UNIVERSITI KUALA LUMPUR KAMPUS KOTA
MALAYSIAN INSTITUTE OF INFORMATION TECHNOLOGY
ASSESSMENT BRIEF

COURSE DETAILS	
CAMPUS	MIIT
COURSE NAME	ASP.NET WEB PROGRAMMING
COURSE CODE	ITD21203
COURSE LEADER	MDM HANA MUNIRA MUHD MUKHTAR (LO1)
PART-TIME LECTURER	SIR MOHD ZUHAIR MUHAMMAD KHAZANI (LO2) SIR EISOMULLAH LUKMAN (LO3) SIR EZZAT AKARUDIM SULAIMAN (LO4)
YEAR/SEMESTER	2 / 4

ASSESSMENT DETAILS	
TITLE/NAME	PROJECT
WEIGHT	20%
REQUIREMENT	1. ASP.NET a. Master Page b. User Controls c. Validation Controls d. Class & Object (Calculation)
PRESENTATION	WEEK 13

GROUP DETAILS	
NAME	1) MUHAMMAD IZZAT [redacted]
GROUP	Group 8 L03 – B01

Introduction

For this project, I have chosen the QUESTION 2: which is CAR FOR SALE. This question requires an UI and a database **carforsale** to be created for CAR MANAGER to ***insert*** several car informations into the table. The information should then be ***listed*** on some webpage and able to be ***viewed***/***filtered*** by the range of cc or price. Apart from that, another web form is required for the CAR SALESMAN to ***enter*** their sales record such as PURCHASER's information and car details. Furthermore, create a web page that will ***displays*** the TOTAL SALES and will lists all cars that have been sold by the salesman.

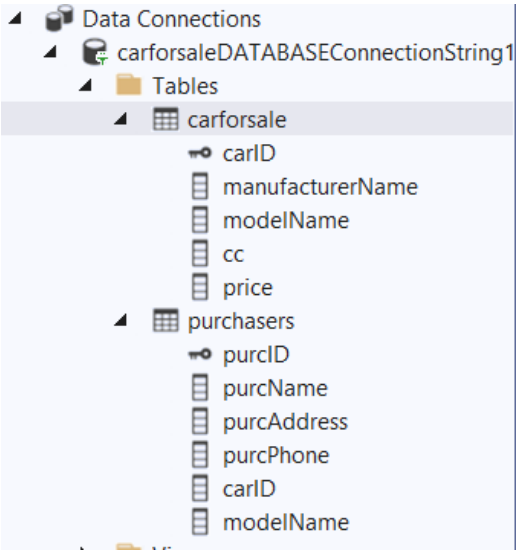
Table carforsale below shows the several information that should be inserted by the CAR MANAGER.

carID	manufacturerName	modelName	cc	price
1	Perodua	Ativa	1.0	RM 71, 200
2	Perodua	Aruz	1.5	RM 73, 266
3	Nissan	Almera	1.5	RM 92, 310
4	Nissan	X-Trail	2.5	RM 159, 888

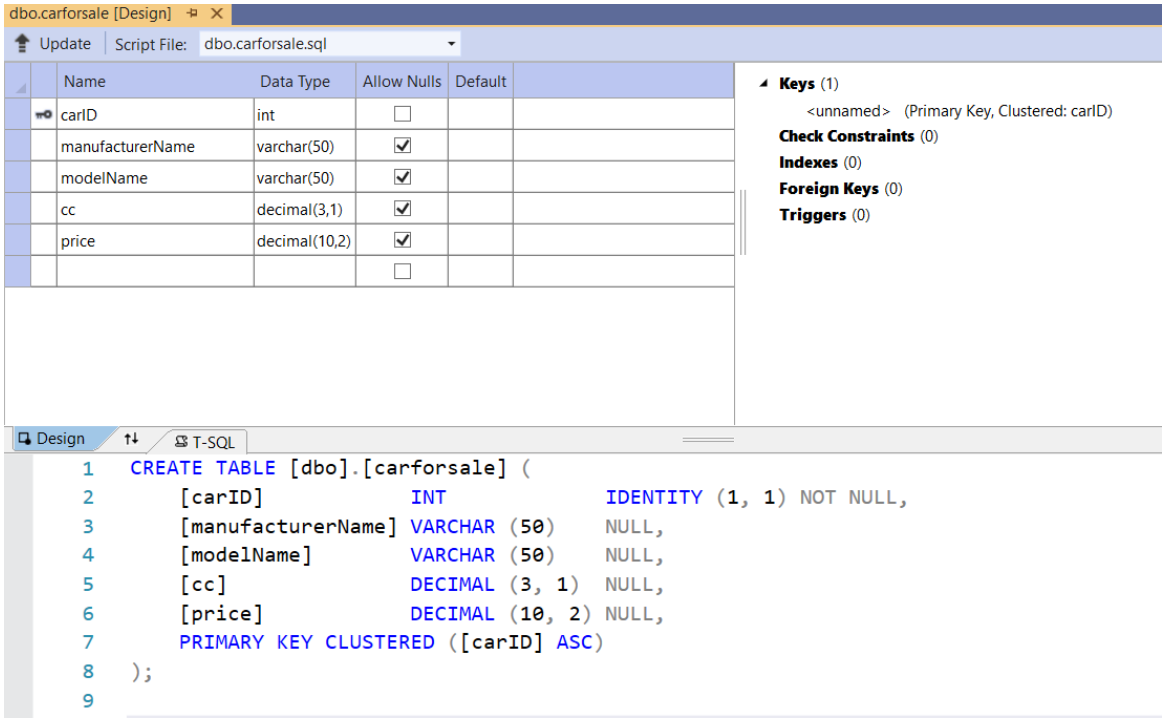
Table carforsale

Data Structure

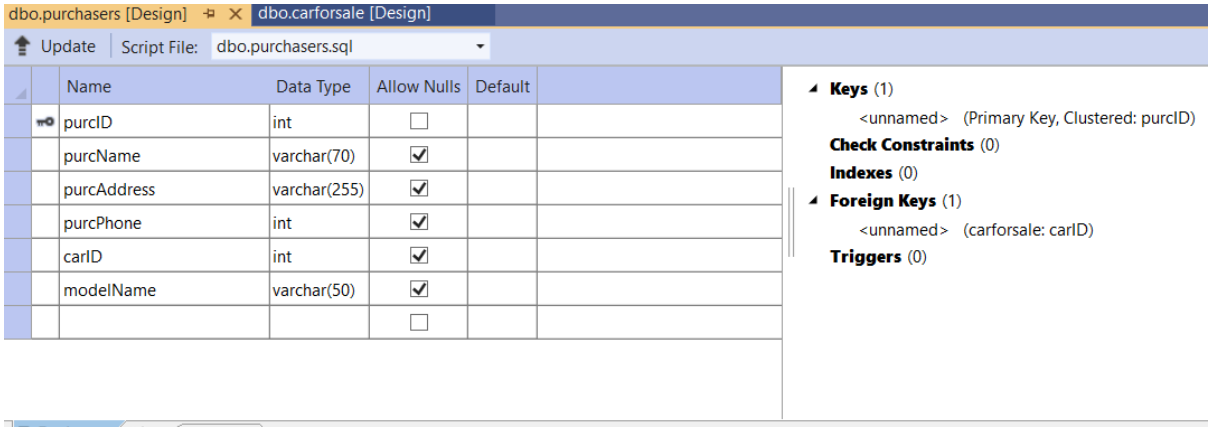
In my project, I've created two tables into my database. One is **carforsale** for CAR MANAGER to insert the car details, and the second is **purchasers** for CAR SALESMAN to record the purchaser's information and their car details.



For table **carforsale**, there are *carID* as PRIMARY KEY with IDENTITY(1,1), *manufacturerName*, *modelName*, *cc*, *price*. The data structure as in below:



As for table **purchasers**, there are *purcID* as a PRIMARY KEY with IDENTITY(3001, 1), *purcName*, *purcAddress*, *purcPhone*, *carID* as a FOREIGN KEY REFERENCES table **carforsale**, and *modelName*. This table is ideally created to easily link the price of bought car along with the purchaser id for calculating the total price later. Below is the data sructure for the table:

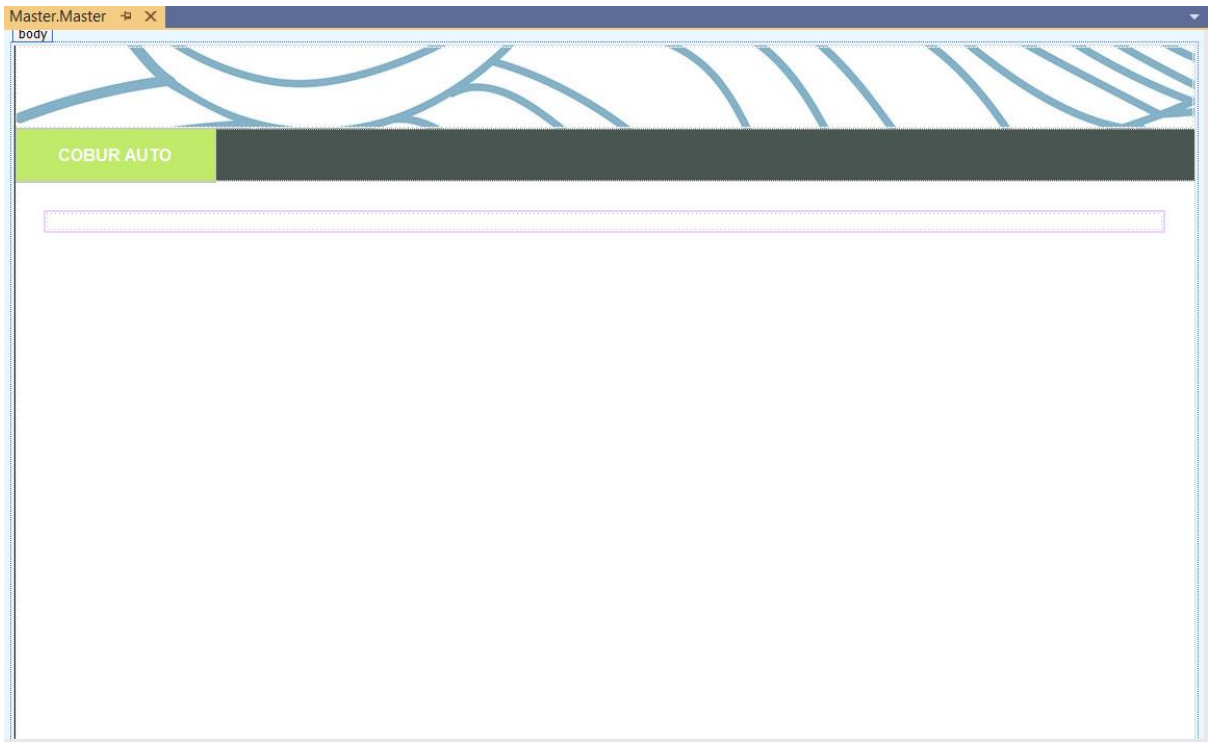


Design | T-SQL

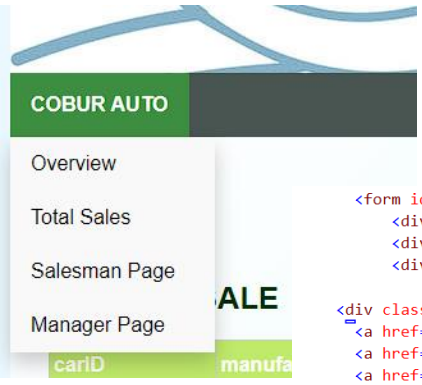
```sql
CREATE TABLE [dbo].[purchasers] (
    [purcID]      INT           IDENTITY (3001, 1) NOT NULL,
    [purcName]    VARCHAR (70)  NULL,
    [purcAddress] VARCHAR (255) NULL,
    [purcPhone]   INT           NULL,
    [carID]       INT           NULL,
    [modelName]   VARCHAR (50)  NULL,
    PRIMARY KEY CLUSTERED ([purcID] ASC),
    FOREIGN KEY ([carID]) REFERENCES [dbo].[carforsale] ([carID])
);
```

Master Page

In Master page, I simply put a picture in the *header*, coloured the *content* background with teal gradient to make the page looks pleasant, then there is a *navigation menu* in the menu section with my made up car dealership's name COBUR AUTO.



Four options/navigations will appear once user hovered onto the navigation COBUR AUTO to redirect user to each four different pages. The four options are Overview, Total Sales, Salesman Page, and Manager Page.




```
<form id="form1" runat="server">
    <div id="page">
    <div id="header" class="auto-style3"></div>
    <div id="menu">
        <div class="dropdown"><button class="dropbtn"><strong>COBUR AUTO</strong></button>
<div class="dropdown-content">
  <a href="overview.aspx">Overview</a>
  <a href="TotalSales.aspx">Total Sales</a>
  <a href="CarSalesman.aspx">Salesman Page</a>
  <a href="CarManager.aspx">Manager Page</a>
</div>
</div> </div>
    <div id="content">
        <asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">
```

Overview Page

This page will show the tables of data of both carforsale and purchasers that has been inserted by Manager and/or Salesman.

carID	manufacturerName	modelName	cc	price
1	Perodua	Ativa	1.0	71200.00
2	Perodua	Aruz	1.5	73266.00
3	Nissan	Almera	1.5	92310.00
4	Nissan	X-Trail	2.5	159888.00

purcID	purcName	purcAddress	purcPhone	carID	modelName
3001	Ammar Danish	No 9 Rah Residensi	1231241567	1	Ativa
3002	Saleem Chong	Jalan Inang 49 Taman Meru	1922353641	4	X-Trail

There is nothing much to do in this page as the page is only to view data to Manager or Salesman.

