100 SQL Questions StrataScratch

Hard SQL Questions

▼ 1 (ID 9952) **Name to Medal Connection**

```
-- ESPN | Hard | General Practice | ID 9952
/*
Find the connection between the number of letters in the athlete's first name and the number of medals won for each type for medal, inc
Output the length of the name along with the corresponding number of no medals, bronze medals, silver medals, and gold medals
*/

with
cte1 as
    (select
        *
        , case when medal = 'Gold' then 1 else 0 end as is_gold -- does she/he have medal?
        , case when medal = 'Bronze' then 1 else 0 end as is_bronze
        , case when medal = 'Silver' then 1 else 0 end as is_silver
        , case when medal is null then 1 else 0 end as is_no_medal
        , char_length(split_part(name, ' ', 1))::char as name_len -- take the first name then count the length
    from olympics_athletes_events)

select
    name_len
    , sum(is_gold) as num_gold
    , sum(is_silver) as num_silver
    , sum(is_bronze) as num_bronze
    , sum(is_no_medal) as num_no_medal
from cte1
group by 1
order by 1
;
```

▼ given table & result

olympics_athletes_events — Preview

id:	int
name:	varchar
sex:	varchar
age:	float
height:	float
weight:	datetime
team:	varchar
noc:	varchar
games:	varchar
year:	int
season:	varchar
city:	varchar
sport:	varchar
event:	varchar
medal:	varchar
non_team:	datetime

Output — View the output in a separate browser tab

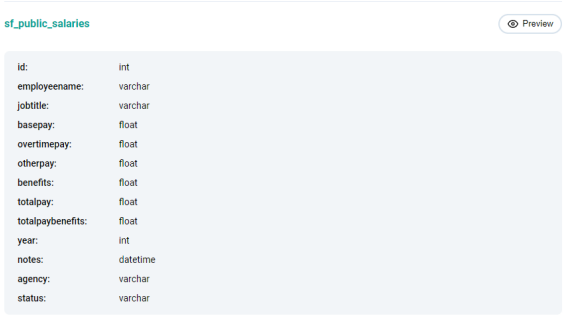
name_len	num_gold	num_silver	num_bronze	num_no_medal
1	0	2	2	4
2	0	1	0	5
3	3	1	3	10
4	6	9	8	27
5	12	7	2	48
6	13	12	6	71
7	5	8	3	35
8	4	2	3	23
9	5	2	0	12

▼ 2 (ID 9989) **Highest Paid City Employees**

```
-- City of San Francisco | Hard | General Practice | ID 9989
/*
Find the top 2 highest paid City employees for each job title.
Output the job title along with the corresponding highest and second-highest paid employees.
*/

with
```

```
cte as
    (select
        jobtitle
        , employeename
        , totalpay
        , rank() over(partition by jobtitle order by totalpay desc) as ranking
    from sf_public_salaries)

select
    *
from cte
where ranking <= 2
;
```

▼ given table & result



jobtitle	employeename	totalpay	ranking
CAPTAIN III (POLICE DEPARTMENT)	PATRICIA JACKSON	297608.92	1
CAPTAIN III (POLICE DEPARTMENT)	ANNA BROWN	238551.88	2
Deputy Sheriff	Lance A Obtinalla Jr	102504.84	1
Deputy Sheriff	Petra Hahn	102369.64	2
Eligibility Worker	Kenia C Coronado	39666.28	1
Eligibility Worker	Tretha T Stroughter	28182.36	2
EMT/Paramedic/Firefighter	Alexander M Lamond	106319.44	1
EMT/Paramedic/Firefighter	Teresa L Cavanaugh	105207.29	2
Estate Investigator	Grace Lin	82901.08	1
Estate Investigator	Andrew G Chen	82052.03	2
Firefighter	Gregory B Bovo	23757.5	1
Firefighter	Ryan C Crow	23757.5	1
GENERAL MANAGER-METROPOLITAN TRANSIT AUTHORITY	NATHANIEL FORD	567595.43	1
GENERAL MANAGER-METROPOLITAN TRANSIT AUTHORITY	EDWARD REISKIN	230827.12	2
IS Programmer Analyst	Dennis J Gerbino	67667.55	1
IS Programmer Analyst-Senior	Clark Bell	72045.4	1

▼ 3 (ID 9844) **Find all workers who joined on February 2014**

```
-- Amazon | Hard | General Practice | ID 9844
/*
Find all workers who joined on February 2014.
*/

select *
from worker
where extract(month from joining_date) = 02
    and extract(year from joining_date) = 2014
;
```

▼ given table & result

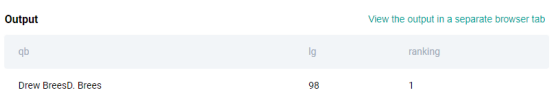
Output View the output in a separate browser tab

worker_id	first_name	last_name	salary	joining_date	department
1	Monika	Arora	100000	2014-02-20 09:00:00	HR
3	Vishal	Singhal	300000	2014-02-20 09:00:00	HR
4	Amitah	Singh	500000	2014-02-20 09:00:00	Admin

▼ 4 (ID 9966) **Quarterback With The Longest Throw**

```
-- ESPN | Hard | General Practice | ID 9966
/*
Find the quarterback who threw the longest throw in 2016. Output the quarterback name along with their corresponding longest throw.
The 'lg' column contains the longest completion by the quarterback.
*/

with
cte1 as
    (select
        qb
        , lg::integer
    from qbstats_2015_2016
    where year = 2016 and lg not like '%t')

, cte2 as
    (select
        qb
        , lg
    from qbstats_2015_2016
    where year = 2016 and lg like '%t')

, cte3 as
    (select
        qb
        , replace(lg, 't', '')::integer as lg
    from cte2)

, cte_rank as
    (select
        *
        , rank() over(order by lg desc) as ranking
    from (select *
        from cte1
        union
        select *
        from cte3) x)

select *
from cte_rank
where ranking = 1
;
```

▼ given table & result

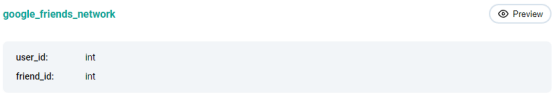


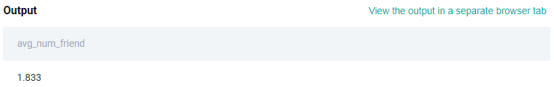
▼ 5 (ID 9822) **Find the average number of friends a user has**

```
-- Google | Hard | General Practice | ID 9822
/*
Find the average number of friends a user has.
*/

select
    avg(count_friend) as avg_num_friend
from(select
        user_id
        , count(distinct friend_id) as count_friend
    from google_friends_network
```

```
    group by 1) x
;
```

▼ given table & result

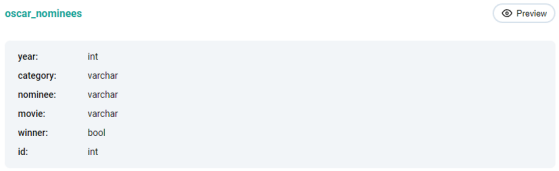


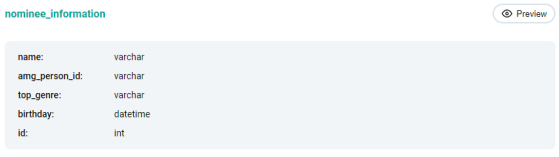


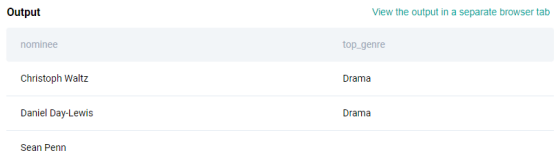
▼ 6 (ID 10171) **Find the genre of the person with the most number of Oscar winnings**

```
-- Netflix | Hard | General Practice | ID 10171
/*
Find the genre of the person with the most number of oscar winnings.
If there are more than one person with the same number of oscar wins, return the first one in alphabetic order based on their name. Use
*/

with
cte_winner as
    (select
        nominee
        , count(winner) as num_win
    from oscar_nominees
    where winner = TRUE
    group by 1)

, cte_rank_one as
    (select *
    from(select
            *
            , dense_rank() over(order by num_win desc) as ranking
        from cte_winner) x
    where ranking = 1
    order by nominee asc)

select
    o.nominee
    , i.top_genre
from cte_rank_one o
left join nominee_information i on o.nominee = i.name

;
```

▼ given table & result





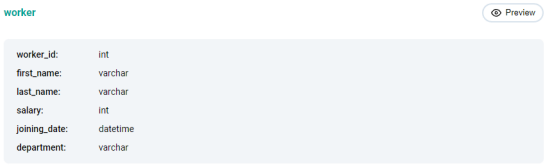


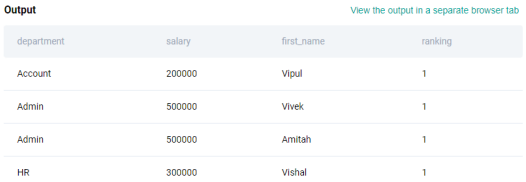
▼ 7 (ID 9865) **Highest Salaried Employees**

```
-- Amazon | Hard | General Practice | ID 9865
/*
Find the employee with the highest salary in each department.
Output the department name, employee's first name, and the salary.
```

```
*/

select *
from (select
        department
        , salary
        , first_name
        , rank() over(partition by department order by salary desc) as ranking
    from worker) x
where ranking = 1
;
```

▼ given table & result

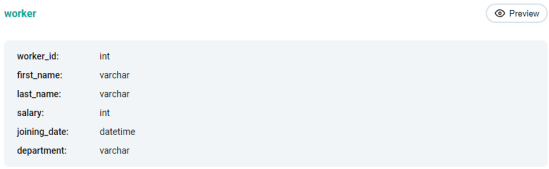


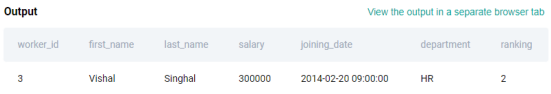


▼ 8 (ID 9857) **Find the second highest salary without using ORDER BY**

```
-- Amazon | Hard | General Practice | ID 9857
/*
Find the second highest salary without using ORDER BY.
*/

select
    *
from(select
        *
        , dense_rank() over(order by salary desc) as ranking
    from worker) x
where ranking = 2
;
```

▼ given table & result

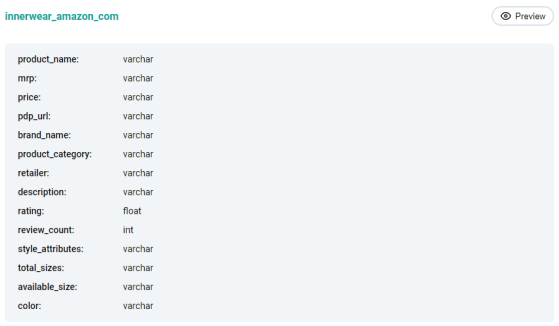


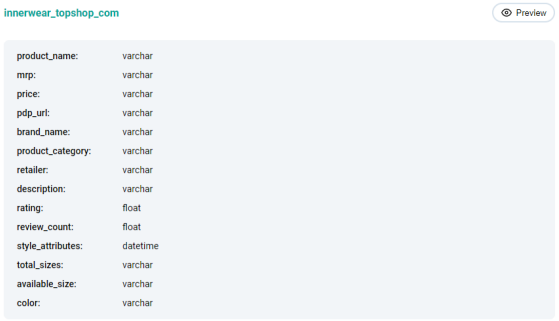


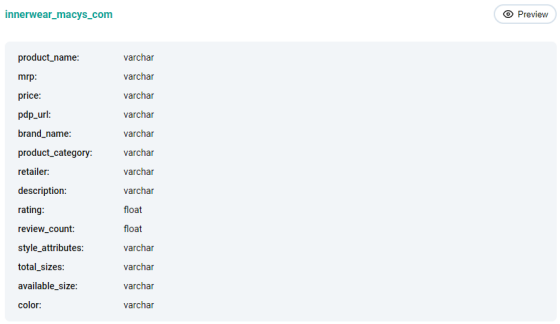
▼ 9 (ID 9608) **Exclusive Amazon Products**

```
-- Amazon | Hard | General Practice | ID 9608
/*
Find products which are exclusive to only Amazon and therefore not sold at Top Shop and Macy's. Your output should include the product
Two products are considered equal if they have the same product name and same maximum retail price (mrp column).
*/

select
    product_name
    , brand_name
    , price
    , rating
from innerwear_amazon_com
where product_name not in (select product_name from innerwear_topshop_com)
    and product_name not in (select product_name from innerwear_macys_com)

;
```

▼ given table & result







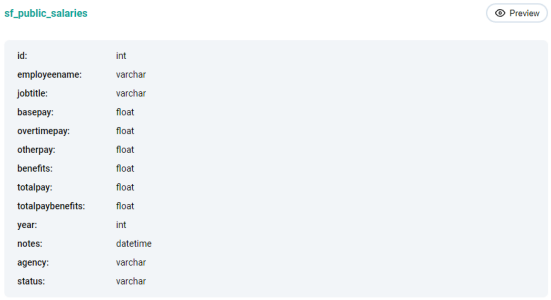
product_name	brand_name	price	rating
Calvin Klein Women's Bottoms Up Hipster Panty	Calvin-Klein	$11.00	4.5
Wacoal Women's Retro Chic Underwire Bra	Wacoal	$60.00	4.4
Calvin Klein Women's Carousel 3 Pack Thong	Calvin-Klein	$19.99	4
b.tempt'd by Wacoal Women's Lace Kiss Bralette	b-temptd	$11.65	4
Wacoal Women's Front Close T-Back Bra	Wacoal	$46.00	4.2
Calvin Klein Women's Modern Cotton Bralette and Bikini Set	Calvin-Klein	$44.00	4.6
Calvin Klein Women's 3 Pack Invisibles Hipster Panty	Calvin-Klein	$29.75	3.9
Wacoal Womens Basic Beauty Contour T-Shirt Bra	Wacoal	$11.95	4.2
Wacoal Women's Underwire Sport Bra	Wacoal	$65.00	4.3
Hanky Panky Women's Vikini Panty	Hanky-Panky	$30.00	4
Wacoal Women's Halo Underwire Bra	Wacoal	$48.00	4.4
Wacoal Women's Basic Beauty Contour Bra	Wacoal	$55.00	4
Wacoal Women's Retro Chic Underwire Bra	Wacoal	$60.00	4.5
Wacoal Women's How Perfect Soft Cup Bra	Wacoal	$51.52	4.2
Wacoal Women's Body By Wacoal Underwire Bra	Wacoal	$43.00	4.4
Wacoal Women's How Perfect Soft Cup Bra	Wacoal	$60.00	4.1
Wacoal Women's Retro Chic Contour Bra	Wacoal	$65.00	4.2

▼ 10 (ID 10145) **Make a pivot table to find the highest payment in each year for each employee**

```
-- City of San Francisco | Hard | General Practice | ID 10145
/*
Make a pivot table to find the highest payment in each year for each employee.
Find payment details for 2011, 2012, 2013, and 2014.
Output payment details along with the corresponding employee name.
Order records by the employee name in ascending order
*/

with
cte_table_rank as
    (select
        *
        , dense_rank() over(partition by year order by totalpay desc) as ranking
    from sf_public_salaries)

select
    year
    , employeename
    , ranking
from cte_table_rank
where year in (2011, 2012, 2013, 2014)
```

```
    and ranking = 1
;
```

▼ given table & result



sf_public_salaries

Preview

id:	int
employeename:	varchar
jobtitle:	varchar
basepay:	float
overtimepay:	float
otherpay:	float
benefits:	float
totalpay:	float
totalpaybenefits:	float
year:	int
notes:	datetime
agency:	varchar
status:	varchar

Output

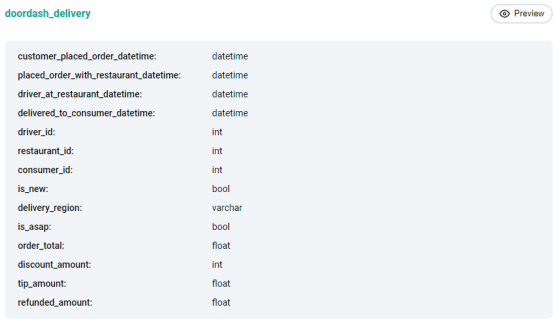
View the output in a separate browser tab

year	employeename	ranking
2011	NATHANIEL FORD	1
2012	Lawrence Chan	1
2013	Kathryn Waaland	1
2014	Mark F Obrochta	1

▼ 11 (ID 2036) **Lowest Revenue Generated Restaurants**

```
-- DoorDash | Hard | Active Interview | ID 2036
/*
Write a query that returns a list of the bottom 2% revenue generating restaurants. Return a list of restaurant IDs and their total reve
You can calculate the total revenue by summing the order_total column. And you should calculate the bottom 2% by partitioning the total
*/

with
cte as
    (select
            restaurant_id
            , sum(order_total) as tot_revenue
        from doordash_delivery
        where extract(year from placed_order_with_restaurant_datetime) = 2020
            and extract(month from placed_order_with_restaurant_datetime) = 05
        group by 1)

select *
from (select
        *
        , cume_dist() over(order by tot_revenue desc)*100.0 as prcntile
    from cte) x
where prcntile <= 2
;
```

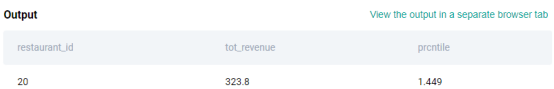
▼ given table & result



doordash_delivery

Preview

customer_placed_order_datetime:	datetime
placed_order_with_restaurant_datetime:	datetime
driver_at_restaurant_datetime:	datetime
delivered_to_consumer_datetime:	datetime
driver_id:	int
restaurant_id:	int
consumer_id:	int
is_new:	bool
delivery_region:	varchar
is_asap:	bool
order_total:	float
discount_amount:	int
tip_amount:	float
refunded_amount:	float



Output

View the output in a separate browser tab

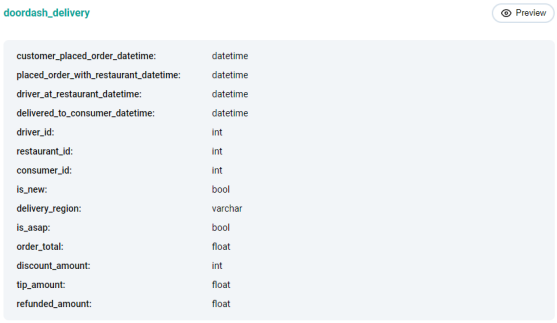
restaurant_id	tot_revenue	prcntile
20	323.8	1.449

▼ 12 (ID 2037) **Delivering and Placing Orders**

```
-- DoorDash | Hard | Active Interview | ID 2037
/*
Check if there is a correlation between average total order value
and average time in minutes between placing the order and delivering the order per restaurant.
*/
```

```
with
cte as
    (select
        *
        , (unix_timestamp(delivered_to_consumer_datetime) - unix_timestamp(customer_placed_order_datetime))/60.0 as diff_in_min
    from doordash_delivery)

select
    restaurant_id
    , avg(order_total) as avg_ord_tot
    , avg(diff_in_min) as avg_diff
from cte
group by 1
order by 2
;
```

▼ given table & result



doordash_delivery — Preview

customer_placed_order_datetime:	datetime
placed_order_with_restaurant_datetime:	datetime
driver_at_restaurant_datetime:	datetime
delivered_to_consumer_datetime:	datetime
driver_id:	int
restaurant_id:	int
consumer_id:	int
is_new:	bool
delivery_region:	varchar
is_asap:	bool
order_total:	float
discount_amount:	int
tip_amount:	float
refunded_amount:	float

Output — View the output in a separate browser tab

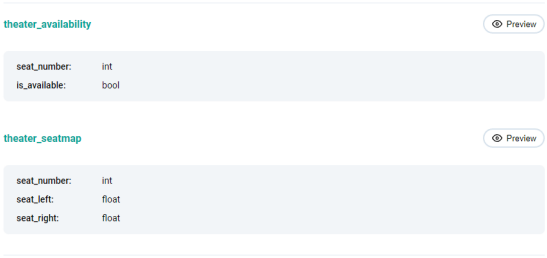
restaurant_id	avg_ord_tot	avg_diff
90	14.16	19
26	14.65	38
298	15.27	71
83	16.33	30
43	17.36	24
44	19	50
6	20.08	34
39	21.71	40
205	23.29	31
349	23.35	47
188	23.835	63.5
47	23.94	57
135	24.49	71
225	24.98	86
84	24.98	22
57	24.98	32

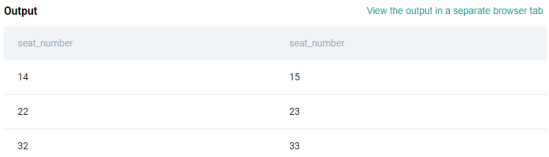
▼ 13 (ID 2088) **Seat Availability**

```
-- Robinhood | Hard | Active Interview | ID 2088
/*
A movie theater gave you two tables: seats that are available for an upcoming screening and neighboring seats for each seat listed. You
Output only distinct pairs of seats in two columns such that the seat with the lower number is always in the first column and the one w
*/

with
cte_available as
    (select
        s.*
        , a1.is_available as seat
        , a2.is_available as is_left
        , a3.is_available as is_right
    from theater_seatmap s
    join theater_availability a1 on s.seat_number = a1.seat_number
    left join theater_availability a2 on s.seat_left = a2.seat_number
    left join theater_availability a3 on s.seat_left = a3.seat_number
    where a1.is_available + a2.is_available + a3.is_available >=2
        and a1.is_available = 1)

, cte_list_seat as
    (select seat_number from cte_available where seat_number is not null
    union all
    select seat_left from cte_available where seat_left is not null
    union all
```

```
    select seat_right from cte_available where seat_right is not null)

select
    x.seat_number
    , y.seat_number
from (select distinct *
    from cte_list_seat
    where seat_number%2 = 0
    order by seat_number) x

join (select distinct *
    from cte_list_seat
    where seat_number%2 != 0
    order by seat_number) y

on x.seat_number+1 = y.seat_number
;
```

▼ given table & result

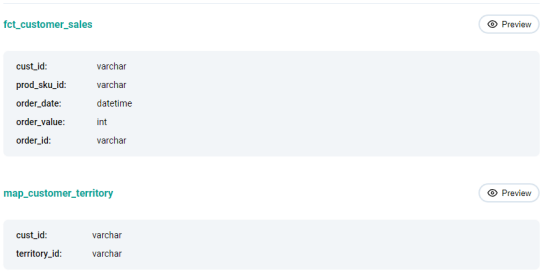




▼ 14 (ID 2111) **Sales Growth per Territory**

```
-- Amazon | Hard | Active Interview | ID 2111
/*
Write a query to return Territory and corresponding Sales Growth. Compare growth between periods Q4-2021 vs Q3-2021.
If Territory (say T123) has Sales worth $100 in Q3-2021 and Sales worth $110 in Q4-2021, then the Sales Growth will be 10% [ i.e. = ((1
Output the ID of the Territory and the Sales Growth. Only output these territories that had any sales in both quarters.
*/

with
cte_main_table as
    (select
        s.order_value
        , t.territory_id
        , if (extract(month from order_date)<10, 3, 4) as quarter
    from fct_customer_sales s
    join map_customer_territory t on s.cust_id = t.cust_id
    where extract(month from order_date) > 6)

, cte_q3 as
    (select
        territory_id
        , sum(order_value) as tot_sales
    from cte_main_table
    where quarter = 3
    group by 1)

, cte_q4 as
    (select
        territory_id
        , sum(order_value) as tot_sales
    from cte_main_table
    where quarter = 4
    group by 1)

select
    q3.territory_id
    , 100.0*(q4.tot_sales - q3.tot_sales)/q3.tot_sales as prcntg_sales_growth
from cte_q4 q4
join cte_q3 q3 on q3.territory_id = q4.territory_id
order by 1
;
```

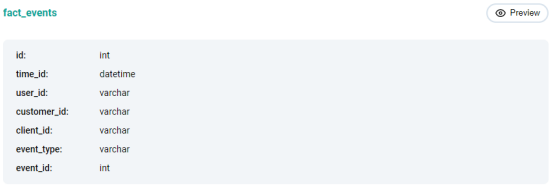
▼ given table & result



▼ 15 (ID 2029) **The Most Popular Client_Id Among Users Using Video and Voice Calls**

```
-- Microsoft | Hard | Active Interview | ID 2029
/*
Select the most popular client_id based on a count of the number of users who have
at least 50% of their events from the following list: 'video call received', 'video call sent', 'voice call received', 'voice call sent
*/

with
cte_user_tot_event as
    (select
        user_id
        , count(1) as num_events
    from fact_events
    group by 1)


, cte_user_events as
    (select
        user_id
        , count(1) as num_spec_event
    from fact_events
    where event_type in ('video call received', 'video call sent', 'voice call received', 'voice call sent')
    group by 1)

select
    t.user_id
    , num_events
    , num_spec_event
    , 100.0 * num_spec_event/num_events as pcnt_events
from cte_user_tot_event t
left join cte_user_events u on t.user_id = u.user_id
where 100.0 * num_spec_event/num_events >= 50
;
```

▼ given table & result

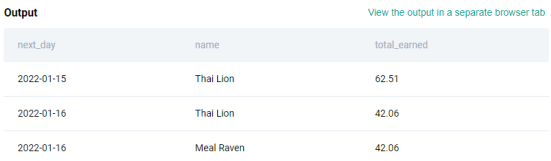


▼ 16 (ID 2094) **Highest Earning Merchants**

```
-- DoorDash | Hard | Active Interview | ID 2094
/*
For each day, find a merchant who earned the highest amount on a previous day. Round total amount to 2 decimals.
Output the date and the name of the merchant but only for the days where the data from the previous day are available.
In the case of multiple merchants having the same highest shared amount, your output should include all the names in different rows.
```

```
*/

with
cte_earned as
     (select
        m.name
        , date(order_timestamp) as date
        , sum(total_amount_earned) as total_earned
    from doordash_orders o
    join doordash_merchants m on o.merchant_id = m.id
    group by 1, 2)

, rank_table as
    (select
        date
        , name
        , total_earned
        , dense_rank() over(partition by date order by total_earned desc) as ranking
    from cte_earned)

, cte_rank_day as
    (select
        *
        , date_add(date, interval 1 day) as next_day
    from rank_table
    where ranking = 1)

select distinct *
from(select
        x1.next_day
        , x1.name
        , x1.total_earned
    from cte_rank_day x1
    join cte_rank_day x2 on x1.next_day = x2.date) y
;
```

▼ given table & result

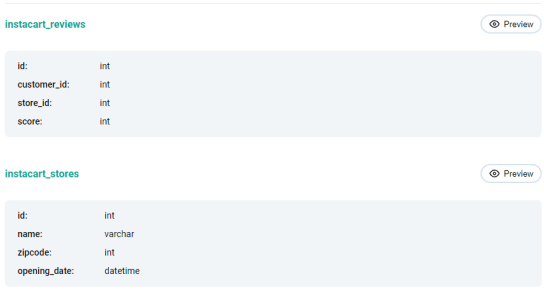


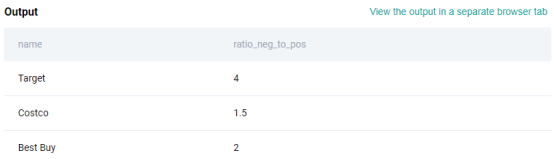
▼ 17 (ID 2087) **Negative Reviews in New Locations**

```
-- Instacart | Hard | Active Interview | ID 2087
/*
Find stores that were opened in the second half of 2021 with more than 20% of their reviews being negative.
A review is considered negative when the score given by a customer is below 5.
Output the names of the stores together with the ratio of negative reviews to positive ones.
*/

with
main_table as
    (select
        name
        , score
    from instacart_reviews r
    join instacart_stores s on r.store_id = s.id
    where opening_date > '2021-06-31')

, num_neg_rev as
    (select
        name
```

```
        , count(score) as num_neg
    from main_table
    where score < 5
    group by name)

, num_all as
    (select name, count(*) as num_all
    from main_table
    group by 1)

, stores as
    (select
        n.name
        , num_neg
    from num_neg_rev n
    left join num_all a on n.name = a.name
    where 100.0*num_neg/num_all > 20)


select
    s.name
    , num_neg/(num_all-num_neg) as ratio_neg_to_pos
from stores s
left join num_all a on s.name = a.name

;
```

▼ given table & result

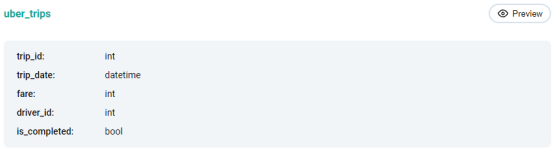


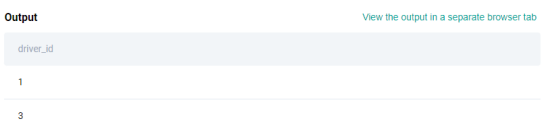


▼ 18 (ID 2076) **Trips in Consecutive Months**

```
-- Uber | Hard | Active Interview | ID 2076
/*
Find the IDs of the drivers who completed at least one trip a month for at least two months in a row.
*/

with
cte_month_num_trip as
    (select
        driver_id
        , extract(month from trip_date) as month
        , count(trip_id) as num_trip
    from uber_trips
    group by 1, 2
    order by 1, 2)

, cte_prev_month as
    (select
        *
        , lag(month) over(partition by driver_id order by month) as prev_month
    from cte_month_num_trip)

select distinct driver_id
from cte_prev_month
where prev_month is not null
    and (month-prev_month) = 1
;
```

▼ given table & result

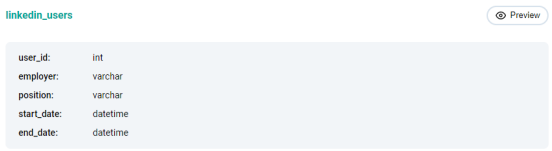


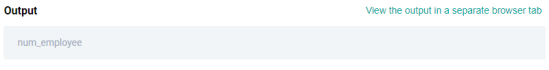


▼ 19 (ID 2078) **From Microsoft to Google**

```
-- LinkedIn | Hard | Active Interview | ID 2078
/*
Consider all LinkedIn users who, at some point, worked at Microsoft.
For how many of them was Google their next employer right after Microsoft (no employers in between)?
*/

with
cte_next_employer as
    (select
        user_id
        , employer
        , start_date
        , lead(employer) over(partition by user_id order by start_date
                        rows between unbounded preceding and unbounded following) as next_employer
    from linkedin_users)

select count(distinct user_id) as num_employee
from cte_next_employer
where employer = 'Microsoft'
    and next_employer = 'Google'
;
```

▼ given table & result





▼ 20 (ID 2073) **Popular Posts**

```
-- Meta/Facebook | Hard | Active Interview | ID 2073
/*
The column 'perc_viewed' in the table 'post_views' denotes the percentage of the session duration time the user spent viewing a post.
Using it, calculate the total time that each post was viewed by users. Output post ID and the total viewing time in seconds, but only f
*/

with
cte as
    (select
        post_id
        , timestampdiff(second, session_starttime, session_endtime)*perc_viewed/100 as view_time
    from user_sessions s
    join post_views v on s.session_id = v.session_id)

select
    post_id
    , sum(view_time) as tot_view_time
from cte
group by 1
having sum(view_time) > 5
;
```

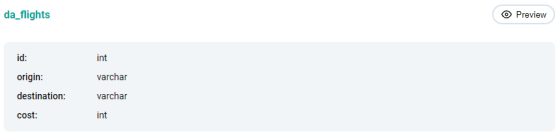
▼ given table & result

▼ 21 (ID 2008) **The Cheapest Airline Connection**

```
-- Delta Airlines | Hard | Active Interview | ID 2008
/*
COMPANY X employees are trying to find the cheapest flights to upcoming conferences. When people fly long distances, a direct city-to-c
Travelers might save even more money by breaking the trip into three flights with two stops. But for the purposes of this challenge, le
  • id - the unique ID of the flight;
  • origin - the origin city of the current flight;
  • destination - the destination city of the current flight;
  • cost - the cost of current flight.
Your task is to produce a trips table that lists all the cheapest possible trips that can be done in two or fewer stops. This table sho
Sort the output table by origin, then by destination. The cities are all represented by an abbreviation composed of three uppercase Eng
*/

with
cte_one_stop as
    (select
        d1.origin
        , d2.destination
        , d1.cost+d2.cost as cost
    from da_flights d1
    join da_flights d2 on d1.destination = d2.origin)

, cte_two_stop as
    (select
        d1.origin
        , d3.destination
        , d1.cost+d2.cost+d3.cost as cost
    from da_flights d1
    join da_flights d2 on d1.destination = d2.origin
    join da_flights d3 on d2.destination = d3.origin)

, cte_all_flights as
    (select
        origin
        , destination
        , concat(origin, '-', destination) as spl
        , cost
    from da_flights
    union

    select
        origin
        , destination
        , concat(origin, '-', destination) as spl
        , cost
    from cte_one_stop
    union

    select
        origin
        , destination
        , concat(origin, '-', destination) as spl
        , cost
    from cte_two_stop)

, cte_ranking as
    (select
        *
        , rank() over(partition by spl order by cost) as ranking
    from cte_all_flights
    order by origin, destination)
```

```
select
    origin
    , destination
    , cost
from cte_ranking
where ranking = 1
;
```

▼ given table & result



Output View the output in a separate browser tab

origin	destination	cost
DFW	JFK	200
DFW	LHR	1200
DFW	MCO	100
JFK	LHR	1000
SFO	DFW	200
SFO	JFK	400
SFO	LHR	1400
SFO	MCO	300

▼ 22 (ID 2012) **Viewers Turned Streamers**

```
-- Twitch | Hard | Active Interview | ID 2012
/*
CFrom users who had their first session as a viewer, how many streamer sessions have they had?
Return the user id and number of sessions in descending order. In case there are users with the same number of sessions, order them by
*/

with
cte_first_as as
    (select
        user_id
        , session_start
        , session_type
        , first_value(session_type) over(partition by user_id order by session_start) as first_as
    from twitch_sessions)

, cte_first_viewer as
    (select distinct user_id
    from cte_first_as
    where first_as = 'viewer')

select
    user_id,
    count(session_id) as num_stream_session
from twitch_sessions
where user_id in (select * from cte_first_viewer)
    and session_type = 'streamer'
group by 1
order by 2 desc, 1 asc
;
```

▼ given table & result

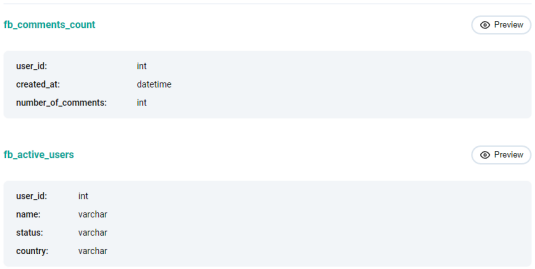
Output View the output in a separate browser tab

user_id	num_stream_session
1	1
3	1

▼ 23 (ID 2007) **Rank Variance Per Country**

```
-- Meta/Facebook | Hard | Active Interview | ID 2007
/*
Which countries have risen in the rankings based on the number of comments between Dec 2019 vs Jan 2020?
Hint: Avoid gaps between ranks when ranking countries.
*/

with
cte_main_table as
    (select
        c.user_id
        , c.created_at
        , c.number_of_comments
        , u.country
    from fb_comments_count c
    join fb_active_users u on c.user_id = u.user_id
    where created_at between '2019-12-01' and '2020-01-31')

, cte_com_dec as
    (select
        country
        , sum(number_of_comments) as num_com_dec
    from cte_main_table
    where created_at between '2019-12-01' and '2019-12-31'
    group by 1)

, cte_com_jan as
    (select
        country
        , sum(number_of_comments) as num_com_jan
    from cte_main_table
    where created_at between '2020-01-01' and '2020-01-31'
    group by 1)

select
    d.country
    , num_com_dec
    , num_com_jan
from cte_com_dec d
join cte_com_jan j on d.country = j.country and num_com_jan>num_com_dec

;
```

▼ given table & result



Output View the output in a separate browser tab

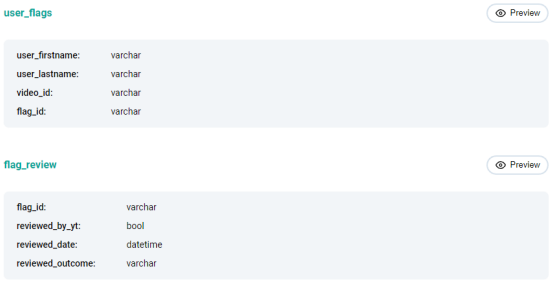
country	num_com_dec	num_com_jan
Australia	5	6
Luxembourg	2	3
Mali	8	11

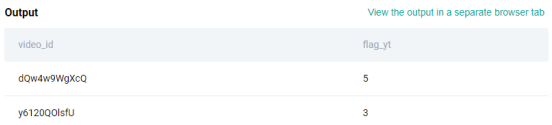
▼ 24 (ID 2103) **Reviewed flags of top videos**

```
-- Google | Hard | Active Interview | ID 2103
/*
For the video (or videos) that received the most user flags,
how many of these flags were reviewed by YouTube? Output the video ID and the corresponding number of reviewed flags.
*/

with
cte_main_table as
    (select
        video_id
        , f.flag_id
        , reviewed_by_yt
    from user_flags f
    left join flag_review r on f.flag_id = r.flag_id)
```

```
, cte_video_id as
    (select video_id
    from (select
            video_id
            , rank() over(order by count(flag_id) desc) as ranking
        from cte_main_table
        group by 1) x
    where ranking = 1)

select
    video_id
    , sum(reviewed_by_yt) as flag_yt
from cte_main_table
where video_id in (select * from cte_video_id)
group by 1
order by 2 desc
;
```

