MONDIALI DI CALCIO

Squadra (Nazione, Anno, Allenatore, PosizioneInClassifica)
Organizzazione (Anno, Nazione)
Giocatore (ID, Nome)
Partecipazione (IDGioc, Anno, Nazione, Ruolo, GoalFatti)

Scrivere il comando SQL per creare la tabella PARTECIPAZIONE, effettuando opportune e ragionevoli ipotesi su domini, vincoli e reazioni ai cambiamenti.

```
create table Partecipazione (
   IDGiocatore integer
      references Giocatore(ID)
         on update cascade on delete no action,
   Anno integer > 1900
      references Squadra(Anno)
         on update cascade on delete no action,
   Nazione varchar(32) not null
      references Squadra(Nazione)
         on update cascade on delete no action,
   Ruolo varchar(16),
   GoalSegnati integer >=0 default 0,
   primary key(IDGiocatore, Anno)    )
```

OK ?

Esempio di istanza

Squadra (Anno, Nazione, Allenatore, PosizioneInClassifica)

1982, Italia, …, 1
1998, Croazia, …, X
2006, Francia, …, 2
2006, Italia, …, 1

Partecipazione (IDGioc, Anno, Nazione, Ruolo, GoalFatti)

[zndzdn, 2006, Francia, *ariete*?, …]
[chzcivic, 1982, Croazia, *profeta*?, …]

```
create table Partecipazione (
   IDGiocatore integer
      references Giocatore(ID)
        on update cascade on delete no action,
   Anno integer > 1900,
   Nazione varchar(32) not null,
   Ruolo varchar(16),
   GoalSegnati integer >=0 default 0,
   primary key(IDGiocatore, Anno),
   foreign key (Anno, Nazione)
     references Squadra(Anno, Nazione)
        on update cascade on delete no action  )
```

Estrarre il nome delle Nazioni che non hanno mai vinto ***il*** *mondiale organizzato da loro.*

Squadra (Nazione, Anno, Allenatore,PosInClass)
Organizzazione (Anno, Nazione)
Giocatore (ID, Nome)
Partecipazione (IDGioc,Anno,Nazione,Ruo,GoalF)

```
select S.Nazione
from Squadra S join Organizzazione O
        on S.Nazione = O.Nazione
where S.Anno = O.Anno and
      PosizioneInClassifica <> 1
```

*Estrarre il nome delle Nazioni che non hanno mai vinto **il** mondiale organizzato da loro (3 min).*

```
select Nazione
from Organizzazione O
where Nazione not in
      ( select Nazione
        from Squadra
        where Anno = O.Anno
          and PosizioneInClassifica = 1 )
```

Squadra (Nazione, Anno, Allenatore,PosInClass)
Organizzazione (Anno, Nazione)
Giocatore (ID, Nome)
Partecipazione (IDGioc,Anno,Nazione,Ruo,GoalF)

Estrarre il nome delle Nazioni che non hanno mai vinto ***il*** *mondiale organizzato da loro.*

```
select Nazione
from Squadra
except
select S.Nazione
from Squadra S join Organizzazione O on
        (S.Nazione,S.Anno) = (O.Nazione,O.Anno)
where PosizioneInClassifica = 1
```

Squadra (Nazione, Anno, Allenatore,PosInClass)
Organizzazione (Anno, Nazione)
Giocatore (ID, Nome)
Partecipazione (IDGioc,Anno,Nazione,Ruo,GoalF)

*Estrarre il nome delle Nazioni che non hanno mai vinto **UN** mondiale organizzato da loro.*

```
select Nazione
from Squadra
  except
select S.Nazione
from Squadra S join Organizzazione O on
          (S.Nazione,S.Anno) = (O.Nazione,O.Anno)
where PosizioneInClassifica = 1
```

Squadra (Nazione, Anno, Allenatore,PosInClass)
Organizzazione (Anno, Nazione)
Giocatore (ID, Nome)
Partecipazione (IDGioc,Anno,Nazione,Ruo,GoalF)

..equivalente a :

```
select Nazione
from Squadra
where Nazione not in ( select Nazione
                       from Squadra natural join Organizzazione
                       where PosizioneInClassifica = 1 )
```

Estrarre il nome delle Nazioni che non hanno mai vinto il mondiale organizzato da loro (3 min).

