Lab 1: SQL Practice

Due date: Wednesday, January 17, 11:00am.

MySQL and MySQL accounts

MySQL Accounts

These setup instructions refer to running mysql client from the CSL linux machines, such as the ones found in the database lab, 14-302. They also should work when you are working on one of the CSL servers (`unix1, unix2`, etc.)

Our MySQL server. Our MySQL server is installed on the CSL VM `cslvm74.csc.calpoly.edu`. **You DO NOT HAVE access to the** `csclvm74` server itself, nor do you need this access in order to work with the MySQL server. Instead, you will set up your mysql client to access the course MySQL server.

Our server is a MySQL server, Version 5.5.54.

All of you have accounts on the CSC 365 MySQL server. The accounts are protected by a password. Instructions for modifying the password are found further in this document. Your MySQL account name is the same as your CalPoly loginId.

`mysql` **client.** The `mysql` client prgram is installed on all machines in CSL and on all CSL servers.

```
dekhtyar@csclnx11:~ $ which mysql
/usr/bin/mysql
```

In order to correctly access the CSC 365 MySQL server, you shall type the following

```
$ mysql -h cslvm74.csc.calpoly.edu -p
```

or

```
$ mysql -h cslvm74.csc.calpoly.edu -u <username> -p
```

where `<username>` is your Cal Poly login id.

Next, switch to the database you will be using for this lab:

```
mysql[none]> use ebakery
Database changed
mysql[ebakery]>
```

or

```
mysql[none]> use nyse
Database changed
mysql[nyse]>
```

You are now ready to work on the assignment.

For more information on working with mysql command line client, please see the CSC 365 handouts that can be found on the instructor's web site.

Assignment Preparation

This is an individual lab. Each student has to complete all work required in the lab, and submit all required materials **exactly as specified** in this assignment.

The assignment uses two databases.

The first one is a modified version of one of the datasets I use in CSC 365. The new dataset, `EXTENDED BAKERY` removes the portion of the old `BAKERY` dataset, and replaces it with some new structures. The full description of the dataset is distributed as a separate document.

The second database, called `NYSE` stores the stock market information for the New York Stock Exchange (NYSE). The description of this database follows.

The database contains three tables: `Securities`, `Prices`, and `AdjustedPrices`.

The `Securities` table contains the description of each security (company) traded on NYSE, keyed to the NYSE ticker symbol (such as `GOOG` or `IBM`).

The `Prices` and `AdjustedPrices` tables contain information about the day-to-day prices of the securities. For each day and each security, the tables report the opening and closing prices, the highest and the lowest prices for the day, and the trading volume.

The `Prices` tables contains pricing information in terms of the actual share prices on the day of trading. The `AdjustedPrices` table contains the same information but adjusts the price of securities that underwent stock splits to show the value of the initial share of the security before any stock splits occurred. For securities that never underwent stock splits both tables should record the same data (although I have not checked).

The `CREATE TABLE` statements for these tables are shown below.

```
CREATE TABLE Securities(
  Ticker CHAR(8) PRIMARY KEY,  -- NYSE ticker symbol
  Name   Varchar(40),          -- Name of the company/security
  Sector Varchar(30),          -- Sector in which the company operates
  Industry Varchar(50),        -- Sub Industry in which the company operates
  City   Varchar(36),          -- City where the company is headquarterd
  State  Varchar(15),          -- Either US state where the company is headquartered
  Country Varchar(30),         -- Country where the company is headquartered
  StartDate DATE               -- Date the company was listed for the first time
);


-- Stock Information

CREATE TABLE Prices(
  Day DATE,           -- date for which the price data is reported
  Ticker CHAR(8),     -- ticker for which the price data is reported
  Open FLOAT,         -- price at opening
  Close FLOAT,        -- price at closing
  Low   FLOAT,        -- lowest price of the day
  High  FLOAT,        -- highest price of the day
  Volume FLOAT,       -- volume of trading
  PRIMARY KEY(Ticker, Day),
  FOREIGN KEY(Ticker) REFERENCES Securities(Ticker)
);


CREATE TABLE AdjustedPrices(
  Day DATE,           -- date for which the price data is reported
  Ticker CHAR(8),     -- ticker for which the price data is reported
  Open FLOAT,         -- price at opening
  Close FLOAT,        -- price at closing
  Low   FLOAT,        -- lowest price of the day
  High  FLOAT,        -- highest price of the day
  Volume FLOAT,       -- volume of trading
  PRIMARY KEY(Ticker, Day),
  FOREIGN KEY(Ticker) REFERENCES Securities(Ticker)
);
```

The database was constructed from the NYSE stock market dataset released on Kaggle:

https://www.kaggle.com/dgawlik/nyse

Some modifications were made to the `Securities` table: a number of columns was omitted, and the original address column was split into `City`, `State`, and `Country` columns. Please note, that there may be some outlier data in those columns regarding the country of origin of a non-US company.

The two datasets are available to you in the two MySQL databases, `ebakery` and `nyse`. Each of your accounts has `SELECT` permissions to both of these databases.

SQL queries

For this assignment you will prepare a number of SQL queries for each of the databases.

EBAKERY dataset

Write an SQL script containing SQL statements answering the following information requests. Name your file `lab1-ebakery.sql`.

