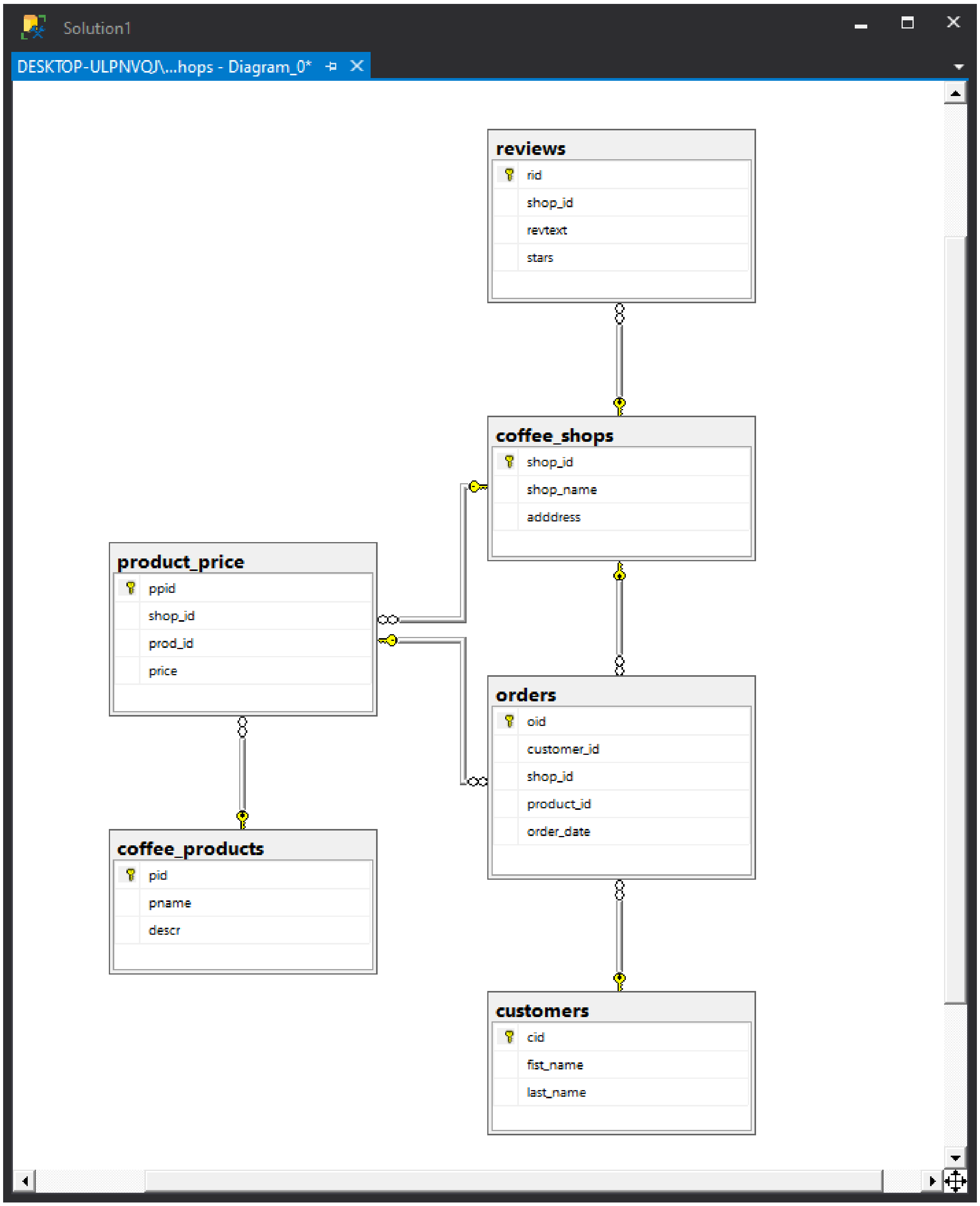
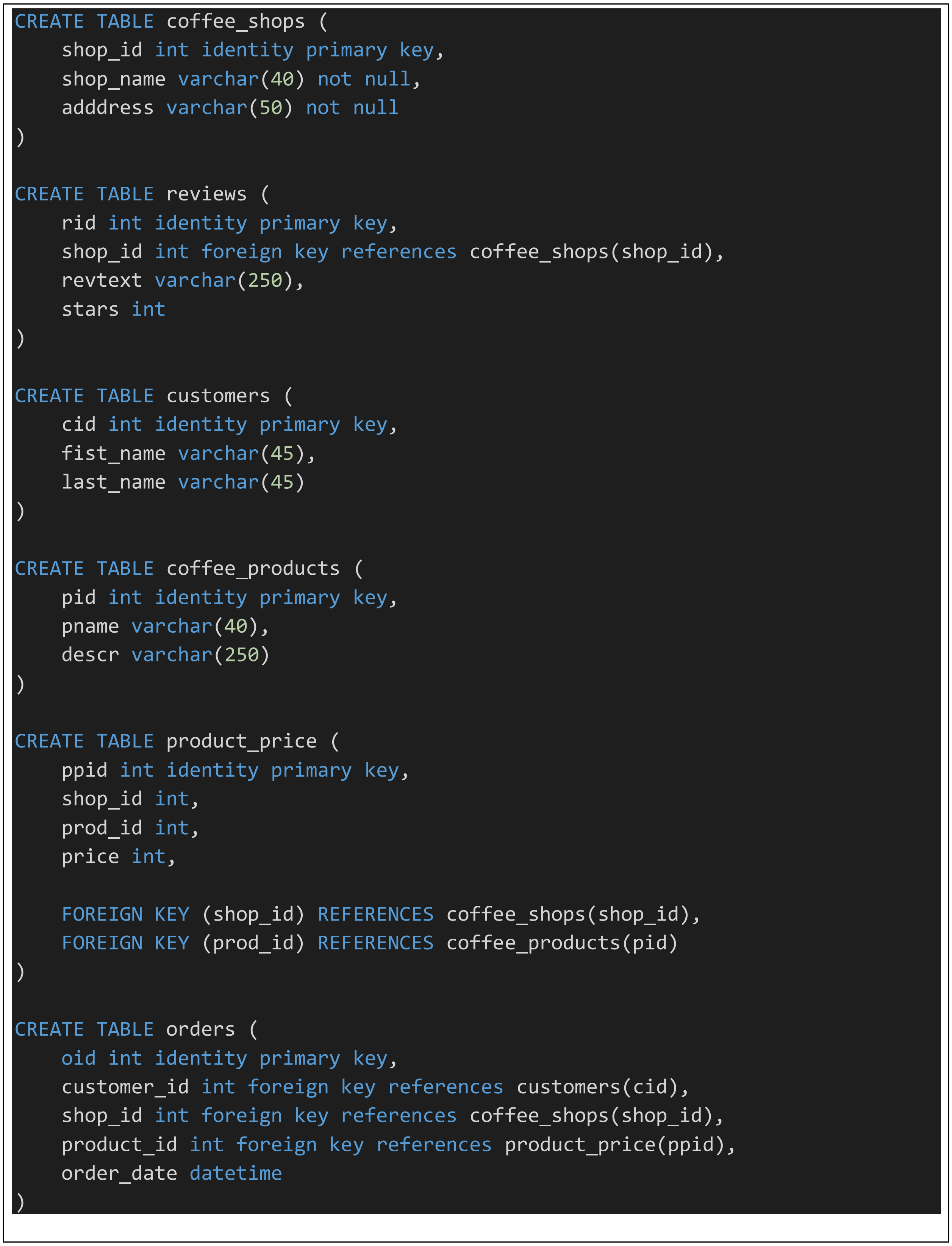
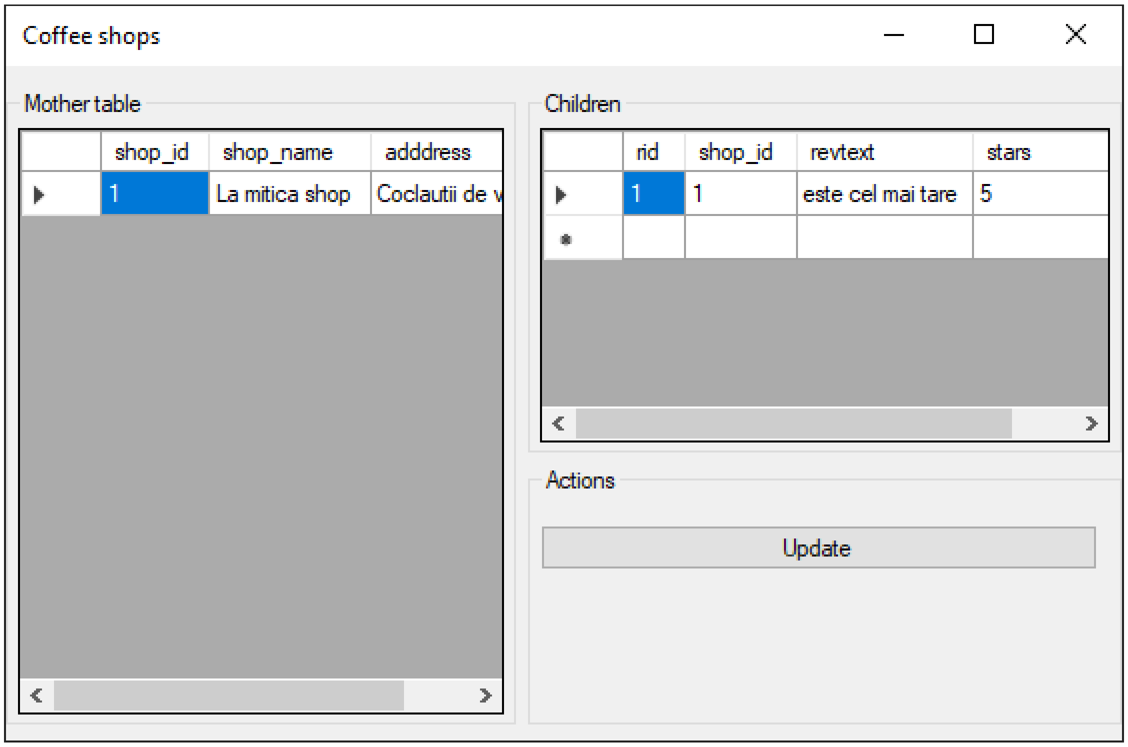

Solution1
DESKTOP-ULPNVQJ\...hops - Diagram_0*
reviews
rid
shop_id
revtext
stars
coffee_shops
shop_id
shop_name
adddress
product_price
ppid
shop_id
prod_id
price
orders
oid
customer_id
shop_id
product_id
order_date
coffee_products
pid
pname
descr
customers
cid
fist_name
last_name



```sql
CREATE TABLE coffee_shops (
    shop_id int identity primary key,
    shop_name varchar(40) not null,
    adddress varchar(50) not null
)

CREATE TABLE reviews (
    rid int identity primary key,
    shop_id int foreign key references coffee_shops(shop_id),
    revtext varchar(250),
    stars int
)

CREATE TABLE customers (
    cid int identity primary key,
    fist_name varchar(45),
    last_name varchar(45)
)

CREATE TABLE coffee_products (
    pid int identity primary key,
    pname varchar(40),
    descr varchar(250)
)

CREATE TABLE product_price (
    ppid int identity primary key,
    shop_id int,
    prod_id int,
    price int,

    FOREIGN KEY (shop_id) REFERENCES coffee_shops(shop_id),
    FOREIGN KEY (prod_id) REFERENCES coffee_products(pid)
)

CREATE TABLE orders (
    oid int identity primary key,
    customer_id int foreign key references customers(cid),
    shop_id int foreign key references coffee_shops(shop_id),
    product_id int foreign key references product_price(ppid),
    order_date datetime
)
```

C#:



	shop_id	shop_name	adddress
▶	1	La mitica shop	Coclautii de v

	rid	shop_id	revtext	stars
▶	1	1	este cel mai tare	5
*				

- Connection code:

```
public string GetConnectionString()
{
    ConnectionStringSettingsCollection connectionString = ConfigurationManager.ConnectionStrings;

    if (connectionString != null)
    {
        return connectionString["connection"].ConnectionString.ToString();
    }

    return null;
}

connection = new SqlConnection(GetConnectionString());
connection.Open();
```

// where app.config has:

```
<connectionStrings>
  <add name="connection" connectionString="Server=DESKTOP-ULPNVQJ\SQLEXPRESS;Database=CoffeeShops;Trusted_Connection=True" />
```

```
        </connectionStrings>
```

I have used app.config to manage the tables and connection queries, so I have a modular application. Whenever I want, I can easily switch to other 1:n relationship with no changes in my code.

- Fetch data into first dataGridView (here I used a class responsible for first data grid, but I'll add only the important parts of the code)

Class that gets the data:

```
try
{
    connection = new SqlConnection(GetConnectionString());
    connection.Open();
    Console.WriteLine("Successfully connected to db!");
    Console.WriteLine(connection.ConnectionTimeout);
    Console.WriteLine(connection.ConnectionString);

    String selectStatement = this.cmd;
    SqlCommand command = new SqlCommand(selectStatement, connection);

    var adapter = new SqlDataAdapter(command);

    var ds = new DataSet();

    adapter.Fill(ds, ConfigurationManager.AppSettings["parentTable"]);
    connection.Close();

    return ds;

}
catch (SqlException ex)
{
    Console.WriteLine("Could not conenct to the db " + ex.Message);
    throw new Exception("Could not conenct to the db " + ex.Message);
}
```

The reason why I used that additional class was that I wanted to make my app more modular, but there were a few problems with my project structure(e.g. maintaining a connection through the whole application run), but finally I decided to use that class for "mother table" and isolate it from From1.cs and the code specific to child table (easy to read 😊)

Form class:

```
     var ReviewsDataSet = new DataSet();
DBConnect db = new DBConnect();  // the class above

string leftHandSelect = GetMotherTable();

db.PrepeareCommand(leftHandSelect);
EmployeesDataSet = db.Run();
this.dgvCoffeeShops.DataSource = ReviewsDataSet;
this.dgvCoffeeShops.DataMember = ConfigurationManager.AppSettings["parentTable"
];
```

The code responsible for the second dataGridView

```
private void employeesDataGrid_CellClick(object sender, DataGridViewCellEventAr
gs e)
{
    try
    {
        connection = new SqlConnection(GetConnectionString());
        connection.Open();
        Console.WriteLine("Successfully connected to db!");
        Console.WriteLine(connection.ConnectionTimeout);
        Console.WriteLine(connection.ConnectionString);

        int selectedrowindex = -1;

        selectedrowindex = dgvCoffeeShops.SelectedCells[0].RowIndex;
        DataGridViewRow selectedRow = dgvCoffeeShops.Rows[selectedrowindex];

        Console.WriteLine(selectedrowindex);

        String selectStatement = GetChildTable();
        String parentKey = ConfigurationManager.AppSettings["parentKey"];

        SqlCommand command = new SqlCommand(selectStatement, connection);

        command.Parameters.AddWithValue("@key", Int32.Parse(selectedRow.Cells[p
arentKey].Value.ToString()));

        adapter = new SqlDataAdapter(command);

        selDS = new DataSet();

        adapter.Fill(selDS, ConfigurationManager.AppSettings["childTable"]);
        dgvReviews.DataSource = selDS;
        dgvReviews.DataMember = ConfigurationManager.AppSettings["childTable"];

    }
    catch (SqlException ex)
    {
        Console.WriteLine("Could not conenct to the db " + ex.Message);
        throw new Exception("Could not conenct to the db " + ex.Message);
    }
}
```

The function responsible for acquiring the reviews for a coffee shops is above, binded to first dataGridView CellClick event handler. To know which shop I refer to, here comes the app.config file where I specify which are the primary keys and foreign keys, so my program knows which data to handle for selection. Also, I added some modularity to my query as well and I that why I have query command generators. Additional app.config lines:

```
<connectionStrings>
    <add name="connection" connectionString="Server=DESKTOP-
ULPNVQJ\SQLEXPRESS;Database=CoffeeShops;Trusted_Connection=True" />
</connectionStrings>

<appSettings>
    <add key="parentTable" value="coffee_shops"/>
    <add key="parentKey" value="shop_id"/>
    <add key="parentSelection" value="*"/>

    <add key="childTable" value="reviews"/>
    <add key="childForeignKey" value="rid"/>
    <add key="childSelection" value="*"/>
</appSettings>
```

- Commit changes to the database (button event handler):

```
private void button1_Click_1(object sender, EventArgs e)
{

    if(adapter != null)
    {
        adapter.Update(selDS, ConfigurationManager.AppSettings["childTable"
]);
    }
// checks if the adapter is not null, cause it’s initialized when
//a cell from main data grid view is clicked (so it knows which shop to ref
er to)
}
```

- Phantom read:

```
-- Phantom Reads pt2
use [CoffeeShops]
go

begin transaction
waitfor delay '00:00:05'
insert into coffee_shops(shop_name, adddress) values ('La Mitica Shop & Cus
toms', 'Coclautii de vale')
commit transaction

-- Phantom Reads pt2
use [CoffeeShops]
go

-- Set transaction isolation level repeatable read
-- Solve by changing isolation level to serializable

begin transaction
select * from coffee_shops
waitfor delay '00:00:05'
select * from coffee_shops
commit transaction
```

Initially, both transaction start at the same time, but the second one waits for an amount of time, an interval in which the first transactions inserts a new record. Thus, from here comes the phantom read that can be observed with two select queries in the second transaction (the second select has one additional record, added by the first transaction, that being the phantom read). This situation can be solved by simply changing the isolation level to serializable, as mentioned in the code comments above.