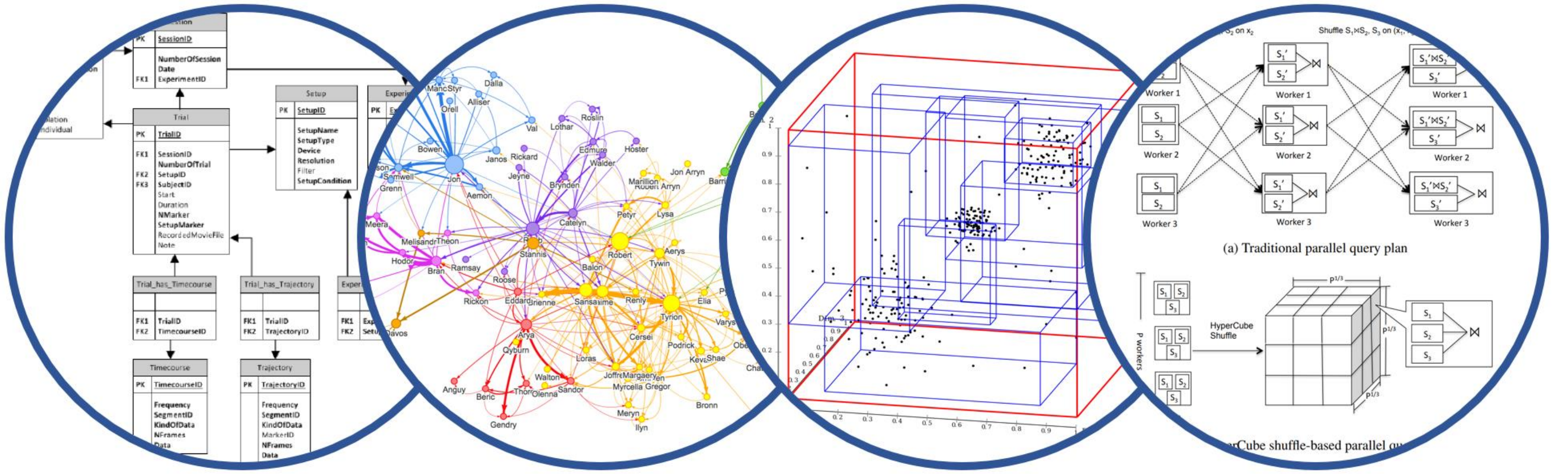


CSE 344: Intro to Data Management

SQL Subqueries

Paul G. Allen School of Computer Science and Engineering
University of Washington, Seattle

Announcements (1/2)

- HW2 due tonight
- HW3 is posted
  - **Accept the invite from Azure! It expires soon!**
  - Instructions for HW3 included
  - Sections on Thursday will walk you through the setup

Announcements (2/2)

No in-person lectures Monday&Wednesday next week!

- Lectures will be recorded: canvas→zoom
- Please watch the lectures

Subqueries in FROM

Subqueries in FROM

What is the average salary of car drivers?

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

Subqueries in FROM

What is the average salary of car drivers?

```
WITH Cardrivers AS
   (SELECT DISTINCT P.*
    FROM Payroll P, Regist R
    WHERE P.UserId=R.UserID)
SELECT avg(Salary)
FROM Cardrivers;
```

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

Subqueries in FROM

What is the average salary of car drivers?

```
WITH Cardrivers AS
   (SELECT DISTINCT P.*
    FROM Payroll P, Regist R
    WHERE P.UserId=R.UserID)
SELECT avg(Salary)
FROM Cardrivers;
```

Side note:
This is called a
semi-join

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

Subqueries in FROM

What is the average salary of car drivers?

```
WITH Cardrivers AS
   (SELECT DISTINCT P.*
    FROM Payroll P, Regist R
    WHERE P.UserId=R.UserID)
SELECT avg(Salary)
FROM Cardrivers;
```

Side note:
This is called a
semi-join

A semi-join is a join of two relations,
followed by a projection on the attributes of the first relation

Subqueries in FROM

What is the average salary of car drivers?

```
WITH Cardrivers AS
   (SELECT DISTINCT P.*
    FROM Payroll P, Regist R
    WHERE P.UserId=R.UserID)
SELECT avg(Salary)
FROM Cardrivers;
```

```
SELECT avg(C.Salary)
FROM (SELECT DISTINCT P.*
      FROM Payroll P, Regist R
      WHERE P.UserId=R.UserID)as C;
```

