Università di Pisa

Corso di Laurea Magistrale in

Data Science and
Business Informatics

Decision Support Systems: Laboratory of Data Science

Relazione finale di progetto - Gruppo 480

Scritto da
Daniele Borghesi (578406)
Francesco Pio Capoccello (659791)

Contents

1 Introduzione

Il presente report descrive il lavoro svolto per lo sviluppo di un sistema di data warehousing e business intelligence, basato sui dati contenuti nel file `computer_sales.csv`. Il file in questione contiene dettagli sulle vendite di computer avvenute tra marzo 2013 e aprile 2018. Ogni voce del dataset fornisce anche particolari tecnici di ciascun computer, come caratteristiche specifiche di CPU, GPU e RAM. Inoltre, il file aggiuntivo `geography.csv` arricchisce le informazioni con dettagli riguardanti le località geografiche associate a ciascuna vendita.

Il progetto si articola in diverse fasi, che includono la costruzione di un data warehouse (sezione 2), la realizzazione di un processo ETL (Extract, Transform, Load) (sezione 3), la creazione di un cubo di dati (sezione 4), l'esecuzione di query MDX per l'analisi multidimensionale (sezione 5), e infine la realizzazione di una dashboard per la visualizzazione interattiva dei risultati (sezione 6). Ogni sezione del report illustra le fasi principali e i passaggi fondamentali che hanno portato al completamento del progetto, descrivendo le tecniche utilizzate, le decisioni prese e i risultati ottenuti.

2 Costruzione del data warehouse

La costruzione del data warehouse si è composta di due fasi distinte: la prima fase (sezione 2.1) ha consistito nella creazione dei file `csv` necessari alla popolazione delle tabelle del data warehouse, mentre nella seconda fase (sezione 2.2) abbiamo proceduto all'effettiva creazione e popolazione del data warehouse.

2.1 Creazione delle Tabelle

In questa prima fase del progetto, l'obiettivo principale è stato quello di elaborare i dati contenuti nei file `computer_sales.csv` e `geography.csv` per la creazione di nuovi file `csv` coerenti con il modello di datawarehouse proposto in figura 1.

Ogni file `csv` relativo alle tabelle è stato generato utilizzando script Python specifici per garantire la corretta integrazione e formattazione dei dati. Di seguito sono descritte delle operazioni eseguite per ciascuno script.

geography.py Per la creazione di `Geography.csv`, è stato implementato uno script Python che ha gestito l'estrazione e la strutturazione dei dati a partire dal file `geography.csv`.

Inizialmente sono stati caricati i dati dal file `geography.csv` in un dizionario Python, per facilitare l'accesso e la manipolazione. Ogni record è stato elaborato per estrarre le informazioni chiave, tra cui l'ID geografico (*geo_id*), il continente (*continent*), il paese (*country*), la regione (*region*) e la valuta di quel paese (*currency*). Dato che nei dati contenuti nel file `geography.csv` non era presente la valuta utilizzata in ogni paese, abbiamo creato manualmente la colonna *currency* sulla base delle seguenti associazioni. Anche se la valuta è già presente all'interno del file `computer_sales.csv`, creare un piccolo dizionario è risultato essere più veloce e immediato, rispetto all'estrazione della valuta direttamente dal file originale.

