எளிய தமிழில் MySQL

• து.நித்யா

பாகம் 2

எளிய தமிழில் MySQL — பாகம் 2

து.நித்யா

மின்னூல் வெளியீடு : http://FreeTamilEbooks.com

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.

உள்ளடக்கம்

- எளிய தமிழில் MySQL — பாகம் 2
- உரிமை
- பதிப்பாசிரியர் உரை
- சமர்ப்பணம்
- ஆசிரியர் உரை
- பொருளடக்கம்
- Retrival of Data
- Functions & Operators
- Working with Dates - தேதிகளைக் கையாளுதல்
- Conditional Expressions
- Groups
- JOIN
- Subqueries
- Set Operators
- Ranks
- Stored Procedures
- Triggers
- முடிவுரை
- ஆசிரியர் பற்றி
- ஆசிரியரின் பிற மின்னூல்கள்

- கணியம் பற்றி
- நன்கொடை

சமர்ப்பணம்

தம் குடும்பத்தினருக்காகவும், சுற்றத்தினருக்காகவும்
உழைக்கும் அனைத்து சாதனைப்
பெண்மணிகளுக்கும் இந்தப் புத்தகம் சமர்ப்பணம்.

ஆசிரியர் உரை

"உன் கையை நம்பி உயர்ந்திட பாரு! உனக்கென எழுது ஒரு வரலாறு! உனக்குள்ள சக்தி இருக்கு! அதை உசுப்பிட வழிபாரு! " எனும் படையப்பா பாடல் வரிகளைக் கேட்கும் போதெல்லாம் நாம் ஏதாவது சாதிக்க வேண்டும்; எனக்கென ஒரு வரலாற்றை உருவாக்க வேண்டும் என்று உற்சாகமாகச் செயல்படுவேன்.

கணவனின் முன்னேற்றத்துக்கு துணை புரிவதுடன், தனக்கென்று ஒரு தனிப்பாதையை உருவாக்கி அதில் முன்னேறும் பெண்களையே என் *role model*-ஆகக் கொண்டேன். கணவனின் முன்னேற்றத்தில் பெருமை அடைவதுடன், தானும் ஏதேனும் சாதித்து முன்னேறும் பெண்களையே நான் பிரம்மிப்பாகப் பார்ப்பேன்.

ஆனால் இப்போது என்னுடைய பார்வை மாறியுள்ளது. ஆம்! நான் புத்தகம் எழுதிக்கொண்டிருக்கும்போது, என் குழந்தை

அழுதால், எதிர் வீட்டில் வசிக்கும் பானுமதி *aunty* என் குழந்தையை தூக்கிச் சென்று சமாதானப்படுத்தி பார்த்துக்கொள்வார். “உன் வேலையை முடிச்சிட்டு வந்து குழந்தையை வாங்கிக்கோ” என்று என்னிடம் சொல்வர்.

அவ்வாறே தற்போது நான் தங்கியிருக்கும் வீட்டில் நானும், ஹான்னா அக்காவும் சேர்ந்தே உணவு சமைப்பது வழக்கம். ஆனால் நான் புத்தகம் எழுதத் தொடங்கி விட்டால், உணவு சமைக்கும் வேலையே எனக்கு மறந்துவிடும். எப்போது எனக்கு பசி வருகிறதோ, அப்போதுதான் எழுந்து உணவு சமைக்க ஒடுவேன். அங்கு சென்று பார்த்தால், அந்த அக்கா ஒருவராகவே சமைத்து முடித்துவிட்டிருப்பார்கள். “நீ ஏதோ வேலையா இருந்த; அதான் உன்னை கூப்பிடலை. நீ சாப்டிட்டு போய் உன் வேலையை பாரு” என்று சொல்வார்கள். இதுபோன்ற பெண்களையெல்லாம் பார்க்கும்போது, எனக்கு மிகவும் பிரம்மிப்பாக இருக்கிறது. இந்தப் புத்தகத்தின் ஆசிரியர் என்று என் பெயர் இருக்கலாம். ஆனால், இதுபோன்ற பெண்களெல்லாம் இல்லை என்றால் எனது எந்தப்

புத்தகமும் புத்தகம் இல்லை.

எனவே இப்போதெல்லாம் தன்னுடைய பெயர் வெளியே தெரியாவிட்டாலும், மற்றவர்களுக்காகவும், குடும்பத்துக்காகவும் உழைத்துக் கொண்டிருக்கும் பெண்களே என் கண்களுக்கு சாதனைப் பெண்களாகத் தெரிகிறார்கள். ஒவ்வொரு வீட்டிலும் உள்ள சாதனைப் பெண்களுக்கும் என் மனமார்ந்த நன்றிகள்.

தமிழில் கணிணி நுட்பங்களைப் பகிர, ஒரு களமாக உள்ள ‘கணியம்’ தளத்தில், இதுவரை வெளியான எனது மின்னூல்களுக்கு வாசகர்கள் தரும் ஆதரவு பெருமகிழ்ச்சி அளிக்கிறது.

இலக்கியம், சினிமா, அரசியல் பற்றி மட்டும்தான் தமிழில் இருக்கும் என்ற நிலை மாறி, பல்வேறு துறைசார்ந்த நுட்ப விஷயங்களும் தமிழில் இருக்கும் என்ற நிலை உருவாக உழைத்து வரும் கணியம் குழுவினருக்கு நன்றி.

தொடர்ந்து ஊக்கம் அளிக்கும் என்

குடும்பத்தினருக்கும், கணியம் குழுவினருக்கும், *FreeTamilEbooks.com* குழுவினருக்கும், வாசகர்களுக்கும் நன்றிகள்.

து. நித்யா

நியூ காசில்,
இங்கிலாந்து.,

4 மே 2015

மின்னஞ்சல்: *nithyadurai87@gmail.com*

வலை பதிவு:
http://nithyashrinivasan.wordpress.com

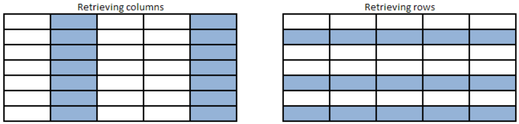
1

Retrival of Data

MySQL-ன் முதலாம் பாகத்தில் *database* மற்றும் *tables*-ஐ எவ்வாறு உருவாக்குவது, அதனை எவ்வாறு பயன்படுத்துவது என்பது போன்ற அடிப்படையான விஷயங்களைப் பற்றிப் பார்த்தோம். இந்தப் புத்தகத்தில் பல்வேறு வகையான *queries*-ஐப் பயன்படுத்தி வெவ்வேறு விதங்களில் தகவல்களை எவ்வாறு வெளிக் கொண்டு வருவது என்பது பற்றிப் பார்க்கப் போகிறோம்.

1 *Retrival of Data*

ஒரு *database*-ல் *columns*-ஐத் தேர்ந்தெடுக்கும் போது என்ன நிகழ்கிறது, *rows*-ஐத் தேர்ந்தெடுக்கும்போது என்ன நிகழ்கிறது என்பதைப் பின்வரும் படத்தின் மூலம் தெளிவாகப் புரிந்து கொள்ளலாம



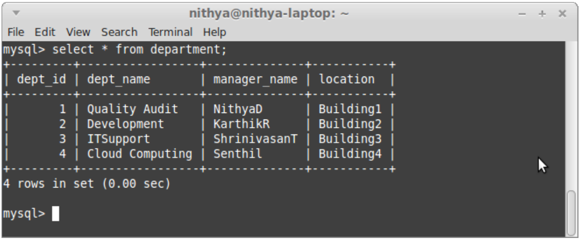
1.1 Basic select statement

Query 1

ஒரு *table*-ல் உள்ள அனைத்து *columns*-ன் தகவல்களும் வெளிப்பட வேண்டுமெனில் பின்வரும்

query-ஐப் பயன்படுத்தலாம்.

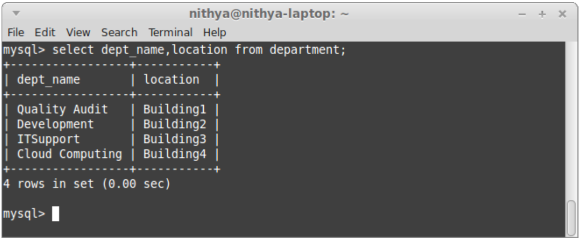
*select * from department;*



Query 2

ஒருசில குறிப்பிட்ட *columns*-ன் தகவல்கள் வெளிப்பட வேண்டுமெனில், *select*-ஐத் தொடர்ந்து அத்தகைய *columns*-ன் பெயர்களைக் குறிப்பிட வேண்டும்.

select dept_name,location from department;

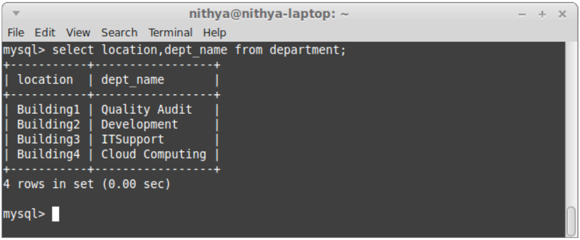


```
nithya@nithya-laptop: ~
File Edit View Search Terminal Help
mysql> select dept_name,location from department;
+-----------------+-----------+
| dept_name       | location  |
+-----------------+-----------+
| Quality Audit   | Building1 |
| Development     | Building2 |
| ITSupport       | Building3 |
| Cloud Computing | Building4 |
+-----------------+-----------+
4 rows in set (0.00 sec)

mysql>
```

Query 3

select-ஐத் தொடர்ந்து கொடுக்கப்படும் *columns*, முன்னும் பின்னுமாகக் கூட இருக்கலாம்.

select location,dept_name from department;



```
nithya@nithya-laptop: ~
File Edit View Search Terminal Help
mysql> select location,dept_name from department;
+-----------+-----------------+
| location  | dept_name       |
+-----------+-----------------+
| Building1 | Quality Audit   |
| Building2 | Development     |
| Building3 | ITSupport       |
| Building4 | Cloud Computing |
+-----------+-----------------+
4 rows in set (0.00 sec)

mysql> 
```

1.2 Arithmetic Expressions

கூட்டல், கழித்தல், பெருக்கல், வகுத்தல் ஆகிய நான்குமே *arithmetic operators* எனப்படுகின்றன. இவற்றைப் பயன்படுத்தி நாம் தகவல்களை நமக்கு விரும்பிய வடிவில் பெறலாம். இதனைப் பின்வரும் உதாரணத்தில் பார்க்கலாம்.

Query 4

ஒரு நிறுவனத்தில் உள்ள அனைத்து ஊழியர்களுக்கும், அவர்களின் தற்போதைய

சம்பளத்துடன் 2475 ரூபாய் உயர்த்திக் கொடுக்க உத்தரவு வருகிறதெனில், ஒவ்வொருவருடைய புதிய சம்பளத்தையும் கணிக்க கூட்டல் எனும் *arithmetic operator*-ஐப் பயன்படுத்தலாம்.

select name,salary,salary+2475 from employees;

```
nithya@nithya-laptop: ~
File  Edit  View  Search  Terminal  Help
mysql> select name,salary,salary+2475 from employees;
+---------+--------+-------------+
| name    | salary | salary+2475 |
+---------+--------+-------------+
| Sudha   |  12000 |       14475 |
| Revathi |  19500 |       21975 |
| Sherlin |   4500 |        6975 |
| Malathi |   7500 |        9975 |
| Jeeva   |  21500 |       23975 |
+---------+--------+-------------+
5 rows in set (0.00 sec)

mysql> 
```

இதில் *name* மற்றும் *salary* என்பது ஏற்கனவே அந்த *tables*-ல் உள்ள *columns* ஆகும். ஆனால் *salary*+2475 என்பது அந்த *table*-ல் இல்லாத, நமது தேவைக்காக நாம் உருவாக்கிய ஒரு புதிய *column*.

Query 5

Arithmetic operations-ஐ ஒரு *column*-ல் உள்ள மதிப்புகளின் மீது செலுத்தி, ஒரு புதிய *column*-ஐ நாம் உருவாக்கும்போது, '*Operator Precedence*' எனும் ஒரு முக்கியமான விஷயத்தை நினைவில் கொள்ள வேண்டும். உதாரணத்துக்கு சென்ற *Query*-ல், நாம் கண்டுபிடித்த ஒவ்வொருவரின் புதிய மாத சம்பளத்தையும் வருடத்திற்குக் கணக்கிட அதனை 12-ஆல் பெருக்கினால் போதுமானது. எனவே *query*-ஐப் பின்வருமாறு அமைப்போம்.

*select name,salary,12*salary+2475 from employees;*

```
nithya@nithya-laptop: ~
File  Edit  View  Search  Terminal  Help
mysql> select name,salary,12*salary+2475 from employees;
+---------+--------+----------------+
| name    | salary | 12*salary+2475 |
+---------+--------+----------------+
| Sudha   |  12000 |         146475 |
| Revathi |  19500 |         236475 |
| Sherlin |   4500 |          56475 |
| Malathi |   7500 |          92475 |
| Jeeva   |  21500 |         260475 |
+---------+--------+----------------+
5 rows in set (0.01 sec)

mysql>
```

இதில் ஒவ்வொருவருடைய தற்போதைய சம்பளமும் 12-ஆல் பெருக்கப்பட்டு, பின்னர் அதனுடன் 2475 ரூபாய் கூட்டப்படுகிறது. அதாவது கூட்டல் கணக்கு நடைபெற்று, புதிய சம்பளம் கண்டுபிடிப்பதற்கு முன்னரே, பெருக்கல் கணக்கு நடைபெற்றுவிடுகிறது. இதுவே '*Operator Precedence*' ஆகும். இதனைத் தவிர்ப்பதற்கு நாம் அடைப்புக்குறியைப் பயன்படுத்தலாம். அதாவது எந்தக் கணக்கு முதலில் நடைபெற வேண்டுமோ அதனை அடைப்புக்குறிக்குள் கொடுக்க வேண்டும். இது பின்வருமாறு.

```
select name,salary,12*(salary+2475) from employees;
```

```
nithya@nithya-laptop: ~
File  Edit  View  Search  Terminal  Help
mysql> select name,salary,12*(salary+2475) from employees;
+---------+--------+------------------+
| name    | salary | 12*(salary+2475) |
+---------+--------+------------------+
| Sudha   |  12000 |           173700 |
| Revathi |  19500 |           263700 |
| Sherlin |   4500 |            83700 |
| Malathi |   7500 |           119700 |
| Jeeva   |  21500 |           287700 |
+---------+--------+------------------+
5 rows in set (0.00 sec)

mysql> 
```

1.3 Null Values

Query 6

பின்வரும் query-ல் commission_pct எனும் column ஒவ்வொருவருக்கும் எவ்வளவு சதவீதம் commission கிடைக்கிறது எனும் மதிப்பினைப் பெற்றுள்ளது. இதில் ஒருசில நபர்களுக்கு Null எனும் மதிப்பு உள்ளது. இத்தகைய Null எனும் மதிப்பினைப் பெற்றவர்களுக்கு commission பூஜ்ஜியம் என்றோ அல்லது எதுவும் கிடையாது என்றோ அர்த்தம் இல்லை. அவர்களுக்கு இன்னும் commission வரையறுக்கப்படவில்லை என்றே அர்த்தம்.

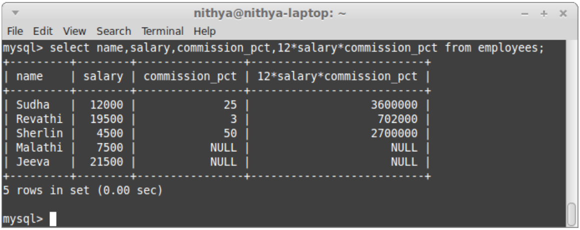
select name,role,salary,commission_pct from employees;

```
nithya@nithya-laptop: ~
File Edit View Search Terminal Help
mysql> select name,role,salary,commission_pct from employees;
+---------+--------------------+--------+----------------+
| name    | role               | salary | commission_pct |
+---------+--------------------+--------+----------------+
| Sudha   | Testing Engineer   |  12000 |             25 |
| Revathi | ITSupport Engineer |  19500 |              3 |
| Sherlin | Asst. Engineer     |   4500 |             50 |
| Malathi | Java Developer     |   7500 |           NULL |
| Jeeva   | ITSuppot Engineer  |  21500 |           NULL |
+---------+--------------------+--------+----------------+
5 rows in set (0.00 sec)

mysql> 
```

Query **7**

Null மதிப்பினைப் பெற்றுள்ள *column*-ன் மீது *arithmetic operations*-ஐ செலுத்தி உருவாக்கப்படும் புதிய *column*-ம் *Null* மதிப்பினையே பெற்றிருக்கும்.

*select name,salary,commission_pct,*12**salary*commission_p from employees;*



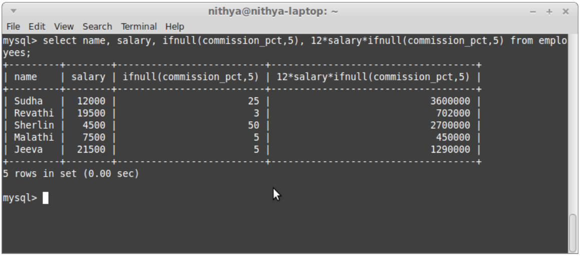
```
nithya@nithya-laptop: ~
File Edit View Search Terminal Help
mysql> select name,salary,commission_pct,12*salary*commission_pct from employees;
+---------+--------+----------------+--------------------------+
| name    | salary | commission_pct | 12*salary*commission_pct |
+---------+--------+----------------+--------------------------+
| Sudha   |  12000 |             25 |                  3600000 |
| Revathi |  19500 |              3 |                   702000 |
| Sherlin |   4500 |             50 |                  2700000 |
| Malathi |   7500 |           NULL |                     NULL |
| Jeeva   |  21500 |           NULL |                     NULL |
+---------+--------+----------------+--------------------------+
5 rows in set (0.00 sec)

mysql>
```

1.4 Null functions

Query-8

ஒரு *column*, *'NULL'* மதிப்பினை வெளிப்படுத்துவதற்கு பதிலாக வேறு ஏதேனும் மதிப்பினை வெளிப்படுத்துமாறு செய்ய *IFNULL function*-ஐப் பயன்படுத்தலாம்.

*select name, salary, ifnull(commission_pct,5), 12*salary*ifnull(commission_pct,5) from employees;*



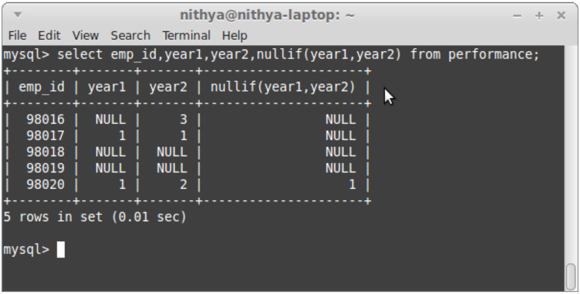
```
nithya@nithya-laptop: ~
File Edit View Search Terminal Help
mysql> select name, salary, ifnull(commission_pct,5), 12*salary*ifnull(commission_pct,5) from emplo
yees;
+---------+--------+--------------------------+------------------------------------+
| name    | salary | ifnull(commission_pct,5) | 12*salary*ifnull(commission_pct,5) |
+---------+--------+--------------------------+------------------------------------+
| Sudha   |  12000 |                       25 |                            3600000 |
| Revathi |  19500 |                        3 |                             702000 |
| Sherlin |   4500 |                       50 |                            2700000 |
| Malathi |   7500 |                        5 |                             450000 |
| Jeeva   |  21500 |                        5 |                            1290000 |
+---------+--------+--------------------------+------------------------------------+
5 rows in set (0.00 sec)

mysql>
```

இதில் *commission_pct* என்பது *null* மதிப்பினைப் பெற்றிருப்பின், அதனை 5 எனக் கொள்ளுமாறு அமைத்துள்ளது.

