1. Select \* from tablename ……………..**fetch all table data**
2. Select \* from database name.schemaname(dbo).tablename ……..**fetech all table data from anywhere like from any database screen**
3. Select count(\*)/count(1/any value) from tablename ……..**no of records in table**
4. Select count(column name) from tablename ……**no of records in column exclude null value**
5. **Null count from columns**= select count(\*) –count(column name) from tablename
6. **Uniq records from column**= select distinct(column name) from tablename
7. **Count of distinct from columnn**= select count (distinct(column name)) from tablename

Note: **count and distinct only used for single column we can not use multiple column at one time for count and distinct**

1. Joins question on single column table by just looking 2 table….inner join, left outer join, right outer join, full join, cross join
2. Set operators: union, union all, intersect, except / minus (oracle)
3. Database diagram for show relation between the tables like FK,PK for writing the join query
4. If we join n numbers of table then we will use (n-1) on condition used in join query
5. Self join- employee manger question
6. **Case statement** =select case when condition then condition else condition end from tablename
7. Date function:

-Select datepart(yyyy, hire date) from tablename ……**specific year fetch**

-Select count(\*),datepart(yyyy, hire date) as year from tablename group by datepart(yyyy, hire date) ………..**year wise count show**

-select count(\*), datepart(mm, hire date) as month from tablename where datepart(yyyy, hire date= year group by datepart(mm, hire date) ……..**month wise count show in specific year**

-select count(\*) , year(hiredate) year, month(hiredate) month from tablename

Group by year(hiredate), month(hiredate)

Order by year(hiredate), month(hiredate) ………**show year wise and month wise count**

-select datediff(yyyy, hiredate, getdate()) from tablename ……..**experince in year**

- select datediff(mm, hiredate, getdate()) from tablename ……..**exp in months**

- select datediff(dd, hiredate, getdate()) from tablename …….**exp in days**

15) – select max(salary) from tablename ………**only max salary from table**

- select \* from tablename where salary in (select max(salary) from tablename) ……..**fetch all details from table whose salary is max**

- select \* from tablename order by desc

where salary in

(select max(salary) from tablename where salary <(select max(salary) from tablename) ………..**2nd max salary from table**

**Sudo column**: dense rank, rank , row number

1. Select \* from

(select \*, denserank() over (order by salary desc) as rnk from tablename) as temp

Where rnk=20 ……………….**show 20th max salary from table**

Where rnk not in (3,5,8) …. **except 3 5 8th max salary from table at once**

With cte as

(select \*, denserank() over (order by salary desc) as rnk from tablename)

Select \* from cte where rnk =20 ………. **show 20th max salary from table**

Where rnk in (3,5,8) ……..**show 3 5 8th max salary from** **table at once**

**Partition by:**

With cte as

(select \*, denserank() over (partition by column name(eg.gender) order by salary desc) as rnk from tablename)

Select \* from cte where rnk =1 …..**do partition based on gender and then display max salary from each gender**

1. **Department wise count:**

-Count(\*), join (if we have more table) , group by department name

**19) max / avg salary of department:**

- max/ avg (salary) , join , group by department name

**20) max salary person from respective department:**

With cte as

(select respective column we want, dense rank() over (partition by column name(eg.gender) order by salary desc) as rnk from tablename) join

Select \* from cte where rnk=1

**21) substring and char index:**

**22) oracle ssms difference:**

**Dual**- dummy column 1row and 1 column

-Select sysdate from dual = select getdate()

-substr = substring

-decode = case statement

-select tablename.\*,denserank = select \*, densrank

-rownum = rownumber

-rowid = -

- is null = nvl

-minus = except

- side by side result display = one above one result display in multiple query run

**23) query optimization:**

-instead of \* used all column name at the time of query run

-if any join takes more timing

-parallel.\*

**PROJECT 1: EMPLOYEE DATA TRANSFER**

**CLIENT wants to move there data from source file to MSSQL**

**Step1**: discussion with business team, development and testing team, data engineering, higher mgt and finalized the requirements

**Step2:** dev and testing team got document as

-HLD (HIGH LEVEL DESIGN)

-LLD(LOW LEVEL DESIGN)

-DDD()

-STM(SOURCE TO TARGET MAPPING SHEET)/ god document

**STEP3:** Dev team started developing code

**Step4**: testing team started writing test plan and test cases

**Project flow:**

1.source file …..types of file?

