Esercizi

Una porzione del database per una compagnia di taxi è caratterizzata dal seguente schema logico relazionale:[1]

SEDI(codice, citta, indirizzo, telefono)
TAXI(targa, sede$_{fk}$, modello, anno_immatricolazione, scadenza_revisione)
AUTISTI(matricola, cognome_nome, telefono, numero_patente, scadenza_patente)
TURNI(taxi$_{fk}$, giorno, orario, autista$_{fk}$)
CORSE(progr, [taxi, giorno, orario]$_{fk(turni)}$, partenza, destinazione, km, importo, contanti$_{bool}$)

Con riferimento allo schema suddetto, esprimere le seguenti interrogazioni in linguaggio SQL (ove non altrimenti specificato).

1. Scrivere l'istruzione DDL per la definizione della relazione TURNI includendo, oltre ai vincoli indicati nello schema, il vincolo che impone che nell'attributo 'orario' sia memorizzato uno dei valori dell'insieme $\{0, 6, 12, 18\}$.

2. Aggiungere alla tabella AUTISTI il vincolo che impedisce a due autisti di avere lo stesso numero di patente.

3. Eliminare dal database tutti i taxi che soddisfano tutte le seguenti condizioni: l'anno di immatricolazione è precedente al 2010; la data di scadenza della revisione è passata da almeno due anni; non sono stati utilizzati in alcun turno dalla data di scadenza della revisione fino alla data odierna.

4. Elencare gli autisti che, pur avendo effettuato almeno un turno nell'ultima settimana, non hanno effettuato alcuna corsa durante tutti i turni dell'ultima settimana.

5. Definire la vista relazionale CONTANTI_PER_TURNO, che associa ad ogni turno la somma degli importi delle corse effettuate durante quel turno che sono state pagate in contanti. Oltre ai dati suddetti, la vista deve includere la matricola e nome dell'autista che ha effettuato il turno, nonché la città della sede del taxi.

6. Elencare gli autisti che hanno effettuato almeno 100 turni, ordinandoli in base all'importo totale incassato nelle corse effettuate (a parità di importo incassato, usare l'ordinamento in base al nome dell'autista).

7. Usando l'algebra relazionale, estrarre i dati degli autisti che hanno effettuato almeno una corsa di lunghezza superiore a 50 km nell'ultimo mese. Esprimere la stessa interrogazione in SQL.

[1] Un attributo annullabile è indicato con la notazione A*. I vincoli di chiave esterna sono indicati con la notazione A$_{fk}$; la relazione a cui fa riferimento il vincolo è ricavabile facilmente dal nome dell'attributo.

Es: 1)
Scrivere l'istruzione DDL per la definizione della relazione turni includendo, oltre ai vincoli indicati nello schema, il vincolo che impone che nell'attributo 'orario' sia memorizzato uno dei valori dell'insieme {0, 6, 12, 18}

```
create table turni (
        taxi varchar(20) not null references taxi(targa),
        giorno date not null,
        orario numeric(2) not null,
        autista varchar(8) not null,
        foreign key (autista) references autista(matricola),
        primary key (taxi, giorno, orario),
        check( orario in (0,6,12,18) )
);
```

Es: 2)
Aggiungere alla tabella autisti il vincolo che impedisce a due autisti di avere lo stesso numero di patente.

```
alter table autisti
add constraint patente_unica unique( numero_patente );
```

Es: 3)
Eliminare dal database tutti i taxi che soddisfano tutte le seguenti condizioni: l'anno di immatricolazione è precedente al 2010; la data di scadenza della revisione è passata da almeno due anni; non sono stati utilizzati in alcun turno dalla data di scadenza della revisione fino alla data odierna.

```
delete from taxi
where anno_immatricolazione < 2010
        and current_date - scadenza_revisione >= 365 * 2
        and targa not in (  select taxi
                            from turni
                            where giorno >= scadenza_revisione
                            and giorno <= current_date
                          );
```

Es: 4)
Elencare gli autisti che, pur avendo effettuato almeno un turno nell'ultima settimana, non hanno effettuato alcuna corsa durante tutti i turni dell'ultima settimana.

```
select distinct a.matricola, a.cognome_nome
from autisti a, turni t
where a.matricola = t.autista
        and (current_date - t.giorno) between 0 and 6
        and not exists (
```

```
select *
from corse c, turni t2
where c.taxi = t2.taxi
        and t.giorno = t2.giorno
        and c.orario = t2.orario
        and (current_date - c.giorno) between 0 and 6
        and t2.autista = a.autista
);
```

Es: 5)

Definire la vista relazionale contanti per turno, che associa ad ogni turno la somma degli importi delle corse effettuate durante quel turno che sono state pagate in contanti. Oltre ai dati suddetti, la vista deve includere la matricola e nome dell'autista che ha effettuato il turno, nonché la città della sede del taxi.

```
create or replace view contanti_per_turno
        ( taxi, giorno, orario, importo_contanti, matricola_autista, nome_autista, città_taxi)
as
        select c.taxi, c.giorno, c.orario, sum(c.importo), a.matricola, a.cognome_nome, s.citta
        from corse c, turni t, autista a, taxi tx, sedi s
        where c.taxi = t.taxi
                and c.giorno = t.giorno
                and c.orario = t.orario
                and t.autista = a.matricola
                and t.taxi = tx.targa
                and tx.sede = s.codice
                and c.contanti
        group by c.taxi, c.giorno, c.orario, a.matricola, a.cognome_nome, s.citta;
```

Es: 6)

Elencare gli autisti che hanno effettuato almeno 100 turni, ordinandoli in base all'importo totale incassato nelle corse effettuate (a parità di importo incassato, usare l'ordinamento in base al nome dell'autista).

```
select a.matricola, a.cognome_nome, sum(c.importo) as importo_totale,
from autisti a, turni t, corse c
where a.matricola = t.autista
        and t.taxi = c.taxi
        and t.giorno = c.giorno
        and t.orario = c.orario
group by a.matricola, a.cognome_nome
having count(distinct t.taxi, t.giorno, t.orario) >= 100
order by importo_totale desc, a.cognome_nome
```

Es: 7)

Usando l'algebra relazionale, estrarre i dati degli autisti che hanno effettuato almeno una corsa di lunghezza superiore a 50 km nell'ultimo mese. Esprimere la stessa interrogazione in SQL.

A := AUTISTI
T := TURNI
C := SEL_{km > 50 and (giorno-current_date) <= 30} (CORSE)

```
PROJ_{A.*} (
        A
                JOIN_{A.matricola=T.autista}
        T
                JOIN_{T.taxi and T.giorno = C.giorno and T.orario = C = orario}
        C
)
```