第６章　関数、述語、CASE式

　　６－１　いろいろな関数

関数・・・ある値を入力するとそれに対応した値を出力すること。

|  |  |  |
| --- | --- | --- |
| m | n | p |
| 8.760 |  |  |
| 500.000 | 0 |  |
| -180.000 | 0 |  |
|  |  |  |
|  | 7 | 3 |
|  | 5 | 2 |
|  | 4 |  |
| 8.000 |  | 3 |
| 2.270 | 1 |  |
| 5.555 | 2 |  |
|  | 1 |  |
| 8.760 |  |  |

ちなみに入力を引数、出力を戻り値と呼ぶ。

関数の種類

　　　・算術関数（数値の計算を行うための関数）

　　　・文字列関数（文字列を操作するための関数）

　　　・日付関数（日付を操作するための関数）

　　　・変換関数（データ型や値を変換するための関数）

　　　・集約関数（データの集計を行うための関数）

関数では、COUNT,SUM、AVG、MAX、MINがだいたい主流ですが

ほかにも２００種類以上ある。

今回は右の表を使って勉強していきます。

ABS・・・絶対値を求める関数

絶対値・・・数値の符号を考えない大きさを表す数値。

|  |  |
| --- | --- |
| m | ABS |
| 8.760 | 8.760 |
| 500.000 | 500.000 |
| -180.000 | 180.000 |
|  |  |

例

SELECT m,

ABS(m) AS abs\_col

FROM samplemath;

右のものが絶対値マイナスのものがプラスになってるのが大切。

|  |  |  |
| --- | --- | --- |
| n | p | mod |
| 0 |  |  |
| 7 | 3 | 1 |
| 5 | 2 | 1 |
| 4 |  |  |
|  | 3 |  |
| 1 |  |  |

MOD・・・割り算の余りを求める関数

例

SELECT n,p,MOD(n,p) AS mod\_col

FROM sampleMath;

このように数字があるところは表示され、n,pともに数字がある

ところはn÷pをしてそのあまりの数字がMODのところに出ている

|  |  |  |
| --- | --- | --- |
| m | n | round |
| 8.760 |  |  |
| 500.000 | 0 | 500.000 |
| 8.000 |  |  |
| 2.270 | 1 | 2.300 |
| 5.555 | 2 | 5.560 |
| 8.760 |  |  |

ROUND・・・四捨五入、

例

SELECT m,n,ROUND(n,p) AS round\_col

FROM sampleMath;

M、N両方あるところのmが四捨五入された

|  |  |  |
| --- | --- | --- |
| Str1 | Str2 | Str3 |
| あいう | えお |  |
| abc | def |  |
| 山田 | 太郎 | です |
| aaa |  |  |
|  | あああ |  |
| @!#$% |  |  |
| abc太郎 | abc | ABC |
| abcdefabc | abc | ABC |
| ミックマック | ッ | っ |

文字列関数・・・文字列を使う関数

文字列はstrともよばれ、日本語や英語などの文字の列のこと

今回は右の表を使って学んでいきます。

　　・CONCAT文・・・合体

　　・LENGTH文・・・文字数数え

　　・LOWER文・・・大文字を小文字に変更

　　・REPLACE・・・文字列の中の置き換え

　　・SUBDTRING・・・文字列の切り出し

　　・UPPER・・・小文字を大文字に変更

CONCAT文・・・Aの文とBの文を合体させる。

例

SELECT str1,str2,CONCAT(str1,str2) AS str\_concat

|  |  |  |
| --- | --- | --- |
| Str1 | Str2 | Str3 |
| あいう | えお | あいうえお |
| aaa |  |  |
|  | あああ |  |
| @!#$% |  |  |
| abc太郎 | abc | abc太郎abc |

FROM samplestr;

右のようにstr1とstr２のものが合体されて表示された

※CONCATの中は複数でも可能。

　例　CONCAT(str1,str2,str3)

　　　これで三つの文字列が合体できる。

LENGTH・・・文字列が何文字何文字かを数える関数

例

|  |  |
| --- | --- |
| str | length |
| あいう | 3 |
| abc | 3 |
| 山田 | 2 |
| aaa | 3 |
| @!#$% | 5 |
| ミックマック | 6 |

