Tell me

1. Right now, I am a senior data engineer with capgemini, I am working for the client VV. Earlier I worked with different clients such as Capital One, RCCL, elliemae. As a part of Capgemini COE, I provided services to different clients such as Mcdi, HP, GE, AIG, J&J
2. I have been working with data warehousing and Big Data areas right from the start of my career
3. I started with datastage, worked extensively with talend & ETL, recent years more focused on Spark , Big Data and the cloud areas
4. I worked with on Premise and both the cloud envs AWS , Azure
5. I have good proficiency in developing, testing performance tuning
6. Implemented SCDs almost all the datawarehousing concepts, Big Data Batch loads, real time data processing and event driven data loads.
7. I handled data from different types of data sources

Message Brokers such as Kafka, ActiveMQ,Azure Event Hubs, Queues

Files such as parquet, xml, json,excel, fixedwidth, delimited, csv

Various databases such as hive, redshift, postgresql, dremio, hbase, sql server, oracle, my sql etc

Cloud Areas AWS S3, ADLS, HDFS, Blob, Salesforce Objects, Platform Events, Push topics

1. Developed stored procedures, functions, views, complex sql queries etc
2. Worked with version controlling softwares such as Git, bitbucket, Nexus etc for the CICD pipelines.
3. I worked with Open source and enterprise versions of different technologies
4. Worked with Advanced concepts like Metaservlet API calls, Dynamic Schema etc
5. As and when needed, I developed custom code using unix and python scripts, windows powershell scripts
6. I worked with tableau , ESB apache camel**,** Informatica **,** Data Modeling using Erwin as per the requirements
7. I worked as an individual contributor , team of developers, lead the onshore , offshore teams,

Coordinated with the business and technical teams. I can say I am a good team player .I am a hands on guy

1. I worked with different types of projects, Migration Projects, Customer 360 Dashboards, recommendation engines. Data lake, DI processes I worked some of the projects right from the scratch and some I joined in the middle. So I am good at understanding the long term project development issues at multiple levels
2. I continue developing my skills and knowledge time to time to become more valuable to the company and to complete the projects in time with quality.
3. My prior experience, skills, enthusiasm, passion makes me feel that I can be a fit for this role.

Tell about project

Data Integration , Data Lake , for recommendation engines Kafka , AWS

Peformance Tuning Techniques

1. Having Partitions, indexes, primary keys, uniques keys also affect the performance. It is good to have indexes while extracting the data and removing them while inserting the data. Using DB partitions where volume of data in the tables is more than 2 GB or as per the requirements
2. Store temp data option allows you to choose to store the temporary data onto a directory of your disk instead. Adjust buffer size.
3. Using Tmssql connection also decreases the performance. Using shared db connection wherever required.
4. Implementing parallelism techniques & components like tparallelise, multithread execution, dynamic schema, //elism in queries etc.
5. Using Hash, buffer, bulk components wherever required.
6. Removing redundant components \ amount of code. Like using tmap instead of tfilterrow, tjava, tajavarow, tparallelise, treplicate if tmap has no perf issues. Split the jobs into child jobs
7. Filter rows and fields which not required. Using wild cards like ‘%’ and or reduces the number of records returned, Using Limit ,Appropiate use of sub query, temp tables etc
8. Allocating more memory also increases the performance for technologies like Spark.Also in case of java heap space issues
9. Using/writing to files instead of databases
10. Use the appropriate setting in excel component to reduce the memory consumption
11. Increasing the Commit size Batch size also increases the performance. In case of sql, commiting at regular intervals.
12. Developing the code to create less parquet files
13. Writing queries instead of using components ex tfilterow, tfiltercolumns
14. Removing locks on tables
15. Using reusable pieces of code, routines, joblets, maplets, trunjob, subjobs etc
16. Using SP,views instead of huge queries in some cases
17. Using distinct also decreases the performance
18. Where is more efficient than having
19. Use of exists instead of count increases the performance

IF (SELECT COUNT(1) FROM EMPLOYEES WHERE FIRSTNAME LIKE '%JOHN%') > 0

PRINT 'YES'

--------------------

IF EXISTS(SELECT FIRSTNAME FROM EMPLOYEES WHERE FIRSTNAME LIKE '%JOHN%')

PRINT 'YES'

1. Using joins instead of correlated sub query

SELECT c.Name, c.City,(SELECT CompanyName FROM Company WHERE ID = c.CompanyID) AS CompanyName FROM Customer c

