**Query A**

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 seriesAndAge AS(

SELECT startYear,

(endYear - startYear) AS Age,

COUNT(\*) AS countSY

FROM title\_basics\_arr

WHERE startYear IS NOT NULL AND startYear >= 1960

AND titleType = 'tvSeries'

GROUP BY startYear, (endYear - startYear)

ORDER BY startYear limit 100),

totalCountEachYear AS(

SELECT startYear,

COUNT(\*) AS totalCount

FROM title\_basics\_arr

WHERE startYear IS NOT NULL AND startYear >= 1960

AND titleType = 'tvSeries'

GROUP BY startYear

ORDER BY startYear)

SELECT a.startYear,

a.Age,

a.countSY AS Number\_of\_TVSeries,

CAST(a.countSY AS DOUBLE)\*100/b.totalCount AS Percentage\_start\_year

FROM seriesAndAge a

INNER JOIN totalCountEachYear b ON a.startYear = b.startYear

ORDER BY a.startyear, a.Age

LIMIT 10;

**Dear group, almost there. But don’t we need to stay away from picking 2020 stuff? What happens if endyear is ‘\N’? We need to force it to 2019. So we need a case clause to handle something like that and then you can calculate endyear – startyear.**

## Query B

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.

SELECT startYear,

COUNT(\*) AS Number\_of\_movies

FROM title\_basics\_arr

WHERE startYear IS NOT NULL AND startYear >= 1960

AND titleType = 'movie'

AND (originalTitle IS NULL OR LENGTH(originalTitle)=0)

AND (LENGTH(primaryTitle)>0 OR primaryTitle IS NOT NULL)

GROUP BY startYear

ORDER BY startYear;

**Dear group, don’t we also need to go to title\_akas\_tsv table and make sure that same titleId has an alternative title?**

## Query C

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.

SELECT startYear,

COUNT(\*) AS Number\_of\_movies

FROM title\_basics\_arr

WHERE tConst IN ( SELECT titleID

FROM aka\_titles\_arr

WHERE LENGTH(title)>0 AND isOriginalTitle=0)

AND startYear >= 1960

AND titleType = 'movie'

GROUP BY startYear

ORDER BY startYear;

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

## Query D

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.

SELECT startYear,

COUNT(startYear) AS Number\_of\_movies

FROM title\_basics\_arr

WHERE endYear IS NOT NULL

AND startYear >= 1960

AND titleType = 'movie'

GROUP BY startYear

ORDER BY startYear;

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

## Query E

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.

WITH arra AS (

SELECT startYear,

genres1,

genres2,

runtimeminutes

FROM title\_basics\_arr

CROSS JOIN UNNEST(genresa) WITH ORDINALITY AS t(genres1, genres2)

WHERE startYear >= 1960 AND runtimeminutes IS NOT NULL

LIMIT 100

)

SELECT DISTINCT startYear,

genres1,

COUNT(genres1) OVER(PARTITION BY arra.genres1) AS number\_of\_movies,

MIN(runtimeminutes) OVER(PARTITION BY arra.genres1) AS minRT,

MAX(runtimeminutes) OVER(PARTITION BY arra.genres1) AS maxRT

FROM arra

ORDER BY startYear

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? You are saying “PARTITION BY arra.genres1” but don’t we need “PARTITION BY array.startYear,arra.genres1”? Do we need genres2?**

## 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 (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.

with cte1 as(

select a.startyear as year,

a.primarytitle as movie,

round(b.averagerating) as rating,

count(a.primarytitle) over () as cnt

from title\_basics\_arr a

inner join title\_ratings\_arr b on a.tconst=b.tconst

where a.titletype = 'movie'

and a.startyear >= 2015

order by a.startyear

),

cte2 as

(

select distinct cte1.rating as rating,

count(cte1.movie) over (partition by cte1.rating order by cte1.rating) as rating\_1,

cast(cte1.cnt as double) as tot

from cte1

)

select cte2.rating,

round((cte2.rating\_1/cte2.tot),4) as den

from cte2

order by cte2.rating

**Dear group, how but putting startyear <= 2019 also? You are almost there.**

## 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 aux1 as (select (case when numvotes >= 1 and numvotes <= 1000 then 1000 when numvotes > 1000 and numvotes <= 2000 then 2000

