

# SQL – Subqueries

---

## CS 4750 Database Systems

[A. Silberschatz, H. F. Korth, S. Sudarshan, Database System Concepts, Ch.5.3]

# Aggregation: Order of Actions

```
SELECT select_list
FROM table_source
[WHERE search_condition]
[GROUP BY group_by_expression]
[HAVING search_condition]
[ORDER BY order_expression [ASC | DESC] ]
```

## Order matters

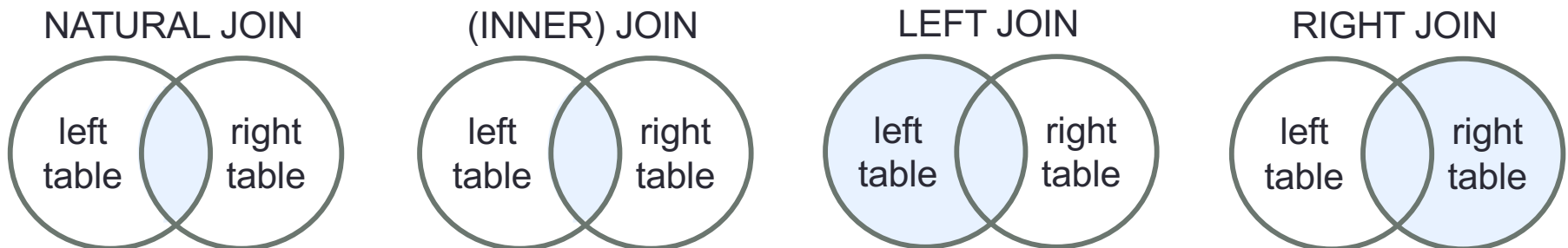
1. The FROM clause generates the data set
2. The WHERE clause filters the data set generated by the FROM clause
3. The GROUP BY clause aggregates the data set that was filtered by the WHERE clause (note: GROUP BY does not sort the result set)
4. The HAVING clause filters the data set that was aggregated by the GROUP BY clause
5. The SELECT clause transforms the filtered aggregated data set
6. The ORDER BY clause sorts the transformed data set

# Joins

Join combines data across tables

- Nested-loop
- Natural join (most common)
- Inner join (filter Cartesian product)
- Outer joins (preserve non-matching tuples)
- Self join pattern

Different joining techniques can be used to achieve particular goals



# Recap 1: JOIN

Find the total number of unique sailors who have reserved each boat (ordered the number of sailors in descending order).  
Display the count, boat name, and boat id

Boats (bid, bname, color)

Sailors (sid, sname, rating, age)

Reserves (sid, bid, day)

Number of unique sailors  
who have reserved this boat

```
SELECT COUNT(DISTINCT sid), bname, bid
FROM Boats NATURAL JOIN Reserves
GROUP BY bname, bid
ORDER BY COUNT(DISTINCT sid) DESC
```

Refer to <http://www.cs.virginia.edu/~up3f/cs4750/inclass/alldbs.sql>

# Recap 2: JOIN

Find the average age of sailors who have reserved each boat.  
Show boat name, boat id, and the average age.

Order results by boat id.

```
Boats (bid, bname, color)
Sailors (sid, sname, rating, age)
Reserves (sid, bid, day)
```

```
SELECT bname, bid, AVG(age)
FROM Sailors NATURAL JOIN Reserves NATURAL JOIN Boats
GROUP BY bname, bid
ORDER BY bid
```

Refer to <http://www.cs.virginia.edu/~up3f/cs4750/inclass/alldbs.sql>

# Recap 3: JOIN

Find the average age of sailors who have reserved each boat? Show boat name, bid, and the average age. Order results by bid. (*from Recap 2*)

In addition, only show the boat info where the average age of sailors who have reserved that boat is > 35 years old.

Boats (bid, bname, color)

Sailors (sid, sname, rating, age)

Reserves (sid, bid, day)

```
SELECT bname, bid, AVG(age)
FROM Sailors NATURAL JOIN Reserves NATURAL JOIN Boats
GROUP BY bname, bid
HAVING AVG(age) > 35
ORDER BY bid
```

Refer to <http://www.cs.virginia.edu/~up3f/cs4750/inclass/alldbs.sql>

# Recap 4: Self Join

Find the average salary for each job that is greater than the average salary of all employees

practice\_emp

empno	ename	job	sal
7369	Smith	Clerk	1200
7499	Allen	Salesman	2000
7521	Ward	Salesman	1650
7566	Jones	Manager	3375
7654	Martin	Salesman	1650
7698	Blake	Manager	3250
7782	Clark	Manager	2850
7788	Scott	Analyst	3500
7839	King	President	6500
7844	Turner	Salesman	1900
7876	Adams	Clerk	1500
7900	James	Clerk	1350
7902	Ford	Analyst	3500
7934	Miller	Clerk	1700

```
SELECT E1.job, AVG(E1.sal) AS AvgSal
FROM practice_emp E1, practice_emp E2
GROUP BY E1.job
HAVING AVG(E1.sal) > AVG(E2.sal)
```

## Idea:

1. Self-join practice\_emp
2. Use one copy to aggregate, find average salary of all employees
3. Use one copy to keep the original job, find average salary of each job
4. Compare average salary of each job and average salary of all employees

(Note: The table shows sample data, not a complete set of data,  
refer to <https://www.cs.virginia.edu/~up3f/cs4750/assigns/employees.sql>)

# Recap 5: Self Join

Find all students (sid) who live in the same city and on the same street as their mentor

```
Mentorship (mentee_sid, mentor_sid)
    -- mentor_sid is a mentor of another student mentee_sid
Study (sid, credits) -- credits the student has taken
Enrollment (dept_id, sid) -- dept the student is enrolled in
Student (sid, street, city) -- street, city the student lives
```

```
SELECT S1.sid
FROM   Student S1, Student S2, Mentorship M
WHERE  S1.sid = M.mentee_sid AND
       S2.sid = M.mentor_sid AND
       S1.street = S2.street AND
       S1.city = S2.city
```

