Food Delivery Analytics Portal (Streamlit in Snowflake)

Project Title: Food Delivery Analytics Portal: Real-Time Strategic Insights

Developer: K.Dinkar & Tanmay Meda | **Platform:** Streamlit in Snowflake (SiS)

Mentor: Anand Jha Sir

I. Project Goal & Overview

A. Project Purpose: Actionable Insights

This portal gives food delivery businesses **immediate, actionable information** to manage the high-velocity, low-margin environment. The goal is to **optimize operations, drive profitability, and enhance customer loyalty** through analysis of platform economics, restaurant quality, and customer behavior.

Key Goals:

- **Boost Profit:** Maximize **Net Profit** by finding top restaurants and setting optimal commission rates.
- **Improve Loyalty:** Enhance **retention programs** and marketing through customer frequency and demographics.
- **Run Smarter:** Better **restaurant selection** and menu decisions using ratings, cuisine, and commission data.

B. Why Streamlit in Snowflake (SiS)? ❄️

SiS provides native security and performance by running the app **near the data** inside the Snowflake Data Cloud.

- **Zero Data Movement:** Queries run in Snowflake, reducing latency and data exposure (Core SiS Benefit).
- **Security:** Uses **Snowflake RBAC** (Role-Based Access Control) for secure, role-based access.
- **Simplified Deployment:** Snowflake manages hosting, eliminating external infrastructure and secrets management.
- **Developer Ergonomics:** Streamlit's minimal API makes it ideal for building analytics UIs.

🧠 **Why Use Streamlit?**

📊 Streamlit vs. Traditional BI Tools (Power BI)

Feature	Streamlit in Snowflake	Traditional BI (PBI)
Code Integration	Full Python ecosystem (ML, advanced statistics)	Limited to M/DAX languages
Data Movement	Zero—app runs near the data	Requires data export or live connection with latency risk
Deployment Model	Simplified deployment via SQL/Git integration	Requires external service (PBI Service) and gateway management
Cost Model	Usage-based (Snowflake Compute Credits)	Subscription-based (Per-user/capacity licenses)
Security	Inherits Snowflake's native RBAC	Requires separate security configuration (Azure AD/PBI Service)

Traditionally, if someone wanted to build a web app, they needed to learn:

- Frontend tools like HTML, CSS, JavaScript
- Web frameworks like Flask or Django
- Hosting and deployment

Streamlit removes all that complexity. You write a Python script, add a few special **st.** commands (like **st.write**, **st.button**, **st.slider**), and Streamlit handles everything else — including the UI.

C. Core Streamlit and Python Components

Feature	Code Component	Function in Project
Data Connection	`st.connection("snowflake")`	Automatically links to Snowflake using app owner's role (Best Practice).
Caching	`@st.cache_data`	Caches heavy query results, saving warehouse consumption and boosting performance.
UI Layout	`st.tabs`, `st.columns`	Structures the app, KPI tiles, and sidebar filters.
User Input	`st.multiselect`, `st.date_input`	Creates dynamic filters and AI query inputs.
Custom Styling	`st.markdown(css, ...)`	Implements the **custom Dark/Light Mode theme** and stylized KPI boxes.
Visualization	`plotly.express as px`	Generates advanced charts (Line, Bar) for trend and correlation analysis.
State Management	`st.session_state`	Manages the persistent state of the **Theme Toggle**.