We declare that we have completed this assignment completely and entirely on our own, without any consultation with others. We have read the UAB Academic Honor Code and understand that any breach of the Honor Code may result in severe penalties.

We also declare that the following percentage distribution ***faithfully*** represents individual group members' contributions to the completion of the assignment

Name	Overall Contribution (%)	Major work items completed by me	Signature or initials	Date
Vikram Reddy Dasari	33.33	Application concept,ER diagram,Sql Table creation,created 2 views,created 1 index,1 non key constraint, 1 trigger, report	V.D	12/06/20
Yellanti Venkat Vivek	33.33	Application concept, explation for Er diagram,Data base normalization into 3nf, created 2 views, created 1 index, non key constraint, Stored procedure,report	Y.V.V	12/06/20
Nalluri Revanth Kumar	33.33	Application concept, assumptions and db constraints,data population, created 2 views, created 1 index, non key constraint, 1 Trigger,report	N.R	12/06/20

Web Username : vicky96web

Password: qwerty

Term Project

Vehicle Dealership Application

A.1.) Application Overview:

This is a application to manage vehicle dealership. Customer can book vehicles from different locations of dealership. Dealership have sales_inventory which stores total available vehicles and total sales made. Vehicles are of two types SUV and Sedan. The price of a booking are defined by the type of vehicle and number of days.

Customer books a vehicle of desired type from any of the location of the dealership. Bookings are done number of days basis. Booking cannot be done for less than a day. Prices of vehicles and deposits are defined by vehicle model and type.

There are 6 entities in the implementation model and 2 ISA entities.

Dealership: Stores the details of a dealership.

Sales_inventory: Stores total_available vehicles and Sales.

Vehicles: This is an entity for vehicles inventory for the dealership and it has two ISA entities for two types of vehicle SUV and Sedan.

Bookings: Stores the details of all the bookings.

Customer: To store the information of customers.

Address: To store the address of customers.

A.2.) E-R diagram of the application

We have 6 classes in our database. The relationships are Dealership, Vehicle, Booking, Sales Inventory, Customer, Address. The relationship between Dealership and vehicle is one to many as a dealership can own many vehicles but a vehicle can only be owned by a single dealership. The relationship between dealership and sales inventory is one to one as a dealer can have one sales inventory and one inventory can be owned by a single dealer. The relationship between customer and address is one to many because a customer can have one address and multiple customers can live at a single address. There is a three-way relationship between vehicle, booking and customer. The relationship between vehicle and customer is one to many as a customer can book multiple vehicles but a vehicle can only booked by one customer at a time. Similarly between vehicle and booking the relationship is one to many as in a single booking made by the customer a customer can select multiple vehicles but a vehicle can only be assigned to one booking at a time.

Dealership Entity dealership_id is primary key, inventory is foreign key refers to id in sales_inventory entity.

Sales_inventory entity id attribute is the primary key.

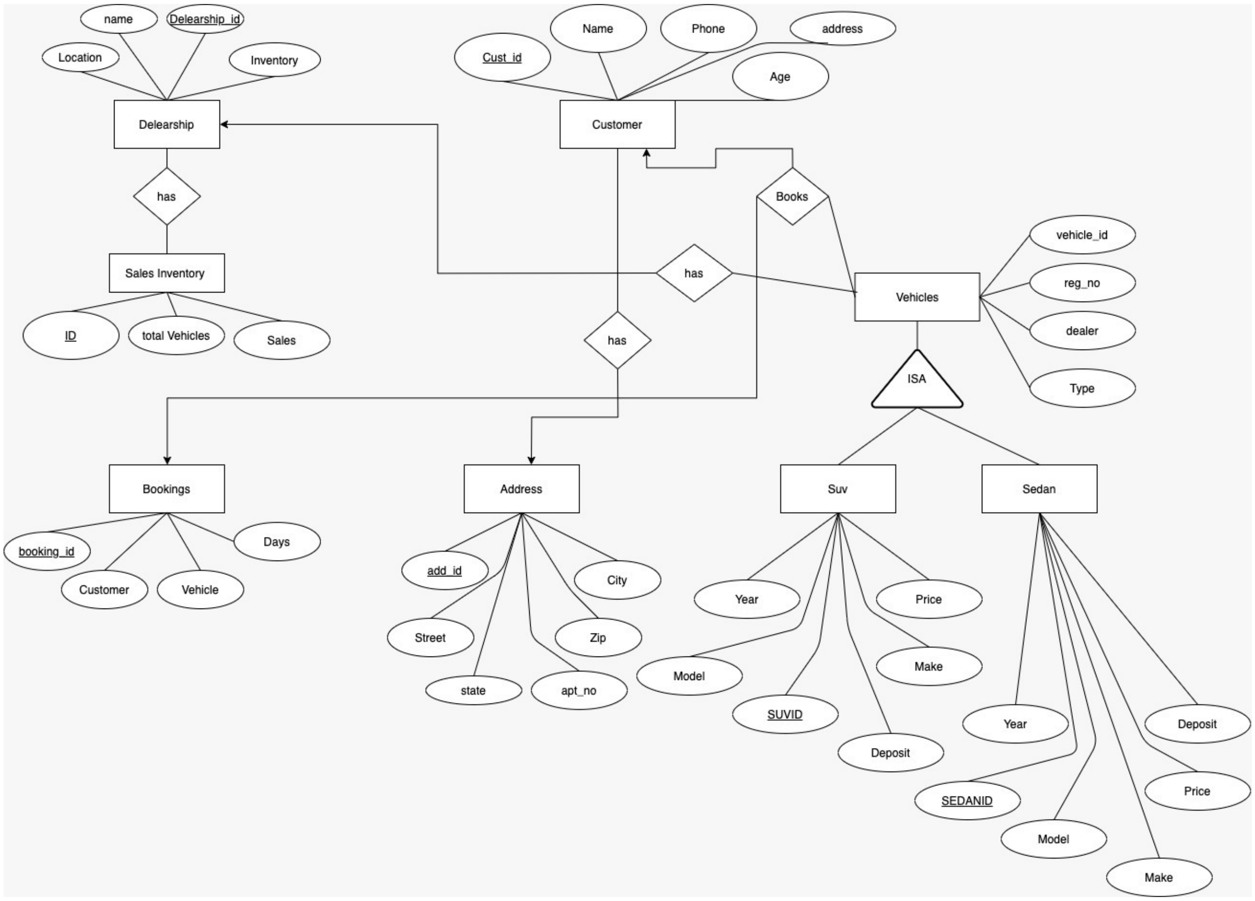
Booking entity booking_id is the primary key, customer attribute refers to customer id in customer entity and vehicle refers to vehicle id in vehicle entity. Default constraint on days with default value 1

Customer entity customer id is primary key and address is foreign key refers to address is in address entity. Check constraint on age for >= 18. Unique constraint on phone number.

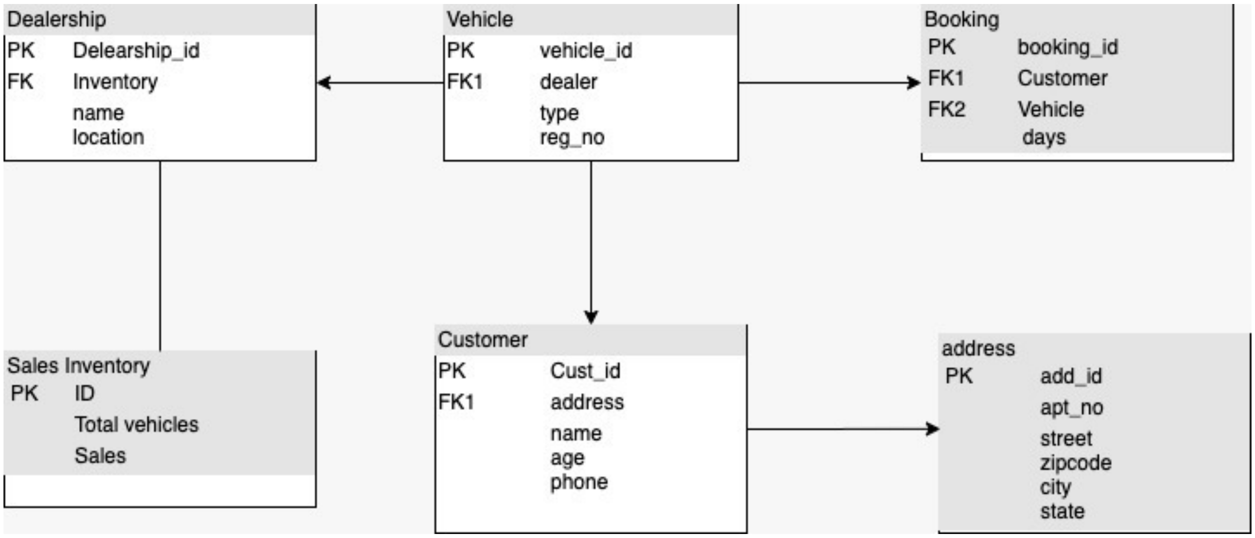
Address entity add_id is the primary key.

For vehicle entity vehicle id is primary key, dealer is foreign key refers to dealership_id in dealership and type refers to id in sedan and suv entities.

For suv and sedan entities suv_id and sedan_id is primary key



B). Relational Schema



Dealership	
PK	Delearship_id
FK	Inventory
	name
	location

Vehicle	
PK	vehicle_id
FK1	dealer
	type
	reg_no

Booking	
PK	booking_id
FK1	Customer
FK2	Vehicle
	days

Sales Inventory	
PK	ID
	Total vehicles
	Sales

Customer	
PK	Cust_id
FK1	address
	name
	age
	phone

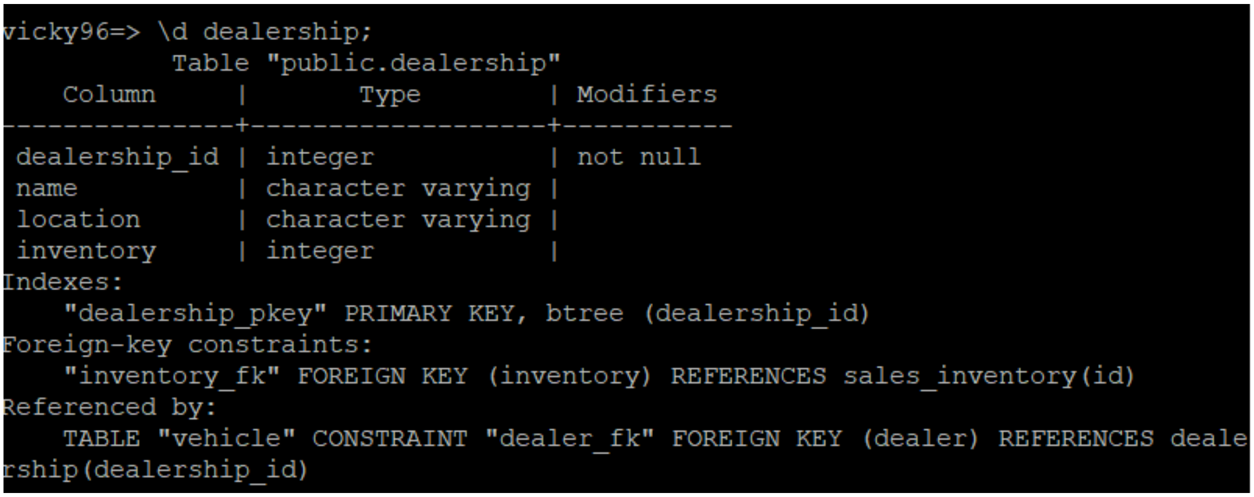
address	
PK	add_id
	apt_no
	street
	zipcode
	city
	state

Here it is in 1NF and 2NF and one attribute doesn’t depend on another attribute i.e there is no transitive dependency. So this relational schema is in 3NF. All the attribute values are atomic and no partial dependency .

Note:Create statements below does not have constraints as they were added later based on requirements. However table schema included all the contarints.

