Dexter’s Movie Shop

Structure of a relational database

Retrieving information from a relational database

**SQL queries and relational databases**

Now that you have been able to construct a relational database to store the data required by Dexter’s Movie Shop, Dexter needs you to retrieve information from the database tables.

You will need to open the DBMS file for Dexter’s Movie Shop for this activity.

**SQL syntax**

The following SQL syntax will be covered in this unit:

* SELECT
* FROM
* WHERE
* GROUP BY
* HAVING
* ORDER BY

**WHERE**

Operators (+, -, \*, /, <, >, =) can be used with the WHERE syntax. Also BETWEEN and LIKE can be used to filter results.

**AGGREGATE FUNCTIONS**

SUM, AVG, COUNT, MAX, MIN functions can also be used with the SELECT syntax.

**SQL using multiple tables**

The syntax of a SQL query varies slightly when information is being retrieved from more than one table.

* The table name needs to be added to each field name listed in the SELECT command.

E.g. SELECT tblCustomers.First, tblSales.SalesDate

* Each table needs to be listed in the FROM command.

E.g. FROM tblCustomers, tblSales

* A JOIN statement is required to combine the records from each table. It creates a link between a field in one table and a field in a second table.

E.g. WHERE tblCustomers.CustomerID = tblSales.CustomerID;

**Dexter’s Movie Shop tables**

tblCustomers

|  |  |  |  |
| --- | --- | --- | --- |
| **CustomerID** | **First** | **Last** | **Email** |
| 1 | Grace | Park | grace@qq.com |
| 2 | David | Wu | wud@163.com |
| 3 | Jane | Zhang | jane@yahoo.com |
| 4 | May | Troung | may@yahoo.com.my |
| 5 | Grace | Park | grace@go-myanmar.com |
| 6 | Syed | Shah | ssshad@inbox.com |

tblMovies

|  |  |  |  |
| --- | --- | --- | --- |
| **MovieID** | **Title** | **Genre** | **CostPerCopy$** |
| 1 | Wolf Warriors | Action | $7.00 |
| 2 | Furious 7 | Action | $9.00 |
| 3 | Transformers | Sci-Fi | $7.00 |
| 4 | The Mermaid | Romance | $8.00 |

tblSales

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SalesID** | **SalesDate** | **Copies** | **Cash?** | **CustomerID** | **MovieID** |
| 23 | 01-Dec-17 | 2 | Yes | 1 | 1 |
| 24 | 01-Dec-17 | 1 | Yes | 1 | 2 |
| 25 | 01-Dec-17 | 3 | Yes | 1 | 3 |
| 26 | 01-Dec-17 | 0 | Yes | 1 | 4 |
| 27 | 02-Dec-17 | 0 |  | 2 | 1 |
| 28 | 02-Dec-17 | 2 |  | 2 | 2 |
| 29 | 02-Dec-17 | 3 |  | 2 | 3 |
| 30 | 05-Dec-17 | 4 | Yes | 3 | 4 |
| 31 | 09-Dec-17 | 3 |  | 4 | 1 |
| 32 | 09-Dec-17 | 2 |  | 4 | 2 |
| 33 | 09-Dec-17 | 1 |  | 4 | 3 |
| 34 | 09-Dec-17 | 2 |  | 5 | 4 |
| 35 | 15-Dec-17 | 5 | Yes | 6 | 1 |
| 36 | 15-Dec-17 | 2 | Yes | 6 | 2 |
| 37 | 15-Dec-17 | 2 | Yes | 6 | 3 |

**Required**

Create the following queries to illustrate the different syntax used when retrieving data from more than one table.

**Note:** Save each query as qryA, qryB, qryC, etc.

**A. Retrieve** Title and Genre from the Movies table.

SELECT Title, Genre  
 FROM tblMovies;

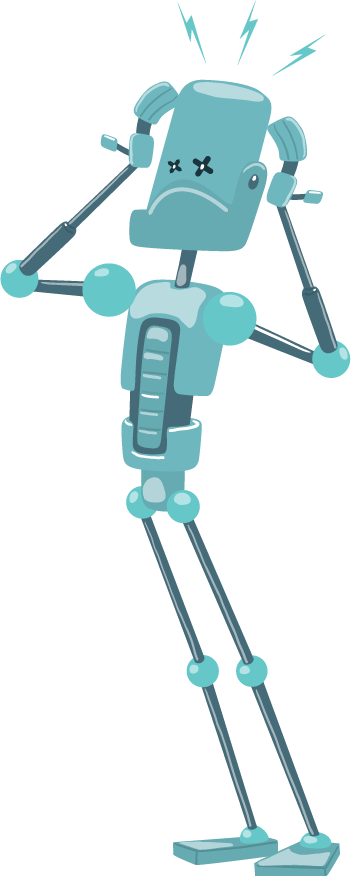
**Note:** This query is only retrieving data from the one table so only the field name is required in the SELECT command.

**B. Retrieve** Title and Genre from the Movies table, and Copies from the Sales table.

SELECT tblMovies.Title, tblMovies.Genre, tblSales.Copies  
FROM tblMovies, tblSales;

**Note:** This query is retrieving data from multiple tables so the table name must be added to each field name in the SELECT command.

All the tables involved must be included in the FROM command.

**Problem**

The query above has returned 60 records. This is because each record in the Movies table (4) has been merged with each record in the Sales table (15). As a result, 4 \* 15 = 60 records have been returned.

The problem is that most of these records are useless. Unrelated data has been merged together in many records producing inaccurate and irrelevant data.

**JOIN**

To stop unrelated data being retrieved a SQL multiple table query requires a JOIN clause.

A JOIN is created between fields in each table that contains common data e.g. MovieID.

There are a number of types of joins, but we will focus on creating a JOIN using the WHERE command.

Create the query again, this time adding the line of code for WHERE.

**C. Retrieve** Title and Genre from the Movies table, and Copies from the Sales table.

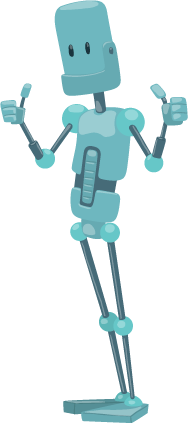
SELECT tblMovies.Title, tblMovies.Genre, tblSales.Copies  
FROM tblMovies, tblSales  
WHERE tblMovies.MovieID = tblSales.MovieID;

**Note:** The code in the WHERE command is creating a join. A JOIN is created between MovieID’s. This ensures that all records returned contain data that is related.

The query should now return 15 records. The number of copies involved in each sale has been combined with the title and genre of the movie.

Let’s now create a query that retrieves data from all three tables.

**D. Retrieve** customer first and last name from the customer table, title from the movies table and copies from the Sales table.

SELECT tblCustomers.First, tblCustomers.Last, tblMovies.Title, tblSales.Copies  
FROM tblCustomers, tblMovies, tblSales  
WHERE tblCustomers.CustomerID = tblSales.CustomerID AND tblMovies.MovieID = tblSales.MovieID;

**Note:** Two JOIN’s are required, one between the customer and sales table and one between the movies and sales table.

The query returns 15 records, showing first name, last name, title and copies sold in each sale.

Now you have the hang of it, create SQL queries to retrieve the following information.

**E. Retrieve** customer first and last name from the customer table, title from the movies table and copies from the Sales table, where copies sold are 2 or more.

**F. Retrieve** customer first and last name, movie title and cost, sales date and number of copies.

**G. Retrieve** customer first and last name, movie title and cost, sales date and number of copies and calculate the total cost of each order.

**H. Retrieve** customer first and last name, movie title and cost, sales date and number of copies and calculate the total cost of each order, where the customer has paid in cash.

**I. Retrieve** movie title and SUM the copies of each sale as TotalCopies, then group by title, only showing the titles that have greater than 4 TotalCopies.

**J. Retrieve** genre and SUM the copies of each sale as TotalCopies, group by genre, then sort by genre in reverse alphabetical order.

**K.** **Retrieve** movie title and SUM the copies of each sale as TotalCopies, then group by title, showing only titles that have sold 8 or more copies.

