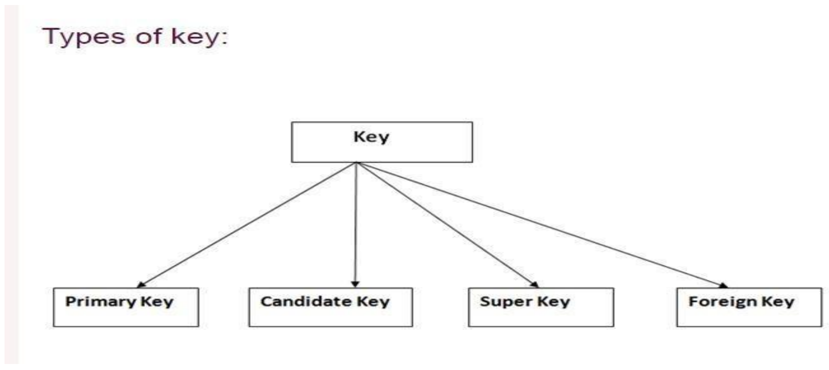
Ex.No:3 *INTEGRITY CONSTRAINT ENFORCEMENT AND SIMPLE SQL QUERIES*

Aim:

To create Primary key, Foreign key, Not null, Unique, Check constraints for the relations and to execute simple SQL queries.

Description



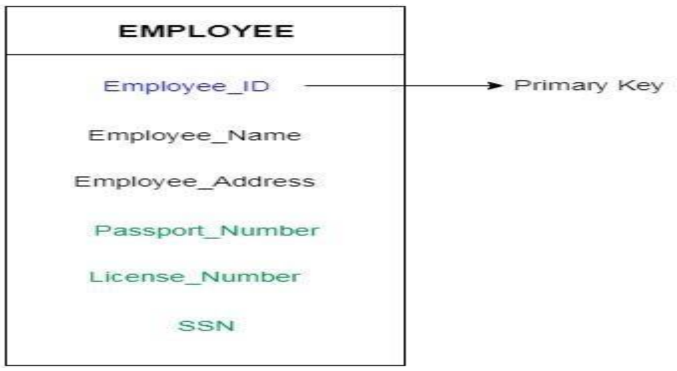
A.KEYS

1. Primary key

It is the first key which is used to identify one and only one instance of an entity uniquely. An entity can contain multiple keys as we saw in PERSON table. The key which is most suitable from those lists become a primary key.

In the EMPLOYEE table, ID can be primary key since it is unique for each employee. In the EMPLOYEE table, we can even select License_Number and Passport_Number as primary key since they are also unique.

For each entity, selection of the primary key is based on requirement and developers.



SYNTAX:

Example:

```
CREATE TABLE Employee(
   ID int NOT NULL PRIMARY KEY,
   LastName varchar(255) NOT NULL,
   FirstName varchar(255),
   Age int
);
```

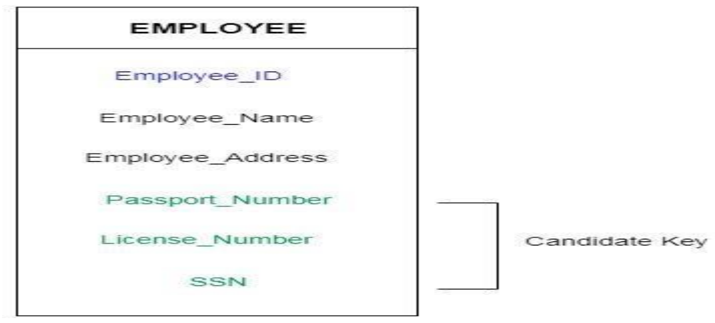
Example

```
ALTER TABLE Emplyee ADD PRIMARY KEY (ID);
```

2. Candidate key ○ A candidate key is an attribute or set of an attribute which can uniquely identify a tuple.

- The remaining attributes except for primary key are considered as a candidate key. The candidate keys are as strong as the primary key.

For example: In the EMPLOYEE table, id is best suited for the primary key. Rest of the attributes like SSN, Passport_Number, and License_Number, etc. are considered as a candidate key.



3. Super Key

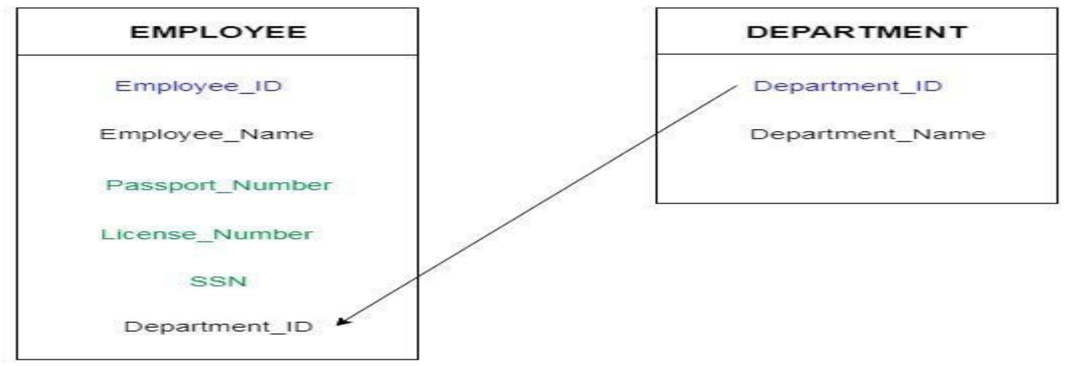
Super key is a set of an attribute which can uniquely identify a tuple. Super key is a superset of a candidate key.

For example: In the above EMPLOYEE table, for(EMPLOEE_ID, EMPLOYEE_NAME) the name of two employees can be the same, but their EMPLYEE_ID can't be the same. Hence, this combination can also be a key.

The super key would be EMPLOYEE-ID, (EMPLOYEE_ID, EMPLOYEE-NAME), etc.

4. Foreign key ○ Foreign keys are the column of the table which is used to point to the primary key of another table.

- In a company, every employee works in a specific department, and employee and department are two different entities. So we can't store the information of the department in the employee table. That's why we link these two tables through the primary key of one table.
- We add the primary key of the DEPARTMENT table, Department_Id as a new attribute in the EMPLOYEE table.
- Now in the EMPLOYEE table, Department_Id is the foreign key, and both the tables are related.



"Persons" table:

PersonID	FName	LName	Age

"Orders" table:

OrderID	OrderNumber	PersonID

The "PersonID" column in the "Persons" table is the PRIMARY KEY in the "Persons" table.

The "PersonID" column in the "Orders" table is a FOREIGN KEY in the "Orders" table.

Example :
```
CREATE TABLE Orders (
   OrderID int NOT NULL PRIMARY KEY,
   OrderNumber int NOT NULL,
   PersonID int  REFERENCES Persons(PersonID)
);
```

INSERT QUERIES

1) Insert Query

Oracle insert query is used to insert records into table.

Insert Statement

In Oracle, INSERT statement is used to add a single record or multiple records into the table.

Syntax: (Inserting a single record using the Values keyword):

INSERT INTO table (column1, column2, ... column_n) VALUES (expression1, expression2, ... expression_n);

Example:

insert into customers values(101,'rahul','delhi');

INSERT INTO suppliers (supplier_id, supplier_name) VALUES (50, 'Flipkart');

Output:

1 row(s) inserted.

Insert Example: By SELECT statement

Syntax:

INSERT INTO table (column1, column2, ... column_n) SELECT expression1, expression2, ... expression_n FROM source_table WHERE conditions;

Parameters:

1) table: The table to insert the records into.
2) column1, column2, ... column_n: The columns in the table to insert values.
3) expression1, expression2, ... expression_n: The values to assign to the columns in the table. So column1 would be assigned the value of expression1, column2 would be assigned the value of expression2, and so on.
4) source_table: The source table when inserting data from another table.
5) conditions: The conditions that must be met for the records to be inserted.

Example:

In this method, we insert values to the "suppliers" table from "customers" table. Both tables are already created with their respective columns.

INSERT INTO suppliers (supplier_id, supplier_name) SELECT age, address FROM customers WHERE age > 20;

Output:

4 row(s) inserted**.**

Example

```
  SELECT count(*)     FROM customers     WHERE age > 20;
```

Output:

```
    Count(*)

    4
```

B.SQL Constraints

SQL constraints are used to specify rules for the data in a table.

Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.

Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.

The following constraints are commonly used in SQL:

- **NOT NULL** - Ensures that a column cannot have a NULL value
- **UNIQUE** - Ensures that all values in a column are different
- **PRIMARY KEY** - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- **FOREIGN KEY** - Uniquely identifies a row/record in another table
- **CHECK** - Ensures that all values in a column satisfies a specific condition •
  DEFAULT - Sets a default value for a column when no value is specified
- **INDEX** - Used to create and retrieve data from the database very quickly

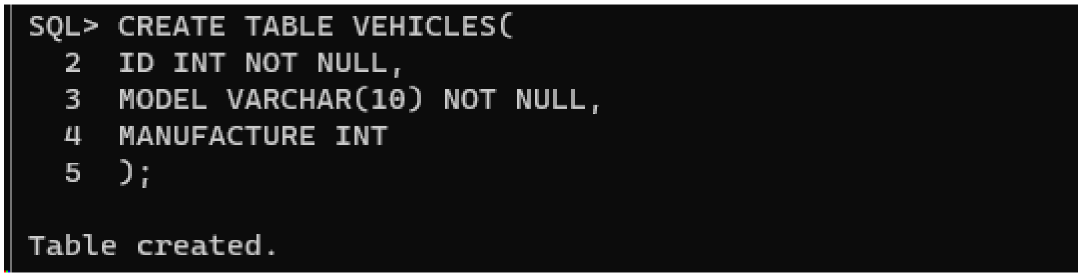
Query:

1.NOT NULL Constraint

Syntax:

Create table tablename (columnname datatype() not null);

Example 1



```
SQL> CREATE TABLE VEHICLES(
  2  ID INT NOT NULL,
  3  MODEL VARCHAR(10) NOT NULL,
  4  MANUFACTURE INT
  5  );

Table created.
```

Example 2:

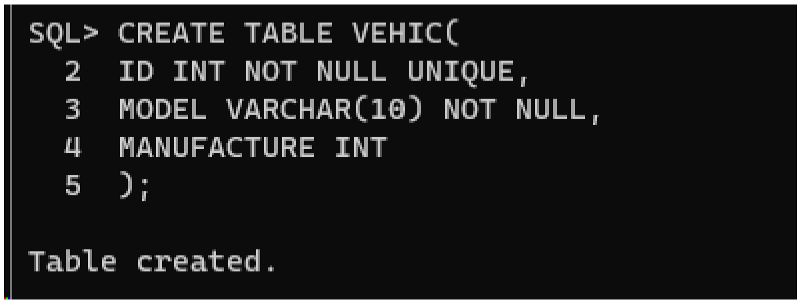
ALTER TABLE Persons MODIFY Age int NOT NULL;

2. UNIQUE CONSTRAINT

Syntax:

Create table tablename (columnname datatype() unique);

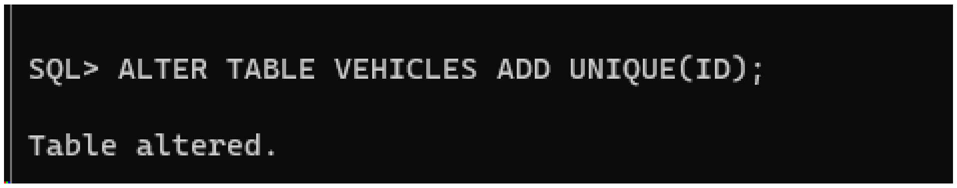
Example 1:



```
SQL> CREATE TABLE VEHIC(
  2  ID INT NOT NULL UNIQUE,
  3  MODEL VARCHAR(10) NOT NULL,
  4  MANUFACTURE INT
  5  );

Table created.
```

Example 2 : (Using Alter Table)

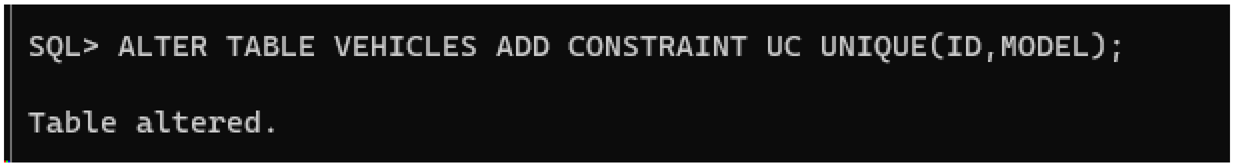
ALTER TABLE Persons ADD UNIQUE (ID);



```
SQL> ALTER TABLE VEHICLES ADD UNIQUE(ID);

Table altered.
```

Example 3: (Multiple Columns)

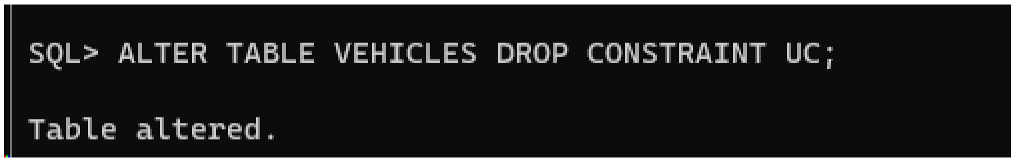
ALTER TABLE Persons ADD CONSTRAINT UC_Person UNIQUE (ID,LastName);



```
SQL> ALTER TABLE VEHICLES ADD CONSTRAINT UC UNIQUE(ID,MODEL);

Table altered.
```

Example 4 : (Drop Constraint)

ALTER TABLE Persons DROP CONSTRAINT UC_Person;



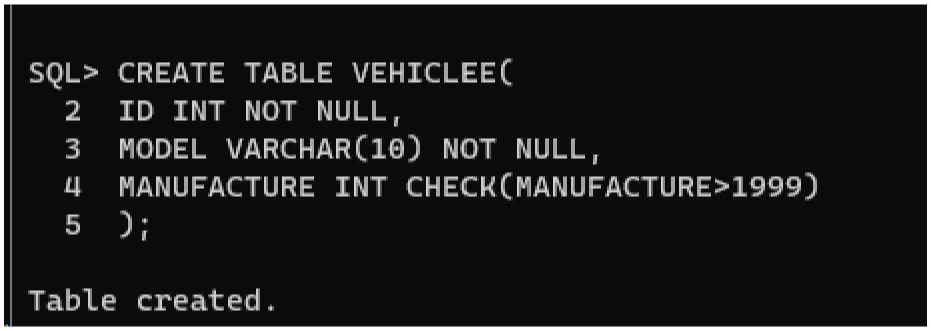
```
SQL> ALTER TABLE VEHICLES DROP CONSTRAINT UC;

Table altered.
```

3. CHECK Constraint

Syntax

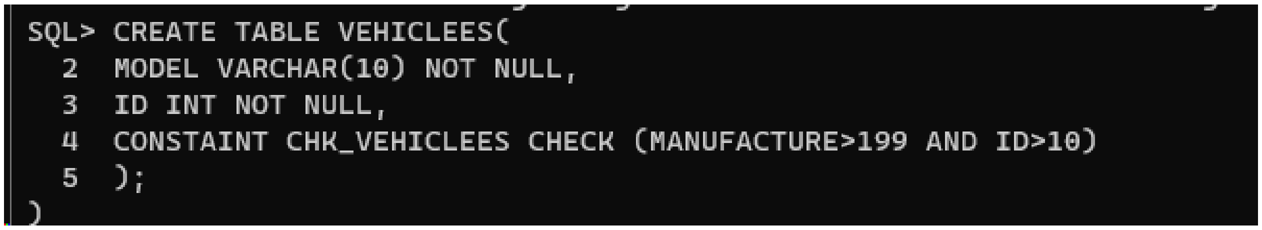
Create table tablename(columnname datatype() CHECK(condition));

Example 1 :



```
SQL> CREATE TABLE VEHICLEE(
  2  ID INT NOT NULL,
  3  MODEL VARCHAR(10) NOT NULL,
  4  MANUFACTURE INT CHECK(MANUFACTURE>1999)
  5  );

Table created.
```

Example 2 : (Multiple Columns)



```
SQL> CREATE TABLE VEHICLEES(
  2  MODEL VARCHAR(10) NOT NULL,
  3  ID INT NOT NULL,
  4  CONSTAINT CHK_VEHICLEES CHECK (MANUFACTURE>199 AND ID>10)
  5  );
)
```

Example 3:

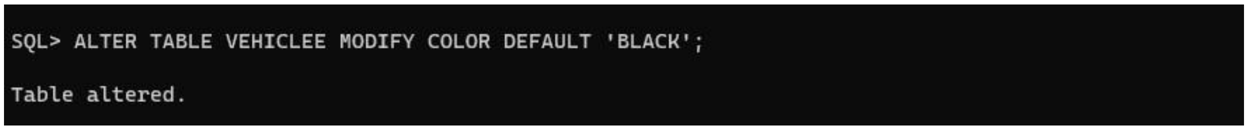
ALTER TABLE Persons
ADD CONSTRAINT CHK_PersonAge CHECK (Age>=18 AND City='Sandnes');

5. DEFAULT CONSTRAINT

Example

ALTER TABLE Persons

MODIFY City DEFAULT 'Sandnes';

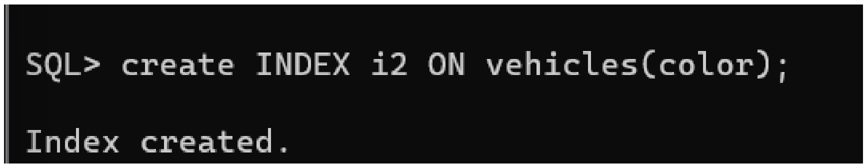


```
SQL> ALTER TABLE VEHICLEE MODIFY COLOR DEFAULT 'BLACK';

Table altered.
```

5.INDEX

Syntax:

CREATE INDEX *index_name* ON
table_name (*column1*, *column2*, ...);



```
SQL> create INDEX i2 ON vehicles(color);

Index created.
```

Example:

CREATE INDEX idx_lastname ON
Persons (LastName); **SIMPLE**
QUERIES

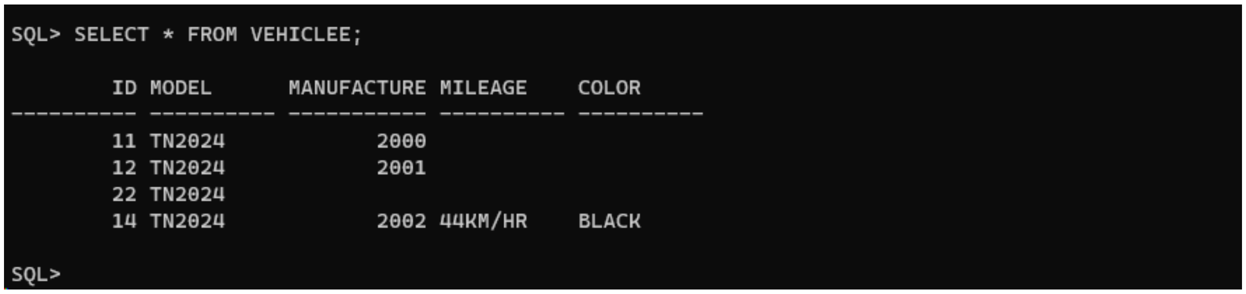
1)Select Query

Oracle select query is used to fetch records from database.

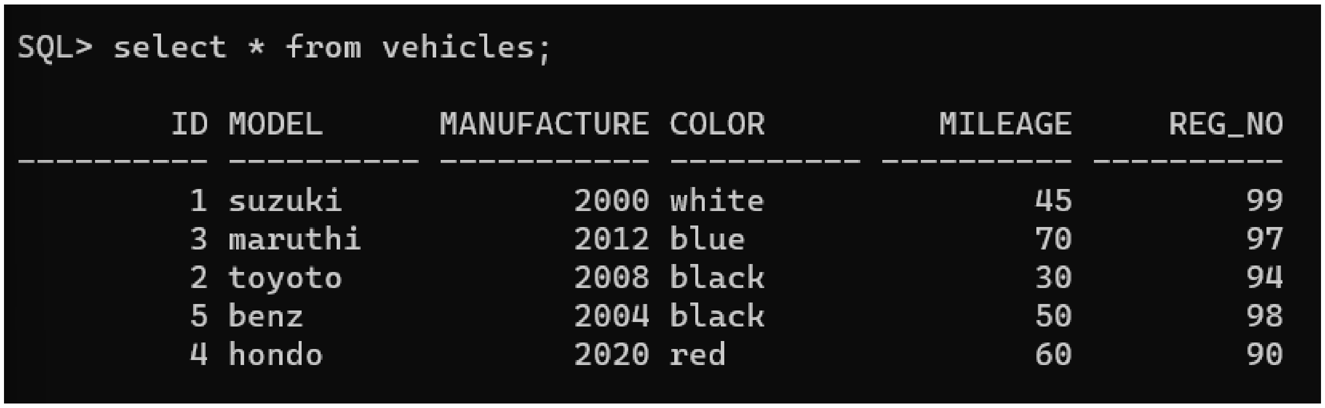
Syntax:
SELECT expressions **FROM** tables **WHERE** conditions;

Example:

SELECT * from customers;



```
SQL> SELECT * FROM VEHICLEE;

        ID MODEL      MANUFACTURE MILEAGE    COLOR
---------- ---------- ----------- ---------- ----------
        11 TN2024            2000
        12 TN2024            2001
        22 TN2024
        14 TN2024            2002 44KM/HR    BLACK

SQL>
```



```
SQL> select * from vehicles;

        ID MODEL      MANUFACTURE COLOR         MILEAGE     REG_NO
---------- ---------- ----------- ---------- ---------- ----------
         1 suzuki            2000 white              45         99
         3 maruthi           2012 blue               70         97
         2 toyoto            2008 black              30         94
         5 benz              2004 black              50         98
         4 hondo             2020 red                60         90
```

WHERE clause uses some conditional selection

=	Equal
>	greater than
<	less than
>=	greater than or equal
<=	less than or equal
< >	not equal to

2.SQL SELECT DISTINCT

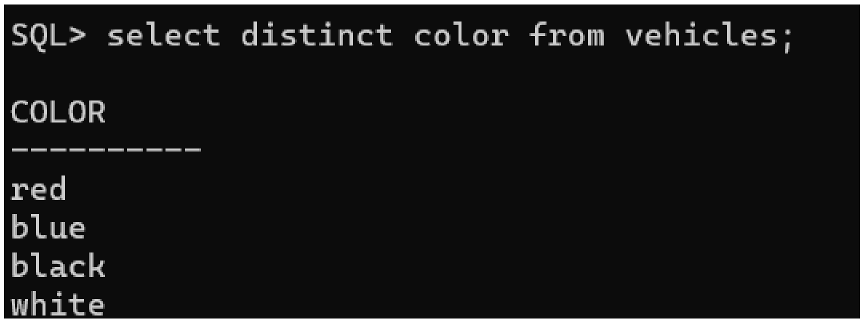
The **SQL DISTINCT command** is used with SELECT key word to retrieve only distinct or unique data.

In a table, there may be a chance to exist a duplicate value and sometimes we want to retrieve only unique values. In such scenarios, SQL SELECT DISTINCT statement is used.

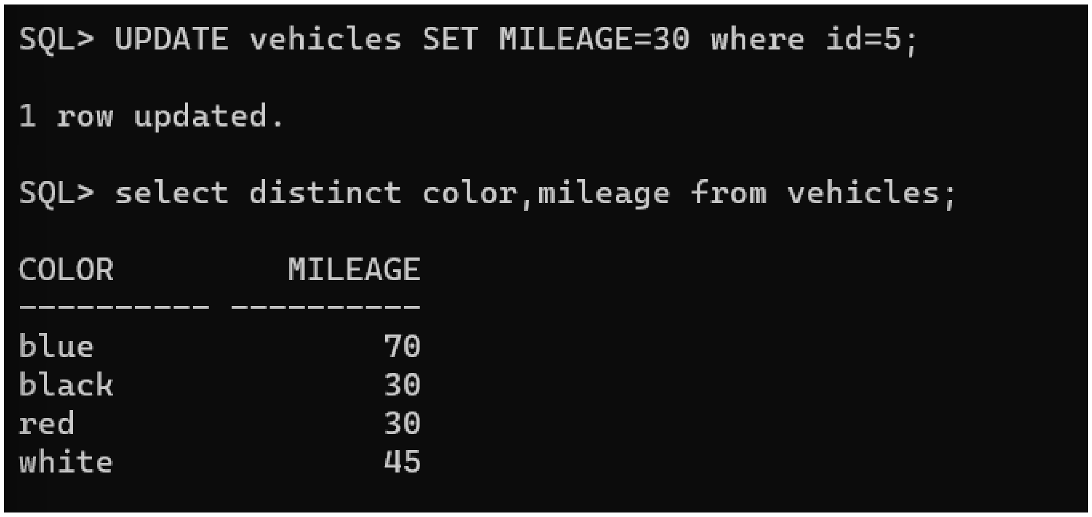
Note: SQL SELECT UNIQUE and SQL SELECT DISTINCT statements are same.

Let's see the syntax of select distinct statement.

SELECT DISTINCT column_name ,column_name **FROM** table_name;



```
SQL> select distinct color from vehicles;

COLOR
----------
red
blue
black
white
```



```
SQL> UPDATE vehicles SET MILEAGE=30 where id=5;

1 row updated.

SQL> select distinct color,mileage from vehicles;

COLOR         MILEAGE
---------- ----------
blue               70
black              30
red                30
white              45
```

Student_Name	Gender	Mobile_Number	HOME_TOWN
Rahul Ojha	Male	7503896532	Lucknow
Disha Rai	Female	9270568893	Varanasi
Sonoo Jaiswal	Male	9990449935	Lucknow

Here is a table of students from where we want to retrieve distinct information For example: distinct home-town.

SELECT DISTINCT home_town **FROM** students Now, it will return two rows.

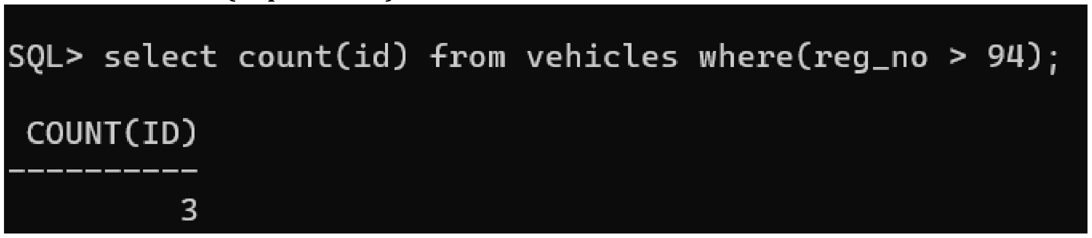
HOME_TOWN

Lucknow
Varanasi

3.SQL SELECT COUNT

The **SQL COUNT()** function is used to return the number of rows in a query.

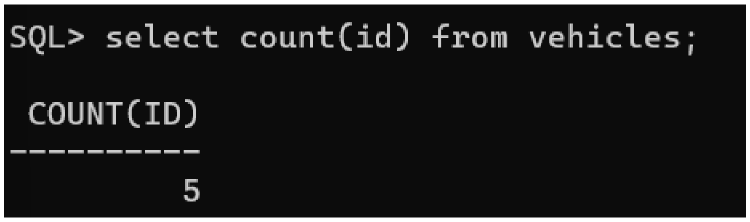
Syntax
SELECT COUNT (expression) FROM tables WHERE conditions;



```
SQL> select count(id) from vehicles where(reg_no > 94);

 COUNT(ID)
----------
         3
```

Example: SQL SELECT COUNT(column_name)

SELECT COUNT(name) FROM employee_table;

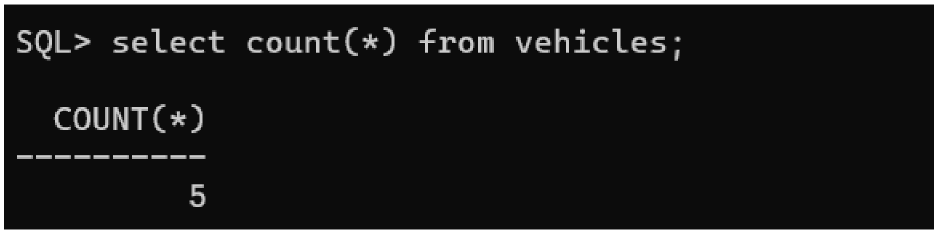


```
SQL> select count(id) from vehicles;

 COUNT(ID)
----------
         5
```

It will return the total number of names of employee_table. But null fields will not be counted.

Example :SQL SELECT COUNT(*)

SELECT COUNT(*) FROM employee_table;

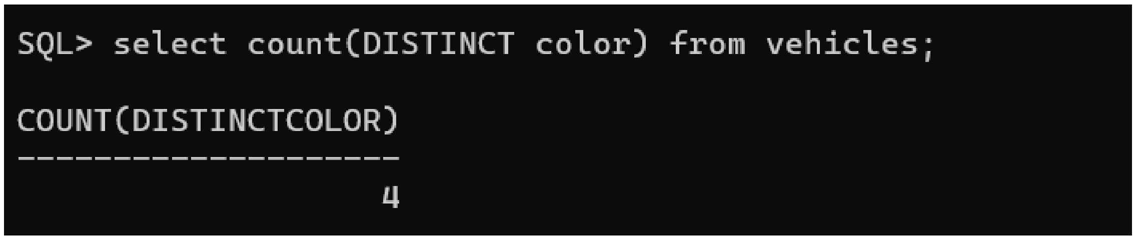


```
SQL> select count(*) from vehicles;

  COUNT(*)
----------
         5
```

The "select count(*) from table" is used to return the number of records in table.

Example : SQL SELECT COUNT(DISTINCT column_name)

SELECT COUNT(DISTINCT name) FROM employee_table;



```
SQL> select count(DISTINCT color) from vehicles;

COUNT(DISTINCTCOLOR)
--------------------
                   4
```

4.SQL SELECT TOP

The SQL SELECT TOP Statement is used to select top data from a table. The top clause specifies that how many rows are returned.

Example:

SELECT TOP 2 * **FROM** employee

5.SQL SELECT FIRST

The SQL first() function is used to return the first value of the selected column.

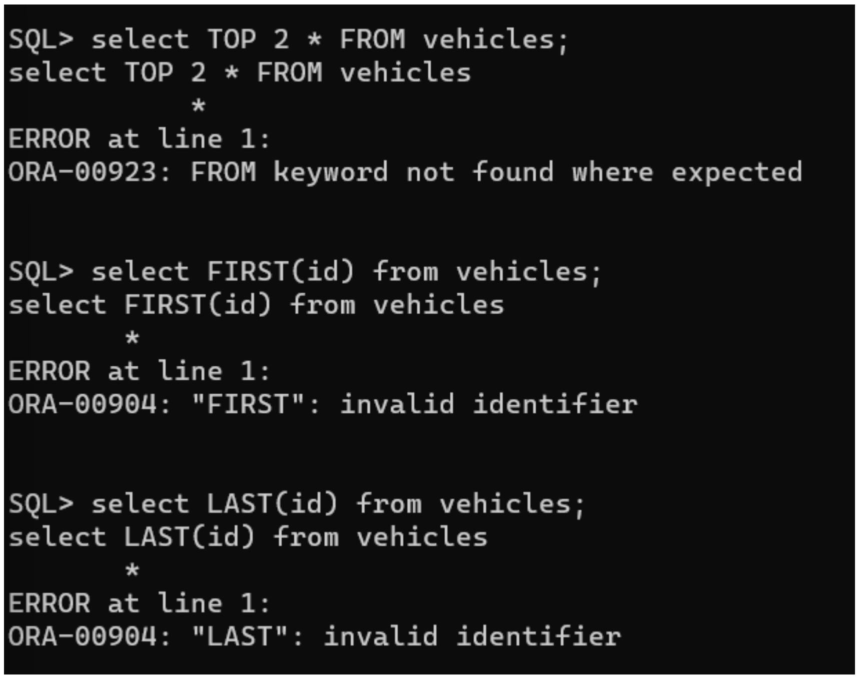
Syntax:

SELECT FIRST(column_name) FROM table_name;

Example:
SELECT FIRST(customer_name) AS first_customer FROM customers;

6.SQL SELECT LAST

Example:
SELECT LAST (CUSTOMER_NAME) **AS** LAST_CUSTOMER **FROM** CUSTOMERS;



```
SQL> select TOP 2 * FROM vehicles;
select TOP 2 * FROM vehicles
           *
ERROR at line 1:
ORA-00923: FROM keyword not found where expected


SQL> select FIRST(id) from vehicles;
select FIRST(id) from vehicles
       *
ERROR at line 1:
ORA-00904: "FIRST": invalid identifier


SQL> select LAST(id) from vehicles;
select LAST(id) from vehicles
       *
ERROR at line 1:
ORA-00904: "LAST": invalid identifier
```

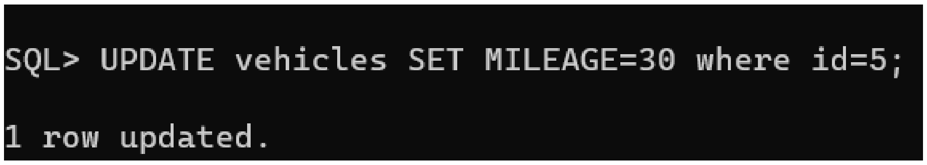
7) Update Query

Update query is used to update records of a table.

Syntax:

UPDATE table_name SET attribute_name = value WHERE condition; **Example:**

update customers set name='bob', city='london' where id=101;



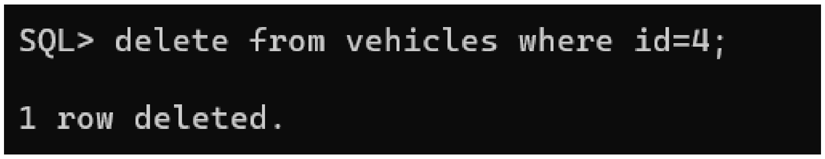
```
SQL> UPDATE vehicles SET MILEAGE=30 where id=5;

1 row updated.
```

8) Delete Query

Oracle update query is used to delete records of a table from database.

Example:

delete from customers where id=101;



```
SQL> delete from vehicles where id=4;

1 row deleted.
```

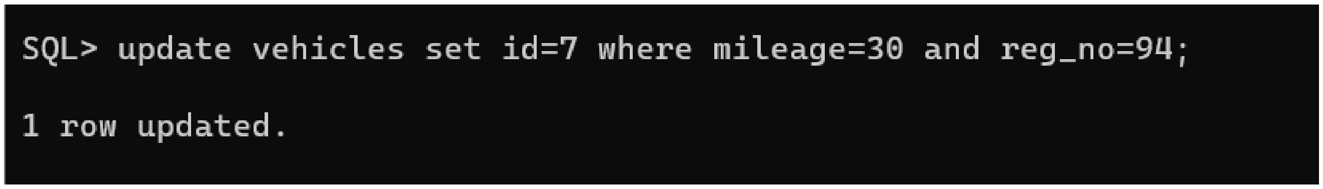
9. SQL AND

The SQL AND condition is used in SQL query to create two or more conditions to be met.

It is used in SQL SELECT, INSERT, UPDATE and DELETE statements.

Syntax:

SELECT columns **FROM** tables **WHERE** condition 1 AND condition 2;

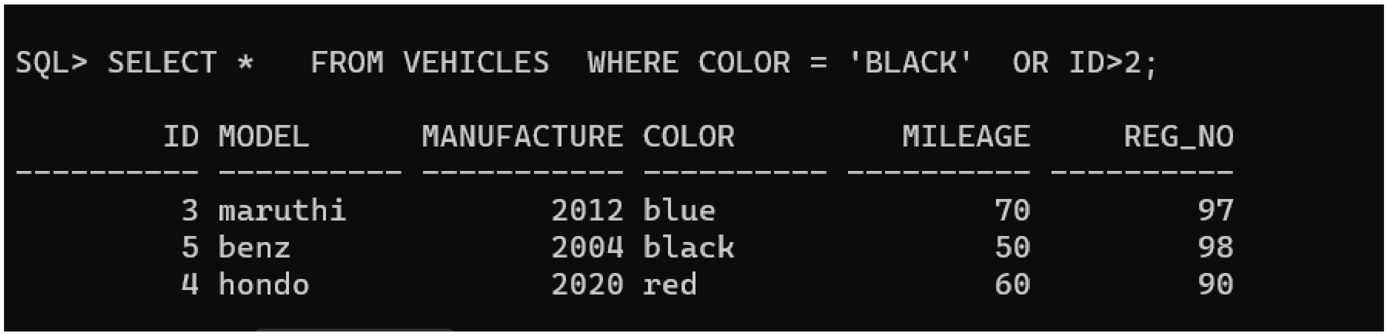


```
SQL> update vehicles set id=7 where mileage=30 and reg_no=94;

1 row updated.
```

Example:**UPDATE** suppliers

SET supplier_name = 'HP' **WHERE** supplier_name = 'IBM' AND offices = 8;

10. SQL OR



```
SQL> SELECT *   FROM VEHICLES  WHERE COLOR = 'BLACK'  OR ID>2;

        ID MODEL      MANUFACTURE COLOR         MILEAGE     REG_NO
---------- ---------- ----------- ---------- ---------- ----------
         3 maruthi           2012 blue               70         97
         5 benz              2004 black              50         98
         4 hondo             2020 red                60         90
```

The **SQL OR condition** is used in a SQL query to create a SQL statement where records are returned when any one of the condition met. It can be used in a SELECT statement, INSERT statement, UPDATE statement or DELETE statement

Syntax:

SELECT columns FROM tables WHERE condition 1 OR condition 2;

Example:

SELECT * FROM suppliers WHERE city = 'New York' OR available_products >= 250**;**

11. SQL ORDER BY Clause

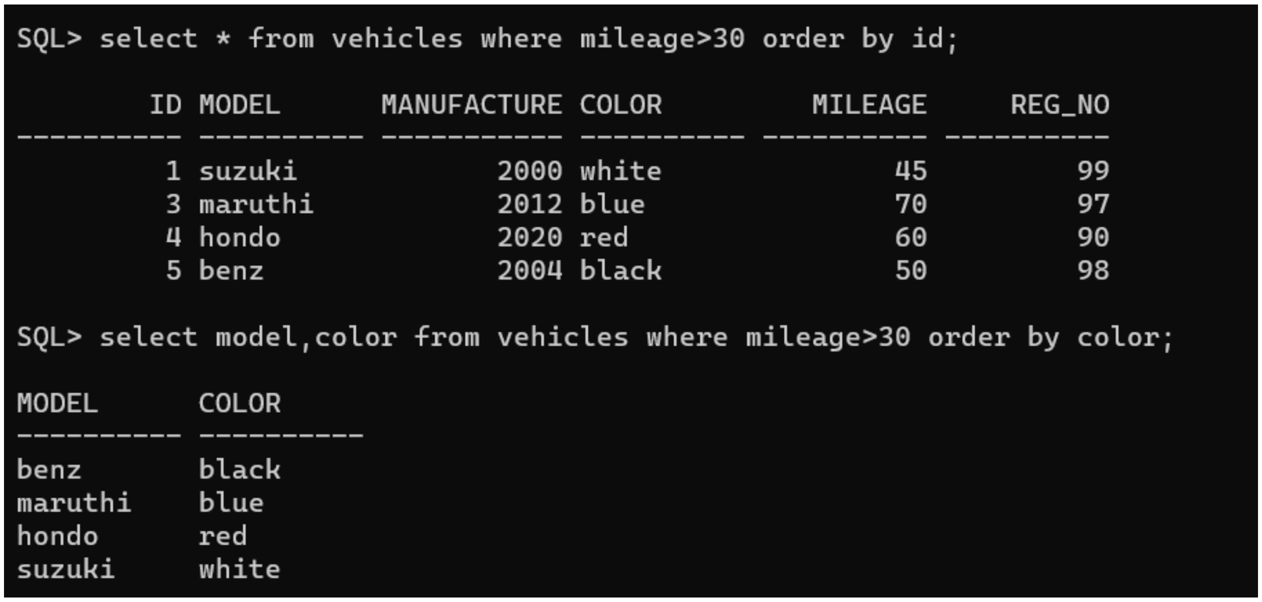
The SQL ORDER BY clause is used for sorting data in ascending and descending order based on one or more columns.

Some databases sort query results in ascending order by default.

Syntax:

SELECT expressions **FROM** tables **WHERE** conditions **ORDER BY** expression [**ASC | DESC**];

Example:



```
SQL> select * from vehicles where mileage>30 order by id;

        ID MODEL      MANUFACTURE COLOR         MILEAGE     REG_NO
---------- ---------- ----------- ---------- ---------- ----------
         1 suzuki            2000 white              45         99
         3 maruthi           2012 blue               70         97
         4 hondo             2020 red                60         90
         5 benz              2004 black              50         98

SQL> select model,color from vehicles where mileage>30 order by color;

MODEL      COLOR
---------- ----------
benz       black
maruthi    blue
hondo      red
suzuki     white
```

Let us take a CUSTOMERS table having the following records:

ID	NAME	AGE	ADDRESS	SALARY
1	Himani gupta	21	Modinagar	22000
2	Shiva tiwari	22	Bhopal	21000
3	Ajeet bhargav	45	Meerut	65000
4	Ritesh yadav	36	Azamgarh	26000
5	Balwant singh	45	Varanasi	36000
6	Mahesh sharma	26	Mathura	22000

Example:

SELECT * **FROM** CUSTOMERS **ORDER BY NAME**, SALARY; This

would produce the following result.

ID	NAME	AGE	ADDRESS	SALARY
3	Ajeet bhargav	45	Meerut	65000
5	Balwant singh	45	Varanasi	36000
1	Himani gupta	21	Modinagar	22000
6	Mahesh sharma	26	Mathura	22000
4	Ritesh yadav	36	Azamgarh	26000
2	Shiva tiwari	22	Bhopal	21000

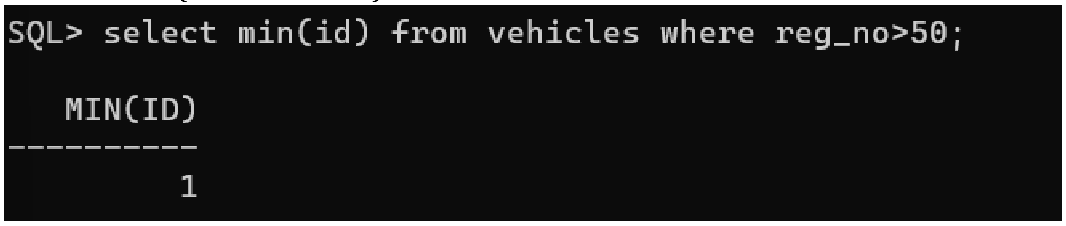
12. The SQL MIN() and MAX() Functions

The MIN() function returns the smallest value of the selected column.