Query-9

NULLIF என்பது இரண்டு மதிப்புகளை ஒப்பிட்டு அவை சமமாக இருந்தாலோ அல்லது ஏதேனும் ஒரு மதிப்பு *null*-ஆக இருந்தாலோ *NULL* மதிப்பினை வெளிப்படுத்துகிறது.

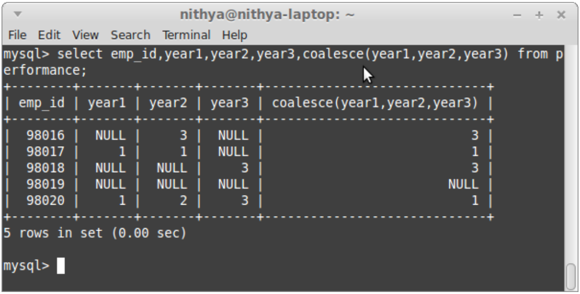
select emp_id,year1,year2,nullif(year1,year2) from performance;



Query-10

COALESCE மூலம் ஒன்றன்பின் ஒன்றாக நிறைய மதிப்புகளைக் குறிப்பிட்டு, முதல் மதிப்பு null-ஆக இருந்தால் இரண்டாவது மதிப்பினையும், இரண்டாவதும் null-ஆக இருந்தால் மூன்றாவது மதிப்பினையும், அதுவும் null-ஆக இருப்பின் அதற்கு அடுத்தடுத்த மதிப்புகளையும் வெளிப்படுத்துமாறு செய்யலாம்.

select
emp_id,year1,year2,year3,coalesce(year1,year2,year3)
from performance;



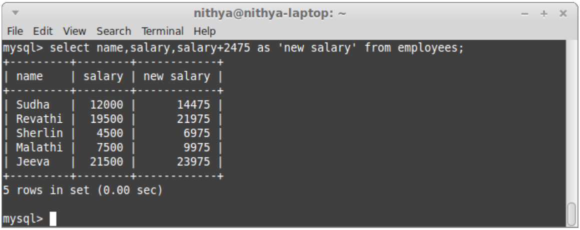
```
nithya@nithya-laptop: ~
File Edit View Search Terminal Help
mysql> select emp_id,year1,year2,year3,coalesce(year1,year2,year3) from p
erformance;
+--------+-------+-------+-------+-----------------------------+
| emp_id | year1 | year2 | year3 | coalesce(year1,year2,year3) |
+--------+-------+-------+-------+-----------------------------+
|  98016 |  NULL |     3 |  NULL |                           3 |
|  98017 |     1 |     1 |  NULL |                           1 |
|  98018 |  NULL |  NULL |     3 |                           3 |
|  98019 |  NULL |  NULL |  NULL |                        NULL |
|  98020 |     1 |     2 |     3 |                           1 |
+--------+-------+-------+-------+-----------------------------+
5 rows in set (0.00 sec)

mysql>
```

1.5 Column aliases

Query-11

நாம் உருவாக்கும் புதிய *column*-க்கு ஒரு பெயர் வைக்க விரும்பினால் *column alias*-ஐப் பயன்படுத்தலாம். உதாரணத்துக்கு *salary+2475* என்று நாம் உருவாக்கிய புதிய *column*-க்கு '*New Salary*' என்று பெயர் வைக்க விரும்பினால், அது பின்வருமாறு.

select name,salary,salary+2475 as 'new salary' from employees;

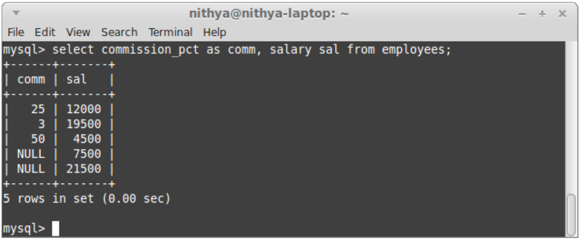


```
nithya@nithya-laptop: ~
File Edit View Search Terminal Help
mysql> select name,salary,salary+2475 as 'new salary' from employees;
+---------+--------+------------+
| name    | salary | new salary |
+---------+--------+------------+
| Sudha   |  12000 |      14475 |
| Revathi |  19500 |      21975 |
| Sherlin |   4500 |       6975 |
| Malathi |   7500 |       9975 |
| Jeeva   |  21500 |      23975 |
+---------+--------+------------+
5 rows in set (0.00 sec)

mysql>
```

Query-12

சென்ற *quey*-ல் *new salary* எனும் *alias name* இரண்டு பெயராக உள்ளதால், அது *double quotes*-க்குள் கொடுக்கப்பட்டுள்ளது. இங்கு ஒரே ஒரு பெயர் *alias* பெயராக உள்ளத்தால், அது *double quotes* இல்லாமல் கொடுக்கப்படுவதை கவனிக்கவும். அவ்வாறே *as* எனும் *keyword, optional* ஆகும். எனவே அது இல்லாமலேயே ஒரு *column*-க்கு பெயர் வைக்கப்பட்டிருப்பதையும் கவனிக்கவும்.

select commission_pct as comm, salary sal from employees;



```
nithya@nithya-laptop: ~
File Edit View Search Terminal Help
mysql> select commission_pct as comm, salary sal from employees;
+------+-------+
| comm | sal   |
+------+-------+
|   25 | 12000 |
|    3 | 19500 |
|   50 |  4500 |
| NULL |  7500 |
| NULL | 21500 |
+------+-------+
5 rows in set (0.00 sec)

mysql> 
```

2

Functions & Operators

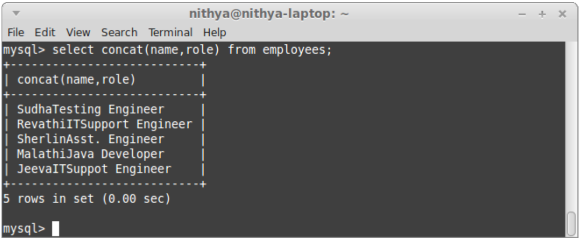
Mysql-ல் பல்வேறு வகையான *functions* மற்றும் *operators* இருந்தாலும் ஒருசில முக்கியமானவைகளைப் பற்றி இங்கு பார்ப்போம்.

2.1 Concat function

Query-13

இரண்டு தனித்தனி *columns*-ல் உள்ள மதிப்புகளை இணைத்து ஒரே மதிப்பாக வெளியிடும் வேலையை *concat function* செய்கிறது. இது பின்வருமாறு.

select concat(name,role) from employees;



இதில் *name* மற்றும் *role* என்பது இரண்டு தனித்தனி *columns* ஆகும். அவற்றின் மதிப்பு *concat* மூலம் ஒன்றாக இணைத்து வெளியிடப்பட்டுள்ளது

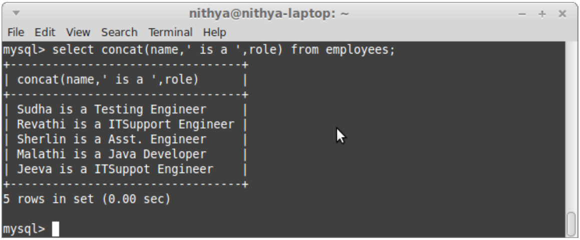
2.2 Literals

Query-14

இரண்டு *columns*-ன் மதிப்புகளை இணைத்து

வெளிப்படுத்துவதோடு அல்லாமல் நாம் விரும்பும் ஒருசில வார்த்தைகளையும் சேர்த்து வெளிப்படுத்த முடியும். இவ்வாறு இணைக்கப்படும் வார்த்தைகள் '*literals*' எனப்படும்.

select concat(name,' is a ',role) from employees;



இதில் '*is a*' என்பது *literals* ஆகும்.

2.3 Escape sequence

Query-15

பொதுவாக '*literals*' என்பவை எப்போதும் *single quotes*-க்குள் காணப்படும். ஆனால் *single quote*-ஐ உள்ளடக்கிய ஒருசில வார்த்தைகளை நாம் *literal*-ஆக கொடுக்க விரும்பினால் அது *back slash*-ஐப் பயன்படுத்தி பின்வருமாறு அமையும். இதனை *escape sequence* எனலாம்.

```
nithya@nithya-laptop: ~
File  Edit  View  Search  Terminal  Help
mysql> select concat(name,'\'s role is ',role) as Informations from employees;
+--------------------------------------+
| Informations                         |
+--------------------------------------+
| Sudha's role is Testing Engineer     |
| Revathi's role is ITSupport Engineer |
| Sherlin's role is Asst. Engineer     |
| Malathi's role is Java Developer     |
| Jeeva's role is ITSuppot Engineer    |
+--------------------------------------+
5 rows in set (0.00 sec)

mysql> 
```

2.4 Distinct

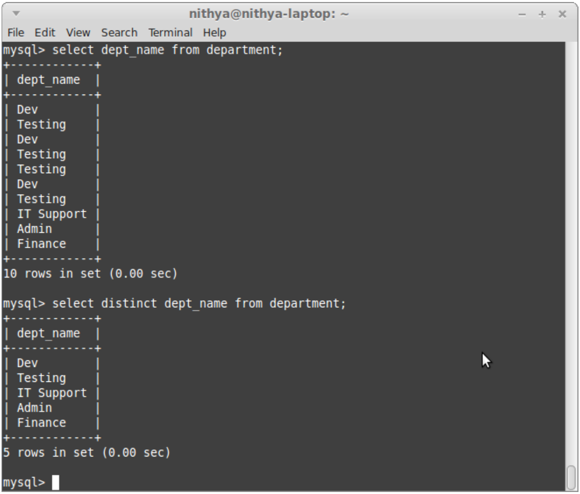
Query-16

Distinct-ஆனது ஒரு *column*-ல் ஒன்றுக்கும்

மேற்பட்ட ஒரே மாதிரியான மதிப்புகள் காணப்பட்டால் அதனை ஒரே ஒரு முறை மட்டும் வெளிப்படுத்தும். உதாரணத்துக்கு பின்வரும் *query*, '*dept_name*' *column*-ல் உள்ள அனைத்து மதிப்புகளையும் வெளிப்படுத்துகிறது. பின்னர் *distinct dept_name* எனக் கொடுக்கும் போது ஒரு் மதிப்பினை ஒருமுறை மட்டுமே வெளிப்படுத்துகிறது.

select dept_name from department;

select distinct dept_name from department;



nithya@nithya-laptop: ~

File Edit View Search Terminal Help

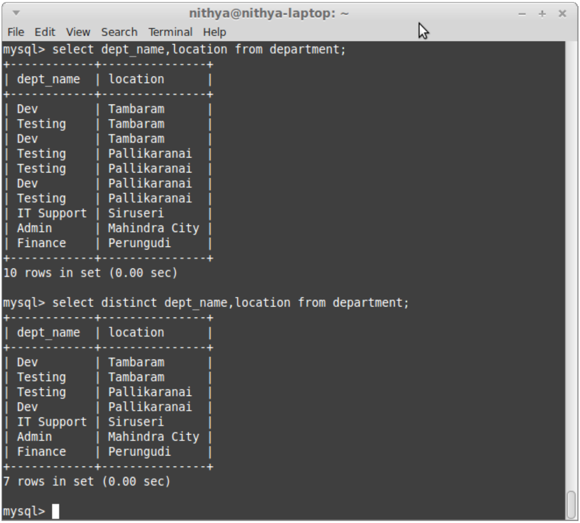
```
mysql> select dept_name from department;
+------------+
| dept_name  |
+------------+
| Dev        |
| Testing    |
| Dev        |
| Testing    |
| Testing    |
| Dev        |
| Testing    |
| IT Support |
| Admin      |
| Finance    |
+------------+
10 rows in set (0.00 sec)

mysql> select distinct dept_name from department;
+------------+
| dept_name  |
+------------+
| Dev        |
| Testing    |
| IT Support |
| Admin      |
| Finance    |
+------------+
5 rows in set (0.00 sec)

mysql>
```

Query-17

distinct dept_name,location எனக் கொடுக்கும் போது *dept_name* மதிப்பினை ஒரு *location*-க்கு ஒருமுறை வெளிப்படுத்துகிறது. இது பின்வருமாறு.

select dept_name,location from department;

select distinct dept_name,location from department;



இங்கு *Testing* என்பது *Distinct* மதிப்பாக

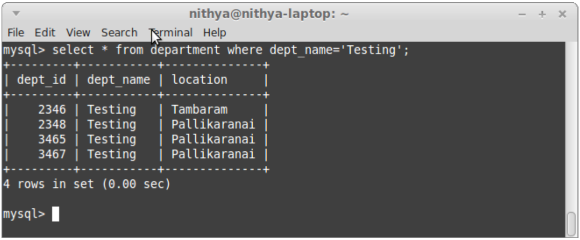
இருந்தாலும், *location* வேறுபடுவதால், இருமுறை வருகிறது.

2.5 Simple Conditions

Query-18

ஏதேனும் ஒரு கட்டளையின் அடிப்படையில் தகவல்களை வெளியிட *where* பயன்படுகிறது. உதாரணத்துக்கு *'Testing' department*-ல் உள்ள விவரங்களை மட்டும் பட்டியலிட*where dept_name ='Testing'* எனக் கொடுக்க வேண்டும். இது பின்வருமாறு.

*select * from department where dept_name='Testing';*

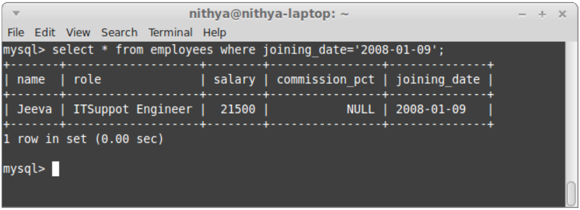


```
nithya@nithya-laptop: ~
File Edit View Search Terminal Help
mysql> select * from department where dept_name='Testing';
+---------+-----------+--------------+
| dept_id | dept_name | location     |
+---------+-----------+--------------+
|    2346 | Testing   | Tambaram     |
|    2348 | Testing   | Pallikaranai |
|    3465 | Testing   | Pallikaranai |
|    3467 | Testing   | Pallikaranai |
+---------+-----------+--------------+
4 rows in set (0.00 sec)

mysql>
```

Query-19

தேதியை அடிப்படையாகக் கொண்டும் கட்டளைகளை அமைக்கலாம். இது பின்வருமாறு.

*select * from employees where joining_date='2008-01-09';*

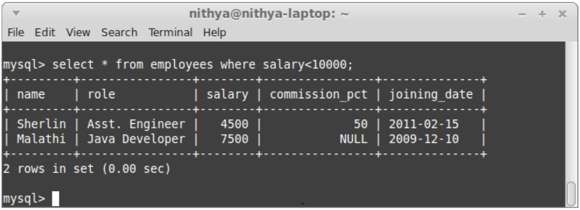


2.6 Conditions with comparison operators

Query-20

ஒரு *column*-ஐ ஒரு குறிப்பிட்ட மதிப்புடன் ஒப்பிட்டு அதனடிப்படையில் கட்டளைகளை அமைப்பதே *conditions with comparison operators* எனப்படும். உதாரணத்துக்கு 10,000 ரூபாய்க்கு கீழ் சம்பளம் வாங்கும் நபர்களைப் பட்டியலிட *comparison operator*-ஐப் பின்வருமாறு அமைக்கலாம்.

*select * from employees where salary<10000;*

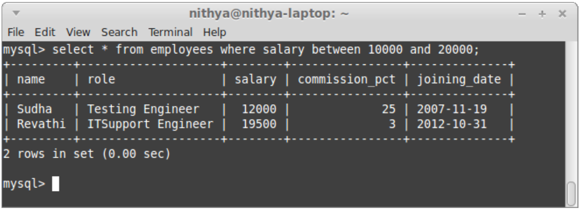


```
nithya@nithya-laptop: ~
File Edit View Search Terminal Help

mysql> select * from employees where salary<10000;
+---------+----------------+--------+----------------+--------------+
| name    | role           | salary | commission_pct | joining_date |
+---------+----------------+--------+----------------+--------------+
| Sherlin | Asst. Engineer |   4500 |             50 | 2011-02-15   |
| Malathi | Java Developer |   7500 |           NULL | 2009-12-10   |
+---------+----------------+--------+----------------+--------------+
2 rows in set (0.00 sec)

mysql>
```

Query-21

between operator-ஐப் பயன்படுத்தி இரண்டு மதிப்புகளைக் கொடுத்து அதனிடையில் அமையும் தகவல்களை எல்லாம் பெற முடியும். அந்த இரண்டு மதிப்புகளில் ஒன்று *lower limit*-ஆகவும், மற்றொன்று *upper limit*-ஆகவும் அமையும். உதாரணத்துக்கு ரூபாய் 10,000-லிருந்து 20,000-வரை சம்பளம் வாங்கும் நபர்களைப் பட்டியலிட *between operator*-ஐப் பின்வருமாறு அமைக்கலாம்.

*select * from employees where salary between 10000 and 20000;*

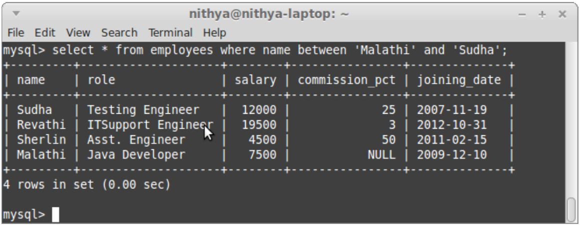


```
nithya@nithya-laptop: ~
File  Edit  View  Search  Terminal  Help
mysql> select * from employees where salary between 10000 and 20000;
+---------+--------------------+--------+----------------+--------------+
| name    | role               | salary | commission_pct | joining_date |
+---------+--------------------+--------+----------------+--------------+
| Sudha   | Testing Engineer   |  12000 |             25 | 2007-11-19   |
| Revathi | ITSupport Engineer |  19500 |              3 | 2012-10-31   |
+---------+--------------------+--------+----------------+--------------+
2 rows in set (0.00 sec)

mysql>
```

Query-22

between operators-ன் *lower* மற்றும் *upper limits*-ஆக பெயர்களையும் கொடுக்க முடியும். பின்வரும் உதாரணத்தில் ‘*Malathi*’ மற்றும் ‘*Sudha*’ எனும் இரண்டு பெயர்களுக்கிடையில் அமையும் அனைத்துப் பெயர்களும் பட்டியலிடப்படும்.

*select * from employees where name between ‘Malathi’ and ‘Sudha*';

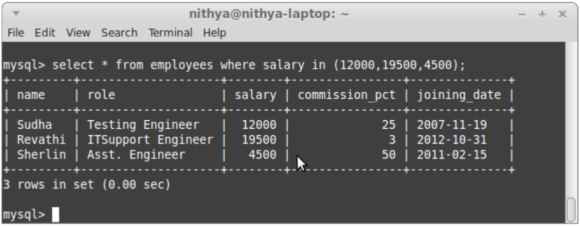


```
nithya@nithya-laptop: ~
File Edit View Search Terminal Help
mysql> select * from employees where name between 'Malathi' and 'Sudha';
+---------+-------------------+--------+----------------+--------------+
| name    | role              | salary | commission_pct | joining_date |
+---------+-------------------+--------+----------------+--------------+
| Sudha   | Testing Engineer  |  12000 |             25 | 2007-11-19   |
| Revathi | ITSupport Engineer|  19500 |              3 | 2012-10-31   |
| Sherlin | Asst. Engineer    |   4500 |             50 | 2011-02-15   |
| Malathi | Java Developer    |   7500 |           NULL | 2009-12-10   |
+---------+-------------------+--------+----------------+--------------+
4 rows in set (0.00 sec)

mysql>
```

Query-23

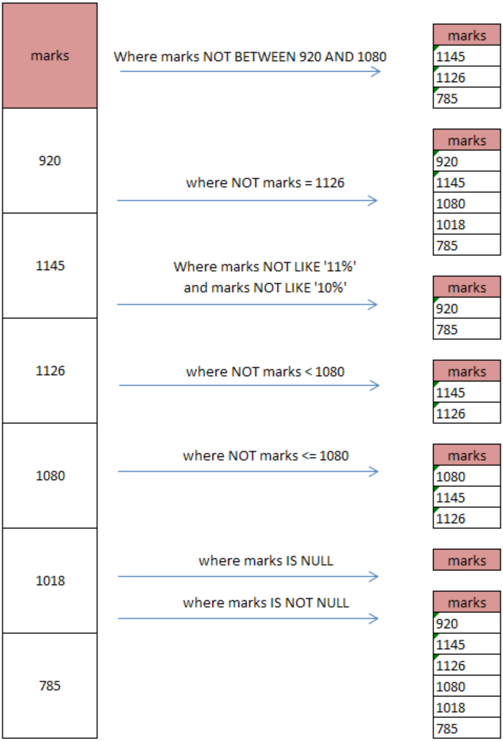
தொடர்ச்சியாக ஒருசில குறிப்பிட்ட மதிப்புகளைக் கொடுத்து அதனைப் பெற்று விளங்கும் தகவல்களை மட்டும் பட்டியலிட *in operator*-ஐப் பயன்படுத்தலாம். இது பின்வருமாறு.

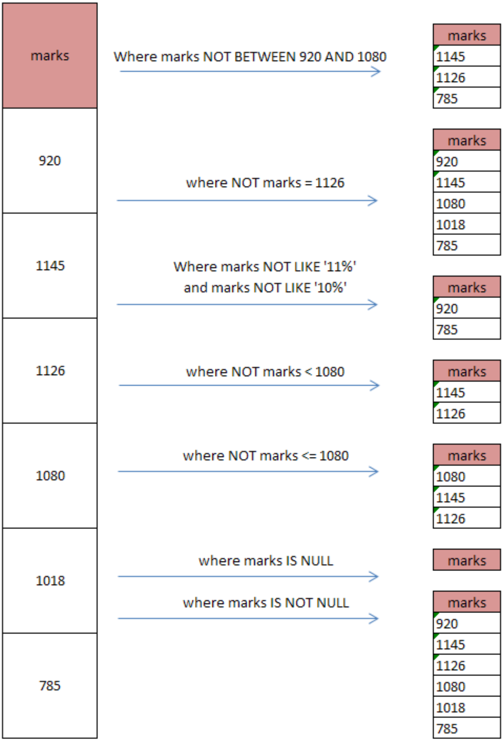
*select * from employees where salary in (12000,19500,4500);*



```
nithya@nithya-laptop: ~
File Edit View Search Terminal Help

mysql> select * from employees where salary in (12000,19500,4500);
+---------+--------------------+--------+----------------+--------------+
| name    | role               | salary | commission_pct | joining_date |
+---------+--------------------+--------+----------------+--------------+
| Sudha   | Testing Engineer   |  12000 |             25 | 2007-11-19   |
| Revathi | ITSupport Engineer |  19500 |              3 | 2012-10-31   |
| Sherlin | Asst. Engineer     |   4500 |             50 | 2011-02-15   |
+---------+--------------------+--------+----------------+--------------+
3 rows in set (0.00 sec)

mysql> 
```

இன்னும் சில *comparison operators* என்ன செய்கிறது என்பதைப் பின்வரும் படத்தில் காணலாம்.


marks
920
1145
1126
1080
1018
785
Where marks NOT BETWEEN 920 AND 1080
marks
1145
1126
785
where NOT marks = 1126
marks
920
1145
1080
1018
785
Where marks NOT LIKE '11%'
and marks NOT LIKE '10%'
marks
920
785
where NOT marks < 1080
marks
1145
1126
where NOT marks <= 1080
marks
1080
1145
1126
where marks IS NULL
marks
where marks IS NOT NULL
marks
920
1145
1126
1080
1018
785



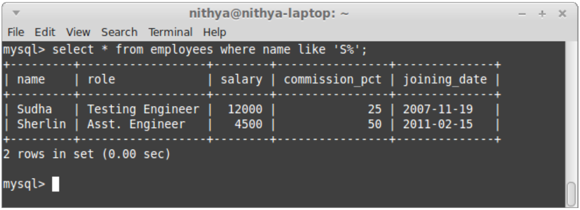
marks
920
1145
1126
1080
1018
785
Where marks NOT BETWEEN 920 AND 1080
marks
1145
1126
785
where NOT marks = 1126
marks
920
1145
1080
1018
785
Where marks NOT LIKE '11%'
and marks NOT LIKE '10%'
marks
920
785
where NOT marks < 1080
marks
1145
1126
where NOT marks <= 1080
marks
1080
1145
1126
where marks IS NULL
marks
where marks IS NOT NULL
marks
920
1145
1126
1080
1018
785

2.7 Pattern Matching

Query-24

Like operator-ஆனது ஒரே மாதிரியான *pattern*-ல் அமையும் தகவல்களைப் பட்டியலிடும். உதாரணத்துக்கு *S* எனும் எழுத்தில் தொடங்கும் நபர்களின் பெயர்களைப் பட்டியலிட % எனும் *wildcard character* பின்வருமாறு பயன்படுகிறது.

*select * from employees where name like 'S%';*

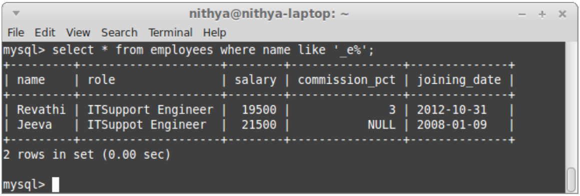


```
nithya@nithya-laptop: ~
File Edit View Search Terminal Help
mysql> select * from employees where name like 'S%';
+---------+------------------+--------+----------------+--------------+
| name    | role             | salary | commission_pct | joining_date |
+---------+------------------+--------+----------------+--------------+
| Sudha   | Testing Engineer |  12000 |             25 | 2007-11-19   |
| Sherlin | Asst. Engineer   |   4500 |             50 | 2011-02-15   |
+---------+------------------+--------+----------------+--------------+
2 rows in set (0.00 sec)

mysql>
```

Query-25

அவ்வாறே முதல் எழுத்து எதுவாக இருந்தாலும் இரண்டாம் எழுத்து *e*-வாக இருக்கும் பெயர்களைப் பட்டியலிட *underscore* எனும் *wildcard character* பின்வருமாறு பயன்படுகிறது.

*select * from employees where name like '_e%';*



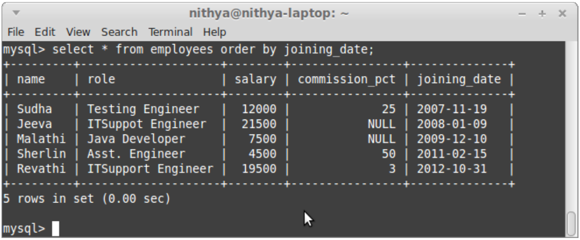
```
nithya@nithya-laptop: ~
File Edit View Search Terminal Help
mysql> select * from employees where name like '_e%';
+---------+--------------------+--------+----------------+--------------+
| name    | role               | salary | commission_pct | joining_date |
+---------+--------------------+--------+----------------+--------------+
| Revathi | ITSupport Engineer |  19500 |              3 | 2012-10-31   |
| Jeeva   | ITSuppot Engineer  |  21500 |           NULL | 2008-01-09   |
+---------+--------------------+--------+----------------+--------------+
2 rows in set (0.00 sec)

mysql>
```

2.8 Order by

Query-26

Order by என்பது இயல்பாகத் தகவல்களை ஏறுவரிசையில் முறைப்படுத்திக் காட்ட உதவுகிறது. உதாரணத்துக்கு ஒருவர் தேர்வு செய்யப்பட்ட ஆண்டின் அடிப்படையில் தகவல்களை முறைப்படுத்த *query*-யைப் பின்வருமாறு அமைக்கலாம்.

*select * from employees order by joining_date;*



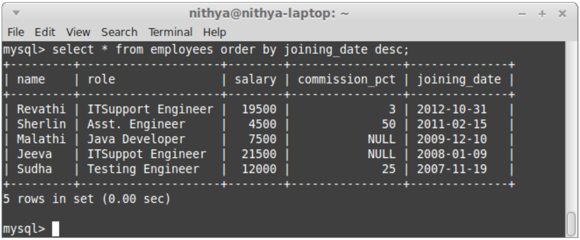
```
nithya@nithya-laptop: ~
File Edit View Search Terminal Help
mysql> select * from employees order by joining_date;
+---------+--------------------+--------+----------------+--------------+
| name    | role               | salary | commission_pct | joining_date |
+---------+--------------------+--------+----------------+--------------+
| Sudha   | Testing Engineer   |  12000 |             25 | 2007-11-19   |
| Jeeva   | ITSuppot Engineer  |  21500 |           NULL | 2008-01-09   |
| Malathi | Java Developer     |   7500 |           NULL | 2009-12-10   |
| Sherlin | Asst. Engineer     |   4500 |             50 | 2011-02-15   |
| Revathi | ITSupport Engineer |  19500 |              3 | 2012-10-31   |
+---------+--------------------+--------+----------------+--------------+
5 rows in set (0.00 sec)

mysql>
```

Query-27

அவ்வாறே ஒருவர் தேர்வு செய்யப்பட்ட ஆண்டின் அடிப்படையில் தகவல்களை இறங்குவரிசையில் முறைப்படுத்த *desc* என்பதனை இறுதியில் குறிப்பிட

வேண்டும்.

*select * from employees order by joining_date desc;*

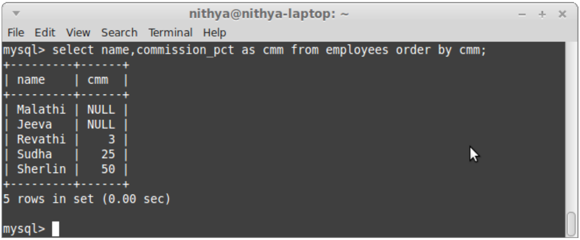


```
nithya@nithya-laptop: ~
File  Edit  View  Search  Terminal  Help
mysql> select * from employees order by joining_date desc;
+---------+--------------------+--------+----------------+--------------+
| name    | role               | salary | commission_pct | joining_date |
+---------+--------------------+--------+----------------+--------------+
| Revathi | ITSupport Engineer |  19500 |              3 | 2012-10-31   |
| Sherlin | Asst. Engineer     |   4500 |             50 | 2011-02-15   |
| Malathi | Java Developer     |   7500 |           NULL | 2009-12-10   |
| Jeeva   | ITSuppot Engineer  |  21500 |           NULL | 2008-01-09   |
| Sudha   | Testing Engineer   |  12000 |             25 | 2007-11-19   |
+---------+--------------------+--------+----------------+--------------+
5 rows in set (0.00 sec)

mysql>
```

Query-28

alias name-ஐப் பயன்படுத்திக் கூட தகவல்களை முறைப்படுத்த முடியும்.

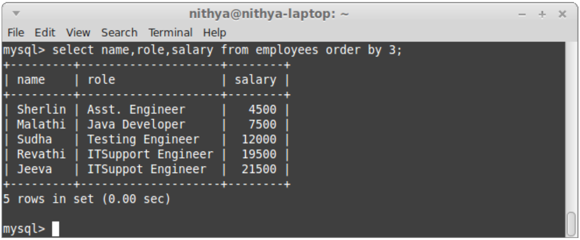
select name,commission_pct as cmm from employees order by cmm;



Query-29

order by-ஐத் தொடர்ந்து *3* எனக் கொடுக்கும் போது, *select statement*-ல் *3*-வதாக அமைந்துள்ள *column*-ன் அடிப்படையில் தகவல்கள் வரிசைப்படுத்தப்படுகின்றன..

select name,role,salary from employees order by 3;



```
nithya@nithya-laptop: ~
File Edit View Search Terminal Help
mysql> select name,role,salary from employees order by 3;
+---------+--------------------+--------+
| name    | role               | salary |
+---------+--------------------+--------+
| Sherlin | Asst. Engineer     |   4500 |
| Malathi | Java Developer     |   7500 |
| Sudha   | Testing Engineer   |  12000 |
| Revathi | ITSupport Engineer |  19500 |
| Jeeva   | ITSuppot Engineer  |  21500 |
+---------+--------------------+--------+
5 rows in set (0.00 sec)

mysql> 
```

Query-30

ஒன்றுக்கும் மேற்பட்ட *columns*-ன் அடிப்படையிலும் நாம் தகவல்களை முறையிட முடியும். பின்வரும் உதாரணத்தில் *department* மூலம் ஏறுவரிசையில் முறைப்படுத்தப்பட்ட தகவல்கள், பின்னர் ஒரே *department*-க்குள், *salary* மூலம் இறங்குவரிசையில் முறைப்படுத்தப்படுகின்றன.

*select * from organisation order by department,salary desc;*

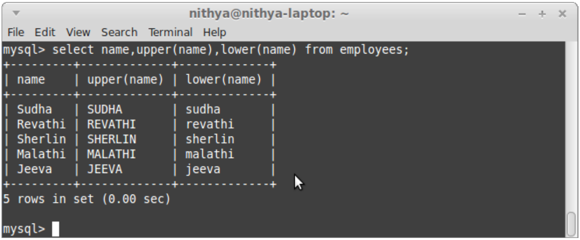
```
nithya@nithya-laptop: ~
File  Edit  View  Search  Terminal  Help
mysql> select * from organisation order by department,salary desc;
+--------+------------+----------------+--------+--------------+
| Emp_ID | Emp_name   | Department     | salary | joining_date |
+--------+------------+----------------+--------+--------------+
|  98598 | Karthika   | Development    |  18550 | 0000-00-00   |
|  98018 | Porkodi    | Development    |  12500 | 0000-00-00   |
|  98020 | Kalaiselvi | Development    |   9750 | 0000-00-00   |
|  98020 | Malathi    | Development    |   5750 | 0000-00-00   |
|  98016 | Kothai     | IT_Finance     |  15000 | 0000-00-00   |
|  98019 | Nalini     | IT_Finance     |   8500 | 0000-00-00   |
|  98017 | Ezhil      | IT_HR          |  25000 | 0000-00-00   |
|  98022 | Jayanthi   | IT_Immigration |  30000 | 0000-00-00   |
|  98021 | Jothi      | Testing        |  18700 | 0000-00-00   |
|  98021 | Renuka     | Testing        |  11725 | 0000-00-00   |
|  98021 | Nirmala    | Testing        |   8700 | 0000-00-00   |
|  98021 | Sangeetha  | Testing        |   7000 | 0000-00-00   |
+--------+------------+----------------+--------+--------------+
12 rows in set (0.00 sec)

mysql> 
```

2.9 Character functions

Query-31

UPPER எழுத்துக்களை பெரிய எழுத்தில் மாற்றிக் காட்டுகிறது. LOWER எழுத்துக்களை சிறிய எழுத்தில் மாற்றிக் காட்டுகிறது. இது பின்வருமாறு.

select name,upper(name),lower(name) from employees;



```
nithya@nithya-laptop: ~
File  Edit  View  Search  Terminal  Help
mysql> select name,upper(name),lower(name) from employees;
+---------+-------------+-------------+
| name    | upper(name) | lower(name) |
+---------+-------------+-------------+
| Sudha   | SUDHA       | sudha       |
| Revathi | REVATHI     | revathi     |
| Sherlin | SHERLIN     | sherlin     |
| Malathi | MALATHI     | malathi     |
| Jeeva   | JEEVA       | jeeva       |
+---------+-------------+-------------+
5 rows in set (0.00 sec)

mysql>
```

Query-32

CONCAT என்பது இரண்டு தனித்தனி *column*-ல் உள்ள மதிப்புகளை ஒன்றாக இணைத்து வெளிப்படுத்துகிறது. *LENGTH* என்பது *column* மதிப்பில் உள்ள எழுத்துக்களின் எண்ணிக்கையை வெளிப்படுத்துகிறது. *INSTR* மூலம் நாம் ஏதேனும் ஒரு எழுத்து மற்றும் *column*-ஐக் கொடுத்து, அந்த எழுத்து *column*-ல் எத்தனையாவதாக உள்ளது

என்பதைக் கண்டு பிடிக்க முடியும். *SUBSTR* மூலம் நாம் ஏதேனும் ஒரு *column* மற்றும் எந்த இடத்திலிருந்து எந்த இடம் வரை எழுத்துக்களை சோதிக்க வேண்டும் எனும் எல்லைகளைக் கொடுத்து அதற்கு ஏற்றார் போல் தகவல்களைப் பெற முடியும். இவை பின்வருமாறு.

```
select
name,role,concat(name,role),length(role),instr(name,'
from employees;

select * from employees where
substr(name,4,7)='athi';
```

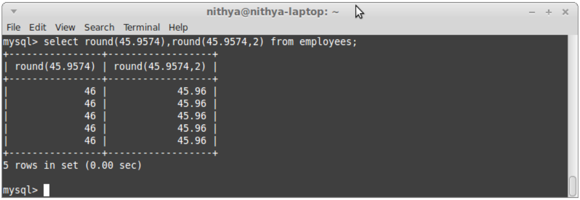
```
nithya@nithya-laptop: ~
File Edit View Search Terminal Help
mysql> select name,role,concat(name,role),length(role),instr(name,'a') from employees;
+---------+--------------------+--------------------------+--------------+-----------------+
| name    | role               | concat(name,role)        | length(role) | instr(name,'a') |
+---------+--------------------+--------------------------+--------------+-----------------+
| Sudha   | Testing Engineer   | SudhaTesting Engineer    |           16 |               5 |
| Revathi | ITSupport Engineer | RevathiITSupport Engineer |          18 |               4 |
| Sherlin | Asst. Engineer     | SherlinAsst. Engineer    |           14 |               0 |
| Malathi | Java Developer     | MalathiJava Developer    |           14 |               2 |
| Jeeva   | ITSuppot Engineer  | JeevaITSuppot Engineer   |           17 |               5 |
+---------+--------------------+--------------------------+--------------+-----------------+
5 rows in set (0.00 sec)

mysql> select * from employees where substr(name,4,7)='athi';
+---------+--------------------+--------+----------------+--------------+
| name    | role               | salary | commission_pct | joining_date |
+---------+--------------------+--------+----------------+--------------+
| Revathi | ITSupport Engineer |  19500 |              3 | 2012-10-31   |
| Malathi | Java Developer     |   7500 |           NULL | 2009-12-10   |
+---------+--------------------+--------+----------------+--------------+
2 rows in set (0.01 sec)

mysql> 
```

2.10 Number functions

Query-33

ROUND-ஆனது தசம எண்களை அதற்கு நெருங்கிய முழு எண்களாக மாற்றும். *ROUND(any column, 2)* எனக் கொடுக்கும் போது அந்த *column*-ல் உள்ள தசம எண்களை இரண்டு தசம இலக்கத்தில் வெளிப்படுத்தும்.

select round(45.9574),round(45.9574,2) from employees;



```
mysql> select round(45.9574),round(45.9574,2) from employees;
+----------------+------------------+
| round(45.9574) | round(45.9574,2) |
+----------------+------------------+
|             46 |            45.96 |
|             46 |            45.96 |
|             46 |            45.96 |
|             46 |            45.96 |
|             46 |            45.96 |
+----------------+------------------+
5 rows in set (0.00 sec)

mysql>
```

3

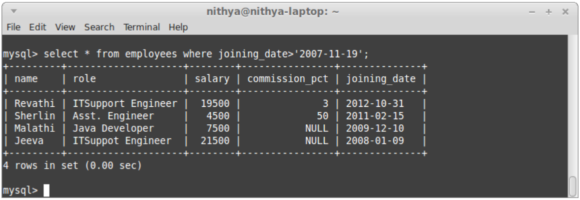
Working with Dates - தேதிகளைக் கையாளுதல்

Query-34

உதாரணத்துக்கு ஒரு நிறுவனத்தில் *November 19, 2007*-க்கு மேல் வேலைக்கு சேர்ந்த அனைத்து நபர்களையும் பட்டியலிட, அந்த தேதியை *condition*-ல் கொடுத்தால் போதுமானது. தானாகவே அதற்கு மேலுள்ள தேதியில் சேர்ந்த அனைவரின் பெயர்களும் பட்டியலிடப் பட்டுவிடும்.

*select * from employees where*

joining_date>'2007-11-19';



```
nithya@nithya-laptop: ~
File Edit View Search Terminal Help

mysql> select * from employees where joining_date>'2007-11-19';
+---------+--------------------+--------+----------------+--------------+
| name    | role               | salary | commission_pct | joining_date |
+---------+--------------------+--------+----------------+--------------+
| Revathi | ITSupport Engineer |  19500 |              3 | 2012-10-31   |
| Sherlin | Asst. Engineer     |   4500 |             50 | 2011-02-15   |
| Malathi | Java Developer     |   7500 |           NULL | 2009-12-10   |
| Jeeva   | ITSuppot Engineer  |  21500 |           NULL | 2008-01-09   |
+---------+--------------------+--------+----------------+--------------+
4 rows in set (0.00 sec)

mysql>
```

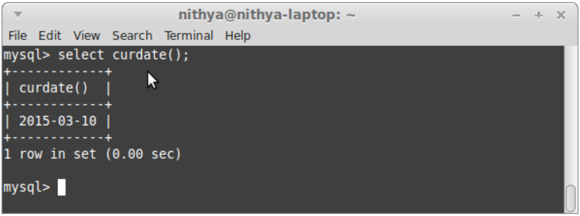
Query-35

SYSDATE என்பது தற்போதைய தேதி மற்றும் நேரத்தை வெளிப்படுத்தும் ஒரு *date function* ஆகும்.

select sysdate();

CURDATE என்பது தற்போதைய தேதியை மட்டும் வெளிப்படுத்தும்.

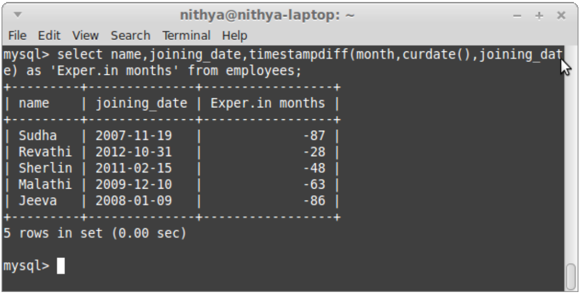
select curdate();



Query-36

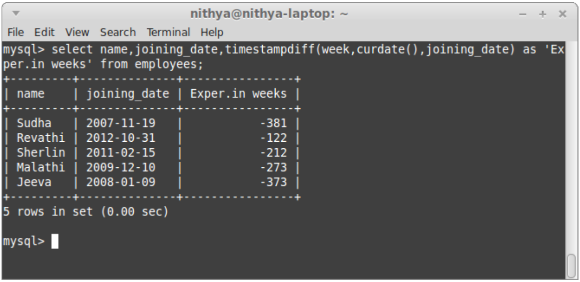
நமது நிறுவனத்தில் ஒருவர் சேர்ந்து எத்தனை காலங்கள் ஆகியுள்ளன என்பதை *timestampdiff* எனும் *function* மூலம் கண்டுபிடிக்கலாம். எத்தனை மாதம் என்பதைக் கண்டுபிடிக்க அந்த *function*-க்குள் *month* எனவும், எத்தனை வாரங்கள் என்பதைக் கண்டுபிடிக்க *week* எனவும், எத்தனை நாட்கள் என்பதனை *date* எனவும் கொடுத்து கண்டுபிடிக்க முடியும். இவை பின்வருமாறு.

```
select
name,joining_date,timestampdiff(month,curdate(),join
as 'Exper.in months' from employees;
```



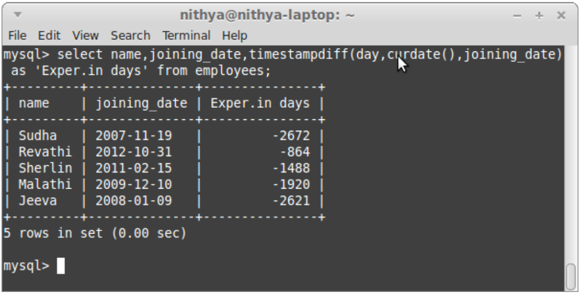
```
nithya@nithya-laptop: ~
File Edit View Search Terminal Help
mysql> select name,joining_date,timestampdiff(month,curdate(),joining_dat
e) as 'Exper.in months' from employees;
+---------+--------------+-----------------+
| name    | joining_date | Exper.in months |
+---------+--------------+-----------------+
| Sudha   | 2007-11-19   |             -87 |
| Revathi | 2012-10-31   |             -28 |
| Sherlin | 2011-02-15   |             -48 |
| Malathi | 2009-12-10   |             -63 |
| Jeeva   | 2008-01-09   |             -86 |
+---------+--------------+-----------------+
5 rows in set (0.00 sec)

mysql>
```

select
name,joining_date,timestampdiff(week,curdate(),joinin
as 'Exper.in weeks' from employees;



```
nithya@nithya-laptop: ~
File  Edit  View  Search  Terminal  Help
mysql> select name,joining_date,timestampdiff(week,curdate(),joining_date) as 'Ex
per.in weeks' from employees;
+---------+--------------+-----------------+
| name    | joining_date | Exper.in weeks  |
+---------+--------------+-----------------+
| Sudha   | 2007-11-19   |            -381 |
| Revathi | 2012-10-31   |            -122 |
| Sherlin | 2011-02-15   |            -212 |
| Malathi | 2009-12-10   |            -273 |
| Jeeva   | 2008-01-09   |            -373 |
+---------+--------------+-----------------+
5 rows in set (0.00 sec)

mysql>
```

select name,joining_date,timestampdiff(day,curdate(),joining
as 'Exper.in days' from employees;



3.1 Date functions

ஒரு சில *date functions*, ஒரு *column*-ல் உள்ள மதிப்புகளின் மீது செயல்பட்டு, அதனை எவ்வாறு மாற்றுகின்றன என்பதனைப் பின்வரும் படத்தில் காணலாம்.

DOB	ADDDATE (DOB,15) →	DOB
1983-02-18		1983-03-05
2014-12-26		2015-01-10
1987-08-04		1987-08-19

DOB	ADDTIME (DOB,'05:30:00') →	DOB
1983-02-18 17:30:45		1983-02-18 23:00:45
2014-12-26 12:12:54		2014-12-26 17:42:54
1987-08-04 04:20:30		1987-08-04 09:50:30

DOB	DATE (DOB) →	DOB
1983-02-18 17:30:45		1983-02-18
2014-12-26 12:12:54		2014-12-26
1987-08-04 04:20:30		1987-08-04

DOB	DATE_FORMAT(DOB, '%D %W %M %Y') →	DOB
1983-02-18 17:30:45		18th Friday February 1983
2014-12-26 12:12:54		26th Friday December 2014
1987-08-04 04:20:30		4th Tuesday August 1987

DOB	DATE_FORMAT(DOB, '%d %w %m %y') →	DOB
1983-02-18 17:30:45		18 5 02 83
2014-12-26 12:12:54		26 5 12 14
1987-08-04 04:20:30		04 2 08 87

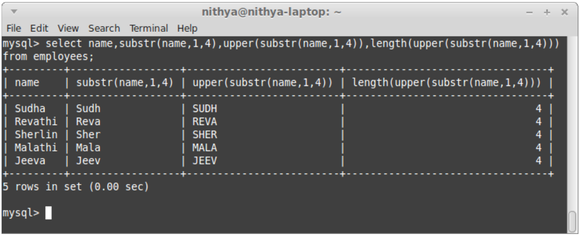
DOB	CONVERT_TZ (DOB, 'GMT','MET') →	DOB
1983-02-18 17:30:45		1983-02-18 18:30:45
2014-12-26 12:12:54		2014-12-26 13:12:54
1987-08-04 04:20:30		1987-08-04 06:20:30

3.2 *Nesting of functions*

Query-37

ஒன்றுக்கும் மேற்பட்ட *functions*-ஐ ஒன்றன் மீது ஒன்றாக இணைத்து வெளிப்படுத்துவது *Nesting of functions* எனப்படும். பின்வரும் உதாரணத்தில், *length, upper, substr* எனும் மூன்று *functions*-ம் ஒன்றன்மீது ஒன்றாக செயல்பட்டுள்ளன.

select name,substr(name,1,4),upper(substr(name,1,4)),length from employees;



```
nithya@nithya-laptop: ~
File Edit View Search Terminal Help
mysql> select name,substr(name,1,4),upper(substr(name,1,4)),length(upper(substr(name,1,4)))
from employees;
+---------+------------------+-------------------------+---------------------------------+
| name    | substr(name,1,4) | upper(substr(name,1,4)) | length(upper(substr(name,1,4))) |
+---------+------------------+-------------------------+---------------------------------+
| Sudha   | Sudh             | SUDH                    |                               4 |
| Revathi | Reva             | REVA                    |                               4 |
| Sherlin | Sher             | SHER                    |                               4 |
| Malathi | Mala             | MALA                    |                               4 |
| Jeeva   | Jeev             | JEEV                    |                               4 |
+---------+------------------+-------------------------+---------------------------------+
5 rows in set (0.00 sec)

mysql>
```

4

Conditional Expressions

4.1 Case Statement

Query-38

CASE என்பது ஒரு *column*-ல் உள்ள வெவ்வேறு மதிப்புகளுக்கு வெவ்வேறு விதமான செயல்களைச் செய்யுமாறு ஆணைகளை அளிக்கப் பயன்படுகிறது. உதாரணத்துக்கு ஒரு நிறுவனத்தில் *development department*-க்கு 50% சம்பள உயர்வும், *testing department*-க்கு 30% சம்பள உயர்வும், மற்றவர்களுக்கு 15% சம்பள உயர்வும், அந்த

நிறுவனம் அளிக்கிறது எனில், ஒவ்வொருவருடைய புதிய சம்பளத்தையும் கண்டுபிடிப்பதற்கான *query* பின்வருமாறு அமையும்.

SELECT emp_name, department, salary, CASE

*WHEN Department='Testing' THEN (salary+ (salary*0.50))*

*WHEN Department='Development' THEN (salary+(salary*0.30))*

*ELSE (salary+(salary*0.15))*

END AS 'new salary'

FROM organisation;

```
nithya@nithya-laptop: ~
File  Edit  View  Search  Terminal  Help
mysql> SELECT emp_name, department, salary, CASE
    -> WHEN Department='Testing' THEN (salary+(salary*0.50))
    -> WHEN Department='Development' THEN (salary+(salary*0.30))
    -> ELSE (salary+(salary*0.15))
    -> END AS 'new salary'
    -> FROM organisation;
+------------+----------------+--------+------------+
| emp_name   | department     | salary | new salary |
+------------+----------------+--------+------------+
| Kothai     | IT_Finance     |  15000 |   17250.00 |
| Ezhil      | IT_HR          |  25000 |   28750.00 |
| Porkodi    | Development    |  11725 |   15242.50 |
| Karthika   | Development    |  18550 |   24115.00 |
| Nalini     | IT_Finance     |   8500 |    9775.00 |
| Kalaiselvi | Development    |   9750 |   12675.00 |
| Malathi    | Development    |   5750 |    7475.00 |
| Jothi      | Testing        |  18700 |   28050.00 |
| Renuka     | Testing        |  11725 |   17587.50 |
| Nirmala    | Testing        |   8700 |   13050.00 |
| Sangeetha  | Testing        |   7000 |   10500.00 |
| Jayanthi   | IT_Immigration |  30000 |   34500.00 |
+------------+----------------+--------+------------+
12 rows in set (0.01 sec)

mysql>
```

4.2 Logical Operators

Query-39

இரண்டு தனித்தனி கட்டளைகளை ஒன்றாக இணைத்து அதனடிப்படையில் விவரங்களைப் பட்டியலிட *logical operator* பயன்படுகிறது.

உதாரணத்துக்கு 2013-ஆம் ஆண்டு தேர்வு செய்யப்பட்டு, 15,000 ரூபாய்க்கு கீழ் சம்பளம் வாங்கும் நபர்களைப் பட்டியலிட *AND* என்ற *logical operator*-ஐப் பின்வருமாறு அமைக்கலாம்.

```
select * from organisation where
year(joining_date) = 2013 and salary < 15000;
```

```
mysql> select * from organisation;
+--------+------------+----------------+--------+--------------+
| Emp_ID | Emp_name   | Department     | salary | joining_date |
+--------+------------+----------------+--------+--------------+
|  98016 | Kothai     | IT_Finance     |  15000 | 2013-10-12   |
|  98017 | Ezhil      | IT_HR          |  25000 | 2013-10-12   |
|  98018 | Porkodi    | Development    |  12500 | 2012-02-15   |
|  98598 | Karthika   | Development    |  18550 | 2010-11-18   |
|  98019 | Nalini     | IT_Finance     |   8500 | 2015-03-01   |
|  98020 | Kalaiselvi | Development    |   9750 | 2014-06-28   |
|  98020 | Malathi    | Development    |   5750 | 2009-11-26   |
|  98021 | Jothi      | Testing        |  18700 | 2011-12-10   |
|  98021 | Renuka     | Testing        |  11725 | 2013-07-04   |
|  98021 | Nirmala    | Testing        |   8700 | 2015-03-01   |
|  98021 | Sangeetha  | Testing        |   7000 | 2011-08-04   |
|  98022 | Jayanthi   | IT_Immigration |  30000 | 2013-10-12   |
+--------+------------+----------------+--------+--------------+
12 rows in set (0.00 sec)

mysql> select * from organisation where year(joining_date) = 2013 and salary
<15000;
+--------+----------+------------+--------+--------------+
| Emp_ID | Emp_name | Department | salary | joining_date |
+--------+----------+------------+--------+--------------+
|  98021 | Renuka   | Testing    |  11725 | 2013-07-04   |
+--------+----------+------------+--------+--------------+
1 row in set (0.00 sec)

mysql>
```

Query-40

அவ்வாறே 2013-ஆம் ஆண்டு தேர்வு செய்யப்பட்ட நபர் அல்லது 15,000 ரூபாய்க்கு கீழ் சம்பளம் வாங்கும் நபர் என்று கட்டளையை மாற்றி அமைக்க *OR operator* பயன்படுகிறது. இது பின்வருமாறு.

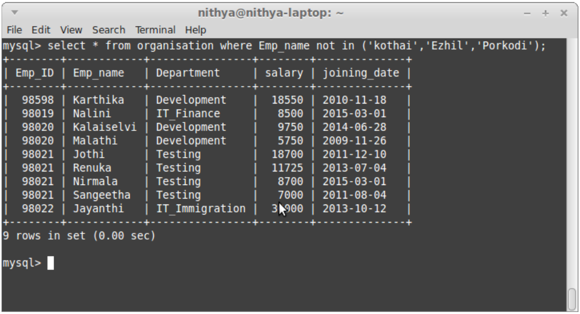
*select * from organisation where year(joining_date) = 2013 or salary < 15000;*

```
nithya@nithya-laptop: ~
File  Edit  View  Search  Terminal  Help
mysql> select * from organisation where year(joining_date) = 2013 or salary<15000;
+--------+------------+----------------+--------+--------------+
| Emp_ID | Emp_name   | Department     | salary | joining_date |
+--------+------------+----------------+--------+--------------+
|  98016 | Kothai     | IT_Finance     |  15000 | 2013-10-12   |
|  98017 | Ezhil      | IT_HR          |  25000 | 2013-10-12   |
|  98018 | Porkodi    | Development    |  12500 | 2012-02-15   |
|  98019 | Nalini     | IT_Finance     |   8500 | 2015-03-01   |
|  98020 | Kalaiselvi | Development    |   9750 | 2014-06-28   |
|  98020 | Malathi    | Development    |   5750 | 2009-11-26   |
|  98021 | Renuka     | Testing        |  11725 | 2013-07-04   |
|  98021 | Nirmala    | Testing        |   8700 | 2015-03-01   |
|  98021 | Sangeetha  | Testing        |   7000 | 2011-08-04   |
|  98022 | Jayanthi   | IT_Immigration |  30000 | 2013-10-12   |
+--------+------------+----------------+--------+--------------+
10 rows in set (0.00 sec)

mysql>
```

Query-41

Kothai, Ezhil, Porkodi -ஆகிய மூன்று நபர்களைத் தவிர மற்ற நபர்களைப் பட்டியலிட *NOT IN operator* பயன்படுகிறது.

