

# MySQL

1.	SELECT CLAUSE.....	2
2.	WHERE CLAUSE.....	2
3.	LOGICAL OPERATORS.....	2
4.	IN OPERATOR.....	3
5.	BETWEEN OPERATOR.....	4
6.	LIKE OPERATOR.....	4
7.	REGEXP OPERATOR.....	5
8.	IS NULL OPERATOR.....	5
9.	ORDER BY CLAUSE.....	5
10.	LIMIT CLAUSE .....	6
11.	INNER JOIN.....	6
12.	SELF JOIN.....	9
13.	OUTER JOIN – LEFT JOIN / RIGHT JOIN .....	11
14.	SELF OUTER JOIN.....	14
15.	USING CLAUSE .....	14
16.	CROSS JOINS.....	15
17.	UNIONS .....	15
18.	INSERTING, UPDATING AND DELETING DATA.....	17
19.	CREATE A COPY OF A TABLE.....	18
20.	SUMMARIZING DATA.....	21
21.	WRITING COMPLEX QUERY.....	23
22.	ESSENTIAL MYSQL FUNCTIONS .....	29
23.	STORED PROCEDURES .....	34
24.	TRIGGERS AND EVENTS.....	38

## 1. SELECT Clause

```
SELECT * FROM customers;
```

customer_id	first_name	last_name	birth_date	phone	address	city	state	points
1	Babara	MacCaffrey	1986-03-28	781-932-9754	0 Sage Terrace	Waltham	MA	2273
2	Ines	Brushfield	1986-04-13	804-427-9456	14187 Commercial Trail	Hampton	VA	947
3	Freddi	Boagey	1985-02-07	719-724-7869	251 Springs Junction	Colorado Springs	CO	2967
4	Ambur	Roseburgh	1974-04-14	407-231-8017	30 Arapahoe Terrace	Orlando	FL	457
5	Clemmie	Betchley	1973-11-07	NULL	5 Spohn Circle	Arlington	TX	3675
6	Elka	Twiddell	1991-09-04	312-480-8498	7 Manley Drive	Chicago	IL	3073
7	Ilene	Dowson	1964-08-30	615-641-4759	50 Lillian Crossing	Nashville	TN	1672
8	Thacher	Naseby	1993-07-17	941-527-3977	538 Mosinee Center	Sarasota	FL	205
9	Romola	Rumgay	1992-05-23	559-181-3744	3520 Ohio Trail	Visalia	CA	1486
10	Levy	Mynett	1969-10-13	404-246-3370	68 Lawn Avenue	Atlanta	GA	796

## 2. WHERE Clause

```
SELECT *  
FROM customers  
WHERE points > 3000;
```

customer_id	first_name	last_name	birth_date	phone	address	city	state	points
5	Clemmie	Betchley	1973-11-07	NULL	5 Spohn Circle	Arlington	TX	3675
6	Elka	Twiddell	1991-09-04	312-480-8498	7 Manley Drive	Chicago	IL	3073

2 rows in set (0.00 sec)

## 3. Logical Operators

-- AND

```
SELECT *  
FROM customers  
WHERE birth_date > '1990-01-01' AND points > 1000;
```

-- OR

```
SELECT *  
FROM customers  
WHERE birth_date > '1990-01-01' OR points > 1000;
```

customer_id	first_name	last_name	birth_date	phone	address	city	state	points
1	Babara	MacCaffrey	1986-03-28	781-932-9754	0 Sage Terrace	Waltham	MA	2273
3	Freddi	Boagey	1985-02-07	719-724-7869	251 Springs Junction	Colorado Springs	CO	2967
5	Clemmie	Betchley	1973-11-07	NULL	5 Spohn Circle	Arlington	TX	3675
6	Elka	Twiddell	1991-09-04	312-480-8498	7 Manley Drive	Chicago	IL	3073
7	Ilene	Dowson	1964-08-30	615-641-4759	50 Lillian Crossing	Nashville	TN	1672
8	Thacher	Naseby	1993-07-17	941-527-3977	538 Mosinee Center	Sarasota	FL	205
9	Romola	Rumgay	1992-05-23	559-181-3744	3520 Ohio Trail	Visalia	CA	1486

7 rows in set (0.00 sec)

-- NOR / NOT

```
SELECT *
FROM customers
WHERE NOT (birth_date > '1990-01-01' OR points > 1000);
```

customer_id	first_name	last_name	birth_date	phone	address	city	state	points
2	Ines	Brushfield	1986-04-13	804-427-9456	14187 Commercial Trail	Hampton	VA	947
4	Ambur	Roseburgh	1974-04-14	407-231-8017	30 Arapahoe Terrace	Orlando	FL	457
10	Levy	Mynett	1969-10-13	404-246-3370	68 Lawn Avenue	Atlanta	GA	796

3 rows in set (0.00 sec)

-- AND + OR

```
SELECT *
FROM customers
WHERE birth_date > '1990-01-01' OR points > 1000 AND state = 'VA';
```

customer_id	first_name	last_name	birth_date	phone	address	city	state	points
6	Elka	Twiddell	1991-09-04	312-480-8498	7 Manley Drive	Chicago	IL	3073
8	Thacher	Naseby	1993-07-17	941-527-3977	538 Mosinee Center	Sarasota	FL	205
9	Romola	Rumgay	1992-05-23	559-181-3744	3520 Ohio Trail	Visalia	CA	1486

3 rows in set (0.00 sec)

## 4. IN Operator

-- Instead of writing like this:

```
SELECT *
FROM customers
WHERE state = 'VA' OR state = 'GA' OR state = 'FL';
```

-- Using **IN** we could write like this:

```
SELECT *
FROM customers
WHERE state IN ('VA', 'FL', 'GA');
```

customer_id	first_name	last_name	birth_date	phone	address	city	state	points
2	Ines	Brushfield	1986-04-13	804-427-9456	14187 Commercial Trail	Hampton	VA	947
4	Ambur	Roseburgh	1974-04-14	407-231-8017	30 Arapahoe Terrace	Orlando	FL	457
8	Thacher	Naseby	1993-07-17	941-527-3977	538 Mosinee Center	Sarasota	FL	205
10	Levy	Mynett	1969-10-13	404-246-3370	68 Lawn Avenue	Atlanta	GA	796

4 rows in set (0.00 sec)

## 5. BETWEEN Operator

-- Instead of writing like this:

```
SELECT *
FROM customers
WHERE points >= 1000 AND points <= 3000;
```

-- Using BETWEEN we could write like this:

```
SELECT *
FROM customers
WHERE points BETWEEN 1000 AND 3000;
```

customer_id	first_name	last_name	birth_date	phone	address	city	state	points
1	Babara	MacCaffrey	1986-03-28	781-932-9754	0 Sage Terrace	Waltham	MA	2273
3	Freddi	Boagey	1985-02-07	719-724-7869	251 Springs Junction	Colorado Springs	CO	2967
7	Ilene	Dowson	1964-08-30	615-641-4759	50 Lillian Crossing	Nashville	TN	1672
9	Romola	Rumgay	1992-05-23	559-181-3744	3520 Ohio Trail	Visalia	CA	1486

4 rows in set (0.00 sec)

## 6. LIKE Operator

-- Return customers whose last names contains the letter 'b', whether it's in the beginning, middle or end:

```
SELECT *
FROM customers
WHERE last_name LIKE '%b%';
```

customer_id	first_name	last_name	birth_date	phone	address	city	state	points
2	Ines	Brushfield	1986-04-13	804-427-9456	14187 Commercial Trail	Hampton	VA	947
3	Freddi	Boagey	1985-02-07	719-724-7869	251 Springs Junction	Colorado Springs	CO	2967
4	Ambur	Roseburgh	1974-04-14	407-231-8017	30 Arapahoe Terrace	Orlando	FL	457
5	Clemmie	Betchley	1973-11-07	NULL	5 Spohn Circle	Arlington	TX	3675
8	Thacher	Naseby	1993-07-17	941-527-3977	538 Mosinee Center	Sarasota	FL	205

5 rows in set (0.00 sec)

## 7. REGEXP Operator

-- Return customers whose last names contains 'field' in any position.

```
SELECT *  
FROM customers  
WHERE last_name REGEXP 'field';
```

customer_id	first_name	last_name	birth_date	phone	address	city	state	points
2	Ines	Brushfield	1986-04-13	804-427-9456	14187 Commercial Trail	Hampton	VA	947

1 row in set (0.00 sec)

