HealthKart Influencer Campaign Analysis – Project Documentation

1. PROJECT OBJECTIVE

HealthKart runs influencer campaigns across platforms like Instagram, YouTube, and Twitter to promote products under brands such as MuscleBlaze, HKVitals, and Gritzo.

The goal of this project was to build an interactive and analytical dashboard that:

- Tracks influencer campaign performance
- Calculates both ROAS and Incremental ROAS
- Surfaces insights about top-performing influencers, personas, and cities
- Highlights underperforming influencers
- Supports exploratory filtering by brand and platform
- Monitors payouts made to influencers (per post or per order)

2. PROJECT STRUCTURE

The project was designed with modular and scalable components:

```
healthkart-influencer-dashboard/
│
├── data/                → Auto-generated datasets and insights
├── modeling/              → Schema definitions
├── simulation/            → Data simulation logic
├── analysis/              → Performance & financial metric calculators
├── app.py                 → Streamlit dashboard interface
├── requirements.txt       → Python dependencies
└── README.md              → Project documentation
```

3. COMPONENT-BY-COMPONENT SUMMARY

A. Data Modeling (`modeling/data_modeling.py`)
- Defines schema for all datasets
- Ensures consistency across modules
- Includes influencer, post, tracking, and payout structures

B. Data Simulation (`simulation/simulate_data.py`)
- Created synthetic influencer and campaign data using Faker
- Incorporated business logic like brand-product mapping, personas, age group, and cities
- Output stored in `/data` as CSVs for analysis and dashboarding

C. Performance Tracking (`analysis/performance_tracking.py`)
- Aggregated orders and revenue by influencer

- Used to understand overall impact per influencer
- Output: `performance_summary.csv`

D. ROAS Calculation (`analysis/roas_calculation.py`)
- Merged influencer revenue with payout data
- Computed:
    - ROAS = Revenue / Payout
    - Incremental ROAS = (Adjusted Revenue - Baseline) / Payout
- Output: `roas_summary.csv`

E. Insight Filtering (`analysis/filtering_insights.py`)
- Identified top-performing personas (category + gender)
- Grouped city-wise ROAS for geographic insights
- Flagged bottom 10% of influencers by ROAS
- Output: `city_roas.csv`, `poor_influencers.csv`

F. Interactive Dashboard (`app.py`)
- Built using Streamlit
- Features:
    - Filters: Brand, Platform
    - Summary metrics: Avg ROAS, Top ROAS
    - Visual charts: Persona & City ROAS
    - Tables: Top influencers, Bottom 10%
- Styled with color palette:
 - #6A89A7, #88BDF2, #BDDDFC, #384959

4. WHY THIS APPROACH

- *Synthetic Data*: Using `faker` allowed realistic but privacy-safe data for demo and testing
- *Modular Design*: Each stage (simulation → analysis → dashboard) is independent, easy to update or scale
- *Payout Integration*: ROAS reflects both revenue and influencer cost structure (per post or per order)
- *Persona & Geo Segments*: Enables actionable insights beyond raw numbers
- *Streamlit*: Chosen for rapid dashboarding and non-technical stakeholder usability

5. RESULTS & INSIGHTS (Based on Output Files)

Influencer Performance (`performance_summary.csv`)
- Revenue and orders per influencer varied widely
- Some influencers with lower follower counts outperformed large ones (likely due to niche audiences)

ROAS Analysis (`roas_summary.csv`)
- Top influencers achieved ROAS > 6.0
- Incremental ROAS indicated diminishing returns in some high-revenue campaigns

Persona-Level ROI
- Certain personas like `Fitness_Male` or `Nutrition_Female` had significantly higher average ROAS

City Insights (`city_roas.csv`)
- Cities like Bangalore and Delhi showed the best ROI
- Cities with high follower counts did not always yield high ROAS

Underperformance Flags (`poor_influencers.csv`)
- Influencers with high payouts but low order conversion were identified
- These should be deprioritized or monitored in future campaigns

6. NEXT STEPS / POTENTIAL EXTENSIONS

- Add real-time campaign data ingestion via APIs
- Integrate cost-per-click and impressions for multi-channel ROAS
- Build ML model for influencer performance prediction
- Export insights to PDF/CSV within dashboard
- Support A/B testing for incremental lift measurement

7. HOW TO RUN THE PROJECT

```
# 1. Install Dependencies
pip install -r requirements.txt

# 2. Generate Data
python simulation/simulate_data.py

# 3. Analyze Data
python analysis/performance_tracking.py
python analysis/roas_calculation.py
python analysis/filtering_insights.py

# 4. Launch Dashboard
streamlit run app.py
```

8. CONCLUSION

This system demonstrates a full data pipeline — from simulation and analysis to visualization — for evaluating influencer campaign ROI. The dashboard offers a scalable and practical way for marketing teams to make data-informed decisions.