Homework 3: SQL queries

1. The following relations are given (primary keys are underlined; optional attributes are denoted with *):

AUTHOR (CodAuthor, Name, Surname, Department, University)
ARTICLE (CodArticle, Title, Topic)
ARTICLE_AUTHORS (CodArticle, CodAuthor)
CONFERENCE_EDITIONS (Conference, Edition, EditionName, StartDate, EndDate, Editor)
AUTHOR_PRESENTS_ARTICLE (CodAuthor, Date, StartHour, EndHour, Room, CodArticle, Conference, Edition)

Write the following query in SQL language:

- For each author who has presented at least 4 articles on the topic 'Data Mining', but who has never submitted articles on the topic 'Deep Learning' or 'CyberSecurity', show the name, surname, university of the author and the total number of articles presented by the author in each edition of each conference.

```
SELECT Name, Surname, University, Conference, Edition, COUNT(*)
FROM AUTHOR AU, AUTHOR_PRESENTS_ARTICLE APA
WHERE AU.CodAuthor = APA.CodAuthor
AND AU.CodAuthor IN (SELECT AA1.CodAuthor
                     FROM ARTICLE_AUTHORS AA1, ARTICLE AR1
                     WHERE AR1.CodArticle = AA1.CodArticleAND AR1.Topic = 'Data Mining'
                     GROUP BY AA1.CodAuthor
                     HAVING COUNT(*) >=4)
AND AU.CodAuthor NOT IN (SELECT AU2.CodAuthor
                         FROM ARTICLE_AUTHORS AA2, ARTICLE AR2
                         WHERE AR2.CodArticle = AA2.CodArticle
                         AND (AR1.Topic = 'Deep Learning OR AR1.Topic='CyberSecurity'))
GROUP BY AU.CodAuthor, Name, Surname, University, Conference, Edition
```

2. The following relations are given (primary keys are underlined; optional attributes are denoted with *):

STUDENT (StudentID, Name, Surname, Degree Course)
HOMEWORK_TO_SUBMIT (HWCode, Title, Topic, ExpectedDueDate)
PROFESSOR (PCode, Name, Surname, Department)
SUBMITTED_HOMEWORK_EVALUATION (StudentID, HWCode, PCode, SubmissionDate, EvaluationDate, Evaluation)

Write the following query in SQL language:

- For each student, show the student's first and last name, and the title and subject of each homework delivered by the student, for which the student has obtained a higher rating than the average rating achieved on that homework by all students.

Correlation

```
SELECT Name, Surname, Title, Topic
FROM SUBMITTED_HOMEWORK_EVALUATION SHE, STUDENT S, HOMEWORK_TO_SUBMIT HTS
WHERE S.StudentID = SHE.StudentID AND HTS.HWCode = SHE.HWCode
Evaluation > (SELECT AVG(Evaluation)
                        FROM SUBMITTED_HOMEWORK_EVALUATION SHE2
                        WHERE SHE.HWCode = SHE2.HWCode)
```

Derived table

```
SELECT Name, Surname, Title, Topic
FROM STUDENT S, SUBMITTED_HOMEWORK_EVALUATION SHE,  HOMEWORK_TO_SUBMIT HTS
        (SELECT HWCode, AVG(Evaluation) AS Avg_Eval
         FROM SUBMITTED_HOMEWORK_EVALUATION SHE2
         GROUP BY HWCode) AS HW_AVG
WHERE S.StudentID = SHE.StudentID AND HTS.HWCode = SHE.HWCode AND SHE.HWCode =
HW_AVG.HWCode AND Evaluation > Avg_Eval
```

CTE

```
WITH HW_AVG AS
      (SELECT HWCode, AVG(Evaluation) AS Avg_Eval
       FROM SUBMITTED_HOMEWORK_EVALUATION SHE2
       GROUP BY HWCode)

SELECT Name, Surname, Title, Topic
FROM STUDENT S, SUBMITTED_HOMEWORK_EVALUATION SHE, HW_AVG, HOMEWORK_TO_SUBMIT HTS
WHERE S.StudentID = SHE.StudentID AND HTS.HWCode = SHE.HWCode AND SHE.HWCode =
HW_AVG.HWCode AND Evaluation > Avg_Eval
```

3. The following relations are given (primary keys are underlined; optional attributes are denoted with *):

STUDENT (StudentID, Name, Surname, Degree Course)
HOMEWORK_TO_SUBMIT (HWCode, Title, Topic, ExpectedDueDate)
PROFESSOR (PCode, Name, Surname, Department)
SUBMITTED_HOMEWORK_EVALUATION (StudentID, HWCode, PCode, SubmissionDate, EvaluationDate, Evaluation)

Write the following query in SQL language:

- For each student who has delivered *all* the homeworks of topic 'databases', and always *before* the expected due date (SubmissionDate < ExpectedDueDate), view the student's surname and, with regard to the submitted homeworks of the topic 'databases', the average evaluation received, the total number of different teachers who carried out the evaluations, and the average number of days in which the student submitted the homeworks in advance of the expected due date (ExpectedDueDate – SubmissionDate).

```
SELECT Surname, AVG(Evaluation), COUNT(DISTINCT SHE.PCode),  AVG(ExpectedDueDate-SubmissionDate)
FROM STUDENT S, SUBMITTED_HOMEWORK_EVALUATION SHE, HOMEWORK_TO_SUBMIT HTS
WHERE HTS.Topic = 'database '
AND SHE.HWCode = HTS.HWCode
AND SHE.SubmissionDate < HTS.ExpectedDueDate
AND S.StudentID = SHE.StudentID
GROUP BY S.StudentID, Surname
HAVING COUNT(*) = (SELECT COUNT(*)
                   HOMEWORK_TO_SUBMIT HTS2
                   AND Topic = 'database')
```

