|  |
| --- |
| **Part A** |
| **Aim:**SQL commands:   1. To perform SQL Numeric Function, String, Date and conversion functions. |
| **Prerequisite:** Oracle. |
| **Outcome:** Understanding and use of various Oracle functions. |
| Theory:  Numeric Functions  ABS: returns the absolute value of n  abs(n)  POWER: returns m raised to nth power  power(m,n)  ROUND: returns n rounded to m places to the right of the decimal point  round(n,m)  SQRT: returns square root of n  sqrt(n)  EXP: returns e raised to nth power  exp(n)  FLOOR: returns the largest integer value that is equal to or less than a number  floor(n)  CEILING: returns the smallest integer value that is equal to or greater than a number  ceiling(n)  MOD(m,n): returns the remainder of m divided by n. Returns m if n is 0.  RAND: returns a random number or a random number within a range.  Rand()  Rand()\*(upper value-lower value) e.g. rand()\*(10-1)  String Functions:  SUBSTR:  return a portion of string, beginning at character position, substring\_length characters long.   * If position is 0, then it is treated as 1. * If position is positive, then Oracle Database counts from the beginning of string to find the first character. * If position is negative, then Oracle counts backward from the end of string. * If substring\_length is omitted, then Oracle returns all characters to the end of string. If substring\_length is less than 1, then Oracle returns null.     LEN: returns the length of the specified string.  LEN( string )  CONCAT: allows you to concatenate strings together.  CONCAT( string1, string2, ... string\_n )  LOWER: converts all letters in the specified string to lowercase (same for Upper).  LOWER( string )  **Date Functions:**  **CURRENT\_TIMESTAMP:** returns the current date and time (GETDATE function can also be used).  **CURRENT\_DATE** returns the current date in the session time zone, in a value in the Gregorian calendar of datatype DATE.  **DATEADD:** returns a date after which a certain time/date interval has been added.  DATEADD( date, format, days, months, years )    **DIFFDATE:** returns the difference between two date values, based on the interval specified.  DIFFDATE ( interval, date1, date2 )  Example  SELECT DATEDIFF(day,'2016-06-05','2016-08-05') AS DiffDate  Result:  DiffDate  61  **Sysdate**:  SQL>SELECT SYSDATE FROM DUAL; 29-DEC-08  **next\_day:**  SQL>SELECT NEXT\_DAY(SYSDATE,’WED’) FROM DUAL;  05-JAN-09  **add\_months:**  SQL>SELECT ADD\_MONTHS(SYSDATE,2)FROM DUAL;  28-FEB-09  **last\_day:**  SQL>SELECT LAST\_DAY(SYSDATE)FROM DUAL;  31-DEC-08  **months\_between:**  SQL>SELECT MONTHS\_BETWEEN(SYSDATE,HIREDATE)FROM EMP;  4  **Least:**  SQL>SELECT LEAST('10-JAN-07','12-OCT-07') FROM DUAL;  10-JAN-07  **Greatest:**  SQL>SELECT GREATEST('10-JAN-07','12-OCT-07')FROM DUAL;  10-JAN-07  **to\_char:**  SQL> select to\_char(sysdate, "dd\mm\yy") from dual;  24-mar-05.  **to\_date:**  SQL> select to date (sysdate, "dd\mm\yy") from dual;  24-mar-05  **CONVERSION FUNCTIONS:**  To\_char: TO\_CHAR (number) converts n to a value of VARCHAR2 data type, using the optional number format fmt. The value n can be of type NUMBER, BINARY\_FLOAT, or BINARY\_DOUBLE.  The following are number examples for the TO\_CHAR function.  TO\_CHAR(1210.73, '9999.9')  *Result:* ' 1210.7'  TO\_CHAR(-1210.73, '9999.9')  *Result:* '-1210.7'  TO\_CHAR(1210.73, '9,999.99')  *Result:* ' 1,210.73'  TO\_CHAR(1210.73, '$9,999.00')  *Result:* ' $1,210.73'  TO\_CHAR(21, '000099')  *Result:* ' 000021'  To\_number : TO\_NUMBER converts expr to a value of NUMBER data type.  SQL>Select to\_number ('1234.64') from Dual; 1234.64  To\_date:TO\_DATE converts char of CHAR, VARCHAR2, NCHAR, or NVARCHAR2 data type to a value of DATE data type.  SQL>SELECT TO\_DATE('January 15, 1989, 11:00 A.M.')FROM DUAL; TO\_DATE --------- 15-JAN-89 |
| **Procedure:**   1. Formulate the query for given problem. 2. Write the SQL query with proper input. 3. Execute the query. |
