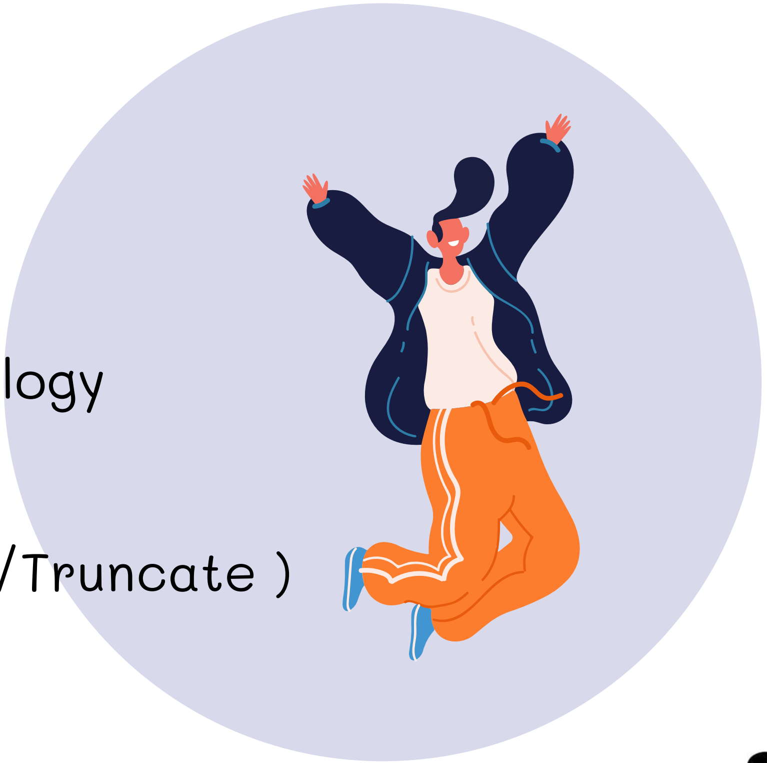
PostgreSQL
Day01

Prepared by:

Noha Shehab
Teaching Assistant
Information Technology Institute (ITI)

Table of contents

Table of contents



Why postgreSQL?

- Designed for high volume environments.
- Cross platform
- Low / No Cost.
- Stability
- Open Source

History of PostgreSQL

- PostgreSQL is derived from the POSTGRES package written at the University of California at Berkeley.
- IT is object-relational database management system.
- Created by Michael Stonebraker
- In 1994, Berkeley graduate students Andrew Yu and Jolly Chen replaced the PostQUEL query language interpreter with one for the SQL query language, creating Postgres95
- **PostgreSQL**, to reflect the relationship between the original POSTGRES and the more recent versions with SQL capability.

Relational database

- **RDBMS software:**
  - Enables you to implement a database with tables, columns and indexes.
  - Guarantees the Referential Integrity between rows of various tables.
  - Interprets an SQL query and combines information from various tables.
  - C/C++, Java Interface

Relational database

- RDBMS software:

ORACLE

Microsoft SQL Server

MySQL

PostgreSQL

ERD

- **Entity Relationship Diagram**
- **Attribute**
  - An attribute is a property or characteristic of an entity, relationship. For example, the attribute Inventory Item Name is an attribute of the entity Inventory Item.
- **Multivalued Attribute**
  - If an attribute can have more than one value, it is called a multivalued attribute.

ERD

- **Derived Attribute:**
  - An attribute based on another attribute. This is found rarely in ER diagrams. Such as calculations
- **Relationship:**
  - A relationship describes how entities interact.
  - Example: Student, address, track, staff, courses, desk.

Relational database concepts

- **Table** :A Collection of related data.
  - The table has a name; a number of columns and a number of rows, a table in a database looks like a simple spreadsheet.
- **Columns:**
  - Each column in the table has a unique name and contains different data.
  - Each column has an associated data type as an integer, strings or Timestamp and so on .
  - Columns are sometimes called fields or **attributes**.

Relational database concepts

- **Rows** a group of related data.
  - Because of the tabular format, each row has the same attributes.
  - Rows are also called records or tuples.
- **Values:**
  - Each row consists of a set of individual values that correspond to columns.
  - Each value must have the data type specified by its column.

