CHAPTER-3 SQL(Structured Query Language)

3.1 Data Types

- **TYPES DATATYPE**
  - **BIT(n)**
  - **CHAR(size)**
  - **VARCHAR(size)/VARCHAR2(size)**
  - **DECIMAL(p, s) or NUMBER(P,S)**
  - **INTEGER :** integer number
  - **FLOAT(P):** floating point number with precision equal to or greater than 'p'
  - **DATE**
  - **TIME :** expressed as HH:MM:SS
  - **TIMESTAMP :** absolute time expressed as DD:MM:YYYY HH:MM:SS
  - **LONG**
  - **RAW/LONG RAW**

- **BIT(n)**
  - Fixed length bit string of 'n' bits numbered 1- n.

- **CHAR(SIZE)**
  - This data type is used to store character strings values of fixed length.
  - The size in brackets determines the number of character the cell can hold.
  - The maximum number characters this data type can hold is 255 characters
  - Example: Name CHAR(60)

- **VARCHAR(size)/VARCHAR2(size)**
  - This data type used store variable length alphanumeric data.
  - It is more flexible form of the CHAR data type.
  - The maximum this data type can hold up to 4000 characters.
  - VARCHAR can hold 1 to 255 characters.
  - That CHAR is much faster than VARCHAR, sometimes up to 50%.

- **DATE**
  - This data type is used to represent date and time.
  - The standard format is DD-MON-YY as in 21-JUN-09.
  - Date time stores date in the 24-hours format.
  - Valid dates range from January 1, 4712 B.C. to December 31, 4712 A.D.

- **DECIMAL(p,s) or NUMBER(P,S)**
  - The NUMBER data type is used store numbers (fixed and floating point).
  - Number of virtually any magnitude maybe stored up to 38 digits of precision

- Number may be expressed in two ways: first, with number 0 to 9, the sign + and -, and a decimal point (.); second, in scientific notation, such as, 1.85E3 for 1850.
- The precision (P), determines the maximum length of data, whereas the scale(s), determines the number of places to the right of the decimal.

➢ **LONG**
- This data type is used to store variable length character strings containing 2 GB.
- LONG data can be used to store arrays of binary data in ASCII format.

➢ **RAW/LONG RAW**
- The RAW/LONG RAW data type are used to store binary data, such as digitized picture or image
- RAW data type can have a maximum length of 255 bytes.
- LONG RAW data type can contain up to 2 GB.

3.2 DDL(Data-Definition Language)

➢ **"**We specify a database schema by a set of definition expressed by a special language called a **data-definition language(DDL)"**
➢ **CREATE, ALTER, TRUNCATE, DROP TABLE**

i. CREATE

- The CREATE statement is used to create different database objects like table, view, function trigger etc
- Here we are creating table with CREATE statements. It specifies table name, each column name, data type, size etc.

Syntax:

```
CREATE TABLE <table name>
        (
        <columnname1> <datatype> (<size>),
        <columnname2> <datatype> (<size>)
        );
```

Example:

```
CREATE TABLE client_master
(
        client_no  varchar2(6),
        name  varchar2(20),
        city  varchar2(15),
        pincode number(8),
        state varchar2(15)
);
```

ii. ALTER

- The structure of a table can modified by using ALTER TABLE command
- With ALTER TABLE it is possible to add or delete columns, create or destroy indexes, change the data type of existing columns, or rename columns or the table itself.

Syntax1: Adding New Columns

ALTER TABLE <TableName>
ADD (<NewColumnName <Datatype> (<size>),
<NewColumnName <Datatype> (<size>)….);

Example: ALTER TABLE client_master ADD (telephone_no number(10));

Syntax2: Modifying Existing Columns

ALTER TABLE <TableName>
MODIFY(<ColumnName> <NewDatatype>(<NewSize>));

Example: ALTER TABLE product_master MODIFY (SELL_PRICE number(10,2));

iii. TRUNCATE TABLE

- TRUNCATE TABLE empties a table completely
- Logically, this is equivalent to DELETE statement that deletes all rows
- TRUNCATE operations drop an re-create the table, which is much faster than deleting rows one by one
- TRUNCATE operations are not transaction-safe
- The number of deleted rows are not returned

Syntax: TRUNCATE TABLE <TableName>

Example: TRUNCATE TABLE client_master;

iv. DROP TABLE

- Such situation using the DROP TABLE statement with table name can destroy a specific table.

