ADVANCED SQL OPERATIONS.

OBJECTIVE

Our Main Objective is to perform as many Advanced SQL Operations on already existing and created table in our MySQL Database Server.

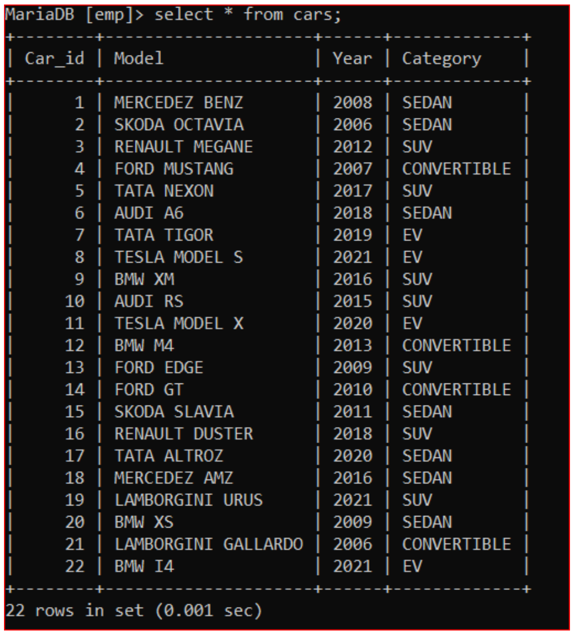
We will be performing necessary SQL tasks as well as queries that might seem necessary for Solving Complex Tasks as well as maintain the integrity of our Data in MySQL Database Server.

In this project we will cover topics like :-

- Normalization.
- Stored Procedures.
- Triggers.
- Cursors.
- Windows Function.

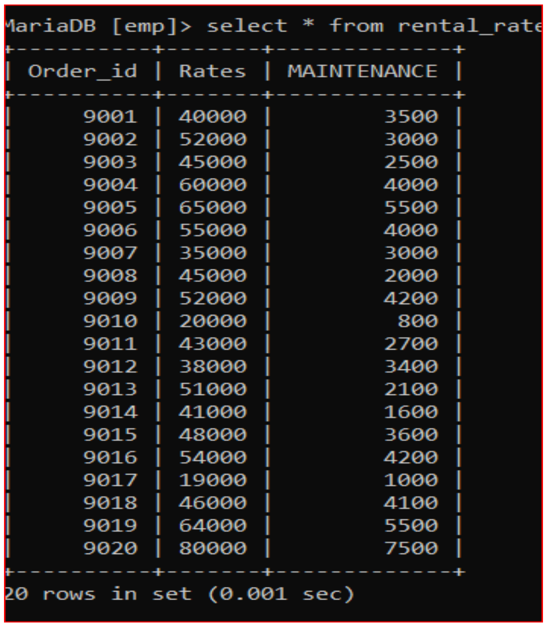
Here are our already Created SQL Tables that we will work on

CARS



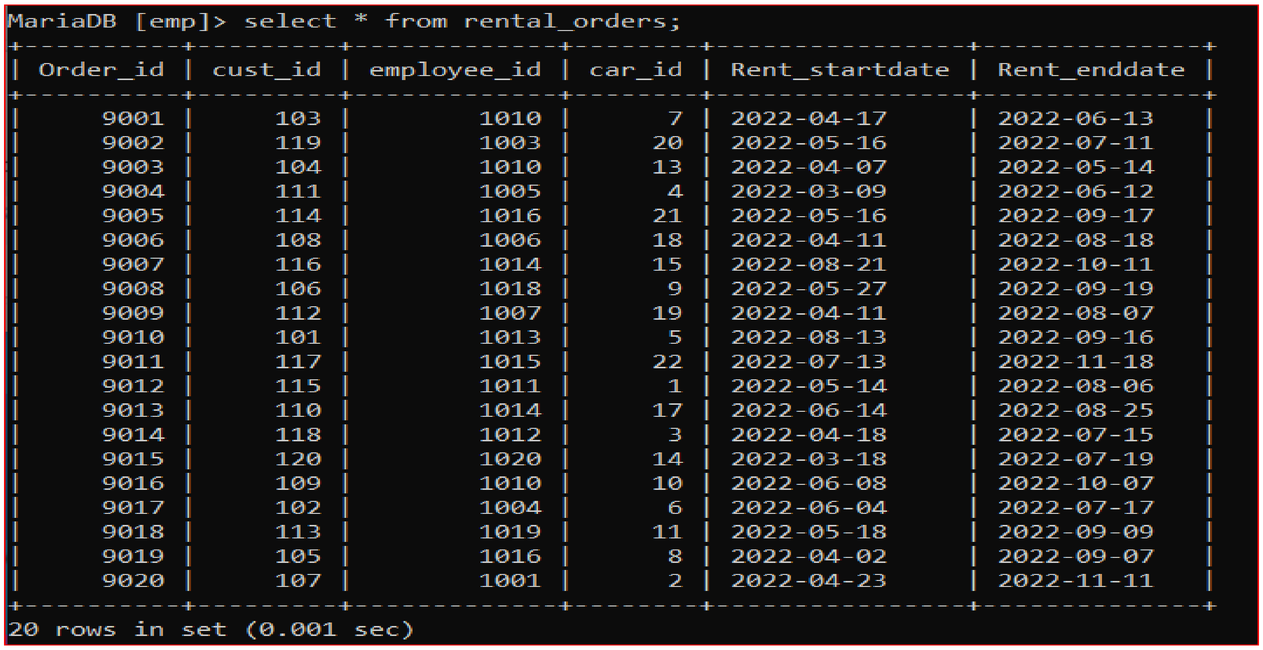
```
MariaDB [emp]> select * from cars;
+--------+--------------------+------+-------------+
| Car_id | Model              | Year | Category    |
+--------+--------------------+------+-------------+
|      1 | MERCEDEZ BENZ      | 2008 | SEDAN       |
|      2 | SKODA OCTAVIA      | 2006 | SEDAN       |
|      3 | RENAULT MEGANE     | 2012 | SUV         |
|      4 | FORD MUSTANG       | 2007 | CONVERTIBLE |
|      5 | TATA NEXON         | 2017 | SUV         |
|      6 | AUDI A6            | 2018 | SEDAN       |
|      7 | TATA TIGOR         | 2019 | EV          |
|      8 | TESLA MODEL S      | 2021 | EV          |
|      9 | BMW XM             | 2016 | SUV         |
|     10 | AUDI RS            | 2015 | SUV         |
|     11 | TESLA MODEL X      | 2020 | EV          |
|     12 | BMW M4             | 2013 | CONVERTIBLE |
|     13 | FORD EDGE          | 2009 | SUV         |
|     14 | FORD GT            | 2010 | CONVERTIBLE |
|     15 | SKODA SLAVIA       | 2011 | SEDAN       |
|     16 | RENAULT DUSTER     | 2018 | SUV         |
|     17 | TATA ALTROZ        | 2020 | SEDAN       |
|     18 | MERCEDEZ AMZ       | 2016 | SEDAN       |
|     19 | LAMBORGINI URUS    | 2021 | SUV         |
|     20 | BMW XS             | 2009 | SEDAN       |
|     21 | LAMBORGINI GALLARDO| 2006 | CONVERTIBLE |
|     22 | BMW I4             | 2021 | EV          |
+--------+--------------------+------+-------------+
22 rows in set (0.001 sec)
```

Rental Rates



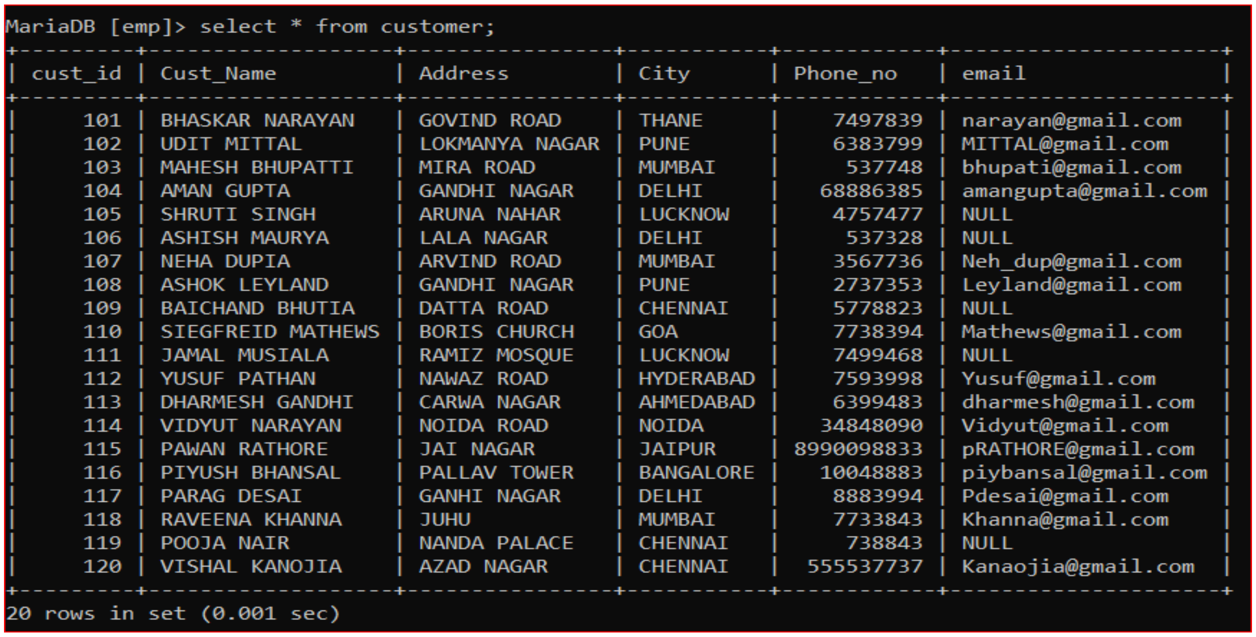
```
MariaDB [emp]> select * from rental_rate
+----------+-------+-------------+
| Order_id | Rates | MAINTENANCE |
+----------+-------+-------------+
|     9001 | 40000 |        3500 |
|     9002 | 52000 |        3000 |
|     9003 | 45000 |        2500 |
|     9004 | 60000 |        4000 |
|     9005 | 65000 |        5500 |
|     9006 | 55000 |        4000 |
|     9007 | 35000 |        3000 |
|     9008 | 45000 |        2000 |
|     9009 | 52000 |        4200 |
|     9010 | 20000 |         800 |
|     9011 | 43000 |        2700 |
|     9012 | 38000 |        3400 |
|     9013 | 51000 |        2100 |
|     9014 | 41000 |        1600 |
|     9015 | 48000 |        3600 |
|     9016 | 54000 |        4200 |
|     9017 | 19000 |        1000 |
|     9018 | 46000 |        4100 |
|     9019 | 64000 |        5500 |
|     9020 | 80000 |        7500 |
+----------+-------+-------------+
20 rows in set (0.001 sec)
```

