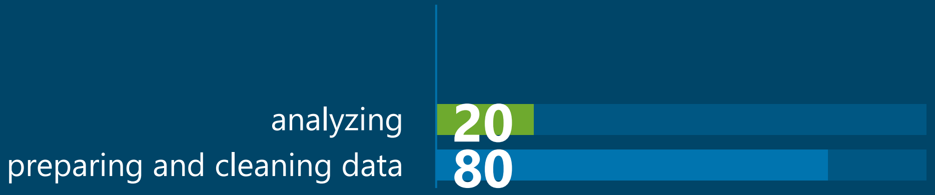
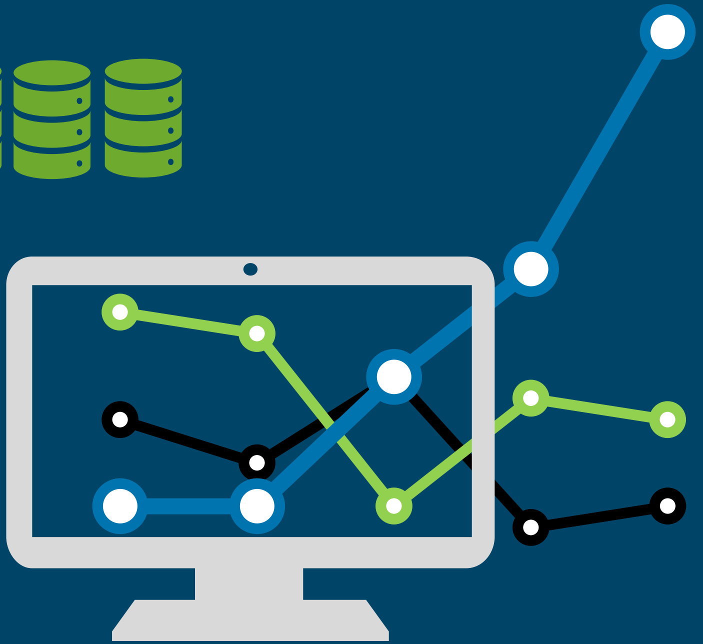
Telco Analytics AI System

Over 70% of enterprise data goes unused for analytics





Project Overview & Business Value

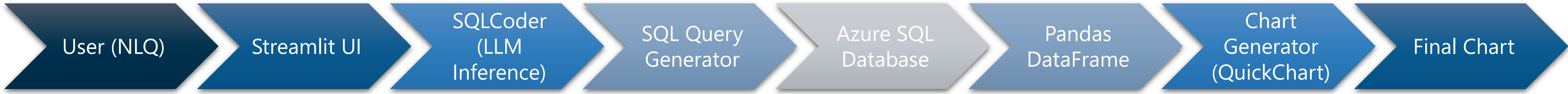
Objective: Enable telecom stakeholders to query business data using natural language.
Solution: A full-stack AI system that translates natural language questions into SQL, executes them on Azure SQL, and visualizes the result.
Business Value:

- Empowers non-technical users to query data easily
- Speeds up telecom business analysis and decision-making
- Bridges the gap between business questions and data insights using LLMs

Tech Stack:

- **SQL generation:** defog/sqlcoder-7b (GGUF quantized version)
- **Backend:** FastAPI + Streamlit
- **Charting:** QuickChart API
- **DB:** Azure SQL (data uploaded via Azure Data Studio)