4. The following relations are given (primary keys are underlined; optional attributes are denoted with *):

CLIENT (TaxCode, NameC, Surname, BirthDate, City)
RESTAURANT (IdS, NameR, Address, City, Cuisine)
ORDER (CodO, TaxCode, Date, Time, Price, IdS)
EMPLOYEE (EID, NameE, Surname, Qualification)
WORKS_IN (EID, Date, IdS)

Write the following query in SQL language:

- For each restaurant that received the most orders among all restaurants in its city, view the name of the restaurant and the total revenue earned by the restaurant on each date.

```
SELECT NameR, Date, SUM(Price)
FROM RESTAURANT R, ORDER O
WHERE R.IdS = O.IdS
AND R.IdS IN (SELECT O1.IdS
                FROM ORDER O1, RESTAURANT R1
                WHERE O1.IdS=R1.IdS
                GROUP BY O1.IdS, O1.City
                HAVING COUNT(*) = (SELECT MAX(OrdersNum)
                        FROM (SELECT O2IdS, City, COUNT(*) AS OrdersNum
                            FROM ORDER O2, RESTAURANT R2
                            WHERE O2.IdS=R2.IdS
                            GROUP BY O2.IdS, City) AS city_orders
                        WHERE city_orders.City = O1.City))
GROUP BY R.IdS, NameR, Date
```

5. Trigger

The following relations are given (primary keys are underlined; optional attributes are denoted with *):

HOTEL (CodH, Name, Category, Address, City)
HOTEL_REVIEW (CodR, ReviewDate, CodH, Score, Comment)
REVIEWS_SUMMARY (CodH, ReviewsNumber, TotalScore)
REVIEWS_OUTCOME_NOTIFICATION (CodH, ReviewDate, HotelTotalScore, CategoryAvgScore)

Write the trigger to manage the submitted hotel reviews collected through a web portal.

The HOTEL table contains the list of hotels for which you can submit a review. The table REVIEWS_SUMMARY contains, for each hotel, the *total number* of reviews received, and the *overall score* assigned to each hotel through the reviews. Consider that a hotel is listed in the REVIEWS_SUMMARY table only if at least one review has been entered for that hotel.

A new review for a hotel is submitted through the portal (insertion of a record in the table HOTEL_REVIEW). You must update the table REVIEWS_SUMMARY considering the review just submitted. Also consider the case that this is the first review entered for the hotel.

You must then enter a new record in the table REVIEWS_OUTCOME_NOTIFICATION with information on the *overall score assigned to the hotel* (HotelTotalScore attribute). The *average score assigned per category* (CategoryAvgScore attribute) calculated by considering the overall score assigned to all hotels in the same category of the hotel that received the review must also be notified.

Indications for the performance of the exercise:

You are asked to write the trigger to manage hotel reviews in the manner described above.

If necessary, use the function raise_application_error (...) to report an error. You are not required to specify parameters passed to the function.

```
CREATE TRIGGER ReviewsManagement
AFTER INSERT INTO HOTEL_REVIEW
FOR EACH ROW
DECLARE
X number; TS number;
Categoryhotel char (10); MediaCategory number;

BEGIN
---- Check if this is the first review for the Hotel
SELECT COUNT(*) into X
FROM REVIEWS_SUMMARY
WHERE CodH = :NEW.CodH;

IF (X =0) THEN
        INSERT INTO REVIEWS_SUMMARY (CodH, ReviewsNumber, TotalScore)
        VALUES (:NEW.CodH, 1, :NEW.Score);

      TS := :New.Score;
ELSE
        UPDATE REVIEWS_SUMMARY
        SET ReviewsNumber = ReviewsNumber +1,
        TotalScore = TotalScore + :NEW.Score
        WHERE CodH = :NEW.CodH;

        SELECT TotalScore INTO TS
        FROM REVIEWS_SUMMARY
        WHERE CodH = :NEW.CodH;

END IF;

SELECT Category INTO CategoryHotel
FROM HOTEL
WHERE CodH = :NEW.CodH:

SELECT AVG(TotalScore) INTO MediaCategory
FROM REVIEWS_SUMMARY RS, HOTEL H
WHERE Category = CategoryHotel
AND RS.CodH = H.CodH;

[---alternative solution
SELECT AVG(TotalScore)
FROM REVIEWS_SUMMARY RS, HOTEL H
```

```
WHERE Category IN (SELECT Category
                   FROM HOTEL
                   WHERE CodH = :NEW.CodH)
AND RS.CodH = H.CodH; ]

INSERT INTO REVIEWS_OUTCOME_NOTIFICATION (CodH, ReviewDate, HotelTotalScore, CategoryAvgScore)
VALUES (:NEW.CodH, :NEW.ReviewDate, TS, MediaCategory);

END;
```