Carreras de Formula 1

Soluciones a los procedimientos, funciones y triggers:

1.
```sql
DELIMITER $$
CREATE PROCEDURE `getRacesInAYear`(IN `year` INT)
BEGIN
        SELECT races.name, COUNT(DISTINCT results.constructorId) AS numConstructors
    FROM results
            INNER JOIN races ON races.raceId = results.raceId
    WHERE races.year = year
    GROUP BY races.raceId;
END$$
DELIMITER ;
```

2.
```sql
DELIMITER $$
CREATE PROCEDURE getsOnRaceMessages(IN cod VARCHAR(3), OUT msg VARCHAR(200))
BEGIN
        CASE cod
    WHEN 'E01' THEN SET msg = 'Error en la presión de las ruedas';
    WHEN 'E02' THEN SET msg = 'Pinchazo';
    WHEN 'E03' THEN SET msg = 'Temperatura alta en el motor';
    WHEN 'E04' THEN SET msg = 'Frenos sobre-calentados';
    WHEN 'E05' THEN SET msg = 'Error presión del aceite';
    ELSE SET msg = 'Error de comando';
    END CASE;
END$$
DELIMITER ;
```

3.
```sql
DELIMITER $$
CREATE PROCEDURE getDriversByNationality (IN nat VARCHAR(250))
BEGIN
    SELECT *
    FROM drivers
    WHERE nationality = nat;
END$$
DELIMITER ;
```

4.
```sql
DELIMITER $$
CREATE FUNCTION puntosCampeon (year INTEGER)
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
        DECLARE points DECIMAL(10,2);

    SELECT MAX(T.totalPoints) INTO points
    FROM (SELECT SUM(results.points) AS totalPoints
          FROM results
              INNER JOIN races ON races.raceId = results.raceId
          WHERE races.year = year
          GROUP BY results.driverId) AS T;

    RETURN (points);
END$$
DELIMITER ;
```

5.
```sql
DELIMITER $$
CREATE FUNCTION mediaPuntosConstructor (constructor VARCHAR(200))
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
        DECLARE points DECIMAL(10,2);

    SELECT AVG(T.totalPoints) INTO points
    FROM (SELECT SUM(results.points) AS totalPoints
          FROM results
              INNER JOIN races ON races.raceId = results.raceId
```

```
            INNER JOIN constructors ON constructors.constructorId = results.constructorId
        WHERE constructors.name = constructor
        GROUP BY races.year) AS T;

    RETURN (points);
END$$
DELIMITER ;
```

6.
```
DELIMITER $$
CREATE FUNCTION añosEnActivo (id INTEGER)
RETURNS INTEGER
DETERMINISTIC
BEGIN
        DECLARE años INTEGER;

    SELECT COUNT(DISTINCT races.year) INTO años
    FROM results
            INNER JOIN races ON races.raceId = results.raceId
    WHERE results.driverId = id;

    RETURN (años);
END$$
DELIMITER ;
```

7.
```
ALTER TABLE drivers ADD COLUMN añosEnActivo INTEGER NULL;

DELIMITER $$
CREATE PROCEDURE actualizarAñosEnActivo ()
BEGIN
    DECLARE done INTEGER DEFAULT FALSE;
    DECLARE id INTEGER;

    DECLARE cur CURSOR FOR SELECT driverId FROM drivers;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
    OPEN cur;
    read_loop: LOOP
        FETCH cur INTO id;

        UPDATE drivers
            SET drivers.añosEnActivo = añosEnActivo(id)
            WHERE driverId = id;

        IF done THEN
            LEAVE read_loop;
        END IF;
    END LOOP;
    CLOSE cur;
END$$
DELIMITER ;
```