Rental Orders



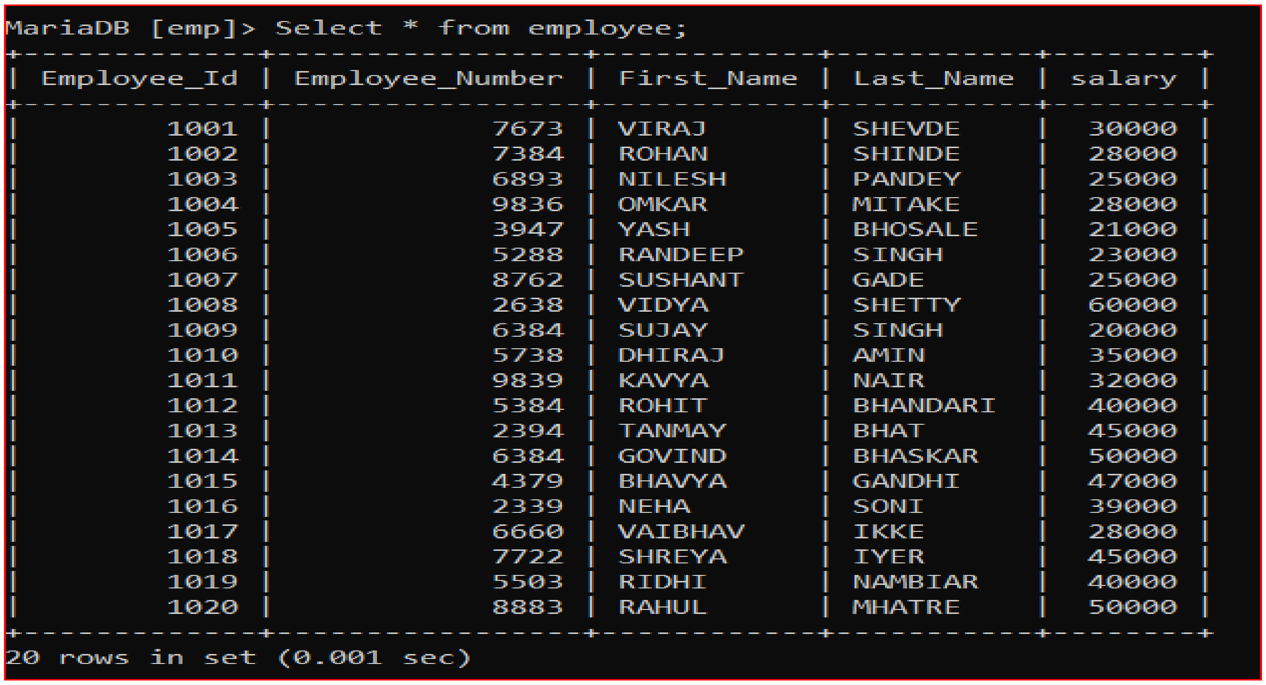
```
MariaDB [emp]> select * from rental_orders;
+----------+---------+-------------+--------+-----------------+--------------+
| Order_id | cust_id | employee_id | car_id | Rent_startdate  | Rent_enddate |
+----------+---------+-------------+--------+-----------------+--------------+
|     9001 |     103 |        1010 |      7 | 2022-04-17      | 2022-06-13   |
|     9002 |     119 |        1003 |     20 | 2022-05-16      | 2022-07-11   |
|     9003 |     104 |        1010 |     13 | 2022-04-07      | 2022-05-14   |
|     9004 |     111 |        1005 |      4 | 2022-03-09      | 2022-06-12   |
|     9005 |     114 |        1016 |     21 | 2022-05-16      | 2022-09-17   |
|     9006 |     108 |        1006 |     18 | 2022-04-11      | 2022-08-18   |
|     9007 |     116 |        1014 |     15 | 2022-08-21      | 2022-10-11   |
|     9008 |     106 |        1018 |      9 | 2022-05-27      | 2022-09-19   |
|     9009 |     112 |        1007 |     19 | 2022-04-11      | 2022-08-07   |
|     9010 |     101 |        1013 |      5 | 2022-08-13      | 2022-09-16   |
|     9011 |     117 |        1015 |     22 | 2022-07-13      | 2022-11-18   |
|     9012 |     115 |        1011 |      1 | 2022-05-14      | 2022-08-06   |
|     9013 |     110 |        1014 |     17 | 2022-06-14      | 2022-08-25   |
|     9014 |     118 |        1012 |      3 | 2022-04-18      | 2022-07-15   |
|     9015 |     120 |        1020 |     14 | 2022-03-18      | 2022-07-19   |
|     9016 |     109 |        1010 |     10 | 2022-06-08      | 2022-10-07   |
|     9017 |     102 |        1004 |      6 | 2022-06-04      | 2022-07-17   |
|     9018 |     113 |        1019 |     11 | 2022-05-18      | 2022-09-09   |
|     9019 |     105 |        1016 |      8 | 2022-04-02      | 2022-09-07   |
|     9020 |     107 |        1001 |      2 | 2022-04-23      | 2022-11-11   |
+----------+---------+-------------+--------+-----------------+--------------+
20 rows in set (0.001 sec)
```

Customer



```
MariaDB [emp]> select * from customer;
+---------+-------------------+----------------+-----------+------------+---------------------+
| cust_id | Cust_Name         | Address        | City      | Phone_no   | email               |
+---------+-------------------+----------------+-----------+------------+---------------------+
|     101 | BHASKAR NARAYAN   | GOVIND ROAD    | THANE     |    7497839 | narayan@gmail.com   |
|     102 | UDIT MITTAL       | LOKMANYA NAGAR | PUNE      |    6383799 | MITTAL@gmail.com    |
|     103 | MAHESH BHUPATTI   | MIRA ROAD      | MUMBAI    |     537748 | bhupati@gmail.com   |
|     104 | AMAN GUPTA        | GANDHI NAGAR   | DELHI     |   68886385 | amangupta@gmail.com |
|     105 | SHRUTI SINGH      | ARUNA NAHAR    | LUCKNOW   |    4757477 | NULL                |
|     106 | ASHISH MAURYA     | LALA NAGAR     | DELHI     |     537328 | NULL                |
|     107 | NEHA DUPIA        | ARVIND ROAD    | MUMBAI    |    3567736 | Neh_dup@gmail.com   |
|     108 | ASHOK LEYLAND     | GANDHI NAGAR   | PUNE      |    2737353 | Leyland@gmail.com   |
|     109 | BAICHAND BHUTIA   | DATTA ROAD     | CHENNAI   |    5778823 | NULL                |
|     110 | SIEGFREID MATHEWS | BORIS CHURCH   | GOA       |    7738394 | Mathews@gmail.com   |
|     111 | JAMAL MUSIALA     | RAMIZ MOSQUE   | LUCKNOW   |    7499468 | NULL                |
|     112 | YUSUF PATHAN      | NAWAZ ROAD     | HYDERABAD |    7593998 | Yusuf@gmail.com     |
|     113 | DHARMESH GANDHI   | CARWA NAGAR    | AHMEDABAD |    6399483 | dharmesh@gmail.com  |
|     114 | VIDYUT NARAYAN    | NOIDA ROAD     | NOIDA     |   34848090 | Vidyut@gmail.com    |
|     115 | PAWAN RATHORE     | JAI NAGAR      | JAIPUR    | 8990098833 | pRATHORE@gmail.com  |
|     116 | PIYUSH BHANSAL    | PALLAV TOWER   | BANGALORE |   10048883 | piybansal@gmail.com |
|     117 | PARAG DESAI       | GANHI NAGAR    | DELHI     |    8883994 | Pdesai@gmail.com    |
|     118 | RAVEENA KHANNA    | JUHU           | MUMBAI    |    7733843 | Khanna@gmail.com    |
|     119 | POOJA NAIR        | NANDA PALACE   | CHENNAI   |     738843 | NULL                |
|     120 | VISHAL KANOJIA    | AZAD NAGAR     | CHENNAI   |  555537737 | Kanaojia@gmail.com  |
+---------+-------------------+----------------+-----------+------------+---------------------+
20 rows in set (0.001 sec)
```

Employee



```
MariaDB [emp]> Select * from employee;
+-------------+-----------------+------------+-----------+--------+
| Employee_Id | Employee_Number | First_Name | Last_Name | salary |
+-------------+-----------------+------------+-----------+--------+
|        1001 |            7673 | VIRAJ      | SHEVDE    |  30000 |
|        1002 |            7384 | ROHAN      | SHINDE    |  28000 |
|        1003 |            6893 | NILESH     | PANDEY    |  25000 |
|        1004 |            9836 | OMKAR      | MITAKE    |  28000 |
|        1005 |            3947 | YASH       | BHOSALE   |  21000 |
|        1006 |            5288 | RANDEEP    | SINGH     |  23000 |
|        1007 |            8762 | SUSHANT    | GADE      |  25000 |
|        1008 |            2638 | VIDYA      | SHETTY    |  60000 |
|        1009 |            6384 | SUJAY      | SINGH     |  20000 |
|        1010 |            5738 | DHIRAJ     | AMIN      |  35000 |
|        1011 |            9839 | KAVYA      | NAIR      |  32000 |
|        1012 |            5384 | ROHIT      | BHANDARI  |  40000 |
|        1013 |            2394 | TANMAY     | BHAT      |  45000 |
|        1014 |            6384 | GOVIND     | BHASKAR   |  50000 |
|        1015 |            4379 | BHAVYA     | GANDHI    |  47000 |
|        1016 |            2339 | NEHA       | SONI      |  39000 |
|        1017 |            6660 | VAIBHAV    | IKKE      |  28000 |
|        1018 |            7722 | SHREYA     | IYER      |  45000 |
|        1019 |            5503 | RIDHI      | NAMBIAR   |  40000 |
|        1020 |            8883 | RAHUL      | MHATRE    |  50000 |
+-------------+-----------------+------------+-----------+--------+
20 rows in set (0.001 sec)
```

NORMALIZATION

Normalization is the process to eliminate data redundancy and enhance data integrity in the table. Normalization also helps to organize the data in the database. It is a multi-step process that sets the data into tabular form and removes the duplicated data from the relational tables.

TYPES OF NORMALIZATION

- ***1NF (FIRST NORMAL FORM)***
- ***2NF (SECOND NORMAL FORM)***
- ***3NF (THIRD NORMAL FORM)***
- ***BCNF (BOYCE CODD NORMAL FORM)***

1st Normal Form (1NF)

- A table is referred to as being in its First Normal Form if atomicity of the table is 1.
- Here, atomicity states that a single cell cannot hold multiple values. It must hold only a single-valued attribute.

Second Normal Form (2NF)

- The first condition for the table to be in Second Normal Form is that the table has to be in First Normal Form.

- The table should not possess partial dependency. The partial dependency here means the proper subset of the candidate key should give a non-prime attribute.

Third Normal Form (3NF)

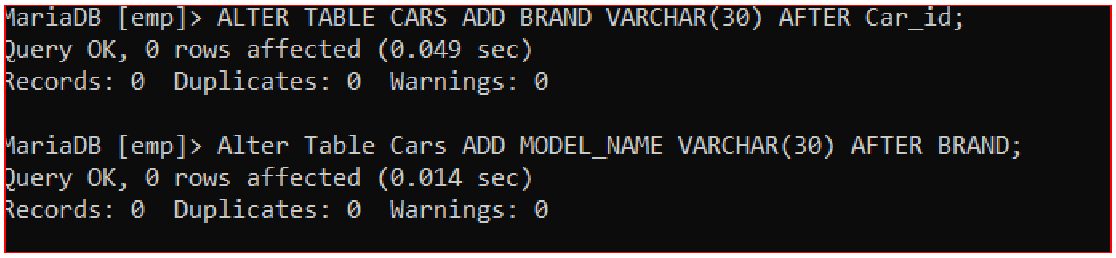
- The first condition for the table to be in Third Normal Form is that the table should be in the Second Normal Form.
- The second condition is that there should be no transitive dependency for non-prime attributes, which indicates that non-prime attributes (which are not a part of the candidate key) should not depend on other non-prime attributes in a table.
-

Boyce Codd Normal Form (BCNF)

- Boyce Codd Normal Form is also known as 3.5 NF.