Relational database concepts

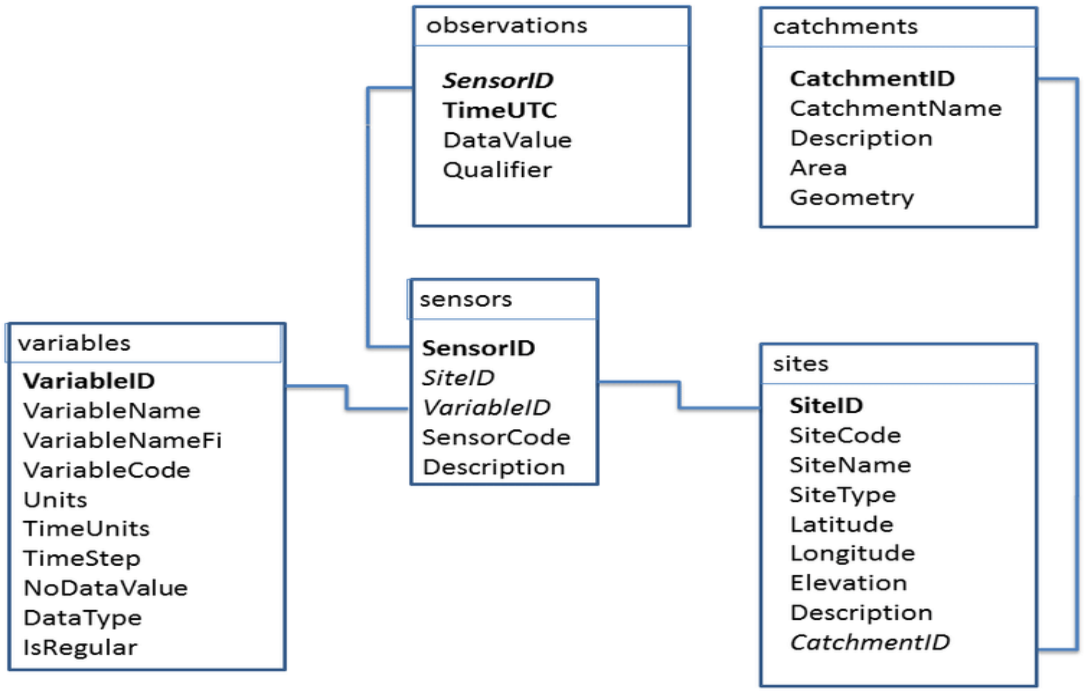
- **Primary key**
  - A primary key is unique.
  - A key value can not occur twice in one table. With a key, you can find at **most one row.**
- **Foreign key:**
  - A foreign key is the linking pin between two tables.
- **Referential Integrity:**
  - Referential Integrity makes sure that a foreign key value always points to an existing row

Relational database concepts

- **Index**
  - it is a data structure that improves the speed of data retrieval operations on a database table –
    - Disadvantages: Storage Size & Insertion Time
- **Schema**
  - It is akin to a blueprint for the database.
  - A schema should show the tables along with their columns, and the primary key of each table and any foreign keys.
  - A schema does not include any data.

Relational database concepts

- Schema



ORDBMS

- **ORDBMS:**
  - O for Object: The basic goal for the Object-relational database is to bridge the gap between relational databases and the object-oriented modeling used in programming languages such as Java, C+
  - R for Relational: It's called relational because all the data is stored into different tables and relations are established using primary keys or other keys known as foreign keys.

Designing Your Database

- **Simple tables**
  - That describe a real-world object. They might also contain keys to other simple objects with
  - which they have a one-to-one or one-to-many relation-ship.
- **Linking tables**
  - that describe a many-to-many relationship between two real objects.

