WORLD’S MOST ADVANCED OPEN SOURCE object-RELATIONAL DATABASE

Applications & Tools:
PGAdmin 4 (PostgreSQL GUI)
Dummy & Random Data Generator Tool: **https://www.mockaroo.com/**
/?   //for help in psql
\help <command_name>

Install Postgre on linux

1. sudo apt-get update
2. sudo apt-get upgrade
3. sudo apt-get install postgresql postgresql-contrib

Start PostgreSQL CLI

- service postgresql status //running status of PostgreSQL
- sudo su postgres   //login as postgres user
- psql   //start PostgreSQL cli in Terminal
- \q   //to exit PostgreSQL cli

In PostgreSQL CLI   (psql   //start PostgreSQL cli in Terminal)

- \l (small L)   //list of databases
- \du   //list of users of PSQL DBMS
- CREATE DATABASE test ;
- DROP DATABASE test ;
-
- **ALTER USER postgres PASSWORD 'admin';   \\ to alter password of a user**
- **\c test   //to connect to a database;**
-

Connect to database (after loging as sudo su postgre)

psql --help
Default Hostname: "/var/run/postgresql"
Default Port: "5432"
Default Username: "postgres"
~~**-Default Password-**~~ I had set my postgresql password admin

CMD: psql -h localhost -p 5432 -U postgres DB_name
Exit: \q
or
\l
\c DB_name

In a Database   (\c db_name   //to connect to a database;)

- CREATE TABLE employee(
  id BIGSERIAL **NOT NULL PRIMARY KEY,**
  age INT **NOT NULL**,
  full_name VARCHAR(60) **NOT NULL**,
  gender VARCHAR(7) **NOT NULL**,
  dob DATE **NOT NULL**,
  );
  BIGSERIAL == BIGINT it increment by themselves.

- \d employee; //structure of table
  in pgadmin: Servers-> learn->db_name->Schemas->public->Tables

- DROP TABLE table_name;

- INSERT INTO table_name(col1,col2,...) VALUES(102, 'val2' , ...) ;
  INSERT INTO table_name VALUES(102, 'val2' , ..., 'valn') ; //no need to mention col_name if we are inserting in all colmns;
  EX: **INSERT INTO** person (first_name,last_name,gender, dob) **VALUES** ('Anne', 'Smith', DATE '1988-01-09') **;**

- SELECT ***** FROM table_name;
  SELECT col1, col2 FROM table_name;
  SELECT **DISTINCT** col1, col2 FROM table_name;

- SELECT **DISTINCT** col1, col2 FROM table_name **WHERE** col1='fdddff';
- **AND || OR || ORDER BY** col_name | col1, col2 | **ASC** (default) **| DESC**
- **LIMIT** 9 | 2*3-1;

- **UPDATE** table_name **SET** col_name = 'new updated val' **WHERE** col_name2 = 'val to search' **;**

- **DELETE FROM** table_name **WHERE** col_name2 = 'val to search' **;**
- **DROP TABLE** table_name**;** (erase the table from db + unrestorable + no log is maintained)
- **TRUNCATE TABLE** table_name; (delete all the records of the data + log is maintained)

- ALTER TABLE:
  ADD NEW COL =>**ALTER TABLE** table_name **ADD** newcol_name datatype;
  DROP A COL =>**ALTER TABLE** table_name **DROP** col_name;
  MODIFY A COL=>**ALTER TABLE** table_name **MODIFY** col_name newdatatype;

- WHERE col_name **BETWEEN** val1 **AND** val2; ==== colname>=val1 **AND** colname<=val2 **;**
- Comparison Operators: **Equal to (==): = , Not Equal to (!=): <> , rest....is same**
- WHERE col1 **IN (**val1, val2, val3**);**
- WHERE col1 **LIKE '**p**%' ;** OR **'_**p**'** //LIKE is CASE SENSITIVE
  WHERE col1 **ILIKE '**p**%' ;** OR **'_**p**'** ==(LIKE 'P%' + LIKE 'p%) //ILIKE is CASE IN-SENSITIVE

- **GROUP BY:**
  SELECT country, COUNT(*) FROM person **GROUP BY** country ORDER BY country;

- **HAVING:** (must be after GROUP BY and before ORDER BY)
  SELECT country, COUNT(*) FROM person GROUP BY country **HAVING** COUNT(*) > 40 ORDER BY country;

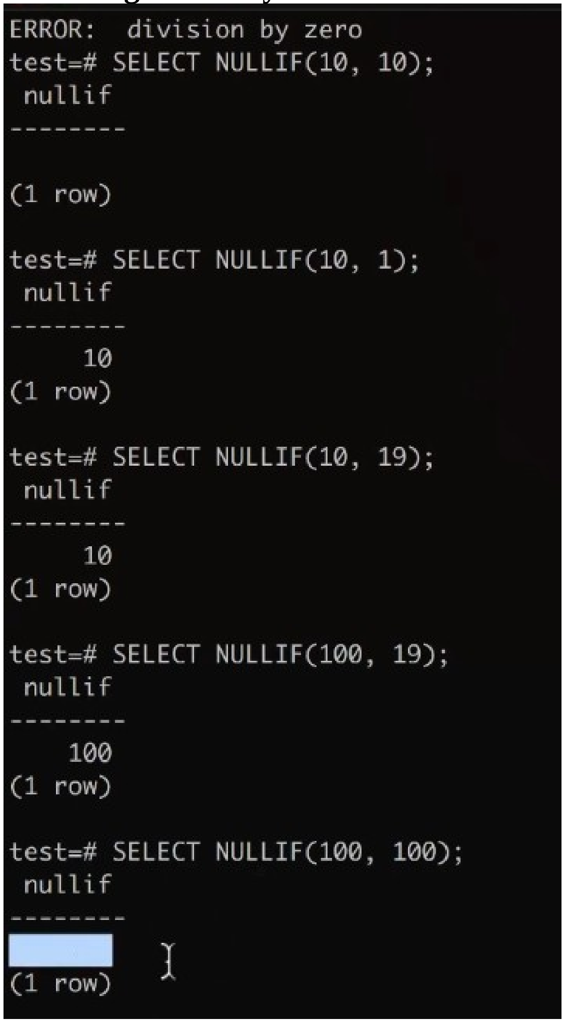
- AGGREGATORS : **MIN, MAX, COUNT,etc**
  MAX: SELECT **MAX**(price) FROM car;
  Eg: SELECT make, **MAX(**price**)** FROM car GROUP BY make;

