SAVA DARIUS-STEFAN BAZA DE DATE FORMULA 1 GRUPA 142

FORMULA 1

Realizat de Sava Darius-Stefan

Grupa 142

Anul I

CUPRINS

1. Descrierea modelului real, a utilitatii acestuia si a regulilor de functionare3

2. Prezentarea constrangerilor (restrictii, reguli) impuse asupra modelului3

3. Descrierea entitatilor, incluzand precizarea cheii primare4

4. Descrierea relatiilor, incluzand precizarea cardinalitatii acestora4

5. Descrierea atributelor, incluzand tipul de date si eventualele constangeri, valori implicite, valori posibile ale atributelor5

6. Realizarea diagramei etitate-relatie corespunzatoare descrierii de la punctele 3-57

7. Realizarea diagramei conceptual corespunzatoare diagramei entitate-relatie proiectate la punctul 67

8. Enumerarea schemelor relaționale corespunzătoare diagramei conceptuale proiectate la punctul 78

## 9. Realizarea normalizarii pana la forma normal 3 (FN1 - FN3)8

### 10. Crearea unei secvente ce va fi utilizata in inserarea inregistrarilor in tabele (punctul 11)11

### 11. Crearea tabelelor SQL si inserarea de date coerente in fiecare dintre acestea (minimum 5 inregistrari in fiecare tabel neasociativ; minim 10 inregistrari in tabelele associative; maxim 30 de inregistrari in fiecare tabel).13

12. Formularea in limbaj natural si implementarea a 5 cereri SQL complexe27

13. Implementarea a 3 operatii de actualizare si de suprimare a datelor utilizand subcereri 32

PROIECT BAZA DE DATE

FORMULA 1

**1. Descrierea modelului real, a utilitatii acestuia si a regulilor de functionare.**

**Modelul real:**

Acest model reprezinta un sistem de gestionare a datelor pentru o competitie de Fomrula 1. El urmareste sa organizeze informatii despre echipe, piloti, vehicule, curse, bilete, circuite, arene si sponsori.

**Utilitatea modelului:**

* Organizarea curselor: Planificarea si gestionarea detaliilor despre fiecare cursa, inclusiv circuitul, data, ora si participantii.
* Gestionarea participantilor: Inregistrarea informatiilor despre echipe, piloti si vehiculele acestora.
* Vanzarea de bilete: Urmarirea vanzarilor de bilete, a locurilor disponibile si a preturilor.
* Gestionarea sponsorilor: Inregistrarea informatiilor despre sponsorii competitiei si a contractelor acestora.
* Analiza performantei: Generarea de rapoarte si statistici despre performanta echipelor si a pilotilor.

**Reguli de functionare:**

* Fiecare cursa are loc pe un circuit specific si intr-o arena.
* Fiecare echipa are unul sau mai multi piloti.
* Fiecare pilot conduce un vehicul specific.
* Fiecare cursa are un numar limitat de bilete disponibile.
* Sponsorii pot sponsoriza atat curse, cat si echipe.
* Fiecare vehicul are anumite caracteristici tehnice ce pot influenta performanta.
* Fiecare circuit are anumite caracteristici, cum ar fi lungimea, numarul de viraje, care influenteaza performanta vehiculelor.

**Exemple de utilizare:**

* Un administrator poate introduce date despre o noua cursa, inclusiv circuitul, data, ora si echipele participante.
* Un fan poate cumpara bilete online pentru o anumita cursa.
* Un sponsor poate vizualiza echipele pe care le sponsorizeaza.
* Un inginer poate analiza datele despre vehicule si performanta lor pe diferite circuite.
* Un fan isi poate vedea echipa favorita, pilotii care sunt in echipa si masinile lor.

**2. Prezentarea constrangerilor (restrictii, reguli) impuse asupra modelului.**

* Echipa poate avea mai multi piloti, dar un pilot poate apartine unei singure echipe.
* Un vehicul poate fi condus de un singur pilot, dar o echipa poate detine mai multe vehicule.
* Echipa poate fi sustinuta de mai multi sponsori, iar un sponsor poate sustine mai multe echipe.
* Un pilot poate participa la mai multe curse,iar la o cursa pot participa mai multi piloti.
* Un circuit este asociat cu o singura arena, iar o arena apartine unui singur circuit.
* Biletele sunt vandute pentru o singura cursa, iar o cursa poate avea mai multe bilete vandute.

**3. Descrierea entitatilor, incluzand precizarea cheii primare.**

|  |  |  |
| --- | --- | --- |
| **ENTITATE** | **CHEIE PRIMARA** | **OBSERVATII** |
| Echipa | ID\_Echipa | Informatii despre echipele din Formula 1 |
| Pilot | ID\_Pilot | Date personale despre pilot |
| Vehicul | ID\_Vehicul | Informatii cu privire la vehicul, producator, specificatii tehnice |
| Sponsor | ID\_Sponsor | Fiecare echipe este sustinuta de sponsori. |
| Cursa | ID\_Cursa | Informatii specifice pentru cursa |
| Bilet | ID\_Bilet | Biletele pot fi de mai multe feluri : Bronze, Silver, Gold, Exclusive, VIP |
| Circuit | ID\_Circuit | Pe circuitul se intrec pilotii pentru a castiga cursa |
| Arena | ID\_Arena | Arena este locul special amenajat pentru spectatori |

**4. Descrierea relatiilor, incluzand precizarea cardinalitatii acestora.**

|  |  |  |
| --- | --- | --- |
| **RELATIE** | **CARDINALITATE** | **OBSERVATII** |
| include | ECHIPA-PILOT  one-to-many | O echipa include mai multi piloti.  Un pilot face parte dintr-o singura echipa. |
| conduce | PILOT-VEHICUL  one-to-one | Un pilot conduce un singur vehicul.  Un vehicul este condus de un singur pilot. |
| participa | PILOT-CURSA  many-to-many | Un pilot participa la mai multe.  La o cursa pot participa mai multi piloti. |
| detine | ECHIPA-VEHICUL  one-to-many | O echipa detine mai multe vehicule.  Un vehicul este detinut de o singura echipa. |
| sustine | ECHIPA-SPONSOR  many-to-many | Un sponsor sustine mai multe echipe.  O echipa este sustinuta de mai multi sponsori. |
| vinde | CURSA-BILET  one-to-many | O cursa poate sa vanda mai multe bilete.  Un bilet este vandut la o singura cursa. |
| se desfasoara | CURSA-CIRCUIT  one-to-many | O cursa se desfasoara pe un sinfur circuit.  Pe un circuit se pot desfasura mai multe curse. |
| apartine | ARENA-CIRCUIT  one-to-one | Un circuit apartine unei arene.  Unei arene ii corespunde un singur circuit. |

**5. Descrierea atributelor, incluzand tipul de date si eventualele constangeri, valori implicite, valori posibile ale atributelor**

