BY MANISH KUMAR CHAUDHARY

Interview Question

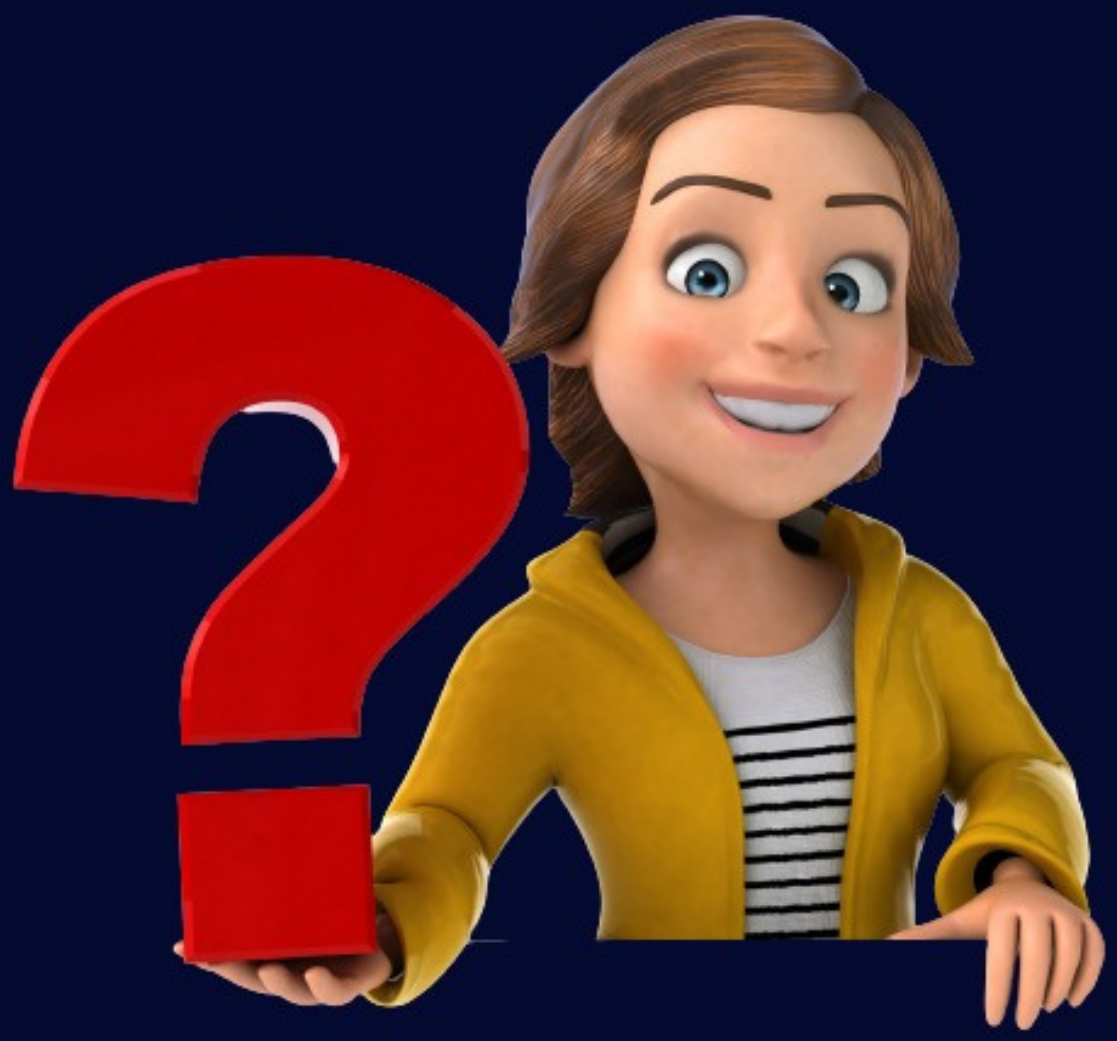
SQL

Blocked Users

DIFFICULTY LEVEL :MEDIUM

Question From

stratascratch



PROBLEM STATEMENT

You are given a table of users who have been blocked from Facebook, together with the date, duration, and the reason for the blocking. The duration is expressed as the number of days after blocking date and if this field is empty, this means that a user is blocked permanently.

For each blocking reason, count how many users were blocked in December 2021. Include both the users who were blocked in December 2021 and those who were blocked before but remained blocked for at least a part of December 2021.

fb_blocked_users

FIELD	TYPE
user_id	int
block_reason	varchar
block_date	datetime
block_duration	float

fb_blocked_users Input Table

user_id	block_reason	block_date	block_duration
3642	Fake Account	2021-12-03	15
2847	Fake Account	2021-12-15	120
1239	Fake Account	2021-11-19	11
3642	Fake Account	2021-12-23	15
2134	Fake Account	2021-11-03	
1309	Fake Account	2021-11-29	14
2049	Spreading Fakenews	2022-01-12	
1382	Spreading Fakenews	2021-12-31	2
4295	Spreading Fakenews	2020-12-14	
3598	Spreading Fakenews	2021-10-15	90
9285	Inappropriate Posts	2021-12-01	
4833	Inappropriate Posts	2021-07-14	30
2348	Inappropriate Posts	2020-11-25	15
1387	Inappropriate Posts	2021-12-26	25

MENTAL APPROAHC

1.We have a block date now e need to get to when they are blocked by adding block duration to the block date.

2.After this we check if either of the block date or the blocked till date belongs to December month.

3.Now we will count the number of users blocked on december month.

QUERY

```sql
SELECT block_reason
      ,COUNT(1) number_of_users
FROM fb_blocked_users
WHERE MONTH(block_date) =12
OR   MONTH(DATEADD(DAY,block_duration,block_date))=12
GROUP BY block_reason
```

QUERY EXAPLANATION

1.We are counting the blocked users for each block reason.

Here, we are checking if month of block date is December.

We are also checking by adding duration to block date to get date till when they are blocked and filtering that it is December month.

OUTPUT

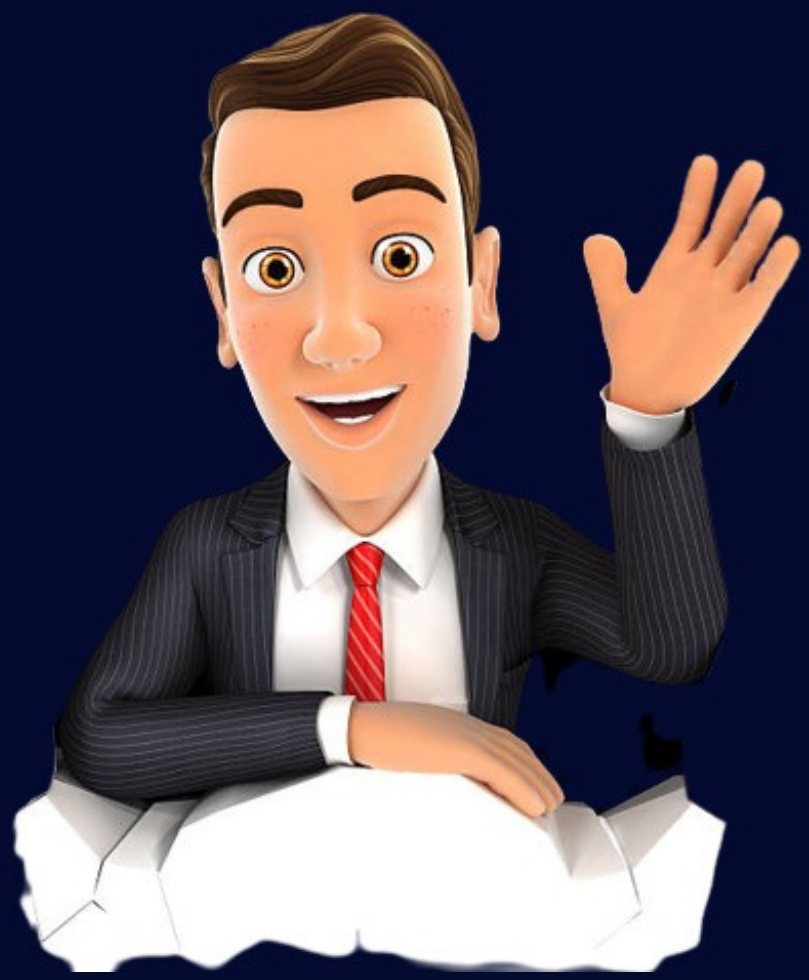
block_reason	number_of_users
Fake Account	4
Inappropriate Posts	3
Spreading Fakenews	2

BY MANISH KUMAR CHAUDHARY

THANK YOU

"Change the way you look at things and the things you look at change."

Wayne W. Dyer



BY MANISH KUMAR CHAUDHARY

Interview Question

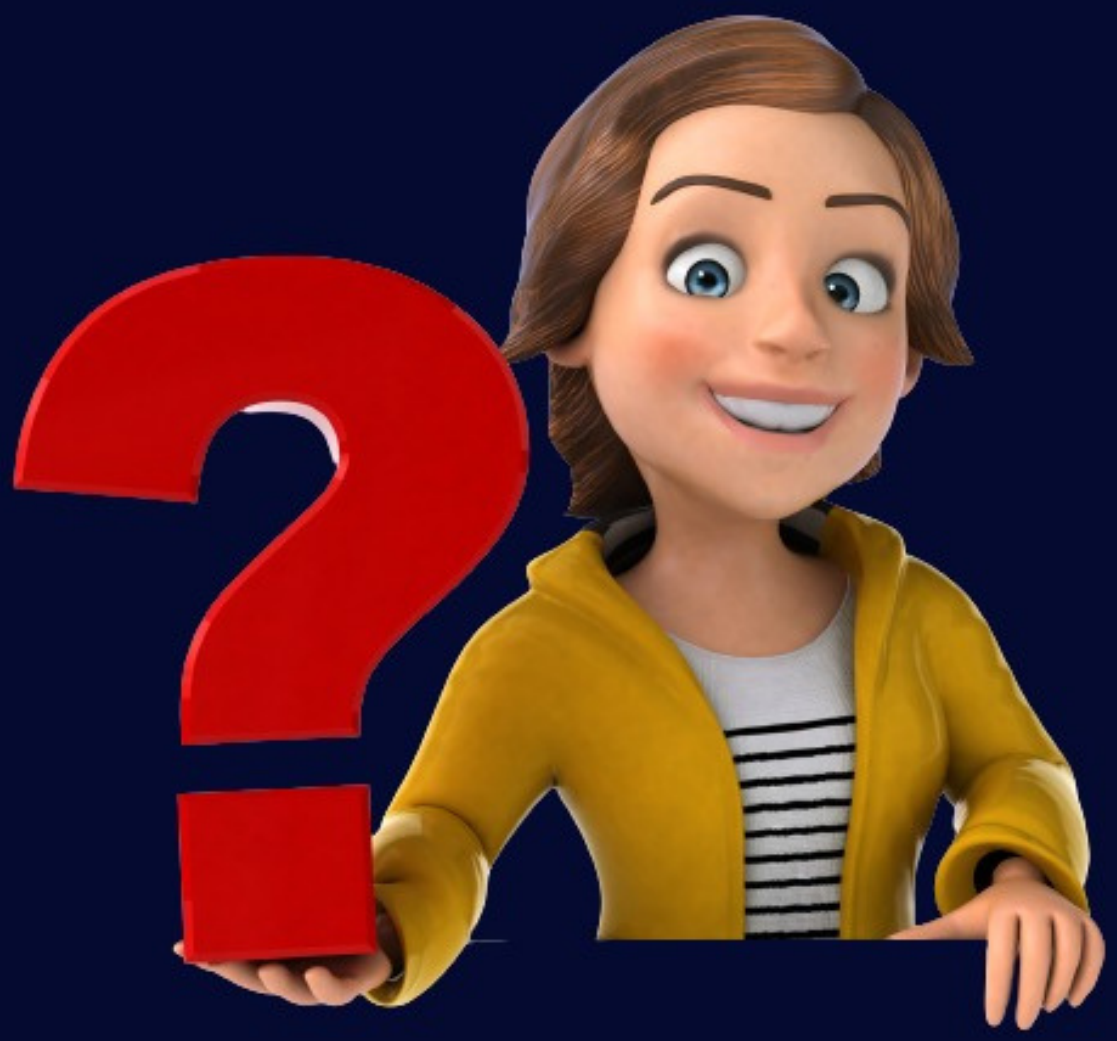
SQL

Comments Distribution

DIFFICULTY LEVEL :HARD

Question From

stratascratch



PROBLEM STATEMENT

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.

fb_users Table

FIELD	TYPE
id	int
name	varchar
joined_at	datetime
city_id	int
device	int

fb_comments Table

FIELD	TYPE
user_id	int
body	varchar
created_at	datetime

fb_users Sample Input Table

id	name	joined_at	city_id	device
4	Ashley Sparks	2020-06-30	63	2185
8	Zachary Tucker	2018-02-18	78	3900
9	Caitlin Carpenter	2020-07-23	60	8592
18	Wanda Ramirez	2018-09-28	55	7904
21	Tonya Johnson	2019-12-02	62	4816
24	Carlos Newman	2020-02-06	74	861
25	Natasha Bradford	2020-02-12	60	9401
27	Jessica Farrell	2019-03-24	67	7190
32	Catherine Hurst	2018-08-22	51	5529
33	Amanda Leon	2017-10-06	77	4597

fb_comments Sample Input Table

user_id	body	created_at
89	Wrong set challenge guess college as position.	2020-01-16
33	Interest always door health military bag. Store smile factor player goal detail TV loss.	2019-12-31
34	Physical along born key leader various. Forward box soldier join.	2020-01-08
46	Kid must energy south behind hold. Research common long state get at issue. Weight technology live plant. His size approach loss.	2019-12-29
25	Or matter will turn only woman fact.	2019-12-21
8	Western east tax group character establish professor. Forward growth material. Before garden military product. Over southern manager. Along series civil theory force language clear.	2020-01-13

QUERY

```sql
WITH comment_cte AS (
SELECT u.id
      ,COUNT(1) no_of_comments
FROM fb_users u
INNER JOIN fb_comments c
ON u.id=c.user_id
WHERE YEAR(u.joined_at) BETWEEN 2018 AND 2020
AND FORMAT(CAST(c.created_at AS date),'MM-yyyy') = '01-2020'
AND u.joined_at ≤ c.created_at
GROUP BY u.id
)
SELECT  no_of_comments
      ,COUNT(id) no_of_users
FROM comment_cte
GROUP BY no_of_comments
ORDER BY no_of_comments
```

QUERY EXPLANATION

1.We have made use of comment_cte to get user_id and COUNT of comments they have made.
Here, we have joined both tables and filtered out to get the records for users who joined between 2018 and 2020. Also filtered out to get the comments that were created on January of 2020 using FORMAT function.
Here, we have also given condition that created_at date is >= joined_at date because there were few records that were commented before joining which is error over here.

2.Now we are simply querying the no_of_comments and COUNTING the number of users.
Here, this will help us to get the number of users for that particular number of comments.

OUTPUT

no_of_comments	no_of_users
1	4
2	6
3	1
4	1
6	1

BY MANISH KUMAR CHAUDHARY

THANK YOU

“Smart people learn from everything and everyone, average people from their experiences, stupid people already have all the answers.”

Socrates

BY MANISH KUMAR CHAUDHARY

Interview Question

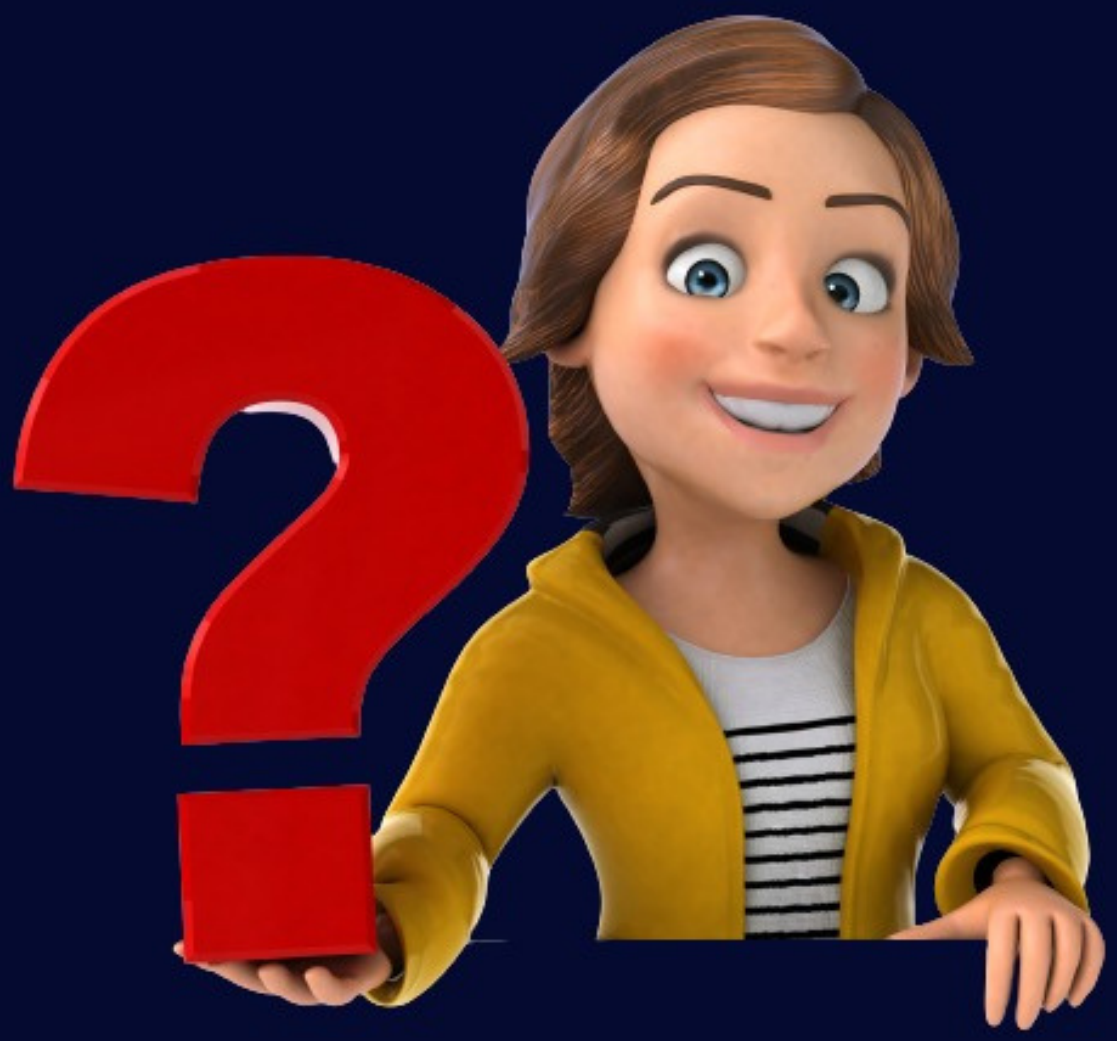
SQL

Completed Trip within 168 Hours

DIFFICULTY LEVEL :HARD

Question From

stratascratch



PROBLEM STATEMENT

An event is logged in the events table with a timestamp each time a new rider attempts a signup (with an event name 'attempted_su') or successfully signs up (with an event name of 'su_success').

For each city and date, determine the percentage of signups in the first 7 days of 2022 that completed a trip within 168 hours of the signup date. HINT: driver id column corresponds to rider id column

signup_events

FIELD	TYPE
rider_id	varchar
city_id	varchar
event_name	varchar
timestamp	datetime

trip_details

FIELD	TYPE
id	varchar
client_id	varchar
driver_id	varchar
city_id	varchar
client_rating	float
driver_rating	float
request_at	datetime
predicted_eta	datetime
actual_time_of_arrival	datetime
status	varchar

signup_events Sample Input Table

rider_id	city_id	event_name	timestamp
r01	c001	su_success	2022-01-01 07:00:00
r02	c002	su_success	2022-01-01 08:00:00
r03	c002	su_success	2022-01-01 08:00:00
r04	c001	attempted_su	2022-01-02 08:00:00
r06	c001	attempted_su	2022-01-02 08:00:00
r04	c001	su_success	2022-01-02 08:15:00
r05	c001	su_success	2022-01-02 08:15:00

trip_details Sample Input Table

id	client_id	driver_id	city_id	client_rating	driver_rating	request_at	predicted_eta	actual_time_of_arrival	status
t01	cl12	r01	c001	4.9	4.5	2022-01-02 09:00:00	2022-01-02 09:10:00	2022-01-02 09:08:00	completed
t02	cl10	r01	c001	4.9	4.8	2022-01-02 11:00:00	2022-01-02 11:10:00	2022-01-02 11:13:00	completed
t03	cl9	r04	c001	4.9	4.8	2022-01-03 11:00:00	2022-01-03 11:10:00	2022-01-03 11:13:00	completed

QUERY

```sql
WITH signups_compeleted_cte AS (
SELECT s.city_id
      ,CAST(s.timestamp AS DATE) date
     ,COUNT(1) signups
     ,COUNT(DISTINCT CASE WHEN t.request_at
                          BETWEEN s.timestamp AND DATEADD(HOUR,168,s.timestamp)
                         AND t.status ='completed'
           THEN t.driver_id
           END) completed_trips
FROM signup_events s
INNER JOIN trip_details t
ON s.rider_id=t.driver_id
WHERE s.event_name='attempted_su'
OR s.event_name='su_sccess'
AND CAST(timestamp AS DATE) BETWEEN '2022-01-01' AND '2022-01-07'
GROUP BY s.city_id
      ,CAST(timestamp AS DATE)
)
SELECT city_id
      ,date
      ,CONCAT(CAST(completed_trips*100.0/signups AS DECIMAL(15,2)),'%')
FROM signups_compeleted_cte
```

QUERY EXPLANATION

1.With signups_completed_cte we are counting the signups and completed rides with the conditions provided.
Here to get the completed rides we are using **CASE WHEN** statement along with **DATEADD** function to get count for rides that got completed within 168 hours after the rider has signed up. We have used COUNT DISTINCT on top of CASE WHEN so that we don't count the same riders multiple times.

2.Now we are calculating the percentage of signups that has completed the rides.
We are using formula
completed_rides*100/signups

OUTPUT OF signups_completed_cte

city_id	date	signups	completed_trips
c001	2022-01-02	10	2
c002	2022-01-03	2	1
c002	2022-01-05	5	1
c001	2022-01-06	2	1

OUTPUT

city_id	date	perc_signups_completed
c001	2022-01-02	20.00%
c002	2022-01-03	50.00%
c002	2022-01-05	20.00%
c001	2022-01-06	50.00%

BY MANISH KUMAR CHAUDHARY

THANK YOU

“You only live once, but if you do it right, once is enough.”

Mae West, actress.

BY MANISH KUMAR CHAUDHARY

Interview Question

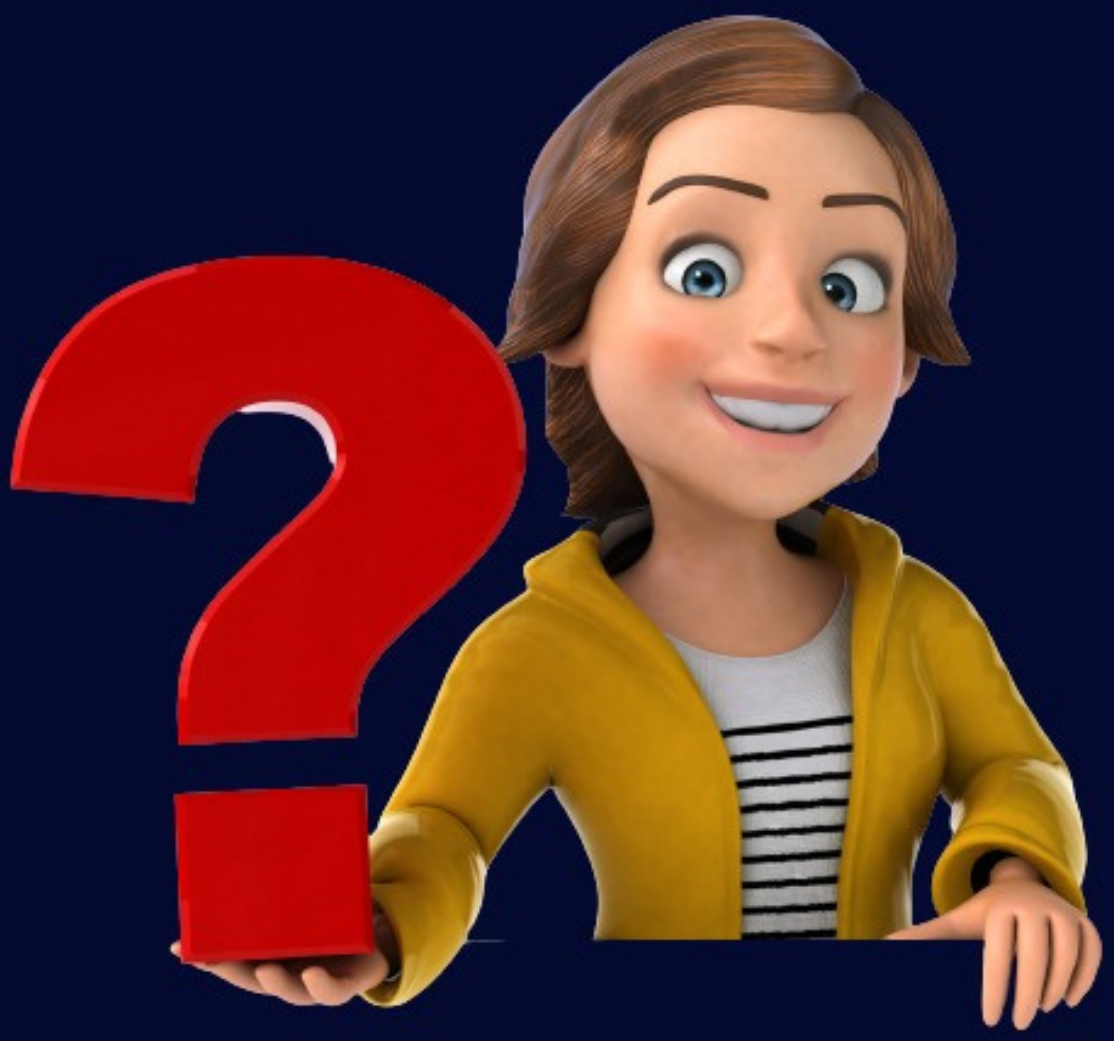
SQL

Consecutive Days

DIFFICULTY LEVEL :HARD

Question From

stratascratch



PROBLEM STATEMENT

Find all the users who were active for 3 consecutive days or more.

sf_events

FIELD	TYPE
date	datetime
acount_id	varchar
user_id	varchar

sf_events Half Input Table

date	account_id	user_id
2021-01-01	A1	U1
2021-01-01	A1	U2
2021-01-06	A1	U3
2021-01-02	A1	U1
2020-12-24	A1	U2
2020-12-08	A1	U1
2020-12-09	A1	U1
2021-01-10	A2	U4
2021-01-11	A2	U4
2021-01-12	A2	U4
2021-01-15	A2	U5
2020-12-17	A2	U4

sf_events Other Half Input Table

2020-12-25	A3	U6
2020-12-25	A3	U6
2020-12-25	A3	U6
2020-12-06	A3	U7
2020-12-06	A3	U6
2021-01-14	A3	U6
2021-02-07	A1	U1
2021-02-10	A1	U2
2021-02-01	A2	U4
2021-02-01	A2	U5
2020-12-05	A1	U8

MENTAL APPROACH

We simply need to find the consecutive days.

To get the consecutive days we must manipulate our data so that for each consecutive date we get same value in corresponding column.

For example: we have these date

2021-04-01	**1**
2021-04-02	2
2021-04-03	3
2021-04-08	4

Now if we subtract the number from that date we will get following result

2021-04-01	**1**	**2021-03-31**
2021-04-02	2	2021-03-31
2021-04-03	3	2021-03-31
2021-04-08	4	2021-04-04

Thus we can see that for date that are consecutive we are getting the same date in another column

QUERY

```sql
WITH rn_cte AS (
SELECT user_id
      ,date
      ,ROW_NUMBER() OVER(PARTITION BY user_id ORDER BY date) rn
FROM sf_events
)
SELECT user_id
FROM rn_cte
GROUP BY user_id,DATEADD(D,-rn,date)
HAVING COUNT(DATEADD(D,-rn,date))>=3
```

OUTPUT

user_id
U4

QUERY EXPLANATION

We have used cte just to make the code look simple and clean, it can be done within one single query as well

1.With rn_cte we are simplying geting the row number for each dates so that we can subtract it from them.
Here row number is given on the basis of partition by user id and order by date in ascending order.

2.We are now simply querying the user_id
from the rn_cte.
We are grouping on the basis of user id and DATEADD function because we want consecutive days to be grouped together.
Now we are also filtering so that our consecutive day is more than or equal to 3 days.

BY MANISH KUMAR CHAUDHARY

THANK YOU

“The present is life – all else is either memory or imagination.”

BY MANISH KUMAR CHAUDHARY

Interview Question

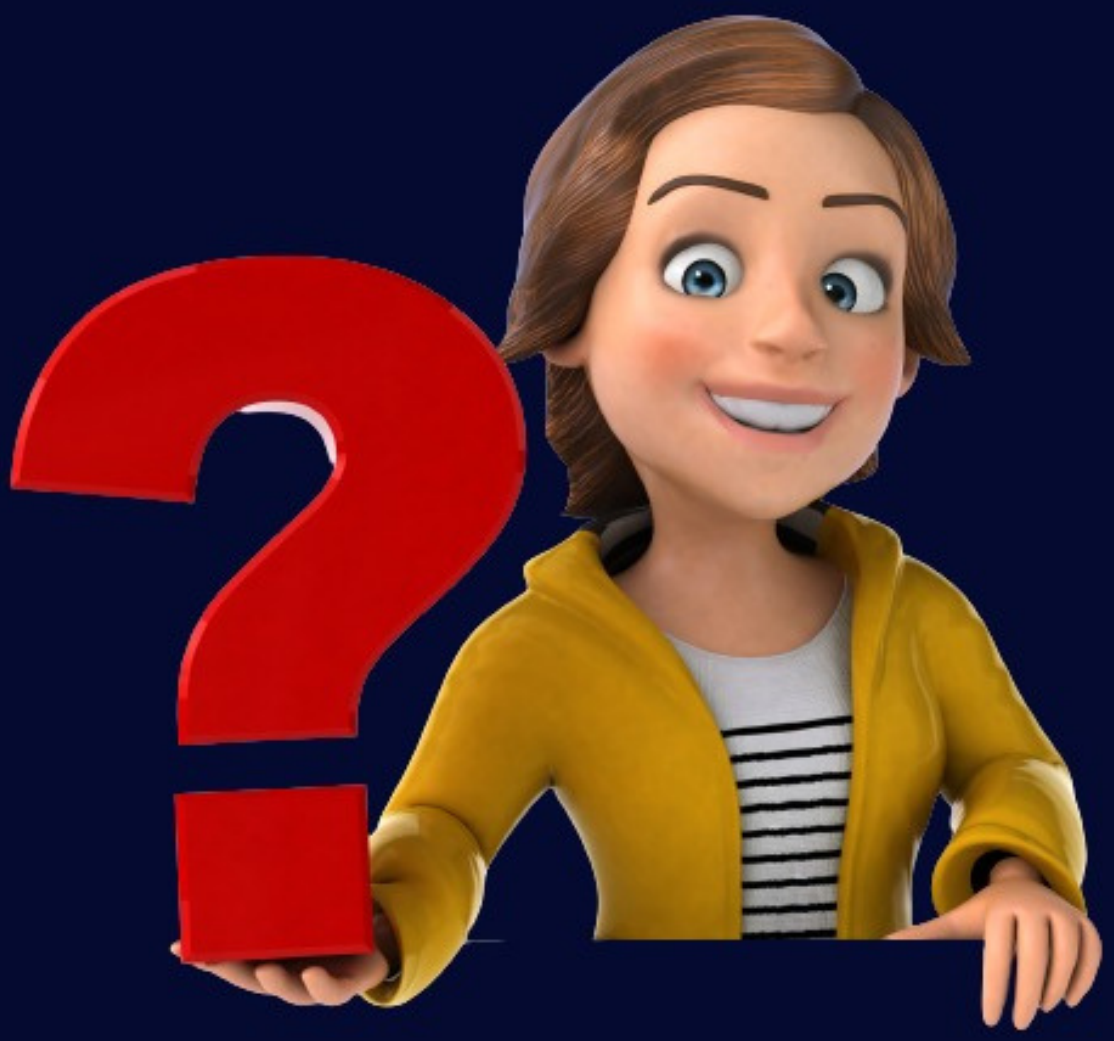
SQL

Counting Instances in Text

DIFFICULTY LEVEL :HARD

Question From

stratascratch



PROBLEM STATEMENT

Find the number of times the words 'bull' and 'bear' occur in the contents. We're counting the number of times the words occur so words like 'bullish' should not be included in our count. Output the word 'bull' and 'bear' along with the corresponding number of occurrences.

google_file_store

FIELD	TYPE
filename	varchar
contents	varchar

google_file_store Input Table

filename	contents
draft1.txt	The stock exchange predicts a bull market which would make many investors happy.
draft2.txt	The stock exchange predicts a bull market which would make many investors happy, but analysts warn of possibility of too much optimism and that in fact we are awaiting a bear market.
final.txt	The stock exchange predicts a bull market which would make many investors happy, but analysts warn of possibility of too much optimism and that in fact we are awaiting a bear market. As always predicting the future market is an uncertain game and all investors should follow their instincts and best practices.

QUERY

```sql
WITH words_cte AS (
SELECT REPLACE(VALUE,'.','') word
FROM google_file_store
CROSS APPLY STRING_SPLIT (contents,' ')
)
SELECT word
      ,COUNT(1) no_of_occurrences
FROM words_cte
WHERE word IN ('bull','bear')
GROUP BY word
```

QUERY EXPLANATION

1.With words_cte CTE we are spliting the text into single words using STRING_SPLIT function.
Here, we are simply printing the VALUES i.e words generated by spliting with STRING _SPLIT function.
Here, we have used REPLACE function so that we can replace '.' from our words as in where condition we are giving 'bull' and 'bear'. Chances are that there may occur words like 'bull.' OR 'bear.' while spliting them.

2.Now we are printing the required words and counting the number of occurrences by that particular.
Here we have filter with IN keyword as we want exact word 'bull' and 'bear' otherwise we would have gone with LIKE operator.

OUTPUT

word	no_of_occurrences
bear	2
bull	3

BY MANISH KUMAR CHAUDHARY

THANK YOU

Every new beginning comes from some other beginning's end.

BY MANISH KUMAR CHAUDHARY

Interview Question

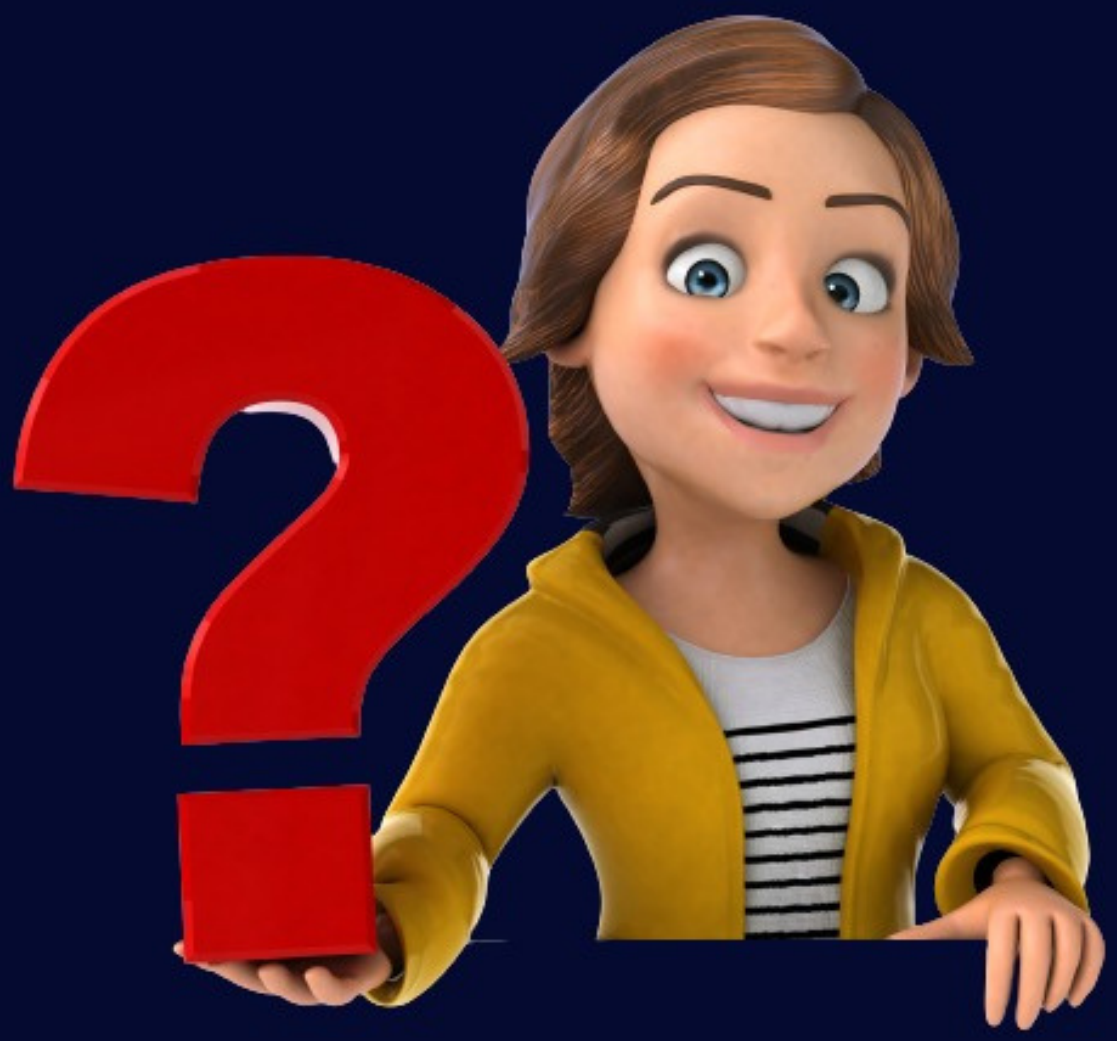
SQL

Customer Consumable Sales Percentages

DIFFICULTY LEVEL :MEDIUM

Question From

stratascratch



PROBLEM STATEMENT

Following a recent advertising campaign, you have been asked to compare the sales of consumable products across all brands.

Compare the brands by finding the percentage of unique customers who purchased consumable products from each brand.

Your output should contain the brand_name and percentage_of_customers rounded to the nearest whole number and ordered in descending order.

online_orders

FIELD	TYPE
product_id	int
promotion_id	int
cost_in_dollars	int
customer_id	int
date	datetime
unit_solds	int

online_products

FIELD	TYPE
product_id	int
product_class	varchar
brand_name	varchar
is_low_fat	varchar
is_recycle	varchar
product_category	int
product_family	varchar

online_orders Sample Input Table

product_id	promotion_id	cost_in_dollars	customer_id	date	units_sold
1	1	2	1	2022-04-01	4
3	3	6	3	2022-05-24	6
1	2	2	10	2022-05-01	3
1	2	3	2	2022-05-01	9
2	2	10	2	2022-05-01	1

online_products Sample Input Table

product_id	product_class	brand_name	is_low_fat	is_recyclable	product_category	product_family
1	ACCESSORIES	Fort West	N	N	3	GADGET
2	DRINK	Fort West	N	Y	2	CONSUMABLE
3	FOOD	Fort West	Y	N	1	CONSUMABLE
4	DRINK	Golden	Y	Y	3	CONSUMABLE
5	FOOD	Golden	Y	N	2	CONSUMABLE
6	FOOD	Lucky Joe	N	Y	3	CONSUMABLE
7	ELECTRONICS	Lucky Joe	N	Y	2	GADGET

QUERY

```sql
select p.brand_name
      ,COUNT(DISTINCT o.customer_id)*100/COUNT(*) perc
from online_orders o
INNER JOIN online_products p
ON o.product_id=p.product_id
WHERE p.product_family='CONSUMABLE'
GROUP BY p.brand_name
ORDER BY perc DESC;
```

QUERY EXPLANATION

1.We are counting the distinct customer for each brand and then dividing it by total customers they have for each brand to get the percentage of unique customers.

Here, we are also filtering to get the Consumable products only.

BY MANISH KUMAR CHAUDHARY

THANK YOU

“When you have a dream, you’ve got to grab it and never let go.”

Carol Burnett

BY MANISH KUMAR CHAUDHARY

Interview Question

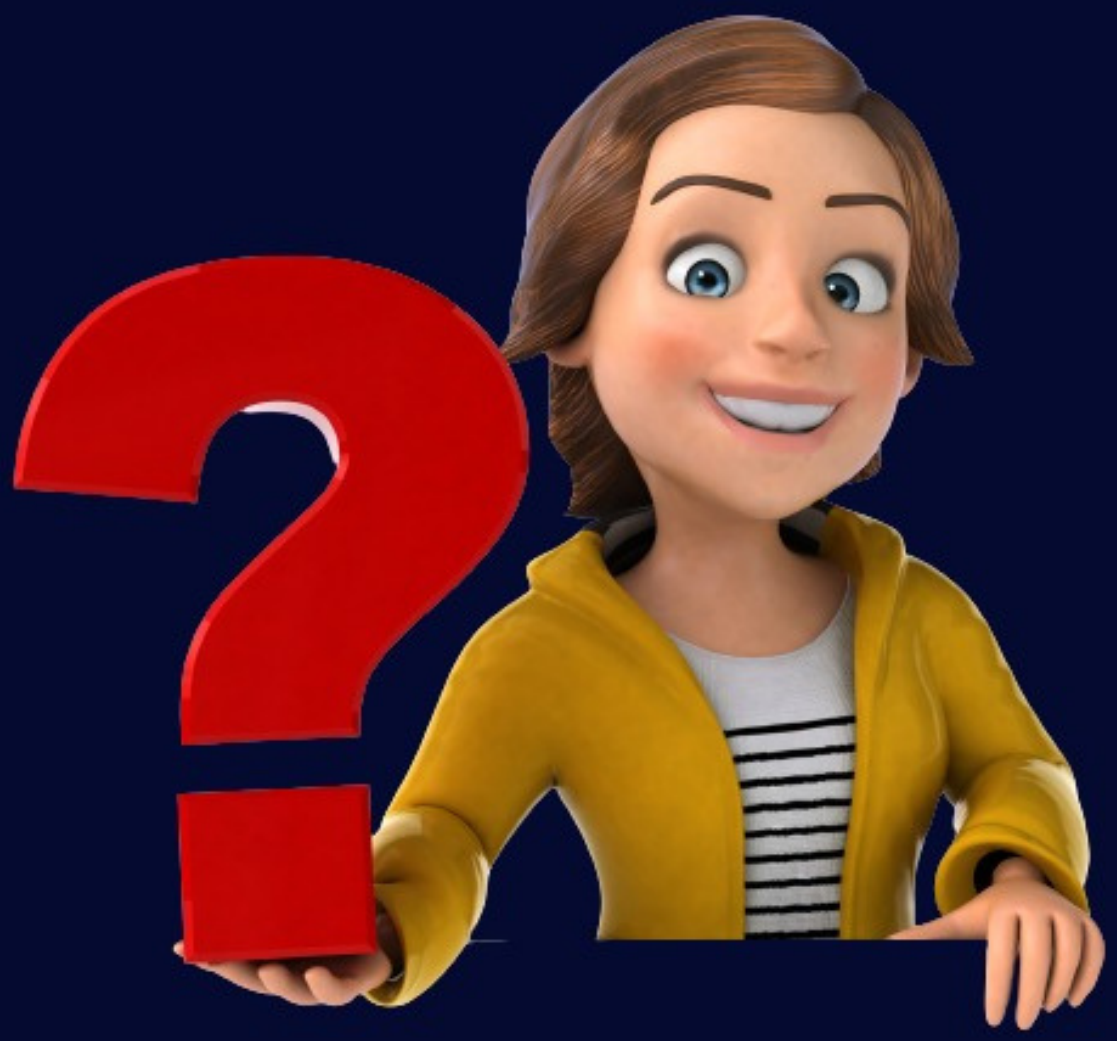
SQL

Customer Tracking

DIFFICULTY LEVEL :HARD

Question From

stratascratch



PROBLEM STATEMENT

Given the users' sessions logs on a particular day, calculate how many hours each user was active that day.

Note: The session starts when state=1 and ends when state=0

cust_tracking

FIELD	TYPE
cust_id	varchar
state	int
timestamp	datetime

cust_tracking Half Input Table

cust_id	state	timestamp
c001	1	07:00:00
c001	0	09:30:00
c001	1	12:00:00
c001	0	14:30:00
c002	1	08:00:00
c002	0	09:30:00
c002	1	11:00:00
c002	0	12:30:00
c002	1	15:00:00
c002	0	16:30:00

cust_tracking Other Half Input Table

c003	1	09:00:00
c003	0	10:30:00
c004	1	10:00:00
c004	0	10:30:00
c004	1	14:00:00
c004	0	15:30:00
c005	1	10:00:00
c005	0	14:30:00
c005	1	15:30:00
c005	0	18:30:00

MENTAL APPROACH

1.First we need to get start and end time for each session for each individual customer.
Like customer c001 has two sessions one starting at 7:00 and ending at 9:30 and another starting at 12:00 and ending at 2:30

2.So we will now simply find the hours they spent by finding difference between end and start time and SUM them up if one user has multiple sessions.

QUERY

```sql
WITH separate_time_cte AS (
SELECT cust_id
      ,CASE WHEN state=1 THEN timestamp END start_time
      ,CASE WHEN state=0 THEN timestamp END end_time
      ,RANK() OVER (PARTITION BY cust_id ORDER BY timestamp) rnk
FROM cust_tracking
),
start_end_cte AS (
SELECT cust_id
      ,start_time
      ,LEAD(end_time) OVER (PARTITION BY cust_id ORDER BY rnk) end_time
FROM separate_time_cte
)
SELECT cust_id
      ,DATEDIFF(SECOND,start_time,end_time)/3600.0 active_hours
FROM start_end_cte
WHERE DATEDIFF(SECOND,start_time,end_time)/3600.0  IS NOT NULL
```

QUERY EXPLANATION

1.With separate_time_cte we are separating the start and end time for all users.
Here, we have also used RANK() to rank the records on the basis of time and for each customers.

2.With start_end_cte we are geting customer id, start_time and end_time.
Here to get the end_time to be on corresponding row of start_time we are using LEAD(end_time) and paritiong on the basis of customer and ORDER on the basis of rank.

3.Now we are simply query for customer id and get the active hours.
To get active hours we are using DATEDIFF() function to get the difference in seconds and then dividing it by 60*60 to get it in hours.

SAMPLE OUTPUT for separate_time_cte

cust_id	start_time	end_time	rnk
c001	07:00:00.0000000	NULL	1
c001	NULL	09:30:00.0000000	2
c001	12:00:00.0000000	NULL	3
c001	NULL	14:30:00.0000000	4
c002	08:00:00.0000000	NULL	1
c002	NULL	09:30:00.0000000	2
c002	11:00:00.0000000	NULL	3
c002	NULL	12:30:00.0000000	4
c002	15:00:00.0000000	NULL	5
c002	NULL	16:30:00.0000000	6

SAMPLE OUTPUT FOR start_end_cte

cust_id	start_time	end_time
c001	07:00:00.0000000	09:30:00.0000000
c001	NULL	NULL
c001	12:00:00.0000000	14:30:00.0000000
c001	NULL	NULL
c002	08:00:00.0000000	09:30:00.0000000
c002	NULL	NULL
c002	11:00:00.0000000	12:30:00.0000000
c002	NULL	NULL
c002	15:00:00.0000000	16:30:00.0000000
c002	NULL	NULL
c003	09:00:00.0000000	10:30:00.0000000
c003	NULL	NULL
c004	10:00:00.0000000	10:30:00.0000000

FINAL OUTPUT

cust_id	active_hours
c001	5.000000
c002	4.500000
c003	1.500000
c004	2.000000
c005	7.500000

BY MANISH KUMAR CHAUDHARY

THANK YOU

To be beautiful means to be yourself. You don’t need to be accepted by others. You need to accept yourself.”

