1. Les vues

1. Créez une vue nommée pts_GP_Malaisie

```
CREATE OR REPLACE VIEW pts_GP_Malaisie(prenom, nom, Position_Arrivee, Pts_Asquis)
  AS
SELECT nompilote, prenompilote, positionarrivee,
        coalesce((select points from bareme where place=positionarrivee::int),0)
  FROM grandprix G JOIN courir C ON C.idgp=G.idgp
                   JOIN pilote P ON P.idpilote=C.idpilote
  WHERE nomgp='Grand Prix de Malaisie'
  AND positionarrivee !='A'
  AND positionarrivee !='0' ;
```

test le de la vue : `SELECT * FROM pts_GP_Malaisie;`

prenom	nom	position_arrivee	pts_asquis
Rosberg	Nico	1	25
Kvyat	Daniil	2	18
Bottas	Valtteri	3	15
Nasr	Felipe	4	12
Pérez	Sergio	5	10
Hülkenberg	Nico	6	8
Massa	Felipe	7	6
Ricciardo	Daniel	8	4
Hamilton	Lewis	9	2
Gutiérrez	Esteban	10	1
Alonso	Fernando	11	0
Button	Jenson	12	0
Grosjean	Romain	13	0
Palmer	Jolyon	14	0
Verstappen	Max	16	0
Räikkönen	Kimi	17	0
Ericsson	Marcus	18	0
Vettel	Sebastian	19	0
Wehrlein	Pascal	20	0
Sainz Jr.	Carlos	21	0

2. Créez une vue nommée nb_km_GP,

```
CREATE OR REPLACE VIEW nb_km_GP(nomGP, longueurGP) AS
  SELECT nomgp, nbtour*longpiste
  FROM grandprix G JOIN circuit C ON C.id_circuit=G.id_circuit ;
```

*test le de la vue : SELECT * FROM nb_km_GP ;*

nomgp	longueurgp
Grand Prix d'Australie	577.20
Grand Prix des Etats-Unis	1070.88
Grand Prix de Bahreïn	1330.86
Grand Prix d'Espagne	1145.40
Grand Prix du Canada	819.00
Grand Prix d'Allemagne	1335.00
Grand Prix de Hongrie	921.20
Grand Prix du Brésil	1190.40
Grand Prix de Monaco	846.80
Grand Prix d'Italie	1905.00
Grand Prix d'Autriche	2214.80
Grand Prix de Malaisie	1009.92
Grand Prix de Chine	3069.40
Grand Prix de Grande-Bretagne	1470.00
Grand Prix de Singapour	834.60
Grand Prix de Belgique	2440.00
Grand Prix de Russie	3064.40
Grand Prix du Japon	1405.80
Grand Prix d'Abou Dabi	765.60

19 ligne(s)

2. Les règles

1. Règle sur un update sur la vue *view_position_arr_pilote*

2.1.a / affiche le résultat de la vue

```
SELECT * FROM view_position_arr_pilote;
```

2.1.b /

```
CREATE or replace rule regle_update_view_position_arr_pilote
AS ON UPDATE TO view_position_arr_pilote
DO INSTEAD
(
  --  modification du nom et/ou prenom du pilote
  UPDATE pilote
  SET nompilote = NEW.nom,
      prenompilote = NEW.prenom
  WHERE idpilote = OLD.id_pilote;

  --  modification du nom et/ou prenom du pilote
  UPDATE courir
  SET positionarrivee = NEW.position
  WHERE idpilote = OLD.id_pilote AND idgp=OLD.id_gp;
) ;
```

Test de la règle

```
  UPDATE view_position_arr_pilote
  SET nom='DURAND',
      prenom='Carlos',
      position=20
  WHERE id_pilote=18 AND id_gp=19 ;

  UPDATE view_position_arr_pilote
  SET position=1
  WHERE id_pilote=18 AND id_gp=19
```