▼ given table & result

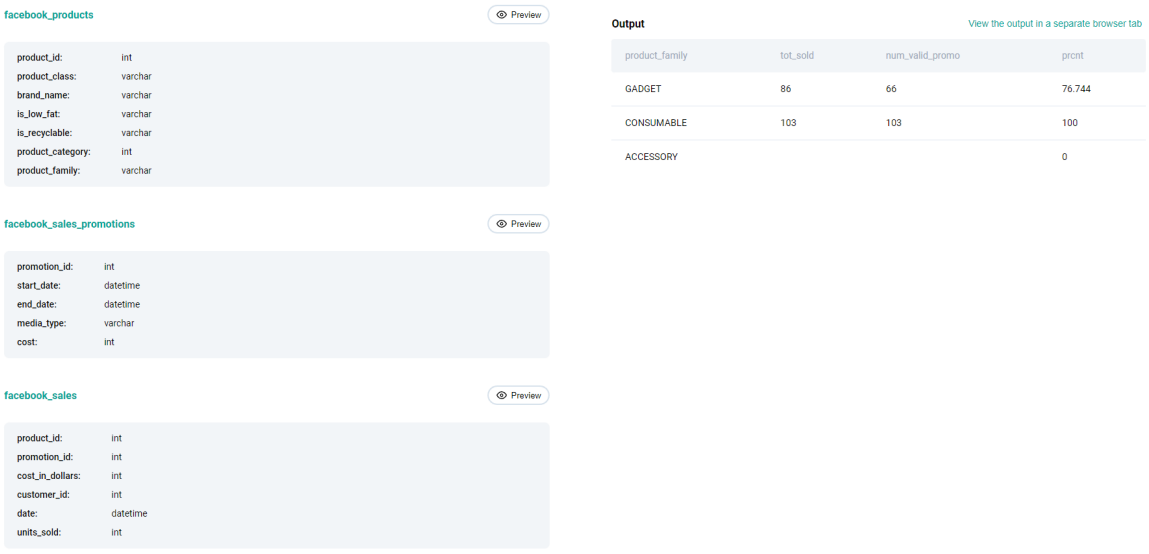




▼ 25 (ID 2123) **Product Families**

```
-- Meta/Facebook | Hard | Active Interview | ID 2123
/*
The CMO is interested in understanding how the sales of different product families are affected by promotional campaigns.
To do so, for each product family, show the total number of units sold, as well as the percentage of units sold that had a valid promot
If there are NULLS in the result, replace them with zeroes. Promotion is valid if it's not empty and it's contained inside promotions t
*/

with
cte_valid_promo as
    (select
        p.product_family
        , sum(units_sold) as num_valid_promo
    from facebook_products p
    left join facebook_sales s on p.product_id = s.product_id
    join facebook_sales_promotions r on r.promotion_id = s.promotion_id
    group by 1)

, cte_tot_sold as
    (select
        p.product_family
        , sum(s.units_sold) as tot_sold
    from facebook_products p
    left join facebook_sales s on p.product_id = s.product_id
    group by 1)

select
    t.product_family
    , tot_sold
    , num_valid_promo
    , if(tot_sold = 0, 0, 100.0*num_valid_promo/tot_sold) as prcnt
from cte_tot_sold t
left join cte_valid_promo v on t.product_family = v.product_family
;
```

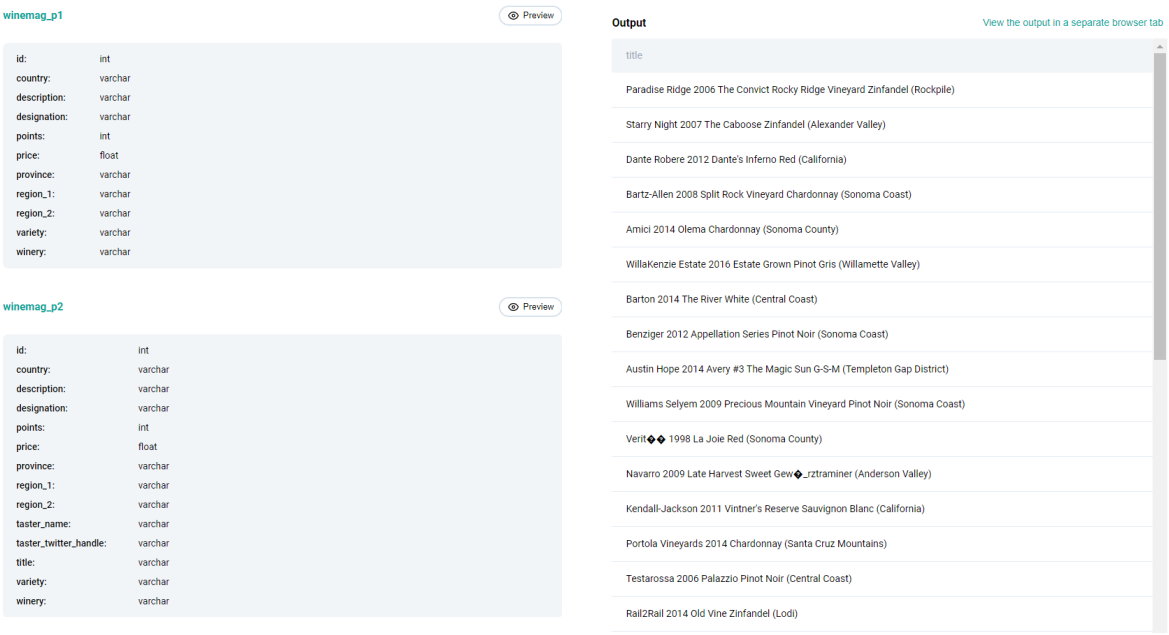
▼ given table & result



▼ 26 (ID 10040) **Find all wines from the winemag_p2 dataset which are produced in countries that have the highest sum of points in the winemag_p1 dataset**

```
-- Wine Magazine | Hard | General Practice | ID 10040
/*
Find all wines from the winemag_p2 dataset which are produced in the country
that have the highest sum of points in the winemag_p1 dataset.
*/

with
cte_sum_points as
    (select
        country
        , sum(points) as sum_points
    from winemag_p1
    group by 1
    order by 2 desc)

, cte_rank_country as
    (select
        *
        , rank() over(order by sum_points desc) as ranking
    from cte_sum_points)


select title
from winemag_p2
where
country = (select country
        from cte_rank_country
        where ranking = 1)

;
```

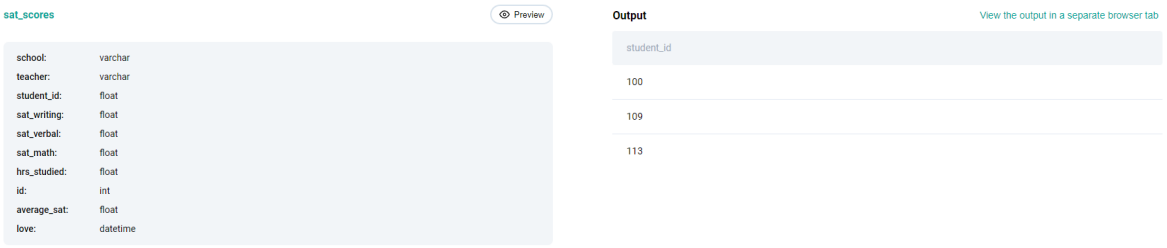
▼ given table & result



▼ 27 (ID 9610) **Find students with a median writing score**

```
-- Google | Hard | General Practice | ID 9610
/*
Output ids of students with a median score from the writing SAT.
*/

with
cte_sat_score as
    (select
        PERCENTILE_CONT(0.5) WITHIN GROUP(ORDER BY sat_writing)
        -- , PERCENTILE_DISC(0.5) WITHIN GROUP(ORDER BY sat_writing)
    from sat_scores )

select student_id
from sat_scores
where sat_writing = (select * from cte_sat_score)
;
```

▼ given table & result

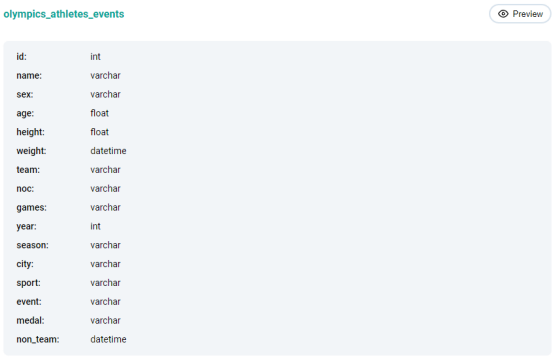


▼ 28 (ID 9960) **Top Teams In The Rio De Janeiro 2016 Olympics**

```
-- ESPN | Hard | General Practice | ID 9960
/*
Find the top 3 medal-winning teams by counting the total number of medals for each event in the Rio De Janeiro 2016 olympics.
In case there is a tie, order the countries by name in ascending order. Output the event name along with the top 3 teams as the 'gold t
with the team name and the total medals under each column in format "{team} with {number of medals} medals". Replace NULLs with "No Tea
*/
```

```
with
cte_ranking as
    (select
        games
        , team
        , count(medal) as count_medal
        , dense_rank() over(order by count(medal) desc) as ranking
    from olympics_athletes_events
    where city like '%Rio%'
    group by 1, 2
    order by 4)

select
    *
    , case when ranking = 1 then 'gold team'||' with '||count_medal
    when ranking = 3 then 'bronze team'||' with '||count_medal
    else 'silver team'||' with '||count_medal end as team_stat
from cte_ranking
where ranking <= 3
;
```

▼ given table & result



games	team	count_medal	ranking	team_stat
2016 Summer	United States	5	1	gold team with 5
2016 Summer	Germany	4	2	silver team with 4
2016 Summer	Great Britain	4	2	silver team with 4
2016 Summer	Australia	4	2	silver team with 4
2016 Summer	Italy	3	3	bronze team with 3
2016 Summer	China	3	3	bronze team with 3

▼ 29 (ID 9979) **Find the top 5 highest paid and top 5 least paid employees in 2012**

```
-- City of San Francisco | Hard | General Practice | ID 9979
/*
Find the top 5 highest paid and top 5 least paid employees in 2012.
Output the employee name along with the corresponding total pay with benefits.
Sort records based on the total payment with benefits in ascending order.
*/

with
cte_rank as
    (select
        employeename
        , totalpaybenefits
        , dense_rank() over(order by totalpaybenefits desc) as rank_desc
        , dense_rank() over(order by totalpaybenefits) as rank_asc
    from sf_public_salaries
    order by 2 desc)

select
    employeename
    , totalpaybenefits
from cte_rank
where rank_desc <= 5
or rank_asc <= 5
order by 2
;
```

▼ given table & result

sf_public_salaries

Preview

id:	int
employeename:	varchar
jobtitle:	varchar
basepay:	float
overtimepay:	float
otherpay:	float
benefits:	float
totalpay:	float
totalpaybenefits:	float
year:	int
notes:	datetime
agency:	varchar
status:	varchar

Output

View the output in a separate browser tab

employeename	totalpaybenefits
NATHANIEL FORD	567595.43
PATRICIA JACKSON	297608.92
ANNA BROWN	238551.88
EDWARD REISKIN	230827.12
DOUGLAS MCEACHERN	196494.14
Maria E Zuniga	114.54
Brighton M Leung	108.43
Vanessa E Almaguer	88.18
Grace Salud	46.27
Renato C Gurion	7.24

▼ 30 (ID 10173) **Days At Number One**

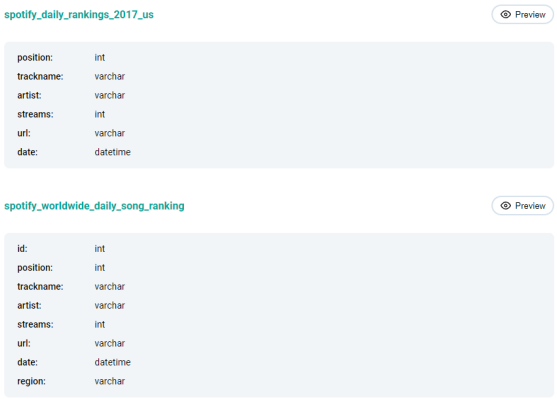
```
-- Spotify | Hard | General Practice | ID 10173
/*
You have a table with US rankings of tracks and another table with worldwide rankings of tracks.
Find the number of days a US track has stayed in the 1st position for both the US and worldwide rankings.
Output the track name and the number of days in the 1st position. Order your output alphabetically by track name.
*/

with
cte_us as
    (select
        trackname
        , artist
        , min(date) as us_date
    from spotify_daily_rankings_2017_us
    where position = 1
    group by 1, 2)

, cte_us_world as
    (select
        w.trackname
        , w.artist
        , w.date
        , w.position
        , u.us_date
    from spotify_worldwide_daily_song_ranking w
    join cte_us u on w.trackname = u.trackname and w.artist = u.artist)


select
    trackname
    , artist
    , abs(max(date) - min(us_date))
from cte_us_world
group by 1, 2
;
```

▼ given table & result

Output

View the output in a separate browser tab

trackname	artist	abs
HUMBLE.	Kendrick Lamar	21
Bad and Boujee (feat. Lil Uzi Vert)	Migos	76
Unforgettable	French Montana	61
Look What You Made Me Do	Taylor Swift	8

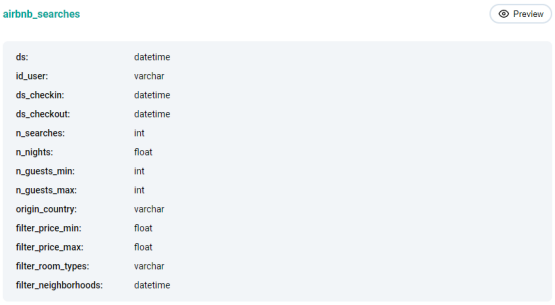


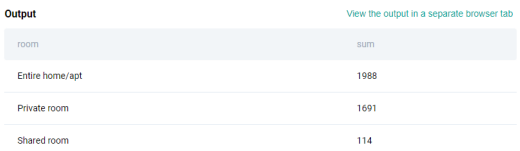
▼ 31 (ID 9763) **Most Popular Room Types**

```
-- Airbnb | Hard | General Practice | ID 9763
/*
Find the room types that are searched by most people.
Output the room type alongside the number of searches for it.
If the filter for room types has more than one room type, consider each unique room type as a separate row.
Sort the result based on the number of searches in descending order.
*/

-- Private room, Entire home/apt, Shared room
-- select distinct filter_room_type from airbnb_searches

with
private as
    (select
        n_searches
        ,case when filter_room_types like '%Private room%' or filter_room_types like 'Private room%' or filter_room_types like '%Privat
        else null end as room
    from airbnb_searches s)

, entire as
    (select
        n_searches
        , case when filter_room_types like '%Entire home/apt%' or filter_room_types like 'Entire home/apt%' or filter_room_types like '
        else null end as room
    from airbnb_searches s)

, shared as
    (select
        n_searches
        , case when filter_room_types like '%Shared room%' or filter_room_types like 'Shared room%' or filter_room_types like '%Shared
        else null end as room
    from airbnb_searches s)

, entire_sum as
    (select
        room
        , sum(n_searches)
    from entire
    where room is not null
    group by 1)

, shared_sum as
    (select
        room
        , sum(n_searches)
    from shared
    where room is not null
    group by 1)

, private_sum as
    (select
        room
        , sum(n_searches)
    from private
    where room is not null
    group by 1)
```

```
select *
from
    (select * from entire_sum
    union
    select * from shared_sum
    union
    select * from private_sum) x
order by "sum" desc

;
```

▼ given table & result

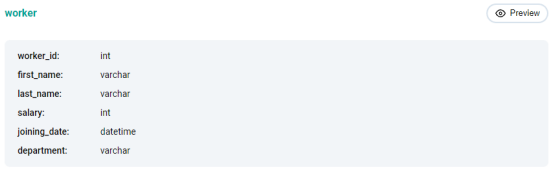


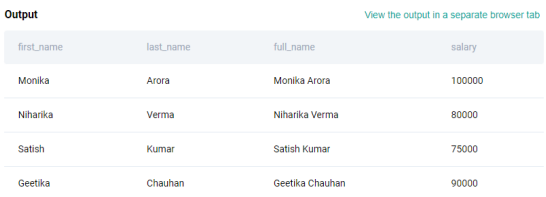


▼ 32 (ID 9846) **Find the full name of workers whose salaries >= 50000 and <= 100000**

```
-- Amazon | Hard | General Practice | ID 9846
/*
Find the full name of workers whose salaries >= 50000 and <= 100000
Output the worker's first name and last name in one column along with their salaries
*/

select
    first_name
    , last_name
    , first_name||' '||last_name as full_name
    , salary
from worker
where salary <= 100000 and salary >= 50000
;
```

▼ given table & result



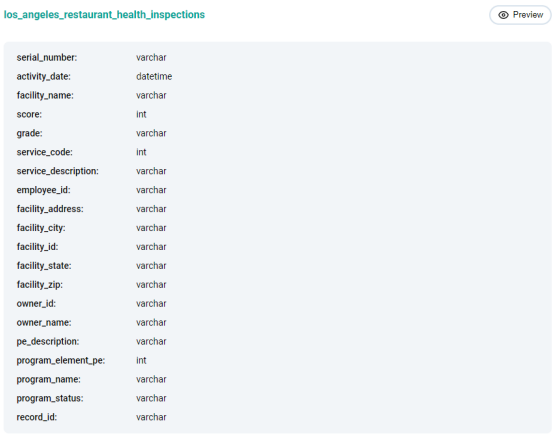


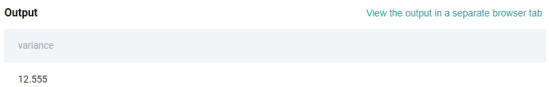
▼ 33 (ID 9708) **Find the variance and the standard deviation of scores that have grade A**

```
-- City of Los Angeles | Hard | General Practice | ID 9708
/*
Find the variance of scores that have grade A using the formula AVG((X_i - mean_x) ^ 2).
Output the result along with the corresponding standard deviation.
*/

with
avg_score as
    (select
```

```
        avg(score)
    from los_angeles_restaurant_health_inspections)

, diff as
    (select
        (score - (select * from avg_score))^2 as difference
    from los_angeles_restaurant_health_inspections
    where grade = 'A')

select avg(difference) as variance from diff
;
```

▼ given table & result

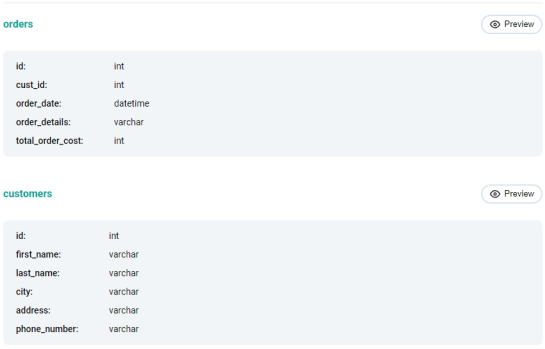


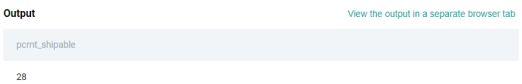


▼ 34 (ID 10090) **Find the percentage of shippable orders**

```
-- Amazon | Hard | General Practice | ID 10090
/*
Find the percentage of shipable orders.
Consider an order is shipable if the customer's address is known.
*/

select
    100.0*count(address) / count(o.id) as pcrnt_shipable
from orders o
join customers c on o.cust_id = c.id
;
```