*select * from organisation where Emp_name not in ('kothai','Ezhil','Porkodi');*

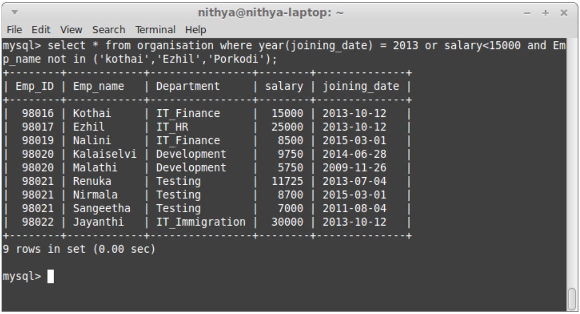

```
mysql> select * from organisation where Emp_name not in ('kothai','Ezhil','Porkodi');
+--------+------------+----------------+--------+--------------+
| Emp_ID | Emp_name   | Department     | salary | joining_date |
+--------+------------+----------------+--------+--------------+
|  98598 | Karthika   | Development    |  18550 | 2010-11-18   |
|  98019 | Nalini     | IT_Finance     |   8500 | 2015-03-01   |
|  98020 | Kalaiselvi | Development    |   9750 | 2014-06-28   |
|  98020 | Malathi    | Development    |   5750 | 2009-11-26   |
|  98021 | Jothi      | Testing        |  18700 | 2011-12-10   |
|  98021 | Renuka     | Testing        |  11725 | 2013-07-04   |
|  98021 | Nirmala    | Testing        |   8700 | 2015-03-01   |
|  98021 | Sangeetha  | Testing        |   7000 | 2011-08-04   |
|  98022 | Jayanthi   | IT_Immigration | 3[illegible]900 | 2013-10-12   |
+--------+------------+----------------+--------+--------------+
9 rows in set (0.00 sec)

mysql>
```

Query-42

இரண்டுக்கும் மேற்பட்ட கட்டளைகளை எவ்வாறு

இணைப்பது என்று இதில் பார்ப்போம். 2013-ஆம் ஆண்டு தேர்வு செய்யப்பட்ட நபர் அல்லது 15,000 ரூபாய்க்கு கீழ் சம்பளம் வாங்கும் நபர் என்பது முதல் இரண்டு கட்டளைகள். பின்னர் அது தரும் *result*-ல், *kothai*, *ezhil*, *porkodi* ஆகிய மூவரின் விவரங்களைத் தவிர்க்கவும் என்பது மூன்றாவது கட்டளை. இவற்றை இணைத்து பின்வருமாறு *query*-யை அமைக்கவும்.

```
select * from organisation where
year(joining_date) = 2013 or salary<15000 and
Emp_name not in ('kothai','Ezhil','Porkodi');
```



```
nithya@nithya-laptop: ~
File Edit View Search Terminal Help
mysql> select * from organisation where year(joining_date) = 2013 or salary<15000 and Em
p_name not in ('kothai','Ezhil','Porkodi');
+--------+------------+----------------+--------+--------------+
| Emp_ID | Emp_name   | Department     | salary | joining_date |
+--------+------------+----------------+--------+--------------+
|  98016 | Kothai     | IT_Finance     |  15000 | 2013-10-12   |
|  98017 | Ezhil      | IT_HR          |  25000 | 2013-10-12   |
|  98019 | Nalini     | IT_Finance     |   8500 | 2015-03-01   |
|  98020 | Kalaiselvi | Development    |   9750 | 2014-06-28   |
|  98020 | Malathi    | Development    |   5750 | 2009-11-26   |
|  98021 | Renuka     | Testing        |  11725 | 2013-07-04   |
|  98021 | Nirmala    | Testing        |   8700 | 2015-03-01   |
|  98021 | Sangeetha  | Testing        |   7000 | 2011-08-04   |
|  98022 | Jayanthi   | IT_Immigration |  30000 | 2013-10-12   |
+--------+------------+----------------+--------+--------------+
9 rows in set (0.00 sec)

mysql>
```

இது விடைகளைத் தவறாகப் பட்டியலிடுவதைக் காணலாம். எனவே எந்த இரு கட்டளைகள் முதலில் ஒப்பிடப்பட வேண்டுமோ அதனை அடைப்புக்குறிக்குள் கொடுக்க வேண்டும். பின்னர் மூன்றாவது கட்டளையை இணைக்க வேண்டும் இதுவே '*Operator Precedence*' ஆகும்.

*select * from organisation where (year(joining_date) = 2013 or salary<15000) and Emp_name not in ('kothai','Ezhil','Porkodi');*

nithya@nithya-laptop: ~

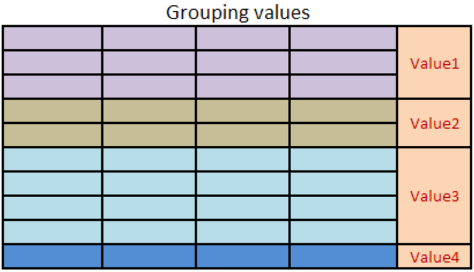
File Edit View Search Terminal Help

```
mysql> select * from organisation where (year(joining_date) = 2013 or salary<15000) and
Emp_name not in ('kothai','Ezhil','Porkodi');
+--------+------------+----------------+--------+--------------+
| Emp_ID | Emp_name   | Department     | salary | joining_date |
+--------+------------+----------------+--------+--------------+
|  98019 | Nalini     | IT_Finance     |   8500 | 2015-03-01   |
|  98020 | Kalaiselvi | Development    |   9750 | 2014-06-28   |
|  98020 | Malathi    | Development    |   5750 | 2009-11-26   |
|  98021 | Renuka     | Testing        |  11725 | 2013-07-04   |
|  98021 | Nirmala    | Testing        |   8700 | 2015-03-01   |
|  98021 | Sangeetha  | Testing        |   7000 | 2011-08-04   |
|  98022 | Jayanthi   | IT_Immigration |  30000 | 2013-10-12   |
+--------+------------+----------------+--------+--------------+
7 rows in set (0.00 sec)

mysql>
```

5

Groups

Grouping எவ்வாறு நடைபெறுகிறது என்பதைப் பின்வரும் படத்தின் மூலம் தெளிவாகப் புரிந்து கொள்ளலாம். அதாவது ஏதோ ஒரு விதத்தில் ஒரே மாதிரியான தகவல்கள் *group* செய்யப்பட்டு மதிப்புகள் வெளிப்படுத்தப் படுகின்றன.



Mysql-ல் உள்ள ஒருசில *grouping functions*-ஐப் பின்வருமாறு காணலாம்.

5.1 *Group functions*

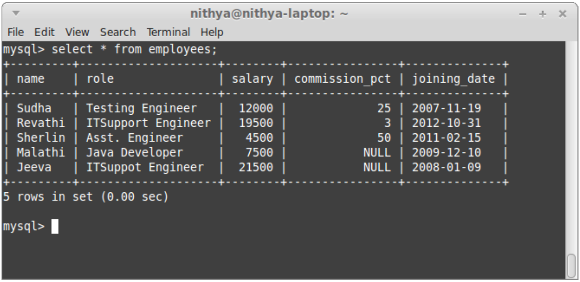
Query-43

ஒரு *column*-ல் உள்ள மதிப்புகளில் *MIN()* என்பது

மிகச்சிறிய மதிப்பிணையும், *MAX()* என்பது மிகப்பெரிய மதிப்பிணையும், *SUM()* என்பது அனைத்து மதிப்புகளின் கூட்டுத் தொகையையும், *AVG()* என்பது அதன் சராசரி மதிப்யையும் வெளிப்படுத்துகிறது. இவை பின்வருமாறு.

select min(salary),max(salary) from employees;

select sum(salary),avg(salary) from employees;



```
nithya@nithya-laptop: ~
File Edit View Search Terminal Help
mysql> select * from employees;
+---------+--------------------+--------+----------------+--------------+
| name    | role               | salary | commission_pct | joining_date |
+---------+--------------------+--------+----------------+--------------+
| Sudha   | Testing Engineer   |  12000 |             25 | 2007-11-19   |
| Revathi | ITSupport Engineer |  19500 |              3 | 2012-10-31   |
| Sherlin | Asst. Engineer     |   4500 |             50 | 2011-02-15   |
| Malathi | Java Developer     |   7500 |           NULL | 2009-12-10   |
| Jeeva   | ITSuppot Engineer  |  21500 |           NULL | 2008-01-09   |
+---------+--------------------+--------+----------------+--------------+
5 rows in set (0.00 sec)

mysql>
```

```
nithya@nithya-laptop: ~
File Edit View Search Terminal Help
mysql> select min(salary),max(salary) from employees;
+-------------+-------------+
| min(salary) | max(salary) |
+-------------+-------------+
|        4500 |       21500 |
+-------------+-------------+
1 row in set (0.00 sec)

mysql> select sum(salary),avg(salary) from employees;
+-------------+-------------+
| sum(salary) | avg(salary) |
+-------------+-------------+
|       65000 |  13000.0000 |
+-------------+-------------+
1 row in set (0.00 sec)

mysql> 
```

Query-44

மேற்கூறிய அதே *functions*-ஐ *date* மற்றும் *characters*-ன் மீதும் செலுத்தலாம். இது பின்வருமாறு.

```
mysql> select min(name),max(name) from employees;
+-----------+-----------+
| min(name) | max(name) |
+-----------+-----------+
| Jeeva     | Sudha     |
+-----------+-----------+
1 row in set (0.01 sec)

mysql> select min(joining_date),max(joining_date) from employees;
+-------------------+-------------------+
| min(joining_date) | max(joining_date) |
+-------------------+-------------------+
| 2007-11-19        | 2012-10-31        |
+-------------------+-------------------+
1 row in set (0.00 sec)

mysql>
```

Query-45

COUNT() என்பது அதற்குள் * ஐப் பெற்றிருப்பின் அந்த *table*-ல் உள்ள *rows*-ன் எண்ணிக்கையையும், அதற்குள் ஏதேனும் ஒரு *column*- ஐப் பெற்றிருப்பின் *null* மதிப்பிணைத் தவிர்த்துவிட்டு அந்த *column* பெற்றுள்ள *rows*-ன் எண்ணிக்கையையும் வெளிப்படுத்தும்.

```
nithya@nithya-laptop: ~
File Edit View Search Terminal Help
mysql> select count(*) from employees;
+----------+
| count(*) |
+----------+
|        5 |
+----------+
1 row in set (0.00 sec)

mysql> select count(commission_pct) from employees;
+-----------------------+
| count(commission_pct) |
+-----------------------+
|                     3 |
+-----------------------+
1 row in set (0.00 sec)

mysql> 
```

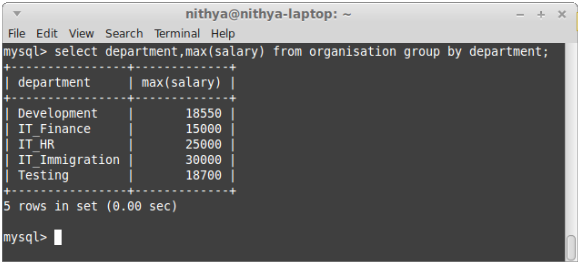
5.2 Grouping rows

Query-46

GROUP BY என்பது group functions-ஐப் பயன்படுத்தும் queries-ல் மட்டுமே காணப்படும். அதாவது SELECT list-ல் காணப்படும் group functions தவிர மற்ற அனைத்து columns-ம்

GROUP BY-ஐத் தொடர்ந்து எழுதப்பட வேண்டும். உதாரணத்துக்கு ஒரு நிறுவனத்தின் ஒவ்வொரு *department*-லும் ஒருவர் வாங்கும் அதிக பட்ச சம்பளத்தைத் தெரிந்துகொள்ள, *SELECT*-ல் *department, max(salary)* என எழுதிவிட்டு பின்னர் *GROUP BY*-ஐத் தொடர்ந்து *department* எனக் குறிப்பிட வேண்டும். அப்போதுதான் அதிகபட்ச சம்பளம் ஒவ்வொரு *department*-க்கும் ஒருங்கிணைக்கப்பட்டு வெளிப்படுத்தப்படும்.

```
select department,max(salary) from organisation
group by department;
```

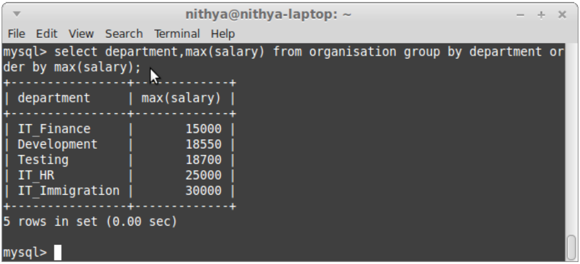


```
nithya@nithya-laptop: ~
File Edit View Search Terminal Help
mysql> select department,max(salary) from organisation group by department;
+----------------+-------------+
| department     | max(salary) |
+----------------+-------------+
| Development    |       18550 |
| IT_Finance     |       15000 |
| IT_HR          |       25000 |
| IT_Immigration |       30000 |
| Testing        |       18700 |
+----------------+-------------+
5 rows in set (0.00 sec)

mysql>
```

Query-47

group functions-ஐ நாம் *order by*-ல் கொடுத்து, தகவல்களை ஏறுவரிசையிலோ அல்லது இறங்குவரிசையிலோ முறைப்படுத்திப் பார்க்கவும் முடியும்.

select department,max(salary) from organisation group by department order by max(salary);

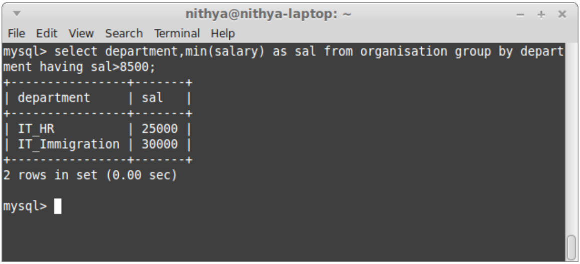


```
nithya@nithya-laptop: ~
File Edit View Search Terminal Help
mysql> select department,max(salary) from organisation group by department or
der by max(salary);
+----------------+-------------+
| department     | max(salary) |
+----------------+-------------+
| IT_Finance     |       15000 |
| Development    |       18550 |
| Testing        |       18700 |
| IT_HR          |       25000 |
| IT_Immigration |       30000 |
+----------------+-------------+
5 rows in set (0.00 sec)

mysql> 
```

Query-48

group functions-ஐ நாம் *condition*-ல் கொடுத்து, அதனடிப்படையில் நாம் தகவல்களைப் பெறவும் முடியும். பொதுவாக *WHERE*-ஐத் தொடர்ந்து *conditions*-ஐக் குறிப்பிடுவோம் அல்லவா? ஆனால் *group functions*-ஐ *condition*-ஆகப் பயன்படுத்தும் போது மட்டும் *WHERE*-க்கு பதிலாக *HAVING* எனக் கொடுக்க வேண்டும்.

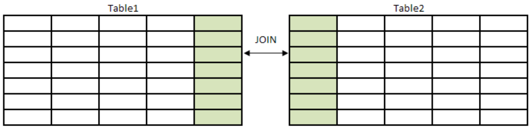
select department,min(salary) as sal from organisation group by department having sal>8500;



6

JOIN

இரண்டு வெவ்வேறு *table*-ல் இருக்கும் ஒரு பொதுவான *column*-ஐப் பயன்படுத்தி அவற்றை இணைத்து, அதன்பின் இரண்டிலிருந்தும் தகவல்களைப் பெறுவதற்கு *JOIN* பயன்படுகிறது.



இதனை *Inner Join, Outer Join, Cross Join* என்று மூன்று வகையாகப் பிரிக்கலாம். *Outer Join*-ஐ *left outer, right outer* என்று இரண்டு வகையாகப் பிரிக்கலாம். இவை எவ்வாறு இணைந்து தகவல்களை வெளிப்படுத்துகின்றன என்பதைப் பின்வரும் படத்தின் மூலம் புரிந்து கொள்ளலாம்.

Table1

Table2

Inner Join

Left Outer Join

			Null	Null
			Null	Null

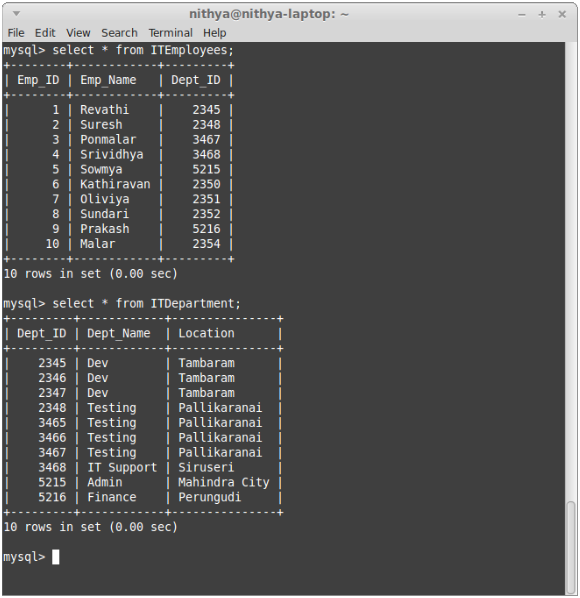
Right Outer Join

	Null	Null		
	Null	Null		

Full Outer Join

			Null	Null
			Null	Null
	Null	Null		
	Null	Null		

ITEmployees, ITDepartment எனும் *tables*-ல் உள்ள தகவல்களைப் பின்வரும் படத்தில் காணலாம். பின்னர் இவை எவ்வாறு இணைந்து தகவல்களை வெளிப்படுத்துகின்றன என்பதை ஒவ்வொன்றாகப் பார்க்கலாம்.



```
nithya@nithya-laptop: ~
File Edit View Search Terminal Help
mysql> select * from ITEmployees;
+--------+------------+---------+
| Emp_ID | Emp_Name   | Dept_ID |
+--------+------------+---------+
|      1 | Revathi    |    2345 |
|      2 | Suresh     |    2348 |
|      3 | Ponmalar   |    3467 |
|      4 | Srividhya  |    3468 |
|      5 | Sowmya     |    5215 |
|      6 | Kathiravan |    2350 |
|      7 | Oliviya    |    2351 |
|      8 | Sundari    |    2352 |
|      9 | Prakash    |    5216 |
|     10 | Malar      |    2354 |
+--------+------------+---------+
10 rows in set (0.00 sec)

mysql> select * from ITDepartment;
+---------+------------+---------------+
| Dept_ID | Dept_Name  | Location      |
+---------+------------+---------------+
|    2345 | Dev        | Tambaram      |
|    2346 | Dev        | Tambaram      |
|    2347 | Dev        | Tambaram      |
|    2348 | Testing    | Pallikaranai  |
|    3465 | Testing    | Pallikaranai  |
|    3466 | Testing    | Pallikaranai  |
|    3467 | Testing    | Pallikaranai  |
|    3468 | IT Support | Siruseri      |
|    5215 | Admin      | Mahindra City |
|    5216 | Finance    | Perungudi     |
+---------+------------+---------------+
10 rows in set (0.00 sec)

mysql>
```

6.1 *Inner Join*

Query-49

இதில் *INNER JOIN* என்பது இணைப்பிற்காக நாம் பயன்படுத்தியிருக்கும் பொதுவான *column*-ன் மதிப்பு இரண்டு *table*-களிலும் ஒன்றாக இருந்தால் மட்டுமே தகவல்களை வெளிப்படுத்தும். இதனை *Equi Join* எனவும் அழைக்கலாம்.

