UNIT-4 @python_world-in

Structural Query language

Introduction to SQL:-

In SQL we didn't use Back space

* Structured query language (SQL) is a database query language used for storing & managing data in relational DBMS (RDBMS)
* SQL was the first commercial language Introduced for E.F Codd's relational model of database
* Today almost all RDBMS [MySQL) Oracle, Informix Sybase, mx-axis] use the SQL as the standard database query language. SQL is used to perform all types of data operations in RDBMS
* In simple way query is a mission question
* A query is formulated for a relation (or) table to retrie some useful information from the table

Applications of SQL:-

* SQL is one of the most widely used query language over the databases
  1) Allows @users to access data in the relational database management
  2) Allows users to describe the data
  3) Allows users to define the data in a database & manaoulate the data

4) -Allows to Embade with in other languages using SQL modules, libraries & pre compilers

5) Allows users to create & drop databases & Tables

6) -Allows users to create views, Stored processes. functions in a database

7) -Allows users to set permissions on tables procedures, & views

-History of SQL:-

* In 1970 Doctor of) Dr. Edger. F. Teed Codd of IBM (International Bussiness Missions Corporation) is known as the father of Relational database. He described a Relational model for databases

* In 1974 - structured Query language appeared

* In 1978 - IBM worked to develop Codd's Ideas & released a product named system/R.

* In 1986 - IBM developed the first prototype of Relational database & standardised by (ANSI) -American National Standard Institute the -first Relational database was released by Relational software which later came to be known as oracle

Commands in SQL :-

* SQL commands are instructions, coded into SQL statements, which are used to communicate with the database to perform specific task words functions & query's with data
* SQL commands can be used not only for searching the database but also to perform various other functions
* Example you can create a table, add to tables, modify the adjucting the data, drop the database are table set permissions for users
* SQL commands are grouped into "5" major categories depending on their functionality
* Data definition language (DDL)
* Data manipulate ~~manipulate~~ language (DML)
* Data Control language (DCL)
* Transcation control language (TCL)
* Data Querry language (DQL)

@python_world_in

Data Definition language (DDL)

* Data definition language (DDL) is a language that allows the user to define the data & their relationships to other types of data
* Data Definition language statements work with the structure of the database table

  Various data types used in defining columns in a database table

  Integrity & value constraints

  Viewing, modifing & Removing a table structure
* The DDL commands are create, Alter, Drop, Rename & truncate
* All DDL commands are autocommited that means it saves all the changes permanently in the database

DDL Commands

* The Data Definition language (DDL) Commands are * CREATE - It is used to create a New table are a new database
* ALTER - It is used to alter are change the structure of the database table
* DROP - It is used to delete a table, Index are views from the database
* RENAME - It is used to Rename an object from the database
* TRUNCATE - It is used to delete the records are data from the table, but it's structure

remains as it as.

CREATE COMMAND:-

Create is a DDL and SQL command used to create a database are a table in a Relational DataBase Management System (RDBMS)

CREATE A DATABASE:-

To create a database in RDBMS, then we are using the create command

syntax:- CREATE database databasename;

Eg:- CREATE database testDB;

CREATE A TABLE

* Create command can also be used to create tables
* Now when we create a table, we have to specify the details of the colums of the table too
* We can specify the names and the datatypes of various colums in the command itself.

Syntax:- CREATE table table_name (or) [tablename]

we will Express this like also

(column1 datatype,)
Column2 datatype,
Column3 datatype,
- - - - - - - - - -
ColumnN datatype,
PRIMARY KEY(column)
); → It is single statement because ends with one semicolon

This all steps [∴ Diff of SQL – Single state, PL/SQL – Multiple state]

Eg 1:- CREATE table student [we will use bracket from after name also (or) down line also it will support]

(
studid int,
name varchar(10),
age int
);

Eg 2:- CREATE table customers

(
ID int not null,
NAME varchar(20) not null, [we will write fields small (or) Big letters but write only one way]
AGE int not null,
ADDRESS char (25),
SALARY Decimal (8,2),
PRIMARY KEY (ID)
);

[not null didn't accept null but accept duplicate
unique accept nulls didn't accept duplicate]

[The combo of not null & unique is primary key]

* ALTER Command:-

ALTER command is used for Altering the table structure such as

1. To Add a column to Existing table
2. To Remove any Existing column
3. To change data type of any column or to modify it's size
4. To drop a column from the table.

@python_world_in

Add a New column:-

Using ALTER command we can add a column to any Existing table.

Syntax:- ALTER table tablename
add (column- name datatype);

Eg:- ALTER table student
add (address varchar(20));

-Add multiple New columns:-

Using alter command we can add multiple new Columns to any Existing table

Syntax:- ALTER table tablename
add (column$_1$ datatype,
column$_2$ datatype,
column$_3$ datatype,
);

Eg:- ALTER table student
add (father-name varchar(20),
mother-name varchar(20),
DOB date
);

-Add column with default value:-

ALTER command can add a new column to an Existing table with a default value too The default value is used when no value is inserted in the column

Syntax:- ALTER table tablename
add (column datatype default name-value);

Eg:- ALTER table student
add (dob data default "01-Jan-99");

Modify an Existing column:-

ALTER command can also be used to modify the datatype of any Existing column

Syntax:- ALTER table tablename
modify (column-name datatype);

Eg:- ALTER table student
modify (address varchar(200));

Rename a column:-

Using ALTER command you can rename an Existing column

Syntax:- ALTER table tablename
rename {old-column-name to
new-column-name;

Eg:- ALTER table student;
rename address to location;

* Drop a column:-

ALTER command can also be used DROP of Remove columns

Syntax:-
ALTER table tablename
Drop (column-name);

Eg:- ALTER table student
Drop (address);

* Drop Command:-

Drop Command Completely removes a table from the database This command will also destroy the table structure & data stored in it

Syntax:- Drop table tablename;

Eg:- Drop table student;

Rename Command:-

Rename Command is used to set a new for any Existing table.

Syntax:- RENAME table old-table_name to new_table_name;

Eg:- RENAME table student to student_info;

TRUNCATE command:-

TRUNCATE command Removes all the records from a table but this command will not destroy the table structure

Syntax:- TRUNCATE table tablename

Eg:- Truncate table student;

Data Manuculation language (DML) :-

* DML commands ale used for manculating the data stored in the table and not that table it self
* DML commands are not auto commited it means changes are not peminentant to database, they can be rolled back

DML Commands :-

* INSERT - It is used to incert data into used a table
* UPDATE - It is used to update Existing data with in a table
* DELETE - It is used to delete records from a database table
* SELECT :- It is used to shows the Records of the specified table

Example 1 :- let's take a student table consisting of the following Records

Student Table		
Sid	S Name	Age
101	Alex	18
102	Adam	19
103	Abhi	20

Insert Command :-

The Insert query command SQL provides a way to add new rows of information (or) data inside a specific database of the RDBMS insert can be Executed using to Esigtaxs

Syntax :-

```
INSERT into tablename
Values ( data1, data2, data3, ....);
```

Eg :- INSERT to student
~~Values (101, 'Rani', 90);~~ (@pythonworld-in)
Values (101, 'Rani', 90);

INSERT values into only specified coloums :-

* We can use the insert command values for only some specific colums of a row
* We can specify the colum names along with the values will be inserted

Eg :- INSERT into Student (SNO, SName, age)
values (101, 'Rani', 90);

INSERT NULL VALUES TO A COLUMN :-

Ex :- INSERT into student (SNO, SName)
Values (101, 'Rani');

(OR)

INSERT into student
Values (102, 'Alex', null);

INSERT Multiple rows at a time :-

Ex:- INSERT into student

values (&SNO, '&sname', &age);

Enter SNO: 101.

Enter Sname: Alex

Enter age : 20

SQL>/

UPDATE

It is used update any record of data in a table

Syntax:- UPDATE tablename set

Column_name = namevalue where condition;

Ex:- UPDATE student set age = 20

where sid = 102;

Update multiple colums

we can also update values of multiple colums using a single update statement.

Ex:- UPDATE tablename set

Sname = 'Rani', age = 30 where sid = 103;

Incrementing Integer values in UPDATE :-

Eg:- UPDATE student set age = age + 1;

DELETE COMMAND :-

It is used to delete data from a table

Syntax :- DELETE from table_name ;

→ DELETE All records in a table :-

Ex :- DELETE * from student

The above command will delete all the records from the table student

→ DELETE a particular record from a table

In our student table if you use want to delete a single record we can use the where clause to provide a condition in our delete statement

Ex :- DELETE from student where sid = 101 ;

(or)

DELETE from student where sname = 'abhi' ;

* SQL Data Types :-

Data types defines what types of data a column can contain

* character [(length)] (or) char [(length)]
* Varchar [length
* Boolean
* Small int
* Integer (or) int
* Decimal [(P[,S])] (or) Dec [(P[,S])]
* Numeric [(P [,d])]
* Real
* float (p)

[@python_world-in]

* Double precision
* Date
* Time
* Time_stamp
* CLOB[(length)] (or) character large object [length char large] [(length)]
* BLOB [(length)] (or) Binary large object [(length)]
* character [(length)] (or) char [(length)]
* The character datatype accepts character strings, including unicode of a fixed length
* The length of the character string should be specified in the data type declaration

Eg:- character(n)

where n represents the the desired length of the character string

* If no length is specified during the declaration, the default length is 1.
* The minimum length of the character datatype is 1 & it can have a maximum length up to the table page size
* Character strings that are larger then the page size of the table can be stored as a character large object (CLOB).

Note :- Character (0) is not allowed & raised an Exception

Ex :- Char (10) / character (10)

- Valid
  'Race car'
  'Race car'
  '84865'
  '1998-10-05'
- Invalid
  28465
  1998-10-05

[@python_world_in]

* Varchar (length) :-
* The Varchar datatype accepts character string including unicode of a variable length up to the maximum length specified in the data type declaration
* Eg :- Varchar (n);
* It can accept any length of character string upto n Character in length

Note :-
Varchar (0) is not allowed & raised an Exception

* If you need to store character strings that are longer than the current table page size, the character large object (CLOB) datatype should be used

Eg:- Varchar(10)

'Race Car'

'RACECAR'

'94865'

⇒ Boolean:-

* It supports the storage of two values: TRUE or -false. No parameters are required when declaring a boolean data type

Eg:- BOOLEAN

TRUE

TRUE

TRUE

FALSE

⇒ Small int:-

* This datatype accepts numeric values with an important scale of zero

* If you assign a numeric value with a precision a scale to a smallint data-type, the scale portion town without rounding

Eg:- Small int

-32768

-0

-30.3(right of the decimal point truncated)

Integer or int:-

The integer data type accepts numeric values with a implied scale of zero

Eg:- -214783.648
-1025
0
1025.98 (right of the decimal point truncated)

Decimal [(P,S)] (OR) Dec [(P[,S])]:-

* The decimal datatype accepts numeric values for which you may define a precision & a scale in the data declaration

* DECIMAL data types can be declared in one of 3 different ways.

1. DECIMAL - precision defaults to 38, scale default
2. DECIMAL(P) scale defaults to 0
3. DECIMAL (P,S) - precision & scale are defined by the user

Eg:- DECIMAL (10,3)

Valued :- 1234567

12345667.123

1234567.1234 (final digit is truncated)

Numeric [(P[,S])]:-

* point base treats the NUMERIC data type in Exactly the same way as the DECIMAL data type.

REAL:-

* It accepts approximate numeric values upto a precision of 64
* No parameters are required when declaring of REAL data type

Eg:- REAL

Valid :- 2345
0
1.045

FLOAT(P):-

* This type accepts approximate numeric values for which you may define a precision up to a maximum of 64
* If no precision is specified during the declaration the default precision is 64

Eg:- FLOAT(8)

Valid :- 12345678
1.2
123.45678

DOUBLE PRECISION:-

* The REAL data type accepts approximate numeric values, upto a precision of 64
* No parameters are required when declaring a DOUBLE PRECISION data type
* If you attempt to assign a value with a precision greater than 64 an Error is raised

Eg:- DOUBLE PRECISION

Valid :- 123456789- - - -

Invalid :- 123,456,793- - -

DATE :-

* The DATE datatype accepts data value. No parameters are required when declaring a DATE data type

* Date values should be specified in the form:
YYYY-MM-DD

* Values assigned to the DATE data type should be enclosed in single quotes, preceded by the case insensitive keyword DATE

Eg:- DATE

Valid :- '1999-01-01'

DATE :- '2000-2-2'

Invalid :-

DATE '1999-13-1'

TIME :-

* The TIME data accepts time values No parameters are required when declaring a TIME data type.

* Time values should be specified in the form : HH:MM:SS

* The minutes & seconds values must be two digits hour values should be between 0 & 23 minutes values should be between 00 & 59 61.999999.

* values assigned to the TIME Data type should be Enclosed in single quotes, proceded by the case intensitive keyword TIME

TIMESTAMP:-

* This data type accepts timestamp values which are a combination of DATE value & a TIME ~~star~~ value
* No parameters are required when declaring a TIMESTAMP data types
* TIMESTAMP values should be specified in the form: YYYY-MM-DD HH:MM:SS
* There is a space operator b/w the data & time portions of the timestamp

Eg:- TIMESTAMP "1999-04-04 07:30:00"

CLOB[(length)]/character large object[(length)]
char large object[(length)]:-

* This datatype accepts character strings larger than those that are allowed in the character[(length)] (or) VARCHAR(length) data types.

Syntax :- n[K/M/G]

Where n is an unsigned integer that represents the length, K, M, G correspond the Kilobytes, (or) Megabytes (or) Gigabytes respectively.

* If K, M (or) G is specified in addition to n, then the actual length of n is

K = n * 1000

M = n * 1,048,576

G = n * 1,073,741,824

* The maximum size allowed for CLOB data types is 9 Gigabytes
* If a length is not specified, then a default length of one byte is used

Note :- The CLOB data type supports unicode data

BLOB [(length)] / Binary large object [(length)] :-

* This datatype accepts binary values :-

Syntax :- n[K/M/G]

Where n is an unsigned integer that represents the length K, M, G correspond the Kilobytes, Megabytes (or) Gigabytes respectively

* If K, M & G is specified in addition to n then the actual length of n is :-

K = n * 1024

M = n * 1,048,576

G = n * 1,073,741,824

* The maximum size allowed for BLOB data type is 9 gigabytes
* If a length is not specified then a default length of one byte is used

Note :- BLOB data types cannot be used with SQL

Status functions

Aggregate functions :-

Short Intro: The ISO standard defines "5" aggregate functions

* Count
* Sum
* Average (AVG)
* min
* max

(@ Python-world-in)

Use of Aggregate functions

* from a business perspective, different Organization levels have different information requirements top level managers are usually interested in knowing whole figures & not necessary the individual details
* Aggregate functions allows us to usually produce summerized data from our data base

Ex :- Movierentals table

Reference number	transaction data	return data	membership number	movie-id
11	20-06-2021	NULL	1	1
12	20-06-2021	25-06-21	1	2
13	20-06-2021	25-06-2021	3	2
14	21-06-2021	24-06-2021	2	2
15	23-06-2021	NULL	1	3

Count() function

* It returns the total number of values in specified field
* It works on both numeric & non-numeric datatypes
* All Aggratic functions by default Exclude null values before working on the data
* Count (*) is a special implimentation of the count function that returns the count of all the rows in a specified table count(*) also considers nulls & duplicates

Ex:-

```
select COUNT(movie_id)
from movierentals
where movie_id = 2;
```

Output:-

[@python-world-in]

COUNT(movie_id)
3

DISTINCT KEYWORD:-

The DISTINCT keyword that allows us to vomit duplicates from our results

```
Select movie_id from movierentals
```

Output:-

movie_id
1
2
2
3

Now Execute the same query with the distinct keyword

Ex:-

```
Select DISTINCT movie_id from movierentals;
```

Output:-

movie_id
1
9
3

MIN() function:-

It returns the smallest value in the specified table-field

Ex:-

```
Select MIN(reference_number)
FROM movierentals;
```

Output:-

MIN(reference_number)
11

MAX()-function:-

It returns the largest value in the specified table field. It is the opposite of the MIN function

Ex:-

```
Select MAX(reference_number)
from movierentals;
```

O/P:-

MAX(reference_number)

SUM() function :-

* It returns the sum of all the values specified column sum works on numeric fields only
* Null values are Excluded from the result returned

Eg :-

```
Select SUM(reference_number)
FROM movierentals;
```

O/p :- SUM(reference_number)

65

AVG
~~Avg~~() function :- [@python_world_in]

* It returns the ~~Avg~~ AVG of the values in a specified coloum
* Just like the sum function it works only on numeric data types

Syntax :-

```
Select AVG(reference_number)
FROM movierentals;
```

O/p :- AVG(reference_number)

13

S.D
M.D

* Selection & projection operations :-

Relational Algebra operations

- Basic operations
  - Unary operations
    - project operation (π)
    - selection operation (σ)
    - Rename operation (ρ)
  - Binary operations
    - union operation (∪)
    - cross product operation (×)
    - minus/set difference (−)
- Extended / Derived operations
  - join operation
  - Division operation (÷)
  - intersection operation (∩)

* Selection operation (σ):-

Selection operation is used to retrive data for row (or) Tuple

Denoted by (σ) (sigma)

Syntax:-

$$\sigma_{\text{<selection condition>}}(R)$$

R = Relation / table

Eg:- $\sigma_{eid = 1}(Emp)$

Relation: Emp

eid	ename
1	A
2	B
3	C

Output:-

eid	ename
1	A

selection condition can use compaisions using Eq (=, ≠, <, ≤) = , != , <, <=, ≥ > =

It allows compaisions b/w two op attributes

Selection operation is a unary operator

Connectivities are used AND (∧), OR (∨) & NOT (¬)

project operator (Π):-

It selects certain columns from the table & discards the other colums / attribute

Syntax:-

$$\Pi_{\text{<attribute list>}}(R)$$

Eg:-

$$\Pi_{\text{lname, fname, salary}}(\text{Employee});$$

Relation : Employee

ENo	EName	Fname	Lname	Salary
1	Sri sree	Sri	Sree	10,000
2	Kavya Sri	Kavya	Sri	20,000
3	Sid Sriram	Sid	Sriram	40,000
4	Sai vijay	Sai	Vijay	1,00000

O/p :-

Lname	Fname	Salary
Sree	Sri	10,000
Sri	Kavya	20,000
Sriram	Sid	40,000
Vijay	Sai	1,00000

5m

* Imposition of constraints / SQL constraints :-

SQL constraints are rules / conditions used to limit the type of data that can go into a table to maintain the accuracy & integrety of the data inside the table.

SQL constraints can be divided into two types :-

1. Column level constraints - limits only column data
2. Table constraints - limits whole table data
3. Some of the most used constraints that can be applied to the table are :-
   - NOT NULL
   - UNIQUE
   - PRIMARY KEY
   - FOREIGN KEY
   - CHECK
   - DEFAULT

4. Constraints are used make store that the integrity of data is maunted in the database

NOT NULL CONSTRAINS:-

It restricts a column from having a null value

Once NOT NULL Constraints is applied to a Column you can't pass a null value to the Column. It inforces a column to Certain a proper value

Note :- It can be define at a table level.

Syntax:-

```
CREATE table tablename
( column1 datatype Constraint,
  column2 datatype Constraint,
  column 3 datatype Constraint;
);
```

Eg:-

```
CREATE table student
( sid int NOTNULL
  sname varchar (20) NOT NULL,
  age int
);
```

In the Example sid & sname fields of student table will not take 'null values

UNIQUE Constraints:-

* It Ensures that a field/column will only have unique value
* A unique Constraint field will not duplicate data

Note:- This Constraints can be applied at level of table level

Syntax:- CREATE table tablename
(column1 datatype constraint, → column constraint
column2 datatype constraint,
column3 datatype constraint,
CONSTRAINT constraint_name constraint_type → table constraint
((column3));

Eg:- CREATE table person
(ID INT NOT NULL, → column constraint
lastname varchar(20) NOT NULL,
firstname Varchar(10),
age int,
CONSTRAINT Uni_person UNIQUE (ID, lastname)); → Table Constraint

3) PRIMARY KEY: Constraint:-
The primary key constraint uniquelly identifies Each record

* primary key must contain Unique value & cannot contain Null values

A table can have only one primary key
A primary key can consist of single or multiple columns (fields)

Column level / table level Constraints:-

* Column level constraints is declared at the time of creating a table but table level constraints is atter table is created.

* NOT NULL constraint can not be created at table used.
* Composite primary key must be declared at table level.

Syntax :- for column level

```
CREATE table tablename
(column1 datatype constraint,
 column2 datatype constraint,
 column3 datatype constraint,
 columnN datatype constraint.
);
```

Syntax :- for table level [@python-world-in]

```
CREATE table tablename
(column1 datatype,
 column2 datatype,
 columnN datatype,
Constraint Constraint_name constraint_type(
                column1, column2, .... ));
```

Eg :-

```
CREATE table person
(ID int NOT NULL,
 lastname Varchar(20) NOT NULL,
 firstname Varchar(20),
 age int,
CONSTR
CONSTANT PK_PERSON PRIMARY KEY (ID last
                                name));
```

Foreign key Constraint:-

* A foreign key is a key used to link two tables together

* A foreign key is a field or collection of fields in one table that refers to the primary key in another table

* The table containing the foreign key is called the child table, and the table containing the primary key is called the referenced (or) parent table

person (parent table)

person ID	last name	first name	age
1	Demon	Vampire	30
2	Lim	Tove	28
3	Hansh	desh	90

orders (child table)

Order ID	order number	person id
1	77895	3
2	44682	3
3	22456	2
4	24562	1

Ex: parent table

```
CREATE table person
( Personal int primary key,
 lastname Varchar(90),
 firstname Varchar(15),
 Age int
 );
```

[@python-world-in]

Ex: Child table

```
CREATE table odels
(OrderID int primary key
Ordername INT NOT null,
personal int,
CONSTRANT FK_odels FOREIGN KEY (personId)
REFERENCES personid(personID)
);
```

Check Constraint :-

* It is used to limit the value range that can be placed in a column
* If you define check Constraint on a int Single Column it allow only certain values for this column
* If you define a check constraint on a table it can limit the values in certain columns based on values in other columns in the row

Ex:- CREATE table person

(ID int NOT NULL,
lastname Varchar(20) NOTNULL,
firstname Varchar(15),
age int CHECK CHECK (age >=18));

CHECK constraint on multiple columns:-

Ex:- CREATE table person

(ID int NOT NULL,
lastname varchar(20) NOTNULL,
firstname Varchar(15),
age int,
City Varchar(20),
CONSTRAINT chk_person CHECK(age>=18 AND City=
"Hyderabad"));

[@python_world-in]

DEFAULT CONSTRAINT:-

* It is used to provide a default value for a column
* The default value will be added to all new records if know other value is specified

Ex:- CREATE table person

(ID int NOT NULL,
lastname Varchar(20) NOTNULL,
firstname Varchar(15),
age int,
City Varchar(20) DEFAULT 'HYD'
);

The Default Constraint can also be used to insert system values, by using functions:

like get date GETDATE()

Ex:- CREATE table odels

(ID int NOT NULL,
Ordernumber int NOT NULL,
Order_date date DEFAULT GETDATE());

SET operations in SQL

SQL supports two set operation which can be performed on table data this are used to get meaningfull results from data stored in the table, under different special conditions

* Set operations are

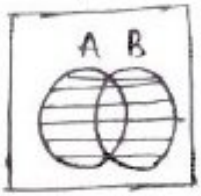
→ UNION
→ UNION ALL
→ INTERSECT
→ MINUS

(@python_world_in)

UNION operation:-

* Union is used to combine the results of two (or) more select statements
* However it will Eliminate duplicate rows from its result set

* In case of UNION, number of colums & data types must be same in the both tables, on which Union operation is being applied



Ex:- first table

ID	Name
1	Abhi
2	Adam

second table

ID	Name
1	Adam
3	Ravi

Union SQL query will be; SELECT * from first UNION

SELECT * from second;

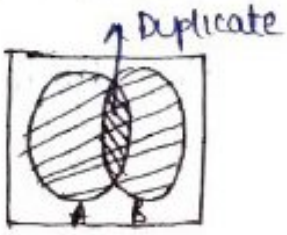
output:-

ID	Name
1	Abhi
2	Adam
3	Ravi

[@python_world_in]

UNION ALL:-

* This operation is similar to UNION but it also shows the duplicate rows



* Union ALL query will be

Ex:- SELECT * from first
UNION ALL
SELECT * from second;

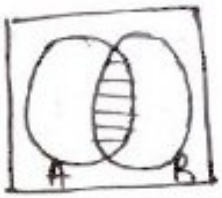
O/P:-

ID	Name
1	Abhi
2	Adam
3	Ravi
1	Adam
3	Ravi

INTERSECT

* It is used to combine two select statements, but it only returns the records which are common from both select statements

* In case of intersect the number of colums & data type must be same



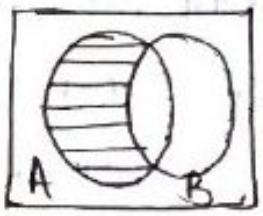
Intersect query will be

Ex:- SELECT * from first
INTERSECT
SELECT * from second;

Output:- ID Name
2 Adam

MINUS:-

* The MINUS operation combines results of two select statements & return only those in the final result, which belongs to the first set of the result



Ex:- SELECT * from first
MINUS
SELECT * from second;

Output:- ID Name
1 Abhi

VIEWS :-

Table (or) Relation

- A Table is a fundamental structure in oracle
It is a database object that contains rows of data you can perform DML operations like INSERT UPDATE and DELETE on table.

VIEW :-

* A view is simply any select query that has been given a name and saved in the database for this reason, a view is sometimes called a named query (or) a stored query
* views are read only views doesnot allow you to modify data
* views are useful when we have complex join queries. & we are using this queries at multiple places. In such cases we can give name to such Complex queries that is we can create view (or) Named query.
* views are also useful when we want to hide certain columns from users which we cannot do using tables by creating a view, we can also achieves security
* View is a virtual table whose contents by a query
* A view doesn't exist as a stored set of data values in a database.

Table :-

* A table is preliminary storage for storing data & information in relation Data Base Management System (RDBMS)
* A table is a collection of related data entries & it consists of rows & columns
* Table is a physical representation of data & view is a biological representation of data

Creating View statement :-

* In SQL, A view is a virtual table based on the result set of an SQL statement.
* A view contains rows & columns just like a real table
* The fields in a view are fields from 1 or more real tables in the database
* We can add SQL functions, where (WHERE & JOIN) statements to a view

Create view syntax :-

```
CREATE VIEW viewname AS
SELECT column1, column2, ...
FROM table_name
WHERE conditions;
```

Note :- A view always shows upto_date data

Ex :-

```
CREATE VIEW Vi_customers AS
SELECT Cus_name, contact_name
FROM customers
WHERE country = 'INDIA';
```

* We can see the view statement by using SELECT statement

Ex: SELECT * from vi_customers

* A view that selects every product in the products table with a price higher than the Average price

```
CREATE VIEW vi_products AS
SELECT productname, price
FROM products
WHERE price > (SELECT Avg (price) FROM products);
```

* SELECT * from vi_products

UPDATING A VIEW:-

A view can be updated with the CREATE (OR) REPLACE VIEW command

Syntax:-

```
CREATE OR REPLACE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Ex:- VIEW

```
CREATE OR REPLACE VIEW vi_customers AS
SELECT ws_name, contactname, city
FROM customers
WHERE country = 'INDIA';
```

(@ python world-in)

DROPING A VIEW

* A view is deleted with the DROP VIEW command

Syntax:-

```
DROP VIEW view_name;
```

Ex:-

```
DROP View vi_customers;
```

Table	View
1. A table is used to organise data in the form of rows & columns & displayed them in a structure format. It makes the stored information more understable to the human	1. Views are treated as virtual or logical table used to view or manipulate parts of the table. It is a database object contains rows & columns the same as real table.
2. Table is a physical entity that means data is actually stored in the table.	2. The view is a virtual Entity which means data is not actually stored in the table
3. It is used to store the data	3. It is used to Extract data form
4. It generates a fast result	4. The view generates a slow result because it renders the information from the table every time we quer
5. It is an independent data object	5. It depends on the table therefore, we cannot create a view without using tables

6. Table allows us to perform DML operations	6. The view will unable, us to perform DML operations
7. It is not an Easy task to replace the table directly because of its physical storage	7. It is an Easy task to replace the views & recreate it when Ever needs.
8. It occupies space on the system	8. It doesnot occupied space on the system