Introduction

The coronavirus took the entire world by surprise, changing everyone's daily routine. City dwellers no longer spent their free time outside, going to cafes and malls; more people were at home, reading books. That attracted the attention of startups that rushed to develop new apps for book lovers.

We've been given a database of one of the services competing in this market. It contains data on books, publishers, authors, and customer ratings and reviews of books. This information will be used to generate a value proposition for a new product.

1 Loading Libraries and Database

Lets start by loading the necessary libraries and continue by connected to the given database:

In [1]:

```
import pandas as pd
from sqlalchemy import create_engine


db_config = {'user': 'praktikum_student',         # user name
             'pwd': 'Sdf4$2;d-d30pp', # password
             'host': 'rc1b-wcoijxj3yxfsf3fs.mdb.yandexcloud.net',
             'port': 6432,              # connection port
             'db': 'data-analyst-final-project-db'}         # the name of the data base

connection_string = 'postgresql://{}:{}@{}:{}/{}'.format(db_config['user'],
                                                             db_config['pwd'],
                                                               db_config['host'],
                                                               db_config['port'],
                                                               db_config['db'])

engine = create_engine(connection_string, connect_args={'sslmode':'require'})
```

2 Studying The Data

Lets have a look how does each of the tables look like, starting with the ***books*** table:

In [2]:

```
query = '''
    SELECT * FROM books
'''

allbooks = pd.io.sql.read_sql(query, con = engine) # reading everything from books table
display(allbooks.head())
allbooks.info()

if pd.io.sql.read_sql('SELECT book_id FROM books GROUP BY book_id', con = engine).shape[0]
    print('\nNo duplicates!')
```

	book_id	author_id	title	num_pages	publication_date	publisher_id
0	1	546	'Salem's Lot	594	2005-11-01	93
1	2	465	1 000 Places to See Before You Die	992	2003-05-22	336
2	3	407	13 Little Blue Envelopes (Little Blue Envelope...	322	2010-12-21	135
3	4	82	1491: New Revelations of the Americas Before C...	541	2006-10-10	309
4	5	125	1776	386	2006-07-04	268

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 6 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   book_id           1000 non-null   int64 
 1   author_id         1000 non-null   int64 
 2   title             1000 non-null   object
 3   num_pages         1000 non-null   int64 
 4   publication_date  1000 non-null   object
 5   publisher_id      1000 non-null   int64 
dtypes: int64(4), object(2)
memory usage: 47.0+ KB

No duplicates!
```

Looks great, we have 1000 distinct ***book_id***s and 1000 rows - no dups! **Next** lets look at **authors** table:

In [3]:

```
query = '''
    SELECT * FROM authors
'''

allauthors = pd.io.sql.read_sql(query, con = engine) # reading everything from authors tabl
display(allauthors.head())
allauthors.info()


if pd.io.sql.read_sql('SELECT author_id FROM authors GROUP BY author_id', con = engine).sha
    print('\nNo duplicates!')
```

	author_id	author
0	1	A.S. Byatt
1	2	Aesop/Laura Harris/Laura Gibbs
2	3	Agatha Christie
3	4	Alan Brennert
4	5	Alan Moore/David Lloyd

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 636 entries, 0 to 635
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   author_id  636 non-null    int64 
 1   author     636 non-null    object
dtypes: int64(1), object(1)
memory usage: 10.1+ KB

No duplicates!
```

Nothing wrong here, we got all the information we need about the authors. **Now**, lets check the last two tables: ***reviews*** and ***ratings***:

In [4]:

```
query = '''
    SELECT * FROM reviews
'''

allrev = pd.io.sql.read_sql(query, con = engine) # reading everything from reviews table
display(allrev.head())
allrev.info()

if pd.io.sql.read_sql('SELECT review_id FROM reviews GROUP BY review_id', con = engine).sha
    print('\nNo duplicates!')
```

	review_id	book_id	username	text
0	1	1	brandtandrea	Mention society tell send professor analysis. ...
1	2	1	ryanfranco	Foot glass pretty audience hit themselves. Amo...
2	3	2	lorichen	Listen treat keep worry. Miss husband tax but ...
3	4	3	johnsonamanda	Finally month interesting blue could nature cu...
4	5	3	scotttamara	Nation purpose heavy give wait song will. List...

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2793 entries, 0 to 2792
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   review_id  2793 non-null   int64 
 1   book_id    2793 non-null   int64 
 2   username   2793 non-null   object
 3   text       2793 non-null   object
dtypes: int64(2), object(2)
memory usage: 87.4+ KB

No duplicates!
```

We have actual reviews! We know who made those reviews, how much there are and for what book they belong, this is good.

Lets check ratings:

In [5]:

```
query = '''
    SELECT * FROM ratings
'''

allrates = pd.io.sql.read_sql(query, con = engine) # reading everything from ratings table
display(allrates.head())
allrates.info()

if pd.io.sql.read_sql('SELECT rating_id FROM ratings GROUP BY rating_id', con = engine).sha
    print('\nNo duplicates!')
```

	rating_id	book_id	username	rating
0	1	1	ryanfranco	4
1	2	1	grantpatricia	2
2	3	1	brandtandrea	5
3	4	2	lorichen	3
4	5	2	mariokeller	2

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6456 entries, 0 to 6455
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   rating_id  6456 non-null   int64 
 1   book_id    6456 non-null   int64 
 2   username   6456 non-null   object
 3   rating     6456 non-null   int64 
dtypes: int64(3), object(1)
memory usage: 201.9+ KB

No duplicates!
```

Now that we have looked at our tables, we know what we are up to and we know we don't have any problematic data, we can start analyzing them.

3 Data Analysis

In order to estimate how many books were made per year, lets calculate the number of books published after the Jan 1st, 2000:

In [6]:

```
query = '''
    SELECT COUNT(title) as book_count
    FROM books
    WHERE publication_date > '2000-01-01'
'''

books_after_2000 = pd.io.sql.read_sql(query, con = engine)
books_after_2000
```

Out[6]:

	book_count
0	819

636 books - that's impressive. The last date recorded was 2020, meaning 20 years of publishing books - an average of 32 book per year.

Next: lets check how many reviews did each book get and what was the average rating per book.

In [25]:

```
query_1 = '''
    SELECT re.book_id, title, re.review_count, ra.avg_rating FROM books
    LEFT JOIN (SELECT book_id, COUNT(text) AS review_count FROM reviews GROUP BY book_id) A
    LEFT JOIN (SELECT book_id, AVG(rating) AS avg_rating  FROM ratings GROUP BY book_id) AS
'''

df = pd.io.sql.read_sql(query_1, con = engine)
df.head(5)
```

Out[25]:

	book_id	title	review_count	avg_rating
0	1.0	'Salem's Lot	2.0	3.666667
1	2.0	1 000 Places to See Before You Die	1.0	2.500000
2	3.0	13 Little Blue Envelopes (Little Blue Envelope...	3.0	4.666667
3	4.0	1491: New Revelations of the Americas Before C...	2.0	4.500000
4	5.0	1776	4.0	4.000000

In [19]:

```
#reviewrs code
pd.io.sql.read_sql("""SELECT b.title, q1.reviews_count, q2.average_rating FROM books b
LEFT JOIN (SELECT book_id, count(text) as reviews_count FROM reviews GROUP BY book_id) as q
LEFT JOIN (SELECT book_id, avg(rating) as average_rating FROM ratings GROUP BY book_id) as
""", con = engine)
```

Out[19]:

	title	reviews_count	average_rating
0	The Body in the Library (Miss Marple #3)	2.0	4.500000
1	Galápagos	2.0	4.500000
2	A Tree Grows in Brooklyn	5.0	4.250000
3	Undaunted Courage: The Pioneering First Missio...	2.0	4.000000
4	The Prophet	4.0	4.285714
...	...	...	...
995	Alice in Wonderland	4.0	4.230769
996	A Woman of Substance (Emma Harte Saga #1)	2.0	5.000000
997	Christine	3.0	3.428571
998	The Magicians' Guild (Black Magician Trilogy #1)	2.0	3.500000
999	The Plot Against America	2.0	3.000000

1000 rows × 3 columns

Alot of books actually have a full rating of 5.0 - not so much reviews on the other hand.

Lets take a look at publishers with greatest amount of books - ONLY BOOKS WITH 50 PAGES OR MORE INCLUDED.

In [8]:

```
query = '''
    SELECT
        p.publisher,
        p.publisher_id,
        COUNT(b.book_id) AS count
    FROM publishers AS p
    LEFT JOIN books AS b ON b.publisher_id = p.publisher_id
    WHERE b.num_pages > 50
    GROUP BY p.publisher_id
    ORDER BY count DESC
    LIMIT 10;
'''

greatest_pubs = pd.io.sql.read_sql(query, con = engine)
greatest_pubs
```

Out[8]:

	publisher	publisher_id	count
0	Penguin Books	212	42
1	Vintage	309	31
2	Grand Central Publishing	116	25
3	Penguin Classics	217	24
4	Ballantine Books	33	19
5	Bantam	35	19
6	Berkley	45	17
7	St. Martin's Press	284	14
8	Berkley Books	46	14
9	Delta	83	13

Above we can see the list of top publishers, with ***Penguin Books*** at the top holding 42 book publishes.

What about authors, which authors have the highest average rating? - ONLY BOOKS WITH 50 OR MORE RATINGS WILL BE INCLUDED.

In [9]:

```
query = '''
    SELECT
        a.author,
        br.ratings
    FROM (SELECT
        b.book_id,
        b.author_id,
        b.title,
        COUNT(ra.rating),
        AVG(ra.rating) as ratings
        FROM books AS b
        LEFT JOIN ratings AS ra ON ra.book_id = b.book_id
        GROUP BY b.author_id, b.book_id
        HAVING COUNT(ra.rating) > 49
        ORDER BY ratings DESC) AS br
    LEFT JOIN authors AS a ON a.author_id = br.author_id
    LIMIT 1;
'''

pd.io.sql.read_sql(query, con = engine)
```

Out[9]:

	author	ratings
0	J.K. Rowling/Mary GrandPré	4.414634

J.K. Rowling/Mary GrandPré are the author of the book with the highest average rating.

Next we'll look at the average number of reviews per user - ONLY INCLUDING USERS WITH AT LEAST 50 BOOK RATINGS.

In [10]:

```
query = '''
    SELECT AVG(final.review_count)
    FROM (SELECT
            re.username,
            tr.rating_count,
            COUNT(re.review_id) AS review_count
        FROM (SELECT
            username,
            COUNT(rating) as rating_count
            FROM ratings
            GROUP BY username
            HAVING COUNT(rating) > 50) AS tr
        LEFT JOIN reviews AS re ON re.username = tr.username
        GROUP BY re.username, tr.rating_count) AS final
'''

pd.io.sql.read_sql(query, con = engine)
```

Out[10]:

	avg
0	24.333333

Looks like top reviewers had around 24 reviews in average, less than half of the ratings they gave that's for sure!

4 Summary ¶

After finishing all the necessary studying, we've come up with:

1. 636 books were published after the 1st of January, 2000.
2. ***Penguin Books*** is the top publisher, holding 42 book publishes.
3. J.K. Rowling/Mary GrandPré are the authors of the book with the highest average rating.
4. The average number of text reviews among users who rated more than 50 books is 24
5. This is a list of all books with the number of reviews and their average rating: