



The lecture will start soon at  
**08:30**



Aalto University  
School of Business

# MySQL for Data Analytics

*Lecturer: Yong Liu*

*Contact me at: [Yong.liu@aalto.fi](mailto:Yong.liu@aalto.fi)*

# Content

- **If / case when**
- **Join**
- **Update columns/variables**
- **Table (Advance)**
- **Table vs. View**

# IF

- **The IF() function returns a value if a condition is TRUE, or another value if a condition is FALSE.**
- **IF(expression, expr\_true, expr\_false);**

# Examples

- **SELECT IF(500<1000, 5, 10) as result;**

Result #1 (1×1)	
result	
5	

- **SELECT IF(500<1000, IF(100 > 50, 1,0), 10)**  
**as result;**

Result #1 (1×1)	
result	
1	

# Examples (II)

- **SELECT** productCode, quantityOrdered,  
**IF**(quantityOrdered>30, "Large", "Small")  
as category  
**FROM** orderdetails;



 orderNumber	 productCode	quantityOrdered
10,107	S10_1678	30
10,121	S10_1678	34
10,134	S10_1678	41
10,145	S10_1678	45
10,159	S10_1678	49
10,168	S10_1678	36
10,180	S10_1678	29
10,188	S10_1678	48
10,201	S10_1678	22
10,211	S10_1678	41
10,223	S10_1678	37

Table: orderdetails

 productCode	quantityOrdered	category
S18_1749	30	Small
S18_2248	50	Large
S18_4409	22	Small
S24_3969	49	Large
S18_2325	25	Small
S18_2795	26	Small
S24_1937	45	Large
S24_2022	46	Large
S18_1342	39	Large

Output of query

ZIP_code	State	Submitted_via	Data_received	Data_sent_to_company	Company	Company_response
55,008	MN	Referral	2011-12-02	2011-12-02	GE Capital Retail	Closed without relief
10,065	NY	Referral	2011-12-02	2011-12-02	HSBC	Closed without relief
98,362	WA	Referral	2011-12-02	2011-12-02	HSBC	Closed without relief
80,537	CO	Referral	2011-12-02	2011-12-02	HSBC	Closed without relief
45,458	OH	Referral	2011-12-02	2011-12-02	JPMorgan Chase	Closed without relief
48,462	MI	Referral	2011-12-02	2011-12-02	JPMorgan Chase	Closed with relief
60,645	IL	Referral	2011-12-02	2011-12-02	JPMorgan Chase	Closed with relief
11,418	NY	Referral	2011-12-02	2011-12-02	JPMorgan Chase	Closed with relief

By comparing the Data\_received and Data\_sent\_to\_company, we say that if the difference between two dates are more than 3 days, it is in “slow procedure” and otherwise in “fast procedure”. We defined this “slow procedure” and “fast procedure” as a new variable of “category”.

Please identify the numbers of complaints that companies received in correspondence to “slow procedure” and “fast procedure”.

Please order the results by category and number of complaints.

# Answer

Select

```
if(datediff(Data_sent_to_company, Data_received)>3,  
    'Slow Procedure', 'Fast Procedure') as category,  
company, count(*) as fre  
from cfpb_complaints_2500  
group by category, company  
order by category, fre desc
```

cfpb_complaints_2500 (3x8)		
category	company	fre
Fast Procedure	Bank of America	399
Fast Procedure	JPMorgan Chase	239
Fast Procedure	Citibank	200
Fast Procedure	Wells Fargo	195
Fast Procedure	Capital One	167
Slow Procedure	Bank of America	182
Slow Procedure	JPMorgan Chase	92
Slow Procedure	Wells Fargo	81



# Case when

**CASE** *expression*

**WHEN** *condition1* **THEN** result1

**WHEN** *condition2* **THEN** result2

...

**WHEN** conditionN **THEN** resultN

**ELSE** result

**END**

# Examples

**SELECT**

**CASE**

**WHEN** 2<3 **THEN** 'this is true'

**ELSE** 'this is false'

**END;**

# Examples (1)

```
SELECT orderNumber, quantityOrdered,  
CASE  
WHEN quantityOrdered > 30 THEN "the quantity is over 30"  
WHEN quantityOrdered = 30 THEN "the quantity is 30"  
ELSE "the quantity is less than 30"  
END  
from orderdetails;
```

orderdetails (3x2,996)		
orderNumber	quantityOrdered	CASE
10,100	30	the quantity is 30
10,100	50	the quantity is over 30
10,100	22	the quantity is less than 30
10,100	49	the quantity is over 30
10,101	25	the quantity is less than 30
10,101	26	the quantity is less than 30

orderNumber	A Z ↓ productCode	priceEach
10,107	S10_1678	81.35
10,121	S10_1678	86.13
10,134	S10_1678	90.92
10,145	S10_1678	76.56
10,159	S10_1678	81.35
10,168	S10_1678	94.74
10,180	S10_1678	76.56
10,188	S10_1678	95.7
10,201	S10_1678	82.3
10,211	S10_1678	90.92
10,223	S10_1678	80.39
10,237	S10_1678	91.87
10,251	S10_1678	93.79
10,263	S10_1678	89
10,275	S10_1678	81.35
10,285	S10_1678	95.7
10,299	S10_1678	76.56
10,309	S10_1678	94.74
10,318	S10_1678	84.22
10,329	S10_1678	80.39
10,341	S10_1678	84.22
10,354	S10_1678	84.22
10,361	S10_1678	92.83
10,375	S10_1678	76.56

## Examples (2)

Please calculate the numbers of different products that have been sold in a price that is i) less than 70, ii) between 70 and 90, and iii) above 90, respectively.

# Examples (2)

**SELECT**

**CASE**

**WHEN** priceEach < 70 **THEN** "the price is less than 70"

**WHEN** priceEach **between** 70 **and** 90 **THEN** "the price is between 70 and 90"

**ELSE** "the price is over 90"

**END** **as** category, **count**(**distinct** productCode)

**from** orderdetails **group by** category;

Result #1 (2×3)	
category	count(distinct productCode)
the price is between 70 and 90	40
the price is less than 70	46
the price is over 90	60

# Examples (3)

customerNumber	checkNumber	paymentDate	amount
103	HQ336336	2004-10-19	6,066.78
103	JM555205	2003-06-05	14,571.44
103	OM314933	2004-12-18	1,676.14
112	B0864823	2004-12-17	14,191.12
112	HQ55022	2003-06-06	32,641.98
112	ND748579	2004-08-20	33,347.88
114	GG31455	2003-05-20	45,864.03
114	MA765515	2004-12-15	82,261.22
114	NP603840	2003-05-31	7,565.08
114	NR27552	2004-03-10	44,894.74
119	DB933704	2004-11-14	19,501.82
119	LN373447	2004-08-08	47,924.19
119	NG94694	2005-02-22	49,523.67
121	DB889831	2003-02-16	50,218.95
121	FD317790	2003-10-28	1,491.38
121	KI831359	2004-11-04	17,876.32
121	MA302151	2004-11-28	34,638.14
124	AE215433	2005-03-05	101,244.59
124	BG255406	2004-08-28	85,410.87
124	CQ287967	2003-04-11	11,044.3
124	ET64396	2005-04-16	83,598.04
124	HI366474	2004-12-27	47,142.7
124	HR86578	2004-11-02	55,639.66

Your boss asked you to compute the frequency of payment against each year for each customer in order to see the trend, what you would do?



customerNumber	2003	2004	2005	2006
103	1	2	0	0
112	1	2	0	0
114	2	2	0	0
119	0	2	1	0
121	2	2	0	0
124	3	4	2	0
128	2	2	0	0
129	2	1	0	0
131	1	2	0	0
141	4	6	3	0
144	1	1	0	0
145	1	3	0	0

Output: a Cross Table

```

SELECT customerNumber,
case when YEAR(paymentDate) = '2003' then 1 ELSE 0 end AS '2003',
case when YEAR(paymentDate) = '2004' then 1 ELSE 0 end AS '2004',
case when YEAR(paymentDate) = '2005' then 1 ELSE 0 end AS '2005',
case when YEAR(paymentDate) = '2006' then 1 ELSE 0 end AS '2006' FROM
payments GROUP BY customerNumber

```

customerNumber	2003	2004	2005	2006
103	0	1	0	0
112	0	1	0	0
114	1	0	0	0
119	0	1	0	0
121	1	0	0	0
124	0	0	1	0
128	1	0	0	0
129	0	1	0	0
131	1	0	0	0
141	1	0	0	0
144	0	1	0	0
145	0	1	0	0

How to improve the code  
to generate the right  
output?

```

SELECT customerNumber,
SUM(case when YEAR(paymentDate) = '2003' then 1 ELSE 0 end) AS '2003',
SUM(case when YEAR(paymentDate) = '2004' then 1 ELSE 0 end) AS '2004',
SUM(case when YEAR(paymentDate) = '2005' then 1 ELSE 0 end) AS '2005',
SUM(case when YEAR(paymentDate) = '2006' then 1 ELSE 0 end) AS '2006'
FROM payments GROUP BY customerNumber

```

customerNumber	2003	2004	2005	2006
103	1	2	0	0
112	1	2	0	0
114	2	2	0	0
119	0	2	1	0
121	2	2	0	0
124	3	4	2	0
128	2	2	0	0
129	2	1	0	0
131	1	2	0	0
141	4	6	3	0
144	1	1	0	0
145	1	3	0	0
146	1	2	0	0

### Key takeaway:

The output of “case when” function can be used by another function.

