PRACTICE:

1.UPPER()

madhu=# select product\_name,upper(product\_name) as Upper\_product\_name from OT.products LIMIT 10;

product\_name | upper\_product\_name

----------------------------------+----------------------------------

Intel Xeon E5-2699 V3 (OEM/Tray) | INTEL XEON E5-2699 V3 (OEM/TRAY)

Intel Xeon E5-2697 V3 | INTEL XEON E5-2697 V3

Intel Xeon E5-2698 V3 (OEM/Tray) | INTEL XEON E5-2698 V3 (OEM/TRAY)

Intel Xeon E5-2697 V4 | INTEL XEON E5-2697 V4

Intel Xeon E5-2685 V3 (OEM/Tray) | INTEL XEON E5-2685 V3 (OEM/TRAY)

Intel Xeon E5-2695 V3 (OEM/Tray) | INTEL XEON E5-2695 V3 (OEM/TRAY)

Intel Xeon E5-2697 V2 | INTEL XEON E5-2697 V2

Intel Xeon E5-2695 V4 | INTEL XEON E5-2695 V4

Intel Xeon E5-2695 V2 | INTEL XEON E5-2695 V2

Intel Xeon E5-2643 V2 (OEM/Tray) | INTEL XEON E5-2643 V2 (OEM/TRAY)

(10 rows)

2.LOWER()

madhu=# select product\_name,lower(product\_name) as Upper\_prLower\_name from OT.products LIMIT 10;

product\_name | upper\_prlower\_name

----------------------------------+----------------------------------

Intel Xeon E5-2699 V3 (OEM/Tray) | intel xeon e5-2699 v3 (oem/tray)

Intel Xeon E5-2697 V3 | intel xeon e5-2697 v3

Intel Xeon E5-2698 V3 (OEM/Tray) | intel xeon e5-2698 v3 (oem/tray)

Intel Xeon E5-2697 V4 | intel xeon e5-2697 v4

Intel Xeon E5-2685 V3 (OEM/Tray) | intel xeon e5-2685 v3 (oem/tray)

Intel Xeon E5-2695 V3 (OEM/Tray) | intel xeon e5-2695 v3 (oem/tray)

Intel Xeon E5-2697 V2 | intel xeon e5-2697 v2

Intel Xeon E5-2695 V4 | intel xeon e5-2695 v4

Intel Xeon E5-2695 V2 | intel xeon e5-2695 v2

Intel Xeon E5-2643 V2 (OEM/Tray) | intel xeon e5-2643 v2 (oem/tray)

(10 rows)

3.Initcap

madhu=# select product\_name,initcap(product\_name) as init\_product\_name from OT.products LIMIT 10;

product\_name | init\_product\_name

----------------------------------+----------------------------------

Intel Xeon E5-2699 V3 (OEM/Tray) | Intel Xeon E5-2699 V3 (Oem/Tray)

Intel Xeon E5-2697 V3 | Intel Xeon E5-2697 V3

Intel Xeon E5-2698 V3 (OEM/Tray) | Intel Xeon E5-2698 V3 (Oem/Tray)

Intel Xeon E5-2697 V4 | Intel Xeon E5-2697 V4

Intel Xeon E5-2685 V3 (OEM/Tray) | Intel Xeon E5-2685 V3 (Oem/Tray)

Intel Xeon E5-2695 V3 (OEM/Tray) | Intel Xeon E5-2695 V3 (Oem/Tray)

Intel Xeon E5-2697 V2 | Intel Xeon E5-2697 V2

Intel Xeon E5-2695 V4 | Intel Xeon E5-2695 V4

Intel Xeon E5-2695 V2 | Intel Xeon E5-2695 V2

Intel Xeon E5-2643 V2 (OEM/Tray) | Intel Xeon E5-2643 V2 (Oem/Tray)

(10 rows)

4.Length()

madhu=# select product\_name,length(product\_name) as Length from OT.products LIMIT 10;

product\_name | length

----------------------------------+--------

Intel Xeon E5-2699 V3 (OEM/Tray) | 32

Intel Xeon E5-2697 V3 | 21

Intel Xeon E5-2698 V3 (OEM/Tray) | 32

Intel Xeon E5-2697 V4 | 21

Intel Xeon E5-2685 V3 (OEM/Tray) | 32

Intel Xeon E5-2695 V3 (OEM/Tray) | 32

Intel Xeon E5-2697 V2 | 21

Intel Xeon E5-2695 V4 | 21

Intel Xeon E5-2695 V2 | 21

Intel Xeon E5-2643 V2 (OEM/Tray) | 32

5.concatination

madhu=# select product\_id||' '||product\_name as product\_format from OT.products limit 10;

product\_format

--------------------------------------

228 Intel Xeon E5-2699 V3 (OEM/Tray)

248 Intel Xeon E5-2697 V3

249 Intel Xeon E5-2698 V3 (OEM/Tray)

2 Intel Xeon E5-2697 V4

45 Intel Xeon E5-2685 V3 (OEM/Tray)

46 Intel Xeon E5-2695 V3 (OEM/Tray)

47 Intel Xeon E5-2697 V2

51 Intel Xeon E5-2695 V4

91 Intel Xeon E5-2695 V2

92 Intel Xeon E5-2643 V2 (OEM/Tray)

(10 rows)

6.substr

madhu=# select substr(product\_name,1,5) as substring from OT.products limit 10;

substring

-----------

Intel

Intel

Intel

Intel

Intel

Intel

Intel

Intel

Intel

Intel

(10 rows)

-When negative index is given then the length-index number of characters are considered.

madhu=# select substr(product\_name,-1,10) from Ot.products limit 5;

substr

----------

Intel Xe

Intel Xe

Intel Xe

Intel Xe

Intel Xe

(5 rows)

7. position()

madhu=# madhu=# select product\_name,position('t' in product\_name) as separator from OT.products limit 10;

product\_name | separator

----------------------------------+-----------

Intel Xeon E5-2699 V3 (OEM/Tray) | 3

Intel Xeon E5-2697 V3 | 3

Intel Xeon E5-2698 V3 (OEM/Tray) | 3

Intel Xeon E5-2697 V4 | 3

Intel Xeon E5-2685 V3 (OEM/Tray) | 3

Intel Xeon E5-2695 V3 (OEM/Tray) | 3

Intel Xeon E5-2697 V2 | 3

Intel Xeon E5-2695 V4 | 3

Intel Xeon E5-2695 V2 | 3

Intel Xeon E5-2643 V2 (OEM/Tray) | 3

(10 rows)

8.floor(), ceil(), round()

madhu=# select ceil(standard\_cost) as ceil\_stnd\_cost,floor(standard\_cost) as floor\_stnd\_cost, round(standard\_cost) as round\_stnd\_cost from OT.products limit 10;

ceil\_stnd\_cost | floor\_stnd\_cost | round\_stnd\_cost

----------------+-----------------+-----------------

2868 | 2867 | 2868

2327 | 2326 | 2326

2036 | 2035 | 2035

2145 | 2144 | 2144

2013 | 2012 | 2012

1926 | 1925 | 1925

2102 | 2101 | 2102

1781 | 1780 | 1780

1794 | 1793 | 1794

1941 | 1940 | 1940

(10 rows)

madhu=# select least(1,2,3,-1);

least

-------

-1

(1 row)

madhu=# select greatest(1,2,3434);

greatest

----------

3434

(1 row)

madhu=# select mod(10,2);

mod

-----

0

(1 row)

madhu=# select mod(9,2);

mod

-----

1

(1 row)

madhu=# select sqrt(9);

sqrt

------

3

(1 row)

madhu=# select sqrt(5);

sqrt

------------------

2.23606797749979

(1 row)

madhu=# select pow(2,3);

pow

-----

8

(1 row)

madhu=# select round(9.23456);

round

-------

9

(1 row)

madhu=# select round(9.2345,2);

round

-------

9.23

(1 row)

JSON HANDLING using jsonb

Queries

1. Create a table for storing json data

create table emp\_details (id serial,first\_name text, last\_name text, data jsonb);

1. Inserting data

insert into emp\_details (first\_name,last\_name,data) values('madhu','koralla','{"designation":"intern","hobbies":["playing","reading"]}');

1. Accessing a particular value in a json string

select first\_name||' '||last\_name as emp\_name,data->'designation' as Designation from emp\_details;

1. Applying conditions to retrieve data

select \* from emp\_details where data->'designation'='"intern"' ;

1. Get the list as separate rows

select first\_name||' '||last\_name as emp\_name, jsonb\_array\_elements\_text(data->'hobbies') as Hobbbies from emp\_details;

1. Select distinct values from a json key of all rows.

select distinct(jsonb\_array\_elements\_text(data->'hobbies')) as Hobbies, first\_name||' '||last\_name as emp\_name from emp\_details;

