

Database Course

Ahmad Yoosofan

SQL

University of Kashan

create table

```
create table s (
    sn      char(10) primary key,
    sname   char(30),
    status  int default 0,
    city    char(20)
);
```

DSL(Data Sub Language)

SQL (Structured Query Language)

- DDL: Data Definition Language
- DML: Data Manipulation Language
- DCL: Data Control Language

DDL: create table

SQLite

Online

1. <https://sql.js.org/examples/GUI/>
2. <https://sql.js.org/#/>
3. <https://www.sqlitetutorial.net/>
4. <https://sqliteonline.com/>
5. <https://extendsclass.com/sqlite-browser.html#>
6. <https://inloop.github.io/sqlite-viewer/>
7. <https://github.com/inloop/sqlite-viewer>
8. <https://github.com/sql-js/sql.js>
9. <https://sql.js.org/#/>
10. <http://sqlfiddle.com/>
11. <https://github.com/coleifer/sqlite-web>
12. <https://sqlitestudio.pl/>
13. <https://www.onworks.net/programs/sqlite-online?amp=0>
14. <https://www.heidisql.com/#featurelist>
15. <https://sqlzoo.net/>

Terminal and GUI

1. <https://www.sqlite.org/download.html>
2. <https://sqlite.org/src/timeline>
3. <https://github.com/sqlite/sqlite>
4. <https://www.sqlite.org/cli.html>
5. <https://sqlitebrowser.org/>
6. <https://github.com/sqlitebrowser/sqlitebrowser>

```
create table s (
    sn      char(10) primary key,
    sname   char(30),
    status  int default(0),
    city    char(20)
);

create table p (
    pn      char(10) primary key,
    pname   char(30),
    color   char(20),
    weight  NUMERIC(9, 2),
    city    char(20)
);

create table sp (
    sn      char(10) references s,
    pn      char(10) references p,
    qty    int default(0),
    primary key (sn, pn)
);
```

Database

SP database includes the following

- s, p, sp

Database Schema

```
create database sp;
```

DBMS(Database Management System)

- An application
- RDBMS
- DB2, Oracle, PostgreSQL, MySQL, SqlServer, MariaDB
- SQLite (Lack of DCL commands, each db on a file)

insert

DML

```
insert into s(sn, sname, status, city)
  values('s1', 'Smith', 20, 'London')
;
insert into s(sn, sname, status, city)
  values('s2', 'Jones', 10, 'Paris' )
;
insert into s(sn, sname, status, city)
  values('s3', 'Blake', 30, 'Paris' )
;
insert into s(sn, sname, "status", city)
  values('s4', 'Clark', 20, 'London' )
;
insert into s(sname, status, city, sn)
  values('Adams', 30, 'Athens', 's5')
;
insert into s
  values('s6', 'Ali', 40, 'کاشان')
;
```

P

```
insert into p(pn, pname, color, weight, city)
  values('p1','Nut' , 'Red' , 12.0,'London')
;
insert into p(pn, pname, color, weight, city)
  values
    ('p2', 'Bolt' , 'Green', 17.0, 'Paris' ),
    ('p3', 'Screw', 'Blue' , 17.0, 'Oslo' ),
    ('p4', 'Screw', 'Red' , 14.0, 'London'),
    ('p5', 'Cam' , 'Blue' , 12.0, 'Paris' ),
    ('p6', 'Cog' , 'Red' , 19.0, 'London')
;

insert into p(pn, pname, color, city)
  values('p7', 'Nut', 'Red', 'London')
;
insert into p(pn, pname, color, city)
  values('p8', 'Bolt', 'Green', 'Paris')
;
```

SP

```
insert into sp(sn, pn, qty)
values
('s1', 'p1', 300),
('s1', 'p2', 200),
('s1', 'p3', 400),
('s1', 'p4', 200),
('s1', 'p5', 100),
('s1', 'p6', 100),
('s2', 'p1', 300),
('s2', 'p2', 400),
('s3', 'p2', 200),
('s4', 'p2', 200),
('s4', 'p4', 300),
('s4', 'p5', 400),
('s6', 'p2', 350)
```

;

نام قطعه‌ها را بیابید.

```
select pname  
from p  
;
```

```
p{pname};
```

pname
Nut
Bolt
Screw
Screw
Cam
Cog
Nut
Bolt

نام قطعه‌ها و وزن آنها را بیابید.

```
select pname, weight  
from p  
;
```

```
p{pname, weight} ;
```

pname	weight
Nut	12
Bolt	17
Screw	17
Screw	14
Cam	12
Cog	19
Nut	
Bolt	

نام قطعه‌ها و وزن آنها را به گرم بیابید.

```
select pname, weight * 1000
from p
;
```

pname	weight * 1000
Nut	12000
Bolt	17000
Screw	17000
Screw	14000
Cam	12000
Cog	19000
Nut	
Bolt	

as (rename)

```
select pname, weight * 1000 as gweight  
from p  
;
```

pname	gweight
Nut	12000
Bolt	17000
Screw	17000
Screw	14000
Cam	12000
Cog	19000
Nut	
Bolt	

نام عرضه‌کنندگان شهر کاشان را بیابید.

```
select sname  
from s  
where city = 'کاشان'  
;
```

```
-- (s where city = 'کاشان') {pname}
```

```
select sname  
from s  
where city = 'Paris'  
;
```

sname
Jones
Blake

شماره قطعه‌های عرضه شده را بیابید.

```
select pn  
from sp  
;
```

pn
p1
p2
p3
p4
p5
p6
p1
p2
p2
p2
p4
p5
p2

نام قطعه‌های عرضه شده را بیابید.

```
select pname  
from p, sp  
where p.bn = sp.bn  
;
```

```
(  
  (  
    (  
      p rename bn as ppn  
    )  
    times sp  
  ) where ppn = bn  
) {pname}
```

pname
Nut
Bolt
Screw
Screw
Cam
Cog
Nut
Bolt
Bolt
Bolt
Screw
Cam
Bolt

join

نام قطعه‌های عرضه شده را بیابید.

```
select pname  
from p natural join sp  
;
```

(p **join** sp) {pname}

```
select pname  
from p join sp using(pn)  
;
```

```
select pname  
from p join sp on p.pn=sp.pn  
;
```

نام قطعه‌هایی را بباید که در شهر آن قطعه‌ها عرضه کننده‌ای وجود داشته باشد

```
select pname  
from p join s using(city)  
;
```

```
select pname  
from p natural join s  
;
```

pname
Nut
Nut
Bolt
Bolt
Screw
Screw
Cam
Cam
Cog
Cog
Nut
Nut
Bolt
Bolt

اطلاعات عرضه‌کنندگان را بیابید

```
select *  
from s  
;
```

sn	sname	status	city
s1	Smith	20	London
s2	Jones	10	Paris
s3	Blake	30	Paris
s4	Clark	20	London
s5	Adams	30	Athens
s6	Ali	40	کاشان

اطلاعات عرضه‌کنندگان و قطعه‌هایی را که عرضه کرده‌اند، بیابید.

```
select *
from (p join sp using(pn))
      join s using(sn)
;
```

pn	pname	color	weight	city	sn	qty	sname	status	city
p1	Nut	Red	12	London	s1	300	Smith	20	London
p2	Bolt	Green	17	Paris	s1	200	Smith	20	London
p3	Screw	Blue	17	Oslo	s1	400	Smith	20	London
p4	Screw	Red	14	London	s1	200	Smith	20	London
p5	Cam	Blue	12	Paris	s1	100	Smith	20	London
p6	Cog	Red	19	London	s1	100	Smith	20	London
p1	Nut	Red	12	London	s2	300	Jones	10	Paris
p2	Bolt	Green	17	Paris	s2	400	Jones	10	Paris
p2	Bolt	Green	17	Paris	s3	200	Blake	30	Paris
p2	Bolt	Green	17	Paris	s4	200	Clark	20	London
p4	Screw	Red	14	London	s4	300	Clark	20	London
p5	Cam	Blue	12	Paris	s4	400	Clark	20	London
p2	Bolt	Green	17	Paris	s6	350	Ali	40	کاشان

نام قطعاتی را بباید که عرضه‌کنندگان از شهر کاشان آنها را عرضه کرده باشد.

```
select pname
from (p natural join sp)
      join s on s.sn=sp.sn
where s.city = 'کاشان'
;
```

```
select pname
from (p natural join sp)
      join s using(sn)
where s.city = 'کاشان'
;
```

pname
Bolt

نام قطعات را بیابید و نام ستون آن را name بگذارید

```
select pname as name  
from p  
;
```

name
Nut
Bolt
Screw
Screw
Cam
Cog
Nut
Bolt

شماره قطعه‌های عرضه شده را بدون شماره تکراری بیابید

```
select distinct pn  
from sp  
;
```

pn
p1
p2
p3
p4
p5
p6

نام قطعاتی را بباید که وزن آنها بیشتر از ۲۰ است

```
select pname  
from p  
where weight > 20  
;
```

pname

نام شهرهای عرضه‌کنندگان را بدون تکرار بیابید

```
select distinct city  
from s  
;
```

city
London
Paris
Athens
کاشان

Use Another name for a Table in Query

```
create table t (
    a int primary key,
    name char(20)
);

insert into t values (1, 'a'),(2, 'b');
```

```
select *
from t, t as M;
```

```
select t.name
from t, t as M
where t.a < M.a;
```

```
select *
from t join t as M
on t.a < M.a;
```

a	name	a	name
1	a	1	a
1	a	2	b
2	b	1	a
2	b	2	b

name
a

a	name	a	name
1	a	2	b

Use Another name for a Table in Query

نام قطعاتی را بباید که وزن آنها دست کم از وزن یک قطعه دیگر

بیشتر باشد

نام همه قطعات را بباید به جز قطعه یا

قطعه‌هایی که کمترین وزن را دارند

```
select T.pname  
from p as T  
;
```

```
select T.pname  
from p as T, p  
where p.weight < T.weight  
;
```

```
select T.pname  
from p as T join p on  
    p.weight < T.weight  
;
```

pname
Bolt
Bolt
Bolt
Screw
Cog
Cog
Cog
Cog

مانند مسئله پیش با این تفاوت که نامهای تکراری در پاسخ نباشد

```
select distinct T.pname  
from p as T, p  
where p.weight < T.weight  
;
```

راه حل دیگر

```
select distinct T.pname  
from p as T join p on  
    p.weight < T.weight  
;
```

pname
Bolt
Screw
Cog

نام قطعاتی را بباید که وزن آنها دست کم از وزن یک قطعه دیگر کمتر باشد

```
select distinct T.pname  
from p as T join p on  
    p.weight > T.weight  
;
```

pname
Nut
Bolt
Screw
Cam

نام قطعه‌های عرضه شده را همراه با نام عرضه‌کنندگان‌شان بیابید

```
select pname, sname
from s, sp, p
where s.sn = sp.sn and
      p.pn = sp.pn
;
```

```
select pname, sname
from s natural join sp
      join p using(pn)
;
```

pname	sname
Nut	Smith
Bolt	Smith
Screw	Smith
Screw	Smith
Cam	Smith
Cog	Smith
Nut	Jones
Bolt	Jones
Bolt	Blake
Bolt	Clark
Screw	Clark
Cam	Clark
Bolt	Ali

