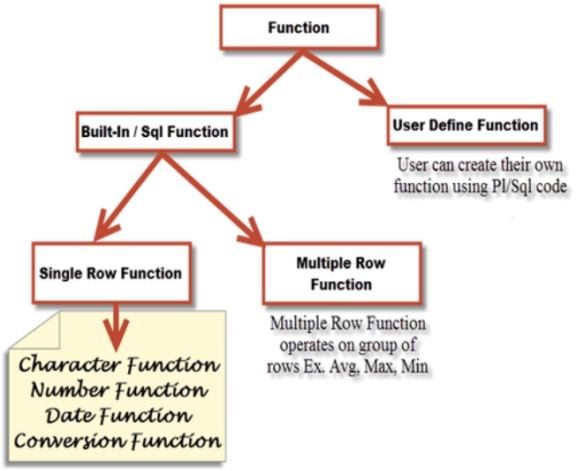
FUNCTION

A function is a block of code that sometimes take arguments and always returns a value.
SQL functions can be used to do the following:

1. Perform calculations on data.
2. Modify individual data items.
3. Manipulate output for groups of rows.
4. Format dates and numbers for display.
5. Convert column data types.



SINGLE ROW FUNCTION

- Manipulates data items.
- Accepts arguments and returns one value.
- Acts on each row returned.
- Returns one result per row.
- May modify the data type.
- Can be nested.
- Accepts arguments which can be a column or an expression.
- function_name [(arg1, arg2,...)]

CHARACTER FUNCTION

Single-row character functions accept character data as input and can return both character and numeric values.

Function	Purpose
Lower(column|exp)	Converts alpha character values to lower case.
Upper(column|exp)	Converts alpha character values to upper case.
Initcap(column|exp)	Converts alpha character values to upper case for the first letter of each word.
Concat(col1|exp1,col2|exp2)	Concatenates two columns or expression values.
Substr(col|exp,m,[n])	Returns specified characters from the character value starting at character position m, n characters long
Length(col|exp)	Returns the number of characters in the expression.

CHARACTER FUNCTION - EXAMPLE

Function	Example	Output
Lower(column|exp)	SELECT LOWER(Cname) FROM Customer WHERE cid=1;	Tom
Upper(column|exp)	SELECT UPPER(Cname) FROM Customer WHERE cid=1;	TOM
Initcap(column|exp)	SELECT INITCAP(Cname) FROM Customer WHERE cid=1;	Tom
Concat(col1|exp1,col2|exp2)	SELECT CONCAT (Cname, Address) FROM Customer WHERE cid=1;	Tomchennai
Substr(col|exp,m,[n])	SELECT SUBSTR(Cname,1,2) FROM Customer WHERE cid=1; SELECT SUBSTR(Cname,1) FROM Customer WHERE cid=1; SELECT SUBSTR(Cname,-2,2) FROM Customer WHERE cid=1;	To Tom om
Length(col|exp)	SELECT LENGTH(Cname) FROM Customer WHERE cid=1;	3

NUMBER FUNCTION

Number functions accept numeric input and return numeric values.

Function	Purpose
Abs(col|exp)	returns the absolute value of n.
Ceil(col|exp)	returns the smallest integer value that is greater than or equal to a *number*.
Floor(col|exp)	returns the largest integer value that is equal to or less than a *number*.
Mod(m,n)	returns the remainder of *m* divided by *n*.
Round(n[,m])	returns a number rounded to a certain number of decimal places. m refers to decimal places
Trunc(n[,m]	returns a number truncated to a certain number of decimal places.

NUMBER FUNCTION – EXAMPLE

Function	Example	Output
Abs(col|exp)	SELECT ABS(-5) FROM DUAL;	5
Ceil(col|exp)	SELECT CEIL(5.3) FROM DUAL;	6
Floor(col|exp)	SELECT FLOOR(5.8) FROM DUAL;	5
Mod(m,n)	SELECT MOD(10,2) FROM DUAL;	0
Round(n[,m])	SELECT ROUND(10.678,1) FROM DUAL; SELECT ROUND(10.678) FROM DUAL;	10.7 11
Trunc(n[,m]	SELECT TRUNC(10.678,1) FROM DUAL; SELECT TRUNC(10.67) FROM DUAL;	10.6 10

Dual – dummy table with a varchar data type column

DATE FUNCTION

To get current system date

SELECT SYSDATE from Dual;

