Chapter 13

Interview Hints

DRAFT

1 Interview Hints

In this section we will go over some advice for preparing for SQL interview questions. The end of this chapter contains a number of real interview questions that candidates have gotten when interviewing over the last few years.

One quick note before jumping into the below. Most of my experience and the experience of students that I speak with is industry focused and specifically "tech" jobs. If you are looking at a position in government, non-profit, or even non-tech fields, the information below may not be as applicable. YMMV.

The three most common ways that SQL is assessed during an interview are:

1. **Recruiter oral assessment**
2. **Whiteboard style assessment**
3. **Automated Assessment**

Before doing interview prep in earnest it is important to think about each interview assessment mechanism and how to best prepare for it.

In the first case, a *recruiter* asking a few SQL questions, the purpose is to filter out people who are bluffing their knowledge. In these cases, the interviewer will ask a question or two based off a simple table that they describe. The person being interviewed is expected to respond with something "that sounds about right". The recruiter (generally) isn't writing your code down, they are listening to what you say and comparing it to the solutions that have been given to them.

A whiteboard style assessment is a human evaluated interview style where the interviewer provides the person being interviewed with a series of questions and usually a description of some data structure. While the specifics of this can change a bit the key feature is that you are expected to know the topic before showing up to the interview. Generally speaking some type googling is allowed, but outside of minor syntax (e.g. remembering the order of arguments) it is generally frowned upon. Pre-covid these were all done "on a whiteboard" in an office, but post-covid many of these interviews take place over zoom. Instead of a physical whiteboard, the candidate is expected to share their screen and talk through the problem. You are generally expected to ask questions about the data.

The final common interview tool is an online automated assessment, such as those found on leetcode or hackerrank, though there are a variety of these platforms. In this style of assessment you are provided a prompt (and usually a timer) and sometimes some type of recording / browser lock which prevents you from accessing the internet. You then enter your answers into the online test and submit. Depending on the parameters you may get multiple chances to submit different answers, but most of the time you do not see the response.

So, given the above, how do you prepare?

Recruiter oral assessment

These interviews are easy, the goal is to speak with confidence and get "roughly" the correct answer. The failure mode on this first type of interview is demonstrating a lack of conviction about your knowledge. This is less about what you know and more about knowing a little bit and being convincing.

Whiteboard Interviews

The basic idea is to do lots and lots of problems. In a whiteboard interview you are assessed based on your knowledge, speed and expertise. Having the information at your finger tips is the best way to succeed in these situations. On that note, you'll find some helpful hints on how to review below.

DRAFT

To prepare for this type of interview you need to *stop writing queries at your computer and start writing queries on paper*. This bears repeating: To prepare for whiteboard SQL questions you need to stop writing queries on your computer, it is a crutch that you will not have during an interview.

I recommend printing out the homework assignments, especially the first few, and answering them on pen and paper. This will train you to stop relying on assistance from the computer (no more auto-complete or syntax highlighting) and focus your energy on what you do not know full understand. In the next section I describe a few "levels" of difficulty that candidates frequently encounter.

Automated Assessment

Automated candidate assessment mechanisms are becoming more and more common in technology related fields. As an interviewer, I (personally) find them to be higher variance than traditional whiteboard interviews. However, in situations where the number of candidates for a position is incredibly high relative to the team size I have been tempted to use these.

That being said, if I were a candidate and faced one of these I would do my best, but accept that it isn't a reflection

DRAFT

There are a few ways that SQL interviews are completed. Even pre-pandemic most SQL interviews were done either over the phone (very simple questions), over a video call or via automated systems (like hacker rank or leetcode). There were some traditional white board interviews with SQL, but that was relatively uncommon. Before jumping into the interviews it is very, very important to keep in mind that SQL, while a "standard" is loose and interviewers will often restrict the available features that are available to the person being interviewed. For example, depending on what options are chosen in the automated system, CTEs and analytics functions may not be available.

In Figure 13.1 you can see an example of a former student who got lucky – CTEs were not allowed in their interview but they were allowed through to the next round. In a bit of foreshadowing, the next round of their interview process ended up doing some whiteboard SQL work and they were cut at that stage.

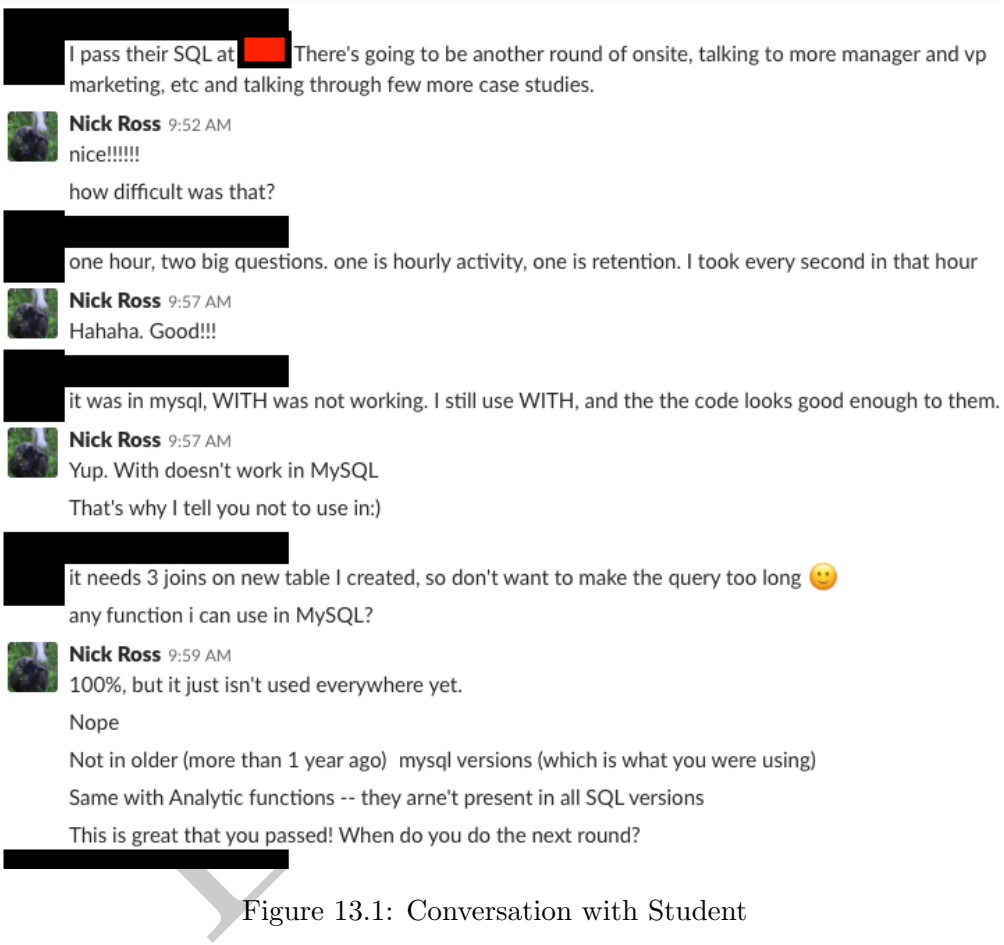

Figure 13.1: Conversation with Student

- **Simple WHERE:** Most questions like this include presenting a table and asking for a query which selects a subset of rows and columns. For example, using the stocks2016 data: write a query which returns all the rows where the volume traded is greater than 10,000. Most of the time, interviewers will not ask specific questions about functions (such as date functions), they are only looking to see if you can write a basic query.

  To prepare for this level of questions:

  - Review the WHERE clause and make sure that you are comfortable answering questions from homework #1. Make sure that you are comfortable writing queries which use SELECT, FROM and WHERE.
  - Focus on making sure you understand basic syntax and how to combine multiple conditions within a WHERE statement using AND and OR.

  - Make sure you understand ORDER BY for sorting data.

- **Aggregations:** Questions like this include computing the MAX, MIN, SUM, AVG and COUNT. For example: Count how many rows in stocks2016.d2010 have volume more than 10,000. The purpose of these questions are to make sure that you understand how to collapse data using a GROUP BY. They may also be testing that you understand that WHERE is evaluated *before* the GROUP BY.

  To prepare for this level of questions:

  - Review homework #2. Focus on the syntax and making sure you understand how WHERE and GROUP BY are applied.
  - Understand how to name columns.

- **Complex aggregations with sub queries:** Questions like this include using aggregations on a single table, but multiple times. For example, how many stocks in 2010 have an average price over $50. To complete this you will need to compute the average price for all stocks and then exclude those with an average less than $50 and then count those remaining stocks.

  To prepare for this level of questions:

  - Review homework #3.
  - Focus on naming and how multiple queries fit together.

- **Joins:** Usually this involves either a LEFT or INNER join. The interviewer is trying to test to see if you know the join syntax.

  To prepare for this level of questions:

  - Review Homework #4 as well as read Lesson #4 and make sure you understand the examples in that chapter.
  - One common question asked when doing SQL interviews is to provide an interview with different datasets and ask how using LEFT, RIGHT, INNER or OUTER command will effect a join.
  - I have never heard of an interview that used a CROSS JOIN.

- **More advanced syntax:** The most complex queries I have seen in interviews ask questions about NULL, HAVING or require the use of a CASE statement. In particular, questions like "how to make this long dataset wide" or, "when we sort, do NULL values come first or last?"

  To prepare for this level of questions:

  - Review the more advanced questions in each homework assignment.
  - Spend time on the practice exams.

Two final points when doing SQL interviews:

First, when doing an interview, make sure that you understand the data before starting to write the query. Ask questions about what columns are unique, which ones have duplicates and what different data types are in each column (if you do not know or they are not obvious). The most common failure, aside from not knowing any SQL, is making an assumption about the data that is incorrect.

Secondly, be careful when writing the query. Write it nicely on the board/keyboard, use space on the board to make the query easy to read. Do not write this:

```
select count(1) as numsales , name from transaction left join salesperson
using( sid ) group by name order by 1 desc limit 5;
```

DRAFT

Write this instead:

```
select
        count(1) as numsales
        , name
from
        transaction
left join
        salesperson
using( sid )
group by name
order by 1 desc
limit 5;
```

2 Example Interview #1

The following questions were given to a student for an interview for a full-time data focused role at Facebook.[1] This was part of the initial screening and was done on a video call with screen sharing set up. The student was required to open a text editor on their screen, which the interviewer could see. The interview then provided the student with information and a set of queries they were required to answer. All of the above information was provided to the person being interviewed.

Note that there were some fairly straightforward warm-up questions (e.g. write a query which returns all rows and columns about a particular user), but those were not recorded below.

The information in the table is about a Facebook product called *Community Translator* which allows people to provide translations or vote on translations. The table is a log of actions that are taken by users and has the following form:

Name	Type	Examples
date	STRING	format - 2019-03-31
user_id	BIGINT	format - 81238123
language	STRING	Arabic, Spanish, Swahili, etc.
device	STRING	2 possible values - desktop OR mobile
action	STRING	2 possible values - vote OR translate

Questions

1. What were the top 10 languages yesterday, in terms of number of unique users who were translators?
2. Return a table with the number of translations completed per language.
3. How many users submitted more than one action yesterday?
4. How many users voted yesterday but didn't translate yesterday?

Answers

1. What were the top 10 languages yesterday, in terms of number of unique users who were translators?

[1] While this *exact* student was interviewing for a data analyst level position, multiple people have coorobated that this is done for data science roles.

DRAFT

```
select
        language
from
        table
where action = 'Translate'
order by count(distinct user_id) desc
limit 10;
```

2. Return a table with the number of translations completed per language.

```
select
        language, count(1) as ct
from
        table
where action  = 'Translate'
group by 1;
```

3. How many users submitted more than one action yesterday?

```
select count(1) as ct
from
(select
        user_id
from table
where date = date(now()) - 1
group by 1
having count(distinct action ) > 1) as innerQ
```

4. How many users voted yesterday but didn't translate yesterday?

```
select
        count(distinct user_id)
from table
        where
                user_id NOT in
                        (select distinct userid from table
                                where date = date(now()) - 1 and action ='Translate')
        and action = 'Vote' and date = date(now()) - 1;
```

Or using a join:

```
select count(1) as ct
from
(select distinct user_ID from table
        where date = date(now()) - 1 and actions = 'Vote') as lhs
left join
(select distinct user_ID from table
        where date = date(now()) - 1 and actions = 'Translate') as rhs
on lhs.user_ID = rhs.user_iD
where rhs.user_ID is null;
```

Or using a CASE statement:

```
select
        count(1) as ct
from
        (select
                user_id
                , max( case when actions = 'Vote' then 1 else 0 end) as m1
                , max( case when actions = 'Translate' then 1 else 0 end) as m2
        from
                table
        where date = date(now()) -1
        group by 1) as innerQ
where m1 = 1 and m2 = 0;
```

3 Example Interview #2

In this whiteboard interview, the interviewee was given the following information on two tables:

- **orders table:** Contains information on orders and had three columns (order_id (int), vendor_id (int) and order_value (float))
- **returns table:** Contains information on returns and had three columns (return_id (int), order_id (int) and return_reason (string))

Both order_id and return_id were integer ids that incremented on their respective tables. Not all orders had returns.

Questions

1. Top 5 vendors (vendor_id) in terms of the largest order.
2. Return rate (number of returns divided by number of orders) for each vendor. Make sure to return zero as the rate if the vendor has no returns.
3. For each vendor return the most common return reason.

Answers

1. Top 5 vendors in terms of the largest order.

```
select
        vendor_id
from
        orders
group by 1
order by max( order_value ) desc
limit 1;
```

2. Return rate (number of returns divided by number of orders) for each vendor. Make sure to return zero as the rate if the vendor has no returns.

DRAFT

```
select
        vendor_id
        , count( return_id )::float / count(orders.order_id) as return_rate
from
        orders
left join
        returns
on orders.order_id = returns.order_id
group by 1;
```

3. For each vendor return the most common return reason.

   (a) Using Analytic Functions

```
select
        vendor_id, return_reason
from
        (select *
                , max( reason_ct) over(partition by vendor_id
                                order by reason_ct desc
                                rows between unbounded preceding
                                        and unbounded following) as mxct
        from
                (select
                        return_reason
                        , vendor_id
                        , count(1) as reason_ct
                from returns group by 1,2) as iq
        ) as iq2
where maxct = reason_ct;
```

   (b) Using CTE

```
with
        total_return_count as
                ( select return_reason, vendor_id, count(1) as reason_ct
                        from returns group by 1,2)
select
        lhs.vendor_id, rhs.return_reason
from
        (select max( reason_ct) as maxct, vendor_id
                from total_return_count group by 2) as lhs
join
        total_return_count as rhs
where lhs.vendor_id = rhs.vendor_id and lhs.maxct = rhs.reason_ct;
```

   (c) No Analytic Functions, no CTEs:

```
select lhs.return_reason, lhs.vendor_id
from
        (select return_reason, vendor_id, count(1) as reason_ct
                from returns group by 1,2) as lhs
join
        (select vendor_id, max(reason_ct) as maxct
                from
        (select return_reason, vendor_id, count(1) as reason_ct
                from returns group by 1,2) as innerRHS
        group by 1) as RHS
on lhs.vendor_id = rhs.vendor_id and lhs.reason_ct = maxct
```

4 Example Interview #3

This set of questions was given during 2019. Note that these require some knowledge of date functions. There are two tables, each with two columns. Columns with the same name can be assumed to match. Note that it is possible for a person to convert without appearing in the visit table.

- **Conversion Table:** Information on users converting (paying) on a website.
  - **User_id:** The ID of the user (integer).
  - **Converstion_ts:** The timestamp (date and time) of when the conversion occurred.
- **Visit Table:** Information about users visiting an advertising web site.
  - **User_id:** The ID of the user (integer).
  - **vist_ts:** The timestamp (date and time) of when the user visited the site.

1. How many visits did each user have to the website?

```
select user_id, count(1) as num_visits
from
visit
group by 1;
```

2. What were the top-5 users in terms of visits?

```
select user_id
from
visit
group by user_id
order by count(1) desc
limit 5;
```

3. Return a list of users who visited the site today, yesterday and the day before yesterday.

DRAFT

```
select distinct v1.user_id
from
(select distinct user_id from visit where date(visit_ts) = date(now()) ) as v1
join
(select distinct user_id from visit where date(visit_ts) = date(now()) - 1) as v2
on v1.user_id = v2.user_id
join
(select distinct user_id from visit where date(visit_ts) = date(now()) - 2) as v3
on v1.user_id = v3.user_id;
```

4. List the three most recent visit timestamps which occurred before the conversion timestamp. If no visit occurred then return one row with a null timestamp. You can also assume that a user_id has, at most, a single conversion.

   There are a few different ways to answer this question, the first is the most general and relies only upon standard SQL syntax. The basic logic is to first exclude all visits which are after the conversion and then join the visit table on itself counting the number of rows after aggregating down on one side. If the number is less than or equal to 3, you know that the number of visits is within 3.

```
select
        lhs.user_id, lhs.conversion_ts, rhs1.visit_ts
from
        conversion as lhs
left join
        visit as rhs1
on lhs.user_id = rhs1.user_id
        and rhs1.visit_ts <= lhs.conversion_ts
left join
        visit as rhs2
on lhs.user_id = rhs2.user_id
        and rhs2.visit_ts <= lhs.conversion_ts
        and rhs1.conversion_ts <= rhs2.conversion_ts
group by 1,2,3
having count(1) <= 3;
```

5 Example Interview #4

This interview was given in 2023 to a student in UChicago's MACSS program. The interview was conducted with the student sharing their screen. Two tables were described in the interview with the column "SID" connecting them.

- **salespeople** contains information on sales people at a company
  - SID
  - name
- **sales** contains information on sales that were made by each sales person. Note that SID is *not* unique as a sales person can have multiple sales.
  - SID
  - sales_amt

1. Tell me the total amount of sales generated by each sales person (name).

```
select name, sum(sales_amt) as total_sold
from
salespeople left join sales using(SID)
group by 1;
```

2. Tell me the names of salespeople who did *not* have any sales.

   Two options. Depending on the size of the table there could be significant performance differences.

```
select lhs.side
from
select sid from salespeople as lhs
left join
(select distinct sid from sales) as rhs
on lhs.sid = rhs.sid
where rhs.sid is null;
```

```
select sid from salespeople where
sid not in (select distinct sid from sales)
```

DRAFT