# ITAD 138 SQL

# Midterm Exam

2.5 hours

Midterm Exam is a “hands on”, “open textbook” exam.

**Restrictions**

* You can use the textbook Murach’s MySQL, and no other books.
* Use of e-mail is not allowed during the exam.
* Only workstations in the classroom can be used to take the exam. You cannot use your own laptops.
* No flash drives are allowed in the computer.

**Grading**

* There are 7 questions for a total of 100 points.

**After you completed your work:**

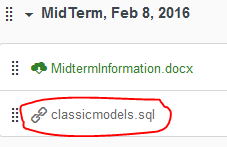
Submit your result

* You should have a file yourName.sql (replace yourName with your name).
* Submit yourName.sql to Canvas.
* Download yourName.sql to a provided flash drive.

Please log in to Canvas.

Download the file classicmodels.sql to generate the database. The file can be found in Canvas under Modules -> Midterm -> classicmodels.sql

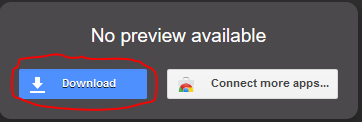
* To download, double left click on ClassicModels.sql as indicated in the snip below



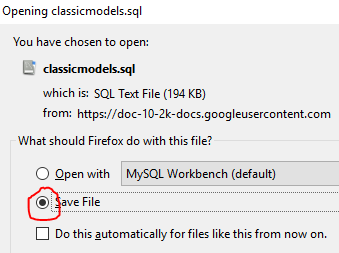
* Next left click on classicmodels.sql



* Next click on Download as shown in the snip below



* Next choose ‘Save File’ as in the snip below



* This should save classicmodels.sql to your download folder

**To Create Database**

Run the script called **classicmodels.sql** using  and . The script classicmodels.sql should be in your download folder. This script creates all the tables and populates them with data.

**Database**

You will be working with database called classicmodels. The database keeps info for a business that sells scale models of cars, motorcycles, ships etc. The database consists of eight tables:

* Customers: customer information
* Orders: Orders placed by customers
* Order Details: Line items within an order.
* Products: The list of scale model cars
* Product Lines: The list of product line classification
* Employees: All employees, including sales reps who work with customers.
* Offices: sales offices
* Payments: Payments made by customers against their account

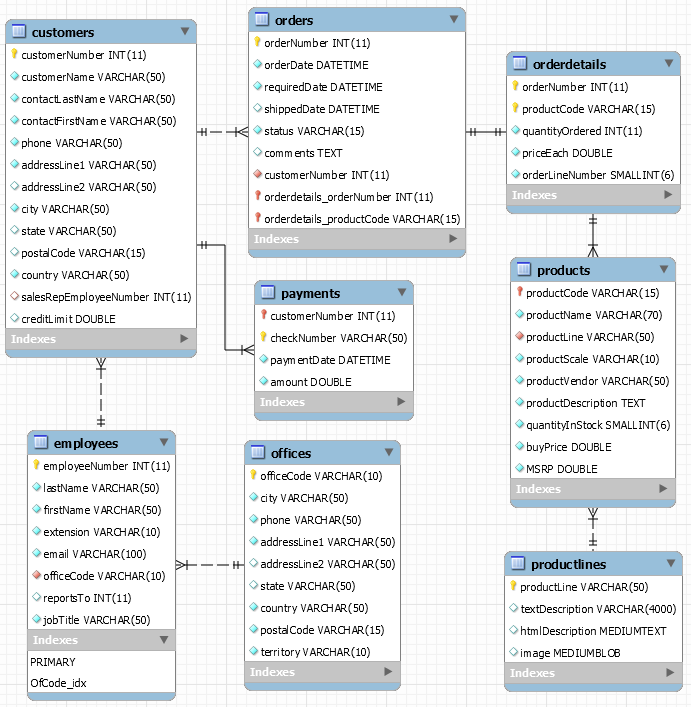
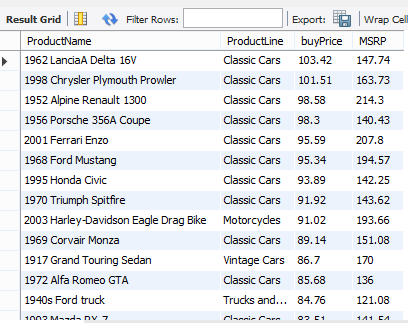
****

