1. For each starting year (beginning at 1960) going up, generate a report of years of run until 2019 called Age (going up as 1, 2, 3 etc.) and number of TV Series that started on that year and ran for so many years and also the percentage for that start\_year. Example report should look like:

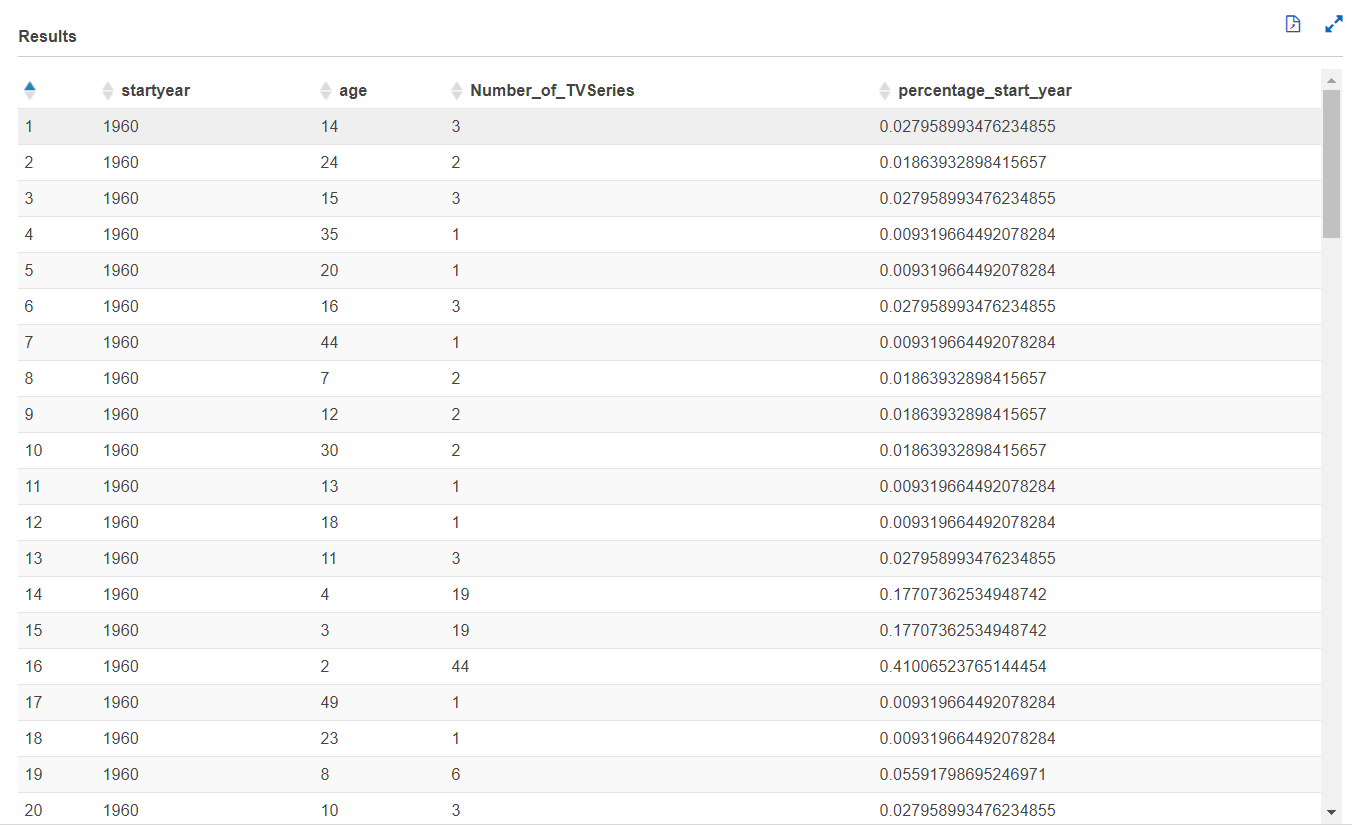
|  |  |  |  |
| --- | --- | --- | --- |
| Start\_year | Age(years) | Number\_of\_TVSeries | Percentage\_start\_year |
| 1960 | 1 | 7 | 85.72 |
| 1960 | 4 | 1 | 14.28 |
| 1965 | 2 | 5 | 71.42 |
| 2018 | 1 | 2 | 28.57 |

