**Case Study: Eniac’s Discount Strategy**

**Start Date: 13.10.2025**

**Presentation Day : 27.10.2025**

We working as a **Data Analyst for Eniac**, an e-commerce company. This time, you’ll use **Python** to analyze the company’s **internal data**, which is a bit messy.

The company has a debate going on: **should they offer discounts or not?**

* **Marketing Team’s View:**  
  They think **discounts are good** because:
  + They attract **more customers**.
  + Customers are **happier** and may come back (**retention**).
  + It helps the company **grow in the long run**.
* **Investors’ View:**  
  They are **cautious about discounts** because:
  + Even though more orders were placed recently, the **total revenue went down**.
  + They prefer the company to focus on **quality**, not trying to be the cheapest.

Our Job is **analyze the data** and see if offering discounts really helps the company **grow profits, retain customers, and improve satisfaction**, or if it’s just **reducing revenue**.

**1.Business Questions:**

**1. Classifying products into categories**

* Purpose: Makes analysis easier and helps identify trends.
* Example categories could be based on:
  + **Type**: Electronics, Clothing, Home, Beauty
  + **Price range**: Low (<€50), Medium (€50–€200), High (>€200)
  + **Discounted vs. Non-discounted**
* Python approach: Create a new column in your dataset like Category using conditions.

**2. Distribution of product prices**

* Purpose: Understand the pricing landscape.
* How to do it:
  + Calculate **min, max, average, median** prices per category.
  + Plot **histograms or boxplots** to see price distribution visually.
* Insight: You’ll see which categories have high-priced vs. low-priced items.

**3. How many products are discounted**

* Purpose: Know how common discounts are.
* How to do it:
  + Count the number of products where Discount > 0.
  + Group by category to see which types of products get discounts more often.

**4. Size of the discounts**

* Purpose: Understand the magnitude of discounts.
* How to do it:
  + Calculate Discount Percentage = (Discount / Original Price) \* 100.
  + Analyze average discount per category.
* Insight: Some categories might have big discounts, others very small.

**5. Seasonality and special dates**

* Purpose: Identify trends and plan marketing.
* How to do it:
  + Group sales by **month, week, or day**.
  + Highlight special events: Black Friday, Christmas, New Year.
  + Use line charts to show spikes in sales during these periods.
* Insight: Helps decide **when to offer discounts** and which products to promote.

**6. Improving data collection**

* Purpose: Make future analysis easier and more accurate.
* Suggestions:
  + Include **product category and subcategory** in the database.
  + Track **customer demographics** and **purchase behavior**.
  + Record **discount campaigns** with start and end dates.
  + Track **special events and seasonal trends** explicitly.

**7. Presenting your findings**

* Keep it **concise but engaging**:
  + Use charts (histograms, boxplots, line charts).
  + Highlight **key insights**: which discounts work, which products sell more, seasonal trends.
  + Conclude with **recommendations**: e.g., “Offer moderate discounts on high-demand products during Christmas.”

**2.The Problem**

The data you received is **messy and inconsistent** — basically, it’s “corrupted.”

* The **Database Admin (Khader)** thinks it’s because the **pipeline** connecting the online store to the database is buggy.
* The **Software Engineer (Lina)** thinks it’s because of **wrong encodings** or **poor database maintenance**.

The bottom line: nobody exactly knows why, but the data **cannot be trusted as-is**.

**Why This Matters**

Before you can answer **business questions** like discounts, sales trends, or product categories, you need to **clean and check the data**.

* If you analyze bad data, your results will be **wrong or misleading**.
* Even the smartest analysis tools or fancy algorithms **cannot fix corrupted data automatically**.

**The Reality of Data Work**

* In the data world, people often say that **80% of your time goes into cleaning and preparing data**.
* This number isn’t exact, but it **captures the truth**: most of a Data Analyst’s work is making messy data usable and trustworthy.

**What We need to do here…**

* **Check for missing values** (empty cells).
* **Fix inconsistencies** (e.g., spelling errors, wrong encodings).
* **Correct data types** (numbers stored as text, dates in wrong format).
* **Document any assumptions or fixes** so your analysis is transparent.

