# Operation Analytics and Investigating Metric Spike

```
-- Case study 1 (Job Data)
-- Create Job table
CREATE TABLE job_table (
ds timestamp without time zone NOT NULL DEFAULT now(),
job_id INTEGER,
actor_id INTEGER,
event VARCHAR(255),
language VARCHAR(255),
time_spent INTEGER,
org VARCHAR(5)
)
-- Insert data into job table by right clicking on table in navigator
section and clicking on import
-- View data

SELECT
*
FROM
job_table

-- Calculate the number of jobs reviewed per hour per day for November
2020?

SELECT
        ds,
        ROUND(1.0 * COUNT(job_id) * 3600 / SUM(time_spent),2) AS jobs_reviewed_per_hour
FROM
        job_table
WHERE
        ds BETWEEN '2020-11-01' AND '2020-11-30'
AND event IN ('transfer' , 'decision')
GROUP BY ds

-- Let's say the above metric is called throughput. Calculate 7 day rolling average of throughput?
-- Calculating jobs_reviewed_per_sec (throughput)
SELECT
ds,
ROUND(COUNT(job_id)/ SUM(time_spent), 2) AS jobs_reviewed_per_sec_throughput
FROM
job_table
```

```
WHERE
ds BETWEEN '2020-11-01' AND '2020-11-30'
AND event IN ('transfer' , 'decision')
GROUP BY ds

-- Calculate 7 day rolling average of throughput?

SELECT
ds,
ROUND(AVG(jobs_reviewed_per_sec_throughput) OVER (ORDER BY ds ROWS 6
PRECEDING), 4)
FROM (SELECT
ds,
ROUND(COUNT(job_id) / SUM(time_spent),4) AS jobs_reviewed_per_sec_throughput
FROM job_table
WHERE ds BETWEEN '2020-11-01' AND '2020-11-30'
AND event IN ('transfer', 'decision')
GROUP BY ds) a
ORDER BY ds

-- Calculate the percentage share of each language in the last 30 days?

SELECT
language,
(100 * a.language_count / b.total_count) AS percentage_share
FROM (SELECT
language,
COUNT(*) AS language_count
FROM job_table
WHERE ds BETWEEN '2020-11-01' AND '2020-11-30'
AND event IN ('transfer', 'decision')
GROUP BY language) a
CROSS JOIN (SELECT
COUNT(*) AS total_count
FROM job_table
WHERE ds BETWEEN '2020-11-01' AND '2020-11-30'
AND event IN ('transfer', 'decision')) b

-- Let’s say you see some duplicate rows in the data. How will you display duplicates from the
table?
-- we will create another table and add a duplicate row to understand the process for displaying
duplicate rows

CREATE TABLE job_table_2 (
```

```
ds  timestamp without time zone NOT NULL DEFAULT now(),
job_id INTEGER,
actor_id INTEGER,
event VARCHAR(255),
language VARCHAR(255),
time_spent INTEGER,
org VARCHAR(5)
)

-- Inserting values into table from base table

INSERT INTO job_table_2(ds,job_id,actor_id,event,language,time_spent,org) VALUES
('2020-11-30',21,1001,'skip','English',15,'A');
INSERT INTO job_table_2(ds,job_id,actor_id,event,language,time_spent,org) VALUES
('2020-11-30',22,1006,'transfer','Arabic',25,'B');
INSERT INTO job_table_2(ds,job_id,actor_id,event,language,time_spent,org) VALUES
('2020-11-29',23,1003,'decision','Persian',20,'C');
INSERT INTO job_table_2(ds,job_id,actor_id,event,language,time_spent,org) VALUES
('2020-11-28',23,1005,'transfer','Persian',22,'D');
INSERT INTO job_table_2(ds,job_id,actor_id,event,language,time_spent,org) VALUES
('2020-11-28',25,1002,'decision','Hindi',11,'B');
INSERT INTO job_table_2(ds,job_id,actor_id,event,language,time_spent,org) VALUES
('2020-11-27',11,1007,'decision','French',104,'D');
INSERT INTO job_table_2(ds,job_id,actor_id,event,language,time_spent,org) VALUES
('2020-11-26',23,1004,'skip','Persian',56,'A');
INSERT INTO job_table_2(ds,job_id,actor_id,event,language,time_spent,org) VALUES
('2020-11-25',20,1003,'transfer','Italian',45,'C');
SELECT
*
FROM job_table_2

-- Adding duplicate row to the duplicate table

INSERT INTO job_table_2
SELECT
*
FROM job_table
WHERE actor_id = 1006

-- View duplicate table

SELECT
*
FROM job_table_2
```

```
-- Displaying duplicate rows
SELECT
*
FROM (SELECT
*,
ROW_NUMBER() OVER (PARTITION BY ds, job_id, actor_id, event, language,time_spent, org
ORDER BY job_id) AS row_num
FROM job_table_2) a
WHERE row_num >1
```

**-- Case study 2 (Investigating metric spike)**
**-- Create Tables**

```
CREATE TABLE users (
user_id integer NOT NULL,
created_at datetime,
company_id integer,
language varchar(50),
activated_at datetime,
state varchar(50)
)
CREATE TABLE events (
user_id integer NOT NULL,
occured_at datetime,
event_type varchar(50),
event_name varchar(50),
location varchar(50),
device varchar(255),
user_type integer
)
CREATE TABLE email_events (
user_id integer NOT NULL,
occured_at datetime,
action varchar(255),
user_type integer
)
-- Calculate the weekly user engagement?
SELECT
DATE_TRUNC('week', e.occurred_at) AS week_interval,
COUNT(DISTINCT e.user_id) AS weekly_active_users
FROM
events e
WHERE
```

```
e.event_type = 'engagement'
AND e.event_name = 'login'
GROUP BY
week_interval
ORDER BY
week_interval
-- Calculate the user growth for product?
SELECT
DATE_TRUNC('day', created_at) AS DAY,
COUNT(*) AS all_users,
COUNT(
CASE
WHEN activated_at IS NOT NULL THEN u.user_id
ELSE NULL
END
) AS activated_users
FROM
users u
WHERE
created_at >= '2014-06-01'
AND created_at < '2014-09-01'
GROUP BY
DAY
ORDER BY
DAY
-- Calculate the weekly retention of users-sign up cohort?
SELECT DATE_TRUNC('week',a.occurred_at) AS "week",
COUNT(DISTINCT CASE WHEN a.user_age > 70 THEN a.user_id ELSE NULL
END) AS "10+ weeks",
COUNT(DISTINCT CASE WHEN a.user_age < 70 AND a.user_age >= 63 THEN
a.user_id ELSE NULL END) AS "9 weeks",
COUNT(DISTINCT CASE WHEN a.user_age < 63 AND a.user_age >= 56 THEN
a.user_id ELSE NULL END) AS "8 weeks",
COUNT(DISTINCT CASE WHEN a.user_age < 56 AND a.user_age >= 49 THEN
a.user_id ELSE NULL END) AS "7 weeks",
COUNT(DISTINCT CASE WHEN a.user_age < 49 AND a.user_age >= 42 THEN
a.user_id ELSE NULL END) AS "6 weeks",
COUNT(DISTINCT CASE WHEN a.user_age < 42 AND a.user_age >= 35 THEN
a.user_id ELSE NULL END) AS "5 weeks",
COUNT(DISTINCT CASE WHEN a.user_age < 35 AND a.user_age >= 28 THEN
a.user_id ELSE NULL END) AS "4 weeks",
COUNT(DISTINCT CASE WHEN a.user_age < 28 AND a.user_age >= 21 THEN
a.user_id ELSE NULL END) AS "3 weeks",
COUNT(DISTINCT CASE WHEN a.user_age < 21 AND a.user_age >= 14 THEN
```

```
a.user_id ELSE NULL END) AS "2 weeks",
COUNT(DISTINCT CASE WHEN a.user_age < 14 AND a.user_age >= 7 THEN
a.user_id ELSE NULL END) AS "1 week",
COUNT(DISTINCT CASE WHEN a.user_age < 7 THEN a.user_id ELSE NULL
END) AS "Less than a week"
FROM (
SELECT e.occurred_at,
u.user_id,
DATE_TRUNC('week',u.activated_at) AS activation_week,
EXTRACT('day' FROM '2014-09-01'::TIMESTAMP -
u.activated_at) AS user_age
FROM users u
JOIN events e
ON e.user_id = u.user_id
AND e.event_type = 'engagement'
AND e.event_name = 'login'
AND e.occurred_at >= '2014-05-01'
AND e.occurred_at < '2014-09-01'
WHERE u.activated_at IS NOT NULL
) a
GROUP BY DATE_TRUNC('week',a.occurred_at)
ORDER BY DATE_TRUNC('week',a.occurred_at)
LIMIT 100
-- Calculate the weekly engagement per device?
SELECT DATE_TRUNC('week', occurred_at) AS week,
COUNT(DISTINCT e.user_id) AS weekly_active_users,
COUNT(DISTINCT CASE WHEN e.device IN ('macbook pro','lenovo
thinkpad','macbook air','dell inspiron notebook',
'asus chromebook','dell inspiron desktop','acer aspire
notebook','hp pavilion desktop','acer aspire desktop','mac mini')
THEN e.user_id ELSE NULL END) AS computer,
COUNT(DISTINCT CASE WHEN e.device IN ('iphone 5','samsung galaxy
s4','nexus 5','iphone 5s','iphone 4s','nokia lumia 635',
'htc one','samsung galaxy note','amazon fire phone') THEN e.user_id
ELSE NULL END) AS phone,
COUNT(DISTINCT CASE WHEN e.device IN ('ipad air','nexus 7','ipad
mini','nexus 10','kindle fire','windows surface',
'samsumg galaxy tablet') THEN e.user_id ELSE NULL END) AS tablet
FROM events e
WHERE e.event_type = 'engagement'
AND e.event_name = 'login'
GROUP BY week
ORDER BY week
LIMIT 100
```