1. FizzBuzz

```
using System;

public class Program
{
   public static void Main()
   {
      int n=int.Parse(Console.ReadLine());
      for (int i = 1; i <= n; i++)
      {
         if (i % 3 == 0 && i % 5 == 0)
            Console.WriteLine("FizzBuzz");
         else if (i % 3 == 0)
            Console.WriteLine("Fizz");
         else if (i % 5 == 0)
            Console.WriteLine("Buzz");
         else
            Console.WriteLine(i);
      }
   }
}
using System;

class Program
{
   static void Main()
```

```
  {
    int n = int.Parse(Console.ReadLine());
    bool prime = n > 1;
    for (int i = 2; i * i <= n && prime; i++)
      if (n % i == 0)
        prime = false;

    if (prime)
        Console.WriteLine("Prime");
    else
    {
      Console.Write("Not Prime. Factors: ");
      for (int i = 1; i <= n; i++)
        if (n % i == 0)
        Console.Write(i + " ");
    }
  }
}
```

3.Sum of digits

```
using System;

class Program
{
  static void Main()
  {
    Console.Write("Enter a number: ");
```

```
        int num = int.Parse(Console.ReadLine());

        while (num >= 10)
        {
            int sum = 0;
            while (num > 0)
            {
                sum += num % 10;
                num /= 10;
            }
            num = sum;
        }

        Console.WriteLine("Single digit sum is: " + num);
    }
}
```

4.Reverse number

```
using System;

class Program
{
    static void Main()
    {
        Console.Write("Enter a number: ");
        int num = int.Parse(Console.ReadLine());
        int original = num;
```

```
        int reversed = 0;

        while (num > 0)
        {
            int digit = num % 10;
            reversed = reversed * 10 + digit;
            num /= 10;
        }

        Console.WriteLine("Reversed number: " + reversed);

        if (original == reversed)
            Console.WriteLine("It is a palindrome.");
        else
            Console.WriteLine("It is not a palindrome.");
    }
}
```

5.GCD AND LCM

```
using System;

class Program
{
    static void Main()
    {
        Console.Write("Enter first number: ");
        int a = int.Parse(Console.ReadLine());
```

```
      Console.Write("Enter second number: ");
      int b = int.Parse(Console.ReadLine());

      int gcd = 1;

      for (int i = 1; i <= a && i <= b; i++)
      {
         if (a % i == 0 && b % i == 0)
            gcd = i;
      }

      int lcm = (a * b) / gcd;

      Console.WriteLine("GCD = " + gcd);
      Console.WriteLine("LCM = " + lcm);
   }
}
using System;

class Program
{
   static void Main()
   {

      Console.Write("Enter how many numbers: ");
      int n = int.Parse(Console.ReadLine());
```

```
int[] arr = new int[n];

Console.WriteLine("Enter the numbers:");
for (int i = 0; i < n; i++)
    arr[i] = int.Parse(Console.ReadLine());
int first = int.MinValue, second = int.MinValue;

foreach (int num in arr)
{
    if (num > first)
    {
        second = first;
        first = num;
    }
    else if (num > second && num != first)
    {
        second = num;
    }
}

if (n < 2 || second == int.MinValue)
    Console.WriteLine("There is no second largest element (all numbers might be same).");
else
    Console.WriteLine("Second largest number is: " + second);
```

```
    }



int first = int.MinValue, second = int.MinValue;


foreach (int num in arr)
{
    if (num > first)
    {
        second = first;
        first = num;
    }
    else if (num > second && num != first)
    {
        second = num;
    }
}
}
```

7.Remove duplicates

```
using System;


class Program
{
    static void Main()
    {
        string[] input = Console.ReadLine().Split();
```

```
        int n = input.Length;
        int[] arr = new int[n];

        for (int i = 0; i < n; i++)
            arr[i] = int.Parse(input[i]);

        for (int i = 0; i < n; i++)
        {
            bool isDuplicate = false;
            for (int j = 0; j < i; j++)
            {
                if (arr[i] == arr[j])
                {
                    isDuplicate = true;
                    break;
                }
            }
            if (!isDuplicate)
                Console.Write(arr[i] + "\t");
        }
    }
}
```

8.Rotate the array

```
using System;

class Program
```

```
{
   static void Main()
   {
      int[] arr = { 1, 2, 3, 4, 5 };
      Console.Write("Enter k (positions to rotate): ");
      int k = int.Parse(Console.ReadLine());
      int n = arr.Length;
      k = k % n;

      Console.Write("Rotated Array: ");
      for (int i = n - k; i < n; i++)
         Console.Write(arr[i] + " ");
      for (int i = 0; i < n - k; i++)
         Console.Write(arr[i] + " ");
   }
}
```

