

SQL Commands

1-SELECT=

The SELECT statement is used to select data from a database.

The data returned is stored in a result table, called the result-set.

```
-- select all from customer table
```

```
SELECT *
```

```
FROM customers;
```

```
-- select specific columns
```

```
SELECT first_name, last_name
```

```
FROM customers;
```

```
-- select with arthemetic expressions
```

```
SELECT first_name, last_name, points, points+10
```

```
FROM customers;
```

```
-- select with an alias
```

```
SELECT
```

```
first_name,
```

last_name,

points,

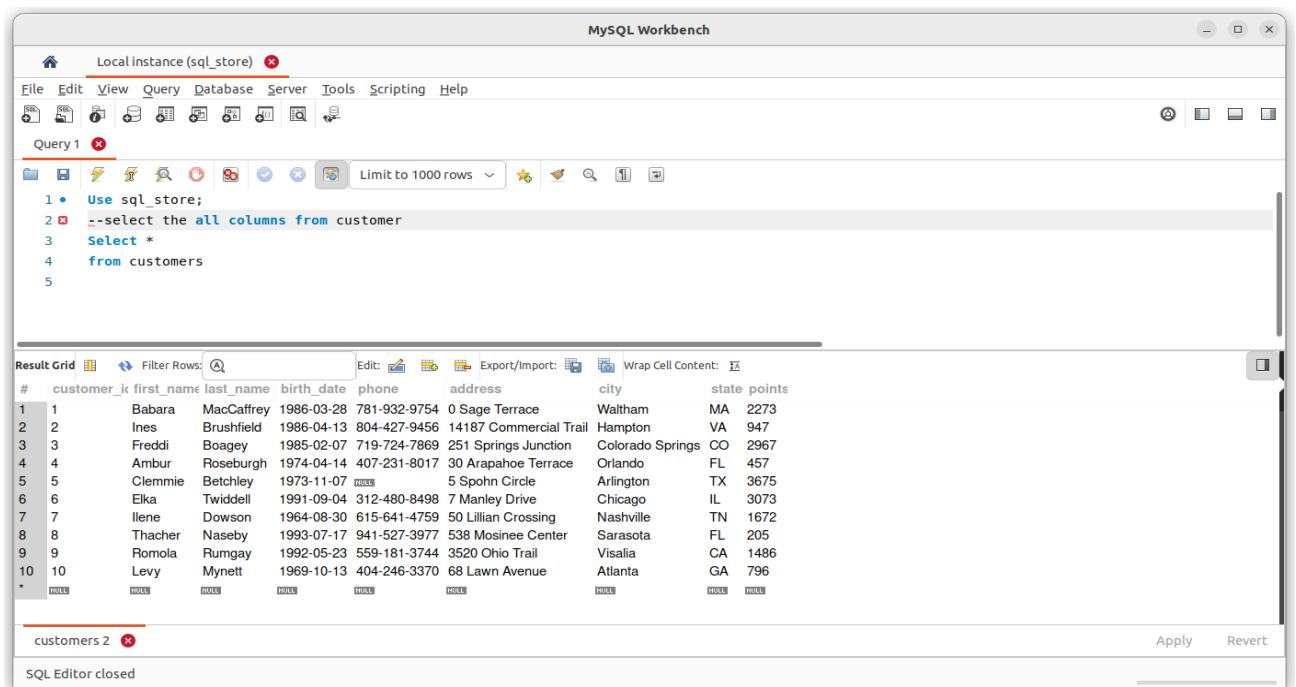
points+10 AS Points_new

FROM customers;

-- select distinct values

SELECT DISTINCT state

FROM customers;



MySQL Workbench

Local Instance

File Edit View Query Database Server Tools Scripting Help

Query 1

```
1 • USE sql_store;
2 • SELECT last_name,first_name,points,points+10*100
3   FROM customers
4
5
```

Result Grid Filter Rows: Export: Wrap Cell Content:

#	last_name	first_name	points	points+10*100
1	MacCaffrey	Barbara	2273	3273
2	Brushfield	Ines	947	1947
3	Boagey	Freddi	2967	3967
4	Roseburgh	Ambur	457	1457
5	Betchley	Clemmie	3675	4675
6	Twiddell	Elka	3073	4073
7	Dowson	Ilene	1672	2672
8	Naseby	Thacher	205	1205
9	Rumgay	Romola	1486	2486
10	Mynett	Levy	796	1796

Result 2

Query Completed

Result Grid Form Editor Field Types Query Stats

MySQL Workbench

Local Instance

File Edit View Query Database Server Tools Scripting Help

Query 1

```
1 • USE sql_store;
2 • SELECT last_name,first_name,points,
3     (points+10)*100 AS 'discount factor'
4 FROM customers
5
6
```

Result Grid Filter Rows: Export: Wrap Cell Content:

#	last_name	first_name	points	discount facts
1	MacCaffrey	Barbara	2273	228300
2	Brushfield	Ines	947	95700
3	Boagey	Freddi	2967	297700
4	Roseburgh	Ambur	457	46700
5	Betchley	Clemmie	3675	368500
6	Twiddell	Elka	3073	308300
7	Dowson	Ilene	1672	168200
8	Naseby	Thacher	205	21500
9	Rumgay	Romola	1486	149600
10	Mynett	Levy	796	80600

Result 3

Query Completed

Result Grid Form Editor Field Types Query Stats

MySQL Workbench

Local Instance

File Edit View Query Database Server Tools Scripting Help

Schemas

- sql_hr
- sql_inventory
- sql_invoicing
- sql_store
- sys

Object Info Session

Schema: sql_store

Query 1

```
1 • USE sql_store;
2 • SELECT state
3 FROM customers;
```

Result Grid Filter Rows: Export: Wrap Cell Content:

#	state
1	VA
2	VA
3	CO
4	FL
5	TX
6	IL
7	TN
8	FL
9	CA
10	GA

customers 6

Query Completed

Result Grid Form Editor Field Types Query Stats

The screenshot shows the MySQL Workbench interface. In the top navigation bar, 'Local Instance' is selected. The left sidebar displays the 'SCHEMAS' tree, with 'sql_store' selected. The main area contains a 'Query 1' tab with the following SQL code:

```
1 • USE sql_store;
2 • SELECT DISTINCT state
3   FROM customers;
4
5
```

The results are displayed in a 'Result Grid' table:

#	state
1	VA
2	CO
3	FL
4	TX
5	IL
6	TN
7	CA
8	GA

The status bar at the bottom indicates 'Query Completed'.

2- where-The **WHERE** clause is used to filter records.

It is used to extract only those records that fulfill a specified condition.

Basic Syntax-

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

MySQL Workbench

Local Instance

File Edit View Query Database Server Tools Scripting Help

Schemas

sql_store

Tables

customers

order_item_notes

order_items

orders

order_statuses

products

shippers

Views

Stored Procedures

Object Info Session

Table: order_items

Columns:

- order_id int AI PK
- product_id int PK
- quantity int
- unit_price decimal(4,2)

Query 1 customers

```
1 • USE sql_store;
2 • SELECT *
3 FROM customers
4 where points>3000;
5
6 • FROM products;
7
```

Result Grid

#	customer_id	first_name	last_name	birth_date	phone	address	city	state	points
1	5	Clemmie	Betchley	1973-11-07	NULL	5 Spohn Circle	Arlington	TX	3675
2	6	Elka	Twiddell	1991-09-04	312-480-8498	7 Manley Drive	Chicago	IL	3073
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

customers 10

Apply Revert

Query interrupted

MySQL Workbench

Local Instance

File Edit View Query Database Server Tools Scripting Help

Schemas

sql_store

Tables

customers

order_item_notes

order_items

orders

order_statuses

products

shippers

Views

Stored Procedures

Object Info Session

Table: order_items

Columns:

- order_id int AI PK
- product_id int PK
- quantity int
- unit_price decimal(4,2)

Query 1 customers

```
1 • USE sql_store;
2 -- where clause with string
3 • SELECT *
4 FROM customers
5 WHERE state='VA';
6
7
```

Result Grid

#	customer_id	first_name	last_name	birth_date	phone	address	city	state	points
1	1	Babara	MacCaffrey	1986-03-28	781-932-9754	0 Sage Terrace	Waltham	VA	2273
2	2	Ines	Brushfield	1986-04-13	804-427-9456	14187 Commercial Trail	Hampton	VA	947
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

customers 12

Apply Revert

Query Completed

MySQL Workbench

Local Instance

File Edit View Query Database Server Tools Scripting Help

Schemas

sql_store

Tables

customers

order_item_notes

order_items

orders

order_statuses

products

shippers

Views

Stored Procedures

Object Info Session

Table: order_items

Columns:

- order_id int AI PK
- product_id int PK
- quantity int
- unit_price decimal(4,2)

Query 1 customers

```

1 • USE sql_store;
2 -- where clause with string
3 • SELECT *
4 FROM customers
5 WHERE birth_date>'1990-01-01';
6
7

```

Result Grid

#	customer_id	first_name	last_name	birth_date	phone	address	city	state	points
1	6	Elka	Twiddell	1991-09-04	312-480-8498	7 Manley Drive	Chicago	IL	3073
2	8	Thacher	Naseby	1993-07-17	941-527-3977	538 Mosinee Center	Sarasota	FL	205
3	9	Romola	Rumgay	1992-05-23	559-181-3744	3520 Ohio Trail	Visalia	CA	1486
*	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL

customers 13

Apply Revert

Query Completed

MySQL Workbench

Local Instance

File Edit View Query Database Server Tools Scripting Help

Schemas

sql_store

Tables

customers

order_item_notes

order_items

orders

order_statuses

products

shippers

Views

Stored Procedures

Object Info Session

Table: order_items

Columns:

- order_id int AI PK
- product_id int PK
- quantity int
- unit_price decimal(4,2)

Query 1 customers

```

1 • USE sql_store;
2 -- get the orders placed this year
3 • SELECT *
4 FROM orders
5 WHERE order_date>='2019-01-01';
6
7

```

Result Grid

#	order_id	customer_id	order_date	status	comments	shipped_date	shipper_id
1	1	6	2019-01-30	1	HULL	HULL	HULL
*	HULL	HULL	HOLE	HULL	HULL	HULL	HULL

orders 15

Apply Revert

Query Completed

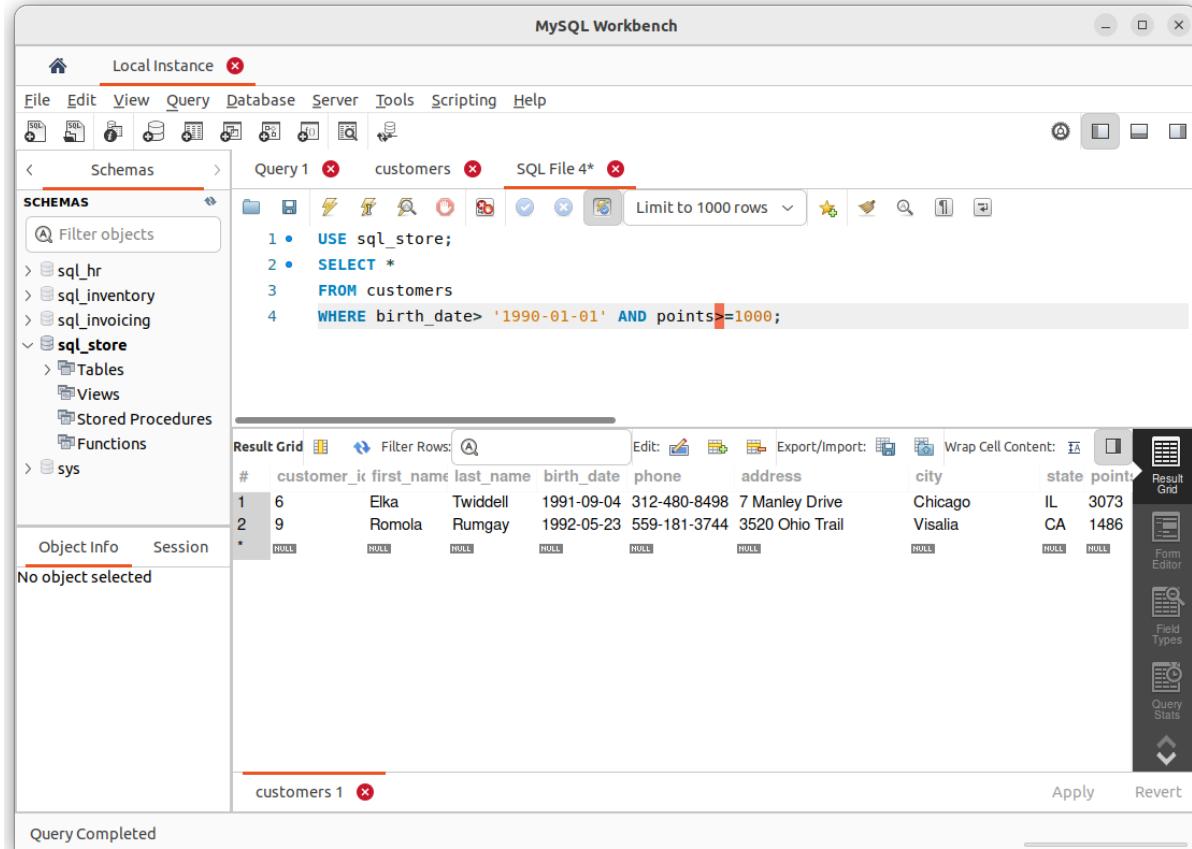
The SQL AND, OR and NOT Operators

The **WHERE** clause can be combined with **AND**, **OR**, and **NOT** operators.

The **AND** and **OR** operators are used to filter records based on more than one condition:

- The **AND** operator displays a record if all the conditions separated by **AND** are TRUE.
- The **OR** operator displays a record if any of the conditions separated by **OR** is TRUE.

The **NOT** operator displays a record if the condition(s) is NOT TRUE.



The screenshot shows the MySQL Workbench interface with a query editor and a results grid. The query editor contains the following SQL code:

```
1 • USE sql_store;
2 • SELECT *
3 FROM customers
4 WHERE birth_date > '1990-01-01' AND points >= 1000;
```

The results grid displays the following data:

#	customer_id	first_name	last_name	birth_date	phone	address	city	state	points
1	6	Elka	Twiddell	1991-09-04	312-480-8498	7 Manley Drive	Chicago	IL	3073
2	9	Romola	Rumgay	1992-05-23	559-181-3744	3520 Ohio Trail	Visalia	CA	1486
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

The status bar at the bottom left says "Query Completed".

MySQL Workbench

Local Instance

File Edit View Query Database Server Tools Scripting Help

Schemas

Query 1 customers SQL File 4*

```

1 • USE sql_store;
2 • SELECT *
3   FROM customers
4   WHERE birth_date>'1990-01-01' OR points>=1000;
    
```

Result Grid

#	customer_id	first_name	last_name	birth_date	phone	address	city	state	points
1	1	Barbara	MacCaffrey	1986-03-28	781-932-9754	0 Sage Terrace	Waltham	VA	2273
2	3	Freddi	Boagey	1985-02-07	719-724-7869	251 Springs Junction	Colorado Springs	CO	2967
3	5	Clemmie	Betchley	1973-11-07	NULL	5 Spohn Circle	Arlington	TX	3675
4	6	Elka	Twiddell	1991-09-04	312-480-8498	7 Manley Drive	Chicago	IL	3073
5	7	Ilene	Dowson	1964-08-30	615-641-4759	50 Lillian Crossing	Nashville	TN	1672
6	8	Thacher	Naseby	1993-07-17	941-527-3977	538 Mosinee Center	Sarasota	FL	205
7	9	Romola	Rumgay	1992-05-23	559-181-3744	3520 Ohio Trail	Visalia	CA	1486
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

customers 2

Query Completed

MySQL Workbench

Local Instance

File Edit View Query Database Server Tools Scripting Help

Schemas

Query 1 customers SQL File 4*

```

1 • USE sql_store;
2 • SELECT *
3   FROM customers
4   WHERE NOT (birth_date >'1990-01-01' OR state='VA');
    
```

Result Grid

#	customer_id	first_name	last_name	birth_date	phone	address	city	state	points
1	3	Freddi	Boagey	1985-02-07	719-724-7869	251 Springs Junction	Colorado Springs	CO	2967
2	4	Ambur	Roseburgh	1974-04-14	407-231-8017	30 Arapahoe Terrace	Orlando	FL	457
3	5	Clemmie	Betchley	1973-11-07	NULL	5 Spohn Circle	Arlington	TX	3675
4	7	Ilene	Dowson	1964-08-30	615-641-4759	50 Lillian Crossing	Nashville	TN	1672
5	10	Levy	Mynett	1969-10-13	404-246-3370	68 Lawn Avenue	Atlanta	GA	796
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

customers 7

Query Completed

IN operator-

```
-- in statement
SELECT *
FROM customers
WHERE state IN ('VA' , 'GA' , 'FL')
```

The screenshot shows the MySQL Workbench interface with a query editor and a results grid.