SQl create statement: CREATE TABLE dealership(dealership_id INTEGER PRIMARY KEY, name VARCHAR, location VARCHAR, inventory INTEGER);

Dealership(dealership_id, name, location, Inventory)

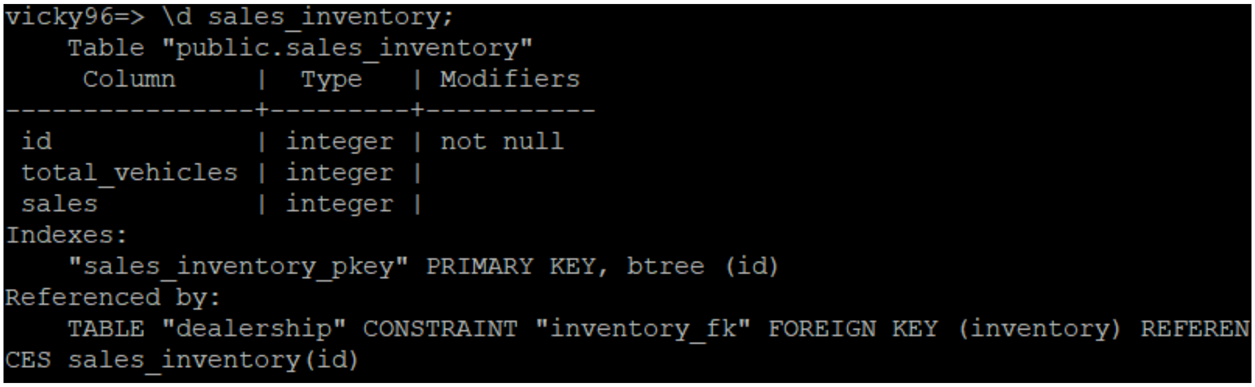


```
vicky96=> \d dealership;
           Table "public.dealership"
    Column     |       Type        | Modifiers
---------------+-------------------+-----------
 dealership_id | integer           | not null
 name          | character varying |
 location      | character varying |
 inventory     | integer           |
Indexes:
    "dealership_pkey" PRIMARY KEY, btree (dealership_id)
Foreign-key constraints:
    "inventory_fk" FOREIGN KEY (inventory) REFERENCES sales_inventory(id)
Referenced by:
    TABLE "vehicle" CONSTRAINT "dealer_fk" FOREIGN KEY (dealer) REFERENCES deale
rship(dealership_id)
```

SQl create statement:

CREATE TABLE sales_inventory(id INTEGER PRIMARY KEY, total_sales INTEGER, sales INTEGER):

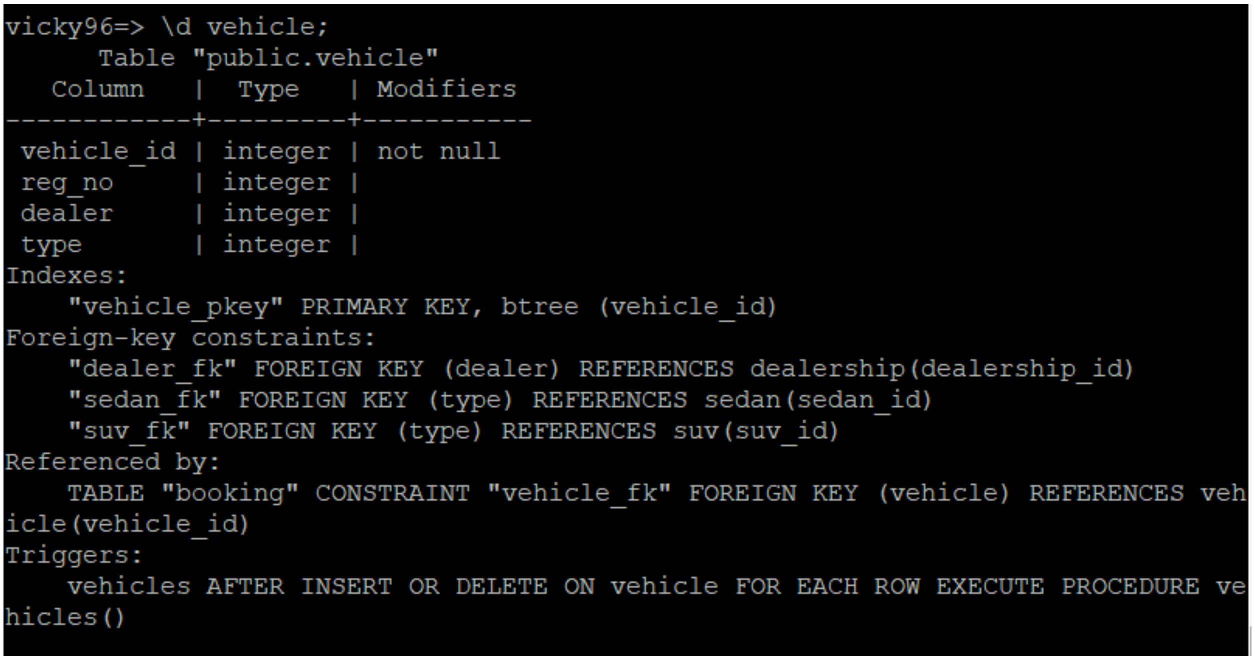
Sales_inventory(id, total_vehicles, sales)



```
vicky96=> \d sales_inventory;
    Table "public.sales_inventory"
     Column     |  Type   | Modifiers
----------------+---------+-----------
 id             | integer | not null
 total_vehicles | integer |
 sales          | integer |
Indexes:
    "sales_inventory_pkey" PRIMARY KEY, btree (id)
Referenced by:
    TABLE "dealership" CONSTRAINT "inventory_fk" FOREIGN KEY (inventory) REFEREN
CES sales_inventory(id)
```

SQl create statement: CREATE TABLE vehicle(vehicle_id INTEGER PRIMARY KEY, reg_no INTEGER, dealer INTEGER, type INTEGER);

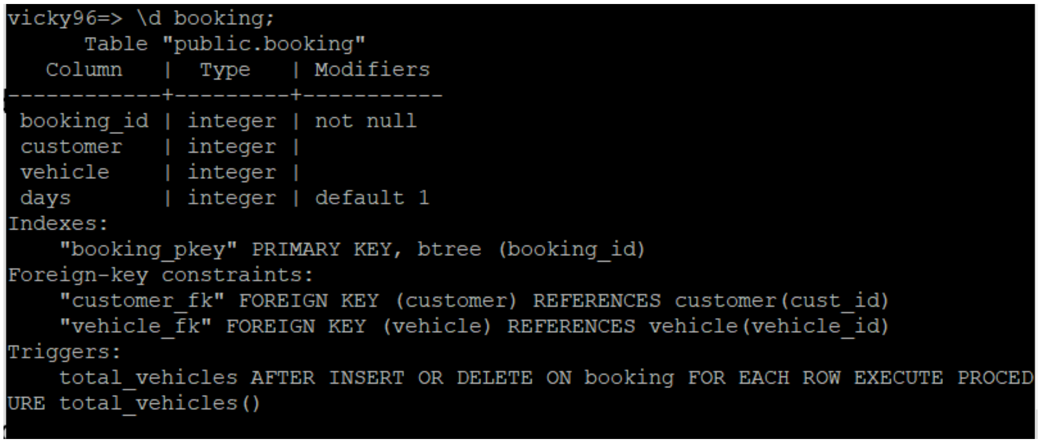
Vehicle(<u>vehicle_id</u>, reg_no, dealer, type)



```
vicky96=> \d vehicle;
      Table "public.vehicle"
   Column   |  Type   | Modifiers
------------+---------+-----------
 vehicle_id | integer | not null
 reg_no     | integer |
 dealer     | integer |
 type       | integer |
Indexes:
    "vehicle_pkey" PRIMARY KEY, btree (vehicle_id)
Foreign-key constraints:
    "dealer_fk" FOREIGN KEY (dealer) REFERENCES dealership(dealership_id)
    "sedan_fk" FOREIGN KEY (type) REFERENCES sedan(sedan_id)
    "suv_fk" FOREIGN KEY (type) REFERENCES suv(suv_id)
Referenced by:
    TABLE "booking" CONSTRAINT "vehicle_fk" FOREIGN KEY (vehicle) REFERENCES veh
icle(vehicle_id)
Triggers:
    vehicles AFTER INSERT OR DELETE ON vehicle FOR EACH ROW EXECUTE PROCEDURE ve
hicles()
```

SQl create statement: CREATE TABLE booking(booking_id INTEGER PRIMARY KEY , customer INTEGER, vehicle INTEGER, days INTEGER);

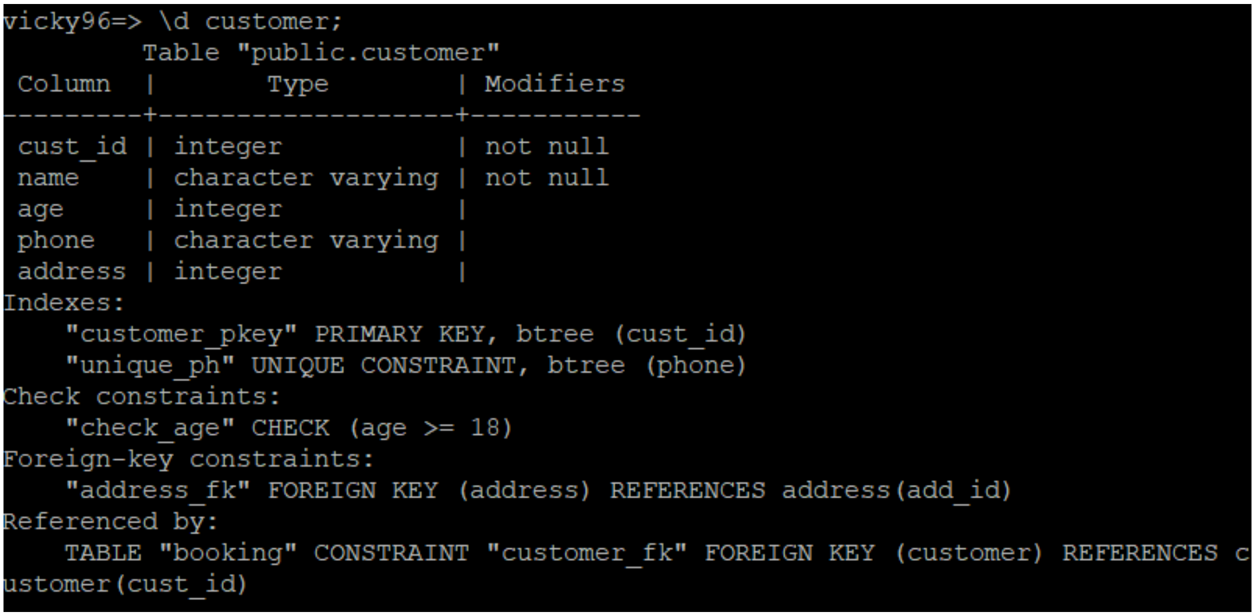
Booking(<u>booking_id</u>, customer, vehicle, days)



```
vicky96=> \d booking;
      Table "public.booking"
   Column   |  Type   | Modifiers
------------+---------+-----------
 booking_id | integer | not null
 customer   | integer |
 vehicle    | integer |
 days       | integer | default 1
Indexes:
    "booking_pkey" PRIMARY KEY, btree (booking_id)
Foreign-key constraints:
    "customer_fk" FOREIGN KEY (customer) REFERENCES customer(cust_id)
    "vehicle_fk" FOREIGN KEY (vehicle) REFERENCES vehicle(vehicle_id)
Triggers:
    total_vehicles AFTER INSERT OR DELETE ON booking FOR EACH ROW EXECUTE PROCED
URE total_vehicles()
```

SQl create statement: CREATE TABLE customer (cust_id INTEGER PRIMARY KEY, name VARCHAR, age INTEGER, phone VARCHAR, address INTEGER):;

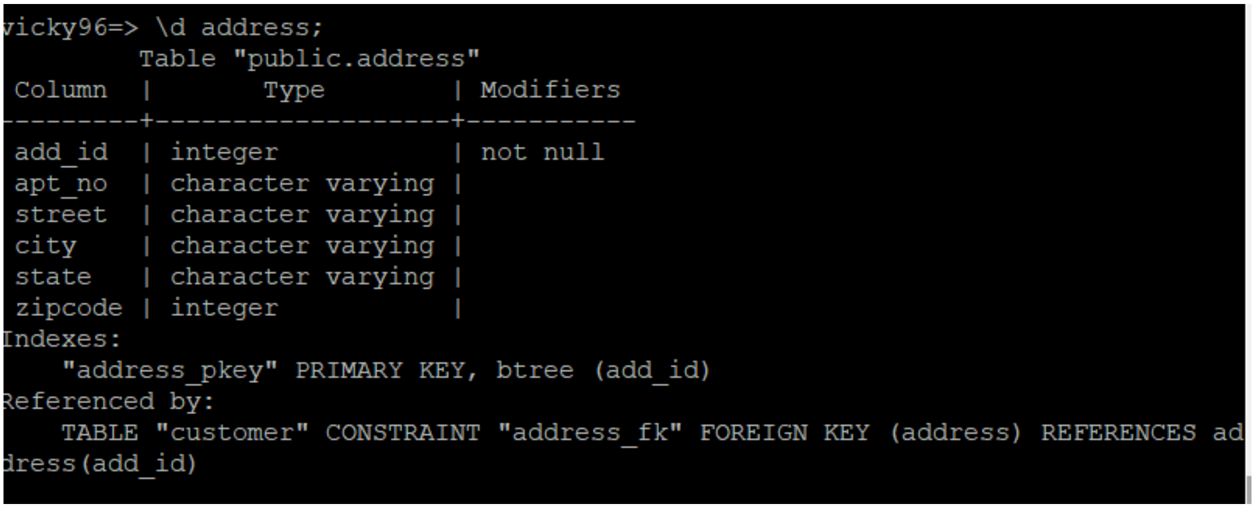
Customer(cust_id, name, age, phone, address)



```
vicky96=> \d customer;
         Table "public.customer"
 Column  |       Type        | Modifiers
---------+-------------------+-----------
 cust_id | integer           | not null
 name    | character varying | not null
 age     | integer           |
 phone   | character varying |
 address | integer           |
Indexes:
    "customer_pkey" PRIMARY KEY, btree (cust_id)
    "unique_ph" UNIQUE CONSTRAINT, btree (phone)
Check constraints:
    "check_age" CHECK (age >= 18)
Foreign-key constraints:
    "address_fk" FOREIGN KEY (address) REFERENCES address(add_id)
Referenced by:
    TABLE "booking" CONSTRAINT "customer_fk" FOREIGN KEY (customer) REFERENCES c
ustomer(cust_id)
```

SQl create statement: CREATE TABLE address(add_id INTEGER PRIMARY KEY, apt_no VARCHAR, street VARCHAR, city VARCHAR, state VARCHAR, zipcode INTEGER);

