**19-Oct-2021**

Oracle: RDBMS Software PRODUCT, where used to store the data and perform operations on the DATA.

Oracle :

1. SQL
2. PLSQL

What is DATA?

It is a collection of Raw facts.

**Example**: 101 dinesh 2000

102 mahesh 3000

In above example, there no meaningful data such data is known as Raw facts.

What is INFORMATION?

**INFORMATION**: It is a collection of meaningful data or processed data.

Example: EmpID Ename Salary

101 dinesh 2000

102 mahesh 3000

What are different ways to store the data?

**DATA STORE:** It is a place where we can store data or information.

1) Books & Papers

2) Flat files

3) Database

**FLAT FILES**: This is a traditional mechanism which is used to store data or information in

individual unrelated files. These files are also called as Flat Files.

What all challenges if you store the data in flat files?

**Drawbacks of Flat files:**

1) Data Retrieval

2) Data Redundancy

3) Data Integrity

4) Data Security

5) Data Indexing

**20-Oct-2021**

1**) Data Retrieval:** If we want to retrieve data from flat files then we must develop

application program in high level languages, where as if we want to retrieve data from

databases then we are using Sequel Language.

SEQUEL (Structured English Query Language)

2) **Data Redundancy:** Sometimes we are maintaining multiple copies of the same data in

different locations this data is also called as Duplicate data or Redundant data. In Flat files

mechanism when we are modifying data in one location it is not effected in another

location. This is called INCONSISTENCY.

In databases, every transaction internally having 4 properties. These properties are known as

ACID properties.

**ACID Properties:**

A mean Atomicity (ROLLBACK)

C mean Consistency

I mean Isolation

D mean Durability (COMMIT)

These properties only automatically maintains consistent data in databases.

**3) Data Integrity:** Integrity means to maintain proper data. If we want to maintain proper

data then we are defining set of rules, these rules are also called as “ Business rules”. In

databases, we are maintaining proper data using „constraints‟, „triggers‟. If we want to

maintain proper data in flat files we must develop application programs in high level

languages like COBOL, JAVA, ETC…..

**4) Data Security:** Data stored in flat files cannot be secured because flat files doesn‟t

provides security mechanism. Whereas databases provides “ROLE based security”.

**5) Data Indexing:** If we want to retrieve data very quickly from database then we are using

indexing mechanism. Whereas flat files doesn‟t provide indexing mechanism.

To overcome all the above problems, a new software used by all organization to store data or

information in secondary storage devices. This is called DBMS software.

What all the advantages if you store the data in Oracle DB?

By using RDBMS we can achieve the above drawbacks.

**Flat File Scenario, How the data looks like**

|  |  |  |
| --- | --- | --- |
| Invoice info | Sales Info | Customer Info |
| Cust No  Customer Name  Invoice Details | Cust No  Customer Name  Sale Info | Cust No  Customer Name  Customer Address |

**Oracle DBMS scenario:**

|  |  |
| --- | --- |
| **Customer Info**  Cust No (Primary Key)  Customer Name  Customer Address  Customer Country  Customer City | **Invoice Info**  Cust No (Foreign Key in My Invocie)  Invoice Details |
| **Sale Info**  Cust No(Foreign Key in My Sale Info)  Sales Info |

Select

custno,

Customer name,

Invoice\_details,sales info

from

Customer\_info CI,

INVOICE\_info II,

SALE\_INfO SI

WHERE

CI.CUST NO=II.CUST NO

AND CI.CUSTNO=SI.CUSTNO;

SELECT CUSTNO AS “Customer No” FROM Customer\_info CI WHERE CI.CUSTNO=’001’ ;

SELECT \* FROM INVOICE\_info;

SELECT \* FROM SALE\_INFO;

ORACLE RDBMS : ORACLE is a Relational Database product which is used to store data permanently in secondary storage devices.

**21-Oct-2021**

**DATABASE:** It is an organized collection of interrelated data used by application program in

an organization. Once data stored in database it can be shared by number of users

simultaneously and also this data can be integrated.

