The Evolution of SQL Databases

Shaping the Relational Databases That We Use Today and What’s to Come in the Future

Sarah Grossheim  
 Computer Science  
University of North Alabama  
 Florence, Alabama United States  
 [sgrossheim@una.edu](mailto:email@email.com)

1 Introduction

In this paper, we will be observing the programming language SQL. We will particularly observe the evolution of SQL from the past to the future. This paper will be about the path SQL has taken from being just an idea to being brought to life to shape the relational databases we use today. We will also ponder upon what the future development of relational databases could look like. Artificial intelligence has become the focus of many fields as machine learning can accurately predict future trends, behaviors, and outcomes based on historical data. As artificial intelligence and machine learning continue to gain popularity, SQL could be further innovated to integrate advanced analytics and predictive capabilities into relational databases to help organizations make better business decisions.

**2 Background**

To create a database, one must gather, manage, and handle the data. There were three unique waves in the history of database technologies. The first wave had immense physical libraries full of data that were handled using comprehensive index systems. The second wave was in the form of punch card technology where data had to be processed mechanically by a tabulating machine. The third wave was when the electronic computer was created. People of this time had never seen such vast amounts of information before. These databases were stored on disk, and only the most refined programmers were able to retrieve the data from the disk. Although the creation of databases on computers was revolutionary during those times, the world needed an easier way to access this data without the need for overly complicated programming [1]. In the next section, the paper will outline how the computer scientists who created SQL did just that.

**2.1 Historical background**

In the early 1970s, the focus of research in computer science was to create a system that addressed persistent data. Persistent data was data that was to remain in a computer system until it was intentionally removed. E.F. (Ted) Codd was a computer scientist at IBM. He created the “Relational Data Model,” which was a new way to sort data into rows and columns. He also created two languages called Relational Algebra and Relational Calculus that were used to query these databases. Since Codd had an extensive background in mathematics, many operators in the languages were represented by complex mathematical symbols.

Codd’s model and languages were very influential to the creators of SQL. The men who created SQL are Don Chamberlin and Ray Boyce. Both men were captivated by how Codd’s languages could represent highly complex queries. However, both Chamberlin and Ray thought that there should be an easier way to represent these queries. They felt as though it should be possible to create a language that could be used by people who did not know much about mathematics and computer programming.

Chamberlin and Ray went to work on the System R project at the San Jose Laboratory in 1973. This is where they began their work on Sequel. Sequel allowed for queries to be displayed in a form that presented better readability, unlike previous languages. Since Ted Codd’s languages were written with mathematical notation, it was hard for an average person to read. Sequel aimed for users to be able to read the language as if it were just ordinary English. This is how Sequel got its name. Sequel stands for “A Structured English Query Language.” However, in 1977, Sequel was shortened to SQL because of trademark issues.

In 1974, Ray Boyce died suddenly at age 26 from a ruptured brain aneurysm. This happened about a month after presenting a paper about Sequel at a conference in Ann Arbor, Michigan. Ray and Chamberlin only got to work on two papers together. The first was called “Sequel: A Structured English Query Language.” This paper became widely known. The second was called “Using a Structured English Query Language as a Data Definition Facility,” but was never published outside of IBM. Chamberlin stated that he enjoyed working with Ray and that he thought they had a seamless partnership. He also said he thought that Ray would have been pleased to see the impact his ideas had on the world.

SQL went on to be used commercially in the late 1970s and early 1980s. It became a standard language definition called “Database Language SQL” in 1986. This standard was formally adopted by the American National Standards Institute, ANSI, and the International Organization for Standardization, ISO. These are two very widely known standard groups. SQL is still standard today.

As was stated earlier, the original plan for SQL was to be simple enough so it could be used by professionals whose area of expertise was not limited to just database management. However, this plan did not hold up as it is not as approachable to users with little to no training as the creators had originally hoped. SQL is now being used regularly by people who specialize in databases. These specialists repeatedly use SQL to process transactions such as bank deposits, credit card purchases, and online auctions [2].

**2.2 Language Overview**

SQL is unlike other programming languages. This language is not written with algorithms that perform operations sequentially. Instead, the user simply states the requirements of what they need, and the database management system will use its framework to determine the steps it needs to take to retrieve the user-requested information [3]. Thus, SQL is a declarative language.

