171;

SQL> select count(\*) from EMP\_DISHA\_171;

**Q6: Determine the max and min salary and rename the column as max\_salary and min\_salary.**

Ans: SQL> select max(sal) as max\_salary, min(sal) as min\_salary from EMP\_DISHA\_171;

**Q7: Display the month between “1-jun-10”and 1-aug-10 in full.**

Ans: SQL> select MONTHS\_BETWEEN(TO\_DATE('1-JUN-2010', 'DD-MON-YYYY'),TO\_DATE('1-AUG-2010', 'DD-MON-YYYY')) from dual;

**Q8: Display the last day of that month in “05-Oct-09”.**

Ans: SQL>select LAST\_DAY(TO\_DATE('1-JUN-2009', 'DD-MON-YYYY'))from dual;

**Q9: Find how many job titles are available in employee table.**

Ans: SQL> select count(job) from EMP\_DISHA\_171;

SQL> select count(distinct job) from EMP\_DISHA\_171;

**Q10: What is the difference between maximum and minimum salaries of employees in the organization?**

Ans: SQL> select max(sal), min(sal) from EMP\_DISHA\_171;

**Result:**

Thus the nested Queries and join Queries was performed successfully and executed.