SQL 1

Introduction

- *SQL—Structured Query Language*
  - Pronounced "S-Q-L" or "sequel"
  - The query language of all commercial database systems (including Oracle, MS Access, MySQL, etc.)

- You have seen some examples of SQL in the last lecture.

Basic SQL Query

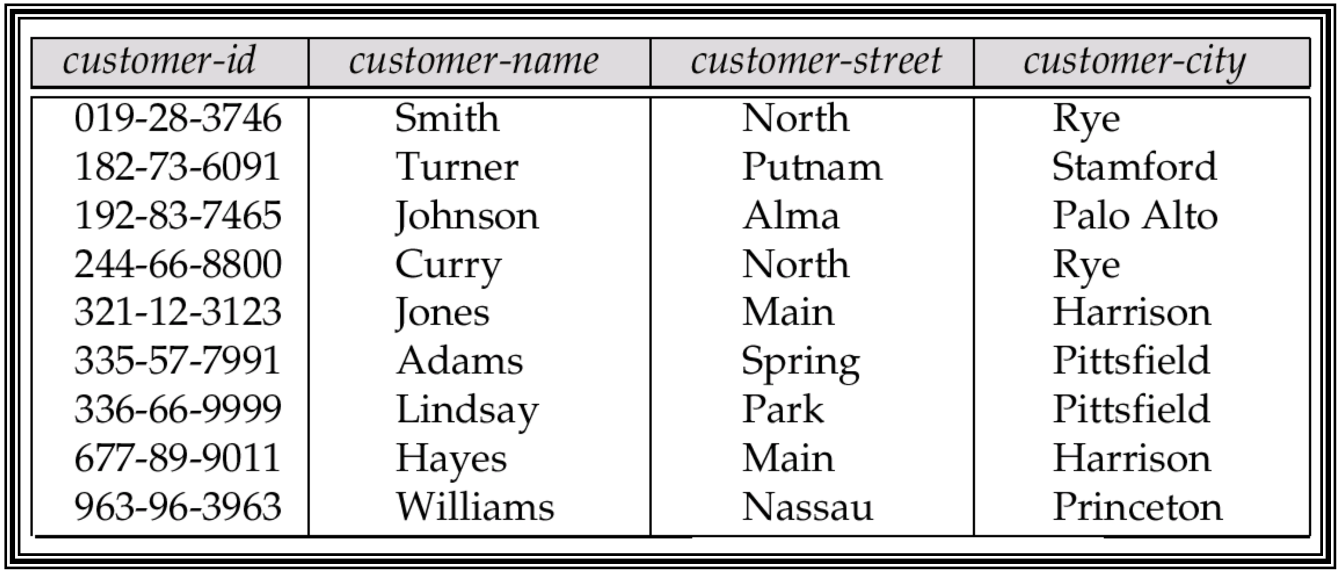
```
SELECT     [DISTINCT]  target-list
FROM       relation-list
WHERE      qualification
```

- relation-list A list of relation names.
- target-list A list of attributes of relations in the relation-list.
- qualification Query conditions combined using AND and OR.
- DISTINCT is an optional keyword indicating that the answer should not contain duplicates. Default is that duplicates are not eliminated!

SQL Example 1 (SELECT and FROM)

