docker run -d --rm -p 3000:3000 -p 15432:5432 --name=de-sprint-1-server-local sindb/sprint-1:latest

CREATE TABLE clients\_cluster\_metrics\_m (

month date,

client\_id bigint,

utm\_campaign varchar(30),

reg\_code varchar(3) ,

total\_events bigint,

visit\_events bigint,

registration\_events  dbigint,

login\_events bigint,

visit\_to\_login\_events    dbigint,

total\_pay\_events bigint,

accepted\_method\_actions  bigint,

avg\_payment  double precision,

made\_payments    bigint,

sum\_payments     double precision,

rejects\_share    double precision

);

insert into clients ( client\_id , client\_firstname , client\_lastname  , client\_email  ,  client\_phone  ,  client\_city  ,  age  )

Select client\_id , client\_firstname , client\_lastname  , client\_email  ,  client\_phone  ,  client\_city  ,  age

from (select b.\* ,

Case when  a is null then 'I'

when a = b or b is null then ''

 else 'U' END as action

from   clients\_inc as b left join clients as a on a.client\_id =b. client\_id) as t

WHERE t.action ='I';

Update clients

SET  client\_firstname = c.client\_firstname , client\_lastname = c.client\_lastname , client\_email =c.client\_email  ,  client\_phone = c.client\_phone ,  client\_city =c.client\_city   ,  age =c.age

from (select b.\* ,

Case when  a is null then 'I'

when a = b or b is null then ''

 else 'U' END as action

from   clients\_inc as b left join clients as a on a.client\_id =b. client\_id) as c

WHERE  clients.client\_id =c.client\_id and c.action ='U'

/flush\_db.sh  - база к изначальному состоянию

SELECT client\_id, phone

FROM

(SELECT client\_id,

phone,

ROW\_NUMBER() OVER (PARTITION BY client\_id

ORDER BY created\_at DESC) row\_num

FROM user\_contacts) t

WHERE row\_num = 1;

SELECT regexp\_replace (phone,'[() ]','', 'g') AS phone

FROM user\_contacts;

ALTER TABLE user\_contacts ADD CONSTRAINT user\_contacts\_client\_id\_fkey

FOREIGN KEY (client\_id)

REFERENCES user\_attributes (client\_id);

ALTER TABLE user\_contacts ALTER COLUMN client\_id SET NOT NULL

ALTER TABLE user\_contacts ALTER COLUMN phone SET NOT NULL

SELECT

client\_id ,

CAST(DATE\_TRUNC ('month', hitdatetime) as date) as month ,

COUNT (CASE WHEN  action = 'visit' THEN 1

             END) as visit\_events,

COUNT (CASE WHEN   action = 'registration' THEN 1

             END) as registration\_events,

COUNT (CASE WHEN   action = 'login' THEN 1

             END) as login\_events

FROM user\_activity\_log

GROUP BY client\_id,  month

With tt as (SELECT client\_id, hitdatetime,

          lag(action) OVER (PARTITION BY client\_id

                                 ORDER BY hitdatetime) as prev\_event

FROM user\_activity\_log)

SELECT a.client\_id ,

CAST(DATE\_TRUNC ('month', a.hitdatetime) as date) as month ,

COUNT (CASE WHEN  action = 'login'  and prev\_event = 'visit' THEN 1

         END) as visit\_to\_login\_events

FROM user\_activity\_log a left join tt on tt.client\_id = a.client\_id and tt.hitdatetime = a.hitdatetime

GROUP BY a.client\_id,  month

SELECT client\_id,

 COALEsce(CAST(nullif(count (CASE

 WHEN extract(hour from  hitdatetime) between 12 and 17 THEN action else null END ),0) as float)/

 cast(nullif (count(action),0) as float),0)\*100 as daily\_actions\_pct

from user\_payment\_log

group by client\_id

<iframe src="https://team.effem.com/sites/DigitalSOP/\_layouts/15/Doc.aspx?sourcedoc={904a3f6a-ea31-42cf-9a8f-c1b94a1ee583}&amp;action=embedview&amp;wdAr=1.7777777777777777" width="476px" height="288px" frameborder="0">This is an embedded <a target="\_blank" href="https://office.com">Microsoft Office</a> presentation, powered by <a target="\_blank" href="https://office.com/webapps">Office</a>.</iframe>

-- CREATE TABLE clients\_cluster\_metrics\_m (

-- month date,

-- client\_id bigint,

-- utm\_campaign varchar(30),

-- reg\_code varchar(3) ,

-- total\_events bigint,

-- visit\_events bigint,

-- registration\_events   bigint,

-- login\_events bigint,

-- visit\_to\_login\_events     bigint,

-- total\_pay\_events bigint,

-- accepted\_method\_actions   bigint,

-- avg\_payment   double precision,

-- made\_payments     bigint,

-- sum\_payments  double precision,

-- rejects\_share     double precision

-- );

truncate table clients\_cluster\_metrics\_m;

insert into clients\_cluster\_metrics\_m

with act\_lg as (

    select date\_trunc('month',hitdatetime)::date "month",

        client\_id,

        count(1) total\_events,

        count(case when "action" = 'visit' then 1 end) visit\_events,

        count(case when "action" = 'registration' then 1 end) registration\_events,

        count(case when "action" = 'login' then 1 end) login\_events,

        count(case when ("action" = 'login') and (prev\_action = 'visit') then 1 end) visit\_to\_login\_events

    from (

        select \*,

            lag("action") over (partition by client\_id order by hitdatetime) prev\_action

        from user\_activity\_log

        where extract(year from hitdatetime) = 2021

            and "action" != 'N/A'

        )t

    group by 1,2

    ),

pmnts as (

    select date\_trunc('month',hitdatetime)::date "month",

        client\_id,

        count(1) total\_pay\_events,

        count(case when "action" = 'accept-method' then 1 end) accepted\_method\_actions,

        count(case when "action" = 'make-payment' then 1 end) made\_payments,

        avg(case when "action" = 'make-payment' then payment\_amount else 0 end) avg\_payment,

        sum(case when "action" = 'make-payment' then payment\_amount else 0 end) sum\_payments,

        sum(case when "action" = 'reject-payment' then payment\_amount else 0 end)

            / nullif(sum(case when "action" = 'make-payment' then payment\_amount else 0 end),0) rejects\_share

    from user\_payment\_log

    where extract(year from hitdatetime) = 2021

    group by 1,2

)

select coalesce(a."month",p."month") "month",

    ua.client\_id,

    ua.utm\_campaign,

    coalesce(a.total\_events,0) total\_events,

    coalesce(a.visit\_events,0) visit\_events,

    coalesce(a.registration\_events,0) registration\_events,

    coalesce(a.login\_events,0) login\_events,

    coalesce(a.visit\_to\_login\_events,0) visit\_to\_login\_events,

    coalesce(p.total\_pay\_events,0) total\_pay\_events,

    coalesce(p.accepted\_method\_actions,0) accepted\_method\_actions,

    coalesce(p.avg\_payment,0) avg\_payment,

    coalesce(p.made\_payments,0) made\_payments,

    coalesce(p.sum\_payments,0) sum\_payments,

    p.rejects\_share

from act\_lg a

full join pmnts p on p."month" = a."month"

                and p.client\_id = a.client\_id

join user\_attributes ua on ua.client\_id = coalesce(a.client\_id,p.client\_id);

-- добавьте свой код ниже

insert into clients\_cluster\_metrics\_m

with act\_lg as (

    select date\_trunc('month',hitdatetime)::date "month",

        client\_id,

        count(1) total\_events,

        count(case when "action" = 'visit' then 1 end) visit\_events,

        count(case when "action" = 'registration' then 1 end) registration\_events,

        count(case when "action" = 'login' then 1 end) login\_events,

        count(case when ("action" = 'login') and (prev\_action = 'visit') then 1 end) visit\_to\_login\_events

    from (

        select \*,

            lag("action") over (partition by client\_id order by hitdatetime) prev\_action

        from user\_activity\_log\_arch

        -- where extract(year from hitdatetime) in (2019, 2020)

        --     and "action" != 'N/A'

        )t

    group by 1,2

    ),

pmnts as (

    select date\_trunc('month',hitdatetime)::date "month",

        client\_id,

        count(1) total\_pay\_events,

        count(case when "action" = 'accept-method' then 1 end) accepted\_method\_actions,

        count(case when "action" = 'make-payment' then 1 end) made\_payments,

        avg(case when "action" = 'make-payment' then payment\_amount else 0 end) avg\_payment,

        sum(case when "action" = 'make-payment' then payment\_amount else 0 end) sum\_payments,

        sum(case when "action" = 'reject-payment' then payment\_amount else 0 end)

            / nullif(sum(case when "action" = 'make-payment' then payment\_amount else 0 end),0) rejects\_share

    from user\_payment\_log\_arch

    -- where extract(year from hitdatetime) in (2019, 2020)

    group by 1,2

)

select coalesce(a."month",p."month") "month",

    ua.client\_id,

    ua.utm\_campaign,

    coalesce(a.total\_events,0) total\_events,

    coalesce(a.visit\_events,0) visit\_events,

    coalesce(a.registration\_events,0) registration\_events,

    coalesce(a.login\_events,0) login\_events,

    coalesce(a.visit\_to\_login\_events,0) visit\_to\_login\_events,

    coalesce(p.total\_pay\_events,0) total\_pay\_events,

    coalesce(p.accepted\_method\_actions,0) accepted\_method\_actions,

    coalesce(p.avg\_payment,0) avg\_payment,

    coalesce(p.made\_payments,0) made\_payments,

    coalesce(p.sum\_payments,0) sum\_payments,

    p.rejects\_share

from act\_lg a

full join pmnts p on p."month" = a."month"

                and p.client\_id = a.client\_id

join user\_attributes ua on ua.client\_id = coalesce(a.client\_id,p.client\_id);

select count (\*) from  clients\_cluster\_metrics\_m

insert into  load\_dates (date, batch\_id )

SELECT Max(month ) ,450 from clients\_cluster\_metrics\_m

insert into  load\_dates (date)

SELECT Max(month ) from clients\_cluster\_metrics\_m

-- добавьте код сюда

insert into clients\_cluster\_metrics\_m

with act\_lg as (SELECT date\_trunc('month',hitdatetime)::date "month",

        client\_id,

        count(1) total\_events,

        count(case when "action" = 'visit' then 1 end) visit\_events,

        count(case when "action" = 'registration' then 1 end) registration\_events,

        count(case when "action" = 'login' then 1 end) login\_events,

        count(case when ("action" = 'login') and (prev\_action = 'visit') then 1 end) visit\_to\_login\_events

    from (

        select \*,

            lag("action") over (partition by client\_id order by hitdatetime) prev\_action

        from user\_activity\_log

        where hitdatetime > (SELECT  date from load\_dates order by date desc limit 1 )

            and "action" != 'N/A'

        )t

    group by 1,2

    ),

pmnts as (

    select date\_trunc('month',hitdatetime)::date "month",

        client\_id,

        count(1) total\_pay\_events,

        count(case when "action" = 'accept-method' then 1 end) accepted\_method\_actions,

        count(case when "action" = 'make-payment' then 1 end) made\_payments,

        avg(case when "action" = 'make-payment' then payment\_amount else 0 end) avg\_payment,

        sum(case when "action" = 'make-payment' then payment\_amount else 0 end) sum\_payments,

        sum(case when "action" = 'reject-payment' then payment\_amount else 0 end)

            / nullif(sum(case when "action" = 'make-payment' then payment\_amount else 0 end),0) rejects\_share

    from user\_payment\_log

    where  hitdatetime > (SELECT  date from load\_dates order by date desc limit 1)

    group by 1,2

)

select coalesce(a."month",p."month") "month",

    ua.client\_id,

    ua.utm\_campaign,

    coalesce(a.total\_events,0) total\_events,

    coalesce(a.visit\_events,0) visit\_events,

    coalesce(a.registration\_events,0) registration\_events,

    coalesce(a.login\_events,0) login\_events,

    coalesce(a.visit\_to\_login\_events,0) visit\_to\_login\_events,

    coalesce(p.total\_pay\_events,0) total\_pay\_events,

    coalesce(p.accepted\_method\_actions,0) accepted\_method\_actions,

    coalesce(p.avg\_payment,0) avg\_payment,

    coalesce(p.made\_payments,0) made\_payments,

    coalesce(p.sum\_payments,0) sum\_payments,

    p.rejects\_share

from act\_lg a

full join pmnts p on p."month" = a."month"

                and p.client\_id = a.client\_id

join user\_attributes ua on ua.client\_id = coalesce(a.client\_id,p.client\_id);

-- добавьте код сюда

--drop table clients\_cluster\_metrics\_m

CREATE TABLE clients\_cluster\_metrics\_m (

month date,

client\_id bigint,

utm\_campaign varchar(30),

reg\_code varchar(3) ,

total\_events bigint,

visit\_events bigint,

registration\_events   bigint,

login\_events bigint,

visit\_to\_login\_events     bigint,

total\_pay\_events bigint,

accepted\_method\_actions   bigint,

avg\_payment   double precision,

made\_payments     bigint,

sum\_payments  double precision,

rejects\_share     double precision ,

CONSTRAINT clients\_cluster\_metrics\_m\_month\_client\_id\_pkey PRIMARY KEY (month, client\_id),

CONSTRAINT  clients\_cluster\_metrics\_m\_client\_id\_fkey FOREIGN KEY (client\_id) REFERENCES user\_attributes(client\_id)

);

truncate table clients\_cluster\_metrics\_m;

insert into clients\_cluster\_metrics\_m

with act\_lg as (

    select date\_trunc('month',hitdatetime)::date "month",

        client\_id,

        count(1) total\_events,

        count(case when "action" = 'visit' then 1 end) visit\_events,

        count(case when "action" = 'registration' then 1 end) registration\_events,

        count(case when "action" = 'login' then 1 end) login\_events,

        count(case when ("action" = 'login') and (prev\_action = 'visit') then 1 end) visit\_to\_login\_events

    from (

        select \*,

            lag("action") over (partition by client\_id order by hitdatetime) prev\_action

        from user\_activity\_log

        where extract(year from hitdatetime) = 2021

            and "action" != 'N/A'

        )t

    group by 1,2

    ),

pmnts as (

    select date\_trunc('month',hitdatetime)::date "month",

        client\_id,

        count(1) total\_pay\_events,

        count(case when "action" = 'accept-method' then 1 end) accepted\_method\_actions,

        count(case when "action" = 'make-payment' then 1 end) made\_payments,

        avg(case when "action" = 'make-payment' then payment\_amount else 0 end) avg\_payment,

        sum(case when "action" = 'make-payment' then payment\_amount else 0 end) sum\_payments,

        sum(case when "action" = 'reject-payment' then payment\_amount else 0 end)

            / nullif(sum(case when "action" = 'make-payment' then payment\_amount else 0 end),0) rejects\_share

    from user\_payment\_log

    where extract(year from hitdatetime) = 2021

    group by 1,2

)

select coalesce(a."month",p."month") "month",

    ua.client\_id,

    ua.utm\_campaign,

    coalesce(a.total\_events,0) total\_events,

    coalesce(a.visit\_events,0) visit\_events,

    coalesce(a.registration\_events,0) registration\_events,

    coalesce(a.login\_events,0) login\_events,

    coalesce(a.visit\_to\_login\_events,0) visit\_to\_login\_events,

    coalesce(p.total\_pay\_events,0) total\_pay\_events,

    coalesce(p.accepted\_method\_actions,0) accepted\_method\_actions,

    coalesce(p.avg\_payment,0) avg\_payment,

    coalesce(p.made\_payments,0) made\_payments,

    coalesce(p.sum\_payments,0) sum\_payments,

    p.rejects\_share

from act\_lg a

full join pmnts p on p."month" = a."month"

                and p.client\_id = a.client\_id

join user\_attributes ua on ua.client\_id = coalesce(a.client\_id,p.client\_id);

-- добавьте свой код ниже

insert into clients\_cluster\_metrics\_m

with act\_lg as (

    select date\_trunc('month',hitdatetime)::date "month",

        client\_id,

        count(1) total\_events,

        count(case when "action" = 'visit' then 1 end) visit\_events,

        count(case when "action" = 'registration' then 1 end) registration\_events,

        count(case when "action" = 'login' then 1 end) login\_events,

        count(case when ("action" = 'login') and (prev\_action = 'visit') then 1 end) visit\_to\_login\_events

    from (

        select \*,

            lag("action") over (partition by client\_id order by hitdatetime) prev\_action

        from user\_activity\_log\_arch

        -- where extract(year from hitdatetime) in (2019, 2020)

        --     and "action" != 'N/A'

        )t

    group by 1,2

    ),

pmnts as (

    select date\_trunc('month',hitdatetime)::date "month",

        client\_id,

        count(1) total\_pay\_events,

        count(case when "action" = 'accept-method' then 1 end) accepted\_method\_actions,

        count(case when "action" = 'make-payment' then 1 end) made\_payments,

        avg(case when "action" = 'make-payment' then payment\_amount else 0 end) avg\_payment,

        sum(case when "action" = 'make-payment' then payment\_amount else 0 end) sum\_payments,

        sum(case when "action" = 'reject-payment' then payment\_amount else 0 end)

            / nullif(sum(case when "action" = 'make-payment' then payment\_amount else 0 end),0) rejects\_share

    from user\_payment\_log\_arch

    -- where extract(year from hitdatetime) in (2019, 2020)

    group by 1,2

)

select coalesce(a."month",p."month") "month",

    ua.client\_id,

    ua.utm\_campaign,

    coalesce(a.total\_events,0) total\_events,

    coalesce(a.visit\_events,0) visit\_events,

    coalesce(a.registration\_events,0) registration\_events,

    coalesce(a.login\_events,0) login\_events,

    coalesce(a.visit\_to\_login\_events,0) visit\_to\_login\_events,

    coalesce(p.total\_pay\_events,0) total\_pay\_events,

    coalesce(p.accepted\_method\_actions,0) accepted\_method\_actions,

    coalesce(p.avg\_payment,0) avg\_payment,

    coalesce(p.made\_payments,0) made\_payments,

    coalesce(p.sum\_payments,0) sum\_payments,

    p.rejects\_share

from act\_lg a

full join pmnts p on p."month" = a."month"

                and p.client\_id = a.client\_id

join user\_attributes ua on ua.client\_id = coalesce(a.client\_id,p.client\_id);

docker run -d --rm -p 15432:5432 -p 3000:3000 --name=de-project-sprint-1-server-local sindb/project-sprint-1:latest

 ghp\_9vUPTLrTebYg7DLCUavGfYX7TcoIyC2D7tsD

ghp\_9vUPTLrTebYg7DLCUavGfYX7TcoIyC2D7tsD

**ALTER** **TABLE** lesson47.task3\_record\_log

**ALTER** **COLUMN** row\_dttm **SET** **DATA** **TYPE** **timestamp** **WITH** **time** **ZONE**

**USING** **to\_timestamp**(row\_dttm, 'YYYY-MM-DD hh24:mi:ss') **AT** **time** **ZONE** 'Europe/Moscow';

**SELECT** row\_dttm

**FROM** lesson47.task3\_record\_log **AS** t

**WHERE** row\_id = 451;