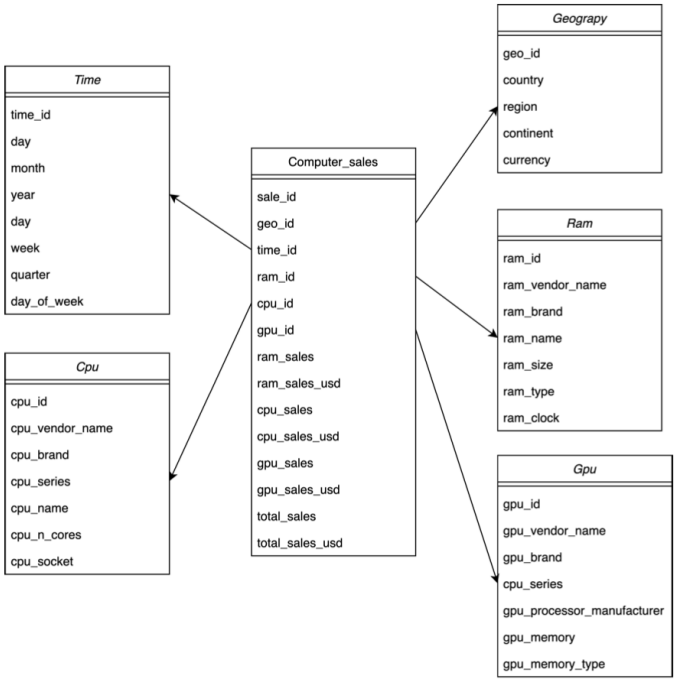

Figure 1: Schema di riferimento del Datawarehouse

- **Germany, Spain, Belgium, France, Ireland, e Italy:** EUR
- **Australia:** AUD
- **United Kingdom:** GBP
- **Canada:** CAD
- **New Zealand:** NZD
- **United States of America:** USD

time.py Il processo è iniziato con l'elaborazione dei codici di tempo (*time_ code*) presenti nel file `computer_sales.csv`, corrispondenti a vere e proprie date. Da ogni codice di tempo sono stati estratti il giorno del mese (*day*), il mese dell'anno (*month*), l'anno (*year*), la settimana dell'anno (*week*), il trimestre dell'anno (*quarter*) e il giorno della settimana (*day_ of_ week*).

Per l'estrazione del trimestre dell'anno in forma numerica (*n_ quarter*) è stata utilizzata la formula nell'equazione 1.

$$n\_quarter = (month - 1)//3 + 1 \tag{1}$$

Il risultato dell'equazione 1 è stato poi convertito in stringa e concatenato a una "Q", come da indicazioni del progetto (assigment 2).

Al contempo, l'attributo *day_ of_ week* è stato convertito da valore numerico a nome del giorno in formato stringa (es.: 1="Monday", 7="Sunday"), utilizzando l'apposita funzione di Python `dt.strftime('%A')`, disponibile nella libreria `datetime`. Successivamente, per ogni record è stato generato un identificativo artificiale (*time_ id*).

I dati sono infine stati scritti nel file `Time.csv`, che rappresenta la tabella temporale completa e strutturata.

cpu.py, gpu.py, e ram.py Per la creazione dei file di dati specifici di RAM, CPU e GPU, sono stati sviluppati tre differenti script Python per gestire l'estrazione e la strutturazione delle informazioni a partire dal file `computer_sales.csv`. Il processo ha seguito una metodologia comune per tutte e tre le componenti hardware.

Inizialmente, ogni script ha analizzato i dati di vendita per identificare le configurazioni uniche di RAM, CPU e GPU. Ogni configurazione è stata estratta e organizzata in una struttura dati Python, evitando duplicati e assicurando che ogni record fosse unico. Per ogni componente hardware, è stato assegnato un identificativo unico: *ram_id* per le RAM, *cpu_id* per le CPU, e *gpu_id* per le GPU.

I dati finali sono stati quindi scritti nei rispettivi file `RAM.csv`, `CPU.csv` e `GPU.csv`, secondo le indicazioni illustrate in figura 1.

computer_sales.py Per l'estrazione dei dati relativi alle vendite dei computer, è stato sviluppato uno script Python dedicato all'estrazione e alla trasformazione dei dati provenienti dal file `computer_sales.csv`.

In primo luogo, sono stati estratti all'interno dello script i dati dalle tabelle di dimensione create in precedenza, ovvero le tabelle per le informazioni geografiche, temporali, e relative a RAM, CPU e GPU. Questi dati sono stati organizzati in dizionari Python per un accesso e una manipolazione efficienti.

