use database "DEMO\_DATABASE";

CREATE OR REPLACE TABLE Employee

(

EmployeeID INT PRIMARY KEY,

EmployeeName VARCHAR(100) NOT NULL,

Gender VARCHAR(1) NOT NULL,

StateCode VARCHAR(20) NOT NULL,

Salary NUMBER(10,2) NOT NULL

) ;

describe table Employee;

--INSERTING RECORDS

INSERT INTO EMPLOYEE VALUES (211, 'Manisha', 'F', 'IN', 80000.0000);

INSERT INTO EMPLOYEE VALUES (212, 'Vikas', 'M', 'IN', 5000.0000);

INSERT INTO EMPLOYEE VALUES (201, 'Jerome', 'M', 'FL', 83000.0000);

INSERT INTO EMPLOYEE VALUES (202, 'Ray', 'M', 'AL', 88000.0000);

INSERT INTO EMPLOYEE VALUES (203, 'Stella', 'F', 'AL', 76000.0000);

INSERT INTO EMPLOYEE VALUES (204, 'Gilbert', 'M', 'Ar', 42000.0000);

INSERT INTO EMPLOYEE VALUES (205, 'Edward', 'M', 'FL', 93000.0000);

INSERT INTO EMPLOYEE VALUES (206, 'Ernest', 'F', 'Al', 64000.0000);

INSERT INTO EMPLOYEE VALUES (207, 'Jorge', 'F', 'IN', 75000.0000);

INSERT INTO EMPLOYEE VALUES (208, 'Nicholas', 'F', 'Ge', 71000.0000);

INSERT INTO EMPLOYEE VALUES (209, 'Lawrence', 'M', 'IN', 95000.0000);

INSERT INTO EMPLOYEE VALUES (210, 'Salvador', 'M', 'Co', 75000.0000);

SELECT \* FROM EMPLOYEE;

--The SQL CASE statement allows you to perform IF-THEN-ELSE functionality within an SQL statement.

-- The CASE statement allows you to perform an IF-THEN-ELSE check within an SQL statement.

/\* It’s good for displaying a value in the SELECT query based on logic that you have defined.

As the data for columns can vary from row to row, using a CASE SQL expression can help make your data more readable and useful to the user or to the application. "\*/

-- It’s quite common if you’re writing complicated queries or doing any kind of ETL work.

-- SYNTAX

/\* The syntax of the SQL CASE expression is:

CASE [expression]

WHEN condition\_1 THEN result\_1

WHEN condition\_2 THEN result\_2 ...

WHEN condition\_n THEN result\_n

ELSE result

END case\_name

\*/

/\*

The CASE statement and comparison operator

In this format of a CASE statement in SQL, we can evaluate a condition using comparison operators. Once this condition is satisfied, we get an expression from corresponding THEN in the output.

Suppose we have a salary band for each designation.

If employee salary is in between a particular range,

we want to get designation using a Case statement.

In the following query, we are using a comparison operator and evaluate an expression.

\*/

CREATE OR REPLACE TABLE AJ\_EMPLOYEE\_DESIGNATION\_BUCKET AS

Select \*,

CASE

WHEN Salary >=10000 AND Salary < 30000 THEN 'Data Analyst Trainee'

WHEN Salary >=30000 AND Salary < 50000 THEN 'Data Analyst'

WHEN Salary >=50000 AND Salary < 80000 THEN 'Consultant'

WHEN Salary >=80000 AND Salary < 100000 THEN 'Senior Consultant'

WHEN Salary >= 100000 THEN 'Senior Folks'

Else 'Contractor'

END AS Designation

from Employee;

select \* from AJ\_EMPLOYEE\_DESIGNATION\_BUCKET;

grant select on AJ\_EMPLOYEE\_DESIGNATION\_BUCKET to public;

/\* Case Statement with Order by clause

We can use Case statement with order by clause as well.

In SQL, we use Order By clause to sort results in ascending or descending order.

Suppose in a further example; we want to sort result in the following method.

For Female employee, employee salaries should come in descending order

For Male employee, we should get employee salaries in ascending order

We can define this condition with a combination of Order by and Case statement.

In the following query, you can see we specified Order By and Case together.

We defined sort conditions in case expression. \*/

Select EmployeeName,Gender,Salary

from Employee

ORDER BY CASE Gender WHEN 'F' THEN Salary END DESC ,

CASE WHEN Gender = 'M' THEN Salary

END;

/\*

Case Statement in SQL with Group by clause

We can use a Case statement with Group By clause as well.

Suppose we want to group employees based on their salary.

We further want to calculate the minimum and maximum salary for a particular range of employees.

In the following query, you can see that we have Group By clause and it contains i with the condition to get the required output.

\*/

DESCRIBE TABLE EMPLOYEE;

Select

CASE

WHEN Salary >=10000 AND Salary < 30000 THEN 'Data Analyst Trainee'

WHEN Salary >=30000 AND Salary < 50000 THEN 'Data Analyst'

WHEN Salary >=50000 AND Salary < 80000 THEN 'Consultant'

WHEN Salary >=80000 AND Salary < 100000 THEN 'Senior Consultant'

WHEN Salary >= 100000 THEN 'Senior Folks'

Else 'Contractor'

END AS Designation,

Min(salary) as MinimumSalary,

Max(Salary) as MaximumSalary

from Employee

Group By

CASE

WHEN Salary >=10000 AND Salary < 30000 THEN 'Data Analyst Trainee'

WHEN Salary >=30000 AND Salary < 50000 THEN 'Data Analyst'

WHEN Salary >=50000 AND Salary < 80000 THEN 'Consultant'

WHEN Salary >=80000 AND Salary < 100000 THEN 'Senior Consultant'

WHEN Salary >= 100000 THEN 'Senior Folks'

Else 'Contractor'

END;

/\* Case Statement limitations

We cannot control the execution flow of stored procedures, functions using a Case statement in SQL

We can have multiple conditions in a Case statement;

however, it works in a sequential model. If one condition is satisfied, it stops checking further conditions

We cannot use a Case statement for checking NULL values in a table

\*/