Project Title	AI-Powered Blinkit Business Decision Platform
Skills take away From This Project	• Advanced SQL & Data ModelingMarketing Analytics • Full-Stack Data Science: Integrating SQL (Backend), Python/ML (Logic), and Streamlit (Frontend). • Predictive ModelingGenerative AI (RAG) • Business Intelligence
Domain	Quick Commerce (Q-Commerce) / Digital Marketing Analytics / Supply Chain Tech

Problem Statement:

In fast-paced Q-Commerce companies like Blinkit, data is generated in silos.

1. **The Marketing Team** spends millions on SMS/App campaigns but struggles to link that spend directly to daily revenue spikes.
2. **The Operations Team** deals with delivery delays but lacks predictive tools to foresee them.
3. **The Management Team** lacks a unified view, often relying on disparate Excel sheets.

The Goal: Build a **Unified Decision Platform** that ingests multi-source data to:

- Calculate real-time **Marketing ROI (ROAS)**.
- Predict **Delivery Delays** before they happen.
- Answer business questions instantly via an **AI Chatbot**.

Business Use Cases:

1. **Marketing ROI Optimization:** The system calculates the exact **ROAS (Return on Ad Spend)** per day. If ROAS drops below 2.0x, the dashboard alerts the manager to stop the campaign.
2. **Proactive Delay Management:** A predictive model flags orders with a high risk of delay (e.g., "Order #105 has an 85% chance of being late due to Peak Traffic in Indiranagar").
3. **AI Business Assistant:** Managers can ask, *"Why did sales drop yesterday?"* and the AI analyzes both marketing spend (was it too low?) and operations (were cancellations high?) to give an answer.
4. **Inventory Planning:** Correlating high-demand marketing campaigns with inventory levels to prevent stockouts.

Here, the given dataset can be used to solve many problems and you are always to challenge yourself to analyse it and solve as many as possible.

Approach:

Layer 1: Data Engineering (The Foundation)

"Turning Raw Chaos into Organized Intelligence"

- **The Problem:** In the real world, data never comes in a single, perfect file.
    1. **Marketing Data** looks like a summary: *"On Nov 1st, we spent ₹5,000 on SMS ads."* (1 Row per Day).
    2. **Sales Data** looks like a receipt: *"Order #101 bought Milk for ₹50 at 10:00 AM."* (Thousands of Rows per Day).
    3. **The Challenge:** You cannot calculate **ROAS (Return on Ad Spend)** because you can't join "Order #101" to "SMS Campaign" directly.
- **The Technical Solution (SQL):**
  Students will learn **Aggregation** and **Common Table Expressions (CTEs)**.
    1. **Step A (Squash):** They must write a query to "squash" the thousands of orders from Nov 1st into one single number: Total_Revenue = ₹50,000.
    2. **Step B (Join):** Now that both tables have 1 row per day (Nov 1st), they can be joined.
    3. **Step C (Calculate):** Create the ROAS metric: Revenue / Spend.
- **Student Task:** Write a PostgreSQL query that transforms 6 raw CSVs into one "Master Analytical View."
- **Interview Flex:** *"I didn't just load CSVs. I solved a granularity mismatch problem by aggregating transactional data to join it with time-series marketing data."*

Layer 2: The Analytics Dashboard (The "Rear-View Mirror")

"Visualizing What Happened So Managers Can React"

- **The Problem:** A SQL query gives you a table of numbers. A Business Manager cannot read a table of 10,000 rows to find a pattern. They need to see **trends**.
- **The Technical Solution (Streamlit):**
  Students will build an interactive UI using Python.
  - **Dual-Axis Charts:** On the X-Axis is Time (Days). The Left Y-Axis is **Revenue** (Green Line). The Right Y-Axis is **Ad Spend** (Red Bars).
  - **The Insight:** If the Red Bar goes UP, but the Green Line stays FLAT, the student has visually proven that the marketing campaign is failing.
- **Student Task:** Use streamlit and plotly libraries.
  - Create a Date Picker (e.g., "Last 7 Days").
  - Display KPI Cards (Big Bold Numbers): "Total Revenue: ₹1.5L", "Avg Delay: 12 mins".
- **Interview Flex:** *"I built a dashboard that monitors Marketing ROI in real-time, helping the team identify non-performing campaigns instantly."*

Layer 3: Predictive Machine Learning (The "Windshield")

"Predicting Problems Before They Happen"