1. Find all employees for the San Luis Obispo store. Report their names and positions.
2. Report all dates on which the San Luis Obispo store had at least one sale of `Raspberry Lemonade`. Order the dates chronologically, and report each date only once.
3. Report all sales that occurred in the San Luis Obispo store on January 22, 2000. For each sale, report the receipt number and the name of the employee who conducted the sale.
4. List all the least expensive items on the menu. For each type of item list its name (Flavor, Food) and type (drink or pastry).
5. For each Los Angeles location list the total number of recorded sales (receipts) in the database. For each location supply store number, street address and city. Output the results ordered by the total number of sales.
6. For each store with more than three employees report the total sales amount. Report the store number, city and state and the total sales amount. Sort the output by the total sales amount.
7. Find the bakery/bakeries that sold the largest number of `Walnut Cookies`. Report store number, city, state, street address and the number of cookies sold.
8. For each type (the value of `Food` attribute) of pastry (but not drink)[1], report the total amount in sales from California stores. Sort results in descending order by the total sales amount.

[1]Check the values in the `Goods.Type` attribute

9. Find the employee(s) responsible for the largest number of sales. Report their name(s) and location(s) (store number, city, state, street).

10. For each Arizona and Nevada store report the most popular pastries (that is, the pastry/pastries that the store sold more than other pastries). Report the city, state, street address for the store and the flavor and food type for the pastry.

NYSE dataset

Represent the following information needs as SQL queries. For each information need write a single SQL query. Place all your SQL queries and comments in the `lab1-nyse.sql` file.

Note: If a stock ticker is listed next to the name of the security, you can use it in your query. If it is not listed, you have to find the ticker as part of your query.

Note: This is a relatively large database - the `AdjustedPrices` table you will be using for most of the queries has around 851,000 tuples. Therefore, some of your queries may take non-trivial time to process. Most of the queries limit the information to one or few stock tickers, or to a limited time, so MySQL's optimizer should be able to execute them efficiently, but there is no guarantee. Please be aware of possible delays with executing your queries and be careful in crafting them. All your queries should have fairly small outputs.

1. Report the annual change in stock price for Microsoft (MSFT) for year 2016. The change should be measured in terms of the difference between the opening price on January 4 and the closing price on December 30 of that year.

2. Report average daily volume of trade for each month of 2016 for IBM (IBM).

3. Find any days in November 2016 when the price of the Apple (AAPL) stock went up, while the price of Amgen Inc (AMGN) stock went down.

4. Find if there are any securities whose price remained between \$50 and \$55 (i.e. < 55.00) for the entirety of the month of May of 2012. Report the Ticker symbols only in alphabetical order. If there are more than 20 securities, report only the first 20 using the `LIMIT` clause. For this query you are interested in high and low prices for each day, not open and close prices.

5. Find if any securities lost more than 3% of their stock in a single day of trading on August 1, 2016. Report the tickers of any of the securities, and the percent of the stock price that was lost.

6. For each month in 2015 and 2016 (report the months in as e.g. 'February 2016' either as one or two columns) report the security with the highest traded volume.

7. Report the date on which the stock `'Alphabet Inc Class C'` security had the highest closing.

8. Find the non-US security that had the highest relative increase in value in 2015. Report the name of the security, its ticker, the percent increase in the value, and the country of origin. The change in value should be measured as the difference between the open price on the first day of trading in 2015 and the close price on the last day of trading. The first and last days of trading have to be computed properly inside your SQL query (hardcoding these dates in this query is considered cheating).

9. Using both `AdjustedPrices` and `Prices` tables, can you discover for each security (that had it), the date of the first stock split? Report your output in alphabetical order by the stock ticker, and limit the number of tuples in the output to 10 (you can use the `LIMIT` clause here for this purpose). Report the Ticker of the stock, the date of the split, and (this is more free form), the evidence of the split (in the form of whatever information you choose to put in your response).

10. Find the sector of the economy (the `Sector` attribute of the `Securities` table) that appeared to be the healthiest in 2016. This is more of an open-ended query. You should formulate the criteria you are planning on using to judge the health of the sector, translate these criteria into a single SQL query, and make certain the query produces a reasonable answer. In the comments in front of your SQL code for the query please provide the text in English explaining exactly how you are measuring the health of the sector.

Notes

We use the same conventions concerning the use of SQL in this course as I use in CSC 365. In particular:

- Queries must be written using ONLY the information contained in the text description of the information needs. You are not allowed to look up keys in the database and use them - this what joins are for.
- MySQL's `SELECT ... ORDER BY <X> LIMIT 1` construct **CANNOT BE USED** in the outer query implementing a "find objects with minimal/maximal amount of some property" types of queries.
- You may choose between the use of `JOIN` syntax or the use of cartesian product syntax in the `FROM` clauses of your `SELECT` statements.

- Remember that MySQL has `UNION` operator, but does not have `INTERSECTION` and `DIFFERENCE` (or `MINUS`) operators.
- Where possible, privelege the use ANSI SQL rather than MySQL-specific syntax.
- Specifically, if your query contains a `GROUP BY` clause, the use in `SELECT` clause of attributes *not mentioned* in the `GROUP BY` clause (that is - direct use w/o aggregates) is **absolutely prohibited**. Your query will score 0 if you do it regardless of whether it returns correct answer.
- use `AS` for aliasing columns in `SELECT` clause. Use no keywords for aliasing tables in the `FROM` clause.
- use `LIMIT` only where proscribed in the query.

If in doubt regarding the use of a specfic SQL feature – ask me.

Submission Instructions

Submit two files: `lab1-ebakery.sql` and `lab1-nyse.sql`.

Submit the file using `handin` as follows:

```
handin dekhtyar lab01 <files>
```