  MIN: SELECT **MIN**(price) FROM car;
  AVG: SELECT **AVG**(price) FROM car;

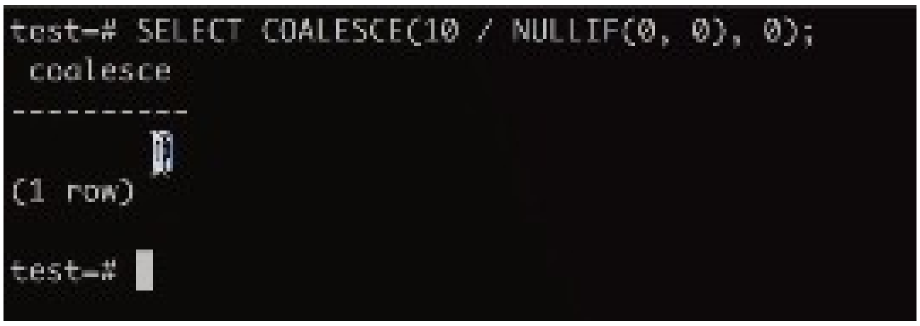
  ROUND: SELECT **ROUND(**AVG(price)**)** FROM car;
  EG: SELECT make, price,**ROUND(**price*.10 , 2**) AS** discount FROM car;
  //it will show 10% price of cars upto 2 precision.

  SUM: SELECT make, **SUM(**price**)** FROM car GROUP BY make;

- **Handling Null Values:**
  SELECT email FROM person; //it will print values all records on email col and if it is NULL blank will be print.
  SELECT **COALESCE(**email, **“<Default Value>”)** FROM person; //it will print Default in place of NULL.
- **Handling Divide by 0 Error:**



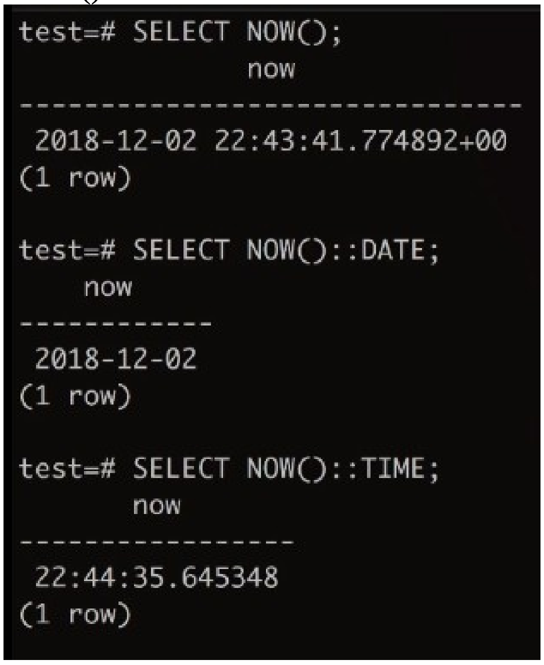
```
ERROR:  division by zero
test=# SELECT NULLIF(10, 10);
 nullif
--------

(1 row)

test=# SELECT NULLIF(10, 1);
 nullif
--------
     10
(1 row)

test=# SELECT NULLIF(10, 19);
 nullif
--------
     10
(1 row)

test=# SELECT NULLIF(100, 19);
 nullif
--------
    100
(1 row)

test=# SELECT NULLIF(100, 100);
 nullif
--------

(1 row)
```



```
test=# SELECT COALESCE(10 / NULLIF(0, 0), 0);
 coalesce
----------
        0
(1 row)

test=# 
```

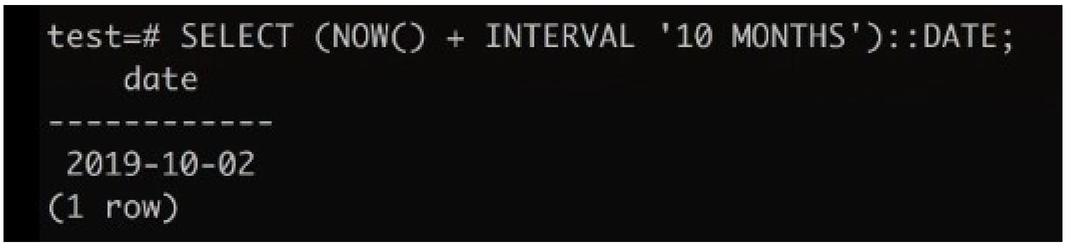
- **Timestamp and Date**

NOW()



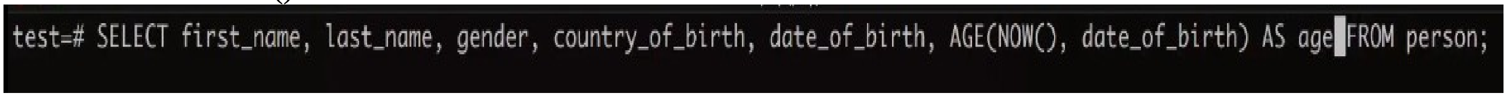
```
test=# SELECT NOW();
              now
-------------------------------
 2018-12-02 22:43:41.774892+00
(1 row)

test=# SELECT NOW()::DATE;
    now
------------
 2018-12-02
(1 row)

test=# SELECT NOW()::TIME;
       now
-----------------
 22:44:35.645348
(1 row)
```

INTERVAL

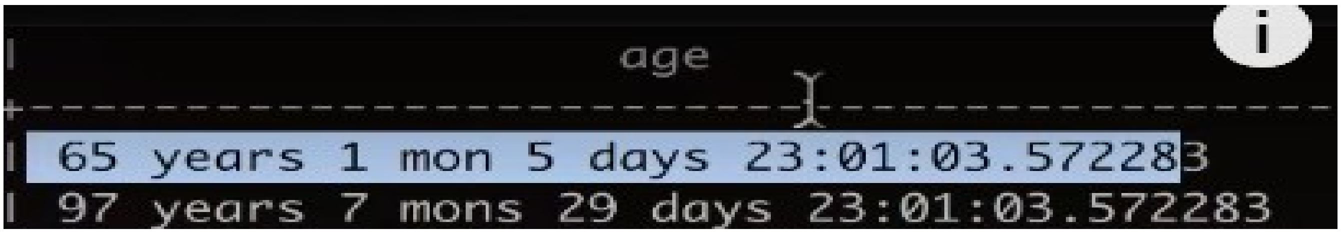


```
test=# SELECT (NOW() + INTERVAL '10 MONTHS')::DATE;
    date
------------
 2019-10-02
(1 row)
```

AGE()



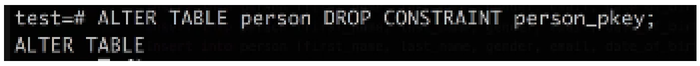
```
test=# SELECT first_name, last_name, gender, country_of_birth, date_of_birth, AGE(NOW(), date_of_birth) AS age FROM person;
```



```
|                    age
+--------------------------------------------
| 65 years 1 mon 5 days 23:01:03.572283
| 97 years 7 mons 29 days 23:01:03.572283
```

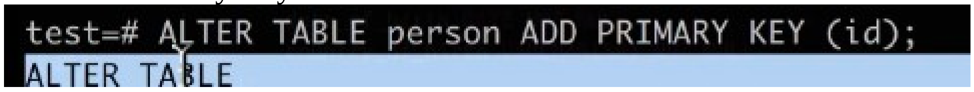
- **Handling CONSTRAINT:**

  Drop all CONSTRAINTs including UNIQUE and PRIMARY KEY.:



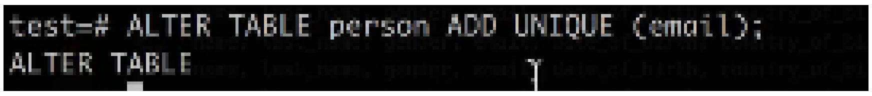
```
test=# ALTER TABLE person DROP CONSTRAINT person_pkey;
ALTER TABLE
```

  Add Back Primary Key:

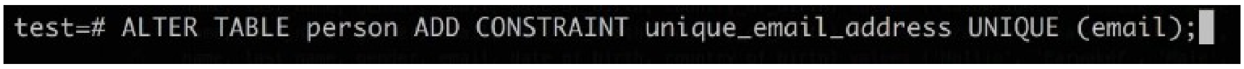


```
test=# ALTER TABLE person ADD PRIMARY KEY (id);
ALTER TABLE
```

  Add UNIQUE Constraints:

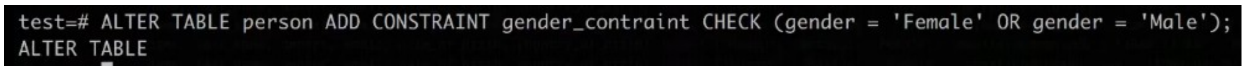


```
test=# ALTER TABLE person ADD UNIQUE (email);
ALTER TABLE
```



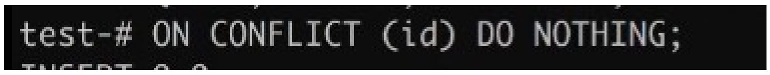
```
test=# ALTER TABLE person ADD CONSTRAINT unique_email_address UNIQUE (email);
```

  Add CHECK Contraints:

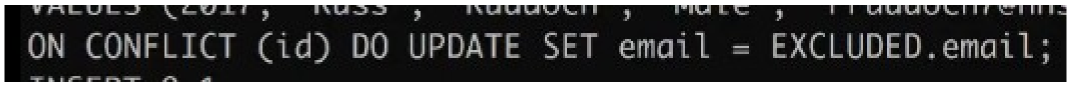


```
test=# ALTER TABLE person ADD CONSTRAINT gender_contraint CHECK (gender = 'Female' OR gender = 'Male');
ALTER TABLE
```

  Add ON CONFLICT Constraints:

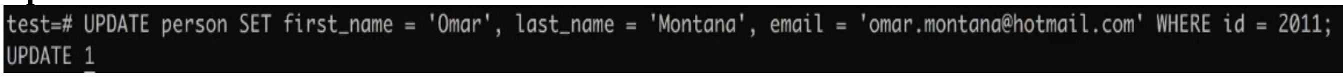


```
test-# ON CONFLICT (id) DO NOTHING;
```



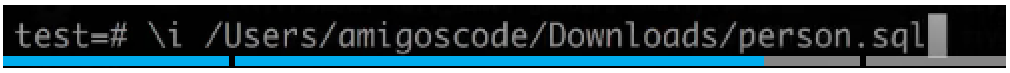
```
ON CONFLICT (id) DO UPDATE SET email = EXCLUDED.email;
```

- **Update Records:**



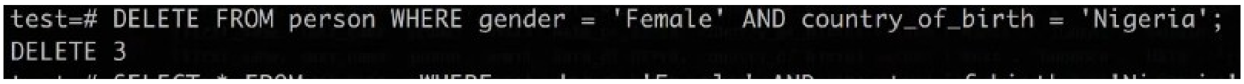
```
test=# UPDATE person SET first_name = 'Omar', last_name = 'Montana', email = 'omar.montana@hotmail.com' WHERE id = 2011;
UPDATE 1
```

- **Execute a file in PostgreSQL**



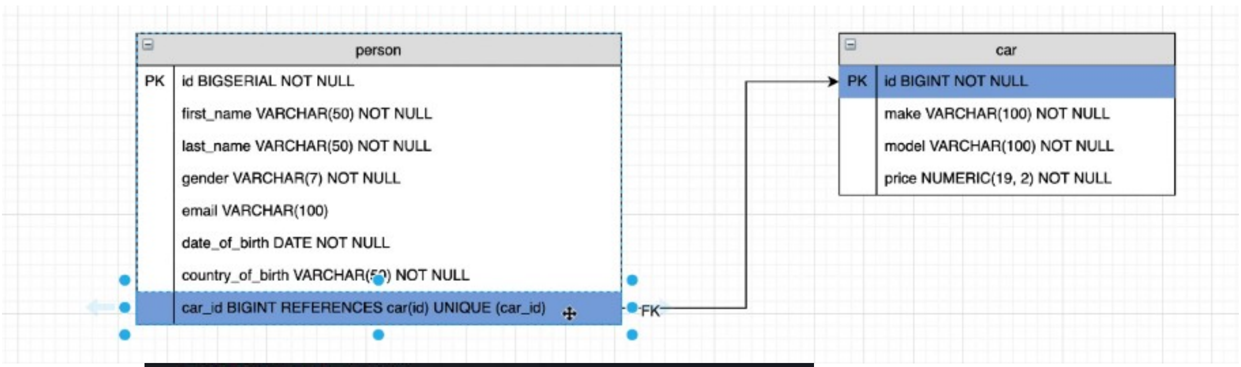
```
test=# \i /Users/amigoscode/Downloads/person.sql
```

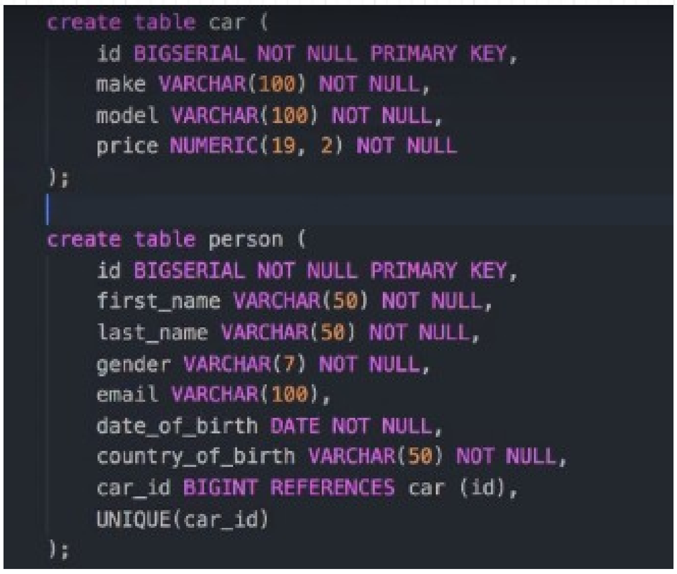
- **DELETE a record:**



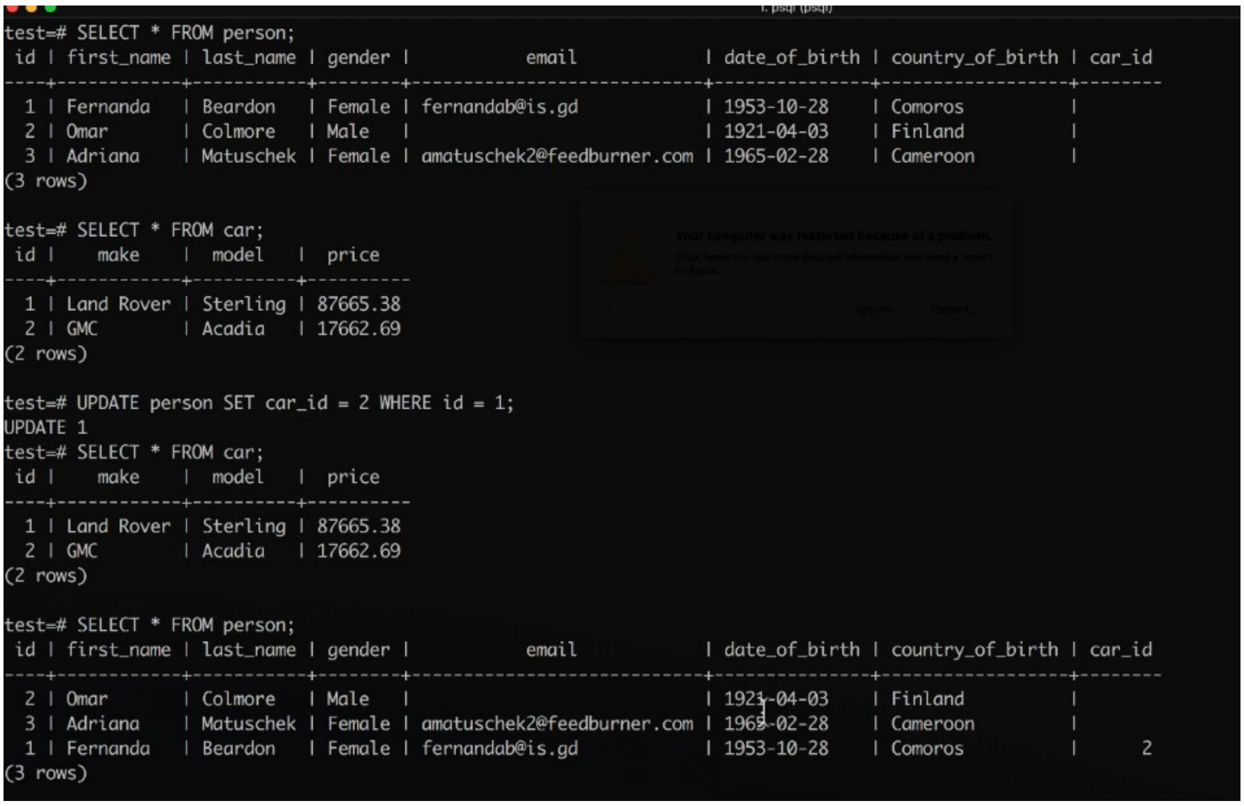
```
test=# DELETE FROM person WHERE gender = 'Female' AND country_of_birth = 'Nigeria';
DELETE 3
```

Relationship:


person
PK
id BIGSERIAL NOT NULL
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
car_id BIGINT REFERENCES car(id) UNIQUE (car_id)
FK
car
PK
id BIGINT NOT NULL
make VARCHAR(100) NOT NULL
model VARCHAR(100) NOT NULL
price NUMERIC(19, 2) NOT NULL



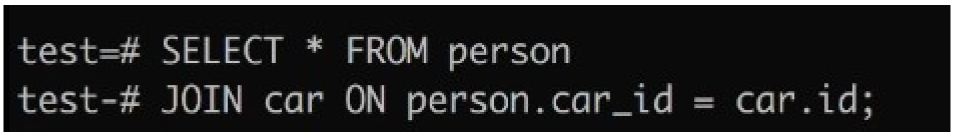
```
create table car (
    id BIGSERIAL NOT NULL PRIMARY KEY,
    make VARCHAR(100) NOT NULL,
    model VARCHAR(100) NOT NULL,
    price NUMERIC(19, 2) NOT NULL
);

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,
    car_id BIGINT REFERENCES car (id),
    UNIQUE(car_id)
);
```



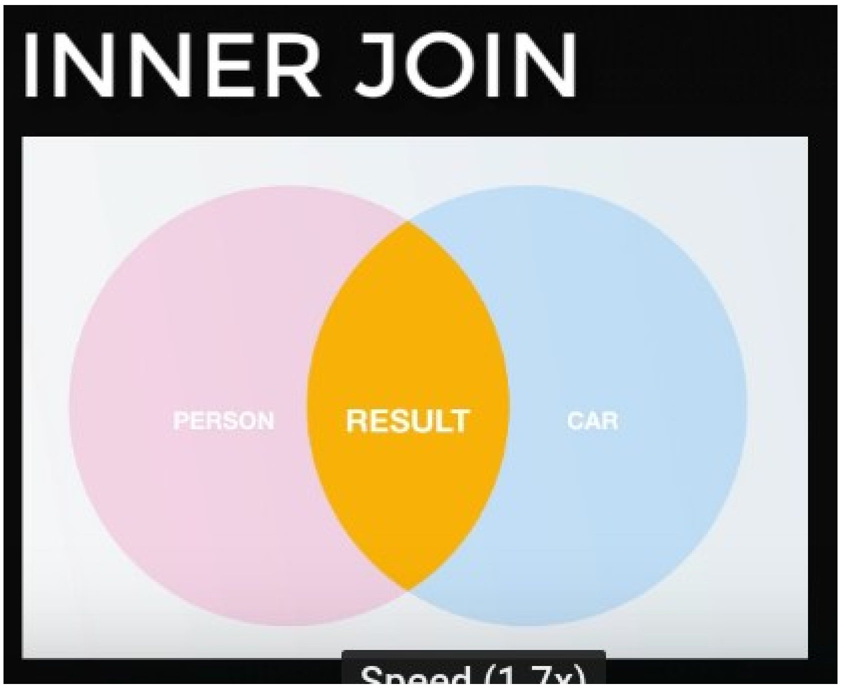
```
test=# SELECT * FROM person;
 id | first_name | last_name | gender |           email            | date_of_birth | country_of_birth | car_id
----+------------+-----------+--------+----------------------------+---------------+------------------+--------
  1 | Fernanda   | Beardon   | Female | fernandab@is.gd            | 1953-10-28    | Comoros          |
  2 | Omar       | Colmore   | Male   |                            | 1921-04-03    | Finland          |
  3 | Adriana    | Matuschek | Female | amatuschek2@feedburner.com | 1965-02-28    | Cameroon         |
(3 rows)

test=# SELECT * FROM car;
 id |    make    |  model   |  price
----+------------+----------+----------
  1 | Land Rover | Sterling | 87665.38
  2 | GMC        | Acadia   | 17662.69
(2 rows)

test=# UPDATE person SET car_id = 2 WHERE id = 1;
UPDATE 1
test=# SELECT * FROM car;
 id |    make    |  model   |  price
----+------------+----------+----------
  1 | Land Rover | Sterling | 87665.38
  2 | GMC        | Acadia   | 17662.69
(2 rows)

test=# SELECT * FROM person;
 id | first_name | last_name | gender |           email            | date_of_birth | country_of_birth | car_id
----+------------+-----------+--------+----------------------------+---------------+------------------+--------
  2 | Omar       | Colmore   | Male   |                            | 1921-04-03    | Finland          |
  3 | Adriana    | Matuschek | Female | amatuschek2@feedburner.com | 1965-02-28    | Cameroon         |
  1 | Fernanda   | Beardon   | Female | fernandab@is.gd            | 1953-10-28    | Comoros          |      2
(3 rows)
```

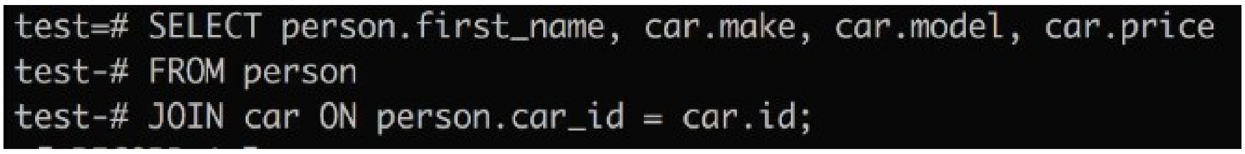
- **JOINS:**

Inner Join



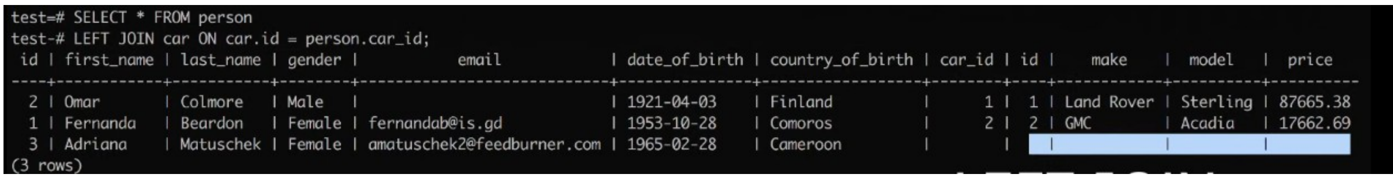
```
test=# SELECT * FROM person
test-# JOIN car ON person.car_id = car.id;
```



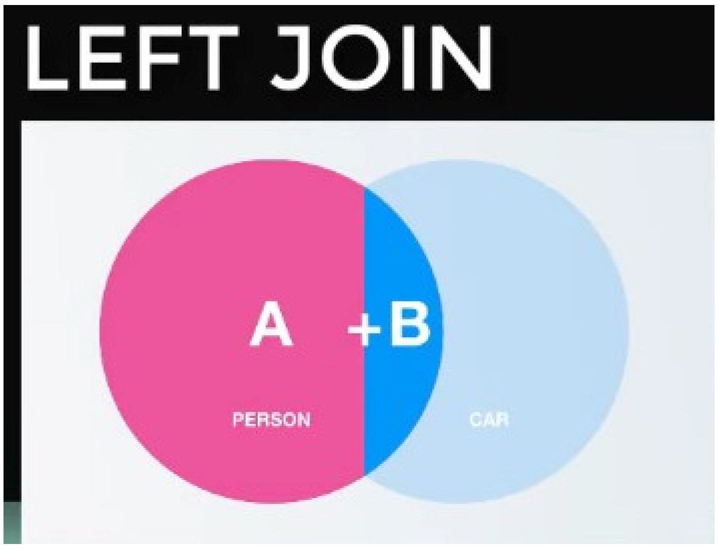


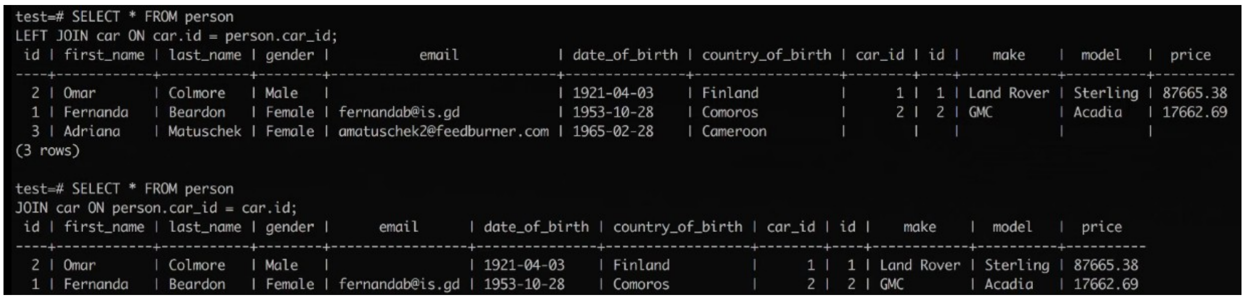
```
test=# SELECT person.first_name, car.make, car.model, car.price
test-# FROM person
test-# JOIN car ON person.car_id = car.id;
```

LEFT JOIN



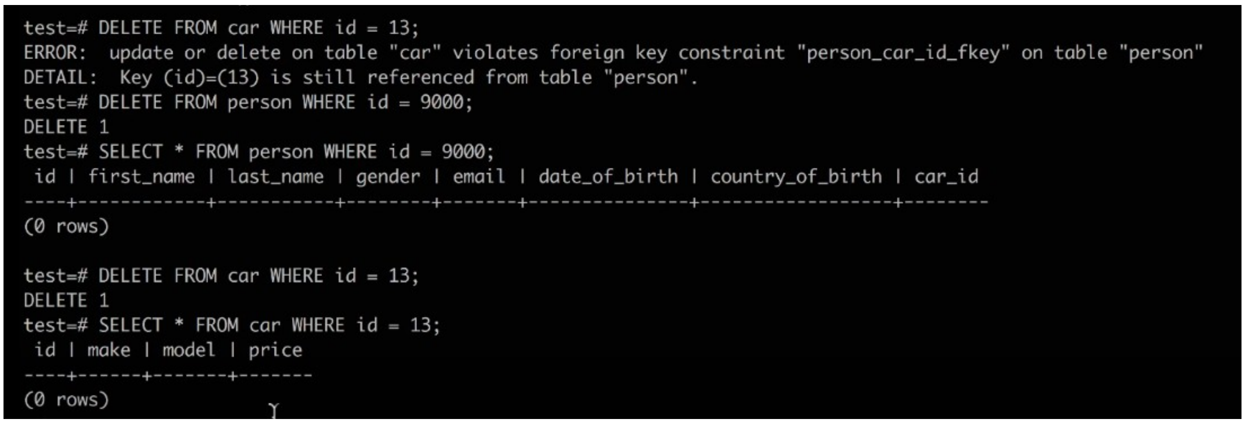
```
test=# SELECT * FROM person
test-# LEFT JOIN car ON car.id = person.car_id;
 id | first_name | last_name | gender |           email            | date_of_birth | country_of_birth | car_id | id |    make    |  model   |  price
----+------------+-----------+--------+----------------------------+---------------+------------------+--------+----+------------+----------+----------
  2 | Omar       | Colmore   | Male   |                            | 1921-04-03    | Finland          |      1 |  1 | Land Rover | Sterling | 87665.38
  1 | Fernanda   | Beardon   | Female | fernandab@is.gd            | 1953-10-28    | Comoros          |      2 |  2 | GMC        | Acadia   | 17662.69
  3 | Adriana    | Matuschek | Female | amatuschek2@feedburner.com | 1965-02-28    | Cameroon         |        |    |            |          |
(3 rows)
```





```
test=# SELECT * FROM person
LEFT JOIN car ON car.id = person.car_id;
 id | first_name | last_name | gender |           email            | date_of_birth | country_of_birth | car_id | id |    make    |  model   |  price
----+------------+-----------+--------+----------------------------+---------------+------------------+--------+----+------------+----------+----------
  2 | Omar       | Colmore   | Male   |                            | 1921-04-03    | Finland          |      1 |  1 | Land Rover | Sterling | 87665.38
  1 | Fernanda   | Beardon   | Female | fernandab@is.gd            | 1953-10-28    | Comoros          |      2 |  2 | GMC        | Acadia   | 17662.69
  3 | Adriana    | Matuschek | Female | amatuschek2@feedburner.com | 1965-02-28    | Cameroon         |        |    |            |          |
(3 rows)

test=# SELECT * FROM person
JOIN car ON person.car_id = car.id;
 id | first_name | last_name | gender |      email      | date_of_birth | country_of_birth | car_id | id |    make    |  model   |  price
----+------------+-----------+--------+-----------------+---------------+------------------+--------+----+------------+----------+----------
  2 | Omar       | Colmore   | Male   |                 | 1921-04-03    | Finland          |      1 |  1 | Land Rover | Sterling | 87665.38
  1 | Fernanda   | Beardon   | Female | fernandab@is.gd | 1953-10-28    | Comoros          |      2 |  2 | GMC        | Acadia   | 17662.69
```

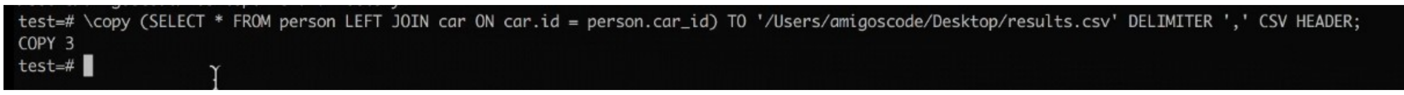
DELETING RECORDS IN REFERNCED AND REFRENCING TABLE:



```
test=# DELETE FROM car WHERE id = 13;
ERROR:  update or delete on table "car" violates foreign key constraint "person_car_id_fkey" on table "person"
DETAIL:  Key (id)=(13) is still referenced from table "person".
test=# DELETE FROM person WHERE id = 9000;
DELETE 1
test=# SELECT * FROM person WHERE id = 9000;
 id | first_name | last_name | gender | email | date_of_birth | country_of_birth | car_id
----+------------+-----------+--------+-------+---------------+------------------+--------
(0 rows)

test=# DELETE FROM car WHERE id = 13;
DELETE 1
test=# SELECT * FROM car WHERE id = 13;
 id | make | model | price
----+------+-------+-------
(0 rows)
```

we can also use CASADE DELETE, it will delete other linked records also (but it is a bad practice)

- **Exporting Query Result to CSV file**



```
test=# \copy (SELECT * FROM person LEFT JOIN car ON car.id = person.car_id) TO '/Users/amigoscode/Desktop/results.csv' DELIMITER ',' CSV HEADER;
COPY 3
test=# 
```

Data Types

Numeric Types:-

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**
decimal	**variable**	**user-specified precision,exact**	**up to 131072 digits before the decimal point; up to 16383 digits after the decimal point**
numeric	**variable**	**user-specified precision,exact**	**up to 131072 digits before the decimal point; up to 16383 digits after the decimal point**
real	4 bytes	variable-precision,inexact	6 decimal digits precision
double precision	8 bytes	variable-precision,inexact	15 decimal digits precision
smallserial	2 bytes		
serial	4 bytes		
bigserial	**8 bytes**	**large autoincrementing integer**	**1 to 9223372036854775807**

Monetary Types

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

Character Types

S. No.	Name & Description
1	**character varying(n), varchar(n)** variable-length with limit
2	**character(n), char(n)** fixed-length, blank padded
3	**text** variable unlimited length

Binary Data Types

Name	Storage Size	Description
bytea	1 or 4 bytes plus the actual binary string	variable-length binary string

Date/Time Types

Name	Storage Size	Description	Low Value	High Value
timestamp [(p)] [without time zone]	**8 bytes**	**both date and time (no time zone)**	**4713 BC**	**294276 AD**
TIMESTAMPTZ	8 bytes	both date and time, with time zone	4713 BC	294276 AD
date	**4 bytes**	**date (no time of day)**	**4713 BC**	**5874897 AD**
time [(p)] [without time zone]	**8 bytes**	**time of day (no date)**	**00:00:00**	**24:00:00**
time [(p)] with time zone	**12 bytes**	**times of day only, with time zone**	**00:00:00+1459**	**24:00:00-1459**
interval [fields] [(p)]	**12 bytes**	**time interval**	**-178000000 years**	**178000000 years**

Boolean Type

Name	Storage Size	Description
boolean	1 byte	state of true or false

Enumerated Type

```
CREATE TYPE week AS ENUM ('Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun');
```

Geometric Type

Name	Storage Size	Representation	Description
point	**16 bytes**	**Point on a plane**	**(x,y)**
line	32 bytes	Infinite line (not fully implemented)	((x1,y1),(x2,y2))
lseg	32 bytes	Finite line segment	((x1,y1),(x2,y2))
box	32 bytes	Rectangular box	((x1,y1),(x2,y2))
path	16+16n bytes	Closed path (similar to polygon)	((x1,y1),...)

path	16+16n bytes	Open path	[(x1,y1),...]
polygon	40+16n	Polygon (similar to closed path)	((x1,y1),...)
circle	24 bytes	Circle	<(x,y),r> (center point and radius)

Network Address Type

Name	Storage Size	Description
cidr	7 or 19 bytes	IPv4 and IPv6 networks
inet	**7 or 19 bytes**	**IPv4 and IPv6 hosts and networks**
macaddr	**6 bytes**	**MAC addresses**

UUID Type

A UUID (Universally Unique Identifiers) is written as a sequence of lower-case hexadecimal digits,

An example of a UUID is − 550e8400-e29b-41d4-a716-446655440000

Array Type

Declaration of Arrays

```
CREATE TABLE monthly_savings (
   name text,
   saving_per_quarter integer ARRAY[4],
   scheme text[][]
);
```

Inserting values

```
INSERT INTO monthly_savings
VALUES (‘Manisha’,
‘{20000, 14600, 23500, 13250}’,
‘{{“FD”, “MF”}, {“FD”, “Property”}}’);
```

Accessing Arrays

```
SELECT name FROM monhly_savings WHERE saving_per_quarter[2] >
saving_per_quarter[4];
```

Modifying Arrays

```
UPDATE monthly_savings SET saving_per_quarter = '{25000,25000,27000,27000}'
WHERE name = 'Manisha';
```

or using the ARRAY expression syntax −

```
UPDATE monthly_savings SET saving_per_quarter = ARRAY[25000,25000,27000,27000]
WHERE name = 'Manisha';
```

Searching Arrays

If Size of Array is known:

```
SELECT * FROM monthly_savings WHERE saving_per_quarter[1] = 10000 OR
saving_per_quarter[2] = 10000 OR
saving_per_quarter[3] = 10000 OR
saving_per_quarter[4] = 10000;
```

If Size of Array is **not known**:

```
SELECT * FROM monthly_savings WHERE 10000 = ANY (saving_per_quarter);
```

Composite Types

Declaration of Composite Types

```
CREATE TYPE inventory_item AS (
   name text,
   supplier_id integer,
   price numeric
);
```

Using:

```
CREATE TABLE on_hand (
   item inventory_item,
   count integer
);
```

Composite Value Input

```
INSERT INTO on_hand VALUES (ROW('fuzzy dice', 42, 1.99), 1000);
```

Accessing Composite Types

```
SELECT (item).name FROM on_hand WHERE (item).price > 9.99;
```

SELECT (on_hand.item).name FROM on_hand WHERE (on_hand.item).price > 9.99;