Εργασία DB1 2020-2021

ΟΝΟΜΑΤΕΠΩΝΥΜΟ	ΑΡΙΘΜΟΣ ΜΗΤΡΩΟΥ
ΠΑΝΑΓΙΩΤΟΠΟΥΛΟΣ ΔΗΜΗΤΡΙΟΣ	Π19130
ΑΥΓΕΡΙΝΟΣ ΧΡΗΣΤΟΣ	Π19020
ΑΘΑΝΑΣΙΟΣ ΒΙΤΑΚΗΣ	Π19247

Contents

Ερώτημα 1

CreateTables

```
create table vehicle
(
   vin_number       varchar(17) not null,
   vcategory        varchar(25) not null,
   car_registration varchar(8)  not null,
   constructor      varchar     not null,
   model            varchar     not null,
   color            varchar     not null,
   release_date     integer     not null,
   price_of_car     numeric     not null,
            primary key (vin_number)
);

create table drivers
(
   drivers_licence_number integer     not null,
   name                   varchar(40) not null,
   gender                 varchar(10) not null,
   birthday               date        not null,
   street                 varchar(40) not null,
   addr_number            integer     not null,
   postal_code            integer     not null,
   city                   varchar(40) not null,
   country                varchar(30) not null,
            primary key (drivers_licence_number)
);
```

```
create table violations
(
        violation_id   integer     not null,
   violation_code varchar(10) not null,
   violation_date date       not null,
   violation_time time       not null,
   description    varchar     not null,
        primary key(violation_id)
);

create table customers
(
   customers_licence_number integer     not null,
   name                     varchar(40) not null,
   gender                   varchar(10) not null,
   birthday                 date        not null,
   street                   varchar(40) not null,
   addr_number              integer     not null,
   postal_code              integer     not null,
   city                     varchar(40) not null,
   country                  varchar(30) not null,
   phonenumber              varchar(10) not null,
   homenumber               varchar(10),
   email                    varchar(40) not null,
        primary key (customers_licence_number)
);

create table insurance_contracts
```

