Για το RDD (Εργασία 2):

Q1:

```
lines = spark.sparkContext.textFile("hdfs://master:9000/Ergasia/movies.csv")
mylist = lines.filter(lambda x:(x.split(",")[3] >= "1995" and x.split(",")[5] > "0" and x.split(",")[6] >= "0" ))
mylist2 = mylist.map(lambda x:  (x.split(',')[1],  int(x.split(',')[5])-int(x.split(',')[6])))
print(mylist2.collect())
```

Αφού φιλτράρω την στήλη των εσόδων, την στήλη του κόστους αλλά και την στήλη του έτους, ουσιαστικά, σπάω κάθε στήλη, την χωρίζω με την συνάρτηση split για να μπορέσω να βρω σωστά τα έσοδα.

Q2:

```
def split_complex(x):
   return list(csv.reader(StringIO(x), delimiter=','))[0]


movies = \
        spark.sparkContext.textFile("hdfs://master:9000/Ergasia/movies.csv"). \
        map(split_complex)

ratings = \
        spark.sparkContext.textFile("hdfs://master:9000/Ergasia/ratings.csv"). \
        map(lambda x: (x.split(",")[1], x.split(",")[2]))

cesareid = \
    movies.filter(lambda x: (x[1] == "Cesare deve morire")). \
    map(lambda x: (x[0])). \
    collect()[0]

print("cesareid:", cesareid) #id
mylist8 = ratings.filter(lambda x: (x[0] == cesareid))
print(mylist8.collect()) #o enwmenos pinakas me ta id kai tis kritikes

lines = spark.sparkContext.textFile("hdfs://master:9000/Ergasia/ratings.csv")
mylist0 = lines.filter(lambda x: (x.split(",")[0] == cesareid))
mylist4 = mylist0.map(lambda x: (x.split(",")[0]))
rtg_num = print(mylist4.count()) # Posoi vathmologhsan thn sygkekrimenh tainia
```

Χρησιμοποιώ την συνάρτηση split_complex, για να χωριστούν σωστά οι στήλες του αρχείου movies. Αφού κρατήσω ότι χρειάζομαι και από το αρχείο ratings, πρέπει να κρατήσω από την ταινία που με ενδιαφέρει το id. Οπότε, πάω στις ταινίες και αφού βρω την ταινία αυτή, με το map κρατάω από την πρώτη στήλη (στήλη 0 δηλαδή) το id. Ενώνω από τα δύο αρχεία τις στήλες που με νοιάζουν και χρησιμοποιώντας τα filter και map κατάλληλα, υπολογίζω πόσοι βαθμολόγησαν την συγκεκριμένη ταινία.

Q3:

```
lines = spark.sparkContext.textFile("hdfs://master:9000/Ergasia/movies.csv")

movies = \
    spark.sparkContext.textFile("hdfs://master:9000/Ergasia/movies.csv") .\
    filter(lambda x:(x.split(",")[3] == "1995" and x.split(",")[6] > "0")) .\
    map(lambda x: (x.split(",")[0], x.split(",")[6]))

movies_genres = \
    spark.sparkContext.textFile("hdfs://master:9000/Ergasia/movie_genres.csv"). \
    filter(lambda x: (x.split(",")[1] == "Animation")) .\
    map(lambda x: (x.split(",")[0], x.split(",")[1]))

poso = movies.join(movies_genres)
x = poso.sortBy(lambda x: x[1], ascending=False) # Taxinomhsh
print(x.take(1))

id_mv = "54648"
lines1 = spark.sparkContext.textFile("hdfs://master:9000/Ergasia/movies.csv")
mylist = lines1.filter(lambda x: (x.split(",")[0] == id_mv))
mylist1 = mylist.map(lambda x: (x.split(",")[1]))
print("H tainia me to megalytero poso:", mylist1.collect())
```

Φιλτράρω τις στήλες των εσόδων και τις στήλες του έτους και κρατάω από τα αρχεία movies και movies_genres το έτος 1995 με τα έσοδα και την κατηγορία animation, αντίστοιχα. Ενώνω τα δύο αυτά αρχεία και για να κρατήσω το id της ταινίας με τα μεγαλύτερα έσοδα, κάνω μία φθίνουσα ταξινόμηση και κρατάω το πρώτο id που εμφανίζεται. Έτσι, αφού έχω βρει το id και ουσιαστικά την ταινία, κρατάω την στήλη με τα ονόματα και εκτυπώνω το όνομα της ταινίας αυτής.

Q4:

```
def taxhnomhsh(g):
    return g.sort((x[0][1][1]))
    max(g, key=itemgetter([0][0][1]))



movies_genres = \
    spark.sparkContext.textFile("hdfs://master:9000/Ergasia/movie_genres.csv"). \
    filter(lambda x: (x.split(",")[1] == "Comedy")) .\
    map(lambda x: (x.split(",")[0], x.split(",")[1]))

movies = \
    spark.sparkContext.textFile("hdfs://master:9000/Ergasia/movies.csv") .\
    filter(lambda x:(x.split(",")[3] >= "1995")) .\
    map(lambda x: (x.split(",")[0],(x.split(",")[1],x.split(",")[3], x.split(",")[7])))

g = movies.join(movies_genres) .\
    map(lambda x : (x[0],x[1])) .\
    groupBy( lambda x: (x[1][1], x[1][0]))

# g_sort = g.sortBy(lambda x: (x[0][1][1]))

g.mapValues(taxhnomhsh).collect()
```

Προσπάθησα να φτιάξω μία συνάρτηση, η οποία θα γυρνάει σορταρισμένο τα ενωμένα αρχεία από τα movies και movies_genres. Αφού φιλτράρω την στήλη του έτους, από το αρχείο movies_genres, κρατάω την κατηγορία Comedy και τα id τους και από το αρχείο movies κρατάω μόνο τα στοιχεία για τις ταινίες που το έτος είναι >=1995, τα id, την δημοτικότητα. Έχω φτιάξει τον ενωμένο πίνακα, με σωστά χωρισμένα τα στοιχεία που χρειάζομαι.

Q5:

```
movies = \
    spark.sparkContext.textFile("hdfs://master:9000/Ergasia/movies.csv") .\
    filter(lambda x:(x.split(",")[3] > "0" and x.split(",")[6] > "0")) .\
    map(lambda x: (x.split(",")[1],x.split(",")[3], x.split(",")[6])))


lines = spark.sparkContext.textFile("hdfs://master:9000/Ergasia/movies.csv")
mylist = lines.filter(lambda x:(x.split(",")[3] > "0" and x.split(",")[6] > "0"))
mylist2 = mylist.map(lambda x: int(x.split(',')[6]))
print(mylist2.collect())
mylist2 = mylist.reduce( lambda x,y: int(x.split(',')[6]+y.split(',')[6]))
```

Φιλτράρω τις εγγραφές που έχουν μηδενικές τιμές για το έτος στηλών και τα έσοδα και κρατάω τα έσοδα, τα ονόματα και το έτος. Και αφού με την συνάρτηση map κρατήσω την στήλη εσόδων, προσθέτω τα έσοδα με την συνάρτηση reduce.

Dataframe (Εργασία 3):

```
if __name__ == '__main__':
    spark = SparkSession \
        .builder \
        .appName("Python Spark SQL basic example") \
        .getOrCreate()

    movies_schema = StructType([
        StructField("id", IntegerType(), True),
        StructField("name", StringType(), True),
        StructField("description", StringType(), True),
        StructField("year", IntegerType(), True),
        StructField("duration", IntegerType(), True),
        StructField("cost", LongType(), True),
        StructField("revenue", LongType(), True),
        StructField("popularity", FloatType(), True),
    ])


movies = spark.read.csv("hdfs://master:9000/Ergasia/movies.csv", schema=movies_schema) #\
new_movies = spark.read.parquet("hdfs://master:9000/Ergasia/new_movies.parquet")
#new_movies.show()
# .write.save("hdfs://master:9000/Ergasia/new_movies.parquet", format="parquet")

rating_schema = StructType([
    StructField("id_user", IntegerType(), True),
    StructField("id", IntegerType(), True),
    StructField("rating", FloatType(), True),
    StructField("time", LongType(), True),
 ])

rating = spark.read.csv("hdfs://master:9000/Ergasia/ratings.csv", schema=rating_schema) #\
new_rating = spark.read.parquet("hdfs://master:9000/Ergasia/new_rating.parquet")
#new_rating.show()
# .write.save("hdfs://master:9000/Ergasia/new_rating.parquet", format="parquet")


movies_genres_schema = StructType([
   StructField("id", IntegerType(), True),
   StructField("type", StringType(), True),
])
```

```
movies_genres = spark.read.csv("hdfs://master:9000/Ergasia/movie_genres.csv", schema = movies_genres_schema) #\
new_movies_genres = spark.read.parquet("hdfs://master:9000/Ergasia/new_movies_genres.parquet")
#new_movies_genres.show()
# .write.save("hdfs://master:9000/Ergasia/new_movies_genres.parquet", format="parquet")


departments = spark.read.csv("hdfs://master:9000/Ergasia/departmentsR.csv") #\
new_departments = spark.read.parquet("hdfs://master:9000/Ergasia/new_departments.parquet")
#new_departments.show()
# .write.save("hdfs://master:9000/Ergasia/new_departments.parquet", format="parquet")

employees = spark.read.csv("hdfs://master:9000/Ergasia/employeesR.csv") #\
new_employees = spark.read.parquet("hdfs://master:9000/Ergasia/new_employees.parquet")
#new_employees.show()
# .write.save("hdfs://master:9000/Ergasia/new_employees.parquet", format="parquet")
```

Για αρχή μετατρέπω τα αρχεία από csv σε parquet και τα ανεβάζω στο HDFS. Επίσης, για να διευκολυνθώ αλλάζω τα ονόματα από κάθε στήλη αρχείου για να καταλαβαίνω ποια στήλη είναι ποια.

Q1:

```
# Q1

new_movies.filter((new_movies.year >= 1995) & (new_movies.revenue > 0) & (new_movies.cost > 0)).show()
(new_movies.select("cost").subtract(new_movies.select("revenue"))).show()
```

Ακολουθώ τα ίδια βήματα, φιλτράρω το έτος, τα έσοδα και το κόστος και κάνω την αφαίρεση των δύο τελευταίων στηλών για τον υπολογισμό των χρημάτων που δαπανήθηκαν.

Q2:

```
#Q2

new_movies.filter(new_movies.name == "Cesare deve morire") \
.select("id").show()

print("O arithmos gia to posoi vathmologhsan einai:", new_rating.select('rating').where(new_rating.id == 96821).count())

new_rating.agg({'rating': 'sum'}).show()

print("H mesh vathmologia einai peripou: 0.00489760363")
```

Φιλτράρω για να βρω την συγκεκριμένη ταινία που θέλω. Με τις παραπάνω συναρτήσεις κρατάω τα ratings μέσω του id της ταινίας που έχω βρει ήδη. Έτσι, για να βρω τον μέσο όρο προσθέτω την συνολική βαθμολογία και την διαιρώ με το πόσοι βαθμολόγησαν.

Q3:

```
#Q3

movies1 = new_movies.filter((new_movies.year >= 1995) & (new_movies.revenue > 0))
movies1.show()
genres1 = new_movies_genres.filter(new_movies_genres.type == "Animation")
genres1.show()

joined_df = movies1.join(genres1, movies1.id == genres1.id)

joined_df.show()

sorted = joined_df.sort(joined_df.revenue.desc())

sorted.show(1)
```

Φιλτράρω ξανά το έτος και τα έσοδα καθώς και το αρχείο movies_genres για τις ταινίες Animation και με την χρήση της συνάρτησης join ενώνει τα δύο αυτά αρχεία. Έτσι, στο

ενωμένο αρχείο κάνω μία φθίνουσα συνάρτηση για να κρατήσω την πρώτη ταινία η οποία είναι και η καλύτερη ταινία κινουμένων σχεδίων από άποψη εσόδων του 1995.

Q4+Q5:

```
#Q4

genres1 = new_movies_genres.filter(new_movies_genres.type == "Comedy")
movies1 = new_movies.filter((new_movies.year >= 1995) & (new_movies.revenue > 0))

#Q5
movies1 = new_movies.filter((new_movies.year >= 0) & (new_movies.revenue > 0))
```

Έχω φιλτράρει απλά το κάθε αρχείο για να κρατήσω μόνο αυτά που χρειάζομαι.