Case Study 1: Job Data Analysis

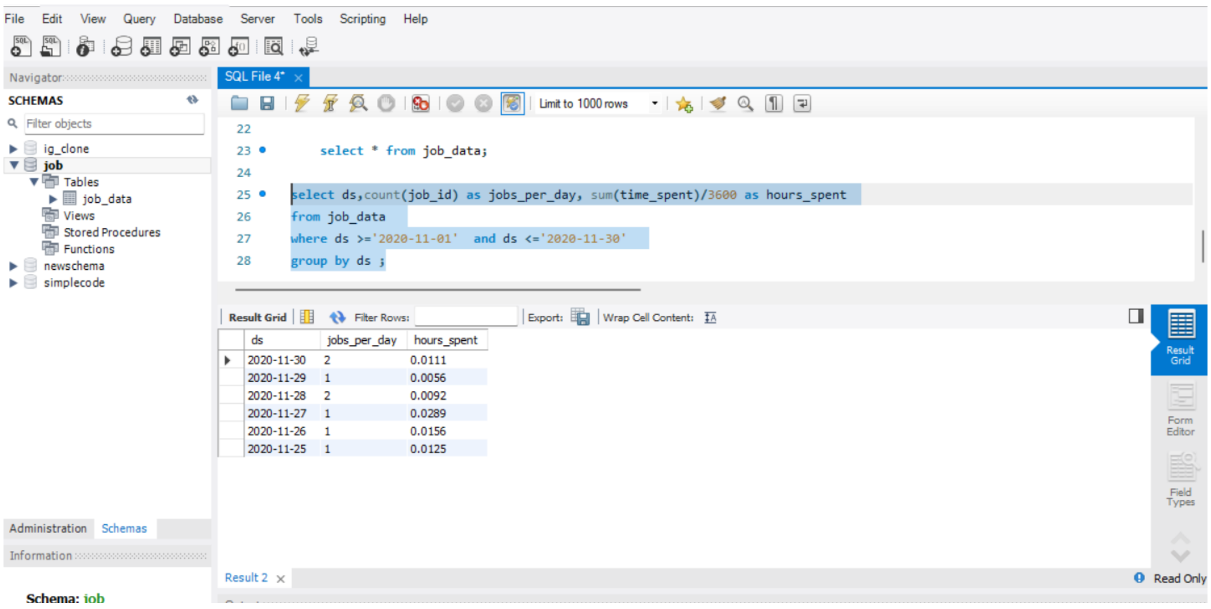
Tasks:

A. Jobs Reviewed Over Time:

- Objective: Calculate the number of jobs reviewed per hour for each day in November 2020.
- Your Task: Write an SQL query to calculate the number of jobs reviewed per hour for each day in November 2020.

Ans:

```
select ds,count(job_id) as jobs_per_day, sum(time_spent)/3600 as hours_spent
from job_data
where ds >='2020-11-01'  and ds <='2020-11-30'
group by ds ;
```



```
select * from job_data;

select ds,count(job_id) as jobs_per_day, sum(time_spent)/3600 as hours_spent
from job_data
where ds >='2020-11-01'  and ds <='2020-11-30'
group by ds ;
```

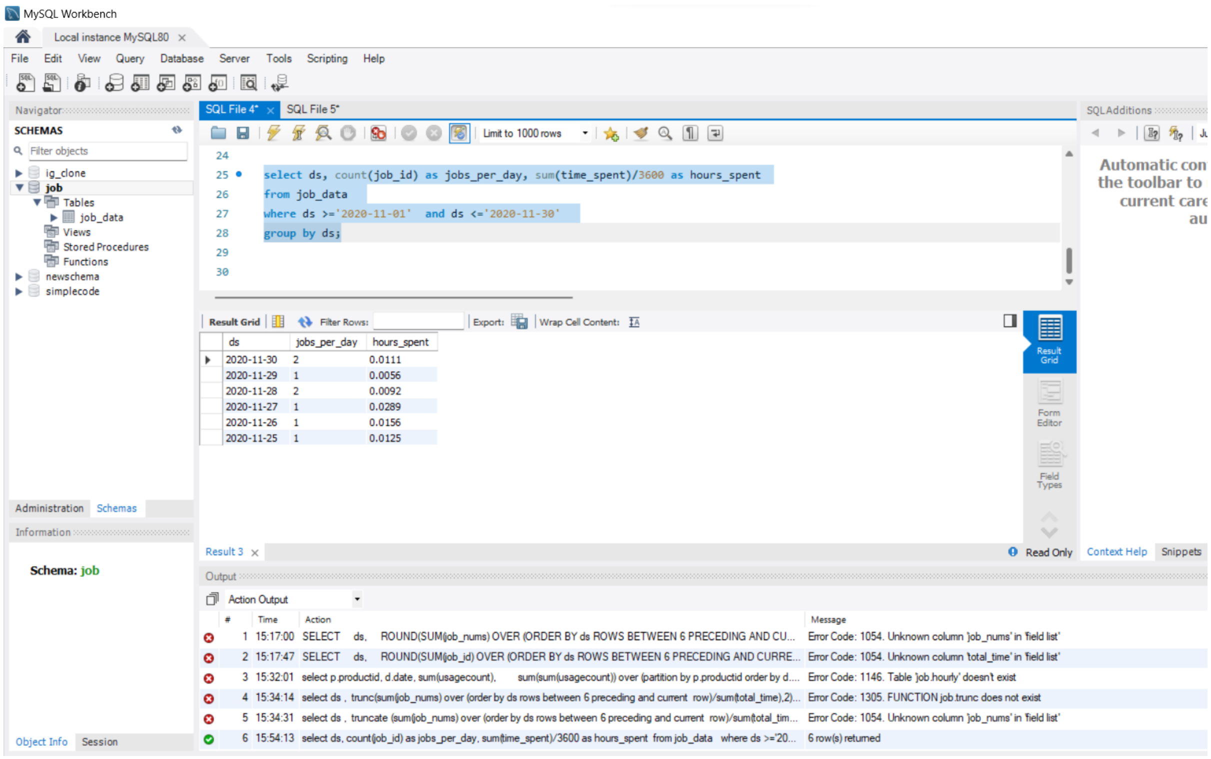
ds	jobs_per_day	hours_spent
2020-11-30	2	0.0111
2020-11-29	1	0.0056
2020-11-28	2	0.0092
2020-11-27	1	0.0289
2020-11-26	1	0.0156
2020-11-25	1	0.0125

B. **Throughput Analysis:**

- Objective: Calculate the 7-day rolling average of throughput (number of events per second).
- Your Task: Write an SQL query to calculate the 7-day rolling average of throughput. Additionally, explain whether you prefer using the daily metric or the 7-day rolling average for throughput, and why.

Ans:

```
select ds, count(job_id) as jobs_per_day, sum(time_spent)/3600 as hours_spent
from job_data
where ds >='2020-11-01'  and ds <='2020-11-30'
group by ds;
```



```
select ds, count(job_id) as jobs_per_day, sum(time_spent)/3600 as hours_spent
from job_data
where ds >='2020-11-01'  and ds <='2020-11-30'
group by ds;
```

ds	jobs_per_day	hours_spent
2020-11-30	2	0.0111
2020-11-29	1	0.0056
2020-11-28	2	0.0092
2020-11-27	1	0.0289
2020-11-26	1	0.0156
2020-11-25	1	0.0125

```
#  Time      Action                                                                                      Message
1  15:17:00  SELECT    ds,    ROUND(SUM(job_nums) OVER (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CU...     Error Code: 1054. Unknown column 'job_nums' in 'field list'
2  15:17:47  SELECT    ds,    ROUND(SUM(job_id) OVER (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRE...    Error Code: 1054. Unknown column 'total_time' in 'field list'
3  15:32:01  select p.productid, d.date, sum(usagecount),      sum(sum(usagecount)) over (partition by p.productid order by d....   Error Code: 1146. Table 'job.hourly' doesn't exist
4  15:34:14  select ds ,  trunc(sum(job_nums) over (order by ds rows between 6 preceding and current  row)/sum(total_time),2)...   Error Code: 1305. FUNCTION job.trunc does not exist
5  15:34:31  select ds ,  truncate (sum(job_nums) over (order by ds rows between 6 preceding and current  row)/sum(total_tim...   Error Code: 1054. Unknown column 'job_nums' in 'field list'
6  15:54:13  select ds, count(job_id) as jobs_per_day, sum(time_spent)/3600 as hours_spent  from job_data   where ds >='20...   6 row(s) returned
```

The daily metric provides accurate values for each day, reflecting the actual throughput for that specific day, but

The rolling average helps smooth out short-term fluctuations, making it easier to identify trends and patterns over a longer period.

If you need a more granular view of **daily fluctuations**, then daily metric might be more appropriate.

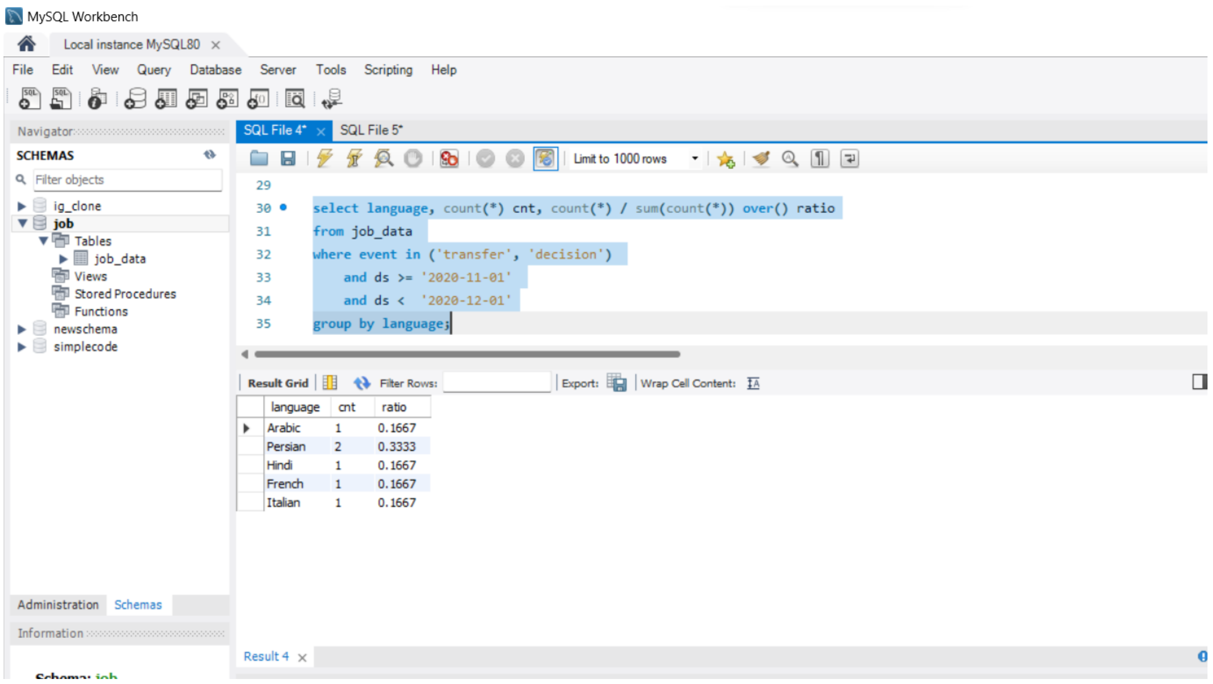
If you're interested in **identifying trends** over a more extended period while smoothing out short-term variations, the 7-day rolling average might be a better choice.

C. Language Share Analysis:

- Objective: Calculate the percentage share of each language in the last 30 days.
- Your Task: Write an SQL query to calculate the percentage share of each language over the last 30 days.

Ans:

```
select language, count(*) cnt, count(*) / sum(count(*)) over() ratio
from job_data
where event in ('transfer', 'decision')
   and ds >= '2020-11-01'
   and ds <  '2020-12-01'
group by language
```



```
select language, count(*) cnt, count(*) / sum(count(*)) over() ratio
from job_data
where event in ('transfer', 'decision')
    and ds >= '2020-11-01'
    and ds <  '2020-12-01'
group by language;
```

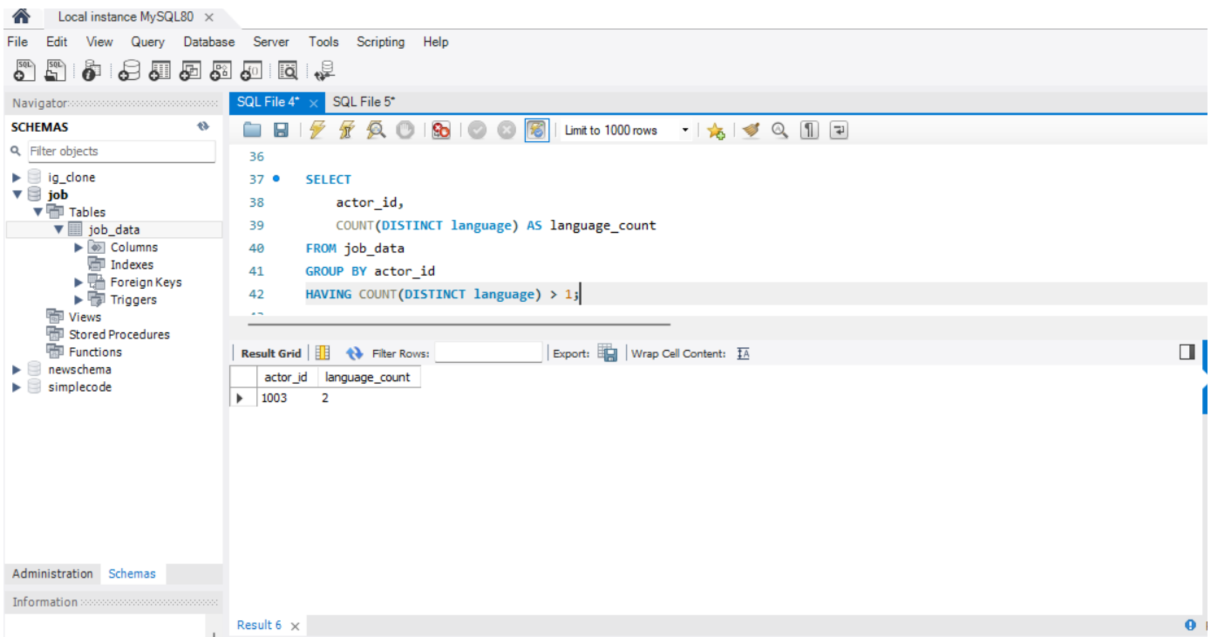
```
language  cnt  ratio
Arabic    1    0.1667
Persian   2    0.3333
Hindi     1    0.1667
French    1    0.1667
Italian   1    0.1667
```

D. **Duplicate Rows Detection:**

- Objective: Identify duplicate rows in the data.
- Your Task: Write an SQL query to display duplicate rows from the job_data table.

Ans:

```
SELECT
   actor_id,
   COUNT(DISTINCT language) AS language_count
FROM job_data
GROUP BY actor_id
HAVING COUNT(DISTINCT language) > 1;
```



```
SELECT
    actor_id,
    COUNT(DISTINCT language) AS language_count
FROM job_data
GROUP BY actor_id
HAVING COUNT(DISTINCT language) > 1;
```

actor_id	language_count
1003	2