Task 1

```
with a as (
	select EOMONTH(ord_datetime) report_date, count(1) q, gr_name
	from Orders o
	left join Analysis a
		on o.ord_an = a.an_id
	left join Groups g
		on a.an_group = g.gr_id
	group by gr_name, EOMONTH(ord_datetime)
	)
select  gr_name
		, report_date
		, SUM(q) OVER (PARTITION BY gr_name ORDER BY report_date RANGE
UNBOUNDED PRECEDING) s
from a
order by gr_name, report_date;
```

Task 2

Существует несколько способов дедубликации данных. Для выбора более оптимального решения, я бы посмотрела, как быстро каждое из них работает на конкретной базе с конкретными данными

```
select client_id, client_name, client_balance_date, max(client_balance_value) client_balance_value
from ClientBalance
group by client_id, client_name, client_balance_date;

select client_id, client_name, client_balance_date, client_balance_value
from ClientBalance
group by client_id, client_name, client_balance_date, client_balance_value;

select client_id, client_name, client_balance_date, client_balance_value
from ClientBalance
union
select top 0 client_id, client_name, client_balance_date, client_balance_value
from ClientBalance;

select client_id, client_name, client_balance_date, client_balance_value
from ClientBalance
intersect
select client_id, client_name, client_balance_date, client_balance_value
from ClientBalance;

select distinct client_id, client_name, client_balance_date, client_balance_value
from ClientBalance;
```

Task 3

```
with a as (
    select case when a.color > b.color then b.color else a.color end color1
        , case when a.color > b.color then a.color else b.color end color2
    from ColorsTable a, ColorsTable b
    where a.color <> b.color
    )
select color1, color2 from a
group by color1, color2;
```

Task 4

В задании я интерпретировала данные т.о., что они записываются в таблицу последовательно в хронологическом порядке (по мере возникновения и завершения событий), значит, пары start_second и end_second можно объединить, предварительно отсортировав по времени:

```
with a as (
    select i.*, row_number() over(partition by event_id order by start_second) rn
    from tbl_input i
    ), b as (
    select o.*, row_number() over(partition by event_id order by end_second) rn
    from tbl_output o
    )
select a.event_id, start_second, end_second, (end_second - start_second) duration
from a join b
    on a.event_id = b.event_id and a.rn = b.rn;
```

Task 5

В задании сказано, что для каждой записи необходимо брать ID первой сессии при их объединении (если разница между сессиями менее 15 минут), но не увидела, где в результате необходимо вывести ID, поэтому сохранила промежуточный результат во временную таблицу, которая содержит данную информацию.

Далее, на основе нее, получила запросы для требуемых результатов:

```
with a as (
    select ID, RemoteUserId
    , lag(SessionLastUsed) OVER (partition by RemoteUserId order by SessionStart)
SessionLastUsed_prev
    , SessionStart
    , SessionLastUsed
    ,lead(SessionStart) OVER (partition by RemoteUserId order by SessionStart) SessionStart_next
    ,row_number() over(order by RemoteUserId, SessionStart) rn
    from SOAPsessions
    )
    , b as (
    select a.*
        , DATEDIFF(mi, SessionLastUsed_prev, SessionStart) mi
        , iif(SessionLastUsed_prev is null, ID, iif(DATEDIFF(mi, SessionLastUsed_prev,
SessionStart) > 15, ID, null)) ID_NEW
```

```
        , iif(SessionLastUsed_prev is null, SessionStart, iif(DATEDIFF(mi, SessionLastUsed_prev,
SessionStart) > 15, SessionStart, null)) SessionStart_NEW
    from a
    )
    , c as (
    select RemoteUserId, ID_NEW, SessionStart_NEW, lead(SessionStart_NEW) OVER (partition
by RemoteUserId order by rn) SessionStart_new_next
    from b
    where ID_NEW is not null
    )

    select ID_NEW ID, SessionStart_NEW SessionStart, max(a.SessionLastUsed)
SessionLastUsed
    into #Sessions
    from a
        left join c
            on a.RemoteUserId = c.RemoteUserId and a.SessionStart >= c.SessionStart_NEW and
a.SessionStart < isnull(c.SessionStart_new_next, getdate())
    group by ID_NEW, SessionStart_NEW;
```

1) Кол-во сессий, за каждую дату

```
    select cast(SessionStart as date) report_date, count(1) q_sessions
    from #Sessions
        group by cast(SessionStart as date);
```

2) Средняя, минимальная и максимальная продолжительность сессии в минутах

```
    with a as (
        select DATEDIFF(mi, SessionStart, SessionLastUsed) mi
        from #Sessions
    )
    select avg(mi) avg_sess, min(mi) min_sess, max(mi) max_sess
    from a;
```