Explain DDL, DML, DCL and DQL. OR Describe component of SQL.

DDL (Data Definition Language)

- It is a set of SQL commands used to create, modify and delete database objects such as tables, views, indices, etc.
- It is normally used by DBA and database designers.
- It provides commands like:
  - ✓ CREATE: to create objects in a database.
  - ✓ ALTER: to alter the schema, or logical structure, of the database.
  - ✓ DROP: to delete objects from the database.
  - ✓ TRUNCATE: to remove all records from the table.

DML (Data manipulation Language)

- It is a set of SQL commands used to insert, modify and delete data in a database.
- It is normally used by general users who are accessing database via pre-developed applications.
- It provides commands like:
  - ✓ INSERT: to insert data into a table.
  - ✓ UPDATE: to modify existing data in a table.
  - ✓ DELETE: to delete records from a table.
  - ✓ LOCK: to lock tables to provide concurrency control among multiple users.

DQL (Data Query Language)

- It is a component of SQL that allows data retrieval from the database.
- It provides command like SELECT. This command is a heart of SQL, and allows data retrieval in different ways.

DCL (Data Control Language)

- It is set of SQL commands used to control access to data and database. Occasionally DCL commands are grouped with DML commands.
- It provides commands like:
  - ✓ COMMIT: to save work permanently.
  - ✓ ROLLBACK: to undo work and restore database to previous state.
  - ✓ SAVEPOINT: to identify a point in a transaction to which work can be undone.
  - ✓ GRANT: to give access privileges to users on the database.
  - ✓ REVOKE: to withdraw access privileges given to users on the database.

Various Syntax in SQL

SQL SELECT Statement

```
SELECT column1, column2....columnN
FROM   table_name;
```

```
Ex. SELECT * FROM EMPLYEE;    (TO SELECT ALL ATTRIBUTES)
    SELECT EMP_NO,EMP_NAME FROM EMPLOYEE;  (TO SELECT SELECTED ATTRIBUTES)
```

SQL DISTINCT Clause

```
SELECT DISTINCT column1, column2....columnN
FROM   table_name;
```

SQL WHERE Clause

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  CONDITION;
```

```
EX.  SELECT EMP_NO,EMP_NAME FROM EMPLOYEE WHRE EMP_NO=101;
```

SQL AND/OR Clause

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  CONDITION-1 {AND|OR} CONDITION-2;
```

```
EX.  SELECT EMP_NO,EMP_NAME FROM EMPLOYEE WHRE EMP_NO=101 AND EMP_SAL > 20000 ;
```

SQL IN Clause

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  column_name IN (val-1, val-2,...val-N);
```

```
EX.  SELECT EMP_NO,EMP_NAME FROM EMPLOYEE WHRE DEPT_NO IN (10,20);
```

SQL BETWEEN Clause

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  column_name BETWEEN val-1 AND val-2;
```

```
EX.  SELECT EMP_NO,EMP_NAME FROM EMPLOYEE WHRE DEPT_NO BETWEEN 10 AND 20;
```

SQL LIKE Clause

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  column_name LIKE { PATTERN };
```

```
EX.  SELECT EMP_NO,EMP_NAME FROM EMPLOYEE WHRE EMP_NAME LIKE 'a_%';
```

SQL ORDER BY Clause

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  CONDITION
ORDER BY column_name {ASC|DESC};
```

EX. SELECT EMP_NO,EMP_NAME FROM EMPLOYEE WHRE EMP_NAME LIKE 'a_%' ORDER BY DEPT_NO DESC;

SQL GROUP BY Clause

```
SELECT SUM(column_name)
FROM   table_name
WHERE  CONDITION
GROUP BY column_name;
```

EX. SELECT MAX(EMP_SAL) FROM EMPLOYEE GROUP BY DEPT_NO;

SQL COUNT Clause

```
SELECT COUNT(column_name)
FROM   table_name
WHERE  CONDITION;
```

EX. SELECT COUNT(EMP_NAME) FROM EMPLOYEE WHERE EMP_SAL > 50000;

SQL HAVING Clause

```
SELECT SUM(column_name)
FROM   table_name
WHERE  CONDITION
GROUP BY column_name
HAVING (arithematic function condition);
```

EX. SELECT SUM(EMP_SAL) FROM EMPLOYEE GROUP BY DEPT_NO HAVING SUM(EMP_SAL) > 45000;

SQL CREATE TABLE Statement

```
CREATE TABLE table_name(
column1 datatype,
column2 datatype,
column3 datatype,
.....
columnN datatype,
PRIMARY KEY( one or more columns )
);
```

