SQL Language: Exercises

1. Given the following relational schema (primary keys are underlined, optional attributes are indicated by "*")

WORKSHOP (WSID, Name, Address, City)
VEHICLE (LicensePlate, Model, Brand, Category, Power, YearRegistration, TaxCode)
CUSTOMER (TaxCode, Name, Surname, BirthDate, Address, City)
SERVICE (LicensePlate, WSID, Date, Cost)

For workshops that have serviced at least 200 different vehicles registered to customers born between 1970 and 1980, display the name and address of the workshop that carried out the most services (including all services) among workshops located in the same city. Also view the total cost of services carried out and the number of different vehicle models serviced.

Solution trace

- Find ELIGIBLE_WORKSHOPS = workshops that have serviced at least 200 different vehicles registered to customers born between 1970 and 1980
- For those workshops, display the name and address of the workshop that carried out the most services (including all services) among workshops located in the same city:
  - Compute the number of services per workshop
  - Compute the maximum number of services per workshop
  - Find the workshop which corresponds to the maximum value

```
WITH WORKSHOPS_200 AS (
SELECT WSID,                COUNT(DISTINCT LicensePlate) As LicensePlateNo FROM SERVICE S,
VEHICLE V, CUSTOMER C
WHERE S.LicensePlate = V.LicensePlate AND V.TaxCode = C.TaxCode
AND BirthDate >= 1/1/1970 AND BirthDate <= 31/12/1980
GROUP BY WSID
HAVING COUNT(DISTINCT LicensePlate) >= 200)

OR

WITH WORKSHOPS_200 AS
SELECT WID
FROM SERVICE S
WHERE LicensePlate IN (SELECT LicensePlate FROM VEHICLE
                       WHERE TaxCode IN (SELECT TaxCode
                                         FROM CUSTOMER WHERE BirthDate ….))
GROUP BY WID
HAVING COUNT(DISTINCT LicensePlate) >= 200
```

```
SELECT W.Name, W.Address, SUM(Cost), COUNT(DISTINCT Model)
FROM SERVICE S, WORKSHOP W, VEHICLE V
WHERE W.WID = S.WID AND S.LicensePlace = V.LicensePlate
AND W.WID IN ( SELECT WISD FROM WORKSHOPS_200 )
GROUP BY W.WID, W.Name, W.Address
HAVING COUNT(*) = ( SELECT MAX(NumServices)
                    FROM ( SELECT W2.WID, COUNT(*) As NumServices
                    FROM SERVICE S2
                     WHERE  S2.WID IN ( SELECT WISD FROM WORKSHOPS_200 )
                    GROUP BY W2.WID) AS NumServicesCity , WORKSHOP W2
                     WHERE S2.WID = W2.WID
                     AND NumServicesCity.City = W.City -- correlation condition
                     )
```

```
WITH WORKSHOPS_200 AS (
SELECT WSID, COUNT(DISTINCT LicensePlate) As LicensePlateNo FROM SERVICE S, VEHICLE V,
CUSTOMER C
WHERE S.LicensePlate = V.LicensePlate AND V.TaxCode = C.TaxCode  --two join conditions
AND BirthDate >= 1/1/1970 AND BirthDate <= 31/12/1980  --customers born between 1970 and
1980
GROUP BY WSID
HAVING COUNT(DISTINCT LicensePlate) >= 200)


SERVICES_BY_WORKSHOP AS (
SELECT WSID, Name, Address, City, COUNT(*) AS NumServices, SUM(Cost) AS SumCost,
COUNT(DISTINCT LicensePlate) AS CountDistinctModels
FROM SERVICE S2, WORKSHOP W2
WHERE S2.WID = W2.WID
            AND W2.WSID IN (SELECT WSID FROM WORKSHOPS_200)
GROUP BY W2.WID, Name, Address, City)


MAX_SERVICES_BY_CITY AS (
SELECT City, MAX(NumServices) AS MaxNoServicesCity
FROM SERVICES_BY_WORKSHOP
GROUP BY City)

SELECT Name, Address, SumCost, CountDistinctModels
FROM MAX_SERVICES_BY_CITY MS, SERVICES_BY_WORKSHOP SW
WHERE MS.City = SW.City AND SW.NumServices = MS.MaxNoServices
```

2. Given the following relational schema (primary keys are underlined, optional attributes are indicated by "*")

TECHNICIAN (ID, Name, Surname, BirthDate, Gender, Type)
INTERVENTION (IntID, Name, Description, HourlyCost)
BUILDING (BuildingID, Address, City, Province, Region, Type)
PERFORM_INTERVENTION (ID, IntID, Date, BuildingID, Duration)

Considering only the buildings located in the province of Turin, view the date in March 2022 in which the highest number of interventions was carried out in the buildings considered.

```
WITH BUILDING_TURIN AS
(SELECT BuildingID
FROM BUILDING
WHERE Province='Turin')   -- buildings in the province of Turin

NUM_INTERVENTIONS_DATE AS
(SELECT Date, COUNT(*) As NumInt
FROM PERFORM_INTERVENTION
WHERE Date >= 1/3/2022 AND Date < 1/04/2022    -- date is in March 2022
AND BuildingID IN (SELECT BuildingID FROM BUILDING_TURIN)
GROUP BY Date)

SELECT Date
FROM NUM_INTERVENTIONS_DATE
WHERE NumInt = (SELECT MAX(NumInt) FROM NUM_INTERVENTIONS_DATE))

or

MAX_NUM_INTERVENTIONS_DATE AS
(SELECT MAX(NumInt) AS MaxInt
FROM NUM_INTERVENTIONS_DATE)

SELECT Date FROM NUM_INTERVENTIONS_DATE ND, MAX_NUM_INTERVENTIONS_DATE MD
WHERE ND.NumInt = MD.MaxInt
```

```
SELECT Date
FROM PERFORM_INTERVENTION  P, BUILDING B
WHERE P.BuildingID = B.BuildingID AND
Date >= 1/3/2022 AND Date < 1/04/2022 AND Province = 'Turin'
GROUP BY Date
HAVING COUNT(*) = (SELECT MAX(NumInt)
                   FROM (SELECT Date, COUNT(*) As NumInt
                         FROM PERFORM_INTERVENTION  P2, BUILDING B2
                         WHERE P2.BuildingID = B2.BuildingID AND Province = 'Turin'
                        GROUP BY Date) )
```

3. Given the following relational schema (primary keys are underlined, optional attributes are indicated by "*")

LOCATION (LocID, Name, City, Region, CapacityMax)
EVENT (EvID, Title, Type)
EDITION (EvID, Date, LocID, NumberParticipants)

Among the events for which editions have been organized in at least 3 different cities, view the title of the event in which the largest number of people participated overall (considering all editions of the event).

```
-- events for which editions have been organized in at least 3 different cities

WITH EVENTS_3CITIES AS
        (SELECT EvID
        FROM EDITION  E, LOCATION L
        WHERE L.LocID = E.LocID
        GROUP BY EvID
        HAVING COUNT(DISTINCT City) >=3)
-- the total number of participants per event considering all editions

PART_EVENTS AS
        (SELECT EvID,  Title, SUM(NumberParticipants) AS TotalParticipants
        FROM EDITION ED, EVENTS_3CITIES E3
        WHERE ED.EvID = E3.EvID
        GROUP BY E3.EvID, Title)

-- find the maximum of this total

MAX_PART_EVENTS AS
    (SELECT MAX(TotalParticipants) AS MaxTotalParticipants
    FROM PART_EVENTS)


-- final query finds the event that corresponds to the maximum

SELECT Title
FROM PART_EVENTS PE, MAX_PART_EVENTS MPE
WHERE PE.TotalParticipants = MPE.MaxTotalParticipants;
```

```
SELECT Title
FROM PART_EVENTS PE
WHERE TotalParticipants = (SELECT MAX(TotalParticipants) FROM  PART_EVENTS)
SELECT  Title
FROM EVENT E, EDITION ED    -- EVENT contains the Title and EDITION the NumberParticipants
WHERE ED.EvID = E.EvID AND ED1.EvID IN (SELECT EvID
                                        FROM EDITION  E, LOCATION L
                                        WHERE L.LocID = E.LocID
                                        GROUP BY EvID
                                        HAVING COUNT(DISTINCT City) >=3
                                        )
GROUP BY EvID, Title
HAVING SUM(NumberParticipants) =
                        (SELECT MAX(TotParticipants)
                         FROM (

                                SELECT SUM(NumberParticipants) AS  TotalParticipants
                                 FROM EDITION ED1
                                 WHERE ED1.EvID IN (SELECT EvID
                                        FROM EDITION  E, LOCATION L
                                        WHERE L.LocID = E.LocID
                                        GROUP BY EvID
                                        HAVING COUNT(DISTINCT City) >=3)
                                 )
                                 GROUP BY EvID) AS Part_Event
                        );
```

4. Given the following relational schema (primary keys are underlined, optional attributes are indicated by "*", attributes with the same names indicate a foreign key)

FILM (CodF, Title, ReleaseDate, Genre, DurationMinutes)
CINEMA (CodC, Name, Address, City)
HALL(CodC, HallNumber, Capacity)
SCREENING (CodC, HallNumber, Date, StartTime, EndTime, CodF)

a) View the title of each film that has a shorter duration than the average duration of films, and that has been screened a number of times greater than the average number of screenings of films.

Solution 1

```
SELECT F1.CodF, Title , COUNT(*) AS NoScreenings
FROM FILM F1 ,  SCREENING S
WHERE DurationMinutes <
          (
              SELECT AVG(DurationMinutes)
               FROM FILM  F2
          )
AND F1.CodF = S.CodF   --join condition between S and F1
GROUP BY F1.CodF, Title     -- equivalent clause: GROUP BY S.CodF, Title
HAVING  COUNT(*)  >
--compute the average number of screening across all films
          (
            SELECT AVG(Partial)
            FROM (
                  SELECT COUNT(*) AS Partial
                  FROM SCREENING
                  GROUP BY CodF) AS PartialTable
          )
```

Solution 2 (CTE)

```
WITH AVG_DURATION AS
(SELECT AVG(F.DurationMinutes) AS AvgDuration
FROM FILM F),    -- average duration of films

SCREENING-FILM AS (SELECT S.CodF, COUNT(*) AS N
FROM SCREENING S
GROUP BY S.CodF),  -- computing the number of screenings per film

AVG_SCREENING_NUMBER AS
(SELECT AVG(N) AS AvgScreening
FROM SCREENING-FILM)  -- computing the average number of screening across all films

SELECT Title
FROM   FILM F, AVG_DURATION AD, SCREENING-FILM SF, AVG_SCREENING_NUMBER ASG
WHERE F.DurationMinutes < AD.AvgDuration  AND SF.N > ASG.AvgScreening AND F.CodF = SF.CodF
```

Solution 3 (CTE)

```
WITH AVG_DURATION AS
(SELECT AVG(F.DurationMinutes) AS AvgDuration
FROM FILM F),    -- average duration of films

SCREENING-FILM AS (SELECT F.CodF, Title, DurationMinutes, COUNT(*) AS N
FROM SCREENING S, FILM F
WHERE F.CodF = S.CodF
GROUP BY F.CodF, Title, DurationMinutes)  -- compute the number of screenings per film

SELECT Title
FROM   AVG_DURATION AD, SCREENING-FILM SF, AVG_SCREENING_NUMBER ASG
WHERE SF.DurationMinutes < AD.AvgDuration  AND SF.N > ASG.AvgScreening
```

Solution 4

```
WITH SCREENING-FILM AS (SELECT F.CodF, COUNT(*) AS N
FROM SCREENING S
GROUP BY S.CodF)  -- compute the number of screenings per film

AVG_SCREENING_NUMBER AS
(SELECT AVG(N) AS AvgScreening
FROM SCREENING-FILM)  -- compute the average number of screening across all films


SELECT Title
FROM FILM F1, SCREENING F
WHERE DurationMinutes <
            (
                SELECT AVG(DurationMinutes)
                 FROM FILM  F2
            )
AND F.CodF = F1.CodF
GROUP BY F1.CodF, Title
HAVING  COUNT(*)  >
--compute the average number of screening across all films
           (
              SELECT AvgScreening FROM AVG_SCREENING_NUMBER
           )
```

Solution 4

```
WITH SHORT_FILMS AS
(SELECT CodF, Title
FROM FILM F
WHERE DurationMinutes < (SELECT AVG(DurationMinutes)
FROM FILM),

SCREENING-FILM AS (SELECT S.CodF, COUNT(*) AS N
FROM SCREENING S
GROUP BY S.CodF
HAVING COUNT(*) > (SELECT AVG(N)
                          FROM (SELECT COUNT(*) AS N
```

```
                                FROM SCREENING S GROUP BY CodF),  -- compute the number of screenings per film
SELECT Title
FROM LONG_FILMS LF, SCREENING_FILM SF
WHERE LF.CodF = SF.CodF;
```

b) View the title of each film that has a shorter duration than the average duration of films *in the same genre*, and that has been screened a number of times greater than the average number of screenings of films *in the same genre*.

Solution 1

```
SELECT Title
FROM FILM F1, SCREENING S
WHERE DurationMinutes <
          (
              SELECT AVG(DurationMinutes)
               FROM FILM  F2
              WHERE F2.Genre = F1.Genre   -- correlation condition
          )
AND S.CodF = F1.CodF
GROUP BY F1.CodF, Title
HAVING  COUNT(*) >
--compute the average number of screening ins the same genre
          (
            SELECT AVG(Partial)
            FROM (
                  SELECT CodF, COUNT(*) AS Partial
                  FROM SCREENING
                  GROUP BY CodF) AS PS,  FILM F2    -- AS PS assigns a name to the inner query
            WHERE  F2.CodF = PS.CodF AND F2.Genre = F1.Genre – correlation condition
          )
```

Solution 2 (CTE)

```
WITH SCREENING-FILM AS
(SELECT F.CodF, Title ,Genre, DurationMinutes, COUNT(*) AS N
FROM FILM F, SCREENING S
WHERE F.CodF = S.CodF
GROUP BY F.CodF, Title, Genre, DurationMinutes)

AVG_DURATION AS
(SELECT Genre, AVG(DurationMinutes) AS AvgDurationGenre
FROM FILM
GROUP BY Genre)

AVG_SCREENING_NUMBER-GENRE AS
(SELECT Genre, AVG(N) AS AvgScreeningGenre
FROM SCREENING-FILM
GROUP BY Genre)

SELECT Title
FROM AVG_DURATION AD, SCREENING-FILM SF, AVG_SCREENING_NUMBER-GENRE ASG
WHERE AD.Genre = SF.Genre AND SF.Genre = ASG.Genre  --equivalent to the correlation condition
```

AND **SF.DurationMinutes < AD.AvgDurationGenre**
AND **SF.N > ASG.AvgScreeningGenre**

Combining AVG_DURATION and AVG_SCREENING_NUMBER in a single CTE

```
WITH SCREENING-FILM AS
(SELECT F.CodF, Genre, DurationMinutes, COUNT(*) AS N
FROM FILM F, SCREENING S
WHERE F.CodF = S.CodF
GROUP BY F.CodF, Genre, DurationMinutes)

AVG_GENRE AS
(SELECT Genre, AVG(DurationMinutes) AS AvgDurationGenre, AVG(N) AS AvgScreening
FROM SCREENING-FILM
GROUP BY Genre)

SELECT Title
FROM SCREENING-FILM SF, AVG_GENRE AG
WHERE SF.Genre = AG.Genre
AND SF.DurationMinutes < AG.AvgDurationGenre
AND SF.N > AG.AvgScreening
```