- **The Problem:** Dashboards only show you what *already* went wrong. If a dashboard says "Avg Delay: 30 mins," the damage is already done. We want to know *before* the driver leaves.
- **The Technical Solution (Classification Model):**
  Students will train an AI model to guess the future.
  - **Feature Engineering:** The model cannot understand "2024-11-01 18:30:00". The student must convert this to:
    - Hour_of_Day = 18 (Evening Peak).
    - Day_of_Week = Friday (High Traffic).
    - Area = Indiranagar (Traffic Congestion Zone).
  - **The Training:** Feed this data into an algorithm. The model learns: *"Whenever it is Friday at 6 PM in Indiranagar, orders are usually late."*
- **Student Task:**

    - Train the model in Python (scikit-learn).
    - Save it as a .pkl file.
    - Build a **"Risk Calculator"** in the App: The manager types "Indiranagar, 6 PM", and the app says: **"⚠️ High Risk of Delay (85%)."**
- **Interview Flex:** *"I moved beyond simple analysis to predictive intelligence. My model helps the operations team allocate extra riders during predicted high-risk hours."*

Layer 4: Generative AI & RAG (The "Brain")

"Talking to Your Data"

- **The Problem:** Structured data (SQL) tells you *what* happened (Sales dropped). But it can't tell you *why*. The "Why" is hidden in the text comments: *"The mangoes were rotten," "The driver was rude," "Package was open."*
    - A human cannot read 5,000 feedback comments.
- **The Technical Solution (RAG - Retrieval Augmented Generation):** This is the cutting-edge part.
    - **Retrieval (Search):** The student converts all feedback text into math vectors (Embeddings). When a user asks *"Why are fruit sales down?"*, the system searches for comments related to "Fruit," "Rotten," "Quality."
    - **Generation (Answer):** It sends those specific comments to an LLM (like GPT-3.5 or Llama) with a prompt: *"Read these complaints and summarize the root cause."*
    - **The Output:** *"Customers are reporting that Alphonso Mangoes in the South Zone are arriving damaged due to poor packaging."*
- **Student Task:**
    - Use langchain or openai API or Groq API.
    - Create a Chat Interface in Streamlit.
- **Interview Flex:** *"I implemented a RAG pipeline that democratizes data access. Non-technical managers can now perform root-cause analysis on thousands of customer feedback rows just by asking questions."*

Summary: How they connect

1. **Layer 1** cleans the data so it's accurate.

2. **Layer 2** visualizes the data so we see the trend.
3. **Layer 3** uses the data history to predict the future.
4. **Layer 4** reads the text data to explain the "Why" behind the numbers.

Results:

By the end of the project, learners will have a deployed application with:

1. **Marketing Intelligence:** A chart proving which days yielded profitable ad spend vs. wasted budget.
2. **Operational Intelligence:** A "Risk Calculator" for delivery times.
3. **Customer Intelligence:** An automated summary of why customers are complaining (via GenAI).

Project Evaluation metrics:

1. **Data Logic:** Correct implementation of the Date-based Join for ROAS calculation (handling missing dates or zero spend).
2. **Model Accuracy:** Delivery Prediction model should achieve **AUC > 0.80**.
3. **Business Impact:** The Dashboard should clearly highlight "Profitable" vs. "Loss-Making" days
4. **Code Modularity:** Separation of SQL queries, ML training scripts, and UI code.

Technical Tags:

Python PostgreSQL Streamlit Marketing Analytics ROAS Machine Learning XGBoost Generative AI RAG ETL

Data Set:

Source: 6 Relational Tables (CSV/SQL).

These are the **three core data sources** (tables) that drive this entire project. Think of them as the three different departments of the company: **Marketing Team**, **Sales Team**, and **Logistics Team**.

1. Table A: Marketing Data ("The Investment")

Source: blinkit_marketing_performance.csv
This table tracks **money leaving the company** to buy ads. It tells you how much effort the marketing team is putting in to bring customers to the app.

- **Date:** The day the ad ran (e.g., Nov 1st).
- **Channel:** Where the ad was shown (e.g., SMS, Email, Facebook, App Notification).
- **Spend:** How much money was paid for the ad (e.g., ₹5,000).
- **Impressions:** How many people *saw* the ad (e.g., 10,000 people).

Why do we need it? To calculate **ROI**. If we spent ₹15,000 on Nov 1st, we need to check Table B to see if we made more than ₹15,000 back.

2. Table B: Orders Data ("The Revenue")

Source: blinkit_orders.csv
This table tracks **money coming into the company**. Every time a customer presses "Pay," a new row is created here.

- **Order_ID:** A unique number for the receipt (e.g., #1001).
- **Order_Date:** The exact timestamp of the purchase.
- **Order_Total:** The bill amount (Revenue).

Why do we need it? This is the **Truth**. Marketing says "We brought users," but this table proves if they actually bought anything. We sum up Order_Total by date to compare against Marketing Spend.

