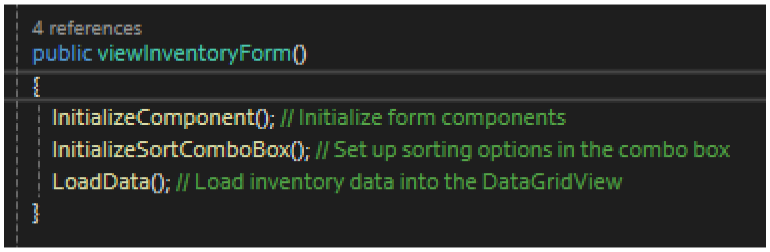
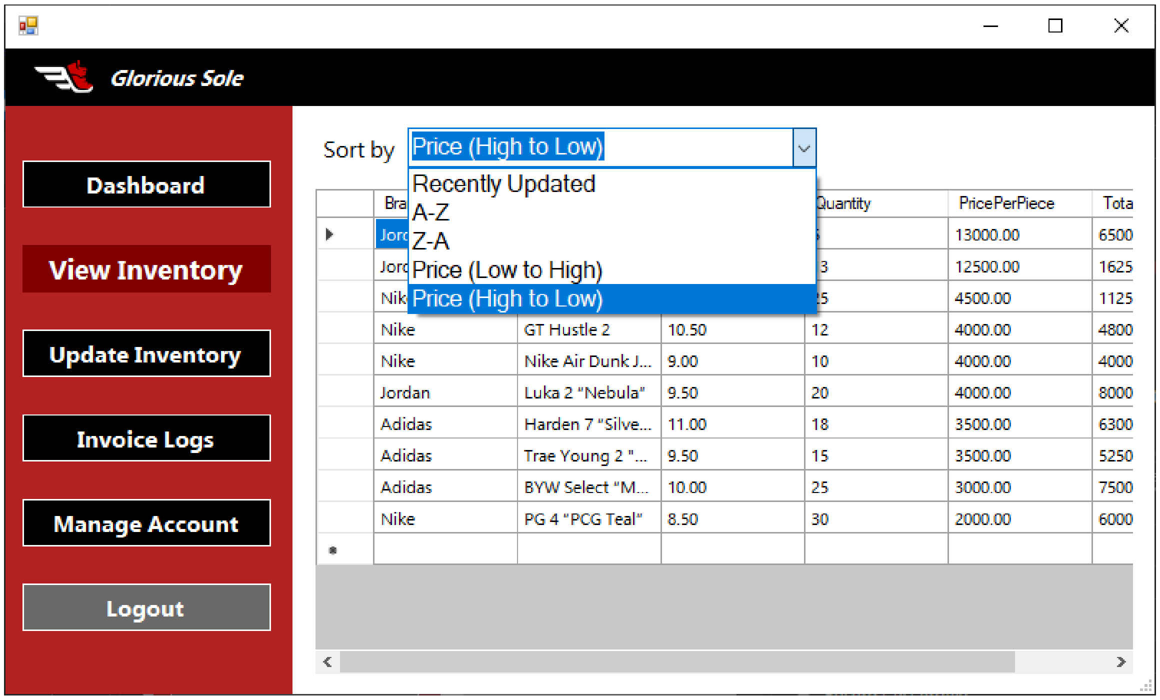
Topic:	View Inventory Form
Assigned Members:	Presno & Tojino

Opening the View Inventory form this would be initialized:



```
4 references
public viewInventoryForm()
{
    InitializeComponent(); // Initialize form components
    InitializeSortComboBox(); // Set up sorting options in the combo box
    LoadData(); // Load inventory data into the DataGridView
}
```

Sort Options:

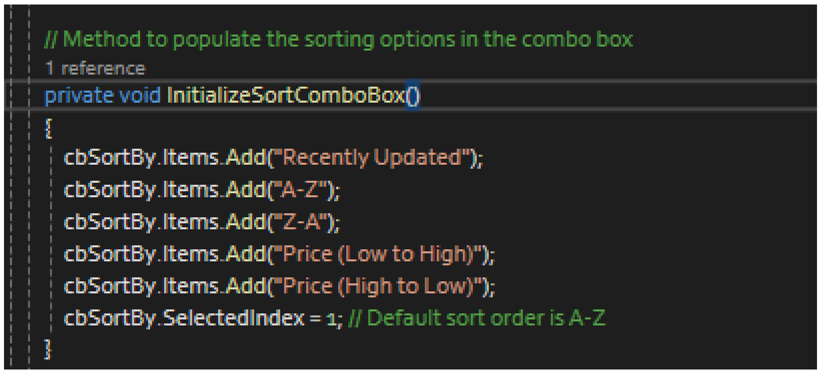


Sort by Price (High to Low)

Recently Updated
A-Z
Z-A
Price (Low to High)
Price (High to Low)

Bra			Quantity	PricePerPiece	Tota
Jorc				13000.00	6500
Jorc			3	12500.00	1625
Nik			25	4500.00	1125
Nike	GT Hustle 2	10.50	12	4000.00	4800
Nike	Nike Air Dunk J...	9.00	10	4000.00	4000
Jordan	Luka 2 "Nebula"	9.50	20	4000.00	8000
Adidas	Harden 7 "Silve...	11.00	18	3500.00	6300
Adidas	Trae Young 2 "...	9.50	15	3500.00	5250
Adidas	BYW Select "M...	10.00	25	3000.00	7500
Nike	PG 4 "PCG Teal"	8.50	30	2000.00	6000

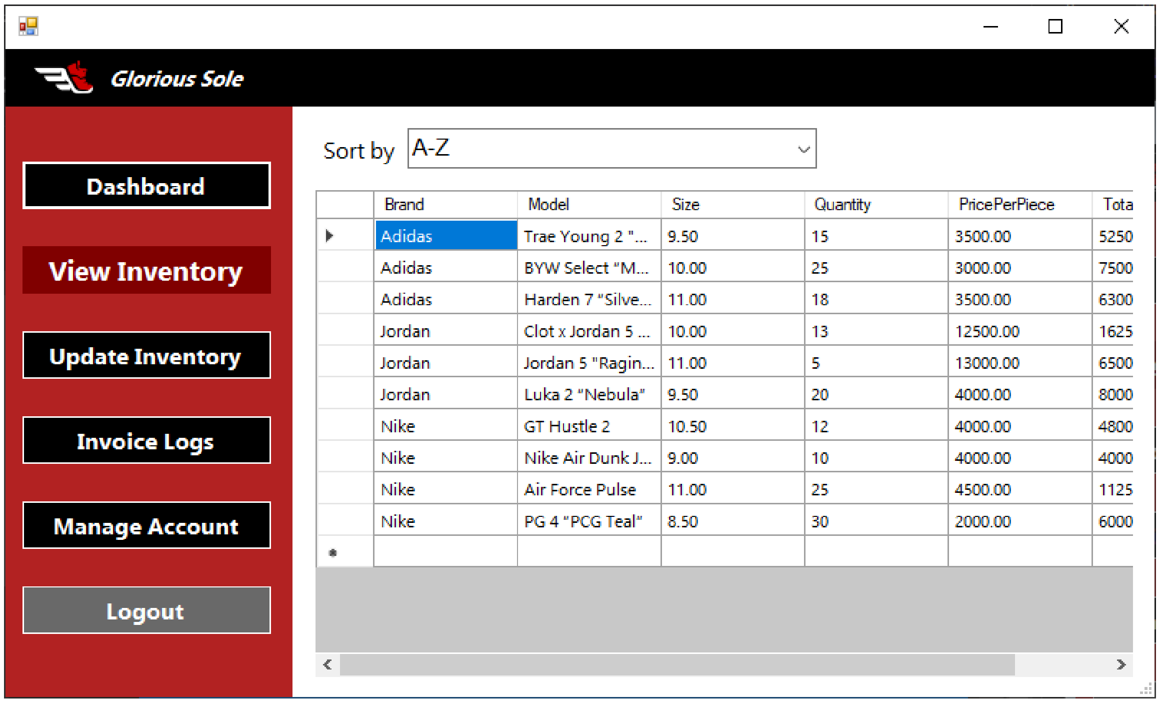
This sets the sorting options on the Combo Box:



```
// Method to populate the sorting options in the combo box
1 reference
private void InitializeSortComboBox()
{
    cbSortBy.Items.Add("Recently Updated");
    cbSortBy.Items.Add("A-Z");
    cbSortBy.Items.Add("Z-A");
    cbSortBy.Items.Add("Price (Low to High)");
    cbSortBy.Items.Add("Price (High to Low)");
    cbSortBy.SelectedIndex = 1; // Default sort order is A-Z
}
```