This is a very useful skill of apply “case when” function!

Can you use “if” function to replace “case when” function to obtain the same results?



```
SELECT customerNumber,  
SUM(case when YEAR(paymentDate) = '2003' then 1 ELSE 0 end) AS '2003',  
SUM(case when YEAR(paymentDate) = '2004' then 1 ELSE 0 end) AS '2004',  
SUM(case when YEAR(paymentDate) = '2005' then 1 ELSE 0 end) AS '2005',  
SUM(case when YEAR(paymentDate) = '2006' then 1 ELSE 0 end) AS '2006'  
FROM payments GROUP BY customerNumber
```

```
SELECT customerNumber,  
SUM(If(YEAR(paymentDate) = '2003',1 ,0)) AS '2003',  
SUM(If(YEAR(paymentDate) = '2004',1 ,0)) AS '2004',  
SUM(If(YEAR(paymentDate) = '2005',1 ,0)) AS '2005',  
SUM(If(YEAR(paymentDate) = '2006',1 ,0)) AS '2006' FROM payments GROUP  
BY customerNumber
```

# Question




customerNumber	2003	2004	2005	2006
103	14,571.44	7,742.92	0.00	0.00
112	32,641.98	47,539.00	0.00	0.00
114	53,429.11	127,155.96	0.00	0.00
119	0.00	67,426.01	49,523.67	0.00
121	51,710.33	52,514.46	0.00	0.00
124	167,783.08	231,562.53	184,842.63	0.00
128	34,650.82	41,286.94	0.00	0.00
129	40,461.78	26,248.78	0.00	0.00
131	22,292.62	85,347.32	0.00	0.00
141	189,840.15	293,765.51	232,133.32	0.00
144	7,674.94	36,005.71	0.00	0.00
145	53,959.21	53,487.29	0.00	0.00
146	39,712.10	90,593.25	0.00	0.00



If we want to obtain the results shown in the left table (the sum of payment to the company per year per customer), how to write/modify the code?

## What to change?

```
SELECT customerNumber,  
SUM(If(YEAR(paymentDate) = '2003',1 ,0)) AS '2003',  
SUM(If(YEAR(paymentDate) = '2004',1 ,0)) AS '2004',  
SUM(If(YEAR(paymentDate) = '2005',1 ,0)) AS '2005',  
SUM(If(YEAR(paymentDate) = '2006',1 ,0)) AS '2006' FROM payments GROUP  
BY customerNumber
```

# How join the values of two tables?

 customerNumber	customerName	contactLastName	 contactFirstName	 phone
103	Atelier graphique	Schmitt	Carine	40.32.2555
112	Signal Gift Stores	King	Jean	7025551838
114	Australian Collectors, Co.	Ferguson	Peter	03 9520 4555
119	La Rochelle Gifts	Labrune	Janine	40.67.8555
121	Baane Mini Imports	Bergulfsen	Jonas	07-98 9555
124	Mini Gifts Distributors Ltd.	Nelson	Susan	4155551450
125	Havel & Zbyszek Co	Piestrzeniewicz	Zbyszek	(26) 642-7555
128	Blauer See Auto, Co.	Keitel	Roland	+49 69 66 90 2555
129	Mini Wheels Co.	Murphy	Julie	6505555787
131	Land of Toys Inc.	Lee	Kwai	2125557818
141	Euro+ Shopping Channel	Freyre	Diego	(91) 555 94 44
144	Volvo Model Replicas, Co	Berglund	Christina	0921-12 3555
145	Danish Wholesale Imports	Petersen	Jytte	31 12 3555
146	Saveley & Henriot, Co.	Saveley	Mary	78.32.5555
148	Dragon Souvenirs, Ltd.	Natividad	Eric	+65 221 7555
151	Muscle Machine Inc	Young	Jeff	2125557413

 customerNumber	 checkNumber	paymentDate	amount
103	HQ336336	2004-10-19	6,066.78
103	JM555205	2003-06-05	14,571.44
103	OM314933	2004-12-18	1,676.14
112	BO864823	2004-12-17	14,191.12
112	HQ55022	2003-06-06	32,641.98
112	ND748579	2004-08-20	33,347.88
114	GG31455	2003-05-20	45,864.03
114	MA765515	2004-12-15	82,261.22
114	NP603840	2003-05-31	7,565.08
114	NR27552	2004-03-10	44,894.74
119	DB933704	2004-11-14	19,501.82
119	LN373447	2004-08-08	47,924.19
119	NG94694	2005-02-22	49,523.67
121	DB889831	2003-02-16	50,218.95

