INITIAL SETUP

In [1]:

```
#INGESTING CRITICS DATA FILE INTO CRITICS
import boto3
import pandas as pd
s3_client = boto3.client("s3")

BUCKET='ncinemas'
KEY='cx/cx.csv'

response = s3_client.get_object(Bucket=BUCKET, Key=KEY)
cx = pd.read_csv(response.get("Body"))
cx.head(10)
```

Out[1]:

	ID_Movie	Expert	Score	Sentiment	Review
0	7369	RogerEbert.com	88	3	Call Me Lucky will be an especially grueling r...
1	7369	New York Daily News	80	1	Angry, quixotic, tragic, heroic — Crimmins' li...
2	7369	Village Voice	80	7	Call Me Lucky is a loving but fair portrait of...
3	7369	TheWrap	75	2	There should be more Crimmins performance foot...
4	7369	Movie Nation	75	3	Call Me Lucky is another of those "the funnies...
5	7369	The A.V. Club	67	1	Goldthwait stays behind the camera, but his lo...
6	7369	Slant Magazine	63	-2	Bobcat Goldthwait's hand too nervously tempers...
7	7369	The New York Times	50	2	The movie strains to drum up mystery as to the...
8	7369	Austin Chronicle	50	3	You'll be the richer for spending time in Crim...
9	7369	Washington Post	37	0	Ironically, Call Me Lucky, a worshipful new do...

In [2]:

```
#INGESTING USERS DATA FILE INTO USER

BUCKET='ncinemas'
KEY='user/user.csv'

response = s3_client.get_object(Bucket=BUCKET, Key=KEY)
user = pd.read_csv(response.get("Body"))
user.head(10)
```

Out[2]:

	ID_Movie	Score	Sentiment	User	Review
0	1	7	12	DemiRonin	$9.99 is a series of unique short stories. How...
1	1	4	2	steven	I don't mean to be a Debbie Downer and I am al...
2	3	9	0	RayJ.	Superb.
3	3	9	0	MichaelV.	Lillo is so hot!
4	3	6	-4	GilbertMulroneycakesAndFriends	What the hell is that title all about? I assum...
5	3	10	0	Ice-T	My movies rock!
6	4	7	-4	Swati	It had its moments. I could not shake off the ...
7	4	7	5	applesandorange	This movie is unique and not like any other lo...
8	4	10	7	Famousdog	I loooove coming into a film with absolutely n...
9	4	8	17	drlowdon	Starring Joseph Gordon-Levitt and the lovely Z...

In [3]:

```
#INGESTING SALE DATA TABLE INTO SALE

s3_client = boto3.client("s3")
BUCKET='ncinemas'
KEY='sale/sale.csv'

response = s3_client.get_object(Bucket=BUCKET, Key=KEY)
sale = pd.read_csv(response.get("Body"))
sale.head()
```

Out[3]:

	MovieName	Rank_data	PreviousWeekRank	GrossW	Theaters
0	Stuart Little	1	1	13012299	2979
1	The Green Mile	2	3	12521303	2678
2	The Talented Mr. Ripley	3	2	11780319	2316
3	Any Given Sunday	4	4	10971011	2505
4	Galaxy Quest	5	6	9784389	2450

In [4]:

```
#INGESTING PRODUCT DATA TABLE IN PRODUCT

BUCKET='ncinemas'
KEY='product/product.csv'

response = s3_client.get_object(Bucket=BUCKET, Key=KEY)
product = pd.read_csv(response.get("Body"))
product.head()
```

Out[4]:

	ID	Title	Publisher	Metascore	Meta_Pos_Count	Meta_Neut_Count	Meta_
0	1	9.99	Regent Releasing	68.0	12.0	3.0	
1	2	$pent	Regent Releasing	34.0	1.0	3.0	
2	3	'R Xmas	Pathfinder Pictures	55.0	4.0	5.0	
3	4	(500) Days of Summer	Fox Searchlight Pictures	76.0	33.0	3.0	
4	5	1	IFC Midnight	60.0	3.0	2.0	

AFTER LOOKING AT THE DATA TABLE MANUALLY, IT'S NOTED THAT PRODUCT AND SALE DATA TABLES ARE TWO MOST USEFUL DATASETS FOR OUR PROJECT.

