SQL 2

Advanced SQL

Outline

- Aggregate function
- Group by
- Having
- Nested query
- All
- Some
- Exists/ Not Exists
- In/ Not in
- With

Aggregate Function

- In real applications, we often want to perform some summarization tasks.

- Some representative aggregate functions:
  - COUNT([DISTINCT] A): the number of (unique) values in the A column.
  - SUM([DISTINCT] A): the sum of all (unique) values in the A column.
  - AVG([DISTINCT] A): the average of all (unique) values in the A column.
  - MAX(A): the maximum value in the A column.
  - MIN(A): the minimum value in the A column.

Example 1 (Aggregate Function)

- SELECT MAX(*amount*) FROM DEPOSIT

- Answer: 100k

DEPOSIT

acc-id	*cust-id*	*amount*
A1	1	20k
A2	1	5k
A3	2	35k
A3	3	100k
A1	1	35k

- If we replace MAX in the earlier query with MIN, COUNT, SUM, and AVG, the answers are 5k, 5, 195k, and 39k, respectively.

Example 2 (Aggregate Function)

DEPOSIT

acc-id	*cust-id*	*amount*
A1	1	20k
A2	1	5k
A3	2	35k
A3	3	100k
A1	1	35k

- Find the total amount of money that were deposited into the account A1.

- SELECT SUM(*amount*)
  FROM DEPOSIT
  WHERE *acc-id* = ‘A1’

- Find the total number of times that account A1 has been deposited into.

- SELECT COUNT(*)
  FROM DEPOSIT
  WHERE *acc-id* = ‘A1’

Example 2 (cont.)

DEPOSIT

acc-id	*cust-id*	*amount*
A1	1	20k
A2	1	5k
A3	2	35k
A3	3	100k
A1	1	35k

- Find the number of distinct customers that deposited into the account A1.

- SELECT COUNT(DISTINCT *cust-id*)
  FROM DEPOSIT
  WHERE *acc-id* = ‘A1’
  - Answer: 1

- SELECT COUNT(DISTINCT *cust-id*)
  FROM DEPOSIT
  WHERE *acc-id* = ‘A3’
  - Answer: 2

A Common Mistake for Using Aggregate Functions

- Find the account that has the largest deposit amount.

DEPOSIT

acc-id	*cust-id*	*amount*
A1	1	20k
A2	1	5k
A3	2	35k
A3	3	100k
A1	1	35k

- SELECT *acc-id*, MAX(amount) FROM DEPOSIT

- The above query is wrong!

- We cannot simply put the attribute with the aggregate function together in the SELECT clause, unless the query has the GROUP BY clause.

Possible Solutions

- SELECT *acc-id*, *amount*
  FROM DEPOSIT
  ORDER BY *amount* DESC LIMIT 1

- SELECT *acc-id*, *amount*
  FROM DEPOSIT
  WHERE *amount*=MAX(SELECT *amount*
  FROM DEPOSIT)

GROUP BY Clause

- GROUP BY is used when you want to apply aggregate functions to a group of tuples in a relation.
  - e.g., You want to know the number of tuples having the same acc-id

- The SELECT clause can involve (i) **attribute name(s)** and (ii) aggregate functions.
  - Those **attribute(s)** must be some attributes that appear in the GROUP BY clause.
  - Aggregate function can take any attribute as argument.
  - *It must have a single value per group*!

GROUP BY Clause

- SELECT attribute-set-A, aggregate-func(attribute X)
  FROM Table-Name
  GROUP BY attribute-set-B

- attribute-set-A $\subseteq$ attribute-set-B

Example 3 (GROUP BY)

DEPOSIT

acc-id	*cust-id*	*amount*
A1	1	20k
A2	1	5k
A3	2	35k
A3	3	100k
A1	1	35k

- SELECT *acc-id*, SUM(*amount*)
  FROM DEPOSIT
  GROUP BY *acc-id*

acc-id	
A1	55k
A2	5k
A3	135k

3 groups

- SELECT *acc-id*, *cust-id*, SUM(*amount*)
  FROM DEPOSIT
  GROUP BY *acc-id*, *cust-id*

acc-id	*cust-id*	
A1	1	55k
A2	1	5k
A3	2	35k
A3	3	100k

4 groups

A Common Mistake for Using the GROUP BY Clause

DEPOSIT

acc-id	*cust-id*	*amount*
A1	1	20k
A2	1	5k
A3	2	35k
A3	3	100k
A1	1	35k

- The following query (using the GROUP BY clause) is wrong.

