# SQL TEST

**1) What are the 10 most expensive products in the company?**

**Answer:** Using the query "SELECT * FROM looqbox_challenge.data_product ORDER BY PRODUCT_VAL DESC limit 10"

In [1]:
```
import mysql.connector
import pandas as pd
import matplotlib.pyplot as plt
from scipy import stats

mydb = mysql.connector.connect(host = '35.199.127.241',
                               user = 'looqbox-challenge',
                               passwd  = 'looq-challenge')
cursor = mydb.cursor()
```

In [2]:
```
resposta1 = pd.read_sql('''SELECT PRODUCT_NAME, PRODUCT_VAL FROM looqbox_challenge.data_product
ORDER BY PRODUCT_VAL DESC limit 10''', mydb)
display(resposta1)
```

| | PRODUCT_NAME | PRODUCT_VAL |
|---|---|---|
| **0** | Whisky Escoces THE MACALLAN Ruby Garrafa 700ml... | 741.99 |
| **1** | Whisky Escoces JOHNNIE WALKER Blue Label Garra... | 735.90 |
| **2** | Cafeteira Expresso 3 CORACOES Tres Modo Vermelho | 499.00 |
| **3** | Vinho Portugues Tinto Vintage QUINTA DO CRASTO... | 445.90 |
| **4** | Escova Dental Eletrica ORAL B D34 Professional... | 399.90 |
| **5** | Champagne Rose VEUVE CLICQUOT PONSARDIM Garraf... | 366.90 |
| **6** | Champagne Frances Brut Imperial MOET Rose Garr... | 359.90 |
| **7** | Conjunto de Panelas Allegra em Inox TRAMONTINA... | 359.00 |
| **8** | Whisky Escoces CHIVAS REGAL 18 Anos Garrafa 750ml | 329.90 |
| **9** | Champagne Frances Brut Imperial MOET & CHANDON... | 315.90 |

---

**2) What sections do the 'BEBIDAS' and 'PADARIA' departments have?**

**Answer:** Using the query "SELECT DISTINCT SECTION_NAME FROM looqbox_challenge.data_product WHERE DEP_NAME = 'BEBIDAS' OR DEP_NAME = 'PADARIA'"

In [3]:
```
resposta2 = pd.read_sql('''SELECT DISTINCT SECTION_NAME FROM looqbox_challenge.data_product
WHERE DEP_NAME = 'BEBIDAS' OR DEP_NAME = 'PADARIA' ''', mydb)
display(resposta2)
```

| | SECTION_NAME |
|---|---|
| **0** | BEBIDAS |
| **1** | VINHOS |
| **2** | DOCES-E-SOBREMESAS |
| **3** | QUEIJOS-E-FRIOS |
| **4** | CERVEJAS |
| **5** | PADARIA |
| **6** | REFRESCOS |
| **7** | GESTANTE |

---

**3) What was the total sale of products of each Business Area in the first quarter of 2019?**
**Answer:** Using the query "SELECT SUM(A.SALES_VALUE), B.DEP_NAME FROM looqbox_challenge.data_product_sales as A JOIN looqbox_challenge.data_product as B ON A.PRODUCT_CODE = B.PRODUCT_COD WHERE A.DATE BETWEEN '2019-01-01' AND '2019-04-01' GROUP BY B.DEP_NAME"

In [4]:
```
resposta3 = pd.read_sql('''SELECT B.DEP_NAME, SUM(A.SALES_VALUE)
FROM looqbox_challenge.data_product_sales as A
JOIN looqbox_challenge.data_product as B
ON A.PRODUCT_CODE = B.PRODUCT_COD
WHERE A.DATE BETWEEN '2019-01-01' AND '2019-04-01'
GROUP BY B.DEP_NAME
ORDER BY SUM(A.SALES_VALUE) DESC''', mydb)
display(resposta3)
```

| | DEP_NAME | SUM(A.SALES_VALUE) |
|---|---|---|
| **0** | BEBES | 61734500.71 |
| **1** | PERFUMARIA | 52411043.58 |
| **2** | MERCEARIA | 37209083.78 |
| **3** | MEDICAMENTOS REFERÊNCIA | 30045693.96 |
| **4** | FRIOS | 30025587.26 |
| **5** | PET-SHOP | 29449293.60 |
| **6** | BEBIDAS | 28807393.28 |
| **7** | CARNES | 25939728.36 |
| **8** | FLV | 21326530.58 |
| **9** | MEDICAMENTOS GENÉRICOS | 17525096.10 |
| **10** | PADARIA | 15199616.74 |

---

## CASES

**1) The Dev Team was tired of developing the same old queries just varying the filters accordingly to their boss demands. As a new member of the crew, your mission now is to create a dynamic function, on the most flexible of ways, to produce queries and retrieve a dataframe based on three parameters:**

**product_code: integer**

**store_code: integer**

**date: list of ISO-like strings**

**Date e.g.**

**['2019-01-01', '2019-01-31'] → Python**

**c('2019-01-01', '2019-01-31') → R**

**It should look like this my_data = retrieve_data(product_code, store_code, date)**

**Make your team proud!**

**Extra instructions: Retrieve all columns from table data_product_sales.**

In [5]:
```
# Answer: this function will return a dataframe based on the parameters that the boss pass.
def querye(product_code, store_code, date):
    querie = pd.read_sql(r'''SELECT *
                     FROM looqbox_challenge.data_product_sales
                     WHERE product_code = {} AND
                     store_code = {} AND
                     date = '{}'
                     '''.format(product_code, store_code, date),
                    mydb)
    display(querie)
#resultado da query
querye(18,1,'2019-01-01')
```

| | STORE_CODE | PRODUCT_CODE | DATE | SALES_VALUE | SALES_QTY |
|---|---|---|---|---|---|
| **0** | 1 | 18 | 2019-01-01 | 708.5 | 65.0 |

---

**2) A brand new client sent you two ready-to-go queries. Those are listed below: Query 1:**

**SELECT STORE_CODE, STORE_NAME, START_DATE, END_DATE, BUSINESS_NAME, BUSINESS_CODE FROM data_store_cad Query 2:**

**SELECT STORE_CODE, DATE, SALES_VALUE, SALES_QTY FROM data_store_sales WHERE DATE BETWEEN '2019-01-01' AND '2019-12-31' In addition, he gave you this set of instructions:**

**You must not modify my queries!**

**Please filter the period between this given range:**

**['2019-10-01','2019-12-31']**

**Answer:** The query that will return the rows betwwen the dates that the custumer resquested is: "SELECT STORE_CODE, DATE, SALES_VALUE, SALES_QTY FROM looqbox_challenge.data_store_sales WHERE DATE BETWEEN '2019-10-01' AND '2019-12-31' "

---

## ANSWER BUILDING VISUALIZATION

**Building your own visualization Create at least one chart using the table IMDB_movies. The code must be in R or Python, and you are free to use any libraries, data in the table and graphic format. Explain why you chose the visualization (or visualizations) you are submitting.**

In [6]:
```
filmes_top_rev = pd.read_sql('''SELECT Title, Genre, Year, Rating, RevenueMillions
                     FROM looqbox_challenge.IMDB_movies
                     ORDER BY RevenueMillions  DESC LIMIT 10''',
                    mydb)
filmes_por_genre = pd.read_sql('''SELECT Genre, AVG(Metascore)
                     FROM looqbox_challenge.IMDB_movies
                     GROUP BY Genre
                     ORDER BY AVG(Metascore) desc
                     LIMIT 10
                     ''',
                    mydb)
genre_por_duracao = pd.read_sql('''SELECT Genre, AVG(Runtime)
                    FROM looqbox_challenge.IMDB_movies
                    GROUP BY Genre
                    ORDER BY AVG(Runtime) DESC
                    LIMIT 5

                     ''',
                    mydb)
metascore_and_revenue = pd.read_sql('''SELECT Metascore, RevenueMillions
FROM looqbox_challenge.IMDB_movies
where RevenueMillions is not null and
Metascore is not null
''',
mydb
)
```

In [7]:
```
display(filmes_top_rev)
```

| | Title | Genre | Year | Rating | RevenueMillions |
|---|---|---|---|---|---|
| **0** | Star Wars: Episode VII - The Force Awakens | Action,Adventure,Fantasy | 2015 | 8.0 | 937.0 |
| **1** | Avatar | Action,Adventure,Fantasy | 2009 | 8.0 | 761.0 |
| **2** | Jurassic World | Action,Adventure,Sci-Fi | 2015 | 7.0 | 652.0 |
| **3** | The Avengers | Action,Sci-Fi | 2012 | 8.0 | 623.0 |
| **4** | The Dark Knight | Action,Crime,Drama | 2008 | 9.0 | 533.0 |
| **5** | Rogue One | Action,Adventure,Sci-Fi | 2016 | 8.0 | 532.0 |
| **6** | Finding Dory | Animation,Adventure,Comedy | 2016 | 7.0 | 486.0 |
| **7** | Avengers: Age of Ultron | Action,Adventure,Sci-Fi | 2015 | 7.0 | 459.0 |
| **8** | The Dark Knight Rises | Action,Thriller | 2012 | 9.0 | 448.0 |
| **9** | The Hunger Games: Catching Fire | Action,Adventure,Mystery | 2013 | 8.0 | 425.0 |

In [8]:
```
display(filmes_por_genre)
```

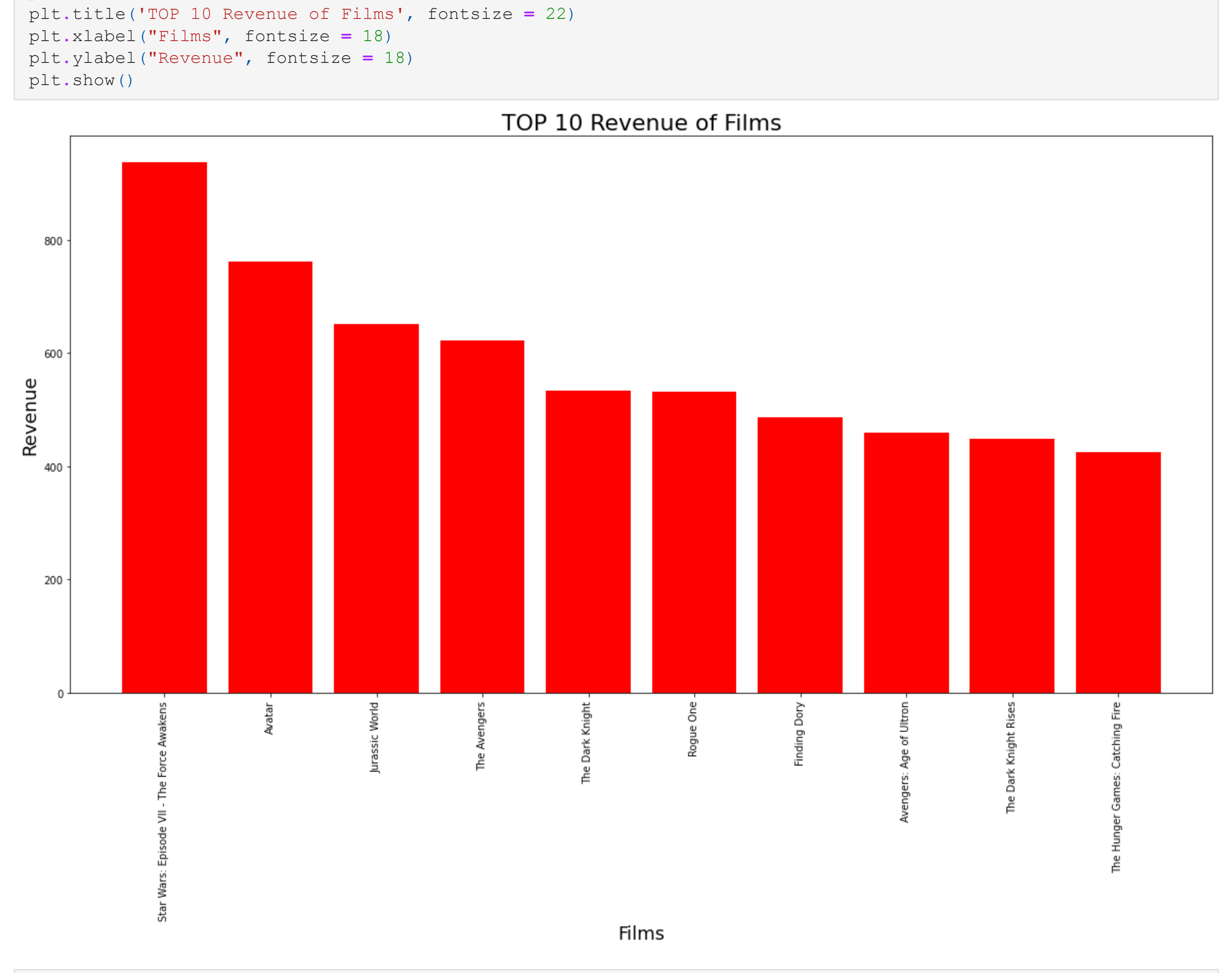
| | Genre | AVG(Metascore) |
|---|---|---|
| **0** | Drama,Fantasy,War | 98.00 |
| **1** | Animation,Fantasy | 86.00 |
| **2** | Crime,Drama,History | 85.50 |
| **3** | Animation,Comedy,Drama | 85.00 |
| **4** | Drama,History,Thriller | 84.75 |
| **5** | Animation,Adventure,Family | 84.00 |
| **6** | Drama,Horror,Musical | 83.00 |
| **7** | Drama,Western | 81.00 |
| **8** | Comedy,Fantasy,Romance | 81.00 |
| **9** | Action,Comedy,Sci-Fi | 81.00 |

In [9]:
```
display(genre_por_duracao)
```

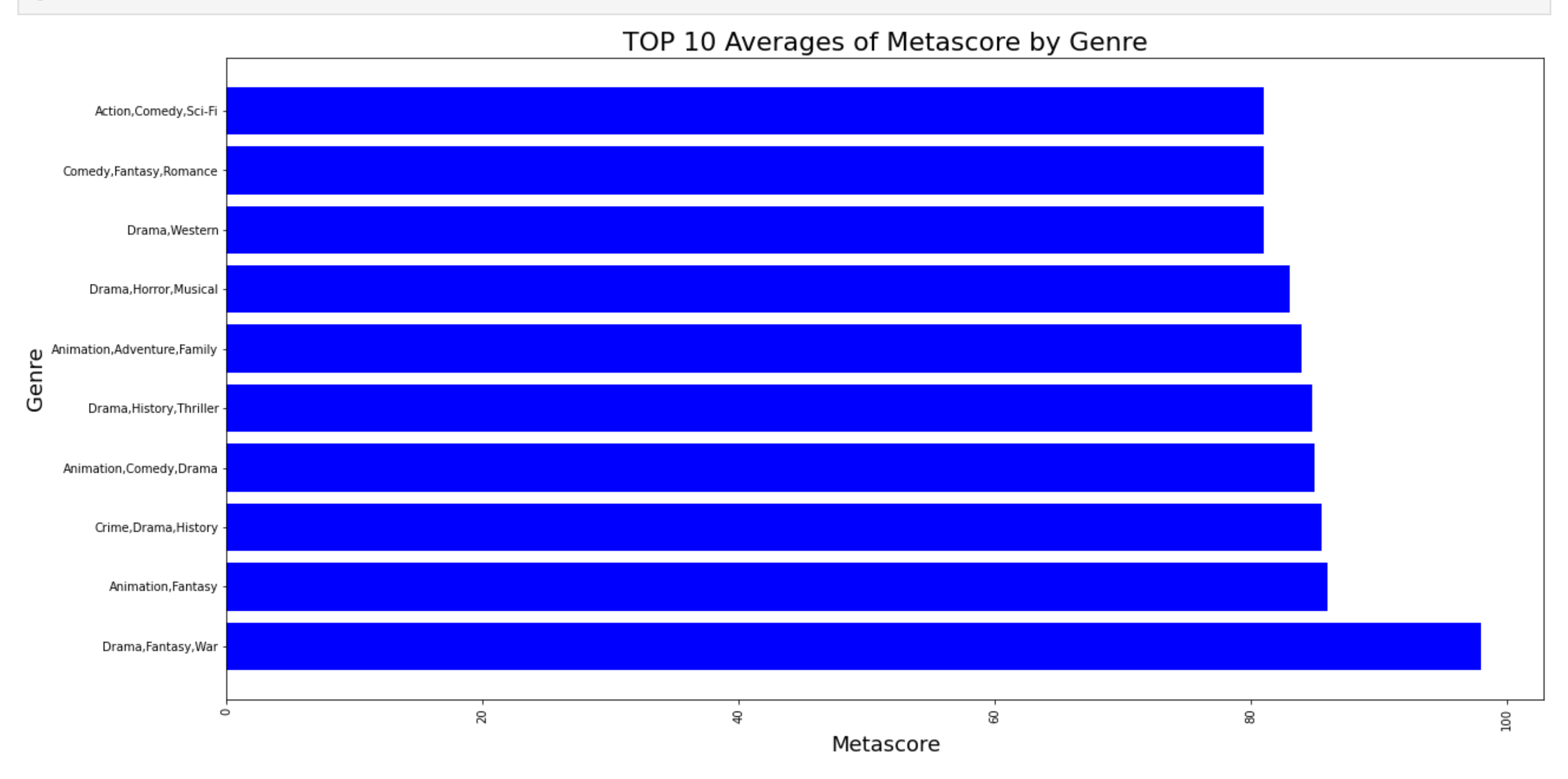
| | Genre | AVG(Runtime) |
|---|---|---|
| **0** | Biography,Comedy,Crime | 180.0 |
| **1** | Drama,Western | 165.0 |
| **2** | Drama,Family,Music | 165.0 |
| **3** | Adventure,Drama,History | 161.0 |
| **4** | Drama,Musical,Romance | 158.0 |

3)

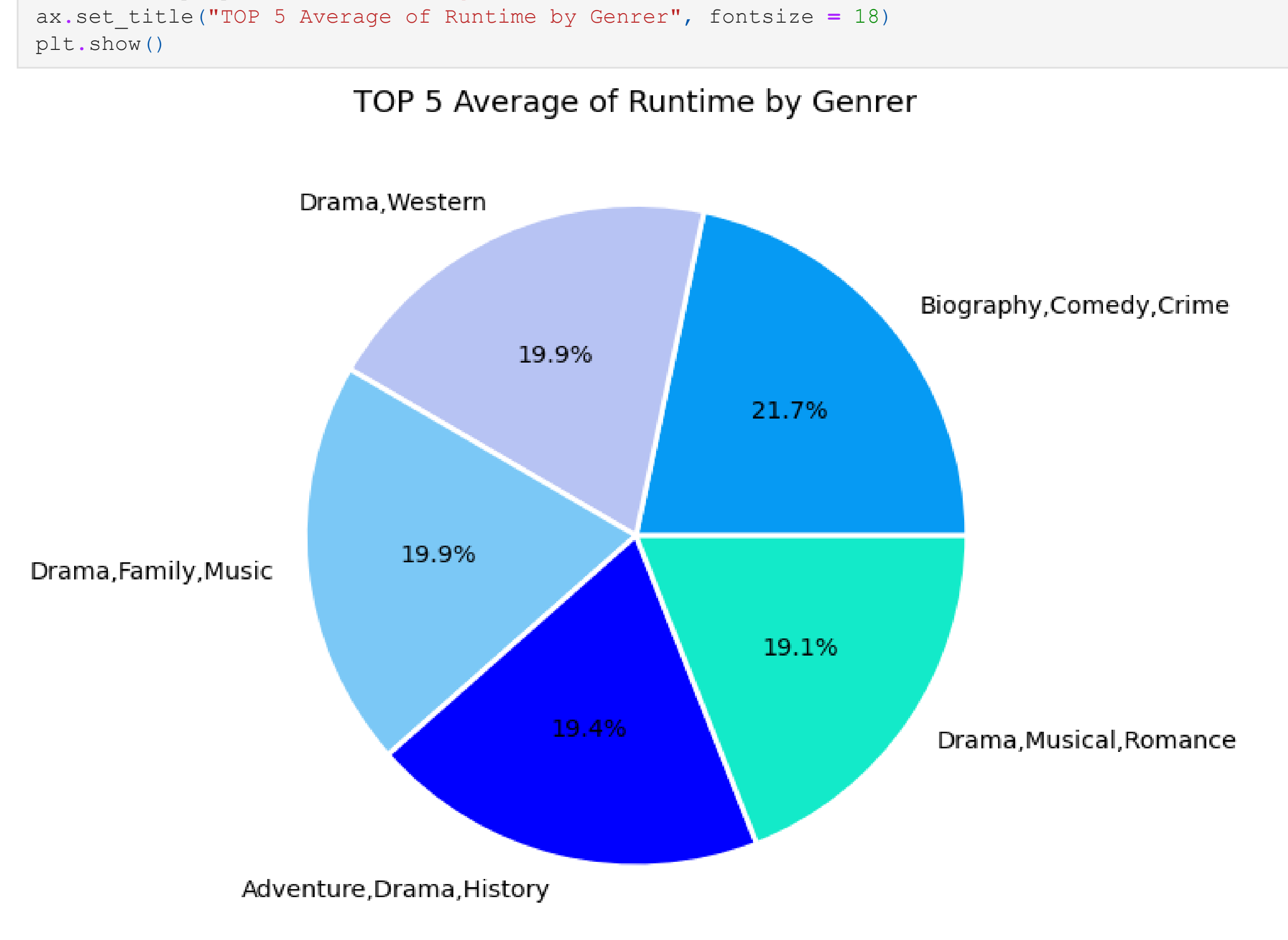
In [10]:
```
plt.figure(figsize=(20,10))
plt.bar(filmes_top_rev['Title'], filmes_top_rev['RevenueMillions'], color = 'red')
plt.xticks(rotation = 'vertical')
plt.title('TOP 10 Revenue of Films', fontsize = 22)
plt.xlabel("Films", fontsize = 18)
plt.ylabel("Revenue", fontsize = 18)
plt.show()
```



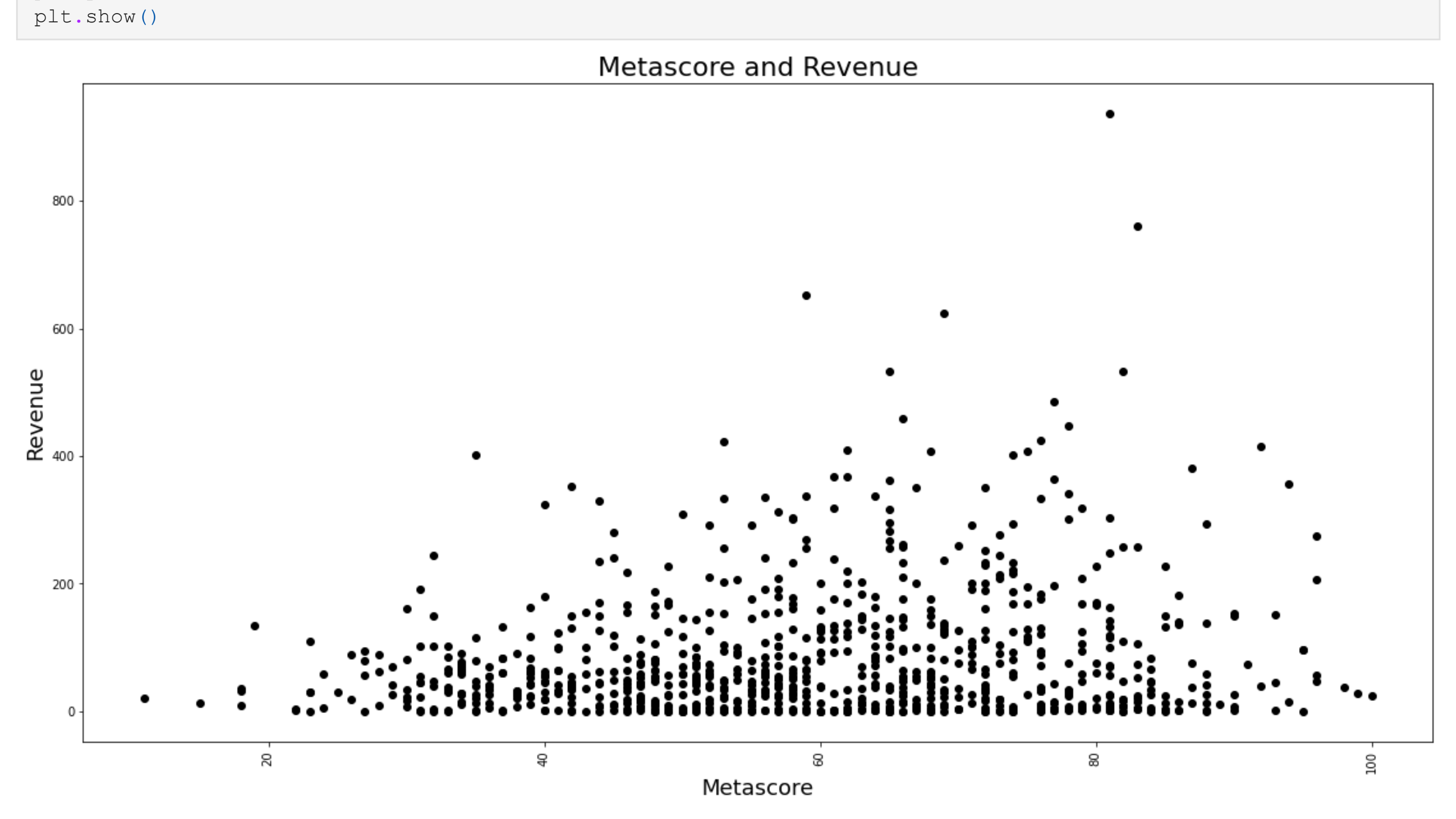
In [11]:
```
plt.figure(figsize=(20,10))
plt.barh(filmes_por_genre['Genre'],filmes_por_genre['AVG(Metascore)'] , color = 'blue')
plt.xticks(rotation = 'vertical')
plt.title('TOP 10 Averages of Metascore by Genre', fontsize = 22)
plt.xlabel("Metascore", fontsize = 18)
plt.ylabel("Genre", fontsize = 18)
plt.show()
```



In [12]:
```
colors = ['#069AF3', '#B7C3F3', '#7BC8F6', 'b', '#13EAC9']
fig, ax =  plt.subplots(figsize=(20, 9), subplot_kw=dict(aspect="equal"))
ax.pie(genre_por_duracao['AVG(Runtime)'], labels= genre_por_duracao['Genre'], autopct='%.1f%%', wedgeprops={'li
       textprops={'size': 'x-large'}, colors=colors)
ax.set_title("TOP 5 Average of Runtime by Genrer", fontsize = 18)
plt.show()
```



In [13]:
```
plt.figure(figsize=(20,10))
plt.scatter(metascore_and_revenue['Metascore'],metascore_and_revenue['RevenueMillions'] , color = 'black')
plt.xticks(rotation = 'vertical')
plt.title('Metascore and Revenue', fontsize = 22)
plt.xlabel("Metascore", fontsize = 18)
plt.ylabel("Revenue", fontsize = 18)
plt.show()
```



### Explain why you chose the visualization (or visualizations) you are submitting.

**Answer:** I think that this 3 graphs show important metrics about the database IMBD_movies. Showing the total amount of observations can provides saturation of information, because of that i decide to build visualitions that show top 10 or 5 rows of the query. The only graph that i decide plot all the observations is the Metascore and Revenue, where we can see if this two values have some correlation. We can see the coeficient of correlation in the chunk below, in this case the coeficient is 0.14, indicanting that this two variables have just a little positive correlation.

In [14]:
```
stats.pearsonr(metascore_and_revenue['Metascore'],metascore_and_revenue['RevenueMillions'])
```

Out[14]:
```
(0.14238405137452126, 3.523250523195723e-05)
```