End-to-End Project (DS + ML + DevOps)

Customer Churn & Revenue Optimization Intelligence System

Data-driven decision system powered by ML and operated using DevOps

1️⃣ What Historical Data You Use (Concrete)

You will work with **structured tabular data**, like real companies do.

Typical columns:

- customer_id
- signup_date
- last_active_date
- monthly_charges
- total_spend
- usage_frequency
- complaints_count
- payment_delays
- contract_type
- churn (0/1) ← historical outcome

2️⃣ What "Useful Output" You Extract for the Company

This is the **core of your project**.

From historical data, your system will output:

A. Business Insights (Non-ML)

Before ML, you extract:

- Which customer segments churn the most
- Revenue loss due to churn
- High-risk customer profiles
- Time-based patterns (tenure vs churn)

This shows **data analyst maturity**.

B. Predictive Insight (ML starts here)

You answer:

> “Which current customers are likely to churn next month?”

This is **predictive analytics**, not theory.

Output:

- Churn probability per customer (0–100%)
- Risk buckets:
  - Low
  - Medium
  - High

Now the company can **act before losing money**.

C. Prescriptive Insight (This is what impresses)

You go one step further:

> “If we can’t save everyone, who should we focus on first?”

You combine:

- Churn probability
- Customer lifetime value (CLV)

Output:

- Priority score = Risk × Revenue impact

This is **decision intelligence**, not just ML.

3️⃣ Now EXACTLY What ML You Are Building

No ambiguity.

🎯 ML Problem Type

Binary Classification

Target:

```
Willthis customer churnin the next period? (Yes / No)
```

📌 ML Algorithms (Simple but correct)

You are NOT doing deep learning.

You will use:

- Logistic Regression (baseline)
- Random Forest (non-linear)
- XGBoost (optional, if time permits)

Why?

- Interpretable
- Industry-standard
- Recruiter-safe

📊 ML Outputs (Important)

Your model produces:

- Probability score (not just yes/no)
- Feature importance (why churn happens)
- Model metrics:
  - ROC-AUC
  - Precision / Recall

This proves **you understand evaluation**, not accuracy worship.

4️⃣ What You Are ACTUALLY "Making" in This Project

Let me be painfully specific.

You are building:

1️⃣ A Data Pipeline

- Raw CSV → cleaned tables
- SQL warehouse
- Feature tables

2️⃣ A Prediction Engine

- Trained ML model
- Saved artifacts
- Versioned models

3️⃣ A Decision API

- FastAPI endpoint:

```
{
"customer_id":123,
"churn_probability":0.82,
"risk_level":"HIGH",
"expected_revenue_loss":5400
}
```

This is **industry-grade output**.

4️⃣ A Business Dashboard

Power BI / Tableau shows:

- Current churn risk
- High-value at-risk customers

- Trend analysis
- Model prediction vs actuals

This makes your ML **usable**.

5️⃣ A Fully Automated DevOps System

This is what ties everything together.

Flow:

```
Newdata →
Model retrains →
Metrics checked →
If better →
Docker build →
Kubernetes deploy
```

No human babysitting.

7️⃣ Why This Is Market-Level (Be Honest with Yourself)

This project:

- Solves a real problem
- Uses historical data
- Produces actionable outputs
- Runs automatically
- Scales
- Can be explained to business + tech teams

This is **exactly what entry-level data/ML engineers are expected to understand**, even if they don't build all of it alone in a job.

STEP-BY-STEP PIPELINE PLAN (END-TO-END)

This is the **actual lifecycle** your project will follow.

🔷 STEP 1: Raw Data Ingestion

- Load CSV
- Store as raw table in SQL
- No transformations here

Why: Data traceability (production rule)

🔷 STEP 2: Data Cleaning

Actions:

- Convert `TotalCharges` to numeric
- Handle missing values
- Normalize categorical values
- Drop impossible rows

Output:

`clean_customers` table

🔷 STEP 3: Feature Engineering

This is where you **prove competence**.

You will:

- Derive signup & activity dates
- Compute usage_frequency
- Simulate operational features
- Encode categoricals
- Scale numeric features

Output:

`customer_features` table

🔷 STEP 4: EDA (Exploratory Analysis)

Questions answered:

- Who churns the most?
- Which features drive churn?
- Revenue impact of churn?

Artifacts:

- Plots
- Summary tables
- Business insights (for README)

🔷 STEP 5: Model Training

- Train baseline model
- Train advanced model
- Compare metrics
- Select best model

Saved artifacts:

- model.pkl
- scaler.pkl
- feature_list.json

🔷 STEP 6: API Layer

Expose predictions using FastAPI:

- `/predict`
- `/health`

Model runs **only through API**, not notebooks.

🔷 STEP 7: Automation & Deployment

- Dockerize API
- CI/CD pipeline
- Kubernetes deployment
- Auto-retraining trigger (future-ready)

3️⃣ STARTER CODE TEMPLATES (CLEAN & REAL)

📁 Project Structure (Non-Negotiable)

```
ml-churn-platform/
│
├── data/
│   ├── raw/
│   ├── processed/
│
├── src/
│   ├── ingestion.py
│   ├── cleaning.py
│   ├── features.py
│   ├── train.py
│   ├── evaluate.py
│
├── api/
│   ├── app.py
│
├── models/
│   ├── churn_model.pkl
│   ├── scaler.pkl
│
```

```
├── notebooks/
│   ├── eda.ipynb
│
├── requirements.txt
├── README.md
```