Database Access Using MySQL

ADAPT PA Intro to SQL

Bryon Gill
Advanced Computing Systems Specialist
Pittsburgh Supercomputing Center

August 10, 2023

What Is MySQL

MySQL is an RDBMS (Relational DataBase Management System)

Free software originating in 1995

Client and server programs

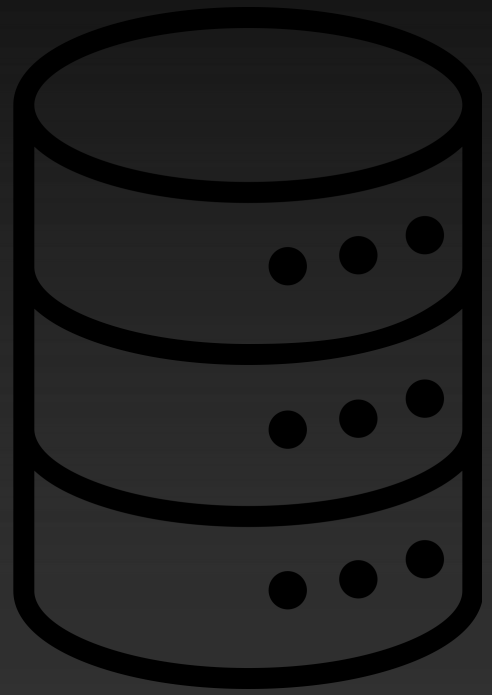
- “mysql” client program connects to mysql-server
- Server stores the data, client asks server for access

Databases

Databases are tablular collections of data

One Database, many tables

"Folder full of spreadsheets"



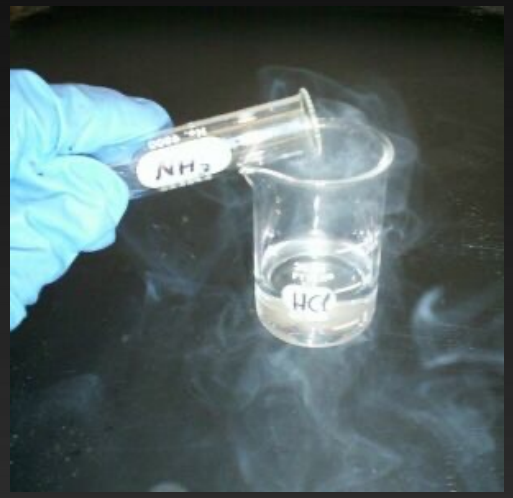
Multiple concurrent users

Schema is a synonym for Database in MySQL
(but not in other RDBMS's)

Databases (Cont'd)

ACID Guarantees

- Atomicity: "All or nothing"
- Consistency: All transactions validated
- Isolation: Transactions executed independently
- Durability: Changes are stored permanently



Example: Bank Transactions

Tables

Collection of related data in columns and rows

Columns:

- Columns have data types (STRING, DATE, VARCHAR, etc.)
- Keys
  - primary
    - (may autoincrement eg. orderId)
    - Can’t be NULL or duplicate
  - foreign
    - refers to primary key in another table
    - orders.orderId->orderdetails.orderId

Rows:

- Data
- Each row is one record

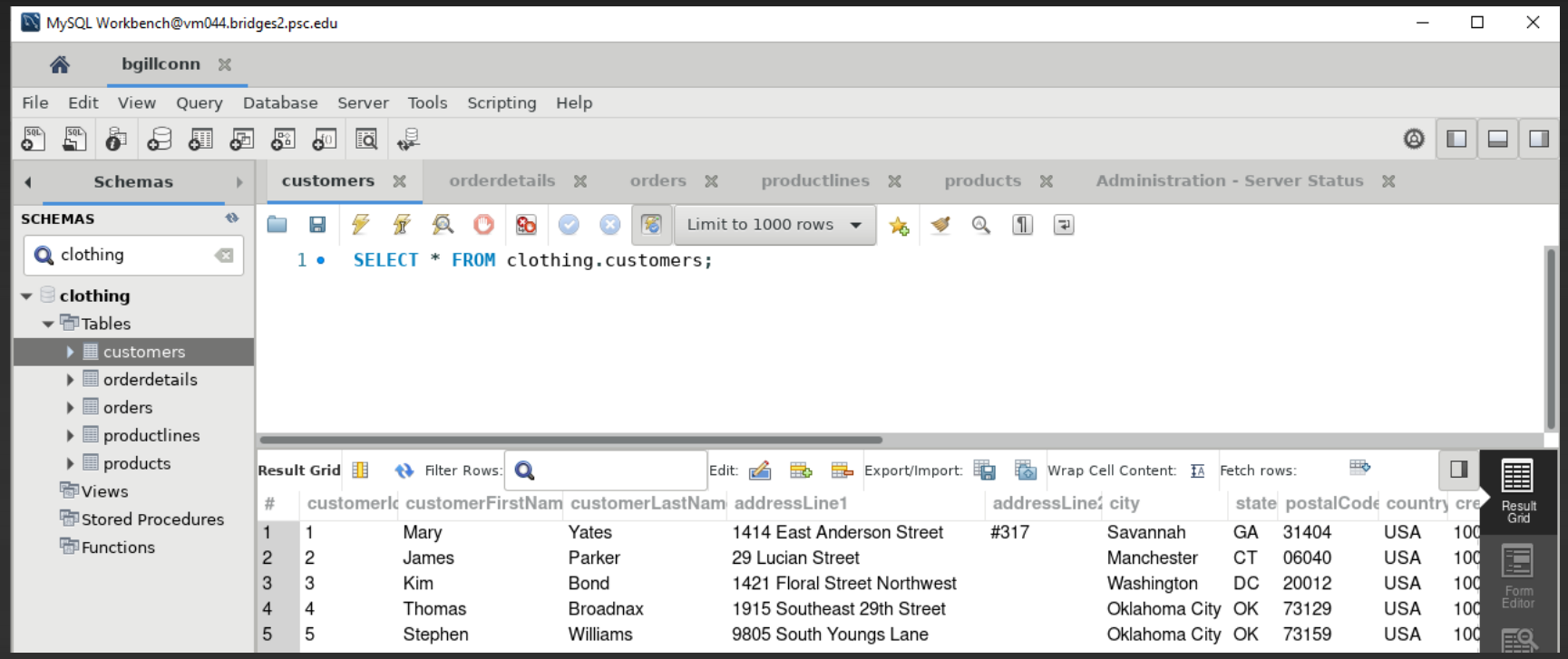
A Clothing Company Database

```
mysql> SHOW TABLES FROM clothing;
+--------------------+
| Tables_in_clothing |
+--------------------+
| customers          |
| orderdetails       |
| orders             |
| productlines       |
| products           |
+--------------------+
5 rows in set (0.00 sec)
```

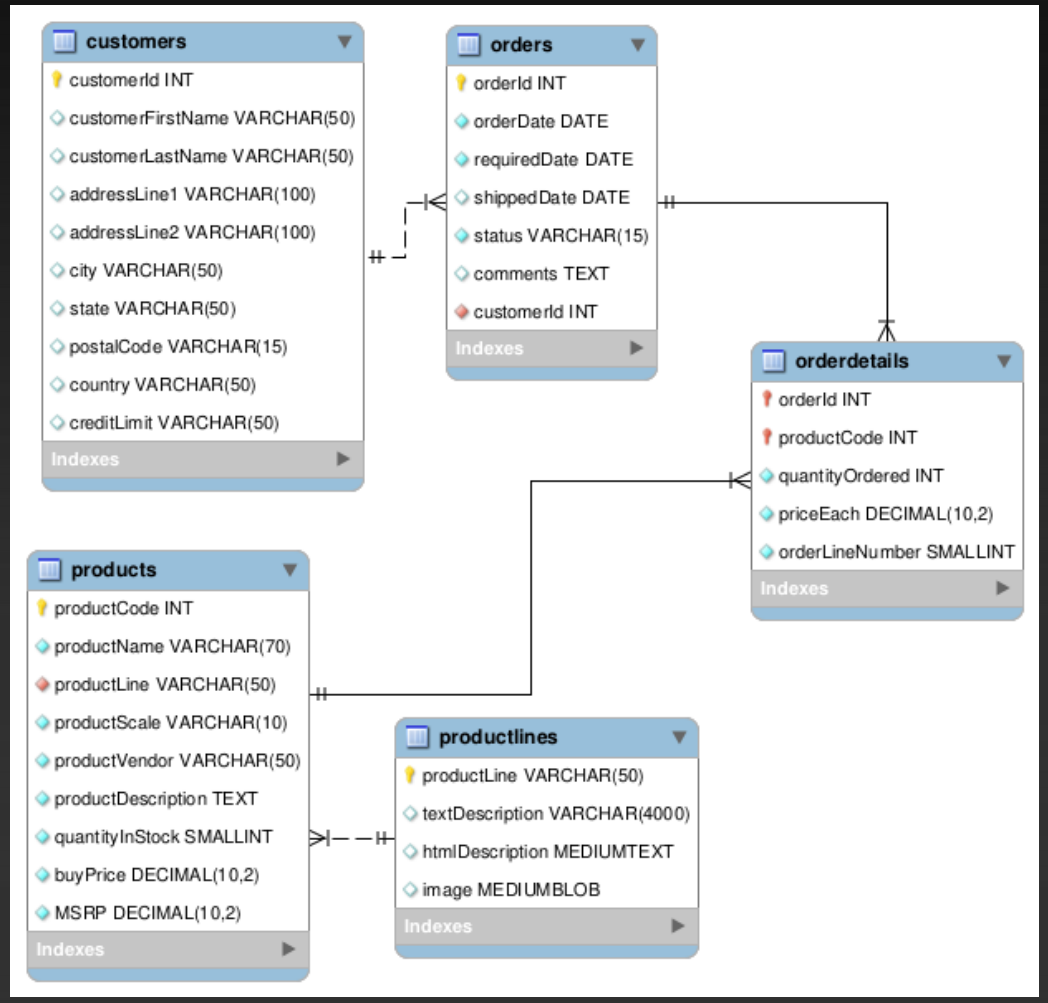
A Clothing Company Database (Cont'd)

