############Clone schema##########

Step01:- Make clone function

```
CREATE OR REPLACE FUNCTION clone_schema(
source_schema text,
dest_schema text)
RETURNS void AS
$BODY$
DECLARE
object text;
buffer text;
default_ text;
column_ text;
constraint_name_ text;
constraint_def_ text;
trigger_name_ text;
trigger_timing_ text;
trigger_events_ text;
trigger_orientation_ text;
trigger_action_ text;
BEGIN
-- replace existing schema
EXECUTE 'DROP SCHEMA IF EXISTS ' || dest_schema || ' CASCADE';
-- create schema
EXECUTE 'CREATE SCHEMA ' || dest_schema ;
-- create sequences
FOR object IN
SELECT sequence_name::text FROM information_schema.SEQUENCES WHERE
sequence_schema = source_schema
LOOP
EXECUTE 'CREATE SEQUENCE ' || dest_schema || '.' || object;
END LOOP;

-- create tables
FOR object IN
SELECT table_name::text FROM information_schema.TABLES WHERE table_schema =
source_schema
LOOP
buffer := dest_schema || '.' || object;
-- create table
EXECUTE 'CREATE TABLE ' || buffer || ' (LIKE ' || source_schema || '.' ||
object || ' INCLUDING CONSTRAINTS INCLUDING INDEXES INCLUDING DEFAULTS)';
-- fix sequence defaults
FOR column_, default_ IN
SELECT column_name::text, REPLACE(column_default::text, source_schema||'.',
dest_schema||'.') FROM information_schema.COLUMNS WHERE table_schema =
dest_schema AND table_name = object AND column_default LIKE 'nextval(%' ||
source_schema || '.%::regclass)'
LOOP
EXECUTE 'ALTER TABLE ' || buffer || ' ALTER COLUMN ' || column_ || ' SET
DEFAULT ' || default_;
END LOOP;
-- create triggers
FOR trigger_name_, trigger_timing_, trigger_events_, trigger_orientation_,
trigger_action_ IN
SELECT trigger_name::text, action_timing::text,
string_agg(event_manipulation::text, ' OR '), action_orientation::text,
action_statement::text FROM information_schema.TRIGGERS WHERE
event_object_schema=source_schema and event_object_table=object GROUP BY
trigger_name, action_timing, action_orientation, action_statement
LOOP
EXECUTE 'CREATE TRIGGER ' || trigger_name_ || ' ' || trigger_timing_ || ' ' ||
trigger_events_ || ' ON ' || buffer || ' FOR EACH ' || trigger_orientation_ ||
' ' || trigger_action_;
END LOOP;
END LOOP;
-- reiterate tables and create foreign keys
FOR object IN
SELECT table_name::text FROM information_schema.TABLES WHERE table_schema =
source_schema
LOOP
buffer := dest_schema || '.' || object;
-- create foreign keys
FOR constraint_name_, constraint_def_ IN
SELECT conname::text, REPLACE(pg_get_constraintdef(pg_constraint.oid),
source_schema||'.', dest_schema||'.') FROM pg_constraint INNER JOIN pg_class
ON conrelid=pg_class.oid INNER JOIN pg_namespace ON
pg_namespace.oid=pg_class.relnamespace WHERE contype='f' and relname=object
and nspname=source_schema
LOOP
EXECUTE 'ALTER TABLE '|| buffer ||' ADD CONSTRAINT '|| constraint_name_ ||'
'|| constraint_def_;
END LOOP;
END LOOP;
```

```
END;

$BODY$
LANGUAGE plpgsql VOLATILE
COST 100;
```

Step2:- create clone schema
Existing schema:- test
New schema:-write

Command:- `select clone_schema('test', 'write')`

With these command we can restore all table under old schema

NOTE:-

1. To change existing schema to new schema:-
Exsiting schame:- public
Want to connect:- test

Command:-SET search_path TO my_schema, public;

Example:-
Exsiting schame:- public
Want to connect:- test

SET search_path TO test, public;

2. To see current schema:-

Select current_schema;