Successivamente, sono stati convertiti i valori delle vendite (*ram_sales*, *cpu_sales*, *gpu_sales*, *total_sales*) in dollari statunitensi (USD), originariamente espressi in diverse valute. Per la conversione sono stati usati i seguenti tassi di cambio rispetto a USD (1.00):

- **EUR:** 1.10
- **GBP:** 1.30
- **AUD:** 0.70
- **CAD:** 0.75
- **NZD:** 0.65

In questo modo è stato possibile ottenere i valori per gli attributi *ram_sales_usd*, *cpu_sales_usd*, *gpu_sales_usd*, e *total_sales_usd*.

Infine, grazie ai dati provenienti dalle tabelle di dimensione, e i nuovi valori calcolati con i tassi di conversione, è stato possibile scrivere il nuovo file `Computer_sales.csv`, secondo la configurazione indicata in figura 1.

2.2 Popolazione del database

Per il secondo assignment del progetto, abbiamo sviluppato uno script Python chiamato `loadData.py` per popolare il datawarehouse indicato (*Group_ID_480_DB*) con i dati ottenuti nelle fasi precedenti. Lo script stabilisce una connessione al database remoto e carica i dati dei file `Geography.csv`, `Time.csv`, `Cpu.csv`, `Gpu.csv`, `Ram.csv`, e `Computer_sales.csv` nelle omonime tabelle.

In primo luogo, lo script controlla che le tabelle di nostro interesse esistano già all'interno del datawarehouse. Nel caso in cui una tabella non esista, lo script procede a crearla con un apposita query SQL di tipo `CREATE TABLE`. All'interno dello script è stato preparato un dizionario contenente una query apposita per ogni tabella.

Lo script procede quindi con la popolazione delle tabelle di dimensione, per poi concludere con la *fact table*. Per rendere il processo più rapido ed efficiente, la popolazione è stata eseguita in *batches* di 1000 istanze. L'intero processo è stato monitorato attraverso un indicatore di progresso per garantire che il caricamento fosse completato correttamente.

Una volta che tutti i dati sono stati inseriti nelle rispettive tabelle, lo script esegue il commit delle transazioni e chiude la connessione al database.

3 ETL Process

In questa fase del progetto (assignment 3) abbiamo sviluppato un processo ETL (Extract, Transform, and Load) che si compone di due fasi:

1. Identificare, per ciascuna combinazione di anno (*year*) e regione (*region*), i computer (*sale_id*) con la CPU venduta al prezzo più alto
2. Calcolare, per ogni computer individuato nella fase 1, la percentuale corrispondente alla vendita di quel computer sul totale delle vendite di computer con la stessa serie di CPU (*cpu_series*) nella stessa combinazione di anno e regione