**Comment:** This report may be used to analyze trends in ages of TV\_series along start\_year.

**Query A**

with t2 AS (select startyear , (endyear - startyear) as age, count(startyear) as count  
from title\_basics\_new where startyear between 1960 and 2019 and endyear between 1960 and 2019 group by startyear , (endyear - startyear) order by startyear asc), t1 as (select startyear, count(startyear) as count2 from title\_basics\_new where titletype = 'tvseries' group by startyear)

select a.startyear, a.age, a.count as Number\_of\_TVSeries, cast(a.count as double)\*100/b.count2 as percentage\_start\_year from t2 a inner join t1 b on a.startyear = b.startyear order by a.startyear

[](https://user-images.githubusercontent.com/46951312/74114733-5b196500-4b71-11ea-9c58-e20181f8b44f.png)

**Dear group, totally how many rows did you get in result? Was it too much data? Is your calculation right? Look at query for t1 where we need to apply limits for startyear and endyear etc. For each startyear like 1960 total percentage should add upto 1. Will it? Why are you using titletype = ‘tvepisode’? Are we not looking for ‘tvseries’? What happens if endyear is ‘\N’?**

**Answer:** There are around 1330 rows for this query.

We applied the limits to startyear and endyear to keep them between 1960 to 2019. The calculation is based upon finding the percentage of number of tv series that started in a specific year and having a certain age. For example, in the first row, the tv series that started in 1960, and having an age of 14 years form 2.79 % of distribution amongst all the tv series that started in 1960

We made a type for selecting titletype. It should be ‘tvepisode’’. We made update to the query.

If Athena finds endyear as ‘\N’ then it does not calculate the age and keeps the age as an empty space and creates a combination of each startyear with age (empty space) and calculates Number\_of\_TvSeries and percentage\_start\_year accordingly.

1. Data inconsistency check: For each year (beginning at 1960) going up, generate a report of Number of movies that have alternative title but no original title. Example report should look like:

|  |  |
| --- | --- |
| Start\_year | Number\_of\_movies |
| 1960 | 1 |
| 1990 | 4 |
| 1995 | 2 |
| 2018 | 1 |

**Comment:** This report may be used to analyze trends in inconsistency of data along year.

## Query B

select a.startYear, count(a.startYear) as Number\_of\_movies from title\_basics\_new a inner join  
title\_akas\_new b on a.tconst = b.titleId where a.startYear >=1960 and a.titleType = 'movie' and b.isOriginalTitle = 0 group by a.startyear order by a.startyear asc;

**Dear group, don’t we need to check if it is alternative title but the same movie has no original? Is it right with the way we are checking original?**

**Answer:** Since the movies which have ’0’ value for the column ‘isOriginalTitle’, are the ones which do not have an original title but have an alternative title. That is why by keeping ‘b.isOriginalTitle = 0’ we get number of movies which have alternative title but not original title

1. Data inconsistency check: For each year (beginning at 1960) going up, generate a report of Number of multi-lingual movies that have a second language title (or localized title) but no original title. Example report should look like:

|  |  |
| --- | --- |
| Start\_year | Number\_of\_movies |
| 1960 | 1 |
| 1990 | 4 |
| 1995 | 2 |
| 2018 | 1 |

**Comment:** This report may be used to analyze trends in inconsistency of data along startYear.

## Query C

select a.startYear, count(distinct(b.titleId)) as Number\_of\_movies from title\_basics\_new a inner join  
title\_akas\_new b on a.tconst = b.titleId where a.startYear >=1960 and a.titletype = 'movie' and b.isOriginalTitle = 0 and b.types = ‘imdbDisplay’ group by a.startyear order by a.startyear asc;

**Dear group, but its types have to have ‘imdbDisplay’ and number of languages have to be more than 1 and no original?**

**Answer:** In this situation, whenever a movie has ‘0’ value in the column ‘isOriginalTitle’, it shows that the movie has an alternative title. Which means that movie has more than 1 title. This is because it has a different localized title in different locations. So whenever we see ‘is.OriginalTitle = 0’, this means that the movie has a localized title / second language title which is not original, based on the location and that proves that movie is multi lingual. Now, we have selected ‘count(distinct(b.titleId)) as Number\_of\_movies’. This distinct function will help returning precise count of the multilingual movies, that is the movies which have a localized or a second language title.

For type to be imdbDisplay, we made change to the query

1. Data inconsistency check: Verify that for each movie, endYear is always ‘\N’. For each year beginning at 1960 (if we have data) going up, generate a report of Number of movies that carry an endYear value not equal to ‘\N’. Example report should look like:

|  |  |
| --- | --- |
| Start\_year | Number\_of\_movies |
| 1960 | 1 |
| 1990 | 4 |
| 1995 | 2 |
| 2018 | 1 |

**Comment:** This report may be used to analyze trends in inconsistency of data along startYear.

## Query D

select startyear, count(startyear) as Number\_of\_movies from title\_basics\_new where startyear>= 1960 and titletype = 'movie' and endyear is not null group by startyear order by startyear asc;

**Dear group, endyear is not ‘\N’ otherwise query is right.**

**Answer:** Athena does not accept ‘\N’ but in the query it recognized ‘is not null’ statement for the values which were empty spaces (or ‘\N’)

1. Limits check: For each year (beginning at 1960) going up, generate a report of minimum & maximum runtime for different types of movies such as ‘Documentary’, ‘Short’, ‘Animation’ (these are in title\_akas\_tsv.genres) **only**. Example report should look like:

|  |  |  |  |
| --- | --- | --- | --- |
| Start\_year | Genre | Minimum\_runtime (min) | Maximum\_runtime (min) |
| 1960 | ‘Documentary’ | 4 | 7 |
| 1990 | ‘Animation’ | 40 | 64 |
| 1990 | ‘Short’ | 50 | 52 |
| 1990 | ‘Documentary’ | 3 | 5 |
| 1995 | ‘Documentary’ | 4 | 6 |
| 2018 | ‘Documentary’ | 5 | 8 |

**Comment:** This report may be used to analyze trends in run-time along startYear.

## Query E

select startyear, genre\_1, min(runtimeminutes) as min\_runtime, max(runtimeminutes) as max\_runtime from title\_basics\_new CROSS JOIN UNNEST(genre) AS t(genre\_1) where startyear >= 1960 and titletype = 'movie' group by startyear, genre\_1 order by startyear asc;

**Dear group, don’t we need to filter only genres containing ‘Documentation’ or ‘Short’ or ‘Animation’ in addition?**

**Answer:** As per the business plan, we thought that all the genres present in the data must be considered and the ‘Documentary’, ‘Short’, ‘Animation’ were the examples given in the question (e). But if you want us to do the query just for only these 3 genres, we can do that.

1. Consolidated Distribution over rating: We would like to plot the density of last five years’ (2015-2019) all **movies** against audience average rating. First calculate total count of movies over last five years. Report the density of number of movies (made over last five years) for each rating (after converting rating to nearest integer) and the corresponding rating (1-10). Example report should look like:

|  |  |
| --- | --- |
| Average\_Rating | Density\_Total\_no\_of\_movies\_in\_last\_5\_yrs |
| 1 | 0.05 |
| 2 | 0.10 |
| 4 | 0.20 |
| 6 | 0.40 |
| 7 | 0.20 |
| 8 | 0.05 |

**Comment:** The sum of densities is 1.0. Ignore movies made more than 5 years ago. This report may be extended to include the year as another attribute.

