TEACHING OF SQL

THROUGH A GAME

A Master's Project

Presented to

School of Science

State University of New York
Polytechnic Institute

Utica, New York

In Partial Fulfillment

of the Requirements for the

Master of Information Design

and Technology

---

by

Patrick Ward

May 2015

Patrick T. Ward 2015

SUNY POLYTECHNIC INSTITUTE

DEPARTMENT OF ARTS AND SCIENCE

CERTIFICATE OF APPROVAL

Approved and recommended for acceptance as a thesis in partial fulfillment of the requirements for the degree of Master of Science in Computer and Information Science

____

Date

____

Dr. Ibrahim Yucel

Project Advisor

____

Dr. Russell Kahn

Project Advisor

Abstract

The project seeks to provide an effective alternate method for teaching SQL through the use of a Game. There is value in learning SQL, as SQL skills are still in the top ten list of sought after IT skills for 2015 (Greenspan, 2014). However, lecture based teaching may not fully engage the learner. Therefore, the game was constructed with the MDA framework and Problem-Based Learning in mind. These methods help the learner bridge the gaps between lesson content, problems, and solutions. The game itself was constructed with MSSQL, Adobe Cold Fusion, HTML, CSS, and JavaScript. The game presents lessons on SQL concepts and quiz-based challenges for students to solve. Students faced increasingly difficult challenges as their level SQL knowledge expanded. The project may be found here: http://www.patrick-ward.net/SQLGame/

Table of Contents

Research Questions

The research questions explored in this project were the following:

Primary:

- How does one effectively build a game in a web based environment that is an alternative path to learning SQL than the traditional lecture-based methodology?

Secondary:

- Would a game be an effective teaching tool for a SQL game?
- How does Problem-Based Learning effect learning in the SQL game environment?

Background for Project

SQL, or Structured Query Language, is a way by which people and programs can interact with a RDBMS (Relational Database Management System) such as MSSQL, MSSQL, Oracle, DB2, etc. SQL, as the acronym meaning suggests is a Structured Language, meaning it has a specific way to ask the database for information, this is known as a query. For some SQL is their full-time job, others it may be part of their job, or simply a hobby. Knowing SQL can lead to higher paying jobs and being more powerful in the job they are currently in.

I chose to make learning SQL into a game for a few reasons. First, I have had it in my mind that it would be really useful if someone made learning SQL into game. I know a number of people who want to learn SQL, but are intimidated to learn it. Furthermore, self-learning SQL can be very challenging and requires someone who is motivated to be successful. Even if someone wanted to teach themselves SQL or learn it through an institution it is generally not considered a "fun" experience. A game navigates around these roadblocks. "Fun is about learning in a context where there is no pressure, and that is why games matter" (Koster, 2005). A game is not intimidating because you can fail with no consequences. A game leads and encourages a user through a course. While the user is doing all of the work of teaching themselves, it does not feel like work, so the user is able to do more. It may sound backwards, but games have the power to transform subjects, that at first appear as boring, into something exciting (Lee & Hammer, n.d.). If done properly, learning SQL through this game will be a "fun" experience. The project may be found here: http://www.patrick-ward.net/SQLGame/

Methodology

MDA Framework

Mechanics deals with the components of the game (Hunicke et al., 2004). Perhaps a more specific definition might be: "game mechanics are methods invoked by agents, designed for interaction with the game state" (Sicart, 2008). This could be how high a character jumps or how fast they are able to run. While this game is not quite a first person shooter, it does have its own mechanics. The game follows a quiz based approach. This translates to a player progressing through levels by writing queries against a timer. When a user first enters a level, they will study a lesson. When they feel confident that they can continue to the objective, they click the "Start Game" button. The lesson will disappear and the objective, tables that the user has access to, the solution, and a query window will appear. When the timer runs out, the user will be alerted and will have to start the level over. This will give them a chance to review the lesson. As the user approaches the timer limit, they will be alerted by the characters face. When the user types a query in wrong they will be presented with an error message. Completing levels quicker leads to more stars received on the main menu. Each level can have its own timer settings. For example, if a user completes a level in under 30 seconds, they will get 3 stars. If they complete the level right before the timer is up, they will get 1 star. The more stars received the higher the skill level of the gamer will be. Completing more challenges in faster times also enables bonus big stars to be awarded to the user. The players, or agents, invoke the methods within the game such as running the query or enabling the collapsible panel, actions which were designed for interaction within the game state, in order lead to a desired state.

The **Dynamics** can be thought of as what happens when the game starts; how the user interacts with the rules (mechanics) set in the game (Hunicke et al., 2004). "Dynamics emerge from mechanics" (Nutt, 2008). This game is in no ways a free-roaming world in which the user can explore endless decision making. However, this game offers a playground in which they can learn SQL without feeling defeat if they do not get it right the first time. One of the ways that this helps the user explore their world is the collapsible panels for reference materials. By clicking on the blue buttons/banners the content will toggle between visible and hidden states. This can be used to the player's advantage when they get stuck with a query. Where the game shines is that they user can type in their own queries that actually run against the database (with limited permissions). After the user has typed in their query, the page will refresh with a view of the solution, the results from the query that they wrote, if applicable a few reasons that the query did not match, and their query will still be visible in the query window. They will be able to re-edit their query to make the necessary adjustments. On success, they will be rewarded with a success message. The main menu will show their progress. This feedback loop show what the game does based on its rules (mechanics) and how the user can interact with the rules to progress through the game (dynamics).

Aesthetics deal with evoking the right feelings from the player as they play the game (Hunicke et al., 2004). First, this game presents a challenge to the user. The user has to combine their knowledge (gained from the lesson), use their knowledge to stitch together a query, all before the timer runs out. Second, the user is presented with storylines to put their new-found knowledge to a practical use. These stories help them solidify uses for their SQL knowledge. The goal is avoid the user from saying "when am I ever going to use this?" These stories make the experience more personal. It is not the typical programming examples of "foo-bar" or "apples, oranges, bananas,...etc". It is "your manager has asked you...", not unlike the real world. The achievements, also give the game some depth. The player aspires to further their career through the skill levels, which are gained by earning stars. As the user completes levels quickly, they will accumulate more stars. This provides some level of motivation to know the material enough to better their results. Since skill levels and "big stars" are rewarded after earning a number of small stars, it motivates the player to re-play levels. There are different characters that can be chosen. I have named these (in the code), the professor, male boss, and female boss. The characters allow the player to have more of an emotional connection to the game. The lessons and objectives are no-longer faceless, but now have a character and expressions to provide feedback as they write their queries.

Problem-Based Learning

In recent years, the medical field has adopted a form of learning which places students' education in the same context as the problems they are trying to solve (Donner & Bickley, 1993). PBL arose out of the need for "graduates who were prepared to deal with the information explosion, and who could think critically and solve complex problems" (Major & Palmer, 2001). Kilroy elaborates on what makes a good PBL (problem-based learning) problem, here are few items that stand out, the problem should (Kilroy, 2004):

- present a "...realistic and common" situation

- include problem solving activity
- contain some level of complexity

Problem-based learning appears to be used in the medical field, but this game is applying it toward technological learning. Based on this methodology, the game-based use of PBL seems to be easily applicable. This game presents students with a lesson and then is closely followed by an objective where students are able to apply what they have learned towards a new problem. The objectives follow real-to-life situations in which the student may find themselves in while in a work place environment. The problem solving is slightly different than the medical application of PBL; whereas, the medical field did not provide the students with the answer to the problem, in this game the student is given the answer and has to figure how to get there. In SQL, both the journey and destination are important for continued and repeatable success in real life. While in the beginning the game does not provide a high degree of complexity, towards the end the user has to use their cumulative knowledge to apply the lesson information to increasingly complex examples. One further difference is that PBL often takes place an group based environment, this games does not incorporate other users in order to problem solve (Major & Palmer, 2001).