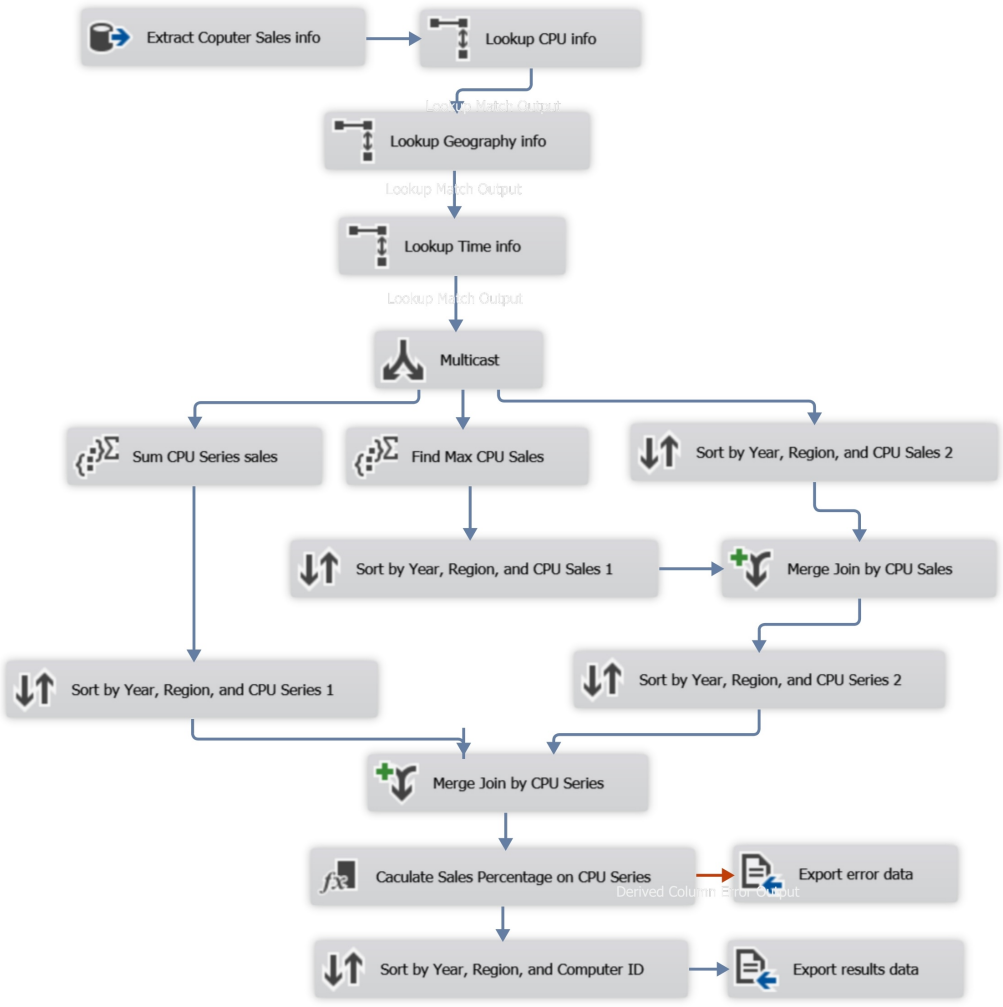

Figure 2: ETL process

Come mostrato dalla figura 2, il processo si compone di numerose fasi:

1. **Extract Computer Sales Info:** La prima fase del processo consiste nell'estrazione delle informazioni sulle vendite di computer dalla nostra base dati. Questo rappresenta il punto di ingresso dei dati nel flusso ETL.

2. **Lookup CPU, Geography, and Time Info:** Successivamente, il processo esegue delle operazioni di *lookup* per arricchire i dati estratti con le informazioni necessarie relative alle CPU, alla geografia e al tempo. Abbiamo deciso di utilizzare dei *lookup* e non dei *merge join*, in quanto le tabelle in questione risultano essere piccole e veloci da esplorare. Un merge join avrebbe imposto di estrarre direttamente i dati dalle tabelle del database, e di ordinarli tutti, allungando il processo.

3. **Multicast:** A questo punto, i dati vengono duplicati per poter essere processati in tre sotto-flussi distinti:

   (a) **Sum CPU Series Sales:** Nel primo sotto-flusso, i dati vengono aggregati per ottenere il totale delle vendite di ogni serie di CPU per ogni combinazione di anno e regione. Successivamente, i dati aggregati vengono ordinati per anno, regione e serie di CPU.

   (b) **Find Max CPU Sales:** Nel secondo sotto-flusso, si calcola il valore massimo a cui è stata venduta una CPU per ogni combinazione di anno e regione. Anche qui, i dati vengono ordinati per anno, regione e prezzo di vendita della CPU.

   (c) **Merge Join by CPU Sales:** Nel terzo sotto-flusso, il sotto-flusso b viene riunito alle informazioni originali delle vendite tramite un *merge join*, in base alle combinazioni di anno, regione e prezzo di vendita della CPU.[1]

4. **Merge Join by CPU Series:** per riunire il sotto-flusso a con il risultato dei sotto-flussi b e c, il processo effettua un *merge join* sulle combinazioni di Anno, Regione e serie di CPU.[2] In questo modo avremo finalmente arricchito le informazioni sulle vendite con i dati necessarie alla seconda fase del processo.

