Retrieving Data Study Notes

SELECT

A query is simply a question or request to the database.

SQL statements should always end with a semicolon, even though they may run without them.

The terms 'Records' and 'Rows' are often used interchangeably.

The terms 'Fields' and 'Columns' are often used interchangeable.

Text values should be enclosed in single or double quotes.

Numeric values for numeric data type fields should not be enclosed in quotes, even though they will work with quotes in MySQL.

How do I Switch to the Right Database?

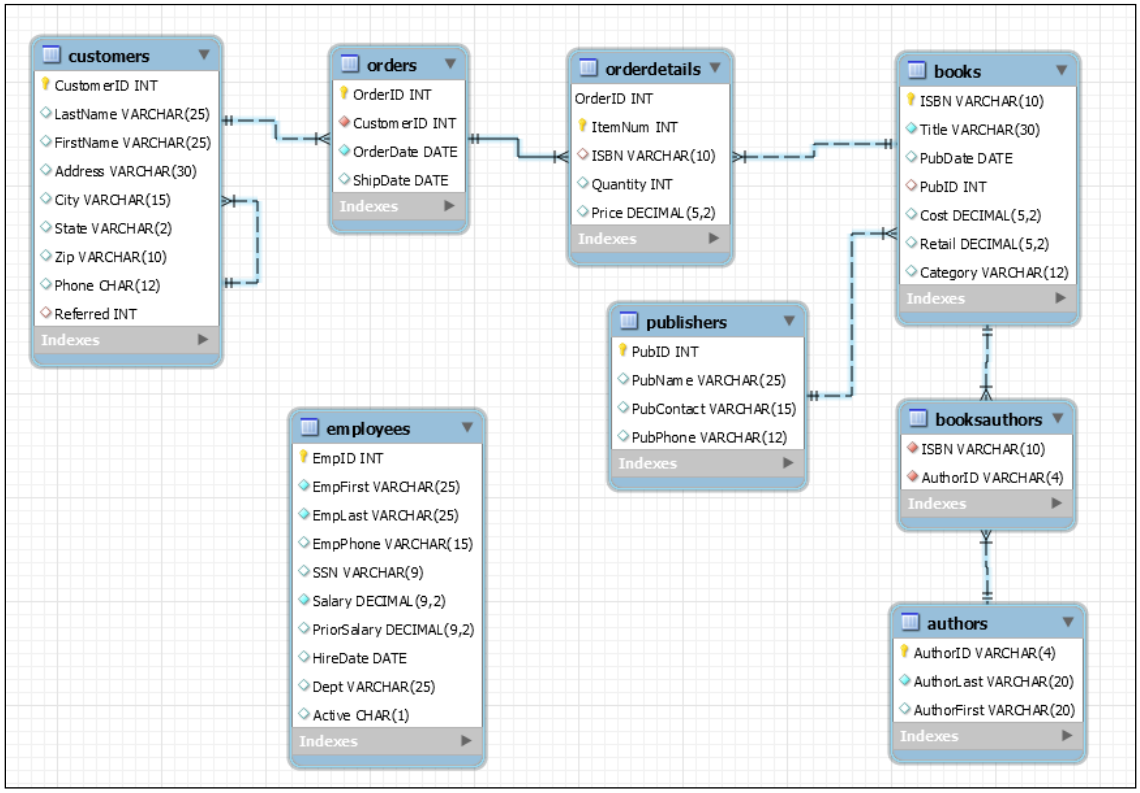
To let MySQL know what database to work with, execute the USE statement. This statement changes the current database to the one specified. It remains the current database until another USE statement is executed or the session is closed.

Practice:

```
USE BookLane;
```

Know Your Database

It's important to know the tables and fields in your database so you can work more effectively and efficiently. Here's an EER diagram of the tables and fields in the BookLane database. Notice the field names use upper camel case, meaning each word in the name begins with uppercase.



Does Capitalization and Indentions Matter?

Upper/lower case does not matter to MySQL, but it does matter to *good* developers. Using a capitalization standard helps with readability and helps spot errors which saves time and money. Many developers put keywords and clauses in uppercase, and other words in lowercase or mixed case. The most important thing is to be consistent!

For this class, please use uppercase for all keywords and clauses. Use upper camel case for field names as indicated in the EER diagram shown earlier in this document.

