

Acum că Alex știe cum să se conecteze la o bază de date MySQL și să efectueze operații CRUD din Python, următorul pas este să învețe **cum să analizeze și să vizualizeze datele** folosind puternicele biblioteci Python **Pandas** și [Matplotlib](#).

De ce este important acest lucru?

SQL permite **stocarea structurată a datelor**, dar analiza necesită adesea **o abordare mai flexibilă**, care facilitează explorarea și interpretarea datelor.

- **Pandas** permite manipularea datelor sub forma unui tabel, similar cu Excel, ceea ce face analiza și filtrarea mai ușoară.
- **Matplotlib** și [Seaborn](#) permit vizualizarea rezultatelor prin grafice, diagrame și alte reprezentări vizuale ale datelor.

Ce vom învăța în această lecție?

- Cum să conectăm **Pandas** la o bază de date MySQL.
- Cum să convertim rezultatele interogării SQL într-un [Pandas DataFrame](#) pentru o analiză mai ușoară.
- Cum să utilizăm **Pandas** pentru **agregarea, filtrarea și gruparea datelor**.
- Cum să vizualizăm datele folosind **Matplotlib** și **Seaborn**.

Prin această lecție, Alex va învăța cum să folosească Python pentru analiza datelor de nivel superior, permițându-i **să extragă informații utile** și să le prezinte într-un mod clar și intuitiv.

Instalarea și pregătirea mediului pentru analiza datelor

Înainte de a începe să analizăm datele dintr-o bază de date MySQL

folosind **Pandas** și să vizualizăm datele folosind **Matplotlib** și **Seaborn**, trebuie să ne asigurăm că toate bibliotecile necesare sunt instalate.

1. Instalarea bibliotecilor necesare

Dacă nu ați instalat anterior bibliotecile pe care le veți folosi, executați următoarea comandă într-un terminal:

```
pip install pandas matplotlib seaborn mysql-connector-python
```

Ce instalăm?

- **Pandas** - permite lucrul cu date în formă tabelară (DataFrame).
- **Matplotlib** - o bibliotecă de bază pentru crearea graficelor și [vizualizarea datelor](#).
- **Seaborn** - un upgrade la Matplotlib cu vizualizări mai frumoase și mai avansate.
- **MySQL Connector** - permite conectarea Python-ului la o bază de date MySQL.

2. Conectarea la baza de date MySQL și descărcarea datelor

Alex vrea acum să aplice cunoștințele pe care le-a dobândit și să se conecteze la baza de date **MySQL** cu numele **library** care conține tabele precum **Book**, **Author**, **Genre**, **User** și **Rental**, pentru a analiza datele despre cărți.

Primul pas în acest proces este preluarea datelor din baza de date și convertirea acestora într-un format mai potrivit pentru analiză - **Pandas DataFrame**.

Să vedem cum Alex preia, adică selectează date dintr-o bază de date MySQL și le convertește în Pandas DataFrame, folosind bibliotecile **Pandas**, **Matplotlib** și **Seaborn**!

Exemplu: Selectarea datelor dintr-o bază de date MySQL și crearea unui Pandas DataFrame

În momentul de față, datele sunt stocate în tabelul MySQL cu numele **Book**, iar Alex dorește să le preia în mediul Python pentru a putea să le exploreze și vizualizeze.

```
import mysql.connector
import pandas as pd

# Creating a connection to the MySQL database
conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="root",
    database="library"
)

# SQL query to retrieve book data
query = "SELECT book_id, title, published, genre_id FROM Book"
df_books = pd.read_sql(query, conn)

# Displaying the first 5 rows of data
print(df_books.head())

# Closing the database connection
conn.close()
```

