PAGE

Samantha Brennen-Lisko

Syracuse University



TABLE of CONTENTS

General Environment……………………………………………………………………….……3

Information Environment…………………………………...……………………………………6

Technical Environment…………………………………………………………………………..11

Summary…………………………………………………………………………………………25

References………………………………………………………………………………………..28

Tables and Diagrams……………………………………………………………………………..29

Data Dictionary…………………………………………………………………………………..47

GENERAL ENVIRONMENT

Partners for the Advancement of Gifted Education of Wake County, Inc. (PAGE) describes itself as:

“PAGE is a nonprofit volunteer-run organization of parents, teachers, and others interested in supporting and encouraging the development of gifted students by promoting an understanding of the characteristics and needs of gifted students and fostering local appropriate educational opportunities at home, school, and in the larger community.” (PAGE)

While PAGE wants to be able to deliver these services to the community as well as continue to organize the Wake County Spelling Bee, it runs into the challenges of being able to raise enough money to support these services as well as how to communicate this with some of the key stakeholders. These stakeholders are the students that PAGE serves, the parents who PAGE would like to do more for, the schools in the area, and local business who support PAGE.

The first challenge PAGE has is how it raises funds and how to increase those funds. As of right now PAGE has two main ways to raise funds, the first is through their Super Saturday classes and memberships, which is done twice a year. The second is through donations from local businesses. As of five years ago PAGE Super Saturday was bringing in receipts close to $18,000 a year, yet as of last year PAGE brought in receipts of only $10,000. This decrease in receipts has endangered the Wake Country Spelling Bee as well as the ability to afford mailings for parents. This lack of the ability to afford to do mailings, has hindered the ability to reach parents who do not have an email address. PAGE would like to get a better understanding as to why this has happened and how they can change this. Also, with the economy in an upward trend, PAGE would like to be able to take advantage of this by finding ways to increase donations from local businesses. PAGE has began to meet with new local businesses to see how they could develop relationships with them that would be beneficial to both PAGE and the local business. These businesses began with looking at PAGE’s Mission statement:

“Our mission is to provide enrichment, advocacy, and support for the entire gifted community. Through our planned classes each year, our students are able to learn about topics of interest in an environment that suits their learning styles. Our goal is to be part of the conversation in Wake County for how gifted children are identified and provided for in classrooms. We also participate in hosting speakers whose work relates to educating and raising gifted children.” (PAGE)

After looking at the mission statement and having some conversations with PAGE’s president, Brandi Brown, they began to ask themselves some very important questions:

1. Do we have a Super Saturday and Spelling contact and their contact information at all schools?
2. Since PAGE cannot have income information about students, do we have a contact at the school who can pass Super Saturday class information on to students from lower socioeconomic backgrounds?
3. What are the different ways in which PAGE can communicate with parents?
   1. Phones Calls?
   2. Letters in the mail?
   3. Emails, do all parents have an email address?
   4. Letters from the school?
   5. Do we know their primary language, if not English?
4. Are parents are signed up to volunteer with PAGE?
5. Do schools that host their Spelling Bee also have students participating in Super Saturday?
6. Do all students participate in Super Saturday classes?
7. Does PAGE offer a balance between creative and STEM classes?
8. Who are the pronouncer and judges for Spelling Bee?
9. PAGE needs one pronouncer and three judges for the Spelling Bee, does PAGE have enough in the database to choose from?

This led PAGE to ask themselves, how does PAGE pull together this information so that we can answer these questions as well as get a better understanding of how to grow PAGE? PAGE decided to contact a company offering data organizational services so that they can not only answer these questions but get a better understanding on how many students and schools PAGE is reaching and where they need to improve this.

INFORMATION ENVIRONMENT

At this present moment, the only software PAGE is using is QuickBooks so that they can track their finances of both the Spelling Bee and Super Saturday, as well as, be able to file all needed IRS tax forms. Unfortunately, this program does not give PAGE the ability to collect other information such as information on the students, the school, PAGE volunteers, or the students’ parents. PAGE began by putting the data they have collected for both Super Saturday and the Wake County Spelling Bee into excel spreadsheets to see if there are common entities and attributes. They noticed that the student, their address, and email were in both spreadsheets, yet they also noticed that key pieces of information that was gathered for Spelling Bee was not gathered for Super Saturday. One of these key pieces of information was the school that students participating in Super Saturday attended. This process has helped the board to really examine how can PAGE is reaching all students when PAGE is not gathering even gathering which schools students who participate in Super Saturday attend. (see spreadsheets pages 29-30)

PAGE began looking at each column from both of the spreadsheets and saw that these entities and attributes were where they decided to pull into one database serving all of PAGE’s needs. They began with the following list:

* Student
  + Name
  + Address
  + Birthdate
  + Gender
  + Shirt size
* Parent
  + Name
  + Parents’ address
  + Phone number
  + Primary language
  + Email address
* School
  + School address
  + School phone
  + Super Saturday facilitator
  + Spelling Bee facilitator
  + Grade levels
* Super Saturday
  + Classes
  + Vendors

But the Board realized that they were missing information about the Spelling Bee pronouncer, judges, and volunteers. They decided to add these to the list:

* Spelling Bee
  + Pronouncer
    - Phone number
    - Email address
  + Judges
    - Phone number
    - Email Address
* Volunteers
  + Name
  + Phone Number
  + Email Address

Now that they have put together all the information, PAGE’s board soon understood that they were going to need more than a spreadsheet to collect and sort their data. They brought in We Do Databases, Inc. (WDD) to take this project to the next level.

WDD began by taking their list and drawing up what is called a Conceptual Model of the database. They began by asking questions of the board to get a better understanding of what information must be collected and what information PAGE would like but does not have to have. For instance, WDD determined that all the information listed under the student is essential, not all is under the parents. Parent phone number and email is one area that WDD had to make changes and it is discovered not all parents have both a home number and cell number so they decided that just a phone number was required. Also, not all parents have access to the internet and email address. PAGE wants to be able to have ways to reach lower socioeconomic families and to do this they need to have other ways to be able to communicate to them. PAGE also noticed that more and more parents have English and a secondary language so PAGE and WDD decided that Parent Primary Language needs to be a required field. From these meetings WDD broke PAGE’s entities down to eight entities with their attributes:

* Student
  + Name
  + Address
  + Birthdate
  + Gender
  + Shirt size
* Parent
  + Name
  + Parents’ address
  + Phone number
  + Primary language
  + Email address
* School
  + School address
  + School phone
  + Super Saturday facilitator
  + Spelling Bee facilitator
  + Grade levels
* Super Saturday Classes
  + SS class name
* Vendors
  + Vendor name
  + Vendor address
  + Vendor phone number
  + Vendor email address
* Spelling Bee
  + Pronouncer name
  + Pronouncer phone number
  + Pronouncer email address
  + Judge name
  + Judge phone number
  + Judge email address
* Volunteers
  + Volunteer name
  + Volunteer phone number
  + Volunteer email address

WDD then drew up a model that would show the tentative relationships between the entities that would be the best way to organize the data PAGE wants to capture so that they can better understand their non-profit. They also determined that the database program will be used to create the unique primary key for each entity as none of them have a unique attribute. (see model page 32)

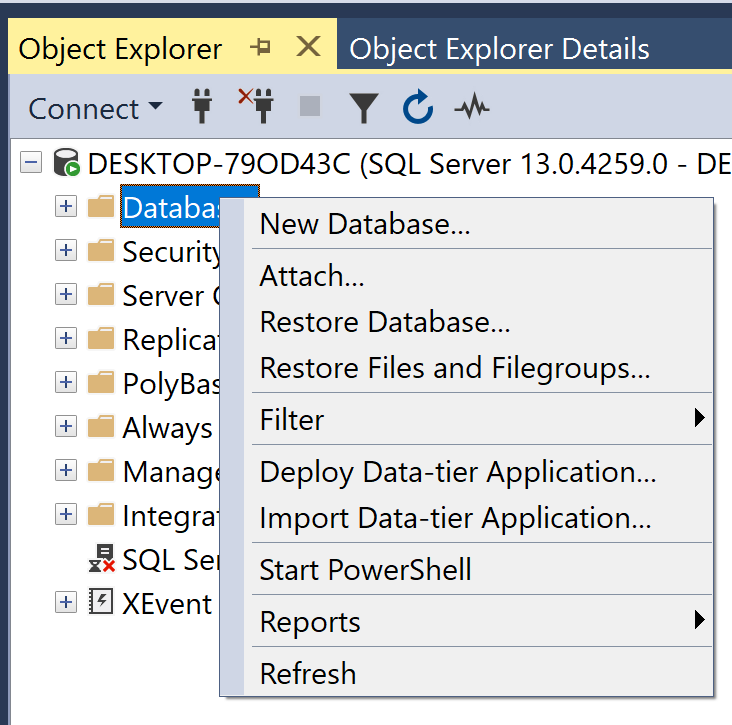
WDD then took this model and the information PAGE provided about the relationships between the entities and planned the Logical Model. Some areas they needed to focus on were the relationships between the Student and Parents entities, Super Saturday classes and Grade Levels entities, School and Grade Level entities, Super Saturday classes and Volunteers entities, and Spelling Bee and Volunteers entities. WDD and PAGE determined that a student could have one or more parents. They could have one parent due to a death, both parents married and living together, both parents divorced living in separate homes, or divorced parents remarried. It was also determined that a parents or parents could have one child or more than one child. With this information WDD determined that this created a many to many relationship between the student table and parent table so a bridge table will need to be placed between them.

WDD and PAGE then examined the relationship between Super Saturday classes and Grade-Levels entities, and School and Grade-Levels entities. It was determined that each Super Saturday class has multiple grade levels and each grade level has multiple Super Saturday classes, thus a bridge table needed to be placed between them. Another bridge table was placed between the School and Grade Level entities due to each school having multiple grade levels and each grade level being assigned to each school. Both companies were able to compose the list of business rules for all the tables which can be found on page 34.

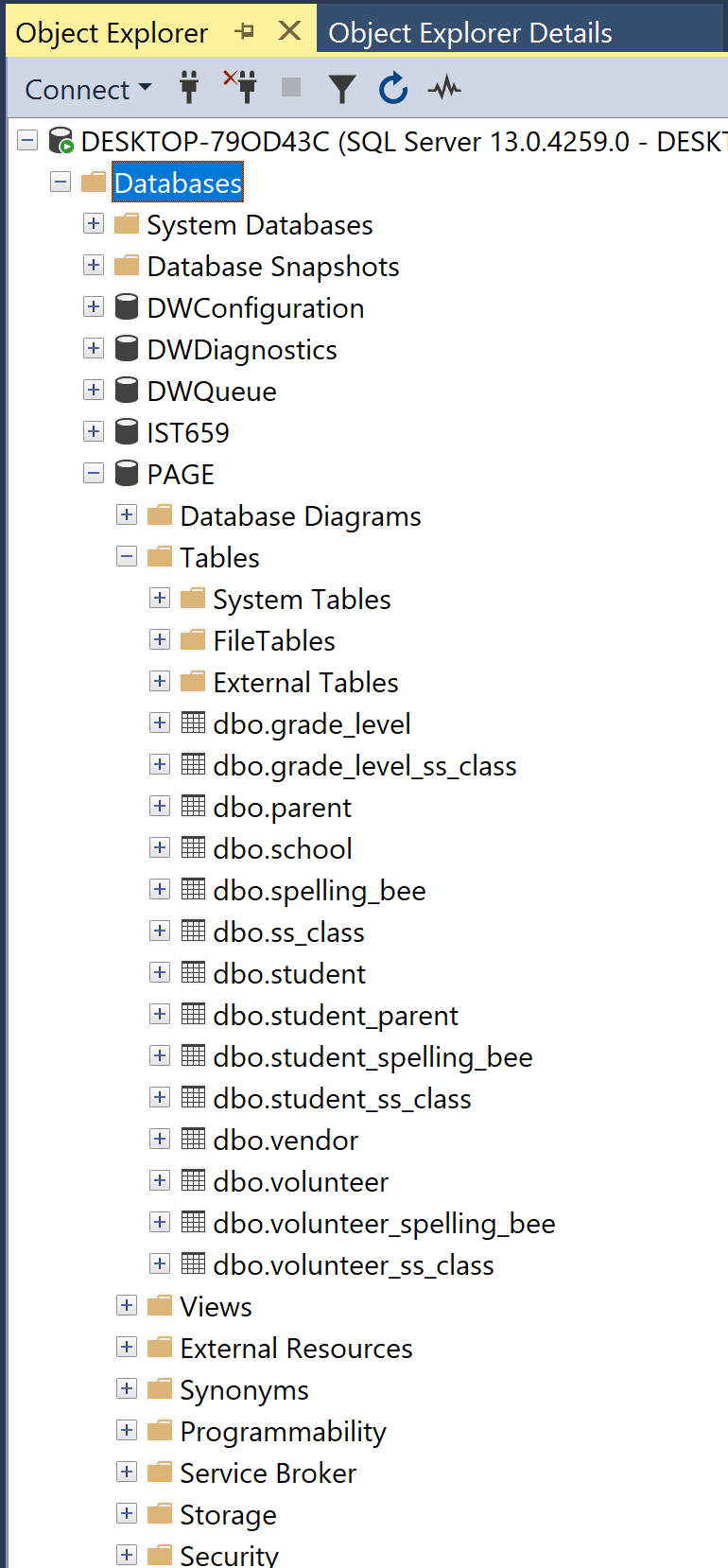
WDD then went on the design the tables from the entities in the Conceptual Model. They broke down the attribute name in the student, parent, spelling bee, volunteer and school entities to a column for the first name and a column for the last name. They then broke down the address in the student, parent, vendor and school entities to a column for the street address, the city, the state, and the zip code. In this phase all the entities become tables of the database and the attributes become columns within those tables. The first column of each table becomes the primary key, which is important because it is used as an identifier of that table as well as a link between tables. As was stated before a bridge table needs to be placed between the student and parents table due what was found to be a many to many relationship. For the student table the primary key is student ID and for the parent table the primary key is parent ID. When the bridge table student\_parent is placed between the two tables, the primary keys student ID and parent ID is placed in the table to create the link. These primary keys become what is called a foreign key in this bridge. The many to many relationship now becomes a one to many relationship between the student and student\_parent tables and the parent and student\_parent tables. The foreign keys are placed in the bridge table because the many side of the relationship is on the bridge table side. WDD continued went on to draw the remaining tables, columns, and join these tables. (see diagram 2 page 33)

