PER SCHOLAS

Lesson 304.2 - Introduction to SQL Language

PER SCHOLAS

The SQL and MySQL

Learning Objectives:

By the end of this lesson, learners will be able to:

- Describe the SQL language and MySQL.
- Demonstrate how to install and configure MySQL.
- Download and import data into MySQL.
- Describe and utilize DQL, DML, DCL and TCL.

PER SCHOLAS

Table of Contents

Structured Query Language

SQL (pronounced “ess-que-el”) stands for **Structured Query Language**.

- SQL is used to communicate with a database. According to the American National Standards Institute (ANSI), SQL is the standard language for relational database management systems. SQL statements are used to perform tasks such as updating data on a database or retrieving data from a database.
- Although most database systems use SQL, most of them also have their own additional proprietary extensions. However, the standard SQL commands include ***“Select,” “Insert,” “Update,” “Delete,” “Create,”*** and ***“Drop.”***
- In this cohort, we will use SQL by using a MySQL database management system.

PER SCHOLAS

Structured Query Language (continued)

Structured Query Language (SQL) has five sub-languages:

- DQL - Data Query Language.
- DDL - Data Definition Language.
- DML- Data Manipulation Language.
- TCL - Transaction Control Language.
- DCL - Data Control Language.

MySQL

The term, MySQL is simply the combination of “My” and “SQL”: **MySQL**.

- ❑ MySQL is a database management system that allows you to manage relational databases. It is an open-source software backed by Oracle.
- ❑ MySQL can run on various platforms such as UNIX, Linux, Windows, etc. You can install it on a server or even on a desktop. MySQL is reliable, scalable, and fast.
- ❑ **InnoDB** became the default **storage engine** since the launch of **MySQL 5.5**.

Hands-On Lab - Installing MySQL

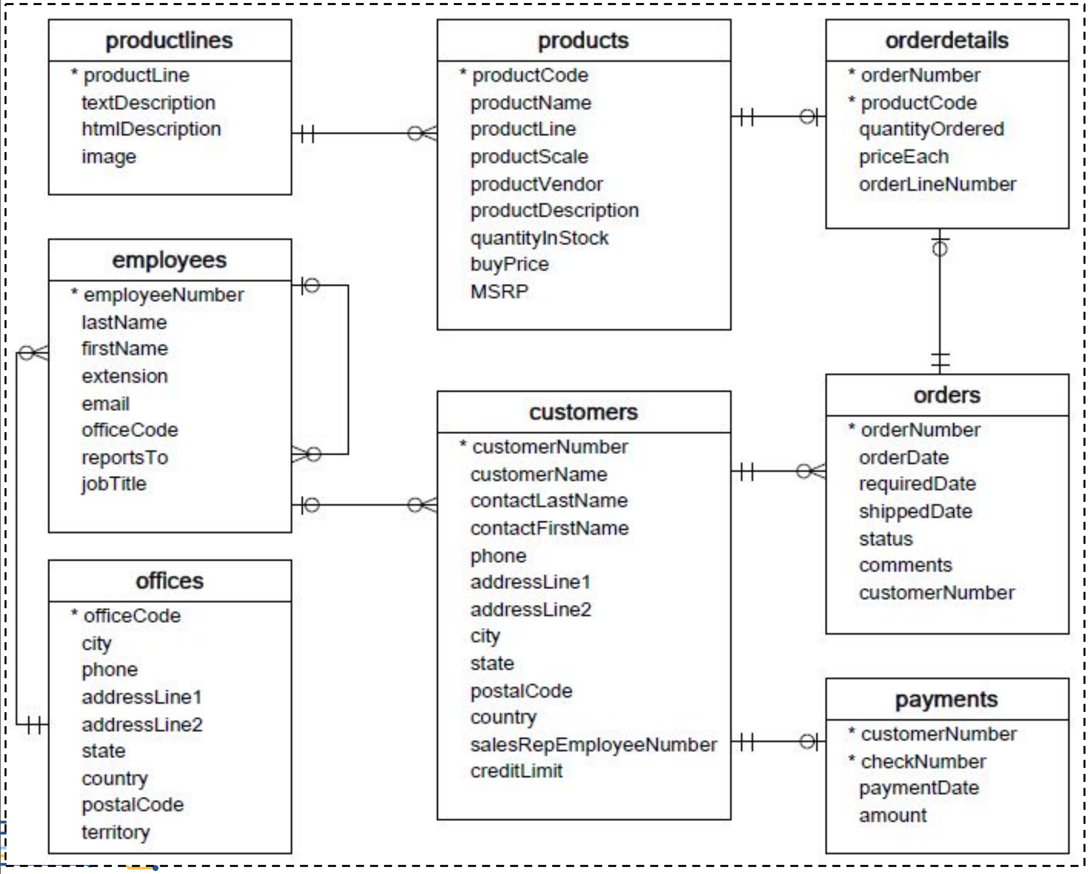
In this Lab, you will demonstrate installing the MySQL Workbench and a MySQL Server. You can find the following labs on Canvas under the **Assignment** section.

➔ **GLAB - 304.2.1 - Installing MySQL Workbench on the Windows OS and MacOS**

If you have any technical questions while performing the lab activity, ask your instructors for assistance.

PER SCHOLAS

Database Schema Diagram



We will use the **classicmodels** database as an SQL sample database for the demonstration to help your work with SQL effectively. The **classicmodels** database is a retailer of scale models of classic cars. It contains typical business data such as customers, products, sales orders, and sales order line items, etc. See the schema diagram to the left.

You can find GLAB - 304.2.2 - Download "Classicmodels" Database Database on Canvas under the Assignment section.

PER SCHOLAS

Data Query Language

- Data Query Language (DQL) statements are used for performing queries on data within schema objects. It allows for getting data from the database and imposing order.
- DQL includes the **SELECT** statement, which allows for getting the data out of the database and performing operations with it. When a **SELECT** statement is fired against a table or tables, the result is compiled into a further temporary table, which is displayed or perhaps received by the program (e.g., a front-end).

PER SCHOLAS

SELECT Statement Syntax

SELECT Statement with “DQL” clause ordering:

SELECT [ALL | DISTINCT | DISTINCTROW | Columns | Expressions] clause specifies columns. to retrieve.

FROM clause specifies table(s) from which to retrieve.

WHERE clause filters rows to retrieve.

GROUP BY clause specifies how to group results.

HAVING clause selects among groups.

ORDER BY clause specifies the row ordering.

LIMIT clause specifies the limits of the retrieve records.

continue…

Example - Querying Data

Client Requirement: Janet, the VP of Marketing, has requested a list of all customer names and contacts, as well as their states, cities, and phone numbers for an outreach campaign. How can we help her?

- Use the **SELECT** keyword to write a query.

PER SCHOLAS

Example - Querying Data

Syntax:

```
SELECT field1 [, field2, field3 ...]
FROM tablename;
```

Example:

```
SELECT customerName, contactLastName, contactFirstName, city, state
FROM customers;
```

Selection of specific columns from a table is termed as a *projection operation* in relational algebra.

Querying Everything

Client Requirement: Janet realized that she needed more information. To minimize her demands on your time, she said, ***“give me everything about our customers.”***

PER SCHOLAS

Example - SELECT *

Syntax:

```
SELECT *
FROM tablename;
```

Example

```
SELECT *
FROM customers;
```

This selection must be used with care. Selecting all data from tables with millions of rows is an exercise in frustration.

To retrieve just the number of rows from a table, we can use:

```
SELECT count(*)
FROM customers;
```

Queries that return a single value are *scalar* queries.

PER SCHOLAS

Retrieving Specific Rows

Client Requirement: Janet has decided to focus her campaign on customers in “Germany.” We can refine our query results using a **WHERE** clause:

```
SELECT *
FROM customers
WHERE country='Germany';
```

Selection of specific rows from a table is termed as a *restriction operation* in relational algebra.

PER SCHOLAS

The distinct Clause

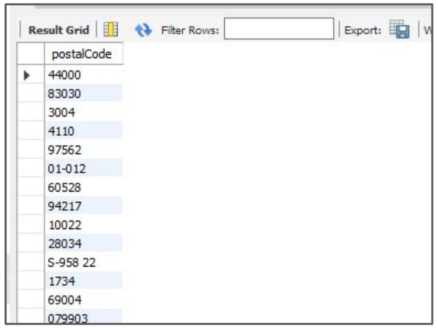
A select statement may return identical rows. To explicitly specify that you want to remove duplicate rows from the result set, use the **distinct** clause.

```
Syntax:    select    distinct    column_name    from table_name;
```

In the above syntax, the **distinct** keyword appears after the **"select"** keyword and before "columns or expressions" of the select list.

The query will return only distinct values from the "column_name" in the "table_name."