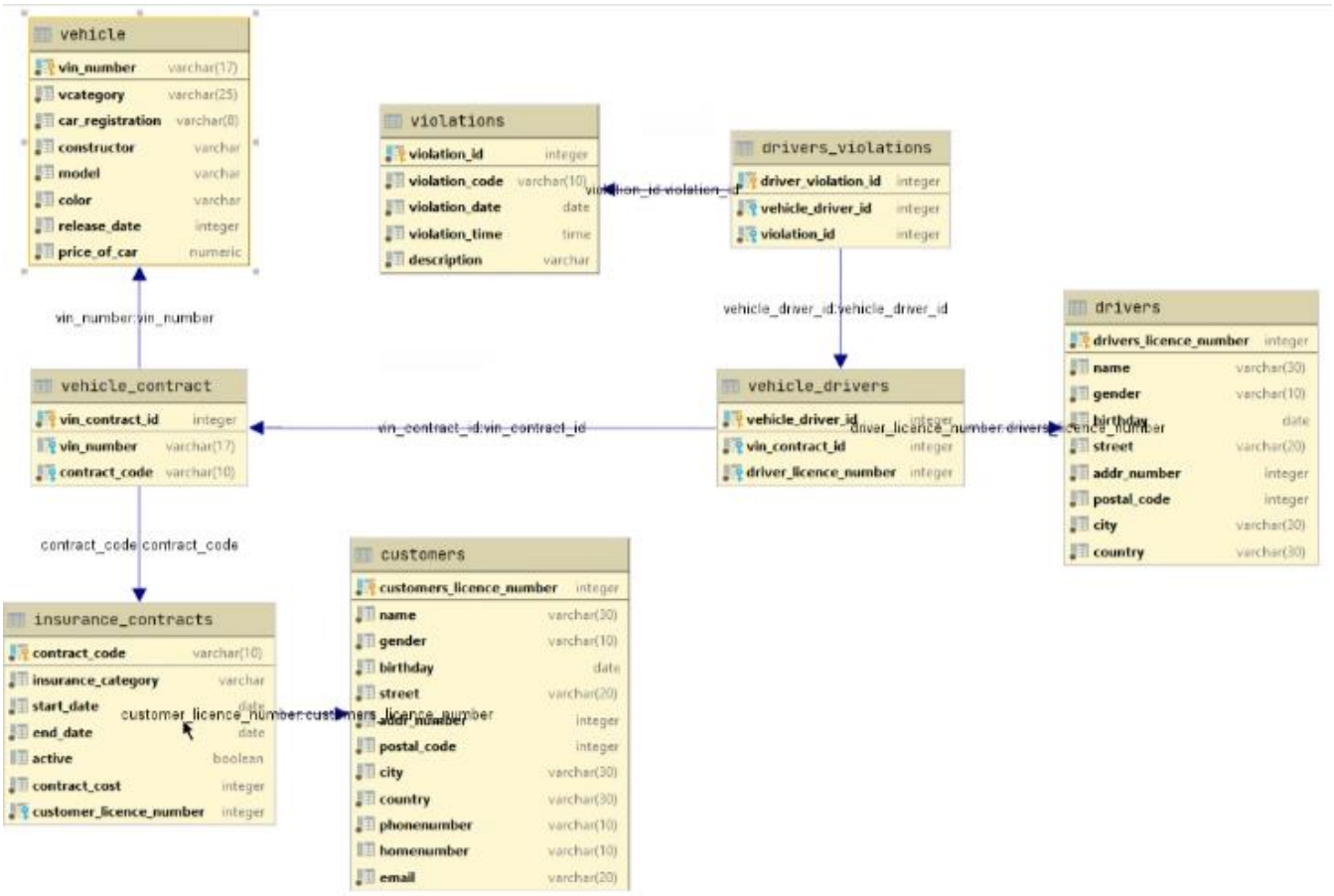
```
(
   contract_code          varchar(10) not null,
   insurance_category     varchar     not null check( insurance_category = 'Private' or insurance_category
= 'Mixed' or insurance_category = 'Professional'),
   start_date             date        not null,
   end_date               date        not null,
   active                 boolean,
   contract_cost          integer     not null,
   customer_licence_number integer     not null,
           primary key(contract_code),
           foreign key (customer_licence_number) references customers
);

create table vehicle_contract
(
   vin_contract_id serial      not null,
   vin_number      varchar(17) not null  UNIQUE,
   contract_code   varchar(10) not null  UNIQUE,
           primary key(vin_contract_id),
           foreign key (vin_number) references vehicle,
           foreign key (contract_code) references insurance_contracts
);

create table vehicle_drivers
(
   vehicle_driver_id     serial  not null,
   vin_contract_id       integer not null,
   driver_licence_number integer not null,
           primary key(vehicle_driver_id),
```

```
        foreign key (vin_contract_id) references vehicle_contract,
        foreign key (driver_licence_number) references drivers
);

create table drivers_violations
(
   driver_violation_id serial  not null,
   vehicle_driver_id   integer not null,
   violation_id        integer not null,
        primary key(driver_violation_id),
        foreign key (vehicle_driver_id) references vehicle_drivers,
        foreign key (violation_id) references violations
);
```



Κανονικοποίηση

Πίνακας Vehicle(vin_number,vcategory,car_registration,constructor,model,color,release_date,price_of_car)

K1 = vin_number K2 = car_registration

Fd1: vin_number → vcategory,car_registration,constructor,model,color,release_date,price_of_car

Fd2: car_registration → vin_number,vcategory,constructor,model,color,release_date,price_of_car

Είναι BCNF

Πίνακας insurance_contracts(contract_code,insurance_category,start_date,end_date,active,contract_cost,customer_license_number)

K1 = contract_code K2 = customer_licence_number

Fd1:contract_code→insurance_category,start_date,end_date,active,contract_cost,customer_license_number

Fd2:customer_licence_number→contract_code,insurance_category,start_date,end_date,active,contract_cost

Είναι BCNF

Πίνακας customers(customers_licence_number,name,gender,birthday,street,address_number,postal_code,city,country,phonenumber,homenumber,email)

K1 = customers_license_number

Fd1: customers_license_number → name,gender,birthday,street,address_number,postal_code,city,country,phonenumber,homenumber,email **Είναι BCNF**

Πίνακας vehicle_contract(vin_contract_id, vin_number,contract_code)

K1 = vin_contract_id , K2 = vin_number , K3 = contract_code

Fd1: vin_contract_id→ vin_number,contract_code

Fd2: vin_number →vin_ contract_id, contract_code

Fd3: contract_code →vin_contract_id,vin_number

Είναι BCNF

Πίνακας drivers(drivers_license_number,name,gender,birthday,street,addr_number,postal_code,city,country)

K1=drivers_license_number

Fd1:drivers_license_number → name,gender,birthday,street,addr_number,postal_code,city,country

Είναι BCNF

Πίνακας vehicle_drivers(vehicle_driver_id, vin_contract_id, driver_license_number)

K1= vehicle_driver_id K2=vin_contract_id,driver_licence_number

Fd1: vehicle_driver_id → contract_id, driver_license_number

Fd2: vin_contract_id,driver_licence_number →vehicle_driver_id

Είναι BCNF

Πίνακας violations(violation_id,violation_code,violation_date,violation_time,description)

K1=violation_id

Fd1:violation_id→ violation_code,violation_date,violation_time,description

Είναι BCNF

Πίνακας drivers_violations(driver_violation_id,vehicle_driver_id,violation_id)

K1=driver_violation_id

Fd1: driver_violation_id → vehicle_driver_id,violation_id

Είναι BCNF

Ερώτημα 2

---A query

select insurance_contracts.contract_code,start_date,end_date,c.name as customer_name,d.name as driver_name

from insurance_contracts

join customers c on insurance_contracts.customer_licence_number= c.customers_licence_number

join vehicle_contract vc on insurance_contracts.contract_code = vc.contract_code

join vehicle_drivers vd on vc.vin_contract_id = vd.vin_contract_id

join drivers d on vd.driver_licence_number = d.drivers_licence_number

where start_date>current_date - interval '1 month' and start_date <= current_date

Query Editor Query History

```
---A querie
select insurance_contracts.contract_code,start_date,end_date,c.name as customer_name,d.name as driver_name
from insurance_contracts
join customers c on insurance_contracts.customer_licence_number= c.customers_licence_number
join vehicle_contract vc on insurance_contracts.contract_code = vc.contract_code
join vehicle_drivers vd on vc.vin_contract_id = vd.vin_contract_id
join drivers d on vd.driver_licence_number = d.drivers_licence_number
where start_date>current_date - interval '1 month' and start_date <= current_date
```

Data Output Explain Messages Notifications

	contract_code character varying (10)	start_date date	end_date date	customer_name character varying (40)	driver_name character varying (40)
1	KkuxNuyJ4Y	2021-06-01	2023-06-01	Brooks Heineken	Gwynne Heatlie
2	akTf5ECqOl	2021-06-23	2023-06-23	Donella Shelton	Aguste Gallone
3	s6EYSnI8Yw	2021-06-05	2023-06-05	Shaughn Gheorghescu	Cleon Ferenczy
4	MqmymcfWkG	2021-06-02	2023-06-02	Al Rounding	Abbie Godar
5	6OVGc7NjU7	2021-06-20	2023-06-20	Thelma Maddern	Harley Bosma
6	6OVGc7NjU7	2021-06-20	2023-06-20	Thelma Maddern	Ellswerth Elam

--B query

select insurance_contracts.contract_code,start_date,end_date,c.phonenumber,c.homenumber

from insurance_contracts

join customers c on c.customers_licence_number = insurance_contracts.customer_licence_number

where end_date>=current_date and end_date<=current_date + interval '1 month'

ergasiav2/postgres@PostgreSQL 13

Query Editor | Query History

```
--B querie
select insurance_contracts.contract_code,start_date,end_date,c.phonenumber,c.homenumber
from insurance_contracts
join customers c on c.customers_licence_number = insurance_contracts.customer_licence_number
where end_date>=current_date and end_date<=current_date + interval '1 month'
```

Data Output | Explain | Messages | Notifications

	contract_code character varying (10)	start_date date	end_date date	phonenumber character varying (10)	homenumber character varying (10)
1	MyUvUMWX4R	2019-06-28	2021-06-28	9394917993	[null]
2	vO89QosNdd	2019-07-10	2021-07-10	7285535265	[null]
3	FnkTVv9kd8	2019-07-16	2021-07-16	6799355602	[null]

