EGA Backend Developer Challenges Part 2

Exercise 2

Approach and solution SQL.

Directly below is the sql (and a link to that sql online) that I created to solve the sql query on the table file_processing_events. This solution below is a faster solution, which followed on from my first solution (the first solution is shown below in the section titled "Index").
the first solution timings were:
explain analyse
Planning Time: 0.590 ms
Execution Time: 170.452 ms

and this solution is
Planning Time: 0.743 ms
Execution Time: 76.880 ms

Solution SQL

This latest version of this sql is available online at:

https://github.com/KenBuckley/imdb/blob/master/part_2/part2.sql

```
WITH daily_events AS (
    SELECT
        event_date,
        num_entries_per_day,
        total_bytes_processed,
        average_processing_time,
        file_list as files,
        SUM(total_bytes_processed) OVER (ORDER BY event_date) AS
total_bytes_cumulative,
        all_tags,
        min_time_ms,
        min_event_id  -- Add this to uniquely identify the slowest entry
    FROM (
        SELECT
            date_trunc('day', event_time) AS event_date,
            COUNT(*) AS num_entries_per_day,
            SUM(bytes_processed) AS total_bytes_processed,
            AVG(processing_time_ms) as average_processing_time,
            array_agg(DISTINCT file_name) AS file_list,
            array_agg(DISTINCT tag) FILTER (WHERE tag IS NOT NULL) AS
```

```
all_tags,
            MIN(processing_time_ms) FILTER (WHERE metadata->>'source' =
'HTSGET') as min_time_ms,
            -- extra:get the event_id of the row with the
min(processing_time_ms).
            (array_agg(event_id ORDER BY processing_time_ms)
             FILTER (WHERE metadata->>'source' = 'HTSGET'))[1] as
min_event_id
        FROM file_processing_events
        LEFT JOIN LATERAL unnest(COALESCE(tags, ARRAY[]::text[])) AS tag ON
true
        WHERE metadata->>'source' = 'HTSGET'
        AND metadata ? 'weights'   --todo check if required
        AND jsonb_typeof(metadata->'weights') = 'array'  --required or will
get error
        GROUP BY event_date
    ) as day_stats
    ORDER BY event_date
),
-- Pre-calculate weights for the slowest entries
slowest_entries_weights AS (
    SELECT
        event_id,
        date_trunc('day', event_time) as event_date,
        (SELECT AVG(weight::numeric)
         FROM jsonb_array_elements_text(metadata->'weights') AS weight) AS
avg_weight,
        (SELECT SUM(weight::numeric)
         FROM jsonb_array_elements_text(metadata->'weights') AS weight) AS
sum_weight
    FROM file_processing_events
    WHERE metadata->>'source' = 'HTSGET'
    AND metadata ? 'weights'
    AND jsonb_typeof(metadata->'weights') = 'array' -- neccessary
)
SELECT
    de.event_date,
    de.num_entries_per_day,
    de.total_bytes_processed,
    de.average_processing_time,
    de.files,
    de.total_bytes_cumulative,
    de.all_tags,
```

```
    sew.avg_weight as average_weight_from_slowest_entry,
    sew.sum_weight as total_weight_from_slowest_entry
FROM daily_events de
JOIN slowest_entries_weights sew ON de.min_event_id = sew.event_id
ORDER BY de.event_date;
```

Questions

Q.Did you need to do any assumptions because of any ambiguity or any edge cases.

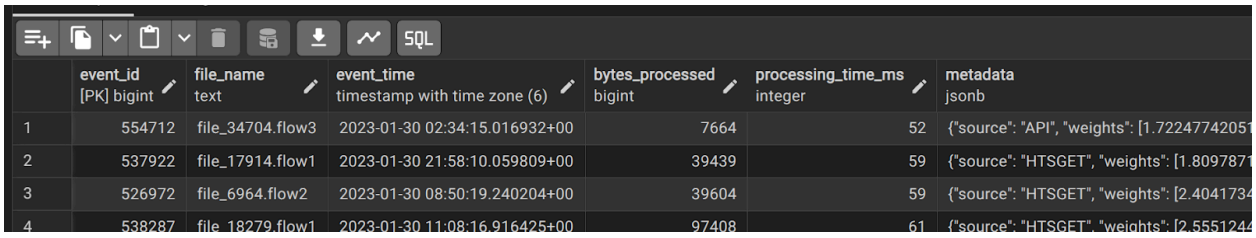
Yes,

1.There seems to be a problem with the aggregate counts -arising from where tag is null. Ideally we would replace all the nulls with empty [] .That would also change the results.
If we update tags to be [] instead of null using

```
update file_processing_events
set tags ='{}'
where tags is null;
```

then we can simplify the query.

2. there are a few records that have the same minimum processing_time_ms, for example in this day we have two rows with processing_time_ms = 59. So which one should we choose?



	event_id [PK] bigint	file_name text	event_time timestamp with time zone (6)	bytes_processed bigint	processing_time_ms integer	metadata jsonb
1	554712	file_34704.flow3	2023-01-30 02:34:15.016932+00	7664	52	{"source": "API", "weights": [1.7224774205
2	537922	file_17914.flow1	2023-01-30 21:58:10.059809+00	39439	59	{"source": "HTSGET", "weights": [1.809787
3	526972	file_6964.flow2	2023-01-30 08:50:19.240204+00	39604	59	{"source": "HTSGET", "weights": [2.404173
4	538287	file_18279.flow1	2023-01-30 11:08:16.916425+00	97408	61	{"source": "HTSGET", "weights": [2.555124

