A. 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.

citta(id, nome, nazione, numero_cittadini, mappa)
uscite(citta, sigla_autostrada, nazione_autostrada, nome_uscita, longitudine, latidudine, data_ora_chiusura, data_ora_riapertura)
autostrade(sigla, nazione, nome_eu, a_pagamento, larghezza, data_inaugurazione)

Vincoli integrità referenziale
uscite. citta -> citta.id
uscite.sigla_autostrada, uscite.nazione_autostrada -> autostrade.sigla, autostrade.nazione

Altri vincoli
A parte le chiavi, tutti gli altri campi possono essere nulli.
In ogni nazione tutte le città hanno nome diverso
Il campo mappa della città deve poter contenere delle immagini fino a 10MB
Il campo latitudine deve essere rappresentato come un decimale con 2 cifre intere e 7 cifre di mantissa.
Il campo longitudine deve essere rappresentato come un decimale con 3 cifre intere e 7 cifre di mantissa.
La latitudine e la longitudine, quando non sono nulle, devono essere comprese fra -90 e 90 e -180 e 180, rispettivamente.
I campi data_ora_chiusura e data_ora_riapertura servono a memorizzare temporanee chiusure delle uscite. Tali campi devono poter memorizzare data e ora della chiusura e della riapertura.
Il campo sigla dell'autostrade è una stringa lunga fino a 10 caratteri.
Il campo nome_eu è una stringa lunga esattamente 4 caratteri.
Il campo a_pagamento deve contenere uno dei valori "0" o "1" e deve valere "1" per default
Il campo larghezza di autostrade deve contenere valori in virgola mobile

Note utili a comprendere lo schema
La tabella uscite contiene le uscite dell'autostrade. Le uscite di ciascun'autostrada hanno un nome unico in modo da distinguerle fra loro. Per questo motivo si è scelto di usare la tripla sigla_autostrada, nazione_autostrada, nome_uscita come chiave di uscite.

Sia dato il seguente schema relazionale

Stazioni(id, nome, citta, numero_binari)
fermate(stazione, treno, arrivo, partenza)
treni(id, nome, lunghezza)
note(id, stazione, treno, descrizione)

Vincoli integrità referenziale
fermate.stazione -> stazioni.id
fermata.treno -> treni.id
note.stazione, note.treno -> fermata.stazione, fermata.treno

Note utili a comprendere lo schema
La tabella note contiene annotazioni sulle fermate dei treni. Ogni nota fa riferimento a una fermata (vedi vincoli integrità). Ad una fermata possono essere associate più note.

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

1. Trovare tutte le fermate dei treni in arrivo alle stazioni con id>100, restituendo il nome della stazione, il nome del treno, l'ora della fermata, e il tempo trascorso fra l'arrivo e la partenza: Per calcolare la differenza fra due orari si può usare semplicemente usare il simbolo "-". Restituire anche le stazioni in cui non arriva nessun treno.
2. Trovare le coppie di stazioni che si trovano nella stessa città, restituendo la citta e il nome di entrambe le stazioni. Ordinare i risultati per citta e, a parità di citta, per nome di una delle due stazioni (ad es., quella della prima colonna).
3. Si calcoli il numero di note esistenti nell'archivio per ciascuna fermata, restituendo l'id del treno, l'id della stazione e il numero delle note.
4. Trovare il numero delle fermate fatte da ciascuno treno dopo le 12 (arrivo>12), restituendo l'id del treno e il numero delle fermate. Mostrare solo i treni con almeno 2 fermate dopo le 12.
5. Usando gli operatori insiemistici (senza sottointerrogazioni), si trovino gli orari in cui arriva o parte un treno nella stazione di Siena. L'interrogazione deve restituire una tabella con una sola colonna dove ogni riga mostra un orario che può essere di partenza o di arrivo.
6. Usando le sottointerrogazioni (senza operatori insiemistici), calcolare il numero delle stazioni in cui non si ferma nessun treno .
7. Trovare il nome della stazione con pi*ù* binari
8. Trovare il numero medio e il numero minimo di fermate fatte dai treni dell'archivio.

```
create table citta(
        id serial not null primary key,
        nome varchar(20),
        nazione varchar(20),
        numero_cittadini int,
        mappa blob(10m),
        unique(nome,nazione))

create table autostrade(
        sigla varchar(10) not null,
        nazione varchar(20) not null,
        nome_eu char(4),
        a_pagamento bit default '1',
        larghezza float,
        data_inaugurazione date,
        primary key(sigla,nazione) )

create table uscite(
        citta integer references citta(id) references citta(id),
        sigla_autostrada varchar(10) not null,
        nazione_autostrada varchar(20) not null,
        nome_uscita varchar(20) not null,
        longitudine decimal(9,7)
                check (longitudine isnull or (longitudine>=-90 and  longitudine<=90)),
        latidudine decimal(9,7)
                check (latidudine isnull or (latidudine>=-180 and  latidudine<=180)),
        data_ora_chiusura timestamp,
        data_ora_riapertura timestamp,
        foreign key (sigla_autostrada,nazione_autostrada) references autostrade(sigla,nazione),
        primary key(sigla_autostrada,nazione_autostrada,nome_uscita))
```

```
create table stazioni(
        id serial not null primary key,
        nome varchar(20),
        citta varchar(20),
        numero_binari int)

create table treni(
        id serial not null primary key,
        nome varchar,
        lunghezza float)

create table fermate(
        stazione int references stazioni(id),
        treno int references treni(id),
        arrivo time,
        partenza time)

create table note(
        id serial not null primary key,
        stazione int,
        treno int,
        descrizione varchar(100),
        foreign key(stazione,treno) references fermate(stazione, treno))
```

