Dato il seguente schema logico e i seguenti vincoli, si scriva il codice SQL che crea le tabelle corrispondenti. Si ricordi di creare le tabelle nell'ordine richiesto dal linguaggio SQL.

esami_medici(dataEOra, nazione, ssn, file,descrizione)
pazienti(nazione, ssn, nome,cognome,sesso,data_nascita,altezza,peso)
nazioni(id, sigla,nome,estensione,europea,numero_abitanti)

Vincoli integrità referenziale
pazienti.nazione -> nazione.id
esami_medici.nazione, esami_medici.ssn -> pazienti.nazione, pazienti.ssn

Altri vincoli e osservazioni
A parte le chiavi, tutti gli altri campi possono essere nulli.
Il nome di ogni nazione è unico
La sigla di una nazione è una sequenza di tre caratteri ed è unica.
Il campo europea deve contenere uno dei valori "0" o "1" e deve avere per defaul "1"
Il campo ssn di paziente è il social security number ed è campo di caratteri a lunghezza variabile fino ad un massimo di 30 caratteri
L'estensione di una nazione è memorizzata con un valore in virgola mobile
Il campo file di esami_medici deve permettere la memorizzazione di file binari fino a 50M
Il campo sesso, quando non è nullo, deve contenere uno dei seguenti due caratteri "u", "d".
I campi altezza e peso devono poter contenere valori in cirgola fissa fino a 999.99
Il campo dataEOra contiene la data e l'ora dell'esame

Soluzione

```
create table nazioni(
        id serial not null primary key,
        sigla char(3) unique,
        nome varchar(20) unique,
        estensione float,
        europea bt default 1,
        numero_abitanti int)

create table pazienti(
        nazione int not null references nazioni(id),
        ssn varchar(30) not null,
        nome varchar(20),
        cognome  varchar(20),
        sesso char check(sesso is null or sesso='u' or sesso='d'),
        data_nascita date,
        altezza decimal(5,2),
        peso decimal(5,2),
        primary key (nazione,ssn),
        )

create table esami_medici(
        dataEOra  timestamp not null,
        nazione int not null,
        ssn varchar(30) not null,
        file BLOB(50M),
        descrizione varchar(20),
        primary key(dataEOra,nazione,ssn),
        foreign key (nazione,ssn) references pazienti(nazione,ssn)
```

Sia dato il seguente schema relazionale

giocatori(<u>id</u>, nomeGiocatore,dataNascita)
contratti(<u>giocatore,squadra,</u>costo,anno)
squadre(<u>id</u>, nomeSquadra, citta)
fratelli(<u>giocatore1,giocatore2</u>)

Vincoli integrità referenziale
contratti.squadra -> squadre.id
contratti.giocatore -> giocatori.id
fratelli.giocatore1 -> giocatori.id
fratelli.giocatore2 -> giocatori.id

A. Scrivere le seguenti interrogazioni in algebra relazionale con theta join e in calcolo su tuple

1. Trovare il costo del contratto più costoso.

$contratti2 = \delta_{costo2<-costo}(\pi_{costo}(contratti))$
$contratti1 = \pi_{costo}(contratti)$
$costoNonMassimo = \pi_{costo}(contratti1 \bowtie_{costo<costo2} contratti2)$
$\pi_{costo}(contratti) - costoNonMassimo$

{c.costo |c(contratti) | ¬∃ c2(contratti) c2.costo>c.costo}

1a Trovare il costo del secondo contratto più costose.

$contratti2 = \delta_{costo2<-costo}(\pi_{costo}(contratti))$
$contratti1 = \pi_{costo}(contratti)$
$costoNonMassimo = \pi_{costo}(contratti1 \bowtie_{costo<costo2} contratti2)$
$costoMassimo = \pi_{costo}(contratti) - costoNonMassimo$
$costoNonMassimo2 = \delta_{costo2<-costo}(costoNonMassimo)$
costoNonInSecondaPosizioneNePrima=
$\pi_{costo}(costoNonMassimo \bowtie_{costo<costo2} costoNonMassimo2)$
$\pi_{costo}(contratti) - costoMassimo -$ costoNonInSecondaPosizioneNePrima

