Load the data:

movies = LOAD 'hdfs://Hadoop-Master:9000/user/hduser/mergedinput.txt' USING PigStorage(',') AS (Released:chararray, imdbID:chararray, Director:chararray, Title:chararray, Actors:chararray, imdbRating:float, imdbVote:int, AmazonAvgScore:float, runTime:int, Awards:chararray, Year:int, Language:chararray, Country:chararray, AmazonTitle:chararray, AmazonID:chararray, Genre:chararray, Writer:chararray, Poster:chararray, OscarWinner:int, OscarNominated:int, OtherAwards:int);

- Top 5 movies by genre:

  -- Comedy

      filterComedy = filter movies by (Genre matches '.\*(comedy|Comedy|COMEDY).\*');

        -- Won oscars

        FilterWonOscarByComedy = filter filterComedy by (Awards matches '.\*(won|WON|Won).\*' and Awards matches '.\*(oscar|Oscars|oscars|Oscar|OSCAR|OSCARS).\*');

            -- Removing columns not required

            removingColumns = foreach FilterWonOscarByComedy generate Released,imdbID,Director,Title,Actors,imdbRating,imdbVote,runTime,Awards,Year,Language,Genre,Writer,OscarWinner,OscarNominated,OtherAwards;

                --Fetching Distinct records

                distinctComedyMovies = Distinct removingColumns;

                    -- Order it by IMDB rating

                    orderComedyMoviesImdbRating = order distinctComedyMovies by imdbRating desc;

                        -- Top 10 comedy movies by IMDB rating

                        TopTenComedyMovies = LIMIT orderComedyMoviesImdbRating 10;

  -- Horror

      filterHorror = filter movies by (Genre matches '.\*(horror|Horror|HORROR).\*');

        -- Won oscars

        FilterWonOscarByHorror = filter filterHorror by (Awards matches '.\*(won|WON|Won).\*' and Awards matches '.\*(oscar|Oscars|oscars|Oscar|OSCAR|OSCARS).\*');

            -- Removing columns not required

            removingColumns = foreach FilterWonOscarByHorror generate Released,imdbID,Director,Title,Actors,imdbRating,imdbVote,runTime,Awards,Year,Language,Genre,Writer,OscarWinner,OscarNominated,OtherAwards;

               --Fetching Distinct records

               distinctHorrorMovies = Distinct removingColumns;

                    -- Order it by IMDB rating

                    orderHorrorMoviesImdbRating = order distinctHorrorMovies by imdbRating desc;

                        -- Top 10 Horror movies by IMDB rating

                        TopTenHorrorMovies = LIMIT orderHorrorMoviesImdbRating 10;

                            --storing data:

                             STORE TopTenHorrorMovies into 'TopTenHorrorMoviesNewData' using PigStorage(',');

  -- Romance

      filterRomance = filter movies by (Genre matches '.\*(romance|Romance|ROMANCE).\*');

        -- Won oscars

        FilterWonOscarByRomance = filter filterRomance by (Awards matches '.\*(won|WON|Won).\*' and Awards matches '.\*(oscar|Oscars|oscars|Oscar|OSCAR|OSCARS).\*');

            -- Removing columns not required

            removingColumns = foreach FilterWonOscarByRomance generate Released,imdbID,Director,Title,Actors,imdbRating,imdbVote,runTime,Awards,Year,Language,Genre,Writer,OscarWinner,OscarNominated,OtherAwards;

                --Fetching Distinct records

                distinctRomanticMovies = Distinct removingColumns;

                    -- Order it by IMDB rating

                    orderRomanticMoviesImdbRating = order distinctRomanticMovies by imdbRating desc;

                        -- Top 10 Romantic movies by IMDB rating

                        TopTenRomanticMovies = LIMIT orderRomanticMoviesImdbRating 10;

                            --storing data:

                            STORE TopTenRomanticMovies into 'TopTenRomanticMoviesNewData' using PigStorage(',');

  -- Thriller Mystery

      filterMysThrill = filter movies by (Genre matches '.\*(Thriller|thriller|THRILLER|Mystery|mystery|MYSTERY).\*');

        -- Won oscars

        FilterWonOscarByMysThrill = filter filterMysThrill by (Awards matches '.\*(won|WON|Won).\*' and Awards matches '.\*(oscar|Oscars|oscars|Oscar|OSCAR|OSCARS).\*');

            -- Removing columns not required

            removingColumns = foreach FilterWonOscarByMysThrill generate Released,imdbID,Director,Title,Actors,imdbRating,imdbVote,runTime,Awards,Year,Language,Genre,Writer,OscarWinner,OscarNominated,OtherAwards;

                --Fetching Distinct records

                distinctMysThrillMovies = Distinct removingColumns;