TECHNICAL ENVIRONMENT

Once the Logical Model has been approved by both WDD and the PAGE Board, WDD decide to use Microsoft SQL Server to implement the database that PAGE needs. First step WDD needs to do is create the database in SQL Server. To connect to SQL Server, WDD will use the application SQL Server Management Studio (SSMS) because SQL Server cannot be directly interacted so another application must be used to communicate with the database model. Once WDD opened SQL Server the first window asked them to connect to the server. In this window SQL Server will ask for the “Server Type” and for this line WDD needs to make sure “Database Engine” is selected on the dropdown menu. The next line is for the “Server Name” which for WDD will be the servers IP address of 13.0.4259.0. The final line before logging into SSMS is the “Authentication,” because there is more than one employee the works in SQL Server, WDD has it set up that “SQL Server Authentication” so that the employees user name and password has to be entered. This will take the user to the opening page and the window of Object Explorer where all the databases can be found and created. To create a database the user will need to right click Database under Object Explorer which will open the menu where the PAGE database can be created. Once the database has been created it will appear in the list once “Database” is expanded.



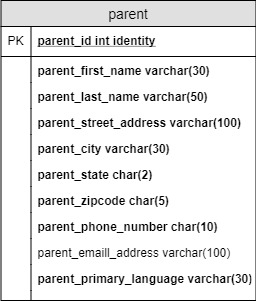
*SQL Server creation menu for PAGE database*



Now that the PAGE database has been created in SQL Server, WDD will need to add the tables and relationships that were designed in the logical model. In SQL Server this is done by writing SQL code. The code that WDD writes to create the database needs to not only be executable but also readable for other users. SQL code can be written in a few lines or as multiple lines. Since there are many parts to the CREATE TABLE statement it is best show each part on a separate line. In the following statement for the Parent Table you can see the writer separated each column with a new line and a comma. While the new lines are not required for SQL Server the commas between the columns are. It is also highly suggested that when writing the statement that the author also put in explanations to help the reader or future users to understand why the code was written. This is accomplished by starting the line with two dashes (- -) followed by the note. In SQL Server any comments will appear in green. If the comment needs to be longer than one line the author would do a forward slash (/) followed by an asterisk (\*) then write their comments ending with an asterisk (\*) and then a forward slash (/). WDD put in explanations thorough the code so that anyone using the editing the database would have a complete understanding of what was written and why.

PAGE database tables in SQL Server

The Parent table was the first one created since it does not have any foreign keys. WDD had to be careful in the order in which the tables were created because the foreign keys were not null therefore the tables without foreign keys needed to be created before the tables that contain the foreign keys.

-- Create Parent Table

CREATE TABLE parent (

-- Columns for the parent table

parent\_id int identity,

parent\_first\_name varchar(30) not null,

parent\_last\_name varchar(50) not null,

parent\_street\_address varchar(100) not null,

parent\_city varchar(30) not null,

parent\_state char(2) not null,

parent\_zipcode char(5) not null,

parent\_phone\_number char(10) not null,

parent\_email\_address varchar(100),

parent\_primary\_language varchar(30) not null,

-- Constraints on Parent Table

CONSTRAINT PK\_parent PRIMARY KEY (parent\_id)

)

-- End creating Parent Table.

*Parent Table Logical Model* *SQL Server code for Parent Table*

It is very important that WDD follow the Logical Model that they designed based off of the needs of the PAGE Board. In the code above you can see that WDD wrote the code following the Logical Model. All column names that are bold in the table’s picture are not null in the code. The primary key is clearly labeled in the code matches the primary key identified in the table. After the code for the table has been written, it needs to be highlighted, and then click on the Execute button on the toolbar to execute the command. Now the remaining tables need to be written and the was done in the following order so not to cause issues with the foreign keys, the School table, the Vendor table, the Grade-Level table, the Spelling Bee table, the Volunteer table, the Student table, the Student-Parent table, the Student-Spelling Bee table, the Volunteer-Spelling Bee table, the Super Saturday table, the Grade-Level-Super Saturday table, the Student-Super Saturday table, then finally the Volunteer-Super Saturday table. The bridge table are predominantly towards the end as they contain foreign keys from two different tables. Since PAGE is new to having a database, WDD decided that it was best to keep the foreign keys the same in both tables, though in some databases they could have been different. It was discovered that in the Spelling Bee table that pronouncer was spelled wrong in one of the columns. Now this can be fixed using the ALTER statement or in SQL Server it can be done simply by right cliking the column in the Object Explorer window. This note was placed in the code explaining how the name was changed:

/\* To correct column name pronounder\_first\_name varchar(30) not null,

I right clicked the column under Object Explorer, then corrected

the column name to pronouncer\_first\_name varchar(30) not null,

\*/

Now the table name is correct when the using the SELECT statement to pull up the table.

Once the tables have been written and executed, WDD needs to enter or insert the data from the PAGE Board into these tables. This is accomplished by using the INSERT statement.

For the Parent table the first line of code starts the manipulation of the data with the statement INSERT INTO, then stats the table name, and finally in parentheses are the columns names separated by a comma. While SQL Server is not picky capitalization, it is picky about parentheses and quotation marks. This it then followed by the VALUES statement a parentheses and then all the values needing to be entered into the PAGE database. Each value should have a single quote at the beginning and end of the value to tell SQL Server where the value begins and ends. This also causes the coade to turn red which is very helpful when you mis a quotation mark. The following statement was written to enter lines of code into values into the Parent table.

-- Adding data into Parent Table

INSERT INTO parent(parent\_first\_name, parent\_last\_name, parent\_street\_address, parent\_city, parent\_state, parent\_zipcode

, parent\_phone\_number, parent\_email\_address, parent\_primary\_language)

VALUES

('Lindy', 'Shwenn', '1 Milwaukee Street', 'Cary', 'NC', '27519', '9199717801', 'lshwenn0@geocities.jp', 'English'),

('Venus', 'Ogers', '770 Oak Point', 'Raleigh', 'NC', '27506', '9195693844', 'vogers1@wiley.com', 'Spanish'),

('Emlen', 'Readwin', '212 Novick Street', 'Raleigh', 'NC', '25706', '9197027306', 'ereadwin2@webnode.com', 'English'),

('Desmund', 'Rosewell', '91 Debs Crossing', 'Apex', 'NC', '27511', '9194536439', 'drosewell3@themeforest.net', 'English'),

('Devon', 'Rawles', '590 Briar Crest Terrace', 'Morrisville', 'NC', '27560', '9195777560', 'drawles4@yahoo.co.jp', 'Hindi'),

('Jesus', 'Madre', '53317 Ilene Plaza', 'Raleigh', 'NC', '27509', '9196477809', 'jmadre5@webs.com', 'Spanish'),

('Nan', 'Kennicott', '1296 Ruskin Place', 'Cary', 'NC', '25719', '9197527459', 'nkennicott6@wp.com', 'Mandarin'),

('Henry', 'Lieb', '2683 Tennessee Court', 'Apex', 'NC', '27511', '9193335063', 'hlieb7@dailymotion.com', 'English'),

('Opal', 'Coulthard', '48073 Crowley Way', 'Wake Forest', 'NC', '25709', '9191403281', 'ocoulthard8@hc360.com', 'English'),

('Eveleen', 'Lum', '270 Quincy Court', 'Knightdale', 'NC', '27580', '9196047295', 'elum9@tumblr.com', 'English')

-- Examine Parent table

SELECT \* FROM parent

Once the values have been entered into the table a good programmer always checks the table to see that the data has been inserted correctly. This is done with the statement SELECT \* FROM parent. The asterisk after the Select statement tells SQL Server to return all the columns of the table. While WDD was inserting the data into the table, it was discovered that a couple of parent volunteers were missing from the volunteer table. This was easily correct by writing a second Insert Statement and notes stating that additional values were entered. This can be seen in the following lines of code and notes.

-- Enter extra data into volunteer table

INSERT INTO volunteer(volunteer\_first\_name, volunteer\_last\_name, volunteer\_phone\_number, volunteer\_email\_address)

VALUES

('Opal', 'Coulthard', '9191403281', 'ocoulthard8@hc360.com'),

('Eveleen', 'Lum', '9196047295', 'elum9@tumblr.com')

-- Examine Volunteer table

SELECT \* FROM volunteer

After all the data has been entered into the database it is now time to write some queries to extract the needed data to be able to answer the business questions the PAGE Board has gathered. Because of the type of data that PAGE wanted inserted into the database and the type of questions they wanted answered, WDD was limited to just using the CREAT VIEW command. WDD does have some suggestions as to further information that could be collected which would give PAGE the ability to use more functions of SQL Server and give them the ability to get an even better understanding of where they need to grow. These suggestions will be covered in the analyzation section towards the end of this document. One of the questions the PAGE Board wanted to be able to answer is how can PAGE reach out to parents since not all parents have email. WDD put together a view named ParentContactInfo which list all the information PAGE collects on parents. This view gives the PAGE Board members the ability to gather all the needed information in one report. Only when the code was originally written the values under ParentAddress did not display correctly. WDD then wrote a DROP VIEW statement which permanently deleted the ParentContactInfo View. They then rewrote the statement adding commas between the values for the address. As shown in the code below it shows the original CREATE VIEW statement, then the DROP VIEW statement, and then the new CREATE VIEW statement.

-- To create Parent contact list (for Question 3)

GO

CREATE VIEW ParentContactInfo

AS

SELECT

parent\_last\_name + ',' + parent\_first\_name AS ParentName

, parent\_street\_address + parent\_city + parent\_state + parent\_zipcode AS ParentAddress

, parent\_phone\_number AS ParentNumber

, parent\_email\_address AS ParentEmail

FROM parent

GO

-- Drop ParentContactView due to errors

DROP VIEW ParentContactInfo

-- To create Corrected Parent contact list (for Question 3)

GO

CREATE VIEW ParentContactInfo

AS

SELECT

parent\_last\_name + ',' + parent\_first\_name AS ParentName

, parent\_street\_address + ',' + parent\_city + ',' + parent\_state + ' ' + parent\_zipcode AS ParentAddress

, parent\_phone\_number AS ParentNumber

, parent\_email\_address AS ParentEmail

FROM parent

GO

-- Examine Parent Contact List

SELECT \* FROM ParentContactInfo

Creating Views in SQL Server helps to combine values from the table to make it more visual when creating reports in Microsoft Access. Because WDD already set-up this database with login credentials, they had set it up so that PAGE had its own log-in credentials that gives them the ability to change values within the tables but not the ability to change the actual tables. This was done because PAGE does not have a member with the ability to understand how to alter the database without compromising the integrity of the data. (see full SQL code starting on page 34)