## Idea:

1. Self-join Student
2. Use one copy for mentee
3. Use one copy for mentor
4. Check the pair of mentee and mentor against Mentorship
5. Check that the street and city match



# Subqueries: Core Idea

Split your problem into sub-problems



Solve them separately



Compose them

The smaller the problem, the simpler to solve, the easier to debug

# Subqueries

- **Subquery** = a query that is part of another query
- A subquery can have subqueries

## Usage:

- Return a **single constant** that can be used to compute an associated value in a **SELECT** clause
- Return a **single constant** that can be compared to another value in a **WHERE** clause
- Return a **relation** that can be compared or evaluated in a **WHERE** clause
- Return a **relation** that can be used as input for another query, in a **FROM** clause

# Equivalent Query Example

Find the average salary for each job

practice\_emp

empno	ename	job	sal
7369	Smith	Clerk	1200
7499	Allen	Salesman	2000
7521	Ward	Salesman	1650
7566	Jones	Manager	3375
7654	Martin	Salesman	1650
7698	Blake	Manager	3250
7782	Clark	Manager	2850
7788	Scott	Analyst	3500
7839	King	President	6500
7844	Turner	Salesman	1900
7876	Adams	Clerk	1500
7900	James	Clerk	1350
7902	Ford	Analyst	3500
7934	Miller	Clerk	1700

```
SELECT job, AVG(sal)
FROM practice_emp
GROUP BY job
```



job	AvgSal
Analyst	3500.0000
Clerk	1437.5000
Manager	3158.3333
President	6500.0000
Salesman	1800.0000

Idea (self join):

1. Self-join practice\_emp
2. Use one copy to aggregate, group by job
3. Use one copy to keep the original job

```
SELECT E1.job, AVG(E2.sal) AS AvgSal
FROM practice_emp E1, practice_emp E2
WHERE E1.job = E2.job
GROUP BY E1.job
```

(Note: The table shows sample data, not a complete set of data,  
refer to <https://www.cs.virginia.edu/~up3f/cs4750/assigns/employees.sql>)

# (Equivalent) Subquery (SELECT)

Find the average salary for each job

practice\_emp

empno	ename	job	sal
7369	Smith	Clerk	1200
7499	Allen	Salesman	2000
7521	Ward	Salesman	1650
7566	Jones	Manager	3375
7654	Martin	Salesman	1650
7698	Blake	Manager	3250
7782	Clark	Manager	2850
7788	Scott	Analyst	3500
7839	King	President	6500
7844	Turner	Salesman	1900
7876	Adams	Clerk	1500
7900	James	Clerk	1350
7902	Ford	Analyst	3500
7934	Miller	Clerk	1700

**“Correlated”** query  
Recomputed for each tuple  
(can't be run independently  
of the outer query)

```
SELECT E1.job,  
      (SELECT AVG(E2.sal)  
       FROM practice_emp AS E2  
       WHERE E1.job = E2.job) AS AvgSal  
FROM practice_emp E1  
GROUP BY E1.job
```

job	AvgSal
Analyst	3500.0000
Clerk	1437.5000
Manager	3158.3333
President	6500.0000
Salesman	1800.0000

Idea:

1. Group by job
2. For each tuple, compute aggregate

A subquery in **SELECT** returns a **single value** – used to compute an associated value

# (Equivalent) Subquery (FROM)

Find the average salary for each job

**“Uncorrelated”** query  
Independent of outer query

practice\_emp

empno	ename	job	sal
7369	Smith	Clerk	1200
7499	Allen	Salesman	2000
7521	Ward	Salesman	1650
7566	Jones	Manager	3375
7654	Martin	Salesman	1650
7698	Blake	Manager	3250
7782	Clark	Manager	2850
7788	Scott	Analyst	3500
7839	King	President	6500
7844	Turner	Salesman	1900
7876	Adams	Clerk	1500
7900	James	Clerk	1350
7902	Ford	Analyst	3500
7934	Miller	Clerk	1700

```
SELECT E1.job, AvgSal
FROM practice_emp E1,
  (SELECT job, AVG(sal) AS AvgSal
   FROM practice_emp
   GROUP BY job) AS E2
WHERE E1.job = E2.job
GROUP BY E1.job
```

job	AvgSal
Analyst	3500.0000
Clerk	1437.5000
Manager	3158.3333
President	6500.0000
Salesman	1800.0000

Idea:

1. Compute aggregate for each job
2. Join the original practice\_emp

A subquery in **FROM** returns a **relation** – used as input for another query

# Subqueries in WHERE

Find employee name (or names) who earns the highest salary for each job

practice\_emp

empno	ename	job	sal
7369	Smith	Clerk	1200
7499	Allen	Salesman	2000
7521	Ward	Salesman	1650
7566	Jones	Manager	3375
7654	Martin	Salesman	1650
7698	Blake	Manager	3250
7782	Clark	Manager	2850
7788	Scott	Analyst	3500
7839	King	President	6500
7844	Turner	Salesman	1900
7876	Adams	Clerk	1500
7900	James	Clerk	1350
7902	Ford	Analyst	3500
7934	Miller	Clerk	1700

```
SELECT E1.ename
FROM practice_emp E1
WHERE E1.sal =
    (SELECT MAX(E2.sal)
     FROM practice_emp AS E2
     WHERE E1.job = E2.job)
```

**“Correlated”**

ename
Allen
Jones
Scott
King
Ford
Miller

[more subqueries in WHERE later]

A subquery in **WHERE** returns a **single value** – to be compared to another value in a **WHERE** clause

