Relational Databases and SQL (Structured Query Language)

A database is an organized collection of structured information, typically stored in the form of tables (rows & columns). Relational databases allow storing and retrieving different kinds of related information e.g. products, customers, and orders for an online shopping site. Structured Query Language or SQL (pronounced "sequel") is the most widely used language for interacting with relational databases, and is an essential skill for Data Science professionals.

The following topics are covered in this tutorial:

- Use cases and design of relational databases and SQL
- Setting up a database locally using MySQL server
- Creating, modifying and deleting databases and database tables
- SQL Data types and constraints (primary key, foreign key)
- CRUD (Create, Read, Update and Delete) operations on tables
- Exporting and importing data from relational databases

Problem Statement

We'll learn about relational database and SQL by working through the following problem (source):
QUESTION: Classic Models Inc. is a manufacturer of small scale models of cars, motorcycles, planes, ships trains etc. Products manufactured by Classic Models are sold in toy & gift stores around the world. Here's a small sample of their products (source):

Classic Models has offices around the world with dozens of employees. The customers of Classic Models are typically toy/gift stores. Each customer has a designated sales representative (an employee of Classic Models) they interact with. Customers typically place orders requesting several products in different quantities and pay for multiple orders at once via cheques.

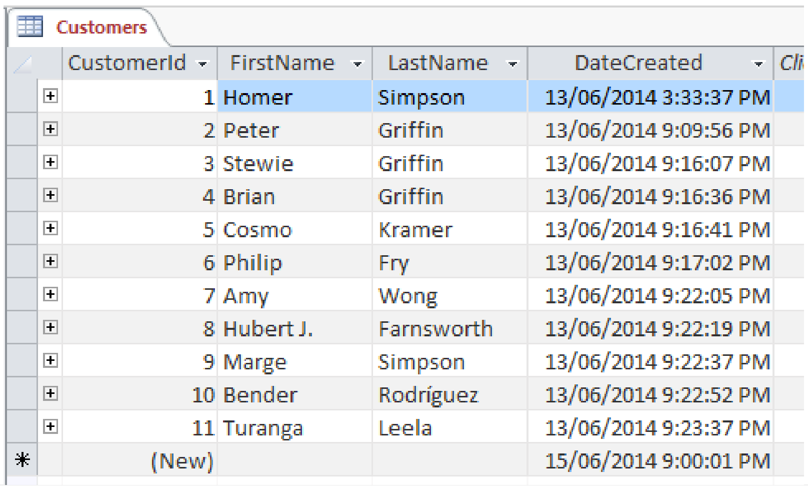
Create a database to record and manage all of the above information. The database will also be used for day-to-day operations (adding customers, placing orders, recording payments, hiring employees etc.).

Relational Databases

There are many ways of storing data on a computer (text files, JSON files, CSV files, spreadsheets etc.). A relational database is a data storage system with the following properties:

1. Data is stored in tables e.g., `customers`, `products`, `offices`, `employees`, etc..
2. Each table has a set of columns. Each column is used to store a specific type of data.
3. Data is stored as rows (also called records) within database tables.
4. Tables support CRUD operations on rows: Create, Read, Update and Delete
5. Table can be connected to other tables using relationship constraints (e.g. an employee works at a specific office).
6. Information can be retrieved from the database using the Structured Query Language (SQL)
7. Databases can be hosted locally (on your computer) or on the cloud, for distributed access.

Here's what a table in a relational database looks like:

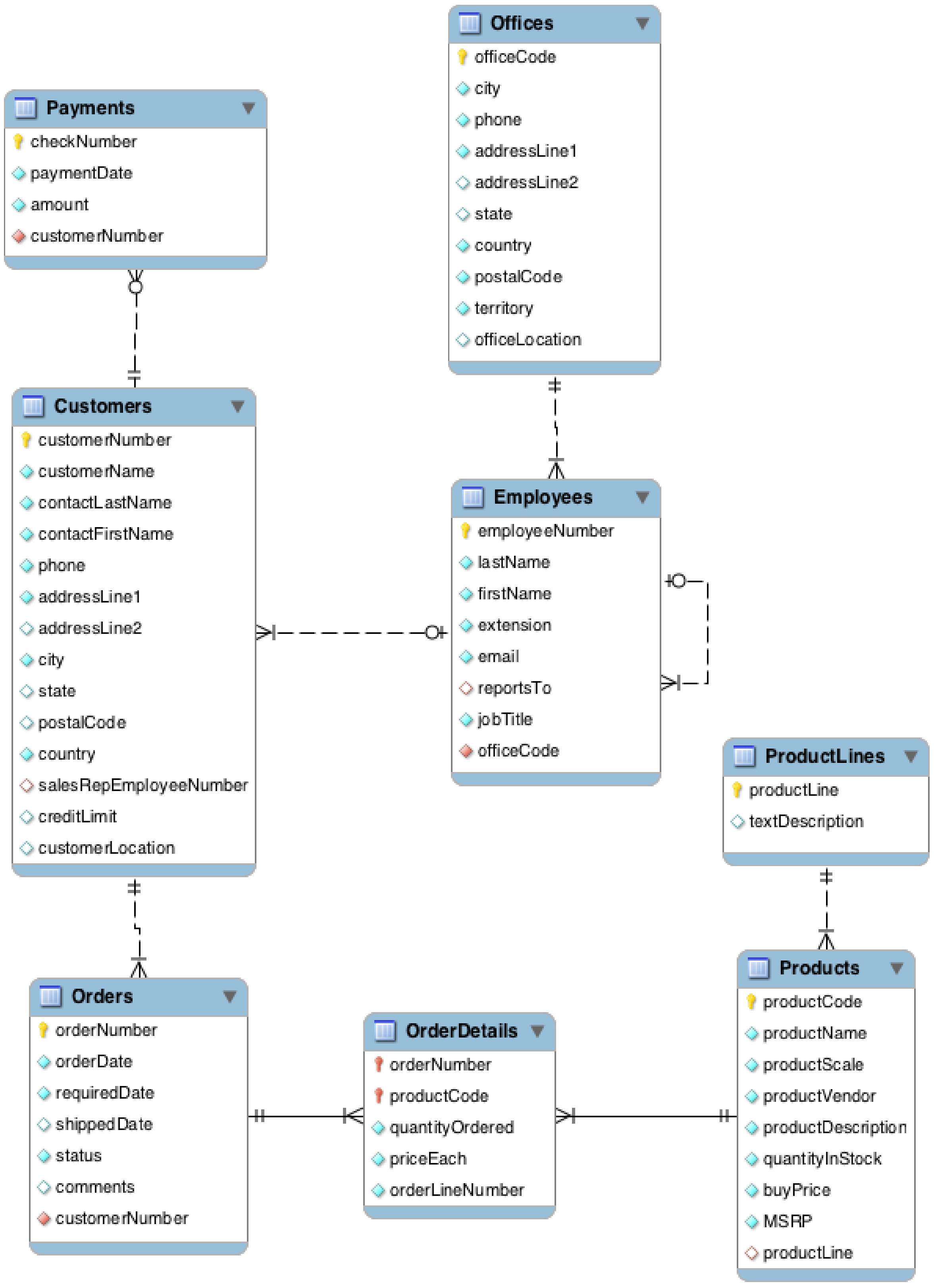


Customers

CustomerId	FirstName	LastName	DateCreated	Cli
1	Homer	Simpson	13/06/2014 3:33:37 PM	
2	Peter	Griffin	13/06/2014 9:09:56 PM	
3	Stewie	Griffin	13/06/2014 9:16:07 PM	
4	Brian	Griffin	13/06/2014 9:16:36 PM	
5	Cosmo	Kramer	13/06/2014 9:16:41 PM	
6	Philip	Fry	13/06/2014 9:17:02 PM	
7	Amy	Wong	13/06/2014 9:22:05 PM	
8	Hubert J.	Farnsworth	13/06/2014 9:22:19 PM	
9	Marge	Simpson	13/06/2014 9:22:37 PM	
10	Bender	Rodríguez	13/06/2014 9:22:52 PM	
11	Turanga	Leela	13/06/2014 9:23:37 PM	
(New)			15/06/2014 9:00:01 PM	

Entity Relationship Diagrams

While setting up a relational database, it's common to create an "Entity Relationship Diagram" (ERD) to describe all the tables within the database and the relations between them. ERDs can be created using drawing tools like LucidChart. Here's the ERD for the Classic Models database:



Relational Database Software

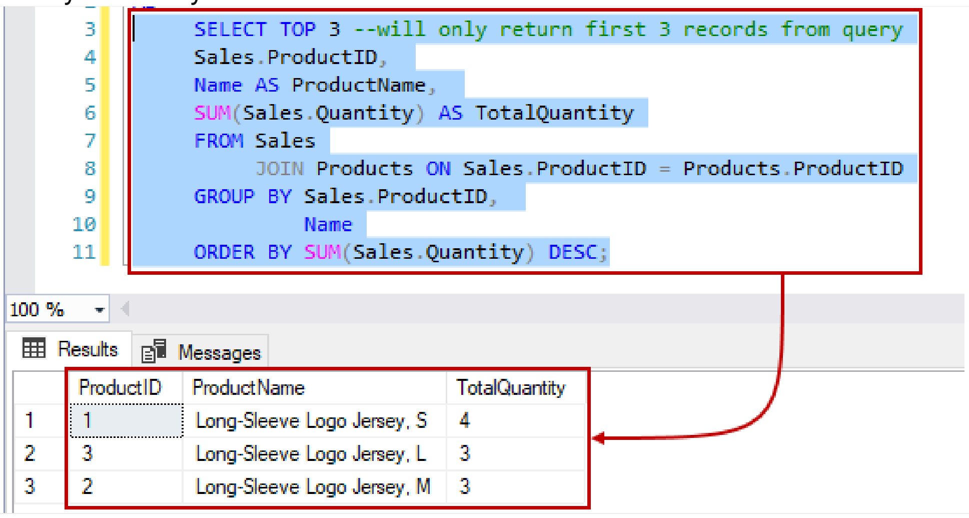
There are several relational database software packages. Some are free and open source, while others are paid and proprietary. Here some popular relational database distributions:

1. MySQL
2. Postgres
3. SQLite
4. Microsoft SQL Server
5. MariaDB
6. Oracle
7. IBM DB2

For this tutorial, we'll use MySQL, a free and open source relational database.

Structured Query Language

Structured Query Language or SQL is a programming language for interacting with relational databases. Unlike general purpose programming languages like Python, Java, C++ etc., SQL has a very limited syntax.



```
SELECT TOP 3 --will only return first 3 records from query
Sales.ProductID,
Name AS ProductName,
SUM(Sales.Quantity) AS TotalQuantity
FROM Sales
    JOIN Products ON Sales.ProductID = Products.ProductID
GROUP BY Sales.ProductID,
         Name
ORDER BY SUM(Sales.Quantity) DESC;
```

	ProductID	ProductName	TotalQuantity
1	1	Long-Sleeve Logo Jersey, S	4
2	3	Long-Sleeve Logo Jersey, L	3
3	2	Long-Sleeve Logo Jersey, M	3

SQL Statements and Syntax

There are three types of SQL statements:

1. Data Definition Language(DDL): The Data Definition Language contains commands that are less frequently used. DDL commands modify the actual structure of a database, rather than the database's contents. example:

- Generating a table
- Modifying a structure of a table (altering)

2. Data Control Language(DCL): The Data Control Language allows you to manipulate and manage user access rights on database objects. It consists of two commands:

- the GRANT command, used to add database permissions for a user,
- and the REVOKE command, used to remove existing permissions.

These two commands form the core of the relational database security model.

3. Data Manipulation Language (DML): Data Manipulation Language contains the subset of SQL commands used most frequently. It is used for searching, inserting, updating, and deleting data that we will see in this notebook.

Some quick notes on the SQL syntax:

- SQL is case insensitive. You can type statements in upper case, lowercase or a mixture of both
- Database names can be typed with or without backquotes i.e. `classicmodels` or ``` `classicmodels` ```
- SQL statement can span over multiple lines and must end with a semicolon `;`
- The statements in this tutorial can be executed on the MySQL command line or the workbench

The SQL syntax for each relational database software package is slightly different. Check the official documentation of your DB for details.

Setting up MySQL Server Locally

Install the following:

- MySQL server: https://dev.mysql.com/downloads/mysql/
- MySQL workbench: https://dev.mysql.com/downloads/workbench/

You'll be asked to set a root password while installing MySQL server.

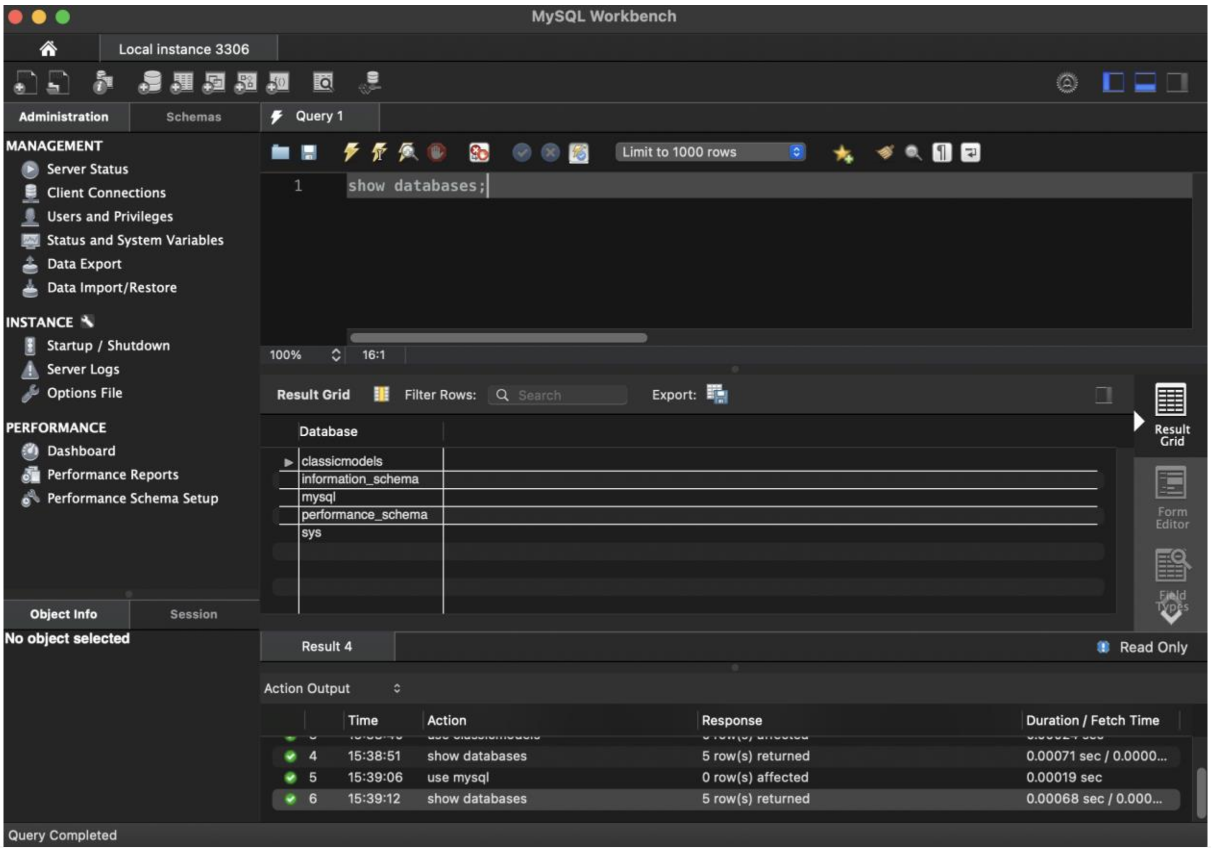
Note: Make sure the password contains only numbers and alphabets. Avoid spaces/special characters in your password.

To interact with the MySQL server via the terminal use:

```
$ /usr/local/mysql/bin/mysql -u root -p
```

Depending on your operating system the path `/usr/local/mysql/bin/mysql` may be different. If you're unable to connect, make sure that the server is running and you're using the correct password.

Alternatively, the MySQL Workbench can be used to interact with a MySQL server (local or remote) via a GUI.



```
show databases;
```

Database
classicmodels
information_schema
mysql
performance_schema
sys

	Time	Action	Response	Duration / Fetch Time
4	15:38:51	show databases	5 row(s) returned	0.00071 sec / 0.0000...
5	15:39:06	use mysql	0 row(s) affected	0.00019 sec
6	15:39:12	show databases	5 row(s) returned	0.00068 sec / 0.000...

Working with Databases

You can find all the SQL statements used in this tutorial in this SQL file.

Let's set up a database for Classic Models. We'll explore SQL statements for listing, creating and deleting databases along the way.

Listing Databases

To display available databases, use the statement:

```
SHOW DATABASES;
```

Output:

```
+--------------------+

| Database           |

+--------------------+

| information_schema |

| mysql              |

| performance_schema |

| sys                |
+--------------------+
4 rows in set (0.01 sec)
```

Creating Databases

Databases can be created using the `CREATE DATABASE` statement.

```
CREATE DATABASE `classicmodels`;
```

Once created, the database should appear in the list of databases

```
SHOW DATABASES;
```

Output:

```
+--------------------+
| Database           |
+--------------------+
| classicmodels      |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)
```

We can also conditionally create a database if it doesn't already exist using:

```
CREATE DATABASE IF NOT EXISTS `classicmodels`;
```

Selecting a Database

A database must be selected before it can be used. This is done with the `USE` statement.

```
USE `classicmodels`;
```

All future statements like listing tables, creating tables, inserting or querying data will use the selected database.

Deleting a Database

To delete a database, use `DROP DATABASE`

```
DROP DATABASE `classicmodels`;
```

The database will no longer show up in the list of databases.

```
SHOW DATABASES;
```

Output:

```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.01 sec)
```

EXERCISE: Create, use and delete some databases using the commands covered in this section. Can you figure out how to rename an existing database?

Let's recreate the database before continuing:

```
CREATE DATABASE IF NOT EXISTS `classicmodels`;

USE `classicmodels`;
```

Working with Tables

Let's create the tables for the Classic Models database according to the given Entity Relationship Diagram. We will also insert some data along the way.

- Tables in a database can be listed using the `SHOW TABLES` statement
- Tables are created using the `CREATE TABLE` statement
- Tables are deleted using the `DROP TABLE` statement
- You can view the structure of a table using the `DESCRIBE` statement
- Data can be inserted into a table using the `INSERT INTO` statement

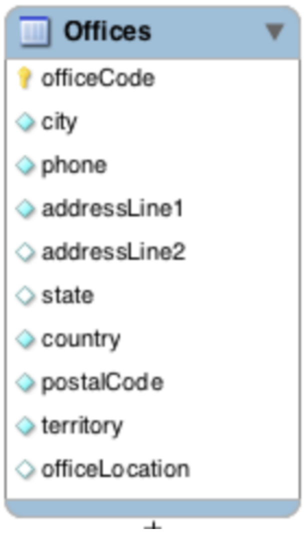
While creating tables, we specify a list of columns and the data type for each column.

```
CREATE TABLE table_name (
    column1 datatype,

    column2 datatype,

    column3 datatype,

   ....

);
```

SQL Data Types

The following data types are supported in most relational databases (source):

Offices



The offices table can be created as follows:

```
DROP TABLE IF EXISTS `offices`;



CREATE TABLE `offices` (

  `officeCode` varchar(10) NOT NULL,

  `city` varchar(50) NOT NULL,

  `phone` varchar(50) NOT NULL,

  `addressLine1` varchar(50) NOT NULL,

  `addressLine2` varchar(50),

  `state` varchar(50),

  `country` varchar(50) NOT NULL,
  `postalCode` varchar(15) NOT NULL,

  `territory` varchar(10) NOT NULL,

  PRIMARY KEY (`officeCode`)

);
```

Primary Key: Note that we've included a `PRIMARY KEY` constraint on the `officeCode` column.

- The PRIMARY KEY constraint uniquely identifies each record in a table.
- Primary keys must contain UNIQUE values, and cannot contain NULL values.
- A table can have only ONE primary key constraint; and in the table, this primary key can consist of single or multiple columns (fields).

We can check that the database is created using `SHOW TABLES;` and view information about it using `DESCRIBE `offices`;`

Inserting Data: We can insert some rows into the table using the `INSERT INTO` statement.

```
INSERT INTO `offices`



(`officeCode`,`city`,`phone`,`addressLine1`,`addressLine2`,`state`,
`country`,`postalCode`,`territory`) VALUES



('1','San Francisco','+1 650 219 4782','100 Market Street','Suite
300','CA','USA','94080','NA'),



('2','Boston','+1 215 837 0825','1550 Court Place','Suite
102','MA','USA','02107','NA'),



('3','NYC','+1 212 555 3000','523 East 53rd Street','apt.
5A','NY','USA','10022','NA'),



('4','Paris','+33 14 723 4404','43 Rue Jouffroy
D\'abbans',NULL,NULL,'France','75017','EMEA'),



('5','Tokyo','+81 33 224 5000','4-1 Kioicho',NULL,'Chiyoda-
Ku','Japan','102-8578','Japan'),
('6','Sydney','+61 2 9264 2451','5-11 Wentworth Avenue','Floor
#2',NULL,'Australia','NSW 2010','APAC'),


('7','London','+44 20 7877 2041','25 Old Broad Street','Level
7',NULL,'UK','EC2N 1HN','EMEA');
```

The `INSERT` statement has the following general syntax:

```
INSERT INTO table_name(column_1, column_2,...) VALUES
(value1_1,value1_2,...), (value2_1,value2_2,...), ...;
```

If you're inserting values for all the columns, you can skip the column names:

```
INSERT INTO table_name VALUES (value1_1, value1_2, ...),
(value2_1,value2_2,...), ...;
```

Viewing Data: The simplest way to view data from the table is using the `SELECT` statement. It has the following syntax:

```
SELECT column1, column2, ... FROM table_name;
```

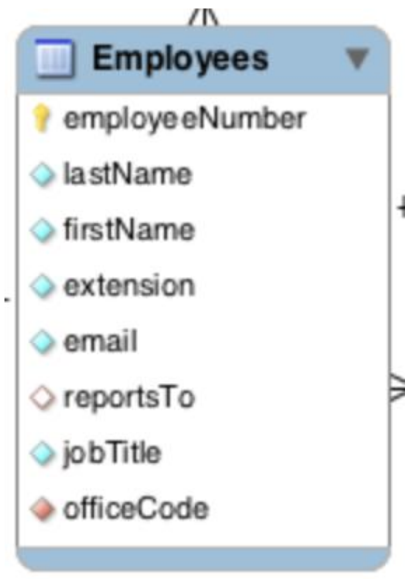
You can also view data from all columns using

```
SELECT * FROM `offices`;
```

EXERCISES:

1. Add some more entires into the `offices` table, using just the required (NOT NULL) columns.
2. Explore what happens if you don't provide a value for a column marked as `not null`.
3. Try adding an entry with a primary key matching the an existing entry.
4. Retrieve and display just the city and phone number information for each office.

Employees



The employees table can be created using the following code:

```
DROP TABLE IF EXISTS `employees`;



CREATE TABLE `employees` (

  `employeeNumber` int(11) NOT NULL,

  `lastName` varchar(50) NOT NULL,

  `firstName` varchar(50) NOT NULL,

  `extension` varchar(10) NOT NULL,

  `email` varchar(100) NOT NULL,

  `officeCode` varchar(10) NOT NULL,

  `reportsTo` int(11) DEFAULT NULL,

  `jobTitle` varchar(50) NOT NULL,

  PRIMARY KEY (`employeeNumber`),

  FOREIGN KEY (`reportsTo`) REFERENCES `employees`
(`employeeNumber`),

  FOREIGN KEY (`officeCode`) REFERENCES `offices` (`officeCode`)

);
```

Foreign Keys: A FOREIGN KEY is a field (or collection of fields) in one table, that refers to the PRIMARY KEY in another table.

1. We've added a foreign key constraint for the the `officeCode` column, connecting it with the `offices` table. This ensure that we can't set an employee's `officeCode` to an invalid value.
2. A foreign key constraint can also be established between a table and itself, as done in the case of the `reportsTo` column.

