Studenti,Esami,Insegnamenti

ii. Trovare matricola e nome degli studenti che hanno fallito la prova d'esame (Voto < 18) nel minimo numero di insegnamenti diversi

```
create view StudInsFalliti as
 select s.matricola,s.Nome,count(distinct e.codIns) as num
 from Studenti s natural join Esami e
 where e.voto < 18
 group by s.matricola,s.nome
union
 select s.matricola,s.nome,0 as num
 from Studenti s
 where s.matricola not in
  (select e.matricola from Esami e where e.voto <18)
```

```
select s.matricola,s.nome
from StudInsFalliti s
where s.num = (select min(num) from StudInsFalliti)

```

iii.Trovare gli studenti (matricola e nome) che hanno sostenuto solo prove relative ad un unico insegnamento,fornendo anche il titolo dell'insegnamento.

```
select distinct s.matricola,s.nome,i.titolo
from Studenti s natural join Esami e natural join Insegnamenti i
where not exists
 (select * from Esami e1 where (e1.matricola = e.matricola) and (e1.codins <> e.codins))

```

iv. Fornire l'insegnamento (o gli insegnamenti) dove il maggior numero di studenti hanno fallito almeno una prova.

```
create view Fallimenti(codins,titolo,num) as
 select i.codins,i.titolo,count(distinct e.matricola) as num
 from Esami e natural join Insegnamenti i
 where e.voto < 18
 group by i.codins,i.titolo
```

v. Cancellare gli studenti che hanno fallito almeno tre prove di uno stesso insegnamento (non preuccoparsi dei vincoli di integrità referenziale)

```
delete from Studenti where matricola in
 (select e.matricola
  from Esami e
  where e.voto < 18
  group by e.matricola,e.codins
  having count(*) >= 3)
```

Squadre,Partite,Reti,Gicatori

i. Trovare per ogni giocatore,l'identificatore,il nome,la nazione ed il numero delle reti realizzate,esclusa le autoreti

IdG<Perogni> Nome,Nazione,COUNT(*) as NumeroReti (<Restrizione Auto!=True> (Reti natural join Giocatori))

ii. Trovare id e nome del capocannoniere (giocatore che hanno realizzato il massimo numero di reti escluse le autoreti

```
select p.IdP
from Partite p
where p.IdP not in
 (select p1.IdP
 from Partite p1 natural join Reti r)
```

iii. Trovare la squadra che ha subito meno reti (si può assumere che ogni squadra abbia subito almeno una rete.

```
create VIEW View1 as
select g.Nazione,COUNT(*) as NumeroReti
from Reti r natural join Giocatori g
where r.Auto = "false"
group by g.Nazione;

select v.Nazione
from View1 v
where v.NumeroReti > (select AVG(NumeroReti) from View1);
```

iv. Cancellare le squadre che hanno almeno 11 giocatori (senza preoccuparsi di mantenere l'integrità referenziale)

```
update Squadre s set s.Allenatore =
 (select MIN(g.IdG)
 from Giocatori g
 where s.Nazione = g.Nazione)
where s.Allenatore is null;
```

Squadre,Partite,Reti,Gicatori (fatte con Ricky,non sono sicuro)

i. Trovare per ogni giocatore,l'identificatore,il nome,la nazione ed il numero delle reti realizzate, escludedo le autoreti.

```
select g.IdG,g.Nome,g.Nazione,count(r.IdG)
from Giocatori g natural join Reti r
where r.auto=0
group by r.IdG
```

ii. Ritornare le partite che si sono concluse con risultato 0-0

```
select p.Squadra1,p.Squadra2
from Partite p natural join Reti r
where r.IdP NOT IN
 (select r.IdP from Reti r where r.IdG!=0)
```

iii. Trovare le squadre che hanno fatto un numero di reti superiore alla media, escludendo nel calcolo delle reti fatte da una squadra le autoreti,sia delle squadre avversarie che proprie (si può assumere che almeno una squadra abbia una rete)

```
Creo una vista in cui conto quante reti ha fatto ciascuna squadra

create view Reti1(Nome,Reti) as
 select g.Nazione,count(r.IdG)
 from Reti r natural join Giocatori g
 where r.Auto=0
 group by g.Nazione

Creo una vista con la media delle reti
create view Media1(media) as
 select avg(Reti.reti)
 from Reti1
 group by Reti1.Nazione

 select Nazione from Reti1 where Reti1.reti > (select Media1.media from Media1)
```

iv. Per ogni squadra che non ha un allenatore (allenatore è null) inserire come allenatore il giocatore della squadra con IdG minimo.

```
update Squadre set allenatore =
 (select g.Nome from Giocatori g where g.IdG =
  (select min(g.IdG from giocatori g where g.Nazione =
   (select s.Nazione from Squadre s where s.allenatore is NULL)))
   where allenatore is NULL)
```

