**Video Library Management System**

**Computer Science**

**Project**

Made By:-

Debashish Majumdar

XII – C

Roll no – 7

**Certificate**

This is to certify that Debashish Majumdar, student of Class XII-C, Air Force Bal Bharati School, Lodhi Road has completed the project titled “Analysis and Design of Video

Library Management system”  during the academic year 2016-2017 towards partial fulfillment of credit for the Computer Science practical evaluation of CBSE 2017, and submitted satisfactory report, as compiled in the following pages, under my supervision.

(INTERNAL) (EXTERNAL)

Acknowledgements

I would like to express my sincere gratitude to my physics mentor MRS. Rawley, for her vital support, guidance and encouragement, without which this project would not have come forth. I would also like to express my gratitude to the OTHER staff of the Department of Computer Science for their support during the making of this project.

(Debashish Majumdar)

Input code:

import os

from pickle import load, dump

import datetime

import string

MFile = "Master.dat"

File1 = "Cassettes.dat"

File2 = "Balance.dat"

File3 = "Customer.dat"

Cdate = datetime.datetime.now() # Current date and time

# Class for date

class Cast\_Date:

def \_\_init\_\_(self):

self.dd = Cdate.day

self.mm = Cdate.month

self.yy = Cdate.year

class Master:

# Constructor

def \_\_init\_\_(self):

self.Cast\_Code = 0 # cassette/CD code - (Like, 1, 2, 3, etc.)

self.Cast\_Name = " " # Title of the cassette/CD

self.Cast\_Comp = " " # cassette/CD company

self.Cast\_Price = 0 # Price per cassette/CD

def Check\_Code(self, C\_Code):

MList = list()

TRec = list()

Flag = False # To check if Cast\_Code is in Master.dat or not

if os.path.isfile(MFile):

Mobj = open(MFile, 'rb')

try:

while True:

MRec = [] # For extracting Master.dat records

MRec = load(Mobj)

if (C\_Code == MRec[0]):

TRec = MRec

MList.append(MRec[0])

except EOFError:

pass

for i in range(len(MList)):

if (C\_Code == MList[i]):

Flag = True

break

Mobj.close()

# Flag for Master data entry and TRec for Cassette data entry

return Flag, TRec

# For Master data entry

def Master\_Entry(self):

TRec = list() # A temporary list to store master record

print("Add Master Cassette/CD");

ch ='Y'

while ch=='Y':

while True:

self.Cast\_Code = int(input("Cassette/CD Code (1/2/3...) # "))

Flag, TRec = self.Check\_Code(self.Cast\_Code)

if (Flag == False):

while True:

self.Cast\_Name = input("Cassette/CD Name : ")

if (self.Cast\_Name == 0 or len(self.Cast\_Name) > 25):

print("Cassette/CD Name should not greater than 25")

else:

break

while True:

self.Cast\_Comp = input("Company Name : ")

if (self.Cast\_Comp == 0 or len(self.Cast\_Comp) > 25):

print("Company Name should not greater than 25")

else:

break

while True:

self.Cast\_Price = float(input("Individual Cassette/CD price : "))

if (self.Cast\_Price <= 0):

print("Enter valid price for Cassette/CD")

else:

break;

with open(MFile, 'ab+') as Mobj:

if not Mobj:

print (MFile, "does not created")

else:

# Appends data into a sequnce object

MList = list()

MList.append(self.Cast\_Code)

MList.append(self.Cast\_Name)

MList.append(self.Cast\_Comp)

MList.append(self.Cast\_Price)

# Write data into binary file

dump(MList, Mobj)

else:

print ("Code", self.Cast\_Code, "is already in 'Master.dat' file")

ch = input("Add new Cassette/CD code? <Y/N>: ")

ch = ch.upper()

if ch=='Y':

continue

else:

break

def Master\_Display(self):

if not os.path.isfile(MFile):

print (MFile, "file does not exist")

else:

Mobj = open(MFile, 'rb')

print ("\nCassette/CD Master Report")

print ("=" \* 25)

print ("{0:<7} {1:<30} {2:<20} {3:>8}".format(" Code", "Cassette/CD Name", "Company Name", "Price"))

print ("-" \* 70)

try:

while True:

MRec = []

MRec = load(Mobj)

print ("{0:<7} {1:<30} {2:<20} {3:>8.2f}"

.format(' '+str(MRec[0]), MRec[1], MRec[2], MRec[3]))

except EOFError:

pass

print ("-" \* 70)

Mobj.close()

class Cassettes:

# Constructor

def \_\_init\_\_(self):

self.Cast\_Code = 0 # cassette/CD code - (Like, 1, 2, 3, etc.)

self.Tot\_Cast = 0 # Total cassette/CD purchased

self.dd = self.mm = self.yy = 0 # Cassette/CD purchase date

# For cassettes/CDs entry into the cassettes.dat data file

def New\_Cassettes(self):

M = Master()

B = Balance()

CDt = Cast\_Date()

self.dd = CDt.dd

self.mm = CDt.mm

self.yy = CDt.yy

print("Add New Stock cassette/CD");

ch ='Y'

while ch=='Y':

TRec = list() # A temporary list to store master record

Flag = False # To check if Cast\_Code is in Master.dat or not

print("Date: %s-%s-%s" % (CDt.dd, CDt.mm, CDt.yy))

while True:

self.Cast\_Code = int(input("Cassette/CD Code (1/2/3/...) # "))

# Function call to check cassette/CD code in Master.dat

Flag, TRec = M.Check\_Code(self.Cast\_Code)

if (Flag == True):

self.Cast\_Name = TRec[1] # Title of the cassette/CD

self.Cast\_Comp = TRec[2] # cassette/CD company

self.Cast\_Price = TRec[3] # Price per cassette/CD

print("Cassette/CD Name :", self.Cast\_Name)

print("Company Name : ", self.Cast\_Comp)

print("Individual Cassette/CD price : ",self.Cast\_Price)

while True:

self.Tot\_Cast = int(input("Enter new stock cassettes/CDs purchased (Stock): "))

if (self.Tot\_Cast <= 0):

print("Enter valid Cassette/CD number");

else:

break

ch = input("Do you want to save the record <Y/N>: ")

if ch == 'y' or ch == 'Y':

CList = list()

with open(File1, 'ab') as Cobj:

if not Cobj:

print (File1, "does not created")

else:

# Appends data into a sequnce object

CList.append(self.Cast\_Code)

CList.append(self.Tot\_Cast)

CList.append(self.dd)

CList.append(self.mm)

CList.append(self.yy)

# Write data into binary file

dump(CList, Cobj)

#B.Add\_to\_File(self.Cast\_Code, self.Tot\_Cast, self.Cast\_Price, self.dd, self.mm, self.yy)

B.AddUpdateBalance(CList, self.Cast\_Price)

print("Record saved")

ch = input("Stock more cassette/CD record? <Y/N>: ")

ch = ch.upper()

if ch=='Y':

continue

else:

break

# For cassettes/CDs entry into the cassettes.dat data file

def Display\_Cassettes(self):

M = Master()

if not os.path.isfile(File1):

print (File1, "file does not exist")

else:

Cobj = open(File1, 'rb')

print ("\nCassette/CD entry Register")

print ("=" \* 26)

print ("{0:>5} {1:<25} {2:<20} {3:>10} {4:>8} {5:<12}"

.format("Code", "Name", "Company Name", "Quantity", "Price", "Date"))

print ("-" \* 85)

try:

while True:

CRec = []

CRec = load(Cobj)

TRec = list()

Flag, TRec = M.Check\_Code(CRec[0])

nDt = Set\_DateFormat(CRec[2], CRec[3], CRec[4])

if (Flag == True):

print ("{0:>5} {1:<25} {2:<20} {3:>10} {4:>8.2f} {5:<12}"

.format(CRec[0], TRec[1], TRec[2], CRec[1], TRec[3], nDt))

except EOFError:

pass

print ("-" \* 85)

