1. There is a table that tracks every time a user turns a feature on or off, with columns user_id, action ("on" or "off"), date, and time.
   In Postgres:
   These SQL-standard functions all return values based on the start time of the current transaction:
   ...
   `CURRENT_TIMESTAMP`
   …
   `transaction_timestamp()` is equivalent to `CURRENT_TIMESTAMP`, but is named to clearly reflect what it returns.

   `now()` is a traditional PostgreSQL equivalent to `transaction_timestamp()`.

   In MYSQL:

   `CURRENT_TIMESTAMP`, `CURRENT_TIMESTAMP()` are synonyms for `NOW()`

   a. How many users turned the feature on today?
      MYSQL:
      SELECT COUNT(DISTINCT user_id)
      FROM TABLE
      WHERE date=CURDATE() AND action LIKE “%ON%”;

      POSTGRES:
      SELECT COUNT(DISTINCT user_id)
      FROM TABLE
      WHERE date=now() AND action LIKE “%ON%”;

   b. How many users have ever turned the feature on?
      Select count(user_id) from table
      Where Action LIKE “%ON%”
      Group by user_id
      Having Count(Action) >=1;

   c. In a table that tracks the status of every user every day, how would you add today's data to it?
      INSERT INTO status_table (user_id, name, status, date_time)
      VALUES(‘chid1234, ‘chidam’, ‘Married’, NOW()); (NOW() - example: 2019-08-09 22:18:11)

2. 9.48 am - 10.10 am

   If 70% of Facebook users on iOS use Instagram, but only 35% of Facebook users on Android use Instagram, how would you investigate the discrepancy?

   Before I begin, I would like to get a few pieces of information to understand the situation better.

   I would like to know the time period during which this happened like the second week of a particular month in a year. In short, I would like to when this happened.
   Where this happened: specific to which geographic location: the US or where?
   Understanding the distribution of the data better: average number of Android users signing into Instagram on a day per week. Understanding this will shed light on whether to consider the situation as discrepancy or not.

   I would approach the situation in the following way:

   1. I will check if the situation should be considered under discrepancy or not. Maybe this is a recurring pattern, say the same week in the last month or last year, whenever IOS releases a new version of OS to download.
   2. At the same time, it's possible that the Instagram app was down on Android phones due to technical difficulties; sync issues. If yes then that clearly explains the situation. If not, I will proceed further.
   3. If there does not exist any clear pattern then I will verify whether there was any glitch in collecting the data.
   4. At this point I understand that there is a glitch and will check if there are any competitors' trends affected the increased use from the IOS users. **For example:** Apple introduced a new feature similar to Instagram or a new photo sharing app launched on the Apple store.

   This breaking down should shed some light on what really happened.

3. 10.20 am to 10.34

There is a messages table showing the messages sent and date information. Find the top 10 users that sent the most messages. Then again the same question but filter the users to the most active users (active after a certain date)

a. Question to ask: most messages sent on a particular date or in a month or amongst all the dates available in the table?

Have to understand the schema. If the distribution of data is like messages sent on different dates in a month then I have to count and then order by in descending order and limit by 10. If it's on a particular date then find I can find the max messages sent grouped by user_id and then order by in descending order and limit by 10. Not sure whether there will be any changes if date falls on a single day or in a month.

1. Max messages sent on a particular date

SELECT user_id, COUNT(message_sent) AS sent_messages
FROM TABLE
GROUP BY user_id
WHERE MONTH(date)=5 for example (if use MONTH_NAME(date)="january")
ORDER BY sent_messages DESC
LIMIT 10;

2. SELECT user_id, COUNT(message_sent) AS sent_messages
FROM TABLE
GROUP BY user_id
WHERE date >=""
ORDER BY sent_messages DESC
LIMIT 10;

(Example: `SELECT * FROM users WHERE created < DATE_ADD(CURDATE(), INTERVAL 1 DAY);)` it's possible to add intervals like this.

4. If you use ORDER BY, what is the default sequence of your result?
Ascending order

5. How to count unique elements in a column of a table?
COUNT(DISTINCT Column_name)

6. How to count unique number of columns in a table?
dbo is the default schema in SQL Server. You can create your own schemas to allow you to better manage your object namespace.

```
SELECT COUNT(COLUMN_NAME)

FROM INFORMATION_SCHEMA.COLUMNS

WHERE TABLE_CATALOG = 'Database name' AND TABLE_SCHEMA = 'dbo' AND TABLE_NAME = 'table name'
```

7. Question to ask: is there a separate column Month or the date in date format

A. What is the revenue by advertisers for the month of March? (From one table)

```
SELECT advertisers, SUM(revenue)
FROM TABLE
GROUP BY advertisers
WHERE MONTH(date)=3;
```

B. 11.36 am
What is the ROI for each advertiser for the same month? (Now there are two tables)
Table 1: date, advertisers, project_name, revenue
Table 2: advertisers, cost of investment (by the advertiser): meaning how much they paid FB to place ads, date
ROI: benefit or revenue/investment

- If the cost of investment also in the same table:

```
SELECT advertisers, (SUM(revenue)/investment.cost_invested) AS ROI
FROM TABLE t1
Join
(
SELECT advertisers, SUM(cost_of_investment) AS cost_invested
FROM TABLE t2
GROUP BY advertisers
WHERE MONTH(date)=3
) investment
ON t1.advertisers=investment.advertisers
GROUP BY advertisers
WHERE MONTH(date)=3;
```

- If cost of investment and revenue in two separate tables: before joining tables you can check whether the tables have null values.

```
SELECT t1.advertisers, COUNT(t1.revenue)/t2.cost_invested AS ROI
FROM TABLE1 AS t1
Join
Table2 AS t2
ON t1.advertisers=t2.advertisers
GROUP BY t1.advertisers
WHERE MONTH(date)=3;
```

8. I had to calculate accepted friendship requests ratio.

Question to ask: friendship request ratio for every user_id?

accepted friendship requests=number of friend requests accepted/number of friend requests sent

Assume a table has the following details and also assume that if a request is accepted only then friend_id appears, otherwise the values will be null for friend_id:
Which means that I have to calculate null values to account for the total value.
User_id
Friend_id: if request is accepted friend_id appears otherwise null values appear
Request_sent:

Scenario 1: To find friendship request ratio for every user_id
SELECT count(Friend_id)/count(Request_sent), user_id
From table request
Group by user_id;

Does COUNT include non-null values:

Example - **COUNT** Function only **includes** NOT **NULL** Values, but Count(*) does include non-null and null values.
Not everyone realizes this, but the **COUNT** function will only **count** the records where the expression is NOT **NULL** in **COUNT**(expression) . When the expression is a **NULL**value, it is not included in the **COUNT** calculations.

COUNT(*) is all rows in the table, COUNT(Expression) is where the expression is non-null only.

If all columns are NULL (which indicates you don't have a primary key, so this shouldn't happen in a normalized database) COUNT(*) *still* returns all of the rows inserted. Just don't do that.

---

9. The SQL question was quite easy. Given a table with detailed customer complaint tickets of different types, calculate the share of processed tickets within each type.
Assume a table with the following columns:

Customer_id, complaint, ticket_num
Complaint: delay (in sending messages), login, sync
Ticket_status: processed, pending

Questions:
1. should I assume there will be only one processed ticket per customer?
2. Is it possible that duplicates exist?

Select Count()/(Select Count(*) AS total_complaint from table) t #`COUNT(*): counts all rows, even null/duplicates`
From table
Group by complaint
Where Ticket_status=”processed”;

10. We have a table called ad_accounts(account_id, date, status). Status can be active/closed/fraud.

A) what percent of active accounts are fraud? (question: does this mean accounts that are active had been fraud or are fraud now? Is it ok to assume some accounts that are currently active were fraud at some time in the past and to calculate the percentage of those accounts?)
B) How many accounts became fraud today for the first time?
C) What would be the financial impact of letting fraud accounts become active (how would you approach this question)?

B) How many accounts became fraud today for the first time?
If the table has millions records then I will create index on status and date and query.
CREATE INDEX date_status_index ON table(‘date’, ‘status’);

Select t1.account_id from table t1
Where t1.date=CURDATE() AND t1.status LIKE “%fraud%”
JOIN
(select t2.account_id AS acc_id from table t2
Group by t2.account_id #i dont think the group by is necessary here
Where t2.date < CURDATE() AND t2.status NOT LIKE “%fraud%”) non_fraud
ON t1.account_id=non_fraud.acc_id;

C) What would be the financial impact of letting fraud accounts become active (how would you approach this question)?

11. simple pandas task to mimic sql filtered query

Simple filtering using WHERE clause:

```
SELECT *
FROM tips
WHERE time = 'Dinner'
LIMIT 5;
```

The same done using pandas:

```
tips[tips['time'] == 'Dinner'].head(5)
```

Just like SQL's OR and AND, multiple conditions can be passed to a DataFrame using | (OR) and & (AND).

```
-- tips of more than $5.00 at Dinner meals
SELECT *
FROM tips
WHERE time = 'Dinner' AND tip > 5.00;
```

Pandas:

```
tips[(tips['time'] == 'Dinner') & (tips['tip'] > 5.00)]
```

https://pandas.pydata.org/pandas-docs/stable/getting_started/comparison/comparison_with_sql.html

https://stackoverflow.com/questions/44537249/grouping-by-with-where-conditions-in-pandas

```
SELECT * FROM (
  SELECT
    t.*,
    RANK() OVER(PARTITION BY sex ORDER BY tip) AS rnk
  FROM tips t
  WHERE tip < 2
)
WHERE rnk < 3

ORDER BY sex, rnk;
```

```
(tips[tips['tip'] <
2].assign(rnk_min=tips.groupby(['sex'])['tip'].rank(method='min')).query('rnk_min < 3')

   ....:     .sort_values(['sex', 'rnk_min']))
```

---

12. Given the following tables how would you know who has the most friends

REQUESTS
date | sender_id | accepter_id

ACCEPTED
accepted_at | accepter_id | sender_id

Answer to the question:
https://www.glassdoor.com/Interview/Given-the-following-tables-how-would-you-know-who-has-the-most-friends-REQUESTS-date-sender-id-accepter-QTN_2520726.htm

Cool links:
https://nifannn.github.io/2017/10/26/SQL-Notes-Leetcode-597-Friend-Requests-1-Overall-Acceptance-Rate/

https://nifannn.github.io/2017/10/27/SQL-Notes-Leetcode-602-Friend-Requests-2-Who-Has-the-Most-Friends/

```
Finally, sort by the number of each id in descending order and output the first
entry:
SELECT t.id, COUNT(t.id) AS num FROM
(SELECT requester_id AS id FROM request_accepted
 UNION ALL
 SELECT accepter_id AS id FROM request_accepted) AS t
GROUP BY t.id
ORDER BY num DESC LIMIT 1;
```

Question: Write a query to find the overall acceptance rate of requests rounded to 2 decimals, which is the number of acceptance, divide the number of requests.

select(IFNULL(ROUND(select count(distinct(requester_id, accepter_id)) AS accepted_requests/select count(distinct(sender_id, send_to_id)) AS requests_sent, 2), 0)) AS acceptance_rate

---

13. How do you calculate monthly active users, churned users and resurrected users from a user activity log with userID and DateTime
https://www.glassdoor.com/Interview/How-do-you-calculate-monthly-active-users-churned-users-and-resurrected-users-from-a-user-activity-log-with-userID-and-Dat-QTN_2541934.htm

Questions: should I assume within a same year

CREATE INDEX date_index ON table(DateTime); because I have to calculate the month from datetime, but I cannot use month() function on an indexed column date_index. Therefore
(some of the questions you can ask before creating index: Is this table has a large number of records? Would we update, delete, or insert records into this table often? How often?)

1. **Monthly Active Users:**
   Select count(distinct userID) AS active_users, MONTH(DateTime) AS month_det
   from table
   Group by month_det;

2. **Churned Users & Resurrected Users**

Select count(case when (`CURDATE()-60 IS NOT NULL) AND (CURDATE()-30 IS NULL) THEN 'churned_users')) AS 'churned_users'`
`count(`case when (`CURDATE()-60 IS NULL) AND (CURDATE()-30 IS NOT NULL) THEN 'resurrected_users')AS 'resurrected_users'`
`END AS user_status`
`From table;`

---

14. Determine fraction of active users messaging at least 5 people in a given day.
Define who are active users here:

Select userid from table
Where activity='messaging'

---

15. Given 2 tables, one with the phone numbers that Facebook sends the confirmation message and another one with the phone numbers that confirmed the verification, write a sql query to calculate the confirmation percentage.

Table 1
Userid phone_num confirmation_sent
1 - yes
0 - no

Table 2
phone_num confirmed_status
Yes, no

Question:
Is it possible? Looks like FB can send many confirmation codes and sometimes it's likely that someone is trying to hack the account.
Are there any null values in either of the tables?

confirmation percentage=number of confirmed messages/total number of messages sent (should the total be distinct or can contain duplicates?) In order to understand that I needed to know the purpose of the percentage calculation.

1. Select the distinct phone numbers in table 2 where confirmed_status='yes'
2. Select the distinct phone numbers in table 1 where confirmation_sent IS NOT NULL.
3. Join the two selections on unique phone numbers.
4. Divide the joined selection by total confirmation messages sent from table 1.

Analysis:

Create index confirm_index2 on table2(confirmed_status);
Create index confirm_index1 on table1(confirmation_sent);

1. Select distinct(phone_num) as p2 from table2

Where confirmed_status='yes';

2. Select distinct(phone_num) as p1 from table1
   Where confirmation_sent=1;

3. **Solution**

```
select(count(
(Select distinct(phone_num) as p2 from table2
Where confirmed_status='yes')t2
JOIN
(Select distinct(phone_num) as p1 from table1
Where confirmation_sent=1)t1
On t2.p2=t1.p1)kite)/select(count(t1.p1));
```