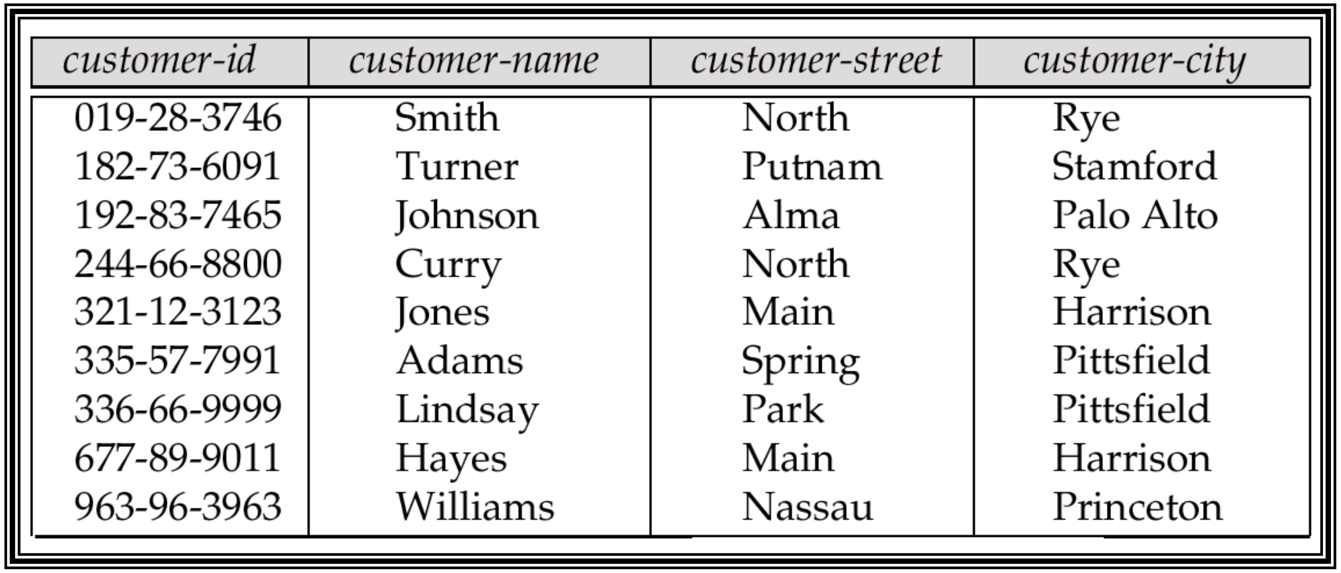
- SELECT *
  FROM CUST

- Simply returns the entire table CUSTOMER.
- * = all attributes



customer-id	*customer-name*	*customer-street*	*customer-city*
019-28-3746	Smith	North	Rye
182-73-6091	Turner	Putnam	Stamford
192-83-7465	Johnson	Alma	Palo Alto
244-66-8800	Curry	North	Rye
321-12-3123	Jones	Main	Harrison
335-57-7991	Adams	Spring	Pittsfield
336-66-9999	Lindsay	Park	Pittsfield
677-89-9011	Hayes	Main	Harrison
963-96-3963	Williams	Nassau	Princeton

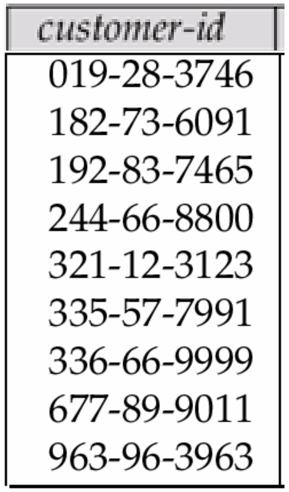
SQL Example 2 (SELECT and FROM)



customer-id	*customer-name*	*customer-street*	*customer-city*
019-28-3746	Smith	North	Rye
182-73-6091	Turner	Putnam	Stamford
192-83-7465	Johnson	Alma	Palo Alto
244-66-8800	Curry	North	Rye
321-12-3123	Jones	Main	Harrison
335-57-7991	Adams	Spring	Pittsfield
336-66-9999	Lindsay	Park	Pittsfield
677-89-9011	Hayes	Main	Harrison
963-96-3963	Williams	Nassau	Princeton

