MARCEAN Mihai-Alexandru
IAG – Gr. 10LF432

Proiect E-Commerce

Data Analysis

EDP – Exploratory Data Analysis

realizat de

Marcean Mihai-Alexandru
(Grupa: 10LF432)

Universitatea
Transilvania
din Brașov
FACULTATEA DE MATEMATICĂ
ȘI INFORMATICĂ

Facultatea de Matematica Informatica
IAG – Informatica Aplicata in limba Germana
Universitatea Transilvania, Brasov 2025

Contents

Contents

1. Introducere

În era actuală a volumelor masive de date generate zilnic, capacitatea de a extrage informații valoroase din aceste date reprezintă un avantaj strategic esențial. În acest context, proiectul de față își propune să realizeze o analiză exploratorie a datelor (Exploratory Data Analysis – EDA) utilizând un lanț tehnologic format din: **Python** (pentru prelucrare și încărcare date), **Microsoft SQL Server** (ca bază de date relațională) și **Power BI** (pentru vizualizare interactivă și raportare).

Tema aleasă reflectă o abordare practică și actuală în domeniul analizei datelor, integrând instrumente utilizate frecvent în mediul profesional. Un astfel de proiect poate servi drept fundament pentru dashboard-uri interactive utilizate în companii pentru a înțelege comportamentul clienților, dinamica vânzărilor și alte metrici relevante.

Scopul acestui proiect este de a transforma un set de date brute – în cazul de față, un dataset real de tranzacții comerciale online – într-o colecție coerentă de informații utile, accesibile printr-un dashboard Power BI conectat la o bază de date SQL Server. Această platformă poate fi extinsă și reutilizată în contexte reale de business, devenind o soluție scalabilă pentru analiza și raportarea automată a datelor.

GitHub Link: https://github.com/alexmarcean/edp-ecommerce-data-analysis

Structura lucrarii

Lucrarea este organizată pe următoarele capitole/sectiuni:

1. **Introducere** – prezentarea temei și a justificării practice.
2. **Literature Review** – un scurt review asupra practicilor de EDA și a instrumentelor similare.
3. **Descrierea și Structura Proiectului** – prezentarea componentelor utilizate (Python, SQL Server, Power BI) și exemple de cod relevante.
4. **Analiză comparativă și critică** – discuție asupra avantajelor, dezavantajelor și alternativelor tehnologice.
5. **Concluzii** – concluzii asupra impactului și utilității soluției dezvoltate.
6. **Bibliografie** – surse academice și tehnice utilizate în documentare.

2. Literature Review - Tehnologia aleasa si contributii similare

Exploratory Data Analysis (EDA) este recunoscută ca etapă preliminară în orice proces de modelare statistică sau machine learning. În lucrarea sa fundamentală *Exploratory Data Analysis* (1977), John Tukey sublinia faptul că statistica trebuie să fie orientată spre descoperirea informațiilor, nu doar spre confirmarea ipotezelor. Astăzi, această filosofie este aplicată extensiv în industrii diverse: de la marketing și e-commerce până la analiza riscurilor financiare și detectarea fraudelor.

2.1 Tehnologii implicate și rolul lor

Python s-a impus ca limbajul principal în analiza datelor datorită ecosistemului său bogat:

- `pandas` pentru manipulare de tabele și agregări rapide;
- `matplotlib` și `seaborn` pentru vizualizări;
- `numpy` pentru calcule numerice;
- `pyodbc` și `sqlalchemy` pentru conectivitate cu baze de date.

Microsoft SQL Server este o soluție matură și scalabilă de stocare a datelor, utilizată de companii la scară largă. Oferă suport pentru interogări complexe, indexare eficientă și vizualizare prin SSMS (SQL Server Management Studio). În contextul proiectului, SQL Server devine interfața centrală de stocare și centralizare a datelor brute și prelucrate.

Power BI transformă datele în insight-uri vizuale. Este alegerea populară în multe organizații pentru că:

- permite conectarea la surse diverse (Excel, CSV, baze de date, API-uri);
- oferă unelte de modelare (tabele de calendar, coloane calculate, măsurători cu DAX);
- suportă filtrare interactivă, segmentări și drill-down pentru analize detaliate.

2.2 Abordări similare în industrie

Proiecte similare sunt frecvent întâlnite în zona de **dashboarding operațional** – spre exemplu:

- analiza vânzărilor în retail și e-commerce;
- urmărirea performanței angajaților în HR analytics;
- monitorizarea comenzilor și retururilor în logistică.

Studiile de caz oferite de **Microsoft** în **Power BI Community** sau Kaggle Projects includ fluxuri în care:

- datele sunt extrase prin Python sau R,
- sunt stocate în SQL Server sau Azure Data Warehouse,
- și apoi sunt vizualizate în Power BI sau Tableau.

Aceste proiecte însă se bazează adesea pe servicii plătite (Azure, Power BI Service), ceea ce le face mai greu accesibile studenților sau companiilor mici.

2.3 Valoarea adăugată a proiectului curent

Proiectul propus se remarcă prin:

- **Combinarea coerentă** a mai multor tehnologii open-source și gratuite;
- **Accesarea programatică** a unui dataset de tip real-world (Online Retail II de la UCI Machine Learning);
- **Modelare relațională corectă** prin SQL Server, înlocuind fișierele nestructurate;
- **Realizarea unui dashboard complet interactiv** în Power BI, capabil să fie actualizat cu date noi printr-un script Python reutilizabil.

2.4 Remarci critice și perspective

Un obstacol des întâlnit este lipsa interoperabilității între tool-uri: de exemplu, Power BI nu poate rula cod Python direct pe surse externe fără **Power BI Service**. Proiectul nostru evită aceste limitări printr-un model în care **Python** joacă rolul de preprocesor și **ETL** (Extract-Transform-Load), în timp ce **SQL Server devine sursa intermediară** pentru raportare.

Această arhitectură modulară, ușor de replicat, permite:

- înlocuirea dataset-ului fără refactorizare masivă;
- extinderea cu surse multiple în viitor (API-uri, CSV-uri, scraping web);
- adaptarea ușoară la contexte industriale reale.

3. Descrierea și Structura Proiectului

Această secțiune constituie nucleul practic al lucrării și reflectă integrarea reală a mai multor tehnologii – Python, SQL Server și Power BI – într-un flux coerent de tip ETL + BI. Procesul implementat ilustrează transformarea unui fișier Excel brut într-o bază de date relațională, modelarea logică a datelor și, în final, extragerea de informații relevante prin vizualizări interactive.

Vom împărți explicația în 4 sub-secțiuni:

- **3.1 Extragerea și prelucrarea datelor cu Python**
- **3.2 Încărcarea datelor în SQL Server**
- **3.3 Modelare și transformări în Power BI**
- **3.4 Vizualizări și indicatori de performanță**

3.1 Extragerea și prelucrarea datelor cu Python

Setul de date utilizat este *Online Retail II*, disponibil public prin UCI Machine Learning Repository. Acesta conține informații despre comenzile efectuate într-un magazin online britanic în perioada 2009–2011.

Fișierul Excel conține două foi: una pentru anul 2009–2010 și alta pentru 2010–2011. Pentru procesare, a fost folosit un script Python care:

- Încarcă ambele foi într-un `DataFrame` unic;
- Normalizează datele lipsă (`NaN`) în `None` (compatibile cu SQL);
- Se conectează la un server SQL local folosind `pyodbc`;
- Creează tabela `OnlineRetail` dacă aceasta nu există deja.

Fragment de cod utilizat pentru citirea și concatenarea sheet-urilor:

```
import pandas as pd
import pyodbc

# === Step 1: Load Excel File and Both Sheets ===
excel_file = "online_retail_II.xlsx"  # Ensure this file is in the same folder
sheet_names = ["Year 2009-2010", "Year 2010-2011"]

# Read and concatenate both sheets
try:
    df_list = [pd.read_excel(excel_file, sheet_name=sheet) for sheet in sheet_names]
    df = pd.concat(df_list, ignore_index=True)
    print("Both Excel sheets loaded and merged successfully.")
except Exception as e:
    print(f"Failed to load Excel sheets: {e}")
    exit()
```

Figure 1 – Citire si concatenare din sheet excel.

3.2 Încărcarea datelor în SQL Server

După prelucrarea inițială a datasetului folosind Python, a urmat etapa de persistare a datelor într-o bază de date relațională – **Microsoft SQL Server 2021 Developer Edition**. Scopul acestui pas este acela de a avea o sursă de date stabilă și ușor de integrat cu instrumente de analiză precum Power BI, dar și pentru a învăța lucrul cu tehnologii de tip **ETL** (Extract – Transform – Load) într-un flux complet.

Crearea bazei de date și a conexiunii

Baza de date `OnlineRetailDB` a fost creată din SQL Server Management Studio (SSMS), oferind un spațiu logic pentru păstrarea tabelului principal. Conectarea la server s-a realizat folosind librăria `pyodbc`, care permite conectarea Python cu SQL Server prin driver ODBC:

```
# === Step 2: Connect to SQL Server ===
try:
    conn = pyodbc.connect(
        'DRIVER={ODBC Driver 17 for SQL Server};'
        'SERVER=localhost;'                    # Change if needed
        'DATABASE=OnlineRetailDB;'             # Your database name
        'Trusted_Connection=yes;'              # Or use UID and PWD
    )
    cursor = conn.cursor()
    print("Connected to SQL Server.")
except Exception as e:
    print(f"Database connection failed: {e}")
    exit()
```

Figure 2 – Connect to SQL Server

Această conexiune este esențială pentru a rula comenzi T-SQL din cod Python, fără intervenție manuală.

Structura tabelei OnlineRetail

Tabelul principal a fost denumit `OnlineRetail` și a fost creat doar dacă nu exista deja în baza de date. Comanda de creare a fost executată tot din Python, folosind un query T-SQL:

```
# === Step 3: Create Table If Not Exists ===
create_table_query = """
IF NOT EXISTS (SELECT * FROM sysobjects WHERE name='OnlineRetail' AND xtype='U')
CREATE TABLE OnlineRetail (
    Invoice NVARCHAR(55),
    StockCode NVARCHAR(55),
    Description NVARCHAR(255),
    Quantity INT,
    InvoiceDate DATETIME,
    Price FLOAT,
    Customer_ID INT,
    Country NVARCHAR(100)
)
"""
```

Figure 3 – Comanda de creare a tabelelor

Justificarea alegerii tipurilor de date:

- `NVARCHAR` pentru coloanele de tip text (coduri, descrieri, țări), pentru a accepta și caractere internaționale.
- `INT` pentru coloane numerice întregi (cantitate, ID client).
- `FLOAT` pentru coloana de preț, unde sunt necesare zecimale.
- `DATETIME` pentru a permite filtrarea pe intervale temporale în Power BI.

Inserarea datelor

După citirea și combinarea celor două foi Excel în Python (2009-2010 și 2010-2011), s-a realizat un loop prin rândurile DataFrame-ului pentru a introduce datele în SQL Server. Fiecare inserție este făcută în mod parametrizat, ceea ce oferă protecție împotriva erorilor și vulnerabilităților de tip SQL Injection.

```
try:
    cursor.execute(create_table_query)
    conn.commit()
    print("Table checked/created.")
except Exception as e:
    print(f"Table creation failed: {e}")
    conn.close()
    exit()
```

Figure 4 – Executarea comenzii de creare a tabelului in SSMS 2021

Inserarea se face **iterativ**, cu un mesaj de progres afișat la fiecare 1000 de rânduri. Acest mecanism este util mai ales când se lucrează cu fișiere mari și este necesară monitorizarea performanței.

În total, în urma rulării scriptului, peste **1.000.000 de rânduri** au fost încărcate cu succes în baza de date locală SQL Server.

Tratarea valorilor lipsă și a erorilor

Fiind un dataset real, unele câmpuri conțineau valori lipsă (NaN). Acestea au fost convertite în None, care este interpretat în SQL drept NULL. Acest lucru a prevenit erorile de tip incompatibilitate la inserare:

```
# Replace NaN with None (SQL-compatible nulls)
df = df.where(pd.notnull(df), None)
```

Figure 5 – Tratarea valorilor lipsa

De asemenea, codul a fost structurat cu blocuri try-except pentru a prinde și gestiona elegant eventuale erori apărute în timpul conectării, creării tabelului sau inserării datelor.

Rezultatul final

După rularea completă a scriptului Python, baza de date OnlineRetailDB a fost populată cu date structurate, curate și coerente. Acest rezultat a oferit fundamentul solid pentru etapa următoare – conectarea Power BI la baza de date și construirea modelelor analitice și a vizualizărilor.

3.3 Modelarea datelor și Calendarul Power BI

După conectarea cu succes la baza de date SQL Server și importul tabelei OnlineRetail în Power BI, următorul pas esențial a fost construirea unui **model de date semantic** pentru analiză. Acest model permite explorarea datelor pe dimensiuni de timp, identificarea sezonalităților și oferă o flexibilitate crescută în construcția vizualizărilor.

Crearea tabelului Calendar

Tabelul Calendar este esențial în orice proiect Power BI care presupune analiză temporală (vânzări lunare, sezonalitate, Q-by-Q etc). Deoarece fișierul sursă conținea doar o coloană InvoiceDate, a fost nevoie să generăm o dimensiune de timp completă în Power BI, folosind DAX.

```
Calendar =
ADDCOLUMNS (
    CALENDAR (
        TRUNC(MIN(OnlineRetail[InvoiceDate])),
        TRUNC(MAX(OnlineRetail[InvoiceDate]))
    ),
    "Year", YEAR([Date]),
    "Month", FORMAT([Date], "MMMM"),
    "MonthNumber", MONTH([Date]),
    "Weekday", FORMAT([Date], "dddd"),
    "Quarter", "Q" & FORMAT([Date], "Q"),
    "WeekNumber", WEEKNUM([Date], 2)
)
```

Figure 6 – Crearea Calendarului in PowerBI folosind DAX

Crearea relației între *Calendar* și *OnlineRetail*

După crearea tabelului **Calendar**, acesta trebuie **conectat** la tabelul **OnlineRetail** pentru a permite filtrarea corectă în funcție de timp.