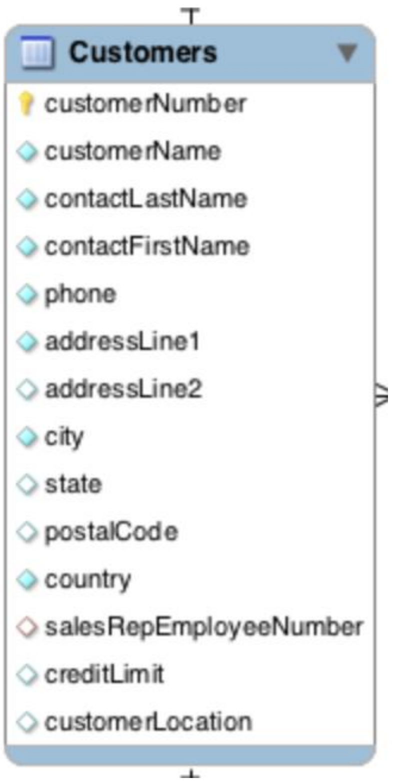
Inserting Data: Let's insert some data into the table:

```
INSERT INTO `employees` VALUES

(1002,'Murphy','Diane','x5800','dmurphy@classicmodelcars.com','1',NULL,'President'),

(1056,'Patterson','Mary','x4611','mpatterso@classicmodelcars.com','1',1002,'VP Sales'),

(1076,'Firrelli','Jeff','x9273','jfirrelli@classicmodelcars.com','1',1002,'VP Marketing'),

(1088,'Patterson','William','x4871','wpatterson@classicmodelcars.com','6',1056,'Sales Manager (APAC)'),

(1102,'Bondur','Gerard','x5408','gbondur@classicmodelcars.com','4',1056,'Sale Manager (EMEA)'),

(1143,'Bow','Anthony','x5428','abow@classicmodelcars.com','1',1056,'Sales Manager (NA)'),

(1165,'Jennings','Leslie','x3291','ljennings@classicmodelcars.com','1',1143,'Sales Rep'),

(1166,'Thompson','Leslie','x4065','lthompson@classicmodelcars.com','1',1143,'Sales Rep'),

(1188,'Firrelli','Julie','x2173','jfirrelli@classicmodelcars.com','2',1143,'Sales Rep'),

(1216,'Patterson','Steve','x4334','spatterson@classicmodelcars.com','2',1143,'Sales Rep'),

(1286,'Tseng','Foon Yue','x2248','ftseng@classicmodelcars.com','3',1143,'Sales Rep'),

(1323,'Vanauf','George','x4102','gvanauf@classicmodelcars.com','3',1143,'Sales Rep'),

(1337,'Bondur','Loui','x6493','lbondur@classicmodelcars.com','4',1102,'Sales Rep'),
(1370,'Hernandez','Gerard','x2028','ghernande@classicmodelcars.com'
,'4',1102,'Sales Rep'),

(1401,'Castillo','Pamela','x2759','pcastillo@classicmodelcars.com',
'4',1102,'Sales Rep'),

(1501,'Bott','Larry','x2311','lbott@classicmodelcars.com','7',1102,
'Sales Rep'),

(1504,'Jones','Barry','x102','bjones@classicmodelcars.com','7',1102
,'Sales Rep'),

(1611,'Fixter','Andy','x101','afixter@classicmodelcars.com','6',108
8,'Sales Rep'),

(1612,'Marsh','Peter','x102','pmarsh@classicmodelcars.com','6',1088
,'Sales Rep'),

(1619,'King','Tom','x103','tking@classicmodelcars.com','6',1088,'Sa
les Rep'),

(1621,'Nishi','Mami','x101','mnishi@classicmodelcars.com','5',1056,
'Sales Rep'),

(1625,'Kato','Yoshimi','x102','ykato@classicmodelcars.com','5',1621
,'Sales Rep'),

(1702,'Gerard','Martin','x2312','mgerard@classicmodelcars.com','4',
1102,'Sales Rep');
```

EXERCISES:

1. Try inserting an entry into `employees` with an invalid value for `officeCode`.
2. Try inserting an entry into `employees` with an invalid value for `reportsTo`.
3. Create a new office location and add some employees for the new location.

Customers



The customers table can be created using the following code:

```
CREATE TABLE IF NOT EXISTS `customers` (

  `customerNumber` INT(11) NOT NULL,

  `customerName` VARCHAR(50) NOT NULL,

  `contactLastName` VARCHAR(50) NOT NULL,

  `contactFirstName` VARCHAR(50) NOT NULL,

  `phone` VARCHAR(50) NOT NULL,

  `addressLine1` VARCHAR(50) NOT NULL,

  `addressLine2` VARCHAR(50) NULL DEFAULT NULL,

  `city` VARCHAR(50) NOT NULL,

  `state` VARCHAR(50) NULL DEFAULT NULL,

  `postalCode` VARCHAR(15) NULL DEFAULT NULL,

  `country` VARCHAR(50) NOT NULL,

  `salesRepEmployeeNumber` INT(11) NULL,

  `creditLimit` DOUBLE NULL DEFAULT NULL,
  `customerLocation` POINT NOT NULL,

  PRIMARY KEY (`customerNumber`),

  FOREIGN KEY (`salesRepEmployeeNumber`) REFERENCES `employees`
(`employeeNumber`));
```

EXERCISE: Add some customers to the database using the data from this SQL file: https://raw.githubusercontent.com/harsha547/ClassicModels-Database-Queries/master/database.sql

Querying with `WHERE`

We can add a `WHERE` clause to a `SELECT` statement to select just the rows satisfying the given clause. Here's an example:

```
SELECT * FROM `employees` WHERE `jobTitle`="Sales Rep";
```

Note that values with spaces must be placed within quotes e.g. `"Sales Rep"`.

The `WHERE` clause supports the following operators:

- `=`
- `>`
- `<`
- `>=`
- `<=`
- `<>` or `!=`
- `BETWEEN`
- `LIKE`
- `IN`

Expressions within a `WHERE` clause can be combined using `AND` and `OR`.

```
SELECT column1, column2, ...

FROM table_name

WHERE condition1 AND condition2 AND condition3 ...;
```

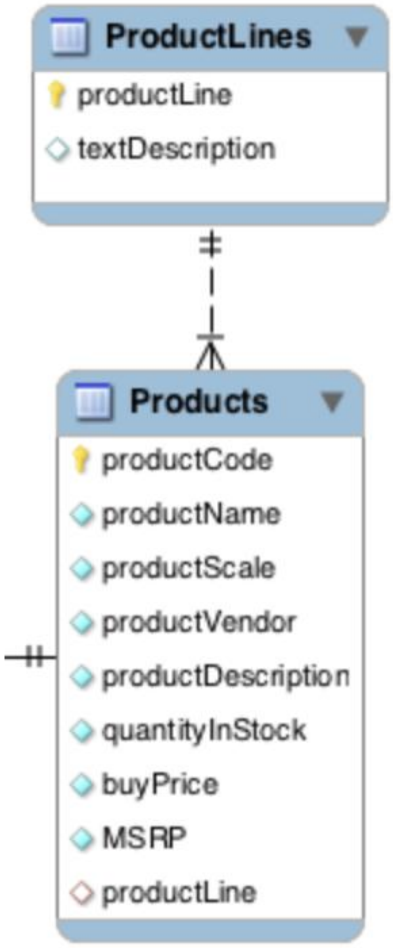
An expression within a `WHERE` clause can be negated using `NOT`.

```
SELECT column1, column2, ...

FROM table_name

WHERE NOT condition;
```

You can use a `WHERE` clause with SELECT, UPDATE, and DELETE statements.
EXERCISES:

1. List the customers in the United States with a credit limit higher than $1000.
2. List the employee codes for sales representatives of customers in Spain, France and Italy. Make another query to list the names and email addresses of those employees.
3. Change the job title "Sales Rep" to "Sales Representative"
4. Delete the entries for Sales Representatives working in London.
5. Show a list of employees who are not sales representatives
6. Show a list of customers with "Toys" in their name

Product Lines and Products



The Product Lines table can be created as follows:

```
DROP TABLE IF EXISTS `productlines`;


CREATE TABLE `productlines` (

  `productLine` varchar(50) NOT NULL,
  `textDescription` varchar(4000) DEFAULT NULL,
  `htmlDescription` mediumtext,
  `image` mediumblob,
  PRIMARY KEY (`productLine`)
);
```

The Products table can be created as follows:

```
DROP TABLE IF EXISTS `products`;


CREATE TABLE `products` (
  `productCode` varchar(15) NOT NULL,
  `productName` varchar(70) NOT NULL,
  `productLine` varchar(50) NOT NULL,
  `productScale` varchar(10) NOT NULL,
  `productVendor` varchar(50) NOT NULL,
  `productDescription` text NOT NULL,
  `quantityInStock` smallint(6) NOT NULL,
  `buyPrice` decimal(10,2) NOT NULL,
  `MSRP` decimal(10,2) NOT NULL,
  PRIMARY KEY (`productCode`),
  CONSTRAINT `products_ibfk_1` FOREIGN KEY (`productLine`)
REFERENCES `productlines` (`productLine`)
);
```

EXERCISE: Add some product lines and products from this SQL file: https://raw.githubusercontent.com/harsha547/ClassicModels-Database-Queries/master/database.sql

`ORDER BY` and `LIMIT`

The results of a Select statement can be ordered using the `ORDER BY` clause:

```
SELECT column1, column2, ...

FROM table_name

ORDER BY column1, column2, ... ASC|DESC;
```

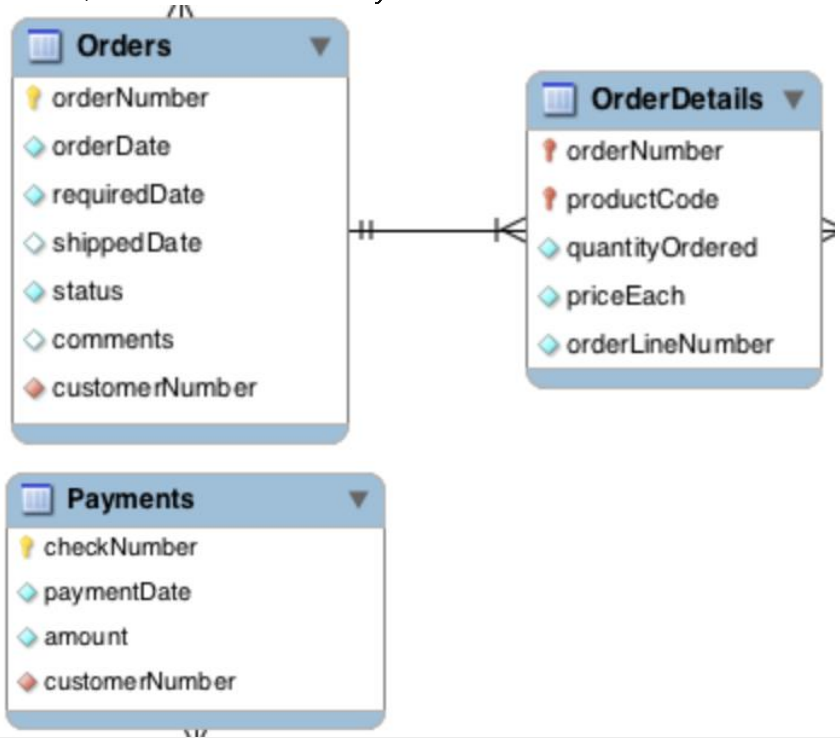
To limit the number of results, use the `LIMIT` clause:

```
SELECT column_name(s)

FROM table_name

WHERE condition

LIMIT number;
```

EXERCISES:

1. List the 5 most expensive products from the "Planes" product line
2. Identify the products that are about to run out of stock (quantity in stock < 100)
3. List 10 products in the "Motorcycles" category with the lowest buy price and more than 1000 units in stock

Orders, Order Details and Payments



EXERCISE: Create the Orders, Order Details and Payments tables using the `CREATE TABLE` statement. Include proper primary key and foreign key constraints. Add some data into each of the tables (sample data).

Modifying Table Structure

The structure of the table can be modified using `ALTER TABLE`

Adding a new column to a table:

```
ALTER TABLE table_name ADD column_name datatype;
```

Removing a column from a table:

```
ALTER TABLE table_name DROP COLUMN column_name;
```

Changing the data type of a column:

```
ALTER TABLE table_name MODIFY COLUMN column_name datatype;
```

Renaming a column:

```
ALTER TABLE table_name RENAME COLUMN old_column_name TO
new_column_name;
```

EXERCISE: Add, remove and modify one column in each of the tables created above. What happens when you rename or remove a column that is a primary key?