نام قطعاتی را بیابید که وزن شان دست کم از وزن یک قطعه با رنگ قرمز کمتر باشد

```
select distinct T.pname
from p as T, p
where p.weight > T.weight
  and p.color='Red'
;
```

```
select distinct T.pname
from p as T join p on
      p.weight > T.weight
where p.color='Red'
;
```

```
select distinct p.pname
from p as p1 join p on
      p1.weight > p.weight and
      p1.color = 'Red'
;
```

pname
Nut
Bolt
Screw
Cam

نام قطعاتی را بباید که نام شهر آنها با L آغاز شده باشد

```
select pname  
from p  
where city like 'L%'  
;
```

```
select *  
from p  
;
```

pname
Nut
Screw
Cog
Nut

pn	pname	color	weight	city
p1	Nut	Red	12	London
p2	Bolt	Green	17	Paris
p3	Screw	Blue	17	Oslo
p4	Screw	Red	14	London
p5	Cam	Blue	12	Paris
p6	Cog	Red	19	London
p7	Nut	Red		London
p8	Bolt	Green		Paris

نام شهرهای قطعاتی را بباید که با P آغاز شده باشد

```
select city  
from p  
where city like 'P%'  
;
```

pname	city
Bolt	Paris
Cam	Paris
Bolt	Paris

نام قطعاتی را بیابید که نام شهر آنها پنج حرفی باشد با S آغاز شده باشد

```
select pname  
from p  
where city like 'S____'  
;
```

pname
Screw

نام شهر قطعاتی را بباید که درون نام شهر آنها رشتہ `is` وجود داشته باشد

```
select city
from p
where city like '%is%'
;
```

city
Paris

نام قطعات و شهرهای آنها را بیابید که شهر آنها دست کم سه حرفی باشند و با رشتة زیر آغاز شود.

bn_

```
select pname, city
from p
where city like "bn\_%"
;
```

escape

```
select pname
from p
where city like 'P\_%' escape '\'
;
```

```
select pname
from p
where city like 'P!_%' escape '!'
;
```

```
select pname
from p
where city like 'P#_%' escape '#'
;

select pname
from p
where city like "an\"%" escape "\"
; -- "
```

نام قطعاتی را بیابید که نام شهر آنها با an پایان نیافته باشد

```
select pname  
from p  
where city not like "%an"  
;
```

pname
Nut
Bolt
Screw
Screw
Cam
Cog
Nut
Bolt

نام قطعاتی را بیابید که در شهر پاریس باشند و پاسخ بر پایه نام قطعه از کوچک به بزرگ مرتب شده باشد.

```
select pname
from p
where city='Paris'
order by pname
;
```

نام و وزن قطعاتی را بیابید که در شهر پاریس هستند و پاسخ بر پایه وزن قطعه از کوچک به بزرگ مرتب شده باشد

```
select pname, weight
from p
where city='Paris'
order by weight
;
```

```
select pname, weight
from p
where city='Paris'
order by weight asc
;
```

نام و وزن قطعاتی را بیابید که در شهر پاریس هستند و پاسخ بر پایه وزن قطعه از
بزرگ به کوچک مرتب شده باشد

```
select pname, weight
from p
where city='Paris'
order by weight desc
;
```

pname	weight
Bolt	17
Cam	12
Bolt	

نام و وزن قطعاتی را بباید که وزن شان بین ۱۲ و ۱۴ باشد

```
select pname, weight  
from p  
where weight >= 12 and weight <= 14  
;
```

```
select pname, weight  
from p  
where weight between 12 and 14;
```

pname	weight
Nut	12
Screw	14
Cam	12

نام و وزن قطعاتی را بباید که وزن شان بین ۱۲ و ۱۴ نباشد

```
select pname, weight
from p
where not (weight >= 12 and weight <= 14)
;
```

```
select pname, weight
from p
where weight not between 12 and 14
;
```

```
select pname, weight
from p
where weight < 12 or weight > 14
;
```

pname	weight
Bolt	17
Screw	17
Cog	19

Record Comparison

نام قطعاتی را باید که عرضه کندهای در شهر آن قطعه‌ها آنها را عرضه کرده باشد

```
select pname
from p, s, sp
where (p.city, p.bn) = (s.city, sp.bn)
    and s.bn = sp.bn
;
```

```
select pname
from p, s, sp
where p.city = s.city and
      p.bn = sp.bn and
      s.bn = sp.bn
;
```

```
select pname
from p join s on
      p.city = s.city
join sp on
      (p.bn, s.bn) = (sp.bn, sp.bn)
;
```

```
select pname
from p natural join sp natural join s
;
```

pname
Nut
Screw
Cog
Bolt
Bolt
Screw

Union

```
select pname
from p
where city='Paris'
union
select pname
from p
where weight>12
;
```

```
select distinct pname
from p
where city = 'Paris' or
      weight > 12
;
```

```
select pname
from p
where city = 'kashan'
union all
select pname
from p
where weight>10
;
```

pname
Bolt
Cam
Cog
Screw

.
pname
Nut
Bolt
Screw
Screw
Cam
Cog

Style of Writing

```
select pname
from p
where city='Paris'
union
select pname
from p
where weight>12
;
```

```
select pname
from p
where city='kashan'
union
select pname
from p
where weight>10
;
```

```
select pname
from p
where city='kashan'

union

select pname
from p
where weight>10
;
```

Intersect

```
select pname
from p
where city='Paris'
intersect
select pname
from p
where weight>10
;
```

```
select distinct pname
from p
where city='Paris' and
      weight>10
;
```

```
select pname
from p
where city = 'Paris'
intersect all
select pname
from p
where weight > 10
;
```

```
select pname
from p
where city='Paris' and
      weight>10
;
```

pname
Bolt
Cam

.

pname
Bolt
Cam

Except

```
select pname
from p
where city = 'Paris'
except
select pname
from p
where weight > 14
;
```

```
select distinct pname
from p
where city='Paris' and
      weight<=14
;
```

```
select pname
from p
where city='Paris'
except all
select pname
from p
where weight>10
;
```

```
select pname
from p
where city='Paris' and
      weight<=14
;
```

pname
Cam

pname
Cam

نام شهرهای قطعاتی را بیابید که در آنها عرضه‌کننده‌ای وجود ندارد

```
select city
from p
except
select city
from s
;
```

city
Oslo

شماره قطعات و شماره عرضه‌کنندگانی را بیابید که قطعات یاد شده را آن عرضه کنندگان عرضه نکرده باشند

```
select pn, sn
from p, s
except
select pn, sn
from sp
;
```

pn	sn
p4	s3
p4	s5
p4	s6
p1	s5
p1	s6
p2	s5
p3	s2
p3	s3
p3	s4
p3	s5
p3	s6
p4	s2
p4	s3
p4	s5
p4	s6
p5	s2
p5	s3
p5	s5
p5	s6
p6	s2
p6	s3
p6	s4
p6	s5
p6	s6
p7	s1
p7	s2
p7	s3
p7	s4
p7	s5
p7	s6
p8	s1
p8	s2
p8	s3
p8	s4
p8	s5
p8	s6

نام قطعات و نام عرضه‌کنندگان را بیابید که قطعات یاد شده را آن عرضه‌کنندگان

عرضه نکرده باشند

```
select pname, sname -- نادرست
from p, s
except
select pname, sname
from p natural join sp
natural join s;
```

```
select pname, sname from p, s
except
select pname, sname
from s natural join sp
join p using(pn);
```

```
select sname , pname
from (
  select pn, sn from p, s
except
  select pn, sn from sp
) join p using (pn)
join s using (sn);
```

pname	sname
Cog	Clark
Bolt	Adams
Cog	Jones
Cam	Adams
Cam	Ali
Cam	Blake
Cam	Jones
Cog	Adams
Cog	Ali
Cog	Blake
Screw	Adams
Screw	Ali
Screw	Blake
Screw	Jones

زوج نام عرضه کنندگانی را بیابید که در یک شهر باشند

```
-- (1)  
نادرست  
select s.sname, T.sname  
from s, s as T  
where s.city = T.city  
;
```

```
-- (2)  
نادرست  
select s.sname, T.sname  
from s, s as T  
where s.city = T.city and  
    s.sn != T.sn  
;
```

```
-- (3)  
select s.sname, T.sname  
from s, s as T  
where s.city = T.city and  
    s.sn < T.sn  
;
```

```
-- (4)  
select s.sname, T.sname  
from s as T join s using(city)  
where s.sn < T.sn  
;
```

```
-- (5)  
select s.sname, T.sname  
from s as T join s on  
    T.city = s.city and  
    s.sn < T.sn  
;
```

sname	sname
Smith	Clark
Jones	Blake

Exists

نام عرضهکنندگانی را بباید که قطعه‌ای در شهر آنها باشد

```
select sname
from s
where exists (
    select *
    from p
    where p.city = s.city
)
;
```

```
select distinct sname
from s natural join p
; -- may have different result
```

sname
Smith
Jones
Blake
Clark

نام قطعاتی را بیابید که وزن آنها از دست کم یک قطعه دیگر بیشتر باشد

```
select pname
from p as T
where exists (
    select *
    from p
    where T.weight > p.weight
)
;
```

```
select distinct T.pname
from p as T join p on
T.bn < p.bn and
T.weight > p.weight
; -- wrong
```

```
select distinct T.pname
from p as T join p on
T.bn <> p.bn and
T.weight > p.weight
; -- May have different result
```

pname
Bolt
Screw
Screw
Cog

نام قطعاتی را باید که وزن آنها دست کم از یک قطعه دیگر در شهر پاریس بیشتر باشد

```
select pname
from p as T
where exists (
    select *
    from p
    where city = 'Paris' and
          T.weight > p.weight
)
;
```

pname
Bolt
Screw
Screw
Cog

نام قطعاتی را بباید که وزن آنها از همه قطعات دیگر کمتر باشد

نام قطعاتی را بباید که وزن آنها از هیچ قطعه دیگری بیشتر نباشد

```
select pname
from p as T
where not exists (
    select *
    from p
    where T.weight > p.weight
)
;
```

pname
Nut
Cam
Nut
Bolt

pname	weight
Nut	12
Cam	12
Nut	
Bolt	

نام شهرهای عرضه کنندگانی را بباید که در آن شهرها هیچ قطعه‌ای وجود ندارد

```
select city
from s
where not exists(
    select *
    from p
    where p.city = s.city
)
;
```

```
select city
from s
except all
select city
from p
;
```

city
Athens
کاشان

نام قطعه‌هایی را بباید که فقط عرضه کنندگان درون آن شهرها آنها را عرضه کرده باشند یا اصلاً عرضه نشده باشند.

نام قطعه‌هایی را بباید که عرضه کنندگان خارج از شهر آن قطعه‌ها، آنها را عرضه نکرده باشند

```
select pname
from p
where not exists(
    select *
    from s natural join sp
    where sp.pn = p.pn and
        p.city <> s.city
)
;
-- or
select pname
from p
where not exists(
    select *
    from s
    where s.city <> p.city and
        exists(
            select *
            from sp
            where sp.pn = p.pn and
                sp.sn = s.sn
        )
)
;
```

pname
Screw
Cog
Nut
Bolt

نام قطعه‌های عرضه شده‌ای را بیابید که فقط عرضه کنندگان درون آن شهرها آنها را عرضه کرده باشند.

```
select pname
from p natural join sp as T
where not exists(
    select *
    from s natural join sp
    where sp.pn = p.pn and
        p.city <> s.city
);
```

pname
Screw
Cog
Screw

```
select pname
from p natural join sp
where not exists(
    select *
    from s
    where s.city <> p.city and
        exists(
            select *
            from sp
            where sp.pn = p.pn and
                sp.sn = s.sn
        )
);
```

نام قطعه‌های عرضه شده متفاوتی را بیابید که فقط عرضه کنندگان درون آن شهرها آنها را عرضه کرده باشند

```
select distinct pname
from p natural join sp
where not exists(
    select *
    from sp,s
    where sp.sn = s.sn and
        sp.pn = p.pn and
        p.city <> s.city
)
;
```

```
select distinct pname
from p
where exists(
    select *
    from sp natural join s
    where sp.pn = p.pn and
        p.city = s.city
) and not exists(
    select *
    from sp natural join s
    where sp.pn = p.pn and
        p.city <> s.city
)
;
```

pname
Screw
Cog

نام قطعاتی را بیابید که همه عرضه کنندگان آنها را عرضه کرده باشند

نام قطعاتی را بیابید که عرضه کننده‌ای وجود نداشته باشد که این قطعات را عرضه نکرده باشد.