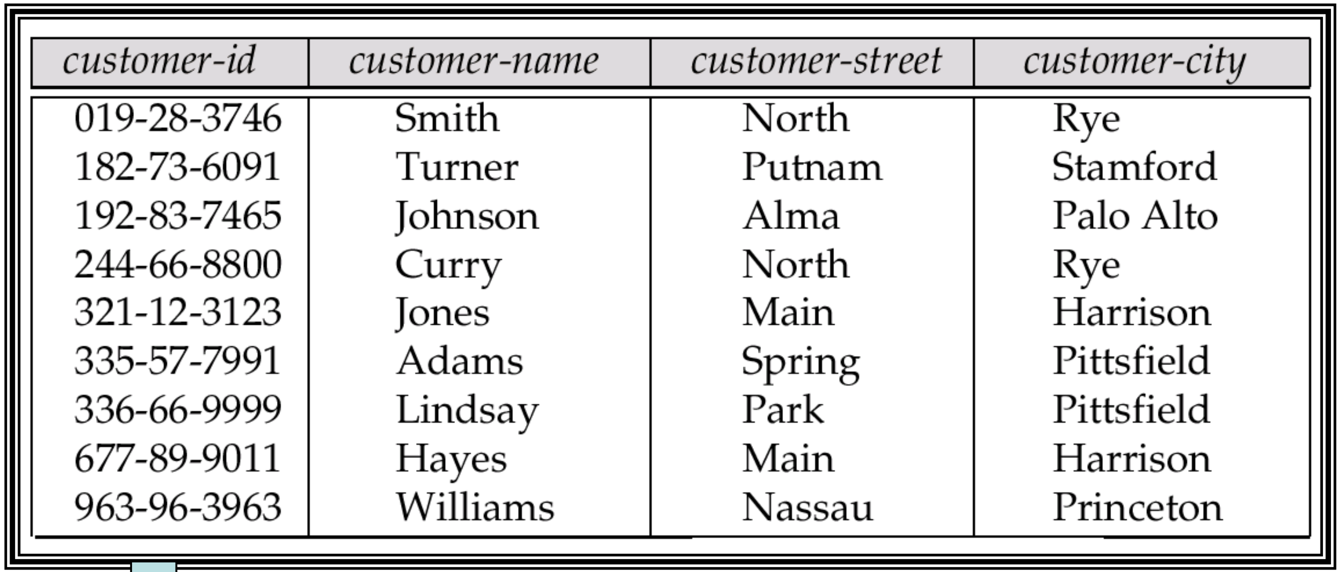
- SELECT *customer-id*
  FROM CUSTOMER

- Only retains the customer-id attributes of all records in this relation.



customer-id
019-28-3746
182-73-6091
192-83-7465
244-66-8800
321-12-3123
335-57-7991
336-66-9999
677-89-9011
963-96-3963

SQL Example 2 (AS)



customer-id	*customer-name*	*customer-street*	*customer-city*
019-28-3746	Smith	North	Rye
182-73-6091	Turner	Putnam	Stamford
192-83-7465	Johnson	Alma	Palo Alto
244-66-8800	Curry	North	Rye
321-12-3123	Jones	Main	Harrison
335-57-7991	Adams	Spring	Pittsfield
336-66-9999	Lindsay	Park	Pittsfield
677-89-9011	Hayes	Main	Harrison
963-96-3963	Williams	Nassau	Princeton

cid
019-28-3746
182-73-6091
192-83-7465
244-66-8800
321-12-3123
335-57-7991
336-66-9999
677-89-9011
963-96-3963

- SELECT *customer-id* AS cid
  FROM CUSTOMER

- Use **AS** in the SELECT clause to rename output columns.

SQL Example 3 (DISTINCT)

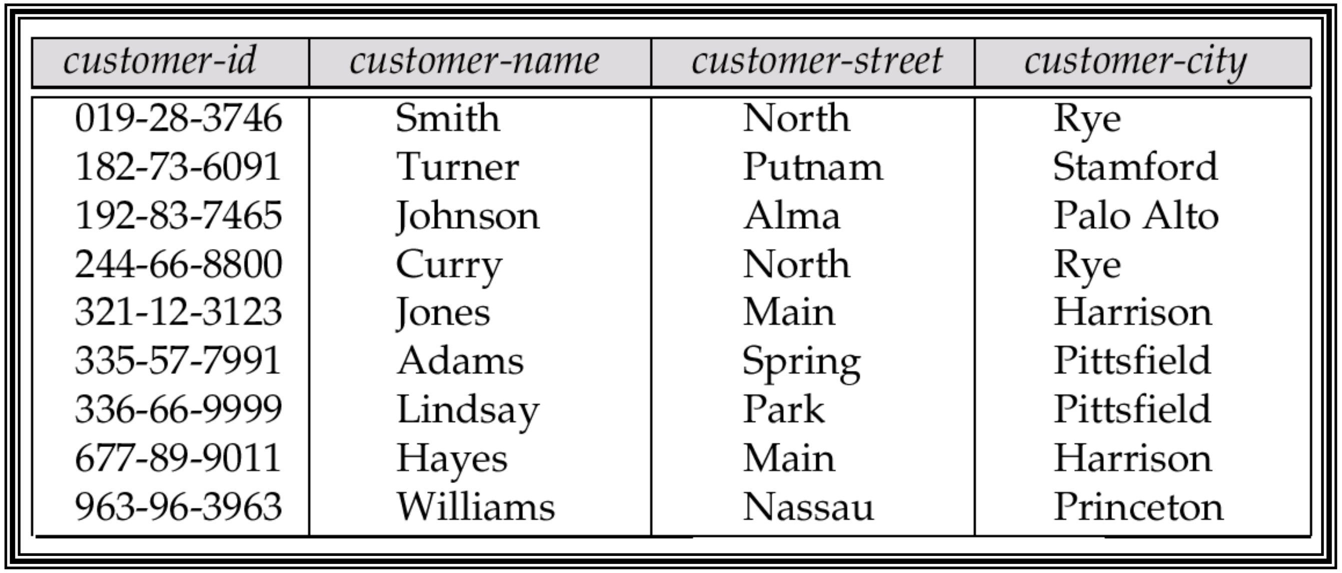
customer-id	*customer-name*	*customer-street*	*customer-city*
019-28-3746	Smith	North	Rye
182-73-6091	Turner	Putnam	Stamford
192-83-7465	Johnson	Alma	Palo Alto
244-66-8800	Curry	North	Rye
321-12-3123	Jones	Main	Harrison
335-57-7991	Adams	Spring	Pittsfield
336-66-9999	Lindsay	Park	Pittsfield
677-89-9011	Hayes	Main	Harrison
963-96-3963	Williams	Nassau	Princeton