**Business question:** provide contact information of customers pertinent to their payment

# JOIN

- Template

**Select** attributes (e.g. **tb1.columnX**, **tb2.ColumnY**)

**from** **tb1 inner join** **tb2**

---

**on** (**tb1.column1 = tb2.columnA** **and**  
**tb1.column2 = tb2.columnB**

...)

**Where** conditions;

🔑 customerNumber	customerName	contactLastName	phone	creditLimit	🌱 salesRepEmployeeNumber
<b>103</b>	Atelier graphique	Schmitt	40.32.2555	21,000	1,370
<b>112</b>	Signal Gift Stores	King	7025551838	71,800	1,166
<b>114</b>	Australian Collectors, Co.	Ferguson	03 9520 4555	117,300	1,611
<b>119</b>	La Rochelle Gifts	Labrune	40.67.8555	118,200	1,370
<b>121</b>	Baane Mini Imports	Bergulfsen	07-98 9555	81,700	1,504

Table: customers

🔑 customerNumber	🔑 checkNumber	paymentDate	amount
<b>103</b>	<b>HQ336336</b>	2004-10-19	6,066.78
<b>103</b>	<b>JM555205</b>	2003-06-05	14,571.44
<b>103</b>	<b>OM314933</b>	2004-12-18	1,676.14
<b>112</b>	<b>BO864823</b>	2004-12-17	14,191.12
<b>112</b>	<b>HQ55022</b>	2003-06-06	32,641.98

Table: payments

**Question: How can we retrieve the contact information of those who made a payment over 100. 000, including their payment amount?**

**SELECT** \*  
**FROM** customers  
**INNER JOIN** payments **ON**  
customers.customerNumber = payments.customerNumber  
**WHERE** payments.amount > 100000



\* here covers  
the all  
columns from  
both tables

Use  
tbName.colName  
to specify a  
column in a table

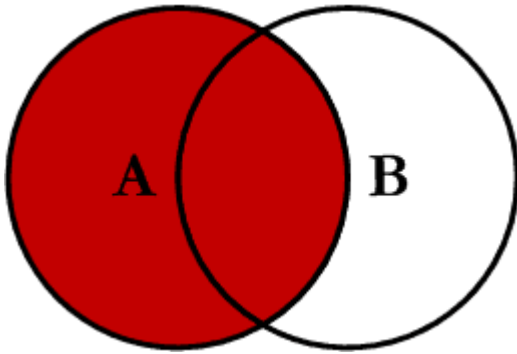
Specify the  
table Name  
before column  
name

# If columns are selected from both tables

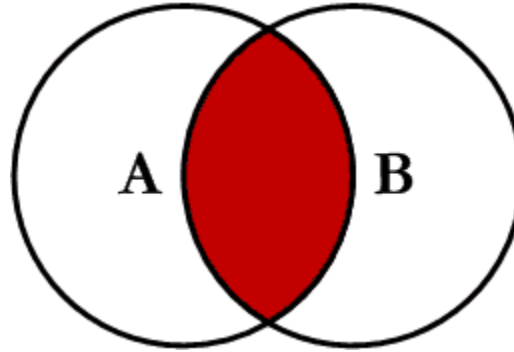
```
SELECT payments.customerNumber, payments.paymentDate,  
customers.creditLimit, customers.salesRepEmployeeNumber  
FROM customers  
INNER JOIN payments ON  
customers.customerNumber = payments.customerNumber  
WHERE payments.amount > 100000
```

 customerNumber	paymentDate	creditLimit	 salesRepEmployeeNumber
124	2005-03-05	210,500	1,165
124	2003-08-15	210,500	1,165
141	2004-12-31	227,600	1,370
141	2005-03-18	227,600	1,370
148	2003-12-26	103,800	1,621

