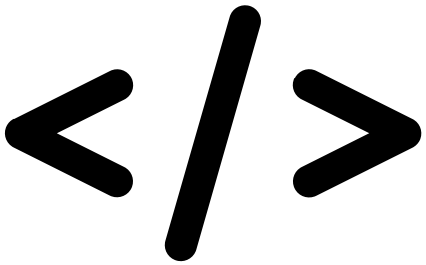

Computer Science Program File

Name
Idhant Gulati

Session
2022-23

Subject
Computer Science (083)

Project Guide
Mr. Sarthak Arora

Certificate

This is to certify that Idhant Gulati, a student of class XII at Indraprastha World School has successfully completed the computer science program work under the guidance of Mr. Sarthak Arora (Subject Teacher) in the academic year 2022-2023.

_____

Signature

Acknowledgment

I am deeply grateful to our esteemed computer science teacher, Mr Sarthak Arora, for his invaluable assistance and supervision throughout this project. His unwavering support and sage advice was invaluable in enabling me to progress and ultimately succeed in my endeavours. I am truly thankful for his guidance and willingness to help me along the journey, and I deeply appreciate his commitment to ensuring the success of my project.

</>

Python Programs

Program 1

Make a menu-driven program to push, pop, peek, display stack.

Code

```
def push(stack, item):
    stack.append(item)
    print("Item pushed to stack")


def pop(stack):
    if len(stack) == 0:
        print("Stack is empty")
    else:
        stack.pop()
        print("Item popped from stack")


def peek(stack):
    if len(stack) == 0:
        print("Stack is empty")
    else:
        print(stack[-1])


def display(stack):
    if len(stack) == 0:
        print("Stack is empty")
    else:
        print(stack)


def main():
    stack = []
    choice = 0
    while choice != 6:
        print("1. Push")
        print("2. Pop")
        print("3. Peek")
        print("4. Display")
        print("5. Exit")
```

```
        choice = int(input("Enter your choice : "))
        if choice == 1:
            item = input("Enter the item to be pushed : ")
            push(stack, item)
        elif choice == 2:
            pop(stack)
        elif choice == 3:
            peek(stack)
        elif choice == 4:
            display(stack)
        elif choice == 5:
            exit()
        else:
            print("Wrong choice")


if __name__ == '__main__':
    main()
```

Output

```
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter your choice : 1
Enter the item to be pushed : 12
Item pushed to stack
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter your choice : 1
Enter the item to be pushed : 45
Item pushed to stack
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter your choice : 1
Enter the item to be pushed : 67
Item pushed to stack
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter your choice : 2
Item popped from stack
```

```
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter your choice : 3
45
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter your choice : 4
['12', '45']
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter your choice : 5
```

Program 2

Write a python program to check whether a string is a palindrome or not using stack.

Code

```
stack = []
top = -1

# push function
def push(ele):
  global top
  top += 1
  stack[top] = ele

# pop function
def pop():
  global top
  ele = stack[top]
  top -= 1
  return ele

# Function that returns 1 if string is a palindrome
def isPalindrome(string):
  global stack
  length = len(string)
```

```
  # Allocating the memory for the stack
  stack = ['0'] * (length + 1)

  # Finding the mid
  mid = length // 2
  i = 0
  while i < mid:
    push(string[i])
    i += 1

  # Checking if the length of the string is odd, if odd then neglect the middle character
  if length % 2 != 0:
    i += 1

  # While not the end of the string
  while i < length:
    ele = pop()

    # If the characters differ then the given string is not a palindrome
    if ele != string[i]:
      return False
    i += 1
  return True
string = input("Enter string to check: ")

if isPalindrome(string):
        print("Yes, the string is a palindrome")
else:
        print("No, the string is not a palindrome")
```

Output

```
Enter string to check: jalaj
No, the string is not a palindrome
```

Program 3

Write a Python program to check if a given year is a leap year or not.

Code

```
def is_leap(year):
    leap = False
    if year % 4 == 0:
        if year % 100 == 0:
```

```
            if year % 400 == 0:
                leap = True
            else:
                leap = False
        else:
            leap = True
    else:
        leap = False

    return leap

year = int(input())
print(is_leap(year))
```

Output

```
Enter an year 2030
False
```

Program 4

Write a Program to enter the number of terms and to print the Fibonacci Series.

Code

```
def fibonnaci(num):
    if num == 0:
        return 0
    elif num == 1:
        return 1
    else:
        return fibonnaci(num-1) + fibonnaci(num-2)


for i in range(int(input())):
    print(fibonnaci(i))
```

Output

```
Enter the number which you to find the Fibonnaci series- 7
0
1
1
2
3
5
8
```

Program 5

Write a python program to write, read, and search data of a student in binary file.

Code

```
import pickle

def write():
    f=open("student.dat","ab")
    name=input("Enter the name of student: ")
    rollno=int(input("Enter the roll no of student: "))
    marks=int(input("Enter the marks of student: "))
    s=[name,rollno,marks]
    pickle.dump(s,f)
    f.close()

def read():
    f=open("student.dat","rb")
    try:
        while True:
            s=pickle.load(f)
            print("Name:",s[0])
            print("Roll no:",s[1])
            print("Marks:",s[2])
            print()
    except EOFError:
        f.close()

def search():
    f=open("student.dat","rb")
    rollno=int(input("Enter the roll no to be searched: "))
    try:
        while True:
            s=pickle.load(f)
            if s[1]==rollno:
                print("Name:",s[0])
                print("Rollno:",s[1])
```

```
                print("Marks:",s[2])
                print()

    except EOFError:
        f.close()

print('1.Enter Data\n2.Read Data\n3.Search Data\n4.Exit')
lol=False
while not lol:
    choice=int(input("Enter Choice: "))
    if choice==1:
        write()
    elif choice==2:
        read()
    elif choice==3:
        search()
    elif choice==4:
        lol=True
    else:
        print('Invalid Choice')
```

Code

```
1.Enter Data
2.Read Data
3.Search Data
4.Exit
Enter Choice: 1
Enter the name of student: Idhant
Enter the roll no of student: 19
Enter the marks of student: 95
Enter Choice: 1
Enter the name of student: Viv
Enter the roll no of student: 35
Enter the marks of student: 97
Enter Choice: 2
Name: Idhant
Roll no: 19
Marks: 95

Name: Viv
Roll no: 35
Marks: 97

Enter Choice: 3
Enter the roll no to be searched:33
5
Name: Viv
Rollno: 35
Marks: 97

Enter Choice: 4
```

Program 6

To calculate factorial of a number

Code

```
x=int(input("Enter the number whose factorial you want to calculate- "))
fac=1
for i in range(x,1,-1):
    fac=fac*i
print("Factorial of",x,"is",fac)
```

Output

```
Enter the number whose factorial you want to calculate- 12
Factorial of 12 is 479001600
```

Program 7

Write a menu-driven program to add an element, search an element, display the dictionary, sort the dictonary, and delete an element.

Code

```
class dicto:
    # Method
    def menu(self):
        print("Please enter your choise")
        print("1. Add element")
        print("2. Search element")
        print("3. Display dictionary")
        print("4. Sort dictionary")
        print("5. Delete element")
        try:
            ch = int(input("Enter your choice: "))
            # Choice
            if ch == 1:
                self.add()
```

```
            elif ch == 2:
                self.search()
            elif ch == 3:
                self.display()
            elif ch == 4:
                self.sort()
            elif ch == 5:
                self.delete()
        except:
            # Exception
            print("Enter only number.")
        else:
            # calling
            d = dicto()
            d.menu()

    # method
    def add(self):
        #Taking
        name = input("Enter your friend name: ")
        num = int(input("Enter his/her number: "))
        friends[name] = num
        #Printvalue
        print(name, " and ", num, " are added in dictionary.")


    #Method
    def delete(self):
        #taking
        a = input("Enter name: ")
        del friends[a]
        print(a, " and his/her number deleted successfully.")


    # Method
    def search(self):
        a = input("Enter name: ")
        print(friends[a], type(friends[a]), friends.get(a), type(friends.get(a)))


    # Method
    def sort(self):
        temp = sorted(friends)
        print(temp)

    # Method
    def display(self):
        for name, num in friends.items():
            print(name, " : ", num)



# object and calling of dicto class
friends = {}
d = dicto()
d.menu()
```

Output

```
Please enter your choise
1. Add element
2. Search element
3. Display dictionary
4. Sort dictionary
5. Delete element
Enter your choice: 1
Enter your friend name: Viv
Enter his/her number: 9650146907
Viv  and  9650146907  are added in dictionary.
Please enter your choise
1. Add element
2. Search element
3. Display dictionary
4. Sort dictionary
5. Delete element
Enter your choice: 1
Enter your friend name: Adi
Enter his/her number: 9310472464
Adi  and  9310472464  are added in dictionary.
Please enter your choise
1. Add element
2. Search element
3. Display dictionary
4. Sort dictionary
5. Delete element
Enter your choice: 2
Enter name: Viv
9650146907 <class 'int'> 9650146907 <class 'int'>
Please enter your choise
1. Add element
2. Search element
3. Display dictionary
4. Sort dictionary
5. Delete element
Enter your choice: 3
Viv  :  9650146907
Adi  :  9310472464
Please enter your choise
1. Add element
2. Search element
3. Display dictionary
4. Sort dictionary
5. Delete element
Enter your choice: 4
['Adi', 'Viv']
Please enter your choise
1. Add element
2. Search element
```

```
3. Display dictionary
4. Sort dictionary
5. Delete element
Enter your choice: 5
Enter name: Adi
Adi  and his/her number deleted successfully.
```

Program 8

Generate a strong password

Code

```
import string
import random

characters = list(string.ascii_letters + string.digits + "!@#$%^&*()")

def generate_random_password():
    length = int(input("Enter password length: "))
    random.shuffle(characters)
    password = []
    for i in range(length):
        password.append(random.choice(characters))
    random.shuffle(password)
    print("".join(password))

generate_random_password()
```

Output

```
Enter password length: 12
ne!cbRawX9f)
```

Program 9

Sort a list using bubble sort

Code

```
#bubble sort
def bubble_sort(list):
    for i in range(len(list)-1,0,-1):
        for j in range(i):
            if list[j]>list[j+1]:
                temp=list[j]
                list[j]=list[j+1]
                list[j+1]=temp

#_main_
list=[19,2,31,45,6,11,121,27]
print(list)
bubble_sort(list)
print("Sorted list ",list)
```

Output

```
Unsorted list [19, 2, 31, 45, 6, 11, 121, 27]
Sorted list  [2, 6, 11, 19, 27, 31, 45, 121]
```

Program 10

Sort a list using selection sort

Code

```
def selection_sort(list):
    for i in range(len(list)-1,0,-1):
        max_pos=0
        for j in range(1,i+1):
            if list[j]>list[max_pos]:
                max_pos=j
        temp=list[i]
        list[i]=list[max_pos]
        list[max_pos]=temp

#_main_
list=[19,2,31,45,6,11,121,27]
print(list)
selection_sort(list)
print("Sorted list ",list)
```

Output

```
Unsorted list [19, 2, 31, 45, 6, 11, 121, 27]
Sorted list [2, 6, 11, 19, 27, 31, 45, 121]
```

Program 11

Sort a list using insertion sort

Code

```
def insertion_sort(list):
    for i in range(1,len(list)):
        current_value=list[i]
        position=i
        while position>0 and list[position-1]>current_value:
            list[position]=list[position-1]
            position=position-1
        list[position]=current_value

#_main_
list=[19,2,31,45,6,11,121,27]
print(list)
insertion_sort(list)
print("Sorted list ",list)
```

Output

```
Unsorted list [19, 2, 31, 45, 6, 11, 121, 27]
Sorted list [2, 6, 11, 19, 27, 31, 45, 121]
```

Program 12

Sort a list using insertion sort

Code

```
def merge_sort(list):
    if len(list)>1:
        mid=len(list)//2
        left_list=list[:mid]
        right_list=list[mid:]

        merge_sort(left_list)
        merge_sort(right_list)

        i=0
        j=0
        k=0
        while i<len(left_list) and j<len(right_list):
            if left_list[i]<right_list[j]:
                list[k]=left_list[i]
                i=i+1
            else:
                list[k]=right_list[j]
                j=j+1
            k=k+1
        while i<len(left_list):
            list[k]=left_list[i]
            i=i+1
            k=k+1
        while j<len(right_list):
            list[k]=right_list[j]
            j=j+1
            k=k+1

#_main_
list=[19,2,31,45,6,11,121,27]
print("Unsorted list", list)
merge_sort(list)
print("Sorted list",list)
```

Output

```
Unsorted list [19, 2, 31, 45, 6, 11, 121, 27]
Sorted list  [2, 6, 11, 19, 27, 31, 45, 121]
```

Program 13

A menu driven program for CSV file to read, write, and search data.

