# Student Instructions: Lab 2 - The Refinery (ETL)

Module: 2 | Time Estimate: 3 Hours

Role: Data Engineer

Tools: Google Cloud Platform (GCS & BigQuery)

### The Scenario

The IT Director at **RetroRetail Inc.** is impressed with the speed of BigQuery (from Lab 1), but now he has a bigger problem.

He hands you a file (Dirty\_Customers.csv) containing customer records from their legacy system. It is a disaster.

* **Duplicates:** Customers appear multiple times.
* **Inconsistent States:** Some say "OH", others "Ohio", "OHIO", or "Oh."
* **Dirty Emails:** Some emails are missing; others are invalid text.
* **Formatting:** Names have random capitalization and leading spaces.

**Your Mission:**

1. **Ingest:** Load the raw legacy file into BigQuery.
2. **Discover:** Query the data to identify the specific quality issues.
3. **Refine:** Write a professional SQL pipeline (using RegEx and Window Functions) to clean the data into a "Trusted" table.

### Part 1: The Ingest (Load "Raw")

1. **Download:** Download the Dirty\_Customers.csv file provided by the instructor on Canvas.
2. **Upload to Cloud Storage:**
   * Go to your **Google Cloud Storage** bucket (created in Lab 1).
   * Upload Dirty\_Customers.csv.
3. **Load into BigQuery:**
   * Navigate to **BigQuery**.
   * Select your retro\_raw dataset.
   * **Create Table:**
     + **Source:** Select Google Cloud Storage 🡪 Browse for Dirty\_Customers.csv.
     + **Table Name:** customers\_raw.
     + **Schema:** Check "Auto-detect".
   * Click **Create Table**.

### Part 2: The Discovery (Find the Rot)

*Before fixing the data, you must understand the scope of the mess.*

1. Run the following query to identify the inconsistent state codes:  
   SQL  
   SELECT state, COUNT(\*) as count  
   FROM `retro\_raw.customers\_raw`  
   GROUP BY state  
   ORDER BY count DESC;
2. **Take Screenshot #1:** Capture the result showing the variations (e.g., Ohio, OHIO, OH appearing as separate rows).

### Part 3: The Refinery (SQL Transformation)

*This is the core engineering task. You must fix all 4 problems in a single query.*

1. **Develop Your Query:** Write a SQL query that performs the following transformations:
   * **Standardize Names:** Use INITCAP and TRIM to fix capitalization and spaces.
   * **Fix States:** Use a CASE statement to map 'Ohio', 'OHIO', 'Oh.' --> 'OH'. (Do the same for KY and IN).
   * **Clean Emails:** Use LOWER() to standardize case. Use COALESCE to replace empty/null emails with 'unknown@noemail.com'.
   * **Deduplicate:** Use the QUALIFY ROW\_NUMBER() window function to keep only the *most recent* row for each customer\_id.
2. **Save the View:**
   * Once your query works, click **Save** --> **Save View**.
   * **Dataset:** retro\_raw (or create a new dataset called retro\_clean).
   * **Table Name:** view\_customers\_clean.

**Hint (Skeleton Code):**

SQL

SELECT  
 customer\_id,  
 -- Fix Name  
 INITCAP(TRIM(full\_name)) as clean\_name,  
 -- Fix State  
 CASE  
 WHEN UPPER(state) LIKE 'OH%' THEN 'OH'  
 WHEN UPPER(state) LIKE 'KY%' THEN 'KY'  
 WHEN UPPER(state) LIKE 'IN%' THEN 'IN'  
 ELSE 'UNKNOWN'  
 END as clean\_state,  
 -- Fix Email (Add logic here)  
 ...  
FROM `retro\_raw.customers\_raw`  
WHERE ... -- Filter out invalid emails if needed  
QUALIFY ROW\_NUMBER() OVER(PARTITION BY customer\_id ORDER BY customer\_id DESC) = 1

1. **Take Screenshot #2:** Run SELECT \* FROM retro\_raw.view\_customers\_clean LIMIT 20 and capture the clean results.

### The Deliverable: Client Report #2

Submit a single PDF named LastName\_Lab2.pdf containing:

1. **Before & After (Screenshots):**
   * **Image 1:** The "State" count query showing the messy data.
   * **Image 2:** Your final SQL Code used for the cleaning.
   * **Image 3:** The final result showing clean names, states, and unique IDs.
2. **Executive Summary:**
   * **Deduplication Strategy:** Explain why you used ROW\_NUMBER instead of just DISTINCT.
   * **Business Impact:** Why does normalizing "Ohio/OHIO" to "OH" matter for the business intelligence team?
3. **AI Disclosure:** Paste any prompts used to assist with the SQL syntax.