

Esercizi SQL

Le cose che non vogliamo vedere

Target list **miste** quando non c'è la clausola group by
Attributi nella select o nella having che non siano **anche** nella
group by (quando c'è una clausola group by)

Aggregati di aggregati

Aggregati nella clausola where ["WHERE max(X)"]

HAVING max(X). → *max non è un predicato!!*

Clausole where auto-contraddittorie

["WHERE anno=1992 and anno=1993"]

IN / NOT IN con

Niente a sinistra ["WHERE NOT IN ..."]

Schemi che non si corrispondono

Predicati con query nidificate a dx senza ANY o ALL

Aeroporti

AEROPORTO (Città, Nazione, NumPiste)

VOLO (IdVolo, GiornoSett, CittàPart, OraPart,
CittàArr, OraArr, TipoAereo)

AEREO (TipoAereo, NumPasseggeri, QtaMerci)

Rinfreschiamoci la memoria

Trovare le città da cui partono voli diretti a Roma,
ordinate alfabeticamente

Rinfreschiamoci la memoria

Trovare le città da cui partono voli diretti a Roma,
ordinate alfabeticamente

```
SELECT DISTINCT CittàPar  
FROM Volo  
WHERE CittàArr= 'Roma'  
ORDER BY CittàPar
```

Rinfreschiamoci la memoria

Trovare le città da cui partono voli diretti a Roma,
ordinate alfabeticamente

```
SELECT DISTINCT CittàPar  
FROM Volo  
WHERE CittàArr= 'Roma'  
ORDER BY CittàPar
```

Trovare le città con un aeroporto di cui non è noto
il numero di piste

Rinfreschiamoci la memoria

Trovare le città da cui partono voli diretti a Roma,
ordinate alfabeticamente

```
SELECT DISTINCT CittàPar  
FROM Volo  
WHERE CittàArr= 'Roma'  
ORDER BY CittàPar
```

Trovare le città con un aeroporto di cui non è noto
il numero di piste

```
SELECT Città  
FROM Aeroporto  
WHERE NumPiste IS NULL
```

**Di ogni volo misto (merci e passeggeri) estrarre
il codice e i dati relativi al trasporto**

**Di ogni volo misto (merci e passeggeri) estrarre
il codice e i dati relativi al trasporto**

```
SELECT IdVolo, NumPasseggeri, QtaMerci  
FROM VOLO AS V, AEREO AS A  
WHERE V.TipoAereo = A.TipoAereo and  
       NumPasseggeri > 0 and QtaMerci > 0
```

**Di ogni volo misto (merci e passeggeri) estrarre
il codice e i dati relativi al trasporto**

```
SELECT IdVolo, NumPasseggeri, QtaMerci  
FROM VOLO AS V, AEREO AS A  
WHERE V.TipoAereo = A.TipoAereo and  
        NumPasseggeri > 0 and QtaMerci > 0
```

(sintassi equivalente)

```
SELECT IdVolo, NumPasseggeri, QtaMerci  
FROM VOLO V JOIN AEREO A  
ON V.TipoAereo = A.TipoAereo  
WHERE    NumPasseggeri > 0 and QtaMerci > 0
```

Le nazioni di partenza e arrivo del volo *AZ274*

Le nazioni di partenza e arrivo del volo AZ274

```
SELECT A1.Nazione, A2.Nazione
FROM (AEROPORTO A1 JOIN VOLO
      ON A1.Città=CittàArr)
     JOIN AEROPORTO A2
      ON CittàPar=A2.Città
WHERE IdVolo= 'AZ274'
```

**Trovare l' aeroporto italiano con il
maggior numero di piste**

**Trovare l' aeroporto italiano con il
maggior numero di piste**

```
SELECT Città, max(NumPiste)
FROM      AEROPORTO
WHERE     Nazione = 'Italia'
```

**Trovare l' aeroporto italiano con il
maggior numero di piste (errore sintattico)**

```
SELECT Città, max(NumPiste)
FROM      AEROPORTO
WHERE     Nazione = 'Italia'
```

NO!

**Trovare l' aeroporto italiano con il
maggior numero di piste**

```
SELECT Città, max(NumPiste)
FROM      AEROPORTO
WHERE     Nazione = 'Italia'
GROUP BY Città
```


**Trovare l' aeroporto italiano con il
maggior numero di piste (errore semantico)**

```
SELECT Città, max(NumPiste)
FROM AEROPORTO
WHERE Nazione = 'Italia'
GROUP BY Città
```

NO!

Trovare l' aeroporto italiano con il maggior numero di piste (soluzione corretta)

Ad esempio si può usare una query annidata

```
SELECT Città, NumPiste
FROM AEROPORTO
WHERE Nazione= 'Italia' and
      NumPiste = (SELECT max(numPiste)
                  FROM AEROPORTO
                  WHERE Nazione= 'Italia' )
```

Trovare l' aeroporto italiano con il maggior numero di piste (soluzione corretta)

oppure

```
SELECT Città, NumPiste
FROM     AEROPORTO
WHERE    Nazione= 'Italia' and
        NumPiste >= ALL
        (SELECT numPiste
         FROM AEROPORTO
         WHERE Nazione= 'Italia' )
```

**Per ogni nazione, trovare quante piste ha
l' aeroporto con più piste.**

**Per ogni nazione, trovare quante piste ha
l' aeroporto con più piste.**

```
SELECT Nazione, max(NumPiste)  
FROM AEROPORTO  
GROUP BY Nazione
```

Per ogni nazione, trovare quante piste ha l' aeroporto con più piste (purché almeno 3).

```
SELECT Nazione, max(NumPiste)
```

```
FROM AEROPORTO
```

```
GROUP BY Nazione
```

.....

Per ogni nazione, trovare quante piste ha l' aeroporto con più piste (purché almeno 3).

```
SELECT Nazione, max(NumPiste)
FROM AEROPORTO
GROUP BY Nazione
HAVING max(NumPiste) > 2
```

Dobbiamo raggruppare tutte le tuple e poi considerare solo i gruppi di tuple (a pari nazione) in cui il massimo numero di piste sia almeno 3

Per ogni nazione, trovare quante piste ha l' aeroporto con più piste (purché almeno 3).

```
SELECT Nazione, max(NumPiste)
FROM AEROPORTO
WHERE NumPiste > 2
GROUP BY Nazione
```

Soluzione alternativa: scarta subito tutte le tuple che non abbiano almeno tre piste; poi raggruppa solo quelle, e considera tutti i gruppi, ma chiaramente l'effetto è lo stesso

PER INCLUDERE LA CITTA' BISOGNA CAMBIARE STRATEGIA

Trovare le città in cui si trovano gli aeroporti con più piste di ogni nazione

indicare città, nazione e numero di piste
(ancora col vincolo che siano almeno 3)

Trovare le città in cui si trovano gli aeroporti con più piste di ogni nazione

indicare città, nazione e numero di piste
(*ancora col vincolo che siano almeno 3*)

```
SELECT *  
FROM AEROPORTO  
WHERE ( Nazione, NumPiste ) IN  
      (SELECT Nazione, max(NumPiste)  
       FROM AEROPORTO  
       GROUP BY Nazione  
       HAVING max(NumPiste) > 2)
```

Trovare le città in cui si trovano gli aeroporti con più piste di ogni nazione

indicare città, nazione e numero di piste
(*ancora col vincolo che siano almeno 3*)

SELECT *

FROM AEROPORTO **A1**

WHERE **NumPiste** IN

(SELECT max(NumPiste)

FROM AEROPORTO **A2**

WHERE **A2.Nazione**= **A1.Nazione**

GROUP BY Nazione

HAVING max(NumPiste) > 2)

Trovare gli aeroporti da cui partono voli *internazionali*

Trovare gli aeroporti da cui partono voli *internazionali*

```
SELECT  DISTINCT CittàPar
FROM    (AEROPORTO AS A1 JOIN VOLO
        ON CittàPar = A1.Città)
        JOIN AEROPORTO AS A2
        ON CittàArr = A2.Città
WHERE   A1.Nazione <> A2.Nazione
```

Trovare gli aeroporti da cui partono voli *internazionali*

```
SELECT  DISTINCT CittàPar
FROM    (AEROPORTO AS A1 JOIN VOLO
        ON CittàPar = A1.Città)
        JOIN AEROPORTO AS A2
        ON CittàArr = A2.Città
WHERE   A1.Nazione <> A2.Nazione
```

Il distinct è essenziale per la chiarezza e leggibilità del risultato

Trovare il numero **totale** di partenze internazionali (*del giovedì*) da tutti gli aeroporti

```
SELECT ?  
FROM (AEROPORTO AS A1 JOIN VOLO  
ON CittàPar=A1.Città)  
JOIN AEROPORTO AS A2  
ON CittàArr=A2.Città  
WHERE A1.Nazione <> A2.Nazione  
and GiornoSett = 'Giovedì'
```

Trovare il numero **totale** di partenze internazionali (*del giovedì*) da tutti gli aeroporti

```
SELECT count(*)
FROM (AEROPORTO AS A1 JOIN VOLO
ON CittàPar=A1.Città)
JOIN AEROPORTO AS A2
ON CittàArr=A2.Città
WHERE A1.Nazione <> A2.Nazione
and GiornoSett = 'Giovedì'
```

qui niente distinct!

Trovare il numero di aeroporti che hanno almeno una partenza internazionale (*al giovedì*)

```
SELECT ?  
FROM (AEROPORTO AS A1 JOIN VOLO  
ON CittàPar=A1.Città)  
JOIN AEROPORTO AS A2  
ON CittàArr=A2.Città  
WHERE A1.Nazione <> A2.Nazione  
and GiornoSett = 'Giovedì'
```

Trovare il numero di aeroporti che hanno almeno una partenza internazionale (*al giovedì*)

```
SELECT count( distinct CittàPar )
FROM (AEROPORTO AS A1 JOIN VOLO
ON CittàPar=A1.Città)
JOIN AEROPORTO AS A2
ON CittàArr=A2.Città
WHERE A1.Nazione <> A2.Nazione
and GiornoSett = 'Giovedì'
```

Trovare il numero di partenze internazionali (*del giovedì*) **da ogni aeroporto**

```
SELECT      ?  
FROM        (AEROPORTO AS A1 JOIN VOLO  
            ON CittàPar=A1.Città)  
            JOIN AEROPORTO AS A2  
            ON CittàArr=A2.Città  
WHERE       A1.Nazione <> A2.Nazione  
            and GiornoSett = 'Giovedì'
```

?

Trovare il numero di partenze internazionali (*del giovedì*) **da ogni aeroporto**

```
SELECT  CittàPar, count(*) AS NumPartInt
FROM    (AEROPORTO AS A1 JOIN VOLO
        ON CittàPar=A1.Città)
        JOIN AEROPORTO AS A2
        ON CittàArr=A2.Città
WHERE   A1.Nazione <> A2.Nazione
        and GiornoSett = 'Giovedì'

GROUP BY CittàPar
```

Le città francesi da cui *ogni settimana* partono più di 20 voli diretti x la Germania

```
SELECT CittàPar, count(*) AS NumVoliGer
FROM (AEROPORTO AS A1 JOIN VOLO
      ON CittàPar=A1.Città)
      JOIN AEROPORTO AS A2
      ON CittàArr=A2.Città
WHERE A1.Nazione= 'Francia' AND
      A2.Nazione= 'Germania'
GROUP BY CittàPar
```

?

.....

