THEDATAMONK.COM
PRESENTS

WRITE BETTER SQL CODE + SQL INTERVIEW QUESIONS

A book on how to become a Business Analyst

The data monk is there to help everyone know what is data science and how to make a career in this industry. This is one of the youngest industry and with the expansion of internet domain by Internet of Things and Artificial intelligence, the sector will boom in the coming couple of years. If you are reading this book, then you are interested to explore this domain. If you are planning to make a career in data science please spare some time to look into the other books from The Data Monk :-

a. 100 Questions to crack Data Science interview

b. 100 Questions to crack Business Analyst Interview

c. 112 Questions to crack Business Analyst Interview using SQL

d. 100 Questions to learn R in 6 hours

e. How to start a career in Data Science

f. Web Analytics – The way we do it

g. Top Interview questions and all about Adobe Analytics

h. Complete Analytical Project before Data Science Interview

i. Passion is a myth, Chase Opportunities

j. 100 Hadoop Questions to crack data science interview: Hadoop cheat sheet

k. Business Analyst and MBA aspirant's complete guide to Case Studies

These books will prepare you for the worst of all interviews. Go through them. If you are not financially sound to afford all the books, do write to contact@thedatamonk.comfor your free copy. We are here to help ??

SQL is the heart and soul of a data scientist and a going to be business analyst. You don't only need to know how to write SQL queries but you also have to make it look good. Its easy, in fact very easy to write a query, but writing a query which is easily understandable and which will never fail, no matter what the exception is, this is difficult.
The Data Monk as you know is a place where we aspire to create an army of awesome data scientists and business analysts. We work with some of the domain experts and this book is written by one such author.
The author of this book has been working on SQL for last 5 years and is an Oracle certified SQL developer. He passed Oracle 1Z0-051 and 1Z0-052 with 94 and 93% respectively. We vouch that after going through the book, you will start looking at your code in a different way. You will start judging codes written by others and will try to better every code you come across with.

Remember SQL is not just about writing queries, in fact this is the very first step in the business

I assume you know nothing Jon Snow !!

The book is divided into 3 parts.

1. The first part deal with all the resources you need to know to practice SQL queries. It includes some of the important command which you need to master
2. The second part deal with all the important concepts and ways to write a better SQL query. If you have followed the first part religiously, then the second part will make you more mature. You will start dealing with the codes in a different way.
3. The third part will contain some of the most important interview and tricky questions. This section will definitely enhance your skills

P.S. Apart from the second section, we may or may not be following what we are teaching in the second section. This is just to make you look at the code in order to judge your understanding. Go through the book once and when you read it back, try to better the codes. Send us the changed code and win a set of 4 books from thedatamonk.com.

Mail - contacts@thedatamonk.com

Part 1 – Master the skill of SQL

Let me give you a road map for how to start SQL and what are the better online resources to learn this language. In case you have studied SQL before, jump to the writing better SQL queries section .

What is SQL – SQL stands for Structured Query Language. **SQL** is used to communicate with a database. According to ANSI (American National Standards Institute), it is the standard language for relational database management systems. **SQL** statements are used to perform tasks such as update data on a database, or retrieve data from a database.

“ I want to learn SQL as asked by my Manager. How much time will it take and where to start with?”
The answer is simple. Experts says that it takes 10,000 hours to master a skill. But, this is not the case with SQL :P.

Stage 1 - An average person can learn how to write SQL codes in 2-4 hours.
Stage 2 - An average can learn most of the less used but very useful command like ROW_NUMBER, CAST, Inner queries in 6-10 hours.
Stage 3 - An average person can clear a first level Oracle certification with >90% in ~100 hours.
Stage 4 - An average person can clear a second level of Oracle certification with >90% in ~120 hours.

So, all you need to do is to spare ~230 hours if you want to master the skill and get it certified by the master itself.

“Thanks for motivating me, but I am slow and I give up on things very easily. I will start it today but I can assure you that I won’t complete all the steps”
I have been a student and I am always a learner. I would like you to imagine the advantages you get after completing the first 2 stages i.e. you can confidently write SQL queries. After completing the next two stages, you can apply for SQL developer jobs right away. With just ~230 hours of efforts, you can become a certified professional and crack interviews.

To ease you out, We have a book on Amazon which will help you get through the first 2 stages.

