SQL

CS 355 Database Management System Design

S. Taneja

SQL

SQL (Structured Query Language) is a language designed to manage and query data stored in a relational database. SQL is based on relational tuple calculus and some algebra.

Originally named SEQUEL (Structured English Query Language), SQL was developed by Chamberlin and Royce at IBM, as part of the System R project.

The first commercial version of SQL was made available in 1979 by Oracle, followed shortly by IBM.

SQL was adopted as an ANSI standard in 1986, as an ISO standard in 1987. The standard has undergone many revisions and updates, with the latest being SQL:2016 (ISO/IEC 9075:2016).

SQL is supported by a large number of database products including IBM's DB2 (commercial version of system R), ORACLE, SYBASE, SQLServer, MySQL, and many more. I'm unaware of any major database vendor that fully supports the SQL standard. Many SQL implementations are, at least in some respects, vendor-dependent.

Sample relations for example queries *Based on Codd’s “suppliers” database*

```
Suppliers
 snum | sname | status |  city
------+-------+--------+--------
 S1   | Smith |     20 | London
 S2   | Jones |     10 | Paris
 S3   | Blake |     30 | Paris
 S4   | Clark |     20 | London
 S5   | Adams |     30 | Athens
 S6   | Brown |     15 | Berlin
```

```
Parts
 pnum | pname | color  | weight |  city
------+-------+--------+--------+--------
 P1   | Nut   | Red    |     12 | London
 P2   | Bolt  | Green  |     17 | Paris
 P3   | Screw | Blue   |     17 | Rome
 P4   | Screw | Red    |     14 | London
 P5   | Cam   | Blue   |     12 | Paris
 P6   | Cog   | Red    |     19 | London
 P7   | Gear  | Yellow |     18 | Berlin
```

```
Projects
 jnum |  jname   |  city
------+----------+--------
 J1   | Sorter   | Paris
 J2   | Punch    | Rome
 J3   | Reader   | Athens
 J4   | Console  | Athens
 J5   | Filler   | London
 J6   | Layer    | Oslo
 J7   | Tape     | London
```

```
Shipments
 snum | pnum | jnum | qty
------+------+------+-----
 S1   | P1   | J1   | 200
 S1   | P1   | J4   | 700
 S2   | P3   | J1   | 400
 S2   | P3   | J2   | 200
 S2   | P3   | J3   | 200
 S2   | P3   | J4   | 500
 S2   | P3   | J5   | 600
 S2   | P3   | J6   | 400
 S2   | P3   | J7   | 800
 S2   | P5   | J2   | 100
 S3   | P3   | J1   | 200
 S3   | P4   | J2   | 500
 S4   | P6   | J3   | 300
 S4   | P6   | J7   | 300
 S5   | P2   | J2   | 200
 S5   | P2   | J4   | 100
 S5   | P5   | J5   | 500
 S5   | P5   | J7   | 100
 S1   | P4   | J1   | 100
 S1   | P6   | J2   | 200
```

Why Filter Data?

Be specific about the data that we want to retrieve

Reduce the number of records we retrieve

Increase query performance

Reduce the strain on the client application

And so forth…

SQL SELECT statement

Most basic form:

```
SELECT <attribute list>
FROM   <table list>
```

Basic form:

```
SELECT <attribute list>
FROM   <table list>
WHERE  <condition>;
```

More complete form:

```
SELECT <attribute list>
FROM   <table list>
[WHERE  <condition>]
[GROUP BY <grouping attribute(s)>
[HAVING <group condition>]]
[ORDER BY <attribute list>]
```

Basic form, core relational algebra operators, and joins

SQL SELECT statement

Most basic form:

```
SELECT <attribute list>
FROM   <table list>
```

```
SELECT snum
FROM   suppliers;
```

```
 snum
------
 S1
 S2
 S3
 S4
 S5
 S6
(6 rows)
```

SQL SELECT statement

Most basic form:

```
SELECT <attribute list>
FROM   <table list>
```

```
SELECT snum, sname
FROM   suppliers;
```

```
 snum | sname
------+-------
 S1   | Smith
 S2   | Jones
 S3   | Blake
 S4   | Clark
 S5   | Adams
 S6   | Brown
(6 rows)
```

SQL SELECT statement

Most basic form:

```
SELECT <attribute list>
FROM   <table list>
```

```
SELECT snum
FROM   shipments;
```

A list of rows:

```
 snum
------
 S1
 S1
 S2
 S2
 S2
 S2
 S2
 S2
 S2
 S2
 S3
 S3
 S4
 S4
 S5
 S5
 S5
 S5
 S1
 S1
(20 rows)
```

SQL SELECT statement

Most basic form:

```
SELECT <attribute list>
FROM   <table list>
```

```
SELECT DISTINCT snum
FROM   shipments;
```

A list of rows:

```
 snum
------
 S3
 S1
 S4
 S5
 S2
(5 rows)
```

SQL SELECT statement

Basic/common form:

```
SELECT <attribute list>
FROM   <table list>
WHERE  <condition>;
```

```
SELECT snum, sname
FROM   suppliers
WHERE  city = 'Paris';
```

```
 snum | sname
------+-------
 S2   | Jones
 S3   | Blake
(2 rows)
```

SQL SELECT statement

Basic/common form:

```
SELECT sname, pnum, qty
FROM   suppliers, shipments;
```

Cross product, not a join

```
 sname | pnum | qty
-------+------+-----
 Smith | P1   | 200
 Smith | P1   | 700
 Smith | P3   | 400
 Smith | P3   | 200
 Smith | P3   | 200
 Smith | P3   | 500
 Smith | P3   | 600
 Smith | P3   | 400
 Smith | P3   | 800
 Smith | P5   | 100
 Smith | P3   | 200
 . . .
```

```
 . . .
 Brown | P3   | 800
 Brown | P5   | 100
 Brown | P3   | 200
 Brown | P4   | 500
 Brown | P6   | 300
 Brown | P6   | 300
 Brown | P2   | 200
 Brown | P2   | 100
 Brown | P5   | 500
 Brown | P5   | 100
 Brown | P4   | 100
 Brown | P6   | 200
(120 rows)
```

SQL SELECT statement

Basic/common form:

```
SELECT sname, pnum, qty
FROM   suppliers, shipments
WHERE  suppliers.snum = shipments.snum;
```

```
 sname | pnum | qty
-------+------+-----
 Smith | P1   | 200
 Smith | P1   | 700
 Jones | P3   | 400
 Jones | P3   | 200
 Jones | P3   | 200
 Jones | P3   | 500
 Jones | P3   | 600
 Jones | P3   | 400
 Jones | P3   | 800
 Jones | P5   | 100
```

```
 Blake | P3   | 200
 Blake | P4   | 500
 Clark | P6   | 300
 Clark | P6   | 300
 Adams | P2   | 200
 Adams | P2   | 100
 Adams | P5   | 500
 Adams | P5   | 100
 Smith | P4   | 100
 Smith | P6   | 200
(20 rows)
```

Quick digression – order of evaluation

SQL is more declarative than procedural, but there is an order of evaluation that is usually observed.

Different DBMS implementations could behave differently, but this is typical.

```
5 ---SELECT <attribute list>
1 ---FROM   <table list>
2 ---[WHERE  <condition>]
3 ---[GROUP BY <grouping attribute(s)>
4 ---[HAVING <group condition>]]
6 ---[ORDER BY <attribute list>]
```

SQL – joins

```
SELECT sname, pnum, qty
FROM   suppliers, shipments
WHERE  suppliers.snum = shipments.snum;
```

This is semantically equivalent to a join, but SQL-92 explicitly added join syntax.

“ANSI joins” are specified as the following table expressions in the FROM clause.

(inner joins)

```
table1 JOIN table2 ON condition
```

```
table1 JOIN table2 USING (columns)
```

```
table1 NATURAL JOIN table2
```

There’s also a (redundant) CROSS JOIN operator (“unqualified join”):

```
table1 CROSS JOIN table2
```

SQL – joins

```sql
SELECT sname, pnum, qty
FROM   suppliers JOIN shipments ON suppliers.snum = shipments.snum;
```

```
 sname | pnum | qty
-------+------+-----
 Smith | P1   | 200
 Smith | P1   | 700
 Jones | P3   | 400
 Jones | P3   | 200
 Jones | P3   | 200
 Jones | P3   | 500
 Jones | P3   | 600
 Jones | P3   | 400
 Jones | P3   | 800
 Jones | P5   | 100
```

```
 Blake | P3   | 200
 Blake | P4   | 500
 Clark | P6   | 300
 Clark | P6   | 300
 Adams | P2   | 200
 Adams | P2   | 100
 Adams | P5   | 500
 Adams | P5   | 100
 Smith | P4   | 100
 Smith | P6   | 200
(20 rows)
```

SQL – joins

```
SELECT sname, pnum, qty
FROM   suppliers JOIN shipments USING (snum);
```

```
 sname | pnum | qty
-------+------+-----
 Smith | P1   | 200
 Smith | P1   | 700
 Jones | P3   | 400
 Jones | P3   | 200
 Jones | P3   | 200
 Jones | P3   | 500
 Jones | P3   | 600
 Jones | P3   | 400
 Jones | P3   | 800
 Jones | P5   | 100
```

```
 Blake | P3   | 200
 Blake | P4   | 500
 Clark | P6   | 300
 Clark | P6   | 300
 Adams | P2   | 200
 Adams | P2   | 100
 Adams | P5   | 500
 Adams | P5   | 100
 Smith | P4   | 100
 Smith | P6   | 200
(20 rows)
```

SQL – joins

```
SELECT sname, pnum, qty
FROM   suppliers NATURAL JOIN shipments;
```

```
 sname | pnum | qty
-------+------+-----
 Smith | P1   | 200
 Smith | P1   | 700
 Jones | P3   | 400
 Jones | P3   | 200
 Jones | P3   | 200
 Jones | P3   | 500
 Jones | P3   | 600
 Jones | P3   | 400
 Jones | P3   | 800
 Jones | P5   | 100
```

```
 Blake | P3   | 200
 Blake | P4   | 500
 Clark | P6   | 300
 Clark | P6   | 300
 Adams | P2   | 200
 Adams | P2   | 100
 Adams | P5   | 500
 Adams | P5   | 100
 Smith | P4   | 100
 Smith | P6   | 200
(20 rows)
```

SQL – joins

Outer joins are also provided in SQL by using the keywords LEFT, RIGHT, or FULL in conjunction with any of the three inner join forms (ON, USING, NATURAL).

```
SELECT snum, pnum, suppliers.city
FROM   suppliers JOIN parts USING (city);
```

Inner join

```
 snum | pnum |  city
------+------+--------
 S1   | P6   | London
 S1   | P4   | London
 S1   | P1   | London
 S2   | P5   | Paris
 S2   | P2   | Paris
 S3   | P5   | Paris
 S3   | P2   | Paris
 S4   | P6   | London
 S4   | P4   | London
 S4   | P1   | London
 S6   | P7   | Berlin
(11 rows)
```

SQL – joins

Outer joins are also provided in SQL by using the keywords LEFT, RIGHT, or FULL in conjuntion with any of the three inner join forms (ON, USING, NATURAL).

```
SELECT snum, pnum, suppliers.city
FROM   suppliers LEFT JOIN parts USING (city);
```

Left outer join

```
 snum | pnum |  city
------+------+--------
 S1   | P6   | London
 S1   | P4   | London
 S1   | P1   | London
 S2   | P5   | Paris
 S2   | P2   | Paris
 S3   | P5   | Paris
 S3   | P2   | Paris
 S4   | P6   | London
 S4   | P4   | London
 S4   | P1   | London
 S5   |      | Athens
 S6   | P7   | Berlin
(12 rows)
```

SQL – joins

Outer joins are also provided in SQL by using the keywords LEFT, RIGHT, or FULL in conjuntion with any of the three inner join forms (ON, USING, NATURAL).

```
SELECT snum, pnum, suppliers.city
FROM   suppliers RIGHT JOIN parts USING (city);
```

Right outer join

```
 snum | pnum |  city
------+------+--------
 S1   | P6   | London
 S1   | P4   | London
 S1   | P1   | London
 S2   | P5   | Paris
 S2   | P2   | Paris
 S3   | P5   | Paris
 S3   | P2   | Paris
 S4   | P6   | London
 S4   | P4   | London
 S4   | P1   | London
 S6   | P7   | Berlin
      | P3   |
(12 rows)
```

SQL – joins

Outer joins are also provided in SQL by using the keywords LEFT, RIGHT, or FULL in conjuntion with any of the three inner join forms (ON, USING, NATURAL).

```
SELECT snum, pnum, suppliers.city
FROM   suppliers FULL JOIN parts USING (city);
```

Full outer join

```
 snum | pnum |  city
------+------+--------
 S1   | P6   | London
 S1   | P4   | London
 S1   | P1   | London
 S2   | P5   | Paris
 S2   | P2   | Paris
 S3   | P5   | Paris
 S3   | P2   | Paris
 S4   | P6   | London
 S4   | P4   | London
 S4   | P1   | London
 S5   |      | Athens
 S6   | P7   | Berlin
      | P3   |
(13 rows)
```

SQL – union and difference

SQL supports all the core relational algebra operators: projection, selection, cross product, union, and difference.

```
SELECT pnum FROM parts WHERE city = 'Paris'
UNION
SELECT pnum FROM shipments WHERE jnum = 'J2';
```

```
 pnum
------
 P3
 P6
 P2
 P4
 P5
(5 rows)
```

The union operation eliminates duplicates before returning the result.

SQL – union and difference

SQL supports all the core relational algebra operators: projection, selection, cross product, union, and difference.

```
SELECT pnum FROM parts WHERE city = 'Paris'
UNION ALL
SELECT pnum FROM shipments WHERE jnum = 'J2';
```

```
 pnum
------
 P2
 P5
 P3
 P5
 P4
 P2
 P6
(7 rows)
```

The union all operation leaves duplicates in the result.

SQL – union and difference

SQL supports all the core relational algebra operators: projection, selection, cross product, union, and difference.

```
SELECT pnum FROM parts
EXCEPT
SELECT pnum FROM shipments WHERE jnum = 'J2';
```

```
 pnum
------
 P1
 P7
(2 rows)
```

The except operation eliminates duplicates before returning the result.
If duplicates are needed, EXCEPT ALL should be used.

SQL – intersection

```
SELECT pnum FROM parts WHERE city = 'Paris’
INTERSECT
SELECT pnum FROM shipments WHERE jnum = 'J2';
```

```
 pnum
------
 P5
 P2
(2 rows)
```

The intersect operation eliminates duplicates before returning the result. If duplicates are needed, INTERSECT ALL should be used.

Example queries using these operations

Queries in SQL

Suppliers database

Query: Retrieve the name and id number of all suppliers in Paris having status greater than 20.

```
SELECT sname, snum
FROM   suppliers
WHERE  city = 'Paris’
AND    status > 20;
```

```
 sname | snum
-------+------
 Blake | S3
(1 row)
```

Queries in SQL

Suppliers database

Query: Retrieve the name and id number of all suppliers in Paris having status greater than 20.

```
(SELECT sname, snum
FROM    suppliers
WHERE   city = ‘Paris’)
INTERSECT
(SELECT sname, snum
FROM    suppliers
WHERE   status > 20);
```

```
 sname | snum
-------+------
 Blake | S3
(1 row)
```

Queries in SQL

Suppliers database

Query: Retrieve the name of suppliers who supply part P2.

```
SELECT sname
FROM   suppliers, shipments
WHERE  shipments.pnum = 'P2';
```

Incorrect

```
 sname
-------
 Smith
 Smith
 Jones
 Jones
 Blake
 Blake
 Clark
 Clark
 Adams
 Adams
 Brown
 Brown
(12 rows)
```

Queries in SQL

Suppliers database

Query: Retrieve the name of suppliers who supply part P2.

```
SELECT sname
FROM   suppliers, shipments
WHERE  suppliers.snum = shipments.snum
AND    shipments.pnum = 'P2';
```

```
 sname
-------
 Adams
 Adams
(2 rows)
```

Queries in SQL

Suppliers database

Query: Retrieve the name of suppliers who supply part P2.

```
SELECT sname
FROM   suppliers NATURAL JOIN shipments
WHERE  shipments.pnum = 'P2';
```

```
 sname
-------
 Adams
 Adams
(2 rows)
```

Queries in SQL

Suppliers database

Query: Retrieve the name of suppliers who supply part P2.

```
SELECT DISTINCT sname
FROM   suppliers, shipments
WHERE  suppliers.snum = shipments.snum
AND    shipments.pnum = 'P2';
```

```
 sname
-------
 Adams
(1 row)
```

```
SELECT DISTINCT sname
FROM   suppliers NATURAL JOIN shipments
WHERE  shipments.pnum = 'P2';
```

```
SELECT DISTINCT sname
FROM   suppliers JOIN shipments USING (snum)
WHERE  shipments.pnum = 'P2';
```

```
SELECT DISTINCT sname
FROM   suppliers JOIN shipments ON (suppliers.snum = shipments.snum)
WHERE  shipments.pnum = 'P2';
```

Queries in SQL

Suppliers database

Query: Retrieve the name and number of all parts that are carried in any of the following colors: red, yellow, green.

```
SELECT pname, pnum
FROM   parts
WHERE  color = 'Red' OR color = 'Yellow' OR color = 'Green';
```

```
 pname | pnum
-------+------
 Nut   | P1
 Bolt  | P2
 Screw | P4
 Cog   | P6
 Gear  | P7
(5 rows)
```

Queries in SQL

Suppliers database

Query: Retrieve the name and number of all parts that are carried in any of the following colors: red, yellow, green.

```
(SELECT pname, pnum FROM parts WHERE color = 'Red')
UNION
(SELECT pname, pnum FROM parts WHERE color = 'Yellow')
UNION
(SELECT pname, pnum FROM parts where color = 'Green');
```

```
 pname | pnum
-------+------
 Nut   | P1
 Gear  | P7
 Bolt  | P2
 Cog   | P6
 Screw | P4
(5 rows)
```

Queries in SQL

Suppliers database

Query: Retrieve the name and number of all suppliers who supply at least one red part.

```
SELECT suppliers.sname, suppliers.snum
FROM   suppliers, shipments, parts
WHERE  suppliers.snum = shipments.snum
AND    parts.pnum = shipments.pnum
AND    parts.color = 'Red';
```

```
 sname | snum
-------+------
 Smith | S1
 Smith | S1
 Blake | S3
 Clark | S4
 Clark | S4
 Smith | S1
 Smith | S1
(7 rows)
```

Queries in SQL

Suppliers database

Query: Retrieve the name and number of all suppliers who supply at least one red part.

```
SELECT DISTINCT suppliers.sname, suppliers.snum
FROM   suppliers, shipments, parts
WHERE  suppliers.snum = shipments.snum
AND    parts.pnum = shipments.pnum
AND    parts.color = 'Red';
```

```
 sname | snum
-------+------
 Clark | S4
 Blake | S3
 Smith | S1
(3 rows)
```

Queries in SQL

Suppliers database

Query: Retrieve the name and number of all suppliers who supply at least one red part.

```
SELECT DISTINCT suppliers.sname, suppliers.snum
FROM   suppliers NATURAL JOIN shipments NATURAL JOIN parts
WHERE  parts.color = 'Red';
```

!

Remember what a natural join does!

```
 sname | snum
-------+------
 Clark | S4
 Smith | S1
(2 rows)
```

Queries in SQL

Suppliers database

Query: Retrieve the name and number of all suppliers who supply at least one red part.

```
SELECT DISTINCT suppliers.sname, suppliers.snum
FROM   suppliers, shipments, parts
WHERE  suppliers.snum = shipments.snum
AND    parts.pnum = shipments.pnum
AND    parts.color = 'Red';
```

```
SELECT DISTINCT suppliers.sname, suppliers.snum
FROM   suppliers NATURAL JOIN shipments
       JOIN parts USING (pnum)
WHERE  parts.color = 'Red';
```

```
 sname | snum
-------+------
 Clark | S4
 Blake | S3
 Smith | S1
(3 rows)
```

Queries in SQL

Suppliers database

Query: Retrieve the name and number of all suppliers who supply at least one red part. List the results in ascending order of last name.

```
SELECT DISTINCT suppliers.sname, suppliers.snum
FROM   suppliers NATURAL JOIN shipments
       JOIN parts USING (pnum)
WHERE  parts.color = 'Red'
ORDER BY suppliers.sname;
```

```
 sname | snum
-------+------
 Blake | S3
 Clark | S4
 Smith | S1
(3 rows)
```

Queries in SQL

Suppliers database

Query: Retrieve the name and number of all suppliers who supply either a red part or a blue part.

```
SELECT DISTINCT suppliers.sname, suppliers.snum
FROM    suppliers
   JOIN shipments USING (snum)
   JOIN parts USING (pnum)
WHERE  (parts.color = 'Red' OR parts.color = 'Blue');
```

```
 sname | snum
-------+------
 Jones | S2
 Blake | S3
 Smith | S1
 Clark | S4
 Adams | S5
(5 rows)
```

Queries in SQL

Suppliers database

Query: Retrieve the name and number of all suppliers who supply either a red part or a blue part.

```
(SELECT sname, snum
FROM    parts
   JOIN shipments USING (pnum)
   JOIN suppliers USING (snum)
WHERE   color = 'Red')
UNION
(SELECT sname, snum
FROM    parts
   JOIN shipments USING (pnum)
   JOIN suppliers USING (snum)
WHERE   color = 'Blue');
```

```
 sname | snum
-------+------
 Blake | S3
 Jones | S2
 Smith | S1
 Clark | S4
 Adams | S5
(5 rows)
```

Queries in SQL

Suppliers database

Query: Retrieve the name and number of all suppliers who supply both a red part and a blue part.

```
(SELECT sname, snum
FROM    parts JOIN shipments USING (pnum)
        JOIN suppliers USING (snum)
WHERE   color = 'Red')
INTERSECT
(SELECT sname, snum
FROM    parts JOIN shipments USING (pnum)
        JOIN suppliers USING (snum)
WHERE   color = 'Blue');
```

```
 sname | snum
-------+------
 Blake | S3
(1 row)
```

Queries in SQL

Suppliers database

Query: Retrieve all pairs of suppliers that are located in the same city.

```
SELECT suppliersA.snum, suppliersB.snum
FROM   suppliers AS suppliersA, suppliers AS suppliersB
WHERE  suppliersA.city = suppliersB.city;
```

```
 snum | snum
------+------
 S1   | S4
 S1   | S1
 S2   | S3
 S2   | S2
 S3   | S3
 S3   | S2
 S4   | S4
 S4   | S1
 S5   | S5
 S6   | S6
```

Queries in SQL

Suppliers database

Query: Retrieve all pairs of suppliers that are located in the same city.

```
SELECT DISTINCT suppliersA.snum, suppliersB.snum
FROM   suppliers AS suppliersA, suppliers AS suppliersB
WHERE  suppliersA.city = suppliersB.city;
```

```
 snum | snum
------+------
 S1   | S4
 S1   | S1
 S2   | S3
 S2   | S2
 S3   | S3
 S3   | S2
 S4   | S4
 S4   | S1
 S5   | S5
 S6   | S6
```

Same result!

Queries in SQL

Suppliers database

Query: Retrieve all pairs of suppliers that are located in the same city.

```
SELECT suppliersA.snum, suppliersB.snum
FROM   suppliers AS suppliersA, suppliers AS suppliersB
WHERE  suppliersA.city = suppliersB.city
AND    suppliersA.snum != suppliersB.snum;
```

```
 snum | snum
------+------
 S1   | S4
 S2   | S3
 S3   | S2
 S4   | S1
```

Better result

Queries in SQL

Suppliers database

Query: Retrieve all pairs of suppliers that are located in the same city.

```
SELECT suppliersA.snum, suppliersB.snum
FROM   suppliers AS suppliersA, suppliers AS suppliersB
WHERE  suppliersA.city = suppliersB.city
AND    suppliersA.snum < suppliersB.snum;
```

```
 snum | snum
------+------
 S1   | S4
 S2   | S3
```

Best result

Queries in SQL

Suppliers database

Query: Retrieve the name and number of all suppliers who supply both a red part and a blue part.

```
select   suppliers.snum, suppliers.sname
from     suppliers, shipments as shipmentsRed, shipments as shipmentsBlue,
         parts as partsRed, parts as partsBlue
where    partsBlue.color = 'Blue'
and      partsRed.color = 'Red'
and      suppliers.snum = shipmentsRed.snum
and      suppliers.snum = shipmentsBlue.snum
and      partsRed.pnum = shipmentsRed.pnum
and      partsBlue.pnum = shipmentsBlue.pnum;
```

```
 snum | sname
------+-------
 S3   | Blake
(1 row)
```

Queries in SQL

Suppliers database

Query: Retrieve the name, number, and color of all parts that are supplied by supplier S3.

```
SELECT DISTINCT parts.pname, parts.pnum, parts.color
FROM   parts JOIN shipments USING (pnum)
WHERE  shipments.snum = 'S3';
```

```
 pname | pnum | color
-------+------+-------
 Screw | P3   | Blue
 Screw | P4   | Red
(2 rows)
```

Queries in SQL

Suppliers database

Query: Retrieve the name, number, and color of all parts that are **NOT** supplied by supplier S3.

```
SELECT DISTINCT parts.pname, parts.pnum, parts.color
FROM   parts JOIN shipments USING (pnum)
WHERE  shipments.snum != 'S3';
```

```
 pname | pnum | color
-------+------+-------
 Screw | P4   | Red
 Screw | P3   | Blue
 Cog   | P6   | Red
 Bolt  | P2   | Green
 Nut   | P1   | Red
 Cam   | P5   | Blue
(6 rows)
```

Incorrect. This returns the parts shipped by suppliers other than S3.

Queries in SQL

Suppliers database

Query: Retrieve the name, number, and color of all parts that are **NOT** supplied by supplier S3.

```
SELECT parts.pname, parts.pnum, parts.color
FROM   parts
EXCEPT
SELECT parts.pname, parts.pnum, parts.color
FROM   parts JOIN shipments USING (pnum)
WHERE  shipments.snum = 'S3';
```

```
 pname | pnum | color
-------+------+--------
 Gear  | P7   | Yellow
 Cog   | P6   | Red
 Bolt  | P2   | Green
 Nut   | P1   | Red
 Cam   | P5   | Blue
(5 rows)
```

Queries in SQL

Suppliers database

Query: Retrieve the name and number of suppliers who are currently not shipping any parts.

```
(SELECT sname, snum
FROM    suppliers)
EXCEPT
(SELECT sname, snum
FROM    suppliers JOIN shipments USING (snum));
```

```
 sname | snum
-------+------
 Brown | S6
(1 row)
```

Queries in SQL

Suppliers database

Query: Retrieve the name and number of suppliers who are currently not shipping any parts.

Start with a left join:

```
SELECT suppliers.sname, shipments.snum
FROM   suppliers LEFT JOIN shipments USING (snum);
```

```
sname | snum
-------+------
 Smith | S1
 Smith | S1
 Jones | S2
 Jones | S2
 Jones | S2
 Jones | S2
 Jones | S2
 Jones | S2
 Jones | S2
 Jones | S2
 . . .
```

```
 . . .
 Blake | S3
 Blake | S3
 Clark | S4
 Clark | S4
 Adams | S5
 Adams | S5
 Adams | S5
 Adams | S5
 Smith | S1
 Smith | S1
 Brown |
(21 rows)
```

Brown doesn't ship any parts.

Queries in SQL

Suppliers database

Query: Retrieve the name and number of suppliers who are currently not shipping any parts.

```
SELECT suppliers.sname, shipments.snum
FROM   suppliers LEFT JOIN shipments USING (snum)
WHERE  shipments.snum IS NULL;
```

```
 sname | snum
-------+------
 Brown |
(1 row)
```

But we want their supplier number to show up.

Queries in SQL

Suppliers database

Query: Retrieve the name and number of suppliers who are currently not shipping any parts.

```sql
SELECT suppliers.sname, suppliers.snum
FROM   suppliers LEFT JOIN shipments USING (snum)
WHERE  shipments.snum IS NULL;
```

```
 sname | snum
-------+------
 Brown | S6
(1 row)
```

Comparing to NULL

[more to come…]

NULL is considered a special marker that indicates the **absence of value**.

NULL corresponds to Codd's original idea of "missing or inapplicable data."

NULL is not a member of any data domain and is not considered a "value" at all. (So something can't "equal" NULL.)

```sql
SELECT suppliers.sname, suppliers.snum
FROM   suppliers LEFT JOIN shipments USING (snum)
WHERE  shipments.snum == NULL;
```

```
 sname | snum
ERROR:  operator does not exist: text == unknown
LINE 3: WHERE  shipments.snum == NULL;
                              ^
HINT:  No operator matches the given name and argument type(s). You might need to add
explicit type casts.-------+------
```

```sql
SELECT suppliers.sname, suppliers.snum
FROM   suppliers LEFT JOIN shipments USING (snum)
WHERE  shipments.snum IS NULL;
```

```
 sname | snum
-------+------
 Brown | S6
(1 row)
```

Aggregate functions

Queries in SQL

Suppliers database

Query: Retrieve the total number of suppliers in the database.

```
SELECT COUNT(snum)
FROM   suppliers;
```

```
 count
-------
     6
(1 row)
```

```
SELECT COUNT(*)
FROM   suppliers;
```

```
 count
-------
     6
(1 row)
```

```
SELECT COUNT(DISTINCT snum)
FROM   suppliers;
```

```
 count
-------
     6
(1 row)
```

Queries in SQL

Suppliers database

Query: Retrieve the total number of suppliers currently shipping parts.

```
SELECT COUNT (snum)
FROM  shipments;
```

Incorrect

```
 count
-------
    20
(1 row)
```

```
SELECT COUNT (DISTINCT snum)
FROM  shipments;
```

```
 count
-------
     5
(1 row)
```

Queries in SQL

Suppliers database

Query: Retrieve the total number of shipments of part P2.

```
SELECT COUNT(*)
FROM   shipments
WHERE  pnum = 'P2';
```

```
 count
-------
     2
(1 row)
```

Queries in SQL

Suppliers database

Query: Retrieve the total quantity of part P2 that is being shipped.

```
SELECT SUM(qty)
FROM   shipments
WHERE  pnum = 'P2';
```

```
 sum
-----
 300
(1 row)
```

Queries in SQL

A given DBMS will likely support many aggregate functions, but the standard set are:

AVG
COUNT
MIN
MAX
SUM

Group by and Having

Queries in SQL

Suppliers database

Query: For each part being shipped, retrieve the total quantity being shipped.

```sql
SELECT pnum, SUM(qty)
FROM   shipments
GROUP BY pnum;
```

```
 pnum | sum
------+------
 P2   |  300
 P6   |  800
 P4   |  600
 P3   | 3300
 P5   |  700
 P1   |  900
(6 rows)
```

Thinking about GROUP BY conceptually …

We can *conceptually* think of the GROUP BY clause arranging the table into the specifed groups and then returning one result row per group.

```
Shipments
 snum | pnum | jnum | qty
------+------+------+-----
 S1   | P1   | J1   | 200
 S1   | P1   | J4   | 700
 S2   | P3   | J1   | 400
 S2   | P3   | J2   | 200
 S2   | P3   | J3   | 200
 S2   | P3   | J4   | 500
 S2   | P3   | J5   | 600
 S2   | P3   | J6   | 400
 S2   | P3   | J7   | 800
 S2   | P5   | J2   | 100
 S3   | P3   | J1   | 200
 S3   | P4   | J2   | 500
 S4   | P6   | J3   | 300
 S4   | P6   | J7   | 300
 S5   | P2   | J2   | 200
 S5   | P2   | J4   | 100
 S5   | P5   | J5   | 500
 S5   | P5   | J7   | 100
 S1   | P4   | J1   | 100
 S1   | P6   | J2   | 200
```

```
Shipments
 snum | pnum | jnum | qty
------+------+------+-----
 S1   | P1   | J4   | 700
 S1   | P1   | J1   | 200
 S5   | P2   | J4   | 100
 S5   | P2   | J2   | 200
 S2   | P3   | J4   | 500
 S2   | P3   | J5   | 600
 S2   | P3   | J6   | 400
 S2   | P3   | J7   | 800
 S3   | P3   | J1   | 200
 S2   | P3   | J1   | 400
 S2   | P3   | J2   | 200
 S2   | P3   | J3   | 200
 S1   | P4   | J1   | 100
 S3   | P4   | J2   | 500
 S5   | P5   | J7   | 100
 S2   | P5   | J2   | 100
 S5   | P5   | J5   | 500
 S1   | P6   | J2   | 200
 S4   | P6   | J7   | 300
 S4   | P6   | J3   | 300
```

Thinking about GROUP BY conceptually …

```
SELECT pnum, SUM(qty)
FROM   shipments
GROUP BY pnum;
```

```
Shipments
 snum | pnum | jnum | qty
------+------+------+-----
 S1   | P1   | J1   | 200
 S1   | P1   | J4   | 700
 S2   | P3   | J1   | 400
 S2   | P3   | J2   | 200
 S2   | P3   | J3   | 200
 S2   | P3   | J4   | 500
 S2   | P3   | J5   | 600
 S2   | P3   | J6   | 400
 S2   | P3   | J7   | 800
 S2   | P5   | J2   | 100
 S3   | P3   | J1   | 200
 S3   | P4   | J2   | 500
 S4   | P6   | J3   | 300
 S4   | P6   | J7   | 300
 S5   | P2   | J2   | 200
 S5   | P2   | J4   | 100
 S5   | P5   | J5   | 500
 S5   | P5   | J7   | 100
 S1   | P4   | J1   | 100
 S1   | P6   | J2   | 200
```

```
Shipments
 snum | pnum | jnum | qty
------+------+------+-----
 S1   | P1   | J4   | 700
 S1   | P1   | J1   | 200
 S5   | P2   | J4   | 100
 S5   | P2   | J2   | 200
 S2   | P3   | J4   | 500
 S2   | P3   | J5   | 600
 S2   | P3   | J6   | 400
 S2   | P3   | J7   | 800
 S3   | P3   | J1   | 200
 S2   | P3   | J1   | 400
 S2   | P3   | J2   | 200
 S2   | P3   | J3   | 200
 S1   | P4   | J1   | 100
 S3   | P4   | J2   | 500
 S5   | P5   | J7   | 100
 S2   | P5   | J2   | 100
 S5   | P5   | J5   | 500
 S1   | P6   | J2   | 200
 S4   | P6   | J7   | 300
 S4   | P6   | J3   | 300
```

Quick digression – order of evaluation

SQL is more declarative than procedural, but there is an order of evaluation that is usually observed.

Different DBMS implementations could behave differently, but this is typical.

```
5 ---SELECT <attribute list>
1 ---FROM   <table list>
2 ---[WHERE  <condition>]
3 ---[GROUP BY <grouping attribute(s)>
4 ---[HAVING <group condition>]]
6 ---[ORDER BY <attribute list>]
```

Queries in SQL

Suppliers database

Query: For each part in the database, retrieve the total quantity being shipped.

```sql
SELECT parts.pnum, SUM(qty)
FROM   parts LEFT JOIN shipments USING (pnum)
GROUP BY pnum;
```

```
 pnum | sum
------+------
 P4   |  600
 P3   | 3300
 P1   |  900
 P2   |  300
 P6   |  800
 P7   |
 P5   |  700
(7 rows)
```

NULL

Not zero! [More later …]

Queries in SQL

Suppliers database

Query: For each supplier currently shipping at least one part, list their supplier number, status, and total quantity of parts being shipped. List the result in descending order of status.

```
SELECT   snum, status, SUM(qty) AS totalQty
FROM     suppliers JOIN shipments USING (snum)
GROUP BY snum
ORDER BY status DESC;
```

```
snum | status | totalqty
------+--------+----------
S5   |     30 |      900
S3   |     30 |      700
S1   |     20 |     1200
S4   |     20 |      600
S2   |     10 |     3200
(5 rows)
```

Queries in SQL

Suppliers database

Query: Retrieve the part numbers for all parts being shipped by more than one supplier.

```
SELECT pnum
FROM   shipments
GROUP BY pnum
HAVING COUNT(DISTINCT snum) > 1;
```

```
 pnum
------
 P3
 P4
 P5
 P6
(4 rows)
```

Queries in SQL

Suppliers database

Query: For those suppliers shipping more than one kind of part, retrieve their supplier number.

```
SELECT snum
FROM   shipments
GROUP BY snum
HAVING COUNT(DISTINCT pnum) > 1;
```

```
 snum
------
 S1
 S2
 S3
 S5
(4 rows)
```

Queries in SQL

Suppliers database

Query: For those suppliers shipping more than one kind of part, retrieve their supplier number **and all associated part numbers**.

Incorrect

```
SELECT snum, pnum
FROM   shipments
GROUP BY snum
HAVING COUNT(DISTINCT pnum) > 1;
```

```
ERROR:  column "shipments.pnum" must appear in the
GROUP BY clause or be used in an aggregate function
```

```
 snum | pnum
------+------
 S1   | P1
 S1   | P4
 S1   | P6
 S2   | P3
 S2   | P5
 S3   | P3
 S3   | P4
 S5   | P2
 S5   | P5
(9 rows)
```

Queries in SQL

Suppliers database

Query: For those suppliers shipping more than one kind of part, retrieve their supplier number **and all associated part numbers**.

Incorrect

```
SELECT snum, pnum
FROM   shipments
GROUP BY snum, pnum
HAVING COUNT(DISTINCT pnum) > 1;
```

```
 snum | pnum
------+------
(0 rows)
```

```
 snum | pnum
------+------
 S1   | P1
 S1   | P4
 S1   | P6
 S2   | P3
 S2   | P5
 S3   | P3
 S3   | P4
 S5   | P2
 S5   | P5
(9 rows)
```

Subqueries

Queries in SQL

Suppliers database

Part of the data source
needs to be computed

Query: For those suppliers shipping more than one kind of part, retrieve their supplier number **and all associated part numbers**.

```
SELECT DISTINCT shipments.snum, shipments.pnum
FROM   shipments
  JOIN (SELECT snum
        FROM   shipments
        GROUP BY snum
        HAVING COUNT(DISTINCT pnum) > 1) AS mult_supplier
 USING (snum)
ORDER BY snum, pnum;
```

```
 snum | pnum
------+------
 S1   | P1
 S1   | P4
 S1   | P6
 S2   | P3
 S2   | P5
 S3   | P3
 S3   | P4
 S5   | P2
 S5   | P5
(9 rows)
```

Subqueries in SQL

Subqueries can be used in many places where the type of value that they return is appropriate.

The most common places to use a subquery are the FROM and WHERE clauses.

Operators for subqueries in the WHERE clause:

x **IN** (subquery) checks if x is in the result of the subquery

x **NOT IN** (subquery) checks if x is not in the result of the subquery

EXISTS (subquery) checks if the result of subquery is non-empty

NOT EXISTS (subquery) checks if the result of subquery is empty

x comparison-operator (subquery)

x comparison-operator **ALL** (subquery)

x comparison-operator **SOME** (subquery)

- comparison-operator: =, !=, <, >, <=, >=
- result of the subquery must have only one column
- If only one tuple results from the subquery, no quantifier is needed
- ALL: comparison true for all tuples in result
- SOME: comparison true for some tuple in result

Queries in SQL

Suppliers database

Query: For those suppliers shipping more than one kind of part, retrieve their supplier number **and all associated part numbers**.

```
SELECT DISTINCT snum, pnum
FROM   shipments
WHERE  snum IN
              (SELECT snum
               FROM   shipments
               GROUP BY snum
               HAVING COUNT(DISTINCT pnum) > 1)
ORDER BY snum, pnum;
```

```
 snum | pnum
------+------
 S1   | P1
 S1   | P4
 S1   | P6
 S2   | P3
 S2   | P5
 S3   | P3
 S3   | P4
 S5   | P2
 S5   | P5
(9 rows)
```

Queries in SQL

Suppliers database

Query: For those suppliers shipping more than one kind of part, retrieve their supplier number **and all associated part numbers**.

```
SELECT DISTINCT shipmentsA.snum, shipmentsA.pnum
FROM   shipments shipmentsA
WHERE  EXISTS
              (SELECT shipmentsB.snum
               FROM   shipments shipmentsB
               WHERE  shipmentsA.snum = shipmentsB.snum
               GROUP BY snum
               HAVING COUNT(DISTINCT pnum) > 1)
ORDER BY snum, pnum;
```

```
 snum | pnum
------+------
 S1   | P1
 S1   | P4
 S1   | P6
 S2   | P3
 S2   | P5
 S3   | P3
 S3   | P4
 S5   | P2
 S5   | P5
(9 rows)
```

Queries in SQL

Suppliers database

Query: Retrieve the name and number of suppliers with current maximum status.

```
SELECT sname, snum
FROM   suppliers
WHERE status = (SELECT MAX(status)
                FROM   suppliers);
```

```
 sname | snum
-------+------
 Blake | S3
 Adams | S5
(2 rows)
```

Queries in SQL

Suppliers database

Query: Retrieve the name and number of suppliers who are currently not shipping any parts.

```
SELECT sname, snum
FROM   suppliers
WHERE  snum NOT IN
       (SELECT DISTINCT snum
        FROM   shipments);
```

```
 sname | snum
-------+------
 Brown | S6
```

Subqueries in SQL

Retrieve the name and number of suppliers who are currently shipping at least one part.

```
SELECT sname, snum
FROM   suppliers
WHERE  snum IN
       (SELECT DISTINCT snum
        FROM   shipments);
```

```
SELECT sname, snum
FROM   suppliers
WHERE  snum = SOME
       (SELECT DISTINCT snum
        FROM   shipments);
```

```
SELECT sname, snum
FROM   suppliers
WHERE  EXISTS
       (SELECT DISTINCT snum
        FROM   shipments
        WHERE shipments.snum = suppliers.snum);
```

Subqueries in SQL

Retrieve the name and number of suppliers who are currently not shipping any parts.

```
SELECT sname, snum
FROM   suppliers
WHERE  snum NOT IN
       (SELECT DISTINCT snum
        FROM   shipments);
```

```
SELECT sname, snum
FROM   suppliers
WHERE  snum != ALL
       (SELECT DISTINCT snum
        FROM   shipments);
```

```
SELECT sname, snum
FROM   suppliers
WHERE  NOT EXISTS
       (SELECT DISTINCT snum
        FROM   shipments
        WHERE shipments.snum = suppliers.snum);
```

Subqueries in SQL

Retrieve the name and number of suppliers who have the current maximum status.

```
SELECT sname, snum
FROM   suppliers
WHERE  status >= ALL
       (SELECT status
        FROM   suppliers);
```

Retrieve the name and number of suppliers who don’t have the minimum status.

```
SELECT sname, snum
FROM   suppliers
WHERE  status > SOME
       (SELECT status
        FROM   suppliers);
```

Queries in SQL

Suppliers database

Revisit using subqueries.