SQL to show table carforsale

SELECT statement:

```
SELECT [carID], [manufacturerName], [modelName], [cc], [price] FROM [carforsale]
```

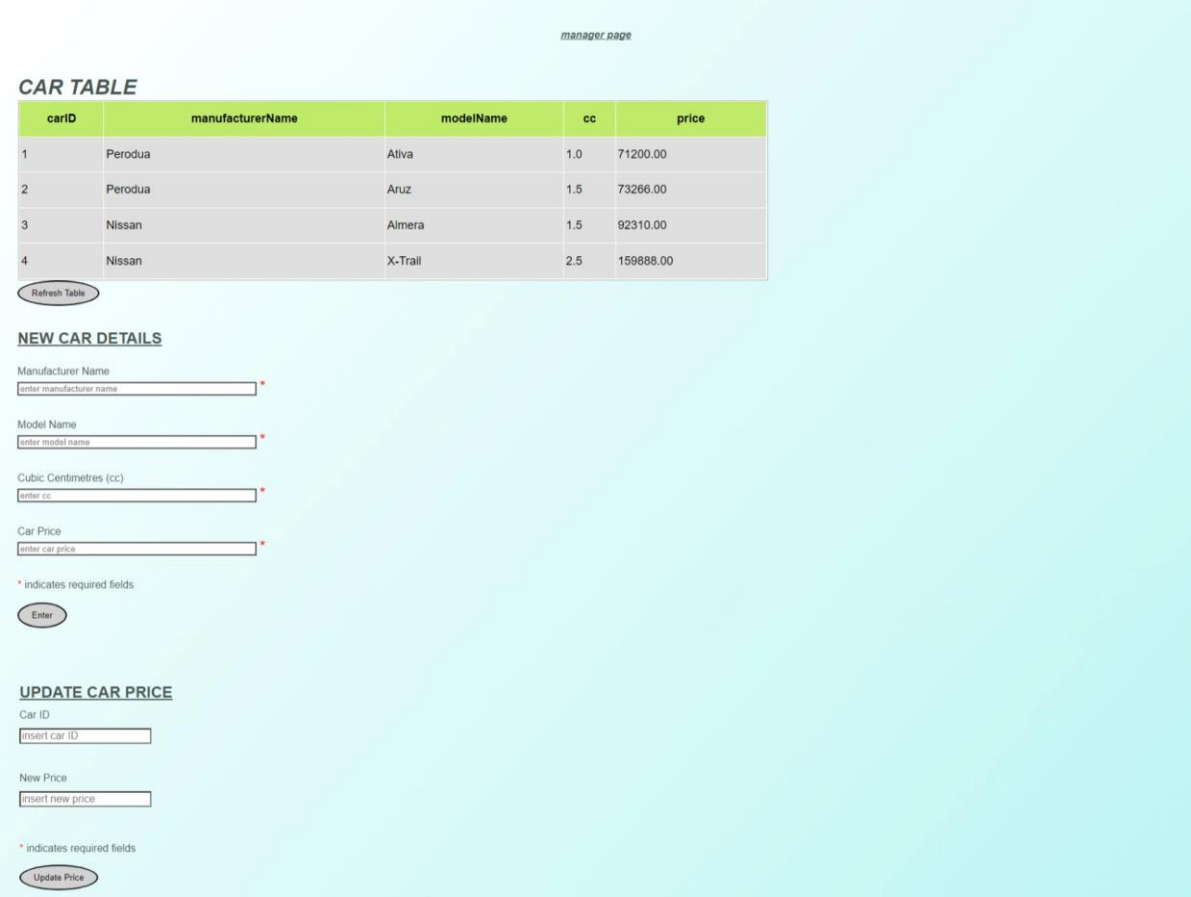
SQL to show table purchasers

SELECT statement:

```
SELECT [purcID], [purcName], [purcAddress], [purcPhone], [carID], [modelName] FROM [purchasers]
```

Manager Page

This page is used by the Car Manager to *insert* a new car details into database table carforsale. The field that can be inserted is manufacturerName, modelName, cc, and price. Other than that, Manager can also *update* a new price of the existing car (only price that can be updated as we cannot change the specs of the car). This page will also shows the carforsale table so the manager can keep track of the data as manager inserting them.