“100 Questions to Crack Business Analyst interview using SQL”

We are also going to launch our set of two books:-

1. Learn SQL the certification way
2. Learn SQL the certification way 2.0

These 2 books will reduce your number of required hours from 220 to 50. You should be able to pass the exams with flying colors without using any dumps or any unethical material. It will be all your efforts, but directional

Stage 1

If you are very new to the concepts, then start with

1. www.tutorialspoint.com
2. www.w3schools.com
3. www.datacamp.com

Once you are done with the basics, please do check our 100 question series to sharpen your skill and to cover interview/tricky questions

Stage 2

Now you have solved 100 interview questions and at least 40 more questions from different websites. The second stage is to get familiar with the most important concepts like:-
1. Row number
2. Partition by
3. Writing subqueries
4. Functions in SQL

If you have time then do go through the sqlzoo website, they have some really awesome questions for practice. Answers are also available for most of the questions on the internet.

Stage 1 and Stage should not take more than 1 day. Maintain a notebook for quick revision. Target to solve ~150 questions which includes 20 basics, 100 tricky or interview questions and 30 concepts from above mentioned

Stage 3 and Stage 4

Its up to you if you want to enhance your skill. Each of the two tests cost you around $130 or Rs. 8500. But I think these are worth every penny as it gives a cutting edge to your resume, but going through the answer dumps and learning the answers won't help you as any ways you have to defend your expertise with the answers to interviewers questions !!

For stage 3 and 4, go through the official syllabus and book. The book is ~500 page long. We will be publishing a very concise version of the book

"Learn SQL the certification way". You can refer to the book for quick concept revision.

How to write better SQL Queries

Now, you know how to write a query. You write a piece of code and save it somewhere after your purpose is satisfied. From here one of the following thing is bound to happen:-

1. You decided to leave the company and you have to document everything. You don't know Jack about your codes anymore. You are doomed !!
2. There is a new resource and he is asking all sort of questions from your previous codes. Congratulations again, you are doomed !!
3. You are explaining something to your boss/manager and he suddenly asks for the code. Tada.. You are gone !!
4. There has been some change in the backend and you query is

throwing either some funny numbers or there is absolutely no numbers. You waste your whole day figuring out that there has been some issue from the regular table from where you were pulling the data

Each of these scenario depicts only one thing i.e. your code was not ready, at least not standardized. Your code was not commented properly or your code failed due to change in input state or any reason. Writing a code is like an art, writing a standard code is like painting of Monalisa, it got to be perfect.

Here we will discuss all the ways to write better queries with all sort of examples.

1. First and foremost thing to learn is the sequence of execution, the sequence in which SQL executes the query.

FROM and JOINS
ON
WHERE
GROUP BY
HAVING
SELECT
DISTINCT
ORDER BY
TOP

Remember the sequence of execution until SQL dies and mind you SQL is far from dead. it's one of the most in-demand skills that you find in job descriptions from the data science industry, whether you're applying for a data analyst, a data engineer, a data scientist or any other roles. This is confirmed by 70% of the respondents of the 2016 O'Reilly Data Science Salary Survey, who indicate that they use SQL in their professional context. What's more, in this survey, SQL stands out way above the R (57%) and Python (54%) programming languages.

2. No extra data pull

An example – You need to pull the data for last one year, but you are pulling the complete data and more the required columns. The result will be increase in run time plus you have to any ways apply filters to get thing correct and if you miss that part then you will end up sending the incorrect numbers to the Client.

So, apply filters in the code itself.

3. Ignore * in SELECT statement, try to keep the number of columns optimum because for every useful row you are pulling, there is an useless column being pulled. Aim to remove unnecessary column names from the SELECT statement in order to pull only the important columns

Don't

SELECT * FROM Clickstreamdata

Do

SELECT visitor_id, date_of_visit FROM Clickstreamdata

4. LIKE statement – Use it carefully

You want to search all the customers whose name is Robert. You wrote a query

SELECT A,B,C
FROM XYZ
WHERE Customer_name like 'Robert'

You got 1566 rows, you are happy and you sent it to your boss. Boss came up with a fiery face.

“Your numbers are not correct, Amit’s numbers are correct”

Now you are wondering why this simple ask was not completed by you. You saw Amit’s code:-

```
SELECT A,B,C
FROM XYZ
WHERE lower(Customer_name) like ‘robert’
```

With this he was able to pull every possible occurrence of the word.

Remember, the aim of this book is to make you a perfect SQL coder and you can’t afford to miss on these.