1. Trovare tutte le fermate dei treni in arrivo alle stazioni con id>100, restituendo il nome della stazione, il nome del treno, l'ora della fermata, e il tempo trascorso fra l'arrivo e la partenza: Per calcolare la differenza fra due orari si può usare semplicemente usare il simbolo "-". Restituire anche le stazioni in cui non arriva nessun treno.

   ```
   select stazioni.nome, treni.nome, arrivo, partenza-arrivo
   from stazioni join fermate on stazioni.id=fermate.stazione
       join treni on fermate.treno=treni.id
   where stazioni.id>100
   ```

2. Trovare le coppie di stazioni che si trovano nella stessa città, restituendo la citta e il nome di entrambe le stazioni. Ordinare i risultati per citta e, a parità di citta, per nome di una delle due stazioni (ad es., quella della prima colonna).

   ```
   select stazione1.citta, stazione1.nome as nome_stazione1 , stazione2.nome as nome_stazione2
   from stazioni as stazione1, stazioni as stazione2
   where stazione1.citta=stazione2.citta and
        stazione1.id>stazione2.id
   order by stazione1.citta, stazione1.nome
   ```

3. Si calcoli il numero di note esistenti nell'archivio per ciascuna fermata, restituendo l'id del treno, l'id della stazione e il numero delle note.

```
select fermate.treno, fermate.stazione, count(*) as numero_note
from fermate, note
where fermate.treno=note.treno and fermate.stazione=note.stazione
group by fermate.treno, fermate.stazione
```

4. Trovare il numero delle fermate fatte da ciascuno treno dopo le 12 (arrivo>12), restituendo l'id del treno e il numero delle fermate. Mostrare solo i treni con almeno 2 fermate dopo le 12.

```
select fermate.treno, count(*) as numero_fermate
from fermate
where arrivo > '12:00'
group by fermate.treno
having count(*) >=2
```

5. Usando gli operatori insiemistici (senza sottointerrogazioni), si trovino gli orari in cui arriva o parte un treno nella stazione di Siena. L'interrogazione deve restituire una tabella con una sola colonna dove ogni riga mostra un orario che può essere di partenza o di arrivo.

```
select partenza as partenza_arrivo
from fermate, stazioni
where fermate.stazione=stazioni.id
and stazioni.nome='siena'
union
select arrivo as partenza_arrivo
from fermate, stazioni
where fermate.stazione=stazioni.id
and stazioni.nome='siena'
```

6. Usando le sottointerrogazioni (senza operatori insiemistici), calcolare il numero delle stazioni in cui non si ferma nessun treno.

```
select count(*) numero_stazioni
from stazioni
where id not in (select stazione from fermate)
```

oppure

```
select count(*) numero_stazioni
from stazioni
where  not exists (select * from fermate where fermate.stazione=stazioni.id )
```

7. Trovare il nome della stazione con pi*ù* binari

```
select nome
```

```
from stazioni
where numero_binari = (select max(numero_binari) from stazioni)
```

8. Trovare il numero medio e il numero minimo di fermate fatte dai treni dell'archivio.

```
create view numero_fermate(treno, n_fermate)
as select treno, count(*)
from fermate
group by treno

select avg(numero_fermate), min( n_fermate)
from numero_frermate
```