customer-city
Rye
Stamford
Palo Alto
Rye
Harrison
Pittsfield
Pittsfield
Harrison
Princeton

- SELECT *customer-city*
  FROM CUSTOMER
- SELECT DISTINCT *customer-city*
  FROM CUSTOMER
- Removes duplicates using this SQL query.

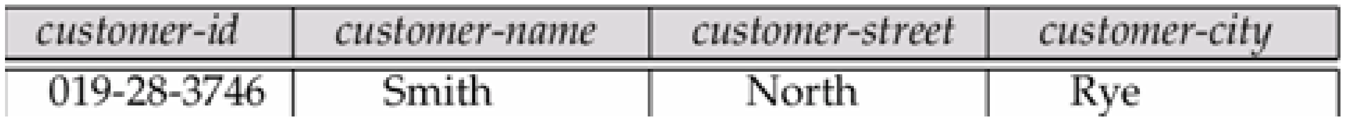
customer-city
Rye
Stamford
Palo Alto
~~Rye~~
Harrison
Pittsfield
~~Pittsfield~~
~~Harrison~~
Princeton

SQL Example 4 (WHERE)



customer-id	*customer-name*	*customer-street*	*customer-city*
019-28-3746	Smith	North	Rye
182-73-6091	Turner	Putnam	Stamford
192-83-7465	Johnson	Alma	Palo Alto
244-66-8800	Curry	North	Rye
321-12-3123	Jones	Main	Harrison
335-57-7991	Adams	Spring	Pittsfield
336-66-9999	Lindsay	Park	Pittsfield
677-89-9011	Hayes	Main	Harrison
963-96-3963	Williams	Nassau	Princeton

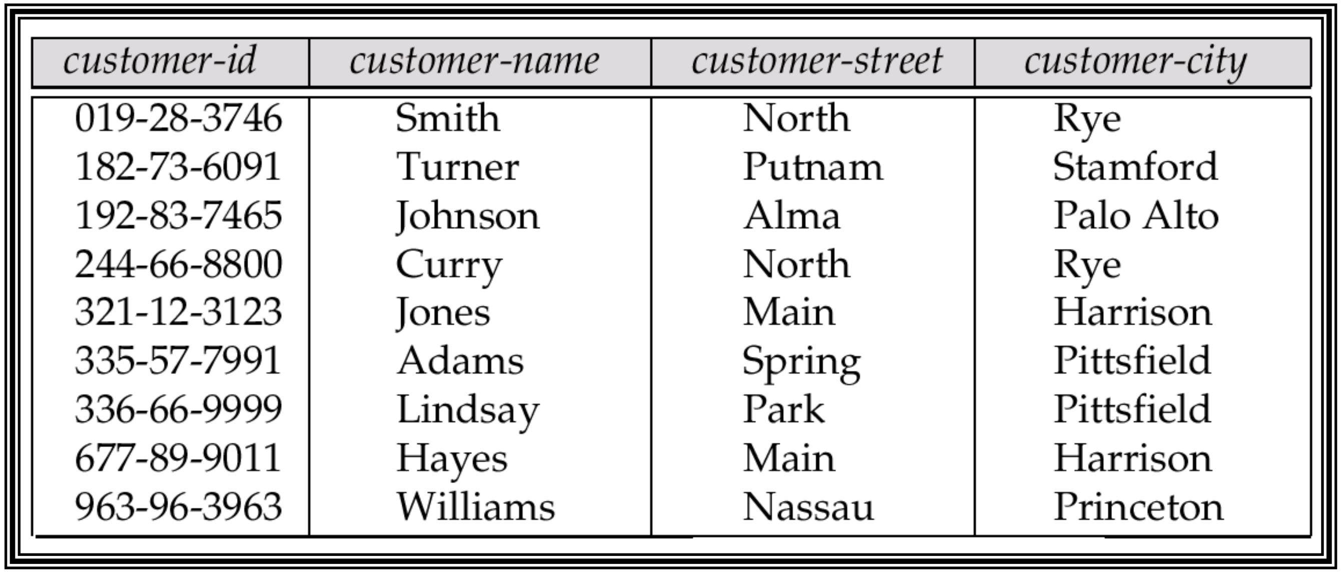
- SELECT * FROM CUSTOMER
  WHERE *customer-name* = ‘Smith’



