DBS LAB EXAM

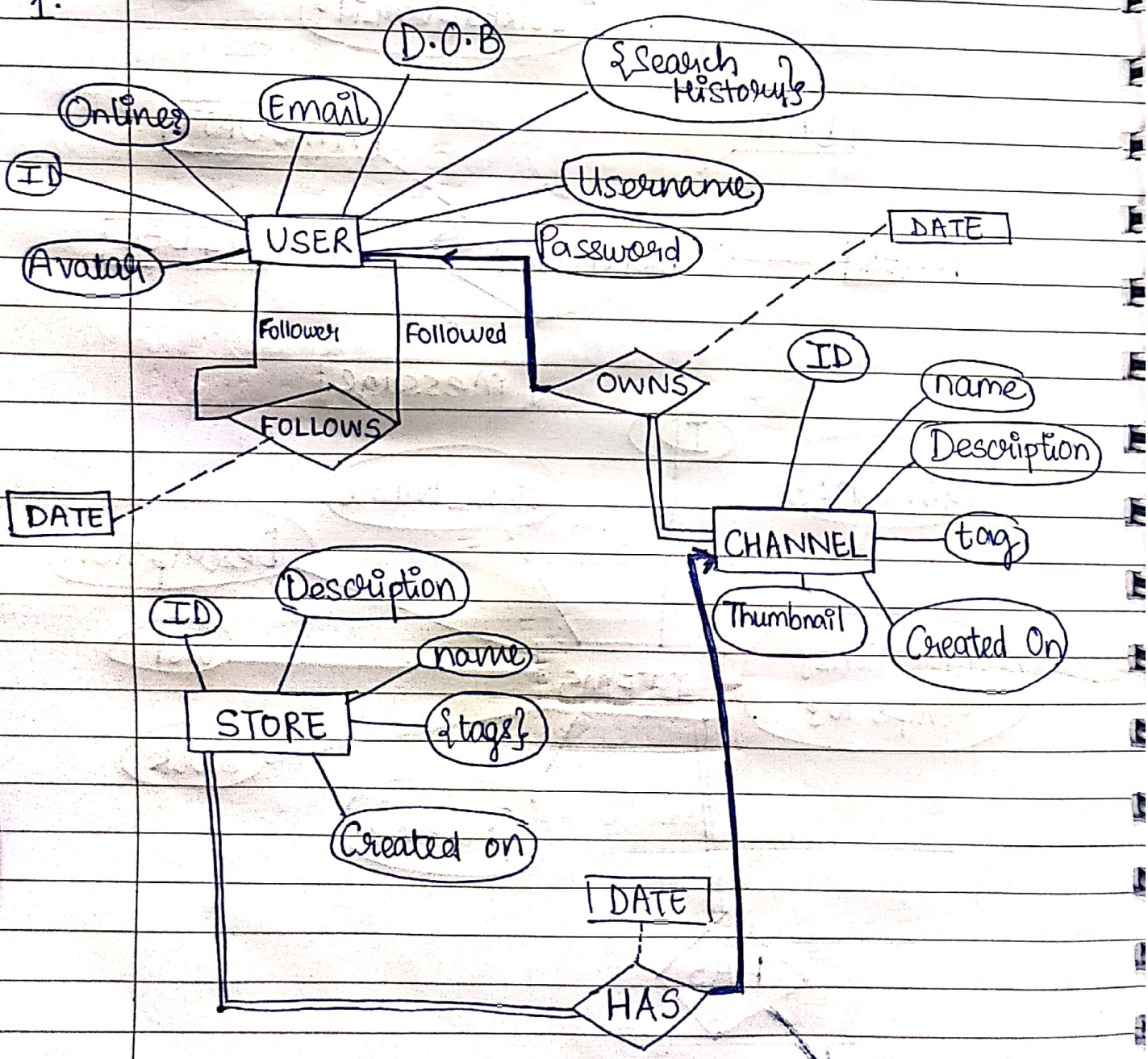
__/__/____

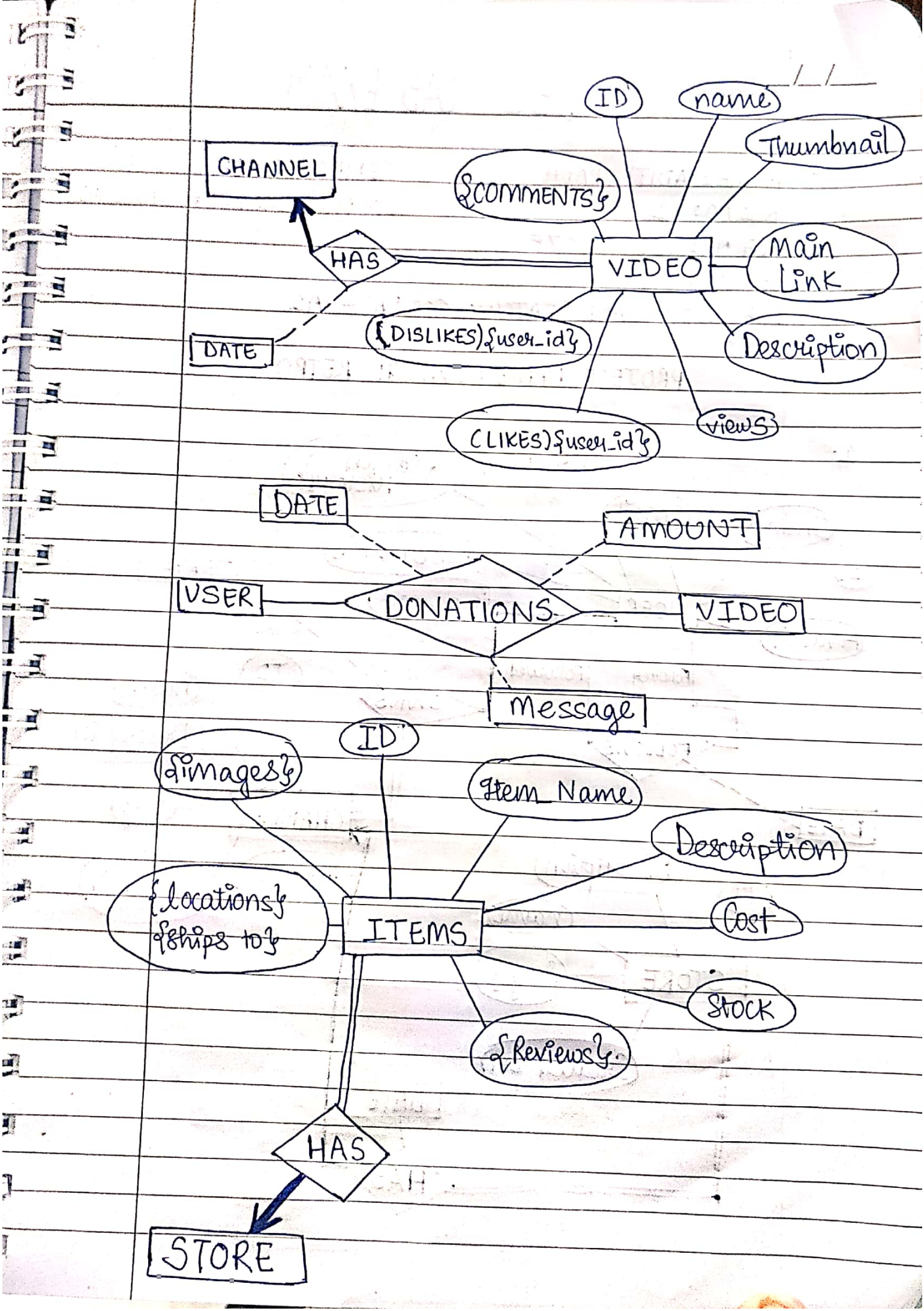
NAME: ADITI PALA
ROLL NO: 12
REG NO: 180953035
SECTION: CCE 'B'
LAB SECTION AND BATCH: CCE 'A' – A1

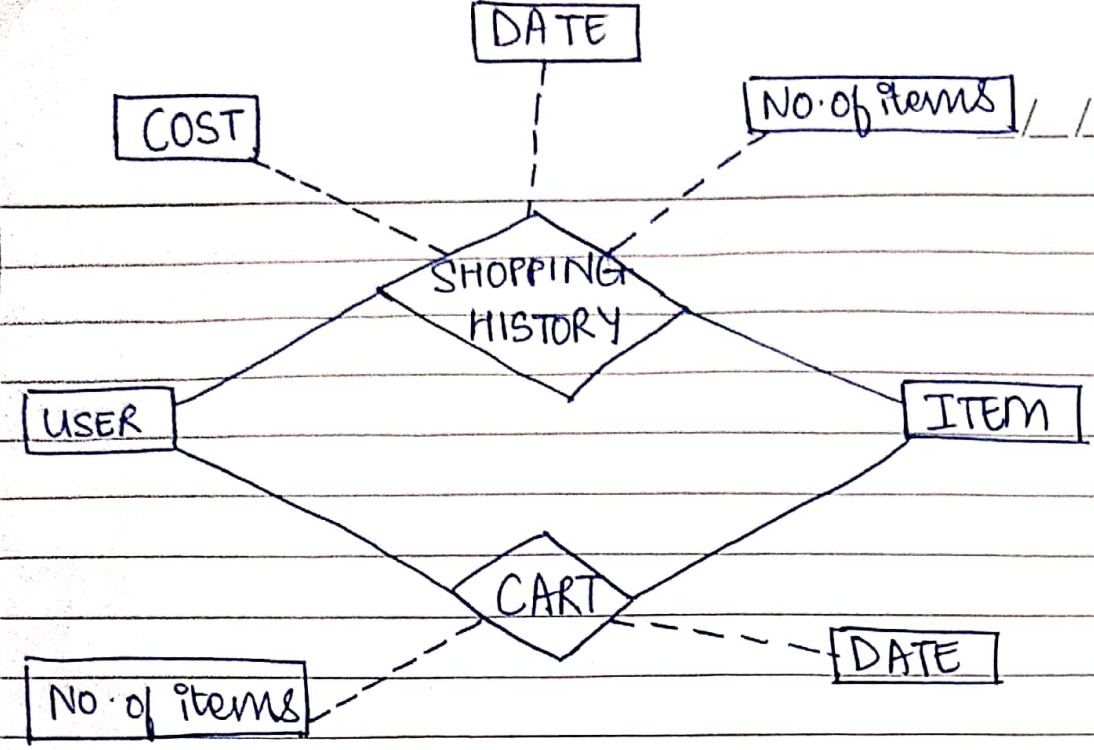
SEMESTER: IV
COURSE: DATABASE SYSTEMS

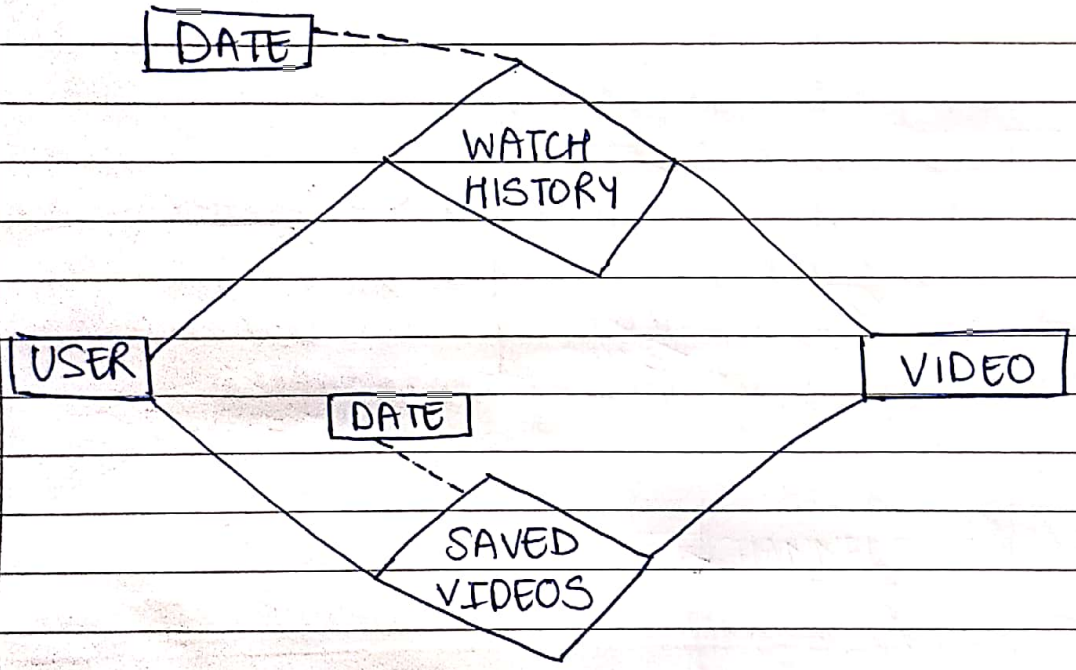
PROJECT NAME: TWITCH RETRO

1.









__/__/____

2. Functional Dependencies

USER: ID → Email Username Password Avatar d.o.b isOnline (BCNF)

CHANNEL: ID → Owner_id name description Tag Created_On Thumbnail (BCNF)

ROLE: ID → Channel_id Name Description Created_On (BCNF)

VIDEO:- ID → Channel_id Name Thumbnail Main_Link DRscription Views Upload_date (BCNF)

ITEMS: ID → Store_id item_name Description Cost Stock (BCNF)

SUBSCRIBER: User_id Channel_id → Subscribed_on (BCNF)

VIDEOLD:- Video_id User_id → Comment (BCNF)

Video_Tags:- Video_id tag → Vid_id tag (BCNF)

ITEM_REVIEW:- Item_id User_id → rating review (BCNF)

WATCH HISTORY:- Userid videoid → date (BCNF)

USER_SHOPPING HISTORY:- Userid item_id date → Amount Number_of_items (Bcnf)

__/__/____

3.a) Contribution

My contribution to the project was FrontEnd i.e UI. A little introduction to the concept: Twitch is a video streaming service operated by Twitch Interactive, a subsidiary of Amazon. The design of the project is themed retro considering the ~~fact~~ idea that this service was launched in the 80's. Implementing the idea in UI, using Unity UI was very efficient & simple. Talking specifically about portrayal of each attribute in the form of, say for example, buttons was easy. Unity supports importation of many 2D patterns etc which aided in making of retro-versioned scheme. I used MySql & Unity to connect frontend & backend so as use their friendly OS variety (Mac, Android, Windows) and simplicity in implementation.

__/__/____

5.b) Code Snippet for UI Implementation of TWITCH RETRO.

DB CONNECTIVITY

CONNECTING FRONT END (UNITY) WITH SQL DATABASE

```
using UnityEngine;
using Mono.Data.Sqlite;
using System.Data;
using UnityEngine.UI;

public class testdb: MonoBehaviour
{
    void AnyEventListener()
    {
      string conn = "URI=file:" + Application.dataPath
                    + "databasename.db";

      IDbConnection dbconn;
      dbconn = (IDbConnection) new SqliteConnection
                                  (conn);

      dbconn.Open();

      IDbCommand dbcmd = dbConn.CreateCommand();

      string sqlQuery = "SELECT id, userName,
                         password" + " FROM User";

      dbcmd.CommandText = sqlQuery;

      IDataReader reader = dbcmd.ExecuteReader();
```

P.T.O →

```
while (reader.Read())
{   int id = reader.GetInt32(0);
    string username = reader.GetString(1);
    string password = reader.GetString(2);

    Text datatext = gameObject.GetComponent<Text>();
    if (password.length > 8)
    {   datatext = id + "  " + username + "" + password;
    }
    else
    {
     datatext.text = "invalid password!";
     datatext.color = Color.red;
    }
}
reader.Close();
dbcmd.Dispose();
dbconn.Close();
}
}
```

CODE SNIPPET

4- CONNECTING FRONT END (UNITY) WITH MYSQL SERVER

```
using System;
using System.Data;
using System.Data.SqlClient;

public class Test
{ public void AnyEVENTlistener()
  { string connectionString = "Server=TestServer;"
        + "Database=test;" + "User ID=sa;"
        + "Password=MightyMighty;";
```

```
IDbConnection dbcon;

using(dbcon = new SqlConnection(connectionString))
{ dbcon.Open();

  using(IDbCommand dbcmd = dbcon.CreateCommand())
  { string sql = "SELECT fname, lname" +
                 "FROM employee";

    dbcmd.CommandText = sql;
    using (IDataReader reader = dbcmd.ExecuteReader())
    { while (reader.Read())
      { string firstName = (string) reader["fname"];
        string lastName = (string) reader["lname"];
        Console.WriteLine("Name:" + firstName +
                          " " + lastName);
      }
    }
  }
}
```

Solution