## Query F

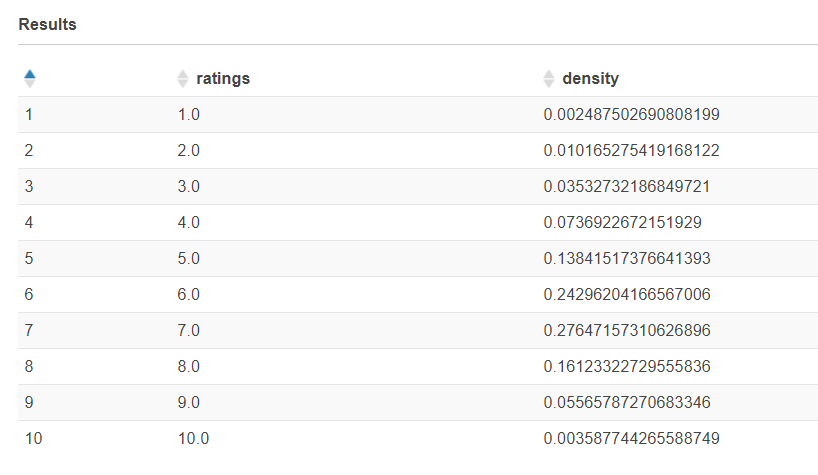
1. Consolidated Distribution over rating: We would like to plot the density of last five years’ (2015-2019) all **movies** against audience average rating. First calculate total count of movies over last five years. Report the density of number of movies (made over last five years) for each rating (afterconverting rating to nearest integer) and the corresponding rating (1-10). Example report should look like:

|  |  |
| --- | --- |
| Average\_Rating | Density\_Total\_no\_of\_movies\_in\_last\_5\_yrs |
| 1 | 0.05 |
| 2 | 0.10 |
| 4 | 0.20 |
| 6 | 0.40 |
| 7 | 0.20 |
| 8 | 0.05 |

**Comment:** The sum of densities is 1.0. Ignore movies made more than 5 years ago. This report may be extended to include the year as another attribute.

create table if not exists ratings\_in\_each\_year\_1 as select round(title\_ratings.averagerating,0) as ratings, count(title\_ratings.averagerating) as count from title\_ratings inner join title\_basics\_new on title\_ratings.tconst= title\_basics\_new.tconst where title\_basics\_new.titletype = 'movie' and title\_basics\_new.startyear >=2015 and title\_basics\_new.startyear<=2019 group by round(title\_ratings.averagerating,0) order by round(title\_ratings.averagerating,0) asc;

SELECT a.ratings, cast(a.count as double)/b.sum\_count as density FROM ratings\_in\_each\_year\_1 a CROSS JOIN ( SELECT SUM(count) as sum\_count FROM ratings\_in\_each\_year\_1 ) b

[](https://user-images.githubusercontent.com/46951312/74116098-9d45a500-4b77-11ea-8895-59ffdcc48075.png)

**Dear group, looks fine to me. Probably instead of saying “group by** **round(title\_ratings.averagerating,0) order by round(title\_ratings.averagerating,0) asc”, you could have made it shorter by saying “group by ratings order by ratings” . But this query is good.**

**Answer:** We kept the query like this because we had values of ratings in decimals (example 1.1, 1.2, 1.3). For example, in the data set there were many different values for integer 1 followed by a decimal, further followed by different figures (eg 1.1, 1.2, 1.3). So we had to group it by round(title\_ratings.averagerating,0) because we wanted to select rounded off values of ratings, in particular groups (from 1 to 10) as per the requirement.

## Query G

1. Consolidated Distribution over numVotes: We would like to plot the density of last five years’ (2015-2019) all **movies** against ranges of numVotes. See numVotes and if it is in like 100,000 s, then each range should be 100,000s (like 0 - 100,000; 100,001 – 200,000; 200,001-300,000; etc.). First calculate total count of movies over last five years. Report the density of number of movies (made over last five years) for each range of numVotes and the corresponding range. Example report should look like:

|  |  |
| --- | --- |
| Range\_of\_numVotes | Density\_Total\_no\_of\_movies\_in\_last\_5\_yrs |
| 100000 | 0.05 |
| 200000 | 0.10 |
| 300000 | 0.15 |
| 400000 | 0.40 |
| 500000 | 0.20 |
| 600000 | 0.05 |
| 700000 | 0.05 |