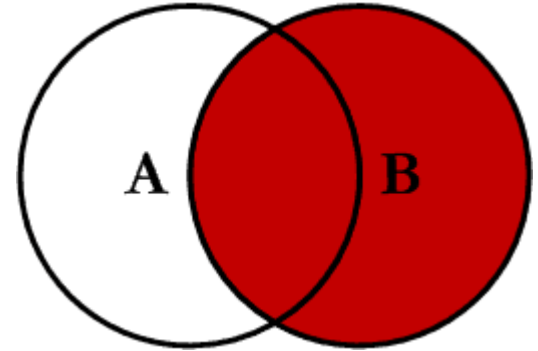
# left join, right join, inner join



Left join



Inner join

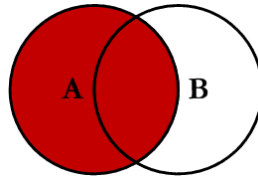


Right join

In MySQL, **JOIN**, **CROSS JOIN**, and **INNER JOIN** are syntactic equivalents (they can replace each other).



select \* from A  
left join B  
on A.A\_ID = B.B\_ID



A_ID	AWord	B_ID	BWord
1	A_1	1	B_1
2	A_2	2	B_2
3	A_3	3	B_3
4	A_4	(NULL)	(NULL)
5	A_5	(NULL)	(NULL)
6	A_6	(NULL)	(NULL)

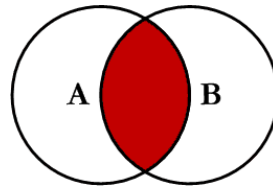
Left table A

A_ID	AWord
1	A_1
2	A_2
3	A_3
4	A_4
5	A_5
6	A_6

Right table B

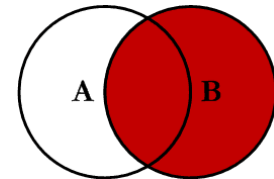
B_ID	BWord
1	B_1
2	B_2
3	B_3
7	B_7
8	B_8
9	B_9

select \* from A  
Inner join B  
on A.A\_ID = B.B\_ID



Result #1 (4x3)			
A_ID	AWord	B_ID	BWord
1	A_1	1	B_1
2	A_2	2	B_2
3	A_3	3	B_3

select \* from A  
Right join B  
on A.A\_ID = B.B\_ID



Result #1 (4x6)			
A_ID	AWord	B_ID	BWord
1	A_1	1	B_1
2	A_2	2	B_2
3	A_3	3	B_3
(NULL)	(NULL)	7	B_7
(NULL)	(NULL)	8	B_8
(NULL)	(NULL)	9	B_9

🔑 customerNumber	🔑 checkNumber	▲ paymentDate	amount
363	IS232033	2003-01-16	10223.83
128	DI925118	2003-01-28	10549.01
181	GQ132144	2003-01-30	5494.78
121	DB889831	2003-02-16	50218.95
145	JJ246391	2003-02-20	53959.21
141	JN722010	2003-02-25	40206.2
278	GP636783	2003-03-02	52151.81
385	EK785462	2003-03-09	51001.22
131	CL442705	2003-03-12	22292.62
486	JB117768	2003-03-20	25833.14

Table payments

## Question

- How to add customer's first and last name, and phone number to the table **payments**?


🔑 customerNumber	▲ contactLastName	contactFirstName	phone
406	Perrier	Dominique	(1) 47.55.6555
311	Koskitalo	Pirkko	981-443655
299	Klaeboe	Jan	+47 2212 1555
456	Choi	Yu	2125551957
181	Frick	Michael	2125551500
424	Hernandez	Maria	2125558493
131	Lee	Kwai	2125557818

Table customers

```
alter table payments add column contactLastName varchar(25);  
alter table payments add column contactFirstName varchar(25);  
alter table payments add column phone varchar(25);
```

```
update payments join customers  
on payments.customerNumber = customers.customerNumber  
set payments.contactLastName = customers.contactLastName,  
payments.contactFirstName = customers.contactFirstName,  
payments.phone = customers.phone
```

```
update payments t1 join customers t2  
on t1.customerNumber = t2.customerNumber  
set t1.contactLastName = t2.contactLastName,  
t1.contactFirstName = t2.contactFirstName,  
t1.phone = t2.phone
```



**“payments t1”  
means t1 is  
alias of the  
table payment**

# Both works, but 'on' is recommended

```
update payments join customers
on payments.customerNumber = customers.customerNumber
set payments.contactLastName = customers.contactLastName,
payments.contactFirstName = customers.contactFirstName,
payments.phone = customers.phone
```

```
update payments join customers
set payments.contactLastName = customers.contactLastName,
payments.contactFirstName = customers.contactFirstName,
payments.phone = customers.phone
where payments.customerNumber = customers.customerNumber
```