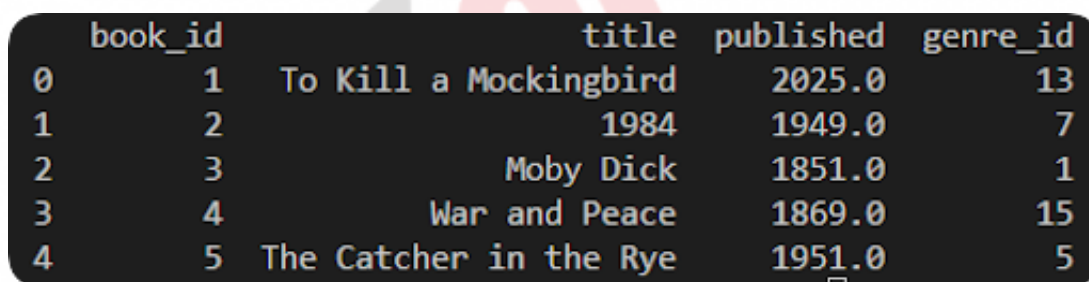
Ce se întâmplă aici?

- **Conectarea la baza de date** – creăm o conexiune la baza de

date MySQL folosind `mysql.connector`.

- **Executarea interogării SQL** - selectăm datele despre cărți folosind interogarea `SELECT`.
- **Conversia în Pandas DataFrame** - folosim `pd.read_sql()` pentru a stoca datele rezultate în format Pandas.
- **Afișarea datelor** - funcția `df_books.head()` ne permite să vedem primele câteva rânduri ale unui DataFrame.
- **Închiderea conexiunii** - după ce preluăm datele, închidem conexiunea la baza de date.

Rezultatul poate arăta astfel:



	book_id	title	published	genre_id
0	1	To Kill a Mockingbird	2025.0	13
1	2	1984	1949.0	7
2	3	Moby Dick	1851.0	1
3	4	War and Peace	1869.0	15
4	5	The Catcher in the Rye	1951.0	5

Imaginea 26.1. LibraryBooks

Acum că Alex are datele în **Pandas DataFrame**, le poate analiza și vizualiza folosind bibliotecile **Pandas**, **Matplotlib** și **Seaborn**!

3. Analiza datelor folosind Pandas

Acum că am preluat datele în DataFrame, le putem analiza folosind **metodele Pandas**.

Reprezentarea statisticilor de bază ale datelor

Pentru a obține o perspectivă rapidă asupra caracteristicilor numerice ale datelor din tabelul `df_books`, putem folosi funcția `describe()` din biblioteca Pandas. Această funcție generează un rezumat al datelor statistice.

Codul pentru generarea statisticilor de bază ale datelor:

```
print(df_books.describe())
```

Ce obținem cu această reprezentare?

- Numărul de valori nenule din fiecare coloană numerică (`count`).
- Valoarea medie (`mean`).
- Abaterea standard – cât de mult se abat datele de la medie (`std`).
- Cea mai mică și cea mai mare valoare dintr-un set de date (`min` și `max`).
- Quartile, sau valori care împart datele în sferturi (25%, 50%, 75%).

Rezultatul poate arăta astfel:

	book_id	published	genre_id
count	510.000000	506.000000	510.000000
mean	264.680392	1969.879447	23.880392
std	154.364397	148.388116	11.238065
min	1.000000	-800.000000	1.000000
25%	131.250000	1986.250000	14.250000
50%	262.500000	2002.000000	25.000000
75%	400.750000	2011.000000	34.000000
max	528.000000	2025.000000	43.000000

Imaginea 26.2. Ilustrarea datelor statistice de bază

Această reprezentare poate ajuta la analiza datelor, la detectarea potențialelor anomalii sau la identificarea tendințelor în anii de publicare și genurile de carte.

Reprezentarea tuturor genurilor diferite în baza de date

Pentru a afla ce genuri există în baza noastră de date, putem folosi funcția `unique()` din biblioteca **pandas**. Această funcție ne permite să extragem toate valorile unice din coloana `genre_id`, fără repetări.

Codul pentru generarea reprezentării tuturor genurilor diferite din baza de date este următorul:

```
print(df_books['genre_id'].unique())
```

Ce obținem cu acest cod?

- O listă cu toate valorile **genre_id** diferite care există în tabel.
- O reprezentare rapidă a genurilor disponibile în baza de date, care poate fi utilă pentru analiza datelor sau crearea interogărilor suplimentare.
- Această metodă ne poate ajuta să verificăm dacă toate genurile sunt introduse corect și dacă există erori sau valori lipsă.

Acest cod afișează **toate genurile unice** din baza de date.

Gruparea datelor - numărul de cărți după gen

Pentru a afla câte cărți aparțin fiecărui gen, putem folosi

funcția `groupby()` din biblioteca **pandas**. Prin gruparea datelor după coloana **genre_id** și numărând cărțile din fiecare grup, obținem o imagine de ansamblu asupra numărului de cărți după gen.

Codul pentru gruparea datelor din baza de date:

```
df_grouped = df_books.groupby("genre_id")["book_id"].count()  
print(df_grouped)
```

Cum funcționează?

- `groupby("genre_id")` - gruparea datelor în funcție de **genre_id**.
- `["book_id"].count()` - numărarea cărților care aparțin fiecărui gen.

Ce obținem cu acest cod?

- Reprezentarea numărului total de cărți de fiecare gen.
- O perspectivă asupra genurilor care sunt cel mai întâlnite și cel mai puțin reprezentate în baza de date.
- Util pentru a analiza popularitatea genurilor și pentru a echilibra colecția de cărți.

Rezultatul poate arăta astfel:

genre_id	total_books
1	8
2	8
3	4
4	6
5	2
6	28

Tabelul 26.1. GenreTotalBooks

Prin această grupare simplă, putem analiza rapid distribuția cărților pe genuri. Dacă dorim o analiză suplimentară, putem extinde interogarea și adăuga sortarea, filtrarea sau vizualizarea datelor folosind grafice!

3. Vizualizarea datelor folosind Matplotlib și Seaborn

Când lucrăm cu seturi mari de date, **reprezentările grafice** ne pot ajuta să înțelegem mai ușor tendințele și relațiile dintre date.

Histograma - distribuția anilor de publicare a cărții

Una dintre cele mai bune moduri de a vizualiza modul în care cărțile sunt aranjate în funcție de anul publicării este utilizarea unei **histograme**. Acest tip de diagramă ne permite să vedem câte cărți aparțin unor anumite intervale de timp, ceea ce poate ajuta la analiza tendințelor în domeniul publicării.

Cod pentru generarea histogramei:

```
import mysql.connector
import pandas as pd
import matplotlib.pyplot as plt

# Creating a connection to the MySQL database
conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="root",
    database="library"
)

# SQL query to retrieve book data
query = "SELECT book_id, title, published, genre_id FROM
Book"
```



```
df_books = pd.read_sql(query, conn)

# Display the first 5 rows of data
print(df_books.head())

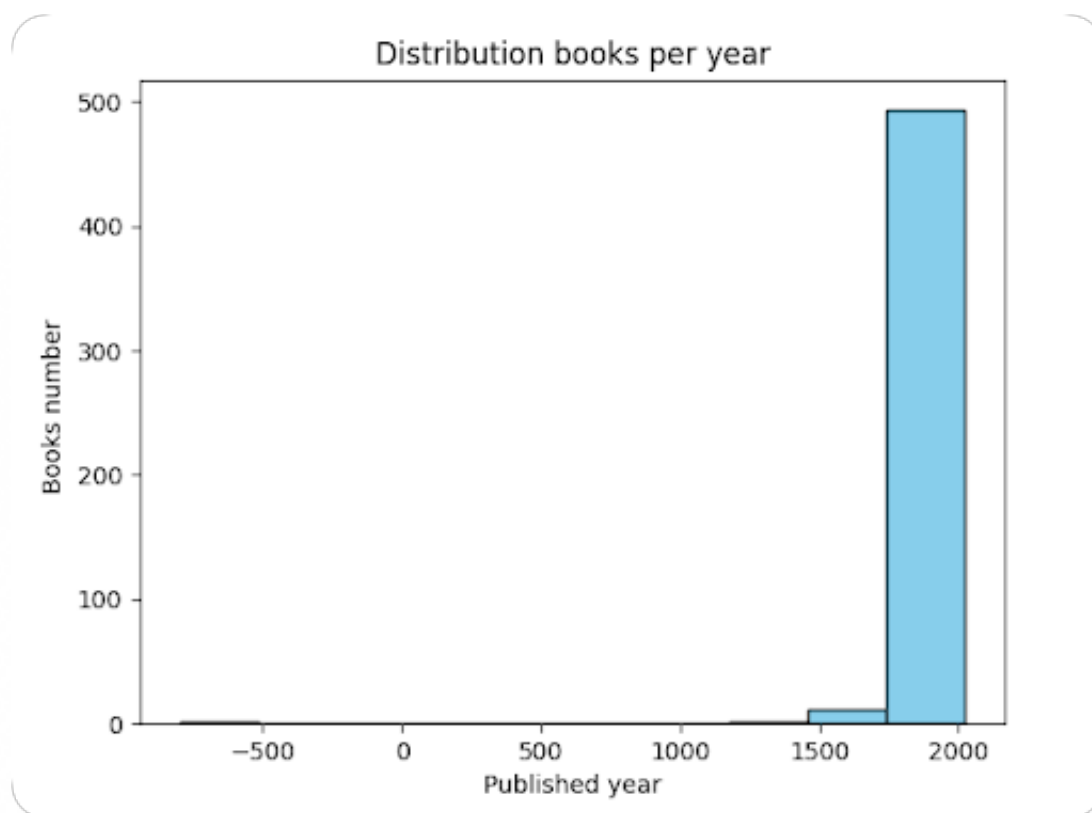
# Closing the database connection
conn.close()

# Creating a histogram to visualize the distribution of
books by publication year
plt.hist(df_books['published'], bins=10, color='skyblue',
edgecolor='black')
plt.xlabel("Published Year")
plt.ylabel("Number of Books")
plt.title("Distribution of Books by Year")
plt.show()
```

Cum funcționează?

- `plt.hist(df_books['published'], bins=10, color='skyblue', edgecolor='black')`
 - Creăm o histogramă pe baza coloanei **published**, care conține anii publicării cărților.
 - Argumentul **bins=10** împarte datele în 10 intervale, ceea ce ne oferă o reprezentare a împărțirii lor în diferite perioade de timp.
 - Folosim culoarea **skyblue** pentru o vizibilitate mai bună și margini de culoare **black** pentru separarea clară a barelor.
- `plt.xlabel("Year of Publication")` – marcajul pentru axa x afișează anii publicării.
- `plt.ylabel("Number of Books")` – marcajul axei y afișează numărul de cărți în fiecare interval.
- `plt.title("Distribution of Books by Year")` – titlul diagramei descrie clar ce afișează aceasta.

Rezultatul poate arăta astfel:



Imaginea 26.3. BooksPerYear

Ce obținem cu acest cod?

- Vedem în ce perioade de timp au fost publicate cele mai multe cărți.
- Putem detecta tendințele, de exemplu, în cazul în care cărțile mai vechi sunt mai rar reprezentate în baza de date sau dacă există o perioadă cu un număr mare de ediții.
- O diagramă ca aceasta poate fi utilă pentru a analiza creșterea publicării în diferite genuri sau epoci.

Sfat bonus: Dacă dorim o împărțire mai precisă a anilor, putem crește numărul **bins** (de exemplu, `bins=20`) pentru a obține o analiză mai fină.

Bar chart - Numărul de cărți în funcție de gen

Pentru a obține o reprezentare vizuală mai bună a distribuției cărților în funcție de gen, folosim un **bar chart (diagramă cu bare)**. Acest tip de diagramă este ideal pentru afișarea datelor ce țin de categorie, cum ar fi numărul de cărți pentru fiecare gen.

Codul pentru generarea unui bar chart:

```
import mysql.connector
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Creating a connection to the MySQL database
conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="root",
    database="library"
)

# SQL query to retrieve book data
query = "SELECT book_id, title, published, genre_id FROM Book"
df_books = pd.read_sql(query, conn)

# Closing the database connection
conn.close()

# Grouping data by genre and counting the number of books
df_grouped = df_books.groupby("genre_id")["book_id"].count()

# Data visualization
plt.figure(figsize=(10, 6))
sns.barplot(x=df_grouped.index, y=df_grouped.values,
```

```
palette="viridis")

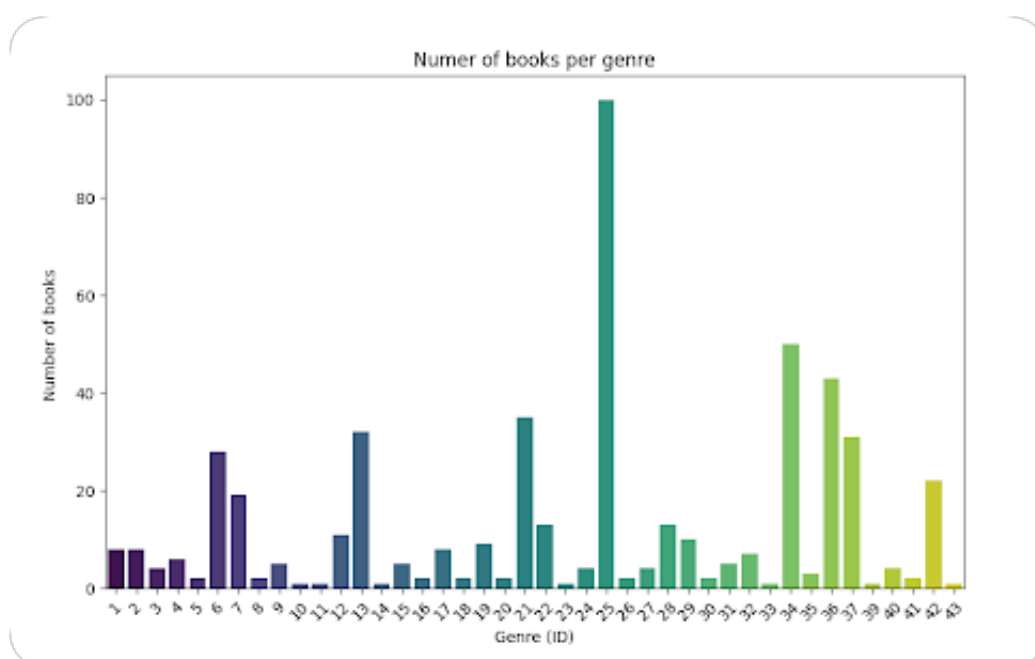
plt.xlabel("Genre (ID)")
plt.ylabel("Number of Books")
plt.title("Number of Books per Genre")
plt.xticks(rotation=45)&nbsp;

# Rotating labels if there are many
plt.show()
```

Cum funcționează?

- `plt.figure(figsize=(10, 6))` – am setat dimensiunea diagramei la 10x6 inch pentru a fi mai clară.
- `sns.barplot(x=df_grouped.index, y=df_grouped.values, palette="viridis")`
 - Creăm diagrama cu bare cu ajutorul bibliotecii **Seaborn**.
 - **Axa X** reprezintă **ID-ul genului**, iar **axa Y** reprezintă **numărul de cărți** din fiecare gen.
 - `palette="viridis"` adaugă o culoare de gradient frumoasă, pentru un efect vizual mai bun.
- `plt.xlabel("Genre (ID)")` și `plt.ylabel("Number of Books")` – setăm marcasele axelor pentru o interpretare mai bună a datelor.
- `plt.xticks(rotation=45)` – dacă există multe genuri, rotim marcasele pe axa X, pentru o lizibilitate mai bună.
- `plt.show()` – afișăm diagrama.

Rezultatul poate arăta astfel:



Imaginea 26.4. NumberBooksPerGenre

Ce obținem cu acest cod?

- Putem vedea foarte simplu care gen are cele mai multe cărți.
- Dacă unele genuri sunt semnificativ subreprezentate, acest lucru poate indica necesitatea extinderii bibliotecii în acele categorii.
- Vizualizarea ajută la o analiză mai bună a datelor decât doar simple reprezentări numerice.

Sfat bonus: Dacă dorim să adăugăm denumirile specifice ale genurilor în loc de **ID-uri**, putem conecta din timp datele cu tabelul genurilor pentru a avea o prezentare mai clară.

Concluzia lecției

În această lecție, Alex a învățat cum să conecteze **Python** la **baza de**

date MySQL și cum să folosească **Pandas** și **Matplotlib** pentru analiza și vizualizarea datelor. Aceste instrumente îi permit să exploreze în mod eficient datele, să identifice tendințele și să ia decizii informate.

Conceptele cheie pe care le-am învățat:

- Cum se utilizează **Pandas** pentru preluarea datelor dintr-o bază de date MySQL și pentru a le procesa.
- Cum se analizează datele utilizând `describe()`, `groupby()`, `unique()` și alte funcții.
- Cum se creează **diagrame** folosind **Matplotlib** și **Seaborn** pentru vizualizarea datelor.

De ce este important acest lucru?

SQL îi permite lui Alex să **colecteze** date, în timp ce Python cu Pandas și Matplotlib îi permite să le **analizeze** și să le **reprezinte** într-un mod intuitiv. Aceste abilități sunt esențiale pentru analiza avansată a datelor și luarea deciziilor bazate pe date.

Următorul pas: Testarea cunoștințelor printr-o sarcină practică

Felicitări pentru finalizarea cu succes a modulului! Acum este timpul să testăm tot ce am învățat printr-o **sarcină practică**, care simulează un scenariu de analiză a datelor din lumea reală.

Provocarea cu care se confruntă Alex

Alex trebuie acum să analizeze datele din baza de date **movies_db** folosind **Python** și biblioteca **Pandas**. Prin această sarcină, vom testa abilitatea noastră de a:

- stabili o conexiune la baza de date MySQL din Python;

- executa **interogări SQL** pentru a prelua date relevante;
- folosi **Pandas** pentru prelucrarea și analiza datelor;
- crea **vizualizări** de date folosind **Matplotlib**.

Ce vom face?

Vom analiza baza de date de filme **movies_db** în așa fel încât:

- extragem informații cheie despre filme, actori și evaluări;
- efectuăm agregări și analize statistice ale datelor;
- afișăm tendințele prin diagrame (de exemplu, numărul de filme în funcție de gen, evaluările medii pe an).

Sfaturi pentru succes:

- **Ne conectăm corect** la baza de date MySQL folosind `mysql.connector`.
- **Planificăm interogările SQL** înainte de a le scrie – luăm în considerare de ce date avem nevoie.
- **Folosim Pandas** pentru a filtra, grupa și analiza cu ușurință datele.
- **Vizualizăm datele** folosind **Matplotlib** și **Seaborn** pentru a le face mai ușor de interpretat.
- **Testăm codul** parte cu parte – este mai bine să facem iterații mici decât să corectăm erorile mari.

Sunteți pregătiți? Aceasta este oportunitatea noastră de a demonstra cât de mult am stăpânit analiza datelor și implementarea interogărilor SQL în mediul Python! **Să trecem la provocare!**