SELECT \* FROM SampleStr;

SELECT str1,CHAR\_LENGTH(str1)

FROM samplestr;

レングスをつけることでstrのものが何文字あるかを表示できる。

　　※MY SQLだけはLENGTHだけではなくCHAR＿LENGTHと

　　　　しないと正しい文字数は表示されない。

LOWER・・小文字化する関数（大文字アルファベットのみ反応）

例

|  |  |
| --- | --- |
| ｓｔｒ１ | LOWER |
| @!#$% | @!#$% |
| ABC | abc |
| aBC | abc |
| abc太郎 | abc太郎 |

SELECT str1,LOWER(str1)

FROM sampleStr;

大文字のアルファベットのみ小文字のアルファベットに変更される。

日本語や小文字はそのままの表示。

※逆に小文字を大文字にする場合はUPPERを使う。

REPLACE・・・文字列中にある１部分の文字列を別の文字列に変える関数

例

|  |  |  |  |
| --- | --- | --- | --- |
| ｓｔｒ１ | ｓｔｒ２ | ｓｔｒ３ | REPLACE |
| あいう | えお |  |  |
| 山田 | 太郎 | です | 山田 |
| aaa |  |  |  |
|  | あああ |  |  |
| abc太郎 | abc | ABC | ABC太郎 |
| abcdefabc | abc | ABC | ABCdefABC |
| ミックマック | ッ | っ | ミっクマっク |

SELECT str1,str2,str3,REPLACE(str1,str2,str3)

FROM sampleStr;

これでREPLASE(str1,str2,str3)のところが

・ｓｔｒ1の文字列の中にあるｓｔｒ２の文字列が

　ｓｔｒ３に置き換えるという意味になる。

例えば。ミックマックの場合

ミックマックの中にあるッをっに置き換えるという意味　でミックマックがマっクマっクに変わった。

SUBSTRING・・・文字列の中の１部分の文字列を切り出すときに使う関数

例

|  |  |
| --- | --- |
| ｓｔｒ１ | SUB |
| あいう | う |
| abc | c |
| 山田 |  |
| aaa | a |
|  |  |
| @!#$% | #$ |
| ABC | C |
| aBC | C |
| abc太郎 | c太 |
| abcdefabc | cd |
| ミックマック | クマ |

SELECT str1,SUBSTRING(str1 FROM 3 FOR 2)

FROM sampleStr;

今回の場合はstr１のなかの３文字目から２文字を表示します

という意味になる。

例えばミックマックの場合、

ミックマックの３文字目のクから２文字を表示するので、クマ

が表示される。

※PSTGRE SQLとMySQLでのみ使える。

日付関数・・・日付関連のものを表す関数

|  |
| --- |
| CURRENT\_DATE |
| '2023-09-22' |

CURRENT\_DATE・・・現在の日付を表す関数

例　SELECT CURRENT\_DATE;

　　今日の日付が表示された。

　　　　CURRENT\_TIME・・・現在の時間を表す関数

|  |
| --- |
| CURRENT\_TIME |
| 11:12:14 |

例　SELECT CURRENT\_TIME;

　　 現在の時刻が表示された。

CURRENT\_TIMESTAMP・・・現在の日時を表示する関数

|  |
| --- |
| CURRENT\_TIMESTAMP |
| 2023-09-22 11:14:13 |

例SELECT CURRENT\_TIMESTAMP;

現在の日時が表示された。

EXTRACT・・・日付要素を切り出す関数

例

SELECT CURRENT\_TIMESTAMP,

EXTRACT(YEAR FROM CURRENT\_TIMESTAMP) AS YEAR, という細かい切り出しができる

EXTRACT(MONTH FROM CURRENT\_TIMESTAMP) AS MONTH,←CURRENT\_STAMPからMONTHを切

EXTRACT(DAY FROM CURRENT\_TIMESTAMP) AS DAY, りだすよーという意味

EXTRACT(MINUTE FROM CURRENT\_TIMESTAMP) AS MINUTE,

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CURRENT\_TIMESTAMP | YEAR | MONTH | DAY | MINUTE | SECOND |
| 2023-09-22 11:20:07 | 2023 | 9 | 22 | 20 | 7 |

