ARDBMS MINI PROJECT DOCUMENTATION

PROJECT TITLE: CAR PARKING MANAGEMENT SYSTEM

TEAM MEMBERS:

CAR PARKING

MANAGEMENT

SYSTEM

1) NAME: SHREYA KARDILE

ENROLLMENT NUMBER:2003457

2) NAME: MANN MERWANA

ENROLLMENT NUMBER:2000955

3) NAME: ANKITA SUTAR

ENROLLMENT NUMBER:2001440

Project Guide Name: MRS SHILPA DANGE

Project Guide Signature:

PROBLEM DESCRIPTION: BEFORE THE MODERNIZED AND DIGITALIZED CAR PARKING MANAGMENT SYSTEM, WE HAD TO TACKLE MANY PROBLEMS SUCH AS MANPOWER, NO SYSTEM FOR STORAGE OF DETAILS OF THE CAR AND THE OWNER, UNORGANIZED MANGEMENT, ETC. BUT DUE TO TECHNOLOGY AND DIGITALIZED SYSTEM, ALL THE PROBLEMS ARE TACKLED.

STUDY OF EXISITNG SYSTEM: OUR EXISITNG CAR MANAGMENT SYSTEM IS COMPLETELY COMPUTERIZED. WITH THE HELP OF NFC BASED INFRA RED SCANNERS, THE NUMBER PLATE OF THE CAR CAN BE SCANED AND ALL ITS DETAILS ALONG WITH OWNER ID AND BANK DETAILS IS STORED IN THE DATABASE. THE CCTV CAMERAS IN THE PARKING AREA IS A COMBO, AS IT RECORDS THE TIME FOR WHICH THE CAR WAS PARKED AND ALSO MAINTAINS THE SAFETY OF THE CAR.

DRAWBACKS OF THE EXISTING SYSTEM:

1) ONLY DIGITAL PAYMENTS ARE BEING ACCEPTED.

2) WE DON'T HAVE ENOUGH SPACE FOR BULKIER CAR TYPES.

3) IF THE PARKING SLOT IS OCCUPIED AND A CAR OWNER WANTS TO PARK THE CAR IN OUR AREA, WE DON'T HAVE AN ALTERNATIVE FOR THIS SITUATION.

SCOPE OF THE PROPOSED SYSTEM:

IN THE PROPOSED SYSTEM, NO MANPOWER IS REQUIRED AS IT WAS A MUST IN EARLIER SYSTEMS. THE PAYMENTS ACCEPTED WERE ONLY CASH, SO THAT ALSO HAVE BEEN MODIFIED. NOW, ONLY DIGITAL PAYMENTS ARE ACCEPTED.THERE IS AN ORAGANIZED SYSTEM FOR THE MANAGEMENT OF CARS.

WITH THE HELP OF NEW TECHNOLOGIES SUCH AS INFRA RED SCANNERS BASED ON THE NEAR FIELD COMMUNICATION TECHNIQUE, IT IS EASIER TO SCAN THE DETAILS OF THE CAR, BANK DETAILS, OWNER ID AUTOMATICALLY AND IT COULD BE STORED DIRECTLY TO THE DATABASE.

AND THE COMPUTER GENERATED INVOICE IS DIRECTLY SENT TO THE CONTACT NUMBER OF THE OWNER VIA OWNER ID, AND THE FEES IS AUTOMATICALLY DEDUCTED FROM THE OWNER’S BANK ACCOUNT. THIS ENSURES HASSEL FREE TRANSACTION OF THE PAYMENT. SO, IN THIS WAY, THE INVOICE IS SENT TO OWNER’S CONTACT NUMBER.

IDENTIFY THE END USERS:

THE CAR PARKING MANAGEMENT SYSTEM HAS THE STAFF AS THE MAIN END USERS ,AS THEY HAVE ACCESS TO THE DATABASE WHERE ALL THE DETAILS OF THE OWNER ,CAR , STATUS OF THE PAYMENT CAN BE FETCHED. THE CAR OWNERS ARE NOT THE END USERS AS THEY PLAY NO ROLE IN THE MANAGEMENT .

INPUT DATA OF THE SYSTEM:

INTO THE SYSTEM THE DETAILS OF THE CAR ,OWNER ID AND ,AMOUNT PAID i.e FEES IS BEING INPUTTED WITH THE HELP OF CCTV CAMERAS AND THE INRA RED SCANNERS .

OUTPUT DATA OF THE SYSTEM:

THE DETAILS OF CAR ,OWNER ID AND TIME IS BEING DISPLAYED IN THE COMPUTER GENERATED INVOICES ALONG WITH THE AMOUNT PAID i.e FEES .

