Sunbeam School, Varuna, Varanasi

Estd. 1972

COMPUTER SCIENCE (083)

Practical File:
Python & MySQL

Session: 2020-2021

Under the Guidance of:

Mrs. Ruhi Seth

Made by:

Tanishq Singh
Class: XII - C
Roll no:

PROGRAM: 01

```
# Python code to find largest prime
# factor of number
import math
# A function to find largest prime factor
def maxPrimeFactors (n=12345):
    # Initialize the maximum prime factor
    # variable with the lowest one
        maxPrime = -1
    # Print the number of 2s that divide n
      while n % 2 == 0:
            maxPrime = 2
             # equivalent to n /= 2
    # n must be odd at this point,
    # thus skip the even numbers and
    # iterate only for odd integers
            for i in range(3, int(math.sqrt(n)) + 1, 2):
                  while n % i == 0:
                        maxPrime = i
                        n = n / i
    # This condition is to handle the
    # case when n is a prime number
    # greater than 2
            if n > 2:
                  maxPrime = n
    return int(maxPrime)

maxPrimeFactors ()
```

PROGRAM: 02

```
# Python program to print prime factors
import math
# A function to print all prime factors of
# a given number n
def primeFactors(n):
    # Print the number of two\'s that divide n
        while n % 2 == 0:
                print (2)
                n = n / 2
    # n must be odd at this point
    # so a skip of 2 ( i = i + 2) can be used
            for i in range(3,int(math.sqrt(n))+1,2):
    # while i divides n , print i ad divide n
                while n % i== 0:
                        print (i)
                        n = n / i
    # Condition if n is a prime
    # number greater than 2
            if n > 2:
                    print (n)
n = 315
primeFactors(n)
```

PROGRAM: 03

```
# Python program to find sum of given
# series.
def productPrimeFactors(n):
    product = 1
    for i in range(2, n+1):
            if (n % i == 0):
                    isPrime = 1
            for j in range(2, int(i/2 + 1)):
                    if (i % j == 0):
                            isPrime = 0
                            break
            # condition if \'i\' is Prime number
                    # as well as factor of num
                     if (isPrime):
                            product = product * i
    return product
# main()
n = 44
print (productPrimeFactors(n))
```

PROGRAM: 04

```
# Python program to find sum of all divisors
# of n.
import math
# Returns sum of all factors
# of n.
def sumofoddFactors( n ):
    # Traversing through all
        # prime factors.
            res = 1
    # ignore even factors by
        # of 2
            while n % 2 == 0:
                    n = n // 2
    for i in range(3, int(math.sqrt(n) + 1)):
        # While i divides n, print
                # i and divide n
                        count = 0
                        curr_sum = 1
                        curr_term = 1
                        while n % i == 0:
                                count+=1
            n = n // i
                        curr_term *= i
                                curr_sum += curr_term
        res *= curr_sum
    # This condition is to
        # handle the case when
            # n is a prime number.
                if n >= 2:
                        res *= (1 + n)
    return res
# Driver code
n = 30
print(sumofoddFactors(n))
```

PROGRAM: 05

```
# Python program to solve coin
# Change problem
def count(S, m, n):
    # We need n+1 rows as the table is constructed
        # in bottom up manner using the base case 0 value
            # case (n = 0)
                table = [[0 for x in range(m)] for x in range(n+1)]
    # Fill the entries for 0 value case (n = 0)
        for i in range(m):
                table[0][i] = 1
    # Fill rest of the table entries in bottom up manner
        for i in range(1, n+1):
                for j in range(m):
            # Count of solutions including S[j]
                        x = table[i - S[j]][j] if i-S[j] >= 0 else 0
            # Count of solutions excluding S[j]
                        y = table[i][j-1] if j >= 1 else 0
            # total count
                        table[i][j] = x + y
    return table[n][m-1]
# Driver program to test above function
arr = [1, 2, 3]
m = len(arr)
n = 4
print(count(arr, m, n))
```

PROGRAM: 06

```
# Python function to solve the tower of hanoi
def TowerOfHanoi(n , source, destination,
auxiliary):
    if n==1:
            print "Move disk 1 from source",source,"to
destination",destination
                    return
                        TowerOfHanoi(n-1, source, auxiliary,
destination)
                            print "Move disk",n,"from
source",source,"to destination",destination
                            TowerOfHanoi(n-1, auxiliary,
destination, source)
# Driver code
n = 4
TowerOfHanoi(n,'A','B','C')
# A, C, B are the name of rods
```

