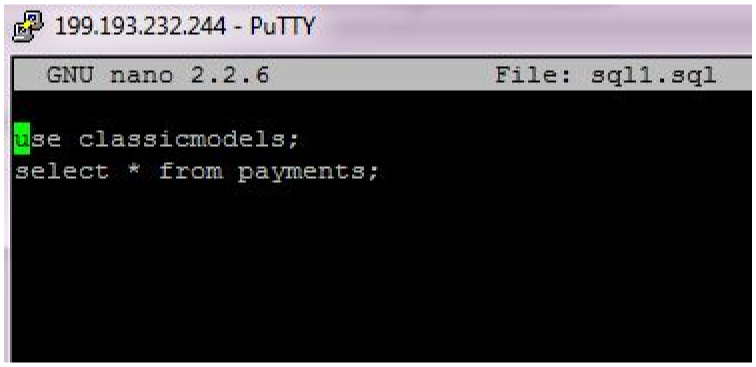
SQL Notes

Running SQL statements from a file

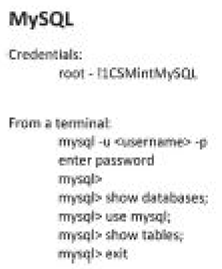
1. Open two putty windows. In one start up MYSQL
2. Use the other for file manipulation

3. The text file sql1 has two statements, one to tell the database, the second to run a select. You can sequence as many statements as necessary.



```
199.193.232.244 - PuTTY
  GNU nano 2.2.6                 File: sql1.sql

use classicmodels;
select * from payments;
```

Starting MYSQL



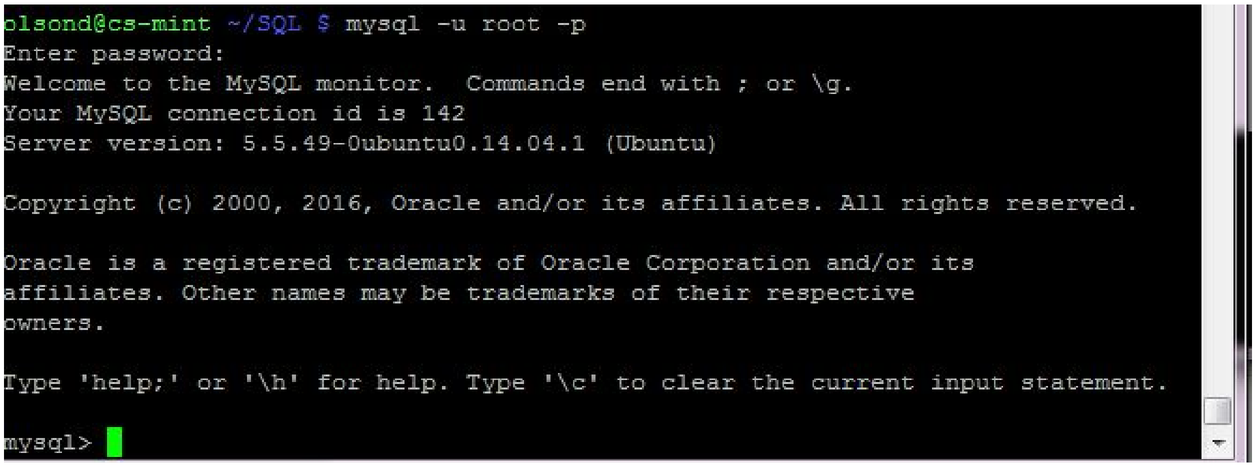
MySQL

Credentials:

```
        root - !1CSMintMySQL
```

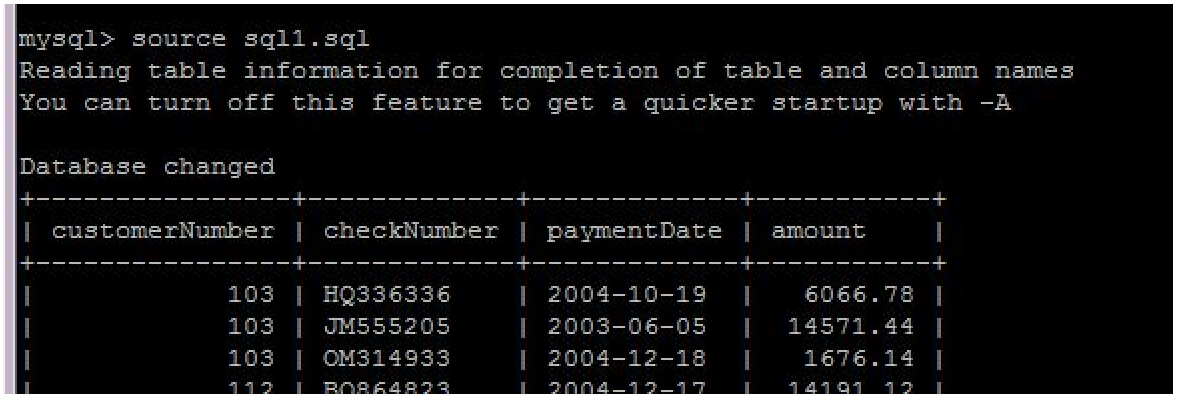
From a terminal:

```
        mysql -u <username> -p
        enter password
        mysql>
        mysql> show databases;
        mysql> use mysql;
        mysql> show tables;
        mysql> exit
```



```
olsond@cs-mint ~/SQL $ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 142
Server version: 5.5.49-0ubuntu0.14.04.1 (Ubuntu)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Shows the commands for starting mysql and the resulting command prompt