THUS, THESE TWO TABLES ARE COMBINED, DROPPED THE UNNECESSARY COLUMNS, FORMAT THE RUN-TIME(STRNG) INTO NUMERICAL COLUMN AS RUNTIME ** AND RATING COLUMNS ARE CATEGORIZED ACCORDINGLY AND SAVED INTO THE DATA TABLE NAMED 'MOVIERE'

In [5]:

```
#INGESTING MOVIERE DATA TABLE INTO MOVIERE

BUCKET='ncinemas'
KEY='etiocinemas/etiocinemas.csv'

response = s3_client.get_object(Bucket=BUCKET, Key=KEY)
etiocinemas = pd.read_csv(response.get("Body"))
etiocinemas.head()
```

Out[5]:

_Score	User_Pos_Count	User_Neut_Count	User_Neg_Count	MovieName	GrossW
6.8	0	0	0	Stuart Little	13012299
6.8	0	0	0	Stuart Little	13560203
6.8	0	0	0	Stuart Little	7554094
6.8	0	0	0	Stuart Little	5690856
6.8	0	0	0	Stuart Little	5394977

ATHENA SECTION

In [6]:

```
#Locating the S3 bucket

!aws s3 ls s3://ncinemas/cx/
!aws s3 ls s3://ncinemas/user/
!aws s3 ls s3://ncinemas/product/
!aws s3 ls s3://ncinemas/sale/
!aws s3 ls s3://ncinemas/etiocinemas/
```

```
2022-03-27 22:22:48          0
2022-03-27 22:23:01   43253320 cx.csv
2022-03-27 22:22:04          0
2022-03-27 22:22:30    1081196 user.csv
2022-03-27 22:24:53          0
2022-03-27 22:25:19    9678192 product.csv
2022-03-27 22:23:55          0
2022-03-27 22:24:19    3983441 sale.csv
2022-03-27 23:49:45          0
2022-03-27 23:50:25    7577491 etiocinemas.csv
```

In [7]:

```
import boto3
import sagemaker
import pandas as pd
sess = sagemaker.Session()
bucket = sess.default_bucket()
role = sagemaker.get_execution_role()
region = boto3.Session().region_name
account_id = boto3.client("sts").get_caller_identity().get("Account")
sm = boto3.Session().client(service_name="sagemaker", region_name=regio
n)
```

In [8]:

```
#SET S3 SOURCE LOCATION (PUBLIC S3 BUCKET)
s3_public_path_csv = "s3://ncinemas"
%store s3_public_path_csv
```

```
Stored 's3_public_path_csv' (str)
```

In [9]:

```
#SET S3 DESTINATION LOCATION (PRIVATE S3 BUCKET)
s3_private_path_csv = "s3://{}/ncinemas".format(bucket)
print(s3_private_path_csv)
```

```
s3://sagemaker-us-east-1-364962763824/ncinemas
```

In [10]:

```
#COPY DATA FROM PUBLIC BUCKET S3 TO OUR PRIVATE S3 BUCKET
!aws s3 cp --recursive $s3_public_path_csv/ $s3_private_path_csv/ --exc
lude "*" --include "cx/cx.csv"
!aws s3 cp --recursive $s3_public_path_csv/ $s3_private_path_csv/ --exc
lude "*" --include "user/user.csv"
!aws s3 cp --recursive $s3_public_path_csv/ $s3_private_path_csv/ --exc
lude "*" --include "product/product.csv"
!aws s3 cp --recursive $s3_public_path_csv/ $s3_private_path_csv/ --exc
lude "*" --include "sale/sale.csv"
!aws s3 cp --recursive $s3_public_path_csv/ $s3_private_path_csv/ --exc
lude "*" --include "etiocinemas/etiocinemas.csv"
```

```
copy: s3://ncinemas/cx/cx.csv to s3://sagemaker-us-east-1-36
4962763824/ncinemas/cx/cx.csv
copy: s3://ncinemas/user/user.csv to s3://sagemaker-us-east-
1-364962763824/ncinemas/user/user.csv
copy: s3://ncinemas/product/product.csv to s3://sagemaker-us
-east-1-364962763824/ncinemas/product/product.csv
copy: s3://ncinemas/sale/sale.csv to s3://sagemaker-us-east-
1-364962763824/ncinemas/sale/sale.csv
copy: s3://ncinemas/etiocinemas/etiocinemas.csv to s3://sage
maker-us-east-1-364962763824/ncinemas/etiocinemas/etiocinema
s.csv
```