Le città francesi da cui *ogni settimana* partono più di 20 voli diretti x la Germania

```
SELECT CittàPar, count(*) AS NumVoliGer
FROM (AEROPORTO AS A1 JOIN VOLO
      ON CittàPar=A1.Città)
      JOIN AEROPORTO AS A2
      ON CittàArr=A2.Città
WHERE A1.Nazione= 'Francia' AND
      A2.Nazione= 'Germania'
GROUP BY CittàPar
HAVING count(*) > 20
```

Trovare il numero di **voli del giovedì** di ogni aeroporto *da cui partano almeno 100 voli a settimana*

SELECT CittàPart, **count(*)**

FROM VOLO

WHERE *GiornoSett* = **'Giovedì'** ?

GROUP BY CittàPart

HAVING **count(*)** >= 100

Trovare il numero di **voli del giovedì** di ogni aeroporto *da cui partano almeno 100 voli a settimana*

```
SELECT    CittàPart, count(*)  
FROM      VOLO  
WHERE     GiornoSett = 'Giovedì'  
GROUP BY CittàPart  
HAVING    count(*) >= 100
```

Il secondo conteggio deve avvenire su **tutti** i voli dell' aeroporto, non solo su quelli del giovedì

```
SELECT    CittàPart, count(*)  
FROM      VOLO  
WHERE     GiornoSett = 'Giovedì' AND CittàPart IN  
          ( SELECT CittàPart FROM VOLO  
            GROUP BY CittàPart HAVING count(*) >= 100 )  
GROUP BY CittàPart
```


Filmografie

REGISTA (Nome, DataNascita, Nazionalità)

ATTORE (Nome, DataNascita, Nazionalità)

INTERPRETA (Attore, Film, Personaggio)

FILM (Titolo, NomeRegista, Anno)

PROIEZIONE (NomeCin, CittàCin, TitoloFilm)

CINEMA (Città, NomeCinema, #Sale, #Posti)

Selezionare le Nazionalità dei registi che hanno diretto qualche film nel 1992 **ma non** hanno diretto **alcun** film nel 1993

Selezionare le Nazionalità dei registi che hanno diretto qualche film nel 1992 **ma non** hanno diretto **alcun** film nel 1993

```
SELECT DISTINCT Nazionalità
FROM REGISTA
WHERE Nome IN
( SELECT NomeRegista
  FROM FILM WHERE Anno= '1992' )
AND Nome NOT IN
( SELECT NomeRegista
  FROM FILM WHERE Anno= '1993' )
```

Selezionare le Nazionalità dei registi che hanno diretto qualche film nel 1992 **ma non** hanno diretto **alcun** film nel 1993

```
SELECT DISTINCT Nazionalità
FROM REGISTA, FILM
WHERE Nome = NomeRegista AND Anno= '1992'
AND Nome NOT IN
(SELECT NomeRegista
FROM FILM WHERE Anno= '1993' )
```

Nazionalità dei registi con film nel 1992 **ma non** nel 1993
(soluzione alternativa)

Si può usare EXCEPT (a patto di discriminare in base alla chiave)

SELECT Nazionalità

FROM REGISTA

WHERE Nome IN

(SELECT NomeRegista

FROM FILM WHERE Anno = 1992

EXCEPT

SELECT NomeRegista

FROM FILM WHERE Anno = 1993)

NON si può usare la EXCEPT direttamente se nella target list non è incluso l' attributo discriminante per l' esclusione

~~SELECT Nazionalità~~

~~FROM FILM JOIN REGISTA ON NomeRegista=Nome~~

~~WHERE Anno = 1992~~

~~EXCEPT~~

~~SELECT Nazionalità~~

~~FROM FILM JOIN REGISTA ON NomeRegista=Nome~~

~~WHERE Anno = 1993~~

Attenzione:

in SQL gli operatori insiemistici eliminano i duplicati (come se davanti a Nazionalità ci fosse distinct)

Nazionalità dei registi con film nel 1992 **ma non** nel 1993
(errore tipico)

SBAGLIATO ricorrere ad un JOIN con
condizione nella WHERE:

~~SELECT Nazionalità~~

~~FROM Regista JOIN Film~~

~~ON Nome = NomeRegista~~

~~WHERE Anno = 1992 AND Anno <> 1993~~

perché la WHERE agisce a livello di TUPLA

Nomi dei registi che hanno diretto nel 1993 più film di quanti ne avevano diretti nel 1992

Nomi dei registi che hanno diretto nel 1993 più film di quanti ne avevano diretti nel 1992

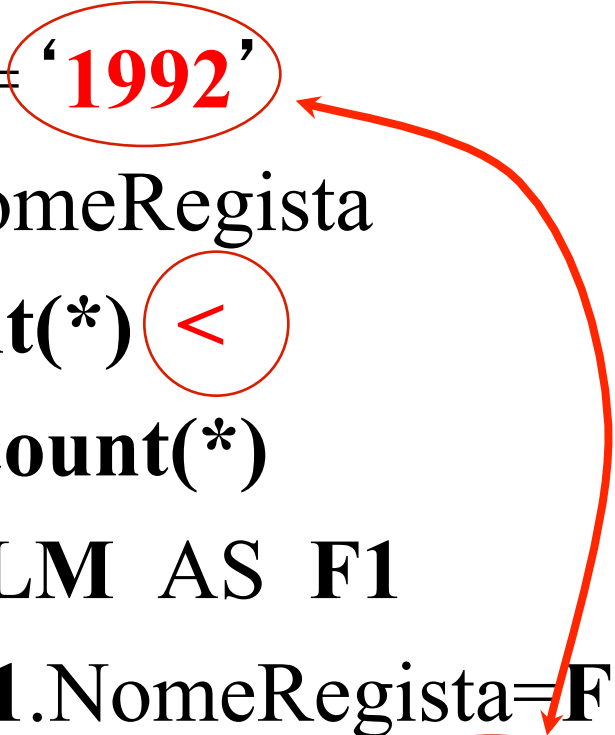
```
SELECT NomeRegista
FROM FILM AS F
WHERE Anno='1993'
GROUP BY NomeRegista
HAVING count(*) >
( SELECT count(*)
  FROM FILM AS F1
  WHERE F1.NomeRegista=F.NomeRegista
    AND Anno='1992' )
```

Nomi dei registi che hanno diretto nel 1993 più film di quanti ne avevano diretti nel 1992: INVERSIONE?

```
SELECT NomeRegista
FROM FILM AS F
WHERE Anno= '1992'
GROUP BY NomeRegista
HAVING count(*) <
( SELECT count(*)
  FROM FILM AS F1
  WHERE F1.NomeRegista=F.NomeRegista
    AND Anno= '1993' )
```

Nomi dei registi che hanno diretto nel 1993 più film di quanti ne avevano diretti nel 1992: INVERSIONE?

```
SELECT NomeRegista
FROM FILM AS F
WHERE Anno='1992'
GROUP BY NomeRegista
HAVING count(*) <
( SELECT count(*)
  FROM FILM AS F1
  WHERE F1.NomeRegista=F.NomeRegista
    AND Anno='1993')
```



Errore: dimentica i registi che non hanno diretto ALCUN film nel 92

Nomi dei registi che hanno diretto nel 1993 più film di quanti ne avevano diretti nel 1992: vista intermedia

```
CREATE VIEW NumPerAnno (Nom, Ann, Num) AS
SELECT NomeRegista, Anno, count(*)
FROM FILM
GROUP BY NomeRegista, Anno
```

```
SELECT Nom AS NomeRegistaCercato
FROM NumPerAnno N1
WHERE Ann = 93 AND
      Nom NOT IN ( SELECT Nom
                   FROM NumPerAnno N2
                   WHERE N2.Ann = 92 AND
                         N1.Num <= N2.Num )
```

Le date di nascita dei registi che hanno diretto film in proiezione sia a Torino sia a Milano

Le date di nascita dei registi che hanno diretto film in proiezione sia a Torino sia a Milano

```
select distinct NomeRegista, DataNascita
from REGISTA join FILM
    on Nome=NomeRegista
where Titolo in ( SELECT TitoloFilm
                  FROM PROIEZIONE
                  WHERE CittàCin= 'Milano' )
AND Titolo in ( SELECT TitoloFilm
                  FROM PROIEZIONE
                  WHERE CittàCin= 'Torino' )
```

Le date di nascita dei registi che hanno diretto film in proiezione sia a Torino sia a Milano

```
select distinct NomeRegista, DataNascita
from REGISTA join FILM
  on Nome=NomeRegista
where Titolo in ( SELECT TitoloFilm
                  FROM PROIEZIONE
                  WHERE CittàCin= 'Milano'
                  AND CittàCin= 'Torino' )
```

Le date di nascita dei registi che hanno diretto film in proiezione sia a Torino sia a Milano

```
select distinct NomeRegista, DataNascita
from REGISTA join FILM
  on Nome=NomeRegista
where Titolo in ( SELECT TitoloFilm
                  FROM PROIEZIONE
                  WHERE CittàCin= 'Milano'
                  OR CittàCin= 'Torino' )
```


Film proiettati nel maggior numero di cinema di Milano

Film proiettati nel maggior numero di cinema di Milano

```
SELECT TitoloFilm, count(*) AS NumCin
```

```
FROM PROIEZIONE
```

```
WHERE Città= 'Milano'
```

```
GROUP BY TitoloFilm
```

```
HAVING count(*) >=
```

```
( SELECT count(*)
```

```
FROM PROIEZIONE
```

```
WHERE Città= 'Milano'
```

```
GROUP BY TitoloFilm)
```

*NumCin non è richiesto
dalla specifica, ma
migliora la leggibilità*

Film proiettati nel maggior numero di cinema di Milano

```
SELECT TitoloFilm, count(*) AS NumCin
```

```
FROM PROIEZIONE
```

```
WHERE Città= 'Milano'
```

```
GROUP BY TitoloFilm
```

```
HAVING count(*) >= ALL
```

```
( SELECT count(*)
```

```
FROM PROIEZIONE
```

```
WHERE Città= 'Milano'
```

```
GROUP BY TitoloFilm)
```

*NumCin non è richiesto
dalla specifica, ma
migliora la leggibilità*

Film proiettati nel maggior numero di cinema di Milano

```
SELECT TitoloFilm, count(*) AS NumCin
```

```
FROM PROIEZIONE
```

```
WHERE Città= 'Milano'
```

```
GROUP BY TitoloFilm
```

```
HAVING count(*) >= ALL
```

```
( SELECT count(*)
```

```
FROM PROIEZIONE
```

```
WHERE Città= 'Milano'
```

```
GROUP BY TitoloFilm)
```

NumCin non è richiesto dalla specifica, ma migliora la leggibilità

BLOCCHI IDENTICI: si può usare una vista

Film proiettati nel maggior numero di cinema di Milano (vista intermedia)

```
CREATE VIEW ProiezMilano (Titolo, Num) AS
SELECT TitoloFilm, count(*)
FROM PROIEZIONE
WHERE Città= 'Milano'
GROUP BY TitoloFilm
```

```
SELECT Titolo, Num
FROM ProiezMilano
```

```
WHERE Num = ( SELECT max(Num)
                FROM ProiezMilano )
```

*Attenzione alle
condizioni con
aggregati!*

Trovare gli attori che hanno interpretato più
personaggi in uno stesso film (+ di 1 !!)