{c.costo |c(contratti) | ∃ c2(contratti) c2.costo>c.costo ∧ ¬∃ c3(contratti) c3.costo>c2.costo }

2. Trovare l'id delle squadre che non hanno giocatori con costo maggiore di 100.

$squadreCostose = \pi_{squadre}(\sigma_{costo>100}(contratti))$
$\pi_{id}(squadre) - \delta_{id<-squadre}(squadreCostose)$

{s.id |s(squadra) | ¬∃ c(contratti) c.costo>100 ∧ s.id=c.squadra}}

1 **Scrivere le seguente interrogazioni sia in SQL**
(se necessario, definire delle viste)

3. Trovare tutti i contratti delle squadre milanesi (citta=milano), restituendo il nome della squadra, il nome dl giocatore e l'ammontare del contratto. Restituire anche le squadre senza giocatori. Ordinare i risultati per nome della squadra e, a parità di squadra, per nome del giocatore.

```
SELECT squadre.nome, giocatori.nome, contratti.costo
FROM squadre LEFT JOIN contratti ON squadre.id=contratti.squadre
        LEFT JOIN giocatori ON contratti.giocatore=giocatori.id
WHERE citta='milano'
ORDER BY squadre.nome, giocatori.nome
```

4. Trovare le coppie di giocatori che sono fratelli, restituendo il nome e la data di nascita di entrambi.

```
SELECT g1.nome, g2nome
FROM giocatori g1, giocatori g2, fratelli
WHERE g1.id=frattelli.giocatore1 AND g2.1id=fratelli.giocatore2
        AND g1.id>g2.id
```

5. Si supponga che il campo costo della tabella contratti contenga il costo netto del contratto e che il costo lordo possa essere stimato moltiplicando il costo netto per 1,7. Si calcoli la somma totale del costo lordo stimato per tutti i contratti di ciascuna squadra, restituendo il nome della squadra e la somma.

```
SELECT squadra.nome, 1.7*SUM(costo) AS coso_lordo
FROM squadre, contratti
WHERE squadre.id=cotratti.squadra
GROUP BY squadra.id, squadra.nome
```

6. Trovare il numero dei contratti avuti da ogni giocatore, restituendo il nome del giocatore e il numero dei contratti. Nel conteggio si considerino solo i contratti stipulati dopo il 2000 (anno=2000) e si restituisca solo i giocatori che hanno stipulato più di 2 contratti in tale periodo.

```
SELECT giocatori.nome, count(*) AS umer_contratti
FROM contratti, giocatori
WHERE contratti.gocatore=giocatori.id AND anno>2000
GROUP BY giocatori.id, giocatori.nome
HAVING count(*)>2
```

7. Usando gli operatori insiemistici (senza sottointerrogazioni), si trovi id e nome delle squadre che hanno fatto dei contratti sia nel 2010 che nel 2011

```
SELECT id, nome
FROM squadre, contratti
WHERE squadre.id=contratti.squadre AND anno=2010
INTERSECT
SELECT id, nome
FROM squadre, contratti
WHERE squadre.id=contratti.squadre AND anno=2011
```

8. Usando le sottointerrogazioni (senza operatori insiemistici), trovare il nome dei giocatori che non hanno fatto contratti nel 2013.

```
SELECT nome
FROM giocatori
WHERE NOT EXISTS(
        SELECT *
    FROM contratti
    WHERE anno=2013 AND giocatori.id=contratti.giocatore
)
```

9. Trovare il nome dei giocatori che hanno avuto contratti il cui costo è minore del costo medio dei contratti.

```
SELECT nome
FROM giocatori, contratti
WHERE giocatori.id=contratti.giocatore AND contratti.costo< (SELECT AVG(costo)
                                                             FROM contratti )
```

10. Trovare quanto ha speso la società che ha speso di più per i contratti stipulati (nb non è necessario trovare anche il nome lo società che raggiunge tale massimo; si usino i costi netti dei contratti).

```
CREATE VIEW costi(costo)
AS
SELECT sum(costo)
FROM contratti
GROUP BY squadra

SELECR max(costo)
FROM costi
```