  SELECT *acc-id*, *cust-id*, SUM(*amount*)
  FROM DEPOSIT
  GROUP BY *acc-id*

- *cust-id* should not appear in the SELECT clause.

A Common Mistake for Using the GROUP BY Clause

- Find the total amount of money that were ever deposited into each account, provided that the account has been deposited at least twice.

- SELECT *acc-id*, SUM(*amount*)
  FROM DEPOSIT
  WHERE COUNT(*) >= 2
  GROUP BY *acc-id*
  (The query is wrong!)

DEPOSIT

acc-id	*cust-id*	*amount*
A1	1	20k
A2	1	5k
A3	2	35k
A3	3	100k
A1	1	35k

- No aggregate function can be used in the WHERE clause.

- Remember: The WHERE clause filters tuples, while an aggregate function applies to a group. As such, they are incompatible.

Question 1

1. Consider the table, called DEPOSIT. Determine whether the following SQL queries are correct. If it is correct, write down the answer. Otherwise, you need to state the reason.

(a) SELECT *acc-id*, COUNT(*cust-id*)
FROM DEPOSIT
GROUP BY *acc-id*, *cust-id*

(b) SELECT *cust-id*, COUNT(*amount*)
FROM DEPOSIT
GROUP BY *cust-id*

(c) SELECT *acc-id*, SUM(*amount*)
FROM DEPOSIT
GROUP BY *cust-id*

DEPOSIT

acc-id	*cust-id*	*amount*
A1	1	20k
A2	1	5k
A3	2	35k
A3	3	100k
A1	1	35k

Solution to Question 1

-

acc-id	
A1	2
A2	1
A3	1
A3	1

-

cust-id	
1	3
2	1
3	1

- It is incorrect since acc-id in the SELECT clause is not in the attributes of the GROUP BY clause.

HAVING Clause

- HAVING clause is to specify qualifications over groups.
  - e.g., just show the groups in which the number of tuples are more than 2
- HAVING clause <u>applies only to groups</u>, and hence, can be used with GROUP BY only.
- HAVING clause usually contains only aggregate functions.

Example 4 (HAVING)

- Find the total amount of money that were ever deposited into each account, provided that the account has been deposited at least twice.

- SELECT *acc-id*, SUM(*amount*)
  FROM DEPOSIT
  GROUP BY *acc-id*
  HAVING COUNT(*) >= 2

DEPOSIT

acc-id	*cust-id*	*amount*
A1	1	20k
A2	1	5k
A3	2	35k
A3	3	100k
A1	1	35k

Example 4 (cont. 1)

- SELECT *acc-id*, SUM(*amount*)
  FROM DEPOSIT
  GROUP BY *acc-id*
  HAVING COUNT(*) >= 2

DEPOSIT

acc-id	*cust-id*	*amount*
A1	1	20k
A2	1	5k
A3	2	35k
A3	3	100k
A1	1	35k

- First, process GROUP BY.

DEPOSIT

	acc-id	*cust-id*	*amount*
GROUP 1	A1	1	20k
	A1	1	35k
GROUP 2	A2	1	5k
GROUP 3	A3	2	35k
	A3	3	100k

Example 4 (cont. 2)

- SELECT *acc-id*, SUM(*amount*)
  FROM DEPOSIT
  GROUP BY *acc-id*
  HAVING COUNT(*) >= 2

DEPOSIT

acc-id	*cust-id*	*amount*
A1	1	20k
A2	1	5k
A3	2	35k
A3	3	100k
A1	1	35k

- Then, process HAVING to eliminate the groups that do not qualify the HAVING condition.

