```sql
SELECT * FROM netflix_data;

-- TASK 1. HANDLING FOREIGN CHARACTERS
-- We have removed foreign characters by updating datatype of title to nvarchar

-- TASK 2. REMOVE DUPLICATES
SELECT show_id, COUNT(*)
FROM netflix_data
GROUP BY show_id --This will group records by show_id
HAVING COUNT(*)>1 --If any show_id has more than one record, the result will appear below.

--Since there is no duplicates in show_id, we will consider it as a primary key and update the table again.

-- TASK 3. Now we will be checking for duplicates in title (IMPORTANT)
SELECT * FROM netflix_data
WHERE CONCAT(UPPER(title), type) in(    --we need to check the whole data for duplicates, so we will concat title and type as there can only be one coulumn in 'in' clause.
SELECT CONCAT(UPPER(title), type) --checking duplicate for title and considering type too as some titles are same but the type is different
FROM netflix_data
GROUP BY CONCAT(UPPER(title), type)
HAVING COUNT(*)>1)
ORDER BY title

WITH cte as( --a temporary table is created named cte
SELECT *
, ROW_NUMBER() OVER (PARTITION BY title,type ORDER BY show_id) as rn --The data is grouped by title and type. The row number is assigned based on show_id order.
FROM netflix_data
)
SELECT * FROM cte
WHERE rn=1  --this displays the results from temporary table after removing duplicates

--TASK 4. NEW TABLE FOR LISTED_IN, DIRECTOR, COUNTRY, CAST (Why?, Because they have multiple values in it which becomes difficult when doing the analysis)
--for director
SELECT show_id, TRIM(VALUE) AS director
INTO netflix_directors
FROM netflix_data
CROSS APPLY STRING_SPLIT(director,',') --this will split the director first name and last name
SELECT * FROM netflix_directors --stores the data in a table

--for country
SELECT show_id, TRIM(VALUE) AS country --We are also adding show_id because its
```

```
  primary key and will be used for JOINS
INTO netflix_country
FROM netflix_data
CROSS APPLY STRING_SPLIT(country,',')
SELECT * FROM netflix_country

--for cast
SELECT show_id, TRIM(VALUE) AS cast
INTO netflix_cast
FROM netflix_data
CROSS APPLY STRING_SPLIT(cast,',')
SELECT * FROM netflix_cast

--for genre
SELECT show_id, TRIM(VALUE) AS genre
INTO netflix_genre
FROM netflix_data
CROSS APPLY STRING_SPLIT(listed_in,',')

SELECT * FROM netflix_genre

--TASK 5. Populate date_added as date. To do this, we need to cast the column by
  using 'cast(date_added as date) as date_added'

-- TASK 6. Now we will find the Null values in our dataset through Python
--show_id            0
--type               0
--title              0
--director        2634
--cast             825
--country          831
--date_added        10
--release_year       0
--rating             4
--duration           3
--listed_in          0
--description        0
--dtype: int64

-- TASK 7. Populate missing values in Country (Basically this means that we need to
   replace null values by country by mapping the country table)
INSERT INTO netflix_country -- Once the below is completed, all the show_id that
  had null value country will be added with country name after mapping
SELECT show_id, m.country --here m.country will represent the values of country
  from the inner join that is used for mapping
FROM netflix_data nd
INNER JOIN(
SELECT director, country
FROM netflix_country nc
```

```
INNER JOIN netflix_directors AS nd ON nc.show_id=nd.show_id
GROUP BY director, country)m on nd.director=m.director -- nd.director is the Null values which will be matched with m.director and replace the NULL values with country name in inner join
WHERE nd.country is NULL --so we have 831 rows where country is NULL
                      --(So basically, we need to populate all show_id whose country is NULL in netflix_country and should not be NULL)


-- TASK 8. Check if duration is null and fill in null values
SELECT *
FROM netflix_data
WHERE duration IS NULL --the results show that there are 3 Null values in duration but also the rating has duration values.

WITH cte as( --a temporary table is created named cte
SELECT *
, ROW_NUMBER() OVER (PARTITION BY title,type ORDER BY show_id) as rn --The data is grouped by title and type. The row number is assigned based on show_id order.
FROM netflix_data
)
SELECT show_id, type, title, cast(date_added as date) as date_added, release_year,
rating, CASE WHEN duration is NULL THEN rating ELSE duration END AS duration, description --This command has all columns apart from splitted tables and
INTO netflix
FROM cte -- All the null values of duration is replaced by rating (because when we checked, rating was in minutes and duration was NULL)
SELECT * FROM netflix  --THIS IS THE CLEANED TABLE NOW. THE SPLITTED TABLES LIKE COUNTRY, CAST, DIRECTORS, AND GENRE CAN BE MERGED AND USED FOR DATA ANALYSIS.



--NETFLIX DATA ANALYSIS

-- QUESTION 1. FOR EACH DIRECTOR, COUNT THE NUMBER OF MOVIES AND SHOWS CREATED BY THEM IN SEPARATE COLUMNS
--   FOR DIRECTORS WHO HAVE CREATED MOVIES AND TV SHOWS BOTH

SELECT COUNT(distinct n.type) AS distinct_type, nd.director
, COUNT(distinct case when n.type = 'Movie' then n.show_id end) as no_of_movies
, COUNT(distinct case when n.type = 'TV Show' then n.show_id end) as no_of_shows
FROM netflix n
INNER JOIN netflix_directors nd ON n.show_id=nd.show_id
GROUP BY nd.director
HAVING COUNT(distinct n.type)>1
ORDER BY distinct_type DESC

-- QUESTION 2. WHICH COUNTRY HAS HIGHEST NUMBER OF COMEDY MOVIES (MAKE SURE WE NEED TO ONLY FIND MOVIES AND NOT tv SHOWS) TIP: THERE WILL BE 2 JOINS OF NETFLIX, ONE
```

```sql
   WITH GENRE AND ONE WITH COUNTRY
SELECT count (distinct ng.show_id) as no_of_movies, nc.country, n.type, ng.genre
FROM netflix_genre ng
INNER JOIN netflix_country nc ON nc.show_id=ng.show_id --this will join the country
   and genre tables to find out countries with highest comedies
inner join netflix n ON n.show_id = ng.show_id --this will join netflix and genre
  table to filter type only to movies and not comedy shows
WHERE ng.genre = 'Comedies' AND n.type='Movie'
GROUP BY nc.country, n.type, ng.genre
ORDER BY no_of_movies DESC --just to confirm the type and genre, I have added the
  columns for them too


-- QUESTION 3. FOR EACH YEAR (as per date added to netflix), WHICH DIRECTOR HAS
  MAXIMUM NUMBER OF MOVIES RELEASED? (We need to sort it through date added and not
   through release date)
with cte as (  --cte will have the details of the following data extracted from
  below query (it will have year, director, count of show ID as number of movies)
SELECT YEAR(date_added) AS date_year, nd.director, COUNT(n.show_id) AS no_of_movies
   -- converted date_added in year, counting number of shows and getting director
  names from director table
FROM netflix n
INNER JOIN netflix_directors nd on nd.show_id=n.show_id --joining the director and
  main table on show ID to take matching records
WHERE type='Movie' -- where type = Movie and not TV show
GROUP BY nd.director, YEAR(date_added)
)

, cte2 as(
SELECT *
, ROW_NUMBER() OVER (PARTITION BY date_year ORDER BY no_of_movies desc, director)
  as rn
FROM cte
)
SELECT * FROM cte2
WHERE rn=1
ORDER BY no_of_movies DESC

-- QUESTION 4. WHAT IS THE AVERAGE DURATION OF MOVIE IN EACH GENRE
SELECT ng.genre, AVG (Cast(replace(duration, ' min','') as INT)) as
  avg_duration_int --since we need to find the average duration, initially, we will
   cast the duration to int
FROM netflix n
    -- Also, we will replace min by empty and then calculate the average
inner join netflix_genre ng on ng.show_id=n.show_id
    -- Also, we need to join netflix and genre table to get matching records
WHERE type='Movie'
    --For type=Movie
GROUP BY ng.genre
```