In [11]:

```
 print(s3_private_path_csv)
```

```
s3://sagemaker-us-east-1-364962763824/ncinemas
```

In [12]:

```
!aws s3 ls $s3_private_path_csv/cx/
!aws s3 ls $s3_private_path_csv/user/
!aws s3 ls $s3_private_path_csv/product/
!aws s3 ls $s3_private_path_csv/sale/
!aws s3 ls $s3_private_path_csv/etiocinemas/
```

```
2022-03-28 00:37:21   43253320 cx.csv
2022-03-28 00:37:22    1081196 user.csv
2022-03-28 00:37:23    9678192 product.csv
2022-03-28 00:37:24    3983441 sale.csv
2022-03-28 00:37:25    7577491 etiocinemas.csv
```

In [13]:

```
!pip install --disable-pip-version-check -q PyAthena==2.1.0
from pyathena import connect
```

```
/opt/conda/lib/python3.7/site-packages/secretstorage/dhcrypt
o.py:16: CryptographyDeprecationWarning: int_from_bytes is d
eprecated, use int.from_bytes instead
  from cryptography.utils import int_from_bytes
/opt/conda/lib/python3.7/site-packages/secretstorage/util.p
y:25: CryptographyDeprecationWarning: int_from_bytes is depr
ecated, use int.from_bytes instead
  from cryptography.utils import int_from_bytes
WARNING: Running pip as the 'root' user can result in broken
permissions and conflicting behaviour with the system packag
e manager. It is recommended to use a virtual environment in
stead: https://pip.pypa.io/warnings/venv
```

In [14]:

```
# Set S3 staging directory -- this is a temporary directory used for At
hena queries
s3_private_path_cx = "s3://{}/ncinemas/cx/".format(bucket)
print(s3_private_path_cx)
s3_private_path_user = "s3://{}/ncinemas/user/".format(bucket)
print(s3_private_path_user)
s3_private_path_product = "s3://{}/ncinemas/product/".format(bucket)
print(s3_private_path_product)
s3_private_path_sale = "s3://{}/ncinemas/sale/".format(bucket)
print(s3_private_path_sale)
s3_private_path_etiocinemas = "s3://{}/ncinemas/etiocinemas/".format(bu
cket)
print(s3_private_path_etiocinemas)
```

```
s3://sagemaker-us-east-1-364962763824/ncinemas/cx/
s3://sagemaker-us-east-1-364962763824/ncinemas/user/
s3://sagemaker-us-east-1-364962763824/ncinemas/product/
s3://sagemaker-us-east-1-364962763824/ncinemas/sale/
s3://sagemaker-us-east-1-364962763824/ncinemas/etiocinemas/
```

In [15]:

```
s3_staging_dir = "s3://{0}/athena/staging".format(bucket)
conn = connect(region_name=region, s3_staging_dir=s3_staging_dir)
```

In [16]:

```
database_name = "ncinemas"
statement = "CREATE DATABASE IF NOT EXISTS {}".format(database_name)
print(statement)
pd.read_sql(statement, conn)
statement = "SHOW DATABASES"
df_show = pd.read_sql(statement, conn)
df_show.head(20)
```

```
CREATE DATABASE IF NOT EXISTS ncinemas
```

Out[16]:

	database_name
0	default
1	ncinemas

In [17]:

```
#SETTING UP ATHENA PARAMETERS
cx = "cx"
user = "user"
product = "product"
sale = "sale"
etiocinemas = "etiocinemas"
```

