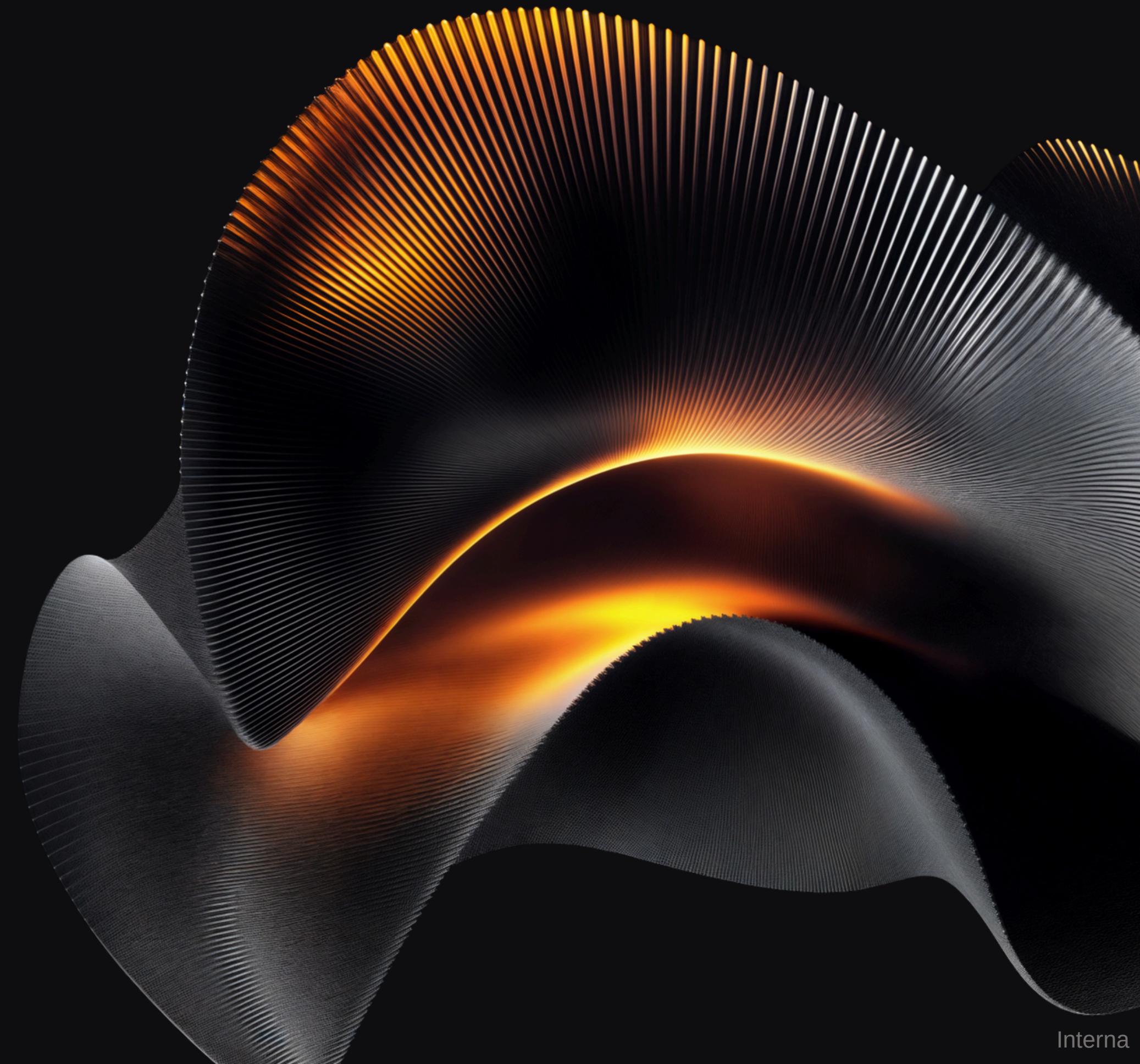




DESAFIO HACKATHON - SQL CONVERTER

Squad 6

part of AI&R



INTEGRANTES DA SQUAD

ANA CLAUDIA SGANZERLA

ALYSSON DE ARAÚJO BITENCOURT

FREDERICO TAUFER

TIAGO SOARES

DÉBORA LARISSA ARRUDA

WALMIR ROSA

YASMIN CONSTANTINO

VITOR FREGULIA

JOÃO VICTOR LEMES

OBJETIVO DO PROJETO

- TRADUZIR CONSULTAS SQL EM CÓDIGO EQUIVALENTE DO PYSPARK DATAFRAME API OU SPARKSQL.
- ÚTIL PARA: ENTENDER A EQUIVALÊNCIA ENTRE SQL E APIs DO PYSPARK E AUTOMATIZAR MIGRAÇÕES DE SCRIPTS SQL.