# Self Join

employeeNumber	lastName	firstName	reportsTo	extension	email	officeCode	jobTitle
1,337	Bondur	Loui	1,102	x6493	lbondur@classic...	4	Sales Rep
1,102	Bondur	Gerard	1,056	x5408	gbondur@class...	4	Sale Manager (EMEA)
1,501	Bott	Larry	1,102	x2311	lbott@classicm...	7	Sales Rep
1,143	Bow	Anthony	1,056	x5428	abow@classic...	1	Sales Manager (NA)
1,401	Castillo	Pamela	1,102	x2759	pcastillo@class...	4	Sales Rep
1,076	Firrelli	Jeff	1,002	x9273	jfirrelli@classic...	1	VP Marketing
1,188	Firrelli	Julie	1,143	x2173	jfirrelli@classic...	2	Sales Rep
1,611	Fixter	Andy	1,088	x101	afixter@classic...	6	Sales Rep
1,702	Gerard	Martin	1,102	x2312	mgerard@class...	4	Sales Rep
1,370	Hernandez	Gerard	1,102	x2028	ghernande@cla...	4	Sales Rep
1,165	Jennings	Leslie	1,143	x3291	ljennings@clas...	1	Sales Rep
1,504	Jones	Barry	1,102	x102	bjones@classic...	7	Sales Rep
1,625	Kato	Yoshimi	1,621	x102	ykato@classic...	5	Sales Rep
1,619	King	Tom	1,088	x103	tking@classicm...	6	Sales Rep
1,612	Marsh	Peter	1,088	x102	pmarsh@classi...	6	Sales Rep
1,002	Murphy	Diane	(NULL)	x5800	dmurphy@clas...	1	President

Please show who is report to who [names] in the table.

```

SELECT
CONCAT(m.lastname,', ',m.firstname) AS Manager,
CONCAT(e.lastname,', ',e.firstname) AS 'Report to'
FROM employees e
INNER JOIN employees m
ON m.employeeNumber = e.reportsTo
ORDER BY manager;

```

employeeNumber	lastName	firstName	reportsTo
1,337	Bondur	Loui	1,102
1,102	Bondur	Gerard	1,056
1,501	Bott	Larry	1,102
1,143	Bow	Anthony	1,056
1,401	Castillo	Pamela	1,102
1,076	Firrelli	Jeff	1,002
1,188	Firrelli	Julie	1,143
1,611	Fixter	Andy	1,088

Result #1 (2x22)	
Manager	Report to
Bondur, Gerard	Castillo, Pamela
Bondur, Gerard	Hernandez, Gerard
Bondur, Gerard	Bondur, Loui
Bondur, Gerard	Gerard, Martin
Bondur, Gerard	Jones, Barry
Bondur, Gerard	Bott, Larry
Bow, Anthony	Patterson, Steve
Bow, Anthony	Firrelli, Julie
Bow, Anthony	Thompson, Leslie
Bow, Anthony	Jennings, Leslie
Bow, Anthony	Vanauf, George
Bow, Anthony	Tseng, Foon Yue
Murphy, Diane	Firrelli, Jeff
Murphy, Diane	Patterson, Mary
Nishi, Mami	Kato, Yoshimi
Patterson, Mary	Patterson, William

 employeeNumber	lastName	firstName
1,002	Murphy	Diane
1,056	Patterson	Mary
1,076	Firrelli	Jeff
1,088	Patterson	William
1,102	Bondur	Gerard
1,143	Bow	Anthony
1,165	Jennings	Leslie
1,166	Thompson	Leslie
1,188	Firrelli	Julie
1,216	Patterson	Steve
1,286	Tseng	Foon Yue
1,323	Vanauf	George
1,337	Bondur	Loui

 reportsTo	lastName	firstName
(NULL)	Murphy	Diane
1,002	Patterson	Mary
1,002	Firrelli	Jeff
1,056	Patterson	William
1,056	Bondur	Gerard
1,056	Bow	Anthony
1,143	Jennings	Leslie
1,143	Thompson	Leslie
1,143	Firrelli	Julie
1,143	Patterson	Steve
1,143	Tseng	Foon Yue
1,143	Vanauf	George
1,102	Bondur	Loui
1,102	Hernandez	Gerard



Select employeeNumber, lastName,  
firstName from employees;

Select reportsTo, lastName, firstName  
from employees;

# Reflection

id	uid
1	a
1	b
1	c
1	d
2	a
2	b
2	c
2	e
3	b
3	c
3	e
3	f

```
SELECT  tb1.uid as word1, tb2.uid as word2,  
COUNT(*) as cnt  
FROM    my_table tb1 JOIN my_table tb2  
ON tb2.id = tb1.id AND tb2.uid > tb1.uid  
GROUP BY tb1.uid, tb2.uid
```



word1	word2	cnt
a	b	2
a	c	2
a	d	1
a	e	1
b	c	3





# Join more than two tables

create table purchase as (select  
customerNumber, count(\*) as Num\_of\_order  
from orders group by customerNumber)