In [18]:

```
#SQL STATEMENT FOR CRITICS TABLE

statement_cx = """CREATE EXTERNAL TABLE IF NOT EXISTS {}.{}(
         ID_Movie  int,
         Expert    string,
         Score     int,
         Sentiment int,
         Review    string
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n'
 LOCATION '{}'
TBLPROPERTIES ('skip.header.line.count'='1')""".format(
 database_name, cx, s3_private_path_cx
)
pd.read_sql(statement_cx, conn)
```

Out[18]:

—

In [19]:

```
#CHECKING IF THE TABLE WAS CREATED PROPERLY

statement = """SELECT * FROM {}.{} limit 10""".format( database_name, c
x)
print(statement)
df = pd.read_sql(statement, conn)
df.head()
```

```
SELECT * FROM ncinemas.cx limit 10
```

Out[19]:

	id_movie	expert	score	sentiment	review
0	7369	RogerEbert.com	88	3	Call Me Lucky will be an especially grueling r...
1	7369	New York Daily News	80	1	"Angry
2	7369	Village Voice	80	7	Call Me Lucky is a loving but fair portrait of...
3	7369	TheWrap	75	2	"There should be more Crimmins performance foo...
4	7369	Movie Nation	75	3	Call Me Lucky is another of those “the funnies...

In [20]:

```
#SQL STATEMENT FOR USER TABLE

statement_user = """CREATE EXTERNAL TABLE IF NOT EXISTS {}.{}(
         ID_Movie int,
         Score int,
         Sentiment int,
         User string,
         Review string
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n'
 LOCATION '{}'
TBLPROPERTIES ('skip.header.line.count'='1')""".format(
 database_name, user, s3_private_path_user
)
pd.read_sql(statement_user, conn)
```

Out[20]:

In [21]:

```
#CHECKING IF THE TABLE WAS CREATED PROPERLY

statement = """SELECT * FROM {}.{} limit 10""".format( database_name, u
ser)
print(statement)
df = pd.read_sql(statement, conn)
df.head()
```

```
SELECT * FROM ncinemas.user limit 10
```

Out[21]:

	id_movie	score	sentiment	user	review
0	1	7	12	DemiRonin	"$9.99 is a series of unique short stories. Ho...
1	1	4	2	steven	"I don't mean to be a Debbie Downer and I am a...
2	3	9	0	RayJ.	Superb.
3	3	9	0	MichaelV.	Lillo is so hot!
4	3	6	-4	GilbertMulroneycakesAndFriends	"What the hell is that title all about? I assu...

In [22]:

```
#SQL STATEMENT FOR SALE TABLE

statement_sale = """CREATE EXTERNAL TABLE IF NOT EXISTS {}.{}(
         MovieName          string,
         Rank_data          int,
         PreviousWeekRank   int,
         GrossW             bigint,
         Theaters           int

)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n'
 LOCATION '{}'
TBLPROPERTIES ('skip.header.line.count'='1')""".format(
 database_name, sale , s3_private_path_sale
)
pd.read_sql(statement_sale, conn)
```

Out[22]:

—

In [23]:

```
#CHECKING IF THE TABLE WAS CREATED PROPERLY

statement = """SELECT * FROM {}.{} limit 10""".format( database_name, s
ale)
print(statement)
df = pd.read_sql(statement, conn)
df.head()
```

```
SELECT * FROM ncinemas.sale limit 10
```

Out[23]:

	moviename	rank_data	previousweekrank	grossw	theaters
0	Stuart Little	1	1	13012299	2979
1	The Green Mile	2	3	12521303	2678
2	The Talented Mr. Ripley	3	2	11780319	2316
3	Any Given Sunday	4	4	10971011	2505
4	Galaxy Quest	5	6	9784389	2450

In [24]:

```
#CHECKING IF THE TABLE WAS CREATED PROPERLY

statement = """SELECT * FROM {}.{} limit 10""".format( database_name, p
roduct)
print(statement)
df = pd.read_sql(statement, conn)
df.head()
```

```
SELECT * FROM ncinemas.product limit 10
```