| **Practice Exercise:**  Create a table EMPLOYEE with following schema:  (Emp\_no, E\_name, E\_address, E\_ph\_no, Dept\_no, Dept\_name,Job\_id, Designation , Salary)   1. List the E\_no, E\_name, Salary of all employees working for MANAGER. 2. Display all the details of the employee whose salary is more than the Sal of any IT PROFF.. 3. List the employees in the ascending order of Designations of those joined after 1981. 4. List the employees along with their Experience and Daily Salary. 5. List the employees who are either ‘CLERK’ or ‘ANALYST’ . 6. List the employees who joined on 1-MAY-81, 3-DEC-81, 17-DEC-81,19-JAN-80 . 7. List the employees who are working for the Deptno 10 or20. 8. List the Enames those are starting with ‘S’ . 9. Dislay the name as well as the first five characters of name(s) starting with ‘H’ 10. List all the emps except ‘PRESIDENT’ & ‘MGR” in asc order of Salaries. |
| **Instructions:**   1. Write and execute the query in Oracle SQL server. 2. Paste the snapshot of the output in input & output section. |
| **Part B** |
| **Code and Output:**  **TABLE SCHEMA** |
| **INSERTING TABLE INTO DATA** |
| **1.** |
| **2.** |
| **3.** |
| **4.** |
| **5.** |
| **6.** |
| **7.** |
| **8.** |
| **9.** |
| **10.** |
| **Observation & Learning:**  From this experiment, we observed and learned how the following **SQL** commands are used to perform **SQL Numeric Function, String, Date and conversion functions** in the **ORACLE DATABASE.** |
| **Conclusion:**  In this experiment, the following **SQL commands** are executed to perform **SQL Numeric Function, String, Date and conversion functions in the queries** in the **ORACLE DATABASE** and the Outputs are obtained as per queries. |
| **Questions:**   1. **What is the use of aggregate function?** 2. An **aggregate function** performs a calculation one or more values and returns a single value. The aggregate function is often used with the **GROUP BY** clause and **HAVING** clause of the **SELECT** statement.   **a)** The **AVG** aggregate function calculates the average of **non-NULL** values in a set  **b)** The **COUNT** function returns the total number of values in the specified field. It works on both numeric and non-numeric data types.  **c)COUNT (\*)** is a special implementation of the **COUNT** function that returns the count of all the rows in a specified table. **COUNT (\*)** also considers Nulls and duplicates.  **d)**The **MIN** function returns the smallest value in the specified table field.  **e)** The **MAX** function returns the largest value from the specified table field.  **f)** The **SUM** function which returns the sum of all the values in the specified column. **SUM** works on numeric fields only. **Null** values are excluded from the result returned.   1. **How different number of rows can be counted?** 2. The **SQL COUNT ()** function returns the number of rows in a table satisfying the criteria specified in the WHERE clause. It sets the number of rows or **non -NULL** column values. **COUNT ()** returns **0** if there were no matching rows. To calculate the different number of rows we use the **DISTINCT** keyword in **COUNT ()** function which returns the **distinct or different** rows. 3. **What is the difference between having and where clause?** 4. **1)**The **WHERE** clause is used in the selection of rows according to given conditions whereas the **HAVING** clause is used in column operations and is applied to aggregated rows or groups. 5. If **GROUP BY** is used then it is executed after the **WHERE** clause is executed in the query. It means it selects the rows before grouping is done or aggregate calculations are performed. That’s why, **WHERE** clause is also called Pre-filter. 6. But, **GROUP BY** is executed before the execution of the **HAVING** clause. It means it selects the rows after aggregate calculations are performed. That’s why, **HAVING** clause is also called as Post-filter. 7. We cannot use the **HAVING** clause without **SELECT** statement whereas the **WHERE** clause can be used with **SELECT, UPDATE, DELETE** etc. 8. We can use aggregate functions like **sum, min, max, avg**, etc with the **HAVING** clause but they can never be used with **WHERE** clause. 9. **Does *WHERE* clause work with aggregate functions?** 10. **WHERE** clause does not work with aggregate functions. An **aggregate function** may not appear in the **WHERE** clause unless it is in a subquery contained in a **HAVING** clause or a select list, and the column being aggregated is an outer reference. |