**ENTITATE: ECHIPA**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ATRIBUT** | **TIP** | **DIMENSIUNE/PRECIZIE** | **VALORI POSIBILE SI VALORI DEFAULT** | **OBSERVATII, OBLIGATORIU/OPTIONAL** |
| ID\_Echipa | Integer | - | - |  |
| Nume | String | - | - |  |
| Puncte | Integer | 0-100 | - |  |
| Sediu | String | - | - | Orasul in care are sediu echipa. |
| Nr\_Angajati | Integer | - | - |  |
| Buget | Integer | - | - | Bugetul fiecarei echipe este reprezentat in EURO |

**ENTITATE: PILOT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ATRIBUT** | **TIP** | **DIMENSIUNE/PRECIZIE** | **VALORI POSIBILE SI VALORI DEFAULT** | **OBSERVATII, OBLIGATORIU\OPTIONAL** |
| ID\_Pilot | Integer | - | - |  |
| Nume | String | - | - |  |
| Prenume | String | - | - |  |
| Data\_Nasterii | Date | - | - | Format : DD/MM/YEAR |
| Nationalitate | String | - | - |  |
| Salariu | Integer | - | - | Salariul fiecarui pilot este reprezentat in EURO |
| Titluri | Integer | - | - | Titluri = de cate ori a iesit pe locul #1 un pilot |

**ENTITATE: VEHICUL**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ATRIBUT** | **TIP** | **DIMENSIUNE/PRECIZIE** | **VALORI POSIBILE SI VALORI DEFAULT** | **OBSERVATII, OBLIGATORIU\OPTIONAL** |
| ID\_Vehicul | Integer | - | - |  |
| Greutate | Float | - | - | Greutatea este reprezentata in kilograme (KG) |
| Pret | Integer | - | - | Pretul este in EURO. |
| Pneuri | String | - | Soft/Medium/Hard | Pneurile depend de conditiile de rulare (Uscat, Umed{ploaie usoara}, Umed{ploaie}) |
| Putere | Integer | - | - | Puterea este reprezentata in Cai Putere (CP) |
| Model | String | - | - |  |

**ENTITATE: CURSA**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ATRIBUT** | **TIP** | **DIMENSIUNE/PRECIZIE** | **VALORI POSIBILE SI VALORI DEFAULT** | **OBSERVATII, OBLIGATORIU\OPTIONAL** |
| ID\_Cursa | Integer | - | - |  |
| Data | Date | - | - | FORMAT : DD/MM/YEAR |
| Durata | Time | - | - |  |
| Record | Time | - | - | Timpul cel mai rapid inregistrat de un pilot in timpul unei curse |
| Tip\_Cursa | String | - | Grand Prix/Sprint Race/Test Session |  |
| Vreme | String | - | - |  |

**ENTITATE: CIRCUIT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ATRIBUT** | **TIP** | **DIMENSIUNE/PRECIZIE** | **VALORI POSIBILE SI VALORI DEFAULT** | **OBSERVATII, OBLIGATORIU\OPTIONAL** |
| ID\_Circuit | Integer | - | - |  |
| Nume | String | - | - |  |
| Locatie | String | - | - |  |
| Lungime | Float | - | - | Lungimea este masurata in KM |
| Viraje | Integer | - | - | Numarul de viraje pe care le are un circuit |
| DRS\_Zone | Integer | - | - |  |

**ENTITATE: ARENA**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ATRIBUT** | **TIP** | **DIMENSIUNE/PRECIZIE** | **VALORI POSIBILE SI VALORI DEFAULT** | **OBSERVATII, OBLIGATORIU\OPTIONAL** |
| ID\_Arena | Integer | - | - |  |
| Capacitate | Integer | - | - | Numarul de locuri din arena |
| Tribune | Integer | - | - |  |
| Parcare | String | - | DA/NU | Daca spectatatorii isi pot parca masina |
| Restaurante | Integer | - | - | Numarul de restaurant din arena |
| Zona\_Acoperita | String | - | DA/NU |  |

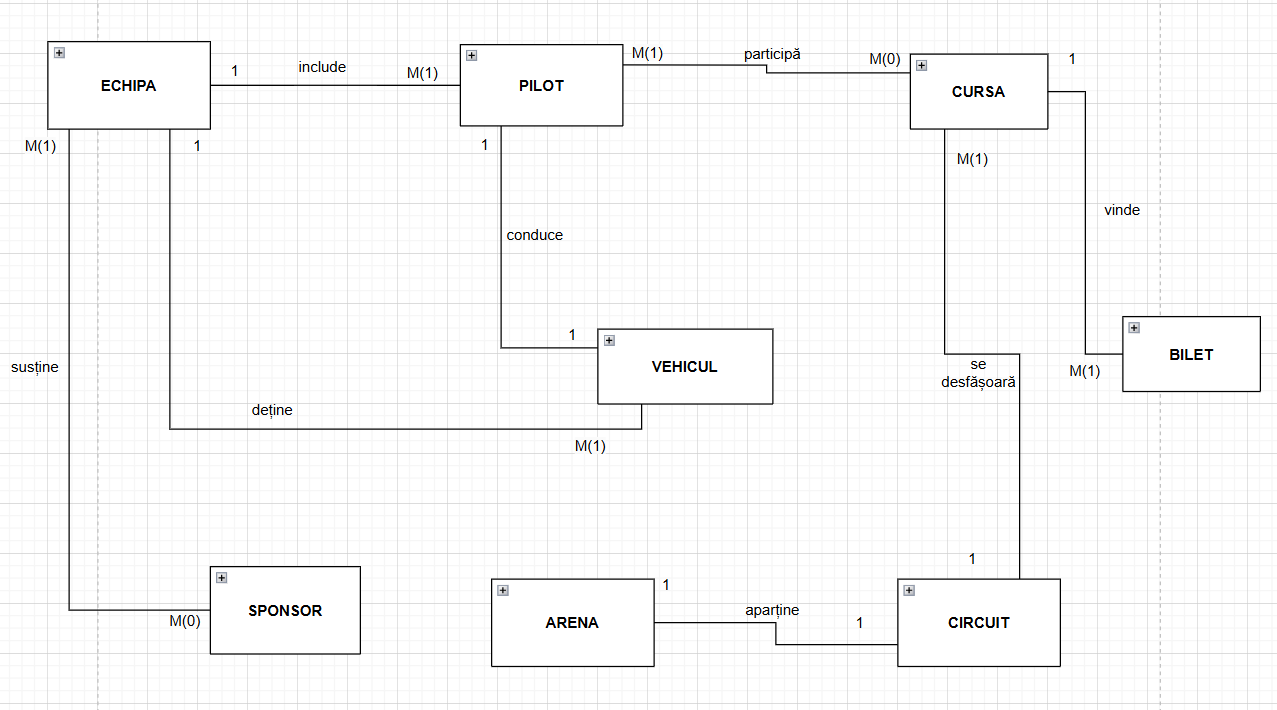
**ENTITATE: SPONSORI**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ATRIBUT** | **TIP** | **DIMENSIUNE/PRECIZIE** | **VALORI POSIBILE SI VALORI DEFAULT** | **OBSERVATII, OBLIGATORIU\OPTIONAL** |
| ID\_Sponsor | Integer | - | - |  |
| Nume | String | - | - |  |
| Contributie\_Financiara | Integre | - | - | Contributia sponsorului la o echipa exprimata in EURO |
| Tip\_Parteneriat | String | - | Strategic/Marketing/Financiar/Excusiv |  |

