Intermediate SQL

FOR DATA ANALYSIS

Agenda

Installing SQLiteStudio

Subqueries, Derived Tables, and Unions

Regular Expressions

Advanced Joins

Window Functions

Programming with SQL (Python, R and Java)

About the Speaker

Thomas Nield

Business consultant for Southwest Airlines in Schedule Initiatives

Author of *Getting Started with SQL* by O'Reilly and *Learning RxJava* by Packt

Teaches a few online trainings at O'Reilly

- *SQL Fundamentals for Data*
- *Advanced SQL for Data Analysis*
- *Reactive Python for Data Science*


thomasnield9727


https://github.com/thomasnield

Setting Up SQLite

DB Browser for SQLite can be downloaded at the official site:

https://sqlitebrowser.org/

You can install DB Browser, or use a portable version where you can download, unzip, and launch the executable.

If you cannot install or download any software, you can use SQLiteOnline.com which is an online-only SQLite browser.

Getting Resource Files

The few resources needed for this class are available on GitHub:

https://github.com/thomasnield/oreilly_advanced_sql_for_data

Unzip the contents to a location of your choice, and note where you put them

Contents include:

- A SQLite database file called **thunderbird_manufacturing.db**
- Class notes with all examples (in three formats)
- A **customer_order.sql** SQL script file to create a CUSTOMER_ORDER table

Section II Exercise

Bring in all fields from CUSTOMER_ORDER, but for each record show the total quantity ordered for that given CUSTOMER_ID and PRODUCT_ID.

Section III Exercise

Find all customers with an address ending in "Blvd" or "St"