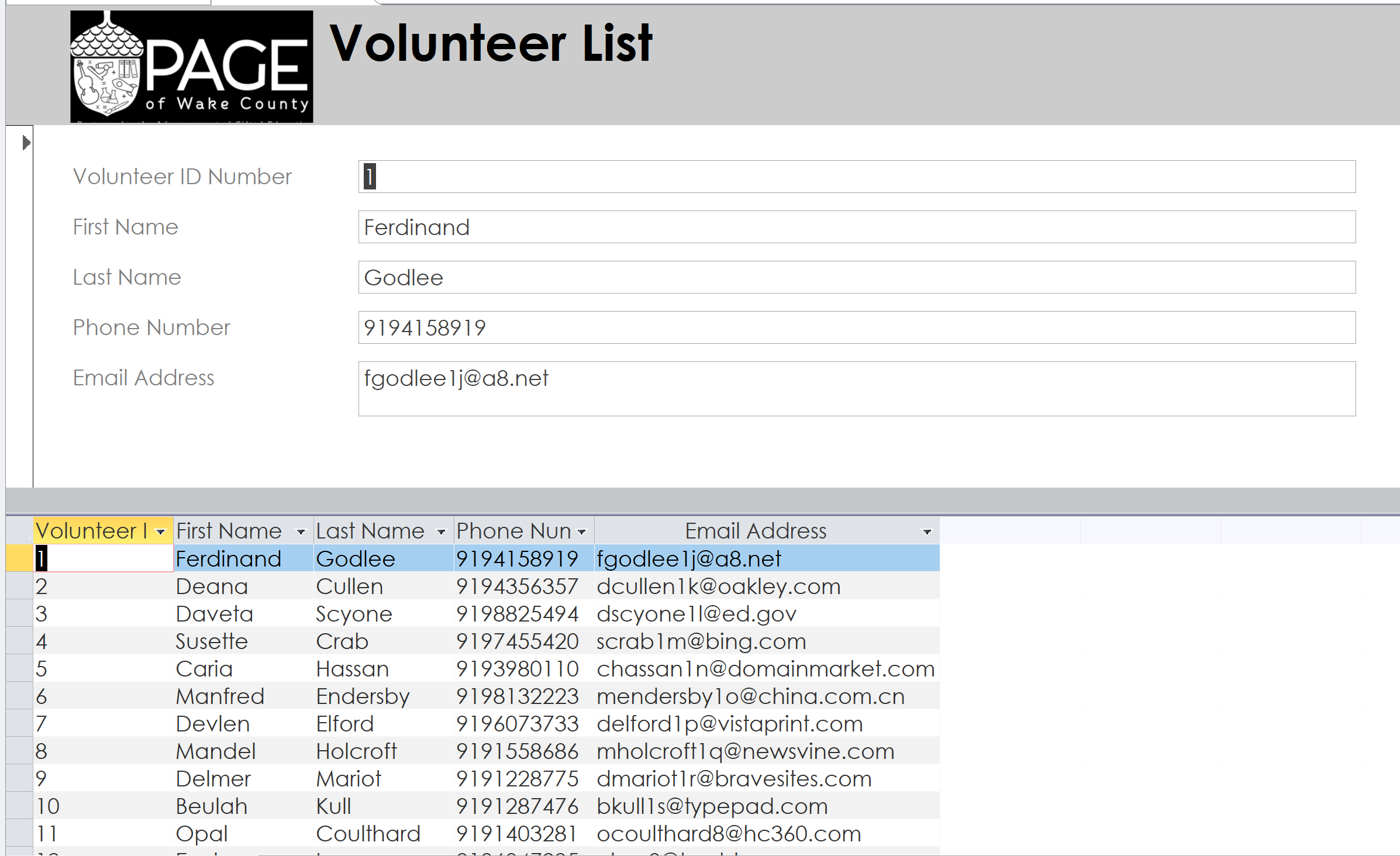
The next step WDD did was to create a live link between Microsoft Access and SQL Server. This will give the PAGE Board the ability to update information themselves without having to contact WDD for everything. First WDD created an Open Database Connectivity (ODBC) by opening the OBC Data Source Administrator applet on PAGE’s computer. Once the window open, they will click the PAGE name under the Name column and then click the add button. This will then take them the Create New Data Source window where WDD will click on ODBC Driver 17 and click Finish. On the Create a New Data Source to SQL Server window, WDD entered the following information:

Name: PAGE

Description: 64-bit PAGE Connection

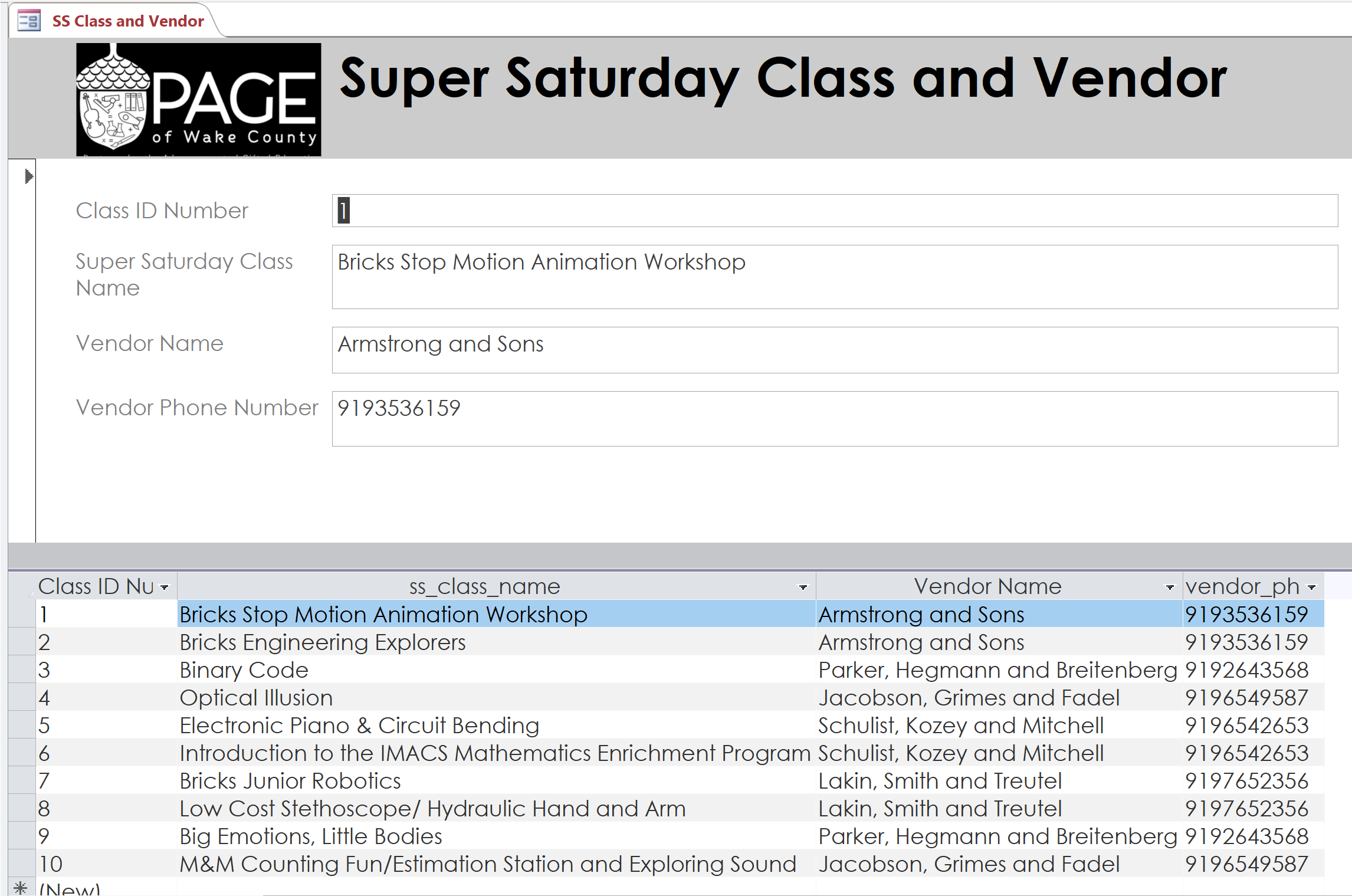
Server: localhost

WDD then clicks Next and on the next window select the button next to “With Integrated Windows authentication” then click Next. The next window is where the correct database is selected. The box next to “Change default database to” should be checked and the PAGE database should be selected, click Next and then Finish. On the final screen WDD clicked test Connection. This returned the message “TESTS COMPLETED SUCCESSFULLY.” Now WDD will be able to link SQL Server to Access. Now on the PAGE computer, WDD opened up Access and created a database called PAGE. They then clicked on the tab “External Data,” clicked on “New Data Source,” then clicked on “From Other Sources,” and then finally “ODBC Database.” On the next window WDD made sure to click “Link to the data source by creating a linked table,” and then chose all the tables and views from the PAGE database. This linked everything from SQL Server to Access. After the tables have been linked to Access, the relationships between the tables needs to be established. Because there are quite a few tables, it should be done two tables at a time. In Access, WDD clicked on the “Database Tools” tab, click on “Show Table” and start with Parent and Student-Parent tables. Then drag the primary key of the Parent table to the Parent-Student table. This will open another window in which both tables and shown, the primary key for each table is shown, and then at the bottom the relationship is one-to-many, click create. WDD then continued to do this with all remaining tables. This established the relationship between all the tables. This process goes more smoothly when the primary key and foreign key is stated the same in both tables, which is why WDD chooses to always state them the same. Now that SQL Server is linked to Access PAGE will be able to create Forms and Reports to present the needed information. WDD created two forms for PAGE, the first form shows the how easy it is to pull needed volunteer information when Spelling Bee or Super Saturday arrives.



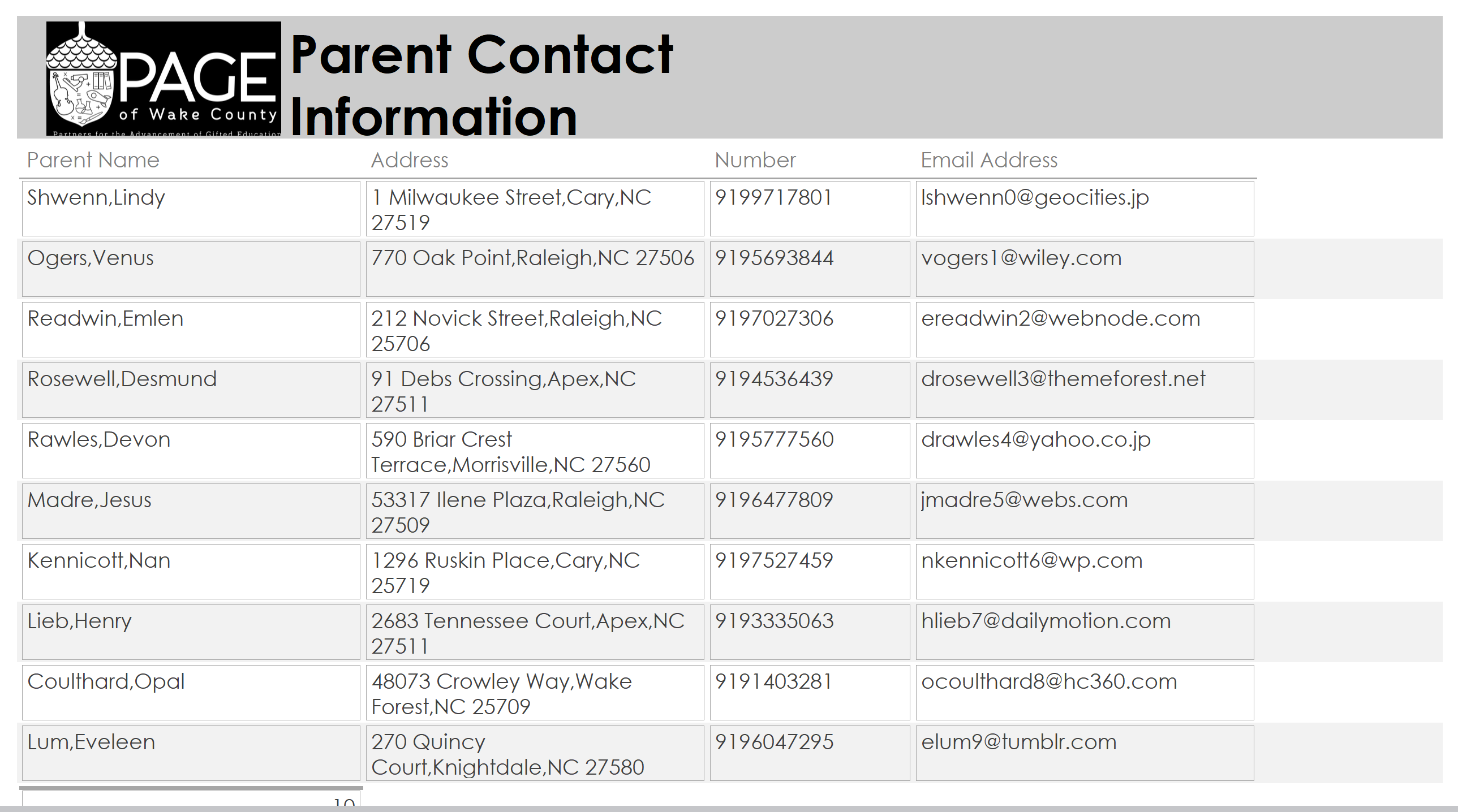
*PAGE Access Form for Volunteer information*

The second form WDD created combines columns from two tables, the Super Saturday table and the Vendor table. This will give PAGE the ability to see which vendor teaches which class in a timely manner.



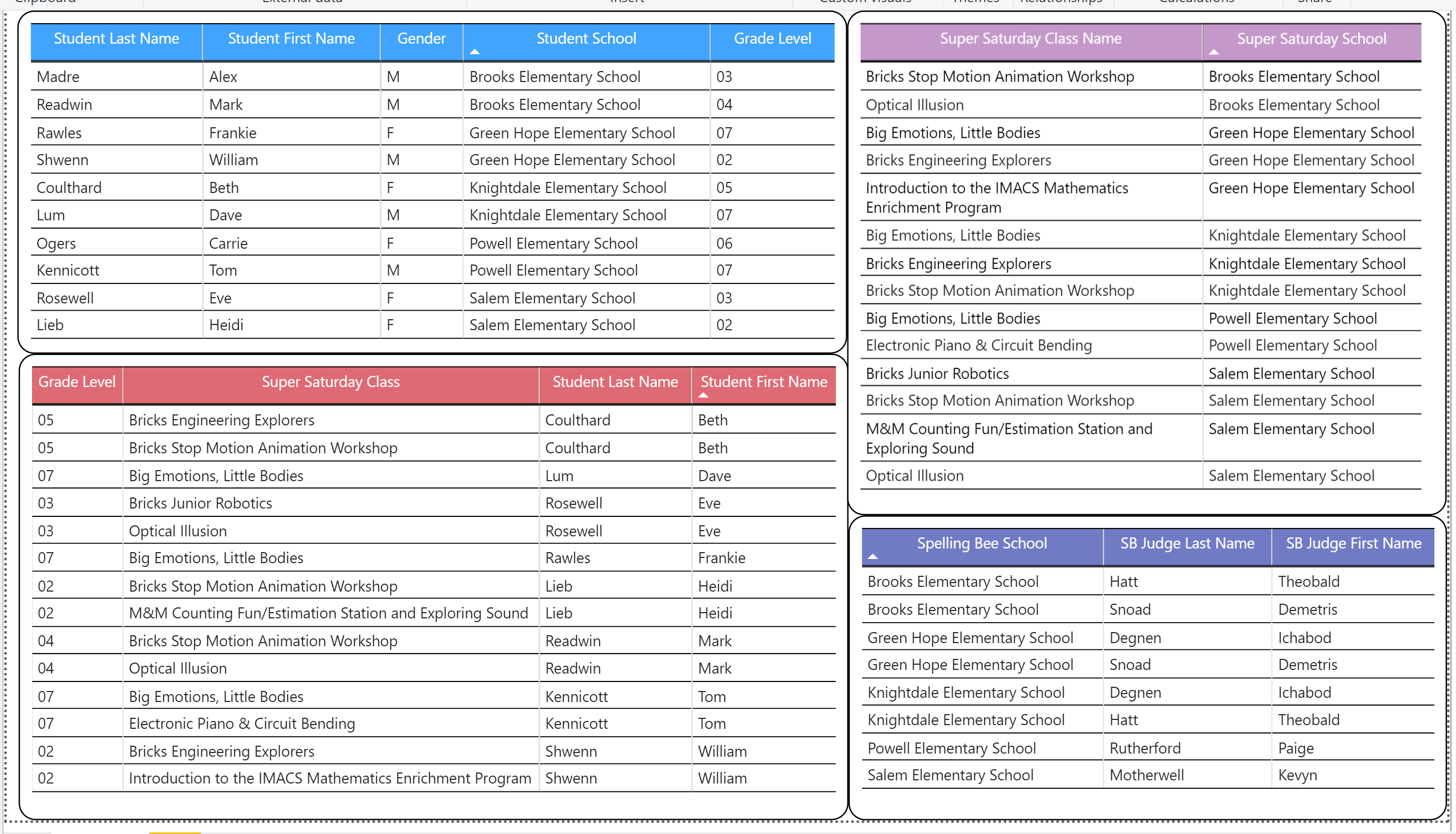
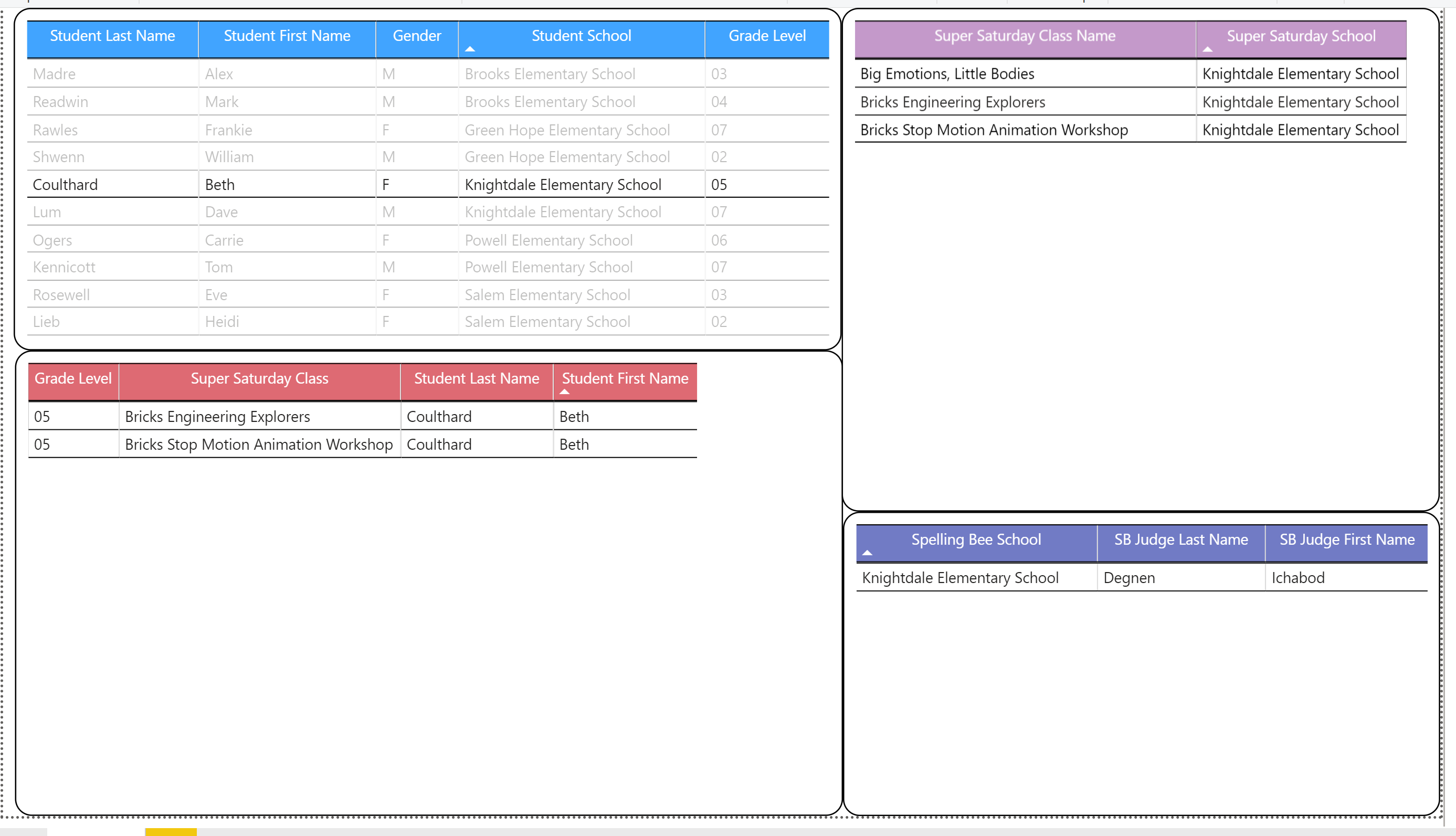
*PAGE Access Form for Super Saturday and Vendor information*

WDD also created a Report in Access to show PAGE how they can use the Create View statements to create reports for many different things. Since being able to contact parents was very important to PAGE, WDD chose to use the PaerentContactInfo view for the report.



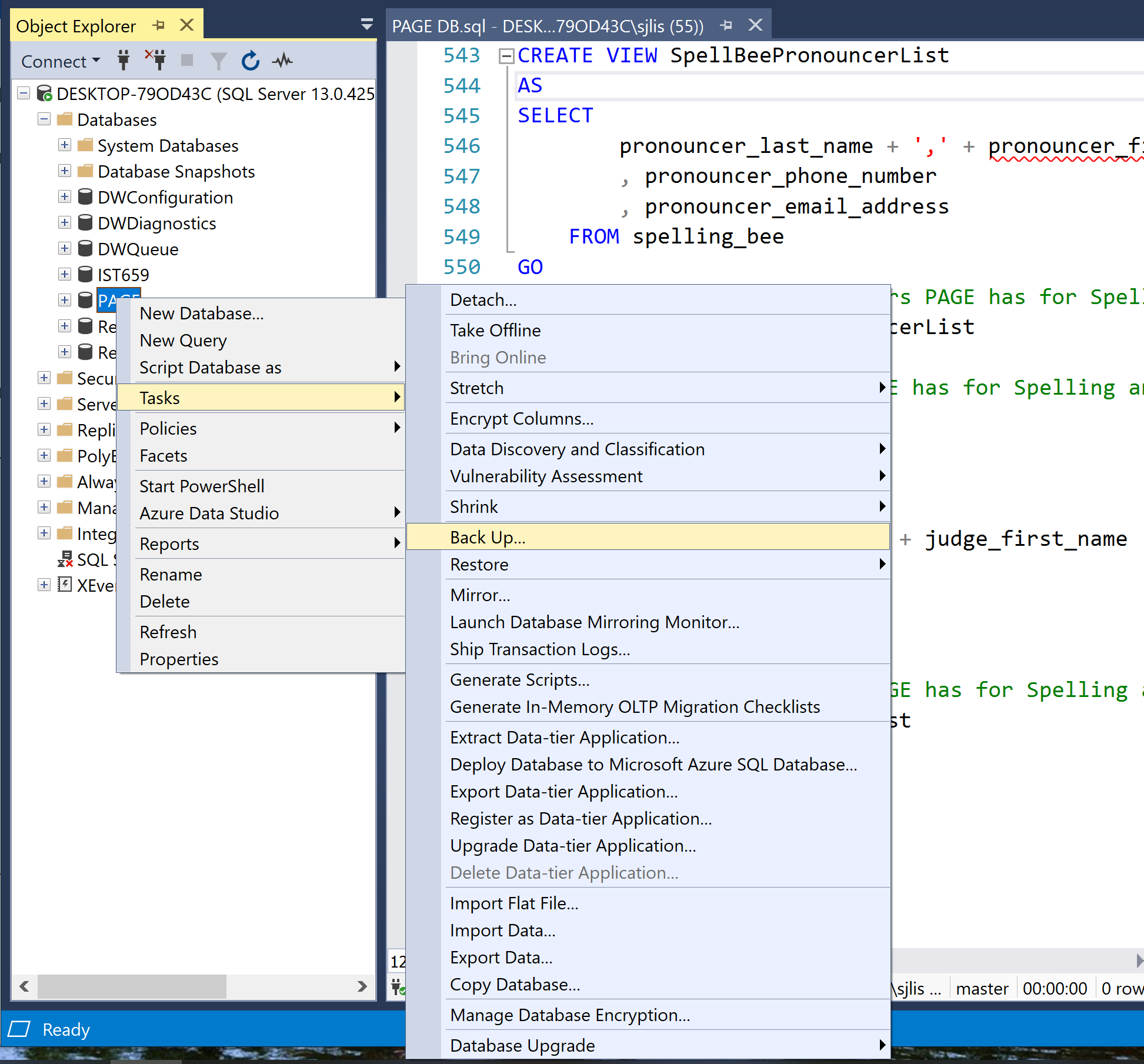
*PAGE Access Report for Parent information*

Access will also give PAGE the ability to add or delete values in the tables and since Access and SQL Server are linked it will update in SQL Server.

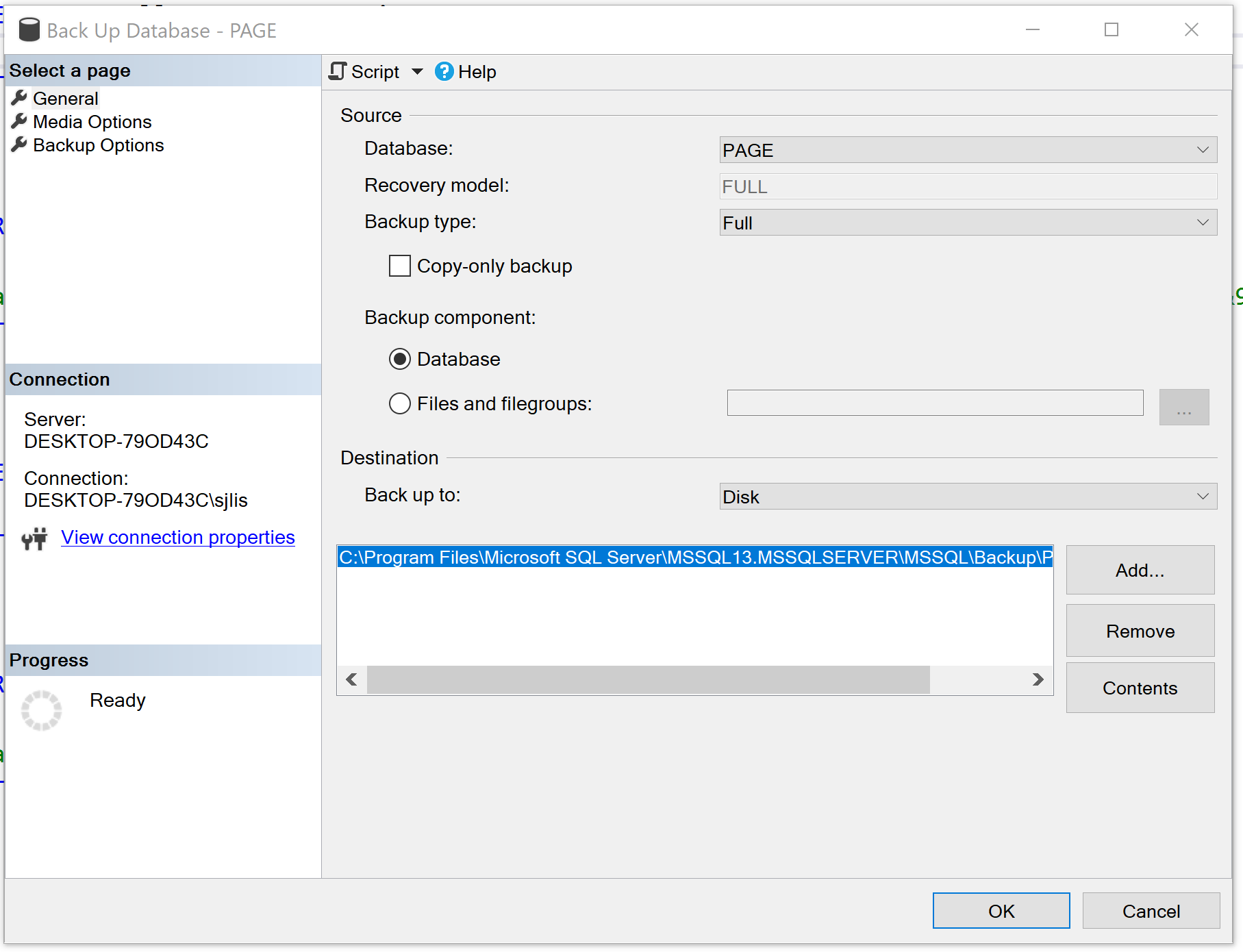
WDD also imported PAGE’s SQL Server database into Power BI. Power BI is Microsoft’s analytical software that many data scientists use to analyze a business’s data. WDD used it to show how PAGE could create charts on a dashboard and then filter through the data within that dashboard. WDD created four text tables, one showing students information, one showing thos same students with their Super Saturday classes, one showing which schools Super Saturday classes are taught at, and the last one showing which judge was at each schools spelling bee. WDD then filtered the information by clicking a student in the first chart which changed the other three. Both before and after filtering pictures showing how powerful Power BI would be for Page can be seen below. *PAGE Power BI dashboard*