9.Find missing number

```
using System;

class Program
{
   static void Main()
   {
      int n = int.Parse(Console.ReadLine());
      string[] input = Console.ReadLine().Split();
```

```
        int sum = 0;
        for (int i = 0; i < input.Length; i++)
        {
            sum += int.Parse(input[i]);
        }

        int total = n * (n + 1) / 2;
        int missing = total - sum;

        Console.WriteLine(missing);
    }
}
```

10.check if array is sorted

```
using System;

class Program
{
    static void Main()
    {
        int[] arr = { 1, 2, 3, 4, 5 };
        bool ascending = true, descending = true;

        for (int i = 0; i < arr.Length - 1; i++)
        {
            if (arr[i] > arr[i + 1])
```

```
            ascending = false;
        if (arr[i] < arr[i + 1])
            descending = false;
    }

    if (ascending)
        Console.WriteLine("Array is sorted in ascending order.");
    else if (descending)
        Console.WriteLine("Array is sorted in descending order.");
    else
        Console.WriteLine("Array is not sorted.");
  }
}
```

11.Anagram checker

```
using System;

class Program
{
  static void Main()
  {
    Console.Write("Enter first word: ");
    string str1 = Console.ReadLine().ToLower();

    Console.Write("Enter second word: ");
    string str2 = Console.ReadLine().ToLower();
```

```
        char[] a = str1.ToCharArray();
        char[] b = str2.ToCharArray();

        Array.Sort(a);
        Array.Sort(b);

        string sorted1 = new string(a);
        string sorted2 = new string(b);

        if (sorted1 == sorted2)
            Console.WriteLine("The words are anagrams.");
        else
            Console.WriteLine("The words are not anagrams.");
    }
}
```

12.count vowels and consonants

```
using System;

class Program
{
    static void Main()
    {
        Console.Write("Enter a string: ");
        string str = Console.ReadLine().ToLower();

        int vowels = 0, consonants = 0;
```

```
        foreach (char c in str)
        {
            if ("aeiou".Contains(c))
                vowels++;
            else if (char.IsLetter(c))
                consonants++;
        }

        Console.WriteLine("Vowels: " + vowels);
        Console.WriteLine("Consonants:  " + consonants);
    }
}
```

13.longest word in sentence

```
using System;

class Program
{
    static void Main()
    {
        Console.Write("Enter a sentence: ");
        string sentence = Console.ReadLine();

        string[] words = sentence.Split(' ');
        string longest = "";
```

```
        foreach (string word in words)
        {
            if (word.Length > longest.Length)
                longest = word;
        }

        Console.WriteLine("Longest word: " + longest);
    }
}
```

14.Toggle case of each character

```
using System;

class Program
{
    static void Main()
    {
        Console.Write("Enter a string: ");
        string str = Console.ReadLine();
        string result = "";

        foreach (char c in str)
        {
            if (char.IsUpper(c))
                result += char.ToLower(c);
            else if (char.IsLower(c))
                result += char.ToUpper(c);
```

```
        else
            result += c;
        }

        Console.WriteLine("Toggled  string: " + result);
    }
}
```

15.Check Pangram

```
using System;

class Program
{
    static void Main()
    {
        Console.Write("Enter a sentence: ");
        string sentence = Console.ReadLine().ToLower();
        bool isPangram = true;

        for (char c = 'a'; c <= 'z'; c++)
        {
            if (!sentence.Contains(c))
            {
                isPangram = false;
                break;
            }
        }
```

```
        if (isPangram)
            Console.WriteLine("The sentence is a pangram.");
        else
            Console.WriteLine("The sentence is not a pangram.");
    }
}
```

16.Armstrong

```
using System;

class Program
{
    static void Main()
    {
        Console.Write("Enter a number: ");
        int num = int.Parse(Console.ReadLine());
        int temp = num, sum = 0;
        int digits = num.ToString().Length;

        while (temp > 0)
        {
            int digit = temp % 10;
            sum += (int)Math.Pow(digit, digits);
            temp /= 10;
        }
```

```
        if (sum == num)
            Console.WriteLine($"{num} is an Armstrong number.");
        else
            Console.WriteLine($"{num} is not an Armstrong number.");
    }
}
```