نام قطعاتی را می‌خواهیم که وجود نداشته باشد عرضه کننده‌ای که برایش وجود نداشته باشد عرضه‌ای که آن عرضه از آن عرضه کننده و آن قطعه باشد.

```
select pname
from p
where not exists(
    select *
    from s
    where not exists(
        select *
        from sp
        where s.sn = sp.sn
            and p.pn = sp.pn
    )
)
;
```

pname

نام قطعات متفاوتی را بباید که همه عرضه کنندگان با وضعیت بالای ۱۰۰ آنها را
عرضه کرده باشند

```
select distinct pname
from p
where not exists(
    select *
    from s
    where status > 100 and
        not exists(
            select *
            from sp
            where s.sn = sp.sn and
                p.pn = sp.pn
        )
    )
;
```

pname
Nut
Bolt
Screw
Cam
Cog

DELETE / DROP TABLE

DELETE

DML

```
delete from s  
where sn = 's5'  
;  
  
delete from p;
```

DROP TABLE

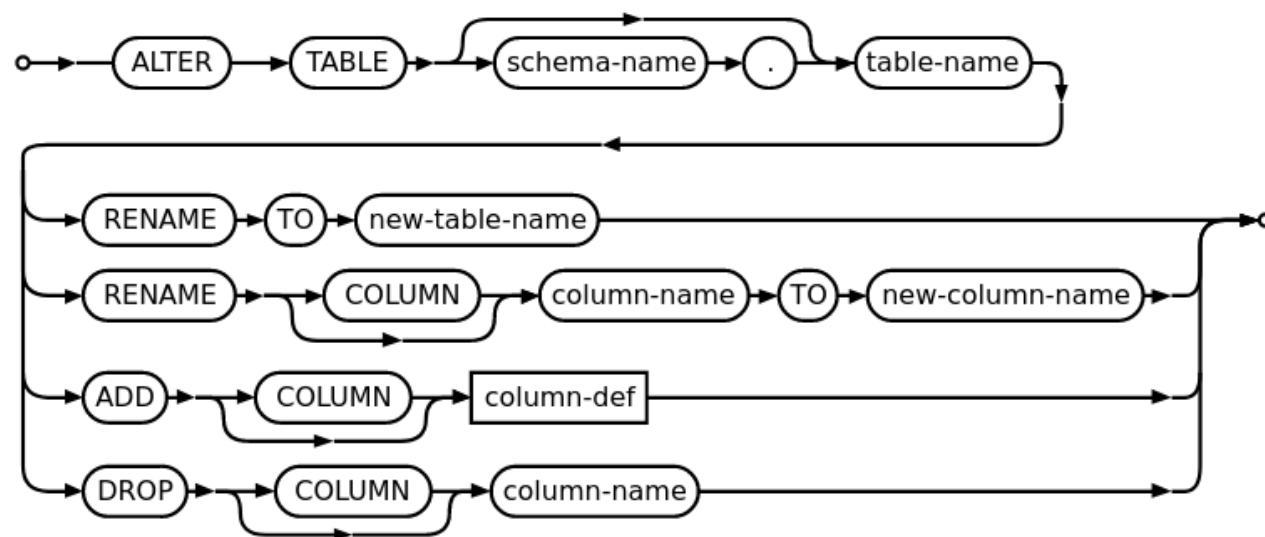
DDL

```
drop table sp;  
drop table s;  
drop table p;
```

Alter Table

DDL

```
alter table sp add "comment" varchar(50);  
  
alter table sp drop "comment";  
  
alter table sp add "comment" varchar(50) default '';
```



NULL

```
insert into p(pn, pname, color, city)
values('p7', 'Nut', 'Red', 'London')
;
```

1. Do not know the value
2. Not applicable
 - Address: city, street, alley, number

```
select pname
from p
where weight is null;
```

```
select pname
from p
where weight is not null;
```

```
create table s (
    sn      char(10) primary key,
    sname   char(30) not null,
    status  int default 0,
    city    char(20)
);
```

Aggregation Functions

Sum

asw
91

```
select sum(weight) as asw  
from p  
;
```

جمع وزن قطعات را بیابید.

sqt
3100

```
select sum(qty) as sqt  
from sp  
;
```

جمع همه عرضه‌ها(qty) را بیابید.

sqt
700

```
select sum(qty) as sqt  
from sp  
where sp.sn = 'S2'  
;
```

جمع عرضه‌های عرضه کننده S2 را بیابید.

sn	pn	qty
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

جمع وزن قطعات متفاوتی را بباید که عرضه‌کننده‌ای در شهر پاریس آنها را عرضه کرده باشد(۱).

```
1 select sum(weight) as swg
2 from p
3 where exists (
4     select *
5         from sp natural join s
6         where p.bn = sp.bn and s.city = 'Paris'
7 )
8 ;
```

swg
29

```
1 select sum(weight) as swg
2 from (p natural join sp)
3     join s using(sn)
4 where s.city = 'Paris'
5 ;
```

swg
46

جمع وزن قطعات متفاوتی را بباید که عرضه‌کننده‌ای در شهر پاریس آنها را عرضه کرده باشد(۲).

```
select sum(weight) as swg
from p
where exists(
    select *
    from s
    where s.city = 'Paris' and exists(
        select *
        from sp
        where sp.sn = s.sn and sp.pn = p.bn
    )
);
```

swg
29

جمع وزن قطعات متفاوتی را بباید که عرضه‌کننده‌ای در شهر پاریس آنها را عرضه کرده باشد^(۳).

```
select sum(weight) as swg
from p
where exists (
    select *
    from sp natural join s
    where p.bn = sp.bn and s.city = 'Paris'
)
;
```

swg
29

```
select sum(weight) as swg
from (p natural join sp)
      join s using(sn)
where s.city = 'Paris'
;
```

swg
46

```
select pn, weight, sn, s.city
from (p natural join sp)
      join s using(sn)
where s.city = 'Paris'
;
```

pn	weight	sn	city
P1	12	S2	Paris
P2	17	S2	Paris
P2	17	S3	Paris

جمع وزن قطعات متفاوتی را بباید که عرضه‌کنندگان در شهر پاریس آنها را عرضه کرده باشد(۴).

```
select sum(weight) as swg
from p
where exists (
    select *
    from sp natural join s
    where p.bn = sp.bn and s.city = 'Paris'
)
;
```

swg
29

```
select sum(distinct weight) as swg
from (p natural join sp)
    join s using(sn)
where s.city = 'Paris'
;
```

swg
29

آیا این راه حل آخری با distinct درست است؟

[راه حل دیگر](#)

جمع وزن قطعات متفاوتی را بباید که عرضه‌کنندگان در شهر پاریس آنها را عرضه کرده باشد(۵).

```
select swg
from (
    select distinct pn, sum(weight) as swg
    from (p natural join sp)
        join s using(sn)
    where s.city = 'Paris'
)
;
```

swg
46

```
select sum(weight) as swg
from (
    select distinct pn, weight
    from (p natural join sp)
        join s using(sn)
    where s.city = 'Paris'
)
;
```

swg
29

جمع وزنی قطعات عرضه شده به تعداد qty را بباید که عرضه کنندہ ای در شهر پاریس آنها را عرضه کرده باشد.

```
select sum(qty * weight) as swg
from (p natural join sp)
      join s using(sn)
where s.city = 'Paris'
;
```

swg
13800

```
select pn, qty, weight
from (p natural join sp)
      join s using(sn)
where s.city = 'Paris'
;
```

pn	qty	weight
p1	300	12
p2	400	17
p2	200	17

Average

میانگین وزن قطعه‌ها را بیابید

```
select avg(weight) as awg  
from p  
;
```

میانگین مقدار عرضه‌ها(qty) را بیابید

```
select avg(qt) as sqt  
from sp  
;
```

میانگین وزن قطعات را در شهر پاریس بیابید

```
select avg(weight) as awg
from p
where city='Paris'
;
```

awg
14.5

میانگین مقدار عرضه‌های(qty) عرضه‌کنندگان شهر پاریس را بباید

```
select avg(qty) as paqt  
from s natural join sp  
where s.city = 'Paris'  
;
```

Count

تعداد قطعات را بیابید

```
select count(pn) as awg  
from p  
;
```

awg
8

```
select count(weight) as awg  
from p  
;
```

awg
6

Count(*)

تعداد قطعات را بیابید

```
select count(*) as awg  
from p  
;
```

awg
8

```
select count(city) as ccy  
from p  
;
```

ccy
8

```
select count(*) as sqt  
from sp;
```

-- همه رکوردها را منشمارد به فیلد خاصی مربوط نیست --

sqt
12

count(distinct)

تعداد شهرهای قطعات را بیابید

```
select count(distinct city) as ccy  
from p  
;
```

ccy
3

```
select city  
from p  
;
```

city
Oslo
London
Paris
London
London
Paris

تعداد عرضه‌کنندگانی را بباید که قطعه‌ای عرضه کرده باشند

```
select count(distinct sn) as sqt  
from sp;
```

شماره‌های تکراری را نمی‌شمارد --

sqt
4

```
select distinct sn  
from sp;
```

sn
S1
S2
S3
S4

تعداد عرضه‌کنندگانی را بباید که قطعه‌ قرمزی را به تعداد عرضه(qty) بیشتر از ۵ عرضه کرده باشند.

```
select count(distinct sn) as scc
from sp natural join p
where qty > 5 and
      p.color = 'Red'
;
```

scc
3

تعداد عرضه‌کنندگانی را بباید که دست کم مقدار یکی از عرضه‌های آنها بیشتر از ۵ باشد و عرضه‌ای از قطعه‌ای به رنگ قرمز نیز داشته باشند(۱).

```
select count(*) as csn
from (
    select sn
    from sp
    where qty > 5
intersect
    select sn
    from sp join p using (pn)
    where p.color = 'Red'
);
```

```
select count(*) as csn -- May have error
from (
    select sname
    from s natural join sp
    where qty > 5
intersect
    select sname
    from s natural join sp
        join p using (pn)
    where p.color = 'Red'
);
```

تعداد عرضه‌کنندگانی را بباید که دست کم مقدار یکی از عرضه‌های آنها بیشتر از ۵ باشد و عرضه‌ای از قطعه‌ای به رنگ قرمز نیز داشته باشند(۲).

```
select count(sn) as csn
from s
where exists(
    select *
    from sp
    where sp.sn = s.sn
        and sp.qty > 5
) and exists(
    select *
    from sp natural join p
    where s.sn = sp.sn and
        p.color = 'Red'
)
;
```

csn
3

Min

کمترین وزن قطعه را بیابید

```
select min(weight) as wgt  
from    p  
;  
--- کمترین
```

max

بیشترین وزن قطعه را بباید

```
select max(weight) as wgt
from   p
;
--- بیشترین
--- به اینها تابع تجمعی گفته می شود ---aggregation function
```

group by

شماره و مجموع عرضه‌های عرضه‌کنندگانی را بباید که قطعه‌ای عرضه کرده‌اند.

```
select sn, sum(qty) as sqt  
from sp  
group by sn  
;
```

- بر پایه شماره عرضه کننده دسته‌بندی می‌کند
- سپس برای هر دسته
- شماره آن عرضه کننده (که با آن گروه‌بندی انجام شده است)

sn	sqt
s1	1300
s2	700
s3	200
s4	900
s6	350

شماره و مجموع عرضه‌های قطعاتی را بباید که عرضه شده باشند.

```
1 select pn, sum(qty) as sqt
2 from sp
3 group by pn
4 ;
```

pn	sqt
p1	600
p2	1350
p3	400
p4	500
p5	500
p6	100

شماره قطعات با وزن بیشتر از ۱۲ را همراه با جمع عرضه‌های هر کدام بیابید

```
select pn, sum(qty) as sqt
from sp join p using(pn)
where weight > 12
group by pn
;
```

pn	sqt
p2	1350
p3	400
p4	500
p6	100

نخست شرط where اعمال می‌شود سپس بر روی رکوردهای باقیمانده دسته‌بندی انجام می‌شود.