Cobj.close()

# Function to set the date as: DD-MM-YYYY

def Set\_DateFormat(d1, m1, y1):

fDt = ''

d11 = str(d1)

m11 = str(m1)

y11 = str(y1)

if (len(d11)==1):

d11 = '0'+d11

if (len(m11)==1):

m11 = '0'+m11

fDt = d11+'-'+m11+'-'+y11

return fDt

class Balance:

def \_\_init\_\_(self):

# Instance attributes of Balance.dat data file

self.Cast\_Code = 0 # cassette/CD code to be balance

self.Cast\_Bal = 0 # Total number of cassettes/CDs in balance

self.Cast\_Price = 0 # Unit price of cassettes/CDs on code wise

self.dd = self.mm = self.yy = 0 # Balance date

def Give\_Balance(self, C\_Code):

Tbalance = 0

if not os.path.isfile(File2):

# When file does not exit

return False

else:

Brec = list() # A list to extract record from Balance.dat

Tbalance = 0

Bobj = open(File2, 'rb')

try:

while True:

BRec = load(Bobj)

if (C\_Code == BRec[0]):

Tbalance = BRec[1] # E.g. Cast\_Bal

break;

except EOFError:

pass

Bobj.close()

return Tbalance

def AddUpdateBalance(self, CList, CPrice):

# To know the balance cassette in 'Balance.dat'

Cbalance = Balance.Give\_Balance(self, CList[0])

if (Cbalance == False): # If file does not exist, add the record for first time

BRec = list()

with open(File2, 'ab') as Bobj:

BRec.append(CList[0]) # Cast\_Code

BRec.append(CList[1]) # Cast\_Bal

BRec.append(CPrice) # Cast\_Price

BRec.append(CList[2]) # Day

BRec.append(CList[3]) # Month

BRec.append(CList[4]) # Year

dump(BRec, Bobj)

elif (Cbalance >= 0):

Bobj = open(File2, 'rb')

Tobj = open("Temp.dat", 'wb')

try:

while True:

BRec = list() # A list to extract record from Balance.dat

BRec = load(Bobj)

if (CList[0] != BRec[0]):

# Write data into Temp.dat file

dump(BRec, Tobj)

else:

BRec[1] = Cbalance + CList[1]

#self.Cast\_Bal = self.Cast\_Bal + Cbalance

dump(BRec, Tobj)

except EOFError:

pass

Tobj.close()

Bobj.close()

os.remove("Balance.dat")

os.rename("Temp.dat", "Balance.dat")

def UpdateBalance(self, CList):

Bobj = open(File2, 'rb')

Tobj = open("Temp.dat", 'wb')

try:

while True:

BRec = list() # A list to extract record from Balance.dat

BRec = load(Bobj)

if (CList[0] != BRec[0]):

# Write data into Temp.dat file

dump(BRec, Tobj)

else:

BRec[1] = BRec[1] - CList[4]

dump(BRec, Tobj)

except EOFError:

pass

Tobj.close()

Bobj.close()

os.remove("Balance.dat")

os.rename("Temp.dat", "Balance.dat")

print('Balance.dat updated')

def Balance\_Cassettes(self):

M = Master()

if not os.path.isfile(File2):

print (File2, "file does not exist")

else:

TAmount = 0

print ("\nBalance Stock Register (Cassette/CD)")

print ("=" \* 35)

Bobj = open(File2, 'rb')

print ("{0:>5} {1:<26} {2:<20} {3:>10} {4:>8} {5:>10}"

.format("Code", "Name", "Company Name", "Quantity", "Price", "Amount"))

print ("-" \* 86)

try:

while True:

BRec = []

BRec = load(Bobj)

TRec = list()

Flag, TRec = M.Check\_Code(BRec[0])

if (Flag == True):

Amount = BRec[1] \* BRec[2]

TAmount = TAmount + Amount

print ("{0:>5} {1:<26} {2:<20} {3:>10} {4:>8.2f} {5:>10.2f}"