Note: The BookLane database was created using camel case for table names, but MySQL shows them as all lowercase. The BookLane database also uses camel case for column names as seen in the EER diagram above.

MySQL recognizes the end of a statement when it sees the semicolon, not the end of a line. Frequently a statement spans across multiple lines, with each clause indented on a new line for readability.

Example:

```
SELECT *
   FROM Customers;
```

How do I Retrieve Data?

The SQL SELECT Statement is used to retrieve data from tables.

The syntax of an SQL statement provides the basic structure, or rules, for a command. The SELECT statement's syntax looks pretty complicated, but it's actually not that difficult; It just has a lot of variations.

Refer to the syntax of the SELECT Statement shown here.

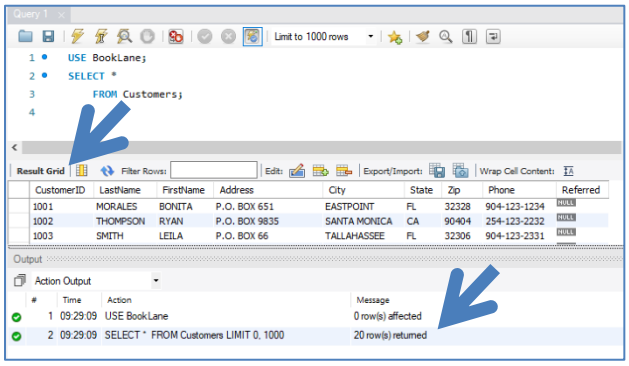
```
SELECT [DISTINCT | UNIQUE]  (*, columnname [ AS alias], …)
        FROM        tablename
        [WHERE      condition]
        [GROUP BY   group_by_expression]
        [HAVING     group_condition]
        [ORDER BY   columnname];
```

- Optional clauses and keywords are shown in square brackets
- SELECT and FROM clauses are required… notice they are not in brackets
- The SELECT clause is followed by the column names you wish to display (or * if you want all columns)
- The FROM clause is followed by the tablename that the data is coming from
- Each clause begins with a keyword
- As a default, every record (row) in the table will be displayed

Practice:

```
SELECT *
      FROM Customers;
```

After a SELECT statement is executed, the Result Grid will show the records. The Action Output will show the number of rows affected, or the total records returned.



How do I Limit the Fields Retrieved?

If you only want specific fields (columns) in your results, use a select list like this:

Practice:

```
SELECT CustomerID, LastName, FirstName
      FROM Customers;
```

Result Grid | Filter Rows:

CustomerID	LastName	FirstName
1001	MORALES	BONITA
1002	THOMPSON	RYAN
1003	SMITH	LEILA
1004	PIERSON	THOMAS
1005	GIRARD	CINDY
1006	CRUZ	MESHIA
1007	GIANA	TAMMY

How do I Limit the Records Retrieved?

If you only want specific records (rows) in your results, use the WHERE clause.

Practice:

```
SELECT *
   FROM Customers
   WHERE CustomerID = 1004;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

CustomerID	LastName	FirstName	Address	City	State	Zip	Phone	Referred
1004	PIERSON	THOMAS	69821 SOUTH AVENUE	BOISE	ID	83707	862-123-2451	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Practice:

```
SELECT *
   FROM Customers
   WHERE CustomerID < 1004;
```

CustomerID	LastName	FirstName	Address	City	State	Zip	Phone	Referred
1001	MORALES	BONITA	P.O. BOX 651	EASTPOINT	FL	32328	904-123-1234	NULL
1002	THOMPSON	RYAN	P.O. BOX 9835	SANTA MONICA	CA	90404	254-123-2232	NULL
1003	SMITH	LEILA	P.O. BOX 66	TALLAHASSEE	FL	32306	904-123-2331	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Practice:

```
SELECT *
   FROM Customers
   WHERE CustomerID > 1002 AND CustomerID < 1006;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

CustomerID	LastName	FirstName	Address	City	State	Zip	Phone	Referred
1003	SMITH	LEILA	P.O. BOX 66	TALLAHASSEE	FL	32306	904-123-2331	NULL
1004	PIERSON	THOMAS	69821 SOUTH AVENUE	BOISE	ID	83707	862-123-2451	NULL
1005	GIRARD	CINDY	P.O. BOX 851	SEATTLE	WA	98115	325-123-4587	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Useful Operators:

=	Equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
<>	Not equal to
!=	Not equal to
AND	Both conditions must be true
OR	Either condition must be true
NOT	Reverses the condition
BETWEEN	Between inclusive (includes operators)
LIKE	Example: LIKE 'ROM%'
	Will locate data that begins with ROM, doesn't matter what follows.
	% is the wildcard that represents no characters to any length of characters
	_ is the wildcard that represents one and only one character
IN	Example: IN('A', 'B', 'C') Will locate data that has A or B or C.