Installing PostgreSQL

- For windows
    - https://www.enterprisedb.com/postgresql-tutorial-resources-training?cid=437

- For linux
    - https://www.digitalocean.com/community/tutorials/how-to-install-postgresql-on-ubuntu-20-04-quickstart

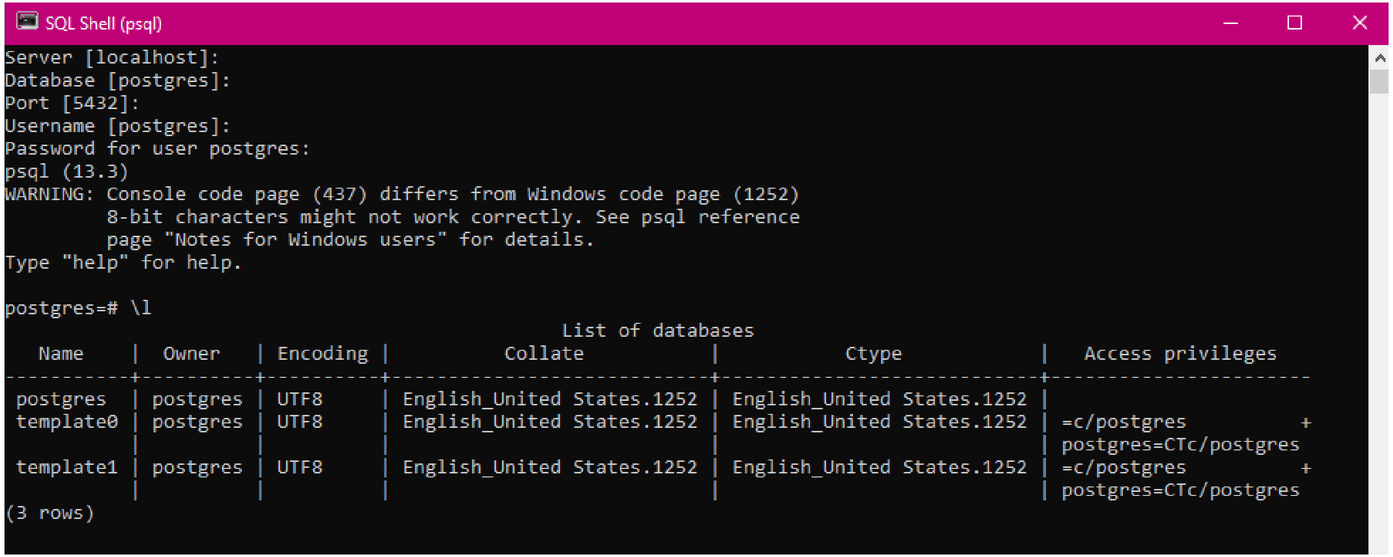
PostgreSQL files

- All the data needed for a database is stored within the data directory, commonly referred to as PGDATA .
- A common location for PGDATA is /var/lib/pgsql/data.

Item	Description
PG_VERSION	A file containing the major version number of PostgreSQL.
base Subdirectory	containing per-database subdirectories
global Subdirectory	containing cluster-wide tables, such as pg_database.
pg_xlog	Subdirectory containing Log files
postmaster.opts	A file recording the command-line options the server was last started with
postmaster.pid	A lock file recording the current server PID and shared memory segmer ID (not present after server shutdown)

Psql command

- Open the psql command line tool
- Use **\l** to list the available databases

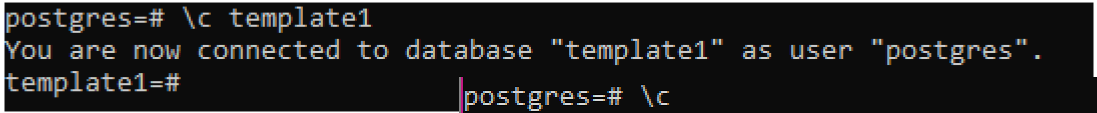


SQL Shell (psql)

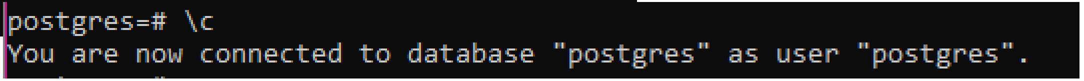
```
Server [localhost]:
Database [postgres]:
Port [5432]:
Username [postgres]:
Password for user postgres:
psql (13.3)
WARNING: Console code page (437) differs from Windows code page (1252)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
Type "help" for help.

postgres=# \l
                                          List of databases
   Name    |  Owner   | Encoding |          Collate           |           Ctype            |   Access privileges
-----------+----------+----------+----------------------------+----------------------------+-----------------------
 postgres  | postgres | UTF8     | English_United States.1252 | English_United States.1252 |
 template0 | postgres | UTF8     | English_United States.1252 | English_United States.1252 | =c/postgres          +
           |          |          |                            |                            | postgres=CTc/postgres
 template1 | postgres | UTF8     | English_United States.1252 | English_United States.1252 | =c/postgres          +
           |          |          |                            |                            | postgres=CTc/postgres
(3 rows)
```

