Esercizio 1

$S_1 \leftarrow \sigma_{\text{DataSol IS NOT NULL AND DataR > DataS}}(\text{PRESTITO})$

Prendo tutti i prestiti in ritardo, quindi i prestiti che hanno ricevuto un sollecito e che sono stati restituiti dopo la data di scadenza.

$S_2 \leftarrow \tilde{\pi}_{\text{Nome}}(\text{PROFILO} \bowtie S_1)$

Prendo i nomi di tutte le persone che hanno avuto un prestito in ritardo

$S_3 \leftarrow \tilde{\pi}_{\text{Nome}}(\text{PROFILO} \bowtie \text{PRESTITO})$

Prendo i nomi di tutte le persone che hanno effettuato un prestito

$\tilde{\pi}_{\text{Nome}}(S_3 - S_1)$

Tolgo a tutte le persone quelle che hanno avuto un prestito in ritardo così da avere solo i nomi di quelli che non li hanno mai avuti.

Esercizio 2

```
SELECT l.Titolo, COUNT(p.CodPrestito) AS Conteggio
FROM LIBRO l
JOIN ESEMPLARE e ON l.ISBN = e.ISBN
JOIN PRESTITO p ON e.CodiceBarre = p.CodiceBarre
WHERE p.DataE LIKE '2022-__-__'
GROUP BY l.Titolo
ORDER BY Conteggio DESC
LIMIT 1;
```

Quello che ho fatto è stato contare il numero di prestiti di ogni libro. Nel WHERE ho controllato che il prestito sia stato effettuato nel 2022 (formato data YYYY-MM-DD). Ho raggruppato il conteggio in base al titolo così da non aggiungere un'ulteriore colonna. Con ORDER BY ho ordinato il conteggio in ordine decrescente così da avere nella prima riga il valore più alto. Con LIMIT infatti vado a prendere solo questo rigo.

Esercizio 3

1)
```
CREATE ASSERTION A1
CHECK (SELECT COUNT(*)
       FROM PRESTITO p
       WHERE p.DataSol IS NULL AND p.DataR is NULL)
       <
       (SELECT pr.MaxPrestito
        FROM PROFILO pr
        WHERE pr.CodProfilo = p.Utente)
```

Ho considerato Utente chiave esterna di CodProfilo. Controllo che il numero di prenotazioni in corso sia sempre minore del numero massimo possibile, facendo attenzione che stia facendo riferimento sempre allo stesso utente.

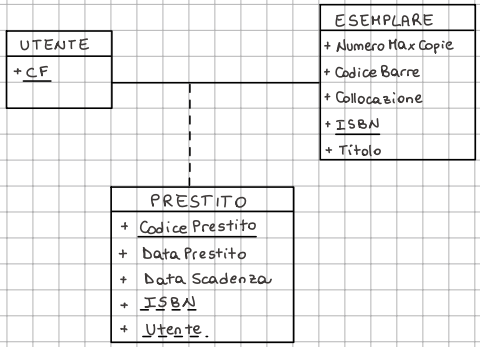
2)
```
ALTER TABLE PRESTITO
ADD CONSTRAINT A2
CHECK (DataSol IS NOT NULL AND DataR > DataS)
```

Basta controllare che entrambe le condizioni richieste siano soddisfatte contemporaneamente.

3)
```
ALTER TABLE UTENTE
ADD CONSTRAINT A3
UNIQUE (CF, CodProfilo)
```

Esercizio 4

E' presente una generalizzazione di tipo overlapping totale. Overlapping perché un'istanza può essere membro di più di una sottoclasse (un esemplare può essere sia digitale sia cartaceo); Totale perché un'istanza non è possibile non appartenga ad alcuna sottoclasse. Io risolvo andando ad accorpare le classi figlie nel padre in quanto non ci sono particolari relazioni da dover considerare. Inoltre l'entità log Esemplare Digitale è ridondante, quindi eliminabile; l'entità libro, data la presenza di una relazione 1..1, possiamo accorparlo alla nuova entità Esemplare. Avendo accorpato le classi figlie nel padre, aggiungo un attributo "Tipologia" che indica se l'esemplare è cartaceo o digitale. Infine aggiungo parole chiave ed esterne mancanti:



UTENTE(CF)
PRESTITO(Codice Prestito, Data Prestito, Data Scadenza, ISBN, Utente)
ESEMPLARE(Numero Max Copie, Codice Barre, Collocazione, ISBN, Titolo)

Esercizio 5

1)
```
CREATE TRIGGER T1
  AFTER INSERT ON PRESTITO
  FOR EACH ROW
  BEGIN
    DELETE FROM PRENOTAZIONE p
    WHERE p.Utente = NEW.Utente
  END;
```

2)
```
CREATE TRIGGER T2
  BEFORE INSERT ON PRESTITO
  FOR EACH ROW
  BEGIN
    DECLARE @ISBN ESEMPLARE.ISBN%TYPE = (SELECT e.ISBN
                                         FROM ESEMPLARE e
                                         WHERE e.CodiceBarre = NEW.CodiceBarre)

    IF (NEW.Utente IN PRENOTAZIONE.Utente AND @ISBN IN PRENOTAZIONE.ISBN) THEN
      DELETE FROM PRENOTAZIONE p
      WHERE NEW.Utente = p.Utente AND @ISBN = p.ISBN;
    END IF;
END;
```