Syntax: DROP TABLE <TableName>

Example: DROP TABLE client_master;

3.3 DML (Data-Manipulation language)

- **INSERT,SELECT,UPDATE,DELETE**

Types of DML:-

- **Procedural DML**
  - Require a user to specify what data are needed and how to get those data
  - SELECT statement in DML used for data retrieval is also known as Data Query Language (DQL).
- **Nonprocedural(Declarative) DML**
  - Require a user to specify what data are needed without specifying how to get those data

i. INSERT

- When inserting a single row data into the table, the insert operation:
  - Creates a new row in the database table
  - Loads the values passed into the column specified

Syntax1: Inserting value in all the fields of table

```
INSERT INTO <table name>
VALUES (value1,value2…….);
```

Example: INSERT INTO client_master (client_no,name)
Values ('C00001','RAHUL');

Syntax2: Inserting value in all the fields of table at runtime, means asking for values from user

```
INSERT INTO <table name>
VALUES (&columnname1,&columnname12…….);
```

Example: INSERT INTO client_master
Values ('&client', '&name');

Syntax3: Inserting value in selected fields of table

```
INSERT INTO <table name> (<columnname1>,<columnname12>)
VALUES (value1,value2…….);
```

Example: INSERT INTO client_master (client_no,name)
Values ('C00001','RAHUL');

ii. UPDATE

- The UPDATE command is used to change or modify data values in a table
- The verb update in SQL is used to either update:
    1. All the rows from table
    2. A select set of rows from table

Syntax1: updating all the rows of table

```
UPDATE <table name>
    SET <columnname1> = <expression1>,
        <columnname1> = <expression2>;
```

Example:
```
UPDATE client_master
SET city='Bombay';
```

Syntax2: updating selected rows of table

```
UPDATE <table name>
    SET <columnname1> = <expression1>,
        <columnname1> = <expression2>
    WHERE columnname=expression;
```

Example:
```
UPDATE client_master
SET city='Bombay'
WHERE cname='ravi';
```

iii. DELETE

- The DELETE command deletes rows from the table that satisfies the condition provided by its where clause, and returns the number of records deleted.
- The verb DELETE in SQL is used to remove either:
    1. All the rows from table
    2. A select set of rows from table

Syntax1: delete All the rows from table

```
DELETE FROM <table name>;
```

Example:
```
DELETE FROM client_master;
```

Syntax2: delete selected set of rows from table

```
DELETE FROM <table name> WHERE <Condition>;
```

Example: DELETE FROM client_master WHERE client_no='C00001';

iv. SELECT

- The SELECT command is used to retrieve rows selected from one or more tables

Syntax1: retrieving all the rows and all columns from the table

SELECT * FROM <TableName>;

Example: SELECT * FROM client_master;

Syntax2: retrieving selected columns from the table

SELECT <ColumnName1>,<ColumnName1>
FROM <TableName>;

Example: SELECT name, city FROM client_master;

Syntax3:retrieving selected rows from the table

SELECT * FROM <TableName> WHERE <Condition>;

Example: SELECT * FROM client_master
WHERE city='Bombay';

3.4 Operators

❖ **Arithmetic Operators**

➢ Oracle allows arithmetic operators to used while viewing records from table or while performing data manipulation operations such as Insert, Update and Delete. These are:

+ Addition
* Multiplication
- Subtraction
/ Division

Example: SELECT product_no, description, sell_price * 0.05
FROM product_master;

- **Logical Operators**
  - **The AND Operator**
  - **The OR Operator**
  - **The NOT Operator**

  - **The AND Operator**
    - The AND Operator allows creating an SQL statement based on two or more conditions being met.
    - It can be used in any valid SQL statement such as select, insert, or delete.

    - Example: SELECT product_no, description, sell_price * 0.05
      FROM product_master
      WHERE sell_price BETWEEN 500 AND 1100;

  - **The OR Operator**
    - The OR condition allows creating an SQL statement where records are returned when any one of the conditions are met
    - It can be used in any valid SQL statement such as select, insert, or delete.

    Example: SELECT client_no, name, city, pincode
      FROM client_master
      WHERE pincode=4000054 OR pincode=4000057;

  - **The NOT Operator**
    - The Oracle engine will process all rows in a table and display only those records that do not satisfy the condition specified

    Example: SELECT *
      FROM client_master
      WHERE NOT (city='Bombay' OR city='Delhi');

  - **Combining the AND and OR Operator**
    - The AND and OR conditions can be combined in a single SQL statement
    - It can be used in any valid SQL statement such as select, insert, or delete.