.format(BRec[0], TRec[1], TRec[2], BRec[1], BRec[2], Amount))

except EOFError:

pass

print ("-" \* 86)

print ("%s Total Amount: %s %.2f" % (' ' \* 56, ' ' \* 4, TAmount))

Bobj.close()

class Customer:

def \_\_init\_\_(self):

# Instance attributes of Customer.dat data file

self.Cast\_Code = 0 # cassette/CD code

self.C\_Name = '' # Customer name

self.C\_Address = '' # Customer address

self.C\_MPhone = 0 # Customer mobile no.

self.No\_Of\_Cast = 0 # Number of Cassette/CD

self.dd = self.mm = self.yy = 0 # Sale date

def Cassette\_Sale(self):

M = Master()

B = Balance()

CDt = Cast\_Date()

self.dd = CDt.dd

self.mm = CDt.mm

self.yy = CDt.yy

Cbalance = 0

print("Customer sales cassette/CD");

ch ='Y'

while ch=='Y':

TRec = list() # A temporary list to store master record

Flag = False # To check if Cast\_Code is in Master.dat or not

print("Date: %s-%s-%s" % (CDt.dd, CDt.mm, CDt.yy))

while True:

self.Cast\_Code = int(input("Cassette/CD Code (1/2/3/...) # "))

# Function call to check cassette/CD code in Master.dat

Flag, TRec = M.Check\_Code(self.Cast\_Code)

Cbalance = B.Give\_Balance(self.Cast\_Code)

if (Flag == True):

self.Cast\_Name = TRec[1] # Title of the cassette/CD

self.Cast\_Comp = TRec[2] # cassette/CD company

self.Cast\_Price = TRec[3] # Price per cassette/CD

print("Cassette/CD Name :", self.Cast\_Name)

print("Company Name : ", self.Cast\_Comp)

print("Individual Cassette/CD price : ",self.Cast\_Price)

print('\nEnter Customer details')

self.C\_Name = input("Customer name: ").upper()

self.C\_Address = input("Customer addres: ")

self.C\_MPhone = int(input("Customer mobile no.: "))

while True:

self.No\_Of\_Cast = int(input("Enter sales cassettes/CDs nos.: "))

if (self.No\_Of\_Cast > Cbalance):

print("Out of Stock");

else:

break

ch = input("Sales confirm <Y/N>: ").upper()

if ch == 'Y':

CustList = list()

with open(File3, 'ab') as CustObj:

if not CustObj:

print (File3, "does not created")

else:

# Appends data into a sequnce object

CustList.append(self.Cast\_Code)

CustList.append(self.C\_Name)

CustList.append(self.C\_Address)

CustList.append(self.C\_MPhone)

CustList.append(self.No\_Of\_Cast)

CustList.append(self.dd)

CustList.append(self.mm)

CustList.append(self.yy)

B.UpdateBalance(CustList)

dump(CustList, CustObj)

ch = input("More sale? <Y/N>: ")

ch = ch.upper()

if ch!='Y':

break

# Function to search individual customer on mobile no.

def Return\_CustomerName(self, Mno):

M = Master()

CName = ''

if not os.path.isfile(File2):

print (File3, "file does not exist")

else:

CustObj = open(File3, 'rb')

try:

while True:

CustRec = []

CustRec = load(CustObj)

if Mno == CustRec[3]:

CName = CustRec[1]

break

except EOFError:

pass

CustObj.close()

return CName

# Function to display Sales report for a particular month in a calender year.

def MonthlySales\_Report(self):

M = Master()

if not os.path.isfile(File2):

print (File3, "file does not exist")

else:

monthNo = int(input('Enter month no.: '))

yearNo = int(input('Enter year: '))

CDt = Cast\_Date()

self.dd = CDt.dd

self.mm = CDt.mm

self.yy = CDt.yy

if (monthNo <= 12 and monthNo <= self.mm and yearNo <= self.yy):

# Function call for a character month

MonthName = Month\_Name(monthNo)