2.landing area (pre-ELAQA)

3. staging (ELAQA)

4. data warehouse (BDWDQA)

5. Reporting tool

Que:

1.**types of files?** ….csv, flat file, xml file, excel file etc

2. **where is source file?** …………available on QA server

3.**how to access files from server?** ……….by using putty or winscp

4. **who will give source file?** …………..developer or data engineering team …..developer create source file its called as multi data dimensional analysis.

5. **data mock up/ handcraft**: create the dummy data for sometimes by referring column name its called data mock up

6. what is the use of landing area, staging area, data warehouse?

7. job run do by developer by using SSIS using visual studio excecute the package

8. job run : 1. File reader……….check file available in source or not

2. file loader ………….load file

3. file deleter ……..delete file after loading the file from source server (QA)

9. after job run fail what will we do? …………do the analysis

**10. after data come to landing area tester work:**

1. write the test cases and send to client and after review from client get approval from client

2. test cases written on excel sheet

3. overall test cases written with priority

4. structure validation

5. initial load validation

6. delta load validation

11. test case write: daily work for tester

**Templates:**

1. Test case name format: Projectname-test case no-table name-structure validation

2. Pre-condtition:

3. step name (design step)

4. descripation

5. expected result

6. table name

7. load type

8. existing /new

9. status (pass / fail )

9. designer

12. **structure validation need:** to avoid data loss

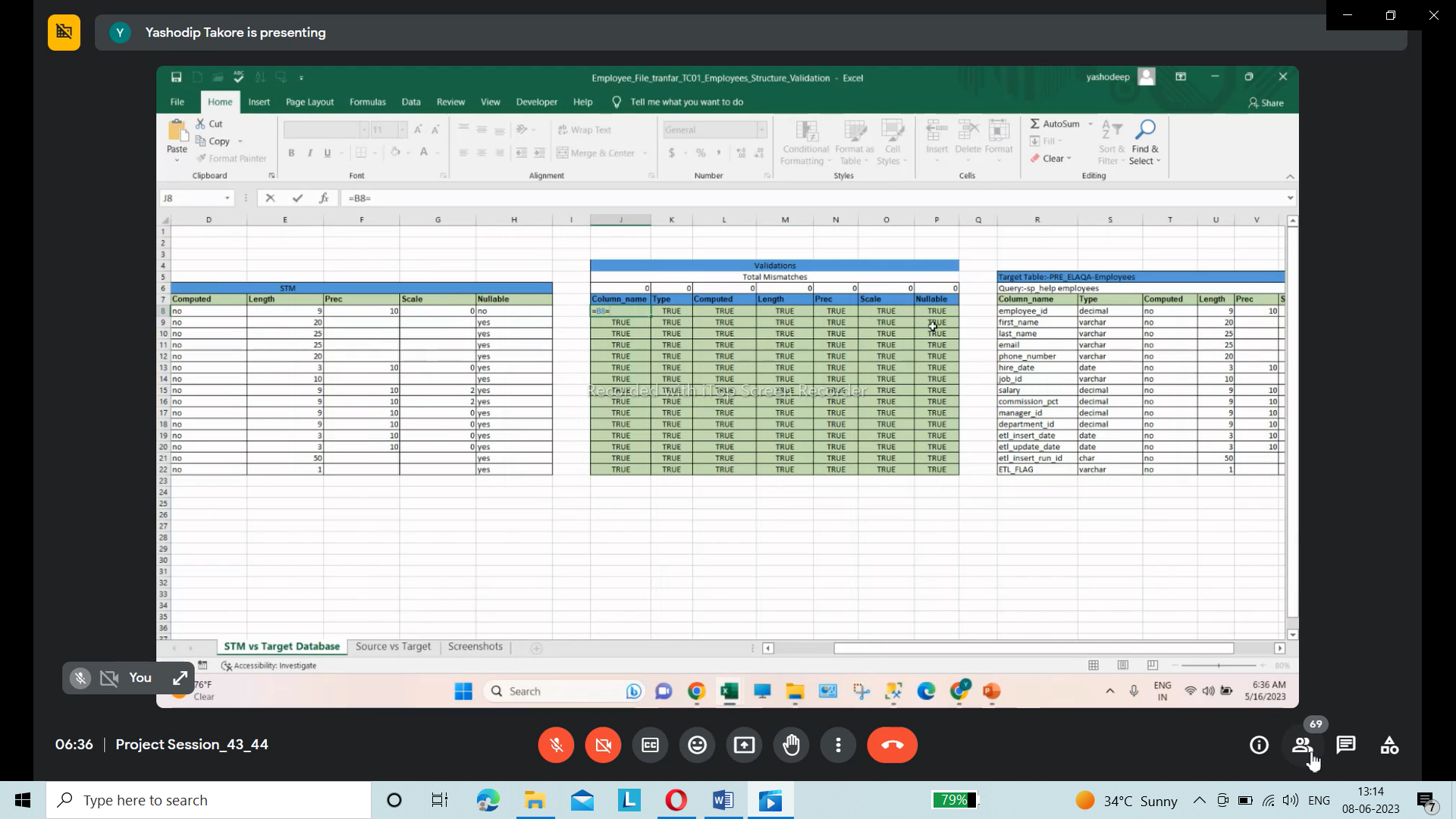
**Table sturucture show query:**

----- Sp\_help table name / select \* from information chema.columns where tablename=’specific column’

