The Hong Kong
University of Science
and Technology

□□□□□□□□

Structured Query Language 2

❖Find the ids of the accounts which have
been deposited into by more than one
customer.

❖An answer without any nested query:

<u>*dep-id*</u>	*acc-id*	*cust-id*	*amount*
070940	A1	1	2K
070941	A1	1	1K
070943	A2	1	1K
070945	A2	2	3K
070959	A3	3	2K
080341	A3	2	5K

select *acc-id*
from *deposit*
group by *acc-id*
having count (*cust-id*) >= 2

select *acc-id*
from *deposit*
group by *acc-id*
having count (distinct *cust-id*) >= 2

If there is no **distinct** here, A1 will also be displayed.

- Consider table: *deposit* (*<u>dep-id</u>*, *acc-id*, *cust-id*, *amount*).
  We want to retrieve the *cust-id* of the customers who deposited into two accounts with *acc-id* 'A1' and 'A2', respectively.

<u>*dep-id*</u>	*acc-id*	*cust-id*	*amount*
070940	A1	1	2K
070941	A1	1	1K
070943	A2	1	1K
070945	A2	2	3K
070959	A3	3	2K
080341	A3	2	5K

- Write an SQL query with **intersect**.

(**select distinct** *cust-id* **from** *deposit* **where** *acc-id* = 'A1')
intersect
(**select distinct** *cust-id* **from** *deposit* **where** *acc-id* = 'A2')

- Write a nested SQL query without **intersect**.

```
select  distinct cust-id from deposit
where acc-id = 'A1' and cust-id in
               (select cust-id from deposit
                where acc-id = 'A2')
```

- Again consider table: *deposit* (*<u>dep-id</u>*, *acc-id*, *cust-id*, *amount*).
  We want to retrieve the *cust-id* of the customers who deposited into two accounts with *acc-id* 'A1' and 'A2', respectively.

- Write an SQL query that contains only one **select** .

```
select  distinct T1.cust-id
from    deposit T1, deposit T2
where   T1.cust-id = T2.cust-id and
        T1.acc-id = 'A1' and
        T2.acc-id = 'A2'
```

<u>dep-id</u>	*acc-id*	*cust-id*	*amount*
070940	A1	1	2K
070941	A1	1	1K
070943	A2	1	1K
070945	A2	2	3K
070959	A3	3	2K
080341	A3	2	5K

- Again consider table: *deposit* (*<u>dep-id</u>*, *acc-id*, *cust-id*, *amount*).
  We want to retrieve the *cust-id* of the customers who deposited into the account with *acc-id* = 'A1' or 'A2' but not both.

- Write an SQL query that contains only one SELECT.

select *cust-id* **from** *deposit*
where *acc-id* = 'A1' **or** *acc-id* = 'A2'
group by *cust-id*
having count (distinct *acc-id*) = 1

dep_id	*acc-id*	*cust-id*	*amount*
070940	A1	1	2K
070941	A1	1	1K
070943	A2	1	1K
070945	A2	2	3K
070959	A3	3	2K
080341	A2	2	5K

- Again consider table: *deposit* (*dep-id*, *acc-id*, *cust-id*, *amount*).
  We want to retrieve the *cust-id* of the customers who deposited into the account with *acc-id* = 'A1' or 'A2' but not both.

- Write an SQL query that contains only one SELECT.

select *cust-id* **from** *deposit*
where *acc-id* = 'A1' **or** *acc-id* = 'A2'
group by *cust-id*
having count (**distinct** *acc-id*) = 1

Indispensable!

dep_id	acc-id	cust-id	amount
070940	A1	1	2K
070941	A1	1	1K
070943	A2	1	1K
070945	A2	2	3K
080341	A2	2	5K

- ❖ *deposit* (*<u>dep-id</u>*, *acc-id*, *cust-id*, *amount*).
  Retrieve the *cust-id* of the customer who deposited the largest number of times.

<u>dep-id</u>	*acc-id*	*cust-id*	*amount*
070940	A1	1	2K
070941	A1	1	1K
070943	A2	1	1K
070945	A2	2	3K
070959	A3	3	2K
080341	A3	2	5K

```
select  cust-id from deposit
group by cust-id
having count  (*) >= all
            (select count (*) from deposit
              group by cust-id)
```

OR

```
select  cust-id

from   (select cust-id, count (*) as num from deposit
             group by cust-id) as Temp

where  num = (select max(num) from Temp)
```

- We all know how to retrieve the "largest" tuple. Now let us see how to retrieve the second largest!

- *account* (*acc-id*, *balance*).
  Write an SQL query to retrieve the *acc-id* of the account with the **second** largest balance.

- *account* (*acc-id*, *balance*).
- Write an SQL query to retrieve the *acc-id* of the *account* with the second largest balance.

```
select  T1.acc-id from account T1
where T1.balance not in (select max (balance) from account)      ⇒   T1<>MAX
        and not exists
        (select * from account T2                                      !(T1<T2 &
         where T1.balance < T2.balance and                             T2!=MAX)
              T2.balance not in (select max (balance) from account))
```

- The next slide shows yet another answer.

- *account* (*acc-id*, *balance*).
- Write an SQL query to retrieve the *acc-id* of the *account* with the second largest balance.
- Second largest balance = max(balance <> max(balance))

```
select  acc-id from account
where balance =
         (select max (balance)
          from account
          where balance <>
         (select max (balance) from account))
```

- The next slide shows another solution.