PROGRAM: 07

```
# Python program to find minimum
# sum of product of number
# To find minimum sum of
# product of number
def findMinSum(num):
    sum = 0
    # Find factors of number
        # and add to the sum
     i = 2
     while(i * i <= num):
          while(num % i == 0):
                sum += i
                num /= i
                i += 1
                sum += num
    # Return sum of numbers
        # having minimum product
    return sum
# Driver Code
num = 12
print findMinSum(num)
```

PROGRAM: 08

```
# Python program to find all pairs in
# a list of integers with given sum
def findPairs(lst, K):
    res = []
    while lst:
        num = lst.pop()
        diff = K - num
        if diff in lst:
            res.append((diff, num))
    res.reverse()
    return res
# Driver code
lst = [1, 5, 3, 7, 9]
K = 12
print(findPairs(lst, K))
```

PROGRAM: 09

```
# Python program to check if difference
between sum of
# odd digits and sum of even digits is 0
or not
def isDiff(n):
    return (n % 11 == 0)
# Driver code
n = 1243;
if (isDiff(n)):
    print("Yes")
    else:
        print("No")
```

PROGRAM: 10

```
# Python program to print the number
# series without using loop
def PrintNumber(N, Original, K, flag):
    #print the number
        print(N, end = " ")
    # change flag if number
        # become negative
    if (N <= 0):
            if(flag==0):
                flag = 1
            else:
                flag = 0
    # base condition for
        # second_case (Adding K)
    if (N == Original and (not(flag))):
            return
    # if flag is true
        # we subtract value until
        # number is greater then zero
    if (flag == True):
            PrintNumber(N - K, Original, K, flag)
                return
    # second case (Addition )
        if (not(flag)):
                PrintNumber(N + K, Original, K, flag)
                return
N = 20
K = 6
PrintNumber(N, N, K, True)
```

PROGRAM: 11

```
# Python program to demonstrate
# Test if all digits starts from % K digit
# using list comprehension + map()
# initializing list
test_list = [65, 3, 92, 332]
    # printing original list
print("The original list : " + str(test_list))
    # initializing K
K = 3
    # using list comprehension + map()
    # Test if all digits starts from % K digit
res = len(set( not(int(sub[0]) % K) for sub
in map(str, test_list))) == 1
    # print result
print("Does each element start with % K
digit ? " + str(res))
```

PROGRAM: 12

```
# Python3 code to demonstrate working of
# Elements with specific digits
# Using list comprehension + all()
# initializing list
test_list = [345, 23, 128, 235, 982]
# printing original list
print("The original list is : " + str(test_list))
# initializing digit list
dig_list = [2, 3, 5, 4]
# checking for all digits using all()
res = [sub for sub in test_list if all(int(ele) in
dig_list for ele in str(sub))]
# printing result
    print("Extracted elements : " + str(res))
```

PROGRAM: 13

```
#Python program to reverse the content of a file and store it in
another file

# Open the file in write mode

f1 = open("output1.txt", "w")
# Open the input file and get
# the content into a variable data

with open("file.txt", "r") as myfile:

    data = myfile.read()
# For Full Reversing we will store the
# value of data into new variable data_1
# in a reverse order using [start: end: step],
# where step when passed -1 will reverse
# the string

data_1 = data[::-1]
# Now we will write the fully reverse
# data in the output1 file using
# following command
f1.write(data_1)
f1.close()
```

PROGRAM: 14

```
# Python program for reversing the order of
lines
# Open the file in write mode

f2 = open("output2.txt", "w")
# Open the input file again and get
# the content as list to a variable data

with open("file.txt", "r") as myfile:

    data = myfile.readlines()
# We will just reverse the
# array using following code

data_2 = data[::-1]
# Now we will write the fully reverse
# list in the output2 file using
# following command
f2.writelines(data_2)
f2.close()
```

PROGRAM: 15

```
#Python program to append
#content of one text file to another
# entering the file names
firstfile = input("Enter the name of first file ")
secondfile = input("Enter the name of second file ")
    # opening both files in read only mode to read initial contents
f1 = open(firstfile, 'r')
f2 = open(secondfile, 'r')
    # printing the contens of the file before appending
print('content of first file before appending -', f1.read())
print('content of second file before appending -', f2.read())
    # closing the files
f1.close()
f2.close()
    # opening first file in append mode and second file in read mode
f1 = open(firstfile, 'a+')
f2 = open(secondfile, 'r')
    # appending the contents of the second file to the first file
f1.write(f2.read())
    # relocating the cursor of the files at the beginning
f1.seek(0)
f2.seek(0)
    # printing the contents of the files after appendng
print('content of first file after appending -', f1.read())
print('content of second file after appending -', f2.read())
    # closing the files
f1.close()
f2.close()
```