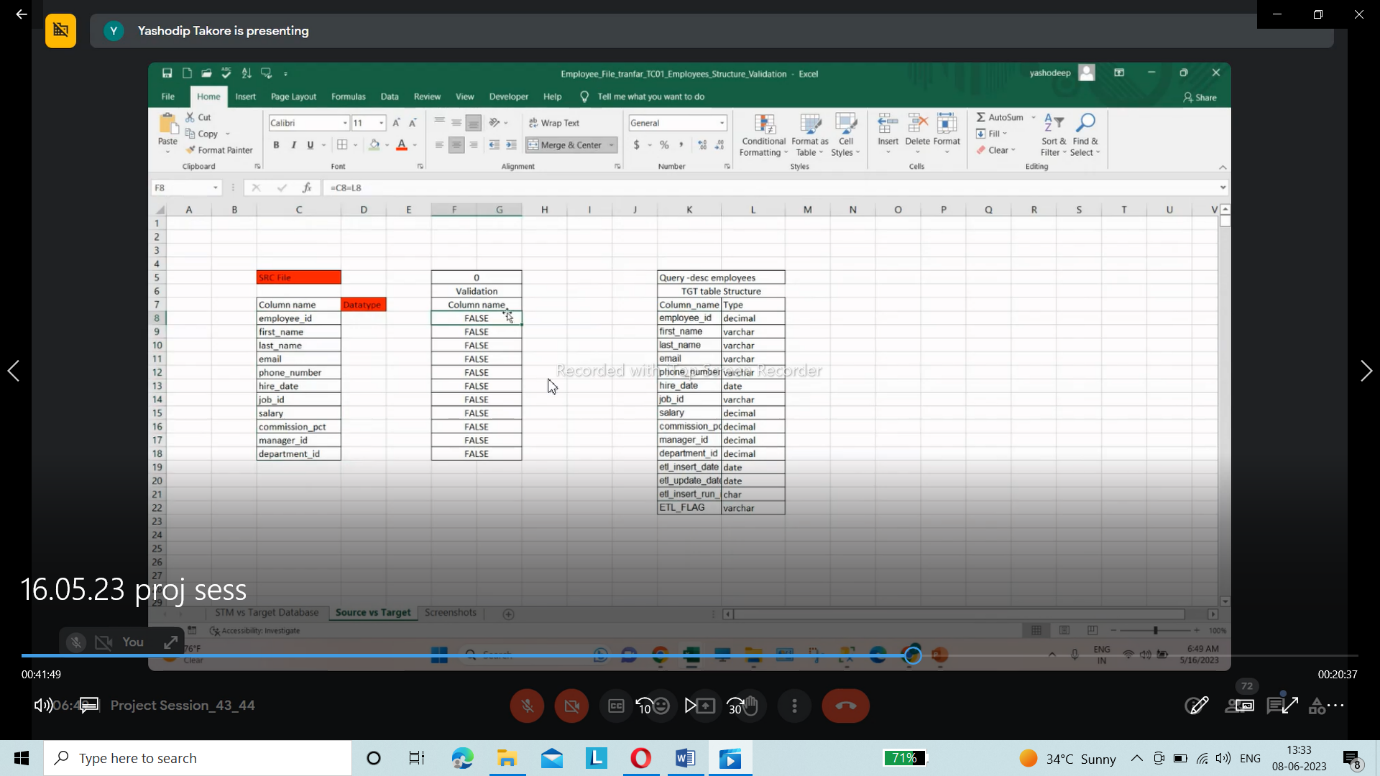
**Step1: STM VS TARGET DATABASE**

--- STM data copy in to excel sheet

---- table structure from target (Sp\_help table name) copy as header and paste in to excel sheet

---- STM VALIADATION TABLE TARGET TABLE 

**--- STEP 2: SOURCE VS TARGET** source file not having data types , constraints that time only validate column name and count of column



13**) INITIAL LOAD VALIDATATION**: data first time load in to the table that is called initial load

-----------Same templates used as structure validatation

Descriopation:

1. **Validate the file row count matches with the target table count**
2. **Validate the target table data matches with source file**
3. **Validate the duplicate records present in the table**
4. **Validate the null values are populating in tgt table**
5. **Validate the control column are populated as per STM**

**SSIS ----ETL TOOL**

**Job fail analysis:**

----after executing package job fail then go to progress tab to see the reasons for job fail and infprm to developer with those reasons

1. **Validate the file row count matches with the target table count:**

---source file data ctl+shift+down get cout from excel sheet

File is in different format like csv, text,xml then converting in to excel as

1.csv ----right click the file--open with excel

2. text ----copy all text and paste in to excel---data—text to column---done

3. xml ----data ---from other sources—from XML data format---done

**Target:**

Select count(\*) from table name

1. **Validate the target table data matches with source file:**
2. Source file: copy data from file an paste in to excel sheet as source records
3. Target : select \* from tablename -----ctl all copy with headers / ctl all save result as csv/excel file----copy this data in to target records for validatation
4. Data validatation do as structure validation in excel sheet

NOTE: If data is more then do sampling data validatation with discuss with client if client is not agree then do process automatation by using the python etc for complete data validatation

1. **Validate the duplicate records present in the table:**

**Target:**

--Select count(distinct(pk column)) from tablename = select count(\*) from table name

**Source:**

Source data validatation tester not involved we just trust on this source data

**Duplicate records find query:**

1. Select count(\*) from table name group by (all columns in table) having count(\*)>1
2. With cte as

(Select count(\*) cnt from table name group by (all columns in table))

Select \* from cte where cnt>1

1. **Validate the null values are populating in tgt table:**

**Target:**

1. Select count(\*) –count(column name not null specific) from table name
2. Validate the null value present column count as per source file count in target table as per above query
3. **Validate the control column/ audit column are populated as per STM**: just want to see any updatation happens in data apart from initial (flag)

**Target:**

Select count(\*) , control columns from table name group by control columns

----show the count of all data with control columns info

NOTE: ----Source file to landing stage data is mismatch observed for millions of data then find out the mismatched data we have to import source file in the sql by creating same table structure and then run minus query then we will get the mismatched data

**LANDING TO STAGING:**

**STEP1**: Structure validatation done like source to landing

**Step 2**: initial load validatation

**Minus query validatation**: same environment required for run this query

----Select all column name excluding control column from target table

Except/minus (in oracle)

Select all column name excluding control column from source table

---- Select all column name excluding control column from source table

Except/minus (in oracle)

Select all column name excluding control column from target table

----select \* from (Select all column name excluding control column from target table

Except/minus (in oracle)

Select all column name excluding control column from source table) temp1

Union all

---- select \* from (Select all column name excluding control column from source table

Except/minus (in oracle)

Select all column name excluding control column from target table) temp

----**Line by line records compare in minus query**

----**minus query used to ensure having same data in source and target so minus query result should be null if its not null then raise issue**

**STAGING TO DATAWAREHOUSE:**

Here source table having multiple tables but data warehouse i.e target having only one table always

---Same stages like above structure validatation, initial load validatation

----**Derived column**: column transform with using some transformation those column in target table called as derived columns