▼ given table & result

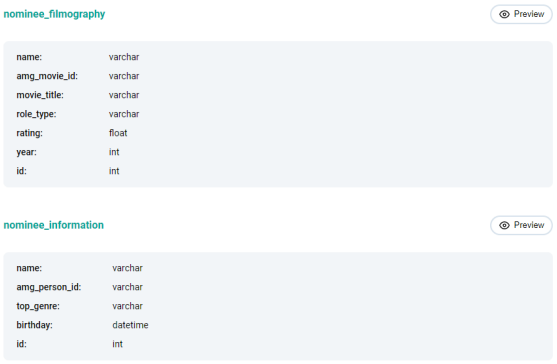




▼ 35 (ID 9605) **Find the average rating of movie stars**

```
-- Netflix | Hard | General Practice | ID 9605
/*
Find the average rating of each movie star along with their names and birthdays.
Sort the result in the ascending order based on the birthday.
Use the names as keys when joining the tables.
*/

select
    f.name
    , i.birthday
    , avg(f.rating) avg_rating
from nominee_filmography f
join nominee_information i on f.name = i.name
group by 1, 2
order by 2
;
```

▼ given table & result



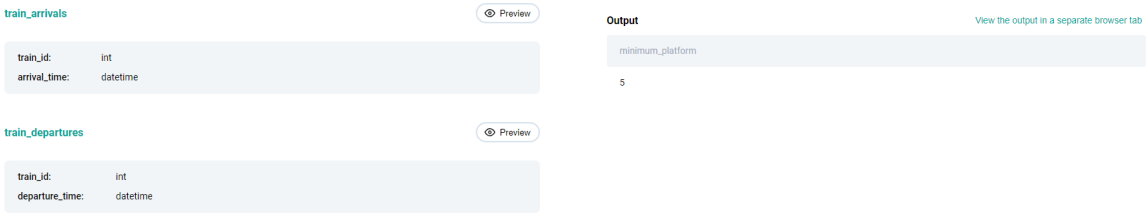
name	birthday	avg_rating
Ruby Dee	1924-10-27	1
Hal Holbrook	1925-02-17	7
Cloris Leachman	1926-04-30	2.5
Rosemary Harris	1930-09-19	6
Martin Landau	1931-06-20	7
Ian Holm	1931-09-12	6
Debbie Reynolds	1932-04-01	4
Michael Caine	1933-03-14	4.5
Alan Alda	1936-01-28	7
Albert Finney	1936-05-09	5
Morgan Freeman	1937-06-01	5.5
Anthony Hopkins	1937-12-31	6
Ian McKellen	1939-05-25	5.5
Tom Conti	1941-11-22	6
Bob Hoskins	1942-10-26	3
Joe Pesci	1943-02-09	
Robert De Niro	1943-08-17	2

▼ 36 (ID 2082) **Minimum Number of Platforms**

```
-- Goldman Sachs | Hard | Active Interview | ID 2082
/*
You are given a day worth of scheduled departure and arrival times of trains at one train station.
One platform can only accommodate one train from the beginning of the minute it's scheduled to arrive until the end of the minute it's
Find the minimum number of platforms necessary to accommodate the entire scheduled traffic.
*/

with
cte_train as
    (select
        a.train_id
        , a.arrival_time
        , d.departure_time
    from train_arrivals a
    join train_departures d on a.train_id = d.train_id)

, cte_count_id as
    (select count(distinct train_id) as count_id
    from cte_train)

, cte_match_table as
    (select
```

```
        t1.train_id as t1_train_id
        , t1.arrival_time as t1_arr_time
        , t1.departure_time as t1_dep_time
        , t2.train_id as t2_train_id
        , t2.arrival_time as t2_arr_time
        , t2.departure_time as t2_dep_time
    from cte_train t1
    join cte_train t2 on t1.departure_time < t2.arrival_time
    order by t1.train_id, t2.train_id)

, cte_diff as
    (select
        t1_train_id
        , (select * from cte_count_id) - count(1) as count_diff
    from cte_match_table
    group by 1)

select min(count_diff) as minimum_platform
from cte_diff
-- from cte_match_table
-- from cte_train
;
```

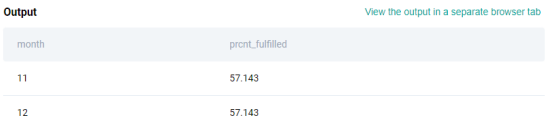
▼ given table & result



▼ 37 (ID 2115) **More Than 100 Dollars**

```
-- DoorDash | Hard | Active Interview | ID 2115
/*
For each month of 2021, calculate what percentage of restaurants,
out of these that fulfilled any orders in a given month,
fulfilled more than 100$ in monthly sales?
*/

with
cte_sales_tot as
    (select
        extract(month from order_placed_time) as month
        , restaurant_id
        , sum(sales_amount)
        , case when sum(sales_amount) > 100 then 1 else 0 end as fulfilled
    from delivery_orders d
    join order_value v on d.delivery_id = v.delivery_id
    where order_placed_time between '2021-01-01' and '2021-12-31'
    group by 1, 2
    order by 1, 2)

, cte_num_resto as
    (select count(distinct restaurant_id)
    from delivery_orders)

select
    month
    , sum(fulfilled) / (select * from cte_num_resto) *100.0 as prcnt_fulfilled
from cte_sales_tot
group by 1

;
```

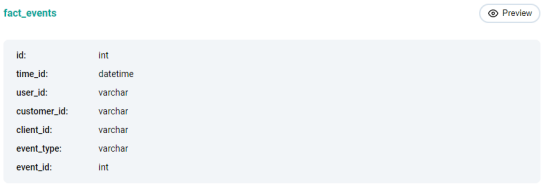
▼ given table & result



▼ 38 (ID 2028) **New And Existing Users**

```
-- Microsoft | Hard | Active Interview | ID 2028
/*
Calculate the share of new and existing users for each month in the table. Output the month, share of new users, and share of existing
New users are defined as users who started using services in the current month (there is no usage history in previous months). Existing
but they also used services in any previous month.
Assume that the dates are all from the year 2020.
*/

with
cte_main as
    (select
        user_id
        , extract(month from time_id) as month
    from fact_events
    order by 1, 2)

, cte_dup_main as
    (select distinct c.month, c.user_id
    from cte_main c
    join cte_main c1 on c.user_id = c1.user_id and c.month = c1.month
    order by 2, 1)

, cte_b4_month as
    (select
        *
        , lag(month) over(partition by user_id order by month) as b4_month
    from cte_dup_main)

, cte_stat as
    (select
        *
        , case when month-b4_month = 1 then 'Existing'
        else 'New' end as stat
    from cte_b4_month)

, cte_new as
    (select
        month
        , count(1) as num_new_user
    from cte_stat
    where stat = 'New'
    group by 1)

, cte_exist as
    (select
        month
        , count(1) as num_exist_user
    from cte_stat
    where stat = 'Existing'
    group by 1)


select
    n.month
    , num_new_user
    , num_exist_user
from cte_new n
left join cte_exist e on n.month = e.month
;
```

▼ given table & result

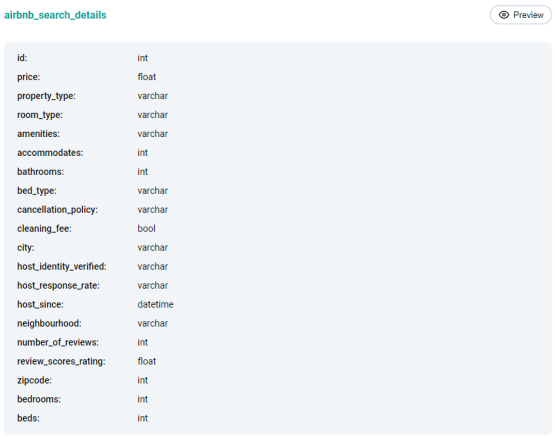


month	num_new_user	num_exist_user
2	13	
3	5	12
4	2	12

▼ 39 (ID 9634) **Host Response Rates With Cleaning Fees**

```
-- Airbnb | Hard | Interview Questions | ID 9634
/*
Find the average host response rate with a cleaning fee for each zipcode. Present the results as a percentage along with the zip code v
Convert the column 'host_response_rate' from TEXT to NUMERIC using type casts and string processing (take missing values as NULL).
Order the result in ascending order based on the average host response rater after cleaning.
*/

with
cte_main as
    (select
        zipcode
        , case when host_response_rate like '%' then replace(host_response_rate, '%', '')::numeric
        else null end response_rate
    from airbnb_search_details
    where cleaning_fee = true)

select
    zipcode
    , avg(response_rate) as avg_rate
from cte_main
group by 1
order by 2
;
```

▼ given table & result



zipcode	avg_rate
91324	0
90028	25
10035	67
90703	70
10039	76
90265	78.5
11106	80
90057	80
10065	80
10027	88.429
90292	89
90501	90

▼ 40 (ID 9734) **Number Of Inspections By Zip**

```
-- City of San Francisco | Hard Interview | Questions | ID 9734
/*
```

```
Find the number of inspections that happened in the municipality
with postal code 94102 during January, May or November in each year.
Output the count of each month separately.
*/

select
    extract(year from inspection_date) as year
    -- , extract(month from inspection_date) as month
    , count(inspection_type) as num_inspection
from sf_restaurant_health_violations
where business_postal_code = 94102 and extract(month from inspection_date) in (1, 5, 11)
group by 1
order by 1
;
```

▼ given table & result

sf_restaurant_health_violations

Preview

business_id:	int
business_name:	varchar
business_address:	varchar
business_city:	varchar
business_state:	varchar
business_postal_code:	float
business_latitude:	float
business_longitude:	float
business_location:	varchar
business_phone_number:	float
inspection_id:	varchar
inspection_date:	datetime
inspection_score:	float
inspection_type:	varchar
violation_id:	varchar
violation_description:	varchar
risk_category:	varchar

Output View the output in a separate browser tab

year	num_inspection
2016	3
2017	2
2018	1

▼ 41 (ID 9955) **Norwegian Alpine Skiers**

```
-- ESPN | Hard | General Practice | ID 9955
/*
Find all Norwegian alpine skiers who participated in 1992 but
didn't participate in 1994. Output unique athlete names.
*/

with
cte_1994 as
    (select distinct name
    from olympics_athletes_events
    where team = 'Norway' and year = 1994)

select distinct name
from olympics_athletes_events
where team = 'Norway' and year = 1992 and name not in (select * from cte_1994)
;
```

▼ given table & result

Output View the output in a separate browser tab

name
Marianne Kjrstad
Atle Skrdal
Vibecke Caroline Gedde-Dahl
Jan Einar Thorsen

olympics_athletes_events

Preview

id:	int
name:	varchar
sex:	varchar
age:	float
height:	float
weight:	datetime
team:	varchar
noc:	varchar
games:	varchar
year:	int
season:	varchar
city:	varchar
sport:	varchar
event:	varchar
medal:	varchar
non_team:	datetime

▼ 42 (ID 10303) **Top Percentile Fraud**

```
-- Netflix | Hard | Interview Questions | ID 10303
/*
ABC Corp is a mid-sized insurer in the US and in the recent past their fraudulent claims have increased significantly for their persona
They have developed a ML based predictive model to identify propensity of fraudulent claims. Now, they assign highly experienced claim
Your objective is to identify the top 5 percentile of claims from each state. Your output should be policy number, state, claim cost, a
*/

with
cte_percentile as
    (select
        *
        , cume_dist() over(partition by state order by claim_cost desc) * 100.0 as percentile
    from fraud_score)

select
    policy_num
    , state
    , claim_cost
    , fraud_score
from cte_percentile
where percentile <= 5
;
```

▼ given table & result

fraud_score

Preview

policy_num:	varchar
state:	varchar
claim_cost:	int
fraud_score:	float

Output

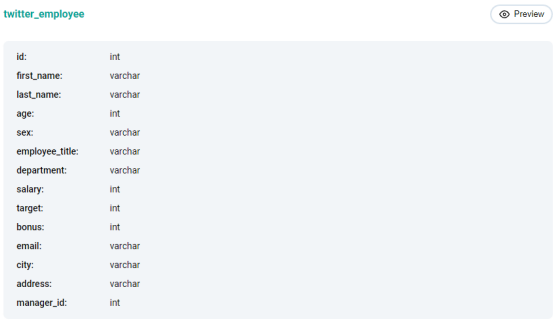
View the output in a separate browser tab

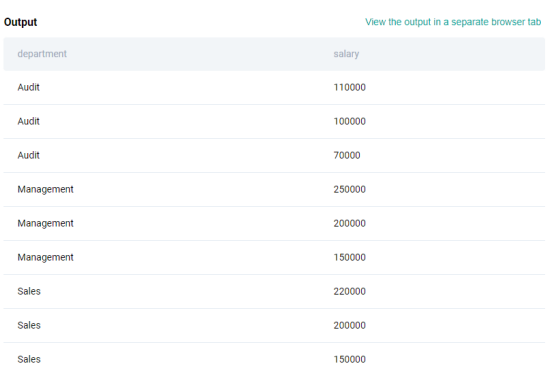
ABCD1059	CA	4949	0.44
ABCD1021	CA	4898	0.947
ABCD1091	CA	4863	0.744
ABCD1280	FL	4875	0.477
ABCD1293	FL	4780	0.287
ABCD1231	FL	4774	0.577
ABCD1268	FL	4742	0.078
ABCD1248	FL	4636	0.506
ABCD1117	NY	4903	0.978
ABCD1186	NY	4879	0.404
ABCD1148	NY	4870	0.541
ABCD1182	NY	4839	0.854
ABCD1146	NY	4819	0.434
ABCD1375	TX	4969	0.708

▼ 43 (ID 9898) **Distinct Salaries**

```
-- Twitter | Hard | Interview Questions | ID 9898
/*
Find the top three distinct salaries for each department.
Output the department name and the top 3 distinct salaries by each department.
Order your results alphabetically by department and then by highest salary to lowest.
*/

with
cte_ranking as
    (select
        *
        , dense_rank() over(partition by department order by salary desc) as ranking
    from twitter_employee)

select
    department
    , salary
from cte_ranking
where ranking in (1, 2, 3)
group by 1, 2
order by 1, 2 desc
;
```

▼ given table & result



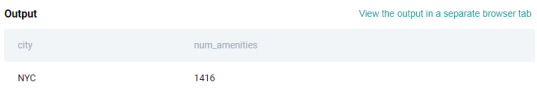


▼ 44 (ID 9633) **City With Most Amenities**

```
-- Airbnb | Hard | Interview Questions | ID 9633
/*
You're given a dataset of searches for properties on Airbnb.
For simplicity, let's say that each search result (i.e., each row) represents a unique host.
Find the city with the most amenities across all their host's properties. Output the name of the city.
*/

with
main_table as
    (select
        city
        , regexp_split_to_table(amenities, ',') as amenities
    from airbnb_search_details)

, clean_main_table as -- clean double quote, curly brackets
    (select
        city
        , replace(replace(replace(amenities, '"', ''), '{', ''), '}', '') as clean
    from main_table)

, cte_amenities_no_dup as -- clean duplication
    (select
```

```
        city
        , clean as amenit
    from clean_main_table
    group by 1, 2
    order by 1, 2)

select
    city
    , count(clean) as num_amenities
from clean_main_table
group by 1
order by 2 desc
limit 1

-- no duplication of amenities in each city in my opinion is more meaningful
-- select
--     city
--     , count(amenit) as num_distinct_amenit
-- from cte_amenities_no_dup
-- group by 1
-- order by 2 desc
-- limit 1
;
```

▼ given table & result

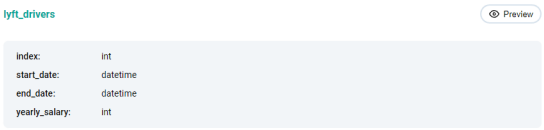


▼ 45 (ID 10017) **Year Over Year Churn**

```
-- Lyft | Hard | Interview Questions | ID 10017
/*
Find how the number of drivers that have churned changed in each year compared to the previous one.
Output the year (specifically, you can use the year the driver left Lyft) along with
the corresponding number of churns in that year, the number of churns in the previous year,
and an indication on whether the number has been increased (output the value 'increase'),
decreased (output the value 'decrease') or stayed the same (output the value 'no change').
*/

with
cte_main_table as
    (select
        extract(year from end_date) as end_year
    from lyft_drivers
    order by 1)

, num_churn_year as
    (select
        end_year
        , num_churn
        , lag(num_churn) over() as num_churn_before
    from (select
            end_year
            , count(end_year) over(partition by end_year) as num_churn
```

```
        from cte_main_table
        where end_year is not null) x
    group by 1, 2
    order by 1)

select *
    , case when num_churn > num_churn_before then 'increase'
    when num_churn < num_churn_before then 'decrease'
    else 'no change' end as status
from num_churn_year
;
```

▼ given table & result



Output View the output in a separate browser tab

end_year	num_churn	num_churn_before	status
2015	5		no change
2016	5	5	no change
2017	8	5	increase
2018	25	8	increase
2019	7	25	decrease

▼ 46 (ID 9740) **Daily Violation Counts**

```
-- City of San Francisco | Hard | Interview Questions | ID 9740
/*
Determine the change in the number of daily violations by calculating
the difference between the count of current and previous violations by inspection date.
Output the inspection date and the change in the number of daily violations.
Order your results by the earliest inspection date first.
*/

with
cte_main as
    (select
        inspection_date
        , count(violation_id) as num_violation
        , lag(count(violation_id)) over(order by inspection_date) as prev_num_violation
    from sf_restaurant_health_violations
    group by 1
    order by 1)

select
    *
    , num_violation - prev_num_violation as change_violation
from cte_main
;
```

▼ given table & result

sf_restaurant_health_violations

Preview

business_id:	int
business_name:	varchar
business_address:	varchar
business_city:	varchar
business_state:	varchar
business_postal_code:	float
business_latitude:	float
business_longitude:	float
business_location:	varchar
business_phone_number:	float
inspection_id:	varchar
inspection_date:	datetime
inspection_score:	float
inspection_type:	varchar
violation_id:	varchar
violation_description:	varchar
risk_category:	varchar

Output

View the output in a separate browser tab

inspection_date	num_violation	prev_num_violation	change_violation
2015-09-08T00:00:00	1		
2015-09-15T00:00:00	0	1	-1
2015-09-18T00:00:00	0	0	0
2015-09-23T00:00:00	0	0	0
2015-09-28T00:00:00	1	0	1
2015-09-30T00:00:00	5	1	4
2015-10-01T00:00:00	1	5	-4
2015-11-17T00:00:00	1	1	0
2015-11-19T00:00:00	1	1	0
2015-12-02T00:00:00	1	1	0
2015-12-07T00:00:00	1	1	0
2015-12-24T00:00:00	0	1	-1
2016-01-06T00:00:00	0	0	0

▼ 47 (ID 10297) **Comments Distribution**

```
-- Meta/Facebook | Hard | Interview Questions | ID 10297
/*
Write a query to calculate the distribution of comments by the count of users
that joined Meta/Facebook between 2018 and 2020, for the month of January 2020.
The output should contain a count of comments and the corresponding number of users
that made that number of comments in Jan-2020.

For example, you'll be counting how many users made 1 comment, 2 comments, 3 comments, 4 comments, etc in Jan-2020.
Your left column in the output will be the number of comments while your right column in the output will be the number of users.
Sort the output from the least number of comments to highest.

To add some complexity, there might be a bug where an user post is dated before the user join date.
You'll want to remove these posts from the result.
*/

with
cte_main as
    (select
        u.id
        , count(body) as num_comments
    from fb_users u
    join fb_comments c on u.id = c.user_id
    where (joined_at between '2018-01-01' and '2020-12-31')
        and (joined_at < created_at)
        and (created_at between '2020-01-01' and '2020-01-31')
    group by 1
    order by 1)

select
    num_comments
    , count(id) as num_user
from cte_main
group by 1
order by 1
;
```

▼ given table & result

fb_users

Preview

id:	int
name:	varchar
joined_at:	datetime
city_id:	int
device:	int

fb_comments

Preview

user_id:	int
body:	varchar
created_at:	datetime

Output

View the output in a separate browser tab

num_comments	num_user
1	4
2	6
3	1
4	1
6	1

▼ 48 (ID 10319) **Monthly Percentage Difference**

```
-- Amazon | Hard | Interview Questions | ID 10319
/*
Given a table of purchases by date, calculate the month-over-month percentage change in
revenue. The output should include the year-month date (YYYY-MM) and percentage change,
rounded to the 2nd decimal point, and sorted from the beginning of the year to the end of the year.

The percentage change column will be populated from the 2nd month forward and
can be calculated as ((this month's revenue - last month's revenue) / last month's revenue)*100.
*/

with
cte_main as
    (select
        extract(month from created_at) as month
        , extract(year from created_at) as year
        , sum(value) as revenue
        , lag(sum(value)) over(order by extract(month from created_at), extract(year from created_at)) as prev_revenue
    from sf_transactions
    group by 1, 2)

select
    month||'-'||"year" as mmyy
    , revenue
    , prev_revenue
    , round(100.0*(revenue-prev_revenue)/prev_revenue, 2) as prcnt_change
from cte_main
;
```

▼ given table & result

sf_transactions

Preview

id:	int
created_at:	datetime
value:	int
purchase_id:	int

Output

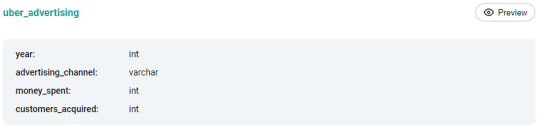
View the output in a separate browser tab

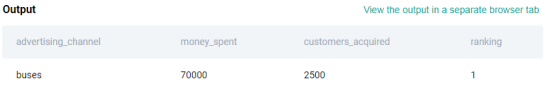
mmyy	revenue	prev_revenue	prcnt_change
1-2019	1332636		
2-2019	952031	1332636	-28.56
3-2019	1174373	952031	23.35
4-2019	1011869	1174373	-13.84
5-2019	1148390	1011869	13.49
6-2019	1116470	1148390	-2.78
7-2019	1049530	1116470	-6
8-2019	1347176	1049530	28.36
9-2019	1280233	1347176	-4.97
10-2019	1117846	1280233	-12.68
11-2019	1137016	1117846	1.71
12-2019	1113028	1137016	-2.11

▼ 49 (ID 10013) **Positive Ad Channels**

```
-- Uber | Hard | General Practice | ID 10013
/*
Find the advertising channel with the smallest maximum yearly spending that
still brings in more than 1500 customers each year.
*/

select *
from (select
        advertising_channel
        , money_spent
        , customers_acquired
        , dense_rank() over(order by money_spent) as ranking
    from uber_advertising
    where customers_acquired > 1500) x
where ranking = 1
;
```

▼ given table & result

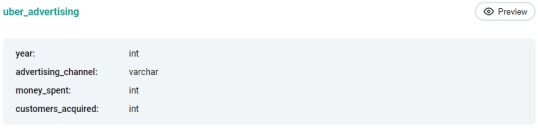


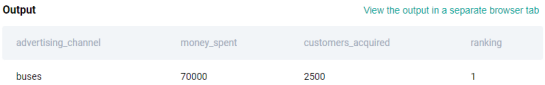


▼ 49 (ID 10013) **Positive Ad Channels**

```
-- Uber | Hard | General Practice | ID 10013
/*
Find the advertising channel with the smallest maximum yearly spending that
still brings in more than 1500 customers each year.
*/

select *
from (select
        advertising_channel
        , money_spent
        , customers_acquired
        , dense_rank() over(order by money_spent) as ranking
    from uber_advertising
    where customers_acquired > 1500) x
where ranking = 1
;
```

▼ given table & result



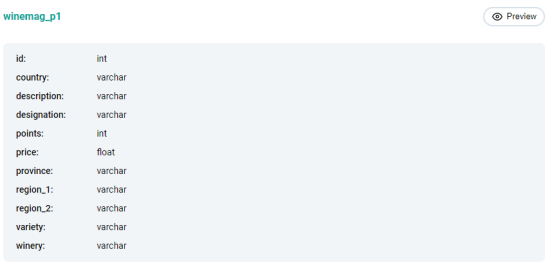


▼ 50 (ID 10041) **Most Expensive And Cheapest Wine**

```
-- Wine Magazine | Hard | Interview Questions | ID 10041
/*
Find the cheapest and the most expensive variety in each region.
Output the region along with the corresponding most expensive and the cheapest variety.
Be aware that there are 2 region columns, price from that row applies to both of them.
```

```
*/

with
cte_main as
    (select
        region_1 as reg
        , variety
        , price
    from winemag_p1
    where region_1 is not null and price is not null
    union
    select
        region_2 as reg
        , variety
        , price
    from winemag_p1
    where region_2 is not null and price is not null)

, cte_ranking as
    (select
        reg
        , variety
        , price
        , rank() over(partition by reg order by price) as ranking
    from cte_main
    order by 1)

, cte_cheap_expe as
    (select
        reg
        , first_value(variety) over(partition by reg order by price) as cheapest
        , last_value(variety) over(partition by reg order by price
        rows between unbounded preceding and unbounded following) as most_expensive
    from cte_ranking)

select
    reg
    , cheapest
    , most_expensive
from cte_cheap_expe
group by 1, 2, 3
order by 1
;
```

▼ given table & result



Output View the output in a separate browser tab

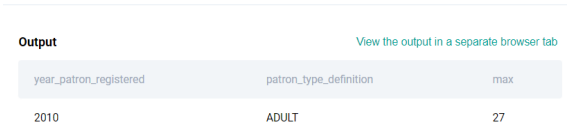
reg	cheapest	most_expensive
Alexander Valley	Merlot	Cabernet Sauvignon
Anderson Valley	Pinot Noir	Pinot Noir
Barbaresco	Nebbiolo	Nebbiolo
Brunello di Montalcino	Sangiovese	Sangiovese
California	Sauvignon Blanc	Pinot Noir
California Other	Sauvignon Blanc	Pinot Noir
Carił±ena	Red Blend	Red Blend
Central Coast	Pinot Noir	Pinot Noir
Chablis	Chardonnay	Chardonnay
Chalone	Pinot Noir	Pinot Noir
Champagne	Champagne Blend	Champagne Blend
Chianti Classico	Sangiovese	Sangiovese
Columbia Valley	Gewİ_rztraminer	Syrah
Columbia Valley (WA)	Gewİ_rztraminer	Merlot
Conegliano Valdobbiadene Prosecco Superiore	Glera	Glera
Diamond Mountain District	Cabernet Sauvignon	Cabernet Sauvignon

Medium SQL Questions

▼ 1 (ID 9926) **Find library types with the highest total checkouts made by adults registered in 2010**

```
-- City of San Francisco | Medium | General Practice | ID 9926
/*
Find library types with the highest total checkouts made by adults registered in 2010.
Output the year patron registered, home library definition along with the corresponding highest total checkouts.
*/

select
    year_patron_registered,
    patron_type_definition,
    max(total_checkouts)
from library_usage
where patron_type_definition = 'ADULT'
    and year_patron_registered = 2010
group by 1,2
;
```

▼ given table & result



▼ 2 (ID 2025) **Users Exclusive Per Client**

```
-- Microsoft | Medium | Active Interview | ID 2025
/*
Write a query that returns a number of users who are exclusive to only one client.
Output the client_id and number of exclusive users.
*/

with cte1 as(
select user_id
from fact_events
where client_id = 'mobile'),

    cte2 as(
select user_id
from fact_events
where client_id != 'mobile')

select *
from(select count(distinct user_id) as desktop
    from cte2
    where user_id not in
```

```
        (select distinct user_id
        from cte1)) x

cross join (select count(distinct user_id) as mobile
    from cte1
    where user_id not in
        (select distinct user_id
        from cte2)) y
;
```

▼ given table & result

fact_events

Preview

id:	int
time_id:	datetime
user_id:	varchar
customer_id:	varchar
client_id:	varchar
event_type:	varchar
event_id:	int

Output

View the output in a separate browser tab

desktop	mobile
2	0

▼ 3 (ID 10073) **Favorite Host Nationality**

```
-- Airbnb | Medium | Interview Questions | ID 10073
/*
For each guest reviewer, find the nationality of the reviewer's favorite host based on the guest's highest review score given to a host
Output the user ID of the guest along with their favorite host's nationality. In case there is more than one favorite host from the sam
list that country only once (remove duplicates).
*/

select
    distinct x.from_user as user_id,
    a.nationality as nationality
from (select
        from_user,
        to_user,
        max(review_score)
    from airbnb_reviews
    where from_type = 'guest' and to_type = 'host'
    group by 1
    order by 1) x
join airbnb_hosts a on x.to_user = a.host_id
order by 1
;
```

▼ given table & result

airbnb_reviews

Preview

from_user:	int
to_user:	int
from_type:	varchar
to_type:	varchar
review_score:	int

airbnb_hosts

Preview

host_id:	int
nationality:	varchar
gender:	varchar
age:	int

Output

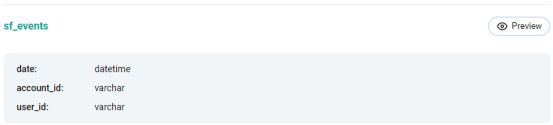
View the output in a separate browser tab

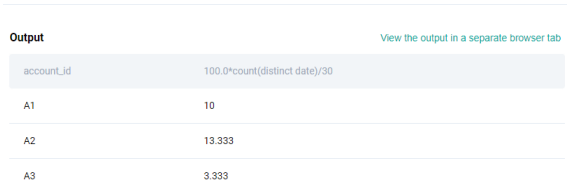
user_id	nationality
0	Brazil
1	Brazil
2	USA
3	Mali
4	China
5	USA
6	China
7	China
8	Australia
9	China
10	USA
11	Brazil

▼ 4 (ID 2050) **Daily Active Users**

```
-- Salesforce | Medium | Active Interview | ID 2050
/*
Find the average daily active users for January 2021 for each account.
Your output should have account_id and the average daily count for that account.
*/

select
    account_id,
    100.0*count(distinct date)/30
from sf_events
where date between '2021-01-01' and '2021-01-31'
group by 1
order by date;
```

▼ given table & result

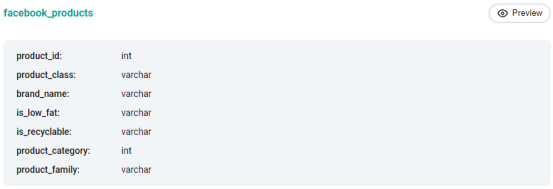


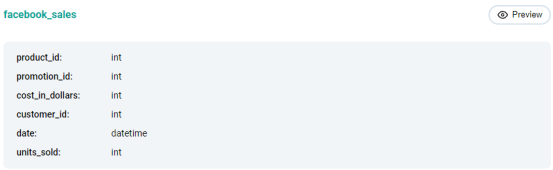


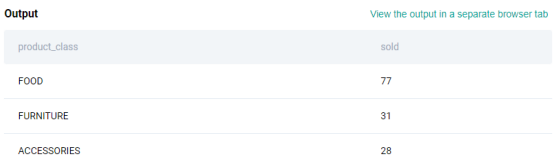
▼ 5 (ID 2070) **Top Three Classes**

```
-- Meta/Facebook | Medium | Active Interview | ID 2070
/*
The marketing department wants to launch a new promotion for the most successful product classes.
What are the top 3 product classes by number of sales?
*/

select
    x.product_class,
    sum(y.units_sold) as sold
from facebook_products x
inner join facebook_sales y on x.product_id = y.product_id
group by 1
order by 2 desc
limit 3
;
```

▼ given table & result







▼ 6 (ID 9706) **Find the month which had the lowest number of inspections across all years**

```
-- City of Los Angeles | Medium | General Practice | ID 9706
/*
Find the month which had the lowest number of inspections across all years.
Output the number of inspections along with the month.
*/

select
    extract(month from activity_date),
    count(service_description)
from los_angeles_restaurant_health_inspections
group by 1
order by 2
;
```

▼ given table & result

los_angeles_restaurant_health_inspections

Preview

serial_number:	varchar
activity_date:	datetime
facility_name:	varchar
score:	int
grade:	varchar
service_code:	int
service_description:	varchar
employee_id:	varchar
facility_address:	varchar
facility_city:	varchar
facility_id:	varchar
facility_state:	varchar
facility_zip:	varchar
owner_id:	varchar
owner_name:	varchar
pe_description:	varchar
program_element_pe:	int
program_name:	varchar
program_status:	varchar
record_id:	varchar

Output

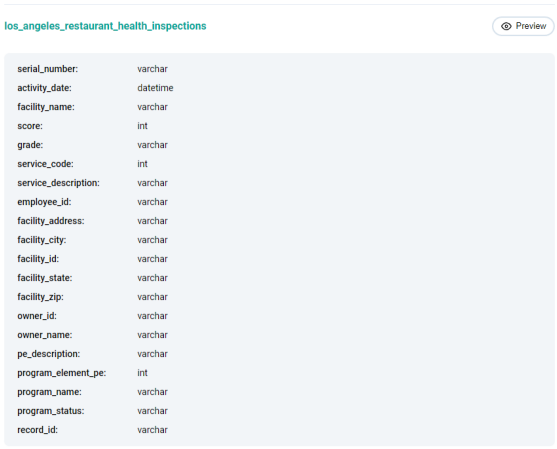
View the output in a separate browser tab

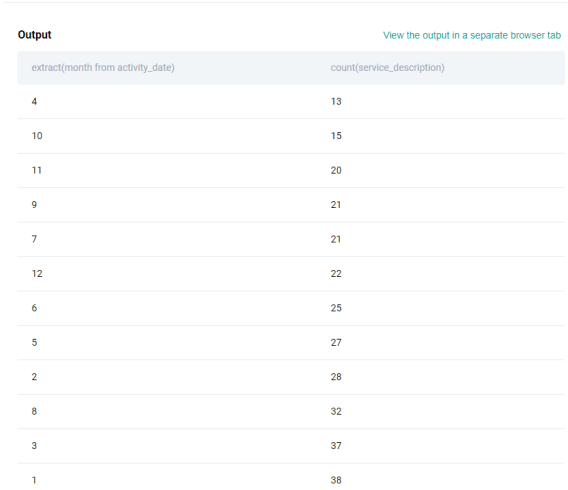
extract(month from activity_date)	count(service_description)
4	13
10	15
11	20
9	21
7	21
12	22
6	25
5	27
2	28
8	32
3	37
1	38

▼ 7 **(ID 10039) Macedonian Vintages**

```
-- Wine Magazine - Medium - General Practice - ID 10039
/*
Find the vintage years of all wines from the country of Macedonia.
The year can be found in the 'title' column. Output the wine (i.e., the 'title') along with the year.
The year should be a numeric or int data type.
*/

select
    id,
    title,
    regexp_matches(title, '[0-9]{4}', 'g')
from winemag_p2
where country = 'Macedonia';
```

▼ given table & result





extract(month from activity_date)	count(service_description)
4	13
10	15
11	20
9	21
7	21
12	22
6	25
5	27
2	28
8	32
3	37
1	38

▼ 8 **(ID 9658) Underweight/Overweight Athletes**

```
-- ESPN | Medium | General Practice | ID 9658
/*
Identify colleges with underweight and overweight athletes.
Consider athletes with weight < 180 pounds as underweight and players with weight > 250 pounds as overweight.
Output the college along with the total number of overweight and underweight players.
If the college does not have any underweight/overweight players, leave the college out of the output.
You can assume that each athlete's full name is unique on their college.
*/

with
stat as
    (select
        name,
        college,
        weight,
        case when weight > 250 then 'overweight'
            when weight < 180 then 'underweight'
            else 'normal' end as status
    from nfl_combine),

uw as
    (select
        college,
        count(status) as uw
    from stat
    where status = 'underweight'
    group by 1
    order by 2),

ow as
    (select
        college,
        count(status) as ow
    from stat
    where status = 'overweight'
    group by 1)


select *
from ow x
join uw y on x.college = y.college;
```

▼ given table & result

los_angeles_restaurant_health_inspections

Preview

serial_number:	varchar
activity_date:	datetime
facility_name:	varchar
score:	int
grade:	varchar
service_code:	int
service_description:	varchar
employee_id:	varchar
facility_address:	varchar
facility_city:	varchar
facility_id:	varchar
facility_state:	varchar
facility_zip:	varchar
owner_id:	varchar
owner_name:	varchar
pe_description:	varchar
program_element_pe:	int
program_name:	varchar
program_status:	varchar
record_id:	varchar

Output

View the output in a separate browser tab

extract(month from activity_date)	count(service_description)
4	13
10	15
11	20
9	21
7	21
12	22
6	25
5	27
2	28
8	32
3	37
1	38

▼ 9 **(ID 10030) Total Wine Revenue**

```
-- Wine Magazine | Medium | Interview Questions | ID 10030
/*
You have a dataset of wines. Find the total revenue made by each winery and variety that has at least 90 points.
Each wine in the winery, variety pair should be at least 90 points in order for that pair to be considered in the calculation.
Output the winery and variety along with the corresponding total revenue. Order records by the winery in ascending order and total reve
*/

select
    variety,
    winery,
    sum(price) as reve
from winemag_p1
where points >= 90
group by 2,1
order by 2, reve desc
;
```

▼ given table & result

winemag_p1

Preview

id:	int
country:	varchar
description:	varchar
designation:	varchar
points:	int
price:	float
province:	varchar
region_1:	varchar
region_2:	varchar
variety:	varchar
winery:	varchar

