Postgresql

**Data type**

Char(5): max 5 characters

Varchar: any length characters

Varchar(20):20 characters

Text:any length characters

Smallserial:1 to 32767

Serial:1 to 2147483647

BigSerial: 1 to ….

Smallint: -32768 to 32768

Integer

Bigint

Decimal

Numeric

Real

Double precision

Float

Date

Timestamp

Interval:represent duration of time(1 D 1 H 1 S….)

Currency

Binary

JSON

Range

Array

XML

UUID

**CREATE TYPE**

CREATE TYPE sex\_type as enum (‘M’,’F’);

**UPDATE COLUMN TO TYPE**

Alter table customer alter column sex TYPE sex\_type using sex::sex\_type;

Alter table customer alter column zip TYPE integer

**RENAME COLUMN NAME**

Alter table sales\_item RENAME COLUMN day\_of\_week TO weekday; - rename column

**RENAME TABLE NAME**

Alter table tablename RENAME TO newTableName

**REVERSE ACTION**

Truncate table tablename

**DROP TABLE**

Drop table tablename

**EXTRACT METHOD**

Select \* from bill where EXTRACT(MONTH from update\_on) = 4

**CONCAT FUNCTION**

Select concat(first\_name,’ ’,last\_name) as name from customer

**IS NOT <>**

Select \* from customer where state <> ‘CA’

**Regular Expression**

% after specific word with any words,example ‘M%’

A screenshot of a computer

Description automatically generated with medium confidence

Select \* from customer where first\_name similar to ‘M%’

Select \* from customer where first\_name ~ ‘^M’

**CREATE FUNCTION**

1. return void

CREATE OR REPLACE FUNCTION fn\_functionName1(state\_name char(2))

RETURNS void as

$body$

Update sales\_person set state = ‘PA’ where state = state\_name;

$body$

LANGUAGE SQL

SELECT fn\_functionName1(“US”);

2. return numeric

CREATE OR REPLACE FUNCTION fn\_functionName2()

RETURNS numeric as

$body$

Select max(price) from item

$body$

LANGUAGE SQL

SELECT fn\_functionName2();

3. return table

CREATE OR REPLACE FUNCTION fn\_functionName3()

RETURNS SETOF sales\_person as

$body$

Select \* from sales\_person

$body$

LANGUAGE SQL

SELECT (fn\_functionName3()).\*; //re-format

**CREATE FUNCTION WITH plqgsql language**

1. example1

CREATE OR REPLACE FUNCTION fg\_get\_sum(val1 int,val2 int)

RETURNS int AS

$body$

DECLARE

Ans int;

BEGIN

Ans := val1+val2;

RETURN Ans;

END

$body$

LANGUAGE plpgsql

Select fn\_get\_sum(4,6)

2. using in & out

CREATE OR REPLACE FUNCTION fg\_get\_sum2(IN val1 int, IN val2 int, OUT ans int)

AS

$body$

Ans := v1+v2

$body$

LANGUAGE plpgsql

Select fn\_get\_sum2(4,6)

3. IN & OUT Example 2

CREATE OR REPLACE FUNCTION fg\_get\_name(IN state,OUT first\_name varchar,last\_name varchar)

AS

$body$

Select fist\_name,last\_name

Into first\_name, last\_name

From customer

Where state = state

$body$

LANGUAGE plpgsql

Select fg\_get\_name (“US”)

4. IF STATEMENT

CREATE OR REPLACE FUNCTION fn\_month\_orders (month int)

RETURNS varchar as

$body$

DECLARE

Total\_orders int;

BEGIN

Select count(order\_number)

Into Total\_orders

From sales\_order

If Total\_orders > 1 THEN

RETURN “Good”

ELSEIF total = 0 THEN

RETURN “NO ORDER”;

ELSE

RETURN “ERROR”

END IF;

END;

$body$

LANGUAGE SQL

SELECT fn\_month\_orders (1);

5. SWITCH STATEMENT

CREATE OR REPLACE FUNCTION fn\_month\_orders2 (month int)

RETURNS varchar as

$body$

DECLARE

Total\_orders int;

BEGIN

Select count(order\_number)

Into Total\_orders

From sales\_order

CASE

WHEN Total\_orders > 1 THEN

RETURN “Good”

WHEN total = 0 THEN

RETURN “NO ORDER”;

ELSE

RETURN “ERROR”

END CASE;

END;

$body$

LANGUAGE SQL

SELECT fn\_month\_orders2 (1);