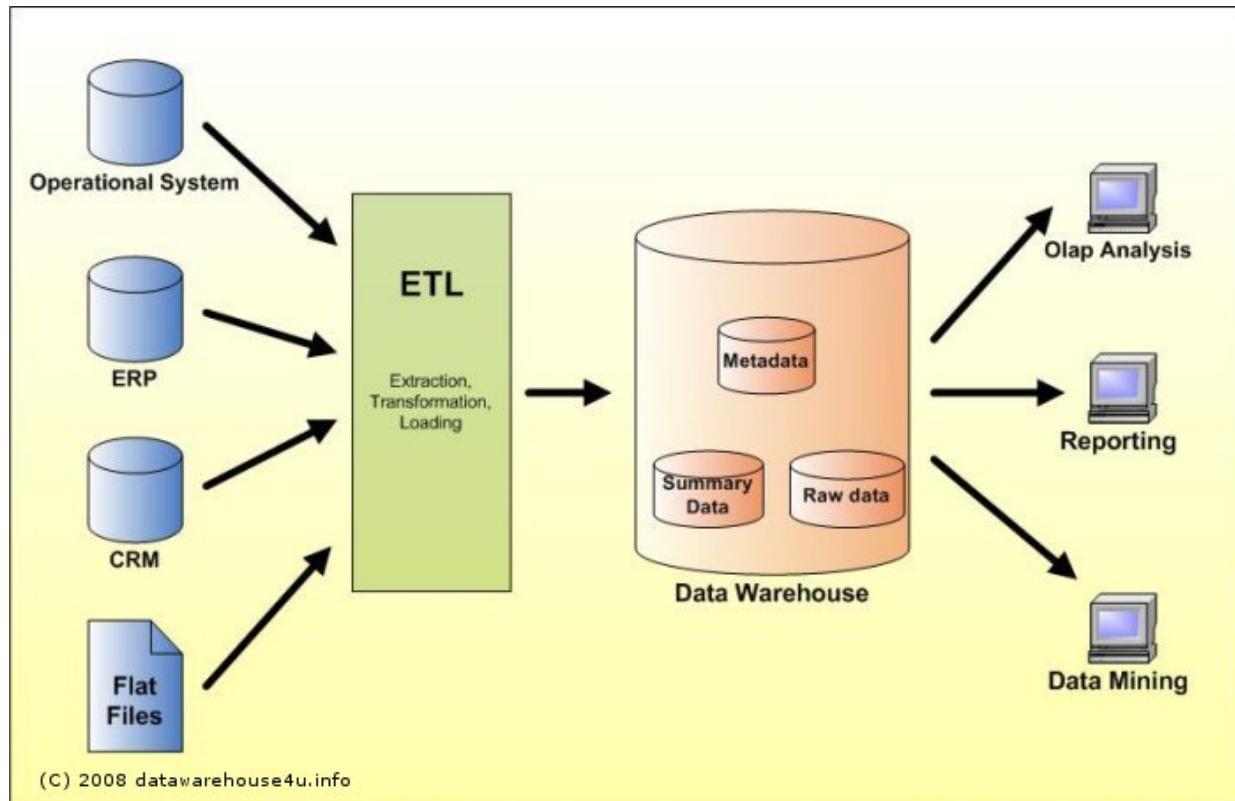


Data Warehousing

Practical Task # 01 Hands-on MySQL

Over View of Data Warehouse:

Data warehouse is an information system that stores historical and commutative data from single or multiple sources.



1. Installation of MySQL

MySQL Installation Guide

Useful Concepts:

Setup are available on Datum network for Vista & Win.7, Win 10. Oracle for MAC OS is not available.

Datum Access:

- 1) Press Win+R
- 2) Enter \\172.16.100.104
- 3) Username: niusb\i181234
- 4) Password: Your Wifi's password

OR

Download it from Internet

Step 1: Go to the link given below

<https://dev.mysql.com/downloads/>

④ MySQL Community Downloads

- MySQL Yum Repository
- MySQL APT Repository
- MySQL SUSE Repository
- MySQL Community Server
- MySQL Cluster
- MySQL Router
- MySQL Shell
- MySQL Workbench
- MySQL Installer for Windows
- MySQL for Excel
- MySQL for Visual Studio
- MySQL Notifier
- C API (libmysqlclient)
- Connector/C++
- Connector/J
- Connector/.NET
- Connector/Node.js
- Connector/ODBC
- Connector/Python
- MySQL Native Driver for PHP
- MySQL Benchmark Tool
- Time zone description tables
- Download Archives

Step 2: Install MYSQL Community Server then MYSQL Workbench.

Installation of MYSQL Community Server

- Click on the MYSQL Community Server from the above list OR Go to the directly from the link below:

<https://dev.mysql.com/downloads/mysql/>

a window will appear as follows:

The screenshot shows the MySQL Community Server 8.0.19 download page for Microsoft Windows. At the top, it says "MySQL Community Server 8.0.19". Below that, there's a dropdown menu for "Select Operating System" set to "Microsoft Windows". To the right, a button says "Looking for previous GA versions?". Under "Recommended Download:", there's a large image of the MySQL Installer for Windows icon, which features a blue square with a white dolphin and several smaller squares with icons. Below the icon, text reads "MySQL Installer for Windows" and "All MySQL Products. For All Windows Platforms. In One Package.". A note at the bottom left says "Starting with MySQL 5.6 the MySQL Installer package replaces the standalone MSI packages." At the bottom, there are two download links: "Windows (x86, 32 & 64-bit), MySQL Installer MSI" and "Windows (x86, 64-bit), ZIP Archive". The "ZIP Archive" link is highlighted with a blue arrow pointing to it. Below these are "Other Downloads:" sections for ZIP and MSI archives.

Download Type	Version	File Size	Action
Windows (x86, 64-bit), ZIP Archive (mysql-8.0.19-winx64.zip)	8.0.19	187.8M	Download
Windows (x86, 64-bit), ZIP Archive	8.0.19	406.7M	Download

- Click on “**Go to Download Page**” new page will appear as follows:

The screenshot shows the MySQL Installer 8.0.19 download page. At the top, it says "General Availability (GA) Releases". Below that, the title "MySQL Installer 8.0.19" is displayed. A dropdown menu "Select Operating System:" shows "Microsoft Windows". To the right, a link "Looking for previous GA versions?" is visible. The main content area lists two download options:

Version	File Type	Size	Action
8.0.19	Windows (x86, 32-bit), MSI Installer (mysql-installer-web-community-8.0.19.0.msi)	18.6M	Download
8.0.19	Windows (x86, 32-bit), MSI Installer (mysql-installer-community-8.0.19.0.msi)	398.9M	Download

Below the download links, there is a note: "We suggest that you use the [MD5 checksums](#) and [GnuPG signatures](#) to verify the integrity of the packages you download."

- Click on *mysql-installer-community-8.0.19.0.msi* then click on “*no thanks, just start my download.*”

④ MySQL Community Downloads

[Login Now](#) or [Sign Up](#) for a free account.

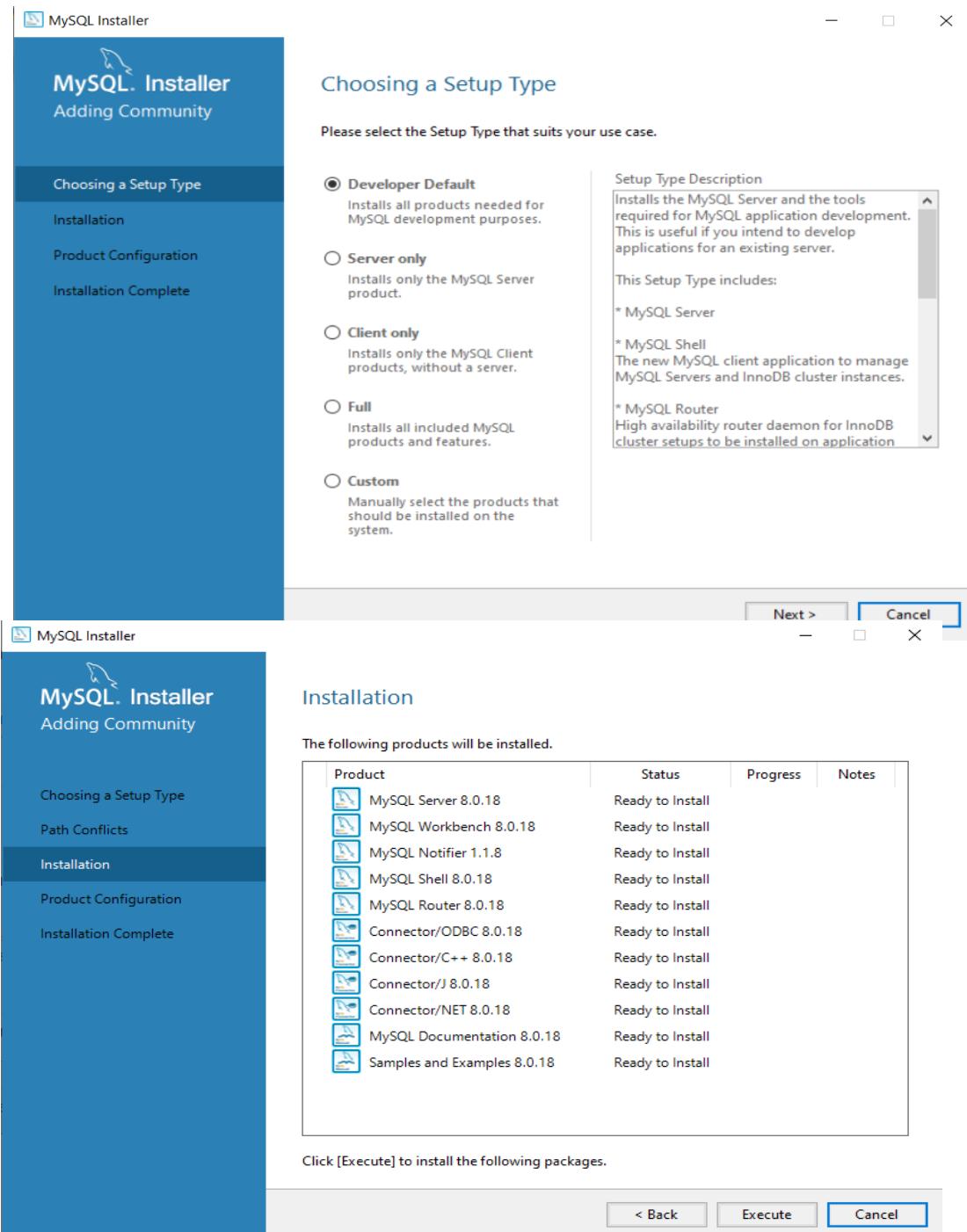
An Oracle Web Account provides you with the following advantages:

- Fast access to MySQL software downloads
- Download technical White Papers and Presentations
- Post messages in the MySQL Discussion Forums
- Report and track bugs in the MySQL bug system

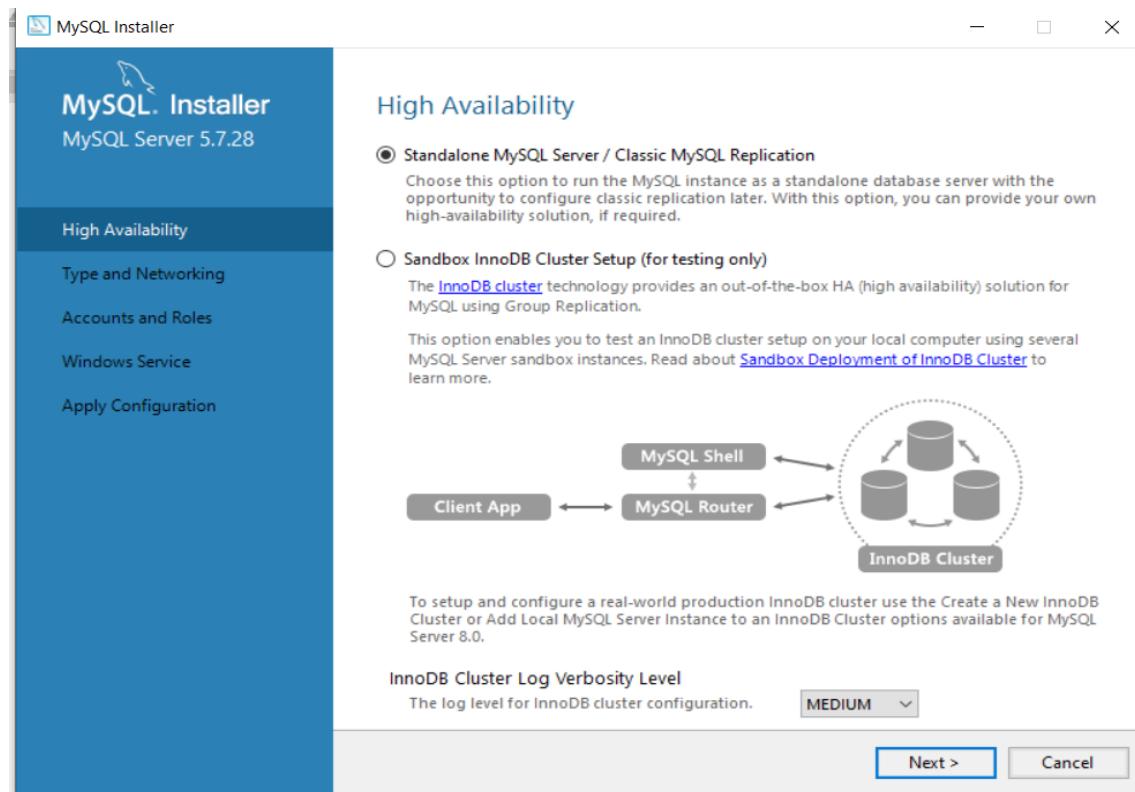
The screenshot shows the "Login Now" and "Sign Up" buttons for Oracle Web Account. The "Login" button is blue and says "Login » using my Oracle Web account". The "Sign Up" button is green and says "Sign Up » for an Oracle Web account". Below these buttons, a note states: "MySQL.com is using Oracle SSO for authentication. If you already have an Oracle Web account, click the Login link. Otherwise, you can signup for a free account by clicking the Sign Up link and following the instructions."

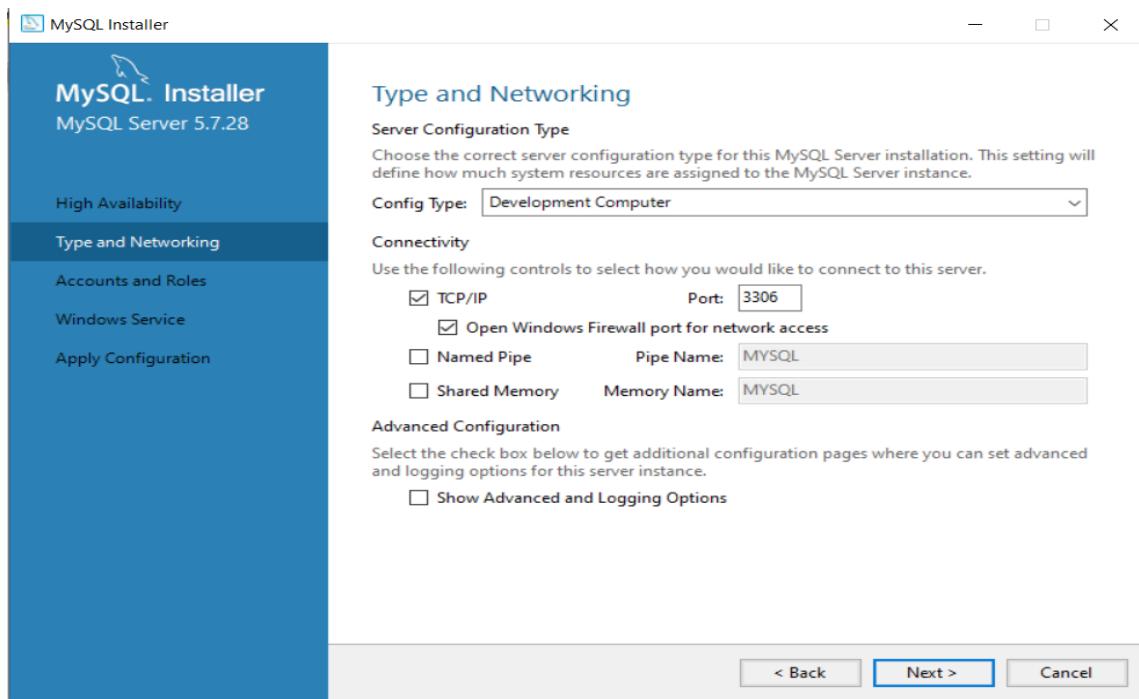
[No thanks, just start my download.](#)

- Download will be start. Double Click on the downloaded file in order to start the installation.
- Run the setup. The following window will appear and click on next.

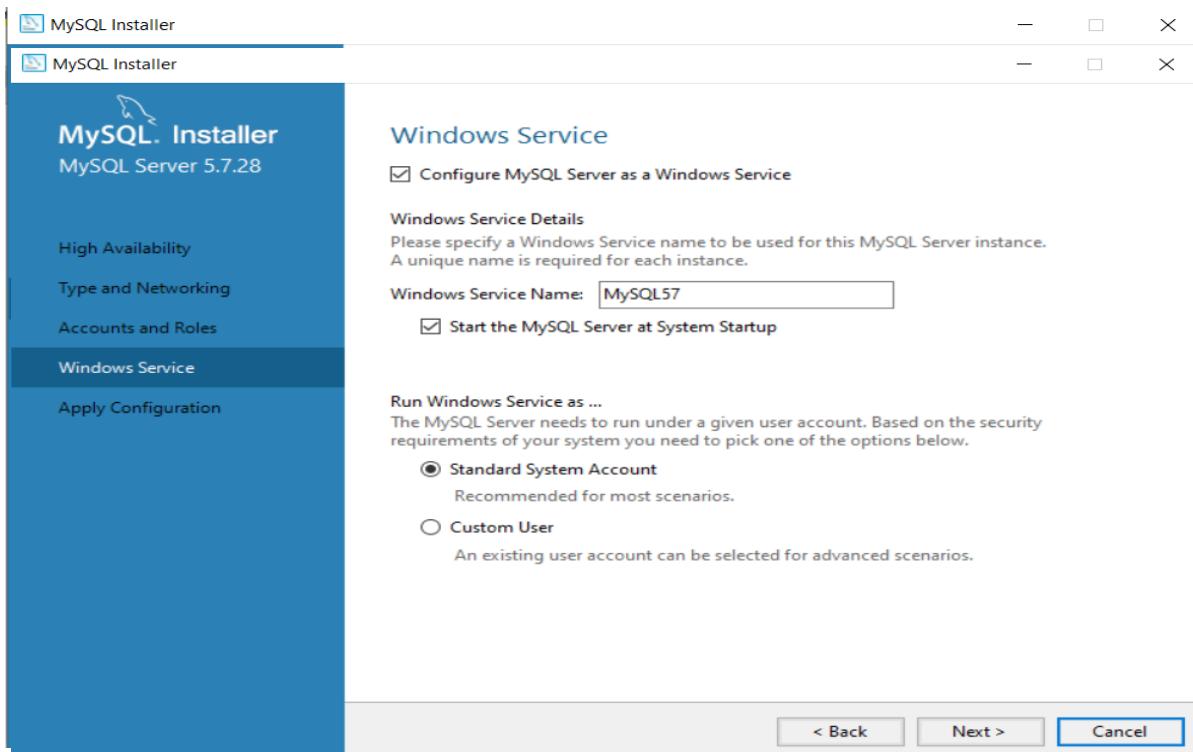


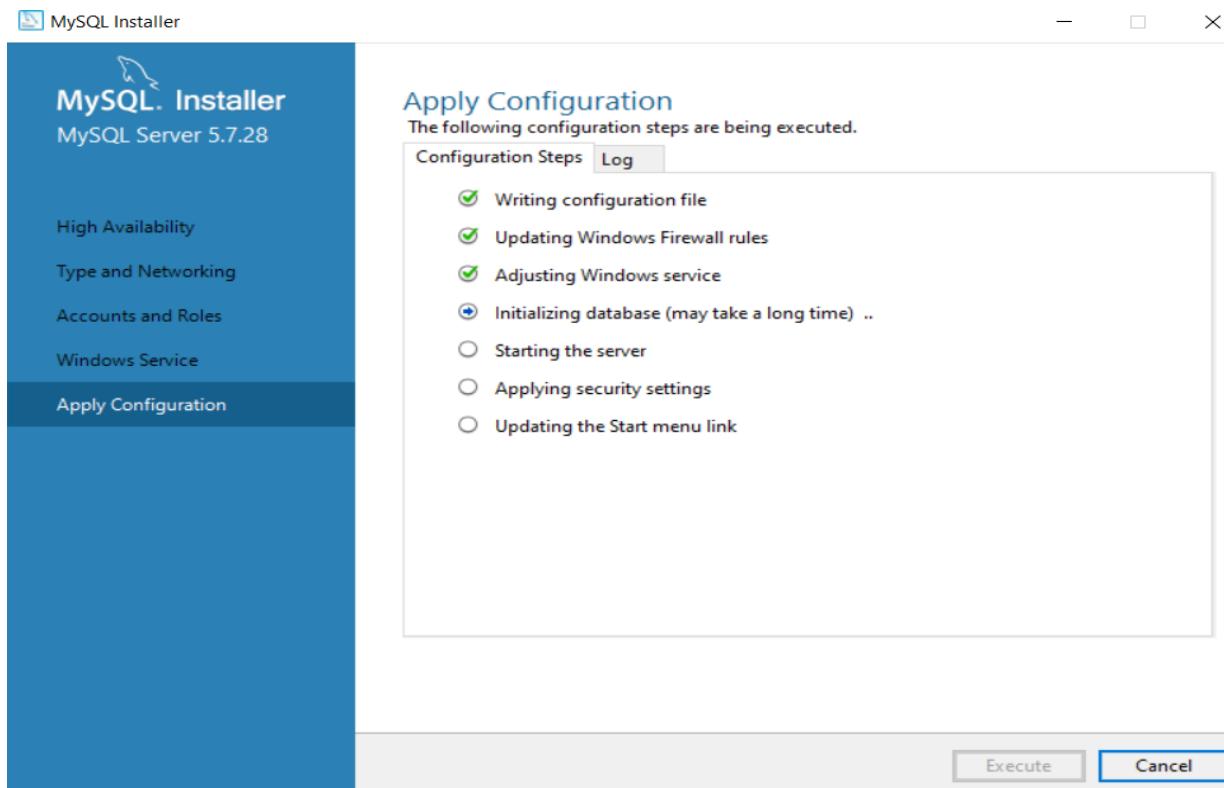
- Click on next.





- Enter a **strong** password and click next.





- Click Next

2nd Method to download and execute MYSQL Community server

1. Download MySQL Community Server ZIP ARCHIVE from
<https://dev.mysql.com/downloads/mysql/>

MySQL Community Server 8.0.19

Select Operating System:

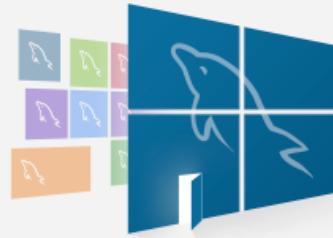
Microsoft Windows

Looking for previous GA versions?

Recommended Download:

MySQL Installer for Windows

All MySQL Products. For All Windows Platforms.
In One Package.



Starting with MySQL 5.6 the MySQL Installer package replaces the standalone MSI packages.

Windows (x86, 32 & 64-bit), MySQL Installer MSI

[Go to Download Page >](#)

Other Downloads:

→ [Windows \(x86, 64-bit\), ZIP Archive](#)

8.0.19

187.8M

[Download](#)

(mysql-8.0.19-winx64.zip)

MD5: f52c52e7b499958acc5f08ce0a869cab | [Signature](#)

[Windows \(x86, 64-bit\), ZIP Archive](#)

8.0.19

406.7M

[Download](#)

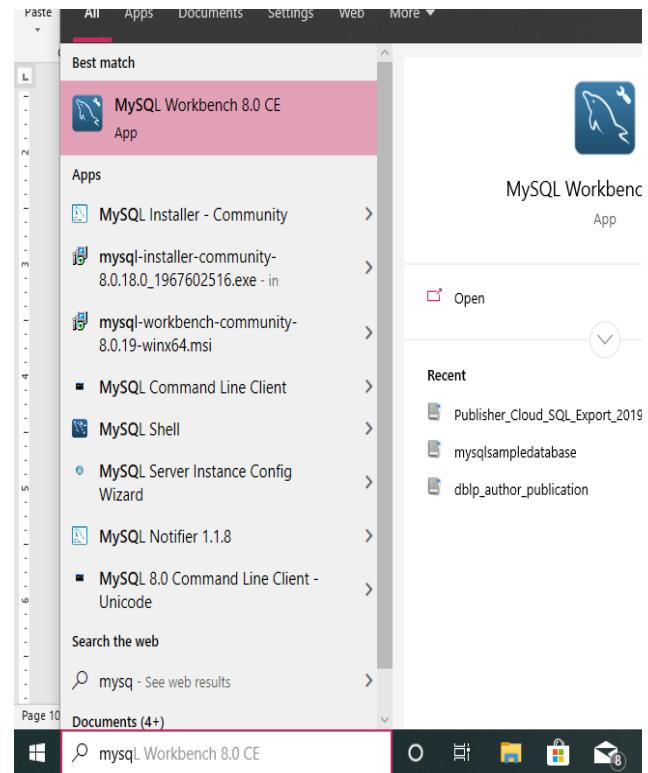
- Under "General Available (GA) Releases" tab.
 - Under "MySQL Community Server 8.0.{xx}", where {xx} is the latest update number
⇒ In "Select Operating System", choose "Microsoft Windows".
 - Under "Other Downloads", download "**Windows (x86, 64-bit), ZIP ARCHIVE (mysql-8.0.{xx}-winx64.zip)**".
 - Under "Begin your Download", there is NO need to "Login" or "Sign up" - Just click "**No thanks, just start my downloads!**"
2. UNZIP the downloaded file into your project directory "C:\DataWarehouse". MySQL will be unzipped as "c:\DataWarehouse\mysql". Take note and remember your MySQL installed directory!!!

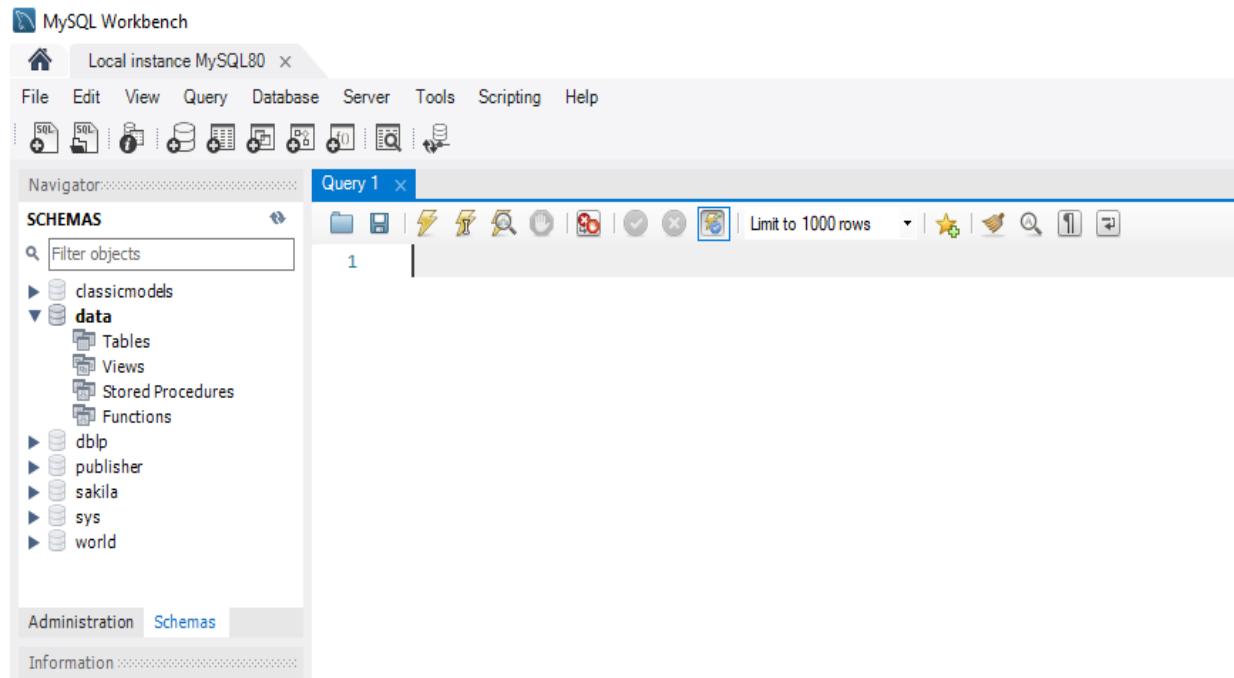
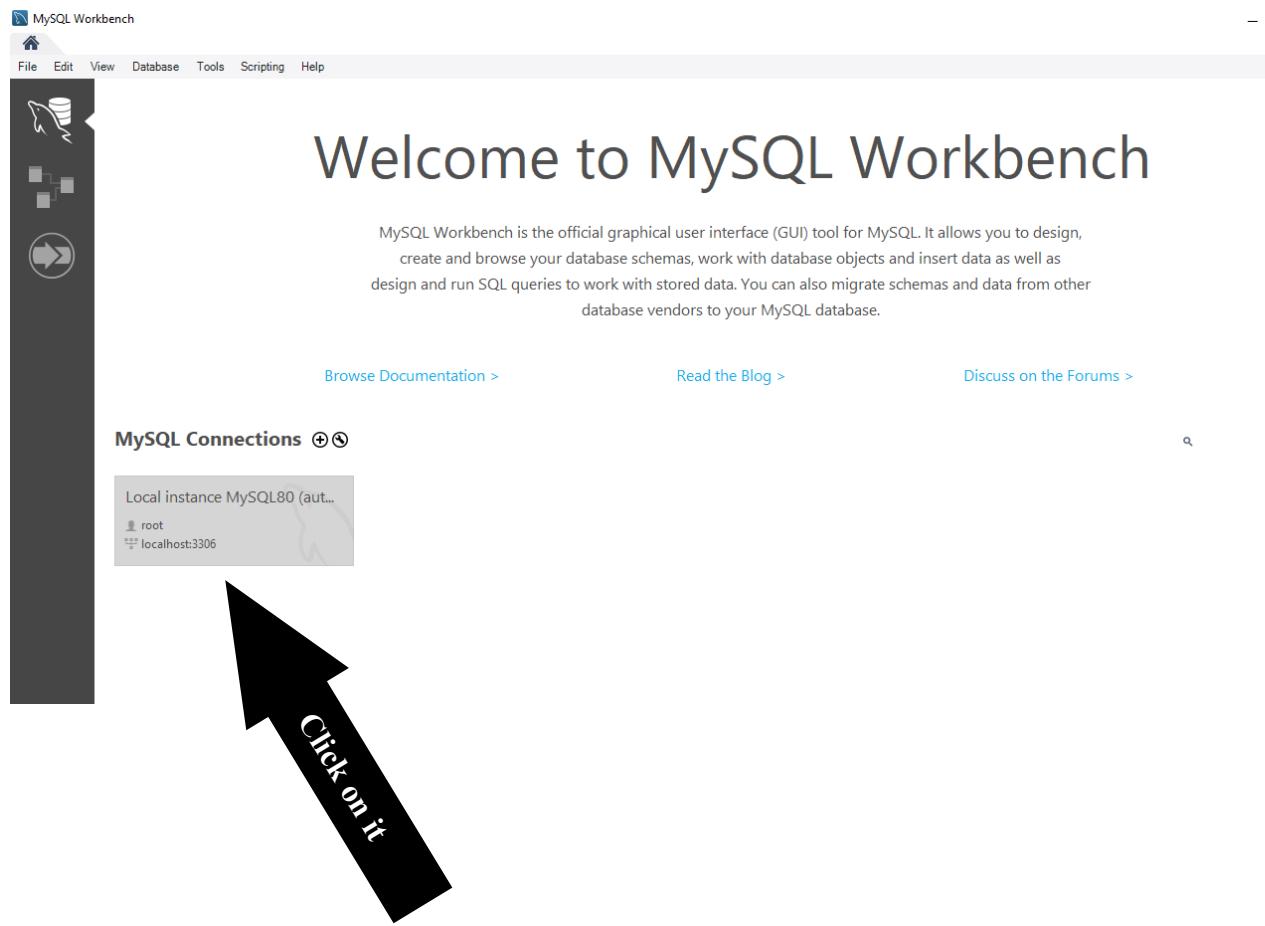
3. Start a CMD (as administrator) ("Search" button ⇒ Enter "cmd" ⇒ Right-Click on "Command Prompt" ⇒ Run as Administrator) and issue these commands.

```
// Change directory to the MySQL's binary directory  
// Suppose that your MySQL is installed in " c:\DataWarehouse\mysql "  
c:  
cd \DataWarehouse\mysql
```

4. During the installation, a superuser called root is created with a temporary password, as shown above. **TAKE NOTE of the PASSWORD, COPY and save it somewhere, and TAKE A PICTURE!!!!**
5. If you make a mistake or forgot your password, DELETE the entire MySQL directory "C:\DataWarehouse\mysql", and REPEAT step 2 and 3.

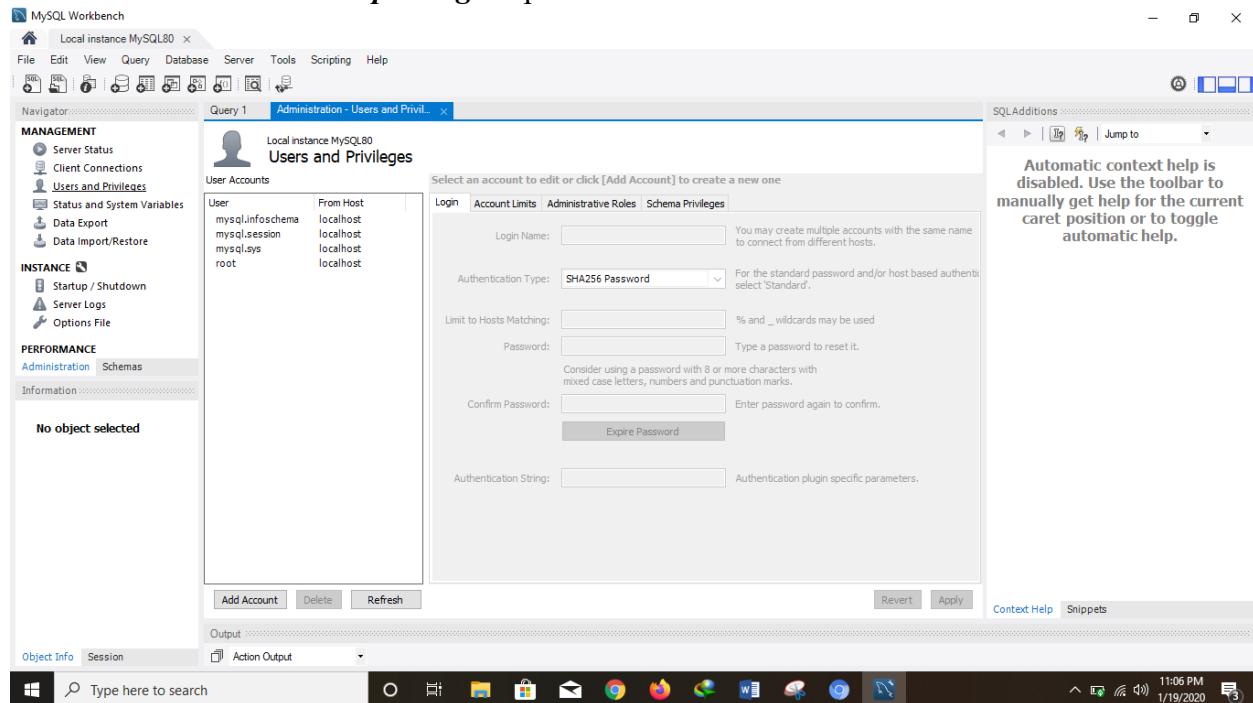
-
- After installation. Go to the Start>MySQL Workbench
 - Click to open it.



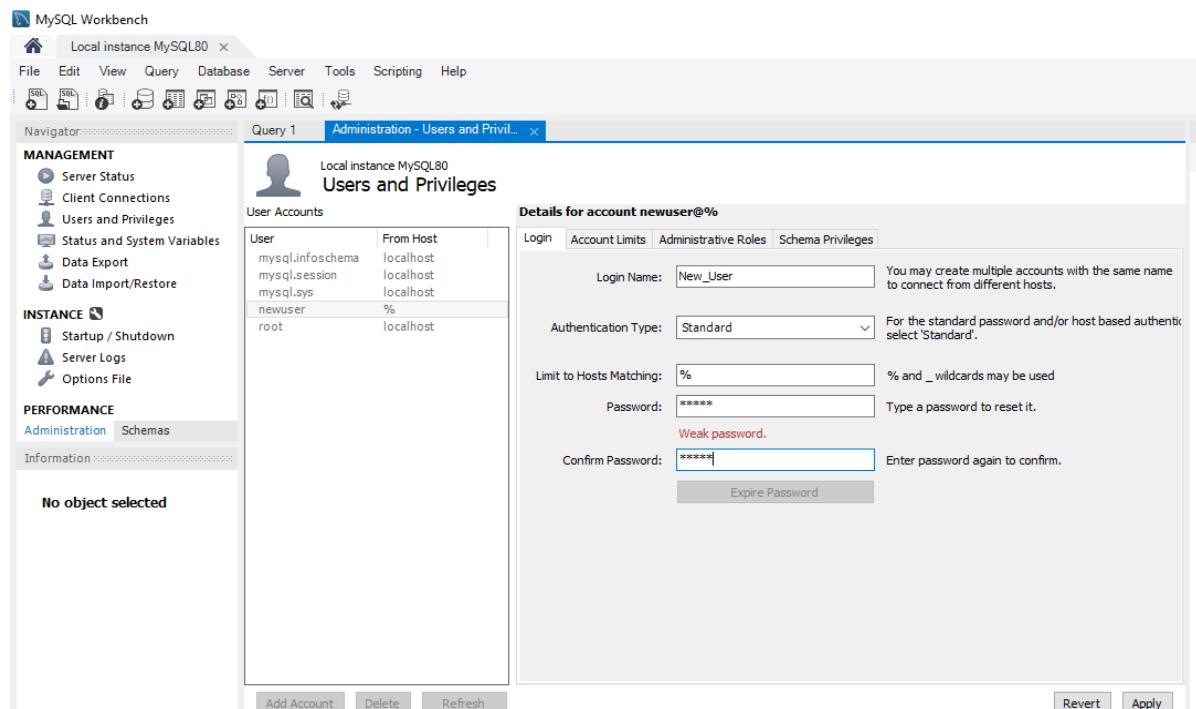


How to Add User:

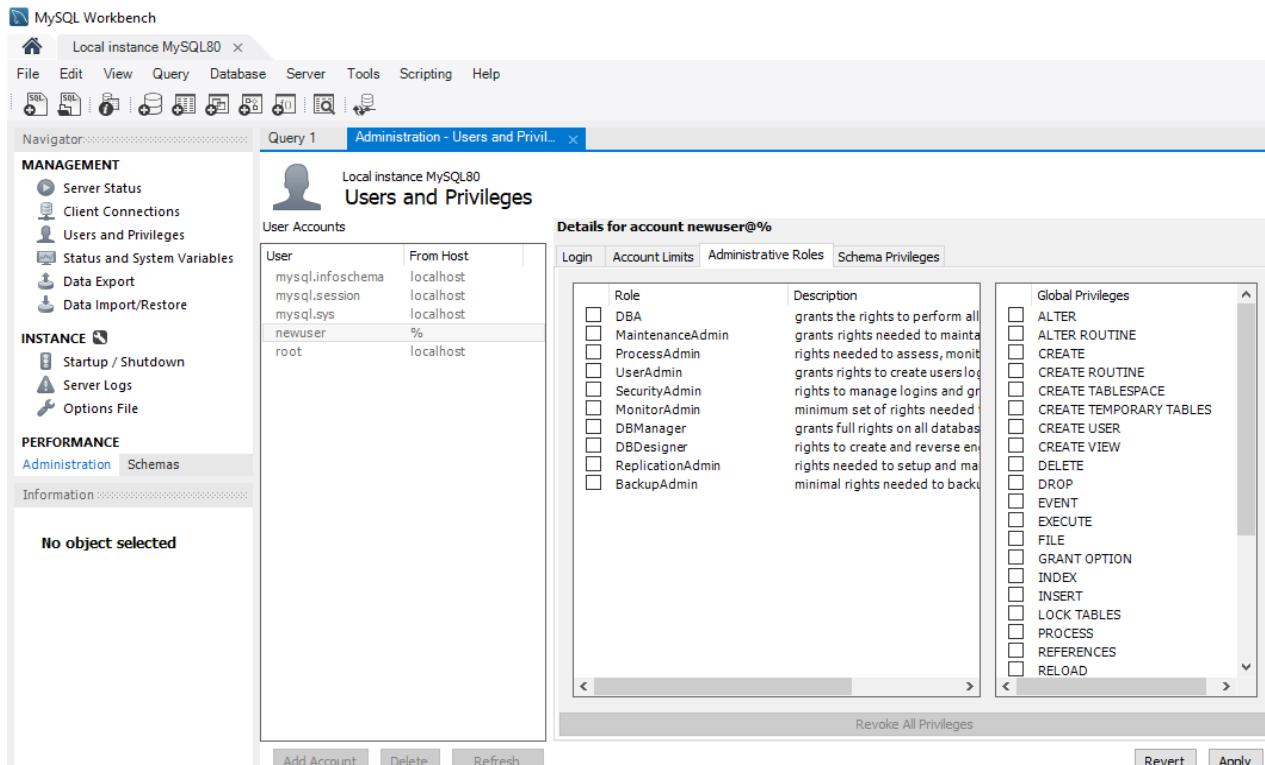
- Go to the **Users and privileges** option under MANAGEMENT



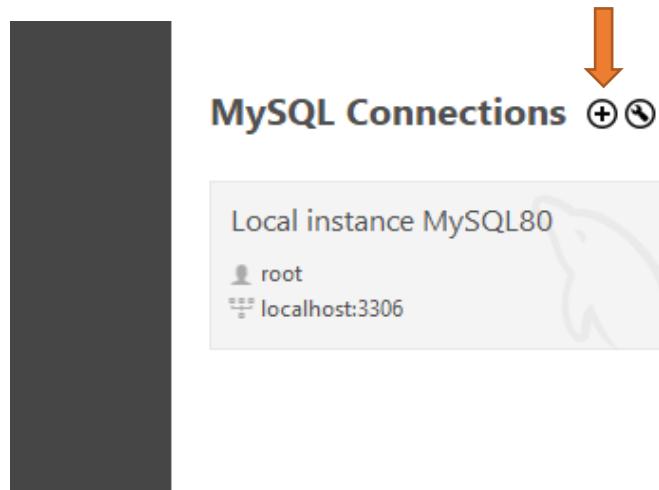
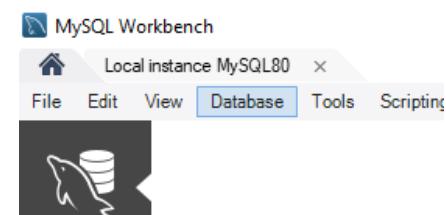
- Click on Add user. Enter User Name and password for the new connection



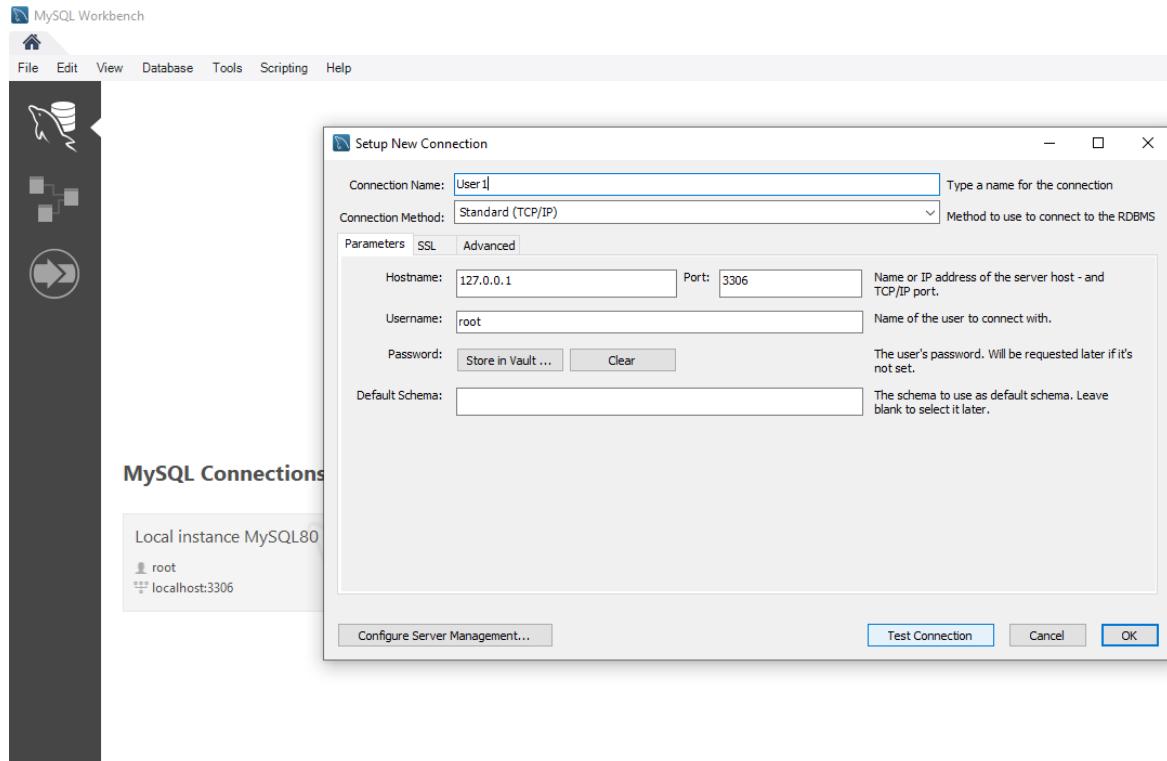
- Set privileges to the new user.