Practice:

```
SELECT *
   FROM Customers
   WHERE CustomerID BETWEEN 1002 AND 1006;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

CustomerID	LastName	FirstName	Address	City	State	Zip	Phone	Referred
1002	THOMPSON	RYAN	P.O. BOX 9835	SANTA MONICA	CA	90404	254-123-2232	NULL
1003	SMITH	LEILA	P.O. BOX 66	TALLAHASSEE	FL	32306	904-123-2331	NULL
1004	PIERSON	THOMAS	69821 SOUTH AVENUE	BOISE	ID	83707	862-123-2451	NULL
1005	GIRARD	CINDY	P.O. BOX 851	SEATTLE	WA	98115	325-123-4587	NULL
1006	CRUZ	MESHIA	82 DIRT ROAD	ALBANY	NY	12211	254-254-4521	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Notice 1002 and 1006 are also included, not just the numbers 'in between' them. That's because the BETWEEN clause is inclusive in MySQL.

Practice:

```
SELECT *
   FROM Customers
   WHERE State IN('CA', 'WA', 'NY');
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

CustomerID	LastName	FirstName	Address	City	State	Zip	Phone	Referred
1002	THOMPSON	RYAN	P.O. BOX 9835	SANTA MONICA	CA	90404	254-123-2232	NULL
1005	GIRARD	CINDY	P.O. BOX 851	SEATTLE	WA	98115	325-123-4587	NULL
1006	CRUZ	MESHIA	82 DIRT ROAD	ALBANY	NY	12211	254-254-4521	NULL
1009	PEREZ	JORGE	P.O. BOX 8564	BURBANK	CA	91510	805-412-4257	1003
1016	DAUM	MICHELL	9851231 LONG ROAD	BURBANK	CA	91508	856-452-4569	1010
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Practice:

```
SELECT *
   FROM Customers
   WHERE State NOT IN('CA', 'WA', 'NY');
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

CustomerID	LastName	FirstName	Address	City	State	Zip	Phone	Referred
1001	MORALES	BONITA	P.O. BOX 651	EASTPOINT	FL	32328	904-123-1234	NULL
1003	SMITH	LEILA	P.O. BOX 66	TALLAHASSEE	FL	32306	904-123-2331	NULL
1004	PIERSON	THOMAS	69821 SOUTH AVENUE	BOISE	ID	83707	862-123-2451	NULL
1007	GIANA	TAMMY	9153 MAIN STREET	AUSTIN	TX	78710	862-542-7458	1003
1008	JONES	KENNETH	P.O. BOX 137	CHEYENNE	WY	82003	325-785-7458	NULL
1010	LUCAS	JAKE	114 EAST SAVANNAH	ATLANTA	GA	30314	325-745-4125	NULL
1011	MCGOVERN	REESE	P.O. BOX 18	CHICAGO	IL	60606	862-252-7854	NULL
1012	MCKENZIE	WILLIAM	P.O. BOX 971	BOSTON	MA	02110	325-547-4582	NULL

Notice the resulting records in this NOT condition are the reverse set of those in the above example without the NOT.

Practice:

```
SELECT *
   FROM Customers
   WHERE LastName LIKE 'MC%';
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Conte

CustomerID	LastName	FirstName	Address	City	State	Zip	Phone	Referred
1011	MCGOVERN	REESE	P.O. BOX 18	CHICAGO	IL	60606	862-252-7854	NULL
1012	MCKENZIE	WILLIAM	P.O. BOX 971	BOSTON	MA	02110	325-547-4582	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

How do I Search for NULL values?

Recall that NULL means there is no value at all. Therefore, you cannot locate NULL using the = operator. You must use the 'IS NULL'. You can use 'IS NOT NULL' to reverse the results.

Practice: View the orders that have not shipped

```
SELECT *
   FROM orders
   WHERE ShipDate IS NULL;
```

Result Grid | Filter Rows:

OrderID	CustomerID	OrderDate	ShipDate
1012	1017	2019-04-03	NULL
1015	1020	2019-04-04	NULL
1016	1003	2019-04-04	NULL
1018	1001	2019-04-05	NULL
1019	1018	2019-04-05	NULL
1020	1008	2019-04-19	NULL
NULL	NULL	NULL	NULL

Notice what happens when you forget and use `= NULL` instead. This should create an error message but it doesn't. It looks as though it worked and that there are no orders that haven't shipped.

Practice COMMON problem:

```
SELECT *
   FROM orders
   WHERE ShipDate = NULL;
```

Result Grid | Filter Rows:

OrderID	CustomerID	OrderDate	ShipDate
NULL	NULL	NULL	NULL

How do I Sort the Results?

As a default, MySQL displays the result in the least costly way to retrieve your records. Frequently that's in the order they were entered into the table. Most of the time however, we need our records displayed in a specific order. Use the ORDER BY clause to specify the order the records should be displayed.

Practice:

```
SELECT *
   FROM Customers
   ORDER BY LastName, Firstname;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

CustomerID	LastName	FirstName	Address	City	State	Zip	Phone	Referred
1006	CRUZ	MESHIA	82 DIRT ROAD	ALBANY	NY	12211	254-254-4521	NULL
1016	DAUM	MICHELL	9851231 LONG ROAD	BURBANK	CA	91508	856-452-4569	1010
1020	FALAH	KENNETH	P.O. BOX 335	TRENTON	NJ	08607	201-526-7452	NULL
1007	GIANA	TAMMY	9153 MAIN STREET	AUSTIN	TX	78710	862-542-7458	1003
1005	GIRARD	CINDY	P.O. BOX 851	SEATTLE	WA	98115	325-123-4587	NULL
1008	JONES	KENNETH	P.O. BOX 137	CHEYENNE	WY	82003	325-785-7458	NULL
1014	LEE	JASMINE	P.O. BOX 2947	CODY	WY	82414	606-555-7548	NULL

To order from greatest to least, add DESC (descending) after the column name.

Practice:

```
SELECT *
   FROM Customers
   ORDER BY LastName DESC, Firstname;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

CustomerID	LastName	FirstName	Address	City	State	Zip	Phone	Referred
1002	THOMPSON	RYAN	P.O. BOX 9835	SANTA MONICA	CA	90404	254-123-2232	NULL
1019	SMITH	JENNIFER	P.O. BOX 1151	MORRISTOWN	NJ	07962	908-254-1204	1003
1003	SMITH	LEILA	P.O. BOX 66	TALLAHASSEE	FL	32306	904-123-2331	NULL
1015	SCHELL	STEVE	P.O. BOX 677	MIAMI	FL	33111	609-325-4521	NULL
1004	PIERSON	THOMAS	69821 SOUTH AVENUE	BOISE	ID	83707	862-123-2451	NULL
1009	PEREZ	JORGE	P.O. BOX 8564	BURBANK	CA	91510	805-412-4257	1003
1013	NGUYEN	NICHOLAS	357 WHITE EAGLE AVE.	CLERMONT	FL	34711	325-745-4521	1006
1017	NELSON	BECCA	P.O. BOX 563	KALMAZOO	MI	49006	325-785-7458	NULL

How do I Limit the Results to the First 5 Rows?

To limit the results to the first however many, use the LIMIT clause.

Practice:

```
SELECT *
    FROM Customers
    ORDER BY LastName, Firstname
    LIMIT 5;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

CustomerID	LastName	FirstName	Address	City	State	Zip	Phone	Referred
1006	CRUZ	MESHIA	82 DIRT ROAD	ALBANY	NY	12211	254-254-4521	NULL
1016	DAUM	MICHELL	9851231 LONG ROAD	BURBANK	CA	91508	856-452-4569	1010
1020	FALAH	KENNETH	P.O. BOX 335	TRENTON	NJ	08607	201-526-7452	NULL
1007	GIANA	TAMMY	9153 MAIN STREET	AUSTIN	TX	78710	862-542-7458	1003
1005	GIRARD	CINDY	P.O. BOX 851	SEATTLE	WA	98115	325-123-4587	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

How do I Concatenate?

Use the CONCAT() function to combine text fields and/or literals values. Use single or double quotes around the literal values.

Practice:

```
SELECT CONCAT(LastName, ', ', FirstName)
      FROM Customers;
```

Result Grid | Filter Ro

CONCAT(LastName, ', ', FirstName)
MORALES, BONITA
THOMPSON, RYAN
SMITH, LEILA
PIERSON, THOMAS
GIRARD, CINDY

Practice:

```
SELECT CONCAT('Full Name: ', LastName, ', ',
FirstName)
      FROM Customers;
```

Result Grid | Filter Rows:

CONCAT("Full Name: ", LastName, ", ", FirstName)
Full Name: MORALES, BONITA
Full Name: THOMPSON, RYAN
Full Name: SMITH, LEILA
Full Name: PIERSON, THOMAS
Full Name: GIRARD, CINDY

How do I Change the Column Heading?

Column headings normally display as the field names or the expression used. To specify a different column heading, use a column alias. Notice in the last two examples, the column header for the concatenated column is the actual concatenation expression. To indicate a better header, use an alias as shown below. If an alias name contains spaces or special symbols, the name must be enclosed in quotation marks. The 'AS' is actually optional and can be left out. The statement is more readable if the AS keyword is used, so I recommend you use it.

Practice:

```
SELECT CONCAT(LastName, ', ', FirstName) AS 'Customer Name'
      FROM Customers;

SELECT CustomerID AS ID, LastName AS 'Last Name'
      FROM Customers;
```

Result Grid

Customer Name
MORALES, BONITA
THOMPSON, RYAN
SMITH, LEILA
PIERSON, THOMAS

Result Grid | Filte

ID	Last Name
1001	MORALES
1002	THOMPSON
1003	SMITH
1004	PIERSON
1005	GIRARD

How do I Calculate?

Pay attention to the Order of Operation rules when calculations are included, otherwise you'll get unexpected results.

- Arithmetic operations are executed left to right.
- Multiplication and division are solved first, then addition and subtraction.
- This order can be overridden with parenthesis just as in algebra.

Order of Operation Examples:

11 + 2 * 3 – 2	= 15
11 + 2 * (3 – 2)	= 13
2 + 8 / 2 * 3	= 14
(2 + 8) / 2 * 3	= 15

For the first example, remember that multiplication and division are done first, so 2*3 = 6, add that to 11 which gives us 17, and then subtract 2 from it, giving us 15. Addition and subtraction are on the same level, so we perform these left to right, whichever comes first.

In the second example, parentheses override the order of operation and force the 3-2 to be done first. That's 1, then multiply that by 2, and add that to 11, giving us 13.

In the third example, we have multiplication and division so we do these left to right... 8/2 = 4, *3 = 12, add that to 2 = 14.

And the fourth example, parenthesis forces the 2+8 to be done first, that's 10, divided by 2 = 5, multiplied by 3 = 15.

Practice:

```
SELECT Title, Retail-Cost AS Profit
  FROM Books;
```

Result Grid | Filter Rows:

Title	Profit
HOW TO GET FASTER PIZZA	12.10
THE WOK WAY TO COOK	9.75
REVENGE OF MICKEY	7.80
BODYBUILD IN 10 MINUTES A DAY	12.20
HANDCRANKED COMPUTERS	3.20
SHORTEST POEMS	18.10

How do I Suppress Duplicates?

If you want to list only the different (distinct) values in a table, use the DISTINCT clause. The DISTINCT clause allows you to suppress duplicates from the result set and return only distinct values.

Practice:

```
SELECT DISTINCT State
      FROM Customers
      ORDER BY State;
```

Result Grid

State
CA
FL
GA
ID
IL
MA
MI

Important note: The DISTINCT keyword is applied to all columns in the colomn list

Practice:

```
SELECT DISTINCT State, Zip
      FROM Customers
      ORDER BY State;
```

Result Grid

State	Zip
CA	90404
CA	91508
CA	91510
FL	32306
FL	32328
FL	33111
FL	34711

Again: The DISTINCT keyword is applied to all columns in the colomn list, **even if it doesn't look like it**. Notice the () in this practice makes it look like the DISTINCT only applies to State, but the results show it applies to all columns in the column list.

```
SELECT DISTINCT(State), Zip
      FROM Customers
      ORDER BY State;
```

Result Grid

State	Zip
CA	90404
CA	91508
CA	91510
FL	32306
FL	32328
FL	33111
FL	34711