Code

```
import csv


def write_data():
    f_obj = open('Data.csv', 'a', newline='')
    w_obj = csv.writer(f_obj)
    n = input('Enter name: ')
    mb = input('Enter mobile number: ')
    addr = input('Enter address: ')
    rec = [n, mb, addr]
    w_obj.writerow(rec)
    f_obj.close()


def read_data():
    f_obj = open('Data.csv', 'r')
    r_obj = csv.reader(f_obj)
    for r in r_obj:
        print(r)
    f_obj.close()


def search_data():
    f_obj = open('Data.csv', 'r')
    r_obj = csv.reader(f_obj)
    n = input('Enter name: ')
    for i in r_obj:
        if n == i[0]:
            print(i)

    f_obj.close()


print('1.Enter Data\n2.Read Data\n3.Search Data\n4.Exit')
check = False
while not check:
    choice = int(input("Enter Choice: "))
    if choice == 1:
        write_data()
    elif choice == 2:
        read_data()
    elif choice == 3:
        search_data()
    elif choice == 4:
        print("Program Closed.")
        check = True
    else:
        print('Invalid Choice')
```

Ouput

```
1.Enter Data
2.Read Data
3.Search Data
4.Exit
Enter Choice: 1
Enter name: Idhant
Enter mobile number: 8383948080
Enter address: B3/1
Enter Choice: 1
Enter name: Reet
Enter mobile number: 9911401212
Enter address: B3/2
Enter Choice: 3
Enter name: Idhant
['Idhant', '8383948080', 'B3/1']
Enter Choice: 2
['Idhant', '8383948080', 'B3/1']
['Reet', '9911401212', 'B3/2']
Enter Choice: 4
Program Closed.
```

Program 14

Program to find simple interest for given principal amount, time and rate of interest.

Code

```
def simple_interest():
  p = int(input('The principal is ',))
  t = int(input('The time period is ',))
  r = int(input('The rate of interest is ',))
  si = (p * t * r)/100
  print('The Simple Interest is', si)

#_main_
simple_interest()
```

Output

```
The principal is 4
The time period is 7
The rate of interest is 9
The Simple Interest is 2.52
```

Program 15

To print the number of words starting with ‘v’ or ‘V’ in a text file

Code

```
file=open("thefile.txt","r")
count=0
a=file.read()
a=a.split()
for i in a:
    if i[0].lower()=="v":
        count=count+1
print("No of letters starting with v/V are",count)
file.close()
```

Output

```
No of letters starting with v/V are 6
```

MySQL Queries

Query 1

1. Create a new database called "school”

```
mysql> CREATE DATABASE school;
Query OK, 1 row affected (0.01 sec)
```

2. Create a new table called "students" with columns for "id" (an integer), "name" (a varchar), and "grade" (an integer):

```
mysql> CREATE TABLE students (
    -> id INTEGER PRIMARY KEY,
    -> name VARCHAR(255),
    -> grade INTEGER
    -> );
Query OK, 0 rows affected (0.02 sec)
```

3. Insert a new row into the "students" table with values for the "id", "name", and "grade" columns:

```
INSERT INTO students (id, name, grade) VALUES (1, 'Alice', 10);
Query OK, 1 row affected (0.01 sec)
```

4. Retrieve all rows from the "students" table:

```
mysql> SELECT * FROM students;
+----+-------+-------+
| id | name  | grade |
+----+-------+-------+
|  1 | Alice |    10 |
+----+-------+-------+
1 row in set (0.00 sec)
```

Query 2

1. Create a database called "employee_database":

```
mysql> CREATE DATABASE employee_database;
Query OK, 1 row affected (0.00 sec)
```

2. Create a table called "employees" with columns for employee id, name, salary, and department:

```
mysql> CREATE TABLE employees (
    -> employee_id INT PRIMARY KEY,
    -> name VARCHAR(255),
    -> salary DECIMAL(10, 2),
    -> department VARCHAR(255)
    -> );
    Query OK, 0 rows affected (0.01 sec)
```