**Count calculation:**

------For source side we have multiple columns so we have to form query (as using join ) to get count for source file

--------target side ----select count(\*) from tablename

**DELTA LOAD VALIDATATION**: 2nd time load the data

Also called as **Subsequent load validatation, incremental load valiadatation**,

**Step1: source to landing**

---update date change, etl flag change from I (initial) to U (update), D (delete)

**Types of delete:**

1. **Soft delete**: delete from source but still available in target and just change the flag in target is called soft delete
2. **Hard delete**: delete data from source as well as target

If soft delete is done then count mismatch happen that time we used query to find out the count from target as excluded delete flag count like

Select count(\*) from tablename where etlflag in (‘I’,’U’) OR

Select count(\*) from tablename where etlflag not in (‘D’)

NOTE: **with the help of etlflag we have know which flag is updated or which one are delete from data**

**Pre-condition for delta load validatation:**

1. Acces to STM
2. Access to qa server, database, and tables
3. There should be some update, insert or delete records in source file
4. Delta load is completed(job run must be done)

**Descripation (design steps): same as initial load**

1. Validate the file row count matches with the target table count

2. Validate the target table data matches with source file

3. Validate the duplicate records present in the table

4. Validate the null values are populating in tgt table

5. Validate the control column are populated as per STM

Note: **if table is maintained with type 2 then we are not able to maintain Pk in the table**

**Step2: landing to staging**

-----Start date, end date, current flag added in the target table

**Current flags:**

Y----Current record

N----old records

**How to know current/latest records from target?**

Select \* from tablename where current flag=’Y’ -----------We will get the all current record OR

Select \* from tablename where end date=’NULL’

**If current flag and end date not available in target table then how we will identifythe current records?**

* **Sorting**: Sort the records in descending order based on a relevant field such as creation date or update date. The record at the top of the sorted list will likely be the current record.
* **Timestamps**: Look for timestamp columns in the database table that indicate when a record was last updated. By comparing the timestamps, you can identify the most recently modified record as the current one.
* **Effective Date Range**: If the records have an effective date range associated with them (e.g., start date and end date), you can identify the current record by checking if the current date falls within the range.
* **Sequence or ID**: If the database table has a sequence or an auto-incrementing ID column, you can use the highest or most recent ID value as an indicator of the current record.

To transfer employee data from a source file to Microsoft SQL Server (MSSQL), you can follow these general steps:

1. **Understand the source file format**: Determine the structure and format of the source file containing employee data. It could be a CSV file, Excel file, or any other supported format.
2. **Create a table in MSSQL**: Design a table in MSSQL that matches the structure of the employee data. Ensure that the table columns align with the data fields in the source file.
3. **Prepare the source file**: Cleanse and format the source file if necessary. Make sure the data is accurate, properly formatted, and consistent with the target table's column definitions.
4. **Connect to the MSSQL database**: Establish a connection to the MSSQL database using a database management tool like SQL Server Management Studio or a programming language like Python with appropriate libraries.
5. **Create an import script**: Write a script or query to import the data from the source file into the MSSQL table. The specific syntax or method may vary depending on the tools or programming language you are using.

-- Assuming you have a table named 'Employees' in MSSQL with columns: EmployeeID, FirstName, LastName, and Department

-- **Step 1**: Create a temporary table to hold the imported data

CREATE TABLE #TempEmployees (

EmployeeID INT,

FirstName VARCHAR(50),

LastName VARCHAR(50),

Department VARCHAR(50)

)

-- **Step 2:** Bulk insert the data from the CSV file into the temporary table

BULK INSERT #TempEmployees

FROM 'C:\Path\To\Source\File.csv' -- Replace with the actual path to your source file

WITH (

FIRSTROW = 2, -- Skip the header row if present

FIELDTERMINATOR = ',', -- Specify the field delimiter in your CSV file

ROWTERMINATOR = '\n' -- Specify the row delimiter in your CSV file

)

-- **Step 3:** Insert the data from the temporary table into the target table

INSERT INTO Employees (EmployeeID, FirstName, LastName, Department)