- *account* (<u>*acc-id*</u>, *balance*).
- Write an SQL query to retrieve the *acc-id* of the *account* with the second largest balance.

```
select  T1.acc-id
from    account T1, account T2
where T1.balance < T2.balance
group by T1.acc-id
having count (distinct T2.balance) = 1
```

Indispensable!

- Question: Consider the following schemas.
  CUST (*cust-id*, *name*), and
  WITHDRAW (*w-id*, *cust-id*, *acc-id*, *date*, *amount*)

- Write an SQL query to retrieve all the names of the customers who have withdrawn more than 1k dollars in a single withdrawal.
  If a customer made several such withdrawals, her/his name should be reported only once.

- Answer: Consider the following schemas.
  CUST (*cust-id*, *name*), and
  WITHDRAW (*w-id*, *cust-id*, *acc-id*, *date*, *amount*)

- Write an SQL query to retrieve all the names of the customers who have withdrawn more than 1k dollars in a single withdrawal.
  If a customer made several such withdrawals, her/his name should be reported only once.

select distinct *name*
from CUST T1, WITHDRAW T2
where T1.*cust-id* = T2.*cust-id* **and** T2.*amount* > 1k

- Question: Consider the following schemas.
  CUST (*<u>cust-id</u>*, *name*), and
  WITHDRAW (*<u>w-id</u>*, *cust-id*, *acc-id*, *date*, *amount*)

- Let us use the name "interesting account" to refer to the account from which the withdrawal with smallest amount was made.
- Retrieve the *acc-id* of accounts from which withdrawals have been made, except the interesting account.

- Answer: Consider the following schemas.
  CUST (<u>*cust-id*</u>, *name*), and
  WITHDRAW (<u>*w-id*</u>, *cust-id*, *acc-id*, *date*, *amount*)

- Let us use the name "interesting account" to refer to the account from which the withdrawal with smallest amount was made.
- Retrieve the *acc-id* of accounts from which withdrawals have been made, except the interesting account.

```
select distinct acc-id from WITHDRAW
where acc-id not in
    (select acc-id from WITHDRAW
     where amount =
          (select min (amount)
           from WITHDRAW))
```

```
select acc-id from WITHDRAW
 where amount <>
 (select min (amount)
 from WITHDRAW)
```

- Question: Consider the following schemas.
  CUST (*cust-id*, *name*), and
  WITHDRAW (*w-id*, *cust-id*, *acc-id*, *date*, *amount*)

- Sometimes there may be a "shared" account, namely, an account with multiple owners.

- Write an SQL query to return the *acc-id* of all the shared accounts. You may assume that all the owners of a shared account have made withdrawals from the account.

- Answer: Consider the following schemas.
  CUST (*cust-id*, *name*), and
  WITHDRAW (*w-id*, *cust-id*, *acc-id*, *date*, *amount*)

- Sometimes there may be a "shared" account, namely, an account (acc-id) with multiple owners (cust-id).

- Write an SQL query to return the *acc-id* of all the shared accounts. You may assume that all the owners of a shared account have made withdrawals from the account.

select T1.*acc-id*
from WITHDRAW T1, WITHDRAW T2
where T1.*cust-id* <> T2.*cust-id* **and** T1.*acc-id* = T2.*acc-id*

select *acc-id*
from WITHDRAW
group by *acc-id*
having count(**distinct** *cust-id*)>1

- Question: As with natural join, outer joins are not compulsory operators. That is, we can implement an outer join using "conventional" SQL.

- Let us verify this for left outer join.

- CS-PROF (*prof-id*, *name*)
- SUPERVISION (*prof-id*, *stu-id*)

- Write an alternative query that returns the same information as

  select *prof-id, name, stu-id*
  from CS-PROF **left outer join** SUPERVISION
  on CS-PROF.*prof-id* = SUPERVISION.*prof-id*

- Answer:
- CS-PROF (*prof-id*, *name*)
- SUPERVISION (*prof-id*, *stu-id*)

```
select prof-id, name, stu-id
from CS-PROF left outer join SUPERVISION
         on CS-PROF.prof-id = SUPERVISION.prof-id
```

```
(select T1.prof-id, name, stu-id
 from CS-PROF T1, SUPERVISION T2
 where T1.prof-id = T2.prof-id)
union
(select prof-id, name, NULL
 from CS-PROF T1
 where not exists
         (select * from SUPERVISION T2
          where T1.prof-id = T2.prof-id))
```

- We can see that **left outer join** simplifies the query significantly.

- Question: Consider MARKS(*stu-id*, *course-id*, *score*)
- Write a query to retrieve the *stu-id* of every student who scored at least 80 in all the courses s/he took, but scored less than 90 in at least one course.
- Your query should not contain more than 2 **select**.

- Answer: Consider MARKS(*stu-id*, *course-id*, *score*).
- Write a query to retrieve the *stu-id* of every student who scored at least 80 in all the courses s/he took, but scored less than 90 in at least one course.

```
(select stu-id from MARKS
 where score < 90)
except
(select stu-id from MARKS
 where score < 80)
```

- Answer: Consider MARKS (*stu-id*, *course-id*, *score*).
- Write a query to retrieve the *stu-id* of every student who scored at least 80 in all the courses s/he took, but scored less than 90 in at least one course.

select *stu-id*
from MARKS
group by *stu-id*
having min (*score*) >= 80 **and min** (*score*) < 90

- What will happen if some of values of score are NULL in table ?

- All aggregate operations except **count(*)** ignore tuples with null values on the aggregated attributes.