EX. CREATE TABLE DEPOSIT (ACTNO VARCHAR2(5) ,CNAME VARCHAR2(18) , BNAME VARCHAR2 (18), AMOUNT NUMBER (8, 2), ADATE DATE);

SQL DROP TABLE Statement

```
DROP TABLE table_name;
```

EX. DROP TABLE DEPOSIT;

SQL DESC Statement

```
DESC table_name;
```

EX. DESC DEPOSIT;

SQL TRUNCATE TABLE Statement (TO REMOVE ALL DATA)

```
TRUNCATE TABLE table_name;
```

EX. TRUNCATE TABLE DEPOSIT;

SQL ALTER TABLE Statement

```
ALTER TABLE table_name {ADD|DROP|MODIFY} column_name {data_ype};
```

EX. ALTER TABLE DEPOSIT ADD (EMP_SSN NUMBER(5,0)); ← TO ADD NEW COLUMN
ALTER TABLE DEPOSIT DROP COLUMN ADATE; ← TO DELETE EXISTIMG COLUMN
ALTER TABLE DEPOSIT MODIFY (ACTNO VARCHAR2(10)) ← TO MODIFY DATA TYPE

SQL INSERT INTO Statement

```
INSERT INTO table_name( column1, column2....columnN)
VALUES ( value1, value2....valueN);
```

EX. INSERT INTO DEPOSIT VALUES(101,'ABC','VRCE', 1000,'1-JAN-96');
INSERT INTO DEPOSIT(ACTNO) VALUES(102);

SQL UPDATE Statement

```
UPDATE table_name
SET column1 = value1, column2 = value2....columnN=valueN
[ WHERE  CONDITION ];
```

EX. UPDATE DEPOSIT SET AMOUNT=2000 WHERE ACTNO=101;

SQL DELETE Statement

```
DELETE FROM table_name
WHERE  {CONDITION};
```

EX. DELETE FROM DEPOSIT WHERE ACTNO=101;

Describe the following SQL functions.

SQL Function	Description	SQL Query Example
Numeric function		
Abs(n)	Returns the absolute value of n.	Select Abs(-15) from dual; O/P : 15
Power (m,n)	Returns m raised to n^{th} power.	Select power(3,2) from dual; O/P : 9
Round (n,m)	Returns n rounded to m places the right of decimal point.	Select round(15.91,1) from dual; o/p : 15.9
Sqrt(n)	Returns square root of n.	Select sqrt(25) from dual; O/P : 5
Exp(n)	Returns e raised to the n^{th} power, e=2.17828183.	Select exp(1) from dual; O/P : 1
Gretest()	Returns the greatest value in a list of values.	Select greatest(10, 20, 30) from dual; O/P : 30
Least ()	Returns the least value in a list of values.	Select least(10,20,30) from dual; O/P : 10
Mod(n,m)	Returns remainder of n divided by m.	Select mod(10,2) from dual; O/P : 0
Ceil(n)	Returns the smallest integer value that is greater than or equal to a number.	Select ceil(24.8) from dual; O/P : 25
ASCII(x)	Returns ASCII value of character.	Select ascii('A') from dual; O/P : 65
Concat()	Concatenates two strings.	Select concat('great','DIET') from dual; O/P : greatDIET
Initcap ()	Changes the first letter of a word in to capital.	Select initcap('diet') from dual; O/P : Diet
Instr ()	Returns a location within the string where search patterns begins.	Select instr('this is test','is) from dual; O/P : 3
Length ()	Returns the number of character in x.	Select length('DIET') from dual; O/P : 4
Lower()	Converts the string to lower case.	Select lower('DIET') from dual; O/P : diet