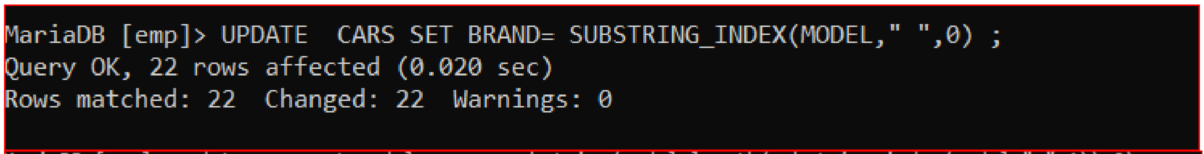
- The first condition for the table to be in Boyce Codd Normal Form is that the table should be in the third normal form. Secondly, every Right-Hand Side (RHS) attribute of the functional dependencies should depend on the super key of that particular table.

If you take a note on our tables in the Database you would notice that the Customer & Cars don't seem to follow the 1NF, so let's fix that shall we.

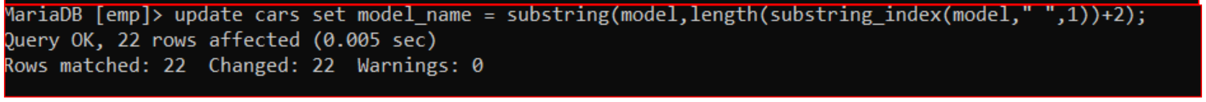


```
MariaDB [emp]> ALTER TABLE CARS ADD BRAND VARCHAR(30) AFTER Car_id;
Query OK, 0 rows affected (0.049 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [emp]> Alter Table Cars ADD MODEL_NAME VARCHAR(30) AFTER BRAND;
Query OK, 0 rows affected (0.014 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Add a new column called as brand & model_name using alter with add column.

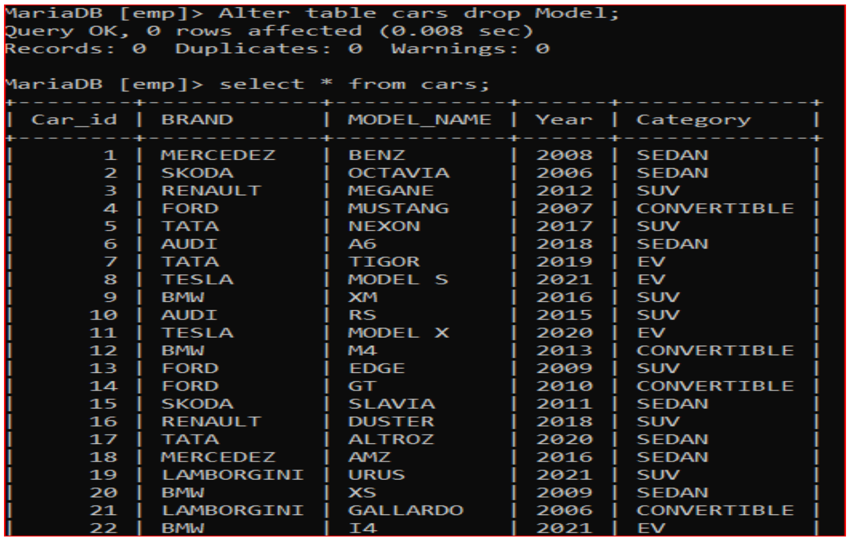


```
MariaDB [emp]> UPDATE  CARS SET BRAND= SUBSTRING_INDEX(MODEL," ",0) ;
Query OK, 22 rows affected (0.020 sec)
Rows matched: 22  Changed: 22  Warnings: 0
```

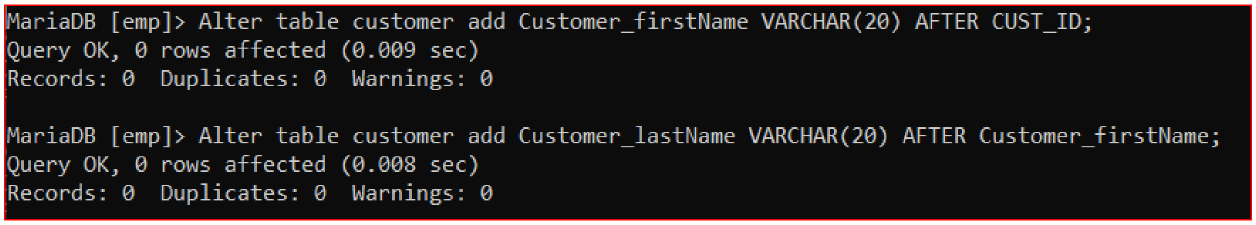


```
MariaDB [emp]> update cars set model_name = substring(model,length(substring_index(model," ",1))+2);
Query OK, 22 rows affected (0.005 sec)
Rows matched: 22  Changed: 22  Warnings: 0
```

Insert the values on both the column by splitting the model column of the cars table using substring & substring_index function. then drop the previous column.

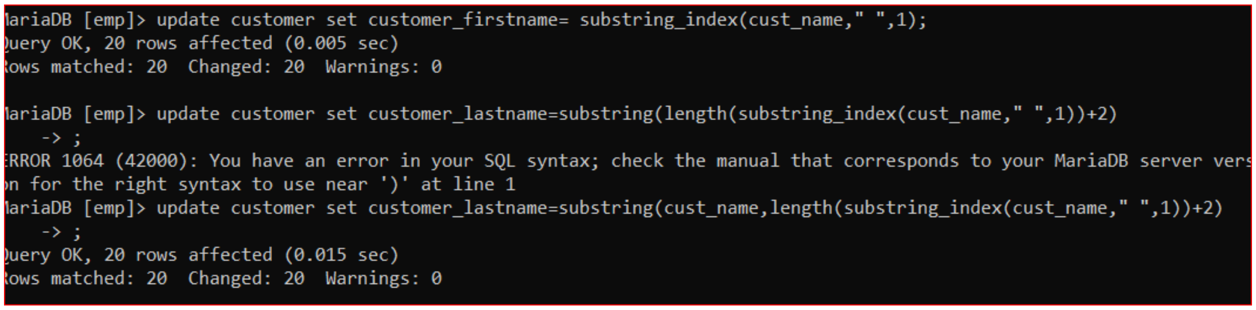


```
MariaDB [emp]> Alter table cars drop Model;
Query OK, 0 rows affected (0.008 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [emp]> select * from cars;
+--------+-------------+------------+------+-------------+
| Car_id | BRAND       | MODEL_NAME | Year | Category    |
+--------+-------------+------------+------+-------------+
|      1 | MERCEDEZ    | BENZ       | 2008 | SEDAN       |
|      2 | SKODA       | OCTAVIA    | 2006 | SEDAN       |
|      3 | RENAULT     | MEGANE     | 2012 | SUV         |
|      4 | FORD        | MUSTANG    | 2007 | CONVERTIBLE |
|      5 | TATA        | NEXON      | 2017 | SUV         |
|      6 | AUDI        | A6         | 2018 | SEDAN       |
|      7 | TATA        | TIGOR      | 2019 | EV          |
|      8 | TESLA       | MODEL S    | 2021 | EV          |
|      9 | BMW         | XM         | 2016 | SUV         |
|     10 | AUDI        | RS         | 2015 | SUV         |
|     11 | TESLA       | MODEL X    | 2020 | EV          |
|     12 | BMW         | M4         | 2013 | CONVERTIBLE |
|     13 | FORD        | EDGE       | 2009 | SUV         |
|     14 | FORD        | GT         | 2010 | CONVERTIBLE |
|     15 | SKODA       | SLAVIA     | 2011 | SEDAN       |
|     16 | RENAULT     | DUSTER     | 2018 | SUV         |
|     17 | TATA        | ALTROZ     | 2020 | SEDAN       |
|     18 | MERCEDEZ    | AMZ        | 2016 | SEDAN       |
|     19 | LAMBORGINI  | URUS       | 2021 | SUV         |
|     20 | BMW         | XS         | 2009 | SEDAN       |
|     21 | LAMBORGINI  | GALLARDO   | 2006 | CONVERTIBLE |
|     22 | BMW         | I4         | 2021 | EV          |
```



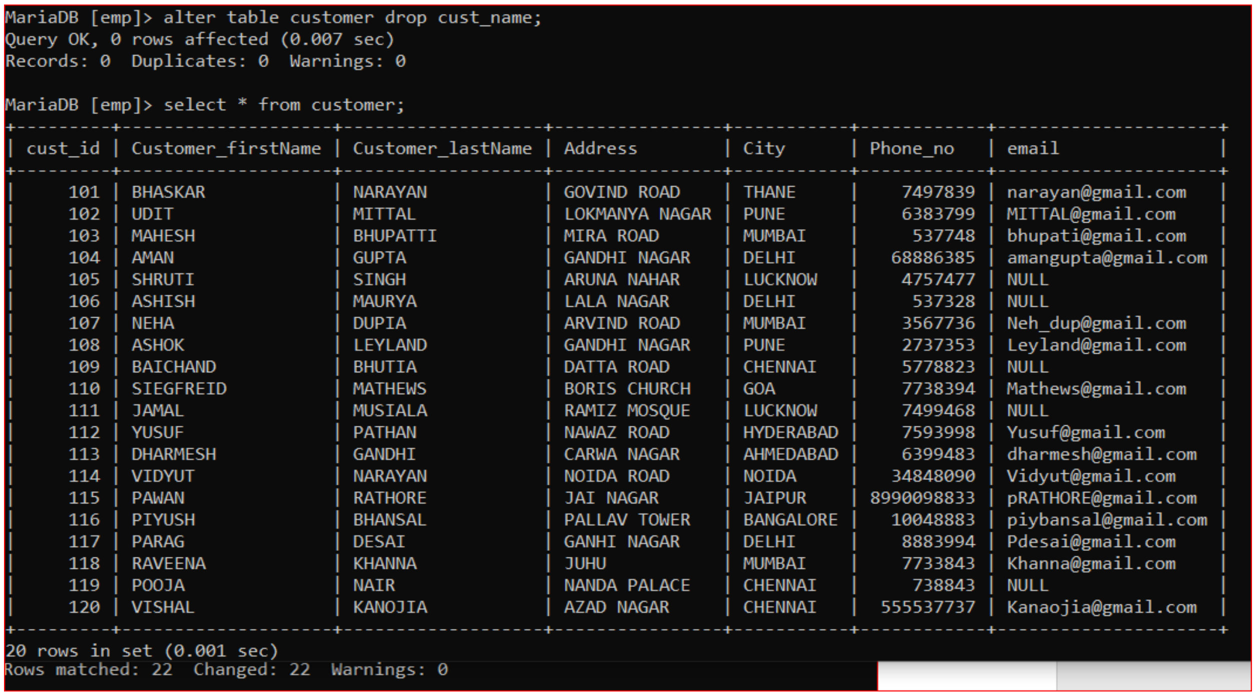
```
MariaDB [emp]> Alter table customer add Customer_firstName VARCHAR(20) AFTER CUST_ID;
Query OK, 0 rows affected (0.009 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [emp]> Alter table customer add Customer_lastName VARCHAR(20) AFTER Customer_firstName;
Query OK, 0 rows affected (0.008 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Create a column Customer_firstName & Customer_lastName