🔑 orderNumber	orderDate	requiredDate	status	▲ customerNumber
10,298	2004-09-27	2004-10-05	Shipped	103
10,345	2004-11-25	2004-12-01	Shipped	103
10,123	2003-05-20	2003-05-29	Shipped	103
10,278	2004-08-06	2004-08-16	Shipped	112
10,346	2004-11-29	2004-12-05	Shipped	112
10,124	2003-05-21	2003-05-29	Shipped	112
10,342	2004-11-24	2004-12-01	Shipped	114
10,347	2004-11-29	2004-12-07	Shipped	114
10,120	2003-04-29	2003-05-08	Shipped	114

Table: orders

customerNumber	Num_of_order
103	3
112	3
114	5
119	4
121	4
124	17
128	4
129	3
131	4

Table: purchase

customerNumber	customerName	contactLastName	contactFirstName	phone
459	Warburg Exchange	Ottlieb	Sven	0241-039123
157	Diecast Classics Inc.	Leong	Kelvin	2155551555
303	Schuyler Imports	Schuyler	Bradley	+31 20 491 9555
496	Kelly's Gift Shop	Snowden	Tony	+64 9 5555500

Table: customers

customerNumber	checkNumber	paymentDate	amount
103	HQ336336	2004-10-19	6,066.78
103	JM555205	2003-06-05	14,571.44
103	OM314933	2004-12-18	1,676.14
112	BO864823	2004-12-17	14,191.12
112	HQ55022	2003-06-06	32,641.98

Table: payments


customerNumber	Num_or_order
103	3
112	3
114	5
119	4
121	4
124	17
128	4
129	3
131	4
141	26

Table: purchase

```

select a.customerNumber, a.customerName, a.creditLimit,
       b.amount, b.paymentDate,
       c.Num_of_order
from customers a
join payments b on a.customerNumber = b.customerNumber
join purchase c on a.customerNumber = c.customerNumber

```

 customerNumber	customerName	creditLimit	amount	paymentDate	Num_of_order
103	Atelier graphique	21,000	6,066.78	2004-10-19	3
103	Atelier graphique	21,000	14,571.44	2003-06-05	3
103	Atelier graphique	21,000	1,676.14	2004-12-18	3
112	Signal Gift Stores	71,800	14,191.12	2004-12-17	3
112	Signal Gift Stores	71,800	32,641.98	2003-06-06	3
112	Signal Gift Stores	71,800	33,347.88	2004-08-20	3
114	Australian Collectors, Co.	117,300	45,864.03	2003-05-20	5

# A select command as a new table

- **A select command can be used as a new but temporal table to e.g., join other tables.**

```
select a.customerNumber, a.customerName, a.creditLimit,  
       b.amount, b.paymentDate,  
       c.Num_of_order
```

```
from customers a
```

```
join payments b
```

```
on a.customerNumber = b.customerNumber
```

```
join (select customerNumber, count(*) as Num_or_order  
from orders group by customerNumber) c
```

```
on a.customerNumber = c.customerNumber
```



The alias "c" here  
cannot be  
ignored

ZIP_code	State	Submitted_via	Data_received	Data_sent_to_company	Company	Company_response
55,008	MN	Referral	2011-12-02	2011-12-02	GE Capital Retail	Closed without relief
10,065	NY	Referral	2011-12-02	2011-12-02	HSBC	Closed without relief
98,362	WA	Referral	2011-12-02	2011-12-02	HSBC	Closed without relief
80,537	CO	Referral	2011-12-02	2011-12-02	HSBC	Closed without relief
45,458	OH	Referral	2011-12-02	2011-12-02	JPMorgan Chase	Closed without relief
48,462	MI	Referral	2011-12-02	2011-12-02	JPMorgan Chase	Closed with relief
60,645	IL	Referral	2011-12-02	2011-12-02	JPMorgan Chase	Closed with relief
11,418	NY	Referral	2011-12-02	2011-12-02	JPMorgan Chase	Closed with relief

By comparing the Data\_received and Data\_sent\_to\_company,  
 we say that i) if the difference between two dates is more than 4 days, it is in “slow procedure”  
 ii) if the difference between two dates is between 3 and 4 days, it is in “normal procedure”  
 iii) if the difference between two dates is less than 3 days, it is in “fast procedure”

We termed this “slow procedure”, “normal procedure” and “fast procedure” as a new variable of “category”.