# Subqueries in WITH

Find employee name (or names) who earns the highest salary for each job

**"Uncorrelated"**

practice\_emp

empno	ename	job	sal
7369	Smith	Clerk	1200
7499	Allen	Salesman	2000
7521	Ward	Salesman	1650
7566	Jones	Manager	3375
7654	Martin	Salesman	1650
7698	Blake	Manager	3250
7782	Clark	Manager	2850
7788	Scott	Analyst	3500
7839	King	President	6500
7844	Turner	Salesman	1900
7876	Adams	Clerk	1500
7900	James	Clerk	1350
7902	Ford	Analyst	3500
7934	Miller	Clerk	1700

```
WITH temp AS
```

```
(SELECT job, MAX(sal) AS maxSal  
FROM practice_emp  
GROUP BY job)
```

```
SELECT E1.ename
```

```
FROM practice_emp AS E1, temp AS T
```

```
WHERE E1.sal = T.maxSal AND  
      E1.job = T.job
```

ename

Allen

Jones

Scott

King

Ford

Miller

A subquery in **WITH** clause returns a temporary **relation** that can be used by an associated query

[WITH -- not supported by MySQL 5.6, 5.7; work on MySQL 8.0 (GCP and CS server) and XAMPP MariaDB]

# Let's Try 1: Self Join

For each person, find the average salary of their job (assume we will display empno, ename, and average salary of the person's job)

practice\_emp

empno	ename	job	sal
7369	Smith	Clerk	1200
7499	Allen	Salesman	2000
7521	Ward	Salesman	1650
7566	Jones	Manager	3375
7654	Martin	Salesman	1650
7698	Blake	Manager	3250
7782	Clark	Manager	2850
7788	Scott	Analyst	3500
7839	King	President	6500
7844	Turner	Salesman	1900
7876	Adams	Clerk	1500
7900	James	Clerk	1350
7902	Ford	Analyst	3500
7934	Miller	Clerk	1700

```
SELECT E1.empno, E1.ename, AVG(E2.sal)
FROM   practice_emp E1, practice_emp E2
WHERE  E1.job = E2.job
GROUP BY E1.empno, E1.ename
```

## Idea:

1. Self-join practice\_emp
2. Use one copy to get each person info
3. Use one copy to compute average salary of the person's job

empno	ename	AVG(E2.sal)
7934	Miller	1437.5000
7900	James	1437.5000
7876	Adams	1437.5000
7369	Smith	1437.5000
7844	Turner	1800.0000
7654	Martin	1800.0000
7521	Ward	1800.0000
7499	Allen	1800.0000
7782	Clark	3158.3333
7698	Blake	3158.3333
7566	Jones	3158.3333
7902	Ford	3500.0000
7788	Scott	3500.0000
7839	King	6500.0000

(Note: The table shows sample data, not a complete set of data, refer to <https://www.cs.virginia.edu/~up3f/cs4750/assigns/employees.sql>)



# Let's Try 1: Subqueries in SELECT

For each person, find the average salary of their job (assume we will display empno, ename, and average salary of the person's job)

practice\_emp

empno	ename	job	sal
7369	Smith	Clerk	1200
7499	Allen	Salesman	2000
7521	Ward	Salesman	1650
7566	Jones	Manager	3375
7654	Martin	Salesman	1650
7698	Blake	Manager	3250
7782	Clark	Manager	2850
7788	Scott	Analyst	3500
7839	King	President	6500
7844	Turner	Salesman	1900
7876	Adams	Clerk	1500
7900	James	Clerk	1350
7902	Ford	Analyst	3500
7934	Miller	Clerk	1700

Step 1: Find each person's empno and ename

```
SELECT E1.empno, E1.ename, avg
FROM   practice_emp E1
```

Step 2: Given the job of the person, find the average salary of that job

```
SELECT E1.empno, E1.ename,
       (SELECT AVG(E2.sal)
        FROM   practice_emp E2
        WHERE  E1.job = E2.job)
FROM   practice_emp E1
```

Nested and correlated

(Note: The table shows sample data, not a complete set of data, refer to <https://www.cs.virginia.edu/~up3f/cs4750/assigns/employees.sql>)

# Let's Try 1: Subqueries in FROM

For each person, find the average salary of their job (assume we will display empno, ename, and average salary of the person's job)

practice\_emp

empno	ename	job	sal
7369	Smith	Clerk	1200
7499	Allen	Salesman	2000
7521	Ward	Salesman	1650
7566	Jones	Manager	3375
7654	Martin	Salesman	1650
7698	Blake	Manager	3250
7782	Clark	Manager	2850
7788	Scott	Analyst	3500
7839	King	President	6500
7844	Turner	Salesman	1900
7876	Adams	Clerk	1500
7900	James	Clerk	1350
7902	Ford	Analyst	3500
7934	Miller	Clerk	1700

Step 1: Find average salary of each job

```
SELECT job, AVG(sal) AS avg
FROM   practice_emp
GROUP BY job
```

Step 2: For each person, find the average salary of that the person's job

```
SELECT E1.empno, E1.ename, E2.avg
FROM   practice_emp AS E1,
      (SELECT job, AVG(sal) AS avg
       FROM   practice_emp
       GROUP BY job) AS E2
WHERE  E1.job = E2.job
```

(Note: The table shows sample data, not a complete set of data, refer to <https://www.cs.virginia.edu/~up3f/cs4750/assigns/employees.sql>)

# Let's Try 1: Subqueries in WITH

For each person, find the average salary of their job (assume we will display empno, name, and average salary of the person's job)

practice\_emp

empno	ename	job	sal
7369	Smith	Clerk	1200
7499	Allen	Salesman	2000
7521	Ward	Salesman	1650
7566	Jones	Manager	3375
7654	Martin	Salesman	1650
7698	Blake	Manager	3250
7782	Clark	Manager	2850
7788	Scott	Analyst	3500
7839	King	President	6500
7844	Turner	Salesman	1900
7876	Adams	Clerk	1500
7900	James	Clerk	1350
7902	Ford	Analyst	3500
7934	Miller	Clerk	1700