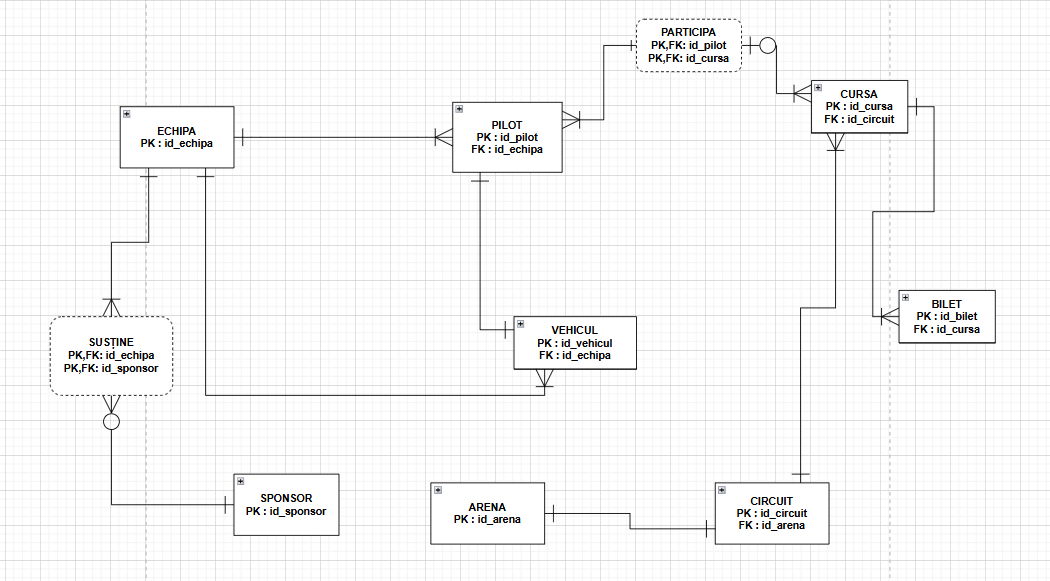
**ENTITATE: BILET**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ATRIBUT** | **TIP** | **DIMENSIUNE/PRECIZIE** | **VALORI POSIBILE SI VALORI DEFAULT** | **OBSERVATII, OBLIGATORIU\OPTIONAL** |
| ID\_Bilet | Integer | - | - |  |
| Pret | Float | - | - | Pretul unui bilet este in EURO |
| Tip | String | - | - | Bronze, Silver, Gold, Exclusive, VIP |
| Loc\_Tribuna | VarChar | - | - | Ex. : A, B, C |
| Loc\_Rand | Integer |  |  | Ex. : 12, 34, 23 |
| Loc\_Numar | Integer |  |  | Ex. : 10, 8, 30 |
| Stare | String | - | - |  |

**6. Realizarea diagramei etitate-relatie corespunzatoare descrierii de la punctele 3-5.**



**7. Realizarea diagramei conceptual corespunzatoare diagramei entitate-relatie proiectate la punctul 6.**



**8. Enumerarea schemelor relaționale corespunzătoare diagramei conceptuale proiectate la punctul 7.**

* **VEHICUL**(#id\_vehicul, greutate, pret, pneuri, putere, model #id\_model)
* **BILET**(#id\_bilet, tip\_bilet, loc\_tribuna, loc\_rand, loc\_numar, stare, #id\_cursa)
* **SPONSOR**(#id\_sponsor, nume, contributie\_financiara, tip\_parteneriat)
* **ECHIPA**(#id\_echipa, nume, puncte, sediu, nr\_angajati, buget)
* **PILOT**(#id\_pilot, nume, prenume, data\_nasterii, nationalitate, salariu, titluri, #id\_echipa)
* **ARENA**(#id\_arena, capacitate, tribune, parcare, restaurante, zona\_acoperita)
* **CIRCUIT**(#id\_circuit, nume, locatie, lungime, viraje, drs\_zone, #id\_arena)
* **CURSA**(#id\_cursa, data, durata, record, tip\_cursa, vreme, #id\_circuit)
* **SUSTINE**(#id\_echipa, #id\_sponsor)
* **PARTICIPA**(#id\_pilot, #id\_cursa)

## 9. Realizarea normalizarii pana la forma normal 3 (FN1-FN3)

## 1. VEHICUL(ID\_Vehicul, Greutate, Pret, Pneuri, Putere, Model)

F = { ID\_Vehicul → (Greutate, Pret, Pneuri, Model), Model → Putere }

Forma actuala:

* Este in FN1: toate atributele sunt atomice
* Este in FN2: toate atributele depind de cheia primara
* Nu este in FN3: avem dependenta tranzitiva: ID\_Vehicul → Model → Putere

Descompunere FN3:

* VEHICUL(ID\_Vehicul, Greutate, Pret, Pneuri, Model)
* MODEL(ID\_Model, Putere)

## 2. SPONSOR(ID\_Sponsor, Nume, Contributie\_Financiara, Tip\_Parteneriat)

F = { ID\_Sponsor → (Nume, Tip\_Parteneriat, Contributie\_Financiara) }

### Forma actuală:

* Este in **FN1**: toate atributele sunt atomice
* Este in **FN2**: toate atributele depind de cheia primara
* Este in **FN3**: nu avem dependente tranzitive

## Nu necesita descompunere.

## 3. ARENA(ID\_Arena, Capacitate, Tribune, Restaurante, Zona\_Acoperita, Parcare)

F = { ID\_Arena → (Capacitate, Tribune, Restaurante, Zona\_Acoperita, Parcare)}

### Forma actuală:

* Este in **FN1**: toate atributele sunt atomice
* Este in **FN2**: toate atributele depind de cheia primara
* Este in **FN3**: nu avem dependente tranzitive

### Nu necesita descompunere.

## 4. BILET(ID\_Bilet, Pret, Tip, Loc\_Tribuna, Loc\_Rand, Loc\_Numar, Stare)

F = { ID\_Bilet → (Tip, Loc\_Tribuna, Loc\_Rand, Loc\_Numar, Stare), Tip → Pret }

### Forma actuala:

* Este in **FN1**: toate atributele sunt atomice
* Este in **FN2**: toate atributele depind de cheia primara
* Nu este in **FN3**: avem dependenta tranzitiva: ID\_Bilet → Tip → Pret

### Descompunere FN3:

* BILET(ID\_Bilet, Tip, Loc\_Tribuna, Loc\_Rand, Loc\_Numar, Stare)
* TIP\_BILET(ID\_Tip\_Bilet, Pret)

## 5. ECHIPA(ID\_Echipa, Nume, Puncte, Sediu, Nr\_Angajati, Buget)

F = { ID\_Echipa → (Nume, Puncte, Sediu, Nr\_Angajati, Buget) }

### Forma actuala:

* Este in **FN1**: toate atributele sunt atomice
* Este in **FN2**: toate atributele depind de cheia primara
* Este in **FN3**: nu avem dependente tranzitive

### Nu necesita descompunere

## 6. PILOT(ID\_Pilot, Nume, Prenume, Data\_Nasterii, Nationalitate, Salariu, Titluri)

F = { ID\_Pilot → Nume, Prenume, Data\_Nasterii, Nationalitate, Salariu, Titluri }

### Forma actuala:

* Este in **FN1**: toate atributele sunt atomice
* Este in **FN2**: toate atributele depind de cheia primara
* Este in **FN3**: nu avem dependente tranzitive

### Nu necesita descompunere

## 7. CURSA(ID\_Cursa, Data, Durata, Record, Tip\_Cursa, Vreme)

F = { ID\_Cursa → (Data, Durata, Record, Tip\_Cursa, Vreme) }

### Forma actuală:

* Este in **FN1**: toate atributele sunt atomice
* Este in **FN2**: toate atributele depind de cheia primara
* Este in **FN3**: nu avem dependente tranzitive

### Nu necesita descompunere

## 8. CIRCUIT(ID\_Circuit, Nume, Locatie, Lungime, Viraje, DRS\_Zone)

F = { ID\_Circuit → (Nume, Locatie, Lungime, Viraje, DRS\_Zone) }

### Forma actuala:

* Este in **FN1**: toate atributele sunt atomice
* Este in **FN2**: toate atributele depind de cheia primara
* Este in **FN3**: nu avem dependente tranzitive

### Nu necesita descompunere

### DIAGRAMA DUPA DESCOMPUNERE :

### 

### 10. Crearea unei secvente ce va fi utilizata in inserarea inregistrarilor in tabele (punctul 11)

### Am definit cinci secvente pentru a genera automat valori unice pentru cheile primare in anumite tabele:

### tip\_bilet\_seq → Folosita pentru tabela TIP\_BILET

### pilot\_seq → Folosita pentru tabela PILOT

### echipa\_seq → Folosita pentru tabela ECHIPA

### circuit\_seq → Folosita pentru tabela CIRCUIT

### arena\_seq → Folosita pentru tabela ARENA

### Aceste secvente asigura ca fiecare inregistrare noua primeste un identificator unic (de obicei incremental: 1, 2, 3...), fara a fi necesara specificarea manuala a valorii.

### COD SQL :

CREATE SEQUENCE tip\_bilet\_seq;

CREATE SEQUENCE pilot\_seq;

CREATE SEQUENCE echipa\_seq;

CREATE SEQUENCE circuit\_seq;

CREATE SEQUENCE arena\_seq;

### 

### Script Output :

### 

### Pentru celelalte tabele, am folosit GENERATED BY DEFAULT AS IDENTITY pe coloanele cheilor primare. Aceasta abordare ofera flexibilitate :

### Genereaza automat valori daca nu se specifica manual

### Permite suprascrierea (daca se doreste introducerea manuala a unui ID)

### COD SQL :

### 

### 

### 11. Crearea tabelelor SQL si inserarea de date coerente in fiecare dintre acestea (minimum 5 inregistrari in fiecare tabel neasociativ; minim 10 inregistrari in tabelele associative; maxim 30 de inregistrari in fiecare tabel).

### Tabelul ARENA\_F1

CREATE TABLE ARENA\_F1 (

id\_arena INT PRIMARY KEY,

capacitate INT,

tribune INT,

parcare VARCHAR(3),

restaurante INT,

zona\_acoperita VARCHAR(3)

);

CREATE SEQUENCE arena\_seq

START WITH 1 INCREMENT BY 1;

INSERT INTO ARENA\_F1 VALUES (arena\_seq.NEXTVAL, 100000, 50, 'DA', 10, 'DA');

INSERT INTO ARENA\_F1 VALUES (arena\_seq.NEXTVAL, 80000, 40, 'NU', 8, 'DA');

INSERT INTO ARENA\_F1 VALUES (arena\_seq.NEXTVAL, 90000, 45, 'DA', 12, 'NU');

INSERT INTO ARENA\_F1 VALUES (arena\_seq.NEXTVAL, 70000, 35, 'NU', 7, 'DA');

INSERT INTO ARENA\_F1 VALUES (arena\_seq.NEXTVAL, 85000, 48, 'DA', 9, 'DA');

INSERT INTO ARENA\_F1 VALUES (arena\_seq.NEXTVAL, 75000, 38, 'DA', 11, 'NU');

INSERT INTO ARENA\_F1 VALUES (arena\_seq.NEXTVAL, 65000, 30, 'NU', 5, 'DA');

INSERT INTO ARENA\_F1 VALUES (arena\_seq.NEXTVAL, 78000, 42, 'DA', 10, 'DA');

INSERT INTO ARENA\_F1 VALUES (arena\_seq.NEXTVAL, 82000, 44, 'NU', 6, 'NU');

INSERT INTO ARENA\_F1 VALUES (arena\_seq.NEXTVAL, 90000, 50, 'DA', 13, 'DA');

### 

### 

### 

### 

### Tabelul CIRCUIT\_F1

CREATE TABLE CIRCUIT\_F1 (

id\_circuit INT PRIMARY KEY,

nume VARCHAR(100),

locatie VARCHAR(100),

lungime FLOAT,

viraje INT,

drs\_zone INT,

id\_arena INT,

FOREIGN KEY (id\_arena) REFERENCES ARENA\_F1(id\_arena)

);

CREATE SEQUENCE circuit\_seq

START WITH 1 INCREMENT BY 1;

INSERT INTO CIRCUIT\_F1 VALUES (circuit\_seq.NEXTVAL, 'Silverstone', 'UK', 5.891, 18, 2, 1);

INSERT INTO CIRCUIT\_F1 VALUES (circuit\_seq.NEXTVAL, 'Monza', 'Italy', 5.793, 11, 1, 2);

INSERT INTO CIRCUIT\_F1 VALUES (circuit\_seq.NEXTVAL, 'Spa-Francorchamps', 'Belgium', 7.004, 19, 2, 3);

INSERT INTO CIRCUIT\_F1 VALUES (circuit\_seq.NEXTVAL, 'Suzuka', 'Japan', 5.807, 18, 2, 4);

INSERT INTO CIRCUIT\_F1 VALUES (circuit\_seq.NEXTVAL, 'Monaco', 'Monaco', 3.337, 19, 1, 5);

INSERT INTO CIRCUIT\_F1 VALUES (circuit\_seq.NEXTVAL, 'Barcelona', 'Spain', 4.655, 16, 1, 6);

INSERT INTO CIRCUIT\_F1 VALUES (circuit\_seq.NEXTVAL, 'Melbourne', 'Australia', 5.303, 16, 1, 7);

INSERT INTO CIRCUIT\_F1 VALUES (circuit\_seq.NEXTVAL, 'Interlagos', 'Brazil', 4.309, 15, 1, 8);

INSERT INTO CIRCUIT\_F1 VALUES (circuit\_seq.NEXTVAL, 'Silverstone Short', 'UK', 3.660, 12, 0, 9);

INSERT INTO CIRCUIT\_F1 VALUES (circuit\_seq.NEXTVAL, 'Austin', 'USA', 5.513, 20, 2, 10);

