28/12/22

Unit-1

→ Introduction to 'SQL'

SQL:

* It Stands for 'Structured Query Language'
* It is composed of Commands that allows user to create data base & table structure and also performs various types of data manipulation
* SQL functions are divided into 2 categories
  (i) Data definition language (DDL).

- it includes commands to create database objects like tables, index & view

Commands	Description
• CREATE TABLE	• Creates a new Table
• CREATE INDEX	• Creates a index for a Table
• CREATE VIEW	• Creates a dynamic Sub set of Table
• DROP TABLE	• Permanently deletes a Table
• DROP INDEX	• permanetly deletes index.
• DROP VIEW	• permanetly deletes a view.

(ii) Data manipulation Language: (DML).

SQL includes commands to insert, update delete & retrive data within the database table.

Command.	Description.
• INSERT	• it is use to insert row in a Table
• UPDATE	• modifies an attribute value in a table.
• Delete	• Deletes one a more rows from table.
• SELECT	• Retives data from table.
• COMMIT	• premanetly Saves data changes.
• ROLLBACK	• Restores data to there original values
• Comparison operator (<,>, ≤, ≥, =)	• Used in conditional exprension
• Logical operation (E, O, NOT)	• used in condit² exp.

- Special operator • used in condi^{n} exp.

⇒ SQL Data Type

- SQL provides following Data types.

Datatype	format	Descripti^{n}	Example
Numeric	Numbe(L,D)	it stores No. with decimal places	Numbe(7,2) 123.25.
	integer(int)	use to store whole CountingNo.	int(10) 798
	Smallint	it is same as integer but limited to 6 digits.	Smallint (3) 12.
	decimal(L,D)	it is same as No. but length is a min. req.	decimal(5,2) 12.25
Characte	Char(L)	it is fixed length chau. data upto 255 characte. →	BCA The unused space cannt be use by othe

varchar2(L)	it is variable length char. Data. it will not leave unused spaces.	varchar2(5) BCA	
Date	date	it Stores date in 'Julian' date format	28-DEC-2022

⇒ Creating Table Structure.

- To create a Table, 'CREATE TABLE' command is used.

Syntax:

```
CREATE TABLE tablename (
    Column1 datatype [constraint],
    Column2   "       "
    :
    :
    Columnn datatype [Constraint]);
```

Eg: Create table Student (
Rollno. int (3),
name varchar2(10),
dob date,
gender char(1),
marks number(3));

⇒ SQL Constraints:

- The constraints are used to validate data in SQL
- There are 6 types of Constraints used in SQL.

(i) Primary Key:

- it is used to ensure that the column has unique value only.
- it doesnot accept NULL or duplicate value.
- it is used to identify rec. in a table

Syntax:

columnname datatype PRIMARY KEY;
or
PRIMARY KEY (Columnname),

Eg. ROLLNO int(3) PRIMARY KEY;

(ii) Not NULL:

- It is used to unsure that the column doesnot accept NULL value

Syntax:

Columnname datatype NOTNULL;

eg: name varchar2(10) notnull;

(iii) UNIQUE.

- it unsures that all the values in a col. are unique.
- it is same as P.K. but it accepts null val

Syntax:

Columnname datatype UNIQUE;

eg: Course varchar2(5) unique;

(iv) DEFAULT:

- it assignes a value to an athibute when a new row is added to a table

Syntax:

Columnname datatype DEFAULT 'value';

eg: Course varchar2(5) default 'BCA';

(v) CHECK:

- it is use to validate data when the athibute value is entred.

Syntax:

Columnname datatype CHECK(Columnname IN ('value1', 'val2'));

eg: gender char(1) check (gender IN ('m', 'F'));

(vi) FOREIGN KEY.

- it is use to give ref. of the other table.
- it can be duplicate or NULL.

Syntax:

FOREIGN KEY (colname) REFERANCE tablenam (colname);

Eg: FOREIGN KEY (cid) reference course(cid));

→ Create product table with fields
P-id, P-name, price, Qty., Mfg.date,
v-id., Category.