Psql command

- To connect or select a desired database, use **\c databasename**



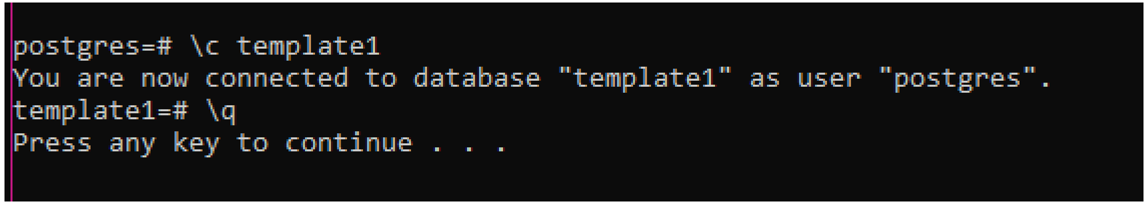
```
postgres=# \c template1
You are now connected to database "template1" as user "postgres".
template1=#
```



```
postgres=# \c
You are now connected to database "postgres" as user "postgres".
```

- You are now logged into PostgreSQL template1 and ready to execute your commands inside template1.

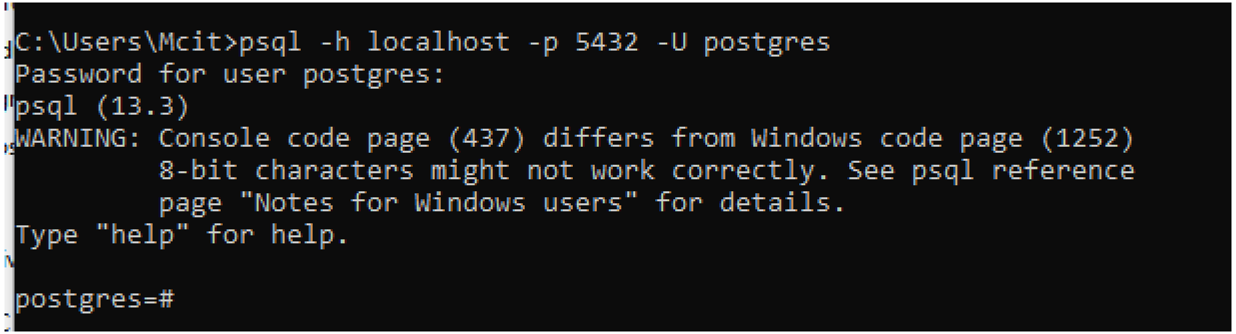
- To exit from the database: you can use **\q.**



```
postgres=# \c template1
You are now connected to database "template1" as user "postgres".
template1=# \q
Press any key to continue . . .
```

