Stored Procedures

1. **Write a stored procedure that gives number of problems solved by given user:**

```
SET SEARCH_PATH TO codehorses;

CREATE OR REPLACE FUNCTION solved_problems_of(given_id INT) RETURNS INT AS
$BODY$

DECLARE


            total INTEGER;

            sub submission%ROWTYPE;

BEGIN

            total:=0;

            FOR sub IN SELECT * FROM submission WHERE given_id=userid

            LOOP

                        IF sub.status='AC' THEN

                         total=total+1;

                        END IF;

            END LOOP;


            RETURN total;




END;

$BODY$ LANGUAGE 'plpgsql';
```

2. **Write a stored procedure that gives the weakest category of the user:**

```
CREATE OR REPLACE VIEW combine AS SELECT * FROM submission NATURAL JOIN
tag NATURAL JOIN categorized;


CREATE OR REPLACE FUNCTION weakest_cat(given INTEGER) RETURNS
VARCHAR(20) AS $BODY$
DECLARE

            usr _user.userid%TYPE;
            sub submission%ROWTYPE;
            sub2 submission%ROWTYPE;
            comb combine%ROWTYPE;
            tg tag%ROWTYPE;
            cur INTEGER;
            total INTEGER;
            ans real:=100;
            x INTEGER;
            str VARCHAR(20);
BEGIN

            FOR tg IN SELECT * FROM tag
            LOOP
                  cur:=0;
                  total:=0;
                  FOR sub IN SELECT * FROM submission WHERE userid=given
                  LOOP
                        x:=0;
                        FOR comb IN SELECT * FROM combine WHERE
userid=given AND problemid=sub.problemid AND contestid=sub.contestid
```

```
                    LOOP
                            if comb.tagid=tg.tagid then
                            x=x+1;
                            end if;
                    end loop;

                    if x>0 then
                            if comb.status='ACCEPTED' then
                                    cur=cur+1;
                            end if;
                                    total=total+1;
                    end if;
            end loop;
            if total>0 then
                    if cur/total<ans then
                            ans=cur/total;
                            str=tg.tag_name;
                    end if;
            end if;
    END LOOP;
    RETURN str;


END;
$BODY$ LANGUAGE 'plpgsql';
```

Triggers

1. **Trigger for updating user's rating when he gives any contest and his rating is changed.**

```
CREATE OR REPLACE FUNCTION fun() RETURNS trigger

    AS

    $change$

    BEGIN

        IF (TG_OP = 'INSERT') THEN

            UPDATE _user

            SET rating = rating + NEW.ratingchange

            WHERE userid = NEW.userid;

            RETURN NEW;

        ELSEIF (TG_OP = 'DELETE') THEN

            UPDATE _user

            SET rating = rating - OLD.ratingchange

            WHERE userid = OLD.userid;

            RETURN OLD;

        ELSEIF (TG_OP = 'UPDATE') THEN

            UPDATE _user

            SET rating = rating - OLD.ratingchange

            WHERE userid = OLD.userid;


            UPDATE _user

            SET rating = rating + NEW.ratingchange

            WHERE userid = NEW.userid;

            RETURN NEW;
```

```
                END IF;
        RETURN NULL;
        END;
        $change$ LANGUAGE plpgsql;

create trigger fun
before insert or delete or update on gives
for each row execute procedure fun();
```

2. **Trigger for updating user's contribution whenever user writes blog.**

```
CREATE OR REPLACE FUNCTION blog() RETURNS trigger
        AS
        $change$
        BEGIN
                IF (TG_OP = 'INSERT') THEN
                        UPDATE _user
                                SET contribution = contribution + NEW.upvote -
        NEW.downvote
                        WHERE userid = NEW.writerid;
                        RETURN NEW;
                ELSEIF (TG_OP = 'DELETE') THEN
                        UPDATE _user
                                SET contribution = contribution + NEW.upvote -
        NEW.downvote
                        WHERE userid = OLD.writerid;
                        RETURN OLD;
                ELSEIF (TG_OP = 'UPDATE') THEN
                        UPDATE _user
                                SET contribution = contribution + NEW.upvote -
        NEW.downvote
                        WHERE userid = OLD.writerid;


                        UPDATE _user
                                SET contribution = contribution + NEW.upvote -
        NEW.downvote
                        WHERE userid = NEW.writerid;
                        RETURN NEW;
                END IF;
        RETURN NULL;
        END;
        $change$ LANGUAGE plpgsql;


create trigger blog
before insert or delete or update on blog
for each row execute procedure blog();
```