INDEX

[PYTHON 5](#_Toc151057627)

[1. Linear Search Using List 6](#_Toc151057628)

[Code 6](#_Toc151057629)

[Output 7](#_Toc151057630)

[2. Number Guessing Game 7](#_Toc151057631)

[Code 7](#_Toc151057632)

[Output 8](#_Toc151057633)

[3. Saving Data in Binary File 9](#_Toc151057634)

[Code 9](#_Toc151057635)

[Output 10](#_Toc151057636)

[4. Saving Data in Text file 11](#_Toc151057637)

[Code 11](#_Toc151057638)

[Output 12](#_Toc151057639)

[5. Employee Details in CSV File 12](#_Toc151057640)

[Code 12](#_Toc151057641)

[Output 13](#_Toc151057642)

[6. Insertion Sort Using List 14](#_Toc151057643)

[Code 15](#_Toc151057644)

[Output 16](#_Toc151057645)

[7. Bubble Sort in Sequence 16](#_Toc151057646)

[Code 16](#_Toc151057647)

[Output 17](#_Toc151057648)

[8. Stack Implementation Using List 18](#_Toc151057649)

[Code 18](#_Toc151057650)

[Output: 19](#_Toc151057651)

[9. Queue Implementation Using List 21](#_Toc151057652)

[Code 21](#_Toc151057653)

[Output 22](#_Toc151057654)

[10. Displaying Log Table 24](#_Toc151057655)

[Code 24](#_Toc151057656)

[Output 25](#_Toc151057657)

[11. Table Creator in MySQL 25](#_Toc151057658)

[Code 25](#_Toc151057659)

[Output 27](#_Toc151057660)

[12. Updating Table in MySQL 27](#_Toc151057661)

[Code 27](#_Toc151057662)

[13. Viewing Tables in MySQL 29](#_Toc151057663)

[Code 29](#_Toc151057664)

[Output 31](#_Toc151057665)

[14. Tables Viewer in MySQL 31](#_Toc151057666)

[Code 31](#_Toc151057667)

[Output 33](#_Toc151057668)

[15. Deleteing Table in MySQL 33](#_Toc151057669)

[Code 33](#_Toc151057670)

[Output 34](#_Toc151057671)

[MySQL 36](#_Toc151057672)

[Creating Table 37](#_Toc151057673)

[1. Creating Table 37](#_Toc151057674)

[Inserting Data 37](#_Toc151057675)

[2. Inserting single Record 37](#_Toc151057676)

[3. Inserting Multiple Records 38](#_Toc151057677)

[4. Inserting Specific Parameters 39](#_Toc151057678)

[Alter Table 39](#_Toc151057679)

[5. Modifying Column 39](#_Toc151057680)

[6. Adding Column 40](#_Toc151057681)

[7. Deleting Column 40](#_Toc151057682)

[Renaming Table 41](#_Toc151057683)

[8. Renaming Table 41](#_Toc151057684)

[Updating Data 42](#_Toc151057685)

[9. Updating Whole Column 42](#_Toc151057686)

[10. Updating Single Elemet 42](#_Toc151057687)

[Viewing Table and Other Advance Statements 43](#_Toc151057688)

[11. Setting Default Values 43](#_Toc151057689)

[12. Updating Constraints 44](#_Toc151057690)

[13. Viewing Specific Columns 44](#_Toc151057691)

[14. Viewing Columns by Aliasing 45](#_Toc151057692)

[15. Deleting A Record 45](#_Toc151057693)

[Modifying the Table 46](#_Toc151057694)

[16. Inserting Values Using Default 46](#_Toc151057695)

[USING LIKE KEYWORD 46](#_Toc151057696)

[17. Format ‘%~’ 47](#_Toc151057697)

[18. Format ‘~%’ 47](#_Toc151057698)

[19. Format ‘%~%’ 47](#_Toc151057699)

[20. Format ‘\_~%’ 48](#_Toc151057700)

[21. Format ‘\_\_\_\_’ 48](#_Toc151057701)

[22. Group By Clause 49](#_Toc151057702)

[23. Having Clause 49](#_Toc151057703)

[24. Deleting All data from the Table 49](#_Toc151057704)

[25. Deleting Entire Table 50](#_Toc151057705)

PYTHON

# Linear Search Using List

## Code

|  |
| --- |
| linear\_search.py |

1 # A program for linear search.

2 def linear\_Search(sequence, search):

3 *"""*

4  *Performs linear search on the list given as input*

5  *"""*

6 for i in range(0, len(sequence)):

7 if sequence[i] == search:

8 return i

9 return -1

10

11 # Program interface

12 print("Program to search first name of CS students.")

13 names = ['Abhijeet', 'Aparna', 'Arpit', 'Harshit', 'Kajal', 'Kavya', 'Nevideta', 'Parashant']

14 search = input("Write the name of the student :")

15 result = linear\_Search(names, search)

16

17 #Returning the results.