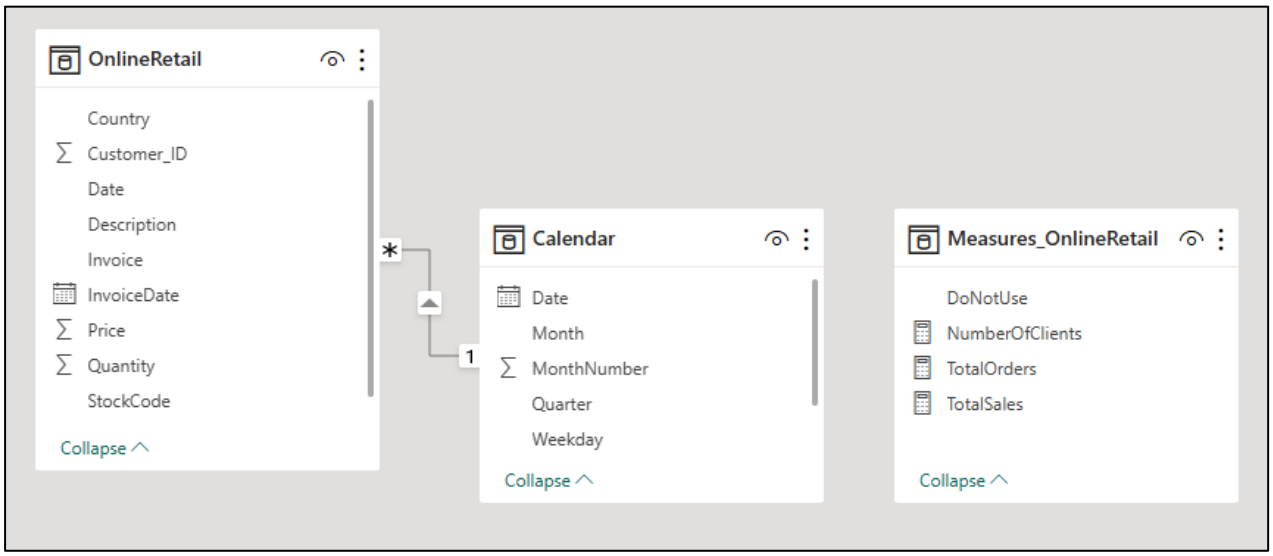

Figure 7 – Crearea relatiei intre tabele

Crearea măsurilor DAX (KPI-uri)

Pe baza datelor și a calendarului, au fost definite 3 măsuri fundamentale:

1. TotalSales (vânzări totale):

```
TotalSales =
    SUMX(OnlineRetail, OnlineRetail[Quantity] * OnlineRetail[Price])
```

Figure 8 – TotalSales PowerBI (DAX)

2. NumberOfClients (număr unic de clienți)

```
NumberOfClients =
    DISTINCTCOUNT(OnlineRetail[Customer_ID])
```

Figure 9 – NumberOfClients PowerBI (DAX)

3. TotalOrders (număr unic de comenzi)

```
TotalOrders =
    DISTINCTCOUNT(OnlineRetail[Invoice])
```

Figure 10 – TotalOrders PowerBI (DAX)

Importanța acestei etape

Această etapă asigură:

- corectitudinea analizelor de-a lungul timpului;
- posibilitatea filtrării pe ani, luni, săptămâni, trimestre;
- agregarea măsurilor prin relații bine definite;
- și, nu în ultimul rând, **performanță crescută** în vizualizări și interactivitate.

3.4 Crearea Vizualizărilor în Power BI

Vizualizări în Power BI

Pentru a facilita interpretarea și analiza vizuală a datelor prelucrate, au fost utilizate mai multe tipuri de vizualizări disponibile în Power BI. Aceste componente grafice au avut rolul de a oferi perspective clare asupra performanțelor de vânzare, comportamentului clienților și distribuției geografice a comenzilor. Vizualizările selectate au fost alese în funcție de natura datelor și de obiectivele analizei.

Matrix

Vizualizarea de tip Matrix a fost utilizată pentru a construi un tabel pivotat, care afișează vânzările totale agregate pe două dimensiuni: țara și luna. Acest tip de tabel permite ierarhizarea datelor și evidențiază în mod eficient corelațiile între locație și perioadă. Fiecare rând din tabel corespunde unei țări, iar coloanele corespund lunilor calendaristice. Valorile celulelor indică vânzările totale (măsura *TotalSales*), oferind astfel o imagine sintetică și comparativă între regiuni și intervale de timp.

Slicere

Slicerele reprezintă filtre vizuale care permit utilizatorului să selecteze dinamica datelor afișate în celelalte componente ale raportului. În cadrul acestui proiect, au fost inserate slicere pentru câmpurile *Year*, *Month* și *Country*, toate extrase din tabelul Calendar. Aceste slicere permit o interactivitate ridicată, permițând utilizatorilor să restrângă vizualizările la o anumită perioadă sau regiune de interes. Astfel, dashboard-ul devine flexibil și adaptabil în timp real, în funcție de selecțiile utilizatorului.

Area Chart

Graficul de tip *Area* a fost ales pentru a evidenția evoluția vânzărilor în timp. Pe axa X este plasată dimensiunea *YearMonth*, rezultată din concatenarea anului și lunii din tabelul Calendar, iar pe axa Y este reprezentată măsura *TotalSales*. Suprafața umplută sub linie accentuează variațiile de volum de la o lună la alta. Acest grafic este deosebit de util pentru a urmări trendurile și pentru a identifica sezonalitatea sau punctele de maxim/minim în comportamentul de cumpărare al clienților.

