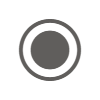
**Interview Invitation – Technical Discussion \_ Muhammad Rehan Malik – Data Engineer (Google US)-20250324\_100711-Meeting Recording**

March 24, 2025, 2:07PM

30m 11s

 **Raheem Khan** started transcription

 **Rehan** 0:03  
The data engineering goal I have been working as a lead data engineer in most of my in the latest few roles in data engineering.  
So I have good experience in AWS as a cloud platform. I've also worked on Google Cloud, but my major experience is within AWS and Google.  
I've explored the query, dataflow and all of this, but as you all I have some more experience but my major experience within data engineering.  
Is around building data pipelines. Distributed systems like I have over extensively on data warehouse migration from on premise to cloud based Data Lake be it on S3 be it on big query and then also I have work extensively using programming languages like Python And Java, Scala also.  
Work, but on awake on a couple of projects majorly for the Spark pipelines build. I've used Scala in my recent in my latest experience latest job I have.  
I'm working with the Baptist.  
Baptist is in the health domain and I'm working as a senior data engineer or I'm also playing the role of an architect where I'm designing the data platform for Baptists. In terms of the data lake in terms of the data quality frameworks and also in the data line.  
And all of these data platform capabilities, so my experience.  
Is both around building data platforms and as well as data pipelines in the.  
Data like.

 **Ramakrishna Medisetti** 1:33  
OK.  
That's really great.  
And GCP side how long?  
How many years you have done it?

 **Rehan** 1:39  
So I would say around four to five years, but yeah, I'm it's it's been on and off with GCP and AWS to be honest.  
But yeah, I wouldn't say I have not worked, but I have worked.

 **Ramakrishna Medisetti** 1:46  
OK.

 **Rehan** 1:49  
But it's been on and off, so I would say right now I'm the most closest to AWS.

 **Ramakrishna Medisetti** 1:55  
OK.  
That's great.  
Like let's say, OK.  
You're designing an ETL pipeline which process the large amounts of data and batches.  
So how do you design this model?

 **Rehan** 2:10  
Yeah. So for building pipelines that you know process large volume of data, I have majorly used spark.  
I have both used spark with Scala as well as Spy Spark. So in case of data pipelines I've used EMR AWS EMR which is the lasting MapReduce service by AWS and this service basically helps you run your workloads spark workloads on a EMR cluster. I've used both of.  
Flavors of it.  
One is the serverless and the provisioned cluster as well which is.  
Cluster and this pipeline would actually based on the kind of processing we are doing like for a data Mart or a data warehouse we have been building layers like the landing layer, the staging layer which is the bronze, silver and the gold layer, right, based on the data.  
Model we perform the we calculate the metrics and we also capture the dimension data right from multiple sources. So we connect to multiple data sources using Spark.  
Capture the data from some data sanitization data cleansing, and then apply the business rules to build the dimensions and facts.  
And this is all done as part of the Spark pipeline.  
So these are all pyspark pipelines that are there and upon, yeah and finally the final data is being actually fed to this one redshift.

 **Ramakrishna Medisetti** 3:17  
OK.

 **Rehan** 3:24  
In our case, I've also used Bigquery.  
We have one of the projects that I've worked on uses the query to load the data for the presentation into a big query.

 **Ramakrishna Medisetti** 3:34  
OK. And what about the file storage systems, right?  
Like when you're extracting, what are the different extraction methods? I mean, right?

 **Rehan** 3:42  
So I've used majorly.  
I've used parquet, but I've also used the Delta file format where we are storing data in the form of delta so that the time travel and the merge operations are bit easier. With delta parking. It's a bit tricky, right?  
So I have used parquet as well as delta and lately I've been trying some Poc's on iceberg but not used it in a project.

 **Ramakrishna Medisetti** 4:02  
And what about scheduling?  
Scheduling as is very important, right?  
Like the.

 **Rehan** 4:05  
Yeah, scheduling of the pipelines. I have major use airflow which was installed on a Kubernetes cluster in my case and we were using airflow to all these pipelines.  
So we had a wrapper script, a runner script which would actually help you run your pyspark job.  
And these runner scripts were the entry points for you to run.  
So based on the pipeline name or the source, we would run these pipelines passing those runtime arguments to the airflow that will build the airflow DAG. All of that was, you know, abstracted.  
As a utility to run as on a scheduling platform a care flow I've used in flow. Sorry in house scheduling tools as well, but within.

 **Ramakrishna Medisetti** 4:37  
Mm.

 **Rehan** 4:46  
The database environment when it comes to serverless, I've used lambdas which were, you know, orchestrated or scheduled by and there was one in House platform which has, which has now moved to provide to you like title. Then it even for scheduling.

 **Ramakrishna Medisetti** 5:04  
OK. And in case of streaming data, right, like what is the approach you follow?

 **Rehan** 5:11  
So for streaming, streaming would be like you're consuming data from a source a producer is producing data, and you're subscribed to a, let's say a topic or a queue, right? Kafka example.  
Kafka is a queue where a producer is producing constant data, right?  
And there's you want to consume data from that.  
So I have user patches Flink.  
To basically build a streaming pipelines.  
To which will subscribe to a topic, a data topic, and which will keep listening to this data topic, a Kafka topic and based on the partitions we have defined, it will read the data from the streaming topic.  
And it's a basically a endless pipeline that keeps running.  
There's no point.  
You have to schedule a real time streaming pipeline so it keeps running and keeps processing the data in a micro batches.  
So in our case the micro batch was so like the TPS, the transactions per second that we were processing were around 100 transactions per second.  
So that was the TPS that we had in our pipeline and streaming.

 **Ramakrishna Medisetti** 6:14  
OK.  
Like what are the key?  
You know, Major, I don't.  
Kind of differences.  
Are you observed between the the streaming and pipe batching pipelines complexity?

 **Rehan** 6:26  
But OK.

 **Ramakrishna Medisetti** 6:28  
Yeah, yeah, yeah.

 **Rehan** 6:29  
So in terms of complexity, batching is more about processing huge volume of data. Streaming is not about processing huge volume of data.

 **Ramakrishna Medisetti** 6:29  
In terms of, yes.

 **Rehan** 6:36  
Streaming is streaming is like you are getting the data you have to capture the data. Maybe perform some simple transformations or maybe aggregate for a very small window.  
Batching is about when you're getting huge volume data and you are doing some data for analytics purpose, right?  
You want to, let's say, populate a dashboard like a tabular clicks in dashboard and you want to see how the sales of a company or a product is going on for.  
By date, by week by month. So that is a use case for batching real time. The use cases much different, right?  
For example, real time A use case would be if a customer has subscribed to a product, right?  
I want that notification that event to come in so that I could trigger marketing campaigns now right?  
If a user has newly come to a product, I would want to send him the welcome e-mail, right?  
So that real time event that happens on.  
A product that needs to be captured and for that capturing.  
I need to have a real time event that will actually trigger my.  
Marketing campaigns or I will do some some XYZ setup, right?  
So all of that is not analytics purpose.  
All of that is basically your day-to-day regular things that regular actions that will actually influence the customer's engagement with the product.

 **Ramakrishna Medisetti** 7:51  
OK. And how it is different with the change change data capture, CDC data replication process?

 **Rehan** 7:57  
So chain data capture is basically about when you read data from a source, right there will be.  
So I mean all this data is getting produced in real time only from the product right now it depends on us what based on the use case as I mention give to give 2 examples right based on the use case whether you are reading this data from the.  
Data Lake so materializer would be typically writing this data from a streaming topic to a location called S3 location or a file system, right?  
And you read from there.  
So when you read data from the source in a batch mode, right when you want to run it, let's say once a day or once a week, now you want to read the latest and the greatest data by latest and the greatest. I mean you want to read.  
All the new data because you cannot read the entire data because the data is getting accumulated in huge quantity, huge volume, right?  
You cannot process the entire data because there's a cost to it, right?  
You cannot read every day billions of rows and of course billions of rows do not change every day, right?  
So change the capture the process by by which you only read the added records. The newly added records or the updated records.  
Or maybe if you have a system which is doing deletes also, you may also need deletes right? Based on certain features, certain flags or delete data whatsoever.  
Right now you read incremental data basically, which may be within present in a partition, right?

 **Ramakrishna Medisetti** 9:11  
OK.

 **Rehan** 9:16  
The You are writing the data each day in a partition in the data lake and you read that partition only.  
So that you read on small volume data and you process upon that.

 **Ramakrishna Medisetti** 9:26  
OK. OK, fine. And let's say like.  
How do you handle a scheme evaluation in data pipeline? Mainly like when dealing with the you know adding new fields to the database.

 **Rehan** 9:42  
OK.

 **Ramakrishna Medisetti** 9:42  
Please yeah.

 **Rehan** 9:43  
So a schema evolution is majorly a contract right between the producer and a consumer, right?  
Let's say a producer would produce some data which adheres to the schema, let's say X, right?  
Which has 5-6 columns or whatever attributes. Now consumer would build this pipeline too. Based on that data, right?  
Based on that schema, now one way is every time you read the data you do a auto schema.  
A read.  
Write that whatever the schema is, you read that data and then you based on what schema you are registered, you validate whether the schema is same or not and based on that you further do the processing.  
That's one way of it, right where usually producing consumer comes to a common ground, they sign up their schema, register their schema in some registry and you can then produce, consume or produce based on that schema, right?  
Or there's another way where you at least bring that data to a landing zone, such as like a Dynamodb where you do not need to define your schema or hard make it hardbound that the schema has to be defined, for example in redshift, if you're loading that data.  
In Redshift, you'll have to follow a schema, right?  
But if you're loading the data into a Nosql DB like a Dynamodb, you may not have to define the schema.  
So at least you give users.  
A way to query the data and then finally in the further layers you could then have schema control.  
Right. That's one way of handling schema.

 **Ramakrishna Medisetti** 11:11  
OK.  
Well, in terms of like the pipeline scalability and handling millions of records per day.  
Like what is your approach like? The minimal performance degradation?

 **Rehan** 11:25  
So for batch data, right?

 **Ramakrishna Medisetti** 11:27  
Right, yes.

 **Rehan** 11:28  
Yeah, for batch data, I think all these cloud resources, right example if it comes to EMR also big query has its internal scale up mechanism. So I don't think big query needs to, you just need to define how much compute units does the big query would require, right?  
So one way of scaling up is you define the autoscaling policies right within AWS. There are auto scaling policies.  
Where you could define that OK if the CPU utilization of the memory utilization goes, let's say 70 percent, 80% above, right?  
You scale up, you add more nodes, you add more worker nodes to the cluster, right? So that it can process more data. Because if we spin up an over provision cluster everyday, it adds to the cost, right?  
So instead we add a scaling policy so that it only scales when it is required, when it is, when there's a peak load right.

 **Ramakrishna Medisetti** 12:21  
Apart from auto scaling like, are there any other you know techniques that you for?  
Implementation.

 **Rehan** 12:26  
If there's a data increase apart from scaling.

 **Ramakrishna Medisetti** 12:31  
Partitioning and data sharing.

 **Rehan** 12:35  
Partitioning and again partitioning is again one of the approaches where you read the write partitions right?  
And that's one thing. You partition the data while processing, right?  
But again, it all boils down to if you have multiple partitions, you need to have multiple workers to read those partitions also, right?  
It will not be as easy as we create multiple partitions and then those workers are not able to read the partitions.  
So hardware has to also scale up if the.  
Partitions are increasing.

 **Ramakrishna Medisetti** 13:04  
And what about implementing the data sharing?

 **Rehan** 13:09  
So data sharding, when it comes to storing data, it is important to implement data sharding because that will actually help you distribute your load in terms of. If I talk about database right it will actually help in terms of how you store your data and how it could.  
Essentially become easy for you know to access your data.  
So when you Shard your data a large database, you can spread it to multiple.  
Small chunks, right?  
And those chunks will be distributed across multiple servers, so you eventually you are distributing a load and that actually helps you to improve your scalability and performance, because now only one server is not reading the entire data, it is distributed, right.  
So that actually helps you to read improve performance.

 **Ramakrishna Medisetti** 14:01  
OK.  
Like.  
How do you, you know, design the the data pipeline which ensures you know minimal downtime and fault tolerance in the event of failures?

 **Rehan** 14:21  
Minimal downtime.  
So you're talking about batch or streaming now in this case.

 **Ramakrishna Medisetti** 14:25  
Not batch only.  
Batch you can give me an example for both right batch and streaming in both the cases I will do yeah.

 **Rehan** 14:31  
OK.  
So in batching a typical approach, I will tell you if there's a. If there's a resource issue, right.  
If you're not able to, let's submit a job because there's a source contention.  
Usually we have a retry mechanism.  
In batching, we do not build anything.  
I'm I've not seen anywhere built.  
Like if there's some issue with your processing, you would typically retry, right?  
And your pipeline is idempotent in the sense if it is reading from, let's say, a certain date, the batch data from a certain date. If it fails in between, it will still go on read from the same date, right?  
Unless you, you know, say you.  
So we maintain watermarking.  
Watermarking is what is the last date you have to read data from.  
So if you have done a successful read and a write and process the data write then the watermark will.  
It was a current date, so the next one happens. You read from the current date in case there's a failure, your watermark will not be updated, so you restart from the scratch only like like scratch.  
I mean you process the data which was not there in the target table. So that way your pipeline is fault tolerant, right?

 **Ramakrishna Medisetti** 15:33  
OK.

 **Rehan** 15:35  
You never miss any data.  
There's no data missing.  
There's no data gaps and whenever you start the pipeline, the pipeline will start from the last successful run.  
Till the last successful data it has.  
Loaded.  
In case of real time.  
This is again managed by Kafka, so if you're reading from a Kafka topic, Kafka ensures that the consumer reads when you have a consumer group, right?  
Kafka ensures that when you read your read has been acknowledged by the consumer.  
Well, if that read from a partition has not happened, Kafka will not let it.  
You know, I should say not.  
I would.  
I wouldn't say delete, but it would take care of. Take care whether your read has happened or not, right?  
It will not be like.  
Your application went down for a few seconds and Kafka messages are lost.  
So it's a queue mechanism, right?  
So you read all the messages. So Kafka ensures that it's a one time. It ensures onetime delivery.  
Successful delivery, right?  
So that's done by Kafka.

 **Ramakrishna Medisetti** 16:41  
OK. OK, fine. Now on the SQL, right?  
How strong are you out of five? How can you rate?

 **Rehan** 16:49  
After five, I would say 4.

 **Ramakrishna Medisetti** 16:51  
OK.  
Let's see you have a query, right?  
With like multiple you know joins, joined operations and millions of records and which is running slowly. How do you optimize this query?  
Like what are the different, you know, techniques you follow?

 **Rehan** 17:08  
So I think based on the type of database where the query is running right, I would first look into.  
On what keys is the data table being joined, right? So the keys are joined. Let's say if I'm looking at redshift right, I would typically see with the two tables are being joined, other dimension key or not, right?  
If I if they're joined on dimension key which is similar to a primary key in a traditional database, right?  
So you would be able to expect that OK, the same data, same keys are colocated when the data was stored by the database system, right?  
Or the Ledger note.  
So that way that's one aspect of it.  
You check whether the data is Co located.  
When you're doing joins again.  
When you're joining with two join 2 tables, if we could also see whether one of the tables is a smaller table. If it's a smaller table.  
It makes sense to, you know, distribute that broadcast, that table on all the nodes, so that there's less of data movements across the machine side, so that also will improve the performance. Then if you're doing some filter filtration, there's a filter clause where clause in the query, right?  
On whatever conditions there's a where clause. We could also go and check whether there are indexes or sort keys. In case of redshift are defined in on those columns so that your data search is faster.  
Your table is able to database is able to search that data. It does not have to do a full scan.  
Of the data, so that will eventually help you improve the performance.

 **Ramakrishna Medisetti** 18:31  
OK.  
What about explain pen? Like? Have you ever used explain plan?

 **Rehan** 18:34  
Yeah, explain plan is a way where you could see.  
What is a logical query plan of the query?  
So when you write the query, the database would actually generate a logical plan and a physical query plan that what kind of join operation.  
Whether it's doing a full join or a full table scan or a sort more joiner hash join.  
So there are multiple types of joins in terms of database terminology sort merge has joined. Then even the type of scan that.  
Did server is doing on the database?  
On the table, right, whether it's a full scan, if it's a full scan, then partition pruning is not happening right?  
So we need to see whether the table.  
Is getting the partitions are getting pruned.  
We are only reading the right partitions.  
Example, if I have a query which is filtering data by let's say a month.  
Or it's a certain date I should see that I should only go and read those partitions. I should not do a full read of the entire data, right?  
That will make the queries go slower.

 **Ramakrishna Medisetti** 19:30  
OK.  
Let's say you need to build a new report, right?  
Which aggregates the data from. You know like sales tables from the multiple regions.  
All right, which includes the the specific time periods.

 **Rehan** 19:46  
Mm hmm.

 **Ramakrishna Medisetti** 19:52  
Like in order to build the efficient no?  
Kind of a query like which would be like a large volume value of data.  
Like what is your approach to building this creating this report?  
You got my question then.

 **Rehan** 20:07  
Yeah, yeah, yeah. I got a question.

 **Ramakrishna Medisetti** 20:08  
Yeah.

 **Rehan** 20:09  
So most of the logic in terms of processing, if you are doing heavy computations. So we usually do those computations as part of the APL layer only, right?

 **Ramakrishna Medisetti** 20:14  
Right.

 **Rehan** 20:18  
So whatever metrics we want to calculate, let's say you want the sales by region, then by week by quarter or by month whatsoever, we do all these calculations in the fact layer only in the metric layer and we store that data whatever we need in a reporting table.  
Called as let's say in the redshift or a big query.  
So that when you do, when you're doing your reporting, you only read the data which is computed precomputed so that.  
These heavy computations are not done by the reporting layer.  
Which is click sensitive now in terms of when you have all of this data. Usually we store the data by their keys surrogate keys, right?  
You join the data with the dimension table.  
Now that dimension table should be.  
A broadcast should be broadcasted on all the nodes in Rachel.  
Let's say so that ratio is also able to do a faster join, right?  
And again, having the right dimension keys having the right sort keys is all important for you to have a faster refresh.  
We are reporting layer.

 **Ramakrishna Medisetti** 21:22  
OK on Python right?  
How can you rate yourself out of right some.

 **Rehan** 21:29  
89890 Sorry out of 544, yeah.

 **Ramakrishna Medisetti** 21:33  
OK.  
Like, how do you handle missing data in data frame using pandas?

 **Rehan** 21:39  
We have a fill NA method which we can use to fill null nulls.

 **Ramakrishna Medisetti** 21:45  
OK. What are the other methods like?

 **Rehan** 21:48  
I mean, we could.  
Other than that, I think we could use collase in our data frame like you if the value is null, it will replace it with some default value, so that's also one way.

 **Ramakrishna Medisetti** 22:00  
OK. Drop in. Yeah. Drop now any.

 **Rehan** 22:02  
Top N is 2. Remove those duplicate records. But if you want to fill with some value default values we use fill in it.

 **Ramakrishna Medisetti** 22:12  
OK.  
OK, Alex, how do you, you know perform the vectorizer operations with Numpy to improve the performance of your data manipulation task?

 **Rehan** 22:23  
To be honest numbai, I have not used much in the context of data engineering because NUM again, these are all libraries which are very their use cases are very little, very few right.

 **Ramakrishna Medisetti** 22:26  
OK.

 **Rehan** 22:34  
Mostly pandas is the one. If you are dealing with smaller. Otherwise Sparc is the one, yeah.

 **Ramakrishna Medisetti** 22:36  
OK.  
OK, like to match the OR not two different data frames in pandas, like handling potential complex in the in the column names. What do you do?

 **Rehan** 22:49  
We want to match two data frames.

 **Ramakrishna Medisetti** 22:50  
No, we want to merge merge two data frames merging.

 **Rehan** 22:53  
Merge OK, OK.

 **Ramakrishna Medisetti** 22:55  
Yeah.

 **Rehan** 22:55  
I think there's a merge function in the pandas. If I don't if I'm recalling, there's a union function.  
Also you can union combine the two data sets in a single one. We have. If you have two data frames and you want to merge on a key, let's say right example there is one data called data frame called Employee Table which has all the employee ID and.  
All of that.  
And then the other one has employee ID with salary and you want to join 2 vertically.  
Then you'll use merge function in pandas.

 **Ramakrishna Medisetti** 23:28  
Hey, can you give me like, OK.

 **Rehan** 23:31  
Once again.

 **Ramakrishna Medisetti** 23:32  
Syntax syntax here.

 **Rehan** 23:34  
Syntex would be like Ufa Panda first pandas data frame. You import pandas and then you pandas as let's say APD and then you do PD you have first data from PD or data frame data one and then you have pdr second data frame. You could do PD like.  
Pandas dot merge you pass both the data frames. You pass the condition on which column you are doing it and how you want to do it like on and how.

 **Ramakrishna Medisetti** 24:02  
OK.  
And have you all done any visoration so?

 **Rehan** 24:18  
I've worked on Tableau but quite little.  
Not more into table, yeah.

 **Ramakrishna Medisetti** 24:31  
OK.

 **Rehan** 24:34  
But I can manage though.  
Yeah, Tableau is not something which is very difficult.  
Tabular is manageable I guess.

 **Ramakrishna Medisetti** 24:44  
OK, OK.  
Say you need to write an ETL script in Python that process large datasets efficiently.  
Processing both memory and time constraints.  
How do you do that?

 **Rehan** 24:55  
In Python.

 **Ramakrishna Medisetti** 24:57  
Yes.

 **Rehan** 24:58  
If it's a large data set, I wouldn't recommend to use just Python.  
It would be pyspark that should be a recommended framework.  
Yeah, so pyspark would be the right framework to use here and that is simple as you read the data frame from the source location, data from the source location to data frame and.  
Pyspark data frames could be used with Spark SQL. We could do the processing.  
Join with multiple tables.  
Get the data prepared.  
For dumping into the final table and then finally write the data so you have spark dot write dot data frame spark dot write dot parquet using which you can write data back to S3 or Google Bucket.  
So that could be done.

 **Ramakrishna Medisetti** 25:45  
OK.  
Like what are the different, you know, libraries, strategies you would use to handle the large datasets?  
And like which cannot you know fit into memory during the like a transmission step? Yeah, during.

 **Rehan** 26:03  
Which could not fit in the memory.

 **Ramakrishna Medisetti** 26:06  
I mean, right?  
To yeah, Mukesh should not store in the memory. Yeah.  
During. Yeah.

 **Rehan** 26:14  
I actually didn't get your question.  
So you want to process the large data, but you don't want it to be in memory.

 **Ramakrishna Medisetti** 26:20  
Right, right. Yes.

 **Rehan** 26:22  
Then it's a it's a non spark kind of option processing, right?  
You then insert.  
Load that data into databases, right?  
And then they don't come in to disk, right?  
Or you within even with spark you could write the data to disk only right? That's option.

 **Ramakrishna Medisetti** 26:38  
Honey. Yeah, they.  
Discard wax, OK?  
Like it?  
Yeah. OK.  
Like handling the situations where ETL script is taking too long to process the data. So how?

 **Rehan** 26:59  
That's all performance operation that I mentioned, right?

 **Ramakrishna Medisetti** 27:02  
Do we?  
What are the different performance like use kind of you know?

 **Rehan** 27:05  
So sorting is one of them.  
We could do sorting in terms of spark. We could check whether there's a resource contention there, data skewness. So that's another one.  
Data skewness is one. Then partition pruning is happening or not.  
So we could prune the partitions. We could do a pushdown predicate like push down the predicate logic onto the source layer so that if the source layer could filter that data and we only read the required data from the source layer into memory using spark.  
So these are few of the opportunities that we do for optimization as part pipeline.

 **Ramakrishna Medisetti** 27:38  
Like parallel processing and like a data.

 **Rehan** 27:41  
Park, KS parallel processing.

 **Ramakrishna Medisetti** 27:44  
OK, optimize the data loading.  
Yeah, profiling the script.

 **Rehan** 27:47  
Yeah.

 **Ramakrishna Medisetti** 27:52  
Like a debugging right? OK how what?  
What approaches you follow in like debugging the scripting Python?

 **Rehan** 27:59  
Debugging in Python right? So.

 **Ramakrishna Medisetti** 28:00  
Right, right.  
Which runs out of the memory. Yeah, yeah.

 **Rehan** 28:03  
Yeah. So we we have IDs, right?  
Like Intellij, IF I'm running some code on Python right, I can use a debugger tool in Python to see or what exactly are the logs? First of all, if it's failing our production, I would go and see the logs in the production environment.  
What are the logs?  
And then trace it back from there, to which part of the code is failing, and then to to replicate the issue locally I could use a debugger, get the same data, try to, you know, replicate that situation locally and then try to fix the code.

 **Ramakrishna Medisetti** 28:35  
OK.  
Like the you know Python script that process the data you know which contains some redundant insufficient code, right?  
How do you know?  
Refactor the code and improve the readability our performance.

 **Rehan** 28:56  
So readability is more about you add docstrings to the Python methods and all you can have document stubs written for each of the Python modules and then there are tools like for lending the Python code. Right? There's black, there's isot.  
You could use all of them to make your code better.  
Follow some standards.  
If they're not following, so that actually helps you remove unnecessary imports, maybe define the type of the variables used, right?  
And then documentation, something which mostly you could leverage Nai over there to document the code. But that's something which you could add docstrings to the Python code classes.  
That improves readability.

 **Ramakrishna Medisetti** 29:41  
OK.  
Yeah, alright. OK.  
Uh.  
Right, Rehan, right. So I'm done with my.  
Questions. So do you have any?  
Anything to ask?

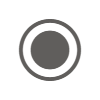
 **Rehan** 29:59  
No, I think I'm good.  
For now, I'll wait for the feedback and.

 **Ramakrishna Medisetti** 30:03  
OK, OK.  
That's great.  
Yeah. Thank you for your time today.

 **Rehan** 30:05  
Cool. Thank you. Bye.

 **Ramakrishna Medisetti** 30:07  
Bye.

 **Raheem Khan** 30:10  
Julia, thank you.  
What is?

 **Raheem Khan** stopped transcription