**DBMS Architecture:** American National Standard Institute(ANSI) has established three level

architecture for database. This architecture is also called as “ansi/sparc” (Standard Planning

And Requirements Committee) architecture.

Main objective of DBMS architecture is to separate users view of the database from the where

physically it is stored.

This architecture mainly consists of three levels. They are:

1) External level

2) Conceptual level

3) Internal level



**DBMS Architecture or Three Level Architecture**

DBMS architecture provides “DATA INDEPENDENCE”.

**Data Independence:** Upper levels are unaffected by changes in the lower levels is called as

“Data Independence”. DBMS architecture have two types of Data Independences:

1) Logical Data Independence

2) Physical Data Independence

**1) Logical Data Independence:** Changes to the conceptual level do not required to change

to the external level this is called Logical Data Independence.

**Example**: Adding a new entity in conceptual level does not effect in external level.

**2) Physical Data Independence:** Changes to the internal level do not required to changes

in conceptual level. This is called Physical Data Independence.

**Example**: Adding an index to the internal level it is not affected in conceptual level.

**CONCEPTUAL LEVEL:** It describes logical representation of the database. Conceptual level

defines type of data storing in database and also defines what type of data does not store in

database using constraints and also specifies the relationship between data items.

NOTE: This level does not define how data is stored in database. In relational databases we

are defining conceptual levels through tables.

**EXTERNAL LEVEL:** This level describes end user view of the database. i.e., in this level

some group of users access only part of the database. In this level only we are defining the

view and those views given to the number of users.

**INTERNAL LEVEL:** Internal level describes how physically data is stored in database. This

level is handled by database administrator only. In relational databases indexes, cluster are

available in internal level.



**Data Models:**

1. **Hierarchical Data Model**

In this data model organizes data in tree like structure, we are representing data in parent child hierarchy. In this data model also data is represented in the format of records and also record type is also same as table in relational data model.

This Data model having more duplicate records because this data model is implemented

based on one- to – many relationships. That is why in this data model always child segments

are repeated. In this data model products, we are retrieved data very slowly because in this data

model products data base servers searching data based on root node onwards.

Root

|  |  |
| --- | --- |
| Node 1 | Node 2 |



1. **Network Model:**

In this data model is implemented based on many – to – many relationships. In this data model also data is stored in format of records. And also records type is also same as table in Relational Data model.



1. **Relational Model**

In this data model we are storing data in 2- dimensional tables. Relational data model mainly consist of 3 components.

1) Collection of Objects. (Tables)

2) Set of Operators. (<,>,=,<>)

3) Set of Integrity rules. (Constraints)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Dept No | Dept Name |  | Emp | DeptNo | Joining Date |
| 10 | IT | 1001 | 10 | 01-Sep-2021 |
| 20 | HR | 1002 | 10 | 01-Sep-2021 |
| 30 | SE | 1003 | 20 | 01-Oct-2021 |
| Primary Key | | 1004 | 30 | 15-Oct-2021 |
| Index will be created automatically. | |  | Primary Key Foreign Key references to Dept (Dept NO) | | |

Select

EmpId, Deptno, Dept\_name, Joining\_date

From Employee E, Dept D

Where e.dept\_no=d.dept\_no and emp\_no=1001;

**ORACLE VERSIONS**

1)**Oracle 2.0** -> 1979

-> First public release

-> In this 2.0 only basic SQL functionality is there “joins”.

2) **Oracle 3.0** -> 1983

-> Rewritten in C language.

-> Commit, Roll back.

3) **Oracle 4.0** -> 1984

-> Read Consistancy

-> exp/imp utility programs [export/import].

4) **Oracle 5.0** -> 1985

-> Client server architecture.

5) **Oracle 6.0** -> 1988

-> Introduced PL/SQL

-> Row level locks.

6) **Oracle 7.0** -> 1992

-> Roles

-> Integrity constraints.

-> Stored Procedures

-> Stored Functions

-> Packages

-> Triggers

-> Data type “varchar” changed into “varchar2”

-> Truncate table.

7) **Oracle 7.1** -> 1994

-> Introduced dynamic SQL

-> ANSI/ISO SQL-92.

8) **Oracle 7.2** -> 1995

-> Inline views or sub queries used in from clause.

-> ref cursor (cursor variable)

9) **Oracle 7.3** -> 1996

-> Bit map indexes.

-> utl\_file package.

10) **Oracle 8.0** -> 1997

-> Object technology

-> Columns increased per a table upto “1000”.

-> nested table, varray.

-> Instead of triggers.

11) **Oracle 8i** (i- internet) -> 1999

-> Materialized views.

-> Function based indexes

-> Case conditional statements.

-> Analytical functions.

-> Autonomous trasactions

-> rollup, cube.

-> BULK BIND

12) **Oracle 9i** -> 2001

-> 9i joins or ansi joins

-> merge statements.

-> multi table insert

-> flash back queries.

-> Renaming a column

13) **Oracle 10g (g- grid)** -> 2003

-> recycle bin

-> flash back table

-> indices of clause

-> regular expressions

-> wm\_concat().

14) **Oracle 11g** -> 2007

-> Introduced continue statement in PL/SQL loops

-> Read only tables

-> Virtual Columns

-> Pivot() fuction.

-> Compound trigger

-> enable, disable clauses used in trigger specification

-> follow clause

-> Sequences used in PL/SQL without using dual table.

-> Named, mixed notations are used in a subprogram executed used select statement

**SQL (STRUCUTRED QUERY LANGUAGE):**

It’s a non-procedural language which is used to operate all relational database products.

**22-Oct-21**

**SQL Sub Languages in every database:**

**Data Definition Language (DDL)**

CREATE, ALTER, TRUNCATE, DROP (PURGE to remove permanently from BIN), RENAME

**Ex** : create table test(sno number(7));

**Data Manipulation Language(DML)**

INSERT, UPDATE, DELETE (IN (‘1008’,’1009’), = ‘1008’) rownum<11, MERGE

Ex : insert into test1 values(99.9);

Commit;

**Data Retrieval Language(DRL) or Data Query Language(DQL)**

SELETE

**Ex :** select \* from test1;

**Transactional Control Language (TCL)**

ROLLBACK, COMMIT and SAVEPOINT

**Ex :** insert into test1 values(101.5);

Rollback;

**Data Control Language(DCL)**

GRANT, REVOKE (Type of Grants : SELECT, ALL, INSERT, UPDATE, DELETE)

**Ex :** Grant select on test1 to user(username);

**Data Types:**

Data types specifies type of data within a table column Oracle have following data types.

1. **Number : Number values into the table (38)**
2. **Char : It allows text into table column (2000)**
3. **Date : it allow date value into the table column**
4. **Varchar2 : It allows text into table column (4000)**
5. **Long : It allow more than 4000 Chars (2GB)**
6. **LOBs : Large no.of Data, Images**

**23-Oct-21**

**Questions:**

1. **What is Table and what are rules applicable to create a table?**
2. **What is DDL and List out the DDL commands?**
3. **What is DML and List out the DML commands?**
4. **How to query or retrieve the data from Database?**
5. **What is the difference between Delete and Truncate?**
6. **What is DROP command and Usage of DROP command?**
7. **How to remove a table from Database permanently?**
8. **What is Merge statement and usage of Merge statement.?**
9. **What is the difference between DDL and DML commands?**

**Practice**

1. Create a Employee table with below columns.

|  |  |
| --- | --- |
| Column Name | Datatype |
| Empno | number |
| Ename | Varchar2 |
| Deptno | number |
| Job | Varchar2 |
| Sal | Number |
| Comm | Number |
| Manager | Number |
| Hiredate | Date |
| Inserted Date | Date (Assign Default value as sysdate) |
| Inserted By | Varchar2 (Assign Default value as current system user) |