شماره قطعات با وزن بیشتر از ۱۲ را همراه با جمع عرضه‌های هر کدام باید به شرطی که بیشتر از دو عرضه کننده آنها را عرضه کرده باشند

```
1 select pn, sum(qty) as sqt
2 from sp join p using(pn)
3 where weight>12
4 group by pn
5 having count(sn)>2
6 ;
```

مانند پیشین با این تفاوت که گروههایی برگردانده می‌شوند که شرط having را نیز داشته باشند.

pn	sqt
p2	1350

گام به گام

```
select pn, qty as sqt  
from sp join p using(pn)  
where weight>12  
;
```

pn	sn	qty
p1	s1	300
p1	s2	300

pn	sn	qty
p2	s1	200
p2	s2	400
p2	s3	200
p2	s4	200
p2	s6	350

pn	sn	qty
p4	s1	200
p4	s4	300

pn	sn	qty
p3	s1	400

.

pn	sn	qty
p6	s1	100

pn	sn	qty
p5	s1	100
p5	s4	400

نام قطعاتی را بیابید که بیشتر از دو عرضه کننده آنها را عرضه کرده باشند

```
1 select distinct pname
2 from sp join p using(pn)
3 where exists(
4     select *
5     from sp as T
6     where T.sn <> sp.sn and
7         T.pn = sp.pn and exists(
8         select *
9         from sp as T2
10        where T2.sn <> sp.sn and
11            T2.sn <> sp.sn and
12            T2.sn <> T.sn and
13            T2.pn = sp.pn
14    )
15 )
```

```
1 select distinct pname
2 from p natural join (
3     select pn
4     from sp join p using(pn)
5     group by pn
6     having count(distinct sn) > 2
7 );
8
9 -- Second solution
10 select distinct pname
11 from p join sp using(pn) join
12 sp as T1 using(pn) join
13 sp as T2 using(pn)
14 where T1.sn <> sp.sn and
15 T2.sn <> sp.sn and
16 T2.sn <> T1.sn;
```

نام شهرهای قطعاتی را بباید که عرضه‌کننده‌ای با وضعیت بیشتر از ۱ نیز آنها را

عرضه کرده باشد و جمع عرضه‌های هر کدام از آن شهرهای قطعه‌ها بیشتر از ۲

باشد(۱)

```
1 select city  
2 from p  
3 ;
```

```
1 select p.city as pcity  
2 from p join sp using(pn)  
3 join s using(sn)  
4 where status > 10  
5 ;
```

```
1 select p.city as pcity --wrong  
2 from p join sp using(pn)  
3 join s using(sn)  
4 where status > 10  
5 group by p.city  
6 having sum(qty) > 20  
7 ;
```

```
1 select p.city as pcity  
2 from p  
3 where exists(  
4 select *  
5 from sp join s using(sn)  
6 where p.pn = sp.pn and  
7 status > 10  
8 )  
9 ;
```

نام شهرهای قطعاتی را بباید که عرضه‌کننده‌ای با وضعیت بیشتر از ۱۰ نیز آنها را عرضه کرده باشد و جمع عرضه‌های هر کدام از آن شهرهای قطعه‌ها بیشتر از ۲۰ باشد(۲)

```
1 select p.city as pcity
2 from p natural join sp
3 where exists(
4   select *
5     from s
6    where s.sn = sp.sn and
7      status > 10
8 )
9 group by p.city
10 having sum(qty) > 20
11 ;
```

نام شهرهای قطعاتی را بباید که عرضه‌کننده‌ای با وضعیت بیشتر از ۱ نیز آنها را عرضه کرده باشد و جمع عرضه‌های هر کدام از آن شهرهای قطعه‌ها بیشتر از ۲ باشد

```
1 select p.city
2 from p join sp using(pn)
3 where exists (
4     select *
5         from s
6         where s.sn = sp.sn and status > 10
7 )
8 group by p.city
9 having sum(qty) > 20
10 ;
```

city
London
Oslo
Paris

```
1 select p.city --- نادرست
2 from s natural join sp
3 join p using(pn)
4 where status > 10
5 group by p.city
6 having sum(qty) > 20
7 ;
8
9 select city -- نادرست
10 from sp natural join p
11 group by pn
12 having status > 10 and sum(qty) > 20
13 ;
```

```
1 select p.city
2 from p join sp using(pn)
3 where exists (
4     select *
5         from (s natural join sp) as T
6         where T.pn = p.pn and status > 10
7 )
8 group by p.city
9 having sum(qty) > 20
10 ;
```

نام شهرهای قطعاتی را بباید که فقط عرضه‌کنندگان با وضعیت بیشتر از ۱۰ آنها را عرضه کرده باشند و جمع عرضه‌های هر کدام از آن شهرهای قطعه‌ها بیشتر از ۲۰ باشد

```
1 select p.city
2 from s natural join sp
3   join p using(pn)
4 where status > 10
5 group by p.city
6 having sum(qty) > 20
7 ;
```

city
London
Oslo
Paris

```
1 select p.city
2 from sp join p using(pn) natural join
3 (
4   select sn
5     from s
6    where status > 10
7 )
8 group by p.city
9 having sum(qty) > 20
10 ;
```

نام پروژهایی را بیابید که عرضه‌کننده‌ای با وضعیت بیشتر از ۲۰ نیز آنها را عرضه کرده باشند و جمع وزنی عرضه‌های هر کدام از آن پروژه‌ها بیشتر از ۱۰۰ باشد

```
1 select jname
2 from spj join j using(jn)
3   join p using(pn)
4 where exists (
5   select *
6     from s
7     where s.sn = spj.sn and
8       s.status > 20
9   )
10 group by jn
11 having(sum(weight*qty)>100)
12 ;
```

```
1 select jname
2 from j natural join (
3   select jn
4     from spj join j using(jn)
5       join p using(pn)
6   where exists (
7     select *
8       from s
9       where s.sn = spj.sn and
10         s.status > 20
11   )
12   group by jn
13   having(sum(weight*qty)>100)
14 )
15 ;
```

شماره قطعات با وزن بیشتر از ۱۲ را همراه با جمع عرضه‌های هر کدام بباید که
بیشتر از دو عرضه کننده آنها را عرضه کرده باشند

```
1 select pn, sum(qty)
2 from sp natural join p
3 where p.weight > 12
4 group by pn
5 having count(sn)>2
6 ;
```

```
1 select pn, sum(qty) -- same result
2 from sp natural join p
3 where p.weight > 12
4 group by pn
5 having count(distinct sn)>2
6 ;
```

```
1 select pn, sum(qty) -- wrong
2 from spj natural join p
3 where p.weight > 12
4 group by pn
5 having count(sn)>2
6 ;
```

```
1 select pn, sum(qty)
2 from spj natural join p
3 where p.weight > 12
4 group by pn
5 having count(distinct sn)>2
6 ;
```

نام شهرهای قطعاتی را بباید که عرضه‌کنندگان با وضعیت بیشتر از ده ، دست کم یکی از قطعات درون آن شهرها را عرضه کرده باشد و مجموع عرضه‌های قطعه‌های آن شهرها بیشتر از ۲۰ باشد به شرطی که تعداد قطعات در آن شهر قطعه بیشتر از دو باشد(I).

```
1 select p.city -- wrong answer
2 from p join sp using(pn)
3   join s using(sn)
4 where s.status > 10
5 group by p.city
6 having sum(qty) > 20 and
7   count(distinct pn) > 2
8 ;
```

```
1 SELECT p.city -- wrong answer
2 FROM p NATURAL JOIN sp
3 WHERE EXISTS(
4   SELECT * FROM s NATURAL JOIN sp
5   WHERE s.sn=sp.sn AND p.city=s.city
6     AND s.status > 10
7   )
8 GROUP by pn
9 HAVING count(pn)>2 and sum(qty)>20
10 ;
```

```
1 select p.city
2 from p join sp using(pn)
3 where exists(
4   select *
5   from s -- Mohammad Javad Akbari
6   where status > 10 and
7     s.sn = sp.sn
8   )
9 group by p.city
10 having sum(qty) > 20 and
11   count(pn) > 2
12 ;
```

نام شهرهای قطعاتی را بباید که عرضهکنندهای با وضعیت بیشتر از ده ، دست کم یکی از قطعات درون آن شهرها را عرضه کرده باشد و مجموع عرضههای قطعههای آن شهرها بیشتر از ۲۰ باشد به شرطی که تعداد قطعات در آن شهر قطعه بیشتر از دو باشد.(II)

```
1 select p.city
2 from p join sp using(pn)
3 where exists(
4     select *
5     from s
6     where status > 10 and
7         s.sn = sp.sn
8 )
9 group by p.city
10 having sum(qty) > 20 and
11     count(distinct pn) > 2
12 ;
```

```
1 select p.city
2 from p join sp using(pn)
3 where exists(
4     select *
5     from s
6     where status > 10 and
7         s.sn = sp.sn
8 ) and exists(
9     select *
10    from p as p2
11    where p.city = p2.city and
12        p.pn <> p2.city
13 )
14 group by p.city
15 having sum(qty) > 20
16 ;
```

نام پروژهای را بباید که عرضه‌کننده‌ای با وضعیت بیشتر از ۲۰ برای آن پروژه‌ها عرضه کرده باشد و مجموع وزن قطعات عرضه شده برای آن نام پروژه (یا پروژه‌ها) بیشتر از ۱۰۰ باشد

دقت کنید مجموع وزن قطعات باید تعداد در وزن ضرب شود

```
1 select jname
2 from spj join j on
3 spj.jn = j.jn join
4 p using(pn)
5 where exists(
6     select *
7     from s
8     where s.sn = spj.sn
9         and s.status > 20
10    )
11 group by jname
12 having(sum(weight*qty)>100)
13 ;
```

