Capstone Project Kickoff

Term Project (Units 1–3 Integrated, emphasis on Unit 3)

Weight:	30% of course grade

Milestones:	Proposal Tue, November 25;
	Milestone demo Week 15 Thu;
	Final deliverables Week 16 Thu

Datasets:	Team-chosen **Kaggle** dataset (primary **batch** source) **Plus** one **streaming** public API (secondary live source)
Public APIs	Free and Open Public APIs (No Auth Needed) https://mixedanalytics.com/blog/list-actually-free-open-no-auth-needed-apis/
Tools:	Colab, GitHub, GCS, BigQuery, **Cloud Functions 2nd gen**, **Pub/Sub**, **Dataflow template**, **BQML**, Looker Studio, Gemini prompts

Minimum Requirements

1. **Batch Ingest (from Kaggle)**
    - Store raw data in **GCS**, load curated tables into **BigQuery** (document schema & partitioning).
    - Provide **one** data quality check and **one** transformation logic explanation.
2. **Streaming Ingest (from API)**
    - **Cloud Function (2nd gen)** producer that calls a public API periodically (e.g., weather, crypto, transit, air quality, finance; no auth secrets in code).
    - Publish payloads to **Pub/Sub** as normalized JSON.
    - **Pipeline (API → Pub/Sub → BigQuery)** template writing to a streaming fact table.
    - Validate with queries showing near real-time rows (timestamps).

Minimum Requirements

3. **Analytics & Modeling**
   - **At least one** BQML model (regression *or* classification) that uses **both** batch and streaming features (e.g., join last N streaming metrics to batch entities, or score batch data with a streaming covariate).
   - **ML.EVALUATE** and **ML.EXPLAIN_PREDICT** examples; threshold discussion if classification.
4. **Executive Dashboard**
   - Looker Studio OR Plotly dashboard with **3–5 KPIs** and **at least one** time-series component fed by the streaming table.

Minimum Requirements

5. **Prompt Engineering & DIVE**
   - Include **prompt design** for analysis and **DIVE** journal entries showing how prompts evolved, where they failed, and how you validated.
6. **Architecture + Ops**
   - **Diagram** (GCS → BQ; Function → Pub/Sub → BQ; BQML; Looker/Plotly).
   - Reproducibility instructions to spin up/down the pipeline.

Milestones

- **Nov. 25 (Tue): Blueprint** (as in Lab 9) – business problem, sources, architecture, ML plan, KPIs, risks.

- **Dec. 11 (Thu): Live pipeline demo** (Cloud Function invoke + Pub/Sub → BigQuery row appears; show dashboard refresh).

- **Dec. 13 (Sun):** Final **presentation** + code freeze.

Deliverables

- **Individual (60 points total):**
  - **Notebook:** `Final_<Name>_analysis.ipynb`
    - Prompt logs, DIVE entries for one substantive question, at least **one** interactive Plotly figure, and a link to the dashboard section they influenced.
  - **Contribution MD (1 page):** `Final_<Name>_contrib.md` (exact tasks, PR links, lessons learned).
- **Team (40 points total):**
  - **Architecture README:** how to deploy/teardown; service enablement; IaC checklist (gcloud commands OK).
  - **Live Demo (8–10 slides + 5-min demo):** problem, architecture, pipeline proof, model results, dashboard, business impact.
  - **Governance Note (1–2 pages):** assumptions, data ethics, privacy/security notes, failure playbook.

Submission: D2L ZIP + repo URL + Looker link. Include **two screenshots**: BQ table screenshot with latest rows.

Suggested Repo Layout

```
TermProject_TeamX/
├─ notebooks/
│  ├─ Final_Alice_analysis.ipynb
│  ├─ Final_Bob_analysis.ipynb
│  └─ ...
├─ pipeline/
│  ├─ function/   (main.py, requirements.txt)
│  └─ infra/      (gcloud commands, enable APIs, SA roles)
├─ bq/
│  └─ sql/        (queries for eval/predict/join)
├─ dashboards/
│  └─ kpis.md     (definitions + Looker link)
├─ docs/
│  ├─ blueprint.pdf
│  ├─ governance.pdf
│  └─ ops_runbook.md
└─ README.md
```

Rubric (100 + up to 10 EC)

Individual (60)

- DIVE rigor (clear pivot from first to validated insight) – **20**
- Analytical depth + correct use of prompts & Plotly – **20**
- Clear, reproducible notebook & contribution mapping – **20**

Team (40)

- Pipeline completeness (batch + streaming) & validation – **15**
- BQML relevance + evaluation/explainability – **10**
- Dashboard clarity & KPI correctness – **10**
- Cost/ops considerations – **5**

Extra Credit (up to +10)

- Machine learning model quality

Unit 3 - Lab 2

OpenSky to BigQuery Table

Building Data Pipelines with BigQuery ML

↓

2. Google Cloud Storage

Data Flow

Upload data to GCS bucket for staging

↓

3. BigQuery Table

Load data into BigQuery table for analysis

↓

4. BQML Regression

ML Model

Predict flight velocity using linear regression

↓

4. BQML Regression ML Model

Predict flight velocity using linear regression

5. BQML Classification ML Model

Classify if flight is on ground

Critical Learning: NULL Handling

The classification model initially failed because the WHERE clause filtered out all 383 on_ground=TRUE records (which had NULL altitude/velocity values). Solution: Use `COALESCE(altitude, 0)` to handle NULLs and preserve label diversity.

Cell 1: Python Packages and Authentication

Purpose

This cell installs required Python packages and authenticates the user to Google Cloud Platform (GCP).

Packages Installed

- **google-cloud-storage:** For interacting with Google Cloud Storage
- **google-cloud-bigquery:** For interacting with BigQuery
- **requests:** Popular HTTP library for API calls

> **Note:** Authenticates user to enable access to GCP services

Cell 2: Project Configuration

Purpose

Configures essential project-specific variables for Google Cloud operations and sets the gcloud project.

Variables Defined

- **PROJECT_ID:** GCP project identifier
- **GCP_REGION:** Cloud region for services
- **GCS_BUCKET_NAME:** Cloud Storage bucket name
- **GCS_FOLDER_PATH:** Data storage folder path
- **BQ_DATASET:** BigQuery dataset name
- **BQ_TABLE:** BigQuery table for flight data
- **FLIGHT_RECORD_LIMIT:** API record limit

Cell 3: OpenSky API Class

Purpose

Defines the OpenSkyApi class and helper functions for data parsing and formatting.

Key Components

- OpenSky API client class implementation
- Data parsing functions for flight information
- Data formatting utilities for BigQuery compatibility

Cell 4: Initialize GCP Clients & Schema

Purpose

Initializes GCP clients, defines BigQuery schema, and implements the complete data pipeline logic.

Key Activities

- Initialize Google Cloud Storage and BigQuery clients
- Define BigQuery table schema for flight data
- Implement pipeline logic (API → GCS → BigQuery)

Cell 5: Execute Data Pipeline

Purpose

Executes the complete data pipeline from API to BigQuery.

Pipeline Flow

Fetch Data → Upload to GCS → Load into BigQuery

Cell 6: Full Pipeline Orchestration

Purpose

Orchestrates the complete end-to-end data pipeline from API ingestion through to BigQuery storage.

- Coordinates API data fetching
- Manages GCS upload operations
- Handles BigQuery table loading

Cell 7: BQML Regression Model

Purpose

Creates a BigQuery ML regression model to predict flight velocity.

Model Details

- **Type:** Linear Regression
- **Target:** Flight velocity
- **Features:** Altitude, latitude, longitude

Cell 8: Label Diversity Analysis

Purpose

Analyzes the on_ground label diversity in BigQuery to troubleshoot classification model issues.

Investigation Focus

- Examines distribution of on_ground values
- Identifies NULL handling issues
- Determines why initial model training failed

Cell 9: BQML Classification Model

Purpose

Creates a BigQuery ML logistic regression model to classify whether a flight is on the ground.

Model Specifications

- **Type:** Logistic Regression
- **Label:** on_ground (boolean)
- **Features:** altitude, velocity
- **Fix:** Uses COALESCE for NULL handling

Lab Summary: Key Learnings

Primary Issue & Resolution

Classification model failed due to WHERE clause filtering out TRUE labels. Fixed with COALESCE NULL handling.

Key Findings

- 383 on_ground=TRUE records had NULL values
- Successfully created regression and classification models

Best Practices

- Verify data after applying filters
- Use COALESCE for NULL handling