Oppure possiamo sfruttare il raggruppamento!

Squadra (Nazione, Anno, Allenatore,PosInClass)
Organizzazione (Anno, Nazione)
Giocatore (ID, Nome)
Partecipazione (IDGioc,Anno,Nazione,Ruo,GoalF)

```
select S.Nazione
from Squadra S join Organizzazione O
      on (S.Nazione,S.Anno)=(O.Nazione,O.Anno)
group by S.Nazione
having min(PosizioneInClassifica) > 1
```

Determinare, per ogni campionato mondiale, la Nazionale che ha convocato il numero più elevato di attaccanti

```
select Anno, Nazione, count(*) as NumAttConvocati
from Partecipazione P
where Ruolo = "Attaccante"
group by Anno, Nazione
having count(*) >= all ( select count(*)
                         from Partecipazione
                         where Anno = P.Anno
                           and Ruolo="Attaccante"
                         group by Nazione )
```

La query esterna raggruppa le partecipazioni di ogni anno ad ogni squadra, mentre la query annidata confronta il conteggio di ogni nazionale soltanto con i conteggi delle nazionali della stessa edizione (**`Anno = P.Anno`**)

ATTENZIONE !! *Non si può fare:*

```
select Anno, Nazione, count(*) as NumAttConvocati
from Partecipazione P
where Ruolo = "Attaccante"
group by Anno, Nazione
having count(*) >= all ( select count(*)
                         from Partecipazione
                         where Ruolo="Attaccante"
                         group by Anno, Nazione )
```

QUESTA VERSIONE CONFRONTA
OGNI AGGREGATO CON IL MASSIMO
DI TUTTA LA STORIA DEI MONDIALI

In alternativa, con una vista intermedia:

```
create view NumAttConv(Ediz, Squadra, AttConv) as
  select Anno, Nazione, count(*)
  from Partecipazione
  where Ruolo = "Attaccante"
  group by Anno, Nazione

select Ediz, Squadra, AttConv
from NumAttConv N
where AttConv = ( select max(AttConv)
                  from NumAttConv
                  where Ediz = N.Ediz )
```

Estrarre i nomi dei giocatori che hanno partecipato a 3 edizioni diverse del mondiale oppure che hanno partecipato con più di una Nazionale.

```
select Nome
from Giocatore G
where 3 = ( select count(*)
            from Partecipazione
            where IDGiocatore = G.ID )
   or 1 < ( select count(distinct Nazione)
            from Partecipazione
            where IDGiocatore = G.ID )
```

Estrarre i nomi dei giocatori che hanno partecipato a 3 edizioni diverse del mondiale oppure che hanno partecipato con più di una Nazionale.

```
select Nome
from Giocatore
where ID in ( select IDGiocatore
              from Partecipazione
              group by IDGiocatore
              having count(*) = 3 or
                     count(distinct Nazione) > 1 )
```