- **Comparison Operators**
  - Comparison operators are used in condition to compare one expression with other. The comparison operators are =, <, >, >=, <=, !=, BETWEEN, LIKE, IS NULL and IN operators

  - **BETWEEN operator** is used to check between two values

    - Example: SELECT * FROM salesman_master
      WHERE salary BETWEEN 5000 AND 8000;

- The above select statement will display only those rows where salary of salesman is between 5000 and 8000.

➢ **IN operator:**
- The IN operator can be used to select rows that match one of the values in a list
- Example: SELECT * FROM client_master
  WHERE client_no IN (C00001, C00003);
- The above query will retrieve only those rows where client_no is either in C00001 or C00003

➢ **LIKE operator:**
- Like operator is used to search character pattern, we need not know the exact character value. The **LIKE** operator is used with special character % and _(underscore)

  Example: SELECT * FROM client_master WHERE city LIKE ' b% ';
- The above select statement will display only those rows where city is start with 'B' followed by any number of any characters.
- % sign is used to refer number of character (it similar to * asterisk wildcard in DOS).
- While _(underscore) is used to refer single character.

  Example: SELECT * FROM client_master WHERE name LIKE '_ahul';

3.5 SQL functions

❖ **Single row functions**

➢ **Date function**

- **ADD_MONTHS()**
  - It returns the date after adding the number of months specified within the function
  - Syntax: SELECT ADD_MONTHS(date,n) FROM dual;
  - Example: SELECT ADD_MONTHS ('04-jun-2009',2) FROM dual;
  - Output: 04-AUG-2009

- **MONTHS_BETWEEN**
  - It returns the date the number of months between the specified dates.
  - Syntax: SELECT MONTHS_BETWEEN(date1,date2) FROM dual;
  - Example: SELECT MONTHS_BETWEEN ('02-FEB-09','02-JAN-09')
    FROM dual;
  - Output: 1

- **Round()**
  - It rounds a date to 12 A.M(midnight) if time of the date is before noon, otherwise it rounds up to next day.
  - Syntax: SELECT ROUND(date) FROM dual;
  - Example: SELECT ROUND(14-08-09) FROM dual;
  - Output:-3

- **NEXT_DAY**
  - It returns the date of the first weekday named by 'char' that is after the date named by 'date'. 'char' must be day of the week.
  - Syntax: SELECT NEXT_DAY(date,'char') FROM dual;
  - Example: SELECT NEXT_DAY('04-FEB-09','FRIDAY') FROM dual;
  - Output: 06-FEB-09

- **TRUNC()**
  - It always sets a date to 12 A.M (midnight).
  - Syntax: SELECT TRUNC(date) FROM dual;
  - Example: SELECT TRUNC('02-08-09') FROM dual;
  - Output:-15

- **GREATEST()**
  - It displays the latest date from a list of dates
  - Syntax: SELECT GREATEST (date1, date2, date3 …) FROM dual;
  - Example: SELECT GRETEST('02-FEB-09', '02-JAN-09') FROM dual;
  - Output: 02-JAN-09

- **NEW_TIME()**
  - It display the date in the required time zone
  - Syntax: : SELECT NEW_TIME (date,currenttimezone,newtimezone) FROM dual;
  - Example: SELECT NEW_TIME(sysdate,'EST','HST') FROM dual;
  - Output: 22-JUL-09

➢ **Numeric function**

- **ABS()**
  - It returns the absolute value.
  - Syntax: SELECT ABS(n) FROM dual;
  - Example: SELECT ABS(-15) FROM dual;
  - Output: 15

- **CEIL()**
  - It returns the smallest integer that is greater than or equal to a specific value
  - Syntax: SELECT CEIL(n) FROM dual;

- Example: SELECT CEIL(1.3) FROM dual;
- Output:2
- Example: SELECT CEIL(2) FROM dual;
- Output:2
- Example: SELECT CEIL(-2.3) FROM dual;
- Output:2

- **COS()**
  - It returns the cosine of a given value.
  - Syntax: SELECT COS(no) FROM dual;
  - Example: SELECT COS(45) FROM dual;
  - Output: 1

- **COSH()**
  - It returns hyperbolic cosine of a given value.
  - Syntax: SELECT COSH(no) FROM dual;
  - Example: SELECT COSH(45) FROM dual;
  - Output: 1.7467E+19

- **EXP()**
  - It returns **e** raised to the **nth** power, where e=2.71828183.
  - Syntax: SELECT EXP(n) FROM dual;
  - Example: SELECT EXP(5) "Exponent" FROM dual;
  - Output: Exponent
    148.413159
- **FLOOR()**
  - It returns the greatest integer that is less than or equal to a specific value.
  - Syntax: SELECT FLOOR(n) FROM dual;
  - Example: SELECT FLOOR(1.3) FROM dual;
  - Output: 1
  - Example: SELECT FLOOR(2) FROM dual;
  - Output: 2
  - Example: SELECT FLOOR(-2.3) FROM dual;
  - Output: -3

- **POWER()**
  - It returns the value raised to a given positive exponent.
  - Syntax: SELECT POWER(value, exponent) FROM dual;
  - Example: SELECT POWER(3,2) FROM dual;
  - Output: 9
  - Example: SELECT POWER(64,0.5) from dual;
  - Output: 8

- **MOD()**

- It divides a value by a divisor and returns the remainder.
- Syntax: SELECT MOD(value, divisor) FROM dual;
- Example: SELECT MOD(100,10) FROM dual;
- Output: 0
- Example: SELECT MOD(10,3) FROM dual;
- Output: 1
- Example: SELECT MOD(-30.23,7) FROM dual;
- Output: -2.23

▪ **ROUND()**
- It rounds a number to given number of digits of precision
- Syntax: SELECT ROUND(value, precision) FROM dual;
- Example: SELECT ROUND(66.666,2) FROM dual;
- Output: 66.67

▪ **TRUNC()**
- It truncates digits of precision from a number
- Syntax: SELECT TRUNC(value, precision) FROM dual;
- Example: SELECT ROUND(66.666,2) FROM dual;
- Output: 66.66

▪ **SORT()**
- It returns the square root given number
- Syntax: SELECT SQRT(n) FROM dual;
- Example: SELECT SQRT(64) FROM dual;
- Output: 8

➢ **Character function**

▪ **INITCAP()**
- Returns a string with the first letter of each word in UPPER CASE
- Syntax: SELECT INITCAP(char) FROM dual;
- Example: SELECT INITCAP('rahul kumar') FROM dual;
- Output: Rahul Kumar

▪ **LOWER()**
- It takes any string or column and converts it into lowercase
- Syntax: SELECT LOWER(string) FROM dual;
- Example: SELECT LOWER(‘RAHUL’) FROM dual;
- Output: rahul

▪ **UPPER()**
- It takes any string or column and converts it into upper case
- Syntax: SELECT UPPER(string) FROM dual;
- Example: SELECT UPPER(‘rahul’) FROM dual;

- Output: RAHUL

- **LTRIM()**
  - It removes character from the left of char with initial characters removed up to the first character not in set
  - Syntax: SELECT LTRIM(char, set) FROM dual;
  - Example: SELECT LTRIM('RAHUL','R') FROM dual;
  - Output: AHUL

- **RTRIM()**
  - It returns char with final character removed after the last character not in the set. 'set' is optional, it defaults to spaces.
  - Syntax: SELECT RTRIM(char, set) FROM dual;
  - Example: SELECT RTRIM('RAHUL','L') FROM dual;
  - Output: RAHU

- **TRANSLATE()**
  - Replace a sequence of character in a string with another set of character.
  - Syntax: SELECT TRANSLATE(string1,string to replace, replacement string) FROM dual;
  - Example: SELECT TRANSLATE('1sct523','123','7a9') FROM dual;
  - Output: 7sct5a9
- **REPLACE()**
  - It replaces a character in a string with zero or more character.
  - Syntax: SELECT REPLACE(string1, character to be replaced, characters) FROM dual;
  - Example: SELECT REPLACE('Hello', 'o', 'rec') FROM dual;
  - Output: hellrec

- **SUBSTRING()**
  - It returns a substring from a given string
  - It returns a portion of char, beginning at character m exceeding up to n characters.
  - Syntax: SELECT SUBSTR(string) FROM dual;
  - Example: SELECT SUBSTR('SECURE',3,4) FROM dual;
  - Output: CURE

- **Conversion function**

  - **TO_CHAR()**
    - It converts a value of DATE data type to CHAR value. It accept a date as well as the format in which the date has to appear. The format must be a date format
    - Syntax: SELECT TO_CHAR(date, format) FROM dual;
    - Example: SELECT TO_CHAR(SYSDATE,'DD-MM-YY') FROM dual;

- Output: 22-07-09

- **TO_DATE()**
  - It converts a CHAR filed to a DATE filed.
  - Syntax: SELECT TO_DATE ('char', format) FROM dual;
  - Example: SELECT TO_DATE('26/08/09','DD/MM/YY') FROM dual;
  - Output: 26-AUG-09

