**ST. XAVIER’S COLLEGE (AUTONOMOUS)**

**M Sc (Big Data Analytics)**

**Semester-III**

**3802- Enabling Technologies for Data Science**

**Roll No. : 20-PBD-002**

**Shraddha P Jain**

**Sparks Assignment**

**Set-2**

Table : Item

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Itemcode | Itemdesc | Price | Qty\_on\_hand(in Pcs.) | Reorder\_level |
| I001 | Round-neck T-shirt | 235.00 | 500 | 200 |
| I002 | Hooded T-shirt | 543.00 | 100 | 50 |
| I003 | Polo-neck T-shirt | 345.00 | 50 | 25 |
| I004 | Turtle-neck T-shirt | 450.00 | 50 | 25 |

Table : Customer

|  |  |  |  |
| --- | --- | --- | --- |
| Custcode | CustName | Address | Balance |
| C1 | A.R. Patel | 23, Acme house | 5000.00 |
| C2 | S.M. Bhaduria | Navrangpura | 2000.00 |

# Table : Invoice

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Invoiceno | Invdate | Custcode | Itemno | Qty | Price |
| 1034 | 23-03-1999 | C1 | I004 | 200 | 450.00 |
| 1034 | 23-03-1999 | C1 | I003 | 50 | 345.00 |

val ItemList = List(("I001","Round-neck T-Shirt",235.00,500,200),("I002","Hooded T-shirt", 543.00, 100,50),("I003","Polo-neck T-shirt",345.00,50,25),("I004","Turtle-neck T-shirt", 450.00,50,25))

val Item = ItemList.toDF("Itemcode","Itemdesc","Price","Qty","Reorderlevel")

Item.show()

val CustoemrList = List(("C1","A.R.Patel","23, Acme house",5000.00),("C2","S.M.Bhaduria","Navrangpura",2000.00))

val Customer = CustoemrList.toDF("Custcode","Custname","Address","Balance")

Customer.show()

val InvoiceList = List((1034,"1999-03-23","C1","I004",200,450.00),(1034,"1999-03-23","C1","I003",50,345.00))

val Invoice = InvoiceList.toDF("Invoiceno","Invdate","Custcode","Itemno","Qty","Price")

Invoice.show()

1. **Display invoice wise total price in descending order of invoiceno.**

Invoice.groupBy("Invoiceno").sum().select("Invoiceno","sum(Price)").sort(desc("Invoiceno")).show()

1. **Display number of items that are monthly sold.**

val ans2 = Invoice.withColumn("Invdate",(col("Invdate").cast("date")))

val ans2\_1 = ans2.withColumn("Month",from\_unixtime(unix\_timestamp($"Invdate"

,"yyyy-MM-dd"),"MMMMM"))

ans2\_1.groupBy("Month").sum().select("Month","sum(Qty)").show()

1. **Print all the records of ITEM table for which quantity on hand is less than reordered level.**

ItemDF.filter("Qty < Reordedlevel").show

1. **Display customer wise total purchase.**

Invoice.groupBy("Custcode").sum().select("Custcode","sum(Price)").join(Custo

mer,"Custcode").show()

1. **Display the name of the customers who live in a place that starts with the letter N.**

Customer.filter("Address like 'N%'" ).show()

**Scala Assignment**

1. **Write a Scala function to find the Max of three numbers.**

object ans1 {

def test(a: Int, b: Int, c: Int): Int = {

List(a,b,c).max

}

def main(args: Array[String]): Unit = {

println("Result: " + test(1,2,5));

println("Result: " + test(5,25,10));

}

}

1. **Write a Scala program to reverse a string**

object ans2 extends App

{

val x = "BDA Exam"

println(x)

println(x.reverse)

}

1. **Write a Scala program that accepts an integer (n) and computes the value of n+nn+nnn.**

import java.util.Scanner;

object ans3{

def main(args: Array[String]) {

var scanner = new Scanner(System.in);

println("Enter the integer n : ");

var n = scanner.nextInt();

println("The Result is : "+(n+n\*n+n\*n\*n));

}

}

1. **Write a Scala program to convert temperatures to and from celsius, fahrenheit. [ Formula : c/5 = f-32/9 [ where c = temperature in celsius and f = temperature in fahrenheit ] *Expected Output* : 60°C is 140 in Fahrenheit 45°F is 7 in Celsius**

object ans4{

def main(args: Array[String]) {

print(ans\_4func("C",36))

}

def ans4\_func (un:String,v:Double) {

if(un=="C"){

print(v+“ in C is ”+(v \* 9 / 5 + 32).round + “in F”)

}

else if(un=="F"){

print(v+“ in F is ” +((v - 32) \* 5 / 9).round+ “in C”))

}

else{

print()

}

}

}

1. **Write a Scala program to get the volume of a sphere with radius 6.(V = 4/3 × π × r3)**

object ans extends App{

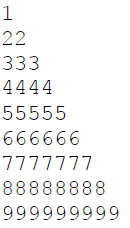
val radius = 6

val volume = (4\*22\*radius\*radius\*radius)/3\*7

println(" Volume is:" + volume)

}

1. Write a Scala program to construct the following pattern, using a nested loop number.



object ans6 {

def main(args: Array[String]) {

for(i <- 0 to 8){

for(j <- 0 to i){

print(i+1)

}

println("")

}

}

}

1. Write a Scala program to test whether a passed letter is a vowel or not.

object ans7 {

def main(args: Array[String]) {

var a:Char=0;

print("Enter character: ")

a=scala.io.StdIn.readChar()

a match{

case 'A'=>printf("%c is a VOWEL.\n",a);

case 'E'=>printf("%c is a VOWEL.\n",a);

case 'I'=>printf("%c is a VOWEL.\n",a);

case 'O'=>printf("%c is a VOWEL.\n",a);

case 'U'=>printf("%c is a VOWEL.\n",a);

case 'a'=>printf("%c is a VOWEL.\n",a);

case 'e'=>printf("%c is a VOWEL.\n",a);

case 'i'=>printf("%c is a VOWEL.\n",a);

case 'o'=>printf("%c is a VOWEL.\n",a);

case 'u'=>printf("%c is a VOWEL.\n",a);

case \_=>printf("%c is a CONSONANT.\n",a);

}

}

}

1. Write a Scala program to find numbers between 100 and 400 (both included) where each digit of a number is an even number. The numbers obtained should be printed in a comma-separated sequence.
2. Write a Scala program that accepts a number from the user and checks whether it is an amrstrong number or not. 13+53+33=153is an amrstrong number

object ans9{

def main(args: Array[String]){

println(isArmstrongNumber(153))

}

def isArmstrongNumber(n: Long): Boolean = {

val power = n.toString.length

var sum = n

var number = n

while (number > 0) {

val d = number % 10

sum -= math.pow(d, power).longValue()

if (sum < 0) {

return false

}

number /= 10

}

sum == 0

}

}

1. Write a Scala program to get the Fibonacci series between 0 to 50. Note: The Fibonacci Sequence is the series of numbers: 0, 1, 1, 2, 3, 5, 8, 13, 21, ....

object ans10{

def fibonacci( n : Int ) : Unit = {

var a = 0

var b = 1

var i = 0

var c=0

while( c < n ) {

c = a + b

a = b

b = c

i = i + 1

print(a+", ")

}

}

def main (args: Array[String]) : Unit ={

fibonacci(50)

}

}