ELECTRIC CARS

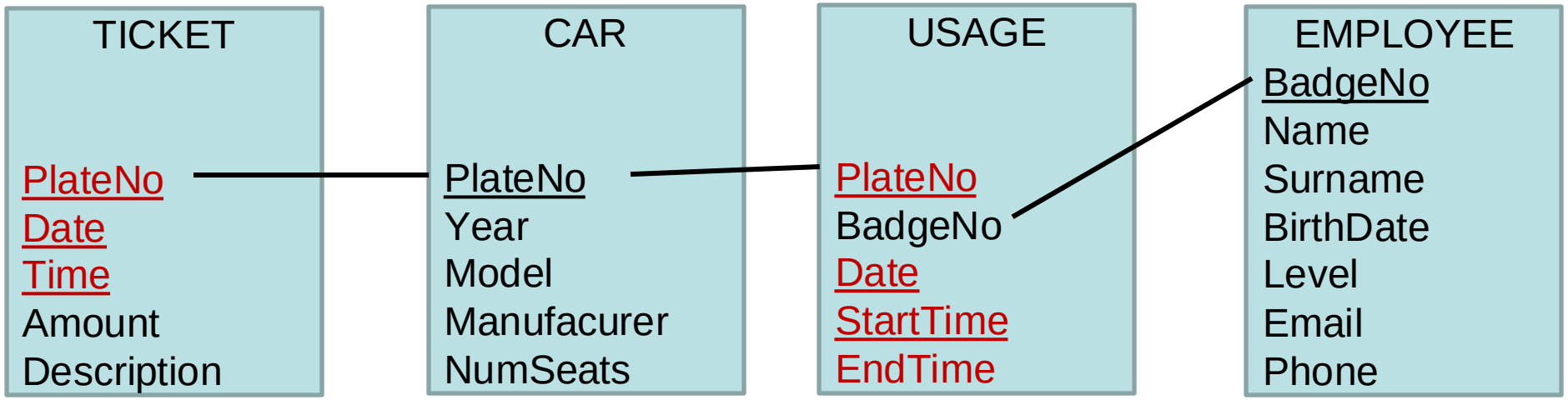
CAR (PLATENO, YEAR, MODEL, MANUFACTURER, NUMSEATS)

EMPLOYEE (BADGENO, NAME, SURNAME, BIRTHDATE, LEVEL, EMAIL, PHONE)

USAGE (PLATENO, BADGENO, DATE, STARTTIME, ENDTIME)

TICKET (PLATENO, DATE, TIME, AMOUNT, DESCRIPTION)

Lo schema sopra descrive l'utilizzo di auto elettriche degli impiegati di un'azienda di Milano, per spostamenti giornalieri durante le ore di ufficio. Le auto sono ritirate nel garage dell'azienda avvicinando il badge al lettore sulla porta e vengono considerate restituite quando un sensore le registra al rientro nel garage. Se i guidatori violano regole del codice della strada e ricevono delle multe mentre usano l'auto, l'ammontare corrispondente viene sottratto direttamente dal loro stipendio successive.



CAR (PLATENO, YEAR, MODEL, MANUFACTURER, NUMSEATS)

EMPLOYEE (BADGENO, NAME, SURNAME, BIRTHDATE, LEVEL, EMAIL, PHONE)

USAGE (PLATENO, BADGENO, DATE, STARTTIME, ENDTIME)

TICKET (PLATENO, DATE, TIME, AMOUNT, DESCRIPTION)

1. Scrivere le istruzioni di creazione delle tabelle **USAGE** e **TICKET**, definendo ragionevoli vincoli di tupla e di dominio ed esprimendo opportuni vincoli di integrità referenziale (foreign keys) verso altre tabelle.

```
create table USAGE (
        PlateNo   char(7) references Car(PlateNo)
                                on delete no action on update cascade,
        BadgeNo          integer references Employee(BadgeNo)
                                on delete no action on update cascade,
        Date             date,
        StartTime        time,
        EndTime time,
        primary key( PlateNo, Date, StartTime )
)
```

```
create table TICKET (
        PlateNo   char(7)
            references Car(PlateNo) on delete no action on update cascade,
        Date                date,
        Time                time,
        Amount   decimal(8,2),
                --- ignorando le raccomandazioni di GAAP di avere 13,4
        Description         varchar(128),
        primary key( PlateNo, Date, Time )
)
```

CAR (PLATENO, YEAR, MODEL, MANUFACTURER, NUMSEATS)
EMPLOYEE (BADGENO, NAME, SURNAME, BIRTHDATE, LEVEL, EMAIL, PHONE)
USAGE (PLATENO, BADGENO, DATE, STARTTIME, ENDTIME)
TICKET (PLATENO, DATE, TIME, AMOUNT, DESCRIPTION)