Trovare gli attori che hanno interpretato più personaggi in uno stesso film (+ di 1 !!)

```
select distinct P1.Attore
from INTERPRETA P1 , INTERPRETA P2
where P1.Attore = P2.Attore
and P1.Film = P2.Film
and P1.Personaggio <> P2.Personaggio
```

```
select distinct Attore
from INTERPRETA
group by Attore, Film
having count(*) > 1
```

← *PIU' EFFICIENTE*
Tipicamente riesce a sfruttare un indice definito sulla chiave per raggruppare rapidamente

SELECT Attore as Chi, Film as Dove, count() as Quanti*

Trovare i film in cui recita un solo attore che però interpreta più personaggi

Trovare i film in cui recita un solo attore che però interpreta più personaggi

```
SELECT Film
FROM INTERPRETA
GROUP BY Film
HAVING count(*) > 1
      AND count(distinct Attore) = 1
```

Attori italiani che non hanno mai recitato con altri italiani

Attori italiani che non hanno mai recitato con altri italiani

```
SELECT Nome
FROM ATTORE A1
WHERE Nazionalità = "Italiana" AND
    A1.Nome not in (
        SELECT I1.Attore
        FROM INTERPRETA I1,INTERPRETA I2,
            ATTORE A2
        WHERE I1.Titolo = I2.Titolo
            AND I2.Attore = A2.Nome
            AND A2.Nome <> A1.Nome
            AND A2.Nazionalità = "Italiana" )
```

Attori italiani che non hanno mai recitato con altri italiani

*In alternativa si
può definire un'
opportuna vista
intermedia*

```
CREATE VIEW Interp-italiano AS
SELECT Film, Attore
FROM INTERPRETA
WHERE Attore IN
      (SELECT Nome
       FROM ATTORE
       WHERE
        Nazionalità="Italiana")
```

```
SELECT Attore
FROM Interp-italiano
WHERE Attore NOT IN
      SELECT X.Attore
      FROM Interp-italiano X, Interp-italiano Y
      WHERE X.Film=Y.Film AND X.Nome<>Y.Nome
```


Registi che hanno recitato in (almeno) un loro film

```
SELECT DISTINCT NomeRegista  
FROM FILM join INTERPRETA  
on Titolo=Film  
WHERE NomeRegista=Attore
```

I registi che hanno recitato in almeno 4 **loro** film interpretandovi un totale di almeno 5 personaggi diversi

```
select NomeRegista
from FILM join INTERPRETA
      on Titolo=Film
where NomeRegista=Attore
group by NomeRegista
having count( distinct Titolo ) >= 4 and
      count( distinct Personaggio ) >= 5
```

NB: non trattiamo il caso in cui un regista/attore interpreta personaggi *diversi* che però hanno lo stesso nome, in film diversi

“Anagrafe”

PERSONA(CodFis,Nome,DataNascita,
CFMadre,CFPadre)

MATRIMONIO(Codice,CFMoglie,CFMarito,
Data,NumeroInvitati)

TESTIMONI(CodiceMatr,CFTestimone)

Estrarre tutti i matrimoni del 2010

PERSONA(CodFis,Nome,DataNascita,
CFMadre,CFPadre)

MATRIMONIO(Codice,CFMoglie,CFMarito,
Data,NumeroInvitati)

TESTIMONI(CodiceMatr,CFTestimone)

SELECT *

FROM MATRIMONIO

WHERE Data >= '1/1/2010' AND Data <= '31/12/2010'

Estrarre i dati dei genitori delle
persone che si sono sposate nel
2010

```
PERSONA(CodFis,Nome,DataNascita,  
CFMadre,CFPadre)
```

```
MATRIMONIO(Codice,CFMoglie,CFMarito,  
Data,NumeroInvitati)
```

```
TESTIMONI(CodiceMatr,CFTestimone)
```

```
SELECT P1.*  
FROM PERSONA P1, PERSONA P2  
WHERE (P1.CodFis =P2.CFMadre OR P1.CodFis =P2.CFPadre) AND  
(P2.CodFis IN (SELECT CFMoglie  
FROM MATRIMONIO  
WHERE Data>= '1/1/2010' AND  
Data<= '31/12/2010' ) OR  
P2.CodFis IN (SELECT CFMarito  
FROM MATRIMONIO  
WHERE Data>= '1/1/2010' AND  
Data<= '31/12/2010' ))
```

Coppie di persone sposatesi dopo la nascita di più di 3 [loro] figli.

```
PERSONA(CodFis,Nome,DataNascita,  
CFMadre,CFPadre)
```

```
MATRIMONIO(Codice,CFMoglie,CFMarito,  
Data,NumeroInvitati)
```

```
TESTIMONI(CodiceMatr,CFTestimone)
```

```
SELECT CFMoglie,CFMarito  
FROM MATRIMONIO M  
WHERE (SELECT count(*)  
FROM PERSONA P  
WHERE P.CFMadre=M.CFMoglie AND P.CFPadre=M.CFMarito  
AND P.DataNascita<M.Data)>3
```

Matrimoni in cui entrambi i coniugi erano precedentemente sposati.

```
PERSONA(CodFis,Nome,DataNascita,  
CFMadre,CFPadre)
```

```
MATRIMONIO(Codice,CFMoglie,CFMarito,  
Data,NumeroInvitati)
```

```
TESTIMONI(CodiceMatr,CFTestimone)
```

```
SELECT *  
FROM MATRIMONIO M  
WHERE CFMoglie IN (SELECT CFMoglie  
FROM Matrimonio M1  
WHERE M1.CFMoglie=M.CFMoglie  
AND M1.Data<M.Data)  
AND CFMarito IN (SELECT CFMarito  
FROM Matrimonio M2  
WHERE M2.CFMarito=M.CFMarito  
AND M2.Data<M.Data)
```

Estrarre i nomi delle coppie di individui sposati che risultano entrambi figli di genitori sposati tra loro

```
PERSONA(CodFis,Nome,DataNascita,  
CFMadre,CFPadre)
```

```
MATRIMONIO(Codice,CFMoglie,CFMarito,  
Data,NumeroInvitati)
```

```
TESTIMONI(CodiceMatr,CFTestimone)
```

```
SELECT P1.Nome, P2.Nome  
FROM MATRIMONIO M, PERSONA P1, PERSONA P2  
WHERE M.CFMoglie=P1.CodFis AND M.CFMarito=P2.CodFis  
AND CFMoglie IN (SELECT CodFis  
FROM Persona P,Matrimonio M1  
WHERE M1.CFMoglie=P.CFMadre  
AND M1.CFMarito=P.CFPadre)  
AND CFMarito IN (SELECT CodFis  
FROM Persona P,Matrimonio M1  
WHERE M1.CFMoglie=P.CFMadre  
AND M1.CFMarito=P.CFPadre)
```

Estrarre le persone sposate, figlie
di persone non sposate [tra loro]

```
PERSONA(CodFis,Nome,DataNascita,  
CFMadre,CFPadre)  
MATRIMONIO(Codice,CFMoglie,CFMarito,  
Data,NumeroInvitati)  
TESTIMONI(CodiceMatr,CFTestimone)
```

```
SELECT *  
FROM PERSONA P, MATRIMONIO M  
WHERE (P.CodFis=M.CFMoglie OR P.CodFis=M.CFMarito)  
AND (SELECT count(*)  
FROM Matrimonio M1  
WHERE M1.CFMoglie=P.CFMadre  
AND M1.CFMarito=P.CFPadre)=0
```

Estrarre i matrimoni che sono nel primo 20% per numero di invitati

```
PERSONA(CodFis,Nome,DataNascita,  
CFMadre,CFPadre)
```

```
MATRIMONIO(Codice,CFMoglie,CFMarito,  
Data,NumeroInvitati)
```

```
TESTIMONI(CodiceMatr,CFTestimone)
```

```
SELECT *  
FROM MATRIMONIO M  
WHERE (SELECT count(*)  
FROM Matrimonio M1  
WHERE M1.NumeroInvitati >= M.NumeroInvitati)  
<= 0.2*(SELECT count(*)  
FROM Matrimonio)
```

Estrarre Donne che hanno
sposato due omonimi

```
PERSONA(CodFis,Nome,DataNascita,  
CFMadre,CFPadre)
```

```
MATRIMONIO(Codice,CFMoglie,CFMarito,  
Data,NumeroInvitati)
```

```
TESTIMONI(CodiceMatr,CFTestimone)
```

```
SELECT *  
FROM PERSONA P  
WHERE P.CodFis IN  
(SELECT M1.CFMoglie  
FROM Matrimonio M1,Matrimonio M2,PERSONA P1,PERSONA P2  
WHERE M1.CFMarito=P1.CodFis AND M2.CFMarito=P2.CodFis  
AND M1.CFMoglie=M2.CFMoglie  
AND P1.Nome=P2.Nome)
```


Estrarre le donne che hanno
sposato due omonimi

```
PERSONA(CodFis,Nome,DataNascita,  
CFMadre,CFPadre)
```

```
MATRIMONIO(Codice,CFMoglie,CFMarito,  
Data,NumeroInvitati)
```

```
TESTIMONI(CodiceMatr,CFTestimone)
```

```
SELECT *  
FROM PERSONA P  
WHERE P.CodFis IN  
(SELECT M1.CFMoglie  
FROM Matrimonio M1,Matrimonio M2,PERSONA P1,PERSONA P2  
WHERE M1.CFMarito=P1.CodFis AND M2.CFMarito=P2.CodFis  
AND M1.CFMoglie=M2.CFMoglie  
AND P1.Nome=P2.Nome AND P1.CodFis<>P2.CodFis)
```

Estrarre gli uomini che sono stati testimoni di nozze di una loro ex-moglie

```
PERSONA(CodFis,Nome,DataNascita,  
CFMadre,CFPadre)
```

```
MATRIMONIO(Codice,CFMoglie,CFMarito,  
Data,NumeroInvitati)
```

```
TESTIMONI(CodiceMatr,CFTestimone)
```

```
SELECT *  
FROM PERSONA P  
WHERE (SELECT count(*)  
FROM Matrimonio M1  
WHERE M1.CFMarito=P.CodFis)  
AND M1.CFMoglie IN (SELECT CFMoglie  
FROM Matrimonio M2,TESTIMONI T  
WHERE M2.Codice=T.Codice  
AND T.CFTestimone=P.CodFis  
AND M2.Data>M1.Data  
) ) > 0
```

Le Affinità Elettive (cfr. J.W.Goethe, 1810): estrarre le coppie AB e CD si ricombinano in AD e BC, dopo essersi frequentate

PERSONA(CodFis,Nome,DataNascita,

CFMadre,CFPadre)

MATRIMONIO(Codice,CFMoglie,CFMarito,

Data,NumeroInvitati)

TESTIMONI(CodiceMatr,CFTestimone)

```
SELECT AB.CFMoglie, AB.CFMarito, CD.CFMoglie, CD.CFMarito,
FROM MATRIMONIO AB, Matrimonio CD
WHERE (SELECT count(*)
        FROM Matrimonio AD
        WHERE AD.CFMarito=CD.CFMarito AND AD.Moglie=AB.Moglie
            AND AD.Data>=AB.Data AND AD.Data>=CD.Data ) > 0
AND
(SELECT count(*)
 FROM Matrimonio BC
 WHERE BC.CFMarito=AB.CFMarito AND BC.Moglie=CD.Moglie
     AND BC.Data>=AB.Data AND BC.Data>=CD.Data) > 0
```

Dato il seguente schema relazionale:

AGENTE(Nome, Percentuale)

ARTICOLO(Nome, Descrizione, Tipo)

CLIENTE(Nome, Indirizzo, Telefoni)

VENDITA(Nome-Comp, Nome-Art,
Nome-Ag, Data, Quantità, Importo,
Validità)

Nomi degli agenti che hanno venduto più di 5 articoli di tipo "automobile" nel 1993

```
CREATE VIEW V1(Nome, Quantità) AS
SELECT Ag.Nome, V.Quantità
FROM Agente Ag, Articolo Ar, Vendita V
WHERE Ar.Nome=V.NomeArt
      AND Ag.Nome=V.NomeAg
      AND V.Data between 1/1/93 and 31/12/93
      AND Ar.Tipo="automobile"
```

```
SELECT Nome
FROM V1
GROUP BY Nome
HAVING sum(Quantità) > 5
```

Selezionare gli Agenti che hanno venduto qualche articolo di tipo “scarpa” ma non hanno venduto nulla a clienti il cui indirizzo è “via Po’ , Milano”

```
SELECT V.NomeAg
FROM ARTICOLO A, VENDITA V
WHERE A.Nome=V.NomeArt
and A.Tipo="scarpa" and Vendite.NomeAg NOT IN
(SELECT Vendite.NomeAg
FROM Cliente,Vendita
WHERE Cliente.Nome=Vendite.NomeComp
AND Cliente.Indirizzo = “via Po’ , Milano” )
```

Calcolare il totale dei guadagni degli agenti che vendono articoli di tipo 'immobile'

```
CREATE VIEW Implmm (NomAg, Tot) as  
SELECT NomeAg, sum(Importo) as ImpTot  
FROM Vendita join Articolo on Nome=NomeArt  
WHERE Tipo= 'immobile'  
GROUP BY NomeAg
```

```
SELECT Nome, Tot*Percentuale/100 as totGuad  
FROM Implmm JOIN Agente ON NomAg=Nome
```

Dato il seguente schema relazionale:

AUTORE(NOME, COGNOME, Data-N, Nazionalita)

AUTORELIBRO(NOME, COGNOME, SEGNATURA)

LIBRO(SEGNATURA, Scaffale, Argomento, Lingua)

Selezionare il COGNOME degli autori tedeschi di libri in italiano con argomento “filosofia” o “logica”

```
SELECT Cognome
FROM Autore A, Libro L, Autorelibro AL,
WHERE A.Nome=AL.Nome
and A.Cognome=AL.Cognome
and A.Segnatura=L.Segnatura
and Nazionalita=“tedesca”
and Lingua=“italiano” and
(Argomento=“filosofia” OR Argomento=“logica”)
```

Selezionare la data di nascita degli autori italiani di libri in inglese di Argomento “informatica”, che non sono autori di libri di Argomento “matematica”.

```
SELECT Data_N  
FROM Autore AS A JOIN Autorelibro ON  
    (A.Nome=Autorelibro.Nome AND  
    A.Cognome=Autorelibro.Cognome)  
JOIN Libro ON  
    (Autorelibro.Segnatura=Libro.Segnatura)  
WHERE Nazionalita="IT" AND Lingua="ING"  
AND Argomento="INF" AND  
(A.Nome, A.Cognome) NOT IN  
( SELECT AL.Nome, AL.Cognome  
FROM Autorelibro AS AL JOIN Libro AS L  
ON (AL.Segnatura=L.Segnatura)  
WHERE A.Nome=AL.Nome  
    AND A.Cognome=AL.Cognome  
    AND Argomento="MATEMATICA" )
```

Selezionare quegli autori (selezionati in base al loro Nome e Cognome) che hanno più di 10 libri diversi contenuti nel terzo scaffale della biblioteca

```
SELECT Nome, Cognome  
FROM Autorelibro JOIN Libro ON  
Autorelibro.Segnatura=Libro.Segnatura  
WHERE Scaffale="3"  
GROUP BY Cognome, Nome  
HAVING COUNT(*) > 10
```

Schema musica

CD (CDNumber, Title, Year, Price)

Track (CDNumber, PerformanceCode, trackNo)

Recording (Performance, SongTitle, Year)

Composer (CompName, SongTitle)

Singer (SingerName, PerformanceCode)

I cantautori (persone che hanno scritto e cantato la stessa canzone) il cui nome è 'David'

```
SELECT SingerName
FROM ( Singer S join Recording R on
      S.PerformanceCode=R.Performance )
join Composer C on R.SongTitle=C.SongTitle
WHERE SingerName=CompName
      AND SingerName = 'David'
```

I titolo dei dischi che contengono canzoni di cui non si conosce l' anno di registrazione

```
SELECT Title
FROM CD
    JOIN Track AS T ON
        CD.CDNumber=T.CDNumber
    JOIN Recording AS R ON
        T.PerformanceCode=
            R.PerformanceCode
WHERE R.Year IS NULL
```

I pezzi del disco con numero di serie 78574, ordinati per numero progressivo, con indicazione degli interpreti associati

```
SELECT TrackNo, SingerName
FROM Track JOIN Singer ON
        Track.PerformanceCode=
                Singer.PerformanceCode
WHERE CDNumber=78574
ORDER BY TrackNo
```


Gli autori che non hanno mai inciso una canzone
scritta da loro

```
SELECT CompName
FROM Composer
WHERE CompName NOT IN
(SELECT CompName
FROM Composer AS C
JOIN Recording AS R ON
    C.SongTitle=R.SongTiltle
JOIN Singer ON
    Performance=PerformanceCode
WHERE CompName=SingerName )
```

Il cantante del CD che contiene il maggior numero di canzoni

```
create view CdwithNumber(CdNum,NumOfSongs)  
as select CDNumber, count(*)  
from Track  
group by CDNumber
```

```
select SingerName  
from Singer S join Track T on  
    S.PerformanceCode = T.PerformanceCode  
join CdwithNumber C on  
    T.CDNumber = C.CDNum  
where NumOfSongs = (select max (NumOfSongs)  
                    from CdwithNumber)
```

9/3/2007

- Un database gestisce le bollette telefoniche di una compagnia di telefonia mobile.

CLIENTE (codicefiscale, nome, cognome, numTelefonico,
PianoTariffario)

PIANOTARIFFARIO (codice, costoScattoAllaRisposta,
costoAlSecondo)

TELEFONATA (codicefiscale, data, ora,
numeroDestinatario, durata)

BOLLETTA (codicefiscale, mese, anno, cifra)

Selezionare i clienti per i quali l' ammontare complessivo delle bollette del 2006 supera di oltre il 20% l' ammontare delle proprie bollette nell' anno 2005.

```
SELECT codfiscale, SUM(cifra)
FROM BOLLETTA B1
WHERE anno = 2006
GROUP BY codfiscale
HAVING SUM(cifra) > 1,20 * (
    SELECT SUM(cifra)
    FROM BOLLETTA B2
    WHERE B1.codfiscale = B2.codfiscale
    AND B2.anno = 2005
)
```

Selezionare i clienti per i quali il costo vivo delle telefonate (inteso senza scatto alla risposta) sia mediamente inferiore allo scatto alla risposta del piano tariffario da essi sottoscritto. Si utilizzi una vista per calcolare il costo vivo di ogni telefonata.

```
CREATE VIEW CostoVivo (codicefiscale, data, ora, costo) AS
SELECT T.codicefiscale, T.data, T.ora, T.durata * P.costoAlSecondo
FROM (TELEFONATA T JOIN CLIENTE C
      ON T.codicefiscale = C.codicefiscale)
     JOIN PIANOTARIFFARIO P ON C.pianoTariffario = P.codice)
```

```
SELECT codicefiscale
FROM CostoVivo CV
GROUP BY codicefiscale
HAVING avg(costo) > ALL (SELECT costoScattoAllaRisposta
                        FROM PIANOTARIFFARIO P JOIN CLIENTE C
                        ON P.codice = C.pianoTariffario
                        WHERE C.codicefiscale = CV.codicefiscale)
```

5/7/2007

- Il seguente schema rappresenta i dati relativi alle prenotazioni alberghiere effettuate presso una agenzia viaggi.

HOTEL(Codice, NomeH, Citta, Classe)

CLIENTE(CodiceFiscale, NomeC, CognomeC, Indirizzo, Telefono)

PRENOTAZIONE(CodiceCliente, CodiceHotel, DataPartenza,
CostoGiornaliero, Durata)

Estrarre il nome, la città e la classe degli hotel in cui nel 2006 qualche cliente ha soggiornato per almeno 2 volte

```
CREATE VIEW SOGGIORNI06(CodiceHotel,CodiceCliente,NroSoggiorni) AS
SELECT CodiceHotel, CodiceCliente, Count(*)
FROM PRENOTAZIONE
WHERE DataPartenza >= '01.01.2006' AND DataPartenza <= '31.12.2006'
GROUPBY CodiceHotel, CodiceCliente
```

```
SELECT NomeH, Citta, Classe
FROM HOTEL
WHERE Codice IN ( SELECT CodiceHotel
                  FROM SOGGIORNI06
                  WHERE NroSoggiorni >=2 )
```

Determinare il soggiorno più costoso per quei clienti che non hanno mai prenotato soggiorni di durata superiore ai 7 giorni. Si estraggano il codice fiscale del cliente, la data di partenza, il costo del soggiorno e il nome dell' hotel

```
CREATE VIEW COSTOSOGGIORNO (CodiceFiscale, Costo) AS  
SELECT CodiceCliente, CostoGiornaliero* Durata  
FROM PRENOTAZIONE
```

```
SELECT P.CodiceCliente, P.DataPartenza, C1.Costo, H.NomeHotel  
FROM PRENOTAZIONE P, COSTOSOGGIORNO C1, HOTEL H  
WHERE P.CodiceCliente=C1.CodiceFiscale AND  
P.CodiceHotel=H.Codice AND  
C1.Costo = (SELECT MAX(Costo)  
FROM COSTOSOGGIORNO as C2  
WHERE C1. CodiceFiscale=C2. CodiceFiscale) AND  
P.CodiceCliente NOT IN (SELECT CodiceCliente  
FROM PRENOTAZIONE  
WHERE Durata>7)
```


5/9/2007

- Il seguente schema rappresenta i dati relativi ai campionati mondiali di calcio.

SQUADRA(Nazione, Anno, Allenatore,
PosizioneInClassifica)

ORGANIZZAZIONE (Anno, Nazione)

GIOCATORE (ID, Nome)

PARTECIPAZIONE (IDGiocatore, Anno, Nazione, Ruolo,
GoalSegnati)

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

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

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

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

```
select Nazione
from Squadra
where Nazione not in (select Nazione
                      from Organizza O, Squadra S
                      where O.Nazione=s.Nazione
                      and S.Anno = O.Anno and PosizioneInClassifica = 1 )
```

```
select Nazione
from Organizza O, Squadra S
where O.Nazione=s.Nazione
and S.Anno = O.Anno and PosizioneInClassifica != 1 )
```

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

SQUADRA(Nazione, Anno, Allenatore, PosizioneInClassifica)

ORGANIZZAZIONE (Anno, Nazione)

GIOCATORE (ID, Nome)

PARTECIPAZIONE (IDGiocatore, Anno, Nazione, Ruolo, GoalSegnati)

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

```
select Anno, Nazione, count(*) as NumeroConvocazioni  
from Partecipazione P  
group by Anno, Nazione  
having count(*) >= all ( select count(*)  
                        from Partecipazione  
                        where Anno = P.Anno  
                        group by Nazione )
```

```
select Anno, Nazione, count(*) as NumeroConvocazioni  
from Partecipazione P  
group by Anno, Nazione  
having count(*) >= all ( select count(*)  
                          from Partecipazione  
                          group by Anno, Nazione )
```

In alternativa, con una vista intermedia:

```
create view NumeroConv(Edizione,Squadra,Convocati) as  
select Anno, Nazione, count(*)  
from Partecipazione P  
group by Anno, Nazione
```

```
select Edizione, Squadra, Convocati  
from NumeroConv N  
where Convocati = ( select max(Convocati)  
                    from NumeroConv  
                    where Edizione = N.Edizione )
```

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

SQUADRA(Nazione, Anno, Allenatore, PosizioneInClassifica)

ORGANIZZAZIONE (Anno, Nazione)

GIOCATORE (ID, Nome)

PARTECIPAZIONE (IDGiocatore, Anno, Nazione, Ruolo, GoalSegnati)

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 )
```


1/2/2008

- Il seguente schema rappresenta le informazioni riguardo alla gestione di una videoteca:

DVD (CodiceDVD, TitoloFilm, Regista, Durata)

CLIENTE (CodiceFiscale, Nome, Cognome, Indirizzo, Telefono, Categoria)

NOLEGGIO (CodiceFiscale, CodiceDVD, DataInizio, DataFine, CostoGiornaliero)

Scrivere in SQL l'interrogazione che estrae i clienti che non hanno mai noleggiato due film dello stesso regista.

```
SELECT Codicefiscale, Nome, Cognome  
FROM CLIENTE  
WHERE CodicdeFiscale NOT IN  
(  
    SELECT N1.CodiceFiscale  
    FROM DVD D1, NOLEGGIO N1, DVD D2, NOLEGGIO N2  
    WHERE N1. CodiceFiscale=N2. CodiceFiscale AND  
          N1.CodiceDVD=D1.CodiceDVD AND  
          N2.CodiceDVD=D2.CodiceDVD AND  
          D1.Regista=R2.Regista AND  
          D1.Titolo<>D2.Titolo  
)
```

Scrivere in SQL l'interrogazione che estrae il cliente con il maggior numero di noleggi iniziati nel 2007.

```
SELECT Codicefiscale, Nome, Cognome  
FROM CLIENTE  
WHERE CodicdeFiscale IN
```

```
(
```

```
    SELECT CodiceFiscale  
    FROM NOLEGGIO  
    WHERE DataInizio>=1/1/2007 AND DataInizio <=31/12/2007  
    GROUP BY CodiceFiscale  
    HAVING count(*) >=ALL SELECT count(*)  
                                FROM NOLEGGIO  
                                WHERE DataInizio>=1/1/2007 AND  
                                    DataInizio <=31/12/2007  
                                GROUP BY CodiceFiscale
```

```
)
```

Catalogo prodotti

FORNITORI (CodiceForn, Nome, Indirizzo, Città)

PRODOTTO (Codice, Nome, Descrizione, Marca, Modello,
QtaMagazzino)

CATALOGO (CodiceForn, CodiceProd, Costo)

CLIENTE(CodCliente, Nome, Indirizzo, Città)

ORDINE(Numero, CodCliente, Data, Importo)

PARTIORDINE(NroOrdine, CodProdotto, Quantita,

PrezzoUnitario)

I codici di tutti i prodotti distribuiti da *almeno due* fornitori

I codici di tutti i prodotti distribuiti da *almeno due* fornitori

```
SELECT DISTINCT C.CodiceProd
FROM    Catalogo AS C,
        Catalogo AS C1
WHERE C.CodiceForn <> C1.CodiceForn
      AND C.CodiceProd=C1.CodiceProd
```

I codici di tutti i prodotti distribuiti da *almeno due* fornitori

```
SELECT DISTINCT C.CodiceProd
FROM   Catalogo AS C,
       Catalogo AS C1
WHERE  C.CodiceForn > C1.CodiceForn
       AND C.CodiceProd=C1.CodiceProd
```

*“Dimezza” la
dimensione della
tabella coinvolta*

I codici di tutti i prodotti distribuiti da *almeno due* fornitori

SQL permette anche di ragionare sui gruppi:

```
SELECT CodiceProd  
FROM Catalogo  
GROUP BY CodiceProd  
HAVING count (*) >1
```


Di ogni prodotto calcolare il costo *medio* di fornitura **in**
ciascuna città

Di ogni prodotto calcolare il costo *medio* di fornitura **in**
ciascuna città

```
SELECT CodiceProd, Città, avg(costo) AS CostoMedio  
FROM Catalogo C, Fornitori F  
WHERE C.CodiceForn=F.CodiceForn  
GROUP BY Città, CodiceProd
```

Nomi dei fornitori “universali” – cioè che distribuiscono tutti i prodotti in catalogo

Nomi dei fornitori “universali” – cioè che distribuiscono tutti i prodotti in catalogo

Cerchiamo i fornitori tali per cui non ci sia un prodotto tale per cui non ci sia in catalogo un accoppiamento tra QUEL fornitore e QUEL prodotto

Nomi dei fornitori “universali” – cioè che distribuiscono tutti i prodotti in catalogo (versione più pulita)

```
SELECT CodiceForn, Nome  
FROM Fornitori
```

```
WHERE CodiceForn NOT IN
```

```
( SELECT CodiceForn
```

```
FROM Prodotti, Fornitori
```

```
WHERE (CodiceProd, CodiceForn) NOT IN
```

```
( SELECT CodiceProd, CodiceForn
```

```
FROM Catalogo C ) )
```

**Versione che usa
solo il NOT IN**

**Prodotto
Cartesiano**



Nomi dei fornitori “universali” – cioè che distribuiscono tutti i prodotti in catalogo (versione intuitiva)

```
SELECT Nome
FROM Fornitori F JOIN Catalogo C
    ON F.CodiceForn=C.CodiceForn
GROUP BY F.CodiceForn, Nome
HAVING count(*) = (select count(*)
                    from Prodotti)
```

Attenzione, però: con “**tutti i prodotti**” è comodo perché c’è una tabella apposta.

In generale non è così banale.

Nomi dei clienti che non hanno
mai ordinato prodotti che siano
stati ordinati anche dalla ditta
“Brambilla”

Nomi dei clienti che non hanno
mai ordinato prodotti che siano
stati ordinati anche dalla ditta
“Brambilla”

```
SELECT Nome  
FROM Cliente
```

```
WHERE Nome not in
```

```
( SELECT nome
```

```
FROM cliente c, ordine o, partiordine p
```

```
WHERE c.codcliente=o.codcliente
```

```
AND numero=nroordine AND codprodotto in
```

```
( SELECT codprodotto
```

```
FROM cliente c2, ordine o2, partiordine p2
```

```
WHERE nome="Brambilla" AND
```

```
c2.codcliente=o2.codcliente AND
```

```
numero=nroordine))
```


Visualizzare i nomi dei clienti con
l'ammontare totale degli ordini effettuati

Visualizzare i nomi dei clienti con
l'ammontare totale degli ordini effettuati

```
SELECT C.CodCliente, C.Nome, sum(Importo) AS ImportoTot  
FROM Cliente AS C, Ordine AS O  
WHERE O.CodCliente=C.CodCliente  
GROUP BY C.CodCliente, C.Nome
```

Visualizzare i nomi dei clienti con
l'ammontare totale degli ordini effettuati
ordinati per ImportoTot

```
SELECT C.CodCliente, C.Nome, sum(Importo) AS ImportoTot  
FROM Cliente AS C, Ordine AS O  
WHERE O.CodCliente=C.CodCliente  
GROUP BY C.CodCliente, C.Nome  
ORDER BY 3
```

Trovare le descrizioni dei prodotti di cui si è
venduta nel 1995 una quantità maggiore almeno del
35% rispetto alla quantità venduta nel 1994

Trovare le descrizioni dei prodotti di cui si è venduta nel 1995 una quantità maggiore almeno del 35% rispetto alla quantità venduta nel 1994

```
CREATE VIEW
```

```
vista1 (CodProdotto, Somma, Data) AS
```

```
SELECT P.CodProdotto,
```

```
Sum(P.Quantità) AS Somma,
```

```
O.Data
```

```
FROM Ordine O, PartiOrdine P
```

```
WHERE P.NroOrdine=Numero
```

```
GROUP BY P.CodProdotto, O.Data
```

```
SELECT descrizione
FROM vista1, prodotto
WHERE prodotto.codice= vista1.CodProdotto
AND vista1.data=1995
AND vista1.codprodotto IN
(SELECT a.codprodotto
FROM vista1 as a, vista1 as b
WHERE vista1.data=1995
AND a.data=1994
AND a.codprodotto=b.codprodotto
AND b.somma>1.35*a.somma);
```

Esercizio

- Si considerino le tabelle (gli attributi sottolineati rappresentano la chiave di ogni tabella):

Motore (Codice, Nome, CostoTotale)

ComponentiMotore (CodiceMotore, CodiceComponente)

Componente (Codice, Nome, Costo)

- 1) Estrarre il nome del motore con il maggior numero di componenti.
- 2) Estrarre i motori che contengono solo componenti che costano più di 40 euro.
- 3) Trovare il motore per cui è massima la differenza tra il costo totale e la somma dei costi dei suoi componenti.

1.

Select Nome

From Motore

Where Codice IN (Select CodiceMotore

From ComponentiMotore

Group by CodiceMotore

Having Count (*) >= ALL (Select Count (*)

From ComponentiMotore

Group by CodiceMotore))

2.

Select *

From Motore

Where Codice NOT IN (Select CodiceMotore

From ComponentiMotore Inner Join Componente on

ComponentiMotore.CodiceComponente =

Componente.Codice

Where Costo <= 40)

3.

```
create view CostoMotore as (Select CodiceMotore , SUM(Costo) as CostoComponenti
    From ComponentiMotore Join Componente
    on ComponentiMotore.CodiceComponente = Componente.Codice
    Group by CodiceMotore )
```

```
Select *
From Motore Inner Join CostoMotore
on Motore.Codice = CostoMotore.CodiceMotore
Where (CostoTotale – CostoComponenti)>=ALL(Select CostoTotale - CostoComponenti
    From Motore Inner Join CostoMotore
    on Motore.Codice = vwCostoMotore.CodiceMotore)
```

Esercizio (tde 1-2-2008)

- Il seguente schema rappresenta le informazioni riguardo alla gestione di una videoteca:

DVD (CodiceDVD, TitoloFilm, Regista, Durata)

CLIENTE (CodiceFiscale, Nome, Cognome, Indirizzo, Telefono, Categoria)

NOLEGGIO (CodiceFiscale, CodiceDVD, DataInizio, DataFine, CostoGiornaliero)

1. Scrivere in SQL l'interrogazione che estrae i clienti che hanno noleggiato due film dello stesso regista.
2. Scrivere in SQL l'interrogazione che estrae i clienti per cui esiste un regista di cui non hanno noleggiato due film
3. Scrivere in SQL l'interrogazione che estrae i clienti che non hanno mai noleggiato due film dello stesso regista.
4. Scrivere in SQL l'interrogazione che estrae il cliente con il maggior numero di noleggi iniziati nel 2007.

1.

```
select distinct codiceFiscale
from DVD D join NOLEGGIO N on N.codiceDVD=D.codiceDVD
group by cf, regista
having count(distinct titoloFilm)>=2
```

2.

```
select distinct codiceFiscale
from DVD D join NOLEGGIO N on N.codiceDVD=D.codiceDVD
group by cf, regista
having count(distinct titoloFilm)<2
```

3.

```
select codiceFiscale
from NOLEGGIO
where codiceFiscale not in (select distinct codiceFiscale
                             from DVD D join NOLEGGIO N on N.codiceDVD=D.codiceDVD
                             group by cf, regista
                             having count(distinct titoloFilm)>=2 )
```

4.

```
select codiceFiscale
from NOLEGGIO
where dataInizio>=1.1.2007 && dataInizio<=31.12.2007
group by codiceFiscale
having count(*)>=ALL ( select count(*)
                       from NOLEGGIO
                       where dataInizio>=1.1.2007 and dataInizio<=31.12.2007
                       group by codiceFiscale )
```

Esercizio (tde 25-2-2008)

- Il seguente schema rappresenta le informazioni riguardo alle elezioni (con un sistema elettorale di fantasia):
CANDIDATO (CodiceFiscale, Cognome, Nome, NomeListaDiAppartenenza, PosizioneInLista, VotiRaccolti)
LISTA (Nome, Simbolo)
- Scrivere in SQL l'interrogazione che estrae il candidato che ha raccolto personalmente il maggior numero di voti.
- Scrivere in SQL l'interrogazione che estrae la lista i cui candidati hanno raccolto complessivamente più voti.

```
select *  
from candidato  
where votiraccolti = select max(votiraccolti)  
                    from candidato
```

```
select distinct nomelista  
from candidato  
group by nomelista  
having sum(voti raccolti) >= all select sum(votiraccolti)  
                                from candidato  
                                group by nomelista
```

Dato il seguente schema relazionale:

AGENTE(Nome, Percentuale)

ARTICOLO(Nome, Descrizione, Tipo)

CLIENTE(Nome, Indirizzo, Telefoni)

VENDITA(Nome-Comp, Nome-Art,
Nome-Ag, Data, Quantità, Importo,
Validità)

Nomi degli agenti che hanno venduto più di 5 articoli di tipo "automobile" nel 1993

```
CREATE VIEW V1(Nome, Quantità) AS
SELECT Ag.Nome, V.Quantità
FROM Agente Ag, Articolo Ar, Vendita V
WHERE Ar.Nome=V.NomeArt
      AND Ag.Nome=V.NomeAg
      AND V.Data between 1/1/93 and 31/12/93
      AND Ar.Tipo="automobile"
```

```
SELECT Nome
FROM V1
GROUP BY Nome
HAVING sum(Quantità) > 5
```

Selezionare gli Agenti che hanno venduto qualche articolo di tipo “scarpa” ma non hanno venduto nulla a clienti il cui indirizzo è “via Po’ , Milano”

```
SELECT V.NomeAg
FROM ARTICOLO A, VENDITA V
WHERE A.Nome=V.NomeArt
and A.Tipo="scarpa" and Vendite.NomeAg NOT IN
(SELECT Vendite.NomeAg
FROM Cliente,Vendita
WHERE Cliente.Nome=Vendite.NomeComp
AND Cliente.Indirizzo = “via Po’ , Milano” )
```


Calcolare il totale dei guadagni degli agenti che vendono articoli di tipo 'immobile'

```
CREATE VIEW Implmm (NomAg, Tot) as  
SELECT NomeAg, sum(Importo) as ImpTot  
FROM Vendita join Articolo on Nome=NomeArt  
WHERE Tipo= 'immobile'  
GROUP BY NomeAg
```

```
SELECT Nome, Tot*Percentuale/100 as totGuad  
FROM Implmm JOIN Agente ON NomAg=Nome
```

Dato il seguente schema relazionale:

AUTORE(NOME, COGNOME, Data-N, Nazionalita)

AUTORELIBRO(NOME, COGNOME, SEGNATURA)

LIBRO(SEGNATURA, Scaffale, Argomento, Lingua)

Selezionare il COGNOME degli autori tedeschi di libri in italiano con argomento “filosofia” o “logica”

```
SELECT Cognome
FROM Autore A, Libro L, Autorelibro AL,
WHERE A.Nome=AL.Nome
and A.Cognome=AL.Cognome
and A.Segnatura=L.Segnatura
and Nazionalita=“tedesca”
and Lingua=“italiano” and
(Argomento=“filosofia” OR Argomento=“logica”)
```

Selezionare la data di nascita degli autori italiani di libri in inglese di Argomento “informatica”, che non sono autori di libri di Argomento “matematica”.

```
SELECT Data_N  
FROM Autore AS A JOIN Autorelibro ON  
    (A.Nome=Autorelibro.Nome AND  
    A.Cognome=Autorelibro.Cognome)  
JOIN Libro ON  
    (Autorelibro.Segnatura=Libro.Segnatura)  
WHERE Nazionalita="IT" AND Lingua="ING"  
AND Argomento="INF" AND  
(A.Nome, A.Cognome) NOT IN  
( SELECT AL.Nome, AL.Cognome  
FROM Autorelibro AS AL JOIN Libro AS L  
ON (AL.Segnatura=L.Segnatura)  
WHERE A.Nome=AL.Nome  
    AND A.Cognome=AL.Cognome  
    AND Argomento="MATEMATICA" )
```

Selezionare quegli autori (selezionati in base al loro Nome e Cognome) che hanno più di 10 libri diversi contenuti nel terzo scaffale della biblioteca

```
SELECT Nome, Cognome  
FROM Autorelibro JOIN Libro ON  
Autorelibro.Segnatura=Libro.Segnatura  
WHERE Scaffale="3"  
GROUP BY Cognome, Nome  
HAVING COUNT(*) > 10
```

Schema musica

CD (CDNumber, Title, Year, Price)

Track (CDNumber, PerformanceCode, trackNo)

Recording (Performance, SongTitle, Year)

Composer (CompName, SongTitle)

Singer (SingerName, PerformanceCode)

I cantautori (persone che hanno scritto e cantato la stessa canzone) il cui nome è 'David'

```
SELECT SingerName
FROM ( Singer S join Recording R on
      S.PerformanceCode=R.Performance )
join Composer C on R.SongTitle=C.SongTitle
WHERE SingerName=CompName
      AND SingerName = 'David'
```


I titolo dei dischi che contengono canzoni di cui non si conosce l' anno di registrazione

```
SELECT Title
FROM CD
    JOIN Track AS T ON
        CD.CDNumber=T.CDNumber
    JOIN Recording AS R ON
        T.PerformanceCode=
            R.PerformanceCode
WHERE R.Year IS NULL
```

I pezzi del disco con numero di serie 78574, ordinati per numero progressivo, con indicazione degli interpreti associati

```
SELECT TrackNo, SingerName
FROM Track JOIN Singer ON
        Track.PerformanceCode=
                Singer.PerformanceCode
WHERE CDNumber=78574
ORDER BY TrackNo
```

Gli autori che non hanno mai inciso una canzone
scritta da loro

```
SELECT CompName
FROM Composer
WHERE CompName NOT IN
(SELECT CompName
FROM Composer AS C
JOIN Recording AS R ON
    C.SongTitle=R.SongTiltle
JOIN Singer ON
    Performance=PerformanceCode
WHERE CompName=SingerName )
```

Il cantante del CD che contiene il maggior numero di canzoni

```
create view CdwithNumber(CdNum,NumOfSongs)
as select CDNumber, count(*)
from Track
group by CDNumber
```

```
select SingerName
from Singer S join Track T on
    S.PerformanceCode = T.PerformanceCode
join CdwithNumber C on
    T.CDNumber = C.CDNum
where NumOfSongs = (select max (NumOfSongs)
                    from CdwithNumber)
```

9/3/2007

- Un database gestisce le bollette telefoniche di una compagnia di telefonia mobile.

CLIENTE (codicefiscale, nome, cognome, numTelefonico,
PianoTariffario)

PIANOTARIFFARIO (codice, costoScattoAllaRisposta,
costoAlSecondo)

TELEFONATA (codicefiscale, data, ora,
numeroDestinatario, durata)

BOLLETTA (codicefiscale, mese, anno, cifra)

Selezionare i clienti per i quali l' ammontare complessivo delle bollette del 2006 supera di oltre il 20% l' ammontare delle proprie bollette nell' anno 2005.

```
SELECT codfiscale, SUM(cifra)
FROM BOLLETTA B1
WHERE anno = 2006
GROUP BY codfiscale
HAVING SUM(cifra) > 1,20 * (
    SELECT SUM(cifra)
    FROM BOLLETTA B2
    WHERE B1.codfiscale = B2.codfiscale
    AND B2.anno = 2005
)
```

Selezionare i clienti per i quali il costo vivo delle telefonate (inteso senza scatto alla risposta) sia mediamente inferiore allo scatto alla risposta del piano tariffario da essi sottoscritto. Si utilizzi una vista per calcolare il costo vivo di ogni telefonata.

```
CREATE VIEW CostoVivo (codicefiscale, data, ora, costo) AS
SELECT T.codicefiscale, T.data, T.ora, T.durata * P.costoAlSecondo
FROM (TELEFONATA T JOIN CLIENTE C
      ON T.codicefiscale = C.codicefiscale)
     JOIN PIANOTARIFFARIO P ON C.pianoTariffario = P.codice)
```

```
SELECT codicefiscale
FROM CostoVivo CV
GROUP BY codicefiscale
HAVING avg(costo) > ALL (SELECT costoScattoAllaRisposta
                        FROM PIANOTARIFFARIO P JOIN CLIENTE C
                        ON P.codice = C.pianoTariffario
                        WHERE C.codicefiscale = CV.codicefiscale)
```

5/7/2007

- Il seguente schema rappresenta i dati relativi alle prenotazioni alberghiere effettuate presso una agenzia viaggi.

HOTEL(Codice, NomeH, Citta, Classe)

CLIENTE(CodiceFiscale, NomeC, CognomeC, Indirizzo, Telefono)

PRENOTAZIONE(CodiceCliente, CodiceHotel, DataPartenza,
CostoGiornaliero, Durata)

Estrarre il nome, la città e la classe degli hotel in cui nel 2006 qualche cliente ha soggiornato per almeno 2 volte

```
CREATE VIEW SOGGIORNI06(CodiceHotel,CodiceCliente,NroSoggiorni) AS
SELECT CodiceHotel, CodiceCliente, Count(*)
FROM PRENOTAZIONE
WHERE DataPartenza >= '01.01.2006' AND DataPartenza <= '31.12.2006'
GROUPBY CodiceHotel, CodiceCliente
```

```
SELECT NomeH, Citta, Classe
FROM HOTEL
WHERE Codice IN ( SELECT CodiceHotel
                  FROM SOGGIORNI06
                  WHERE NroSoggiorni >=2 )
```

Determinare il soggiorno più costoso per quei clienti che non hanno mai prenotato soggiorni di durata superiore ai 7 giorni. Si estraggano il codice fiscale del cliente, la data di partenza, il costo del soggiorno e il nome dell' hotel

```
CREATE VIEW COSTOSOGGIORNO (CodiceFiscale, Costo) AS  
SELECT CodiceCliente, CostoGiornaliero* Durata  
FROM PRENOTAZIONE
```

```
SELECT P.CodiceCliente, P.DataPartenza, C1.Costo, H.NomeHotel  
FROM PRENOTAZIONE P, COSTOSOGGIORNO C1, HOTEL H  
WHERE P.CodiceCliente=C1.CodiceFiscale AND  
P.CodiceHotel=H.Codice AND  
C1.Costo = (SELECT MAX(Costo)  
FROM COSTOSOGGIORNO as C2  
WHERE C1. CodiceFiscale=C2. CodiceFiscale) AND  
P.CodiceCliente NOT IN (SELECT CodiceCliente  
FROM PRENOTAZIONE  
WHERE Durata>7)
```

5/9/2007

- Il seguente schema rappresenta i dati relativi ai campionati mondiali di calcio.

SQUADRA(Nazione, Anno, Allenatore,
PosizioneInClassifica)

ORGANIZZAZIONE (Anno, Nazione)

GIOCATORE (ID, Nome)

PARTECIPAZIONE (IDGiocatore, Anno, Nazione, Ruolo,
GoalSegnati)

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

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

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

```
select Anno, Nazione, count(*) as NumeroConvocazioni  
from Partecipazione P  
group by Anno, Nazione  
having count(*) >= all ( select count(*)  
                        from Partecipazione  
                        where Anno = P.Anno  
                        group by Nazione )
```

In alternativa, con una vista intermedia:

```
create view NumeroConv(Edizione,Squadra,Convocati) as  
select Anno, Nazione, count(*)  
from Partecipazione P  
group by Anno, Nazione
```

```
select Edizione, Squadra, Convocati  
from NumeroConv N  
where Convocati = ( select max(Convocati)  
                    from NumeroConv  
                    where Edizione = N.Edizione )
```

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 )
```

1/2/2008

- Il seguente schema rappresenta le informazioni riguardo alla gestione di una videoteca:

DVD (CodiceDVD, TitoloFilm, Regista, Durata)

CLIENTE (CodiceFiscale, Nome, Cognome, Indirizzo, Telefono, Categoria)

NOLEGGIO (CodiceFiscale, CodiceDVD, DataInizio, DataFine, CostoGiornaliero)

Scrivere in SQL l'interrogazione che estrae i clienti che non hanno mai noleggiato due film dello stesso regista.

```
SELECT Codicefiscale, Nome, Cognome  
FROM CLIENTE  
WHERE CodicdeFiscale NOT IN  
(  
    SELECT N1.CodiceFiscale  
    FROM DVD D1, NOLEGGIO N1, DVD D2, NOLEGGIO N2  
    WHERE N1. CodiceFiscale=N2. CodiceFiscale AND  
        N1.CodiceDVD=D1.CodiceDVD AND  
        N2.CodiceDVD=D2.CodiceDVD AND  
        D1.Regista=R2.Regista AND  
        D1.Titolo<>D2.Titolo  
)
```

Scrivere in SQL l'interrogazione che estrae il cliente con il maggior numero di noleggi iniziati nel 2007.

```
SELECT Codicefiscale, Nome, Cognome  
FROM CLIENTE
```

```
WHERE CodicdeFiscale IN
```

```
(
```

```
SELECT CodiceFiscale
```

```
FROM NOLEGGIO
```

```
WHERE DataInizio>=1/1/2007 AND DataInizio <=31/12/2007
```

```
GROUP BY CodiceFiscale
```

```
HAVING count(*) >=ALL SELECT count(*)
```

```
FROM NOLEGGIO
```

```
WHERE DataInizio>=1/1/2007 AND
```

```
    DataInizio <=31/12/2007
```

```
GROUP BY CodiceFiscale
```

```
)
```

Esercizio (tde 10-9-2008)

- Il seguente schema rappresenta le informazioni riguardo un'edizione delle Olimpiadi:

ATLETA (CodiceFiscale, Cognome, Nome, Nazionalità)

MEDAGLIE (CodiceFiscale, Specialità, Data, Metallo)

- Scrivere in SQL l'interrogazione che estrae l'atleta che ha vinto più medaglie d'oro.
- Scrivere in SQL l'interrogazione che estrae la lista degli atleti che non hanno vinto nessuna medaglia.

Esercizio (tde 26-1-2009)

- Il seguente schema descrive i dati di un social network e consiste di due tabelle (chiavi in maiuscolo):
Utente(CODICE, Nome, Score)
Raccomanda(CODUTENTE, CODRACCOMANDATO)
- Utente con codice, nome e indice di gradimento nel social network (Score). L'utente con codice CodUtente raccomanda l'utente con codice CodRaccomandato.
 - 1) Scrivere una query in SQL che determina l'utente con lo score più elevato
 - 2) Scrivere una query in SQL che determina il nome della persona che ha più raccomandazioni

Esercizio (tde 18-2-2009)

- Il seguente schema rappresenta le informazioni riguardo alla gestione del personale:

DIPENDENTE (Matricola, Cognome, Nome, Bonus)

ASSENZA (Matricola, Data)

- Scrivere in SQL l'interrogazione che estrae per ogni dipendente l'ultima assenza.
- Scrivere in SQL l'interrogazione che estrae il dipendente con più assenze nel gennaio 2009.

Esercizio (tde 9-6-2009)

- Il seguente schema descrive una base di dati relativa ad una catena di autolavaggi che intende avviare un programma di fidelizzazione dei propri clienti.

CLIENTE (CODCLI, NOME, CITTÀ)

VEICOLO (TARGA, TIPO, CODCLI)

IMPIANTO (LOCALITÀ, NUMEROLINEE, DATAAPERTURA)

LAVAGGIO (TARGA, DATA, ORAMINUTO, LOCALITÀ, COSTO)

- Scrivere una query SQL che estrae il Nome dei clienti di Bergamo che non hanno mai lavato un motociclo (un veicolo di Tipo "Motociclo").
- Formulare in SQL l'interrogazione che per ogni cliente restituisce il primo lavaggio effettuato.

Esercizio (tde 7-9-2009)

- Il seguente schema descrive una base di dati relativa ad una catena di hotel che intende avviare un programma di fidelizzazione dei propri clienti.

CLIENTE (CODCLI, NOME, CITTÀ)

HOTEL (LOCALITÀ, NUMEROCAMERE, DATAAPERTURA, COSTODOPPIA, COSTOSINGOLA)

PRENOTAZIONE (CODCLI, LOCALITÀ, DATAINIZIO, NUMEROGIORNI, SINGOLAODOPPIA)

- Scrivere una query SQL che estrae il Nome dei clienti di Bergamo che non hanno mai prenotato una camera doppia.
- Formulare in Algebra Relazionale, Calcolo Relazionale, Datalog e SQL l'interrogazione che restituisce gli hotel che hanno avuto almeno una prenotazione il primo giorno di apertura.

Esercizio (tde 9-2-2010)

- Il seguente schema descrive i dati di una carrozzeria e consiste di due tabelle (chiavi in maiuscolo):

Cliente(CODICEFISCALE, Nome, TargaVeicolo, Indirizzo)

Riparazione(CODFISCLIENTE, DATAINIZIO, DataFine, Descrizione, Costo)

1. Scrivere una query in SQL che estrae i clienti che hanno effettuato meno di due riparazioni nel 2009 (zero o una)
2. Scrivere una query in SQL che determina il nome del cliente che complessivamente ha speso di più nell' officina.

1.
SELECT *
FROM Cliente
WHERE CODICEFISCALE IN (SELECT CODFISCLIENTE
FROM Riparazione
GROUP BY CODFISCLIENTE
HAVING count(*)<2)

2.
SELECT C.Nome
FROM Cliente C JOIN Riparazione R ON C.CODICEFISCALE=R.CODFISCLIENTE
GROUP BY C.CODICEFISCALE, C.Nome
HAVING SUM(Costo) >= (SELECT SUM(Costo)
FROM Riparazione
GROUP BY CODFISCLIENTE)

oppure

SELECT Nome
FROM Cliente
WHERE CODICEFISCALE IN (SELECT CODFISCLIENTE
FROM Riparazione
GROUP BY CODFISCLIENTE
HAVING SUM(Costo) >= (SELECT SUM(Costo)
FROM Riparazione
GROUP BY CODFISCLIENTE))

Esercizio (tde 25-2-2010)

- Il seguente schema rappresenta le informazioni riguardo alla gestione del personale e delle trasferte:

DIPENDENTE (Matricola, Cognome, Nome, Bonus)

TRASFERITA (Matricola, DataPartenza, DataRitorno, Destinazione, Costo)

- Scrivere in SQL l'interrogazione che estrae per ogni dipendente la trasferta più costosa.
- Scrivere in SQL l'interrogazione che estrae il dipendente con più trasferte iniziate nel gennaio 2010.

Esercizio (tde 9-7-2010)

- La seguente base di dati descrive i voli di una compagnia internazionale. Si assuma che: un passeggero sia presente su un volo se e solo se ha una prenotazione per quel volo ed ha successivamente fatto check-in; i ritardi siano espressi in minuti; sommando un orario ad un ritardo si ottenga un nuovo orario; la differenza tra due orari restituisca come risultato un intervallo in minuti; un passeggero arrivi e parta una sola volta in un determinato giorno da un determinato aeroporto.

VOLO(NUMERO, DATA, COMPAGNIA, LOC-PARTENZA, LOC-ARRIVO, ORA-PARTENZA, ORA-ARRIVO,

RITARDO-PARTENZA, RITARDO-ARRIVO,

FLAGNOTTURNO)

PRENOTAZIONE(ID-PASS, NUMERO-VOLO, DATA, NOMINATIVO, RECAPITO, CITTA' , NAZIONALITA')

CHECK-IN(ID-PASS, NUMERO-VOLO, DATA, POSTOASSEGNATO)

- Estrarre la compagnia che ha accumulato il maggior ritardo medio in arrivo a Linate nel mese di maggio 2010.
- Calcolare il tempo medio di permanenza in aeroporto, tenendo conto dei ritardi, dei passeggeri che hanno transitato il 3/5/2010 da Linate (erano su un aereo che è arrivato a Linate e sono successivamente ripartiti nella medesima giornata).

Esercizio (tde 10-9-2010)

- La seguente base di dati descrive i dati di un concorso a premi. La tabella PUNTI memorizza giorno per giorno i punti raccolti da ogni concorrente.
CLIENTI(CODICECLIENTE, NOME, COGNOME, INDIRIZZO, CITTA' , NAZIONALITA')
PUNTI(CODICECLIENTE, DATA, PUNTI)
PREMIO(CODICEPREMIO, PUNTINECESSARI)
- Estrarre la somma dei punti accumulati da tutti i clienti di Milano.
- Estrarre il premio che richiede più punti.

Esercizio (tde 8-2-2011)

- Sia dato il seguente database relazionale, relativo ad un archivio musicale:
ARTISTA (NomeArtista, DataDiNascita, Genere)
ALBUM (TitoloAlbum, NomeArtista, Anno)
CANZONE (Titolo, TitoloAlbum, Durata, Posizione)
- Si assuma che il campo **Durata** contenga la durata della canzone espressa in secondi.
- Si scriva in SQL l'interrogazione che estrae gli album di artisti rock realizzati nel 2010 e la loro durata complessiva
- Si scriva in SQL l'interrogazione che estrae l'elenco degli artisti le cui canzoni hanno tutte durata inferiore a 4 minuti

Esercizio (tde 24-2-2011)

- Il seguente schema descrive la base di dati di un concessionario di autoveicoli multimarca.