　　　変換関数・・・様々な特殊な変換をする関数。

|  |
| --- |
| CAST |
| 1 |

CAST…型の変換をする関数

例

SELECT CAST('0001'AS SIGNED INTEGER);

0001という文字を数字の１に変換

COALESCE・・・NULLを値に変換する関数

|  |  |  |
| --- | --- | --- |
| 上の行 | 真ん中の行 | 下の行 |
| 1 | test | 2009-11-01 |

例

SELECT COALESCE(NULL,1),

　　　 COALESCE(NULL,'test',NULL),

COALESCE(NULL,NULL,'2009-11-01');

|  |
| --- |
| Str2 |
| えお |
| Def |
| 太郎 |
| NULLです |
| あああ |
| NULLです |
| NULLです |
| NULLです |
| abc |

NULLのところに字が追加されるが見にくいのでもう一つ例題

SELECT COALESCE(str2,'NULLです')

FROM sampleStr;

このようにstr２の中にあるNULL(空白)のところにすべて

NULLです　という文字列が配置された。

※COALESCEの中にほかの関数を入れることができる。

例COALESCE（SUBSTRING(A FROM 1 FOR1),A,’NULL文字’）

述語・・・抽出条件を記述するときにつかうもの。（TRUE,FALSE.UNKNOWNのようなもの）

|  |
| --- |
| strcol |
| abcdd |
| abcddd |
| abddc |
| abdddc |
| ddabc |
| dddabc |

今回は右の図をつかってやります。

今回やる語句

・LIKE・・・いろいろな検索方法

・BETWEEN・・・○○以上○○以下などの範囲指定

・IS NULL,IS NOT NULL・・・ＮＵＬＬのところの表示、非表示

・IN・・・○○or○○or○○などをまとめたもの

・EXISTS・・・条件に合致するレコードの有無

前方一致・・・検索条件となる文字列の最初に位置しているレコードだけが選択される検索

中間一致・・・検索条件の文字列がどこかに含まれていればレコードが検索される仕方。最初でも

　　最後でも問題ない。

後方一致・・・検索条件となる文字列が文字列の最後にあるレコードだけを表示する検索

|  |
| --- |
| strcol |
| dddabc |

前方一致の検索

SELECT \*

FROM sampleLike

Where strcol LIKE 'ddd%';

最初にdddがあるものだけが表示された。

|  |
| --- |
| abddc |
| abdddc |
| ddabc |

中間一致の検索

　 SELECT \*

FROM sampleLike

Where strcol LIKE '%ddd%';

　　　　後方一致の検索

|  |
| --- |
| strcol |
| abcddd |

COMMIT;

SELECT \*

FROM sampleLike

　　　 文字列の後ろに文字列があるものの検索

|  |
| --- |
| strcol |
| abcdd |

SELECT \*

FROM sampleLike

Where strcol LIKE 'abc\_\_';←＿　はふたつある

これでabc○○というものが検索された

　　　　範囲検索の方法

BETWEEN・・・○○から○○という検索方法ができる。

例

|  |  |
| --- | --- |
| 穴あけパンチ | 500 |
| フォーク | 500 |
| おろしがね | 880 |
| ボールペン | 100 |

SELECT shohin\_mei,hanbai\_tanka

FROM shohin

WHERE hanbai\_tanka BETWEEN 100 AND 1000;

これで１００円以上１０００以下のものが表示された。

　　　　IS NULL・・・NULLのところの表示する検索方法

例

|  |  |
| --- | --- |
| フォーク |  |
| ボールペン |  |

SELECT shohin\_mei,shiire\_tanka

FROM shohin

WHERE shiire\_tanka IS NULL;

これで仕入単価がNULLのものが表示される。

　　　　IS NOT NULL・・・逆にNULL以外の行を表示する検索方法

例

SELECT shohin\_mei,shiire\_tanka

FROM shohin

WHERE shiire\_tanka IS NOT NULL;

これで仕入単価がNULL以外のものが表示される。

OR・・・そのもの自体を指定する検索方法

例

|  |  |
| --- | --- |
| Tシャツ | 500 |
| 穴あけパンチ | 320 |
| 圧力鍋 | 5000 |

SELECT shohin\_mei,shiire\_tanka

FROM shohin

WHERE shiire\_tanka = 320

OR shiire\_tanka = 500

OR shiire\_tanka = 5000;

これで仕入単価が320.500.5000のものが表示されたが毎回打つのが面倒なので出たのが、

　　　　IN述語・・・ORの簡略化したもの

例

|  |  |
| --- | --- |
| Tシャツ | 500 |
| 穴あけパンチ | 320 |
| 圧力鍋 | 5000 |

SELECT shohin\_mei,shiire\_tanka

FROM shohin

WHERE shiire\_tanka IN(320,500,5000);

このように簡略化することができる。

NOT IN ・・・先ほどの逆で指定したもの以外を表示する方法

|  |  |
| --- | --- |
| カッターシャツ | 2800 |
| 包丁 | 2800 |
| おろしがね | 790 |

例

SELECT shohin\_mei,shiire\_tanka

FROM shohin

WHERE shiire\_tanka NOT IN(320,500,5000);

こうすれば、320.500.5000以外のものが表示される。

|  |  |  |  |
| --- | --- | --- | --- |
| 000A | 東京 | 0001 | 30 |
| 000A | 東京 | 0002 | 50 |
| 000A | 東京 | 0003 | 15 |
| 000B | 名古屋 | 0002 | 30 |
| 000B | 名古屋 | 0003 | 120 |
| 000B | 名古屋 | 0004 | 20 |
| 000B | 名古屋 | 0006 | 10 |
| 000B | 名古屋 | 0007 | 40 |
| 000C | 大阪 | 0003 | 20 |
| 000C | 大阪 | 0004 | 50 |
| 000C | 大阪 | 0006 | 90 |
| 000C | 大阪 | 0007 | 70 |
| 000D | 福岡 | 0001 | 100 |
|  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 商品ID | 商品名 | 商品分類 | 販売単価 | 単価 | 登録日 |
| 0001 | Tシャツ | 衣類 | 1000 | 500 | 2009-09-20 |
| 0002 | 穴あけパンチ | 事務用品 | 320 | 320 | 2009-09-11 |
| 0003 | カッターシャツ | 衣類 | 4000 | 2800 |  |
| 0004 | 包丁 | キッチン用品 | 2800 | 2000 | 2009-09-20 |
| 0005 | 圧力鍋 | キッチン用品 | 6800 | 5000 | 2009-01-15 |

IN述語の引数にサブクエリを指定する

上の表と右の表を使います。

この二つの表を重ね合わせてみていきます。

まず

|  |
| --- |
| 0003 |
| 0004 |
| 0006 |
| 0007 |

SELECT shohin\_id これでまず店舗ＩＤが000Ｃのものの商品

FROM tenposhohin ＩＤが表示されます。

WHERE tenpo\_id ='000C';

続いてサブクエリに上のものを入れることでその店舗商品のものが表示できます。

|  |  |
| --- | --- |
| カッターシャツ | 3000 |
| 包丁 | 3000 |
| フォーク | 500 |
| おろしがね | 880 |

SELECT shohin\_mei,hanbai\_tanka

FROM shohin

WHERE shohin\_id IN(SELECT shohin\_id←両方共有されているもの

　　　FROM tenposhohin

　　　WHERE tenpo\_id = '000C');

サブクエリでＩＤを表示させることで店舗ＩＤのものの名前と単価を表示することができる。

　　※↑のＩＮのところをＮＯＴ　ＩＮ　にすることで逆に店舗ＩＤが000Ｃ以外のものを表示することができます。

ＥＸＩＳＴＳ・・・ある条件に合致するレコードがあるかないかの検索

|  |  |
| --- | --- |
| カッターシャツ | 3000 |
| 包丁 | 3000 |
| フォーク | 500 |
| おろしがね | 880 |