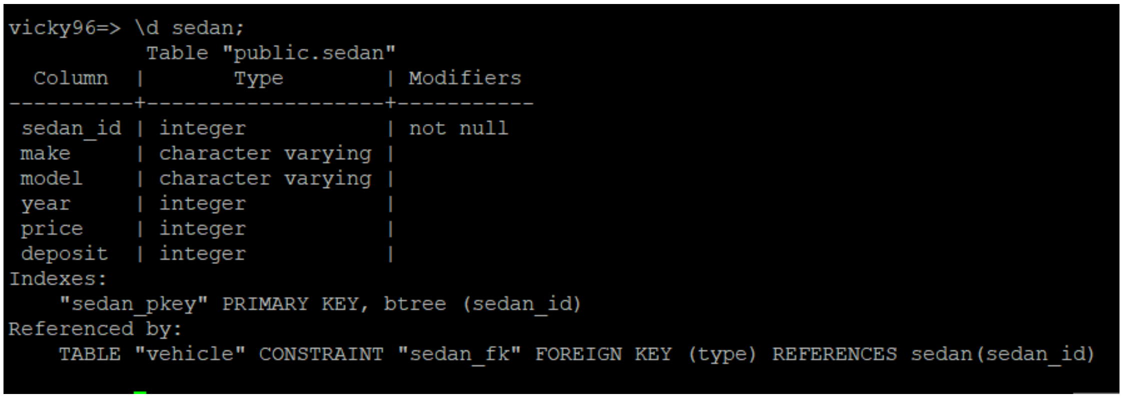
Address(add_id, apt_no, street, city, state, zipcode)



```
vicky96=> \d address;
         Table "public.address"
 Column  |       Type        | Modifiers
---------+-------------------+-----------
 add_id  | integer           | not null
 apt_no  | character varying |
 street  | character varying |
 city    | character varying |
 state   | character varying |
 zipcode | integer           |
Indexes:
    "address_pkey" PRIMARY KEY, btree (add_id)
Referenced by:
    TABLE "customer" CONSTRAINT "address_fk" FOREIGN KEY (address) REFERENCES ad
dress(add_id)
```

SQL Create Statement: CREATE TABLE sedan(sedan_id INTEGER PRIMARY KEY, make VARCHAR, model VARCHAR, year INTEGER, price INTEGER, deposit INTEGER);

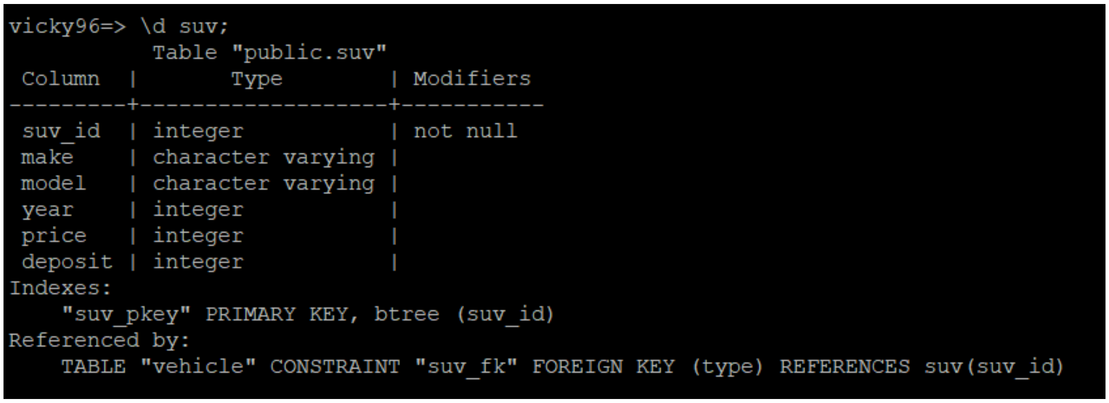
Sedan(sedan_id,make,model,year,price,deposit)



```
vicky96=> \d sedan;
           Table "public.sedan"
  Column  |       Type        | Modifiers
----------+-------------------+-----------
 sedan_id | integer           | not null
 make     | character varying |
 model    | character varying |
 year     | integer           |
 price    | integer           |
 deposit  | integer           |
Indexes:
    "sedan_pkey" PRIMARY KEY, btree (sedan_id)
Referenced by:
    TABLE "vehicle" CONSTRAINT "sedan_fk" FOREIGN KEY (type) REFERENCES sedan(sedan_id)
```

SQL Create Statement: CREATE TABLE suv(suv_id INTEGER PRIMARY KEY, make VARCHAR, model VARCHAR, year INTEGER, price INTEGER, deposit INTEGER);

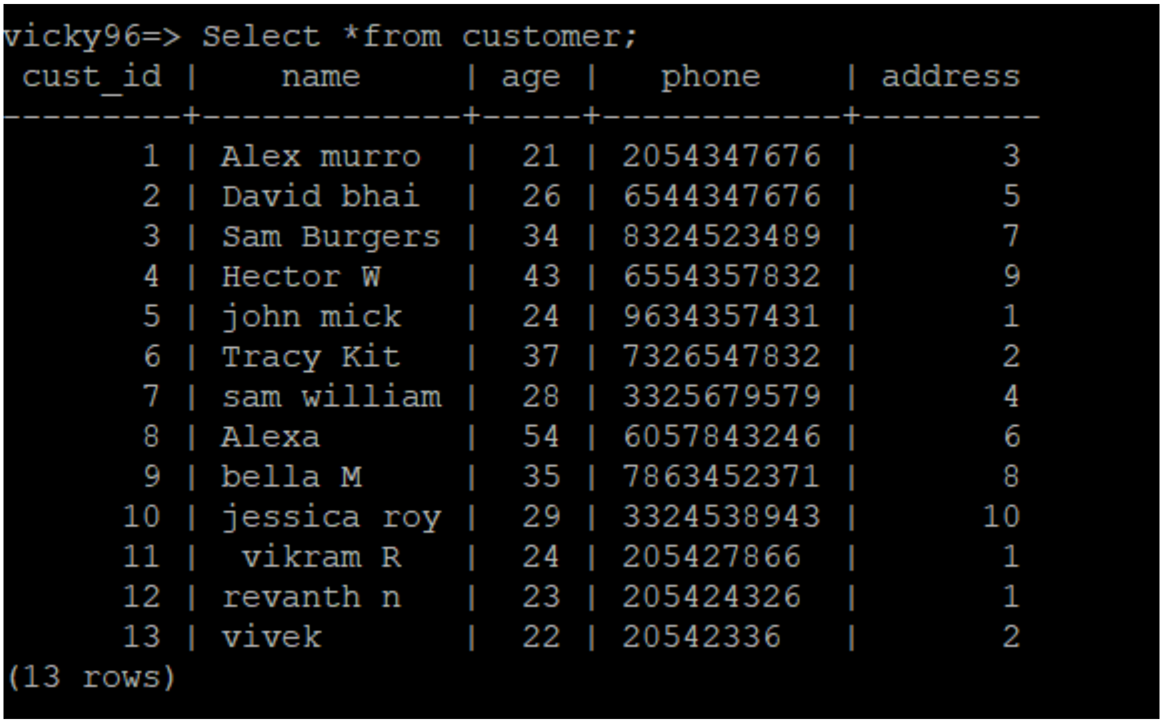
Suv(suv_id,make,model,year,price,deposit)



```
vicky96=> \d suv;
           Table "public.suv"
 Column  |       Type        | Modifiers
---------+-------------------+-----------
 suv_id  | integer           | not null
 make    | character varying |
 model   | character varying |
 year    | integer           |
 price   | integer           |
 deposit | integer           |
Indexes:
    "suv_pkey" PRIMARY KEY, btree (suv_id)
Referenced by:
    TABLE "vehicle" CONSTRAINT "suv_fk" FOREIGN KEY (type) REFERENCES suv(suv_id)
```

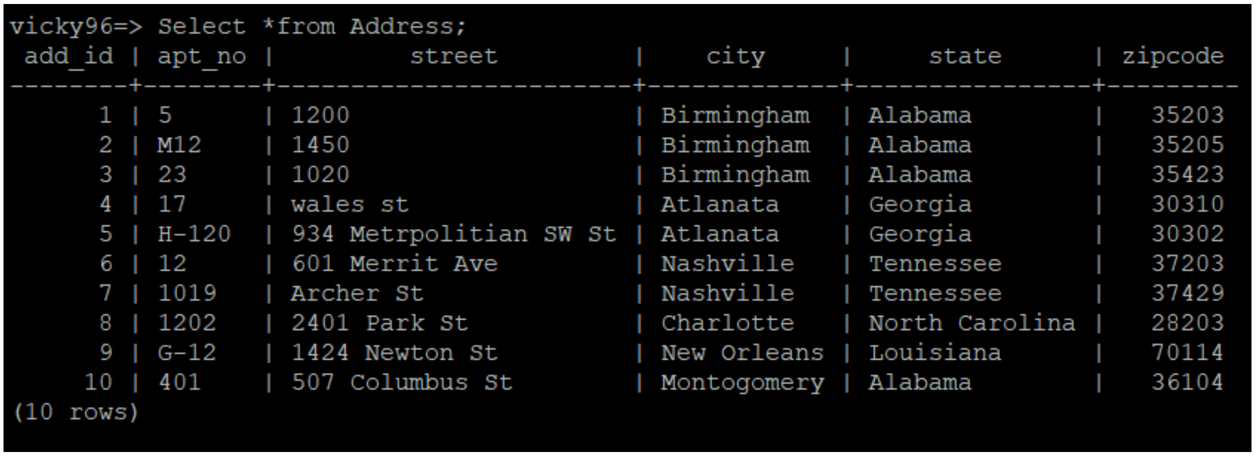
C). Sample Data

Customer relation:



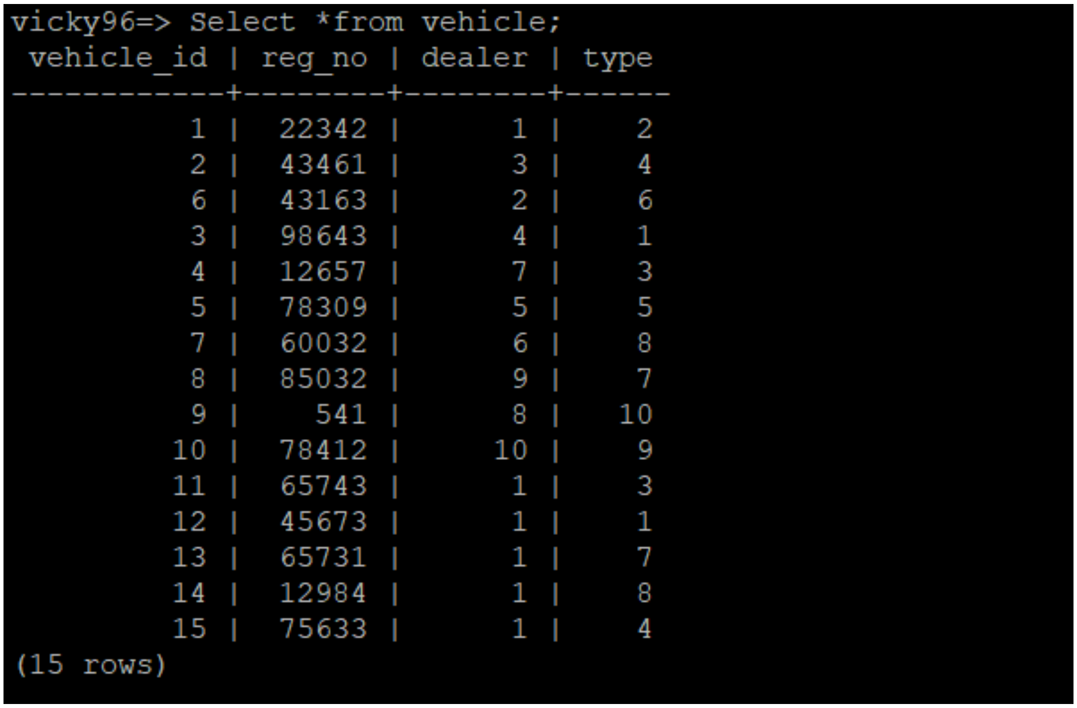
```
vicky96=> Select *from customer;
 cust_id |    name     | age |   phone    | address
---------+-------------+-----+------------+---------
       1 | Alex murro  |  21 | 2054347676 |       3
       2 | David bhai  |  26 | 6544347676 |       5
       3 | Sam Burgers |  34 | 8324523489 |       7
       4 | Hector W    |  43 | 6554357832 |       9
       5 | john mick   |  24 | 9634357431 |       1
       6 | Tracy Kit   |  37 | 7326547832 |       2
       7 | sam william |  28 | 3325679579 |       4
       8 | Alexa       |  54 | 6057843246 |       6
       9 | bella M     |  35 | 7863452371 |       8
      10 | jessica roy |  29 | 3324538943 |      10
      11 |  vikram R   |  24 | 205427866  |       1
      12 | revanth n   |  23 | 205424326  |       1
      13 | vivek       |  22 | 20542336   |       2
(13 rows)
```

Address Relation:



