**1(a). Create a trigger on Table of your choice to check if the Primary key ID already exists or not before inserting a new record. & Send a custom reply instead of an error message.**

CREATE OR REPLACE FUNCTION "cf\_db".Tri\_student()

RETURNS trigger

LANGUAGE 'plpgsql'

AS $$

DECLARE

i record;

BEGIN

FOR i in (select \* from "cf\_db"."students")

LOOP

IF(NEW.s\_id=i.s\_id) then

raise notice 'Student with same ID exists';

RETURN NULL;

END IF;

END LOOP;

RETURN NEW;

END

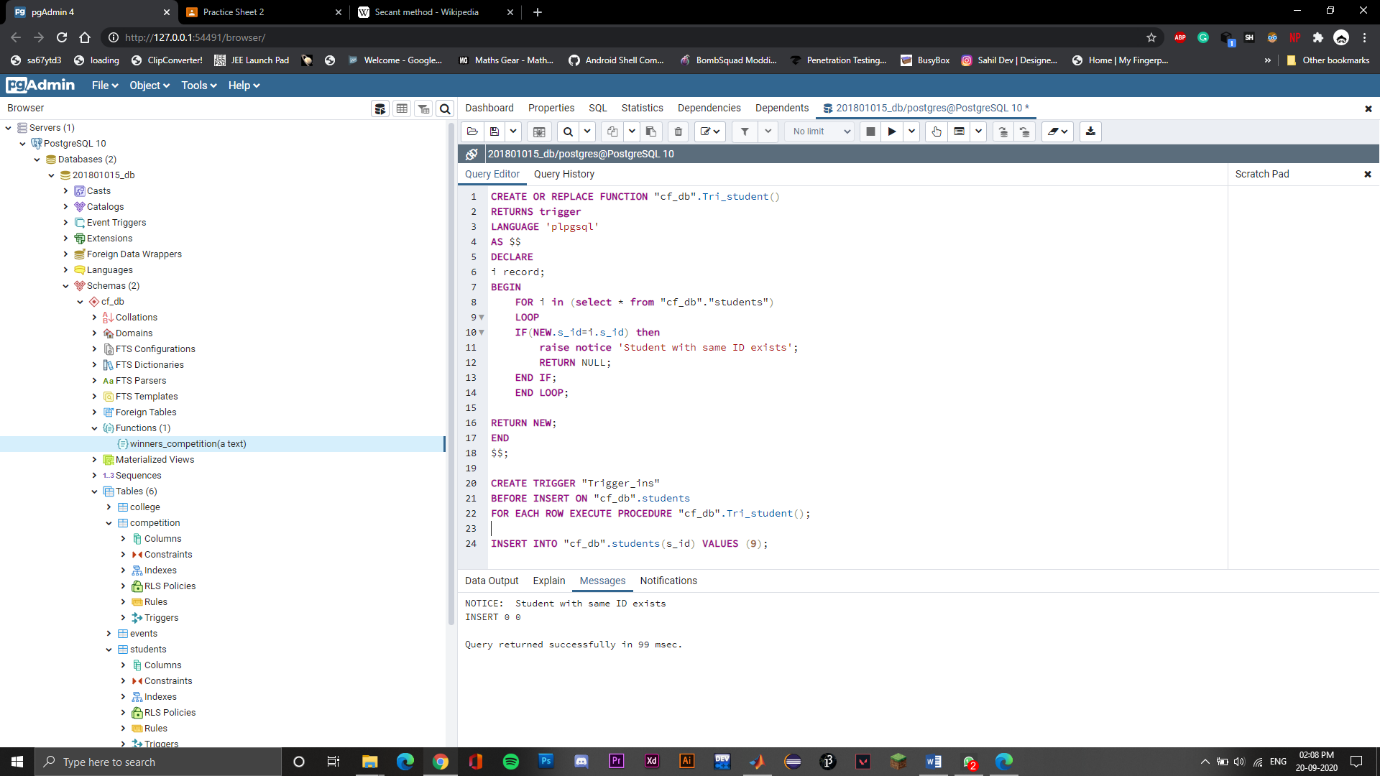
$$;

CREATE TRIGGER "Trigger\_ins"

BEFORE INSERT ON "cf\_db".students

FOR EACH ROW EXECUTE PROCEDURE "cf\_db".Tri\_student();

INSERT INTO "cf\_db".students(s\_id) VALUES (9);



**1(b). Create a trigger on Table of your choice to check if the Foreign key ID already exists or not before inserting a new record. & Send a custom reply instead of an error message.**

CREATE OR REPLACE FUNCTION "cf\_db".Tri\_comp()

RETURNS trigger

LANGUAGE 'plpgsql'

AS $$

DECLARE

curr\_e record;

BEGIN

FOR curr\_e in (select \* from "cf\_db"."events")

LOOP

IF(NEW.e\_id=curr\_e.e\_id) then

RETURN NEW;

END IF;

END LOOP;

raise notice 'No event exists for the given ID, creating link not possible';

RETURN NULL;

END

$$;

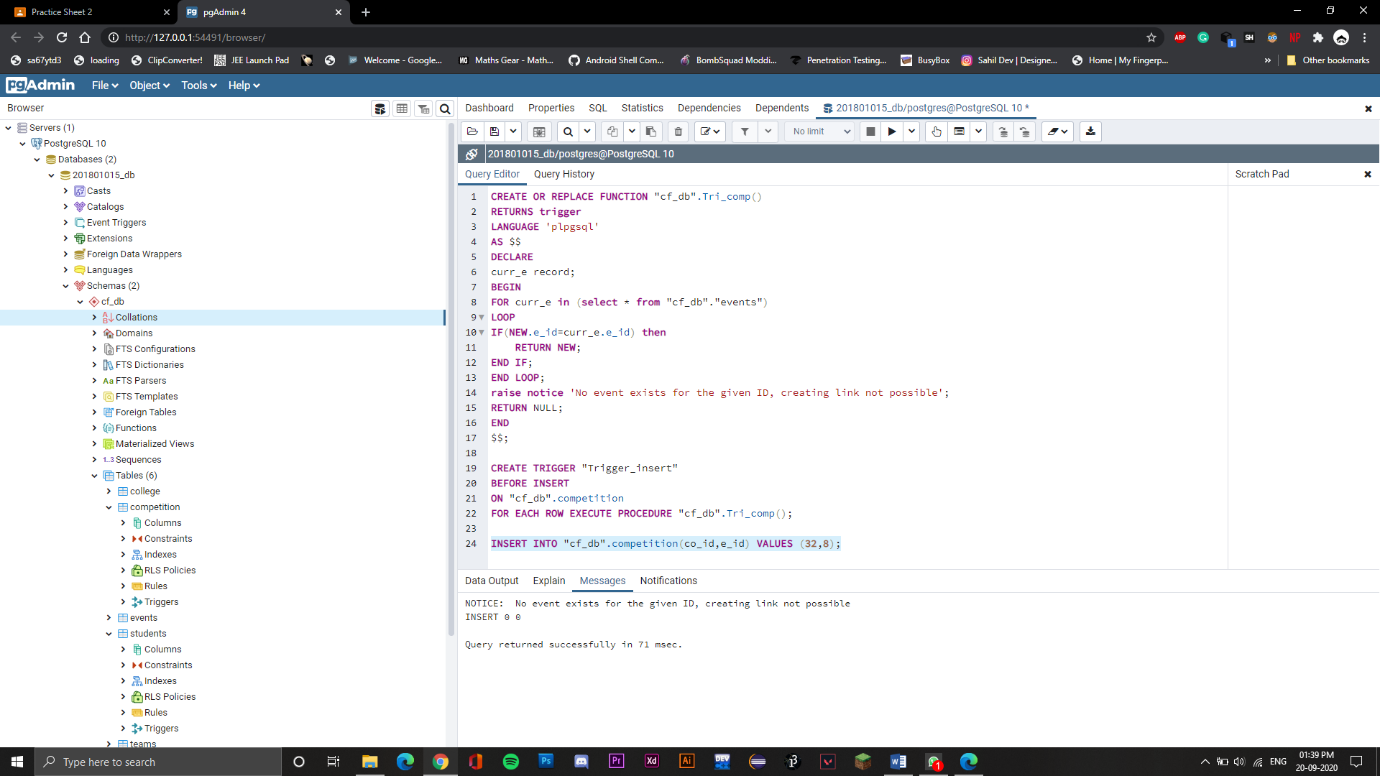
CREATE TRIGGER "Trigger\_insert"

BEFORE INSERT

ON "cf\_db".competition

FOR EACH ROW EXECUTE PROCEDURE "cf\_db".Tri\_comp();

INSERT INTO "cf\_db".competition(co\_id,e\_id) VALUES (32,8);