Example:
SELECT **distinct** postalCode
FROM **customers**



postalCode
44000
83030
3004
4110
97562
01-012
60528
94217
10022
28034
S-958 22
1734
69004
079903

PER SCHOLAS

SQL alias for Columns

Sometimes, column names are so technical that make the query's output very difficult to understand. To give a column a descriptive name, use a column **alias**. The following statement illustrates how to use the column **alias**:

```
SELECT
   [column_1 | expression] AS descriptive_name
FROM table_name ;
```

To assign an alias to a column, use the **AS** keyword, followed by the **alias**. *But The* **AS** *keyword is optional, so you can omit it.*

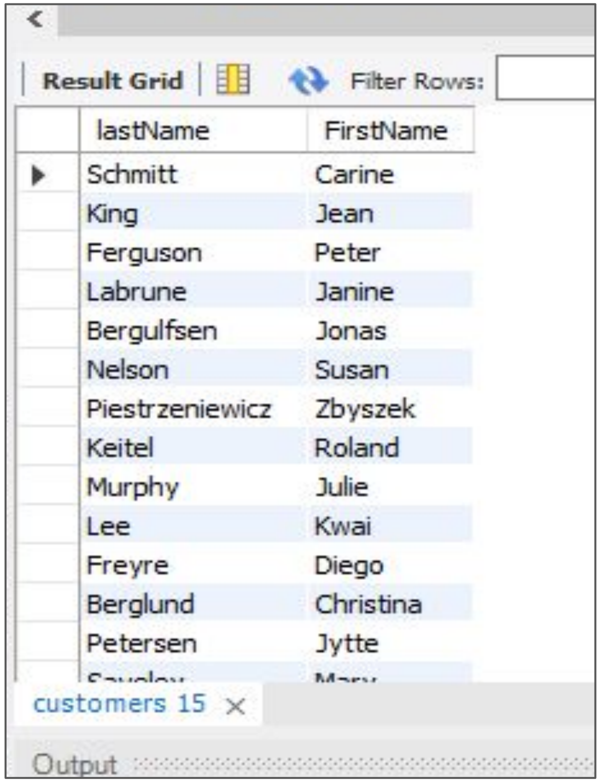
If the **alias** contains spaces, you must quote it as follows:

```
SELECT
   [column_1 | expression] AS `descriptive name`
FROM
   table_name;
```

PER SCHOLAS

Example - SQL alias for Columns

```
SELECT
    contactLastname AS lastName,
    contactFirstname As FirstName
FROM customers;
```



lastName	FirstName
Schmitt	Carine
King	Jean
Ferguson	Peter
Labrune	Janine
Bergulfsen	Jonas
Nelson	Susan
Piestrzeniewicz	Zbyszek
Keitel	Roland
Murphy	Julie
Lee	Kwai
Freyre	Diego
Berglund	Christina
Petersen	Jytte

PER SCHOLAS

SQL alias for Tables

You can assign a table an **alias** by using the **AS** keyword (optional) as the following syntax:

```
SELECT
   [column_1 | expression]
FROM table_name AS table_alias
```

Example: The query below shows how to assign the employees table **alias** as **e:**

With AS keyword:

SELECT * FROM employees **AS** **e**;

Without AS keyword:

SELECT * FROM employees **e**;

PER SCHOLAS

Example: alias for Table

Once a table is assigned an **alias**, you can refer to the columns as shown below:

```
SELECT
    e.firstName,
    e.lastName
FROM
    employees e
ORDER BY e.firstName;
```

PER SCHOLAS

Practice Assignment

1. Complete the Practice Assignment: **PA - 304.2.1 - Practice Assignment - Select Statement Queries**. You can find this assignment on Canvas under the **Practice Assignment** section.

2. Complete the following assignments by opening the links below.
   Please take note that some questions may be difficult for you because they cover advanced topics; however, we will cover such advanced topics in later presentations.

   2.1. **SELECT_Basics**

   2.2. **SELECT from WORLD**

   2.3. **SELECT from Nobel**

Note: Use your office hours to complete the assignments. If you have technical questions while performing the practice assignment, ask your instructors for assistance.

PER SCHOLAS

Data Manipulation Language

Data Manipulation Language (DML) allows you to modify the database instance by inserting, modifying, and deleting its data. DML is responsible for performing all types of data modification in a database.

Here are some important DML Statements in SQL:

- INSERT.
- UPDATE.
- DELETE.

PER SCHOLAS

INSERT Statement

The INSERT statement is used to insert data into the row of a table.

Syntax:

```
INSERT [into] tablename [(col1, col2, ...)]
VALUES ({expr | DEFAULT}, ...);
```

PER SCHOLAS

Example: INSERT Statement

Client Requirement: Janet's campaign was successful! Now ***you need to add new customers to the customers table.***

Use the **INSERT** statement to add new row(s) to a table.

Example: INSERT Statement (continued)

```
INSERT into customers VALUES
(497, 'Jenny','Jenny','Jones','917-433-2828','12 Mockingbird Lane',
NULL,'Omaha','NE','54550','USA',1056,2000.00);
```

The above statement inserts *one* row into the **customers** table.

- Strings are enclosed in single quotes.
- Numbers are given as literals.
- Null provides a missing value.

PER SCHOLAS

INSERT Statement - Bulk Records Inserts

The INSERT statement can accept more than ***one*** set of values.

```
INSERT into payments (customerNumber, checkNumber, paymentDate, amount)
VALUES
(112, 'XL55793', '2017-12-13', 1258.16),
(128, 'QQ27862', '2017-12-13', 3426.26);
```

In the above example, we have included the field-list.

Note: Acceptable date formats are vendor-specific.

PER SCHOLAS

Bulk Inserts From Another Table

Inserts can be made from another table in the database (or a different database entirely) using **INSERT… SELECT** syntax:

Step 1: Quickly create a new table for demonstration by using the query below:

```
INSERT into payments (customerNumber,
checkNumber, paymentDate, amount)
VALUES
(112, 'XL55793', '2017-12-13', 1258.16),
(128, 'QQ27862', '2017-12-13', 3426.26);
```

Step 2: The query below will copy data from payment table into the temporary_payments table.

```
INSERT into temporary_payments
SELECT * from payments
WHERE amount>2000;
```

Note: If you do not understand the above query, do not worry about it. We will demonstrate this in upcoming lectures.

PER SCHOLAS

UPDATE Statement

Existing field(s) can be modified using the **UPDATE** statement.

Example:

```
UPDATE customers
SET phone='477-931-2258'
WHERE customerNumber=497;

UPDATE customers
SET addressLine1='27 Lands End', city='San Francisco'
WHERE customerNumber=212;
```

The **WHERE** clause is important in **UPDATE** statements. If omitted, fields of ALL rows will be updated with the given value(s).

PER SCHOLAS

DELETE Statement

It is occasionally necessary to permanently remove one or more rows from a table. If so, this can be done using the **DELETE** statement.

Example

```
DELETE
FROM employees
WHERE employeeNumber=1188;
```

- The DELETE statement permanently deletes entire records matching criteria given in the **WHERE** clause.
- Use carefully: If the **WHERE** clause is omitted, all rows will be deleted.

PER SCHOLAS

Deleting All Records

All records can be deleted from a table using either **DELETE** or **TRUNCATE** statement.

```
DELETE
FROM payments_high;             -- deletes ALL rows.

TRUNCATE payments_high;         -- deletes ALL rows.
```

PER SCHOLAS

Atomicity

Databases guarantee **Atomicity**:

- When two or more DML statements execute, the database can ensure that all succeed or all fail.
- When all statements “fail,” the database remains unchanged from its initial state.
- This capability is provided using **Transactions**.

Transaction Control Language

Transaction Control Language (TCL) commands deal with the **transactions** within the database.

- There are three main statements in a Transaction block.
  - **START**: Indicates the start point of an explicit or local Transaction.
  - **COMMIT**: Commits a Transaction.
    - Changes made within a Transaction are invisible to other users of the database until the COMMIT statement is issued.
  - **ROLLBACK**: Rolls back a Transaction in case of errors.

Databases guarantee Atomicity:

- When two or more DML statements execute, the database can ensure that all succeed or all fail.
- When all statements “fail,” the database remains unchanged from its initial state.
- This capability is provided using Transactions.

PER SCHOLAS

start, rollback, and commit Statements

Syntax:

```
start transaction;
// [one or more DML statements]
rollback; or commit;
```

rollback is usually executed within a procedure’s exception handler.

commit is issued after all statements successfully execute.

PER SCHOLAS

Example: rollback Statement

```
select count(*) from employees;  -- returns 23 record
start transaction;

delete from employees where employeeNumber = 1143;     -- one row deleted
select count(*) from employees;  -- returns 22 record

rollback;
select count(*) from employees;  -- return 23 record
```

All the changes will be reverted because of the **rollback** statement (rolling back a transaction). Other users of the database will not see the changes until you *commit* the Transaction.