INVOICE:

|  |
| --- |
| ABC CAR PARKING  RAJ TOWERS,AUNDH,PUNE.   * OWNER ID: * CAR TYPE: * CAR ID: * CAR NUMBER: * AMOUNT PAID:   TIME: 12:00 HRS  THANK YOU, VISIT AGAIN. |

ENTITY RELATIONSHIP MODELING:

ENTITY RELATIONSHIP MODELING:

1. ENTITY NAME: car

ATTRIBUTES: (car\_id int primary key, user\_id int, car\_no int, car\_type varchar(33) )

1. ENTITY NAME: parking

ATTRIBUTES: (p\_id int primary key, p\_car\_id int , p\_fees float)

1. ENITITY NAME: parking\_slot

ATTRIBUTES: (p\_slot\_id int primary key,p\_slot\_c\_id int ,p\_block varchar(30),car\_arrival\_time time,car\_dept\_time time)

1. ENTITY NAME: fees

ATTRIBUTES: (f\_id int unique primary key,f\_amount money ,f\_type varchar(30),f\_desc varchar (20))

1. ENITITY NAME: car\_owner

ATTRIBUTES: ( owner\_id int primary key,owner\_name varchar(30),owner\_no int,owner\_address varchar(33))

1. ENITITY NAME: car\_parking\_details

ATTRIBUTES: (car\_id int references car(car\_id) on delete cascade on update cascade,p\_id int references parking(p\_id) on delete cascade on update cascade ,p\_slot\_id int references parking\_slot(p\_slot\_id) on delete cascade on update cascade,f\_id int references fees(f\_id) on delete cascade on update cascade ,owner\_id int references car\_owner(owner\_id) on delete cascade on update cascade)

ENTITY RELATIONSHIP DIAGRAM:

car\_owner

HAS

car

MANAGE

parking\_slot

parking

Has

fees

Designing the Normalized Database:

|  |
| --- |
| TABLE NAME 1 REALTIONSHIP TABLE NAME 2 |
| car ONE TO ONE car owner |
| parking ONE TO MANY car\_parking\_details |
| car ONE TO MANY parking\_slots |
| fees ONE TO MANY parking,parking\_slot |
| car\_parking\_details ONE TO MANY car,car\_owner |
| car MANY TO ONE parking\_slot,car\_owner |
| car\_owner,car MANY TO MANY fees,parking\_slot |

DATABASE AND TABLES:

1 car:

postgres=# \c car\_parking\_system;

You are now connected to database "car\_parking\_system" as user "postgres".

car\_parking\_system=# \d car;

Table "public.car"

Column | Type | Collation | Nullable | Default

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

car\_id | integer | | not null |

user\_id | integer | | |

car\_no | integer | | |

car\_type | character varying(33) | | |

Indexes:

"car\_pkey" PRIMARY KEY, btree (car\_id)

Referenced by:

TABLE "car\_parking\_details" CONSTRAINT "car\_parking\_details\_car\_id\_fkey" FOREIGN KEY (car\_id) REFERENCES car(car\_id) ON UPDATE CASCADE ON DELETE CASCADE

car\_parking\_system=# select \* from car;

car\_id | user\_id | car\_no | car\_type

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

1 | 111 | 4567 | hatchback

2 | 112 | 4568 | sedan

3 | 113 | 4569 | suv

4 | 114 | 4570 | non\_sedan

5 | 115 | 4571 | coupe

6 | 116 | 4572 | crossover

7 | 117 | 4573 | mpv

8 | 118 | 4574 | convertible

(8 rows) (33) | | |

2 parking:

car\_parking\_system=# \d parking;

Table "public.parking"

Column | Type | Collation | Nullable | Default

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

p\_id | integer | | not null |

p\_car\_id | integer | | |

p\_fees | double precision | | |

Indexes:

"parking\_pkey" PRIMARY KEY, btree (p\_id)

Referenced by:

TABLE "car\_parking\_details" CONSTRAINT "car\_parking\_details\_p\_id\_fkey" FOREIGN KEY (p\_id) REFERENCES parking(p\_id) ON UPDATE CASCADE ON DELETE CASCADE

car\_parking\_system=# select \* from parking;

p\_id | p\_car\_id | p\_fees

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

1 | 11 | 50

2 | 12 | 60

3 | 13 | 70

4 | 14 | 80

5 | 15 | 90

6 | 16 | 100

7 | 17 | 110

(7 rows)

3 parking\_slot:

car\_parking\_system=# \d parking\_slot;

Table "public.parking\_slot"

Column | Type | Collation | Nullable | Default

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

p\_slot\_id | integer | | not null |

p\_slot\_c\_id | integer | | |

p\_block | character varying(30) | | |

car\_arrival\_time | time without time zone | | |

car\_dep\_time | time without time zone | | |

Indexes:

"parking\_slot\_pkey" PRIMARY KEY, btree (p\_slot\_id)

Referenced by:

TABLE "car\_parking\_details" CONSTRAINT "car\_parking\_details\_p\_slot\_id\_fkey" FOREIGN KEY (p\_slot\_id) REFERENCES parking\_slot(p\_slot\_id) ON UPDATE CASCADE ON DELETE CASCADE

car\_parking\_system=# select \* from parking\_slot;

p\_slot\_id | p\_slot\_c\_id | p\_block | car\_arrival\_time | car\_dep\_time

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

101 | 1 | A | 12:00:00 | 13:00:00

102 | 2 | B | 13:00:00 | 14:00:00

103 | 3 | C | 14:00:00 | 15:00:00

104 | 4 | D | 15:00:00 | 16:00:00

105 | 5 | E | 16:00:00 | 17:00:00

106 | 6 | F | 17:00:00 | 18:00:00

107 | 7 | G | 18:00:00 | 19:00:00

(7 rows)

4 fees:

car\_parking\_system=# \d fees;

Table "public.fees"

Column | Type | Collation | Nullable | Default

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

f\_id | integer | | not null |

f\_amount | money | | |

f\_type | character varying(30) | | |

f\_desc | character varying(20) | | |

Indexes:

"fees\_pkey" PRIMARY KEY, btree (f\_id)

Referenced by:

TABLE "car\_parking\_details" CONSTRAINT "car\_parking\_details\_f\_id\_fkey" FOREIGN KEY (f\_id) REFERENCES fees(f\_id) ON UPDATE CASCADE ON DELETE CASCADE

car\_parking\_system=# select \* from fees;

f\_id | f\_amount | f\_type | f\_desc

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

10 | ? 50.00 | digital | paid

20 | ? 60.00 | digital | paid

30 | ? 70.00 | digital | unpaid

40 | ? 80.00 | digital | paid

50 | ? 90.00 | digital | unpaid

60 | ? 100.00 | digital | unpaid

70 | ? 110.00 | digital | paid

(7 rows)

5 car\_owner:

car\_parking\_system=# \d car\_owner;

Table "public.car\_owner"

Column | Type | Collation | Nullable | Default

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

owner\_id | integer | | not null |

owner\_name | character varying(30) | | |

owner\_no | integer | | |

owner\_address | character varying(30) | | |

Indexes:

"car\_owner\_pkey" PRIMARY KEY, btree (owner\_id)

Referenced by:

TABLE "car\_parking\_details" CONSTRAINT "car\_parking\_details\_owner\_id\_fkey" FOREIGN KEY (owner\_id) REFERENCES car\_owner(owner\_id) ON UPDATE CASCADE ON DELETE CASCADE

car\_parking\_system=# select \* from car\_owner;

owner\_id | owner\_name | owner\_no | owner\_address

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

100 | shreya | 986053388 | pune

200 | ankita | 986053367 | pune

300 | krishna | 986053347 | mumbai

400 | saloni | 986053332 | pune

500 | rajesh | 986053341 | mumbai

600 | janak | 986053632 | pune

(6 rows)

6 car\_parking\_details:

car\_parking\_system=# \d car\_parking\_details;

Table "public.car\_parking\_details"

Column | Type | Collation | Nullable | Default

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

car\_id | integer | | |

p\_id | integer | | |

p\_slot\_id | integer | | |

f\_id | integer | | |

owner\_id | integer | | |

Foreign-key constraints:

"car\_parking\_details\_car\_id\_fkey" FOREIGN KEY (car\_id) REFERENCES car(car\_id) ON UPDATE CASCADE ON DELETE CASCADE

"car\_parking\_details\_f\_id\_fkey" FOREIGN KEY (f\_id) REFERENCES fees(f\_id) ON UPDATE CASCADE ON DELETE CASCADE

"car\_parking\_details\_owner\_id\_fkey" FOREIGN KEY (owner\_id) REFERENCES car\_owner(owner\_id) ON UPDATE CASCADE ON DELETE CASCADE

"car\_parking\_details\_p\_id\_fkey" FOREIGN KEY (p\_id) REFERENCES parking(p\_id) ON UPDATE CASCADE ON DELETE CASCADE

"car\_parking\_details\_p\_slot\_id\_fkey" FOREIGN KEY (p\_slot\_id) REFERENCES parking\_slot(p\_slot\_id) ON UPDATE CASCADE ON DELETE CASCADE

car\_parking\_system=# select \* from car\_parking\_details;

car\_id | p\_id | p\_slot\_id | f\_id | owner\_id

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

1 | 1 | 101 | 10 | 100

2 | 2 | 102 | 20 | 200

3 | 3 | 103 | 30 | 300

4 | 4 | 104 | 40 | 400

5 | 5 | 105 | 50 | 500

6 | 6 | 106 | 60 | 600

(6 rows)

Designing queries related to Functional requirements

NESTED QUERIES :

1 FIND THE OWNER ADDRESS OF CAR OWNER WHOSE NAME IS SHREYA.

ANSWER:

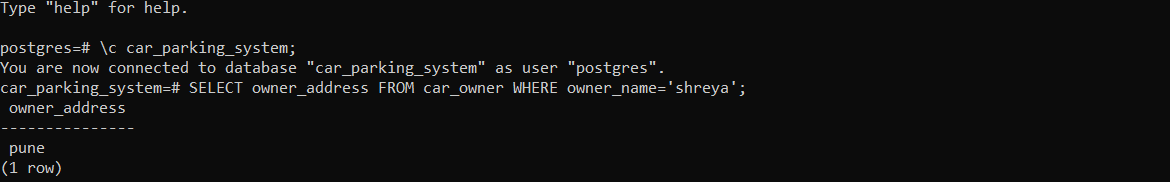
car\_parking\_system=# SELECT owner\_address FROM car\_owner WHERE owner\_name='shreya';

owner\_address

---------------

pune

(1 row)



2 FIND THE MAXIMUM FEES AMOUNT TO BE PAID FOR THE INVOICE.

ANSWER:

car\_parking\_system=# select max(f\_amount) from fees;

max

----------

? 110.00

(1 row)



VIEWS:

1 CREATE A VIEW WHICH DISPLAYS THE DETAILS OF OWNERS

ANSWER:

car\_parking\_system=# CREATE VIEW v1 AS SELECT car\_owner.owner\_id,car\_owner.owner\_name,car\_owner.owner\_no,car\_owner.owner\_address FROM car\_parking\_details,car\_owner WHERE car\_owner.owner\_id=car\_parking\_details.owner\_id;

CREATE VIEW

car\_parking\_system=# select \* from v1;

owner\_id | owner\_name | owner\_no | owner\_address

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

100 | shreya | 986053388 | pune

200 | ankita | 986053367 | pune

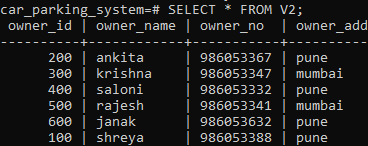
300 | krishna | 986053347 | mumbai

400 | saloni | 986053332 | pune

500 | rajesh | 986053341 | mumbai

600 | janak | 986053632 | pune

(6 rows)



2 CREATE A VIEW WHICH DISPLAYS OWNER DETAILS WHO’S FEES IS $100

ANSWER:

car\_parking\_system=# CREATE VIEW V2 AS SELECT car\_owner.owner\_name FROM car\_parking\_details, parking, car\_owner WHERE car\_owner.owner\_id = car\_parking\_details.owner\_id AND parking.p\_id = car\_parking\_details.p\_id AND p\_fees = 100;

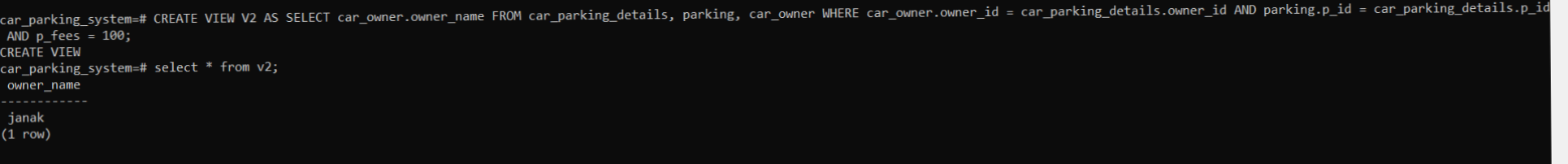
CREATE VIEW

car\_parking\_system=# select \* from v2;

owner\_name

------------

janak

(1 row)

STORED FUNCTIONS:

1. Write a function that returns the total number of cars of a particular address. ( Accept address as input parameter.)

INPUT:

create or replace function cust(nm char(30))returns int as $$

declare cnt int;

begin

select count(car\_id) into cnt from car\_parking\_details where owner\_id in(select owner\_id from car\_owner where owner\_address=nm);

return cnt;

end;

$$ language plpgsql;

OUTPUT:

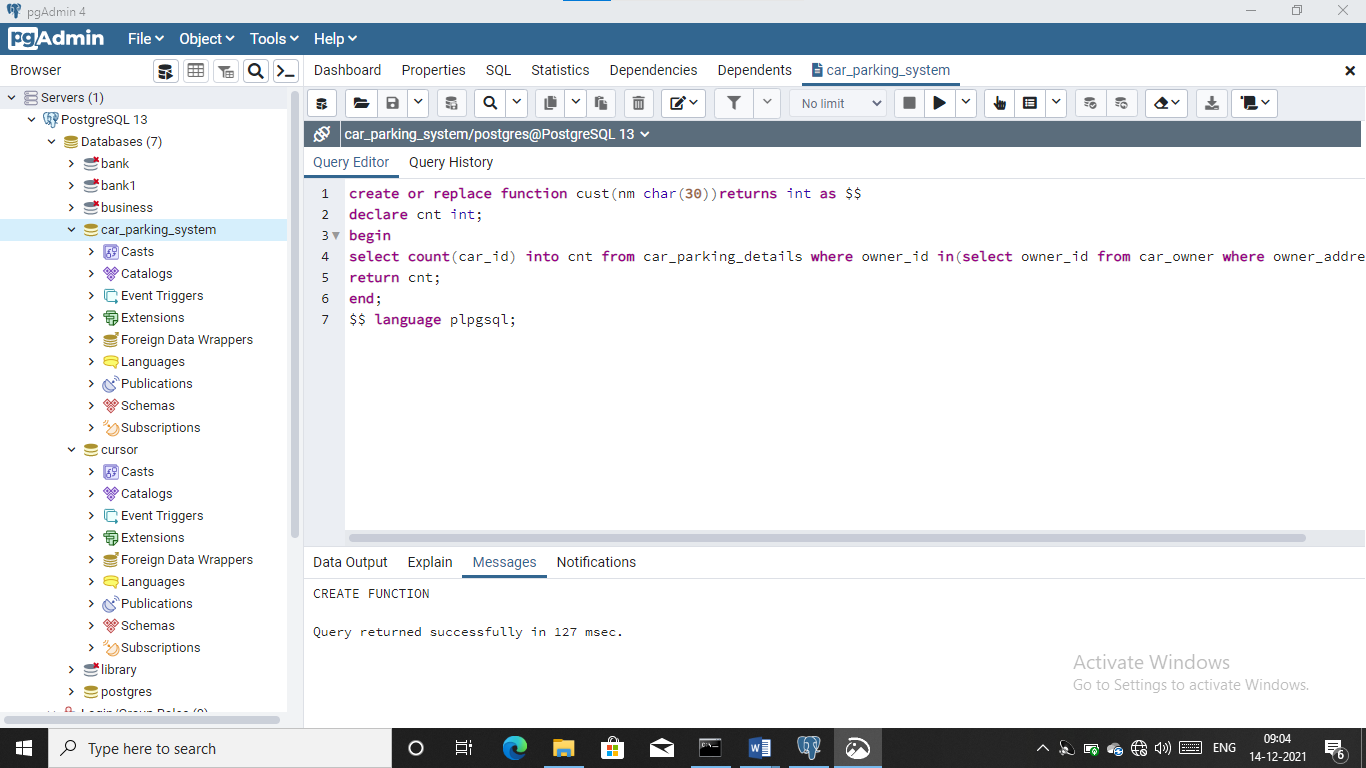
car\_parking\_system=# select cust('pune');

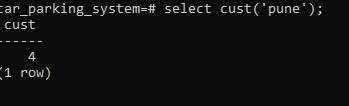
cust

------

4

(1 row)





2 Write a function to display car\_no for particular c\_type ( Accept car type as input parameter).

INPUT:

CREATE OR REPLACE FUNCTION cidno(cno CHARACTER (30)) RETURNS INT AS $$

DECLARE no CHARACTER;

BEGIN

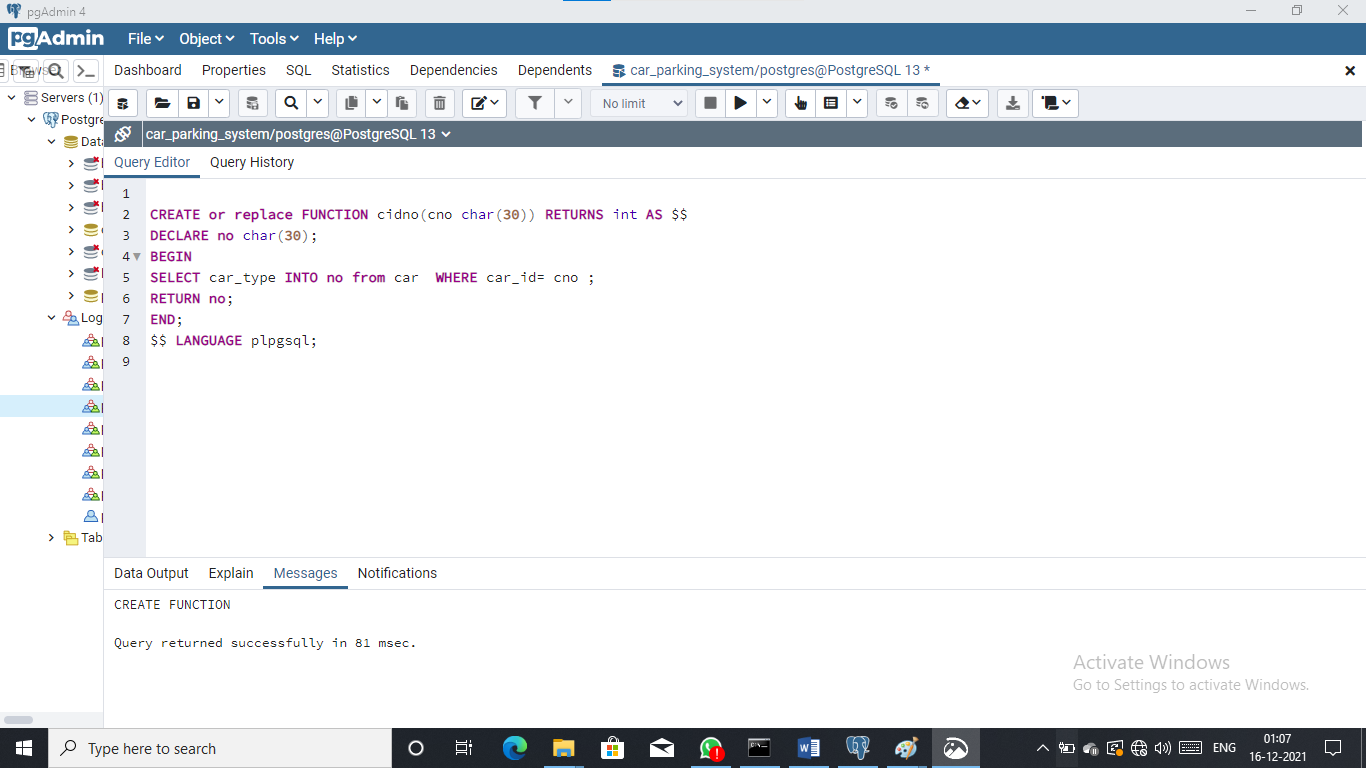
SELECT car INTO no WHERE c\_type = cno;