### 

### 

### 

### 

### Tabelul MODEL\_F1

CREATE TABLE MODEL\_F1 (

id\_model INT PRIMARY KEY,

putere INT

);

INSERT INTO MODEL\_F1 VALUES

(1, 950),

(2, 900),

(3, 980),

(4, 970),

(5, 960),

(6, 940),

(7, 930),

(8, 920),

(9, 910),

(10, 900);

### 

### 

### 

### 

### Tabelul ECHIPA\_F1

CREATE TABLE ECHIPA\_F1 (

id\_echipa INT PRIMARY KEY,

nume VARCHAR(100),

puncte INT,

sediu VARCHAR(100),

nr\_angajati INT,

buget INT

);

CREATE SEQUENCE echipa\_seq

START WITH 1 INCREMENT BY 1;

INSERT INTO ECHIPA\_F1 VALUES (echipa\_seq.NEXTVAL, 'Mercedes', 450, 'UK', 200, 250000000);

INSERT INTO ECHIPA\_F1 VALUES (echipa\_seq.NEXTVAL, 'Red Bull', 420, 'Austria', 180, 230000000);

INSERT INTO ECHIPA\_F1 VALUES (echipa\_seq.NEXTVAL, 'Ferrari', 430, 'Italy', 190, 240000000);

INSERT INTO ECHIPA\_F1 VALUES (echipa\_seq.NEXTVAL, 'McLaren', 390, 'UK', 170, 200000000);

INSERT INTO ECHIPA\_F1 VALUES (echipa\_seq.NEXTVAL, 'Alpine', 350, 'France', 160, 180000000);

### 

### 

### 

### 

### Tabelul VEHICUL\_F1

CREATE TABLE VEHICUL\_F1 (

id\_vehicul INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

greutate FLOAT,

pret INT,

pneuri VARCHAR(10),

id\_model INT,

id\_echipa INT,

FOREIGN KEY (id\_model) REFERENCES MODEL\_F1(id\_model),

FOREIGN KEY (id\_echipa) REFERENCES ECHIPA\_F1(id\_echipa)

);

INSERT INTO VEHICUL\_F1 (greutate, pret, pneuri, id\_model, id\_echipa) VALUES

(752.5, 1500000, 'Soft', 1, 1),

(755.0, 1400000, 'Medium', 2, 1),

(753.0, 1480000, 'Hard', 3, 2),

(756.0, 1490000, 'Soft', 4, 2),

(750.0, 1510000, 'Medium', 5, 3),

(754.0, 1470000, 'Hard', 6, 3),

(749.0, 1460000, 'Soft', 7, 4),

(751.0, 1450000, 'Medium', 8, 4),

(753.5, 1440000, 'Hard', 9, 5),

(755.5, 1430000, 'Soft', 10, 5);

### 

### 

### 

### 

### Tabelul TIP\_BILET\_F1

CREATE TABLE TIP\_BILET\_F1 (

id\_tip\_bilet INT PRIMARY KEY,

tip VARCHAR(20),

pret FLOAT

);

CREATE SEQUENCE tip\_bilet\_seq

START WITH 1 INCREMENT BY 1;

INSERT INTO TIP\_BILET\_F1 VALUES (tip\_bilet\_seq.NEXTVAL, 'VIP', 300.0);

INSERT INTO TIP\_BILET\_F1 VALUES (tip\_bilet\_seq.NEXTVAL, 'Exclusive', 200.0);

INSERT INTO TIP\_BILET\_F1 VALUES (tip\_bilet\_seq.NEXTVAL, 'Gold', 150.0);

INSERT INTO TIP\_BILET\_F1 VALUES (tip\_bilet\_seq.NEXTVAL, 'Silver', 100.0);

INSERT INTO TIP\_BILET\_F1 VALUES (tip\_bilet\_seq.NEXTVAL, 'Bronze', 50.0);

### 

### 

### 

### 

### Tabelul CURSA\_F1

CREATE TABLE CURSA\_F1 (

id\_cursa INT PRIMARY KEY,

data DATE,

durata VARCHAR(8),

record VARCHAR(9),

tip\_cursa VARCHAR(50),

vreme VARCHAR(50),

id\_circuit INT,

FOREIGN KEY (id\_circuit) REFERENCES CIRCUIT\_F1(id\_circuit)

);

INSERT INTO CURSA\_F1 VALUES

(1, DATE '2025-06-15', '01:30:00', '1:25.345', 'Grand Prix', 'Soare', 1),

(2, DATE '2025-07-20', '01:40:00', '1:28.234', 'Sprint', 'Nori', 2),

(3, DATE '2025-08-10', '01:35:00', '1:26.789', 'Grand Prix', 'Soare', 3),

(4, DATE '2025-09-05', '01:50:00', '1:29.101', 'Grand Prix', 'Ploaie', 4),

(5, DATE '2025-10-12', '01:45:00', '1:27.456', 'Sprint', 'Nori', 5),

(6, DATE '2025-11-18', '01:38:00', '1:28.789', 'Grand Prix', 'Soare', 6),

(7, DATE '2025-12-08', '01:42:00', '1:30.123', 'Grand Prix', 'Ceata', 7),

(8, DATE '2026-01-15', '01:36:00', '1:27.890', 'Sprint', 'Soare', 8),

(9, DATE '2026-02-20', '01:34:00', '1:26.345', 'Grand Prix', 'Nori', 9),

(10, DATE '2026-03-25', '01:39:00', '1:29.567', 'Sprint', 'Ploaie', 10);

### 

### 

### 

### 

### Tabelul BILET\_F1

CREATE TABLE BILET\_F1 (

id\_bilet INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

id\_tip\_bilet INT,

loc\_tribuna VARCHAR(2),

loc\_rand INT,

loc\_numar INT,

stare VARCHAR(20),

id\_cursa INT,

FOREIGN KEY (id\_tip\_bilet) REFERENCES TIP\_BILET\_F1(id\_tip\_bilet),

FOREIGN KEY (id\_cursa) REFERENCES CURSA\_F1(id\_cursa)

);

INSERT INTO BILET\_F1 (id\_tip\_bilet, loc\_tribuna, loc\_rand, loc\_numar, stare, id\_cursa) VALUES

(1, 'A1', 1, 1, 'VANDUT', 1),

(2, 'A2', 1, 2, 'LIBER', 1),

(3, 'A3', 1, 3, 'REZERVAT', 2),

(4, 'B1', 2, 1, 'VANDUT', 2),

(5, 'B2', 2, 2, 'LIBER', 3),

(1, 'B3', 2, 3, 'REZERVAT', 3),

(2, 'C1', 3, 1, 'VANDUT', 4),

(3, 'C2', 3, 2, 'LIBER', 4),

(4, 'C3', 3, 3, 'REZERVAT', 5),

(5, 'D1', 4, 1, 'VANDUT', 5);

### 

### 

### https://cdn.discordapp.com/attachments/734164453766987786/1374518856462110911/image.png?ex=682e57f0&is=682d0670&hm=d7e0c318290b20d51ef8b0d34c977c771e2ce6c63bf164c6228b7b26dbb6faaa&