8.
```
-------------TABLA results ----------
DELIMITER $$
CREATE TRIGGER noMasDeDosPilotos
BEFORE INSERT ON results
FOR EACH ROW
BEGIN
    DECLARE numResults INTEGER;
    DECLARE numDrivers INTEGER;

    SELECT COUNT(*) INTO numResults
    FROM results
    WHERE constructorId = NEW.constructorId
      AND driverId = NEW.driverId;

    IF numResults = 0 THEN
        SELECT COUNT(DISTINCT driverId) INTO numDrivers
        FROM results
        WHERE constructorId = NEW.constructorId;

        IF numDrivers > 1 THEN
            SIGNAL SQLSTATE '03000'
            SET MESSAGE_TEXT = 'Error: no puede haber equipos con más de dos pilotos';
        END IF;
```

```
    END IF;
END$$
DELIMITER ;
------------TABLA qualifying ----------
DELIMITER $$
CREATE TRIGGER noMasDeDosPilotos
BEFORE INSERT ON qualifying
FOR EACH ROW
BEGIN
    DECLARE numResults INTEGER;
    DECLARE numDrivers INTEGER;

    SELECT COUNT(*) INTO numResults
    FROM qualifying
    WHERE constructorId = NEW.constructorId
      AND driverId = NEW.driverId;

    IF numResults = 0 THEN
        SELECT COUNT(DISTINCT driverId) INTO numDrivers
        FROM qualifying
        WHERE constructorId = NEW.constructorId;

        IF numDrivers > 1 THEN
            SIGNAL SQLSTATE '03000'
            SET MESSAGE_TEXT = 'Error: no puede haber equipos con más de dos pilotos';
        END IF;
    END IF;
END$$
DELIMITER ;
```

9.

```
CREATE TABLE crashes (
           crashId INTEGER UNIQUE NOT NULL AUTO_INCREMENT,
    driverId INT NOT NULL,
    description VARCHAR(250) DEFAULT NULL,
          PRIMARY KEY (crashId),
        CONSTRAINT
                FOREIGN KEY (driverId)
                REFERENCES drivers (driverId)
)

DELIMITER $$
CREATE TRIGGER resgistrarAccidentes
AFTER INSERT ON results
FOR EACH ROW
BEGIN
    IF NEW.statusId = 3 OR NEW.statusId = 4 THEN
            INSERT INTO crashes (driverId, description) VALUES (NEW.driverId, 'blah blah blah');
    END IF;
END$$
DELIMITER ;
```

10.

```
-- Para la solución se asume que "estar" en un equipo es participar en una carrera (tabla results)

DELIMITER $$
CREATE TRIGGER no_more_than_one_teams_in_a_year
BEFORE INSERT ON results
FOR EACH ROW
BEGIN
    DECLARE y YEAR;
    DELCARE num_races INTEGER;
    DELCARE num_races_with_constructor INTEGER;

    SELECT year INTO y FROM races WHERE raceId = NEW.raceId;

    SELECT COUNT(*), COUNT(constructorId = NEW.constructorId) INTO num_races,
    ↪  num_races_with_constructor
    FROM results
        INNER JOIN races ON races.raceId = results.raceId
    WHERE races.year = y
      AND driverId = NEW.driverId;

    IF num_races > 0 AND num_races_with_constructor = 0 THEN
        SIGNAL SQLSTATE '03000'
        SET MESSAGE_TEXT = 'Error: no se puede competir con un equipo con el que no hayas competido
        ↪  en ese año';
```

```
    END IF;
END$$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER no_more_than_two_teams_in_a_year
BEFORE UPDATE ON results
FOR EACH ROW
BEGIN
    DECLARE y YEAR;
    DELCARE num_races INTEGER;
    DELCARE num_races_with_constructor INTEGER;

    IF NEW.raceId <> OLD.raceId OR NEW.driverId <> OLD.driverId OR NEW.constructorId <>
    ↪  OLD.constructorId THEN
        SELECT year INTO y FROM races WHERE raceId = NEW.raceId;

        SELECT COUNT(*), COUNT(constructorId = NEW.constructorId) INTO num_races,
        ↪  num_races_with_constructor
        FROM results
            INNER JOIN races ON races.raceId = results.raceId
        WHERE races.year = y
        AND driverId = NEW.driverId;

        IF num_races > 0 AND num_races_with_constructor = 0 THEN
            SIGNAL SQLSTATE '03000'
            SET MESSAGE_TEXT = 'Error: no se puede competir con un equipo con el que no hayas
            ↪  competido en ese año';
        END IF;
    END IF;
END$$
DELIMITER ;
```