customer-id	*customer-name*	*customer-street*	*customer-city*
019-28-3746	Smith	North	Rye

- WHERE gives a filtering condition.

SQL Example 5 (WHERE)



customer-id	*customer-name*	*customer-street*	*customer-city*
019-28-3746	Smith	North	Rye
182-73-6091	Turner	Putnam	Stamford
192-83-7465	Johnson	Alma	Palo Alto
244-66-8800	Curry	North	Rye
321-12-3123	Jones	Main	Harrison
335-57-7991	Adams	Spring	Pittsfield
336-66-9999	Lindsay	Park	Pittsfield
677-89-9011	Hayes	Main	Harrison
963-96-3963	Williams	Nassau	Princeton

- If we want to “find the tuples for customers living in Pittsfield”, what should be the SQL query?

- SELECT * FROM CUSTOMER
  WHERE *customer-city* = ‘Pittsfield’

SQL Example 6 (AND)

- Table schema:
  CUSTOMER(*customer-id*, *customer-name*, *customer-street*, *customer-city*)

- Find the names of the customers who are living in the ‘Park’ street in city ‘Pittsfield’.

- SELECT *customer-name*
  FROM CUSTOMER
  WHERE *customer-street* = ‘Park’ **AND** *customer-city* = ‘Pittsfield’

SQL Example 7 (OR)

- Table schema:
  CUSTOMER(*customer-id*, *customer-name*, *customer-street*, *customer-city*)

- Find the customer-id and names of the customers who are living in city ‘Pittsfield’ or ‘Rye’.

- SELECT *customer-id*, *customer-name*
  FROM CUSTOMER
  WHERE *customer-city* = ‘Pittsfield’ **OR** *customer-city* = ‘Rye’

SQL Example 8 (LIKE)

- Table schema:
  CUSTOMER(*customer-id*, *customer-name*, *customer-street*, *customer-city*)