نام پروژهایی را بباید که عرضه‌کننده‌ای با وضعیت بیشتر از ۲۰ برای آن پروژه‌ها عرضه کرده باشد و مجموع وزن قطعات عرضه شده برای آن پروژه (و نه همراه با همنام‌هایش) بیشتر از ۱۰۰ باشد

```
1 select jname
2 from spj join j on
3   spj.jn = j.jn join
4   p using(pn)
5 where exists(
6   select *
7     from s
8     where s.sn = spj.sn
9       and s.status > 20
10  )
11 group by jn
12 having(sum(weight*qty)>100)
13 ;
```

```
1 select jname
2 from j natural join (
3   select jn
4     from spj join j on
5       spj.jn = j.jn join
6       p using(pn)
7   where exists(
8     select *
9       from s
10      where s.sn = spj.sn
11        and s.status > 20
12    )
13   group by jn
14   having(sum(weight*qty)>100)
15  )
16 ;
```

دسته‌بندی در یک گروه

```
1 -- Totally wrong
2 select pn, count(distinct pn)
3 from p natural join sp
4 group by pname
5 -- having count(distinct pn) > 1
6 ;
```

```
1 select pname, count(sn)
2 from p natural join sp
3 group by pn
4 -- having count(distinct pn) > 1
5 ;
```

جمع وزن قطعه‌هایی را بباید که دست‌کم عرضه کنده‌ای از پاریس نیز آنها را عرضه کرده باشد.

```
1 select sum(weight) as swg
2 from (p natural join sp)
3   join s using(sn)
4 where s.city = 'Paris'
5 ;
```

swg
46

```
1 select pn, weight, sn, s.city
2 from (p natural join sp)
3   join s using(sn)
4 where s.city = 'Paris'
5 ;
```

pn	weight	sn	city
P1	12	S2	Paris
P2	17	S2	Paris
P2	17	S3	Paris

```
1 select sum(weight) as swg
2 from (
3   select distinct pn, weight
4     from (p natural join sp)
5       join s using(sn)
6     where s.city = 'Paris'
7   )
8 ;
```

swg
29

جمع وزن قطعه‌هایی را بباید که دست‌کم عرضه‌کننده‌ای از پاریس نیز آنها را عرضه کرده باشد.

```
1 select sum(weight) as swg
2 from p natural join sp
3 where exists(
4     select *
5         from s
6         where s.sn = sp.sn and
7             s.city = 'Paris'
8 )
9 ;
```

swg
46

```
1 select sum(weight) as swg
2 from (
3     select distinct pn, weight
4         from (p natural join sp)
5             join s using(sn)
6             where s.city = 'Paris'
7 )
8 ;
```

swg
29

بحث اصلی

LIMIT

```
select distinct city  
from p  
order by weight, city  
;
```

city
London
Oslo
Paris

```
select distinct city  
from p  
order by weight, city  
limit 2  
;
```

city
London
Oslo

Scalar value(I)

شماره و وزن قطعاتی را بیابید که وزن آنها از میانگین وزن همه قطعات بیشتر است.

```
select pn, weight
from p
where weight > (
    select avg(weight)
    from p
)
;
```

pn	weight
p2	17
p3	17
p6	19

Scalar value(II)

شماره و وزن قطعاتی را بباید که کمترین وزن را داشته باشند.

```
select pn, weight
from p
where weight = (
    select min(weight)
    from p
);
```

pn	weight
p1	12
p5	12

```
select pn, weight
from p
where weight = (
    select weight
    from p
    order by weight asc
    limit 1
);
```

Scalar value(V)

```
1 select pn, 1 as qt
2 from p
3 where city = 'Paris'
4 ;
```

pn	qt
P2	1
P5	1
P8	1

شماره همه قطعات را همراه با جمع تعداد عرضه های (qty) آن قطعات بیابید(۱).

```
select pn, sum(qty) as sqty
from sp
group by pn;
```

-- wrong

pn	sqty
p1	600
p2	1350
p3	400
p4	500
p5	500
p6	100

شماره همه قطعات را همراه با جمع تعداد عرضه‌های (qty) آن قطعات بیابید(۲).

```
select pn, (
    select sum(qty)
    from sp
    where p.pn = sp.pn
) as sqty
from p
```

pn	sqty
p1	600
p2	1350
p3	400
p4	500
p5	500
p6	100
p7	
p8	

شماره همه قطعات را همراه با جمع وضعیت عرضه کنندگان درون شهر آن قطعات به همراه شهر قطعه بیابید که به ترتیب نزولی وزن قطعه نشان داده شده باشند.

```
1 select pn,
2      (select sum(status)
3       from s
4       where s.city = p.city
5     ) as sum_status,
6       city
7  from p
8 order by weight desc
```

```
1 select pn, sum(status) , p.city
2 from p natural join sp natural join s
3 order by weight desc -- wrong
```

```
1 select pn, sum(status) , p.city
2 from p join s using(city)
3 order by weight desc -- wrong
```

pn	sum_status	city
p6	40	London
p2	40	Paris
p3		Oslo
p4	40	London
p1	40	London
p5	40	Paris
p7	40	London
p8	40	Paris

شماره همه قطعات را همراه جمع تعداد عرضه‌های آنها بیابید.

```
1 select pn, sum(qty) as sqty
2 from p natural join sp
3 group by pn; -- wrong
```

```
select pn, (
    select sum(qty)
    from sp
    where sp.pn = p.pn
) as sqty
from p;
```

pn	sqty
P1	600
P2	1000
P3	400
P4	500
P5	500
P6	100

pn	sqty
P1	600
P2	1000
P3	400
P4	500
P5	500
P6	100
P7	NULL
P8	NULL

Left Outer Join(I)

```
select pn, sum(qty) as sqty  
from p natural left outer join sp  
group by pn;
```

```
select pn, sum(qty) as sqty  
from p left outer join sp using(pn)  
group by pn;
```

pn	sqty
p1	600
p2	1350
p3	400
p4	500
p5	500
p6	100
p7	
p8	

Left Outer Join(II)

```
select p.pn, sum(qty) as sqty
from p left outer join sp on p.pn = sp.pn
group by p.pn;
```

```
1 select pn, (
2     select sum(qty)
3     from sp
4     where sp.pn = p.pn
5 ) as sqty
6 from p;
```

نام شهرهای همه قطعاتی را بباید که عرضه‌کننده‌ای با وضعیت بیشتر از ۵ ، دست کم یکی از قطعات درون آن شهرها را عرضه کرده باشد و مجموع عرضه‌های قطعه‌های آن شهرها بیشتر از ۲۰ باشد به شرطی که تعداد قطعات در آن شهر قطعه بیشتر از دو باشد.(III).

```
1 select p.city
2 from p left outer join sp using(pn)
3 where exists(
4     select *
5     from s
6     where status > 10 and
7         s.sn = sp.sn
8 )
9 group by p.city
10 having sum(qty) > 20 and
11     count(distinct pn) > 2
12 ;
```

Full Outer Join(I)

```
select pn, sum(qty) as sqty  
from p natural full outer join sp  
group by pn;
```

```
select pn, sum(qty) as sqty  
from p full outer join sp using(pn)  
group by pn;
```

```
select p.pn, sum(qty) as sqty  
from p full outer join sp on p.pn = sp.pn  
group by p.pn;
```

```
select distinct p.city, s.city  
from p natural left outer join s;
```

Full Outer Join(II)

```
select distinct p.city, s.city  
from p natural full outer join s;
```

```
select distinct p.city, s.city  
from p, s; -- very different result
```

نام همه شهرهای عرضه کنندگان را در کنار نام شهر قطعاتی که همشهری آنها هستند بنویسید و اگر قطعه‌ای همشهری آن عرضه کننده نبود نام شهر عرضه کننده همراه با null باید

```
select s.city, p.city  
from s left outer join p using(city);
```

```
select p.city, s.city --wrong  
from p full outer join s on p.city = s.city;
```

```
select p.city, s.city --wrong  
from p full outer join s using(city);
```

```
select p.city, s.city --wrong  
from p natural full outer join s;
```

```
select p.city, s.city  
from p natural right outer join s;
```

```
select p.city, s.city --wrong  
from p natural left outer join s;
```

نام همه شهرهای عرضه کنندگان را در کنار نام شهر قطعاتی که همشهری آنها هستند بنویسید و اگر قطعه‌ای همشهری آن عرضه کننده نبود نام شهر عرضه کننده همراه با null بیاید

```
select s.city as scity, p.city as pcity
from s left outer join p using(city);
```

```
select s.city as scity, p.city as pcity
from s, p -- wrong
```

scity	pcity
London	London
Paris	Paris
London	London
Athens	
کاشان	

is null / is not null

```
select pname  
from p  
where weight is null;
```

```
select sname  
from s  
where status is not null;
```

Scalar value(III)

شماره و وزن قطعاتی را بیابید که کمترین وزن را داشته باشند(I).

```
1 select pn, weight -- wrong
2 from p
3 order by weight asc
4 limit 1
5 ;
```

pn	weight
p7	

```
1 select pn, weight -- wrong
2 from p
3 where weight is not null
4 order by weight asc
5 limit 1
6 ;
```

pn	weight
p1	12

Scalar value(IV)

شماره و وزن قطعاتی را بیابید که کمترین وزن را داشته باشند(II).

```
1 select pn, weight -- wrong
2 from p
3 where weight is not null
4 order by weight asc
5 limit 1
6 ;
```

pn	weight
p1	12

```
1 select pn, weight
2 from p
3 where weight = (
4     select weight
5     from p
6     where weight is not null
7     order by weight asc
8     limit 1
9 )
10 ;
```

pn	weight
p1	12
p5	12

Scalar value(IV)

شماره و وزن قطعاتی را بیابید که کمترین وزن را داشته باشند.(III).

```
1 select pn, weight
2 from p
3 where weight = (
4     select weight
5         from p
6         where weight is not null
7             order by weight asc
8             limit 1
9     )
10 ;
```

pn	weight
p1	12
p5	12

```
1 select pn, weight
2 from p
3 where weight = (
4     select min(weight)
5         from p
6     )
7 ;
```

pn	weight
p1	12
p5	12

نام قطعات و شماره عرضه‌کنندگان همشهری را بیابید.

```
select pname, sn
from p natural join (select city, sn from s);
```

```
select pname, sn
from p natural join s;
```

pname	sn
Nut	s1
Nut	s4
Bolt	s2
Bolt	s3
Screw	s1
Screw	s4
Cam	s2
Cam	s3
Cog	s1
Cog	s4
Nut	s1
Nut	s4
Bolt	s2
Bolt	s3

نام پروژهای را بیابید که عرضه‌کننده‌ای با وضعیت بیشتر از ۲۰ برای آن پروژه‌ها عرضه کرده باشد و مجموع وزن قطعات عرضه شده برای آن پروژه بیشتر از ۱۰۰ باشد. دقت کنید مجموع وزن قطعات باید qty در weight ضرب شود.

```
select jname -- نادرست
from spj join j using(jn) join s
using(sn) join p using(pn)
where s.status > 20
group by jn
having sum(weight * qty) > 100)
```

```
select jname -- نادرست
from j join spj using(jn) join s
using(sn) join P using(pn)
where status > 20
group by pn
having sum(qty * weight) > 100;
```

```
select jname -- درست
from p join spj using(pn) join
j using(jn)
where exists(
  select *
  from s natural join (spj as T)
  where T.jn=j.jn and status > 20
)
group by jn
having (weight*qty)>100
```

```
select jname -- درست
from j natural join (
  select jn
  from p join spj using(pn) join
  j using(jn)
  where exists(
    select *
    from s natural join (spj as T)
    where T.jn=j.jn and status > 20
  )
  group by jn
  having (weight*qty)>100
```

