**Project 3 (SQL)**

|  |
| --- |
| **Project Three-Part 1- Northwoods University Design -**  The Northwoods University Student Registration Database  Northwoods University has decided to replace its aging mainframe-based student registration system with a more modern client/server database system.  School officials want students to be able to retrieve course availability information, register for courses, and print transcripts using personal computers located in the student computer labs. In addition, faculty members must be able to retrieve student course lists, drop and add students, and record course grades. Faculty members must also be able to view records for the students they advise.  Security is prime concern, so student and course records must be protected by password access.  The data items for the Northwoods database are:   * Student name, address, telephone number, class (freshman, sophomore, junior, or senior), date of birth, PIN (personal identification number), and advisor * Course call number (such as COM340), course name, credits, location, duration, maximum enrollment, instructor, and term offered * Instructor name, office location, telephone number, rank (Professor, Instructor, etc.), and PIN * Student enrollment and grade information.   The database must be able to allow multiple sections of the same course to be taught by different instructors and on different days and times.  You have been asked to design a relational database to support the required tasks. You will provide three diagrams for this project.   * A Semantic Object Model of the basic information: Student, Course, and Instructor (this will be considered a conceptual diagram). * An Entity Relationship Diagram (ERD) using the Chen model (this will be considered a more detailed conceptual diagram, showing each table that will exist and how they relate). * A Table Diagram using the Crow’s Foot Model (this will be considered an internal diagram)   The tables will need to be in Third Normal form, which will be covered in Lesson 5. |

|  |
| --- |
| **Project three Part2 : Simple Queries -**  You will need to write SELECT commands to answer the following questions. Print the SELECT command and your results using PHPMyAdmin. Your instructor will give show you how to create the tables**. The table descriptions start on Page 111 (Alpine Inventory database).**   1. Select all the rows from the alp\_item table. Display Description and Category 2. Select alp\_inventory items that have a price of less than 100 dollars. Display Id, Size, Price and Quantity on Hand. 3. List the alp\_inventory items that have a quantity on hand of more than 30. Display Id, Quantity on Hand and Price 4. List the alp\_customers in ‘Washburn ’ and ‘Silver Lake’. Display first\_name, last\_name, mi and city 5. Select the prices that occur in the alp\_inventory table. A specific price should only appear once. Display the Price. 6. Select the alp\_inventory items that are in stock. Display Id, Price and Quantity on Hand 7. Select the alp\_orders placed before November 1, 2007. Display the Order\_id and Order\_date 8. List the alp\_inventory items that are ‘Coral’ or ‘Olive’ and have a Quantity on Hand of less than 105. Display Id and Quantity on Hand 9. List the alp\_items that contain the word ‘Fleece’ in the item description. Display Id, Description, and Category 10. List all the alp\_inventory items that do not have a size or a color assigned. Display the Id and Price. 11. Determine the number of orders placed on 10 October 2007. Display Number of Orders   Hint: Use the COUNT function   1. Determine the extended price for each row in the alp\_orderline table. Display Order\_id, inv\_id, and Extended Price 2. Determine the number of different items on each order. Display Order\_ID and Number of Items Hint: Determine the number of different products ordered, not the total quantity ordered. This query requires a GROUP BY clause. 3. Determine the number of orders placed by each customer. Only display the data for customers who have placed more than one order. Display Cust\_id and Number of Orders Hint: This query requires a GROUP BY clause and a HAVING clause. 4. Determine the order total for each order that has an order total greater than 100. Display ‘*Order Id*’ and ‘*Order Total’.* Make sure the results are in ascending order total sequence. 5. Determine what is the most expensive price, the least expensive price, and the average price in the alp\_inventory table. 6. Now that you know the average price in the inventory table, display all of the information for inventory items whose price is greater than the average price. |