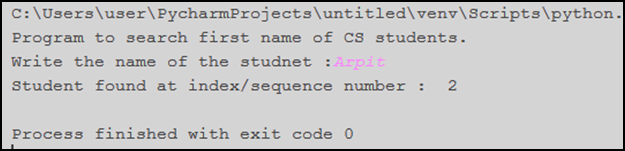
18 if result == -1:

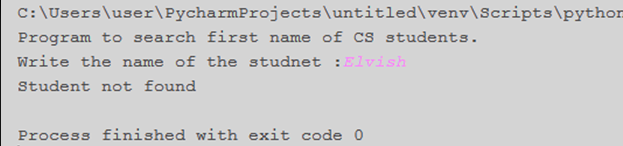
19 print("Student not found")

20 else:

21 print("Student found at index/sequence number : ", result)

## Output





# Number Guessing Game

## Code

|  |
| --- |
| number\_guessing\_game.py |

1 # A program for the use of random module.

2 import random

3

4 print("Number Guessing Game".center(120))

5 while True :

6 number = random.randint(0, 20)

7 guess = int(input("Make a guess from 0 - 20 :"))

8 # Logic for checking guesses.

9 while guess != number :

10 if guess > number :

11 guess = int(input("Guess a smaller number : "))

12 elif guess < number :

13 guess = int(input("Guess a greater number : "))

14 else:

15 print("You made a right guess!")

16

17 # Asking for replay

18 replay = input("Would you like to replay the game ?(Y or Yes if you want...)\n" + chr(9755))

19 if replay not in ["Y", "Yes"]:

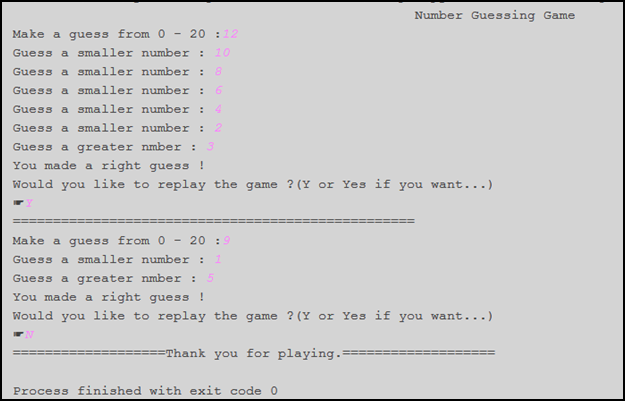
20 break

21 else :

22 print("="\*50)

23 print("Thank you for playing.".center(60, '='))

## Output



# Saving Data in Binary File

## Code

|  |
| --- |
| binary\_file.py |

1 # Using a binary file.

2 import pickle

3

4 print("A program to save the names of the students in a class.")

5 students = []

6

7 # Taking input of the list of students.

8 entries = int(input("How many students do you want to add? : "))

9 for i in range(entries):

10 name = input("Name ==>")

11 students.append(name)

12 print("ADDED")

13

14 # Confirmation for saving the file.

15 save = input("Would you like to save the data in a binary file [Press Y or Yes if you want to...] :")

16 if save in ['y', 'yes', 'Y', 'Yes']:

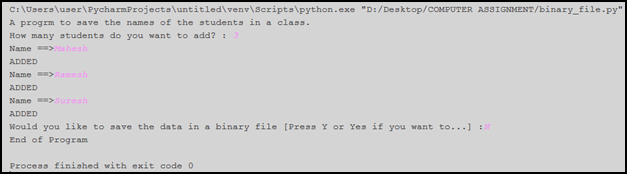
17 with open("Students\_Details.bin", "wb") as file:

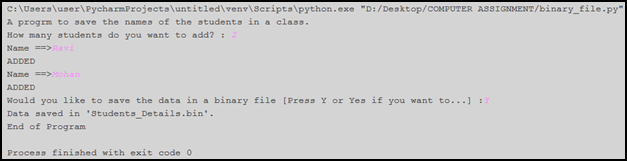
18 pickle.dump(students, file)

19 print("Data saved in 'Students\_Details.bin'.")

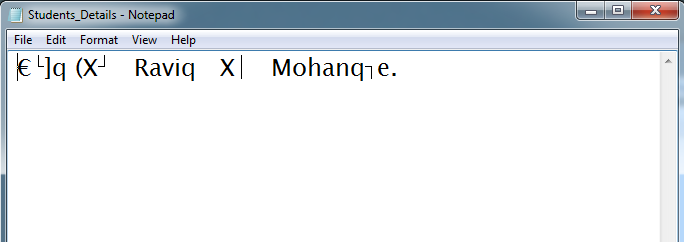
20 print("End of Program")

## Output





*This second output will also make a file named ‘Students\_details.bin’ in the cwd.*



# Saving Data in Text file

## Code

|  |
| --- |
| text\_file.py |

1 # Use of text file.

2 print("TEXT FILE WRITER")

3 data = input("Please write the data you want to save :\n" + chr(9755))

4

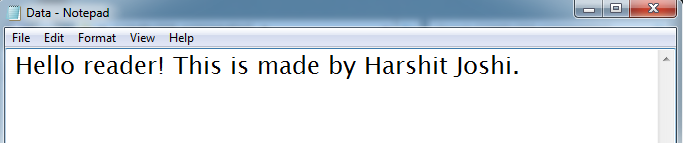
5 # Saving the data into the file.