17.Fibonnaci series

```
using System;

class Program
{
    static void Main()
    {
        Console.Write("Enter number of terms: ");
        int n = int.Parse(Console.ReadLine());

        int a = 0, b = 1;

        Console.Write("Fibonacci Series: ");
        for (int i = 0; i < n; i++)
        {
            Console.Write(a + " ");
            int next = a + b;
            a = b;
            b = next;
        }
```

```
    }
}
```

18.Find all perfect number

```
using System;

class Program
{
    static void Main()
    {
        Console.Write("Enter a number (N): ");
        int n = int.Parse(Console.ReadLine());

        Console.WriteLine("Perfect numbers up to " + n + ":");
        for (int num = 1; num <= n; num++)
        {
            int sum = 0;
            for (int i = 1; i <= num / 2; i++)
            {
                if (num % i == 0)
                    sum += i;
            }
            if (sum == num)
                Console.Write(num + " ");
        }
    }
}
```

19.Convert decimal to binary and vice versa

```
using System;

class Program
{
   static void Main()
   {
      Console.WriteLine("1. Decimal to Binary");
      Console.WriteLine("2. Binary to Decimal");
      Console.Write("Choose  option: ");
      int choice = int.Parse(Console.ReadLine());

      if (choice == 1)
      {
         Console.Write("Enter decimal number: ");
         int num = int.Parse(Console.ReadLine());
         string binary = "";

         while (num > 0)
         {
            binary = (num % 2) + binary;
            num /= 2;
         }

         Console.WriteLine("Binary: " + binary);
      }
```

```
    else if (choice == 2)
    {
        Console.Write("Enter binary number: ");
        string bin = Console.ReadLine();
        int dec = 0, power = 0;

        for (int i = bin.Length - 1; i >= 0; i--)
        {
            if (bin[i] == '1')
                dec += (int)Math.Pow(2, power);
            power++;
        }

        Console.WriteLine("Decimal: " + dec);
    }
    else
    {
        Console.WriteLine("Invalid choice.");
    }
  }
}
```

20.Sum of prime number in range

```
using System;

class Program
{
```

```
static void Main()
{
   Console.Write("Enter start of range: ");
   int start = int.Parse(Console.ReadLine());
   Console.Write("Enter end of range: ");
   int end = int.Parse(Console.ReadLine());

   int sum = 0;

   for (int num = start; num <= end; num++)
   {
      if (IsPrime(num))
         sum += num;
   }

   Console.WriteLine("Sum of prime numbers = " + sum);
}

static bool IsPrime(int n)
{
   if (n < 2)
      return false;
   for (int i = 2; i <= n / 2; i++)
   {
      if (n % i == 0)
         return false;
```

```
    }
    return true;
  }
}
```

21.Pascal

```
using System;

class Program
{
  static void Main()
  {
    Console.Write("Enter number of rows: ");
    int n = int.Parse(Console.ReadLine());

    for (int i = 0; i < n; i++)
    {
      int num = 1;
      for (int j = 0; j <= i; j++)
      {
        Console.Write(num + " ");
        num = num * (i - j) / (j + 1);
      }
      Console.WriteLine();
    }
  }
}
```

22.Leader element in array

```
using System;

class Program
{
   static void Main()
   {
      int[] arr = { 16, 17, 4, 3, 5, 2 };
      int n = arr.Length;

      Console.Write("Leader elements: ");
      for (int i = 0; i < n; i++)
      {
         bool isLeader = true;
         for (int j = i + 1; j < n; j++)
         {
            if (arr[i] <= arr[j])
            {
               isLeader = false;
               break;
            }
         }
         if (isLeader)
            Console.Write(arr[i] + " ");
      }
   }
```

```
}
```

23.Matrix transpose

```
using System;

class Program
{
   static void Main()
   {
      int[,] matrix = {
         {1, 2, 3},
         {4, 5, 6},
         {7, 8, 9}
      };

      int rows = 3, cols = 3;
      int[,] transpose = new int[cols, rows];

      for (int i = 0; i < rows; i++)
         for (int j = 0; j < cols; j++)
            transpose[j, i] = matrix[i, j];

      Console.WriteLine("Transposed Matrix:");
      for (int i = 0; i < cols; i++)
      {
         for (int j = 0; j < rows; j++)
            Console.Write(transpose[i, j] + " ");
```

```
        Console.WriteLine();
      }
    }
}
```