ENTER button

This button will execute sql to insert certain car details without bothering the primary key

```
protected void btnEnter_Click(object sender, EventArgs e)
{
    string sql;
    string manName, modelName;
    decimal cc, price;

    manName = txtMan.Text;
    modelName = txtModel.Text;
    cc = Convert.ToDecimal(txtCC.Text);
    price = Convert.ToDecimal(txtPrice.Text);

    sql = "insert into carforsale (manufacturerName, modelName, cc, price) " +
        "values('" + manName + "', '" + modelName + "',  " + cc + ",  " + price + ")";

    executeSQL(sql, "insert");
}
```

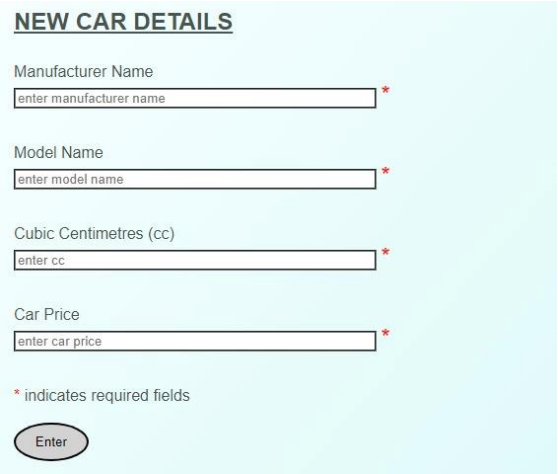
UPDATE button

This button will execute sql to update price field into the existing car ID

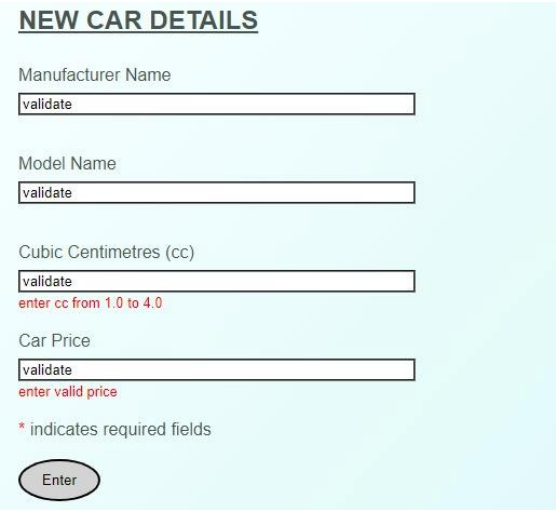
```
protected void btnUpdate_Click(object sender, EventArgs e)
{

    string sql;
    int carID;
    decimal newPrice;

    carID = Convert.ToInt32(txtCarID.Text);
    newPrice = Convert.ToDecimal(txtUpdatePrice.Text);

    sql = "update carforsale set price = '" + newPrice + "' where carID = '" + carID + "'";

    executeSQL(sql, "update");

}
```

For validation, it is divided into two sections which are group1 and group2. Validation group1 is to validate the NEW CAR DETAILS section and group 2 is for UPDATE CAR PRICE section.

NEW CAR DETAILS



Required field validator is set on all of the field, it would only shows star(*) with reminder below so that another related validator can be fit as well.



Range validator is set on two of them:

1) Cubic Centimetres –
   Minimum 1.0 with Maximum 4.0

2) Car Price –
   Minimum 0, Maximum 1000000

UPDATE CAR PRICE

Required field validator been set on all of the field, it would only shows star(*) with indication below so that another related validator can fit as well.

Range validator is set on two of them:

1) Car ID –
   Minimum 1 with Maximum 100

2) Car Price –
   Minimum 0, Maximum 1000000

Once the inputs has passed the validator, the entered values will be filled into the table carforsale.

Salesman Page

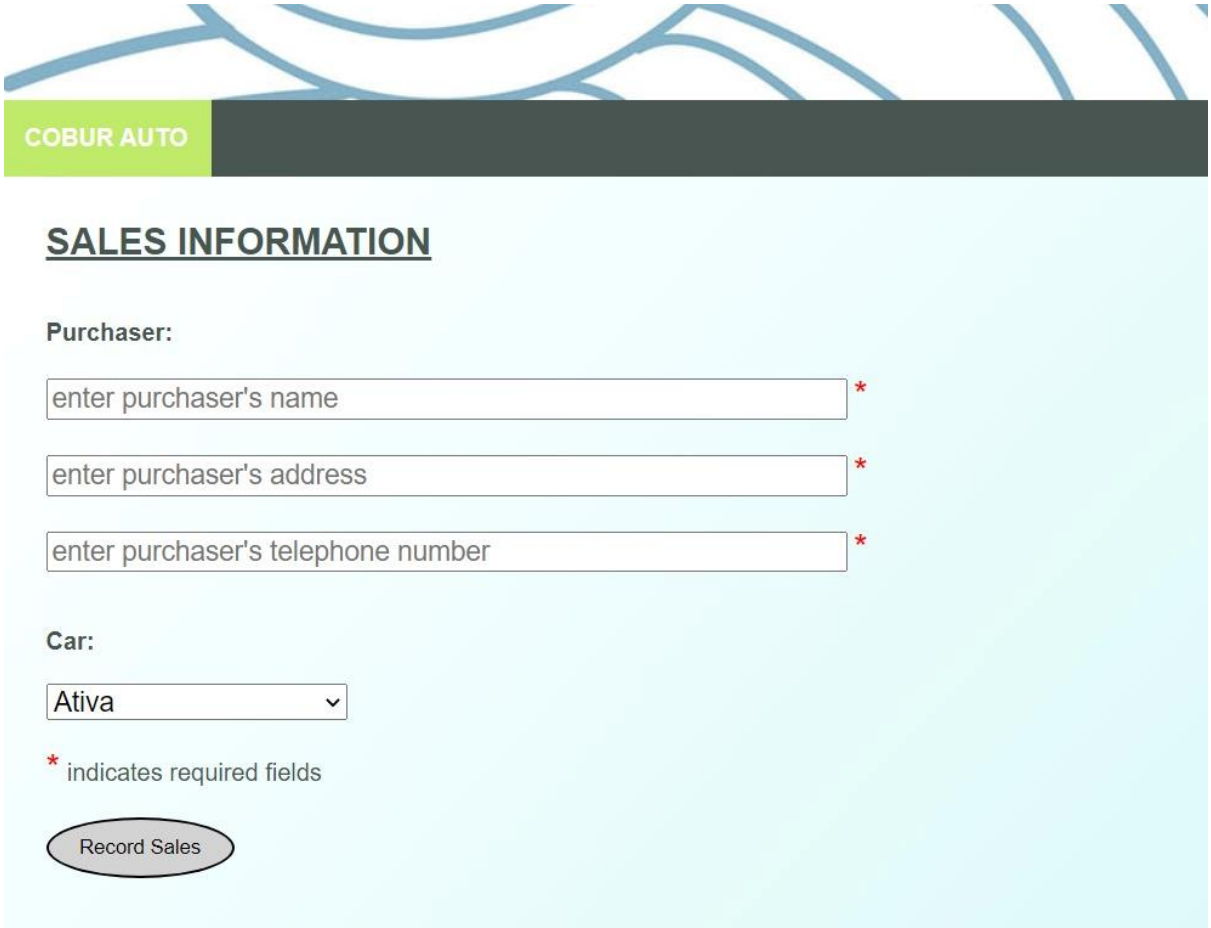
This page is used by Car Salesman to view the list of cars that have been entered by the Manager. The Salesman can filter to find cars by the range of cc. The filtered car will be shown in ListView1.

Filter button

Code behind page for filter button:

```
protected void btnFilter_Click(object sender, EventArgs e)
{
    string sql;
    SqlCommand command;
    DataTable dt = new DataTable();
    decimal cc = Convert.ToDecimal(ddlFilter.SelectedValue);

    sql = "select * from dbo.carforsale where cc = '" + cc + "' ";

    connection.Open();
    command = new SqlCommand(sql, connection);
    SqlDataAdapter da = new SqlDataAdapter(command);
    da.Fill(dt);
    command.ExecuteNonQuery();

    if (dt.Rows.Count > 0)
    {
        ListView1.DataSource = dt;
        ListView1.DataBind();
    }
    else
    {
        Response.Write("No Record");
    }
    command.Dispose();
    connection.Close();
}
```

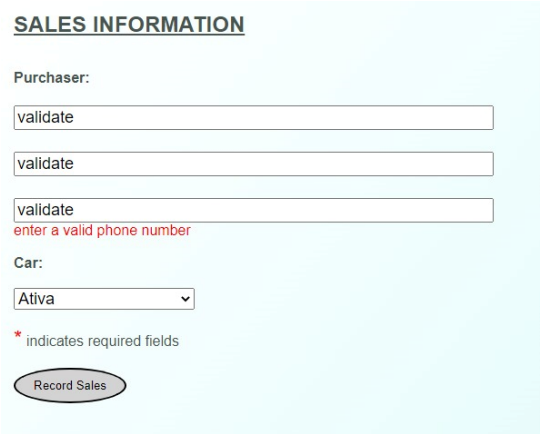
Salesman can then click the `Record Sales` on **Salesman page** button to redirect to the **Sales Info page** to record sales made. The record will contain purcName, purcAddress, purcPhone, and Car Details (CarID, modelName).