Subqueries in FROM

What is the average salary of car drivers?

```
WITH Cardrivers AS
   (SELECT DISTINCT P.*
    FROM Payroll P, Regist R
    WHERE P.UserId=R.UserID)
SELECT avg(Salary)
FROM Cardrivers;
```

```
SELECT avg(C.Salary)
FROM  (SELECT DISTINCT P.*
       FROM Payroll P, Regist R
       WHERE P.UserId=R.UserID) as C;
```

Subquery in the FROM clause

Subqueries in FROM

What is the average salary of car drivers?

```
WITH Cardrivers AS
   (SELECT DISTINCT P.*
    FROM Payroll P, Regist R
    WHERE P.UserId=R.UserID)
SELECT avg(Salary)
FROM Cardrivers;
```

```
SELECT avg(C.Salary)
FROM (SELECT DISTINCT P.*
      FROM Payroll P, Regist R
      WHERE P.UserId=R.UserID)as C;
```

Must have an alias

Subquery in the FROM clause

Discussion

- Subquery in FROM is the same as one in WITH
- Sometimes WITH makes the query easier to read
- Some DBMS may not support one or the other

Subqueries in SELECT

Subqueries in SELECT

We can use subqueries in SELECT, but caveat:

- A subquery returns a set…
- …while in SELECT we must list single values!
- Must ensure that our query returns a single value

Subqueries in SELECT

For each user, find the average salary of their job type

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Subqueries in SELECT

For each user, find the average salary of their job type

We want this

Name	Salary	Avg
Jack	50000	55000
Allison	60000	55000
Magda	90000	95000
Dan	100000	95000

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Subqueries in SELECT

For each user, find the average salary of their job type

```
SELECT P.Name, (SELECT AVG(P1.Salary)
                  FROM Payroll AS P1
                 WHERE P.Job = P1.Job)
  FROM Payroll AS P;
```

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Subqueries in SELECT

For each user, find the average salary of their job type

```
SELECT P.Name, (SELECT AVG(P1.Salary)
                  FROM Payroll AS P1
                 WHERE P.Job = P1.Job)
  FROM Payroll AS P;
```

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Semantics:
Nested for loops!

Subqueries in SELECT

```
SELECT P.Name, (SELECT AVG(P1.Salary)
                  FROM Payroll AS P1
                 WHERE P.Job = P1.Job)
  FROM Payroll AS P;
```

Payroll P

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

A single FOR loop: Payroll P

Subqueries in SELECT

```
SELECT P.Name,  (SELECT AVG(P1.Salary)
                   FROM Payroll AS P1
                  WHERE P.Job = P1.Job)
  FROM Payroll AS P;
```

Payroll P

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

A single FOR loop: Payroll P

For each P, compute a subquery

Payroll P1

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Subqueries in SELECT

```
SELECT P.Name,  (SELECT AVG(P1.Salary)
                   FROM Payroll AS P1
                  WHERE P.Job = P1.Job)
  FROM Payroll AS P;
```

Payroll P

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

A single FOR loop: Payroll P

For each P, compute a subquery

Payroll P1

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Subqueries in SELECT

```
SELECT P.Name, (SELECT AVG(P1.Salary)
                FROM Payroll AS P1
                WHERE P.Job = P1.Job)
  FROM Payroll AS P;
```

Payroll P

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

55000

For each P, compute a subquery

Payroll P1

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Subqueries in SELECT

```
SELECT P.Name, (SELECT AVG(P1.Salary)
                  FROM Payroll AS P1
                 WHERE P.Job = P1.Job)
  FROM Payroll AS P;
```

Payroll P

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

55000

Subqueries in SELECT

```
SELECT P.Name, (SELECT AVG(P1.Salary)
                  FROM Payroll AS P1
                 WHERE P.Job = P1.Job)
  FROM Payroll AS P;
```

Payroll P

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

55000

Payroll P1

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Subqueries in SELECT

```
SELECT P.Name, (SELECT AVG(P1.Salary)
                FROM Payroll AS P1
                WHERE P.Job = P1.Job)
  FROM Payroll AS P;
```

Payroll P

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

55000

Payroll P1

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Subqueries in SELECT

```
SELECT P.Name, (SELECT AVG(P1.Salary)
                  FROM Payroll AS P1
                 WHERE P.Job = P1.Job)
  FROM Payroll AS P;
```