-------

SELECT c.Name,

c.City,

co.CompanyName

FROM Customer c

LEFT JOIN Company co

ON c.CompanyID = co.CompanyID

1. **Do Your Constraints in Application Code Rather Than the Database**

Foreign key constraints are nothing new in relational databases, and the stop orphaned records or duplicate data. Constraints are one of the main features of a relational database, but they also take a toll on performance. You can leave your foreign keys active, but it’s better to perform the logic for data storage in the application.When a foreign constraint rule is violated, the database needs to roll back the transaction and send the error back to your application. It’s better to stay a step ahead and do any logic needed in the application, send the data for storage, and then get confirmation from the database. It eliminates the overhead of rollbacks, and you still stick to your foreign constraint rules.

Foreign keys constraints ensure data integrity at the cost of performance. Therefore, if performance is your primary goal you can push the data integrity rules to your application layer. A good example of a database design that avoids foreign key constraints is the System tables in most databases. Every major RDBMS has a set of tables known as system tables. These tables contain meta data information about user databases. Although there are relationships among these tables, there is no foreign key relationship. This is because the client, in this case the database itself, enforces these rules.

Vaccuming, repairing , reorganizing – Database Tuning—Checking the execution plans etc

<https://www.tutorialspoint.com/sql/sql-database-tuning.htm>

Difference between open source & enterprise versions of Talend

1. Some of the components like Tparallise are anot available
2. Tac is not available. So functionalities which we are having with TAC will not be available.
3. Git, svn integration is avialble with Enterprise
4. Remote execution from studio will not be available with open source
5. Data viewer in studio. We can view data from the components itself. In open source, data viewer is not available
6. //el execution options are not available
7. Advanced concepts like Job script , Metaservlet Api, Dynamic schema , Talend Data Mapper, Talend Metadatabridge are not available with open source

Greatest challenge

1. 70 TB Data , Dynamic Schema , metadata driven framework, 10 million project, Talend Metadata bridge

Challenges faced during work env

1. Multiple challenges
2. Handling huge data we, went with partitions in case of performances
3. Unstructured data issues, unexpected date formats, \n,special characters
4. Data loss happened in case of batch loads with backup data, offset changing
5. Because the data consumed is being used by all downstreams. Our env including dev should be running so that others r not blocked.
6. Sync issues with external tables
7. Data needs are getting increased , so we should update accordingly

Exceeded the expectations

1. Real time jobs ensured there is no data loss with polling script otherwise ESB cost more.
2. Open source softwares in case of budget constraints

Work in innovative way

Rediuced the BLOB>manY instances

Started with small initiative. Made it large.

Real time reporting with open source Talend

how would u pick a technology stack interview question answer

writing to hdfs location creating multiple files vs databases

Develop a data model for uber

Rider dimension , dreiver dim, ride details

Cosmetic

1. Naming subjobs , jobs according to standards
2. Keeping things in orderly fashion instead of keeping randoms

<https://community.talend.com/t5/Design-and-Development/The-differences-between-the-tJoin-and-tMap-components/ta-p/21708>

<https://www.thegeekstuff.com/2014/01/sql-vs-nosql-db/?utm_source=tuicool>

sql, nosql

1. rdms, distributed
2. table based data, data in the form of document , key value, graph , column store
3. predefined fixed schema Examples of such predefined schema based applications that use SQL include Payroll Management System, Order Processing, and Flight Reservations., no fixed schema it is dynamic
4. more ram , more servers. Vertically scaled, horizontally scaled.
5. structured query language, unstructiured
6. oracle, sql server postgresql, redshift etc, mongo db, couch db, hbase etc
7. for high & complex trnasactions, sql is good. Huge loiads of data , nosql is good.

SQL join operation is used to retrieve data from related tables associated to each other through foreign key.

NoSQL database systems follows principle of non relational model and supports flexibility in schema definition since it is intended to offer high performance admist high volume of data stored into database.

<https://www.dezyre.com/article/nosql-vs-sql-4-reasons-why-nosql-is-better-for-big-data-applications/86>

<https://hackernoon.com/database-scaling-horizontal-and-vertical-scaling-85edd2fd9944>

Vertical scaling focuses on increasing the power and memory, whereas horizontal scaling increases the number of machines.