Thich-Nhat Hanh

BY MANISH KUMAR CHAUDHARY

Interview Question

SQL

Data Analyst Case Study by A Major Travel Company

DIFFICULTY LEVEL :MEDIUM

Question From

Ankit Bansal

Yotube Channel

PROBLEM STATEMENT

Data Analyst Case Study by A Major Travel Company

There are two tables one booking_table and users_table.
We are requested to solve the following 4 questions.

1.Write a query to find the total number of users for each segment and total number of users who booked flight in April 2022.

2.Write a query to identify users whose first booking was a hotel booking.

3.Write a query to calculate the days between first and last booking of each user.

4.Write a query to count the number of flight and hotel bookings in each user segments for the year 2022

Sample booking_table Input Table

Booking_id	Booking_date	User_id	Line_of_business
b1	2022-03-23	u1	Flight
b2	2022-03-27	u2	Flight
b3	2022-03-28	u1	Hotel
b4	2022-03-31	u4	Flight
b5	2022-04-02	u1	Hotel
b6	2022-04-02	u2	Flight
b7	2022-04-06	u5	Flight
b8	2022-04-06	u6	Hotel
b9	2022-04-06	u2	Flight
b10	2022-04-10	u1	Flight
b11	2022-04-12	u4	Flight
b12	2022-04-16	u1	Flight
b13	2022-04-19	u2	Flight

users_table Input Table

User_id	Segment
u1	s1
u2	s1
u3	s1
u4	s2
u5	s2
u6	s3
u7	s3
u8	s3
u9	s3
u10	s3

1.Write a query to find the total number of users for each segment and total number of users who booked flight in April 2022.

QUERY

```sql
SELECT u.segment
      ,COUNT(DISTINCT u.User_id) total_users
      ,COUNT(DISTINCT CASE WHEN Line_of_business='Flight'
                           AND FORMAT(booking_date,'MM-yyyy')= '04-2022'
                           THEN u.user_id
            END) total_flights_booked
FROM user_table u
LEFT JOIN booking_table b
ON b.User_id=u.User_id
GROUP BY u.segment
```

QUERY EXPLANATION

1.To calculate the number of users from each segment we are simply using COUNT with Distinct user_id.

2.To get the number of users who booked flight in April 2022 we are using CASE WHEN statment.
Here, to get date as April 2022 we have made use of FORMAT function.

We have made use of LEFT join so that we can count all the users from each segment. If we hadn't used LEFT join then there might have been chances that all users have booked hotel or flight.

OUTPUT

segment	total_users	total_flights_booked
s1	3	2
s2	2	2
s3	5	1

2.Write a query to identify users whose first booking was a hotel booking.

QUERY

```sql
WITH rnk_cte AS (
SELECT booking_date
      ,user_id
      ,Line_of_business
      ,DENSE_RANK() OVER (PARTITION BY user_id
                          ORDER BY booking_date) rnk
FROM booking_table
)
SELECT user_id
FROM rnk_cte
WHERE rnk=1
AND Line_of_business='Hotel'
```

QUERY EXPLANATION

1.We are using rnk_cte to get the required details and rank on the basis of Order date for each user. This will help us to get the first booking done by that particular user.

2.We simply querying the required records and filtering with WHERE condition so that we can get records with rnk=1 and its business was 'Hotel'

OUTPUT

user_id
u6

3.Write a query to calculate the days between first and last booking of each user.

QUERY

```sql
SELECT user_id
      ,DATEDIFF(DAY,MIN(booking_date),MAX(booking_date)) diff_days
FROM booking_table
GROUP BY user_id
```

QUERY EXAPLANATION

1.We are using DATEDIFF to calculate the day difference between first booking date and last booking date for each user_id.

Here, we have used MIN() for getting the first booking date and MAX() for getting last booking date.

OUTPUT

user_id	diff
u1	44
u2	32
u4	34
u5	14
u6	16

4.Write a query to count the number of flight and hotel bookings in each user segments for the year 2022

QUERY

```sql
SELECT u.segment
      ,SUM(CASE WHEN b.Line_of_business='Flight' THEN 1 END)
       AS no_of_flight_bookings
      ,SUM(CASE WHEN b.Line_of_business='Hotel' THEN 1 END)
       AS no_of_hotel_bookings
FROM user_table u
INNER JOIN booking_table b
ON u.User_id=b.User_id
WHERE DATEPART(YEAR,b.booking_date)=2022
GROUP BY u.segment
```

QUERY EXPLANATION

1.By using CASE WHEN statement we are flagging with 1 if booking was for 'Flight' or 'Hotel' and then by using SUM function we are adding these 1s to count them.
To filter for year 2022 only we have used DATEPART Function.

OUTPUT

segment	np_of_flight_bookings	no_of_hotel_bookings
s1	8	4
s2	3	3
s3	1	1

Questions were good and I was able to solve them. All concepts for these questions were already covered in Ankit Bansal's interview series. Thus, I was exactly able to solve similar way Ankit Sir did.

BY MANISH KUMAR CHAUDHARY

THANK YOU

Don't judge each day by the harvest you reap but by the seeds that you plant.

Robert Louis Stevenson

BY MANISH KUMAR CHAUDHARY

Interview Question

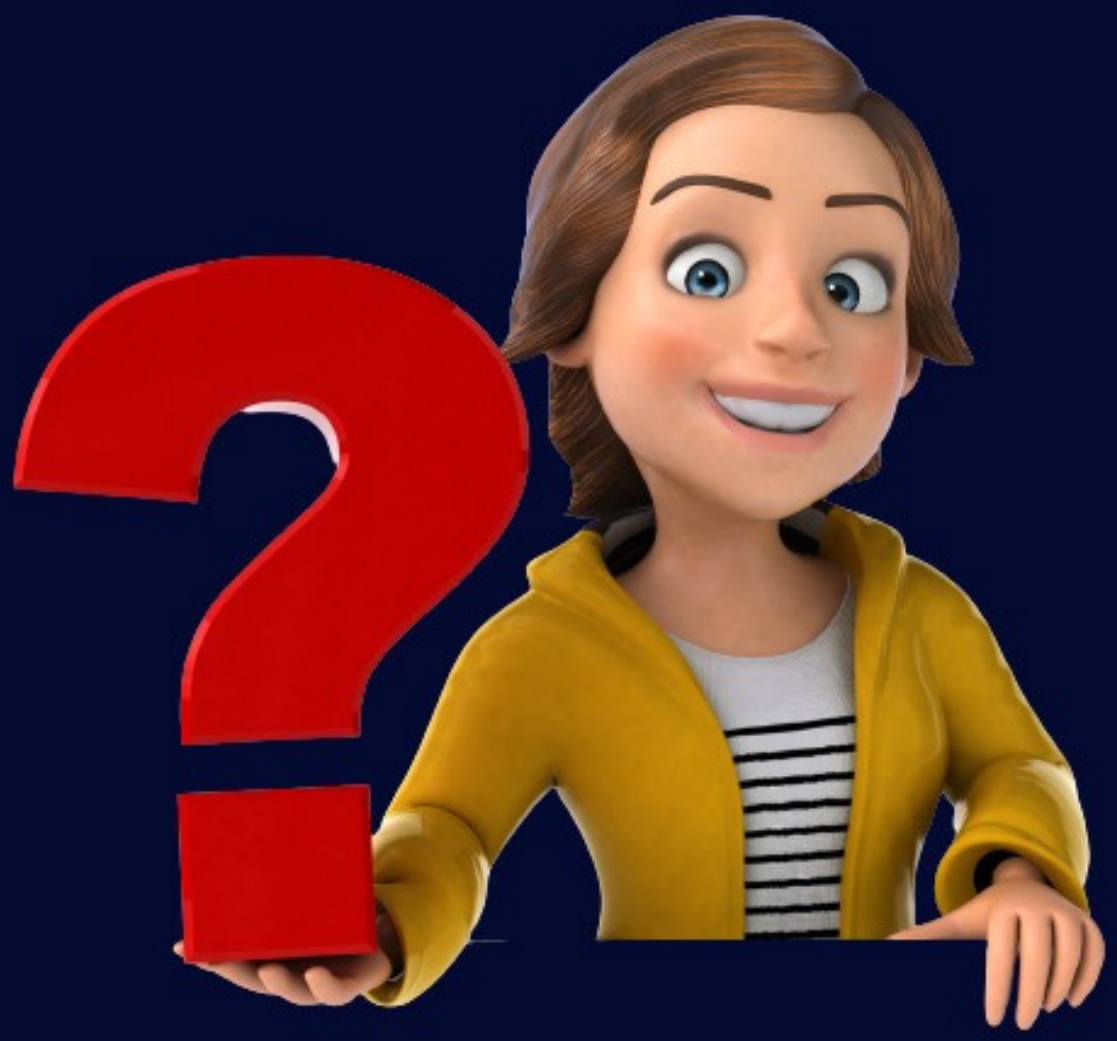
SQL

Date of Highest User Activity

DIFFICULTY LEVEL :MEDIUM

Question From

stratascratch



PROBLEM STATEMENT

Tiktok want to find out what were the top two most active user days during an advertising campaign they ran in the first week of August 2022 (between the 1st to the 7th).

Identify the two days with the highest user activity during the advertising campaign.
They've also specified that user activity must be measured in terms of unique users.
Output the day, date, and number of users.

user_streaks

FIELD	TYPE
user_id	varchar
date_visited	datetime

user_streaks Sample Input Table

user_id	date_visited
u001	2022-08-01
u001	2022-08-01
u004	2022-08-01
u005	2022-08-01
u005	2022-08-01
u003	2022-08-02
u004	2022-08-02
u004	2022-08-02
u004	2022-08-02

MENTAL APPROACH

1.For each day from 1st August to 7th August we will count the number of unique customer visited on that paritcular day.
2.We will pick two days on which their maximum count of unique users.

QUERY

```sql
SELECT TOP 2 DAY(date_visited) day
       ,date_visited
       ,COUNT(DISTINCT user_id) no_of_users
FROM user_streaks
WHERE date_visited BETWEEN '2022-08-01' AND '2022-08-07'
GROUP BY DAY(date_visited),date_visited
ORDER BY no_of_users DESC
```

QUERY EXPLANATION

1.We are extracting day from date_visited and for each day we are getting the count of unique users.

2.We have filtered in WHERE condition so that we get result between 1st and 7th of August.

I have solved using TOP function. Come up with your solution but other approaches.

OUTPUT

day	date_visited	no_of_users
7	2022-08-07	5
3	2022-08-03	4

BY MANISH KUMAR CHAUDHARY

THANK YOU

" Live life as if everything is rigged in your favor."

BY MANISH KUMAR CHAUDHARY

Interview Question

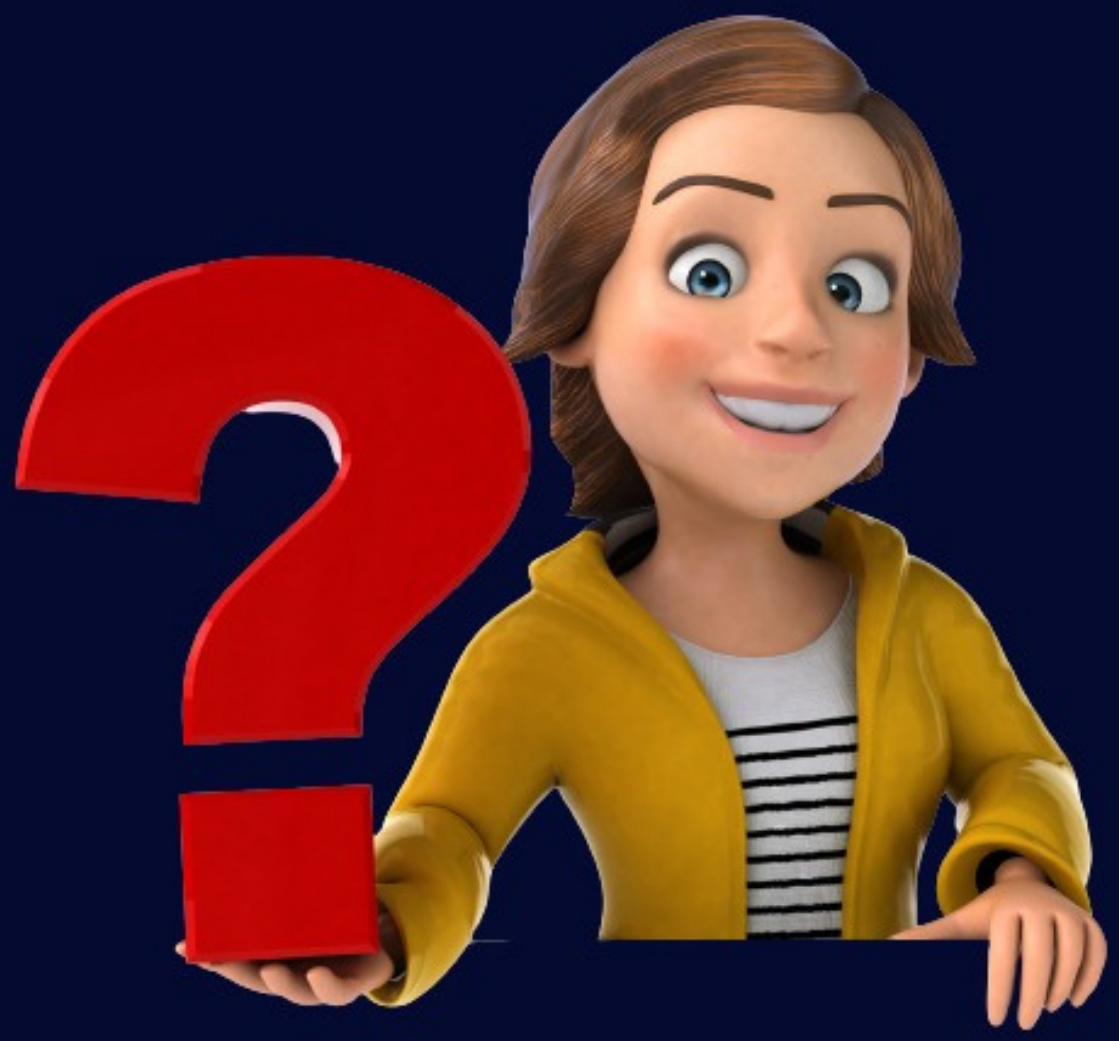
SQL

Dates Of Inspection

DIFFICULTY LEVEL :HARD

Question From

stratascratch



PROBLEM STATEMENT

Find the latest inspection date for the most sanitary restaurant(s). Assume the most sanitary restaurant is the one with the highest number of points received in any inspection (not just the last one). Only businesses with 'restaurant' in the name should be considered in your analysis.

Output the corresponding facility name, inspection score, latest inspection date, previous inspection date, and the difference between the latest and previous inspection dates. And order the records based on the latest inspection date in ascending order.

los_angeles_restaurant_health_inspectin

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

QUERY

```sql
SELECT r1.facility_name
      ,r1.score
      ,MAX(r1.activity_date) latest_inspection_date
      ,MAX(r2.activity_date) previous_inspection_date
      ,DATEDIFF(DAY,MAX(r2.activity_date),MAX(r1.activity_date)) difference
FROM los_angeles_restaurant_health_inspections r1
LEFT JOIN los_angeles_restaurant_health_inspections r2
ON r1.facility_name=r2.facility_name
AND r1.activity_date>r2.activity_date
WHERE r1.facility_name LIKE '%restaurant%'
AND r1.score IN (SELECT MAX(score)
                 FROM los_angeles_restaurant_health_inspections)
GROUP BY r1.facility_name, r1.score
ORDER BY latest_inspection_date
```

QUERY EXPLANATION

Assumption: Have not considered program_status being active and inactive as I was not able to figure what it is supposed to be.

1.Here we are using SELECT query to get the facility_name, score and latest_inspection_date and previous_inspection date.
To get the previous inspection date we have used LEFT SELF JOIN on the basis of facility_name and then filtered with condition that activity_date of first table is >= acitvity_date of second_table. And after this we are selecting the MAX of activity_date from second table to get the previous date.
It was possible to get this because when we filtered then for second table we will not get the actual maximum date but will get the activity_date which is second maximum.

NOTE: We used LEFT SELF JOIN because we also want those records which appear only for single activity_date

2.As we have used LEFT SELFT JOIN we will get those records only that appeared single time. And for these records we will not have previous inspection date and difference of dates as they are inspected only one time.

3.To get the records having Restaurant in facility name we have used this in WHERE condition using LIKE operator. Here we have also filtered using SUBQUERY to get those records only where it is having MAX score.

SAMPLE OUTPUT

facility_name	score	latest_inspection_date	previous_inspection_date	difference
D'LIDO RESTAURANT AND BAKERY	100	2016-02-03		
TEXIS RESTAURANT AND ENTERTAINMENT	100	2016-08-26		
LEONARDO'S RESTAURANT	100	2016-09-16	2016-03-04	196
LOS ARCOS RESTAURANT	100	2016-11-02	2016-05-23	163
LA FUENTE RESTAURANT	100	2016-11-03		
B.L. RESTAURANT	100	2016-12-01		
NENA RESTAURANT	100	2016-12-11		
CANCUN RESTAURANT	100	2017-02-01		

BY MANISH KUMAR CHAUDHARY

THANK YOU

"The only person you are destined to become is the person you decide to be."

Ralph Waldo Emerson

BY MANISH KUMAR CHAUDHARY

Interview Question

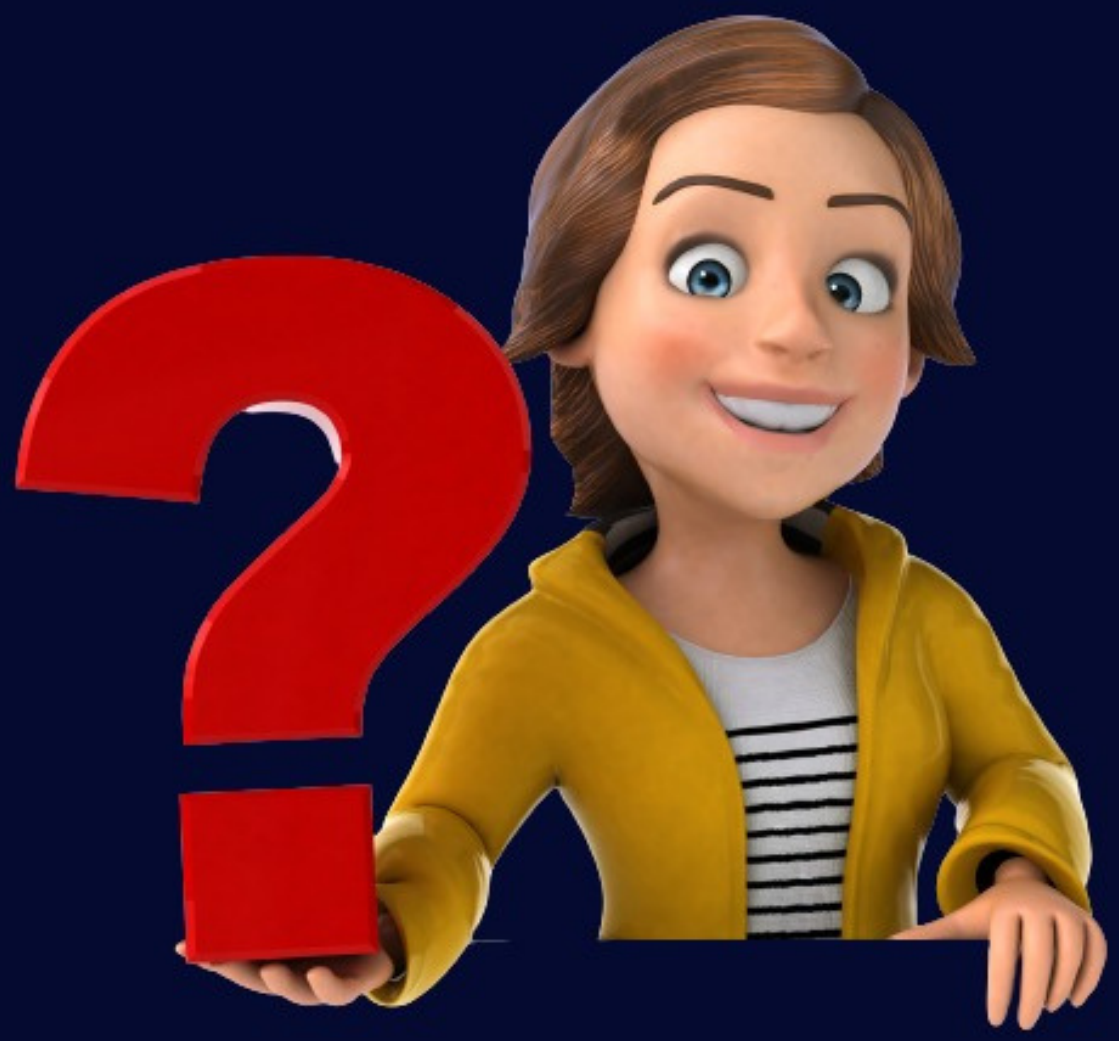
SQL

Distance Per Dollar

DIFFICULTY LEVEL :HARD

Question From

stratascratch



PROBLEM STATEMENT

You're given a dataset of uber rides with the traveling distance ('distance_to_travel') and cost ('monetary_cost') for each ride. For each date, find the difference between the distance-per-dollar for that date and the average distance-per-dollar for that year-month. Distance-per-dollar is defined as the distance traveled divided by the cost of the ride.

The output should include the year-month (YYYY-MM) and the absolute average difference in distance-per-dollar (Absolute value to be rounded to the 2nd decimal).
You should also count both success and failed request_status as the distance and cost values are populated for all ride requests. Also, assume that all dates are unique in the dataset. Order your results by earliest request date first.

uber_request_logs Input Table

request_id	request_date	request_status	distance_to_travel	monetary_cost	driver_to_client_distance
1	2020-01-09	success	70.59	6.56	14.36
2	2020-01-24	success	93.36	22.68	19.9
3	2020-02-08	fail	51.24	11.39	21.32
4	2020-02-23	success	61.58	8.04	44.26
5	2020-03-09	success	25.04	7.19	1.74
6	2020-03-24	fail	45.57	4.68	24.19
7	2020-04-08	success	24.45	12.69	15.91
8	2020-04-23	success	48.22	11.2	48.82
9	2020-05-08	success	56.63	4.04	16.08

QUERY

```sql
WITH distance_cte AS (
SELECT FORMAT(CAST(request_date AS DATE),'yyyy-MM') date
      ,request_status
      ,distance_to_travel/monetary_cost distance_per_dollar
      ,AVG(distance_to_travel/monetary_cost)
          OVER (PARTITION BY FORMAT(CAST(request_date AS DATE),'yyyy-MM'))
          avg_distance_per_dollar
FROM uber_request_logs
)
SELECT date
      ,ABS(distance_per_dollar - avg_distance_per_dollar) difference
      ,SUM(CASE WHEN request_status='success' THEN 1 ELSE 0 END)
count_success
      ,SUM(CASE WHEN request_status='fail' THEN 1 ELSE 0 END) count_fail
FROM distance_cte
GROUP BY date,ABS(distance_per_dollar - avg_distance_per_dollar)
ORDER BY date
```

QUERY EXPLANATION

I have made use of CTE to make query look simpler, it can be done in one single query as well.

1.Used cte for getting distance_per_dollar and avg_distance_per_dollar.
For avg we have used it as window function as we need to calculate difference and we have partition on the basis of date so that for each date we get the required records.

2.Simply SELECTING the required records along with the COUNT of success and fail requests.
Here, we have used CASE WHEN statement to flag the request status and summing them up as we have flagged with 1 and 0

SAMPLE OUTPUT TABLE

date	difference	count_sucess	count_fail
2020-01	3.322	1	0
2020-01	3.322	1	0
2020-02	1.58	0	1
2020-02	1.58	1	0
2020-03	3.127	1	0
2020-03	3.127	0	1
2020-04	1.189	1	0
2020-04	1.189	1	0
2020-05	6.437	1	1
2020-06	4.971	0	2
2020-07	2.024	1	1
2020-08	11.613	2	0

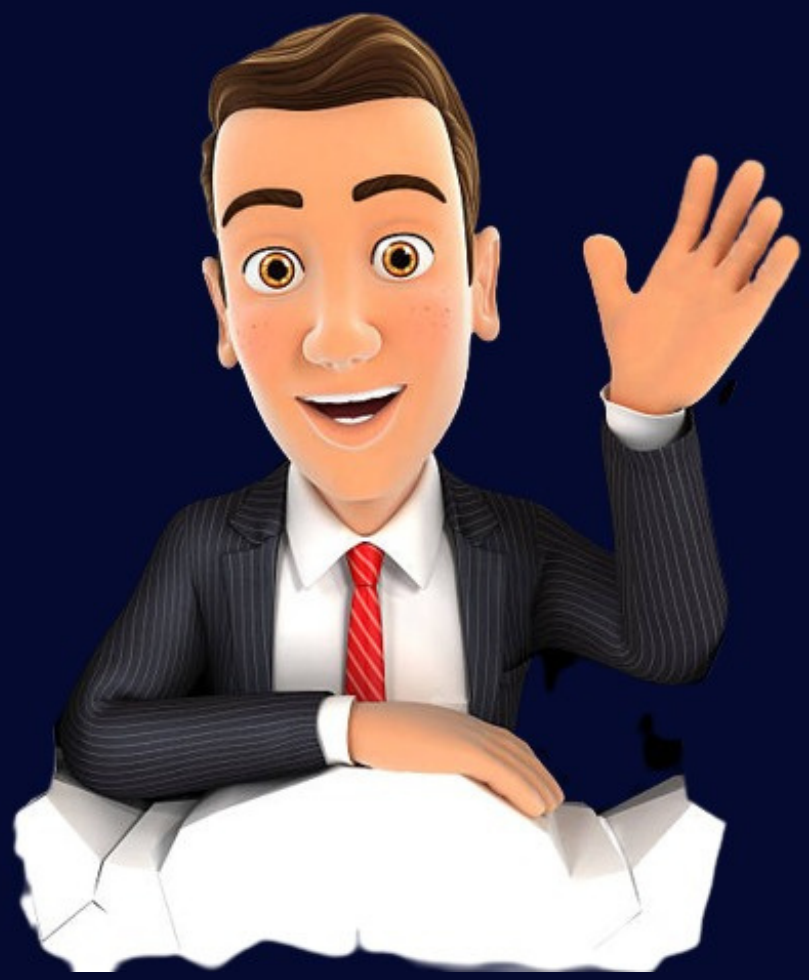
I was not able to validate this, and I think something is wrong. If you know correct answer then please let me know and also comment if it is correct.

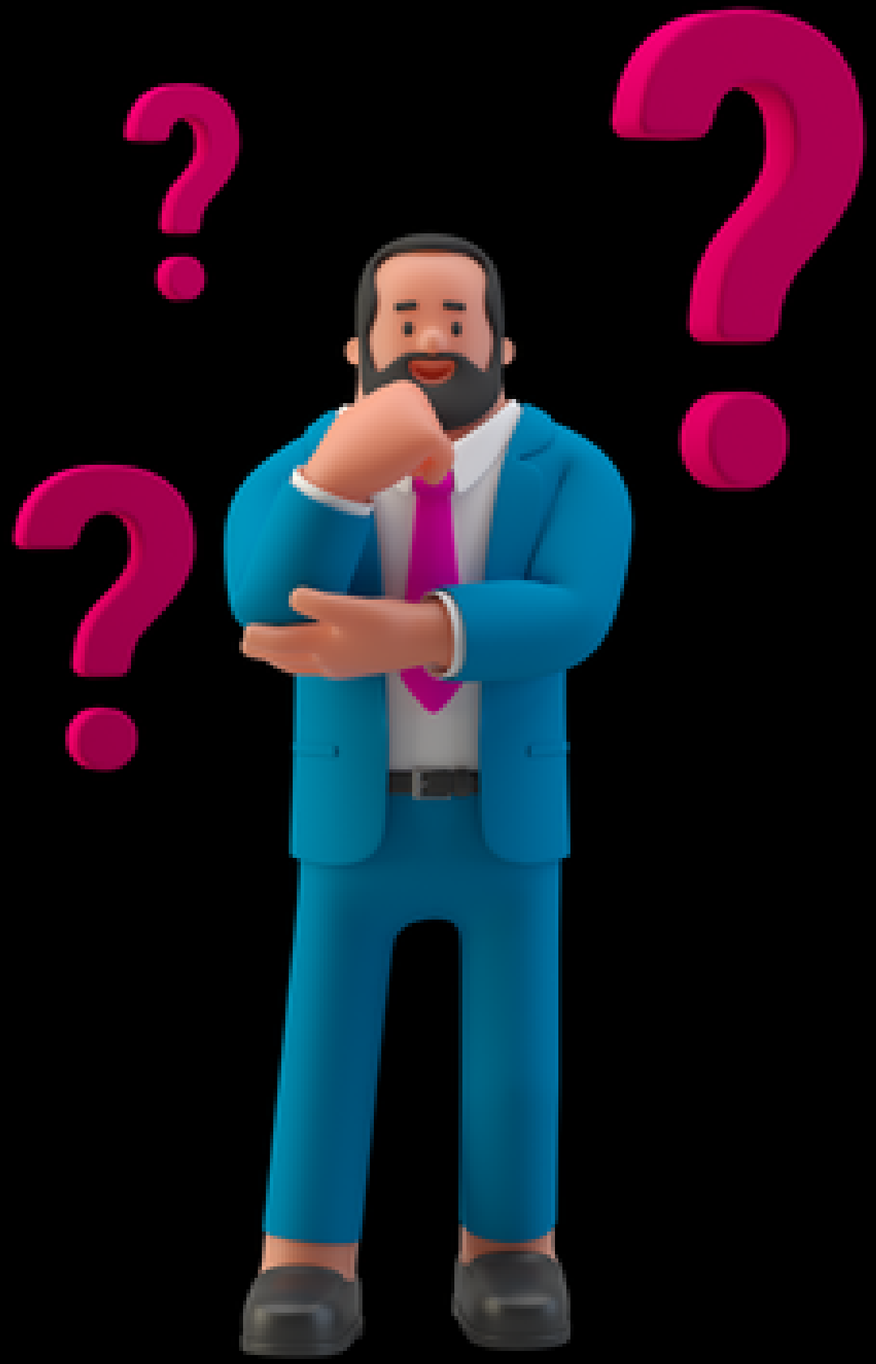
BY MANISH KUMAR CHAUDHARY

THANK YOU

"By changing nothing, nothing changes."

Tony Robbins





DO YOU KNOW?

Mathematical Functions

Interview Question

SQL

BY MANISH KUMAR CHAUDHARY

1. **ABS:** Returns the absolute (positive) value of a number.

agraph text

```
SELECT ABS(-10) -- Output: 10
```

2. **ROUND Function:** The ROUND function is used to return a number rounded to a specified number of decimal places.

```
Syntax: ROUND(n, d)
n - value
d - number of digits to round
```

```
SELECT ROUND(3.14159, 2) -- Output: 3.14
```

3. **CEILING Function :** The CEILING function is used to return the smallest integer greater than or equal to a given number.

```
SELECT CEILING(3.5) -- Output: 4
```

4. **FLOOR Function:** The FLOOR function is used to return the largest integer less than or equal to a given number.

```
SELECT FLOOR(3.5) -- Output: 3
```

5. **SQRT Function:** The SQRT function is used to return the square root of a given number.

```
SELECT SQRT(16) -- Output: 4
```

6. **POWER Function:** The POWER function is used to return the value of a number raised to a specified power.

```
Syntax: POWER(n, p)
n - numeric value
p - power value
```

```
SELECT POWER(2, 3) -- Output: 8
```

7. **LOG Function:** The LOG function is used to return the natural logarithm of a given number.

```
SELECT LOG(10) -- Output: 2.30258509299405
```

8. **EXP Function:** The EXP function is used to return the value of e raised to the power of a given number.

```
SELECT EXP(2) -- Output: 7.38905609893065
```

9. **PI Function :** The PI function returns the value of pi (3.14159265358979).

```
SELECT PI() -- Output: 3.14159265358979
```

10. **RAND Function:** The RAND function is used to return a random float value between 0 and 1.

```
SELECT RAND() -- Output: 0.587717128826336
```

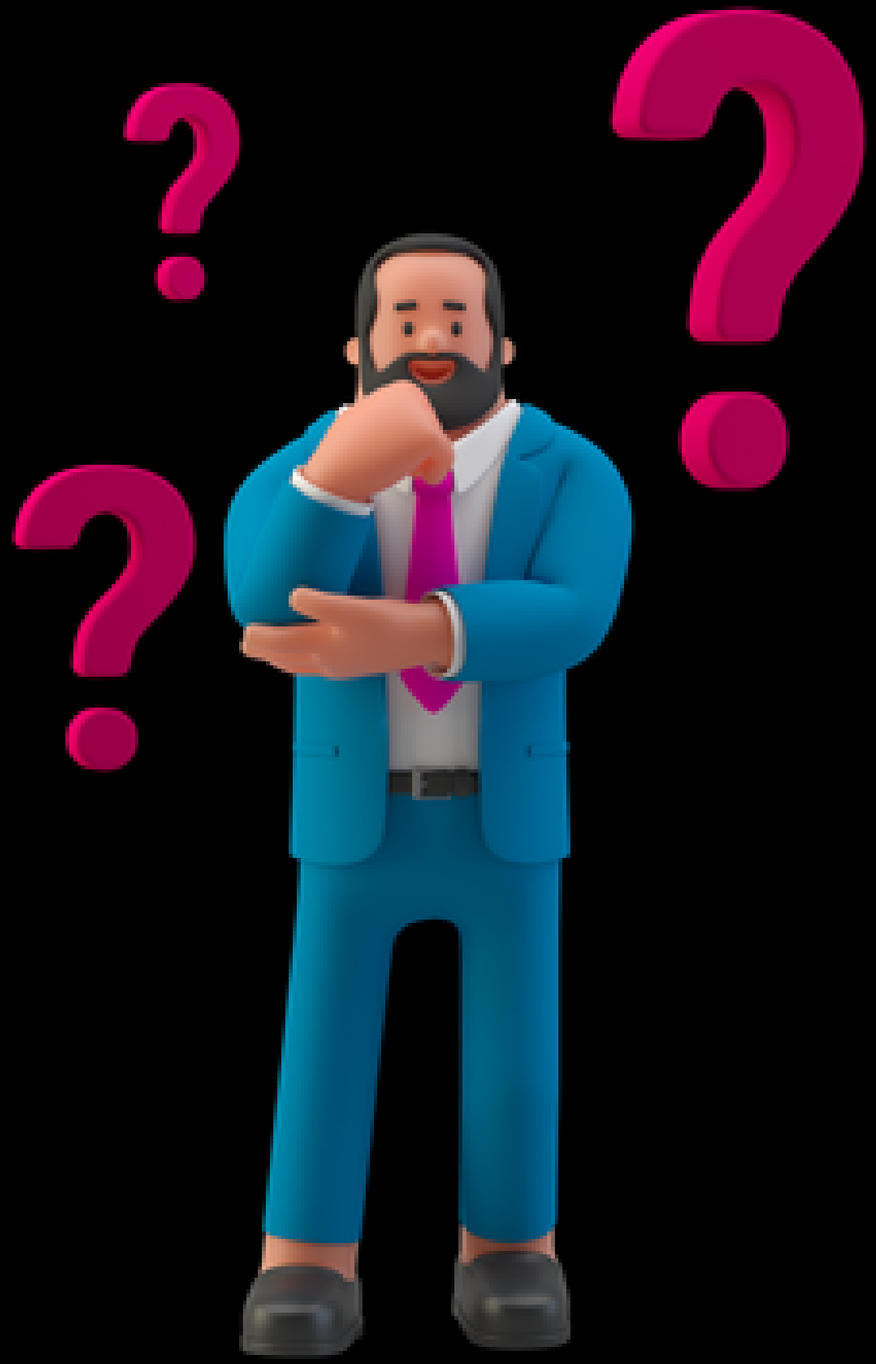
11. SIGN Function: The SIGN function is used to return the sign of a given number (1 for positive numbers, -1 for negative numbers, and 0 for zero).

```
SELECT SIGN(10) -- Output: 1
SELECT SIGN(-10) -- Output: -1
SELECT SIGN(0) -- Output: 0
```

THANK YOU

NO MATTER HOW EDUCATED, TALENTED, RICH, OR COOL YOU BELIEVE YOU ARE, HOW YOU TREAT PEOPLE ULTIMATELY TELLS ALL.

INTEGRITY IS EVERYTHING.



DO YOU KNOW?

Replacing Null Values

Interview Question

SQL

BY MANISH KUMAR CHAUDHARY

ISNULL

This function returns the specified value if the expression is NULL, otherwise, it returns the expression itself. For example:

```
SELECT ISNULL(column_name, 'N/A') AS column_alias
FROM table_name;
```

This will return 'N/A' if the value of column_name is NULL.

COALESCE

This function returns the first non-NULL expression from a list of expressions. For example:

```
SELECT COALESCE(column_name1, column_name2, 'N/A') AS column_alias
FROM table_name;
```

This will return the value of column_name1 if it is not NULL, otherwise the value of column_name2 if it is not NULL, otherwise 'N/A'.

NULLIF

This function returns NULL if the two expressions are equal, otherwise, it returns the first expression. For example:

```
SELECT NULLIF(column_name, '') AS column_alias
FROM table_name;
```

This will return NULL if the value of column_name is an empty string.

CASE Statement

This statement can be used to conditionally replace NULL values. For example:

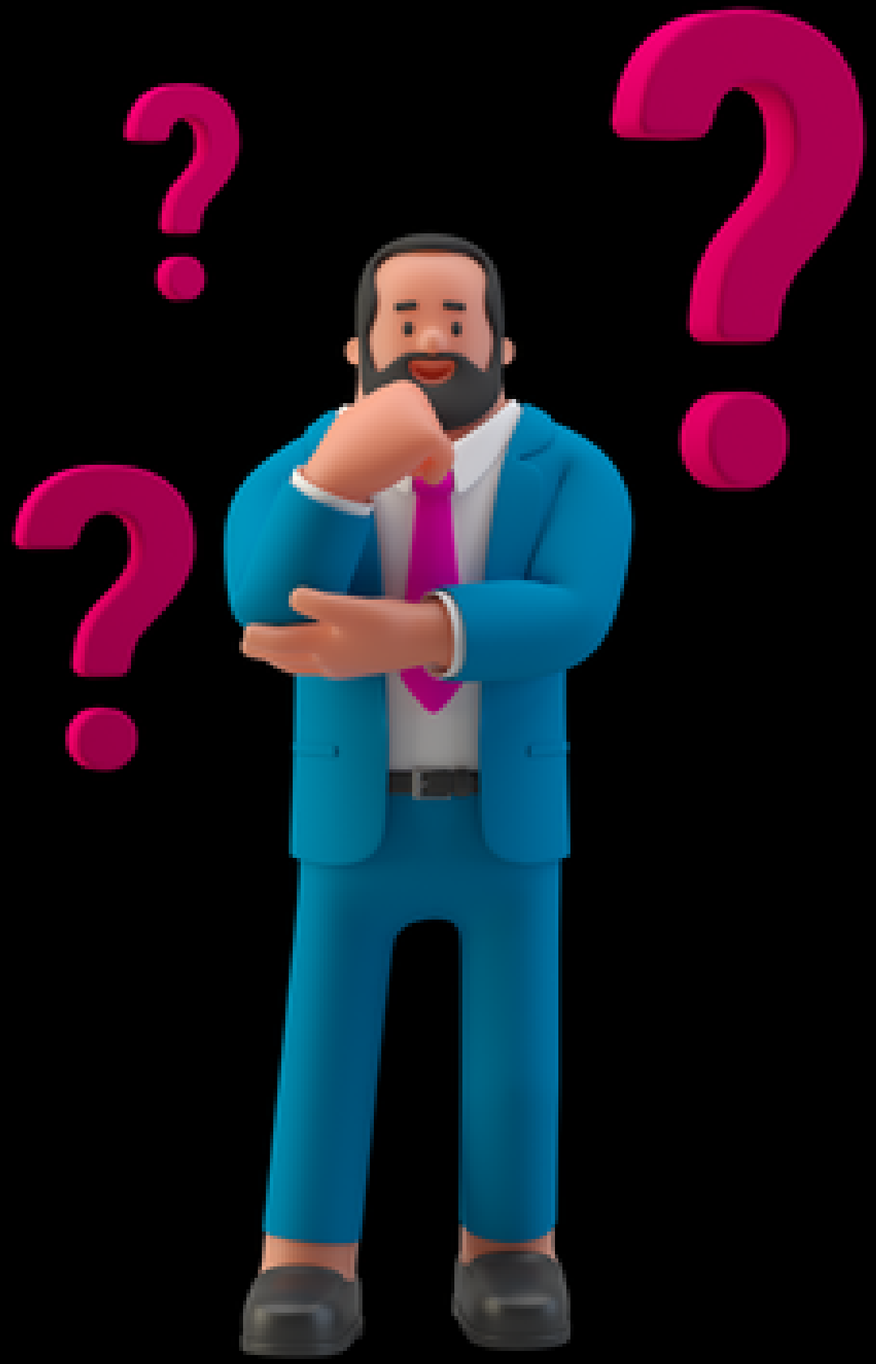
```sql
SELECT
  CASE
    WHEN column_name IS NULL THEN 'N/A'
    ELSE column_name
  END AS column_alias
FROM table_name;
```

This will return 'N/A' if the value of column_name is NULL.

THANK YOU

"The difference between ordinary and extraordinary is that little extra."

Jimmy Johnson



DO YOU KNOW?

Subquery in SQL

Interview Question

SQL

BY MANISH KUMAR CHAUDHARY

SUBQUERY

A subquery is a query that is nested inside a SELECT, INSERT, UPDATE, or DELETE statement, or inside another subquery.

The outer query is called as main query and inner query is called as subquery.

Subqueries can be used with many SQL statements, including SELECT, INSERT, UPDATE, and DELETE.

In a SELECT statement, a subquery can be used in the WHERE clause to filter the results based on the output of the subquery. It can also be used in the FROM clause to create a temporary table that is used to join with another table. It can also be used within the HAVING clause to filter the results based on the output of the subquery.

In an INSERT statement, a subquery can be used in the VALUES clause to insert the results of the subquery into a table.

In an UPDATE statement, a subquery can be used in the SET clause to update a column based on the output of the subquery.

In a DELETE statement, a subquery can be used in the WHERE clause to delete rows based on the output of the subquery.

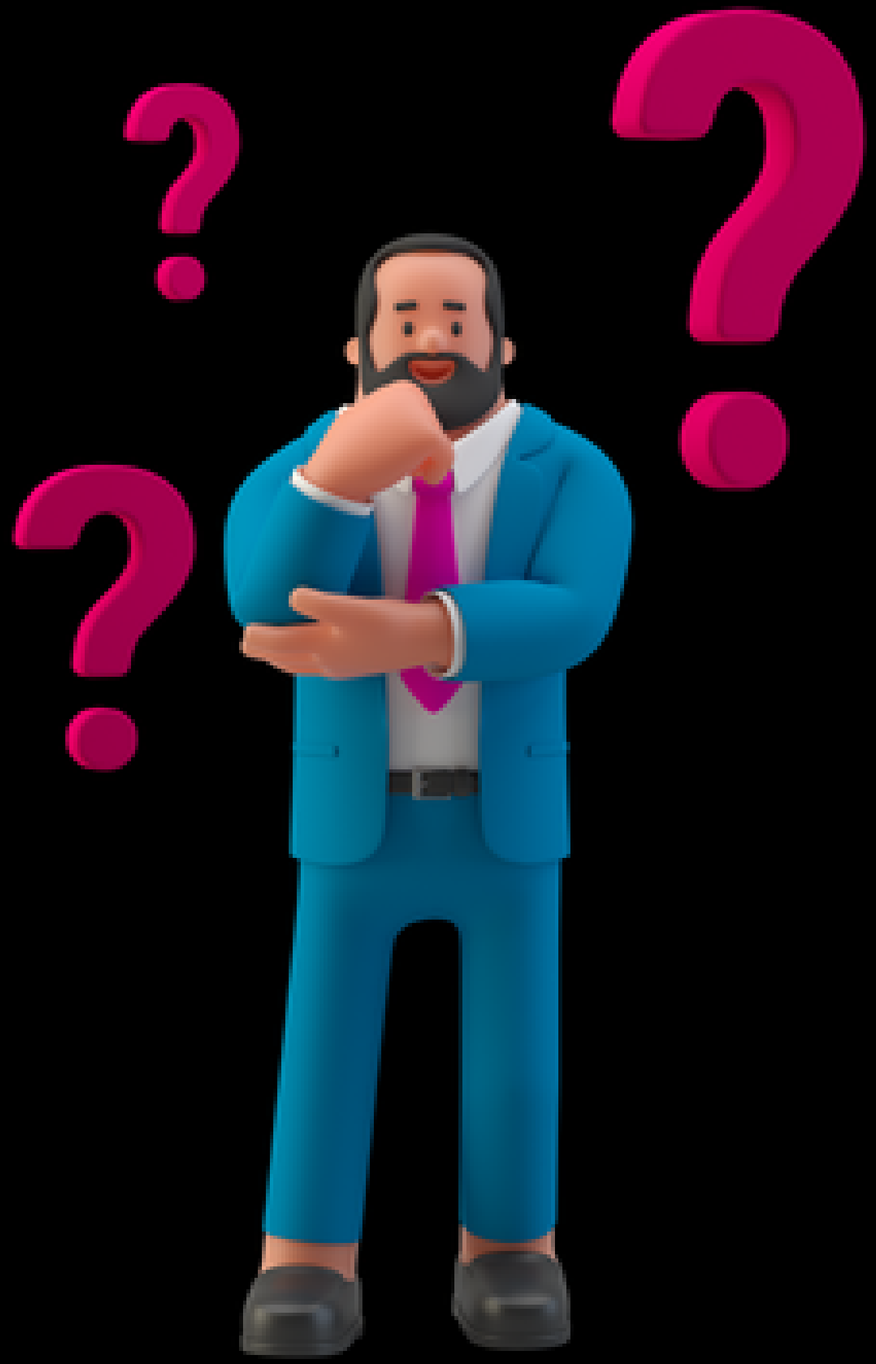
Important Point

The subquery generally executes first when the subquery doesn’t have any co-relation with the main query, when there is a co-relation the parser takes the decision on the fly on which query to execute on precedence and uses the output of the subquery accordingly.

THANK YOU

It is better to fail in originality than to succeed in imitation.

Herman Melville



DO YOU KNOW?

DATA MODELING

Interview Question

SQL

BY MANISH KUMAR CHAUDHARY

WHAT IS DATA MODELING?

Data Modeling is the process of developing a data model for storing data in a database. This data model is a conceptual representation of data objects, data object associations, and data object rules.

Steps to be followed while Data Modeling are:

1. Determine the requirements: Define the data model's purpose and objectives, as well as the important entities and features of the data.

2. Create the conceptual model: Make a high-level conceptual model of the essential entities and their relationships. This can be accomplished through the use of entity-relationship diagrams or other modelling tools.

3. Make a logical model: Provide a thorough logical model that outlines the data's qualities, relationships, and constraints. This can be accomplished with data modelling software such as ERwin, PowerDesigner, or Visio.

4. Normalize the data: To eliminate redundancy and increase data integrity, ensure that the data is normalised. This entails dividing tables into smaller, more manageable chunks and ensuring that each table has a primary key.

5. Make a physical model: Define the tables, columns, indexes, and other database objects to implement the logical model in a physical database. SQL Server Management Studio or other database management tools can be used to accomplish this.

6. Optimize the model: Tune the database structure, indexing method, and other performance-related parameters to optimise the database model. This can be accomplished using a variety of strategies, including query optimisation, index optimisation, and partitioning.

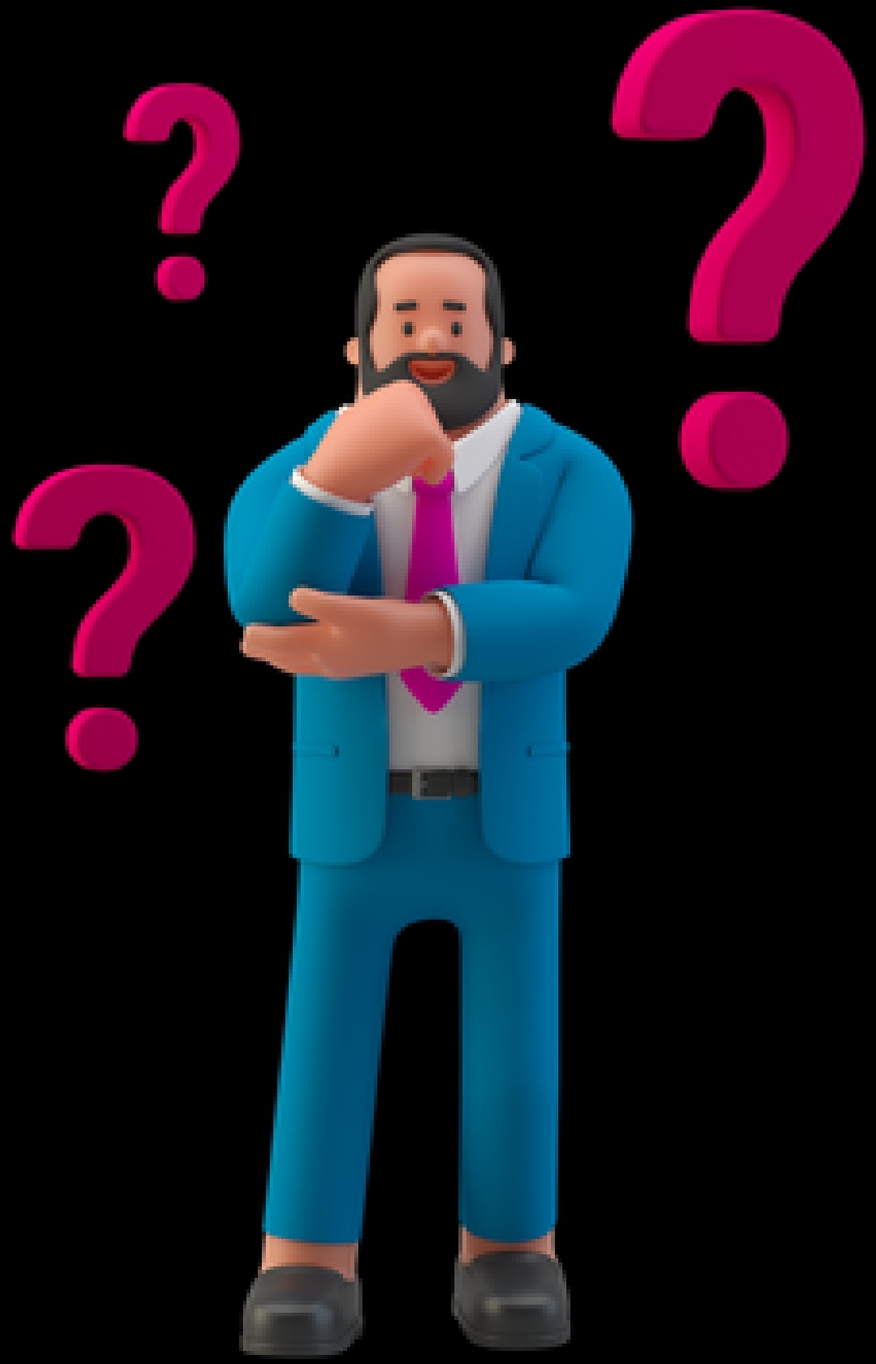
7. Validate and test the model: Validate the data model by running it through real-world scenarios and ensuring that it fulfils the project's needs and objectives.

8. Maintain the model: The data model should be updated as needed to reflect changes in the data or business requirements. This includes monitoring database performance and tracking schema changes.

THANK YOU

“SOMETIMES IT TAKES A GOOD FALL TO REALLY KNOW WHERE YOU STAND”

HAYLEY WILLIAMS



DO YOU KNOW?

Date Time Functions

Interview Question

SQL

BY MANISH KUMAR CHAUDHARY

1. **GETDATE() :** This function returns the current date and time according to the SQL Server instance where it is executed.

```
SELECT GETDATE();
```

```
2023-03-17 08:30:15.310
```

2. **DATEPART():** Returns a specific part of a datetime value, such as year, month, or day.

```
SELECT DATEPART(datepart, date);
```

```
SELECT DATEPART(year, '2022-06-15');
```

```
2022
```

There are many dateparts like year, month,week and many more you can check on google.

3. **DATEADD():** Adds a specified number of intervals (such as days, months, or years) to a datetime value.

```
SELECT DATEADD(interval, number, date);
```

```
SELECT DATEADD(month, 3, '2022-06-15');
```

2022-09-15

4. **DATEDIFF():** Returns the difference between two datetime values, in a specified interval (such as days, hours, or minutes).

```
SELECT DATEDIFF(interval, date1, date2);
```

```
SELECT DATEDIFF(day, '2022-06-15', '2022-07-15');
```

30

5. CONVERT(): Converts a datetime value from one format to another.

```
SELECT CONVERT(datatype, date, style);
```

```
SELECT CONVERT(varchar, '2022-06-15', 101);
```

06/15/2022

6. FORMAT(): Formats a datetime value as a string, according to a specified format.

```
SELECT FORMAT(date, format);
```

```
SELECT FORMAT('2022-06-15', 'MM/dd/yyyy');
```

06/15/2022

There are many more codes for style and different formats you check on google.

7. DATENAME(): Returns a character string representing the specified part of a datetime value.

```
SELECT DATENAME(datepart, date);
```

```
SELECT DATENAME(month, '2022-06-15');
```

June

8. MONTH(): Returns the month part of a datetime value.

```
SELECT MONTH(date);
```

```
SELECT MONTH('2022-06-15');
```

6

9. YEAR(): Returns the year part of a datetime value.

```
SELECT YEAR(date);
```

```
SELECT YEAR('2022-06-15');
```

2022

10. DAY(): Returns the day of the month from a specified datetime value.

```
SELECT DAY(date);
```

```
SELECT DAY('2022-06-15');
```

15

11. **ISDATE():** It checks whether a string expression is a valid date, time, or datetime value, and returns 1 if it is, and 0 if it is not.

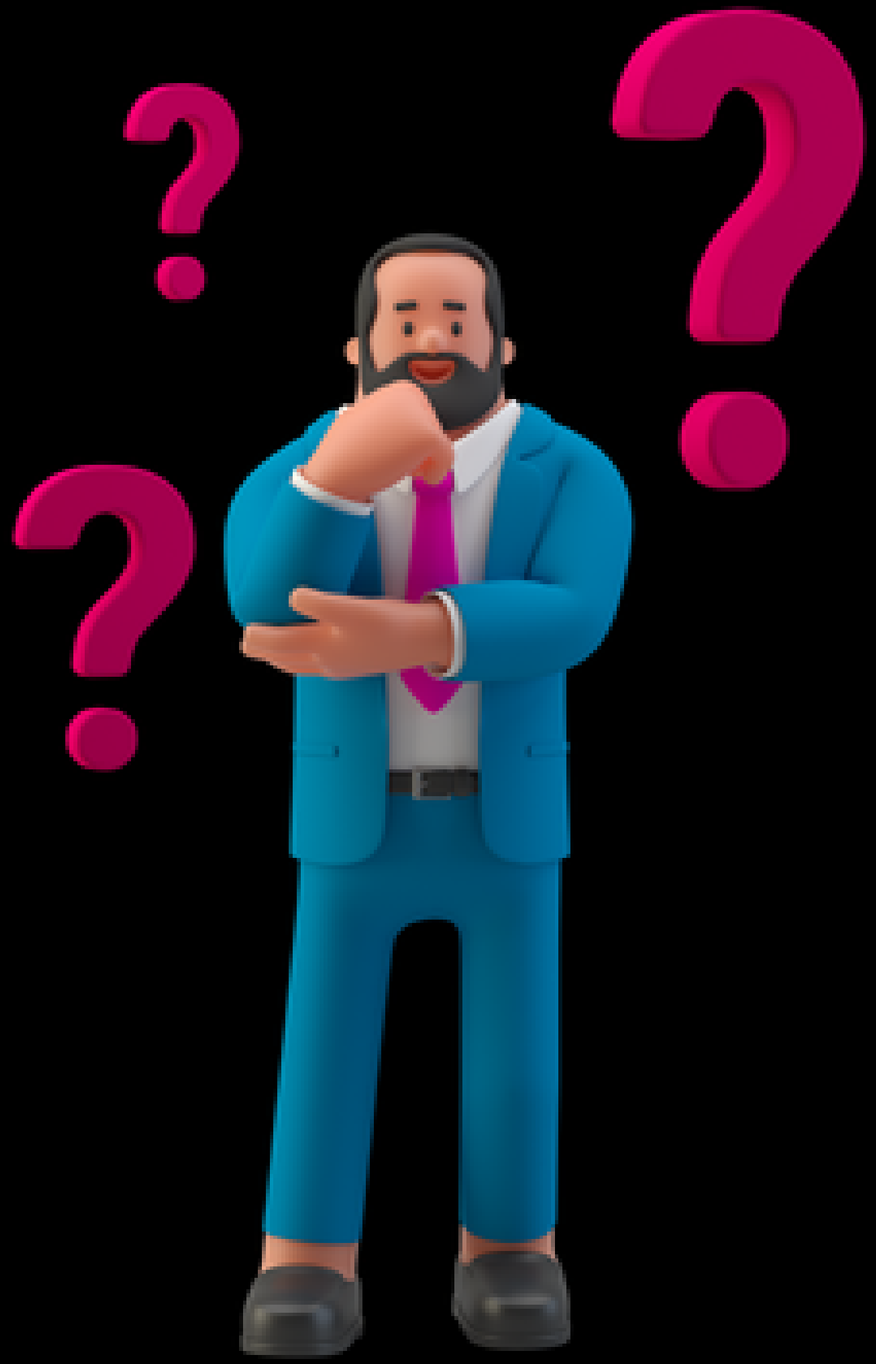
```
SELECT ISDATE('2022-06-15');
```

1

THANK YOU

"Happiness is not something ready-made. It comes from your own actions."

Dalai Lama



DO YOU KNOW?

String Functions

Interview Question

SQL

BY MANISH KUMAR CHAUDHARY

1. **LEN:** This function returns the length of a string.

```
SELECT LEN('Hello World') -- Returns 11
```

2. **SUBSTRING:** This function returns a part of a string.

```
SELECT SUBSTRING('Hello World', 1, 5) -- Returns 'Hello'
```

3. **CONCAT:** This function concatenates two or more strings.

```
SELECT CONCAT('Hello', ' ', 'World') -- Returns 'Hello World'
```

4. **LOWER:** This function converts a string to lowercase.

```
SELECT LOWER('Hello World') -- Returns 'hello world'
```

5. **UPPER:** This function converts a string to uppercase.

```
SELECT UPPER('Hello World') -- Returns 'HELLO WORLD'
```

6. **REPLACE:** This function replaces a substring in a string with another substring.

```
SELECT REPLACE('Hello World', 'Hello', 'Hi') -- Returns 'Hi World'
```

7. **LTRIM:** This function removes leading spaces which are on left side of a string.

```
SELECT LTRIM('   Hello World') -- Returns 'Hello World'
```

8. **RTRIM:** This function removes leading spaces which are on right side of a string.

```
SELECT RTRIM('Hello World   ') -- Returns 'Hello World'
```

9. **CHARINDEX:** This function returns the position of a substring in a string.

```
SELECT CHARINDEX('o', 'Hello World', 5) -- Returns 8
```

10. **LEFT:** This function returns the left part of a string with a specified number of characters.

```
SELECT LEFT('Hello World', 5) -- Returns 'Hello'
```

11. **RIGHT:** This function returns the right part of a string with a specified number of characters.

```
SELECT RIGHT('Hello World', 5) -- Returns 'World'
```

12. **STUFF:** This function replaces a part of a string with another string.

```
SELECT STUFF('Hello World', 7, 5, 'Universe') -- Returns 'Hello Universe'
```

13. **TRIM:** This function removes leading and trailing spaces from a string.

```
SELECT TRIM('   Hello World   ') -- Returns 'Hello World'
```

14. **REVERSE:** This function reverses a string.

```
SELECT REVERSE('Hello World') -- Returns 'dlroW olleH'
```

THANK YOU

"The biggest adventure you can ever take is to live the life of your dreams."

Oprah Winfrey

BY MANISH KUMAR CHAUDHARY

Interview Question

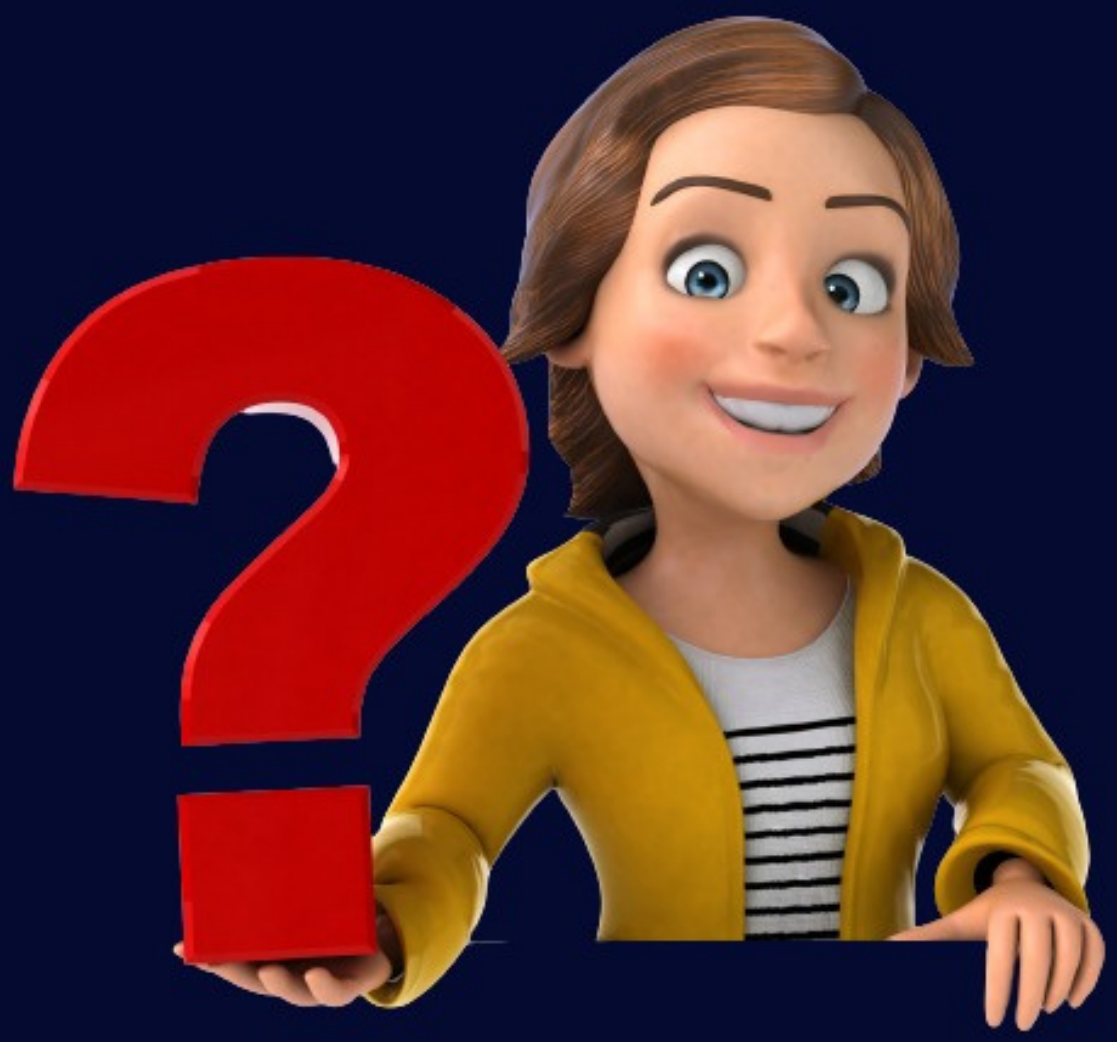
SQL

Find the fifth highest salary without using TOP or LIMIT.

DIFFICULTY LEVEL :MEDIUM

Question From

stratascratch



PROBLEM STATEMENT

You have been asked to find the fifth highest salary without using TOP or LIMIT.

Note: Duplicate salaries should not be removed.

worker Table

FIELD	TYPE
worker_id	int
first_name	varchar
last_name	varchar
salary	int
joining_date	datetime
department	varchar

worker Table Input

worker_id	first_name	last_name	salary	joining_date	department
1	Monika	Arora	100000	2014-02-20	HR
2	Niharika	Verma	80000	2014-06-11	Admin
3	Vishal	Singhal	300000	2014-02-20	HR
4	Amitah	Singh	500000	2014-02-20	Admin
5	Vivek	Bhati	500000	2014-06-11	Admin
6	Vipul	Diwan	200000	2014-06-11	Account
7	Satish	Kumar	75000	2014-01-20	Account
8	Geetika	Chauhan	90000	2014-04-11	Admin
9	Agepi	Argon	90000	2015-04-10	Admin
10	Moe	Acharya	65000	2015-04-11	HR
11	Nayah	Laghari	75000	2014-03-20	Account
12	Jai	Patel	85000	2014-03-21	HR

QUERY 1

```sql
SELECT w1.salary
FROM worker w1
WHERE (SELECT COUNT( w2.salary)
       FROM worker w2
       WHERE w2.salary>w1.salary)=4
```

QUERY EXPLANATION

1.We are using correlated subquery to get this fifth highest salary.

2.With WHERE condition we are counting the number of salary values from w2 table (inner query) that are greater than the each salary value from w1 table (outer query).

Here, we are equating it to 4 because counting start from 0. Thus, for fith highest value we will get 4 as count.

HOW IT IS COUNTING IN BACKGROUND

salary	count
500000	0
500000	0
300000	2
200000	3
100000	4
90000	5
90000	5
85000	7
80000	8
75000	9
75000	9
65000	11

OUTPUT

highest_salary
100000

QUERY 2

```
SELECT salary
FROM worker
ORDER BY salary DESC
OFFSET 4 ROWS
FETCH NEXT 1 ROWS ONLY
```

QUERY EXPLANATION

1.In this query we are ordering salary on the basis of descending order.

So, we will get highest salary at top.

2.We are using OFFSET 4. It will skip 4 rows. And here FETCH NEXT 1 Row will help to get the fifth highest salary.

We can do same thing ROW_NUMBER also. So you can try this from your side.

BY MANISH KUMAR CHAUDHARY

THANK YOU

"Don't watch the clock; do what it does. Keep going."

Sam Levenson

BY MANISH KUMAR CHAUDHARY

Interview Question

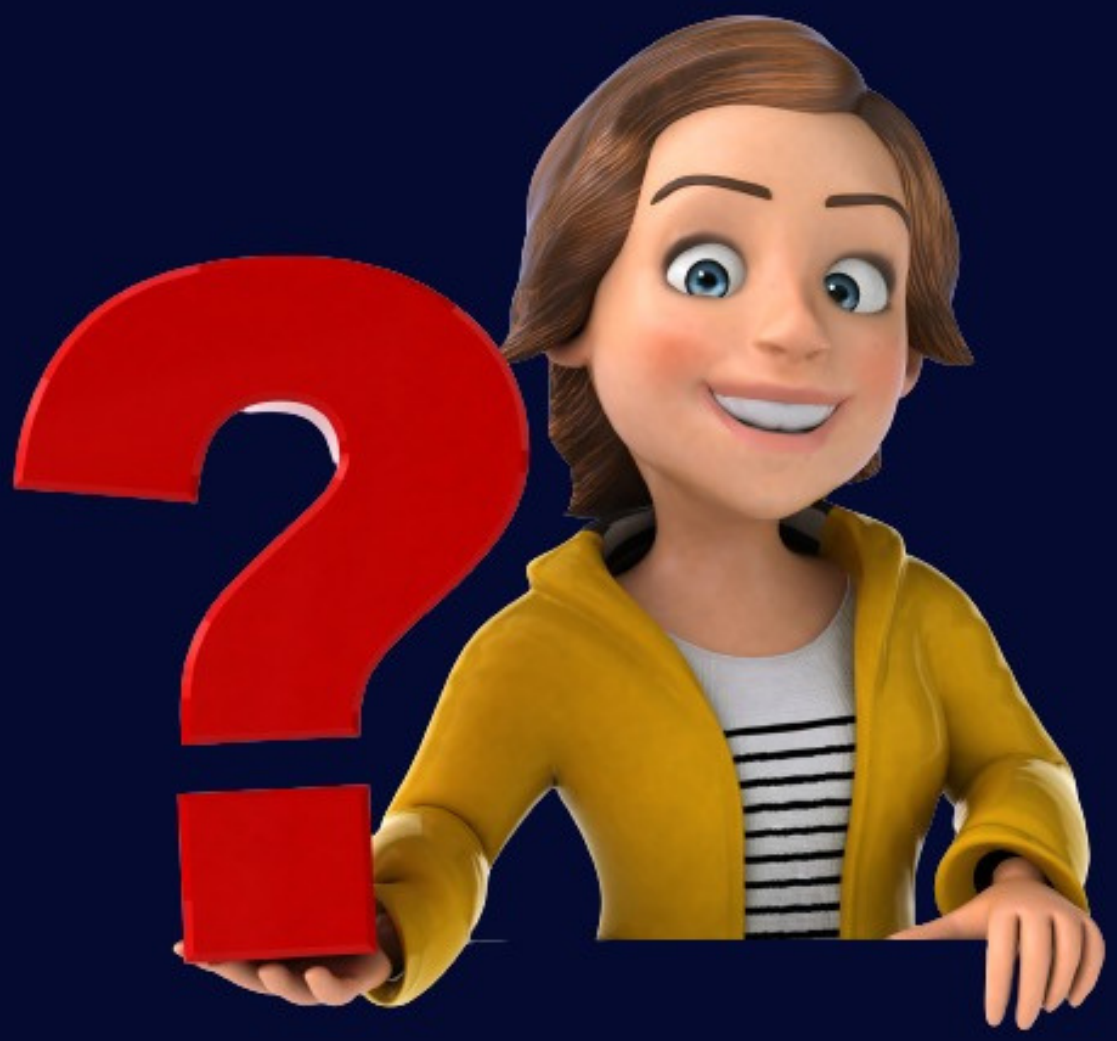
SQL

From Microsoft to Google

DIFFICULTY LEVEL :HARD

Question From

stratascratch



PROBLEM STATEMENT

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)?

linkedin_users

FIELD	TYPE
user_id	int
employer	varchar
position	varchar
start_date	datetime
end_date	datetime

linkedin_users Input Table

user_id	employer	position	start_date	end_date
1	Microsoft	developer	2020-04-13	2021-11-01
1	Google	developer	2021-11-01	
2	Google	manager	2021-01-01	2021-01-11
2	Microsoft	manager	2021-01-11	
3	Microsoft	analyst	2019-03-15	2020-07-24
3	Amazon	analyst	2020-08-01	2020-11-01
3	Google	senior analyst	2020-11-01	2021-03-04
4	Google	junior developer	2018-06-01	2021-11-01
4	Google	senior developer	2021-11-01	
5	Microsoft	manager	2017-09-26	
6	Google	CEO	2015-10-02	

MENTAL APPROACH

1.We just need to find the previous and current employer for all users and if previous employer is Microsoft and new employer is Google then we will count that user.

QUERY

```sql
WITH cte AS (
SELECT user_id
       ,employer
       ,LEAD(employer,1,'NA') OVER (PARTITION BY user_id
                                    ORDER BY start_date) new_employer
FROM linkedin_users
)
SELECT COUNT(user_id) no_of_employers
FROM cte
WHERE employer='Microsoft'
AND new_employer='Google'
```

QUERY EXPLANATION

1.With cte we are getting the required columns along with the new_employer column.
For getting new_employer column we are using LEAD function. So with this corresponding to old employer we will have record of new_employer also.
Here in LEAD function we have provided offset value as 1 and Default value as 'NA'. If there is no new_employer then we will get 'NA' for them.

2.Now we are simply COUNTING the users who fulfill the condition. Condition is that previous employer be Microsft and just new employer be 'Google'

OUTPUT

no_of_employers
1

BY MANISH KUMAR CHAUDHARY

THANK YOU

"The only limit to our realization of tomorrow will be our doubts of today."

Franklin D. Roosevelt

BY MANISH KUMAR CHAUDHARY

Interview Question

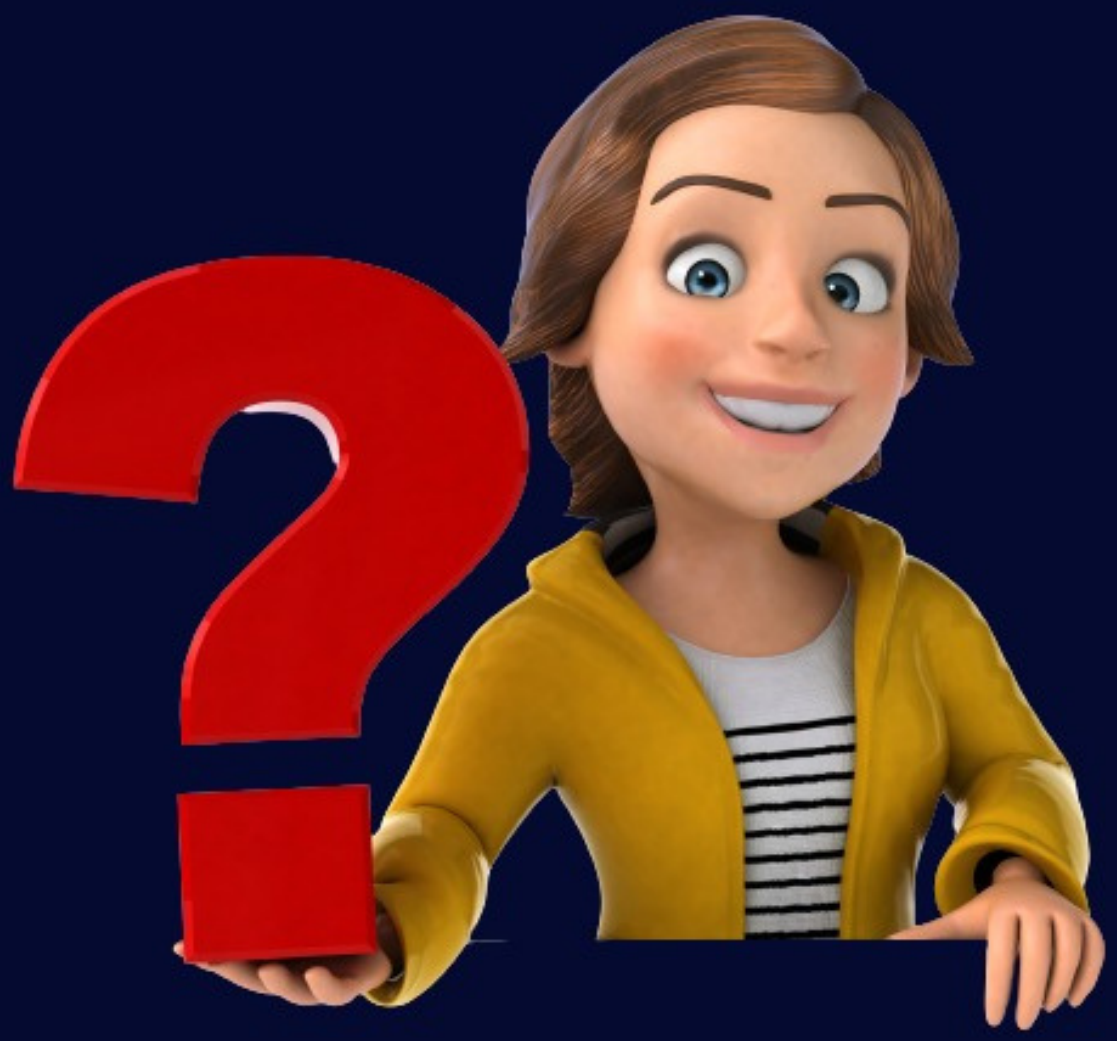
SQL

Invalid Bank Transactions

DIFFICULTY LEVEL :MEDIUM

Question From

stratascratch



PROBLEM STATEMENT

Bank of Ireland has requested that you detect invalid transactions in December 2022.
An invalid transaction is one that occurs outside of the bank's normal business hours.
The following are the hours of operation for all branches:

Monday – Friday 09:00 – 16:00
Saturday & Sunday Closed
Irish Public Holidays 25th and 26th December

Determine the transaction ids of all invalid transactions.

boi_transactions

FIELD	TYPE
transaction_id	int
time_stamp	varchar

boi_transactions Sample Table

transaction_id	time_stamp
1001	2022-12-01 09:08
1002	2022-12-01 09:36
1003	2022-12-01 09:48
1004	2022-12-01 10:05
1005	2022-12-01 12:12

QUERY

```sql
SELECT transaction_id
FROM boi_transactions
WHERE FORMAT(CAST(time_stamp AS datetime),'dddd')
      IN ('Saturday','Sunday')
OR DATEPART(HOUR,time_stamp) NOT BETWEEN 9 AND 16
AND FORMAT(CAST(time_stamp AS datetime),'dd MMMM')
    IN ('25 December','26 December')
```

QUERY EXPLANATION

1.Here we have simply giving the conditions as per requirement.

We have extracted using Datepart and Format. Here, we have CASTED to datetime because original datatype of time_stamp was VARCHAR

BY MANISH KUMAR CHAUDHARY

THANK YOU

“Believe in yourself. Stay in your own lane. There’s only one you.”

Queen Latifah

BY MANISH KUMAR CHAUDHARY

Interview Question

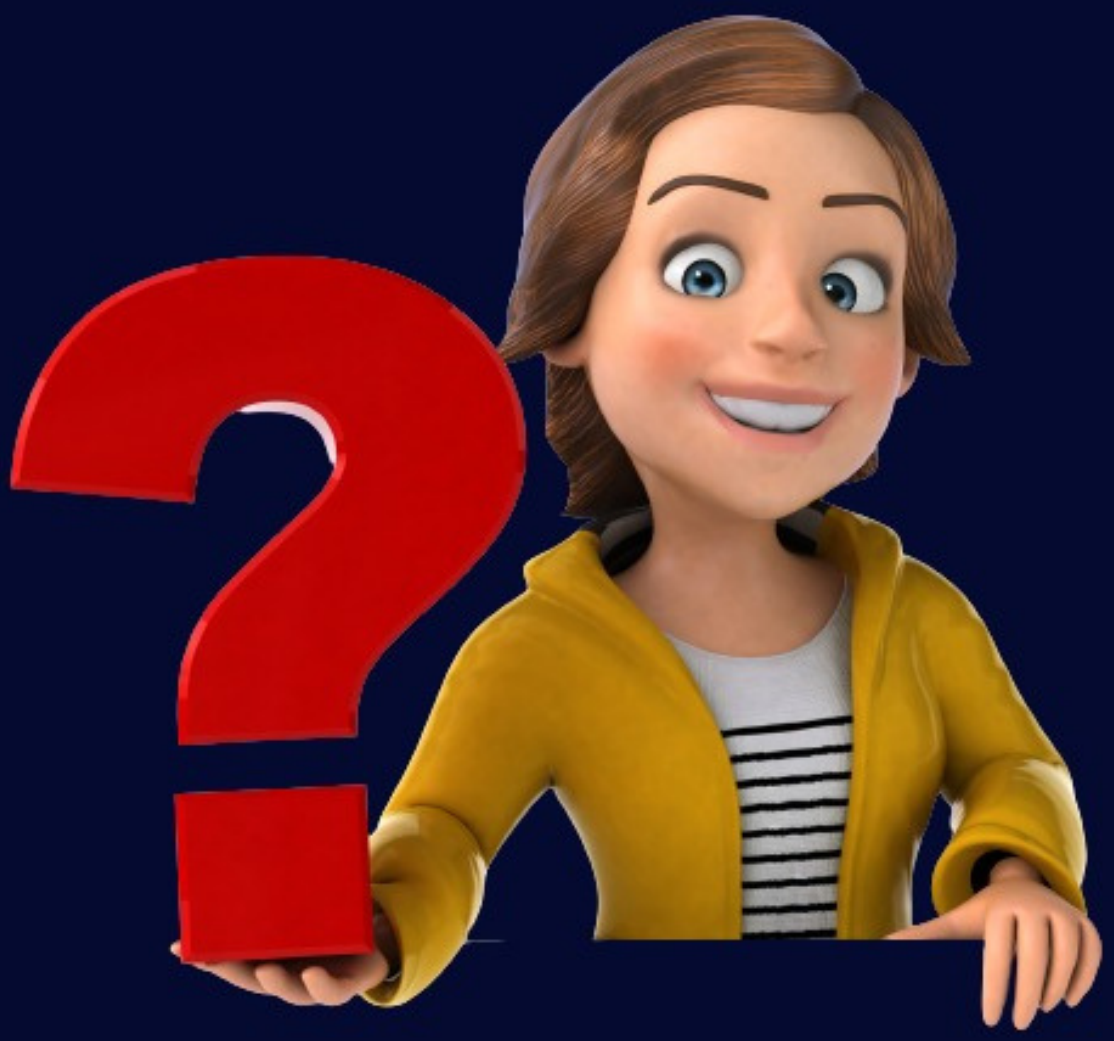
SQL

Naive Forecasting

DIFFICULTY LEVEL :HARD

Question From

stratascratch



NAIVE FORECASTING

Naive forecasting is a simple forecasting technique that assumes a variable's future value will be the same as its present or previous value, without taking into account any other factors or trends that may influence the variable. This method is often referred to as the "naive method," the "last value method," or the "persistence method."

In some circumstances, where the underlying variable is generally stable and consistent across time, and there are no big changes or trends that could alter it, naive forecasting can be effective. However, because it does not account for variables that are prone to major fluctuations or trends, it may not be appropriate for these variables. More advanced forecasting methods, such as time series analysis or regression analysis, may be more applicable in such instances.

PROBLEM STATEMENT

Some forecasting methods are extremely simple and surprisingly effective. Naïve forecast is one of them; we simply set all forecasts to be the value of the last observation. Our goal is to develop a naïve forecast for a new metric called "distance per dollar" defined as the (distance_to_travel/monetary_cost) in our dataset and measure its accuracy.

HOW TO ACHIEVE

To develop this forecast, sum "distance to travel" and "monetary cost" values at a monthly level before calculating "distance per dollar". This value becomes your actual value for the current month. The next step is to populate the forecasted value for each month. This can be achieved simply by getting the previous month's value in a separate column. Now, we have actual and forecasted values. This is your naïve forecast. Let's evaluate our model by calculating an error matrix called root mean squared error (RMSE). RMSE is defined as sqrt(mean(square(actual - forecast))). Report out the RMSE rounded to the 2nd decimal spot.

QUERY

```sql
WITH naive_cte AS (
SELECT MONTH(request_date) month_no
       ,SUM(distance_to_travel)/SUM(monetary_cost) actual
       ,LAG(SUM(distance_to_travel)/SUM(monetary_cost))
          OVER (ORDER BY MONTH(request_date)) forecast
FROM uber_request_logs
GROUP BY MONTH(request_date)
)
SELECT month_no
       ,actual
       ,forecast
       ,CAST(SQRT(AVG(SQUARE(actual-forecast))) AS DECIMAL(15,2)) RMSE
FROM naive_cte
GROUP BY month_no,actual,forecast
```

QUERY EXPLANATION

1.Used naive_cte to get month number along with distance_per_dollar as per the required condition.
Here, to get the previous month distance_per_dollar, I have used LAG window function and ordered by month number.

2.Now with SELECT query, fetched the rquired records along with the RMSE which is calculated which the formula provided to us.

OUTPUT

month_no	actual	forecast	RMSE
1	5.607		
2	5.806	5.607	0.20
3	5.949	5.806	0.14
4	3.042	5.949	2.91
5	3.657	3.042	0.61
6	6.636	3.657	2.98
7	2.591	6.636	4.04
8	4.413	2.591	1.82
9	1.328	4.413	3.08
10	2.808	1.328	1.48

BY MANISH KUMAR CHAUDHARY

THANK YOU

“You only live once, but if you do it right, once is enough.”

Mae West

BY MANISH KUMAR CHAUDHARY

Interview Question

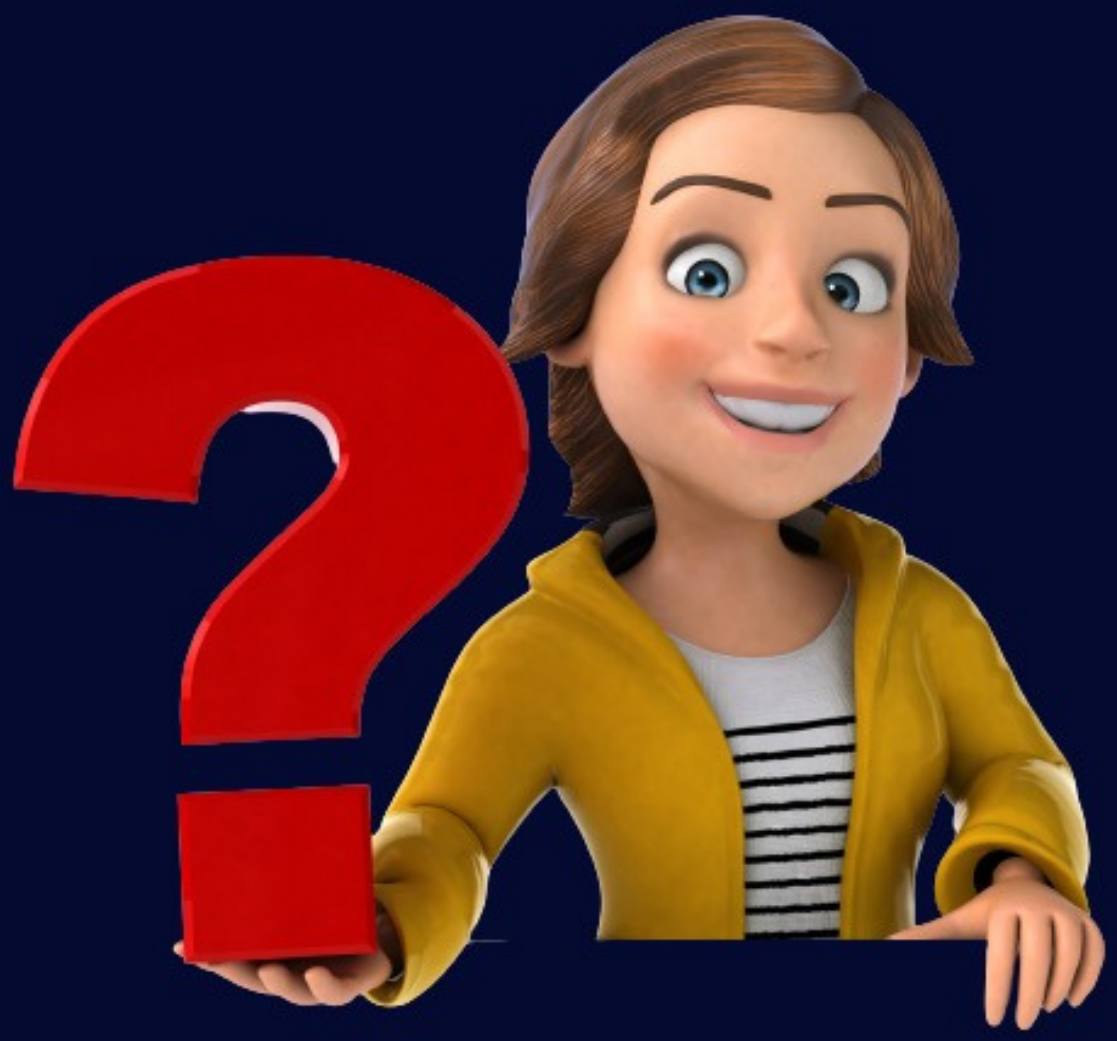
SQL

Old And Young Athletes

DIFFICULTY LEVEL :MEDIUM

Question From

stratascratch



PROBLEM STATEMENT

Find the old-to-young player ratio for each Olympic games. 'Old' is defined as ages 50 and older and 'young' is defined as athletes 25 or younger. Output the Olympic games, number of old athletes, number of young athletes, and the old-to-young ratio.

olympics_athletes_events Table

FIELD	TYPE
id	int
name	varchar
sex	varchar
age	float
games	varchar
year	int

olympics_athletes_events Sample Input Table

id	name	sex	age	games	year
3520	Guillermo J. Amparan	M		1924 Summer	1924
35394	Henry John Finchett	M		1924 Summer	1924
21918	Georg Frederik Ahrensborg Clausen	M	28	1924 Summer	1924
110345	Marinus Cornelis Dick Sigmond	M	26	1924 Summer	1924
54193	Thodore Tho Jeitz	M	55	1924 Summer	1924
23240	Charles Barney Cory	M	47	1904 Summer	1904

QUERY

```sql
WITH old_new_cte AS (
SELECT games
       ,SUM(CASE WHEN age >= 50 THEN 1
                 ELSE 0
        END) no_of_old_athletes
       ,SUM(CASE WHEN age <= 50 THEN 1
                 ELSE 0
        END) no_of_young_athletes
FROM olympics_athletes_events
WHERE age IS NOT NULL
GROUP BY games
)
SELECT games
       ,no_of_old_athletes
       ,no_of_young_athletes
       ,CAST(no_of_old_athletes*1.0/no_of_young_athletes AS
             DECIMAL(15,3)) old_to_young_ratio
FROM old_new_cte
```

Assumption: We are ignore those records where age is not available cause we don't know whether they belong to old or young.

QUERY EXPLANATION

I have used CTE for making code look clean and simple. We can do with single query without using cte also.

1.I have create old_new_cte to get the number of old and young athletes for each games.
For this i made use of CASE WHEN statment along with SUM function. Here, CASE statment is helping to flag 1 where condition is met and SUM is helping to add those 1s.

Here we have not specified condition for our assumption because CASE WHEN statement will itself now flag the null values.

2.In SELECT query we are simply geting the ration of old to young athletes.
Here we have used CAST to round the ratio.

SAMPLE OUTPUT

games	no_of_old_athletes	no_of_young_athletes	old_to_young_ratio
1900 Summer	1	27	0.037
1904 Summer	2	31	0.065
1908 Summer	3	55	0.055
1924 Summer	1	93	0.011
1936 Summer	1	3	0.333
1984 Summer	0	1	0.000
1988 Winter	0	2	0.000

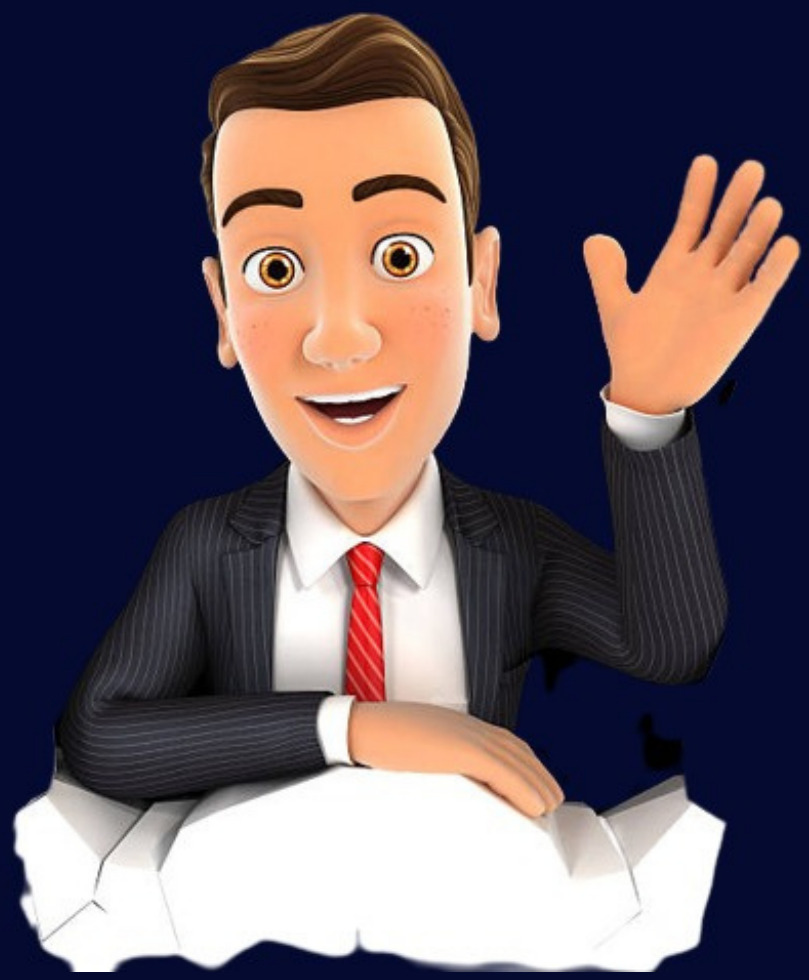
Please try to solve this other way. I will hope you will try

BY MANISH KUMAR CHAUDHARY

THANK YOU

Expect problems and eat them for breakfast.

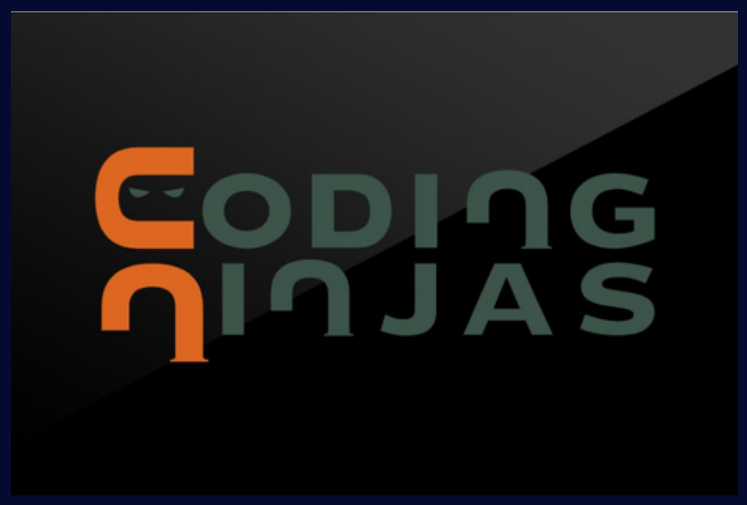
Alfred A. Montapert



BY MANISH KUMAR CHAUDHARY

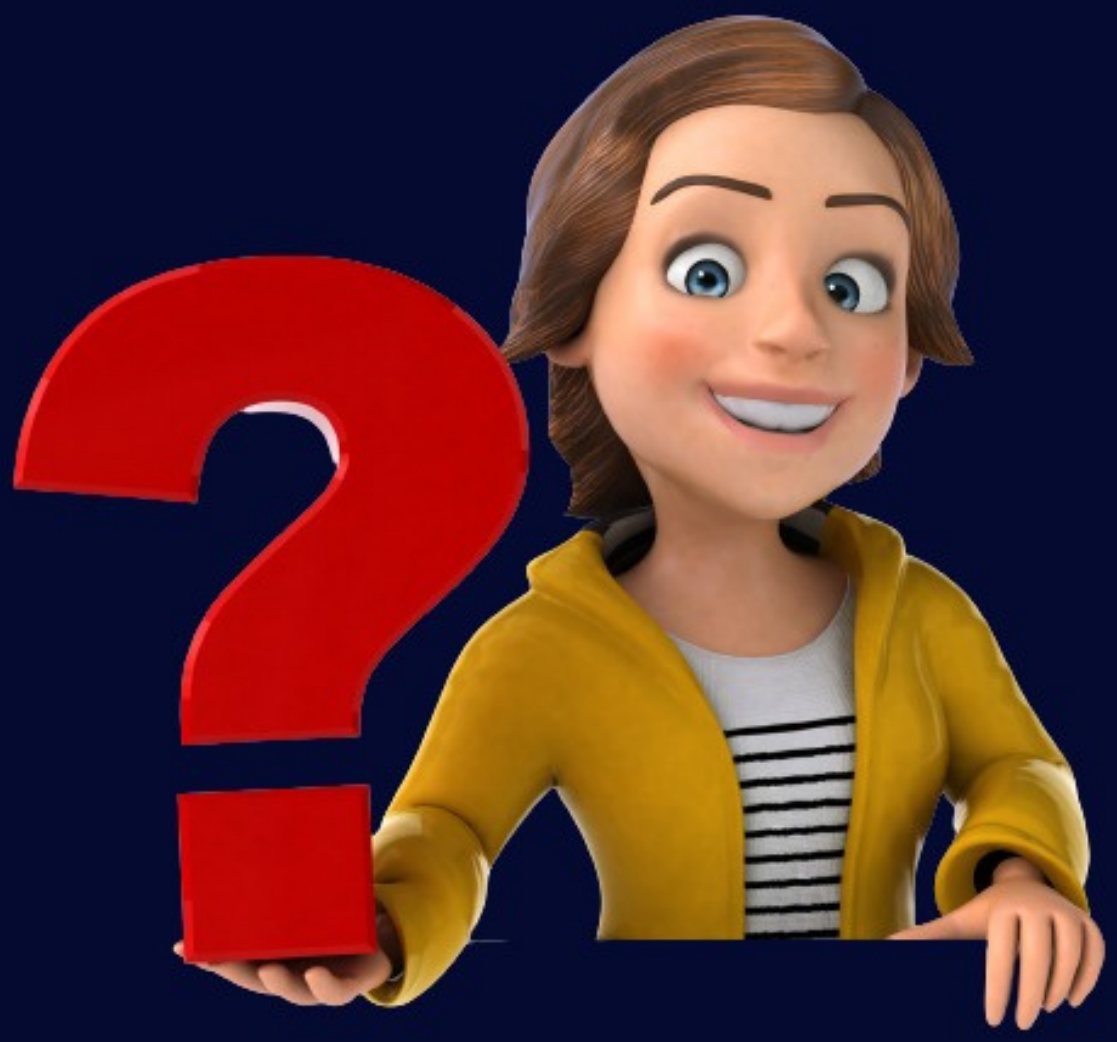
Interview Question

SQL



Report Contiguous Dates Problem

DIFFICULTY LEVEL :HARD



PROBLEM STATEMENT

A system is running one task every day. Every task is independent of the previous tasks. The tasks can fail or succeed.

Write an SQL query to generate a report of period_state for each continuous interval of days in the period from 2019-01-01 to 2019-12-31.

period_state is ‘failed’ if tasks in this interval failed or ‘succeeded’ if tasks in this interval succeeded. Interval of days are retrieved as start_date and end_date.

Table: Succeeded

Column Name	Type
success_date	date

Table: Failed

Column Name	Type
fail_date	date

Succeeded table:

success_date
2018-12-30
2018-12-31
2019-01-01
2019-01-02
2019-01-03
2019-01-06

Failed table:

fail_date
2018-12-28
2018-12-29
2019-01-04
2019-01-05

Result table:

period_state	start_date	end_date
succeeded	2019-01-01	2019-01-03
failed	2019-01-04	2019-01-05
succeeded	2019-01-06	2019-01-06

The report ignored the system state in 2018 as we care about the system in the period 2019-01-01 to 2019-12-31.
From 2019-01-01 to 2019-01-03 all tasks succeeded and the system state was "succeeded".
From 2019-01-04 to 2019-01-05 all tasks failed and system state was "failed".
From 2019-01-06 to 2019-01-06 all tasks succeeded and system state was "succeeded".

MENTAL APPROACH

1.First we need to combine both tables as we need to find contagious report dates and both table contains different dates.

2.Now go through contagious dates and bifurcate on the basis of succeeded and failed state.
Here, we are also separating start and end date for each conatgious dates on the basis of succeded and failed state.

QUERY

```sql
WITH union_cte AS (
SELECT success_date date_
      ,'succeeded' period_state
FROM succeeded
UNION ALL
SELECT fail_date date_
      ,'failed' period_state
FROM failed
),
row_num_cte AS (
SELECT date_
      ,period_state
      ,ROW_NUMBER() OVER(ORDER BY date_) row_num1
      ,ROW_NUMBER() OVER (PARTITION BY period_state
                          ORDER BY date_ ASC) row_num2
      ,ROW_NUMBER() OVER(ORDER BY date_)  -
       ROW_NUMBER() OVER (PARTITION BY period_state
                          ORDER BY date_ ASC) diff
FROM union_cte
WHERE YEAR(date_)=2019
)
SELECT period_state
      ,MIN(date_) start_date
      ,MAX(date_) end_date
FROM row_num_cte
GROUP BY period_state,diff
```

QUERY EXPLANATION

1.In first cte union_cte we are simply combining the dates from both table along with their state wether it is succeeded or failed.

2.The second cte row_num_cte we are finding the difference between row_num1 and row_num2 which will give value that will be same for contagious dates.
Here, we have also filtered to get results of the year 2019 only.

3.At last we are SELECTING the required records out of row_num_cte.
Here, we have GROUPED on the basis of period_state and diff because for each contagious date these two things will be same.

OUTPUT

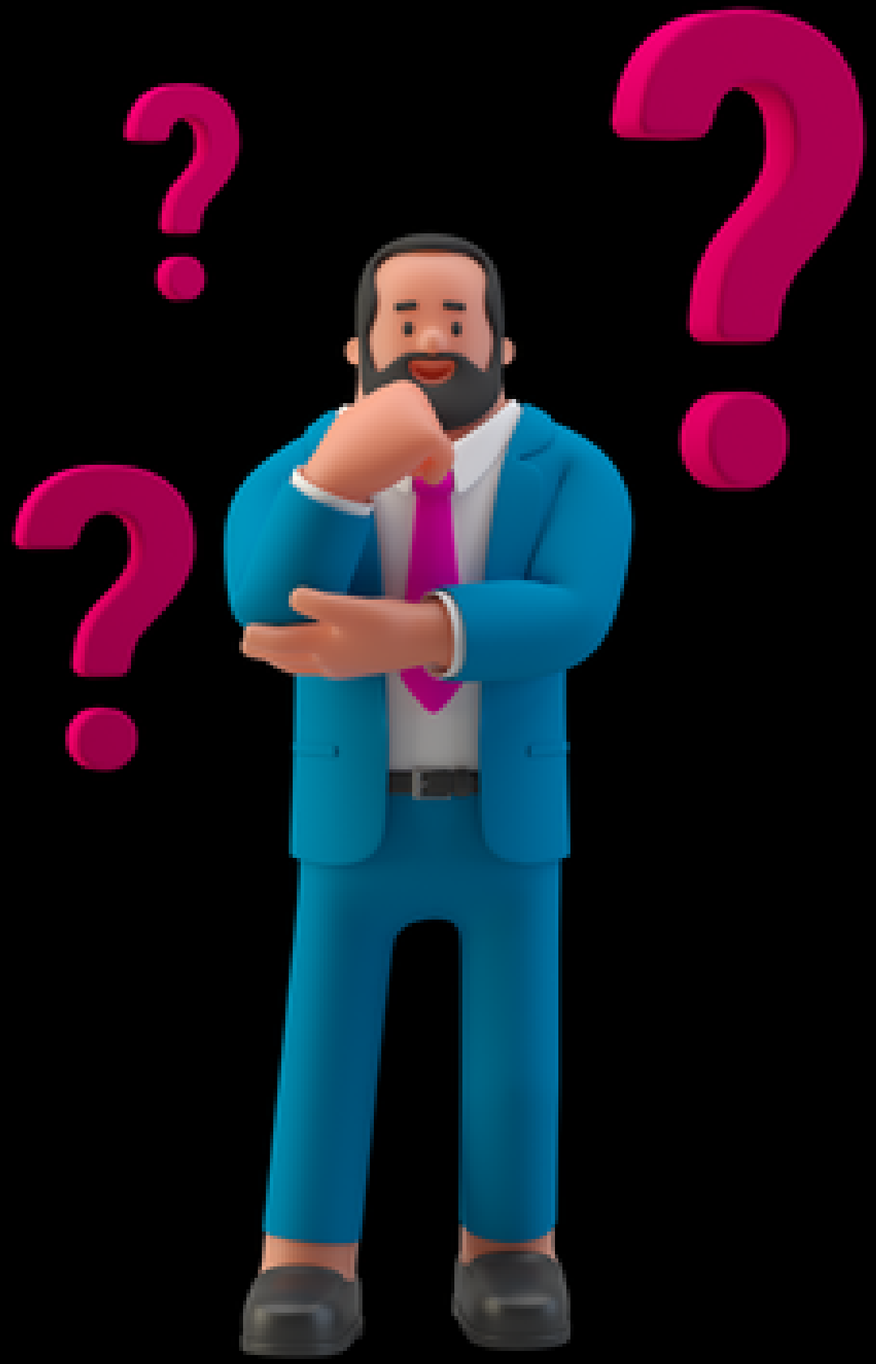
period_state	start_date	end_date
succeeded	2019-01-01	2019-01-03
succeeded	2019-01-06	2019-01-06
failed	2019-01-04	2019-01-05

BY MANISH KUMAR CHAUDHARY

THANK YOU

"Be happy in the moment, that's enough. Each moment is all we need, not more."

Mother Teresa



DO YOU KNOW?

Ranking Functions: ROW_NUMBER(), RANK() and DENSE_RANK()

Interview Question

SQL

BY MANISH KUMAR CHAUDHARY

WHAT IS RANKING FUNCTION?

Ranking functions return a ranking value for each row in a partition. Depending on the function that is used, some rows might receive the same value as other rows.

ROW_NUMBER(), RANK(), DENSE_RANK() all of them are ranking functions in SQL.

ROW_NUMBER()

ROW_NUMBER() is a function that assigns a unique sequential number to each row within a result set. It does not assign the same rank to multiple rows with the same value. For example, if two rows have the same value, then the first row will get row number 1, and the second row will get row number 2.

RANK()

RANK() is a function that assigns a rank to each row within a result set. It assigns the same rank to the rows with the same value and skips the next rank. For example, if two rows have the same value, then they will get the same rank, and the next row will get the rank of the skipped rank. For example, if two rows have rank 2, the next row will get rank 4.

DENSE_RANK()

DENSE_RANK() is a function that assigns a rank to each row within a result set. It assigns the same rank to the rows with the same value and does not skip the next rank. For example, if two rows have the same value, then they will get the same rank, and the next row will get the next rank. For example, if two rows have rank 2, the next row will get rank 3.

employees Table

id	name	department	salary
1	John	Marketing	50000
2	Jane	Sales	55000
3	Bob	Marketing	60000
4	Mary	Sales	55000
5	David	Marketing	70000
6	Alice	Sales	70000
7	Tom	HR	80000
8	Emily	HR	85000
9	Peter	IT	90000
10	Amy	IT	55000

We will make use of all ranking functions and rank the employees on the basis of decreasing order of salary

```sql
SELECT name
      ,salary
      ,ROW_NUMBER() OVER (ORDER BY salary DESC) row_num
      ,RANK() OVER (ORDER BY salary DESC) rnk
      ,DENSE_RANK() OVER (ORDER BY salary DESC) dense_rnk
FROM employees
```

OUTPUT

name	salary	row_num	rnk	dense_rnk
Peter	90000	1	1	1
Emily	85000	2	2	2
Tom	80000	3	3	3
David	70000	4	4	4
Alice	70000	5	4	4
Bob	60000	6	6	5
Mary	55000	7	7	6
Jane	55000	8	7	6
Amy	55000	9	7	6
John	50000	10	10	7

OUTPUT EXPLANATION

In output we can see that ROW_NUMBER is simplying providing rank without considering whether employess having same salary or not.

RANK and DENSE_RANK is providing rank by considering the same salary except RANK is providing same rank to same salary employees and skipping the sequence as many times same salary occurs.Wheras DENSE_RANKis also ranking same number to same salary but it is not skipping the sequence.

THANK YOU

“SOMETIMES IT TAKES A GOOD FALL TO REALLY KNOW WHERE YOU STAND”

HAYLEY WILLIAMS

BY MANISH KUMAR CHAUDHARY

Interview Question

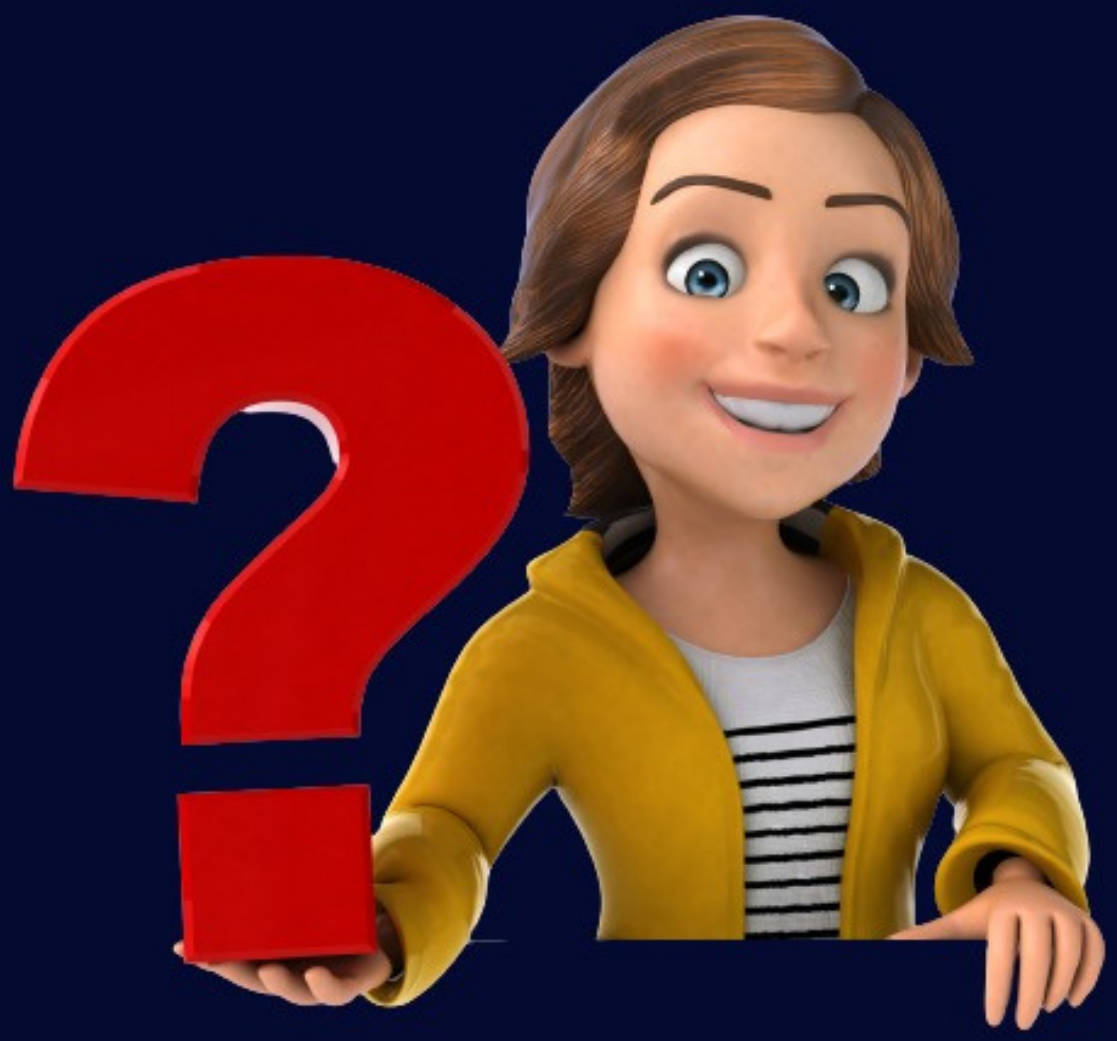
SQL

Seat Availability

DIFFICULTY LEVEL :HARD

Question From

stratascratch



PROBLEM STATEMENT

A movie theater gave you two tables: seats that are available for an upcoming screening and neighboring seats for each seat listed. You are asked to find all pairs of seats that are both adjacent and available.

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 with the higher number is in the second column.

theater_availability

FIELD	TYPE
seat_number	int
is_available	bool

theater_seatmap

FIELD	TYPE
seat_number	int
seat_left	int
seat_right	int

theater_availability Input Table

seat_number	is_available
11	FALSE
12	TRUE
13	FALSE
14	TRUE
15	TRUE
21	TRUE
22	TRUE
23	TRUE
25	FALSE
31	FALSE
32	TRUE
33	TRUE
35	TRUE

theater_seatmap Input Table

seat_number	seat_left	seat_right
11		12
12	11	13
13	12	14
14	13	15
15	14	
21		22
22	21	23
23	22	24
25	24	
31		32
32	31	33
33	32	34
35	34	

MENTAL APPROACH

1.Combine both tables and then check for seat numbers that are available.

2.Now find that seat number and its adjacent seat numbers.

3.Again check if adjacent seats for that particular seat number is available or not.
If both left and right adjacent seats are available then we will have two pairs of seats that will be available (one with left adjacent and that seat and another with right adjacent seat and that seat)

4.After getting all such combination just count the distinct pairs.

QUERY

```
WITH cte AS (
SELECT CASE WHEN seat_left IN (SELECT seat_number
                               FROM theater_availability
                               WHERE is_available='TRUE')
            THEN seat_left
       END available_left
      ,a.seat_number
      ,CASE WHEN seat_right IN (SELECT seat_number
                                FROM theater_availability
                                WHERE is_available='TRUE')
            THEN seat_right
       END available_right
FROM theater_availability a
INNER JOIN theater_seatmap s
ON a.seat_number=s.seat_number
WHERE a.is_available='TRUE'
),
seats AS (
SELECT available_left seat
      ,seat_number
FROM cte
WHERE seat_number-available_left=1
UNION ALL
SELECT available_right seat
      ,seat_number
FROM cte
WHERE available_right-seat_number=1
)
SELECT seat
      ,seat_number
FROM seats
WHERE seat<seat_number
```

QUERY EXPLANATION

1.With cte we are join both tables on the basis of seat number and flagging with left and right seats available by using CASE WHEN statement.
To check if the seat_left and seat_right is available or not we are using subquery inside the CASE WHEN statement.

2.Now with seats cte we are checking if the the seats are adjacent to each other or not.
To check this we are using conditionin WHERE condition. For left and right we are using different SELECT query and combining them with UNION operator.

3. Now we are simply selecting the reequired records and providing the condition that first column value is less than second column value.
We are doing this filter because we know that seat will repeat again with next row. Because for one row a seat may be right side but for the next row that same seat will be left side seat.

OUTPUT FOR cte

available_left	seat_number	available_right
	12	
	14	15
14	15	
	21	22
21	22	23
22	23	
	32	33
32	33	
	35	

OUTPUT for seats cte

seat	seat_number
14	15
21	22
22	23
32	33
15	14
22	21
23	22
33	32

FINAL OUTPUT

seat	seat_number
14	15
21	22
22	23
32	33

BY MANISH KUMAR CHAUDHARY

THANK YOU

“The present is life – all else is either memory or imagination.”

BY MANISH KUMAR CHAUDHARY

Interview Question

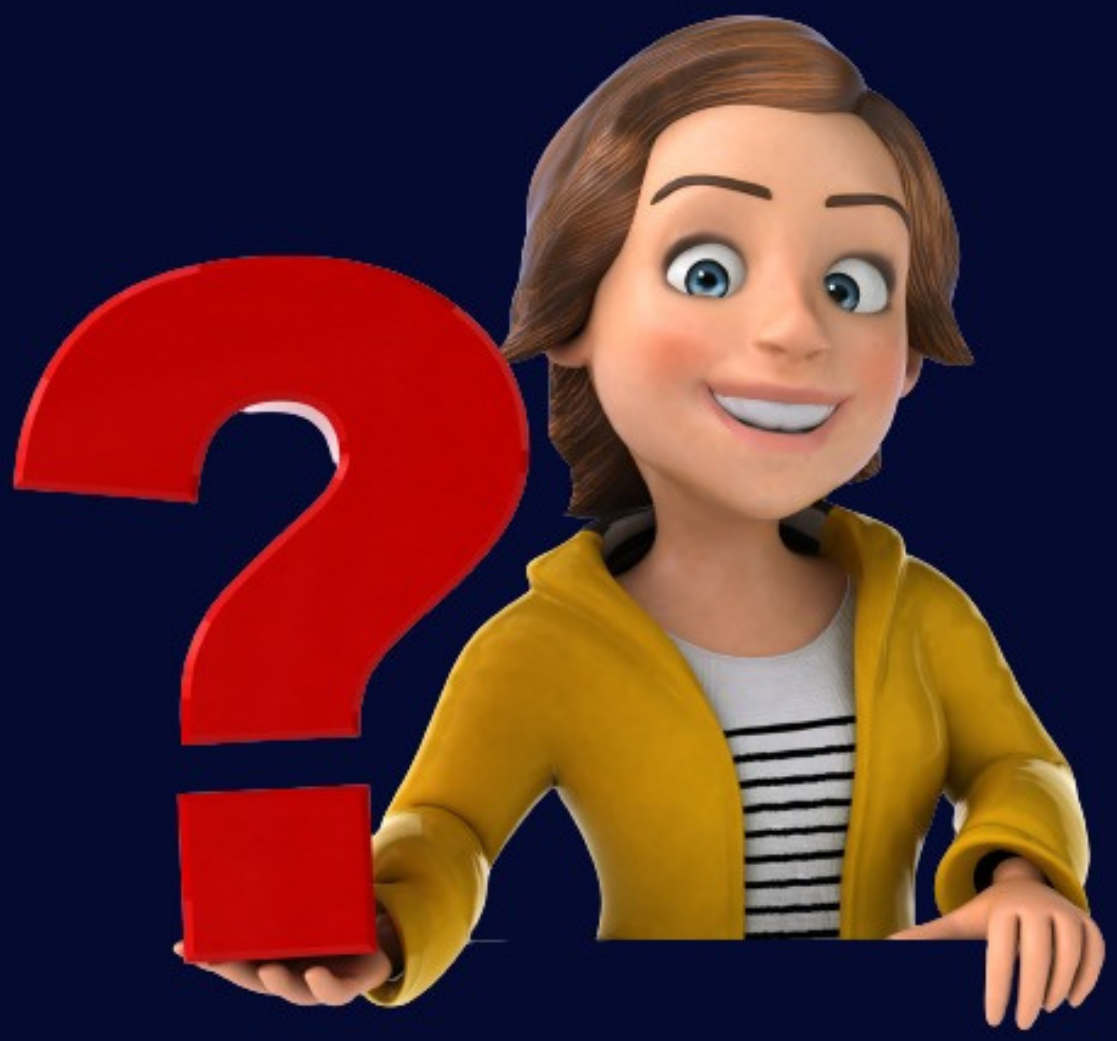
SQL

Top Percentile Fraud

DIFFICULTY LEVEL :HARD

Question From

stratascratch



PROBLEM STATEMENT

ABC Corp is a mid-sized insurer in the US and in the recent past their fraudulent claims have increased significantly for their personal auto insurance portfolio. They have developed a ML based predictive model to identify propensity of fraudulent claims. Now, they assign highly experienced claim adjusters for top 5 percentile of claims identified by the model. Your objective is to identify the top 5 percentile of claims from each state. Your output should be policy number, state, claim cost, and fraud score.

fraud_score Table

FIELD	TYPE
policy_num	varchar
state	varchar
claim_coste	int
fraud_score	float

fraud_score Sample Input Table

ABCD1097	CA	3330	0.653
ABCD1098	CA	1749	0.778
ABCD1099	CA	4692	0.075
ABCD1100	NY	1479	0.923
ABCD1101	NY	3338	0.532
ABCD1102	NY	3156	0.828
ABCD1103	NY	3279	0.013
ABCD1104	NY	4041	0.518
ABCD1105	NY	3138	0.091
ABCD1106	NY	2681	0.328
ABCD1107	NY	3354	0.508

QUERY

```sql
WITH percentile_rank_cte AS (
SELECT policy_num
      ,state
      ,claim_cost
      ,fraud_score
      ,NTILE(100) OVER (PARTITION BY state
                        ORDER BY fraud_score DESC) percentile_rank
FROM fraud_score)
SELECT policy_num
      ,state
      ,claim_cost
      ,fraud_score
FROM percentile_rank_cte
WHERE percentile_rank <= 5
```

QUERY EXPLANATION

1.We are using **percentile_rank_cte** to get all the required columns along with its percentile rank.

Here, we have used **NTILE(100)** function to divide the records into 100 equal parts on the basis of descending order of fraud_score and we are also partitioning on the basis of state.

2.With **SELECT** query we are simply fetching the required records and filtering so that we can get top 5 percetniles for each state.

SAMPLE OUTPUT

policy_num	state	claim_cost	fraud_score
ABCD1027	CA	2663	0.988
ABCD1016	CA	1639	0.964
ABCD1069	CA	1426	0.948
ABCD1222	FL	2392	0.988
ABCD1218	FL	1419	0.961
ABCD1291	FL	2581	0.939
ABCD1189	NY	3577	0.982
ABCD1117	NY	4903	0.978
ABCD1187	NY	3722	0.976
ABCD1196	NY	2994	0.973
ABCD1304	TX	1407	0.996
ABCD1398	TX	3191	0.978
ABCD1366	TX	2453	0.968

BY MANISH KUMAR CHAUDHARY

THANK YOU

Life isn't about getting and having, it's about giving and being.

Kevin Kruse

BY MANISH KUMAR CHAUDHARY

Interview Question

SQL

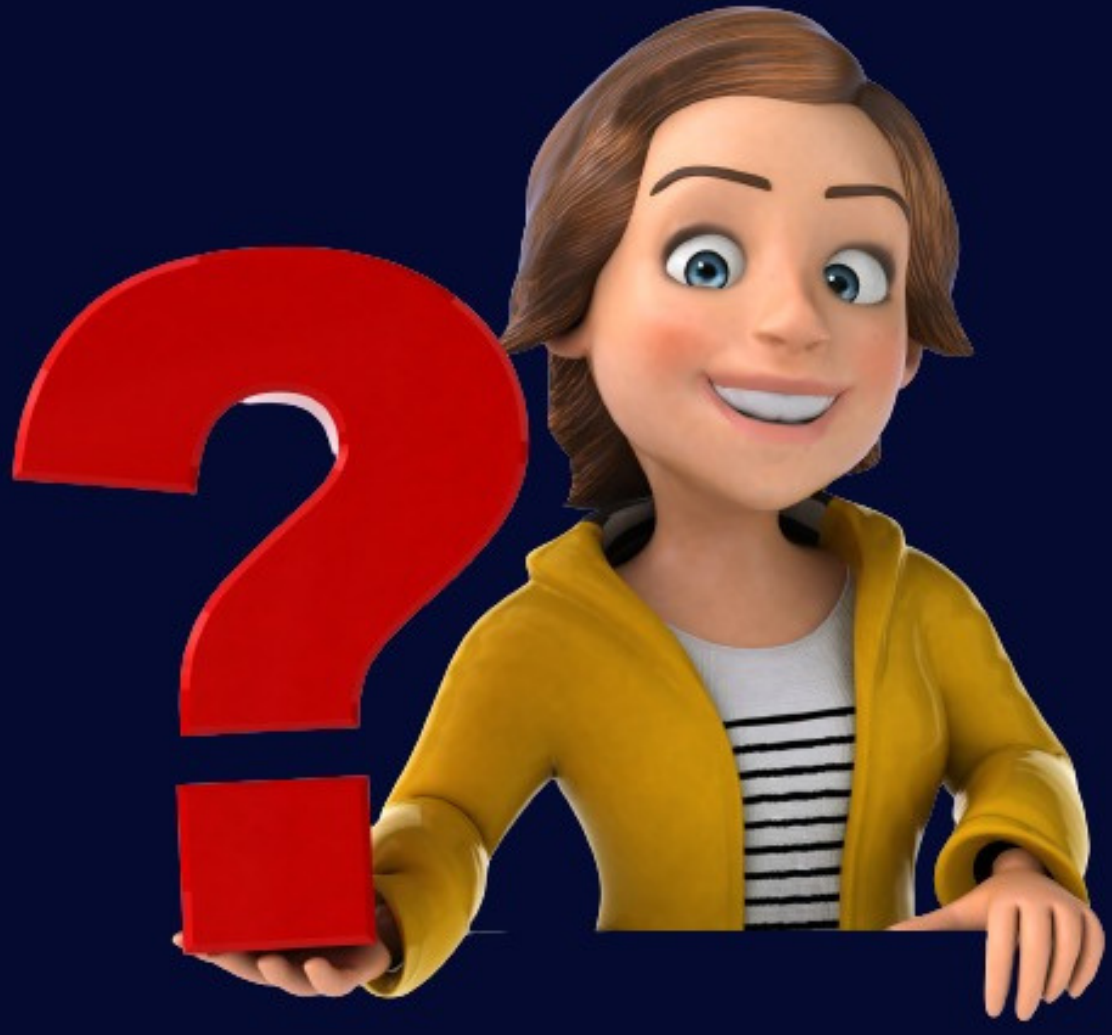
Total Sales By Year

DIFFICULTY LEVEL :HARD

Question From

Ankit Bansal

Yotube Channel



PROBLEM STATEMENT

Write an SQL query to report the Total sales amount of each item for each year, with corresponding product_id, report_year.
Dates of the sales years are between 2018 to 2020.
Return the result table ordered by product_id and report_year.

sales

FIELD	TYPE
product_id	int
period_start	date
period_end	date
average_daily_sales	int

sales Table Input

product_id	period_start	period_end	average_daily_sales
1	2019-01-25	2019-02-28	100
2	2018-12-01	2020-01-01	10
3	2019-12-01	2020-01-31	1

MENTAL APPROACH

1.Between period_start and period_end we will find the number of days there is and this we will do for all individual years that falls between.

2.Now we will find the total sales amount for each year for each product by using formula number of days*average_daily_sales

QUERY

```sql
WITH rec_cte AS (
SELECT MIN(period_start) dates
      ,MAX(period_end) max_date
FROM sales
UNION ALL
SELECT DATEADD(DAY,1,dates) dates
      ,max_date
FROM rec_cte
WHERE dates<max_date
),
join_cte AS (
SELECT r.dates
      ,r.max_date
      ,s.product_id
      ,s.average_daily_sales
FROM rec_cte r
INNER JOIN sales s
ON dates BETWEEN s.period_start AND s.period_end
)
SELECT product_id
      ,YEAR(dates) report_year
      ,SUM(average_daily_sales) total_amount
FROM join_cte
GROUP BY product_id,YEAR(dates)
ORDER BY product_id
OPTION (MAXRECURSION 1000)
```

QUERY EXPLANATION

1.We are using recursive cte rec_cte to get the all the dates from minimum date to maximum date that is available in our data.
Here. fist statement before UNION ALL is called anchor as it run one time only and and after UNION ALL it runs recursively till the maximum date.

2.Now we are using CTE join_cte to join the sales table on the basis of dates so that we can get average_saels_amount and product_id from sales table on our record.

3.At last we are SELECTING the required records and getting the total sales amount for each product id for each year.

STEP 1 OUTPUT will be all dates between minimum period_start and maximum period_end

STEP 1 and 2 together will give output as below for each product for each dates.
Here, I have only pasted sample data.

dates	max_date	product_id	average_daily_sales
2019-02-28	2020-01-31	1	100
2019-01-25	2020-01-31	1	100
2019-01-26	2020-01-31	1	100
2019-01-27	2020-01-31	1	100
2019-01-28	2020-01-31	1	100
2019-01-29	2020-01-31	1	100
2019-01-30	2020-01-31	1	100
2019-01-31	2020-01-31	1	100

FINAL OUTPUT

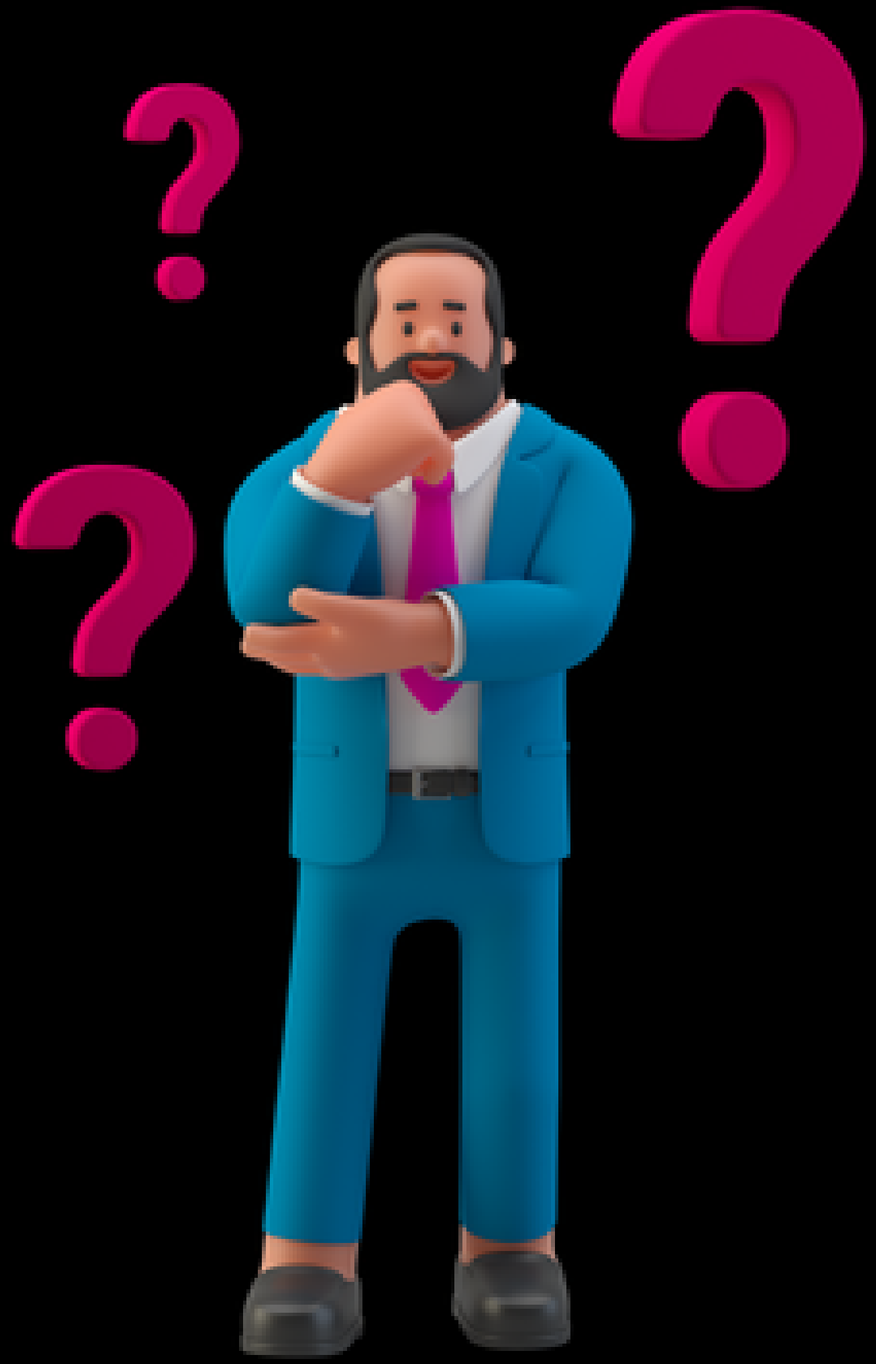
product_id	report_year	total_amount
1	2019	3500
2	2018	310
2	2019	3650
2	2020	10
3	2019	31
3	2020	31

BY MANISH KUMAR CHAUDHARY

THANK YOU

If you're going through hell, keep going.

Winston Churchill



DO YOU KNOW?

Types of Constraints in SQL

Interview Question

SQL

BY MANISH KUMAR CHAUDHARY

CONSTRAINTS IN SQL

Constraints in SQL are rules that we can apply to the data stored in a database table. These rules help ensure that the data is accurate and consistent by enforcing certain conditions on the data.

Types of constraints: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK , DEFAULT

NOT NULL

This constraint ensures that a column cannot have null or empty values. It requires that a value must be entered in that column for every row.

UNIQUE

This constraint ensures that each value in a column is unique and does not repeat. It helps to prevent duplicates in the data.

PRIMARY KEY

This constraint is a combination of NOT NULL and UNIQUE. It identifies a unique record in a table and helps to ensure that it is easily retrievable.

FOREIGN KEY

This constraint is used to link data between tables. It ensures that data in one table matches data in another table. It is used to maintain referential integrity between tables.

CHECK

This constraint is used to limit the values that can be entered into a column. It allows you to specify a condition that must be met before a value can be inserted or updated.

DEFAULT

This constraint provides a default value for a column when no value is specified. It ensures that a value is always present, even if it is not specified explicitly.

THANK YOU

“Success is liking yourself, liking what you do, and liking how you do it.”

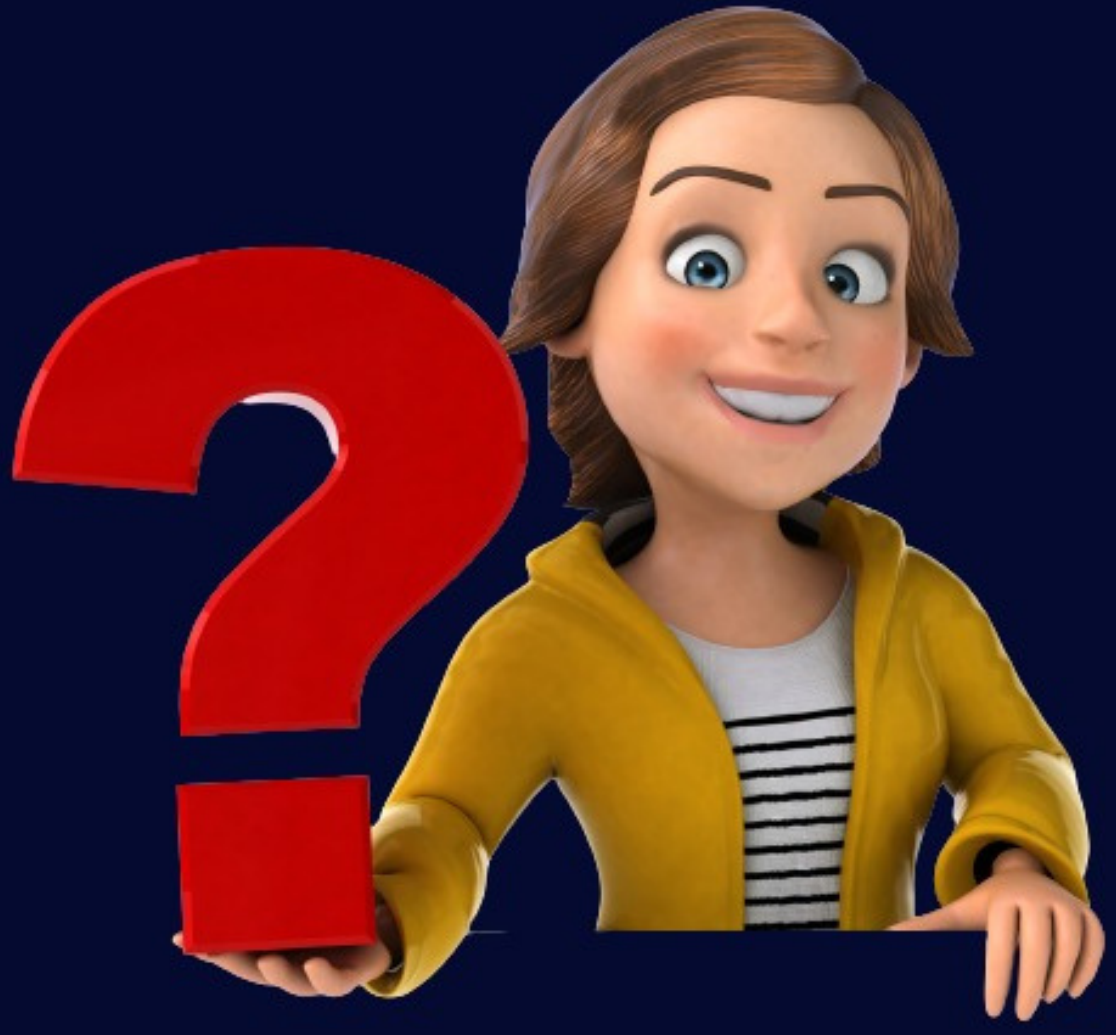
Maya Angelou

BY MANISH KUMAR CHAUDHARY

Interview Question

SQL

Write SQL to merge events with overlapping dates.



DIFFICULTY LEVEL :HARD

Question From

Ankit Bansal

Yotube Channel

PROBLEM STATEMENT

Write SQL to merge events with overlapping dates.

Input:
HallEvents table:

hall_id	start_day	end_day
1	2023-01-13	2023-01-14
1	2023-01-14	2023-01-17
1	2023-01-18	2023-01-25
2	2022-12-09	2022-12-23
2	2022-12-13	2022-12-17
3	2022-12-01	2023-01-30

Output:

hall_id	start_day	end_day
1	2023-01-13	2023-01-17
1	2023-01-18	2023-01-25
2	2022-12-09	2022-12-23
3	2022-12-01	2023-01-30

QUERY

```sql
WITH cte AS (
SELECT hall_id
       ,start_date
       ,end_date
       ,LAG(end_date) OVER (PARTITION BY hall_id
                            ORDER BY end_date) previous_end_date
FROM hall_events
)
SELECT hall_id
       ,MIN(start_date) start_date
       ,MAX(end_date) end_date
FROM cte
WHERE start_date<=previous_end_date OR previous_end_date IS NULL
GROUP BY hall_id
UNION
SELECT hall_id
       ,MIN(start_date) start_date
       ,MAX(end_date) end_date
FROM cte
WHERE start_date>previous_end_date
GROUP BY hall_id
```

QUERY EXPLANATION

1.With cte we are getting the required records along with the previous end date using LAG function.

Output we will get

hall_id	start_date	end_date	previous_end_date
1	2023-01-13	2023-01-14	NULL
1	2023-01-14	2023-01-17	2023-01-14
1	2023-01-15	2023-01-17	2023-01-17
1	2023-01-18	2023-01-25	2023-01-17
2	2022-12-13	2022-12-17	NULL
2	2022-12-09	2022-12-23	2022-12-17
3	2022-12-01	2023-01-30	NULL

2.Now we are querying for the required records using UNION. First SELECT query we have used so that we get the records where start_date<=previous_end_date OR previous_end_date IS NULL

This will give output as

hall_id	start_date	end_date
1	2023-01-13	2023-01-17
2	2022-12-09	2022-12-23
3	2022-12-01	2023-01-30

3.Second SELECT query after UNION we using to get the records WHERE start_date >previous_end_date. This will help us to get those records which are not within any criteria range.

Output for this will be

hall_id	start_date	end_date
1	2023-01-18	2023-01-25

FINAL OUTPUT

hall_id	start_date	end_date
1	2023-01-13	2023-01-17
1	2023-01-18	2023-01-25
2	2022-12-09	2022-12-23
3	2022-12-01	2023-01-30

QUERY FROM ANKIT SIR

```sql
WITH iterator_cte AS (
SELECT hall_id
       ,start_date
       ,end_date
       ,ROW_NUMBER() OVER (ORDER BY hall_id) iterator_id
FROM hall_events
),
rcte AS (
(SELECT hall_id
        ,start_date
        ,end_date
        ,iterator_id
        ,1 AS flag
 FROM iterator_cte
 WHERE iterator_id=1)
 UNION ALL
 SELECT i.hall_id
        ,i.start_date
        ,i.end_date
        ,i.iterator_id
        ,CASE WHEN r.hall_id=i.hall_id
              AND (i.start_date BETWEEN r.start_date AND r.end_date
               OR (r.start_date BETWEEN i.start_date AND i.end_date))
               THEN 0 ELSE 1 END +flag AS flag
  FROM rcte r
  INNER JOIN iterator_cte i
  ON r.iterator_id+1=i.iterator_id
  )
 SELECT hall_id
        ,MIN(start_date)
        ,MAX(end_date)
FROM rcte
GROUP BY hall_id,flag
```

QUERY EXPLANATION

1.With iterator_cte we are simply providing a numbering to the records so that we can iterate using the recursive cte.

Output for this will be

hall_id	start_date	end_date	iterator_id
1	2023-01-13	2023-01-14	1
1	2023-01-14	2023-01-17	2
1	2023-01-15	2023-01-17	3
1	2023-01-18	2023-01-25	4
2	2022-12-09	2022-12-23	5
2	2022-12-13	2022-12-17	6
3	2022-12-01	2023-01-30	7

2.Now we are using rcte for recursively running the query.
The first query before UNION is an anchor query that will act as our first start from where we will begin iterating.
Here we are flaging the record as 1 so that we can group them together if they fulfill the criteria.

3. With UNION operator now we are iterating for all records by joining the second cte with our first cte on the basis of iterator_id and increasing it by 1 for every iteration..
Here, we have used CASE WHEN to match the required condition and flag them. After flaging them we adding the flag we have already set that is 1. This we are doing so that for second time when it is different record we get 2 as the flag.

Output we get for this is as below

hall_id	start_date	end_date	iterator_id	flag
1	2023-01-13	2023-01-14	1	1
1	2023-01-14	2023-01-17	2	1
1	2023-01-15	2023-01-17	3	1
1	2023-01-18	2023-01-25	4	2
2	2022-12-09	2022-12-23	5	3
2	2022-12-13	2022-12-17	6	3
3	2022-12-01	2023-01-30	7	4

4.Now we simply querying for the records records that is hall_id, MIN(start_date) and MAX(end_date) for each id by merging the overlapping dates.
We are eliminating the overlapping dates by grouping them on the basis of hall_id and flag that we created to group the records.

hall_id	start_date	end_date
1	2023-01-13	2023-01-17
1	2023-01-18	2023-01-25
2	2022-12-09	2022-12-23
3	2022-12-01	2023-01-30

BY MANISH KUMAR CHAUDHARY

THANK YOU

"Success is not about being the best. It's about always getting better."

Behdad Sami