Quaderno 3: Linguaggio SQL

1. Sono date le seguenti relazioni (le chiavi primarie sono sottolineate, gli attributi opzionali sono indicati con *):

AUTORE (CodAutore, Nome, Cognome, Dipartimento, Università)
ARTICOLO (CodArticolo, Titolo, Argomento)
AUTORI_ARTICOLO (CodArticolo, CodAutore)
EDIZIONI_CONFERENZA (Conferenza, Edizione, NomeEdizione, DataInizio, DataFine, Editore)
AUTORE_PRESENTA_ARTICOLO (CodAutore, Data, OraInizio, OraFine, Sala, CodArticolo, Conferenza, Edizione)

Esprimere la seguente interrogazione in linguaggio SQL

- Per ciascun autore che ha presentato almeno 4 articoli di argomento 'Data Mining', ma che non ha mai presentato articoli di argomento 'Deep Learning' o 'CyberSecurity', visualizzare il nome, il cognome, l'università di afferenza dell'autore e il numero totale di articoli presentati dall'autore in ciascuna edizione di ogni conferenza.

2. Sono date le seguenti relazioni (le chiavi primarie sono sottolineate, gli attributi opzionali sono indicati con *):

STUDENTE (MatricolaS, Nome, Cognome, Corso di Laurea)
HOMEWORK_DA_CONSEGNARE (CodHW, Titolo, Argomento, DataScadenzaPrevista)
DOCENTE (CodDocente, Nome, Cognome, Dipartimento)
VALUTAZIONE_HOMEWORK_CONSEGNATI (MatricolaS, CodHW, CodDocente, DataConsegna, DataValutazione, Valutazione)

Esprimere la seguente interrogazione in linguaggio SQL

- Per ciascuno studente, visualizzare il nome e il cognome dello studente, e il titolo e l'argomento di ciascun homework per cui lo studente ha ottenuto una valutazione superiore alla valutazione media conseguita su quell'homework da tutti gli studenti

1

```
SELECT Nome, Cognome, Università, NArticoli
FROM Autore AS A, (SELECT CodAutore, count(distinct CodArticolo) AS NArticoli
                   FROM Autore_presente_Articolo AS APA
                   GROUP BY CodAutore AS NArticoliAut)
WHERE CodAutore in (SELECT CodAutore
                    FROM Autore_presente_Articolo AS APA, Articolo AS A
                    WHERE APA.CodAutore = A.CodArticolo
                    and Argomento = 'Programmi'
                    HAVING count(distinct CodArticolo) > 3
                    GROUP BY CodAutore)
and CodAutore not in (SELECT CodAutore
                      FROM Autore_presente_Articolo AS APA, Articolo AS A
                      WHERE APA.CodAutore = A.CodArticolo
                      and Argomento = 'Cybersecurity'
                      GROUP BY CodAutore)
and CodAutore not in (SELECT CodAutore
                      FROM Autore_presente_Articolo AS APA, Articolo AS A
                      WHERE APA.CodAutore = A.CodArticolo
                      and Argomento = 'Basi di dati'
                      GROUP BY CodAutore)
AND A.CodAutore = NArticolo.CodAutore;
```

2

```
SELECT Nome, Cognome, Titolo, Argomento
FROM Studente AS S, Valutazione_Homework_Consegnati AS HC, Homework_da_Consegnare AS H
WHERE S.NArticoloS = HC.NArticoloS AND HC.CodHW = H.CodHW
and Valutazione > (SELECT AVG(Valutazione)
                   FROM Valutazione_Homework_Consegnati AS HC
                   WHERE CodHW = HC.CodHW
                   GROUP BY CodHW);
```

3. Sono date le seguenti relazioni (le chiavi primarie sono sottolineate, gli attributi opzionali sono indicati con *):

STUDENTE (MatricolaS, Nome, Cognome, CorsodiLaurea)
HOMEWORK_DA_CONSEGNARE (CodHW, Titolo, Argomento, DataScadenzaPrevista)
DOCENTE (CodDocente, Nome, Cognome, Dipartimento)
VALUTAZIONE_HOMEWORK_CONSEGNATI (MatricolaS, CodHW, CodDocente, DataConsegna, DataValutazione, Valutazione)

