

AskSQL

Transforma Linguagem Natural em SQL, Relatórios e Gráficos

Cristiano Antunes Nicolau

108536

Universidade de Aveiro

Departamento de Electrónica, Telecomunicações e Informática

Mestrado em Ciencia de Dados

Sistemas de Base Dados

<https://github.com/cristiano-nicolau/AskSQL>

Contents

1	Introdução	3
2	State of the Art	3
2.1	Graph Maker [1]	3
2.2	NL4DV [2]	3
2.3	Text2SQL.AI [3]	3
2.4	Spider [4]	4
2.5	Modelos Text-to-SQL do Hugging Face [5]	5
2.6	ChatGPT [6]	5
3	Metodologia	5
3.1	Parte 1: Experimentar modelos NLP para quando recebo datasets de uma tabela	5
3.2	Parte 2: Trabalhar com modelos Text-to-SQL para conexões com base de dados de múltiplas tabelas	6
3.3	Parte 3: Visualização dos dados em uma interface interativa	6
4	Modelo Text to SQL	6
4.1	T5-Base-Text-to-SQL	7
4.2	t5-small-awesome-text-to-sql	7
4.3	text-to-sql-with-table-schema	7
4.4	flan-t5-text2sql-with-schema-v2	7
4.5	Groq - Llama-3.3-70b-versatile	8
5	Implementação	9
5.1	Interface	9
5.2	Funcionalidades da Aplicação	9
5.3	Estrutura do Chatbot	9
5.3.1	Geração de Relatórios	9
5.3.2	Geração de Gráficos	10
5.3.3	Construção de Consultas SQL	10
5.4	Comandos Adicionais	11
6	Testes	12
6.1	Teste 1: Consulta sobre Alunos e Cursos	12
6.1.1	Consulta: Listar todos os alunos matriculados em Sistemas de Base de Dados e nascidos em 2003	12
6.1.2	Consulta: Lista a unidade curricular com maior numero de inscritos, os seus nomes e datas de nascimento.	13
6.1.3	Testes ao SQL Generator	14
6.2	Teste 2: AdventureWorks2012 Database	14
6.2.1	Questões Utilizadas	14
6.2.2	Resultados Obtidos	16
6.3	Teste 3: TPC-H Database	16
6.3.1	Questões Utilizadas	17
6.3.2	Resultados Obtidos	18
7	Conclusões	18

1 Introdução

Com o avanço das tecnologias e da inteligência artificial (IA), cada vez mais pessoas procuram realizar tarefas complexas sem a necessidade de conhecimento especializado, recorrendo a aplicações inteligentes que simplificam processos.

No contexto da análise de dados, essa tendência tem se refletido na procura por soluções que permitam a qualquer utilizador extrair informações valiosas sem precisar dominar linguagens de programação e/ou ferramentas de BI. Diante deste cenário, este projeto visa criar uma aplicação que permita que qualquer pessoa, mesmo sem conhecimentos técnicos, consiga criar relatórios e gráficos a partir de consultas em linguagem natural.

A ideia principal é proporcionar uma experiência intuitiva ao utilizador, eliminando a necessidade de conhecimento avançado em base de dados ou ferramentas de BI. O utilizador escreve algo como:

"Quero um relatório das vendas por categoria e país em 2023."

A aplicação interpreta o pedido utilizando processamento de linguagem natural (PLN) e, através de um modelo de linguagem de grande escala (LLM), gera automaticamente a query SQL correspondente. Em seguida, a consulta é executada no banco de dados e os dados extraídos são processados e apresentados em relatórios interativos e visualizações dinâmicas, como gráficos de barras, pizza e linhas.

2 State of the Art

Nesta secção, vamos abordar alguns trabalhos semelhantes já existentes e explorar as soluções propostas por eles. O objetivo é perceber as abordagens utilizadas, identificar seus pontos fortes e limitações e, com isso, fundamentar as escolhas para o desenvolvimento da nossa solução.

Serão analisadas ferramentas que utilizam processamento de linguagem natural para a criação de consultas SQL, bem como sistemas que apresentam relatórios dinâmicos com base em consultas automáticas. Além disso, serão exploradas pesquisas acadêmicas e soluções comerciais que procuram facilitar a interação entre utilizadores não técnicos e base de dados.

2.1 Graph Maker [1]

Este site permite aos utilizadores carregar conjuntos de dados em formatos como CSV e Excel. Com base na query fornecida pelo utilizador, o sistema interpreta a solicitação, agrupa os dados e os organiza conforme as colunas do dataset. A ferramenta visa facilitar a visualização dos dados sem exigir conhecimento técnico avançado, proporcionando gráficos automáticos e personalizáveis conforme as necessidades do usuário.

No entanto, sua funcionalidade pode ser limitada quando comparada a soluções que utilizam modelos de linguagem natural para a interpretação de consultas mais complexas. Além disso, o Graph Maker aceita apenas conjuntos de dados provenientes de uma única tabela, não oferecendo suporte a bases de dados relacionais com múltiplas tabelas, o que pode restringir sua aplicabilidade.

2.2 NL4DV [2]

A ferramenta NL4DV (Natural Language for Data Visualization) é uma interface de linguagem natural voltada para a visualização de dados, projetada para permitir que os utilizadores interajam com dados e façam perguntas em linguagem natural, sem a necessidade de conhecimento técnico aprofundado em ferramentas tradicionais de visualização.

Os autores do artigo desenvolveram uma biblioteca em Python que recebe uma consulta (query) em linguagem natural junto com o conjunto de dados e gera visualizações apropriadas como resposta. A biblioteca utiliza técnicas de processamento de linguagem natural (NLP) para interpretar a query do utilizador, identifica os tipos de dados e o contexto da solicitação, e em seguida, determinar qual visualização seria a mais adequada.

Uma das maiores limitações é que a biblioteca só aceita conjuntos de dados estruturados em formato de tabela, ou seja, como DataFrames em bibliotecas como o pandas, e não permite a integração direta com base de dados relacionais ou outras fontes de dados mais complexas, como bases de dados NoSQL ou dados em tempo real.

2.3 Text2SQL.AI [3]

A ferramenta Text2SQL.AI utiliza Processamento de Linguagem Natural (PLN) para converter perguntas feitas em linguagem natural em consultas SQL automaticamente. Isso facilita a análise de dados, permitindo que os utilizadores sem conhecimento avançado em SQL interajam de forma intuitiva com base de dados, sem a necessidade

de compreender a sintaxe complexa de SQL.

Para avaliar a eficácia da ferramenta, realizei dois testes com diferentes níveis de complexidade de consultas, sem fornecer explicitamente a estrutura da base de dados. A seguir, estão os detalhes dos testes:

1. Teste de consulta simples:

Consulta: *'List of all students enrolled in the Database Systems course born in 2003.'*

```
1 SELECT s.name FROM students s
2 JOIN enrollments e ON s.student_id = e.student_id
3 JOIN courses c ON e.course_id = c.course_id
4 WHERE c.course_name = 'Database Systems'
5 AND YEAR(s.birth_date) = 2003;
```

Neste primeiro caso, a ferramenta conseguiu gerar a consulta SQL corretamente, realizando os joins necessários entre as tabelas `students`, `enrollments` e `courses`, e aplicando o filtro adequado para selecionar os estudantes nascidos em 2003. A consulta gerada foi precisa e atendeu à expectativa.

2. Teste de consulta mais complexa:

Consulta: *'List the functionaries that have a salary bigger than 1300\$ and work in OPORTO city, and have more than 57 years.'*

```
1 SELECT name
2 FROM employees
3 WHERE salary > 1300
4 AND city = 'Oporto'
5 AND age > 57;
```

Embora a consulta SQL gerada estivesse sintaticamente correta, ela não retornou o resultado esperado, pois faltava considerar as tabelas adicionais que associam os funcionários às suas respectivas cidades e outras condições relevantes. Isso ocorre porque a consulta não envolveu os joins necessários entre as tabelas, o que resultou em uma resposta incompleta.