Several validator has been set onto those fields;

1) Required Field Validator:
   - All of the textboxes

2) Regular Expression Validator:
   - txtphone = 11 digits



Once the inputs passed all the validator, Salesman may then click on the `Record Sales` to record all data into table purchasers and to redirect Salesman to the **Total Sales page.**

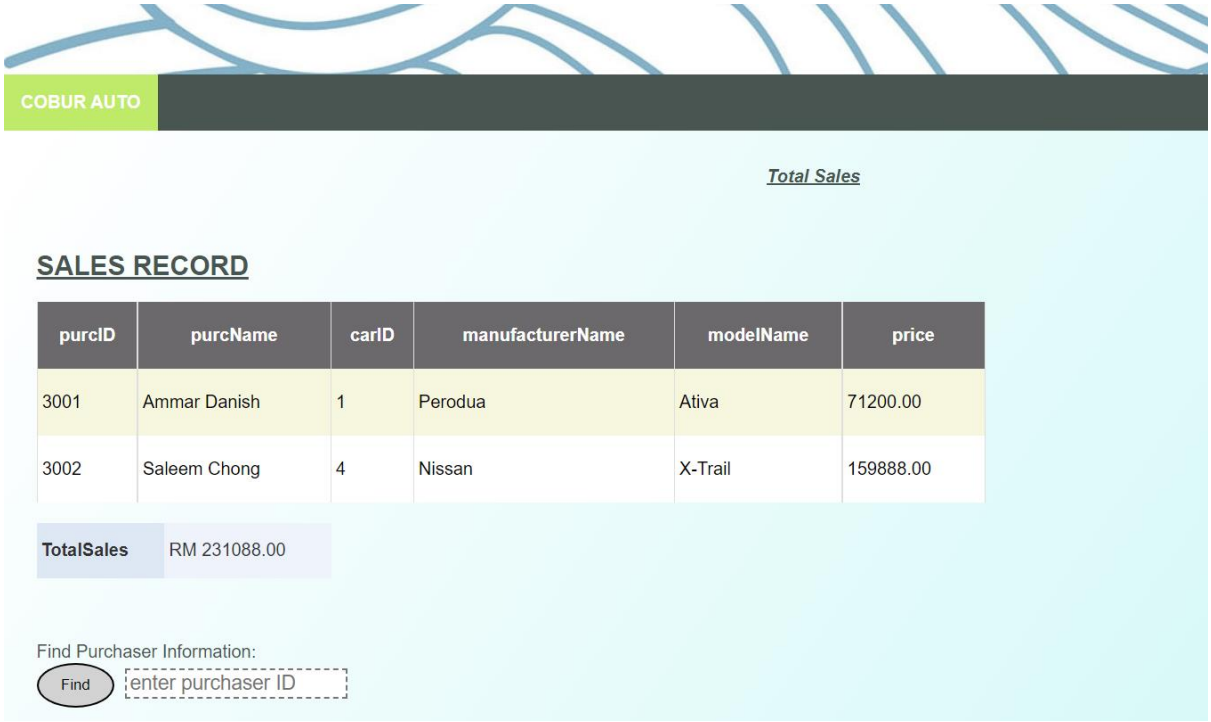
Record Sales button

This button will execute sql to insert purchasers data without bothering the purcID

```
protected void btnRecord_Click(object sender, EventArgs e)
{
    string sql;
    string name, address, modelName;
    int phone, carID;

    name = txtName.Text;
    address = txtAddress.Text;
    phone = Convert.ToInt32(txtPhone.Text);
    carID = Convert.ToInt32(ddlCar.SelectedValue);
    modelName = ddlCar.SelectedItem.Text;

    sql = "insert into purchasers (purcName, purcAddress, purcPhone, carID, modelName) values " +
        "('" + name + "', '" + address + "', '" + phone + "', '" + carID + "', '" + modelName + "')";

    executeSQL(sql, "insert");

    Response.Redirect("TotalSales.aspx");
}
```

Total Sales Page

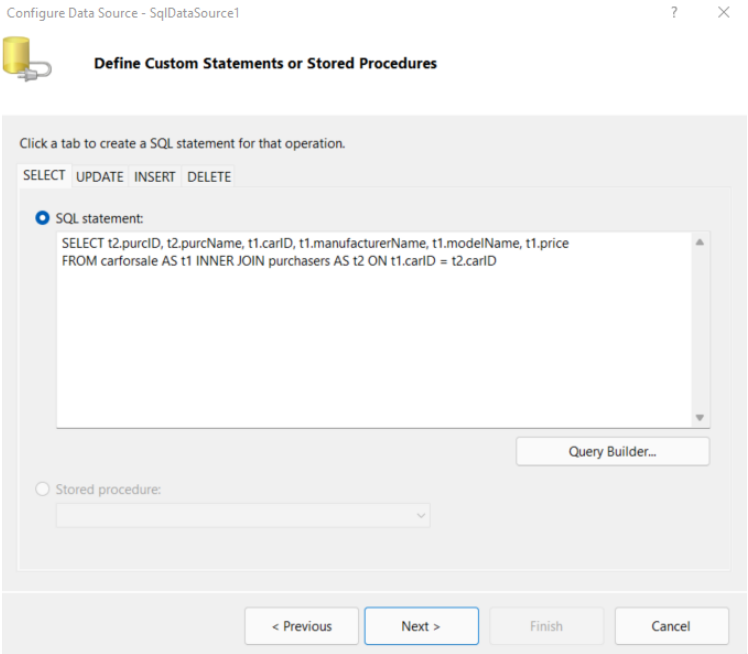
This page contains table of purchasers data along with the car price and the total sales made by salesman below the table.