DEPOSIT

	acc-id	*cust-id*	*amount*
GROUP 1	A1	1	20k
	A1	1	35k
GROUP 3	A3	2	35k
	A3	3	100k

Example 4 (cont. 3)

- SELECT *acc-id*, SUM(*amount*)
  FROM DEPOSIT
  GROUP BY *acc-id*
  HAVING COUNT(*) >= 2

DEPOSIT

acc-id	*cust-id*	*amount*
A1	1	20k
A2	1	5k
A3	2	35k
A3	3	100k
A1	1	35k

- Finally, process SELECT.

acc-id	
A1	55k
A3	135k

Question 2

Consider the following three tables, CUST, ACC, and DEPOSIT.

CUST

cust-id	*name*
1	John
2	Smith

ACC

acc-id	*balance*
A1	20k
A2	5k

DEPOSIT

acc-id	*cust-id*	*amount*
A1	1	1k
A2	1	1k
A2	2	3k

1) If a customer deposited in an account, we consider he/she owns the account. Write the SQL query for displaying the name and the total balances of all accounts of each customer who has made at least two deposits.

2) Write the SQL query for displaying the name and the total balances of all accounts of each customer whose balances are smaller than 10k.

Solution to Question 2

1) SELECT name, SUM(balance)
   FROM CUST, ACC, DEPOSIT
   GROUP BY CUST.cust-id, name
   HAVING COUNT(*)>=2

2) SELECT name, SUM(balance)
   FROM CUST, ACC, DEPOSIT
   GROUP BY CUST.cust-id, name
   HAVING SUM(balance)<10k

Example 5 (Joining Tables with the GROUP BY Clause)

CUST

cust-id	*name*
1	John
2	Smith

ACC

acc-id	*balance*
A1	20k
A2	5k

DEPOSIT

acc-id	*cust-id*	*amount*
A1	1	1k
A2	1	1k
A2	2	3k

- For each customer, we need to display his/her name and the total amount of money in his/her accounts. If a customer deposited in an account, we consider he/she owns the account.

- SELECT *name*, SUM(*balance*)
  FROM CUST, ACC, DEPOSIT
  WHERE CUST.*cust-id* = DEPOSIT.*cust-id* AND ACC.*acc-id* = DEPOSIT.*acc-id*
  GROUP BY *CUST.cust-id*, *name*

Example 5 (cont. 1)

CUST

cust-id	*name*
1	John
2	Smith

ACC

acc-id	*balance*
A1	20k
A2	5k

DEPOSIT

acc-id	*cust-id*	*amount*
A1	1	1k
A2	1	1k
A2	2	3k

- SELECT *name*, SUM(*balance*)
  FROM CUST, ACC, DEPOSIT
  WHERE CUST.*cust-id* = DEPOSIT.*cust-id* AND ACC.*acc-id* = DEPOSIT.*acc-id*
  GROUP BY *CUST.cust-id*, *name*
- First, consider the cartesian product.

CUST. *cust-id*	CUST. *name*	ACC. *acc-id*	ACC. *balance*	DEPOSIT. *acc-id*	DEPOSIT. *cust-id*	DEPOSIT. *amount*
1	John	A1	20k	A1	1	1k
1	John	A1	20k	A2	1	1k
1	John	A1	20k	A2	2	3k
1	John	A2	5k	A1	1	1k
1	John	A2	5k	A2	1	1k
1	John	A2	5k	A2	2	3k
2	Smith	A1	20k	A1	1	1k
2	Smith	A1	20k	A2	1	1k
2	Smith	A1	20k	A2	2	3k
2	Smith	A2	5k	A1	1	1k
2	Smith	A2	5k	A2	1	1k
2	Smith	A2	5k	A2	2	3k

Example 5 (cont. 2)

CUST

cust-id	*name*
1	John
2	Smith

ACC

acc-id	*balance*
A1	20k
A2	5k

DEPOSIT

acc-id	*cust-id*	*amount*
A1	1	1k
A2	1	1k
A2	2	3k

- SELECT *name*, SUM(*balance*)
  FROM CUST, ACC, DEPOSIT
  WHERE CUST.*cust-id* = DEPOSIT.*cust-id* AND ACC.*acc-id* = DEPOSIT.*acc-id*
  GROUP BY *CUST.cust-id*, *name*

- Then, obtain those tuples that fulfill "CUST.*cust-id* = DEPOSIT.*cust-id*" AND ACC.*acc-id* = DEPOSIT.*acc-id*.