**Comment:** The sum of densities is 1.0. Ignore movies made more than 5 years ago. This report may be extended to include the year as another attribute.

with cte1 as (select case when numvotes >=1 and numvotes <=100000 then 100000 when numvotes >100000 and numvotes<=200000 then 200000 when numvotes >200000 and numvotes<=300000 then 300000 when numvotes >300000 and numvotes<=400000 then 400000 when numvotes >400000 and numvotes<=500000 then 500000 when numvotes >500000 and numvotes<=600000 then 600000 when numvotes >600000 and numvotes<=700000 then 700000 when numvotes >700000 and numvotes<=800000 then 800000 when numvotes >800000 and numvotes<=900000 then 900000 when numvotes >900000 and numvotes<=1000000 then 1000000 when numvotes >1000000 and numvotes<=1100000 then 1100000 when numvotes >1100000 and numvotes<=1200000 then 1200000 when numvotes >1200000 and numvotes<=1300000 then 1300000 when numvotes >1300000 and numvotes<=1400000 then 1400000 when numvotes >1400000 and numvotes<=1500000 then 1500000 when numvotes >1500000 and numvotes<=1600000 then 1600000 when numvotes >1600000 and numvotes<=1700000 then 1700000 when numvotes >1700000 and numvotes<=1800000 then 1800000 when numvotes >1800000 and numvotes<=1900000 then 1900000 when numvotes >1900000 and numvotes<=2000000 then 2000000 when numvotes >2000000 and numvotes<=2100000 then 2100000 when numvotes >2100000 and numvotes<=2200000 then 2200000 else NULL end as num, t2.primarytitle, count(t2.primarytitle) over () as cnt from title\_ratings t1 inner join title\_basics\_new t2 on t1.tconst = t2.tconst where t2.startyear >=2015 and t2.startyear <= 2019 and t2.titletype = 'movie') , grouped\_num\_1 as (

select distinct t1.num as range, count(t1.primarytitle) over (partition by t1.num) as num\_count, cast(t1.cnt as double) as tot from cte1 t1 )

select t.range, (t.num\_count/t.tot) as density from grouped\_num\_1 t order by t.range

**Dear group, probably it works. Notice, you are doing “count(x) over (partition by t1.num)” kind of thing instead of “count(x) group by”. This query may be fine.**

**Answer:** We performed partitioning over t1.num because we wanted to calculate the count of primary titles for grouped ranges (or num) but we did not want to use group by clause at the end of the statement, because we don’t need a group by to calculate distinct t1.num and t1.cnt.

## Query H

1. Consolidated Distribution over run-time: We would like to plot the density of last five years’ (2015-2019) all **movies** against runtimeMinutes. First calculate total count of movies over last five years. Report the density of number of movies (made over last five years) for each runtime (minutes) and the corresponding runtime. Example report should look like:

|  |  |
| --- | --- |
| runtimeMinutes | Density\_Total\_no\_of\_movies\_in\_last\_5\_yrs |
| 30 | 0.05 |
| 60 | 0.10 |
| 95 | 0.20 |
| 130 | 0.40 |
| 140 | 0.20 |
| 160 | 0.05 |

**Comment:** The sum of densities is 1.0. This report may be extended to include the year as another attribute.

create table if not exists runtimeminutes\_in\_each\_year as select runtimeminutes, count(runtimeminutes) as count\_of\_movies from title\_basics\_new where titletype = 'movie' and startyear >=2015 and startyear <= 2019 group by runtimeminutes order by runtimeminutes asc;

select runtimeminutes, cast(a.count\_of\_movies as double)/b.total\_sum\_of\_movies as density from runtimeminutes\_in\_each\_year a CROSS JOIN ( select sum(count\_of\_movies) as total\_sum\_of\_movies from runtimeminutes\_in\_each\_year ) b