## 8. IS NULL Operator

```
SELECT *  
FROM customers  
WHERE phone IS NULL;
```

customer_id	first_name	last_name	birth_date	phone	address	city	state	points
5	Clemmie	Betchley	1973-11-07	NULL	5 Spohn Circle	Arlington	TX	3675

1 row in set (0.00 sec)

## 9. ORDER BY Clause

-- ASC ORDER (by default)

```
SELECT *  
FROM customers  
ORDER BY first_name;
```

customer_id	first_name	last_name	birth_date	phone	address	city	state	points
4	Ambur	Roseburgh	1974-04-14	407-231-8017	30 Arapahoe Terrace	Orlando	FL	457
1	Babara	MacCaffrey	1986-03-28	781-932-9754	0 Sage Terrace	Waltham	MA	2273
5	Clemmie	Betchley	1973-11-07	NULL	5 Spohn Circle	Arlington	TX	3675
6	Elka	Twiddell	1991-09-04	312-480-8498	7 Manley Drive	Chicago	IL	3073
3	Freddi	Boagey	1985-02-07	719-724-7869	251 Springs Junction	Colorado Springs	CO	2967
7	Ilene	Dowson	1964-08-30	615-641-4759	50 Lillian Crossing	Nashville	TN	1672
2	Ines	Brushfield	1986-04-13	804-427-9456	14187 Commercial Trail	Hampton	VA	947
10	Levy	Mynett	1969-10-13	404-246-3370	68 Lawn Avenue	Atlanta	GA	796
9	Romola	Rumgay	1992-05-23	559-181-3744	3520 Ohio Trail	Visalia	CA	1486
8	Thacher	Naseby	1993-07-17	941-527-3977	538 Mosinee Center	Sarasota	FL	205

10 rows in set (0.00 sec)

-- DESC ORDER

```
SELECT *
FROM customers
ORDER BY first_name DESC;
```

customer_id	first_name	last_name	birth_date	phone	address	city	state	points
8	Thacher	Naseby	1993-07-17	941-527-3977	538 Mosinee Center	Sarasota	FL	205
9	Romola	Rumgay	1992-05-23	559-181-3744	3520 Ohio Trail	Visalia	CA	1486
10	Levy	Mynett	1969-10-13	404-246-3370	68 Lawn Avenue	Atlanta	GA	796
2	Ines	Brushfield	1986-04-13	804-427-9456	14187 Commercial Trail	Hampton	VA	947
7	Ilene	Dowson	1964-08-30	615-641-4759	50 Lillian Crossing	Nashville	TN	1672
3	Freddi	Boagey	1985-02-07	719-724-7869	251 Springs Junction	Colorado Springs	CO	2967
6	Elka	Twiddell	1991-09-04	312-480-8498	7 Manley Drive	Chicago	IL	3073
5	Clemmie	Betchley	1973-11-07	NULL	5 Spohn Circle	Arlington	TX	3675
1	Babara	MacCaffrey	1986-03-28	781-932-9754	0 Sage Terrace	Waltham	MA	2273
4	Ambur	Roseburgh	1974-04-14	407-231-8017	30 Arapahoe Terrace	Orlando	FL	457

10 rows in set (0.00 sec)

## 10. LIMIT Clause

-- Return only the first 3 customers;

```
SELECT *
FROM customers
LIMIT 3;
```

customer_id	first_name	last_name	birth_date	phone	address	city	state	points
1	Babara	MacCaffrey	1986-03-28	781-932-9754	0 Sage Terrace	Waltham	MA	2273
2	Ines	Brushfield	1986-04-13	804-427-9456	14187 Commercial Trail	Hampton	VA	947
3	Freddi	Boagey	1985-02-07	719-724-7869	251 Springs Junction	Colorado Springs	CO	2967

3 rows in set (0.00 sec)

-- Skip the first 6 customers and return the next 3 customers

```
SELECT *
FROM customers
LIMIT 6, 3;
```

customer_id	first_name	last_name	birth_date	phone	address	city	state	points
7	Ilene	Dowson	1964-08-30	615-641-4759	50 Lillian Crossing	Nashville	TN	1672
8	Thacher	Naseby	1993-07-17	941-527-3977	538 Mosinee Center	Sarasota	FL	205
9	Romola	Rumgay	1992-05-23	559-181-3744	3520 Ohio Trail	Visalia	CA	1486

3 rows in set (0.00 sec)

## 11. Inner JOIN

-- Orders Table

order_id	customer_id	order_date	status	comments	shipped_date	shipper_id
1	6	2019-01-30	1	NULL	NULL	NULL
2	7	2018-08-02	2	NULL	2018-08-03	4
3	8	2017-12-01	1	NULL	NULL	NULL
4	2	2017-01-22	1	NULL	NULL	NULL
5	5	2017-08-25	2		2017-08-26	3
6	10	2018-11-18	1	Aliquam erat volutpat. In congue.	NULL	NULL
7	2	2018-09-22	2	NULL	2018-09-23	4
8	5	2018-06-08	1	Mauris enim leo, rhoncus sed, vestibulum sit amet, cursus id, turpis.	NULL	NULL
9	10	2017-07-05	2	Nulla mollis molestie lorem. Quisque ut erat.	2017-07-06	1
10	6	2018-04-22	2	NULL	2018-04-23	2

10 rows in set (0.00 sec)

## -- Customers Table

customer_id	first_name	last_name	birth_date	phone	address	city	state	points
1	Babara	MacCaffrey	1986-03-28	781-932-9754	0 Sage Terrace	Waltham	MA	2273
2	Ines	Brushfield	1986-04-13	804-427-9456	14187 Commercial Trail	Hampton	VA	947
3	Freddi	Boagey	1985-02-07	719-724-7869	251 Springs Junction	Colorado Springs	CO	2967
4	Ambur	Roseburgh	1974-04-14	407-231-8017	30 Arapahoe Terrace	Orlando	FL	457
5	Clemmie	Betchley	1973-11-07	NULL	5 Spohn Circle	Arlington	TX	3675
6	Elka	Twiddell	1991-09-04	312-480-8498	7 Manley Drive	Chicago	IL	3073
7	Ilene	Dowson	1964-08-30	615-641-4759	50 Lillian Crossing	Nashville	TN	1672
8	Thacher	Naseby	1993-07-17	941-527-3977	538 Mosinee Center	Sarasota	FL	205
9	Romola	Rumgay	1992-05-23	559-181-3744	3520 Ohio Trail	Visalia	CA	1486
10	Levy	Mynett	1969-10-13	404-246-3370	68 Lawn Avenue	Atlanta	GA	796

-- a) Select `order\_id` column from "Orders" table, `first\_name` and `last\_name` from "Customers" table and join them together where the id is equal

```
SELECT order_id, first_name, last_name, status
FROM orders
JOIN customers
ON orders.customer_id = customers.customer_id;
```

order_id	first_name	last_name	status
1	Elka	Twiddell	1
2	Ilene	Dowson	2
3	Thacher	Naseby	1
4	Ines	Brushfield	1
5	Clemmie	Betchley	2
6	Levy	Mynett	1
7	Ines	Brushfield	2
8	Clemmie	Betchley	1
9	Levy	Mynett	2
10	Elka	Twiddell	2

10 rows in set (0.00 sec)

-- Using ALIAS = AS

-- b) Select `order\_id` and `customers\_id` column from "Orders" table and `first\_name`, and `last\_name` from "Customers" table and join them together

```
SELECT order_id, o.customer_id, first_name, last_name, status
FROM orders AS o
JOIN customers c
ON o.customer_id = c.customer_id;
```

order_id	customer_id	first_name	last_name	status
1	6	Elka	Twiddell	1
2	7	Ilene	Dowson	2
3	8	Thacher	Naseby	1
4	2	Ines	Brushfield	1
5	5	Clemmie	Betchley	2
6	10	Levy	Mynett	1
7	2	Ines	Brushfield	2
8	5	Clemmie	Betchley	1
9	10	Levy	Mynett	2
10	6	Elka	Twiddell	2

10 rows in set (0.00 sec)

-- c) two different databases: **sql\_store** and **sql\_hinventory**

```
USE sql_store;
SELECT * FROM order_items;
```

order_id	product_id	quantity	unit_price
1	4	4	3.74
2	1	2	9.10
2	4	4	1.66
2	6	2	2.94
3	3	10	9.12
4	3	7	6.99
4	10	7	6.40
5	2	3	9.89
6	1	4	8.65
6	2	4	3.28
6	3	4	7.46
6	5	1	3.45
7	3	7	9.17
8	5	2	6.94
8	8	2	8.59
9	6	5	7.28
10	1	10	6.01
10	9	9	4.28