Payroll P

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

55000

55000

Payroll P1

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Subqueries in SELECT

```
SELECT P.Name, (SELECT AVG(P1.Salary)
                  FROM Payroll AS P1
                 WHERE P.Job = P1.Job)
  FROM Payroll AS P;
```

Payroll P

UserID	Name	Job	Salary	
123	Jack	TA	50000	55000
345	Allison	TA	60000	55000
567	Magda	Prof	90000	95000
789	Dan	Prof	100000	

Payroll P1

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Subqueries in SELECT

```
SELECT P.Name, (SELECT AVG(P1.Salary)
                  FROM Payroll AS P1
                 WHERE P.Job = P1.Job)
  FROM Payroll AS P;
```

Payroll P

UserID	Name	Job	Salary	
123	Jack	TA	50000	55000
345	Allison	TA	60000	55000
567	Magda	Prof	90000	95000
789	Dan	Prof	100000	95000

Payroll P1

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Subqueries in SELECT

For each person find the average salary of their job

```
SELECT P.Name,  (SELECT AVG(P1.Salary)
                   FROM Payroll AS P1
                  WHERE P.Job = P1.Job)
  FROM Payroll AS P;
```

Same query, unnested

```
SELECT P1.Name, AVG(P2.Salary)
  FROM Payroll AS P1, Payroll AS P2
 WHERE P1.Job = P2.Job
 GROUP BY P1.UserID, P1.Name;
```

Subqueries in SELECT

- A subquery in SELECT can be unnested
- Careful: sometimes it requires left outer joins

Subqueries in SELECT

For each person find the number of cars they drive

Subqueries in SELECT

For each person find the number of cars they drive

```
SELECT P.Name, (SELECT COUNT(R.Car)
                FROM Regist AS R
               WHERE P.UserID =
                     R.UserID)
  FROM Payroll AS P;
```

Subqueries in SELECT

For each person find the number of cars they drive

```
SELECT P.Name,  (SELECT COUNT(R.Car)
                   FROM Regist AS R
                  WHERE P.UserID =
                        R.UserID)
  FROM Payroll AS P;
```

⇕

```
SELECT P.Name, COUNT(R.Car)
  FROM Payroll AS P, Regist AS R
 WHERE P.UserID = R.UserID
 GROUP BY P.UserID, P.Name;
```

Subqueries in SELECT

For each person find the number of cars they drive

```
SELECT P.Name,  (SELECT COUNT(R.Car)
                    FROM Regist AS R
                   WHERE P.UserID =
                         R.UserID)
  FROM Payroll AS P;
```

Not the same!
Why?

```
SELECT P.Name, COUNT(R.Car)
  FROM Payroll AS P, Regist AS R
 WHERE P.UserID = R.UserID
 GROUP BY P.UserID, P.Name;
```

Subqueries in SELECT

For each person find the number of cars they drive

```
SELECT P.Name,  (SELECT COUNT(R.Car)
                   FROM Regist AS R
                  WHERE P.UserID =
                        R.UserID)
  FROM Payroll AS P;
```

0-count case not covered!

```
SELECT P.Name, COUNT(R.Car)
  FROM Payroll AS P, Regist AS R
 WHERE P.UserID = R.UserID
 GROUP BY P.UserID, P.Name;
```

Subqueries in SELECT

For each person find the number of cars they drive

```
SELECT P.Name, (SELECT COUNT(R.Car)
                  FROM Regist AS R
                 WHERE P.UserID =
                       R.UserID)
  FROM Payroll AS P;
```

Name	Count
Jack	1
Allison	0
Magda	2
Dan	0

0-count case not covered!

```
SELECT P.Name, COUNT(R.Car)
  FROM Payroll AS P, Regist AS R
 WHERE P.UserID = R.UserID
 GROUP BY P.UserID, P.Name;
```

Name	Count
Jack	1
Magda	2

Subqueries in SELECT

For each person find the number of cars they drive

```
SELECT P.Name, (SELECT COUNT(R.Car)
                  FROM Regist AS R
                 WHERE P.UserID =
                       R.UserID)
  FROM Payroll AS P;
```

Still possible to unnest

Subqueries in SELECT

For each person find the number of cars they drive

```
SELECT P.Name,  (SELECT COUNT(R.Car)
                   FROM Regist AS R
                  WHERE P.UserID =
                        R.UserID)
  FROM Payroll AS P;
```