                    -- Order it by IMDB rating

                    orderMysThrillMoviesImdbRating = order distinctMysThrillMovies by imdbRating desc;

                        -- Top 10 MysThrill movies by IMDB rating

                        TopTenMysThrillMovies = LIMIT orderMysThrillMoviesImdbRating 10;

                            --storing data:

                            STORE TopTenMysThrillMovies into 'TopTenMysThrillMoviesNewData' using PigStorage(',');

  -- Sci-Fi

      filterSciFi = filter movies by (Genre matches '.\*(Sci|sci|Sci-Fi).\*');

        -- Won oscars

        FilterWonOscarBySciFi = filter filterSciFi by (Awards matches '.\*(won|WON|Won).\*' and Awards matches '.\*(oscar|Oscars|oscars|Oscar|OSCAR|OSCARS).\*');

            -- Removing columns not required

            removingColumns = foreach FilterWonOscarBySciFi generate Released,imdbID,Director,Title,Actors,imdbRating,imdbVote,runTime,Awards,Year,Language,Genre,Writer,OscarWinner,OscarNominated,OtherAwards;

                 --Fetching Distinct records

                 distinctSciFiMovies = Distinct removingColumns;

                     -- Order it by IMDB rating

                     orderSciFiMoviesImdbRating = order distinctSciFiMovies by imdbRating desc;

                         -- Top 10 SciFi movies by IMDB rating

                         TopTenSciFiMovies = LIMIT orderSciFiMoviesImdbRating 10;

                             --storing data:

                             STORE TopTenSciFiMovies into 'TopTenSciFiMoviesNewData' using PigStorage(',');

---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Analyzing the frequency of movies and average duration of movies in a gap of 10 years:

-(2010-Present): avg: 62.168168168168165,  count:333

filterMovies\_2010 = filter movies by (Year>2010 and Year<=2020);

groupMovies\_2010 = GROUP filterMovies\_2010 ALL;

averageDuration\_2010 = FOREACH groupMovies\_2010 GENERATE AVG(filterMovies\_2010.runTime);

totalCount\_2010 = FOREACH groupMovies\_2010 GENERATE COUNT(filterMovies\_2010);

dump averageDuration\_2010;

dump totalCount\_2010;

-(2000-2010): avg: 85.81951219512194, count: 1435

filterMovies\_2000 = filter movies by (Year>2000 and Year<=2010);

groupMovies\_2000 = GROUP filterMovies\_2000 ALL;

averageDuration\_2000 = FOREACH groupMovies\_2000 GENERATE AVG(filterMovies\_2000.runTime);

totalCount\_2000 = FOREACH groupMovies\_2000 GENERATE COUNT(filterMovies\_2000);

dump averageDuration\_2000;

dump totalCount\_2000;

-(1990-2000): avg: 95.30345245305875, count: 1651

filterMovies\_1990 = filter movies by (Year>1990 and Year<=2000);

groupMovies\_1990 = GROUP filterMovies\_1990 ALL;

averageDuration\_1990 = FOREACH groupMovies\_1990 GENERATE AVG(filterMovies\_1990.runTime);

totalCount\_1990 = FOREACH groupMovies\_1990 GENERATE COUNT(filterMovies\_1990);

dump averageDuration\_1990;

dump totalCount\_1990;

-(1980-1990): avg: 99.00475624256838, count: 841

filterMovies\_1980 = filter movies by (Year>1980 and Year<=1990);

groupMovies\_1980 = GROUP filterMovies\_1980 ALL;

averageDuration\_1980 = FOREACH groupMovies\_1980 GENERATE AVG(filterMovies\_1980.runTime);

totalCount\_1980 = FOREACH groupMovies\_1980 GENERATE COUNT(filterMovies\_1980);

dump averageDuration\_1980;

dump totalCount\_1980;

-(1970-1980): 94.93983402489627

filterMovies\_1970 = filter movies by (Year>1970 and Year<=1980);

groupMovies\_1970 = GROUP filterMovies\_1970 ALL;

averageDuration\_1970 = FOREACH groupMovies\_1970 GENERATE AVG(filterMovies\_1970.runTime);

totalCount\_1970 = FOREACH groupMovies\_1970 GENERATE COUNT(filterMovies\_1970);

dump averageDuration\_1970;

dump totalCount\_1970;

-(1960-1970): 98.29305135951661

filterMovies\_1960 = filter movies by (Year>1960 and Year<=1970);

groupMovies\_1960 = GROUP filterMovies\_1960 ALL;

averageDuration\_1960 = FOREACH groupMovies\_1960 GENERATE AVG(filterMovies\_1960.runTime);

totalCount\_1960 = FOREACH groupMovies\_1960 GENERATE COUNT(filterMovies\_1960);

dump averageDuration\_1960;

dump totalCount\_1960;

-(1950-1960): 94.47107438016529

filterMovies\_1950 = filter movies by (Year>1950 and Year<=1960);

groupMovies\_1950 = GROUP filterMovies\_1950 ALL;

averageDuration\_1950 = FOREACH groupMovies\_1950 GENERATE AVG(filterMovies\_1950.runTime);

totalCount\_1950 = FOREACH groupMovies\_1950 GENERATE COUNT(filterMovies\_1950);

dump averageDuration\_1950;

dump totalCount\_1950;

-(1940-1950):

filterMovies\_1940 = filter movies by (Year>1940 and Year<=1950);

groupMovies\_1940 = GROUP filterMovies\_1940 ALL;

averageDuration\_1940 = FOREACH groupMovies\_1940 GENERATE AVG(filterMovies\_1940.runTime);

totalCount\_1940 = FOREACH groupMovies\_1940 GENERATE COUNT(filterMovies\_1940);

dump averageDuration\_1940;

dump totalCount\_1940;

-(1930-1940):

filterMovies\_1930 = filter movies by (Year>1930 and Year<=1940);

groupMovies\_1930 = GROUP filterMovies\_1930 ALL;

averageDuration\_1930 = FOREACH groupMovies\_1930 GENERATE AVG(filterMovies\_1930.runTime);

totalCount\_1930 = FOREACH groupMovies\_1930 GENERATE COUNT(filterMovies\_1930);

dump averageDuration\_1930;

dump totalCount\_1930;

-(1920-1930):

filterMovies\_1920 = filter movies by (Year>1920 and Year<=1930);

groupMovies\_1920 = GROUP filterMovies\_1920 ALL;

averageDuration\_1920 = FOREACH groupMovies\_1920 GENERATE AVG(filterMovies\_1920.runTime);

totalCount\_1920 = FOREACH groupMovies\_1920 GENERATE COUNT(filterMovies\_1920);

dump averageDuration\_1920;

dump totalCount\_1920;

-(1910-1920)

filterMovies\_1910 = filter movies by (Year>1910 and Year<=1920);

groupMovies\_1910 = GROUP filterMovies\_1910 ALL;

averageDuration\_1910 = FOREACH groupMovies\_1910 GENERATE AVG(filterMovies\_1910.runTime);

totalCount\_1910 = FOREACH groupMovies\_1910 GENERATE COUNT(filterMovies\_1910);

dump averageDuration\_1910;

dump totalCount\_1910;

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Distribution of ratings for various Genres:

-- Comedy

    filterComedy = filter movies by (Genre matches '.\*(comedy|Comedy|COMEDY).\*');

    groupfilterComedy = GROUP filterComedy ALL;

    maxRatingOfComedy = FOREACH groupfilterComedy GENERATE MAX(filterComedy.imdbRating);

    dump maxRatingOfComedy;  9.3

    minRatingOfComedy = FOREACH groupfilterComedy GENERATE MIN(filterComedy.imdbRating);

    dump minRatingOfComedy;  1.5

-- Actionexit

    filterAction = filter movies by (Genre matches '.\*(Action|action|ACTION).\*');

    groupfilterAction = GROUP filterAction ALL;

    maxRatingOfAction = FOREACH groupfilterAction GENERATE MAX(filterAction.imdbRating);

    dump maxRatingOfAction;  9.0

    minRatingOfAction = FOREACH groupfilterAction GENERATE MIN(filterAction.imdbRating);

    dump minRatingOfAction; 1.5

-- Animation

    filterAnimation = filter movies by (Genre matches '.\*(Animation|animation|ANIMATION).\*');

    groupfilterAnimation = GROUP filterAnimation ALL;

    maxRatingOfAnimation = FOREACH groupfilterAnimation GENERATE MAX(filterAnimation.imdbRating);

    dump maxRatingOfAnimation;  9.0

    minRatingOfAnimation = FOREACH groupfilterAnimation GENERATE MIN(filterAnimation.imdbRating);

    dump minRatingOfAnimation;  4.4

-- Drama

    filterDrama = filter movies by (Genre matches '.\*(Drama|drama|DRAMA).\*');

    groupfilterDrama = GROUP filterDrama ALL;

    maxRatingOfDrama = FOREACH groupfilterDrama GENERATE MAX(filterDrama.imdbRating);

    dump maxRatingOfDrama;  9.4

    minRatingOfDrama = FOREACH groupfilterDrama GENERATE MIN(filterDrama.imdbRating);

    dump minRatingOfDrama;  1.6

-- Documentary

    filterDocumentary = filter movies by (Genre matches '.\*(Documentary|documentary|DOCUMENTARY).\*');

    groupfilterDocumentary = GROUP filterDocumentary ALL;

    maxRatingOfDocumentary = FOREACH groupfilterDocumentary GENERATE MAX(filterDocumentary.imdbRating);

    dump maxRatingOfDocumentary;  9.3

    minRatingOfDocumentary = FOREACH groupfilterDocumentary GENERATE MIN(filterDocumentary.imdbRating);

    dump minRatingOfDocumentary;  2.1

-- Romance

    filterRomance = filter movies by (Genre matches '.\*(Romance|romance|ROMANCE).\*');

    groupfilterRomance = GROUP filterRomance ALL;

    maxRatingOfRomance = FOREACH groupfilterRomance GENERATE MAX(filterRomance.imdbRating);

    dump maxRatingOfRomance; 9.0

    minRatingOfRomance = FOREACH groupfilterRomance GENERATE MIN(filterRomance.imdbRating);

    dump minRatingOfRomance; 2.4

-- Horror

    filterHorror = filter movies by (Genre matches '.\*(horror|Horror|HORROR).\*');

    groupfilterHorror = GROUP filterHorror ALL;

    maxRatingOfHorror = FOREACH groupfilterHorror GENERATE MAX(filterHorror.imdbRating);

    dump maxRatingOfHorror;  9.0

    minRatingOfHorror = FOREACH groupfilterHorror GENERATE MIN(filterHorror.imdbRating);

    dump minRatingOfHorror;

-- Sci-Fi

    filterSciFi = filter movies by (Genre matches '.\*(Sci|sci|Sci-Fi|sci-fi).\*');

    groupfilterSciFi = GROUP filterSciFi ALL;

    maxRatingOfSciFi = FOREACH groupfilterSciFi GENERATE MAX(filterSciFi.imdbRating);

    dump maxRatingOfSciFi;

    minRatingOfSciFi = FOREACH groupfilterSciFi GENERATE MIN(filterSciFi.imdbRating);

    dump minRatingOfSciFi;

To read from s3:

raw = LOAD 's3n://[uw-cse-344-oregon.aws.amazon.com/btc-2010-chunk-000](http://uw-cse-344-oregon.aws.amazon.com/btc-2010-chunk-000)' USING TextLoader as (line:chararray);

movies = LOAD 's3://amazonmovies/latestClean.txt' USING PigStorage(',') AS (Released:chararray, imdbID:chararray, Director:chararray, Title:chararray, Actors:chararray, imdbRating:float, imdbVote:int, AmazonAvgScore:float, runTime:int, Awards:chararray, Year:int, Language:chararray, Country:chararray, AmazonTitle:chararray, AmazonID:chararray, Genre:chararray, Writer:chararray, Poster:chararray, OscarWinner:int, OscarNominated:int, OtherAwards:int);

<https://s3.us-east-2.amazonaws.com/amazonmovies/input/latestClean.txt>

                 groupMoviesByDirector = group movies by (Director, Genre);

                 countMoviesOfEachDirector = foreach groupMoviesByDirector generate group, COUNT(movies), OscarWinner;

                 orderTopDirectors = order countMoviesOfEachDirector by $3 desc;

                 limitingTopDirectors = LIMIT orderTopDirectors 11;

                 groupMoviesByYear = group movies by Year;

                 countMoviesOfEachYear = foreach groupMoviesByYear generate group as Year, COUNT(movies);

                 orderYears = order countMoviesOfEachYear by $1 desc;

                 totalNumberOfYears = foreach groupMoviesByYear generate group as COUNT(movies);

                 filterMoviesBwn00\_05 = filter movies by (Year>2000 and Year<=2005);

                 groupMovies = GROUP filterMoviesBwn00\_05 ALL;

                 averageDuration = FOREACH groupMovies GENERATE AVG(filterMoviesBwn00\_05.runTime);

                 groupByYear = GROUP filterMoviesYGT2000 by Year;

                 averageDur = foreach groupByYear generate group as Year, AVG(filterMoviesYGT2000.runTime);

                filterMoviesLT2000durGT120 = order (filter movies by (runTime>120 and Year<2000 and OscarWinner>0 and imdbRating>8.0)) by imdbRating desc;  521

                filterMoviesLT2000durLT120 = order (filter movies by (runTime<120 and Year<2000 and OscarWinner>0 and imdbRating>8.0)) by imdbRating desc;  3275

                filterMoviesGT2000durGT120 = order (filter movies by (runTime>120 and Year>=2000 and OscarWinner>0 and imdbRating>8.0)) by imdbRating desc);

                filterMoviesGT2000durLT120 = order (filter movies by (runTime<120 and Year>=2000 and OscarWinner>0 and imdbRating>8.0)) by imdbRating desc; 1736