```
mysql> source sql1.sql
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
+----------------+-------------+-------------+-----------+
| customerNumber | checkNumber | paymentDate | amount    |
+----------------+-------------+-------------+-----------+
|            103 | HQ336336    | 2004-10-19  |   6066.78 |
|            103 | JM555205    | 2003-06-05  |  14571.44 |
|            103 | OM314933    | 2004-12-18  |   1676.14 |
|            112 | BO864823    | 2004-12-17  |  14191.12 |
```

Running the script sql1.sql

Cape Cod database for use with chapter 2

1. HINT Having two putty windows on the linux machine helps a bunch. One for mySQL and one for the linux command prompt
2. Start up mySQL and create a database for inserting the Cape Cod data
   a. This involves a create database sql statement and
   b. An sql use of the database you created.
3. Moodle contains two SQL files which you should download.
   a. Create02.sql
   b. Insert02.sql

c.

4. Use FTP to upload them to your SQL directory on the Linux machine
5. Maker sure you are using your newly created database.
6. You source to run the Create02.SQL, and InsertO2.sql
7. You should now be able to run the SQL commands in chapter 2

Nested Queries

JOINS

Tables

Code to create the tables. You must create a database to load the tables into.

```
CREATE TABLE department
 (
 DepartmentID INT Primary key,
 DepartmentName VARCHAR(20)
 );

CREATE TABLE employee
(
LastName VARCHAR(20),
DepartmentID INT references department(DepartmentID)
);
INSERT INTO department VALUES(31, 'Sales');
INSERT INTO department VALUES(33, 'Engineering');
INSERT INTO department VALUES(34, 'Clerical');
INSERT INTO department VALUES(35, 'Marketing');
INSERT INTO employee VALUES('Rafferty', 31);
INSERT INTO employee VALUES('Jones', 33);
INSERT INTO employee VALUES('Heisenberg', 33);
INSERT INTO employee VALUES('Robinson', 34);
INSERT INTO employee VALUES('Smith', 34);
INSERT INTO employee VALUES('Williams', NULL)
```

Queries of the tables.

```
+----------------------+
| Tables_in_simpleJoin |
+----------------------+
| department           |
| employee             |
+----------------------+
2 rows in set (0.00 sec)

+--------------+----------------+
| DepartmentID | DepartmentName |
+--------------+----------------+
|           31 | Sales          |
|           33 | Engineering    |
|           34 | Clerical       |
|           35 | Marketing      |
+--------------+----------------+
4 rows in set (0.02 sec)

+------------+--------------+
| LastName   | DepartmentID |
+------------+--------------+
| Rafferty   |           31 |
| Jones      |           33 |
| Heisenberg |           33 |
| Robinson   |           34 |
| Smith      |           34 |
| Williams   |         NULL |
+------------+--------------+
6 rows in set (0.00 sec)
```

```
select * from department;
select * from employee;
```

```
select 'employee NJ department' as '';
select * from employee
NATURAL JOIN department;
```

```
select 'department NJ employee' as '';
select * from department
NATURAL JOIN employee;


select 'employee LOJ department' as '';
select *
FROM employee
LEFT OUTER JOIN department
ON employee.DepartmentID = department.DepartmentID;


use simpleJoin;
show tables;


select * from department;
select * from employee;


select 'employee NJ department' as '';
select * from employee
NATURAL JOIN department;

select 'department NJ employee' as '';
select * from department
NATURAL JOIN employee;


select 'employee LOJ department' as '';
select *
FROM employee
LEFT OUTER JOIN department
ON employee.DepartmentID = department.DepartmentID;

use simpleJoin;
show tables;


select * from department;
select * from employee;
select 'employee NJ department' as '';
select * from employee
NATURAL JOIN department;

select 'department NJ employee' as '';
select * from department
NATURAL JOIN employee;


select 'employee LOJ department' as '';
select *
FROM employee
LEFT OUTER JOIN department
ON employee.DepartmentID = department.DepartmentID;

use simpleJoin;
show tables;


select * from department;
select * from employee;
```

```
select 'employee NJ department' as '';
select * from employee
NATURAL JOIN department;

select 'department NJ employee' as '';
select * from department
NATURAL JOIN employee;


select 'employee LOJ department' as '';
select *
FROM employee
LEFT OUTER JOIN department
ON employee.DepartmentID = department.DepartmentID;


use simpleJoin;
show tables;
```

```
select * from department;
select * from employee;

#a
select 'employee NJ department' as '';
select * from employee
NATURAL JOIN department;


#b
select 'department NJ employee' as '';
select * from department
NATURAL JOIN employee;

#c
select 'employee LOJ department' as '';
select *
FROM employee
LEFT OUTER JOIN department
ON employee.DepartmentID = department.DepartmentID;

#d
select 'department LOJ employee' as '';
select *
FROM department
LEFT OUTER JOIN employee
ON employee.DepartmentID = department.DepartmentID;
```

Stored Procedures

Here are the programs demonstrating stored procedures.

STORED Procedure 1

```
use classicmodels;

drop procedure HelloWorld();

create
procedure HelloWorld();
```

select 'Hello test';

CALL HelloWorld();

STORED procedure 2
use classicmodels;

#2 uses select statement
drop procedure Number2;
create procedure Number2()

Select productCode,priceEach
from orderdetails
where priceEach > 80.00;

CALL Number2();

STORED procedure 3

use classicmodels;

#3 Uses a paramter
drop procedure Number3;
CREATE
PROCEDURE Number3(IN minPrice INT)

Select productCode,priceEach
from orderdetails
where priceEach > minPrice;

CALL Number3(25);

STORED procedure 4
use classicmodels;

#4 changes delimiter

```
DELIMITER $$
drop procedure Number4$$
CREATE
PROCEDURE  Number4(IN minPrice INT)

BEGIN

Select productCode,priceEach
from orderdetails
where  priceEach > minPrice;

END$$

DELIMITER ;
```

STORED procedure 5

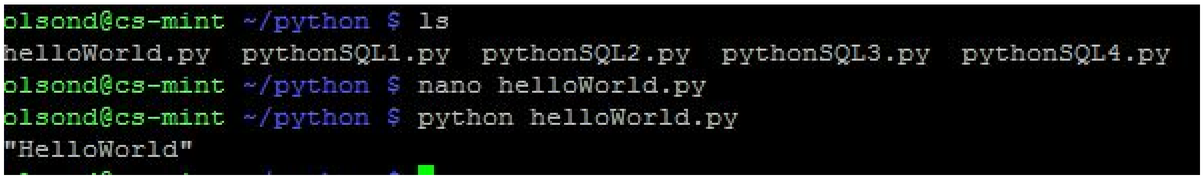
```
#5 Uses a local variable @mult shows how the variable can be assigned from a funtion
DELIMITER $$
drop procedure Number5$$
CREATE
PROCEDURE  Number5(IN minPrice INT)

BEGIN

SELECT @mult := STRCMP('cat','dog');
Select productCode,(priceEach  * @mult)
from orderdetails
where  priceEach > @mult * minPrice;

END$$

DELIMITER ;
```

PYTHON with MySQL

```
#hello world remark
```

```
print '"HelloWorld"';
```

Running helloWorld



```
olsond@cs-mint ~/python $ ls
helloWorld.py  pythonSQL1.py  pythonSQL2.py  pythonSQL3.py  pythonSQL4.py
olsond@cs-mint ~/python $ nano helloWorld.py
olsond@cs-mint ~/python $ python helloWorld.py
"HelloWorld"
```

Simple Python with connection to MySQL

```
import MySQLdb as mdb
import sys

try:
  con=mdb.connect('localhost','root','!1CSMintMySQL','classicmodels');
  cur = con.cursor();
  cur.execute("SELECT VERSION()")

  ver = cur.fetchone()
  print "Database version: %s" % ver

except mdb.Error, e:
  print("error")
  print "Error: %d %s" % e.args[0],e.args[0]
  #print("Error %d: %s" % (e.args[0],e.args[1])
  sys.exit(1)

finally:
    if con:
       con.close()
```

Second program. Executes an embedded SQL. fetchall brings back the entire result set. Maybe too large.