**Get the data**

**🧾 1️⃣ orders.csv — *Customer Orders Overview***

**What it contains:**  
Each row is **one order placed by a customer** — it tells you when it was made, how much was paid, and what stage the order is in.

**Columns:**

| **Column** | **Meaning** | **Notes** |
| --- | --- | --- |
| order\_id | Unique ID for each order | Use this to join (connect) with orderlines.csv |
| created\_date | Timestamp when the order was created | Format: YYYY-MM-DD HH:MM:SS |
| total\_paid | Total amount (in euros) that the customer paid | Helps you calculate total sales revenue |
| state | Status of the order | Important for filtering active/completed/cancelled orders |

**Possible state values:**

* 🛒 **Shopping basket** – Items added to cart but not yet ordered
* 📦 **Place Order** – Order placed, awaiting shipping
* 💳 **Pending** – Payment not yet confirmed
* ✅ **Completed** – Order placed and paid (finalized sale)
* ❌ **Cancelled** – Order cancelled and refunded

📌 **Import Notes:**

* You’ll likely analyze **only “Completed” orders** for real sales.
* Use order\_id as the **primary key** when joining with other tables.

**📦 2️⃣ orderlines.csv — *Order Details (Products in Each Order)***

**What it contains:**  
Each row represents **one product item within an order** (so one order can have multiple rows here).

**Columns:**

| **Column** | **Meaning** | **Notes** |
| --- | --- | --- |
| id | Unique ID for each record in this table | Not used for joins |
| id\_order | Matches with orders.order\_id | Use this to connect to orders |
| product\_id | Old internal product ID (not used now) | Can ignore |
| product\_quantity | Quantity of the product purchased | Needed for totals |
| sku | Stock Keeping Unit (unique code for product) | Connects to products.csv |
| unit\_price | Price (in euros) per unit at the time of order | May differ from base price due to promotions |
| date | Timestamp for product processing | Helps with order timing analysis |

📌 **Import Notes:**

* Always join orderlines.csv with:
  + orders.csv (via id\_order)
  + products.csv (via sku)
* This table lets you **see exactly which products were sold**, in what quantity, and at what price.

**🛍️ 3️⃣ products.csv — *Product Information***

**What it contains:**  
Details about each product sold in the store.

**Columns:**

| **Column** | **Meaning** | **Notes** |
| --- | --- | --- |
| sku | Unique product code | Connects to both orderlines.csv and brands.csv |
| name | Product name | For display and analysis |
| desc | Product description | Often text-heavy, not always needed for analytics |
| price | Regular (base) price | Compare this with promo\_price |
| promo\_price | Discounted price | If null, product not on sale |
| in\_stock | Indicates if product was in stock (True/False) | Useful for stock analysis |
| type | Numerical product type/category | Use to classify products |

📌 **Import Notes:**

* The first **3 letters of sku** correspond to the **brand code** from brands.csv.
* You can analyze **price vs. promo\_price** to see discounts.

**🏷️ 4️⃣ brands.csv — *Brand Reference Table***

**What it contains:**  
Brand name and its corresponding short code.

**Columns:**

| **Column** | **Meaning** | **Notes** |
| --- | --- | --- |
| short | 3-letter brand code | Found as first 3 letters in products.sku |
| long | Full brand name | Use this for human-readable outputs |

📌 **Import Notes:**

* Join brands.csv with products.csv using this rule:  
  LEFT(products.sku, 3) = brands.short
* This lets you know **which brand** each product belongs to.

**🔗 Relationships Summary (Very Important)**

Here’s how the files are related:

orders.csv (order\_id)

|

| id\_order

v

orderlines.csv (sku)

|

| sku

v

products.csv (sku)

|

| first 3 letters of sku = short

v

brands.csv

📊 **Example Workflow for Analysis:**

1. Start from orders.csv → filter only Completed orders.
2. Join with orderlines.csv on order\_id = id\_order.
3. Join with products.csv on sku.
4. Optionally join brands.csv using the first 3 letters of sku.
5. Now you can analyze total sales by product, brand, type, date, etc.