Step 1: Find average salary of each job

```
SELECT job, AVG(sal) AS avgSal
FROM   practice_emp
GROUP BY job
```

Step 2: For each person, find the average salary of that the person's job

```
WITH temp AS
  (SELECT job, AVG(sal) AS avgSal
   FROM practice_emp
   GROUP BY job)
SELECT E1.empno, E1.ename, T.avgSal
FROM   practice_emp AS E1, temp AS T
WHERE  E1.job = T.job
```

(Note: The table shows sample data, not a complete set of data,  
refer to <https://www.cs.virginia.edu/~up3f/cs4750/assigns/employees.sql>)

# Let's Try 2: Join (1)

For each sailor, find the number of boats they have reserved  
(assume we will display sname and the number of boats)

Sailors

sid	sname	rating	age
22	Yuppy	9	35
31	Lubber	8	55.5
44	Guppy	5	35
48	Ole Red	8	92.3
58	Rusty	10	40

Reserves

sid	bid	day
22	101	2003-06-05
22	104	2003-06-15
44	102	2003-06-05
48	105	2003-06-14
58	103	2003-06-07

```
SELECT sname, COUNT(bid)
FROM   Sailors NATURAL JOIN Reserves
GROUP BY sname
```

sname	COUNT(bid)
Yuppy	2
Guppy	1
Ole Red	1
Rusty	1

0-count case not covered

(Note: The table shows sample data, not a complete set of data,  
refer to <http://www.cs.virginia.edu/~up3f/cs4750/inclass/alldbs.sql>)

# Let's Try 2: Join (2)

For each sailor, find the number of boats they have reserved  
(assume we will display sname and the number of boats)

Sailors

sid	sname	rating	age
22	Yuppy	9	35
31	Lubber	8	55.5
44	Guppy	5	35
48	Ole Red	8	92.3
58	Rusty	10	40

Reserves

sid	bid	day
22	101	2003-06-05
22	104	2003-06-15
44	102	2003-06-05
48	105	2003-06-14
58	103	2003-06-07

Count(NULL)  
results in 0

```
SELECT sname, COUNT(bid)
FROM   Sailors
       NATURAL LEFT OUTER JOIN Reserves
GROUP BY sname
```

sname	COUNT(bid)
Yuppy	2
Lubber	0
Guppy	1
Ole Red	1
Rusty	1

(Note: The table shows sample data, not a complete set of data,  
refer to <http://www.cs.virginia.edu/~up3f/cs4750/inclass/alldbs.sql>)

# Let's Try 2: Subqueries in SELECT

For each sailor, find the number of boats they have reserved  
(assume we will display sname and the number of boats)

Sailors

sid	sname	rating	age
22	Yuppy	9	35
31	Lubber	8	55.5
44	Guppy	5	35
48	Ole Red	8	92.3
58	Rusty	10	40

Reserves

sid	bid	day
22	101	2003-06-05
22	104	2003-06-15
44	102	2003-06-05
48	105	2003-06-14
58	103	2003-06-07

Count(empty result)  
results in 0

```
SELECT sname,  
      (SELECT COUNT(bid)  
       FROM Reserves  
       WHERE Sailors.sid = Reserves.sid)  
FROM Sailors
```

Nested and  
correlated

sname	COUNT(bid)
Yuppy	2
Lubber	0
Guppy	1
Ole Red	1
Rusty	1

(Note: The table shows sample data, not a complete set of data,  
refer to <http://www.cs.virginia.edu/~up3f/cs4750/inclass/alldbs.sql>)

# Subqueries and Set Operations

## UNION

(sub-result1)

UNION

(sub-result2)

## INTERSECT

[not supported by MySQL 5.6, 5.7; work on MySQL 8.0 (GCP and CS server) and local XAMPP 10.4.11-MariaDB]

(sub-result1)

INTERSECT

(sub-result2)

## EXCEPT

[not supported by MySQL 5.6, 5.7; work on MySQL 8.0 (GCP and CS server) and local XAMPP 10.4.11-MariaDB]

(sub-result1)

EXCEPT

(sub-result2)

### Requirements:

- Same number of columns
- Same order of columns
- Same column data types

We talked about UNION and INTERSECT. Let's consider EXCEPT

# Example: Let's Solve A Problem

Use the following schema. Find IDs and names of all customers who have purchased products sold by company 7777 only. Do not list customers who have purchased from any other companies.

```
Product(pid, name, cid)
    -- cid is foreign key to Company.cid
Company(cid, cname, city)
Customer(custId, name, city)
Purchase(purchase_date, pid, custId, quantity, price)
    -- pid is foreign key to Product.pid,
    -- custId is foreign key to Customer.custId
```

Assume each customer may purchase the same product multiple times

## How should we solve this problem?

Refer to <https://www.cs.virginia.edu/~up3f/cs4750/inclass/product-purchase-for-subquery.sql>  
(rename the tables to make it easy to demo)



# Example: Let's Solve A Problem

How should we solve this problem?