Card și Donut Chart

Vizualizarea *Card* a fost utilizată pentru a evidenția valori sintetice importante, precum *TotalSales*, *NumberOfClients* și *TotalOrders*. Aceste carduri oferă o formă de sumarizare numerică, concentrând atenția asupra principalilor indicatori de performanță (KPI). În completare, *Donut Chart*-ul a fost folosit pentru a reprezenta proporția vânzărilor în funcție de țări. Acesta oferă o imagine de ansamblu asupra distribuției procentuale a vânzărilor, contribuind la identificarea rapidă a piețelor dominante. Amplasarea celor două vizualizări în proximitate permite corelarea valorilor absolute cu proporțiile relative.

Map

Vizualizarea de tip *Map* a permis integrarea unei hărți interactive în raport, prin care pot fi localizate geografic comenzile. Fiecare țară este reprezentată printr-un punct pe hartă, a cărui dimensiune reflectă valoarea totală a vânzărilor generate în acea regiune. Harta utilizează serviciul Bing Maps integrat în Power BI pentru geocodare automată și permite interacțiune prin zoom, selecție și hover pentru detalii adiționale. Acest tip de vizualizare este valoros în analizarea distribuției geografice și în identificarea piețelor cu performanțe ridicate.

Toate aceste vizualizări contribuie la obținerea unui dashboard interactiv, informativ și ușor de utilizat, oferind utilizatorilor instrumentele necesare pentru o analiză aprofundată și intuitivă a datelor comerciale.

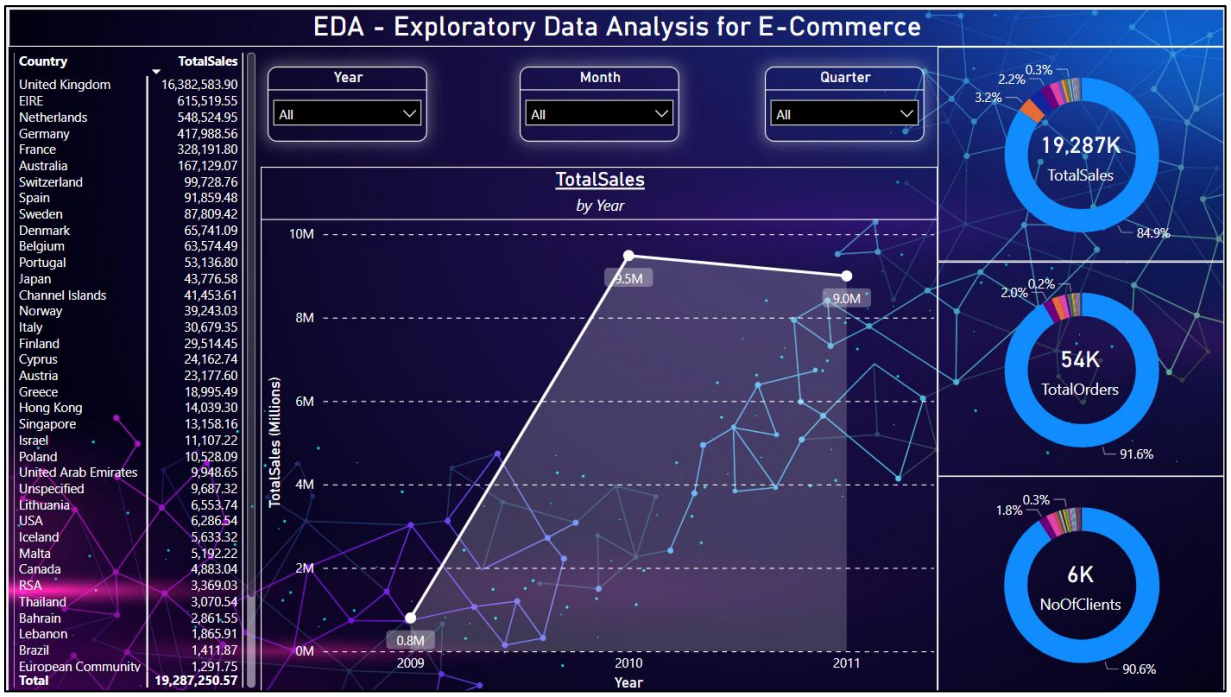

Figure 11 – Sales Data (Year)