VEICOLO (CODICEVEICOLO, MARCA, MODELLO, ALLESTIMENTO)

CLIENTE (CODICEFISCALE, NOME, CITTÀ, PROVINCIA, DATANASCITA)

VENDITA (CODICEFISCALE, CODICEVEICOLO, DATA,
NUMEROFATTURA, IMPORTO)

- Estrarre la Marca di Veicolo che produce modelli mai venduti in provincia di “Milano”
- Estrarre l'elenco delle vendite aventi un importo superiore all'importo del 90% delle vendite

Esercizio (tde 8-7-2011)

- Il seguente schema descrive la base di dati di una libreria.
AUTORE (NOME, ISBNLIBRO)
LIBRO (ISBN, TITOLO, EDITORE, ANNODIPUBBLICAZIONE)
CLIENTE (CODICEFISCALE, NOME, CITTÀ, PROVINCIA, DATANASCITA)
VENDITA (CODICEFISCALE, ISBNLIBRO, DATA, IMPORTO)
- Estrarre il titolo del libro più venduto a clienti residenti in provincia di “Milano”
- Estrarre l'elenco dei clienti che hanno comprato più di 30 libri nel 2010

Esercizio (tde 12-9-2011)

- Il seguente schema descrive la base di dati di una libreria.
AUTORE (NOME, ISBNLIBRO)
LIBRO (ISBN, TITOLO, EDITORE, GENERE, ANNODIPUBBLICAZIONE)
CLIENTE (CODICEFISCALE, NOME, CITTÀ, PROVINCIA, DATANASCITA)
VENDITA (CODICEFISCALE, ISBNLIBRO, DATA, IMPORTO)
- Estrarre il nome dell' autore che ha totalizzato il maggior incasso nel 2010
- Estrarre le città in cui non risiede alcun cliente che abbia comprato un libro di genere "saggio"

Esercizio (tdeB 16-7-2009)

- La seguente base di dati rappresenta i rapporti di amicizia in un social network. Quando un utente A chiede a un utente B di diventare suo amico, si inserisce un record in RICHIESTA in Stato “pending”. Se B conferma, lo stato passa a “confirmed”, e si inseriscono *due* record in AMICI (per dire che A è amico di B e che B è amico di A, con i valori di Usr1 e Usr2 scambiati). Se B rifiuta, lo stato diventa “ignored”, ma la richiesta non è mai cancellata:

MEMBRO (Username, Nome, Cognome, Sesso, Città, DataNascita)

RICHIESTA (Richiedente, Usr2, Stato, TestoDiSaluto)

AMICI (Usr1, Usr2)

1. Estrarre in SQL tutte le coppie di membri della stessa città che hanno un amico in comune ma non sono amici tra loro
2. Estrarre in SQL il membro di sesso femminile che ha avuto il maggior numero di richieste rifiutate

1.

Verifichiamo che A e C siano amici, che siano amici di un qualche B, e che non siano amici tra loro:

```
select A.Username, C.Username
from Membro A, Membro C, Amici AB, Amici BC
where A.Città = C.Città and
      A.Username = AB.Usr1 and AB.Usr2 = BC.Usr1 and C.Username = BC.Usr2 and
      ( A.Username, C.Username ) not in ( select * from Amici )
```

2.

```
select Username, Nome, Cognome
from Membro M join Richiesta on Username = Richiedente
where M.Sesso = 'F' and Stato = 'ignored'
group by Username, Nome, Cognome
having count(*) >= ALL ( select count(*)
                        from Membro M join Richiesta on Username=Richiedente
                        where M.Sesso = 'F' and Stato = 'ignored'
                        group by Username, Nome, Cognome )
```

Esercizio (tdeB 16-9-2009)

- La seguente base di dati rappresenta i voti registrati dagli studenti di una università italiana:

STUDENTE (Matr, Nome, Cognome, Sesso, Città, DataNascita)

ESAME (Matr, CodCorso, Data, Voto, Lode)

CORSO (CodCorso, Titolo, NomeDocente, CFU, Anno, Semestre)

1. Estrarre in SQL le matricole degli studenti che hanno preso almeno due volte 30 e almeno due volte 18.
2. Estrarre in SQL le coppie di studenti che in tutti gli esami sostenuti da entrambi hanno preso lo stesso voto

1.

```
select Matr
from Esame E
where 1 < ( select count(*)
            from Esame
            where Matr = E.Matr and Voto = 18 )
and 1 < ( select count(*)
            from Esame
            where Matr = E.Matr and Voto = 30 )
```

2.

```
select s1.Matricola, s2.Matricola
from Studente s1, Studente s2
where s1.Matr <> s2.Matr and
not exists ( select *
            from Esame e1 join Esame e2 on e1.CodCorso = e2.CodCorso
            where e1.Matricola = s1.Matricola and
                  e2.Matricola = s2.Matricola and
                  e1.Voto <> e2.Voto )
```

Esercizio (tdeB 16-9-2009)

- Si consideri la solita base di dati, relativa alla registrazione degli esami in una università lombarda:

STUDENTE (Matricola, Nome, Cognome, DataNascita, CittàNascita)

ESAME (Matr, CodCorso, Data, Voto)

CORSO (Codice, Nome, CFU, MatrDocente)

DOCENTE (Matricola, Nome, Cognome, DataNascita, CittàNascita)

1. Estrarre in SQL nome e cognome dei docenti titolari di almeno due corsi da 10 CFU
2. Estrarre in SQL Nome e Cognome degli studenti che non hanno *mai* preso due volte lo stesso voto. [Cioè che non hanno *ancora* preso due volte lo stesso voto... al più tardi alla registrazione del 15° esame, infatti, inevitabilmente almeno un voto si ripete]

1.

```
select Nome, Cognome
from Docente join Corso on Matricola = MatrDocente
where CFU = 10
group by Matricola, Nome, Cognome
having count(*) > 1
```

Preferendo una query annidata, senza join:

```
select Nome, Cognome
from Docente
where Matricola in ( select MatrDocente
                    from Corso
                    where CFU = 10
                    group by MatrDocente
                    having count(*) > 1 )
```

2.

```
select Nome, Cognome
from Studente S
where Matricola not in (select Matricola
                       from Esame E1 join Esame E2 on E1.Matr=E2.Matr
                       where E1.Codice <> E2.Codice and E1.Voto = E2.Voto )
```

Esercizio (tdeB 25-2-2010)

- La seguente base di dati rappresenta i dati relativi a un festival annuale dedicato alla canzone italiana. Si assume che i titoli siano univoci nella storia della manifestazione (dal 1951 ad oggi):

CANTANTE (NomeArte, Nome, Cognome, DataNascita, CittàNascita)

CANZONE (Titolo, Anno, Interprete, DirettoreOrchestra)

AUTORE (TitoloCanzone, NomeAutore)

CLASSIFICA (Titolo, Anno, Posizione)

- Estrarre in SQL gli autori che hanno partecipato alla scrittura di più di quattro canzoni in una stessa edizione del festival
- Estrarre in SQL il cantante che è arrivato secondo il maggior numero di volte

Esercizio (tdeB 25-2-2010)

- La seguente base di dati rappresenta l'orario (con periodicità quotidiana) dei voli di varie compagnie aeree, con le prenotazioni e poi gli effettivi check-in dei clienti. La prenotazione è sempre obbligatoria.

VOLO (Codice, Compagnia, AeropPartenza, AeropArrivo, OraPart, OraArrivo)

PRENOTAZIONE (IdPasseggero, CodiceVolo, DataVolo, Nome, Cognome, DataNascita)

CHECKIN (IdPasseggero, CodiceVolo, DataVolo, Posto, OraEffettivaCheckIn, GruppoPriorità, Note)

- Estrarre in SQL il numero dei passeggeri minorenni effettivamente imbarcati sul volo AZ-284 del 21 aprile 1991
- Estrarre in SQL nome e cognome dei passeggeri che avevano prenotato qualche volo per il mese di giugno 2010, ma poi non sono partiti

1.

```
select count(*)
```

```
from CheckIn C join Prenotazione P on C.IdPasseggero = P.IdPasseggero
```

```
where C.CodiceVolo = 'AZ-284' and C.DataVolo = 21.4.1991
```

```
and P.CodiceVolo = 'AZ-284' and P.DataVolo = 21.4.1991
```

```
and DataNascita > 21.4.1973
```

2.

Esercizio (tdeB 25-2-2010)

- La seguente base di dati rappresenta i voti registrati e le tesi assegnate in una università italiana. Le tesi sono sempre collegate a un corso, e sono inserite nel database al momento dell'assegnazione. L'attributo *Conclusa*, inizialmente pari a *false*, assume valore *true* dopo il superamento dell'esame di laurea.
STUDENTE (Matr, Nome, Cognome, Sesso, Città, DataNascita)
ESAME (Matr, CodCorso, Data, Voto, Lode)
CORSO (CodCorso, Titolo, NomeDocente, CFU, Anno, Semestre)
TESI (Matr, Titolo, CodCorsoCollegato, DataInizio, Conclusa)
1. Estrarre in SQL Nome e Cognome degli studenti che hanno scelto una tesi collegata a un corso del primo anno per il quale hanno preso 18
 2. Estrarre in SQL la matricola degli studenti già laureati che hanno iniziato la tesi solo dopo la registrazione del loro ultimo esame

Esercizio (tdeB 9-2-2011)

- Si consideri il seguente database, definito in supporto al ricettario contenuto nel sito Web di un celebre cyberenogastrocromatodietologo. Le dosi sono riferite a porzioni per una persona:

RICETTA (NomeR, Categoria, Origine, DescrizioneProcedimento)

COMPOSIZIONE (NomeR, NomeI, QuantitàGr)

INGREDIENTE (NomeI, Colore, CaloriePerGrammo)

- Estrarre in SQL i nomi dei piatti che per la cui preparazione occorrono almeno un ingrediente bianco, un ingrediente rosso, e un ingrediente verde.
- Estrarre in SQL il nome del piatto più calorico (considerando il contributo di tutti gli ingredienti)

Esercizio (tdeB 25-2-2011)

- La seguente base di dati è relativa a un festival annuale dedicato alla canzone italiana. Si assume per semplicità che i titoli delle canzoni e i nomi delle persone siano univoci nella storia della manifestazione. Si noti che le canzoni possono avere più di un interprete e più di un autore.

ARTISTA (Nome, DataNascita, CittàNascita)

CANZONE (Titolo, Anno, DirettoreOrchestra, PosizioneClassificaFinale)

AUTORE (TitoloC, NomeAutore)

CANTANTE (TitoloC, NomeInterprete)

- Estrarre in SQL gli artisti che hanno vinto al festival in qualità di interpreti e poi, in un'edizione successiva, in qualità di autori.
- Estrarre in SQL il l' autore che ha scritto il maggior numero di canzoni vincitrici.

Esercizio (tdeB 29-7-2011)

- Per agevolare la logistica globale, nello stato libero (federale) di Bananas ogni ministero ha sede in un comune diverso, e viene frequentemente spostato. Del resto anche i ministri sono spesso sostituiti. Per localizzare i ministri e i ministeri, quindi, la stessa pubblica amministrazione si serve di un database:

COMUNE (NomeC, Provincia, Regione, NumAbitanti)

DICASTERO (NomeD, Sede, Ministro, NumDipendenti, Budget,
DataUltimoTrasferimento)

MINISTRO (NomeM, DataNascita, ComuneResidenza)

- Estrarre in SQL i nomi dei ministri che risiedono nella stessa regione in cui ha sede il dicastero di cui sono titolari
- Estrarre in SQL il nome del più popoloso tra i comuni che non sono sede di un ministero

Esercizio (tdeB 14-9-2011)

- La seguente base di dati è relativa alla registrazione degli esami in una università lombarda:

STUDENTE (Matricola, Nome, Cognome, DataNascita, CittàNascita)

ESAME (Matr, CodCorso, Data, Voto)

CORSO (Codice, Nome, Anno, CFU, NomeDocente)

- Estrarre in SQL Nome, Cognome e Matricola degli studenti che hanno sostenuto gli esami *sempre e solo* in appelli di settembre [*le funzioni year(), month() e day() restituiscono interi estratti dai relativi campi delle date*]
- Estrarre in SQL le matricole degli studenti che hanno sostenuto più esami del 2° anno che del 1° anno.