PostgreSQL

**IMP**

NOT NULL

UNIQUE CONSTRAINT

CHECK CONSTRAINT

PRIMARY KEY

DATA TYPE

LENGTH

**Create Database: --**

CREATE DATABASE db\_name;

/l

**Connect to DB: --**

-h hostUrl -p 5423 -u username dbName

-h localhost -p 5423 -u jayanta test

\c test

**Creating Table: --**

CREATE TABLE table\_name (column name + data type + constrains if any)

CREATE TABLE persion (

Id BIGSERIAL NOT NULL PRIMARY KEY,

first\_name VARCHAR (50) NOT NULL,

last\_name VARCHAR (50) NOT NULL,

gender VARCHAR (50) NOT NULL,

date\_of\_birth DATE NOT NULL,

)

test\_db=# CREATE TABLE person (

test\_db(# id BIGSERIAL NOT NULL PRIMARY KEY,

test\_db(# first\_name VARCHAR(50) NOT NULL,

test\_db(# last\_name VARCHAR(50) NOT NULL,

test\_db(# gender VARCHAR(10) NOT NULL,

test\_db(# date\_of\_birth DATE NOT NULL,

test\_db(# email VARCHAR(200));

\d or \d persion

gender

**DROP Table**

DROP TABLE table\_name

**Insert Records into table: --**

INSERT INTO person (

First\_name,

Last\_name,

Gender,

Date\_of\_birth) values (‘jayanta’, ‘lahkar’, DATE ‘1996-01-09’);

**READ Query**

SELECT \* FROM person;

**MOCKARO**

<https://www.mockaroo.com/>

Fake data generate for db

Open the generated data in VS cCode

Make the respective columns as NOT NULL and data type

\?

\i /Users/30k/Downloads/person.sql

All data will be inserted to table

**ORDER BY**

ASC, DESC

SELECT \* FROM person ORDER BY country ASC;

SELECT \* FROM person ORDER BY country DESC;

Only Show Country in ASC order ->

SELECT country from person ORDER BY country ASC;

Show only country without repetition ->

SELECT DISTINCT country from person ORDER BY country ASC;

**WHERE Clause**

SELECT \* FROM person WHERE gender = 'Male';

SELECT \* FROM person WHERE gender = 'Male' AND (country = 'India' OR country = 'China');

LIMIT and OFFSET

SELECT \* FROM person LIMIT 10;

Select the data from 11 column ->

SELECT \* FROM person OFFSET 10;

**IN Keyword**

SELECT \* FROM person where country IN ('India', 'Bhutan', 'Brazil');

**BETWEEN Keyword**

SELECT \* FROM person WHERE date\_of\_birth BETWEEN '2001-01-07' AND '2021-01-01';

**LIKE Operator**

Any person email end with .com ->

SELECT \* FROM person WHERE email LIKE '%.com';

Any person email exactly end with google.com ->

SELECT \* FROM person WHERE email LIKE '%@google.com';

It will match anything after @google ->

SELECT \* FROM person WHERE email LIKE '%@google.%';

Select all the people whose first name start with P->

SELECT \* FROM person WHERE first\_name LIKE 'P%';

LIKE is case sensitive so we use ILIKE so it will match with lowercase and uppercase P ->

SELECT \* FROM person WHERE first\_name ILIKE 'p%';

**GROUP BY**

No of person present in each country ->

SELECT country, COUNT (\*) FROM person GROUP BY country;

**HAVING filtering**

SELECT country, COUNT(\*) FROM person GROUP BY country HAVING COUNT(\*)>5 ORDER BY country ASC;

**MAX, MIN, AVG, ROUND**

SELECT MAX (price) FROM car;

SELECT MIN (price) FROM carr

**PRIMARY KEY**

Primary key uniquely identifies any records in a table

**Drop Primary Key**

\d table name

ALTER TABLE person DROP CONSTRAIN person\_pkey;

**ADD Primary Key**

ALTER TABLE person ADD PRIMARY KEY (id);

**DELETE a Records**

DELETE FROM person WHERE id=1;