Dreamweaver & CF11

- Dreamweaver was used to code all of the HTML, JavaScript, CSS, and Cold Fusion code. Dreamweaver was used for a number of reasons including syntax highlighting, the ability to perform FTP to transfer files to the "production" site (DailyRazor host), and allows me to use an object explorer to access files quickly. It has syntax highlighting and code hinting rules for each language built in, which was helpful at times.
- There were a number of scripts that were developed to support the game.

- /cfc/levels.cfc - This contained functions to start/end levels, compare scores, and to get the appropriate number of stars for the main page.
- /cfc/queries.cfc - This contained the code to convert a Cold Fusion query object into a formatted HTML table.
- /css/elements.css -Styling for structural elements of the pages.
- /css/typography.css - Styling for styling text on the page.
- /custom_tags/collapsible.cfm - Code to form the collapsible panels on the page.
- /custom_tags/end_level.cfm - When a user clicks the exit level calls the appropriate function/stored procedure to clean up any tables that were created during the course of a level.
- /custom_tags/start_level.cfm - This is likely the most important file in the game. Here is an outline of the page:
    - Declare variables and components (cfc's) that will be used. This file is used similar to a function. It accepts parameters stored in the ATTRIBUTES scope. This file is used for the content of all levels and objectives.
    - JavaScript to start the level and control characteristics about the level, e.g. pausing a level, starting a level, the timer, etc.
    - Lesson content. The lesson content is being sourced from the /levels/ folder.
    - Objectives content. The objectives content is being sourced from the /objectives/ folder.
    - Tables that the user has access to.

        - The solution that the user must match.
        - Code to handle the user's query and outputting potential issues or results.
        - The Query window.
        - Actionable buttons.
    - /images/{female_boss | male_boss | professor}/ - These contain the images for each of the characters. The files are named the same in each, so that they can easily mapped during the game.
    - /js/main.js - A small amount of code to allow things like the counter and collapsible panels to work.
    - /levels/ - Each level is represented by a file in this folder. These files contain the lesson portion of each level.
    - /objectives/ - Each level is represented by a file in this folder. These files contain the object portion of each level.
    - /references/ - References can be found in collapsible panels on the main menu and sprinkled throughout the game. These are used to help the user get through the level objectives. However, they can also be used as a concise reference for any SQL they use later on.

SSMS & SQL Server 2014

- SQL Server Management Studio (SSMS) was used to access the databases on the database servers both locally and through the DailyRazor host. I created a SQL script in SSMS that initialized the game and all of the solutions. The script includes setting up permissions for the webuser, creating tables, populating tables to be used in the game, creating a view for each of the solutions, populating the tables to run the game, and finally a few stored procedures that help create/remove users, and start/end levels.

- User: webuser - This is the user that will have access to all schemas and the ability to create tables on the fly. The web interface will be using this user to create tables.
- Tables
  - dbo.levels - A list of tables or views used in each level. When a user starts level 3, for example, this table will have a list of all tables or views used in level 3. The tables for level 3 will be copied into the user's schema.
  - dbo.solutions - The view name used for each solution.
  - dbo.users - A list of users who have access to the game. This also includes dummy users that are used to base the high, medium, and low scores.
  - dbo.scores - A list of completion times for each user. The high, medium, and low scores are stored here as well.
  - dbo.levels.people - A list of people used in the game: employees, customers, or co-workers.
  - levels.categories - Product categories used in the game.
  - levels.products - A list of products used in the game.
  - levels.orders - A list of orders used in the game. This is a "fact" table which ties people and products together.
  - levels.properties - A list of properties or attributes that can be associated with products.
  - levels.sales_2013 - A view of how much each user purchased in the year 2013.
  - levels.sales_2014 - A view of how much each user purchased in the year 2014.

- levels.sales_2015 - A view of how much each user purchased in the year 2015.
- levels.all_sales - A super view joining the tables levels.people, levels.orders, levels.products, and levels.categories. This is what a denormalized table would look like.
- v_solution_{level id} - After querying the database, the user's query should match exactly what is found in these views.
- levels.sp_start_level - A stored procedure that places all of the tables relevant to the current level into the user's schema. For example, if the user is "user1", this procedure would copy the tables for the current level into the schema named "user1". When the user clicks on a level from the main menu, this stored procedure is run to initialize the tables needed to run the level.
- levels.sp_end_level - A stored procedure that removes all tables from the user's schema for the current level. When the user clicks "Exit Level" from the user interface, this stored procedure is run to clean up the tables.
- dbo.sp_create_user - This stored procedure creates a database user and schema. It gives the appropriate access to the user and allows the webuser to impersonate the user. The user will have the ability to SELECT, UPDATE, DELETE, and INSERT on their own schema. When the web interface is running a query, webuser is running a statement to run as if it was the user. e.g. EXECUTE AS USER = '{username}' /* ...user query...*/
- dbo.sp_delete_user - This stored procedure removes the specified user, login, and schema from the database.

3rd Party Libraries

- jQuery (//code.jquery.com/jquery-1.11.2.min.js, //code.jquery.com/jquery-migrate-1.2.1.min.js) - jQuery helps perform a lot of JavaScript functionality with very little lines of code. Also, jQuery provides browser agnostic functionality. This means that I only need to code something once and then it is done.
- Flex Box Grid (http://flexboxgrid.com/) - A responsive grid system that allows websites to scale content to fit any screen width, from phone, tablet, laptop, desktop, to television. Due to Flex Box's Grid system, this game could be played on any web enabled platform.
- Google Fonts (http://fonts.googleapis.com/css?family=Oswald, http://fonts.googleapis.com/css?family=PT+Sans, http://fonts.googleapis.com/css?family=Indie+Flower, http://fonts.googleapis.com/css?family=VT323) - Google provides a number of open sourced fonts that can fit a large number of projects. Most notably, this can be seen on the main menu where each level is displayed.
- Button Styling (http://css3buttongenerator.com/) - This generates css code which will transform div's with the class of "btn" into a nice looking mobile friendly button. The collapsible panels and action buttons use this styling.
- Characters (http://www.mightydeals.com/deal/tooncharacters.html) - I purchased a set of characters that could be used alongside each lesson and objective. It gives the game a little more personality and storyline. As the timer increases during the challenge part of a level, the character image changes based on the character images provided in this character pack.

Applied Design Principles/Theories

Hierarchy of Needs (124-125) (Lidwell et al., 2010)

During development I sought to build functionality first, then reliability. The game has sought to be usable by the user, easy navigation and easy layout. The layout and content should increase the proficiency of the user. The user has to be creative when crafting solutions based on objectives given and their knowledge of the content.

Performance Load (178-179) (Lidwell et al., 2010)

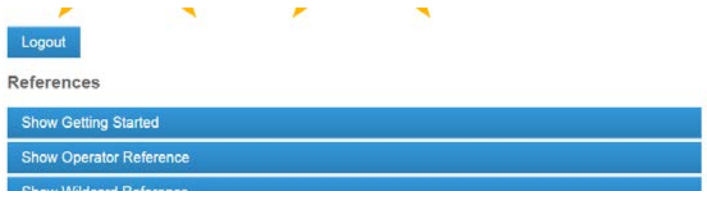
The game is designed to not be too burdensome on the user cognitively nor kinematic. Cognitively, the game is designed in small chunks, levels, in order to teach the user incrementally. Kinematically, the game is designed to be performed with relatively few steps. With two clicks the user could start typing their query.

Progressive Disclosure (188-189) (Lidwell et al., 2010)

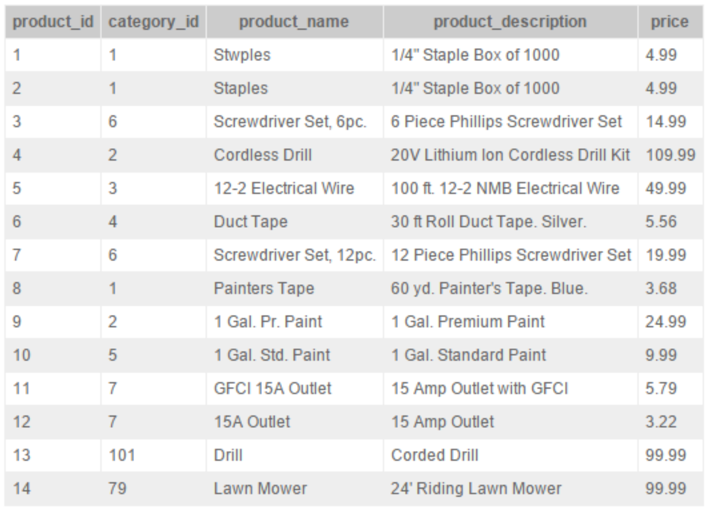
There are a few ways in which the game is designed to minimize the noise of showing all content. One way is by having collapsible panels. These allow the user to only see relevant content when necessary. The main menu allows the user to only play/see one level at a time. When the user enters the level, they only see the lesson content. When they are ready to play the challenge they can only see the objective content. By default the objective tables are hidden to reduce interface clutter. Reference materials are also packaged into collapsible panels.

Uniform Connectedness (246-247) & Consistency (56-57) (Lidwell et al., 2010)

Elements within the game have been designed in a uniform manner to increase connectedness in meaning. Buttons and panels are styled in blue and change state slightly when hovered over.



All collapsible panels are styled similar and contain the word "Show" when collapsed and "Hide" when expanded. Similar, all tables have been styled the same to help the user identify table content.



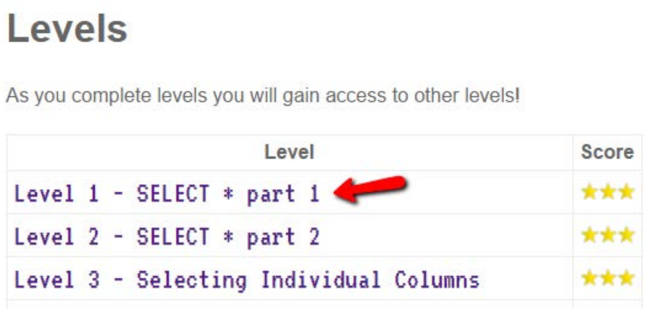
product_id	category_id	product_name	product_description	price
1	1	Stwples	1/4" Staple Box of 1000	4.99
2	1	Staples	1/4" Staple Box of 1000	4.99
3	6	Screwdriver Set, 6pc.	6 Piece Phillips Screwdriver Set	14.99
4	2	Cordless Drill	20V Lithium Ion Cordless Drill Kit	109.99
5	3	12-2 Electrical Wire	100 ft. 12-2 NMB Electrical Wire	49.99
6	4	Duct Tape	30 ft Roll Duct Tape. Silver.	5.56
7	6	Screwdriver Set, 12pc.	12 Piece Phillips Screwdriver Set	19.99
8	1	Painters Tape	60 yd. Painter's Tape. Blue.	3.68
9	2	1 Gal. Pr. Paint	1 Gal. Premium Paint	24.99
10	5	1 Gal. Std. Paint	1 Gal. Standard Paint	9.99
11	7	GFCI 15A Outlet	15 Amp Outlet with GFCI	5.79
12	7	15A Outlet	15 Amp Outlet	3.22
13	101	Drill	Corded Drill	99.99
14	79	Lawn Mower	24' Riding Lawn Mower	99.99

All levels have a unified layout and placement of key buttons, lesson content, and objective content. Each of the characters display similar poses and expressions for the time limits. The game was built with object oriented concepts. While the content of each level is different, the structure and flow is simplified. This should make the game more usable.

Game Description

(Note: For more details see the "Getting Started" collapsible panel on the main menu.)

After logging in you will be able to select from the levels that you have access to.

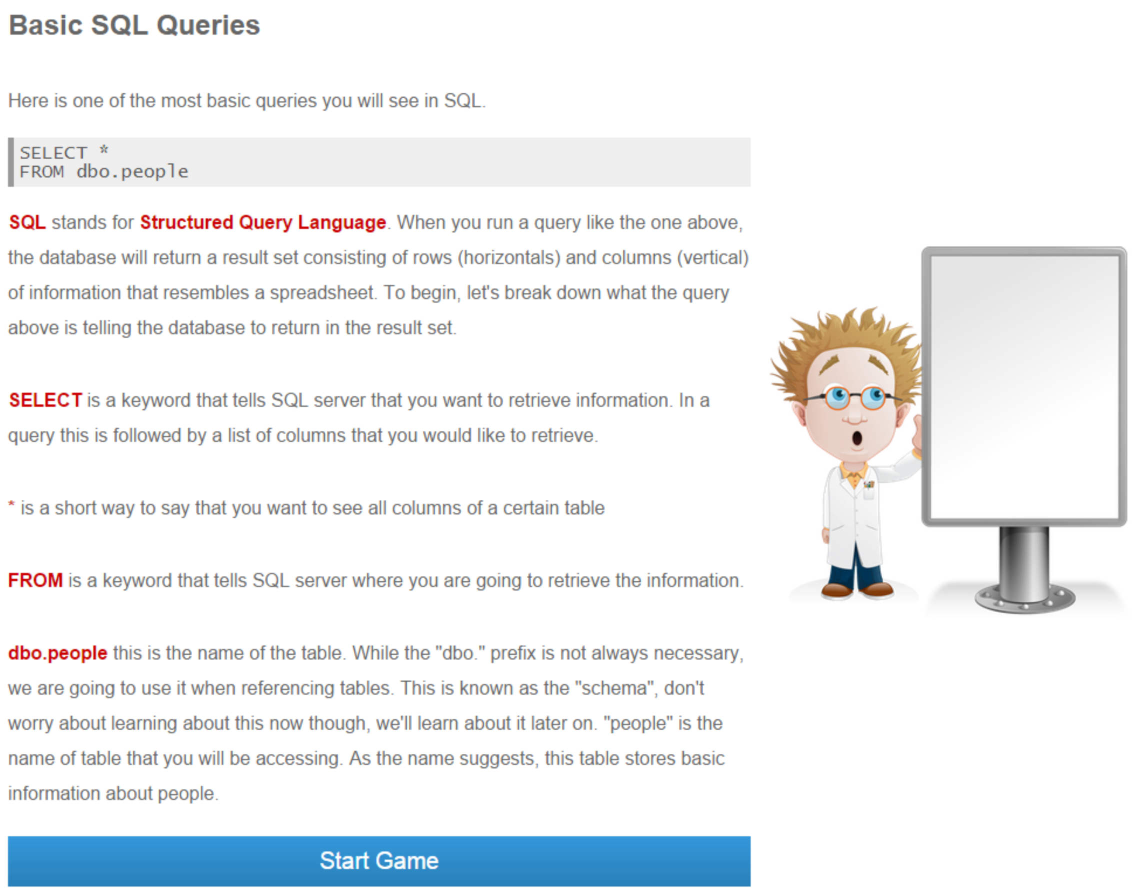


Levels

As you complete levels you will gain access to other levels!

Level	Score
Level 1 - SELECT * part 1	★★★
Level 2 - SELECT * part 2	★★★
Level 3 - Selecting Individual Columns	★★★

By clicking on these links, you will start a level. The beginning of the level will usually be a lesson/tutorial on how to write the SQL statement. When you feel comfortable with the content of the lesson/tutorial click "Start Game".



Basic SQL Queries

Here is one of the most basic queries you will see in SQL.

```
SELECT *
FROM dbo.people
```

SQL stands for **Structured Query Language**. When you run a query like the one above, the database will return a result set consisting of rows (horizontals) and columns (vertical) of information that resembles a spreadsheet. To begin, let's break down what the query above is telling the database to return in the result set.

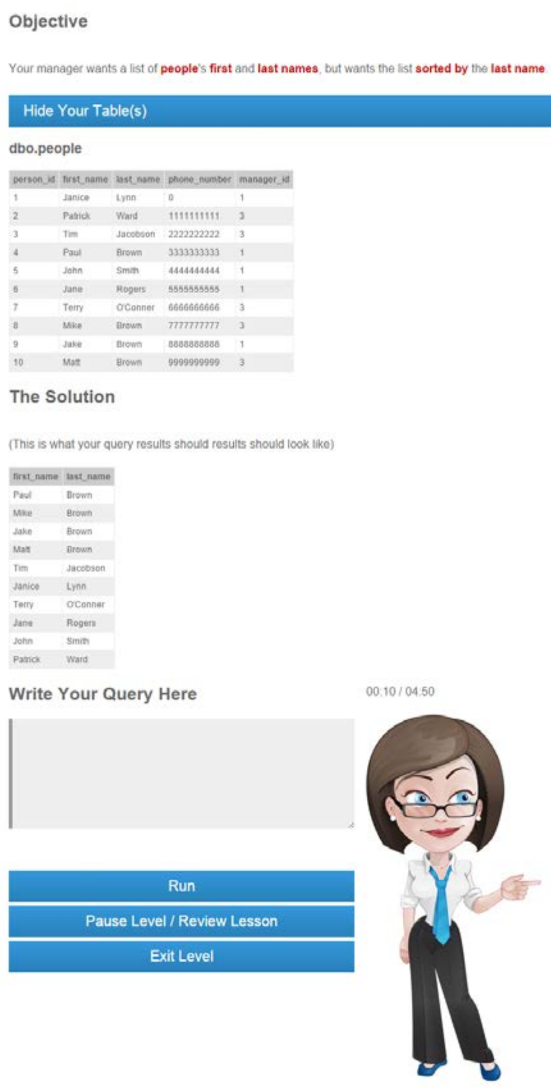
SELECT is a keyword that tells SQL server that you want to retrieve information. In a query this is followed by a list of columns that you would like to retrieve.

* is a short way to say that you want to see all columns of a certain table

FROM is a keyword that tells SQL server where you are going to retrieve the information.

dbo.people this is the name of the table. While the "dbo." prefix is not always necessary, we are going to use it when referencing tables. This is known as the "schema", don't worry about learning about this now though, we'll learn about it later on. "people" is the name of table that you will be accessing. As the name suggests, this table stores basic information about people.

When you click "Start Game" you will be presented with an objective. This is the challenge that you must overcome in order to progress through the game. The objective will be based on the content you have learned in the lesson/tutorial section and previous levels. The objective, tables you have access to, the solution, and the query window will be visible.



Objective

Your manager wants a list of **people**'s **first** and **last names**, but wants the list **sorted by** the **last name**

dbo.people

person_id	first_name	last_name	phone_number	manager_id
1	Janice	Lynn	0	1
2	Patrick	Ward	1111111111	3
3	Tim	Jacobson	2222222222	3
4	Paul	Brown	3333333333	1
5	John	Smith	4444444444	1
6	Jane	Rogers	5555555555	1
7	Terry	O'Conner	6666666666	3
8	Mike	Brown	7777777777	3
9	Jake	Brown	8888888888	1
10	Matt	Brown	9999999999	3

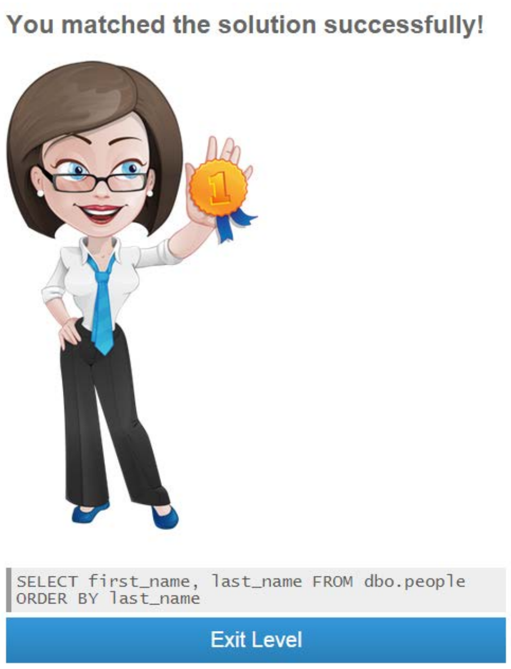
The Solution

(This is what your query results should results should look like)

first_name	last_name
Paul	Brown
Mike	Brown
Jake	Brown
Matt	Brown
Tim	Jacobson
Janice	Lynn
Terry	O'Conner
Jane	Rogers
John	Smith
Patrick	Ward

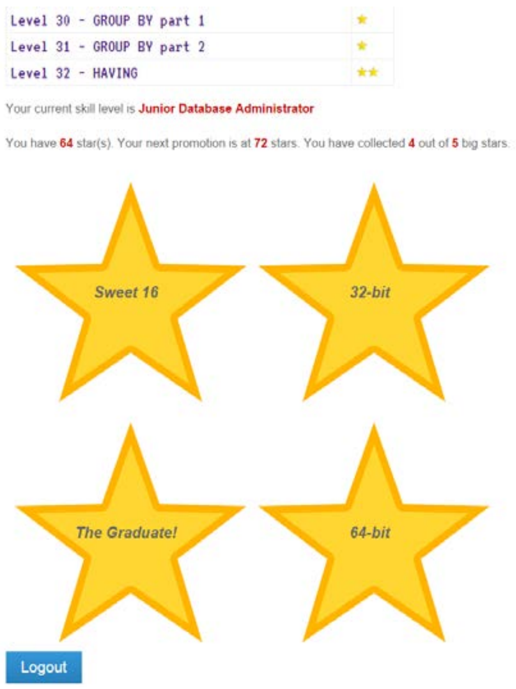
Write Your Query Here

Once you have typed into the query window and feel that your query matches both the objective and the solution, click "Run". If the query does not match, feedback will be given to indicate why it does not match. The feedback can range from having the wrong number of columns, the wrong order of rows, to having a syntax error. If the user has typed in the correct query, they will be shown a success message.


You matched the solution successfully!

```
SELECT first_name, last_name FROM dbo.people
ORDER BY last_name
```

After completing a number of levels, the user will accumulate 1-3 stars for each level. These stars will unlock skill levels which correlate to job titles. They will also unlock milestones in the form of big stars listing the achievement that they have unlocked.



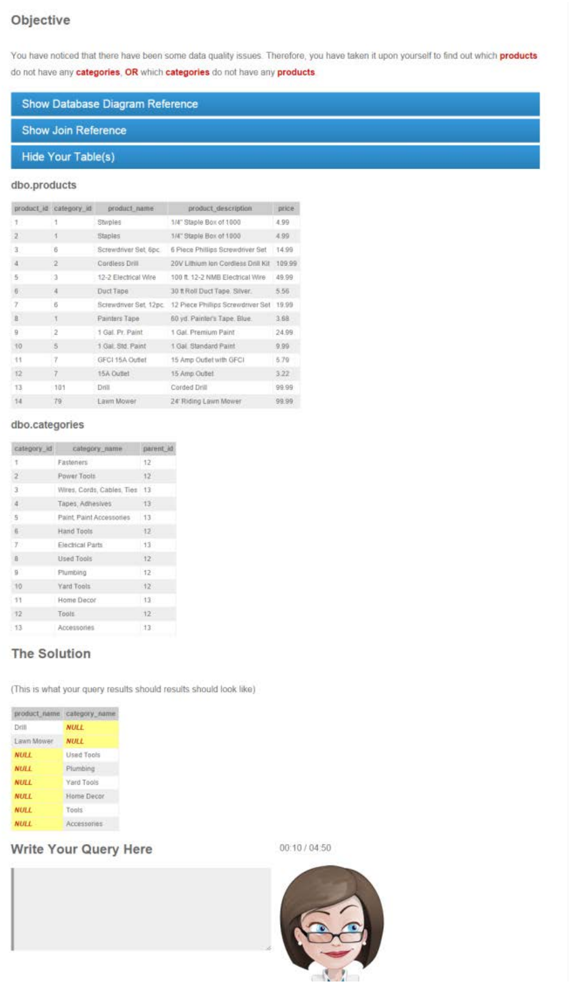
Level 30 - GROUP BY part 1	★
Level 31 - GROUP BY part 2	★
Level 32 - HAVING	★★

Your current skill level is **Junior Database Administrator**

You have **64** star(s). Your next promotion is at **72** stars. You have collected **4** out of **5** big stars.

Findings

Collapsible Panels/Real Estate on the Screen for Content



Objective

You have noticed that there have been some data quality issues. Therefore, you have taken it upon yourself to find out which **products** do not have any **categories**, **OR** which **categories** do not have any **products**.

dbo.products

product_id	category_id	product_name	product_description	price
1	1	Stwples	1/4" Staple Box of 1000	4.99
2	1	Staples	1/4" Staple Box of 1000	4.99
3	6	Screwdriver Set, 6pc.	6 Piece Phillips Screwdriver Set	14.99
4	2	Cordless Drill	20V Lithium Ion Cordless Drill Kit	109.99
5	3	12-2 Electrical Wire	100 ft. 12-2 NMB Electrical Wire	49.99
6	4	Duct Tape	30 ft Roll Duct Tape. Silver.	5.56
7	6	Screwdriver Set, 12pc.	12 Piece Phillips Screwdriver Set	19.99
8	1	Painters Tape	60 yd. Painter's Tape. Blue.	3.68
9	2	1 Gal. Pr. Paint	1 Gal. Premium Paint	24.99
10	5	1 Gal. Std. Paint	1 Gal. Standard Paint	9.99
11	7	GFCI 15A Outlet	15 Amp Outlet with GFCI	5.79
12	7	15A Outlet	15 Amp Outlet	3.22
13	101	Drill	Corded Drill	99.99
14	79	Lawn Mower	24' Riding Lawn Mower	99.99

dbo.categories

category_id	category_name	parent_id
1	Fasteners	12
2	Power Tools	12
3	Wires, Cords, Cables, Ties	13
4	Tapes, Adhesives	13
5	Paint, Paint Accessories	13
6	Hand Tools	12
7	Electrical Parts	13
8	Used Tools	12
9	Plumbing	12
10	Yard Tools	12
11	Home Decor	13
12	Tools	12
13	Accessories	13

The Solution

(This is what your query results should results should look like)

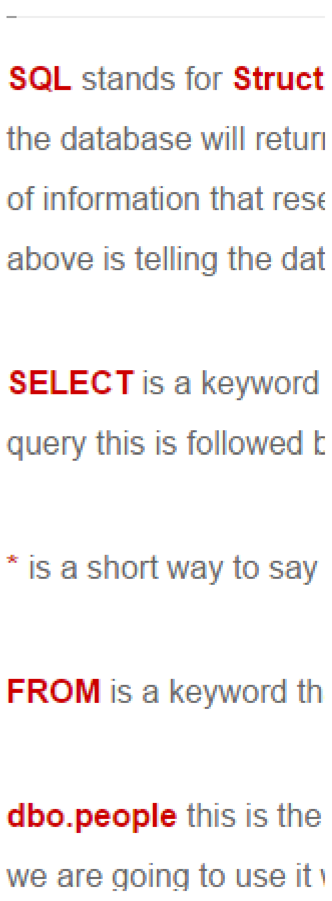
product_name	category_name
Drill	*NULL*
Lawn Mower	*NULL*
NULL	Used Tools
NULL	Plumbing
NULL	Yard Tools
NULL	Home Decor
NULL	Tools
NULL	Accessories

Write Your Query Here

00:10 / 04:50

The above image shows a sample level with the objective, reference material, solution, and query window. This content takes up a lot of room and can be too much for the user to take in. By placing content in collapsible panels, this has made it so that content is only visible when needed. Otherwise, the user is able to see all of the content that they need to see in order to complete the objective. This also leads to a cleaner interface.

Type



SQL stands for **Struct**
the database will returı
of information that rese
above is telling the dat

SELECT is a keyword
query this is followed b

* is a short way to say

FROM is a keyword th

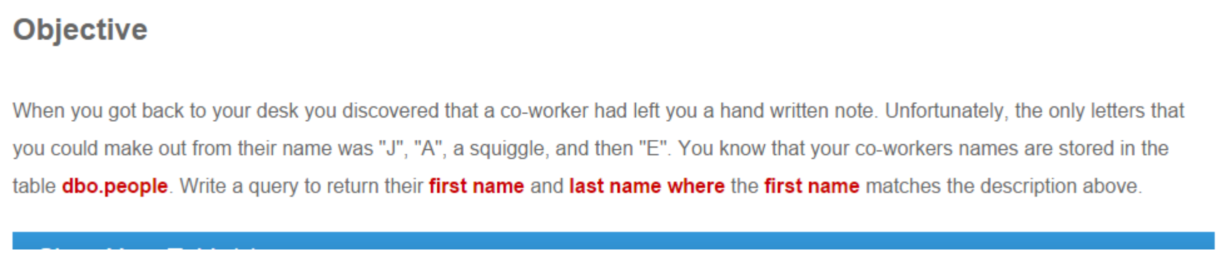
dbo.people this is the
we are going to use it

The use of type to make keywords stand out was found to be useful for learning concepts.

Problem Based Learning

The most helpful feature seemed to revolve around problem based learning.

In recent years, the medical field has adopted a form of learning which places students' education in the same context as the problems they are trying to solve (Donner & Bickley, 1993). In a similar way, students are given a brief lesson on a concept and then immediately put in a situation where they have to problem solve using the concept. The objectives featured ministorylines that brought depth to the learning experience. It did not seem as if the user was in a sterile, rules based environment. The storylines brought a little life to the game.


Objective

When you got back to your desk you discovered that a co-worker had left you a hand written note. Unfortunately, the only letters that you could make out from their name was "J", "A", a squiggle, and then "E". You know that your co-workers names are stored in the table **dbo.people**. Write a query to return their **first name** and **last name where** the **first name** matches the description above.

A good game is a "...series of interesting choices for the player to make" (Totilo, 2012). In this environment, the learner was given a lesson, a mini-story, and placed in scenario where they had to make decisions on how to correctly write the query.

Research Questions

How does one effectively build a game in a web based environment that is an alternative path to learning SQL than the traditional lecture-based methodology?

It is without question that having more education leads to better economic and career fulfillment ("Education: The Rising Cost of Not Going to College," 2014). The big problem with education is that only 46% of those with a 4-year degree found that their education was very useful in preparation for a job or career ("Education: The Rising Cost of Not Going to College," 2014). Yet, "Gamers voluntarily invest countless hours in developing their problem-solving skills" (Lee & Hammer, n.d.). The goal of an educational game like this is to have learners invest their hours for little-to-no money and be able to feel that it was useful for their job without feeling like they have worked. However, does this mean that education and gaming can lead to an effective experience?

The combination of gaming and education can lead to an average user being able to learn a skill like becoming proficient at interacting with databases without feeling like they have worked. This can be described as "Playbor" or "Weisure", that is play-labor or work-leisure (Anderson & Rainie, 2012). Based on feedback, the user indicated that the game was practical enough to be referred back to during work. The game, using problem based learning, made it useful in the user's life and could be considered a success.

The user also indicated that the "real life" examples within the game made easier for them to learn. The user was given a pre- and post-course evaluation consisting of 10 identical questions. During the pre-course evaluation, the user scored a 60%, which by most standards would be considered failing. However, the post-course evaluation, the user scored a 90%. This is significant in identifying that the game-based environment was an effective tool in teaching SQL. The user successfully completed all challenges.

Stated in the section below, " Limitations and Suggestions for Future Game Development", the hosting environment may play a role in the effectiveness of being able to have a web-based environment. In order for this game to be a viable option in the future, it would be better run on a platform which can limit the security for the end user to allow for multiple users. The web based environment made it easy to make the game responsive, styled, and interactive. It was an effective way to deliver content and was cross-browser compatible.

Would a game be an effective teaching tool for a SQL game?

The feedback received indicates that this was successful as a teaching tool. Here is some positive feedback received:

- Certain words were in bold, this helped them to stand out

- I liked that when I went to level 2 it showed what I learned in the last level
- I liked that I could go back and review the lesson again
- I liked that I got more stars the faster I finished something
- In day 1 and day 2 there are clear "real life" examples that I can relate to and this is making it easier for me to learn.
- Good, applicable explanations that help me to understand the concepts

Not all of the feedback was positive. I provided a word document to be filled out during user testing with a few simple questions to help improve the game. I found that this was an incredibly useful tool. The questions asked each day until the course could be completed were:

- What did you like?
- What could be improved?
- Do you feel that you have enough time to complete the queries?

Through this simple questionnaire I was able to discover a number of typos and improvements to the user interface. I was also able to identify that this game really does work for its intended purpose.

How does Problem-Based Learning effect learning in the SQL game environment?

Problem Based Learning appears to have been an successful way to teach SQL in game form. As stated earlier, there are a few concepts which should be included in PBL (Kilroy, 2004), the problem should:

- present a "...realistic and common" situation
- include problem solving activity
- contain some level of complexity

Users playing the game were taught a lesson and then immediately given a context in which they could test their knowledge on what was learned. Problems were presented in "real life" example situations, for example:

> *Your manager wants a list of people's first and last names, but wants the list sorted by both the last name, and first name descending.*

In the feedback I received, the user noted "In day 1 and day 2 there are clear "real life" examples that I can relate to and this is making it easier for me to learn."

Users are also given challenges that incrementally taught them new concepts, while building on concepts that were previously learned. One example of this is that users are given lessons early on how to select individual columns, then throughout the game they must select individual columns even though they are learning new concepts. This appears to have been successful as all levels were completed by the user.

Limitations and Suggestions for Future Game Development

Security Concerns

There were a number of challenges using a web based environment that would need to be overcome in the future. While testing locally I had downloaded the developer version of Cold Fusion Server 11 and had purchased the developer version of MSSQL Server 2014. For both of these products I had full control over my environment. I was the web server administrator for the Cold Fusion 11 server and had the sysadmin role for my MSSQL Server 2014.

I used DailyRazor as my Cold Fusion and MSSQL web hosting solution to make this project accessible for project submission ("Expert & Award-Winning Professional ColdFusion Web

Hosting," n.d.). During this process I found a number of unique challenges with permissions and services. First, DailyRazor uses a shared hosting environment which appears to apply both to the web and database server. This means that problems with another person's website caused 500 server errors frequently with my website. After a series of back and forth emails with their technical support, the errors have seemed to disappear.

The biggest challenge was permissions. For Cold Fusion, this meant that I had limited control on how I could set up my data sources and mappings. I was able to change settings to accommodate these differences in my Cold Fusion code. For MSSQL Server, I had to create users and credentials through their user interface instead of being able to script permissions. This was a trial and error process to see what would or would not work. As a result, I feel that users were given too much access. I would have liked to lock down the users' permissions much more.

As noted before, I feel that the user was given too much access to the database. This is partially a problem due to being on the DailyRazor's web hosting environment. Without going into too much detail there are essentially two database users used during the game: the "webuser" and "user1". The webuser runs all of the queries that pull from the schema's "levels" and "dbo", while user1 only has access to the schema "user1". When the gamer submits a query in the game the webuser impersonates user1 to run the query. This makes it so that new data sources do not need to be created, but the gamer can only SELECT, UPDATE, DELETE, and INSERT on the "user1" schema. On the web hosted environment, it appears that the "user1" user has additional permissions (such as db_ddladmin and db_securityadmin).

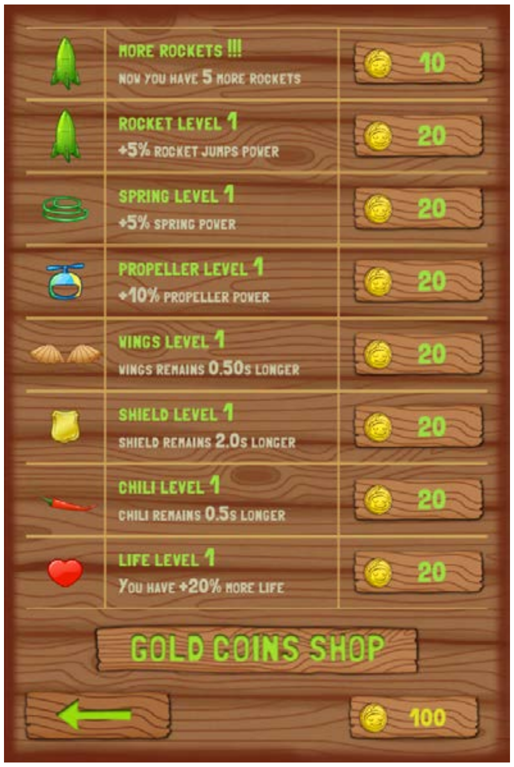
In the future, I would highly consider finding a different hosting solution that allows me to limit the control of the "user1" user, or use some sort of self-hosted solution.

Otherwise, perhaps a more secure way to handle this would be to allow SELECT statements on only the levels schema to be run. If this limitation were in place, the game could become readily available to the public. An example of why the DailyRazor hosting was not quite the best security solution was that I was able to type the following query: SELECT * FROM INFORMATION_SCHEMA.COLUMNS. This returned a list of all schemas, tables, views, columns, and data types for the current database. A hacker could easily use this information to modify their own scores, other people's scores, or worse.

Achievements/Goals/Reward System

While there was an achievement section, I feel that this could be improved upon in the future. For example, the current system is based on how quick the user gets the correct answer. That is not always the best way to develop code. Sometimes the best code is "elegant", which is a nice way to say that it does a lot with less lines of code. Elegant code can take time to provide, but is flexible and resistant to errors. Users could be rewarded based on limited number of characters perhaps.

The interface displayed the achievements near the levels, but the layout could use a little work. For example, consider this example (“iPhone Game Review: Fat Jump Pro,” n.d.):



MORE ROCKETS !!! NOW YOU HAVE 5 MORE ROCKETS	10
ROCKET LEVEL 1 +5% ROCKET JUMPS POWER	20
SPRING LEVEL 1 +5% SPRING POWER	20
PROPELLER LEVEL 1 +10% PROPELLER POWER	20
WINGS LEVEL 1 WINGS REMAINS 0.50s LONGER	20
SHIELD LEVEL 1 SHIELD REMAINS 2.0s LONGER	20
CHILI LEVEL 1 CHILI REMAINS 0.5s LONGER	20
LIFE LEVEL 1 YOU HAVE +20% MORE LIFE	20

The interface is well designed and allows the user to do something with the content they earned. Perhaps, the stars earned could be used to purchase content within the game such as new levels or skill levels.

Incremental Learning

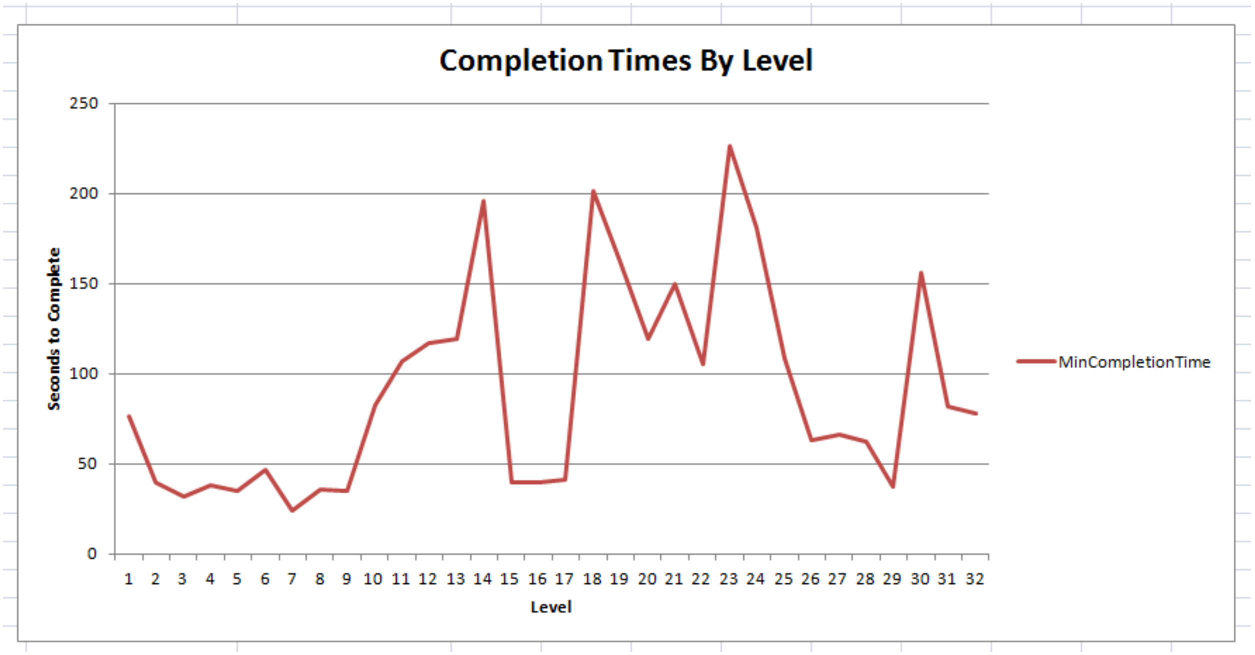
There was a few instances of incremental learning, but due to time constraints I do not feel that it was fully developed. For example, earlier lessons taught the user how to select individual columns, then later lessons often used this to only select certain columns in the output. By the end of the game, the user should have mastery over all concepts learned. There were a number of concepts that were only touched on one time. Given more time, it would have been better to incorporate more concepts previously learned in future lessons.

Number of Levels

The number of levels enabled the user to touch on many key SQL concepts. However, to fully develop a user, I feel that it would have been better to incorporate many more levels. I believe that incorporating two-to-three levels per concept would be sufficient. By the end of each concept, the user would likely have more solid foundation to begin work at a company.

Time Limits

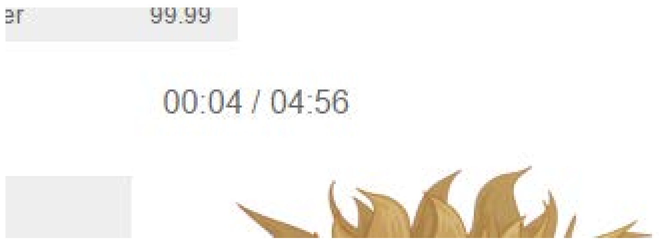
Towards the end of the game, levels became more complicated and required more typing, yet the limits were kept the same for each level. These should be adjusted based on the user feedback and dbo.scores table. For example, during the levels dealing with joins, it was obvious that five minutes was not enough for someone just learning joins to complete the objective.



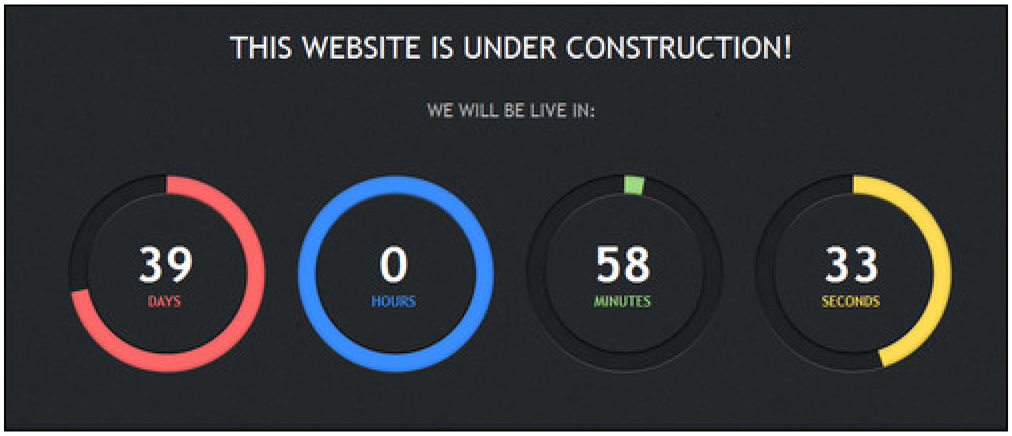
In the chart above, we can see that the highest time to complete a level was for the level dealing with FULL OUTER JOINs. With increasing the time limit, special attention should be given to these levels to ensure that the content being displayed is clear.

Time Limit Display

Feedback was given to say that a warning should be given to warn the user of the approaching timeout limit. Here is the current timer, it is styled appropriately for a professional game:



Perhaps, when the user exceeds 25% of time limit set for the level, the countdown should be bold and yellow. At the 10% range, the timer should turn red. For example, these timers would look good and really help the user indicate how much time was left (Vraa, 2013).



Content per Lesson

The amount of content to put into each lesson is a delicate balance between teaching new content without overloading the learner, but at the same time not boring the user if not enough content is present. I feel that some lessons could have been spread out over more levels. For example, the level "Level 30 - GROUP BY Part 1" teaches the user about COUNT(*), SUM, MIN, MAX, and AVG. While I feel that the lesson presents the content in an understandable way, it could easily been spread out for 2 or 3 lessons. "Level 32 - HAVING" is another example of a lesson that could be spread out for 2 or 3 levels. This is not due to the amount of content, but due to being able to understand how to use it and apply it in a real world situation. One lesson on this, and other levels that are complex-to-entry level users, does not seem like enough time or practice to gain a solid understanding of the content. The same is true for some of the JOIN levels, UNION, INSERT, UPDATE, and DELETE. It would be a good idea in the future to plot the level content out again to right-size each level.

