

Lecture 8: SQL basic 2

BADM/ACCY 352

Spring 2017

Instructor: Yi Yang, PhD

Last lecture

Retrieve data from table

```
SELECT columns FROM table  
WHERE conditions;
```

Some common mistakes

- IS NULL

```
SELECT * FROM EMP WHERE EMP_MGR = NULL;
```

- Use AND/OR comparison.

```
SELECT COUNT(*) FROM EMP  
WHERE EMP_TITLE = 'Mrs.' or 'Ms.';
```

- SQL statement ends with semicolon ;
- Table name, column name is case-sensitive
- Single quotation mark for date and text

List only X rows: LIMIT

- Syntax:

```
SELECT columns FROM tables  
WHERE conditions  
LIMIT X;
```
- For example, to list only 5 rows of EMP table.

```
SELECT EMP_LNAME, EMP_FNAME FROM EMP  
LIMIT 5;
```

EMP LNAME	EMP FNAME
Kolmycz	George
Lewis	Rhonda
VanDam	Rhett
Jones	Anne
Lange	John

List only 5 male employees.

This is very useful when table is BIG.

Ordering: ORDER BY

- Order the result table by columns.

- Syntax:

```
SELECT columns FROM tables  
WHERE conditions  
ORDER BY columns ASC|DESC  
LIMIT X;
```

- For example, order the rows by employee's date of birth.

```
SELECT EMP_LNAME, EMP_FNAME, EMP_DOB FROM EMP  
ORDER BY EMP_DOB  
LIMIT 5;
```

EMP LNAME	EMP FNAME	EMP DOB
Kolmycz	George	1945-06-15 00:00:00
Brandon	Marie	1956-11-02 00:00:00
VanDam	Rhett	1958-11-14 00:00:00
Johnson	Edward	1961-05-14 00:00:00
Smith	George	1961-06-18 00:00:00

DESC or ASC

- ASC is the default order.

```
SELECT EMP_LNAME, EMP_FNAME, EMP_DOB FROM EMP  
ORDER BY EMP_DOB DESC  
LIMIT 5;
```

EMP_LNAME	EMP_FNAME	EMP_DOB
Williams	Robert	1975-03-14 00:00:00
Jones	Anne	1974-10-16 00:00:00
Diante	Jorge	1974-08-21 00:00:00
Saranda	Hermine	1972-07-25 00:00:00
Lange	John	1971-11-08 00:00:00

SQL clinic

- List employee whose last name is Smith, and order by their DOB.

```
select * from EMP  
order by EMP_DOB  
where EMP_LNAME = 'Smith';
```


Syntax Error!!!

```
select * from EMP  
where EMP_LNAME = 'Smith'  
order by EMP_DOB;
```

Order BY multiply columns

- The results are ordered by the first column, then the second, and so on for as many columns as the ORDER BY clause includes.

Column1	Column2
=====	=====
1	Smith
2	Jones
1	Anderson
3	Andrews



Column1	Column2
=====	=====
1	Anderson
1	Smith
2	Jones
3	Andrews

```
SELECT Column1, Column2 FROM thedata ORDER BY Column1, Column2
```

Return all male employees from EMP table
ordered by last name, then first name

Listing Unique Values: DISTINCT

In a table, a column may contain many duplicate values; and sometimes you only want to list the different (distinct) values.

EMP AREACODE
615
615
901
615
901
615
615
615
615
901
901
615
615
615
901
615
615

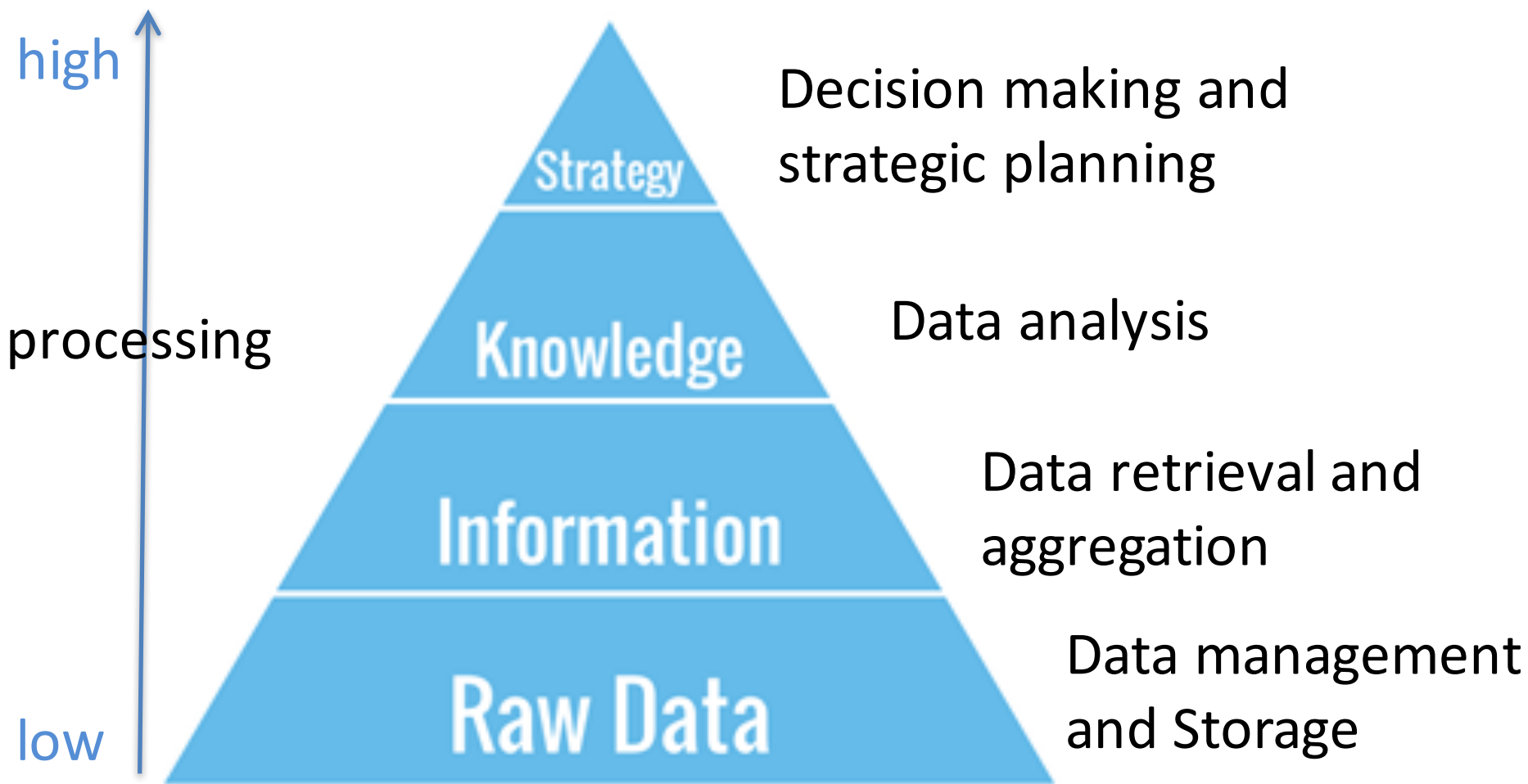
Syntax:

```
SELECT DISTINCT column_name  
FROM table;
```

```
SELECT DISTINCT EMP_AREACODE  
FROM EMP;
```



EMP AREACODE
615
901



MySQL Aggregate functions

- The data that you need is not always stored in the tables. However, you can get it by performing the calculations on the selected data.
- An aggregate function performs a calculation on a set of values and returns a single value.
- COUNT() is an aggregate function.