**Dear group, looks absolutely fine to me.**

**Answer:** Thanks

## Query I

1. Consolidated Distribution over startYear: We would like to plot the density of number of all **movies** made in each year versus the startYear (for last 20 years say 2000-2019 and ignore others). First calculate total count of movies over each startYear (ignoring bdefore 2000 or 2020). Report the density of number of movies for each startYear and the corresponding startYear. Example report should look like:

|  |  |
| --- | --- |
| startYear | Density\_Total\_no\_of\_movies\_in\_last\_5\_yrs |
| 2000 | 0.05 |
| 2002 | 0.10 |
| 2007 | 0.20 |
| 2010 | 0.20 |
| 2016 | 0.20 |
| 2019 | 0.25 |

**Comment:** The sum of densities is 1.0.

create table if not exists movies\_in\_each\_year as select startyear, count(startyear) as count\_of\_movies from title\_basics\_new where titletype = 'movie' and startyear >=2000 and startyear <= 2019 group by startyear order by startyear asc;

select startyear, cast(a.count\_of\_movies as double)/b.total\_sum\_of\_movies as density from movies\_in\_each\_year a CROSS JOIN ( select sum(count\_of\_movies) as total\_sum\_of\_movies from movies\_in\_each\_year ) b

**Dear group, looks absolutely fine to me.**

**Answer:** Thanks

## Query J

1. Top 10 versatile movies: List those movies with 10 largest sets of genres starting from the top largest and going down by size of genres array with average rating above 5. Display the rank, movie id and its title and the genres. Example report should look like:

|  |  |  |  |
| --- | --- | --- | --- |
| Rank | Movie\_title\_id | Movie\_title | Genres |
| 1 | CK00102 | Avatar | Fiction,Adventure,Romance, Family |
| 2 | CK00107 | Beyond The Sea | ScinceFiction, Adventure, Romance |
| 3 | CK00111 | SlumDog Millionaire | Romance, Adventure, Mystery |
| 4 | CK00101 | Titanic | Romance, Adventure, Exuberance |
| 5…9 | CK00103 ……… CK00104 | The Prestige…….. | …… |
| 10 | CK00106 | Spirited Away | Adventure, Mystery |

**Comment:** If there is conflict with more than 10, take any ten.

select row\_number() over (order by cardinality(genre) desc) Rank, Movie\_title\_id, Movie\_title, genre from (select a.tconst as Movie\_title\_id, a.primarytitle as Movie\_title, a.genre from title\_basics\_new a inner join title\_ratings b on a.tconst = b.tconst where b.averagerating > 5.0 and a.titletype = 'movie')

**Dear group, probably it works. Notice, you are doing “over(order by …) ” kind of thing instead of “group by …. Order by …”. This query may be fine.**

**Answer:** Thanks, Apoorba. The Over clause helps grouping by or ordering by a particular column instead of the whole output result. This helped in getting desired output.

## Query K

1. Top 10 co-directed movies: List those movies with 10 largest sets of co-directors starting from the top largest and going down by size of co-director array with average rating above 5. Display the rank, movie id and its popular title and the directors. Example report should look like:

|  |  |  |  |
| --- | --- | --- | --- |
| Rank | Movie\_title\_id | Movie\_title | Directors |
| 1 | CK00102 | Avatar | X, Y, Z, M |
| 2 | CK00107 | Beyond The Sea | A, B, C, D |
| 3 | CK00111 | SlumDog Millionaire | P, Q, R |
| 4 | CK00102 | Titanic | L, M, N |
| 5…9 | CK00102 ……… CK00102 | The Prestige…….. | …… |
| 10 | CK00102 | Spirited Away | C, T |

**Comment:** If there is conflict with more than 10, take any ten.

select row\_number() over (order by cardinality(directors) desc) Rank, Movie\_title\_id, Movie\_title, directors from (select a.tconst as Movie\_title\_id, b.primarytitle as Movie\_title, a.directors from title\_crew\_new a inner join title\_basics\_new b on a.tconst = b.tconst inner join title\_ratings c on a.tconst = c.tconst where c.averagerating > 5.0 and b.titletype = 'movie')

**Dear group, probably it works. Notice, you are doing “over(order by …) ” kind of thing instead of “group by …. Order by …”. This query may be fine.**

**Answer:** Thanks