PROGRAM: 16

```
# Python program to modify the
# content of binary file


# Function to update the
# content of binary file

def update_binary(word, new)

    # string variable to store
    # each word after reading
    # from the file
    string = b""

    # Flag variable to check
    # if the record is found or
    # not
    Flag = 0

    # Open the file in r + b mode which means
    # opening a binary file for reding and
    # writing
    with open('file.txt', 'r + b') as file:

        pos = 0

        # Reading the content of the
        # file character by character
        data = string = file.read(1)

        # Looping till the end of
        # file is reached
        while data:
            data = file.read(1)

            # Checking if the space is reached
            if data == b" ":
```

```
            # checking the word read with
            # the word entered by user
            if string == word:

                # Mpving the file pointer
                # at the end of the previously
                # read record
                file.seek(pos)

                # Updating the content of the file
                file.write(new)
                Flag = 1
                break
            else:
                # storing the position of
                # current file pointer i.e. at
                # the end of previously read record
                pos = file.tell()
                data = string = file.read(1)
        else:

            # Storing the data of the file
            # in the string variable
            string += data
            continue


    if Flag:
        print("Record successfully updated")
    else:
        print("Record not found")

# Dirver's code
# Input the word to be found
# and the new word

word = input("Enter the word to be replaced: ").encode()

new = input("\nEnter the new word: ").encode()

update_binary(word, new)
```

PROGRAM: 17

```
# Python program to split array and move first
# part to end.
def splitArr(arr, n, k):
    for i in range(0, k):
            x = arr[0]
                for j in range(0, n-1):
                            arr[j] = arr[j + 1]
        arr[n-1] = x
# main
arr = [12, 10, 5, 6, 52, 36]
n = len(arr)
position = 2
splitArr(arr, n, position)
for i in range(0, n):
    print(arr[i], end = ' ')
```

PROGRAM: 18

```
# Python program to search all
# anagrams of a pattern in a text
MAX = 256
# This function returns true
# if contents of arr1[] and arr2[]
# are same, otherwise false.
def compare(arr1, arr2):
    for i in range(MAX):
            if arr1[i] != arr2[i]:
                    return False
            else:
                    return True
            # This function search for all
            # permutations of pat[] in txt[]
def search(pat, txt):
    M = len(pat)
    N = len(txt)
    # countP[]:  Store count of
    # all characters of pattern
    # countTW[]: Store count of
    # current window of text
```

```
    countTW = [0]*MAX
    for i in range(M):
            (countP[ord(pat[i]) ]) += 1
                    (countTW[ord(txt[i]) ]) += 1
    # Traverse through remaining
    # characters of pattern
            for i in range(M, N):
        # Compare counts of current
        # window of text with
        # counts of pattern[]
                                if compare(countP, countTW):
                                        print("Found at Index", (i-M))
        # Add current character to current window
                (countTW[ ord(txt[i]) ]) += 1
        # Remove the first character of previous window
                (countTW[ ord(txt[i-M]) ]) -= 1
    # Check for the last window in text
        if compare(countP, countTW):
                print("Found at Index", N-M)
# Driver program to test above function
txt = "BACDGABCDA"
pat = "ABCD"
search(pat, txt)
```

PROGRAM: 19

```
# Python program to remove all duplicates
# and permutations in nested list
#Initialisation
listOfPermut = [[-11, 0, 11], [-11, 11, 0],
                [-11, 0, 11], [-11, 2, -11],
                [-11, -11, 2], [2, -11, -11]]
# Sorting tuple then removing
output = set(map(lambda x:
tuple(sorted(x)),listOfPermut))
# printing output
print(output)
```

PROGRAM: 20

```
# Python code to demonstrate
# sorting and removal of duplicates

# Using sorted() + set() + count()

# initializing list
test_list = [5, 6, 2, 5, 3, 3, 6, 5, 5, 6, 5]

# printing original list
print("The original list : " + str(test_list))

# using sorted() + set() + count()
# sorting and removal of duplicates
res = sorted(set(test_list), key = lambda ele:
test_list.count(ele))

# print result
print("The list after sorting and removal : " +
str(res))
```