Esprimere la seguente interrogazione in SQL

- Per ogni studente che ha consegnato *tutti* gli homework di argomento 'basi di dati', e sempre *prima* della data di consegna prevista (DataConsegna < DataScadenzaPrevista), visualizzare il cognome dello studente e, relativamente agli homework di argomento 'basi di dati' consegnati, la valutazione media ricevuta, il numero complessivo di docenti diversi che hanno effettuato le valutazioni, ed il numero medio di giorni in cui lo studente ha consegnato gli homework in anticipo rispetto alla data di consegna prevista (DataScadenzaPrevista – DataConsegna),

4. Sono date le seguenti relazioni (le chiavi primarie sono sottolineate, gli attributi opzionali sono indicati con *):

CLIENTE (CodFiscale, NomeC, Cognome, DataNascita, Città)
RISTORANTE (CodR, NomeR, Indirizzo, Città, Cucina)
ORDINI (CodO, CodFiscale, Data, Ora, Prezzo, CodR)
DIPENDENTE (MatrD, NomeD, Cognome, Qualifica)
LAVORA_IN (MatrD, Data, CodR)

Esprimere la seguente interrogazione in SQL

- Per ogni ristorante che ha ricevuto complessivamente il maggior numero di ordini tra i ristoranti presenti nella sua stessa città, visualizzare il nome del ristorante e l'incasso totale ottenuto dal ristorante in ciascuna data.

3

```
SELECT cognome, AVG(V.valutazione) AS valarrot,
       AVG(DataScadenzaPrevista - DataConsegna) AS ritardogiorni,
       COUNT(DISTINCT CodDocente) AS numdocenti valutatori,
FROM Studente S, Homework_da_consegnare H, Valutazione_Homework_consegnati V, Docente D
WHERE S.Matricola S = V.Matricola S AND S.MatricolaS = V.MatricolaS AND H.CodHW = V.CodHW
AND V.CodDocente <> D.CodDocente AND H.Argomento = 'Basi di Dati'
AND V.DataConsegna < H.DataScadenzaPrevista
HAVING COUNT(DISTINCT CodHW = (SELECT COUNT(*)
                               FROM Homework_da_consegnare AS H
                               WHERE Argomento = 'Basi di Dati'
                               GROUP BY CodHW)
GROUP BY Matricola, Cognome;
```

4

```
WITH DataTab AS (SELECT CodR, COUNT(*) AS Nomini, SUM(Prezzo) AS IncassoTot, Data
                 FROM DataTab AS T, Ristorante AS R
                 WHERE T.CodR = R.CodR
                 GROUP BY CodR)
OrdiniTab AS (SELECT CodR, SUM(Nomini) AS NominiTot, Città
              FROM DataTab AS T, Ristorante AS R
              WHERE T.CodR = R.CodR
              GROUP BY CodR)
SELECT Nome R, IncassoTot, Data
FROM Ristorante, DataTab, OrdiniTab
WHERE OrdiniTab.NominiTot = (SELECT MAX(NominiTot)
                             FROM OrdiniTab AS T
                             WHERE T.Città = Ristorante.Città)
AND R.CodR = DataTab.CodR AND R.CodR = OrdiniTab.CodR
```

5. Sia dato il seguente schema relazionale (le chiavi primarie sono sottolineate, gli attributi opzionali sono indicati con '*'):

HOTEL (CodH, Nome, Categoria, Indirizzo, Città)
RECENSIONE_HOTEL (CodR, DataRecensione, CodH, Punteggio, Commento)
RIASSUNTO_RECENSIONI (CodH, NumeroRecensioni, PunteggioComplessivo)
NOTIFICA_RECENSIONI_COMPLESSIVE (CodH, DataRecensione,
PunteggioComplessivoHotel, PunteggioMedioCategoria)

Si scriva il trigger per gestire le recensioni di hotel raccolte attraverso un portale web.

La tabella HOTEL contiene l'elenco degli hotel per cui è possibile inviare una recensione. La tabella RIASSUNTO_RECENSIONI contiene, per ogni hotel, il *numero complessivo* di recensioni ricevute e il *punteggio complessivo* assegnato a ciascun hotel mediante le recensioni. Si consideri che un hotel è presente nella tabella RIASSUNTO_RECENSIONI solo se è stata inserita almeno una recensione per quell'hotel.

Viene inserita attraverso il portale una nuova recensione per un hotel (inserimento di un record nella tabella RECENSIONE_HOTEL). Il trigger deve svolgere le seguenti operazioni.

Si deve aggiornare la tabella RIASSUNTO_RECENSIONI tenendo conto della recensione appena inoltrata. Si consideri anche il caso che questa sia la prima recensione inserita per l'hotel.

Si deve quindi inserire un nuovo record nella tabella NOTIFICA_ RECENSIONI_COMPLESSIVE con le informazioni sul *punteggio complessivo assegnato all'hotel* (attributo PunteggioComplessivoHotel). Deve inoltre essere notificato il *punteggio medio assegnato per categoria* (attributo PunteggioMedioCategoria) calcolato come punteggio complessivo medio per gli hotel della stessa categoria dell'hotel che ha ricevuto la recensione.

Indicazioni per lo svolgimento dell'esercizio:

Si chiede di scrivere il trigger per gestire le recensioni di hotel secondo le modalità sopra riportate.

Se necessario, usare la funzione raise_application_error (…) per segnalare un errore. Non è richiesto di specificare i parametri passati alla funzione.

(5) • EVENTO: - inserimento nuovo evento calendario

• CONDIZIONE: nessuna

• MODALITÀ: [immediato/differito] [before/after] - immediato - after

• GRANULARITÀ: [tupla/istruzione] - tupla - inserimento un evento (no tutto)

• AZIONE: - aggiornamento/inserimento in SOMMARIO_CATEGORIA

- aggiornamento: se ∃ già un evento della stessa categoria nello stesso anno aggiornare il conteggio ed il costo complessivo

```
CREATE OR REPLACE TRIGGER nuove_recensioni
AFTER INSERT ON recensione_hotel
FOR EACH ROW
DECLARE N     NUMBER;
        PT    NUMBER;
        PM    NUMBER;
        NCAT  CHAR;
BEGIN
-- aggiorna numero totale recensioni e punteggio complessivo hotel
  SELECT COUNT(*) INTO N
  FROM riassunto_recensioni AS RR
  WHERE RR.CodH = :NEW.CodH
-- verifica se è la 1° recensione per l'hotel
  IF N <> 0
  UPDATE riassunto_recensioni
  SET numero_recensioni = numero_recensioni + 1
  SET punteggio_complessivo = punteggio_complessivo + :NEW.punteggio
  WHERE riassunto_recensioni.CodH = :NEW.CodH
  ELSE
  INSERT INTO riassunto_recensioni
  VALUES (:NEW.CodH, 1, :NEW.punteggio);
  END IF;
```

```
-- Calcolo il punteggio complessivo Hotel
   SELECT PunteggioComplessivo INTO PT
   FROM   RiassuntoRecensioni AS RR
   WHERE  RR.CodH = :NewCodH;
-- Calcolo punteggio categoria Hotel
   SELECT CategoriaH INTO NCat
   FROM Hotel AS H
   WHERE H.CodH = :NewCodH;
-- Calcolo punteggio medio Hotel
   SELECT AVG (PunteggioComplessivo PN
   FROM RiassuntoRecensioni AS RR, Hotel H
   WHERE RR.CodH = H.CodH AND H.CategoriaH = NCat;
   INSERT INTO NotificaRecensioniComplessive
   VALUES (:NewCodH, :NewDataRecensione, PT, PN);
END;
```