Psql Connection from CMD

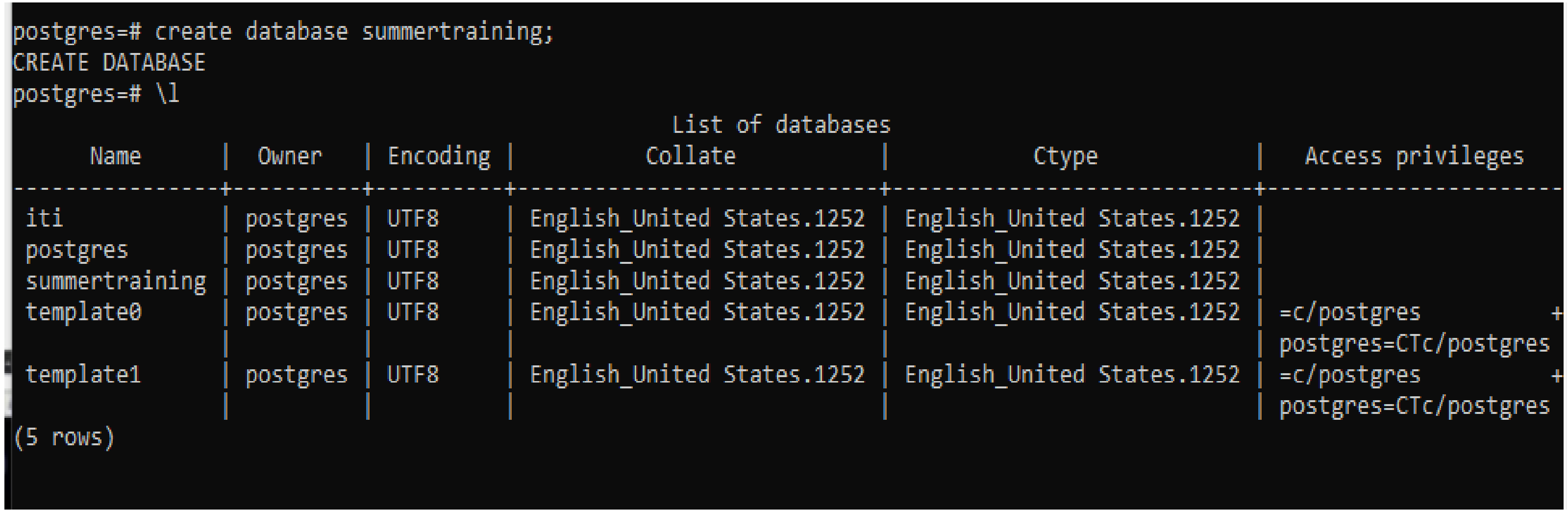
- To connect or select a desired database
    - **Psql –h localhost –p 5432 –U postgres**



```
C:\Users\Mcit>psql -h localhost -p 5432 -U postgres
Password for user postgres:
psql (13.3)
WARNING: Console code page (437) differs from Windows code page (1252)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
Type "help" for help.

postgres=#
```

Create database

- Using SQL command, **Create database databasename;**



```
postgres=# create database summertraining;
CREATE DATABASE
postgres=# \l
                                                         List of databases
      Name      |  Owner   | Encoding |          Collate           |           Ctype            |   Access privileges
----------------+----------+----------+----------------------------+----------------------------+-----------------------
 iti            | postgres | UTF8     | English_United States.1252 | English_United States.1252 |
 postgres       | postgres | UTF8     | English_United States.1252 | English_United States.1252 |
 summertraining | postgres | UTF8     | English_United States.1252 | English_United States.1252 |
 template0      | postgres | UTF8     | English_United States.1252 | English_United States.1252 | =c/postgres          +
                |          |          |                            |                            | postgres=CTc/postgres
 template1      | postgres | UTF8     | English_United States.1252 | English_United States.1252 | =c/postgres          +
                |          |          |                            |                            | postgres=CTc/postgres
(5 rows)
```

Create database

- Using SQL command, **Createdb databasename with linux distributions (bash shell)**

- Write sql code that sent to the server to execute the creation
    - **Createdb –h hostname –p portname –U username –e databasename**

- -e will echo the query sent to the server

- Many options can be used to customize the database creation

Createdb options

Option	Description
-e	Shows the commands being sent to the server.
-V	Print the app version and exit.
--help	Show help about dropdb command-line arguments, and exit.
-h host	Specifies the host name of the machine on which the server is running.
-p port	Specifies the TCP port on which the server is listening for connections.
-U username	Username to connect to the database

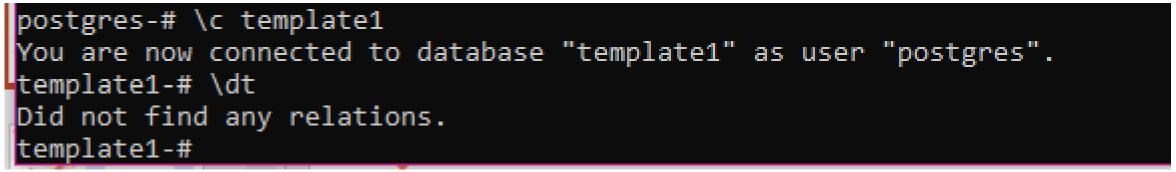
Createdb command

- createdb -h localhost -p 5432 -U postgress testdb
    - password ******

- Above command will prompt you for password of the PostgreSQL admin user which is postgres by default so provide password and proceed to create your new database...

