**4.1**

**Customer Purchase Patterns (Asuminng the a Customer Table Exists)**

- This insight can be useful for understanding customer behavior (e.g., repeat purchases, average order value, and purchase frequency) helping with targeted marketing campaigns, loyalty programs, and personalized recommendations.

- How to get it, we are going to required the table `customers` linked to the `orders` table by a `customer\_id` column and the `orders` and `products` tables to calculate average revenue by summing the product of `quantity` and `price` grouped by `created\_date` or other time intervals.

- Query:

“SELECT c.id as Client\_id, AVG(o.quantity p.price) AS avg\_revenue, DATE\_TRUNC('month', o.created\_date) AS month

FROM orders o

JOIN customers c ON o.product\_id = c.id

JOIN products p ON o.product\_id = p.id

GROUP BY month

ORDER BY month; “

**Revenue Trends Over Time**

- This insight can be useful for analyzing revenue trends over weeks, months, or quarters and can highlight seasonality or growth opportunities. This helps optimize inventory and marketing efforts.

- How to get it, we are going to use the `orders` and `products` tables to calculate total revenue by summing the product of `quantity` and `price` grouped by `created\_date` or other time intervals.

- This query helps with the idea of how to get it:

“SELECT DATE\_TRUNC('month', o.created\_date) AS month, SUM(o.quantity p.price) AS total\_revenue

FROM orders o

JOIN products p ON o.product\_id = p.id

GROUP BY month

ORDER BY month; “

**Product Performance by Category**

- This insight can be useful for understanding which categories generate the most sales or revenue and can guide product development, stocking decisions, and marketing focus.

- How to get it, we will use the `products` and `orders` tables to group sales data by `category` and rank them by total quantity or revenue.

- This query helps with the idea of how to get it:

“SELECT p.category, SUM(o.quantity) AS total\_quantity, SUM(o.quantity p.price) AS total\_revenue

FROM orders o

JOIN products p ON o.product\_id = p.id

GROUP BY p.category

ORDER BY total\_revenue DESC; “

**4.2**

Personally, I would choose Apache Airflow because it is a tool I have experience with from past projects. It allows for the programmatic design, scheduling, and monitoring of data pipelines.

Steps to create the pipeline with Apache airflow:

**1. Set Up the Environment:**

-Install Airflow: using pip: pip install apache-airflow

-Configure Connections:

* PostgreSQL Connection: Add a connection in Airflow pointing to your PostgreSQL database (source).
* BigQuery Connection: Use a service account JSON key to authenticate Airflow with BigQuery. Add it as a connection in Airflow (Google Cloud type).

**2. Define the Pipeline (DAG)**

- A Directed Acyclic Graph (DAG) in Airflow will represent your pipeline.

- Key Tasks:

1. Extract Data from PostgreSQL: Use Airflow's `PostgresOperator` or `PythonOperator` to query data from the `products` and `orders` tables.

2. Transform Data: Perform any required transformations (e.g., calculate total revenue, aggregate categories) using Python or SQL.

3. Load Data into BigQuery: Use Airflow’s `BigQueryInsertJobOperator` to insert transformed data into a BigQuery table.

**3. Write the DAG Code**

-The DAG code can be represented as a Python code file using the library

**4. Deploy the DAG**

- Place the DAG file in Airflow's `dags/` directory.

- Start the Airflow scheduler and webserver to execute the pipeline.