**2 (Stored Procedure) 1.**

**Create user defined function ‘winning\_competitions’ with one input argument that show the list of winning competitions for particular team name. (Consider top 3 place for winners)**

create or replace function "cf\_db"."winners\_competition"(s character varying(250))

returns table(b character varying(250))

language 'plpgsql'

as $$

declare

i RECORD;

begin

for i in (select competition.co\_name from CF\_DB.competition join (select teams.t\_name, winners.co\_id from CF\_DB.winners join CF\_DB.teams on winners.team\_id = teams.team\_id where winners.position <= 3 and teams.t\_name = s) as M on competition.co\_id = M.co\_id)

loop b:=(i.co\_name);

return next;

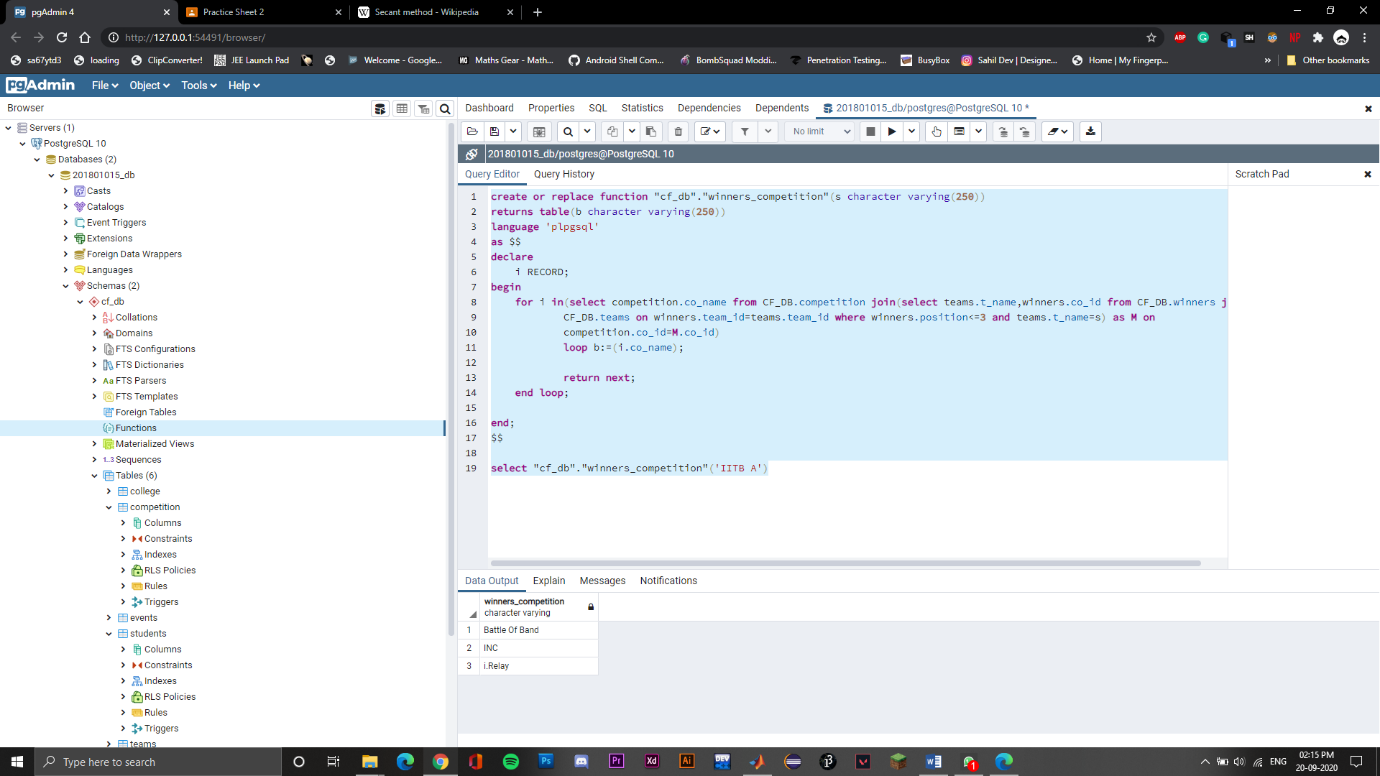
end loop;

end;

$$

select "cf\_db"."winners\_competition"('IITB A')