*PAGE Power BI dashboard filtered on the student Beth Coulthard*

The final step is to work with PAGE on how often to backup the database. Since data is not entered or deleted on a consistent basis, WDD recommends backing up the database every time it is worked on. WDD has it set up that all that needs to be done to backup the database is to right click the PAGE database in the Object Explorer window, click on “Tasks,” then click on “Back Up…” This will take them to the window where WDD will have it set up to backup to the external hard drive that PAGE uses to backup QuickBooks. PAGE will just have to hit “Ok” and the database will then back up to their external hard drive. PAGE then disconnects the hard drive, like they do when they back up QuickBooks, and store it in the safe. This way PAGE’s information will always be available.



*PAGE’s SQL Server backup menu*



*PAGE’s SQL Server backup page*

SUMMARY and ANALIZATION

Based on analyzing the SQL Server database in Power BI there are some adjustments that WDD recommends the PAGE Board consider changing in their newly designed database. This will give the PAGE Board even better information about how the Spelling Bee and Super Saturday are doing. Now that PAGE is expanding its marketing for donations, a donation table collecting all of the donor’s information should be created and it should be linked to both the Spelling Bee table and Super Saturday table as donor could choose to donate to either cause. This table should collect all the needed information on the donor, including but not limited to:

* Donor ID
* Donor Business Name
* Donor Last Name
* Donor First Name
* Donor Address
* Donor Phone Number
* Donation Amount

With this information, PAGE will very easily be able to pull the needed information on the donors so that they can send out letters at the end of the year to all the donors for the year thanking them for their donations and giving the donors PAGE’s tax information.

On the Spelling Bee table, WDD recommends entering the first and last name of the winner, the winning word, students grade level and the year won. WDD also recommends creating a pronouncer table, moving the judges to the volunteer table, and then reformatting the columns in the Spelling Bee table. The Pronouncer table should have a one to many relationship to the Spelling Bee table as each spelling bee only has one pronouncer but each pronouncer can work for zero to many spelling bees. While WDD understands PAGE’s argument as to why they were not collecting this information for example, the same family has won the spelling bee for the last five years or that Scripps collects the winner’s information, WDD has more than one reason as to why PAGE should be collecting it. The first argument WDD presented is that this winning family’s children will eventually age out of being able to participate in the spelling bee and then PAGE could have different winners each year. Also, it would be good for PAGE to keep the winning word, winner’s school and grade level for each year so that PAGE can see which schools and grade levels are in the top five and what the winning words were. Also, the judges should be moved to the volunteer table as they could also volunteer to help with Super Saturday classes. The new columns should be as follows:

* Spelling Bee ID
* Year
* Winner Last Name
* Winner First Name
* Winner School
* Winner Grade Level
* Winning Word
* Pronouncer Last Name
* Pronouncer First Name
* Judge 1 Last Name
* Judge 1 First Name
* Judge 2 Last Name
* Judge 2 First Name
* Judge 3 Last Name
* Judge 3 First Name

With these changes, PAGE would be better able to track how many times a family’s child/ren win, which pronouncers and judges are more committed to the spelling bee and if there is a trend with the winning word difficulty.

On the Super Saturday Class table, WDD recommends entering columns in the Super Saturday Class table for the Super Saturday class fee and the vendor fee. While analyzing the data in Power BI, WDD could see which students took classes and what schools they are from but they could not analyze what classes brought in more money for PAGE. While WDD understands that PAGE does document this information in QuickBooks, if it was also brought into this database then PAGE could breakdown the total income by class and vendor. PAGE could take this information and see which classes bring in the most money, which classes are the most popular, and are those most popular classes bringing in the most money. Since PAGE is not incorporating that information into this database, it is missing out on key information.

PAGE of Wake Co. needs to be able to understand what is happening within their organization, as well as, be able to use that information to better serve the students, parents, teachers, schools, and businesses that support them. PAGE is a non-profit that serves the students of Wake County yet does not have a way to better understand students they are servicing and which students they are not reaching. By having a well-designed database PAGE will be able to better access the needed information to help them reach schools about Super Saturday, reach more parents about programs to help them understand their gifted child/ren, develop support systems for schools that are not being reached, and work with local business to get the message out to the community. Being a parent of a gifted child or children is a beautiful challenge and parents need all the help they can get in help them be there for their child/ren.

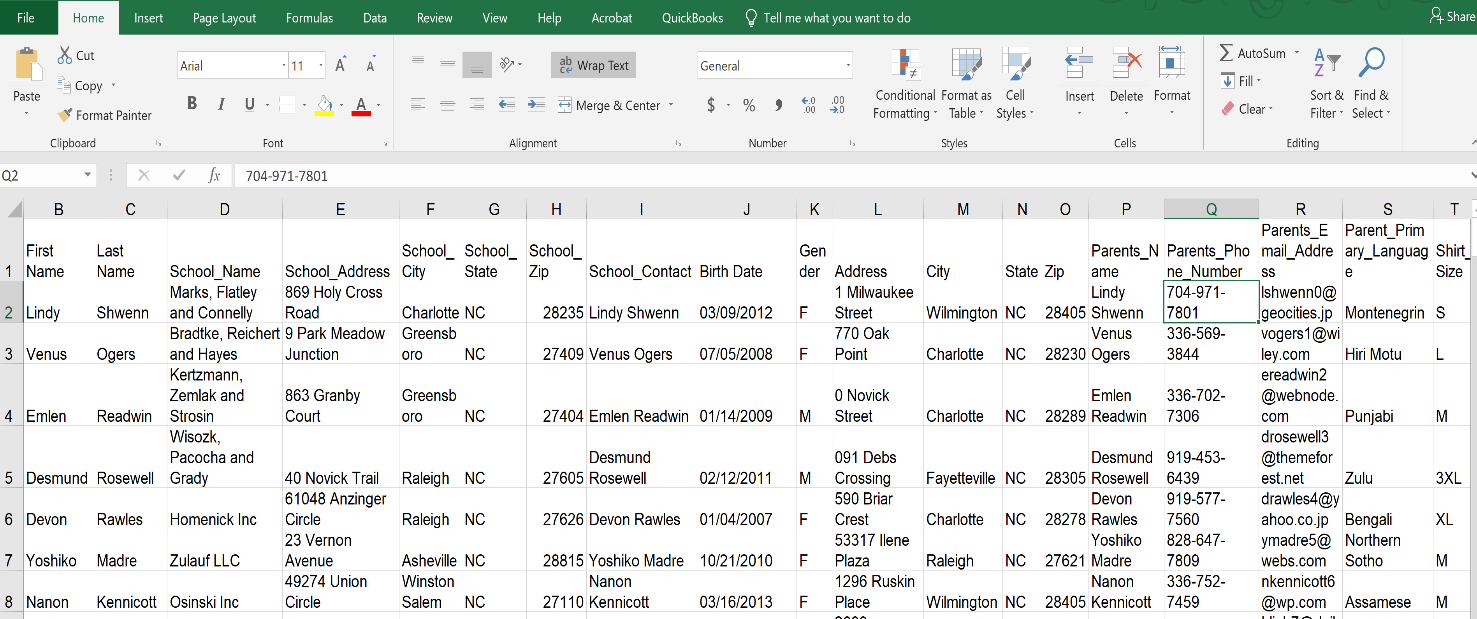
REFERENCES

PAGE: Partners for the Advancement of Gifted Education of Wake County, Retrieved from http://www.wakepage.org

TABLES, DIAGRAMS, and SQL CODE

Table 1

PAGE Spelling Bee Raw Data



Note: PAGE’s raw data collected by the Spelling Bee Coordinator

Table 2

PAGE Super Saturday Raw Data

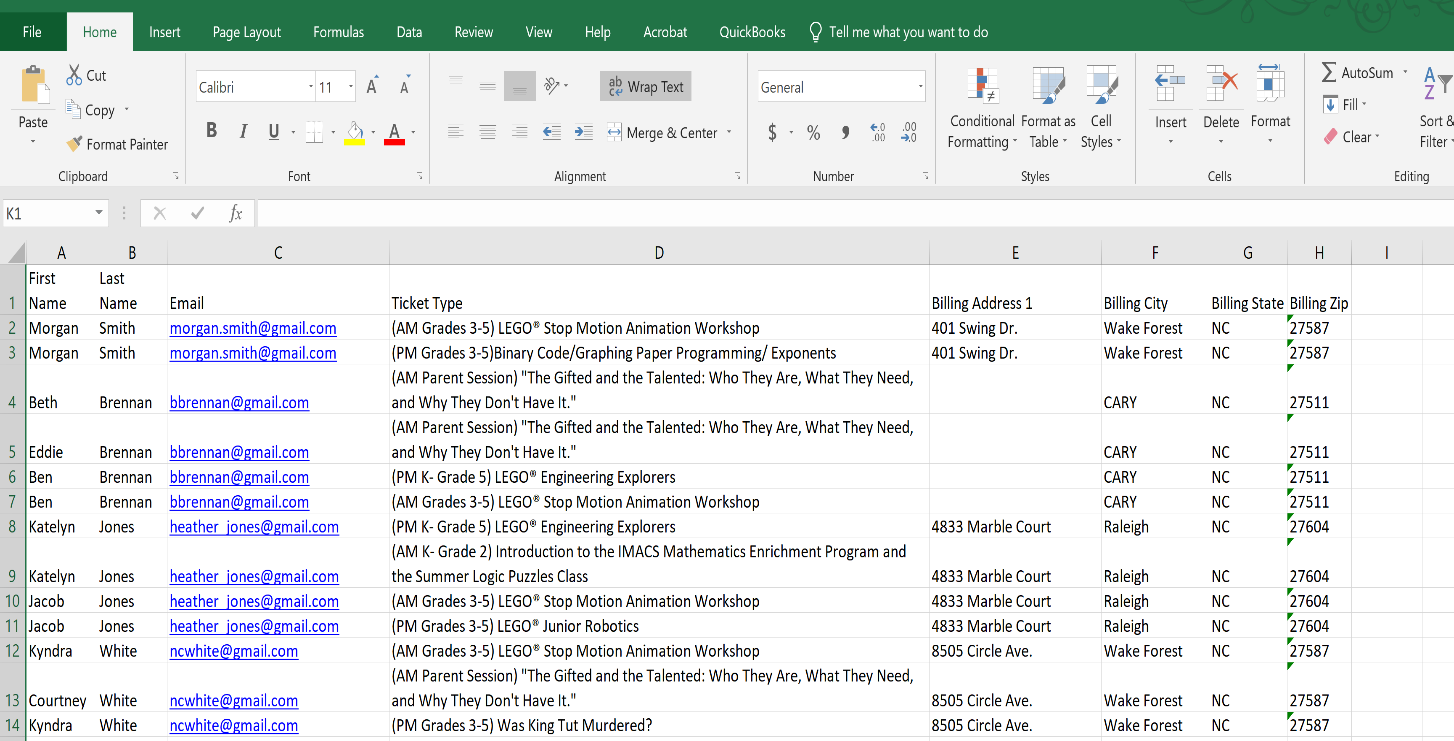
 Note: PAGE’s raw data collected by the Super Saturday Coordinator

Table 2

PAGE’s Business Rules

|  |  |
| --- | --- |
|  | **Rule Description:** |
| 1 | Each Student has one to many Parents |
| 2 | Each Parent has one to many Students |
| 3 | Each Student is assigned to one and only one School |
| 4 | Each School is assigned one to many Students |
| 5 | Each Student is assigned to one and only one Grade Level |
| 6 | Each Grade Level is assigned one to many Students |
| 7 | Each Student can sign up for zero or many Super Saturday classes |
| 8 | Each Super Saturday class can have zero to many Students |
| 9 | Each Student can particiate in zero to many Spelling Bees |
| 10 | Each Spelling Bee has one to many Students |
| 11 | Each Grade Level is assigned to one to many Super Saturday classes |
| 12 | Each Super Saturday class is assigned one to many Grade Levels |
| 13 | Each Super Saturday class has one and only one Vendor |
| 14 | Each Vendor can teach zero to many Supr Saturday classes |
| 15 | Each Super Saturday class needs one to many Volunteers |
| 16 | Each Volunteer can volunteer for zero to many Super Saturday classes |
| 17 | Each Volunteer can volunteer for zero to many Spelling Bees |
| 18 | Each Spelling Bee needs one to many Volunteers |

Note: PAGE’s business rules were established by the Board

Diagram 1

PAGE of Wake County Conceptual Database Model

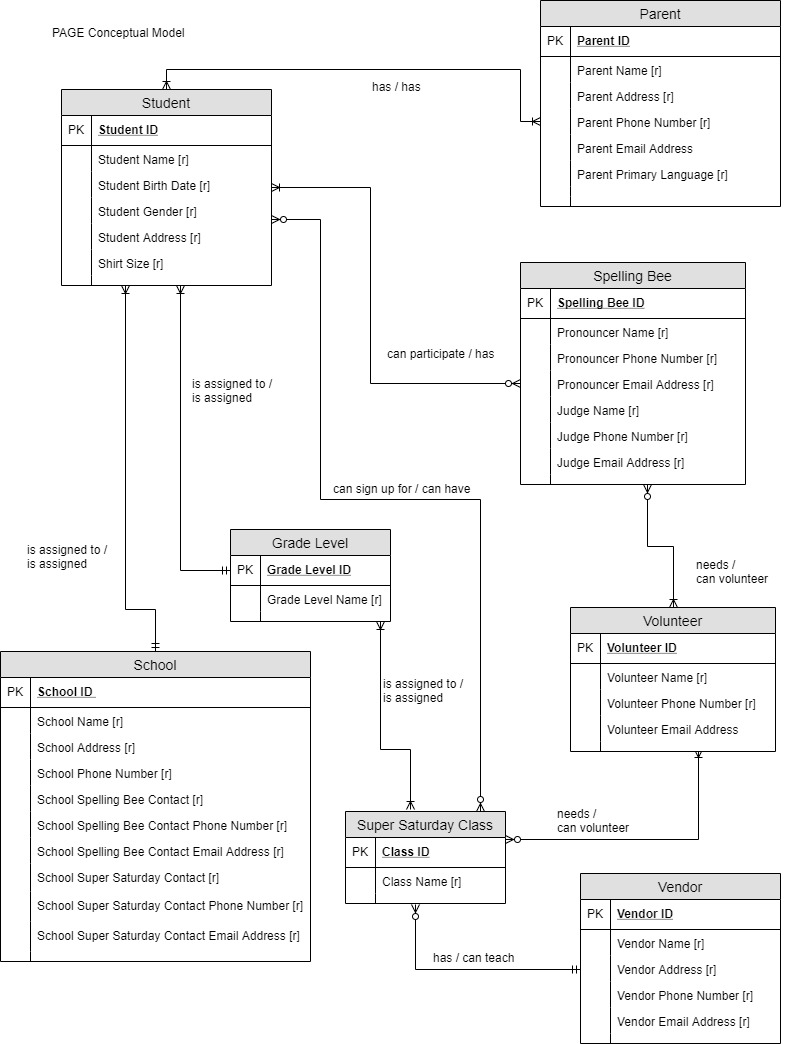


Figure 1. This is the Conceptual Database model of the data from both Super Saturday and Spelling Bee.

Diagram 2

PAGE of Wake County Logical Model

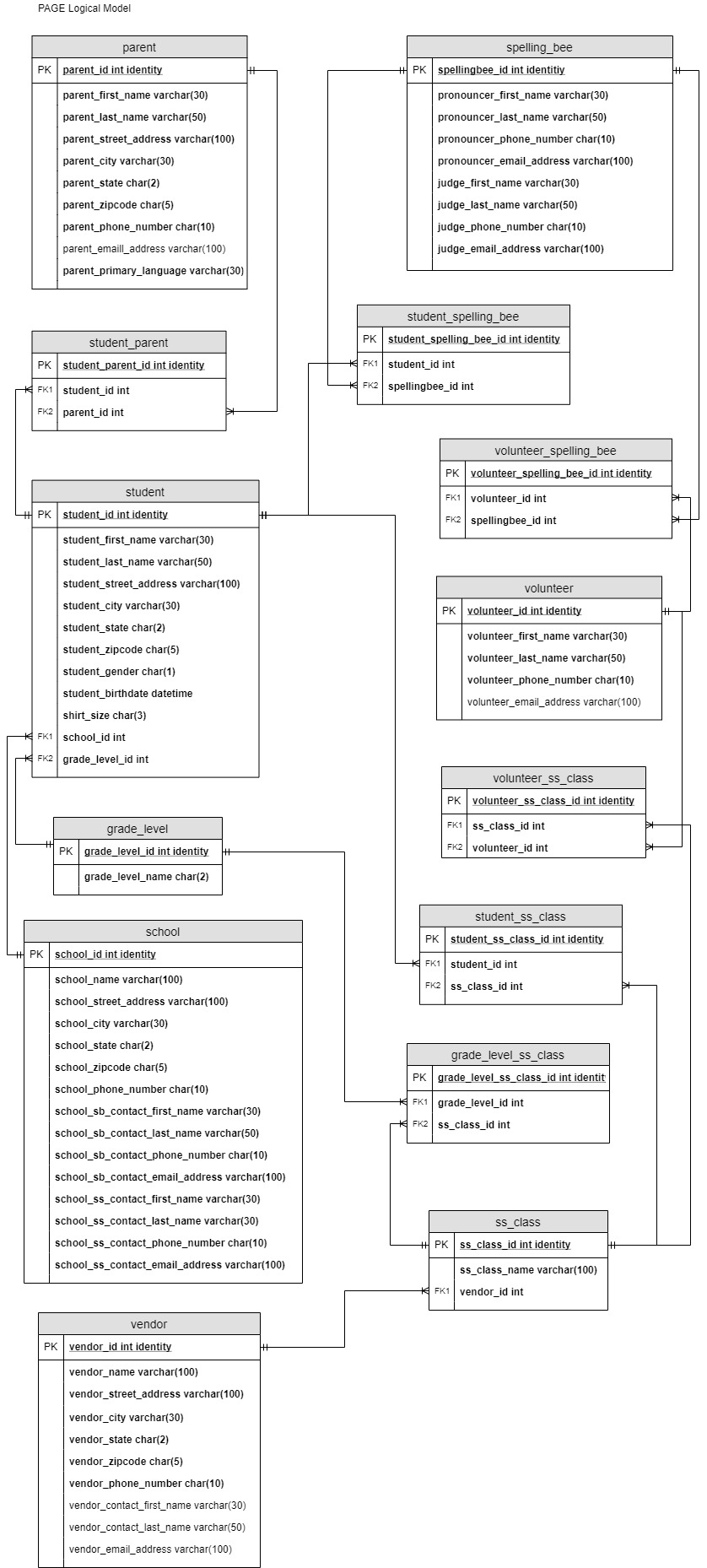


Figure 1. This is the Logical Database model of the data from both Super Saturday and Spelling Bee.

SQL Code 1

SQL Code for creating the tables of the PAGE of Wake County SQL Server Database

-- Create Parent Table

CREATE TABLE parent (

-- Columns for the parent table

parent\_id int identity,

parent\_first\_name varchar(30) not null,

parent\_last\_name varchar(50) not null,

parent\_street\_address varchar(100) not null,

parent\_city varchar(30) not null,

parent\_state char(2) not null,

parent\_zipcode char(5) not null,

parent\_phone\_number char(10) not null,

parent\_email\_address varchar(100),

parent\_primary\_language varchar(30) not null,

-- Constraints on Parent Table

CONSTRAINT PK\_parent PRIMARY KEY (parent\_id)

)

-- End creating Parent Table.

-- Create School Table

CREATE TABLE school (

-- Columns for the school table

school\_id int identity,

school\_name varchar(100) not null,

school\_street\_address varchar(100) not null,

school\_city varchar(30) not null,

school\_state char(2) not null,

school\_zipcode char(5) not null,

school\_phone\_number char(10) not null,

school\_sb\_contact\_first\_name varchar(30) not null,

school\_sb\_contact\_last\_name varchar(50) not null,

school\_sb\_contact\_phone\_number char(10) not null,

school\_sb\_contact\_email\_address varchar(100) not null,

school\_ss\_contact\_first\_name varchar(30) not null,

school\_ss\_contact\_last\_name varchar(50) not null,

school\_ss\_contact\_phone\_number char(10) not null,

school\_ss\_contact\_email\_address varchar(100) not null,

-- Constraints on School Table

CONSTRAINT PK\_school PRIMARY KEY (school\_id)

)

-- End creating School Table.

-- Create Vendor Table

CREATE TABLE vendor (

-- Columns for the vendor table

vendor\_id int identity,

vendor\_name varchar(100) not null,

vendor\_street\_address varchar(100) not null,

vendor\_city varchar(30) not null,

vendor\_state char(2) not null,

vendor\_zipcode char(5) not null,

vendor\_phone\_number char(10) not null,

vendor\_contact\_first\_name varchar(30),

vendor\_contact\_last\_name varchar(50),

vendor\_email\_address varchar(100),

-- Constraints on Vendor Table

CONSTRAINT PK\_vendor PRIMARY KEY (vendor\_id)

)

-- End creating Vendor Table.

-- Create Grade Level table

CREATE TABLE grade\_level (

-- Columns for the grade level table

grade\_level\_id int identity,

grade\_level\_name char(2),

-- Constraints on Grade Level Table

CONSTRAINT PK\_grade\_level PRIMARY KEY (grade\_level\_id)

)

-- End creating Grade Level Table

-- Create Spelling Bee Table

CREATE TABLE spelling\_bee (

-- Columns for the spelling bee table

spellingbee\_id int identity,

pronounder\_first\_name varchar(30) not null,

pronouncer\_last\_name varchar(50) not null,

pronouncer\_phone\_number char(10) not null,

pronouncer\_email\_address varchar(100) not null,

judge\_first\_name varchar(30) not null,

judge\_last\_name varchar(50) not null,

judge\_phone\_number char(10) not null,

judge\_email\_address varchar(100) not null,

-- Constraints on Vendor Table

CONSTRAINT PK\_spelling\_bee PRIMARY KEY (spellingbee\_id)

)

-- End creating Spelling Bee Table.

/\* To correct column name pronounder\_first\_name varchar(30) not null,

I right clicked the column under Object Explorer, then corrected

the coulmn name to pronouncer\_first\_name varchar(30) not null,

\*/

-- Create Volunteer Table

CREATE TABLE volunteer (

-- Columns for the volunteer table

volunteer\_id int identity,

volunteer\_first\_name varchar(30) not null,

volunteer\_last\_name varchar(50) not null,

volunteer\_phone\_number char(10) not null,

volunteer\_email\_address varchar(100),

-- Constraints on Volunteer Table

CONSTRAINT PK\_volunteer PRIMARY KEY (volunteer\_id)

)

-- End creating Volunteer Table.

-- Create Student Table

CREATE TABLE student (

-- Columns for the student table

student\_id int identity,

student\_first\_name varchar(30) not null,

student\_last\_name varchar(50) not null,

student\_street\_address varchar(100) not null,

student\_city varchar(30) not null,

student\_state char(2) not null,

student\_zipcode char(5) not null,

student\_gender char(1) not null,

student\_birthdate DATETIME not null,

shirt\_size char(3) not null,

school\_id int not null,

grade\_level\_id int not null,

-- Constraints on Student Table

CONSTRAINT PK\_student PRIMARY KEY (student\_id),

CONSTRAINT FK1\_student FOREIGN KEY (school\_id) REFERENCES school(school\_id),

CONSTRAINT FK2\_student FOREIGN KEY (grade\_level\_id) REFERENCES grade\_level(grade\_level\_id),

)

-- End creating Student Table.

-- Create Student Parent Table

CREATE TABLE student\_parent (

-- Columns for the student parent table

student\_parent\_id int identity,

student\_id int not null,

parent\_id int not null,

-- Constraints on Student Parent Table

CONSTRAINT PK\_student\_parent PRIMARY KEY (student\_parent\_id),

CONSTRAINT FK1\_student\_parent FOREIGN KEY (student\_id) REFERENCES student(student\_id),

CONSTRAINT FK2\_student\_parent FOREIGN KEY (parent\_id) REFERENCES parent(parent\_id),

)

-- End creating Student Parent Table.

-- Create Student Spelling Bee Table

CREATE TABLE student\_spelling\_bee (

-- Columns for the student spelling bee table

student\_spelling\_bee\_id int identity,

student\_id int not null,

spellingbee\_id int not null,

-- Constraints on Student Spelling Bee Table

CONSTRAINT PK\_student\_spelling\_bee PRIMARY KEY (student\_spelling\_bee\_id),

CONSTRAINT FK1\_student\_spelling\_bee FOREIGN KEY (student\_id) REFERENCES student(student\_id),

CONSTRAINT FK2\_student\_spelling\_bee FOREIGN KEY (spellingbee\_id) REFERENCES spelling\_bee(spellingbee\_id),

)

-- End creating Student Spelling Bee Table.

