```scala
//Task 11
// SQL based spark code compute the number of movies produced in each year
// Used Group By year to count movies for each year ignored the null year by adding condition year is not null
// Used order by to sort the output in ascending order
val sqlWay = spark.sql("""
SELECT year,count(title) as count
FROM movies_table where year is not null
GROUP BY year ORDER BY year
""")
sqlWay.show()// showing the output year and count(number of movies)
```

▸ (2) Spark Jobs

▸ sqlWay: org.apache.spark.sql.DataFrame = [year: integer, count: long]

```
+----+-----+
|year|count|
+----+-----+
|1961|    2|
|1967|    2|
|1972|   12|
|1973|    5|
|1975|    5|
|1977|   40|
|1978|   30|
|1979|   37|
|1980|   47|
|1981|   53|
|1982|  103|
|1983|  119|
|1984|  149|
|1985|  133|
|1986|  174|
|1987|  126|
|1988|  111|
|1989|  152|
```

Scala

```scala
//task 15
// Dataframe based spark code  to find the title and year for every movie that Tom Hanks acted in
val dataFrameWay3=df
.where(col("actor").equalTo("Hanks, Tom")) // filtered actor column with actor equal "Hanks, Tom"
.select("year", "title").orderBy("year")// Select the year and title to show in output sorted by year

dataFrameWay3.show(false)// showing the out put year and title show(false) to show the full title
```

(1) Spark Jobs

dataFrameWay3: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [year: integer, title: string]

```
+----+-------------------------------------------------+
|year|title                                            |
+----+-------------------------------------------------+
|1984|Bachelor Party                                   |
|1993|Sleepless in Seattle                             |
|1993|Philadelphia                                     |
|1994|Forrest Gump                                     |
|1995|Apollo 13                                        |
|1995|Toy Story                                        |
|1998|Saving Private Ryan                              |
|1998|You've Got Mail                                  |
|1999|The Green Mile                                   |
|1999|Toy Story 2                                      |
|2000|Cast Away                                        |
|2002|Catch Me If You Can                              |
|2002|Road to Perdition                                |
|2004|The Polar Express                                |
|2004|The Terminal                                     |
|2004|The Ladykillers                                  |
|2005|Magnificent Desolation: Walking on the Moon 3D|
```

Scala

```scala
//task 14
// Dataframe based spark code find the five top most actors who acted in the most number of movies
val dataFrameWay2 =df
.groupBy("actor")// Used Group By actor to count movies by each actor
.count()
.withColumnRenamed("count", "number_of_movies")// Renamed the count with number_of_movies
.sort(desc("number_of_movies"))// Used sort function with asc for sorting them in descending order based on the number of movies
.limit(5)// Used Limit 5 for showing the top 5 actors who acted in the most number of movies
dataFrameWay2.show()// showing the output actor and number_of_movies
```

(2) Spark Jobs

dataFrameWay2: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [actor: string, number_of_movies: long]

```
+-----------------+----------------+
|            actor|number_of_movies|
+-----------------+----------------+
|  Tatasciore, Fred|              38|
|     Welker, Frank|              38|
|Jackson, Samuel L.|              32|
|     Harnell, Jess|              31|
|       Damon, Matt|              27|
+-----------------+----------------+
```

dataFrameWay2: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [actor: string, number_of_movies: bigint]

Command took 1.51 seconds -- by sc26s@login.missouristate.edu at 10/12/2023, 4:53:04 PM on HW 2

```scala
//Task 13
// SQL based spark code find the five top most actors who acted in the most number of movies
// Used Group By actor to count movies by each actor
// Renamed the count(title) with number_of_movies
// Used Order by count(title) and DESC for sorting them in descending order based on the number of movies
// Used Limit 5 for showing the top 5 actors who acted in the most number of movies

val sqlWay2 = spark.sql("""
SELECT actor,count("title") as number_of_movies
FROM movies_table
GROUP BY actor ORDER BY count("title") DESC LIMIT 5
""")
sqlWay2.show()// showing the output actor and number_of_movies
```

▸ (2) Spark Jobs

▸ sqlWay2: org.apache.spark.sql.DataFrame = [actor: string, number_of_movies: long]

```
+------------------+----------------+
|             actor|number_of_movies|
+------------------+----------------+
|  Tatasciore, Fred|              38|
|     Welker, Frank|              38|
|Jackson, Samuel L.|              32|
|     Harnell, Jess|              31|
|       Damon, Matt|              27|
+------------------+----------------+
```

```
sqlWay2: org.apache.spark.sql.DataFrame = [actor: string, number_of_movies: bigint]
```

Command took 1.93 seconds -- by sc26s@login.missouristate.edu at 10/12/2023, 4:53:04 PM on HW 2

Scala

```scala
//task 12
// dataframe based spark code compute the number of movies produced in each year
// Used Group By year to count movies for each year ignored the null year by adding condition year is not null
// Used sort function with asc to show the output in ascending order
val dataFrameWay =df.where("year is not null")
.groupBy("year")
.count()
.withColumnRenamed("count", "count")
.sort(asc("year"))
dataFrameWay.show()// showing the output year and count(number of movies)
```

▸ (2) Spark Jobs

▾ dataFrameWay: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row]

```
year: integer
count: long
```

```
+----+-----+
|year|count|
+----+-----+
|1961|    2|
|1967|    2|
|1972|   12|
|1973|    5|
|1975|    5|
|1977|   40|
|1978|   30|
|1979|   37|
|1980|   47|
```