Query Editor:

```
1 • USE sql_store;
2 • SELECT *
3   FROM products
4   WHERE quantity_in_stock NOT IN (49,38,72);
```

Result Grid:

#	product_id	name	quantity_in_stoc	unit_price
1	1	Foam Dinner Plate	70	1.21
2	4	Broccolini - Gaylan, Chinese	90	4.53
3	5	Sauce - Ranch Dressing	94	1.63
4	6	Petit Baguette	14	2.39
5	7	Sweet Pea Sprouts	98	3.29
6	8	Island Oasis - Raspberry	26	0.74
7	9	Longan	67	2.26
8	10	Broom - Push	6	1.09
*	NULL	NULL	NULL	NULL

Status Bar: products 2

Message Bar: Query Completed

BETWEEN-

```
SELECT *
FROM table_name
WHERE column_name Between value_1 AND value_2;
```

```

USE sql_store;
-- between statement
SELECT *
FROM products
WHERE quantity_in_stock BETWEEN 70 AND 90;

```

#	product_id	name	quantity_in_stock	unit_price
1	1	Foam Dinner Plate	70	1.21
2	4	Brocolinni - Gaylan, Chinese	90	4.53
	HULL	HULL	HULL	HULL

LIKE operator-

The **LIKE** operator is used in a **WHERE** clause to search for a specified pattern in a column.

There are two wildcards often used in conjunction with the **LIKE** operator:

- The percent sign (%) represents zero, one, or multiple characters
- The underscore sign (_) represents one, single character

LIKE SYNTAX-

```

SELECT column1, column2, ...
FROM table_name
WHERE columnN LIKE pattern;

```

MySQL Workbench

Local Instance

File Edit View Query Database Server Tools Scripting Help

Schemas

Query 1 customers SQL File 4* SQL File 5* SQL File 6* products

Limit to 1000 rows

```
1 • USE sql_store;
2 -- last_name start with b
3 • SELECT *
4 FROM customers
5 WHERE last_name like 'b%';
```

Result Grid

#	customer_id	first_name	last_name	birth_date	phone	address	city	state	points
1	2	Ines	Brushfield	1986-04-13	804-427-9456	14187 Commercial Trail	Hampton	VA	947
2	3	Freddi	Boagey	1985-02-07	719-724-7869	251 Springs Junction	Colorado Springs	CO	2967
3	5	Clemmie	Betchley	1973-11-07	NULL	5 Spohn Circle	Arlington	TX	3675
*	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL

customers 6

Apply Revert

Query Completed

MySQL Workbench

Local Instance

File Edit View Query Database Server Tools Scripting Help

Schemas

Query 1 customers SQL File 4* SQL File 5* SQL File 6* products

Limit to 1000 rows

```
1 • USE sql_store;
2 -- last_name start with before and after b
3 • SELECT *
4 FROM customers
5 WHERE last_name like '%b%';
```

Result Grid

#	customer_id	first_name	last_name	birth_date	phone	address	city	state	points
1	2	Ines	Brushfield	1986-04-13	804-427-9456	14187 Commercial Trail	Hampton	VA	947
2	3	Freddi	Boagey	1985-02-07	719-724-7869	251 Springs Junction	Colorado Springs	CO	2967
3	4	Amber	Roseburgh	1974-04-14	407-231-8017	30 Arapahoe Terrace	Orlando	FL	457
4	5	Clemmie	Betchley	1973-11-07	NULL	5 Spohn Circle	Arlington	TX	3675
5	8	Thacher	Naseby	1993-07-17	941-527-3977	538 Mosinee Center	Sarasota	FL	205
*	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL

customers 7

Apply Revert

Query Completed

MySQL Workbench

Local Instance

File Edit View Query Database Server Tools Scripting Help

Schemas: customers, SQL File 4*, SQL File 5*, SQL File 6*, products, SQL File 7*

Query 1: customers

```

1 • USE sql_store;
2 -- get the customer whose address contain trail or avenue
3 • SELECT *
4 FROM customers
5 WHERE address LIKE '%trail%'
6 OR address LIKE '%AVENUE';

```

Result Grid:

#	customer_id	first_name	last_name	birth_date	phone	address	city	state	points
1	2	Ines	Brushfield	1986-04-13	804-427-9456	14187 Commercial Trail	Hampton	VA	947
2	9	Romola	Rumgay	1992-05-23	559-181-3744	3520 Ohio Trail	Visalia	CA	1486
3	10	Levy	Mynett	1969-10-13	404-246-3370	68 Lawn Avenue	Atlanta	GA	796
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

customers 8

Query Completed

MySQL Workbench

Local Instance

File Edit View Query Database Server Tools Scripting Help

Schemas: customers, SQL File 4*, SQL File 5*, SQL File 6*, products, SQL File 7*

Query 1: customers

```

1 • USE sql_store;
2 -- phone number end with 9
3 • SELECT *
4 FROM customers
5 WHERE phone LIKE '%9';

```

Result Grid:

#	customer_id	first_name	last_name	birth_date	phone	address	city	state	points
1	3	Freddi	Boagey	1985-02-07	719-724-7869	251 Springs Junction	Colorado Springs	CO	2967
2	7	Ilene	Dowson	1964-08-30	615-641-4759	50 Lillian Crossing	Nashville	TN	1672
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

customers 9

Query Completed

The screenshot shows the MySQL Workbench interface. In the top navigation bar, 'Local Instance' is selected. Below it, the 'Schemas' tab is active, showing a tree view of databases: sql_hr, sql_inventory, sql_invoicing, and sql_store. Under sql_store, there are tables: customers, order_item_notes, order_items, orders, order_statuses, and products. The 'products' table is currently selected. In the main pane, a query editor window titled 'Query 1' contains the following SQL code:

```

1 • USE sql_store;
2 -- phone number not end with 9
3 • SELECT *
4 FROM customers
5 WHERE phone NOT LIKE '%9';

```

Below the query editor is a results grid titled 'Result Grid'. It displays the following data from the 'products' table:

#	customer_id	first_name	last_name	birth_date	phone	address	city	state	points
1	1	Barbara	MacCaffrey	1986-03-28	781-932-9754	0 Sage Terrace	Waltham	VA	2273
2	2	Ines	Brushfield	1986-04-13	804-427-9456	14187 Commercial Trail	Hampton	VA	947
3	4	Ambur	Roseburgh	1974-04-14	407-231-8017	30 Arapahoe Terrace	Orlando	FL	457
4	6	Elka	Twiddell	1991-09-04	312-480-8498	7 Manley Drive	Chicago	IL	3073
5	8	Thacher	Naseby	1993-07-17	941-527-3977	538 Mosinee Center	Sarasota	FL	205
6	9	Romola	Rumgay	1992-05-23	559-181-3744	3520 Ohio Trail	Visalia	CA	1486
7	10	Levy	Mynett	1969-10-13	404-246-3370	68 Lawn Avenue	Atlanta	GA	796

At the bottom of the results grid, there are buttons for 'Apply' and 'Revert'.

The REGEXP operator:

It is a short for regular expression, and regular expressions are extremely powerful when it comes to searching for strings.

-- regexp for string/char anywhere in lastname

SELECT *

FROM customers

WHERE last_name REGEXP 'field';

MySQL Workbench

Local Instance

File Edit View Query Database Server Tools Scripting Help

Schemas: customers SQL File 4* SQL File 5* SQL File 6* products SQL File 7*

Limit to 1000 rows

```

1 • USE sql_store;
2 |
3 • SELECT *
4   FROM customers
5   WHERE last_name REGEXP 'field$';

```

Result Grid: Filter Rows: Edit: Export/Import: Wrap Cell Content:

#	customer_id	first_name	last_name	birth_date	phone	address	city	state	points
1	2	Ines	Brushfield	1986-04-13	804-427-9456	14187 Commercial Trail	Hampton	VA	947
*	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL

customers 13

Query Completed

-- regexp for string/char in begining of lastname

SELECT *

FROM customers

WHERE last_name REGEXP '^brush';

MySQL Workbench

Local Instance

File Edit View Query Database Server Tools Scripting Help

Schemas: customers SQL File 4* SQL File 5* SQL File 6* products SQL File 7*

Limit to 1000 rows

```

1 • USE sql_store;
2 -- ^ carrot operator
3 • SELECT *
4   FROM customers
5   WHERE last_name REGEXP '^brush';

```

Result Grid: Filter Rows: Edit: Export/Import: Wrap Cell Content:

#	customer_id	first_name	last_name	birth_date	phone	address	city	state	points
1	2	Ines	Brushfield	1986-04-13	804-427-9456	14187 Commercial Trail	Hampton	VA	947
*	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL

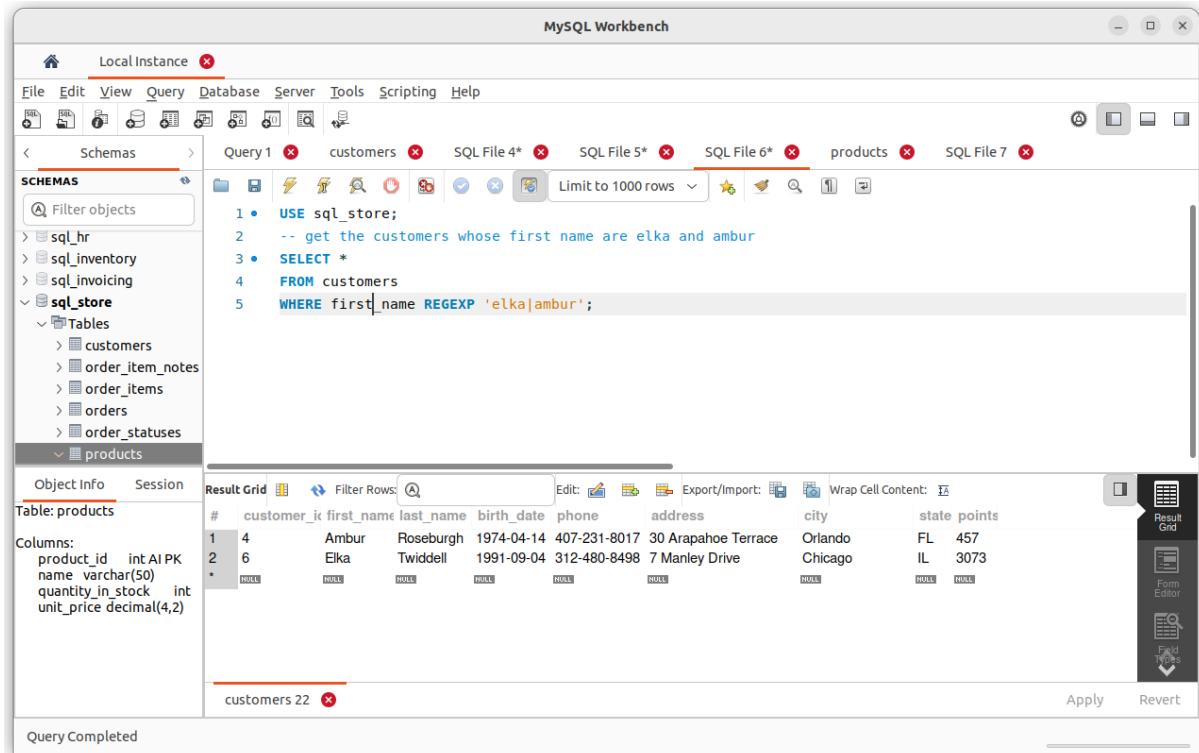
customers 15

Query Completed

-- regexp for multiple string/char anywhere in lastname

SELECT *

FROM customers
WHERE last_name REGEXP 'field|mac';



The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** Local Instance (selected)
- Query Editor:** Query 1 (customers) containing the following SQL code:

```

1 • USE sql_store;
2 -- get the customers whose first name are elka and ambur
3 • SELECT *
4 FROM customers
5 WHERE first_name REGEXP 'elka|ambur';
    
```
- Result Grid:** Shows the results of the query for the 'products' table.
- Table Definition:** Shows the columns and data types for the 'products' table:

```

Columns:
product_id int AI PK
name varchar(50)
quantity_in_stock int
unit_price decimal(4,2)
    
```
- Grid Headers:** customer_id, first_name, last_name, birth_date, phone, address, city, state, points
- Grid Data:**

#	customer_id	first_name	last_name	birth_date	phone	address	city	state	points
1	4	Ambur	Roseburgh	1974-04-14	407-231-8017	30 Arapahoe Terrace	Orlando	FL	457
2	6	Elka	Twiddell	1991-09-04	312-480-8498	7 Manley Drive	Chicago	IL	3073
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
- Buttons:** Result Grid, Filter Rows, Edit, Export/Import, Wrap Cell Content, Apply, Revert

IS NULL Syntax

```

SELECT column_names
FROM table_name
WHERE column_name IS NULL;
    
```

MySQL Workbench

Local Instance

Schemas: sql_hr, sql_inventory, sql_invoicing, sql_store

Tables: customers, order_item_notes, order_items, orders, order_statuses, products

Query 1: customers

```

1 • USE sql_store;
2 -- null operator
3 • SELECT *
4 FROM customers
5 WHERE phone IS NULL;
    
```

Result Grid:

#	customer_id	first_name	last_name	birth_date	phone	address	city	state	points
1	5	Clemmie	Betchley	1973-11-07	NULL	5 Spohn Circle	Arlington	TX	3675
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Object Info: products

Table: products

Columns:

```

product_id int AI PK
name varchar(50)
quantity_in_stock int
unit_price decimal(4,2)
    
```

Query Completed

3- ORDER BY

-- order by to sort data in SQL queries

SELECT *

FROM customers

ORDER BY first_name;

MySQL Workbench

Local Instance

Schemas: sql_hr, sql_inventory, sql_invoicing, sql_store

Tables: customers, order_item_notes, order_items, orders, order_statuses, products

Query 1: customers

```

1 • USE sql_store;
2 -- null operator
3 • SELECT *
4 FROM customers
5 WHERE phone IS NULL;
    
```

Result Grid:

#	customer_id	first_name	last_name	birth_date	phone	address	city	state	points
1	5	Clemmie	Betchley	1973-11-07	NULL	5 Spohn Circle	Arlington	TX	3675
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Object Info: products

Table: products

Columns:

```

product_id int AI PK
name varchar(50)
quantity_in_stock int
unit_price decimal(4,2)
    
```

Query Completed

-- if you want to sort data in descending order use desc

```

SELECT *
FROM customers
ORDER BY first_name DESC;

```

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** Local Instance, Schemas selected.
- Query 1:** customers
- SQL File 6***: Contains the following SQL code:

```

1 • USE sql_store;
2 -- if you want to sort data in descending order use desc
3 • SELECT *
4 FROM customers
5 ORDER BY first_name DESC;
6

```
- Result Grid:** Displays 10 rows of customer data. The columns are: #, customer_id, first_name, last_name, birth_date, phone, address, city, state, points. The data is as follows:

#	customer_id	first_name	last_name	birth_date	phone	address	city	state	points
1	8	Thacher	Naseby	1993-07-17	941-527-3977	538 Mosinee Center	Sarasota	FL	205
2	9	Romola	Rumgay	1992-05-23	559-181-3744	3520 Ohio Trail	Visalia	CA	1486
3	10	Levy	Mynett	1969-10-13	404-246-3370	68 Lawn Avenue	Atlanta	GA	796
4	2	Ines	Brushfield	1986-04-13	804-427-9456	14187 Commercial Trail	Hampton	VA	947
5	7	Ilene	Dowson	1964-08-30	615-641-4759	50 Lillian Crossing	Nashville	TN	1672
6	3	Freddi	Boagey	1985-02-07	719-724-7869	251 Springs Junction	Colorado Springs	CO	2967
7	6	Elka	Twiddell	1991-09-04	312-480-8498	7 Manley Drive	Chicago	IL	3073
8	5	Clemmie	Betchley	1973-11-07	NULL	5 Spohn Circle	Arlington	TX	3675
9	1	Barbara	MacCaffrey	1986-03-28	781-932-9754	0 Sage Terrace	Waltham	VA	2273
10	4	Amber	Roseburgh	1974-04-14	407-231-8017	30 Arapahoe Terrace	Orlando	FL	457
- Customer Count:** 25
- Buttons:** Apply, Revert
- Status:** Query Completed

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** Local Instance, Schemas selected.
- Query 1:** customers
- SQL File 6***: Contains the following SQL code:

```

1 • USE sql_store;
2 -- ordering data in descending order in new column
3 • SELECT *,quantity * unit_price AS total_price
4 FROM order_items
5 WHERE order_id=2
6 ORDER BY quantity * unit_price DESC;
7

```
- Result Grid:** Displays 3 rows of data from the order_items table. The columns are: #, order_id, product_id, quantity, unit_price, total_price. The data is as follows:

#	order_id	product_id	quantity	unit_price	total_price
1	2	1	2	9.10	18.20
2	2	4	4	1.66	6.64
3	2	6	2	2.94	5.88
- Result Count:** 26
- Buttons:** Read Only
- Status:** Query Completed

– sort the data by a

The screenshot shows the MySQL Workbench interface with a query editor and results grid. The query is:

```

1 • USE sql_store;
2 -- ordering data in descending order in new column
3 • SELECT *,quantity * unit_price AS total_price
4 FROM order_items
5 WHERE order_id=2
6 ORDER BY total_price DESC;
7

```

The results grid displays the following data:

#	order_id	product_id	quantity	unit_price	total_price
1	2	1	2	9.10	18.20
2	2	4	4	1.66	6.64
3	2	6	2	2.94	5.88

Object Info and Session tabs are visible on the left. A sidebar on the right includes icons for Result Grid, Form Editor, Field Types, and Query Stats.