例

SELECT shohin\_mei,hanbai\_tanka　・・商品名と単価の表示

FROM shohin AS S　…商品からでニックネームはＳ

WHERE EXISTS(SELECT \*　　・・・イグジスツ

　　 FROM tenposhohin AS TS　・・・店舗商品　ＡＳと呼ぶ

　　　 WHERE TS.tenpo\_id ='000C'　・・・店舗商品のてんぽＩＤ＝000Ｃのものを条件

　　　　 AND TS.shohin\_id =S.shohin\_id);・・・ＴＳＩＤとＳＩＤを比べるよ

※相関サブクエリ部分のセレクト句の＊のところは何でも問題はないがマナーとして＊にしてる。

６－３　CASE式

CASE式とは・・・1+1などの一種の演算を行う機能。

　　　　条件分岐をする。

　　　　使い方

CASE　WHEN　＜評価式＞　THEN ＜式＞

　　　　 WHEN　＜評価式＞　THEN ＜式＞

　　　　　　　　　　WHEN　＜評価式＞　THEN ＜式＞

ELSE　＜式＞

END

という使い方をする

|  |  |
| --- | --- |
| Tシャツ | A:衣服 |
| 穴あけパンチ | C:事務用品 |
| カッターシャツ | A:衣服 |
| 包丁 | B:キッチン用品 |
| 圧力鍋 | B:キッチン用品 |
| フォーク | B:キッチン用品 |
| おろしがね | B:キッチン用品 |
| ボールペン | C:事務用品 |

例

SELECT shohin\_mei,

CASE WHEN shohin\_bunrui = '衣服'

　　　 THEN 'A:' || shohin\_bunrui

　　　　　 　　　WHEN shohin\_bunrui = '事務用品'

　　　　THEN 'B:' || shohin\_bunrui

WHEN shohin\_bunrui = 'キッチン用品'

　　　　　THEN 'C:' || shohin\_bunrui

　　　　 ELSE NULL

END AS abc\_shohin\_bunrui

FROM shohin;

これで各商品分類にA,B,Cをつけることができる。

|  |  |
| --- | --- |
| 衣服 | 5000 |
| 事務用品 | 600 |
| キッチン用品 | 11180 |

つぎはこれと行として並べるのではなく、列で並べてみましょう

例

SELECT shohin\_bunrui,　SUM(hanbai\_tanka) AS sum\_tanka

FROM shohin

GROUP BY shohin\_bunrui;

これの結果はこうなるがこれを列で並べるようにするには。

SELECT SUM(CASE WHEN shohin\_bunrui = '衣服'

THEN hanbai\_tanka else 0 END)AS sum\_tanka\_ihuku,

　　　　　　SUM(CASE WHEN shohin\_bunrui = 'キッチン用品'

THEN hanbai\_tanka else 0 END)AS sum\_tanka\_kitchin,

　　　　　　SUM(CASE WHEN shohin\_bunrui = '事務用品'

THEN hanbai\_tanka else 0 END)AS sum\_tanka\_jimu

FROM shohin;

これで一つ一つは列としてあらわすことができる。

|  |  |
| --- | --- |
| Tシャツ | A:衣服 |
| 穴あけパンチ | C:事務用品 |
| カッターシャツ | A:衣服 |
| 包丁 | B:キッチン用品 |
| 圧力鍋 | B:キッチン用品 |
| フォーク | B:キッチン用品 |
| おろしがね | B:キッチン用品 |
| ボールペン | C:事務用品 |

単純CASE式・・・先ほどのCASE式の計算式を簡略化した式

　例

SELECT shohin\_mei,

CASE shohin\_bunrui

WHEN '衣服' THEN CONCAT('A:',shohin\_bunrui)

WHEN 'キッチン用品'THEN CONCAT('B:',shohin\_bunrui)

WHEN '事務用品'THEN CONCAT('C:',shohin\_bunrui)

ELSE null

END AS abc\_shohin\_bunrui

FROM shohin;

このように簡略化することもできる。