-- Create Volunteer Spelling Bee Table

CREATE TABLE volunteer\_spelling\_bee (

-- Columns for the volunteer spelling bee table

volunteer\_spelling\_bee\_id int identity,

volunteer\_id int not null,

spellingbee\_id int not null,

-- Constraints on Volunteer Spelling Bee Table

CONSTRAINT PK\_volunteer\_spelling\_bee PRIMARY KEY (volunteer\_spelling\_bee\_id),

CONSTRAINT FK1\_volunteer\_spelling\_bee FOREIGN KEY (volunteer\_id) REFERENCES volunteer(volunteer\_id),

CONSTRAINT FK2\_volunteer\_spelling\_bee FOREIGN KEY (spellingbee\_id) REFERENCES spelling\_bee(spellingbee\_id),

)

-- End creating Student Spelling Bee Table.

-- Create Super Saturday Class Table

CREATE TABLE ss\_class (

-- Columns for the super saturday class table

ss\_class\_id int identity,

ss\_class\_name varchar(100) not null,

vendor\_id int not null,

-- Constraints on Super Saturday Class Table

CONSTRAINT PK\_ss\_class PRIMARY KEY (ss\_class\_id),

CONSTRAINT FK1\_ss\_class FOREIGN KEY (vendor\_id) REFERENCES vendor(vendor\_id),

)

-- End creating Super Saturday Class Table.

-- Create Grade Level Super Saturday Class Table

CREATE TABLE grade\_level\_ss\_class (

-- Columns for the Grade Level Super Saturday Class table

grade\_level\_ss\_class\_id int identity,

grade\_level\_id int not null,

ss\_class\_id int not null,

-- Constraints on Grade Level Super Saturday Class Table

CONSTRAINT PK\_grade\_level\_ss\_class PRIMARY KEY (grade\_level\_ss\_class\_id),

CONSTRAINT FK1\_grade\_level\_ss\_class FOREIGN KEY (grade\_level\_id) REFERENCES grade\_level(grade\_level\_id),

CONSTRAINT FK2\_grade\_level\_ss\_class FOREIGN KEY (ss\_class\_id) REFERENCES ss\_class(ss\_class\_id),

)

-- End creating Grade Level Super Saturday Class Table.

-- Create Student Super Saturday Class Table

CREATE TABLE student\_ss\_class (

-- Columns for the Student Super Saturday Class table

student\_ss\_class\_id int identity,

student\_id int not null,

ss\_class\_id int not null,

-- Constraints on Student Super Saturday Class Table

CONSTRAINT PK\_student\_ss\_class PRIMARY KEY (student\_ss\_class\_id),

CONSTRAINT FK1\_student\_ss\_class FOREIGN KEY (student\_id) REFERENCES student(student\_id),

CONSTRAINT FK2\_student\_ss\_class FOREIGN KEY (ss\_class\_id) REFERENCES ss\_class(ss\_class\_id),

)

-- End creating Student Super Saturday Class Table.

-- Create Volunteer Super Saturday Class Table

CREATE TABLE volunteer\_ss\_class (

-- Columns for the Volunteer Super Saturday Class table

volunteer\_ss\_class\_id int identity,

ss\_class\_id int not null,

volunteer\_id int not null,

-- Constraints on Volunteer Super Saturday Class Table

CONSTRAINT PK\_volunteer\_ss\_class PRIMARY KEY (volunteer\_ss\_class\_id),

CONSTRAINT FK1\_volunteer\_ss\_class FOREIGN KEY (ss\_class\_id) REFERENCES ss\_class(ss\_class\_id),

CONSTRAINT FK2\_volunteer\_ss\_class FOREIGN KEY (volunteer\_id) REFERENCES volunteer(volunteer\_id),

)

-- End creating Volunteer Super Saturday Class Table.

SQL Code 2

SQL Code for inserting data into the tables of the PAGE of Wake County SQL Server Database

-- Adding data into Parent Table

INSERT INTO parent(parent\_first\_name, parent\_last\_name, parent\_street\_address, parent\_city, parent\_state, parent\_zipcode

, parent\_phone\_number, parent\_email\_address, parent\_primary\_language)

VALUES

('Lindy', 'Shwenn', '1 Milwaukee Street', 'Cary', 'NC', '27519', '9199717801', 'lshwenn0@geocities.jp', 'English'),

('Venus', 'Ogers', '770 Oak Point', 'Raleigh', 'NC', '27506', '9195693844', 'vogers1@wiley.com', 'Spanish'),

('Emlen', 'Readwin', '212 Novick Street', 'Raleigh', 'NC', '25706', '9197027306', 'ereadwin2@webnode.com', 'English'),

('Desmund', 'Rosewell', '91 Debs Crossing', 'Apex', 'NC', '27511', '9194536439', 'drosewell3@themeforest.net', 'English'),

('Devon', 'Rawles', '590 Briar Crest Terrace', 'Morrisville', 'NC', '27560', '9195777560', 'drawles4@yahoo.co.jp', 'Hindi'),

('Jesus', 'Madre', '53317 Ilene Plaza', 'Raleigh', 'NC', '27509', '9196477809', 'jmadre5@webs.com', 'Spanish'),

('Nan', 'Kennicott', '1296 Ruskin Place', 'Cary', 'NC', '25719', '9197527459', 'nkennicott6@wp.com', 'Mandarin'),

('Henry', 'Lieb', '2683 Tennessee Court', 'Apex', 'NC', '27511', '9193335063', 'hlieb7@dailymotion.com', 'English'),

('Opal', 'Coulthard', '48073 Crowley Way', 'Wake Forest', 'NC', '25709', '9191403281', 'ocoulthard8@hc360.com', 'English'),

('Eveleen', 'Lum', '270 Quincy Court', 'Knightdale', 'NC', '27580', '9196047295', 'elum9@tumblr.com', 'English')

-- Examine Parent table

SELECT \* FROM parent

-- Adding data into School Table

INSERT INTO school(school\_name, school\_street\_address, school\_city, school\_state, school\_zipcode, school\_phone\_number

, school\_sb\_contact\_first\_name, school\_sb\_contact\_last\_name, school\_sb\_contact\_phone\_number, school\_sb\_contact\_email\_address

, school\_ss\_contact\_first\_name, school\_ss\_contact\_last\_name, school\_ss\_contact\_phone\_number, school\_ss\_contact\_email\_address)

VALUES

('Brooks Elementary School', '53 Troy Pass', 'Raleigh', 'NC', '25706', '9195357600'

, 'Yulma', 'Monson', '9195357696', 'ymonsonn@goo.ne.jp', 'Merridie', 'Hutcheons', '9195357655', 'mhutcheonso@bravesites.com'),

('Powell Elementary School', '15362 Hooker Way', 'Raleigh', 'NC', '27516', '9197179800'

, 'Ric', 'Lelliott', '9197179812', 'rlelliottp@ocn.ne.jp', 'Benetta', 'Boyton', '9197179820', 'bboytonq@fc2.com'),

('Salem Elementary School', '24627 Eastlawn Center', 'Apex', 'NC', '27511', '9196522600'

, 'Neils', 'Thomazet', '9196522650', 'nthomazetr@dailymotion.com', 'Elwood', 'Garbert', '9196522645', 'egarberts@sitemeter.com'),

('Green Hope Elementary School', '14 Dapin Drive', 'Cary', 'NC', '27519', '9198487500'

, 'Philippe', 'Janauschek', '9198487515', 'pjanauschekt@utexas.edu', 'Lew', 'Horley', '9198487526', 'lhorleyu@sohu.com'),

('Knightdale Elementary School', '187 Lyons Junction', 'Knightdale', 'NC', '27580', '9196584500'

, 'Danell', 'April', '9196584512', 'daprilv@privacy.gov.au', 'Charissa', 'Vivers', '9196584585', 'cviversw@webeden.co.uk')

-- Examine School table

SELECT \* FROM school

-- Adding data into Vendor Table

INSERT INTO vendor(vendor\_name, vendor\_street\_address, vendor\_city, vendor\_state, vendor\_zipcode, vendor\_phone\_number

, vendor\_contact\_first\_name, vendor\_contact\_last\_name, vendor\_email\_address)

VALUES

('Armstrong and Sons', '851 Northwestern Alley', 'Raleigh', 'NC', '27509', '9193536159', 'Hunter', 'Kingswell', 'hkingswell14@amazon.de'),

('Schulist, Kozey and Mitchell', '762 Schmedeman Hill', 'Raleigh', 'NC', '25706', '9196542653', 'Anna', 'Gruszka', 'agruszka15@domainmarket.com'),

('Lakin, Smith and Treutel', '50324 Farragut Road', 'Cary', 'NC', '27511', '9197652356', 'Abbot', 'Oswick', 'aoswick16@unesco.org'),

('Parker, Hegmann and Breitenberg', '71 Jay Center', 'Raleigh', 'NC', '27506', '9192643568', 'Normay', 'Hare', 'nhare17@accuweather.com'),

('Jacobson, Grimes and Fadel', '34577 Butternut Place', 'Cary', 'NC', '27519', '9196549587', 'Billi', 'Harriott', 'bharriott18@myspace.com')

-- Examine Vendor table

SELECT \* FROM vendor

-- Adding data into Grade Level Table

INSERT INTO grade\_level(grade\_level\_name)

VALUES

('00'), ('01'), ('02'), ('03'), ('04'), ('05'), ('06'), ('07'), ('08'), ('09'), ('10'), ('11'), ('12')

-- Examine Grade Level table

SELECT \* FROM grade\_level

-- Adding data into Spelling Bee table

INSERT INTO spelling\_bee(pronounder\_first\_name, pronouncer\_last\_name, pronouncer\_phone\_number, pronouncer\_email\_address, judge\_first\_name

, judge\_last\_name, judge\_phone\_number, judge\_email\_address)

VALUES

('Joscelin', 'Martinuzzi', '9197640528', 'jmartinuzzi19@ftc.gov', 'Kevyn', 'Motherwell', '3369902143', 'kmotherwell1a@altervista.org'),

('Hermione', 'Penna', '9104465981', 'hpenna1b@cnet.com', 'Demetris', 'Snoad', '9198052831', 'dsnoad1c@webeden.co'),

('Leslie', 'Wharton', '9199406682', 'lwharton1d@1688.com', 'Ichabod', 'Degnen', '9193766811', 'idegnen1e@hubpages.com'),

('Packston', 'Frisch', '9197908054', 'pfrisch1f@g.co', 'Theobald', 'Hatt', '9102155243', 'thatt1g@icq.com'),

('Barrett', 'Newrick', '9197969774', 'bnewrick1h@hud.gov', 'Paige', 'Rutherford', '9191111283', 'prutherford1i@reverbnation.com')

-- Examine Spelling Bee table

SELECT \* FROM spelling\_bee

-- Adding data into Volunteer table

INSERT INTO volunteer(volunteer\_first\_name, volunteer\_last\_name, volunteer\_phone\_number, volunteer\_email\_address)

VALUES

('Ferdinand', 'Godlee', '9194158919', 'fgodlee1j@a8.net'),

('Deana', 'Cullen', '9194356357', 'dcullen1k@oakley.com'),

('Daveta', 'Scyone', '9198825494', 'dscyone1l@ed.gov'),

('Susette', 'Crab', '9197455420', 'scrab1m@bing.com'),

('Caria', 'Hassan', '9193980110', 'chassan1n@domainmarket.com'),

('Manfred', 'Endersby', '9198132223', 'mendersby1o@china.com.cn'),

('Devlen', 'Elford', '9196073733', 'delford1p@vistaprint.com'),

('Mandel', 'Holcroft', '9191558686', 'mholcroft1q@newsvine.com'),

('Delmer', 'Mariot', '9191228775', 'dmariot1r@bravesites.com'),

('Beulah', 'Kull', '9191287476', 'bkull1s@typepad.com')

-- Examine Volunteer table

SELECT \* FROM volunteer

-- Enter extra data into volunteer table

INSERT INTO volunteer(volunteer\_first\_name, volunteer\_last\_name, volunteer\_phone\_number, volunteer\_email\_address)

VALUES

('Opal', 'Coulthard', '9191403281', 'ocoulthard8@hc360.com'),

('Eveleen', 'Lum', '9196047295', 'elum9@tumblr.com')

-- Examine Volunteer table

SELECT \* FROM volunteer

-- Adding data into Student table

INSERT INTO student(student\_first\_name, student\_last\_name, student\_street\_address, student\_city, student\_state

, student\_zipcode, student\_gender, student\_birthdate, shirt\_size, school\_id, grade\_level\_id)

VALUES

('William', 'Shwenn', '1 Milwaukee Street', 'Cary', 'NC', '27519', 'M', '03/09/2012', 'YS', '4', '03'),

('Carrie', 'Ogers', '770 Oak Point', 'Raleigh', 'NC', '27506', 'F', '07/05/2008', 'YL', '2', '07'),

('Mark', 'Readwin', '212 Novick Street', 'Raleigh', 'NC', '25706', 'M', '01/14/2009', 'YM', '1', '05'),

('Eve', 'Rosewell', '91 Debs Crossing', 'Apex', 'NC', '27511', 'F', '02/12/2011', 'YM', '3', '04'),

('Frankie', 'Rawles', '590 Briar Crest Terrace', 'Morrisville', 'NC', '27560', 'F', '01/04/2007', 'AS', '4', '08'),

('Alex', 'Madre', '53317 Ilene Plaza', 'Raleigh', 'NC', '27509', 'M', '10/21/2010', 'AS', '1', '04'),

('Tom', 'Kennicott', '1296 Ruskin Place', 'Cary', 'NC', '25719', 'M', '03/16/2007', 'YM', '2', '08'),

('Heidi', 'Lieb', '2683 Tennessee Court', 'Apex', 'NC', '27511', 'F', '06/11/2011', 'YS', '3', '03'),

('Beth', 'Coulthard', '48073 Crowley Way', 'Wake Forest', 'NC', '25709', 'F', '07/08/2009', 'YL', '5', '06'),

('Dave', 'Lum', '270 Quincy Court', 'Knightdale', 'NC', '27580', 'M', '07/08/2007', 'AS', '5', '08')

-- Examine Student table

SELECT \* FROM student

-- Adding data into Super Saturday table

INSERT INTO ss\_class(ss\_class\_name, vendor\_id)

VALUES

('Bricks Stop Motion Animation Workshop', '1'),

('Bricks Engineering Explorers', '1'),

('Binary Code', '4'),

('Optical Illusion', '5'),

('Electronic Piano & Circuit Bending', '2'),

('Introduction to the IMACS Mathematics Enrichment Program', '2'),

('Bricks Junior Robotics', '3'),

('Low Cost Stethoscope/ Hydraulic Hand and Arm', '3'),

('Big Emotions, Little Bodies', '4'),

('M&M Counting Fun/Estimation Station and Exploring Sound', '5')

-- Examine Super Saturday table

SELECT \* FROM ss\_class

-- Adding data into Student Parent table

INSERT INTO student\_parent(student\_id, parent\_id)

VALUES

('1', '1'), ('2', '2'), ('3', '3'), ('4', '4')

, ('5', '5'), ('6', '6'), ('7', '7')

, ('8', '8'), ('9', '9'), ('10', '10')

-- Examine Student Parent table

SELECT \* FROM student\_parent

-- Adding data into Student Parent table

INSERT INTO student\_spelling\_bee(student\_id, spellingbee\_id)

VALUES

('1', '2'), ('2', '5'), ('3', '2'), ('4', '1')

, ('5', '3'), ('6', '4'), ('7', '5')

, ('8', '1'), ('9', '3'), ('10', '4')

-- Examine Student Spelling Bee table

SELECT \* FROM student\_spelling\_bee

-- Adding data into Volunteer Spelling Bee table

INSERT INTO volunteer\_spelling\_bee(volunteer\_id, spellingbee\_id)

VALUES

('1', '1'), ('4', '1'), ('7', '2'), ('9', '2')

, ('10', '3'), ('5', '3'), ('2', '4')

, ('3', '4'), ('6', '5'), ('8', '5')

-- Examine Volunteer Spelling Bee table

SELECT \* FROM volunteer\_spelling\_bee

-- Adding data into Grade Level Super Satuerday Class table

INSERT INTO grade\_level\_ss\_class(grade\_level\_id, ss\_class\_id)

VALUES

('1', '2'), ('1', '6'), ('1', '10'), ('2', '2')

, ('2', '6'), ('2', '10'), ('3', '2'), ('3', '6')

, ('3', '10'), ('4', '1'), ('4', '2'), ('4', '3')

, ('4', '4'), ('4', '7'), ('4', '8'), ('5', '1')

, ('5', '2'), ('5', '3'), ('5', '4'), ('5', '7')

, ('5', '8'), ('6', '1'), ('6', '2'), ('6', '3')

, ('6', '4'), ('6', '7'), ('6', '8'), ('7', '5')

, ('7', '9'), ('8', '5'), ('8', '9'), ('9', '5')

, ('9', '9')

-- Examine Grade Level Super Saturday table

SELECT \* FROM grade\_level\_ss\_class

-- Adding data into Student Super Saturday table

INSERT INTO student\_ss\_class(student\_id, ss\_class\_id)

VALUES

('1', '2'), ('1', '6'), ('3', '1'), ('3', '4')

, ('4', '4'), ('4', '7'), ('5', '9'), ('7', '5')

, ('7', '9'), ('8', '1'), ('8', '10'), ('9', '1')

, ('9', '2'), ('10', '9')

-- Examine Student Super Saturday Class table

SELECT \* FROM student\_ss\_class

-- Adding data into Volunteer Super Saturday Class table

INSERT INTO volunteer\_ss\_class (ss\_class\_id, volunteer\_id)

VALUES

('1', '1'), ('2', '2'), ('3', '3'), ('4', '4')

, ('5', '5'), ('6', '6'), ('7', '7')

, ('8', '8'), ('9', '9'), ('10', '10')

-- Examine Volunteer Super Saturday Class table

SELECT \* FROM volunteer\_ss\_class

SQL Code 3

SQL Code for querying data from the tables of the PAGE of Wake County SQL Server Database

-- To create Super Saturday and Spelling Bee school contact list (for Question 1 and 2)

GO

CREATE VIEW SBandSSContacts

AS

SELECT

school\_name AS SchoolName

, school\_sb\_contact\_last\_name + ',' + school\_sb\_contact\_first\_name AS SpellBName

, school\_sb\_contact\_phone\_number AS SpellBNumber

, school\_sb\_contact\_email\_address AS SpellBEmail

, school\_ss\_contact\_last\_name + ',' + school\_ss\_contact\_first\_name AS SupSatName

, school\_ss\_contact\_phone\_number AS SupSatNumber

, school\_ss\_contact\_email\_address AS SupSatEmail

FROM school

GO

-- Examine Create View SBandSSContacts

SELECT \* FROM SBandSSContacts

-- To create Parent contact list (for Question 3)

GO

CREATE VIEW ParentContactInfo

AS

SELECT

parent\_last\_name + ',' + parent\_first\_name AS ParentName

, parent\_street\_address + parent\_city + parent\_state + parent\_zipcode AS ParentAddress

, parent\_phone\_number AS ParentNumber

, parent\_email\_address AS ParentEmail

FROM parent

GO

-- Drop ParentContactView due to errors

DROP VIEW ParentContactInfo

-- To create Corrected Parent contact list (for Question 3)

GO

CREATE VIEW ParentContactInfo

AS

SELECT

parent\_last\_name + ',' + parent\_first\_name AS ParentName

, parent\_street\_address + ',' + parent\_city + ',' + parent\_state + ' ' + parent\_zipcode AS ParentAddress

, parent\_phone\_number AS ParentNumber

, parent\_email\_address AS ParentEmail

FROM parent

GO

-- Examine Parent Contact List

SELECT \* FROM ParentContactInfo

-- To see if parents are also willing to volunteer for PAGE (for Question 4)

GO

CREATE VIEW ParentVolunteerList

AS

SELECT

parent\_last\_name + ',' + parent\_first\_name AS ParentName

, volunteer\_last\_name + ',' + volunteer\_first\_name AS VolunteerName FROM parent, volunteer

WHERE parent\_last\_name = volunteer\_last\_name

GO

-- Examine how many parent also volunteer

SELECT \* FROM ParentVolunteerList

-- To see if schools have both a Spelling Bee and have students in Super Saturday classes (Question 5)

GO

CREATE VIEW SpellingBeeandSupSatClassSchools

AS

SELECT

student\_last\_name + ',' + student\_first\_name AS StudetnName

, school\_name AS School

, ss\_class\_name AS SSClassName

FROM school

INNER JOIN student ON school.school\_id = student.school\_id

INNER JOIN ss\_class

INNER JOIN student\_ss\_class ON ss\_class.ss\_class\_id = student\_ss\_class.ss\_class\_id

ON student.student\_id = student\_ss\_class.student\_id

GO

DROP VIEW SpellingBeeandSupSatClassSchools

-- To see if schools have both a Spelling Bee and have students in Super Saturday classes (Question 5)

GO

CREATE VIEW SpellingBeeandSupSatClassSchools

AS

SELECT

student\_last\_name + ',' + student\_first\_name AS StudentName

, school\_name AS School

, ss\_class\_name AS SSClassName

FROM school

INNER JOIN student ON school.school\_id = student.school\_id

INNER JOIN ss\_class

INNER JOIN student\_ss\_class ON ss\_class.ss\_class\_id = student\_ss\_class.ss\_class\_id

ON student.student\_id = student\_ss\_class.student\_id

GO

-- Examine schools that have Spelling Bee and Sup. Sat. Classes

SELECT \* FROM SpellingBeeandSupSatClassSchools

ORDER BY school

-- To see which students are participating in Super Saturday classes (Question 6).

GO

CREATE VIEW StudentsandSupSatClasses

AS

SELECT

student\_last\_name + ',' + student\_first\_name AS StudentName

, ss\_class\_name AS SSClassName

FROM student

LEFT JOIN student\_ss\_class ON student.student\_id = student\_ss\_class.student\_id

LEFT JOIN ss\_class ON ss\_class.ss\_class\_id = student\_ss\_class.ss\_class\_id

GO

-- Examine which students are participating in Super Saturday classes.

SELECT \* FROM StudentsandSupSatClasses

-- To see which type of classes PAGE is offering on Super Saturday (Question 7)

GO

CREATE VIEW SupSatClassList

AS

SELECT

ss\_class\_name AS SupSatClassList

FROM ss\_class

GO

-- Examine Super Saturday Class list to see the types of classes offered

SELECT \* FROM SupSatClassList

-- To count how many pronouncers are in the database

GO

CREATE FUNCTION dbo.pronouncercount(@pronouncer\_last\_name varchar(30))

RETURNS int AS -- COUNT() is an integer value, so return it as an int

BEGIN

DECLARE @returnValue varchar(30) -- matches the function's return type

SELECT @returnValue = COUNT(pronouncer\_last\_name) FROM spelling\_bee

WHERE spelling\_bee.pronouncer\_last\_name = @pronouncer\_last\_name

RETURN @returnValue

END

GO

-- Function did work the way it was needed, deleted it.

DROP FUNCTION dbo.pronouncercount

-- To see how many pronouncers PAGE has for Spelling and their contact info (Question 8&9).

GO

CREATE VIEW SpellBeePronouncerList

AS

SELECT

pronouncer\_last\_name + ',' + pronouncer\_first\_name AS PronouncerName

, pronouncer\_phone\_number AS PronouncerNumber

, pronouncer\_email\_address AS PronouncerEmail

FROM spelling\_bee

GO

-- Examine how many pronouncers PAGE has for Spelling and their contact info (Question 8&9).

SELECT \* FROM SpellBeePronouncerList

-- To see how many judges PAGE has for Spelling and their contact info (Question 8&9).

GO

CREATE VIEW SpellBeeJudgeList

AS

SELECT

judge\_last\_name + ',' + judge\_first\_name AS JudgeName

, judge\_phone\_number AS JudgeNumber

, judge\_email\_address AS JudgeEmail

FROM spelling\_bee

GO

-- Examine how many judges PAGE has for Spelling and thier contact info (Question 8&9).

SELECT \* FROM SpellBeeJudgeList

-- To create volunteer list.

GO

CREATE VIEW VolunteerList

AS

SELECT

volunteer\_last\_name + ',' + volunteer\_first\_name AS VolunteerName

, volunteer\_phone\_number AS VolunteerNumber

, volunteer\_email\_address AS VolunteerEmail

FROM volunteer

GO

-- Examine Volunteer List.

SELECT \* FROM VolunteerList

DATA DICTIONARY

**Attribute** –Acharacteristic of an entity. For example, customer name, vendor address.

**Bridge Table** – A table that is placed between two tables that have a many to many relationship.

**Column** – A single type of data in a horizontal top to bottom form. For example, last name, first name.

**Conceptual Model** – The descriptive model of the database showing the relationships between the entities.

**Constraints** – Are used in SQL to tell the database what is the primary key and if it has unique or foreign keys.

**Create Table Statement** – This statement tells the database that it is creating a table which is followed by the table name, primary key, and then the rest of the columns.

**Drop statement** – Tells the database to delete the following information, which could be a table or a column

**Entity** – An object in a database that information is collected about. For example, customer or vendor.

**Foreign Key** – The column in a table that links it to another table. This column containing the foreign key in one table is the primary key in the other table.

**From Clause** – Is used to tell the database which tables the data is stored in and how those tables are joined.

**Insert (Into) Statement** – Is used to insert data (values) into the columns of the table

**Logical Model** – The model of the database that gives as much detail as possible.

**Microsoft Access** – A database management system, especially good for smaller data sets.

**Microsoft SQL Server** – A relational database management system that can support large amounts of data.

**Microsoft SQL Server Management Studio** – A software application that is used to interface with Microsoft SQL Server.

**Microsoft Power BI** – A software program that providers users with the ability to analyze data from different platforms, such as SQL Server, Access, and Excel

**Primary Key** – A unique attribute or column used to identify the entity or table. For example, student ID or parent ID

**Row** – All related data in a right to left horizontal form. For example, student ID, then student first name, then student last name, then student street address.

**Select Statement** – Used to select columns or select data from a database.

**Spreadsheet** – A sheet containing rows, columns, and cells that contain data. For example, Microsoft Excel is a spreadsheet program.

**Structured Query Language (SQL)** – A language used for creating tables, manipulating table structure, and extracting data for databases like SQL Server.

**Table** – A element of the database made up of columns and rows that stores data.

**Values Clause** – Tells SQL Server what will be entered into the column, for example judge\_first\_name is Sue. Sue is the value.

**Where Clause** – Is used to tell the database how to extract the data based on the criteria needed.