LIMIT CLAUSE-

```
-- get top 3 data
SELECT *
FROM customers
LIMIT 3;
```

Sql commands 35

```
-- get 3 data from between the data using offset
SELECT *
FROM customers
LIMIT 6, 3;
```

MySQL Workbench

Local Instance

File Edit View Query Database Server Tools Scripting Help

Schemas

SCHEMAS

sql_hr
sql_inventory
sql_invoicing
sql_store

Tables

customers
order_item_notes
order_items
orders
order_statuses
products

Object Info Session

Table: customers

Columns:

```
customer_id int AI PK
first_name varchar(50)
last_name varchar(50)
birth_date date
phone varchar(50)
address varchar(50)
city varchar(50)
state char(2)
```

Query 1 customers SQL File 4* SQL File 5* SQL File 6* products SQL File 7*

Limit to 1000 rows

1 • USE sql_store;
2 -- get top 5 data
3 • SELECT *
4 FROM customers
5 LIMIT 5;

Result Grid

#	customer_id	first_name	last_name	birth_date	phone	address	city	state	points
1	1	Babara	MacCaffrey	1986-03-28	781-932-9754	0 Sage Terrace	Waltham	VA	2273
2	2	Ines	Brushfield	1986-04-13	804-427-9456	14187 Commercial Trail	Hampton	VA	947
3	3	Freddi	Boagey	1985-02-07	719-724-7869	251 Springs Junction	Colorado Springs	CO	2967
4	4	Ambur	Roseburgh	1974-04-14	407-231-8017	30 Arapahoe Terrace	Orlando	FL	457
5	5	Clemmie	Betchley	1973-11-07	NULL	5 Spohn Circle	Arlington	TX	3675

customers 28

Apply Revert

Query Completed

MySQL Workbench

Local Instance

File Edit View Query Database Server Tools Scripting Help

Schemas

SCHEMAS

sql_hr
sql_inventory
sql_invoicing
sql_store

Tables

customers
order_item_notes
order_items
orders
order_statuses
products

Object Info Session

Table: customers

Columns:

```
customer_id int AI PK
first_name varchar(50)
last_name varchar(50)
birth_date date
phone varchar(50)
address varchar(50)
city varchar(50)
state char(2)
```

Query 1 customers SQL File 4* SQL File 5* SQL File 6* products SQL File 7*

Limit to 1000 rows

3 • SELECT *
4 FROM customers
5 LIMIT 5;
-- get 7-9 data
7 • SELECT *
8 FROM customers
9 LIMIT 6,3;

Result Grid

#	customer_id	first_name	last_name	birth_date	phone	address	city	state	points
1	7	Ilene	Dowson	1964-08-30	615-641-4759	50 Lillian Crossing	Nashville	TN	1672
2	8	Thacher	Naseby	1993-07-17	941-527-3977	538 Mosinee Center	Sarasota	FL	205
3	9	Romola	Rumgay	1992-05-23	559-181-3744	3520 Ohio Trail	Visalia	CA	1486

customers 29

Apply Revert

Query Completed

INNER JOIN:

-- inner join to join 2 tables

```
SELECT *  
FROM orders o  
JOIN customers c  
ON o.customer_id=c.customer_id;
```

MySQL Workbench screenshot showing the execution of the inner join query.

The SQL Editor pane contains the following code:

```
1 • USE sql_store;  
2 • SELECT *  
3 • FROM orders  
4 • JOIN customers  
5 • ON orders.customer_id=customers.customer_id;  
6  
7  
8
```

The Result Grid pane displays the joined data from the orders and customers tables. The columns are: #, order_id, customer_id, order_date, status, comments, shipped_date, shipper_id, customer_id, first_name, last_name, birth_date, phone. The data is as follows:

#	order_id	customer_id	order_date	status	comments	shipped_date	shipper_id	customer_id	first_name	last_name	birth_date	phone
1	1	6	2019-01-30	1	NULL	NULL	NULL	6	Elka	Twidell	1991-09-04	312-4
2	2	7	2018-08-02	2	NULL	2018-08-03	4	7	Ilene	Dowson	1964-08-30	615-6
3	3	8	2017-12-01	1	NULL	NULL	NULL	8	Thacher	Naseby	1993-07-17	941-5
4	4	2	2017-01-22	1	NULL	NULL	NULL	2	Ines	Brushfield	1986-04-13	804-4

Query Completed

MySQL Workbench screenshot showing the execution of the query with additional joins.

The SQL Editor pane contains the following code:

```
1 • USE sql_store;  
2 • SELECT *  
3 • FROM orders  
4 • JOIN customers  
5 • ON orders.customer_id=customers.customer_id;  
6 • SELECT order_id,orders.customer_id,first_name,last_name  
7 • FROM orders  
8 • JOIN customers  
9 • ON orders.customer_id=customers.customer_id;  
10  
11  
12
```

The Result Grid pane displays the data from the orders and customers tables. The columns are: #, order_id, customer_id, first_name, last_name. The data is as follows:

#	order_id	customer_id	first_name	last_name
1	4	2	Ines	Brushfield
2	7	2	Ines	Brushfield
3	5	5	Clemmie	Betchley
4	8	5	Clemmie	Betchley

Query Completed

```

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

```

The screenshot shows the MySQL Workbench interface with a query editor and a results grid.

Query Editor:

```

USE sql_store;
SELECT *
FROM orders o
JOIN customers c
ON o.customer_id=c.customer_id;

```

Results Grid:

#	order_id	first_name	last_name
1	4	Ines	Brushfield
2	7	Ines	Brushfield
3	5	Clemmie	Betchley
4	8	Clemmie	Betchley

Result 4 Read Only

```

SELECT *
FROM orders o, customers c
WHERE o.customer_id = c.customer_id;
-- joining across databases
SELECT *
FROM order_items oi
JOIN sql_inventory.products p
ON oi.product_id = p.product_id;

```

MySQL Workbench

Local Instance

Schemas SQL File 5* OUTER_JOIN products sql_SELF_OUTER_JOIN* SQL_SELECT SQL File 9* customers

```

10
11  SELECT o.order_id, c.first_name, c.last_name
12  FROM orders o
13  JOIN customers c
14  ON o.customer_id=c.customer_id;
15
16 •  SELECT *
17  FROM order_items oi
18  JOIN sql_inventory.products p
19  ON oi.product_id = p.product_id;
20

```

Result Grid Filter Rows: Export: Wrap Cell Content:

#	order_id	product_id	quantity	unit_price	product_id	name	quantity_in_stoc	unit_price
1	1	4	4	3.74	4	Brocolinni - Gaylan, Chinese	90	4.53
2	2	1	2	9.10	1	Foam Dinner Plate	70	1.21
3	2	4	4	1.66	4	Brocolinni - Gaylan, Chinese	90	4.53
4	2	6	2	2.94	6	Petit Baguette	14	2.39
5	3	3	10	9.12	3	Lettuce - Romaine, Heart	38	3.35
6	4	3	7	6.99	3	Lettuce - Romaine, Heart	38	3.35
7	4	10	7	6.40	10	Broom - Push	6	1.09
8	5	2	3	9.89	2	Pork - Bacon,back Peameal	49	4.65

Result 5

Query Completed

-- self join

USE sql_hr;

SELECT

e.employee_id,

e.first_name,

m.first_name

FROM employees e

JOIN employees m

ON e.reports_to = m.employee_id;

MySQL Workbench

Local Instance

Schemas

Object Info Session

No object selected

SQL File 5* OUTER_JOIN products sql_SELF_OUTER_JOIN* SQL_SELECT SQL File 9* customers

19 ON oi.product_id = p.product_id;
20 • USE sql_hr;
21 • SELECT
22 e.employee_id,
23 e.first_name,
24 m.first_name
25 FROM employees e
26 JOIN employees m
27 ON e.reports_to = m.employee_id;
28
29

Result Grid Filter Rows Export: Wrap Cell Content: Result 6

#	employee_id	first_name	first_name
1	33391	D'arcy	Yovonnda
2	37851	Sayer	Yovonnda
3	40448	Mindy	Yovonnda
4	56274	Keriann	Yovonnda
5	63196	Alaster	Yovonnda
6	67009	North	Yovonnda
7	67370	Elladine	Yovonnda
8	68249	Nisse	Yovonnda

Result 6

Query Completed

Result Grid Form Editor Field Inspector

Read Only

-- joining multiple tables

```
USE sql_store;
SELECT
o.order_id,
o.order_date,
c.first_name,
c.last_name,
FROM orders o
JOIN customers c
ON o.customer_id = c.customer_id
JOIN order_statuses os
ON o.status = os.order_status_id;
```

MySQL Workbench

Local Instance

File Edit View Query Database Server Tools Scripting Help

Schemas SQL File 5* OUTER_JOIN products sql_SELF_OUTER_JOIN* SQL_SELECT SQL File 9* customers

Filter objects

Stored Procedures Functions sal inventory

No object selected

```

28 • USE sql_store;
29 • SELECT
30   o.order_id,
31   o.order_date,
32   c.first_name,
33   c.last_name,
34   os.name AS status
35   FROM orders o
36   JOIN customers c
37   ON o.customer_id = c.customer_id
38   JOIN order_statuses os

```

Result Grid Filter Rows Export: Wrap Cell Content:

#	order_id	order_date	first_name	last_name	status
1	8	2018-06-08	Clemmie	Betchley	Processed
2	6	2018-11-18	Levy	Mynett	Processed
3	4	2017-01-22	Ines	Brushfield	Processed
4	3	2017-12-01	Thacher	Naseby	Processed
5	1	2019-01-30	Elka	Twiddell	Processed
6	10	2018-04-22	Elka	Twiddell	Shipped
7	9	2017-07-05	Levy	Mynett	Shipped
8	7	2018-09-22	Ines	Brushfield	Shipped

Result 7 Read Only

Query Completed

OUTER JOIN=

There are two types of outer join

1- LEFT JOIN-

-- outer joins for left join

```
SELECT c.customer_id,
c.first_name,
o.order_id
```

FROM customers c

LEFT JOIN orders o

```
ON c.customer_id= o.customer_id
ORDER BY c.customer_id;
```

MySQL Workbench

Local Instance

Schemas: customers, SQL File 4*, SQL File 5*, SQL File 6*, products, SQL File 7, SQL_SELECT

Query 1: USE sql_store;
-- outer joins
SELECT c.customer_id,
c.first_name,
o.order_id
FROM customers c
LEFT JOIN orders o
ON c.customer_id= o.customer_id;
ORDER BY c.customer_id;

Result Grid:

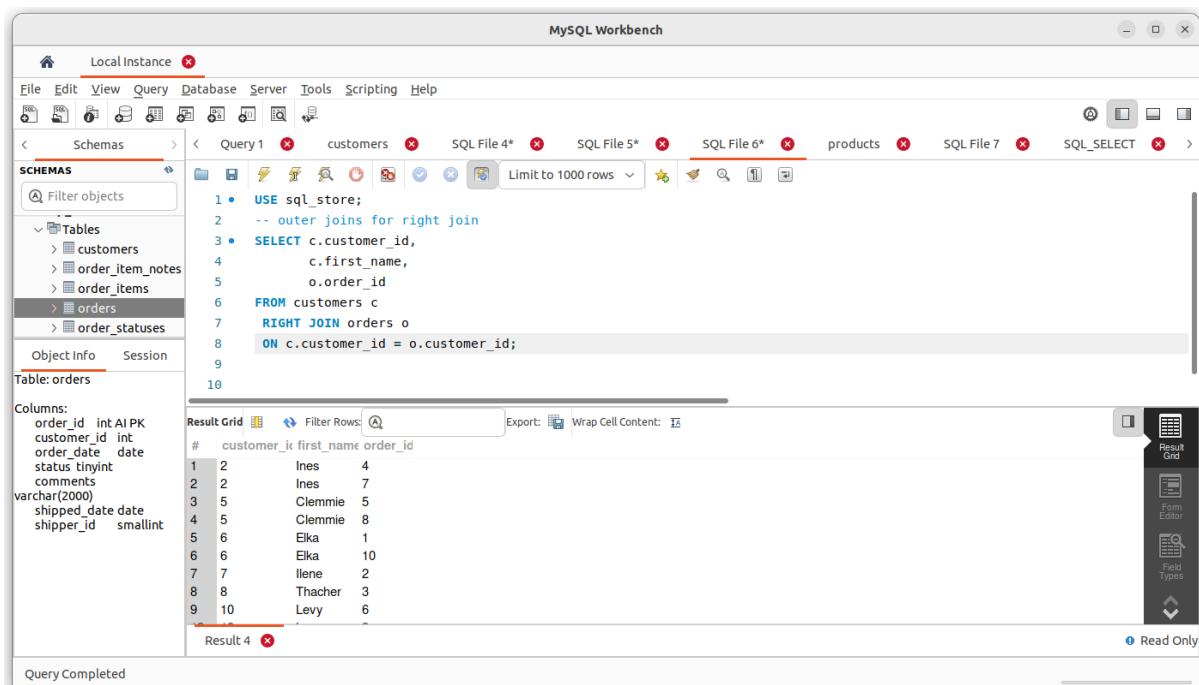
#	customer_id	first_name	order_id
1	1	Barbara	NULL
2	2	Ines	4
3	2	Ines	7
4	3	Freddi	NULL
5	4	Ambur	NULL
6	5	Clemmie	5
7	5	Clemmie	8
8	6	Elka	1
9	6	Elka	10

Query Completed

2-RIGHT JOIN-

-- outer joins for right join
SELECT c.customer_id,
c.first_name,
o.order_id
FROM customers c
RIGHT JOIN orders o

ON c.customer_id = o.customer_id;



The screenshot shows the MySQL Workbench interface with a query editor containing the following SQL code:

```
1 • USE sql_store;
2 • -- outer joins for right join
3 • SELECT c.customer_id,
4 •         c.first_name,
5 •         o.order_id
6 • FROM customers c
7 • RIGHT JOIN orders o
8 • ON c.customer_id = o.customer_id;
9
10
```

The results grid displays the following data:

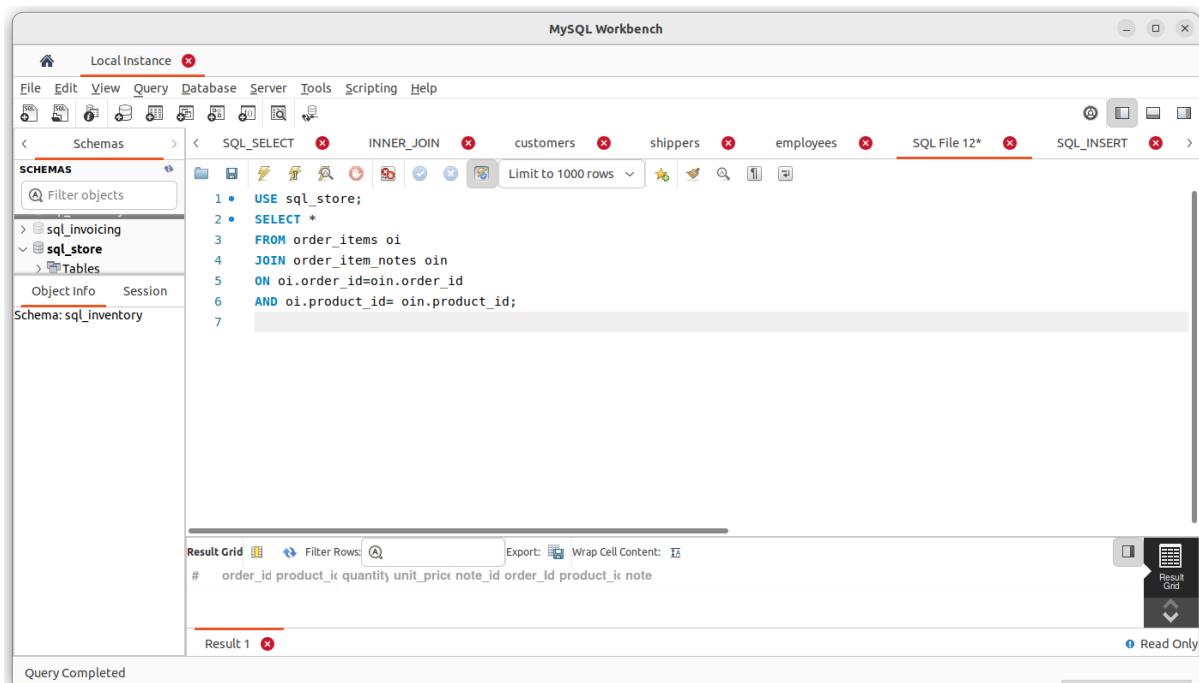
#	customer_id	first_name	order_id
1	2	Ines	4
2	2	Ines	7
3	5	Clemmie	5
4	5	Clemmie	8
5	6	Elka	1
6	6	Elka	10
7	7	Ilene	2
8	8	Thacher	3
9	10	Levy	6

Query Completed

COMPOUND JOIN CONDITIONS-

SELECT *

```
FROM order_items oi
JOIN order_items_notes oin
ON oi.order_id=oin.order_id
AND oi.product_id= oin.product_id;
```



The screenshot shows the MySQL Workbench interface with a query editor containing the following SQL code:

```
1 • USE sql_store;
2 • SELECT *
3 • FROM order_items oi
4 • JOIN order_item_notes oin
5 • ON oi.order_id=oin.order_id
6 • AND oi.product_id= oin.product_id;
7
```

The results grid displays the following data:

#	order_id	product_id	quantity	unit_price	note_id	order_id	product_id	note
1	1	1	10	14.75	1	1	1	Customer comment: This product is great!

