Table of Contents

[Proposed Automation Strategy: 1](#_Toc481349720)

[Evidence of Additional Research: 2](#_Toc481349721)

[Screen Shots: 3](#_Toc481349722)

[Annotate your screen shot to show functionality 9](#_Toc481349723)

[COMMENTED CODE: 17](#_Toc481349724)

[The test Plan: 47](#_Toc481349725)

# Proposed Automation Strategy:

It is the framework for the database project it shows how the scripts works and what methods are used in it and also through the planning it finalizes the process need to be done to complete the project. Usually the project will start when the schema is finalized using ER diagram. In the schema we can solve the problems using procedure, function and trigger. It outlines the overall plan and during working process when following the process, it is done through PL/SQL and if there comes error then it is solved after a while. This is the clearest process to do work because if the checking process is not use in one by one method then it may harm total done project and can find the problem in easy way also can execute highest number of test case in limited time period. This clarification can be used over multiple projects because it is easy and has complete information regarding testing phase, testing environment and plays vital roles and responsibility of all participants.

This is the core parts of many software systems like database and is the core part of modern days’ software system, management and manipulating data related to specific subject. Sometimes different application may take different code some code most required in some software and some code doesn’t require so to test these kind of application so after coding run the code in the application and testing of the result is much important from there it is known that whether the function and coding is working or not. Thus to avoid these kind of application the automated test application should be made because it would be easy and quick process to solve all the problems.

# Evidence of Additional Research:

Starting to database project group 7 has no specific plan. It is rumoring that from where to start, and how to start. So on Members of the group decided to research about the project relevant website and blogs from there many things were found like easy way of doing database project, codes that are more easy and some blogs from where the database problem were solved easily. The database project always been top class and to solve it research helped most without research may not the project will be fulfilling on time. In the research process there were found good practice material on some of the website and blogs some of them are as follows: -

Using the Harvard referencing standard following website is listed:

1. PL/SQL Triggers. 2017. PL/SQL Triggers. [ONLINE] Available at: <https://docs.oracle.com/cloud/latest/db112/LNPLS/triggers.htm#LNPLS99955>. [Accessed 17 April 2017].
2. PL/SQL Control Structures. 2017. PL/SQL Control Structures. [ONLINE] Available at: <https://docs.oracle.com/cd/A97630_01/appdev.920/a96624/04_struc.htm>. [Accessed 30 April 2017].
3. Oraclecom.2017.Oraclecom.[Online].[17 March 2017]. Available from:

https://docs.oracle.com/cd/A97630\_01/appded.920/a96624/04\_struc.htm

1. Ss64com.2017.Ss64com.[Online].[19 April 2017]. Available from:

https://ss64.com/oraplsql/cursor\_fetch\_close.html

# 

# Screen Shots:

Here are some of the screen shot of the sql plus prompt.

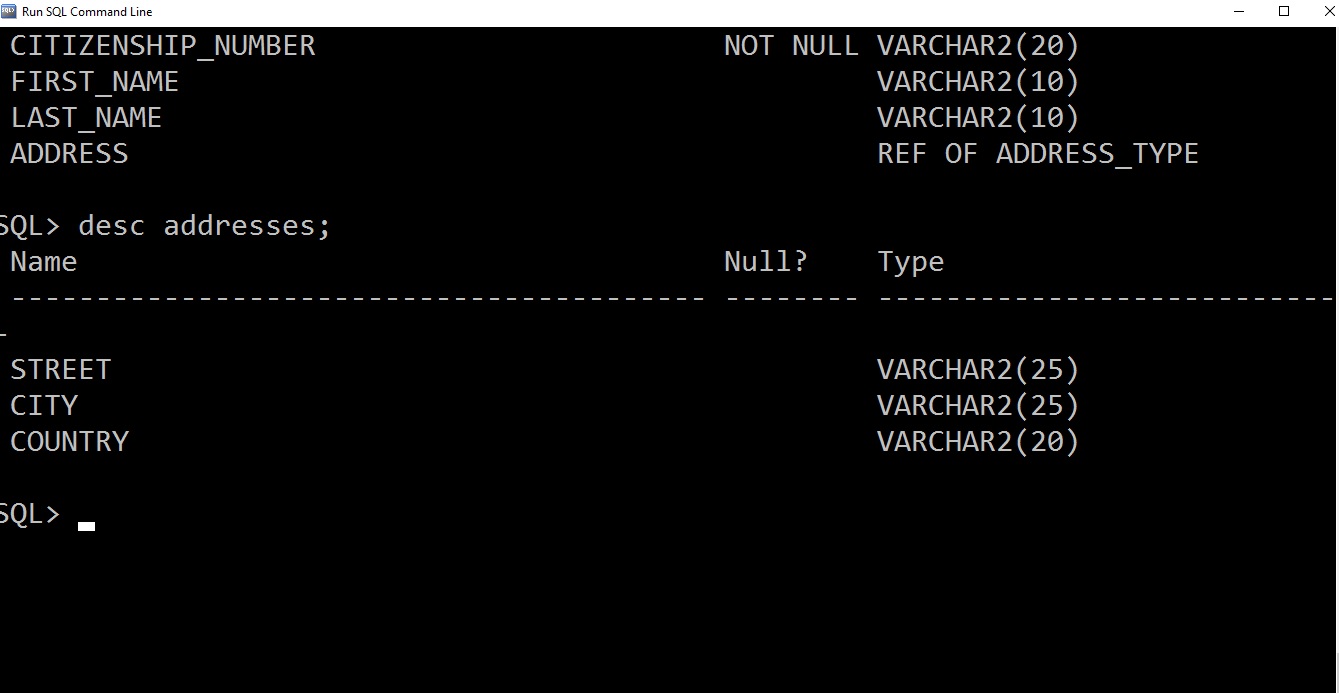


Fig: Addresses table description

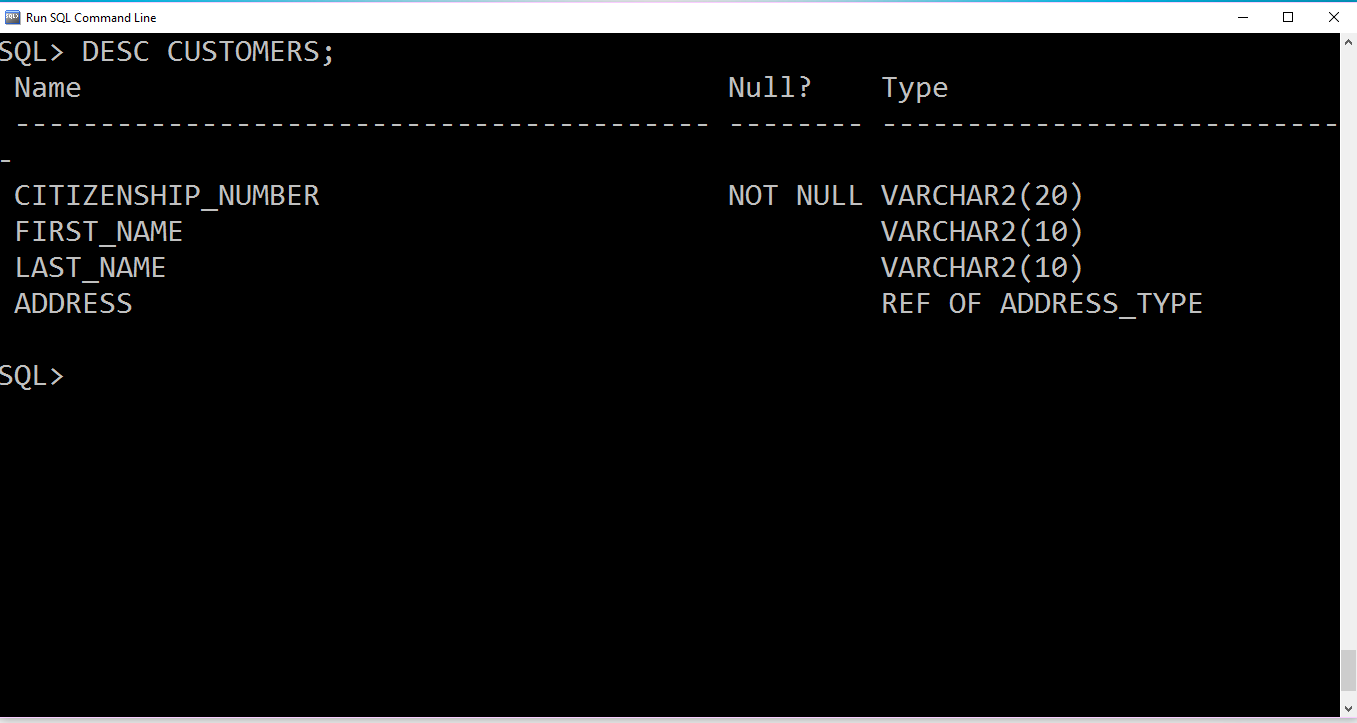


Fig: Customer table description

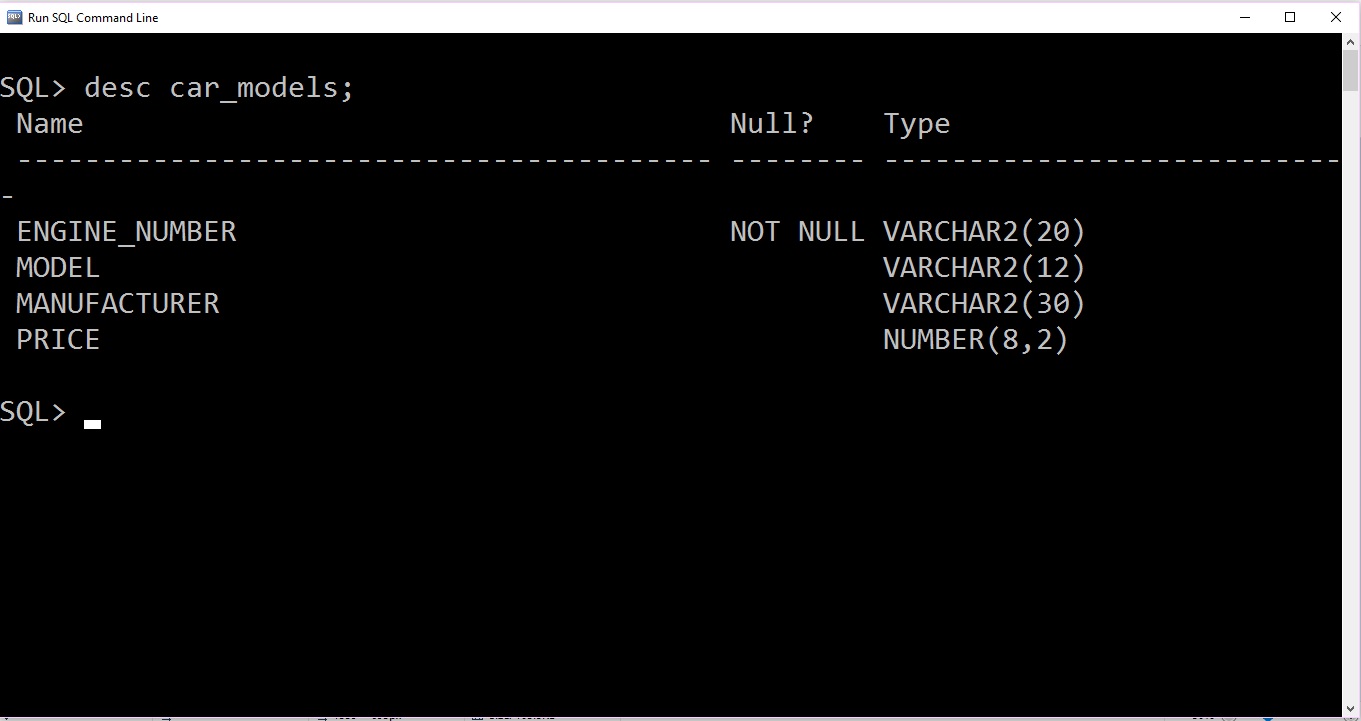


Fig: Car model table description

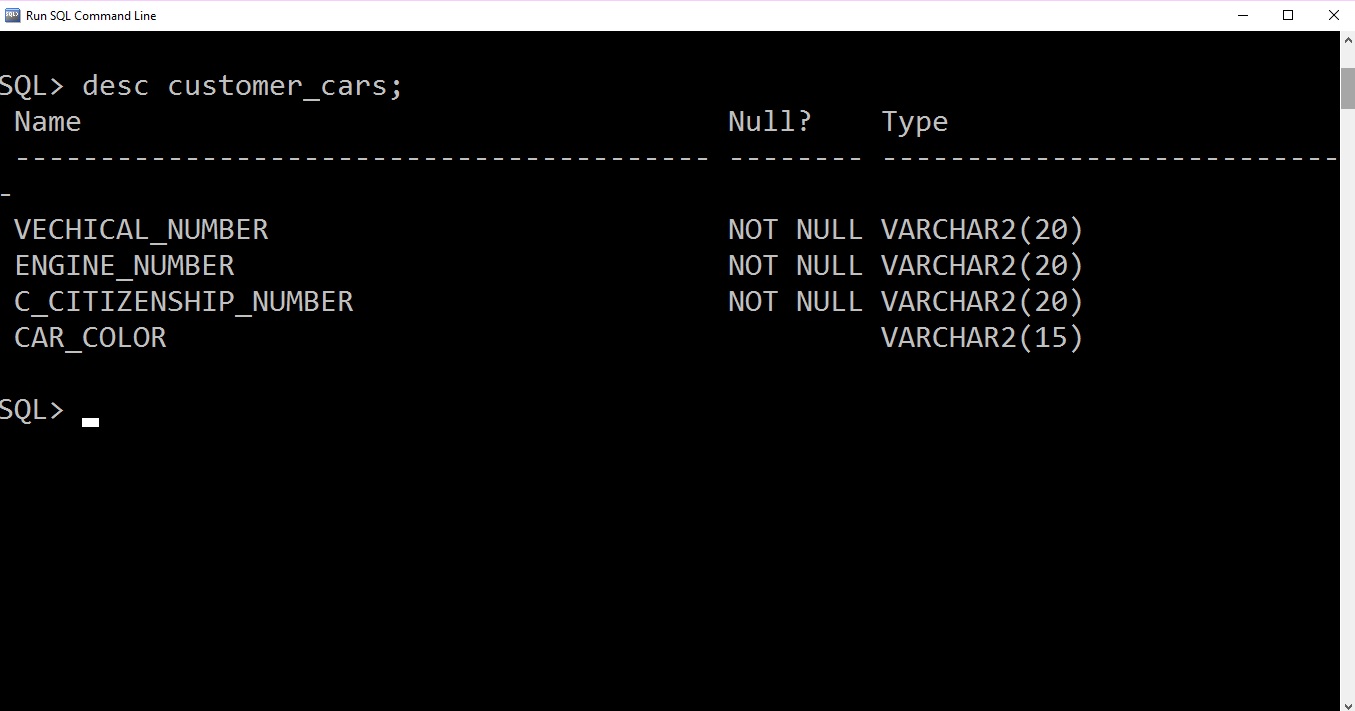


Fig: Customer car description

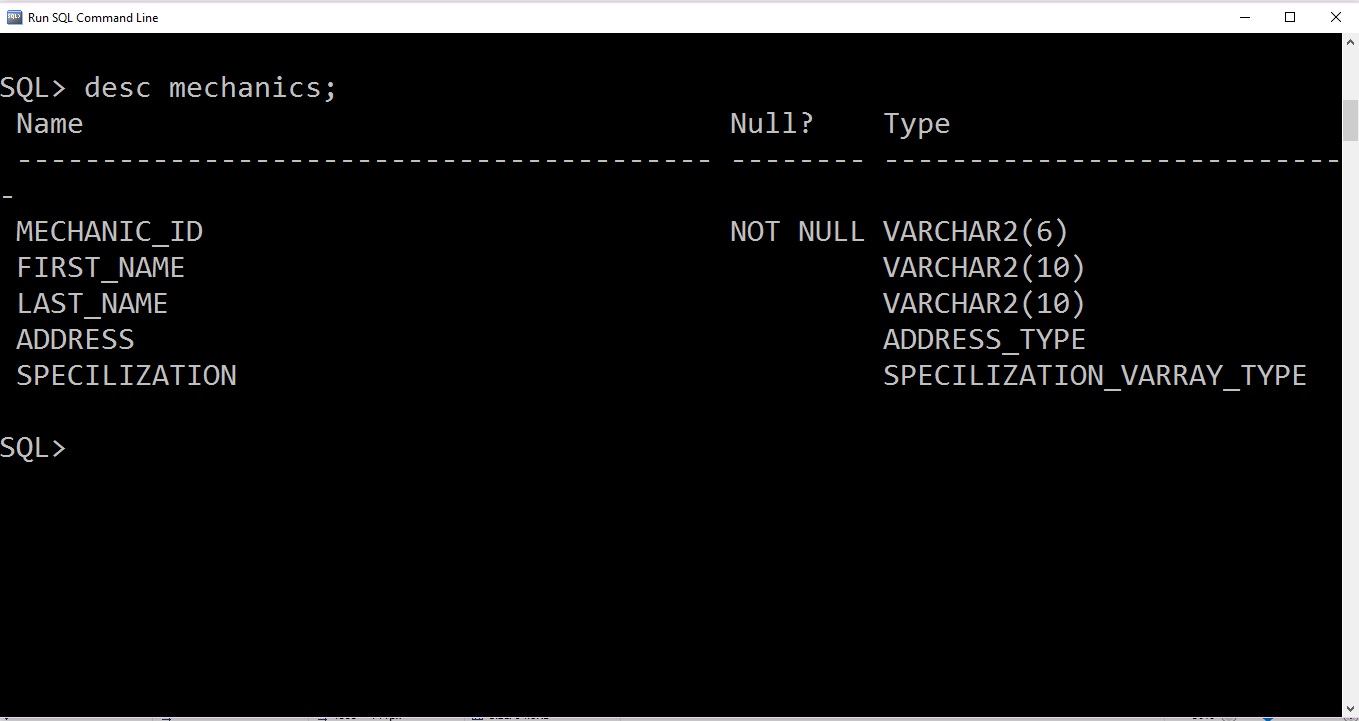


Fig: Mechanic table description

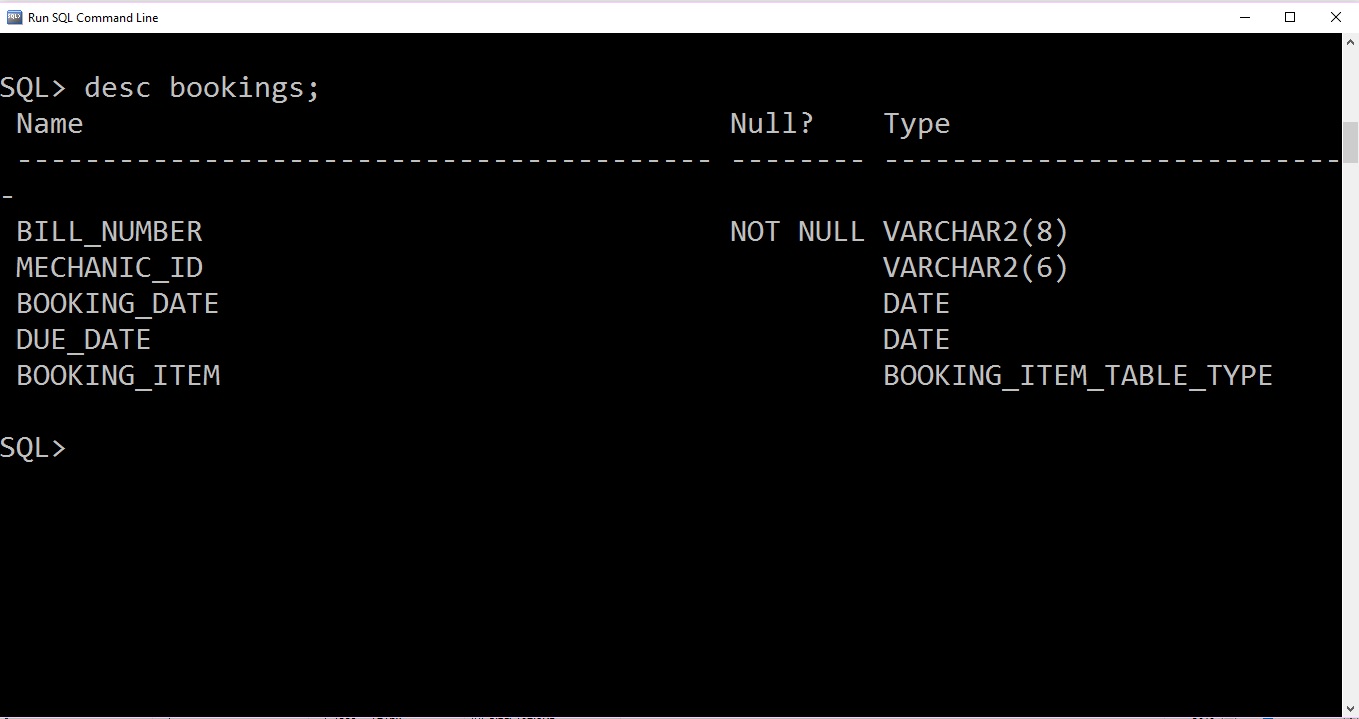


Fig: Booking table description

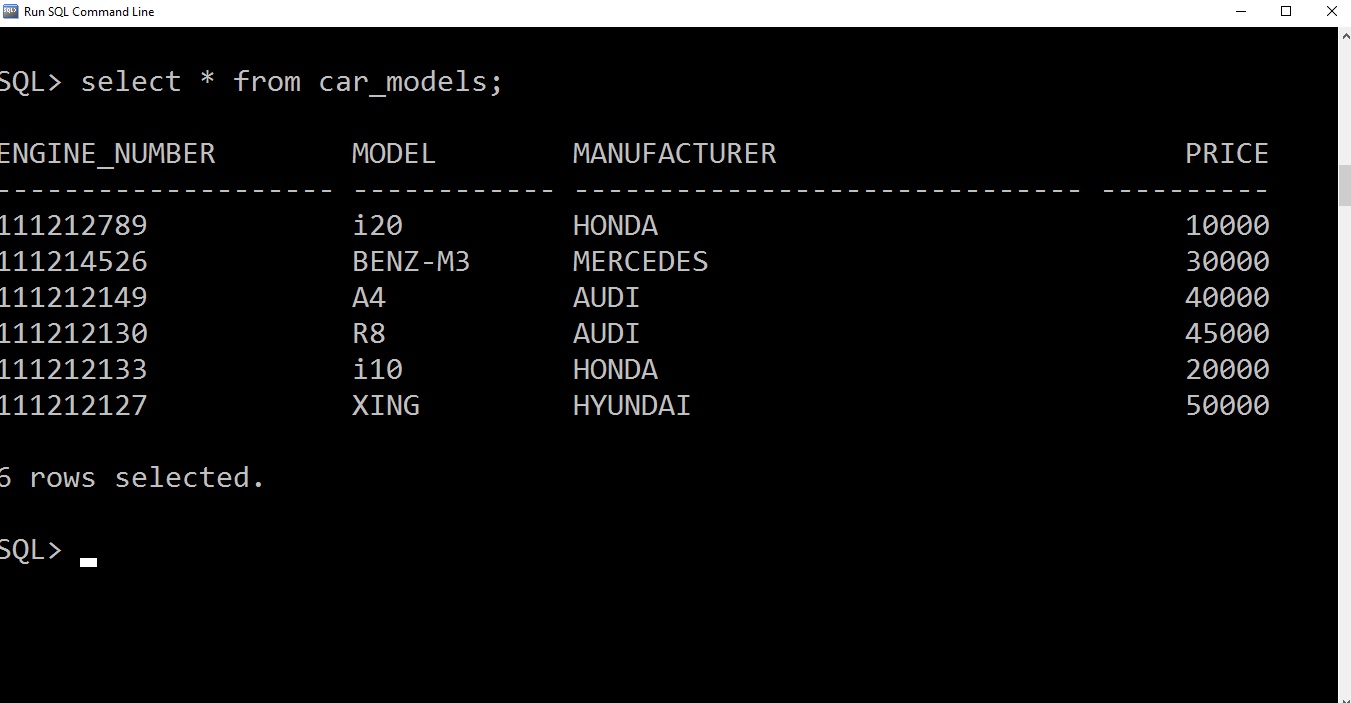


Fig: All car model data

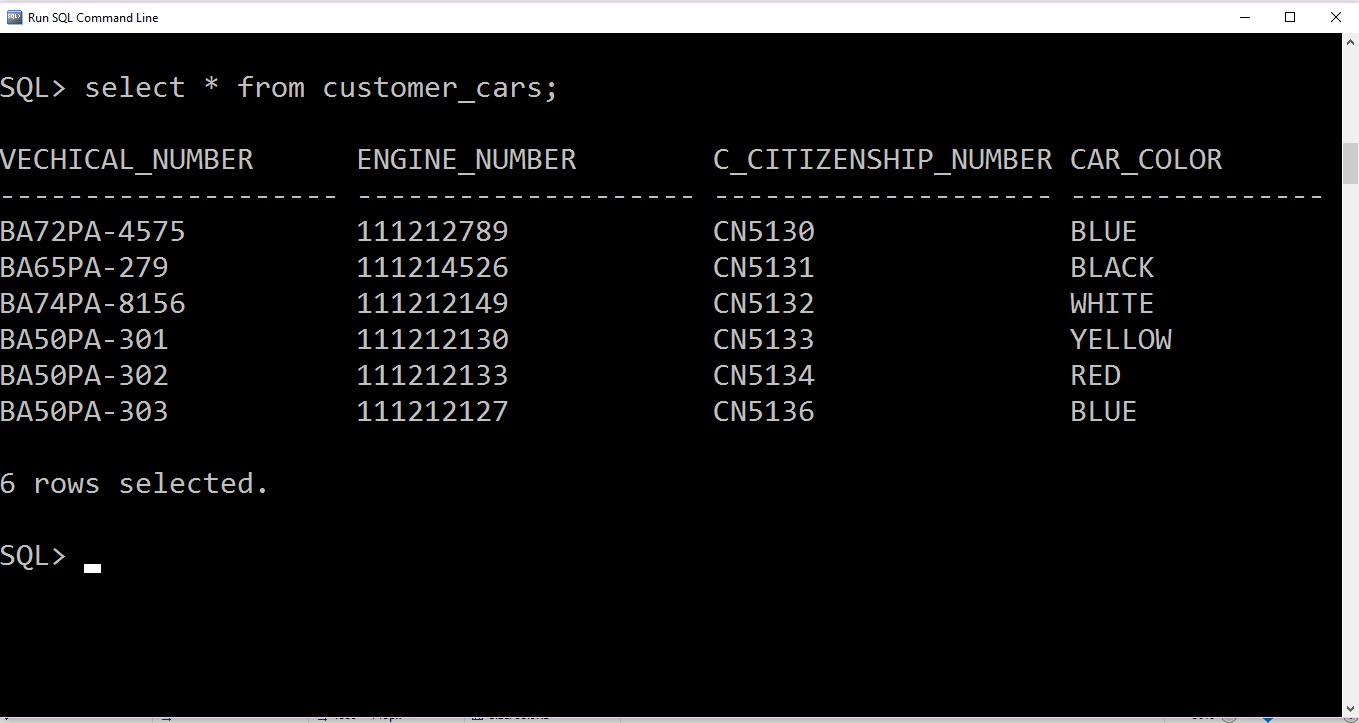


Fig: All customer car data

Listed below is some of the query.

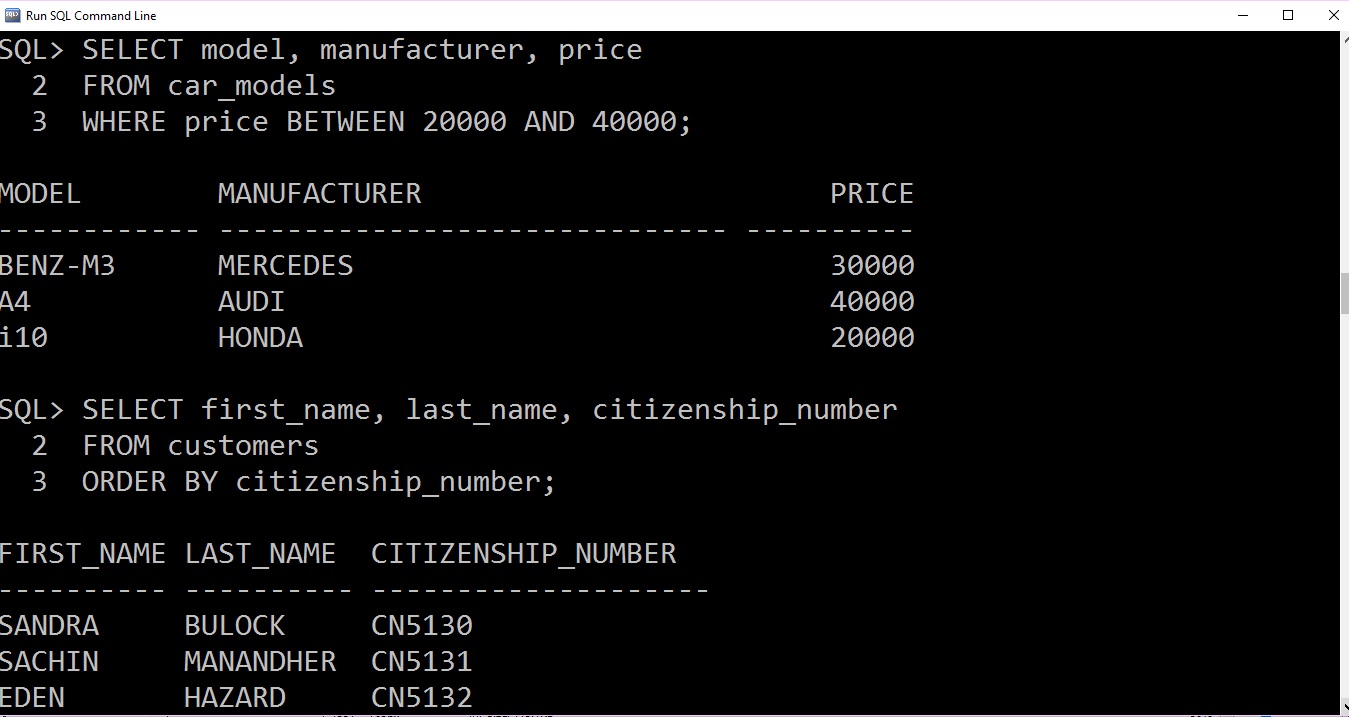


Fig: query 1

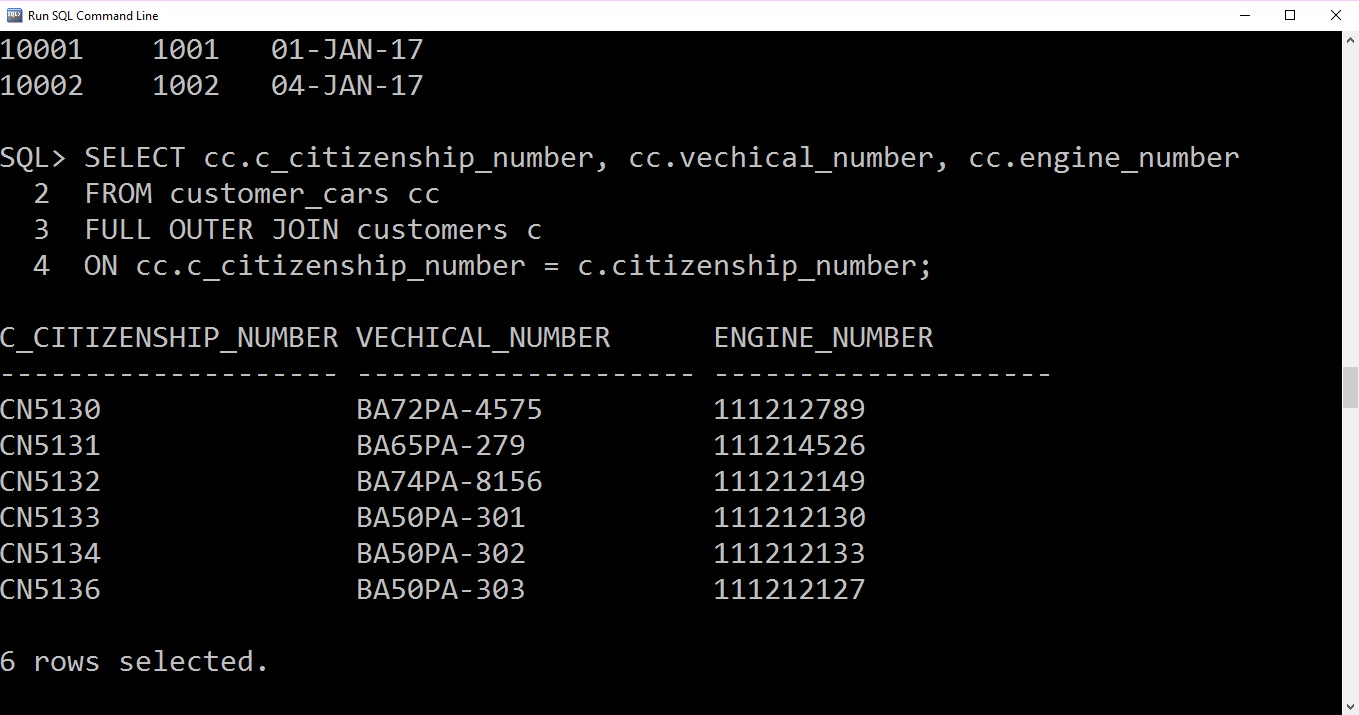


Fig: query2

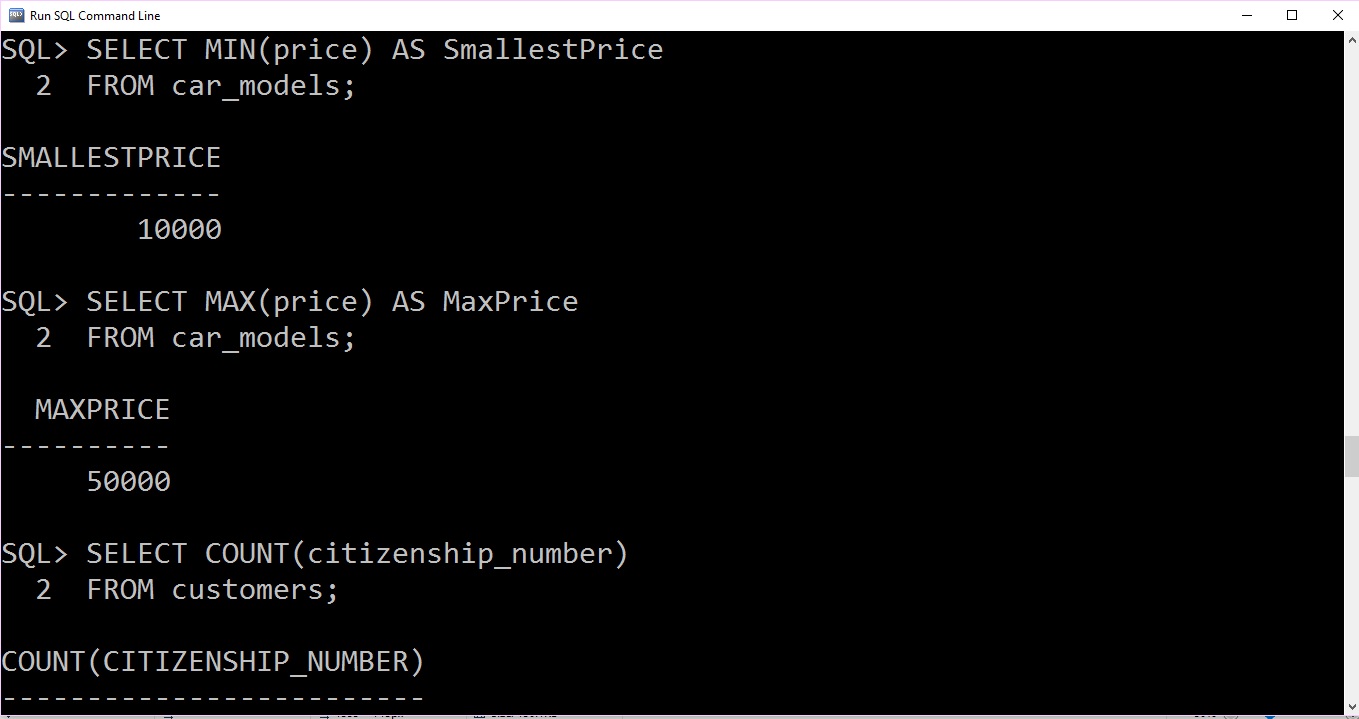


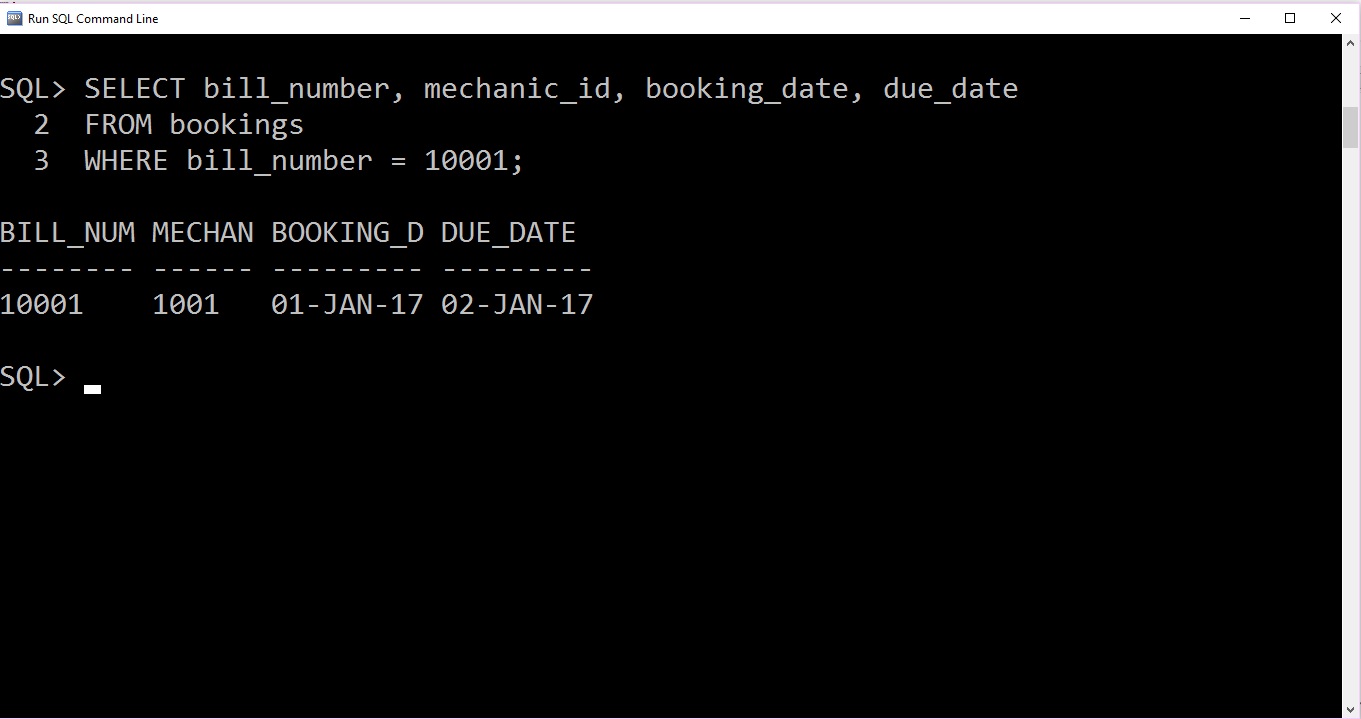
Fig: query3



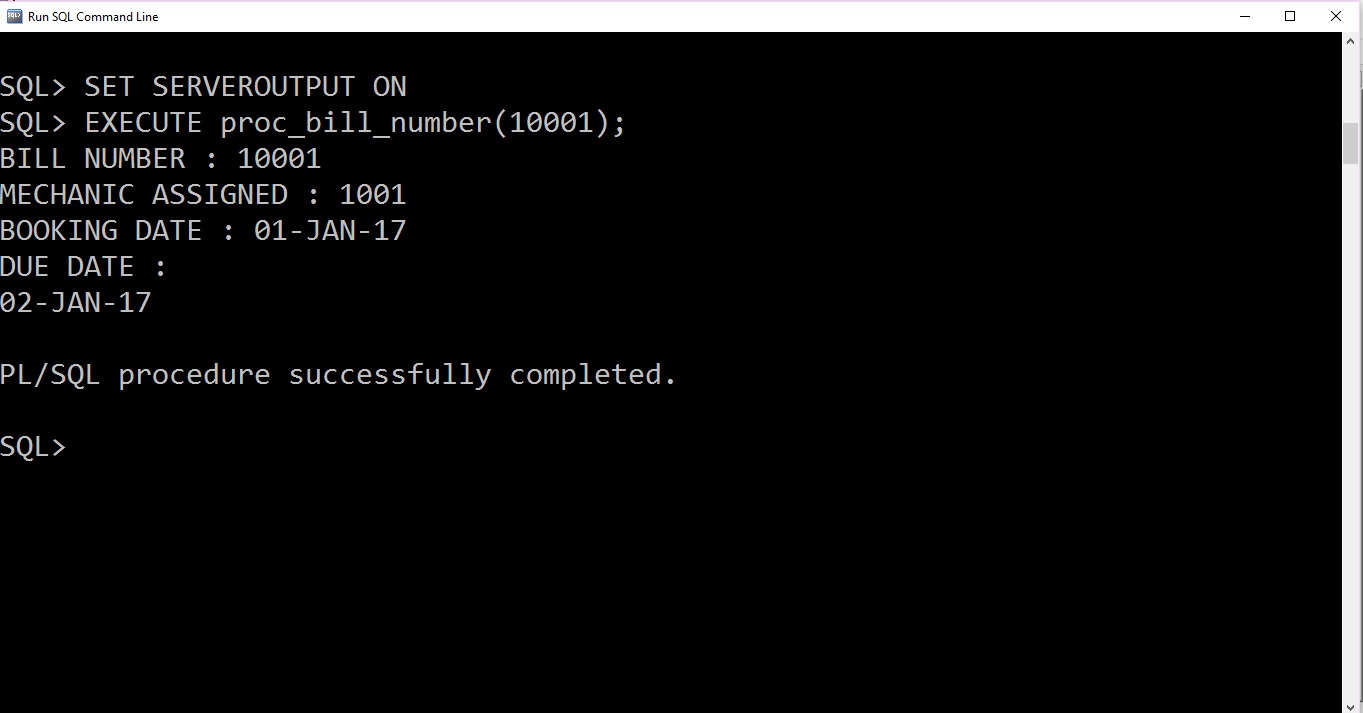
Fig: query4

# Annotate your screen shot to show functionality

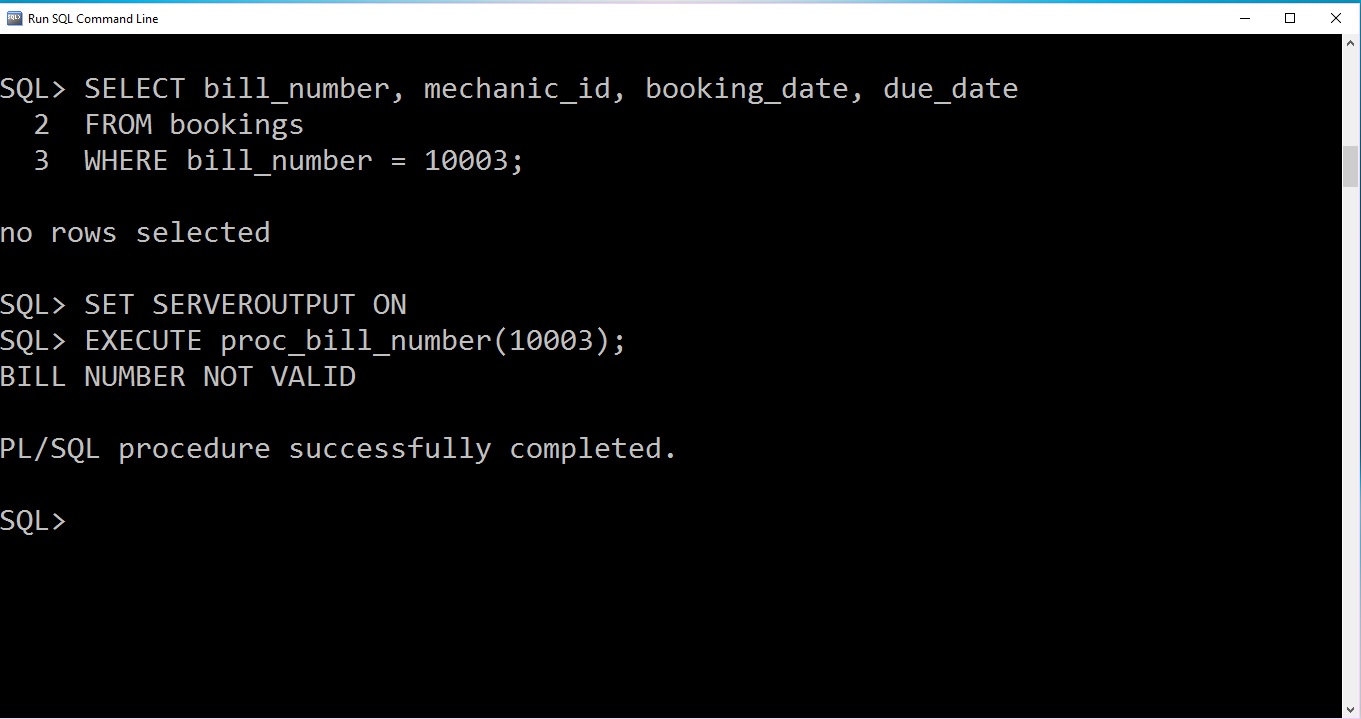
1. Testing for bill number

Valid condition query

After executing procedure proc\_bill\_number



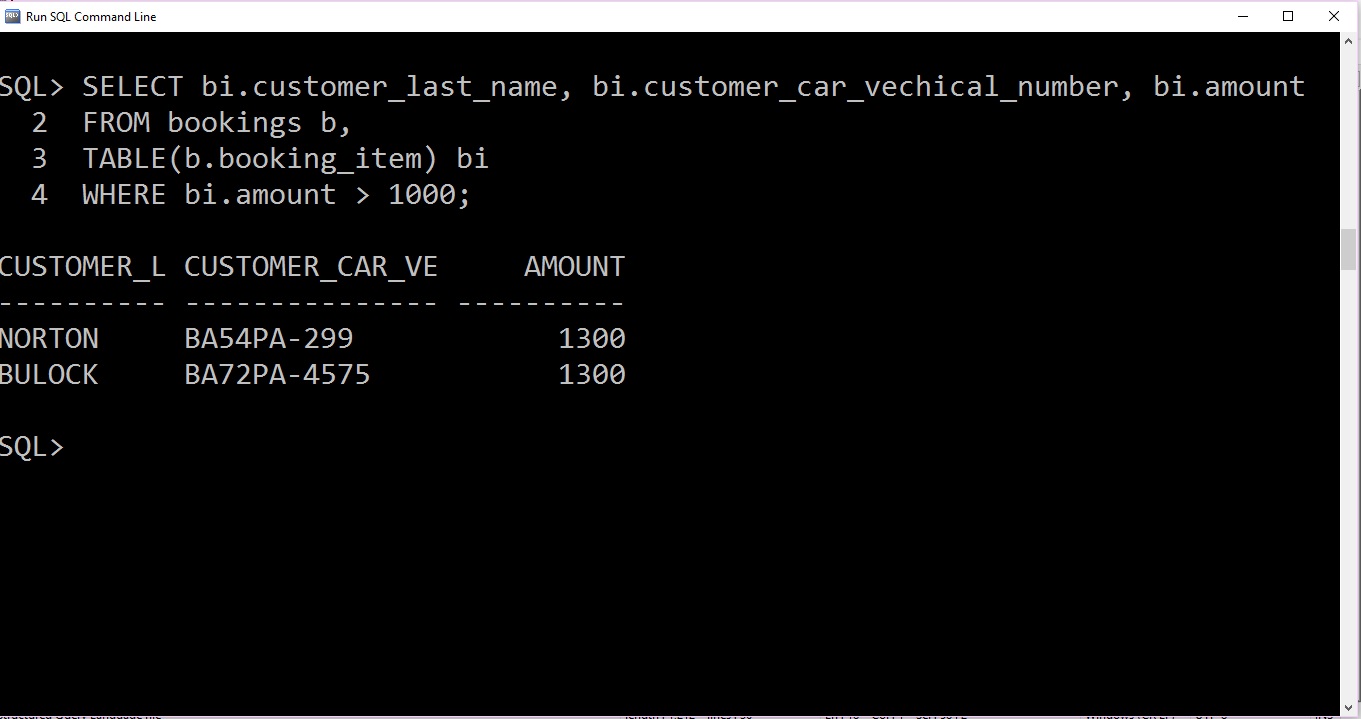
Query with invalid booking id and Procedure proc\_bill\_number with invalid booking id 10003



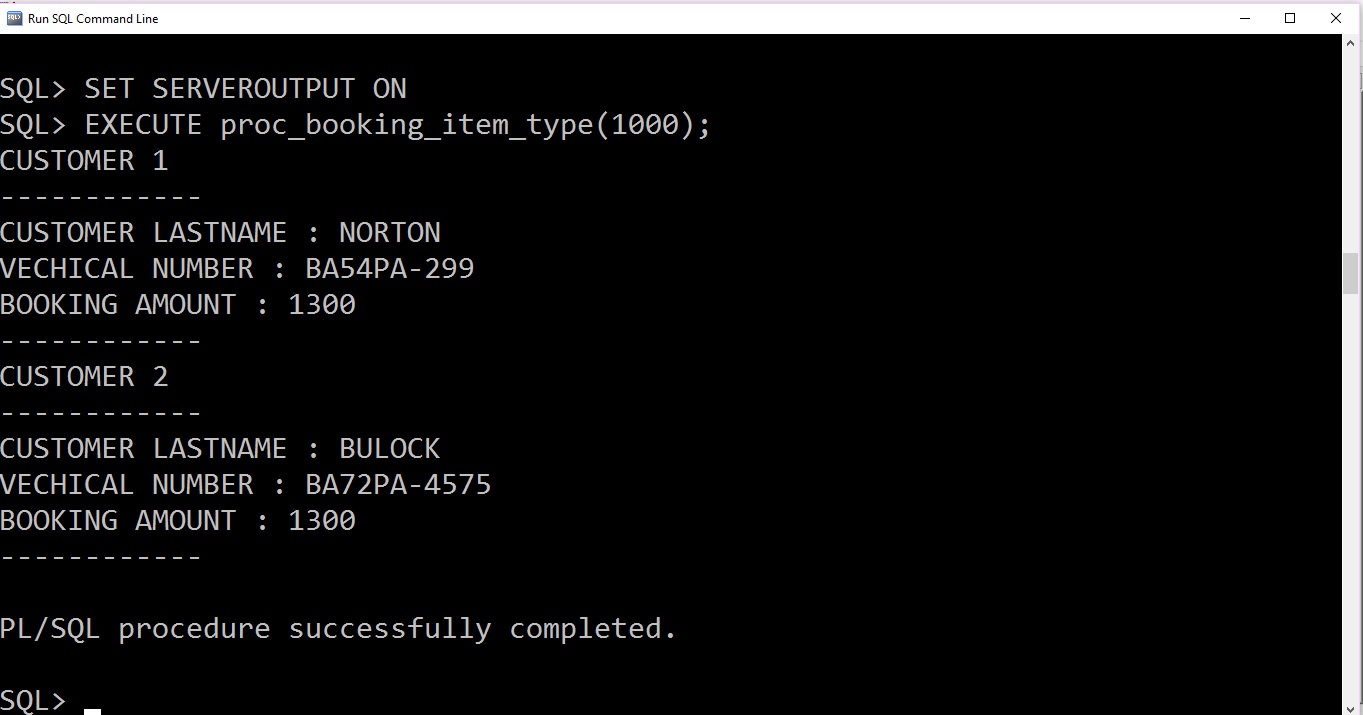
1. TEST CONDITION FOR BOOKING AMOUNT OF BOOKING\_ITEM\_TABLE

VALID CASE

BEFORE CREATING PROCEDURE proc\_booking\_item\_type

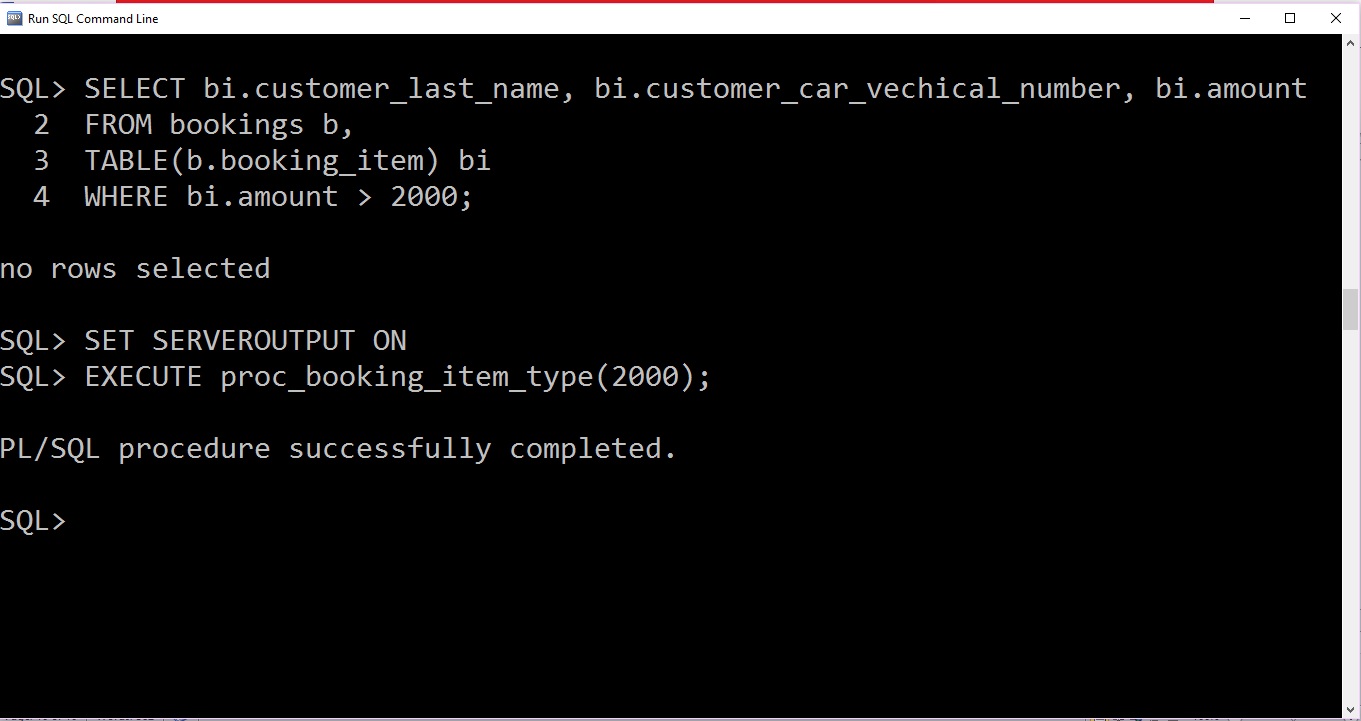


Using procedure proc\_booking\_item\_type



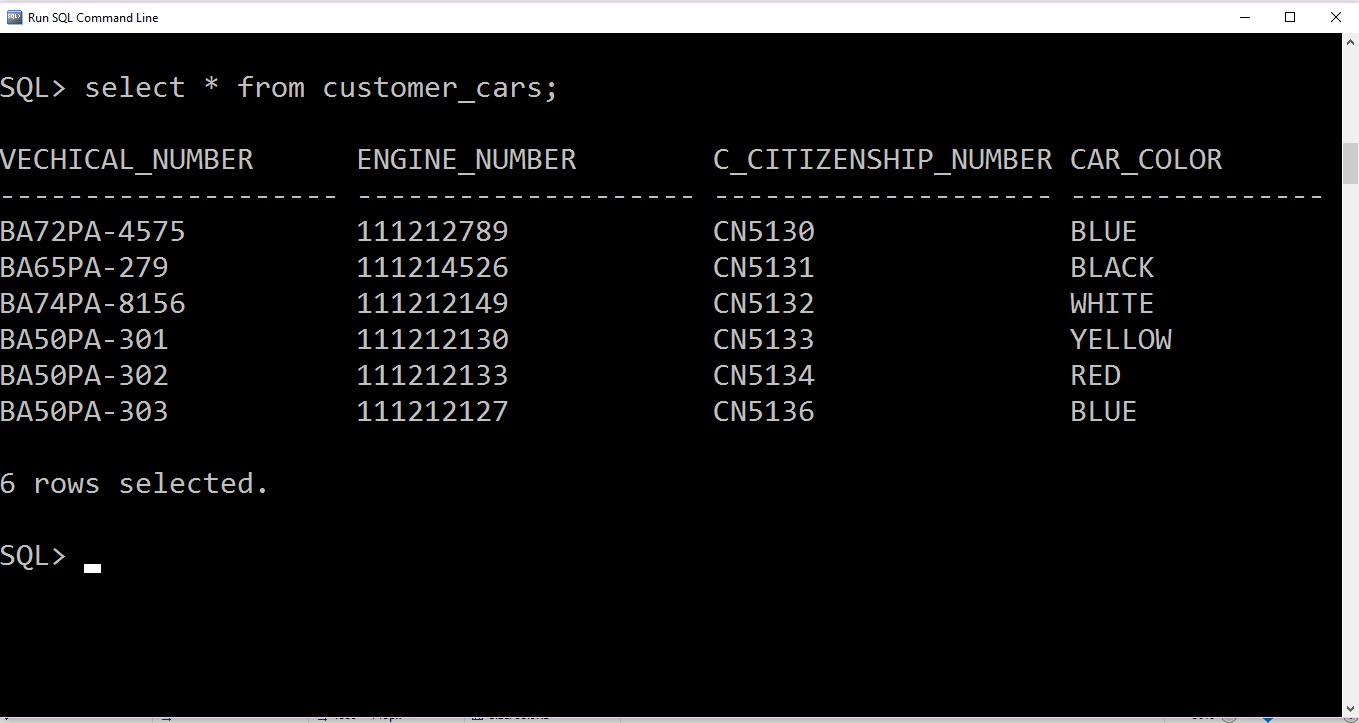
INVALID CASE

Query and procedure with invalid amount

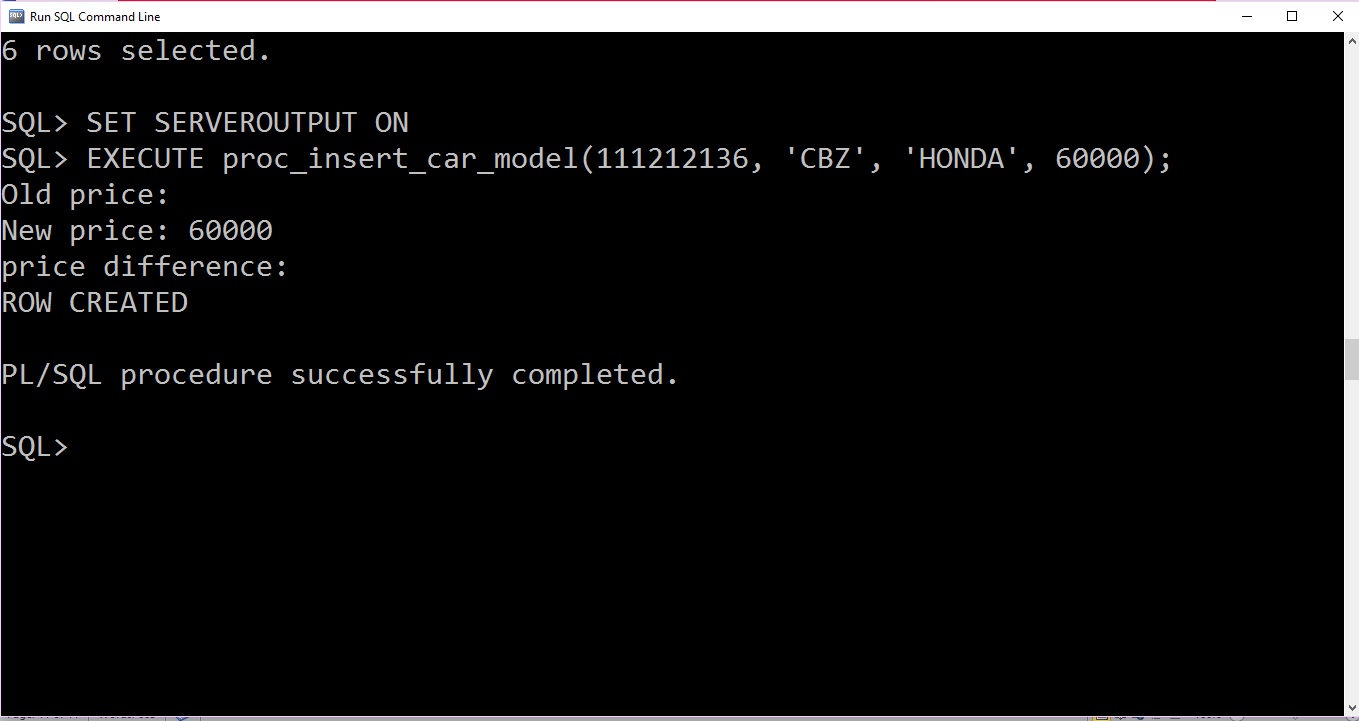


1. TEST CONDITION INSERTING VALUE INTO CUSTOMER CAR TABLE USING PROCEDURE

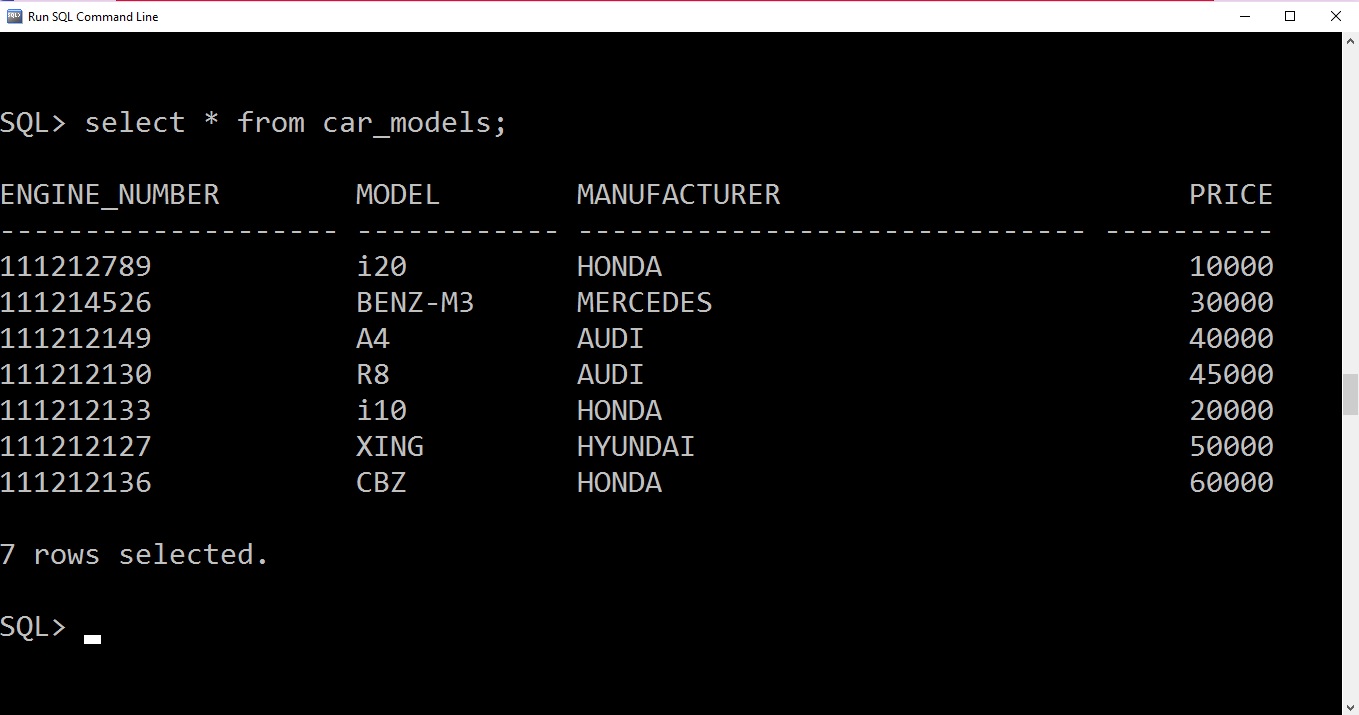
Before inserting value into customer car using procedure



Inserting using procedure



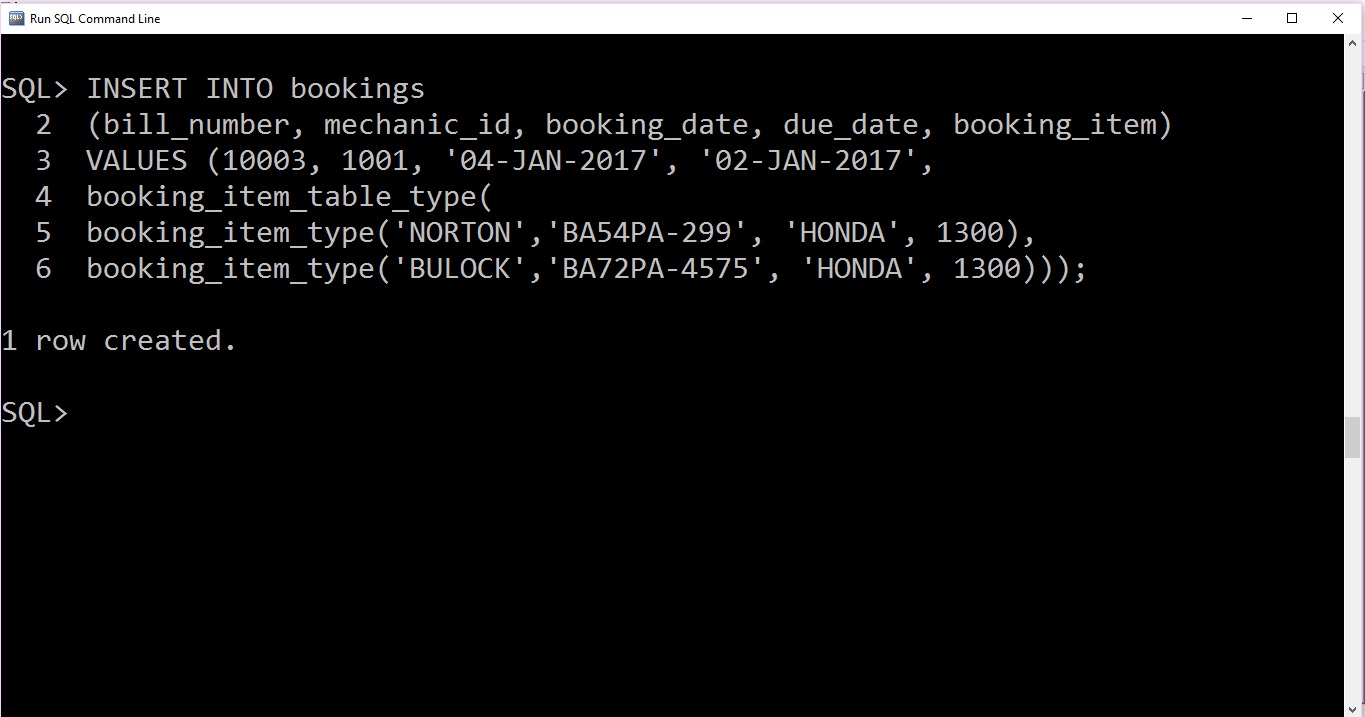
After executing proc\_insert\_car\_model



1. TESTING TRIGGER trig\_booking\_date\_ck, DUE DATE MUST NOT BE LESS THAN BOOKING DATE

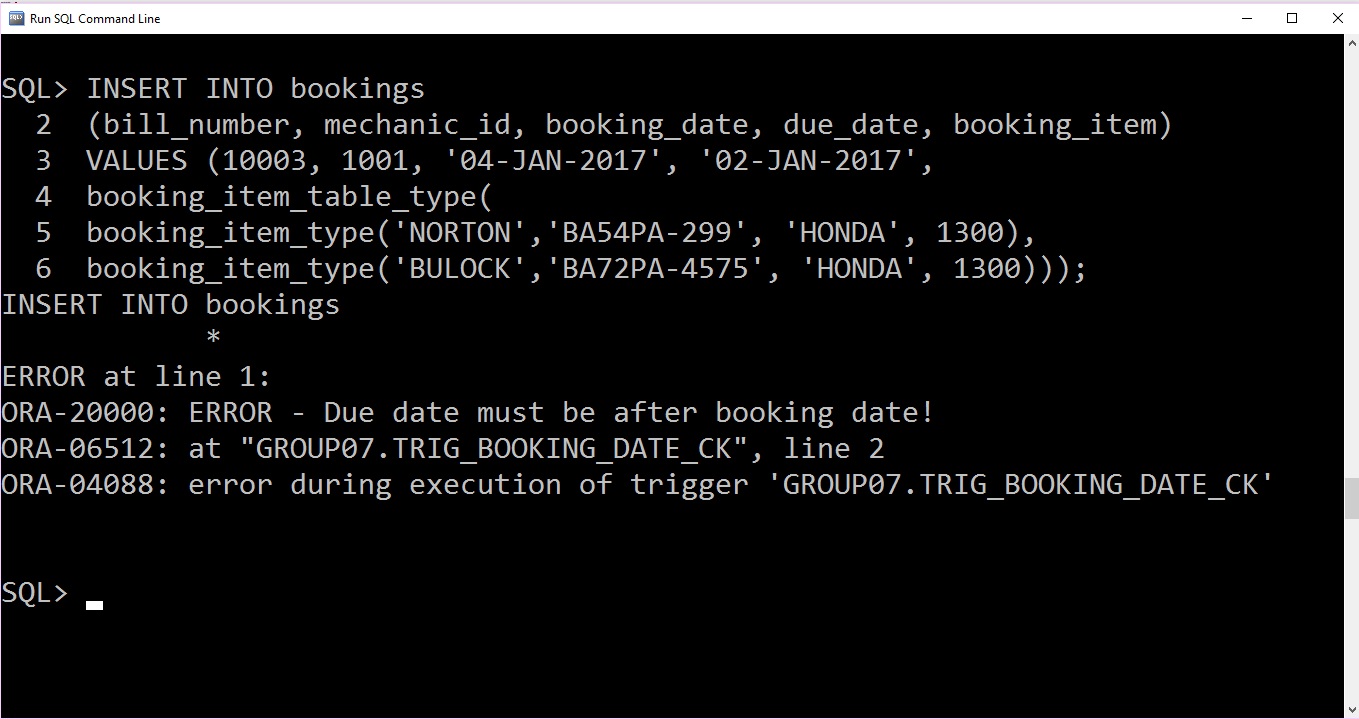
THIS TRIGGER IS FIRE WHENEVER USER TRY IT INSERT DUE DATE LESS THAN BOOKING DATE

Before creating a trigger



Whatever may be the due it will insert into the table, here due date is less than booking date.

After creating a trigger



Here prompt won’t let user to put due date less than booking date.

1. Testing Drop script

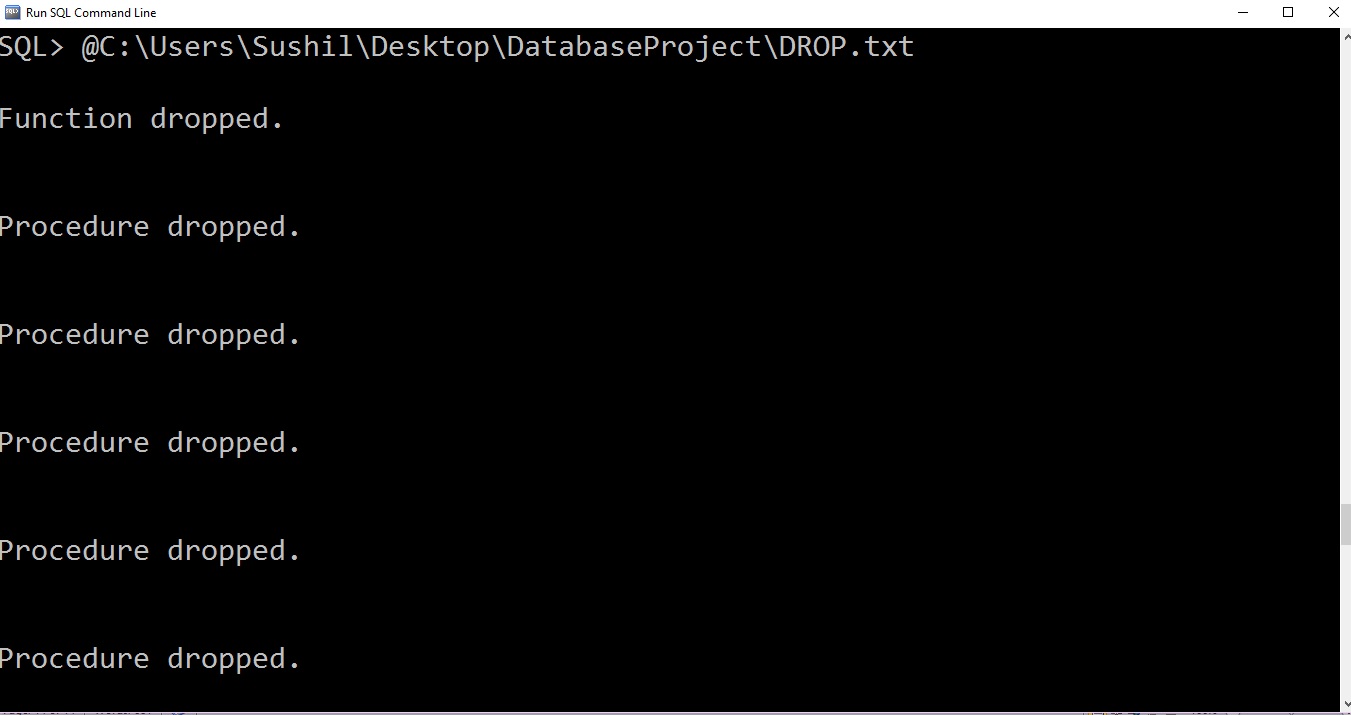


Fig: drop1

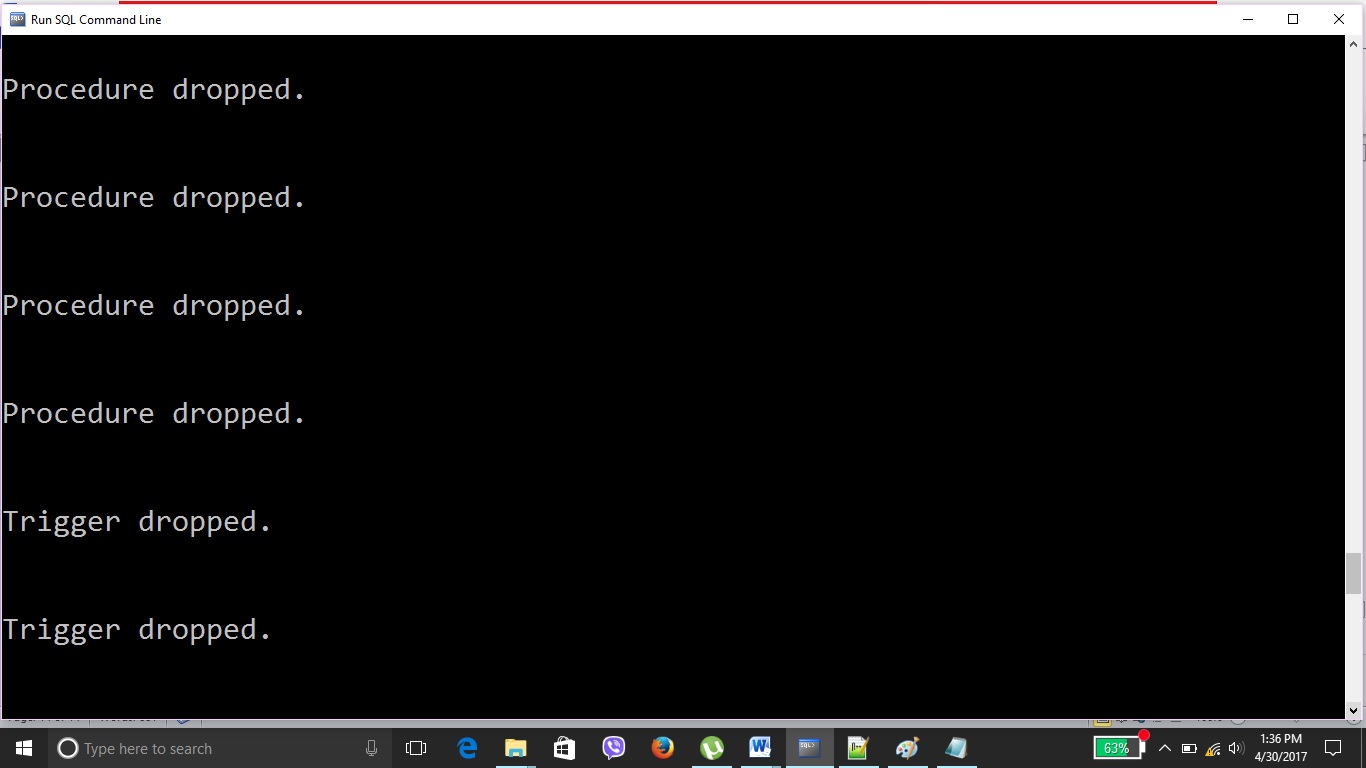


Fig: drop2



Fig: drop3



Fig: drop4

# COMMENTED CODE:

1. CREAT TABLE

--@C:\Users\Sushil\Desktop\DatabaseProject\Create\_Table.sql

-- Author: Sushil Pun

--15/03/2017

-- CREATING AN OBJECT TYPE NAME ADDRESS TYPE

CREATE OR REPLACE TYPE address\_type AS OBJECT (

street VARCHAR2(25),

city VARCHAR2(25),

country VARCHAR2(20) )

/

CREATE TABLE addresses OF address\_type;

-- CREATING A CUSTOMER TABLE

CREATE TABLE customers(

citizenship\_number VARCHAR2(20),

first\_name VARCHAR2(10),

last\_name VARCHAR2(10),

address REF address\_type SCOPE IS addresses);

-- CREATING A CAR MODEL TABLE

CREATE TABLE car\_models(

engine\_number VARCHAR2(20),

model VARCHAR2(12),

manufacturer VARCHAR2(30),

price NUMBER(8,2));

-- CREATING A CUSTOMER CAR TABLE

CREATE TABLE customer\_cars(

vechical\_number VARCHAR2(20),

engine\_number VARCHAR2(20),

c\_citizenship\_number VARCHAR2(20),

car\_color VARCHAR2(15));

CREATE TYPE specilization\_type AS OBJECT(

-- catagory for example engine, wheels or general

catagory VARCHAR2(10),

-- block coloumn specify mechanic works in which department or block

block NUMBER(6),

-- mechanic wage for doing specific task

charge NUMBER(10, 2));

/

CREATE TYPE specilization\_varray\_type AS VARRAY(50) OF specilization\_type;

/

-- -- CREATING A MECHANIC TABLE

CREATE TABLE mechanics (

mechanic\_id VARCHAR2(6),

first\_name VARCHAR2(10),

last\_name VARCHAR2(10),

address address\_type,

specilization specilization\_varray\_type);

-- CREATING A NESTED TABLE OF TYPE BOOKING\_ITEM\_TYPE

CREATE OR REPLACE TYPE booking\_item\_type AS OBJECT(

customer\_last\_name VARCHAR2(10),

customer\_car\_vechical\_number VARCHAR2(15),

manufacturer VARCHAR2(30),

amount NUMBER(10,2));

/

CREATE TYPE booking\_item\_table\_type AS TABLE OF booking\_item\_type;

/

-- CREATING A BOOKING TABLE

CREATE TABLE bookings(

bill\_number VARCHAR2(8),

mechanic\_id VARCHAR2(6),

booking\_date DATE,

due\_date DATE,

booking\_item booking\_item\_table\_type)

NESTED TABLE booking\_item STORE AS booking\_item\_table;

1. ALTER TABLE

--@C:\Users\Sushil\Desktop\DatabaseProject\Alter.sql

-- Author: Sushil Pun

--15/03/2017

--Creating PKs for table

ALTER TABLE customers

ADD CONSTRAINT pk\_customers

PRIMARY KEY (citizenship\_number);

ALTER TABLE car\_models

ADD CONSTRAINT pk\_engine\_number

PRIMARY KEY (engine\_number);

ALTER TABLE customer\_cars

ADD CONSTRAINT pk\_customer\_cars

PRIMARY KEY (vechical\_number, engine\_number, c\_citizenship\_number);

ALTER TABLE mechanics

ADD CONSTRAINT pk\_mechanics

PRIMARY KEY (mechanic\_id);

ALTER TABLE bookings

ADD CONSTRAINT pk\_bookings

PRIMARY KEY (bill\_number);

--Creating FKs for table

ALTER TABLE customer\_cars

ADD CONSTRAINT fk\_cc\_customers

FOREIGN KEY (C\_citizenship\_number)

REFERENCES customers(citizenship\_number);

ALTER TABLE customer\_cars

ADD CONSTRAINT fk\_cc\_car\_models

FOREIGN KEY (engine\_number)

REFERENCES car\_models(engine\_number);

ALTER TABLE bookings

ADD CONSTRAINT fk\_b\_mechanics

FOREIGN KEY (mechanic\_id)

REFERENCES mechanics(mechanic\_id);

1. INSERT INTO TABLE

--@C:\Users\Sushil\Desktop\DatabaseProject\INSERT.sql

-- Author: Sushil Pun

--15/03/2017

-- INSERTING VALUE INTO ADDRESSES TABLE

INSERT INTO addresses (street, city, country)

VALUES ('221 BAKER STREET', 'LONDON', 'UK');

INSERT INTO addresses (street, city, country)

VALUES ('641 FULHAM ROAD', 'LONDON', 'UK');

INSERT INTO addresses (street, city, country)

VALUES ('544 42ND STREET', 'NEW YORK', 'USA');

INSERT INTO addresses (street, city, country)

VALUES ('456 AVENUE CHAMPS', 'PARIS', 'FRANCE');

INSERT INTO addresses (street, city, country)

VALUES ('562 HOCKEY STREET', 'TORONTO', 'CANADA');

-- INSERTING VALUE INTO CAR MODELS TABLE

INSERT INTO car\_models (engine\_number, model, manufacturer, price)

VALUES ('111212789', 'i20', 'HONDA', 10000);

INSERT INTO car\_models (engine\_number, model, manufacturer, price)

VALUES ('111214526', 'BENZ-M3', 'MERCEDES', 30000);

INSERT INTO car\_models (engine\_number, model, manufacturer, price)

VALUES ('111212149', 'A4', 'AUDI', 40000);

INSERT INTO car\_models (engine\_number, model, manufacturer, price)

VALUES ('111212130', 'R8', 'AUDI', 45000);

INSERT INTO car\_models (engine\_number, model, manufacturer, price)

VALUES ('111212133', 'i10', 'HONDA', 20000);

INSERT INTO car\_models (engine\_number, model, manufacturer, price)

VALUES ('111212127', 'XING', 'HYUNDAI', 50000);

-- INSERTING VALUE INTO CUSTOMERS TABLE

INSERT INTO customers (citizenship\_number, first\_name, last\_name)

VALUES ('CN5130', 'SANDRA', 'BULOCK');

UPDATE customers SET address =

(select ref(a) FROM addresses a

WHERE a.street='641 FULHAM ROAD')

WHERE citizenship\_number = 'CN5130';

INSERT INTO customers (citizenship\_number, first\_name, last\_name)

VALUES ('CN5131', 'SACHIN', 'MANANDHER');

UPDATE customers SET address =

(select ref(a) FROM addresses a

WHERE a.street='544 42ND STREET')

WHERE citizenship\_number = 'CN5131';

INSERT INTO customers (citizenship\_number, first\_name, last\_name)

VALUES ('CN5132', 'EDEN', 'HAZARD');

UPDATE customers SET address =

(select ref(a) FROM addresses a

WHERE a.street='221 BAKER STREET')

WHERE citizenship\_number = 'CN5132';

INSERT INTO customers (citizenship\_number, first\_name, last\_name)

VALUES ('CN5133', 'FERNANDO', 'SUCRO');

UPDATE customers SET address =

(select ref(a) FROM addresses a

WHERE a.street='456 AVENUE CHAMPS')

WHERE citizenship\_number = 'CN5133';

INSERT INTO customers (citizenship\_number, first\_name, last\_name)

VALUES ('CN5134', 'GRAHAM', 'NORTON');

UPDATE customers SET address =

(select ref(a) FROM addresses a

WHERE a.street='221 BAKER STREET')

WHERE citizenship\_number = 'CN5134';

INSERT INTO customers (citizenship\_number, first\_name, last\_name)

VALUES ('CN5136', 'FRANK', 'JUNIOR');

UPDATE customers SET address =

(select ref(a) FROM addresses a

WHERE a.street='456 AVENUE CHAMPS')

WHERE citizenship\_number = 'CN5136';

-- INSERTING VALUE INTO CUSTOMER CARS TABLE

INSERT INTO customer\_cars (vechical\_number, engine\_number, c\_citizenship\_number, car\_color)

VALUES ('BA72PA-4575', '111212789', 'CN5130', 'BLUE');

INSERT INTO customer\_cars (vechical\_number, engine\_number, c\_citizenship\_number, car\_color)

VALUES ('BA65PA-279', '111214526', 'CN5131', 'BLACK');

INSERT INTO customer\_cars (vechical\_number, engine\_number, c\_citizenship\_number, car\_color)

VALUES ('BA74PA-8156', '111212149', 'CN5132', 'WHITE');

INSERT INTO customer\_cars (vechical\_number, engine\_number, c\_citizenship\_number, car\_color)

VALUES ('BA50PA-301', '111212130', 'CN5133', 'YELLOW');

INSERT INTO customer\_cars (vechical\_number, engine\_number, c\_citizenship\_number, car\_color)

VALUES ('BA50PA-302', '111212133', 'CN5134', 'RED');

INSERT INTO customer\_cars (vechical\_number, engine\_number, c\_citizenship\_number, car\_color)

VALUES ('BA50PA-303', '111212127', 'CN5136', 'BLUE');

-- INSERTING VALUE INTO MECHANICS TABLE

INSERT INTO mechanics (mechanic\_id, first\_name, last\_name, address, specilization)

VALUES ('1001', 'SUJIN', 'MANANDHER', address\_type('221 BAKER STREET', 'LONDON', 'UK'),

specilization\_varray\_type (specilization\_type ('ENGINE', 2, 1000)));

INSERT INTO mechanics (mechanic\_id, first\_name, last\_name, address, specilization)

VALUES ('1002', 'DEEPAK', 'BASNET', address\_type('562 HOCKEY STREET', 'TORONTO', 'CANADA'),

specilization\_varray\_type (specilization\_type ('WHEEL', 1, 200)));

INSERT INTO mechanics (mechanic\_id, first\_name, last\_name, address, specilization)

VALUES ('1003', 'SURAJ', 'KHANAL', address\_type('456 AVENUE CHAMPS', 'PARIS', 'FRANCE'),

specilization\_varray\_type (specilization\_type ('GENERAL', 3, 100)));

-- INSERTING VALUE INTO BOOKING TABLE

INSERT INTO bookings

(bill\_number, mechanic\_id, booking\_date, due\_date, booking\_item)

VALUES ('10001', '1001', '01-JAN-2017', '02-JAN-2017',

booking\_item\_table\_type(

booking\_item\_type('NORTON','BA54PA-299', 'HONDA', 1300),

booking\_item\_type('BULOCK','BA72PA-4575', 'HONDA', 1300)));

INSERT INTO bookings

(bill\_number, mechanic\_id, booking\_date, due\_date, booking\_item)

VALUES ('10002', '1002', '04-JAN-2017', '06-JAN-2017',

booking\_item\_table\_type(

booking\_item\_type('MANANDHER','BA65PA-279', 'MERCEDES', 1000),

booking\_item\_type('HAZARD','BA74PA-8156', 'AUDI', 1000)));

SHOW ERRORS;

1. FUNCTION AND PROCEDURE

--@C:\Users\Sushil\Desktop\DatabaseProject\car\_models\_price.sql

-- Author: Sushil Pun

--15/03/2017

/\* FUNCTION PARAMETER ACCEPT PRICE FROM USER AND COUNT THE CAR WHICH ARE MORE COSTLY THAN THE USER INPUT

AND RETURN THE NUMBER OF CAR TO THE PROCEDURE, WHICH WILL DISPLAY THE MESSAGE

\*/

CREATE OR REPLACE FUNCTION func\_car\_models\_price(in\_price car\_models.price%TYPE)

RETURN NUMBER IS

vn\_car\_price NUMBER(8);

BEGIN

SELECT COUNT(\*)

INTO vn\_car\_price

FROM car\_models

WHERE price > in\_price;

-- RETURNS THE NUMBER OF CAR COUNTED TO PROCEDURE

RETURN vn\_car\_price;

END;

/

SHOW ERRORS;

-- CREATING PROCEDURE WHICH ACCEPT A PRICE OF TYPE (CAR\_MODELS TABLE PRICE COLOUMN)

CREATE OR REPLACE PROCEDURE proc\_car\_models\_price(in\_price car\_models.price%TYPE) IS

vn\_no\_of\_cars NUMBER(5) ;

BEGIN

-- calling function func\_car\_models\_price and storing it in vn\_no\_of\_cars

vn\_no\_of\_cars := func\_car\_models\_price(in\_price);

DBMS\_OUTPUT.PUT\_LINE ('The Number of Car greater than price '|| in\_price || ' are : '|| vn\_no\_of\_cars );

END proc\_car\_models\_price;

/

SHOW ERRORS;

/\*

SET SERVEROUTPUT ON

EXECUTE proc\_car\_models\_price(20000);

SHOW ERRORS;

\*/

-- PROCEDURE TO CHECK WHETHER THE GIVEN PERSON ID IS A VALID CUSTOMER OR NOT

CREATE OR REPLACE PROCEDURE proc\_customer\_citizenship(in\_citizenship\_number customers.citizenship\_number%TYPE)IS

vc\_searched\_citizen\_number VARCHAR2(20);

vc\_citizen\_first\_name VARCHAR2(20);

vc\_citizen\_last\_name VARCHAR2(20);

BEGIN

SELECT citizenship\_number, first\_name, last\_name

INTO vc\_searched\_citizen\_number, vc\_citizen\_first\_name, vc\_citizen\_last\_name

FROM customers

WHERE citizenship\_number = in\_citizenship\_number;

IF vc\_searched\_citizen\_number = in\_citizenship\_number THEN

DBMS\_OUTPUT.PUT\_LINE('CITIZENSHIP NUMBER : '|| in\_citizenship\_number || CHR(10)||

'FirstName : ' || vc\_citizen\_first\_name || CHR(10) ||

'LastName : ' || vc\_citizen\_last\_name);

END IF;

-- EXCEPTION IS THROWN WHEN THE GIVEN PERSON CITIZENSHIP NUMBER IS NOT FOUND

EXCEPTION

WHEN no\_data\_found THEN

DBMS\_OUTPUT.PUT\_LINE('CITIZENSHIP NOT FOUND');

END;

/

SHOW ERRORS;

-- SET SERVEROUTPUT ON

-- EXECUTE proc\_customer\_citizenship('CN5129');

-- DROP

--DROP PROCEDURE proc\_customer\_citizenship;

-- PROCEDURE TO CHECK WHETHER THE GIVEN BILL NUMBER IS A VALID OR NOT

CREATE OR REPLACE PROCEDURE proc\_bill\_number(in\_bill\_number bookings.bill\_number%TYPE)IS

vn\_bill\_number VARCHAR2(8);

vn\_mechanic\_id VARCHAR2(6);

vd\_booking\_date DATE;

vd\_due\_date DATE;

BEGIN

SELECT bill\_number, mechanic\_id, booking\_date, due\_date

INTO vn\_bill\_number, vn\_mechanic\_id, vd\_booking\_date, vd\_due\_date

FROM bookings

WHERE bill\_number = in\_bill\_number;

IF vn\_bill\_number = in\_bill\_number THEN

DBMS\_OUTPUT.PUT\_LINE('BILL NUMBER : '|| in\_bill\_number || CHR(10) ||

'MECHANIC ASSIGNED : '|| vn\_mechanic\_id || CHR(10) ||

'BOOKING DATE : '|| vd\_booking\_date || CHR(10) ||

'DUE DATE : '|| vd\_due\_date);

END IF;

-- EXCEPTION IS THROWN WHEN THE GIVEN PERSON CITIZENSHIP NUMBER IS NOT FOUND

EXCEPTION

WHEN no\_data\_found THEN

DBMS\_OUTPUT.PUT\_LINE('BILL NUMBER NOT VALID');

END;

/

SHOW ERRORS;

-- SET SERVEROUTPUT ON

-- EXECUTE proc\_bill\_number('500');

SHOW ERRORS;

-- procedure using cursor to find out the booking item details which is a nested table inside bookings table

CREATE OR REPLACE PROCEDURE proc\_booking\_item\_type(in\_amount NUMBER) IS

vc\_name VARCHAR2(25);

vc\_vechical\_number VARCHAR2(15);

vn\_booking\_cost NUMBER(10,2);

CURSOR C1 IS SELECT bi.customer\_last\_name, bi.customer\_car\_vechical\_number, bi.amount INTO vc\_name, vc\_vechical\_number, vn\_booking\_cost

FROM bookings b,

TABLE(b.booking\_item) bi

WHERE bi.amount > in\_amount;

BEGIN

IF NOT C1%isopen THEN

OPEN C1;

END IF;

LOOP

-- reference https://ss64.com/oraplsql/cursor\_fetch\_close.html

FETCH C1 INTO vc\_name, vc\_vechical\_number, vn\_booking\_cost;

EXIT WHEN C1%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('CUSTOMER '|| C1%rowcount);

DBMS\_OUTPUT.PUT\_LINE('------------');

DBMS\_OUTPUT.PUT\_LINE('CUSTOMER LASTNAME : '||vc\_name|| CHR(10)||

'VECHICAL NUMBER : '|| vc\_vechical\_number ||CHR(10)||

'BOOKING AMOUNT : '|| vn\_booking\_cost);

DBMS\_OUTPUT.PUT\_LINE('------------');

END LOOP;

CLOSE C1;

END;

/

SHOW ERROR;

-- SET SERVEROUTPUT ON

-- EXECUTE proc\_booking\_item\_type(2000);

/\* DELETING A ROW FROM A CAR MODEL TABLE USING A PROCEDURE \*/

SET SERVEROUTPUT ON

CREATE OR REPLACE PROCEDURE proc\_delete\_car\_model(in\_model IN car\_models.model%TYPE )

IS

BEGIN

DELETE FROM car\_models

WHERE model = in\_model;

-- IF ABOVE STATEMENT IS TRUE IT WILL DISPLAY ROW DELETED IF NOT IT WILL GO TO ELSE STATEMENT

IF SQL%FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('ROW DELETED');

ELSE

DBMS\_OUTPUT.PUT\_LINE('NO SUCH MODEL AVAILABLE');

END IF;

COMMIT;

END;

/

SHOW ERRORS;

SET SERVEROUTPUT ON

--EXECUTE proc\_delete\_car\_model('CBZ');

/\* EMPTY PARAMETER PROCEDURE USING FOR LOOP TO DISPLAY ALL THE MECHANICS

FIRSTNAME AVAILABLE IN THE TABLE MECHANICS.

\*/

CREATE OR REPLACE PROCEDURE proc\_for\_loop IS

vn\_counter NUMBER := 0;

BEGIN

DBMS\_OUTPUT.put\_line ('Procedure For Loop Starts....');

FOR rec IN (SELECT \* FROM mechanics)

LOOP

vn\_counter := vn\_counter + 1;

DBMS\_OUTPUT.put\_line ('Record ' || vn\_counter || ' mechanics is- ' || rec.first\_name);

END LOOP;

DBMS\_OUTPUT.put\_line ('Procedure For Loop Completed!');

END;

/

SHOW ERRORS;

-- SET SEVEROUTPUT ON

-- EXECUTE proc\_for\_loop;

/\* INSERTING A DATA USING A PROCEDURE \*/

CREATE OR REPLACE PROCEDURE proc\_insert\_car\_model(

in\_engine\_number IN car\_models.engine\_number%TYPE,

in\_model IN car\_models.model%TYPE,

in\_manufacturer IN car\_models.manufacturer%TYPE,

in\_price IN car\_models.price%TYPE)

IS

BEGIN

INSERT INTO car\_models(engine\_number, model, manufacturer, price)

VALUES (in\_engine\_number, in\_model, in\_manufacturer, in\_price);

-- IF ABOVE STATEMENT IS TRUE IT WILL DISPLAY ROW CREATED IF NOT IT WILL GO TO ELSE STATEMENT

IF SQL%FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('ROW CREATED');

ELSE

DBMS\_OUTPUT.PUT\_LINE('INPUT VALUE MAY BE WRONG');

END IF;

COMMIT;

END;

/

SHOW ERRORS;

SET SERVEROUTPUT ON

--EXECUTE proc\_insert\_car\_model('111212136', 'CBZ', 'HONDA', 60000);

/\* THIS IS A TEST.. INSERTING A DATA USING A PROCEDURE

IN THE TABLE CUSTOMER CAR WHERE

ENGINE NUMBER AND CITIZENSHIP NUMBER ARE FOREIGN KEY, IF TRY IT FAIL'S DUE TO INTEGRITY CONSTRAINTS

SO MUST NOT EXECUTE THE PROCEDURE

\*/

CREATE OR REPLACE PROCEDURE proc\_insert\_customer\_car(

in\_vechical\_number IN customer\_cars.vechical\_number%TYPE,

in\_engine\_number IN customer\_cars.engine\_number%TYPE,

in\_citizen\_number IN customer\_cars.c\_citizenship\_number%TYPE,

in\_car\_color IN customer\_cars.car\_color%TYPE)

IS

BEGIN

INSERT INTO customer\_cars (vechical\_number, engine\_number, c\_citizenship\_number, car\_color)

VALUES (in\_vechical\_number, in\_engine\_number, in\_citizen\_number, in\_car\_color);

COMMIT;

END;

/

SHOW ERRORS;

--EXECUTE proc\_insert\_customer\_car('BA50PA-299', '111212136', 'CN5129', 'RED');

/\* USING THE SWITCH CASE STATEMENT PROCEDURE OUTPUT THE CATEGORY OF CAR

IF CAR IS BELOW 10000 THEN IT'S LOW COST CAR

IF-ELSE CAR IS BELOW 40000 IT'S AVERAGE COST CAR

ELSE THE CAR IS HIGH COST CAR.

\*/

-- procedure takes a parameter model and matches it with the model of car\_models table

-- evaluates price and show it's catagory

CREATE OR REPLACE PROCEDURE pro\_car\_category(in\_model car\_models.model%TYPE) IS

vn\_price NUMBER(10, 2);

vc\_manufacturer VARCHAR2(20);

vc\_model VARCHAR2(12);

BEGIN

SELECT manufacturer, model, price

INTO vc\_manufacturer, vc\_model, vn\_price

FROM car\_models

WHERE model = in\_model;

CASE

WHEN vn\_price < 10000 THEN DBMS\_OUTPUT.PUT\_LINE('LOW COST CAR');

WHEN vn\_price < 40000 THEN DBMS\_OUTPUT.PUT\_LINE('AVERAGE COST CAR');

ELSE DBMS\_OUTPUT.PUT\_LINE('HIGH COST CAR');

END CASE;

END;

/

SHOW ERRORS;

SET SERVEROUTPUT ON

EXECUTE pro\_car\_category('R8');

-- REFERENCE https://docs.oracle.com/cd/A97630\_01/appdev.920/a96624/04\_struc.htm

/\* THIS IS A IMPLICIT CURSOR USING THE WHILE LOOP

FIRST IT WILL SHOW THE RECORD OF EACH ROW OF CAR MODEL TABLE

THEN IT ADDS THE PRICE COLOUMN OF EACH RECORD UNTIL NO RECORD IS FOUND

FINALLY IT SHOW'S THE TOTAL PRICE OF CAR MODEL PRICE COLOUMN

\*/

SET SERVEROUTPUT ON

DECLARE

vc\_manufacturer VARCHAR2(20);

vc\_model VARCHAR2(15);

vn\_price NUMBER(10);

vn\_total\_price NUMBER(10) := 0;

CURSOR cursorValue IS

SELECT manufacturer, model, price

INTO vc\_manufacturer, vc\_model, vn\_price

FROM car\_models;

BEGIN

OPEN cursorValue;

FETCH cursorValue INTO vc\_manufacturer, vc\_model, vn\_price;

WHILE (cursorValue%FOUND) LOOP

DBMS\_OUTPUT.PUT\_LINE('VECHICAL '|| cursorValue%rowcount);

DBMS\_OUTPUT.PUT\_LINE('====================');

DBMS\_OUTPUT.PUT\_LINE('MANUFACTURER : '|| vc\_manufacturer|| CHR(10)||

'MODEL : '|| vc\_model || CHR(10) || 'PRICE : '|| vn\_price);

DBMS\_OUTPUT.PUT\_LINE('====================');

vn\_total\_price := vn\_total\_price + vn\_price;

FETCH cursorValue INTO vc\_manufacturer, vc\_model, vn\_price;

END LOOP;

CLOSE cursorValue;

DBMS\_OUTPUT.PUT\_LINE('THE WHOLE TOTAL PRICE OF THE CAR STORE IN DATABASE IS : '|| vn\_total\_price);

END;

/

SHOW ERRORS;

/\* USING THE SWITCH CASE STATEMENT PROCEDURE OUTPUT THE CATEGORY OF CAR

IF CAR IS BELOW 10000 THEN IT'S LOW COST CAR

IF-ELSE CAR IS BELOW 40000 IT'S AVERAGE COST CAR

ELSE THE CAR IS HIGH COST CAR.

\*/

SET SERVEROUTPUT ON

CREATE OR REPLACE PROCEDURE pro\_car\_category(in\_model car\_models.model%TYPE) IS

vn\_price NUMBER(10, 2);

vc\_manufacturer VARCHAR2(20);

vc\_model VARCHAR2(12);

BEGIN

SELECT manufacturer, model, price

INTO vc\_manufacturer, vc\_model, vn\_price

FROM car\_models

WHERE model = in\_model;

CASE

WHEN vn\_price < 10000 THEN DBMS\_OUTPUT.PUT\_LINE('LOW COST CAR');

WHEN vn\_price < 40000 THEN DBMS\_OUTPUT.PUT\_LINE('AVERAGE COST CAR');

ELSE DBMS\_OUTPUT.PUT\_LINE('HIGH COST CAR');

END CASE;

END;

/

SHOW ERRORS;

EXECUTE pro\_car\_category('R8');

-- REFERENCE https://docs.oracle.com/cd/A97630\_01/appdev.920/a96624/04\_struc.htm

SET SERVEROUTPUT ON

/\* THIS TRIGGER IS FIRED WHENEVER THE BOOKING DATE IS GREATER THAN THE DUE DATE OF BOOKINGS TABLE.

\*/

CREATE OR REPLACE TRIGGER trig\_booking\_date\_ck

BEFORE INSERT OR UPDATE OF booking\_date, due\_date ON bookings

FOR EACH ROW

WHEN

(NEW.booking\_date > NEW.due\_date)

BEGIN

RAISE\_APPLICATION\_ERROR(-20000, 'ERROR - Due date must be after booking date!');

END;

/

INSERT INTO bookings

(bill\_number, mechanic\_id, booking\_date, due\_date, booking\_item)

VALUES ('10003', '1001', '04-JAN-2017', '02-JAN-2017',

booking\_item\_table\_type(

booking\_item\_type('NORTON','BA54PA-299', 'HONDA', 1300),

booking\_item\_type('BULOCK','BA72PA-4575', 'HONDA', 1300)));

ROLLBACK;

/\* TRIGGER IS FIRE WHEN USER INSERT OR UPDATE VALUE INTO CAR MODEL TABLE

DELETE OPERATION CANNOT BE DONE BECAUSE IT'S PK IS FK IN ANOTHER TABLE

\*/

CREATE OR REPLACE TRIGGER trig\_car\_price

BEFORE DELETE OR INSERT OR UPDATE ON car\_models

-- reference https://docs.oracle.com/cloud/latest/db112/LNPLS/triggers.htm#LNPLS99955

FOR EACH ROW

WHEN (NEW.engine\_number > 0)

DECLARE

price\_difference NUMBER;

BEGIN

price\_difference := 0;

price\_difference := :NEW.price - :OLD.price;

dbms\_output.put\_line('Old price: ' || :OLD.price);

dbms\_output.put\_line('New price: ' || :NEW.price);

dbms\_output.put\_line('price difference: ' || price\_difference);

END;

/

/\*

INSERT INTO car\_models (engine\_number, model, manufacturer, price)

VALUES ('111212146', 'i10', 'HUNDAI', 22000);

UPDATE car\_models SET price = 30000

WHERE model = 'i20';

-- DELETE CANNOT BE DONE BECAUSE IT'S DATA IS LINK TO CUSTOMER CAR TABLE

\*/

1. DROP

--@C:\Users\Sushil\Desktop\DatabaseProject\DROP.txt

-- Author: Sushil Pun

--15/03/2017

/\* Drop \*/

DROP FUNCTION func\_car\_models\_price;

DROP PROCEDURE proc\_car\_models\_price;

DROP PROCEDURE proc\_customer\_citizenship;

DROP PROCEDURE proc\_bill\_number;

DROP PROCEDURE proc\_booking\_item\_type;

DROP PROCEDURE proc\_insert\_customer\_car;

DROP PROCEDURE proc\_insert\_car\_model;

DROP PROCEDURE proc\_delete\_car\_model;

DROP PROCEDURE pro\_car\_category;

DROP PROCEDURE proc\_for\_loop;

DROP TRIGGER trig\_car\_price;

DROP TRIGGER trig\_booking\_date\_ck;

DROP TABLE bookings;

DROP TABLE customer\_cars;

DROP TABLE car\_models;

DROP TABLE mechanics;

DROP TABLE customers;

DROP TABLE addresses;

DROP TYPE address\_type;

DROP TYPE booking\_item\_table\_type;

DROP TYPE booking\_item\_type;

DROP TYPE specilization\_varray\_type;

DROP TYPE specilization\_type;

PURGE RECYCLEBIN;

# The test Plan:

|  |  |  |
| --- | --- | --- |
| Test | Expected result | Actual result |
| CREATE OR REPLACE TYPE address\_type AS OBJECT | Types must be created with three attribute. | Type created with three attribute. |
| CREATE TABLE addresses OF address\_type; | Needs to create object table addresses using address\_type. | Table created. |
| CREATE TABLE customers referencing addresses | Table customer with its attributes need to be created | Table customer created with attribute |
| CREATE TABLE car\_models | Table car model needs to be created with its attribute | Table car\_models created. |
| CREATE TABLE customers\_car | Table customer \_car with its attributes need to be created. | Table customer\_car created. |
| CREATE OR REPLACE TYPE specialization\_type AS OBJECT | Needs to create specialization type | Specialization type created. |
| CREATE TYPE specialization\_varray\_type AS VARRAY(50) OF contact\_type | Specialization varray type needs to be created | Specialization varray type created. |
| CREATE OR REPLACE TABLE mechanics referencing object type and specialization\_varray\_type | Mechanics table needs to be created with its attribute | Mechanics table created |
| CREATE OR REPLACE TYPE booking\_item\_type AS OBJECT | Booking \_item\_type needs to be created | Booking\_item\_ type created |
| CREATE TYPE booking\_item\_table\_type AS TABLE OF booking\_type | It must create nested table using booking item table type | Nested table created using booking\_item\_type |
| CREATE TABLE booking referencing the booking\_item\_table\_type | Needs to create a booking table with its attribute | Booking table created |