When choosing an option for the Combo Box the LoadData() would load various data from the database and apply its sorting:

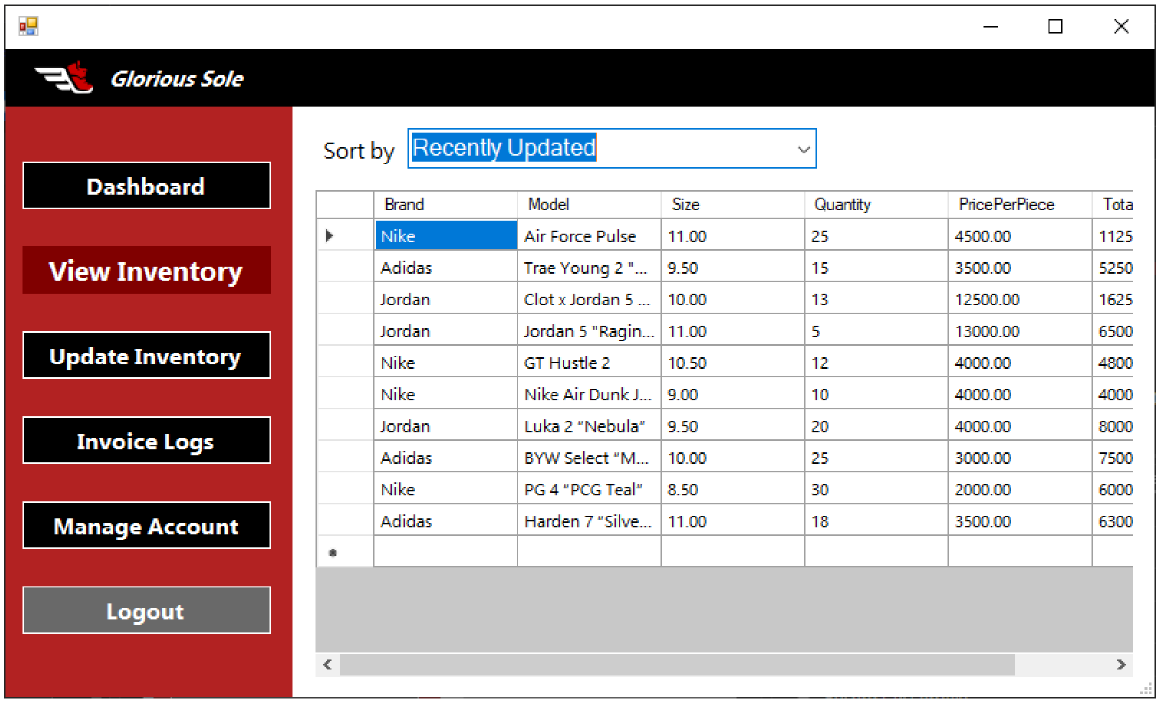
Default (A-Z) sorted view:



Sort by A-Z

Brand	Model	Size	Quantity	PricePerPiece	Tota
Adidas	Trae Young 2 "...	9.50	15	3500.00	5250
Adidas	BYW Select "M...	10.00	25	3000.00	7500
Adidas	Harden 7 "Silve...	11.00	18	3500.00	6300
Jordan	Clot x Jordan 5 ...	10.00	13	12500.00	1625
Jordan	Jordan 5 "Ragin...	11.00	5	13000.00	6500
Jordan	Luka 2 "Nebula"	9.50	20	4000.00	8000
Nike	GT Hustle 2	10.50	12	4000.00	4800
Nike	Nike Air Dunk J...	9.00	10	4000.00	4000
Nike	Air Force Pulse	11.00	25	4500.00	1125
Nike	PG 4 "PCG Teal"	8.50	30	2000.00	6000

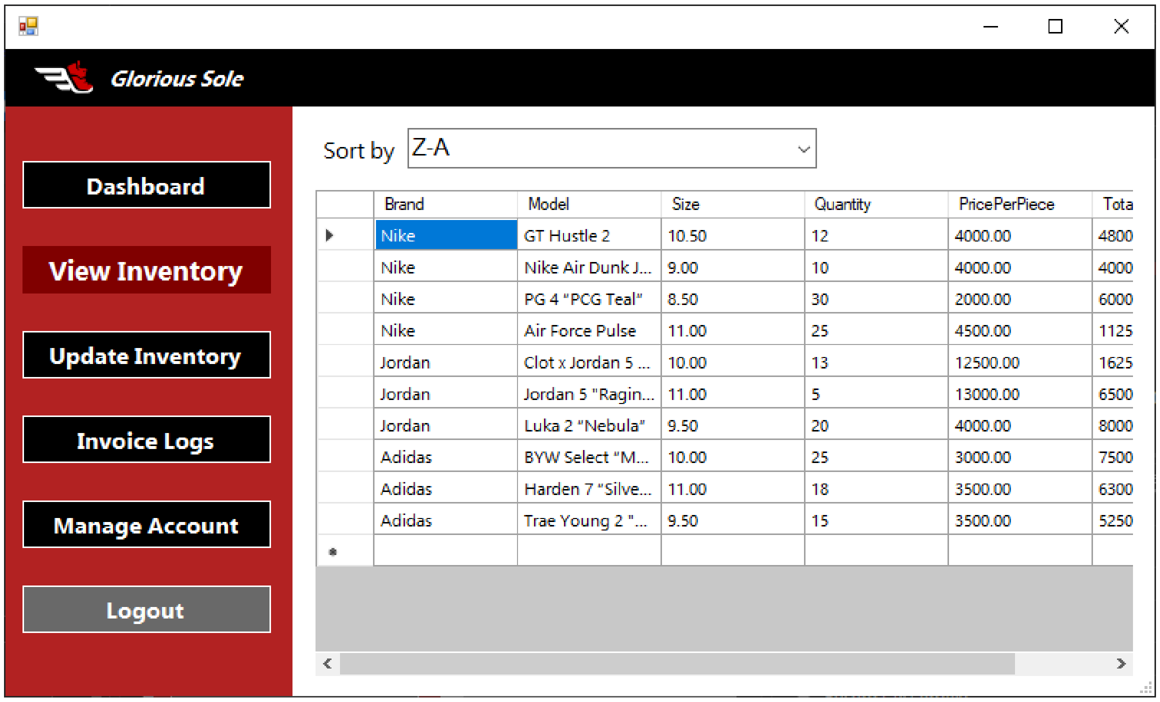
Recently Updated sort:



Sort by Recently Updated

Brand	Model	Size	Quantity	PricePerPiece	Tota
Nike	Air Force Pulse	11.00	25	4500.00	1125
Adidas	Trae Young 2 "...	9.50	15	3500.00	5250
Jordan	Clot x Jordan 5 ...	10.00	13	12500.00	1625
Jordan	Jordan 5 "Ragin...	11.00	5	13000.00	6500
Nike	GT Hustle 2	10.50	12	4000.00	4800
Nike	Nike Air Dunk J...	9.00	10	4000.00	4000
Jordan	Luka 2 "Nebula"	9.50	20	4000.00	8000
Adidas	BYW Select "M...	10.00	25	3000.00	7500
Nike	PG 4 "PCG Teal"	8.50	30	2000.00	6000
Adidas	Harden 7 "Silve...	11.00	18	3500.00	6300

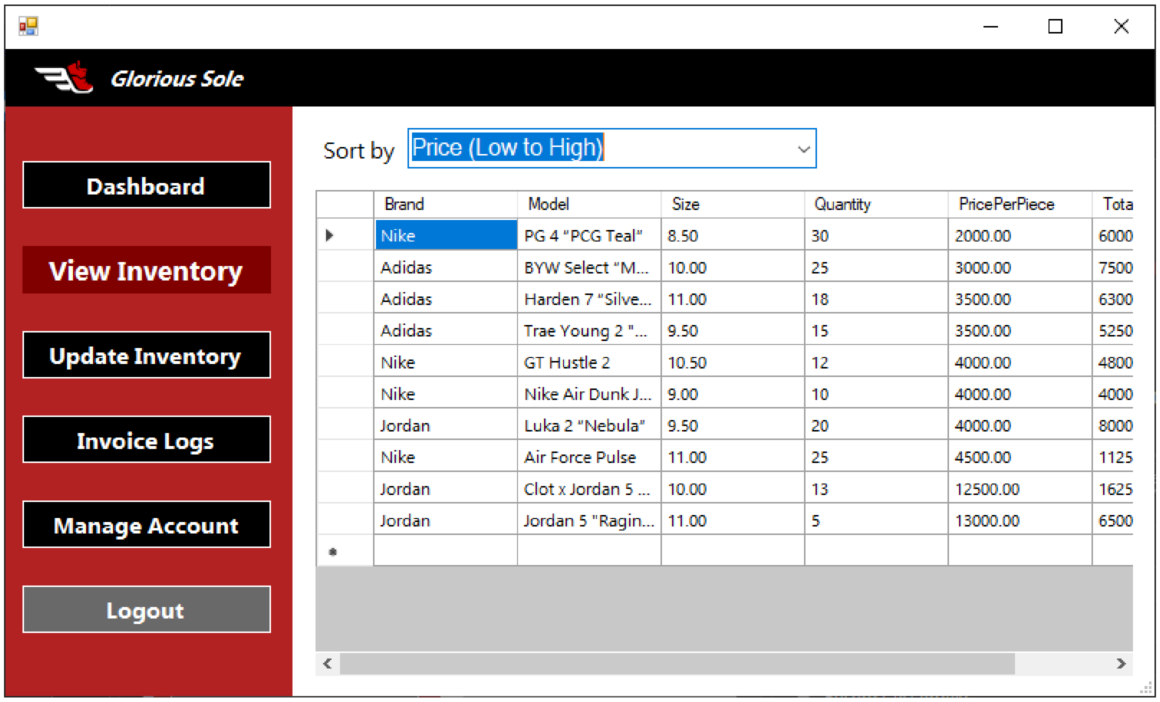
Z-A sort Option:



Sort by Z-A

Brand	Model	Size	Quantity	PricePerPiece	Tota
Nike	GT Hustle 2	10.50	12	4000.00	4800
Nike	Nike Air Dunk J...	9.00	10	4000.00	4000
Nike	PG 4 "PCG Teal"	8.50	30	2000.00	6000
Nike	Air Force Pulse	11.00	25	4500.00	1125
Jordan	Clot x Jordan 5 ...	10.00	13	12500.00	1625
Jordan	Jordan 5 "Ragin...	11.00	5	13000.00	6500
Jordan	Luka 2 "Nebula"	9.50	20	4000.00	8000
Adidas	BYW Select "M...	10.00	25	3000.00	7500
Adidas	Harden 7 "Silve...	11.00	18	3500.00	6300
Adidas	Trae Young 2 "...	9.50	15	3500.00	5250

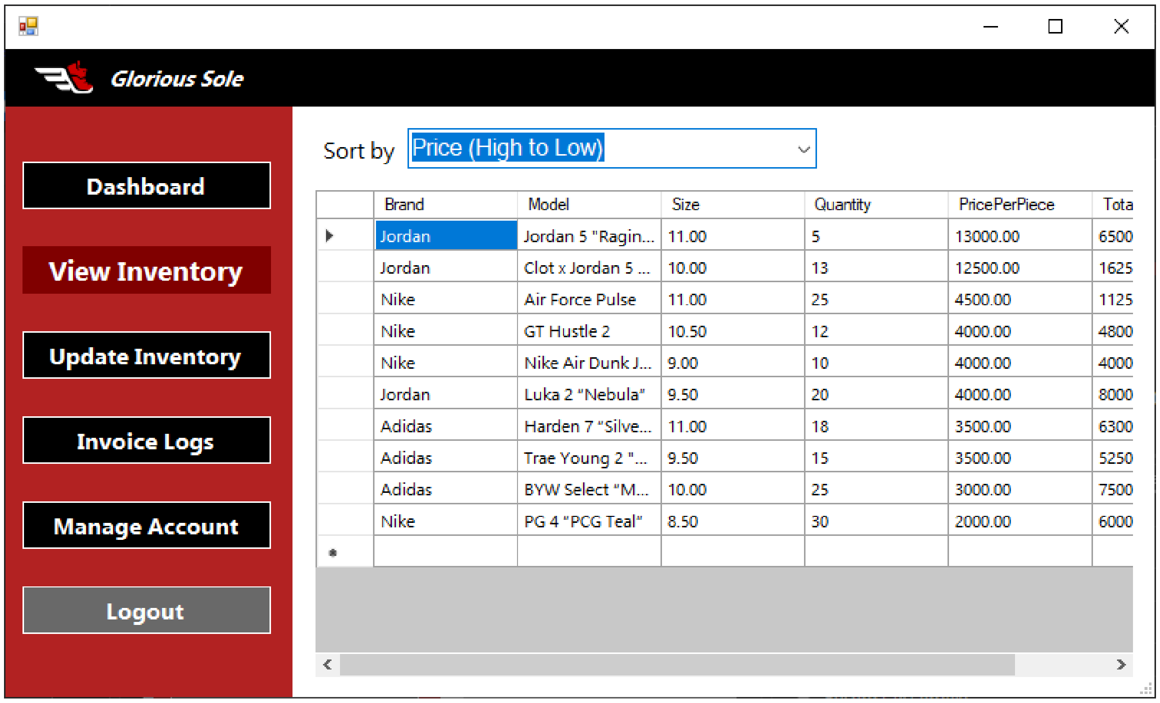
Price (Low to High) sort Option:



Sort by Price (Low to High)

Brand	Model	Size	Quantity	PricePerPiece	Tota
Nike	PG 4 "PCG Teal"	8.50	30	2000.00	6000
Adidas	BYW Select "M...	10.00	25	3000.00	7500
Adidas	Harden 7 "Silve...	11.00	18	3500.00	6300
Adidas	Trae Young 2 "...	9.50	15	3500.00	5250
Nike	GT Hustle 2	10.50	12	4000.00	4800
Nike	Nike Air Dunk J...	9.00	10	4000.00	4000
Jordan	Luka 2 "Nebula"	9.50	20	4000.00	8000
Nike	Air Force Pulse	11.00	25	4500.00	1125
Jordan	Clot x Jordan 5 ...	10.00	13	12500.00	1625
Jordan	Jordan 5 "Ragin...	11.00	5	13000.00	6500

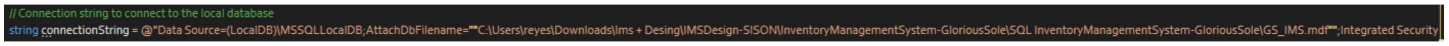
Price (High to Low) sort Option:



Sort by: Price (High to Low)

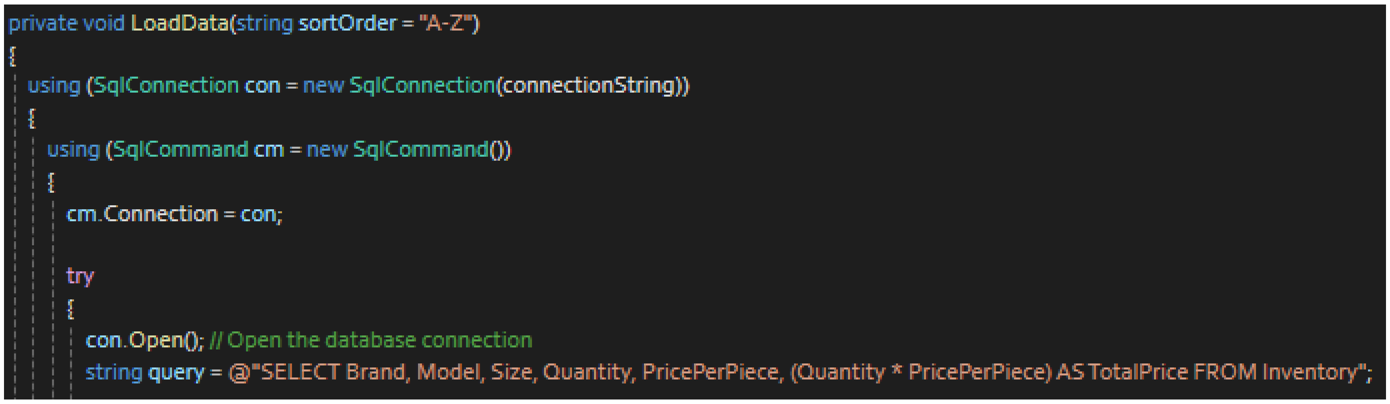
Brand	Model	Size	Quantity	PricePerPiece	Tota
Jordan	Jordan 5 "Ragin...	11.00	5	13000.00	6500
Jordan	Clot x Jordan 5 ...	10.00	13	12500.00	1625
Nike	Air Force Pulse	11.00	25	4500.00	1125
Nike	GT Hustle 2	10.50	12	4000.00	4800
Nike	Nike Air Dunk J...	9.00	10	4000.00	4000
Jordan	Luka 2 "Nebula"	9.50	20	4000.00	8000
Adidas	Harden 7 "Silve...	11.00	18	3500.00	6300
Adidas	Trae Young 2 "...	9.50	15	3500.00	5250
Adidas	BYW Select "M...	10.00	25	3000.00	7500
Nike	PG 4 "PCG Teal"	8.50	30	2000.00	6000

We access the database by using SQL Connections:



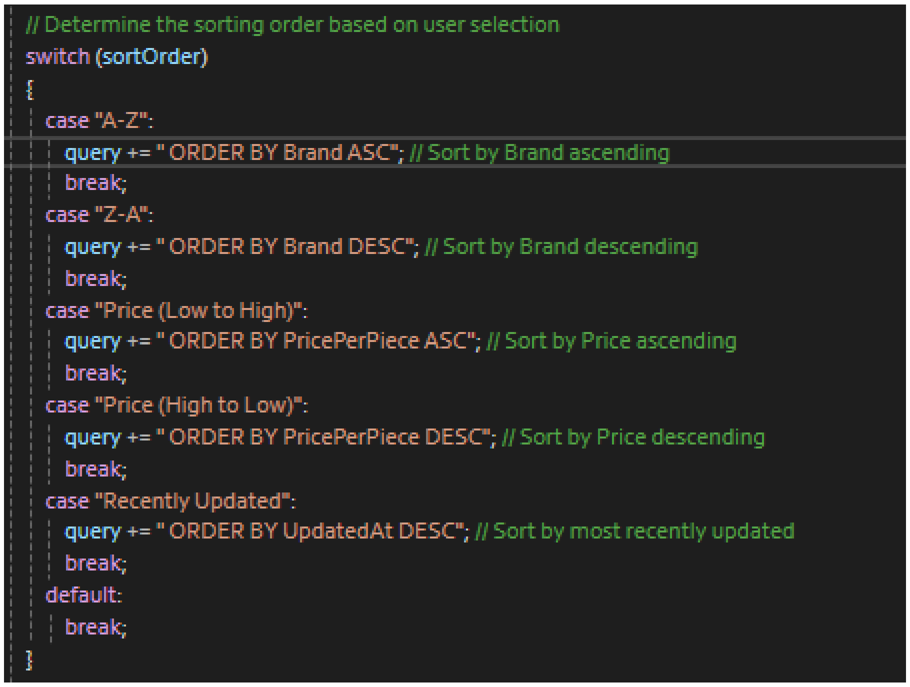
```
// Connection string to connect to the local database
string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=""C:\Users\reyes\Downloads\Ims + Desing\IMSDesign-SISON\InventoryManagementSystem-GloriousSole\SQL InventoryManagementSystem-GloriousSole\GS_IMS.mdf"";Integrated Security
```

With the use of SQL Connection, we can access the overview of the Inventory Table from the Database itself thru this code:



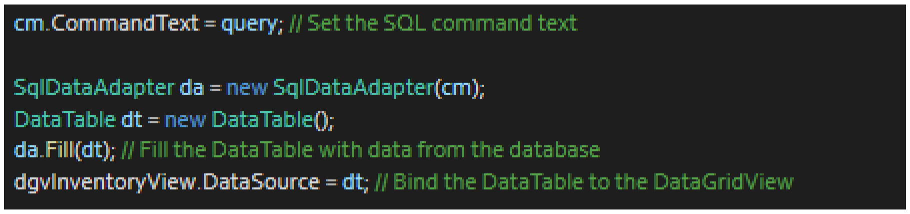
```
private void LoadData(string sortOrder = "A-Z")
{
    using (SqlConnection con = new SqlConnection(connectionString))
    {
        using (SqlCommand cm = new SqlCommand())
        {
            cm.Connection = con;

            try
            {
                con.Open(); // Open the database connection
                string query = @"SELECT Brand, Model, Size, Quantity, PricePerPiece, (Quantity * PricePerPiece) AS TotalPrice FROM Inventory";
```

For them to show we use Switch Case to determine the option chosen by the user:



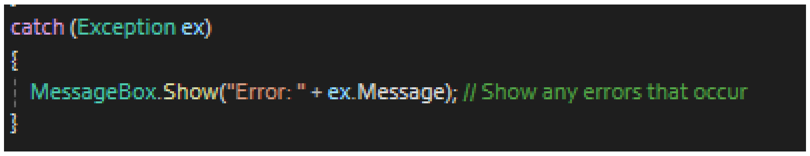
```
// Determine the sorting order based on user selection
switch (sortOrder)
{
    case "A-Z":
        query += " ORDER BY Brand ASC"; // Sort by Brand ascending
        break;
    case "Z-A":
        query += " ORDER BY Brand DESC"; // Sort by Brand descending
        break;
    case "Price (Low to High)":
        query += " ORDER BY PricePerPiece ASC"; // Sort by Price ascending
        break;
    case "Price (High to Low)":
        query += " ORDER BY PricePerPiece DESC"; // Sort by Price descending
        break;
    case "Recently Updated":
        query += " ORDER BY UpdatedAt DESC"; // Sort by most recently updated
        break;
    default:
        break;
}
```

Upon which sorting option does the user picks it would set the SQL command for the query then automatically updates the DataGrid View:



```
cm.CommandText = query; // Set the SQL command text

SqlDataAdapter da = new SqlDataAdapter(cm);
DataTable dt = new DataTable();
da.Fill(dt); // Fill the DataTable with data from the database
dgvInventoryView.DataSource = dt; // Bind the DataTable to the DataGridView
```

And lastly, a catch block would catch any errors that has occurred



```
catch (Exception ex)
{
    MessageBox.Show("Error: " + ex.Message); // Show any errors that occur
}
```