Using W3Schools.com and SQLZoo.Net as resources, complete the following lab.

Ray Henry is the owner of a bookstore chain named Henry Books. Ray has decided to store his data in a SQL 2005 database. He wants to ensure his data is safe, current, and accurate. He would also like to be able to ask questions concerning his data and have you obtain the answers for him quickly and concisely. A new database has been completed and all of his data has been imported into this new database. You; as the companies super database guru/king; are tasked with attaching this new database to your system and answering the questions Mr. Henry has asked.

Attach the HENRY database. Write the SQL query that answers the following scenarios.

1. List the book code and book title of each book. (33 Rows)

var query1 = Books.Select(b => new{   
                                    b.BOOK\_CODE,   
                                    b.BOOK\_TITLE});   
query1.Dump();

1. List the complete PUBLISHER table. (28 rows)

var query2 = Publishers;   
query2.Dump();

1. List the name of each publisher located in Boston. (2 rows)

var query3 = Publishers.Where(p => p.PUBLISHER\_CITY == "Boston")   
                       .Select(p => new{ p.PUBLISHER\_NAME, p.PUBLISHER\_CITY });   
query3.Dump();

1. List the name of each publisher not located in Boston. (26 rows)

var query4 = Publishers.Where(p => p.PUBLISHER\_CITY != "Boston")   
                       .Select(p => new{ p.PUBLISHER\_NAME, p.PUBLISHER\_CITY })   
                       .OrderBy(p => p.PUBLISHER\_CITY);   
query4.Dump();

1. List the name of each branch that has at least nine employees. (3 rows)

var query5 = Branches.Where(b => b.BRANCH\_EMPS >= 9)   
                       .Select(b => b)   
                       .OrderByDescending(b => b.BRANCH\_EMPS);   
query5.Dump();

1. List the book code and book title of each book that has the type SFI. (3 rows)

var query6 = Books.Where(b => b.BOOK\_TYPE == "SFI")   
                       .Select(b => new{ b.BOOK\_CODE,   
                                            b.BOOK\_TITLE,   
                                            b.BOOK\_TYPE});   
                          
query6.Dump();

1. List the book code and book title of each book that has the type SFI and is in paperback. (1 row)

var query7 = Books.Where(b => b.BOOK\_TYPE == "SFI" &&   
                              b.BOOK\_PAPERBACK == "Y")   
                       .Select(b => new{ b.BOOK\_CODE,   
                                            b.BOOK\_TITLE,   
                                            b.BOOK\_TYPE});   
                          
query7.Dump();

1. List the book code and book title of each book that has the type SFI or is published by the publisher with code SC. (5 rows)

var query8 = Books.Where(b => b.BOOK\_TYPE == "SFI" ||   
                              b.PUBLISHER\_CODE == "SC")   
                       .Select(b => new{ b.BOOK\_CODE,   
                                            b.BOOK\_TITLE,   
                                            b.BOOK\_TYPE,   
                                         b.PUBLISHER\_CODE});   
                          
query8.Dump();

1. List the book code, book title, and price of each book with a price between $20 and $30. (4 rows)

var query9 = Books.Where(b => b.BOOK\_PRICE >= 20m &&   
                              b.BOOK\_PRICE <= 30m)   
                       .Select(b => new{ b.BOOK\_CODE,   
                                            b.BOOK\_TITLE,   
                                            b.BOOK\_TYPE,   
                                         b.PUBLISHER\_CODE});   
                          
query9.Dump();

1. List the book code and book title of each book that has the type MYS and a price of less than $20. (2 rows)

var query10 = Books.Where(b => b.BOOK\_TYPE == "MYS" &&   
                              b.BOOK\_PRICE < 20m)   
                       .Select(b => new{ b.BOOK\_CODE,   
                                            b.BOOK\_TITLE,   
                                            b.BOOK\_TYPE,   
                                         b.BOOK\_PRICE});   
                          
query10.Dump();