Create vendor table with v-id, v-name.

Constraint Apply P.K & F.K
Name shouldn't be NULL
product category should be food or
Stationery.

→ Create table vendor (

v-id number(3) primary key,
vname varchar2(10) not null);

Create table product (

p-id int.(3) primary key,
pname varchar 2(10) not null,
price decimal (7,2),
Qty int (3)
mfgdt date,
category varchar2(10) check (cat in
('food', 'stat...'))
v-id number (3)
foreign key (v-id) reference ~~key~~ vendor
(v-id));

→ Data manipulation Commands

* Adding table rows

the 'INSERT' command is use to enter data into table.

INSERT INTO tablename VALUES ('value1', value2 ...);

or

INSERT INTO tablename (colmn1, colum 2) VALUES ('value1', 'value2'...);

or

* INSERT INTO tablename VALUES (&value1, &val2'...);

eg: insert into Student values (1, 'A', 45, 1);
insert into Student (Sid, Sname, mark, cid) values (2, 'B', 37, 1);
insert into Student values (&sid, '&sname', &mark, &cid); [/]

* Saving Table changes:

Any changes made to the table, table contents are not saved so,
to save any changes in table content 'COMMIT' command is used.

COMMIT;

eg Commit;

* Listing Table rows:

The 'SELECT' Command is used to list the content of a Table

SELECT Columnlist From tablename;

The Columnlist represents one or more attributes seperated by ,.
The use of (*) Symbol displays all the attributs.

→ list all the rec. of Students.

SELECT * from Student;

→ Display Student name & there marks.

SELECT Sname, marks from Student;

* Updating table rows:

the 'UPDATE' Command is use to modify data in a table

UPDATE tablename
SET Columnname = value
[Where Condition];

eg: UPDATE marks of STUDENT with 60.

```
UPDATE Student
Set mark=60;
```

Modify marks of Students as 57 whose Cid is 1.

```
UPDATE Student
Set mark=57
[WHERE Cid=1];
```

* Restoring table content:

if we have not used 'COMMIT' command to store the changes permanently.
we can restore the data base to its previous value using 'ROLLBACK' command.
the 'ROLLBACK' undos any changes in the table content.

eg: ROLLBACK;

* DELeting table rows:

The DELETE command is use to delete a table row from the specified table.

```
DELETE From tablename
[Where Condition];
```

delete Students who belongs to Bcom Course.

```
DELETE from Student
WHERE Cid=2;
```

⇒ Select Queries:

The SELECT command is use to retrive data from the table by adding restriction to the search criteria.

(i) Selecting rows with conditional restriction

We can select the partial table content by placing restriction.
the use of 'WHERE' clause to Add condition restriction to the 'SELECT' Statement.

Operators	Desciption.
<	less than
<=	less than equal to
>	greater than
>=	greather than equal to
<>, !=	not equal to

```
SELECT Columnlist
FROM tablename
[WHERE Condition];
```

eg: display sid, name, marks of students whose marks are < 48.

```
SELECT sid, marks, name
FROM STUDENT
WHERE marks < 48 ;
```

eg: display student except Bcom course.

```
SELECT *
From STUDENT
WHERE Cid != 2 ;
```

(iii) Arithmetic operators:

We can select the table content by placing restriction on the rows using Arithmetic op. in WHERE clauses.

Operator	Des.
+	Add
-	Sub
*	Multiply
/	divide
^	Power

```
SELECT Columnlist
from Tablename
[WHERE Condition];
```

eg: Add 5 marks in each student rec.

```
SELECT Sid, Sname, marks+5, cid
    from Student;
```

eg. ~~display~~ deduct 10 marks from STUDENT with cid = 2

```
SELECT Sid, Sname, marks-10, cid
    from Student
    WHERE Cid=2;
```

(iii) Logical Operators:

Sometimes the table content req. multiple condit^n.
For that logical operators are used
there are 3 logical op. used in SQL
AND, OR, NOT.

- AND: The and operator displays records when both condit^n are true.

```
SELECT Columnlist
from tablename
WHERE Condition1 And condition2;
```

display STUDENTS whose marks > 40 & belongs to BCA.

```
SELECT * from STUDENT
where marks >40 AND Sid=1;
```

- OR: the OR operator displays rec. when one of the condition is true.

```
SELECT Columnlist
from tablename
WHERE Condit^n 1 OR Condit^n 2;
```

eg. As Above

```
SELECT * from Student
where marks >40 OR Sid=1;
```

- NOT: The logical not is use to reverse the result of conditional expression it is implemented on one column only.

```
SELECT Columnlist
from tablename
WHERE NOT condition;
```

Display Student other than 3.

```
SELECT * from Student
WHERE NOT Sid=3;
```

→ Special operators:

SQL Allows the use of special operator in conjuction with WHERE clause.
The special op are.

- BETWEEN
- IS NULL
- LIKE
- IN
- EXISTS

- The BETWEEN op. is use to check weather the attribute value is within the range of value.
  it is used in conjaction with logical AN

```
SELECT Colndist
from tablename
WHERE Colname BETWEEN vale1 AND val2
```

eg: display Sid, name & mark of student Bet° the range 30 to 40

```
SELECT Sid, Sname, mark, cid
  from STUDENT
  WHERE marks BETWEEN 30 AND 40.
```

- IS NULL is use to chek wether An Attribute value is null.

```
SELECT Columnlist
from tablename
WHERE Columnname IS NULL;
```

```
SELECT * from STUDENT
where cid IS NULL;
```

- LIKE is used in Conjuction with wildcard Symbols to find pattrn within String Attributs.

SQL Allows two wildcard characters
% , _

% means Any & All preceding or following characters are Allowed.

_ means Any one character may be Substituted.

```
SELECT Columlist
from tablename
WHERE Columname LIKE 'Pattrn';
```

eg: _a%
%a_
%a%

eg: whose name starts with 'A'.

```
SELECT * from student
WHERE Sname LIKE 'A%';
```

eg: name ends with 'a'.

```
SELECT * from student
WHERE Sname LIKE '%a';
```

eg: Sid, name of student whose name has any letter 'A'.

```
SELECT Sid, Sname
from Student
where Sname LIKE '%a%';
```

eg: list of course where the sec. letter of course should be B.

```
Select * from Course
where Cname like '_B%';
```

- IN operator uses a value list specified within a it.
it is same as logical or But IN op. uses a single col. for multiple restriction

```
SELECT Columnlist
from tablename
WHERE Columnname IN ('val1', 'val2');
```

→ Where Sid is either 1 or 2 using in op.

```
Select * from Student
  Where Sid in ('1', '2');
```

- Exists operator can be used when there is a requirement to execute the command based on the result of another Query.

if a Sub-Query returns any row then only the main Query is executed, other wise not.

Syntax:

```
SELECT * From tablename
  WHERE EXISTS (subquery);
```

Display rec of Stud. In those marks < 50 using Exists op.

SELECT * from student
where marks < 50;

or.

SELECT * from student
where exists (Select * from student
where marks < 50);

→ Additional Data Definition commands:

The ALTER TABLE Command is used to change table structure

it is used by the keyword that produces specific change the user want to make

Three options are available - ADD, MODIFY, DROP.

The ADD KEYWORD adds a col, MODIFY changes col. characteristic & DROP del. a col.

- **ADD:** it is use to add column or constraint in a table.

```
ALTER TABLE tablename
    ADD columnname datatype;
        or
ALTER TABLE tablename
    ADD Constraint Constraintname;
```

eg: ADD col. of DOB to Student table & also add Primary Key Cont.

```
alter table Student
    ADD (DOB Date);

alter table Student
    add Primary Key (Sid);
```

- **MODIFY:** the modify is use to change column characteristic like its Datatype or Size.

```
ALTER TABLE tablename
    MODIFY columnname datatype;

alter table Student
    Modify Sid no.(5);
```

- DROP: DROP Keyword del. a col or con. from table

```
ALTER TABLE tablename
  DROP Column Columnname / Constraint
                          Constname;
```

```
ALTER TABLE Student
  DROP Column DoB;
```

(i) Changing column's datatype:

- The ALTER TABLE Command with MODIFY Keyword is use to change cols. datatype
- The datatype can only be change if the col. is empty.

```
Altr table tablename
  modify col-name datatype;
```

```
altr table Student
  modify sname char(10);
```

(ii) Changing col. data characteristics.

The Altr table Command with MODIFY Keyword is use to change Cols Characteristic.

Alter table Student
modify (Marks decimal (3,0));

(iii) Adding a column:

- The Alter Table command with add Key-word is use to add a column to the table.

Alter Table tablename
ADD (Columnname datatype (size));

Alter Table Student
Add (DOB date);

(iv) DROPing a column:

- The Alter table command with drop keyword is use to delet a col. from a table.

Alter table tablename
DROP colname colname;

Alter table Student
DROP (DOB Col.);

(v) Advance data update:

The use of update command is use to make changes to the data in the column of existing rows.

```
UPDATE tablename
SET col.name = value
[WHERE Cond^n]
```

⇒
```
UPDATE Student
SET Mark = 38
Where Sid = 2;
```

(vi) Adding primary & foreign key destination

The Alter table command ~~is use to~~ with add keyword is use to add P.K & F.K.

```
Alter table tablename
Add PK (col),
Add FK (col.) Reference table
                name;
```

alter table Student
add primary key (sid),
add foreign key (cid) refere
courses;

(vii) Deleting a table from the database:

A table can be deleted from DB using DROP Table command.

eg: DROP table Student;

(viii) Coping Parts of a table:

Sometimes it is necessary to Break a table Structure into its componant part.

In coping table, first ~~tat~~ that table is created.

eg: Create table Student((
Sid No(3) primary key,
Sname varchar(10) notnull,
marks No(2),
Cid No.(3),
foreign key (cid) reference course
(cid));

The insert command is use to insert data in the table But to copy that data from pre. table follow syntax is used.

```
INSERT INTO target-tablename (col list)
SELECT source- col.list
FROM.  "  -  "name";
```

→ Additional Select Query Keywords.

* Ordering a listing: the ORDER BY command is use to display data in either ascending or decending order.
By default it displays data in ascending order.

To Display in it dec. order 'DESC' Keyword is used

```
SELECT Column list from tablename
[Where condition]
[ORDER By col.name ASC/DESC];
```

* Listing unique values:

The DISTINCT command is use to display unique values from table

Select distinct col. From Tablename [Where Condit^{n}];

Select distinct cid from Student;

* Agrigate functions:

The Agrigate functions are

→ Sum, min, max, Avg, COUNT.

SUM: This function display total of the Specific col. from table

Select Sum (col.) from tablename [When Condit^{n}];

Min/Max: The min funt^{n} ret min. value of a col. while the max funt^{n} ret. max. value from the col.

Select MIN(Col) from tablename
[where condit^n]

Select MAX(Col) from tablename
[where condit^n]

Avg: The Avg function displays Average of the specified col.

Select Avg(Col) from tablename
[where condit^n];

Select Avg marks from Student
where cid=2;

COUNT: The Count funct^n counts total No. of rec. from the table

Select Count(col) from Tablename
[where condit^n];

Select Count * from Student
where cid=1;

eg. Unique cid. & count them

Select Count(distinct cid) from
Student;

→ display inserted values

Select * from course.

→ data in Student table.

Insert into Student values (&sid, '&sname', '&dob', '&gender', &marks, &cid)

/

Select * from Student

→ Display Sid name & marks of male Students

Select Sid, Sname, marks from Student where gender = 'm';

→ Display rec. of Students other than BCA Course

Select * from Student
where cid != 1; (!= or <>) Same

→ Display Sid name gender & marks of male Students whose marks > 40.

~~Select * from Student~~

Select Sid, Sname, gender, marks from Student
where gender = 'm' and marks > 40;

→ Display rec. of students other than BCA course using logical op.

```
Select * from Student
Where not cid = 1;
```

→ Display name & marks of student bet° 40 & 50.

```
Select Sname, marks from Student
Where marks between 40 and 50;
```

→ list students whose cid is null

```
Select * from Student
Where cid is null;
```

→ list name of student starts with letter 'A'

```
Select Sname from Student
Where Sname like 'A%';
```

→ list student whose name ends with 'a'

```
Select * from Student
Where Sname like '%a';
```

→ list id & name whose name has any occurance of 'a'

```
Select Sid, Sname from Student
Where sname like '%a%';
```

→ increase 10 marks in each Student & Rename the Column.

```
Select Sid, Sname, marks, marks+10
    from Student;

Select Sid, marks, marks+10 as grading
    from Student;
```

→ update marks as 65 of Sid 3

```
~~Select~~ update Student
    Set marks = 65
    where Sid = 3;

Select * from Student;
Commit;
```

→ Add Col. of Contact in Student

```
Alter table Student
Add Contact number (10);

desc Student;
```

→ Delete Contact from Table

```
Alter table Student
    Drop Column Contact;
```

25/1/23

Unit-4

Advance SQL

→ SQL Join operator:

- The relational Join operation merge rows from two tables & returns the rows with one of the following condition:

- Have common values in common column. (natural Join)
- meet a given Join condition equality
- Have common values in common cols or have no matching values. (outer Join)

- Join operation can be classified as inner Join or outter Join

- The inner Join is a traditional Join in which only the row that meet a given condition is selected.

- It can be natural Join, equi-Join or Theta Join

- The outter Join ret. not only the matching rows but also not matching attribute values.

- It can be left outer Join, right-outer Join, full outer Join.

→ Types of Join:

(i) **Cross Join:** A cross Join performs the relational product, Cartacian product of Two Tables.

Syntax:

```
Select col. list from table1, table2;
                or
SELECT col.list FROM table1
CROSS JOIN table2;
```

eg. Select * from Customer, Agent;

Cid	Cname	Aid
1	A	01
2	B	01
3	C	02
4	D	02

Aid	Aname
01	AA
02	BB
03	CC

Cid	Cname	Cust. Aid	Ag. A-id	Aname
1	A	01	01	AA
1	A	01	02	BB
1	A	01	03	CC
2	B	01	01	AA
2	B	01	02	BB
2	B	01	03	CC
3	C	02	01	AA
3	C	02	02	BB
3	C	02	03	CC
4	D	02	01	AA
4	D	02	02	BB

(ii) natural Join: A natural Join gets all rows with matching values in the matching cols & eliminates duplicate cols.

~~Syntax~~: The natural Join performs following steps:

1. The product operator is used to display Cartacian product of 2 table
2. The SELECT operator is performed on the result of Step one that displays only the rows with common values in the common attributes.
3. The project operator is performed that display a single copy of each attribute.

Syntax:

SELECT Columnlist FROM Table1 NATURAL JOIN Table2;

or

SELECT Col.list FROM table1, table2 WHERE table1.Col.name = table2.Col.name;

SELECT * from customer natural Join agent;

or

SELECT * from customa, Agent when Customer.Aid = Agent.aid;

Aid	Cname	Aid	Aname
1	A	01	AA
2	B	01	AA
3	C	02	BB
4	D	02	BB

(iii) Join USING clause:

- The second way to express a Join is through the Using keyword.
- The Query returns on the rows with matching values in the column indicated in USING clause. with the common column.

Syntax: SELECT col.list FROM Table1 JOIN Table2 (Com.col.);
using.

SELECT * from Customer join Agent (Aid);
using

(iv) Join ON clause:

- Another way to express a Join when the tables have no common attribute name is to use the Join ON Keyword.
- The Query will return only the rows that meet the specified Join Condition.

SELECT Col. list FROM table1 JOIN table2
ON table1 = table2. col.
. col.

SELECT * from Customer ~~from~~ JOIN Agent
ON Custome. Aid = agent. Aid.

→ Display Cid, Cname & its related agent name

SELECT Cid, Cname, ~~Aid~~ Aname FROM
Customer, Agent
WHERE Customer. Aid = Agent. Aid;

v) Left outer Join:

- The left outer Join display all the records from the first table including thoughs who do not have matching value in the other table

Syntax:

Select col. list from Table1 left outer Join table2 on table1. Col. = table2. col;

Eg: SELECT * from Customa left outer Join Agent on Customer. Aid = Agent. Aid;

(vi) Right outer Join:

- it displays all the rows of the second table (right) including thoughs who do not have matching value in the first table.

- Select col. list from table1 RIGHT OUTER JOIN table2 ON table1. col. = table2. col;

Select * from customer ~~def~~ Right outer Join Agent ~~table~~ On customer. Aid = Agent. Aid;

Cid	Cname	Aid	Aname
1	A	01	AA
3	C	02	BB
		03	CC
2	B	01	AA
4	D	02	BB.

(vii) Full outer Join:

- it is combination of left & right outer Join.

- it displays the matching & not matching values from both the tables.

Select col. list from table1. Full outer Join table2 on table1.col = table1.col;

Select * from Customer full outer Join Agent on Customer.Aid = Agent.Aid;

→ Relational Set operators:

SQL provides following relational set operators:

- UNION
- UNION ALL
- INTERSECT
- MINUS

1. UNION: - The UNION operator combines rows from two or more tables excluding duplicate rows.

   - To implement UNION op., Both tables must be UNION compatible (must have same cols.)

Syntax: SELECT col. list from Table1.
UNION
SELECT col. list from table2;

Or

CREATE UNION View

Select * from product Union product1;

Product1

Pid	Pname
1	A
2	B
3	C

Product 2

Pid	Pname
3	C
4	D
5	E

Pid	Pname
1	A
2	B
3	C
4	D
5	E

2. UNION ALL: The UNION ALL operator combines rows from Two or more tables including duplicate rows.

To implement UNION ALL, Both tables must UNION compitible.

Eg: Select * from product UNION ALL Product1;

Pid	Pname
1	A
2	B.

→ DATE & TIME functions:

(1.) To_char: It returns a character string or a formated string from a date value.

TO_CHAR (date value, fmt).

WHERE, fmt can be:
- MONTH (name)
- MON (three letter of month name)
- MM (2 digit of month name)
- D (No. of day of week)
- DD (No. of day of month)
- DAY (Name of day of week)
- YY (2 digit year value)
- YYYY (4 digit year value)

→ display Sid, Sname, dob & Birth year of student

Select Sid, Sname, dob, to_char(dob, 'YYYY') from Student;

→ display Students who a Born in Nov. month.

```
Select sid, sname, dob, to_char (dob, 'YYYY')
    from student
where to_char (dob, 'MM') = 11;
```

(2) TO-DATE: It returns a date value using a character string.

TO_DATE (datevalue, FMT)

WHERE, fmt can be:

- MONTH
- MON
- MM
- D
- DD
- DAY
- YY
- YYYY

→ display DOB in MM/DD/YY

```
Select sid, sname, dob, to_char (dob, 'MM/DD/
                                      YY')
                    from students;
```

- NOTE: '11/25/2012' is a Text string, not a date (True)

(3.) SYSDATE: It returns Systems todays date.

Syntax: SYSDATE.

eg. Select SYSDATE from dual;

(4.) ADD_MONTHS: it adds a no. of month or year to the date.

ADD_MONTHS(date value, n)

WHERE n = no. of month.

eg. Add 2 years in MFG date & ~~set~~ rename the col. as exp. date.

Select Mfgdt, add_months(mfgdt, 24) as expdt from prd.;

(5) LAST_DAY: it ret. the day of the last day of a month given in a date.

LAST_DAY (date value);

eg. Select dob, LAST_DAY(dob) from student;

→ Numeric functions:

SQL provides following numeric functions:

- ABS
- ROUND
- CEIL
- FLOOR.