2. Estrarre Name e Surname dell'impiegato che, nel 2018, ha ricevuto multe con l'ammontare massimo di tutta l'azienda

```
select Name, Surname  from Employee
where BadgeNo in (
    select BadgeNo
    from USAGE U join TICKET T on ( U.PlateNo, U.Date ) = ( T.PlateNo, T.Date )
    where year(U.Date) = 2018 and T.Time between U.StartTime and U.EndTime
    group by BadgeNo
    having sum(Amount) >= ALL
        (   select sum(Amount)
            from USAGE U join TICKET T on ( U.PlateNo,U.Date ) = ( T.PlateNo, T.Date )
            where year(U.Date) = 2018 and T.Time between U.StartTime and U.EndTime
            group by BadgeNo ) )
```

```
create view TICKETAMOUNT2108( Bdg, Tot ) as
        (       select BadgeNo, sum(Amount)
                from USAGE U join TICKET T on (U.PlateNo,U.Date) = (T.PlateNo, T.Date)
                where year(U.Date) = 2018 and T.Time between U.StartTime and U.EndTime
                group by BadgeNo )

select Name, Surname
from EMPLOYEE
where BadgeNo in  ( select Bdg
                                from TICKETAMOUNT2018
                                where Tot =  (       select max(Tot)
                                                from TICKETAMOUNT2018  ) )
```

CAR (PLATENO, YEAR, MODEL, MANUFACTURER, NUMSEATS)
EMPLOYEE (BADGENO, NAME, SURNAME, BIRTHDATE, LEVEL, EMAIL, PHONE)
USAGE (PLATENO, BADGENO, DATE, STARTTIME, ENDTIME)
TICKET (PLATENO, DATE, TIME, AMOUNT, DESCRIPTION)

3. Estrarre gli impiegati di livello “Product Manager” che non hanno mai utilizzato la stessa auto due volte.

Restringendo la selezione agli impiegati che hanno guidato almeno una volta

```
select BadgeNo
from EMPLOYEE natural join USAGE
where Level = “Product Manager”
group by BadgeNo
having count(*) = count( distinct PlateNo )
```

Includendo anche gli impiegati che non hanno mai guidato una macchina:

```
select *
from EMPLOYEE
where Level = “Product Manager”
  and BadgeNo not in   (        select BadgeNo
                                from USAGE
                                group by BadgeNo, PlateNo
                                having count(*) > 1  )
```

Con un outer join possiamo includere anche i product manager che non guidano, seguendo lo «stile» della soluzione precedente:

```
select BadgeNo
from EMPLOYEE E left join USAGE U on E.BadgeNo = U.BadgeNo
where Level = “Product Manager”
group by E.BadgeNo
having count( PlateNo ) = count( distinct PlateNo )
```

CAR (PLATENO, YEAR, MODEL, MANUFACTURER, NUMSEATS)

EMPLOYEE (BADGENO, NAME, SURNAME, BIRTHDATE, LEVEL, EMAIL, PHONE)

USAGE (PLATENO, BADGENO, DATE, STARTTIME, ENDTIME)

TICKET (PLATENO, DATE, TIME, AMOUNT, DESCRIPTION)

4. Esprimere il vincolo che controlla the non ci sono due utilizzi distinti di una stessa auto che si sovrappongano nel tempo

Intervalli disgiunti:

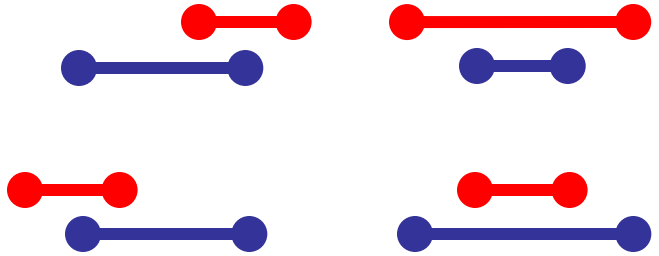
A.End < B.Start OR B.End < A.Start

Intervalli sovrapposti =

NOT ***disgiunti*** =

NOT (A.End < B.Start OR B.End < A.Start) =

A.End >= B.Start AND B.End >= A.Start



```
create assertion NOOVERLAP as check(
    not exists  (   select *
                    from USAGE A join USAGE B
                            on A.PlateNo = B.PlateNo
                    where A.Date = B.Date
                        and A.StartTime > B.StartTime
                        and A.EndTime >= B.StartTime
                        and B.EndTime >= A.StartTime    )    )
```