All the days with multiple processing times are:
"2023-01-30"
"2023-04-19"
"2023-07-23"
"2023-10-16"

As it is, the selection of the minimum record is not deterministic . We need more information to determine which row to select. Also the data will not be guaranteed deterministic [the results might change on the next database release] unless we modify the query to make it

deterministic: like [but this is still just a guess, really we need more information on which row to select].
(array_agg(event_id ORDER BY processing_time_ms, event_id)
 FILTER (WHERE metadata->>'source' = 'HTSGET'))[1] AS min_event_id

3.There are some records with missing metadata (the field weights is missing and the following code must be inserted: jsonb_typeof(metadata->'weights') = 'array

4 The date 2023-05-13 does not have an entry for HTSGET -all the sources are null for that day, so there was some error in the pipeline. This is the code you can use to detect it.
detect days with no source of HTSGET:

```
SELECT DISTINCT CAST(event_time AS DATE) AS calendar_day
FROM public.file_processing_events
WHERE CAST(event_time AS DATE) NOT IN (
    SELECT DISTINCT CAST(event_time AS DATE)
    FROM public.file_processing_events
    WHERE metadata->>'source' = 'HTSGET'
)
ORDER BY calendar_day;
```

There is no clear technical solution here, we must manually intervene and remove the day or manually insert data.

Q. Did I encounter any errors, and how did I solve them

The errors I encountered were the bad data like " the date 2023-05-13 does not have an entry for HTSGET", I cannot solve this without directions.
There are obviously some records with missing metadata, so we have to do jsonb_typeof(metadata->'weights') = 'array

Q.If we were going to create a graph on the website with the event_date, how would we modify something?

We get 200 days of results which is a quite a lot of data points to plot on the x-axis.
Also the data is spread over one calendar year 2023 to 2024.01.01 so we could aggregate the data up to months and show a point per month (12 data points obviously) -or we could go with weeks, it would be best to just export the data as a csv and plot the graph in excel and make a judgement from there.I would just make it a 2d graph in excel to keep it quick.

Q.would you create indexes on the table? Which ones and why.

Originally I built my query exactly as the question was asked in the document, but that led to a query that was nested with long

Yes indexes always help in case of scans. An explain analyse will show the best areas to improve. Looking at my query my immediate thought would be to index the event_time,
This will not help our case here immediately but will help in paging and seeking.

CREATE INDEX idx_event ON file_processing_events (event_time);

Perhaps use materialized view of the tablepublic.file_processing_events to add extra columns to the materialized view:
materialized view + ADD COLUMN event_day date (and then we can replace date_trunc('day', event_time) in the code);
materialized view + ADD COLUMN source_text text (and then we can replace (metadata->>'source') in the code.
)
now you can index on those two fields in the materialized view and change the query to work directly on event_day instead of date_trunc('day', event_time) and
directly on the new field source_text instead of (metadata->>'source')

Also due to the jsonb we should add:
 CREATE INDEX idx_metadata_gin ON file_processing_events USING gin (metadata);
(small decrease in runtime not noticeable with the data)

Index

Original approach:

This was the first query I built , when I created it by following the sequence of requirements as laid out in the document. Once I had that running I looked to use the field id: event_id (to more simply match the joining of tables daily_events and file_processing_events .
After that optimization I did an "explain analyse" to determine any expensive subqueries/loops. The query is relatively slow at 170ms , the final approach above runs at 70ms aprox.

First iteration without using :

```
with daily_events as (
SELECT
    event_date,
    num_entries_per_day,
    total_bytes_processed,
    average_processing_time,
    file_list as files,
    SUM(total_bytes_processed) OVER (ORDER BY event_date) AS
total_bytes_cumulative,
    all_tags,
    min_time_ms
FROM (
    SELECT
        date_trunc('day', event_time) AS event_date,
        COUNT(*) AS num_entries_per_day,
        SUM(bytes_processed) AS total_bytes_processed,
        avg(processing_time_ms) as average_processing_time,
        array_agg(DISTINCT file_name) AS file_list,
        array_agg(DISTINCT tag) FILTER (WHERE tag IS NOT NULL) AS all_tags,
        min(processing_time_ms) filter (where metadata->>'source' =
'HTSGET')  as min_time_ms
    FROM file_processing_events
    LEFT JOIN LATERAL unnest(COALESCE(tags, ARRAY[]::text[])) AS tag ON
true
    GROUP BY event_date
) as day_stats
ORDER BY event_date
)
select de.event_date, de.num_entries_per_day,
```

```
de.total_bytes_processed,de.average_processing_time,de.files,de.total_bytes
_cumulative,de.all_tags,
            (
        SELECT AVG(weight::numeric)
        FROM jsonb_array_elements_text(fpe.metadata->'weights') AS weight
    ) AS average_weight_from_slowest_entry,
    (
        SELECT SUM(wgt::numeric)
        FROM jsonb_array_elements_text(fpe.metadata->'weights') AS wgt
    ) AS total_weight_from_slowest_entry
from daily_events de,  file_processing_events fpe
where date_trunc('day', fpe.event_time) = de.event_date
and de.min_time_ms = fpe.processing_time_ms
and fpe.metadata->>'source' = 'HTSGET'  --ensure we do not pick up other
types
and fpe.metadata ? 'weights'  --maybe not necessary
and jsonb_typeof(fpe.metadata->'weights') = 'array'  --this is necessary!
order by event_date
```

explain analyse
Planning Time: 0.590 ms
Execution Time: 170.452 ms