Corsi,Orario,Aule

i. Trovare i corsi che sono schedulati in qualche aula con numero di posti inferiore agli iscritti

pigreco(IDCorso, Nome)(sigma(Posti<Iscritti)
(Corsi(SIMBOLOJOIN)Orario(SIMBOLOJOIN)Aule))

```sql
SELECT DISTINCT c.IdCorso, c.Nome
FROM Corsi c NATURAL JOIN Orario o NATURAL JOIN Aule a
WHERE a.Posti < c.Iscritti
```

ii. Trovare i corsi che hanno il massimo numero di lezioni in un singolo giorno

```sql
CREATE VIEW LezAlGiorno AS
 SELECT o.IdCorso, o.Giorno, COUNT(o.IdCorso) AS Numero
 FROM Orario o
 GROUP BY o.Giorno, o.IdCorso;

SELECT l.IdCorso
FROM LezAlGiorno l
WHERE l.Numero =
 (SELECT MAX(l1.Numero)
 FROM LezAlGiorno l1);
```

iii. Tra i corsi con numero di iscritti minimo, individuare quelli per i quali la media dle numero di posti nelle aule nelle quali si svolgono è massima.

```sql
CREATE VIEW MinimoIscritti AS
 SELECT c.IdCorso
 FROM Corsi c
 WHERE Iscritti =
  (SELECT MIN(c1.Iscritti)
   FROM Corsi c1);

SELECT mi.IdCorso
FROM Aule a NATURAL JOIN Orario o NATURAL JOIN MinimoIscritti mi
GROUP BY mi.IdCorso
HAVING AVG(a.Posti) >= ALL
 (SELECT AVG(a1.Posti)
  FROM Aule a1 NATURAL JOIN Orario o1 NATURAL JOIN MinimoIscritti mi1
  GROUP BY mi1.IdCorso)
```

iv. Eliminare i corsi con meno di 300 iscritti e le relative entry nell'orario

```sql
DELETE FROM Orario WHERE IdCorso =
 (SELECT c.IdCorso
 FROM Corsi c
 WHERE c.Iscritti < 300);

DELETE FROM Corsi WHERE IdCorso NOT IN
 (SELECT o.IdCorso
 FROM Orario o);
```

Fiumi,Monti,Mari

i. Trovare il nome del fiume (o dei fiumi) con lunghezza minima che sfocia nell'adriatico

```
select f1.nome from Fiumi f1 where f1.lunghezza =
 (select min(lunghezza) from fiumi) and foce='adriatico')
```

ii. Trovare i nomi dei mari nei quali non sfociano fiumi

```
select ma.nome from Mari ma where not in
 (select distinct f.foce from Fiumi f)
```

iii. Per ogni coppia (montagna,mare) dare il nome della montagna,quello del mare ed il numero di fiumi che nascono da quella montagna e sfociano in quel mare (anche quando questo numero sia zero)

```
create view MontiMariFiumi as
 (select ma.mari,ma.nome,count(*) as numfiumi
  from Monti mo natural join Fiumi fi natural join mari ma
  order by mo.nome,ma.nome )
```

iv. Assumendo che esista un unico monte di altezza massima, aumentare di 4m la sua altezza e conseguentemente incrementare di 6m la lunghezza di tutti i fiumi che nascono da esso.

```
update Monti mo
set new.altezza = old.altezza+4
where mo.altezza = (select max(altezza) from Monti)

update Fiumi set new.lunghezza = old.altezza+6 where sorgente =
 (select nome from Monti where altezza = (select max(altezza) from Monti))
```