Após explicar ao chatbot que era necessário realizar joins com outras tabelas para obter as informações corretas, a ferramenta gerou a consulta esperada:

```
1 SELECT e.name
2 FROM employees e
3 JOIN work w ON e.employee_id = w.employee_id
4 JOIN company c ON w.company_id = c.company_id
5 WHERE e.salary > 1300
6 AND c.city = 'Oporto'
7 AND e.age > 57;
```

Desta forma, a ferramenta foi capaz de entender a necessidade de juntar informações das tabelas `employees`, `work` e `company`, gerando a consulta correta que atendeu aos requisitos do enunciado. Isso destaca que, embora o Text2SQL.AI funcione bem em casos simples, ele pode enfrentar desafios em consultas mais complexas, especialmente quando é necessário compreender relações entre múltiplas tabelas.

Estes testes demonstram que, para consultas mais complexas, pode ser necessário fornecer um pouco mais de contexto à ferramenta, para que ela possa interpretar corretamente a estrutura do banco de dados e realizar os joins necessários.

2.4 Spider [4]

O Spider é um benchmark muito utilizado para a avaliação de modelos Text-to-SQL. Criado pelo grupo YALE LILY, contém um extenso e diversificado conjunto de dados, projetado para avaliar a capacidade de modelos de IA em converter perguntas feitas em linguagem natural para consultas SQL. O Spider abrange uma grande variedade de tópicos, de consultas simples a consultas mais complexas, e inclui uma diversidade de base de dados e esquemas para testar a generalização dos modelos em diferentes contextos de dados.

A complexidade do Spider não se limita apenas à variedade das perguntas, mas também à necessidade de o modelo realizar múltiplos joins, subconsultas e manipulações complexas de dados para gerar SQL válido. Essa diversidade torna o Spider um benchmark essencial para o desenvolvimento e avaliação de modelos Text-to-SQL, e é muito utilizado para treinar e testar modelos de machine learning que procuram converter linguagem natural em SQL.

Inicialmente, pensei em treinar um modelo de Text-to-SQL a partir do Spider, devido à riqueza do conjunto de dados. No entanto, durante a pesquisa, encontrei alguns modelos já treinados e otimizado utilizando o Spider, falados na subsecção seguinte

2.5 Modelos Text-to-SQL do Hugging Face [5]

O Hugging Face oferece uma variedade de modelos Text-to-SQL projetados para converter perguntas feitas em linguagem natural em consultas SQL. Esses modelos, que utilizam arquiteturas avançadas como o T5 (Text-To-Text Transfer Transformer) e outros transformers, são treinados em grandes benchmarks, como o Spider, para fornecer soluções eficazes em conversão de texto para SQL.

Estes modelos aproveitam o poder dos transformers para lidar com tarefas de processamento de linguagem natural, como tradução e resposta a perguntas, adaptando essas capacidades para gerar consultas SQL. O T5, por exemplo, é um modelo pré-treinado que pode ser afinado para uma variedade de tarefas de NLP (Natural Language Process), incluindo a conversão de texto para SQL.

Estes modelos Text-to-SQL do Hugging Face mostram-se eficazes em tarefas que exigem a criação de consultas SQL de forma automática. Eles são particularmente úteis em cenários onde os utilizadores não têm conhecimento em SQL, mas precisam realizar consultas em base de dados usando apenas linguagem natural.

2.6 ChatGPT [6]

Como parte da pesquisa, também realizei os mesmos testes de consultas SQL utilizando o ChatGPT. Para isso, forneci ao modelo a descrição das tabelas da base de dados e as questões a serem resolvidas. O objetivo era testar a capacidade do modelo em gerar consultas SQL corretamente a partir de linguagem natural, considerando a estrutura do banco de dados fornecida.

```
1 Transforma a seguinte frase numa querie SQL:
2 "Quero todos os alunos nascidos em 2003 inscritos na uc Sistemas de Base Dados"
3
4 Tenho o seguinte schema:
5 Alunos - nmec, nome, ano_nascimento
6 inscricoes - nmec, id_disciplina
7 uc - id, nome, nr_maximo_inscritos
```

O ChatGPT gerou a seguinte consulta SQL, que foi capaz de fornecer o resultado esperado:

```
1 SELECT a.nmec, a.nome, a.ano_nascimento
2 FROM Alunos a
3 JOIN inscricoes i ON a.nmec = i.nmec
4 JOIN uc u ON i.id_disciplina = u.id
5 WHERE a.ano_nascimento = 2003
6 AND u.nome = 'Sistemas de Base de Dados';
```

A consulta gerada foi precisa e atendeu às expectativas. O modelo foi capaz de realizar corretamente os joins entre as tabelas alunos, inscrições e uc, aplicando os filtros necessários para selecionar os alunos nascidos em 2003 e inscritos na unidade curricular "Sistemas de Base de Dados". Após obter esses resultados com o ChatGPT, comecei o desenvolvimento da aplicação.

3 Metodologia

Nesta secção retrato a metodologia utilizada ao longo do projeto. A metodologia foi dividida em três partes principais, com o objetivo de explorar diferentes abordagens para lidar com consultas SQL a partir de linguagem natural: a primeira foca em trabalhar com datasets simples (compostos por uma única tabela), a segunda aborda bases de dados mais complexas com múltiplas tabelas, e a terceira trata da visualização dos dados em uma interface interativa. Essa divisão permite testar diferentes estratégias e avaliar as limitações dos modelos em cenários distintos.

3.1 Parte 1: Experimentar modelos NLP para quando recebo datasets de uma tabela

O primeiro desafio consiste em trabalhar com datasets compostos por uma única tabela. Com este cenário, o meu objetivo é através de modelos de Processamento de Linguagem Natural (NLP) ser capaz de compreender consultas feitas em linguagem natural e traduzi-las em consultas SQL válidas.

Como o modelo lida apenas com uma tabela, a complexidade é menor, já que não será necessário realizar joins com outras tabelas, mas sim focar em uma tradução direta da linguagem natural para a sintaxe SQL, considerando as colunas dessa única tabela.

3.2 Parte 2: Trabalhar com modelos Text-to-SQL para conexões com base de dados de múltiplas tabelas

A segunda parte da metodologia envolve trabalhar com base de dados que possuem múltiplas tabelas, o que traz desafios adicionais, como a necessidade de realizar joins, aplicar múltiplas condições WHERE, e lidar com relações complexas entre as tabelas.

Para esta parte, pretendo explorar modelos Text-to-SQL já existentes, com o objetivo de gerar consultas SQL a partir de linguagem natural, considerando a estrutura e os relacionamentos entre as diversas tabelas da base de dados. Aqui, a abordagem será testar diferentes modelos para entender como eles lidam com esses casos mais complexos e onde as suas limitações podem surgir.

Esta divisão da metodologia procurar explorar as vantagens e desafios de cada tipo de dado (dataset simples vs. base de dados complexa) e permitirá uma análise detalhada das soluções que podem ser aplicadas para cada cenário.

3.3 Parte 3: Visualização dos dados em uma interface interativa

Após a conversão das consultas para SQL e a extração dos dados, a terceira parte da metodologia concentra-se na visualização dos resultados em uma interface web interativa.

A interface deve ser intuitiva e eficiente, permitindo que os utilizadores insiram consultas em linguagem natural e visualizem os resultados de forma clara. Além disso, para facilitar a análise dos dados, a aplicação deve incluir funcionalidades como:

1. **Geração de gráficos interativos** para representar visualmente as informações extraídas da base de dados.
2. **Criação de relatórios** estruturados para que os utilizadores possam obter insights detalhados a partir das consultas realizadas.