CUST. *cust-id*	CUST. *name*	ACC. *acc-id*	ACC. *balance*	DEPOSIT. *acc-id*	DEPOSIT. *cust-id*	DEPOSIT. *amount*
1	John	A1	20k	A1	1	1k
1	John	A2	5k	A2	1	1k
2	Smith	A2	5k	A2	2	3k

Example 5 (cont. 3)

CUST

cust-id	*name*
1	John
2	Smith

ACC

acc-id	*balance*
A1	20k
A2	5k

DEPOSIT

acc-id	*cust-id*	*amount*
A1	1	1k
A2	1	1k
A2	2	3k

- SELECT *name*, SUM(*balance*)
  FROM CUST, ACC, DEPOSIT
  WHERE CUST.*cust-id* = DEPOSIT.*cust-id* AND ACC.*acc-id* = DEPOSIT.*acc-id*
  GROUP BY *CUST.cust-id, name*

- Third, create groups.

	CUST. *cust-id*	CUST. *name*	ACC. *acc-id*	ACC. *balance*	DEPOSIT. *acc-id*	DEPOSIT. *cust-id*	DEPOSIT. *amount*
group 1	1	John	A1	20k	A1	1	1k
	1	John	A2	5k	A2	1	1k
group 2	2	Smith	A2	5k	A2	2	3k

Example 5 (cont. 4)

CUST

cust-id	*name*
1	John
2	Smith

ACC

acc-id	*balance*
A1	20k
A2	5k

DEPOSIT

acc-id	*cust-id*	*amount*
A1	1	1k
A2	1	1k
A2	2	3k

- SELECT *name*, SUM(*balance*)
  FROM CUST, ACC, DEPOSIT
  WHERE CUST.*cust-id* = DEPOSIT.*cust-id* AND ACC.*acc-id* = DEPOSIT.*acc-id*
  GROUP BY *CUST.cust-id*, *name*

- Finally, do projection and summarize aggregates.
  One tuple per group.

CUST. *name*	
John	25k
Smith	5k

Example 6

CUST

cust-id	*name*
1	John
2	Smith

ACC

acc-id	*balance*
A1	20k
A2	5k

DEPOSIT

acc-id	*cust-id*	*amount*
A1	1	1k
A2	1	1k
A2	2	3k

- For each customer who has made at least 2 deposits, display her/his name, and the total amount of money in her/his accounts.

- SELECT *name*, SUM(*balance*)
  FROM CUST, ACC, DEPOSIT
  WHERE CUST.*cust-id* = DEPOSIT.*cust-id* AND ACC.*acc-id* = DEPOSIT.*acc-id*
  GROUP BY *CUST.cust-id*, *name*
  HAVING COUNT(*) >= 2

- Let us understand the query step-by-step.

Example 6 (cont. 1)

CUST

cust-id	*name*
1	John
2	Smith

ACC

acc-id	*balance*
A1	20k
A2	5k

DEPOSIT

acc-id	*cust-id*	*amount*
A1	1	1k
A2	1	1k
A2	2	3k

- SELECT *name*, SUM(*balance*)
  FROM CUST, ACC, DEPOSIT
  WHERE CUST.*cust-id* = DEPOSIT.*cust-id* AND ACC.*acc-id* = DEPOSIT.*acc-id*
  GROUP BY *CUST.cust-id, name*
  HAVING COUNT(*) >= 2

- Do cartesian product, tuple filtering, and then create groups.

	CUST. *cust-id*	CUST. *name*	ACC. *acc-id*	ACC. *balance*	DEPOSIT. *acc-id*	DEPOSIT. *cust-id*	DEPOSIT. *amount*
group 1	1	John	A1	20k	A1	1	1k
	1	John	A2	5k	A2	1	1k
group 2	2	Smith	A2	5k	A2	2	3k

Example 6 (cont. 2)

CUST

cust-id	*name*
1	John
2	Smith

ACC

acc-id	*balance*
A1	20k
A2	5k

DEPOSIT

acc-id	*cust-id*	*amount*
A1	1	1k
A2	1	1k
A2	2	3k

- SELECT *name*, SUM(*balance*)
  FROM CUST, ACC, DEPOSIT
  WHERE CUST.*cust-id* = DEPOSIT.*cust-id* AND ACC.*acc-id* = DEPOSIT.*acc-id*
  GROUP BY *CUST.cust-id*, *name*
  HAVING COUNT(*) >= 2

- Process HAVING.