2. Règle sur un insert sur la table circuit

```
CREATE RULE regle_insert_circuit  AS ON INSERT TO circuit
WHERE exists (SELECT * FROM circuit WHERE id_circuit=NEW.id_circuit)
DO INSTEAD
  UPDATE circuit
  SET  nomcircuit   =NEW.nomcircuit,
       Payscircuit  =NEW.payscircuit,
       Longpiste    =NEW.longpiste,
       Nbspectateur =NEW.nbspectateur
  WHERE id_circuit  =NEW.id_circuit;
```

-- insertion effective SANS déclenchement de la règle

```
   INSERT INTO circuit  VALUES (20,'le miens','France', 5, 6700);
```

-- insertion effective AVEC déclenchement de la règle

```
   INSERT INTO circuit  VALUES (20,'Le castelet','France', 5, 34000);
```

3. Les fonctions

1. Ecrire la fonction *tour_Effectue(identifiant du grand prix)*

```
CREATE OR REPLACE function tour_Effectue(p_idgp int) RETURNS  SETOF boolean
  AS $$
BEGIN
  RETURN QUERY  SELECT nbtourseffectue >= (nbtour/2)
                  FROM grandprix WHERE idgp=p_idgp ;
END; $$ LANGUAGE plpgsql;
```

Test la focntion :

```
SELECT  tour_Effectue(1);   -- renvoie TRUE
SELECT  tour_Effectue(18);  -- renvoie FALSE
```

2. Ecrire la fonction *point_acquis(date, identifiant du pilote)*

```
CREATE OR REPLACE function point_acquis(p_date date, p_idpilote int)
RETURNS int AS $$
DECLARE
   v_point integer;
BEGIN
  PERFORM * FROM pilote  WHERE idpilote = p_idpilote;

  IF NOT FOUND
  THEN   RAISE EXCEPTION 'le pilote N°% nexiste pas', p_idpilote;
  END IF;

  -- sous requête corrélée pour calculer les points acquis
  SELECT SUM(COALESCE((SELECT points FROM bareme WHERE
  place=positionarrivee::int),0))  INTO v_point
  FROM courir
  WHERE positionarrivee!='A'
  AND idpilote =  p_idpilote
  AND idgp IN (SELECT idgp FROM grandprix WHERE dategp <= p_date) ;

RETURN v_point ;
END; $$ LANGUAGE plpgsql;
```

Test de la fonction

```
SELECT point_acquis('2017-01-30',1) ;  -- renvoie 80
SELECT point_acquis('2016-12-15',2) ;  -- renvoie 71
SELECT point_acquis('2016-12-15',56) ;
```

Erreur SQL :

ERROR: le pilote N°56 n'existe pas

Voir page suivante pour la solution avec curseur

Solution avec curseur

```
CREATE OR REPLACE function point_acquis(p_date date, p_idpilote int)
RETURNS int AS $$
DECLARE
      v_point integer;
      v_place integer;
      v_point_tot integer := 0;
      curpts CURSOR(d date, id integer)
                  IS SELECT positionarrivee::int FROM courir
                  WHERE positionarrivee!='A'
                  AND idpilote = id
                  AND idgp IN (SELECT idgp FROM grandprix WHERE dategp <=d);
BEGIN
      PERFORM * FROM pilote  WHERE idpilote = p_idpilote;

      IF NOT FOUND
      THEN  RAISE EXCEPTION 'le pilote N°% n''existe pas', p_idpilote;
      END IF;

      OPEN curpts(p_date, p_idpilote);
      LOOP
        FETCH curpts into v_point;
        EXIT WHEN NOT FOUND;
        v_place :=(SELECT points FROM bareme WHERE place= v_point);
        v_point_tot :=v_point_tot + COALESCE(v_place,0);
      END LOOP;

      CLOSE curpts;

      RETURN v_point_tot ;

END; $$ LANGUAGE plpgsql;
```

Test de la fonction

```
SELECT point_acquis('2017-01-30',1)  -- renvoie 80
SELECT point_acquis('2016-12-15',2)  -- renvoie 71
```