5. **Calculate Sales Percentage on CPU Series:** Una volta che i dati sono stati integrati, il processo procede al calcolo della percentuale delle della vendita di ciascun computer (in USD) rispetto al totale delle vendite (sempre in USD) dei computer con la stessa serie di CPU all'interno della stessa combinazione di anno e regione. In caso di errori nella creazione della colonna derivata, questi vengono esportati attraverso un apposito file `txt`.

6. **Export results data:** Infine, i dati vengono ordinati per anno, regione e ID del computer, per poi essere esportati in un file `csv`.

[1] Si noti che anche se il *merge join* è nominato `Merge Join by CPU Sales`, il join avviene per le combinazioni di anno, regione e prezzo di vendita delle CPU, non solo per quest'ultimo

[2] Si noti che anche se il *merge join* è nominato `Merge Join by CPU Series`, il join avviene per le combinazioni di anno, regione e serie di CPU, non solo per quest'ultima

4 Data Cube

In questa fase del progetto abbiamo creato un cubo di dati (*Group ID 480 Cube*) utilizzando Visual Studio, con l'obiettivo di analizzare le vendite di computer in modo strutturato e multidimensionale. Una parte cruciale di questo processo è stata la definizione delle dimensioni e la creazione delle gerarchie all'interno di ciascuna di esse, per facilitare un'analisi esaustiva dei dati.

Dimensione *Time* La gerarchia definita per la dimensione *Time* consente un'analisi temporale dettagliata e comprende i seguenti livelli:

- *Year* > *Quarter* > *Month* > *Week* > *Day* > *Time id*

Questa gerarchia permette di navigare attraverso i dati a diversi livelli temporali, dal generale al più specifico. Creare una gerarchia che va dal livello più ampio fino al livello più dettagliato permette di osservare le tendenze e le variazioni delle vendite nel tempo.

Dimensione *Geography* **Gerarchia**: Per la dimensione *Geography*, è stata creata una gerarchia basata su livelli geografici, come segue:

- *Continent* > *Country* > *Region* > *Geo id*

Questa struttura consente di analizzare le vendite su scala globale, continentale, nazionale e regionale. Questo permette di individuare mercati chiave, regioni con prestazioni superiori o inferiori, e di condurre analisi comparative tra diverse aree geografiche.

Dimensione *CPU* **Gerarchia**: La gerarchia della dimensione *CPU* è stata progettata per riflettere le caratteristiche commerciali delle CPU, come indicato di seguito:

- *Brand* > *Series* > *Socket* > *Name* > *Cpu id*

Questa gerarchia è utile per confrontare le performance di vendita tra diverse marche, serie e modelli di CPU.

Dimensione *GPU* **Gerarchia**: La gerarchia nella dimensione *GPU* è stata semplificata per concentrarsi sulle seguenti caratteristiche:

- *Brand* > *Series* > *Gpu id*

Questo consente un'analisi delle vendite delle GPU per marca e serie, facilitando il confronto tra diverse famiglie di prodotti.

Dimensione *RAM* **Gerarchia**: Infine, la gerarchia per la dimensione *RAM* è strutturata come segue:

- *Brand* > *Name* > *Ram id*

Questa gerarchia permette di analizzare le vendite di RAM distinguendo tra le diverse marche e i relativi modelli.

5 MDX Query

Per questo assignment abbiamo sviluppato delle query MDX per estrarre e analizzare i primi 5 brand per vendite in ciascuna categoria hardware (RAM, CPU, e GPU), per ogni regione Europea. I risultati offrono una visione comparativa delle prestazioni dei diversi marchi nelle specifiche regioni in Europa.

La struttura delle query è identica per ogni categoria di hardware. Qui di seguito presentiamo la logica dietro alla query relativa alle CPU:

1. Calcolo delle vendite medie mensili per ogni brand

```
WITH
    MEMBER [Measures].[AvgMonthlyCpuSales] AS
        AVG( [Time].[Month].MEMBERS, [Measures].[Cpu Sales] )
```

   - `MEMBER [Measures].[AvgMonthlyCpuSales] AS`: Crea un membro calcolato chiamato `[AvgMonthlyCpuSales]`.
   - `AVG([Time].[Month].MEMBERS, [Measures].[Cpu Sales])`: Calcola la media delle vendite di CPU (`[Measures].[Cpu Sales]`) per ogni mese (`[Time].[Month].MEMBERS`). Questo calcolo restituisce il valore medio delle vendite mensili di CPU per ciascun brand di CPU.

2. Selezione dei Top 5 Brand di CPU per Regione

```
SET [TopCpuBrands] AS
    TOPCOUNT(
        EXCEPT(
            [Cpu].[Cpu Brand].MEMBERS,
            {[Cpu].[Cpu Brand].[All]} -- Escludi il membro "All"
        ),
        5,
        [Measures].[AvgMonthlyCpuSales]
    )
```

   - `SET [TopCpuBrands] AS`: Definisce un set calcolato chiamato `[TopCpuBrands]` che contiene i top 5 brand di CPU.
   - `TOPCOUNT(5, [Measures].[AvgMonthlyCpuSales])`: restituisce i top 5 membri di un set ordinati in base a una misura specificata. Qui si ordinano i brand di CPU in base alle vendite medie mensili calcolate in precedenza.
   - `EXCEPT([Cpu].[Cpu Brand].MEMBERS, [Cpu].[Cpu Brand].[All])`: Esclude il membro aggregato "All" dalla lista di tutti i brand di CPU (`[Cpu].[Cpu Brand].MEMBERS`). Si considerano solo i brand specifici.

3. Selezione e Visualizzazione dei Dati

```
SELECT {
        [Measures].[AvgMonthlyCpuSales]
    } ON COLUMNS,

    NON EMPTY (
        [TopCpuBrands] * [Geography].[Region].MEMBERS
    ) ON ROWS

FROM [Group ID 480 Cube]

WHERE
    [Geography].[Continent].[Europe]
```

- `[Measures].[AvgMonthlyCpuSales] ON COLUMNS`: La misura calcolata delle vendite medie mensili di CPU viene visualizzata nelle colonne.
- `NON EMPTY([TopCpuBrands] * [Geography].[Region].MEMBERS) ON ROWS`:
  - `NON EMPTY`: Rimuove le righe senza dati.
  - `[TopCpuBrands] * [Geography].[Region].MEMBERS`: Crea una griglia combinando i top 5 brand di CPU con tutte le regioni geografiche.
- `FROM [Group ID 480 Cube]`: Specifica il cubo da cui estrarre i dati.
- `WHERE [Geography].[Continent].[Europe]`: Filtra i dati per considerare solo le vendite in Europa.

6 Dashboard

Nell'ambito dell'analisi dei dati estratti dal nostro cubo multidimensionale, abbiamo sviluppato una dashboard interattiva per visualizzare in modo efficace le informazioni più rilevanti e le tendenze emergenti. La dashboard è stata progettata per fornire una panoramica completa e dettagliata delle vendite di computer, comprese le loro componenti (CPU, RAM e GPU) nelle diverse località geografiche e nei vari periodi di tempo.

La figura 3 mostra la dashboard progettata attraverso il software Microsoft Power BI. All'interno della dashboard è possibile trovare i seguenti elementi:

- **KPI Cards:** nella parte superiore vengono mostrati dei pannelli, ognuno contenente un indicatore per un diverso aspetto di vendite. Partendo da sinistra, è possibile visualizzare il ricavo totale delle vendite, il ricavo totale delle CPU, delle RAM e delle GPU. Infine, a destra, il numero totale dei computer venduti. Per una visione coerente, i valori hanno tutti la stessa valuta, ovvero il dollaro (USD).

- **Lineplot:** nella parte centrale della dashboard è presente un *lineplot* in cui viene mostrato l'andamento del totale delle vendite (asse y) in base al trimestre di ciascun anno (asse x). Grazie agli appositi comandi di Microsoft Power BI, è possible ottenere una visualizzazione ancora più dettagliata sull'asse x, andando visualizzare l'andamento per ogni mese, oppure più generica, rimuovendo l'indicazione del trimestre.

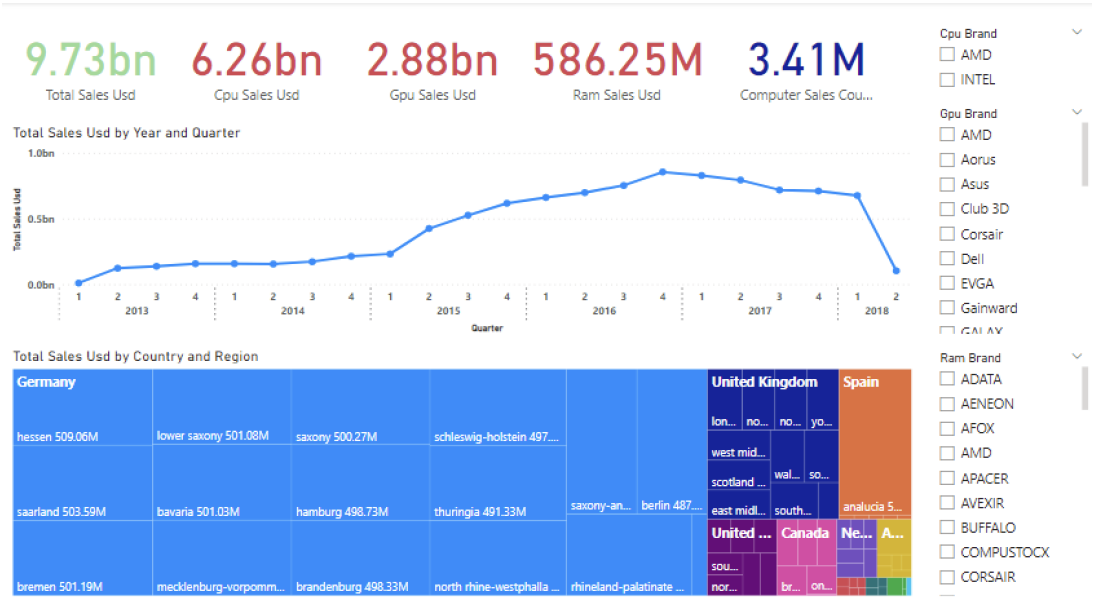

Figure 3: Dashboard Power BI

- **Treemap:** nella parte inferiore della dashboard abbiamo inserito una *treemap* per rappresentare i dati relativi alle vendite per area geografica. La treemap permette di confrontare rapidamente le vendite tra paesi e tra regioni, evidenziando immediatamente quali presentano ricavi maggiori. Abbiamo valutato di utilizzare una *map*, o una *filled map*; tuttavia, dopo alcuni esperimenti, abbiamo notato che queste tipologie di widget sarebbero state troppo dispersive e di meno immediata comprensione. I paesi e le regioni coinvolte, infatti, sono un numero limitato e disperso. La visualizzazione delle informazioni su una *map* sarebbe stata meno chiara.

- **Slicer:** nella parte destra della dashboard sono stati inseriti 3 componenti di tipo *Slicer*. Essi permettono di filtrare e di modificare il valore e l'andamento degli altri componenti sulla base di diversi brandi di CPU, RAM e GPU. In questo modo, è possibile ottenere una panoramica delle informazioni anche per singoli marchi, o specifiche combinazioni di essi. Senza una selezione dei brand, le informazioni visualizzate nella dashboard si riferiscono a tutti marchi presenti nel database.