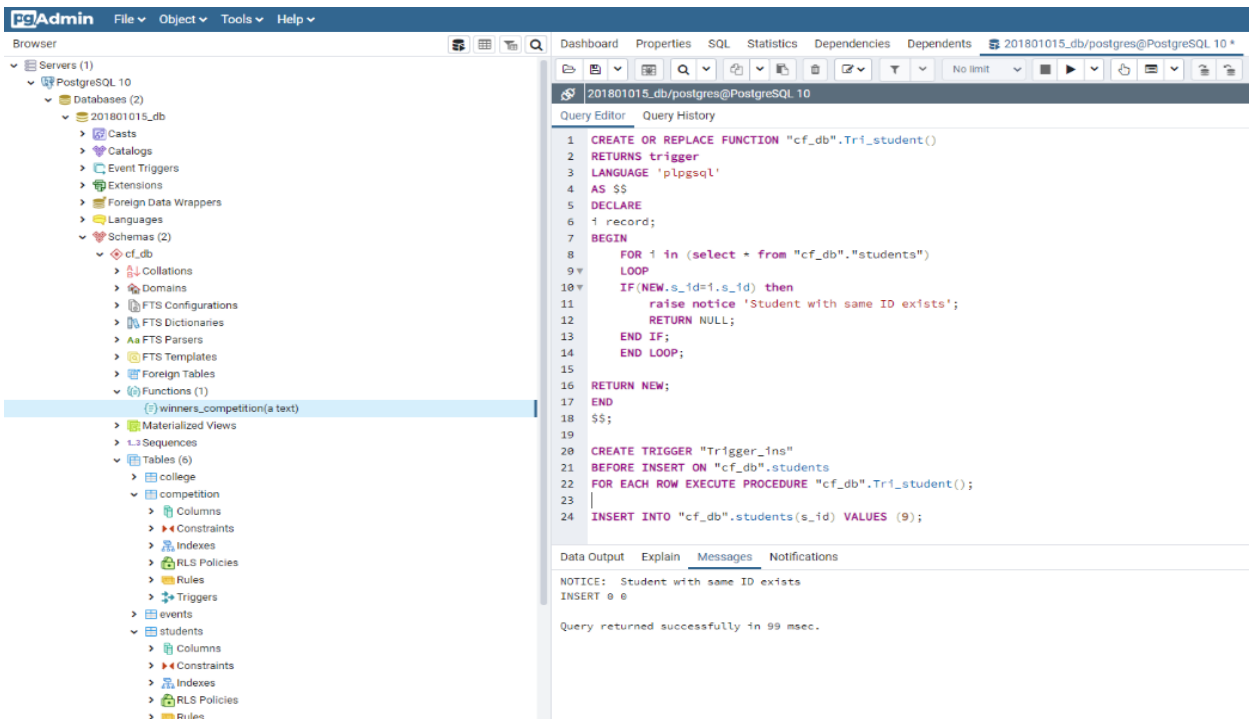
1(a). Create a trigger on Table of your choice to check if the Primary key ID already exists or not before inserting a new record. & Send a custom reply instead of an error message.

```
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);
```

