SQL FUNCTIONS

IOTA ACADEMY

CLASS OUTLINE

- IF Function
- CAST Function
- COALESCE Function
- HAVING Clause
- CASE Function

MISCELLANEOUS FUNCTIONS

IF Function: The IF() function returns a value if a condition is TRUE, or another value if a condition is FALSE.

Syntax :	Example :
SELECT IF(condition, value_if_true, value_if_false);	**SELECT** IF(8>4, "TRUE","FALSE");

```
#Example:
SELECT ProductID, ProductName, IF(Price>20, "Yes", "No") AS `Luxury Items`
FROM Products;
```

MISCELLANEOUS FUNCTIONS…

CAST Function: This is used to convert a value from one datatype to another datatype.

- This function is similar to the **CONVERT()** function.
- It converts the value into a DATE datatype in the **"YYYY-MM-DD"** format. It supports the range of DATE in '1000-01-01' to '9999-12-31'.
- ***Note:*** Generally MySQL implicitly converts datatypes of values for calculations/ queries.

Syntax :	Example :
SELECT CAST(value **AS** datatype)	**SELECT CAST**(100 **AS** CHAR);

MISCELLANEOUS FUNCTIONS…

The **COALESCE()** function returns the **first non-null value** in a list.

Syntax :	**Example :**
SELECT COALESCE(val1, val2, val3, val4,….);	**SELECT COALESCE**(NULL, NULL, 'IOTA', NULL, 2022);

HAVING CLAUSE

- The HAVING clause is used **to specify filter conditions for a group of rows or aggregates.**
- The HAVING clause is often used with the **GROUP BY** clause to filter groups based on a specified condition. If you omit the GROUP BY clause, the HAVING clause behaves like the **WHERE** clause.
- Notice that the HAVING clause applies a filter condition to each group of rows, while the WHERE clause applies the filter condition to each individual row.

HAVING CLAUSE...

Syntax :

```
SELECT column_name(s)
FROM table_name
GROUP BY column_name(s)
HAVING condition;
```

Example :

```
SELECT COUNT(CustomerID), City
FROM Customer
GROUP BY City
HAVING COUNT(CustomerID) >= 3 ;
```

HAVING CLAUSE…

- **Note:** If we don’t use **GROUP BY** clause in a query, **HAVING** clause works like a **WHERE** clause in some cases.

```
#Query:1
SELECT CustomerID, City
FROM Customer
HAVING CustomerID >= 3;
```

```
#Query:2
SELECT CustomerID, City
FROM Customer
WHERE CustomerID >= 3;
```

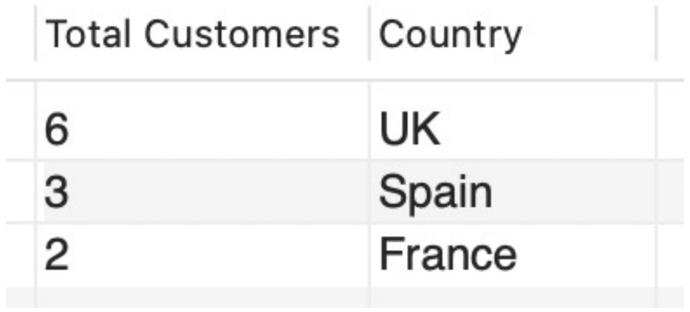
HAVING CLAUSE…

Que: Write a query to find the total number of Customers(Country wise) living in Cities like London, Paris, Madrid & Seattle and having at least 2 registered customers.

HAVING CLAUSE…

Que: Write a query to find the total number of Customers(Country wise) living in Cities like London, Paris, Madrid & Seattle and having at least 2 registered customers.

Expected Output:



Total Customers	Country
6	UK
3	Spain
2	France

HAVING CLAUSE…

- **Solution:**

```sql
SELECT COUNT(CustomerID) AS `Total Customers`, Country
FROM Customer
WHERE City IN ('London', 'Madrid', 'Paris', 'Seattle')
GROUP BY Country
HAVING COUNT(CustomerID)>=2;
```

HAVING VS WHERE CLAUSE

Comparison Basis	WHERE Clause	HAVING Clause
Definition	It is used to perform filtration on **individual rows**.	It is used to perform filtration on **groups**.
Basic	It is implemented in **row operations.**	It is implemented in **column operations.**
Data fetching	The WHERE clause **fetches the specific data** from particular rows based on the specified condition	The HAVING clause first **fetches the complete data**. It then separates them according to the given condition.
Aggregate Functions	It does not work with aggregate functions.	It can work with aggregate functions.
Act as	The WHERE clause acts as a **pre-filter.**	The HAVING clause acts as a **post-filter.**
Used with	We can use the WHERE clause with the **SELECT**, **UPDATE**, and **DELETE** statements.	The HAVING clause can **only** be used with the **SELECT** statement.
GROUP BY	The GROUP BY clause comes **after** the WHERE clause.	The GROUP BY clause comes **before** the HAVING clause.