Importing and Exporting SQL

Multiple SQL commands can be placed into a `.sql` file (like this one) and can be executed together using the `msql` console or workbench.

```
$ /usr/local/mysql/bin/mysql -u root -p < database.sql
```

This is a great way to import data into a SQL database.

To export a database as a SQL file, use the `mysqldump` utility:

```
$ /usr/local/mysql/mysqldump --add-drop-table -u admin dbname -p >
dbname.sql
```

EXERCISES: Perform the following operations, in the given order:

1. Download this SQL file and import data from your local installation of the MySQL server.
2. Export the `classicmodels` database into a SQL file. Compare it with the originally imported SQL file.
3. Drop the `classicmodels` database and recreate it using the exported SQL file.

SQL Challenges

Set up the Classic Models dataset locally with example data using the SQL script: https://raw.githubusercontent.com/harsha547/ClassicModels-Database-Queries/master/database.sql

1. Prepare a list of offices sorted by country, state, city.

```
???
```

2. How many employees are there in the company?

```
???
```

3. What is the total of payments received?

```
???
```

4. List the product lines that contain 'Cars'.

```
???
```

5. Report total payments for October 28, 2004.

```
???
```

6. Report those payments greater than $100,000.

```
???
```

7. List the products in each product line.

```
???
```

8. How many products in each product line?

```
???
```

9. What is the minimum payment received?

```
???
```

10. List all payments greater than twice the average payment.

```
???
```

11. What is the average percentage markup of the MSRP on buyPrice?

```
???
```

12. How many distinct products does ClassicModels sell?

```
???
```

13. Report the name and city of customers who don't have sales representatives?

```
???
```

14. What are the names of executives with VP or Manager in their title? Use the CONCAT function to combine the employee's first name and last name into a single field for reporting.

```
???
```

15. Which orders have a value greater than $5,000?

```
???
```

Solution

Summary and Further Reading

The following topics were covered in this tutorial:

- Use cases and design of relational databases and SQL
- Installation of MySQL and setting up a database locally
- Creating, modifying and deleting databases and database tables
- SQL Data types and constraints (primary key, foreign key)
- CRUD (Create, Read, Update and Delete) operations on tables
- Exporting and importing data from relational databases

Check out the following resources to learn more:

- ClassicModels Database: https://www.mysqltutorial.org/mysql-sample-database.aspx
- SQL Tutorial: https://www.w3schools.com/sql/default.Asp
- SQL Books & Slides: https://www.db-book.com/slides-dir/index.html
- SQL Exercises: https://www.w3resource.com/sql-exercises/
- MySQL Reference Manual: https://dev.mysql.com/doc/refman/8.0/en/

In the next tutorial, we'll cover the following topics:

- Using functions for advanced querying on strings dates, timestamps etc.
- Aggregating data using COUNT, AVERAGE, SUM, MIN, MAX etc.
- Using joins to combine and query data from multiple tables at once
- Improving query performance with keys, indexes and transactions
- Connecting to SQL database from Python for querying and visualization in Jupyter

```
!pip install jovian --upgrade --quiet
import jovian
jovian.commit()
```

Revision Questions

1. What are Relational databases?
2. What are the properties of relational databases?
3. What is an ERD?
4. What are few relational databases software packages?
5. What is SQL?

6. What are SQL statements?
7. How to list all the available databases?
8. What is the syntax to create a new database?
9. How to select a database to in order to use it?
10. What is the syntax to delete a database?
11. What is the syntax to rename an existing database?
12. What is the syntax to check if a database exists or not?
13. How to view structure of a table?
14. What are the different SQL data types?
15. What is a `PRIMARY KEY`? What are the properties of a `PRIMARY KEY`?
16. What is a `FOREIGN KEY`?
17. What is `REFERENCES` clause in SQL?
18. What is a `CONSTRAINT` in SQL?
19. How to view data in a table?
20. How to add new rows of data in SQL?
21. What operators does `WHERE` clause support?
22. How to combine expressions within a `WHERE` clause?
23. What does `ORDERY BY` clause do in SQL?
24. What does `LIMIT` clause do SQL?
25. How to modify a table structure?
26. How to change data type of a column?
27. How to add new column to the table?
28. How to rename a column in SQL?
29. How to import data into SQL database?
30. How to export data from SQL database?

Solutions for Exercises

EXERCISE: Create, use and delete some databases using the commands covered in this section. Can you figure out how to rename an existing database?

Create

```
CREATE DATABASE `jovian_data`;

SHOW DATABASES;
```

Output:

```
+--------------------+

| Database           |

+--------------------+
| classicmodels      |
| information_schema
| jovian_data
| mysql              |
| performance_schema |
| sys                |
+--------------------+
6 rows in set (0.00 sec)
```

Use

```
USE `jovian_data`
```

Delete

```
DROP DATABASE `jovian_data`
```

To rename the existing database - you can create a new database exactly as the old database with a different name and then drop the old database. Renaming database query has been disabled in new versions of MySQL due to security risks.
EXERCISES:

1. Add some more entires into the `offices` table, using just the required (NOT NULL) columns.
2. Explore what happens if you don't provide a value for a column marked as `not null`.
3. Try adding an entry with a primary key matching the an existing entry.
4. Retrieve and display just the city and phone number information for each office.

1. Add some more entires into the `offices` table, using just the required (NOT NULL) columns.

```
DESCRIBE offices;



INSERT INTO offices (officeCode, city, phone, addressline1,
country, postalCode,territory,officelocation)

VALUES ('jovian', 'Bengaluru', '+919744324587', '34/7, Koramangal',
'India', '560010','South',POINT(1,2));
```

2. Explore what happens if you don't provide a value for a column marked as `not null`.

```
INSERT INTO offices
(city,phone,addressline1,country,postalcode,territory)

VALUES('Hyderabad','+919733345588','32/4, Oldcity',
'India','432201','East');
```

```
Error Code: 1364. Field 'officeCode' doesn't have a default value
```

3. Try adding an entry with a primary key matching the an existing entry.

```
INSERT INTO offices (officeCode, city, phone, addressline1,
country, postalCode,territory,officelocation)

VALUES ('jovian', 'Chennai', '+919744323487', '3/7, Beach',
'India', '560220','South',POINT(3,2));
```

```
Error Code: 1062. Duplicate entry 'jovian' for key 'offices.PRIMARY'
```

4. Retrieve and display just the city and phone number information for each office.

```
SELECT city, phone FROM offices;
```

EXERCISES:

1. Try inserting an entry into `employees` with an invalid value for `officeCode`.
2. Try inserting an entry into `employees` with an invalid value for `reportsTo`.
3. Create a new office location and add some employees for the new location.