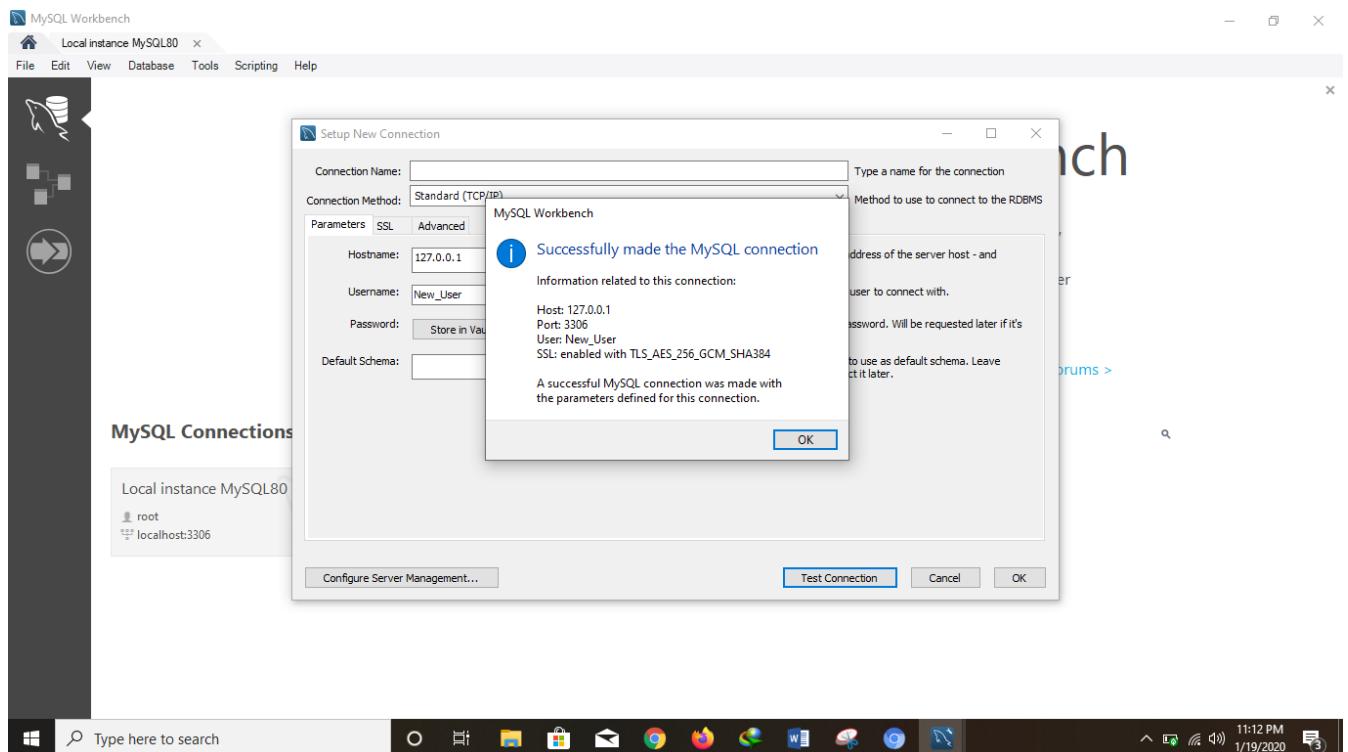
- Click “Apply”. For test the connection, Go to the home button on the top left corner
- Click on the + button on the existing connection.

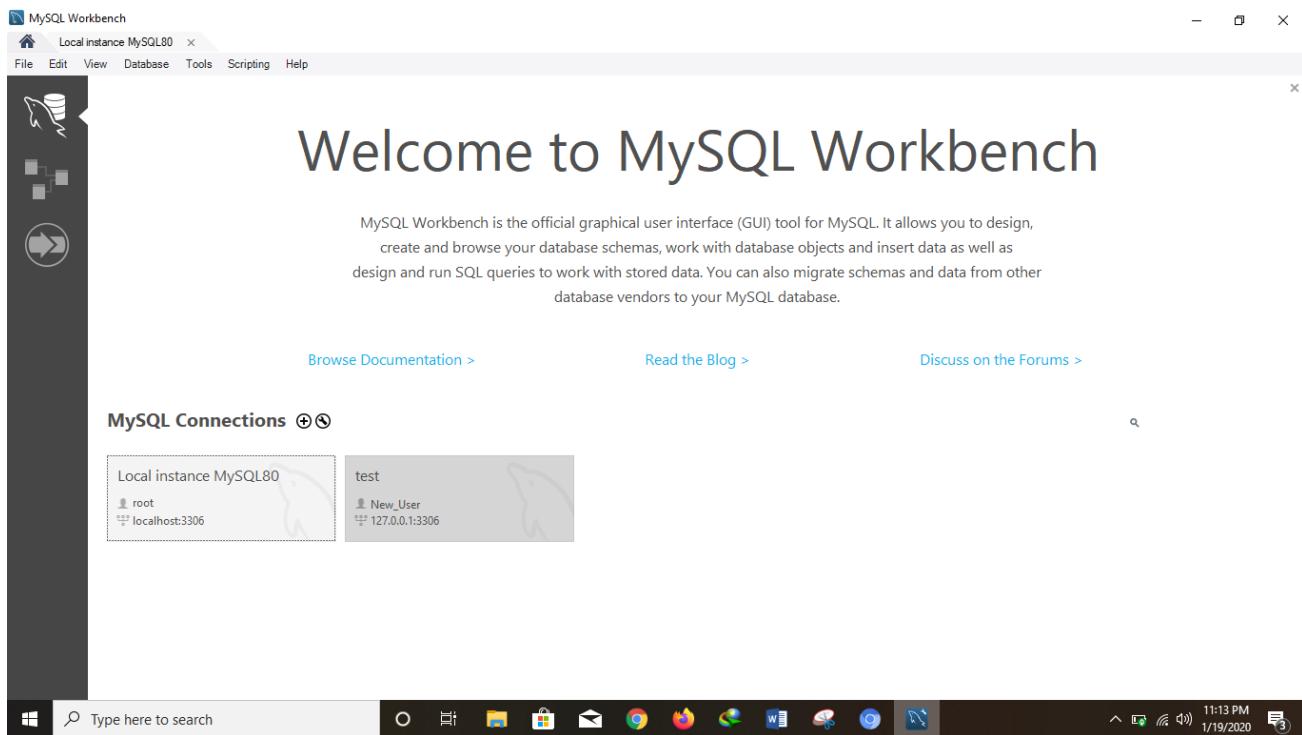


- Enter Connection name and remaining information and click on Test Connection.



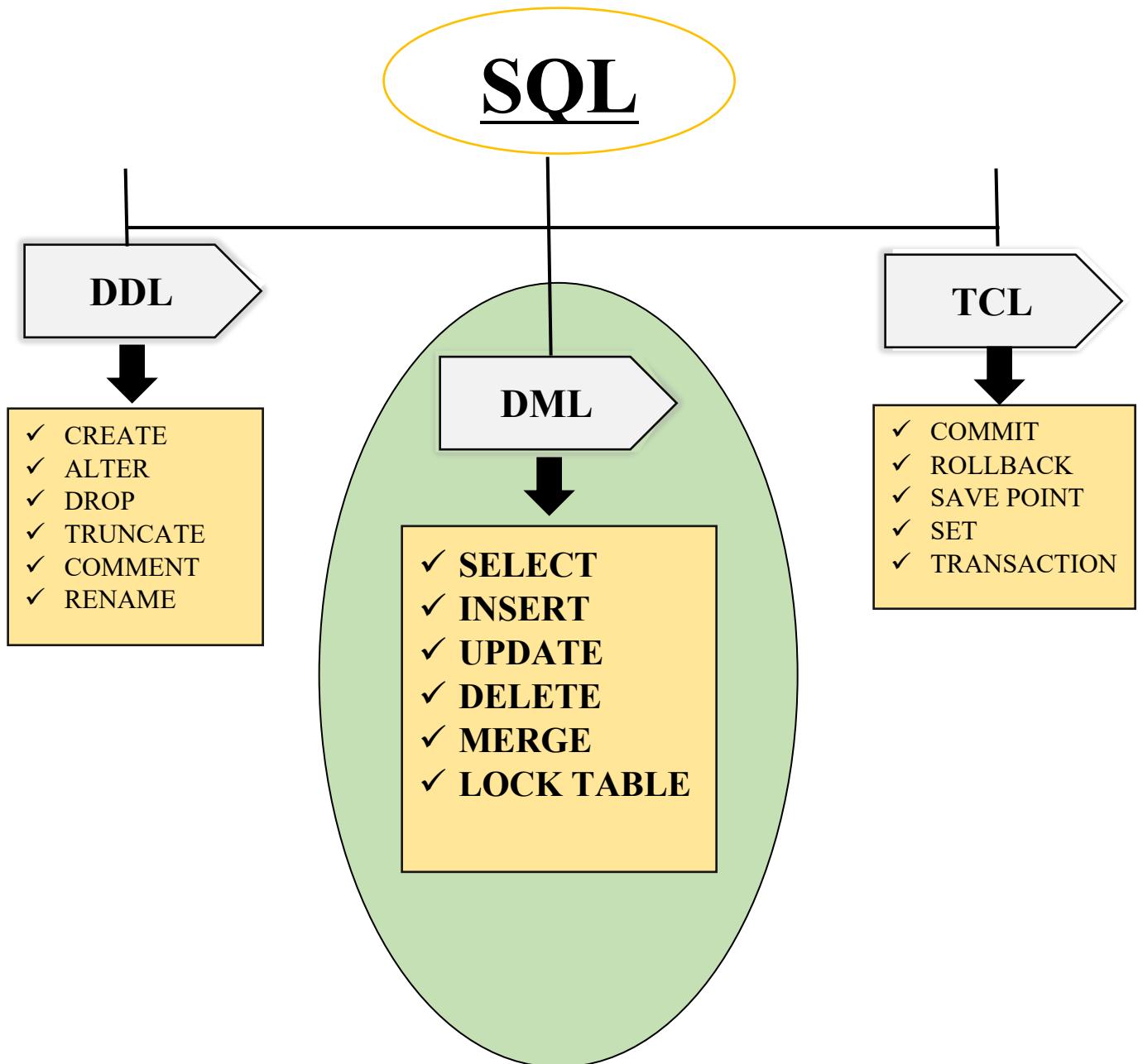
- Enter password for the target user. And click ok. Connection will be established





2. SQL SELECT statement

Structure Query Language (SQL)



Describing Table Schema:

To see the schema of the table, type the following query

This output gives the attribute name, specify this attribute is null or not and also display the data type of the attribute.

The screenshot shows the MySQL Workbench interface. In the Navigator pane, under the 'sakila' schema, the 'customer' table is selected. In the central Query Grid, the command `desc sakila.customer;` is entered. The Result Grid displays the following table structure:

Field	Type	Null	Key	Default	Extra
customer_id	smallint(5) unsigned	NO	PRI	NULL	auto_increment
store_id	tinyint(3) unsigned	NO	MUL	NULL	
first_name	varchar(45)	NO		NULL	
last_name	varchar(45)	NO	MUL	NULL	
email	varchar(50)	YES		NULL	
address_id	smallint(5) unsigned	NO	MUL	NULL	
active	tinyint(1)	NO		1	
create_date	datetime	NO		NULL	
last_update	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED on update CURRENT_TIMESTAMP

SELECT

SELECT statement is used to fetch data from a database table.

Syntax: `SELECT column_name(s) FROM table_name;`

The screenshot shows the MySQL Workbench interface. In the Navigator pane, under the 'sakila' schema, the 'customer' table is selected. In the central Query Grid, the command `select * from sakila.customer;` is entered. The Result Grid displays the following data:

customer_id	store_id	first_name	last_name	email	address_id	active	create_date	last_update
1	1	MARY	SMITH	MARY.SMITH@sakilacustomer.org	5	1	2006-02-14 22:04:36	2006-02-14 22:04:36
2	1	PATRICIA	JOHNSON	PATRICIA.JOHNSON@sakilacustomer.org	6	1	2006-02-14 22:04:36	2006-02-14 22:04:36
3	1	LINDA	WILLIAMS	LINDA.WILLIAMS@sakilacustomer.org	7	1	2006-02-14 22:04:36	2006-02-14 22:04:36
4	2	BARBARA	JONES	BARBARA.JONES@sakilacustomer.org	8	1	2006-02-14 22:04:36	2006-02-14 22:04:36
5	1	ELIZABETH	BROWN	ELIZABETH.BROWN@sakilacustomer.org	9	1	2006-02-14 22:04:36	2006-02-14 22:04:36
6	2	JENNIFER	DAVIS	JENNIFER.DAVIS@sakilacustomer.org	10	1	2006-02-14 22:04:36	2006-02-14 22:04:36
7	1	MARIA	MILLER	MARIA.MILLER@sakilacustomer.org	11	1	2006-02-14 22:04:36	2006-02-14 22:04:36
8	2	SUSAN	WILSON	SUSAN.WILSON@sakilacustomer.org	12	1	2006-02-14 22:04:36	2006-02-14 22:04:36
9	2	MARGARET	MOORE	MARGARET.MOORE@sakilacustomer.org	13	1	2006-02-14 22:04:36	2006-02-14 22:04:36
10	1	DOROTHY	TAYLOR	DOROTHY.TAYLOR@sakilacustomer.org	14	1	2006-02-14 22:04:36	2006-02-14 22:04:36
11	2	LISA	ANDERSON	LISA.ANDERSON@sakilacustomer.org	15	1	2006-02-14 22:04:36	2006-02-14 22:04:36
12	1	NANCY	THOMAS	NANCY.THOMAS@sakilacustomer.org	16	1	2006-02-14 22:04:36	2006-02-14 22:04:36
13	2	KAREN	JACKSON	KAREN.JACKSON@sakilacustomer.org	17	1	2006-02-14 22:04:36	2006-02-14 22:04:36
14	2	BETTY	WHITE	BETTY.WHITE@sakilacustomer.org	18	1	2006-02-14 22:04:36	2006-02-14 22:04:36
15	1	HELEN	HARRIS	HELEN.HARRIS@sakilacustomer.org	19	1	2006-02-14 22:04:36	2006-02-14 22:04:36
16	2	SANDRA	MARTIN	SANDRA.MARTIN@sakilacustomer.org	20	0	2006-02-14 22:04:36	2006-02-14 22:04:36

Select all columns:

The asterisk (*) in **select *** tells the database to return all the columns associated with the given table described in the **FROM** clause.

Selecting Specific Columns:

In the query example given below, each column name is listed in the SELECT clause.

The screenshot shows the MySQL Workbench interface with a query editor window titled "countrylanguage". The query is:

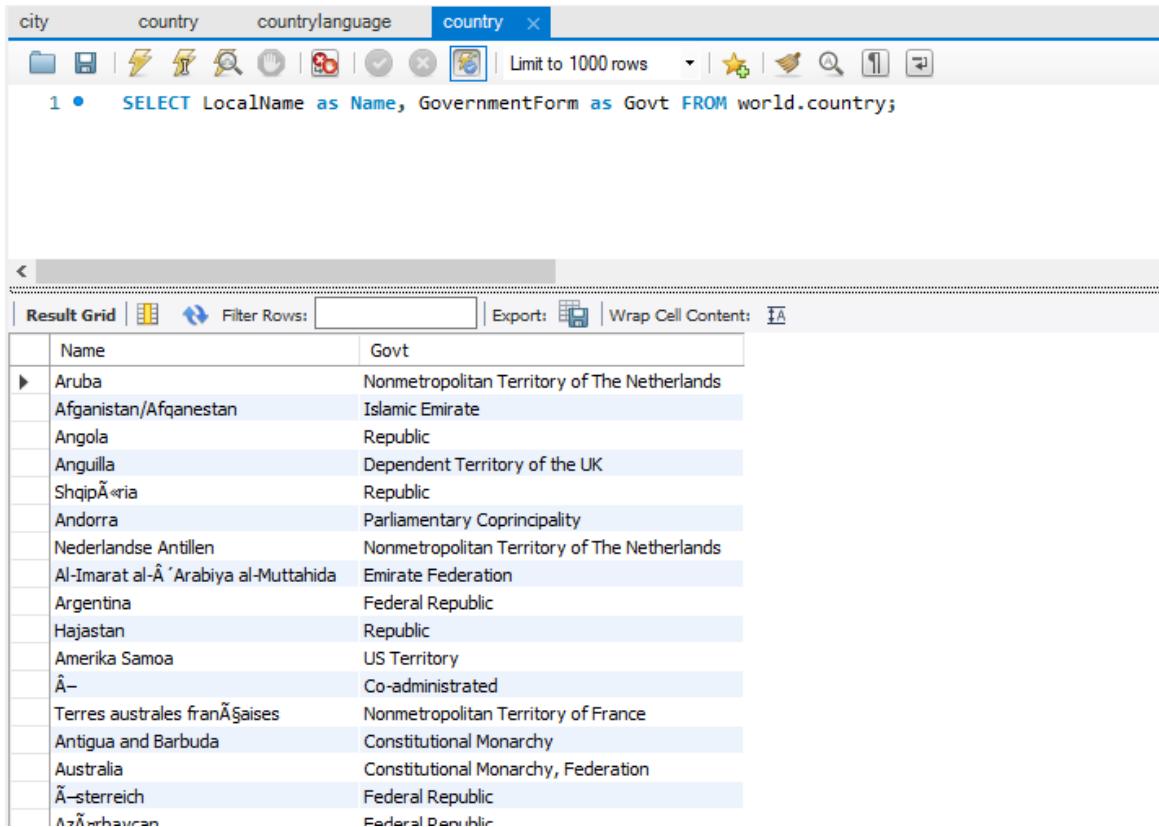
```
SELECT CountryCode, Language FROM world.countrylanguage;
```

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

	CountryCode	Language
▶	ABW	Dutch
	ABW	English
	ABW	Papiamento
	ABW	Spanish
	AFG	Balochi
	AFG	Dari
	AFG	Pashto
	AFG	Turkmenian
	AFG	Uzbek
	AGO	Ambo
	AGO	Chokwe
	AGO	Kongo
	AGO	Luchazi
	AGO	Luimbe-ng...
	AGO	Luvale
	AGO	Makonde

Defining Column Alias:

A column alias renames a column heading. It is useful with calculations immediately follows the column name (There can also be the optional AS keyword between the column name and alias.) It requires double quotation marks if it contains spaces or special characters or if it is case-sensitive.



The screenshot shows the MySQL Workbench interface. The top bar has tabs for 'city', 'country', 'countrylanguage', and 'country' (which is selected). Below the tabs is a toolbar with various icons. A status bar at the bottom says 'Limit to 1000 rows'. The main area displays a query result grid. The first row of the grid contains the column headers 'Name' and 'Govt'. The data rows show country names and their government types. Some names are in English and some are in their native language (e.g., 'Aruba', 'Afghanistan/Afghanistan', 'Angola'). The 'Govt' column includes entries like 'Nonmetropolitan Territory of The Netherlands', 'Islamic Emirate', 'Republic', 'Dependent Territory of the UK', 'Republic', 'Parliamentary Coprincedom', 'Nonmetropolitan Territory of The Netherlands', 'Emirate Federation', 'Federal Republic', 'Republic', 'US Territory', 'Co-administrated', 'Nonmetropolitan Territory of France', 'Constitutional Monarchy', 'Constitutional Monarchy, Federation', 'Federal Republic', and 'Federal Monarchy'.

Name	Govt
Aruba	Nonmetropolitan Territory of The Netherlands
Afghanistan/Afghanistan	Islamic Emirate
Angola	Republic
Anguilla	Dependent Territory of the UK
Shqipëria	Republic
Andorra	Parliamentary Coprincedom
Nederlandse Antillen	Nonmetropolitan Territory of The Netherlands
Al-Imarat al-`Arabiya al-Muttaahida	Emirate Federation
Argentina	Federal Republic
Hajastan	Republic
Amerika Samoa	US Territory
Ã—	Co-administrated
Terres australes franÃ§aises	Nonmetropolitan Territory of France
Antigua and Barbuda	Constitutional Monarchy
Australia	Constitutional Monarchy, Federation
Ã—sterreich	Federal Republic
Ã—sland	Federal Monarchy

Where Clause

The WHERE clause is used to filter records at the time of SELECT.

Syntax: SELECT [*] FROM [Table Name] WHERE [condition(s)]

- WHERE clause can be used to apply various comma separated condition, in one or more tables.
- Using the WHERE clause to select the specified condition.
- Specific conditions using AND or OR operators

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to
BETWEEN ...AND...	Between two values (inclusive)

IN (set)	Match any of a list of values
LIKE	Match a character pattern
IS NULL	Is a null value

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

```
1 •   SELECT * FROM world.city WHERE Population > 1000000;
```

The results grid displays city data from the world.city table, filtered by population greater than 1,000,000. The columns are ID, Name, CountryCode, District, and Population. The data includes major cities like Kabul, Alger, Luanda, Buenos Aires, La Matanza, CÃ³rdoba, Yerevan, Sydney, Melbourne, Brisbane, Perth, Baku, Dhaka, Chittagong, SÃ£o Paulo, Rio de Janeiro, and Salvador.

ID	Name	CountryCode	District	Population
1	Kabul	AFG	Kabol	1780000
35	Alger	DZA	Alger	2168000
56	Luanda	AGO	Luanda	2022000
69	Buenos Aires	ARG	Distrito Federal	2982146
70	La Matanza	ARG	Buenos Aires	1266461
71	CÃ³rdoba	ARG	CÃ³rdoba	1157507
126	Yerevan	ARM	Yerevan	1248700
130	Sydney	AUS	New South Wales	3276207
131	Melbourne	AUS	Victoria	2865329
132	Brisbane	AUS	Queensland	1291117
133	Perth	AUS	West Australia	1096829
144	Baku	AZE	Baki	1787800
150	Dhaka	BGD	Dhaka	3612850
151	Chittagong	BGD	Chittagong	1392860
206	SÃ£o Paulo	BRA	SÃ£o Paulo	9968485
207	Rio de Janeiro	BRA	Rio de Janeiro	5598953
208	Salvador	BRA	Rahis	2307837

In the above example, select the records of only those countries whose population is greater than equal to 1000000.

Using BETWEEN condition:

BETWEEN condition is used to display the rows based on a range of values given in query

The screenshot shows a MySQL Workbench interface with a query editor window titled "city". The query is:

```
1 •  SELECT * FROM world.city WHERE Population between 1000000 and 3000000;
```

The results grid displays the following data:

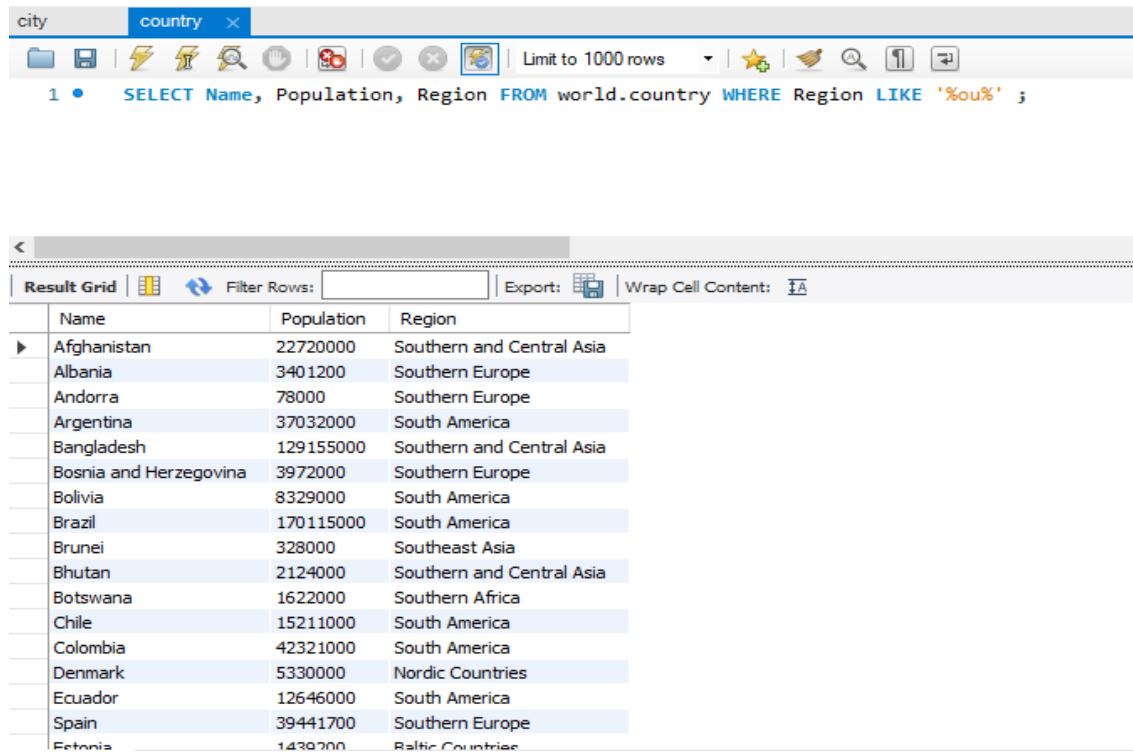
	ID	Name	CountryCode	District	Population
▶	1	Kabul	AFG	Kabul	1780000
	35	Alger	DZA	Alger	2168000
	56	Luanda	AGO	Luanda	2022000
	69	Buenos Aires	ARG	Distrito Federal	2982146
	70	La Matanza	ARG	Buenos Aires	1266461
	71	CÃ³rdoba	ARG	CÃ³rdoba	1157507
	126	Yerevan	ARM	Yerevan	1248700
	131	Melbourne	AUS	Victoria	2865329
	132	Brisbane	AUS	Queensland	1291117
	133	Perth	AUS	West Australia	1096829
	144	Baku	AZE	Baki	1787800
	151	Chittagong	BGD	Chittagong	1392860
...

Using **LIKE** condition:

LIKE condition is used to perform searches of valid search string values. Search conditions can contain either literal characters or numbers. Like condition includes some symbols: % is used to denote zero or many characters, while _ represent one character.

LIKE Operator	Description
WHERE District LIKE 'a% '	Finds any values that start with "a"
WHERE District LIKE '%a'	Finds any values that end with "a"
WHERE District LIKE '%or%'	Finds any values that have "or" in any position
WHERE District LIKE '_r%	Finds any values that have "r" in the second position
WHERE District LIKE 'a__% '	Finds any values that start with "a" and are at least 3 characters in length
WHERE District LIKE 'a%o'	Finds any values that start with "a" and ends with "o"

The mentioned Query shows the Name, Population and Region where Region name have “ou” on any location.



A screenshot of MySQL Workbench showing a query results grid. The title bar says "city country". The toolbar has various icons for file operations and database management. Below the toolbar, a query editor window displays the following SQL code:

```
1 •   SELECT Name, Population, Region FROM world.country WHERE Region LIKE '%ou%' ;
```

The results grid shows the following data:

Name	Population	Region
Afghanistan	22720000	Southern and Central Asia
Albania	3401200	Southern Europe
Andorra	78000	Southern Europe
Argentina	37032000	South America
Bangladesh	129155000	Southern and Central Asia
Bosnia and Herzegovina	3972000	Southern Europe
Bolivia	8329000	South America
Brazil	170115000	South America
Brunei	328000	Southeast Asia
Bhutan	2124000	Southern and Central Asia
Botswana	1622000	Southern Africa
Chile	15211000	South America
Colombia	42321000	South America
Denmark	5330000	Nordic Countries
Ecuador	12646000	South America
Spain	39441700	Southern Europe
Estonia	1439200	Baltic Countries

Logical Conditions:

Operator	Meaning
AND	Returns TRUE if <i>both</i> component conditions are true
OR	Returns TRUE if <i>either</i> component condition is true
NOT	Returns TRUE if the following condition is false

Using the AND Operator:

AND operator is used to perform logical comparisons. AND operator indicates that it gives true result when both conditions are true.

The screenshot shows the MySQL Workbench interface. At the top, there are tabs for 'city', 'country', 'countrylanguage', 'country' (which is the active tab), and 'country'. Below the tabs is a toolbar with various icons. The main area contains a SQL editor with the following query:

```

1 •  SELECT LocalName as Name FROM world.country
2      where Name like '_a%' and Region = "Southern Europe";

```

Below the SQL editor is a 'Result Grid' pane. It has a header row with a single column labeled 'Name'. The data rows are:

Name
Makedonija
Malta
San Marino

In the above example, select Name of those Countries whose name contain ‘a’ at second position and Region is Southern Europe.

ORDER BY Clause:

ORDER BY clause is used to sort retrieved rows with the ORDER BY clause. The ORDER BY clause comes last in the SELECT statement. By default, it is in ascending order, but you can change it in descending order. Following symbols are used:

- ASC: ascending order, default
- DESC: descending order

The screenshot shows the MySQL Workbench interface. At the top, there are tabs for 'city', 'country', 'countrylanguage', 'country', 'country', 'country', 'countrylanguage', 'city', and 'country' (which is the active tab). Below the tabs is a toolbar with various icons. The main area contains a SQL editor with the following query:

```

1 •  SELECT * FROM world.country order by Population;

```

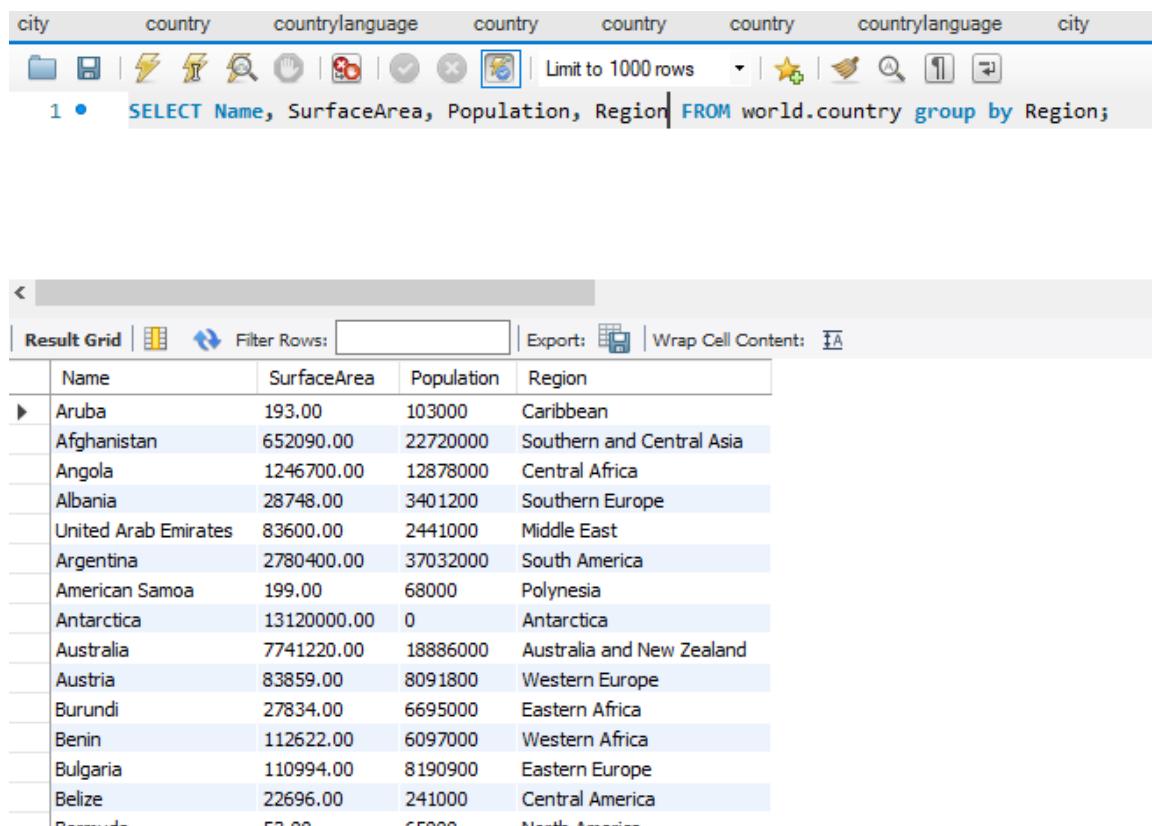
Below the SQL editor is a 'Result Grid' pane. It has a header row with columns: Continent, Region, SurfaceArea, IndepYear, Population, LifeExpectancy, GNP, GNPOld, and LocalName. The data rows are:

Continent	Region	SurfaceArea	IndepYear	Population	LifeExpectancy	GNP	GNPOld	LocalName
Antarctica	Antarctica	7780.00	NULL	0	NULL	0.00	NULL	Terres australes franÃ§aises
Antarctica	Antarctica	59.00	NULL	0	NULL	0.00	NULL	BouvetÃ¸ya
Antarctica	Antarctica	359.00	NULL	0	NULL	0.00	NULL	Heard and McDonald Island
Africa	Eastern Africa	78.00	NULL	0	NULL	0.00	NULL	British Indian Ocean Territc
Antarctica	Antarctica	3903.00	NULL	0	NULL	0.00	NULL	South Georgia and the Sou
Oceania	Micronesia/Caribbean	16.00	NULL	0	NULL	0.00	NULL	United States Minor Outlyir
Oceania	Polynesia	49.00	NULL	50	NULL	0.00	NULL	Pitcairn
Oceania	Australia and New Zealand	14.00	NULL	600	NULL	0.00	NULL	Cocos (Keeling) Islands
Europe	Southern Europe	0.40	1929	1000	NULL	9.00	NULL	Santa Sede/CittÃ del Vatic
South Am...	South America	12173.00	NULL	2000	NULL	0.00	NULL	Falkland Islands
Oceania	Australia and New Zealand	36.00	NULL	2000	NULL	0.00	NULL	Norfolk Island
Oceania	Polynesia	260.00	NULL	2000	NULL	0.00	NULL	Niue
Oceania	Polynesia	12.00	NULL	2000	NULL	0.00	NULL	Tokeleau
Oceania	Australia and New Zealand	135.00	NULL	2500	NULL	0.00	NULL	Christmas Island
Europe	Nordic Countries	62422.00	NULL	3200	NULL	0.00	NULL	Svalbard og Jan Mayen
Africa	Western Africa	314.00	NULL	6000	76.8	0.00	NULL	Saint Helena

Group BY Clause:

The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country".

The GROUP BY statement is often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns.



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

```
1 •   SELECT Name, SurfaceArea, Population, Region
      FROM world.country
     group by Region;
```

The results grid displays the following data:

Name	SurfaceArea	Population	Region
Aruba	193.00	103000	Caribbean
Afghanistan	652090.00	22720000	Southern and Central Asia
Angola	1246700.00	12878000	Central Africa
Albania	28748.00	3401200	Southern Europe
United Arab Emirates	83600.00	2441000	Middle East
Argentina	2780400.00	37032000	South America
American Samoa	199.00	68000	Polynesia
Antarctica	13120000.00	0	Antarctica
Australia	7741220.00	18886000	Australia and New Zealand
Austria	83859.00	8091800	Western Europe
Burundi	27834.00	6695000	Eastern Africa
Benin	112622.00	6097000	Western Africa
Bulgaria	110994.00	8190900	Eastern Europe
Belize	22696.00	241000	Central America

3. SQL Joins and sub Queries

Displaying Data from Multiple Tables (Joins):

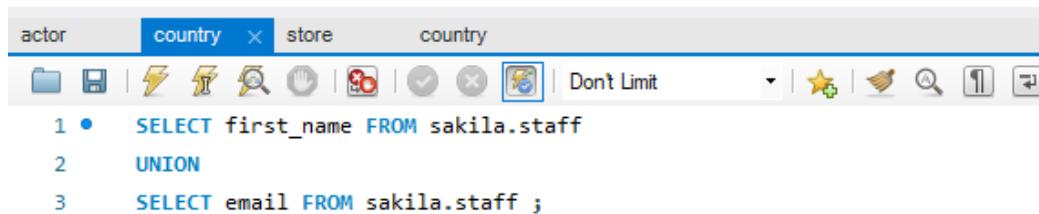
In this lab, we will explore how to write a SELECT statement to access data from more than one table using different types of joins.

1 SQL UNION Clause

- UNION combines the result sets of two queries.
- Column data types in the two queries must match.
- UNION combines by column position rather than column name.

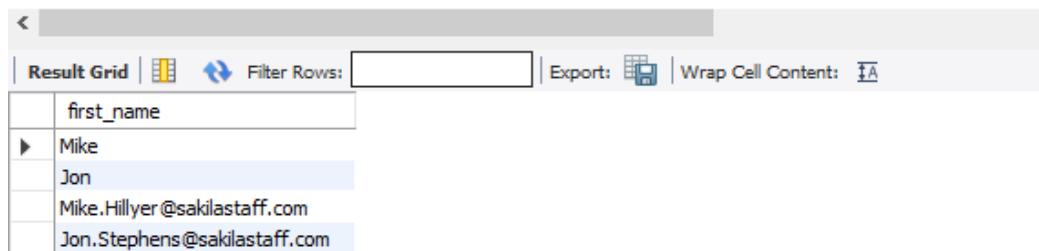
```
SELECT column-names  
  
FROM table-name  
  
UNION  
  
SELECT column-names  
  
FROM table-name
```

Example:



The screenshot shows the MySQL Workbench interface. The top bar has tabs for 'actor', 'country' (which is selected), 'store', and 'country'. Below the tabs is a toolbar with various icons. The main area contains a query editor with the following code:

```
1 •  SELECT first_name FROM sakila.staff  
2      UNION  
3      SELECT email FROM sakila.staff ;
```



The screenshot shows the results of the executed query in a 'Result Grid'. The grid has a single column labeled 'first_name'. The data rows are:

first_name
Mike
Jon
Mike.Hillyer@sakilastaff.com
Jon.Stephens@sakilastaff.com

Join

Join is used to query data from more than one table. Rows in one table can be joined to rows in another table according to common values existing in corresponding columns, that is usually primary and foreign key columns.'

Types of Join

- Inner Join
- Outer join
- Self join
- Cross join

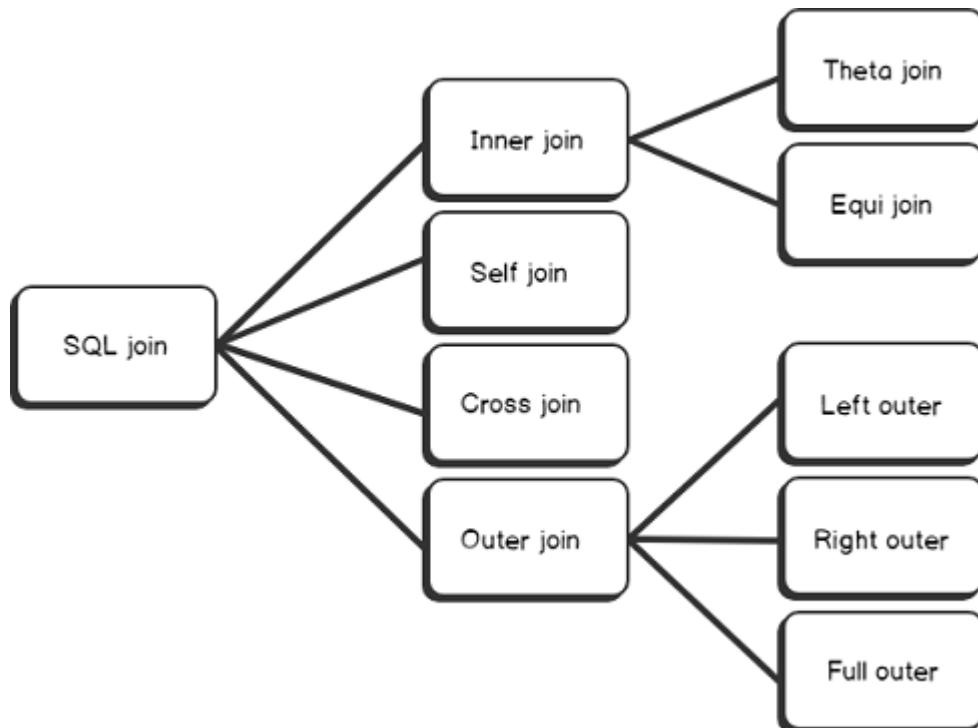


Fig 1: Types of Joins

Cartesian Product / Cross Join

A cartesian product is formed when all rows in the first table are joined to all rows in the second table , and when join condition is omitted or invalid. Cartesian products are useful for some tests when you need to generate a large number of rows to simulate a reasonable amount of time.

The screenshot shows a MySQL Workbench interface. The top bar has tabs for 'actor', 'address', 'city', 'country', 'film_category', 'actor', 'film_actor', 'staff', 'store', and 'address'. Below the tabs is a toolbar with icons for file operations, search, and database navigation. A query editor window contains the following SQL code:

```

1 •  SELECT a.first_name ,a.email, b.*
2   FROM sakila.staff a cross join sakila.store b;
3
4
5

```

Below the code is a 'Result Grid' table with the following data:

	first_name	email	store_id	manager_staff_id	address_id	last_update
▶	Mike	Mike.Hillyer@sakilastaff.com	1	1	1	2006-02-15 04:57:12
	Jon	Jon.Stephens@sakilastaff.com	1	1	1	2006-02-15 04:57:12
	Mike	Mike.Hillyer@sakilastaff.com	2	2	2	2006-02-15 04:57:12
	Jon	Jon.Stephens@sakilastaff.com	2	2	2	2006-02-15 04:57:12

Inner Join:

It will display all the records that have matched. The syntax of Inner join query is:

```
Select schema.table1.column (s)
From schema.table1 join schema.table2
on (schema.table1.column1=schema.table2.column1);
```

Example:

Query that displays staff information and corresponding to their address.

The screenshot shows a MySQL Workbench interface with the same top bar and toolbar as the previous screenshot. The query editor contains the following SQL code:

```

1 •  SELECT a.first_name ,a.email, b.* , c.address
2   FROM sakila.staff a join sakila.store b join sakila.address c
3   on (a.store_id = b.store_id) and (b.address_id=c.address_id);

```

Below the code is a 'Result Grid' table with the following data:

	first_name	email	store_id	manager_staff_id	address_id	last_update	address
▶	Mike	Mike.Hillyer@sakilastaff.com	1	1	1	2006-02-15 04:57:12	47 MySakila Drive
	Jon	Jon.Stephens@sakilastaff.com	2	2	2	2006-02-15 04:57:12	28 MySQL Boulevard

"ON" and "USING" clauses

In above JOIN query examples, we have used ON clause to match the records between table.

USING clause can also be used for the same purpose. The difference with USING is it **needs to have identical names for matched columns in both tables**.

Outer Join:

- **LEFT (OUTER) JOIN:** Select records from the first (left-most) table with matching right table records.

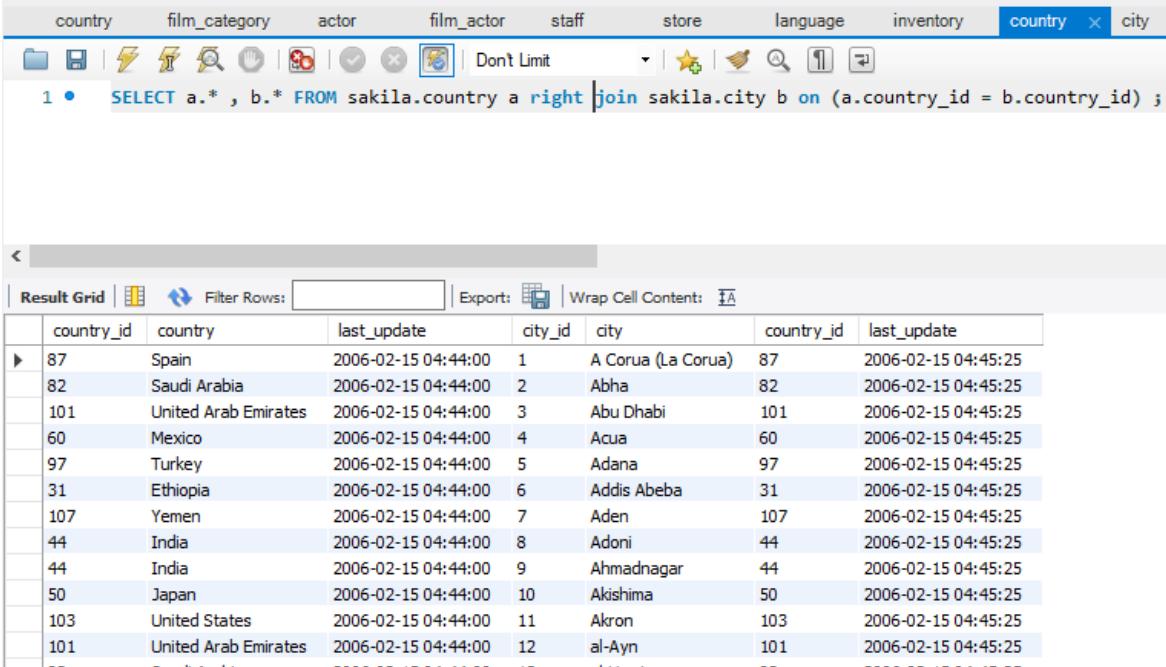
The screenshot shows the MySQL Workbench interface. The query editor window contains the following SQL code:

```
1 •  SELECT a.* , b.* FROM sakila.country a left join sakila.city b on (a.country_id = b.country_id) ;
```

The results grid displays the joined data from the country and city tables. The columns are: country_id, country, last_update, city_id, city, country_id, and last_update. The data includes rows for countries like Afghanistan, Algeria, American Samoa, Angola, Anguilla, and Argentina, along with their corresponding cities and IDs.

	country_id	country	last_update	city_id	city	country_id	last_update
▶	1	Afghanistan	2006-02-15 04:44:00	251	Kabul	1	2006-02-15 04:45:25
	2	Algeria	2006-02-15 04:44:00	59	Batna	2	2006-02-15 04:45:25
	2	Algeria	2006-02-15 04:44:00	63	Bchar	2	2006-02-15 04:45:25
	2	Algeria	2006-02-15 04:44:00	483	Skikda	2	2006-02-15 04:45:25
	3	American Samoa	2006-02-15 04:44:00	516	Tafuna	3	2006-02-15 04:45:25
	4	Angola	2006-02-15 04:44:00	67	Benguela	4	2006-02-15 04:45:25
	4	Angola	2006-02-15 04:44:00	360	Namibe	4	2006-02-15 04:45:25
	5	Anguilla	2006-02-15 04:44:00	493	South Hill	5	2006-02-15 04:45:25
	6	Argentina	2006-02-15 04:44:00	20	Almirante Brown	6	2006-02-15 04:45:25

- **RIGHT (OUTER) JOIN:** Select records from the second (right-most) table with matching left table records.



The screenshot shows the MySQL Workbench interface. A query window at the top contains the SQL command: `SELECT a.* , b.* FROM sakila.country a right join sakila.city b on (a.country_id = b.country_id) ;`. Below it is a result grid titled "Result Grid" showing the joined data. The columns are: country_id, country, last_update, city_id, city, country_id, and last_update. The data includes rows for Spain, Saudi Arabia, United Arab Emirates, Mexico, Turkey, Ethiopia, Yemen, India, India, Japan, United States, and United Arab Emirates.

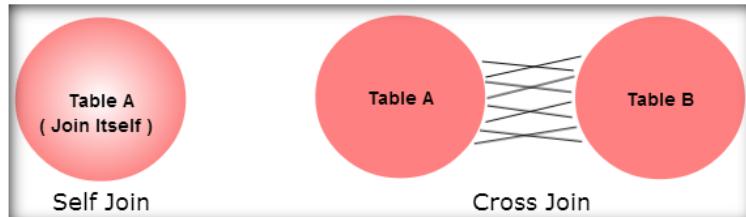
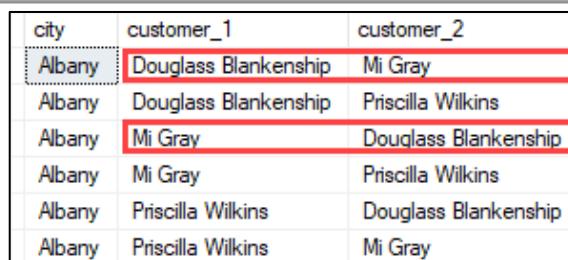
country_id	country	last_update	city_id	city	country_id	last_update
87	Spain	2006-02-15 04:44:00	1	A Corua (La Corua)	87	2006-02-15 04:45:25
82	Saudi Arabia	2006-02-15 04:44:00	2	Abha	82	2006-02-15 04:45:25
101	United Arab Emirates	2006-02-15 04:44:00	3	Abu Dhabi	101	2006-02-15 04:45:25
60	Mexico	2006-02-15 04:44:00	4	Acua	60	2006-02-15 04:45:25
97	Turkey	2006-02-15 04:44:00	5	Adana	97	2006-02-15 04:45:25
31	Ethiopia	2006-02-15 04:44:00	6	Addis Abeba	31	2006-02-15 04:45:25
107	Yemen	2006-02-15 04:44:00	7	Aden	107	2006-02-15 04:45:25
44	India	2006-02-15 04:44:00	8	Adoni	44	2006-02-15 04:45:25
44	India	2006-02-15 04:44:00	9	Ahmadnagar	44	2006-02-15 04:45:25
50	Japan	2006-02-15 04:44:00	10	Akishima	50	2006-02-15 04:45:25
103	United States	2006-02-15 04:44:00	11	Akron	103	2006-02-15 04:45:25
101	United Arab Emirates	2006-02-15 04:44:00	12	al-Ayn	101	2006-02-15 04:45:25

- **FULL (OUTER) JOIN:** Selects all records that match either left or right table records

```
SELECT * FROM t1
LEFT JOIN t2 ON t1.id = t2.id
UNION
SELECT * FROM t1
RIGHT JOIN t2 ON t1.id = t2.id
```

Self join:

A self-join is a join in which a table is joined with itself (which is also called Unary relationships), especially when the table has a FOREIGN KEY which references its own PRIMARY KEY. To join a table itself means that each row of the table is combined with itself and with every other row of the table.

The screenshot shows the MySQL Workbench interface. A query window at the top contains the SQL command: `SELECT city, customer_1, customer_2 FROM sakila.customer WHERE city = 'Albany' ;`. Below it is a result grid showing the data. The columns are: city, customer_1, and customer_2. The data includes rows for Albany, Douglass Blankenship, Mi Gray; Albany, Douglass Blankenship, Priscilla Wilkins; Albany, Mi Gray, Douglass Blankenship; Albany, Priscilla Wilkins, Douglass Blankenship; and Albany, Priscilla Wilkins, Mi Gray.

city	customer_1	customer_2
Albany	Douglass Blankenship	Mi Gray
Albany	Douglass Blankenship	Priscilla Wilkins
Albany	Mi Gray	Douglass Blankenship
Albany	Priscilla Wilkins	Douglass Blankenship
Albany	Priscilla Wilkins	Mi Gray

The screenshot shows a MySQL Workbench interface. The top bar has tabs for 'actor', 'country', 'store', and 'country' (which is selected). Below the tabs is a toolbar with various icons. A query editor window contains the following SQL code:

```

1 •   SELECT A.Name , B.Name , a.Region
2     FROM world.country A, world.country B
3    WHERE A.Code <> B.Code
4    AND A.Region = B.Region
5

```

Below the code is a 'Result Grid' table with three columns: 'Name', 'Name', and 'Region'. The data shows 14 rows where two countries share the same region but have different codes. All countries listed are in the 'Caribbean' region.

	Name	Name	Region
▶	Anguilla	Aruba	Caribbean
	Netherlands Antilles	Aruba	Caribbean
	Antigua and Barbuda	Aruba	Caribbean
	Bahamas	Aruba	Caribbean
	Barbados	Aruba	Caribbean
	Cuba	Aruba	Caribbean
	Cayman Islands	Aruba	Caribbean
	Dominica	Aruba	Caribbean
	Dominican Republic	Aruba	Caribbean
	Guadeloupe	Aruba	Caribbean
	Grenada	Aruba	Caribbean
	Haiti	Aruba	Caribbean
	Jamaica	Aruba	Caribbean
	Saint Kitts and Nevis	Aruba	Caribbean
	Saint Lucia	Aruba	Caribbean

1.1 The SQL subquery syntax

There is no general syntax; subqueries are regular queries placed inside parenthesis. Subqueries can be used in different ways and at different locations inside a query:

Here is an subquery with the IN operator

Syntax:

```

SELECT column-names
FROM table-name1
WHERE value IN
      (SELECT column-name
       FROM table-name2
       WHERE condition)

```

The screenshot shows a MySQL Workbench interface. The top bar has tabs for 'actor', 'country', 'store', and 'country' (selected). Below the tabs is a toolbar with various icons. A query editor window contains the following SQL code:

```

1 •   SELECT
2       Name, LifeExpectancy
3     FROM
4       world.country
5    WHERE
6       Code IN (SELECT Code FROM world.country WHERE Region = 'Southern Europe');

```

Below the code is a 'Result Grid' table with two columns: 'Name' and 'LifeExpectancy'. The data shows life expectancy values for 14 countries in the 'Southern Europe' region.

	Name	LifeExpectancy
▶	Albania	71.6
	Andorra	83.5
	Bosnia and H...	71.5
	Spain	78.8
	Gibraltar	79.0
	Greece	78.4
	Croatia	73.7
	Italy	79.0
	Macedonia	73.8
	Malta	77.9
	Portugal	75.8
	San Marino	81.1
	Slovenia	74.9

Subqueries can also assign column values for each record:

```
SELECT column1 = (SELECT column-name FROM table-name WHERE condition),
column-names

FROM table-name

WHERE condition;
```

Extract data based on some other table

The screenshot shows the MySQL Workbench interface. The query editor window contains the following SQL code:

```
1 •  select first_name , last_name FROM sakila.actor WHERE
2   actor_id in (
3     select b.actor_id from sakila.film_actor b
4     where b.film_id in (
5       SELECT c.film_id FROM sakila.film c
6       WHERE c.special_features = 'Trailers,Deleted Scenes'));
```

The result grid displays the following data:

first_name	last_name
PENELOPE	GUINNESS
NICK	WAHLBERG
ED	CHASE
JENNIFER	DAVIS
BETTE	NICHOLSON
GRACE	MOSTEL
MATTHEW	JOHANSSON
JOE	SWANK
CHRISTIAN	GABLE
ZERO	CAGE
KARL	BERRY
UMA	WOOD
VIVIEN	BERGEN
CLURA	OITVTFR

```
SELECT name , Continent, max(mycount)
FROM (SELECT Name,Continent,COUNT(Name) as mycount
      FROM world.country
      GROUP BY Continent) as cnt;
```

4. SQL INSERT, UPDATE AND DELETE statements

INSERT Statement

It is possible to write the INSERT INTO statement in two ways.

- The first way specifies both the column names and the values to be inserted:

Syntax:

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

- If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. The INSERT INTO syntax would be as follows:

Syntax:

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

```
1 • insert into world.city
2 values ( 10001,"AYESHA","NLD", "GUJAERKHAN",1000000);
3 |
```

Handle NULL Values

It is not possible to test for NULL values with comparison operators, such as =, <, or <>. We will have to use the **IS NULL** and **IS NOT NULL** operators instead.

Syntax:

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

The SQL SELECT TOP Clause

The SELECT TOP clause is used to specify the number of records to return.

Syntax:

```
SELECT column name(s)
FROM table_name
WHERE condition
LIMIT number;
```

1.2 The SQL INSERT INTO SELECT Statement

The INSERT INTO SELECT statement copies data from one table and inserts it into another table.

- INSERT INTO SELECT requires that data types in source and target tables match
- The existing records in the target table are unaffected

1.2.1 INSERT INTO SELECT Syntax

Copy all columns from one table to another table:

Syntax:

```
INSERT INTO table2
SELECT * FROM table1
WHERE condition;
```

Example

The screenshot shows the MySQL Workbench interface. The query editor at the top contains the following SQL code:

```
5
6 • insert into world.city (CountryCode)
7   select CountryCode from world.countrylanguage
8   where Language = "English";
9
```

The result grid below displays the data inserted into the city table. The columns are ID, Name, CountryCode, District, and Population. The data is as follows:

	ID	Name	CountryCode	District	Population
1	10000	RAWAT	AFG	GUJAERKHAN	1000000
2	10001	AYESHA	NLD	GUJAERKHAN	1000000
3	10002		ABW		0
4	10003		AIA		0
5	10004		ANT		0
6	10005		ASM		0
7	10006		ATG		0
8	10007		AUS		0
9	10008		BHR		0
10	10009		BLZ		0
11	10010		BMU		0
12	10011		BRB		0
13	10012		BRN		0
14		-

SQL UPDATE Statement

The UPDATE statement is used to modify the existing records in a table.

Syntax:

```
UPDATE table_name  
SET column1 = value1, column2 = value2, ...  
WHERE condition;
```

- `UPDATE world.city |`
`SET Name = "AYESHA"`
`WHERE ID= 10001;`

Note: You can't update or delete records without specifying a key (ex. primary key) in the where clause. If you want kindly update the save mode by the following query.

OR

- `SET SQL_SAFE_UPDATES = 0; # update save mode`

- `UPDATE world.city`
`SET Name = "RAWAT"`
`WHERE Name = "AYESHA";`

1. Go to `Edit --> Preferences`
2. Click `"SQL Editor"` tab and `uncheck "Safe Updates"` check box
3. `Query --> Reconnect to Server` // logout and then login
4. Now execute your SQL query

The SQL CASE Statement

The CASE statement goes through conditions and returns a value when the first condition is met (like an IF-THEN-ELSE statement). So, once a condition is true, it will stop reading and return the result. If no conditions are true, it returns the value in the ELSE clause.

If there is no ELSE part and no conditions are true, it returns NULL.

Syntax:

```
CASE
    WHEN condition1 THEN result1
    WHEN condition2 THEN result2
        |
        |
    WHEN conditionN THEN resultN
    ELSE result
END;
```

Example:

Before: information of Independent Year of most of the records are missing in the Country column

The screenshot shows the MySQL Workbench interface with the 'country' schema selected. A query window displays the following SQL command:

```
1 •  SELECT * FROM world.country;
```

The result grid shows the following data from the world.country table:

	Code	Name	Continent	Region	SurfaceArea	IndepYear	Population	LifeExpectancy	GNP	GNPOld
▶	ABW	Aruba	North America	Caribbean	193.00	NULL	103000	78.4	828.00	793.00
	AFG	Afghanistan	Asia	Southern and Central Asia	652090.00	1919	22720000	45.9	5976.00	NULL
	AGO	Angola	Africa	Central Africa	1246700.00	1975	12878000	38.3	6648.00	7984.00
	AIA	Anguilla	North America	Caribbean	96.00	NULL	8000	76.1	63.20	NULL
	ALB	Albania	Europe	Southern Europe	28748.00	1912	3401200	71.6	3205.00	2500.00
	AND	Andorra	Europe	Southern Europe	468.00	1278	78000	83.5	1630.00	NULL
	ANT	Netherlands ...	North America	Caribbean	800.00	NULL	217000	74.7	1941.00	NULL
	ARE	United Arab E...	Asia	Middle East	83600.00	1971	2441000	74.1	37966.00	36846.00
	ARG	Argentina	South America	South America	2780400.00	1816	37032000	75.1	340238.00	323310.00
	ARM	Armenia	Asia	Middle East	29800.00	1991	3520000	66.4	1813.00	1627.00
	ASM	American Sa...	Oceania	Polynesia	199.00	NULL	68000	75.1	334.00	NULL
	ATA	Antarctica	Antarctica	Antarctica	13120000.00	NULL	0	NULL	0.00	NULL
	ATF	French South...	Antarctica	Antarctica	7780.00	NULL	0	NULL	0.00	NULL

After Applying following query, you are not only able to change the null values from the column but also replace information according to your need.

The screenshot shows the MySQL Workbench interface with the 'country' schema selected. A query window displays the following SQL command:

```
2 •  SELECT Name, Population,
3   CASE
4     WHEN IndepYear Is null THEN "Missing Independ year"
5     ELSE IndepYear
6   END AS Independent_Year
7   FROM world.country;
```

The result grid shows the following data from the world.country table, with the 'IndepYear' column replaced by 'Independent_Year' and null values replaced by 'Missing Independ year':

	Name	Population	Independent_Year
▶	Aruba	103000	Missing Independ year
	Afghanistan	22720000	1919
	Angola	12878000	1975
	Anguilla	8000	Missing Independ year
	Albania	3401200	1912
	Andorra	78000	1278
	Netherlands ...	217000	Missing Independ year
	United Arab E...	2441000	1971
	Argentina	37032000	1816
	Armenia	3520000	1991
	American Sa...	68000	Missing Independ year
	Antarctica	0	Missing Independ year

What are views?

Simply put, VIEWS are virtual tables. By virtual, we mean, the tables do not store any data of their own but display data stored in other tables.

In other words, VIEWS are nothing but SELECT Queries.

Views syntax

Let's now look at the basic syntax used to create a view in MySQL.

```
CREATE VIEW `view_name` AS SELECT statement;
```

- "**CREATE VIEW `view_name`**" tells MySQL server to create a view object in the database named `view_name`'
- "**AS SELECT statement**" is the SQL statements to be packed in the views. It can be a SELECT statement can contain data from one table or multiple tables.

Result:

The screenshot shows the MySQL Workbench interface. In the Navigator pane, under Schemas, there is a Views folder containing a view named test_tableb. The view has one column, ID, defined with type int(11). In the main workspace, the code for creating the view is shown:

```
8
9 •  create view Test_tableb as
10   select a.ID, a.District , b.* from world.city a join world.country b
11   on( a.CountryCode=b.Code );
12
13 •  select * from world.test_tableb;
```

Below the code, the Result Grid displays the data returned by the query:

ID	District	Code	Name	Continent	Region	SurfaceArea	IndepYear	Population	LifeExpectancy	GNP
129	Â-	ABW	Aruba	North America	Caribbean	193.00	NULL	103000	78.4	828.00
10002		ABW	Aruba	North America	Caribbean	193.00	NULL	103000	78.4	828.00
1	Kabul	AFG	Afghanistan	Asia	Southern and Central Asia	652090.00	1919	22720000	45.9	5976.0
2	Qandahar	AFG	Afghanistan	Asia	Southern and Central Asia	652090.00	1919	22720000	45.9	5976.0
3	Herat	AFG	Afghanistan	Asia	Southern and Central Asia	652090.00	1919	22720000	45.9	5976.0
4	Balkh	AFG	Afghanistan	Asia	Southern and Central Asia	652090.00	1919	22720000	45.9	5976.0
10000	GUJAERKHAN	AFG	Afghanistan	Asia	Southern and Central Asia	652090.00	1919	22720000	45.9	5976.0
56	Luanda	AGO	Angola	Africa	Central Africa	1246700.00	1975	12878000	38.3	6648.0
57	Huambo	AGO	Angola	Africa	Central Africa	1246700.00	1975	12878000	38.3	6648.0
58	Benguela	AGO	Angola	Africa	Central Africa	1246700.00	1975	12878000	38.3	6648.0
59	Benguela	AGO	Angola	Africa	Central Africa	1246700.00	1975	12878000	38.3	6648.0

SQL Delete Statement:

The DELETE statement is used to delete existing records in a table.

Syntax:

```
DELETE FROM table_name WHERE condition;
```

Example:

```

actor      country    staff   address   store   country
31
32 •    DELETE FROM world.city WHERE Name = 'AYESHA';
33
34 •    select * from world.city;
35

```

Result Grid

ID	Name	CountryCode	District	Population
4073	Gweru	ZWE	Midlands	128037
4074	Gaza	PSE	Gaza	353632
4075	Khan Yunis	PSE	Khan Yunis	123175
4076	Hebron	PSE	Hebron	119401
4077	Jabaliya	PSE	North Gaza	113901
4078	Nablus	PSE	Nablus	100231
4079	Rafah	PSE	Rafah	92020
10000	RAWAT	AFG	GUJAERKHAN	1000000
10002	ABW			0
10003	AIA			0
10004	ANT			0
10005	ASM			0
10006	ATG			0
10007	BIH			0

Example # 2:

```

actor      country    staff   address   store   country
30
31 •    DELETE FROM world.city WHERE Population =0;
32
33 •    select * from world.city;
34

```

Result Grid

ID	Name	CountryCode	District	Population
4068	Harare	ZWE	Harare	1410000
4069	Bulawayo	ZWE	Bulawayo	621742
4070	Chitungwiza	ZWE	Harare	274912
4071	Mount Darwin	ZWE	Harare	164362
4072	Mutare	ZWE	Manicaland	131367
4073	Gweru	ZWE	Midlands	128037
4074	Gaza	PSE	Gaza	353632
4075	Khan Yunis	PSE	Khan Yunis	123175
4076	Hebron	PSE	Hebron	119401
4077	Jabaliya	PSE	North Gaza	113901
4078	Nablus	PSE	Nablus	100231
4079	Rafah	PSE	Rafah	92020
10000	RAWAT	AFG	GUJAERKHAN	1000000
*	NULL	NULL	NULL	NULL

- Delete records having more than 2 columns are empty

```
DELETE FROM world.city
where Name ='' and District =''
```

Delete MYSQL TABLE

It is very easy to drop an existing MySQL table, but you need to be very careful while deleting any existing table because the data lost will not be recovered after deleting a table.

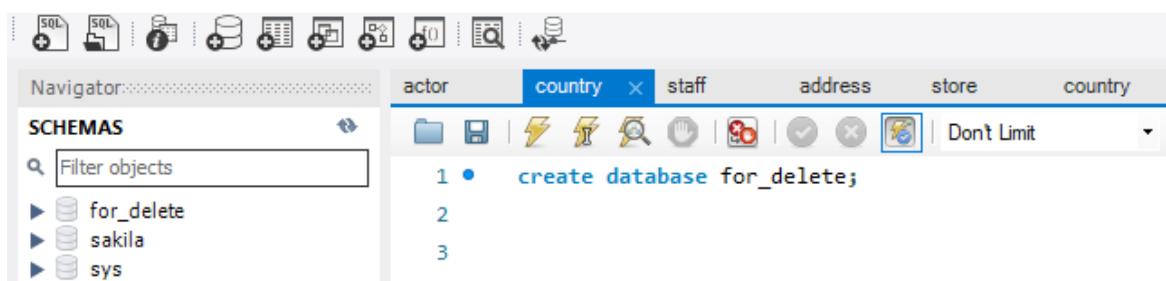
Syntax: `DROP TABLE table_name;`

Delete MYSQL Database

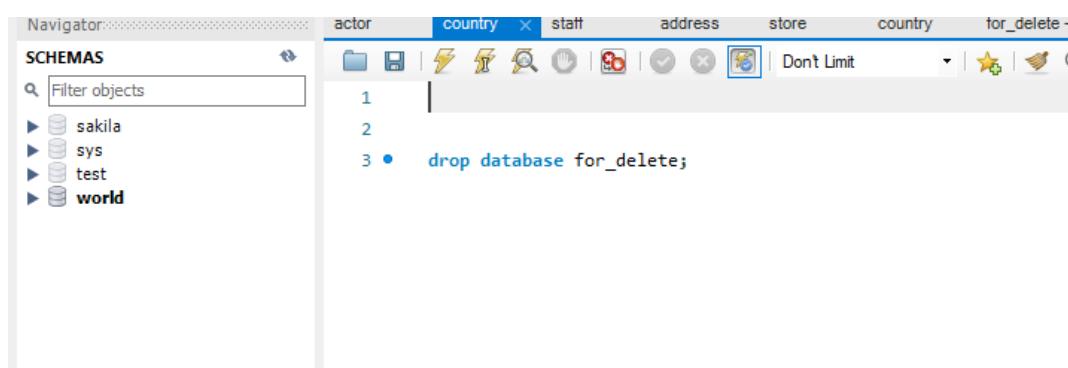
Syntax: `DROP DATABASE database_name;`

Example:

Create the new database:



Now, delete the created database by the following query as follows:



5. SQL CREATE AND ALTER statements

1.3 The SQL CREATE TABLE Statement

The CREATE TABLE statement is used to create a new table in a database.

Syntax:

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
    ...);
```

Data Types	Description
INT	✓ A normal-sized integer that can be signed or unsigned. If signed, the allowable range is from -2147483648 to 2147483647. ✓ You can specify a width of up to 11 digits.
DOUBLE (M, D)	✓ Define the display length (M) and the number of decimals (D). This is not required and will default to 16,4, where 4 is the number of decimals. Decimal precision can go to 53 places for a DOUBLE. ✓ Will default to 10,2 in FLOAT (M, D)
DATE	A date in YYYY-MM-DD format, between 1000-01-01 and 9999-12-31. For example, December 30 th , 1973 would be stored as 1973-12-30.
YEAR(M)	Stores a year in a 2-digit or a 4-digit format. If the length is specified as 2 (for example YEAR (2)), YEAR can be between 1970 to 2069 (70 to 69). If the length is specified as 4, then YEAR can be 1901 to 2155. The default length is 4.
TIME	Stores the time in a HH:MM: SS format.
DATETIME	A date and time combination in YYYY-MM-DD HH:MM: SS format, between 1000-01-01 00:00:00 and 9999-12-31 23:59:59. For example, 3:30 in the afternoon on December 30 th , 1973 would be stored as 1973-12-30 15:30:00.

CHAR(M)	A fixed-length string between 1 and 255 characters in length (for example CHAR (5)), right-padded with spaces to the specified length when stored. Defining a length is not required, but the default is 1.
VARCHAR(M)	A variable-length string between 1 and 255 characters in length. For example, VARCHAR (25). You must define a length when creating a VARCHAR field.

SQL Constraints

- SQL constraints are used to specify rules for the data in a table.
- Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.
- Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.

The following constraints are commonly used in SQL:

- **NOT NULL**
 - Ensures that a column cannot have a NULL value
- **UNIQUE**
 - Ensures that all values in a column are different
- **PRIMARY KEY**
 - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- **FOREIGN KEY**
 - Uniquely identifies a row/record in another table
- **CHECK**
 - Ensures that all values in a column satisfies a specific condition
- **DEFAULT**
 - Sets a default value for a column when no value is specified
- **INDEX**
 - Used to create and retrieve data from the database very quickly

ALTER TABLE Statement

Syntax:

```
ALTER TABLE table_name
  ADD new_column_name column_definition
    [ FIRST | AFTER column_name];
```

We will discuss how to use the MySQL **ALTER TABLE statement** to add a column, modify a column, drop a column, rename a column or rename a table.

NOT NULL

Example:

```
CREATE TABLE Schema.Persons (
    ID int NOT NULL,
    LastName varchar (255) NOT NULL,
    FirstName varchar (255) NOT NULL,
    Age int);
```

Add NOT NULL Constraint after creating table:

```
ALTER TABLE Persons
MODIFY Age int NOT NULL;
```

PRIMARY KEY

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar (255) NOT NULL,
    FirstName varchar (255),
    Age int,
    PRIMARY KEY (ID));
```

2nd Way:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    CONSTRAINT PK_Person PRIMARY KEY (ID, LastName));
```

The screenshot shows the MySQL Workbench interface with the 'actor' schema selected. In the central query editor, the following SQL code is displayed:

```

1 CREATE TABLE world.Person (
2     ID int NOT NULL,
3     LastName varchar(255) NOT NULL,
4     FirstName varchar(255),
5     Age int,
6     PRIMARY KEY (ID,LastName)
7 );
8 • desc world.Person;

```

Below the code, the results of the 'desc world.Person;' command are shown in a table:

Field	Type	Null	Key	Default	Extra
ID	int(11)	NO	PRI	NULL	
LastName	varchar(255)	NO	PRI	NULL	
FirstName	varchar(255)	YES		NULL	
Age	int(11)	YES		NULL	

Add Primary key Constraint after creating table:

```
ALTER TABLE Persons
ADD PRIMARY KEY (ID);
```

SQL FOREIGN KEY Constraint

- A FOREIGN KEY is a key used to link two tables together.
- A FOREIGN KEY is a field (or collection of fields) in one table that refers to the PRIMARY KEY in another table.
- The table containing the foreign key is called the child table, and the table containing the candidate key is called the referenced or parent table

Syntax:

```
CREATE TABLE Orders (
    OrderID int NOT NULL,
```

```
OrderNumber int NOT NULL,  
PersonID int,  
PRIMARY KEY (OrderID),  
FOREIGN KEY (PersonID) REFERENCES Persons (PersonID) );
```

OR

```
CREATE TABLE Orders (  
    OrderID int NOT NULL,  
    OrderNumber int NOT NULL,  
    PersonID int,  
    PRIMARY KEY (OrderID),  
    CONSTRAINT FK_PersonOrder FOREIGN KEY (PersonID)  
        REFERENCES Persons (PersonID));
```

Add foreign key Constraint after creating table:

```
ALTER TABLE Orders  
ADD FOREIGN KEY (PersonID) REFERENCES Persons (PersonID);
```

Index in MYSQL

The `CREATE INDEX` command is used to create indexes in tables (allows duplicate values).

Indexes are used to retrieve data from the database very fast. The users cannot see the indexes, they are just used to speed up searches/queries.

Syntax: `CREATE INDEX index_name ON table_name (column_list)`

The following SQL creates an index named "idx_lastname" on the "LastName" column in the "Persons" table:

```
CREATE INDEX idx_lastname  
ON Persons (LastName);
```

If you want to create an index on a combination of columns, you can list the column names within the parentheses, separated by commas:

```
CREATE INDEX idx_pname  
ON Persons (LastName, FirstName);
```

Show Indexes of a Table

Show indexes from table_name:

The screenshot shows the MySQL Workbench interface. In the top navigation bar, 'Local instance MySQL80' is selected. The 'Schemas' tab is active, showing the 'country' schema. In the central pane, a query editor window displays the following SQL code:

```

1 •  SELECT Name , Region from world.country
2   where Continent = "North America";
3
4 •  create index test on world.country(Continent);
5
6 •  SHOW INDEXES FROM world.country;

```

Below the code, the results of the 'SHOW INDEXES' query are shown in a table:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_col
country	0	PRIMARY	1	Code	A	239	HULL	HULL		BTREE		
country	1	test	1	Continent	A	7	HULL	HULL		BTREE		

Check

Syntax:

```
CREATE TABLE Persons (
    Age int,
    CHECK (Age>=18)
);
```

Using Alter query:

```
ALTER TABLE Persons
ADD CHECK (Age>=18);
```

Default

Syntax:

```
CREATE TABLE Persons (
    City varchar (255) DEFAULT 'ISLAMABAD'
);
```

- ALTER TABLE – Set Default value:

```
ALTER TABLE Persons
ALTER City SET DEFAULT 'ISLAMABAD ';
```

- **ALTER TABLE - ADD Column**

To add a column in a table, use the following syntax:

```
ALTER TABLE table_name
ADD column_name datatype;
```

- **ALTER TABLE - DROP COLUMN**

To delete a column in a table, use the following syntax.

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

Practical Task # 1, Hands-on MySQL

Instructions:

This task is designed to run-through all the basic concepts of database and data warehouse practically. In detailed every concept is discussed above to give you an overview of the installation process, commands and syntax. You have to understand and utilize all the information given in this manual to implement practical task#1. This task is divided into three subtasks respectively. You have to submit report of each subtask per week on Google classroom in case of Online teaching while need to present these to your TA in the class in case the classes are on Campus. Task grading will be based on how smartly you have mined the insights from database and to what extend your output is easily understandable. You have to submit a zip file which must be comprised of task report and .sql script file. The name of your zip file must be in this format: Rollno_FirsName-LastName-Task1A, otherwise your task will not be accepted.

Task 1A

- a. Create schema for Shopify in MySQL and populate data into the schema using attached shopcars.sql file.
- b. Write a query to display number of classic cars and motorcycles shipped per month in the year of 2003 and 2004.
- c. Write a query to display those customers from which city has made least purchase of vintage cars.
- d. Write a query to find the order numbers which have taken more than 4 days to be shipped and display the order number and number of days in a table.
- e. Write a query to display and find the remaining stock of per product in warehouse.

Task 1B

- a. Create a table Amount with two columns i.e ProfitAmount and Dates. Insert the values in the table using script file amount.sql. Write a query to display and find the start_date and end_date when there are continuous values in ProfitAmount column.
- b. How to replace multiple commas from a given text into single comma.
Text: 'ABC,,,DDDD,,,,,,DDE,,2LMDL3EMF,CEWEC,,,,,'
- c. Using the given Grades.sql file create a database and then solve the case study given below.

John gives James a task to generate a report containing three columns: Name, Grade, and Mark. John doesn't want the NAMES of those students who received a grade lower than 8. And if the grades of two students are same then order the name alphabetically.

- d. Create two table T1 and T2.

T1
4
2
1

T2
P
Q
R

Write a SQL query to display a drive of table T3 from T1 and T2.

T3
4PPP
2QQ
1R

Honor Policy: This assignment is a learning opportunity that will be evaluated based on your ability to think, work through a problem in a logical manner. You may however discuss verbally or via email the assignment with your classmates or the course instructor, and use the Internet to do your research, but the written work should be your own. Plagiarized reports or code will get a zero. If in doubt, ask the course instructor.