Insert below data into Employee table.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **EMPNO** | **ENAME** | **JOB** | **MGR** | **HIREDATE** | **SAL** | **COMM** | **DEPTNO** | **Inserted Date** | **Inserted By** |
| 7839 | KING | PRESIDENT |  | 17-NOV-81 12.00.00 AM | 5000 |  | 10 | sysdate | system user |
| 7698 | BLAKE | MANAGER | 7839 | 01-MAY-81 12.00.00 AM | 2850 |  | 30 | sysdate | system user |
| 7782 | CLARK | MANAGER | 7839 | 09-JUN-81 12.00.00 AM | 2450 |  | 10 | sysdate | system user |
| 7566 | JONES | MANAGER | 7839 | 02-APR-81 12.00.00 AM | 2975 |  | 20 | sysdate | system user |
| 7788 | SCOTT | ANALYST | 7566 | 19-APR-87 12.00.00 AM | 3000 |  | 20 | sysdate | system user |
| 7902 | FORD | ANALYST | 7566 | 03-DEC-81 12.00.00 AM | 3000 |  | 20 | sysdate | system user |
| 7369 | SMITH | CLERK | 7902 | 17-DEC-80 12.00.00 AM | 800 |  | 20 | sysdate | system user |
| 7499 | ALLEN | SALESMAN | 7698 | 20-FEB-81 12.00.00 AM | 1600 | 300 | 30 | sysdate | system user |
| 7521 | WARD | SALESMAN | 7698 | 22-FEB-81 12.00.00 AM | 1250 | 500 | 30 | sysdate | system user |
| 7654 | MARTIN | SALESMAN | 7698 | 28-SEP-81 12.00.00 AM | 1250 | 1400 | 30 | sysdate | system user |
| 7844 | TURNER | SALESMAN | 7698 | 08-SEP-81 12.00.00 AM | 1500 | 0 | 30 | sysdate | system user |
| 7876 | ADAMS | CLERK | 7788 | 23-MAY-87 12.00.00 AM | 1100 |  | 20 | sysdate | system user |
| 7900 | JAMES | CLERK | 7698 | 03-DEC-81 12.00.00 AM | 950 |  | 30 | sysdate | system user |
| 7934 | MILLER | CLERK | 7782 | 23-JAN-82 12.00.00 AM | 1300 |  | 10 | sysdate | system user |

1. Create Dept table with below columns and values.

|  |  |  |
| --- | --- | --- |
| **DEPTNO** | **DNAME** | **LOC** |
| 10 | ACCOUNTING | NEW YORK |
| 20 | RESEARCH | DALLAS |
| 30 | SALES | CHICAGO |
| 40 | OPERATIONS | BOSTON |

1. Create a Salgrade table with Below columns and values.

|  |  |  |
| --- | --- | --- |
| **GRADE** | **LOSAL** | **HISAL** |
| 1 | 700 | 1200 |
| 2 | 1201 | 1400 |
| 3 | 1401 | 2000 |
| 4 | 2001 | 3000 |
| 5 | 3001 | 9999 |

**Queries to write**

1. Display the details of all employees.
2. Display the department information from DEPT table.
3. Display the name and job for all the employees
4. Display the name and salary for the all the employees
5. Display the name of all the employees who are working under dept no 10.
6. Display the names of all the employees who are working as clerks, salesman or analyst.
7. Display the names of the employees who are earning comm.
8. Display the names of the employees who do not earn comm.
9. Display sysdate or current date.
10. Display current user.

**26-Oct-21**

**Creating a new table from another table:**

**Syntax:** create table newtablename as select \* from existingtablename;

**Example:** SQL> create table test as select \* from emp;

SQL> select \* from test;

**Note:** In all database systems whenever we are copying a table from another table

constraints are never copied. (Primary table, Foreign key,…..);

Creating a new table from existing table without copying data:

**Syntax:** create table newtablename as select \* from existingtablename where

falsecondition;

**Example:** SQL> create table test1 as select \* from emp where 1=2;

**Testing:** SQL> select \* from test1;

No rows selected.