### 

### Tabelul SPONSOR\_F1

CREATE TABLE SPONSOR\_F1 (

id\_sponsor INT PRIMARY KEY,

nume VARCHAR(100),

contributie\_financiara INT,

tip\_parteneriat VARCHAR(50)

);

INSERT INTO SPONSOR\_F1 VALUES

(1, 'Petronas', 5000000, 'Marketing'),

(2, 'Pirelli', 3000000, 'Strategic'),

(3, 'Red Bull GmbH', 8000000, 'Financiar'),

(4, 'Shell', 4500000, 'Exclusiv'),

(5, 'Mobil 1', 4000000, 'Marketing'),

(6, 'Tag Heuer', 2000000, 'Strategic'),

(7, 'Monster Energy', 3500000, 'Financiar'),

(8, 'Dell', 2800000, 'Exclusiv'),

(9, 'Honda', 6000000, 'Marketing'),

(10, 'Rolex', 5000000, 'Strategic');

### 

### 

### https://cdn.discordapp.com/attachments/734164453766987786/1374518921482342551/image.png?ex=682e5800&is=682d0680&hm=ec05a96f478000fdbe594ba0ad1e6d297c4533cacd5efd233ec0711eb47a98b7&

### 

### Tabelul PILOT\_F1

CREATE TABLE PILOT\_F1 (

id\_pilot INT PRIMARY KEY,

nume VARCHAR(50),

prenume VARCHAR(50),

data\_nastere DATE,

nationalitate VARCHAR(50),

salariu INT,

titluri INT,

id\_echipa INT,

FOREIGN KEY (id\_echipa) REFERENCES ECHIPA\_F1(id\_echipa)

);

CREATE SEQUENCE pilot\_seq

START WITH 1 INCREMENT BY 1;

INSERT INTO PILOT\_F1 VALUES (pilot\_seq.NEXTVAL, 'Hamilton', 'Lewis', '07-01-1985', 'British', 50000000, 7, 1);

INSERT INTO PILOT\_F1 VALUES (pilot\_seq.NEXTVAL, 'Russell', 'George', '15-02-1998', 'British', 15000000, 0, 1);

INSERT INTO PILOT\_F1 VALUES (pilot\_seq.NEXTVAL, 'Verstappen', 'Max', '30-09-1997', 'Dutch', 45000000, 3, 2);

INSERT INTO PILOT\_F1 VALUES (pilot\_seq.NEXTVAL, 'Perez', 'Sergio', '26-01-1990', 'Mexican', 18000000, 0, 2);

INSERT INTO PILOT\_F1 VALUES (pilot\_seq.NEXTVAL, 'Leclerc', 'Charles', '16-10-1997', 'Monegasque', 30000000, 0, 3);

INSERT INTO PILOT\_F1 VALUES (pilot\_seq.NEXTVAL, 'Sainz', 'Carlos', '01-09-1994', 'Spanish', 22000000, 0, 3);

INSERT INTO PILOT\_F1 VALUES (pilot\_seq.NEXTVAL, 'Norris', 'Lando', '13-11-1999', 'British', 14000000, 0, 4);

INSERT INTO PILOT\_F1 VALUES (pilot\_seq.NEXTVAL, 'Piastri', 'Oscar', '06-04-2001', 'Australian', 10000000, 0, 4);

INSERT INTO PILOT\_F1 VALUES (pilot\_seq.NEXTVAL, 'Alonso', 'Fernando', '29-07-1981', 'Spanish', 25000000, 2, 5);

INSERT INTO PILOT\_F1 VALUES (pilot\_seq.NEXTVAL, 'Ocon', 'Esteban', '17-09-1996', 'French', 15000000, 0, 5);

### 

### 

### https://cdn.discordapp.com/attachments/734164453766987786/1374519097533923439/image.png?ex=682e582a&is=682d06aa&hm=cc3c45da3d8a3eb019ec4ae685eed7005a0d006843026e9be39075bf5b4027b4&

### 

### Tabelul PILOT\_CURSA\_F1

CREATE TABLE PILOT\_CURSA\_F1 (

id\_pilot INT NOT NULL,

id\_cursa INT NOT NULL,

PRIMARY KEY (id\_pilot, id\_cursa),

FOREIGN KEY (id\_pilot) REFERENCES PILOT\_F1(id\_pilot),

FOREIGN KEY (id\_cursa) REFERENCES CURSA\_F1(id\_cursa)

);

INSERT INTO PILOT\_CURSA\_F1 VALUES

(1, 1),(1, 2),(2, 1),(2, 3),(3, 2),(3, 4),(4, 3),(4, 5),(5, 4),(5, 6),

(6, 5),(6, 7),(7, 6),(7, 8),(8, 7),(8, 9),(9, 8),(9, 10),(10, 9),(10, 10);

### 

### 

### https://cdn.discordapp.com/attachments/734164453766987786/1374519151506231316/image.png?ex=682e5837&is=682d06b7&hm=a71bc2f75d91bd8a12a1809356ea5ba4a90c1cd16518eda4be01a5c5a7225aa6&

### 

### Tabelul ECHIPA\_SPONSOR\_F1

CREATE TABLE ECHIPA\_SPONSOR\_F1 (

id\_echipa INT,

id\_sponsor INT,

PRIMARY KEY (id\_echipa, id\_sponsor),

FOREIGN KEY (id\_echipa) REFERENCES ECHIPA\_F1(id\_echipa),

FOREIGN KEY (id\_sponsor) REFERENCES SPONSOR\_F1(id\_sponsor)

);

INSERT INTO ECHIPA\_SPONSOR\_F1 VALUES

(1, 1),(1, 2),(2, 3),(2, 4),(3, 5),(3, 6),(4, 7),(4, 8),(5, 9),(5, 10),

(1, 3),(2, 5),(3, 7),(4, 9),(5, 1),(1, 4),(2, 6),(3, 8),(4, 10),(5, 2);

### 

### 

### https://cdn.discordapp.com/attachments/734164453766987786/1374519244615843930/image.png?ex=682e584d&is=682d06cd&hm=30c5d876dafec2d71eec726510868bc664c8fde0a80b2bfe9ad7f46c16351e58&

### 

### 12. Formulati in limbaj natural si implementati 5 cereri SQL complexe ce vor utiliza, in ansamblul lor, urmatoarele elemente :

### a) subcereri sincronizate in care intervin cel putin 3 tabele

### b) subcereri nesincronizate in clauza FROM

### c) grupari de date, functii grup, filtrare la nivel de grupuri cu subcereri nesincronizate (in clauza de HAVING)

### d) ordonari si utilizarea functiilor NVL si DECODE (in cadrul aceleiasi cereri)

### e) utilizarea a cel putin 2 functii pe siruri de caractere, 2 functii pe date calendaristice, a cel putin unei expresii CASE

### f) utilizarea a cel putin 1 bloc de cerere (clauza WITH)

### Observatie : Intr-o cerere se vor regasi mai multe elemente dinre cele enumerate mai sus, astfel incat cele 5 cereri sa le cuprinda pe toate.

### c) + d)

### Sa se afiseze echipele care au un numar de angajati mai mare decat media echipelor si marcheaza daca bugetul este “mare” (>22.000.000), “mediu” sau mic folosind DECODE, afisand si numarul de angajati (daca e NULL, afiseaza 0).

SELECT E.nume,

NVL(E.nr\_angajati, 0) AS nr\_angajati,

DECODE(

SIGN(E.buget - 22000000),

1, 'mare',

0, 'mediu',

-1, 'mic'

) AS categorie\_buget

FROM ECHIPA\_F1 E

GROUP BY E.nume, E.nr\_angajati, E.buget

HAVING E.nr\_angajati>(

SELECT AVG(nr\_angajati) FROM ECHIPA\_F1

);

### 

### Rezultat Query :

### 

### e)

### Sa se afiseze pentru fiecare pilot numele complet, initialele numelui, varsta calculate in ani intregi si o

### clasificare simplificata in functie de numarul de titluri obtinute.

SELECT

P.nume || ' ' || P.prenume AS nume\_complet,

SUBSTR(P.nume, 1, 1) || '.' || SUBSTR(P.prenume, 1, 1) AS initiale,

TRUNC(MONTHS\_BETWEEN(SYSDATE, P.data\_nastere)/12) AS varsta,

P.titluri,

CASE

WHEN P.titluri >= 3 THEN 'Campion Experimentat'

WHEN P.titluri BETWEEN 1 AND 2 THEN 'Pilot Promitator'

ELSE 'Novice'

END AS clasificare

FROM PILOT\_F1 P;

### 

### Rezultat Query :

### 

### f)

### Sa se afiseze echipele care au primit sponsorizari totale mai mari de 20.000.000.

WITH sponsorizari AS (

SELECT ES.id\_echipa, SUM(S.contributie\_financiara) AS total

FROM ECHIPA\_SPONSOR\_F1 ES

JOIN SPONSOR\_F1 S ON ES.id\_sponsor=S.id\_sponsor

GROUP BY ES.id\_echipa

)

SELECT E.nume AS echipa,

S.total AS sponsorizare\_totala

FROM sponsorizari S

JOIN ECHIPA\_F1 E ON E.id\_echipa=S.id\_echipa

WHERE S.total>20000000;

### 

### Rezultat Query :

### 

### a)

### Sa se afiseze numele pilotului si nationalitatea sa, doar daca a participat la o cursa desfasurata pe un circuit care

### are mai mult mult de 15 viraje si o capacitate de public mai mare decat cea a arenei associate circuitului “Monaco”.

SELECT P.nume || ' ' || P.prenume AS pilot,

P.nationalitate

FROM PILOT\_F1 P

WHERE EXISTS (

SELECT 1

FROM PILOT\_CURSA\_F1 PC

JOIN CURSA\_F1 C ON PC.id\_cursa=C.id\_cursa

JOIN CIRCUIT\_F1 CI ON C.id\_circuit=CI.id\_circuit

JOIN ARENA\_F1 A ON CI.id\_arena=A.id\_arena

WHERE PC.id\_pilot=P.id\_pilot

AND CI.viraje > 15

AND A.capacitate > (

SELECT A2.capacitate

FROM CIRCUIT\_F1 CI2

JOIN ARENA\_F1 A2 ON CI2.id\_arena=A2.id\_arena

WHERE CI2.nume='Monaco'

)

);

### 

### Rezultat Query :

### 

### b)

### Sa se afiseze pentru fiecare pilot numele complet, salariul numele echipei din care face parte si media salariilor tuturor pilotilor.

SELECT

P.nume || ' ' || P.prenume AS nume\_complet,

P.salariu,

E.nume AS nume\_echipa,

avg\_salariu.medie\_salariu\_piloti

FROM PILOT\_F1 P,

ECHIPA\_F1 E,

(SELECT AVG(salariu) AS medie\_salariu\_piloti FROM PILOT\_F1) avg\_salariu

WHERE P.id\_echipa=E.id\_echipa

ORDER BY P.nume;

### 

### Rezultat Query :

### 

### 13. Implementarea a 3 operatii de actualizare si de suprimare a datelor utilizand subcereri.

### 1. Sa se actualizeze salariul pilotilor care au salariu mai mic decat media salariilor din echipa lor, marindu-le la media respectiva

UPDATE PILOT\_F1 P

SET salariu = (

SELECT AVG(salariu)

FROM PILOT\_F1

WHERE id\_echipa=P.id\_echipa

)

WHERE salariu<(

SELECT AVG(salariu)

FROM PILOT\_F1

WHERE id\_echipa=P.id\_echipa

);

### 

### 

### 2. Sa se marcheze toate biletele tip “Silver” la curse desfasurate pe un circuit din ‘Monaco’ ca fiind ‘VANDUT’.

UPDATE BILET\_F1 B

SET stare = 'VANDUT'

WHERE id\_tip\_bilet = (

SELECT id\_tip\_bilet

FROM TIP\_BILET\_F1

WHERE tip='Silver'

)

AND id\_cursa IN (

SELECT id\_cursa

FROM CURSA\_F1

WHERE id\_circuit IN (

SELECT id\_circuit

FROM CIRCUIT\_F1

WHERE locatie='Monaco'

)

);

### 

### 

### 3. Sa se stearga vehiculele care au un model cu puteres sub medie si apartin echipelor cu buget mai mic decat media, dar numai daca pretul vehiculului este mai mare de 1.000.000

DELETE FROM VEHICUL\_F1 V

WHERE V.id\_model IN (

SELECT M.id\_model

FROM MODEL\_F1 M

WHERE M.putere<(SELECT AVG(putere) FROM MODEL\_F1)

)

AND V.id\_echipa IN (

SELECT E.id\_echipa

FROM ECHIPA\_F1 E

WHERE E.buget<(SELECT AVG(buget) FROM ECHIPA\_F1)

)

AND V.pret>1000000;

### 

### 

### 14. Crearea unei vizualizari complexe. Dati exemplu de operatie LMD permisa pe vizualizarea respectiva si un exemplu de operatie LMD nepermisa.

### 

### 

### Rezultat query :

SELECT \* FROM V\_PILOTI\_ECHIPA\_STAT

### 

### 

### Exemplu de operatie LMD permisa (interogare SELECT) :

SELECT \* FROM V\_PILOTI\_ECHIPA\_STAT

WHERE comparative = ‘sub medie’;

### 

### Exemplu de operatie LMD nepermisa (UPDATE sau INSERT) :

UPDATE V\_PILOTI\_ECHIPA\_STAT

SET salariu\_mediu\_echipa = 30000000

WHERE nume\_pilot = 'Carlos Sainz';

### 

### 

### Nu este permisa deoarece :

### view-ul contine functii aggregate (AVG)

### are GROUP BY si CASE

### 15. Formulati in limbaj natural si implementati in SQL : o cerere ce utilizeaza operatie outer-join pe minim 4 tabele, o cerere ce utilizeaza operatia division si o cerere care implementeaza analiza top-n.

