Assumptions:

1. If “cats” is NULL, it is changed to 'Category not defined'
2. ‘children’s programs’ category is changed to ‘Kids’
3. 'Documentation' is changed to 'Documentaries'
4. 'Serien' is changed to 'Series'
5. Duration is converted from seconds to hours
6. All the duplicate rows are removed. So final SQL and dashboard is based on 168864 unique rows

Task 1: Write SQL queries

```
--replace missing categories and convert secs to hours
create or replace table `macro-magpie-365619.Zattoo.raw_watch_sessions_cleanV1` as
  select user_id, regionid,
  case
    when cats is NULL then 'Category not defined'
    when cats = 'Documentation' then 'Documentaries'
    when cats = 'Serien' then 'Series'
    when cats = "children's program" then 'Kids'
    else cats
  end as Category, channel, device, unique_sess_id, round((duration/3600),4) as
durationHours
  from `macro-magpie-365619.Zattoo.raw_watch_sessions` ;

--remove duplicate data
create or replace table `macro-magpie-365619.Zattoo.raw_watch_sessions_clean` as
  select distinct user_id, regionid, Category, channel, device, unique_sess_id,
durationHours
  from `macro-magpie-365619.Zattoo.raw_watch_sessions_cleanV1` ;

--Which channel is most popular per region?
create or replace table `macro-magpie-365619.Zattoo.rank_table` as
  SELECT regionid,
  channel,
  count(distinct unique_sess_id) based_on_sess,
  sum(durationHours) based_on_duration,
  count(distinct user_id) based_on_user_count,
  ROW_NUMBER() OVER(PARTITION BY regionid ORDER BY count(distinct unique_sess_id)
desc) AS row_num_1,
  ROW_NUMBER() OVER(PARTITION BY regionid ORDER BY sum(durationHours) desc) AS
row_num_2,
  ROW_NUMBER() OVER(PARTITION BY regionid ORDER BY count(distinct user_id) desc) AS
row_num_3
  FROM `macro-magpie-365619.Zattoo.raw_watch_sessions_clean`
  group by regionid, channel
  order by regionid;

create or replace table `macro-magpie-365619.Zattoo.popular_based_on_sess` as
```

```
  select regionid, channel as famous_based_on_sess from `macro-magpie-
365619.Zattoo.rank_table` where row_num_1 = 1;

create or replace table `macro-magpie-365619.Zattoo.popular_based_on_duration` as
  select regionid, channel as famous_based_on_duration from `macro-magpie-
365619.Zattoo.rank_table` where row_num_2 = 1;

create or replace table `macro-magpie-365619.Zattoo.popular_based_on_user_count` as
  select regionid, channel as famous_based_on_user_count from `macro-magpie-
365619.Zattoo.rank_table` where row_num_3 = 1;

create or replace table `macro-magpie-365619.Zattoo.popular_channel_region` as
  select s.regionid, famous_based_on_sess, famous_based_on_duration,
famous_based_on_user_count
  from `macro-magpie-365619.Zattoo.popular_based_on_sess` s
  inner join `macro-magpie-365619.Zattoo.popular_based_on_duration` d on
s.regionid=d.regionid
  inner join `macro-magpie-365619.Zattoo.popular_based_on_user_count` u on
s.regionid=u.regionid;

select * from `macro-magpie-365619.Zattoo.popular_channel_region`;

--What is the device share for this channel?
create or replace table `macro-magpie-365619.Zattoo.chan_dev_sess` as
  select p.regionid, p.famous_based_on_sess, rs.device, count(distinct
unique_sess_id) sessioncounts,
    round(count(distinct unique_sess_id)*100/ (select count(distinct
unique_sess_id) from `macro-magpie-365619.Zattoo.raw_watch_sessions_clean` rs
  where p.regionid=rs.regionid and p.famous_based_on_sess=rs.channel
  group by p.regionid, p.famous_based_on_sess),2) as Percentage_share
  from `macro-magpie-365619.Zattoo.popular_based_on_sess` p inner join `macro-
magpie-365619.Zattoo.raw_watch_sessions_clean` rs
  on p.regionid=rs.regionid and p.famous_based_on_sess=rs.channel
  group by p.regionid, p.famous_based_on_sess, rs.device
  order by 1 asc;

select * from `macro-magpie-365619.Zattoo.chan_dev_sess`;

--What is the maximum duration per channel per user id?
create or replace table `macro-magpie-365619.Zattoo.dur_chan_user_table` as
  select user_id, channel, max(durationHours) as
maximum_duration_per_channel_per_user_id,
  ROW_NUMBER() OVER(PARTITION BY user_id ORDER BY max(durationHours) desc) AS
row_num_1,
  from `macro-magpie-365619.Zattoo.raw_watch_sessions_cleanV1`
  group by user_id, channel
  order by 1 asc, max(durationHours) desc;

select * from `macro-magpie-365619.Zattoo.dur_chan_user_table` order by 1;
```

```sql
--What are the top 3 most popular channels per category (cats)?
create or replace table `macro-magpie-365619.Zattoo.rank_table_2` as
  SELECT Category, channel,
  count(distinct unique_sess_id) based_on_sess,
  sum(durationHours) based_on_duration,
  count(distinct user_id) based_on_user_count,
  ROW_NUMBER() OVER(PARTITION BY Category ORDER BY count(distinct unique_sess_id) desc) AS row_num_1,
  ROW_NUMBER() OVER(PARTITION BY Category ORDER BY sum(durationHours) desc) AS row_num_2,
  ROW_NUMBER() OVER(PARTITION BY Category ORDER BY count(distinct user_id) desc) AS row_num_3
  FROM `macro-magpie-365619.Zattoo.raw_watch_sessions_clean`
  group by Category, channel
  order by Category;

create or replace table `macro-magpie-365619.Zattoo.popular_chan_based_on_sess` as
select Category, channel as famous_based_on_sess from `macro-magpie-365619.Zattoo.rank_table_2` where row_num_1 <= 3
order by Category asc;

create or replace table `macro-magpie-365619.Zattoo.popular_chan_based_on_dur` as
select Category, channel as famous_based_on_duration from `macro-magpie-365619.Zattoo.rank_table_2` where row_num_2 <= 3
order by Category asc;

create or replace table `macro-magpie-365619.Zattoo.popular_chan_based_on_user_count` as
select Category, channel as famous_based_on_user_count from `macro-magpie-365619.Zattoo.rank_table_2` where row_num_3 <= 3
order by Category asc;
```

Both temporary tables and Common Table Expressions (CTEs) are used in SQL to store intermediate results that can be used in subsequent queries.

Temporary tables can be used to store data that needs to persist across multiple sessions or transactions. They can be particularly useful when working with large amounts of data or when multiple queries need to be executed against the same set of data.

On the other hand, CTEs can be particularly useful when working with complex queries that require multiple levels of nesting or when a query needs to be executed repeatedly with different input parameters.

In general, temporary tables are more efficient when working with large amounts of data or when multiple queries need to be executed against the same set of data. CTEs, on the other hand, are more flexible and can be used to simplify complex queries and make them easier

to read and understand. The choice between temporary tables and CTEs will depend on the specific requirements of the query and the preferences of the developer or DBA.

Task 2: Dashboard EDA https://lookerstudio.google.com/reporting/0d10ad3c-5645-4bb3-b4ae-6dfe5ea3f32f