when numvotes > 2000 and numvotes <= 3000 then 3000

when numvotes > 3000 and numvotes <= 4000 then 4000

when numvotes > 4000 and numvotes <= 5000 then 5000

when numvotes > 5000 and numvotes <= 6000 then 6000

when numvotes > 6000 and numvotes <= 7000 then 7000

when numvotes > 7000 and numvotes <= 8000 then 8000

when numvotes > 8000 and numvotes <= 9000 then 9000

when numvotes > 9000 and numvotes <= 10000 then 10000 else NULL end) as num,

t1.numvotes as numvotes,

count(t1.numvotes) over() as tot

from title\_ratings\_arr t1 inner join title\_basics\_arr t2 on t1.tconst = t2.tconst where t2.startyear >=2015),

aux2 as (select distinct aux1.num as range,

cast(aux1.tot as double) as totalVotes,

count(aux1.numvotes) over(partition by aux1.num) as nVotes

from aux1)

select range, round(nVotes/totalVotes,5) from aux2

where range is not null

order by range

**Dear group, is Density of number of movies calculation right? Why nVotes/totalVotes? Why range of 1000 and 2000 etc? Do you have right data? How but startyear <= 2019? Does not look right.**

## 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.

WITH cte1 AS(

SELECT startyear AS year,

primarytitle as movie,

runtimeminutes as runtime,

COUNT(primarytitle) OVER() AS cnt

FROM title\_basics\_arr

WHERE titletype = 'movie'

AND startyear >= 2015

ORDER BY startyear),

cte2 AS (

SELECT DISTINCT cte1.runtime AS rt,

CAST(cte1.cnt as DOUBLE) AS total,

COUNT(cte1.runtime) OVER(PARTITION BY cte1.runtime) AS runtimeSum

FROM cte1

ORDER BY rt)

SELECT cte2.rt AS runtimeMinutes,

ROUND((cte2.runtimeSum/cte2.total),5) AS Density\_Total\_no\_of\_movies\_in\_last\_5\_yrs

FROM cte2

ORDER BY runtimeMinutes;

**Dear group, almost there. 2 comments: a) on cte1 query where, use startyear between 2015 and 2019 2) I think we should convert a range of runtimeminutes like 25 – 34 into standard 30, 35 – 44 into 40 etc. and then use this new range values. Good work.**

## 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.

WITH cte1 AS(SELECT startyear AS year,

primarytitle AS movie,

COUNT(\*) OVER() AS cnt

FROM title\_basics\_arr

WHERE titletype = 'movie'

AND startyear BETWEEN 2000 AND 2019

ORDER BY startyear),

cte2 AS(SELECT DISCTINCT cte1.year AS startYear,

CAST(cte1.cnt AS DOUBLE) AS total,

COUNT(cte1.year) OVER(PARTITION BY cte1.year) AS yearSum

FROM cte1

ORDER BY startYear)

SELECT cte2.startYear AS startYear,

ROUND((cte2.yearSum/cte2.total),5) AS Density\_Total\_no\_of\_movies\_in\_last\_20\_yrs

FROM cte2

ORDER BY startYear;

**Dear group, what is “DISCTINCT “? That proves this query failed but you are showing results. Is that right?**

## 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 | ScienceFiction, 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 tconst AS movie\_title\_ID,

primaryTitle AS movie\_title,

genresa AS Genres

FROM title\_basics\_arr

WHERE titleType = 'movie' --AND CARDINALITY(genresa) >=5

ORDER BY CARDINALITY(genresa) DESC

LIMIT 10

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, somewhat okay. Don’t we need averageRating > 5 and row number? Neither fully right nor fully wrong; can be improved.**

## 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 a.tconst,

b.primarytitle,

a.directorsa

FROM title\_crew\_arr a

INNER JOIN title\_basics b ON a.tconst=b.tconst

ORDER BY CARDINALITY(a.directorsa) DESC

LIMIT 10

**Dear group, somewhat okay. Don’t we need averageRating > 5 and titleType = ‘movie’? We need a rank or row number also? Needs a little improvement.**