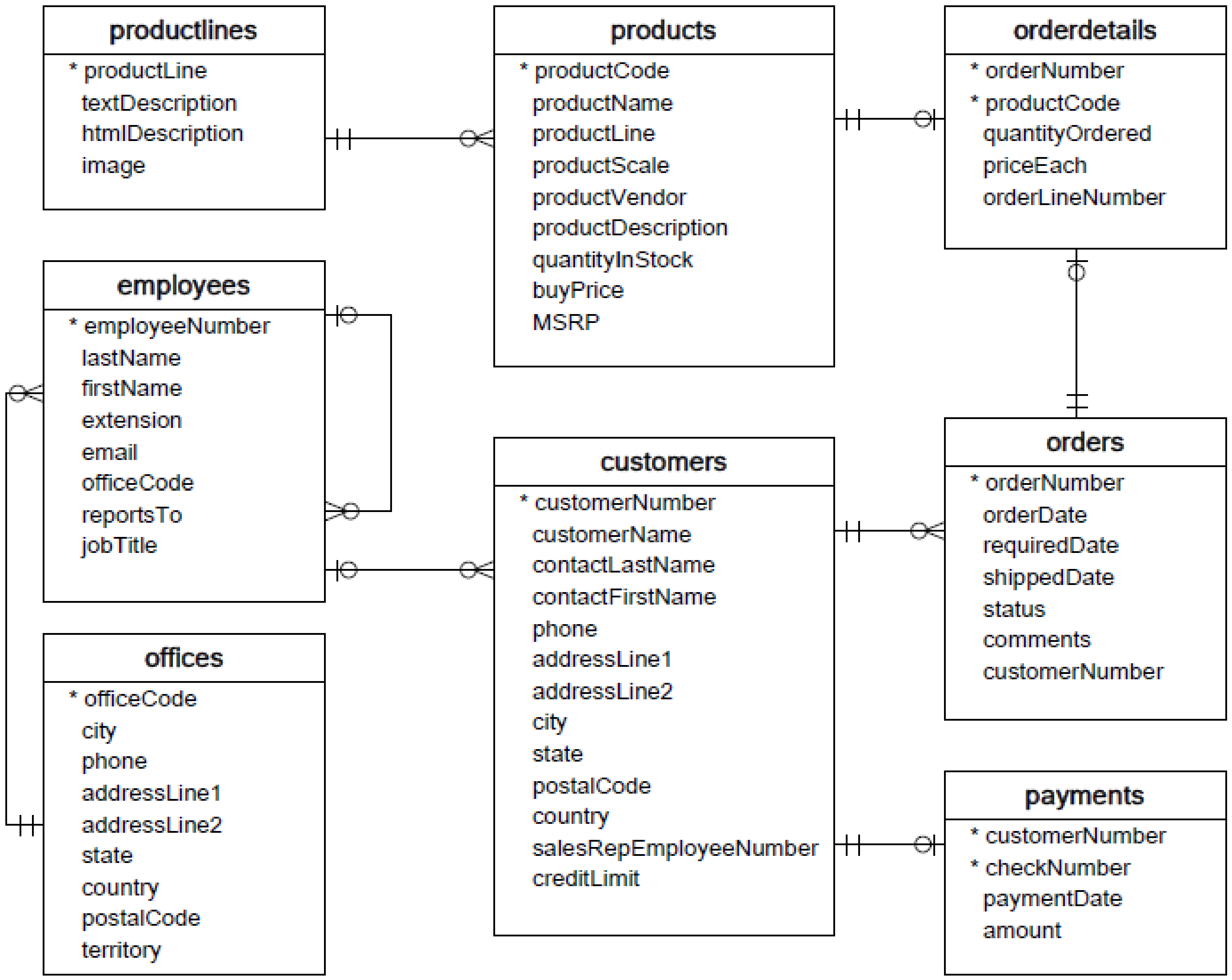
Chapter 5 - SQL

This homework requires to have the "classicmodels" example Mysql database installed.
To get and install the sample database: http://www.mysqltutorial.org/how-to-load-sample-database-into-mysql-database-server.aspx

ERD of ClassicModels



SQL Queries

Note: All SQLs are developed with MYSQL, compatibilities with other RDBMS is not guaranteed.

Implement SQL queries and get the result sets of the following requirements.

