Structured Query Language (SQL)

Introduction to SQL

What is SQL?

- When a user wants to get some information from a database file, he can issue a ***query***.
- A query is a user–request to retrieve data or information with a certain condition.
- SQL is a query language that allows user to specify the conditions. (instead of algorithms)

Introduction to SQL

Concept of SQL

- A **user specifies a certain condition**.
- The program will go through all the records in the database file and select those records that satisfy the condition.(searching).
- Statistical information of the data.
- The result of the query will then be stored in form of a table.

Basic structure of an SQL query

General Structure	SELECT, ALL / DISTINCT, *, AS, FROM, WHERE
Comparison	IN, BETWEEN, LIKE "% _"
Grouping	GROUP BY, HAVING, COUNT( ), SUM( ), AVG( ), MAX( ), MIN( )
Display Order	ORDER BY, ASC / DESC
Logical Operators	AND, OR, NOT
Output	INTO TABLE / CURSOR TO FILE [ADDITIVE], TO PRINTER, TO SCREEN
Union	UNION

The Situation:
Student Particulars

field	type	width	contents
id	numeric	4	student id number
name	character	10	name
dob	date	8	date of birth
gender	character	1	M / F
class	character	2	class
hcode	character	1	house code: R, Y, B, G
dcode	character	3	district code
remission	logical	1	fee remission
mtest	numeric	2	Math test score

General Structure

SELECT ..... FROM ..... WHERE .....

SELECT [ALL / DISTINCT] *expr1* [AS *col1*], *expr2* [AS *col2*]

FROM *tablename* WHERE *condition;*

General Structure

SELECT [ALL / DISTINCT] *expr1* [AS *col1*], *expr2* [AS *col2*]
FROM ***tablename*** WHERE *condition;*

- The query will select rows from the source ***tablename*** and output the result in table form.
  - Expressions ***expr1***, ***expr2*** can be :
    - (1) a column, or
    - (2) an expression of functions and fields.
  - And ***col1***, ***col2*** are their corresponding column names in the output table.

General Structure

SELECT [ALL / DISTINCT] *expr1* [AS *col1*], *expr2* [AS *col2*]
FROM *tablename* WHERE *condition;*

- DISTINCT will eliminate duplication in the output while ALL will keep all duplicated rows.
- ***condition*** can be :
  - (1) an inequality, or
  - (2) a string comparison
  - using logical operators AND, OR, NOT.

General Structure

Before using SQL, open the student file:

```
USE student
```

eg. 1 List all the student records.

```
SELECT * FROM student;
```

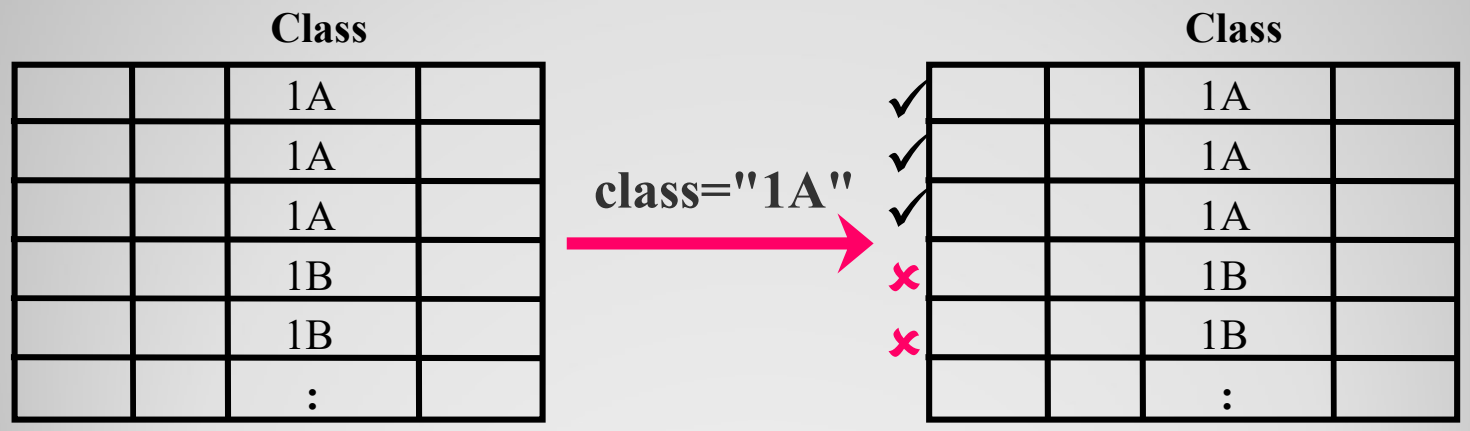
Result

id	name	dob	Gender	class	mtest	hcode	dcode	remission
9801	Peter	06-04-86	M	1A	70	R	SSP	.F.
9802	Mary	01-10-86	F	1A	92	Y	HHM	.F.
9803	Johnny	03-16-86	M	1A	91	G	SSP	.T.
9804	Wendy	07-09-86	F	1B	84	B	YMT	.F.
9805	Tobe	10-17-86	M	1B	88	R	YMT	.F.
:	:	:	:	:	:	:	:	:

General Structure

eg. 2 List the names and house code of 1A students.

```
SELECT name, hcode, class FROM student
WHERE class='1A';
```



General Structure

eg. 2 List the names and house code of 1A students.