*2.2.1 Relational Databases.* What is a relation? A relation is a two-dimensional table where each column has its own specifically defined name. No column name should be the same. The values in the column should also have the same kind of attributes. Also, each row of the column should be distinct. The order of the rows could be rearranged without having serious consequences because the order does not matter. A relational database is just a collection of these relations/tables [4, p. 7]. Figure 1 shows an example of what a relational database could look like. To manipulate these databases, we need a database management system. The most popular and standard approach for manipulating data in a relational database is SQL [5, p. 1]. SQL is the best language that can query, read, and update relational databases [3].

A screenshot of a computer

Description automatically generated

**Figure 1: A relational database that holds information about an employee [2].**

First, we should learn how a user can create a table in SQL. When the user names a column, the rules are: The name can be no longer than 18 characters. It must start with a letter. It can contain letters, numbers, and underscores, but it cannot contain spaces.

CREATE TABLE CUSTOMER

(CUSTNUMB DECIMAL(3),

CUSTNAME CHAR(15),

CUSTADDR CHAR(25),

TOTSALES DECIMAL(7,2))

The code above creates a table named customer with four columns that describe the customer’s number, the customer’s name, the customer's address, and the total number of sales. The customer’s number is a two-digit number, the customer's name is a fifteen-character long string, the customer’s address is a twenty-five-character long string, and the total sales is a seven-digit long number with two decimal places. This table will start completely blank until the user applies data manipulation commands and operations that are specified in the following section [4, p. 26].

*2.2.2 Data Manipulation.* The four basic data manipulation operations in SQL are INSERT, SELECT, UPDATE, and DELETE [5, p. 10].

The most basic SQL command, or SQL query, is SELECT. The SELECT operation has a form of “SELECT-FROM-WHERE” [5, p. 11]. It is primarily used to retrieve data from a database [4, p. 36].

A screenshot of a computer

Description automatically generated

Figure 2: Example of SQL SELECT query [4, p. 11].

The notation in Figure 2 shows that we first select a specific column that we want to print. Secondly, we specify what tables have these columns. Finally, we use a restriction to apply to the query. The example below shows that the user wants to print the customer’s name from the customer table when the customer’s number is equal to 178 [4, pp. 11-12].

SELECT CUSTNAME

FROM CUSTOMER

WHERE CUSTNUMB = 178

The INSERT query is a way to add rows to each table. It has a form of INSERT INTO followed by VALUES. After INSERT INTO, the user should specify which table they would like to add the data into. After VALUES, the user should input the values they want to add to the table. This is relatively simple, but the user must remember that character values should be enclosed by single quotation marks. The code below will add the data enclosed in parentheses to the SLSREP table.

INSERT INTO SLSREP

VALUES

(3, ‘Jones’, ‘Mary’, ‘123 Main, Grant, MI’, 2150.00, 0.05)

A screenshot of a computer

Description automatically generated

Figure 3: The sales rep table after the code above is executed [4, p. 29].

If the user needs to insert values into the table but one column has an unknown value, they should only insert the values that are known [4, p. 30]. The unknown values are what are called null values. Imagine a database has been created that is full of attributes about actors. Examples of null values can include the death date of an actor who has not died yet. It could also include a column of nominations or awards won for an actor who has not yet been nominated or awarded for their acting skills [4, p. 28]. When inserting values that are known, the user should do this by specifying which specific values they are entering into the columns. In the example below, the user is inserting data into the SLSRNUMB, SLRSNAME, TOTCOMM, and COMMRATE columns. The null value is the address, as it has not yet been recorded in the database [4, p. 30].

INSERT INTO SLSREP (SLSRNUMB, SLRSNAME, TOTCOMM, COMMRATE)

VALUES

(12, ‘Brown’, ‘Bob’, 21500.00, 0.03)

To change existing data, the user should use the UPDATE query. Suppose the user made a typo in a customer’s name, or a customer got married and needed to change their last name. The example below shows how the user can update existing values. Customer number 987 is getting their name changed from what it was previously using SET [4, p. 82].

UPDATE CUSTOMER

SET CUSTNAME = ‘Anderson, Allison’

WHERE CUSTNUMB = 987

Also, of course, there is always a need to delete data.

DELETE CUSTOMER

WHERE CUSTNAME = ‘Smith, John’

The user must be careful when deleting data. For example, if the user were to delete a customer named John Smith, all customers named John Smith would be deleted. The best practice is to use a condition that involves a primary key, so the user does not delete unnecessary data [4, p. 85].

*2.2.3 Data Types, Comparison Operators, and Compound Conditions*. The data types that are most encountered in SQL include DECIMAL, CHAR, INTEGER, SMALLINT, and DATE. DECIMAL(x, y) is a floating-point value where x is the length and y is the number of decimal points. CHAR(x) is a string value where x is the length of the string. INTEGER stores a number with no decimal point in the range of -2147483648 to 2147483647. SMALLINT stores an integer value from -32786 to 32767. If the user knows the value will not get much bigger or smaller than the range, it is better to use this as it does not take up as much storage. Finally, DATE stores a day of the year. If the user wanted to store the date, April 8, 2024, it could be stored either as 8-APR-24 or 4/8/2024 [4, pp. 27-28].

The comparison operators in SQL are similar to those found in other programming languages. Equal to is denoted by =. Less than is denoted by <, and less than or equal to is <=. Likewise, greater than is denoted by >, and greater than or equal to is >=. If a user wanted to prove that something is not equal, they would use <> or !=. However, most programmers that use SQL use the <> symbol of not equal rather than the != symbol [4, p. 37].

The compound conditions that are found in SQL are AND, OR, and NOT. When using AND, all conditions set must be true. When using OR, at least one of the conditions set must be true. NOT reverses the truth or falsity of the condition that was set. For example, if the condition was true, then the new condition would be false, and vice versa. Figure 4 demonstrates a query that will print the description of parts that are from warehouse 3 and have more than 100 units on hand [4, pp. 38-39].

A screenshot of a computer

Description automatically generated

Figure 4: Demonstration of the AND compound condition and the equal to and greater than comparison operators in SQL [4, p. 39].

Since there are no special formatting rules in SQL, the code in Figure 4 could also be written as follows:

SELECT PARTDESC

FROM PART

WHERE (WRHSNUMB = 3) AND (UNONHAND > 100)

Notice that the WHERE clause is not on a separate line and is surrounded by parentheses. The code will still do the same thing; however, it was separated in Figure 3 to promote better readability to the user [4, p. 36].

*2.2.4 Built-In Functions*. SQL has several built-in functions that aid the user in helping calculate specific data from databases. These functions include COUNT, SUM, AVG, MAX, and MIN.

The COUNT function calculates how many rows satisfy a specific WHERE clause. WHERE is simply how a user would apply a restriction to the query. The SUM function calculates the sum of a column for all rows that have numerical values. Likewise, the AVG function calculates the average of a column for all rows that have a numerical value. The MAX function returns the largest value in a column. If the columns contain numerical values, it will return the largest number, however, if the column contains a set of names, then the function will return the last name in alphabetical order. In the same manner, the MIN function returns the smallest value in a column. If the column contains numerical values, it returns the smallest number. If the column contains names, it will return the first name in alphabetical order. When one uses the SUM, AVG, MAX, or MIN functions, if there are any null values in the column, those will be disregarded. Figure 5 shows the syntax of the AVG, MAX, and MIN functions. The syntax of each function is written the same. However, the only difference is that each function calculates a different statistic. The output is shown after each query [4, pp. 46-48].

A screenshot of a computer

Description automatically generated

Figure 5: Illustration of the COUNT, AVG, MAX, and MIN functions [4, p. 47].

*2.2.5 Conclusion*. The basic idea of SQL is to make a database useful. To do this, one must be able to add data to it. If needed, the user should be able to change the data in it. It should also be possible for the user to extract and delete data from it. The user also needs to be able to query the database, which is asking questions about the database. This is very beneficial for a company or business that has a large database system that needs to be managed. For example, let’s say a company could be tracking down the customers who have outstanding debt. The user might want to search through the database to find a customer whose balance exceeds the credit limit. A query can find those customers and produce a listing of all those customers’ names and information [4, p. 11]. An SQL query is meant to be entered and then executed to produce results. After this, the process will terminate. However, this is only in stand-alone mode. There is another type of SQL that allows the user to embed SQL commands into another language. It is called embedded SQL. This lets other languages enjoy the many benefits of SQL. One of these languages is called COBOL [4, p. 121]. In the following sections, we will be talking about another programming language that can import SQL databases to be used for Artificial Intelligence.