18 rows in set (0.00 sec)

```
USE sql_inventory;
SELECT * FROM products;
```

product_id	name	quantity_in_stock	unit_price
1	Foam Dinner Plate	70	1.21
2	Pork - Bacon,back Peameal	49	4.65
3	Lettuce - Romaine, Heart	38	3.35
4	Brocolinni - Gaylan, Chinese	90	4.53
5	Sauce - Ranch Dressing	94	1.63
6	Petit Baguette	14	2.39
7	Sweet Pea Sprouts	98	3.29
8	Island Oasis - Raspberry	26	0.74
9	Longan	67	2.26
10	Broom - Push	6	1.09

10 rows in set (0.00 sec)

-- JOIN 2 tables (order\_item and products) from 2 different DataBases: "sql\_store" and "sql\_inventory"

```
USE sql_store;
```



```

SELECT *
FROM order_items AS o
JOIN sql_inventory.products AS p
  ON o.product_id = p.product_id;

```

order_id	product_id	quantity	unit_price	product_id	name	quantity_in_stock	unit_price
2	1	2	9.10	1	Foam Dinner Plate	70	1.21
6	1	4	8.65	1	Foam Dinner Plate	70	1.21
10	1	10	6.01	1	Foam Dinner Plate	70	1.21
5	2	3	9.89	2	Pork - Bacon,back Peameal	49	4.65
6	2	4	3.28	2	Pork - Bacon,back Peameal	49	4.65
3	3	10	9.12	3	Lettuce - Romaine, Heart	38	3.35
4	3	7	6.99	3	Lettuce - Romaine, Heart	38	3.35
6	3	4	7.46	3	Lettuce - Romaine, Heart	38	3.35
7	3	7	9.17	3	Lettuce - Romaine, Heart	38	3.35
1	4	4	3.74	4	Brocolinni - Gaylan, Chinese	90	4.53
2	4	4	1.66	4	Brocolinni - Gaylan, Chinese	90	4.53
6	5	1	3.45	5	Sauce - Ranch Dressing	94	1.63
8	5	2	6.94	5	Sauce - Ranch Dressing	94	1.63
2	6	2	2.94	6	Petit Baguette	14	2.39
9	6	5	7.28	6	Petit Baguette	14	2.39
8	8	2	8.59	8	Island Oasis - Raspberry	26	0.74
10	9	9	4.28	9	Longan	67	2.26
4	10	7	6.40	10	Broom - Push	6	1.09

18 rows in set (0.00 sec)

## 12. Self JOIN

-- employees Table

```

USE sql_hr;
SELECT *
FROM employees;

```

employee_id	first_name	last_name	job_title	salary	reports_to	office_id
33391	Darcy	Nortunen	Account Executive	62871	37270	1
37270	Yovonnda	Magrannell	Executive Secretary	63996	NULL	10
37851	Sayer	Matterson	Statistician III	98926	37270	1
40448	Mindy	Crissil	Staff Scientist	94860	37270	1
56274	Kerian	Alloisi	VP Marketing	110150	37270	1
63196	Alaster	Scutchin	Assistant Professor	32179	37270	2
67009	North	de Clerc	VP Product Management	114257	37270	2
67370	Elladine	Rising	Social Worker	96767	37270	2
68249	Nisse	Voysey	Financial Advisor	52832	37270	2
72540	Guthrey	Iacopetti	Office Assistant I	117690	37270	3
72913	Kass	Hefferan	Computer Systems Analyst IV	96401	37270	3
75900	Virge	Goodrum	Information Systems Manager	54578	37270	3
76196	Mirilla	Janowski	Cost Accountant	119241	37270	3
80529	Lynde	Aronson	Junior Executive	77182	37270	4
80679	Mildrid	Sokale	Geologist II	67987	37270	4
84791	Hazel	Tarbert	General Manager	93760	37270	4
95213	Cole	Kesterton	Pharmacist	86119	37270	4
96513	Theresa	Binney	Food Chemist	47354	37270	5
98374	Estrellita	Daleman	Staff Accountant IV	70187	37270	5
115357	Ivy	Fearey	Structural Engineer	92710	37270	5

20 rows in set (0.00 sec)

-- a) Joining the Employees table with itself. Select only the name of the employee and their managers

```
SELECT e.employee_id, e.first_name, m.first_name AS manager
FROM employees AS e -- employees
JOIN employees AS m -- managers
ON e.reports_to = m.employee_id;
```

employee_id	first_name	manager
33391	Darcy	Yovonnda
37851	Sayer	Yovonnda
40448	Mindy	Yovonnda
56274	Keriann	Yovonnda
63196	Alaster	Yovonnda
67009	North	Yovonnda
67370	Elladine	Yovonnda
68249	Nisse	Yovonnda
72540	Guthrey	Yovonnda
72913	Kass	Yovonnda
75900	Virge	Yovonnda
76196	Mirilla	Yovonnda
80529	Lynde	Yovonnda
80679	Mildrid	Yovonnda
84791	Hazel	Yovonnda
95213	Cole	Yovonnda
96513	Theresa	Yovonnda
98374	Estrellita	Yovonnda
115357	Ivy	Yovonnda

19 rows in set (0.00 sec)

-- b) Join 3 tables – “customers with orders”, “orders with order\_statuses”

Table 1 – customers

customer_id	first_name	last_name	birth_date	phone	address	city	state	points
1	Babara	MacCaffrey	1986-03-28	781-932-9754	0 Sage Terrace	Waltham	MA	2273
2	Ines	Brushfield	1986-04-13	804-427-9456	14187 Commercial Trail	Hampton	VA	947
3	Freddi	Boagey	1985-02-07	719-724-7869	251 Springs Junction	Colorado Springs	CO	2967
4	Ambur	Roseburgh	1974-04-14	407-231-8017	30 Arapahoe Terrace	Orlando	FL	457
5	Clemmie	Betchley	1973-11-07	NULL	5 Spohn Circle	Arlington	TX	3675
6	Elka	Twiddell	1991-09-04	312-480-8498	7 Manley Drive	Chicago	IL	3073
7	Ilene	Dowson	1964-08-30	615-641-4759	50 Lillian Crossing	Nashville	TN	1672
8	Thacher	Naseby	1993-07-17	941-527-3977	538 Mosinee Center	Sarasota	FL	205
9	Romola	Rumgay	1992-05-23	559-181-3744	3520 Ohio Trail	Visalia	CA	1486
10	Levy	Mynett	1969-10-13	404-246-3370	68 Lawn Avenue	Atlanta	GA	796

Table 2 – orders

order_id	customer_id	order_date	status	comments	shipped_date	shipper_id
1	6	2019-01-30	1	NULL	NULL	NULL
2	7	2018-08-02	2	NULL	2018-08-03	4
3	8	2017-12-01	1	NULL	NULL	NULL
4	2	2017-01-22	1	NULL	NULL	NULL
5	5	2017-08-25	2	NULL	2017-08-26	3
6	10	2018-11-18	1	Aliquam erat volutpat. In congue.	NULL	NULL
7	2	2018-09-22	2	NULL	2018-09-23	4
8	5	2018-06-08	1	Mauris enim leo, rhoncus sed, vestibulum sit amet, cursus id, turpis.	NULL	NULL
9	10	2017-07-05	2	Nulla mollis molestie lorem. Quisque ut erat.	2017-07-06	1
10	6	2018-04-22	2	NULL	2018-04-23	2

Table 3 – order\_statuses

order_status_id	name
1	Processed
2	Shipped
3	Delivered

```

USE sql_store;
SELECT *
FROM orders AS o
JOIN customers AS c ON o.customer_id = c.customer_id
JOIN order_statuses AS os ON o.status = os.order_status_id;

```

### -- c) Compound Join condition - One condition

```

USE sql_store;
SELECT *
FROM order_items AS oi
JOIN order_item_notes AS oin
    ON oi.order_id = oin.order_id;

```

order_id	product_id	quantity	unit_price	note_id	order_id	product_id	note
1	4	4	3.74	1	1	2	first note
1	4	4	3.74	2	1	2	second note

2 rows in set (0.00 sec)

### -- b) Compound Join condition - Multiple conditions

```

SELECT *
FROM order_items AS oi
JOIN order_item_notes AS oin
    ON oi.order_id = oin.order_id    -- this is a Compound Join Condition
    AND oi.product_id = oin.product_id;

```

Empty set (0.00 sec)

## 13. Outer JOIN – LEFT JOIN / RIGHT JOIN

### -- Customers Table

customer_id	first_name	last_name	birth_date	phone	address	city	state	points
1	Babara	MacCaffrey	1986-03-28	781-932-9754	0 Sage Terrace	Waltham	MA	2273
2	Ines	Brushfield	1986-04-13	804-427-9456	14187 Commercial Trail	Hampton	VA	947
3	Freddi	Boagey	1985-02-07	719-724-7869	251 Springs Junction	Colorado Springs	CO	2967
4	Ambur	Roseburgh	1974-04-14	407-231-8017	30 Arapahoe Terrace	Orlando	FL	457
5	Clemmie	Betchley	1973-11-07	NULL	5 Spohn Circle	Arlington	TX	3675
6	Elka	Twiddell	1991-09-04	312-480-8498	7 Manley Drive	Chicago	IL	3073
7	Ilene	Dowson	1964-08-30	615-641-4759	50 Lillian Crossing	Nashville	TN	1672
8	Thacher	Naseby	1993-07-17	941-527-3977	538 Mosinee Center	Sarasota	FL	205
9	Romola	Rumgay	1992-05-23	559-181-3744	3520 Ohio Trail	Visalia	CA	1486
10	Levy	Mynett	1969-10-13	404-246-3370	68 Lawn Avenue	Atlanta	GA	796

## -- Orders Table

order_id	customer_id	order_date	status	comments	shipped_date	shipper_id
1	6	2019-01-30	1	NULL	NULL	NULL
2	7	2018-08-02	2	NULL	2018-08-03	4
3	8	2017-12-01	1	NULL	NULL	NULL
4	2	2017-01-22	1	NULL	NULL	NULL
5	5	2017-08-25	2		2017-08-26	3
6	10	2018-11-18	1	Aliquam erat volutpat. In congue.	NULL	NULL
7	2	2018-09-22	2	NULL	2018-09-23	4
8	5	2018-06-08	1	Mauris enim leo, rhoncus sed, vestibulum sit amet, cursus id, turpis.	NULL	NULL
9	10	2017-07-05	2	Nulla mollis molestie lorem. Quisque ut erat.	2017-07-06	1
10	6	2018-04-22	2	NULL	2018-04-23	2

10 rows in set (0.00 sec)

## -- Shippers Table

shipper_id	name
1	Hettinger LLC
2	Schinner-Predovic
3	Satterfield LLC
4	Mraz, Renner and Nolan
5	Waters, Mayert and Prohaska

5 rows in set (0.00 sec)

- LEFT JOIN

-- a) LEFT JOIN returns all the elements from the left table "customers" whether the condition is true or not.

-- LEFT JOIN returns all the record on the mentioned columns below from the "column" table, even though the condition doesn't match or the order order\_id is NULL

```
SELECT c.customer_id, c.first_name, o.order_id
FROM customers AS c
LEFT JOIN orders AS o ON o.customer_id = c.customer_id
ORDER BY c.customer_id;
```

customer_id	first_name	order_id
1	Babara	NULL
2	Ines	4
2	Ines	7
3	Freddi	NULL
4	Ambur	NULL
5	Clemmie	5
5	Clemmie	8
6	Elka	1
6	Elka	10
7	Ilene	2
8	Thacher	3
9	Romola	NULL
10	Levy	6
10	Levy	9

14 rows in set (0.00 sec)

- OUTER JOIN - RIGHT JOIN

-- b) RIGHT JOIN returns all the elements from the right table "customers" whether the condition is true or not.

-- RIGHT JOIN returns all the record on the mentioned columns below, even though the condition doesn't match or the order order\_id is NULL

```
SELECT c.customer_id, c.first_name, o.order_id
FROM orders AS o
RIGHT JOIN customers AS c ON o.customer_id = c.customer_id
ORDER BY c.customer_id;
```

customer_id	first_name	order_id
1	Babara	NULL
2	Ines	4
2	Ines	7
3	Freddi	NULL
4	Ambur	NULL
5	Clemmie	5
5	Clemmie	8
6	Elka	1
6	Elka	10
7	Ilene	2
8	Thacher	3
9	Romola	NULL
10	Levy	6
10	Levy	9

14 rows in set (0.00 sec)

-- Avoiding use RIGHT JOIN and use LEFT JOIN instead

-- c) Join 3 tables: Orders, Customers, Shipper using LEFT JOIN

```
USE sql_store;
SELECT c.customer_id, c.first_name, o.order_id, s.name AS shipper
FROM customers AS c
LEFT JOIN orders AS o ON o.customer_id = c.customer_id
LEFT JOIN shippers AS s ON o.shipper_id = s.shipper_id
```

```
ORDER BY customer_id;
```

customer_id	first_name	order_id	shipper
1	Babara	NULL	NULL
2	Ines	4	NULL
2	Ines	7	Mraz, Renner and Nolan
3	Freddi	NULL	NULL
4	Ambur	NULL	NULL
5	Clemmie	5	Satterfield LLC
5	Clemmie	8	NULL
6	Elka	1	NULL
6	Elka	10	Schinner-Predovic
7	Ilene	2	Mraz, Renner and Nolan
8	Thacher	3	NULL
9	Romola	NULL	NULL
10	Levy	6	NULL
10	Levy	9	Hettinger LLC

14 rows in set (0.00 sec)

## 14. Self OUTER JOIN

-- Get all the employees and their managers, whether they have a manager or not

```
USE sql_hr;
SELECT e.employee_id, e.first_name, m.first_name AS manager
FROM employees AS e
LEFT JOIN employees AS m ON e.reports_to = m.employee_id;
```

employee_id	first_name	manager
33391	Darcy	Yovonnda
37270	Yovonnda	NULL
37851	Sayer	Yovonnda
40448	Mindy	Yovonnda
56274	Keriann	Yovonnda
63196	Alaster	Yovonnda
67009	North	Yovonnda
67370	Elladine	Yovonnda
68249	Nisse	Yovonnda
72540	Guthrey	Yovonnda
72913	Kass	Yovonnda
75900	Virge	Yovonnda
76196	Mirilla	Yovonnda
80529	Lynde	Yovonnda
80679	Mildrid	Yovonnda
84791	Hazel	Yovonnda
95213	Cole	Yovonnda
96513	Theresa	Yovonnda
98374	Estrellita	Yovonnda
115357	Ivy	Yovonnda

20 rows in set (0.00 sec)

## 15. USING Clause

-- Given the following conditions: ON o.customer\_id = c.customer\_id. If the column name is the same in both tables, we can simplify it by means of USING clause

-- Without **USING** Clause

```
USE sql_store;
SELECT o.order_id, c.first_name
FROM orders AS o
JOIN customers AS c ON o.customer_id = c.customer_id;
```

-- With **USING** Clause

```
USE sql_store;
SELECT o.order_id, c.first_name
FROM orders AS o
JOIN customers AS c USING (customer_id)
```

order_id	first_name
4	Ines
7	Ines
5	Clemmie
8	Clemmie
1	Elka
10	Elka
2	Ilene
3	Thacher
6	Levy
9	Levy

10 rows in set (0.00 sec)

## 16. Cross Joins

-- **CROSS JOIN** is used to combine or join every record from the first table with every record in the second table

```
SELECT c.first_name AS customer, p.name AS product
FROM customers AS c
CROSS JOIN products AS p
ORDER BY first_name;
```

## 17. Unions

```
-- a) Active
USE sql_store;
SELECT order_id, order_date, 'Active' AS status
FROM orders
WHERE order_date >= '2019-01-01';
```

order_id	order_date	status
1	2019-01-30	Active

1 row in set (0.00 sec)

```
-- b) Archived
USE sql_store;
SELECT order_id, order_date, 'Archived' AS status
FROM orders
WHERE order_date < '2019-01-01';
```

order_id	order_date	status
2	2018-08-02	Archived
3	2017-12-01	Archived
4	2017-01-22	Archived
5	2017-08-25	Archived
6	2018-11-18	Archived
7	2018-09-22	Archived
8	2018-06-08	Archived
9	2017-07-05	Archived
10	2018-04-22	Archived

9 rows in set (0.00 sec)

-- c) using UNION and combine rows from these two tables

```
USE sql_store;
SELECT order_id, order_date, 'Active' AS status
FROM orders
WHERE order_date >= '2019-01-01'
UNION
SELECT order_id, order_date, 'Archived' AS status
FROM orders
WHERE order_date < '2019-01-01';
```

order_id	order_date	status
1	2019-01-30	Active
2	2018-08-02	Archived
3	2017-12-01	Archived
4	2017-01-22	Archived
5	2017-08-25	Archived
6	2018-11-18	Archived
7	2018-09-22	Archived
8	2018-06-08	Archived
9	2017-07-05	Archived
10	2018-04-22	Archived

10 rows in set (0.00 sec)

-- c) Getting rows from Different Tables

```
USE sql_store;
SELECT first_name
FROM customers
UNION
SELECT name
FROM shippers;
```

first_name
Babara
Ines
Freddi
Ambur
Clemmie
Elka
Ilene
Thacher
Romola
Levy
Hettinger LLC
Schinner-Predovic
Satterfield LLC
Mraz, Renner and Nolan
Waters, Mayert and Prohaska

15 rows in set (0.01 sec)



## 18. Inserting, Updating and Deleting Data

-- Inserting a single row

-- 1. Using NULL and DEFAULT

```
USE sql_store;
INSERT INTO customers
VALUES (DEFAULT, 'John', 'Smith', '1990-01-01', NULL, 'address', 'city', 'CA', DEFAULT)
```

-- Better way - without DEFAULT or NULL

```
USE sql_store;
INSERT INTO customers (first_name, last_name, birth_date, address, city, state)
VALUES ('John', 'Smith', '1990-01-01', 'address', 'city', 'CA')
```

-- Returning the ID that MySQL generates when insert a new row

```
INSERT INTO orders (customer_id, order_date, status)
VALUES (1, '2019-01-02', 1);
SELECT LAST_INSERT_ID(); -- generated ID
```

-- Inserting 3 rows

```
USE sql_store;
INSERT INTO products (name, quantity_in_stock, unit_price)
VALUES
    ('Product1', 10, 1.95),
    ('Product2', 11, 1.95),
    ('Product3', 10, 1.95)
```

-- Inserting 3 rows hierarchical: using LAST\_INSERT\_ID()

```
INSERT INTO order_items
VALUES
    (LAST_INSERT_ID(), 1, 1, 2.95),
    (LAST_INSERT_ID(), 2, 1, 3.95);
```

## 19. Create a copy of a Table

-- I. The original Table

```
SELECT * FROM invoices;
```

invoice_id	number	client_id	invoice_total	payment_total	invoice_date	due_date	payment_date
1	91-953-3396	2	101.79	0.00	2019-03-09	2019-03-29	NULL
2	03-898-6735	5	175.32	8.18	2019-06-11	2019-07-01	2019-02-12
3	20-228-0335	5	147.99	0.00	2019-07-31	2019-08-20	NULL
4	56-934-0748	3	152.21	0.00	2019-03-08	2019-03-28	NULL
5	87-052-3121	5	169.36	0.00	2019-07-18	2019-08-07	NULL
6	75-587-6626	1	157.78	74.55	2019-01-29	2019-02-18	2019-01-03
7	68-093-9863	3	133.87	0.00	2019-09-04	2019-09-24	NULL
8	78-145-1093	1	189.12	0.00	2019-05-20	2019-06-09	NULL
9	77-593-0081	5	172.17	0.00	2019-07-09	2019-07-29	NULL
10	48-266-1517	1	159.50	0.00	2019-06-30	2019-07-20	NULL
11	20-848-0181	3	126.15	0.03	2019-01-07	2019-01-27	2019-01-11
13	41-666-1035	5	135.01	87.44	2019-06-25	2019-07-15	2019-01-26
15	55-105-9605	3	167.29	80.31	2019-11-25	2019-12-15	2019-01-15
16	10-451-8824	1	162.02	0.00	2019-03-30	2019-04-19	NULL
17	33-615-4694	3	126.38	68.10	2019-07-30	2019-08-19	2019-01-15
18	52-269-9803	5	180.17	42.77	2019-05-23	2019-06-12	2019-01-08
19	83-559-4105	1	134.47	0.00	2019-11-23	2019-12-13	NULL

17 rows in set (0.00 sec)

-- a) Copying the Original Table

```
USE sql_invoicing;
```

```
CREATE TABLE invoices_archived AS
```

```
SELECT i.invoice_id, i.number, c.name AS client, invoice_total, i.payment_total, i.invoice_date, i.payment_date,
i.due_date
```

```
FROM invoices AS i
```

```
JOIN clients AS c USING (client_id)
```

```
WHERE payment_date IS NOT NULL;
```

-- The new Table copied from the original one

```
SELECT * FROM invoices_archived;
```

invoice_id	number	client	invoice_total	payment_total	invoice_date	payment_date	due_date
6	75-587-6626	Vinte	157.78	74.55	2019-01-29	2019-01-03	2019-02-18
11	20-848-0181	Yadel	126.15	0.03	2019-01-07	2019-01-11	2019-01-27
15	55-105-9605	Yadel	167.29	80.31	2019-11-25	2019-01-15	2019-12-15
17	33-615-4694	Yadel	126.38	68.10	2019-07-30	2019-01-15	2019-08-19
2	03-898-6735	Topiclounge	175.32	8.18	2019-06-11	2019-02-12	2019-07-01
13	41-666-1035	Topiclounge	135.01	87.44	2019-06-25	2019-01-26	2019-07-15
18	52-269-9803	Topiclounge	180.17	42.77	2019-05-23	2019-01-08	2019-06-12

7 rows in set (0.00 sec)

## -- II. The original Table

```
SELECT * FROM invoices;
```

invoice_id	number	client_id	invoice_total	payment_total	invoice_date	due_date	payment_date
1	91-953-3396	2	101.79	0.00	2019-03-09	2019-03-29	NULL
2	03-898-6735	5	175.32	8.18	2019-06-11	2019-07-01	2019-02-12
3	20-228-0335	5	147.99	0.00	2019-07-31	2019-08-20	NULL
4	56-934-0748	3	152.21	0.00	2019-03-08	2019-03-28	NULL
5	87-052-3121	5	169.36	0.00	2019-07-18	2019-08-07	NULL
6	75-587-6626	1	157.78	74.55	2019-01-29	2019-02-18	2019-01-03
7	68-093-9863	3	133.87	0.00	2019-09-04	2019-09-24	NULL
8	78-145-1093	1	189.12	0.00	2019-05-20	2019-06-09	NULL
9	77-593-0081	5	172.17	0.00	2019-07-09	2019-07-29	NULL
10	48-266-1517	1	159.50	0.00	2019-06-30	2019-07-20	NULL
11	20-848-0181	3	126.15	0.03	2019-01-07	2019-01-27	2019-01-11
13	41-666-1035	5	135.01	87.44	2019-06-25	2019-07-15	2019-01-26
15	55-105-9605	3	167.29	80.31	2019-11-25	2019-12-15	2019-01-15
16	10-451-8824	1	162.02	0.00	2019-03-30	2019-04-19	NULL
17	33-615-4694	3	126.38	68.10	2019-07-30	2019-08-19	2019-01-15
18	52-269-9803	5	180.17	42.77	2019-05-23	2019-06-12	2019-01-08
19	83-559-4105	1	134.47	0.00	2019-11-23	2019-12-13	NULL

17 rows in set (0.00 sec)

## -- b) Updating the table - a single row for client\_id 1

```
USE sql_invoicing;
```

```
UPDATE invoices
```

```
SET payment_total = 10, payment_date = '2019-03-01'
```

```
WHERE invoice_id = 1
```

## -- Updated Table

```
SELECT * FROM invoices;
```

invoice_id	number	client_id	invoice_total	payment_total	invoice_date	due_date	payment_date
1	91-953-3396	2	101.79	10.00	2019-03-09	2019-03-29	2019-03-01
2	03-898-6735	5	175.32	8.18	2019-06-11	2019-07-01	2019-02-12
3	20-228-0335	5	147.99	0.00	2019-07-31	2019-08-20	NULL
4	56-934-0748	3	152.21	0.00	2019-03-08	2019-03-28	NULL
5	87-052-3121	5	169.36	0.00	2019-07-18	2019-08-07	NULL
6	75-587-6626	1	157.78	74.55	2019-01-29	2019-02-18	2019-01-03
7	68-093-9863	3	133.87	0.00	2019-09-04	2019-09-24	NULL
8	78-145-1093	1	189.12	0.00	2019-05-20	2019-06-09	NULL
9	77-593-0081	5	172.17	0.00	2019-07-09	2019-07-29	NULL
10	48-266-1517	1	159.50	0.00	2019-06-30	2019-07-20	NULL
11	20-848-0181	3	126.15	0.03	2019-01-07	2019-01-27	2019-01-11
13	41-666-1035	5	135.01	87.44	2019-06-25	2019-07-15	2019-01-26
15	55-105-9605	3	167.29	80.31	2019-11-25	2019-12-15	2019-01-15
16	10-451-8824	1	162.02	0.00	2019-03-30	2019-04-19	NULL
17	33-615-4694	3	126.38	68.10	2019-07-30	2019-08-19	2019-01-15
18	52-269-9803	5	180.17	42.77	2019-05-23	2019-06-12	2019-01-08
19	83-559-4105	1	134.47	0.00	2019-11-23	2019-12-13	NULL

17 rows in set (0.00 sec)

-- III. The original Table

-- c) Updating the table - multiple rows for client\_id 3 update all the invoices

invoice_id	number	client_id	invoice_total	payment_total	invoice_date	due_date	payment_date
1	91-953-3396	2	101.79	10.00	2019-03-09	2019-03-29	2019-03-01
2	03-898-6735	5	175.32	8.18	2019-06-11	2019-07-01	2019-02-12
3	20-228-0335	5	147.99	0.00	2019-07-31	2019-08-20	NULL
4	56-934-0748	3	152.21	0.00	2019-03-08	2019-03-28	NULL
5	87-052-3121	5	169.36	0.00	2019-07-18	2019-08-07	NULL
6	75-587-6626	1	157.78	74.55	2019-01-29	2019-02-18	2019-01-03
7	68-093-9863	3	133.87	0.00	2019-09-04	2019-09-24	NULL
8	78-145-1093	1	189.12	0.00	2019-05-20	2019-06-09	NULL
9	77-593-0081	5	172.17	0.00	2019-07-09	2019-07-29	NULL
10	48-266-1517	1	159.50	0.00	2019-06-30	2019-07-20	NULL
11	20-848-0181	3	126.15	0.03	2019-01-07	2019-01-27	2019-01-11
13	41-666-1035	5	135.01	87.44	2019-06-25	2019-07-15	2019-01-26
15	55-105-9605	3	167.29	80.31	2019-11-25	2019-12-15	2019-01-15
16	10-451-8824	1	162.02	0.00	2019-03-30	2019-04-19	NULL
17	33-615-4694	3	126.38	68.10	2019-07-30	2019-08-19	2019-01-15
18	52-269-9803	5	180.17	42.77	2019-05-23	2019-06-12	2019-01-08
19	83-559-4105	1	134.47	0.00	2019-11-23	2019-12-13	NULL

17 rows in set (0.00 sec)

-- Updating the Table

```
USE sql_invoicing;
```

```
UPDATE invoices
```

```
SET
```

```
    payment_total = invoice_total * 0.5,
```

```
    payment_date = due_date
```

```
WHERE client_id = 3;
```

-- Result: Updated Table

```
SELECT * FROM invoices;
```

invoice_id	number	client_id	invoice_total	payment_total	invoice_date	due_date	payment_date
1	91-953-3396	2	101.79	10.00	2019-03-09	2019-03-29	2019-03-01
2	03-898-6735	5	175.32	8.18	2019-06-11	2019-07-01	2019-02-12
3	20-228-0335	5	147.99	0.00	2019-07-31	2019-08-20	NULL
4	56-934-0748	3	152.21	76.11	2019-03-08	2019-03-28	2019-03-28
5	87-052-3121	5	169.36	0.00	2019-07-18	2019-08-07	NULL
6	75-587-6626	1	157.78	74.55	2019-01-29	2019-02-18	2019-01-03
7	68-093-9863	3	133.87	66.94	2019-09-04	2019-09-24	2019-09-24
8	78-145-1093	1	189.12	0.00	2019-05-20	2019-06-09	NULL
9	77-593-0081	5	172.17	0.00	2019-07-09	2019-07-29	NULL
10	48-266-1517	1	159.50	0.00	2019-06-30	2019-07-20	NULL
11	20-848-0181	3	126.15	63.08	2019-01-07	2019-01-27	2019-01-27
13	41-666-1035	5	135.01	87.44	2019-06-25	2019-07-15	2019-01-26
15	55-105-9605	3	167.29	83.65	2019-11-25	2019-12-15	2019-12-15
16	10-451-8824	1	162.02	0.00	2019-03-30	2019-04-19	NULL
17	33-615-4694	3	126.38	63.19	2019-07-30	2019-08-19	2019-08-19
18	52-269-9803	5	180.17	42.77	2019-05-23	2019-06-12	2019-01-08
19	83-559-4105	1	134.47	0.00	2019-11-23	2019-12-13	NULL

-- d) Updating all the invoices for the client whose name is "Myworks" but not knowing his client\_id

```
USE sql_invoicing;
UPDATE invoices
SET
    payment_total = invoice_total * 0.5,
    payment_date = due_date
WHERE client_id = (
    SELECT client_id
    FROM clients
    WHERE name = 'Myworks'
)
```

-- d) Delete all the invoices with id = 1

```
USE sql_invoicing;
DELETE FROM invoices
WHERE invoice_id = 1
```

## 20. Summarizing Data

-- Aggregate Functions

-- 1. Selecting the maximum, minimum, average, sum and count of the invoice\_total column from the invoices tables

SELECT

```
MAX(invoice_total) AS highest,
MIN(invoice_total) AS lowest,
AVG(invoice_total) AS average,
SUM(invoice_total) AS total,
COUNT(invoice_total) AS number_of_invoices,
COUNT(payment_date) AS number_payments_done,
COUNT(*) AS total_records -- Total Records on a Table
```

FROM invoices;

highest	lowest	average	total	number_of_invoices	number_payments_done	total_records
189.12	101.79	152.388235	2590.60	17	10	17

1 row in set (0.00 sec)

-- GROUP BY clause

-- 1. Calculating the total sales by client using the GROUP BY Clause

USE sql\_invoicing;

SELECT

```
client_id,
SUM(invoice_total) AS total_sales
```

FROM invoices

GROUP BY client\_id

ORDER BY total\_sales DESC;

client_id	total_sales
5	980.02
1	802.89
3	705.90
2	101.79

4 rows in set (0.01 sec)

-- HAVING clause

-- HAVING is used to filter data AFTER we group our rows. WHERE is used to filter data BEFORE we group our rows.

include only the clients that have a total\_sale of more than 500 \$

```

SELECT
  client_id,
  SUM(invoice_total) AS total_sales
FROM invoices
GROUP BY client_id
HAVING total_sales > 500;

```

client_id	total_sales
1	802.89
3	705.90
5	980.02

## -- ROLLUP operator

-- 3. Grouping by multiple columns

-- The ROLLUP in MySQL is a modifier used to produce the summary output, including extra rows that represent super-aggregate (higher-level) summary operations. It enables us to sum-up the output at multiple levels of analysis using a single query.

```

USE sql_invoicing;
SELECT
  state,
  city,
  SUM(invoice_total) AS total_sales
FROM invoices AS i
JOIN clients AS c USING (client_id)
GROUP BY state, city WITH ROLLUP;

```

state	city	total_sales
CA	San Francisco	705.90
CA	NULL	705.90
NY	Syracuse	802.89
NY	NULL	802.89
OR	Portland	980.02
OR	NULL	980.02
WV	Huntington	101.79
WV	NULL	101.79
NULL	NULL	2590.60

9 rows in set (0.00 sec)

## 21. Writing Complex Query

### a) With Subqueries

```

-- 1. Find all the product that are more expensive than Lettuce (id =3)
USE sql_inventory;
SELECT *
FROM products
WHERE unit_price > (
  SELECT unit_price

```

```
FROM products
WHERE product_id = 3
)
```

product_id	name	quantity_in_stock	unit_price
2	Pork – Bacon,back Peameal	49	4.65
4	Brocolinni – Gaylan, Chinese	90	4.53

b) With IN operator

```
-- 1. Find the products that have never been ordered.
USE sql_store;
SELECT *
FROM products
WHERE product_id NOT IN (
    SELECT DISTINCT product_id -- these are the products that have been ordered
    FROM order_items
);
```

product_id	name	quantity_in_stock	unit_price
7	Sweet Pea Sprouts	98	3.29

Subqueries VS JOIN

-- Select all clients and then join those with invoice table

-- With Subquery	-- With JOIN
<pre>USE sql_invoicing; SELECT * FROM clients WHERE client_id NOT IN(     SELECT DISTINCT client_id     FROM invoices );</pre>	<pre>SELECT c.client_id, c.name, c.address, c.city, c.state, c.phone FROM clients AS c LEFT JOIN invoices USING (client_id) WHERE invoice_id IS NULL;</pre>

client_id	name	address	city	state	phone
4	Kvideo	81674 Westerfield Circle	Waco	TX	254-750-0784

ALL keyword

-- 1. Select invoices larger than all the invoices of the client 3

-- Without ALL keyword

```
USE sql_invoicing;
SELECT *
```



```

FROM invoices
WHERE invoice_total > (
    SELECT MAX(invoice_total) -- returning a single value
    FROM invoices
    WHERE client_id = 3
);

```

-- With ALL keyword

```

USE sql_invoicing;
SELECT *
FROM invoices
WHERE invoice_total > ALL (
    SELECT invoice_total -- returning a list of value
    FROM invoices
    WHERE client_id = 3
);

```

invoice_id	number	client_id	invoice_total	payment_total	invoice_date	due_date	payment_date
2	03-898-6735	5	175.32	8.18	2019-06-11	2019-07-01	2019-02-12
5	87-052-3121	5	169.36	0.00	2019-07-18	2019-08-07	NULL
8	78-145-1093	1	189.12	0.00	2019-05-20	2019-06-09	NULL
9	77-593-0081	5	172.17	0.00	2019-07-09	2019-07-29	NULL
18	52-269-9803	5	180.17	42.77	2019-05-23	2019-06-12	2019-01-08

ANY Key word

-- Returning the clients with at least two invoices.