Upper()	Converts the string to upper case.	Select upper('diet') from dual; O/P : DIET
Lpad()	Pads x with spaces to left to bring the total length of the string up to width characters.	Select lpad('abc',9,'>') from dual; O/P : >>>>>>abc
Rpad()	Pads x with spaces to right to bring the total length of the string up to width characters.	Select rpad('abc',9,'>') from dual; O/P : abc>>>>>>
Ltrim()	Trim characters from the left of x.	Select ltrim('sumita','usae') from dual; O/P : mita
Rtrim()	Trim characters from the right of x.	Select rtrim('sumita','tab') from dual; O/P : sumi
Replace()	Looks for the string and replace the string every time it occurs.	Select replace('this is college','is','may be') from dual; O/P :thmay be may be college
Substr()	Returns the part of string	Select substr('this is college',6,7) from dual; O/P : is coll
Vsize()	Returns storage size of string in oracle.	Select vsize('abc') from dual; O/P : 3
Miscellaneous function		
Uid	Returns integer value corresponding to the UserId.	Select uid from dual; O/P : 618
User	Returns the name of user.	Select user from dual; O/P : admin
Userenv	Returns the information about the current oracle session.	Select userenv ('language') from dual; O/P : AMERICAN_AMERICA.WE8MSWIN1252
Avg(x)	Returns the average value.	Select avg(salary) from employee;
Count(x)	Returns the number of row return by a query.	Select count(deptno) from employee;
Max(x)	Returns the maximum value of x.	Select max(salary) from employee;
Min(x)	Returns the minimum value of x.	Select min(salary) from employee;
Median(x)	Returns the median value of x.	Select median(salary) from employee;
Sum(x)	Returns the sum of x.	Select sum(salary) from employee;
Stddev(x)	Returns standard deviation of x.	Select stddev(salary) from employee;
Variance(x)	Returns variance of x.	Select variance(salary) from employee;
Date functions & conversion function		

To_char	Takes date data type and returns character string according to acceptable format.	Select ldate, to_char(ldate,'dd-mon-yyyy') from demodate; O/P : 1-jan-2011
To_date	Converts x string to date time	Select to_date('1-jan-2011', 'dd-mon-yyyy') from dual; O/P : 1-jan-2011
To_number	Converts character string to number	Select to_number('10') + 20 from dual; O/P : 30
Add_months(x,y)	Gets the result of adding y months to x.	Select add_months('1-jan-2005',1) from dual; O/P : 1-feb-2005
Sysdate	Returns the date of operating system	Select sysdate from dual; O/P : (write today's date)
Last_day	Returns the last day of any month.	Select last_day('01-jan-2005') from dual; O/P : 31-jan-2005
Months_between	Gets the number of months between x and y.	Select months_between('01-jan-2005','01-feb-2005') from dual; O/P : 1
Next_day	Returns date of the next day following x.	Select next_day('01-jan-2005','saturday') from dual; O/P : 08-jan-2005

TO_CHAR Function

Description

The Oracle/PLSQL TO_CHAR function converts a number or date to a string.

Syntax

The syntax for the TO_CHAR function in Oracle/PLSQL is:

```
TO_CHAR( value [, format_mask] [, nls_language] )
```

Parameters or Arguments

value
: A number or date that will be converted to a string.

format_mask
: Optional. This is the format that will be used to convert *value* to a string.

nls_language
: Optional. This is the nls language used to convert *value* to a string.

Applies To

The TO_CHAR function can be used in the following versions of Oracle/PLSQL:

- Oracle 12c, Oracle 11g, Oracle 10g, Oracle 9i, Oracle 8i

Example

Let's look at some Oracle TO_CHAR function examples and explore how to use the TO_CHAR function in Oracle/PLSQL.

Examples with Numbers

For example: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'

Examples with Dates

The following is a list of valid parameters when the TO_CHAR function is used to convert a date to a string. These parameters can be used in many combinations.

Parameter	Explanation
YEAR	Year, spelled out
YYYY	4-digit year
YYY YY Y	Last 3, 2, or 1 digit(s) of year.
IYY IY I	Last 3, 2, or 1 digit(s) of ISO year.
IYYY	4-digit year based on the ISO standard
Q	Quarter of year (1, 2, 3, 4; JAN-MAR = 1).
MM	Month (01-12; JAN = 01).
MON	Abbreviated name of month.
MONTH	Name of month, padded with blanks to length of 9 characters.
RM	Roman numeral month (I-XII; JAN = I).
WW	Week of year (1-53) where week 1 starts on the first day of the year and continues to the seventh day of the year.
W	Week of month (1-5) where week 1 starts on the first day of the month and ends on the seventh.
IW	Week of year (1-52 or 1-53) based on the ISO standard.
D	Day of week (1-7).
DAY	Name of day.
DD	Day of month (1-31).
DDD	Day of year (1-366).
DY	Abbreviated name of day.
J	Julian day; the number of days since January 1, 4712 BC.
HH	Hour of day (1-12).
HH12	Hour of day (1-12).
HH24	Hour of day (0-23).

Parameter	Explanation
MI	Minute (0-59).
SS	Second (0-59).
SSSSS	Seconds past midnight (0-86399).
FF	Fractional seconds.

The following are date examples for the TO_CHAR function.

TO_CHAR(sysdate, 'yyyy/mm/dd')
Result: 2017/08/05

TO_CHAR(sysdate, 'Month DD, YYYY')
Result: August 05, 2017

TO_CHAR(sysdate, 'FMMonth DD, YYYY')
Result: August 05, 2017

TO_CHAR(sysdate, 'MON DDth, YYYY')
Result: 'JUL 09TH, 2003'

TO_CHAR(sysdate, 'FMMON DDth, YYYY')
Result: 'JUL 9TH, 2003'

TO_CHAR(sysdate, 'FMMon ddth, YYYY')
Result: 'Jul 9th, 2003'

You will notice that in some TO_CHAR function examples, the *format_mask* parameter begins with "FM". This means that zeros and blanks are suppressed. This can be seen in the examples below.

TO_CHAR(sysdate, 'FMMonth DD, YYYY')
Result: 'July 9, 2003'

TO_CHAR(sysdate, 'FMMON DDth, YYYY')
Result: 'JUL 9TH, 2003'

TO_CHAR(sysdate, 'FMMon ddth, YYYY')
Result: 'Jul 9th, 2003'

The zeros have been suppressed so that the day component shows as "9" as opposed to "09".

TO_DATE Function

The **TO_DATE** function is used in Oracle to convert a string to a date.

Syntax

The syntax of this function is as follows:

TO_DATE (String, [Format], [NLS Setting])

[NLS Setting] is used to change the output format based on the NLS Territoy and NLS Language (NLS stands for National Language Support). It is optional and is rarely used.

The most important parameter is [Format]. Valid [Format] values are as follows:

Format	Description
AD A.D.	AD indicator to use in conjunction with the year
AM A.M. PM P.M.	Meridian indicator
BC B.C.	BC indicator to use in conjunction with the year
D	Day of week (1-7)
DAY	Name of day
DD	Day of month (1-31)
DDD	Day of year (1-366)
DY	Abbreviated name of day
HH	Hour of day (1-12)
HH24	Hour of day (0-23)
MI	Minutes (0-59)
MM	Month (01-12)
MON	Abbreviated name of month
MONTH	Name of month

RM	Month in Roman Numerals (I - XII)
RR	Accepts a 2-digit input, and returns a 4-digit year. A value between '00' and '49' returns the year in the same century. A value between '50' and '99' returns a year in the previous century.
RRRR	Accepts a 2-digit input or a 4-digit input, and returns a 4-digit year. For 4-digit input, the same value is returned. For 2 digit input, a value between '00' and '49' returns the year in the same century, and a value between '50' and '99' returns a year in the previous century.
SS	Second (0-59)
SSSSS	Seconds past midnight (0-86399)
Y	Accepts a 1-digit input, and returns a 4-digit year in that decade.
YY	Accepts a 2-digit input, and returns a 4-digit year in that century.
YYY	Accepts a 3-digit input, and returns a 4-digit year in that millennium.
YYYY SYYYY	Accepts a 4-digit input, and returns a 4-digits year.

Examples

Below are some examples on using the **TO_DATE** function. For clarity, the results are expressed in the 'YYYY MM DD HH24:MI:SS' format (Year Month Date Hour:Minute:Second, where Hour has a value between 0 and 23):

Example 1

SELECT TO_DATE('20100105', 'YYYYMMDD') FROM DUAL;

Result: 20100105 will be converted in to date format supported by system

05-JAN-10

Example 2

SELECT TO_DATE('1999-JAN-05', 'YYYY-MON-DD') FROM DUAL;

Result:

05-JAN-99

Example 3

SELECT TO_DATE('2005-12-12 03600', 'YYYY-MM-DD SSSSS') FROM DUAL;

Result:

12-DEC-05

3600 seconds equals to 1 hour.

Example 4

SELECT TO_DATE('2005 120 05400', 'YYYY DDD SSSSS') FROM DUAL;

Result:

30-APR-05

April 30th is the 120th day in 2005. 5400 seconds equals to 1 hour and 30 minutes.

Example 5

SELECT TO_DATE('99-JAN-05', 'YY-MON-DD') FROM DUAL;

Result:

05-JAN-99

The 'YY' format converts the year to the current century.

Example 6

SELECT TO_DATE('99-JAN-05', 'RR-MON-DD') FROM DUAL;

Result:

05-JAN-99

The 'RR' logic converts '99' to the previous century, hence the result is 1999.