3. Table C: Operations Data ("The Experience")

Source: Derived from columns inside blinkit_orders.csv (or a logistics sheet).
This table tracks **efficiency**. It tells us if the company kept its promise to the customer.

- **Promised_Time:** The time the app showed the customer (e.g., "Arriving by 9:45 AM").
- **Actual_Time:** The time the driver actually rang the doorbell (e.g., 9:55 AM).
- **Region:** The area of delivery (e.g., Indiranagar, Koramangala).

Why do we need it? To build the **Predictive Model**. If we see that "Whitefield" is always "Late" at "10:00 AM," the AI learns this pattern and warns us next time.

The "Magic" Happens When You Join Them

In this project, you are the **Business Intelligence Hero** because you connect these three:

1. **Marketing (Table A) + Sales (Table B) = ROAS**
    - *"We spent ₹5,000 (A) and made ₹20,000 (B). Good job!"*
2. **Operations (Table C) + Sales (Table B) = Churn Risk**
    - *"Order #1002 was Late (C). That customer hasn't bought anything since (B). We lost them."*

Data Set Explanation:

- **Linking Logic:** The dataset does not have a direct "Campaign ID" in the Orders table. Learners must perform a **Time-Series Merge**: aggregating total revenue per day from Orders and mapping it to the total spend per day from Marketing.
- **Preprocessing:**
    - Convert Order_Date to Date format (removing timestamp).
    - Handle days with **Zero Spend** (infinite ROAS) or **Zero Sales** (0 ROAS).
    - Create a binary target variable Is_Late (1 or 0) for the ML model.

Project Deliverables:

1. **GitHub Repository:**
    - sql/roas_analysis.sql: The query used to join Marketing and Sales.
    - src/app.py: The Streamlit Dashboard.
    - src/model.pkl: The trained predictive model.
2. **Live Demo:** A screen recording showing the user navigating from "Marketing Trends" to "Delay Prediction" to "AI Chat."
3. **Project Report:** A PDF explaining the logic behind the Attribution Modeling and the ML Hyperparameters used.

Project Guidelines:

- **Best Practices:** Use **Common Table Expressions (CTEs)** in SQL for readability when calculating ROAS.
- **Visualization:** Use Dual-Axis charts (Bar for Spend, Line for Revenue) to show correlation clearly.
- **Version Control:** Regular commits documenting the progress from "Data Cleaning" to "Model Deployment."

Timeline:

Phase	Days	Focus Area	Key Deliverable
1. Foundation	1 – 3	Data Engineering (SQL)	A "Master Analytical Dataset" inside PostgreSQL created by joining Marketing (Daily) and Orders (Transactional) data.
2. Visualization	4 – 6	Analytics (Streamlit)	A live Dashboard comparing Marketing Spend vs. Revenue (ROAS) with interactive date filters.
3. Prediction	7 – 9	Machine Learning (ML)	A "Risk Calculator" tab that uses a trained model to predict Delivery Delays.
4. The Brain	10 – 12	GenAI (RAG & LLMs)	An AI Chatbot that answers questions like "Why are customers angry?" by searching customer feedback text.
5. Completion	13 – 15	Polish & Portfolio	A refined UI, a GitHub Repository with documentation, and a final Demo Video.

PROJECT DOUBT CLARIFICATION SESSION (PROJECT AND CLASS DOUBTS)

About Session: The Project Doubt Clarification Session is a helpful resource for resolving questions and concerns about projects and class topics. It provides support in understanding project requirements, addressing code issues, and clarifying class concepts. The session aims to enhance comprehension and provide guidance to overcome challenges effectively.
Note: Book the slot at least before 12:00 Pm on the same day

Timing: Monday-Saturday (4:00PM to 5:00PM)

Booking link :https://forms.gle/XC553oSbMJ2Gcfug9

For DE/BADM project/class topic doubt slot clarification session:

Booking link : https://forms.gle/NtkQ4UV9cBV7Ac3C8

Session timing:

For DE: 04:00 pm to 5:00 pm every saturday
For BADM 05:00 to 07:00 pm every saturday

LIVE EVALUATION SESSION (CAPSTONE AND FINAL PROJECT)

About Session: The Live Evaluation Session for Capstone and Final Projects allows participants to showcase their projects and receive real-time feedback for improvement. It assesses project quality and provides an opportunity for discussion and evaluation.
Note: This form will Open only on Saturday (after 2 PM) and Sunday on Every Week

Timing:

For BADM and DE
Monday-Saturday (11:30AM to 1:00PM)

For DS and AIML
Monday-Saturday (05:30PM to 07:00PM)

Booking link : https://forms.gle/1m2Gsro41fLtZurRA