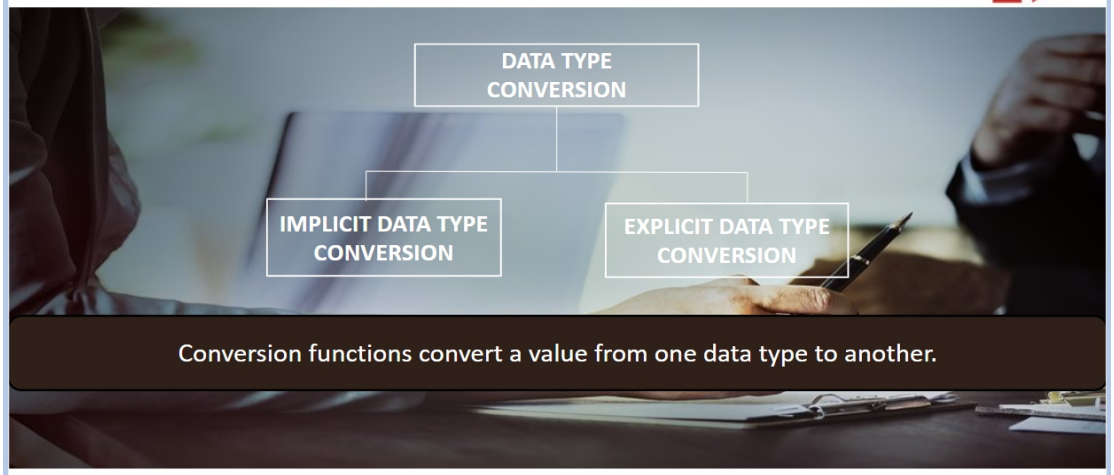
Arithmetic operation

- Add days to date.
- Subtract days from date.
- Subtract two dates.

DATE FUNCTION

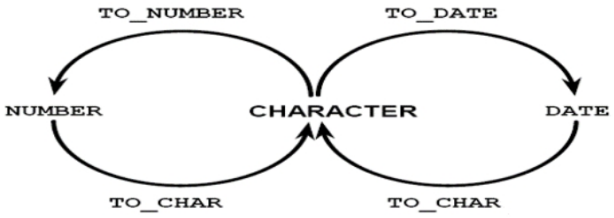
Function	Purpose	Example	Output
Months_between	Number of months between two dates	SELECT cid, cname, ROUND(months_between (sysdate, dob)/12) age FROM customer WHERE cid=1;	25
Add_months	Add calendar months to date	SELECT ADD_MONTHS(SYSDATE,4) FROM DUAL; -- current date : 18-APR-19	18-AUG-19
Next_date	Next date of the day specified	SELECT NEXT_DAY(SYSDATE,'monday') FROM DUAL;	23-APR-19
Last_day	Last day of the month	SELECT LAST_DAY(SYSDATE) FROM DUAL;	30-APR-19
Round	Round date	SELECT ROUND(SYSDATE,'year') FROM DUAL; SELECT ROUND(SYSDATE,'month') FROM DUAL;	01-JAN-19 01-MAY-19
Trunc	Truncate date	SELECT TRUNC(SYSDATE,'year') FROM DUAL; SELECT TRUNC(SYSDATE,'month') FROM DUAL;	01-JAN-19 01-APR-19

CONVERSION FUNCTION



CONVERSION FUNCTION

Function	Purpose
To_char(number|date,[fmt])	Converts a number or date to string
To_date(char,[fmt])	Converts a string to a date
To_number(char,[fmt])	Converts a string to a number



ELEMENTS OF THE DATE FORMAT MODEL

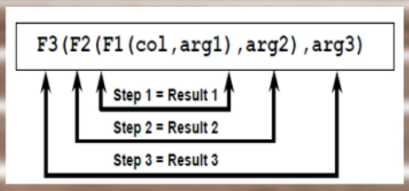
Parameter	Explanation
Year YYYY YY	Year, spelled out 4-digit year 2-digit year
MM MONTH MON	Month number(1-12) Abbreviated name of month Full name of month
WW W	Week of year Week of month
D DAY DD DDD	Day of week Name of day Day of month Day of year
HH HH24	Hour of day(0-12) Hour of day(0-23)
MI	Minute
SS	Second

CONVERSION FUNCTION - EXAMPLE

FUNCTION	EXAMPLE	OUTPUT
TO_CHAR(number|date,[fmt])	SELECT TO_CHAR(SYSDATE,'dd month yyyy') FROM DUAL;	18 APRIL 2019
TO_DATE(char,[fmt])	SELECT TO_DATE('may-12-2019','mon-dd-yyyy') FROM DUAL;	12-MAY-19
TO_NUMBER(char,[fmt])	SELECT TO_NUMBER('234.67',999.99) FROM DUAL;	234.67

NESTING FUNCTIONS

- Single-row functions can be nested to any level.
- Nested functions are evaluated from deepest level to the least deep level.




```
select cid, cname, round(months_between (sysdate, dob)/12) age from customer where cid=1;
```

CONDITIONAL EXPRESSION

Gives the use of IF-THEN-ELSE logic within a SQL statement.

Two methods

- CASE expression.
- DECODE function.

CASE EXPRESSION

Facilitates conditional inquiries by doing the work of an IF-THEN-ELSE statement.

```
SELECT pid,pname,
CASE pname
WHEN 'Money Back Plan' THEN 'Money Savings'
WHEN 'Personal Protect' THEN 'Life Insurance'
ELSE 'Normal Policy'
END "Category of Policies" FROM policy;
```

PID	PNAME	CATEGORY OF POLICIES
MBP	Money Back Plan	Money Savings
PP	Personal Protect	Life Insurance

DECODE FUNCTION

DECODE is a function in Oracle and is used to provide if-then-else type of logic to SQL. It is not available in MySQL or SQL Server.

```
SELECT pid,pname,DECODE(pname, 'Money Back Plan', 'Money Saving', 'Personal Protect', 'Life Insurance ','Normal Policy') "Category of Policies" FROM Policy;
```

PID	PNAME	CATEGORY OF POLICIES
MBP	Money Back Plan	Money Savings
PP	Personal Protect	Life Insurance

GENERAL FUNCTION

These functions work with any data type and pertain to using null value.

FUNCTION	PURPOSE
NVL(exp1,exp2)	Converts a null value to an actual value
NVL2(exp1,exp2,exp3)	If exp1 is not null then return exp2. If exp2 is null then return exp3
NULLIF	Compares two expressions and returns null if they are equal or the first expression if they are not equal
COALESCE	Returns the first not null expression in the expression list

NVL FUNCTION

- Converts a null to an actual value.
- Data types that can be used are date, character, and number.
- Data types must match.

```
SELECT cid, pid, duedate,paiddate,amount,NVL(penalty,0) penalty FROM policyenrollment;
```

CID	PID	DUEDATE	PAIDDATE	AMOUNT	PENALTY
3	MBP	12-Dec-2017	11-Dec-2017	2000	0
1	PP	15-Mar-2018	13-Mar-2018	3000	0
2	PP	15-Feb-2018	22-Feb-2018	4000	200

NVL2 FUNCTION

SELECT cid, pid, duedate,NVL2(Paiddate,'Paid','Not Paid') status FROM policyenrollment;

CID	PID	DUEDATE	STATUS
3	MBP	12-Dec-2017	Paid
1	PP	15-Mar-2018	Paid
2	PP	15-Feb-2018	Not Paid

COALESCE FUNCTION

SELECT cid,cname,COALESCE(emailid,TO_CHAR(phoneno),address) contact FROM customer;

CID	CNAME	CONTACT
1	Tom	9876523190
2	John	john@yahoo.com
3	Ram	ram@gmail.com
4	Tiny	chennai

AGGREGATE OR GROUP FUNCTION

Group functions operate on the entire set of rows on the table and give one result for the entire set.

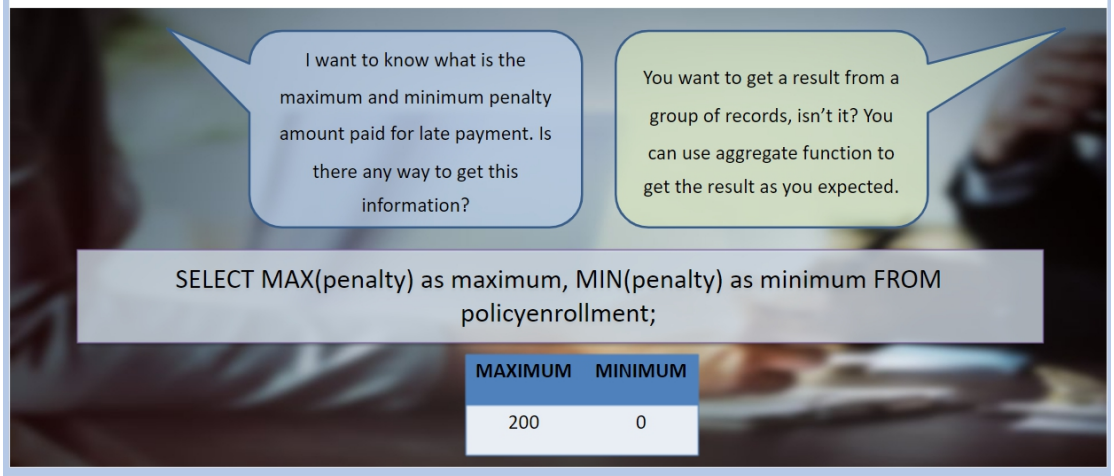
Group functions can be nested to a depth of two.