Result

name	hcode	class
Peter	R	1A
Mary	Y	1A
Johnny	G	1A
Luke	G	1A
Bobby	B	1A
Aaron	R	1A
:	:	:

General Structure

eg. 3 List the residential district of the Red House members.

```
SELECT DISTINCT dcode FROM student
WHERE hcode='R';
```

Result

dcode
HHM
KWC
MKK
SSP
TST
YMT

General Structure

eg. 4 List the names and ages (1 d.p.) of 1B girls.

```
SELECT name, ROUND((DATE( )-dob)/365,1) AS age
FROM student WHERE class='1B' AND Gender='F';
```

Result

name	age
Wendy	12.1
Kitty	11.5
Janet	12.4
Sandy	12.3
Mimi	12.2

General Structure

eg. 5 List the names, id of 1A students with no fee remission.

```
SELECT name, id, class FROM student
WHERE class='1A' AND NOT remission;
```

Result

name	id	class
Peter	9801	1A
Mary	9802	1A
Luke	9810	1A
Bobby	9811	1A
Aaron	9812	1A
Ron	9813	1A
Gigi	9824	1A
:	:	:

Comparison

expr IN (*value1*, *value2*, *value3*)

expr BETWEEN *value1* AND *value2*

expr LIKE "%_"

Comparison

eg. 6 List the students who were born on Wednesday or Saturdays.

```
SELECT name, class, CDOW(dob) AS bdate
FROM student
WHERE DOW(dob) IN (4,7);
```

Result

name	class	bdate
Peter	1A	Wednesday
Wendy	1B	Wednesday
Kevin	1C	Saturday
Luke	1A	Wednesday
Aaron	1A	Saturday
:	:	:

Comparison

eg. 7 List the students who were not born in January, March, June, September.

```
SELECT name, class, dob FROM student
WHERE MONTH(dob) NOT IN (1,3,6,9);
```

Result

name	class	dob
Wendy	1B	07/09/86
Tobe	1B	10/17/86
Eric	1C	05/05/87
Patty	1C	08/13/87
Kevin	1C	11/21/87
Bobby	1A	02/16/86
Aaron	1A	08/02/86
:	:	:

Comparison

eg. 8 List the 1A students whose Math test score is between 80 and 90 (incl.)

```
SELECT name, mtest FROM student
WHERE class='1A' AND
mtest BETWEEN 80 AND 90;
```

Result

name	mtest
Luke	86
Aaron	83
Gigi	84

Comparison

eg. 9 List the students whose names start with "T".

```
SELECT name, class FROM student
WHERE name LIKE 'T%;
```

Result

name	class
Tobe	1B
Teddy	1B
Tim	2A

Comparison

eg. 10 List the Red house members whose names contain "a" as the 2nd letter.

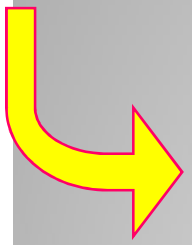
```
SELECT name, class, hcode FROM student
WHERE name LIKE '_a%' AND hcode='R';
```

Result

name	class	hcode
Aaron	1A	R
Janet	1B	R
Paula	2A	R

Grouping

SELECT ..... FROM ..... WHERE *condition*
GROUP BY *groupexpr* [HAVING *requirement*];



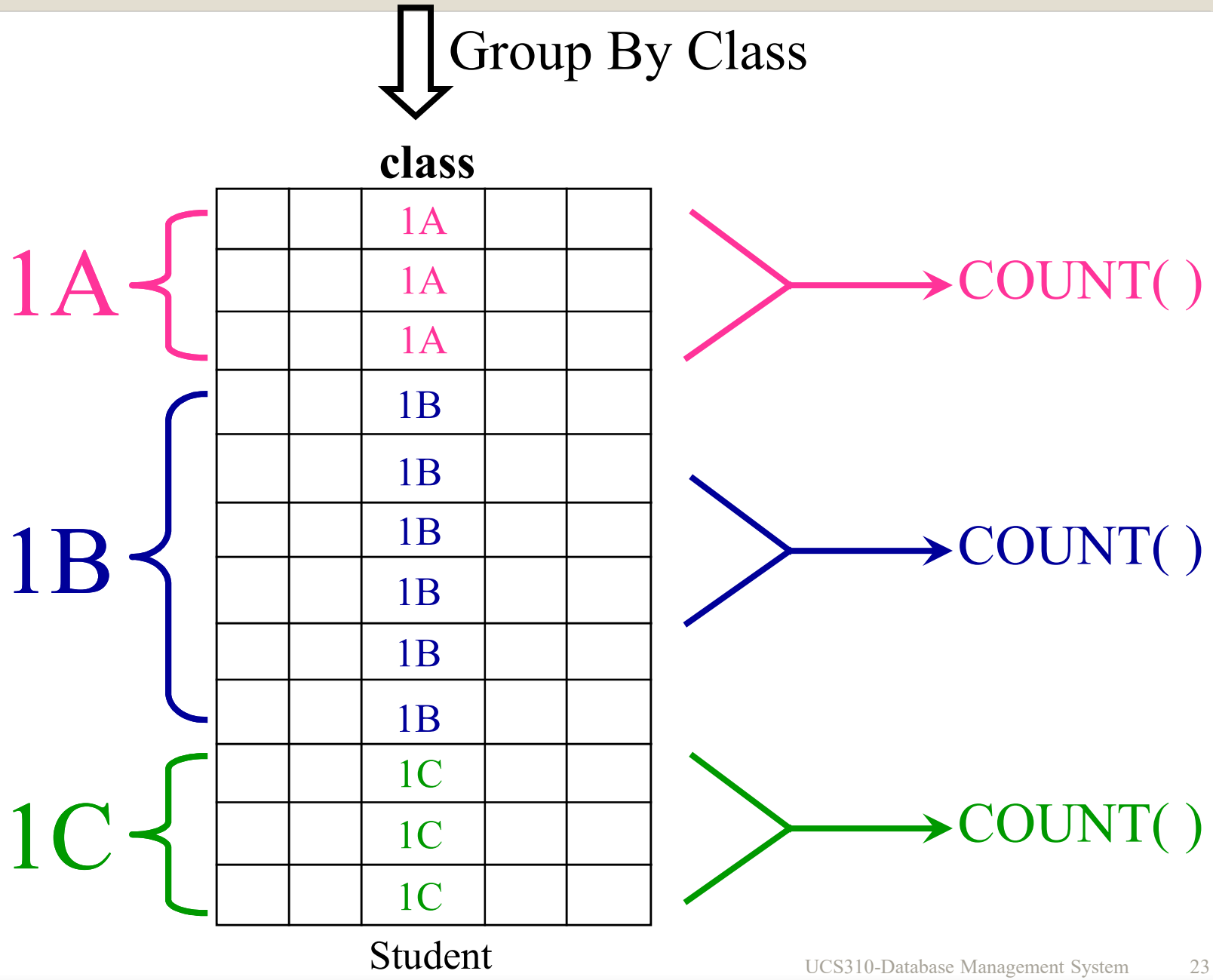
Group functions:

COUNT( ), SUM( ), AVG( ), MAX( ), MIN( )

- ***groupexpr*** specifies the related rows to be grouped as one entry. Usually it is a column.
- WHERE ***condition*** specifies the condition of individual rows before the rows are group. HAVING ***requirement*** specifies the condition involving the whole group.

Grouping

eg. 11 List the number of students of each class.


Group By Class
class
1A
1A
1A
1B
1B
1B
1B
1B
1B
1C
1C
1C
1A
1B
1C
COUNT( )
COUNT( )
COUNT( )
Student

Grouping

eg. 11 List the number of students of each class.

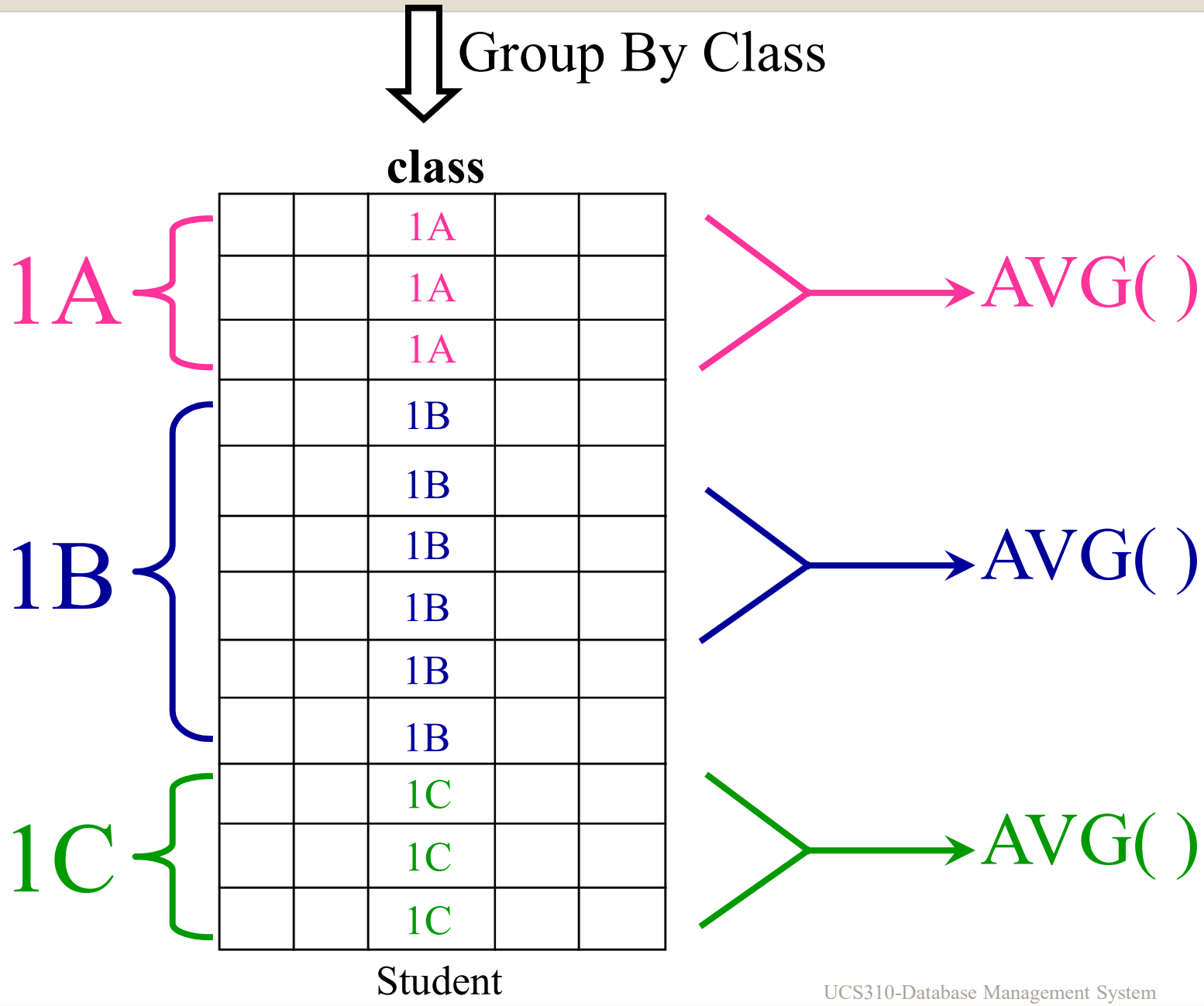
```
SELECT class, COUNT(*) FROM student
GROUP BY class;
```

Result

class	cnt
1A	10
1B	9
1C	9
2A	8
2B	8
2C	6

Grouping

eg. 12 List the average Math test score of each class.


Group By Class
class
1A
1A
1A
1B
1B
1B
1B
1B
1B
1C
1C
1C
1A
1B
1C
AVG( )
AVG( )
AVG( )
Student

Grouping

eg. 12 List the average Math test score of each class.

```
SELECT class, AVG(mtest) FROM student
GROUP BY class;
```

Result

class	avg_mtest
1A	85.90
1B	70.33
1C	37.89
2A	89.38
2B	53.13
2C	32.67

Grouping

eg. 13 List the number of girls of each district.

```
SELECT dcode, COUNT(*) FROM student;
WHERE Gender='F' GROUP BY dcode;
```

Result

dcode	cnt
HHM	6
KWC	1
MKK	1
SSP	5
TST	4
YMT	8

Grouping

eg. 14 List the max. and min. test score of Form 1 students of each district.

```
SELECT MAX(mtest), MIN(mtest), dcode
FROM student
WHERE class LIKE '1_' GROUP BY dcode;
```

Result

max_mtest	min_mtest	dcode
92	36	HHM
91	19	MKK
91	31	SSP
92	36	TST
75	75	TSW
88	38	YMT

Grouping

eg. 15 List the average Math test score of the boys in each class. The list should not contain class with less than 3 boys.

```
SELECT AVG(mtest), class FROM student
WHERE Gender='M' GROUP BY class
HAVING COUNT(*) >= 3;
```

Result

avg_mtest	class
86.00	1A
77.75	1B
35.60	1C
86.50	2A
56.50	2B

Display Order

SELECT ..... FROM ..... WHERE .....
GROUP BY ..... ;
ORDER BY *colname* ASC / DESC

Display Order

eg. 16 List the boys of class 1A, order by their names.

```
SELECT name, id FROM student
WHERE Gender='M' AND class='1A' ORDER BY name;
```

name	id
Peter	9801
Johnny	9803
Luke	9810
Bobby	9811
Aaron	9812
Ron	9813

name	id
Aaron	9812
Bobby	9811
Johnny	9803
Luke	9810
Peter	9801
Ron	9813

Display Order

eg. 17 List the 2A students by their residential district.

```
SELECT name, id, class, dcode FROM student
WHERE class='2A' ORDER BY dcode;
```

Result

name	id	class	dcode
Jimmy	9712	2A	HHM
Tim	9713	2A	HHM
Samual	9714	2A	SHT
Rosa	9703	2A	SSP
Helen	9702	2A	TST
Joseph	9715	2A	TSW
Paula	9701	2A	YMT
Susan	9704	2A	YMT

Display Order

eg. 18 List the number of students of each district (in desc. order).

```
SELECT COUNT(*) AS cnt, dcode FROM student
GROUP BY dcode ORDER BY cnt DESC;
```

Result

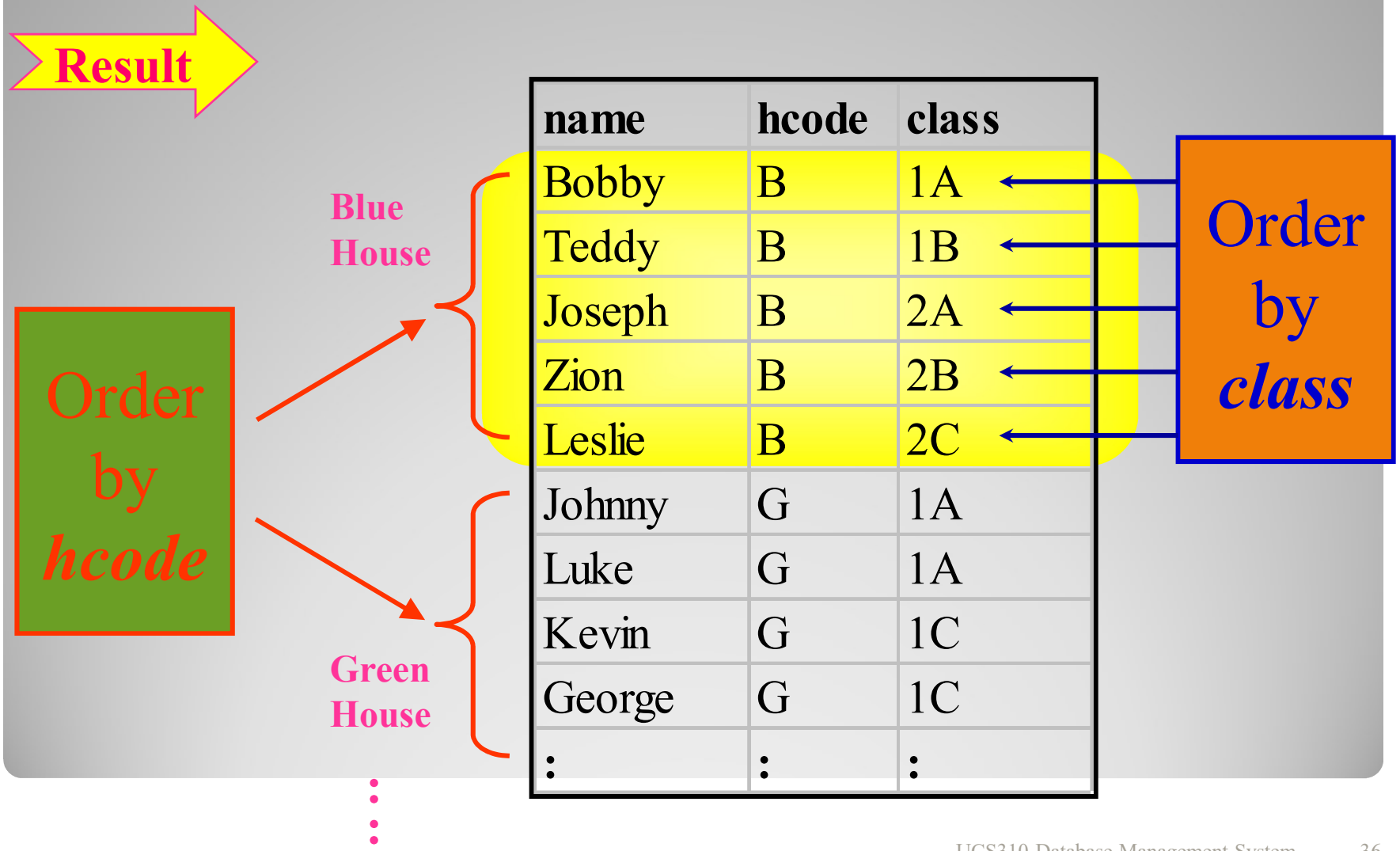
cnt	docode
11	YMT
10	HHM
10	SSP
9	MKK
5	TST
2	TSW
1	KWC
1	MMK
1	SHT

Display Order

eg. 19 List the boys of each house order by the classes. (2-level ordering)

```
SELECT name, class, hcode FROM student
WHERE Gender='M' ORDER BY hcode, class;
```

Display Order



Output

INTO TABLE *tablename*	the output table is saved as a database file in the disk.
INTO CURSOR *temp*	the output is stored in the working memory temporarily.
TO FILE *filename* [ADDITIVE]	output to a text file. (additive = append)
TO PRINTER	send to printer.
TO SCREEN	display on screen.

Output

eg. 20 List the students in desc. order of their names and save the result as a database file name.dbf.

```
SELECT * FROM student
ORDER BY name DESC INTO TABLE name.dbf;
```

Result

id	name	dob	Gender	class	mtest	hcode	dcode	remission
9707	Zion	07-29-85	M	2B	51	B	MKK	.F.
9709	Yvonne	08-24-85	F	2C	10	R	TST	.F.
9804	Wendy	07-09-86	F	1B	84	B	YMT	.F.
9819	Vincent	03-15-85	M	1C	29	Y	MKK	.F.
9805	Tobe	10-17-86	M	1B	88	R	YMT	.F.
9713	Tim	06-19-85	M	2A	91	R	HHM	.T.
9816	Teddy	01-30-86	M	1B	64	B	SSP	.F.
:	:	:	:	:	:	:	:	:

Output

eg. 21 Print the Red House members by their classes, sex and name.

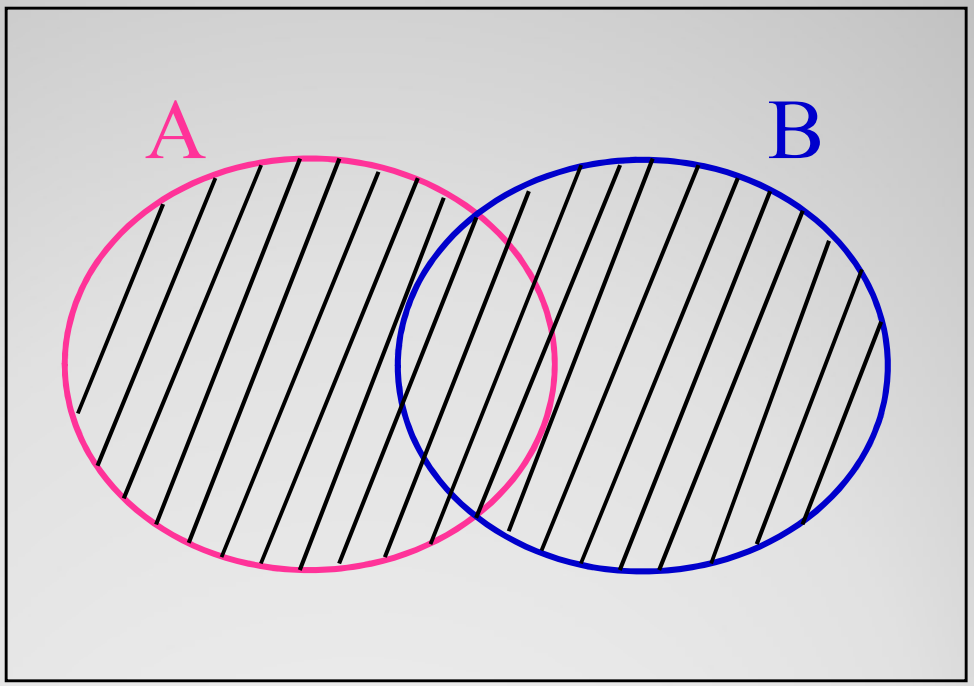
```
SELECT class, name, GenderFROM student
WHERE hcode='R'
ORDER BY class, GenderDESC, name TO PRINTER;
```

Result

class	name	Gdender
1A	Aaron	M
1A	Peter	M
1A	Ron	M
1B	Tobe	M
1B	Janet	F
1B	Kitty	F
1B	Mimi	F
:	:	:

Union, Intersection and Difference of Tables

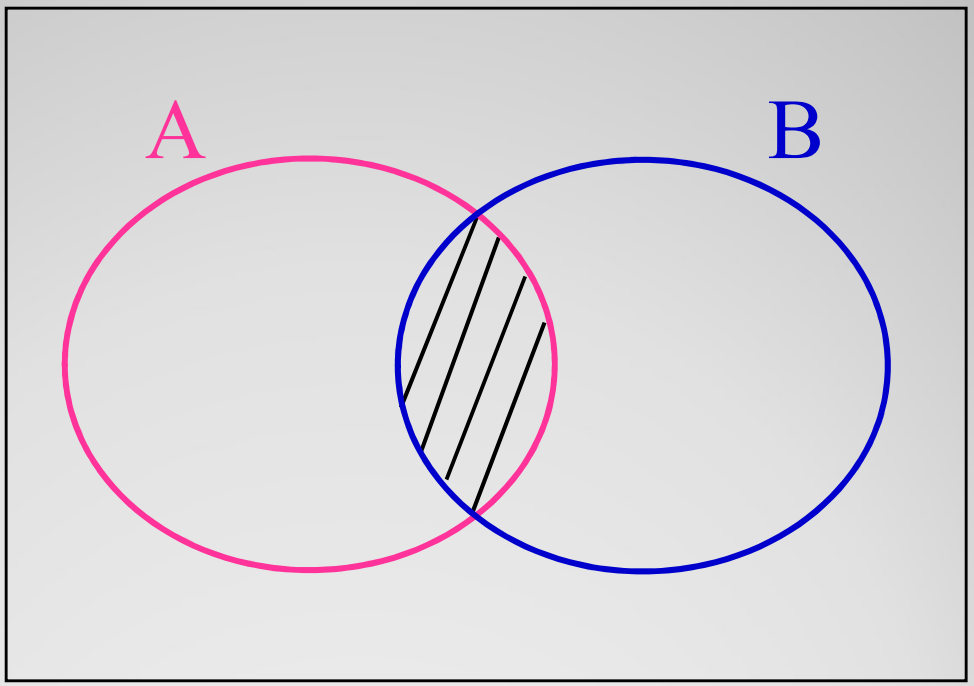
The ***union*** of A and B (A∪B)



A table containing all the rows from **A** and **B**.

Union, Intersection and Difference of Tables

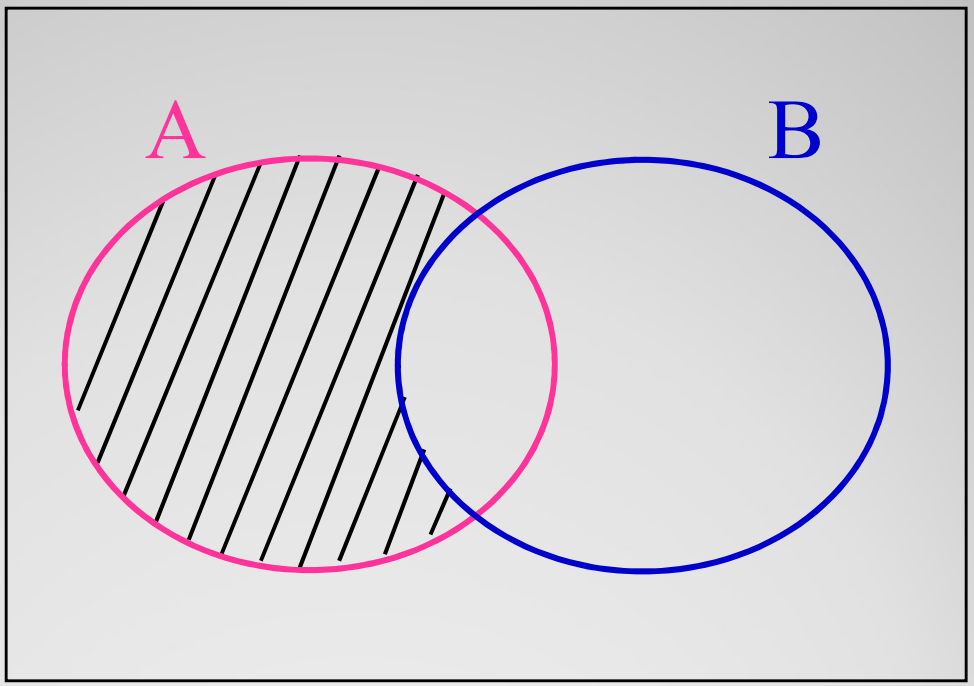
The *intersection* of A and B (A∩B)



A table containing only rows that appear in both **A** and **B**.

Union, Intersection and Difference of Tables

The ***difference*** of A and B (A–B)



A table containing rows that appear in **A** but not in **B**.

The Situation: Bridge Club & Chess Club

Consider the members of the Bridge Club and the Chess Club. The two database files have the same structure:

field	type	width	contents
id	numeric	4	student id number
name	character	10	name
Gender	character	1	M / F
class	character	2	class

Union, Intersection and Difference of Tables

Bridge [A]

	id	name	Gender	class
1	9812	Aaron	M	1A
2	9801	Peter	M	1A
3	9814	Kenny	M	1B
4	9806	Kitty	F	1B
5	9818	Edmond	M	1C
	:	:	:	:

Chess [B]

	id	name	Gender	class
1	9802	Mary	F	1A
2	9801	Peter	M	1A
3	9815	Eddy	M	1B
4	9814	Kenny	M	1B
5	9817	George	M	1C
	:	:	:	:

Before using SQL, open the two tables:

```
SELECT A
USE bridge
SELECT B
USE chess
```

Union, Intersection and Difference of Tables

```
SELECT ..... FROM ..... WHERE .....
UNION
SELECT ..... FROM ..... WHERE .....;
```

eg. 22 The two clubs want to hold a joint party. Make a list of all students. (Union)

Result

```
SELECT * FROM bridge
UNION
SELECT * FROM chess
ORDER BY class, name INTO TABLE party;
```

Union, Intersection and Difference of Tables

```
SELECT ..... FROM table1 ;
WHERE col IN  ( SELECT col FROM table2 )
```

eg. 23 Print a list of students who are members of both clubs. (Intersection)

Result

```
SELECT * FROM bridge
WHERE id IN ( SELECT id FROM chess )
TO PRINTER;
```

Union, Intersection and Difference of Tables

```
SELECT ..... FROM table1 ;
WHERE col NOT IN  ( SELECT col FROM table2 )
```

eg. 24 Make a list of students who are members of the Bridge Club but not Chess Club. (Difference)

Result

```
SELECT * FROM bridge
WHERE id NOT IN ( SELECT id FROM chess )
INTO TABLE diff;
```

Using a Subquery to Solve a Problem

Who has a salary greater than Abel's?

Main Query:

Which employees have salaries greater than Abel's salary?

Subquery

What is Abel's salary?

Subquery Syntax

```
SELECT    select_list
FROM      table
WHERE     expr operator
                        (SELECT       select_list
                        FROM          table);
```

- The subquery (inner query) executes once before the main query.
- The result of the subquery is used by the main query (outer query).

Using a Subquery

```
SELECT last_name
FROM   employees
WHERE  salary >
                (SELECT salary
                 FROM   employees
                 WHERE  last_name = 'Abel');
```

11000

LAST_NAME
King
Kochhar
De Haan
Hartstein
Higgins

Guidelines for Using Subqueries

- Enclose subqueries in parentheses.
- Place subqueries on the right side of the comparison condition.
- The `ORDER BY` clause in the subquery is not needed unless you are performing Top-N analysis.
- Use single-row operators with single-row subqueries and use multiple-row operators with multiple-row subqueries.

Types of Subqueries

- Single-row subquery

- Multiple-row subquery

Single-Row Subqueries

- Return only one row
- Use single-row comparison operators

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to

Executing Single-Row Subqueries

Display the details of those employees whose job-id is same as the job-id of employee_id 141 and salary more than the salary of employeed_id 143.

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  job_id =                    ST_CLERK
                (SELECT job_id
                 FROM   employees
                 WHERE  employee_id = 141)
AND    salary >                    2600
                (SELECT salary
                 FROM   employees
                 WHERE  employee_id = 143);
```

LAST_NAME	JOB_ID	SALARY
Rajs	ST_CLERK	3500
Davies	ST_CLERK	3100

Display the employee details whose salary is equal to the minimum salary

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  salary =
                (SELECT MIN(salary)
                 FROM   employees);
```

2500

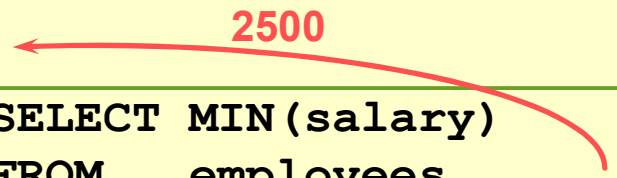
LAST_NAME	JOB_ID	SALARY
Vargas	ST_CLERK	2500

The HAVING Clause with Subqueries

- The Oracle server executes subqueries first.
- The Oracle server returns results into the HAVING clause of the main query.

Query: To display all the departments that have a minimum salary greater than that of the minimum salary of department 50.

```
SELECT    department_id, MIN(salary)
FROM      employees
GROUP BY  department_id
HAVING    MIN(salary) >
          (SELECT MIN(salary)
           FROM   employees
           WHERE  department_id = 50);
```



What is Wrong with this Statement?

```
SELECT employee_id, last_name
FROM   employees
WHERE  salary =
                (SELECT   MIN(salary)
                 FROM     employees
                 GROUP BY department_id);
```

```
ERROR at line 4:
ORA-01427: single-row subquery returns more than
one row
```

Single-row operator with multiple-row subquery

Will this Statement Return Rows?

```
SELECT last_name, job_id
FROM   employees
WHERE  job_id =
                (SELECT job_id
                 FROM   employees
                 WHERE  last_name = 'Haas');
```

```
no rows selected
```

Subquery returns no values

Multiple-Row Subqueries

- Queries that return more than one row from the inner SELECT statement.
-

Operator	Meaning
IN	Equal to any member in the list
ANY	Compare value to each value returned by the subquery
ALL	Compare value to every value returned by the subquery

Using the ANY Operator in Multiple-Row Subqueries

Display the details of those employees whose salary is less than any 'IT_Prog' and who are not 'IT_Prog'

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary   IN
                    (SELECT salary
                     FROM   employees
                     WHERE  job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```

9000, 6000, 4200

Harish	clerk	9000
Ramesh	salesman	6000
Ravi	clerk	4200
Amar	clerk	6000

...

10 rows selected.

Using the ANY Operator in Multiple-Row Subqueries

Display the details of those employees whose salary is less than any 'IT_Prog' and who are not 'IT_Prog'

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary < ANY
                (SELECT salary
                 FROM   employees
                 WHERE  job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```

9000, 6000, 4200

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
124	Mourgos	ST_MAN	5800
141	Rajs	ST_CLERK	3500
142	Davies	ST_CLERK	3100
143	Matos	ST_CLERK	2600
144	Vargas	ST_CLERK	2500

...

10 rows selected.

Using the ALL Operator in Multiple-Row Subqueries

Display the details of those employees whose salary is less than all 'IT_Prog' and who are not 'IT_Prog'

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary < ALL
                (SELECT salary
                 FROM   employees
                 WHERE  job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```

9000, 6000, 4200

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
141	Rajs	ST_CLERK	3500
142	Davies	ST_CLERK	3100
143	Matos	ST_CLERK	2600
144	Vargas	ST_CLERK	2500