RETURN no;

END;

$$ LANGUAGE plpgsql;

OUTPUT:





TRIGGERS:

1) Write a trigger before deleting a car record from the car table. Raise a notice and display the message “car record is being deleted”

INPUT:  
CREATE OR REPLACE FUNCTION f3() RETURNS TRIGGER AS $$

BEGIN

RAISE EXCEPTION 'Student Record Is Deleted %',old;

RETURN old;

END

$$ LANGUAGE plpgsql;

OUTPUT:  
car\_parking\_system=# CREATE TRIGGER tt1 BEFORE DELETE ON car FOR EACH ROW EXECUTE PROCEDURE f3();

CREATE TRIGGER

car\_id | user\_id | car\_no | car\_type

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

1 | 111 | 4567 | hatchback

2 | 112 | 4568 | sedan

3 | 113 | 4569 | suv

4 | 114 | 4570 | non\_sedan

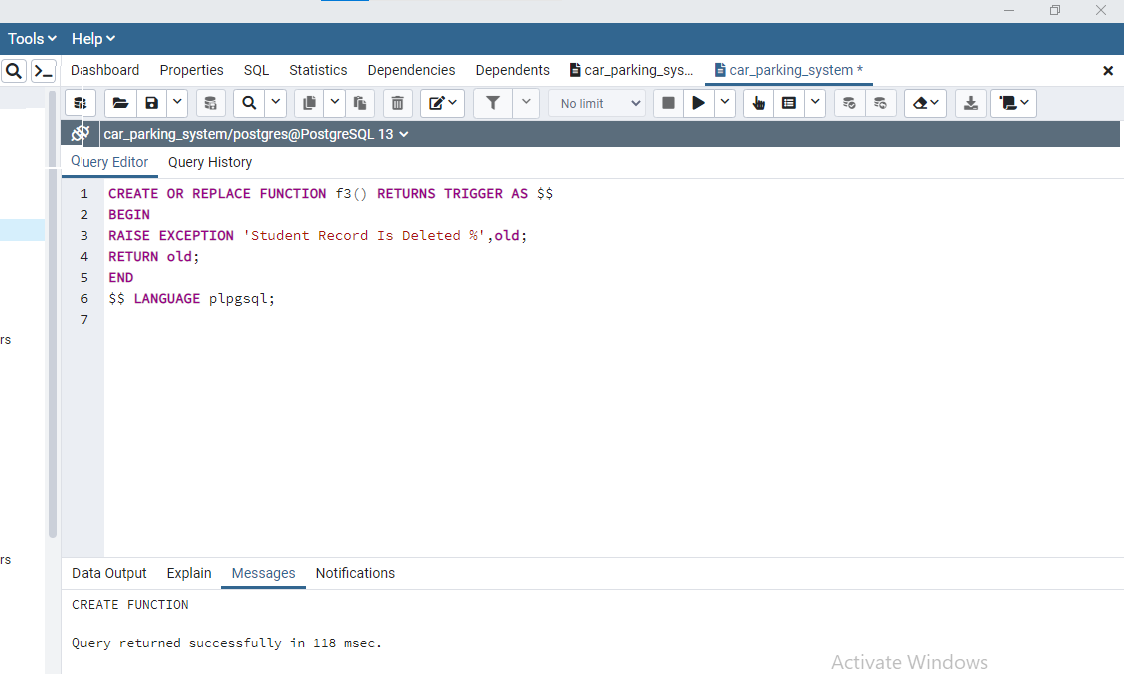
5 | 115 | 4571 | coupe

6 | 116 | 4572 | crossover

7 | 117 | 4573 | mpv

8 | 118 | 4574 | convertible

(8 ROWS)





CURSOR :

1) Write stored function using cursor to find the list of car id's whose address is pune.

INPUT :

CREATE or replace FUNCTION ff1() RETURNS INTEGER AS $$

DECLARE ccr CURSOR FOR SELECT car\_id FROM car, car\_owner, car\_parking\_details WHERE owner\_address = "Pune" AND

car.car\_id = car\_parking\_details.car\_id;

n1 car\_parking\_details.car\_id %type ;

BEGIN

OPEN ccr;

LOOP

FETCH ccr INTO n1;

EXIT WHEN NOT FOUND;

RAISE NOTICE '%',n1;

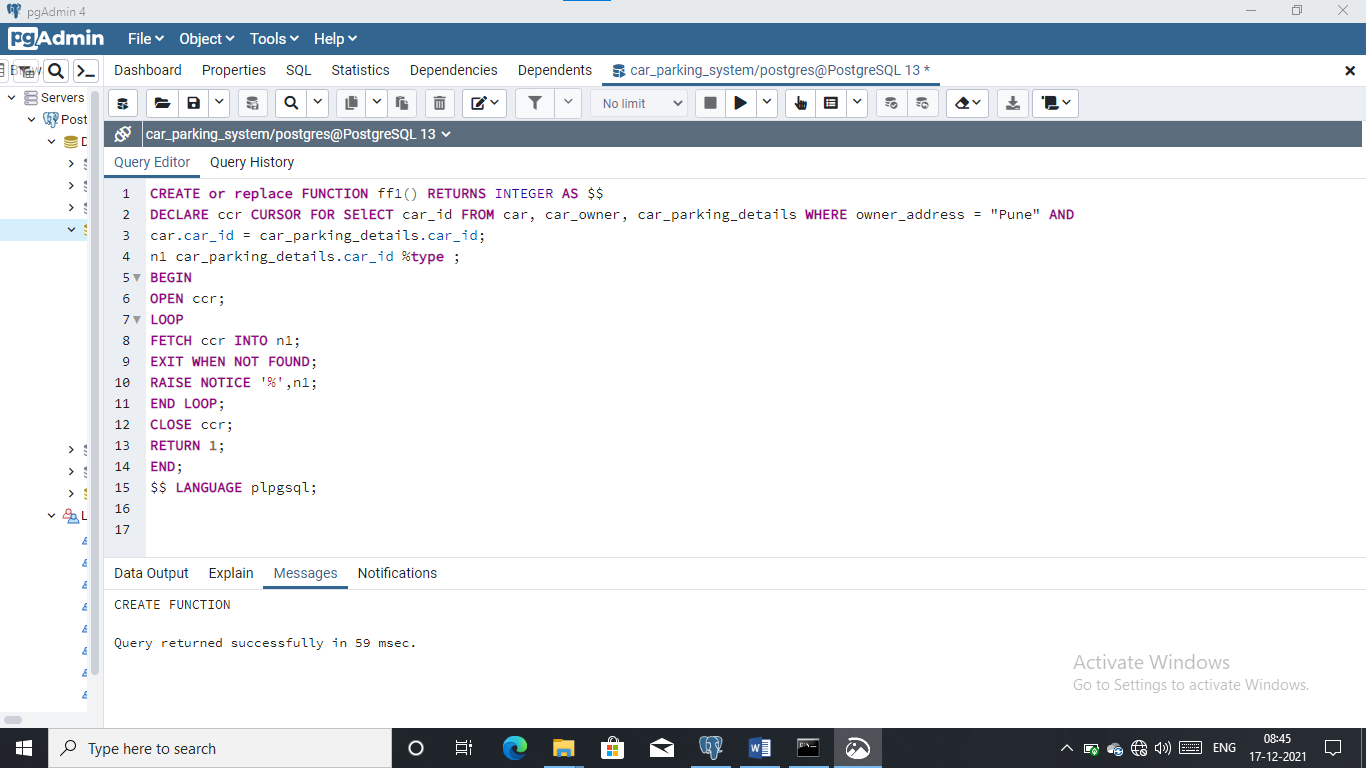
END LOOP;

CLOSE ccr;

RETURN 1;

END;

$$ LANGUAGE plpgsql;



OUTPUT :

car\_parking\_system#= select ff1();

ff1

----------

4

(1 row)

2) Display the data of cars whose fees is unpaid.