	CUST. *cust-id*	CUST. *name*	ACC. *acc-id*	ACC. *balance*	DEPOSIT. *acc-id*	DEPOSIT. *cust-id*	DEPOSIT. *amount*
group 1	1	John	A1	20k	A1	1	1k
	1	John	A2	5k	A2	1	1k

Example 6 (cont. 3)

CUST

cust-id	*name*
1	John
2	Smith

ACC

acc-id	*balance*
A1	20k
A2	5k

DEPOSIT

acc-id	*cust-id*	*amount*
A1	1	1k
A2	1	1k
A2	2	3k

- SELECT *name*, SUM(*balance*)
  FROM CUST, ACC, DEPOSIT
  WHERE CUST.*cust-id* = DEPOSIT.*cust-id* AND ACC.*acc-id* = DEPOSIT.*acc-id*
  GROUP BY *CUST.cust-id*, *name*
  HAVING COUNT(*) >= 2

- Finally, process SELECT.

CUST. *name*	
John	25k

Example 7

CUST

cust-id	*name*
1	John
2	Smith

ACC

acc-id	*balance*
A1	20k
A2	5k

DEPOSIT

acc-id	*cust-id*	*amount*
A1	1	1k
A2	1	1k
A2	2	3k

- For each customer, display her/his name, and the total amount of money in her/his accounts whose balances are larger than 15k.

- SELECT *name*, SUM(*balance*)
  FROM CUST, ACC, DEPOSIT
  WHERE CUST.*cust-id* = DEPOSIT.*cust-id* AND ACC.*acc-id* = DEPOSIT.*acc-id*
  GROUP BY *CUST.cust-id*, *name*
  HAVING SUM(balance) > 15000

- Wrong !
  - Easy to make mistake between using WHERE and HAVING

Example 7 (cont. 1)

CUST

cust-id	*name*
1	John
2	Smith

ACC

acc-id	*balance*
A1	20k
A2	5k

DEPOSIT

acc-id	*cust-id*	*amount*
A1	1	1k
A2	1	1k
A2	2	3k

- For each customer, display her/his name, and the total amount of money in her/his accounts whose balances are larger than 15k.

- SELECT *name*, SUM(*balance*)
  FROM CUST, ACC, DEPOSIT
  WHERE CUST.*cust-id* = DEPOSIT.*cust-id* AND ACC.*acc-id* = DEPOSIT.*acc-id* AND ACC.*balance* > 15000
  GROUP BY *CUST.cust-id*, *name*

- Note: Here we put the addition condition into WHERE, not HAVING.
- Let us understand the query step-by-step.

Example 7 (cont. 2)

CUST

cust-id	*name*
1	John
2	Smith

ACC

acc-id	*balance*
A1	20k
A2	5k

DEPOSIT

acc-id	*cust-id*	*amount*
A1	1	1k
A2	1	1k
A2	2	3k

- SELECT *name*, SUM(*balance*)
  FROM CUST, ACC, DEPOSIT
  WHERE CUST.*cust-id* = DEPOSIT.*cust-id* AND ACC.*acc-id* = DEPOSIT.*acc-id* AND ACC.*balance* > 15000
  GROUP BY *CUST.cust-id, name*

- Do cartesian product, tuple filtering, and then create groups.

CUST. *cust-id*	CUST. *name*	ACC. *acc-id*	ACC. *balance*	DEPOSIT. *acc-id*	DEPOSIT. *cust-id*	DEPOSIT. *amount*
1	John	A1	20k	A1	1	1k

Example 7 (cont. 3)

CUST

cust-id	*name*
1	John
2	Smith

ACC

acc-id	*balance*
A1	20k
A2	5k

DEPOSIT

acc-id	*cust-id*	*amount*
A1	1	1k
A2	1	1k
A2	2	3k

- SELECT *name*, SUM(*balance*)
  FROM CUST, ACC, DEPOSIT
  WHERE CUST.*cust-id* = DEPOSIT.*cust-id* AND ACC.*acc-id* = DEPOSIT.*acc-id* AND ACC.*balance* > 15000
  GROUP BY *CUST.cust-id*, *name*

- Process SELECT.

