Telco Analytics Dashboards Generator — Documentation

# Project Overview & Business Value

The **Telco Analytics Dashboards Generator** is an interactive NLP-driven BI tool that allows business users to explore and analyze telecom data using natural language. It auto-generates SQL queries, fetches insights from an Azure-hosted database, and visualizes results in real-time.

**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

# System Architecture

User (NLQ) → Streamlit UI

↓

SQLCoder (LLM Inference)

↓

SQL Query Generator

↓

Azure SQL Database

↓

Pandas DataFrame

↓

Chart Generator (QuickChart)

↓

Final Insight + Chart

## Core Components

1. **Frontend Layer**
   * Streamlit-based UI with:
     + Natural language input
     + Sample questions
     + Interactive results display
     + Developer debug panel
     + Icons to make it more user friendly
2. **NL-to-SQL Engine**
   * SQLGenerator class (llama-cpp)
   * Schema-aware prompt engineering
   * T-SQL syntax validation
   * Query performance metrics
3. **Data Layer**
   * Azure SQL Database connection
   * 11 normalized tables (7 dimension, 4 fact)
   * PyODBC execution handler
4. **Visualization**
   * ChartGenerator class
   * QuickChart.io integration
   * Automatic chart type detection

# Setup Instructions (Local Deployment)

## Requirements

pip install streamlit==1.28.0 pandas==2.0.3 pyodbc==4.0.39 llama-cpp-python==0.2.20

## Model Configuration

1. Download sqlcoder-7b.Q4\_K\_M.gguf from TheBloke
2. Place in ./models/ directory
3. Configure in SQLGenerator class initialization

## Database Setup

1. Configure azure\_sql.py with:

connection\_string = (

"Driver={ODBC Driver 18 for SQL Server};"

"Server=your\_server.database.windows.net;"

"Database=your\_db;"

"UID=your\_username;"

"PWD=your\_password;"

)

# Data Overview

* Data originated from a **telecom Excel file**, where each sheet was treated as a table.
* The Excel data was uploaded into **Azure SQL Database** using **Azure Data Studio**.