Query Completed

OUTER JOINS BETWEEN MULTIPLE TABLES-

-- outer join for multiple tables

SELECT

```
c.customer_id,  
c.first_name,  
o.order_id  
FROM customers c  
LEFT JOIN orders o  
ON c.customer_id = o.customer_id  
JOIN shippers sh  
ON o.shipper_id = sh.shipper_id  
ORDER BY c.customer_id
```

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** Local Instance, File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Schemas Panel:** Shows the schema tree with `sql_invoicing` and `sql_store` expanded, displaying Tables, Views, Stored Procedures, and Functions.
- Query Editor:** Contains the SQL code for an outer join between multiple tables. The code includes:
 - Line 1: `-- outer join for multiple tables`
 - Line 2: `SELECT`
 - Lines 3-5: `c.customer_id,`
`c.first_name,`
`o.order_id`
 - Line 6: `FROM customers c`
 - Line 7: `LEFT JOIN orders o`
 - Line 8: `ON c.customer_id = o.customer_id`
 - Line 9: `LEFT JOIN shippers sh`
 - Line 10: `ON o.shipper_id = sh.shipper_id`
 - Line 11: `ORDER BY c.customer_id`
- Result Grid:** Displays the results of the query. The columns are `#`, `customer_id`, `first_name`, and `order_id`. The data is as follows:

#	customer_id	first_name	order_id
1	1	Babara	NULL
2	2	Ines	4
3	2	Ines	7
4	3	Freddi	NULL
5	4	Ambur	NULL
6	5	Clemmie	5
- Status:** Query Completed.

SELF OUTER JOIN-

--self outer join

SELECT

```
e.employee_id,  
e.first_name,
```

```

m.first_name AS manager
FROM employees e
LEFT JOIN employees m
ON e.reports_to = m.employee_id

```

The screenshot shows the MySQL Workbench interface with a query editor and a results grid.

Query Editor:

```

1  --self outer join
2  SELECT
3    e.employee_id,
4    e.first_name,
5    m.first_name AS manager
6  FROM employees e
7  LEFT JOIN employees m
8  ON e.reports_to = m.employee_id

```

Results Grid:

#	employee_id	first_name	manager
1	33391	D'arcy	Yovonnda
2	37270	Yovonnda	NULL
3	37851	Sayer	Yovonnda
4	40448	Mindy	Yovonnda
5	56274	Kerianne	Yovonnda
6	63196	Alaster	Yovonnda

Result 1

Query Completed

THE USING CLAUSE-

the USING clause

SELECT

o.order_id,

c.first_name

FROM orders o

JOIN customers c

USING (customer_id)

JOIN shippers sh

USING (shipper_id)

MySQL Workbench

Local Instance

File Edit View Query Database Server Tools Scripting Help

Schemas SQL File 5* OUTER_JOIN products sql_SELF_OUTER_JOIN SQL_SELECT

SCHEMAS: sql_hr

Object Info Session

1 • USE sql_store;
2 -- the USING clause
3 • SELECT
4 o.order_id,
5 c.first_name
6 FROM orders o
7 JOIN customers c
8 USING (customer_id)
9 JOIN shippers sh
10 USING (shipper_id)

Result Grid Filter Rows Export: Wrap Cell Content: Result Grid Form Editor Field Types

#	order_id	first_name
1	9	Levy
2	10	Elka
3	5	Clemmie
4	2	Ilene
5	7	Ines

Result 2 Read Only

Query Completed

The screenshot shows the MySQL Workbench interface with a query editor window. The query is a multi-table join using the 'USING' clause. The results are displayed in a grid with columns for the row number, order ID, and customer first name. The data consists of five rows: (9, Levy), (10, Elka), (5, Clemmie), (2, Ilene), and (7, Ines). The interface includes a sidebar for schemas and objects, and a toolbar with various icons.

NATURAL JOIN-

-- natural join

SELECT o.order_id,

c.first_name

FROM orders o

NATURAL JOIN customers c

MySQL Workbench

Local Instance

File Edit View Query Database Server Tools Scripting Help

Schemas SQL File 5* OUTER_JOIN products sql_SELF_OUTER_JOIN SQL_SELECT

SCHEMAS: sql_hr

Object Info Session

Schema: sql_hr

Result Grid Filter Rows: Export: Wrap Cell Content:

order_id first_name

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

Result 3 Read Only

Query Completed

The screenshot shows the MySQL Workbench interface with a query editor window. The query being run is a NATURAL JOIN between the 'orders' and 'customers' tables. The results are displayed in a grid format, showing the 'order_id' and 'first_name' for each customer. The results are as follows:

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

CROSS JOIN-

```
SELECT o.order_id,  
c.first_name  
FROM orders o  
CROSS JOIN customers c  
ORDER BY o.order_id;
```

The screenshot shows the MySQL Workbench interface. In the top navigation bar, 'Query' is selected. The left sidebar shows the 'Schemas' tree with 'sql_store' selected. The main area contains a query editor with the following SQL code:

```

1 • USE sql_store;
2 -- natural join
3 • SELECT o.order_id,
4   c.first_name
5   FROM orders o
6   NATURAL JOIN customers c
7   -- cross join
8 • SELECT o.order_id,
9   c.first_name
10  FROM orders o
11  CROSS JOIN customers c
12  ORDER BY o.order_id;

```

The result grid below displays the output of the query:

#	order_id	first_name
11	2	Levy
12	2	Clemmie
13	2	Freddi
14	2	Ilene
15	2	Thacher
16	2	Ambur
17	2	Elka

Result 4

UNIONS-

The **UNION** operator is used to combine the result-set of two or more **SELECT** statements.

- Every **SELECT** statement within **UNION** must have the same number of columns
- The columns must also have similar data types
- The columns in every **SELECT** statement must also be in the same order

-- union

SELECT

customer_id,

first_name,

points,

'Bronze' AS TYPE

FROM customers

WHERE points < 2000

UNION

SELECT

customer_id,

first_name,

points,

'Silver' AS TYPE

FROM customers

WHERE points BETWEEN 2000 AND 3000

The screenshot shows the MySQL Workbench interface. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The left sidebar displays the 'Schemas' tree, with 'sql_hr' selected. The main area contains a SQL editor window with the following query:

```
1 -- union
2 • SELECT
3 customer_id,
4 first_name,
5 points,
6 'Bronze' AS TYPE
7 FROM customers
8 WHERE points < 2000
9 UNION
10 SELECT
11 customer_id,
12 first_name,
```

The SQL editor has tabs for 'OUTER_JOIN', 'products', 'sql_SELF_OUTER_JOIN*', and 'SQL_SELECT'. Below the editor is a 'Result Grid' tab. The results grid displays the following data:

#	customer_id	first_name	points	TYPE
2	4	Amund	457	Bronze
3	7	Ilene	1672	Bronze
4	8	Thacher	205	Bronze
5	9	Romola	1486	Bronze
6	10	Levy	796	Bronze
7	1	Babara	2273	Silver
8	3	Freddi	2967	Silver

The bottom status bar indicates 'Query Completed'.

INSERT INTO-The **INSERT INTO** statement is used to insert new records in a table.

-- insert a row into the table

INSERT INTO customers

Values

```
( default,  
'John',  
'Smith',  
'1990-01-01',  
NULL,  
'address',  
'city',  
'CA',  
DEFAULT)
```

-- insert multiple row in table

INSERT INTO shippers (name)

```
VALUES ('Shipper1'),  
       ('Shipper2'),  
       ('Shipper3');
```

The screenshot shows the MySQL Workbench interface. In the top navigation bar, 'Local Instance' is selected. The main window has three tabs: 'Schemas' (selected), 'Tables', 'Views', 'Stored Procedures', and 'Functions'. The 'Tables' tab is currently active, showing the 'customers', 'shippers', and 'employees' tables. Below the tabs, there's a toolbar with various icons. The central area contains the SQL code for inserting data into the 'customers' table. The bottom section, titled 'Action Output', displays the execution log with the following entries:

#	Time	Action	Message
59	16:20:24	SELECT o.order_id, c.first_name FROM orders o CROSS JOIN customers c ORDER BY o.order_id	100 row(s) returned
60	16:52:03	SELECT customer_id, first_name, points, 'Bronze' AS Tiers FROM customers	6 row(s) returned
61	16:56:11	SELECT customer_id, first_name, points, 'Bronze' AS Tiers FROM customers	8 row(s) returned
62	17:18:53	INSERT INTO customers Values (default, 'John', 'Smith', '1990-01-01', NULL, 'address', 'city', 'CA', DEFAULT)	1 row(s) affected
63	17:18:56	INSERT INTO customers Values (default, 'John', 'Smith', '1990-01-01', NULL, 'address', 'city', 'CA', DEFAULT)	1 row(s) affected

The status bar at the bottom left says 'Query Completed'.