Format of content (video?)

Sometimes reading through a tutorial is a chore. It would be useful to have video content for each lesson in conjunction with the written content. The video would cover all of the same material, but would literally guide the user through the lesson. It would be important to have both video and lesson content available at the same time though. While having video is more interactive of an experience, it is very difficult to search the video in order to review learning content. On the other hand, only having textual content, it is a lot of work to read everything. Textual based content puts a lot of work on the end user.

Cold Fusion & MSSQL vs. PHP and MySQL vs. C# and MSSQL

There was a number of advantages of using Cold Fusion and MSSQL. Personally, I have worked with these two technologies for almost five years now, so I have comfortable grasp on what can be done. It is relatively easy to work with query objects in Cold Fusion. In fact, it is relatively easy to work a lot of things in Cold Fusion. The session management and database integration were another advantage of using it for the project. However, the Cold Fusion/MSSQL combination is costly. It is fairly difficult to sift through the web hosting solutions until I found Daily Razor's at the cost I did. That is due to Cold Fusion and MSSQL being proprietary software to ADOBE and Microsoft respectively. Cold Fusion and MSSQL being proprietary also mean the security patches are released to ensure that the application is protected from potential holes.

In previous experience I worked with PHP and MySQL. That being said, either language and RDBMS could have been used for the project. A few major advantages with PHP/MySQL are as follows. PHP/MySQL are both open-source technologies. Due to this, it is very easy to find a web hosting solution that will support PHP/MySQL. The community of developers for PHP/MySQL is thriving, almost anything imaginable has been done by someone else in this combination. If I were to do the project over again and had more time to fiddle with the code, I may opt for PHP/MySQL. Being open-source, however, also can mean that security holes can be left in the software un-patched.

Another solution that lowers the risk of security fallout from security holes is to install the project on the clients computer via executable. A language, such as C# or Java, could be used to create an executable that runs on the client's machine. This executable would also install an instance of SQL Server Express or LocalDB. In this scenario, if the user hacks the database, there is no fallout to other users.

Free Play

Dr. Yucel brought this up after reading the proposal. This would allow the user to type in queries directly without being tied to a challenge or level. In its current state the game does not allow the user to experiment with the content they have learned. However, incorporating this into the game could enable the user to become much more proficient at SQL.

Debug Mode

Another feature that would be extremely helpful for the new beginner would be a debug mode that would help them debug an incorrect query. This would essentially be a checklist for them to fill out. For example:

- ✓ Columns are in the correct order? E.g. "col1, col2" not "col2, col1"
- ✓ Column names are spelled correctly? E.g. first_name, not frist_name
- ✓ Commas placed after all columns in the select list except the last column? E.g. "col1, col2" not "col1, col2,"
- ✓ Table has the correct schema? E.g. dbo.people, not ddbo.people

Conclusion

Using a game to teach users how to write SQL queries appears to be another effective tool in teaching. This game has the ability to help users become better at their jobs or begin a career that interacts with SQL. Since it is based in a game form, it can be an effective way to introduce users to SQL who might otherwise be too intimidated, or busy, to learn. Through the course of the game the user should have the ability to comfortably write SELECT, UPDATE, INSERT, and DELETE statements. Using problem-based learning techniques, the learner should have a deeper interaction with problem solving in SQL.

Bibliography

10 Professional Cartoon Characters in 400+ poses - only $24! (n.d.). [Design]. Retrieved May 21, 2015, from http://www.mightydeals.com/deal/tooncharacters.html

Anderson, J., & Rainie, L. (2012, May 18). The Future of Gamification. Retrieved March 2, 2015, from http://www.pewinternet.org/2012/05/18/the-future-of-gamification/

Donner, R. S., & Bickley, H. (1993). Problem-based learning in American medical education: an overview. Bulletin of the Medical Library Association, 81(3), 294–298.

Education: The Rising Cost of Not Going to College. (2014, February 11). [Research and Statistics]. Retrieved March 18, 2015, from http://www.pewsocialtrends.org/2014/02/11/the-rising-cost-of-not-going-to-college/

Expert & Award-Winning Professional ColdFusion Web Hosting. (n.d.). [Web Hosting]. Retrieved May 21, 2015, from http://www.dailyrazor.com/coldfusion-hosting

Flexbox Grid. (n.d.). [Web Development]. Retrieved May 21, 2015, from http://flexboxgrid.com/

Google Fonts. (n.d.). [Web Development]. Retrieved May 21, 2015, from https://www.google.com/fonts

Greenspan, D. (2014, December 17). Top Ten IT Skills In Demand for 2015 [IT Careers and Training]. Retrieved March 23, 2015, from http://www.itcareerfinder.com/brain-food/blog/entry/top-10-technology-skills-in-demand-2015.html

Hunicke, R., LeBlanc, M., & Zubek, R. (2004). MDA: A Formal Approach to Game Design and Game Research. Retrieved from http://www.aaai.org/Papers/Workshops/2004/WS-04-04/WS04-04-001.pdf

iPhone Game Review: Fat Jump Pro. (n.d.). [Technology]. Retrieved May 21, 2015, from http://www.tapscape.com/iphone-game-review-fat-jump-pro/

jQuery. (n.d.). [Web Development]. Retrieved May 21, 2015, from https://jquery.com/

Kilroy, D. A. (2004). Problem based learning. Emergency Medicine Journal, 21(4), 4.

Koster, R. (2005). A theory of fun for game design. Scottsdale, AZ: Paraglyph Press.

Lee, J. J., & Hammer, J. (n.d.). Gamification in Education: What, How, Why Bother? [Research and Education]. Retrieved March 17, 2015, from https://www.academia.edu/570970/Gamification_in_Education_What_How_Why_Bother

Lidwell, W., Holden, K., & Butler, J. (2010). Universal Principles of Design (rev. and updated). Beverly, Mass: Rockport Publ.

Major, C. H., & Palmer, B. (2001). Assessing the Effectiveness of Problem-Based Learning in Higher Education: Lessons from the Literature. Academic Exchange Quarterly, 5(1). Retrieved from http://www.rapidintellect.com/AEQweb/mop4spr01.htm

Nutt, C. (2008, February 18). GDC: Game Design Workshop: Mechanics, Dynamics, Aesthetics [Gaming Articles]. Retrieved May 25, 2015, from http://www.gamasutra.com/view/news/108415/GDC_Game_Design_Workshop_Mechanics_Dynamics_Aesthetics.php

Sicart, M. (2008). Defining Game Mechanics. The International Journal of Computer Game Research, 8(2). Retrieved from http://gamestudies.org/0802/articles/sicart

Totilo, S. (2012, July 9). The Difference Between A Good Video Game and a Bad One [Gaming Articles]. Retrieved March 24, 2015, from http://kotaku.com/5924387/the-difference-between-a-good-video-game-and-a-bad-one

Vraa, L. (2013, June 25). 25+ Cool jQuery Countdown Scripts – Add Dynamic Timers! [Technology Articles]. Retrieved May 21, 2015, from http://www.tripwiremagazine.com/2013/04/jquery-countdown-scripts.html

Wanderski, S. (n.d.). CSS Button Generator [Web Development]. Retrieved May 21, 2015, from http://css3buttongenerator.com/

Vita

Place of Birth	Sayre, PA
Date of Birth	May 10, 1985
Post High School Education	Davis College Bachelor of Religious Education May, 2009
Positions Held Since Obtaining Bachelor's Degree	Web Developer Programmer Analyst