24.Count frequency

```
using System;
using System.Collections.Generic;

class Program
{
   static void Main()
   {
      Console.Write("Enter a sentence: ");
      string sentence = Console.ReadLine().ToLower();

      string[] words = sentence.Split(' ', StringSplitOptions.RemoveEmptyEntries);
      Dictionary<string, int> freq = new Dictionary<string, int>();

      foreach (string word in words)
      {
         if (freq.ContainsKey(word))
            freq[word]++;
         else
            freq[word] = 1;
      }
```

```
        Console.WriteLine("Word Frequencies:");
        foreach (var item in freq)
            Console.WriteLine($"{item.Key}: {item.Value}");
    }
}
```

25.Suduko row validity

```
using System;
using System.Linq;

class Program
{
    static void Main()
    {
        int[] row = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };

        if (row.Length == 9 && row.Distinct().Count() == 9 && row.All(x => x >= 1 && x <= 9))
            Console.WriteLine("Valid Sudoku row.");
        else
            Console.WriteLine("Invalid Sudoku row.");
    }
}
```

26.Spiral matrix generator

```
using System;

class Program
{
```

```
static void Main()
{
    Console.Write("Enter n: ");
    int n = int.Parse(Console.ReadLine());
    int[,] matrix = new int[n, n];

    int top = 0, bottom = n - 1, left = 0, right = n - 1;
    int num = 1;

    while (top <= bottom && left <= right)
    {
        for (int i = left; i <= right; i++)
            matrix[top, i] = num++;
        top++;

        for (int i = top; i <= bottom; i++)
            matrix[i, right] = num++;
        right--;

        for (int i = right; i >= left; i--)
            matrix[bottom, i] = num++;
        bottom--;

        for (int i = bottom; i >= top; i--)
            matrix[i, left] = num++;
        left++;
```

```
        }

        Console.WriteLine("Spiral Matrix:");
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < n; j++)
                Console.Write(matrix[i, j] + "\t");
            Console.WriteLine();
        }
    }
}
```

27.Password generator

```
using System;

class Program
{
    static void Main()
    {
        Console.Write("Enter a password: ");
        string password = Console.ReadLine();

        bool hasUpper = false, hasLower = false, hasDigit = false, hasSpecial = false;

        foreach (char c in password)
        {
            if (char.IsUpper(c)) hasUpper = true;
```

```
        else if (char.IsLower(c)) hasLower = true;
        else if (char.IsDigit(c)) hasDigit = true;
        else if ("!@#$%^&*()".Contains(c)) hasSpecial = true;
      }

      if (password.Length >= 8 && hasUpper && hasLower && hasDigit && hasSpecial)
        Console.WriteLine("Strong password!");
      else
        Console.WriteLine("Weak password!");
    }
}
```

28.Number pyramid pattern

```
using System;

class Program
{
   static void Main()
   {
      Console.Write("Enter n: ");
      int n = int.Parse(Console.ReadLine());
      int num = 1;

      for (int i = 1; i <= n; i++)
      {
         Console.Write(i + " ");
         for (int j = 1; j <= i; j++)
```

```
            Console.Write(num++ + " ");
        Console.WriteLine();
      }
   }
}
```

29.Digit counter

```
using System;

class Program
{
   static void Main()
   {
      Console.Write("Enter a number: ");
      string num = Console.ReadLine();
      int[] freq = new int[10];

      foreach (char c in num)
      {
         if (char.IsDigit(c))
            freq[c - '0']++;
      }

      for (int i = 0; i < 10; i++)
      {
         if (freq[i] > 0)
            Console.WriteLine($"Digit {i}: {freq[i]} time(s)");
```

```
    }
  }
}
```

30.Magic square

```
using System;

class Program
{
  static void Main()
  {
    int[,] matrix = {
      {2, 7, 6},
      {9, 5, 1},
      {4, 3, 8}
    };
    int n = 3;
    int sum = 0;
    for (int j = 0; j < n; j++)
      sum += matrix[0, j];

    bool isMagic = true;

    // Check rows and columns
    for (int i = 0; i < n; i++)
    {
      int rowSum = 0, colSum = 0;
```

```
        for (int j = 0; j < n; j++)
        {
            rowSum += matrix[i, j];
            colSum += matrix[j, i];
        }
        if (rowSum != sum || colSum != sum)
            isMagic = false;
    }

    // Check diagonals
    int diag1 = 0, diag2 = 0;
    for (int i = 0; i < n; i++)
    {
        diag1 += matrix[i, i];
        diag2 += matrix[i, n - i - 1];
    }
    if (diag1 != sum || diag2 != sum)
        isMagic = false;

    Console.WriteLine(isMagic ? "Magic Square!" : "Not a Magic Square.");
  }
}
```

SQL

1.List all orders with customer names and product names.

SELECT

```
    o.OrderID,
    c.CustomerName,
    p.ProductName,
    o.OrderDate,
    o.Quantity
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
JOIN Products p ON o.ProductID = p.ProductID;
```

2. Find all customers who placed an order for "Product_14".

```
SELECT DISTINCT c.CustomerName, c.City
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
JOIN Products p ON o.ProductID = p.ProductID
WHERE p.ProductName = 'Product_14';
```

3.Show total quantity ordered per customer.

```
SELECT
    c.CustomerName,
    SUM(o.Quantity) AS TotalQuantity
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
GROUP BY c.CustomerName;
```

4. List customers who have never placed an order.

```
SELECT c.CustomerName, c.City
```

```
FROM Customers c
LEFT JOIN Orders o ON c.CustomerID = o.CustomerID
WHERE o.CustomerID IS NULL;
```

5. Find products that have never been ordered.

```
SELECT p.ProductName, p.Price
FROM Products p
LEFT JOIN Orders o ON p.ProductID = o.ProductID
WHERE o.ProductID IS NULL;
```

6. Show total revenue per product (Price × Quantity).

```
SELECT
  p.ProductName,
  SUM(p.Price * o.Quantity) AS TotalRevenue
FROM Orders o
JOIN Products p ON o.ProductID = p.ProductID
GROUP BY p.ProductName
ORDER BY TotalRevenue DESC;
```

7. List orders placed in the last 90 days with customer and product details.

```
SELECT
  o.OrderID, c.CustomerName, p.ProductName, o.OrderDate
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
JOIN Products p ON o.ProductID = p.ProductID
WHERE o.OrderDate >= DATE_SUB(CURDATE(), INTERVAL 90 DAY);
```

8. Find the most frequently ordered product.

```
SELECT
   p.ProductName,
   COUNT(*) AS OrderCount
FROM Orders o
JOIN Products p ON o.ProductID = p.ProductID
GROUP BY p.ProductName
ORDER BY OrderCount  DESC
LIMIT 1;
```

09. Show customers who ordered more than 3 different products.

```
SELECT
   c.CustomerName,
   COUNT(DISTINCT o.ProductID) AS ProductCount
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
GROUP BY c.CustomerName
HAVING COUNT(DISTINCT o.ProductID) > 3;
```

10. List cities with total number of orders placed.

```
SELECT
   c.City,
   COUNT(o.OrderID) AS TotalOrders
```

```
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
GROUP BY c.City;
```

11. Find customers who ordered products priced above 50.

```
SELECT DISTINCT c.CustomerName, p.ProductName, p.Price
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
JOIN Products p ON o.ProductID = p.ProductID
WHERE p.Price > 50;
```

12. Show products ordered by customers from "Chicago".

```
SELECT DISTINCT p.ProductName
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
JOIN Products p ON o.ProductID = p.ProductID
WHERE c.City = 'Chicago';
```

13. List customers who ordered the same product more than once.

```
SELECT
   c.CustomerName,
   p.ProductName,
   COUNT(*) AS TimesOrdered
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
JOIN Products p ON o.ProductID = p.ProductID
```

```
GROUP BY c.CustomerName, p.ProductName
HAVING COUNT(*) > 1;
```

14. Show the latest order date for each customer.

```
SELECT
  c.CustomerName,
  MAX(o.OrderDate) AS LatestOrderDate
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
GROUP BY c.CustomerName;
```

15. Find the top 3 customers by total spending.

```
SELECT
  c.CustomerName,
  SUM(p.Price * o.Quantity) AS TotalSpent
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
JOIN Products p ON o.ProductID = p.ProductID
GROUP BY c.CustomerName
ORDER BY TotalSpent DESC
LIMIT 3;
```

16. List products ordered by more than 5 different customers.

```
SELECT
  p.ProductName,
  COUNT(DISTINCT o.CustomerID) AS UniqueCustomers
```

```
FROM Orders o
JOIN Products p ON o.ProductID = p.ProductID
GROUP BY p.ProductName
HAVING COUNT(DISTINCT o.CustomerID) > 5;
```

17. Show average quantity ordered per product.

```
SELECT
   p.ProductName,
   AVG(o.Quantity) AS AvgQuantity
FROM Orders o
JOIN Products p ON o.ProductID = p.ProductID
GROUP BY p.ProductName;
```

18. Find customers who ordered both "Product_1" and "Product_25".

```
SELECT c.CustomerName
FROM Customers c
JOIN Orders o ON c.CustomerID = o.CustomerID
JOIN Products p ON o.ProductID = p.ProductID
WHERE p.ProductName IN ('Product_1', 'Product_25')
GROUP BY c.CustomerName
HAVING COUNT(DISTINCT p.ProductName) = 2;
```

19. List all orders with product price and total cost.

```
SELECT
   o.OrderID,
   c.CustomerName,
```

```
    p.ProductName,
    p.Price,
    o.Quantity,
    (p.Price * o.Quantity) AS TotalCost
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
JOIN Products p ON o.ProductID = p.ProductID;
```

20. Show customers who ordered products from multiple cities

```
SELECT
    c.CustomerName,
    COUNT(DISTINCT pr.City) AS CityCount
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
JOIN Products p ON o.ProductID = p.ProductID
JOIN ProductLocations pr ON p.ProductID = pr.ProductID  -- hypothetical table
GROUP BY c.CustomerName
HAVING COUNT(DISTINCT pr.City) > 1;
```

Subquery

1. Find customers who placed the highest quantity order.

```
SELECT CustomerName
FROM Customers
WHERE CustomerID IN (
    SELECT CustomerID
    FROM Orders
```

```
  WHERE Quantity = (SELECT MAX(Quantity) FROM Orders)
);
```

2. List products with price above the average product price.

```
SELECT ProductName, Price
FROM Products
WHERE Price > (SELECT AVG(Price) FROM Products);
```

3. Show customers whose total spending is above the average spending.

```
SELECT CustomerName, TotalSpending
FROM (
  SELECT
    c.CustomerName,
    SUM(p.Price * o.Quantity) AS TotalSpending
  FROM Orders o
  JOIN Customers c ON o.CustomerID = c.CustomerID
  JOIN Products p ON o.ProductID = p.ProductID
  GROUP BY c.CustomerName
) AS SpendingTable
WHERE TotalSpending > (
  SELECT AVG(TotalSpending)
  FROM (
    SELECT SUM(p.Price * o.Quantity) AS TotalSpending
    FROM Orders o
    JOIN Products p ON o.ProductID = p.ProductID
    GROUP BY o.CustomerID
  ) AS AvgTable
);
```

4. Find the most expensive product ordered.

```
SELECT ProductName, Price
FROM Products
WHERE Price = (
  SELECT MAX(p.Price)
  FROM Orders o
  JOIN Products p ON o.ProductID = p.ProductID
);
```

5. List customers who ordered the cheapest product.

```
SELECT DISTINCT c.CustomerName
FROM Customers c
JOIN Orders o ON c.CustomerID = o.CustomerID
WHERE o.ProductID = (
  SELECT ProductID
  FROM Products
  WHERE Price = (SELECT MIN(Price) FROM Products)
);
```

ANAGRAM(HACKEREARTH)

```
using System;

class Program
{
  static void Main()
  {
    int T = int.Parse(Console.ReadLine());
```

```
    for (int i = 0; i < T; i++)
    {
        string str1 = Console.ReadLine().ToLower();
        string str2 = Console.ReadLine().ToLower();

        if (IsAnagram(str1, str2))
            Console.WriteLine("YES");
        else
            Console.WriteLine("NO");
    }
}

static bool IsAnagram(string str1, string str2)
{
    if (str1.Length != str2.Length)
        return false;

    char[] a = str1.ToCharArray();
    char[] b = str2.ToCharArray();

    Array.Sort(a);
    Array.Sort(b);

    for (int i = 0; i < a.Length; i++)
    {
```

```
        if (a[i] != b[i])
            return false;
        }

        return true;
    }
}
```

MISSING NUMBER

```
using System;

class GfG {
    static int missingNum(int[] arr) {
        int n = arr.Length + 1;

        // Iterate from 1 to n and check
        // if the current number is present
        for (int i = 1; i <= n; i++) {
            bool found = false;
            for (int j = 0; j < n - 1; j++) {
                if (arr[j] == i) {
                    found = true;
                    break;
                }
            }

            if (!found)
```

```
            return i;
        }
        return -1;
    }

    static void Main() {
        int[] arr = { 8, 2, 4, 5, 3, 7, 1 };
        Console.WriteLine(missingNum(arr));
    }
}
```