نام شهرهای قطعاتی را بباید که عرضه‌کننده‌ای با وضعیت بیشتر از ده یکی از قطعات درون آن شهرها را عرضه کرده باشد و مجموع عرضه‌های قطعه‌های آن شهرها بیشتر از ۲۰ باشد به شرطی که تعداد قطعات در آن شهر قطعه بیشتر از دو باشد.

```
select p.city
from p join sp using(pn) join s using(sn)
where s.status > 10
group by p.city
having sum(qty) > 20 and count(distinct pn) > 2;
```

پاسخ نادرست

```
select p.city
from p join sp using(pn)
where exists(
    select *
    from s
    where s.status > 10 and s.sn = sp.sn
)
group by p.city
having sum(qty) > 20 and count(distinct pn) > 2;
```

پاسخ درست

شماره قطعات با وزن بیشتر از ۱۲ را همراه با جمع عرضه‌های هر کدام بباید که بیشتر از دو عرضه کننده آنها را عرضه کرده باشند.

```
select pn, sum(qty)
from sp natural join p
where p.weight > 12
group by pn
having count(sn)>2
```

شماره قطعات با وزن بیشتر از ۱۲ را همراه با جمع عرضه‌های هر کدام بیابید.

```
select pn, sum(qty)
from sp natural join p
where p.weight > 12
group by pn
```

Update(I)

```
update P  
set weight = null  
where pn='P6';
```

```
update S  
set status = status * 2  
where city = 'London';
```

```
update employees  
set email = LOWER(  
    firstname || "." || lastname || "@chinookcorp.com"  
)
```

```
update employees  
set lastname = 'Smith'  
where employeeid = 3;
```

Update(II)

```
update tableA
set B = 'abcd',
C = case
  when C = 'abc' then 'abcd'
  else C
end
where column = 1;

-- https://stackoverflow.com/a/17081004/886607
update s
set
  status = case
    when city = 'london' then status * 2
    else status
  end
```

```
update s
set
  status = case
    when city = 'London' then status * 2
    when city = 'Paris'  then status * 3
    else status
  end
```

```
update s
set
  status = case
    when city = 'London' then status / 4
    when city = 'Paris'  then status / 3
    else status
  end
```

Condition on delete

```
delete from sp
where qty < (
    select avg(qty)
    from sp
)
;
```

```
update student S
set tot_cred = (
    select sum(credits)
    from takes, course
    where takes.course_id = course.course_id
        and S.ID= takes.ID and
        takes.grade <> 'F' and
        takes.grade is not null
);
```

```
case
    when sum(credits) is not null then sum(credits)
    else 0
end
```

with(I)

```
select pn, pname
from p, (
    select avg(weight) as averagev
    from p
) as temp
where p.weight > temp.averagev
;
```

```
select pn, pname
from p
where p.weight > (
    select avg(weight)
    from p
)
;
```

```
with temp (averagev) as(
    select avg(weight)
    from p
)
select pn, pname
from p, temp
where p.weight > averagev
;
```

with(II)

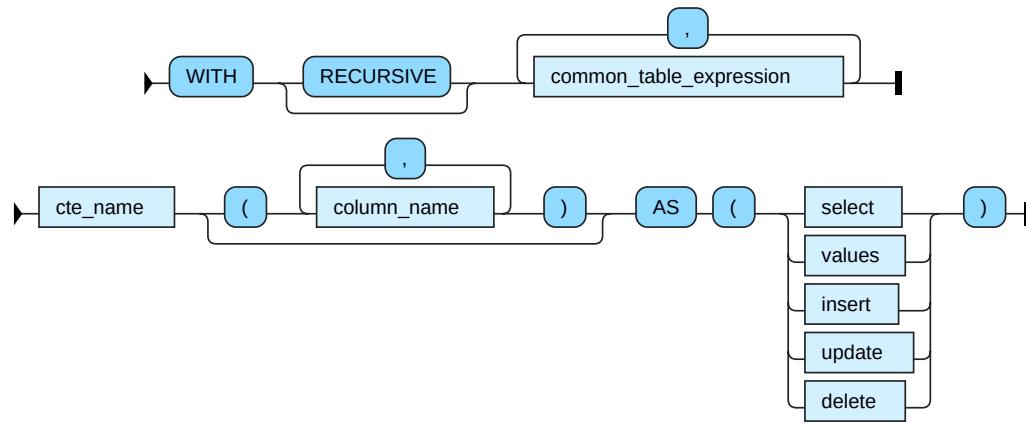
```
select pn
from (
    select avg(weight) as averagev
    from p
) as temp1, (
    select pn, sum(qty) as total
    from sp
    group by pn
) as temp2
where temp2.total > temp1.averagev
;
```

```
with temp1(averagev) as(
    select avg(weight)
    from p
),
temp2(pn, total) as(
    select pn, sum(qty)
    from sp
    group by pn
)
select pn
from temp1, temp2
where temp2.total > temp1.averagev
;
```

with(III)

```
with dept_total (dept_name, value) as(
    select dept_name, sum(salary)
    from instructor
    group by dept_name
),
dept_total_avg(value) as(
    select avg(value)
    from dept_total
)
select dept_name
from dept_total, dept_total_avg
where dept_total.value > dept_total_avg.value
```

With Diagram



Unknown

- $3 = 2 \Rightarrow \text{FALSE}$
- $3 = 3 \Rightarrow \text{TRUE}$
- $\text{NULL} = 3 \Rightarrow \text{UNKNOWN}$
- $3 = \text{NULL} \Rightarrow \text{UNKNOWN}$
- $\text{NULL} = \text{NULL} \Rightarrow \text{UNKNOWN}$
- $\text{NULL} <> 1 \Rightarrow \text{UNKNOWN}$
- $\text{NULL} > 1 \Rightarrow \text{UNKNOWN}$

```
select * -- always empty
from p
where weight = null
;

select *
from p
where weight is null
;

select *
from p
where weight > 13 or city = 'Paris'
;
```

```
select n, d
from t
where n / nullif(d, 0) > 1
;
```

```
expression IS TRUE
expression IS NOT TRUE
expression IS FALSE
expression IS NOT FALSE
expression IS UNKNOWN
expression IS NOT UNKNOWN
```

Three-valued logic

- Boolean logic
- Kleene and Priest logics

(F: false; U: unknown; T: true)

A	$\sim A$
F	T
T	F
U	U

A and B		B		
		T	F	U
A	T	T	F	U
	F	F	F	F
	U	U	F	U

A or B		B		
		T	F	U
A	T	T	T	T
	F	T	F	U
	U	T	U	U

```
1 select *
2 from List_of_tables
3 where (conditions) is unknown;
```

```
1 select *
2 from List_of_tables
3 where (conditions) is not unknown;
```

```
1 select *
2 from List_of_tables
3 where not ( (conditions) is not unknown);
```

```
1 select *
2 from List_of_tables
3 where ( (conditions) is not unknown) is not unknown;
```

Always True

"is not unknown" , "is unknown"

1. Always True or False
2. It cannot be "unknown"
3. The meaning of "null" and "unknown" is different
4. The usages of "null" and "unknown" is different

The final result of an unknown condition is False

نام قطعاتی را بباید که وزن بیشتر از ۱۷ دارند.

```
1 select pname  
2 from p  
3 where weight > 17;
```

Only Conditions with Known Values

Pr : is a large combinations of conditions

```
1 ((pr) is not unknown) and pr ;
```

```
1 select *
2 from List_of_tables
3 where (conditions) is not unknown;
```

Check

```
CREATE TABLE t (
    a NUMERIC CHECK (a >= 0),
    b NUMERIC CHECK (b >= 0),
    CHECK ( a + b <= 10 )
);
```

```
create database sp2;

create table s (
    sn      char(10) primary key,
    sname   char(30) not null,
    status  int default 10 check(status >= 10),
    city    char(20) default 'Shiraz'
);

create table p (
    pn      char(10) primary key,
    pname   char(30) not null,
    color   char(20),
    weight  numeric(9, 2) check(weight > 2 and weight < 90000),
    city    char(20)
);

create table sp (
    sn      char(10) references s on update cascade on delete cascade,
    pn      char(10) references p on update cascade on delete cascade,
    qty    int default 1 check(qty > 0),
    primary key (sn, pn)
);
```

Foreign Key Error

```
create table "Department"(
    "DN" integer default 0 primary key,
    "DeptName" varchar(30) default '',
    "MgrSSN" varchar(20) REFERENCES "Employee"("SSN")
);
create table "Employee"(
    "SSN" varchar(20) primary key,
    "name" varchar(40) not null,
    "salary" numeric(15,2) default 0,
    "Dn" integer default 0 REFERENCES "Department"("DN")
);

insert into "Department"("DN", "DeptName", "MgrSSN")
    values(1, 'computer', '');
insert into "Department"("DN", "DeptName", "MgrSSN")
    values(2, 'Chemistry', '');
```

```
-- SQLite
Result: FOREIGN KEY constraint failed
At line 14:
insert into "Department"("DN", "DeptName", "MgrSSN")
    values(1, 'computer', ''');
```

Foreign Key Error(PostgreSQL)

```
create table "Department"(
    "DN" integer default 0 primary key,
    "DeptName" varchar(30) default '',
    "MgrSSN" varchar(20) REFERENCES "Employee"("SSN")
);
```

ERROR: relation "Employee" does not exist

```
create table "Employee"(
    "SSN" varchar(20) primary key,
    "name" varchar(40) not null,
    "salary" numeric(15,2) default 0,
    "Dn" integer default 0 REFERENCES "Department"("DN")
);
```

ERROR: relation "Department" does not exist

```
create table "Department"(  
    "DN" integer default 0 primary key,  
    "DeptName" varchar(30) default '',  
    "MgrSSN" varchar(20)  
);  
  
create table "Employee"(  
    "SSN" varchar(20) primary key,  
    "name" varchar(40) not null,  
    "salary" numeric(15,2) default 0,  
    "Dn" integer default 0  
);
```

```
insert into "Department"("DN", "DeptName", "MgrSSN")  
    values(1, 'computer', '' );  
insert into "Department"("DN", "DeptName", "MgrSSN")  
    values(2, 'Chemistry', '' );  
  
insert into "Employee"("SSN", "name", "salary", "Dn")  
    values('e2', 'kamran', 1200, 1);  
  
insert into "Employee"("SSN", "name", "salary", "Dn")  
    values('e10', 'ali', 1200, 2);
```

```
update "Department" set "MgrSSN"='e2' where "DN"=1;
update "Department" set "MgrSSN"='e10' where "DN"=2;

alter table "Department" add constraint "departmentManager" foreign key("MgrSSN")
  references "Employee"("SSN") on update cascade on delete no action;

update "Employee" set "Dn"=1 where "SSN"='e2';
update "Employee" set "Dn"=2 where "SSN"='e10';

alter table "Employee" add constraint "employeeDepartment" foreign key("Dn")
  references "Department"("DN") on update cascade on delete no action;
-- set default
-- set null
```

SQLite is incomplete

```
create table tte1(
    myid integer default 0 primary key,
    salary numeric(14,2) check(salary >= 100),
    name1 varchar(20) not null,
    dept2 integer references "Department"("DN"),
    fee   numeric(14, 2) default 10, --check(salary - fee > 80)
    check(salary - fee > 80)
);

insert into tte1(myid, salary, name1, dept2, fee)
values(31, 1200, '200', 1, 'هوشنگ');
```

```
Result: FOREIGN KEY constraint failed
At line 10:
insert into tte1(myid, salary, name1, dept2, fee)
values(31, 1200, '200', 1, 'هوشنگ');
```