5. Avoid subqueries, try to use INNER JOINS instead.

Not all the times, but few times sub queries can be eliminated by using INNER JOINS. The beauty about removing the sub queries is that it will reduce your run time. Both ways of writing have their own pair of advantages and disadvantages.

But I try to replace Sub queries with INNER JOINS where ever possible

6. There is no point in pulling the TOP 10 data using a query with multiple joins. If you gave enough attention, you should be knowing that the query will first JOIN and then pull the top 10 data.

So, don’t put a TOP 10 in the beginning of the query expecting that it will return the top 10 value immediately.

7. Remember that an index is a data structure that improves the speed of the data retrieval in your database table, but it comes at a cost: there will be additional writes and additional storage space is needed to maintain the index data structure. Indexes are used to quickly locate or look up data without having to search every row in a database every time the database table is accessed. Indexes can be created by using one or more columns in a database table.

8. Use IN to replace OR in the code.

```
SELECT Name, Roll Number
FROM School
WHERE Roll Number = 1234
OR Roll Number = 1234456
OR Roll Number = 589498
```

Now try the same thing using IN

```
SELECT Name, Roll Number
FROM School
WHERE Roll Number in (1234,1234456,589498)
```

The query become short and simple

9. Use BETWEEN to replace comparison operator.

You want to get all the students who have the date birth between 2014 and 2016

One way to write the query is : WHERE DOB>=2014 AND DOB<=2016

The other and simpler way to write it is DOB Between 2014 and 2016

10. Use temp tables wisely

You can use temp tables in a number of other situations as well. For example, if you must join a table to a large table and there's a condition on that large table, you can improve performance by pulling out the subset of data you need from the large table into a temp table and joining with that instead. This will greatly decrease the processing power required, and can be helpful if you have several queries in the procedure that have tomake similar joins to the same table .

11. Delete and update tables in small batches

Deleting or updating large amounts of data from huge tables can be a nightmare. The problem is both of these statements run as a single transaction, and if you need to kill them or if something happens while they're working, the system has to roll back the entire transaction. This can take a very long time, as well as also block other transactions for their duration, essentially bottlenecking the system.

The solution is to do deletes or updates in smaller batches. If the transaction gets killed, it only has a small number of rows to roll back, so the database comes back online much faster. And while the smaller batches are committing to disk, others can sneak in and do some work, so concurrency is greatly enhanced.

12. You want to check existence of a table, then please don't do COUNT(*)

Time and again I have witnessed my colleagues doing

```
SELECT COUNT(*)
FROM Student
```

When they just want to see if the table is present in the database or not.

```
IF EXISITS (SELECT 1 FROM Student)
```

BEGIN
<Do something>
END

If you have a large table and you are applying count(*) or some other query , then you are troubling the database. When you deal with big data, every query you make will impact the server's performance. So I insist you to avoid unnecessary queries

In other words, don't count everything in the table. Just get back the first row you find. SQL Server is smart enough to
use EXISTS properly, and the second block of code returns superfast. The larger the table, the bigger difference this will make.

13. Use indexing as much as possible

When you need to compare data row by row with a query that can't use an index—like SELECT * FROM Customers WHERE RegionID <> 3 — it's better to rewrite the query so it can use the index. Like so:

```
SELECT * FROM Customers WHERE RegionID < 3 UNION ALL
SELECT * FROM Customers WHERE RegionID > 3
```

If your data set is large, using the index could greatly outperform the table scan version. Of course, it could also perform worse, so test this before you implement it.

I realize this query breaks Tip No. 13 ("no double dipping"), but that goes to show there are no hard and fast rules. Though we're double dipping here, we're doing it to avoid a costly table scan.

14. The Order Of Tables

When you join two tables, it can be important to consider the order of the tables in your join. If you notice that one table is considerably larger than the other one, you might want to rewrite your query so that the biggest table is placed last in the join.

15. Redundant Conditions on Joins

When you add too many conditions to your joins, you basically obligate SQL to choose a certain path. It could be, though, that this path isn't always the more performant one.
The HAVING claus
The HAVING clause was originally added to SQL because the WHERE keyword could not be used with aggregate functions. HAVING is typically used with the GROUP BY clause to restrict the groups of returned rows to only those that meet certain conditions. However, if you use this clause in your query, the index is not used, which -as you already know- can result in a query that doesn't really perform all that well.

