Target Statistics to be computed.

Across all users (per day and across all days)

* Average lead time
* Average FDR of 70
* Average FDR of 80
* Average number of hypo alerts
* Average number of hypo feedback
* Average MyData pct below range
* Average number of MyData hypo excursions
* Weekly Average MyData pct below range
* Weekly count of MyData hypo excursions
* Weekly count of HYPO glycemic Insights
* Weekly count of In Range glycemic insights
* Average gap time (time without SG)
* Average catchup time (length of batch catchup packets)

Per user (per day and across all days)

* Average lead time
* Average FDR of 70
* Average FDR of 80
* Average MyData pct below range
* Average number of hypo excursions
* Weekly Average MyData pct below range
* Weekly count of MyData hypo excursions
* Weekly count of HYPO glycemic Insights
* Weekly count of In Range glycemic insights
* Average gap time (time without SG)
* Average catchup time (length of batch catchup packets)

1. **Average lead time**

Data specs:

Use the mdt.sensor\_glucose\_data to catch up the latet 24-hour cache, otherwise, use the cleansg.usersg table or parsing sglatest out from hypo\_features table

To calculate lead time in precise, suggest to extract the Kafka hyposcore topic which get score blocking cases excluded. DB2 data won’t be able to provide this info.

Query outline:

Find the true positive alerts by excluding those ones after pairing the alerts and feedback messages.

Exclude those ones created within the latest 4 hours.

Find the start of hypo episode

Compute the leadtime

"with ab as (select a.id from analytics.streams as a Inner join (select person\_id, id, created\_at from analytics.streams where json\_data like '%feedback\_notice.1%' and created\_at >= (CURRENT\_TIMESTAMP - 20 HOURS)) as b on a.person\_id=b.person\_id and abs(TIMESTAMPDIFF(4, char(b.created\_at- 4 hours - a.created\_at)))<=7 where a.json\_data like '%hypo.1%' and a.created\_at >= (CURRENT\_TIMESTAMP - 24 HOURS)),

f as (select c.person\_id, c.id, c.created\_at as alert\_created from analytics.streams as c inner join (select person\_id, updated\_at from mdt.hypo\_score\_data) as d  
on c.person\_id=d.person\_id and c.created\_at < (d.updated\_at - 4 hours) where c.json\_data like '%hypo.1%' and c.created\_at >= (CURRENT\_TIMESTAMP - 24 HOURS) and  
c.id not in (select id from ab)),

summary as (select e.person\_id, e.sg, e.sg\_timestamp, e.sg\_timestamp\_tz, f.alert\_created, ROW\_NUMBER() OVER(PARTITION BY e.person\_id,f.alert\_created ORDER BY e.sg\_timestamp) AS rk from mdt.sensor\_glucose\_data as e inner join f  
on e.person\_id = f.person\_id and e.sg\_timestamp < (f.alert\_created + 4 hours) and e.sg\_timestamp > (f.alert\_created -5 minutes) where e.sg<70 )

select count(\*) as \"number true alerts in last 24 hours\", AVG(TIMESTAMPDIFF(4, char(s.sg\_timestamp-s.alert\_created))) as \"average lead time\" from summary s where s.rk=1"

Done in script: TBD

One of the advantages of using scrits for reporting is it would be more convenient for plotting.

1. **Average number of hypo alerts | Average number of hypo feedback**

Data specs: analytics.streams table

Query outline:

Directly count the alert and feedback based on Json format.

db2 "select AVG(alert\_counts) as \"average hypo alert\" from (select person\_id, count(id) as alert\_counts from analytics.streams where json\_data like '%hypo.1%' and created\_at >= (CURRENT\_TIMESTAMP - 24 HOURS) group by person\_id)"

db2 "select AVG(feedback\_counts) as \"average hypo feedback\" from (select person\_id, count(id) as feedback\_counts from analytics.streams where json\_data like '%feedback\_notice.1%' and created\_at >= (CURRENT\_TIMESTAMP - 24 HOURS) group by person\_id)"

1. **Average FDR of 70 | Average FDR of 80**

Data specs: analytics.streams table (FDR of 70)

analytics.streams table and mdt.sensor\_glucose\_data (FDR of 80)

Pull out Kafka hyposcore message for precise analysis in scripts

Query outline:

Count the false positive alerts after pairing the alerts and feedback messages.