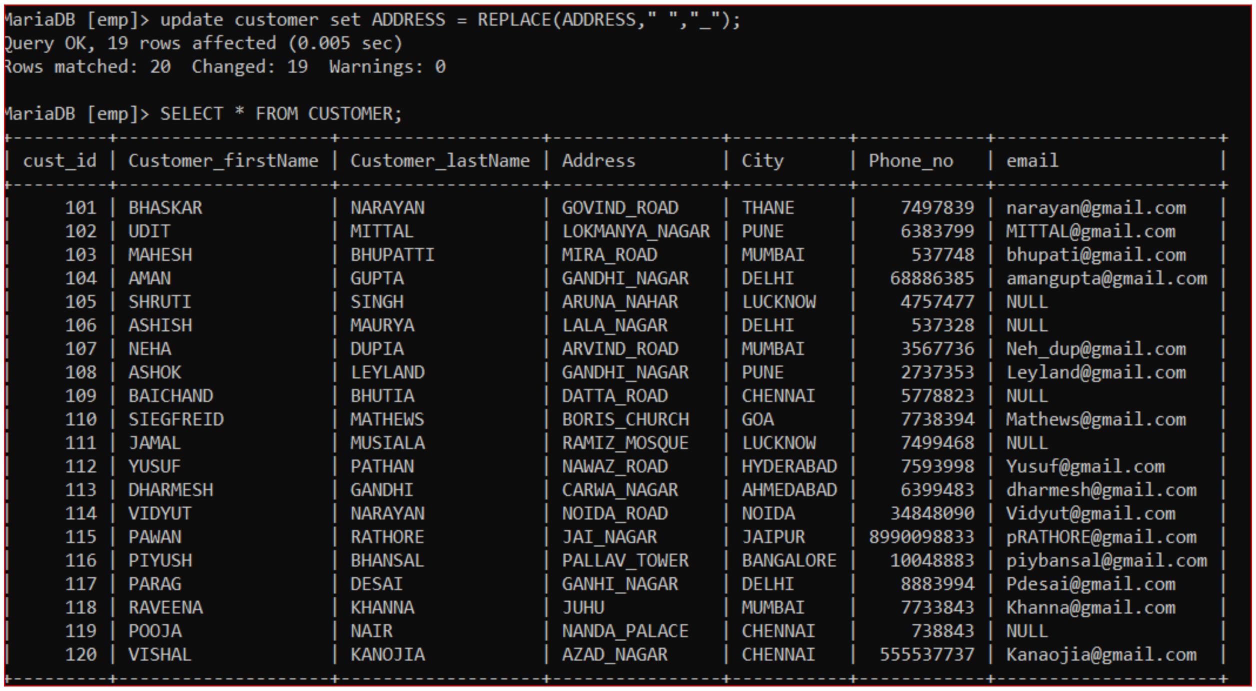


```
lariaDB [emp]> update customer set customer_firstname= substring_index(cust_name," ",1);
Juery OK, 20 rows affected (0.005 sec)
Rows matched: 20  Changed: 20  Warnings: 0

lariaDB [emp]> update customer set customer_lastname=substring(length(substring_index(cust_name," ",1))+2)
    -> ;
RROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server vers
on for the right syntax to use near ')' at line 1
lariaDB [emp]> update customer set customer_lastname=substring(cust_name,length(substring_index(cust_name," ",1))+2)
    -> ;
Juery OK, 20 rows affected (0.015 sec)
Rows matched: 20  Changed: 20  Warnings: 0
```

Insert the values on both the column by splitting the model column of the customer table using substring & substring_index function. then drop the previous column.



```
MariaDB [emp]> alter table customer drop cust_name;
Query OK, 0 rows affected (0.007 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [emp]> select * from customer;
+---------+--------------------+-------------------+----------------+-----------+------------+---------------------+
| cust_id | Customer_firstName | Customer_lastName | Address        | City      | Phone_no   | email               |
+---------+--------------------+-------------------+----------------+-----------+------------+---------------------+
|     101 | BHASKAR            | NARAYAN           | GOVIND ROAD    | THANE     |    7497839 | narayan@gmail.com   |
|     102 | UDIT               | MITTAL            | LOKMANYA NAGAR | PUNE      |    6383799 | MITTAL@gmail.com    |
|     103 | MAHESH             | BHUPATTI          | MIRA ROAD      | MUMBAI    |     537748 | bhupati@gmail.com   |
|     104 | AMAN               | GUPTA             | GANDHI NAGAR   | DELHI     |   68886385 | amangupta@gmail.com |
|     105 | SHRUTI             | SINGH             | ARUNA NAHAR    | LUCKNOW   |    4757477 | NULL                |
|     106 | ASHISH             | MAURYA            | LALA NAGAR     | DELHI     |     537328 | NULL                |
|     107 | NEHA               | DUPIA             | ARVIND ROAD    | MUMBAI    |    3567736 | Neh_dup@gmail.com   |
|     108 | ASHOK              | LEYLAND           | GANDHI NAGAR   | PUNE      |    2737353 | Leyland@gmail.com   |
|     109 | BAICHAND           | BHUTIA            | DATTA ROAD     | CHENNAI   |    5778823 | NULL                |
|     110 | SIEGFREID          | MATHEWS           | BORIS CHURCH   | GOA       |    7738394 | Mathews@gmail.com   |
|     111 | JAMAL              | MUSIALA           | RAMIZ MOSQUE   | LUCKNOW   |    7499468 | NULL                |
|     112 | YUSUF              | PATHAN            | NAWAZ ROAD     | HYDERABAD |    7593998 | Yusuf@gmail.com     |
|     113 | DHARMESH           | GANDHI            | CARWA NAGAR    | AHMEDABAD |    6399483 | dharmesh@gmail.com  |
|     114 | VIDYUT             | NARAYAN           | NOIDA ROAD     | NOIDA     |   34848090 | Vidyut@gmail.com    |
|     115 | PAWAN              | RATHORE           | JAI NAGAR      | JAIPUR    | 8990098833 | pRATHORE@gmail.com  |
|     116 | PIYUSH             | BHANSAL           | PALLAV TOWER   | BANGALORE |   10048883 | piybansal@gmail.com |
|     117 | PARAG              | DESAI             | GANHI NAGAR    | DELHI     |    8883994 | Pdesai@gmail.com    |
|     118 | RAVEENA            | KHANNA            | JUHU           | MUMBAI    |    7733843 | Khanna@gmail.com    |
|     119 | POOJA              | NAIR              | NANDA PALACE   | CHENNAI   |     738843 | NULL                |
|     120 | VISHAL             | KANOJIA           | AZAD NAGAR     | CHENNAI   |  555537737 | Kanaojia@gmail.com  |
+---------+--------------------+-------------------+----------------+-----------+------------+---------------------+
20 rows in set (0.001 sec)
Rows matched: 22  Changed: 22  Warnings: 0
```



```
MariaDB [emp]> update customer set ADDRESS = REPLACE(ADDRESS," ","_");
Query OK, 19 rows affected (0.005 sec)
Rows matched: 20  Changed: 19  Warnings: 0

MariaDB [emp]> SELECT * FROM CUSTOMER;
+---------+--------------------+-------------------+----------------+-----------+------------+---------------------+
| cust_id | Customer_firstName | Customer_lastName | Address        | City      | Phone_no   | email               |
+---------+--------------------+-------------------+----------------+-----------+------------+---------------------+
|     101 | BHASKAR            | NARAYAN           | GOVIND_ROAD    | THANE     |    7497839 | narayan@gmail.com   |
|     102 | UDIT               | MITTAL            | LOKMANYA_NAGAR | PUNE      |    6383799 | MITTAL@gmail.com    |
|     103 | MAHESH             | BHUPATTI          | MIRA_ROAD      | MUMBAI    |     537748 | bhupati@gmail.com   |
|     104 | AMAN               | GUPTA             | GANDHI_NAGAR   | DELHI     |   68886385 | amangupta@gmail.com |
|     105 | SHRUTI             | SINGH             | ARUNA_NAHAR    | LUCKNOW   |    4757477 | NULL                |
|     106 | ASHISH             | MAURYA            | LALA_NAGAR     | DELHI     |     537328 | NULL                |
|     107 | NEHA               | DUPIA             | ARVIND_ROAD    | MUMBAI    |    3567736 | Neh_dup@gmail.com   |
|     108 | ASHOK              | LEYLAND           | GANDHI_NAGAR   | PUNE      |    2737353 | Leyland@gmail.com   |
|     109 | BAICHAND           | BHUTIA            | DATTA_ROAD     | CHENNAI   |    5778823 | NULL                |
|     110 | SIEGFREID          | MATHEWS           | BORIS_CHURCH   | GOA       |    7738394 | Mathews@gmail.com   |
|     111 | JAMAL              | MUSIALA           | RAMIZ_MOSQUE   | LUCKNOW   |    7499468 | NULL                |
|     112 | YUSUF              | PATHAN            | NAWAZ_ROAD     | HYDERABAD |    7593998 | Yusuf@gmail.com     |
|     113 | DHARMESH           | GANDHI            | CARWA_NAGAR    | AHMEDABAD |    6399483 | dharmesh@gmail.com  |
|     114 | VIDYUT             | NARAYAN           | NOIDA_ROAD     | NOIDA     |   34848090 | Vidyut@gmail.com    |
|     115 | PAWAN              | RATHORE           | JAI_NAGAR      | JAIPUR    | 8990098833 | pRATHORE@gmail.com  |
|     116 | PIYUSH             | BHANSAL           | PALLAV_TOWER   | BANGALORE |   10048883 | piybansal@gmail.com |
|     117 | PARAG              | DESAI             | GANHI_NAGAR    | DELHI     |    8883994 | Pdesai@gmail.com    |
|     118 | RAVEENA            | KHANNA            | JUHU           | MUMBAI    |    7733843 | Khanna@gmail.com    |
|     119 | POOJA              | NAIR              | NANDA_PALACE   | CHENNAI   |     738843 | NULL                |
|     120 | VISHAL             | KANOJIA           | AZAD_NAGAR     | CHENNAI   |  555537737 | Kanaojia@gmail.com  |
+---------+--------------------+-------------------+----------------+-----------+------------+---------------------+
```

Replace the Space in the Address column with “_” as it will convert it into one atomic Element without much modification on the Address.

STORED PROCEDURES

A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again. So if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.

SYNTAX FOR STORED PROCEDURES:

```
CREATE or REPLACE PROCEDURE name(parameters)

AS

variables;

BEGIN;

//statements;

END;
```

TYPES OF PARAMETERS IN STORED PROCEDURES

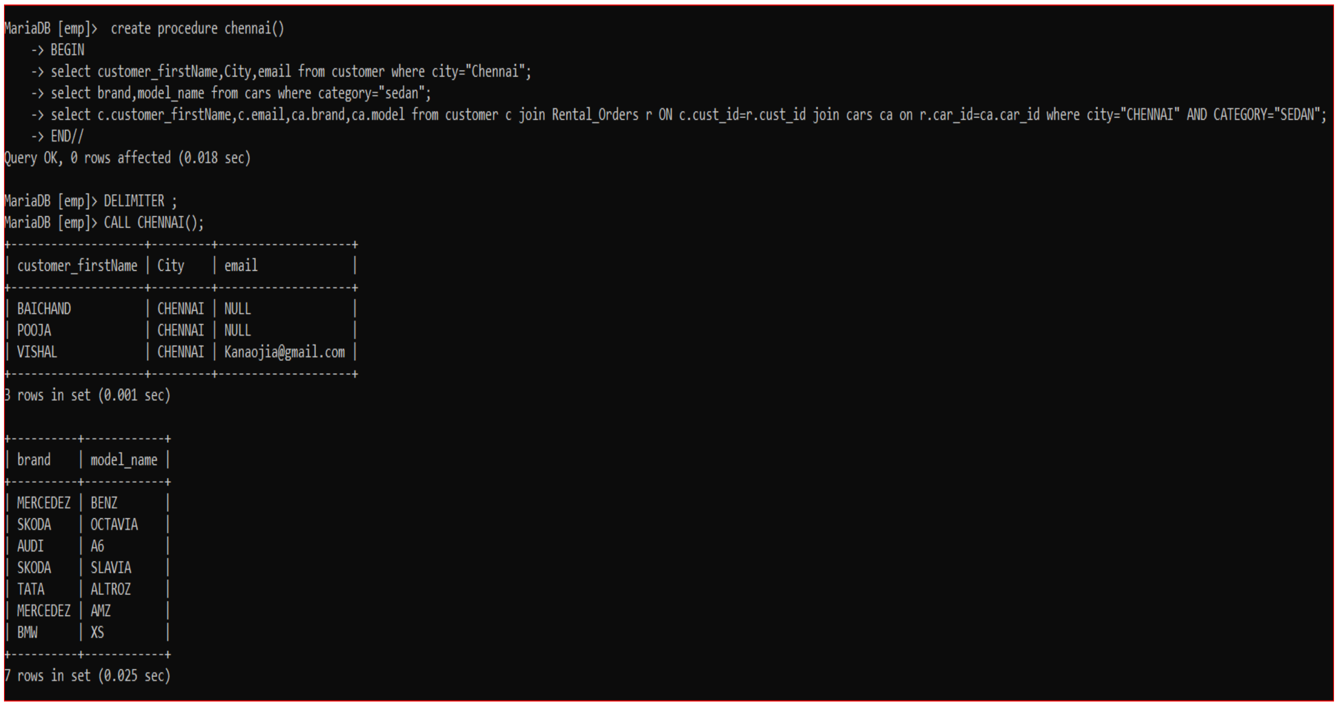
- **NO PARAMETER**
- **INPUT PARAMETER**

- **OUTPUT PARAMETER**
- **IN/OUT PARAMETER**

Now let’s Create a few Stored procedures from our Tables.

LET’S STORE 3 QUERIES IN OUR STORED PROCEDURE CALLED CHENNAI WHERE WE WILL FIND OUT THE NAMES OF THE CUSTOMERS WHO LIVE IN CHENNAI, NAMES OF THE CARS WHICH ARE SEDAN AND THE NAMES OF CUSTOMERS WHO LIVE IN CHENNAI AND OWN A SEDAN.

(NO PARAMETER STORED PROCEDURE)

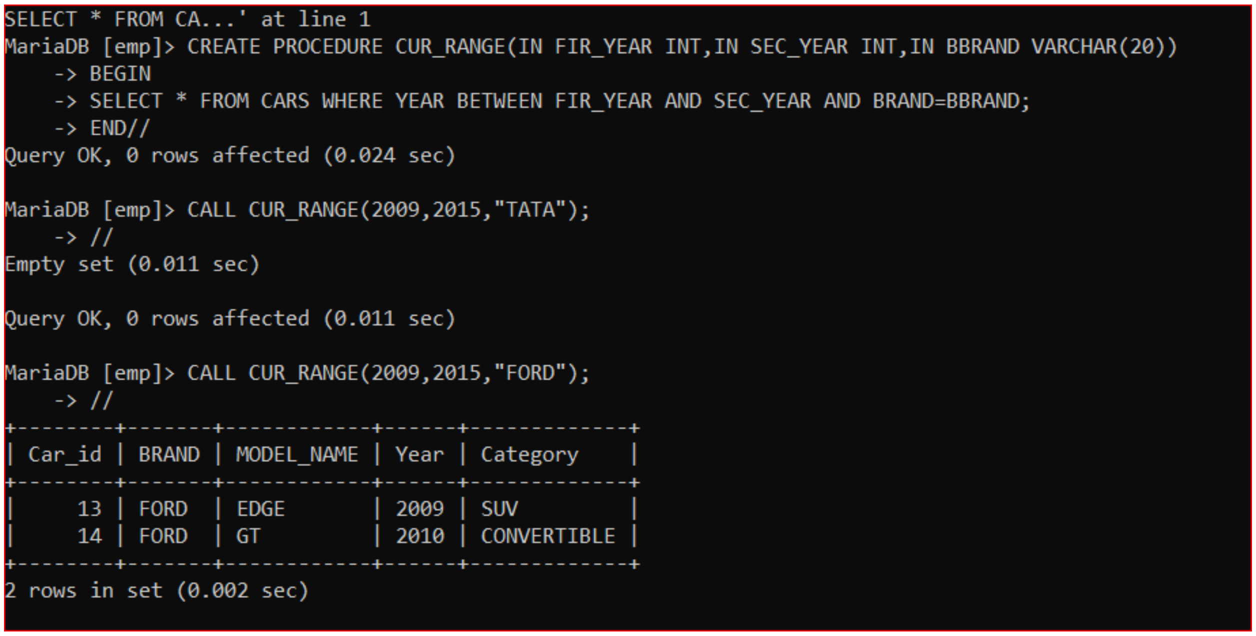


```
MariaDB [emp]>  create procedure chennai()
    -> BEGIN
    -> select customer_firstName,City,email from customer where city="Chennai";
    -> select brand,model_name from cars where category="sedan";
    -> select c.customer_firstName,c.email,ca.brand,ca.model from customer c join Rental_Orders r ON c.cust_id=r.cust_id join cars ca on r.car_id=ca.car_id where city="CHENNAI" AND CATEGORY="SEDAN";
    -> END//
Query OK, 0 rows affected (0.018 sec)

MariaDB [emp]> DELIMITER ;
MariaDB [emp]> CALL CHENNAI();
+-------------------+---------+-------------------+
| customer_firstName | City    | email             |
+-------------------+---------+-------------------+
| BAICHAND          | CHENNAI | NULL              |
| POOJA             | CHENNAI | NULL              |
| VISHAL            | CHENNAI | Kanaojia@gmail.com |
+-------------------+---------+-------------------+
3 rows in set (0.001 sec)

+----------+------------+
| brand    | model_name |
+----------+------------+
| MERCEDEZ | BENZ       |
| SKODA    | OCTAVIA    |
| AUDI     | A6         |
| SKODA    | SLAVIA     |
| TATA     | ALTROZ     |
| MERCEDEZ | AMZ        |
| BMW      | XS         |
+----------+------------+
7 rows in set (0.025 sec)
```

Syntax:- CALL/EXEC PROCEDURE NAME :- TO EXECUTE THE PROCEDURE.

Now we will create a Stored Procedure which takes Multiple YEARS and the Brand of a Car as an input and return the Records according to it.

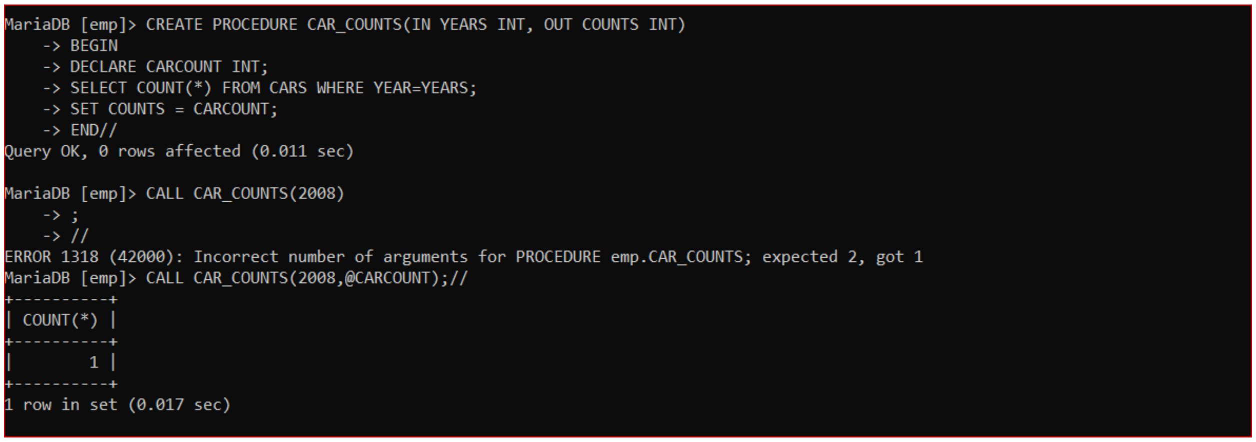
(INPUT PARAMETER)



```
SELECT * FROM CA...' at line 1
MariaDB [emp]> CREATE PROCEDURE CUR_RANGE(IN FIR_YEAR INT,IN SEC_YEAR INT,IN BBRAND VARCHAR(20))
    -> BEGIN
    -> SELECT * FROM CARS WHERE YEAR BETWEEN FIR_YEAR AND SEC_YEAR AND BRAND=BBRAND;
    -> END//
Query OK, 0 rows affected (0.024 sec)

MariaDB [emp]> CALL CUR_RANGE(2009,2015,"TATA");
    -> //
Empty set (0.011 sec)

Query OK, 0 rows affected (0.011 sec)

MariaDB [emp]> CALL CUR_RANGE(2009,2015,"FORD");
    -> //
+--------+-------+------------+------+-------------+
| Car_id | BRAND | MODEL_NAME | Year | Category    |
+--------+-------+------------+------+-------------+
|     13 | FORD  | EDGE       | 2009 | SUV         |
|     14 | FORD  | GT         | 2010 | CONVERTIBLE |
+--------+-------+------------+------+-------------+
2 rows in set (0.002 sec)
```

Now let's find the number of cars produced that year by giving the year at which it was manufactured.

(INPUT-OUTPUT PARAMETER)



```
MariaDB [emp]> CREATE PROCEDURE CAR_COUNTS(IN YEARS INT, OUT COUNTS INT)
    -> BEGIN
    -> DECLARE CARCOUNT INT;
    -> SELECT COUNT(*) FROM CARS WHERE YEAR=YEARS;
    -> SET COUNTS = CARCOUNT;
    -> END//
Query OK, 0 rows affected (0.011 sec)

MariaDB [emp]> CALL CAR_COUNTS(2008)
    -> ;
    -> //
ERROR 1318 (42000): Incorrect number of arguments for PROCEDURE emp.CAR_COUNTS; expected 2, got 1
MariaDB [emp]> CALL CAR_COUNTS(2008,@CARCOUNT);//
+----------+
| COUNT(*) |
+----------+
|        1 |
+----------+
1 row in set (0.017 sec)
```

TRIGGERS

A trigger is a special type of stored procedure that automatically runs when an event occurs in the database server. DML triggers run when a user tries to modify data through a data manipulation language (DML) event.