CASE FUNCTION

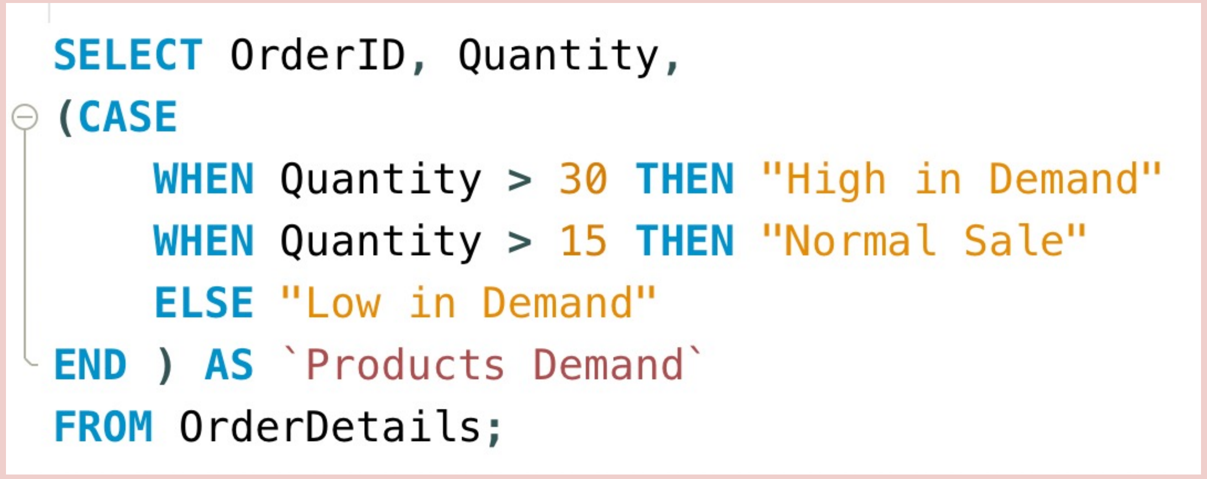
- CASE function is a **control flow structure** that allows you to **add if-else logic** to a query.
- CASE statement goes through conditions and when condition is satisfied return a value corresponding to that condition.
- When a condition is satisfied it stops reading further and returns the output.
- This function returns the statement in the **else part** if none of the stated conditions are true and returns **NULL** if none of the stated conditions are true as well as **there is no else part** also.

CASE FUNCTION…

Syntax :

```
SELECT column1, column2,
(CASE
   WHEN condition1 THEN output1
   WHEN condition2 THEN output2
   WHEN condition3 THEN output3
   ELSE result
END)
FROM table_name;
```

Example :



```
SELECT OrderID, Quantity,
(CASE
     WHEN Quantity > 30 THEN "High in Demand"
     WHEN Quantity > 15 THEN "Normal Sale"
     ELSE "Low in Demand"
END ) AS `Products Demand`
FROM OrderDetails;
```

CASE FUNCTION…

Que: Write a query to categorise Products (as ‘Price Category’) with

- Price >=20 as ‘Expensive’,
- Price between 10-20 as ‘Reasonable’
- Remaining as ‘Discounted’

then sort the result set based on higher product price first and in case the price is not given use CategoryID for sorting.