6 with open("Data.txt", "w") as file :

7 file.write(data)

8 print("Data successfully written to Data.txt")

## Output

  
*This code will also generate a text file as follows:*

# Employee Details in CSV File

## Code

|  |
| --- |
| csv\_file.py |

1 #Use of csv file.

2 import csv

3

4

5 print("A program to record employee details.\t")

6 number\_of\_employees = int(input("How many employee details would you like to add ?"))

7

8 with open("Employees\_details.csv", "w", newline = "") as file:

9 writer = csv.writer(file)

10 # Writing header.

11 writer.writerow(["Name", "Age", "Height", "E-mail ID"])

12 print("\_"\*40)

13

14 # Adding details of the employees.

15 for i in range(number\_of\_employees) :

16 name = input("Please enter the following details...\nName of the Employee : ")

17 age = int(input("Age : "))

18 height = int(input("Height (in cm) : "))

19 e\_mail = input("E-mail : ")

20 details = [name, age, height, e\_mail]

21 writer.writerow(details)

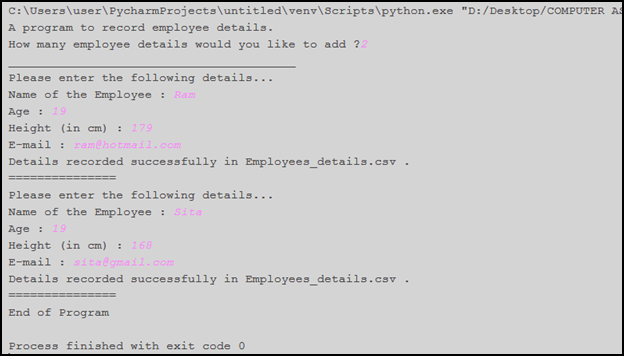
22 print("Details recorded successfully in Employees\_details.csv .")

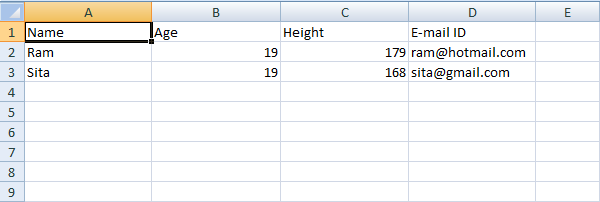
23 print("="\*15)

24

25 print("End of Program")

## Output



*The above code will also create a file, namely ‘Employees\_details.csv’.*

# Insertion Sort Using List

## Code

|  |
| --- |
| 123.py |

1 # Insertion sort.

2 def insertion\_sort(List):

3 length = len(List)

4 for i in range(1, length):

5 key = List[i]

6 j = i -1

7 # Switching elements in case they are not arranged.

8 while j >= 0 and key < List[j] :

9 List[ j+1] = List[j]

10 j -= 1

11 #Passing on to the next iteration if it is arranged.

12 else:

13 List [j+1] = key # Placing the key element at its belonging place.

14 return List

15

16 # Interface for the user

17 list\_ = []

18 elements = int(input("How many elements would you like to sort ? : "))

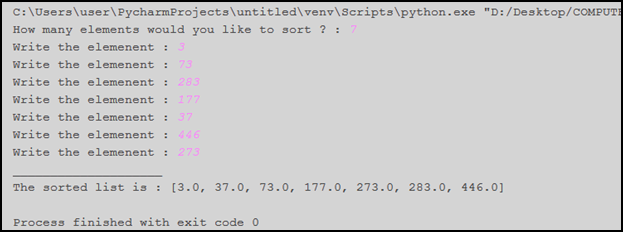
19 for i in range(elements) :

20 element = float(input("Write the elopement : "))

21 list\_.append(element)

22 print("\_"\*20 + "\nThe sorted list is :", insertion\_sort(list\_))

## Output



# Bubble Sort in Sequence

## Code

|  |
| --- |
| Bubble\_sort.py |

1 # Python program for implementation of Bubble Sort

2 def bubble\_sort(seq):

3 length = len(seq)

4

5 # Accessing all the elements of the sequence.

6 for i in range(length - 1):

7 # Accessing the adjacent elements to the original ones.

8 for j in range(0, length - i - 1):

9

10

11 # Swap elements if not arranged.

12 if seq[j] > seq[j + 1]:

13 seq[j], seq[j + 1] = seq[j + 1], seq[j]

14

15 # Running the program.

16 seq = [64, 34, 25, 12, 22, 11, 90]

17

18 print("BUBBLE SORT".center(40))

19 print("Original sequence :", seq)

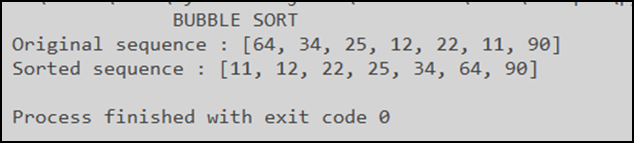
20

21 # calling the function.

22 bubble\_sort(seq)

23 print("Sorted sequence :", seq)

## Output



# Stack Implementation Using List

## Code

|  |
| --- |
| stack.py |

1 # A program for the use of stack.

2 stack = []

3 peek = int(input("What do you want the stack limit to be ? :"))

4

5 while True :

6 print("\_"\*25)

7 print("What operation would you like to perform :\n1) Append\n2) Pop\n3) View(Traverse)\n4) Exit")

8 choice = input("==>")

9

10 # For appending to the stack

11 if choice == '1' :

12 if len(stack) >= peek :

13 print("OVERFLOW")

14 else :

15 data = input("What do you want to add : ")

16 stack.append(data)

17 print("Added successfully")

18

19 # Popping elements from the stack

20 elif choice == '2' :

21 if len(stack) == 0:

22 print("UNDERFLOW")

23 else :

24 deleted = stack.pop()

25 print("Operation Successful\n", deleted, "was removed.")

26

27 # For traversing the stack.

28 elif choice == '3' :

29 print("The stack is :")

30

31 for i in stack[ : : -1] :

32 print(i, end ='\t')

33 print()

34

35 #For exiting the loop and quit.

36 elif choice == '4' :

37 print("End of program")

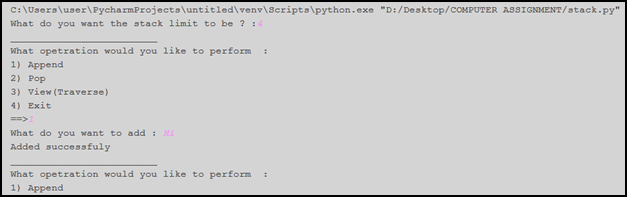
38 break

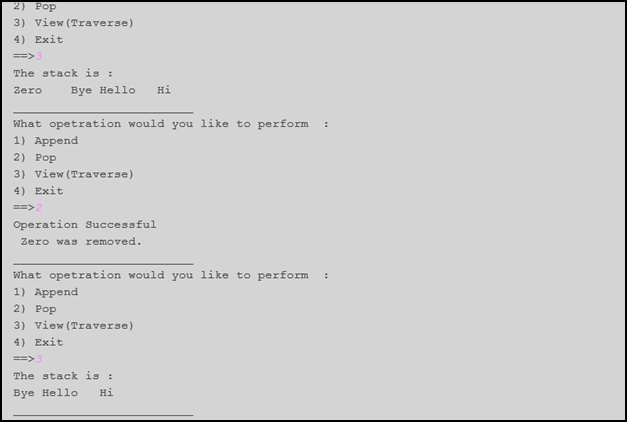
39

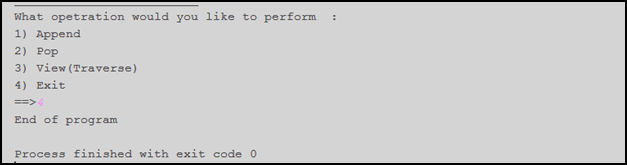
40 else :

41 print("INVALID CHOICE")

## Output:







# Queue Implementation Using List

## Code

|  |
| --- |
| queue.py |

1 # A program for the use of queue.

2 queue = []

3

4 peek = int(input("What do you want the queue limit to be ? :"))

5

6 while True :

7 print("\_"\*25)

8 print("What operation would you like to perform :\n1) Append\n2) Pop\n3) View(Traverse)\n4) Exit")

9 choice = input("==>")

10

11 # For appending an element

12 if choice == '1' :

13 if len(queue) >= peek :

14 print("OVERFLOW".center(15, '\*'))

15 else :

16 data = input("What do you want to add : ")

17 queue.append(data)

18 print("Added successfully")

19

20 # For removing an element from the queue

21 elif choice == '2' :

22 if len(queue) == 0:

23 print("UNDERFLOW".center(15, '\*'))

24 else :

25 deleted = queue.pop(0)

26 print("Operation Successful\n", deleted, "was removed.")

27

28 # To view the queue

29 elif choice == '3' :

30 print("The queue is :")

31 for i in queue :

32 print(i, end ='\t')

33 print()

34

35 # To exit the program

36 elif choice == '4' :

37 print("End of Program")