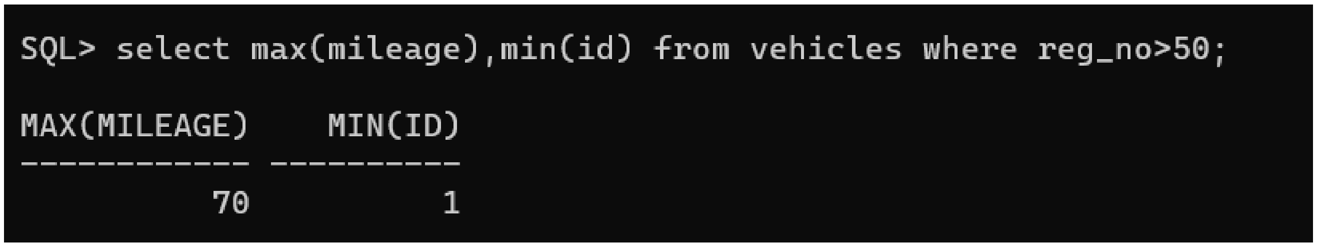
The MAX() function returns the largest value of the selected column.

MIN() Syntax

SELECT MIN(*column_name*) FROM *table_name* WHERE *condition*;



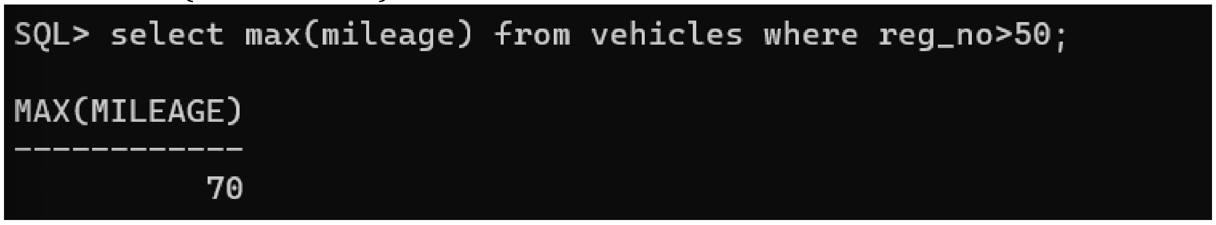
```
SQL> select min(id) from vehicles where reg_no>50;

   MIN(ID)
----------
         1
```



```
SQL> select max(mileage),min(id) from vehicles where reg_no>50;

MAX(MILEAGE)    MIN(ID)
------------ ----------
          70          1
```

MAX() Syntax

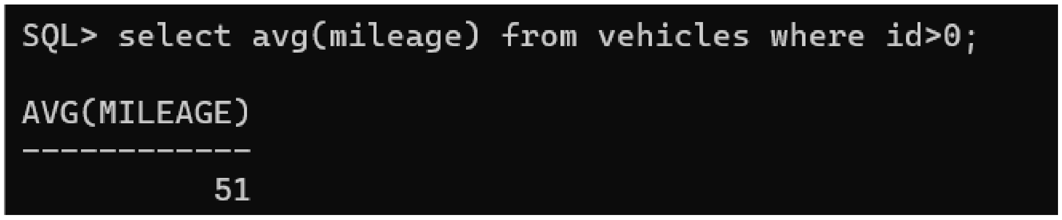
SELECT MAX(*column_name*) FROM *table_name* WHERE *condition*;



```
SQL> select max(mileage) from vehicles where reg_no>50;

MAX(MILEAGE)
------------
          70
```

13. AVG() Syntax

SELECT AVG(*column_name*) FROM *table_name* WHERE *condition*;



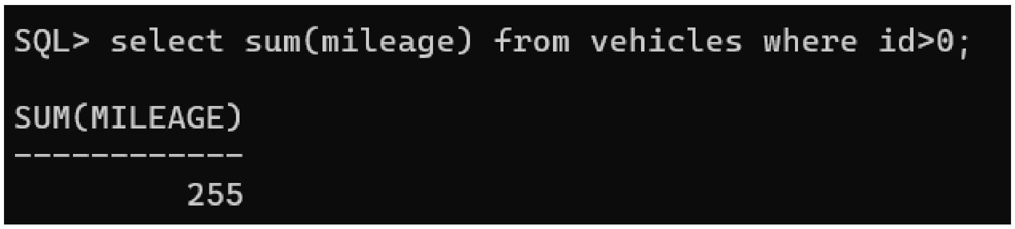
```
SQL> select avg(mileage) from vehicles where id>0;

AVG(MILEAGE)
------------
          51
```

Example:

SELECT AVG(Price) FROM Products;

14. SUM() Syntax

SELECT SUM(*column_name*) FROM *table_name* WHERE *condition*;



```
SQL> select sum(mileage) from vehicles where id>0;

SUM(MILEAGE)
------------
         255
```

Example;

SELECT AVG(Price) FROM Products;

15. The SQL GROUP BY Statement

The GROUP BY statement is often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns.

GROUP BY Syntax

SELECT *column_name(s)* FROM *table_name* WHERE *condition* GROUP BY *column_name(s)* ORDER BY *column_name(s);*

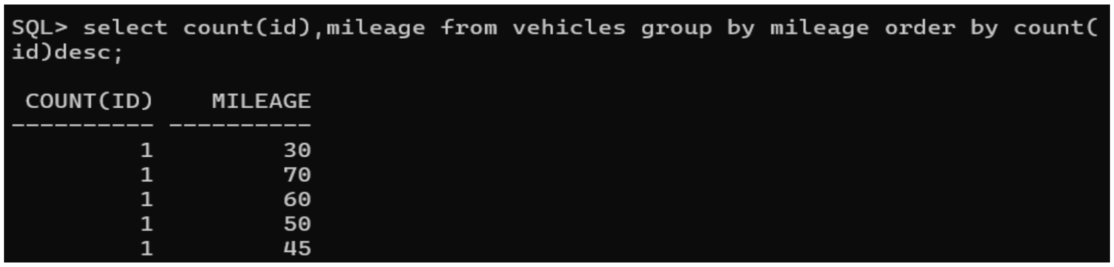
Example

SELECT COUNT(CustomerID), Country FROM Customers

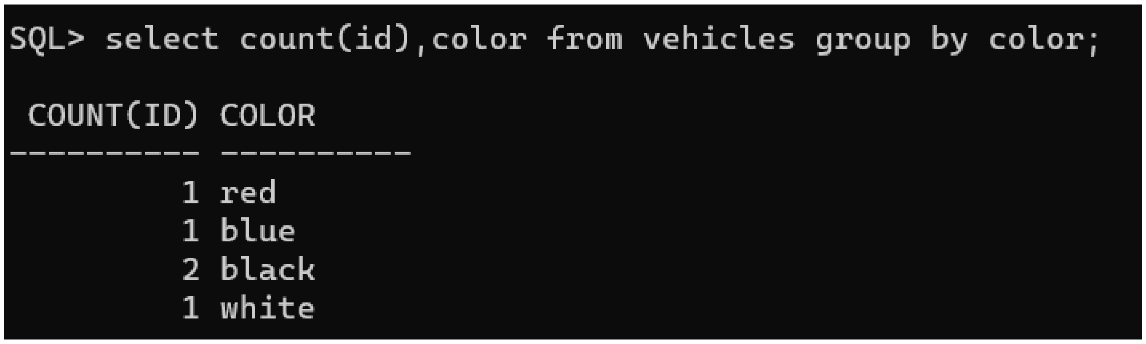
GROUP BY Country;