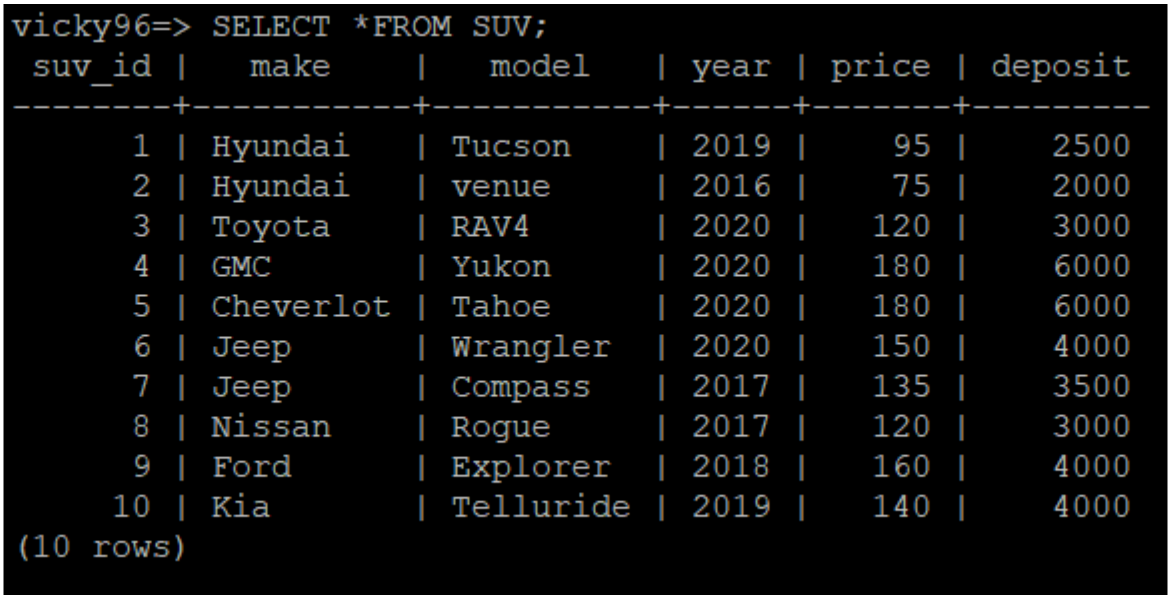
```
vicky96=> Select *from Address;
 add_id | apt_no |         street         |    city     |      state      | zipcode
--------+--------+------------------------+-------------+-----------------+---------
      1 | 5      | 1200                   | Birmingham  | Alabama         |    35203
      2 | M12    | 1450                   | Birmingham  | Alabama         |    35205
      3 | 23     | 1020                   | Birmingham  | Alabama         |    35423
      4 | 17     | wales st               | Atlanata    | Georgia         |    30310
      5 | H-120  | 934 Metrpolitian SW St | Atlanata    | Georgia         |    30302
      6 | 12     | 601 Merrit Ave         | Nashville   | Tennessee       |    37203
      7 | 1019   | Archer St              | Nashville   | Tennessee       |    37429
      8 | 1202   | 2401 Park St           | Charlotte   | North Carolina  |    28203
      9 | G-12   | 1424 Newton St         | New Orleans | Louisiana       |    70114
     10 | 401    | 507 Columbus St        | Montogomery | Alabama         |    36104
(10 rows)
```

Vehicle Relation:



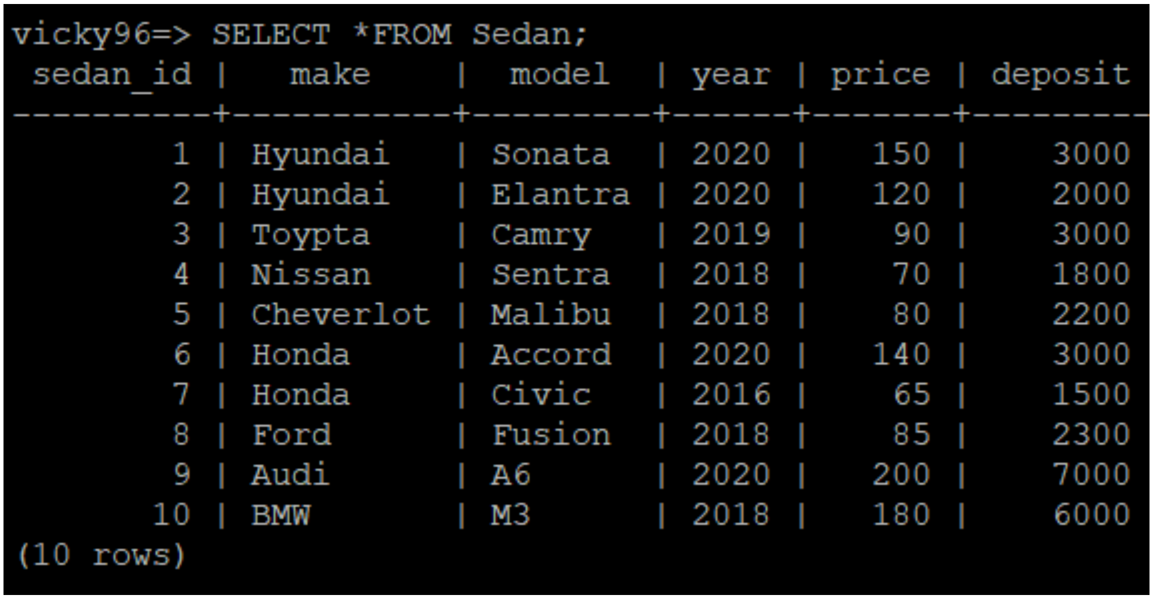
```
vicky96=> Select *from vehicle;
 vehicle_id | reg_no | dealer | type
------------+--------+--------+------
          1 |  22342 |      1 |    2
          2 |  43461 |      3 |    4
          6 |  43163 |      2 |    6
          3 |  98643 |      4 |    1
          4 |  12657 |      7 |    3
          5 |  78309 |      5 |    5
          7 |  60032 |      6 |    8
          8 |  85032 |      9 |    7
          9 |    541 |      8 |   10
         10 |  78412 |     10 |    9
         11 |  65743 |      1 |    3
         12 |  45673 |      1 |    1
         13 |  65731 |      1 |    7
         14 |  12984 |      1 |    8
         15 |  75633 |      1 |    4
(15 rows)
```

SUV Relation:



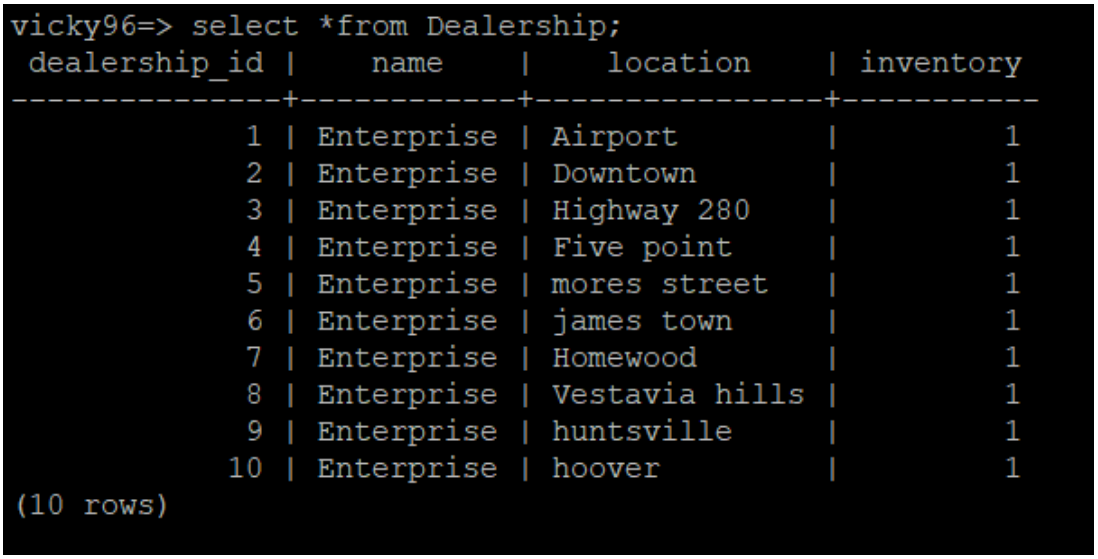
```
vicky96=> SELECT *FROM SUV;
 suv_id |    make    |   model    | year | price | deposit
--------+------------+------------+------+-------+---------
      1 | Hyundai    | Tucson     | 2019 |    95 |    2500
      2 | Hyundai    | venue      | 2016 |    75 |    2000
      3 | Toyota     | RAV4       | 2020 |   120 |    3000
      4 | GMC        | Yukon      | 2020 |   180 |    6000
      5 | Cheverlot  | Tahoe      | 2020 |   180 |    6000
      6 | Jeep       | Wrangler   | 2020 |   150 |    4000
      7 | Jeep       | Compass    | 2017 |   135 |    3500
      8 | Nissan     | Rogue      | 2017 |   120 |    3000
      9 | Ford       | Explorer   | 2018 |   160 |    4000
     10 | Kia        | Telluride  | 2019 |   140 |    4000
(10 rows)
```

Sedan Relation:



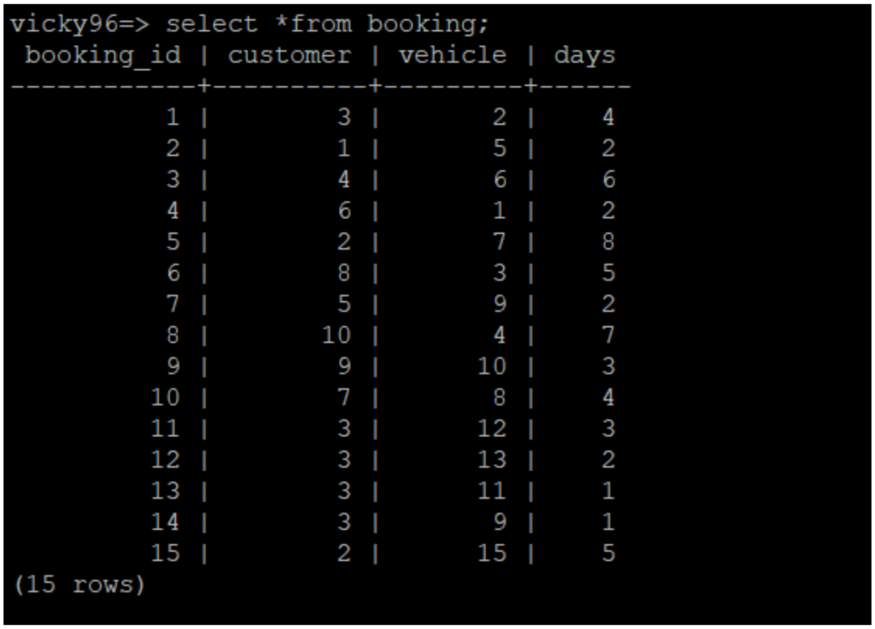
```
vicky96=> SELECT *FROM Sedan;
 sedan_id |   make    |  model  | year | price | deposit
----------+-----------+---------+------+-------+---------
        1 | Hyundai   | Sonata  | 2020 |   150 |    3000
        2 | Hyundai   | Elantra | 2020 |   120 |    2000
        3 | Toypta    | Camry   | 2019 |    90 |    3000
        4 | Nissan    | Sentra  | 2018 |    70 |    1800
        5 | Cheverlot | Malibu  | 2018 |    80 |    2200
        6 | Honda     | Accord  | 2020 |   140 |    3000
        7 | Honda     | Civic   | 2016 |    65 |    1500
        8 | Ford      | Fusion  | 2018 |    85 |    2300
        9 | Audi      | A6      | 2020 |   200 |    7000
       10 | BMW       | M3      | 2018 |   180 |    6000
(10 rows)
```

Dealership Relation:



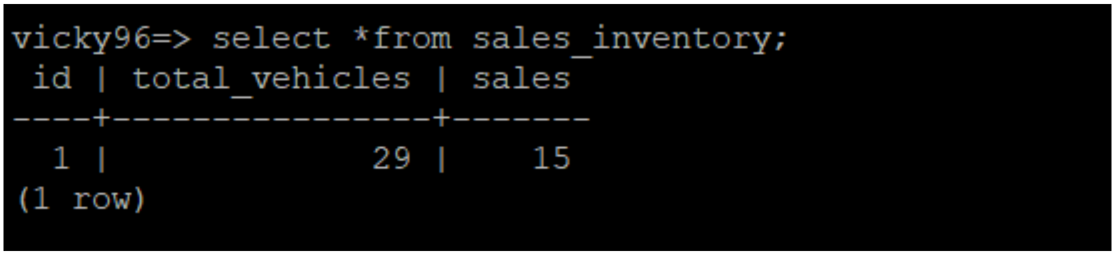
```
vicky96=> select *from Dealership;
 dealership_id |    name    |    location     | inventory
---------------+------------+-----------------+-----------
             1 | Enterprise | Airport         |         1
             2 | Enterprise | Downtown        |         1
             3 | Enterprise | Highway 280     |         1
             4 | Enterprise | Five point      |         1
             5 | Enterprise | mores street    |         1
             6 | Enterprise | james town      |         1
             7 | Enterprise | Homewood        |         1
             8 | Enterprise | Vestavia hills  |         1
             9 | Enterprise | huntsville      |         1
            10 | Enterprise | hoover          |         1
(10 rows)
```

Booking Relation:



```
vicky96=> select *from booking;
 booking_id | customer | vehicle | days
------------+----------+---------+------
          1 |        3 |       2 |    4
          2 |        1 |       5 |    2
          3 |        4 |       6 |    6
          4 |        6 |       1 |    2
          5 |        2 |       7 |    8
          6 |        8 |       3 |    5
          7 |        5 |       9 |    2
          8 |       10 |       4 |    7
          9 |        9 |      10 |    3
         10 |        7 |       8 |    4
         11 |        3 |      12 |    3
         12 |        3 |      13 |    2
         13 |        3 |      11 |    1
         14 |        3 |       9 |    1
         15 |        2 |      15 |    5
(15 rows)
```

Sales_inventory:

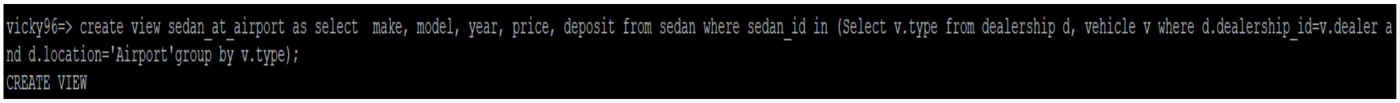


```
vicky96=> select *from sales_inventory;
 id | total_vehicles | sales
----+----------------+-------
  1 |             29 |    15
(1 row)
```

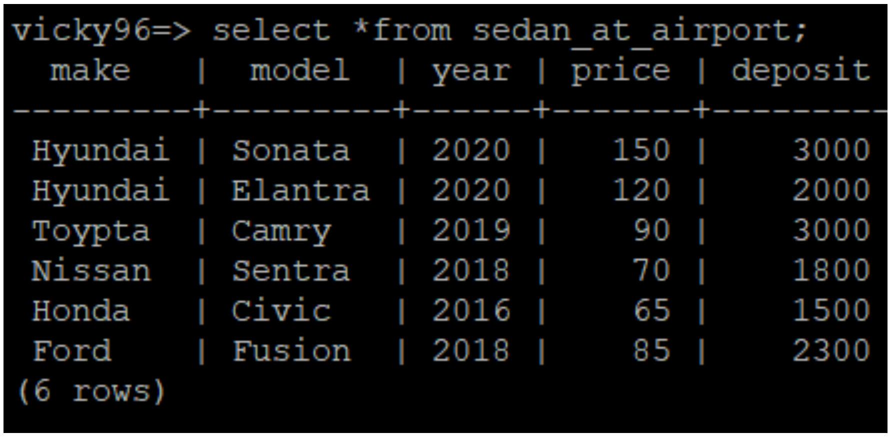
D). VIEWS:

1).This view is for the user who wants to see all the available sedan type cars at airport location.

create view sedan_at_airport as select make, model, year, price, deposit from sedan where sedan_id in (Select v.type from dealership d, vehicle v where d.dealership_id=v.dealer and d.location='Airport'group by v.type);



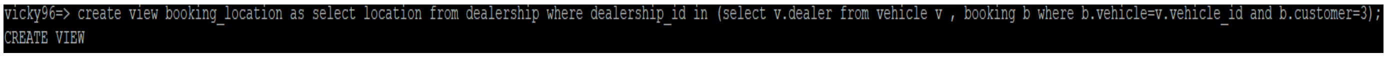
```
vicky96=> create view sedan_at_airport as select  make, model, year, price, deposit from sedan where sedan_id in (Select v.type from dealership d, vehicle v where d.dealership_id=v.dealer a
nd d.location='Airport'group by v.type);
CREATE VIEW
```



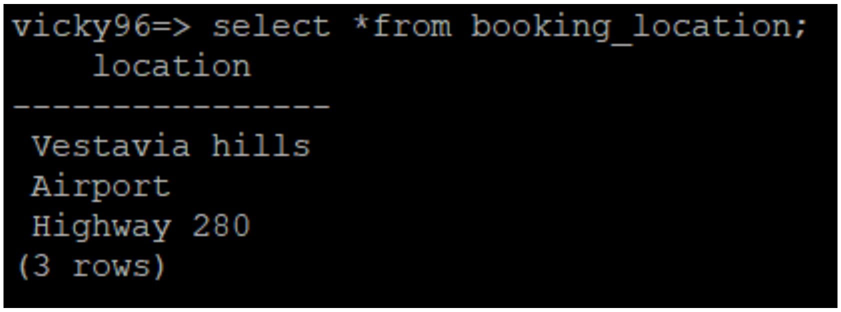
```
vicky96=> select *from sedan_at_airport;
  make    |  model   | year | price | deposit
---------+---------+------+-------+---------
 Hyundai | Sonata  | 2020 |   150 |    3000
 Hyundai | Elantra | 2020 |   120 |    2000
 Toypta  | Camry   | 2019 |    90 |    3000
 Nissan  | Sentra  | 2018 |    70 |    1800
 Honda   | Civic   | 2016 |    65 |    1500
 Ford    | Fusion  | 2018 |    85 |    2300
(6 rows)
```

2). This view is to see all the locations that a customer has made bookings

create view booking_location as select location from dealership where dealership_id in (select v.dealer from vehicle v , booking b where b.vehicle=v.vehicle_id and b.customer=3);



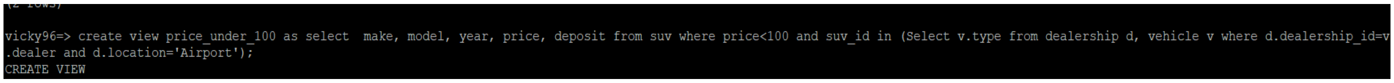
```
vicky96=> create view booking_location as select location from dealership where dealership_id in (select v.dealer from vehicle v , booking b where b.vehicle=v.vehicle_id and b.customer=3);
CREATE VIEW
```



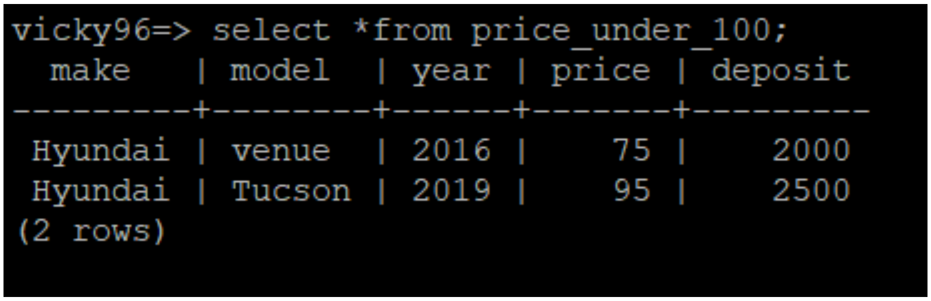
```
vicky96=> select *from booking_location;
    location
----------------
 Vestavia hills
 Airport
 Highway 280
(3 rows)
```

3).This view is to see all the type of vehicles that cost less than 100 at particular dealership location.

create view price_under_100 as select make, model, year, price, deposit from suv where price<100 and suv_id in (Select v.type from dealership d, vehicle v where d.dealership_id=v.dealer and d.location='Airport')



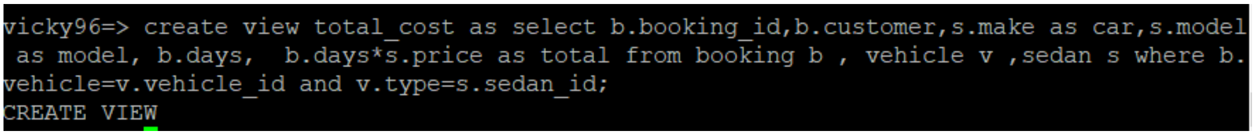
```
(2 rows)

vicky96=> create view price_under_100 as select  make, model, year, price, deposit from suv where price<100 and suv_id in (Select v.type from dealership d, vehicle v where d.dealership_id=v
.dealer and d.location='Airport');
CREATE VIEW
```



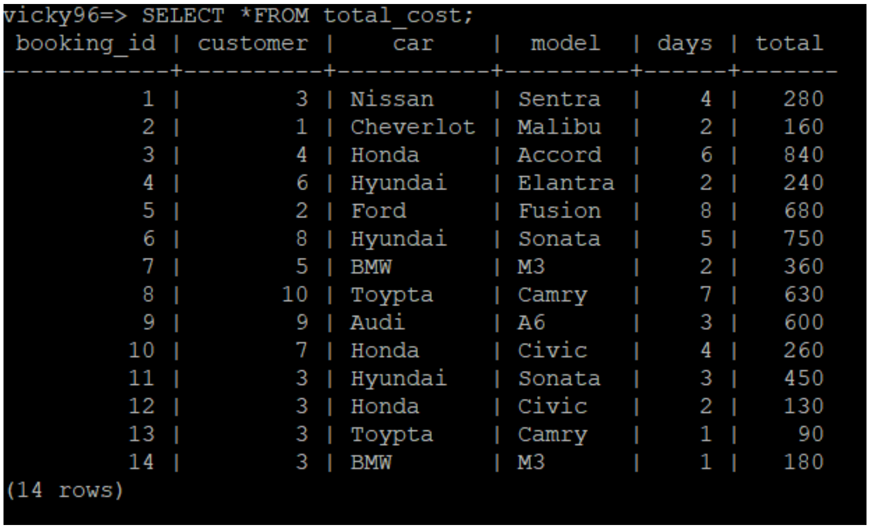
```
vicky96=> select *from price_under_100;
  make    | model   | year | price | deposit
---------+---------+------+-------+---------
 Hyundai | venue   | 2016 |    75 |    2000
 Hyundai | Tucson  | 2019 |    95 |    2500
(2 rows)
```

4.) This view is to see the type of car and model that is booked and total price of the booking.

create view total_cost as select b.booking_id,b.customer,s.make as car,s.model as model, b.days, b.days*s.price as total from booking b , vehicle v ,sedan s where b.vehicle=v.vehicle_id and v.type=s.sedan_id;



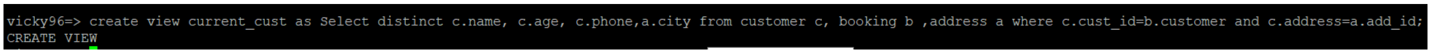
```
vicky96=> create view total_cost as select b.booking_id,b.customer,s.make as car,s.model
 as model, b.days,  b.days*s.price as total from booking b , vehicle v ,sedan s where b.
vehicle=v.vehicle_id and v.type=s.sedan_id;
CREATE VIEW
```



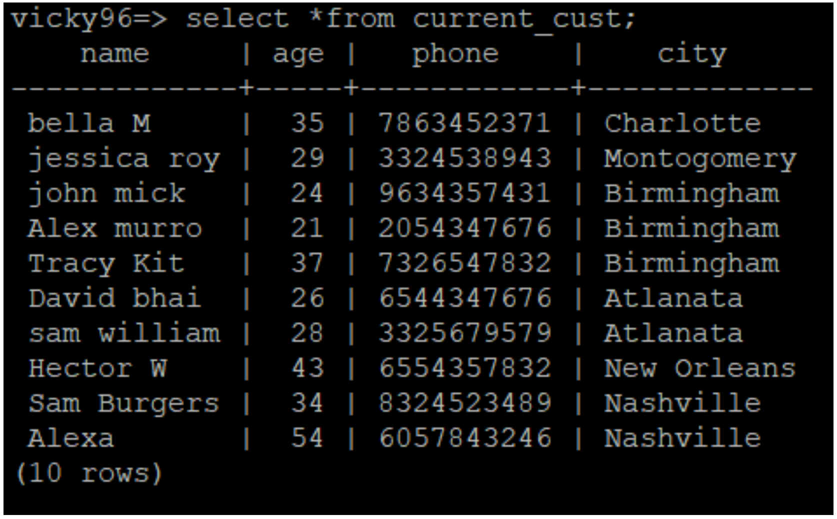
```
vicky96=> SELECT *FROM total_cost;
 booking_id | customer |    car     |  model   | days | total
------------+----------+------------+----------+------+-------
          1 |        3 | Nissan     | Sentra   |    4 |   280
          2 |        1 | Cheverlot  | Malibu   |    2 |   160
          3 |        4 | Honda      | Accord   |    6 |   840
          4 |        6 | Hyundai    | Elantra  |    2 |   240
          5 |        2 | Ford       | Fusion   |    8 |   680
          6 |        8 | Hyundai    | Sonata   |    5 |   750
          7 |        5 | BMW        | M3       |    2 |   360
          8 |       10 | Toypta     | Camry    |    7 |   630
          9 |        9 | Audi       | A6       |    3 |   600
         10 |        7 | Honda      | Civic    |    4 |   260
         11 |        3 | Hyundai    | Sonata   |    3 |   450
         12 |        3 | Honda      | Civic    |    2 |   130
         13 |        3 | Toypta     | Camry    |    1 |    90
         14 |        3 | BMW        | M3       |    1 |   180
(14 rows)
```

5.) This view is to see the details of the customers who are currently made bookings.

create view current_cust as Select distinct c.name, c.age, c.phone,a.city from customer c, booking b ,address a where c.cust_id=b.customer and c.address=a.add_id;



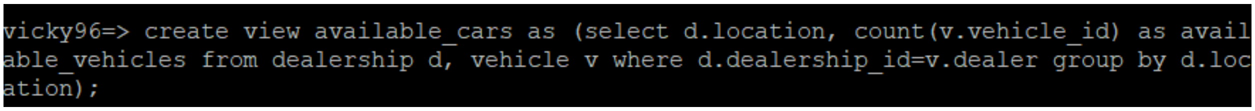
```
vicky96=> create view current_cust as Select distinct c.name, c.age, c.phone,a.city from customer c, booking b ,address a where c.cust_id=b.customer and c.address=a.add_id;
CREATE VIEW
```



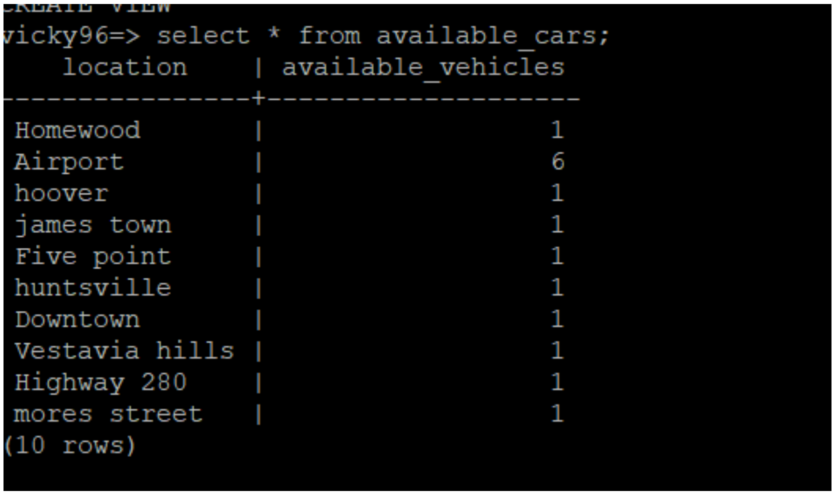
```
vicky96=> select *from current_cust;
    name     | age |   phone    |    city
-------------+-----+------------+-------------
 bella M     |  35 | 7863452371 | Charlotte
 jessica roy |  29 | 3324538943 | Montogomery
 john mick   |  24 | 9634357431 | Birmingham
 Alex murro  |  21 | 2054347676 | Birmingham
 Tracy Kit   |  37 | 7326547832 | Birmingham
 David bhai  |  26 | 6544347676 | Atlanata
 sam william |  28 | 3325679579 | Atlanata
 Hector W    |  43 | 6554357832 | New Orleans
 Sam Burgers |  34 | 8324523489 | Nashville
 Alexa       |  54 | 6057843246 | Nashville
(10 rows)
```

6.) This view is to see total number of vehicles available at each location

create view available_cars as (select d.location, count(v.vehicle_id) as available_vehicles from dealership d, vehicle v where d.dealership_id=v.dealer group by d.location);



```
vicky96=> create view available_cars as (select d.location, count(v.vehicle_id) as avail
able_vehicles from dealership d, vehicle v where d.dealership_id=v.dealer group by d.loc
ation);
```

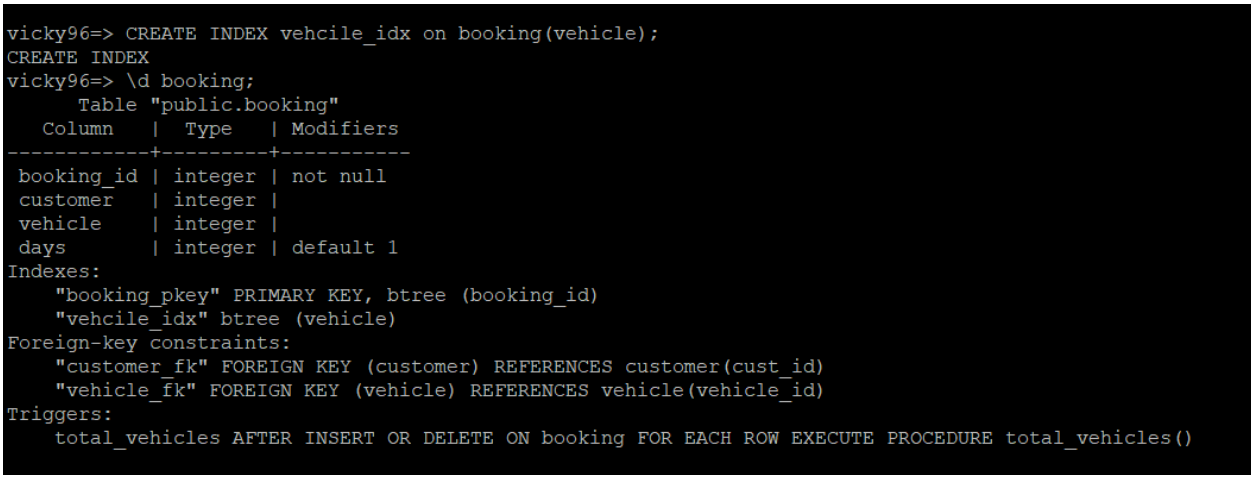


```
CREATE VIEW
vicky96=> select * from available_cars;
    location    | available_vehicles
----------------+--------------------
 Homewood       |                  1
 Airport        |                  6
 hoover         |                  1
 james town     |                  1
 Five point     |                  1
 huntsville     |                  1
 Downtown       |                  1
 Vestavia hills |                  1
 Highway 280    |                  1
 mores street   |                  1
(10 rows)
```

E). INDEXES:

CREATE INDEX vehcile_idx on booking(vehicle);

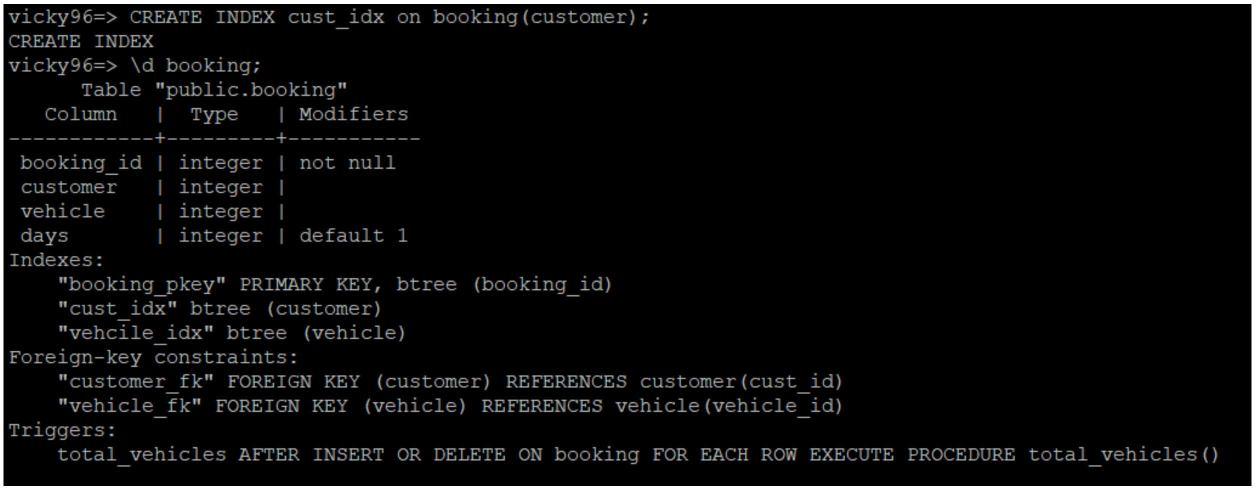
Index on vehicle helps for easy retrival of data from booking while performing comparision in where condition. This helps in the view to see vehicle details.



```
vicky96=> CREATE INDEX vehcile_idx on booking(vehicle);
CREATE INDEX
vicky96=> \d booking;
      Table "public.booking"
   Column   |  Type   | Modifiers
------------+---------+-----------
 booking_id | integer | not null
 customer   | integer |
 vehicle    | integer |
 days       | integer | default 1
Indexes:
    "booking_pkey" PRIMARY KEY, btree (booking_id)
    "vehcile_idx" btree (vehicle)
Foreign-key constraints:
    "customer_fk" FOREIGN KEY (customer) REFERENCES customer(cust_id)
    "vehicle_fk" FOREIGN KEY (vehicle) REFERENCES vehicle(vehicle_id)
Triggers:
    total_vehicles AFTER INSERT OR DELETE ON booking FOR EACH ROW EXECUTE PROCEDURE total_vehicles()
```

CREATE INDEX cust_idx on booking(customer);

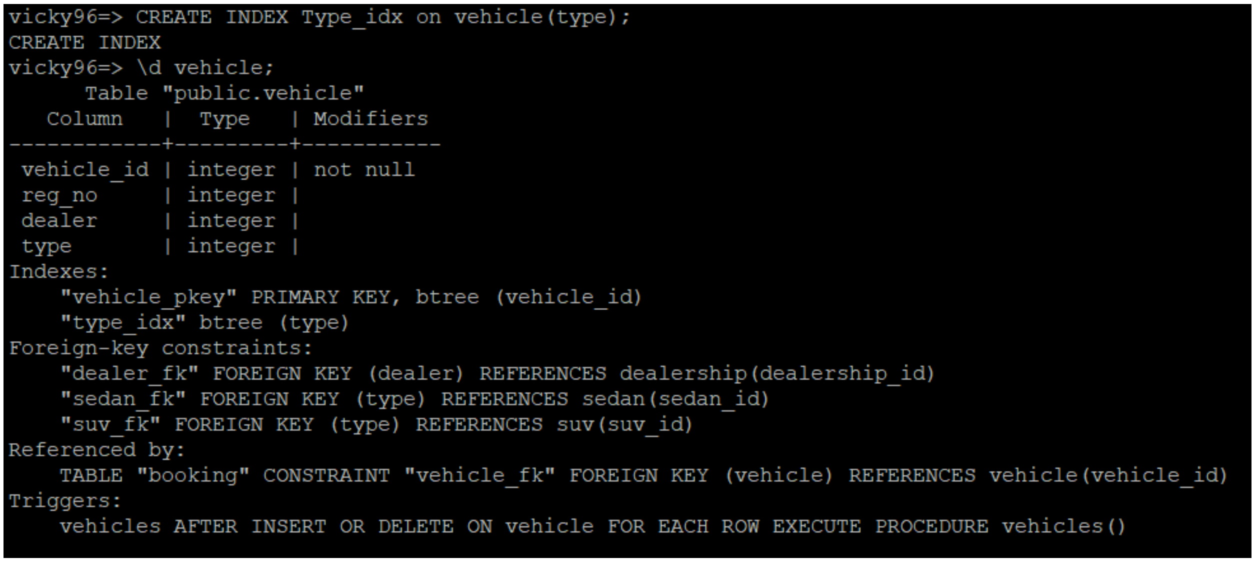
Index on customer in booking helps in view to see the current booking customer details.



```
vicky96=> CREATE INDEX cust_idx on booking(customer);
CREATE INDEX
vicky96=> \d booking;
      Table "public.booking"
   Column   |  Type   | Modifiers
------------+---------+-----------
 booking_id | integer | not null
 customer   | integer |
 vehicle    | integer |
 days       | integer | default 1
Indexes:
    "booking_pkey" PRIMARY KEY, btree (booking_id)
    "cust_idx" btree (customer)
    "vehcile_idx" btree (vehicle)
Foreign-key constraints:
    "customer_fk" FOREIGN KEY (customer) REFERENCES customer(cust_id)
    "vehicle_fk" FOREIGN KEY (vehicle) REFERENCES vehicle(vehicle_id)
Triggers:
    total_vehicles AFTER INSERT OR DELETE ON booking FOR EACH ROW EXECUTE PROCEDURE total_vehicles()
```

CREATE INDEX Type_idx on vehicle(type);

This index helps in the view to see particular type of vehicle details. Type attribute will be always be used in every booking. Index on this improves the performance of database.

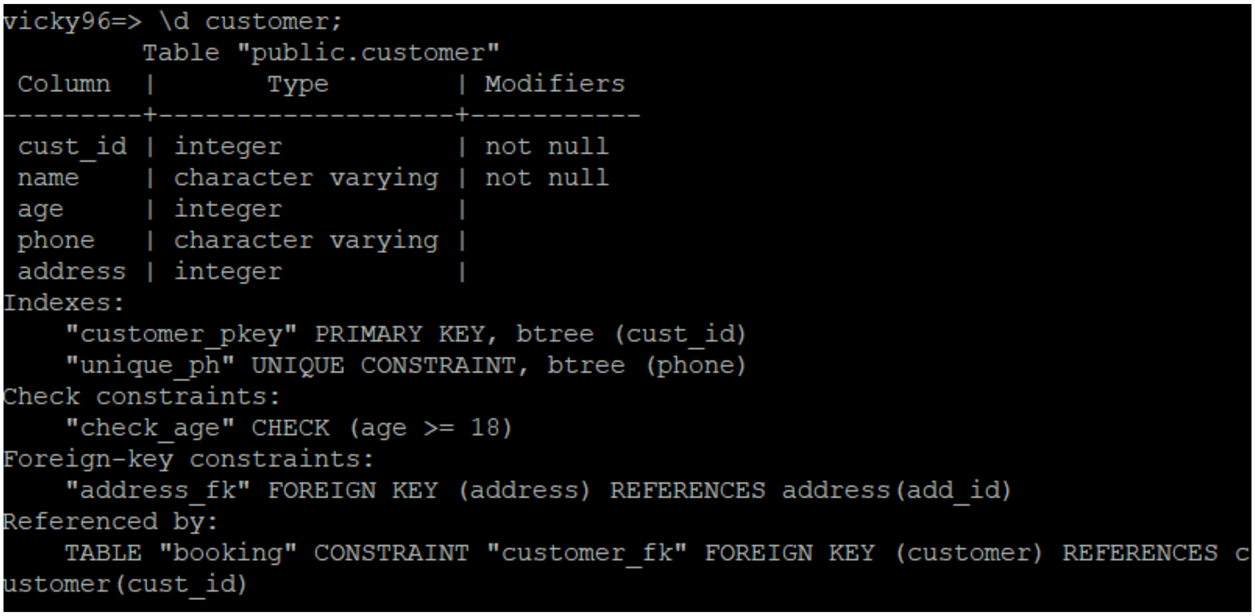


```
vicky96=> CREATE INDEX Type_idx on vehicle(type);
CREATE INDEX
vicky96=> \d vehicle;
      Table "public.vehicle"
   Column   |  Type   | Modifiers
------------+---------+-----------
 vehicle_id | integer | not null
 reg_no     | integer |
 dealer     | integer |
 type       | integer |
Indexes:
    "vehicle_pkey" PRIMARY KEY, btree (vehicle_id)
    "type_idx" btree (type)
Foreign-key constraints:
    "dealer_fk" FOREIGN KEY (dealer) REFERENCES dealership(dealership_id)
    "sedan_fk" FOREIGN KEY (type) REFERENCES sedan(sedan_id)
    "suv_fk" FOREIGN KEY (type) REFERENCES suv(suv_id)
Referenced by:
    TABLE "booking" CONSTRAINT "vehicle_fk" FOREIGN KEY (vehicle) REFERENCES vehicle(vehicle_id)
Triggers:
    vehicles AFTER INSERT OR DELETE ON vehicle FOR EACH ROW EXECUTE PROCEDURE vehicles()
```

F). CONSTRAINTS:

Check Constraint:

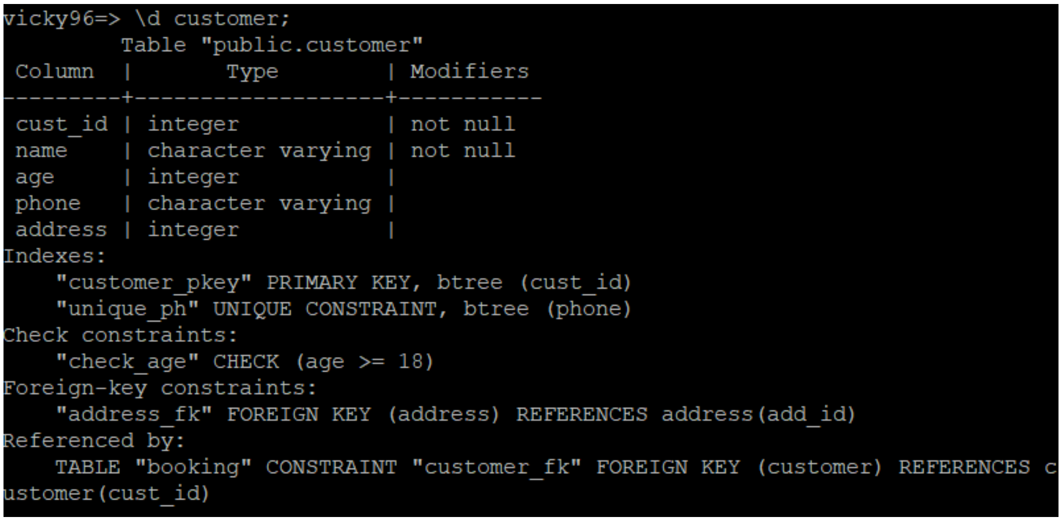
Check constraint on age, For a customer to make a booking of vehicle he/she has to be 18 or more years older.



```
vicky96=> \d customer;
          Table "public.customer"
 Column  |       Type        | Modifiers
---------+-------------------+-----------
 cust_id | integer           | not null
 name    | character varying | not null
 age     | integer           |
 phone   | character varying |
 address | integer           |
Indexes:
    "customer_pkey" PRIMARY KEY, btree (cust_id)
    "unique_ph" UNIQUE CONSTRAINT, btree (phone)
Check constraints:
    "check_age" CHECK (age >= 18)
Foreign-key constraints:
    "address_fk" FOREIGN KEY (address) REFERENCES address(add_id)
Referenced by:
    TABLE "booking" CONSTRAINT "customer_fk" FOREIGN KEY (customer) REFERENCES c
ustomer(cust_id)
```

Unique Constraint:

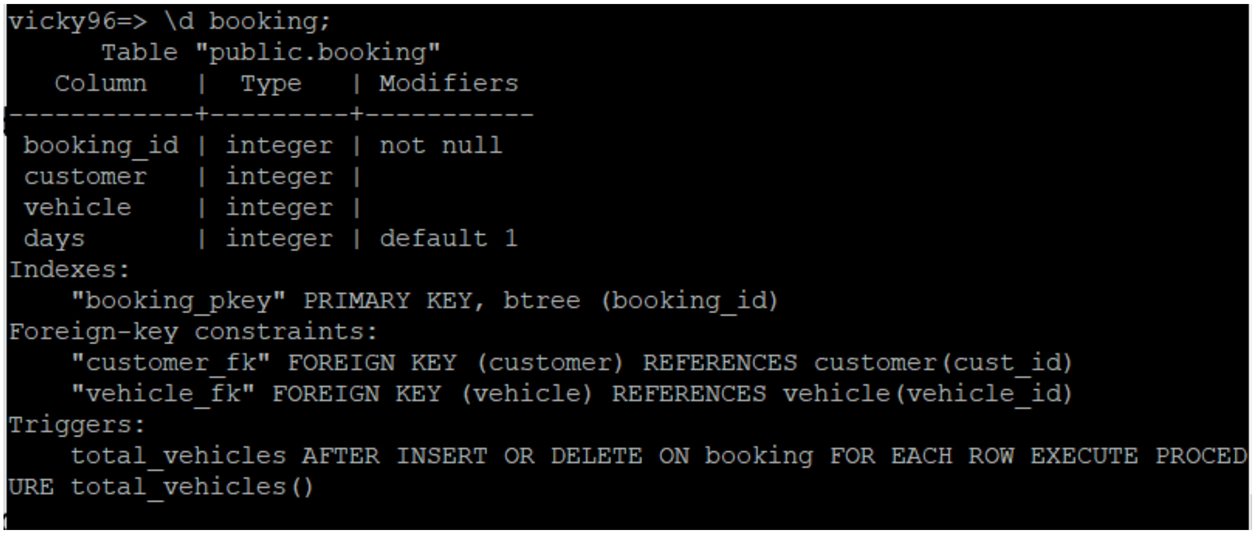
For every customer the phone number has to be unique



```
vicky96=> \d customer;
         Table "public.customer"
 Column  |       Type        | Modifiers
---------+-------------------+-----------
 cust_id | integer           | not null
 name    | character varying | not null
 age     | integer           |
 phone   | character varying |
 address | integer           |
Indexes:
    "customer_pkey" PRIMARY KEY, btree (cust_id)
    "unique_ph" UNIQUE CONSTRAINT, btree (phone)
Check constraints:
    "check_age" CHECK (age >= 18)
Foreign-key constraints:
    "address_fk" FOREIGN KEY (address) REFERENCES address(add_id)
Referenced by:
    TABLE "booking" CONSTRAINT "customer_fk" FOREIGN KEY (customer) REFERENCES c
ustomer(cust_id)
```

Default Constraint:

Booking are only done on number of days basis. So by default for every booking the number of days will be 1.



```
vicky96=> \d booking;
      Table "public.booking"
   Column   |  Type   | Modifiers
------------+---------+-----------
 booking_id | integer | not null
 customer   | integer |
 vehicle    | integer |
 days       | integer | default 1
Indexes:
    "booking_pkey" PRIMARY KEY, btree (booking_id)
Foreign-key constraints:
    "customer_fk" FOREIGN KEY (customer) REFERENCES customer(cust_id)
    "vehicle_fk" FOREIGN KEY (vehicle) REFERENCES vehicle(vehicle_id)
Triggers:
    total_vehicles AFTER INSERT OR DELETE ON booking FOR EACH ROW EXECUTE PROCED
URE total_vehicles()
```

G).TRIGGERS

Every time new booking is done or removed, sales and total_vehicles in sales inventory will be updated. For every new booking total vehicles in ineventory will be decreased by 1 and sales will be increased by 1 and for every delete in bookings, total vehicles will be increased by 1 but sales remain same as the booking is already done.

Trigger 1:

Trigger Function:

```
CREATE or REPLACE FUNCTION total_vehicles()
RETURNS trigger
LANGUAGE plpgsql
AS $function$
BEGIN
if(TG_OP='INSERT') then
UPDATE sales_inventory set sales=sales+1, total_vehicles=total_vehicles-1;
Return NEW;
END IF;
if(TG_op='DELETE') then
UPDATE sales_inventory set total_vehicles=total_vehicles-1;
Return NEW;
END IF;
END;
$function$
;
```

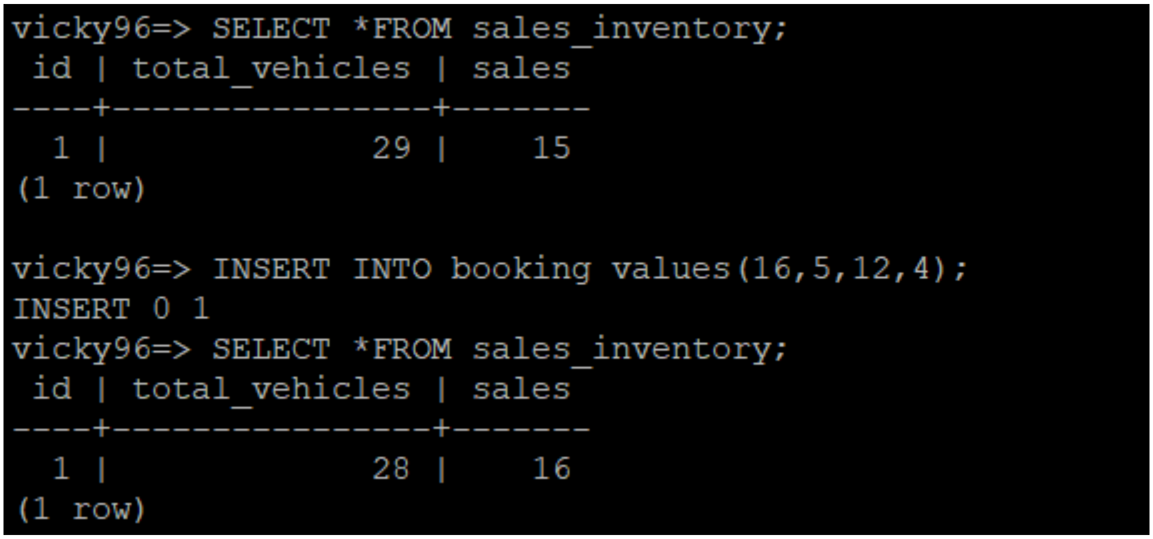
Trigger:

CREATE trigger total_vehicles

vicky96-> AFTER INSERT or DELETE on booking

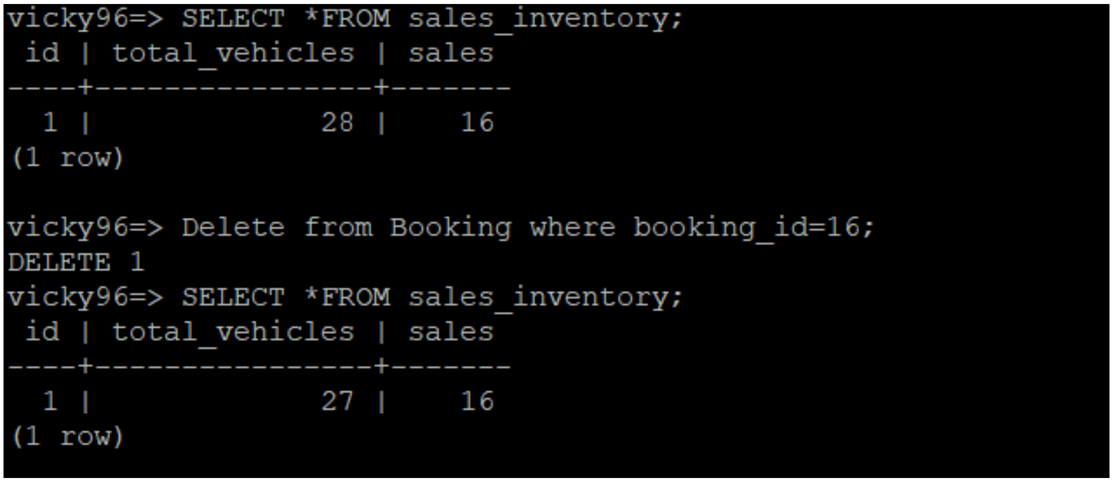
vicky96-> FOR EACH ROW EXECUTE PROCEDURE total_vehicles();

Insert operation of trigger:



```
vicky96=> SELECT *FROM sales_inventory;
 id | total_vehicles | sales
----+----------------+-------
  1 |             29 |    15
(1 row)

vicky96=> INSERT INTO booking values(16,5,12,4);
INSERT 0 1
vicky96=> SELECT *FROM sales_inventory;
 id | total_vehicles | sales
----+----------------+-------
  1 |             28 |    16
(1 row)
```

Delete Operation of trigger:



```
vicky96=> SELECT *FROM sales_inventory;
 id | total_vehicles | sales
----+----------------+-------
  1 |             28 |    16
(1 row)

vicky96=> Delete from Booking where booking_id=16;
DELETE 1
vicky96=> SELECT *FROM sales_inventory;
 id | total_vehicles | sales
----+----------------+-------
  1 |             27 |    16
(1 row)
```

Trigger 2:

This trigger is to update the vehicles in inventory, every time new vehicle is added or removed from vehicle entity. For every insert and delete values will be update accordingly.

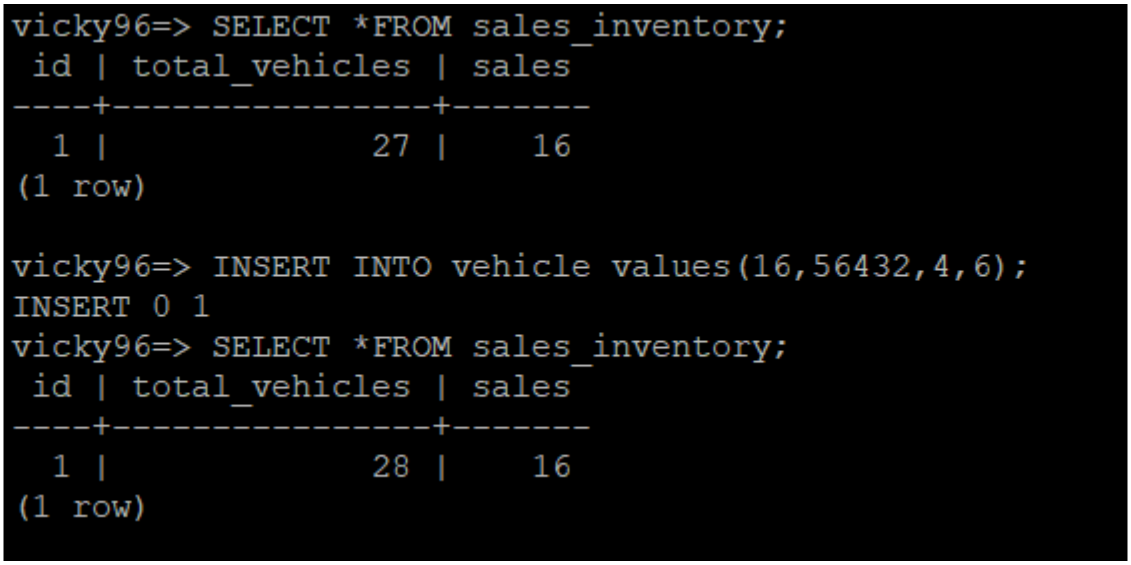
Trigger Function:

```
CREATE or REPLACE FUNCTION vehicles()
RETURNS trigger
LANGUAGE plpgsql
AS $function$
BEGIN
if(TG_OP='INSERT') then
UPDATE sales_inventory set total_vehicles=total_vehicles+1 ;
RETURN NEW;
END IF;
IF(TG_OP='DELETE') then
UPDATE sales_inventory set total_vehicles=total_vehicles-1;
RETURN NEW;
END IF;
END;
$function$
;
```

Trigger:

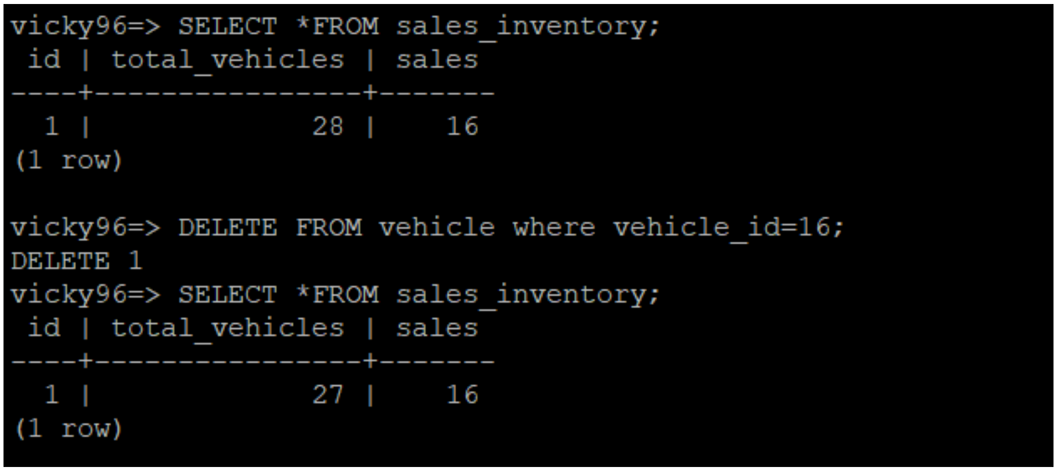
```
CREATE trigger vehicles
AFTER INSERT or DELETE on vehicle
FOR EACH ROW EXECUTE PROCEDURE vehicles();
```

Insert Operation of trigger:



```
vicky96=> SELECT *FROM sales_inventory;
 id | total_vehicles | sales
----+----------------+-------
  1 |             27 |    16
(1 row)

vicky96=> INSERT INTO vehicle values(16,56432,4,6);
INSERT 0 1
vicky96=> SELECT *FROM sales_inventory;
 id | total_vehicles | sales
----+----------------+-------
  1 |             28 |    16
(1 row)
```

Delete Operation of trigger:

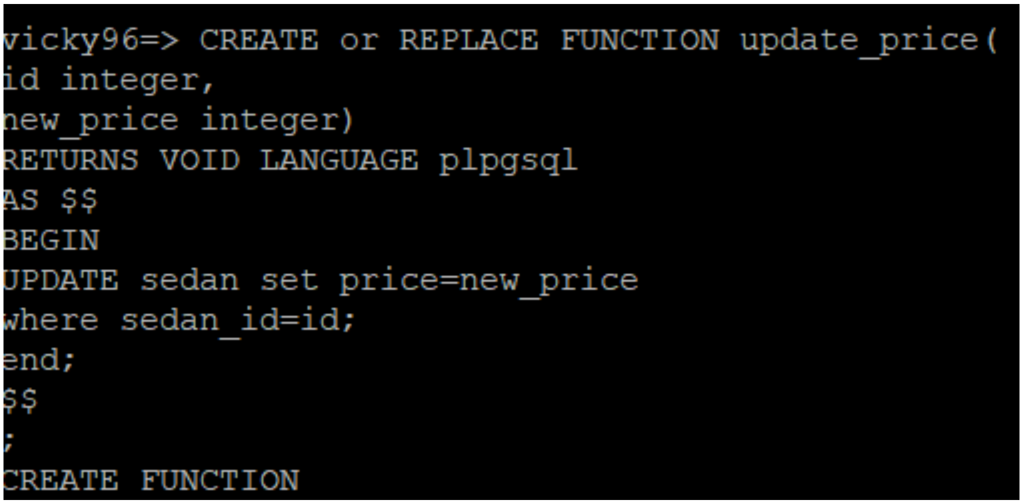


```
vicky96=> SELECT *FROM sales_inventory;
 id | total_vehicles | sales
----+----------------+-------
  1 |             28 |    16
(1 row)

vicky96=> DELETE FROM vehicle where vehicle_id=16;
DELETE 1
vicky96=> SELECT *FROM sales_inventory;
 id | total_vehicles | sales
----+----------------+-------
  1 |             27 |    16
(1 row)
```

H). STORED PROCEDURE

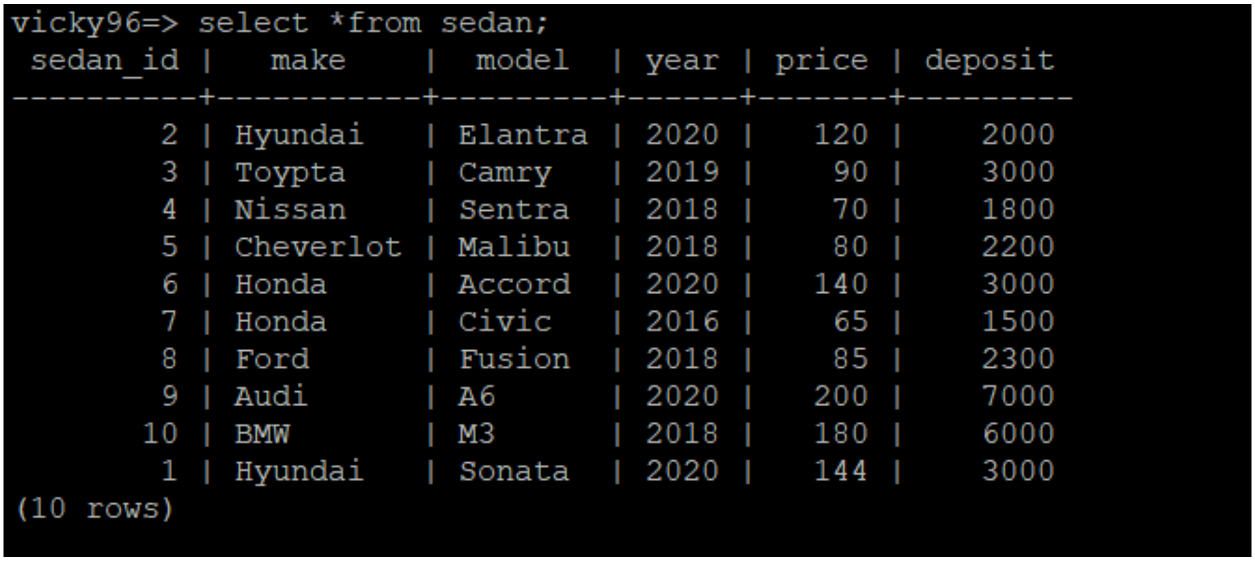
This stored procedure is used to update the price of a sedan car, it takes two arguments sedan_id and new_price and updates the old price with new one.

```
CREATE or REPLACE FUNCTION update_price(
id integer,
new_price integer)
RETURNS VOID LANGUAGE plpgsql
AS $$
BEGIN
UPDATE sedan set price=new_price
where sedan_id=id;
end;
$$
;
```



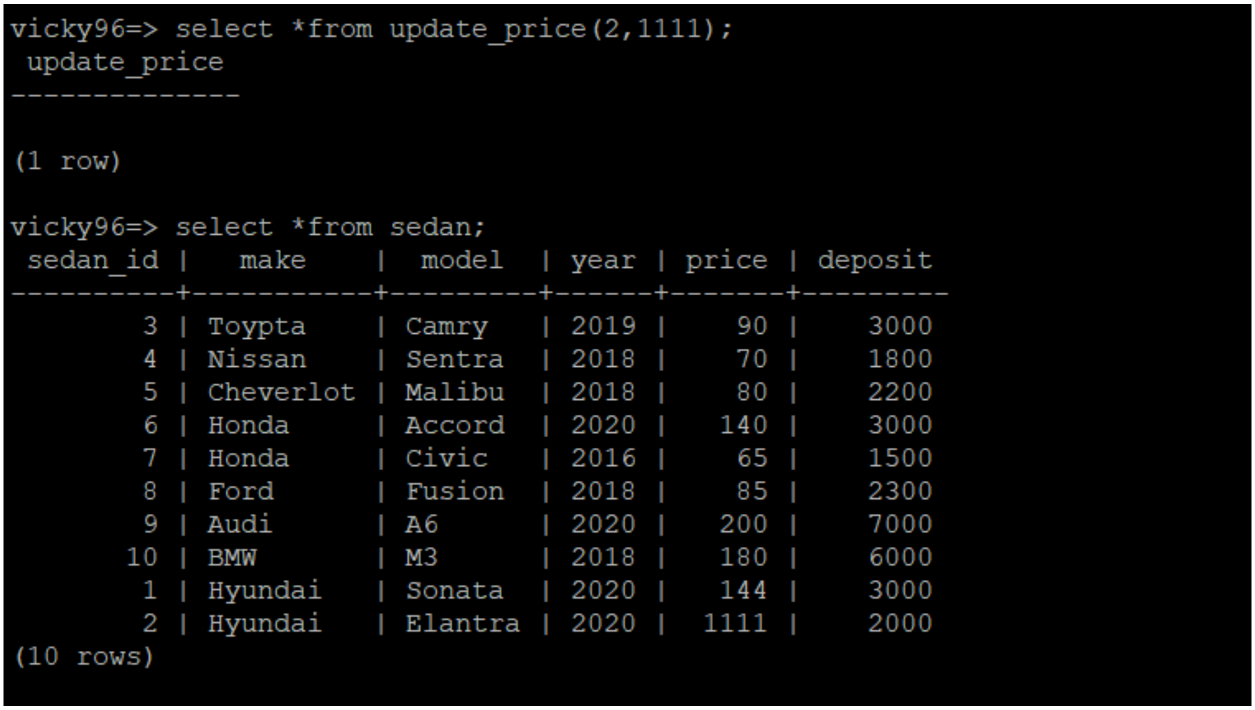
```
vicky96=> CREATE or REPLACE FUNCTION update_price(
id integer,
new_price integer)
RETURNS VOID LANGUAGE plpgsql
AS $$
BEGIN
UPDATE sedan set price=new_price
where sedan_id=id;
end;
$$
;
CREATE FUNCTION
```

Below is the sedan table before running procedure.



```
vicky96=> select *from sedan;
 sedan_id |   make    |  model  | year | price | deposit
----------+-----------+---------+------+-------+---------
        2 | Hyundai   | Elantra | 2020 |   120 |    2000
        3 | Toypta    | Camry   | 2019 |    90 |    3000
        4 | Nissan    | Sentra  | 2018 |    70 |    1800
        5 | Cheverlot | Malibu  | 2018 |    80 |    2200
        6 | Honda     | Accord  | 2020 |   140 |    3000
        7 | Honda     | Civic   | 2016 |    65 |    1500
        8 | Ford      | Fusion  | 2018 |    85 |    2300
        9 | Audi      | A6      | 2020 |   200 |    7000
       10 | BMW       | M3      | 2018 |   180 |    6000
        1 | Hyundai   | Sonata  | 2020 |   144 |    3000
(10 rows)
```

Below is same table after running the procedure, notice the update price for id 2



```
vicky96=> select *from update_price(2,1111);
 update_price
--------------

(1 row)

vicky96=> select *from sedan;
 sedan_id |   make    |  model  | year | price | deposit
----------+-----------+---------+------+-------+---------
        3 | Toypta    | Camry   | 2019 |    90 |    3000
        4 | Nissan    | Sentra  | 2018 |    70 |    1800
        5 | Cheverlot | Malibu  | 2018 |    80 |    2200
        6 | Honda     | Accord  | 2020 |   140 |    3000
        7 | Honda     | Civic   | 2016 |    65 |    1500
        8 | Ford      | Fusion  | 2018 |    85 |    2300
        9 | Audi      | A6      | 2020 |   200 |    7000
       10 | BMW       | M3      | 2018 |   180 |    6000
        1 | Hyundai   | Sonata  | 2020 |   144 |    3000
        2 | Hyundai   | Elantra | 2020 |  1111 |    2000
(10 rows)
```