Say if u have a car which can fit in 4 members, but u have 10 members, buying a bigger car comes like vertical scaling, buying another car comes under horizontal scaling

If number is 100, buying a car to accommodate 100 members is difficult and costly similaryly more CPU , Ram

Why would’t they? SQL db was and is suitable solution for many applications in IT world. What has changed around 2016 that makes you feel people should go for some other technology?

My guess is you think of NoSQL databases which may be called ‘mature’ for some time now. Well, they were invented for a reason but the reason wasn’t “to replace RDBMSes”, just to deal with some shortages of RDBMSes in very specific (I mean quite narrow) areas of application. There is a number of reasons you can choose NoSQL over SQL (or along with SQL) but it is relatively small fraction of whole db applications. You should realize that some pros of NoSQL come with costs which usually are lack of features which are standard for SQL. To name a few, normalization (which is 99% cases what you want, not what you would avoid), constrained relations, ACID transactions and query language.

n almost all situations SQL databases are vertically scalable. This means that you can increase the load on a single server by increasing things like RAM, CPU or SSD. But on the other hand NoSQL databases are horizontally scalable. This means that you handle more traffic by sharding, or adding more servers in your NoSQL database. It is similar to adding more floors to the same building versus adding more buildings to the neighborhood. Thus NoSQL can ultimately become larger and more powerful, making these databases the preferred choice for large or ever-changing data sets.

table scan –slect \* from table without index

index scan – select \* from table with index

index seek – select \* from table where id = 1 index on id field

<https://medium.com/xplenty-blog/the-sql-vs-nosql-difference-mysql-vs-mongodb-32c9980e67b2>

<https://www.geeksforgeeks.org/difference-between-sql-and-nosql/>

sql 🡪 ram , cpu etc should be increased, table based, suited for complex queries, fixed schema, RELATIONAL DATABASE MANAGEMENT SYSTEM (RDBMS), Verticlly Scalable

nosql 🡪 servers can be increased, key-value pairs, document-based, graph databases or wide-column stores, not suited for complex queries, dynamic schema, Non-relational or distributed database system, Horizontally scalable

SQL databases follow ACID properties (Atomicity, Consistency, Isolation and Durability) whereas the NoSQL database follows the Brewers CAP theorem (Consistency, Availability and Partition tolerance).

SQL requires that you use predefined schemas to determine the structure of your data before you even begin to work with it. Your data must also follow the same structure, as well, which can entail both significant up-front preparation along with careful execution.

A NoSQL database features a dynamic schema for unstructured data and the data can be stored in many different ways, whether it be graph-based, document-oriented, column-oriented, or organized as a KeyValue store. This extreme flexibility allows you to create documents without first having to carefully plan and define their structure, add fields as you go, and vary the syntax from database to database. It also allows you to give each document its own unique structure, giving you more freedom overall

Another big difference between SQL and NoSQL is their scalability. In most SQL databases, they are vertically scalable, which means that you can increase the load on a single server by increasing components like RAM, SSD, or CPU. In contrast, NoSQL databases are horizontally scalable, which means that they can handle more traffic simply by adding more servers to the database. NoSQL databases have the ability to become larger and much more powerful, making them the preferred choice for large or constantly evolving data sets.

One thing that Oracle stressed was the relationship between objects. They said that all data should be normalized. This means no data should be stored twice. So instead of putting, for example, the school address in every student record, it would be better to maintain a school table and store the address there. NoSQL databases have gotten rid of this constraint, to a certain degree.

Disk space was expensive in the 1970s and so was memory, so normalization made sense. But it can take some time to do a join operation to bring together a record that is stored in different tables into one logical unit. It also requires the overhead of maintaining index files and writing to those as data is added or deleted

NoSQL databases say all that does not matter as disk space and memory are cheap. Proponents of that say it is OK to, regarding the aforementioned case, put the school address in with the student. This speeds data retrieval time and makes coding easier.

Rest API

Soap API

Bulk API

Streaming API

GET, POST

An index is used to speed up the performance of queries. It does this by reducing the number of database data pages that have to be visited/scanned.

* **Covering index:** A type of index that includes all the columns that are needed to process a particular query. For example, your query might retrieve the FirstName and LastName columns from a table, based on a value in the ContactID column. You can create a covering index that includes all three columns.

## What is the cost of having a database index?

So, what are some of the disadvantages of having a database index? Well, for one thing it takes up space – and the larger your table, the larger your index. Another performance hit with indexes is the fact that whenever you add, delete, or update rows in the corresponding table, the same operations will have to be done to your index. Remember that an index needs to contain the same up to the minute data as whatever is in the table column(s) that the index covers.

**Data science** is an umbrella term that encompasses **data analytics**, **data** mining, machine learning, and several other related disciplines. While a **data scientist** is expected to forecast the future based on past patterns,**data** analysts extract meaningful insights from various **data** sources.

**Machine learning** is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention.

Data Analytics -> Information from analysed data

Data Mining 🡪 predicting forecasts

Machine Learning 🡪 Automation of analytical model building

**data** analysts extract meaningful insights from various **data** sources.

They can be used for similar purposes but there's a few distinct differences between the hash and the buffer components.

They both work by storing the result set in memory but the hash components allow you to store multiple hash objects and retrieve specific hash sets. This can be useful if you need to temporarily store multiple result sets and then join them back in some way for example transforming multiple data sources and then writing the data out in a single entry to your target. You can also append the output of one hash to another to write to the same data set.

The buffer components only have a single append only option where multiple buffer outputs will write into the same, shared buffer. This makes it less flexible than the hash components but can still be useful for many tasks.

What the buffer components offer extra over the hash components is that the buffer can be read by parent jobs to send data back up to the calling parent job. This same mechanism is also used if you want to deploy your Talend job as a web service and return data from it as shown in [this tutorial](http://www.talendforge.org/tutorials/tutorial.php?idTuto=38).

Other options in a similar space but more for when you start dealing with amounts of data that can't be processed in memory easily (but need to be fully contained in memory for some reason rather than being iterated on) are to use the [tCache](http://www.powerupbi.com/talend/tCache.html) family of components that I know a few other posters here quite like (although I have yet to need). This works like the hash components but will also spill to disk if needed.

An embedded H2 database could also be ran in memory to provide a similar effect and quite a lot more options but at the added cost of complexity in your job.

 Partitioning Concepts

Partitioning enhances the performance, manageability, and availability of a wide variety of applications and helps reduce the total cost of ownership for storing large amounts of data. Partitioning allows tables, indexes, and index-organized tables to be subdivided into smaller pieces, enabling these database objects to be managed and accessed at a finer level of granularity. Oracle provides a rich variety of partitioning strategies and extensions to address every business requirement. Moreover, since it is entirely transparent, partitioning can be applied to almost any application without the need for potentially expensive and time consuming application changes.

Partitioning can provide tremendous benefit to a wide variety of applications by improving performance, manageability, and availability. It is not unusual for partitioning to improve the performance of certain queries or maintenance operations by an order of magnitude. Moreover, partitioning can greatly simplify common administration tasks.

Tables greater than 2 GB should always be considered as candidates for partitioning.

Partitioning also enables database designers and administrators to tackle some of the toughest problems posed by cutting-edge applications. Partitioning is a key tool for building multi-terabyte systems or systems with extremely high availability requirements.

#### **Partition-Wise Joins**

Partitioning can also improve the performance of multi-table joins by using a technique known as partition-wise joins. Partition-wise joins can be applied when two tables are being joined together and both tables are partitioned on the join key, or when a reference partitioned table is joined with its parent table. Partition-wise joins break a large join into smaller joins that occur between each of the partitions, completing the overall join in less time. This offers significant performance benefits both for serial and parallel execution.

<https://docs.oracle.com/cd/B28359_01/server.111/b32024/partition.htm>

what is the need of data ware house ?

OLTP store data for short time range . better reporting etc

why people are preferring data lake over data ware house?

Date lake is on the cloud

What is the difference between group by and partition

Group by cannot get fields other than group by statement

Sql performance tuning

Parallelism in sql

Query execution plan

ACID properities

Why querying on no sql databases is difficult?

Normal Forms

Explain plan

Data modelling – how to create a data model based on the requirement

Difference between for and while loop

Erwin Video

Cursors in sql

hierarchical data storage. meaning

<https://www.tutorialspoint.com/postgresql/>

convert

lag function sql

triggers

which performance is good clustered or non clustered?

multidimensional database

execution plan

<https://www.folkstalk.com/2011/12/sql-queries-interview-questions-oracle.html>

# Hierarchical Queries- <https://www.folkstalk.com/2011/10/oracle-scenario-based-questions-with.html>

# <https://docs.oracle.com/cd/B19306_01/server.102/b14200/queries003.htm>

Challenging issue

Bit map index

What is md5

Normal forms

Difference between sql and no sql

Difference between hdinsigjt and emr

Difference between unix xhell and windows power shell

No lock

Normalize

Explain Plan

Fact table

Fact tables are always Normalized.

Where as dimension tables are may be normalized or denormalized.

In Star schema, Fact table is normalized and Dimension table may be normalized or Denormalized.

In Snowflake schema, the Fact table is normalized and Dimension tables are always normalized.

I am highlighting what Kimball says here: " Dimensional models combine **normalized** and **denormalized** **table** structures. The dimension tables of descriptive information are highly **denormalized** with detailed and hierarchical roll-up attributes in the same **table**. Meanwhile, the **fact** tables with performance metrics are typically **normalized**. While we advise against a fully **normalized** with snowflaked dimension attributes in separate tables (creating blizzard-like conditions for the business user), a single **denormalized** ?big wide **table**? containing both metrics and descriptions in the same **table** is also ill-advised."

Kimball > First Data Marts > Combined Ways > Data Warehouse  
Inmon > First Data Warehouse > Data marts

Reporting queries often return large historical data sets, and when you join various types of data in a single report it incurs a lot of overhead on standard OLTP systems. Running these queries on exactly the same databases that the applications are trying to use can result in an overloaded system

If the data

-------------------------------

| Id | Value | ColumnName |

-------------------------------

| 1 | John | FirstName |

| 2 | 2.4 | Amount |

| 3 | ZH1E4A | PostalCode |

| 4 | Fork | LastName |

| 5 | 857685 | AccountNumber |

---------------------------------------------------------------------

| FirstName |Amount| PostalCode | LastName | AccountNumber |

---------------------------------------------------------------------

| John | 2.4 | ZH1E4A | Fork | 857685 |

---------------------------------------------------------------------

select Firstname, Amount, PostalCode, LastName, AccountNumber

from

(

select value, columnname

from yourtable

) d

pivot

(

max(value)

for columnname in (Firstname, Amount, PostalCode, LastName, AccountNumber)

) piv;

DB Link

|  |
| --- |
| SELECT DECODE(1,1,'Equal'); |

In this example, the DECODE() function compares the first argument (one) with the second argument (also one). Because the first argument equals the second one, the function returns the third argument which is the string Equal.

The following example returns NULL because the first argument is not equal to the second one.

|  |  |
| --- | --- |
| 1 | SELECT DECODE(1,2, 'Equal'); |

<https://www.youtube.com/watch?v=ITcOiLSfVJQ>

SQL FOREIGN KEY Constraint

A FOREIGN KEY is a key used to link two tables together.

A FOREIGN KEY is a field (or collection of fields) in one table that refers to the PRIMARY KEY in another table.

The table containing the foreign key is called the child table, and the table containing the candidate key is called the referenced or parent table.

Look at the following two tables:

The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables.

The FOREIGN KEY constraint also prevents invalid data from being inserted into the foreign key column, because it has to be one of the values contained in the table it points to.

The data type of a column defines what value the column can hold: integer, character, money, date and time, binary, and so on

Go through w3 schools from bottom <https://www.w3schools.com/sql/sql_datatypes.asp>

**.How to display 1 to 100 Numbers with query?**

**Answer:**

Select level from dual connect by level <=100;

<http://www.complexsql.com/hierarchical-queries/>

regexp\_count ( ) sql

<http://www.complexsql.com/rowid-rownum/>

An execution plan is basically a road map that graphically or textually shows the data retrieval methods chosen by the SQL server’s query optimizer for a stored procedure or ad hoc query. Execution plans are very useful for helping a developer understand and analyze the performance characteristics of a query or stored procedure, since the plan is used to execute the query or stored procedure.

In many SQL systems, a textual execution plan can be obtained using a keyword such as EXPLAIN, and visual representations can often be obtained as well. In Microsoft SQL Server, the Query Analyzer has an option called “Show Execution Plan” (located on the Query drop down menu). If this option is turned on, it will display query execution plans in a separate window when a query is run.

**ACID (Atomicity, Consistency, Isolation, Durability)** is a set of properties that guarantee that database transactions are processed reliably. They are defined as follows:

* **Atomicity.** Atomicity requires that each transaction be “all or nothing”: if one part of the transaction fails, the entire transaction fails, and the database state is left unchanged. An atomic system must guarantee atomicity in each and every situation, including power failures, errors, and crashes.
* **Consistency.** The consistency property ensures that any transaction will bring the database from one valid state to another. Any data written to the database must be valid according to all defined rules, including constraints, cascades, triggers, and any combination thereof.
* **Isolation.** The isolation property ensures that the concurrent execution of transactions results in a system state that would be obtained if transactions were executed serially, i.e., one after the other. Providing isolation is the main goal of concurrency control. Depending on concurrency control method (i.e. if it uses strict - as opposed to relaxed - serializability), the effects of an incomplete transaction might not even be visible to another transaction.
* **Durability.** Durability means that once a transaction has been committed, it will remain so, even in the event of power loss, crashes, or errors. In a relational database, for instance, once a group of SQL statements execute, the results need to be stored permanently (even if the database crashes immediately thereafter). To defend against power loss, transactions (or their effects) must be recorded in a non-volatile memory.

complex sql interview questions

<https://www.toptal.com/sql/interview-questions>

<http://www.hackerrank.com/>

<https://www.careercup.com/>

data engineer uber interview experience

[**https://icpcarchive.ecs.baylor.edu/index.php?option=com\_onlinejudge&Itemid=8**](https://icpcarchive.ecs.baylor.edu/index.php?option=com_onlinejudge&Itemid=8)

[**https://www.codewars.com/**](https://www.codewars.com/)

leetcode

[**http://www.crackingthecodinginterview.com/**](http://www.crackingthecodinginterview.com/)

<https://numpy.org/>

<https://www.python.org/>

[**https://www.hackerrank.com/?utm\_expid=.2u09ecQTSny1HV02SEVoCg.0&utm\_referrer=**](https://www.hackerrank.com/?utm_expid=.2u09ecQTSny1HV02SEVoCg.0&utm_referrer=)

<http://blog.gainlo.co/index.php/category/uber-interview-questions/?utm_campaign=quora&utm_medium=What+is+the+engineer+hiring+process+like+at+Uber?&utm_source=quora>

<https://www.rooftopslushie.com/request/Uber-Freight-Software-Engineer---Backend-Interview-Process-and-Questions-110>

<https://www.rooftopslushie.com/request/Uber-SDE-Interview-Tips-61>

<https://www.glassdoor.ca/Interview/Uber-Data-Scientist-Interview-Questions-EI_IE575263.0,4_KO5,19_IP2.htm>

<https://www.google.com/search?rlz=1C1OKWM_enUS767US767&sxsrf=ACYBGNQVbO9lPtIXmafX05SvxlGYuX407A:1574223555918&q=uber+sql+interview+questions&sa=X&ved=2ahUKEwilquPe9_flAhXhQd8KHULfA3sQ1QIoAHoECAsQAQ>

Data lake stores all the data. DWH retains the data used for the business

<https://www.blue-granite.com/blog/bid/402596/top-five-differences-between-data-lakes-and-data-warehouses>

<https://exagoinc.com/blog/lakes-swamps-ponds-and-other-bodies-of-data/>

A data vault is a system made up of a model, methodology and architecture that is explicitly designed to solve a complete business problem as requirements change.

Data Vault data is generally RAW data sets.  So, in the case of the Data Vault, reconciling to the source system is a recommended for testing. This can be reconciling to the flat-files that arrive or comparing to the source databases.  Sometimes there is no “system” to reconcile, because the data arrives on a web service.  In this case, would suggest storing the data in a staging layer.

<https://www.cloverdx.com/blog/data-warehouses-lakes-hubs-and-vaults-explained>

### **Data Warehousing Interview Questions and Answers**

**Q1). How will you define the concept of Data Warehousing?**

A data warehouse is the data repository that is used for the decision support system. A data warehouse is made up of a wide variety of data that has a high level of business conditions at a particular point of time. In simple words, this is a repository of integrated information that is available for queries and analysis.

**Q2). Define the concept of Business Intelligence.**

Business Intelligence is also named Decision Support Systems that refers to technologies, applications, and practices for the collection, integration, and analysis of business-related information or data.

**Q28). Define the star schema.**

A star schema is used in data warehousing where a single table references a number of dimension tables. For the star schema, all keys from dimension will flow to the fact table. It is very much similar to the ER diagram so named as the Star Schema.

**Q29). Define the snowflake schema.** This is similar to start schema where a single table references a number of dimension tables. These dimension tables are further normalized into multiple related tables. As tables are snow flaked to smaller other tables, it is called a snowflake schema.