Example

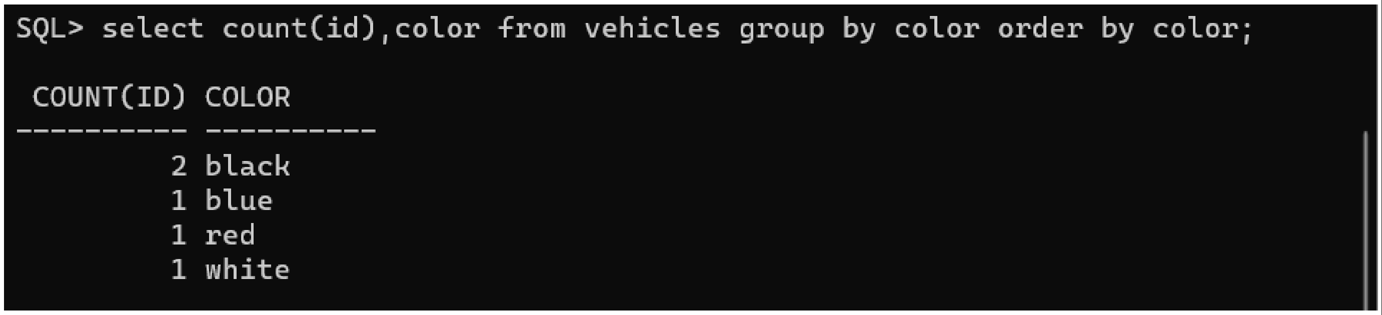
SELECT COUNT(CustomerID), Country FROM Customers GROUP BY Country ORDER BY COUNT(CustomerID) DESC;



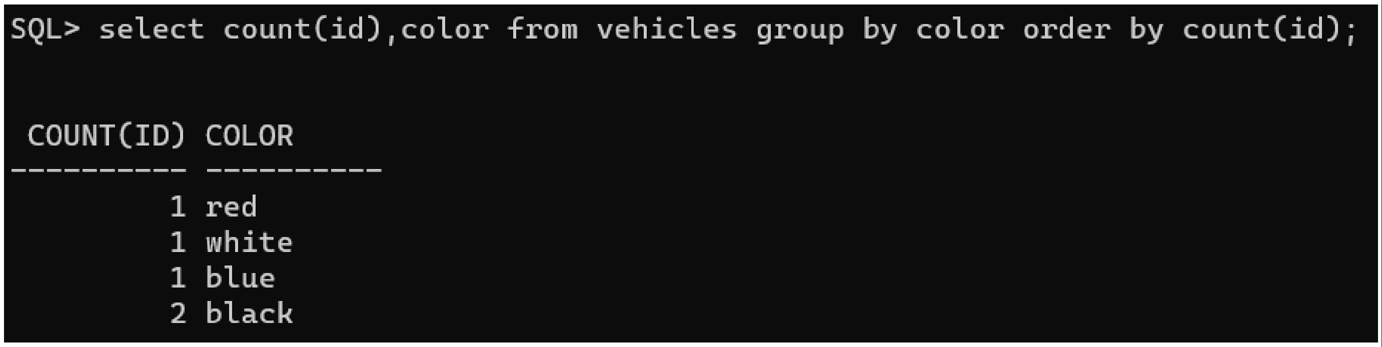
```
SQL> select count(id),mileage from vehicles group by mileage order by count(
id)desc;

 COUNT(ID)    MILEAGE
---------- ----------
         1         30
         1         70
         1         60
         1         50
         1         45
```



```
SQL> select count(id),color from vehicles group by color;

 COUNT(ID) COLOR
---------- ----------
         1 red
         1 blue
         2 black
         1 white
```



```
SQL> select count(id),color from vehicles group by color order by color;

 COUNT(ID) COLOR
---------- ----------
         2 black
         1 blue
         1 red
         1 white
```



```
SQL> select count(id),color from vehicles group by color order by count(id);


 COUNT(ID) COLOR
---------- ----------
         1 red
         1 white
         1 blue
         2 black
```

16. The SQL HAVING Clause

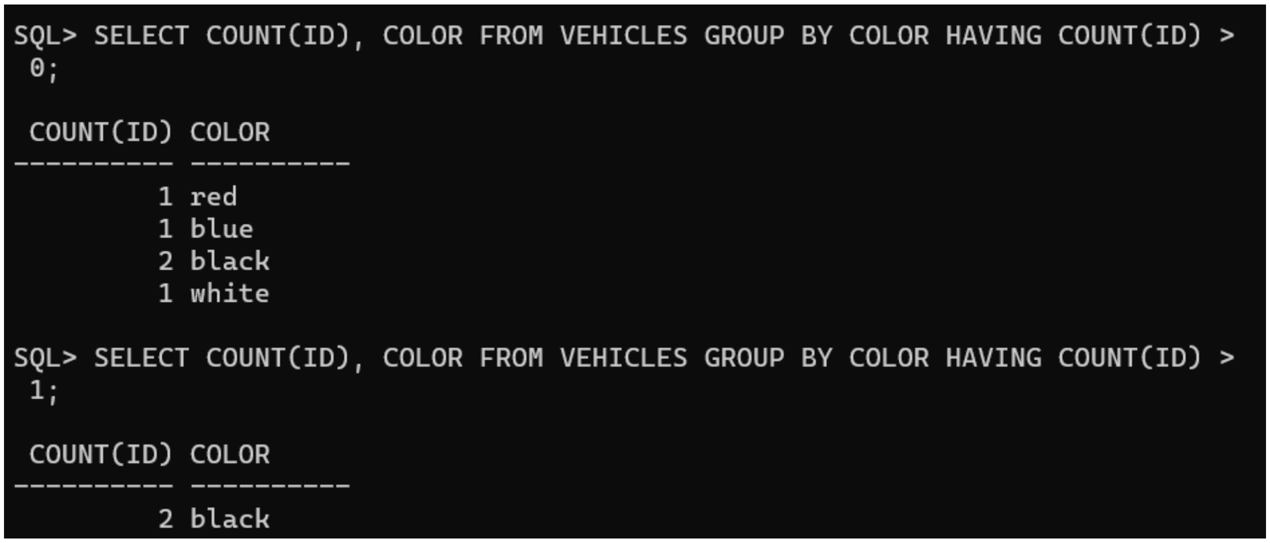
The HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions.

HAVING Syntax

SELECT *column_name(s)* FROM *table_name* WHERE *condition* GROUP BY *column_name(s)* HAVING *condition* ORDER BY *column_name(s);*

Example:

SELECT COUNT(CustomerID), Country FROM Customers GROUP BY Country HAVING COUNT(CustomerID) > 5;



```
SQL> SELECT COUNT(ID), COLOR FROM VEHICLES GROUP BY COLOR HAVING COUNT(ID) >
 0;

 COUNT(ID) COLOR
---------- ----------
         1 red
         1 blue
         2 black
         1 white

SQL> SELECT COUNT(ID), COLOR FROM VEHICLES GROUP BY COLOR HAVING COUNT(ID) >
 1;

 COUNT(ID) COLOR
---------- ----------
         2 black
```

RESULT:
Thus the PRIMARY KEY,FOREIGN KEY,NOT NULL,UNIQUE,Check constraints were created for the relations and simple SQL Queries were executed