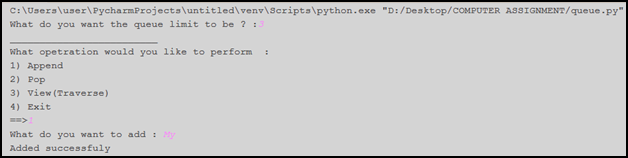
38 break

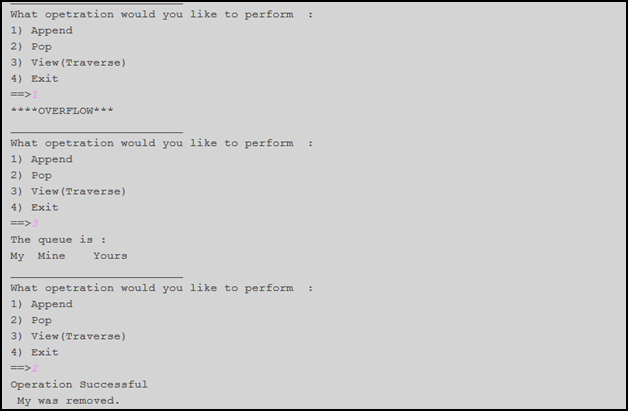
39

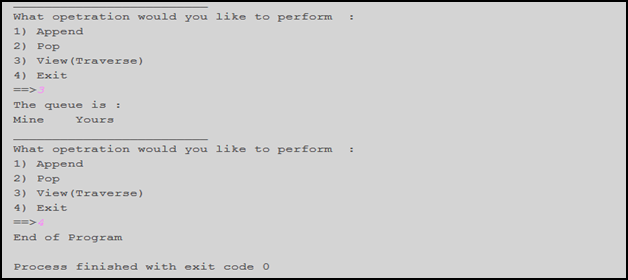
40 else :

41 print("INVALID CHOICE")

## Output







# Displaying Log Table

## Code

|  |
| --- |
| logtable.py |

1 *"""*

2 *# log table making program till desired number*

3 *"""*

4

5 import math

6 print("----- LOG TABLE GENERATOR -----")

7

8 ask\_range = input("Enter the range of the log table ")

9

10 while ask\_range.isnumeric() is False:

11 # if the range is not a number

12 ask\_range = input("Enter the range of the log table ")

13

14 # print the header

15 print("---x---------------------------log10-------------------loge-------------------log2")

16 for number in range(1, int(ask\_range) + 1):

17 # calculating the log of numbers

18 log10 = round(math.log(number, 10), 4) # base 10

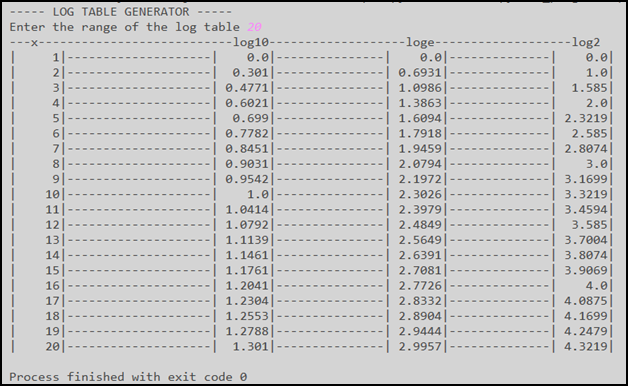
19 loge = round(math.log(number, math.e), 4) # base e natural lo

20 log2 = round(math.log(number, 2), 4) # base 2

21 # print the result row wise

22 print(f"|{number:6}|--------------------|{log10:7}|---------------|{loge:7}|--------------|{log2:7}|")

## Output



# Table Creator in MySQL

## Code

|  |
| --- |
| table\_creator.py |

1 import mysql.connector

2 try:

3 # Establishing connection

4 mycon = mysql.connector.connect(host='localhost', user='root', database='mysql', passwd='')

5 except:

6 print("An error occurred.")

7 else:

8 # In case no error occurs this block executes

9 cursor = mycon.cursor()

10 try:

11 file = open("created.txt", 'x')

12 except:

13 pass

14 else: #Create the table student if it doesn’t already exist.

15 cursor.execute("create table student(roll\_no int unique, name varchar(25))")

16 # Code for making entries.

17 entries = int(input("How many entries do you want to make? :"))

18 for i in range(entries):

19 roll\_no = input("Write your roll number :")

20 name = input("Write your name :")

21 cursor.execute("insert into student values("+roll\_no+",'"+name+"')")

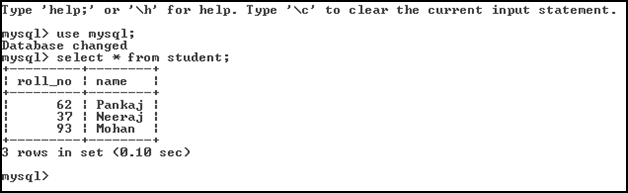
22 mycon.commit()

23 mycon.close()

24 print("Task completed")

## Output



*This program will also create a MySQL relation as follows*

# Updating Table in MySQL

## Code

|  |
| --- |
| Mysql\_data\_updating.py |

1 *"""*

2 *# adding data to given mysql table with defined attributes*

3 *"""*

4

5 import mysql.connector # getting the connection to MySQL

6

7

8 ask\_password = input("Enter the Password of the database ") # asking the database password

9 ask\_database = input("Enter the database of the table ") # asking the database

10 try:

11 cnx = mysql.connector.connect(user='root', passwd=ask\_password, database=ask\_database)

12 # initiating the connection

13 cursor = cnx.cursor()

14

15 # asking the details required for the table:music

16 ask\_song = input("Enter the song name ").strip()

17 ask\_artist = input("Enter the artist name ").strip().title()

18 ask\_album = input("Enter the album name ").strip()

19

20 # executing the commands in mysql

21 cursor.execute(f"insert into music values({ask\_artist!r}, {ask\_album!r}, {ask\_song!r})")

22

23 # committing the changes in the sql

24 cnx.commit()

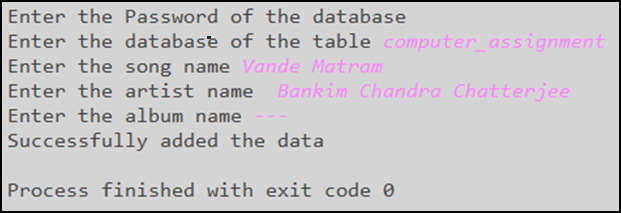
25 print("Successfully added the data ")

26 # printing the success message

27 except mysql.connector.errors.ProgrammingError:

28 print("The password or database is wrong.")