- **TO_NUMBER()**
  - It converts a character value containing a number to a value of NUMBER data type.
  - Syntax: SELECT TO_NUMBER('char') FROM dual;
  - Example: SELECT TO_NUMBER('100') FROM dual;
  - Output: 100

➢ **Miscellaneous function**

- **UID**
  - It returns an integer that uniquely identifies the session user.
  - Example: SELECT UID FROM dual;
  - Output: 18

- **USER**
  - It returns the name by which the current user is know to Oracle.
  - Example: SELECT USER FROM dual;
  - Output: scott

- **NVL**
  - It stands for Null value Substitution.
  - Syntax: SELECT NVL(value, substitute)
  - If the value is NULL, this function is equal to substitute.
  - If the value is not NULL, this unction is equal to value.
  - Example

Table name: Shipping

Client	**Weight**
Johnson tools	59
Inf Software	27
Peterson Industries	**NULL**

SELECT NVL (weight, 43) FROM shipping;

Client	**Weight**
Johnson tools	59

Inf Software	27
Peterson Industries	**43**

In the above output, the NVL function replaces the value specified in wherever it encounters a NULL value. Hence, 43 is inserted for client Peterson Industries.

- **VSIZE**
  - It returns the number of bytes in the internal representation of an expression.
  - Syntax: SELECT VSIZE(expression) FROM dual;
  - Example: SELECT VSIZE('SCT on the net') FROM dual;
  - Output: 14

➢ **Group function**

- **AVG()**
  - It returns average value of the specified column, ignoring NULL values.
  - Syntax: AVG(Column name)
  - Example: select AVG(sell_price) FROM product_master;
  - Output: 2012.345

- **MIN()**
  - It returns the minimum value in the specified column.
  - Syntax: MIN(column name)
  - Example: select MIN(sell_price) FROM product_master;
  - Output: 250

- **MAX()**
  - It returns the maximum value in the specified column.
  - Syntax: MAX(column name)
  - Example: select MAX(sell_price) FROM product_master;
  - Output: 1500

- **SUM()**
  - It returns the sum of values of the specified column.
  - Syntax: SUM(column name)
  - Example: select SUM(salary) FROM salesman_master;
  - Output: 65000

- **COUNT()**
  - It returns the number of rows in specified column or the total number of rows in the table.
  - Syntax: COUNT(column name)
  - Example: select COUNT(salesman_no) FROM salesman_master;
  - Output: 15

- Syntax: COUNT(*)
- Example: select COUNT(*) FROM salesman_master;
- Output: 50

3.6Group by, Having by and order by clause

- **GROUP BY Clause**
  - **GROUP BY** clause is another section of the select statement
  - This optional clause tells Oracle to group rows based on distinct values that exist for specified columns
  - The **GROUP BY** clause creates a data set, containing several sets of records grouped together based on a condition

  - Syntax:
    Select <column name1>,<column name2>, <column nameN>
    AGGREGATE_FUNCTION (<Expression>)
    From Table Name WHERE <condition>
    GROUP BY<column name1>,<column name2>,<column nameN>;

  - Example:
    SELECT branch_no"branch_no",COUNT(emp_no)"No Employees"
    FROM emp_master GROUP BY branch_no;

- **HAVING Clause**
  - **HAVING** Clause can be used in conjunction with the GROUP BY clause HAVING imposed a condition on the GROUP BY clause.
  - which further filter the group created by the groups by clause
  - Syntax:
    Select <column name1>,<column name2>, <column nameN>
    AGGREGATE_FUNCTION (<Expression>)
    From Table Name WHERE <condition>
    GROUP BY<column name1>,<column name2>,<column nameN>;
    HAVING <condition>
  - Example:
    Select product_no,sum (qty_ordered) "total QTY ordered"
    FROM sales_order GROUP BY product_no
    HAVING product_no='P00001' or product_no='P00004'
  - **HAVING** clause is specified row displayed on screen.

- **ORDER BY Clause**
  - **ORDER BY** is user want the information in the order specify.
  - Syntax:
  - Select <column name1>,<column name2>, <column nameN>
    FROM Table Name Where <condition>
    ORDER BY <column name1>;

- Example:
  Select feature, section, page FROM NEWSPAPER
  Where section='F'
  ORDER BY feature;
    - Sort in descending order.
      Select feature, section, page FROM NEWSPAPER
      Where section='F'
      ORDER BY feature desc;