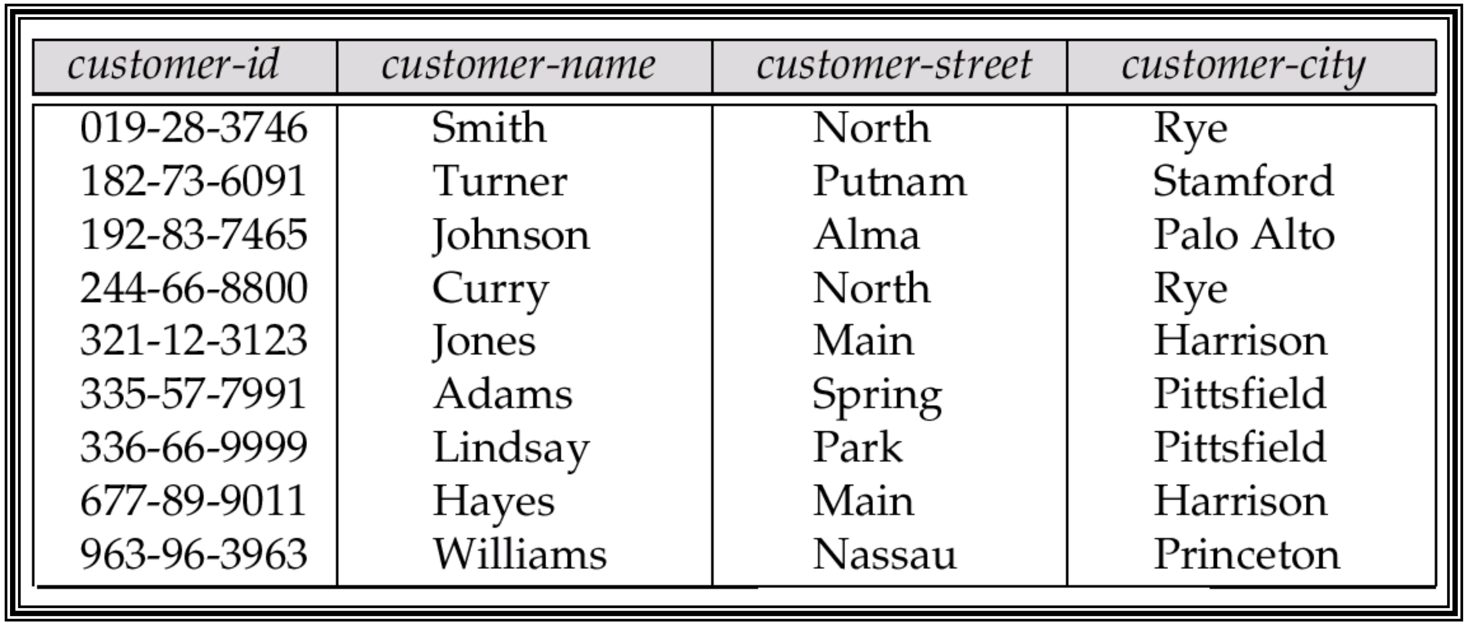
- Find the ids of the customers whose names start with the letter J and contain at least two letters.

- SELECT *customer-id*
  FROM CUSTOMER
  WHERE *customer-name* *LIKE 'J_%'*

- LIKE is used for string matching. `_' stands for any one character and `%' stands for 0 or more arbitrary characters.

Question 1

CUSTOMER



customer-id	*customer-name*	*customer-street*	*customer-city*
019-28-3746	Smith	North	Rye
182-73-6091	Turner	Putnam	Stamford
192-83-7465	Johnson	Alma	Palo Alto
244-66-8800	Curry	North	Rye
321-12-3123	Jones	Main	Harrison
335-57-7991	Adams	Spring	Pittsfield
336-66-9999	Lindsay	Park	Pittsfield
677-89-9011	Hayes	Main	Harrison
963-96-3963	Williams	Nassau	Princeton

1) Suppose that we want to “find the tuples for customers living in Palo Alto, what should be the SQL query?
2) Write the SQL query to only display the names of the customers living in Palo Alto.

Question 1

CUSTOMER

customer-id	*customer-name*	*customer-street*	*customer-city*
019-28-3746	Smith	North	Rye
182-73-6091	Turner	Putnam	Stamford
192-83-7465	Johnson	Alma	Palo Alto
244-66-8800	Curry	North	Rye
321-12-3123	Jones	Main	Harrison
335-57-7991	Adams	Spring	Pittsfield
336-66-9999	Lindsay	Park	Pittsfield
677-89-9011	Hayes	Main	Harrison
963-96-3963	Williams	Nassau	Princeton

3) Write the SQL query to find the names of the customers who are living on the ‘Alma’ street in city ‘Palo Alto’.

4) What is the result for the following SQL query?
SELECT *customer-id*
FROM CUSTOMER
WHERE *customer-street* LIKE ‘N%’ OR *customer-city* LIKE ‘H_%’

SQL Example 9 (ORDER BY)

- The previous queries do not have any ordering requirements.
- We can request ordered results using ‘ORDER BY’.
- SELECT *
  FROM ACC
  WHERE *balance* > 10000
  ORDER BY *balance*
- SELECT *
  FROM ACC
  WHERE *balance* > 10000
  ORDER BY *balance* DESC

ACC

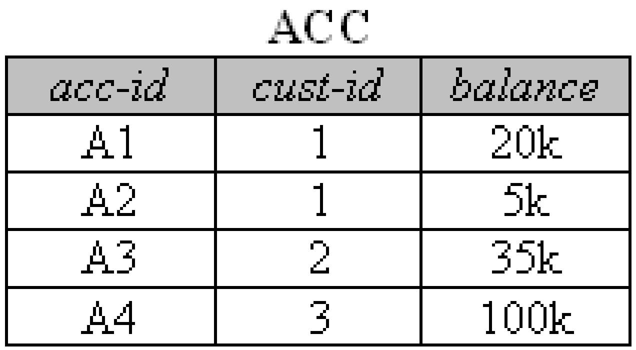
acc-id	*cust-id*	*balance*
A1	1	20k
A2	1	5k
A3	2	35k
A4	3	100k

acc-id	*cust-id*	*balance*
A1	1	20k
A3	2	35k
A4	3	100k

acc-id	*cust-id*	*balance*
A4	3	100k
A3	2	35k
A1	1	20k

SQL Example 10 (Arithmetic Expressions)

- Arithmetic expressions are allowed in SELECT and WHERE clauses
- SELECT balance*0.05 AS interest
  FROM ACC
  WHERE balance*1.05 < 20000
- What is the answer for this query?



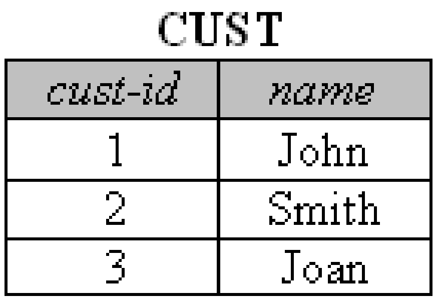
ACC

acc-id	cust-id	balance
A1	1	20k
A2	1	5k
A3	2	35k
A4	3	100k

SQL Example 11 (Cartesian Product)

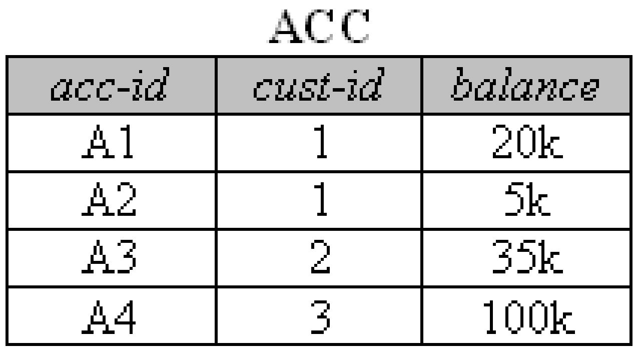
- In previous slides, all our queries retrieve information from a single table.

- Now let us consider two tables: CUST and ACC.



CUST

cust-id	*name*
1	John
2	Smith
3	Joan



ACC

acc-id	*cust-id*	*balance*
A1	1	20k
A2	1	5k
A3	2	35k
A4	3	100k

- What would the following query display?

```
SELECT *
FROM CUST, ACC
```

SQL Example 11 (Cartesian Product)

CUST

cust-id	*name*
1	John
2	Smith
3	Joan

ACC

acc-id	*cust-id*	*balance*
A1	1	20k
A2	1	5k
A3	2	35k
A4	3	100k

CUST. *cust-id*	CUST. *name*	ACC. *acc-id*	ACC. *cust-id*	ACC. *balance*
1	John	A1	1	20k
1	John	A2	1	5k
1	John	A3	2	35k
1	John	A4	3	100k
2	Smith	A1	1	20k
2	Smith	A2	1	5k
2	Smith	A3	2	35k
2	Smith	A4	3	100k
3	Joan	A1	1	20k
3	Joan	A2	1	5k
3	Joan	A3	2	35k
3	Joan	A4	3	100k

The result of this query:
SELECT *
FROM CUST, ACC

SQL Example 12 (Join)

CUST

cust-id	*name*
1	John
2	Smith
3	Joan

ACC

acc-id	*cust-id*	*balance*
A1	1	20k
A2	1	5k
A3	2	35k
A4	3	100k

- Write an SQL query to display, for each account, its acc-id and the name of its owner.
- We cannot answer this query using only one table.
- We need to do filtering and projection on the cartesian product.
- SELECT ACC.*acc-id*, CUST.*name*
  FROM CUST, ACC
  WHERE CUST.*cust-id* = ACC.*cust-id*

ACC. *acc-id*	CUST. *name*
A1	John
A2	John
A3	Smith
A4	Joan

SQL Example 12 (Join)

CUST

cust-id	*name*
1	John
2	Smith
3	Joan

ACC

acc-id	*cust-id*	*balance*
A1	1	20k
A2	1	5k
A3	2	35k
A4	3	100k

- The query can be further simplified as:

  SELECT *acc-id*, *name*
  FROM CUST, ACC
  WHERE CUST.*cust-id* = ACC.*cust-id*

ACC. *acc-id*	CUST. *name*
A1	John
A2	John
A3	Smith
A4	Joan

- No ambiguity can arise because CUST does not have ‘acc-id’ and ACC does not have ‘*name*’.

SQL Example 13 (INTERSECT)

CUST

cust-id	*name*
1	John
2	Smith
3	Joan
4	Mike

ACC

acc-id	*cust-id*	*balance*
A1	1	20k
A2	1	5k
A3	2	35k
A4	3	100k

- (SELECT *cust-id*
  FROM CUST)
  INTERSECT
  (SELECT *cust-id*
  FROM ACC)

cust-id
1
2
3

- INTERSECT automatically removes duplicates.

SQL Example 13 (INTERSECT)

CUST

cust-id	*name*
1	John
2	Smith
3	Joan
4	Mike

ACC

acc-id	*cust-id*	*balance*
A1	1	20k
A2	1	5k
A3	2	35k
A4	3	100k

- (SELECT *cust-id*, *name*
  FROM CUST)
  INTERSECT
  (SELECT *cust-id*
  FROM ACC)

- The above query is wrong!

- Because the two SELECT clauses return different sets of columns. Hence, intersection cannot be performed.

SQL Example 14 (UNION)

CUST

cust-id	*name*
1	John
2	Smith
3	Joan
4	Mike

ACC

acc-id	*cust-id*	*balance*
A1	1	20k
A2	1	5k
A3	2	35k
A4	3	100k

- (SELECT *cust-id*
  FROM CUST)
  UNION
  (SELECT *cust-id*
  FROM ACC)

cust-id
1
2
3
4

- UNION automatically removes duplicates.
- The two SELECT clauses must return the same columns.

SQL Example 15 (EXCEPT)

CUST

cust-id	*name*
1	John
2	Smith
3	Joan
4	Mike

ACC

acc-id	*cust-id*	*balance*
A1	1	20k
A2	1	5k
A3	2	35k
A4	3	100k

- (SELECT *cust-id*
  FROM CUST)
  EXCEPT
  (SELECT *cust-id*
  FROM ACC)

cust-id
4

- Returns the id that is in CUST but not in ACC.
- EXCEPT removes duplicates.
- The two SELECT must return the same columns.

Question 2

- Consider the following tables.

CUST

cust-id	*name*
1	John
2	Smith
3	Joan

ACC

acc-id	*cust-id*	*balance*
A1	1	20k
A2	1	5k
A3	2	35k
A4	3	100k

1) Write the SQL query to find the names of owners of accounts with balances of at least 30k.
2) Find the acc-ids of all accounts except the one with the largest balance.
   HINT: You can rename the table from NAME to N in the FROM clause by using the following SQL query.

   SELECT * FROM NAME N

What have we learned?

- SELECT: projection
- FROM: join
- WHERE: filtering, also called selection
- AND
- OR
- AS
- LIKE
- ORDER BY
- INTERSECTION
- UNION
- EXCEPT

Solution to Question 1

1)
```
SELECT *
FROM CUSTOMER
WHERE customer-city=‘Palo Alto’
```

2)
```
SELECT customer-name
FROM CUSTOMER
WHERE customer-city=‘Palo Alto’
```

Solution to Question 1

3) SELECT customer-name
FROM CUSTOMER
WHERE customer-street=‘Alma’ AND customer-city=‘Palo Alto’

4) The result is

customer-id
019-28-3746
244-66-8800
321-12-3123
677-89-9011
963-96-3963

Solution to Question 2

1)
```
SELECT CUST.name
FROM CUST, ACC
WHERE CUST.cust-id=ACC.cust-id
        AND ACC.balance>=30k
```

2)
```
SELECT DISTINCT T1.acc-id
FROM ACC T1, ACC T2
WHERE T1.balance < T2.balance
```