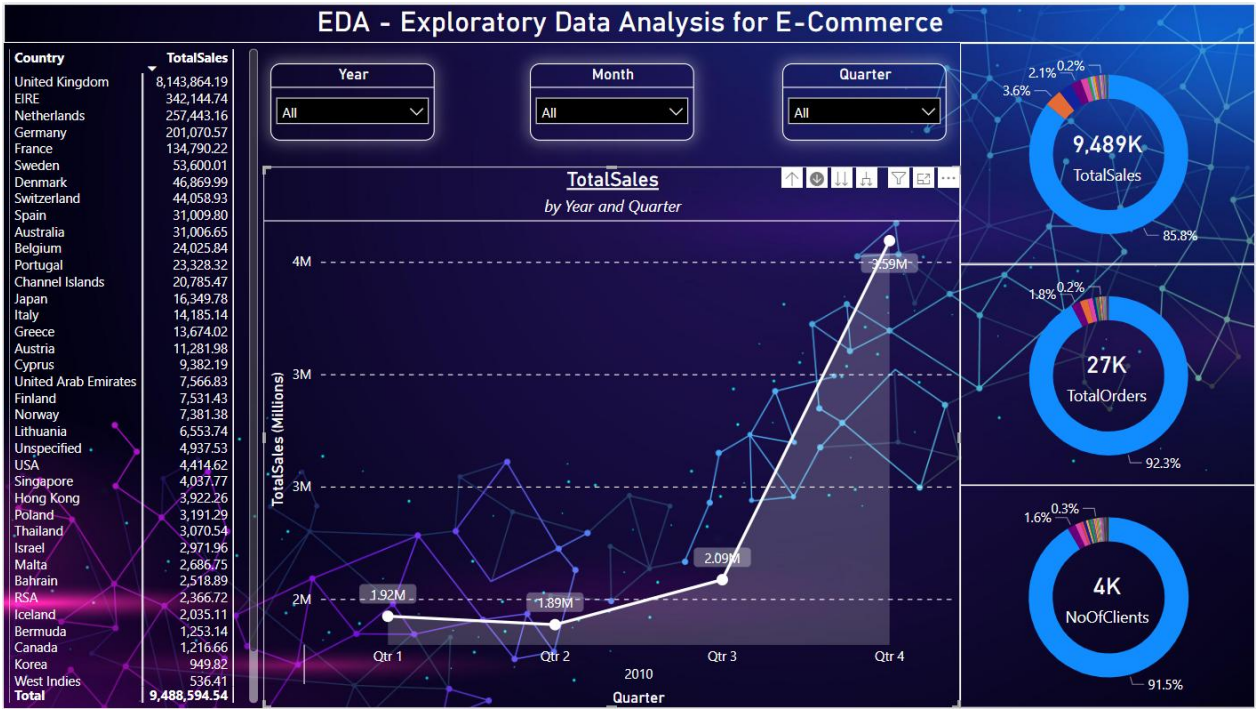

Figure 12 – Sales Data (Quarter)

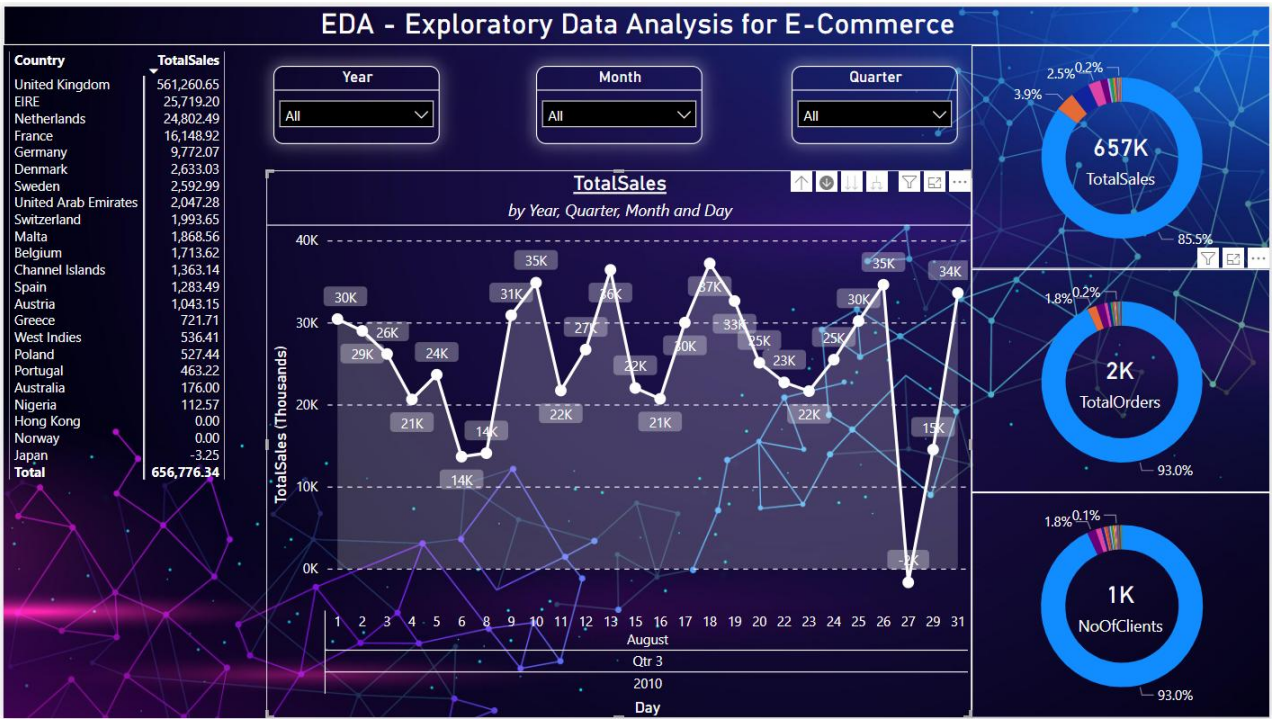

Figure 13 – Sales Data (Days)

