

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 con argomento una query intera

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

HAVING max(X)• (e basta) → *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

Distributori automatici

Si consideri il seguente schema di base di dati, che descrive le vendite di bevande effettuate da una catena di punti vendita costituiti da soli distributori automatici.

BEVANDA (Codice, Nome, Prezzo, QtaCL)

VENDITA (CodDistributore, Data, Ora, CodBevanda, QtaZucchero, UsoChiavetta)

DISTRIBUTORE (Codice, CapienzaCL, DataRifornimento, OraRifornimento)

Le bevande sono “caffè espresso”, “caffè lungo”, “cioccolata”, ...

QtaCL esprime, in centilitri, la quantità di bevanda.

L'attributo *QtaZucchero* specifica, su una scala da 0 a 5 la quantità di zucchero scelta durante l'acquisto

UsoChiavetta assume valore TRUE oppure FALSE a seconda che l'acquisto sia stato effettuato utilizzando la chiavetta o no

CapienzaCL esprime, in centilitri, la capienza totale di bevande del distributore.

BEVANDA (Codice, Nome, Prezzo, QtaCL)

VENDITA (CodDistributore, Data, Ora, CodBevanda, QtaZucchero, UsoChiavetta)

DISTRIBUTORE (Codice, CapienzaCL, DataUltimoRifornimento, OraUltimoRifornimento)

Trovare i codici delle bevande vendute il 15 luglio del 2016

```
SELECT distinct CodBevanda
```

```
FROM Vendita
```

```
WHERE Data = '15/7/2016'
```

BEVANDA (Codice, Nome, Prezzo, QtaCL)

VENDITA (CodDistributore, Data, Ora, CodBevanda, QtaZucchero, UsoChiavetta)

DISTRIBUTORE (Codice, CapienzaCL, DataUltimoRifornimento, OraUltimoRifornimento)

Trovare il codice del distributore e il codice della bevanda delle vendite effettuate nell'agosto 2016

```
SELECT distinct CodDistributore, CodBevanda
```

```
FROM Vendita
```

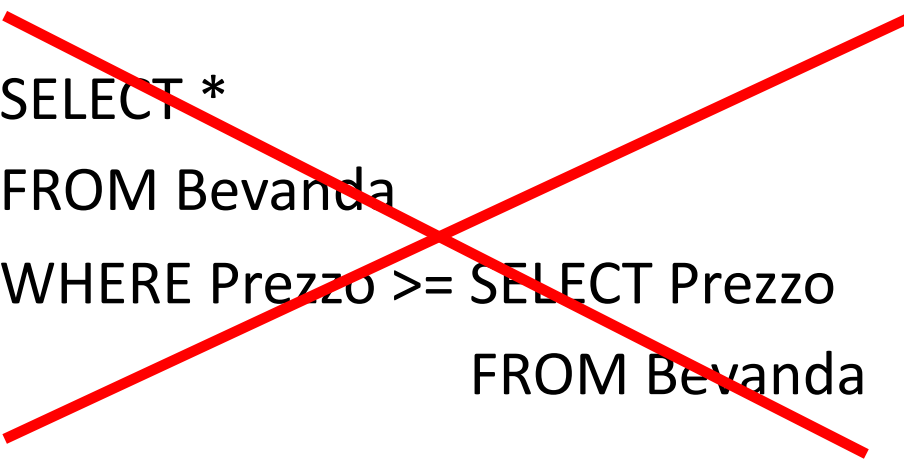
```
WHERE Data >= '1/8/2016' AND Data <= '31/8/2016'
```

BEVANDA (Codice, Nome, Prezzo, QtaCL)

VENDITA (CodDistributore, Data, Ora, CodBevanda, QtaZucchero, UsoChiavetta)

DISTRIBUTORE (Codice, CapienzaCL, DataUltimoRifornimento, OraUltimoRifornimento)

Trovare la bevanda più costosa



```
SELECT *  
FROM Bevanda  
WHERE Prezzo >= SELECT Prezzo  
                  FROM Bevanda
```

```
SELECT *  
FROM Bevanda  
WHERE Prezzo >= ALL SELECT Prezzo  
                  FROM Bevanda
```

BEVANDA (Codice, Nome, Prezzo, QtaCL)

VENDITA (CodDistributore, Data, Ora, CodBevanda, QtaZucchero, UsoChiavetta)

DISTRIBUTORE (Codice, CapienzaCL, DataUltimoRifornimento, OraUltimoRifornimento)

Trovare le vendite fatte dai distributori
successivamente al rifornimento

```
SELECT CodDistributore, Data, Ora, CodBevanda
FROM Vendita JOIN Distributore
      ON CodDistributore=Codice
WHERE Data > DataRifornimento
      OR Data = DataRifornimento
      AND Ora > OraRifornimento
```

BEVANDA (Codice, Nome, Prezzo, QtaCL)

VENDITA (CodDistributore, Data, Ora, CodBevanda, QtaZucchero, UsoChiavetta)

DISTRIBUTORE (Codice, CapienzaCL, DataUltimoRifornimento, OraUltimoRifornimento)

Trovare i distributori che non hanno mai venduto niente

SELECT *

FROM Distributore

WHERE Codice NOT IN

SELECT CodDistributore FROM Vendita

BEVANDA (Codice, Nome, Prezzo, QtaCL)

VENDITA (CodDistributore, Data, Ora, CodBevanda, QtaZucchero, UsoChiavetta)

DISTRIBUTORE (Codice, CapienzaCL, DataUltimoRifornimento, OraUltimoRifornimento)

Trovare i distributori che non hanno fatto nessuna vendita successiva al loro rifornimento

SELECT *

FROM Distributore

WHERE Codice NOT IN

SELECT CodDistributore

FROM Vendita JOIN Distributore

ON CodDistributore=Codice

WHERE Data > DataRifornimento

OR Data = DataRifornimento

AND Ora > OraRifornimento

BEVANDA (Codice, Nome, Prezzo, QtaCL)

VENDITA (CodDistributore, Data, Ora, CodBevanda, QtaZucchero, UsoChiavetta)

DISTRIBUTORE (Codice, CapienzaCL, DataUltimoRifornimento, OraUltimoRifornimento)

Trovare i distributori che non hanno mai venduto la bevanda di codice
"B1"

SELECT *

FROM Distributore

WHERE Codice NOT IN

SELECT CodDistributore

FROM Vendita

WHERE CodBevanda = 'B1'

BEVANDA (Codice, Nome, Prezzo, QtaCL)

VENDITA (CodDistributore, Data, Ora, CodBevanda, QtaZucchero, UsoChiavetta)

DISTRIBUTORE (Codice, CapienzaCL, DataUltimoRifornimento, OraUltimoRifornimento)

Trovare i distributori che non hanno **mai** venduto “*tè al limone*”.