PROGRAM: 21

```
#Python Program to
#Find the LCM of
#Two Numbers

#initilaizing variables and taking input
a=int(input("Enter the first number:"))
b=int(input("Enter the second number:"))

if(a>b):
   min1=a
else:
   min1=b

#using while loop
while(1):
   if(min1%a==0 and min1%b==0):
      print("LCM is:",min1)
      break
   min1=min1+1
```

PROGRAM: 22

```
# Python program to
# demonstrate stack implementation
# using list
stack = []

    # append() function to push
    # element in the stack
stack.append('a')
stack.append('b')
stack.append('c')
print('Initial stack')
print(stack)

    # pop() function to pop
    # element from stack in
    # LIFO order
print('\nElements poped from stack:')
print(stack.pop())
print(stack.pop())
print(stack.pop())
print('\nStack after elements are poped:')
print(stack)

# uncommenting print(stack.pop())
# will cause an IndexError
# as the stack is now empty
```

PROGRAM: 23

```
#Program to read a file story.txt and print
the
#contents of file along with
#number of vowels present in it

f=open("story.txt","r")
st=f.read()
print("Contents of file :")
print(st)
C=0
v=['a', 'e','','o', 'u')
for i in st:
    if i.lower() in v:
            C=C+1
print("*****FILE END*****)
print()
print("Number of vowels in the file =",c)
f.close()
```

PROGRAM: 24

```
#Program to input a list of integers and search for
#a given number using binary search

def bsearch(L, n):
    start-0
    end=len(L)-1
    while start=end:
        mid=(start+end )//2
        if L[mid]==n:
            return True
        elif mide=n:
            start=mid+1
        else:
            end=mid-1
    else:
        return False

L=eval(input("Enter the list of numbers")
n=int(input("Enter the number to find"))
Lsort()
if bsearch(Ln):
    print("Element found")
else:
    print("Element not found")
```

PROGRAM: 25

```
#Program for writing multiple rows in csv file
import csv
row_list = [["SN", "Name", "Contribution"],
[1, "Linus Torvalds", "Linux Kernel"],
[2, "Tim Berners-Lee", "World Wide Web"],
[3, "Guido van Rossum", "Python Programming"]]

#opening and writing in a file
with open('protagonist.csv', 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerows(row_list)
```

PROGRAM: 26

```
#Python program to make a function that fetches words

import random

WORDLIST = 'wordlist.txt'

def get_random_word(min_word_length):
#Get a random word from the wordlist
#using no extra memory
    num_words_processed = 0
    curr_word = None
    with open(WORDLIST, 'r') as f:
        for word in f:
            if '(' in word or ')' in word:
                continue
            word = word.strip().lower()
            if len(word) < min_word_length:
                continue
            num_words_processed += 1
            if random.randint(1, num_words_processed) == 1:
                curr_word = word
    return curr_word
```

PROGRAM: 27

```
#Python program to make functions that handles all the
operations on stacks

#Checking whether the stack is empty or not
def isEmpty(stk):
    if stk==[]:
        return True
    else:
        return False

#Adding values to a stack
def push(stk,item):
    stk.append(item)
    top=len(stk)-1

#Deleting values from the stack in non empty
def Pop(stk):
    if isEmpty(stk):
        return “underflow”
    else:
        item=stk.pop()
        if len(stk)==0:
            top=none
        else:
            top=len(stk)-1
    return item
```

```
#Displaying the peek value of the stack
def peek(stk):
    if isEmpty(stk):
        return “underflow”
    else:
        top=len(stk)-1
    return stk[top]

#Displaying the whole stack
def display(stk):
    if isEmpty(stk):
        print(“stack empty”)
    else:
        top=len(stk)-1
        for a in range(top,-1,-1):
            print(stk[a])
```

PROGRAM: 28

```
#Python program to do all operations on stacks
#_main_
stack=[]
top=None
while True:
    print("Stack operations")
    print("1. Push")
    print("2. Pop")
    print("3. Peek")
    print("4. Display")
    print("5.Exit")
    ch=int(input("Enter your choice"))
    if ch==1:
        item=int(input("Enter item"))
        push(stack,item)
    elif ch==2:
        item=pop(stack)
            if item =="Underflow":
                print("Stack is empty!")
            else:
                print("Popped item is",item)
    elif ch==3:
        item=peek(stack)
        if item== "Underflow":
            print("Stack is empty")
        else:
            print(item)
    elif ch==4:
        display(stack)
    elif ch==5:
        break
    else:
        print("Invalid choice")
```