--C query

select count(contract_code),insurance_category,date_part('year',start_date) as year

from insurance_contracts

group by insurance_category,year

order by year asc , insurance_category desc

ergasiav2/postgres@PostgreSQL 13

Query Editor Query History

```
--C querie
select count(contract_code),insurance_category,date_part('year',start_date) as year
from insurance_contracts
group by insurance_category,year
order by year asc , insurance_category desc
```

Data Output Explain Messages Notifications

	count bigint	insurance_category character varying	year double precision
1	15	Professional	2018
2	17	Private	2018
3	22	Mixed	2018
4	15	Professional	2019
5	14	Private	2019
6	14	Mixed	2019
7	11	Professional	2020
8	22	Private	2020
9	21	Mixed	2020
10	14	Professional	2021
11	20	Private	2021
12	15	Mixed	2021

--2nd option

select count(contract_code),insurance_category,date_part('year',start_date) as start_year,date_part('year',end_date) as end_year

from insurance_contracts

where active=false

group by insurance_category,start_year,end_year

order by start_year asc , insurance_category desc

Query Editor Query History

```
--2nd option
select count(contract_code),insurance_category,date_part('year',start_date) as
from insurance_contracts
where active=false
group by insurance_category,start_year,end_year
order by start_year asc , insurance_category desc
```

Data Output Explain Messages Notifications

	count bigint	insurance_category character varying	start_year double precision	end_year double precision
1	14	Professional	2018	2020
2	17	Private	2018	2020
3	22	Mixed	2018	2020
4	10	Professional	2019	2021
5	7	Private	2019	2021
6	5	Mixed	2019	2021

--D query

with sum_contract_cost(value) as

(select sum(contract_cost) , insurance_category from insurance_contracts group by insurance_category)

select *

from sum_contract_cost

where value in (select max(value) from sum_contract_cost)

Query Editor Query History

```
--D querie
with sum_contract_cost(value) as
(select sum(contract_cost) , insurance_category from insurance_contracts group by insurance_categ
select *
from sum_contract_cost
where value in (select max(value) from sum_contract_cost)
```

Data Output Explain Messages Notifications

	value bigint	insurance_category character varying
1	34601	Private

--2nd option

select count(contract_code),insurance_category,date_part('year',start_date) as start_year,date_part('year',end_date) as end_year

from insurance_contracts

where active=false

group by insurance_category,start_year,end_year

order by start_year asc , insurance_category desc

ergaslav2/postgres@PostgreSQL 13

Query Editor | Query History

```
--2nd option
select count(contract_code),insurance_category,date_part('year',start_date) as start_year,date_part(
from insurance_contracts
where active=false
group by insurance_category,start_year,end_year
order by start_year asc , insurance_category desc
```

Data Output | Explain | Messages | Notifications

	count bigint	insurance_category character varying	start_year double precision	end_year double precision
1	14	Professional	2018	2020
2	17	Private	2018	2020
3	22	Mixed	2018	2020
4	10	Professional	2019	2021
5	7	Private	2019	2021
6	5	Mixed	2019	2021

--E query

Εκτελέστε τον κώδικα της συνάρτησης πριν εκτελέσετε το query

```
select (get_oldness_vehicle(0,4) * 100)/count(contract_code) as percentage_0to4,

(get_oldness_vehicle(5,9) * 100)/count(contract_code) as percentage_5to9,

(get_oldness_vehicle(10,19) * 100)/count(contract_code) as percentage_10to19,

(get_oldness_vehicle(20,200) * 100)/count(contract_code) as percentage_20plus

from vehicle_contract
```

--E query function

```
CREATE OR REPLACE FUNCTION get_oldness_vehicle (first_year integer, last_year integer)

RETURNS TABLE(oldness bigint) AS $$

BEGIN

RETURN QUERY

	select count(vin_number) as oldness

	from vehicle

	where (date_part('year', current_date) - release_date) >= first_year and (date_part('year',
current_date) - release_date) <= last_year;

END;$$

LANGUAGE plpgsql;
```