INNER JOIN

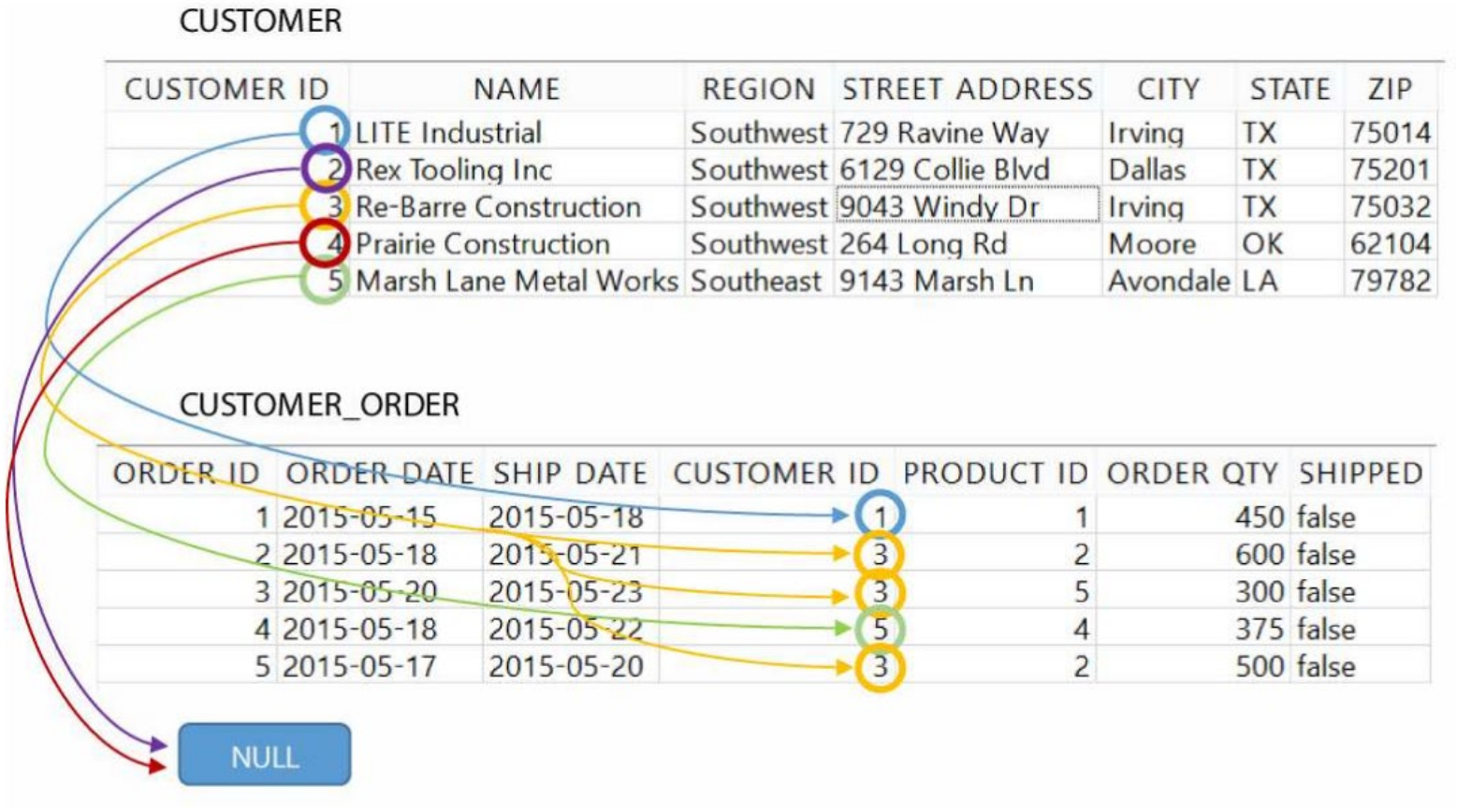
CUSTOMER

CUSTOMER ID	NAME	REGION	STREET ADDRESS	CITY	STATE	ZIP
1	LITE Industrial	Southwest	729 Ravine Way	Irving	TX	75014
2	Rex Tooling Inc	Southwest	6129 Collie Blvd	Dallas	TX	75201
3	Re-Barre Construction	Southwest	9043 Windy Dr	Irving	TX	75032
4	Prairie Construction	Southwest	264 Long Rd	Moore	OK	62104
5	Marsh Lane Metal Works	Southeast	9143 Marsh Ln	Avondale	LA	79782

CUSTOMER_ORDER

ORDER ID	ORDER DATE	SHIP DATE	CUSTOMER ID	PRODUCT ID	ORDER QTY	SHIPPED
1	2015-05-15	2015-05-18	1	1	450	false
2	2015-05-18	2015-05-21	3	2	600	false
3	2015-05-20	2015-05-23	3	5	300	false
4	2015-05-18	2015-05-22	5	4	375	false
5	2015-05-17	2015-05-20	3	2	500	false

LEFT OUTER JOIN



Section VI Exercise

For every CALENDAR_DATE and CUSTOMER_ID, show the total QUANTITY ordered for the date range of 2017-01-01 to 2017-03-31:

Section V Exercise

For the month of March, bring in the rolling sum of QUANTITY ordered (to each ORDER_DATE) by CUSTOMER_ID and PRODUCT_ID.

Windowing Functions Support

Windowing functions are found on many database platforms, including:

- Oracle
- Teradata
- PostgreSQL
- SQL Server
- Apache Spark SQL
- MySQL (as of version 8)
- SQLite (as of version 3.25.0)

These platforms notably do not have windowing functions:

- MySQL (previous to version 8)
- SQLite (previous version 3.25.0)
- MariaDB

Mixing Programming with SQL

When using SQL with a programming platform like Python, Java, or R, you will constantly be making a decision where the onus of processing will happen.

Should the database engine do the computation work, or the programming platform?

- You can simply pull in data and have your Python/Java/R codebase do the heavy-lifting.
- You can also leverage more complex SQL against the database, and have Python/Java/R consume the results.
- With a very large, expensive and calculated dataset you can save it to a temporary table and use it to support your Python/R/Java application.

A good rule of thumb: start with the simplest solution with minimal code/SQL that liberally hits the database as-needed, and gradually introduce caching strategies as performance starts to warrant it.

Never concatenate parameters, and use established SQL libraries to inject parameters safely to prevent SQL injection.