Same Sample in PostgreSQL

```
create table tte1(
    myid integer default 0 primary key,
    salary numeric(14,2) check(salary >= 100),
    name1 varchar(20) not null,
    dept2 integer references "Department"("DN"),
    fee    numeric(14, 2) default 10, --check(salary - fee > 80)
    check(salary - fee > 80)
);

insert into tte1(myid, salary, name1, dept2, fee)
values(31, 1200, '200', 1, 'هونگ');
```

INSERT 0 1

```
create table "Department"(
    "DN" integer default 0 primary key,
    "DeptName" varchar(30) default '',
    "MgrSSN" varchar(20)
);

create table "Employee"(
    "SSN" varchar(20) primary key,
    "name" varchar(40) not null,
    "salary" numeric(15,2) default 0,
    "Dn" integer default 0
);
```

```
insert into "Department"("DN", "DeptName", "MgrSSN")
    values(1, 'computer', '');
insert into "Department"("DN", "DeptName", "MgrSSN")
    values(2, 'Chemistry', '32');

alter table "Department" add constraint "departmentManager" foreign key("MgrSSN")
    references "Employee"("SSN") on update cascade on delete no action;
```

ERROR: insert or update on table "Department" violates foreign key constraint "departmentManager"
DETAIL: Key (MgrSSN)=(32) is not present in table "Employee".

no action, restrict, set null, set default or cascade

1. **no action:** Configuring "no action" means just that: when a parent key is modified or deleted from the database, no special action is taken.
2. **restrict:** The "restrict" action means that the application is prohibited from deleting (for on delete restrict) or modifying (for on update restrict) a parent key when there exists one or more child keys mapped to it. The difference between the effect of a restrict action and normal foreign key constraint enforcement is that the restrict action processing happens as soon as the field is updated - not at the end of the current statement as it would with an immediate constraint, or at the end of the current transaction as it would with a deferred constraint. Even if the foreign key constraint it is attached to is deferred, configuring a restrict action causes SQLite to return an error immediately if a parent key with dependent child keys is deleted or modified.
3. **set null:** If the configured action is "set null", then when a parent key is deleted (for on delete set null) or modified (for on update set null), the child key columns of all rows in the child table that mapped to the parent key are set to contain SQL null values.
4. **set default:** The "set default" actions are similar to "set null", except that each of the child key columns is set to contain the columns default value instead of null. Refer to the create table documentation for details on how default values are assigned to table columns.
5. **cascade:** A "cascade" action propagates the delete or update operation on the parent key to each dependent child key. For an "on delete cascade" action, this means that each row in the child table that was associated with the deleted parent row is also deleted. For an "on update cascade" action, it means that the values stored in each dependent child key are modified to match the new parent key values.

All

نام قطعاتی را بیابید که وزن آن قطعه‌ها از وزن همه قطعه‌های درون شهر پاریس بیشتر باشد

```
select T.pname
from p as T
where not exists(
    select *
    from p
    where city = 'Paris' and
          p.weight >= T.weight
);
```

```
select pname
from p
where weight > all(
    select weight
    from p
    where city='Paris'
)
;
```

Some

نام قطعاتی را بیابید که وزن آن قطعه‌ها از دست کم وزن یک قطعه درون شهر پاریس بیشتر باشد

```
select T.pname
from p as T
where exists(
    select *
    from p
    where city = 'Paris' and
        T.weight > p.weight
)
;
```

```
select pname
from p
where weight > some (
    select weight
    from p
    where city = 'Paris'
)
;
```

IN

نام قطعاتی را بباید که عرضه‌کننده‌ای همشهری آن قطعه‌ها باشد

```
select pname
from p
where city in (
    select city
    from s
)
;
```

کوشش کنید با exists این پرس‌وجو را حل کنید.

```
select pname
from p
where exists (
    select *
    from s
    where p.city = s.city
)
;
```

نام قطعاتی را بیابید که هیچ عرضه‌کنده‌ای در شهر آن قطعات نباشد

```
select pname
from p
where city not in (
    select city
    from s
)
;
```

کوشش کنید با exists این پرس‌وجو را حل کنید.

```
select pname
from p
where not exists (
    select *
    from s
    where p.city = s.city
)
;
```

نام قطعاتی را بیابید که هیچ عرضه کننده همشهری‌شان آنها را عرضه نکرده باشد.

```
select pname
from p
where city, pn not in (
    select city, pn
    from s natural join sp
)
;
```

کوشش کنید با exists این پرس‌وجو را حل کنید.

```
select pname
from p
where not exists(
    select *
    from s natural join sp
    where s.city = p.city
        and sp.pn = p.pn
)
;
```

نام قطعاتی را بباید که هیچ عرضه کننده همشهری‌شان هیچ قطعه‌ای را عرضه نکرده باشد.

نام قطعاتی را بباید که عرضه کننده‌ای در شهرشان نباشد که قطعه‌ای عرضه کرده باشد.

```
select pname
from p
where city not in (
    select city
    from s natural join sp
)
;
```

کوشش کنید با exists این پرس‌و‌جو را حل کنید.

```
select pname
from p
where not exist(
    select *
    from s natural join sp
    where s.city = p.city
)
;
```

نام قطعاتی را بباید که هیچ عرضه کننده همشهری‌شان هیچ قطعه‌ای را عرضه نکرده باشد.

```
select pname from p
where city not in (
    select city
    from s
    where sn not in
        (select sn from sp)
);
```

```
select pname from p
where city in (
    select city
    from s
    where sn not in
        (select sn from sp)
);
```

```
select pname from p
where city not in (
    select city from s
    where sn in
        (select sn from sp)
);
```

نام قطعاتی را بباید که وزن آنها از قطعه‌ای در شهر کاشان بزرگ‌تر باشد.

```
select pname
from p
where weight > some (
    select weight
    from p
    where city = 'Kashan'
)
;
```

```
select pname
from p as T
where exist(
    select *
    from p
    where p.city = 'Kashan'
        and T.weight > p.weight
)
;
```

شماره قطعاتی را بباید که در شهر آنها عرضه کننده‌ای با وضعیت بزرگ‌تر از ۶ وجود داشته باشد.

```
select pn
from p
where city in (
    select city
    from s
    where status > 6
)
;
```

```
select pn
from p
where city = some(
    select city
    from s
    where status > 6
)
;
```

```
select pn
from p
where city = ( -- error
    select city
    from s
    where status > 6
)
;
```

```
select city
from s
where status > 6
;
-- result has more
-- than a row
```

order by

نام و وضعیت عرضه کنندگان را به صورت صعودی بر پایه وضعیت آنها بیابید.

```
select sname, status  
from s  
order by status  
;
```

```
select sname, status  
from s  
order by status asc  
;
```

sname	status
Jones	10
Smith	20
Clark	20
Blake	30
Adams	30
Ali	40

order by desc

نام و وضعیت عرضه کنندگان را به صورت نزولی بر پایه وضعیت آنها بباید.

```
select sname, status  
from s  
order by status desc  
;
```

	sna...	status
Ali		40
Blake		30
Adams		30
Smith		20
Clark		20
Jones		10

```
create table mytemp(
    name varchar(20) not null,
    ssn  bigint primary key
);

insert into mytemp(ssn) values(20);
--- Help: not null constraint failed: mytemp.name
```

```
alter table mytemp add last_name varchar(20);

insert into mytemp(ssn, name) values(20, "ali");
```

```
select *
from mytemp;
```

شماره عرضه‌کنندگانی را بیابید که جمع وزن قطعه‌هایی که عرضه می‌کنند بیشتر از ۱۱ هزار باشد

```
select sn
from spj join p using(pn)
group by sn
having sum(weight * qty) > 11000;
```

شماره و نام عرضه کنندگان را بیابید. اگر وضعیت عرضه کننده بزرگتر از ۲۰ بود کنار مشخصات عرضه کننده کلمه big و در غیر این صورت کلمه small بگذارید.

```
select sn, sname, case
    when status > 20 then 'big'
    else 'small'
end size_of_supplier
from s
;
```

```
select pn
from p
where city = (
    select city
    from s
    where status>6
);
```

```
select pn
from p
where city in (
    select city
    from s
    where status>6
);
```

```
select pn,
       (select sum(status)
        from s
        where s.city=p.city
       ),
       city
      from p
     order by weight desc
```

```
select pn,
       (select sum(status)
        from s
        where s.city=p.city
       ) as sum_status,
       city
      from p
     order by weight desc
;
```

Unique(I)

```
CREATE TABLE Persons (
    ID int NOT NULL UNIQUE,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int
);
```

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    UNIQUE (ID)
);
```

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    CONSTRAINT UC_Person UNIQUE (ID,LastName)
);
```

```
create table "student"(
    "SSN" varchar(20) unique not null,
    "name" varchar(40) not null,
    "student_number" bigint Primary key
);

insert into
    "student"("SSN", "name", "student_number"
values
    ("کامران خدایپرستی", "973433" ,
    ("کوروش پارسایی", "9632847" ,
    ("احمد یوسفان", "(93802932 ,
```

Unique(II)

```
create table contacts(
    contact_id integer primary key,
    first_name text,
    last_name text,
    email text not null UNIQUE
);
```

```
create table shapes(
    shape_id integer primary key,
    background_color text,
    foreground_color text,
    UNIQUE(background_color,foreground_color)
);
```

```
ALTER TABLE Persons
ADD UNIQUE (ID);
```

```
ALTER TABLE Persons
ADD CONSTRAINT UC_Person
UNIQUE (ID,LastName);
```

```
ALTER TABLE Persons
DROP CONSTRAINT UC_Person;
```

Unique(III)

```
create table contacts (
    contact_id integer primary key,
    first_name text not null,
    last_name text not null,
    email text,
    phone text not null
        check (length(phone) >= 10)
);
```

```
create table products (
    product_id integer primary key,
    product_name text not null,
    list_price DECIMAL (10, 2) not null,
    discount DECIMAL (10, 2) not null
        default 0,
    check (list_price >= discount and
           discount >= 0 and
           list_price >= 0)
);
```

Unique condition

```
select pn
from p
where unique(
    select *
    from sp
    where sp.pn = p.pn
)
;
-- not implemented in PostgreSQL
```

```
select pn
from p
where not unique(
    select *
    from sp
    where sp.pn = p.pn
)
;
-- not implemented in PostgreSQL
```

نام شهرهایی را به دست آورید که عرضه کننده‌ای با وضعیت بیشتر از ۱۰۰۰ داشته باشند

```
select distinct city
from   s
where exists
( select * from s as T
  where T.city=s.city and T.status>1000
)
```

```
select distinct city
from   s
where s.status>1000
```

نام شهرهایی را به دست آورید که همه عرضه کنندگان آن شهرها وضعیت بیشتر از
... داشته باشند

```
select distinct city
from   s
where not exists
( select * from s as T
  where T.city=s.city and T.status<1000
)
```

```
select city
from   s
except
select city
from   s
where s.status<=1000
```

ENUM TYPE

```
create type Type_of_Menu as enum ('soft_menu', 'vertical_menu', 'horizontal_menu',
    'reports', 'report_with_submenu', 'report_in_submenu'
);

create table menu_access(
    menu_type Type_of_Menu,
    name char(256),
    visible boolean,
    primary key (menu_type, name)
)
```

Check constraint

```
create type valid_colors as enum ('red', 'green', 'blue');

create table t (
    color VALID_COLORS
);
```

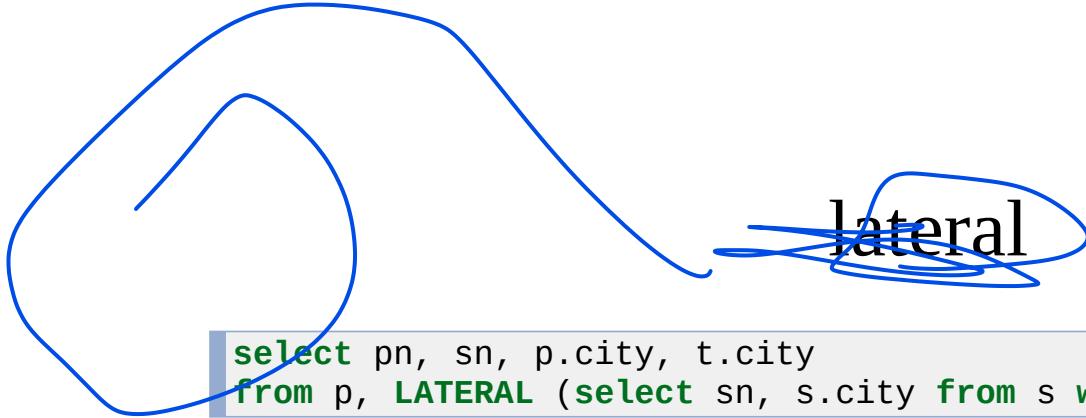
The following does not work

```
select from t where color like 'bl%';
```

```
create table t (
    colors text check (colors in ('red', 'green', 'blue'))
);
```

All native string or numeric operators work.