Query: Retrieve the name and number of all suppliers who supply **either** a red part **or** a blue part.

```
SELECT DISTINCT suppliers.sname, suppliers.snum
FROM   suppliers, shipments, parts
WHERE  suppliers.snum = shipments.snum
AND    parts.pnum = shipments.pnum
AND    parts.color IN ('Red', 'Blue');
```

```
 sname | snum
-------+------
 Jones | S2
 Blake | S3
 Smith | S1
 Clark | S4
 Adams | S5
(5 rows)
```

Queries in SQL

Suppliers database

Query: Retrieve the name and number of all suppliers who supply **both** a red part **and** a blue part.

```
SELECT sname, snum
FROM  suppliers
WHERE snum IN
             (SELECT sup.snum
              FROM   suppliers AS sup JOIN shipments USING (snum)
                     JOIN parts USING (pnum)
              WHERE  parts.color = 'Red')
AND   snum IN
             (SELECT sup.snum
              FROM   suppliers AS sup JOIN shipments USING (snum)
                     JOIN parts USING (pnum)
              WHERE  parts.color = 'Blue');
```

```
 sname | snum
-------+------
 Blake | S3
(1 row)
```

Queries in SQL

Suppliers database

Query: Retrieve the suppliers who are currently shipping to all the projects.

Two common approaches: (1) Implement using only core relational algebra operators. (2) Apply a logical quantification tautology.

Queries in SQL

Suppliers database

Query: Retrieve the name and number of suppliers who are currently shipping all the parts that are carried.

```
select distinct snum from shipments
except
select snum
  from (select snum, pnum
          from (select snum from shipments) as t1,
               (select pnum from parts) as t2
        except
        select snum, pnum from shipments) as t3;
```

```
 sname | snum
-------+------
(0 rows)
```

Queries in SQL

Suppliers database

Query: Retrieve the name and number of suppliers who are currently shipping all the red parts that are being shipped.

```
select distinct snum from shipments
except
select snum
  from (select snum, pnum
          from (select snum from shipments) as t1,
               (select pnum from parts where color = 'Red') as t2
        except
        select snum, pnum from shipments) as t3;
```

```
 sname | snum
-------+------
 Smith | S1
(1 row)
```

Queries in SQL

Suppliers database

Query: Retrieve the name and number of suppliers who are currently shipping all the red parts that are being shipped.

```
SELECT sname, snum
FROM  suppliers
WHERE NOT EXISTS
      (SELECT *
       FROM   parts
       WHERE  color = 'Red'
       AND    NOT EXISTS
              (SELECT *
               FROM   shipments
               WHERE  shipments.pnum = parts.pnum
               AND    shipments.snum = suppliers.snum));
```

```
 sname | snum
-------+------
 Smith | S1
(1 row)
```

Queries in SQL

Suppliers database

Query: Retrieve the name and number of the parts that are being shipped to all the projects in Paris.

```
SELECT pname, pnum
FROM  parts
WHERE NOT EXISTS
      (SELECT *
       FROM   projects
       WHERE  city = 'Paris'
       AND    NOT EXISTS
              (SELECT *
               FROM   shipments
               WHERE  shipments.jnum = projects.jnum
               AND    shipments.pnum = parts.pnum));
```

```
 pname | pnum
-------+------
 Nut   | P1
 Screw | P3
 Screw | P4
(3 rows)
```

Things that can go in the attribute list

SQL SELECT statement – basic examples

```
SELECT *
FROM   suppliers
ORDER BY status;
```

```
 snum | sname | status |  city
------+-------+--------+---------
 S2   | Jones |     10 | Paris
 S6   | Brown |     15 | Berlin
 S1   | Smith |     20 | London
 S4   | Clark |     20 | London
 S3   | Blake |     30 | Paris
 S5   | Adams |     30 | Athens
(6 rows)
```

SQL SELECT statement – basic examples

```
SELECT pnum, weight * 454 AS “Weight in Grams”
FROM   parts;
```

```
 pnum | Weight in Grams
------+-----------------
 P1   |            5448
 P2   |            7718
 P3   |            7718
 P4   |            6356
 P5   |            5448
 P6   |            8626
 P7   |            8172
(7 rows)
```

Views

Queries in SQL

Suppliers database

Query: Retrieve the name and number of all parts that are carried in any of the following colors: red, yellow, green.

```
CREATE VIEW Red_Parts AS
(SELECT pname, pnum FROM parts WHERE color = 'Red');

CREATE VIEW Yellow_Parts AS
(SELECT pname, pnum FROM parts WHERE color = 'Yellow');

CREATE VIEW Green_Parts AS
(SELECT pname, pnum FROM parts where color = 'Green';

(SELECT * FROM Red_Parts) UNION (SELECT * FROM Green_Parts) UNION
(SELECT * FROM Yellow_Parts);
```

```
 pname | pnum
-------+------
 Nut   | P1
 Gear  | P7
 Bolt  | P2
 Cog   | P6
 Screw | P4
(5 rows)
```

Conditional functions

Queries in SQL

Suppliers database

Query 17 For each part in the database, retrieve the total quantity being shipped.

```
SELECT parts.pnum, COALESCE(SUM(qty), 0)
FROM   parts LEFT JOIN shipments USING (pnum)
GROUP BY pnum;
```

```
 pnum | coalesce
------+----------
 P4   |      600
 P3   |     3300
 P1   |      900
 P2   |      300
 P6   |      800
 P7   |        0
 P5   |      700
(7 rows)
```

The COALESCE function returns the first of its arguments that is not NULL.

Summary: Some tips before you leave

Test and Troubleshoot

- DO NOT wait until the end to test queries
- Test after each join or filter
- Are you getting the results you expect?
- Start small and go step-by-step when troubleshooting a query

Format and Comment

- Use correct formatting
- Comment strategically
- Clean code and comments when you revisit and hand off code

Review

- Always review old queries
- Business rules
- Data changes, data indicators
- Work the problem from beginning ot end

Courtesy: SQL for Data Science at UC Davis

Haven’t had enough SQL? Or do you want to keep practicing to improve your skills? SQL puzzles are a great way to do this! Below is a very popular resources for practicing SQL Puzzles.
SQL Authority: SQL Puzzles

In addition, many of you may desire to get a new job or position in this are. Below is a resource that includes quizzes and is recommended by many recruiters to practice SQL for a data science interview.
SQLZOO