A report that has for each office postalCode, a sum of the number of payments

```sql
-- Result set row count: 7
SELECT
    o.postalCode,
    SUM(IFNULL(p.amount, 0)) AS paymentSubtotal
FROM offices o
LEFT JOIN employees e ON o.officeCode=e.officeCode
LEFT JOIN customers c ON e.employeeNumber=c.salesRepEmployeeNumber
LEFT JOIN payments p ON c.customerNumber=p.customerNumber
GROUP BY o.postalCode;
```

A report of customer postalCode sorted by orders(determined by count of the number of items ordered). Sort by orders descending

```sql
-- Result set row count: 95
SELECT
    IFNULL(c.postalCode, 'NOT AVAILABLE') AS postalCode,
    SUM(IFNULL(od.quantityOrdered, 0)) AS quantityOrderedSubtotal
FROM customers c
LEFT JOIN orders o ON c.customerNumber=o.customerNumber
LEFT JOIN orderdetails od ON o.orderNumber=od.orderNumber
GROUP BY c.postalCode
ORDER BY quantityOrderedSubtotal DESC;
```

A report for each product line give count the number of orders

```sql
-- Result set row count: 7
SELECT
    p.productLine,
    SUM(IFNULL(od.quantityOrdered, 0)) AS quantityOrdered
FROM productlines pl
LEFT JOIN products p ON pl.productLine=p.productLine
LEFT JOIN orderdetails od ON p.productCode=od.productCode
GROUP BY p.productLine
ORDER BY quantityOrdered DESC;
```

A report that has the office code, manager name and count of the number of employees that report to that manager.

Add a comment to the query indicating why the sum of the employees from this query is greater than the number of employees.

```sql
-- Result set row count: 6
-- This query's result exactly matches the data stored in employees.
-- The sum of the employees is 22, the whole table record count is 23,
-- which is correct because the president does not report to any.
SELECT
    e1.officeCode,
    CONCAT(e1.lastName, ', ', e1.firstName) AS managerName,
    COUNT(e2.reportsTo) as reportsFrom
FROM employees e1
JOIN employees e2 ON e1.employeeNumber=e2.reportsTo
GROUP BY e1.officeCode, managerName
ORDER BY e1.officeCode;
```

Find manager first name, sales person first name, customer name, customer phone for customers who ordered products that have less than 1000 in stock. Label the output (example Manager Employee Customer ...)

```sql
-- Result set row count: 90
SELECT DISTINCT
    c.customerName as customer,
    c.phone as customerPhone,
    emp.firstName AS salesRepName,
    man.firstName AS managerName
FROM customers c
JOIN orders o ON c.customerNumber=o.customerNumber
JOIN orderdetails od ON o.orderNumber=od.orderNumber
JOIN products p ON od.productCode=p.productCode
JOIN employees emp ON c.salesRepEmployeeNumber=emp.employeeNumber
JOIN employees man ON emp.reportsTo=man.employeeNumber
WHERE p.quantityInStock < 1000
ORDER BY salesRepName, managerName;
```

Find employee with the most orders(determined by order quantity * price summed up)

```sql
-- Result set row count: 1
SELECT emp.*, topSales.salesAmount
FROM employees emp
JOIN (
    SELECT
        e.employeeNumber,
        SUM((od.quantityOrdered * od.priceEach)) AS salesAmount
    FROM orders o
    JOIN orderdetails od ON o.orderNumber=od.orderNumber
    JOIN customers c ON o.customerNumber=c.customerNumber
    JOIN employees e ON c.salesRepEmployeeNumber=e.employeeNumber
    GROUP BY c.salesRepEmployeeNumber
    ORDER BY salesAmount DESC
    LIMIT 1
) topSales ON emp.employeeNumber=topSales.employeeNumber
```

List employee name(first and last name), customer phone number, count of the # comments on orders made by that customer that have these words in them(“reevaluate”, “cancel”, “concerned”)

```sql
-- Result set row count: 15
SELECT e.lastName, e.firstName, c.phone
FROM employees e
JOIN customers c ON e.employeeNumber=c.salesRepEmployeeNumber
JOIN orders o ON c.customerNumber=o.customerNumber
WHERE
    o.comments IS NOT NULL
AND (o.comments LIKE '%reevaluate%'
OR o.comments LIKE '%cancel%'
OR o.comments LIKE '%concerned%')
GROUP BY e.employeeNumber, c.phone
ORDER BY e.lastName, e.firstName
```

For each sales person, for each customer, for each product ordered by the customer figure the average discount((MSRP-buyPrice)/MSRP)

```sql
-- Result set row count: 2532
SELECT
    e.lastName,
    e.firstName,
    c.customerName,
    p.productName,
    AVG((p.MSRP-p.buyPrice)/p.MSRP) AS discount,
    CONCAT(ROUND(AVG((p.MSRP-p.buyPrice)/p.MSRP)*100, 2), '%') AS discountPercentage
FROM employees e
JOIN customers c ON e.employeeNumber=c.salesRepEmployeeNumber
JOIN orders o ON c.customerNumber=o.customerNumber
JOIN orderdetails od ON o.orderNumber=od.orderNumber
JOIN products p ON od.productCode=p.productCode
GROUP BY
    e.employeeNumber,
    c.customerNumber,
    p.productCode
```

Tools and Reference

Reference

- SQL JOINS: LEFT JOIN vs. LEFT OUTER JOIN

Tools

- HediSQL, A lightweight and handy tool for SQL development: HeidiSQL - MySQL, MSSQL and PostgreSQL made easy

```
---- Jason, 09/28/2016
```