**3 rows**



**2 (Stored Procedure) 2.**

**Create user defined Function ‘events\_competition\_list’ that showing records for event’s competitions for particular event name. Competitions list must show like (co\_name,winning\_price, co\_date, co\_time)**

create or replace function "cf\_db"."events\_competition\_list"(s character varying(250))

returns table(b character varying(250),price integer,date\_1 character varying(100),time\_1 character varying(100))

language 'plpgsql'

as $$

declare

i RECORD;

begin

for i in(select competition.co\_name, competition.win\_price, competition.date , competition.time from CF\_DB.competition join CF\_DB.events on events.e\_id=competition.e\_id

where events.e\_name=s)

loop b:=(i.co\_name); price:=(i.win\_price); date\_1:=(i.date); time\_1:=(i.time);

return next;

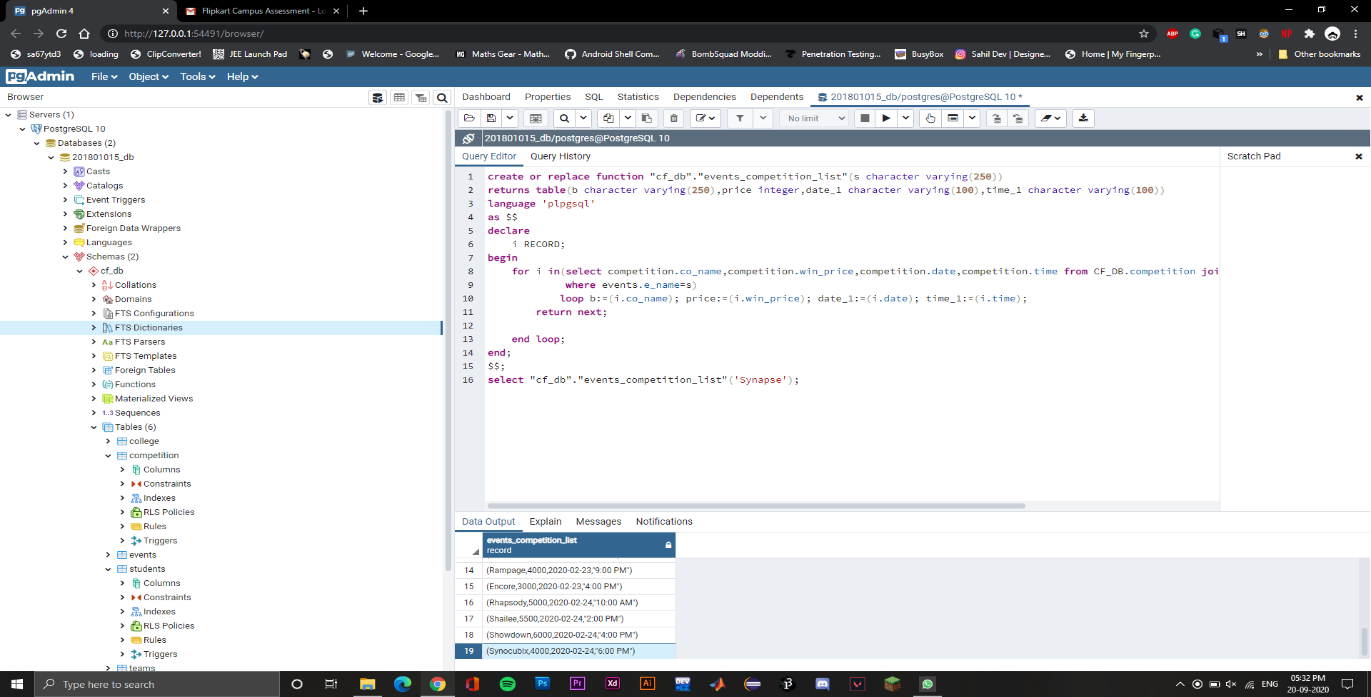
end loop;

end;

$$;

select "cf\_db"."events\_competition\_list"('Synapse');

**19 rows**



**2 (Trigger) 1**

**Create Trigger name for ‘teams’ table that check following conditions before insert record into this teams table. 1. check team id is null or not, 2. check team name is null or not, 3. check team strength less than 1 or not. For these conditions trigger shows respective error message like ‘team id should not be null’.**

create or replace function check\_teams()

returns trigger

language 'plpgsql'

as $$

begin

if(new.team\_id IS NULL)

then raise notice 'Team ID should not be null';

return old;

elsif (new.t\_name IS NULL)

then raise notice 'Team name should not be null';

return old;

elsif (new.t\_strength<1)

then raise notice 'Team strength should not be less than one';

return old;

end if;

return new;

end

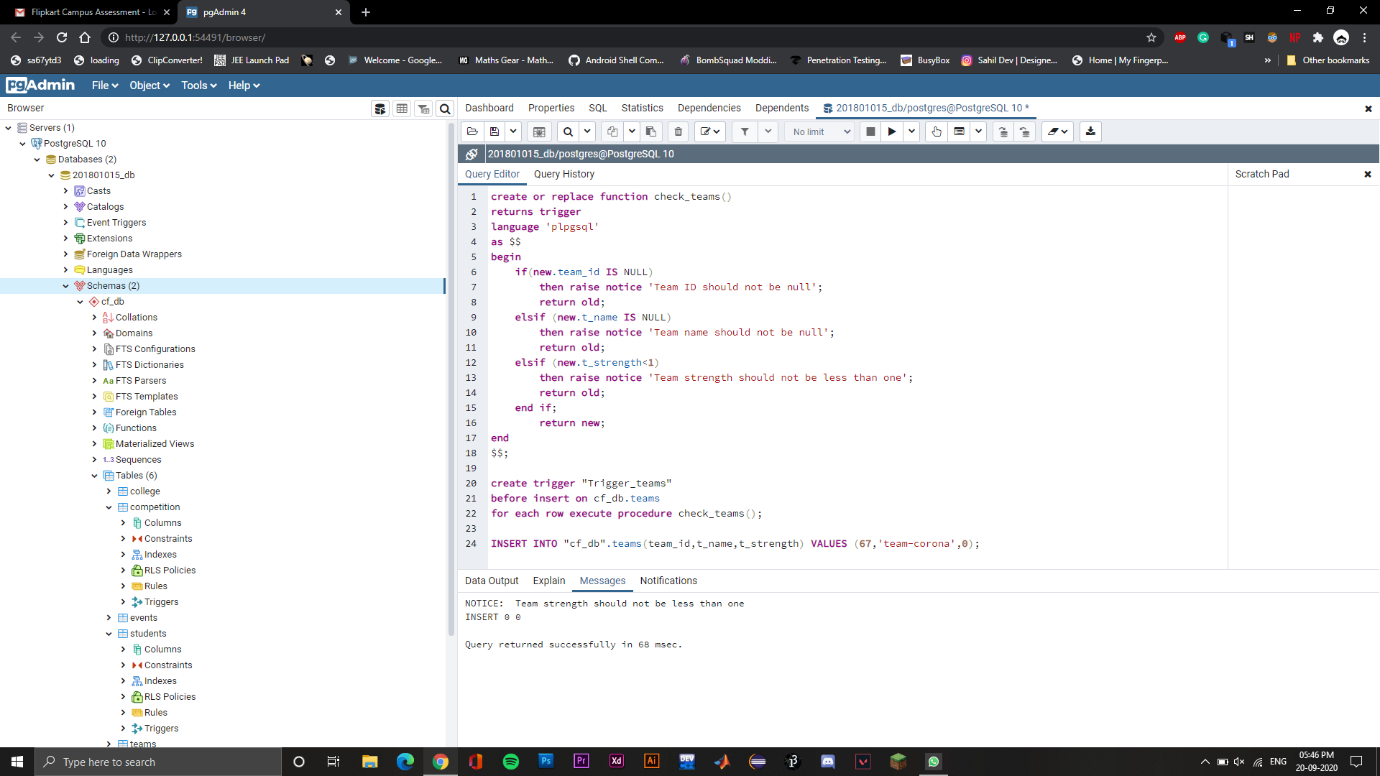
$$;

create trigger "Trigger\_teams"

before insert on cf\_db.teams

for each row execute procedure check\_teams();

INSERT INTO "cf\_db".teams(team\_id,t\_name,t\_strength) VALUES (67,'team-corona',0);



**2 (Trigger) 2.**

**Create Trigger for checking on table ‘competition’ for after insert data you need to check new added competition’s wining price is less than 2000 or not. If yes than you have to updated wining price with default value 2000.**

create or replace function check\_price()

returns trigger

language 'plpgsql'

as $$

begin

if(new.win\_price<2000)

then UPDATE cf\_db.competition SET win\_price = 2000 WHERE co\_id = new.co\_id;

raise notice 'The Win price entered is less than 2000, updated to default price money';

end if;

return new;

end

$$;

create trigger "Winning\_price\_check"

after insert on cf\_db.competition

for each row execute procedure check\_price();

INSERT INTO "cf\_db".competition(co\_id,e\_id,co\_name,win\_price,date,time) VALUES (72,3,'corona\_1',1000,'2020-02-21','6:00 PM');

select \* from cf\_db.competition

**42 rows**