INPUT:

CREATE OR REPLACE FUNCTION FF5() RETURNS VOID AS $$

DECLARE cf1 CURSOR FOR SELECT car\_no FROM car WHERE car\_id IN (SELECT f\_id FROM fees, car\_parking\_details WHERE f\_desc = 'unpaid');

caarid INT;

BEGIN

OPEN cf1;

LOOP

FETCH cf1 INTO caarid;

EXIT WHEN NOT FOUND;

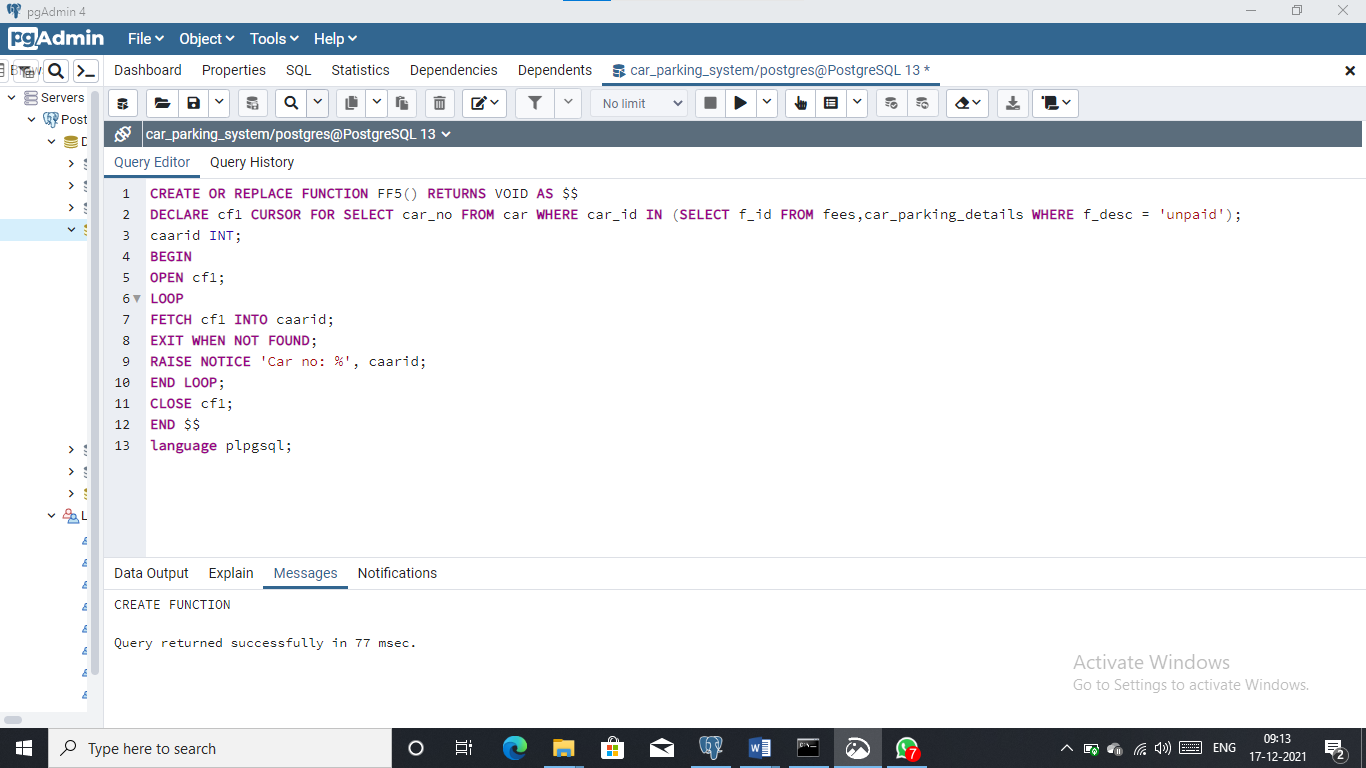
RAISE NOTICE 'Car no: %', caarid;

END LOOP;

CLOSE cf1;

END $$

language plpgsql;



OUTPUT :

car\_parking\_system=# SELECT FF5();

FF5

----------------

4569

4571

4572

(3 row)

BIBLIOGRAPHY:

1. OUR TEACHER’S NOTES
2. SYBSC CS MANUAL
3. <https://www.youtube.com/watch?v=S2DYTxEkGJg>
4. <https://www.pdffiller.com/231181313-2501_ec6epdf-car-parking-system-project-report-doc->
5. <https://phpgurukul.com/vehicle-parking-management-system-using-php-and-mysql/>
6. <https://www.freeprojectz.com/project-source-code-database-download/car-parking-system-project>
7. <https://www.campcodes.com/projects/php/online-parking-management-system-in-php-mysql-free-download/>

THE END

Thank You...!