

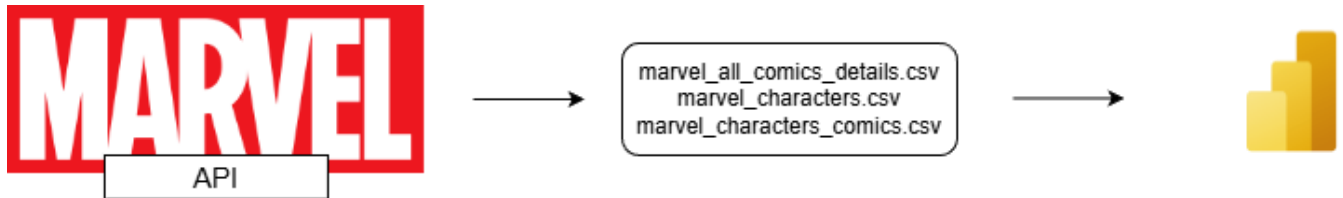
Marvel Project

Table of Contents

1. Introduction.....	2
2. Pre-requisites.....	2
3. Project Development	2
a. Marvel Comics API.....	2
i. Created a developer's account on Marvel's site and obtained Public and Private Keys - https://developer.marvel.com/	2
ii. Created "key.py" file with the public and private keys	3
iii. Extracted data about all characters.....	3
iv. Extracted data about all books	4
v. Extracted data about all characters and all books they appear into.....	5
b. Power BI Desktop	7
i. Loading the data	7
ii. Transforming the data – generating 3 queries for the 3 separate files	7
iii. Data model	9
iv. DAX Measures	9
v. Visuals	10

1. Introduction

The purpose of this project is to deliver information about Marvel characters and books extracted with the use of Marvel's API, stored in CSV files and visualized in Power BI and show the use of Marvel API's functionality and the two new visuals on the Microsoft introduced with the recent versions of Power BI- Card (new) and Slicer (new).



2. Pre-requisites

- Marvel account at <https://developer.marvel.com> with public and private keys
- Python and the used libraries (see code sections)
- Visual Studio Code or similar
- Power BI Desktop

3. Project Development

a. Marvel Comics API

- Created a developer's account on Marvel's site and obtained Public and Private Keys - <https://developer.marvel.com/>

MY DEVELOPER ACCOUNT

Hi 6hsQUd161668313!
Here's your personal Marvel Comics API information:

Your public key

[Redacted public key]

Your private key

[Redacted private key]

Read more about how to use your keys to sign requests. »

Your rate
limit:

3000
calls/day

Number of calls your
application can make per day.

Your authorized referrers

List any domains that can make calls to the Marvel Comics API
using your API key here:

[delete](#)

ii. Created "key.py" file with the public and private keys

```
PUBLIC_KEY="Write_Public_Key"
PRIVATE_KEY= "Write_Private_Key"
```

iii. Extracted data about all characters

This Python code uses the Marvel Comics API to retrieve information about Marvel characters in paginated requests, with each page containing up to 100 characters. It generates a timestamp and hash for API authentication, makes requests to the Marvel Characters API, and writes relevant information about each character (ID, Name, Description, Thumbnail URL) to a CSV file named "marvel_characters.csv." The script retries in case of a failed request.

```
import time
import csv
from key import PUBLIC_KEY, PRIVATE_KEY
from hashlib import md5
import requests
import pandas as pd

CHARACTER_URL = 'http://gateway.marvel.com/v1/public/characters'
characters_list = []

# Function to generate timestamp and hash parameters required for API authentication
def get_hash_and_ts_params():
    ts = str(time.time())
    combined = ''.join([ts, PRIVATE_KEY, PUBLIC_KEY])
    hash_value = md5(combined.encode('ascii')).hexdigest()
    return {'ts': ts, 'hash': hash_value}

# Function to make paged requests to the Marvel Characters API and save results to a CSV file
def paged_requests(page_size=100):
    params = {'apikey': PUBLIC_KEY, 'limit': page_size}
    csv_file_name = "marvel_characters.csv"

    with open(csv_file_name, mode="w", newline="", encoding="utf-8") as csv_file:
        csv_writer = csv.writer(csv_file, delimiter="%")
        csv_writer.writerow(["ID", "Name", "Description", "Thumbnail URL"])

        for i in range(16):
            hash_params = get_hash_and_ts_params()
            params.update(hash_params)
            params.update({'offset': page_size * i})

            while True:
                try:
                    resp = requests.get(CHARACTER_URL, params)
                    resp.raise_for_status()
                    j = resp.json()
```

```

        for char in j['data']['results'][:100]:
            # Extract relevant information about each character
            char_id = char["id"]
            char_name = char["name"]
            char_description = char["description"]
            thumbnail_path = char["thumbnail"]["path"]
            thumbnail_extension = char["thumbnail"]["extension"]
            thumbnail_url = f"{thumbnail_path}.{thumbnail_extension}"

            # Write a row to the CSV file for each character
            csv_writer.writerow([char_id, char_name,
                                char_description, thumbnail_url])

        break
    except requests.exceptions.RequestException as e:
        print(f"Request failed: {e}. Retrying...")
        time.sleep(10)

if __name__ == '__main__':
    paged_requests()
    print('Done')

```

iv. Extracted data about all books

This Python code utilizes the Marvel Comics API to retrieve information about Marvel comics in paginated requests, with each page containing up to 100 comics. Similar to the previous code, it generates a timestamp and hash for API authentication, makes requests to the Marvel Comics API, and writes relevant information about each comic (Comic ID, Title, Format, Variant Description, Description, Thumbnail URL, Price, Price Type) to a CSV file named "marvel_all_comics_details.csv." The script iterates through multiple pages of comic results (584 pages in total).

```

import time
import csv
from key import PUBLIC_KEY, PRIVATE_KEY
from hashlib import md5
import pandas as pd
import requests

COMIC_URL = 'http://gateway.marvel.com/v1/public/comics'

# Function to generate timestamp and hash parameters required for API authentication
def get_hash_and_ts_params():
    ts = str(time.time())
    combined = ''.join([ts, PRIVATE_KEY, PUBLIC_KEY])
    hash_value = md5(combined.encode('ascii')).hexdigest()
    return {'ts': ts, 'hash': hash_value}

# Function to make paged requests to the Marvel Comics API and save results to a CSV file
def paged_requests(page_size=100):
    params = {'apikey': PUBLIC_KEY, 'limit': page_size}
    csv_file_name = "marvel_all_comics_details.csv"

```

```

# Open the CSV file in write mode
with open(csv_file_name, mode="w", newline="", encoding="utf-8") as csv_file:
    csv_writer = csv.writer(csv_file, delimiter="%")

    # Write header row to the CSV file
    csv_writer.writerow(["Comic ID", "Title", "Format", "Variant Description",
"Description", "Thumbnail URL", "Price 1", "Price Type 1"])

    for i in range(584):
        hash_params = get_hash_and_ts_params()
        params.update(hash_params)
        params.update({'offset': page_size * i})
        resp_comics = requests.get(COMIC_URL, params)
        resp_comics.raise_for_status()
        j_comics = resp_comics.json()

        for comic in j_comics['data']['results'][:100]:
            comic_id = comic["id"]
            comic_name = comic["title"]
            comic_format = comic["format"]
            variant_description = comic["variantDescription"]
            description = comic['description']
            thumbnail_path = comic["thumbnail"]["path"]
            thumbnail_extension = comic["thumbnail"]["extension"]
            thumbnail_url = f"{thumbnail_path}.{thumbnail_extension}"
            price_1 = comic["prices"][0]["price"]
            price_type_1 = comic["prices"][0]["type"]

            # Write a row to the CSV file for each comic
            csv_writer.writerow([comic_id, comic_name, comic_format,
variant_description, description, thumbnail_url, price_1, price_type_1])

if __name__ == '__main__':
    paged_requests()
    print('Done')

```

v. Extracted data about all characters and all books they appear into

This Python script systematically explores Marvel characters and their associated comics by making paged requests to the Marvel API. It retrieves comic IDs for each character and writes the pairs of character IDs and comic IDs to a CSV file named "marvel_characters_comics.csv." The script incorporates error handling and retries in case of a failed request.

```

import time
import csv
from key import PUBLIC_KEY, PRIVATE_KEY
from hashlib import md5
import requests

CHARACTER_URL = 'http://gateway.marvel.com/v1/public/characters'
COMIC_LIMIT = 100

```

```

# Function to generate timestamp and hash parameters required for API authentication
def get_hash_and_ts_params():
    ts = str(time.time())
    combined = ''.join([ts, PRIVATE_KEY, PUBLIC_KEY])
    hash_value = md5(combined.encode('ascii')).hexdigest()
    return {'ts': ts, 'hash': hash_value}

# Function to retrieve comic IDs for a specific character
def get_comic_ids(character_id, offset=0):
    COMIC_URL = f'http://gateway.marvel.com/v1/public/characters/{character_id}/comics'
    hash_params = get_hash_and_ts_params()
    params = {'apikey': PUBLIC_KEY, 'offset': offset, 'limit': COMIC_LIMIT, **hash_params}

    retries = 3
    for attempt in range(retries):
        try:
            resp_comics = requests.get(COMIC_URL, params=params)
            resp_comics.raise_for_status()
            comics = resp_comics.json()['data']['results']
            comic_ids = [comic["id"] for comic in comics]
            return comic_ids
        except requests.exceptions.RequestException as e:
            print(f"Error in request: {e}")
            if attempt < retries - 1:
                print(f"Retrying (attempt {attempt + 1} of {retries})...")
                time.sleep(2 ** attempt)
            else:
                print("Max retries reached. Exiting.")
                raise

# Function to write a CSV row for a character and comic ID
def write_csv_row(csv_writer, character_id, comic_id):
    csv_writer.writerow([character_id, comic_id])

# Function to process a character and retrieve their comics
def process_character(character, csv_writer, i):
    character_id = character["id"]
    character_name = character["name"]
    offset = 0

    while True:
        comic_ids = get_comic_ids(character_id, offset=offset)

        for comic_id in comic_ids:
            print(i, character_id, character_name, comic_id)
            write_csv_row(csv_writer, character_id, comic_id)

        if len(comic_ids) < COMIC_LIMIT:
            break
        offset += COMIC_LIMIT

# Function to make paged requests to the Marvel Characters API and retrieve comics for
each character

```

```

def paged_requests(page_size=100):
    params = {'apikey': PUBLIC_KEY, 'limit': page_size}
    csv_file_name = "marvel_characters_comics.csv"

    with open(csv_file_name, mode="w", newline="", encoding="utf-8") as csv_file:
        csv_writer = csv.writer(csv_file, delimiter=",")
        csv_writer.writerow(["Character ID", "Comic ID"])

    for i in range(1, 17):
        hash_params = get_hash_and_ts_params()
        params.update(hash_params)
        params.update({'offset': page_size * i})

        retries = 3
        for attempt in range(retries):
            try:
                resp_characters = requests.get(CHARACTER_URL, params=params)
                resp_characters.raise_for_status()




                for character in resp_characters.json()['data']['results']:
                    process_character(character, csv_writer, i)

            except requests.exceptions.RequestException as e:
                break
            except requests.exceptions.RequestException as e:
                print(f"Error in request: {e}")
                if attempt < retries - 1:
                    print(f"Retrying (attempt {attempt + 1} of {retries})...")
                    time.sleep(60 ** attempt)
                else:
                    print("Max retries reached. Skipping to the next iteration.")
                    break

if __name__ == '__main__':
    paged_requests()
    print('Done')

```

The files are stored in the same folder where I have the python files.

 marvel_all_comics_details	29/11/2023 14:59	Microsoft Excel Com...	17,656 KB
 marvel_characters	02/12/2023 22:21	Microsoft Excel Com...	197 KB
 marvel_characters_comics	04/12/2023 19:03	Microsoft Excel Com...	1,345 KB

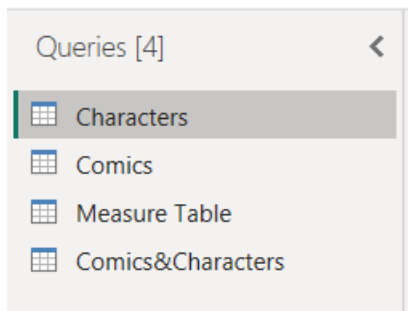
b. Power BI Desktop

i. Loading the data

1. Loading the CSV files using Get Data -> Text/CSV

ii. Transforming the data – generating 3 queries for the 3 separate files

Queries



Characters

let

```
Source = Csv.Document(File.Contents("C:\Users\*****\Documents\Projects and Courses\Python\Marvel API call\Power BI\marvel_characters.csv"),[Delimiter=";", Columns=4, Encoding=1252, QuoteStyle=QuoteStyle.Csv]),
```

```
#"Promoted Headers" = Table.PromoteHeaders(Source, [PromoteAllScalars=true]),
```

```
#"Changed Type" = Table.TransformColumnTypes(#"Promoted Headers",{{"ID", Int64.Type}}),
```

```
#"Removed Duplicates" = Table.Distinct(#"Changed Type", {"ID"})
```

in

```
#"Removed Duplicates"
```

Comics

let

```
Source = Csv.Document(File.Contents("C:\Users\*****\Documents\Projects and Courses\Python\Marvel API call\Power BI\marvel_all_comics_details.csv"),[Delimiter=";", Columns=8, Encoding=65001, QuoteStyle=QuoteStyle.Csv]),
```

```
#"Promoted Headers" = Table.PromoteHeaders(Source, [PromoteAllScalars=true]),
```

```
#"Changed Type" = Table.TransformColumnTypes(#"Promoted Headers",{{"Comic ID", Int64.Type}, {"Title", type text}, {"Format", type text}, {"Variant Description", type text}, {"Description", type text}, {"Thumbnail URL", type text}, {"Price 1", type number}, {"Price Type 1", type text}}),
```

```
#"Trimmed Text" = Table.TransformColumns(#"Changed Type",{{"Title", Text.Trim, type text}}),
```

```
#"Sorted Rows" = Table.Sort(#"Trimmed Text",{{"Title", Order.Ascending}}),
```

```
#"Removed Duplicates" = Table.Distinct(#"Sorted Rows", {"Comic ID"}),
```

```
#"Replaced Value" = Table.ReplaceValue(#"Removed Duplicates", "", "Unknown", Replacer.ReplaceValue, {"Format"}),
```

```
#"Capitalized Each Word" = Table.TransformColumns(#"Replaced Value",{{"Format", Text.Proper, type text}})
```

in

```
#"Capitalized Each Word"
```

Comics&Characters

let

```
Source = Csv.Document(File.Contents("C:\Users\*****\Documents\Projects and Courses\Python\Marvel API call\Power BI\marvel_characters_comics.csv"),[Delimiter=";", Columns=2, Encoding=1252, QuoteStyle=QuoteStyle.None]),
```

```
#"Promoted Headers" = Table.PromoteHeaders(Source, [PromoteAllScalars=true]),
```

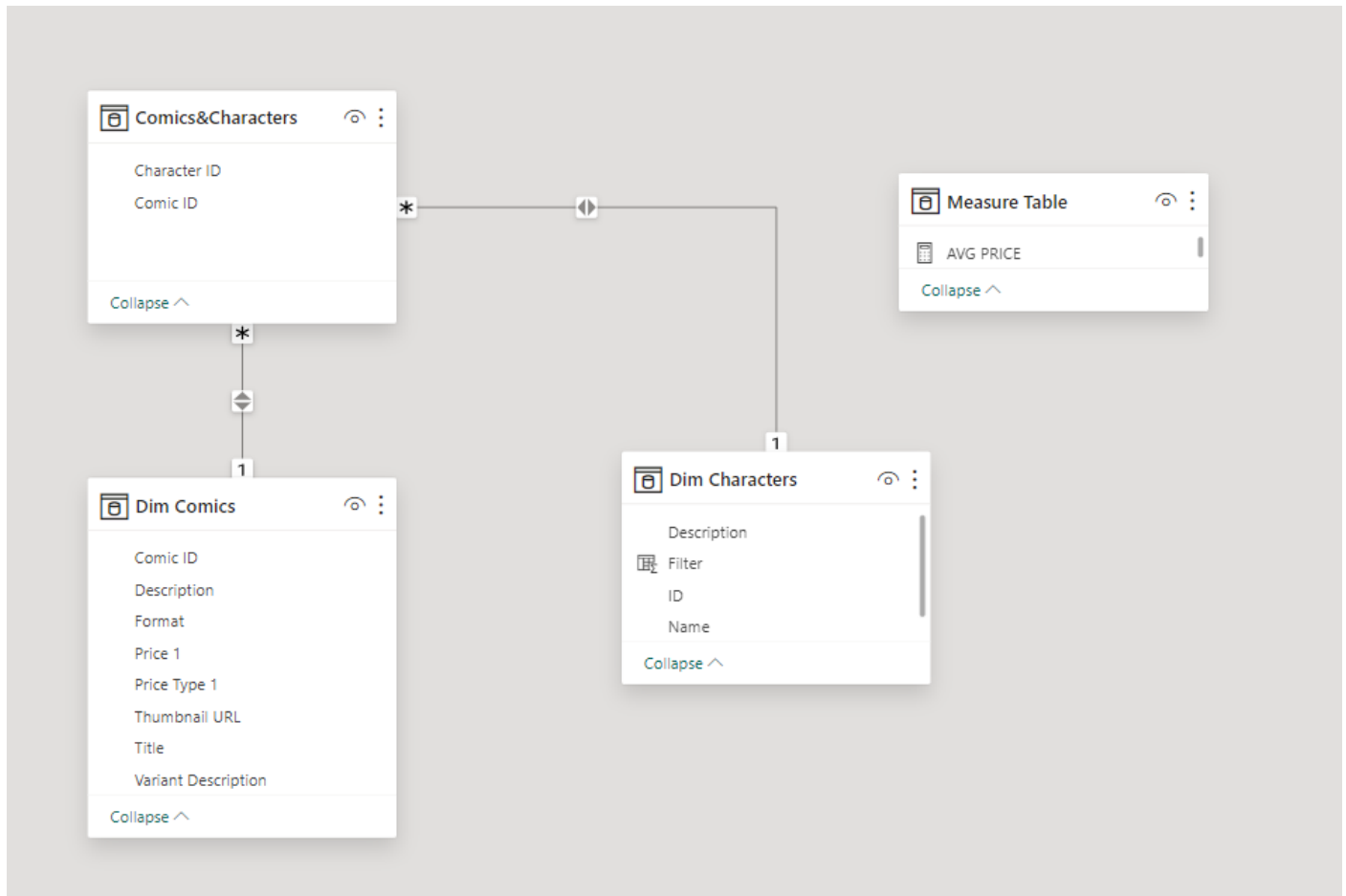
```
#"Changed Type" = Table.TransformColumnTypes(#"Promoted Headers",{{"Comic ID", Int64.Type}, {"Character ID", type text}})
```


in

#"Changed Type"

Measure Table – Contains all measures

iii. Data model



iv. DAX Measures

```
AVG PRICE =  
CALCULATE(  
    AVERAGE('Comics'[Price 1]),  
    'Comics'[Price 1] <> 0  
)
```

```
AVG Price Message =  
IF(  
    [AVG PRICE] = BLANK(),  
    "No price available",  
    "$" & ROUND([AVG PRICE],2)  
)
```

```
Count Comics = COUNT('Comics'[Comic ID])
```

```
Description Book Message =  
VAR _description = SELECTEDVALUE('Comics'[Description])  
RETURN  
IF(  
    _description = BLANK(),
```

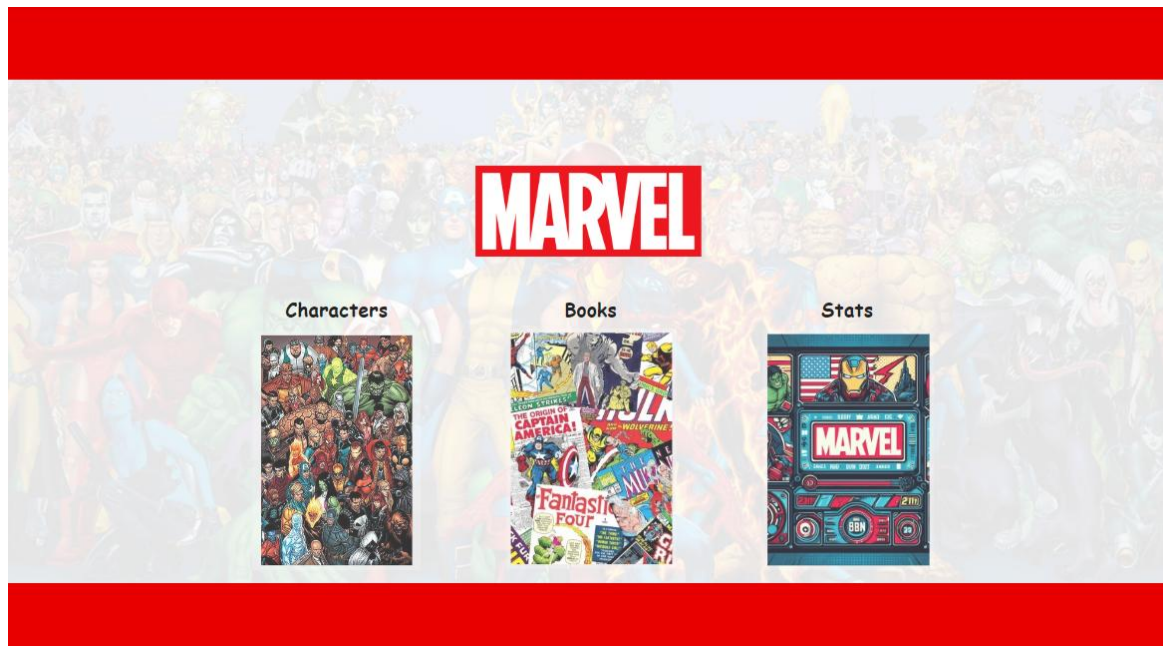
```

    "No description available for this edition",
    IF(
        LEN(_description) > 400,
        CONCATENATE(MID(_description,1,400), "..."),
        _description
    )
)
)
Decription Book Message Full =
VAR _description = SELECTEDVALUE('Comics'[Description])
RETURN
IF(
    _description = BLANK(),
    "No description available for this edition",
    _description
)
)
Decription Character Message =
VAR _description = SELECTEDVALUE('Characters'[Description])
RETURN
IF(
    _description = BLANK(),
    "No description available for this character",
    _description
)
)
Price Total = SUM('Comics'[Price 1])
See full description label = "See full description"
Total Number of Appearances = COUNT('Comics&Characters'[Character ID])

```

v. Visuals

Cover page



The Cover page has 3 buttons that navigate the user to one of the 3 desired main pages with Characters, Books or Stats.

Select Your Character	Average Price	Appearances
<input type="text" value="Search"/> <ul style="list-style-type: none"> <input type="radio"/> 3-D Man <input type="radio"/> A.I.M. <input type="radio"/> A-Bomb (HAS) <input type="radio"/> Abomination (Emil Blonsky) <input type="radio"/> Absorbing Man <input type="radio"/> Abyss <input type="radio"/> Abyss (Age of Apocalypse) <input type="radio"/> Adam Warlock <input type="radio"/> Aegis (Trey Rollins) <input type="radio"/> Agatha Harkness <input type="radio"/> Agent Brand <input type="radio"/> Agent X (Nijo) <input type="radio"/> Agent Zero <input type="radio"/> Agents of Atlas <input type="radio"/> Ajak <input type="radio"/> Ajaxis <input type="radio"/> Alex Power <input type="radio"/> Alex Wilder <input type="radio"/> Alice <input type="radio"/> Alicia Masters 	<p>\$6.97</p>	<p>426</p>

Teenager Miles Morales grew up in Brooklyn, New York. Recently, Miles took on the controversial and pressured role of Spider-Man shortly after the death of the original. Morales made his debut against The Kangaroo, much to the surprise and disapproval of many present at the confrontation.

If one selects one of the books, additional button appears which allows the user to go to a separate page with more details about that particular book.

Spider-Man (Miles Morales)

MARVEL

Select Your Character

- ☐ 3-D Man
- ☐ A.I.M.
- ☐ A-Bomb (HAS)
- ☐ Abomination (Emil Blonsky)
- ☐ Absorbing Man
- ☐ Abyss
- ☐ Abyss (Age of Apocalypse)
- ☐ Adam Warlock
- ☐ Aegis (Trey Rollins)
- ☐ Agatha Harkness
- ☐ Agent Brand
- ☐ Agent X (Nijo)
- ☐ Agent Zero
- ☐ Agents of Atlas
- ☐ Ajak
- ☐ Ajaxis
- ☐ Alex Power
- ☐ Alex Wilder
- ☐ Alice
- ☐ Alicia Masters

Average Price

\$6.97

Wanna see some more details about this book? Click here

Teenager Miles Morales grew up in Brooklyn, New York. Recently, Miles took on the controversial and pressured role of Spider-Man shortly after the death of the original. Morales made his debut against The Kangaroo, much to the surprise and disapproval of many present at the confrontation.

The Amazing Spider-Man (2018) #54.1

MARVEL

Price \$

3.99

LAST REMAINS TIE-IN! The Order of the Web faces Sin-Eater. And if you saw what Sin-Eater did to Spider-Man in ASM #49, you know just how much trouble the other Spiders are in! Also, what is Mary Jane Watson's part in "Last Remains"? Don't miss this chapter that raises her stakes by about a million times!

See full description

Spider-Man (Miles Morales)

Books page

Similar to the first page, here one can find a slicer with all the books on the left-hand side. In the center of the book, there is a description field and the price of the comic book. If the description is too long, one can use the "See full description" label to uncover all details for the selected book.

At the center bottom the user can find the full list of characters that are tagged and appear in the book. Right hand side provides the full cover page of the book.

A+X (2012) #4

MARVEL

Select Your Book

5 Ronin (2010) #4 (MAC)

A.X.E.: Judgment Day...

A.X.E.: Judgment Day...

A+X (2012) #14

A+X (2012) #4

A+X (2012) #4 (Brooks...)

Price \$

3.99

Storm and Black Panther's first post-AVX encounter! Your other two favorite characters fight your favorite villain.

See full description

Stats

On this page the user can find a vertical bar chart positioned on the left which visualizes all characters and the number of their appearances.

In the center top there one can find the total number of Characters, Books, the Average price and the Number of appearances.

On the Top right corner there is a donut chart with number of books split by their format.

The bottom of the page is dedicated for a table with all Book Titles, their Price, Format, Price Type and Variant Description.