Out[24]:

	id	title	publisher	metascore	meta_pos_count	meta_neut_count	meta_n
0	1	9.99	Regent Releasing	68.0	12	3	
1	2	$pent	Regent Releasing	34.0	1	3	
2	3	'R Xmas	Pathfinder Pictures	55.0	4	5	
3	4	(500) Days of Summer	Fox Searchlight Pictures	76.0	33	3	
4	5	1	IFC Midnight	60.0	3	2	

In [25]:

```
#SQL STATEMENT FOR EMOVIES TABLE

statement_etiocinemas = """CREATE EXTERNAL TABLE IF NOT EXISTS {}.{}(
         Unnamed             int,
         Int                 int,
         Rank_data           int,
         PreviousWeekRank    int,
         Metascore           int,
         Meta_Pos_Count      int,
         Meta_Neut_Count     int,
         Meta_Neg_Count      int,
         User_Score          int,
         User_Pos_Count      int,
         User_Neut_Count     int,
         User_Neg_Count      int,
         MovieName           string,
         GrossW              bigint,
         Runtime_Value       int,
         Rated               string,
         Publisher           string,
         Director            string,
         Genre               string

)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n'
 LOCATION '{}'
TBLPROPERTIES ('skip.header.line.count'='1')""".format(
 database_name, etiocinemas , s3_private_path_etiocinemas
)
pd.read_sql(statement_etiocinemas, conn)
```

Out[25]:

In [26]:

```
#CHECKING IF THE TABLE WAS CREATED PROPERLY

statement = """SELECT * FROM {}.{} limit 10""".format( database_name, e
tiocinemas)
print(statement)
df = pd.read_sql(statement, conn)
df.head()
```

```
SELECT * FROM ncinemas.etiocinemas limit 10
```

Out[26]:

	unnamed	int	rank_data	previousweekrank	metascore	meta_pos_count	meta
0	0	0	1	1	61	22	
1	1	1	2	1	61	22	
2	2	2	4	2	61	22	
3	3	3	4	4	61	22	
4	4	4	3	4	61	22	

DATA EXPLORATION

In [27]:

```
#CHECKING TOP TEN MOVIENAME WITH METASCORE

statement = """SELECT MovieName, max(metascore) as Max_Metascore FROM
{}.{}
    group by MovieName
    order by Max_Metascore desc
   """.format(
    database_name, etiocinemas
)
print(statement)
df = pd.read_sql(statement, conn)
df.head(10)
```

```
SELECT MovieName, max(metascore) as Max_Metascore FROM ncine
mas.etiocinemas
    group by MovieName
    order by Max_Metascore desc
```

Out[27]:

	MovieName	Max_Metascore
0	Boyhood	100
1	Moonlight	99
2	Manchester by the Sea	96
3	Parasite	96
4	A Separation	95
5	The Social Network	95
6	Portrait of a Lady on Fire	95
7	Amour	94
8	Sideways	94
9	45 Years	94

In [28]:

```
#CHECKING TOP TEN MOVIENAME WITH USERSCORE

statement = """SELECT MovieName, max(User_Score) as Max_User_Score FROM
{}.{}
    group by MovieName
    order by Max_User_Score desc
    limit 10
   """.format(
    database_name, etiocinemas
)
print(statement)
df = pd.read_sql(statement, conn)
df.head(10)
```

```
SELECT MovieName, max(User_Score) as Max_User_Score FROM nci
nemas.etiocinemas
    group by MovieName
    order by Max_User_Score desc
    limit 10
```

Out[28]:

	MovieName	Max_User_Score
0	Dark Blue World	9
1	Izzy Gets the F*ck Across Town	9
2	Memento	9
3	Love & Basketball	9
4	The Best of Youth	9
5	The Man Who Copied	9
6	Neil Young: Heart of Gold	9
7	Skins	9
8	Quitting	9
9	Diamond Men	9

In [37]:

```
##CHECKING TOP 10 MOVIES THAT RANKED TOP 20 PERCENTILE BY METASCORE AND
USERSCORE


statement = """SELECT MovieName, Metascore, User_Score, Rated FROM {}.
{}
    where Metascore > 80 and User_Score > 7.2
    group by MovieName, Rated, Metascore, User_Score
    order by Metascore desc
    limit 10
   """.format(
    database_name, etiocinemas
)
print(statement)
top20rank = pd.read_sql(statement, conn)
top20rank.head(10)
```

```
SELECT MovieName, Metascore, User_Score, Rated FROM ncinema
s.etiocinemas
    where Metascore > 80 and User_Score > 7.2
    group by MovieName, Rated, Metascore, User_Score
    order by Metascore desc
    limit 10
```

