⇒ We will mainly work with PostgreSQL, also called Postgres.

POSTGRESQL :-

⇒ Postgres is an Open Source relational database management system.

⇒ By default, every Postgres installation comes with one database already created, called as "postgres"

⇒ This is important as Postgres requires to specify the name of a database to make a connection. So, there needs to be always one database.

Postgres DataTypes

- Databases have datatypes just like any programming language

Data Type	Postgres	Python
Numeric	Int, decimal, precision	Int, float
Text	Varchar, text	string
bool	boolean	boolean
sequence	array	list

DATA TYPES :-

Postgres offers a lot variety of different data types. Some of them are character varying, integer, serial.

It mainly supports 3 types of constraints :- NOT NULL, UNIQUE & DEFAULT

For auto increment field, (i.e such a field for which if value is not provided, it will generally produce a value and place it. It is mainly of integer data type), Postgres do not have any keyword.

To make a field having auto-increment feature, we need to make the datatype as serial, smallserial or bigserial.
For any new entry, value of this field will be the previous entry value + 1.

Using Timestamp as Column :-

Suppose, we want to log (in a column) the time when each entry is created. This info is useful.

For such things, Postgres offers certain datatypes :- timestamp with time zone & timestamp without time zone

⇒ Now, we cannot fill data for this field manually each time. It should automatically set the current time while creating entry.

For this to happen, we can set default value to such column, to NOW() function.
This function fetches current system time, when executed.

⇒ Executing SQL commands :-

⇒ [Right click over database on which we want to execute SQL queries] → Query Tool option

A new tab will open, and in its Query subtab, we can write & execute our own SQL queries.

⇒ Query to rename column name :-

```
ALTER TABLE products RENAME id TO p_id;
```

id → old column name (existing)
p_id → new column name

⇒ In PostgreSQL (GUI), string values in SQL queries must be enclosed within single quotes only
[using double quotes gives error]

Ex:- Query for string pattern matching :-

```
SELECT * FROM products WHERE name LIKE '%Bag';
```

All the entries in products table with their names ending with Bag will be fetched.

It is case-sensitive

As we mentioned "Bag" in our string, so, names ending with bag will not be returned, as the case of 'B' varies.

LIMIT & OFFSET :-

- LIMIT keyword is used to limit the no. of records being fetched, according to filters. The integer passed after it, will be the no. of records being fetched.

- OFFSET keyword is used to skip a specific no. of records from the top of the order.

It takes an integer after it, and skips that many records from the top and returns/fetches the rest of the records.

Ex:- For example, suppose this is the demo table test :-

id	name
1	Apple
2	Ball
3	Cat
4	Dog
5	East
6	Fan

```
SELECT * FROM test OFFSET 3 ;
```

Setting 3 as offset skips the top 3 records and fetches the rest.

id	name
4	Dog
5	Fast
6	Fan

```
SELEC * FROM test LIMIT 2 ;
```

id	name
1	Apple
2	Ball

⇒ We can chain both these keywords, to fetch a particular continuous set of entries from the middle of the table, like this :-

```
SELECT * FROM test ORDER BY name ASC
  LIMIT 2  OFFSET 3;
```

id	name
4	Dog
5	Fast

Thus, it skips the first 3 entries, and from the next entry, fetches the next (2) records, as LIMIT is set to 2.

⇒ RETURNING keyword :-

⇒ While we are inserting data in a table, if we want that the data getting inserted should be returned simultaneously, then we use the RETURNING keyword.

- We use it with INSERT command, at the end. After the RETURNING keyword, we mention the column names whose data should be returned.

To return whole record data (all columns), we put * after RETURNING keyword.

Ex:
```
INSERT INTO products (name, price) VALUES ('Car', 10000) RETURNING id;
```
After executing the above statement, the id of the newly created record will be returned. In place of id, we can put *.

Ex:
```
INSERT INTO products (name, price) VALUES ('Car', 10000) RETURNING id, name, created_at;
```

- RETURNING keyword can also be used with commands like DELETE, UPDATE