29: MySQL - I

1) Creating and using Database:

```
>>> CREATE DATABASE tanishq;
>>> USE tanishq;
```

2) Creating Table:

```
>>> CREATE TABLE hackers (hack_id INT PRIMARY KEY, hacker
VARCHAR (20), hacks INT, h_accuracy FLOAT);

>>> CREATE TABLE crackers (crack_id INT PRIMARY KEY, cracker
VARCHAR (20), cracks INT, c_accuracy FLOAT);
```

3) Inserting Values to the table:

```
>>> INSERT INTO hackers VALUES (1, 'Tanishq', 99, 95.7);
>>> INSERT INTO hackers VALUES (2, 'Biden', 72, 47.1);
>>> INSERT INTO hackers VALUES (3, 'Shawn', 82, 39.5);
>>> INSERT INTO hackers VALUES (4, 'Devid', 65, 55.0);
>>> INSERT INTO hackers VALUES (5, 'Lucy', 85, 62.9);

>>> INSERT INTO crackers VALUES (1, 'Swedin', 100, 28.9);
>>> INSERT INTO crackers VALUES (2, 'Kwarty', 828, 81.9);
>>> INSERT INTO crackers VALUES (3, 'Merry', 92, 62.2);
>>> INSERT INTO crackers VALUES (4, 'Gustor', 728, 68.0);
>>> INSERT INTO crackers VALUES (5, 'Robinate', 98, 42.4);
```

4) Showing all records in a table hackers:

```
>>> SELECT * FROM hackers;
>>> SELECT * FROM crackers;
```

5) Display name of hackers whose accuracy is more than 50%:

```
>>> SELECT hacker FROM hackers WHERE h_accuracy > 50.00;
```

6) Display name and accuracy of those crackers whose name starts from 'S' and accuracy is less than 60%:

```
>>> SELECT cracker, c_accuracy FROM crackers WHERE cracker LIKE 'S%' AND c_accuracy < 60.00;
```

7) Display all records of hackers whose accuracy between 50 to 80 and name of 5 characters:

```
>>> SELECT * FROM hackers WHERE h_accuracy BETWEEN 50 & 80 AND hackers LIKE '_____%';
```

8) Display number of records in crackers:

```
>>> SELECT COUNT(*) FROM crackers;
```

9) Change the number of hacks in hackers table such that number of hacks more than 100 and less than 1000 changed as 500:

>>> UPDATE hackers SET hacks = 500 WHERE hacks BETWEEN 100 AND 1000;

10) Change the maximum value of names of hackers as 10 in the hackers table

>>> ALTER TABLE hackers MODIFY hackers hackers VARCHAR(10);

11) Add one column named age in the crackers table:

>>> ALTER TABLE crackers ADD age INT;

12) Delete those records where accuracy less than 50% in the hackers:

>>> DELETE FROM hackers WHERE h_accuracy < 50.00;

13) Delete column age from hackers:

>>> ALTER TABLE hackers DROP COLOMN age;

14) Change the name of crackers to 'Anonymous' whose name include the letter 'a':

>>> UPDATE crackers SET cracker = 'Anonymous' WHERE cracker LIKE '%a%';

15) Display names all of hackers and crackers

>>> SELECT H.hacker, C.cracker FROM hackers H JOIN crackers C ON H.hacker_id = C.cracker_id;

16) Display the name of those hackers whose accuracy is more than 20% in decreasing order of their respective names:

>>> SELECT hacker FROM hackers WHERE h_accuracy > 20.00 ORDER BY hacker;

17) Display the highest accuracy in the crackers table with heading 'Mighty Cracker':

>>> SELECT MAX(c_accuracy) as 'Mighty Cracker' FROM crackers;

18) Add a column 'DOH' to the hackers table for entering date of hack:

>>> ALTER TABLE hackers ADD DOH DATE;

19) In the hackers table set DOH as the current date:

```
>>> UPDATE hackers SET DOH = CURDATE();
```

20) Delete all records of those crackers whose average of cracks and accuracy is less than 60 and name of more than 4 characters or accuracy more than 50%:

```
>>> DELETE FROM crackers WHERE (cracks + c_accuracy) < 60 AND
(cracker LIKE '____%' OR c_accuracy > 50.00);
```