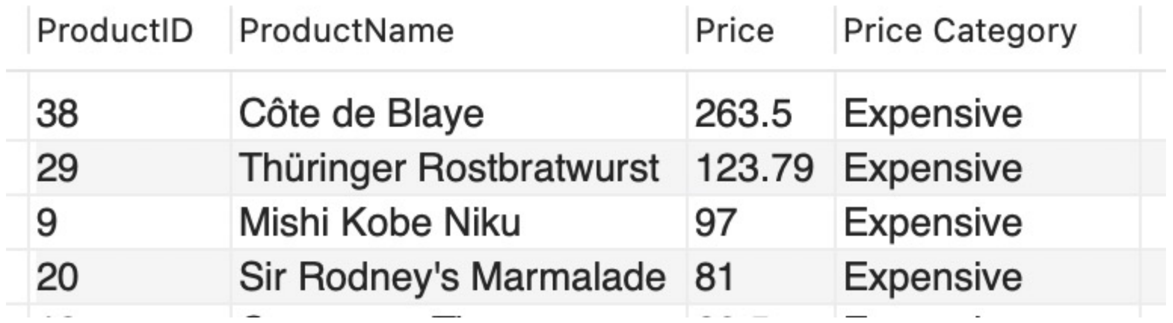
CASE FUNCTION…

Que: Write a query to categorise Products (as ‘Price Category’) with

- Price >=20 as ‘Expensive’,
- Price between 10-20 as ‘Reasonable’
- Remaining as ‘Discounted’

then sort the result set based on higher product price first and in case the price is not given use CategoryID for sorting.

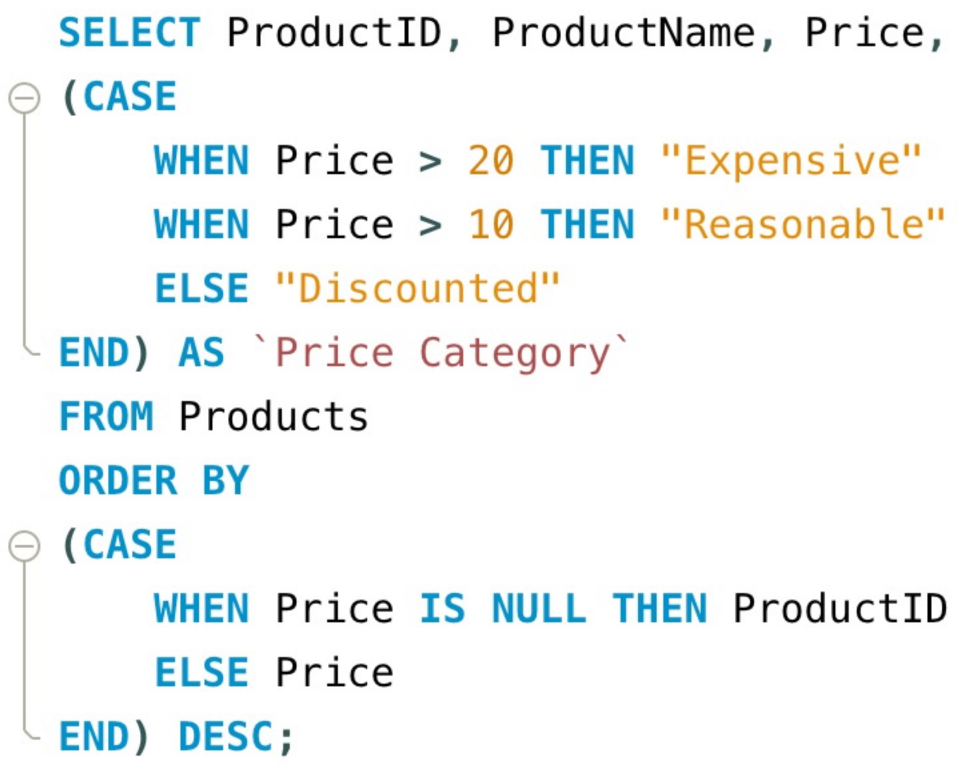
Expected Output:



ProductID	ProductName	Price	Price Category
38	Côte de Blaye	263.5	Expensive
29	Thüringer Rostbratwurst	123.79	Expensive
9	Mishi Kobe Niku	97	Expensive
20	Sir Rodney's Marmalade	81	Expensive

CASE FUNCTION…

- **Solution:**



```sql
SELECT ProductID, ProductName, Price,
(CASE
    WHEN Price > 20 THEN "Expensive"
    WHEN Price > 10 THEN "Reasonable"
    ELSE "Discounted"
END) AS `Price Category`
FROM Products
ORDER BY
(CASE
    WHEN Price IS NULL THEN ProductID
    ELSE Price
END) DESC;
```

THANK YOU