Output

View the output in a separate browser tab

variety	winery	reve
Cabernet Sauvignon	Yao Ming	625
Cabernet Sauvignon	Lokoya	350
Pinot Noir	Domaine Faiveley	294
Cabernet Sauvignon	Joseph Phelps	200
Bordeaux-style Red Blend	Joseph Phelps	200
Nebbiolo	Roagna	190
Malbec	Bodega Noemí_a de Patagonia	152
Red Blend	Niebaum-Coppola	100
Cabernet Sauvignon	Boudreaux Cellars	100
Cabernet Sauvignon	Sullivan	85
Cabernet Sauvignon	Phelan Vineyard	75
Cabernet Sauvignon	Terra Valentine	70
Bordeaux-style Red Blend	Les Belles Collines	67
Sangiovese	Il Poggione	63
Chardonnay	Roche de Bellene	62

▼ 10 **(ID 10016) Churn Rate Of Lyft Drivers**

```
-- Lyft | Medium | General Practice | ID 10016
/*
Find the global churn rate of Lyft drivers across all years. Output the rate as a ratio.
*/

with
ended as
    (select
        extract(year from end_date) as end_year,
        count(extract(year from end_date)) as count_end
    from lyft_drivers
    group by 1
    order by 1),

started as
    (select
        extract(year from start_date) as start_year,
        count(extract(year from start_date)) as count_start
    from lyft_drivers
    group by 1
    order by 1)

select
    s.start_year as year,
    100.0 * count_end/count_start as percent_churn
from ended e
inner join started s on e.end_year = s.start_year

;
```

▼ given table & result

lyft_drivers

Preview

index:	int
start_date:	datetime
end_date:	datetime
yearly_salary:	int

Output

View the output in a separate browser tab

year	percent_churn
2015	16.129
2016	27.778
2017	34.783
2018	86.207

▼ 11 **(ID 9905) Highest Target Under Manager**

```
-- Salesforce | Medium | Interview Questions| ID 9905
/*
Find the highest target achieved by the employee or employees who works under the manager id 13.
Output the first name of the employee and target achieved.
The solution should show the highest target achieved under manager_id=13 and which employee(s) achieved it.
*/

with
target as
    (select
        first_name,
        last_name,
        max(target) as targ
    from salesforce_employees
    where manager_id = 13
    group by 1,2),

ranking_table as
    (select
        first_name,
        last_name,
        targ,
        dense_rank() over(order by targ desc) as ranking
    from target)

select *
from ranking_table
where ranking = 1
;
```

▼ given table & result

salesforce_employees

Preview

id:	int
first_name:	varchar
last_name:	varchar
age:	int
sex:	varchar
employee_title:	varchar
department:	varchar
salary:	int
target:	int
bonus:	int
email:	varchar
city:	varchar
address:	varchar
manager_id:	int

Output

View the output in a separate browser tab

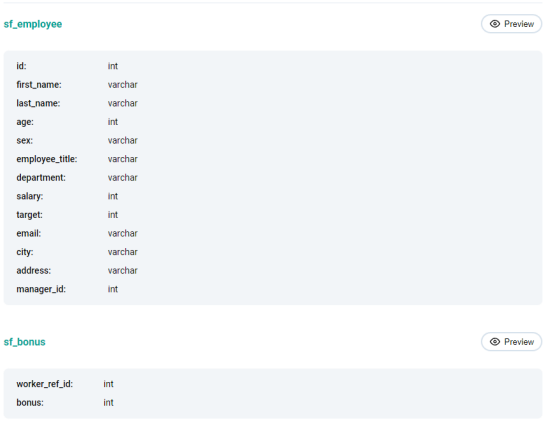
first_name	last_name	targ	ranking
Nicky	Bat	400	1
Steve	Smith	400	1
David	Warner	400	1

▼ 12 **(ID 10077) Income By Title and Gender**

```
-- City of San Francisco| Medium | General Practice | ID 10077
/*
Find the average total compensation based on employee titles and gender.
Total compensation is calculated by adding both the salary and bonus of each employee.
However, not every employee receives a bonus so disregard employees without bonuses in your calculation.
```

```
Employee can receive more than one bonus.
Output the employee title, gender (i.e., sex), along with the average total compensation.
*/

select
    sex,
    employee_title,
    sum(salary+bonus) as compnesation
from sf_employee e
join sf_bonus b on e.id = b.worker_ref_id
group by 1, 2
;
```

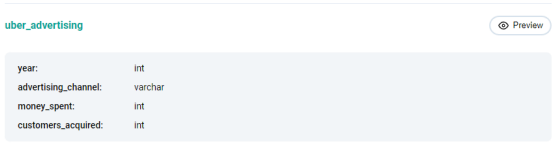
▼ given table & result



▼ 13 **(ID 10009) Find the total costs and total customers acquired in each year**

```
-- Uber | Medium | General Practice | ID 10009
/*
Find the total costs and total customers acquired in each year.
Output the year along with corresponding total money spent and total acquired customers.
*/

select
    year,
    sum(money_spent) as total_spent,
    sum(customers_acquired) as total_customers_acquired
from uber_advertising
group by 1
order by 1
;
```

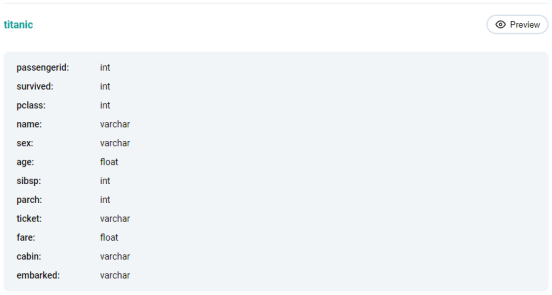
▼ given table & result

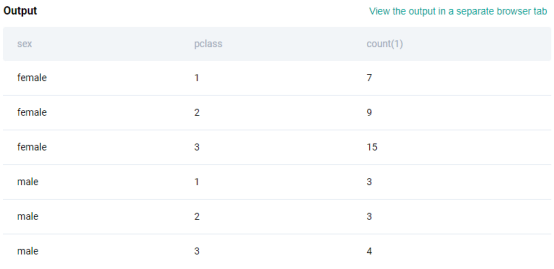


▼ 14 **(**ID 9882**) Find how the survivors are distributed by the gender and passenger classes**

```
-- Google | Medium | General Practice | ID 9882
/*
Find how the survivors are distributed by the gender and passenger classes.
Classes are categorized based on the pclass value as:
pclass = 1: first_class
pclass = 2: second_classs
pclass = 3: third_class
Output the sex along with the corresponding number of survivors for each class.
*/

select
    sex,
    pclass,
    count(1)
from titanic
where survived = 1
group by 1, 2
order by 1, 2
;
```

▼ given table & result



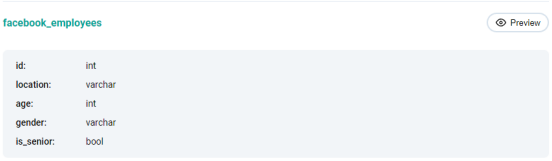


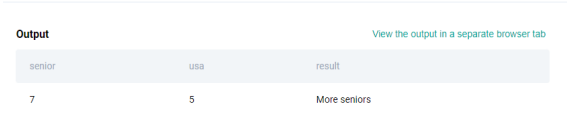
sex	pclass	count(1)
female	1	7
female	2	9
female	3	15
male	1	3
male	2	3
male	3	4

▼ 15 (ID 10065) Find whether the number of seniors works at Meta/Facebook is higher than its number of USA based employees

```
-- Meta/Facebook | Medium | Interview Questions | ID 10065
/*
Find whether the number of senior workers (i.e., more experienced) at Meta/Facebook is higher than number of USA based employees at Fac
If the number of seniors is higher then output as 'More seniors'. Otherwise, output as 'More USA-based'.
*/

with
s as
    (select count(1) as senior
    from facebook_employees
    where is_senior = True),
u as
    (select count(1) as usa
    from facebook_employees
    where location = 'USA')

select *,
    case when senior>usa then 'More seniors'
    else 'More USA-based' end as result
from s
cross join u
;
```

▼ given table & result

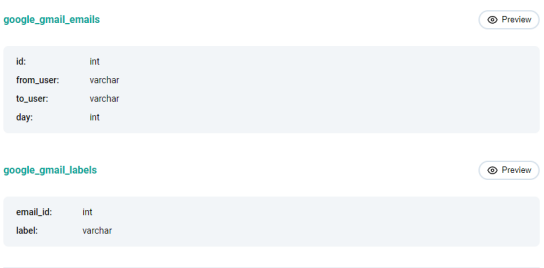


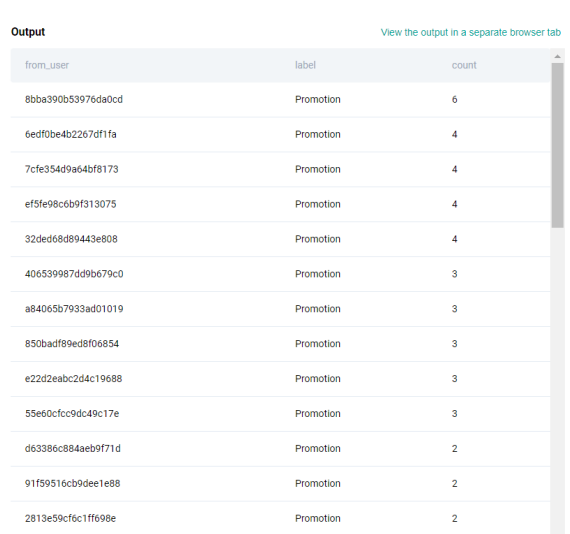


▼ 16 (ID 10068**) User Email Labels**

```
-- Google | Medium | Interview Questions | ID 10068
/*
Find the number of emails received by each user under each built-in email label.
The email labels are: 'Promotion', 'Social', and 'Shopping'.
Output the user along with the number of promotion, social, and shopping mails count,.
*/

select
    e.from_user,
    l.label,
    count(1) as count
from google_gmail_emails e
join google_gmail_labels l on e.id = l.email_id
where l.label in ('Promotion', 'Social', 'Shopping')
group by 1, 2
order by 2, 3 desc;
```

▼ given table & result



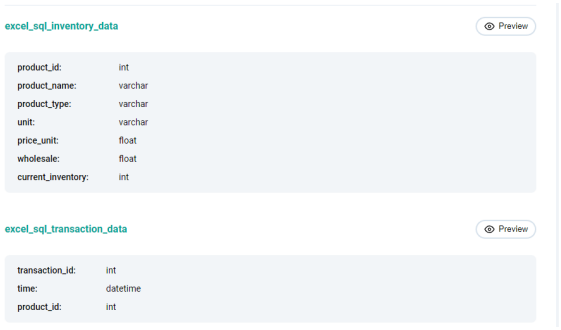


▼ 17 (ID 10163**) Product Transaction Count**

```
-- Microsoft | Medium | General Practice | ID 10163
/*
Find the number of transactions that occurred for each product.
Output the product name along with the corresponding number of transactions and order records by the product id in ascending order.
You can ignore products without transactions.
*/

select
    t.product_id,
    i.product_name,
    count(t.transaction_id)
```

```
from excel_sql_inventory_data i
join excel_sql_transaction_data t on i.product_id = t.product_id
group by 2
order by 1
;
```

▼ given table & result



excel_sql_inventory_data — Preview

product_id:	int
product_name:	varchar
product_type:	varchar
unit:	varchar
price_unit:	float
wholesale:	float
current_inventory:	int

excel_sql_transaction_data — Preview

transaction_id:	int
time:	datetime
product_id:	int

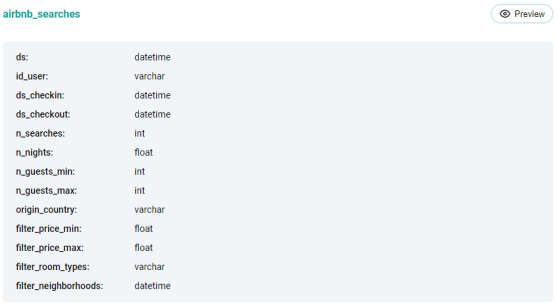
Output — View the output in a separate browser tab

product_id	product_name	count(t.transaction_id)
1	strawberry	5
2	apple_fuji	9
3	orange	6
4	clementines	6
6	blood_lime	5
7	tayberry	6
8	pluot	8
9	tangelo	9
10	pomello	6
11	pineberry	10
12	vegan_egg_substitute	10
14	falafel_chips	7
15	sweet_potato_hockey_pucks	9

▼ 18 (ID 9640**) Find the average number of searches from each user**

```
-- Airbnb | Medium | General Practice | ID 9640
/*
Find the average number of searches made by each user and present the result with their corresponding user id.
*/

select
    id_user,
    avg(n_searches) as avg_search
from airbnb_searches
group by 1
order by 2 desc
;
```

▼ given table & result



airbnb_searches — Preview

ds:	datetime
id_user:	varchar
ds_checkin:	datetime
ds_checkout:	datetime
n_searches:	int
n_nights:	float
n_guests_min:	int
n_guests_max:	int
origin_country:	varchar
filter_price_min:	float
filter_price_max:	float
filter_room_types:	varchar
filter_neighborhoods:	datetime

Output View the output in a separate browser tab

id_user	avg_search
37a63847-b09a-4f32-81a5-97cfb8e84c6d	179
b11cd744-101d-409b-9a55-7e151f2e79d5	109
62d09c95-c3d2-44e6-9081-a3485618227d	88
d528e24b-7c1f-446f-9bb0-a4ecb77c3acd	81
2889fccc-37ab-4a66-8d64-41b31314c7fc	79
bdaf2e68-86dd-40d9-a5a1-9cc95ea25d91	73
ebe81cf8-6037-43f2-81d2-fd386f5da74f	70.5
0a8e121b-c09c-4de1-abcc-81bce87de29e	66
eda96881-7b23-413a-9f6f-b45dc9fe2a5b	57.5
f54989cf-459b-409d-be6a-9534a53cc4a9	53.5
25cfc206-89aa-4e63-b2f4-3cbb8631d9fb	52.5
1a66fe1c-fea6-4ec6-96c4-3ea3e0c7815e	52
ea445eea-3fac-4edd-b1d6-569f57f6abe4	46
02f0a750-34da-4268-94e8-f1a371f0460e	38.429
9e5e2865-f257-4d19-9f79-9388ae925ad7	37

▼ 19 (ID 2042**) Employees' Years In Service**

```
-- Uber | Medium | Active Interview | ID 2042
/*
Find employees who have worked for Uber for more than 2 years (730 days) and check to see if they're still part of the company. Output
Output the first name, last name, whether or not the employee is still working for Uber, and the number of years at the company.
*/

select
    first_name,
    last_name,
    case when termination_date is null then 'Yes'
    else 'No' end as is_still_working,
    DATEDIFF('2021/04/01', hire_date)*1.0/730 AS yearDiff
from uber_employees
where DATEDIFF('2021/04/01', hire_date)*1.0/730>2
order by is_still_working desc, DATEDIFF('2021/04/01', hire_date)*1.0/730 desc;
```

▼ given table & result

uber_employees Preview

first_name:	varchar
last_name:	varchar
id:	int
hire_date:	datetime
termination_date:	datetime
salary:	int

Output

View the output in a separate browser tab

first_name	last_name	is_still_working	yearDiff
Joe	Jarrod	Yes	6.082
Nancy	Soley	Yes	6.082
Kelly	Smalls	Yes	6.082
Joe	Carlos	Yes	5.821
Karen	Adam	Yes	5.762
Smith	John	Yes	5.742
Nataly	Joe	Yes	5.682
David	Adrian	Yes	5.14
Ina	Eli	Yes	4.934
Meena	Santiago	Yes	4.853
Mina	Robert	Yes	4.821
Karla	Olivier	Yes	4.484
Christine	Liam	Yes	4.433
Tooby	Nelson	Yes	4.352
Mounir	Logan	Yes	3.984
Bauer	Miller	Yes	3.638

▼ 20 (ID 9910**) Favorite Customer**

```
-- Amazon | Medium | Interview Questions | ID 9910
/*
Find "favorite" customers based on the order count and the total cost of orders.
A customer is considered as a favorite if he or she has placed more than 3 orders and with the total cost of orders more than $100.


Output the customer's first name, city, number of orders, and total cost of orders.
*/

select
    c.first_name,
    c.city,
    count(o.id) as num_order,
    sum(o.total_order_cost) as total_cost_order
from customers c
join orders o on c.id = o.cust_id
group by 1, 2
having num_order > 3 and total_cost_order > 100
;
```

▼ given table & result

customers

Preview

id:	int
first_name:	varchar
last_name:	varchar
city:	varchar
address:	varchar
phone_number:	varchar

orders

Preview

id:	int
cust_id:	int
order_date:	datetime
order_details:	varchar
total_order_cost:	int

Output

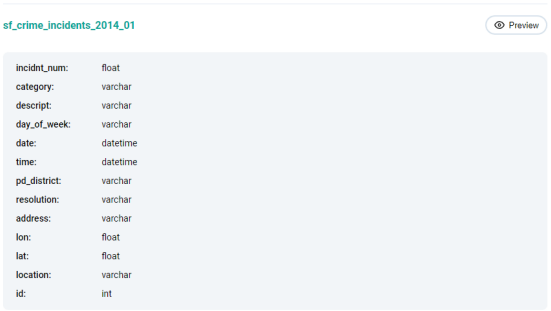
View the output in a separate browser tab

first_name	city	num_order	total_cost_order
Farida	San Francisco	4	260
Jill	Austin	8	535
Mia	Miami	7	540

▼ 21 (ID 9748**) Find districts with the most crime incidents**

```
-- City of San Francisco | Medium | General Practice | ID 9748
/*
Find districts alongside their crime incidents.
Output the district name alongside the number of crime occurrences.
Order records based on the number of occurrences in descending order.
*/

select
    pd_district,
    count(id) as num_crime_incidnt
from sf_crime_incidents_2014_01
group by 1
order by 2 desc
;
```

▼ given table & result



sf_crime_incidents_2014_01 — Preview

incidnt_num:	float
category:	varchar
descript:	varchar
day_of_week:	varchar
date:	datetime
time:	datetime
pd_district:	varchar
resolution:	varchar
address:	varchar
lon:	float
lat:	float
location:	varchar
id:	int

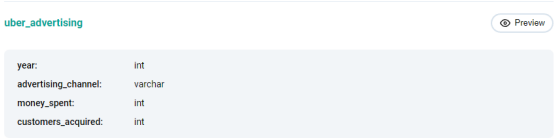
Output — View the output in a separate browser tab

pd_district	num_crime_incidnt
SOUTHERN	23
NORTHERN	18
INGLESIDE	11
TENDERLOIN	11
MISSION	9
CENTRAL	7
BAYVIEW	7
TARAVAL	6
RICHMOND	4
PARK	4

▼ 22 (ID 10012**) Advertising Channel Effectiveness**

```
-- Uber | Medium | General Practice | ID 10012
/*
Find the average effectiveness of each advertising channel in the period from 2017 to 2018 (both included).
The effectiveness is calculated as the ratio of total money spent to total customers aquired.

Output the advertising channel along with corresponding average effectiveness. Sort records by the average effectiveness in ascending o
*/

select
    advertising_channel,
    avg(money_spent/customers_acquired) as effectiveness
from uber_advertising
where year in (2017, 2018)
group by 1
order by 2
;
```