If you're looking for an alternative, consider using the WHERE clause. Consider the following queries:

16. Make your code flexible for varying input

What is a good code?
A code which you or your organization can maintain with very less human intervention. It should be parameterized as much as possible. The good thing about parameterization is that even when you change your code, there is less possibility of error.

There are two types of coders

1.
```
SELECT date, name of customer
FROM Student
WHERE date=”29/07/2016”
```

2.
```
SELECT date,name of customer
FROM Student
WHERE date=current_date
```

The good thing about the second code is that even if you miss to update the code, it will execute successfully without error. Try to

parameterize your code when ever possible. It will reduce the manual effort needed while running it regularly. You just need to modify only those part which are absolutely impossible to parameterize.

17. Naming convention of the code and input/output file

A very basic step, but it can cost your successor(next code owner) a lot of time. Have a proper and consistent naming convention all over the code and file. We will deal with this point in detail in coming pages. At most of the companies we have a tradition of using legacy codes. Just imagine how hard it will be to look into a legacy code with output files like

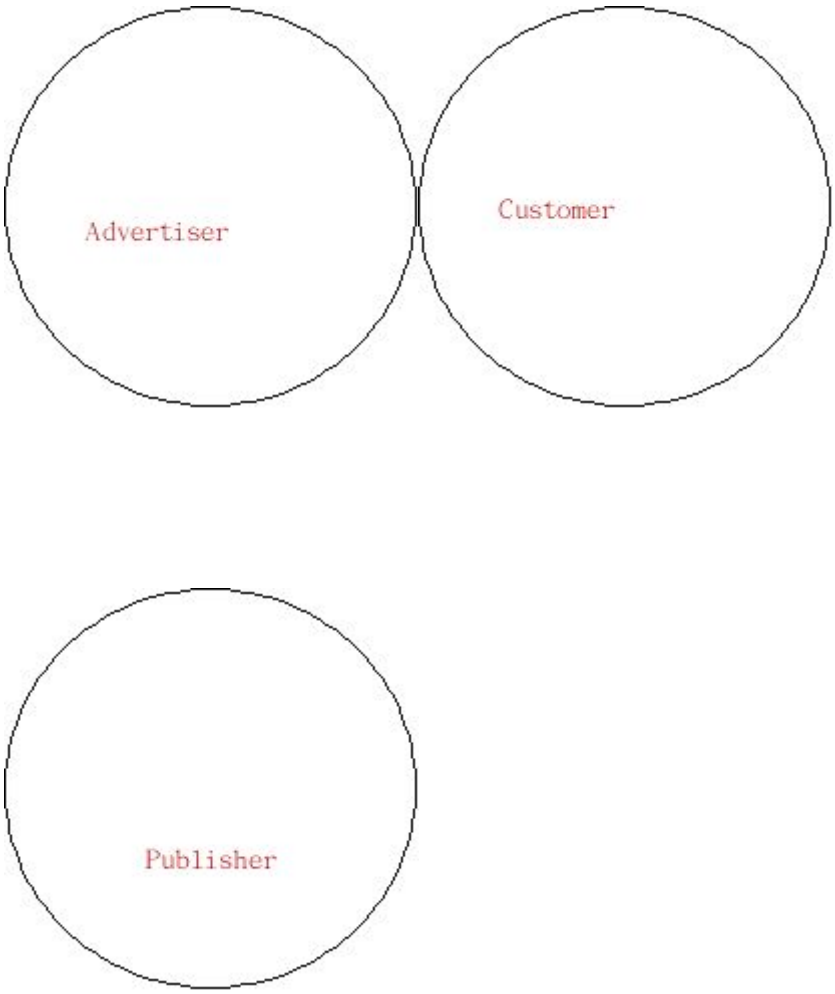
monetization_data_final
monetization_data_final_2
monetization_data_final_final

Poor hygiene more often than not results in inefficiencies like lengthy debugging and code change cycles, refresh errors, etc. - all of which ultimately lead to less time on analysis, and more time on coding and debugging, which anyway leads to more errors

18. Make a free hand diagram of the interaction between different tables and what are you planning to pull from these tables

When you write SQL code, you should be able to visualize the whole process of joins, unions, group by, etc. and it should be in a written format.