SELECT EmployeeID, FirstName, LastName, Department

FROM #TempEmployees

-- **Step 4:** Clean up the temporary table

DROP TABLE #TempEmployees

1. **Map the source file fields to database columns**: Specify how each field in the source file corresponds to the columns in the MSSQL table. Ensure the mapping is accurate to transfer the data correctly.
2. **Handle data transformation:** If there are any data transformations required, such as converting data types or applying business rules, perform them during the data transfer process. This step may involve using SQL functions, programming logic, or data manipulation tools.
3. **Execute the import script:** Run the import script to initiate the transfer process. This will read the source file, transform the data (if necessary), and insert it into the MSSQL table.
4. **Verify the data transfer**: After the transfer is complete, verify that the employee data has been successfully transferred to the MSSQL table. Run SQL queries to check the data's integrity and ensure it matches the source file.
5. **Handle any errors or exceptions**: If any errors or exceptions occur during the data transfer process, handle them appropriately. Log the errors, investigate the causes, and rectify the issues before reattempting the transfer.
6. **Schedule or automate the data transfer**: If the employee data needs to be regularly updated or transferred from the source file to the MSSQL table, consider setting up a schedule or automation process to ensure data consistency and accuracy.

Top of Form

When performing initial load validation in ETL testing, you can use SQL queries to compare and validate the data between the source and target tables. Here are some example SQL queries you can use:

1. **Validate Record Counts:**
   * **To check the record counts in the source and target tables:**

-- Record count in the source table

SELECT COUNT(\*) FROM source\_table;

-- Record count in the target table

SELECT COUNT(\*) FROM target\_table;

Compare the record counts to ensure data completeness during the initial load process.

1. **Verify Data Integrity**:

* **To check primary key and foreign key integrity:**

-- Check primary key violations in the target table

SELECT PK\_Column, COUNT(\*) FROM target\_table

GROUP BY PK\_Column

HAVING COUNT(\*) > 1;

-- Check foreign key violations in the target table

SELECT FK\_Column, COUNT(\*) FROM target\_table

GROUP BY FK\_Column

HAVING COUNT(\*) = 0;

**Perform Data Quality Checks:**

* **To identify data quality issues like null values or duplicates:**

-- Check for null values in specific columns

SELECT \*FROM target\_table WHERE column\_name IS NULL;

-- Check for duplicate records in the target table

SELECT column1, column2, COUNT(\*) FROM target\_table

GROUP BY column1, column2

HAVING COUNT(\*) > 1;

**What is cloud:**

Cloud refers to the delivery of computing resources, including servers, storage, databases, networking, software, and analytics, over the internet ("the cloud") on a pay-as-you-go basis. Instead of hosting these resources on local infrastructure or physical servers, cloud computing relies on a network of remote servers that are housed in data centers managed by cloud service providers.

**Why cloud used:**

1. **Scalability and Flexibility**: Cloud computing enables businesses and individuals to scale their computing resources up or down according to their needs. It allows for flexibility in terms of storage, processing power, and software applications. Learning cloud computing helps you understand how to leverage these capabilities to meet specific requirements efficiently.
2. **Cost Efficiency**: Cloud computing offers a cost-effective solution compared to traditional on-premises infrastructure. Instead of investing in expensive hardware and maintenance, cloud services allow you to pay for what you use on a subscription or pay-as-you-go basis. Learning how to optimize costs and utilize cloud resources effectively can save significant expenses for businesses.
3. **Reliability and Availability**: Cloud providers offer robust infrastructure with high availability and reliability. They have redundant systems, data backups, and disaster recovery mechanisms in place to ensure continuous operations. Learning cloud computing helps you understand how to build resilient and fault-tolerant applications and services that can withstand outages and provide uninterrupted services.
4. **Global Accessibility**: With cloud computing, you can access your data, applications, and services from anywhere with an internet connection. This global accessibility facilitates remote work, collaboration, and data sharing across different teams and locations. Learning cloud computing enables you to leverage these capabilities and build cloud-native applications that support modern work environments.
5. **Innovation and Agility**: Cloud computing platforms provide a wide range of tools, services, and APIs that empower developers and businesses to innovate rapidly. They offer pre-built services for artificial intelligence, machine learning, big data processing, internet of things (IoT), and more. By learning cloud computing, you can harness these tools and services to develop cutting-edge solutions and stay competitive in the digital landscape.
6. **Security and Compliance**: Cloud providers invest heavily in security measures and compliance certifications to protect customer data. By learning cloud computing, you can understand the security best practices, identity management, encryption techniques, and compliance requirements necessary to secure your applications and data effectively.
7. **Career Opportunities**: Cloud computing skills are in high demand across various industries. Learning cloud computing can open up job opportunities as cloud architects, cloud engineers, DevOps professionals, data scientists, and more. Cloud skills are transferrable and applicable to different sectors, making you a valuable asset in the job market.