- 1 (Find all customers who have purchased)  
**EXCEPT**
- 2 (Find all customers who have purchased from other companies, not 7777)

# Use EXCEPT to Solve the Problem

- 1 (Find all customers who have purchased)

**EXCEPT**

- 2 (Find all customers who have purchased from other companies, not 7777)

(sub-result1)

custld	name
991	Humpty
<del>992</del>	<del>Dumpty</del>
<del>992</del>	<del>Dumpty</del>
<del>994</del>	<del>Minnie</del>
<del>995</del>	<del>Duh</del>
<del>995</del>	<del>Duh</del>
997	Duh
<del>998</del>	<del>Duh</del>

—

(sub-result2)

custld	name
992	Dumpty
994	Minnie
995	Duh
998	Duh

=

(difference)

custld	name
991	Humpty
997	Duh

# Use EXCEPT to Solve the Problem (2)

- 1 (Find all customers who have purchased)

Find which companies the customers have purchased.  
Then, find the names of the customers

```
SELECT T1.custId, T2.cid  
FROM Purchase T1 NATURAL JOIN Product T2  
GROUP BY T1.custId, T2.cid
```

custId	cid
991	7777
992	7777
992	7778
994	7778
995	7777
995	7779
997	7777
998	7779

Got all customers who  
have purchased.  
Still need to find the  
names of the customers

# Use EXCEPT to Solve the Problem (3)

- 1 (Find all customers who have purchased)

Find which companies the customers have purchased  
Then, find the names of the customers

```
SELECT T3.custId, T3.name
FROM (SELECT T1.custId, T2.cid
      FROM Purchase T1 NATURAL JOIN Product T2
      GROUP BY T1.custId, T2.cid) T
NATURAL JOIN Customer T3
```

custId	name
991	Humpty
992	Dumpty
992	Dumpty
994	Minnie
995	Duh
995	Duh
997	Duh
998	Duh

# Use EXCEPT to Solve the Problem (4)

- 2 (Find all customers who have purchased from other companies, not 7777)

Find all customers who have purchased from other companies  
Then, find the names of the customers

```
SELECT T1.custId
FROM Purchase T1 NATURAL JOIN Product T2
WHERE T2.cid <> 7777
GROUP BY T1.custId
```

custId
992
994
995
998

Got all customers who have purchased from other companies, not 7777. Still need to find the names of the customers

# Use EXCEPT to Solve the Problem (5)

- 2 (Find all customers who have purchased from other companies, not 7777)

Find all customers who have purchased from other companies  
Then, find the names of the customers

```
SELECT T3.custId, T3.name
FROM (SELECT T1.custId
      FROM Purchase T1 NATURAL JOIN Product T2
      WHERE T2.cid <> 7777
      GROUP BY T1.custId) T
NATURAL JOIN Customer T3
```

custId	name
992	Dumpty
994	Minnie
995	Duh
998	Duh

# Use EXCEPT to Solve the Problem (6)

1

```
(SELECT T3.custId, T3.name
FROM (SELECT T1.custId, T2.cid
      FROM Purchase T1 NATURAL JOIN Product T2
      GROUP BY T1.custId, T2.cid) T
NATURAL JOIN Customer T3)
```

EXCEPT

2

```
(SELECT T3.custId, T3.name
FROM (SELECT T1.custId
      FROM Purchase T1 NATURAL JOIN Product T2
      WHERE T2.cid <> 7777
      GROUP BY T1.custId) T
NATURAL JOIN Customer T3)
```

custId	name
991	Humpty
992	Dumpty
992	Dumpty
994	Minnie
995	Duh
995	Duh
997	Duh
998	Duh

—

custId	name
992	Dumpty
994	Minnie
995	Duh
998	Duh

=

custId	name
991	Humpty
997	Duh

Refer to

<https://www.cs.virginia.edu/~up3f/cs4750/inclass/product-purchase-for-subquery.sql>

# Workaround for EXCEPT

```
SELECT result1.custId, result1.name
FROM
```

```
(SELECT T3.custId, T3.name
FROM (SELECT T1.custId, T2.cid
      FROM Purchase T1 NATURAL JOIN Product T2
      GROUP BY T1.custId, T2.cid) T
NATURAL JOIN Customer T3) result1
```

```
LEFT OUTER JOIN
```

```
(SELECT T3.custId, T3.name
FROM (SELECT T1.custId
      FROM Purchase T1 NATURAL JOIN Product T2
      WHERE T2.cid <> 7777
      GROUP BY T1.custId) T
NATURAL JOIN Customer T3) result2
```

```
ON      result1.custId = result2.custId
WHERE   result2.custId IS NULL
```

Refer to

<https://www.cs.virginia.edu/~up3f/cs4750/inclass/product-purchase-for-subquery.sql>

custId	name
991	Humpty
992	Dumpty
992	Dumpty
994	Minnie
995	Duh
995	Duh
997	Duh
998	Duh

—

custId	name
992	Dumpty
994	Minnie
995	Duh
998	Duh

=

custId	name
991	Humpty
997	Duh



# Example: UNION

```
(SELECT T3.custId, T3.name
FROM (SELECT T1.custId, T2.cid
      FROM Purchase T1 NATURAL JOIN Product T2
      GROUP BY T1.custId, T2.cid) T
NATURAL JOIN Customer T3)
```

## UNION

```
(SELECT T3.custId, T3.name
FROM (SELECT T1.custId
      FROM Purchase T1 NATURAL JOIN Product T2
      WHERE T2.cid <> 7777
      GROUP BY T1.custId) T
NATURAL JOIN Customer T3)
```

custId	name
991	Humpty
992	Dumpty
992	Dumpty
994	Minnie
995	Duh
995	Duh
997	Duh
998	Duh

U

custId	name
992	Dumpty
994	Minnie
995	Duh
998	Duh

=

custId	name
991	Humpty
992	Dumpty
992	Dumpty
994	Minnie
995	Duh
995	Duh
997	Duh
998	Duh

Refer to

<https://www.cs.virginia.edu/~up3f/cs4750/inclass/product-purchase-for-subquery.sql>

# Example: INTERSECT

```
(SELECT T3.custId, T3.name
FROM (SELECT T1.custId, T2.cid
      FROM Purchase T1 NATURAL JOIN Product T2
      GROUP BY T1.custId, T2.cid) T
NATURAL JOIN Customer T3)
```

## INTERSECT

```
(SELECT T3.custId, T3.name
FROM (SELECT T1.custId
      FROM Purchase T1 NATURAL JOIN Product T2
      WHERE T2.cid <> 7777
      GROUP BY T1.custId) T
NATURAL JOIN Customer T3)
```

custId	name
991	Humpty
992	Dumpty
992	Dumpty
994	Minnie
995	Duh
995	Duh
997	Duh
998	Duh

∩

custId	name
992	Dumpty
994	Minnie
995	Duh
998	Duh

=

custId	name
992	Dumpty
994	Minnie
995	Duh
998	Duh

Refer to

<https://www.cs.virginia.edu/~up3f/cs4750/inclass/product-purchase-for-subquery.sql>

# Workaround for INTERSECT

```
SELECT DISTINCT result1.custId, result1.name  
FROM
```

```
(SELECT T3.custId, T3.name  
FROM (SELECT T1.custId, T2.cid  
      FROM Purchase T1 NATURAL JOIN Product T2  
      GROUP BY T1.custId, T2.cid) T  
NATURAL JOIN Customer T3) result1
```

```
JOIN -- inner join
```

```
(SELECT T3.custId, T3.name  
FROM (SELECT T1.custId  
      FROM Purchase T1 NATURAL JOIN Product T2  
      WHERE T2.cid <> 7777  
      GROUP BY T1.custId) T  
NATURAL JOIN Customer T3) result2
```

```
ON result1.custId = result2.custId
```

custId	name
991	Humpty
992	Dumpty
992	Dumpty
994	Minnie
995	Duh
995	Duh
997	Duh
998	Duh

∩

custId	name
992	Dumpty
994	Minnie
995	Duh
998	Duh

=

custId	name
992	Dumpty
994	Minnie
995	Duh
998	Duh

Refer to

<https://www.cs.virginia.edu/~up3f/cs4750/inclass/product-purchase-for-subquery.sql>

# Wrap-Up

- Subqueries in SELECT, FROM
- Abstract immediate result using WITH
- Equivalent queries
- Intro to subqueries in WHERE
- Subqueries and set operations

## Note:

- Avoid nested queries – if aiming for speed
- Be careful of semantics of nested queries
  - Correlated vs. Uncorrelated

## What's next?

- Subqueries in WHERE
- Existential and universal quantifiers
- Triggers and constraints

# More Practice / Example

# Previous Solution

Find employee name (or names) who earns the highest salary for each job

practice\_emp

empno	ename	job	sal
7369	Smith	Clerk	1200
7499	Allen	Salesman	2000
7521	Ward	Salesman	1650
7566	Jones	Manager	3375
7654	Martin	Salesman	1650
7698	Blake	Manager	3250
7782	Clark	Manager	2850
7788	Scott	Analyst	3500
7839	King	President	6500
7844	Turner	Salesman	1900
7876	Adams	Clerk	1500
7900	James	Clerk	1350
7902	Ford	Analyst	3500
7934	Miller	Clerk	1700

```
SELECT E1.ename
FROM practice_emp E1
WHERE E1.sal =
    (SELECT MAX(E2.sal)
     FROM practice_emp AS E2
     WHERE E1.job = E2.job)
```

**“Correlated”**

ename
Allen
Jones
Scott
King
Ford
Miller

[more subqueries in WHERE later]

A subquery in **WHERE** returns a **single value** – to be compared to another value in a **WHERE** clause

# Alternative Ways to Solve

Find the employee name (or employees) with the highest salary for each job title

practice\_emp

empno	ename	job	sal
7369	Smith	Clerk	1200
7499	Allen	Salesman	2000
7521	Ward	Salesman	1650
7566	Jones	Manager	3375
7654	Martin	Salesman	1650
7698	Blake	Manager	3250
7782	Clark	Manager	2850
7788	Scott	Analyst	3500
7839	King	President	6500
7844	Turner	Salesman	1900
7876	Adams	Clerk	1500
7900	James	Clerk	1350
7902	Ford	Analyst	3500
7934	Miller	Clerk	1700

How should we write SQL?

(Note: The table shows sample data, not a complete set of data, refer to <https://www.cs.virginia.edu/~up3f/cs4750/assigns/employees.sql>)

# Option 1: Self-Join

Find the employee name (or employees) with the highest salary for each job title

practice\_emp

empno	ename	job	sal
7369	Smith	Clerk	1200
7499	Allen	Salesman	2000
7521	Ward	Salesman	1650
7566	Jones	Manager	3375
7654	Martin	Salesman	1650
7698	Blake	Manager	3250
7782	Clark	Manager	2850
7788	Scott	Analyst	3500
7839	King	President	6500
7844	Turner	Salesman	1900
7876	Adams	Clerk	1500
7900	James	Clerk	1350
7902	Ford	Analyst	3500
7934	Miller	Clerk	1700

```
SELECT E1.ename, E2.job, MAX(E2.sal)
FROM   practice_emp AS E1,
       practice_emp AS E2
WHERE  E1.job = E2.job
GROUP BY E2.job, E1.sal, E1.ename
HAVING E1.sal = MAX(E2.sal);
```

ename	job	sal
Ford	Analyst	3500
Scott	Analyst	3500
Miller	Clerk	1700
Jones	Manager	3375
King	President	6500
Allen	Salesman	2000



# Option 2: Subqueries

Find the employee name (or employees) with the highest salary for each job title

practice\_emp

empno	ename	job	sal
7369	Smith	Clerk	1200
7499	Allen	Salesman	2000
7521	Ward	Salesman	1650
7566	Jones	Manager	3375
7654	Martin	Salesman	1650
7698	Blake	Manager	3250
7782	Clark	Manager	2850
7788	Scott	Analyst	3500
7839	King	President	6500
7844	Turner	Salesman	1900
7876	Adams	Clerk	1500
7900	James	Clerk	1350
7902	Ford	Analyst	3500
7934	Miller	Clerk	1700

Find max salary for each job



Then find the employee(s) with that max salary

# Option 2: Subqueries

practice\_emp

empno	ename	job	sal
7369	Smith	Clerk	1200
7499	Allen	Salesman	2000
7521	Ward	Salesman	1650
7566	Jones	Manager	3375
7654	Martin	Salesman	1650
7698	Blake	Manager	3250
7782	Clark	Manager	2850
7788	Scott	Analyst	3500
7839	King	President	6500
7844	Turner	Salesman	1900
7876	Adams	Clerk	1500
7900	James	Clerk	1350
7902	Ford	Analyst	3500
7934	Miller	Clerk	1700

Find max salary for each job

```
SELECT E1.job, MAX(E1.sal) AS maxSal
FROM   practice_emp E1
GROUP BY E1.job;
```

job	maxSal
Analyst	3500
Clerk	1700
Manager	3375
President	6500
Salesman	2000

# Option 2: Subqueries

practice\_emp

empno	ename	job	sal
7369	Smith	Clerk	1200
7499	Allen	Salesman	2000
7521	Ward	Salesman	1650
7566	Jones	Manager	3375
7654	Martin	Salesman	1650
7698	Blake	Manager	3250
7782	Clark	Manager	2850
7788	Scott	Analyst	3500
7839	King	President	6500
7844	Turner	Salesman	1900
7876	Adams	Clerk	1500
7900	James	Clerk	1350
7902	Ford	Analyst	3500
7934	Miller	Clerk	1700

How should we write SQL for this?

JOIN  
(on matching  
attributes)

ename	job	sal
Ford	Analyst	3500
Scott	Analyst	3500
Miller	Clerk	1700
Jones	Manager	3375
King	President	6500
Allen	Salesman	2000

job	maxSal
Analyst	3500
Clerk	1700
Manager	3375
President	6500
Salesman	2000

```
SELECT E1.job, MAX(E1.sal) AS maxSal
FROM   practice_emp E1
GROUP BY E1.job;
```

# Option 2: Subqueries

practice\_emp

empno	ename	job	sal
7369	Smith	Clerk	1200
7499	Allen	Salesman	2000
7521	Ward	Salesman	1650
7566	Jones	Manager	3375
7654	Martin	Salesman	1650
7698	Blake	Manager	3250
7782	Clark	Manager	2850
7788	Scott	Analyst	3500
7839	King	President	6500
7844	Turner	Salesman	1900
7876	Adams	Clerk	1500
7900	James	Clerk	1350
7902	Ford	Analyst	3500
7934	Miller	Clerk	1700

```
SELECT *  
FROM practice_emp
```

```
SELECT *  
FROM 1, 2  
WHERE 1.job = 2.job;
```

1

JOIN  
(on matching  
attributes)

2

job	maxSal
Analyst	3500
Clerk	1700
Manager	3375
President	6500
Salesman	2000

```
SELECT E2.job, MAX(E2.sal) AS maxSal  
FROM practice_emp E2  
GROUP BY E2.job;
```

# Option 2: Subqueries

practice\_emp

empno	ename	job	sal
7369	Smith	Clerk	1200
7499	Allen	Salesman	2000
7521	Ward	Salesman	1650
7566	Jones	Manager	3375
7654	Martin	Salesman	1650
7698	Blake	Manager	3250
7782	Clark	Manager	2850
7788	Scott	Analyst	3500
7839	King	President	6500
7844	Turner	Salesman	1900
7876	Adams	Clerk	1500
7900	James	Clerk	1350
7902	Ford	Analyst	3500
7934	Miller	Clerk	1700

```

SELECT *
FROM practice_emp E1,
     (SELECT E2.job, MAX(E2.sal) AS maxSal
      FROM practice_emp E2
      GROUP BY E2.job) T1
WHERE E1.job = T1.job
    
```

1

JOIN  
(on matching  
attributes)

2

job	maxSal
Analyst	3500
Clerk	1700
Manager	3375
President	6500
Salesman	2000

empno	ename	job	mgr	hiredate	sal	comm	deptno	job	maxSal
7369	Smith	Clerk	7902	2002-12-17	1200	0	20	Clerk	1700
7499	Allen	Salesman	7698	2003-02-20	2000	500	30	Salesman	2000
7521	Ward	Salesman	7698	2003-02-22	1650	800	30	Salesman	2000
7566	Jones	Manager	7839	2003-04-02	3375	0	20	Manager	3375
7654	Martin	Salesman	7698	2003-09-28	1650	1400	30	Salesman	2000
7698	Blake	Manager	7839	2003-05-01	3250	0	30	Manager	3375
7782	Clark	Manager	7839	2003-06-09	2850	0	10	Manager	3375
7788	Scott	Analyst	7566	2002-06-27	3500	0	20	Analyst	3500
7839	King	President	NULL	2003-11-17	6500	0	10	President	6500
7844	Turner	Salesman	7698	2003-09-08	1900	0	30	Salesman	2000
7876	Adams	Clerk	7788	2002-07-31	1500	0	20	Clerk	1700
7900	James	Clerk	7698	2003-12-03	1350	0	30	Clerk	1700
7902	Ford	Analyst	7566	2003-12-03	3500	0	20	Analyst	3500
7934	Miller	Clerk	7782	2003-01-23	1700	0	10	Clerk	1700

# Option 2: Subqueries

practice\_emp

empno	ename	job	sal
7369	Smith	Clerk	1200
7499	Allen	Salesman	2000
7521	Ward	Salesman	1650
7566	Jones	Manager	3375
7654	Martin	Salesman	1650
7698	Blake	Manager	3250
7782	Clark	Manager	2850
7788	Scott	Analyst	3500
7839	King	President	6500
7844	Turner	Salesman	1900
7876	Adams	Clerk	1500
7900	James	Clerk	1350
7902	Ford	Analyst	3500
7934	Miller	Clerk	1700