```
import MySQLdb as mdb
import sys

try:
  con=mdb.connect('localhost','root','!1CSMintMySQL','classicmodels');
  cur = con.cursor();
  cur.execute("SELECT customerName,city,country from customers")

  rows = cur.fetchall()
  for row in rows:
    print row

except mdb.Error, e:
  print("error")
  print "Error: %d %s" % e.args[0],e.args[0]
  #print("Error %d: %s" % (e.args[0],e.args[1])
  sys.exit(1)

finally:
     if con:
        print("closing")
        con.close()
```

Third program. Fetchone returns the the first row, must iterate through the result set

```
import MySQLdb as mdb
import sys

try:
  con=mdb.connect('localhost','root','!1CSMintMySQL','classicmodels');
  cur = con.cursor();
  cur.execute("SELECT customerName,city,country from customers limit 10")
  desc = cur.description
  for i in range(cur.rowcount):
    row = cur.fetchone()
    print "%s: %15s" % (desc [0][0], row[0])
    print row[1]
    print row[2]
    print "*******"

except mdb.Error, e:
  print("error")
  print "Error: %d %s" % e.args[0],e.args[0]
  #print("Error %d: %s" % (e.args[0],e.args[1])
  sys.exit(1)

finally:
     if con:
        print("closing")
        con.close()
```

Fourth Program Makes a call to a stored procedure

```
import MySQLdb as mdb
import sys

try:
  con=mdb.connect('localhost','root','!1CSMintMySQL','classicmodels');
  cur = con.cursor();
  cur.execute("call Number4(80)");
  desc = cur.description
  for i in range(cur.rowcount):
    row = cur.fetchone()
    print "%s  %d"% (row[0],row[1])

except mdb.Error, e:
  print("error")
  print "Error: %d %s" % e.args[0],e.args[0]
  #print("Error %d: %s" % (e.args[0],e.args[1])
  sys.exit(1)

finally:
     if con:
        print("closing")
        con.close()
```

Triggers Remember you must create a database before this stuff will work.

```
USE stock;
DROP TABLE invItems;
CREATE TABLE invItems (itemNumber int auto_increment primary key ,
                       description VARCHAR(20),
                       supplier VARCHAR(20),
                        inStock  CHAR(1),
                        date DATE,
                        MSRP Decimal(19,2),
                        discountPercent Decimal(4,2),
                        discount Decimal(10,2)
);
```

```
DROP TABLE invHistory;
CREATE TABLE invHistory (itemNumber int,
                        user VARCHAR(20),
                        description VARCHAR(20),
                        supplier VARCHAR(20),
                         date DATE,
                         MSRP Decimal(19,2),
                         discountPercent Decimal(4,2),
                         discount Decimal(10,2),
                         ACTION VARCHAR(20)
);

DELIMITER $$

##UPDATE TRIGGER

Create TRIGGER  account_before_update
BEFORE UPDATE ON invItems
FOR EACH ROW


INSERT INTO invHistory
set itemNumber=OLD.itemNumber,
description = OLD.description,
supplier = OLD.supplier ,
MSRP = NEW.MSRP,
discountPercent = OLD.discountPercent,
date  = NOW(),
ACTION = 'BEFORE UPDATE';

#### INSERT TRIGGER
Create TRIGGER  account_after_insert

AFTER  INSERT  ON invItems
FOR EACH ROW
INSERT INTO invHistory
set itemNumber= NEW.itemNumber,
description =   NEW.description,
supplier = NEW.supplier ,
MSRP = NEW.MSRP,
discountPercent = NEW.discountPercent,
date  = NOW(),
ACTION = 'AFTER INSERT';
```

```
$$
DELIMITER ;


DELIMITER $$

CREATE TRIGGER account_before__delete
BEFORE DELETE ON invItems FOR EACH ROW

INSERT INTO invHistory
set itemNumber=OLD.itemNumber,
description = OLD.description,
supplier = OLD.supplier ,
MSRP = OLD.MSRP,
discountPercent = OLD.discountPercent,
date  = NOW(),
ACTION = 'BEFORE DELETE';
$$
DELIMITER ;


Insert into invItems(description,supplier,inStock,date,MSRP,discountPercent)
Values('Kellogs Corn Flakes','Whole Food Warehouse','Y','2016-12-24',6.98,0.09);


Insert into invItems(description,supplier,inStock,date,MSRP,discountPercent)
Values('Nabisco Fig Newtons','Albertsons','Y','2016-11-24',4.98,0.08);

Insert into invItems(description,supplier,inStock,date,MSRP,discountPercent)
Values('Nabisco Captain Crunch','Walmart','Y','2016-8-19',6.98,0.06);


UPDATE invItems set MSRP = 1.1 * MSRP;



DELETE FROM invItems
where  MSRP=5.48;


select * from invItems;
select * from invHistory;
```