PER SCHOLAS

Example: commit Statement

```
select count(*) from employees;    -- returns 23 record
start transaction;
delete from employees where employeeNumber=1143;  -- one row deleted
select count(*) from employees;    -- returns 22 record
commit;
select count(*) from employees;    -- return 22 record
```

This time, the **commit** statement saves all of the modifications made in the current transaction.

PER SCHOLAS

Data Control Language

- Data Control Language (DCL) allows us to create users with more or less limited roles.
- DCL mainly deals with the rights, permissions, and other controls of the database system.
- DCL includes statements such as **GRANT** and **REVOKE**.
  - Users are added using **create user**.
  - Privileges are added using **grant**.
  - Privileges are removed using **revoke**.
  - Users are removed by **drop user**.

PER SCHOLAS

CREATE USER Statement

The **CREATE USER** statement creates one or more user accounts with no privileges. This means that the user can log into MySQL, but cannot do anything such as ***select a database*** or ***query data*** from tables.

Here is the basic syntax of the CREATE USER statement:

```
CREATE USER account_name
IDENTIFIED BY 'password';
```

In the above syntax, first, specify the account name after the **CREATE USER** statement, and then specify the password for the user after the **IDENTIFIED BY** keywords.

PER SCHOLAS

Example - Create New User

The query below will create a user named ***testUser***, with a password.

```
Create user testUser identified by 'Demopassword';
```

The above query will create a user with NO privileges. This means that the user (testUser) can log into the MySQL, but cannot do anything. We have to ***grant*** *the user privileges.*

The query below query will show existing users from the current MySQL Server:

```
select user from mysql.user
```

PER SCHOLAS

GRANT Statement

- Defines a user's privileges.
- To allow user accounts to work with ***database objects***, you need to grant the user accounts privileges. The **GRANT** statement grants a user account one or more privileges.
- The following illustrates the basic syntax of the **GRANT** statement.

```
GRANT privilege [,privilege],..
ON privilege_level
TO account_name;
```

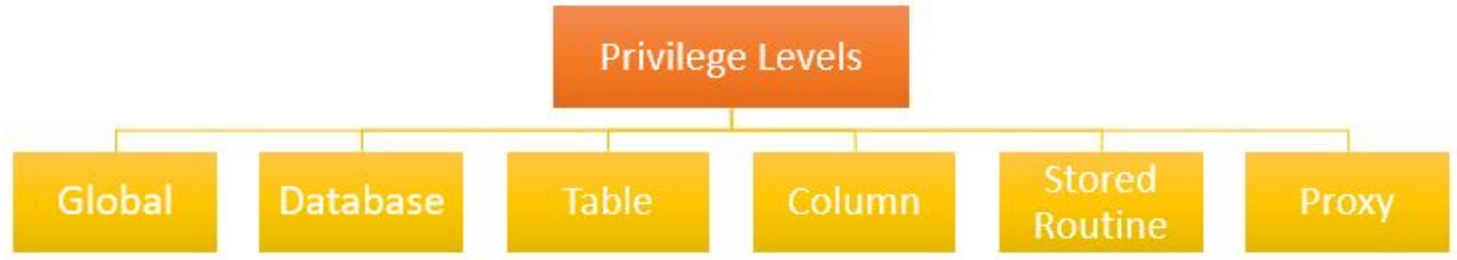
In this syntax:

First, specify one or more privileges after the **GRANT** statement. If you grant multiple privileges, you need to separate privileges by commas.

PER SCHOLAS

Privilege Level in MySQL

MySQL supports the following **main privilege** levels:



To assign global privileges, you can use the *.* For example:

```
GRANT SELECT
ON *.*
TO abcUser;
```

[To learn more about MySQL privileges, visit the Wiki document.]

PER SCHOLAS

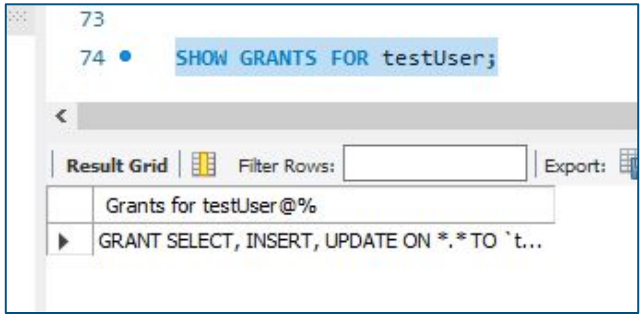
Example - Grant Statement

```
Grant select, update, insert
ON *.*
TO 'testUser';
```

The above query, Grant the account user **'testUser,'** can query data from all tables in all database of the current MySQL Server.

```
SHOW Grants FOR testUser;
```

The above query will display the granted privileges of the **'testUser'** user account:



```
SHOW GRANTS FOR testUser;
```

Result Grid | Filter Rows: | Export:

	Grants for testUser@%
▶	GRANT SELECT, INSERT, UPDATE ON *.* TO `t...

PER SCHOLAS

REVOKE Statement

The **REVOKE** statement revokes one or more privileges from a user account. Basic syntax of the **REVOKE** statement that revokes one or more privileges from user accounts:

```
REVOKE
   privilegee [,privilege]..
ON privilege_level
FROM user1 [, user2] ..;
```

PER SCHOLAS

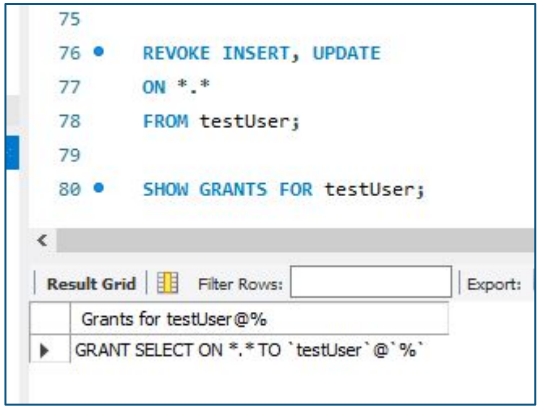
Examples - Revoke Statement

```
REVOKE INSERT, UPDATE
ON *.*
FROM testUser;
```

The above query will revoke the **UPDATE** and **INSERT** privileges from **testUser** account

```
SHOW GRANTS FOR testUser;
```

The above query will display the granted privileges of the **'testUser'** user account. You will notice that ***UPDATE*** and ***INSERT*** privileges are revoked from **testUser**.



```
REVOKE INSERT, UPDATE
ON *.*
FROM testUser;

SHOW GRANTS FOR testUser;
```

```
Grants for testUser@%
GRANT SELECT ON *.* TO `testUser`@`%`
```

PER SCHOLAS

Knowledge Check

- ❑ What is SQL? What is the full form of SQL?
- ❑ What is MySQL?
- ❑ What are the sublanguages of SQL?
- ❑ What does DML stand for?
- ❑ Can you INSERT data from one table into another table? If so, how?
- ❑ The SQL statement that queries or reads data from a table is __________ .
  a. USE
  b. SELECT
  c. READ
  d. QUERY
- ❑ What happens if you do not have a WHERE clause in an UPDATE statement?
- ❑ What is DCL? Provide an explanation of some of the commands.
- ❑ What is TCL? Provide an explanation of some of the commands.

Knowledge Check

- ❑ Which SQL statement is used to return only different values?
  a. SELECT DISTINCT
  b. SELECT UNIQUE
  c. SELECT DIFF
  d. SELECT DIFFERENT

- ❑ Which SQL statement is used to update data in a database?
  a. MODIFY
  b. ALTER
  c. SAVE
  d. UPDATE

PER SCHOLAS

Hackerrank Practice assignment

Find assignment PA - 304.2.2 - Practice Assignment - SQL on canvas under the **Assignment** section.

- This assessment will be administered through HackerRank.
- A Hackerank invitation will be required for this assignment. The instructor must coordinate with the curriculum team to create/set up an invitation link.
- It is very important assignment, this assignment will prepare you for the SBAs and KBAs.

PER SCHOLAS

Summary

Structured Query Language (SQL) is a programming language designed to get information out of and into a relational database. Queries are constructed from a command language that lets you select, insert, update, and locate data. SQL is both an American National Standards Institute (ANSI) and an International Standards Organization (ISO) standard although many databases support SQL with proprietary extensions.

The SQL commands are mainly categorized as follows:

1. DDL – Data Definition Language.
2. DQl – Data Query Language.
3. DML – Data Manipulation Language.
4. DCL – Data Control Language.

PER SCHOLAS

References

https://www.youtube.com/watch?v=7S_tz1z_5bA

https://dev.mysql.com/doc/refman/8.0/en/programs.html

https://www.techonthenet.com/mysql/index.php

PER SCHOLAS

Questions?