MySQL Workbench

Local Instance

File Edit View Query Database Server Tools Scripting Help

Schemas: SQL File 9*, customers, shippers, employees, SQL File 12*, SQL File 13*

SCHEMAS

- > order_items
- > orders
- > order_statuses
- > products
- > shippers
- Views

Object Info Session Schema: sql_hr

```

5   'John',
6   'Smith',
7   '1990-01-01',
8   NULL,
9   'address',
10  'city',
11  'CA',
12  DEFAULT)
13  -- insert multiple row in table
14  INSERT INTO shippers (name)
15  VALUES ('Shipper1'),
16      ('Shipper2'),
17      ('Shipper3');

```

Action Output

#	Time	Action	Message
62	17:18:53	INSERT INTO customers values (default, John, Smith, ...)	1 row(s) affected
63	17:18:56	INSERT INTO customers Values (default, 'John', 'Smith', ...)	1 row(s) affected
64	17:27:29	NSERT INTO shippers (name) VALUES ('Shipper1'), ('S...	Error Code: 1064. You have an error in your SQL s... VALUES ('Shipper1'), ('Shipper2'), ('Sh' at line 1
65	17:27:39	INSERT INTO shippers (name) VALUES ('Shipper1'), ('S...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0

Query Completed

COPY TABLE-
CREATE TABLE orders_arvhived AS
SELECT *
FROM orders

MySQL Workbench

Local Instance

File Edit View Query Database Server Tools Scripting Help

Schemas: SQL SELECT, INNER_JOIN, customers, shippers, employees, SQL File 12*

SCHEMAS

- > sql_store
 - > Tables
 - > customers
 - > order_item_notes
 - > order_items
 - > orders
 - > orders_arvhived
 - > order_statuses
 - > products
 - > shippers
 - Views
 - Stored Procedures

Object Info Session Table: shippers

Columns:

shipper_id	smallint AI
name	varchar(50)

PK name varchar(50)

```

1 • USE sql_store;
2 • CREATE TABLE orders_arvhived AS
3   SELECT *
4   FROM orders
5
6

```

Action Output

#	Time	Action	Message	Duration / Fetch
8	09:04:26	USE sql_store	0 row(s) affected	0.00033 sec
9	09:04:26	CREATE TABLE orders_arvhived AS SELECT * FROM ord...	Error Code: 1050. Table 'orders_arvhived' already exists	0.0010 sec
10	09:05:25	USE sql_store	0 row(s) affected	0.00041 sec
11	09:05:25	CREATE TABLE orders_arvhived AS SELECT * FROM orders	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0	0.023 sec

Query Completed

UPDATE-

-- updating a single row

UPDATE invoices

SET

payment_total = invoice_total * 0.5,

payment_date=due_date

WHERE invoice_id= 1;

-- updating multiple rows

UPDATE invoices

SET

payment_total=invoice_total*0.5, payment_date=due_date

WHERE client_id=3;

-- using subqueries in updates

UPDATE invoices

SET

payment_total=invoice_total*0.5, payment_date=due_date

WHERE client_id=

(SELECT client_id

FROM clients

WHERE name='Myworks'

);

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'Schemas' tree, with 'sql_invoicing' selected. The main area is the 'SQL' tab of the query editor, containing the provided SQL code. The bottom pane shows the 'Action Output' results for the executed queries.

#	Time	Action	Message	Duration / Fetch
15	10:12:41	UPDATE invoices SET payment_total = invoice_total * 0.5, payment_date = due_date WHERE invoice_id = 1;	Rows matched: 1 Changed: 1 Warnings: 1 0 row(s) affected, 1 warning(s) 1265 Data truncated for column 'payment_total' ... Rows matched: 1 Changed: 0 Warnings: 1	0.00045 sec
16	10:13:05	SELECT * FROM sql_invoicing.invoices LIMIT 0, 1000	17 row(s) returned	0.00046 sec / 0.000...

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Schemas Administration

Local Instance (sql_invoicing) Local Instance (sql_invoicing) Local Instance (sql_invoicing)

shippers employees SQL_update* SQL_INSERT* SQL File 16* invoices sql_invoicing.invoices

Filter objects

clients invoices

Columns:

- invoice_id int PK
- number varchar(50)
- client_id int
- invoice_total decimal(9,2)
- payment_total decimal(9,2)
- invoice_date date

Action Output

#	Time	Action	Message	Duration / Fetch
36	11:55:20	UPDATE invoices SET payment_total=invoice_total*0.1 ...	1265 Data truncated for column 'payment_total' ... 1265 Data truncated for column 'payment_total' ... 1265 Data truncated for column 'payment_total' ... Rows matched: 5 Changed: 0 Warnings: 5	0.00044 sec
37	11:55:34	SELECT * FROM sql_invoicing.invoices LIMIT 0, 1000	16 row(s) returned	0.00035 sec / 0.000...

Query Completed

```

1 • USE sql_invoicing;
2
3 • UPDATE invoices
4   SET
5     payment_total = invoice_total * 0.5,
6     payment_date=due_date
7   WHERE invoice_id= 1;
8   -- updating multiple rows
9 • UPDATE invoices
10  SET
11    payment_total=invoice_total*0.1
12  WHERE client_id=3;
13  -- using subqueries in updates
14 • UPDATE invoices
15  SET

```

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Schemas

Local Instance (sql_invoicing) Local Instance (sql_invoicing) Local Instance (sql_invoicing)

shippers employees SQL File 12*

sql_SELF_OUTER_JOIN* INNER_JOIN customers

Filter objects

sql_hr

Tables Views Stored Procedures Functions

sql_inventory

sql_invoicing

Tables clients invoices payment_methods payments

Action Output

#	Time	Action	Message	Duration / Fetch
26	11:36:41	USE sql_invoicing	0 row(s) affected	0.00034 sec
27	11:36:44	USE sql_invoicing	0 row(s) affected	0.00025 sec
28	11:36:44	DELETE FROM invoices WHERE client_id= (SELECT client_...)	1 row(s) affected	0.0063 sec
29	11:42:59	UPDATE invoices SET payment_total=invoice_total * 0.5...	0 row(s) affected Rows matched: 0 Changed: 0 Warnings: 0	0.00058 sec

Query Completed

```

1 • USE sql_invoicing;
2
3 • UPDATE invoices
4   SET
5     payment_total = invoice_total * 0.5,
6     payment_date=due_date
7   WHERE invoice_id= 1;
8

```

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Schemas Local Instance (sql_invoicing) Local Instance (sql_invoicing) Local Instance (sql_invoicing)

SQL_SELECT INNER_JOIN customers shippers employees SQL File 12*

SCHEMAS

- sql_hr
 - Tables
 - Views
 - Stored Procedures
 - Functions
- sql_inventory
- sql_invoicing**
 - Tables
 - clients
 - invoices
 - payment_methods
 - payments

Object Info Session Schema: sql_invoicing

```

9 • UPDATE invoices
10   SET
11     payment_total=invoice_total*0.5, payment_date=due_date
12   WHERE client_id=3;
13   -- using subqueries in updates
14 • UPDATE invoices
15   SET
16     payment_total=invoice_total*0.5, payment_date=due_date
17   WHERE client_id=
18   (
19     SELECT client_id
20     FROM clients
21     WHERE name='Myworks'
22   );
23

```

Action Output

#	Time	Action	Message	Duration / Fetch
28	11:36:44	DELETE FROM invoices WHERE client_id= (SELECT client_...	1 row(s) affected 0 rows(s) affected Rows matched: 0 Changed: 0 Warnings: 0	0.0063 sec
29	11:42:59	UPDATE invoices SET payment_total = invoice_total * 0.5, p...	0 row(s) affected Rows matched: 0 Changed: 0 Warnings: 0	0.00058 sec
30	11:44:59	UPDATE invoices SET payment_total=invoice_total*0.5, p...	0 row(s) affected Rows matched: 0 Changed: 0 Warnings: 0	0.00053 sec

Query Completed

DELETE-

```

USE sql_invoicing;
-- deleting rows
DELETE FROM invoices
WHERE client_id=
(SELECT client_id
FROM clients
WHERE name='Myworks'
);
```

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Schemas

SQL_SELF_OUTER_JOIN* SQL_SELECT INNER_JOIN customers shippers employees SQL File 12*

Limit to 1000 rows

Object Info Session Schema: sql_invoicing

Action Output

#	Time	Action	Message	Duration / Fetch
25	11:19:34	UPDATE customers SET points = points + 50 WHERE birth...	Error Code: 1175. You are using safe update mode... To disable safe mode, toggle the option in Prefer...	0.00053 sec
26	11:36:41	USE sql_invoicing	0 row(s) affected	0.00034 sec
27	11:36:44	USE sql_invoicing	0 row(s) affected	0.00025 sec
28	11:36:44	DELETE FROM invoices WHERE client_id= (SELECT client_...	1 row(s) affected	0.0063 sec

Query Completed

```
1 • USE sql_invoicing;
2 -- deleting rows
3 • DELETE FROM invoices
4 WHERE client_id=
5   (SELECT client_id
6   FROM clients
7 WHERE name='Myworks'
8 );
9
```