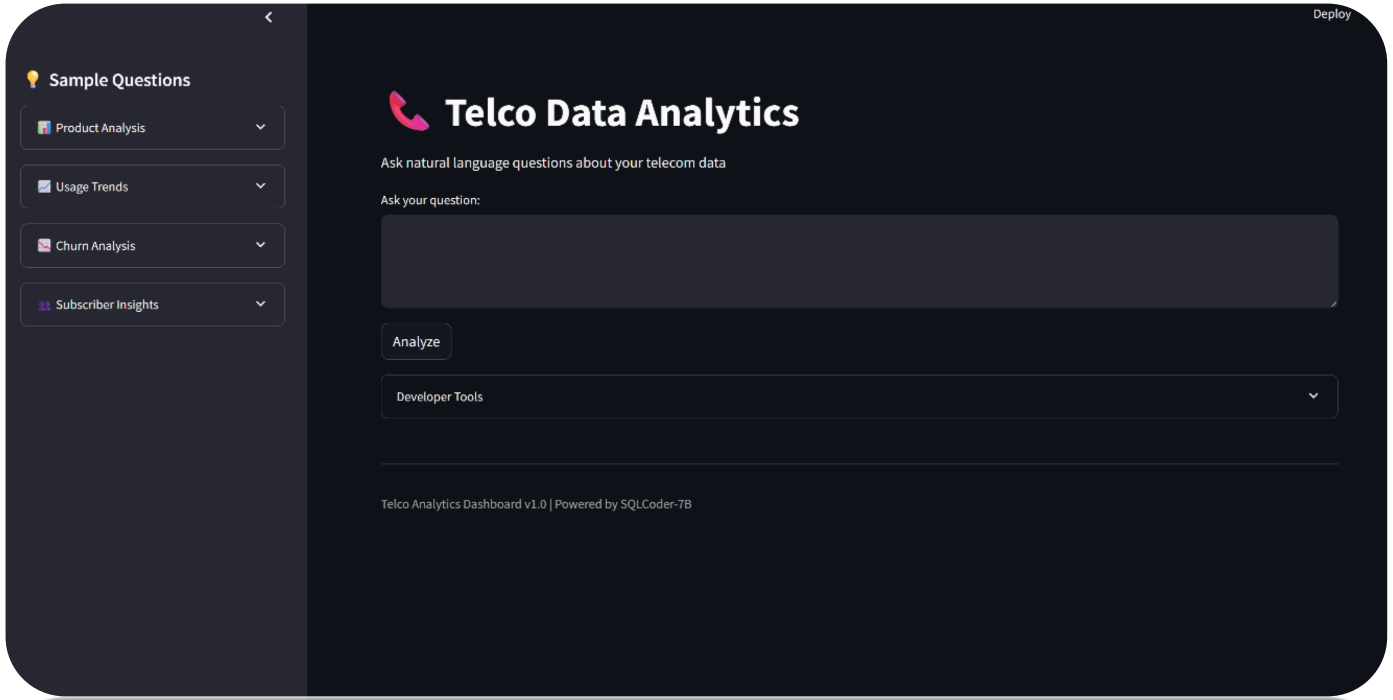
System Architecture



Project Lifecycle

User Input: Natural Language Query
SQL Generation: Prompt passed to SQLCoder (LLM)
Execution: Query executed on Azure SQL
Post-Processing: DataFrame returned
Charting: Visualization generated
Monitoring: Query performance tracked

Model Used: defog/sqlcoder-7b
Variant: Quantized sqlcoder-7b.Q4_K_M.gguf via llama-cpp for local inference
Input: Natural language telecom-related query
Output: Validated T-SQL queries compatible with Azure SQL



Model Enhancements and Validation Summary

SQL Processing Workflow

1. Input Sanitization

- Remove markdown code blocks (sqL)
- Trim whitespace
- Validate query length (> 20 characters)

2. Syntax Correction

- Fix common SQL syntax issues:
  - → Malmared or msissing liases
  - → Unmatched brackets
  - → Stray characters
  - → Missing JOIN cornditions
  - → Inconsistent spacing

Domain-Specific Mapping Rules

Category	Validatrion	Example
Basic Structure	Starts with SELECT or `WITH`	SELECT or WITH
Table Reference	[dbo]_[table] format	ON clause
JOIN Conditions	Using T-SQL, not PostgreSQL	Blocked con connands
Security	Blocked SQL	

Prompt Engineering Enhancements

- Inject schema context dynamically
- + 22 hand-crafted T-SQL rules
- 5 few-shot examples
- Classify question type to guide generation

Post-Processing Steps

- Convert PostgreSQLoutputs to T-SQL

Demo Scenarios

- 📈 *“Show monthly data usage trends for the past year”*
- 📉 *“What are the top churn reasons last year?”*
- 📊 *“Top 5 most expensive products”*

Edge Cases & Error Handling

Case	Behavior
Invalid SQL	Triggers "Execution Error" return
Chart fails	Returns valid SQL/data but reports "Visualization Error"
No results	Returns: No results found
Model fails	Returns: Generation Error

Metrics & Monitoring

Metric	Calculation	Storage
Success Rate	(Successful Executions / Total Queries) × 100	session_metrics
Avg Generation Time	Σ(Generation Times) / Count	response_times
Error Frequency	Error Type Counts	common_errors

Conclusion and Future Enhancements

Conclusion
Built a full working MVP of an LLM-powered telecom data analytics platform.
Enables non-technical users to access business insights.
Future Enhancements
1) Currently using quantized GGUF due to limited GPU (4GB). Will move to original defog/sqlcoder-7b requiring 6GB+ VRAM.
2) Fine-tuned model on telecom schema.
3) To save dashboards and query history create user authentication
4) Support JOIN-heavy questions