**JsonResult Actions and some LINQ:**

**//HW7 Get Request:**

private SearchesContext db = new SearchesContext();

// GET: API Makes a Get Request to giphy.com and returns json data

[HttpGet]

public JsonResult Picture(string id)

{

//get my apiKey for giphy

string apiKey = System.Web.Configuration.WebConfigurationManager.AppSettings["CS460ConfirmationKey"];

//build URL for GET request to giphy, including my key and url parameter

string urlInfo = "https://api.giphy.com/v1/stickers/translate?api\_key=" + apiKey + "&s=" + id;

//Make a request using my urlInfo, then grab response information

WebRequest request = WebRequest.Create(urlInfo);

WebResponse response = request.GetResponse();

Stream information = response.GetResponseStream();

StreamReader reader = new StreamReader(information);

//Grab full response information, read to the end of the data and put it into a string

string responseString = reader.ReadToEnd();

//Parse through the JSON object

var jsonSerialize = new System.Web.Script.Serialization.JavaScriptSerializer();

var data = jsonSerialize.DeserializeObject(responseString);

//create a database search object based on the GET request user information

Search logItem = new Search();

logItem.SearchPhrase = id;

logItem.IpAddress = Request.UserHostAddress;

logItem.Browser = Request.UserAgent;

//Log and Save the request to the DB

db.Searches.Add(logItem);

db.SaveChanges();

//Close the streams

reader.Close();

information.Close();

response.Close();

//Return JSON obj back to javascript for display purposes

return Json(data, JsonRequestBehavior.AllowGet);

}

**//HW8 Return DB Info in a list format:**

private EFAuctionContext db = new EFAuctionContext();

// GET: Returns DB information for details viewpage and returns result in json format

[HttpGet]

public JsonResult List(int? id)

{

//string to return no data if there are no bids on item

string nothing = "";

//new dashboard item created

BidDashBoardVM bidList = new BidDashBoardVM();

//item is pulled from database based on item id

bidList.Item = db.Items.Find(id);

//its list of bids is pulled also based on its specific id

bidList.Bids = db.Bids.Where(x => x.ItemID.Equals(1001)).OrderByDescending(i => i.Price).ToList();

//if the bids list is greater than 0

if (bidList.Item.Bids.Count > 0) {

//create a new var list of bids allowing me to select and name specific columns

var list = db.Bids

.Select(i => new {Ident = i.ItemID, Name = i.BuyerFullName, Amount = i.Price, CreateDate = i.TimeStamp })

.OrderByDescending(i => i.Amount)

.Where(i => i.Ident == id)

.ToList();

//return this list in json format for processing

return Json(list, JsonRequestBehavior.AllowGet);

}

//return an empty string if there are no bids

return Json(nothing, JsonRequestBehavior.AllowGet);

}

**//More Link Queries:**

PersonDashBoardVM vm = new PersonDashBoardVM();

//set vm.Person to database information for that person based on their personal id

vm.Person = db.People.Find(id);

if (vm.Person == null)

{

return HttpNotFound();

}

//If the person has customer data, they are a customer

if (vm.Person.Customers2.Count() > 0)

{

//View information will display in view

ViewBag.Check = true;

//Set an int linking the person id and the customer information id

int custId = db.People.Find(id).Customers2.FirstOrDefault().CustomerID;

//Set vm.Customer information to the first customer object in person's customer database info

vm.Customer = vm.Person.Customers2.FirstOrDefault();

//Set vm.Orders to that customers list of orders placed

vm.Orders = vm.Customer.Orders.ToList();

//Set vm.OrderTotal to the total amount of orders placed by customer

vm.OrderTotal = vm.Customer.Orders.Count;

//Get a list of orders to sum the total amount spent in orders

var total = vm.Orders.SelectMany(i => i.Invoices).SelectMany(inv => inv.InvoiceLines).Select(price => new { price.ExtendedPrice }).ToList();

//Set vm.GrossSales to the total of that list based on ExtendedPrice column

vm.GrossSales = total.Select(i => i.ExtendedPrice).Sum();

//Get a second list of order information for LineProfit data

var secondTotal = vm.Orders.SelectMany(i => i.Invoices).SelectMany(inv => inv.InvoiceLines).Select(price => new

{ price.LineProfit}).ToList();

//Set vm.GrossProfit to the sum of second list

vm.GrossProfit = secondTotal.Select(i => i.LineProfit).Sum();

//Set vm.InvoiceLines list to the list of the top ten most expensive items purchased by customer

vm.InvoiceLines = vm.Orders.SelectMany(i => i.Invoices).SelectMany(inv => inv.InvoiceLines)

.OrderByDescending(i => i.LineProfit).Take(10);

}

//Return vm object for viewing

return View(vm);

}