-- Without ANY keyword

```

SELECT *
FROM clients
WHERE client_id IN (
    SELECT client_id
    FROM invoices
    GROUP BY client_id
    HAVING COUNT(*) >= 2
);

```

-- With ANY keyword

```
SELECT *
FROM clients
WHERE client_id = ANY (
    SELECT client_id
    FROM invoices
    GROUP BY client_id
    HAVING COUNT(*) >= 2
);
```

client_id	name	address	city	state	phone
1	Vinte	3 Nevada Parkway	Syracuse	NY	315-252-7305
3	Yadel	096 Pawling Parkway	San Francisco	CA	415-144-6037
5	Topiclounge	0863 Farmco Road	Portland	OR	971-888-9129

Correlated Subqueries.sql

--

-- Select employees whose salary is above the average in their office (in the same office)  
 -- Correlated subqueries get executed in each row in the main query

```
USE sql_hr;
SELECT *
FROM employees AS e
WHERE salary > (
    SELECT AVG(salary)
    FROM employees
    WHERE office_id = e.office_id -- Salary in the same office
);
```

employee_id	first_name	last_name	job_title	salary	reports_to	office_id
37851	Sayer	Matterson	Statistician III	98926	37270	1
40448	Mindy	Crissil	Staff Scientist	94860	37270	1
56274	Kerian	Alloisi	VP Marketing	110150	37270	1
67009	North	de Clerc	VP Product Management	114257	37270	2
67370	Elladine	Rising	Social Worker	96767	37270	2
72540	Guthrey	Iacopetti	Office Assistant I	117690	37270	3
76196	Mirilla	Janowski	Cost Accountant	119241	37270	3
84791	Hazel	Tarbert	General Manager	93760	37270	4
95213	Cole	Kesterton	Pharmacist	86119	37270	4
98374	Estrellita	Daleman	Staff Accountant IV	70187	37270	5
115357	Ivy	Fearey	Structural Engineer	92710	37270	5

## EXISTS Operator

-- Return the products that have never been ordered

-- With IN operator

```
USE sql_store;
SELECT *
FROM products
WHERE product_id NOT IN(
    SELECT product_id
    FROM order_items
);
```

-- With EXISTS operator

-- 2) second solution with EXISTS

```
USE sql_store;
SELECT *
FROM products AS p
WHERE NOT EXISTS(
    SELECT product_id
    FROM order_items
    WHERE product_id = p.product_id
);
```

product_id	name	quantity_in_stock	unit_price
7	Sweet Pea Sprouts	98	3.29

## Subqueries in the SELECT Clause

--

```
USE sql_invoicing;
SELECT
    invoice_id,
    invoice_total,
    (SELECT AVG(invoice_total)
     FROM invoices) AS invoice_average,
    invoice_total - (SELECT invoice_average) AS difference
FROM invoices;
```

invoice_id	invoice_total	invoice_average	difference
1	101.79	152.388235	-50.598235
2	175.32	152.388235	22.931765
3	147.99	152.388235	-4.398235
4	152.21	152.388235	-0.178235
5	169.36	152.388235	16.971765
6	157.78	152.388235	5.391765
7	133.87	152.388235	-18.518235
8	189.12	152.388235	36.731765
9	172.17	152.388235	19.781765
10	159.50	152.388235	7.111765
11	126.15	152.388235	-26.238235
13	135.01	152.388235	-17.378235
15	167.29	152.388235	14.901765
16	162.02	152.388235	9.631765
17	126.38	152.388235	-26.008235
18	180.17	152.388235	27.781765
19	134.47	152.388235	-17.918235

## Subqueries in the FROM Clause

--

```
SELECT *
FROM (SELECT
  client_id,
  name,
  (SELECT SUM(invoice_total)
   FROM invoices
   WHERE client_id = c.client_id) AS total_sales,
  (SELECT AVG(invoice_total) FROM invoices) AS average,
  (SELECT total_sales - average) AS difference
FROM clients AS c
) AS sales_summary -- Giving an Alias is required for this cases
WHERE total_sales IS NOT NULL;
```

client_id	name	total_sales	average	difference
1	Vinte	802.89	152.388235	650.501765
2	Myworks	101.79	152.388235	-50.598235
3	Yadel	705.90	152.388235	553.511765
5	Topiclounge	980.02	152.388235	827.631765

## 22. Essential MySQL Functions

### Numeric Functions

```
--  
SELECT ROUND(5.73)      -- 1. Round  
SELECT ROUND(5.7345, 2) -- 2. Truncate  
SELECT CEILING(5.2)  
SELECT FLOOR(5.2)  
SELECT ABS(5.2)  
SELECT RAND();
```

### String Functions

```
--  
SELECT LENGTH('sky')      -- 1) Length  
SELECT UPPER('sky')       -- 2) Upper  
SELECT LOWER('SKY')       -- 3) Lower  
  
-- 4. Remove unnecessary spaces  
SELECT LTRIM(' Sky')      -- a) LTRIM  
SELECT RTRIM('SKY  ')    -- b) RTRIM  
SELECT TRIM(' SKY  ')    -- c) TRIM  
SELECT LEFT('Kindergarten', 4)  
SELECT RIGHT('Kindergarten', 5)  
SELECT SUBSTRING('Kindergarten', 3) -- 7. SUBSTRING  
SELECT LOCATE('n','Kindergarten') -- 8. LOCATE - locate the position  
SELECT REPLACE('Kindergarten', 'garten', 'alain')  
SELECT CONCAT('Alain_', 'Pedro')
```

## Date and time Functions

```
--  
SELECT NOW();      -- current date and time  
SELECT CURDATE();  -- current date  
SELECT CURTIME();  -- current time  
SELECT YEAR(NOW()); -- current year  
SELECT MONTH(NOW()); -- current month  
SELECT DAY(NOW());  -- current day  
SELECT HOUR(NOW()); -- current hour  
SELECT MINUTE(NOW()); -- current minute  
SELECT SECOND(NOW()); -- current second  
  
SELECT EXTRACT(YEAR FROM NOW()); -- current year  
SELECT EXTRACT(MONTH FROM NOW()); -- current month  
SELECT EXTRACT(DAY FROM NOW()); -- current day  
  
SELECT MONTHNAME(NOW()); -- current Month as a STRING  
SELECT DAYNAME(NOW()); -- current Day as a STRING
```

## Formatting Date and Time

```
--  
-- 1. Date  
SELECT DATE_FORMAT(NOW(), '%m %d %y') AS 'Date'  
SELECT DATE_FORMAT(NOW(), '%M %D %Y') AS 'Date' -- This format is better.  
  
-- 2. Time  
SELECT DATE_FORMAT(NOW(), '%H:%i %p') AS 'Time' -- PM  
SELECT DATE_FORMAT(NOW(), '%H:%i') AS 'Time'
```

## Calculating Dates and Times

```
--  
-- 1. Returning the next day, month and year  
SELECT DATE_ADD(NOW(), INTERVAL 1 DAY);  
SELECT DATE_ADD(NOW(), INTERVAL 1 MONTH);  
SELECT DATE_ADD(NOW(), INTERVAL 1 YEAR);  
  
-- 2. Returning the last day, month and year  
SELECT DATE_ADD(NOW(), INTERVAL -1 DAY);  
SELECT DATE_SUB(NOW(), INTERVAL 1 DAY);
```

```

SELECT DATE_ADD(NOW() , INTERVAL -1 MONTH);
SELECT DATE_SUB(NOW() , INTERVAL 1 MONTH);

SELECT DATE_ADD(NOW() , INTERVAL -1 YEAR);
SELECT DATE_SUB(NOW() , INTERVAL 1 YEAR);

-- Returning the difference of the days between two times
SELECT DATEDIFF('2019-01-05 09:00', '2019-01-01 17:00');
SELECT DATEDIFF('2019-01-01 17:00', '2019-01-05 09:00');

-- Returning the difference of the days from two date
SELECT TIME_TO_SEC('09:02') - TIME_TO_SEC('2019-01-01 09:00');

