Implementation Report: TravelIQ

1. Introduction

- TravelIQ is designed to analyze hotel booking data, extract key insights, and implement a **Retrieval-Augmented Generation (RAG)** system using **FAISS** and **Mistral AI** to answer booking-related queries.
- The system is built using **FastAPI** and provides a RESTful API to access analytics and Q&A functionalities.

2. Implementation Choices

2.1 Data Processing & Preprocessing

- **Dataset Used:** hotel_bookings.csv
- **Steps Taken:**
  - Cleaned missing values & handled categorical encoding
  - Created new features (total_revenue, total_nights)
  - Converted arrival_date into a structured format
  - Stored precomputed analytics in analytics_result.json for faster retrieval

2.2 Analytics & Reporting

- **Implemented key business insights:**
  - Revenue trends over time
  - Cancellation rate analysis
  - Geographical distribution of bookings
  - Lead time distribution before check-in
  - Market segment analysis & customer behavior insights
- **Visualization Tools:** Matplotlib, Seaborn, Pandas
- **Storage:** Precomputed insights are stored in analytics_result.json for fast retrieval

2.3 Retrieval-Augmented Generation (RAG) with FAISS

- Selected key categorical & numerical features for embedding
- Used sentence-transformers to generate embeddings
- Stored embeddings using FAISS for efficient vector search
- Implemented top-k retrieval to fetch relevant data before LLM processing

2.4 FastAPI Development

- Developed two main API endpoints:
  - POST /analytics → Returns precomputed analytics
  - POST /ask → Answers user queries using FAISS + LLM
- Used Hugging Face API for Mistral-7B-Instruct
- Exposed interactive API docs using Swagger UI (/docs)

3. Challenges & Solutions

Challenge	Solution
Large FAISS Index (~128MB) exceeds GitHub's file limit	Used Google Drive to store faiss_index.bin and provided a download link in README.md
Hugging Face API returning 404 errors	Verified API key, switched to HuggingFaceEndpoint, and tested multiple models
Long API response times for Q&A	Used precomputed analytics for /analytics to reduce processing time
Categorical data encoding causing issues in FAISS retrieval	Kept categorical columns as text instead of integer encoding

4. Future Improvements

- Real-time database updates for dynamic insights
- Query history tracking to optimize user interactions
- Fine-tuning LLM with domain-specific knowledge

5. Conclusion

- This project successfully integrates data analytics, FAISS vector search, and LLM-powered Q&A into a single API system.
- By combining precomputed analytics and real-time question answering, it delivers fast and accurate responses to booking-related queries.