*select * from ITEmployees e inner join ITDepartment d on e.dept_id=d.dept_id;*

```
nithya@nithya-laptop: ~
File  Edit  View  Search  Terminal  Help
mysql> select * from ITEmployees e inner join ITDepartment d on e.dept_id=d.dept
_id;
+--------+----------+---------+---------+------------+---------------+
| Emp_ID | Emp_Name | Dept_ID | Dept_ID | Dept_Name  | Location      |
+--------+----------+---------+---------+------------+---------------+
|      1 | Revathi  |    2345 |    2345 | Dev        | Tambaram      |
|      2 | Suresh   |    2348 |    2348 | Testing    | Pallikaranai  |
|      3 | Ponmalar |    3467 |    3467 | Testing    | Pallikaranai  |
|      4 | Srividhya|    3468 |    3468 | IT Support | Siruseri      |
|      5 | Sowmya   |    5215 |    5215 | Admin      | Mahindra City |
|      9 | Prakash  |    5216 |    5216 | Finance    | Perungudi     |
+--------+----------+---------+---------+------------+---------------+
6 rows in set (0.00 sec)

mysql>
```

6.2 Outer Join

இதனை *Left Outer, Right Outer, Full Outer* என்று மூன்று வகையாகப் பிரிக்கலாம்.

Query-50

LEFT OUTER JOIN என்பது இடப்பக்கம் இருக்கும் *table*-லிருந்து அனைத்து தகவல்களையும் வெளிப்படுத்தும். பின்னர் இணைப்பிற்காக நாம் பயன்படுத்தியிருக்கும் பொதுவான *column*-ன் மதிப்பு, வலப்பக்கத்து மதிப்புடன் ஒத்துப் போனால் மட்டுமே வலப்பக்கத்திலிருந்து தகவல்களை எடுத்து வெளிப்படுத்தும். அப்படி இல்லையென்றால் *Null* மதிப்பினை வெளிப்படுத்தும்.

*select * from ITEmployees e left outer join ITDepartment d on e.dept_id=d.dept_id;*

```
mysql> select * from ITEmployees e left outer join ITDepartment d on e.dept_id=d
.dept_id;
+--------+------------+---------+---------+------------+---------------+
| Emp_ID | Emp_Name   | Dept_ID | Dept_ID | Dept_Name  | Location      |
+--------+------------+---------+---------+------------+---------------+
|      1 | Revathi    |    2345 |    2345 | Dev        | Tambaram      |
|      2 | Suresh     |    2348 |    2348 | Testing    | Pallikaranai  |
|      3 | Ponmalar   |    3467 |    3467 | Testing    | Pallikaranai  |
|      4 | Srividhya  |    3468 |    3468 | IT Support | Siruseri      |
|      5 | Sowmya     |    5215 |    5215 | Admin      | Mahindra City |
|      6 | Kathiravan |    2350 |    NULL | NULL       | NULL          |
|      7 | Oliviya    |    2351 |    NULL | NULL       | NULL          |
|      8 | Sundari    |    2352 |    NULL | NULL       | NULL          |
|      9 | Prakash    |    5216 |    5216 | Finance    | Perungudi     |
|     10 | Malar      |    2354 |    NULL | NULL       | NULL          |
+--------+------------+---------+---------+------------+---------------+
10 rows in set (0.00 sec)

mysql>
```

Query-51

RIGHT OUTER JOIN என்பது வலப்பக்கம் இருக்கும் *table*-லிருந்து அனைத்து தகவல்களையும் வெளிப்படுத்தும். பின்னர் இணைப்பிற்காக நாம் பயன்படுத்தியிருக்கும் பொதுவான *column*-ன் மதிப்பு, இடப்பக்கத்து மதிப்புடன் ஒத்துப் போனால் மட்டுமே இடப்பக்கத்திலிருந்து தகவல்களை எடுத்து வெளிப்படுத்தும். அப்படி இல்லையென்றால் *Null*

மதிப்பினை வெளிப்படுத்தும்.

*select * from ITEmployees e right outer join ITDepartment d on e.dept_id=d.dept_id;*

```
nithya@nithya-laptop: ~
File Edit View Search Terminal Help

mysql> select * from ITEmployees e right outer join ITDepartment d on e.dept_id=
d.dept_id;
+--------+-----------+---------+---------+------------+---------------+
| Emp_ID | Emp_Name  | Dept_ID | Dept_ID | Dept_Name  | Location      |
+--------+-----------+---------+---------+------------+---------------+
|      1 | Revathi   |    2345 |    2345 | Dev        | Tambaram      |
|   NULL | NULL      |    NULL |    2346 | Dev        | Tambaram      |
|   NULL | NULL      |    NULL |    2347 | Dev        | Tambaram      |
|      2 | Suresh    |    2348 |    2348 | Testing    | Pallikaranai  |
|   NULL | NULL      |    NULL |    3465 | Testing    | Pallikaranai  |
|   NULL | NULL      |    NULL |    3466 | Testing    | Pallikaranai  |
|      3 | Ponmalar  |    3467 |    3467 | Testing    | Pallikaranai  |
|      4 | Srividhya |    3468 |    3468 | IT Support | Siruseri      |
|      5 | Sowmya    |    5215 |    5215 | Admin      | Mahindra City |
|      9 | Prakash   |    5216 |    5216 | Finance    | Perungudi     |
+--------+-----------+---------+---------+------------+---------------+
10 rows in set (0.00 sec)

mysql>
```

Query-52

FULL OUTER JOIN என்பது இடப்பக்கத்து மதிப்புகளை வெளிப்படுத்திய பின்னர், வலப்பக்கத்து மதிப்புடன் ஒத்துப் போகவில்லையென்றால் *Null* மதிப்பினையும் அவ்வாறே, வலப்பக்கத்து மதிப்புகளை வெளிப்படுத்திய பின்னர், இடப்பக்கத்து மதிப்புடன் ஒத்துப் போகவில்லையென்றால் *Null* மதிப்பினையும் வெளிப்படுத்தும்.

Mysql-ல் *FULL OUTER JOIN* எனும் *keyword* கிடையாது. ஏனெனில் *LEFT OUTER* மற்றும் *RIGHT OUTER* இரண்டும் சேர்ந்து நடைபெறுவது தான் *FULL OUTER*. ஆகவே அவை இரண்டையும் *UNION* செய்வதன் மூலம் நாம் *full outer*-ஐ நிகழ்த்திவிடலாம். (*UNION* -ஐப் பற்றி நாம் *set operators*-ல் விரிவாகக் காண்போம்)

```
select * from ITEmployees e left outer join ITDepartment d on e.dept_id=d.dept_id union select * from ITEmployees e right outer join ITDepartment d on e.dept_id=d.dept_id;
```

```
mysql> select * from ITEmployees e left outer join ITDepartment d on e.dept_id=d.dep
t_id union select * from ITEmployees e right outer join ITDepartment d on e.dept_id=
d.dept_id;
+--------+------------+---------+---------+------------+---------------+
| Emp_ID | Emp_Name   | Dept_ID | Dept_ID | Dept_Name  | Location      |
+--------+------------+---------+---------+------------+---------------+
|      1 | Revathi    |    2345 |    2345 | Dev        | Tambaram      |
|      2 | Suresh     |    2348 |    2348 | Testing    | Pallikaranai  |
|      3 | Ponmalar   |    3467 |    3467 | Testing    | Pallikaranai  |
|      4 | Srividhya  |    3468 |    3468 | IT Support | Siruseri      |
|      5 | Sowmya     |    5215 |    5215 | Admin      | Mahindra City |
|      6 | Kathiravan |    2350 |    NULL | NULL       | NULL          |
|      7 | Oliviya    |    2351 |    NULL | NULL       | NULL          |
|      8 | Sundari    |    2352 |    NULL | NULL       | NULL          |
|      9 | Prakash    |    5216 |    5216 | Finance    | Perungudi     |
|     10 | Malar      |    2354 |    NULL | NULL       | NULL          |
|   NULL | NULL       |    NULL |    2346 | Dev        | Tambaram      |
|   NULL | NULL       |    NULL |    2347 | Dev        | Tambaram      |
|   NULL | NULL       |    NULL |    3465 | Testing    | Pallikaranai  |
|   NULL | NULL       |    NULL |    3466 | Testing    | Pallikaranai  |
+--------+------------+---------+---------+------------+---------------+
14 rows in set (0.00 sec)

mysql>
```

6.3 Self Join

Query-53

ஒரே ஒரு *table*-ன் இரண்டு *instance* இணைக்கப்பட்டு, அதிலிருந்து தகவல்களைப் பெறுவது *self join* எனப்படும்.

select s1.Emp_id,s1.emp_name from selfie s1 join selfie s2 where s1.Emp_ID=s2.Manager_ID;

```
nithya@nithya-laptop: ~
File Edit View Search Terminal Help
mysql> select * from selfie;
+--------+------------+------------+--------------+
| Emp_ID | Emp_Name   | Manager_ID | Manager_Name |
+--------+------------+------------+--------------+
|      1 | Revathi    |          5 | Sowmya       |
|      2 | Suresh     |          6 | Kathiravan   |
|      3 | Ponmalar   |          7 | Oliviya      |
|      4 | Srividhya  |          8 | Sundari      |
|      5 | Sowmya     |          9 | Prakash      |
|      6 | Kathiravan |         10 | Malar        |
|      7 | Oliviya    |       1111 | CEO          |
|      8 | Sundari    |       1111 | CEO          |
|      9 | Prakash    |       1111 | CEO          |
|     10 | Malar      |       1111 | CEO          |
+--------+------------+------------+--------------+
10 rows in set (0.00 sec)

mysql> select s1.Emp_id,s1.emp_name from selfie s1 join selfie s2 where s1.Emp_ID=s2.Manager_
ID;
+--------+------------+
| Emp_id | emp_name   |
+--------+------------+
|      5 | Sowmya     |
|      6 | Kathiravan |
|      7 | Oliviya    |
|      8 | Sundari    |
|      9 | Prakash    |
|     10 | Malar      |
+--------+------------+
6 rows in set (0.00 sec)

mysql>
```

6.4 Cartesian Join / Cross Join

Cartesian Join அல்லது *Cross Join* என்பது

மிகவும் அரிதாகப் பயன்படக் கூடிய ஒன்று. ஏனெனில் முதல் *table*-ல் 8 வரிகள், இரண்டாவது *table*-ல் 8 வரிகள் உள்ளதெனில், இவ்விரண்டையும் இணைத்து நாம் தகவல்களைப் பெறும் போது அது 64 வரிகளை வெளிப்படுத்தும். அதாவது முதல் *table*-ல் உள்ள 8 வரிகளும், இரண்டாவது *table*-ல் உள்ள முதல் வரியுடன் இணைந்து வெளிப்படும். பின்னர் மீண்டும் முதல் *table*-ல் உள்ள 8 வரிகளும், இரண்டாவது *table*-ல் உள்ள இரண்டாவது வரியுடன் இணைந்து வெளிப்படும். அவ்வாறே ஒவ்வொரு முறையும் நிகழ்ந்து மிகப் பெரிய எண்ணிக்கையில் தகவல்களை வெளிப்படுத்தும் .

Query-54

```
select * from ITEmployees,ITDepartment;
```

nithya@nithya-laptop: ~

File Edit View Search Terminal Help

```
mysql> select * from ITEmployees,ITDepartment;
+--------+------------+---------+---------+------------+--------------+
| Emp_ID | Emp_Name   | Dept_ID | Dept_ID | Dept_Name  | Location     |
+--------+------------+---------+---------+------------+--------------+
|      1 | Revathi    |    2345 |    2345 | Dev        | Tambaram     |
|      2 | Suresh     |    2348 |    2345 | Dev        | Tambaram     |
|      3 | Ponmalar   |    3467 |    2345 | Dev        | Tambaram     |
|      4 | Srividhya  |    3468 |    2345 | Dev        | Tambaram     |
|      5 | Sowmya     |    5215 |    2345 | Dev        | Tambaram     |
|      6 | Kathiravan |    2350 |    2345 | Dev        | Tambaram     |
|      7 | Oliviya    |    2351 |    2345 | Dev        | Tambaram     |
|      8 | Sundari    |    2352 |    2345 | Dev        | Tambaram     |
|      9 | Prakash    |    5216 |    2345 | Dev        | Tambaram     |
|     10 | Malar      |    2354 |    2345 | Dev        | Tambaram     |
|      1 | Revathi    |    2345 |    2346 | Dev        | Tambaram     |
|      2 | Suresh     |    2348 |    2346 | Dev        | Tambaram     |
|      3 | Ponmalar   |    3467 |    2346 | Dev        | Tambaram     |
|      4 | Srividhya  |    3468 |    2346 | Dev        | Tambaram     |
|      5 | Sowmya     |    5215 |    2346 | Dev        | Tambaram     |
|      6 | Kathiravan |    2350 |    2346 | Dev        | Tambaram     |
|      7 | Oliviya    |    2351 |    2346 | Dev        | Tambaram     |
|      8 | Sundari    |    2352 |    2346 | Dev        | Tambaram     |
|      9 | Prakash    |    5216 |    2346 | Dev        | Tambaram     |
|     10 | Malar      |    2354 |    2346 | Dev        | Tambaram     |
|      1 | Revathi    |    2345 |    2347 | Dev        | Tambaram     |
|      2 | Suresh     |    2348 |    2347 | Dev        | Tambaram     |
|      3 | Ponmalar   |    3467 |    2347 | Dev        | Tambaram     |
|      4 | Srividhya  |    3468 |    2347 | Dev        | Tambaram     |
|      5 | Sowmya     |    5215 |    2347 | Dev        | Tambaram     |
|      6 | Kathiravan |    2350 |    2347 | Dev        | Tambaram     |
|      7 | Oliviya    |    2351 |    2347 | Dev        | Tambaram     |
|      8 | Sundari    |    2352 |    2347 | Dev        | Tambaram     |
|      9 | Prakash    |    5216 |    2347 | Dev        | Tambaram     |
|     10 | Malar      |    2354 |    2347 | Dev        | Tambaram     |
|      1 | Revathi    |    2345 |    2348 | Testing    | Pallikaranai |
|      2 | Suresh     |    2348 |    2348 | Testing    | Pallikaranai |
|      3 | Ponmalar   |    3467 |    2348 | Testing    | Pallikaranai |
|      4 | Srividhya  |    3468 |    2348 | Testing    | Pallikaranai |
|      5 | Sowmya     |    5215 |    2348 | Testing    | Pallikaranai |
```

nithya@nithya-laptop: ~

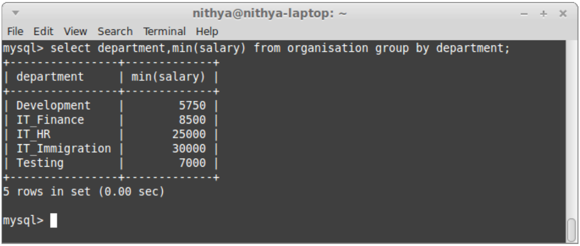
File Edit View Search Terminal Help

```
|        4 | Srividhya  |     3468 |     3467 | Testing    | Pallikaranai  |
|        5 | Sowmya     |     5215 |     3467 | Testing    | Pallikaranai  |
|        6 | Kathiravan |     2350 |     3467 | Testing    | Pallikaranai  |
|        7 | Oliviya    |     2351 |     3467 | Testing    | Pallikaranai  |
|        8 | Sundari    |     2352 |     3467 | Testing    | Pallikaranai  |
|        9 | Prakash    |     5216 |     3467 | Testing    | Pallikaranai  |
|       10 | Malar      |     2354 |     3467 | Testing    | Pallikaranai  |
|        1 | Revathi    |     2345 |     3468 | IT Support | Siruseri      |
|        2 | Suresh     |     2348 |     3468 | IT Support | Siruseri      |
|        3 | Ponmalar   |     3467 |     3468 | IT Support | Siruseri      |
|        4 | Srividhya  |     3468 |     3468 | IT Support | Siruseri      |
|        5 | Sowmya     |     5215 |     3468 | IT Support | Siruseri      |
|        6 | Kathiravan |     2350 |     3468 | IT Support | Siruseri      |
|        7 | Oliviya    |     2351 |     3468 | IT Support | Siruseri      |
|        8 | Sundari    |     2352 |     3468 | IT Support | Siruseri      |
|        9 | Prakash    |     5216 |     3468 | IT Support | Siruseri      |
|       10 | Malar      |     2354 |     3468 | IT Support | Siruseri      |
|        1 | Revathi    |     2345 |     5215 | Admin      | Mahindra City |
|        2 | Suresh     |     2348 |     5215 | Admin      | Mahindra City |
|        3 | Ponmalar   |     3467 |     5215 | Admin      | Mahindra City |
|        4 | Srividhya  |     3468 |     5215 | Admin      | Mahindra City |
|        5 | Sowmya     |     5215 |     5215 | Admin      | Mahindra City |
|        6 | Kathiravan |     2350 |     5215 | Admin      | Mahindra City |
|        7 | Oliviya    |     2351 |     5215 | Admin      | Mahindra City |
|        8 | Sundari    |     2352 |     5215 | Admin      | Mahindra City |
|        9 | Prakash    |     5216 |     5215 | Admin      | Mahindra City |
|       10 | Malar      |     2354 |     5215 | Admin      | Mahindra City |
|        1 | Revathi    |     2345 |     5216 | Finance    | Perungudi     |
|        2 | Suresh     |     2348 |     5216 | Finance    | Perungudi     |
|        3 | Ponmalar   |     3467 |     5216 | Finance    | Perungudi     |
|        4 | Srividhya  |     3468 |     5216 | Finance    | Perungudi     |
|        5 | Sowmya     |     5215 |     5216 | Finance    | Perungudi     |
|        6 | Kathiravan |     2350 |     5216 | Finance    | Perungudi     |
|        7 | Oliviya    |     2351 |     5216 | Finance    | Perungudi     |
|        8 | Sundari    |     2352 |     5216 | Finance    | Perungudi     |
|        9 | Prakash    |     5216 |     5216 | Finance    | Perungudi     |
|       10 | Malar      |     2354 |     5216 | Finance    | Perungudi     |
+----------+------------+----------+----------+------------+---------------+
100 rows in set (0.00 sec)
```

7

Subqueries

Sub query — ஐப் பற்றிக் கற்பதற்கு முன்னர் முதலில் அதன் அவசியத்தைத் தெரிந்து கொள்வோம். பின்வரும் உதாரணத்தில், ஒரு அலுவலகத்திலுள்ள ஒவ்வொரு துறைக்கும் குறைந்தபட்ச சம்பளம் எவ்வளவு தரப்படுகிறது என்பதைக் கண்டுபிடிக்க பின்வரும் *query*-யைப் பயன்படுத்தலாம்.



```
mysql> select department,min(salary) from organisation group by department;
+----------------+-------------+
| department     | min(salary) |
+----------------+-------------+
| Development    |        5750 |
| IT_Finance     |        8500 |
| IT_HR          |       25000 |
| IT_Immigration |       30000 |
| Testing        |        7000 |
+----------------+-------------+
5 rows in set (0.00 sec)

mysql>
```

பின்னர் *IT_Finance*-துறைக்கு அளிக்கப்படும் குறைந்தபட்ச சம்பளத்தைவிட அதிகமாக வாங்கும் துறைகளின் குறைந்த சம்பளத்தைக் கணக்கிட பின்வரும் *query*-யைப் பயன்படுத்தலாம்.

