**Stored Procedures**

**stored procedure for calculating avg score of participants in each sport**

set search\_path to paralympic;

drop function compute\_avg\_score\_individual();

drop type avg\_type;

create type avg\_type as (spid numeric,avg\_score\_sport numeric);

create or replace function compute\_avg\_score\_individual()

returns setof avg\_type as $BODY$

declare

sum\_score numeric; count\_participantid numeric(8,0); avg\_score\_sport numeric;

st sport%rowtype; mt match%rowtype; cit competesasindividual%rowtype;

rec avg\_type;

begin

for st in select \* from sport loop

sum\_score:=0; count\_participantid:=0;

for mt in select \* from match loop

if mt.sportid=st.sid then

for cit in select \* from competesasindividual loop

if cit.matchid=mt.matchid then

sum\_score:=sum\_score+cit.score;

count\_participantid:=count\_participantid+1;

end if;

end loop;

end if;

end loop;

if sum\_score>0 then

avg\_score\_sport=sum\_score/count\_participantid;

rec.spid=st.sid;

rec.avg\_score\_sport=avg\_score\_sport;

return next rec;

end if;

end loop;

return;

end $BODY$ language 'plpgsql';

**Stored procedure for removing invalid teams**

set search\_path to paralympic;

drop function team\_players\_count();

create or replace function team\_players\_count() returns setof total\_player as

$BODY$

declare

x int4; pi participatesinteam%rowtype; ti team%rowtype;

oi total\_player%rowtype;

begin

create temp table output as select teamid, count(participantid) from participatesinteam group by teamid;

for oi in select \* from output loop

if oi.total<5 then

for pi in select \* from participatesinteam loop

if oi.teamid=pi.teamid then

delete from participatesinteam where teamid=oi.teamid;

end if;

end loop;

for ti in select \* from team loop

if oi.teamid=ti.tid then

delete from participatesinteam where tid=oi.teamid;

end if;

end loop;

raise notice 'teamid: %s, has been removed!', quote\_ident(oi.teamid);

end if;

end loop;

drop table output;

end

$BODY$ language 'plpgsql';

**Triggers**

**Trigger for age violation**

create or replace function validation() returns trigger as $$

declare

x numeric; y numeric; z numeric;

begin

y=NEW.pid/10000;

z=extract(year from NEW.dob);

if(y-z>70 or y-z<15) then

raise notice 'Not allowed to participate';

else

insert into participant values(NEW.pname,NEW.country,new.gender,NEW.dob,NEW.pid);

end if;

return null;

end;

$$ language 'plpgsql';

create trigger validate\_age

before insert on participant

for each row

execute procedure validation();

**Trigger for not allowing match deletion**

create or replace function validate\_match\_deletion() returns trigger as $$

declare

x numeric; y numeric; z numeric; ci competesasindividual%rowtype; ct competesasteam%rowtype;

begin

if(TG\_OP='delete') then

for ci in select \* from competesasindividual loop

if (OLD.matchid=ci.matchid) then

raise notice 'You do not have access to delete this';

end if;

end loop;

for ct in select \* from competesasteam loop

if (OLD.matchid=ct.matchid) then

raise notice 'You do not have access to delete this';

end if;

end loop;

end if;

return null;

end;

$$ language 'plpgsql';

create trigger match\_deletion instead of delete on match

for each row

execute procedure validate\_match\_deletion();

**Trigger for checking country of team and participant to be same**

create or replace function team\_check() returns trigger as $$

declare

x int4; yc varchar(50); ti team%rowtype; pi participant%rowtype;

begin

x=0;

for ti in select \* from team loop

if(NEW.teamid=ti.tid) then

yc=ti.country;

end if;

end loop;

for pi in select \* from participant loop

if(NEW.participantid=pi.pid and yc=pi.country) then

insert into participatesinteam values(NEW.teamid,NEW.participantid);

x=1;

end if;

end loop;

if(x=0) then

raise notice 'Not allowed, Please try again!';

end if;

return null;

end;

$$ language 'plpgsql';

create trigger team\_verification

before insert on participatesinteam

for each row

execute procedure team\_check();