29: MySQL - II

1) Using pre-created database:

```
>>> USE tanishq;
```

2) Creating table in MySQL:

```
>>> CREATE TABLE store (item_id INT PRIMARY KEY,
item_name VARCHAR (20), brand VARCHAR (10), price INT);

>>> CREATE TABLE orders (order_id INT PRIMARY KEY,
customer VARCHAR (20), phone_no INT(10), item
VARCHAR(20), quantity INT, amount INT, dop DATE);
```

3) Inserting values in the table:

```
>>> INSERT INTO store VALUES (1, 'Realme 7', 'Oppo',
17999);
>>> INSERT INTO store VALUES (2, 'iPhone 12 Pro', 'Apple',
165000);
>>> INSERT INTO store VALUES (3, 'Redmi Note 9', 'Xiomi',
51999);
>>> INSERT INTO store VALUES (4, 'P40 Pro', 'Huawei',
76500);
>>> INSERT INTO store VALUES (5, 'Z Fold Ultra',
'Samsung', 181000);
```

```
>>> INSERT INTO orders VALUES (1, 'Tanishq', 9876543210, 'Apple
iphone 12 Pro', 2, 330000, '10-11-2020');
>>> INSERT INTO orders VALUES (2, 'John', 9872829386, 'Samsung Z
Fold', 4, 724000, '01-10-2020');
>>> INSERT INTO orders VALUES (1, 'Jeff', 7582583652, 'Huawei P40
Pro', 10, 765000, '08-11-2020');
>>> INSERT INTO orders VALUES (1, 'Steve', 9876543210, 'Apple
iphone 12 Pro', 1, 165000, '16-09-2020');
>>> INSERT INTO orders VALUES (1, 'Yongping', 8686383636, 'Oppo
Realme 7', 100, 1799900, '22-10-2020');
```

4) Display all records of table store and orders:

```
>>> SELECT * FROM store;

>>> SELECT * FROM orders;
```

5) Display the name of customers who purchased above 10 lakhs:

```
>>> SELECT customer FROM orders WHERE amount > 100000;
```

6) Display all records of customers who purchased Apple iPhone 12 Pro:

```
>>> SELECT * FROM orders WHERE item = 'Apple iPhone 12 Pro';
```

7) Calculate the number of brands in store:

```
>>> SELECT COUNT(brand) FROM store;
```

8) Display the name, item purchased of a customer whose name starts from 'S' and in the ascending order of their name:

```
>>> SELECT customer, item FROM orders WHERE customer LIKE 'S%' ORDER BY customer;
```

9) Display the name of customer and item group with item:

```
>>> SELECT customer, item FROM orders GROUP BY item;
```

10) Display records of those customer who bought in the month of October:

```
>>> SELECT * FROM orders WHERE dop >= '01-10-2020' AND dop < '01-11-2020';
```

11) Display the name of item and brand having highest price in comparison to others:

```
>>> SELECT item_name, brand FROM store WHERE price = MAX(price);
```

12) Display the customer name item name and price of unit quantity of item by using join:

```
>>> SELECT O.customer, O.item, S.price FROM store S JOIN orders O ON S.item_id = O.order_id;
```

13) Change the quantity of item to 80 for those customer who purchased number of items less than 100:

```
>>> UPDATE orders SET quantity = 80 WHERE quantity < 100;
```

14) Display the name of that customer who bought most latest:

```
>>> SELECT customer FROM orders WHERE dop = MAX(dop);
```

15) Add a column 'Ranking' to the table store:

```
>>> ALTER TABLE store ADD Ranking INT;
```

16) Modify the maximum range of characters of customer name in the orders table to 15:

```
>>> ALTER TABLE orders MODIFY customer Customer VARCHAR (15);
```

17) Delete those records from table store that have price less than 50000 or brand name starting with 'A':

```
>>> DELETE FROM store WHERE price < 50000 OR brand LIKE 'A%';
```

18) Display the name of customer who bought before 10st of November, 2020 and whose name ends with 't' in the order of latest to older date:

```
>>> SELECT customer FROM orders WHERE dop < '10-11-2020' AND customer LIKE '%t' ORDER BY dop DESC;
```

19) Display the maximum and minimum amount of purchase from the orders table with heading 'Highest' and 'Lowest' respectively:

```
>>> SELECT MAX (amount) AS 'Highest', MIN (amount) AS 'Lowest' FROM orders;
```

20) Display the turn over of the seller ,i.e, the total amount selled to the customer with heading 'Turn Over' and the average price of the phone bought by the customer with heading 'Average Price':

```
>>> SELECT SUM (price) as 'Turn Over', AVG (price) as 'Average Price' FROM orders;
```