1

JOIN  
(on matching  
attributes)

2

job	maxSal
Analyst	3500
Clerk	1700
Manager	3375
President	6500
Salesman	2000

```
SELECT E1.ename, E1.job, E1.sal, T1.maxSal
FROM practice_emp E1,
     (SELECT E2.job, MAX(E2.sal) AS maxSal
      FROM practice_emp E2
      GROUP BY E2.job) T1
WHERE E1.job = T1.job
```

3

ename	job	sal	maxSal
Smith	Clerk	1200	1700
Allen	Salesman	2000	2000
Ward	Salesman	1650	2000
Jones	Manager	3375	3375
Martin	Salesman	1650	2000
Blake	Manager	3250	3375
Clark	Manager	2850	3375
Scott	Analyst	3500	3500
King	President	6500	6500
Turner	Salesman	1900	2000
Adams	Clerk	1500	1700
James	Clerk	1350	1700
Ford	Analyst	3500	3500
Miller	Clerk	1700	1700



# Option 2: Subqueries

practice\_emp

empno	ename	job	sal
7369	Smith	Clerk	1200
7499	Allen	Salesman	2000
7521	Ward	Salesman	1650
7566	Jones	Manager	3375
7654	Martin	Salesman	1650
7698	Blake	Manager	3250
7782	Clark	Manager	2850
7788	Scott	Analyst	3500
7839	King	President	6500
7844	Turner	Salesman	1900
7876	Adams	Clerk	1500
7900	James	Clerk	1350
7902	Ford	Analyst	3500
7934	Miller	Clerk	1700

1

JOIN  
(on matching  
attributes)

2

job	maxSal
Analyst	3500
Clerk	1700
Manager	3375
President	6500
Salesman	2000

```
SELECT T2.ename, T2.job, T2.sal
FROM T2
WHERE T2.sal = T1.maxSal
```

ename	job	sal	maxSal
Smith	Clerk	1200	1700
Allen	Salesman	2000	2000
Ward	Salesman	1650	2000
Jones	Manager	3375	3375

```
SELECT E2.ename, E2.job, E2.sal, T1.maxSal
FROM practice_emp E2,
     (SELECT E1.job, MAX(E1.sal) AS maxSal
      FROM practice_emp E1
      GROUP BY E1.job) T1
WHERE E2.job = T1.job
```

3

ename	job	sal	maxSal
Ford	Analyst	3500	3500
Miller	Clerk	1700	1700

# Option 2: Subqueries

practice\_emp

empno	ename	job	sal
7369	Smith	Clerk	1200
7499	Allen	Salesman	2000
7521	Ward	Salesman	1650
7566	Jones	Manager	3375
7654	Martin	Salesman	1650
7698	Blake	Manager	3250
7782	Clark	Manager	2850
7788	Scott	Analyst	3500
7839	King	President	6500
7844	Turner	Salesman	1900
7876	Adams	Clerk	1500
7900	James	Clerk	1350
7902	Ford	Analyst	3500
7934	Miller	Clerk	1700

```
SELECT T2.ename, T2.job, T2.sal
```

```
FROM (SELECT E2.ename, E2.job,
             E2.sal, T1.maxSal
      FROM practice_emp E2,
           (SELECT E1.job, MAX(E1.sal) AS maxSal
            FROM practice_emp E1
            GROUP BY E1.job) T1
      WHERE E2.job = T1.job) T2
```

```
WHERE T2.sal = T2.maxSal
```

1

JOIN  
(on matching  
attributes)

2

job	maxSal
Analyst	3500
Clerk	1700
Manager	3375
President	6500
Salesman	2000

ename	job	sal
Allen	Salesman	2000
Jones	Manager	3375
Scott	Analyst	3500
King	President	6500
Ford	Analyst	3500
Miller	Clerk	1700



# Option 3: WITH Clause

Find the employee name (or employees) with the highest salary for each job title

practice\_emp

empno	ename	job	sal
7369	Smith	Clerk	1200
7499	Allen	Salesman	2000
7521	Ward	Salesman	1650
7566	Jones	Manager	3375
7654	Martin	Salesman	1650
7698	Blake	Manager	3250
7782	Clark	Manager	2850
7788	Scott	Analyst	3500
7839	King	President	6500
7844	Turner	Salesman	1900
7876	Adams	Clerk	1500
7900	James	Clerk	1350
7902	Ford	Analyst	3500
7934	Miller	Clerk	1700

Find max salary for each job



Then find the employee(s) with that max salary

# Option 3: WITH Clause

practice\_emp

empno	ename	job	sal
7369	Smith	Clerk	1200
7499	Allen	Salesman	2000
7521	Ward	Salesman	1650
7566	Jones	Manager	3375
7654	Martin	Salesman	1650
7698	Blake	Manager	3250
7782	Clark	Manager	2850
7788	Scott	Analyst	3500
7839	King	President	6500
7844	Turner	Salesman	1900
7876	Adams	Clerk	1500
7900	James	Clerk	1350
7902	Ford	Analyst	3500
7934	Miller	Clerk	1700

Abstract immediate result  
using WITH clause

**WITH** MaxSal AS

```
(SELECT job, MAX(sal) AS salary
FROM   practice_emp
GROUP   BY job)
```

```
SELECT E1.ename, E1.job, E1.sal
FROM   practice_emp AS E1, MaxSal AS A
WHERE   E1.job = A.job
        AND E1.sal = A.salary
```

1

JOIN  
(on matching  
attributes)

2

job	maxSal
Analyst	3500
Clerk	1700
Manager	3375
President	6500
Salesman	2000

ename	job	sal
Allen	Salesman	2000
Jones	Manager	3375
Scott	Analyst	3500
King	President	6500
Ford	Analyst	3500
Miller	Clerk	1700