Query Editor Query History

```
--E querie
select (get_oldness_vehicle(0,4) * 100)/count(contract_code) as percentage_0to4,
(get_oldness_vehicle(5,9) * 100)/count(contract_code) as percentage_5to9,
(get_oldness_vehicle(10,19) * 100)/count(contract_code) as percentage_10to19,
(get_oldness_vehicle(20,200) * 100)/count(contract_code) as percentage_20plus
from vehicle_contract
```

Data Output Explain Messages Notifications

	percentage_0to4 bigint	percentage_5to9 bigint	percentage_10to19 bigint	percentage_20plus bigint
1	0	5	47	47

--F query

Εκτελέστε τον κώδικα της συνάρτησης πριν εκτελέσετε το query

```
select (get_oldness_drivers(18,24) * 100)/count(driver_violation_id) as percentage_18to24,
(get_oldness_drivers(25,49) * 100)/count(driver_violation_id) as percentage_24to49,
(get_oldness_drivers(50,69) * 100)/count(driver_violation_id) as percentage_50to69,
(get_oldness_drivers(70,120) * 100)/count(driver_violation_id) as percentage_70plus
from drivers_violations
```

--F query function

```
CREATE OR REPLACE FUNCTION get_oldness_drivers (first_year integer, last_year integer)
RETURNS TABLE(oldness bigint) AS $$
BEGIN
RETURN QUERY
    select count(drivers_licence_number) as oldness
    from drivers
    where (date_part('year', current_date) - (date_part('year', birthday))) >= first_year and
(date_part('year', current_date) - (date_part('year', birthday))) <= last_year;
END;$$
LANGUAGE plpgsql;
```

Query Editor Query History

```
--F querie
select (get_oldness_drivers(18,24) * 100)/count(driver_violation_id) as percentage_18to24,
(get_oldness_drivers(25,49) * 100)/count(driver_violation_id) as percentage_24to49,
(get_oldness_drivers(50,69) * 100)/count(driver_violation_id) as percentage_50to69,
(get_oldness_drivers(70,120) * 100)/count(driver_violation_id) as percentage_70plus
from drivers_violations
```

Data Output Explain Messages Notifications

	percentage_18to24 bigint	percentage_24to49 bigint	percentage_50to69 bigint	percentage_70plus bigint
1	70	29	0	0

Ερώτημα 3

--3.a triggers

```
CREATE OR REPLACE FUNCTION renew_insurance_Contracts () RETURNS trigger

LANGUAGE   plpgsql AS $$

BEGIN


                update insurance_contracts

                set end_date = end_date + interval '1 year',

                        active = TRUE

                where active = FALSE and insurance_category = 'Professional' and current_date =
end_date;

        RETURN NEW;

END $$


CREATE TRIGGER renew_insurance_Contracts

AFTER update on insurance_contracts

FOR EACH ROW EXECUTE PROCEDURE renew_insurance_Contracts();
```

ΠΡΙΝ ΤΟ UPDATE

Data Output Explain Messages Notifications

	contract_code character varying (10)	insurance_category character varying	start_date date	end_date date	active boolean	contract_cost integer	customer_licence_num integer
1	AL7SoJnmpp	Professional	2018-02-28	2021-06-25	false	432	
2	hFF98750x5	Professional	2016-10-02	2021-06-25	false	656	

META TO UPDATE

update insurance_contracts

set active = false --*Το βάλαμε false για τις ανάγκες του update*

where end_date = current_date

	contract_code character varying (10)	insurance_category character varying	start_date date	end_date date	active boolean	contract_cost integer	customer_licence_number integer
1	AL7SoJnmpp	Professional	2018-02-28	2022-06-25	true	432	7406
2	hFF98750x5	Professional	2016-10-02	2022-06-25	true	656	5945

--3.b cursors

```
CREATE OR REPLACE FUNCTION contracts_about_to_expire (cur_date DATE)
  RETURNS TABLE (
        contractCode TEXT,
        startDate DATE,
        endDate DATE,
        phone_number TEXT,
        home_number TEXT
```

```
        )  AS $$
   DECLARE
    rec_contracts RECORD;
    cur_contract CURSOR(cur_date DATE) FOR select
insurance_contracts.contract_code,start_date,end_date,c.phonenumber,c.homenumber
                                                                    from insurance_contracts
                                                                    join customers c on
c.customers_licence_number = insurance_contracts.customer_licence_number
                                                                    where end_date>=cur_date and
end_date<=cur_date + interval '1 month';
        BEGIN
        -- Open the cursor
        OPEN cur_contract(cur_date);
        LOOP
        -- fetch row
            FETCH cur_contract INTO rec_contracts;
            -- exit when no more row to fetch
            EXIT WHEN NOT FOUND;
            -- build the output
                contractCode := rec_contracts.contract_code ;
                    startDate := rec_contracts.start_date ;
                        endDate := rec_contracts.end_date ;
                        phone_number := rec_contracts.phonenumber ;
                        home_number := rec_contracts.homenumber ;
                        RETURN NEXT;
        END LOOP;
        -- Close the cursor
        CLOSE cur_contract;
       END; $$
LANGUAGE plpgsql;
```

Query Editor Query History

```
1  SELECT * FROM contracts_about_to_expire (current_date);
```

Data Output Explain Messages Notifications

	contractcode text	startdate date	enddate date	phone_number text	home_number text
1	MyUvUMWX4R	2019-06-28	2021-06-28	9394917993	[null]
2	vO89QosNdd	2019-07-10	2021-07-10	7285535265	[null]
3	FnkTVv9kd8	2019-07-16	2021-07-16	6799355602	[null]

Ερώτημα 4

Παράδειγμα λειτουργίας

```
Please type your own postgreSQL Database name to connect: [redacted]
Please type your own postgreSQL password to connect: [redacted]
---A querie---
 contract_code |     start_date       |        end_date       |       customer_name       |   driver_name

('KkuxNuyJ4Y', datetime.date(2021, 6, 1), datetime.date(2023, 6, 1), 'Brooks Heineken', 'Gwynne Heatlie')
('akTf5ECqOl', datetime.date(2021, 6, 23), datetime.date(2023, 6, 23), 'Donella Shelton', 'Aguste Gallone')
('s6EYSnI8Yw', datetime.date(2021, 6, 5), datetime.date(2023, 6, 5), 'Shaughn Gheorghescu', 'Cleon Ferenczy')
('MqmymcfWkG', datetime.date(2021, 6, 2), datetime.date(2023, 6, 2), 'Al Rounding', 'Abbie Godar')
('6OVGc7NjU7', datetime.date(2021, 6, 20), datetime.date(2023, 6, 20), 'Thelma Maddern', 'Harley Bosma')
('6OVGc7NjU7', datetime.date(2021, 6, 20), datetime.date(2023, 6, 20), 'Thelma Maddern', 'Ellswerth Elam')

---B querie---
 contract_code |     start_date       |        end_date       |       phonenumber      |   homenumber

('MyUvUMWX4R', datetime.date(2019, 6, 28), datetime.date(2021, 6, 28), '9394917993', None)
('vO89QosNdd', datetime.date(2019, 7, 10), datetime.date(2021, 7, 10), '7285535265', None)
('FnkTVv9kd8', datetime.date(2019, 7, 16), datetime.date(2021, 7, 16), '6799355602', None)

---C querie---
 count | insurance_category | year

(1, 'Professional', 2016.0)
(15, 'Professional', 2018.0)
(17, 'Private', 2018.0)
(22, 'Mixed', 2018.0)
(15, 'Professional', 2019.0)
(14, 'Private', 2019.0)
(14, 'Mixed', 2019.0)
(10, 'Professional', 2020.0)
(22, 'Private', 2020.0)
(21, 'Mixed', 2020.0)
(14, 'Professional', 2021.0)
(20, 'Private', 2021.0)
(15, 'Mixed', 2021.0)

---D querie---
 value | insurance_category

(34601, 'Private')

---E querie---
 0-4% | 5-9% | 10-19% | 20plus%

(0, 5, 47, 47)

---F querie---
 18-24% | 24-49% | 50-69% | 70plus%

(70, 29, 0, 0)
PostgreSQL connection is closed

>>>
```