SQL> desc test1;

**Operators Used in “Select” statement:**

1. Arthematic operator(+,-,\*,/)
2. Relational operator(=,<,<=,>,>=,[<> or!=]not equal).
3. 3. Logical operator (AND,OR,NOT)

**Syntax:**

select col1,col2,…. From tablename where condition

group by columnname

having condition

order by columnname[asc/desc];

1. Special operator

Arithmetic operator is used for “select” (eg: select column1, column2,…)

Relational and Logical and Special operators are used for “where” (eg: where condition)

We can also use arithmetic operators in “where” conditions.

 Arithmetic operators are used in number, data datatype column.

Q) write a query to display ename, sal, annsal from emp table;

Ans: SQL> select ename, sal, sal\*12 annsal from emp;

ENAME SAL ANNSAL

----------- ----------- --------------------

SMITH 700 8400

Q) write a query to display the employees except job as clerk from emp table.

Ans: select \* from emp where job<>‟CLERK‟; {or job != „CLERK‟}

Q) Write a query to display the employees who are getting more than 2000 salary from emptable?

Ans: SQL> select \* from emp where sal>2000;

**Example:** SQL> select \* from emp where job=‟CLERK‟ and sal>2000;

**Note:** In databases if we want to retrieve multiple values within a single column then we

must use “OR” operator.

**Example:** SQL> select \* from emp where job=‟CLERK‟ or job=‟SALESMAN‟;

Q) write a query to display the employees who are belongs to the department numbers

20,50,70,90?

Ans: SQL> select \* from emp where deptno=20 or deptno=50 or deptno=70 or deptno=90;

 SQL> select \* from emp where sal>2000 and sal<5000;

**\*\* Special operators:**

1) „In‟ opposite „not in‟

2) „between‟ opposite „not between‟

3) „is null‟ opposite „is not null‟

4) „like‟ opposite „not like‟

1. **In:** It is used to pick the values one by one from list of values. „In‟ operator performance

is very high compared to „OR‟ operator. If we want to retrieve multiple values in a single

column then we are using „IN‟ operators in place of „OR‟ operator because „IN‟ operator

performance is high.

**Syntax:** select \* from tablename where columnname in(list of values);

Columnname belongs to any data type is possible.

**Example:** SQL> select \* from emp where deptno in(20,30,50,70,90);

Eg: SQL> select \* from emp where ename in(„SMITH‟,‟FORD);

Note: In all datebase systems whenever we are using multiple row sub queries then we

must use “in” operator.

Eg: select \* from emp where deptno not in(10,20,null);

**Note:** In all relational databases “not in” operator doesn‟t work with “null” values.

**27-Oct-21**

**Null:** Null is an undefined, unavailable, unknown value. It is not same as “zero”. Any

arithmetic operations performed on null values again it becomes “null”.

Eg: null+50=> null.

Q) Write a query to display ename, sal, comm, sal+comm of the employee SMITH from emp

table.

Ans: SQL> select ename, sal, comm, sal+comm from emp where ename=”SMITH”;

o/p: ename sal comm. Sal+comm

---------- ------- ----------- ----------------

SMITH 1100 null null

To overcome this problem oracle provided “NVL()” function.

**NVL():** NVL() is a predefined function which is used to replace or substitute user defined

value in place of “null”.

**Syntax:** NVL(exp1,exp2);

Here exp1,exp2 must belongs to same datatype. If exp1 is null then it returns exp2.

Otherwise it returns exp1.

Eg: 1) NVL(null,30) -> 30.

2) NVL(10,20) -> 10.

Solution: select ename, sal, comm, sal+NVL(comm, 0) from emp where ename= „SMITH”;

ENAME SAL COMM SAL+COMM

--------- -------- ----------- ------------------

SMITH 1100 Null 1100

 Sal+ nvl(comm.,0)

 1100 + nvl(1100,0)

 1100 + 0

 1100

**NVL2():** Oracle 9i introduced NVL2() function. This function accepts three parameters.

**Syntax:** NVL2(exp1,exp2,exp3).

Here if exp1 is null, then it returns exp3. Otherwise it returns exp2.