TYPES OF TRIGGERS

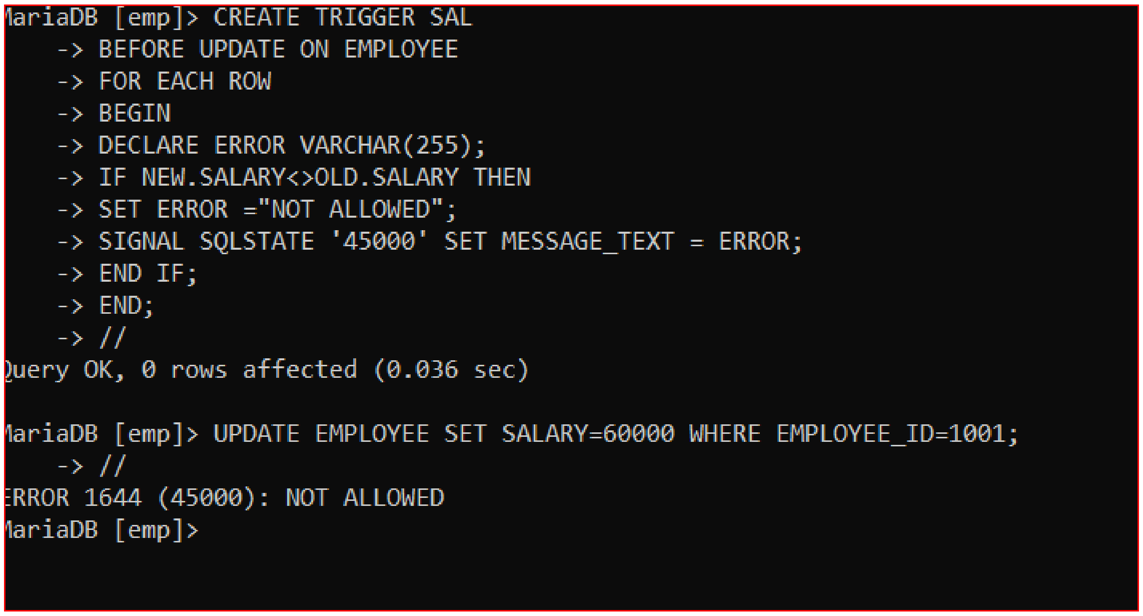
- **DML TRIGGER (INSERT, UPDATE, DELETE)**
- **DDL TRIGGER (CREATE, ALTER, DROP)**
- **LOGON TRIGGER (WHEN A LOGON EVENT TAKES PLACE)**

SYNTAX:-

```
CREATE TRIGGER trigger_name
ON table_name
AFTER (INSERT, UPDATE, DELETE)
AS
(SQL_Statements)
```

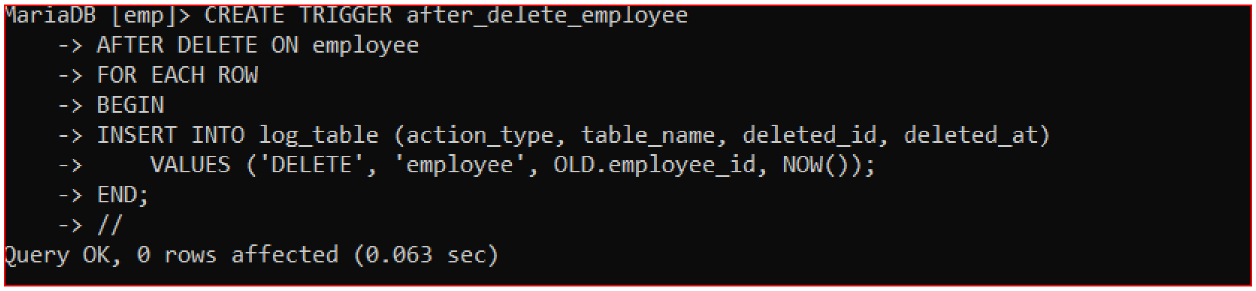
We will now Create a Trigger on Employee Database to prevent any kind of Update on Employees Salary.

If anyone tries to update employee salary in the database the trigger will block it and display the message "not allowed".

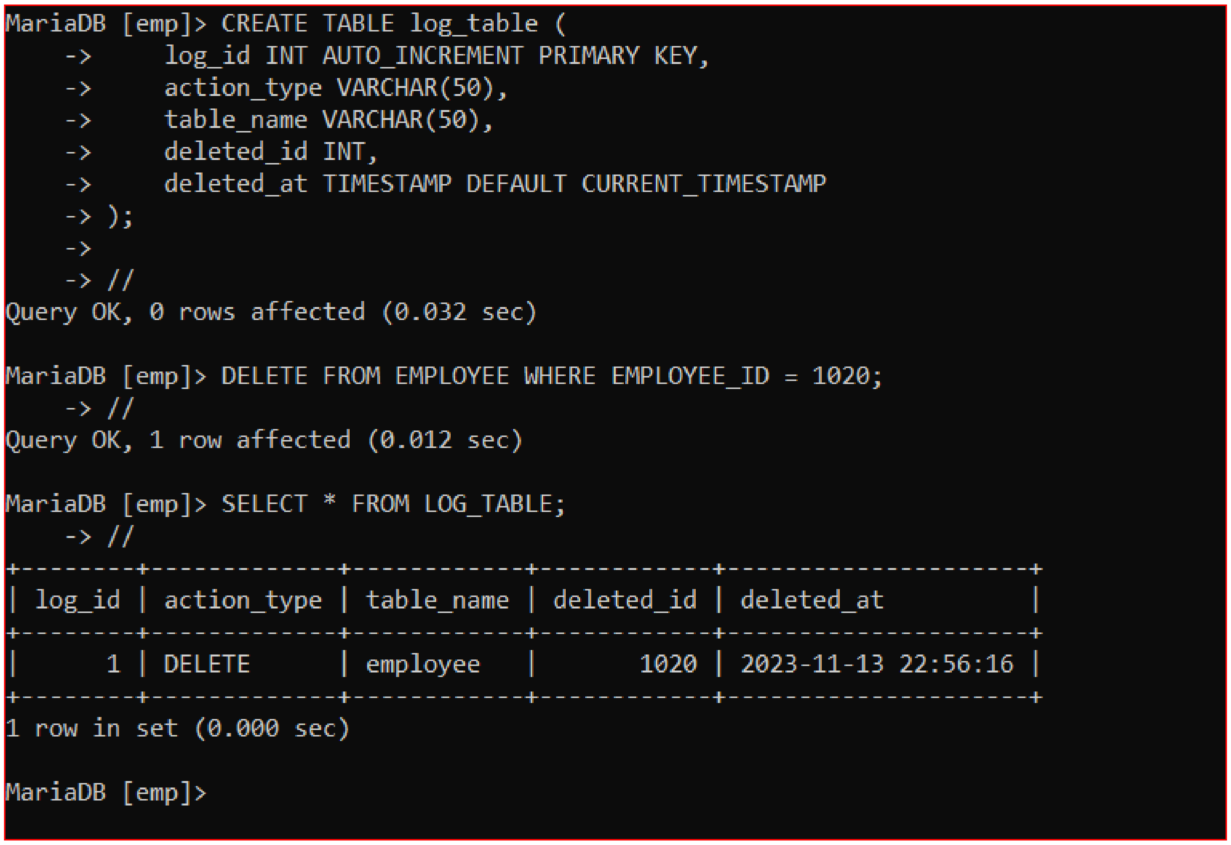


```
lariaDB [emp]> CREATE TRIGGER SAL
    -> BEFORE UPDATE ON EMPLOYEE
    -> FOR EACH ROW
    -> BEGIN
    -> DECLARE ERROR VARCHAR(255);
    -> IF NEW.SALARY<>OLD.SALARY THEN
    -> SET ERROR ="NOT ALLOWED";
    -> SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = ERROR;
    -> END IF;
    -> END;
    -> //
Query OK, 0 rows affected (0.036 sec)

lariaDB [emp]> UPDATE EMPLOYEE SET SALARY=60000 WHERE EMPLOYEE_ID=1001;
    -> //
ERROR 1644 (45000): NOT ALLOWED
lariaDB [emp]>
```

Creating a Trigger Event where all the Records AFTER DELETION gets stored in a particular log_table



```
MariaDB [emp]> CREATE TRIGGER after_delete_employee
    -> AFTER DELETE ON employee
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO log_table (action_type, table_name, deleted_id, deleted_at)
    ->     VALUES ('DELETE', 'employee', OLD.employee_id, NOW());
    -> END;
    -> //
Query OK, 0 rows affected (0.063 sec)
```



```
MariaDB [emp]> CREATE TABLE log_table (
    ->     log_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     action_type VARCHAR(50),
    ->     table_name VARCHAR(50),
    ->     deleted_id INT,
    ->     deleted_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
    -> );
    ->
    -> //
Query OK, 0 rows affected (0.032 sec)

MariaDB [emp]> DELETE FROM EMPLOYEE WHERE EMPLOYEE_ID = 1020;
    -> //
Query OK, 1 row affected (0.012 sec)

MariaDB [emp]> SELECT * FROM LOG_TABLE;
    -> //
+--------+-------------+------------+------------+---------------------+
| log_id | action_type | table_name | deleted_id | deleted_at          |
+--------+-------------+------------+------------+---------------------+
|      1 | DELETE      | employee   |       1020 | 2023-11-13 22:56:16 |
+--------+-------------+------------+------------+---------------------+
1 row in set (0.000 sec)

MariaDB [emp]>
```

AFTER CREATION OF THE TRIGGER, CREATE THE TABLE TO STORE ALL THE DELETED RECORDS AS YOU CAN SEE IN THE IMAGE ABOVE.

CURSOR

Cursor is a temporary memory or temporary workstation.

A SQL cursor is a database object that is used to retrieve data from a result set one row at a time.

A cursor is used when we want to the data needs to be updated row by row.

Cursors are used to store Database Tables and allows you to process individual row returned by Queries.

TYPES OF CURSORS

- **IMPLICIT CURSOR (AUTOMATICALLY)**
- **EXPLICIT CURSOR (MANUALLY)**

IMPLICIT CURSOR

Implicit cursors are also known as default cursors of SQL servers. These are allocated by SQL servers when a user performs DML operations.

EXPLICIT CURSOR

Explicit cursors are created by Users whenever the user requires them. Explicit Cursors are used for Fetching data from Table in Row-by-Row Manner.

METHODS OF CURSOR

- **NEXT**
- **PRIOR**
- **FIRST**
- **LAST**
- **ABSOLUTE N**
- **RELATIVE N**

SYNTAX:

DECLARE CURSOR_NAME CURSOR FOR TABLE_NAME

OPEN CURSOR_NAME

FETCH METHOD FROM CURSOR_NAME

CLOSE CURSOR_NAME

DEALLOCATE CURSOR_NAME.

WINDOWS FUNCTION

A window function is an SQL function where the input values are taken from a "window" of one or more rows in the results set of a SELECT statement.

Window functions are distinguished from aggregate functions by the presence of an OVER clause. If a function has an OVER clause, then it is a window function. If it lacks an OVER clause, then it is an ordinary aggregate function. Window functions might also have a FILTER clause in between the function and the OVER clause.

The window functions are divided into three types value window functions, aggregation window functions, and ranking window functions:

Value window functions

- FIRST_VALUE()
- LAG()
- LAST VALUE()
- LEAD()

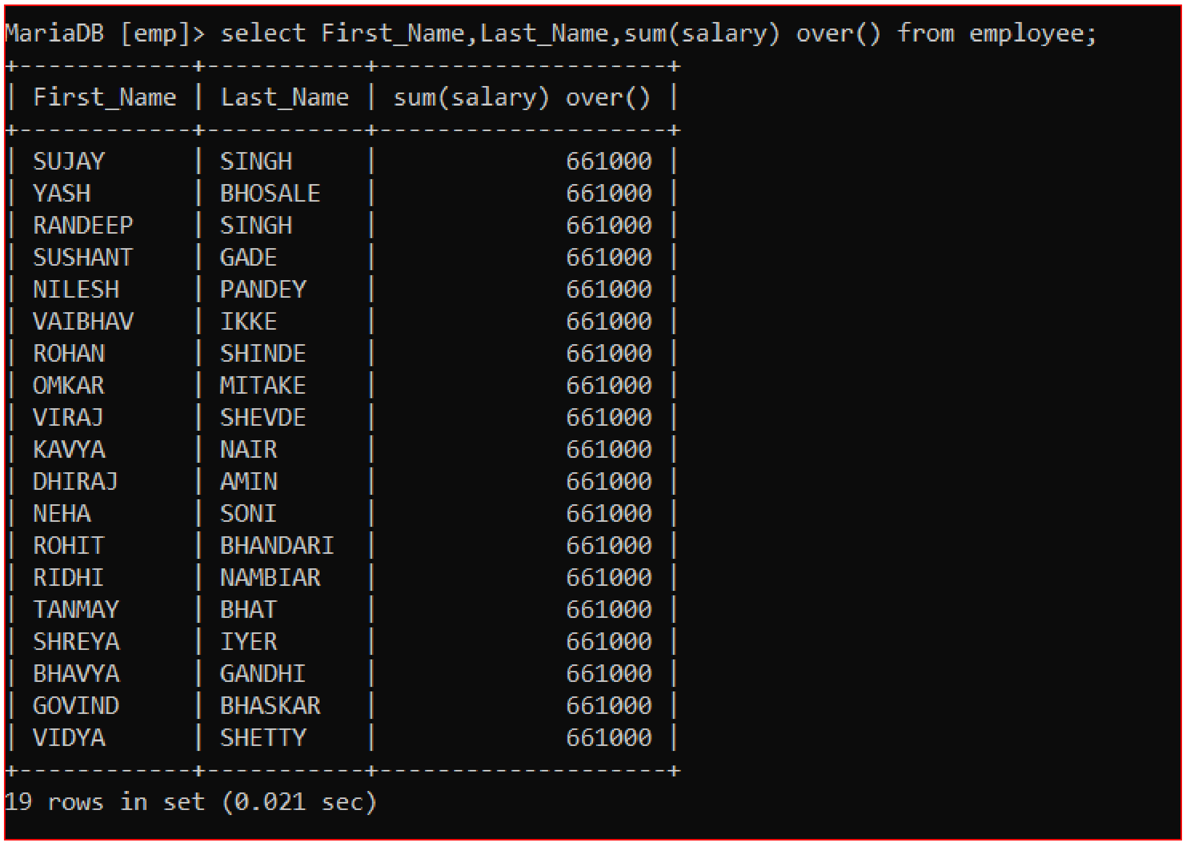
Ranking window functions

- CUME_DIST()
- DENSE_RANK()
- NTILE()
- PERCENT_RANK()
- RANK()
- ROW_NUMBER()

Aggregate window functions

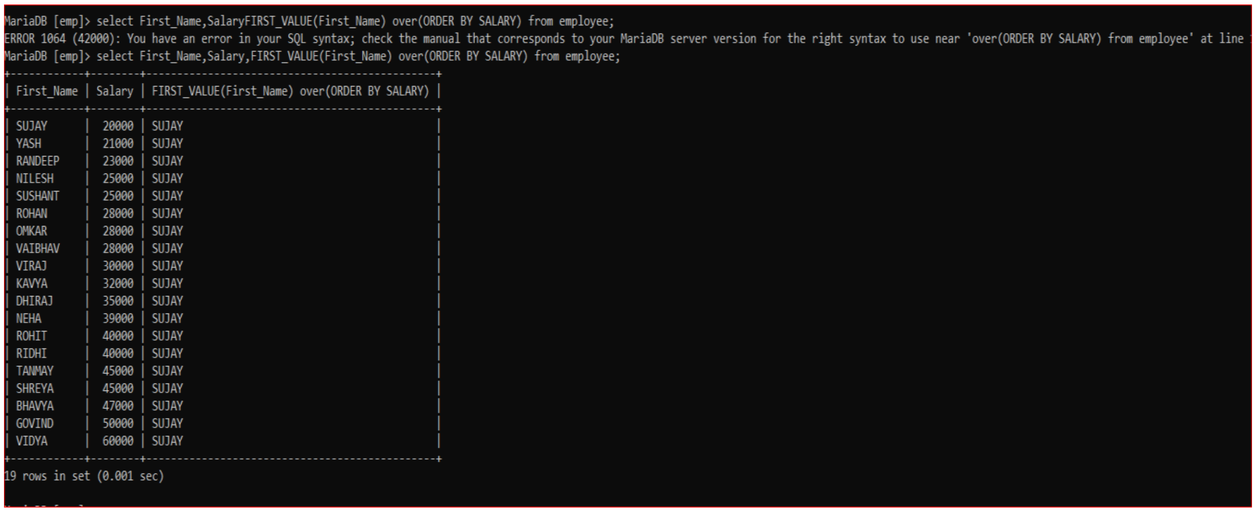
- AVG()
- COUNT()
- MAX()
- MIN()
- SUM()

As you can see the windows function doesn't reduce the records of the tables unlike aggregate functions with or without group by clause.



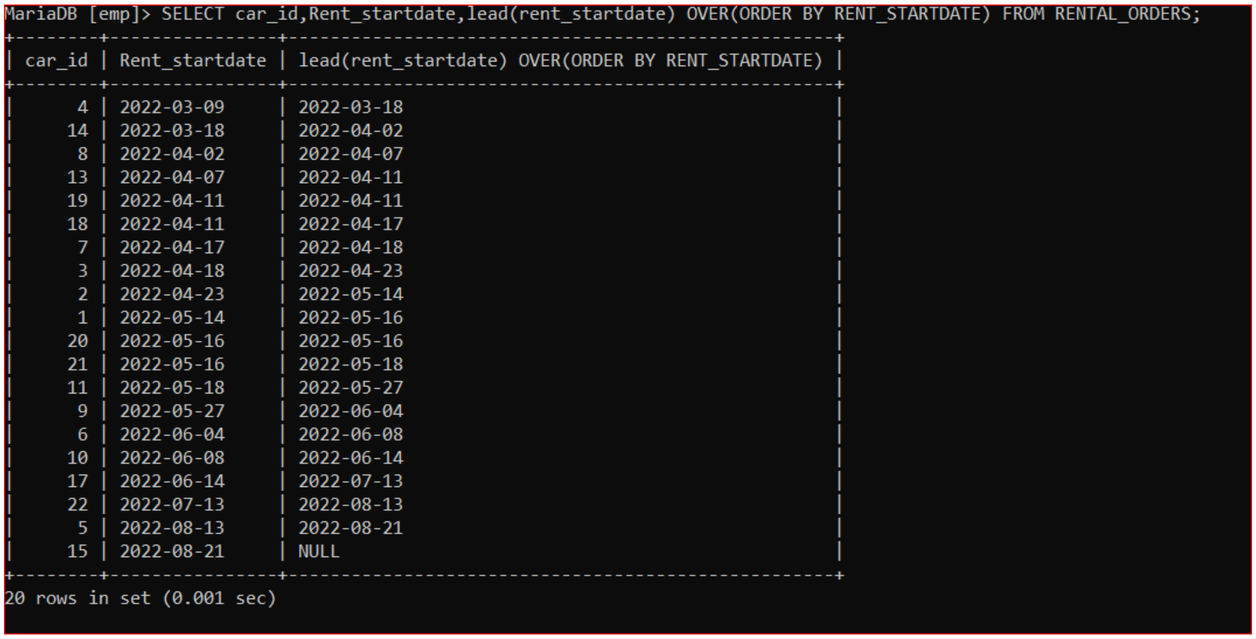
```
MariaDB [emp]> select First_Name,Last_Name,sum(salary) over() from employee;
+------------+-----------+--------------------+
| First_Name | Last_Name | sum(salary) over() |
+------------+-----------+--------------------+
| SUJAY      | SINGH     |             661000 |
| YASH       | BHOSALE   |             661000 |
| RANDEEP    | SINGH     |             661000 |
| SUSHANT    | GADE      |             661000 |
| NILESH     | PANDEY    |             661000 |
| VAIBHAV    | IKKE      |             661000 |
| ROHAN      | SHINDE    |             661000 |
| OMKAR      | MITAKE    |             661000 |
| VIRAJ      | SHEVDE    |             661000 |
| KAVYA      | NAIR      |             661000 |
| DHIRAJ     | AMIN      |             661000 |
| NEHA       | SONI      |             661000 |
| ROHIT      | BHANDARI  |             661000 |
| RIDHI      | NAMBIAR   |             661000 |
| TANMAY     | BHAT      |             661000 |
| SHREYA     | IYER      |             661000 |
| BHAVYA     | GANDHI    |             661000 |
| GOVIND     | BHASKAR   |             661000 |
| VIDYA      | SHETTY    |             661000 |
+------------+-----------+--------------------+
19 rows in set (0.021 sec)
```

TO FIND THE EMPLOYEE WITH LEAST SALARY WITH WINDOWS FUNCTION.



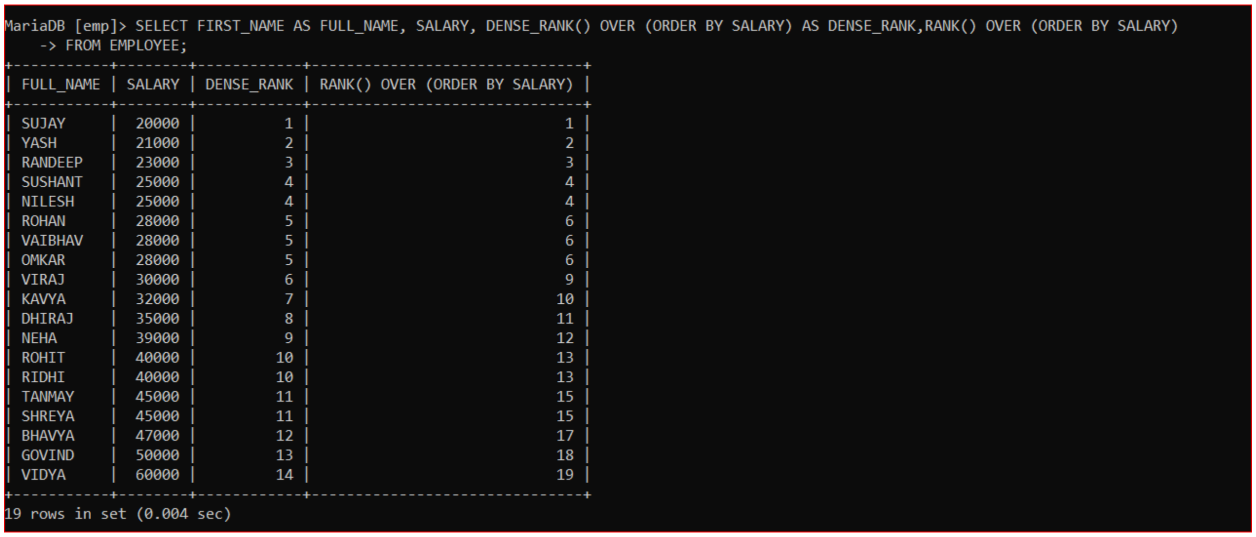
```
MariaDB [emp]> select First_Name,SalaryFIRST_VALUE(First_Name) over(ORDER BY SALARY) from employee;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near 'over(ORDER BY SALARY) from employee' at line
MariaDB [emp]> select First_Name,Salary,FIRST_VALUE(First_Name) over(ORDER BY SALARY) from employee;
+------------+--------+-----------------------------------------------+
| First_Name | Salary | FIRST_VALUE(First_Name) over(ORDER BY SALARY) |
+------------+--------+-----------------------------------------------+
| SUJAY      |  20000 | SUJAY                                         |
| YASH       |  21000 | SUJAY                                         |
| RANDEEP    |  23000 | SUJAY                                         |
| NILESH     |  25000 | SUJAY                                         |
| SUSHANT    |  25000 | SUJAY                                         |
| ROHAN      |  28000 | SUJAY                                         |
| OMKAR      |  28000 | SUJAY                                         |
| VAIBHAV    |  28000 | SUJAY                                         |
| VIRAJ      |  30000 | SUJAY                                         |
| KAVYA      |  32000 | SUJAY                                         |
| DHIRAJ     |  35000 | SUJAY                                         |
| NEHA       |  39000 | SUJAY                                         |
| ROHIT      |  40000 | SUJAY                                         |
| RIDHI      |  40000 | SUJAY                                         |
| TANMAY     |  45000 | SUJAY                                         |
| SHREYA     |  45000 | SUJAY                                         |
| BHAVYA     |  47000 | SUJAY                                         |
| GOVIND     |  50000 | SUJAY                                         |
| VIDYA      |  60000 | SUJAY                                         |
+------------+--------+-----------------------------------------------+
19 rows in set (0.001 sec)
```

FIND THE NEXT RENTAL DATE OF RENTAL ORDERS TABLE BY USING LEAD FUNCTION.



```
MariaDB [emp]> SELECT car_id,Rent_startdate,lead(rent_startdate) OVER(ORDER BY RENT_STARTDATE) FROM RENTAL_ORDERS;
+--------+----------------+----------------------------------------------------+
| car_id | Rent_startdate | lead(rent_startdate) OVER(ORDER BY RENT_STARTDATE) |
+--------+----------------+----------------------------------------------------+
|      4 | 2022-03-09     | 2022-03-18                                         |
|     14 | 2022-03-18     | 2022-04-02                                         |
|      8 | 2022-04-02     | 2022-04-07                                         |
|     13 | 2022-04-07     | 2022-04-11                                         |
|     19 | 2022-04-11     | 2022-04-11                                         |
|     18 | 2022-04-11     | 2022-04-17                                         |
|      7 | 2022-04-17     | 2022-04-18                                         |
|      3 | 2022-04-18     | 2022-04-23                                         |
|      2 | 2022-04-23     | 2022-05-14                                         |
|      1 | 2022-05-14     | 2022-05-16                                         |
|     20 | 2022-05-16     | 2022-05-16                                         |
|     21 | 2022-05-16     | 2022-05-18                                         |
|     11 | 2022-05-18     | 2022-05-27                                         |
|      9 | 2022-05-27     | 2022-06-04                                         |
|      6 | 2022-06-04     | 2022-06-08                                         |
|     10 | 2022-06-08     | 2022-06-14                                         |
|     17 | 2022-06-14     | 2022-07-13                                         |
|     22 | 2022-07-13     | 2022-08-13                                         |
|      5 | 2022-08-13     | 2022-08-21                                         |
|     15 | 2022-08-21     | NULL                                               |
+--------+----------------+----------------------------------------------------+
20 rows in set (0.001 sec)
```

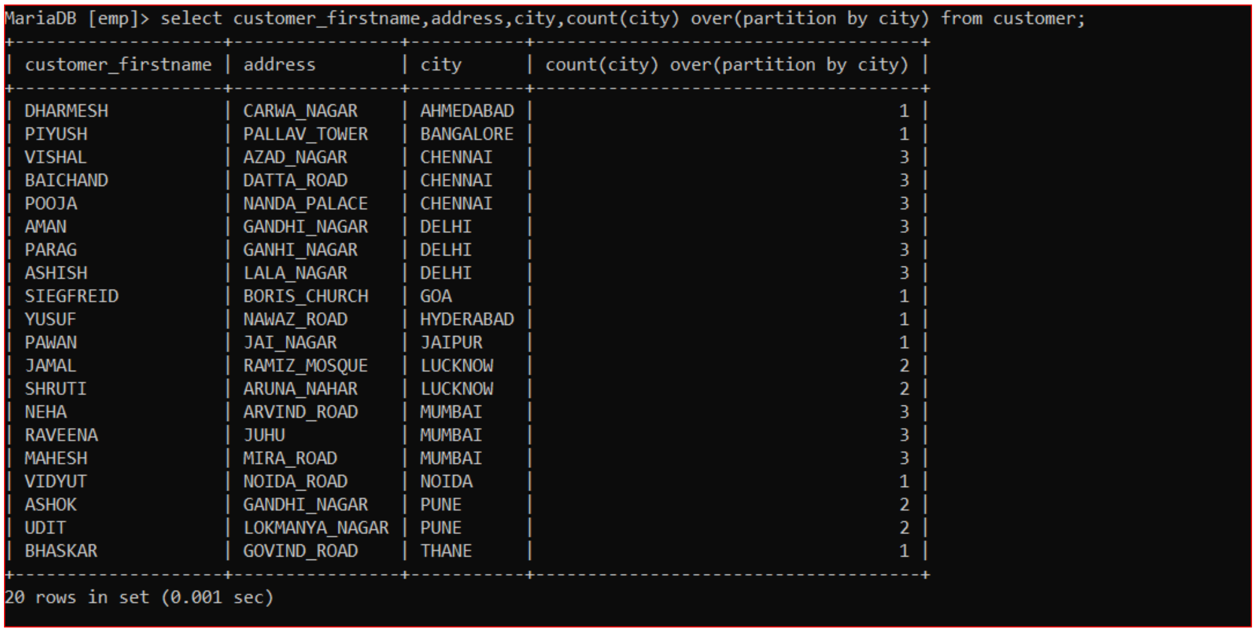
Find the Ranking of SALARY of employee using Dense Rank and Rank



```
MariaDB [emp]> SELECT FIRST_NAME AS FULL_NAME, SALARY, DENSE_RANK() OVER (ORDER BY SALARY) AS DENSE_RANK,RANK() OVER (ORDER BY SALARY)
    -> FROM EMPLOYEE;
+-----------+--------+------------+-----------------------------+
| FULL_NAME | SALARY | DENSE_RANK | RANK() OVER (ORDER BY SALARY) |
+-----------+--------+------------+-----------------------------+
| SUJAY     |  20000 |          1 |                           1 |
| YASH      |  21000 |          2 |                           2 |
| RANDEEP   |  23000 |          3 |                           3 |
| SUSHANT   |  25000 |          4 |                           4 |
| NILESH    |  25000 |          4 |                           4 |
| ROHAN     |  28000 |          5 |                           6 |
| VAIBHAV   |  28000 |          5 |                           6 |
| OMKAR     |  28000 |          5 |                           6 |
| VIRAJ     |  30000 |          6 |                           9 |
| KAVYA     |  32000 |          7 |                          10 |
| DHIRAJ    |  35000 |          8 |                          11 |
| NEHA      |  39000 |          9 |                          12 |
| ROHIT     |  40000 |         10 |                          13 |
| RIDHI     |  40000 |         10 |                          13 |
| TANMAY    |  45000 |         11 |                          15 |
| SHREYA    |  45000 |         11 |                          15 |
| BHAVYA    |  47000 |         12 |                          17 |
| GOVIND    |  50000 |         13 |                          18 |
| VIDYA     |  60000 |         14 |                          19 |
+-----------+--------+------------+-----------------------------+
19 rows in set (0.004 sec)
```

Note :- Dense Rank returns consecutive ranks.

Find the number of customers in a particular city using partition clause of windows function.



```
MariaDB [emp]> select customer_firstname,address,city,count(city) over(partition by city) from customer;
+--------------------+-----------------+-----------+-------------------------------------+
| customer_firstname | address         | city      | count(city) over(partition by city) |
+--------------------+-----------------+-----------+-------------------------------------+
| DHARMESH           | CARWA_NAGAR     | AHMEDABAD |                                   1 |
| PIYUSH             | PALLAV_TOWER    | BANGALORE |                                   1 |
| VISHAL             | AZAD_NAGAR      | CHENNAI   |                                   3 |
| BAICHAND           | DATTA_ROAD      | CHENNAI   |                                   3 |
| POOJA              | NANDA_PALACE    | CHENNAI   |                                   3 |
| AMAN               | GANDHI_NAGAR    | DELHI     |                                   3 |
| PARAG              | GANHI_NAGAR     | DELHI     |                                   3 |
| ASHISH             | LALA_NAGAR      | DELHI     |                                   3 |
| SIEGFREID          | BORIS_CHURCH    | GOA       |                                   1 |
| YUSUF              | NAWAZ_ROAD      | HYDERABAD |                                   1 |
| PAWAN              | JAI_NAGAR       | JAIPUR    |                                   1 |
| JAMAL              | RAMIZ_MOSQUE    | LUCKNOW   |                                   2 |
| SHRUTI             | ARUNA_NAHAR     | LUCKNOW   |                                   2 |
| NEHA               | ARVIND_ROAD     | MUMBAI    |                                   3 |
| RAVEENA            | JUHU            | MUMBAI    |                                   3 |
| MAHESH             | MIRA_ROAD       | MUMBAI    |                                   3 |
| VIDYUT             | NOIDA_ROAD      | NOIDA     |                                   1 |
| ASHOK              | GANDHI_NAGAR    | PUNE      |                                   2 |
| UDIT               | LOKMANYA_NAGAR  | PUNE      |                                   2 |
| BHASKAR            | GOVIND_ROAD     | THANE     |                                   1 |
+--------------------+-----------------+-----------+-------------------------------------+
20 rows in set (0.001 sec)
```