Case Study 1: Job Data Analysis

- Creating database and table

```sql
create database casestudy_1;
CREATE TABLE job_data
(
    ds DATE,
    job_id INT NOT NULL,
    actor_id INT NOT NULL,
    event VARCHAR(15) NOT NULL,
    language VARCHAR(15) NOT NULL,
    time_spent INT NOT NULL,
    org CHAR(2)
);

INSERT INTO job_data (ds, job_id, actor_id, event, language, time_spent, org)
VALUES ('2020-11-30', 21, 1001, 'skip', 'English', 15, 'A'),
    ('2020-11-30', 22, 1006, 'transfer', 'Arabic', 25, 'B'),
    ('2020-11-29', 23, 1003, 'decision', 'Persian', 20, 'C'),
    ('2020-11-28', 23, 1005,'transfer', 'Persian', 22, 'D'),
    ('2020-11-28', 25, 1002, 'decision', 'Hindi', 11, 'B'),
    ('2020-11-27', 11, 1007, 'decision', 'French', 104, 'D'),
    ('2020-11-26', 23, 1004, 'skip', 'Persian', 56, 'A'),
    ('2020-11-25', 20, 1003, 'transfer', 'Italian', 45, 'C');
```

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.

```sql
SELECT DATE(ds) AS review_date,
       COUNT(*) AS jobs_reviewed_per_day,sum(time_spent)/3600 as jobs_reviewed_per_hr
FROM job_data
WHERE ds between '2020-11-01' and '20-11-30'
GROUP BY review_date
ORDER BY review_date;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

review_date	jobs_reviewed_per_day	jobs_reviewed_per_hr
2020-11-25	1	0.0125
2020-11-26	1	0.0156
2020-11-27	1	0.0289
2020-11-28	2	0.0092
2020-11-29	1	0.0056
2020-11-30	2	0.0111

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.

```sql
SELECT ds,
       time_spent,
       AVG(time_spent)
       OVER (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) AS rolling_7_day_average
FROM job_data;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ds	time_spent	rolling_7_day_average
2020-11-25	45	45.0000
2020-11-26	56	50.5000
2020-11-27	104	68.3333
2020-11-28	22	56.7500
2020-11-28	11	47.6000
2020-11-29	20	43.0000
2020-11-30	15	39.0000
2020-11-30	25	36.1429

Result Grid | Form Editor | Result 6 | Read Only

Reason:

Daily Metric: This provides the raw, day-to-day values of throughput without any smoothing or averaging. It might be useful when examining short-term trends or fluctuations. Daily metrics are sensitive to sudden changes and can provide insights into specific days' performances.

7-Day Rolling Average: This smoothed metric helps in identifying overall trends and patterns by averaging the data over a 7-day period, reducing the impact of daily fluctuations or irregularities. It's particularly useful for understanding long-term trends, identifying broader patterns, and eliminating noise in the data.

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.

```sql
select language,
    round(100*count(*)/(select count(*) from job_data),2) as percentage_share
from job_data
group by language
order by language desc;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

language	percentage_share
Persian	37.50
Italian	12.50
Hindi	12.50
French	12.50
English	12.50
Arabic	12.50

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.

```sql
select *
from
(select *,
row_number() over(partition by ds,actor_id,job_id) as row_num
from job_data) a
where row_num>1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ds	job_id	actor_id	event	language	time_spent	org	row_num