Count the total alerts

Compute the FDR

FDR of 70:

db2 "select avg(ROUND((numb\_false\_positive\_alert \* 100.0 / total\_numb\_alert),2)) as \"average FDR 70mgdL pct per users with a false alert in the last 24 hours\" from (select person\_id, count(id) as total\_numb\_alert from analytics.streams where json\_data like '%hypo.1%' and created\_at >= (CURRENT\_TIMESTAMP - 24 HOURS) group by person\_id) as m inner join (select a.person\_id, count(id) as numb\_false\_positive\_alert from analytics.streams as a inner join (select person\_id, created\_at from analytics.streams where json\_data like '%feedback\_notice.1%' ) as b on a.person\_id=b.person\_id and abs(TIMESTAMPDIFF(4,char(b.created\_at - 4 hours - a.created\_at))) <=7 where a.json\_data like '%hypo.1%' and a.created\_at >= (CURRENT\_TIMESTAMP - 24 HOURS) group by a.person\_id) as n on m.person\_id = n.person\_id"

Lead time in individual level:

Suggest to extend the 20-hour and 24-hour duration to a longer period if necessary:

db2 "with n as (select a.person\_id, count(id) as numb\_false\_positive\_alert from analytics.streams as a inner join (select person\_id, created\_at from analytics.streams where json\_data like '%feedback\_notice.1%' and created\_at >= (CURRENT\_TIMESTAMP - 20 HOURS)) as b on a.person\_id=b.person\_id and abs(TIMESTAMPDIFF(4,char(b.created\_at - 4 hours - a.created\_at))) <=7 where a.json\_data like '%hypo.1%' and a.created\_at >= (CURRENT\_TIMESTAMP - 24 HOURS) group by a.person\_id),

m as (select person\_id, count(id) as total\_numb\_alert from analytics.streams where json\_data like '%hypo.1%' and created\_at >= (CURRENT\_TIMESTAMP - 24 HOURS) group by person\_id)

select m.person\_id, m.total\_numb\_alert, n.numb\_false\_positive\_alert, ROUND((n.numb\_false\_positive\_alert \* 100.0 / m.total\_numb\_alert),2) as \"FDR 70mgdL pct with a false alert in the last 24 hours\"

from m inner join n on m.person\_id = n.person\_id "

db2 "with n as (select a.person\_id, count(id) as numb\_false\_positive\_alert from analytics.streams as a inner join (select person\_id, created\_at from analytics.streams where json\_data like '%feedback\_notice.1%' and created\_at >= (CURRENT\_TIMESTAMP - 20 HOURS)) as b on a.person\_id=b.person\_id and abs(TIMESTAMPDIFF(4,char(b.created\_at - 4 hours - a.created\_at))) <=7 where a.json\_data like '%hypo.1%' and a.created\_at >= (CURRENT\_TIMESTAMP - 24 HOURS) group by a.person\_id),

m as (select person\_id, count(id) as total\_numb\_alert, date(created\_at) as date from analytics.streams where json\_data like '%hypo.1%' and created\_at >= (CURRENT\_TIMESTAMP - 24 HOURS) group by person\_id, date(created\_at))

select m.person\_id, m.total\_numb\_alert, n.numb\_false\_positive\_alert, ROUND((n.numb\_false\_positive\_alert \* 100.0 / m.total\_numb\_alert),2) as \"FDR 70mgdL pct with a false alert in the last 24 hours\"

from m inner join n on m.person\_id = n.person\_id "

Query outline:

Find false positive alerts after pairing the alerts and feedback messages.

Exclude those ones having SG<80mgdL in the 4-hr window after creation

Count the total alerts

Compute the FDR

FDR of 80 (**to-do: change the query to compute average in the latest 24-hr**)

db2 "with summary as (select a.person\_id, a.id, a.created\_at as alert\_created from analytics.streams as a inner join (select person\_id, created\_at from analytics.streams where json\_data like '%feedback\_notice.1%' ) as b on a.person\_id=b.person\_id and abs(TIMESTAMPDIFF(4,char(b.created\_at - 4 hours - a.created\_at))) <=7 where a.json\_data like '%hypo.1%')

select m.person\_id, total\_numb\_alert, numb\_false\_positive\_alert, ROUND((numb\_false\_positive\_alert \* 100.0 / total\_numb\_alert),2) as FDR\_80mgdL\_pct from (select person\_id, count(id) as total\_numb\_alert from analytics.streams where json\_data like '%hypo.1%' group by person\_id) as m inner join (select e.person\_id, count(e.id) as numb\_false\_positive\_alert from summary as e where e.id not in (select distinct c.id from summary as c Inner join mdt.sensor\_glucose\_data as d on d.person\_id=c.person\_id and d.sg\_timestamp < (c.alert\_created + 4 hours) and d.sg\_timestamp > (c.alert\_created -5 minutes) where d.sg<80) group by e.person\_id) as n on m.person\_id=n.person\_id"

Put 24-hour contengency:

db2 "with summary as (select a.person\_id, a.id, a.created\_at as alert\_created from analytics.streams as a inner join (select person\_id, created\_at from analytics.streams where json\_data like '%feedback\_notice.1%' and created\_at >= (CURRENT\_TIMESTAMP - 20 HOURS)) as b on a.person\_id=b.person\_id and abs(TIMESTAMPDIFF(4,char(b.created\_at - 4 hours - a.created\_at))) <=7 where a.json\_data like '%hypo.1%' and a.created\_at >= (CURRENT\_TIMESTAMP - 24 HOURS))