Out[37]:

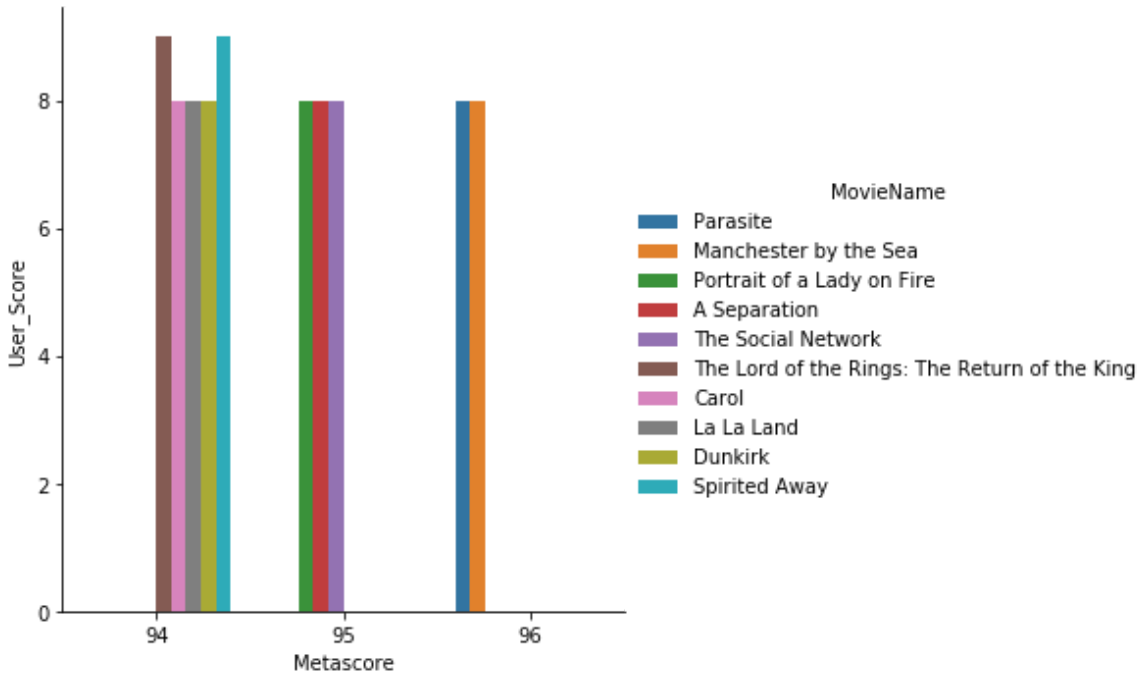
	MovieName	Metascore	User_Score	Rated
0	Parasite	96	8	NR
1	Manchester by the Sea	96	8	R
2	Portrait of a Lady on Fire	95	8	R
3	A Separation	95	8	PG-13
4	The Social Network	95	8	PG-13
5	The Lord of the Rings: The Return of the King	94	9	PG-13
6	Carol	94	8	R
7	La La Land	94	8	PG-13
8	Dunkirk	94	8	PG-13
9	Spirited Away	94	9	PG

VISUALIZATION

In [38]:

```
from matplotlib import pyplot as plt
import seaborn as sns
plt.figure(figsize=(20,8))
ax = sns.catplot(data=top20rank, x="Metascore", y="User_Score", hue =
"MovieName", kind ="bar")
```

```
<Figure size 1440x576 with 0 Axes>
```