29 # if the following error is encountered print this message





# Viewing Tables in MySQL

## Code

|  |
| --- |
| mysqldisplay.py |

1 *"""*

2 *using the mysql-connector to display the file*

3 *"""*

4

5 import mysql.connector # for MySQL connection

6

7 ask\_pass = input("Enter MySQL password :")

8 # asking the password for the MySQL

9 ask\_database = input("Enter the database :")

10 # asking the database for tables

11 try:

12 cnx = mysql.connector.connect(user='root', passwd=ask\_pass, database=ask\_database)

13 # setting up a connection to the MySQL

14 cursor = cnx.cursor()

15 # asking for the table

16 ask\_table = input("Enter the table ")

17

18 # executing the MySQL query

19 cursor.execute(f"select \* from {ask\_table}")

20 # for new line

21 print()

22 # traversing over the cursor

23 for lines in cursor:

24 for words in lines:

25 # printing the data fetched by the cursor

26 print(words, '\t\t', end='')

27 print()

28 # for new line

29

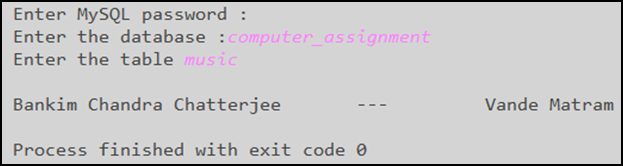
30

31 except mysql.connector.errors.ProgrammingError:

32 print("Wrong password or database does not exists ")

33 # if the following error occurs then print this message

## Output



# Tables Viewer in MySQL

## Code

|  |
| --- |
| mysqltablesearch.py |

1 *"""*

2 *searching the table with mysql*

3 *"""*

4

5 import mysql.connector # connecting to the mysql database

6

7 try:

8 ask\_pass = input("Enter MySQL password :")

9 ask\_database = input("Enter the database :")

10 cnx = mysql.connector.connect(user='root', passwd=ask\_pass, database=ask\_database)

11

12 cursor = cnx.cursor()

13 cursor.execute('show tables')

14

15 lines = cursor.fetchall()

16

17 if len(lines) == 0:

18 print("There are no tables in the database.")

19

20 else:

21 # traversing over the cursor

22 for line in lines:

23 # printing the data fetched by the cursor

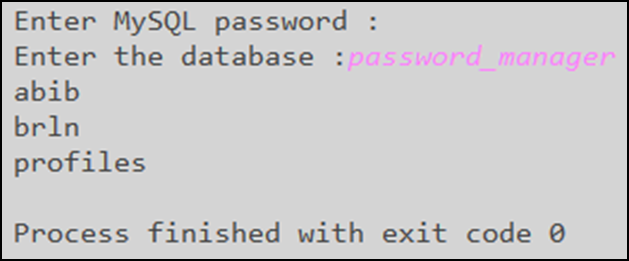
24 print(line[0])

25

26 except mysql.connector.errors.ProgrammingError:

27 print("The password or the database is wrong")

## Output



# Deleteing Table in MySQL

## Code

|  |
| --- |
| mysqldeletetables.py |

1 *"""*

2 *using the mysql connector to delete tables*

3 *"""*

4

5 import mysql.connector

6

7 # Taking input from the user.

8 ask\_pass = input("Enter MySQL password : ")

9 ask\_database = input("Enter the database : ")

10

11 # Establishing connection

12 try:

13 cnx = mysql.connector.connect(user='root', passwd=ask\_pass, database=ask\_database)

14 cursor = cnx.cursor()

15 table\_name = input("Enter the table name ")

16 cursor.execute(f"drop table if exists {table\_name}")

17 conform = input(f"Completely delete the table {table\_name!r} ").strip().casefold()

18

19 # Deleting the table fully.