Città,Artisti,Eventi

i. L'elenco delle città (nome e provincia) che hanno ospitato eventi

```
SELECT DISTINCT c.Nome, c.Provincia
FROM Citta c
NATURAL JOIN Eventi e
```

ii. La provincia nella quel la media degli spettatori per concerto è stata massima

```
select provincia
from Citta c natural join Eventi e
group by provincia
having avg(NumSpettatori) >= all (
 select avg(NumSpettatori)
 from Citta cc natural join Eventi ee
 where ee.provincia <> c.provincia
 group by cc.provincia)
```

iii. Gli artisti che hanno tenuto spettacolo in tutte le provincie elencate nel database

```
select a.nome from Artisti a where a.idartista IN
 (select p.idartista
 from pronvicie_diverse p
 where conta =
  (SELECT COUNT( c.idCitta )
  FROM Citta c))
```

iv. Eliminare gli artisti che non abbiano avuto,complessivamente, in tutti gli eventi, 1000 spettatori.

```
delete from Artisti where idartista in
 (select a.idartisti
 from Artisti a natural join Eventi e
 group by e.idartista
 having sum/e.numspettatori < 1000)
```

Musei,Opere,Artisti

i.Trovare, per ogni artista, il nome dell'artista ed il numero delle opere conservate nei musei della sua nazione,

```
select a.note,count(o.codice)
from Musei m inner join (Opere o inner join Artisti a on a.NomeA = o.NomeA) on o.NomeM = m.NomeM)
where a.Nazione = m.Nazione
group by a.Nome
```

oppure

```
select a.NomeA,count(o.Codice)
from Musei m natural join Opere o natural join Artisti a)
where a.Nazione = m.Nazione
group by a.NomeA
```

ii. Trovare il nome degli artisti con il numero massimo di opere esposte in musei della propria nazione

```
select max(tabella.conta),prova.NomeA
from
 (select count(*) as conta,o.Nome,a.Nazione
 from Opere o
 natural join Musei m
 natural join Artisti a
 where a.Nazione = m.Nazione
 group by a.NomeA) as prova group by Nazione)
```

iii. Trovare gli artisti, Nome e Nazione,che non hanno opere esposte in musei della propria nazione.

```
select a.Nome,a.Nazione
from Artisti a where a.NomeA not in
 (select o.NomeA
  from Opere o natural join Musei m natural join Artisti a
  where a.Nazione = m.Nazione)
```

iv. Verificare se ci sono musei che contengono solo opere di artisti nazionali,fornendo il nome dei tali musei.

```
select m.NomeM
from Musei m
natural join Opere o
where not
exists (
 select *
 from Artisti a
 where m.Nazione <> a.Nazione
 and o.NomeA = a.NomeA
)
```

Pizze,Ricette,Ingredienti,Ordini

i. Trovare il nome delle pizze ordinate dal cliente gino.

<Proiezione> Nome As NomePizza(Restrizione>nomeCliente="Gino"(Ordini natural join Pizze)

```
SELECT  p.nome AS NomePizza
FROM Ordini o NATURAL JOIN Pizze p
WHERE o.nomeCliente = "Gino"
```

ii. Per ogni cliente trovare il nome e il prezzo delle/a pizza/e più care/e che ha ordinato

```
SELECT o.NomeCliente, p.pizza,p.prezzo
FROM Ordini o NATURAL JOIN Pizze p
GROUP BY o.NomeCliente
HAVING MAX(p.prezzo)
```

iii. Trovare il codice e il tempo di preparazione delle pizze che contengono la mozzarella (nome ingrediente = 'mozzarella') e hanno più di 3 ingredienti

```
CREATE VIEW View1 AS
SELECT r.CodPizza
FROM Ricette r NATURAL JOIN Ingredienti i
WHERE i.nome = "mozzarella"
GROUP BY r.CodPizza
HAVING COUNT(*) >3

SELECT v.CodPizza,p.nome,p.tempoPrep
FROM View1 v NATURAL JOIN Pizze p
```

iv. Trovare il codice, il costo e il prezzo delle pizze il cui costo (ottenuto sommando il costo base degli ingredienti tenendo conto delle rispettive quantita` necessarie) `e piu` alto del prezzo della pizza;

```
CREATE VIEW View2 AS
SELECT r.CodPizza,SUM(i.costoBase * r.quantitaNecessaria) AS CostoReale
FROM Ricette r NATURAL JOIN Ingredienti i
GROUP BY r.CodPizza

SELECT v.CodPizza,v.CostoReale,p.prezzo
FROM View2 NATURAL JOIN Pizze p
WHERE v.CostoReale > p.prezzo
```

v. Dimezzare la quantità di cipolla in tutte le pizze che ce l'hanno come ingrediente (nome ingrediente=‘cipolla’).

```
UPGRADE Ricette r
SET r.quantitaNecessaria = r.quantitaNecessaria/2
WHERE r.CodIngrediente IN ( SELECT i.codIgrediente FROM Ingrediente i WHERE i.nome="cipolla")
```