CUST. *name*	
John	20k

Nested Query

- A nested query is a query that has another query embedded within it. The embedded query is called a ***subquery***.

- A subquery typically appears within the WHERE clause, the FROM clause, or the HAVING clause.

Example 8 (Nested Query in WHERE)

- Find the id of the account with the largest balance.

ACC

acc-id	*cust-id*	*balance*
A1	1	20k
A2	1	5k
A3	2	35k

- SELECT *acc-id*
  FROM ACC
  WHERE *balance* = (SELECT MAX(*balance*)
  FROM ACC)

a nested query

Example 8 (Nested Query in WHERE)

- Find the ids of the accounts whose balances are not the largest.

ACC

acc-id	*cust-id*	*balance*
A1	1	20k
A2	1	5k
A3	2	15k
A4	3	100k

- SELECT *acc-id*
  FROM ACC
  WHERE *balance* < (SELECT MAX(*balance*)
  FROM ACC)

Example 9 (Nested Query in FROM)

- Let us define the wealth of a customer as the total amount of money in all his/her accounts.

ACC

acc-id	*cust-id*	*balance*
A1	1	20k
A2	1	2k
A3	2	100k
A4	3	3k
A5	3	1k

- Display the average wealth of the customers whose wealth is larger than 20k.

- SELECT AVG(*wealth*)
  FROM (SELECT SUM(*balance*) as *wealth* FROM ACC
  GROUP BY *cust-id*)
  AS TMP
  WHERE *wealth* > 20000

- Result: 61k

Example 9 (Nested Query in FROM)

- SELECT AVG(*wealth*)
  FROM (SELECT SUM(*balance*) as *wealth*
  FROM ACC
  GROUP BY *cust-id*)
  AS TMP
  WHERE *wealth* > 20000

ACC

acc-id	*cust-id*	*balance*
A1	1	20k
A2	1	2k
A3	2	100k
A4	3	3k
A5	3	1k

- First, execute the nested query in FROM, which produces a table named TMP, involving a single-column *wealth*.

- Conceptually, the above query can be simplified as:
  SELECT AVG(*wealth*)
  FROM TMP
  WHERE *wealth* > 20000

TMP

wealth
22k
100k
4k

- Hence, the result is the average of the first two rows in TMP.

Example 10 (ALL)

- Find the ids of the accounts whose balances are larger than the balances of all accounts owned by the customer with *cust-id* = 1.

ACC

acc-id	*cust-id*	*balance*
A1	1	20k
A2	1	5k
A3	2	15k
A4	3	100k

- SELECT *acc-id*
  FROM ACC
  WHERE *balance* > ALL (SELECT *balance*
  FROM ACC
  WHERE *cust-id* = 1)

has to be a comparison operator (e.g., >, <, =)

Example 11 (SOME)

- Find the ids of the accounts whose balances are larger than the balance of at least one account owned by the customer with *cust-id* = 1.

ACC

acc-id	*cust-id*	*balance*
A1	1	20k
A2	1	5k
A3	2	15k
A4	3	100k

- SELECT *acc-id*
  FROM ACC
  WHERE *balance* > SOME (SELECT *balance*
  FROM ACC
  WHERE *cust-id* = 1)

- Answer: A1, A3, A4

Question 3

Consider the following table, called Instructors.

dept_name	ins_name	salary
CS	Bob	44000
Chem	Smith	38000
CS	Tom	45000
Chem	Anne	50000
Math	Jone	33000

1) Write the SQL query to display the department names (dept_name) and the average salary of those instructors where the average salary must be greater than 42000.
2) Write the SQL query to display the name of each instructor whose salary is greater than the maximum salary of instructors in at least one department.
3) Write the SQL query to display the name of each instructor whose salary is greater than the average salaries of instructors in all departments.

Solution to Question 3

1)
```
SELECT dept_name, avg_salary
FROM (SELECT dept_name, AVG(salary) AS avg_salary
        FROM Instructors
        GROUP BY dept_name)
WHERE avg_salary>42000
```

2)
```
SELECT T1.ins_name
FROM Instructors T1
WHERE T1.salary>SOME(SELECT MAX(T2.salary) AS max_salary
                     FROM Instructors T2
                     GROUP BY T2.dept_name)
```

3)
```
SELECT T1.ins_name
FROM Instructors T1
WHERE T1.salary > ALL(SELECT AVG(T2.salary) AS avg_salary
                      FROM Instructors T2
                      GROUP BY T2.dept_name)
```

Example 12 (EXISTS)

- Find the ids of the accounts whose balances are not the largest.
  - SELECT *acc-id*
    FROM ACC
    WHERE *balance* < (SELECT MAX(*balance*)
    FROM ACC)

- Any other ways?

ACC

acc-id	*cust-id*	*balance*
A1	1	20k
A2	1	5k
A3	2	15k
A4	3	100k

Example 12 (EXISTS)

ACC

acc-id	*cust-id*	*balance*
A1	1	20k
A2	1	5k
A3	2	15k
A4	3	100k

- Find the ids of the accounts whose balances are not the largest.

- SELECT T1.*acc-id*
  FROM ACC T1
  WHERE EXISTS (SELECT *
  FROM ACC T2
  WHERE T1.*balance* < T2.*balance*)

- Note that this nested query is different from the previous nested queries we have seen: It depends on the outside query.
  - T1 in the nested query references the table in the outside query.

- Lets see how it is executed.

Example 12 (cont. 1)

ACC

acc-id	*cust-id*	*balance*
A1	1	20k
A2	1	5k
A3	2	15k
A4	3	100k

- SELECT T1.*acc-id*
  FROM ACC T1
  WHERE EXISTS (SELECT *
  FROM ACC T2
  WHERE T1.*balance* < T2.*balance*)

- Lets go over every tuple in T1 one by one. For each tuple, get its balance, and place it at the position of T1.*balance* to make the nested query complete.
- Specifically, when we are looking at the first tuple in T1, the nested query becomes:
  SELECT *
  FROM ACC T2
  WHERE 20k < T2.*balance*

- Execute it – does it return any tuples?
- Yes, so EXISTS evaluates to true, and the *acc-id* of the tuple in T1 we are looking at is displayed.

Example 12 (cont. 2)

ACC

acc-id	*cust-id*	*balance*
A1	1	20k
A2	1	5k
A3	2	15k
A4	3	100k

- SELECT T1.*acc-id*
  FROM ACC T1
  WHERE EXISTS (SELECT *
  FROM ACC T2
  WHERE T1.*balance* < T2.*balance*)

- Repeat the above process for all tuples in T1.
- The *acc-ids* of all tuples are displayed, until we come to the last tuple, for which the nested query has the form:
  SELECT *
  FROM ACC T2
  WHERE 100k < T2.*balance*

- Execute it – does it return any tuples?
- No, so EXISTS evaluates to false, and the *acc-id* of the last tuple in T1 is not displayed.

Example 13 (NOT EXISTS)

- Find the id of the account whose balance is the largest.
  - SELECT *acc-id*
    FROM ACC
    WHERE *balance* = (SELECT MAX(*balance*)
    FROM ACC)

ACC

acc-id	*cust-id*	*balance*
A1	1	20k
A2	1	5k
A3	2	15k
A4	3	100k

- Any other ways?
  - SELECT T1.*acc-id*
    FROM ACC T1
    WHERE NOT EXISTS (SELECT *
    FROM ACC T2
    WHERE T1.*balance* < T2.*balance*)

Example 14 (IN)

DEPOSIT

acc-id	*cust-id*	*amount*
A1	1	2k
A1	1	1k
A2	1	1k
A2	2	3k
A3	3	2k
A3	2	5k

- Find the ids of accounts that were deposited into by both customers with *cust-id* = 1 and 2.

- SELECT *acc-id*
  FROM DEPOSIT
  WHERE *cust-id* = 2 AND *acc-id* IN (SELECT *acc-id*
  FROM DEPOSIT
  WHERE *cust-id* = 1)

- Answer: A2

- Any other ways?

Example 14 (cont.)

DEPOSIT

acc-id	*cust-id*	*amount*
A1	1	2k
A1	1	1k
A2	1	1k
A2	2	3k
A3	3	2k
A3	2	5k

- Find the ids of accounts that were deposited into by both customers with *cust-id* = 1 and 2.

- (SELECT *acc-id*
  FROM DEPOSIT
  WHERE *cust-id* = 2)

  INTERSECT

  (SELECT *acc-id*
  FROM DEPOSIT
  WHERE *cust-id* = 1)

- Answer: A2

Example 15 (NOT IN)

DEPOSIT

acc-id	*cust-id*	*amount*
A1	1	2k
A1	1	1k
A2	1	1k
A2	2	3k
A3	3	2k
A3	2	5k

- Find the ids of accounts that were deposited into by the customer with *cust-id* = 2 but not by the customer with and *cust-id* = 1.

- SELECT *acc-id*
  FROM DEPOSIT
  WHERE *cust-id* = 2 AND *acc-id* NOT IN (SELECT *acc-id*
  FROM DEPOSIT
  WHERE *cust-id* = 1)

- Answer: A3

- Any other ways?

Example 15 (cont.)

DEPOSIT

acc-id	*cust-id*	*amount*
A1	1	2k
A1	1	1k
A2	1	1k
A2	2	3k
A3	3	2k
A3	2	5k

- Find the ids of accounts that were deposited into by the customer with *cust-id* = 2 but not by the customer with and *cust-id* = 1.

- (SELECT *acc-id*
  FROM DEPOSIT
  WHERE *cust-id* = 2)

  EXCEPT

  (SELECT *acc-id*
  FROM DEPOSIT
  WHERE *cust-id* = 1)

- Answer: A3

Example 16 (WITH)

- Let us define the wealth of a customer as the total amount of money in all her/his accounts.

- Display the average wealth of the customers whose wealth is larger than 20k.

ACC

acc-id	*cust-id*	*balance*
A1	1	20k
A2	1	2k
A3	2	100k
A4	3	3k
A5	3	1k

- WITH TMP AS
  (SELECT SUM(*balance*) as *wealth* FROM ACC
  GROUP BY *cust-id*)
  SELECT AVG(*wealth*)
  FROM TMP
  WHERE *wealth* > 20000

- Result: 61k

Question 4

Consider the following tables, namely ACC and DEPOSIT.

ACC

acc-id	*balance*
A1	20k
A2	18k
A3	10k

DEPOSIT

acc-id	*cust-id*	*amount*
A1	1	2k
A1	1	1k
A2	1	1k
A2	2	3k
A3	3	2k
A3	2	5k

Write the SQL query to find the ids of the accounts which does not appear in the DEPOSIT table.

Solution to Question 4

- SELECT ACC.acc-id
  FROM ACC
  WHERE NOT EXISTS (SELECT *
  FROM DEPOSIT
  WHERE ACC.acc-id=DEPOSIT.acc-id)

Question 5

- Consider the following tables, namely ACC and DEPOSIT.

ACC

acc-id	*balance*
A1	20k
A2	18k
A3	10k

DEPOSIT

acc-id	*cust-id*	*amount*
A1	1	2k
A1	1	1k
A2	1	1k
A2	2	3k
A3	3	2k
A3	2	5k

  - Write two different SQL queries to find the ids of the accounts that were deposited by the customer with *cust-id*=2, and their balances.

Solution to Question 5

- SELECT *acc-id*, *balance*
  FROM ACC
  WHERE *acc-id* IN (SELECT *acc-id*
  FROM DEPOSIT
  WHERE *cust-id* = 2)

- SELECT ACC.*acc-id*, *ACC.balance*
  FROM ACC
  WHERE EXISTS (SELECT *
  FROM DEPOSIT
  WHERE ACC.*acc-id*=DEPOSIT.*acc-id*
  AND *cust-id* = 2)

What have we learned?

- Aggregate functions (MAX, MIN, COUNT, SUM, and AVG)
- GROUP BY, HAVING
- Nested queries
- ALL, SOME
- EXISTS, NOT EXISTS
- IN, NOT IN
- WITH