Eg: SQL> select nvl2(null,10,20) from dual;

20

SQL> select nvl2(30,40,50) from dual;

40

Q) Update employee commission as follows using nvl2() from emp table;

1) If comm is null then update comm.->500

2) If comm is not null then update comm-> 500

Ans: SQL> update emp set comm=nvl2(comm, comm+500,500);

SQL> update emp set comm=nvl2(comm,comm+500,500);

SQL> select \* from emp;

**2) Between:** This operator is used to retrieve range of values. This operator is also call as

Betweem….. And operator.

Syntax: Select \* from tablename where columnname between lowvalue and highvalue.

Eg: select \* from emp where sal between 2000 and 5000;

While using “DATE”:

SQL> select \* from emp where hiredate between ‟01-jan-1981‟ and ‟01-jan-1982‟;

**3) Is null, is not null:** These two operators used in „where” condition only. These two

operators are used to test weather a column having null values or not.

Syntax: select \* from tablename where columnname is null;

Syntax: select \* from tablename where columnname is not null;

Q) write a query to display the employees who are not getting commission from emp table.

Ans: SQL> select \* from emp where comm is null;

Q) write a query to display the employees who are getting commission from emp table?

Ans: SQL> select \* from emp where comm is not null;

**4) Like:** It is used to search strings based on character pattern. “Like” operator

performance is very high compare to searching functions. In all databases along with like

operator we are using 2 wild card charcters.

% -> group of characters if want to match or string

\_ -> single character to match.

**Syntax:** select \* from tablename where columnname like „characterpattern‟;

Q) write a query to display the employees whose ename start with “M” from emp table using

like operator.

Ans: SQL> select \* from emp where ename like „M‟;

MARTIN

MILLER

Q) write a query to display the employees where ename second letter will be „L‟ from emp table using „like‟ operator?

Ans: SQL> select \* from emp where ename like „\_ \_L%‟;

ALLEN

CLARK

BLAKE

Q) write a query to display the employees who are joining in the month December from the emp table using like operator.

Ans: SQL> select \* from emp where hiredate like „\_ \_ \_D%‟; (or) „%DEC%”;

Q) write a query to display the employees who are joining in the year “81” from emp table using like operator?

Ans: SQL> select \* from emp where emp table like „%81‟;

**“Escape” function used in like operator:**

Whenever wild card characters available in character data then we are trying to retrieve the

data using like operator then database servers treated these wild card characters as special

characters to overcome this problem ansi/iso SQL provided a special function “Escape” along

with „like‟ operator. Which is used to escape special characters and also this function treated

as \_,% as same meaning as \_,%.

Syntax: select \* from tablename where columnname like „character pattern‟ escape „escape

character‟;

**Note:** Escape character must be an single character having length „1‟ within character

pattern we must use escape character before wild card character.

Eg: SQL> insert into emp(empno, ename) values(1,‟S\_ MITH‟);

SQL> select \* from emp;

Q) write a query to display the employees whose ename start with „S\_‟ from emp table using

„like‟ operator?

Ans: SQL> select \* from emp where ename like „S\_%‟;

SMITH

SCOTT

S\_MITH

Solution: SQL> select \* from emp where ename like „S?\_%‟ escape‟?‟;

S\_MITH

Eg: insert into emp(empno, ename) values(2,‟S\_\_MITH‟);

SQL> select \* from emp;

Solution: SQL> select \* from emp where ename like „S?\_?\_%‟ escape‟?‟;

S\_\_MITH

 **Concatenation Operator(|| double pipe):** It is not a special operator rarely used in

“SQL” and regularly used in “PL/SQL”. If we want to display column data along with literal

strings then we must use concatenation operator.

Eg: select „my employee names are‟|| ename from emp;

If we want to display our own space in between the columns then also we can use

concatenation operator.

Eg: select ename||‟ „||sal from emp;

**28-Oct-21**

**Functions:** Functions are used to solve particular task and also functions must return a

value. Oracle have two types of functions.

1. Predefined functions

2. User defined functions

1) **Predefined functions:** there are 4 types.

 Number function

 Character function

 Date function

 Group function(or) Aggregate function

1) **Number function:** These functions operate over “number” data.