20 if conform in ['yes', 'y', 'yep']:

21 cnx.commit()

22 print(f"the table {table\_name} is permanently deleted")

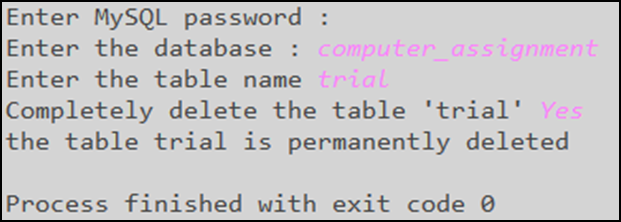
23 else:

24 print("The table is not deleted")

25 except ValueError:

26 print("The password or the database is wrong ")

## Output



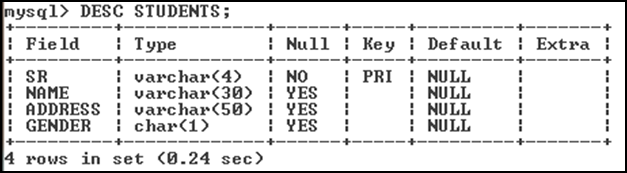
MySQL

# Creating Table

CREATE TABLE <table\_name> (<att1> <datatype> <constraint>, <att2> …);

## Creating Table

D:\Desktop\Untitled.png

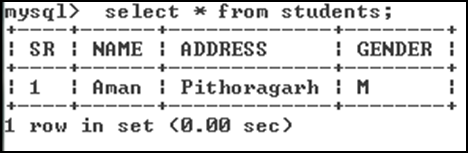


# Inserting Data

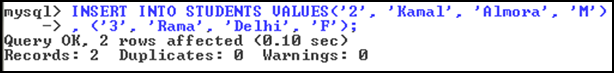
INSERT INTO <table\_name> [(<att1>, <att2>, …)] VALUES (<value1>, <value2>, …)[, (<value1>, <value2>, …), …];

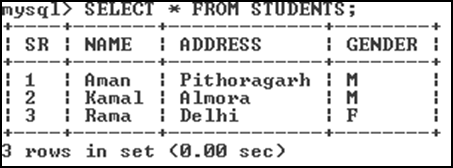
## Inserting single Record

D:\Desktop\Untitled.png



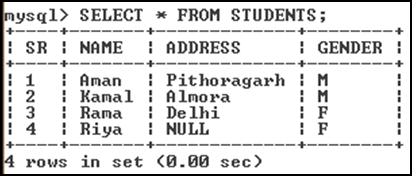
## Inserting Multiple Records





## Inserting Specific Parameters



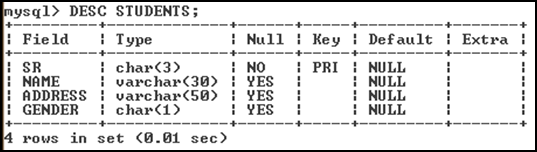


# Alter Table

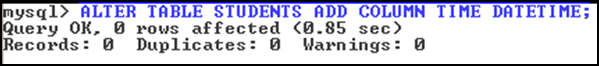
ALTER TABLE <table\_name> <modify/add/drop> COLUMN <column\_name> <new\_datatype>;

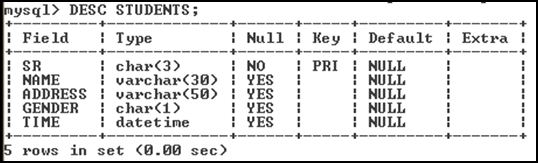
## Modifying Column

D:\Desktop\Untitled.png



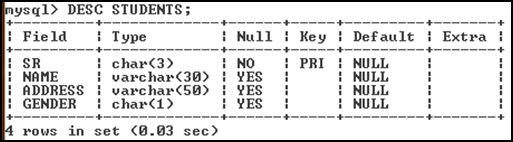
## Adding Column





## Deleting Column

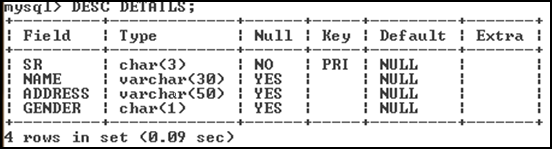




# Renaming Table

## Renaming Table

D:\Desktop\Untitled.png

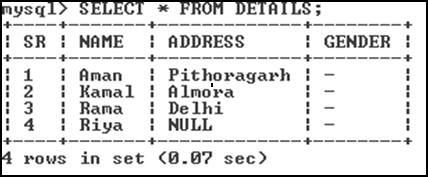


# Updating Data

UPDATE <table\_name> SET <column\_name> = <new\_value> [WHERE <condition>];

