SQL for Data Scientists: Designing SQL Tutorials for Scalable Online Teaching

Uwe Röhm
The University of Sydney
Australia
uwe.roehm@sydney.edu.au

Lexi Brent
The University of Sydney
Australia
lexi.brent@sydney.edu.au

Tim Dawborn
Grok Learning
Australia
tim@groklearning.com

Bryn Jeffries[*]
Grok Learning
Australia
bryn@groklearning.com

ABSTRACT

The SQL query language is the 'lingua franca' of transactional databases, and is essential for scalable data analytics. Learning SQL requires practical exercises with databases, including those parts of SQL with side-effects such as DDL statements, triggers, and stored procedures. Teaching this effectively to a non-technical audience is a challenge, especially in times of COVID-19 without face-to-face classes.

The Grok Learning platform allows to design self-paced online tutorials with auto-graded exercises – but it was originally built for teaching programming languages. In this demo, we show how we extended the Grok platform to teach SQL for Data Scientists with comprehensive online learning. Grok supports a rich user interface with interactive examples where students can explore and experiment with each example query. This is ideal for learning declarative querying. Each query is executed in its own sandbox on a freshly initialised database instance which allows to teach all parts of SQL including DDL statements, stored procedures, triggers and UDFs. At the same time, the platform scales to thousands of concurrent users, while maintaining interactive response times.

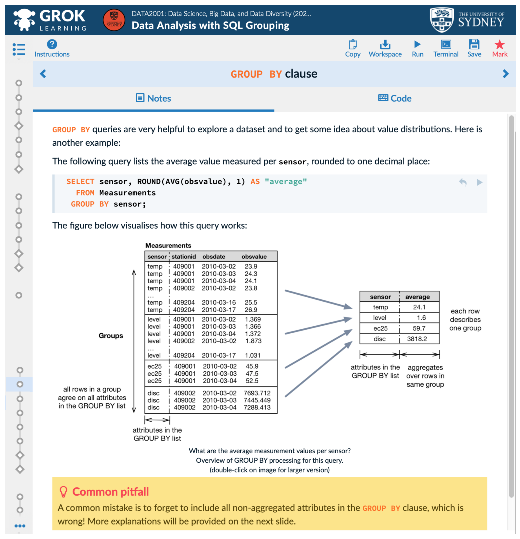
PVLDB Reference Format:
Uwe Röhm, Lexi Brent, Tim Dawborn, and Bryn Jeffries. SQL for Data Scientists: Designing SQL Tutorials for Scalable Online Teaching. *PVLDB*, 13(12): 2989-2992, 2020.
DOI: https://doi.org/10.14778/3415478.3415526

1. INTRODUCTION

Motivation SQL is the lingua franca for databases and large-scale data processing systems. Its declarative nature makes it possible to easily express complex analytical tasks over large datasets without the need to know details of query execution, data distribution, or parallelisation. This declarative thinking has to be carefully taught though to be efficient in query formulation. Users have also to be aware of some pitfalls of the SQL query language, such as the handling of NULLs or not to attempt natural joins between tables without common attributes.

[*]also Honorary Associate of the School of Computer Science, the Faculty of Engineering, the University of Sydney.

Proceedings of the VLDB Endowment, Vol. 13, No. 12
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3415478.3415526


GROK LEARNING
DATA2001: Data Science, Big Data, and Data Diversity (202...
Data Analysis with SQL Grouping
THE UNIVERSITY OF SYDNEY
Instructions
Copy
Workspace
Run
Terminal
Save
Mark
GROUP BY clause
Notes
Code
Measurements
Groups
all rows in a group agree on all attributes in the GROUP BY list
attributes in the GROUP BY list
each row describes one group
attributes in the GROUP BY list
aggregates over rows in same group

GROUP BY queries are very helpful to explore a dataset and to get some idea about value distributions. Here is another example:

The following query lists the average value measured per sensor, rounded to one decimal place:

```
SELECT sensor, ROUND(AVG(obsvalue), 1) AS "average"
  FROM Measurements
 GROUP BY sensor;
```

The figure below visualises how this query works:

sensor	stationid	obsdate	obsvalue
temp	409001	2010-03-02	23.9
temp	409001	2010-03-03	24.3
temp	409001	2010-03-04	24.1
temp	409002	2010-03-02	23.8
...			
temp	409204	2010-03-16	25.5
temp	409204	2010-03-17	26.9
level	409001	2010-03-02	1.369
level	409001	2010-03-03	1.366
level	409001	2010-03-04	1.372
level	409002	2010–03-02	1.873
...			
level	409204	2010-03-17	1.031
ec25	409001	2010-03-02	45.9
ec25	409001	2010-03-03	47.5
ec25	409001	2010-03-04	52.5
disc	409002	2010-03-02	7693.712
disc	409002	2010-03-03	7445.449
disc	409002	2010-03-04	7288.413

sensor	average
temp	24.1
level	1.6
ec25	59.7
disc	3818.2

What are the average measurement values per sensor?
Overview of GROUP BY processing for this query.
(double-click on image for larger version)

Common pitfall
A common mistake is to forget to include all non-aggregated attributes in the GROUP BY clause, which is wrong! More explanations will be provided on the next slide.

Figure 1: Screenshot from the SQL Tutorial.

In data science, SQL is important as a query language to retrieve datasets from a shared database (such as the SQL query interface of SkyServer [5]), to use SQL databases as a sink, or as an effective way to parallelise analysis tasks on distributed data processing platforms (Apache HBASE, Spark or Flink). Silva et al.[3] have proposed that new courses teach SQL in the context of these new systems. Most SQL textbooks and tutorials are, however, written with transactional databases in mind, typically with booking or ordering systems examples - or university schemas. None of those help to explain to data scientists the benefit of SQL or how to formulate data cleaning, data transformation and statistical querying tasks effectively in SQL.

Learning SQL querying skills requires training and lots of practice, and is best done with practical labs. This has led to the development of several online teaching systems for SQL, many of which are however quite static or not very scalable (e.g. www.w3schools.com/sql/). Several universities offer self-paced mini courses or full-fledged MOOCs (Massive Open Online Courses) on databases (e.g. [8, 6]). Those offerings typically cover a wider content, e.g. including relational design theory, with short videos and readings, rather than an auto-graded SQL tutorial. The University of Sydney is using the Grok platform as part of its OLET1301 "Managing and Analysing Data with SQL" mini-course [2].

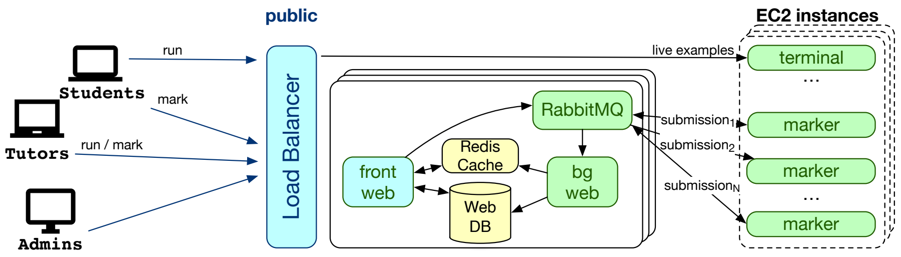


Figure 2: System Architecture of the Grok Learning Platform.

Many automated or semi-automated assessment systems have been developed to assist students in learning elements of SQL. Some more recently reported systems include a unit testing of schema representation to validate DDL [4]; visualisation of relational algebra representations of queries; assessment of SQL queries based upon their outcome as well as use of keywords and formatting style [7].

Contribution In this paper, we present an online SQL tutorial that allows to effectively teach SQL to a non-technical audience, such as Data Scientists, based on the Grok Learning platform [1]. It allows students to learn and experience the complete set of SQL capabilities, including data definition statements, stored procedures, and user-defined functions. To make this possible, Grok was extended to execute each individual query in its own sandbox with a dynamically created database instance. In addition, the systems allows to integrate explanatory pages with text and figures, as well as graded exercises with a flexible mechanism of test cases.

The contributions of this demo paper are as follows:

- We present a comprehensive online SQL tutorial to effectively teach SQL to data scientist students with a variety of real-world scenarios and datasets. In our demonstration, attendees can interactively explore this tutorial, can also try Grok's online tutor helpdesk, and can see the course design interface.
- We give an overview of how the Grok learning platform provides scalability and how it can execute SQL in a sandboxed execution with interactive response times.
- We discuss how to efficiently design test cases to autograde SQL queries with detailed student feedback.

2. SYSTEM OVERVIEW

The SQL tutorial is built using the Grok Learning platform [1]. Grok has been originally designed to teach programming, and its auto-testing mechanism was extended to work with SQL databases too. It is built with a series of clustered micro-services in the AWS cloud, and hence is able to scale to thousands of concurrent users.

The system consists of a front-end for students and tutors, and a scale-out back-end as depicted in Figure 2. The platform distinguishes between three kinds of users: *students* who work through the material and attempt the different exercises; *tutors* who give support and feedback to students and check their progress, and *course designers* who can create and edit content and exercises.

The front-end (blue in Figure 2) allows students to access the course material, to navigate along the teaching material, and to keep track of which learning modules and exercises have been finished already. It is a responsive interface that can be used from any modern web browser and on either computers or mobile devices. It is served from a group of load-balanced web servers which have internal access to a persistent database in AWS RDS, and to volatile cache data in a Redis instance which is managed by AWS Elasticache.

The back-end (green in Figure 2) is responsible for running the users' code and the test cases. Marking tasks are sent from the front-end web servers to the back-end markers via a RabbitMQ message broker. A marker instance picks it up, marks it, and sends the result back to the RabbitMQ message broker, where a background web instance ("bg web") picks it up and writes the result to the database. The markers support different runtime environments. This includes multiple programming languages such as Python, Javascript, bash, R, Java, C/C++, Haskell, Prolog — and database engines such as PostgreSQL or SQLite.

Live examples and users' code executed with 'Run' are handled by separate terminal machines which support the same runtime environments than the markers, but are directly accessed via a WebSocket between the users' browser and the terminal server to provide interactive response times.

Both marker and terminal instances run user code and queries in a sandbox. The sandboxing and process isolation techniques used by Grok are similar to what modern containerisation libraries like Docker use underneath the hood. For the SQL tutorial this means that every query and every test case runs in its own database instance that only exists during the runtime of a query. This gives students full flexibility and enables us to comprehensively teach SQL, including DDL commands with side-effects, but it also requires good design and management of test cases (cf. Section 3).

To minimize the sandbox startup latency, Grok uses a modified version of PostgreSQL which allows to set userid and runtime RAM limits on the query execution process by Grok's sandboxer, and then starts Postgres with a preserialised representation of its database template on a ramdisk. This reduces startup latency to 500ms (cf. Figure 3b).

The admin console supports to create and manage the courses, as described in more detail in the next section. Course designers can create pages and exercises, as well as monitor the progress of their classes.

3. SQL TUTORIAL DESIGN

In the following, we explain how to best design SQL tutorials on the Grok Learning platform, including advanced features for automated testing and interactive exercises:

3.1 Database Preparation

In order to prepare either a marked exercise or an interactive example, course designers can define a query workspace which consists typically of the following components:

init.sql An SQL script that is executed immediately after a new database instance is started. It is typically used to create a schema and to load data.

data.csv While it is possible to include INSERT statements within `init.sql`, a more flexible approach is to provide one or more separate data files which are loaded by `init.sql` (e.g. using Postgres' COPY command). By swapping these data files while keeping the default `init.sql`, a course designer can efficiently create new test cases with slight data variations.

query.sql Optionally course designers can also provide a query template for the student's query editor.

All files can be either internal or public, so that students can see the default setup in different tabs of their workspace editor. If required, they can be set as read-only for students.

3.2 Designing Test Cases for SQL

Exercises can have one or more test cases associated which execute the student submission in the context of a test database, and compare the output of the query against an expected output, similar to the idea of unit tests in programming. The quality of these test cases is crucial to the learning effect for students, and we suggest the following structured approach to create an effective test bank for exercises:

Initialisation — All test cases share a common initialisation of the test driver which can be specialised per each test case. This includes the specification of common files among test cases, such as the database schema. Test cases can replace any of these common files which specialised versions of the files, for example to introduce additional NULL values or to scale some table. Common files can also refer to cloud storage locations which enables to easily upgrade the database schema over the lifetime of the tutorial.

Correctness Tests — It is good practice to define test cases in small, incremental steps where each test case focuses on just one well-defined correctness aspect of the query. This allows for detailed feedback to students on where they went wrong (cf. Section 3.4). This can be achieved by configuring the output checker to focus on specific result parts for each test, and by specifying whether whitespaces and output order should be ignored. In our experience, the following test sequence works well: completeness of result (ignoring order), correctness of output schema (column names), testing on a larger dataset, and correct result ordering (if required).

Testing for Corner Cases — Testing for SQL corner cases is interesting. It involves varying the common dataset with additional NULL values or additional/missing join partners.

Cheating Protection is easily done by using at least one hidden dataset which students don't know and hence cannot submit a static output query (`SELECT` *`correct result`*). When including one hidden dataset with varied key values, one also finds queries which use 'magic' constant IDs instead of joins or sub-queries.

Scalability — Testing for scalability with at least one much larger dataset (typically hidden and if it is really large, than also best linked to a cloud store).

3.3 Teaching Statements with Side Effects

The SQL tutorial also supports SQL statements that do not produce a relational output, but rather modify the database instance or its schema. To do so, a course designer can provide a `final.sql` script that is executed directly after a user submission, and which can check side-effects:

- INSERT, DELETE or UPDATE statements typically have no other output than *n rows affected.* `final.sql` can check the actual modified database content.
- This mechanism can also be used to support DDL statements such as CREATE TABLE or CREATE INDEX by querying the Information_Schema to check that the expected tables or indices have been created.
- Triggers and stored procedures work this way too. The `final.sql` script can invoke the stored procedures or triggers and then check for expected outputs.

3.4 Student Feedback

An important part of an online SQL tutorial is the capability to provide meaningful feedback to students. In our SQL tutorial this is achieved by several means:

Test Titles — Using focused, incremental tests with meaningful titles helps students to understand where their code fails. For example it is better to state "Testing correct ordering of the result" rather than "test 1, test 2, ...".

Expected Output — For the correctness tests on smaller datasets, it is helpful to provide students with a comparison of the output of their submission with the expected output.

Example Solution — Course designers can provide example solutions for each exercise which are either directly available, or after a deadline, or once the student solved a task. The solution can include explanations with markup formatting, figures, and different alternatives (e.g. to explain different solution approaches, such as sub-queries versus join).

Online Helpdesk The Grok platform includes an online helpdesk with which tutors can interact with students using a chat and a sandbox environment in which they can see and trial each student submission themselves.

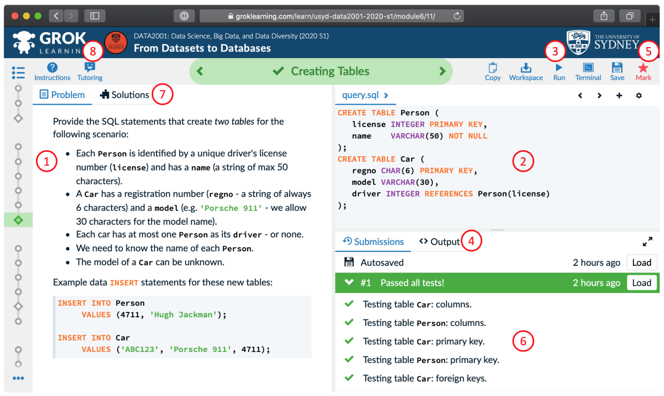
3.5 SQL for Data Scientists

A SQL tutorial for Data Science students should introduce the different SQL concepts in the context of usage scenarios students can relate to, and with limited use of technical jargon in its explanations. In our SQL tutorial, we have use cases from environmental sciences, physics, or social sciences, and the exercises use real datasets (sometimes simplified or shortened). The tutorial's focus is on data import, filtering, transformation, descriptive statistics and correlations, and how SQL can be used in a typical Data Science pipeline. After about four to six content pages, a test exercise allows students to practice the new concepts in context, typically with five exercises per module. With this approach, we are able to built a comprehensive SQL tutorial that covers all aspects of SQL in context.

4. DEMONSTRATION SCENARIO

We demonstrate our approach using a SQL tutorial that we developed for teaching Data Science students at the University of Sydney. The whole tutorial consists of eight modules that cover the complete SQL query language from the import of CSV files, to join queries, working with NULL values, data cleaning and transformation in SQL, nested queries and analytical queries with GROUP BY / HAVING, window functions (e.g. median or rank), and statistical functions such as correlation. Figure 3a shows an example exercise from this tutorial that trains DDL statements.

This tutorial uses PostgreSQL as underlying database system and makes use of several Data Science-oriented datasets that were collected for this course: weather observations, water quality measurements, and the historic Australian convicts transportation database. The following usage scenarios can be demonstrated with this course:



DATA2001: Data Science, Big Data, and Data Diversity (2020 S1)

From Datasets to Databases

✓ Creating Tables

Provide the SQL statements that create *two tables* for the following scenario:

- Each `Person` is identified by a unique driver's license number (`license`) and has a `name` (a string of max 50 characters).
- A `Car` has a registration number (`regno` - a string of always 6 characters) and a `model` (e.g. `'Porsche 911'` - we allow 30 characters for the model name).
- Each car has at most one `Person` as its `driver` - or none.
- We need to know the name of each `Person`.
- The model of a `Car` can be unknown.

Example data `INSERT` statements for these new tables:

```
INSERT INTO Person
     VALUES (4711, 'Hugh Jackman');

INSERT INTO Car
     VALUES ('ABC123', 'Porsche 911', 4711);
```

```
CREATE TABLE Person (
   license INTEGER PRIMARY KEY,
   name    VARCHAR(50) NOT NULL
);
CREATE TABLE Car (
   regno CHAR(6) PRIMARY KEY,
   model VARCHAR(30),
   driver INTEGER REFERENCES Person(license)
);
```

Autosaved 2 hours ago Load

#1 Passed all tests! 2 hours ago Load

✓ Testing table `Car`: columns.

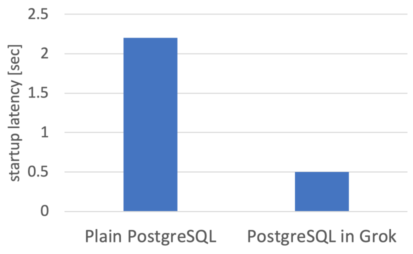
✓ Testing table `Person`: columns.

✓ Testing table `Car`: primary key.

✓ Testing table `Person`: primary key.

✓ Testing table `Car`: foreign keys.

(a) Screenshot of CREATE TABLE exercise.



(b) PostgreSQL Latency Comparison

Figure 3: Demo Screenshot (left) and Latency Comparison of PostgreSQL Sandbox Startup (right)

4.1 Scenario 1: Live Examples and Exercises

The first part of the demo allows attendees to explore the content of the SQL tutorial from the student perspective — which topics are covered with which kind of Data Science-oriented examples, and in particular to experience that all examples on the different pages are *live*: live examples are indicated with a small play button in the upper right corner of the query box (cf. Figure 1). If attendees press this, the content of the example box gets instantaneously executed on one of the terminal instances of Grok (cf. Section 2). Example queries can also be edited, which allows students and demo attendees to freely experiment with the SQL syntax.

Another major part of the tutorial are marked exercises, which attendees can try out too. As shown in Figure 3a, the user sees the problem definition on the left hand side of the browser window ①. This can include schema figures to clarify the example scenario, or as in the example of Figure 3a some code examples which are expected to work on the solution. The example exercise in Figure 3a is about testing the capability to use the SQL's `CREATE TABLE` statement, including primary key and foreign key declarations, and `NOT NULL` constraints. There are over 40 exercises which the attendees can try out during the demo.

Users can enter their SQL solution into the editor window on the right ②, supported by syntax highlighting. They then can first try out their solution on the provided test dataset using the 'Run' button ③, and the output of the query will be shown below the editor ④. Once users are satisfied with this output, they can submit their solution to the marking component by pressing 'Mark'⑤. Their SQL code will now be tested with a combination of public and hidden test cases, and the output of these tests displayed underneath ⑥. Once a submission passed all tests, the user gains some marks for her submission. There are different marker scripts available, e.g. to always give full marks for correct submissions, or to deduct marks for incorrect attempts.

4.2 Scenario 2: Live Tutor Helpdesk

The second part of the demo is about the live tutoring helpdesk of the Grok Learning platform which allows students to ask questions about their current submissions via the 'Tutoring' menu ⑦. These help requests can be answered by a pool of tutors via a central request queue. Demo attendees are able to take on the role of a tutor and experience how they have access to all submissions of a student, including query outputs and error messages, and how they can experiment with different query variants thanks to the sandboxed query execution in Grok. If student and tutor are online at the same time, tutors can chat with the student directly and give them advice on how to improve their queries to solve an exercise. In the current COVID-19 pandemic situation, this online tutoring helpdesk has proven to be very effective to support off-campus online learning.

4.3 Scenario 3: Course Designer View

In the last part of the demo, attendees can see how course designers can edit exercises and test cases. For each question a workspace with initialisation scripts and dataset files can be defined. Data can be specified either in form of small data files which are also visible to students, or by referencing a dataset located on some external cloud store. The latter allows to define exercises on very large databases without the need to download those large datasets to the user's browser, but rather the execution backend can read them directly from the cloud.

Attendees can also see how different dataset variants can be configured in order to check for corner cases of an SQL query, such as NULL values in some attributes or either missing or multiple matching join values. They can also see how hidden test cases can be used with variations in the dataset to avoid that students design test-case specific answers rather than generic queries.

5. REFERENCES

[1] Grok Learning Platform, 2020. https://groklearning.com/universities.

[2] U. Roehm. Managing and Analysing Data with SQL, 2019. https://www.sydney.edu.au/units/OLET1301.

[3] Y. N. Silva, I. Almeida, and M. Queiroz. SQL: From traditional databases to Big Data. In *Proceedings of ACM SIGCSE2016*, page 413–418, 2016.

[4] N. Stanger. Semi-automated assessment of SQL schemas via database unit testing. In *Proc. of the 26th Int'l Conf. on Computers in Education (ICCE)*, 2018.

[5] A. S. Szalay, J. Gray, A. Thakar, and P. Z. K. et al. The SDSS SkyServer: public access to the sloan digital sky server data. In *SIGMOD2002*, pages 570–581, 2002.

[6] UC Davis. SQL for Data Science, 2020. https://www.coursera.org/learn/sql-for-data-science/.

[7] P. J. Wagner. The SQL file evaluation (SQLFE) tool: A flexible and extendible system for evaluation of SQL queries. In *Proceedings of SIGCSE2020*, 2020.

[8] J. Widom. DB: Introduction to Databases, 2020. https://www.classcentral.com/course/stanford-openedx-db-introduction-to-databases-1006.