11.
```
DROP PROCEDURE pilots_win_home_constructor_year;

DELIMITER $$
CREATE PROCEDURE pilots_win_home_constructor_year(IN year_win INTEGER)
BEGIN
    SELECT DISTINCT drivers.forename, drivers.surname, circuits.name
    FROM drivers INNER JOIN results ON drivers.driverId = results.driverId INNER JOIN races ON
    ↪  results.raceId=races.raceId INNER JOIN constructors ON
    ↪  results.constructorId=constructors.constructorId INNER JOIN circuits ON
    ↪  circuits.circuitId=races.circuitId
    WHERE results.positionOrder=1 AND races.year=year_win AND
    ↪  drivers.nationality=constructors.nationality;
END$$
DELIMITER ;
```

12.
```
DELIMITER //
CREATE PROCEDURE allPodiumPositions(IN anyo INTEGER)
BEGIN
SELECT forename, surname
FROM drivers D
WHERE driverId IN(SELECT driverId
                  FROM results JOIN races ON results.raceId=races.raceId
                  WHERE positionOrder=1
                  AND year=anyo)
AND driverId IN(SELECT driverId
                  FROM results JOIN races ON results.raceId=races.raceId
                  WHERE positionOrder=2
                  AND year=anyo)
AND driverId IN(SELECT driverId
                  FROM results JOIN races ON results.raceId=races.raceId
                  WHERE positionOrder=3
                  AND year=anyo);
END//
DELIMITER ;
```

13.
```
DELIMITER $$
CREATE FUNCTION diffPoints (driver1 INTEGER, driver2 INTEGER)
RETURNS DOUBLE
DETERMINISTIC
BEGIN
```

```
    DECLARE points1 DOUBLE;
    DECLARE points2 DOUBLE;

    SELECT SUM(points) INTO points1
    FROM results
    WHERE driverId = driver1;

    SELECT SUM(points) INTO points2
    FROM results
    WHERE driverId = driver2;

    RETURN (points1 - points2);
END$$
DELIMITER ;
```

14.
```
CREATE TABLE sponsors(
    sponsorId INTEGER UNIQUE NOT NULL,
    name VARCHAR(50) NOT NULL,
    type VARCHAR(20),
    amount INTEGER NOT NULL,
    raceId INTEGER,
    PRIMARY KEY(sponsorId),
    CONSTRAINT
    FOREIGN KEY(raceId)
    REFERENCES formula1.races(raceId))

DELIMITER //
CREATE TRIGGER check_spon1 BEFORE INSERT ON sponsors
FOR EACH ROW
BEGIN
IF NEW.amount>5000000 THEN SET NEW.type='Oficial';
ELSE SET NEW.type='Co-oficial';
END IF;
END//

DELIMITER //
CREATE TRIGGER check_spon2 BEFORE UPDATE ON sponsors
FOR EACH ROW
BEGIN
IF NEW.amount>5000000 THEN SET NEW.type='Oficial';
ELSE SET NEW.type='Co-oficial';
END IF;
END//
DELIMITER ;
```

15.
```
DELIMITER //
CREATE PROCEDURE `getsConstructoresYPilotos` (año year)
BEGIN
    SELECT constructors.name, drivers.surname, SUM(results.points)
    FROM constructors JOIN results ON constructors.constructorId = results.constructorID
                JOIN races ON results.raceId = races.raceId
        JOIN drivers ON results.driverId = drivers.driverId
        WHERE races.`year`= año
    GROUP BY constructors.name, drivers.surname;
END//
DELIMITER ;
```

16.
```
DELIMITER $$
CREATE PROCEDURE getNumberOfVictories (IN type VARCHAR(20))
BEGIN
    IF type = 'nationality' THEN
        SELECT drivers.nationality, COUNT(*) AS numVictorires
        FROM results
            INNER JOIN drivers ON drivers.driverId = results.driverId
        WHERE results.positionOrder = 1
        GROUP BY drivers.nationality
        ORDER BY numVictorires DESC;
    ELSE
        SELECT constructors.name, COUNT(*) AS numVictorires
        FROM results
            INNER JOIN constructors ON constructors.constructorId = results.constructorId
        WHERE results.positionOrder = 1
        GROUP BY constructors.name
```

```
        ORDER BY numVictorires DESC;
    END IF;
END$$
DELIMITER ;
```