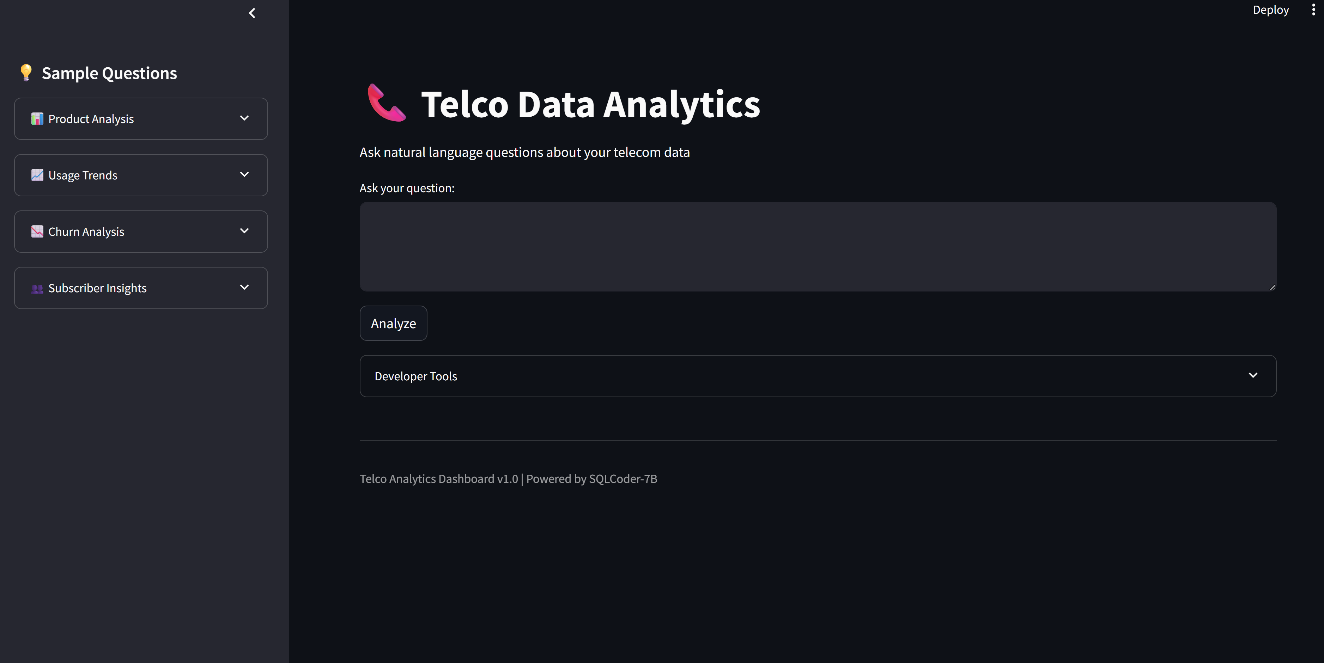
## Data Issues Handled

* **dim\_sales\_channel**: Duplicate records removed.
* **dim\_time.fiscal\_period**: Invalid data type corrected to appropriate format.

## Key Tables

1. fact\_usage
2. fact\_billing
3. fact\_churn
4. dim\_customer
5. dim\_product
6. dim\_network
7. dim\_sales\_channel
8. dim\_device
9. dim\_time

# Project Lifecycle



1. **User Input**: Natural Language Query
2. **SQL Generation**: Prompt passed to SQLCoder (LLM)
3. **Execution**: Query executed on Azure SQL
4. **Post-Processing**: DataFrame returned
5. **Charting**: Visualization generated
6. **Monitoring**: Query performance tracked

# Model Details

* **Model Used:** [defog/sqlcoder-7b](https://huggingface.co/TheBloke/sqlcoder-7B-GGUF)
* **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 Behavior Tracking

* Top Error Types
* Common Query Types
* Average Generation Time
* Success Rate

## SQL Processing Workflow

1. **Input Sanitization**

* Remove markdown code blocks (sql, )
* Strip leading/trailing whitespace
* Validate minimum length (>20 characters)

1. **Syntax Correction Phase** - Fix common SQL syntax issues:

* Fix malformed aliases
* Fix dangling brackets
* Replace stray characters in aliases
* Fix table name prefixes
* Ensure proper spacing
* Fix missing JOIN conditions

## Validation Rules Matrix

|  |  |  |  |
| --- | --- | --- | --- |
| Rule Category | Validation Check | Example | Error Message |
| Basic Structure | Starts with SELECT/WITH | SELECT [col] FROM... | "Query must begin with SELECT or WITH" |
| Table References | Uses [dbo].[table] format | [dbo].[dim\_product] | "Invalid table reference format" |
| JOIN Conditions | Explicit ON clause present | JOIN t ON t.id = t2.id | "Missing JOIN condition" |
| Date Handling | Uses T-SQL date functions | DATEFROMPARTS() | "PostgreSQL date\_trunc() not supported" |
| Security | No dangerous operations | - | "Blocked operation: DROP TABLE" |

## Domain-Specific Mapping Rules

|  |  |  |
| --- | --- | --- |
| Natural Language Term | SQL Element | Example Conversion |
| "top 5 plans" | TOP 5 + [dbo].[dim\_product] | SELECT TOP 5 [product\_name] FROM [dbo].[dim\_product] |
| "mobile models" | [model] column | SELECT [model] FROM [dbo].[dim\_device] |
| "last quarter" | [fiscal\_period] | WHERE [fiscal\_period] = DATEPART(quarter, DATEADD(quarter, -1, GETDATE())) |
| "customer since" | DATEDIFF calculation | DATEDIFF(YEAR, [subscription\_date], GETDATE()) |

## Prompt engineering:

* Schema context injection
* 22 specific T-SQL rules
* 5 few-shot examples
* Question classification

## Generation Parameters:

{ "max\_tokens": 350,

"temperature": 0.1,

"top\_p": 0.9,

"stop": ["###", "\n\n"]}

## Post-Processing:

* PostgreSQL-to-T-SQL conversion
* Syntax correction
* JOIN condition validation

## Supported Query Examples

* “What are the top 5 most expensive products?”
* “Show average data usage by device type.”
* “What are the top churn reasons last year?”
* “Show monthly data usage trends for the past year.”
* “Show subscriber count by gender.”

## 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 |

## Future Enhancements

* **Use Original Model**: Currently using quantized GGUF due to limited GPU (4GB). Will move to original defog/sqlcoder-7b requiring 6GB+ VRAM.
* **Schema-aware generation**: Fine-tuned model on telecom schema.
* **User authentication**: To save dashboards and query history.