<https://stackoverflow.com/a/10984951>



```
select pn, sn, p.city, t.city
from p, LATERAL (select sn, s.city from s where s.city = p.city) as t;
```

```
select pn, sn, p.city, t.city -- Error
from p, (select sn, s.city from s where s.city = p.city) as t;
```

Error invalid reference to FROM-clause entry **for** table "p"
LINE 2: ... from p, (select sn, s.city from s where s.city = p.city) as...
^

HINT: There is an entry **for** table "p", but it cannot be referenced
from this part of the query.

```
select pn, sn, p.city, s.city
from p, s
where s.city = p.city;
```

```
select pn, sn, p.city, s.city
from p natural join s;
```

```
with psk as (select city from p natural join s)
  select * from psk;

with psk as (select city, weight, pn, sn from p natural join s)
  select * from psk
  where weight < (select avg(weight) from p)
;

with psk as (select city, weight, pn, sn from p natural join s)
, jsk as (select city, sn, jn, status from s NATURAL join j)
  select psk.city, psk.weight, psk.sn, psk.pn, jsk.jn from psk, jsk
  where weight < (select avg(weight) from p) and
    jsk.city = psk.city
;
```

```
-- Error
select psk.city, psk.weight, psk.sn, psk.pn, jsk.jn
from (select city, weight, pn, sn from p natural join s) as psk
, (select city, sn, jn, status from s NATURAL join j) as jsk
where weight < (select avg(weight) from psk) and
      jsk.city = psk.city
;
```

Constraint(I)

- 
- Primary key
 - not null (No field part of primary key can be null)
 - foreign keys
 - unique

pragma foreign_keys

```
pragma foreign_keys=off;  
  
pragma foreign_keys=on;  
  
pragma foreign_keys=off;  
insert into "spj"("sn", "pn", "jn", "qty") values('S7', 'P1', 'J1', 123);  
delete from "SPJ" where "sn" = 'S7';  
pragma foreign_keys=on;  
insert into "spj"("sn", "pn", "jn", "qty") values('S7', 'P1', 'J1', 123);  
Error:  
Help: foreign key constraint failed
```

SQLite uses the following terminology

- The parent table is the table that a foreign key constraint refers to. The parent table in the example in this section is the artist table. Some books and articles refer to this as the referenced table, which is arguably more correct, but tends to lead to confusion.
- The child table is the table that a foreign key constraint is applied to and the table that contains the references clause. The example in this section uses the track table as the child table. Other books and articles refer to this as the referencing table.
- The parent key is the column or set of columns in the parent table that the foreign key constraint refers to. This is normally, but not always, the primary key of the parent table. The parent key must be a named column or columns in the parent table, not the rowid.
- The child key is the column or set of columns in the child table that are constrained by the foreign key constraint and which hold the references clause.

Alter table Foreign key

MySQL / SQL Server / Oracle / MS Access

```
alter table Orders
add constraint FK_PersonOrder
foreign key (PersonID) references Persons(PersonID);
```

Vacuum

vacuum;

vacuum full

SQLite vacuum

`vacuum;`

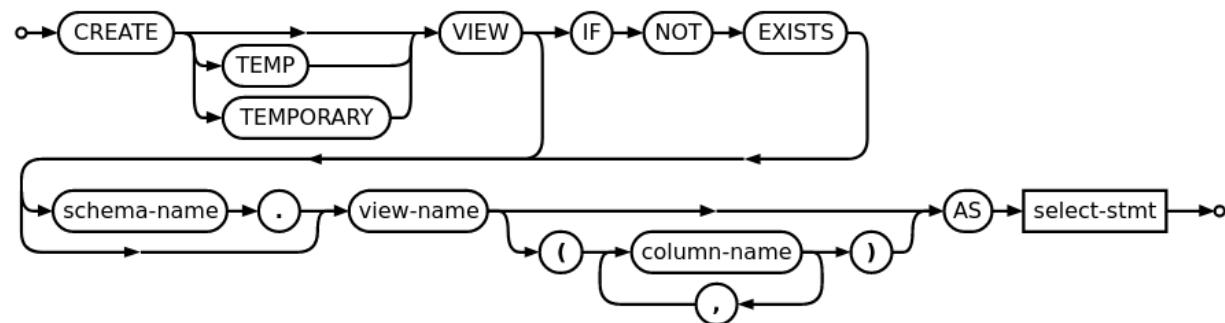
auto_vaccum

```
pragma auto_vaccum = full;
pragma auto_vaccum = incremental;
pragma auto_vaccum = none;
```

vacuum with an into clause

View

```
create view kashan_p as(  
    select *  
    from p  
    where city = 'Kashan'  
)
```



Sample View

```
create table student
  (ID      varchar(5),
   name    varchar(20) not null,
   dept_name  varchar(20),
   tot_cred  numeric(3,0) check (tot_cred >= 0),
   primary key (ID),
   foreign key (dept_name)
     references department (dept_name)
   on delete set null
);
```

```
create table student
  (ID      varchar(5),
   name    varchar(20) not null,
   dept_name  varchar(20),
   primary key (ID),
   foreign key (dept_name)
     references department (dept_name)
   on delete set null
);
```

```
update student S
set tot_cred =
  select sum(credits)
  from takes, course
  where takes.course_id = course.course_id
    and S.ID= takes.ID and
    takes.grade <> 'F' and
    takes.grade is not null
);
```

```
create view student_total as (
  select ID, name, dept_name,
  (select sum(credits)
  from takes, course
  where takes.course_id = course.course_id
    and student.ID= takes.ID and
    takes.grade <> 'F' and
    takes.grade is not null
  ) as total_cred
  from student
);
```

```
ALTER VIEW kashan_p RENAME TO kashan_parts;
DROP VIEW [ IF EXISTS ] kashan_parts;
```

```
drop view if exists m2;
drop view if exists m1;

create view m1 as
    select sn, city
    from s
    where status > 8
;

create view m2 as
    select sn, pn
    from m1 natural join p
;
```

view

- select is fine
- insert, update, delete is problematic
- unless it depends only on one table almost

1. Speed of execution



MATERIALIZED VIEW

```
CREATE MATERIALIZED VIEW table_name
  [ (column_name [, ...] ) ]
  [ WITH ( storage_parameter [= value] [, ... ] ) ]
  [ TABLESPACE tablespace_name ]
AS query
  [ WITH [ NO ] DATA ]
```



```
ALTER MATERIALIZED VIEW [ IF EXISTS ] name
    action [, ... ]
ALTER MATERIALIZED VIEW [ IF EXISTS ] name
    RENAME [ COLUMN ] column_name TO new_column_name
ALTER MATERIALIZED VIEW [ IF EXISTS ] name
    RENAME TO new_name
ALTER MATERIALIZED VIEW [ IF EXISTS ] name
    SET SCHEMA new_schema
```

where action is one of:

```
ALTER [ COLUMN ] column_name SET STATISTICS integer
ALTER [ COLUMN ] column_name SET ( attribute_option = value [, ...] )
ALTER [ COLUMN ] column_name RESET ( attribute_option [, ...] )
ALTER [ COLUMN ] column_name SET STORAGE { PLAIN | EXTERNAL | EXTENDED | MAIN }
CLUSTER ON index_name
SET WITHOUT CLUSTER
SET ( storage_parameter = value [, ...] )
RESET ( storage_parameter [, ...] )
OWNER TO new_owner
SET TABLESPACE new_tablespace
```

```
create materialized view "psc city" ("sn", "pn", "city") AS
  select "sn", "pn", "p"."city"
  from "s" join "p" on "s"."city" = "p"."city";

drop materialized view "psc city";
```

```
REFRESH MATERIALIZED VIEW name
[ WITH [ NO ] DATA ]

REFRESH MATERIALIZED VIEW order_summary;

REFRESH MATERIALIZED VIEW annual_statistics_basis WITH NO DATA;
```

```
create materialized view student_total as (
  select ID, name, dept_name,
    (select sum(credits)
     from takes, course
     where takes.course_id = course.course_id
       and student.ID= takes.ID and
       takes.grade <> 'F' and
       takes.grade is not null
    ) as total_cred
   from student
);
```

```
REFRESH MATERIALIZED VIEW student_total;
```

Recursive query

```
create table course
(course_id      varchar(8),
 title         varchar(50),
 dept_name     varchar(20),
 credits        numeric(2,0)
    check (credits > 0),
 primary key (course_id),
 foreign key (dept_name)
    references department
(dept_name)
 on delete set null
);
```

```
create table prereq
(course_id      varchar(8),
 prereq_id     varchar(8),
 primary key (course_id, prereq_id),
 foreign key (course_id) references course (course_id)
    on delete cascade,
 foreign key (prereq_id) references course (course_id)
);
```

```
with recursive rec_prereq(course_id, prereq_id) as (
    select course_id, prereq_id
    from prereq
    union
    select rec_prereq.course_id, prereq.prereq_id,
    from rec_rereq, prereq
    where rec_prereq.prereq_id = prereq.course_id
)
select *
from rec_prereq;
```

General Recursive Form

```
with recursive r(c1, c2, ...) as (
    -- Non-recursive term.
    (
        select ...
    )

    union [all]

    -- Recursive term. Notice the
    -- so-called recursive
    -- self-reference to r.
    (
        select ... from r ...
    )
)
select ... from r ...;
```

```
with recursive r(n) as (
    -- Non-recursive term.
    (
        values(1)
    )
)
union all
-- Recursive term.
(
    select n + 1
    from r
    where n < 5
)
)
select n from r order by n;
-- n 1 2 3 4 5
```

Maximum one recursive CTE (Common Table Expression)

```
with a1(n) as (select 42),
      a2(n) as (
          with recursive r(n) as (
              values(1)
              union all
              select n + 1
              from r
              where n < 5
          )
          select n from r
      ), a3(n) as (select 99)(
          select n from a1
          union all
          select n from a2
          union all
          select n from a3
      )
  order by n desc;
```

--- n, 99 42 5 4 3 2 1

END