 **Abs() :** It is used to convert negative values into positive values.

Eg: SQL> select abs(-50) from dual;

50

 **Dual():** Dual is a predefined virtual table which contains only one row and one column

Dual table is used to test predefined, user defined functions functionality.

Example for testing predefined functions:

SQL> select nvl(null,30) from dual;

30

SQL> select nvl(20,30) from dual;

20

SQL> select \* from dual;

**Note:** In oracle by default dual table column datatype is varchar2().

Eg: select \* from dual;

Generally, this table is also used to perform mathematical operations.

Eg: SQL> select 10+50 from dual;

60

Eg: SQL> select ename, comm, sal, abs(comm-sal) from emp where comm is not null;

 **Mod(m,n):** It will gives remainder after „m‟ divided by „n‟.

Eg: SQL> select mod(10,5) from dual;

0

 **Round(m,n):** It rounds given floated valued number „m‟ based on „n‟.

Eg: SQL> select round(1.8) from dual;

2

SQL> select round(1.23456,3) from dual;

1.235

**Note:** Round always checks remaining number if remaining number is above 50% then

one added to the rounded number.

Eg: SQL> select round(1285.456,-1) from dual;

1290

**Execution:**

Step1: 1280->5 is replace with 0. Out of 10, 5 is above 50%(>=)

Step2: 1280

+1

---------------

1290

Eg: select ename, sal, round(sal,-1) from emp;

Eg: select ename, sal, round(sal/22), round(sal/22,1), round(sal/22,2) from emp;

 **Trunc(m,n):** It truncates given floated values number „m‟ based on „n‟. This function

doesn‟t check remaining number is above 50% or below 50%.

Eg: SQL>select trunk(1.8) from dual;

Eg: SQL> select trunk(1.23456,3) from dual;

1.234

 **Greatest(exp1,exp2,…. expn), Least(exp1,exp2,…. expn):**

Greatest returns maximum value among given expressions. Where as Least returns

minimum value among given expressions.

Eg: SQL> select greatest(3,5,8,9) from dual ;

9

SQL> select ename, sal, comm, greatest(sal,comm)from emp where comm is not null;

SQL> select max(sal) from emp;

6600

 **Ceil() and Floor():** These two functions always return integer. Ceil() returns nearest

greatest integer where as floor() returns nearest lowest integer.

Eg: select ceil(1.3) from dual;

2

Eg: select floor(1.9) from dual;

1

**2) CHARCTER FUNCTIONS:**

 **Upper():** It is used to convert a string or column values into “ upper case “.

Eg: select upper(„abc‟) from dual;

ABC

For column: SQL> select upper(ename) from dual;

 **Lower():**

Eg: select lower(ename) from emp;

Updating or modifying data within table:

SQL> update emp set ename=lower(ename);

 **Initcap():** It returns first letter is capital and all remaining letters are small.

Eg: SQL> select initcap(ename) from emp;

Eg: SQL> select initcap(„ab cd ef‟) from emp;

Ab Cd Ef

 **Length():** It returns number datatype that is ., it returns total length of the string

including spaces.

Eg: select length(„AB\_CD‟) from dual;

5

**29-Oct-21**

 **Substr():** It will extract portion of the string within given string based on last two

parameters.

Eg: select substr(„ABCDEF‟,2,3) from dual;

BCD

SQL> select substr(„ABCDEF‟,-2,3) from dual;

EF

SQL> select substr(„ABCDEF‟,-4) from dual;

CDEF

**Syntax:** substr(columnname (or) „string name‟, starting position, no. of characters position);

Starting position-> can be +ve or –ve

Starting position and no. of characters position-> must be numbers

Q) write a query to display the employees whose ename second letter would be “LA” from

emp table using substring function?

And: SQL> select \* from emp where substr(ename,2,2)=‟LA‟;

BLAKE

CLARK

**Note:** In all database systems we are not allowed to use group functions in where clause but

we are allowed to use number functions, character functions, date functions in where clause.