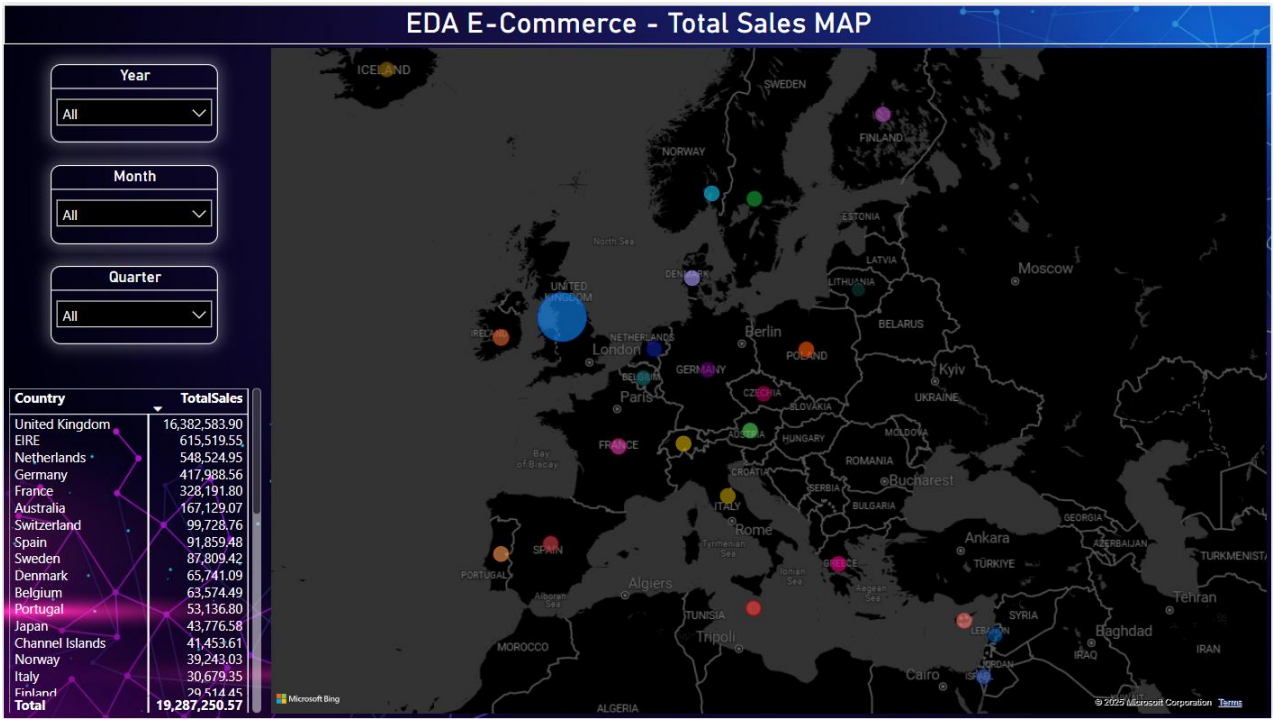

Figure 14 – Countries Data (MAP)

4. Analiză comparativă și critică

În cadrul acestei lucrări a fost utilizat Power BI ca principal instrument de analiză vizuală a datelor, integrat cu Python și SQL Server pentru extragerea și modelarea inițială a datelor. Alegerea acestui ecosistem tehnologic se justifică atât prin capabilitățile analitice ridicate, cât și prin accesibilitatea pentru utilizatorii non-tehnici și integrarea fluidă cu surse de date diverse. Totuși, este important să analizăm în mod comparativ avantajele și limitele acestui set de instrumente în raport cu alte tehnologii și abordări similare disponibile în mediul profesional și academic.

4.1 Compararea Power BI cu alte instrumente BI

Una dintre cele mai apropiate alternative la Power BI este **Tableau**, un alt lider în domeniul vizualizării datelor. Tableau oferă o flexibilitate ridicată în construcția de vizualizări avansate și o interfață performantă pentru explorarea datelor în timp real. Cu toate acestea, Power BI are un avantaj major în integrarea cu alte produse Microsoft, precum Excel, Azure și SQL Server, ceea ce reduce semnificativ barierele de interoperabilitate în ecosistemele Microsoft. În plus, licențierea Power BI este adesea mai accesibilă pentru mediile educaționale și organizațiile mici.

În comparație cu **Google Data Studio**, Power BI oferă funcționalități mai avansate în ceea ce privește transformarea datelor, expresiile DAX, și conectivitatea cu baze de date enterprise. Deși Google Data Studio este gratuit și suficient pentru rapoarte simple, este limitat în cazul analizelor complexe, în special cele care implică calcule interdependente, măsuri compuse sau volume mari de date.

4.2 Rolul Python în procesul de prelucrare a datelor

Utilizarea limbajului Python a fost esențială în etapa de prelucrare inițială a datelor din fișierul Excel și pentru inserarea acestora într-o bază de date relațională SQL Server. Această abordare a oferit control total asupra transformării datelor, inclusiv tratarea valorilor lipsă, unificarea foilor de lucru și tipizarea câmpurilor. Avantajul major al acestei etape este reproductibilitatea și transparența: scriptul Python este ușor de înțeles, reutilizat și scalat pentru noi seturi de date similare.