Com esta abordagem, procuramos uma solução completa que, através da geração de queries SQL, torne a análise de dados mais acessível e eficiente, com uma melhor experiência de utilizador, principalmente para pessoas sem conhecimentos técnicos.

4 Modelo Text to SQL

Nesta seção, explico o caminho seguido no desenvolvimento do meu projeto. A implementação foi dividida em duas partes distintas, com base no tipo de dados com os quais os modelos seriam testados:

- **Parte 1:** Criação de um algoritmo NLP próprio ou a utilização de modelos existentes para lidar com datasets contendo apenas uma única tabela. O objetivo aqui foi desenvolver um modelo que fosse capaz de gerar consultas SQL a partir de uma linguagem natural simples, sem a necessidade de realizar joins ou interagir com múltiplas tabelas.
- **Parte 2:** Uso de modelos Text-to-SQL para transformar consultas em linguagem natural em SQL, focando em bancos de dados com múltiplas tabelas. O propósito nesta segunda parte foi avaliar como diferentes modelos lidariam com consultas SQL mais complexas, que exigem a combinação de múltiplas tabelas e a aplicação de condições diversas.

A ideia é analisar como cada abordagem e modelo se comporta, desde consultas SQL simples até aquelas que exijam maior complexidade, como múltiplos joins e filtros avançados.

A ideia inicial seria dividir em duas partes, mas optei por não dividir esta secção em duas partes, uma vez que ao fazer a investigação para a segunda parte, para as tabelas mais complexas, realizei testes mais simples com casos de uma tabela só.

Na Parte 2 do projeto, o foco foi lidar com bases de dados que envolvem múltiplas tabelas, onde as consultas SQL se tornam mais complexas, exigindo a utilização de joins e múltiplas condições. Para isso, testei diversos modelos Text-to-SQL com o objetivo de avaliar como cada um lida com essas consultas mais complexas.

4.1 T5-Base-Text-to-SQL

Primeiramente, comecei por utilizar o modelo T5-Base-Text-to-SQL do Hugging Face, criado por suriya7 [7], um modelo amplamente utilizado na conversão de consultas em linguagem natural para SQL. No entanto, durante os testes, não consegui obter resultados. O modelo não foi capaz de gerar as consultas SQL para os cenários apresentados, sugerindo limitações no contexto de consultas mais complexas envolvendo múltiplas tabelas.

4.2 t5-small-awesome-text-to-sql

Na sequência, testei o modelo cssupport/t5-small-awesome-text-to-sql [8]. Este modelo teve um melhor desempenho, sendo eficaz na criação de tabelas e na inserção de dados. No entanto, ao tentar gerar consultas SQL do tipo SELECT, o modelo não foi capaz de o fazer, visto que foi criado só e apenas para a criação e inserção de dados SQL.

4.3 text-to-sql-with-table-schema

Em seguida, explorei o modelo juierrr/text-to-sql-with-table-schema [9]. Este modelo mostrou um bom desempenho, funcionando bem para gerar consultas SQL a partir de um único conjunto de dados com várias colunas. No entanto, quando as consultas se tornaram mais complexas, com múltiplos joins ou condições WHERE, o modelo não foi capaz de gerar as consultas corretas, devido ao modelo não conseguir lidar com consultas com várias tabelas.

4.4 flan-t5-text2sql-with-schema-v2

Considerando as limitações encontradas, decidi testar a versão atualizada do modelo do mesmo autor: juierrr/flan-t5-text2sql-with-schema-v2 [10]. Esta versão foi mais eficaz ao lidar com consultas SQL mais simples, mas ainda assim apresentou falhas em consultas mais complexas que exigiam múltiplos joins e condições avançadas WHERE. Embora tivesse melhor desempenho do que as versões anteriores, em três testes mais complicados, o modelo falhou em gerar o SQL correto.

Teste 1:

```
1  Consulta: "List all students who are subscribed in the Database Systems and are born in 2003."
2
3  Estrutura das tabelas:
4  {
5      'Students': ['student_id', 'name', 'birth_year'],
6      'Courses': ['course_id', 'name', 'max_enrollments'],
7      'Enrollments': ['student_id', 'course_id']
8  }
9
10 Consulta SQL gerada:
11 SELECT T1.name
12 FROM students AS T1
13 JOIN enrollments AS T2 ON T1.student_id = T2.student_id
14 WHERE T2.course_id = 'Database Systems'
15 AND T1.birth_year = 2003
```

O código SQL gerado apresentou um erro importante no filtro do nome do curso. A consulta tenta comparar o course_id com o nome do curso diretamente, mas o course_id é um identificador numérico enquanto o nome do curso ("Database Systems") está presente na tabela Courses, e não em Enrollments.

Teste 2:

```
1  Consulta: "List names of employees who have a salary greater than $1300, work in the city of Oporto and are younger than 30 years."
2
3  Estrutura das tabelas:
4  {
5      'employees': ['employee_id', 'name', 'salary', 'age'],
6      'company': ['company_id', 'name', 'city'],
```

```

7      'work': ['employee_id', 'company_id']
8  }
9
10 Consulta SQL gerada:
11 SELECT t1.name
12 FROM employees AS t1
13 JOIN work AS t2 ON t1.employee_id = t2.employee_id
14 JOIN company AS t3 ON t2.company_id = t3.company_id
15 WHERE t3.city = 'Oporto'
16 AND t1.age < 30

```

A consulta SQL gerada está quase correta, mas faltou a condição para filtrar os funcionários com salário superior a \$1300. Este erro mostrou mais uma vez que este modelo tem algumas falhas, impossibilitando o uso do mesmo na minha aplicação.

4.5 Groq - Llama-3.3-70b-versatile

Dado que os modelos mencionados apresentaram limitações nas consultas mais complexas, decidi explorar uma abordagem mais avançada. Pesquisando por modelos LLMs, optei por usar o modelo Llama-3.3-70b-versatile da Groq [11].

A Groq é uma plataforma que oferece aceleração de hardware e software para execução de modelos de inteligência artificial, permitindo um processamento mais rápido e eficiente. O modelo Llama-3.3-70b-versatile é um modelo de linguagem natural desenvolvido pela Meta, com 70 bilhões de parâmetros, o que lhe confere uma boa capacidade de lidar com uma variedade de tarefas, incluindo a conversão de linguagem natural em SQL.

Ao fornecer o schema da base de dados, consegui facilmente obter os resultados esperados, mesmo para consultas mais complexas que exigiam múltiplos joins e condições avançadas WHERE. Esta abordagem superou as limitações dos modelos anteriores, garantindo maior eficiência na extração de dados.

Teste Realizado:

- **Query 1:** 'List all students who are subscribed in the Database Systems and are born in 2003.'
- **Query 2:** 'List names of employees who have a salary greater than \$1300, work in the city of Oporto and are younger than 30 years.'

Estrutura das tabelas:

```

1  {
2      'teste1.Students': ['student_id', 'name', 'birth_year'],
3      'teste1.Courses': ['course_id', 'name', 'max_enrollments'],
4      'teste1.Enrollments': ['student_id', 'course_id'],
5
6      'test2.employees': ['employee_id', 'name', 'salary', 'age'],
7      'test2.company': ['company_id', 'name', 'city'],
8      'test2.work': ['employee_id', 'company_id']
9  }

```

Query 1 gerada:

```

1  SELECT S.name
2  FROM teste1.Students S
3  JOIN teste1.Enrollments E ON S.student_id = E.student_id
4  JOIN teste1.Courses C ON E.course_id = C.course_id
5  WHERE C.name = 'Database_Systems' AND S.birth_year = 2003;

```

Query 2 gerada:

```

1  SELECT E.name
2  FROM test2.employees E
3  JOIN test2.work W ON E.employee_id = W.employee_id
4  JOIN test2.company C ON W.company_id = C.company_id
5  WHERE E.salary > 1300 AND C.city = 'Oporto' AND E.age < 30;

```


O Llama-3.3-70B-Versatile demonstrou excelente desempenho ao interpretar a estrutura da base de dados e gerar consultas SQL corretas e bem otimizadas. Mesmo em cenários que exijam múltiplos joins e filtros complexos, o modelo conseguiu entregar as respostas corretas e esperadas. Dado o seu alto nível de precisão e eficiência na conversão de linguagem natural em SQL, este modelo foi o escolhido como solução para a aplicação. Com isso, os utilizadores, independentemente do seu conhecimento técnico em base de dados, conseguem aceder e visualizar facilmente as informações desejadas.