1. Customers who are part of a special program get a 10% discount off regular book prices. List the book code, book title, and discounted price for each book. Use discounted\_price as the name for the computed column, which should calculate 90% of the current price; that is, 100% less a 10% discount. (33 rows)

var query11 = Books.Select(b => new   
                                        { b.BOOK\_CODE,   
                                            b.BOOK\_TITLE,   
                                            b.BOOK\_TYPE,   
                                         b.BOOK\_PRICE})   
                   .OrderBy(b => b.BOOK\_PRICE);   
                          
query11.Dump();

1. Find the name of each publisher containing the word “and”. (4 rows)

var query12 = Publishers.Where(p => p.PUBLISHER\_NAME.Contains("and"))   
                        .Select(p => p.PUBLISHER\_NAME);                          
query12.Dump();

1. List the book code and book title of each book that has the type SFI, MYS, or ART. (9 rows)

var query13 = Books.Where(b => b.BOOK\_TYPE == "SFI" ||   
                               b.BOOK\_TYPE == "MYS" ||   
                               b.BOOK\_TYPE == "ART")   
                   .Select(b => new {   
                                     b.BOOK\_CODE,   
                                  b.BOOK\_TITLE,   
                                  b.BOOK\_TYPE})   
                   .OrderBy(b => b.BOOK\_TYPE);   
query13.Dump();

1. Same as #13 only list the books in alphabetical order by title. (9 rows)

var query14 = Books.Where(b => b.BOOK\_TYPE == "SFI" ||   
                               b.BOOK\_TYPE == "MYS" ||   
                               b.BOOK\_TYPE == "ART")   
                   .Select(b => new {   
                                     b.BOOK\_CODE,   
                                  b.BOOK\_TITLE,   
                                  b.BOOK\_TYPE})   
                   .OrderBy(b => b.BOOK\_TITLE);   
query14.Dump();

1. Same as #13 but also include the price and list the books in descending order by price. Within a group of books having the same price, further order by book title. (9 rows)

var query15 = Books.Where(b => b.BOOK\_TYPE == "SFI" ||   
                               b.BOOK\_TYPE == "MYS" ||   
                               b.BOOK\_TYPE == "ART")   
                   .Select(b => new {   
                                     b.BOOK\_CODE,   
                                  b.BOOK\_TITLE,   
                                  b.BOOK\_TYPE,   
                                  b.BOOK\_PRICE})   
                   .OrderByDescending(b => b.BOOK\_PRICE);   
query15.Dump();

1. Display the list of book types in the database. List each book type only once. (12 rows)

var query16 = Books.Select(b => b.BOOK\_TYPE)   
                   .Distinct();   
query16.Dump();

1. How many books have the type SFI? (write sql to produce the answer of 3)

var query17 = Books.Count(b => b.BOOK\_TYPE == "SFI");   
query17.Dump();

1. For each type of book, list the type and the average price. (Research: GROUP BY - 12 rows)

var query18 = Books.GroupBy(b => b.BOOK\_TYPE)   
                   .Select(g => new { Type = g.Key,   
                                         AveragePrice =    
                                      g.Average(b => b.BOOK\_PRICE)});   
query18.Dump();

1. Same as #18, but consider only paperback books. (10 rows)

var query19 = Books.Where(b => b.BOOK\_PAPERBACK == "Y")   
                   .GroupBy(b => b.BOOK\_TYPE)   
                   .Select(g => new { Type = g.Key,   
                                         AveragePrice =    
                                      g.Average(b => b.BOOK\_PRICE)});   
query19.Dump();

1. Same as #18, but consider only paperback books for those types which the average price is more than $10. (Research: HAVING. 3 rows)

var query20 = Books.Where(b => b.BOOK\_PAPERBACK == "Y")   
                   .GroupBy(b => b.BOOK\_TYPE)   
                   .Where(g => g.Average(b => b.BOOK\_PRICE) > 10)   
                   .Select(g => new { Type = g.Key,   
                                         AveragePrice =    
                                      g.Average(b => b.BOOK\_PRICE)});   
query20.Dump();