select m.person\_id, total\_numb\_alert, numb\_false\_positive\_alert, ROUND((numb\_false\_positive\_alert \* 100.0 / total\_numb\_alert),2) as FDR\_80mgdL\_pct from (select person\_id, count(id) as total\_numb\_alert from analytics.streams where json\_data like '%hypo.1%' and created\_at >= (CURRENT\_TIMESTAMP - 24 HOURS) group by person\_id) as m inner join (select e.person\_id, count(e.id) as numb\_false\_positive\_alert from summary as e where e.id not in (select distinct c.id from summary as c Inner join mdt.sensor\_glucose\_data as d on d.person\_id=c.person\_id and d.sg\_timestamp < (c.alert\_created + 4 hours) and d.sg\_timestamp > (c.alert\_created -5 minutes) where d.sg<80 and d.sg\_timestamp >= (CURRENT\_TIMESTAMP - 24 HOURS)) group by e.person\_id) as n on m.person\_id=n.person\_id"

Done in script: TBD

1. **Average MyData pct below range | Average number of MyData hypo excursions| Weekly Average MyData pct below range | Weekly count of MyData hypo excursions**

Data specs: analytics.mydata\_entries table

*Note: Average number of MyData hypo may not be feasible because the summaries are based on percentage only and the counts would not be precise if there are missing SGs.*

Count the average days having hypo (per week, per month, per 3-month or out of all) is available.

Query outline:

Extract the below range percentage value from the Json message based on the latest one-day MyData record

Compute the average across users

db2 "select avg(pct\*1.0) as daily\_hypo\_pct from (select person\_id, end\_at, integer(replace(substr(json,locate('belowRange',json)+12,2),',','')) as pct from (select person\_id, end\_at, substr(mydata\_json\_data,90,100) as json, ROW\_NUMBER() OVER(PARTITION BY person\_id ORDER BY end\_at DESC) as rk from analytics.mydata\_entries where period\_type='ONE\_DAY') where rk=1)"

Query outline:

Extract the below range percentage value from the Json message based on the latest one-week MyData record

Compute the average across users

db2 "select avg(pct\*1.0) as weekly\_hypo\_pct from (select person\_id, end\_at, integer(replace(substr(json,locate('belowRange',json)+12,2),',','')) as pct from (select person\_id, end\_at, substr(mydata\_json\_data,90,100) as json, ROW\_NUMBER() OVER(PARTITION BY person\_id ORDER BY end\_at DESC) as rk from analytics.mydata\_entries where period\_type='ONE\_WEEK') where rk=1)"

Query outline:

Extract the below range percentage value from the Json message based on the one-day records within the past 7 days

Count the total number of users and the average number of days having hypo events

db2 "select count(\*) as total\_users, avg(weekly\_hypo\_days) as avg\_weekly\_hypo\_days from (select person\_id, count(end\_at) as weekly\_hypo\_days from (select person\_id, end\_at, integer(replace(substr(json,locate('belowRange',json)+12,2),',','')) as pct from (select person\_id, end\_at, substr(mydata\_json\_data,90,100) as json from analytics.mydata\_entries where period\_type='ONE\_DAY'and (DAYS(CURRENT\_DATE)-DAYS(end\_at))<=7 ) ) where pct>0 group by person\_id)" | more

1. **Weekly count of HYPO glycemic Insights | Weekly count of In Range glycemic insights**

Data specs:

List of glycemic insights assocated with hypo SG stats, including hypo episode duration or hypo trend:

R3:

See the output of the query below

R4:

G31, 33-34, 37-38, 41-42, 53-54,57-58, 63-64, 69-70,75-76, 81-82, 85-86,89,95,107-108,185-186,203,209,219,223,229,255,263,379-380,385-386,391-392,445-446,451,455-456,475-476,480-481,485-486,489-491,495-496,500-501,505-506,509-511,515-516,520-521,525-526,530-531

Query outline:

List hypo events associated glycemic insights

[lcao-us@mdt-pil-qms-db21 ~]$ db2 "select distinct substr(insight\_id,1,15), substr(outcome,1,50), substr(event\_name,1,20) from insights.glycemic\_insight where outcome like '%ypo%'" | more

1 2 3

--------------- -------------------------------------------------- --------------------

glycemic.107 Pct time in hypo within this day CALENDARDAY

glycemic.143 Pct time in hypo within this day CALENDARDAY

glycemic.155 Pct time in hypo within this day CALENDARDAY

glycemic.185 Pct time in hypo within this day CALENDARDAY

glycemic.186 Pct time in severe hypo within this day CALENDARDAY

glycemic.209 Has Hypo event allBolusEvents

glycemic.229 Has Hypo event allBolusEvents

glycemic.255 Has Hypo event allBolusEvents

glycemic.263 Has Hypo event allBolusEvents

glycemic.33 Has hypo events within the next 4 hours HIGHROC

glycemic.37 Has hypo events within the next 4 hours HIGHROC

glycemic.385 Pct time in hypo within the time block TIMEBLOCK

glycemic.391 Pct time in hypo within the time block TIMEBLOCK

glycemic.41 Has hypo events within the next 4 hours HIGHROC

glycemic.480 Has hypo events within the next 4 hours MEAL

glycemic.53 Has hypo events within the next 4 hours HIGHROC

glycemic.54 Has severe hypo events within the next 4 hours HIGHROC

glycemic.57 Has hypo events within the next 4 hours HIGHROC

glycemic.69 Pct time in hypo within this day CALENDARDAY

glycemic.81 Pct time in hypo within this day CALENDARDAY

glycemic.82 Pct time in severe hypo within this day CALENDARDAY

glycemic.89 Pct time in hypo within this day CALENDARDAY

22 record(s) selected.

Query outline:

List non-hypo & non-hyper events (SG in control) associated glycemic insights

[lcao-us@mdt-pil-qms-db21 ~]$ db2 "select distinct substr(insight\_id,1,15), substr(outcome,1,50), substr(event\_name,1,20) from insights.glycemic\_insight where outcome like '%control%'" | more

1 2 3

--------------- -------------------------------------------------- --------------------

glycemic.147 Pct time in control within this day CALENDARDAY

glycemic.159 Pct time in control within this day CALENDARDAY

glycemic.160 Has 80% time in control within this day CALENDARDAY

glycemic.189 Pct time in control within this day CALENDARDAY

glycemic.383 Pct time in control within the time block TIMEBLOCK

glycemic.389 Pct time in control within the time block TIMEBLOCK

glycemic.390 Has 80% time in control within the time block TIMEBLOCK

glycemic.395 Pct time in control within the time block TIMEBLOCK

glycemic.396 Has 80% time in control within the time block TIMEBLOCK

glycemic.479 Pct time in control within the next 4 hours MEAL

glycemic.540 Pct time in control within the next 4 hours MEAL

glycemic.541 Pct time in control within the next 4 hours MEAL

glycemic.542 Pct time in control within the next 4 hours MEAL

glycemic.543 Pct time in control within the next 4 hours MEAL

glycemic.544 Pct time in control within the next 4 hours MEAL

glycemic.545 Pct time in control within the next 4 hours MEAL

glycemic.546 Pct time in control within the next 4 hours MEAL

glycemic.547 Pct time in control within the next 4 hours MEAL

glycemic.548 Pct time in control within the next 4 hours MEAL

glycemic.67 Pct time in control within this day CALENDARDAY

glycemic.68 Has 80% time in control within this day CALENDARDAY

glycemic.73 Pct time in control within this day CALENDARDAY

glycemic.74 Has 80% time in control within this day CALENDARDAY

glycemic.93 Pct time in control within this day CALENDARDAY

glycemic.99 Pct time in control within this day CALENDARDAY

25 record(s) selected.

Query outline:

Count the number of glycemic insights created within the above IDs in the past week.

[lcao-us@mdt-pil-qms-db21 ~]$ db2 "select person\_id, count(\*) as weekly\_hypo\_insights from insights.glycemic\_insight where outcome like '%ypo%' and insight\_ts >= (CURRENT\_TIMESTAMP - 7 DAYS) group by person\_id" | more

db2 "select AVG(weekly\_hypo\_insights) as avg\_weekly\_hypo\_insights from (select person\_id, count(\*) as weekly\_hypo\_insights from insights.glycemic\_insight where outcome like '%ypo%' and insight\_ts >= (CURRENT\_TIMESTAMP - 7 DAYS) group by person\_id)" | more

AVG\_WEEKLY\_HYPO\_INSIGHTS

------------------------

3

1 record(s) selected.

db2 "select person\_id, count(\*) as weekly\_inRange\_insights from insights.glycemic\_insight where outcome like '%control%' and insight\_ts >= (CURRENT\_TIMESTAMP - 7 DAYS) group by person\_id" | more

db2 "select AVG(weekly\_inRange\_insights) as avg\_weekly\_inRange\_insights from (select person\_id, count(\*) as weekly\_inRange\_insights from insights.glycemic\_insight where outcome like '%control%' and insight\_ts >= (CURRENT\_TIMESTAMP - 7 DAYS) group by person\_id)" | more

AVG\_WEEKLY\_INRANGE\_INSIGHTS

---------------------------

13

1. **Average gap time (time without SG)**

Data specs: mdt.hypo\_features table

Note:

* 1. jitter issue
  2. less than 5 min interval issue

Query outline:

Adjust the SG pump time to correct the jitter issue

Divid the SG data flow into continous sessions and gaps (with missing SGs)

Adjust the gap due to the less than 5-min interval

Count the average gap duration per user

[lcao-us@mdt-pil-qms-db21 ~]$ db2 "with sgtime\_adjust as (select person\_id, round\_timestamp(sg\_timestamp,'MI') as pump\_time from mdt.hypo\_features order by person\_id, sg\_timestamp),

> div\_session as (select person\_id, pump\_time, TRUNC\_TIMESTAMP(pump\_time - (CAST(row\_number() over (partition by person\_id order by pump\_time) as INT) \* 5) MINUTES, 'MI') as cont\_session from sgtime\_adjust),

> group\_session as (select distinct person\_id, cont\_session, min(pump\_time) as start\_sgtime, max(pump\_time) as end\_sgtime from div\_session group by person\_id, cont\_session),

> session\_adjust as (select person\_id, cont\_session, start\_sgtime, end\_sgtime, abs(TIMESTAMPDIFF(4, char(cont\_session - lag(cont\_session) over (partition by person\_id order by start\_sgtime) ))) as diff, lag(end\_sgtime) over (partition by person\_id order by start\_sgtime) as prev\_end\_sgtime from group\_session),

> session\_remove as (select person\_id, start\_sgtime, end\_sgtime, prev\_end\_sgtime from session\_adjust where diff > 1 or (diff=1 and start\_sgtime > prev\_end\_sgtime)),

> gap\_session as (select person\_id, start\_sgtime, end\_sgtime, TIMESTAMPDIFF(4, char(start\_sgtime - prev\_end\_sgtime)) as gap\_length\_in\_min from session\_remove)

> select person\_id, AVG(gap\_length\_in\_min) as avg\_gap\_length\_in\_min from gap\_session group by person\_id" | more

Done in script:

1. **Average catchup time (length of batch catchup packets)**

Catchups lead to skipped hypo scoring (and alerts) for batched sg values and potential poor lead times

Data specs: cleansg.usersg table or mdt.hypo\_featues table, or push\_notifications Kafka topic

To exam the created\_at or updated\_at timestamp

Query outline:

Group the SG record creation time into batches, applie with threshold less than 1 mintue

Compute the average catch up time per user

db2 "select person\_id, AVG(created\_batch)\*5 as average\_catchup\_time from (select \* from (select person\_id, count(created\_min)-1 as created\_batch, created\_min, lag(created\_min) over (order by person\_id, created\_min) as pre\_created\_min from (select person\_id, round\_timestamp(created\_at,'MI') as created\_min from cleansg.usersg) group by person\_id, created\_min order by person\_id) where (pre\_created\_min + 1 MINUTE ) < created\_min and created\_batch < 95 and created\_batch>0) group by person\_id"

Note: The result is set in minutes. If the user is not listed above, the user doesn’t have the catchup packets.

Done is script:

1. **Gaps in SG lead to gaps/loss of IQCast hypo score and potential holes in examining false alerts**

Data specs:

Type of gap to investigate: due to data cleaning rule, assuming no data loss during transaction inside Sugar.IQ (kafka, db2 and streams applications)

Differences between the table of mdt.sensor\_glucose\_data (represents the MDTPRIME and MDTLIVE) & mdt.hypo\_features (HYPOFETUREVECTORS for MDTLIVECLEAN)

Other type of gaps: Loss from the origin (sensor issue, calibrate, signal drop, kafka issue etc.)

Query outline:

Adjust the SG pump time to correct the jitter issue

Divid the SG data flow into continous sessions and gaps (with missing SGs)

Adjust the gap due to the less than 5-min interval

Count the average gap duration per user

If for debugging purpose, suggest to specify the person\_id (for example ID=90006534)

db2 "with session\_group as (select person\_id, sg\_timestamp, TRUNC\_TIMESTAMP(sg\_timestamp - (CAST(row\_number() over (order by sg\_timestamp) as INT) \* 5) MINUTES, 'MI') as cont\_session from mdt.hypo\_features where person\_id=90006534 ),

session\_summary as (select person\_id, cont\_session, min(sg\_timestamp) as start\_sgtime, max(sg\_timestamp) end\_sgtime, TIMESTAMPDIFF(4, char(max(sg\_timestamp)-min(sg\_timestamp)) ) as session\_length\_in\_min from session\_group group by person\_id, cont\_session ),

session\_adjust as (select person\_id, start\_sgtime, end\_sgtime, abs(TIMESTAMPDIFF(4, char(cont\_session - lag(cont\_session) over (order by start\_sgtime) ))) as diff from session\_summary),

session\_remove as (select person\_id, start\_sgtime, end\_sgtime from session\_adjust where diff > 1)

> select person\_id, start\_sgtime, end\_sgtime, TIMESTAMPDIFF(4, char(end\_sgtime-start\_sgtime)) as session\_length\_in\_min, TIMESTAMPDIFF(4, char(start\_sgtime - lag(end\_sgtime) over (order by start\_sgtime))) as previous\_gap\_length\_in\_min from session\_remove"

Done in scripts:

Modified queries for individual level and reports on a daily basis:

1. FDR based on hypo threshold of 70mgdL

Lead time in individual level:

Suggest to extend the 20-hour and 24-hour duration to a longer period if necessary:

On personal level within the last 24-hour contingency:

db2 "with n as (select a.person\_id, count(id) as numb\_false\_positive\_alert from analytics.streams as a inner join (select person\_id, created\_at from analytics.streams where json\_data like '%feedback\_notice.1%' and created\_at >= (CURRENT\_TIMESTAMP - 20 HOURS)) as b on a.person\_id=b.person\_id and abs(TIMESTAMPDIFF(4,char(b.created\_at - 4 hours - a.created\_at))) <=7 where a.json\_data like '%hypo.1%' and a.created\_at >= (CURRENT\_TIMESTAMP - 24 HOURS) group by a.person\_id),

m as (select person\_id, count(id) as total\_numb\_alert from analytics.streams where json\_data like '%hypo.1%' and created\_at >= (CURRENT\_TIMESTAMP - 24 HOURS) group by person\_id)

select m.person\_id, m.total\_numb\_alert, n.numb\_false\_positive\_alert, ROUND((n.numb\_false\_positive\_alert \* 100.0 / m.total\_numb\_alert),2) as \"FDR 70mgdL pct with a false alert in the last 24 hours\"

from m inner join n on m.person\_id = n.person\_id "

On personal level, on daily basis (extend to the past 5 days or 120 hours)

db2 "with n as (select a.person\_id, count(id) as numb\_false\_positive\_alert, date(a.created\_at) as day\_created from analytics.streams as a inner join (select person\_id, created\_at from analytics.streams where json\_data like '%feedback\_notice.1%' and created\_at >= (CURRENT\_TIMESTAMP - 116 HOURS)) as b on a.person\_id=b.person\_id and abs(TIMESTAMPDIFF(4,char(b.created\_at - 4 hours - a.created\_at))) <=7 where a.json\_data like '%hypo.1%' and a.created\_at >= (CURRENT\_TIMESTAMP - 120 HOURS) group by a.person\_id, date(a.created\_at)),

m as (select person\_id, count(id) as total\_numb\_alert, date(created\_at) as day\_created from analytics.streams where json\_data like '%hypo.1%' and created\_at >= (CURRENT\_TIMESTAMP - 120 HOURS) group by person\_id, date(created\_at))

select m.person\_id, m.total\_numb\_alert, n.numb\_false\_positive\_alert, ROUND((n.numb\_false\_positive\_alert \* 100.0 / m.total\_numb\_alert),2) as \"FDR 70mgdL pct with a false alert on daily basis\", m.day\_created

from m inner join n on m.person\_id = n.person\_id and m.day\_created = n.day\_created order by m.person\_id, m.day\_created "

1. FDR based on hypo threshold of 80mgdL

On personal level within the last 24-hour contingency:

db2 "with summary as (select a.person\_id, a.id, a.created\_at as alert\_created from analytics.streams as a inner join (select person\_id, created\_at from analytics.streams where json\_data like '%feedback\_notice.1%' and created\_at >= (CURRENT\_TIMESTAMP - 20 HOURS)) as b on a.person\_id=b.person\_id and abs(TIMESTAMPDIFF(4,char(b.created\_at - 4 hours - a.created\_at))) <=7 where a.json\_data like '%hypo.1%' and a.created\_at >= (CURRENT\_TIMESTAMP - 24 HOURS))

select m.person\_id, total\_numb\_alert, numb\_false\_positive\_alert, ROUND((numb\_false\_positive\_alert \* 100.0 / total\_numb\_alert),2) as FDR\_80mgdL\_pct from (select person\_id, count(id) as total\_numb\_alert from analytics.streams where json\_data like '%hypo.1%' and created\_at >= (CURRENT\_TIMESTAMP - 24 HOURS) group by person\_id) as m inner join (select e.person\_id, count(e.id) as numb\_false\_positive\_alert from summary as e where e.id not in (select distinct c.id from summary as c Inner join mdt.sensor\_glucose\_data as d on d.person\_id=c.person\_id and d.sg\_timestamp < (c.alert\_created + 4 hours) and d.sg\_timestamp > (c.alert\_created -5 minutes) where d.sg<80 and d.sg\_timestamp >= (CURRENT\_TIMESTAMP - 24 HOURS)) group by e.person\_id) as n on m.person\_id=n.person\_id"

On personal level, on daily basis (extend to the past 5 days or 120 hours)

db2 "with summary as (select a.person\_id, a.id, a.created\_at as alert\_created from analytics.streams as a inner join (select person\_id, created\_at from analytics.streams where json\_data like '%feedback\_notice.1%' and created\_at >= (CURRENT\_TIMESTAMP - 116 HOURS)) as b on a.person\_id=b.person\_id and abs(TIMESTAMPDIFF(4,char(b.created\_at - 4 hours - a.created\_at))) <=7 where a.json\_data like '%hypo.1%' and a.created\_at >= (CURRENT\_TIMESTAMP - 120 HOURS))

select m.person\_id, total\_numb\_alert, numb\_false\_positive\_alert, ROUND((numb\_false\_positive\_alert \* 100.0 / total\_numb\_alert),2) as FDR\_80mgdL\_pct, m.day\_created from (select person\_id, count(id) as total\_numb\_alert, date(created\_at) as day\_created from analytics.streams where json\_data like '%hypo.1%' and created\_at >= (CURRENT\_TIMESTAMP - 120 HOURS) group by person\_id, date(created\_at)) as m inner join (select e.person\_id, count(e.id) as numb\_false\_positive\_alert, date(alert\_created) as day\_created from summary as e where e.id not in (select distinct c.id from summary as c Inner join mdt.sensor\_glucose\_data as d on d.person\_id=c.person\_id and d.sg\_timestamp < (c.alert\_created + 4 hours) and d.sg\_timestamp > (c.alert\_created -5 minutes) where d.sg<80 and d.sg\_timestamp >= (CURRENT\_TIMESTAMP - 120 HOURS)) group by e.person\_id, date(e.alert\_created)) as n on m.person\_id=n.person\_id and m.day\_created=n.day\_created order by m.person\_id, m.day\_created"

1. Alert lagging time (for all alerts, including false ones)

On personal level within the last 24-hour contingency:

db2 "with all\_alert as (select person\_id, TIMESTAMPDIFF(4, char(alert\_created - lag(alert\_created) over (partition by person\_id order by alert\_created) )) as alert\_lagging\_min from (select c.person\_id, c.id, c.created\_at as alert\_created from analytics.streams as c where c.json\_data like '%hypo.1%' and c.created\_at >= (CURRENT\_TIMESTAMP - 24 HOURS)))

select person\_id, AVG(alert\_lagging\_min) as avg\_lagging from all\_alert group by person\_id"

Note: "-" means the user has only a single alert created in the past 24-hour.

On personal level, on daily basis (extend to the past 5 days or 120 hours)

db2 "with e as (select person\_id, TIMESTAMPDIFF(4, char(alert\_created - lag(alert\_created) over (partition by person\_id order by alert\_created) )) as alert\_lagging\_min, alert\_created from (select c.person\_id, c.id, c.created\_at as alert\_created from analytics.streams as c where c.json\_data like '%hypo.1%' and c.created\_at >= (CURRENT\_TIMESTAMP - 120 HOURS)))

select person\_id, AVG(alert\_lagging\_min) as avg\_lagging, date(alert\_created) as day\_created from e group by person\_id, date(alert\_created) order by person\_id, date(alert\_created)"

Note: the lagging time could be more than 24 but less than 72 hours.

1. Average lead time

On personal level within the last 24-hour contingency:

db2 "with ab as (select a.id from analytics.streams as a Inner join (select person\_id, id, created\_at from analytics.streams where json\_data like '%feedback\_notice.1%' and created\_at >= (CURRENT\_TIMESTAMP - 20 HOURS)) as b on a.person\_id=b.person\_id and abs(TIMESTAMPDIFF(4, char(b.created\_at- 4 hours - a.created\_at)))<=7 where a.json\_data like '%hypo.1%' and a.created\_at >= (CURRENT\_TIMESTAMP - 24 HOURS)),

f as (select c.person\_id, c.id, c.created\_at as alert\_created from analytics.streams as c inner join (select person\_id, updated\_at from mdt.hypo\_score\_data) as d

on c.person\_id=d.person\_id and c.created\_at < (d.updated\_at - 4 hours) where c.json\_data like '%hypo.1%' and c.created\_at >= (CURRENT\_TIMESTAMP - 24 HOURS) and

c.id not in (select id from ab)),

summary as (select e.person\_id, e.sg, e.sg\_timestamp, e.sg\_timestamp\_tz, f.alert\_created, ROW\_NUMBER() OVER(PARTITION BY e.person\_id,f.alert\_created ORDER BY e.sg\_timestamp) AS rk from mdt.sensor\_glucose\_data as e inner join f

on e.person\_id = f.person\_id and e.sg\_timestamp < (f.alert\_created + 4 hours) and e.sg\_timestamp > (f.alert\_created -5 minutes) where e.sg<70 )

select \* from (select person\_id, count(\*) as \"number true alerts in last 24 hours\", AVG(TIMESTAMPDIFF(4, char(s.sg\_timestamp-s.alert\_created))) as \"average lead time\" from summary s where s.rk=1 group by person\_id order by person\_id) where \"average lead time\">0 "

On personal level, extend to the past 5 days or 120 hours

db2 "with ab as (select a.id from analytics.streams as a Inner join (select person\_id, id, created\_at from analytics.streams where json\_data like '%feedback\_notice.1%' and created\_at >= (CURRENT\_TIMESTAMP - 116 HOURS)) as b on a.person\_id=b.person\_id and abs(TIMESTAMPDIFF(4, char(b.created\_at- 4 hours - a.created\_at)))<=7 where a.json\_data like '%hypo.1%' and a.created\_at >= (CURRENT\_TIMESTAMP - 120 HOURS)),

f as (select c.person\_id, c.id, c.created\_at as alert\_created from analytics.streams as c inner join (select person\_id, updated\_at from mdt.hypo\_score\_data) as d

on c.person\_id=d.person\_id and c.created\_at < (d.updated\_at - 4 hours) where c.json\_data like '%hypo.1%' and c.created\_at >= (CURRENT\_TIMESTAMP - 120 HOURS) and

c.id not in (select id from ab)),

summary as (select e.person\_id, e.sg, e.sg\_timestamp, e.sg\_timestamp\_tz, f.alert\_created, ROW\_NUMBER() OVER(PARTITION BY e.person\_id,f.alert\_created ORDER BY e.sg\_timestamp) AS rk from mdt.sensor\_glucose\_data as e inner join f

on e.person\_id = f.person\_id and e.sg\_timestamp < (f.alert\_created + 4 hours) and e.sg\_timestamp > (f.alert\_created -5 minutes) where e.sg<70 )

select \* from (select person\_id, count(\*) as \"number true alerts on daily basis\", AVG(TIMESTAMPDIFF(4, char(s.sg\_timestamp-s.alert\_created))) as \"average lead time\" from summary s where s.rk=1 group by person\_id order by person\_id) where \"average lead time\">0 "

Modified FDR query to include those ones who don’t have false alert in the base population

1. Users who have false alerts in the last 24 hours (Efeedback)

db2 "select a.person\_id, a.id, a.created\_at as alert\_created from analytics.streams as a inner join (select person\_id, created\_at from analytics.streams where json\_data like '%feedback\_notice.1%' and created\_at >= (CURRENT\_TIMESTAMP - 20 HOURS)) as b on a.person\_id=b.person\_id and abs(TIMESTAMPDIFF(4,char(b.created\_at - 4 hours - a.created\_at))) <=7 where a.json\_data like '%hypo.1%' and a.created\_at >= (CURRENT\_TIMESTAMP - 24 HOURS)"

1. Exclude alerts with 70<=sg<80

db2 "with Efeedback as (select a.person\_id, a.id, a.created\_at as alert\_created from analytics.streams as a inner join (select person\_id, created\_at from analytics.streams where json\_data like '%feedback\_notice.1%' and created\_at >= (CURRENT\_TIMESTAMP - 20 HOURS)) as b on a.person\_id=b.person\_id and abs(TIMESTAMPDIFF(4,char(b.created\_at - 4 hours - a.created\_at))) <=7 where a.json\_data like '%hypo.1%' and a.created\_at >= (CURRENT\_TIMESTAMP - 24 HOURS)) select e.person\_id, count(e.id) as numb\_false\_positive\_alert from Efeedback as e where e.id not in

(select distinct c.id from Efeedback as c Inner join mdt.sensor\_glucose\_data as d on d.person\_id=c.person\_id and d.sg\_timestamp < (c.alert\_created + 4 hours) and d.sg\_timestamp > (c.alert\_created -5 minutes) where d.sg<80 and d.sg\_timestamp >= (CURRENT\_TIMESTAMP - 24 HOURS)) group by e.person\_id"

1. Users who have alerts in the last 24 hours

db2 "select person\_id, count(id) as total\_numb\_alert from analytics.streams where json\_data like '%hypo.1%' and created\_at >= (CURRENT\_TIMESTAMP - 24 HOURS) group by person\_id"

4. Compute the FDR by left outer join

db2 "with Efeedback as (select a.person\_id, a.id, a.created\_at as alert\_created from analytics.streams as a inner join (select person\_id, created\_at from analytics.streams where json\_data like '%feedback\_notice.1%' and created\_at >= (CURRENT\_TIMESTAMP - 20 HOURS)) as b on a.person\_id=b.person\_id and abs(TIMESTAMPDIFF(4,char(b.created\_at - 4 hours - a.created\_at))) <=7 where a.json\_data like '%hypo.1%' and a.created\_at >= (CURRENT\_TIMESTAMP - 24 HOURS)),

n as (select e.person\_id, count(e.id) as numb\_false\_positive\_alert from Efeedback as e where e.id not in

(select distinct c.id from Efeedback as c Inner join mdt.sensor\_glucose\_data as d on d.person\_id=c.person\_id and d.sg\_timestamp < (c.alert\_created + 4 hours) and d.sg\_timestamp > (c.alert\_created -5 minutes) where d.sg<80 and d.sg\_timestamp >= (CURRENT\_TIMESTAMP - 24 HOURS)) group by e.person\_id),

m as (select person\_id, count(id) as total\_numb\_alert from analytics.streams where json\_data like '%hypo.1%' and created\_at >= (CURRENT\_TIMESTAMP - 24 HOURS) group by person\_id),

pfdr as (select m.person\_id, m.total\_numb\_alert, n.numb\_false\_positive\_alert, ROUND((n.numb\_false\_positive\_alert \* 100.0 / m.total\_numb\_alert),2) as \"FDR 70mgdL in the last 24 hours\" from m left outer join n on m.person\_id = n.person\_id) select count(person\_id) as total\_users\_who\_have\_alerts, SUM(total\_numb\_alert) as total\_numb\_alerts, SUM(numb\_false\_positive\_alert) as total\_numb\_false\_alerts, AVG(total\_numb\_alert) as average\_numb\_alerts, AVG(numb\_false\_positive\_alert) as average\_numb\_false\_alerts, ROUND((SUM(numb\_false\_positive\_alert) \* 100.0 / SUM(total\_numb\_alert)),2) as \"FDR 70mgdL per user in the last 24 hours\" from pfdr"