Please identify the **Proportion** of complaints that companies received in correspondence to “slow procedure”, “normal procedure” and “fast procedure”.

Please order the results by category and **Proportion** of complaints. Only company with more than **80** complaints in the data are considered.

# Examples

```
SELECT orderNumber, quantityOrdered,  
CASE  
WHEN quantityOrdered > 30 THEN "the quantity is over 30"  
WHEN quantityOrdered = 30 THEN "the quantity is 30"  
ELSE "the quantity is less than 30"  
END  
from orderdetails;
```

orderdetails (3×2,996)		
orderNumber	quantityOrdered	CASE
10,100	30	the quantity is 30
10,100	50	the quantity is over 30
10,100	22	the quantity is less than 30
10,100	49	the quantity is over 30
10,101	25	the quantity is less than 30
10,101	26	the quantity is less than 30

**select** t1.category, t1.company, t1.fre, t2.fre2, t1.fre/t2.fre2 **as** proportion **from**

**(Select**

**case**

**when** datediff(Data\_sent\_to\_company, Data\_received) > 5 **Then** 'Slow Procedure'

**when** datediff(Data\_sent\_to\_company, Data\_received) < 3 **Then** 'Fast Procedure'

**Else** 'Normal Procedure'

**End**

**as** category, company, count(\*) **as** fre

**from** cfpb\_complaints\_2500 **group by** category, company **order by** category, fre desc) **as** t1

**JOIN**

(select company, count(\*) **as** fre2 **from** cfpb\_complaints\_2500 **group by** company) **as** t2

**on** t1.company = t2.company

**having** fre2 > 80

**order by** t1.category, proportion



cfpb_complaints_2500 (5×18)				
category	company	fre	fre2	proportion
Fast Procedure	Ocwen	40	82	0.4878
Fast Procedure	Bank of America	335	581	0.5766
Fast Procedure	Wells Fargo	164	276	0.5942
Fast Procedure	JPMorgan Chase	203	331	0.6133
Fast Procedure	Citibank	174	267	0.6517
Fast Procedure	Capital One	147	211	0.6967
Normal Procedure	Capital One	40	211	0.1896
Normal Procedure	JPMorgan Chase	72	331	0.2175
Normal Procedure	Ocwen	18	82	0.2195
Normal Procedure	Citibank	60	267	0.2247
Normal Procedure	Bank of America	147	581	0.2530
Normal Procedure	Wells Fargo	71	276	0.2572
Slow Procedure	Capital One	24	211	0.1137
Slow Procedure	Citibank	33	267	0.1236
Slow Procedure	Wells Fargo	41	276	0.1486
Slow Procedure	JPMorgan Chase	56	331	0.1692
Slow Procedure	Bank of America	99	581	0.1704
Slow Procedure	Ocwen	24	82	0.2927


## Business insights:

1. Some companies apparently received complaints with much delay, such as Ocwen.
2. Companies were not treated equally by CFPU who sent complaints to some companies with more delays.

# Table VS. View

```
create table tb1 as  
(select * from A  
join B  
on A.A_ID = B.B_ID)
```

```
create view tb2 as  
(select * from A  
join B  
on A.A_ID = B.B_ID)
```

	tb1	16.0 KiB
	tb2	

# Feature of View (1)

- **A table contains data while a view is just a `SELECT` statement.**
- **A view is a kind of “virtual table”, which does not physically host any data by itself.**
- **Modification through a view (e.g. insert, update, delete) generally not permitted.**

## Feature of View (2)

- **The data in a view (result of the select query) will be changed automatically if the data in the source table is changed.**
- **A view always shows up-to-date data!**

# Why data warehouse is needed?

- **Scenario 1:** Your company just merge another similar company that have a very different database.
- **Scenario 2:** In the past years, your company established several new branches at different countries, most of which are using a different database structure.
- Etc.

# MySQL data warehousing



- **Data warehousing** is a technology that aggregates structured data from one or more sources so that it can be compared and analyzed for greater business intelligence.
- Or a **read-only** database for decision analysis.
- <https://www.codeproject.com/Articles/652108/Create-First-Data-WareHouse>



# Data Warehouse

- **Subject oriented.** Data are organized based on how the users refer to them.
- **Integrated.** All inconsistencies regarding naming convention and value representations are removed.
- **Nonvolatile.** Data are stored in read-only format and do not change over time.
- **Time variant.** Data are not current but normally time series.



# A Data Warehouse is Subject Oriented