Questa è una soluzione più performante rispetto a
and A.StartTime <> B.StartTime
dato che si dimezzano i casi da considerare, senza ledere la generalità

Discussione su necessità di aggiungere una condizione che impedica di confrontare uno Usage con se stesso. In casi simili, si impone una diseguaglianza sull'attributo che identifica la tabella. In questo caso la chiave primaria è composta da tre attributi: PlateNo, Date, e StartTime. Basterebbe chiedere che almeno uno dei tre fosse diverso, ma siccome A.PlateNo = B.PlateNo e A.Date = B.Date per la richiesta dell'esercizio, l'unica possibilità è quella di imporre la diseguaglianza tra gli orari di inizio degli intervalli.
Qualcuno potrebbe obiettare che, così facendo, l'assertion non permetterebbe di escludere l'inserimento di, ad esempio, ('AB123CD', 1234, 2021-01-01, 15:00, 16:00) e ('AB123CD', 1234, 2021-01-01, 15:00, 18:00), ma questa evenienza è già esclusa dall'unicità della chiave primaria, che non permette inserimento di tuple con pari targa, data e ora di inizio.

Alternativa che sfrutta la simmetria:

```
create assertion NoOverlap as check(
    not exists  (   select *
                    from Usage A join Usage B
                            on A.PlateNo = B.PlateNo
                     where A.Date = B.Date
                        and A.StartTime <> B.StartTime
                        and A.StartTime between B.StartTime and B.EndTime)    )
```

CAR (PLATENO, YEAR, MODEL, MANUFACTURER, NUMSEATS)

EMPLOYEE (BADGENO, NAME, SURNAME, BIRTHDATE, LEVEL, EMAIL, PHONE)

USAGE (PLATENO, BADGENO, DATE, STARTTIME, ENDTIME)

TICKET (PLATENO, DATE, TIME, AMOUNT, DESCRIPTION)

5. Calcolare, per ogni auto, il numero medio di utilizzi giornalieri e la durata media di questi utilizzi. **Si consideri che le giornate in cui una specifica auto non è stata utilizzata non devono contribuire al calcolo della media (non della durata media, ma solo al numero di utilizzi, contando quel giorno). Inoltre, per semplicità, si assuma che la differenza tra due timestamps ritorni direttamente un intervallo.**

(a)

```
select PlateNo,
        count(*)/count(distinct Date) as AvgNumOfDailyUses,
        avg(EndTime –StartTime) as AvgUseDuration
from USAGE
group by PlateNo
```

Questa non è l'interpretazione corretta, dato che considera solo, per ogni auto, I giorni in cui quell'auto è stata utilizzata almeno una volta, e non –invece– I giorni in cui l'auto non è stata proprio utilizzata. Paradossalmente, un'ipotetica atuo usata solo una volta in tanti anni (un giorno solo!) otterrebbe un valore medio di utilizzi giornalieri = 1…. che non è proprio il risultato desiderato.

*(**b**) Una formulazione possibile calcola la media basandosi sull'attività giornaliera:*

```
create view DAILYACTIVITY( PlNo, Day, NumUs, TotTime ) as
  select PlateNo, Date, count(*), sum(EndTime –StartTime)
  from USAGE
  group by PlateNo, Date

select PlateNo, avg(NumUs) as AvgDailyUses, sum(TotTime)/sum(NumUs) as AvgUseDuration
from DAILYACTIVITY
group by PlateNo
```

*(**c**) Comunque, nella versione (b), i giorni in cui una data auto non è stata usata non sono considerati quando si fa la media della durata per quell'auto. Un modo "semplice" di contare tutti i giorni è il seguente:*

```
create view DAYSOFOPENING( NumDays ) as
  select count( distinct Date )    assuming that no working day ends with zero cars being used overall
  from USAGE

select PlateNo, count(*)/NumDays as AvgNumOfDailyUses, avg(EndTime –StartTime) as AvgUseDuration
from USAGE, DAYSOFOPENING    (not a join, but a "Cartesian product" with just one number)
group by PlateNo
```

Dove la view estrae semplicemente un numero (il numero di giorni "validi", che è ovviamente indipendente dall'auto specifica).

*(**d**) La view in (c) restituisce sicuramente solo un valore, quindi può essere direttamente inclusa nella lista target della Select, rendendo la definizione della view ridondante. Poi, possiamo anche aggiungere le auto che possibilmente non sono mai state usate (per qualsiasi ragione):*

```
select PlateNo,
        count( all StartTime ) / ( select count( distinct Date )
                                   from USAGE ) as AvgNumOfDailyUses,

        avg(EndTime –StartTime) as AvgUseDuration

from USAGE U right join CAR C on U.PlateNo = C.PlateNo
group by C.PlateNo
```

Si noti che in questa formulazione:

(i) il criterio di raggruppamento è C.PlateNo (si sceglie il PlateNo che non è mai nullo, cioè quello di Car, dato che invece U.PlateNo potrebbe essere eventualmente nullo);

(ii) Il count non è più count(*), perché conterebbe 1 anche per le auto che non sono mai state usate, ma è il count di un attributo della tabella USAGE *(un altro attributo, ad esempio BadgeNo, andrebbe bene lo stesso).*

HOTEL DI LUSSO

Il seguente schema rappresenta i dati relativi ai pernottamenti già conclusi dei clienti di un albergo aperto continuativamente dal 2004. Per semplicità, di ogni pernottamento si registrano i dati completi di uno solo degli occupanti della camera.
Le tuple nella tabella SOGGIORNO vengono sempre inserite all'inizio del soggiorno, lasciando gli ultimi due attributi NULL. Gli attributi vengono poi aggiornati con i valori corretti quando i clienti fanno il check out.

CLIENTE(CodiceFiscale, Nome, Indirizzo, Telefono, Email)
CAMERA(Numero, Piano, Tipo, PostiLetto, NumLettiSingoli, NumLettiMatrimoniali, Superficie, Descrizione)
SOGGIORNO(NumCamera, DataCkIn, DataCkOut, CodiceFiscale, CostoTotale)
MINIBAR(NumCamera, Data, ArticoloConsumato, Quantità)

CLIENTE(CodiceFiscale, Nome, Indirizzo, Telefono, Email)
CAMERA(Numero, Piano, Tipo, PostiLetto, NumLettiSingoli, NumLettiMatr, Superf, Descr)
SOGGIORNO(CodiceFiscale, NumCamera, DataCkIn, DataCkOut, CostoTotale)
MINIBAR(NumCamera, Data, ArticoloConsumato, Quantità)

```
create table SOGGIORNO (
        CodiceFiscale integer  references CLIENTE(CodiceFiscale)
                                on delete no action on update cascade,
        NumCamera  integer  references CAMERA(NumCamera)
                               on delete no action on update cascade,
        DataCkIn  Date not null,
        DataCkOut  Date  not null check (DataCkOut > DataCkIn),
        CostoTotale  decimal(8,2) > 0,
        PRIMARY KEY (NumCamera, DataCkIn)
)

create table MINIBAR (
        NumCamera  integer  references CAMERA(NumCamera)
                                on delete no action on update cascade,
    Data date  + qui va il check che un Soggiorno corrispondente deve esistere nella corrispondente Data e NumCamera
        ArticoloConsumato       varchar(100),
        Quantità integer  > 0,
        PRIMARY KEY (NumCamera, Data, ArticoloConsumato)
)
```

Notare che, per fare la join correttamente tra queste due tabelle, richiedere l'uguaglianza del numero della stanza non è sufficiente. Anche la Date deve cadere nell'intervallo corrispondente del soggiorno.

CLIENTE(CodiceFiscale, Nome, Indirizzo, Telefono, Email)
CAMERA(Numero, Piano, Tipo, PostiLetto, NumLettiSingoli, NumLettiMatr, Superf, Descr)
SOGGIORNO(CodiceFiscale, NumCamera, DataCkIn, DataCkOut, CostoTotale)
MINIBAR(NumCamera, Data, ArticoloConsumato, Quantità)

Costruire la classifica dei prodotti più prelevati dai minibar nel 2018 (dai più graditi ai meno graditi).

```
select ArticoloConsumato,
        sum(Quantità) as ItemTotaliConsumati
from Minibar
where Data between 1/1/2018 and 31/12/2018
group by ArticoloConsumato
order by 2 desc
```