▼ given table & result



uber_advertising — Preview

year:	int
advertising_channel:	varchar
money_spent:	int
customers_acquired:	int

Output — View the output in a separate browser tab

advertising_channel	effectiveness
radio	38.333
tv	55.68
busstops	79.167
celebrities	108.365
billboards	186.556
buses	784.195

▼ 23 (ID 9892**) Second Highest Salary**

```
-- Amazon | Medium | Interview Questions | ID 9892
/*
Find the second highest salary of employees.
*/

with
table_salary as
    (select
        first_name,
        last_name,
        salary,
        rank() over(order by salary desc) as ranking
    from employee)

select *
from table_salary
where ranking = 2
;
```

▼ given table & result

employee — Preview

id:	int
first_name:	varchar
last_name:	varchar
age:	int
sex:	varchar
employee_title:	varchar
department:	varchar
salary:	int
target:	int
bonus:	int
email:	varchar
city:	varchar
address:	varchar
manager_id:	int

Output — View the output in a separate browser tab

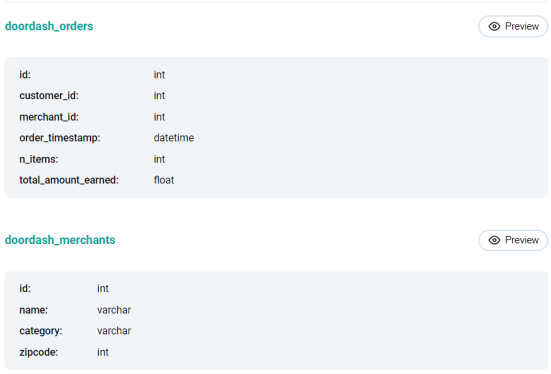
first_name	last_name	salary	ranking
Allen	Wang	200000	2

▼ 24 (ID 2093**) First Time Orders**

```
-- DoorDash | Medium | Active Interview | ID 2093
/*
For each merchant, find how many orders and first-time orders they had. First-time orders are meant from the perspective of a customer
In order words, for how many customers was this the first-ever merchant they ordered with?
Output the name of the merchant, the total number of their orders and the number of these orders that were first-time orders.
*/

with
all_order as
    (select
```

```
        o.merchant_id,
        m.name,
        count(o.id) as orders
    from doordash_orders o
    join doordash_merchants m on o.merchant_id = m.id
    group by 1),

first_orders as
    (select
        customer_id,
        o.merchant_id,
        min(order_timestamp) as first_order
    from doordash_orders o
    join doordash_merchants m on o.merchant_id = m.id
    group by 1),

count_first as
    (select
        merchant_id,
        count(first_order) as count_first_order
    from first_orders
    group by 1)

select
    a.merchant_id,
    a.name,
    a.orders as all_orders,
    f.count_first_order
from all_order a
left join count_first f on a.merchant_id = f.merchant_id
order by 1
;
```

▼ given table & result



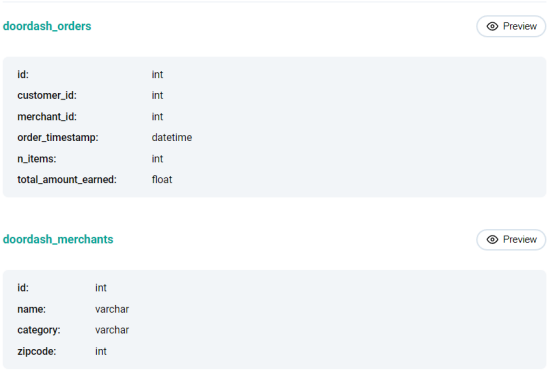
Output View the output in a separate browser tab

merchant_id	name	all_orders	count_first_order
1	Treehouse Pizza	8	5
2	Thai Lion	14	7
3	Meal Raven	12	
4	Burger A1	4	
5	Sushi Bay	7	
6	Tacos You	7	

▼ 25 (ID 10083**) Start Dates Of Top Drivers**

```
-- Lyft | Medium | General Practice | ID 10083
/*
Find contract starting dates of the top 5 most paid Lyft drivers.
Consider drivers who are still working with Lyft.
*/

select
    x.ranking,
    x.start_date,
    x.yearly_salary
from
    (select
        *,
        rank() over(order by yearly_salary desc) as ranking
    from lyft_drivers
    where end_date is null) x
where ranking <= 5
;
```

▼ given table & result



Output View the output in a separate browser tab

merchant_id	name	all_orders	count_first_order
1	Treehouse Pizza	8	5
2	Thai Lion	14	7
3	Meal Raven	12	
4	Burger A1	4	
5	Sushi Bay	7	
6	Tacos You	7	

▼ 26 (ID 9881**) Make a report showing the number of survivors and non-survivors by passenger class**

```
-- Google | Medium | General Practice | ID 9881
/*
Make a report showing the number of survivors and non-survivors by passenger class.
Classes are categorized based on the pclass value as:
pclass = 1: first_class
pclass = 2: second_classs
pclass = 3: third_class
Output the number of survivors and non-survivors by each class.
*/

with
survived as
    (select
        pclass,
        count(passengerid) as svvd
    from titanic
    where survived = 1
    group by 1),

not_survived as
    (select
        pclass,
        count(passengerid) as not_svvd
    from titanic
    where survived != 1
    group by 1)

select
    s.pclass,
    s.svvd,
    n.not_svvd,
    100.0*svvd/(svvd+not_svvd) as prcnt_svvd
from survived s
join not_survived n on s.pclass = n.pclass
order by 1
;
```

▼ given table & result

titanic

Preview

passengerid:	int
survived:	int
pclass:	int
name:	varchar
sex:	varchar
age:	float
sibsp:	int
parch:	int
ticket:	varchar
fare:	float
cabin:	varchar
embarked:	varchar

Output

View the output in a separate browser tab

pclass	svvd	not_svvd	prcnt_svvd
1	10	11	47.619
2	12	6	66.667
3	19	42	31.148

▼ 27 (ID 2122) **Products Never Sold**

```
-- Meta/Facebook | Medium | Active Interview | ID 2122
/*
The VP of Sales feels that some product categories don't sell and can be completely removed from the inventory.
As a first pass analysis, they want you to find what percentage of product categories have never been sold.
*/

with
tot_order as
    (select count(*) as all_order
    from facebook_sales)

select
    p.product_category,
    c.category_name,
    count(p.product_id) as orders,
    all_order,
    100.0*count(p.product_id)/all_order as prcnt
from facebook_sales s
join facebook_products p on s.product_id = p.product_id
right join facebook_product_categories c on p.product_category = c.category_id
cross join tot_order
group by 1
;
```

▼ given table & result

facebook_products

Preview

product_id:	int
product_class:	varchar
brand_name:	varchar
is_low_fat:	varchar
is_recyclable:	varchar
product_category:	int
product_family:	varchar

facebook_sales

Preview

product_id:	int
promotion_id:	int
cost_in_dollars:	int
customer_id:	int
date:	datetime
units_sold:	int

Output

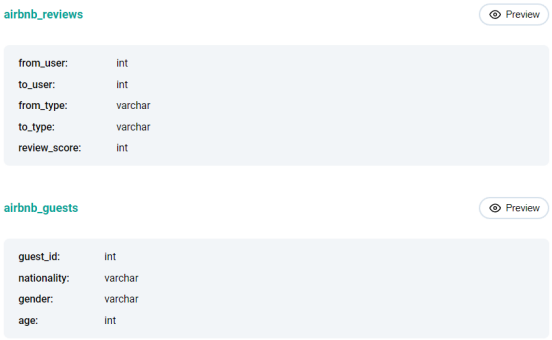
View the output in a separate browser tab

product_category	category_name	orders	all_order	prcnt
1	Fruits	5	30	16.667
2	Kitchen products	15	30	50
3	Gastronomy	10	30	33.333
	Appliances	0	30	0

▼ 28 (ID 10074**) Find the average age of guests reviewed by each host**

```
-- Airbnb | Medium | General Practice | ID 10074
/*
Find the average age of guests reviewed by each host.
Output the user along with the average age.
*/

select
    from_user as id,
    avg(age)
from airbnb_reviews r
join airbnb_guests g on r.to_user = g.guest_id
where r.from_type = 'host' and r.to_type = 'guest'
group by 1
order by 1
;
```

▼ given table & result



Output View the output in a separate browser tab

id	avg(age)
0	28
1	28.107
2	28.067
3	27.044
4	27.4
5	27.933
6	26.731
7	28.35
8	25.071
9	28.04
10	26.1
11	26.913

▼ 29 (ID 9896**) Customers Without Orders**

```
-- Apple | Medium | Interview Questions | ID 9896
/*
Find customers who have never made an order.
Output the first name of the customer.
*/

select
    first_name
from customers c
left join orders o on c.id = o.cust_id
where o.total_order_cost is null
;
```

▼ given table & result

customers

Preview

id:	int
first_name:	varchar
last_name:	varchar
city:	varchar
address:	varchar
phone_number:	varchar

orders

Preview

id:	int
cust_id:	int
order_date:	datetime
order_details:	varchar
total_order_cost:	int

Output View the output in a separate browser tab

first_name
John
Emma
Liam
Mark
Jack
Mona
Lili
Justin
Frank

▼ 30 (ID 10318**) New Products**

```
-- Salesforce | Medium | Interview Questions | ID 10318
/*
You are given a table of product launches by company by year.
Write a query to count the net difference between the number of products companies launched in 2020 with the number of products compani
Output the name of the companies and a net difference of net products released for 2020 compared to the previous year.
*/

with
x as
    (select
        year,
        company_name,
        count(product_name) as count_product
    from car_launches
    group by 1, 2
    order by 2, 1)

select
    company_name,
    year,
    net_diff
from (select
        *,
        lag(count_product) over(partition by company_name order by year) as b4,
        count_product - lag(count_product) over(partition by company_name order by year) as net_diff
    from x) y
where year = 2020
;
```

▼ given table & result

car_launches

Preview

year:	int
company_name:	varchar
product_name:	varchar

Output View the output in a separate browser tab

company_name	year	net_diff
Chevrolet	2020	2
Ford	2020	-1
Honda	2020	-3
Jeep	2020	1
Toyota	2020	-1

▼ 31 (ID 9650**) Find the top 10 ranked songs in 2010**

```
-- Spotify | Medium | General Practice | ID 9650
/*
What were the top 10 ranked songs in 2010?
Output the rank, group name, and song name but do not show the same song twice.
Sort the result based on the year_rank in ascending order.
*/

select
    *,
    max(id)
from billboard_top_100_year_end
where year = 2010 and year_rank <= 10
group by year_rank
;
```

▼ given table & result

billboard_top_100_year_end

Preview

year:	int
year_rank:	int
group_name:	varchar
artist:	varchar
song_name:	varchar
id:	int

Output

View the output in a separate browser tab

year	year_rank	group_name	artist	song_name	id	max(id)
2010	1	KE$HA	KE$HA	TiK ToK	5909	5909
2010	2	LADY ANTEBELLUM	LADY ANTEBELLUM	Need You Now	5910	5910
2010	3	TRAIN	TRAIN	Hey, Soul Sister	5911	5911
2010	4	KATY PERRY FEAT. SNOOP DOGG	KATY PERRY	California Gurls	5912	5913
2010	5	USHER FEAT. WILL.I.AM	USHER	OMG	5914	5915
2010	6	B.O.B FEAT. HAYLEY WILLIAMS	B.O.B	Airplanes	5916	5917
2010	7	EMINEM FEAT. RIHANNA	EMINEM	Love The Way You Lie	5918	5919
2010	8	LADY GAGA	LADY GAGA	Bad Romance	5920	5920
2010	9	TAIO CRUZ	TAIO CRUZ	Dynamite	5921	5921
2010	10	TAIO CRUZ FEAT. LUDACRIS	TAIO CRUZ	Break Your Heart	5922	5923

▼ 32 (ID 10048**) Top Businesses With Most Reviews**

```
-- Yelp | Medium | General Practice | ID 10048
/*
Find the top 5 businesses with most reviews. Assume that each row has a unique business_id such that the total reviews for each busines
Output the business name along with the total number of reviews and order your results by the total reviews in descending order.
*/

with
cte as
    (select
        name,
        review_count,
        rank() over(order by review_count desc) as ranking
    from yelp_business)

select *
from cte
where ranking <= 5;
```

▼ given table & result

yelp_business

Preview

business_id:	varchar
name:	varchar
neighborhood:	varchar
address:	varchar
city:	varchar
state:	varchar
postal_code:	varchar
latitude:	float
longitude:	float
stars:	float
review_count:	int
is_open:	int
categories:	varchar

Output View the output in a separate browser tab

name	review_count	ranking
Iron Chef	331	1
Jacs Dining and Tap House	197	2
Grimaldi's Pizzeria	187	3
Signs Restaurant	120	4
Kassab's	101	5

▼ 33 (ID 9614) **Find the average difference between booking and check-in dates**

```
-- Airbnb | Medium | General Practice | ID 9614
/*
Find the average number of days between the booking and check-in dates for AirBnB hosts.
Order the results based on the average number of days in descending order.
*/

select
    avg(datediff(ds_checkin, ts_booking_at))
from airbnb_contacts
;
```

▼ given table & result

airbnb_contacts

Preview

id_guest:	varchar
id_host:	varchar
id_listing:	varchar
ts_contact_at:	datetime
ts_reply_at:	datetime
ts_accepted_at:	datetime
ts_booking_at:	datetime
ds_checkin:	datetime
ds_checkout:	datetime
n_guests:	int
n_messages:	int

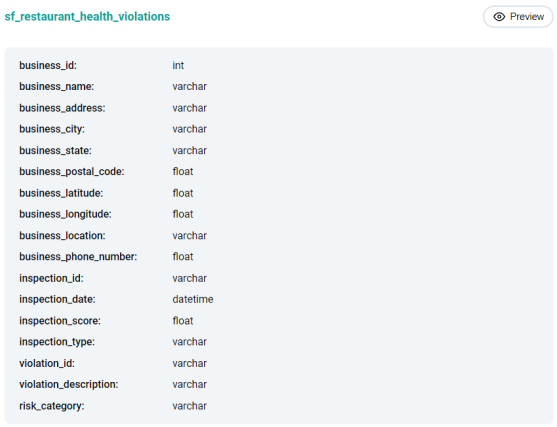
Output View the output in a separate browser tab

avg(datediff(ds_checkin, ts_booking_at))
20.913

▼ 34 (ID 9728) **Number of violations**

```
-- City of San Francisco | Medium | Interview Questions | ID 9728
/*
You're given a dataset of health inspections.
Count the number of violation in an inspection in 'Roxanne Cafe' for each year.
If an inspection resulted in a violation, there will be a value in the 'violation_id' column.
Output the number of violations by year in ascending order.
*/

select
    extract(year from inspection_date) as year
    , count(violation_id) as num_violation
from sf_restaurant_health_violations
where business_name = 'Roxanne Cafe' and violation_id is not null
group by 1
order by 2
;
```

▼ given table & result



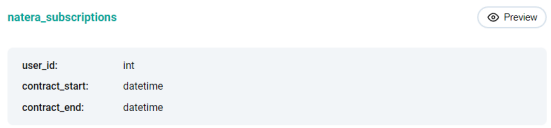
Output View the output in a separate browser tab

year	num_violation
2016	2
2018	3
2015	5

▼ 35 (ID 2074**) Monthly Churn Rate**

```
-- Natera | Medium | Active Interview | ID 2074
/*
Calculate the churn rate of September 2021 in percentages.
The churn rate is the difference between the number of customers on the first day of the month and on the last day of the month, divide
Assume that if customer's contract_end is NULL, their contract is still active.
Additionally, if a customer started or finished their contract on a certain day, they should still be counted as a customer on that day
*/

with
last_month as
    (select count(*) last_day
    from natera_subscriptions
    where contract_end = '2021-09-30' or contract_end is null)

, first_month as
    (select count(*) as first_day
    from natera_subscriptions)

select
    *,
    100.0*(first_day - last_day)/first_day as churn_rate_prcnt
from last_month
cross join first_month
;
```

▼ given table & result



Output View the output in a separate browser tab

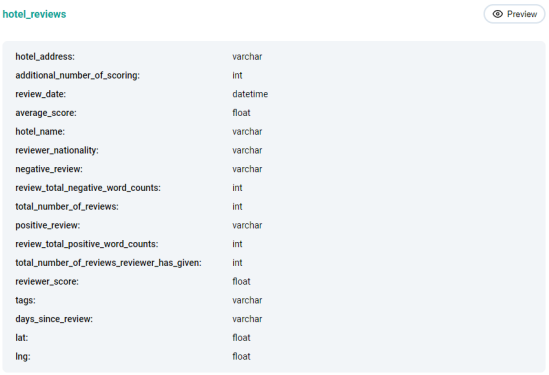
last_day	first_day	churn_rate_prcnt
2	6	66.667

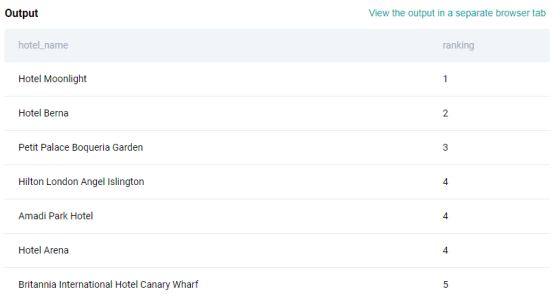
▼ 36 (ID 9880**) Find the top five hotels with the highest total reviews given by a particular reviewer**

```
-- Airbnb | Medium | General Practice | ID 9880
/*
For each hotel find the number of reviews from the most active reviewer. The most active is the one with highest number of total review
Output the hotel name along with the highest total reviews of that reviewer. Output only top 5 hotels with highest total reviews.
Order records based on the highest total reviews in descending order.
```

```
*/

with
active_review as
    (select reviewer_nationality
    from(select
            reviewer_nationality
            , sum(total_number_of_reviews_reviewer_has_given)
        from hotel_reviews
        group by 1
        order by 2 desc
        limit 1) x)

, hotel_active as
    (select
        hotel_name
        , sum(total_number_of_reviews_reviewer_has_given) as num_review
    from hotel_reviews
    where reviewer_nationality = (select * from active_review)
    group by 1
    order by 2 desc)

select *
from (select
        hotel_name,
        dense_rank() over(order by num_review desc) as ranking
    from hotel_active) y
where ranking <= 5
;
```

▼ given table & result



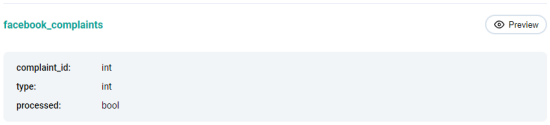


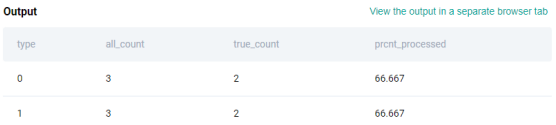
▼ 37 (ID 9781**) Find the rate of processed tickets for each type**

```
-- Meta/Facebook | Medium | General Practice | ID 9781
/*
Find the rate of processed tickets for each type.
*/

with
all_counted as
    (select
        type
        , count(type) as all_count
    from facebook_complaints
    group by 1)

, true_counted as
    (select
        type,
        count(type) as true_count
    from facebook_complaints
    where processed = TRUE
    group by 1)
```

```
select
    a.type
    , all_count
    , true_count
    , 100.0 * true_count/all_count as prcnt_processed
from all_counted a
join true_counted t on a.type = t.type
;
```

▼ given table & result

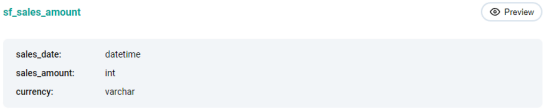


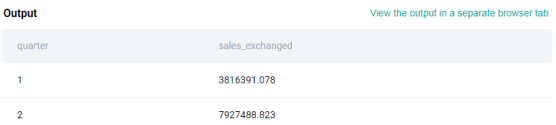


▼ 38 (ID 2041**) Total Sales In Different Currencies**

```
-- Salesforce | Medium | Active Interview | ID 2041
/*
You work for a multinational company that wants to calculate total sales across all their countries they do business in.
You have 2 tables, one is a record of sales for all countries and currencies the company deals with, and the other holds currency excha
Calculate the total sales, per quarter, for the first 2 quarters in 2020, and report the sales in USD currency.
*/

with
sales_table as
    (select
        *
        , case when extract(month from sales_date) <= 3 then 1
        when extract(month from sales_date) <= 6 then 2
        when extract(month from sales_date) <= 9 then 3
        else 4 end as quarter
    from sf_sales_amount)

select
    s.quarter
    , sum(s.sales_amount * e.exchange_rate) as sales_exchanged
from sales_table s
join sf_exchange_rate e on s.currency = e.source_currency
where extract(year from sales_date) = 2020 and s.quarter <= 2
group by 1
;
```

▼ given table & result



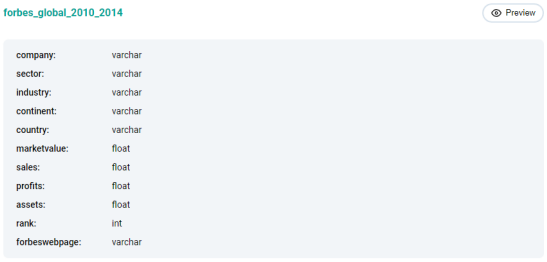


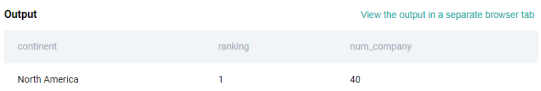
▼ 39 (ID 9804**) Find continents that have the highest number of companies**

```
-- Forbes | Medium | General Practice | ID 9804
/*
```

```
Find continents that have the highest number of companies.
Output the continents along with the corresponding number of companies.
*/

select *
from(select
        continent
        , rank() over(order by count(company) desc) as ranking
        , count(company) as num_company
    from forbes_global_2010_2014
    group by 1) x
where ranking = 1
;
```

▼ given table & result

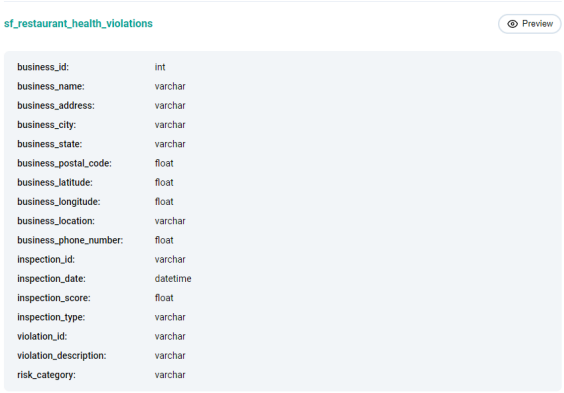




▼ 40 (ID 10130**) Find the number of inspections for each risk category by inspection type**

```
-- City of San Francisco | Medium | Interview Questions | ID 10130
/*
Find the number of inspections that resulted in each risk category per each inspection type.
Consider the records with no risk category value belongs to a separate category.
Output the result along with the corresponding inspection type and the corresponding total number of inspections per that type. The out
Order the result based on the number of inspections per inspection type in descending order.
*/

select
    inspection_type
    , risk_category
    , count(inspection_id) as count_inspection
from sf_restaurant_health_violations
group by 1, 2
order by 1, 2
;
```

▼ given table & result

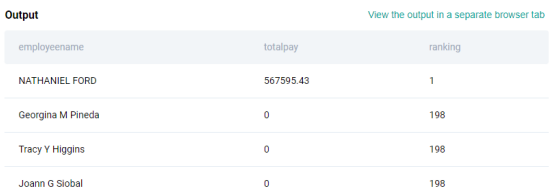


inspection_type	risk_category	count_inspection
Complaint	High Risk	1
Complaint	Low Risk	1
Complaint	Moderate Risk	3
Complaint		5
New Construction		12
New Ownership	High Risk	1
New Ownership	Low Risk	1
New Ownership	Moderate Risk	1
New Ownership		4
Non-inspection site visit		6
Reinspection/Followup	Low Risk	2
Reinspection/Followup	Moderate Risk	2
Reinspection/Followup		29
Routine - Unscheduled	High Risk	36
Routine - Unscheduled	Low Risk	105
Routine - Unscheduled	Moderate Risk	72
Routine - Unscheduled		11

▼ 41 (ID 9978**) Find employees who earned the highest and the lowest total pay without any benefits**

```
-- City of San Francisco | Medium | General Practice | ID 9978
/*
Find employees who earned the highest and the lowest total pay without any benefits.
Output the employee name along with the total pay.
Order records based on the total pay in descending order.
*/

with
table_rank as
    (select
        employeename
        , totalpay
        , rank() over(order by totalpay desc) as ranking
    from sf_public_salaries)

, max_rank as
    (select
        max(ranking)
    from table_rank)

, min_rank as
    (select
        min(ranking)
    from table_rank)

select *
from table_rank
where ranking = (select * from max_rank)
    or  ranking = (select * from min_rank)
;
```

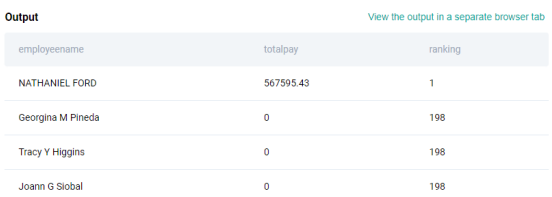
▼ given table & result



▼ 41 (ID 9978**) Find employees who earned the highest and the lowest total pay without any benefits**

```
-- City of San Francisco | Medium | General Practice | ID 9978
/*
Find employees who earned the highest and the lowest total pay without any benefits.
Output the employee name along with the total pay.
Order records based on the total pay in descending order.
*/

with
table_rank as
    (select
        employeename
        , totalpay
        , rank() over(order by totalpay desc) as ranking
    from sf_public_salaries)

, max_rank as
    (select
        max(ranking)
    from table_rank)

, min_rank as
    (select
        min(ranking)
    from table_rank)

select *
from table_rank
where ranking = (select * from max_rank)
    or  ranking = (select * from min_rank)
;
```

▼ given table & result

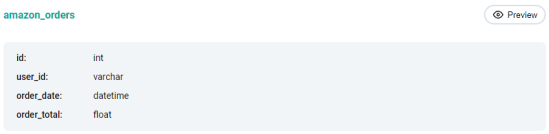


▼ 42 (ID 2095**) Three Purchases**

```
-- Amazon | Medium | Active Interview | ID 2095
/*
```

```
List the IDs of customers who made at least 3 orders in both 2020 and 2021.
*/

with
cte as
    (select
        *
        , extract(year from order_date) as year
    from amazon_orders
    where extract(year from order_date) in (2020, 2021))

select
    user_id
    , year
    , count(user_id) as count_order
from cte
group by 2, 1
having count(user_id)>=3
order by 2, 1
;
```

▼ given table & result



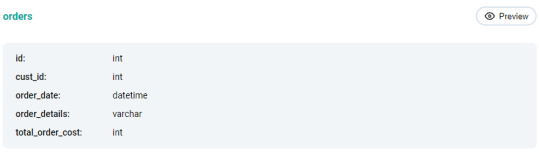
Output View the output in a separate browser tab

user_id	year	count_order
U203	2020	3
U205	2020	5
U206	2020	4
U202	2021	5
U203	2021	4
U205	2021	3

▼ 43 (ID 9782**) Customer Revenue In March**

```
-- Meta/Facebook | Medium | Interview Questions | ID 9782
/*
Calculate the total revenue from each customer in March 2019. Include only customers who were active in March 2019.
Output the revenue along with the customer id and sort the results based on the revenue in descending order.
*/

select
    cust_id
    , sum(total_order_cost) as total_rev
from orders
where order_date between '2019-03-01' and '2019-03-31'
group by 1
order by 2 desc
;
```

▼ given table & result



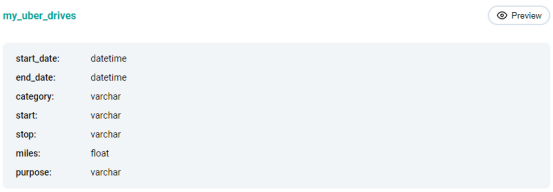
Output View the output in a separate browser tab

cust_id	total_rev
3	210
15	150
7	80
12	20

▼ 44 (ID 10169**) Highest Total Miles**

```
-- Uber | Medium | Interview Questions | ID 10169
/*
You're given a table of Uber rides that contains the mileage and the purpose for the business expense.
You're asked to find business purposes that generate the most miles driven for passengers that use Uber for their business transportati
Find the top 3 business purpose categories by total mileage.
*/

with
cte as
    (select
        purpose
        , sum(miles) as tot_miles
        , rank() over(order by sum(miles) desc) as ranking
    from my_uber_drives
    group by 1
    order by 2 desc)

select *
from cte
where ranking <= 3
;
```

▼ given table & result



Output View the output in a separate browser tab

purpose	tot_miles	ranking
Meeting	283.9	1
Customer Visit	241.1	2
Commute	180.2	3

▼ 45 (ID 9738**) Business Inspection Scores**

```
-- City of San Francisco | Medium | Interview Questions | ID 9738
/*
Find the average inspection score for every available inspection across all businesses. You can include all the inspection scores from
Business types are 'Restaurant', 'Cafe', 'Taqueria', 'Kitchen', 'Garden', 'School' and 'Other'.
Output the average inspection scores by business type and inspection type combination.
*/

with
cte1 as
    (select
        *
        , 'Restaurant' as business_type
    from sf_restaurant_health_violations
    where business_name like ('%Restaurant%'))

, cte2 as
    (select
        *
        , 'Cafe' as business_type
    from sf_restaurant_health_violations
    where business_name like ('%Cafe%'))

, cte3 as
    (select
        *
        , 'Taqueria' as business_type
    from sf_restaurant_health_violations
    where business_name like ('%Taqueria%'))

, cte4 as
    (select
        *
        , 'Kitchen' as business_type
    from sf_restaurant_health_violations
    where business_name like ('%Kitchen%'))
```

```
, cte5 as
    (select
        *
        , 'Garden' as business_type
    from sf_restaurant_health_violations
    where business_name like ('%Garden%'))


, cte6 as
    (select
        *
        , 'School' as business_type
    from sf_restaurant_health_violations
    where business_name like ('%School%'))

, cte7 as
    (select
        *,
        'Other' as business_type
    from sf_restaurant_health_violations
    where business_id not in (select business_id from cte1)
    and business_id not in (select business_id from cte2)
    and business_id not in (select business_id from cte3)
    and business_id not in (select business_id from cte4)
    and business_id not in (select business_id from cte5)
    and business_id not in (select business_id from cte6))

select *
from
    (select
       business_type
       , inspection_type
       , avg(inspection_score) as avg_score
    from cte1
    group by 1, 2
    union
    select
       business_type
       , inspection_type
       , avg(inspection_score) as avg_score
    from cte2
    group by 1, 2
    union
    select
       business_type
       , inspection_type
       , avg(inspection_score) as avg_score
    from cte3
    group by 1, 2
    union
    select
       business_type
       , inspection_type
       , avg(inspection_score) as avg_score
    from cte4
    group by 1, 2
    union
    select
       business_type
       , inspection_type
       , avg(inspection_score) as avg_score
    from cte5
    group by 1, 2
    union
    select
       business_type
       , inspection_type
       , avg(inspection_score) as avg_score
    from cte6
    group by 1, 2
    union
    select
       business_type
       , inspection_type
       , avg(inspection_score) as avg_score
    from cte7
    group by 1, 2) x
order by 1, 2
;
```

▼ given table & result

sf_restaurant_health_violations

Preview

business_id:	int
business_name:	varchar
business_address:	varchar
business_city:	varchar
business_state:	varchar
business_postal_code:	float
business_latitude:	float
business_longitude:	float
business_location:	varchar
business_phone_number:	float
inspection_id:	varchar
inspection_date:	datetime
inspection_score:	float
inspection_type:	varchar
violation_id:	varchar
violation_description:	varchar
risk_category:	varchar

Output

View the output in a separate browser tab

business_type	inspection_type	avg_score
Cafe	Non-inspection site visit	
Cafe	Reinspection/Followup	
Cafe	Routine - Unscheduled	80.1
Garden	Routine - Unscheduled	83
Kitchen	Reinspection/Followup	
Kitchen	Routine - Unscheduled	91
Kitchen	Structural Inspection	
Other	Complaint	
Other	New Construction	
Other	New Ownership	
Other	Non-inspection site visit	
Other	Reinspection/Followup	
Other	Routine - Unscheduled	84.784
Other	Structural Inspection	
Restaurant	New Ownership	
Restaurant	Reinspection/Followup	
Restaurant	Routine - Unscheduled	80.417

▼ 46 (ID 10070**) DeepMind employment competition**

```
-- Google | Medium | General Practice | ID 10070
/*
Find the winning teams of DeepMind employment competition.
Output the team along with the average team score.
Sort records by the team score in descending order.
*/

select
    p.team_id
    , avg(s.member_score) as avg_score
from google_competition_participants p
join google_competition_scores s on s.member_id = p.member_id
group by 1
order by 2 desc
;
```

▼ given table & result

google_competition_participants

Preview

member_id:	int
team_id:	int

google_competition_scores

Preview

member_id:	int
member_score:	float

team_id	avg_score
9	0.816
4	0.79
14	0.786
15	0.784
10	0.778
8	0.77
7	0.77
12	0.765
5	0.759
6	0.756
13	0.754
1	0.747
3	0.737
11	0.715
2	0.715

▼ 47 (ID 9724**) Find the postal code which has the highest average inspection score**

```
-- City of San Francisco | Medium | Interview Questions | ID 9724
/*
Find the postal code which has the highest average inspection score.
Output the corresponding postal code along with the result.
*/

select
    business_postal_code
    , avg(inspection_score) as avg_score
from sf_restaurant_health_violations
group by 1
order by 2 desc
;
```

▼ given table & result

sf_restaurant_health_violations

Preview

business_id:	int
business_name:	varchar
business_address:	varchar
business_city:	varchar
business_state:	varchar
business_postal_code:	float
business_latitude:	float
business_longitude:	float
business_location:	varchar
business_phone_number:	float
inspection_id:	varchar
inspection_date:	datetime
inspection_score:	float
inspection_type:	varchar
violation_id:	varchar
violation_description:	varchar
risk_category:	varchar

business_postal_code	avg_score
94132	92.5
94107	92.125
94127	92
94117	91.333
94115	90.2
94104	90.2
94111	90
	89.75
94122	87.5
94133	86.88
94110	86.769
94114	86.6
94112	85.667
94103	85.381
94105	85.375

▼ 48 (ID 10144**) Average Weight of Medal-Winning Judo**

```
-- ESPN | Medium | General Practice | ID 10144
/*
Find the average weight of medal-winning Judo players of each team with a minimum age of 20 and a maximum age of 30.
Consider players at the age of 20 and 30 too. Output the team along with the average player weight.
*/

select
    team
    , avg(weight) as avg_weight
from olympics_athletes_events
where (age between 20 and 30)
    and (medal is not null)
group by 1
having avg(weight) is not null
order by 2 desc
;
```

▼ given table & result

olympics_athletes_events

Preview

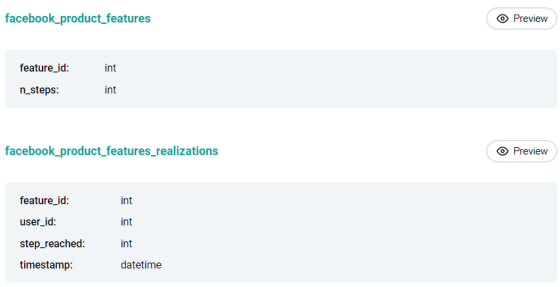
id:	int
name:	varchar
sex:	varchar
age:	float
height:	float
weight:	datetime
team:	varchar
noc:	varchar
games:	varchar
year:	int
season:	varchar
city:	varchar
sport:	varchar
event:	varchar
medal:	varchar
non_team:	datetime

team	avg_weight
Great Britain	91.667
Norway	85
Uruguay	85
Georgia	84
Fiji	81
New Zealand	79
Argentina	77
Argonaut Rowing Club	77
South Africa	76
France	75.667
Winnipeg Shamrocks-1	75
Switzerland	75
Italy	73.5
Denmark	73
United States	70.333

▼ 49 (ID 9792) User Feature Completion

```
-- Meta/Facebook | Medium | General Practice | ID 9792
/*
An app has product features that help guide users through a marketing funnel. Each feature has "steps" (i.e., actions users can take) a
What is the average percentage of completion for each feature?
*/

select
    f.feature_id
    , avg(r.step_reached/f.n_steps)*100.0 as pcnt_completion
from facebook_product_features f
left join facebook_product_features_realizations r on f.feature_id = r.feature_id
group by 1
;
```

▼ given table & result



Output View the output in a separate browser tab

feature_id	pcnt_completion
0	51.667
1	76.19
2	

▼ 50 (ID 10350) Algorithm Performance

```
-- Meta/Facebook | Medium | Interview Questions | ID 10350
/*
Meta/Facebook is developing a search algorithm that will allow users to search through their post history.
```

```
You are assigned to evaluate the performance of this algorithm.

We have a table with the user's search term, search result positions, and whether or not the user clicked on the search result.

Write a query that assigns ratings to the searches in the following way:
- If the search was not clicked for any term, assign the search with rating=1
- If the search was clicked but the top position of clicked terms was outside the top 3 positions, give it a rating=2
- If the search was clicked and the top position of a clicked term was in the top 3 positions, give it a rating=3
*/

select
    search_id
    , case when clicked = 0 then 1
    when (clicked = 1) and (search_results_position>3) then 2
    when (clicked = 1) and (search_results_position<=3) then 3
    end as rating
from fb_search_events
;
```

▼ given table & result

fb_search_events

Preview

search_id:	int
search_term:	varchar
clicked:	int
search_results_position:	int

Output

View the output in a separate browser tab

search_id	rating
1	2
2	2
2	2
3	3
3	2
5	3
6	3
10	1
11	1
14	3
14	3
17	1
18	3
38	1