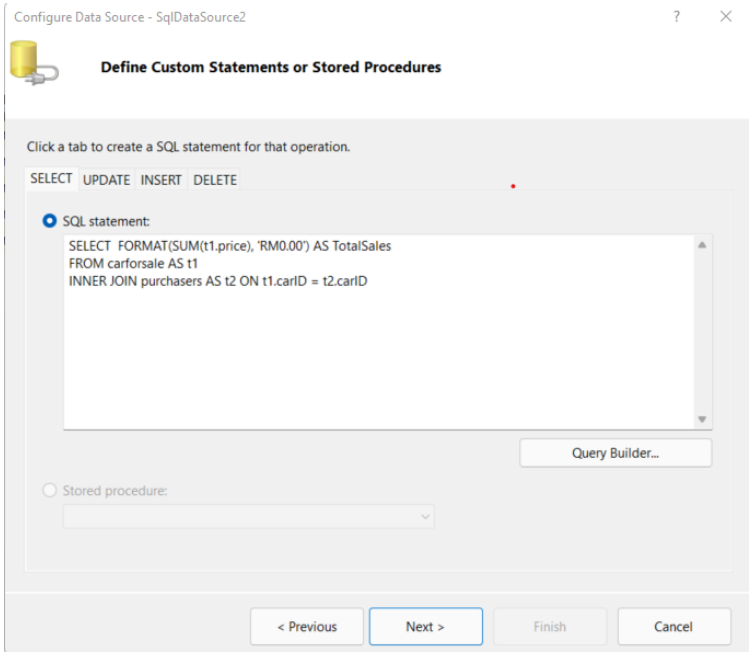


----------→

SQL used to show the table of purchasers that also joined by carPrice from table carfrosale.

! purcAddress and purcPhone will not be shown.



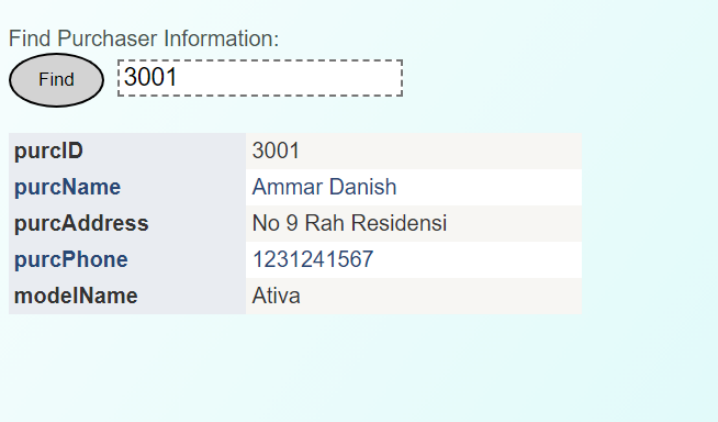


←----------

SQL used to show the total price of sales made by Salesman

The Find button is used to find the details of the purchasers, which in this case is purcAddress and purcPhone as it is not shown on the table above

range validator is set on the field with minimum 3000 and maximum 3999



Code behind page for Find button

```
protected void btnFind_Click(object sender, EventArgs e)
{
    string sql;
    SqlCommand command;
    DataTable dt = new DataTable();
    int purcID = Convert.ToInt32(txtFind.Text);

    sql = "select * from dbo.purchasers where purcID = '" + purcID + "' ";

    connection.Open();
    command = new SqlCommand(sql, connection);
    SqlDataAdapter da = new SqlDataAdapter(command);
    da.Fill(dt);
    command.ExecuteNonQuery();

    if (dt.Rows.Count > 0)
    {
        DetailsView1.DataSource = dt;
        DetailsView1.DataBind();
    }
    else
    {
        Response.Write("No Record");
    }
    command.Dispose();
    connection.Close();
}
```

THANK YOU