CLIENTE(CodiceFiscale, Nome, Indirizzo, Telefono, Email)
CAMERA(Numero, Piano, Tipo, PostiLetto, NumLettiSingoli, NumLettiMatr, Superf, Descr)
SOGGIORNO(CodiceFiscale, NumCamera, DataCkIn, DataCkOut, CostoTotale)
MINIBAR(NumCamera, Data, ArticoloConsumato, Quantità)

Estrarre i nomi dei clienti che hanno consumato un Chinotto nel loro primo soggiorno in albergo.

SOL1:

```
select Nome
from Minibar natural join Soggiorno S natural join Cliente
where ArticoloConsumato = «Chinotto»
   and Data between DataCkIn and DataCkOut
   and DataCkIn = (
                        select min(DataCkIn)
                        from Soggiorno
                        where CodiceFiscale = S. CodiceFiscale
                        )
```

CLIENTE(CodiceFiscale, Nome, Indirizzo, Telefono, Email)
CAMERA(Numero, Piano, Tipo, PostiLetto, NumLettiSingoli, NumLettiMatr, Superf, Descr)
SOGGIORNO(CodiceFiscale, NumCamera, DataCkIn, DataCkOut, CostoTotale)
MINIBAR(NumCamera, Data, ArticoloConsumato, Quantità)

Estrarre i nomi dei clienti che hanno consumato un Chinotto nel loro primo soggiorno in albergo.

SOL2:

```
create view PrimoSoggiorno(CFCli, NumCam, PrimoCkIn, PrimoCkOut) as
  select CodiceFiscale, NumCamera, DataCkIn, DataCkOut
  from Soggiorno
  where (CodiceFiscale, DataCkIn) = (select CodiceFiscale, min(DataCkIn)
                                     from Soggiorno
                                        group by CodiceFiscale )

select Nome
from (Minibar natural join PrimoSoggiorno) join Customer on CFCli = CodiceFiscale
where ArticoloConsumato = «Chinotto»
    and Data between PrimoCkIn and PrimoCkOut
```

CLIENTE(CodiceFiscale, Nome, Indirizzo, Telefono, Email)
CAMERA(Numero, Piano, Tipo, PostiLetto, NumLettiSingoli, NumLettiMatr, Superf, Descr)
SOGGIORNO(CodiceFiscale, NumCamera, DataCkIn, DataCkOut, CostoTotale)
MINIBAR(NumCamera, Data, ArticoloConsumato, Quantità)

Di ciascun cliente che ha speso complessivamente più di 10.000 euro (in tutti e 18 gli anni di apertura) si indichi il numero totale di notti trascorse in albergo in soggiorni “recenti”, cioè iniziati dopo il 1/1/2021.

```
select CodiceFiscale,
        sum(DataCkOut – DataCkIn) as NottiRecentiDaNoi
from Soggiorno
where DataCkIn > 1/1/2021
    and CodiceFiscale in ( select CodiceFiscale
                           from Soggiorno
                           group by CodiceFiscale
                           having sum( CostoTotale )  > 10.000 )
group by CodiceFiscale
```

CLIENTE(CodiceFiscale, Nome, Indirizzo, Telefono, Email)
CAMERA(Numero, Piano, Tipo, PostiLetto, NumLettiSingoli, NumLettiMatr, Superf, Descr)
SOGGIORNO(CodiceFiscale, NumCamera, DataCkIn, DataCkOut, CostoTotale)
MINIBAR(NumCamera, Data, ArticoloConsumato, Quantità)

```
create assertion NessunoSnackPerFantasmi as check (
  not exists (
            select *
            from Minibar M
             where not exists (
                                select *
                                from Soggiorno
                                where NumCamera = M.NumCamera and
                        M.Data between DataCkIn and DataCkOut
                                )
 )

create assertion NessunoSnackPerFantasmi as check (
   not exists (
            select *
            from Minibar M
            where NumCamera not in ( select NumCamera
                                      from Soggiorno
                                      where M.Data between DataCkIn and DataCkOut
                                    )
)
```