TAmount = 0

# Function called to set the date as DD-MM-YYYY

nDt = Set\_DateFormat(self.dd, self.mm, self.yy)

print ("\nCustomer Sales Status Report - Date:", nDt)

print ("For the month of", MonthName, yearNo)

print ("=" \* 27)

CustObj = open(File3, 'rb')

print ("{0:<20} {1:<12} {2:<25} {3:^10} {4:>5} {5:>12} {6:>8}"

.format("Name", "Mobile No.", "Cassette/CD Cide & Name", "Date", "Qty", "Unit Price", "Amount"))

print ("-" \* 100)

try:

while True:

CustRec = []

CustRec = load(CustObj)

TRec = list()

Flag, TRec = M.Check\_Code(CustRec[0])

UPrice = TRec[3]

Amount = (UPrice + (UPrice \* 0.20)) \* CustRec[4] # An additional 20% of Unit price

nDt = Set\_DateFormat(CustRec[5], CustRec[6], CustRec[7])

if (monthNo == CustRec[6] and yearNo == CustRec[7]):

Clength = str(CustRec[0])+'-'+TRec[1]

nName = ''

for i in range(len(Clength)): # Extracts only 24 characters

nName = nName + Clength[i]

if i == 23:

break

print ("{0:20} {1:<12} {2:<25} {3:>10} {4:>5.0f} {5:>12.2f} {6:>8.2f}"

.format(CustRec[1], CustRec[3], nName, nDt, CustRec[4], UPrice, Amount))

except EOFError:

pass

print ("-" \* 100)

print('Note. Amount is calculated as 20% extra on unit price.')

#print ("%s Total Amount: %s %.2f" % (' ' \* 50, ' ' \* 4, TAmount))

CustObj.close()

else:

print ("Month no. and year is not valid")

# Function to display cose wise monthly sales report.

def CodeWiseMonthlySales\_Report(self):

M = Master()

TRec = list() # A temporary list to store master record

Flag = False # To check if Cast\_Code is in Master.dat or not

if not os.path.isfile(File2):

print (File3, "file does not exist")

else:

CCode = int(input("Cassette/CD Code (1/2/3/...) # "))

monthNo = int(input('Enter month no.: '))

yearNo = int(input('Enter year: '))

CDt = Cast\_Date()

self.dd = CDt.dd

self.mm = CDt.mm

self.yy = CDt.yy

# Function call to check cassette/CD code in Master.dat

Flag, TRec = M.Check\_Code(CCode)

if (monthNo <= 12 and monthNo <= self.mm and yearNo <= self.yy and Flag == True):

CName = TRec[1] # Title of the cassette/CD

CComp = TRec[2] # cassette/CD company

CPrice = TRec[3] # Price per cassette/CD

# Function call for a character month

MonthName = Month\_Name(monthNo)

TAmount = 0

# Function called to set the date as DD-MM-YYYY

nDt = Set\_DateFormat(self.dd, self.mm, self.yy)

print ("\nCode wise Sales Report - Date:", nDt)

print ("For the month of", MonthName, yearNo)

print ("Cassette/CD Code: %d Name: %s" % (CCode, CName))

print ("=" \* 40)

CustObj = open(File3, 'rb')

print ("{0:<20} {1:<12} {2:^10} {3:>5} {4:>12} {5:>8}"

.format("Customer Name", "Mobile No.", "Date", "Qty", "Unit Price", "Amount"))

print ("-" \* 74)

ctr = 0

try:

while True:

CustRec = []

CustRec = load(CustObj)

TRec = list()

Flag, TRec = M.Check\_Code(CustRec[0])

UPrice = TRec[3]

Amount = (UPrice + (UPrice \* 0.20)) \* CustRec[4] # An additional 20% of Unit price

nDt = Set\_DateFormat(CustRec[5], CustRec[6], CustRec[7])

if (monthNo == CustRec[6] and yearNo == CustRec[7] and CCode == CustRec[0]):

ctr += 1

print ("{0:20} {1:<12} {2:>10} {3:>5.0f} {4:>12.2f} {5:>8.2f}"

.format(CustRec[1], CustRec[3], nDt, CustRec[4], UPrice, Amount))

except EOFError:

pass

print ("-" \* 74)

if (ctr == 0):

print('No record found on such Code No., Month and Year')

else:

print('Note. Amount is calculated as 20% extra on unit price.')

#print ("%s Total Amount: %s %.2f" % (' ' \* 50, ' ' \* 4, TAmount))

CustObj.close()

else:

print ("Either Code not found or Month no. and year is not valid")

# Function to search individual customer on mobile no.

def CustomerWithMobileSearch(self):

M = Master()

if not os.path.isfile(File2):

print (File3, "file does not exist")

else:

MobileNo = int(input('\nEnter customer mobile no.: '))

Cust\_Name = self.Return\_CustomerName(MobileNo)

TAmount = 0

CDt = Cast\_Date()

self.dd = CDt.dd

self.mm = CDt.mm

self.yy = CDt.yy

# Function called to set the date as DD-MM-YYYY

nDt = Set\_DateFormat(self.dd, self.mm, self.yy)

print ("\nDate:", nDt)

print ("Customer name:", Cust\_Name, '& Mobile No.:', MobileNo)

print ("=" \* 40)

CustObj = open(File3, 'rb')

print ("{0:<30} {1:^10} {2:>5} {3:>12} {4:>8}"

.format("Cassette/CD", "Date", "Qty", "Unit Price", "Amount"))

print ("-" \* 70)

ctr = 0

try:

while True:

CustRec = []

CustRec = load(CustObj)

TRec = list()

Flag, TRec = M.Check\_Code(CustRec[0])

UPrice = TRec[3]

Amount = (UPrice + (UPrice \* 0.20)) \* CustRec[4] # An additional 20% of Unit price

nDt = Set\_DateFormat(CustRec[5], CustRec[6], CustRec[7])

if (MobileNo == CustRec[3]):

ctr += 1

Clength = str(CustRec[0])+'-'+TRec[1]

nName = ''

for i in range(len(Clength)): # Extracts only 24 characters

nName = nName + Clength[i]

if i == 23:

break

print ("{0:<30} {1:>10} {2:>5.0f} {3:>12.2f} {4:>8.2f}"

.format(nName, nDt, CustRec[4], UPrice, Amount))

except EOFError:

pass

print ("-" \* 70)

if (ctr == 0):

print('No record found on such mobile no.')

else:

print('Note. Amount is calculated as 20% extra on unit price.')

#print ("%s Total Amount: %s %.2f" % (' ' \* 50, ' ' \* 4, TAmount))

CustObj.close()

# Function to find a character month on against a month no.

def Month\_Name(mNo):

mDict = {1:'January', 2:'February', 3:'March',

4:'April', 5:'May', 6:'June', 7:'July',

8:'August', 9:'September', 10:'October',

11:'November', 12:'December'}

mName = ''

for key, value in mDict.items():

if (key == mNo):

mName = value

break

return mName

def main():

M = Master()

CS = Cassettes()

BL = Balance()

Cust = Customer()

while True:

print()

print ("\n Video Library Main Menu")

print ("-" \* 30)

print ("| 1 - > Master Cassettes/CDs |")

print ("| 2 - > Stock Cassettes/CDs |")

print ("| 3 - > Customer Sales |")

print ("| 4 - > Exit |")

print ("-" \* 30)

opt = input ("Enter your choice: ")

if opt == 1:

while True:

print()

print ("\n\tMaster Cassette Menu")

print ("-" \* 35)

print ("| 1 - > Cassettes/CDs Stock Entry |")

print ("| 2 - > View Cassettes/CDs |")

print ("| 3 - > Exit |")

print ("-" \* 35)

ch = input("enter your choice :")

if ch == 1:

M.Master\_Entry()

elif ch == 2:

M.Master\_Display()

elif ch == 3:

break

if opt == 2:

while True:

print()

print ("\n\tStock Cassette Menu")

print ("-" \* 33)

print ("| 1 - > Cassette/CD Stock entry |")

print ("| 2 - > Display Cassette/CD |")

print ("| 3 - > Stock/Balance Cassettes |")

print ("| 4 - > Exit |")

print ("-" \* 33)

ch = input("enter your choice: ")

if ch == 1:

CS.New\_Cassettes()

elif ch == 2:

CS.Display\_Cassettes()

elif ch == 3:

BL.Balance\_Cassettes()

elif ch == 4:

break

elif opt == 3:

while True:

print()

print ("\n\tCustomer Sales Menu")

print ("-" \* 34)

print ("| 1 - > Sales Entry |")

print ("| 2 - > Monthly Sales Report |")

print ("| 3 - > Code Wise Monthly Sales |")

print ("| 4 - > Customer Mobile No. Wise |")

print ("| 5 - > Exit |")

print ("-" \* 34)

ch = input("enter your choice: ")

if ch == 1:

Cust.Cassette\_Sale()

elif ch == 2:

Cust.MonthlySales\_Report()

elif ch == 3:

Cust.CodeWiseMonthlySales\_Report()

elif ch == 4:

Cust.CustomerWithMobileSearch()

elif ch == 5:

break

elif opt == 4:

break

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Output(Report):**

Cassette /CD Master Report:

Cassette/CD Master Report

=========================

Code Cassette/CD Name Company Name Price

-----------------------------------------------------------------------------------------------------------

1 Queen Absolutely Greatest Sony 120.00

2 Best of The Beatles Sony 200.00

3 Best of David Bowie Sony 150.00

4 Rollin’ stones Sony 180.00

-----------------------------------------------------------------------------------------------------------

Display cassette:

Cassette/CD entry Register

===========================

Code Name Company Name Quantity Price Date

-------------------------------------------------------------------------------------------------------------------------------

1 Queen Absolutely Greatest Sony 200 120.00 01-12-2016

2 Best of The Beatles Sony 150 200.00 05-12-2016

3 Best of David Bowie Sony 50 150.00 08-12-2016

4 Rollin’ stones Sony 50 180.00 02-12-2016

-------------------------------------------------------------------------------------------------------------------------------

Stock/Balance cassettes:

Balance Stock Register(Cassette/CD)

===================================

Code Name Company Name Quantity Price Amount

-------------------------------------------------------------------------------------------------------------------------------

1 Queen Absolutely Greatest Sony 250 120.00 30000.00

2 Best of The Beatles Sony 200 200.00 40000.00

3 Best of David Bowie Sony 100 150.00 15000.00

4 Rollin’ stones Sony 100 180.00 18000.00

-------------------------------------------------------------------------------------------------------------------------------

Monthly Sales Report:

Customer Sales Status Report – Date:25-12-2016

For the month of December 2016

===============================

Name Mobile No. Cassette CD code & Name Date Qty Unit Price Amount

-------------------------------------------------------------------------------------------------------------------------------

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

-------------------------------------------------------------------------------------------------------------------------------

Code Wise Monthly Sales:

Code Wise Sales Report – Date: 25-12-2016

For the month of December 2016

Cassette/CD Code: 2 Name: Best of The Beatles

==========================================

Customer Name Mobile No. Date Qty Unit Price Amount

-----------------------------------------------------------------------------------------------------------

Debashish 5678934576 25-12-2016 2 200 480.00

Makarios 9870564732 13-12-2016 2 200 480.00

-----------------------------------------------------------------------------------------------------------

Note. Amount is calculated as 20% extra on unit price

Customer Mobile No. Wise:

Date: 25-12-2016

Customer Name: Trisha Majumdar & Mobile No.: 7869546283

===================================================

Cassette/CD Date Qty Unit Price Amount

-----------------------------------------------------------------------------------------------------------

2-Best of The Beatles 25-12-2016 2 200 480.00

4-Rollin’ Stones 13-12-2016 2 180 432.00

-----------------------------------------------------------------------------------------------------------

Note. Amount is calculated as 20% extra on unit price