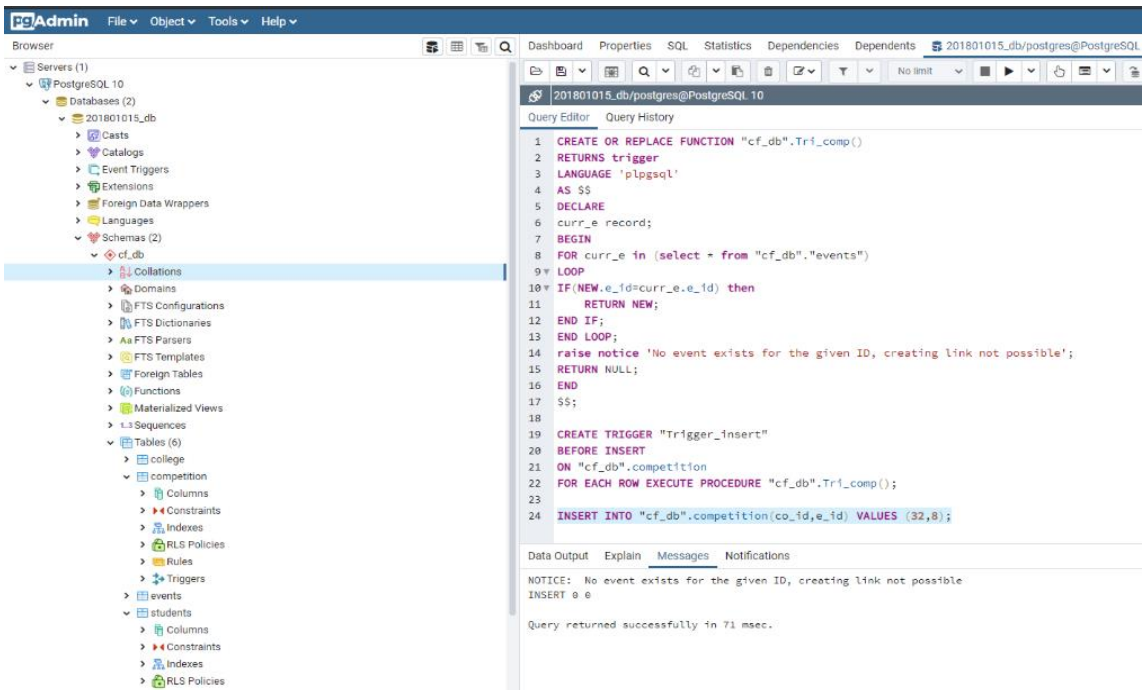


1(b). Create a trigger on Table of your choice to check if the Foreign key ID already exists or not before inserting a new record. & Send a custom reply instead of an error message.

```
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);
```



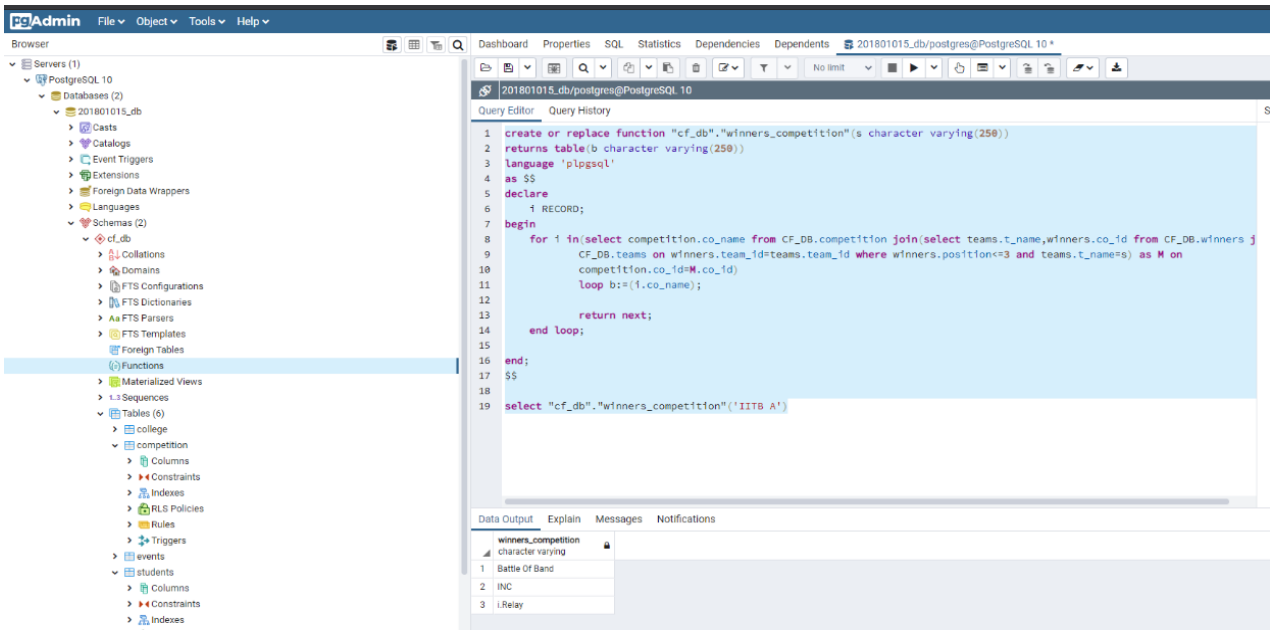
2 (Stored Procedure) 1.