Still possible to unnest

```
SELECT P.Name, COUNT(R.Car)
  FROM Payroll AS P LEFT OUTER JOIN
       Regist AS R ON P.UserID = R.UserID
 GROUP BY P.UserID, P.Name;
```

Subqueries in SELECT

- Lesson:
  - Unnesting queries may require left outer join
- Another issue:
  - Subqueries in SELECT must return a single value
  - Otherwise, they produce an error (except Sqlite…)

Subqueries in SELECT

For each person list the cars that they drive

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

Subqueries in SELECT

For each person list the cars that they drive

Intended answer

Name	Car
Jack	Charger
Magda	Civic
Magda	Pinto

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

Subqueries in SELECT

For each person list the cars that they drive

```
SELECT P.Name,(SELECT R.car
               FROM Regist R
               WHERE P.UserID=R.UserID)
FROM Payroll P;
```

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

Subqueries in SELECT

For each person list the cars that they drive

```
SELECT P.Name,(SELECT R.car
               FROM Regist R
               WHERE P.UserID=R.UserID)
FROM Payroll P;
```

WRONG! Why?

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

Subqueries in SELECT

For each person list the cars that they drive

```
SELECT P.Name, (SELECT R.car
                FROM Regist R
                WHERE P.UserID=R.UserID)
FROM Payroll P;
```

Is not always a single value

Name	Car
Jack	Charger
Allison	…
Magda	Civic Pinto
Dan	…

WRONG! Why?

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

Subqueries in SELECT

For each person list the cars that they drive

```
SELECT P.Name, (SELECT R.car
                FROM Regist R
                WHERE P.UserID=R.UserID)
FROM Payroll P;
```

Is not always a single value

WRONG! Why?

Sqlite returns junk.
Better systems give an error

Name	Car
Jack	Charger
Allison	…
Magda	Civic Pinto
Dan	…

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

Subqueries in SELECT

For each person list the cars that they drive

```
SELECT P.Name,(SELECT R.car
               FROM Regist R
               WHERE P.UserID=R.UserID)
FROM Payroll P;
```

WRONG! Why?

```
SELECT P.Name, R.car
FROM Payroll P, Regist R
WHERE P.UserID=R.UserID;
```

The only right way to write this query

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Subqueries in SELECT

Final wrinkle:

- A query with a subquery in SELECT may introduce unwanted duplicates
- Need DISTINCT

Subqueries in SELECT

Compute the average salary for each job

Want this output:

Job	avg(…)
TA	55000
Prof	95000

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Subqueries in SELECT

Compute the average salary for each job

```
SELECT P.Job, (SELECT avg(P1.Salary)
                 FROM Payroll AS P1
                WHERE P.Job = P1.Job)
  FROM Payroll AS P;
```

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Subqueries in SELECT

Compute the average salary for each job

```
SELECT P.Job, (SELECT avg(P1.Salary)
                 FROM Payroll AS P1
                WHERE P.Job = P1.Job)
  FROM Payroll AS P;
```

How many records are in the output?

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Subqueries in SELECT

Compute the average salary for each job

```
SELECT P.Job, (SELECT avg(P1.Salary)
                    FROM Payroll AS P1
                   WHERE P.Job = P1.Job)
  FROM Payroll AS P;
```

How many records are in the output?

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Job	avg(…)
TA	55000
TA	55000
Prof	95000
Prof	95000

Subqueries in SELECT

Compute the average salary for each job

```
SELECT DISTINCT P.Job,
      (SELECT avg(P1.Salary)
       FROM Payroll AS P1
       WHERE P.Job = P1.Job)
FROM Payroll AS P;
```

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

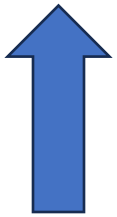
Job	avg(…)
TA	55000
TA	55000
Prof	95000
Prof	95000

Subqueries in SELECT

Compute the average salary for each job

```
SELECT DISTINCT P.Job,
     (SELECT avg(P1.Salary)
      FROM Payroll AS P1
      WHERE P.Job = P1.Job)
FROM Payroll AS P;
```

Job	avg(…)
TA	55000
Prof	95000



Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

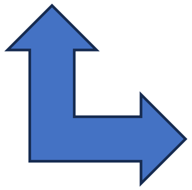
Job	avg(…)
TA	55000
TA	55000
Prof	95000
Prof	95000

Subqueries in SELECT

Compute the average salary for each job

```
SELECT DISTINCT P.Job,
     (SELECT avg(P1.Salary)
      FROM Payroll AS P1
      WHERE P.Job = P1.Job)
FROM Payroll AS P;
```

Under the hood: GROUP BY replaces two loops with one loop over some hash table



```
SELECT P.Job, avg(P.Salary)
FROM Payroll AS P
GROUP BY P.Job;
```

Discussion

- Queries in SELECT must return single value
- Think about edge cases: zero matches, null values
- Best: avoid nested queries when possible
- Appreciate the utility of GROUP BY

Subqueries in WHERE/HAVING

Subqueries in WHERE/HAVING

- Can use subquery that returns single value; same as in SELECT

- Additional predicates:
  - EXISTS / NOT EXISTS
  - IN / NOT IN
  - ANY / ALL

Subqueries in WHERE/HAVING

Find all employees who earn less than the average of their job

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Subqueries in WHERE/HAVING

Find all employees who earn less than the average of their job

```
SELECT P.Name, P.salary
FROM Payroll P
WHERE P.Salary < (SELECT avg(P1.salary)
                  FROM Payroll P1
                  WHERE P.Job = P1.Job);
```

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Subqueries in WHERE/HAVING

Find all employees who earn less than the average of their job

```
SELECT P.Name, P.salary
FROM Payroll P
WHERE P.Salary < (SELECT avg(P1.salary)
                  FROM Payroll P1
                  WHERE P.Job = P1.Job);
```

We can unnest using HAVING

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Subqueries in WHERE/HAVING

Find all employees who earn less than the average of their job

```
SELECT P.Name, P.salary
FROM Payroll P
WHERE P.Salary < (SELECT avg(P1.salary)
                  FROM Payroll P1
                  WHERE P.Job = P1.Job);
```

We can unnest using HAVING

```
SELECT P.Name, P.salary
FROM Payroll P, Payroll P1
WHERE P.Job = P1.Job
GROUP BY P.Name, P.salary
HAVING P.Salary < avg(P1.salary);
```

Subqueries in WHERE/HAVING

SQL has a few predicates that apply to a subquery:

- EXISTS (SELECT ….) checks if it is not empty
  NOT EXISTS (SELECT …) checks if it is empty

Subqueries in WHERE/HAVING

SQL has a few predicates that apply to a subquery:

- EXISTS (SELECT ….) checks if it is not empty
  NOT EXISTS (SELECT …) checks if it is empty
- X in (SELECT Y FROM …) checks output has X
  X not in (SELECT Y …) checks if it doesn't have X

Subqueries in WHERE/HAVING

SQL has a few predicates that apply to a subquery:

- EXISTS (SELECT ….) checks if it is not empty
  NOT EXISTS (SELECT …) checks if it is empty

- X in (SELECT Y FROM …) checks output has X
  X not in (SELECT Y …) checks if it doesn’t have X

- X > ALL(SELECT …)
  X > ANY(SELECT …)
  checks if X is > than one or all values in output

Subqueries in WHERE/HAVING

SQL has a few predicates that apply to a subquery:

Next

- EXISTS (SELECT ….) checks if it is not empty
  NOT EXISTS (SELECT …) checks if it is empty

- X in (SELECT Y FROM …) checks output has X
  X not in (SELECT Y …) checks if it doesn't have X

Next lecture

- X > ALL(SELECT …)
  X > ANY(SELECT …)
  checks if X is > than one or all values in output

Subqueries in WHERE/HAVING

Find people who **do** drive cars

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

Subqueries in WHERE/HAVING

Find people who **do** drive cars

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE exists
      (SELECT *
       FROM Regist R
       WHERE P.UserID = R.UserID);
```

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

Subqueries in WHERE/HAVING

Find people who **do** drive cars

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE exists
       (SELECT *
        FROM Regist R
        WHERE P.UserID = R.UserID);
```

Same as a semi-join

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

UserID	Name
123	Jack
567	Magda

Subqueries in WHERE/HAVING

Find people who **do not** drive cars

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE exists
        (SELECT *
         FROM Regist R
         WHERE P.UserID = R.UserID);
```

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

Subqueries in WHERE/HAVING

Find people who **do not** drive cars

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE not exists
        (SELECT *
         FROM Regist R
         WHERE P.UserID = R.UserID);
```