How the database created?!

- When you type **create database databasename** it makes a new copy from the template1 database ...



```
postgres-# \c template1
You are now connected to database "template1" as user "postgres".
template1-# \dt
Did not find any relations.
template1-#
```

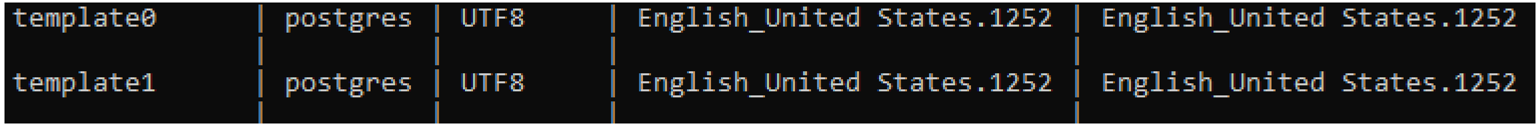
- To go ot template1 database write **\c databasename**
  - \c tempalte1
- To check the relations inside this database
  - \dt → will display the available tables inside this database if exists.

How the database created?!

- If you add objects to template1, these objects will be copied into subsequently created user databases.

- **It is suggested to use lowercase names in identifying your database.**

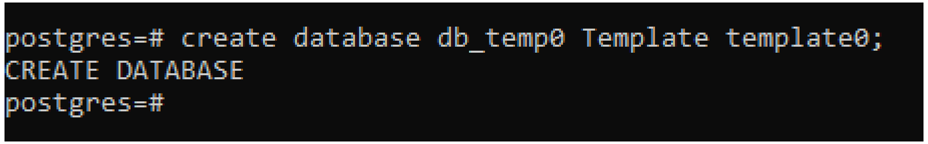
- There is a second standard system database named template0. This database contains the same data as the initial contents of template1, that is, only the standard objects predefined by your version of PostgreSQL



```
template0          | postgres | UTF8      | English_United States.1252 | English_United States.1252
                   |          |           |                            |
template1          | postgres | UTF8      | English_United States.1252 | English_United States.1252
                   |          |           |                            |
```

How the database created?!

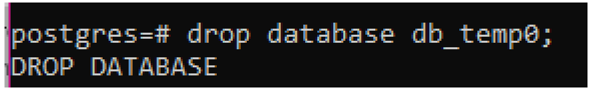
- **CREATE DATABASE** dbname **TEMPLATE** template0;



```
postgres=# create database db_temp0 Template template0;
CREATE DATABASE
postgres=#
```

How be the database Dropped?!

- To drop a database use:
  - **Drop database databasename;**



```
postgres=# drop database db_temp0;
DROP DATABASE
```

- We cannot drop a database that has any open connections, including our own connection from psql or pgAdmin III.
- We must switch to another database or template1 if we want to delete the database we are currently connected to.

How be the database Dropped?!

- **From bash shell use dropdb command to drop the database..**
    - **Dropdb options databasename**
    - dropdb -h localhost -p 5432 -U postgress testdb
    - Password for user postgress: ****

The above command drops database testdb.

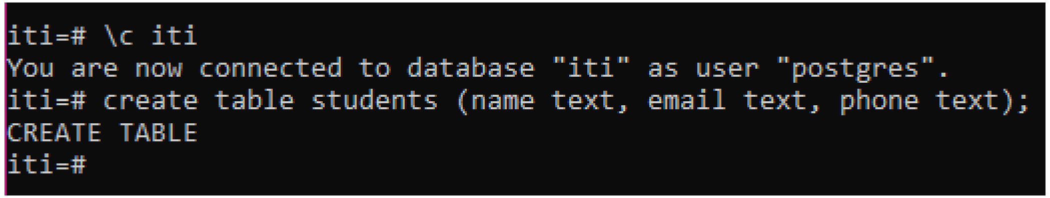
dropdb options

Option	Description
-e	Shows the commands being sent to the server.
-V	Print the app version and exit.
-i	Issues a verification prompt before doing anything destructive.
-if exists	Do not throw an error if the database does not exist.
--help	Show help about dropdb command-line arguments, and exit.
-h host	Specifies the host name of the machine on which the server is running.
-p port	Specifies the TCP port on which the server is listening for connections.
-U username	Username to connect to the database