Figure EER Diagram for classicmodels

**Questions**

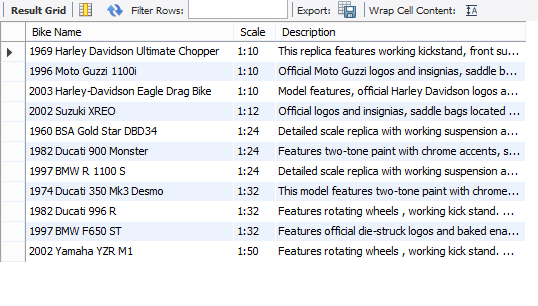
1. (14 pts.) Among the products find those that have ‘buy price’ at least average. Get product name, product line, buy price and MSRP. Sort by buy price in descending order. Hint: Returns 54 rows. See screenshot below.



1. (14 pts.) Find all “Motorcycles” that do not have the scale numbers “1:18”. Include Bike name, scale, and description into the result. Name the columns as in the screenshot below. Sort ascending by Scale and then Bike Name.

Hints:

* Returns 11 rows.
* You will only need the table products. No join is required.

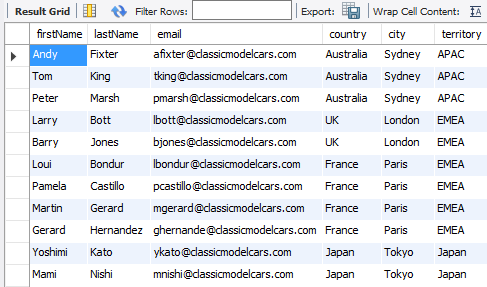


1. (14 pts.) Find first name, last name and e-mails of all sales representatives (as a job title) responsible for territories “Japan”, ”EMEA”, and “APAC”. Include country, city and territory into the resulting table. Sort by territory in ascending, then by country in descending, then by city in ascending order, then by last name ascending, and then by first name ascending.

Hints:

* Returns 11 rows. See screenshot below.
* You will need to join two tables.
* Run the following query to help identify jobTitles

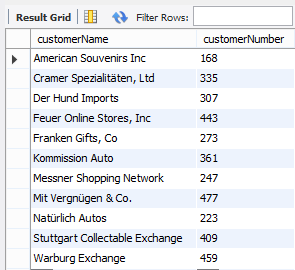
SELECT DISTINCT jobTitle FROM employees;



1. (14 pts.) Find USA and German customers who have not placed any orders. Retrieve customer number and customer name. Sort by customer name.

Hints:

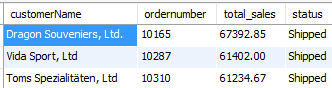
* Returns 11 rows. See screenshot below.
* Use an outer join.



1. (14 pts.) Find all orders that have shipped having total sales for the order greater than $60000. Total sales for an order is the sum of (priceEach\*quantityOrdered). Order by total sales descending. Return columns in the screenshot below.

Hint:

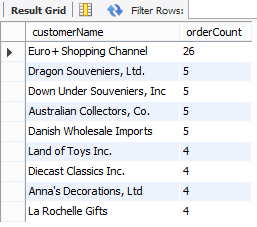
* Returns 3 rows.



1. (15 pts.) Find out how many orders were made by customers with names starting from letters ‘A’, ‘D’, ‘E’ or ‘L’. Retrieve only those who made at least 4 orders. The resulting table should have 2 columns: customerName and orderCount (number of orders made). Sort descending by orderCount.

Hints:

* Returns 9 rows. See screenshot below.



1. (15 pts.) For the two productLines
   * ‘Classic Cars’
   * ‘Vintage Cars’

find the COUNT(\*) and the sum of the quantityOrdered \* priceEach AS sumTotalPrice for

* status = ‘Cancelled’
* status = ‘Disputed’

Order by sumTotalPrice descending.

Hints:

* See the screenshot below.
* you will need to use UNION to combine the results from status = ‘Cancelled’ with the results from status = ‘Disputed’.