5 Implementação

Nesta seção, descrevo a implementação do modelo escolhido no capítulo anterior, bem como as decisões tomadas em relação à interface e ao funcionamento da aplicação.

5.1 Interface

Após a seleção de um modelo eficiente para a transformação das queries em texto para SQL, é necessário o desenvolvimento de uma interface intuitiva e funcional. Para isso, optei pelo Streamlit, uma tecnologia que se destaca pela sua simplicidade, flexibilidade e facilidade na criação de aplicações web interativas com pouco código.

O Streamlit permite uma integração ágil com modelos de IA e bases de dados, tornando-o uma solução ideal para este projeto. Além disso, sua interface dinâmica proporciona uma experiência de utilizador fluida, facilitando a inserção de consultas em linguagem natural e a visualização dos resultados em tempo real.

5.2 Funcionalidades da Aplicação

A aplicação desenvolvida consiste em um chatbot com três funcionalidades principais:

- **Geração de Relatórios** – Criação automatizada de relatórios estruturados com base nos dados fornecidos, permitindo uma análise detalhada e organizada das informações extraídas.
- **Geração de Gráficos** – Visualização interativa dos dados por meio de gráficos dinâmicos, facilitando a interpretação e identificação de padrões nos resultados.
- **Construção de Consultas SQL** – Chat que não apenas gera consultas SQL a partir de linguagem natural, mas também explica a estrutura da query criada, auxiliando o utilizador a compreender melhor o processo e os resultados obtidos.

Essas funcionalidades trabalham em conjunto para oferecer uma experiência intuitiva, permitindo que utilizadores, independentemente do seu conhecimento técnico, consigam explorar e visualizar os dados de maneira eficiente.

5.3 Estrutura do Chatbot

A primeira etapa da aplicação consiste na conexão com a base de dados. Para esta implementação, os detalhes de conexão são passados como parâmetros, pois estamos a utilizar uma base de dados específica. No entanto, para permitir uma maior generalização, o sistema pode ser configurado para que o utilizador forneça suas credenciais de acesso e o token necessário para interagir com o modelo da Groq.

Cada funcionalidade do chatbot foi desenvolvida de forma modular, permitindo ao utilizador escolher entre os três tipos de interação disponíveis. O sistema interpreta as solicitações e executa as ações correspondentes, seja gerando relatórios, criando visualizações gráficas ou construindo consultas SQL personalizadas.

5.3.1 Geração de Relatórios

A funcionalidade de Geração de Relatórios permite que o utilizador obtenha relatórios estruturados com base nos dados disponíveis na base de dados. O processo segue os seguintes passos:

1. O utilizador insere um prompt descrevendo o relatório desejado.
2. A aplicação envia o prompt, juntamente com a estrutura da base de dados, ao modelo da Groq.
3. O modelo gera uma query SQL correspondente à solicitação do utilizador.
4. A aplicação executa a consulta na base de dados e recupera os dados resultantes.

- Os dados são transformados em um DataFrame para facilitar a visualização e interpretação.
- O relatório gerado é apresentado ao utilizador.

Esta abordagem automatiza o processo de criação de relatórios, tornando a análise de dados mais acessível e eficiente.

5.3.2 Geração de Gráficos

A funcionalidade de Geração de Gráficos segue um fluxo semelhante à geração de relatórios, mas com um foco na visualização dos dados. O processo é o seguinte:

- O utilizador insere uma pergunta ou solicitação para gerar um gráfico.
- A aplicação passa o prompt e a estrutura da base de dados ao modelo da Groq.
- O modelo retorna uma query SQL, que é então executada na base de dados.
- Os resultados são transformados em um DataFrame.
- O DataFrame é enviado novamente ao modelo da Groq, solicitando a geração do código necessário para criar um gráfico utilizando Plotly.
- O gráfico é gerado e apresentado na interface.

Esta abordagem permite que utilizadores criem visualizações de dados sem necessidade de conhecimento prévio em SQL ou Plotly, tornando a análise mais intuitiva.

5.3.3 Construção de Consultas SQL

A funcionalidade de Construção de Consultas SQL tem como objetivo facilitar a criação e interpretação de consultas SQL. O processo segue os seguintes passos:

- O utilizador insere um prompt descrevendo a informação que deseja obter da base de dados.
- A aplicação envia o prompt ao modelo da Groq, que retorna:
- A explicação da consulta SQL gerada, detalhando sua estrutura e propósito.
- A interface exibe tanto a explicação quanto a consulta SQL, permitindo ao utilizador compreender e, se necessário, modificar a query antes da execução.

Essa funcionalidade é útil para quem deseja aprender SQL, validar consultas antes da execução ou simplesmente obter consultas otimizadas sem esforço manual.

Exemplos das Interfaces

List all students who are subscribed in the Sistemas de Base de Dados and are born in 2003.

student_id	name
0	1001 João Silva
1	1002 Maria Oliveira
2	1007 Miguel Pereira
3	1009 Ricardo Gomes
4	1032 Laura Ramos
5	1033 Bruno Antunes
6	1038 Eva Almeida
7	1041 Hugo Lima
8	1044 Luísa Soares

View Raw Data

```

[{"id": "1001", "name": "João Silva"}]
[{"id": "1002", "name": "Maria Oliveira"}]
[{"id": "1007", "name": "Miguel Pereira"}]

```

Ask me a question

Figure 1: Exemplo Report Generator



Figure 2: Exemplo Chart Generator

List all students who are subscribed in the Sistemas de Base de Dados and are born in 2003.

Here is the SQL query: To convert the given statement into a SQL query, we need to identify the relevant tables and columns. The statement asks for students who are subscribed to the "Sistemas de Base de Dados" course and were born in 2003.

Given the schema, the relevant tables are:

- `testes.Students` for student information
- `testes.Courses` for course information
- `testes.Enrollments` for enrollment information, linking students to courses

The SQL query to achieve this would be:

```

SELECT s.student_id, s.name
FROM testes.Students s
JOIN testes.Enrollments e ON s.student_id = e.student_id
JOIN testes.Courses c ON e.course_id = c.course_id
WHERE c.name = 'Sistemas de Base de Dados' AND s.birth_year = 2003;

```


This query works as follows:


- It selects the `student_id` and `name` from the `Students` table.
- It joins the `Students` table with the `Enrollments` table on the `student_id` column to link

Ask me a question

Figure 3: Exemplo SQL Generator

Exemplos dos comandos adicionais

 **/explain**

 The results show the number of students enrolled in each course. There are 15 different courses, and the number of students in each course varies.

The course with the most students is 'Sistemas de Base de Dados' with 23 students. On the other hand, 'Segurança da Informação' has the least number of students with 13 students.

Most courses have between 14 to 15 students, with a few having 20 students or more. The results provide a snapshot of the student enrollment in each course, allowing for easy comparison and analysis.

Figure 4: Exemplo do comando /explain

 **/chart**

 Here is the Python code to generate the chart:

```
fig = px.bar(df, x='course_name', y='number_of_students', title='Number of Students')
fig.update_layout(xaxis_title='Course Name', yaxis_title='Number of Students')
```


Figure 5: Exemplo do comando /chart

 **/sql**

 Here is the SQL query:

```
SELECT
  C.course_id,
  C.name AS course_name,
  COUNT(E.student_id) AS number_of_students
FROM
  teste1.Courses C
LEFT JOIN
  teste1.Enrollments E ON C.course_id = E.course_id
GROUP BY
  C.course_id, C.name
ORDER BY
  C.course_id
```

Figure 6: Exemplo do comando /sql

 **/tables**

Database Schema

teste1.Students

- student_id (int)
- name (varchar)
- birth_year (int)

teste1.Courses

- course_id (int)
- name (varchar)
- max_enrollments (int)

teste1.Enrollments

- student_id (int)
- course_id (int)
- FOREIGN KEY (course_id) REFERENCES teste1.Courses(course_id)
- FOREIGN KEY (student_id) REFERENCES teste1.Students(student_id)

Figure 7: Exemplo do comando /tables

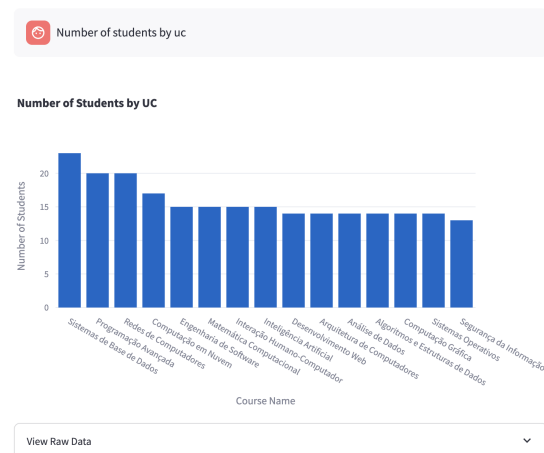


Figure 8: Exemplo do comando /refactor

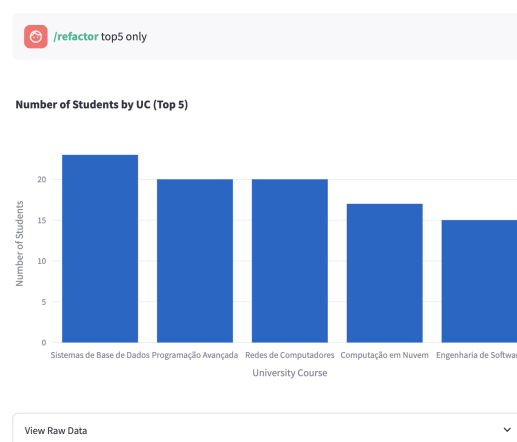


Figure 9: Exemplo do comando /refactor

5.4 Comandos Adicionais

Além das funcionalidades principais descritas anteriormente, foi adicionado ao programa um conjunto de comandos no mesmo estilo do **GitHub Copilot**, com o objetivo de auxiliar os utilizadores que possam ter dúvidas sobre os resultados gerados.

Os comandos implementados são:

- **/sql** – Retorna a query SQL gerada a partir do prompt inserido pelo utilizador.
- **/tables** – Exibe a lista de tabelas disponíveis na base de dados, permitindo ao utilizador compreender melhor a estrutura dos dados.
- **/chart** – Apresenta o código Python utilizado para gerar o gráfico, facilitando a personalização e compreensão da visualização.

- **/explain** – Explica os resultados obtidos no DataFrame gerado a partir da query executada na base de dados, ajudando o utilizador a interpretar os dados retornados.
- **/refactor** – Permite modificar o último prompt inserido com novas informações.
- **/reset** – Elimina toda a conversa da interface em que estiver, permitindo recomeçar uma nova conversa sem histórico anterior.
- **/help** – Exibe a mensagem de ajuda com todos os comandos disponíveis.

Estes comandos foram projetados para melhorar a experiência do utilizador, tornando a aplicação mais interativa e acessível. Com eles, o utilizador pode obter informações detalhadas sobre as consultas geradas, os dados disponíveis e a lógica por trás das respostas fornecidas pelo sistema. O comando ****/reset****, em particular, facilita a reinicialização da conversa, garantindo um ambiente limpo para novas interações.

6 Testes

Para avaliar o desempenho do modelo na conversão de linguagem natural em SQL, realizamos vários testes utilizando a base de dados previamente descrita. A seguir, explico os testes realizados, as consultas em linguagem natural fornecidas ao modelo e as respectivas consultas SQL geradas, a fim de verificar a precisão e a adequação dos resultados.

6.1 Teste 1: Consulta sobre Alunos e Cursos

O primeiro teste usa uma base de dados relacionada ao contexto acadêmico e contém informações sobre alunos, cursos e matrículas.

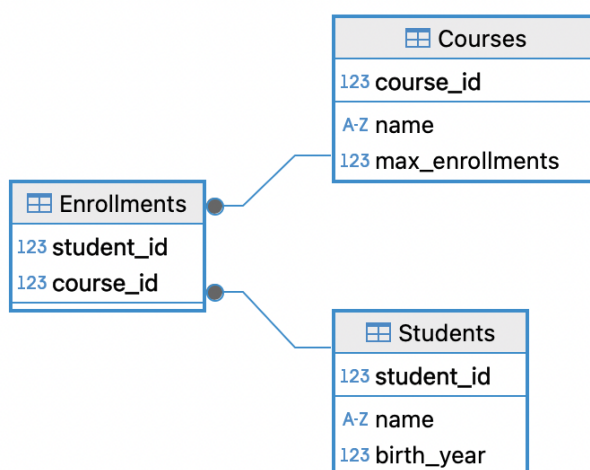


Figure 10: Diagrama Entidade Relação

A base dados é composta pelas seguintes tabelas:

- **Students** – Armazena informações sobre os alunos.
- **Courses** – Armazena informações sobre os cursos oferecidos.
- **Enrollments** – Armazena informações sobre as matrículas dos alunos nos cursos.

Essas tabelas estão interligadas através dos campos `student_id` e `course_id`, permitindo consultas sobre alunos matriculados em cursos específicos ou informações sobre cursos com capacidade máxima de matrículas.

6.1.1 Consulta: Listar todos os alunos matriculados em Sistemas de Base de Dados e nascidos em 2003

A seguir, apresentamos os resultados da consulta solicitada. Mostramos as duas interfaces que geram os resultados: a primeira apresenta um relatório com os dados, e a segunda fornece uma explicação detalhada sobre o código SQL gerado.



Figure 11: Resposta gerada pelo SQL Generator

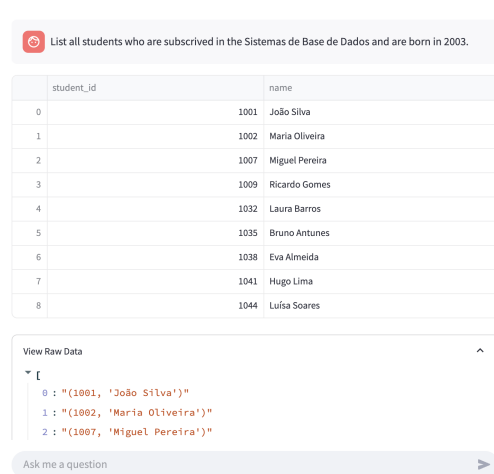


Figure 12: Resposta gerada pelo Report Generator

Explicação da Imagem 11 (SQL Generator): A primeira imagem (Figura 11) mostra a consulta gerada pelo modelo.

```
SELECT s.student_id, s.name
FROM teste1.Students s
JOIN Enrollments e ON s.student_id = e.student_id
JOIN Courses c ON e.course_id = c.course_id
WHERE c.name = 'Sistemas de Base de Dados' AND S.birth_year = 2003;
```

Além da consulta, a imagem apresenta uma explicação detalhada sobre o que cada parte da consulta faz, garantindo uma compreensão completa do processo de construção da query. A explicação fornece informações sobre a junção das tabelas `Students`, `Enrollments` e `Courses`, e a filtragem dos alunos de acordo com o curso e o ano de nascimento.

Explicação da Imagem 12 (Report Generator): A segunda imagem (Figura 12) mostra o resultado gerado pelo sistema Report Generator. Ao contrário da consulta SQL, que apresenta o código da query, o Report Generator exhibe uma tabela com os alunos que atendem aos critérios especificados na consulta. A tabela contém os nomes dos alunos que estão matriculados no curso Sistema de Base de dados e que nasceram em 2003, proporcionando uma visão direta e mais acessível dos resultados.

Ambas as abordagens geraram os resultados esperados, com a primeira focada na explicação da consulta SQL e a segunda fornecendo uma visualização prática e direta dos dados.

6.1.2 Consulta: Lista a unidade curricular com maior numero de inscritos, os seus nomes e datas de nascimento.

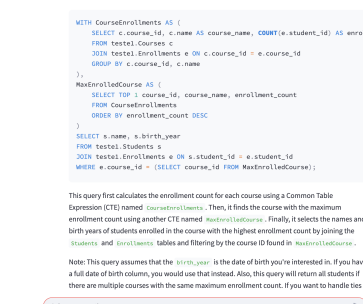
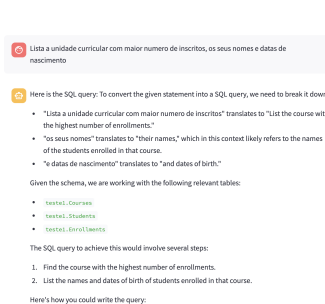


Figure 14: Resposta gerada pelo SQL Generator

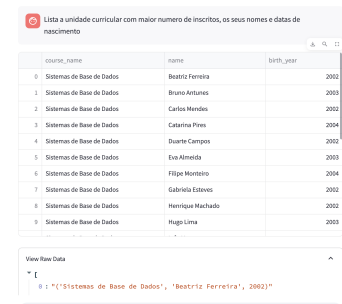


Figure 15: Resposta gerada pelo Report Generator

Explicação da Imagem 13 e Imagem 14 (SQL Generator): A primeira imagem (Figura 11) mostra a consulta gerada pelo modelo. Além da consulta, a imagem apresenta uma explicação detalhada sobre a consulta, garantindo uma compreensão completa do processo de construção da query.

Explicação da Imagem 15 (Report Generator): A imagem (Figura 15) mostra o resultado gerado pelo sistema Report Generator. Ao contrário da consulta SQL, que apresenta o código da query, o Report Generator exibe uma tabela com os alunos que atendem aos critérios especificados na consulta.

Novamente, ambas as abordagens geraram os resultados esperados, com a primeira focada na explicação da consulta SQL e a segunda fornecendo uma visualização prática e direta dos dados.

6.1.3 Testes ao SQL Generator

Após os testes realizados ao SQL Generator e ao Report Generator, realizei testes ao Chart Generator. Esta funcionalidade permite gerar gráficos com base na pergunta do utilizador, analisando a consulta realizada e apresentando a visualização correspondente.



Figure 16: Resposta gerada pelo Chart Generator

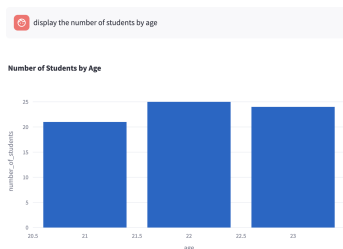


Figure 17: Resposta gerada pelo Chart Generator

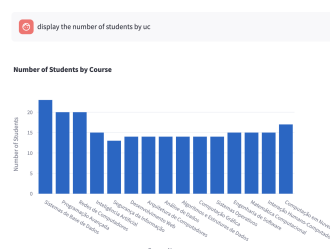


Figure 18: Resposta gerada pelo Chart Generator

Imagem 16 Número de estudantes por ano de nascimento (Chart Generator): A primeira imagem (Figura 16) apresenta o gráfico gerado pelo modelo, refletindo corretamente a distribuição do número de estudantes por ano de nascimento, conforme esperado.

Imagem 17 Número de estudantes por idade (Chart Generator): A Figura 17 exibe a distribuição do número de estudantes por idade, uma tarefa mais complicada uma vez que temos apenas na tabela a data de nascimento de cada indivíduo.

Imagem 18 Número de estudantes por Unidade Curricular (Chart Generator): Na Figura 18, é apresentado um gráfico que mostra a distribuição do número de estudantes por Unidade Curricular (UC).

6.2 Teste 2: AdventureWorks2012 Database

O segundo teste realizado recorre à base de dados AdventureWorks, amplamente utilizada para fins educativos e de prática com SQL. Esta base de dados simula o funcionamento de uma empresa fictícia de fabrico e venda de bicicletas, disponibilizando um conjunto rico de tabelas relacionadas com clientes, encomendas, produtos, funcionários, entre outras.

Para a elaboração deste teste, foram utilizadas perguntas disponíveis no seguinte recurso online, **AdventureWorks SQL Questions – LearnSQL.com [12]**

6.2.1 Questões Utilizadas

As perguntas utilizadas neste teste foram extraídas do recurso online *AdventureWorks SQL Questions – LearnSQL.com [12]*, abrangendo diferentes níveis de complexidade (Fácil, Médio e Difícil). Estas questões centram-se maioritariamente na área de Recursos Humanos (Figura 19) e permitiram avaliar diversas competências em SQL. Esta variedade permitiu testar a capacidade do modelo na geração de consultas SQL a partir da estrutura dos dados fornecida, possibilitando ainda a comparação entre os resultados obtidos pelas queries geradas automaticamente e as soluções disponibilizadas pelo site de origem. Abaixo apresenta-se a lista completa de perguntas:

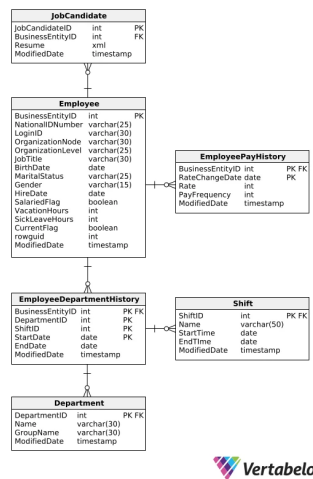


Figure 19: Diagrama Entidade Relação Adventure Works

1. Selecionar o cargo (job title) de todos os funcionários do sexo masculino que não são casados.
2. Selecionar o BusinessEntityID, a taxa (Rate) e a data de alteração da taxa (RateChangeDate) para todos os funcionários cuja taxa de pagamento tenha sido, em algum momento, igual ou superior a 50.
3. Selecionar o BusinessEntityID, o DepartmentID e a data de início (StartDate) para cada funcionário que tenha começado a trabalhar em qualquer departamento durante o ano de 2008.
4. Selecionar os IDs, nomes e nomes dos grupos dos departamentos cujo nome começa por "Prod" OU cujo nome do grupo termina em "ring".
5. Selecionar os nomes dos departamentos que pertencem ao grupo "Research and Development" ou ao grupo "Manufacturing".
6. Selecionar os IDs dos funcionários juntamente com todos os nomes dos departamentos onde trabalharam em qualquer momento.
7. Selecionar o ID e o cargo (job title) do funcionário, juntamente com as datas de mudança de departamento (StartDate), apenas para funcionárias do sexo feminino.
8. Selecionar os cargos (job titles) e os respectivos nomes dos departamentos, de forma a identificar todos os cargos que foram alguma vez utilizados em cada departamento. Não incluir duplicados.
9. Selecionar os nomes distintos de departamentos e turnos (shifts) em que trabalham os funcionários. Renomear as colunas para DepartmentName e ShiftName.
10. Selecionar os IDs dos funcionários, nomes dos departamentos e nomes dos turnos, incluindo apenas funcionários contratados após 01-01-2010 e que trabalham em departamentos pertencentes aos grupos "Manufacturing" ou "Quality Assurance".
11. Selecionar o número mínimo e máximo de horas de baixa médica (sick leave) tiradas pelos funcionários.
12. Selecionar os cargos e o número médio de horas de férias (vacation hours) por cargo.
13. Selecionar o sexo dos funcionários e a contagem de funcionários por cada género.
14. Contar o número de departamentos em cada grupo de departamentos. Listar apenas os grupos de departamentos que têm mais de dois departamentos.
15. Selecionar os nomes dos departamentos e a soma de horas de baixa médica tiradas pelos funcionários que atualmente trabalham em cada departamento. Renomear esta coluna para SumSickLeaveHours.
16. Selecionar os IDs dos funcionários e as suas taxas de pagamento atuais.
17. Selecionar os valores mínimo, médio e máximo das taxas de pagamento atuais dos funcionários.

18. Selecionar os IDs dos funcionários que tiraram mais de 60 horas de férias ou mais de 60 horas de baixa médica.
19. Selecionar os IDs dos funcionários que têm os cargos "Sales Representative" ou "Tool Designer" e que tenham trabalhado (ou estejam a trabalhar) nos departamentos de Vendas (Sales) ou Marketing. Ordenar por valor, do mais alto para o mais baixo.
20. Selecionar os IDs, cargos e nomes dos departamentos dos funcionários associados às funções de "Sales Representative" ou "Marketing Manager".

6.2.2 Resultados Obtidos

Para realizar a comparação, comecei por gerar as 20 consultas SQL utilizando o meu algoritmo. Cada query gerada foi então comparada com as consultas de referência fornecidas pelo site, que representavam as respostas esperadas. O objetivo era verificar a precisão e a eficácia das consultas geradas, avaliando se os resultados estavam de acordo com os dados esperados. As consultas foram avaliadas de acordo com os seguintes critérios:

- Equal: Quando os resultados retornados correspondiam exatamente aos dados esperados, independentemente de variações em nomes de colunas ou ordenações.
- Correct+: Quando a query retornava todos os dados esperados, com a adição de colunas extras que não comprometeram a integridade dos dados.
- Correct-: Quando a consulta retornava dados corretos, com uma correspondência superior a 90% dos dados esperados, mas com pequenas variações na estrutura da resposta.
- Incorrect: Quando a consulta retornava dados completamente diferentes dos esperados, com falhas significativas que indicavam uma abordagem errada na construção da query, resultando em um conjunto de dados que não correspondia à intenção da pergunta original.
- No Results: Quando nenhuma das query retornava nenhum dado, o que indicava um erro no próprio resultado de comparação.

Após a execução e comparação das 20 queries, os seguintes resultados foram observados:

- 15 queries (75%) foram classificadas como "Equal", o que significa que os dados retornados coincidiam exatamente com os dados esperados. Embora em alguns casos houvesse variações nos nomes das colunas ou na ordem delas, as informações retornadas eram consistentes com o que se esperava.
- 1 query (Q2) recebeu a classificação "Correct+", além de retornar todos os dados esperados, incluiu colunas adicionais que enriqueceram a resposta, sem comprometer a integridade dos dados principais.
- 1 query (Q6) foi classificada como "Correct-", indicando que os dados estavam corretos em +90%, mas houve uma pequena diferença em relação ao esperado. Isto pode ter ocorrido devido a uma exceção ausente ou uma comparação incorreta, resultando em uma ligeira discrepância nos resultados.
- 2 queries (Q3 e Q10) resultaram em "No Results", ou seja, não retornaram dados. Esse resultado pode ser atribuído a filtros muito restritivos ou a aplicação de intervalos de datas muito específicos, que não correspondiam aos dados presentes na base de dados.

Os resultados demonstram que o algoritmo foi eficaz em gerar as consultas SQL e que a maioria das queries geradas produziu resultados idênticos aos esperados. A alta taxa de acerto (75%) sugere uma sólida compreensão das técnicas de SQL, especialmente no uso de JOIN, filtros e agregações. As discrepâncias observadas em algumas consultas podem ser atribuídas a diferenças na forma de apresentação dos dados ou a filtros que não encontraram correspondência nos dados da base de dados.

6.3 Teste 3: TPC-H Database

O terceiro teste realizado utilizou a base de dados TPC-H, um benchmark amplamente utilizado para avaliar o desempenho de sistemas de processamento de consultas complexas. A TPC-H simula um ambiente de negócios de uma empresa de vendas de produtos e serviços, com um conjunto de dados que abrange várias tabelas relacionadas a vendas, clientes, linha de pedidos, entre outros.

Para a elaboração deste teste, foram utilizadas questões disponíveis no recurso online **TPC-H Benchmark Queries – TPC.org [13]**. Essas questões foram formuladas com o objetivo de avaliar o desempenho das consultas em diferentes áreas de processamento de dados, incluindo operações de agregação, join e filtragem, simulando cenários complexos de consultas analíticas.

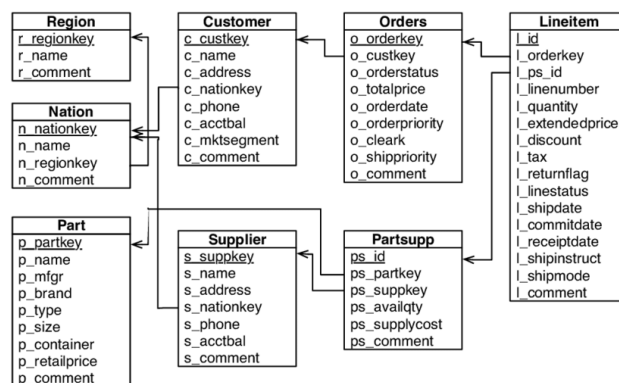


Figure 20: TPC-H Diagrama Entidade Relação

6.3.1 Questões Utilizadas

As questões utilizadas neste teste foram extraídas do benchmark TPC-H, abordando diversas áreas de análise de dados. Essas questões envolvem desde consultas simples de agregação até consultas complexas que utilizam múltiplos joins e agrupamentos. As consultas foram utilizadas para avaliar a eficácia do modelo na geração de queries SQL que correspondessem aos resultados esperados. Abaixo apresenta-se uma lista parcial de perguntas:

1. Selecionar o total de vendas por cada cliente, ordenado pela maior quantidade de vendas.
2. Calcular a média e o desvio padrão das vendas por segmento de mercado.
3. Identificar os produtos que tiveram mais de 50 vendas no último trimestre.
4. Calcular a receita total por ano e por região.
5. Identificar os 10 produtos mais vendidos durante o ano, com base no total de unidades vendidas.
6. Calcular o valor médio das transações para cada ano.
7. Identificar os 5 fornecedores com as maiores vendas no último ano.
8. Calcular o valor total das vendas de produtos em regiões específicas.
9. Identificar os 10 clientes que mais gastaram com a compra de produtos no último trimestre.
10. Calcular a variação anual nas vendas por produto.
11. Calcular a soma das vendas de cada produto por trimestre.
12. Obter a média de vendas por categoria de produto para cada ano.
13. Calcular o valor total de vendas para os produtos que foram vendidos por mais de um fornecedor.
14. Selecionar a receita total por ano e por categoria de produto.
15. Calcular o número de pedidos realizados por cliente e a média de itens por pedido.
16. Obter os produtos que representam mais de 5% da receita total por categoria.
17. Calcular a quantidade de pedidos realizados por cada cliente durante o ano.
18. Identificar as regiões com a maior receita total no último ano.
19. Calcular o total de vendas por região e por categoria de produto.
20. Calcular o número de transações por cliente em cada região.

6.3.2 Resultados Obtidos

Para a execução do teste, as consultas SQL foram geradas automaticamente utilizando o algoritmo proposto. Cada query gerada foi comparada com as consultas de referência fornecidas pelo site, que continham as respostas esperadas. O objetivo era verificar a precisão e a eficácia das consultas geradas, avaliando se os resultados estavam de acordo com os dados fornecidos na base TPC-H.