1. Try inserting an entry into `employees` with an invalid value for `officeCode`.

```
INSERT INTO employees (employeeNumber, lastname, firstname,
extension, email, reportsTo,jobTitle,officeCode)

VALUES (2, 'Bruce', 'Wayne', 'jovian.ai', 'bruce@jovian.ai',
1002,'CEO',12012);
```

```
Error Code: 1452. Cannot add or update a child row: a foreign key
constraint fails (`classicmodels`.`employees`, CONSTRAINT
`fk_Employees_Offices` FOREIGN KEY (`officeCode`) REFERENCES `offices`
(`officeCode`))
```

2. Try inserting an entry into `employees` with an invalid value for `reportsTo`.

```
INSERT INTO employees (employeeNumber, lastname, firstname,
extension, email, reportsTo,jobTitle,officeCode)

VALUES (2, 'Bruce', 'Wayne', 'jovian.ai', 'bruce@jovian.ai',
1234,'CEO',123);
```

```
-- Error Code: 1452. Cannot add or update a child row: a foreign key
constraint fails (`classicmodels`.`employees`, CONSTRAINT
`fk_Employees_Employees` FOREIGN KEY (`reportsTo`) REFERENCES
`employees` (`employeeNumber`))
```

3. Create a new office location and add some employees for the new location.

```
INSERT INTO offices (officeCode, city, phone, addressline1,
country, postalCode,territory,officelocation)

VALUES (197, 'Helsinki', '+44 20 7000 7000', 'Regents Park, Finland
NW1 4SA', 'Finland', '020','North',POINT(51.5,0.12));

INSERT INTO employees (employeeNumber, lastname, firstname,
extension, email, reportsTo,jobTitle,officeCode)

VALUES (19, 'Ray', 'Palmer', 'jovian.ai', 'ray@jovian.ai',
1002,'Engineer',197);




INSERT INTO employees (employeeNumber, lastname, firstname,
extension, email, reportsTo,jobTitle,officeCode)

VALUES (20, 'Alfred', 'Pennyworth', 'jovian.ai',
'alfred@jovian.ai', 1,'Logistics Head',197);
```

EXERCISES:

1. List the customers in the United States with a credit limit higher than $1000.
2. List the employee codes for sales representatives of customers in Spain, France and Italy. Make another query to list the names and email addresses of those employees.
3. Change the job title "Sales Rep" to "Sales Representative"
4. Delete the entries for Sales Representatives working in London.
5. Show a list of employees who are not sales representatives
6. Show a list of customers with "Toys" in their name

1. List the customers in the United States with a credit limit higher than $1000.

```
SELECT * FROM customers WHERE country='USA' AND creditlimit>1000;
```

2. List the employee codes for sales representatives of customers in Spain, France and Italy. Make another query to list the names and email addresses of those employees.

```
SELECT DISTINCT salesRepEmployeeNumber FROM customers WHERE
country='Spain' OR country='Italy' OR country='France';



SELECT firstname, email FROM employees WHERE employeenumber IN
(1370,1337,1702,1401);
```

3. Change the job title "Sales Rep" to "Sales Representative"

```
SET SQL_SAFE_UPDATES = 0;


UPDATE employees SET jobTitle = 'Sales Representative' WHERE
jobTitle = 'Sales Rep';


SET SQL_SAFE_UPDATES = 1;
```

4. Delete the entries for Sales Representatives working in London.

```
SET foreign_key_checks=0;


DELETE FROM employees WHERE jobTitle = 'Sales Representative' AND
officeCode=7;


SET foreign_key_checks=1;
```

5. Show a list of employees who are not sales representatives

```
SELECT * FROM employees WHERE jobTitle != 'Sales Representative' ;
```

6. Show a list of customers with "Toys" in their name

```
SELECT * FROM customers WHERE customername LIKE '%Toys%';
```

EXERCISES:

1. List the 5 most expensive products from the "Planes" product line
2. Identify the products that are about to run out of stock (quantity in stock < 100)
3. List 10 products in the "Motorcycles" category with the lowest buy price and more than 1000 units in stock

1. List the 5 most expensive products from the "Planes" product line

```
SELECT productname FROM products where productline = 'Planes' ORDER
BY buyprice DESC LIMIT 5;
```

2. Identify the products that are about to run out of stock (quantity in stock < 100)

```
SELECT productname,quantityinstock FROM products WHERE
quantityinstock<100;
```

3. List 10 products in the "Motorcycles" category with the lowest buy price and more than 1000 units in stock

```
SELECT productname,buyprice,quantityinstock,productline FROM
products WHERE productline='Motorcycles' AND quantityinstock>1000
ORDER BY buyprice LIMIT 10;
```

EXERCISE: Add, remove and modify one column in each of the tables created above. What happens when you rename or remove a column that is a primary key?

Add

```
ALTER TABLE customers ADD COLUMN status VARCHAR(15)AFTER
creditLimit ;
```

Remove

```
ALTER TABLE customers DROP COLUMN status;
```

Modify

```
ALTER TABLE customers CHANGE postalcode postalcode_zip CHAR(50);
```

Rename

```
ALTER TABLE customers CHANGE customernumber custnumber CHAR(50);
```

```
Error Code: 3780. Referencing column 'customerNumber' and referenced
column 'custnumber' in foreign key constraint 'fk_Orders_Customers' are
incompatible.
```

Remove

```
ALTER TABLE customers DROP COLUMN customernumber;
```

```
Error Code: 1829. Cannot drop column 'customerNumber': needed in a
foreign key constraint 'fk_Orders_Customers' of table 'Orders'
```