Something like this always helps



Here we have 3 tables and each must have some Primary key on which you can join any two tables. The code for the above will be something like this:-

```
SELECT cust.Name, Ad.Advertiser_Name, Pub.Publisher_Category
FROM Customer as cust
INNER JOIN Advertiser as Ad on (cust.Customer_id =
Ad.Customer_Number)
INNER JOIN Publisher as Pub on (pub.Publisher_Id =
cust.Publisher_ID)
```

Using the above diagram you can literally see how you are joining the tables and where are you getting the desired values. This might look simple and unnecessary, but believe me, In my 3 years of data science career, this has been the best way to understand any code and it is always a part of my documentation.

19. Remove duplication from your code

To be a good coder is not a walk in the park. You have to be very particular about your code. It's easy to write 10 codes where each code is duplicating some part of the previous code, but you won't care until you are getting your report done. But when you try to explain the code to some one you will understand the amount of duplication.

Duplication also impacts the performance of the database as you are pulling millions of data for X amount of time which can be done in X/10 number of codes.

Across different adhoc requests, we try to pull the same set of data but on a daily basis.
Example -

Repeating codes while rolling up or aggregating data at different levels is another example of duplication – consider the codes for sales reports where you need to roll up sales at a territory, district, region, area and nation level. Are you writing different codes for all these aggregations? If yes, that is a colossal waste of effort and instead, you should design one comprehensive macro that can run

the sales code across all geography levels by simply toggling a ‘geography’ parameter or macro variable

In every organization there are multiple requests pulling the same data. Suppose you are using many joins (inner, left, right, etc.) and pulling a data every time you get a request. You can easily save your time by either creating a pipeline and automating this process via any scripting language. The ideal output should be giving a date to the function and the function will pull the data for you.
What it does is that

1. it helps the whole team in using one single standard code
2. the team will save the time needed to run the same code again and again
3. Less amount of time loss in Quality Check since you are using a standard code
4. Optimal coding practice as you know this code will be used multiple times across the floor.

20. One function/module for one purpose

This is an important concept to understand before moving forward. I have 8 code snippet and each of them are solving one and only one purpose at a time. My friend Arun is a master coder, he writes one set of code and is able to cover all the data. But the catch is, both of ours code fails more than often but I am able to troubleshoot the code much faster. The reason is, I know that each snippet of my code is solving one specific purpose and when it throws error then it’s easy for me to reform the code. It’s also easy for my colleagues to apply some changes in my absence. Whereas Arun’s code is difficult to trouble shoot and it’s next to impossible for some one else to understand the code.

“Simplicity is the best policy” – I will write more lines in my code but my code will be clean and easy to troubleshoot. No need to prove

your expertise to write a one time masterpiece which becomes a headache for your peers.

A common rule in software architecture is ‘Single Responsibility Principle’2 – implying that each component or module should be responsible for only a specific feature or functionality. The lack of the above implies that there is high dependency/coupling between modules, which means that changes in one module will reflect or impact all neighboring modules, implying manual intervention and damage control.

Example:-

You have to get data from 4 sources, take it into a pipeline and finally roll up the data on a particular attribute. You have two ways to perform this task:-

1. Take data from all the 4 sources and union them. Connect the pipeline in the same code and roll up at the end
2. Write 4 piece of code to pull data from 4 sources. 5^{th} code to merge all the data into one. 6^{th} to align it into a pipeline. 7^{th} to roll up the data

The second way may look lengthy to you, but it will help a lot of time in amending the code on different purposes.

The bottom line is – Try to understand the flow and keep the code in a modular level i.e. one snippet for one and only one process

21. Make you code resistant to input value change

Putting integer as the data type of something which could be a Varchar in the near future.
Taking address as string and putting a limit of 140 character when the number of characters could be more for some one.

The reason why I used these examples is to let you know that your code should be able to deal with change of business rules, input patterns and anything else which can effect the code performance.

Say suppose you have written a code to pull 15 columns from few different tables. Just look for the following thing:-

1. Are any values, anywhere in the program hard-coded - is it absolutely impossible to parameterize it?
2. Is the logic hardcoded? For example, does it expect data to appear only in the 3rd column of a table?

Example:-

1. Change in input time/date variables (Eg: if timeline of report changes by a year, do you have to manually change the date across the report?)

2. Change in input data format (from integer to float) Change in any user defined input parameters (threshold, cut-off, indices etc.)

3. Change in input data dimension/elements – - For example, assume you run a monthly report to track the performance of 5 products pre-selected by the client. The client now suddenly requests that you track the performance of 3 more additional products (a total of 8).
    a. What does this mean for your process? Do you have to manually add the names of 3 products throughout (implying that your program is not input resilient)?
    b. The ideal solution would be that your program requests a one-time user input to take the count and list of products, which is then reflected and accommodated throughout the code.

22. Time complexity of the code?

For queries, however, you're not necessarily classifying them according to their difficulty, but rather to the time it takes to run it and get some results back. This specifically is referred to as time complexity and to articulate or measure this type of complexity, you can use the big O notation.

With the big O notation, you express the runtime in terms of how quickly it grows relative to the input, as the input gets arbitrarily large. The big O notation excludes coefficients and lower order terms so that you can focus on the important part of your query's running time: its rate of growth. When expressed this way, dropping coefficients and lower order terms, the time complexity is said to be described asymptotically. That means that the input size goes to infinity.

In database language, the complexity measures how much longer it takes a query to run as the size of the data tables, and therefore the database, increase.

Note that the size of your database doesn't only increase as more data is stored in tables, but also the mere fact that indexes are present in the database also plays a role in the size.

23. Time complexity of your code

It's very important to understand the time complexity of your code i.e. how much time does it usually take to run one of your code. There are four types of time complexity in the world of coding:-

a. O(1)
b. O(n)
c. O(log (n))
d. O(n^2)

Let's discuss each of them in detail

24. Code with O(1) is not commonly used in the industry as these are mostly for pulling a sample data. O(1) complexity represents a

code which will run in the same time every time it is run. Confused?

O(1) complexity is like a constant time complexity. Suppose you run a LIMIT 10 or TOP 10 in a sample of code. Each time you run this two lines of code, it will take almost the same amount of time

```
SELECT TOP 10
FROM student
```

As a coder you don't have to care about such codes as their performance is fixed in most of the situations.

Length of the code is almost independent of the size of the table in the query. You take a million rows table and pull the top 10 rows and you take a 10,000 rows table and pull the top 10 rows. More or less the amount of time will be the same

25. Code with O(n)

```
SELECT Count(distinct customer_id)
FROM Customer
WHERE purchase_date between 1/1/2016 and 31/1/2016
```

Suppose this code takes 20 minutes to run due to its large size. Now write the following code:-

```
SELECT Count(distinct customer_id)
FROM Customer
WHERE purchase_date between 1/1/2016 and 31/3/2016
```

This code will take some where around 1 hour to run the code.
These type of code have linear relationship with the amount of data which are being hit
The time execution of the code would be directly proportional to the size of the data being pulled or impacted.

COUNT (*) codes will always have a time complexity of O(n) because a full table scan is needed to get to the final number.

A linearly complex code in your code diary should be used carefully as it will hit the database for a much larger time in absence of proper filters.

If you pull all the data of 2016 from the Customer data, then the code will run for 4 hours and you might just need 3 months of data. Be careful with O(n) complex queries

26. O(log(n)) queries

A code with the O(log(n)) complexity is mostly those which needs an index on a particular key to pull the data. The data is present on the leaf node of the table. Take a look on the following code:-

```
SELECT customer_name
FROM Customer
WHERE customer_id  = N
```

Here without the present of the WHERE condition, the code will give you a complexity of O(n), but with this condition the complexity depends on the logarithm of the table. The logarithmic time complexity is true for query plans where clustered index scan is performed.

27. Quadratic time complexity 0(n^2)

The execution time of these queries are proportional to the square of the input size.
Example

```
SELECT *
FROM customer, publisher
WHERE customer.id = publisher.customer_id
```

The minimum complexity would be O(n log(n)), but the maximum complexity could be O(n^2), based on the index information of the join attributes.

These types of codes in your adhoc should be dealt with real care. Select the column on which you want to join the tables on.

28. If the following possibilities are there in the code then there is a lot of scope of modification

a. You have parameterized your code, but despite that you have to change the year part of a weekly refresh in December-January cycle. Then you have to look into the parameterization part
b. If you have to change the input format from X to Y in a new dataset, then your code needs modification to take every possible data type in account

c. If you need to change any user defined parameters over and over again

29. How to keep the ideal input state in your code.

Suppose you report performance of 23 Managers to the organization. Now from next month, there will be 11 more Managers in the organization. Is your code require modification in each of the code snippet ? If Yes, then you need to make the code more flexible towards input

Bad Code

```
SELECT a,b,c,d
FROM Manager
WHERE manager_id IN (123,234,4556,678)
UNION ALL
SELECT x,y,z
```

```
FROM manager_performance
WHERE manager_id IN (123,234,4556,678)
```

Better code

```
SELECT ab,c,d
FROM Manager
WHERE manager_id IN (SELECT manager_id FROM
temp_manager_id)
UNION ALL
SELECT x,y,z
FROM manager_performance
WHERE manager_id IN (SELECT manager_id FROM
temp_manager_id)
```

Creating and maintain a temporary manager id table will help you in saving time from changing the manager id each time a manager is added or removed

This is just a small example of how a better and parameterized code is written which needs very less maintenance and is highly reliable

30. Proper output condition and format

A good SQL code is that code which will not reflect any impact when one part of the output changes. To keep it in a simplified way, suppose you have a website and there is something wrong with one section of the homepage. In this situation, only that section should not work properly, but the website should be in good shape for other part of the website

Same goes in SQL, if you have several output gateways or many output files are being created from a heap of code and something goes wrong in one of the snippet. The code in this case should be so modularized that the non-impacted part should run comfortably.

Do changes in logic or outputs from specific modules or functions within the code snippet impact the overall functioning?
Will the output/failure of one module reflect a likewise change in output/failure of a neighboring module?

If the answer to the above question is Yes, then you need to break the code in pieces and create or re-create a modular framework

31. Readability

Readable code is something which you can understand without the presence of the original author of the code. A readable code is easier to troubleshoot. Readability is not only indentation of code, it consists of the following features as well :-

1. Sensible naming convention of variables, table names and reports
2. Comments – Are you able to understand everything in the code by looking at the comments?
3. Are there snippets of codes which are way too long ? If yes, then the person reading the code might end up getting confused. Try to break the codes in chunks
4. A good indentation practice is a must

32. Try and Catch – Exception handling

We will be presenting a few cases where you might need an exception handler in your code. Exception handling helps your code in dealing with the corner cases. The ‘DivideByZero’ exception – how does your code handle instances of division by zero – is it accurately called out in the logs and pointed out to the user? Does it throw an error and refuse to continue, thus forcing the user to treat it? Or does it carry on with erroneously calculated values?

33. The ‘OutOfMemory’ exception – what if your process runs out of system memory? Does it crash? Does it pause till sufficient

memory is available and then resume? Or worse, does it run while generating/writing half-filled datasets (No errors are thrown, but datasets are generated with lesser rows)?

34. A common instance of poor exception handling is seen in counting null values – Say, the customer level dataset has numerous rows/line-items with blank/null customer_IDs, but with a random age and gender. Assuming the defined business rules indicate that only customers with a valid 8-digit ID should be counted, does your program continue to count blank IDs and classify them as valid customers?

35. Another example could be an SQL stored procedure which calculates a set of only integer metrics could fail due to a string value captured owing to a data capture error

SQL Interview Question

SQL is an easy language to perform basic tasks, but there are a lot of tricky interview questions which can judge the depth of concept. Here we will try to cover some important interview questions before interview

1. How to fetch alternate record from a table?
A. A table in a database contains many rows and columns. There can be a need to fetch only even or only odd rows from the database. Following code can help you with this:

```
Select studentId from (Select rowno, studentId from student) where mod(rowno,2)=1
```

2. What will be the output of the following query and why?

select case when null = null then 'Yup' else 'Nope' end as Result;

A. This query will actually yield “Nope”, seeming to imply that null is not equal to itself! The reason for this is that the proper way to compare a value to null in SQL is with the is operator, not with =.

Accordingly, the correct version of the above query that yields the expected result (i.e., “Yup”) would be as follows:

select case when null is null then 'Yup' else 'Nope' end as Result;

This is because null represents an unknown value. If you don’t know the value, you can’t know whether it equals another value, so = null is always assumed to be false.

3. Write a SQL query to retrieve 10th highest employee salary.
A. This is one of the most asked question in interviews. Try to understand the query for Nth highest employee salary

```
SELECT TOP (1) Salary FROM
(
   SELECT DISTINCT TOP (10) Salary FROM Employee ORDER BY Salary DESC
) AS Emp ORDER BY Salary
```

This works as follows:

First, the SELECT DISTINCT TOP (10) Salary FROM Employee ORDER BY Salary DESC query will select the top 10 salaried employees in the table. However, those salaries will be listed in descending order. That was necessary for the first query to work, but now picking the top 1 from that list will give you the highest salary not the the 10th highest salary.

Therefore, the second query reorders the 10 records in ascending order (which the default sort order) and then selects the top record (which will now be the lowest of those 10 salaries).

Not all databases support the TOP keyword. For example, MySQL and PostreSQL use the LIMIT keyword

4. What is the difference between Rank() and Dense_Rank() function in SQL?
A. The only difference between the RANK() and DENSE_RANK() functions is in cases where there is a “tie”; i.e., in cases where multiple values in a set have the same ranking. In such cases, RANK() will assign non-consecutive “ranks” to the values in the set (resulting in gaps between the integer ranking values when there is a tie), whereas DENSE_RANK() will assign consecutive ranks to the values in the set (so there will be no gaps between the integer ranking values in the case of a tie).

For example, consider the set {25, 25, 50, 75, 75, 100}. For such a set, RANK() will return {1, 1, 3, 4, 4, 6} (note that the values 2 and 5 are skipped), whereas DENSE_RANK() will return {1,1,2,3,3,4}.

5. What is the difference between WHERE and HAVING clause?
A. When GROUP BY is not used, the WHERE and HAVING clauses are essentially equivalent.

However, when GROUP BY is used:

The WHERE clause is used to filter records from a result. The filtering occurs before any groupings are made.
The HAVING clause is used to filter values from a group (i.e., to check conditions after aggregation into groups has been performed).

6. Write a SQL query to transpose a string
A. The following query will transpose “THEDATAMONK” into

T

H
E
D
A
T
A
M
O
N
K

```
Declare @a nvarchar(100)='THEDATAMONK';
Declare @length INT;
Declare @i INT=1;
SET @lenght=LEN(@a)
while @i<=@length
BEGIN
print(substring(@a,@i,1));
set @i=@i+1;
END
```

6. What is the difference between IN and EXISTS?
A. IN:

Works on List result set
Doesn’t work on subqueries resulting in Virtual tables with multiple columns
Compares every value in the result list
Performance is comparatively SLOW for larger resultset of subquery
EXISTS:

Works on Virtual tables
Is used with co-related queries
Exits comparison when match is found
Performance is comparatively FAST for larger resultset of subquery

7. How to get Nth largest salary without using subquery or CTE?
A. The following query can get you the Nth largest salary from employee table

SELECT salary from Employee order by salary DESC LIMIT (N-1),1

8. How to copy data from one table to another?
A.

INSERT INTO table2 (column1, column2, column3, ...)
SELECT column1, column2, column3, ...
FROM table1
WHERE condition;

Apart from these there are few subjective questions which are asked in interviews. I am providing you the questions, follow any book to find answers to these questions.

For more interview questions like this, look into

112 Questions to Crack Business Analyst Interview using SQL

9. What is data manipulation language?
10. What is primary key, foreign key and their difference?
11. What is Row_number function?
12. Defined order by, group by and aggregate functions?
13. What is a view, index and different types of index?
14. What is clustered, non-clustered index in SQL?

A. The clustered index is used to reorder the physical order of the table and search based on the key values. Each table can have only one clustered index. NonClustered Index does not alter the physical order of the table and maintains logical order of data. Each table can have 999 non-clustered indexes.

15. What are different types of joins and what are the differences?
16. What is a trigger?

A. Trigger allows you to execute a batch of SQL code when an insert, update or delete command is executed against a specific table.

Actually triggers are special type of stored procedures that are defined to execute automatically in place or after data modifications

17. What is ACID property?
A.
A = Atomicity
C = Consistency
I = Isolation
D = Durability

18. What is the difference between DELETE and TRUNCATE command?
19. What is the difference between NULL, Zero and blank space?
20. What is COALESCE function?

Going through the basic tutorial stated above will help you in understanding the very basic. After going through the tutorial, go through this book again.

Your ideal path should be something like this:-

1. Online website – w3schools, tutorialspoint, etc.
2. Writing better SQL code
3. 100 Questions to crack Business Analyst Interview using SQL
4. Learn SQL the certification way 1.0(coming soon)
5. Learn SQL the certification way 2.0(coming soon)

Number of hours – 120
Investment – Rs. 800 / $12

This is all you need to crack a SQL interview or master the subject

Thanks,
TheDataMonk.com