Alternativ, Power Query din Power BI ar fi putut fi folosit pentru importul direct al fișierului Excel și modelarea datelor. Însă această soluție este mai puțin flexibilă când vine vorba de procese ETL avansate sau integrarea automată într-un pipeline. În schimb, utilizarea unui limbaj de programare oferă autonomie față de interfața grafică și mai multă precizie în procesarea loturilor de date.

4.3 Limite și provocări identificate

Una dintre limitele întâlnite în utilizarea Power BI a fost legată de capacitatea acestuia de a gestiona volume mari de date în versiunea gratuită. De asemenea, lucrul cu DAX poate deveni dificil în cazul calculelor complexe, mai ales pentru utilizatorii fără experiență anterioară în limbaje funcționale.

Pe de altă parte, integrarea Power BI cu SQL Server a fost eficientă și stabilă, permițând reîmprospătarea datelor din sursa relațională și automatizarea proceselor. Provocările majore au fost legate de compatibilitatea driverelor ODBC și de gestionarea erorilor de tip date lipsă sau neconforme în etapa de încărcare în baza de date.

4.4 Avantaje și perspective viitoare

Principalul avantaj al tehnologiei alese este combinația dintre flexibilitate (prin Python și SQL), integrarea cu instrumente vizuale puternice (Power BI), și posibilitatea publicării online gratuite a rapoartelor prin GitHub. De asemenea, utilizarea unui calendar personalizat și a măsurilor dinamice a dus la crearea unui dashboard interactiv și scalabil.

În perspectivă, analiza poate fi extinsă prin:

- integrarea unui sistem de actualizare automată a datelor prin scripturi programate;
- conectarea la surse de date live (API-uri, baze de date online);
- implementarea de algoritmi de machine learning pentru predicții de vânzări direct în Power BI, folosind limbaje R sau Python.

Astfel, tehnologia utilizată în acest proiect oferă o bază solidă pentru extinderea ulterioară către scenarii avansate de analiză predictivă și Business Intelligence.

5. Concluzii

Proiectul realizat a avut ca obiectiv explorarea și valorificarea tehnologiei Power BI în contextul analizei de date din domeniul e-commerce, folosind ca set de date un fișier istoric de tranzacții comerciale. Prin integrarea Power BI cu Python și SQL Server, a fost realizat un proces complet de colectare, prelucrare, modelare și vizualizare a datelor.

În prima etapă, datele dintr-un fișier Excel structurat pe doi ani au fost încărcate și curățate cu ajutorul bibliotecilor Pandas și PyODBC, apoi inserate într-o bază de date SQL Server. Această alegere a permis o manipulare eficientă a valorilor lipsă, tipurilor de date și structurii tabelare, pregătind astfel o sursă solidă pentru Power BI.

Ulterior, în Power BI, s-a construit un model de date extensibil, care a inclus un tabel calendaristic generat dinamic și o serie de măsuri DAX relevante (vânzări totale, număr de comenzi, număr de clienți). Prin intermediul acestui model, au fost dezvoltate vizualizări dinamice – grafice de tip line chart, area chart, donut chart, hărți geografice și tabele de tip matrix – care au permis o analiză granulară și interactivă a vânzărilor în funcție de timp, locație și comportamentul clientului.

Pe parcursul lucrării au fost discutate atât avantajele, cât și limitările acestei soluții tehnice. Power BI s-a dovedit a fi un instrument intuitiv, dar și puternic, în special în combinație cu limbaje de scripting precum Python. Publicarea pe GitHub a oferit transparență, acces public la sursă și posibilitatea colaborării sau reutilizării proiectului de către alte persoane interesate.

Dincolo de aspectul practic, proiectul a demonstrat importanța abordărilor multidisciplinare în analiza datelor: combinarea programării, bazelor de date, expresiilor DAX și designului de interfață vizuală a dus la o soluție completă și funcțională. Astfel, Power BI nu este doar un instrument de vizualizare, ci un punct central într-un ecosistem de analiză date moderne, cu aplicații reale în afaceri, educație și cercetare.

În concluzie, proiectul a oferit o experiență aplicativă valoroasă și a evidențiat potențialul ridicat al Power BI ca soluție de Business Intelligence, într-un flux de lucru integrat cu surse de date variate și unelte de prelucrare automatizate.

6. Bibliografie

1. Sharda, R., Delen, D., & Turban, E. (2020). *Business Intelligence, Analytics, and Data Science: A Managerial Perspective* (5th ed.). Pearson.
2. McKinney, W. (2022). *Python for Data Analysis: Data Wrangling with pandas, NumPy, and Jupyter* (3rd ed.). O'Reilly Media.
3. Microsoft Learn – Power BI Documentation https://learn.microsoft.com/en-us/power-bi/
4. GitHub – Proiectul personal publicat https://github.com/alexmarcean/edp-ecommerce-data-analysis
5. Mishra, A., & Sharma, D. (2020). *A comparative study of Power BI and Tableau for data visualization*. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*.