- ECONOMIZA TEMPO AO MIGRAR SCRIPTS SQL LEGADOS PARA PROJETOS QUE USAM A API PYSPARK.

CONVERSOR SQL → PYSPARK/SPARKSQL: AUTOMATIZANDO TRADUÇÃO DE QUERIES PARA BIG DATA

A FERRAMENTA CONVERTE SCRIPTS SQL EM COMANDOS EQUIVALENTES EM PYSPARK E SPARKSQL, OTIMIZANDO O DESENVOLVIMENTO DE PIPELINES EM AMBIENTES DISTRIBUÍDOS E REDUZINDO ERROS MANUAIS EM AMBIENTES COM GRANDES VOLUMES DE DADOS.

100%

COMPATÍVEL COM ARQUIVOS DE EXTENSÃO .SQL

200+

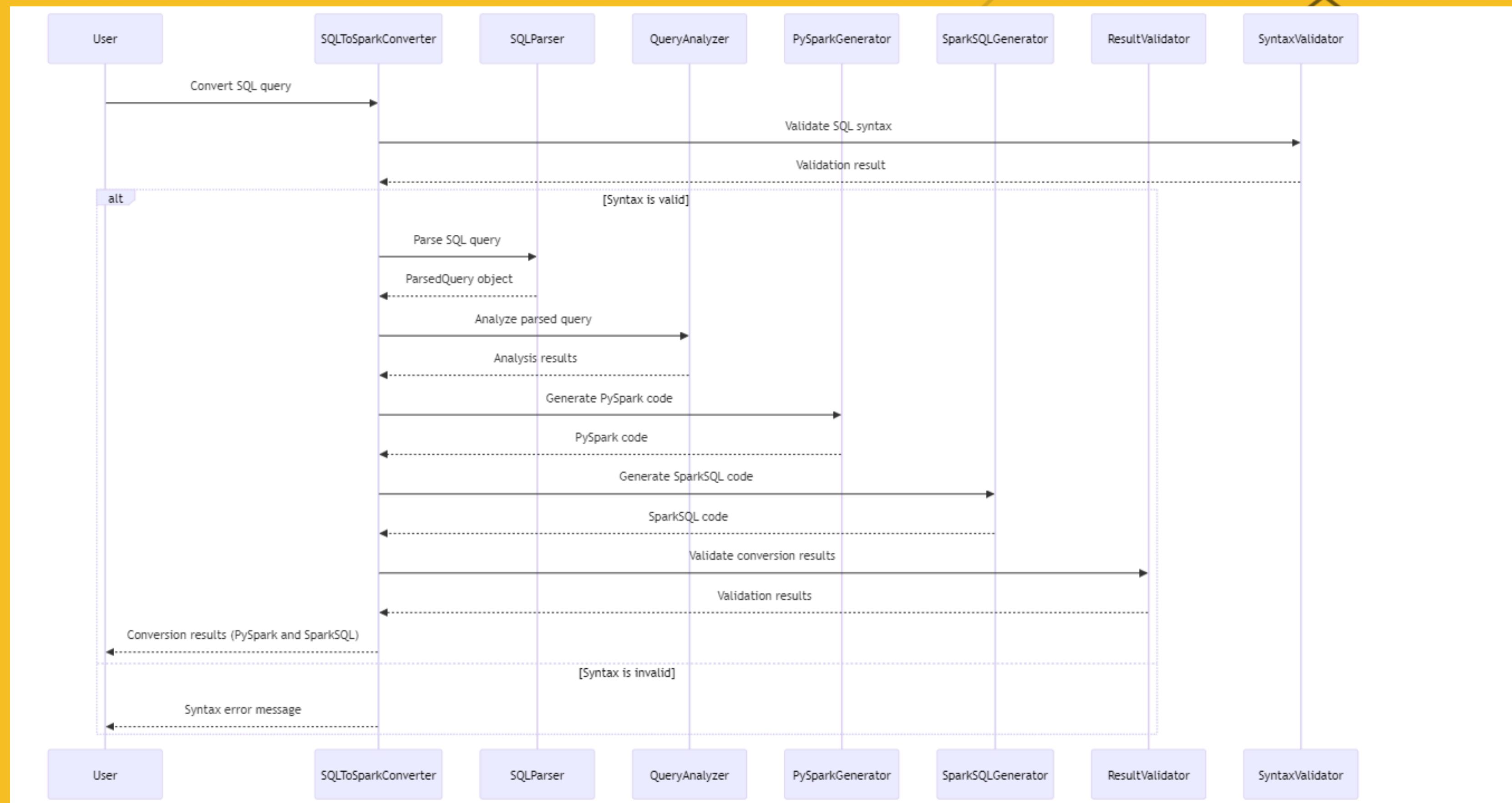
LINHAS DE CÓDIGO CONVERTIDAS COM SUCESSO

99%

DE PRECISÃO MÉDIA NA CONVERSÃO DE QUERIES

3MS

TEMPO MÉDIO PARA CADA CONVERSÃO E VALIDAÇÃO DOS SCRIPTS



FEATURES SQL SUPORTADAS:

- SELECT (PARA ESCOLHER COLUNAS)
- JOIN (PARA UNIR TABELAS)
- WHERE (PARA FILTRAR LINHAS)
- GROUP BY (PARA AGRUPAR)
- ORDER BY (PARA ORDENAR)
- HAVING (PARA FILTRAR GRUPOS)
- SUBQUERIES (CONSULTAS DENTRO DE CONSULTAS)
- WINDOW FUNCTIONS (FUNÇÕES DE JANELA)
- AGGREGATE FUNCTIONS (FUNÇÕES AGREGADAS)
- CASE (PARA CONDIÇÕES DENTRO DO SELECT)
- ALIASES (PARA NOMES TEMPORÁRIOS DE CAMPOS)

COMO PENSAMOS NA SOLUÇÃO ?

UNINDO NOSSAS EXPERIÊNCIAS DE DIFERENTES PB'S UTILIZAMOS PROMPTS
USANDO AI COCKPIT E AMAZON Q PARA ESTRUTURAR O A LÓGICA DO
PROJETO E CONSTRUIR A INTERFACE.

```
sql_to_spark_converter/
├── src/
│   ├── __init__.py
│   ├── parser/
│   │   ├── __init__.py
│   │   ├── sql_parser.py
│   │   └── query_analyzer.py
│   ├── generators/
│   │   ├── __init__.py
│   │   ├── pyspark_generator.py
│   │   └── sparksql_generator.py
│   ├── validators/
│   │   ├── __init__.py
│   │   ├── syntax_validator.py
│   │   └── result_validator.py
│   └── utils/
│       ├── __init__.py
│       ├── templates.py
│       └── helpers.py
└── static/
    ├── style.css
    └── script.js
└── templates/
    └── index.html
examples/
├── simple_query.sql
└── complex_query.sql
└── join_query.sql
app.py
main.py
demo.py
test_converter.py
requirements.txt
README.md
```

Como nossa solução foi pensada?

- 1 Criar um parser SQL**
- 2 Criar um ParsedQuery**
- 3 Gerador de código PySpark**
- 4 Gerador de código Spark SQL**