Κώδικας

```
import psycopg2




try:

    dbn=input("Please type your own postgreSQL Database name to connect: ")

    psw=input("Please type your own postgreSQL password to connect: ")

    con = psycopg2.connect(dbname = dbn, host = 'localhost', port = '5432', user = 'postgres', password =
psw)

    cur = con.cursor()



    #The execute routine executes an SQL statement.

#------------------------------------------------------ A QUERY ------------------------------------------------------------------#

    cur.execute('''select insurance_contracts.contract_code,start_date,end_date,c.name as
customer_name,d.name as driver_name

from insurance_contracts

join customers c on insurance_contracts.customer_licence_number= c.customers_licence_number

join vehicle_contract vc on insurance_contracts.contract_code = vc.contract_code

join vehicle_drivers vd on vc.vin_contract_id = vd.vin_contract_id

join drivers d on vd.driver_licence_number = d.drivers_licence_number

where start_date>current_date - interval '1 month' and start_date <= current_date ''')


    #The fetchall routine fetches all (remaining) rows of a query result, returning a list.

    #An empty list is returned when no rows are available.

    records = cur.fetchall()

    print("---A querie---")

    print(" contract_code |     start_date      |      end_date      |      customer_name      |   driver_name
\n")
```

```
    for row in records:
        print(row)


#------------------------------------------------- B QUERY ------------------------------------------------------------#
    cur.execute('''select
insurance_contracts.contract_code,start_date,end_date,c.phonenumber,c.homenumber


from insurance_contracts
join customers c on c.customers_licence_number = insurance_contracts.customer_licence_number
where end_date>=current_date and end_date<=current_date + interval '1 month' ''')
    records = cur.fetchall()
    print("\n---B querie---")
    print(" contract_code |     start_date      |       end_date      |       phonenumber      |   homenumber
\n")
    for row in records:
        print(row)


#------------------------------------------------- C QUERY ------------------------------------------------------------#
    cur.execute('''select count(contract_code),insurance_category,date_part('year',start_date) as year
from insurance_contracts
group by insurance_category,year
order by year asc , insurance_category desc''')
    records = cur.fetchall()
    print("\n---C querie---")
    print(" count | insurance_category | year     \n")
    for row in records:
        print(row)
```

```
#------------------------------------------------- D QUERY ------------------------------------------------------------#
    cur.execute('''with sum_contract_cost(value) as
(select sum(contract_cost) , insurance_category from insurance_contracts group by insurance_category)
select *
from sum_contract_cost
where value in (select max(value) from sum_contract_cost)
''')
    records = cur.fetchall()
    print("\n---D querie---")
    print(" value | insurance_category    \n")
    for row in records:
        print(row)

#------------------------------------------------- E QUERY ------------------------------------------------------------#
    cur.execute('''select (get_oldness_vehicle(0,4) * 100)/count(contract_code) as percentage_0to4,
(get_oldness_vehicle(5,9) * 100)/count(contract_code) as percentage_5to9,
(get_oldness_vehicle(10,19) * 100)/count(contract_code) as percentage_10to19,
(get_oldness_vehicle(20,200) * 100)/count(contract_code) as percentage_20plus
from vehicle_contract
''')
    records = cur.fetchall()
    print("\n---E querie---")
    print(" 0-4% | 5-9% | 10-19% | 20plus%    \n")
    for row in records:
        print(row)

#------------------------------------------------- F QUERY ------------------------------------------------------------#
```

```
    cur.execute('''select (get_oldness_drivers(18,24) * 100)/count(driver_violation_id) as
percentage_18to24,
(get_oldness_drivers(25,49) * 100)/count(driver_violation_id) as percentage_24to49,
(get_oldness_drivers(50,69) * 100)/count(driver_violation_id) as percentage_50to69,
(get_oldness_drivers(70,120) * 100)/count(driver_violation_id) as percentage_70plus
from drivers_violations''')
    records = cur.fetchall()
    print("\n---F querie---")
    print(" 18-24% | 24-49% | 50-69% | 70plus%    \n")
    for row in records:
        print(row)



#The except block lets you handle the error.
except(Exception, psycopg2.Error) as error:
    print("Error while fetching data from PostgreSQL", error)

#The finally block lets you execute code, regardless of the result of the try- and except blocks.
finally:
    if(con):
        cur.close()
        con.close()
        print("PostgreSQL connection is closed\n")
```