**2.3 Literature Review**

*“*The Lessons of Database History*”* is an entry from a trade journal written in 2015 by Guy Harrison who is an executive director of Research and Development at Dell. This journal talks about Doug Turnbull’s speech from a conference in San Jose about the last thirty to forty years of database history. Indexing systems and tabulating machines were the main methods used before computer databases. These methods were time-consuming and inefficient. The world needed a better way to gather, manage, and handle large amounts of data. One must know this history of databases to understand why computer databases were a necessity to be created [1].

“Early History of SQL” is an entry from a scholarly journal published by IEEE in 2012. This source was written directly by one of the creators of SQL, Donald Chamberlin. He writes about the journey that he and his partner, Ray Boyce, embarked upon to create SQL. He talks about their trial and errors, personal influences for creating the language, and reflects upon the little time he got to spend with his partner before he, unfortunately, passed away at the young age of 26. What better way to explore the history of SQL than through the eyes of one of the creators himself [2]?

“Structured query language” is an entry from a trade journal written by Christine McGeever and published in the year 2000. McGeever gives an insightful definition of SQL and contrasts how it is not like other imperative languages. From this perspective, one was able to conclude that SQL’s paradigm is declarative, as it specifically describes what needs to be done rather than having to write a procedure on how to do it. She also talks about relational databases and the story of SQL. This source gives a great overview of the language to those who are unfamiliar with SQL [3].

“A Guide to SQL” is a book written by Philip J. Pratt and published in 1991. The author is a computer science professor at Grand Valley State University. This is the second edition of this book, which is a textbook aimed at helping students learn how to use SQL. This textbook gives the reader great examples of code and thoroughly explains what it does. It does an outstanding job of explaining the language in simple terms. The textbook could teach the most novice-level programmer how to program in SQL. It lives up to the title of the book, which is “A Guide to SQL,” as it guided in writing the bulk of the language overview section of this paper [4]. Another book called “A Guide to The SQL Standard” was written by C.J. Date and aided in backing up the information provided by the textbook written by Pratt [5].

Mark Terwilliger is a professor at the University of North Alabama. He has taught many students in various classes. One course specifically is CS470, which is about Artificial Intelligence. Many notes from his lectures have been used throughout the methodology section of this paper. These topics include machine learning, supervised learning, Python modules [6], and logistic regression [7].

“Harnessing the Power of SQL and AI for Predictive Analytics and Forecasting with Python” is an article written by Victor Magallanes from IT Solutions Network and published on medium.com. The company he works at is a local company from Texas that provides technical support to individuals and other small businesses. While Magallanes is not very well-known in the tech world, his ideas about combining SQL and AI in Python play a huge part in this research paper. This source may not be the most reputable, however, it was written in February of this year, which shows that SQL could have a bright and upcoming future in machine learning and AI [8].

“How to Perform Machine Learning Tasks with Python and SQL” is a source that came from a website called freecodecamp.org and was written by Oluseye Jeremiah. FreeCodeCamp is a website used by people all over the world for educational content in the realm of coding and computer science. As many do not know how to incorporate a MySQL database into Python code, this was a valuable source for this paper as it teaches one to do so. To be able to prove the hypothesis that SQL could be further updated to incorporate machine learning predictive analytics, it needed to be shown that it is possible to import SQL to Python to create AI code, specifically logistic regression. It may not have come from the most credible source, but it was a helpful aid in understanding how to incorporate SQL in Python [9].

**3 Methodology**

Python is a prevalent programming language used for various tasks. These tasks can include being able to create artificial intelligence models. This is because there are several different modules and libraries that one can import into Python. These modules and libraries contain prebuilt code that provides frameworks and tools that can assist a computer programmer in creating artificial intelligence models. Without these modules, a programmer would have to write code from scratch [6].

A popular sub-discipline of artificial intelligence is called machine learning. Machine learning involves training computers to perform tasks without being explicitly programmed to do so. To do this, the computer learns from labeled training data to eventually make decisions or predictions that are not based on the labeled training data. The two different kinds of machine learning are unsupervised learning and supervised learning. Unsupervised learning is when the user trains the model based on labeled training data. Supervised learning is when the model learns but does not have to rely on the labeled training data [6]. Figure 6 shows an illustration of the machine learning pipeline.

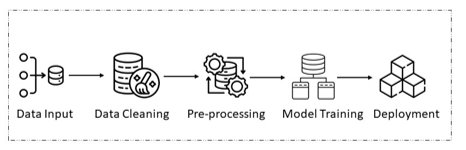


Figure 6: The steps of the machine learning pipeline.

As shown in Figure 6, the first step is data input. SQL is known for its ability to extract data from databases. This makes it a perfect environment for finding the input data. The second step is data cleaning. SQL can clean data by eliminating duplicates, managing missing values, and converting the data into a format that can be used for analysis. Python also has ways to clean data by using Pandas or NumPy by encoding categories to numbers or splitting the data into testing and training sets [8]. The third step is pre-processing. Pre-processing is the act of using feature engineering to create new features to enhance the predictive ability of the machine learning model [6]. SQL can be used to add new features by aggregating, summarizing, or joining data from several tables [8]. The fourth step is model training. SQL does not yet have any way to train data, so this is where Python comes into play.

Python’s library sci-kit-learn provides users with several different machine learning capabilities such as regression, classification, and clustering. Regression and classification are both used in unsupervised learning, while clustering is used in supervised learning [6]. With sci-kit-learn, users can create regression models among various other models that are based on training data to make accurate predictions. Regression is a statistical method used in several disciplines including finance and investing that can help determine the strength and character of the relationship between one dependent variable and a series of other variables [7]. The fifth step is deploying the model. This is when the user selects the model they want to deploy into production [6].

Python can import SQL databases to perform machine learning using the Python libraries called SQLAlchemy or Pandasql [8]. However, the code in Figure 7 uses pymysql, which is a database API. This library helps the user connect to a MySQL database, which is a relational database management system that employs SQL.

A screenshot of a computer

Description automatically generated

Figure 7: Python code that creates a connection to a MySQL database [9].

As shown above in Figure 7, this code also imports the pandas and sklearn (sci-kit learn) modules. Pandas is a way to read input data to be stored in a data frame that has rows and columns [6]. Then the code creates a connection to the SQL database. In Figure 8, Pandas is used to take the information from the database and store it into a pre-built data frame. Using the data frame, the code then prepares the data by taking two features to use for X and the target variable for Y. After the data is prepared, the code creates a logistic regression model using sklearn and trains the model using the X and Y variables in the fit() function. Finally, the code closes the connection as it is not needed anymore [9].

A screenshot of a computer

Description automatically generated

Figure 8: Python code that creates a logistic regression model from a MySQL database [9].

Figure 8 shows code that is a skeleton for importing a MySQL database to be used for Python machine learning. Since it is just a skeleton, it does not provide an actual logistic regression model. An actual logistic regression model could look like the picture shown below in Figure 9.

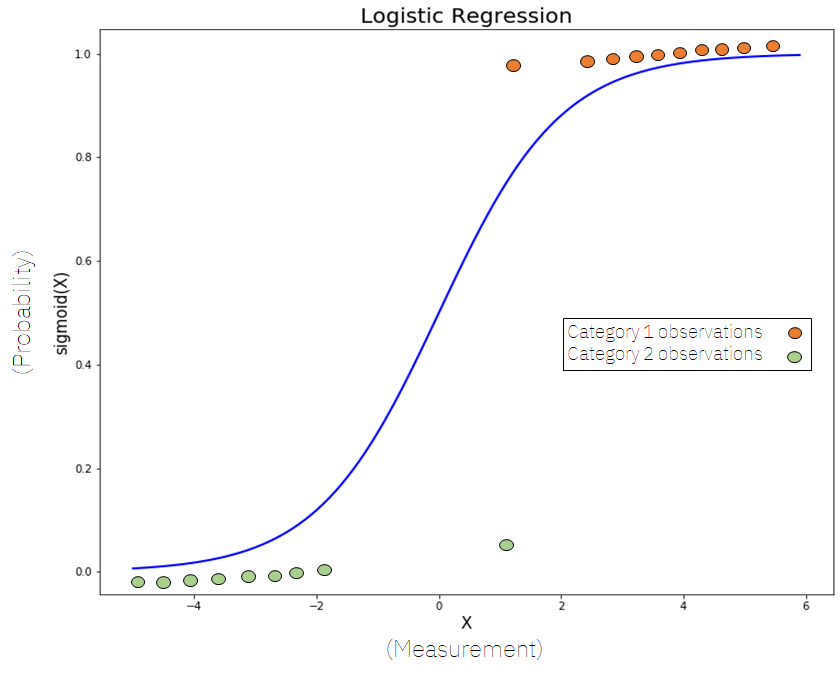


Figure 9: A logistic regression model that measures probability based on the measurement of two observations.

Machine learning is not just limited to logistic regression models, other models can be used such as classification and clustering [6]. Logistic regression, however, is a model that many data analysts use to make predictions. Also, it is best practice to evaluate the model to determine the accuracy of the model’s performance [6].

**4 Results/Conclusions**

As of today, the world we live in is data driven. Most businesses and organizations today rely on analytics. Data analysts are critical assets to any company. Their job is to use historical data to predict future trends and patterns to make informed decisions about their business or organization. Artificial intelligence and machine learning are continuing to expand, so this technology can further complement and contribute to the work of data scientists to help make better business decisions. As was stated in the previous sections, SQL databases can be used in other languages such as COBOL and Python, so why couldn’t other languages be used in SQL?

In the future, SQL should be expanded to incorporate machine learning code, as it would help assist data analysts in companies that frequently use SQL databases. Although the original creators of SQL designed the language to be used as a stand-alone language for queries and for processing online transactions [2], that was well over fifty years ago. If technology is growing at a significant rate, so should SQL.

To import a SQL database to Python, the user first must make a connection to the database. If SQL had machine learning capabilities, the need to make a connection to the database would be eliminated because the databases would already be in the current SQL programming environment. This would make it easier for the people who use SQL as they are already able to insert, select, and delete data as needed. They can use SQL’s functionality to search through their databases to find the exact information needed to use in the machine learning models to help make predictions and decisions for the business.

Incorporating artificial intelligence into SQL does not have to be limited to just machine learning. Several other languages currently use chatbot artificial intelligence in-house to help the programmer debug errors. This helps speed up production and could also help provide a seamless transition to data scientists who are not familiar with other languages that could be further implemented into SQL. Overall, if SQL incorporated other languages that support being able to create artificial intelligence code into the language, it could be revolutionary for the language if done correctly and accurately.

**5 Future Work**

As this paper ends, future research related to the topic and hypothesis could include researching how to incorporate SQL databases with artificial intelligence. Someone could implement an AI model into their company’s SQL database management system. Then, the researcher could use that in their company on a day-to-day basis along with human data scientists to write a paper about whether the AI proved to produce better predictive results than humans do. However, this could potentially pose a threat to several data analysts’ jobs if the AI’s data was significantly better than the human’s. From this research, it could be further elaborated on whether artificial intelligence should be slowed down to prevent the job losses of humans. Humans must have jobs to make money to be able to survive, but artificial intelligence only survives on computers powered by electrical power. Is it ethical to keep creating technology that surpasses a human’s ability if it leads to taking away the jobs of several trained specialists who have worked this career their whole lives?

REFERENCES

|  |
| --- |
| [1] G. Harrison, "The Lessons of Database History," *Database Trends and Applications,* vol. 29, no. 3, p. 54, 2015. |
| [2] D. D. Chamberlin, "Early History of SQL," *IEEE Annals of the History of Computing,* vol. 34, no. 4, pp. 78-82, 21 November 2012. |
| [3] C. McGeever, "Structured query language," *Computerworld,* vol. 34, no. 20, p. 70, 15 May 2000. |
| [4] P. J. Pratt, A Guide to SQL Second Edition, Boston: boyd & fraser publishing company, 1991. |
| [5] C. Date, A Guide to The SQL Standard, Reading, Massachusetts: Addison-Wesley Publishing Company, 1987.  [6] M. Terwilliger, Class Lecture, Topic: “Jupyter, Python Modules, the ML Pipeline, Supervised Learning”, CS470, College of Business and Technology, University of North Alabama, Florence, Feb, 5, 2024.  [7] M. Terwilliger, Class Lecture, Topic: “Regression (single and multiple variables)”, CS470, College of Business and Technology, University of North Alabama, Florence, Feb, 14, 2024.  [8] O. Jeremiah, “How to Perform Machine Learning Tasks with Python and SQL,” FreeCodeCamp, 9 March 2023. [Online]. Available: https://www.freecodecamp.org/news/machine-learning-with-python-and-sql/. [Accessed 6 April 2024].  [9] V. Magallanes, “Harnessing the Power of SQL and AI for Predictive Analytics and Forecasting with Python,” Medium, 6 Feb 2024. [Online]. Available: https://medium.com/@amb39305/harnessing-the-power-of-sql-and-ai-for-predictive-analytics-and-forecasting-with-python-147bd59759df. [Accessed 6 April 2024]. |

|  |
| --- |
|  |
|  |