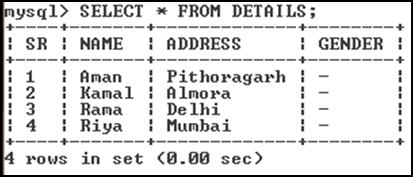
## Updating Whole Column





## Updating Single Elemet

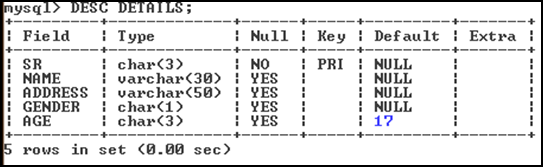




# Viewing Table and Other Advance Statements

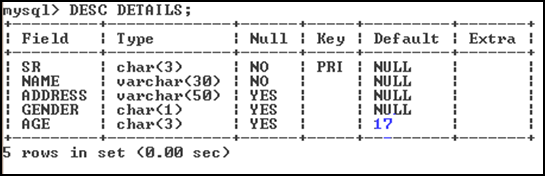
## Setting Default Values



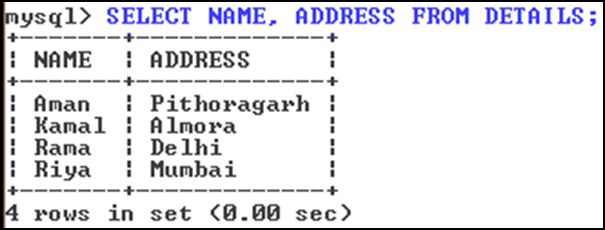


## Updating Constraints

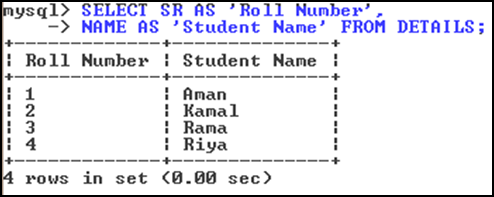




## Viewing Specific Columns



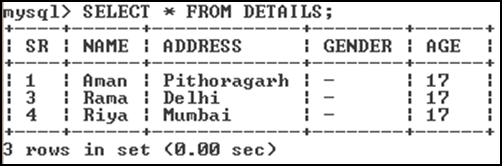
## Viewing Columns by Aliasing



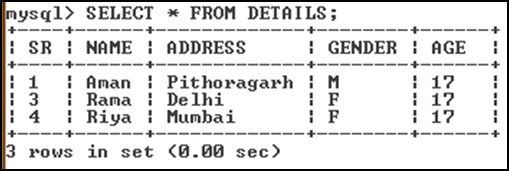
## Deleting A Record

DELETE FROM <able\_name> [WHERE <condition>];

D:\Desktop\Untitled.png

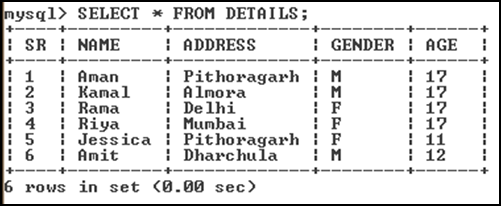


## Modifying the Table



## Inserting Values Using Default

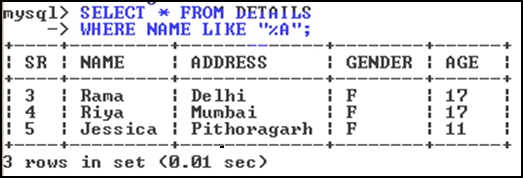




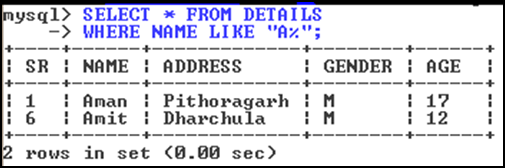
## USING LIKE KEYWORD

SELECT <attributes> FROM <table\_name> WHERE <attribute> LIKE <condition>;

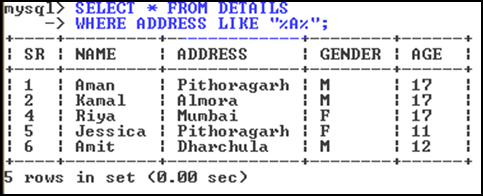
### Format ‘%~’



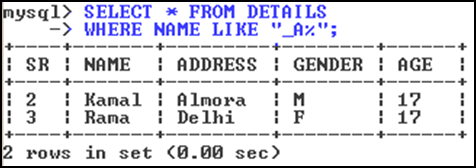
### Format ‘~%’



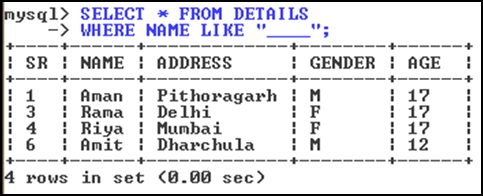
### Format ‘%~%’



### Format ‘\_~%’

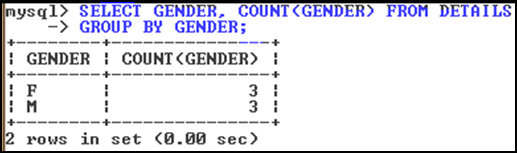


### Format ‘\_\_\_\_’



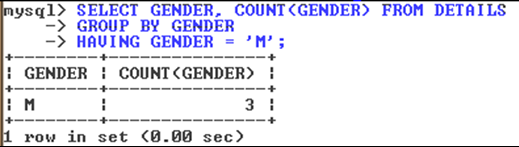
## Group By Clause

SELECT <agg\_function> [<att\_name>] FROM <table\_name> GROUP BY <att\_name>;



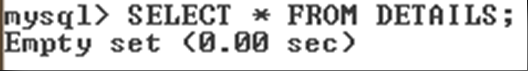
## Having Clause

SELECT <agg\_function> [<att\_name>] FROM <table\_name> GROUP BY <att\_name> HAVING <condition>;



## Deleting All data from the Table





## Deleting Entire Table

DROP TABLE <table\_name>;