1. What is the most expensive book in the database? (Nested Select - 1 row)

var query21 = Books.OrderByDescending(b => b.BOOK\_PRICE)   
                   .FirstOrDefault();   
query21.Dump();

1. What are the title(s)/price(s) of the least expensive book(s) in the DB? (3 rows)

var query22 = Books.OrderBy(b => b.BOOK\_PRICE)   
                   .Take(3);   
query22.Dump();

1. How many employees does Henry Books have?

var query23 = Branches.Sum(b => b.BRANCH\_EMPS);   
query23.Dump();

Console.WriteLine($"There are {query23} total employees at Henry Books");

**Multiple Table**

1. For each book, list the book code, book title, publisher code, and the publisher name. Order the results by publisher name. (33 rows)

var query1 = from b in Books   
        join p in Publishers on b.PUBLISHER\_CODE equals p.PUBLISHER\_CODE   
              orderby p.PUBLISHER\_NAME   
              select new { b.BOOK\_CODE, b.BOOK\_TITLE,   
                             b.PUBLISHER\_CODE, p.PUBLISHER\_NAME};   
query1.Dump();

1. For each book published by Plume, list the book code, book title, and price. (3 rows)

var query2 = from b in Books   
              join p in Publishers on b.PUBLISHER\_CODE equals p.PUBLISHER\_CODE   
              where p.PUBLISHER\_NAME == "Plume"   
              select new { b.BOOK\_CODE, b.BOOK\_TITLE,   
                             b.PUBLISHER\_CODE, p.PUBLISHER\_NAME};   
query2.Dump();

1. List the book title, book code, and price of each book published by Plume that has a book price of at least $14. (1 row)

var query3 = from b in Books   
              join p in Publishers on b.PUBLISHER\_CODE equals p.PUBLISHER\_CODE   
              where p.PUBLISHER\_NAME == "Plume" &&   
                      b.BOOK\_PRICE >= 14m   
              select new { b.BOOK\_CODE, b.BOOK\_TITLE,   
                             b.PUBLISHER\_CODE, p.PUBLISHER\_NAME};   
query3.Dump();

1. List the book code, book title, and units on hand for each book in branch number 4. (9 rows)

var query4 = from i in Inventories   
              join b in Books on i.BOOK\_CODE equals b.BOOK\_CODE   
              where i.BRANCH\_NUMBER == 4   
              select new { b.BOOK\_CODE, b.BOOK\_TITLE,   
                             i.ON\_HAND, i.BRANCH\_NUMBER};   
query4.Dump();

1. List the book title for each book that has the type PSY and that is published by Jove Publications. (1 row)

var query5 = from b in Books   
             join p in Publishers on b.PUBLISHER\_CODE equals p.PUBLISHER\_CODE   
             where b.BOOK\_TYPE == "PSY" &&   
                    p.PUBLISHER\_NAME == "Jove Publications"   
              select b.BOOK\_TITLE;   
query5.Dump();

1. Find the book title for each book written by author number 18. Use the IN operator in your formulation. (2 rows)

var query6 = from b in Books   
             join w in Wrotes on b.BOOK\_CODE equals w.BOOK\_CODE   
             where w.AUTHOR\_NUMBER == 18   
             select b.BOOK\_TITLE;   
query6.Dump();

1. Repeat exercise 6 but use the EXISTS operator. (2 rows)

var query7 = from b in Books   
             where Wrotes.Any(w => w.BOOK\_CODE == b.BOOK\_CODE &&   
                                    w.AUTHOR\_NUMBER == 18)   
             select b.BOOK\_TITLE;   
query7.Dump();

1. Find the book codes and book title for each book located in branch number 2 and written by author 20. (1 row)
2. List book codes for each pair of books that have the same price. (on such pair would be book 0200 and book 7559 because the price of both books is $8.00) Sort the results by first book code and then by second book code. (11 rows)

var query9 = Books.SelectMany(b1 => Books.Where(b2 => b2.BOOK\_PRICE ==   
                   b1.BOOK\_PRICE && b2.BOOK\_CODE != b1.BOOK\_CODE)   
       .Select(b2 => new { Book1 = b1.BOOK\_CODE, Book2 = b2.BOOK\_CODE}))   
                  .Distinct();   
query9.Dump();