**UNIQUE Constrain**

Unique constrains allow unique value per column

It is not same as primary key since Primary key use to uniquely identify a row and unique constrain means that we can have only unique constrain pr column

This will not allow to enter duplicate email in db->

ALTER TABLE person ADD CONSTRAINT unique\_email\_address UNIQUE (email);

Drop a unique constraint

ALTER TABLE CONSTRAINT unique\_email\_address;

ADD a Unique constraint

ALTER TABLE person ADD UNIQUE (email);

**Check Constraint**

Suppose we have gender field in our db which could be MALE/FEMAL

So unique constrain will allow only either male or female in the file. It won’t allow any other value accept male or female

ADD CHECK Constraint

ALTER TABLE person ADD CONSTRAINT gender\_constraint CHECK (gender = 'Male' OR gender = 'Female');

Table "public.person"

Column | Type | Collation | Nullable | Default

---------------+------------------------+-----------+----------+------------------------------------

id | bigint | | not null | nextval('person\_id\_seq'::regclass)

first\_name | character varying(50) | | not null |

last\_name | character varying(50) | | not null |

email | character varying(200) | | |

gender | character varying(10) | | not null |

date\_of\_birth | date | | not null |

country | character varying(50) | | |

Indexes:

"person\_pkey" PRIMARY KEY, btree (id)

"unique\_email\_address" UNIQUE CONSTRAINT, btree (email)

Check constraints:

"gender\_constraint" CHECK (gender::text = 'Male'::text OR gender::text = 'Female'::text)

**UPDATE**

UPDATE person SET email=’jayanta@gmail’ WHERE id=102;

UPDATE person SET first\_name = ‘Jayanta’, last\_name = ‘lahkar’, email =”jayanta@gmail” WHERE id = 102;

**FOREGIN Key**

One person can have one car

One car belongs to only one person

create table person (

    id BIGSERIAL NOT NULL PRIMARY KEY,

    first\_name VARCHAR(50) NOT NULL,

    last\_name VARCHAR(50) NOT NULL,

    gender VARCHAR(7) NOT NULL,

    email VARCHAR(100),

    date\_of\_birth DATE NOT NULL,

    country\_of\_birth VARCHAR(50) NOT NULL,

// NOT NULL because a person may not have car

    car\_id BIGINT REFERENCES car(id),

//UNIQUE because a person can have at most 1 car

    UNIQUE(car\_id)

);

create table car (

    id BIGSERIAL NOT NULL PRIMARY KEY,

    make VARCHAR(100) NOT NULL,

    model VARCHAR(100) NOT NULL,

    price NUMERIC(19, 2) NOT NULL

);

insert into person (first\_name, last\_name, gender, email, date\_of\_birth, country\_of\_birth) values ('Fernanda', 'Beardon', 'Female', 'fernandab@is.gd', '1953-10-28', 'Comoros');

insert into person (first\_name, last\_name, gender, email, date\_of\_birth, country\_of\_birth) values ('Omar', 'Colmore', 'Male', null, '1921-04-03', 'Finland');

insert into person (first\_name, last\_name, gender, email, date\_of\_birth, country\_of\_birth) values ('John', 'Matuschek', 'Male', 'john@feedburner.com', '1965-02-28', 'England');

insert into car (make, model, price) values ('Land Rover', 'Sterling', '87665.38');

insert into car (make, model, price) values ('GMC', 'Acadia', '17662.69');

Assigning car of id 2 to person of id 1

UPDATE person SET car\_id= 2 WHERE id=1;

**INNER Joins**

Inner join is the way of joining or combining tow table. As a result, it will give the common entry of both the table

SELECT \* FROM person JOIN car ON person.car\_id = car.id;

**LEFT JOIN**

Left join use to join a or combine two table as a result it returns all the rows(records) from left table and matching records from right table

**RIGHT JOIN**

Right join use to join a or combine two table as a result it returns all the rows(records) from right table and matching records from left table

**Generate CSV**

\copy (SELECT \* FROM person LEFT JOIN car ON car.id = person.car\_id) TO '/Users/30k/Downloads/result.csv' DELIMITER ',' CSV HEADER;