In [39]:

```
#CHECKING THE GROSS REVENUE OF TOP 20 RANKED MOVIES

statement = """SELECT MovieName, Metascore, User_Score, GrossW, Rated F
ROM {}.{}
    where Metascore > 80 and User_Score > 7.2
    group by GrossW, MovieName, Metascore, User_Score, Rated
    order by GrossW desc
    limit 10
   """.format(
    database_name, etiocinemas
)
print(statement)
rev_top20rank = pd.read_sql(statement, conn)
rev_top20rank.head(10)
```

```
SELECT MovieName, Metascore, User_Score, GrossW, Rated FROM
ncinemas.etiocinemas
    where Metascore > 80 and User_Score > 7.2
    group by GrossW, MovieName, Metascore, User_Score, Rated
    order by GrossW desc
    limit 10
```

Out[39]:

	MovieName	Metascore	User_Score	GrossW	Rated
0	The Dark Knight	82	9	238615211	PG-13
1	Inside Out	94	8	132817010	PG
2	The Lord of the Rings: The Return of the King	94	9	120199783	PG-13
3	The Dark Knight	82	9	112471635	PG-13
4	The Lord of the Rings: The Two Towers	88	9	111143998	PG-13
5	Star Trek	83	8	100610837	PG-13
6	The Bourne Ultimatum	85	8	98673300	PG-13
7	The Incredibles	90	8	93004485	PG
8	The Lord of the Rings: The Return of the King	94	9	90559979	PG-13
9	The Lord of the Rings: The Fellowship of the Ring	92	8	89248852	PG-13

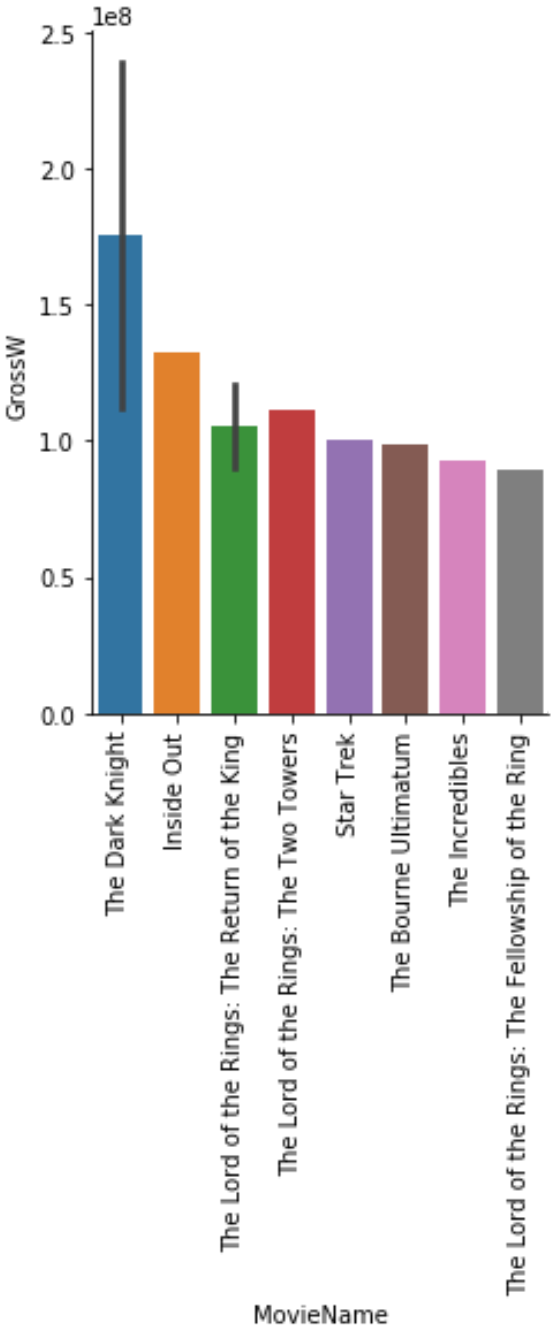
In [40]:

```
from matplotlib import pyplot as plt
import seaborn as sns
plt.figure(figsize=(150,100))
ax = sns.catplot(data=rev_top20rank, x="MovieName", y="GrossW", kind =
"bar")
plt.xticks(rotation=90)
```

Out[40]:

```
(array([0, 1, 2, 3, 4, 5, 6, 7]), <a list of 8 Text xticklab
el objects>)

<Figure size 10800x7200 with 0 Axes>
```



In [ ]: