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.

```
WITH WORKSHOPS_200 AS (
SELECT WSID 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
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 V.LicensePlate = S.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 City, COUNT(*) As NumServices
                   FROM SERVICE S2, WORKSHOP W2
                   WHERE O2.WID = W2.WID AND W2.City = W.City <-correlation condition
                   GROUP BY W2.WID, City ) AS NumServicesCity
                    WHERE NumServicesCity.City = W.City <- correlation condition
                    )
```

```
SELECT W.Name, W.Address, SUM(Cost), COUNT(DISTINCT Model)
FROM SERVICE S, WORKSHOP W, VEHICLE V
WHERE W.WID = S.WID AND V.LicensePlate = S.LicensePlate
AND W.WID IN
        (SELECT WSID 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
            COUNT(DISTINCT LicensePlate) >= 200)
```

```
GROUP BY W.WID, W.Name, W.Address
HAVING COUNT(*) = (SELECT MAX(NumServices)
                   FROM ( SELECT City, COUNT(*) As NumServices
                   FROM SERVICE S2, WORKSHOP W2
                   WHERE O2.WID = W2.WID AND W2.City = W.City <-correlation condition
                   GROUP BY W2.WID, City ) AS NumServicesCity
                    WHERE NumServicesCity.City = W.City <- correlation condition
                    )
```

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' )

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

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

```
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 NumInt = (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).

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

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 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 AVG(Partial)
            FROM (
                  SELECT CodF, COUNT(*) AS Partial
                  FROM SCREENING
                  GROUP BY CodF)
          )
```

Solution 2 (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_DURATION AS
(SELECT Genre, AVG(DurationMinutes) AS AvgDurationGenre
```

```
FROM FILM)

AVG_SCREENING_NUMBER AS
(SELECT Genre, AVG(N) AS AvgScreening
FROM SCREENING-FILM)

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

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 F
WHERE DurationMinutes <
        (
            SELECT AVG(DurationMinutes)
             FROM FILM  F2
            WHERE F2.Genre = F1.Genre   -- correlation condition
        )
AND F.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
               GROUPBY 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 in bold
        )
```

Solution 2 (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_DURATION AS
(SELECT Genre, AVG(DurationMinutes) AS AvgDurationGenre
FROM FILM
GROUP BY Genre)

AVG_SCREENING_NUMBER AS
(SELECT Genre, AVG(N) AS AvgScreening
```

```
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
AND SF.DurationMinutes < AD.AvgDurationGenre
AND SF.N > ASG.AvgScreening
```

Alternative solution:

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
```