***\d***

--List of Relations -- Type All

***\dt***

--List of Relations -- Type TABLE

***\dv***

--List of Relations -- TYPE VIEW

***\df***

--List of functions/store procedure

***\sf***

--Definition of functions/store procedure

***\d+*** <table\_name>/<view>

--describe the table/definition of view

***\l***

--List of databases

***\l+*** <database\_name>

--describe the database

***ALTER TABLE <table\_name> DISTRIBUTE BY REPLICATION;***

--Change the table hash distribution to replication

***SELECT \* FROM pgxc\_node;***

--list the pgxl node details

***SELECT COUNT(\*), xc\_node\_id FROM <table\_name> GROUP BY xc\_node\_id;***

--data distribution accross the nodes for table\_name

***ALTER TABLE <tablename> ADD NODE (<datanode\_name>);***

--add the table to the datanode\_name

***ALTER TABLE <tablename> DELETE NODE (<datanode\_name>);***

--delete the table from the datanode\_name

***ALTER NODE <node\_name> WITH (HOST='<IP>',PORT=<port>,PREFERRED,PRIMARY,TYPE='<coordinator/datanode>');***

--This will alter the <node\_name> depending upon the attributes

***SELECT pgxc\_pool\_reload();***

--This is necessary to reload the configuration whenever you alter the node configuration

***EXECUTE DIRECT on (<node\_name>) '<query>';***

--Example: EXECUTE DIRECT on (datanode\_1) 'SELECT \* FROM pgxc\_node';

--This help to exceute the query on another node from current node

***ALTER DATABASE <database\_name> RENAME TO <new\_database\_name>;***

--To rename the database

***ALTER DATABASE <database\_name> OWNER TO <role\_name> ;***

--To change the ower of the database

***ALTER DATABASE <database\_name> ALLOW\_CONNECTIONS boolean;***

--Example ALTER DATABASE alltrade ALLOW\_CONNECTIONS true;

--This give the privilege for enabling and disabling the connection

***CREATE DATABASE "alltrade"***

***ENCODING 'UTF8'***

***LC\_COLLATE = 'en\_US.UTF-8'***

***LC\_CTYPE = 'en\_US.UTF-8';***

--Create the database with collate and utf encoding

***SELECT \* FROM pg\_prepared\_xacts;***

--This will you gid you can roll back the prepared transaction

***ROLLBACK PREPARED <gid>;***

--This is to rollback the prepared transaction

***SELECT \* FORM pg\_stat\_activity;***

--This will give the current stat of the activities going on database like query,pid,state

***SELECT pg\_terminate\_backend(<pid>);***

--To terminate the query process

***SELECT pg\_cancel\_backend(pid)***

--To cancel the query process

***SELECT query, pid, state, (SELECT pg\_terminate\_backend(pid)) as killed from pg\_stat\_activity WHERE state LIKE 'idle';***

--This will terminate the query which are idle/ you can use correct column and value in WHERE condition to terminate more than one query

--Instead pg\_terminate\_backend you can also use pg\_cancel\_backend

**SHOW <parameter from postgressql.conf of that node>;**

--Example: SHOW max\_connections;

-- SHOW listen\_addresses;

--Show the current value set to that parameter

**SELECT**

**pg\_database.datname,**

**pg\_size\_pretty(pg\_database\_size(pg\_database.datname)) AS size**

**FROM pg\_database;**

--Gives the size of all databases

--Use datname = <database\_name> in WHERE condition to get particular database size

***SELECT***

***relname as "Table",***

***pg\_size\_pretty(pg\_total\_relation\_size(relid)) As "Size",***

***pg\_size\_pretty(pg\_total\_relation\_size(relid) - pg\_relation\_size(relid)) as "External Size"***

***FROM pg\_catalog.pg\_statio\_user\_tables ORDER BY pg\_total\_relation\_size(relid) DESC;***

--Gives the size of all tables

--Use relname = <table\_name> in WHERE condition to get particular table size

***UPDATE local\_service\_requests\_new8***

***SET order\_info\_error\_date\_time = (***

***CASE WHEN order\_info\_error\_date like '\_\_\_\_\_\_\_\_\_\_\_\_\_\_+\_\_\_\_' THEN to\_timestamp(order\_info\_error\_date, 'YYYYMMDDHH24MISS')***

***ELSE NULL***

***END);***

--Simple demonstration of CASE used in UPDATE

--Here I used to update the timestamp column with text column values by using the to\_timestamp function, if its in proper format it will update,

--or else it will be set to NULL

***INSERT INTO <table\_name> (int , text, text[], date, time, timestamp with time zone) values (<int>, '<text>', array['<text>', '<text>'], 'YYYY-MM-DD', 'HH:MM:SS', 'YYYY-MM-DD HH:MM:SS+07');***

--Example: INSERT INTO insert\_test (id , name, address, dates, times, timestamps) values (1, 'mahantesh', --array['Belgaum', 'Karnataka'], '2017-10-31',16:45:20', '2017-10-02 13:11:32+07');

--This demonstrate the insert of various datat-types in database

***CREATE SEQUENCE <sequence\_name>;***

--This will create the sequence, we use this to auto increment integer column in table

***ALTER TABLE <table\_name> ALTER <column\_name> SET DEFAULT NEXTVAL('<sequence\_name>');***

--Example: ALTER TABLE inc1\_test ALTER id SET DEFAULT NEXTVAL('inc\_test');

--This will add the sequence to the column for the table, the column will become autoincrement

***ALTER SEQUENCE <sequence\_name> RESTART <int>;***

--Example: ALTER SEQUENCE inc\_test RESTART 1;

--This restart the sequence from the number/int mentioned in the query