1. Find the book title, author last name, and units on hand for each book in branch number 4. (10 rows)

var query10 = from i in Inventories   
              join b in Books on i.BOOK\_CODE equals b.BOOK\_CODE   
              join w in Wrotes on b.BOOK\_CODE equals w.BOOK\_CODE   
             join a in Authors on w.AUTHOR\_NUMBER equals a.AUTHOR\_NUMBER   
              where i.BRANCH\_NUMBER == 4   
              select new {b.BOOK\_TITLE, a.AUTHOR\_LAST,   
                            i.ON\_HAND, i.BRANCH\_NUMBER};   
query10.Dump();

1. Repeat exercise 10 but list only paperback books. (4 rows)

var query11 = from i in Inventories   
              join b in Books on i.BOOK\_CODE equals b.BOOK\_CODE   
              join w in Wrotes on b.BOOK\_CODE equals w.BOOK\_CODE   
             join a in Authors on w.AUTHOR\_NUMBER equals a.AUTHOR\_NUMBER   
              where i.BRANCH\_NUMBER == 4 &&   
                      b.BOOK\_PAPERBACK == "Y"   
              select new {b.BOOK\_TITLE, a.AUTHOR\_LAST,   
                            i.ON\_HAND, i.BRANCH\_NUMBER};   
query11.Dump();

1. Find the book code and book title for each book whose price is more than $10 and that was published in Boston. (Research Union keyword. 20 rows)

var query12 = from b in Books   
              join p in Publishers on b.PUBLISHER\_CODE equals p.PUBLISHER\_CODE   
              orderby p.PUBLISHER\_CITY, b.BOOK\_PRICE   
              where b.BOOK\_PRICE > 10 &&   
                      p.PUBLISHER\_CITY == "Boston"   
                 
              select new {b.BOOK\_TITLE, b.BOOK\_CODE,   
                            b.BOOK\_PRICE, p.PUBLISHER\_CITY};   
query12.Dump();

1. Find the book code and book title for each book whose price is more than $10 and that was not published in Boston. (18 rows)

var query13 = from b in Books   
              join p in Publishers on b.PUBLISHER\_CODE equals p.PUBLISHER\_CODE   
              orderby p.PUBLISHER\_CITY, b.BOOK\_PRICE   
              where b.BOOK\_PRICE > 10 &&   
                      p.PUBLISHER\_CITY != "Boston"   
                 
              select new {b.BOOK\_TITLE, b.BOOK\_CODE,   
                            b.BOOK\_PRICE, p.PUBLISHER\_CITY};   
query13.Dump();

1. Find the book code and book title for each book whose price is greater than the book price for every book that has the type HOR. (5 rows)
2. Find the book code and book title for each book whose price is greater than the price of at least one book that has the type HOR. (23 rows)

var query15 = from b in Books   
      where b.BOOK\_PRICE > (from b2 in Books where b2.BOOK\_TYPE == "HOR"   
                                  select b2.BOOK\_PRICE).FirstOrDefault()   
              orderby b.BOOK\_PRICE   
              select new { b.BOOK\_CODE, b.BOOK\_TITLE, b.BOOK\_PRICE };   
query15.Dump();

1. List the book code, book title, and units on hand for each book in branch number 2. Be sure each book is included, regardless of whether there are copies of the book currently on hand in branch 2. Order the output by book code. (Research Left Join: 33 rows)

var query16 = from b in Books   
         join i in Inventories on b.BOOK\_CODE equals i.BOOK\_CODE into invGroup   
              from inv in invGroup.DefaultIfEmpty()   
              where inv.BRANCH\_NUMBER == 2 || inv == null   
              orderby b.BOOK\_CODE   
              select new { b.BOOK\_CODE, b.BOOK\_TITLE,    
                           ON\_HAND = inv != null ? inv.ON\_HAND : 0 };   
 query16.Dump();