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

Subqueries in WHERE/HAVING

Find people who **do not** drive cars

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE not exists
       (SELECT *
        FROM Regist R
        WHERE P.UserID = R.UserID);
```

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

UserID	Name
345	Allison
789	Dan

Subqueries in WHERE/HAVING

Find people who **do not** drive cars

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE not exists
      (SELECT *
       FROM Regist R
       WHERE P.UserID = R.UserID);
```

Called an anti-semijoin

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

UserID	Name
345	Allison
789	Dan

Nested Loop Semantics

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE not exists
      (SELECT *
       FROM Regist R
       WHERE P.UserID = R.UserID);
```

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

Nested Loop Semantics

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE not exists
       (SELECT *
        FROM Regist R
        WHERE P.UserID = R.UserID);
```

Payroll

P →

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

Output so far

UserID	Name

Nested Loop Semantics

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE not exists
       (SELECT *
        FROM Regist R
        WHERE P.UserID = R.UserID);
```

Output so far

UserID	Name

Payroll

P →

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

Compute subquery
for P.UserID=123

UserID	Car

Nested Loop Semantics

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE not exists
       (SELECT *
        FROM Regist R
        WHERE P.UserID = R.UserID);
```

Output so far

UserID	Name

Payroll

P

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

R

UserID	Car
123	Charger
567	Civic
567	Pinto

Compute subquery for P.UserID=123

UserID	Car
123	Charger

Nested Loop Semantics

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE not exists
       (SELECT *
        FROM Regist R
        WHERE P.UserID = R.UserID);
```

Output so far

UserID	Name

Payroll

	UserID	Name	Job	Salary
P →	123	Jack	TA	50000
	345	Allison	TA	60000
	567	Magda	Prof	90000
	789	Dan	Prof	100000

Regist

	UserID	Car
	123	Charger
R →	567	Civic
	567	Pinto

Compute subquery for P.UserID=123

UserID	Car
123	Charger

Nested Loop Semantics

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE not exists
       (SELECT *
        FROM Regist R
        WHERE P.UserID = R.UserID);
```

Output so far

UserID	Name

Payroll

P →

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

R →

UserID	Car
123	Charger
567	Civic
567	Pinto

Compute subquery for P.UserID=123

UserID	Car
123	Charger

Nested Loop Semantics

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE not exists
       (SELECT *
        FROM Regist R
        WHERE P.UserID = R.UserID);
```

Payroll

P →

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Output so far

UserID	Name

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

Compute subquery
for P.UserID=123

UserID	Car
123	Charger

Done UserID=123
Exists answers

Nested Loop Semantics

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE not exists
       (SELECT *
        FROM Regist R
        WHERE P.UserID = R.UserID);
```

Payroll

P →

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Output so far

UserID	Name

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

Compute subquery for P.UserID=123

Skip UserID=123

UserID	Car
123	Charger

Done UserID=123
Exists answers

Nested Loop Semantics

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE not exists
       (SELECT *
        FROM Regist R
        WHERE P.UserID = R.UserID);
```

Output so far

UserID	Name

Payroll

P →

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

Nested Loop Semantics

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE not exists
       (SELECT *
        FROM Regist R
        WHERE P.UserID = R.UserID);
```

Output so far

UserID	Name

Payroll

P →

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

Nested Loop Semantics

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE not exists
       (SELECT *
        FROM Regist R
        WHERE P.UserID = R.UserID);
```

Output so far

UserID	Name

Payroll

P →

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

Compute subquery for P.UserID=345

UserID	Car

Nested Loop Semantics

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE not exists
        (SELECT *
         FROM Regist R
         WHERE P.UserID = R.UserID);
```

Output so far

UserID	Name

Payroll

P →

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

R →

UserID	Car
123	Charger
567	Civic
567	Pinto

Compute subquery for P.UserID=345

UserID	Car

Nested Loop Semantics

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE not exists
       (SELECT *
        FROM Regist R
        WHERE P.UserID = R.UserID);
```

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

P →

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

R →

Compute subquery for P.UserID=345

UserID	Car

Output so far

UserID	Name

Nested Loop Semantics

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE not exists
       (SELECT *
        FROM Regist R
        WHERE P.UserID = R.UserID);
```

Payroll

	UserID	Name	Job	Salary
	123	Jack	TA	50000
P →	345	Allison	TA	60000
	567	Magda	Prof	90000
	789	Dan	Prof	100000

Regist

	UserID	Car
	123	Charger
	567	Civic
R →	567	Pinto

Output so far

UserID	Name

Compute subquery for P.UserID=345

UserID	Car

Nested Loop Semantics

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE not exists
       (SELECT *
        FROM Regist R
        WHERE P.UserID = R.UserID);
```

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

P →

Output so far

UserID	Name

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

R →

Compute subquery for P.UserID=345

UserID	Car

Done UserID=345
Not exists answers

Nested Loop Semantics

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE not exists
       (SELECT *
        FROM Regist R
        WHERE P.UserID = R.UserID);
```

Payroll

P →

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Output so far

UserID	Name
345	Allison

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

Compute subquery for P.UserID=345

Output UserID=345

UserID	Car

Done UserID=345
Not exists answers

Nested Loop Semantics

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE not exists
       (SELECT *
        FROM Regist R
        WHERE P.UserID = R.UserID);
```

Output so far

UserID	Name
345	Allison

Payroll

P →

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

Nested Loop Semantics

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE not exists
       (SELECT *
        FROM Regist R
        WHERE P.UserID = R.UserID);
```

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

P

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

Output so far

UserID	Name
345	Allison

Nested Loop Semantics

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE not exists
       (SELECT *
        FROM Regist R
        WHERE P.UserID = R.UserID);
```

Output so far

UserID	Name
345	Allison

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

P →

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

R →

Compute subquery for P.UserID=567

UserID	Car

Nested Loop Semantics

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE not exists
       (SELECT *
        FROM Regist R
        WHERE P.UserID = R.UserID);
```

Output so far

UserID	Name
345	Allison

Payroll

P →

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

R →

UserID	Car
123	Charger
567	Civic
567	Pinto

Compute subquery for P.UserID=567

UserID	Car
567	Civic

Nested Loop Semantics

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE not exists
       (SELECT *
        FROM Regist R
        WHERE P.UserID = R.UserID);
```

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

P →

Output so far

UserID	Name
345	Allison

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

Compute subquery for P.UserID=567

Skip UserID=567

UserID	Car
567	Civic
567	Pinto

Done UserID=567 Exists answers

Nested Loop Semantics

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE not exists
       (SELECT *
        FROM Regist R
        WHERE P.UserID = R.UserID);
```

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

P

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

Output so far

UserID	Name
345	Allison

Nested Loop Semantics

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE not exists
       (SELECT *
        FROM Regist R
        WHERE P.UserID = R.UserID);
```

Output so far

UserID	Name
345	Allison

Payroll

P →

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

Nested Loop Semantics

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE not exists
       (SELECT *
        FROM Regist R
        WHERE P.UserID = R.UserID);
```

Output so far

UserID	Name
345	Allison

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

P → 789

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

R → 123

Compute subquery for P.UserID=789

UserID	Car

Nested Loop Semantics

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE not exists
       (SELECT *
        FROM Regist R
        WHERE P.UserID = R.UserID);
```

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

P

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

R

Compute subquery for P.UserID=789

UserID	Car

Output so far

UserID	Name
345	Allison

Nested Loop Semantics

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE not exists
       (SELECT *
        FROM Regist R
        WHERE P.UserID = R.UserID);
```

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

P → (row 789)

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

R → (row 567 Pinto)

Compute subquery for P.UserID=789

UserID	Car

Output so far

UserID	Name
345	Allison

Nested Loop Semantics

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE not exists
       (SELECT *
        FROM Regist R
        WHERE P.UserID = R.UserID);
```

Output so far

UserID	Name
345	Allison
789	Dan

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

P

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

Compute subquery for P.UserID=789

Output UserID= 567

UserID	Car

Done UserID=567
Not exists answers

Nested Loop Semantics

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE not exists
       (SELECT *
        FROM Regist R
        WHERE P.UserID = R.UserID);
```

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

P →

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

UserID	Name
345	Allison
789	Dan

Final answer

Summary

- Subquery can occur in SELECT/FROM/WHERE
- Sometimes (not always) it is possible to unnest
- Keep in mind edge cases: zero counts
- Most difficult: Existential / universal quantifiers
  Next lecture!