1. Select the rows which have a particular value inside a list.

select first\_name as emp\_name, data from emp\_details where data->'hobbies' @>'["b"]'::jsonb;

practice

* Creating table to insert json data

create table emp\_details (id serial,first\_name text, last\_name text, data jsonb);

CREATE TABLE

1. Inserting values into table

insert into emp\_details (first\_name,last\_name,data) values('madhu','koralla','{"designation":"intern","hobbies":["playing","reading"]}');

INSERT 0 1

insert into emp\_details (first\_name,last\_name,data) values('sai','k','{"designation":"manger","hobbies":["playing","reading","b"]}');

INSERT 0 1

select \* from emp\_details;

id | first\_name | last\_name | data

----+------------+-----------+-------------------------------------------------------------------

1 | madhu | koralla | {"hobbies": ["playing", "reading"], "designation": "intern"}

2 | sai | k | {"hobbies": ["playing", "reading", "b"], "designation": "manger"}

(2 rows)

1. Accessing a particular value in a json string

select first\_name||' '||last\_name as emp\_name,data->'designation' as Designation from emp\_details;

emp\_name | designation

---------------+-------------

madhu koralla | "intern"

sai k | "manger"

(2 rows)

1. Applying conditions to retrieve data

select \* from emp\_details where data->'designation'='"intern"' ;

id | first\_name | last\_name | data

----+------------+-----------+--------------------------------------------------------------

1 | madhu | koralla | {"hobbies": ["playing", "reading"], "designation": "intern"}

(1 row)

1. Get the list as separate rows

select first\_name||' '||last\_name as emp\_name, jsonb\_array\_elements\_text(data->'hobbies') as Hobbbies from emp\_details;

emp\_name | hobbbies

---------------+----------

madhu koralla | playing

madhu koralla | reading

sai k | playing

sai k | reading

sai k | b

(5 rows)

1. Select distinct values from a json key of all rows.

select distinct(jsonb\_array\_elements\_text(data->'hobbies')) as Hobbies, first\_name||' '||last\_name as emp\_name from emp\_details;

hobbies | emp\_name

---------+---------------

playing | sai k

b | sai k

reading | sai k

playing | madhu koralla

reading | madhu koralla

(5 rows)

1. Select the rows which have a particular value inside a list

select first\_name as emp\_name, data from emp\_details where data->'hobbies' @>'["b"]'::jsonb;

emp\_name | data

----------+-------------------------------------------------------------------

sai | {"hobbies": ["playing", "reading", "b"], "designation": "manger"}

JSON HANDLING using json

1. Create a table to store json data

create table orders(id serial, info json);

1. Insert values.

insert into orders(info) values ('{"name":"madhu","items":{"product":"biscuts","qty":9}}');

insert into orders(info) values ('{"name":"sameer","items":{"product":"bucket","qty":12}}');

PostgreSQL provides two native operators -> and ->> to help you query JSON data.

* The operator -> returns JSON object field by key.
* The operator ->> returns JSON object field by text.

1. To get value in json as json object

select info->>'name' as customer\_name from orders where info->'items'->>'products'='biscuts';

1. To get value in json as text

select info->>'name' as customer\_name from orders where info->'items'->>'products'='biscuts';

1. Json with where clause

select info->>'name' as customer\_name from orders where info-> 'items' ->> 'product' = 'biscuts';

1. To apply condition on integer values

select info->>'name' as customer\_name, info-> 'items'->>'product' as product from orders where cast(info->'items'->>'qty' as integer)=9;

1. To get values in separate columns

select info->>'name' as customer\_name, info-> 'items'->>'product' as product from orders;

1. Aggregate functions

select max(cast(info->'items'->>'qty' as integer)) as max\_qty from orders;

select min(cast(info->'items'->>'qty' as integer)) as max\_qty from orders;

)

select count(cast(info->'items'->>'qty' as integer)) as max\_qty from orders;

select sum(cast(info->'items'->>'qty' as integer)) as max\_qty from orders;

select avg(cast(info->'items'->>'qty' as integer)) as max\_qty from orders;

insert into orders(info) values (‘{“name”:”madhu”,products”:{“prod\_id”:90,”order”:[“a”,”b”],”details”:{“a”:”b”,”c”:”d”}}}’)

1. Functions in json

select json\_each(info) as objects from orders;

10.To get key names

select json\_object\_keys(info) from orders;

to get distinct keys: select distinct(json\_object\_keys(info)) from orders;

11.To get datatype of value in json string

select json\_typeof(info->'items'->'qty') from orders;