Eg: select \* from emp where sal=max(sal);

Error: group function is not allowed here.

Q) write a query to display the employee whose employees length is 5 from emp table.

Ans: SQL> select \* from emp where length(ename)=5;

 **Instr():** instr() always returns number datatype that is it returns position of the

delimiter, position of the character, position of the string within given string .

Eg: SQL> select instr(„ABC\*D‟,‟\*‟) from dual;

**Date: 19/3/15**

Eg: SQL> select instr(„ABCDEFGHCDKMNHCDJL‟,‟CD‟,-5,2) from dual;

3

SQL> select instr(„ABCDEFGHCDKMNHCDJL‟,‟CD‟,-4,2) from dual;

9

**Syntax:** instr(columnname(or)‟string name‟, „str‟, searching position, No. of occurrences from

searching position);

Searching position -> +ve or –ve

Searching and No. of occurrences -> must be numbers.

Always in string returns position based on last two parameters but oracle server counts no. of

characters left side first position onwards.

**Lpad():** It will fills remaining spaces with specified character on the left side of the given

string. Here always second parameter returns total length of the string. And also third

parameter is an optional.Syntax: Lpad(columnname (or) „string name‟, total length, „filled

character‟);

Total length -> number

Eg: SQL> select lpad(„ABCD‟,10,‟#‟) from dual;

######ABCD

SQL> select lpad(„ABC‟,5) from dual;

(space)(space)ABC

SQL> select lpad(„ABC‟,2,‟\*‟) from dual;

AB

**Rpad():**

Eg: SQL> select rpad(„ABCD‟,10,‟#‟)from dual;

ABCD######

SQL> select rpad(ENAME,50,‟-„)||sal from emp;

**Ltrim:** It removies specified character on the left side of the given string.

Syntax: Ltrim(columnname(or)‟string name‟, {set of characters});

Eg: SQL> select ltrim(„SSMISSTHSS‟,‟S‟)from dual;

MISSTHSS

SQL> select job,ltrim(job,‟CSM‟) from emp;

JOB LTRIM(JOB)

----------------- -------------------------

CLERK LERK

SALESMAN ALESMAN

MANAGER ANAGER

**Rtrim:**

Eg: SQL> select rtrim(„SSMISSTHSS‟,‟S‟) from dual;

SSMISSTH

**Trim():** Oracle 8i introduced trim function it is used to remove left and right side specified

characters and also it is used to remove leading and trailing spaces.

**Syntax:** trim(„character‟ from „string name‟);

Eg: SQL> select trim(„S‟ from „SSMISSTHSS‟)from dual;

MISSTH

This function also behaves like a ltrim, rtrim based on leading and trailing claused.

Eg: select trim(leading „s‟ from „SSMISSTHSS‟) from dual;

SSMISSTH

Eg: select length(trim(„ welcome „))from dual;

7

**Translate() and Replace():**

Translate() is used to replaces character by character where as replace() is used to replaces

character by string (or) string by string.

Eg: select translate(„india‟,‟in‟,‟xy‟),replace(„india‟,‟in‟,‟xy‟) from dual;

Translate Replace

Xydxa xydia

**Syntax:** translate(„str1‟,‟str2‟,‟str3‟….);

Eg: select translate(„ABCDEF‟,‟FEDCBA‟,123456);

654321

Eg: select replace(„ABC‟,‟ „,‟india‟)from dual;

AindiaBindiaC

Eg: select job,replace(job, „SALESMAN‟,‟MARKETING‟) from emp;

Note: In replace function if we are not specifying third parameter specified character

permanently removed from the string.

Eg: select replace(„SSMISSTHSS‟,‟S‟) from dual;

MITH

If you want to count number of times particular character occurs within a given string then

also we are using replace function along with length function.

Q) Write a query to count number of times that particular „I‟ occurred within given string

„india‟ using replace function?

Ans: SQL> select length(„india‟)-length(replace(„india‟,‟I‟)) from dual;

**2**

**Concat():** It is used to concatenate given two strings.

Eg: select concat(„wel‟,‟come‟)from dual;

Welcome