3. Insert a row into the "employees" table with an employee id of 1, name "Alice", salary of 50000, and department "Marketing":

```
mysql> INSERT INTO employees (employee_id, name, salary, department)
    -> VALUES (1, 'Alice', 50000, 'Marketing');
Query OK, 1 row affected (0.01 sec)
```

4. Update the salary of the employee with id 1 to 55000:

```
mysql> UPDATE employees
    -> SET salary = 55000
    -> WHERE employee_id = 1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

5. Delete the employee with id 1 from the "employees" table:

```
mysql> DELETE FROM employees
    -> WHERE employee_id = 1;
Query OK, 1 row affected (0.02 sec)
```

6. Retrieve all rows from the "employees" table:

```
mysql> SELECT * FROM employees;
Empty set (0.00 sec)
```

Query 3

1. Create a database called "order_database":

```
mysql> CREATE DATABASE order_database;
Query OK, 1 row affected (0.00 sec)
```

2. Create a table called "orders" with columns for order id, customer id, product id, and quantity:

```
mysql> CREATE TABLE orders (
    -> order_id INT PRIMARY KEY,
    -> customer_id INT,
    -> product_id INT,
    -> quantity INT
    -> );
Query OK, 0 rows affected (0.01 sec)
```

3. Create a table called "customers" with columns for customer id, name, and address:

```
mysql> CREATE TABLE customers (
    -> customer_id INT PRIMARY KEY,
    -> name VARCHAR(255),
    -> address VARCHAR(255)
    -> );
Query OK, 0 rows affected (0.01 sec)
```

4. Create a table called "products" with columns for product id, name, and price:

```
mysql> CREATE TABLE products (
    -> product_id INT PRIMARY KEY,
    -> name VARCHAR(255),
    -> price DECIMAL(10, 2)
    -> );
Query OK, 0 rows affected (0.01 sec)
```

5. Insert some data into the "customers" and "products" tables:

```
mysql> INSERT INTO customers (customer_id, name, address)
    -> VALUES (1, 'Alice', '123 Main Street'),
    ->        (2, 'Bob', '456 Market Street');
Query OK, 2 rows affected (0.01 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> INSERT INTO products (product_id, name, price)
    -> VALUES (1, 'Computer', 1000),
    ->        (2, 'Monitor', 500),
    ->        (3, 'Keyboard', 100);
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

6. Insert an order into the "orders" table for a customer with id 1 and a product with id 2, with a quantity of 2:

```
mysql> INSERT INTO orders (order_id, customer_id, product_id, quantity)
    -> VALUES (1, 1, 2, 2);
Query OK, 1 row affected (0.00 sec)
```

7. Update the quantity of the order with id 1 to 3:

```
mysql> UPDATE orders
    -> SET quantity = 3
    -> WHERE order_id = 1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

8. Retrieve the names and addresses of all customers who have placed an order:

```
mysql> SELECT c.name, c.address FROM customers c
    -> INNER JOIN orders o ON c.customer_id = o.customer_id;
+-------+-----------------+
| name  | address         |
+-------+-----------------+
| Alice | 123 Main Street |
+-------+-----------------+
1 row in set (0.00 sec)
```

9. Retrieve the names and quantities of all products that have been ordered:

```
mysql> SELECT p.name, o.quantity FROM products p
    -> INNER JOIN orders o ON p.product_id = o.product_id;
+---------+----------+
| name    | quantity |
+---------+----------+
| Monitor |        3 |
+---------+----------+
1 row in set (0.00 sec)
```

10. Retrieve the total revenue from all orders:

```
mysql> SELECT SUM(p.price * o.quantity) FROM products p
    -> INNER JOIN orders o ON p.product_id = o.product_id;
+---------------------------+
| SUM(p.price * o.quantity) |
+---------------------------+
|                   1500.00 |
+---------------------------+
1 row in set (0.00 sec)
```

Query 4

1. Create a database called "sales_database":

```
mysql> CREATE DATABASE sales_database;
Query OK, 1 row affected (0.01 sec)
```

2. Create a table called "sales" with columns for sale id, product id, date, and quantity:

```
mysql> CREATE TABLE sales (
    -> sale_id INT PRIMARY KEY,
    -> product_id INT,
    -> date DATE,
    -> quantity INT
    -> );
Query OK, 0 rows affected (0.01 sec)
```

3. Create a table called "products" with columns for product id, name, and price:

```
mysql> CREATE TABLE products (
    -> product_id INT PRIMARY KEY,
```

```
    -> name VARCHAR(255),
    -> price DECIMAL(10, 2)
    -> );
Query OK, 0 rows affected (0.01 sec)
```

4. Insert some data into the "products" and "sales" tables:

```
mysql> INSERT INTO products (product_id, name, price)
    -> VALUES (1, 'Computer', 1000),
    ->        (2, 'Monitor', 500),
    ->        (3, 'Keyboard', 100);
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> INSERT INTO sales (sale_id, product_id, date, quantity)
    -> VALUES (1, 1, '2022-01-01', 2),
    ->        (2, 2, '2022-01-02', 3),
    ->        (3, 3, '2022-01-03', 1);
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

5. Update the quantity of the sale with id 1 to 3:

```
mysql> UPDATE sales
    -> SET quantity = 3
    -> WHERE sale_id = 1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

6. Retrieve the names and quantities of all products that have been sold:

```
mysql> SELECT p.name, s.quantity FROM products p
    -> INNER JOIN sales s ON p.product_id = s.product_id;
+----------+----------+
| name     | quantity |
+----------+----------+
| Computer |        3 |
| Monitor  |        3 |
| Keyboard |        1 |
+----------+----------+
3 rows in set (0.00 sec)
```

7. Retrieve the total revenue from all sales:

```
mysql> SELECT SUM(p.price * s.quantity) FROM products p
    -> INNER JOIN sales s ON p.product_id = s.product_id;
+--------------------------+
| SUM(p.price * s.quantity) |
+--------------------------+
|                  4600.00 |
+--------------------------+
1 row in set (0.00 sec)
```

8. Retrieve the total number of computers sold:

```
mysql> SELECT SUM(s.quantity) FROM sales s
    -> INNER JOIN products p ON s.product_id = p.product_id
    -> WHERE p.name = 'Computer';
+-----------------+
| SUM(s.quantity) |
+-----------------+
|               3 |
+-----------------+
1 row in set (0.00 sec)
```

9. Retrieve the total revenue from all sales on 2022-01-01:

```
mysql> SELECT SUM(p.price * s.quantity) FROM products p
    -> INNER JOIN sales s ON p.product_id = s.product_id
    -> WHERE s.date = '2022-01-01';
+--------------------------+
| SUM(p.price * s.quantity) |
+--------------------------+
|                  3000.00 |
+--------------------------+
1 row in set (0.00 sec)
```

Query 5

1. Create a database called "movie_database":

```
mysql> CREATE DATABASE movie_database;
Query OK, 1 row affected (0.00 sec)
```

2. Create a table called "movies" with columns for movie id, title, release year, and genre:

```
mysql> CREATE TABLE movies (
    -> movie_id INT PRIMARY KEY,
    -> title VARCHAR(255),
    -> release_year INT,
    -> genre VARCHAR(255)
    -> );
Query OK, 0 rows affected (0.01 sec)
```

3. Insert some data into the "movies" table:

```
mysql> INSERT INTO movies (movie_id, title, release_year, genre)
    -> VALUES (1, 'The Matrix', 1999, 'Sci-fi'),
    ->         (2, '2001: A Space Odyssey', 1968, 'Sci-fi'),
    ->         (3, 'Interstellar', 2014, 'Sci-fi'),
    ->         (4, 'ted', 2012, 'Comedy'),
    ->         (5, 'Fast and Furious', 2015, 'Action');
Query OK, 5 rows affected (0.00 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

4. Update the release year of the movie with id 5 to 2016:

```
mysql> UPDATE movies
    -> SET release_year = 2016
    -> WHERE movie_id = 5;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

5. Retrieve all movies in the Sci-fi genre:

```
mysql> SELECT * FROM movies
    -> WHERE genre = 'Sci-fi';
+----------+----------------------+--------------+--------+
| movie_id | title                | release_year | genre  |
+----------+----------------------+--------------+--------+
|        1 | The Matrix           |         1999 | Sci-fi |
|        2 | 2001: A Space Odyssey |         1968 | Sci-fi |
|        3 | Interstellar         |         2014 | Sci-fi |
+----------+----------------------+--------------+--------+
3 rows in set (0.00 sec)
```