Various group functions

- MAX
- MIN
- SUM
- AVG
- COUNT

Group Functions ignore the null values.

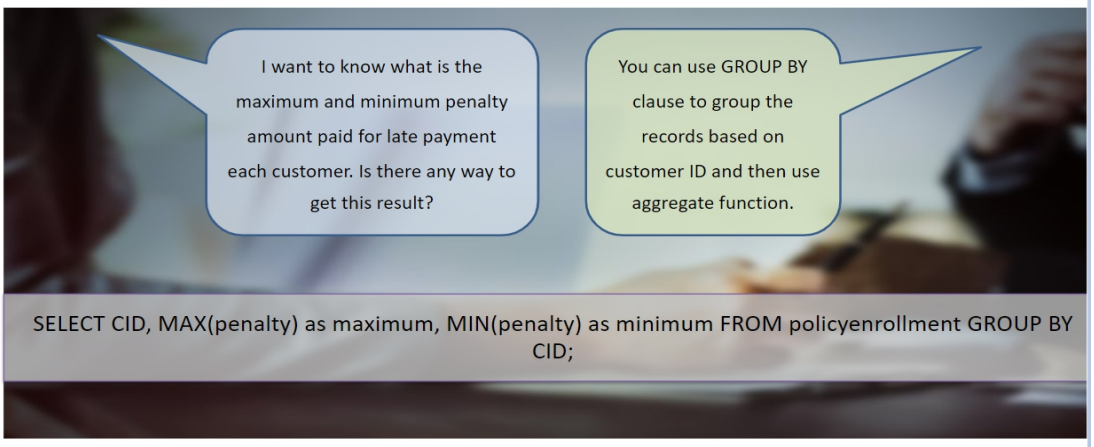
AGGREGATE FUNCTION - EXAMPLE



GROUP BY CLAUSE

- The GROUP BY clause divides the rows in a table into groups.
- Divide row of a table into smaller groups by using the GROUP BY clause and with this GROUP BY a column function results in a single value for each group.
- If a group function is included in a SELECT clause along with other non grouped columns, then these non grouped column's should appear in the GROUP BY clause.
- SQL Compiler generates an error if the non grouped columns are not included in the GROUP BY clause.
- The GROUP BY column does not have to be in the SELECT list.
- A column alias cannot be used in the GROUP BY clause.

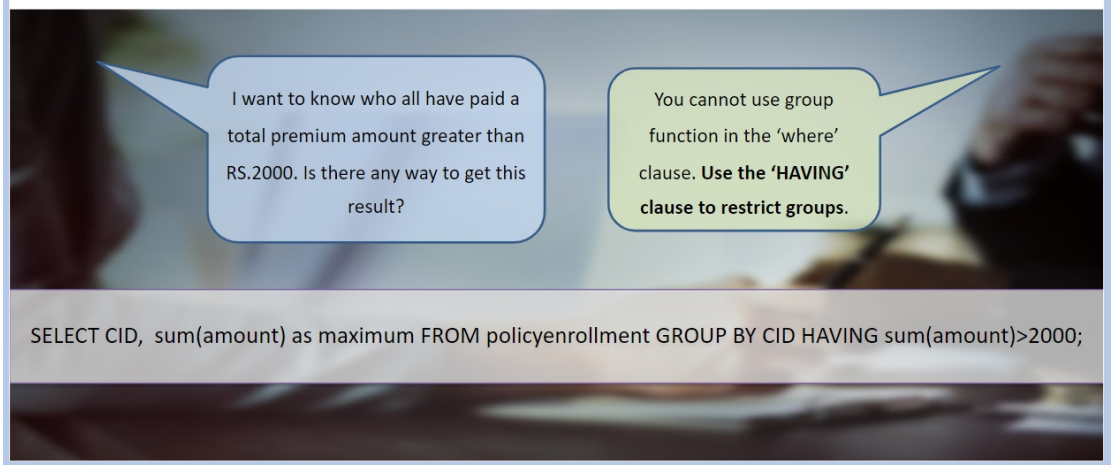
GROUP BY - EXAMPLE



HAVING CLAUSE

- The WHERE clause cannot be used to restrict groups. The group functions cannot be in the WHERE clause.
- Conditions for groups must be specified in the HAVING clause. HAVING clause of select groups satisfy certain conditions.
- HAVING can specify any column function on any column in a table being queried. This column needs not be in the SELECT list.

Having clause




```
SELECT CID,  sum(amount) as maximum FROM policyenrollment GROUP BY CID HAVING sum(amount)>2000;
```