### Sa se afiseze pilotii care au participat la toate cursele din Italia.

SELECT P.id\_pilot, P.nume || ' ' || P.prenume as nume\_pilot

FROM PILOT\_F1 P

WHERE NOT EXISTS (

SELECT C.id\_cursa

FROM CURSA\_F1 C

JOIN CIRCUIT\_F1 CI ON C.id\_circuit = CI.id\_circuit

WHERE CI.locatie = 'Italy'

AND NOT EXISTS (

SELECT 1

FROM PILOT\_CURSA\_F1 PC

WHERE PC.id\_pilot = P.id\_pilot

AND PC.id\_cursa = C.id\_cursa

)

);

### 

### Rezultat Query :

### 

### Sa se afiseze TOP 5 piloti cu cele mai multe curse Grand Prix.

SELECT ROWNUM AS pozitie,

pilot,

curse\_grand\_prix

FROM (

SELECT

P.nume || ' ' || P.prenume AS pilot,

COUNT(\*) AS curse\_grand\_prix

FROM PILOT\_F1 P

JOIN PILOT\_CURSA\_F1 PC ON P.id\_pilot = PC.id\_pilot

JOIN CURSA\_F1 C ON PC.id\_cursa = C.id\_cursa

WHERE C.tip\_cursa = 'Grand Prix'

GROUP BY P.nume, P.prenume

ORDER BY curse\_grand\_prix DESC

)

WHERE ROWNUM <= 5;

### 

### Rezultat Query :

### 

### Afiseaza toti pilotii, impreuna cu : numele echipei din care fac parte (chiar daca nu au echipa), cursele la care au participat (daca au existat), circuitul afferent fiecarei curse (daca exista), locatia circuitului.

SELECT

P.nume || ' ' || P.prenume AS pilot,

E.nume AS echipa,

C.id\_cursa,

CI.nume AS circuit,

CI.locatie

FROM PILOT\_F1 P

LEFT OUTER JOIN ECHIPA\_F1 E ON P.id\_echipa = E.id\_echipa

LEFT OUTER JOIN PILOT\_CURSA\_F1 PC ON P.id\_pilot = PC.id\_pilot

LEFT OUTER JOIN CURSA\_F1 C ON PC.id\_cursa = C.id\_cursa

LEFT OUTER JOIN CIRCUIT\_F1 CI ON C.id\_circuit = CI.id\_circuit

ORDER BY P.nume;

### OUTER JOIN cu ECHIPA\_F1 (pilot poate sa nu aiba echipa)

### OUTER JOIN cu PILOT\_CURSA\_F1 (pilot poate sa nu aiba curse)

### OUTER JOIN cu CURSA\_F1 (cursa poate lipsi sau sa fie stearsa)

### OUTER JOIN cu CIRCUIT\_F1 (unele curse pot fi neasociate unui circuit)

### 

### Rezultat Query :

### 

### 16. Optimizarea unei cereri, aplicand regulile de optimizare ce deriva din proprietatile operatorilor algebrei relationale. Cererea va fi exprimata prin expresie algebrica, arbore algebric si limbaj (SQL), atat anterior cat si ulterior.

### Expresiile algebrice relationale sunt o reprezentare teoretica a operatiunilor asupra datelor, folosing

### simboluri si operatori precum **σ (selectie), π (proiectie), ⨝ (imbinare), ∪ (reuniune), - (diferenta) și × (produs cartezian) etc**. Acestea sunt folosite in optimizarea interogarilor SQL si in teoria bazelor de date.

|  |  |  |
| --- | --- | --- |
| SIMBOL | NUME | DESCRIERE |
| **σ** | SELECTIE | Filtreaza randurile care indeplinesc o anumigta conditie (echivalentul lui WHERE in SQL) |
| π | PROIECTIE | Selecteaza anumite coloane din tabel (echivalentul laSELECT nume, prenum in SQL) |
| ⨝ | IMBINARE | Combina doua tabele pe baza unei conditii (ecchivalentul JOINin SQL) |
| ∪ | REUNIUNE | Combina doua rezultate dinstincte intr-o singura lista (echivalentul la UNION in SQL) |
| − | DIFERENTA | Elimina elemente din primul tabel care apar si in al doilea (echivalentul EXCEPT in SQL) |
| × | PRODUS CARTEZIAN | Creeaza toate combinatiile posibile intre randurile din doua tabele (echivalentul CROSS JOIN in SQL) |
| ∩ | INTERSECTIE | Selecteaza elementele commune din doua tabele (echivalentul INTERSECT in SQL) |
| ρ | REDENUMESTE | Redenumeste o tabela sau o coloana pentru o referinta mai clara (echivalentul AS ins SQL) |
| τ | SORTARE | Sorteaza rezultatele dupa un criteriu (echivalentul luiORDER BY in SQL) |
| γ | GRUPARE | Grupeaza datele si aplica functii de agregare precum SUM, AVG (echivalentul lui GROUP BY in SQL) |

### CERERE SQL INITIALA

### Sa se afiseze numele echipei si al pilotului pentru pilotii care au castigat cel putin un titlu si fac parte dintr-o echipa u buget mai mare decat media bugetelor si echipelor.

SELECT E.nume AS nume\_echipa,

P.nume AS nume\_pilot

FROM

ECHIPA\_F1 E

JOIN PILOT\_F1 P ON E.id\_echipa = P.id\_echipa

WHERE P.titluri > 0

AND E.buget > (SELECT AVG(buget) FROM ECHIPA\_F1);

### 

### Rezultat query :

### 

### *Expresia algebrica :*

### Prescurtari :

### E = Echipa

### P = Pilot\_F1

### Pas 1 : AVG(BUGET)

### γ\_{avg(buget)} (ECHIPA\_F1)

### Pas 2 : JOIN + conditie

σ\_{P.titluri > 0 ∧ E.buget > avg\_buget} (

E ⨝\_{E.id\_echipa = P.id\_echipa} P

)

### Pas 3 : Proiectie finala

π\_{E.nume, P.nume} (

σ\_{P.titluri > 0 ∧ E.buget > avg\_buget} (

E ⨝\_{E.id\_echipa = P.id\_echipa} P

)

)

### *Arbore algebric :*

### 

### *Optimizare :*

### Aplicam :

### Push down selectii : filtram cat mai devreme

### Calculul unei constante : avg\_buget = 220000000 (exemplu)

### *Expresia algebrica optimizata :*

π\_{E.nume, P.nume} (

σ\_{E.buget > 220000000} (

σ\_{P.titluri > 0} (

E ⨝\_{E.id\_echipa = P.id\_echipa} P

)

)

)

### *Varianta cu push-down complet :*

π\_{E.nume, P.nume} (

(σ\_{E.buget > 220000000} (E)) ⨝\_{E.id\_echipa = P.id\_echipa} (σ\_{P.titluri > 0} (P))

)

### *Arbore algebric optimizat :*

### 

### *CERERE SQL OPTIMIZATA :*

WITH media AS (

SELECT AVG(buget) AS avg\_buget FROM ECHIPA\_F1

)

SELECT E.nume AS nume\_echipa,

P.nume AS nume\_pilot

FROM

ECHIPA\_F1 E

JOIN PILOT\_F1 P ON E.id\_echipa = P.id\_echipa

WHERE E.buget > (SELECT avg\_buget FROM media)

AND P.titluri > 0;

ASA