Create your first table

- Create table syntax:

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

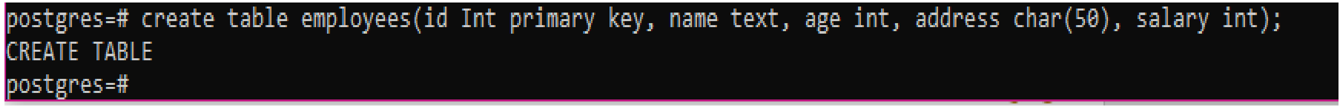


```
iti=# \c iti
You are now connected to database "iti" as user "postgres".
iti=# create table students (name text, email text, phone text);
CREATE TABLE
iti=#
```

Create your first table

- Create table example:

```
CREATE TABLE employees(
ID INT PRIMARY KEY,
NAME TEXT,
AGE INT,
ADDRESS CHAR(50),
SALARY INT
);
```



```
postgres=# create table employees(id Int primary key, name text, age int, address char(50), salary int);
CREATE TABLE
postgres=#
```

Columns' datatypes in PostgreSQL

- Postgres has many different columns datatypes
- User can add new datatypes using **Create type** command
- Data types can be grouped as
    - Numeric data type
    - Monetary data type
    - Character data types
    - Date/Time data type
    - Boolean data type
    - Enumerated Types

Numeric data type

Name	Storage Size	Description	Range
smallint	2 bytes	small-range integer	-32768 to +32767
integer	4 bytes	typical choice for integer	-2147483648 to +2147483647
bigint	8 bytes	large-range integer -	9223372036854775808 to +9223372036854775807
numeric(precision, scale)	Variable	user-specified precision, exact	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point, number 23.5141 has a precision of 6 and a scale of 4.

Numeric data type

Name	Storage Size	Description	Range
smallserial	2 bytes	small autoincrementing integer	1 to 32767
serial	4 bytes	autoincrementing integer	1 to 2147483647
bigserial	8 bytes	large autoincrementing integer	1 to 9223372036854775807

Monetary data type

- The money type stores a currency amount with a fixed fractional precision.
- Input is accepted in a variety of formats, as integer and floating-point literals, as well as typical currency formatting, such as '$1,000.00'

Name	**Storage Size**	**Description**	**Range**
Money	8 bytes	currency amount	-92233720368547758.08 to +92233720368547758.07

Character data types

- An attempt to store a longer string will result in an error, unless the excess characters are all spaces, in this case string will be truncated to the maximum
- If the string is shorter than the declared length, values of type character will be space-padded;
- values of type character varying will simply store the shorter string.

Name	Description
character varying(n), varchar(n)	variable-length with limit
character(n), char(n)	fixed-length, blank padded
text	variable unlimited length

Date/Time data type

- Valid input for the time stamp types consists of the concatenation of a date and a time, followed by an optional time zone, followed by an optional AD or BC.

Name	**Storage Size**	**Description**	**Example**
timestamp [without time zone]	8 bytes	both date and time (no time zone) From 4713 BC to 294276 AD	1999-01-08 04:05:06 January 8 04:05:06 99 BC
timestamp [with time zone]	8 bytes	both date and time (with time zone) From 4713 BC to 294276 AD	1999-01-08 04:05:06 January 8 04:05:06 99 BC

Date/Time data type

Name	**Description**
date	date (no time of day)
Time(without time zone)	time of day (no date)
time with time zone	times of day only, with time zone
interval [fields]	time interval, field can be YEAR, MONTH, DAY, HOUR, MINUTE, SECOND