SELECT *

FROM Distributore

WHERE Codice NOT IN

 SELECT CodDistributore

 FROM Vendita JOIN Bevanda

 ON CodBevanda = Codice

 WHERE Nome='tè al limone'

BEVANDA (Codice, Nome, Prezzo, QtaCL)

VENDITA (CodDistributore, Data, Ora, CodBevanda, QtaZucchero, UsoChiavetta)

DISTRIBUTORE (Codice, CapienzaCL, DataUltimoRifornimento, OraUltimoRifornimento)

Trovare i distributori che hanno venduto **solo** “*cioccolata*”

```
SELECT CodDistributore
```

```
FROM Vendita
```

```
WHERE CodDistributore NOT IN
```

```
    (SELECT CodDistributore
```

```
      FROM Vendita JOIN Bevanda
```

```
        ON CodBevanda = Codice
```

```
      WHERE Nome <> 'cioccolata')
```

BEVANDA (Codice, Nome, Prezzo, QtaCL)

VENDITA (CodDistributore, Data, Ora, CodBevanda, QtaZucchero, UsoChiavetta)

DISTRIBUTORE (Codice, CapienzaCL, DataUltimoRifornimento, OraUltimoRifornimento)

Trovare le bevande che sono state vendute almeno due volte

```
SELECT distinct V1.CodBevanda
```

```
FROM Vendita V1, Vendita V2
```

```
WHERE V1.CodBevanda=V2.CodBevanda
```

```
    AND (V1.CodDistributore <> V2.CodDistributore
```

```
        OR V1.Data <> V2.Data OR V1.Ora <> V2.Ora)
```

```
SELECT CodBevanda
```

```
FROM Vendita
```

```
GROUP BY CodBevanda
```

```
HAVING count(*)>=2
```

BEVANDA (Codice, Nome, Prezzo, QtaCL)

VENDITA (CodDistributore, Data, Ora, CodBevanda, QtaZucchero, UsoChiavetta)

DISTRIBUTORE (Codice, CapienzaCL, DataUltimoRifornimento, OraUltimoRifornimento)

Trovare i distributori che hanno fatto una sola vendita

```
SELECT CodDistributore
```

```
FROM Vendita
```

```
WHERE CodDistributore NOT IN
```

```
    SELECT V1.CodDistributore
```

```
    FROM Vendita V1, Vendita V2
```

```
    WHERE V1.CodDistributore=V2.CodDistributore
```

```
        AND (V1.Data <> V2.Data OR V1.Ora <> V2.Ora)
```

BEVANDA (Codice, Nome, Prezzo, QtaCL)

VENDITA (CodDistributore, Data, Ora, CodBevanda, QtaZucchero, UsoChiavetta)

DISTRIBUTORE (Codice, CapienzaCL, DataUltimoRifornimento, OraUltimoRifornimento)

Trovare i distributori che hanno fatto almeno due vendite, ma non hanno mai venduto “*caffè lungo*”

```
SELECT V1.CodDistributore
```

```
FROM Vendita V1, Vendita V2
```

```
WHERE V1.CodDistributore=V2.CodDistributore
```

```
AND (V1.Data <> V2.Data OR V1.Ora <> V2.Ora)
```

```
AND V1.CodDistributore NOT IN
```

```
    (SELECT CodDistributore
```

```
    FROM Vendita JOIN Bevanda
```

```
        ON CodBevanda = Codice
```

```
    WHERE Nome = 'caffè lungo')
```

BEVANDA (Codice, Nome, Prezzo, QtaCL)

VENDITA (CodDistributore, Data, Ora, CodBevanda, QtaZucchero, UsoChiavetta)

DISTRIBUTORE (Codice, CapienzaCL, DataUltimoRifornimento, OraUltimoRifornimento)

Trovare i distributori che hanno venduto tutte le bevande

```
SELECT CodDistributore
```

```
FROM Vendita
```

```
GROUP BY CodDistributore
```

```
HAVING count(distinct CodBevanda) = (SELECT count(*)
```

```
FROM Bevanda
```


Aeroporti

AEROPORTO (Citta, Nazione, NumPiste)

VOLO (IdVolo, GiornoSett, CittaPart, OraPart,
CittaArr, 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 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
```

Rinfreschiamoci la memoria

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

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

```
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.Citta=CittaPar)  
      JOIN AEROPORTO A2  
      ON CittaArr=A2.Citta  
WHERE IdVolo= 'AZ274'
```

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

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

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

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

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

NO!

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

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

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

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

NO!

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

Ad esempio si può usare una query annidata

```
SELECT Citta, 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 Citta, 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 Città, Nazione, NumPiste
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
and NumPiste > 2)

Trovare gli aeroporti da cui partono voli *internazionali*

Trovare gli aeroporti da cui partono voli *internazionali*

```
SELECT CittaPar
FROM (AEROPORTO AS A1 JOIN VOLO
      ON CittaPar = A1.Citta)
     JOIN AEROPORTO AS A2
      ON CittaArr = A2.Citta
WHERE A1.Nazione <> A2.Nazione
```

Trovare gli aeroporti da cui partono voli *internazionali*

```
SELECT  DISTINCT CittaPar
FROM    (AEROPORTO AS A1 JOIN VOLO
        ON CittaPar = A1.Citta)
        JOIN AEROPORTO AS A2
        ON CittaArr = A2.Citta
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 CittaPar=A1.Citta)  
        JOIN AEROPORTO AS A2  
        ON CittaArr=A2.Citta  
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 CittaPar=A1.Citta)
        JOIN AEROPORTO AS A2
        ON CittaArr=A2.Citta
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 CittaPar=A1.Citta)  
        JOIN AEROPORTO AS A2  
        ON CittaArr=A2.Citta  
WHERE   A1.Nazione <> A2.Nazione  
        and GiornoSett = 'Giovedì'
```

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

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

?

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

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

GROUP BY CittaPar
```

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

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

?

.....

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

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

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

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

?

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

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

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

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

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

```
CREATE VIEW VOLIGIOVCITTA (Citta,Num)
SELECT      CittaPart, count(*)
FROM  VOLO
WHERE      GiornoSett = 'Giovedì' AND CittaPart IN
( SELECT CittaPart FROM VOLO
  GROUP BY CittaPart HAVING count(*) >= 100 )
GROUP BY CittaPart

SELECT avg(Num)
FROM VOLIGIOVCITTA
```

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

```
CREATE VIEW VOLIGIOVCITTA (Citta, Nazione, Num)
SELECT      CittaPart, Nazione, count(*)
FROM  VOLO JOIN AEROPORTO ON Citta=CittaPart
WHERE      GiornoSett = 'Giovedì' AND CittaPart IN
( SELECT CittaPart FROM VOLO
  GROUP BY CittaPart HAVING count(*) >= 100 )
GROUP BY CittaPart,Nazione
```

```
SELECT Nazione, avg(Num)
FROM VOLIGIOVCITTA
GROUP BY Nazione
```


Trovare **la media delle medie per nazione** di voli del giovedì di ogni aeroporto *da cui partano almeno 100 voli a settimana*

```
CREATE VIEW VOLIGIOVCITTA (Citta, Nazione, Num)
SELECT      CittaPart, Nazione, count(*)
FROM  VOLO JOIN AEROPORTO ON Citta=CittaPart
WHERE      GiornoSett = 'Giovedì' AND CittaPart IN
    ( SELECT CittaPart FROM VOLO
      GROUP BY CittaPart HAVING count(*) >= 100 )
GROUP BY CittaPart,Nazione
```

```
CREATE VIEW VOLIGIOVNAZ(Nazione, Num)
SELECT Nazione, avg(Num)
FROM VOLIGIOVCITTA
GROUP BY Nazione
```

```
SELECT avg(Num)
FROM VOLIGIOVNAZ
```

Filmografie

REGISTA (Nome, DataNascita, Nazionalità)

ATTORE (Nome, DataNascita, Nazionalità)

INTERPRETA (Attore, Film, Personaggio)

FILM (Titolo, NomeRegista, Anno)

PROIEZIONE (NomeCin, CittaCin, TitoloFilm)

CINEMA (Citta, NomeCinema, Sale, Posti)

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

REGISTA (Nome, DataNascita, Nazionalità)
ATTORE (Nome, DataNascita, Nazionalità)
INTERPRETA (Attore, Film, Personaggio)
FILM (Titolo, NomeRegista, Anno)
PROIEZIONE(NomeCin, CittaCin, TitoloFilm)
CINEMA (Citta, NomeCinema, Sale, Posti)

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

REGISTA (Nome, DataNascita, Nazionalità)
ATTORE (Nome, DataNascita, Nazionalità)
INTERPRETA (Attore, Film, Personaggio)
FILM (Titolo, NomeRegista, Anno)
PROIEZIONE(NomeCin, CittaCin, TitoloFilm)
CINEMA (Citta, NomeCinema, Sale, Posti)

```
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

REGISTA (Nome, DataNascita, Nazionalità)
ATTORE (Nome, DataNascita, Nazionalità)
INTERPRETA (Attore, Film, Personaggio)
FILM (Titolo, NomeRegista, Anno)
PROIEZIONE(NomeCin, CittaCin, TitoloFilm)
CINEMA (Citta, NomeCinema, Sale, Posti)

```
SELECT DISTINCT Nazionalità
FROM REGISTA, FILM
WHERE Nome = NomeRegista AND 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

REGISTA (Nome, DataNascita, Nazionalità)
ATTORE (Nome, DataNascita, Nazionalità)
INTERPRETA (Attore, Film, Personaggio)
FILM (Titolo, NomeRegista, Anno)
PROIEZIONE (NomeCin, CittaCin, TitoloFilm)
CINEMA (Citta, NomeCinema, Sale, Posti)

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

REGISTA (Nome, DataNascita, Nazionalità)
ATTORE (Nome, DataNascita, Nazionalità)
INTERPRETA (Attore, Film, Personaggio)
FILM (Titolo, NomeRegista, Anno)
PROIEZIONE(NomeCin,CittaCin,TitoloFilm)
CINEMA (Citta,NomeCinema, Sale, Posti)

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

REGISTA (Nome, DataNascita, Nazionalità)
ATTORE (Nome, DataNascita, Nazionalità)
INTERPRETA (Attore, Film, Personaggio)
FILM (Titolo, NomeRegista, Anno)
PROIEZIONE(NomeCin, CittaCin, TitoloFilm)
CINEMA (Citta, NomeCinema, Sale, Posti)

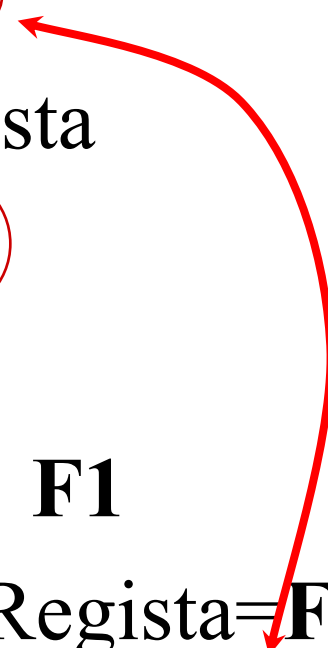
```
SELECT  NomeRegista
FROM    FILM AS F
WHERE   Anno='1993'
GROUP BY NomeRegista
HAVING  count(*) >
        ( SELECT count(*)
          FROM FILM AS F1
          WHERE F1.NomeRegista=F.NomeRegista
            AND Anno='1992')
```


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

REGISTA (Nome, DataNascita, Nazionalità)
ATTORE (Nome, DataNascita, Nazionalità)
INTERPRETA (Attore, Film, Personaggio)
FILM (Titolo, NomeRegista, Anno)
PROIEZIONE(NomeCin,CittaCin,TitoloFilm)
CINEMA (Citta,NomeCinema, Sale, Posti)

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 CittaCin='Milano')
AND Titolo in ( SELECT TitoloFilm
                   FROM PROIEZIONE
                   WHERE CittaCin='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 F
    on Nome=NomeRegista
    join PROIEZIONE P
        on F.Titolo=P.TitoloFilm
WHERE CittaCin='Milano'
AND Titolo in ( SELECT TitoloFilm
                  FROM PROIEZIONE
                  WHERE CittaCin='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 F
    on Nome=NomeRegista
join PROIEZIONE P1
    on F.Titolo=P1.TitoloFilm
join PROIEZIONE P2
    on F.Titolo=P2.TitoloFilm
WHERE P1.CittaCin='Milano'
AND P2.CittaCin='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 CittaCin='Milano'
                  AND CittaCin='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 CittaCin='Milano'
                  OR CittaCin='Torino')
```

Film proiettati nel maggior
numero di cinema di
Milano

REGISTA (Nome, DataNascita, Nazionalità)
ATTORE (Nome, DataNascita, Nazionalità)
INTERPRETA (Attore, Film, Personaggio)
FILM (Titolo, NomeRegista, Anno)
PROIEZIONE(NomeCin,CittaCin,TitoloFilm)
CINEMA (Citta,NomeCinema, Sale, Posti)

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 CittaCin='Milano'
GROUP BY TitoloFilm
HAVING count(*) >= ALL
( SELECT count(*)
  FROM PROIEZIONE
 WHERE CittaCin='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  CittaCin='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 di
un personaggio
nello stesso film

REGISTA (Nome, DataNascita, Nazionalità)
ATTORE (Nome, DataNascita, Nazionalità)
INTERPRETA (Attore, Film, Personaggio)
FILM (Titolo, NomeRegista, Anno)
PROIEZIONE(NomeCin,CittaCin,TitoloFilm)
CINEMA (Citta,NomeCinema, Sale, Posti)

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
```

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


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

REGISTA (Nome, DataNascita, Nazionalità)
ATTORE (Nome, DataNascita, Nazionalità)
INTERPRETA (Attore, Film, Personaggio)
FILM (Titolo, NomeRegista, Anno)
PROIEZIONE(NomeCin,CittaCin,TitoloFilm)
CINEMA (Citta,NomeCinema, Sale, Posti)

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

REGISTA (Nome, DataNascita, Nazionalità)
ATTORE (Nome, DataNascita, Nazionalità)
INTERPRETA (Attore, Film, Personaggio)
FILM (Titolo, NomeRegista, Anno)
PROIEZIONE(NomeCin,CittaCin,TitoloFilm)
CINEMA (Citta,NomeCinema, Sale, Posti)

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

```
SELECT DISTINCT Nome
FROM ATTORE A1, INTERPRETA I1
WHERE I1.Attore = A1.Nome
      AND Nazionalità = "Italiana" AND
      I1.Titolo NON IN (
        SELECT I2.Titoli
        FROM INTERPRETA I2,
             ATTORE A2
        WHERE A2.Nome = I2.NomeAttore
              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

REGISTA (Nome, DataNascita, Nazionalità)
ATTORE (Nome, DataNascita, Nazionalità)
INTERPRETA (Attore, Film, Personaggio)
FILM (Titolo, NomeRegista, Anno)
PROIEZIONE(NomeCin, CittaCin, TitoloFilm)
CINEMA (Citta, NomeCinema, Sale, Posti)

```
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

REGISTA (Nome, DataNascita, Nazionalità)
ATTORE (Nome, DataNascita, Nazionalità)
INTERPRETA (Attore, Film, Personaggio)
FILM (Titolo, NomeRegista, Anno)
PROIEZIONE(NomeCin, CittaCin, TitoloFilm)
CINEMA (Citta, NomeCinema, Sale, Posti)

