**Assignment - SQL Commands - Keys**

1. Create a table called "students" with the following columns: "id" (integer, primary key), "name" (text), "age" (integer), "gender" (text), and "major" (text).

Sol:

The SQL code to create the "students" table with the specified columns:

CREATE TABLE students (

id INTEGER PRIMARY KEY,

name TEXT,

age INTEGER,

gender TEXT,

major TEXT

);

**Explanation:** This will create a new table named "students" with the specified columns and data types. The "id" column is defined as the primary key, which means it will uniquely identify each row in the table. The other columns are defined with their respective data types: "name" as text, "age" as integer, "gender" as text, and "major" as text.

1. Insert a new row into the "students" table with the following values: id=1, name='John Smith', age=22, gender='Male', major='Computer Science'.

Sol:

The SQL code to insert a new row into the "students" table with the specified values:

INSERT INTO students (id, name, age, gender, major)

VALUES (1, 'John Smith', 22, 'Male', 'Computer Science', 3.5);

**Explanation:** This will insert a new row into the "students" table with the specified values. The "INSERT INTO" statement specifies the name of the table and the columns to insert data into. The "VALUES" keyword is used to specify the values to be inserted into each column. The values are provided in the same order as the column names listed in the "INSERT INTO" statement.

1. Write a SQL query to retrieve the names of all students in the "students" table.

Sol:

The SQL code to retrieve the names of all students from the "students" table:

SELECT name

FROM students;

**Explanation:** This will retrieve the "name" column for all rows in the "students" table. The "SELECT" statement is used to specify the columns to retrieve, and the "FROM" keyword is used to specify the table to retrieve data from.

1. Write a SQL query to retrieve the names and ages of all female students in the "students" table.

Sol:

The SQL code to retrieve the names and ages of all female students from the "students" table:

SELECT name, age

FROM students

WHERE gender = 'Female';

**Explanation:** This will retrieve the "name" and "age" columns for all rows in the "students" table where the "gender" column is equal to 'Female'. The "SELECT" statement is used to specify the columns to retrieve, and the "FROM" keyword is used to specify the table to retrieve data. The "WHERE" clause is used to filter the results based on a specified condition. In this case, we're only retrieving rows where the "gender" column is equal to 'Female'.

1. Write a SQL query to update the age of the student with id=1 to 23.

Sol:

The SQL code to update the age of the student with id=1 to 23:

UPDATE students

SET age = 23

WHERE id = 1;

**Explanation:** This will update the "age" column of the row with "id" equal to 1 in the "students" table to the value of 23. The "UPDATE" statement is used to modify existing data in a table, and the "SET" keyword is used to specify the column to update and the new value to set it to. The "WHERE" clause is used to specify the row to update based on a condition. In this case, we're only updating the row where the "id" column is equal to 1.

1. Create a new table called "courses" with the following columns: "id" (integer, primary key), "name" (text), and "description" (text).

Sol:

The SQL code to create a new table called "courses" with the specified columns:

CREATE TABLE courses (

id INTEGER PRIMARY KEY,

name TEXT,

description TEXT

);

**Explanation:** This will create a new table named "courses" with the specified columns and data types. The "id" column is defined as the primary key, which means it will uniquely identify each row in the table. The other columns are defined with their respective data types: "name" as text and "description" as text.

**DATABASE CREATE:-**

1. Create a database ‘classroom’

**Sol:**

Create database classroom;

1. Create a table named ‘science\_class’ with the following properties
   1. columns(enrollment\_no int, name char, science\_marks int)

**Sol:**

use classroom;

Create table science\_class(Enrollment\_no int,

name char(30),

science\_marks int);

**INSERTING &IMPORTING:-**

1. Insert the following data into science\_class using the insert into command

|  |  |  |
| --- | --- | --- |
| 1 | popeye | 33 |
| 2 | olive | 54 |
| 3 | brutus | 98 |

**Sol:**

Insert into science\_class(Enrollment\_no, name, science\_marks) values(1, 'popeye', 33), (2, 'olive', 54), (3, 'brutus', 98);

1. Import data from CSV file ‘student.csv’ attached in resources to science\_class to insert data of the next 8 students

**Sol:**

Here, we need to assume that the student.csv file has the following columns: name, age, gender, grade, email, and phone, and that the science\_class table has corresponding columns, you can import the data from the CSV file using the LOAD DATA INFILE statement. Here's an example:

LOAD DATA INFILE 'D:\SQL Modules & Assignments\Module-4\Assignments\_04\_SQL\_dataset\student.csv'

INTO TABLE science\_class

FIELDS TERMINATED BY ','

ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 ROWS;

SELECT \* FROM science\_class;

**Explanation:** This statement loads the data from the student.csv file into the science\_class table, using commas as field delimiters and double quotes as field enclosures. The IGNORE 1 ROWS option tells MySQL to skip the first row in the CSV file, which is assumed to be a header row.

After executing this statement, you can verify that the data was inserted correctly by running a SELECT statement on the science\_class table.

**SELECT & WHERE:-**

1. Retrieve all data from the table ‘Science\_Class’

**Sol:**

select \* from science\_class;

1. Retrieve the name of students who have scored more than 60 marks

**Sol:** SELECT name FROM science\_class WHERE science\_marks > 60;

1. Retrieve all data of students who have scored more than 35 but less than 60 marks

**Sol:** SELECT \* FROM science\_class WHERE science\_marks > 35 AND science\_marks < 60;

1. Retrieve all other students i.e., who have scored less than or equal to 45 or more than or equal to 60.

**Sol:**

select \* from science\_class where science\_marks<=45 and science\_marks>=60;

**UPDATING TABLES:-**

1. update the marks of popeye to 45

**Sol:**

update science\_class set science\_marks=45 where name='popeye';

1. delete the row containing details of the student named ‘robb’

**Sol:**

delete from science\_class where name='Robb';

1. Rename column ‘name’ to ‘student\_name’

**Sol:**

ALTER TABLE science\_class CHANGE COLUMN name student\_name CHAR(50);