mob

# Data Engineer Takehome Assignment

## Stripe Revenue Recognition Pipeline to BigQuery

**Time estimate:** 3-4 hours
**Submission:** Document with architecture design, code samples, and discussion points

We are excited to see your approach! This problem doesn't have a canonical ‘right’ answer. Instead, we are interested in how you think about data engineering challenges, ask clarifying questions, and make pragmatic trade-offs.

---

## Context

Our finance team currently uses Stripe Sigma templates to generate monthly revenue recognition reports for our e-commerce business. These templates handle complex scenarios like:

- Multi-currency subscriptions
- Annual subscriptions requiring monthly revenue recognition (deferred revenue)
- Mid-period subscription changes and prorations
- Tax calculations (inclusive and exclusive)
- Invoice line items with varying service periods

The finance team wants to move away from manual Sigma exports and build a **daily pipeline** that syncs this data to BigQuery for:

1. Integration with our internal analytics platform of choice
2. Automated revenue recognition workflows
3. Historical trend analysis and forecasting

---

## The Challenge

Design and prototype a data pipeline that replicates the **revenue recognition logic** from Stripe's Sigma templates into BigQuery tables.

### Specific Requirements

**Primary Goal:** Build a system that can accurately calculate and track deferred revenue for subscription invoices on a daily basis.

**Data Sources:**

- Stripe API endpoints (invoices, invoice line items, subscriptions, customers, etc.)
- Focus on invoices with `status='paid'` and line items with subscription periods > 31 days

**Target Output:** A BigQuery table (or set of tables) with the schema you design that should support queries like:

- "What is our recognised revenue for Q2 2025?"
- "How much deferred revenue do we have remaining as of today?"
- "Which customers have revenue recognised this month?"

### Known Complications

These reflect the real-life messiness you will have to deal with in your daily job:

1. **Metadata Inheritance:** Invoice line items with `parent.type="subscription_item_details"` reflect the most recent subscription metadata at retrieval time, not at invoice creation time. How do you handle temporal accuracy?

2. **Tax Ambiguity:** Some invoices have `automatic_tax` enabled, others have manual tax rates, and historical invoices may have `null` tax fields. Tax can be inclusive or exclusive depending on customer settings.

3. **Multi-Currency:** Our business operates in USD, GBP, and EUR. Exchange rates fluctuate. Do you convert everything to a base currency? When? Why?

4. **Prorations:** Mid-period subscription changes create proration line items. These have different `period.start` and `period.end` dates than the parent subscription. How do you attribute these to the correct revenue recognition periods?

5. **API Rate Limits:** Stripe has rate limits. You will need to sync thousands of invoices. How do you design for this?

6. **Data Freshness:** The Stripe API doesn't have an `updated_at` field on all objects. How do you efficiently identify what changed since your last sync?

---

## What We Are Looking For

Please provide a written document (PDF, markdown, or Google Doc) that includes:

### 1. Architecture Design (40%)

- High-level pipeline architecture diagram

- Choice of orchestration approach (batch/streaming/incremental)
- Data flow from Stripe API → BigQuery
- How you handle idempotency and data quality
- Monitoring and alerting strategy

### 2. Schema Design (25%)

- Proposed BigQuery table schema(s)
- Rationale for your table structure (normalised vs. denormalised)
- How you handle slowly changing dimensions (e.g. subscription metadata changes)
- Indexing/clustering/partitioning strategy

### 3. Code Sample (25%)

- A code snippet (Python or R preferred, but any language is fine) demonstrating:
    - How you would fetch and transform invoice data from the Stripe API
    - Logic for calculating recognised revenue for a given period
    - Handling of at least three 'complications' from the list above
- Focus on clarity and correctness over completeness

### 4. Discussion Points (10%)

These are the questions we would explore during the follow-up discussion. For each, provide your initial thoughts (2-3 sentences are fine to get things going):

a) **Tax Treatment:** A customer in Germany has an annual €43.74 subscription with 19% VAT (inclusive). The invoice shows `total=43.74`, `tax=6.98`, and we need to recognise €3.06 in revenue per month (excluding VAT). Walk through your calculation and any edge cases.

b) **Missing Data:** You discover that 3% of paid invoices are missing `line_items.period.end` dates (they're `null`). How do you handle this in production?

c) **Historical Backfill:** We need to backfill 2 years of historical data (say, ~100,000 invoices). What's your strategy for the initial load vs. ongoing daily syncs?

d) **Future-proofing the Infrastructure:** Discuss how your design would scale if data volume increases by orders of magnitude (10x, 100x).

e) **Business Logic Clarification:** The finance team says 'we recognise revenue daily' but the Stripe subscription periods are monthly/yearly. Do you recognise 1/30th of the monthly revenue each day, or the full month's revenue on the 1st? What questions would you ask?

---

## Submission Guidelines

- **Format:** PDF, markdown, or publicly accessible Google Doc link
- **Length:** No strict limit, but be concise and avoid fluff/filler. Aim for 4-8 pages.

- **Code:** Pseudocode is fine, but executable snippets are preferred
- **Diagrams:** Hand-drawn sketches are acceptable; tool-created diagrams are optional
- **References:** Feel free to cite Stripe API documentation, but explain your reasoning
- **AI:** We encourage responsible use of generative AI with this task - but see below!

**What NOT to do:**

- Don't build a full working pipeline - we are not asking for that!
- Don’t just feed this into your LLM of choice; if you don’t deeply understand what you are submitting, you won’t be able to discuss its’ nuances
- Don't spend more than 4 hours on the task
- Don't stress about ‘perfect’ solutions, we want to see your thought process

---

# Resources

- Stripe API Documentation
- Stripe Invoices API
- Stripe Invoice Line Items
- Stripe Subscription API
- Stripe Tax Documentation

---

# Follow-Up Discussion Topics

After reviewing your submission, we will have a 60-minute technical discussion where we will:

- Deep-dive into your architectural choices
- Explore trade-offs you considered
- Talk about edge cases and failure modes
- Collaborate on refining the revenue recognition logic