```
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

Esami Universitari

STUDENTE (Matricola, Nome, Cognome, Indirizzo, Città)

ESAME (CodCorso, MatrStud, Voto, Lode)

CORSO (Codice, Nome, AnnoDiCorso, Facoltà, NumCrediti)

PROFESSORE(Matricola, Nome, Cognome, Città)

INSEGNAMENTO(CodCorso, MatrProf, AnnoAccademico,
NumeroStudenti)

- *Lode può valere 'yes' se e solo se il voto è 30*

**Trovare gli studenti che
hanno sostenuto
esattamente 10 esami**

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

```
SELECT MatrStud  
FROM Esame  
GROUP BY MatrStud  
HAVING COUNT(*)=10
```

**Trovare gli studenti che
non hanno mai sostenuto
nessun esame**

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

~~**SELECT MatrStud
FROM Esame
GROUP BY MatrStud
HAVING COUNT(*)=0**~~

**Trovare gli studenti che
non hanno mai sostenuto
nessun esame**

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

```
SELECT Matricola  
FROM Studente  
WHERE Matricola NOT IN (SELECT MatrStud  
                        FROM Esame)
```

Trovare le matricole dei professori che hanno insegnato in corsi di tutte le facoltà

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

```
SELECT I.MatrProf
FROM Insegnamento I join Corso C on Codice = CodCorso
GROUP BY MatrProf
HAVING COUNT(DISTINCT Facoltà) =
( SELECT COUNT(DISTINCT Facoltà)
  FROM Corso )
```

**Corsi in cui qualche voto
non è mai stato assegnato**

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

```
SELECT E.CodCorso  
FROM Esame E  
GROUP BY E.CodCorso  
HAVING COUNT (DISTINCT Voto) < 13
```

**Di ogni corso estratto
dalla query precedente, i
voti che non sono stati
assegnati**

STUDENTE(Matricola, Nome, Cognome, Indirizzo, Città)
ESAME(CodCorso, MatrStud, Voto, Lode)
CORSO(Codice, Nome, AnnoDiCorso, Facoltà, NumCrediti)
PROFESSORE(Matricola, Nome, Cognome, Città)
INSEGNAMENTO(CodCorso, MatrProf,
AnnoAccademico, NumeroStudenti)

SELECT DISTINCT C.Codice, E.Voto

FROM Corso C, Esame E

WHERE (C.Codice, E.Voto) NOT IN

**(SELECT CodCorso, Voto
FROM Esame)**

SELECT Codice, Voto FROM Corso, Esame

EXCEPT

SELECT CodCorso, Voto FROM Esame

Matricola, nome e cognome degli studenti di Milano che hanno superato esami per un totale di almeno 20 crediti e non hanno mai preso un voto minore di 28

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

```
SELECT S.Matricola, S.Nome, S.Cognome
FROM Studente S, Esame E, Corso C
WHERE S.Matricola = E.MatrStud AND E.CodCorso = C.Codice
AND S.Città='Milano'
AND S.Matricola NOT IN ( SELECT MatrStud
FROM Esame
WHERE Voto<28 )
GROUP BY S.Matricola, S.Nome, S.Cognome
HAVING SUM(C.NumCrediti) >= 20
```

Matricola, nome e cognome degli studenti di Milano che hanno superato esami per un totale di almeno 20 crediti e non hanno mai preso un voto minore di 28

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

```
SELECT Matricola, Nome, Cognome
FROM (Studiante JOIN Esame ON Matricola=MatrStud)
      JOIN Corso ON CodCorso=Codice
WHERE Città='Milano'
GROUP BY Matricola, Nome, Cognome
HAVING sum(NumCrediti)>20 AND min(Voto)=28
```

Il corso con il maggior numero di studenti

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

```
SELECT CodCorso
FROM Insegnamento
WHERE NumeroStudenti = (SELECT MAX(NumeroStudenti)
                        FROM Insegnamento)
```

Oppure:

```
SELECT CodCorso
FROM Insegnamento
WHERE NumeroStudenti >= ALL (SELECT NumeroStudenti
                             FROM Insegnamento)
```

Le edizioni dei corsi con il maggior numero di studenti

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

SELECT CodCorso, MatrProf, AnnoAccademico

FROM Insegnamento I1

WHERE NumeroStudenti >= **ALL**

(SELECT NumeroStudenti

FROM Insegnamento I2

WHERE I1.CodCorso=I2.CodCorso)

Le edizioni dei corsi del primo anno con il maggior numero di studenti

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

```
SELECT C1.Codice
FROM Corso C1, Insegnamento I1
WHERE C1.Codice = I1.CodCorso AND C1.AnnoDiCorso = 1
AND I1.NumeroStudenti = (SELECT MAX(NumeroStudenti)
FROM Corso C2, Insegnamento I2
WHERE C2.Codice = I2.CodCorso
AND C2.AnnoDiCorso=1)
```

Per ogni facoltà, corsi con il minor numero di studenti

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

```
SELECT C1.Facoltà, I1.NumeroStudenti
FROM Corso C1, Insegnamento I1
WHERE C1.Codice = I1.CodCorso AND
I1.NumeroStudenti = (SELECT MIN(I2.NumeroStudenti)
FROM Corso C2, Insegnamento I2
WHERE C2.Codice = I2.CodCorso AND
C2.Facoltà = C1.Facoltà )
```

Oppure

Per ogni facoltà, corsi con il minor numero di studenti

```
STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
               AnnoAccademico,NumeroStudenti)
```

Oppure (usando il costruttore di tupla):

```
SELECT C1.Facoltà, I1.NumeroStudenti
FROM Corso C1, Insegnamento I1
WHERE C1.Codice = I1.CodCorso AND
  (C1.Facoltà, I1.NumeroStudenti) IN
  (SELECT C2.Facoltà, MIN(I2.NumeroStudenti)
   FROM Corso C2, Insegnamento I2
   GROUP BY C2.Facoltà)
```

Studenti che hanno preso più 27 che 24

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

```
SELECT MatrStud
FROM Esame E
WHERE Voto = 27
GROUP BY MatrStud
HAVING COUNT(*) > ( SELECT COUNT(*)
                     FROM Esame E2
                     WHERE E.MatrStud = E2.MatrStud
                     AND Voto = 24 )
```


Studenti che hanno preso più 30L che 30

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

```
SELECT MatrStud
FROM Esame E
WHERE Voto = 30 AND Lode = 'yes'
GROUP BY MatrStud
HAVING COUNT(*) > ( SELECT COUNT(*)
                     FROM Esame E2
                     WHERE E.MatrStud = E2.MatrStud
                        AND Voto = 30 AND Lode = 'no' )
```

Studenti che hanno preso più volte 27 rispetto a qualsiasi altro voto (complessivamente)

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

```
SELECT MatrStud
FROM Esame E
WHERE Voto = 27
GROUP BY MatrStud
HAVING COUNT(*) > ( SELECT COUNT(*)
                     FROM Esame
                     WHERE MatrStud = E.MatrStud
                     AND Voto <> 27 )
```

Equivalente a

Studenti che hanno preso più volte 27 rispetto a qualsiasi altro voto (complessivamente)

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

Equivalente a

```
SELECT MatrStud
FROM Esame E
WHERE Voto = 27
GROUP BY MatrStud
HAVING COUNT(*) > 0.5 * ( SELECT COUNT(*)
                           FROM Esame
                           WHERE MatrStud = E.MatrStud )
```

Studenti che hanno preso più volte 27 rispetto a qualsiasi altro voto (individualmente)

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

```
SELECT MatrStud
FROM Esame E
WHERE Voto = 27
GROUP BY MatrStud
HAVING COUNT(*) > ALL ( SELECT COUNT(*)
FROM Esame
WHERE MatrStud = E.MatrStud
AND Voto <> 27
GROUP BY Voto )
```

**La soluzione precedente
considera il 30 e il 30 e Lode
come un unico voto. Come si
fa a contare separatamente i
30 e i 30 e lode?**

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

```
SELECT MatrStud
FROM Esame E
WHERE Voto = 27
GROUP BY MatrStud
HAVING COUNT(*) > ALL ( SELECT COUNT(*)
FROM Esame
WHERE MatrStud = E.MatrStud
AND Voto <> 27
GROUP BY Voto, Lode)
```

**Matricola, nome e cognome
degli studenti che hanno
superato almeno 3 esami del
secondo anno ma meno di 3
esami del primo**

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

```
SELECT S.Matricola, S.Nome, S.Cognome
FROM Studente S, Esame E, Corso C
WHERE S.Matricola = E.MatrStud AND E.CodCorso = C.Codice
AND C.AnnoDiCorso = 2
AND S.Matricola NOT IN (SELECT E2.MatrStud
FROM Esame E2, Corso C2
WHERE E2.CodCorso = C2.Codice
AND C2.AnnoDiCorso = 1
GROUP BY E2.MatrStud
HAVING COUNT(*) >=3)

GROUP BY S.Matricola, S.Nome, S.Cognome
HAVING COUNT(*) >=3
```

**Matricola, nome e cognome
degli studenti che hanno
superato più esami del terzo
anno che del secondo**

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

```
SELECT S.Matricola, S.Nome, S.Cognome
FROM Studente S, Esame E, Corso C
WHERE S.Matricola = E.MatrStud AND E.CodCorso = C.Codice
      AND C.AnnoDiCorso = 3
GROUP BY E.MatrStud, S.Nome, S.Cognome
HAVING COUNT(*) > (SELECT COUNT(*)
                    FROM Esame E2, Corso C2
                    WHERE E2.CodCorso = C2.Codice
                      AND C2.AnnoDiCorso = 2
                      AND E2.MatrStud = E.MatrStud)
```

**Matricola, nome e cognome
degli studenti che hanno
superato più esami del terzo
anno che del secondo**

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

```
SELECT S.Matricola, S.Nome, S.Cognome
FROM Studente S, Esame E, Corso C
WHERE S.Matricola = E.MatrStud AND E.CodCorso = C.Codice
      AND C.AnnoDiCorso = 3
GROUP BY E.Matricola, S.Nome, S.Cognome
HAVING COUNT(*) > (SELECT COUNT(*)
                    FROM Esame E2, Corso C2
                    WHERE E2.CodCorso = C2.Codice
                      AND C2.AnnoDiCorso = 2
                      AND E2.MatrStud = E.MatrStud)
```


Il corso con la media più bassa

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

```
SELECT CodCorso, AVG(Voto)
FROM Esame
GROUP BY CodCorso
HAVING AVG(Voto) <= ALL (SELECT AVG(Voto)
                             FROM Esame
                             GROUP BY CodCorso)
```

oppure

Il corso con la media più bassa

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

Oppure con vista:

```
CREATE VIEW MediaPerCorso (Corso, Media) AS  
SELECT CodCorso, AVG(Voto)  
FROM Esame  
GROUP BY CodCorso  
  
SELECT Corso  
FROM MediaPerCorso  
WHERE Media = ( SELECT min( Media )  
                FROM MediaPerCorso )
```

Studenti con la media pesata più alta

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

```
CREATE VIEW MediaStudente (Matricola, Media) AS  
SELECT E.MatrStud, SUM(E.Voto*NumCrediti)/SUM(NumCrediti)  
FROM Esame E join Corso C on E.CodCorso = C.Codice  
GROUP BY E.MatrStud
```

```
SELECT Matricola  
FROM MediaStudente  
WHERE Media = ( SELECT MAX(Media)  
                FROM MediaStudente )
```

Oppure

Studenti con la media pesata più alta

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

Oppure

```
SELECT E.MatrStud,  
SUM(E.Voto*C.NumCrediti)/SUM(C.NumCrediti) As Media  
FROM Esame E join Corso C on E.CodCorso = C.Codice  
GROUP BY E.MatrStud  
HAVING SUM(E.Voto*C.NumCrediti)/SUM(C.NumCrediti)  
    >= ALL  
    ( SELECT SUM(E.Voto*C.NumCrediti)/SUM(C.NumCrediti)  
      FROM Esame E join Corso C on E.CodCorso = C.Codice  
      GROUP BY E.MatrStud )
```

Matricola e voto medio degli studenti che hanno un voto medio maggiore del voto medio complessivo

```
STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
               AnnoAccademico,NumeroStudenti)
```

```
CREATE VIEW MediaStudente (Matricola, Media) AS
SELECT E.MatrStud, SUM(E.Voto*NumCrediti)/SUM(NumCrediti)
FROM Esame E join Corso C on E.CodCorso = C.Codice
GROUP BY E.MatrStud
```

```
SELECT Matricola
FROM MediaStudente
WHERE Media > (SELECT AVG(Media)
FROM MediaStudente)
```

Per ogni studente, l'anno di corso in cui ha avuto la media più alta

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

CREATE VIEW MediaStudentePerAnno (Matricola, Anno, Media) **AS**
SELECT E.MatrStud, C.AnnoDiCorso, **SUM**(E.Voto*NumCrediti)/**SUM**(NumCrediti)
FROM Esame E, Corso C
WHERE E.CodCorso = C.Codice
GROUP BY E.MatricolaStudnete, C.AnnoDiCorso

SELECT MS1.Matricola, MS1.Anno
FROM MediaStudentePerAnno MS1
WHERE MS1.Media = (**SELECT MAX**(MS2.Media)
FROM MediaStudentePerAnno MS2
WHERE MS2.Matricola = MS1.Matricola)

Oppure, senza definire la view:

Per ogni studente, l'anno di corso in cui ha avuto la media più alta

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

Oppure, senza definire la view:

```
SELECT E.MatrStud, C.AnnoDiCorso,  
SUM(E.Voto*C.NumCrediti)/SUM(C.NumCrediti)  
FROM Esame E join Corso C on E.CodCorso = C.Codice  
GROUP BY E.MatricolaStudente, C.AnnoDiCorso  
HAVING SUM(E.Voto*C.NumCrediti)/SUM(C.NumCrediti) >= ALL  
    ( SELECT SUM(E2.Voto*C2.NumCrediti)/SUM(C2.NumCrediti)  
      FROM Esame E2 join Corso C2 on E2.CodCorso = C2.Codice  
      WHERE E.MatricolaStudente = E2.MatricolaStudente  
      GROUP BY C2.AnnoDiCorso )
```

**Studenti più regolari, ovvero
quelli con la minima
differenza tra il voto migliore
e il voto peggiore**

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

```
CREATE VIEW StudenteMinMax (Matricola, VotoMigliore, VotoPeggior) AS  
SELECT MatrStud, MAX(Voto), MIN(Voto)  
FROM Esame  
GROUP BY MatrStud  
  
SELECT Matricola  
FROM StudenteMinMax  
WHERE (VotoMigliore-VotoPeggior)=( SELECT MIN(VotoMigliore-VotoPeggior)  
                                FROM StudenteMinMax )
```


**Corsi in cui almeno il 50%
degli studenti ha preso un
voto maggiore di 25**

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

```
SELECT E.CodCorso
FROM Esame E
WHERE E.Voto>25
GROUP BY E.CodCorso
HAVING COUNT(*) >= 0.5 * ( SELECT COUNT(*)
                           FROM Esame E1
                           WHERE E1.CodCorso = E.CodCorso )
```

Studenti che hanno preso lo stesso voto in più di due terzi degli esami sostenuti

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

```
SELECT E.MatrStud
FROM Esame E.
GROUP BY E.MatrStud, E.Voto
HAVING COUNT(*) >= 2/3 * ( SELECT COUNT(*)
                           FROM Esame E2
                           WHERE E2.Matricola = E.Matricola )
```

Trovare i top ten studenti in base alla media pesata, tra quelli che abbiano sostenuto almeno 10 esami

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

```
SELECT MS1.Matricola, MS1.Media
FROM MediaStudente MS1
WHERE MS1.Matricola IN ( SELECT MatrStud
                        FROM Esame
                        GROUP BY MatrStud
                        HAVING COUNT(*)>=10 )
AND 9 >= ( SELECT COUNT(*)
          FROM MediaStudente MS2
          WHERE MS2.Media > MS1.Media )
```


Corsi svolti da professori di Torino che non sono stati superati da nessuno studente di Torino

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

```
SELECT I.CodCorso
FROM Insegnamento I, Professore P
WHERE I.MatrProf = P.Matricola AND P.Città = 'Torino'
AND I.CodCorso NOT IN ( SELECT E.CodCorso
                           FROM Esame E, Studente S
                           WHERE E.MatrStud = S.Matricola
                           AND S.Città = 'Torino' )
```

Studenti che non hanno mai superato esami tenuti da docenti con il loro stesso cognome

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

```
SELECT S.Matricola
FROM Studente S, Esame E
WHERE S.Matricola = E.MatrStud
      AND S.Matricola NOT IN ( SELECT E1.MatrStud
                                FROM Esame E1, Insegnamento I, Professore P
                                WHERE E1.CodCorso = E.CodCorso
                                   AND E1.CodCorso = I.CodCorso
                                   AND I.MatrProf = P.Matricola
                                   AND P.Cognome = S.Cognome )
```

Studenti che hanno sostenuto al più 5 esami di corsi di una stessa facoltà

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

```
SELECT DISTINCT E.MatrStud
FROM Esame E, Corso C
WHERE E.CodCorso = C.Codice
GROUP BY E.MatrStud, C.Facoltà
HAVING COUNT (*) <= 5
```

Studenti che hanno sostenuto almeno due esami di corsi tenuti dallo stesso docente

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

```
SELECT E1.MatrStud
FROM Esame E1, Insegnamento I1, Esame E2, Insegnamento I2
WHERE E1.CodCorso = I1.CodCorso AND E2.CodCorso = I2.CodCorso
      AND E1.CodCorso <> E2.CodCorso AND I1.MatrProf = I2.MatrProf
```


**Studenti che non hanno mai
sostenuto due esami di corsi
tenuti dallo stesso docente**

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

SELECT Matricola

FROM Studente

WHERE Matricola **NOT IN**

(**SELECT** E1.MatrStud

FROM Esame E1, Insegnamento I1, Esame E2, Insegnamento I2

WHERE E1.CodCorso = I1.CodCorso **AND** E2.CodCorso = I2.CodCorso
AND E1.CodCorso <> E2.CodCorso **AND** I1.MatrProf = I2.MatrProf)

Trovare i codici e i nomi dei corsi con il minimo numero di crediti

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

```
SELECT Codice, Nome
FROM Corso
WHERE NumCrediti = ( SELECT MIN(NumCrediti)
                     FROM Corso )
```

Oppure:

```
SELECT Codice, Nome
FROM Corso
WHERE NumCrediti <= ALL ( SELECT NumCrediti
                          FROM Corso )
```

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

Oppure:

[illegible]

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

Oppure (usando il costruttore di tupla):

[illegible]

Facoltà che forniscono il maggior numero di corsi

STUDENTE(<u>Matricola</u> ,Nome,Cognome,Indirizzo,Città)
ESAME(<u>CodCorso</u> , <u>MatrStud</u> ,Voto,Lode)
CORSO(<u>Codice</u> ,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(<u>Matricola</u> ,Nome,Cognome,Città)
INSEGNAMENTO(<u>CodCorso</u> , <u>MatrProf</u> , <u>AnnoAccademico</u> ,NumeroStudenti)

```
SELECT Facoltà
FROM Corso
GROUP BY Facoltà
HAVING COUNT(*) >= ALL ( SELECT COUNT(*)
                           FROM Corso
                           GROUP BY Facoltà )
```

Professori che hanno tenuto il maggior numero di corsi

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

```
SELECT MatrProf
FROM Insegnamento
GROUP BY MatrProf
HAVING COUNT(*) >= ALL ( SELECT COUNT(*)
                           FROM Insegnamento
                           GROUP BY MatrProf )
```

Professori che hanno tenuto il maggior numero di corsi da almeno 5 crediti

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

```
SELECT I.MatrProf
FROM Insegnamento I, Corso C
WHERE I.CodCorso = C.Codice AND C.NumCrediti >= 5
GROUP BY I.MatrProf
HAVING COUNT(*) >= ALL ( SELECT COUNT(*)
                           FROM Insegnamento, Corso
                           WHERE CodCorso = Codice AND NumCrediti >= 5
                           GROUP BY MatrProf )
```

**Corsi in cui almeno uno
studente che ha superato
l'esame aveva lo stesso
cognome del docente**

STUDENTE(Matricola,Nome,Cognome,Indirizzo,Città)
ESAME(CodCorso,MatrStud,Voto,Lode)
CORSO(Codice,Nome,AnnoDiCorso,Facoltà,NumCrediti)
PROFESSORE(Matricola,Nome,Cognome,Città)
INSEGNAMENTO(CodCorso,MatrProf,
AnnoAccademico,NumeroStudenti)

SELECT E.CodCorso
FROM Esame E, Insegnamento I, Professore P, Studente S
WHERE E.CodCorso = I.CodCorso **AND** E.MatrStud = S.Matricola
AND I.MatrProf = P.Matricola **AND** S.Cognome = P.Cognome

“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 DISTINCT 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'))
```

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

```
SELECT P1.CFMadre,P1.CFPadre  
FROM PERSONA P1 JOIN MATRIMONIO M  
ON P1.CodFis=M.CFMoglie OR P1.CodFis=M.CFMarito  
WHERE Data>='1/1/2010' AND Data<='31/12/2010'
```

Estrarre i dati di genitori di cui un figlio non si è mai sposato nel 2010

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

```
SELECT DISTINCT P1.*  
FROM PERSONA P1, PERSONA P2  
WHERE (P1.CodFis =P2.CFMadre OR P1.CodFis =P2.CFPadre) AND  
      (P2.CodFis NOT IN (SELECT CFMoglie  
                        FROM MATRIMONIO  
                        WHERE Data>='1/1/2010' AND  
                        Data<='31/12/2010') AND  
      P2.CodFis NOT IN (SELECT CFMarito  
                        FROM MATRIMONIO  
                        WHERE Data>='1/1/2010' AND  
                        Data<='31/12/2010'))
```

Estrarre i dati di persone di cui
nessun figlio si è sposato nel 2010

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

```
SELECT *  
FROM PERSONA PG  
WHERE PG.CodFis NOT IN  
    SELECT P1.CodFis  
    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') AND  
    P2.CodFis IN (SELECT CFMarito  
                  FROM MATRIMONIO  
                  WHERE Data>='1/1/2010' AND Data<='31/12/2010'))
```

Estrarre i dati di genitori di cui
nessun figlio si è sposato nel 2010

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

```
SELECT DISTINCT PG.*  
FROM PERSONA PG, PERSONA PF  
WHERE (PG.CodFis =PF.CFMadre OR PG.CodFis =PF.CFPadre)  
AND PG.CodFis NOT IN  
    SELECT P1.CodFis  
    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.Data < M.Data)  
AND CFMarito IN (SELECT CFMarito  
                 FROM Matrimonio M2  
                 WHERE 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 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 (P1.CFMadre,P1.CFPadre) IN
      (SELECT CFMoglie, CFMarito
       FROM Matrimonio M1)
AND (P2.CFMadre,P2.CFPadre) IN
      (SELECT CFMoglie, CFMarito
       FROM Matrimonio M1)
```

Estrarre le persone sposate, figlie
di persone che non sono mai state
sposate [tra loro]

PERSONA(<u>CodFis</u> ,Nome,DataNascita, CFMadre,CFPadre)
MATRIMONIO(<u>Codice</u> ,CFMoglie,CFMarito, Data,NumeroInvitati)
TESTIMONI(<u>CodiceMatr</u> ,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 le persone sposate, figlie
di persone che non sono mai state
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 (P.CFMadre, P.CFPadre) NOT IN  
    (SELECT M1.CFMoglie,M1.CFMarito  
     FROM Matrimonio M1 )
```

Estrarre i matrimoni registrati il primo giorno in cui è stato registrato un qualche matrimonio

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

```
SELECT *  
FROM MATRIMONIO  
WHERE Data = (SELECT min(Data) FROM Matrimonio)
```

```
SELECT *  
FROM MATRIMONIO  
WHERE Data<= ALL (SELECT Data FROM Matrimonio)
```

Estrarre il matrimonio con più invitati

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

```
SELECT *  
FROM MATRIMONIO  
WHERE NumeroInvitati =  
      (SELECT max(NumeroInvitati) FROM Matrimonio)
```

```
SELECT *  
FROM MATRIMONIO  
WHERE NumeroInvitati >= ALL  
      (SELECT NumeroInvitati FROM Matrimonio)
```

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

PERSONA(<u>CodFis</u> ,Nome,DataNascita, CFMadre,CFPadre)
MATRIMONIO(<u>Codice</u> ,CFMoglie,CFMarito, Data,NumeroInvitati)
TESTIMONI(<u>CodiceMatr</u> ,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.CodiceMatr  
                           AND T.CFTestimone=P.CodFis  
                           AND M2.Data>M1.Data  
                           ) ) > 0
```

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, Matrimonio M1, Matrimonio M2,TESTIMONI T  
WHERE M1.CFMarito=P.CodFis AND M1.CFMoglie=M2.CFMoglie AND  
M2.Codice=T.CodiceMatr AND T.CFTestimone=P.CodFis AND  
M2.Data>M1.Data
```

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
```

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

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

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, CodiceHotel, Data, Costo) AS  
SELECT CodiceCliente, CodiceHotel, Data, 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=C1.CodiceHotel  
      AND P.Data=C1.Data 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

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

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

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

```
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.

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

```
select Nome
from Giocatore G, JOIN Partecipazione
where IDGiocatore = G.ID
group by G.ID, Nome
having count(*)>=3 OR count(distinct Nazione)>=2
```

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.

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

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 CodiceFiscale 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 CodiceFiscale 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
)
```

Dato il seguente schema relazionale:

AGENTE(Nome, Percentuale)

ARTICOLO(Nome, Descrizione, Tipo)

CLIENTE(Nome, Indirizzo, Telefonoi)

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

AGENTE(<u>Nome</u> , Percentuale)
ARTICOLO(<u>Nome</u> , Descrizione, Tipo)
CLIENTE(<u>Nome</u> , Indirizzo, Telefonoi)
VENDITA(<u>Nome-Comp</u> , <u>Nome-Art</u> , <u>Nome-Ag</u> , <u>Data</u> , 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 >= 1/1/93 and V.Data <= 31/12/93
      AND Ar.Tipo='automobile'
```

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

AGENTE(Nome, Percentuale)

ARTICOLO(Nome, Descrizione, Tipo)

CLIENTE(Nome, Indirizzo, Telefonoi)

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

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 DISTINCT 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' )
```

AGENTE(Nome, Percentuale)

ARTICOLO(Nome, Descrizione, Tipo)

CLIENTE(Nome, Indirizzo, Telefonoi)

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

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

```
CREATE VIEW ImplImm (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 ImplImm 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)

```
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')
```

AUTORE(NOME, COGNOME, Data-N, Nazionalita)
AUTORELIBRO(NOME, COGNOME, SEGNATURA)
LIBRO(SEGNATURA, Scaffale, Argomento, Lingua)

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 DISTINCT A.Nome, 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 Argomento='MATEMATICA')
```

```
AUTORE(NOME, COGNOME, Data-N, Nazionalita)  
AUTORELIBRO(NOME, COGNOME, SEGNATURA)  
LIBRO(SEGNATURA, Scaffale, Argomento, Lingua)
```

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
```

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.

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.

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 l'ammontare complessivo delle bollette del 2006 supera di oltre il 20% l'ammontare delle proprie bollette nell'anno 2005.

```
SELECT *  
FROM CLIENTE C  
WHERE (  
    SELECT SUM(cifra)  
    FROM BOLLETTA B1  
    WHERE anno = 2006 AND C.codfiscale = B1.codfiscale) > 1,20 *  
    (  
        SELECT SUM(cifra)  
        FROM BOLLETTA B2  
        WHERE C.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)
```

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

**Versione che usa
solo il NOT IN**

```
WHERE CodiceForn NOT IN
```

```
( SELECT CodiceForn
```

```
FROM Prodotti, Fornitori
```

**Prodotto
Cartesiano**



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

```
( SELECT CodiceProd, CodiceForn
```

```
FROM Catalogo C ) )
```


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, Telefonoi)

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, CITTA)

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, CITTA)

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) >= ALL (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) >= ALL (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)
TRASFERTA (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, PUNTI NECESSARI)

- 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, CITTA, 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, CITTA, 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 (NAME, ISBNLIBRO)
LIBRO (ISBN, TITOLO, EDITORE, GENERE, ANNODIPUBBLICAZIONE)
CLIENTE (CODICEFISCALE, NOME, CITTA, 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 Urs2 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.Citta = C.Citta 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, Citta, 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, CittaNascita)

ESAME (Matr, CodCorso, Data, Voto)

CORSO (Codice, Nome, CFU, MatrDocente)

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

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, CittaNascita)

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, Citta, DataNascita)

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

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

TESI (Matr, Titolo, CodCorsoCollegato, DataInizio, Conclusa)

- 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
- Estrarre in SQL la matricola degli studenti già laureati che hanno iniziato la tesi solo dopo la registrazione del loro ultimo esame

1.

```
select Nome, Cognome
from Studente S, Esame E, Corso C, Tesi T
where S.Matr = E.Matr and E.CodCorso = C.CodCorso and
      C.CodCorso = CodCorsoCollegato and T.Matr = S.Matr and
      Voto = 18 and Anno = 1
```

2.