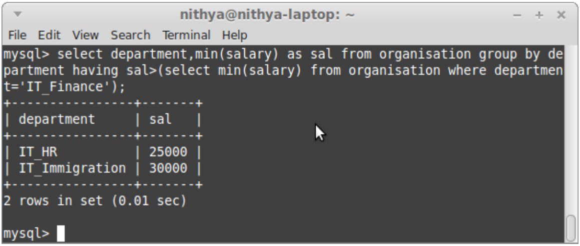
select department,min(salary) as sal from organisation group by department having sal>8500;

```
mysql> select department,min(salary) as sal from organisation group by de
partment having sal>8500;
+----------------+-------+
| department     | sal   |
+----------------+-------+
| IT_HR          | 25000 |
| IT_Immigration | 30000 |
+----------------+-------+
2 rows in set (0.00 sec)

mysql> 
```

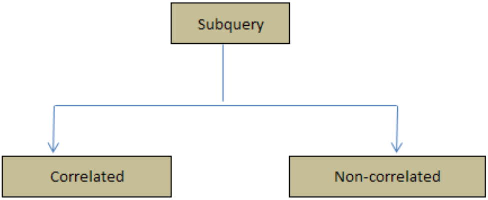
ஆனால் முதலில் *IT_Finance*-துறையின் குறைந்தபட்ச சம்பளத்தைத் தெரிந்து கொண்டு, பின்னர் அந்த மதிப்பினை *condition*-ல் கொடுப்பதற்கு பதிலாக, "*IT_Finance*-துறையின் குறைந்தபட்ச சம்பளத்தைத் தெரிந்து கொள்ளும் *query*-யையே" *condition*-ல் கொடுக்கலாம். இதுவே *sub-query* எனப்படும்.

Query-55

select department,min(salary) as sal from organisation group by department having sal>(select min(salary) from organisation where department='IT_Finance');



இதனை *correlated* மற்றும் *non-correlated* என்று இருவகையாகப் பிரிக்கலாம்.



7.1 *Non-correlated Subquery*

sub query-யில் உள்ள *table*-ம், வெளியில் இருக்கும் *main query*-யில் உள்ள *table*-ம் எந்த ஒரு *condition*-ஆலும் இணைக்கப்படாமல் இரண்டும் தனித்தனியாக அமைந்தால் அது *non-correlated subquery* எனப்படும்.

அதாவது வெறும் *subquery*-ஐ மட்டும் தனியாக எடுத்து *execute* செய்தால் கூட ஏதேனும் ஒரு *result* கிடைக்கும். மேற்கண்ட உதாரணத்தில் நாம் பயன்படுத்தியிருப்பது *non-correlated subquery* வகையைச் சேர்ந்தது.

7.2 *Correlated Subquery*

sub query-யில் உள்ள *table*-ம், வெளியில் இருக்கும் *main query*-யில் உள்ள *table*-ம் ஏதேனும் ஒரு *condition*-ஆல் இணைக்கப்பட்டிருந்தால் அது *correlated subquery* எனப்படும்.

அதாவது இத்தகைய *sub query*-ஆல் தனியாக இயங்க இயலாது. வெளியில் இருக்கும் *query*-உடன் சேர்த்து இயக்கினால் மட்டுமே அது *result*-ஐக் கொடுக்கும்.

Query-56

```
SELECT * FROM organisation org

WHERE exists (SELECT * FROM ITDepartment
itd WHERE org.department = itd.dept_name);
```

nithya@nithya-laptop: ~

File Edit View Search Terminal Help

```
mysql> select * from organisation;
+--------+------------+---------------+--------+--------------+
| Emp_ID | Emp_name   | Department    | salary | joining_date |
+--------+------------+---------------+--------+--------------+
|  98016 | Kothai     | IT_Finance    |  15000 | 0000-00-00   |
|  98017 | Ezhil      | IT_HR         |  25000 | 0000-00-00   |
|  98018 | Porkodi    | Development   |  11725 | 0000-00-00   |
|  98598 | Karthika   | Development   |  18550 | 0000-00-00   |
|  98019 | Nalini     | IT_Finance    |   8500 | 0000-00-00   |
|  98020 | Kalaiselvi | Development   |   9750 | 0000-00-00   |
|  98020 | Malathi    | Development   |   5750 | 0000-00-00   |
|  98021 | Jothi      | Testing       |  18700 | 0000-00-00   |
|  98021 | Renuka     | Testing       |  11725 | 0000-00-00   |
|  98021 | Nirmala    | Testing       |   8700 | 0000-00-00   |
|  98021 | Sangeetha  | Testing       |   7000 | 0000-00-00   |
|  98022 | Jayanthi   | IT_Immigration |  30000 | 0000-00-00   |
+--------+------------+---------------+--------+--------------+
12 rows in set (0.00 sec)

mysql> select * from ITDepartment;
+---------+------------+---------------+
| Dept_ID | Dept_Name  | Location      |
+---------+------------+---------------+
|    2345 | Dev        | Tambaram      |
|    2346 | Dev        | Tambaram      |
|    2347 | Dev        | Tambaram      |
|    2348 | Testing    | Pallikaranai  |
|    3465 | Testing    | Pallikaranai  |
|    3466 | Testing    | Pallikaranai  |
|    3467 | Testing    | Pallikaranai  |
|    3468 | IT Support | Siruseri      |
|    5215 | Admin      | Mahindra City |
|    5216 | Finance    | Perungudi     |
+---------+------------+---------------+
10 rows in set (0.01 sec)
```

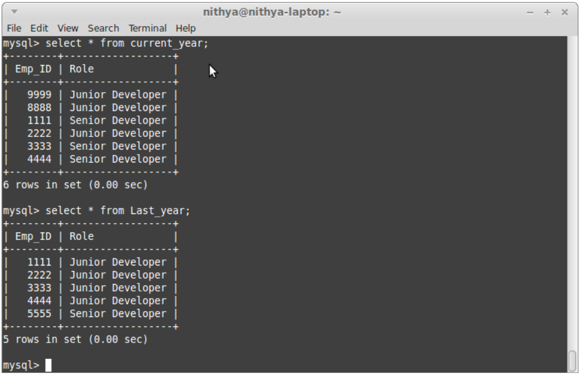
```
nithya@nithya-laptop: ~
File Edit View Search Terminal Help
mysql> SELECT * FROM organisation org WHERE exists (SELECT * FROM ITDepar
tment itd WHERE org.department = itd.dept_name);
+--------+-----------+------------+--------+--------------+
| Emp_ID | Emp_name  | Department | salary | joining_date |
+--------+-----------+------------+--------+--------------+
|  98021 | Jothi     | Testing    |  18700 | 0000-00-00   |
|  98021 | Renuka    | Testing    |  11725 | 0000-00-00   |
|  98021 | Nirmala   | Testing    |   8700 | 0000-00-00   |
|  98021 | Sangeetha | Testing    |   7000 | 0000-00-00   |
+--------+-----------+------------+--------+--------------+
4 rows in set (0.00 sec)

mysql>
```

Set Operators

Union, Unionall, Intersect, Minus ஆகிய நான்கும் *set operators* ஆகும். இரண்டு *table*-களில் இத்தகைய *set operators*-ஐப் பயன்படுத்தும் போது எப்படித் தகவல்கள் வெளிவருகின்றன என்பதைப் பின்வரும் படத்தின் மூலம் சுலபமாகப் புரிந்து கொள்ளலாம்.

8.1 Union & Union All

UNION என்பது இரண்டு வெவ்வேறு *table*-களில் இருக்கும் தகவல்களை ஒன்றாக இணைத்து வெளிப்படுத்துகிறது.

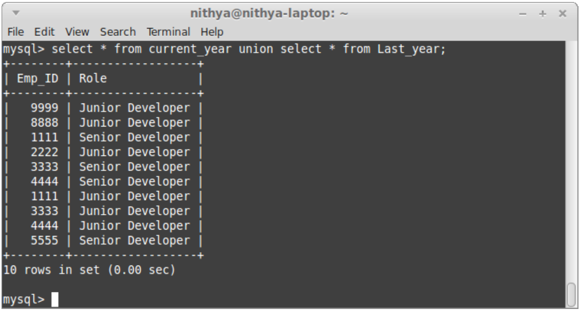


```
mysql> select * from current_year;
+--------+------------------+
| Emp_ID | Role             |
+--------+------------------+
|   9999 | Junior Developer |
|   8888 | Junior Developer |
|   1111 | Senior Developer |
|   2222 | Junior Developer |
|   3333 | Senior Developer |
|   4444 | Senior Developer |
+--------+------------------+
6 rows in set (0.00 sec)

mysql> select * from Last_year;
+--------+------------------+
| Emp_ID | Role             |
+--------+------------------+
|   1111 | Junior Developer |
|   2222 | Junior Developer |
|   3333 | Junior Developer |
|   4444 | Junior Developer |
|   5555 | Senior Developer |
+--------+------------------+
5 rows in set (0.00 sec)

mysql>
```

Query-57

உதாரணத்துக்கு ஒரு நிறுவனத்தில் உள்ளவர்கள் சென்ற வருடத்திலிருந்து இந்த வருடத்துக்கு பதவி உயர்வு பெற்றுள்ளார்களா இல்லையா என்பதை, *current_year, last_year* எனும் இரண்டு வெவ்வேறு *table*-களிலிருந்து அவர்களுடைய பதவியினை எடுத்து ஒப்பிட்டுப் பார்த்து தெரிந்து கொள்ளலாம்.

*select * from current_year union select * from Last_year;*

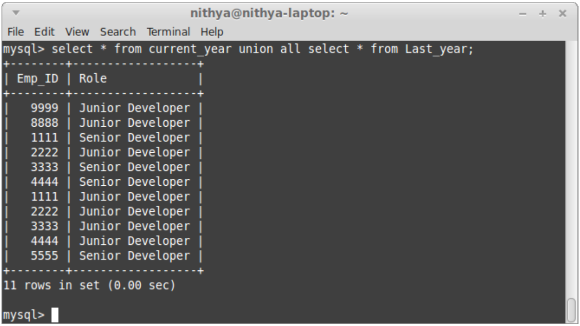


இதில் *Emp_id* ***9999,5555,2222,8888*** போன்றவற்றிற்கு ஒரே ஒரு முறை மட்டும் *entry* காணப்படுகிறது. பொதுவாக *Union* என்பது *duplicate* தகவல்களைத் தவிர்த்துவிடும். எனவே இவர்களெல்லாம் பதவி உயர்வு பெறாமல், இரண்டு *table*-களிலும் ஒரே பதவியுடன் இருப்பவர்கள் என்றே நினைக்கிறேன். எனினும் இதனை உறுதி செய்து கொள்ள *UNION ALL*-ஐப்

பயன்படுத்தலாம். இது *duplicates*-ஐயும் சேர்த்து வெளிப்படுத்தும்.

Query-58

*select * from current_year union all select * from Last_year;*



```
nithya@nithya-laptop: ~
File Edit View Search Terminal Help
mysql> select * from current_year union all select * from Last_year;
+--------+------------------+
| Emp_ID | Role             |
+--------+------------------+
|   9999 | Junior Developer |
|   8888 | Junior Developer |
|   1111 | Senior Developer |
|   2222 | Junior Developer |
|   3333 | Senior Developer |
|   4444 | Senior Developer |
|   1111 | Junior Developer |
|   2222 | Junior Developer |
|   3333 | Junior Developer |
|   4444 | Junior Developer |
|   5555 | Senior Developer |
+--------+------------------+
11 rows in set (0.00 sec)

mysql>
```

இப்பொழுது *Emp_id* 2222-க்கு மட்டும் *duplicate entry* காணப்படுகிறது. எனவே இவர் மட்டுமே பதவி உயர்வு பெறாதவர். மற்றவர்களெல்லாம், இந்த வருடத்தில் வேலைக்குச் சேர்ந்தவர்களாக இருக்க வேண்டும் அல்லது சென்ற வருடத்தில் வேலையை விட்டுச் சென்றவர்களாக இருக்க வேண்டும்.

8.2 Intersect

INTERSECT என்பது இரண்டு வெவ்வேறு *table*-களில் இருக்கும் பொதுவான தகவல்களை எடுத்து வெளிப்படுத்துகிறது. அதாவது ஒருவருடைய பதவி *current_year, last_year* எனும் இரண்டு *table*-களிலும் ஒரே மாதிரியாக இருப்பின் அதனை எடுத்து வெளிப்படுத்துகிறது. எனவே இதன் மூலம் பதவி உயர்வு பெறாதவர்களின் விவரங்களை எளிமையாகக் கண்டுபிடித்து விடலாம்.

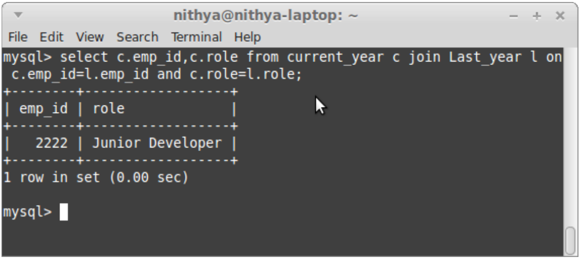
Mysql-ல் *INTERSECT* எனும் *keyword* கிடையாது. ஏனெனில் *intersect*-ன் செயல்பாட்டினை நாம்

INNER JOIN கொண்டே நிகழ்த்தி விட முடியும்.

Query-59

select c.emp_id,c.role from current_year c join Last_year l

on c.emp_id=l.emp_id and c.role=l.role;



8.3 *Minus*

MINUS என்பது முதல் *table*-ல், இருக்கும் விவரங்கள் இரண்டாவது *table*-ல் காணப்பட்டால் அதனைக் கழித்துவிட்டு, இரண்டாவது *table*-ல் இல்லாத விவரங்களை மட்டுமே வெளிப்படுத்தும். எனவே இந்த வருடத்தில் புதிதாக வேலைக்குச் சேர்ந்தவர்களைத் தெரிந்து கொள்ள *current_year MINUS last_year* எனக் கொடுத்தால் போதுமானது.

Mysql-ல் *MINUS* எனும் *keyword* கிடையாது. ஏனெனில் *minus*-ன் செயல்பாட்டினை நாம் *LEFT OUTER JOIN*-உடன் ஒரு சிறிய *condition*-ஐ இணைப்பதன் மூலம் நிகழ்த்தி விட முடியும்.

Query-60

select c.emp_id, c.role from current_year c left join Last_year l on c.emp_id=l.emp_id and c.role=l.role where l.emp_id is null;

nithya@nithya-laptop: ~

File Edit View Search Terminal Help

```
mysql> select c.emp_id,c.role from current_year c left join Last_year l on c.emp_id
=l.emp_id and c.role=l.role where l.emp_id is null;
+--------+------------------+
| emp_id | role             |
+--------+------------------+
|   9999 | Junior Developer |
|   8888 | Junior Developer |
|   1111 | Senior Developer |
|   3333 | Senior Developer |
|   4444 | Senior Developer |
+--------+------------------+
5 rows in set (0.00 sec)

mysql> 
```

9

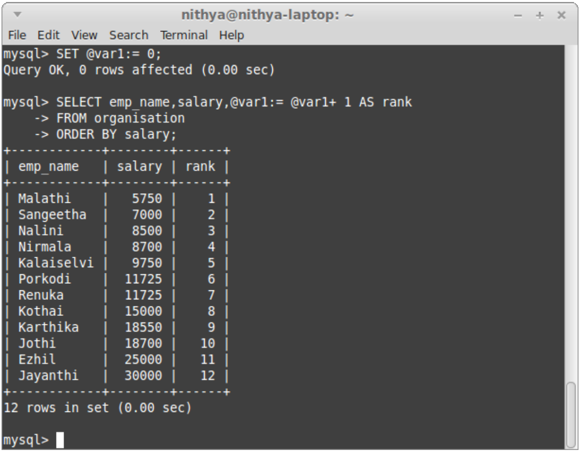
Ranks

ஏதேனும் ஒரு *column*-ல் உள்ள மதிப்புகளை ஏறுவரிசையிலோ, இறங்குவரிசையிலோ முறைப்படுத்திவிட்டு பின்னர் அதற்கு **1,2,3....** என மதிப்புகளைக் கொடுப்பதே *ranking* எனப்படும்.

mysql-ல் *ranking* என்பது *variables*-ஐ வைத்தே நடைபெறுகிறது. **@** எனும் குறியீடு இது ஒரு *variable* என்பதை உணர்த்துகிறது. *SET* எனும் *command* முதன்முதலில், *variable*-க்கு ஒரு மதிப்பினை வழங்கப் பயன்படுகிறது.

Query-61

```
SET @var1:= 0;

SELECT emp_name,salary,@var1:= @var1+ 1
AS rank

FROM organisation

ORDER BY salary;
```



```
nithya@nithya-laptop: ~
File  Edit  View  Search  Terminal  Help
mysql> SET @var1:= 0;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT emp_name,salary,@var1:= @var1+ 1 AS rank
    -> FROM organisation
    -> ORDER BY salary;
+------------+--------+------+
| emp_name   | salary | rank |
+------------+--------+------+
| Malathi    |   5750 |    1 |
| Sangeetha  |   7000 |    2 |
| Nalini     |   8500 |    3 |
| Nirmala    |   8700 |    4 |
| Kalaiselvi |   9750 |    5 |
| Porkodi    |  11725 |    6 |
| Renuka     |  11725 |    7 |
| Kothai     |  15000 |    8 |
| Karthika   |  18550 |    9 |
| Jothi      |  18700 |   10 |
| Ezhil      |  25000 |   11 |
| Jayanthi   |  30000 |   12 |
+------------+--------+------+
12 rows in set (0.00 sec)

mysql>
```

முதலில் ஒரு *variable*-க்கு *0* என்று மதிப்பினை வழங்கி விட வேண்டும். பின்னர் *select statement*-ல் வெளிப்படும் ஒவ்வொரு *record*-க்கும் *variable*-வுடன் ஒவ்வொரு எண்ணாகக் கூட்டி வெளிப்படுத்துமாறு செய்ய வேண்டும்.

எனவே முதல் *record*-க்கு *var(0)+1 = 1* என

வெளிப்படுத்தும். அடுத்த *record*-க்கு *var*-ன் மதிப்பு 1 என மாறியிருந்தால் தான், அதனால் 1+1 = 2 என வெளிப்படுத்த முடியும். . எனவே தான் *select statement*-ல் *var = var* + 1 எனக் கொடுக்காமல், *var := var* + 1 எனக் கொடுத்துள்ளோம். := எனும் குறியீடு *variable*-வுடன் 1-ஐக் கூட்டி வந்த மதிப்பினை மீண்டும் *variable*-க்கு வழங்கப் பயன்படுகிறது.

மேற்கண்ட உதாரணத்தில் ஒரே மதிப்பினைப் பெற்ற அடுத்தடுத்த வரிகளுக்கு, அடுத்தடுத்த *rank* வழங்கப்பட்டிருப்பதை கவனிக்கவும். முறைப்படி பார்த்தால் ஒரே மதிப்புகளுக்கு ஒரே *rank* மட்டுமே வழங்க வேண்டும். எனவே இதுபோன்ற தவறுகளைத் தவிர்க்க அடுத்தடுத்த மதிப்புகளை ஒப்பிடும் வகையில் ஒரு *condition*-ஐ இணைத்தால் போதுமானது. எனவே *query*-யைப் பின்வருமாறு மாற்றி அமைக்கலாம்.

Query-62

```
SET @var2 = NULL;

SET @var1= 0;

SELECT emp_name,salary,CASE

WHEN @var2 = salary THEN @var1

WHEN @var2 := salary THEN @var1:=
@var1+ 1

END AS rank

FROM organisation

ORDER BY salary;
```

```
nithya@nithya-laptop: ~
File  Edit  View  Search  Terminal  Help
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT emp_name,salary,CASE
    -> WHEN @var2 = salary THEN @var1
    -> WHEN @var2 := salary THEN @var1:= @var1+ 1
    -> END AS rank
    -> FROM organisation
    -> ORDER BY salary;
+------------+--------+------+
| emp_name   | salary | rank |
+------------+--------+------+
| Malathi    |   5750 |    1 |
| Sangeetha  |   7000 |    2 |
| Nalini     |   8500 |    3 |
| Nirmala    |   8700 |    4 |
| Kalaiselvi |   9750 |    5 |
| Porkodi    |  11725 |    6 |
| Renuka     |  11725 |    6 |
| Kothai     |  15000 |    7 |
| Karthika   |  18550 |    8 |
| Jothi      |  18700 |    9 |
| Ezhil      |  25000 |   10 |
| Jayanthi   |  30000 |   11 |
+------------+--------+------+
12 rows in set (0.00 sec)

mysql> 
```

இங்கு *var1, var2* என்று இரண்டு *variables* பயன்படுத்தப்பட்டுள்ளது. *var2*- ஆனது *order by* மூலம் நாம் வரிசைப்படுத்தும் *column* மதிப்புகளைப் பெற்று விளங்குமாறும், *var1*-ஆனது *rank* வழங்கவும் பயன்பட்டுள்ளது.

முதலில் *var2*-க்கு *NULL* மதிப்பு வழங்கப்பட்டுள்ளது. பின்னர் *case statement* மூலம் *var2*-ன் மதிப்பு *column*-ல் உள்ள முதல் மதிப்புக்கு சமமாக உள்ளதா என்பதை பரிசோதிக்கிறது. சமமாக இல்லை. எனவே := குறியீடு மூலம் *column*-ல் உள்ள முதல் மதிப்பினை *var2* பெற்றுவிடுகிறது, அதன் தொடர்ச்சியாக *var1(0)*+*1* = *1* என வெளிப்படுத்துகிறது. இப்போது *var2*-ன் மதிப்பு *column*-ல் உள்ள முதல் மதிப்புக்கு சமமாக உள்ளது. இப்போது மீண்டும் *case statement* மூலம் *var2*, *column*-ல் உள்ள அடுத்த மதிப்புடன் சமமாக உள்ளதா என்பதை சோதிக்கிறது. அவை சமமாக இருப்பின் அதே *rank*-ஐ வழங்குமாறும், அவ்வாறு இல்லையெனில், அதற்கு அடுத்த *rank*-ஐ வழங்குமாறும் கட்டளைகளை அமைக்கிறது.

10

Stored Procedures

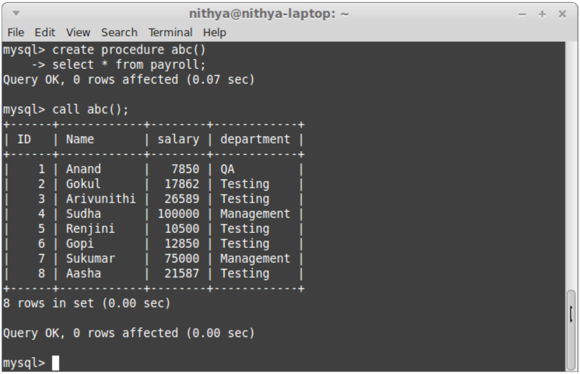
Stored Procedures என்பது ஒன்று அல்லது அதற்கு மேற்பட்ட *query*-களை உள்ளடக்கிய ஒரு தொகுப்பு ஆகும். இவற்றைத் தனித்தனி *query*-களாக *execute* செய்வதைக் காட்டிலும், இதுபோன்று ஒன்றாகத் தொகுத்து *execute* செய்வதன் மூலம் *database*-ன் செயல்திறன் அதிகரிக்கிறது. இதுபோன்ற தொகுப்புகள்(*Procedures*) *database*-ன் *server*-ல் சேமிக்கப்படுவதால் இவை சேமிக்கப்பட்ட தொகுப்புகள்(*Stored Procedures*) என்று அழைக்கப்படுகின்றன.

முதலில் எளிமையான *query*-யை உள்ளடக்கிய *stored procedure*-ஐ எவ்வாறு உருவாக்குவது என்று பார்ப்போம்.

create procedure abc()

*select * from payroll;*

call abc();



```
nithya@nithya-laptop: ~
File Edit View Search Terminal Help
mysql> create procedure abc()
    -> select * from payroll;
Query OK, 0 rows affected (0.07 sec)

mysql> call abc();
+------+------------+--------+------------+
| ID   | Name       | salary | department |
+------+------------+--------+------------+
|    1 | Anand      |   7850 | QA         |
|    2 | Gokul      |  17862 | Testing    |
|    3 | Arivunithi |  26589 | Testing    |
|    4 | Sudha      | 100000 | Management |
|    5 | Renjini    |  10500 | Testing    |
|    6 | Gopi       |  12850 | Testing    |
|    7 | Sukumar    |  75000 | Management |
|    8 | Aasha      |  21587 | Testing    |
+------+------------+--------+------------+
8 rows in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql>
```

இங்கு *abc* எனும் *procedure* உருவாக்கப்பட்டுவிட்டது. இதன் பின்னர் *abc*-ஐ அழைக்கும்போதெல்லாம் அதற்குள் உள்ள *query*, *execute*-செய்யப்பட்டு அதன் *output* மட்டுமே வெளிவரும்.

Query -64

ஒன்றுக்கும் மேற்பட்ட வரிகளை *procedure*-ல் எழுதும்போது அவற்றை *begin, end* எனும் *keyword*-க்குள் எழுத வேண்டும். மேலும் ; க்கு பதிலாக // ஐ *delimiter*-ஆகப் பயன்படுத்த வேண்டும். அப்போதுதான் நாம் உருவாக்கும் *stored procedure, server*-க்குள் சென்று சேமிக்கப்படும்.

எனவே *DELIMITER* // என்பது *default* ஆன ; க்கு பதிலாக // ஐ *delimiter*-ஆக மாற்றி அமைக்கும்.

```
delimiter //

create procedure xyz()

BEGIN
```

```
select * from employees limit 2;

select * from organisation limit 2;

END//

delimiter ;

call xyz();
```

```
nithya@nithya-laptop: ~
File Edit View Search Terminal Help
mysql> delimiter //
mysql> create procedure xyz()
    -> BEGIN
    -> select * from employees limit 2;
    -> select * from organisation limit 2;
    -> END//
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> delimiter ;
mysql>
mysql> call xyz();
+---------+--------------------+--------+----------------+--------------+
| name    | role               | salary | commission_pct | joining_date |
+---------+--------------------+--------+----------------+--------------+
| Sudha   | Testing Engineer   |  12000 |             25 | 2007-11-19   |
| Revathi | ITSupport Engineer |  19500 |              3 | 2012-10-31   |
+---------+--------------------+--------+----------------+--------------+
2 rows in set (0.00 sec)

+--------+----------+------------+--------+--------------+
| Emp_ID | Emp_name | Department | salary | joining_date |
+--------+----------+------------+--------+--------------+
|  98016 | Kothai   | IT_Finance |  15000 | 2013-10-12   |
|  98017 | Ezhil    | IT_HR      |  25000 | 2013-10-12   |
+--------+----------+------------+--------+--------------+
2 rows in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql>
mysql>
mysql>
```

ஒரு *stored procedure* உருவாக்கப்பட்ட பின்னர் மீண்டும் *delimiter*-ஐ ; க்கு மாற்றி அமைக்க *DELIMITER ;* என்று கொடுக்க வேண்டும். இது பின்வருமாறு.

delimiter ;

call xyz();

```
nithya@nithya-laptop: ~
File  Edit  View  Search  Terminal  Help

mysql> delimiter ;
mysql> call xyz();
+---------+--------------------+--------+----------------+--------------+
| name    | role               | salary | commission_pct | joining_date |
+---------+--------------------+--------+----------------+--------------+
| Sudha   | Testing Engineer   |  12000 |             25 | 2007-11-19   |
| Revathi | ITSupport Engineer |  19500 |              3 | 2012-10-31   |
+---------+--------------------+--------+----------------+--------------+
2 rows in set (0.00 sec)

+--------+----------+------------+--------+--------------+
| Emp_ID | Emp_name | Department | salary | joining_date |
+--------+----------+------------+--------+--------------+
|  98016 | Kothai   | IT_Finance |  15000 | 2013-10-12   |
|  98017 | Ezhil    | IT_HR      |  25000 | 2013-10-12   |
+--------+----------+------------+--------+--------------+
2 rows in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> 
```

Passing Parameters

Query — 65

ஏதேனும் ஒரு மதிப்பினை *variable* மூலமாக *procedure*-க்குள் செலுத்த *IN keyword* பயன்படுகிறது. இங்கு *x* என்பது *variable* ஆகும்.

```
delimiter //

create procedure m(IN x int(10))

BEGIN

select * from employees where salary<x;

END

//

delimiter ;

call m(10000);
```

```
nithya@nithya-laptop: ~
File Edit View Search Terminal Help
mysql> delimiter //
mysql> create procedure m(IN x int(10))
    -> BEGIN
    -> select * from employees where salary<x;
    -> END
    -> //
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;
mysql> call m(10000);
+---------+----------------+--------+----------------+--------------+
| name    | role           | salary | commission_pct | joining_date |
+---------+----------------+--------+----------------+--------------+
| Sherlin | Asst. Engineer |   4500 |             50 | 2011-02-15   |
| Malathi | Java Developer |   7500 |           NULL | 2009-12-10   |
+---------+----------------+--------+----------------+--------------+
2 rows in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> 
```

Query — **66:**

ஏதேனும் ஒரு மதிப்பினை procedure நமக்கு வெளிப்படுத்துமாறு செய்ய OUT keyword பயன்படுகிறது.

delimiter //

create procedure count_low_paid_people(IN salary_amount int(10), OUT total int)

BEGIN

```
select count(*) into total from employees where salary<salary_amount;

END

//

delimiter ;

call count_low_paid_people(10000,@total);

select @total;

select * from employees where salary<10000;
```

```
nithya@nithya-laptop: ~
File  Edit  View  Search  Terminal  Help
mysql> delimiter //
mysql> create procedure count_low_paid_people(IN salary_amount int(10), OUT to
tal int)
    -> BEGIN
    -> select count(*) into total from employees where salary<salary_amount;
    -> END
    -> //
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> delimiter ;
mysql>
mysql> call count_low_paid_people(10000,@total);
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> select @total;
+--------+
| @total |
+--------+
|      2 |
+--------+
1 row in set (0.00 sec)

mysql> select * from employees where salary<10000;
+---------+----------------+--------+----------------+--------------+
| name    | role           | salary | commission_pct | joining_date |
+---------+----------------+--------+----------------+--------------+
| Sherlin | Asst. Engineer |   4500 |             50 | 2011-02-15   |
| Malathi | Java Developer |   7500 |           NULL | 2009-12-10   |
+---------+----------------+--------+----------------+--------------+
2 rows in set (0.00 sec)

mysql> 
```

Query — 67:

ஏதேனும் ஒரு மதிப்பினை procedure-க்குள் செலுத்தி, அதன்மீது சில கணக்கீடுகள் செய்து

மீண்டும் அந்த மதிப்பினை *procedure* வெளிப்படுத்துமாறு செய்ய *INOUT keyword* பயன்படுகிறது.

```
DELIMITER //

CREATE PROCEDURE set_counter(INOUT count INT(4),IN inc INT(4))

BEGIN

SET count = count + inc;

END

//

DELIMITER ;

SET @counter = 1;

CALL set_counter(@counter,1); — 2

CALL set_counter(@counter,1); — 3
```

CALL set_counter(@counter,5); — 8

SELECT @counter; — 8

```
nithya@nithya-laptop: ~
File  Edit  View  Search  Terminal  Help
mysql> DELIMITER //
mysql> CREATE PROCEDURE set_counter(INOUT count INT(4),IN inc INT(4))
    -> BEGIN
    -> SET count = count + inc;
    -> END
    -> //
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> DELIMITER ;
mysql>
mysql> SET @counter = 1;
Query OK, 0 rows affected (0.00 sec)

mysql> CALL set_counter(@counter,1);
Query OK, 0 rows affected (0.00 sec)

mysql> CALL set_counter(@counter,1);
Query OK, 0 rows affected (0.00 sec)

mysql> CALL set_counter(@counter,5);
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @counter;
+----------+
| @counter |
+----------+
|        8 |
+----------+
1 row in set (0.00 sec)

mysql>
```

Query — 68:

Case statement-ஐ *query*-ல் எவ்வாறு பயன்படுத்துவது என்று *query* 47-ல் பார்த்தோம். இப்போது அதனையே எவ்வாறு *stored procedure*-ல் அமைப்பது என்று பின்வருமாறு பார்க்கலாம்.

```
DELIMITER //

CREATE PROCEDURE GetCustomerLevel(

in p_customerNumber int(11),

out p_customerLevel varchar(10))

BEGIN

DECLARE creditlim double;

SELECT creditlimit INTO creditlim

FROM customers

WHERE customerNumber = p_customerNumber;
```

```
CASE

WHEN creditlim > 50000 THEN

SET p_customerLevel = 'PLATINUM';

WHEN (creditlim <= 50000 AND creditlim >=
10000) THEN

SET p_customerLevel = 'GOLD';

WHEN creditlim < 10000 THEN

SET p_customerLevel = 'SILVER';

END CASE;

END

//
```

Credit limit:

\> 50 *K* = *Platinum*

<50K & > 10 K = Gold

<10K = Silver

CALL GetCustomerLevel(112,@level);

SELECT @level AS 'Customer Level';

Result:

PLATINUM

11

Triggers

Trigger என்பது *Table* அளவில் சில வேலைகளைத் தானியக்கமாக செய்யப் பயன்படுத்தப்படுகிறது. அதாவது *database*-ல் *table* ல் தகவல்கள் செலுத்தப்படும்போதோ, தகவல்கள் மாற்றப்படும்போதோ அல்லது நீக்கப் படும் போதோ

நமக்கு வேண்டியவாறு வேறு சில வேலைகளையும் சேர்த்து செய்ய வைக்கலாம். இதற்கு *Trigger* பயன்படுகிறது.

இதுவும் *Stored Procedure* போலத்தான். ஆனால் *Trigger* ஆனது குறிப்பிட்ட நிகழ்வின்போது தானாக

அழைக்கப்படுகிறது. ஆனால் *Stored Procedure* ஐ தேவைப்படும் போது, நாம்தான் அழைக்கவேண்டும்.

தகவலை சேமிப்பதற்கு முன்னே சரிபார்க்கவும், *Table* ல் நடக்கும் மாற்றங்களை ஆராயவும் *Trigger* ஐ பயன்படுத்தலாம்.

Syntax:

CREATE

[DEFINER = { user | CURRENT_USER }]

TRIGGER trigger_name

trigger_time trigger_event

ON tbl_name FOR EACH ROW

trigger_body

trigger_time: { BEFORE | AFTER }

trigger_event: { INSERT | UPDATE | DELETE

}

உதாரணம்.

ITEmployees என்ற *Table* ல் நடக்கும் *UPDATE* நிகழ்ச்சிகளை வேறு ஒரு *Table* ல் சேர்த்து கண்காணிப்பது எப்படி என்று பார்ப்போம்.

முதலில் *employees_audit* என்ற ஒரு *Table* ஐ உருவாக்குவோம்.

Query — 69:

```
CREATE TABLE employees_audit (
id int(11) NOT NULL AUTO_INCREMENT,
employeeNumber int(11) NOT NULL,
name varchar(50) NOT NULL,
changedon datetime DEFAULT NULL,
action varchar(50) DEFAULT NULL,
```

PRIMARY KEY (id)

);

பின்வரும் *BEFORE UPDATE* க்கான *Trigger* ஐ உருவாக்கவும்.

Query — *70:*

Formula does not parse

DELIMITER

DELIMITER ;

```
nithya@nithya-laptop: ~
File  Edit  View  Search  Terminal  Help
mysql>
mysql> desc ITEmployees;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| Emp_ID   | int(10)      | YES  |     | NULL    |       |
| Emp_Name | varchar(255) | YES  |     | NULL    |       |
| Dept_ID  | int(10)      | YES  |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql> DELIMITER $$
mysql> CREATE TRIGGER before_employee_update
    ->     BEFORE UPDATE ON ITEmployees
    ->     FOR EACH ROW BEGIN
    ->
    ->     INSERT INTO employees_audit
    ->     SET action = 'update',
    ->      employeeNumber = OLD.Emp_ID,
    ->         name = OLD.Emp_Name,
    ->         changedon = NOW();
    -> END$$
Query OK, 0 rows affected (0.10 sec)
```

இப்போது *ITEmployees* என்ற *Table* ல் நடக்கும் *UPDATE* செயலுகள் யாவும் *employees_audit* என்ற *Table* ல் சேமிக்கப் படுவதைக் காணலாம்.

```
mysql> select * from ITEmployees;
+--------+------------+---------+
| Emp_ID | Emp_Name   | Dept_ID |
+--------+------------+---------+
|      1 | Revathi    |    2345 |
|      2 | Suresh     |    2348 |
|      3 | Ponmalar   |    3467 |
|      4 | Srividhya  |    3468 |
|      5 | Sowmya     |    5215 |
|      6 | Kathiravan |    2350 |
|      7 | Oliviya    |    2351 |
|      8 | Sundari    |    2352 |
|      9 | Prakash    |    5216 |
|     10 | Malar      |    2354 |
+--------+------------+---------+
10 rows in set (0.08 sec)

mysql> select * from employees_audit;
Empty set (0.00 sec)

mysql> update ITEmployees set Emp_Name='Nithya' where Emp_ID=10;
Query OK, 1 row affected (0.16 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from employees_audit;
+----+----------------+-------+---------------------+--------+
| id | employeeNumber | name  | changedon           | action |
+----+----------------+-------+---------------------+--------+
|  1 |             10 | Malar | 2015-05-03 04:37:54 | update |
+----+----------------+-------+---------------------+--------+
1 row in set (0.00 sec)
```

இவ்வாறு எந்த ஒரு *Table* இலும், *INSERT,UPDATE,DELETE* நிகழ்வுகளின் முன்னும் பின்னும் நமக்குத் தேவையான வகையில் *Trigger* எழுத முடியும்.

Trigger பற்றி மேலும் அறிய —

http://www.mysqltutorial.org/mysql-triggers.aspx

12

முடிவுரை

இந்த நூலில் MySQL ன் சில முக்கியக் கூறுகளை மட்டுமே பார்த்துள்ளோம்.

இன்னும் இந்த நூலில் எழுதப் படாதவை பல. அவற்றை வாசகர்கள் இணையத்தில் தேடி, அறிந்து கொள்ள இந்த நூல் ஆர்வத்தைத் தூண்டும் என நம்புகிறேன்.

பின்வரும் இணைப்புகள் மிகவும் பயனுள்ளதாக இருக்கும்.

http://www.mysqltutorial.org/

http://www.tutorialspoint.com/mysql/

https://dev.mysql.com/doc/refman/5.7/en/tutorial.html