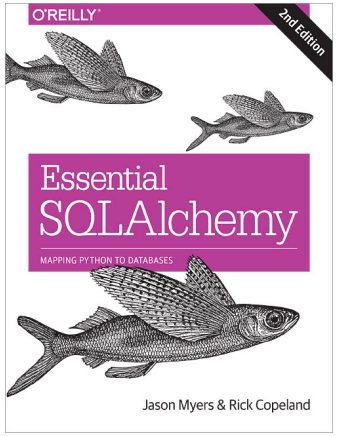
Preventing SQL Injection

To prevent SQL injection, *never* concatenate a SQL string with parameters

Instead, use the right tools and libraries to safely inject parameters for you

For Python, use SQLAlchemy

```
from sqlalchemy import create_engine, text

engine = create_engine('sqlite:///C:\\Users\\thoma\\Dropbox\\rexon_metals.db')
conn = engine.connect()


def customer_for_id(customer_id):
    stmt = text("SELECT * FROM CUSTOMER WHERE CUSTOMER_ID = :id")
    return conn.execute(stmt, id=customer_id).fetchone()


print(customer_for_id(2))
```



More info at:
http://www.sqlalchemy.org/

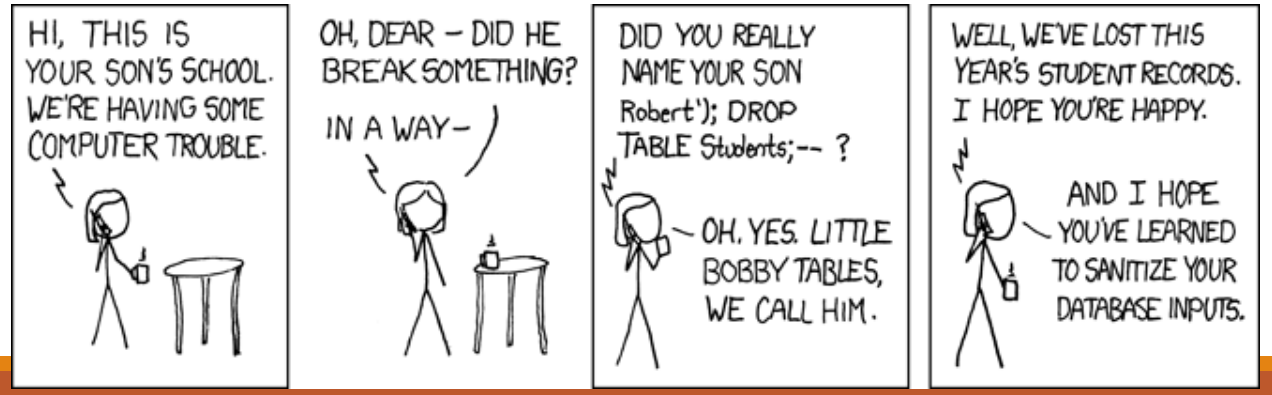
Preventing SQL Injection

For Java, Scala, Kotlin, and other JVM languages use JDBC's PreparedStatement

```
int customerId = 2;

Connection connection =
        DriverManager.getConnection("jdbc:sqlite:C:\\Users\\thoma\\Dropbox\\rexon_metals.db");

String sql = "SELECT * FROM CUSTOMER WHERE CUSTOMER_ID = ?";

PreparedStatement ps = connection.prepareStatement(sql);
ps.setInt(1,customerId);

ResultSet rs = ps.executeQuery();
rs.next();

System.out.println(rs.getInt("CUSTOMER_ID") + " " +  rs.getString("NAME"));

connection.close();
```

More info at:
http://tutorials.jenkov.com/jdbc/index.html
http://www.marcobehler.com/make-it-so-java-db-connections-and-transactions

SQL Injection Humor



SQL Injection in the News

This couple cannot do the simplest things online because their last name is ‘Null’

https://thenextweb.com/insider/2016/03/27/last-name-null-is-tough-for-computers/

Catholic financial services hacked, 130K accounts exposed

http://www.twincities.com/2017/10/16/catholic-united-financial-data-breach-may-have-affected-nearly-130k-accounts/

South Africa’s massive data breach

https://www.moneyweb.co.za/news/tech/revealed-the-real-source-of-sas-massive-data-breach/

TalkTalk gets record £400K fine for failing to prevent October 2015 attack

https://ico.org.uk/about-the-ico/news-and-events/news-and-blogs/2016/10/talktalk-gets-record-400-000-fine-for-failing-to-prevent-october-2015-attack/