```

## The IFNULL Function

```
--
```

<pre> USE sql_store;  SELECT     order_id,     IFNULL(shipper_id, 'Not assigned') AS Shipper FROM orders; </pre>	<table> <thead> <tr> <th>order_id</th> <th>Shipper</th> </tr> </thead> <tbody> <tr><td>1</td><td>Not assigned</td></tr> <tr><td>3</td><td>Not assigned</td></tr> <tr><td>4</td><td>Not assigned</td></tr> <tr><td>6</td><td>Not assigned</td></tr> <tr><td>8</td><td>Not assigned</td></tr> <tr><td>11</td><td>Not assigned</td></tr> <tr><td>9</td><td>1</td></tr> <tr><td>10</td><td>2</td></tr> <tr><td>5</td><td>3</td></tr> <tr><td>2</td><td>4</td></tr> <tr><td>7</td><td>4</td></tr> </tbody> </table>	order_id	Shipper	1	Not assigned	3	Not assigned	4	Not assigned	6	Not assigned	8	Not assigned	11	Not assigned	9	1	10	2	5	3	2	4	7	4
order_id	Shipper																								
1	Not assigned																								
3	Not assigned																								
4	Not assigned																								
6	Not assigned																								
8	Not assigned																								
11	Not assigned																								
9	1																								
10	2																								
5	3																								
2	4																								
7	4																								

## The COALESCE Function

```
--
```

```

USE sql_store;

SELECT
    order_id,
    comments,
    COALESCE(shipper_id, comments, 'Not assigned') AS Shipper
FROM orders;

```

order_id	comments	Shipper
1	NULL	Not assigned
2	NULL	4
3	NULL	Not assigned
4	NULL	Not assigned
5	NULL	3
6	Aliquam erat volutpat. In congue.	Aliquam erat volutpat. In congue.
7	NULL	4
8	Mauris enim leo, rhoncus sed, vestibulum sit amet, cursus id, turpis.	Mauris enim leo, rhoncus sed, vestibulum sit amet, cursus id, turpis.
9	Nulla mollis molestie lorem. Quisque ut erat.	1
10	NULL	2
11	NULL	Not assigned

## The IF Function

```
--
-- If the order belongs to this year returns 'Active' otherwise 'Archived'
USE sql_store;

SELECT
    order_id,
    order_date,
    IF(
        YEAR(order_date) = YEAR(NOW()),
        'Active',
        'Archived'
    ) AS category
FROM orders;
```

order_id	order_date	category
1	2019-01-30	Archived
2	2018-08-02	Archived
3	2017-12-01	Archived
4	2017-01-22	Archived
5	2017-08-25	Archived
6	2018-11-18	Archived
7	2018-09-22	Archived
8	2018-06-08	Archived
9	2017-07-05	Archived
10	2018-04-22	Archived
11	2019-01-02	Archived

## The CASE Operator

```
--
/*Return the categories below based on the order place year:
- current year -> 'Active'
- last year -> 'Last year'
```



```

- earlier -> 'Archived'
*/
USE sql_store;

SELECT
    order_id,
    order_date,
    CASE
        WHEN YEAR(order_date) = YEAR(NOW()) THEN 'Active'
        WHEN YEAR(order_date) = YEAR(NOW()) - 1 THEN 'Last Year'
        WHEN YEAR(order_date) < YEAR(NOW()) - 1 THEN 'Archived'
        ELSE 'Future'
    END AS category
FROM orders;

```

order_id	order_date	category
1	2019-01-30	Archived
2	2018-08-02	Archived
3	2017-12-01	Archived
4	2017-01-22	Archived
5	2017-08-25	Archived
6	2018-11-18	Archived
7	2018-09-22	Archived
8	2018-06-08	Archived
9	2017-07-05	Archived
10	2018-04-22	Archived
11	2019-01-02	Archived

## 23. Stored Procedures

### Creating a Stored Procedure

```
--  
-- In other DBMS such SQL Server, we don't have to change the default delimiter  
-- 1. Creating a get_clients Procedure  
USE sql_invoicing;  
  
DELIMITER $$  
CREATE PROCEDURE get_clients()  
BEGIN  
    SELECT *  
    FROM clients;  
END $$  
  
DELIMITER ;  
  
-- 2. call a procedure  
CALL get_clients();
```

client_id	name	address	city	state	phone
1	Vinte	3 Nevada Parkway	Syracuse	NY	315-252-7305
2	Myworks	34267 Glendale Parkway	Huntington	WV	304-659-1170
3	Yadel	096 Pawling Parkway	San Francisco	CA	415-144-6037
4	Kwideo	81674 Westerfield Circle	Waco	TX	254-750-0784
5	Topiclounge	0863 Farmco Road	Portland	OR	971-888-9129

### Dropping a Procedure

```
--  
DROP PROCEDURE get_clients;           -- First way  
DROP PROCEDURE IF EXISTS get_clients; -- Second way
```

## Add Parameter to a Stored Procedure

```
--  
SELECT * FROM clients;
```

client_id	name	address	city	state	phone
1	Vinte	3 Nevada Parkway	Syracuse	NY	315-252-7305
2	Myworks	34267 Glendale Parkway	Huntington	WV	304-659-1170
3	Yadel	096 Pawling Parkway	San Francisco	CA	415-144-6037
4	Kwideo	81674 Westerfield Circle	Waco	TX	254-750-0784
5	Topiclounge	0863 Farmco Road	Portland	OR	971-888-9129

```
-- 1. Writing a procedure that receive the name of the state and return the clients in that state  
DROP PROCEDURE IF EXISTS get_clients_by_state;  
  
DELIMITER $$  
CREATE PROCEDURE get_clients_by_state  
(  
  -- parameter  
  state CHAR(2) -- state: NA  
)  
BEGIN  
  SELECT *  
  FROM clients AS c  
  WHERE c.state = state; -- comparing the value in the state column, with the state parameter defined above  
      -- column = parameter  
END $$  
DELIMITER ;  
  
-- 2. Calling the procedure  
CALL get_clients_by_state('CA');
```

client_id	name	address	city	state	phone
3	Yadel	096 Pawling Parkway	San Francisco	CA	415-144-6037

```
--
```

## Assign a Default Value to a Parameter

```
--  
-- 1. Setting within the procedure that When calling the procedure if the state is not inserted, so return "CA" by  
default  
DELIMITER $$  
CREATE PROCEDURE get_clients_by_state  
(  
  -- parameter  
  state CHAR(2) -- state: NA  
)  
BEGIN  
  IF state IS NULL THEN  
    SET state = 'CA';  
  END IF;  
  
  SELECT *  
  FROM clients AS c  
  WHERE c.state = state; -- comparing the value in the state column, with the state parameter defined above  
    -- column = parameter  
END $$  
DELIMITER ;  
  
-- 2. Call the created procedure  
CALL get_clients_by_state(NULL);
```

client_id	name	address	city	state	phone
3	Yadel	096 Pawling Parkway	San Francisco	CA	415-144-6037

## Parameter Validation

```
--  
-- 2. Validate the arguments that we're to this make_payment procedure  
DROP PROCEDURE IF EXISTS make_payment;  
DELIMITER $$  
CREATE PROCEDURE make_payment  
(  
    invoice_id INT,  
    payment_amount DECIMAL(9,2),  
    payment_date DATE  
)  
BEGIN  
    IF payment_amount <= 0 THEN  
        SIGNAL SQLSTATE '22003' -- raise an error!!  
        SET MESSAGE_TEXT = 'Invalid payment amount';  
    END IF;  
    UPDATE invoices AS i  
    SET  
        i.payment_total = payment_amount,  
        i.payment_date = payment_date  
    WHERE i.invoice_id = invoice_id;  
END $$  
DELIMITER ;  
  
-- Calling the procedure  
CALL make_payment(2, -100, '2019-01-01');
```

```
mysql> CALL make_payment(2, -100, '2019-01-01');  
ERROR 1644 (22003): Invalid payment amount
```

## Output Parameters

```
--
```

Variables

--

Function

--

## **24. Triggers and Events**

Triggers

--