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

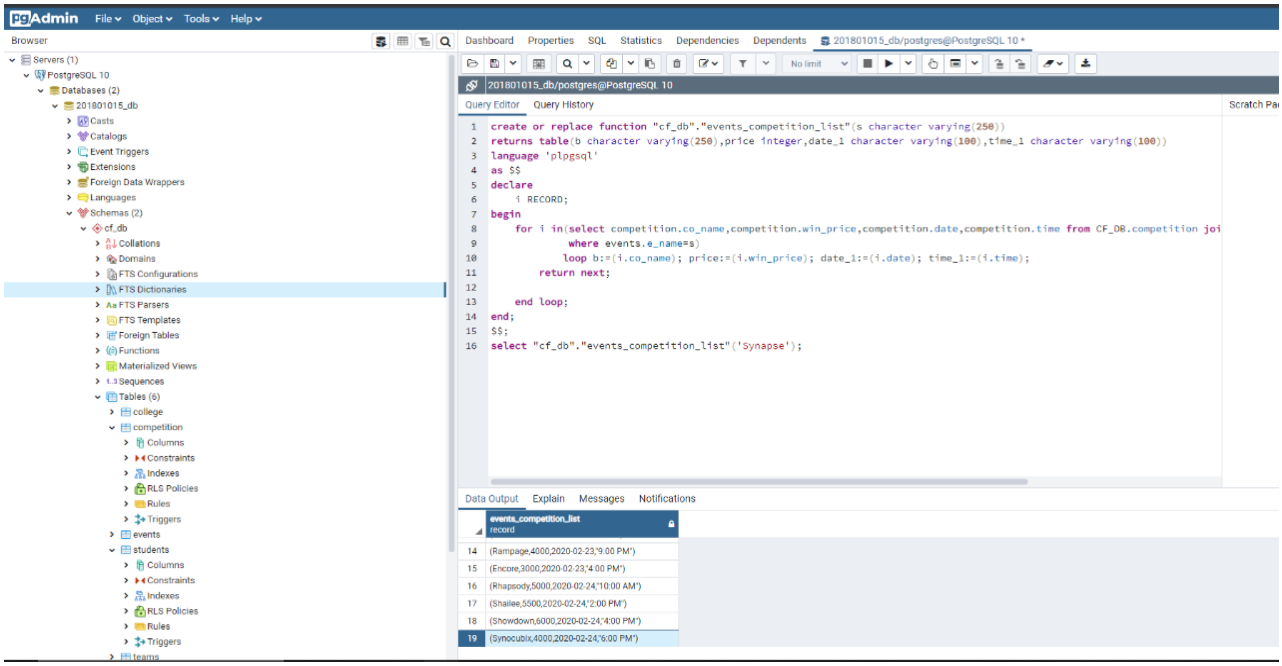


2 (Stored Procedure) 2.

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



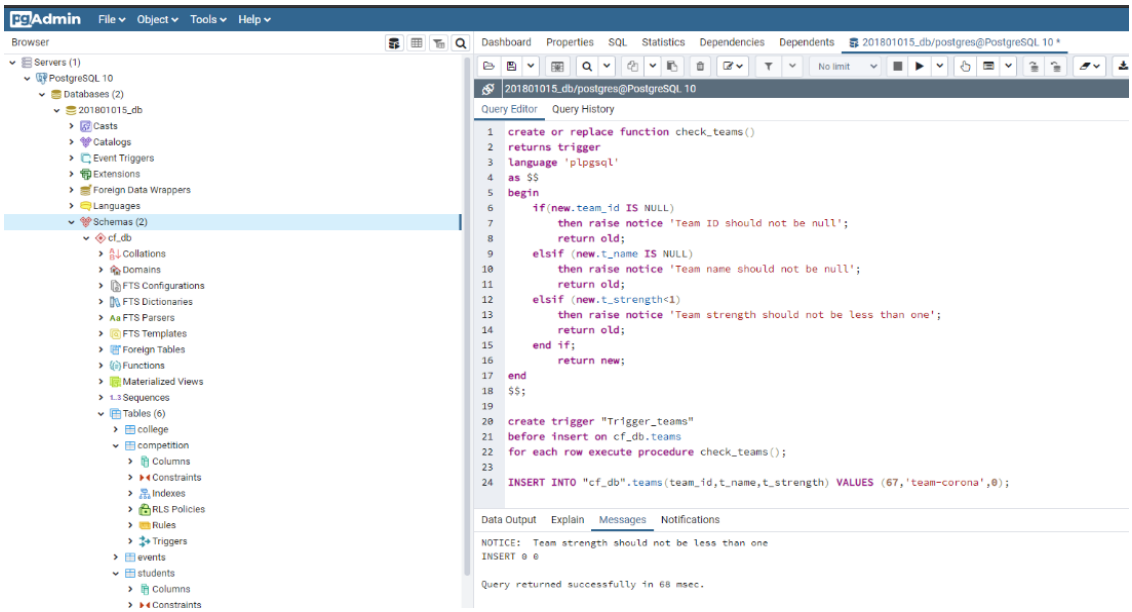
2 (Trigger) 1

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