Tables hold customer, order, and inventory data.

MySQL Workbench provides a graphical interface:



The Clothing DB Schema



Database Command Basics

SELECT

Retrieve and filter data.

USE,CREATE, ALTER, DROP

Manage SQL tables and databases

INSERT, UPDATE, DELETE

Manage SQL data in tables

SHOW, EXPLAIN

Introspection about database and query structure

USE

Select the database with a USE command:

```
USE clothing;
```

Or just invoke it when you start mysql:

```
(base) [bgill@vm044 ~]$ mysql clothing
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 44
Server version: 8.0.30 Source distribution
```

SELECT

SELECT customerId, customerFirstName,customerLastName FROM customers LIMIT 3;

LIMIT 3 means only print 3 records

```
mysql> SELECT customerId,customerFirstName,customerLastName  FROM customers LIMIT 3;
+------------+-------------------+------------------+
| customerId | customerFirstName | customerLastName |
+------------+-------------------+------------------+
|          1 | Mary              | Yates            |
|          2 | James             | Parker           |
|          3 | Kim               | Bond             |
+------------+-------------------+------------------+
3 rows in set (0.00 sec)
```

PITTSBURGH SUPERCOMPUTING CENTER

SELECT (Cont'd)

SELECT * FROM customers;

** is a wildcard meaning "all".*

this query will get a lot of output!

SELECT * FROM customers WHERE customerId = 1;

we can filter our results with a "WHERE" statement

```
mysql> SELECT * FROM customers WHERE customerId = 1;
+------------+-------------------+------------------+---------------------------+--------------+----------+-------+------------+---------+-------------+
| customerId | customerFirstName | customerLastName | addressLine1              | addressLine2 | city     | state | postalCode | country | creditLimit |
+------------+-------------------+------------------+---------------------------+--------------+----------+-------+------------+---------+-------------+
|          1 | Mary              | Yates            | 1414 East Anderson Street | #317         | Savannah | GA    | 31404      | USA     | 100         |
+------------+-------------------+------------------+---------------------------+--------------+----------+-------+------------+---------+-------------+
1 row in set (0.00 sec)
```

CREATE

Create Tables, Databases, and Views

```
CREATE DATABASE clothing;

CREATE TABLE vendors (
  vendorId int NOT NULL AUTO_INCREMENT, vendorName varchar(100) DEFAULT NULL, addressLine1 varchar(100) DEFAULT NULL,
  addressLine2 varchar(100) DEFAULT NULL, city varchar(50) DEFAULT NULL, state varchar(50) DEFAULT NULL,
  postalCode varchar(15) DEFAULT NULL, country varchar(50) DEFAULT NULL,
PRIMARY KEY (vendorId)
); Query OK, 0 rows affected (0.03 sec)

mysql> show columns from vendors;
+--------------+--------------+------+-----+---------+----------------+
| Field        | Type         | Null | Key | Default | Extra          |
+--------------+--------------+------+-----+---------+----------------+
| vendorId     | int          | NO   | PRI | NULL    | auto_increment |
| vendorName   | varchar(100) | YES  |     | NULL    |                |
| addressLine1 | varchar(100) | YES  |     | NULL    |                |
| addressLine2 | varchar(100) | YES  |     | NULL    |                |
| city         | varchar(50)  | YES  |     | NULL    |                |
| state        | varchar(50)  | YES  |     | NULL    |                |
| postalCode   | varchar(15)  | YES  |     | NULL    |                |
| country      | varchar(50)  | YES  |     | NULL    |                |
+--------------+--------------+------+-----+---------+----------------+
8 rows in set (0.00 sec)
```

ALTER TABLE

```
mysql> ALTER TABLE vendors ADD COLUMN comment VARCHAR(200);
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> show columns from vendors;
+--------------+--------------+------+-----+---------+----------------+
| Field        | Type         | Null | Key | Default | Extra          |
+--------------+--------------+------+-----+---------+----------------+
| vendorId     | int          | NO   | PRI | NULL    | auto_increment |
| vendorName   | varchar(100) | YES  |     | NULL    |                |
| addressLine1 | varchar(100) | YES  |     | NULL    |                |
| addressLine2 | varchar(100) | YES  |     | NULL    |                |
| city         | varchar(50)  | YES  |     | NULL    |                |
| state        | varchar(50)  | YES  |     | NULL    |                |
| postalCode   | varchar(15)  | YES  |     | NULL    |                |
| country      | varchar(50)  | YES  |     | NULL    |                |
| comment      | varchar(200) | YES  |     | NULL    |                |
+--------------+--------------+------+-----+---------+----------------+
9 rows in set (0.00 sec)
```

INSERT

```
mysql> INSERT INTO
vendors(vendorName,addressLine1,addressLine2,city,state,postalCode,country,comment) VALUES
('ThreadBlasters','123 Imaginary Place',NULL,'Sampletown','PA','15217','USA',NULL);
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM vendors;
+----------+---------------+--------------------+-------------+------------+-------+------------+---------+---------+
| vendorId | vendorName    | addressLine1       | addressLine2 | city       | state | postalCode | country | comment |
+----------+---------------+--------------------+-------------+------------+-------+------------+---------+---------+
|        1 | ThreadBlasters | 123 Imaginary Place | NULL         | Sampletown | PA    | 15217      | USA     | NULL    |
+----------+---------------+--------------------+-------------+------------+-------+------------+---------+---------+
1 row in set (0.00 sec)
```

UPDATE

```
mysql> UPDATE vendors SET vendorName = 'ThreadBlazers' WHERE vendorId = 1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM vendors;
+----------+--------------+--------------------+-------------+-----------+-------+------------+---------+---------+
| vendorId | vendorName   | addressLine1       | addressLine2 | city      | state | postalCode | country | comment |
+----------+--------------+--------------------+-------------+-----------+-------+------------+---------+---------+
|        1 | ThreadBlazers | 123 Imaginary Place | NULL        | Sampletown | PA    | 15217      | USA     | NULL    |
+----------+--------------+--------------------+-------------+-----------+-------+------------+---------+---------+
1 row in set (0.00 sec)
```

DELETE

```
mysql> DELETE FROM vendors WHERE city = 'Sampletown';
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM vendors;
Empty set (0.00 sec)
```

DROP

```
mysql> DROP TABLE vendors;
Query OK, 0 rows affected (0.02 sec)

mysql> SELECT * FROM vendors;
ERROR 1146 (42S02): Table ‘clothing.vendors' doesn't exist
mysql>
```

Dates

yyyy-mm-dd (*Default date representation*)

```
mysql> SELECT * FROM orders WHERE orderDate < '2000-01-01' LIMIT 1;
+---------+------------+--------------+-------------+--------+----------+------------+
| orderId | orderDate  | requiredDate | shippedDate | status | comments | customerId |
+---------+------------+--------------+-------------+--------+----------+------------+
|       1 | 1995-05-23 | 1995-06-05   | 1995-06-10  | OK     |          |      88351 |
+---------+------------+--------------+-------------+--------+----------+------------+
1 row in set (0.00 sec)
```

Strings

Strings of characters (text/punctuation/numbers-as-text)

Most commonly uses VARCHAR(n) (eg. varchar(50))

LIKE operator useful for matching with % as a wildcard:

```
mysql> SELECT * FROM customers WHERE customerLastName LIKE 'O%' LIMIT 1;
+------------+-------------------+------------------+------------------------+--------------+-----------+-------+------------+---------+-------------+
| customerId | customerFirstName | customerLastName | addressLine1           | addressLine2 | city      | state | postalCode | country | creditLimit |
+------------+-------------------+------------------+------------------------+--------------+-----------+-------+------------+---------+-------------+
|         95 | Lester            | Obrien           | 7013 Collinswood Drive |              | Nashville | TN    | 37221      | USA     | 100         |
+------------+-------------------+------------------+------------------------+--------------+-----------+-------+------------+---------+-------------+
1 row in set (0.00 sec)
```

(Note: copy/paste from Powerpoint will ruin your single quotes and cause errors!)

Less common: TEXT (long format text), CHAR (fixed length text)

Aggregate Functions

COUNT()

```
SELECT COUNT(*) FROM customers;
```

AVG()

```
SELECT AVG(quantityOrdered) FROM orderdetails;
```

SUM ()

```
SELECT SUM(quantityOrdered) FROM orderdetails;
```

MAX()

```
SELECT MAX(orderDate) FROM orders;
```

MIN()

```
SELECT MIN(orderDate) FROM orders WHERE customerId = 1;
```

Joins

Join data from one table with another

INNER JOIN

- Join the data in two tables on values from a common field
- Returns all rows where the match condition is satisfied
- INNER JOIN is the default in MySQL (just JOIN is equivalent)

Joins (Cont'd)

```
mysql> SELECT customers.customerId, customerFirstName, customerLastName,
orderDate FROM customers INNER JOIN orders ON customers.customerId =
orders.customerId ORDER BY orderDate DESC LIMIT 5;
+------------+-------------------+------------------+------------+
| customerId | customerFirstName | customerLastName | orderDate  |
+------------+-------------------+------------------+------------+
|      75502 | Ricky             | Smith            | 2023-07-17 |
|      10325 | Betty             | Marshall         | 2023-07-17 |
|      93668 | Benjamin          | Price            | 2023-07-17 |
|      49928 | Nelson            | Jenkins          | 2023-07-17 |
|      69622 | Doris             | Ault             | 2023-07-17 |
+------------+-------------------+------------------+------------+
5 rows in set (0.61 sec)
```

LEFT AND RIGHT JOINS

All of one table and part of another. Missing values will be NULL:

LEFT JOIN

All the Left, and the matching portion of the right

RIGHT JOIN

All the Right, and the matching portion of the left

Cross Joins

CROSS JOIN

Everything x Everything All At Once

Also known as a Cartesian product

Subqueries

Query a query

Maybe Query a query that queries a query and so on…

```
SELECT MAX(quantityOrdered) FROM
        (SELECT * FROM orderdetails ORDER BY priceEach DESC LIMIT 25) AS smallorders;
```

Useful for chaining queries for precise filtering:

```
SELECT MAX(quantityOrdered) FROM orderdetails WHERE priceEach =
        (SELECT MIN(priceEach) FROM orderdetails);
```

Subqueries (Cont'd) and Comparisons

Comparison operators:

IN, NOT IN, BETWEEN, IS, IS NOT, IS NOT NULL

Query a subgroup

```
mysql> SELECT * FROM customers WHERE customerId NOT IN (SELECT customerId FROM orders WHERE orderDate < '2000-01-01') LIMIT 5;
+------------+-------------------+-----------------+-----------------------------+-------------+---------------+-------+------------+---------+-------------+
| customerId | customerFirstName | customerLastName | addressLine1                | addressLine2 | city          | state | postalCode | country | creditLimit |
+------------+-------------------+-----------------+-----------------------------+-------------+---------------+-------+------------+---------+-------------+
|          1 | Mary              | Yates           | 1414 East Anderson Street   | #317         | Savannah      | GA    | 31404      | USA     | 100         |
|          2 | James             | Parker          | 29 Lucian Street            |              | Manchester    | CT    | 06040      | USA     | 100         |
|          4 | Thomas            | Broadnax        | 1915 Southeast 29th Street  |              | Oklahoma City | OK    | 73129      | USA     | 100         |
|          6 | Mark              | Hinton          | 8642 Yule Street            |              | Arvada        | CO    | 80007      | USA     | 100         |
|         13 | Christi           | Bailey          | 32532 Jean Drive            |              | Union City    | CA    | 94587      | USA     | 100         |
+------------+-------------------+-----------------+-----------------------------+-------------+---------------+-------+------------+---------+-------------+
5 rows in set (0.42 sec)
```

A More Complex Query using GROUP BY:

The GROUP BY statement groups like rows for aggregation.

Let's reorder our customer list by the date of their first order. To do this we'll use an alias (via the AS keyword) to identify each customer's earliest order date as firstorder:

```
mysql> SELECT MIN(orderDate) AS firstorder, customers.*
           FROM orders INNER JOIN customers ON orders.customerId = customers.customerId
           GROUP BY orders.customerId ORDER BY firstorder LIMIT 10;
+------------+------------+-------------------+------------------+--------------------------------------+--------------+--------------+-------+------------+---------+-------------+
| firstorder | customerId | customerFirstName | customerLastName | addressLine1                         | addressLine2 | city         | state | postalCode | country | creditLimit |
+------------+------------+-------------------+------------------+--------------------------------------+--------------+--------------+-------+------------+---------+-------------+
| 1995-05-23 |       2851 | Rocky             | Fairhurst        | 1815 Grove Hill Lane                 |              | Montgomery   | AL    | 36106      | USA     | 100         |
| 1995-05-23 |       2707 | Sarah             | Browder          | 2552 Massachusetts Avenue Northwest  |              | Washington   | DC    | 20008      | USA     | 100         |
| 1995-05-23 |       4490 | Eva               | Williams         | 505 North Washington Avenue          |              | Fayetteville | AR    | 72701      | USA     | 100         |
| 1995-05-23 |      10731 | Alice             | Vargas           | 683 North Wilson Avenue              |              | Fayetteville | AR    | 72701      | USA     | 100         |
| 1995-05-23 |        531 | Kelli             | Watson           | 312 Shepherd Hills Drive             |              | Nashville    | TN    | 37115      | USA     | 100         |
| 1995-05-23 |       2119 | Leonard           | Souliere         | 29 Church Street                     | D            | Easthampton  | MA    | 01027      | USA     | 100         |
| 1995-05-23 |      11182 | Erik              | Clark            | 2 Ambelwood Way                      |              | Savannah     | GA    | 31411      | USA     | 100         |
| 1995-05-23 |       6278 | Tom               | Joslyn           | 10508 Kovats Court                   |              | Louisville   | KY    | 40223      | USA     | 100         |
| 1995-05-23 |       7894 | Jess              | Barth            | 3215 Madsen Street                   |              | Hayward      | CA    | 94541      | USA     | 100         |
| 1995-05-23 |        232 | Shaniqua          | Strickland       | 3504 Mount View Ridge Drive          |              | Nashville    | TN    | 37013      | USA     | 100         |
+------------+------------+-------------------+------------------+--------------------------------------+--------------+--------------+-------+------------+---------+-------------+
10 rows in set (9.01 sec)
```

Union

UNION concatenates query results. Think of it as a vertical join. Here we find the MSRP for a product and the discounted prices for which it was actually sold:

```
mysql> SELECT MSRP FROM products where productCode = 20593 UNION
select priceEach from orderdetails WHERE productCode=20593;
+--------+
| MSRP   |
+--------+
| 145.85 |
|  14.59 |
| 131.26 |
| 116.68 |
|  72.92 |
+--------+
5 rows in set (0.00 sec)
```

CTE Queries

CTE (Common Table Expression) queries are sometimes easier to read than subqueries but generally serve the same purpose.

```
mysql> WITH neworders (orderId) AS (SELECT orderId FROM orders WHERE orderDate >
'2023-01-01') SELECT neworders.orderId, orderdetails.quantityOrdered,
orderdetails.priceEach FROM neworders JOIN orderdetails ON
neworders.orderID=orderdetails.orderId LIMIT 5;
+---------+-----------------+-----------+
| orderId | quantityOrdered | priceEach |
+---------+-----------------+-----------+
|  883527 |               1 |     21.06 |
|  883528 |               1 |     35.91 |
|  883529 |               1 |    121.88 |
|  883530 |               1 |      3.50 |
|  883530 |               1 |    154.66 |
+---------+-----------------+-----------+
5 rows in set (0.38 sec)
```

PITTSBURGH SUPERCOMPUTING CENTER

Recursive CTE Queries

MySQL allows self-referential CTE queries which can do some neat math tricks like this query that computes Fibonacci numbers.

First select statement establishes initial values

Second select computes succeeding values until a condition is met (default limit 1000 iterations)

```
WITH RECURSIVE fibonacci (n, fib_n, next_fib_n) AS
(
  SELECT 1, 0, 1
  UNION ALL
  SELECT n + 1, next_fib_n, fib_n + next_fib_n
    FROM fibonacci WHERE n < 10
)
SELECT * FROM fibonacci;
+------+-------+------------+
| n    | fib_n | next_fib_n |
+------+-------+------------+
|    1 |     0 |          1 |
|    2 |     1 |          1 |
|    3 |     1 |          2 |
|    4 |     2 |          3 |
|    5 |     3 |          5 |
|    6 |     5 |          8 |
|    7 |     8 |         13 |
|    8 |    13 |         21 |
|    9 |    21 |         34 |
|   10 |    34 |         55 |
+------+-------+------------+
10 rows in set (0.00 sec)
```

PITTSBURGH SUPERCOMPUTING CENTER

Introspection: SHOW CREATE DATABASE

```
mysql> SHOW CREATE DATABASE clothing;
+----------+-----------------------------------------------------------
-----------------------------------------------------+
| Database | Create Database
|
+----------+-----------------------------------------------------------
-----------------------------------------------------+
| clothing | CREATE DATABASE `clothing` /*!40100 DEFAULT CHARACTER SET
utf8mb4 COLLATE utf8mb4_0900_ai_ci */ /*!80016 DEFAULT ENCRYPTION='N' */ |
+----------+-----------------------------------------------------------
-----------------------------------------------------+
1 row in set (0.00 sec)
```

PITTSBURGH SUPERCOMPUTING CENTER

Introspection: SHOW TABLES

```
mysql> SHOW TABLES;
+--------------------+
| Tables_in_clothing |
+--------------------+
| customers          |
| orderdetails       |
| orders             |
| productlines       |
| products           |
+--------------------+
5 rows in set (0.00 sec)
```

Introspection: SHOW COLUMNS

```
mysql> SHOW COLUMNS FROM orders;
+--------------+-------------+------+-----+---------+----------------+
| Field        | Type        | Null | Key | Default | Extra          |
+--------------+-------------+------+-----+---------+----------------+
| orderId      | int         | NO   | PRI | NULL    | auto_increment |
| orderDate    | date        | NO   |     | NULL    |                |
| requiredDate | date        | NO   |     | NULL    |                |
| shippedDate  | date        | YES  |     | NULL    |                |
| status       | varchar(15) | NO   |     | NULL    |                |
| comments     | text        | YES  |     | NULL    |                |
| customerId   | int         | NO   | MUL | NULL    |                |
+--------------+-------------+------+-----+---------+----------------+
7 rows in set (0.00 sec)
```

PITTSBURGH SUPERCOMPUTING CENTER

Introspection: SHOW CREATE TABLE

```
mysql> SHOW CREATE TABLE orders;
+--------+-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
---------------------------------------------------------------------------+
| Table  | Create Table
|
+--------+-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
---------------------------------------------------------------------------+
| orders | CREATE TABLE `orders` (
  `orderId` int NOT NULL AUTO_INCREMENT,
  `orderDate` date NOT NULL,
  `requiredDate` date NOT NULL,
  `shippedDate` date DEFAULT NULL,
  `status` varchar(15) NOT NULL,
  `comments` text,
  `customerId` int NOT NULL,
  PRIMARY KEY (`orderId`),
  KEY `customerId` (`customerId`),
  CONSTRAINT `orders_ibfk_1a` FOREIGN KEY (`customerId`) REFERENCES `customers` (`customerId`)
) ENGINE=InnoDB AUTO_INCREMENT=900107 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci |
+--------+-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
---------------------------------------------------------------------------+
1 row in set (0.01 sec)
```

PITTSBURGH SUPERCOMPUTING CENTER

Views

A View is a saved query that can be used in most respects like a virtual table.

```
mysql> CREATE VIEW customerOrders AS SELECT customers.customerId,
customers.customerFirstName, customers.customerLastName, orders.orderId, orders.orderDate
FROM customers JOIN orders ON customers.customerID = orders.customerID;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from customerOrders limit 1;
+------------+-------------------+------------------+---------+------------+
| customerId | customerFirstName | customerLastName | orderId | orderDate  |
+------------+-------------------+------------------+---------+------------+
|      88351 | Lynn              | Reynolds         |       1 | 1995-05-23 |
+------------+-------------------+------------------+---------+------------+
1 row in set (0.00 sec)

mysql> drop view customerOrders;
Query OK, 0 rows affected (0.01 sec)
```

Indexing

An Index is a hint to the database for query optimization. It allows fast lookup (in a B-Tree):

```
ALTER TABLE products ADD INDEX price_index (msrp);
```

Or we could have added it at table creation time:

```
CREATE TABLE `products` (
  `productCode` int NOT NULL AUTO_INCREMENT,
  `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 NOT NULL,
  `buyPrice` decimal(10,2) NOT NULL,
  `MSRP` decimal(10,2) NOT NULL,
  INDEX price_index (MSRP)
  PRIMARY KEY (`productCode`),
  KEY `productLine` (`productLine`),
  CONSTRAINT `products_ibfk_1` FOREIGN KEY (`productLine`) REFERENCES `productlines` (`productLine`)
)
```

Triggers

Triggers cause database actions to happen when certain conditions are met, for instance a new customer being added may trigger another action that adds their name to a list to be periodically checked for duplicates in the customer table:

```
ALTER TABLE customers ADD COLUMN duplicateName BOOLEAN;

CREATE TRIGGER deduplicate
AFTER INSERT ON customers FOR EACH ROW
UPDATE customers SET duplicateName = TRUE
WHERE customerFirstName=NEW.customerFirstName AND
customerLastName=NEW.customerLastName;
```

Procedures

Procedures are stored routines, for instance perhaps we want to query for duplicate customer accounts periodically.

```
DELIMITER $$
CREATE PROCEDURE GetDuplicates()
BEGIN
SELECT * FROM customers WHERE duplicateName = TRUE;
END$$
DELIMITER ;
```

Running the procedure is very simple:

```
CALL GetDuplicates()
```

Connectors

Connectors allow software to connect to databases and interact with data programmatically with more flexibility than the SQL interface can offer. This is extremely common in web applications (such as the LAMP stack). If the clothing database were configured for remote access you could query data like this using the Python MySQL Connector:

```
import datetime
import mysql.connector

cnx = mysql.connector.connect(user='adaptuser', database='clothing')
cursor = cnx.cursor()

query = ("SELECT orderId, orderDate FROM orders "
         "WHERE orderDate BETWEEN %s AND %s")

sale_start = datetime.date(1999, 1, 1)
sale_end = datetime.date(1999, 12, 31)

cursor.execute(query, (sale_start, sale_end))

for (orderId, orderDate) in cursor:
  print("Order {} placed on {:%d %b %Y}".format(
    orderId, orderDate))

cursor.close()
cnx.close()
```

Exporting Results to a File

We can export query results on Bridges2 to a file using INTO OUTFILE into a specific folder where MySQL has permission to write, replacing YOURUSERNAME with your username. You can upload the file from that folder once you've completed the assignment.

```
mysql> SELECT * FROM orders WHERE orderDate < '1997-01-01' LIMIT 100 INTO
OUTFILE '/jet/home/YOURUSERNAME/mysqloutput/MySQLHomework.txt';
```

Note that you'll have to choose a new filename or delete the old file from the folder if you want to reuse the same filename again.

Exercise

You've been tasked with identifying the most active customers who haven't purchased anything in the last few years so that they can be marketed to with a special customized promotion, but their budget limits how many customers they can contact.

Construct a query to retrieve the 10000 customers who made the most purchases in the past but haven't made a purchase since January 1st 2021. You will need to find the customer's name and address, as well as their most recent purchase. Sort the records so that the customers with the most transactions are at the very top of the results so that they get processed first.

For this assignment limit the output of the first 10 records in the results (LIMIT 10). Export the results to a file called MyHomework.txt and download a copy to submit to your instructor.

Extra: For the marketing department's future planning also figure out how many total customers made a purchase before but not after 1/1/2021. Export the result to a file called ExtraCredit.txt and download a copy to submit to your instructor.

Exercise Suggested Solution

```
SELECT * FROM

(SELECT MAX(orderDate) AS lastOrder, COUNT(orderDate) AS numOrders, orders.customerId, MAX(orderId), customerFirstName,
customerLastName, addressLine1, addressLine2, city, state, postalcode

FROM orders INNER JOIN customers ON orders.customerId = customers.customerId

WHERE orders.customerId NOT IN
(SELECT customerId FROM orders WHERE orderDate > '2021-01-01')

GROUP BY customerId ORDER BY numOrders DESC LIMIT 10000) AS lapsedcustomers

ORDER BY numOrders DESC LIMIT 10;
```