Após a execução e comparação das 22 queries, os seguintes resultados foram observados:

- 17 queries (77%) foram classificadas como "Equal", indicando que os dados retornados coincidiram exatamente com os dados esperados. Embora houvesse algumas variações nos nomes das colunas ou na ordem delas, as informações retornadas foram consistentes com o que se esperava.
- 3 queries (Q3, Q7 e Q21) receberam a classificação "Correct-", pois os dados retornados estavam corretos, com mais de 90% de correspondência com os resultados esperados, mas com pequenas diferenças.
- 2 queries (Q9 e Q20) foram classificadas como "Incorrect", ou seja, retornaram dados completamente diferentes do esperado, indicando algumas falhas, provavelmente na agregação.

Estes resultados demonstram que o algoritmo foi eficaz na geração das consultas SQL e que a maioria das queries geradas produziu resultados idênticos aos esperados. A elevada taxa de acerto (77%) sugere novamente uma boa compreensão da estrutura da base de dados TPC-H e das operações SQL necessárias para recuperar os dados de forma eficiente. As discrepâncias observadas em algumas consultas podem ser atribuídas a pequenas variações na forma de apresentação dos dados ou a diferenças na interpretação dos requisitos das perguntas.

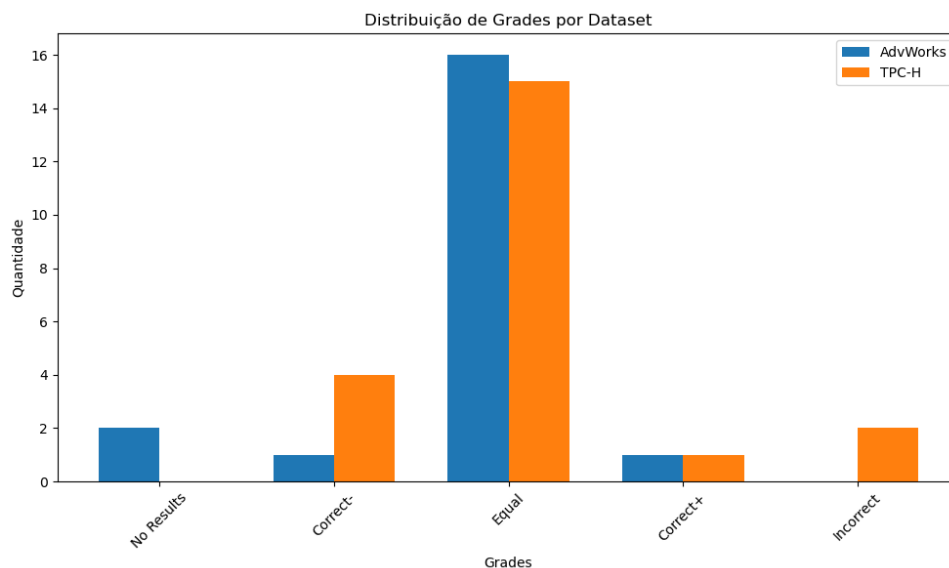


Figure 21: Gráfico de acerto dos testes

7 Conclusões

Nos dias de hoje, a inteligência artificial (IA) tem revolucionado o mercado de diversas formas, trazendo tanto benefícios quanto desafios. Em áreas como a análise de dados e a manipulação de bases de dados, a IA tem se mostrado uma poderosa aliada, ao permitir que pessoas sem formação técnica interajam de maneira simples e intuitiva com grandes volumes de dados, a IA está a facilitar a tomada de decisões, automatizando processos e, principalmente, abrindo portas para que mais pessoas possam tirar proveito do vasto potencial das bases de dados.

No entanto, a aplicação de IA nesse campo não está isenta de dificuldades. Um dos maiores desafios é garantir que os modelos consigam traduzir perguntas em linguagem natural para consultas SQL precisas, especialmente em cenários mais complexos que envolvem múltiplas tabelas e operações avançadas. O caminho até encontrar a solução ideal para o meu projeto foi marcado por vários testes e tentativas de diferentes modelos, até finalmente encontrar uma solução robusta com o Llama-3.3-70b-versatile da Groq. Este modelo mostrou-se capaz de lidar de forma eficiente com as complexidades das consultas SQL, oferecendo respostas mais precisas e adequadas às perguntas feitas.

O objetivo principal deste projeto foi criar uma ferramenta acessível que permitisse a qualquer pessoa, mesmo sem conhecimentos técnicos em bases de dados, gerar consultas SQL a partir de perguntas em linguagem natural. Ao longo do desenvolvimento, foram implementados diversos testes para garantir que a aplicação não apenas gerasse consultas corretas, mas também fosse capaz de recuperar os dados de forma eficaz e apresentar gráficos de maneira precisa. Para melhorar ainda mais a experiência do utilizador, a aplicação inclui funcionalidades adicionais que ajudam na navegação e no entendimento dos dados, como já falado anteriormente.

Estas funcionalidades contribuem para uma experiência mais fluida e intuitiva, permitindo aos utilizadores não apenas obter as respostas que procuram, mas também entender como essas respostas são geradas e apresentadas. Assim, a IA no contexto deste projeto não apenas cumpre a função de facilitar a geração de queries, mas também de educar e apoiar os utilizadores no processo da análise de dados.

Em suma, este projeto é um exemplo de como a inteligência artificial pode ser aplicada para resolver problemas reais e complexos, tornando ferramentas avançadas de análise de dados acessíveis para todos.

References

- [1] GraphMaker, "Graphmaker ai chat," 2025, acessado em: 27 Fev. 2025. [Online]. Available: <https://www.graphmaker.ai/chat>
- [2] R. Mitra, A. Narechania, A. Endert, and J. Stasko, "Facilitating conversational interaction in natural language interfaces for visualization," in *Proceedings of the IEEE Visualization Conference (VIS)*, 2022, ferramenta NL4DV para interação em linguagem natural com visualizações. [Online]. Available: <https://arxiv.org/pdf/2207.00189>
- [3] Text2SQL.ai, "Text2sql.ai: Natural language to sql conversion," 2025, acessado em: 27 Fev. 2025. [Online]. Available: <https://www.text2sql.ai>
- [4] Y. LILY, "Spider: A large-scale text-to-sql benchmark," 2025, acessado em: 27 Fev. 2025. [Online]. Available: <https://yale-lily.github.io/spider>
- [5] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, S. Ruder, G. Gauthier, H. Schwenk, and A. M. Rush, "Hugging face's transformers: State-of-the-art natural language processing," <https://arxiv.org/abs/1910.03771>, 2019.
- [6] OpenAI, "Chatgpt: Conversational ai model," 2025, acessado em: 27 Fev. 2025. [Online]. Available: <https://openai.com/chatgpt>
- [7] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," <https://arxiv.org/abs/1910.10683>, 2019.
- [8] cssupport, "t5-small-awesome-text-to-sql," <https://huggingface.co/cssupport/t5-small-awesome-text-to-sql>, 2025.
- [9] juiererror, "text-to-sql-with-table-schema," <https://huggingface.co/juiererror/text-to-sql-with-table-schema>, 2025.
- [10] —, "flan-t5-text2sql-with-schema-v2," <https://huggingface.co/juiererror/flan-t5-text2sql-with-schema-v2>, 2025.
- [11] G. Inc., "Llama-3.3-70b-versatile," <https://huggingface.co/groq/llama-3.3-70b-versatile>, 2025.
- [12] LearnSQL.com, "Sql practice for beginners: 20+ exercises with solutions," julho 2022, acedido em abril de 2025. [Online]. Available: <https://learnsql.com/blog/sql-practice-for-beginners-adventureworks-exercises/>
- [13] Transaction Processing Performance Council, "Tpc-h: Decision support benchmark," <https://www.tpc.org/tpch/>, 2021, accessed: 2025-04-11.