[illegible]

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)
```

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, CittaNascita)

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 VARI)

- La seguente base di dati è relativa alla registrazione degli esami in una università:
STUDENTE (Matricola, Nome, Cognome, DataNascita, CittaNascita)
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.
- Estrarre in SQL i nomi dei docenti che hanno registrato almeno una volta un voto ad uno studente più anziano di loro.
- Estrarre in SQL il docente più “stakanovista” del 2012, cioè quello che nel 2012 ha registrato il maggior numero di voti.
- Estrarre in SQL le matricole degli studenti che hanno preso un 18 dopo aver preso un 30.
- Estrarre in SQL il dipartimento i cui docenti sono stati complessivamente più “severi” nel 2012, cioè hanno per il 2012 la più bassa media degli esami registrati tra tutti i dipartimenti dell’ateneo.

Esercizio (tdeB 20-09-2012 e 9-9-2013)

- 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, CognomeDocente)
- Estrarre in SQL gli studenti che hanno preso un 30 da un docente con il loro stesso cognome
- Estrarre in SQL i nomi degli studenti che hanno sostenuto almeno 4 esami e hanno preso sempre lo stesso voto
- Estrarre in SQL la “classifica” di tutti gli studenti (*la matricola è sufficiente*) ordinati in base alla media (decrescente) dei soli esami del primo anno
- Estrarre in SQL Nome, Cognome e Matricola degli studenti che hanno preso sempre (e solo) voti compresi tra 23 e 27

Esercizio (tdeB 06-02-2012)

- Il seguente database permette di archiviare su PC in forma relazionale una collezione di messaggi e una rubrica di contatti per un dispositivo che funziona come quello specificato nel primo esercizio:

MESSAGGIO (Id, NumeroMittente, Timestamp, Testo, IdMessaggioAlQualeRisponde)

DESTINATARI (IdMessaggio, NumeroDestinatario)

CONTATTORUBRICA (Numero, Nome, Email, Foto)

- Estrarre in SQL il testo dei messaggi inviati nel 2011 dall'utente registrato in rubrica come "Smilzo" a più di 5 destinatari.
- Esprimere in SQL l'interrogazione che visualizza in ordine cronologico tutti i messaggi inviati e ricevuti dal numero 3216549870.
- Esprimere in SQL l'interrogazione che visualizza in ordine cronologico tutti i messaggi inviati e ricevuti dal contatto memorizzato in rubrica come "Zorro".

Esercizio (tdeB 06-07-2012)

- Il seguente schema descrive i ruoli degli attori nei film e gli eventuali riconoscimenti ottenuti.
 - FILM (Titolo, DataRilascio, Regista, Lingua, Minuti, Incasso)
 - ATTORE (Nome, DataNascita, Nazionalità, Sesso)
 - PERSONAGGIO (TitoloFilm, NomeAttore, NomePersonaggio)
 - PREMIO (NomeAttore, TitoloFilm, TipoPremio, DataAssegnazione)
- Estrarre in SQL i nomi delle attrici di nazionalità canadese o neozelandese che ad oggi non hanno ancora compiuto 30 anni, ma avevano già vinto un Oscar prima dell'uscita di *Titanic*. [**1,5** p]
- Estrarre in SQL i nomi delle attrici che hanno interpretato lo stesso personaggio in tre film diversi.

Esercizio (tde 7-2-2012)

- Sia dato il seguente database relazionale, relativo a un'agenzia di collocamento:
OFFERTA (Codice, Descrizione, TitoloDiStudioRichiesto, Stipendio, AnniDiEsperienzaRichiesti)
CURRICULUM (CodiceFiscale, Nome, Cognome, AnnoDiNascita, TitoloDiStudio, StipendioAttuale)
COLLOQUI (CodiceOfferta, CodiceFiscaleCandidato, Data, Ora)
- Si scriva in SQL l'interrogazione che estrae i colloqui fissati a persone che guadagnano di più di quanto sarebbe loro offerto nel colloquio
- Si scriva in SQL l'interrogazione che estrae le offerte più convenienti per ognuno dei curriculum ricevuti, cioè le offerte che a parità di titolo di studio offrono lo stipendio più alto

Esercizio (tde 23-2-2012)

- Il seguente schema descrive la base di dati usata per gestire le borse di studio di una università.

DOMANDABORSA (Matricola, DataDomanda, Stato)

CORSO (CodCorso, NomeCorso, NumeroCrediti)

GRADUATORIA (Matricola, Media, Cfu, Posizione)

ESAME (CodCorso, Matricola, Data, Voto)

- Estrarre gli studenti che hanno fatto domanda per una borsa di studio pur avendo una media inferiore a 27.
- Estrarre lo studente più in alto nella graduatoria che ha superato l'esame del corso "Informatica A".

Esercizio (tde 6-7-2012, tde 10-9-2012)

- Il seguente schema descrive la base di dati usata per gestire delle gare automobilistiche su un unico circuito.

CONCORRENTE (Codice, Nome, Cognome)

GRADUATORIA (CodiceConcorrente, MediaTempi)

GARA (Codice, Nome, NumConcorrenti)

ESITOGARA (CodiceGara, CodiceConcorrente, Tempo, Posizione)

- La tabella Graduatoria contiene le medie delle gare per ogni concorrente
- Estrarre il nome e il cognome del concorrente con la migliore media dei tempi
- Estrarre il nome e il cognome dei concorrenti che non sono mai arrivati primi in una gara
- Estrarre il nome e il cognome dei concorrenti con la media dei tempi inferiore alla media delle medie
- Estrarre il nome e il cognome dei concorrenti che sono sempre arrivati ultimi nelle gare a cui hanno partecipato

Esercizio (tde 5-2-2013)

- Si considerino le tabelle:

CATALOGO(ID, Titolo, NumPag, Formato, Data, Durata)

CONTIENE(IDCAT,IDPROD, Foto, Descrizione, Sconto)

PRODOTTO(ID, Nome, Categoria, CostoListino)

- Il catalogo descrive delle offerte di vendita con sconti sui prodotti. Lo sconto è espresso in percentuale.
- Esprimere in SQL l'interrogazione che estrae il catalogo più recente che offre uno sconto di almeno il 50% relativo al prodotto "Photoshop"
- Esprimere in SQL l'interrogazione che estrae i cataloghi che non contengono prodotti il cui costo scontato è inferiore a 1000 Euro

Esercizio (tde 27-2-2013)

- Si consideri il seguente schema di base di dati che vuole tenere traccia di alcune attività di una catena di supermercati.

SUPERMERCATO (CodiceSM, Nome, Indirizzo, Città, Tel, Responsabile)

PRODOTTO (CodiceProdotto, Nome, Produttore, Prezzo)

CLIENTE (CF, NumTessera, Nome, Cognome, Indirizzo, Città, Tel)

RIGASCONTRINO (NumTessera, CodiceSM, Data, Ora, CodiceProdotto, Quantità)

- Scrivere in SQL l'interrogazione che trova nome e cognome dei clienti che non hanno mai fatto la spesa due volte nello stesso giorno.
- Scrivere in SQL l'interrogazione che trova codice e nome dei prodotti che sono stati venduti il numero maggiore di volte.