Boolean data type

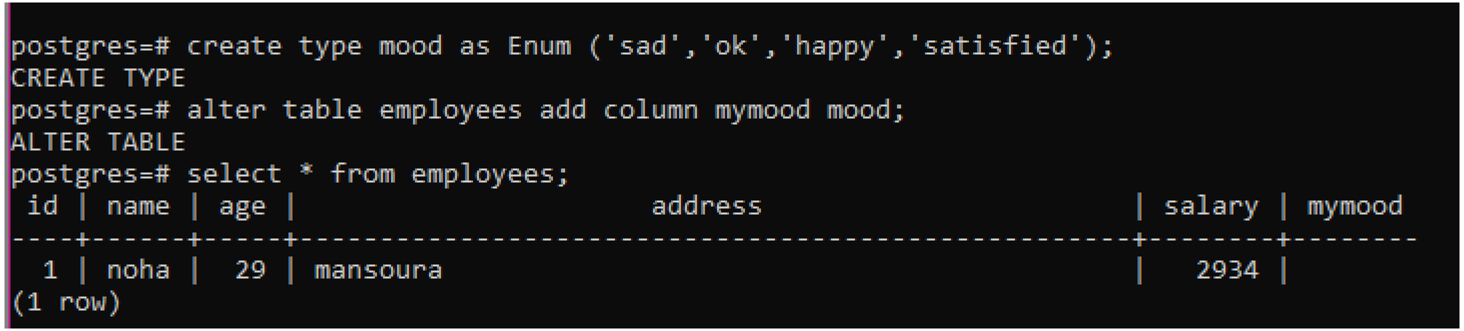
- Valid literal values for the "true" state are:
  - TRUE , 't‘ , 'true‘, 'y‘, 'yes‘, 'on‘ or ‘1‘
- For the "false" state, the following values can be used:
  - FALSE, 'f‘, 'false‘, 'n‘, 'no‘, 'off‘ or '0'

Name	Size	Description
Boolean	1 byte	state of true or false

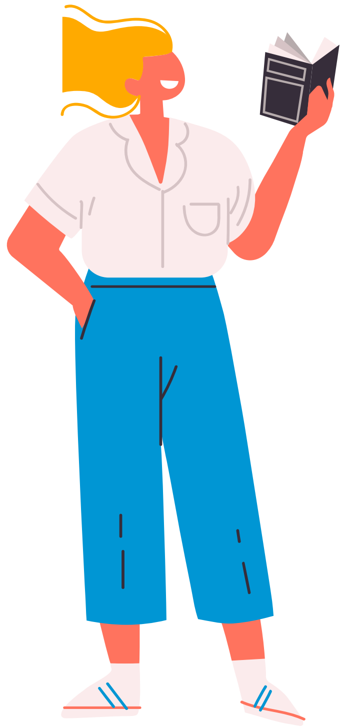
Enumerated Types

- Enumerated (enum) types are data types that comprise a static, ordered set of values.
- Enum types are created using the CREATE TYPE command, for example:

- **CREATE TYPE mood AS ENUM ('sad', 'ok', 'happy');**
- **SELECT * FROM person WHERE current_mood > ’ok’;**



```
postgres=# create type mood as Enum ('sad','ok','happy','satisfied');
CREATE TYPE
postgres=# alter table employees add column mymood mood;
ALTER TABLE
postgres=# select * from employees;
 id | name | age |                    address                     | salary | mymood
----+------+-----+------------------------------------------------+--------+--------
  1 | noha |  29 | mansoura                                       |   2934 |
(1 row)
```



Lab time

Lab 01

- Install PostgreSQL DBMS.
- The scenario is that as an instructor in ITI, you have grade-keeping responsibilities. You want to convert the grading process from a manual operation using a gradebook to an electronic representation using database. In this case:
- For each student, you keep track his name, contact info (email, address), and multiple phone numbers.
- Each Student belong to Track (Telecom, OpenSource, Java, Game,..), and each track have different students

Lab 01

- Each track have different subjects/courses such as (C, CPP, HTML, ...), and each student study subjects/courses based on the track that he belong to it
- For each subject, you need to define the name and the description and max score (total grade 100).
- The students achieve score in subject by exam result.
- For each exam which taken by student you must store Exam date, student score in exam (such as 75).

Lab 01

- Keep track of Students and Track which he belong to it, and subjects owned by Tack , and Subjects which will studied by each student
- Design ERD and write down the mapping schema.
- Create your mapped tables with their columns in PostgreSQL.
- Insert at minimum 5 Rows at each table

Break time

Thanks

Noha
nshehab@iti.gov.eg