FUNCTION	OUTPUT
COUNT	The number of rows containing non-null values
MIN	The minimum attribute value encountered in a given column
MAX	The maximum attribute value encountered in a given column
SUM	The sum of all values for a given column
AVG	The arithmetic mean (average) for a specified column

COUNT

- We have learned COUNT(*), which shows the number of rows of the returned data.
- NULL values will not be counted.

```
SELECT P_CODE, V_CODE FROM PRODUCT;
```

P CODE	V CODE
11QER/31	25595
13-Q2/P2	21344
14-Q1/L3	21344
1546-QQ2	23119
1558-QW1	23119
2232/QTY	24288
2232/QWE	24288
2238/QPD	25595
23109-HB	21225
23114-AA	NULL
54778-2T	21344
89-WRE-Q	24288
PVC23DRT	NULL
SM-18277	21225
SW-23116	21231
WR3/TT3	25595

```
SELECT COUNT(*) FROM PRODUCT;
```



16

```
SELECT COUNT(V_CODE) FROM PRODUCT;
```



14

iClicker question

Assume

```
SELECT DISTINCT V_CODE  
FROM PRODUCT;
```

V_CODE
NULL
21225
21231
21344
23119
24288
25595

What does the following command return?

```
SELECT COUNT(DISTINCT V_CODE )  
FROM PRODUCT;
```

- a). 7 b). 6 c). 0 d). NULL

MAX and MIN

- Return the maximum and minimum value of a column in a table.
- What is highest price in the PRODUCT table?

```
SELECT MAX(P_PRICE) FROM PRODUCT;
```



MAX(P PRICE)
256.99

Just like ORDER BY, MAX and MIN can also be applied to String and Date

```
SELECT MAX(EMP_LNAME) FROM EMP;
```

```
SELECT MIN(EMP_DOB) FROM EMP;
```

Exercise

- Can you return the highest price from PRODUCT table *without* using MAX function?

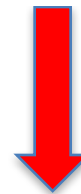
Now we can retrieve the maximum price in Product table, but *which product has that maximum price?*

P CODE	P PRICE
11QER/31	109.99
13-Q2/P2	14.99
14-Q1/L3	17.49
1546-QQ2	39.95
1558-QW1	43.99
2232/QTY	109.92
2232/QWE	99.87
2238/QPD	38.95
23109-HB	9.95
23114-AA	14.40
54778-2T	4.99
89-WRE-Q	256.99
PVC23DRT	5.87
SM-18277	6.99
SW-23116	8.45
WR3/TT3	119.95

```
SELECT P_CODE, P_PRICE FROM PRODUCT  
ORDER BY P_PRICE DESC  
LIMIT 1;
```



```
SELECT P_CODE, MAX(P_PRICE)  
FROM PRODUCT;
```



P CODE	MAX(P PRICE)
11QER/31	256.99



- If I know the maximum product price is 256.99, can you return the products?

```
SELECT P_CODE, P_PRICE FROM PRODUCT  
WHERE P_PRICE = 256.99;
```

- Now replace 256.99 with

```
SELECT MAX(P_PRICE) FROM PRODUCT;
```

```
SELECT P_CODE, P_PRICE FROM PRODUCT  
WHERE P_PRICE = ( SELECT MAX(P_PRICE) FROM PRODUCT );
```

RESULTS:

P_CODE	P_PRICE
89-WRE-Q	256.99

subquery



Subquery

- A subquery (inner query) is a query (SELECT statement) inside another query (outer query);
- A subquery is expressed inside parentheses.
- The inner query is executed first.
- The output of the inner query is used as the input for the outer query

SQL clinic

```
SELECT * FROM PRODUCT  
WHERE MAX(P_PRICE);
```



```
SELECT * FROM PRODUCT  
WHERE P_PRICE = MAX(P_PRICE);
```



```
SELECT P_CODE, P_PRICE FROM PRODUCT  
WHERE P_PRICE = ( SELECT MAX(P_PRICE) FROM PRODUCT );
```

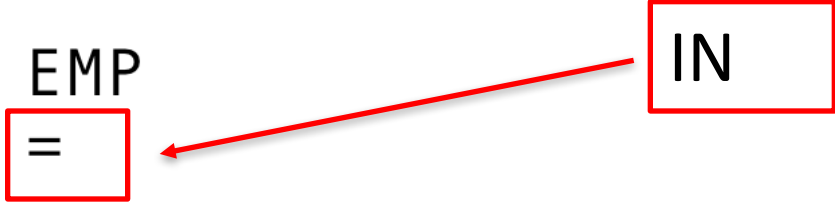
Subquery exercise

- List the youngest employee from EMP table.
- How many employees does Mr. Williams manage?

Subquery

- If a subquery returns a set of values instead of a single value, use IN.

```
SELECT * FROM EMP
where EMP_MGR =
(
  SELECT EMP_NUM FROM EMP
  WHERE EMP_LNAME = 'Williams'
  OR EMP_LNAME = 'Kolmycz'
)
;
```



A red arrow points from the word 'IN' (which is enclosed in a red box) to the equals sign (=) in the SQL query. The equals sign is also enclosed in a red box. This illustrates the replacement of a single-value comparison with a set-value comparison using the IN operator.

SUM and AVG

- As its name suggested, SUM function computes the total sum for the specified attribute.
- AVG function computes the average value for the specified attribute.
- Both SUM and AVG **only compute a numeric column.**
- There are STD(), VARIANCE() as well.

SUM and AVG

```
SELECT SUM(P_PRICE) FROM PRODUCT;
```

902.74

```
SELECT COUNT(P_PRICE) FROM PRODUCT;
```

16

```
SELECT AVG(P_PRICE) FROM PRODUCT;
```

56.42

iClicker question

Say you want to list all products whose price is above average price. Which query is correct?

A `SELECT * FROM PRODUCT
WHERE P_PRICE > (select AVG(*) FROM PRODUCT);`

B `SELECT * FROM PRODUCT
WHERE P_PRICE > (select AVG(P_PRICE) FROM PRODUCT);`

C `SELECT * FROM PRODUCT
WHERE P_PRICE > AVG(P_PRICE);`

Sum up

ORDER BY, LIMIT

Aggregate functions and Subquery

Exercise

Explain what does this query do?

```
SELECT P_CODE, P_PRICE FROM PRODUCT  
WHERE P_PRICE > ( SELECT AVG(P_PRICE) FROM PRODUCT)  
ORDER BY P_PRICE DESC;
```

P CODE	P DESCRIPT	P PRICE	V CODE
11QER/31	Power painter, 15 psi., 3-nozzle	109.99	25595
13-Q2/P2	7.25-in. pwr. saw blade	14.99	21344
14-Q1/L3	9.00-in. pwr. saw blade	17.49	21344
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	39.95	23119
1558-QW1	Hrd. cloth, 1/2-in., 3x50	43.99	23119
2232/QTY	B&D jigsaw, 12-in. blade	109.92	24288
2232/QWE	B&D jigsaw, 8-in. blade	99.87	24288
2238/QPD	B&D cordless drill, 1/2-in.	38.95	25595
23109-HB	Claw hammer	9.95	21225
23114-AA	Sledge hammer, 12 lb.	14.40	NULL
54778-2T	Rat-tail file, 1/8-in. fine	4.99	21344
89-WRE-Q	Hicut chain saw, 16 in.	256.99	24288
PVC23DRT	PVC pipe, 3.5-in., 8-ft	5.87	NULL
SM-18277	1.25-in. metal screw, 25	6.99	21225
SW-23116	2.5-in. wd. screw, 50	8.45	21231
WR3/TT3	Steel matting, 4'x8'x1/6", .5" ...	119.95	25595

How to compute the average product price **per vendor?**

emp_bonus

Employee	Bonus
A	1000
B	2000
A	500
C	700
B	1250

How to compute the total bonus **per employee?**

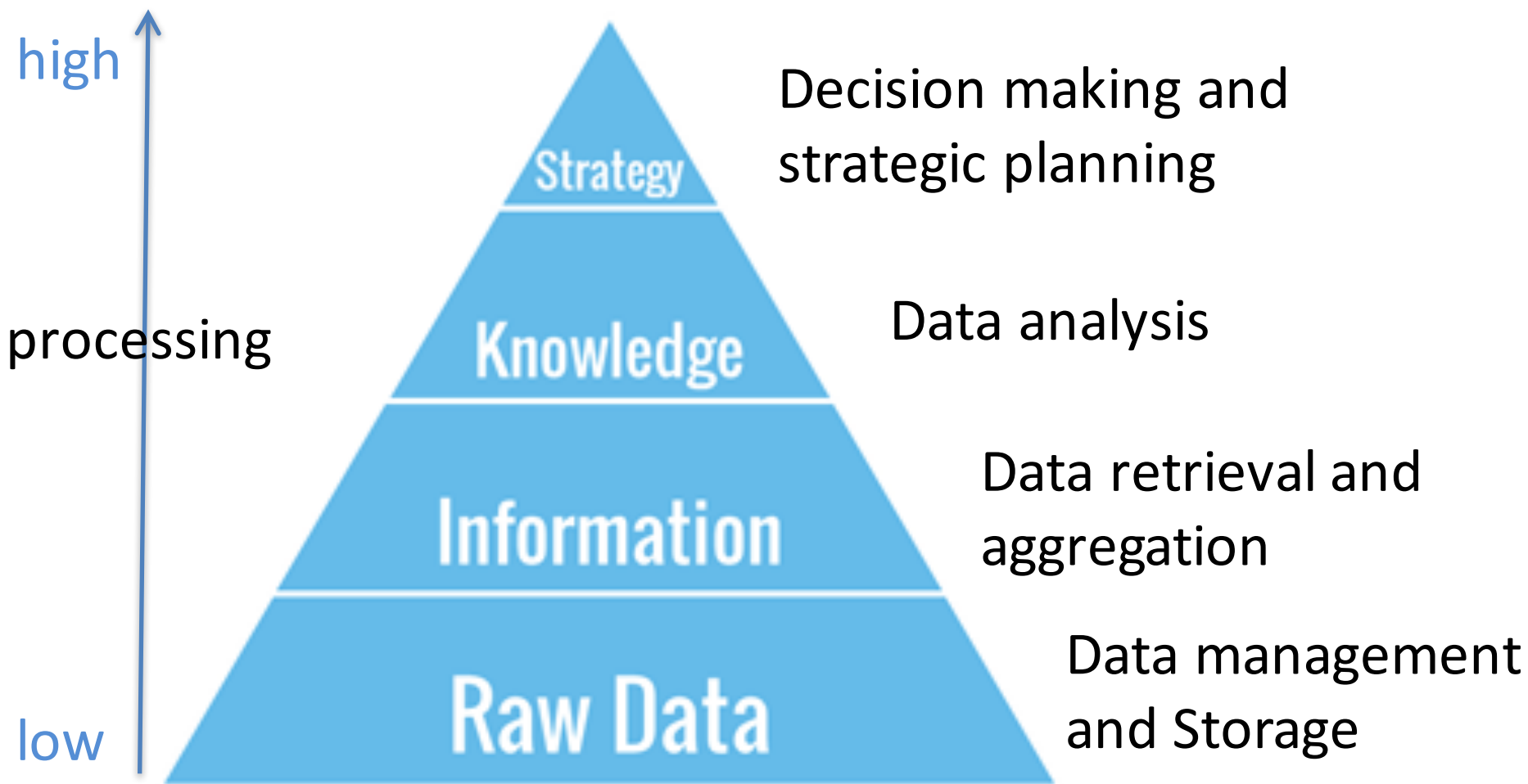
GROUPING DATA: GROUP BY

- Sometimes we need to group rows into smaller collections.
- SQL can group certain rows by a column, and each unique value of the column is a group.
- Syntax:

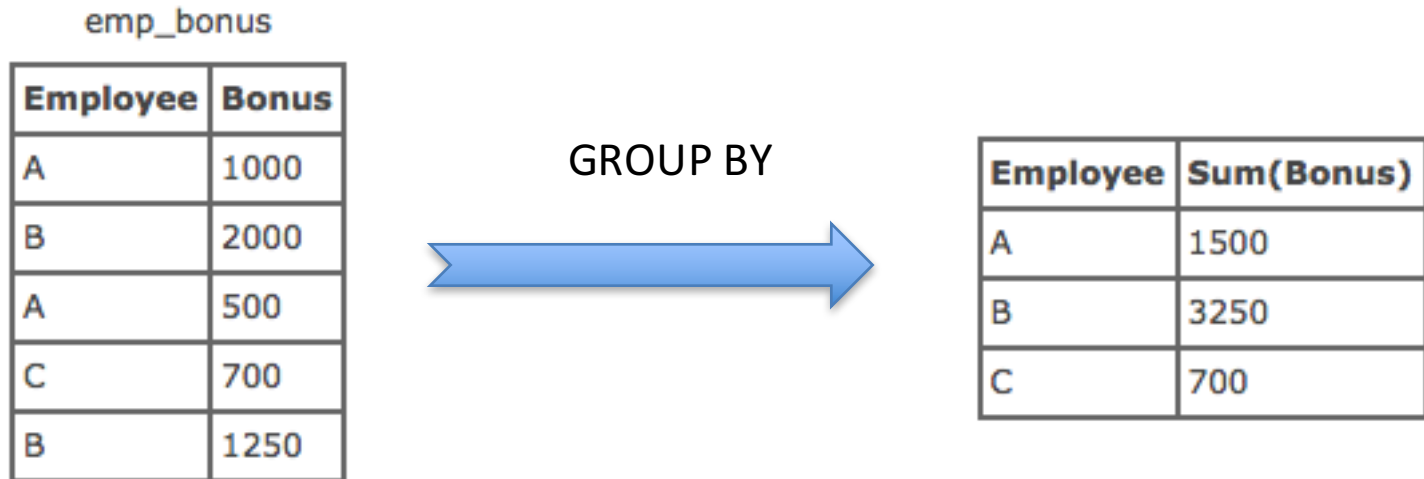
```
SELECT column_name, aggregate_funcation(column_name)
FROM table_name
WHERE conditions
GROUP BY column_name
HAVING conditions;
```

Group By with Aggregate functions is very useful

- For a sales database, you want to know the total sale revenue per day, month, quarter, year, or per store, region.
- For a student database, you want to know student GPA per semester.
- For a stock database, you want to know the stock return per month, quarter, year.



GROUP BY with aggregate function example



```
SELECT employee, SUM(bonus)
FROM emp_bonus
GROUP BY employee;
```

P CODE	P DESCRIPT	P PRICE	V CODE
11QER/31	Power painter, 15 psi., 3-nozzle	109.99	25595
13-Q2/P2	7.25-in. pwr. saw blade	14.99	21344
14-Q1/L3	9.00-in. pwr. saw blade	17.49	21344
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	39.95	23119
1558-QW1	Hrd. cloth, 1/2-in., 3x50	43.99	23119
2232/QTY	B&D jigsaw, 12-in. blade	109.92	24288
2232/QWE	B&D jigsaw, 8-in. blade	99.87	24288
2238/QPD	B&D cordless drill, 1/2-in.	38.95	25595
23109-HB	Claw hammer	9.95	21225
23114-AA	Sledge hammer, 12 lb.	14.40	NULL
54778-2T	Rat-tail file, 1/8-in. fine	4.99	21344
89-WRE-Q	Hicut chain saw, 16 in.	256.99	24288
PVC23DRT	PVC pipe, 3.5-in., 8-ft	5.87	NULL
SM-18277	1.25-in. metal screw, 25	6.99	21225
SW-23116	2.5-in. wd. screw, 50	8.45	21231
WR3/TT3	Steel matting, 4'x8'x1/6", .5" ...	119.95	25595

What if you want to compute the average product price per vendor?

```
SELECT V_CODE, AVG(P_PRICE) FROM PRODUCT
GROUP BY V_CODE;
```

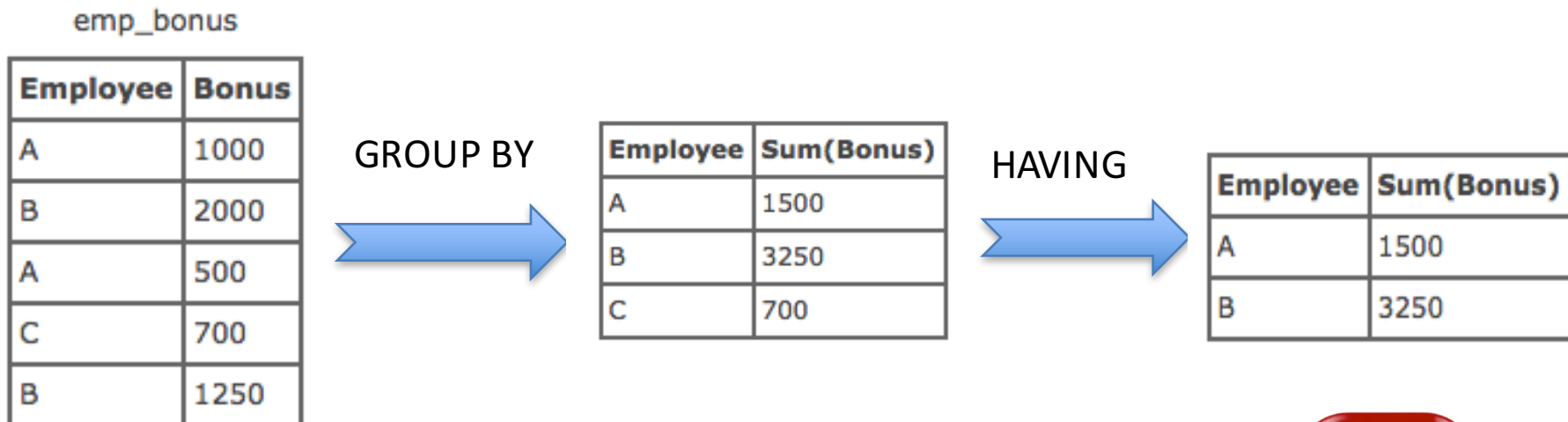
RESULTS:

V CODE	AVG(P PRICE)
NULL	10.135000
21225	8.470000
21231	8.450000
21344	12.490000
23119	41.970000
24288	155.593333
25595	89.630000

NOTE the first row

HAVING vs. WHERE

- HAVING is used with aggregate functions to apply constraint conditions.
- WHERE can't be used with aggregate functions.



```
SELECT employee, SUM(bonus) FROM emp_bonus  
GROUP BY employee WHERE SUM(bonus) > 1000;
```

```
SELECT employee, SUM(bonus) FROM emp_bonus  
GROUP BY employee HAVING SUM(bonus) > 1000;
```



Let's walk through an example

INV_NUMBER	CUS_CODE	INV_DATE
1001	10014	2014-01-16 00:00:00
1002	10011	2014-01-16 01:00:00
1003	10012	2014-01-16 01:00:00
1004	10011	2014-01-17 00:00:00
1005	10018	2014-01-17 01:00:00
1006	10014	2014-01-17 02:00:00
1007	10015	2014-01-17 02:00:00
1008	10011	2014-01-17 02:00:00

1. List all invoices in the INVOICE table.
2. List all invoices that are after '2014-01-17'.
3. Count the total number of invoices per customer, and the invoices satisfy the condition 'after 2014-01-17'.
4. List the cus_code who has more than one invoices after 2014-01-17.

#1

```
SELECT * FROM INVOICE;
```

#2

```
SELECT * FROM INVOICE WHERE INV_DATE >= '2014-01-17';
```

#3

```
SELECT CUS_CODE, COUNT(*) FROM INVOICE  
WHERE INV_DATE >= '2014-01-17'  
GROUP BY CUS_CODE;
```

#4

```
SELECT CUS_CODE, COUNT(*) FROM INVOICE  
WHERE INV_DATE >= '2014-01-17'  
GROUP BY CUS_CODE  
HAVING COUNT(*) > 1;
```

Put it together

```
SELECT columns  
FROM table  
[WHERE condition]  
[GROUP BY column]  
[HAVING condition]  
[ORDER BY column [ASC | DESC]]  
[LIMIT count]
```