In summary, learning cloud computing equips you with the knowledge and skills to leverage the advantages of cloud technology, enabling you to build scalable, cost-effective, reliable, and innovative solutions in the modern digital landscape.

Top of Form

Bottom of Form

**Why learn hadoop:**

Learning Hadoop, which is an open-source framework for distributed storage and processing of large datasets, can be beneficial for several reasons:

1. **Big Data Processing**: Hadoop is specifically designed to handle big data. It provides a scalable and fault-tolerant platform for storing and processing massive volumes of data across a cluster of computers. By learning Hadoop, you can effectively work with and analyze large datasets that are beyond the capacity of traditional systems.
2. **Distributed Computing**: Hadoop utilizes a distributed computing model, which enables parallel processing of data across multiple machines. This distributed architecture allows for faster data processing and the ability to handle complex tasks efficiently. By learning Hadoop, you gain insights into distributed computing principles and techniques, which are valuable in today's data-intensive world.
3. **Cost-Effective Storage**: Hadoop's Hadoop Distributed File System (HDFS) provides a cost-effective solution for storing large volumes of data. It distributes the data across multiple nodes in a cluster, allowing for high storage capacity without relying on expensive hardware. Learning Hadoop helps you understand how to manage and utilize HDFS effectively, reducing storage costs for your organization.
4. **Data Processing and Analysis**: Hadoop's core component, MapReduce, enables parallel processing of data by breaking it down into smaller tasks that can be executed in parallel. This makes it suitable for performing complex data processing and analysis tasks, such as data cleansing, aggregation, transformation, and statistical analysis. By learning Hadoop, you can leverage MapReduce and related technologies to derive insights and make data-driven decisions.
5. **Ecosystem of Tools:** Hadoop has a rich ecosystem of tools and technologies built around it, such as Hive, Pig, Spark, and HBase. These tools provide higher-level abstractions and query languages, making it easier to work with Hadoop and perform tasks like data querying, data warehousing, real-time processing, and machine learning. By learning Hadoop, you can tap into this ecosystem and gain proficiency in using these complementary technologies.
6. **Scalability and Fault Tolerance**: Hadoop's distributed architecture allows for easy scalability by adding more nodes to the cluster as the data and processing requirements grow. Additionally, Hadoop automatically replicates data across multiple nodes, providing fault tolerance and ensuring data availability even in the event of node failures. By learning Hadoop, you can design and manage scalable and fault-tolerant data processing systems.
7. **Career Opportunities**: Big data and Hadoop skills are highly sought after in the job market. Many organizations across various industries are adopting Hadoop and related technologies to handle their growing data needs. By acquiring Hadoop skills, you can position yourself for career opportunities as a data engineer, data analyst, big data architect, or a Hadoop administrator.

In summary, learning Hadoop equips you with the knowledge and skills to effectively handle big data, perform distributed processing, and derive insights from large datasets. It enables you to work with Hadoop's ecosystem of tools and technologies, opening up career opportunities in the field of big data and data engineering.

Top of Form

Canada client for telecommunicatation:

1. Telus
2. Rogers
3. Aussie broadband

Real time project implementation on cloud:

Project name: employee data migration project to cloud (AZURE):

Client wants to move data from oracle to on cloud(azure)

**Step1**: discussion with business team, development and testing team, data engineering, higher mgt and finalized the requirements

**Step2:** dev and testing team got document as

-HLD (HIGH LEVEL DESIGN)

-LLD(LOW LEVEL DESIGN)

-DDD()

-STM(SOURCE TO TARGET MAPPING SHEET)/ god document

**STEP3:** Dev team started developing code

**Step4**: testing team started writing test plan and test cases

Step5: findings after review of required documents(HLD,DDD,STM) by testing team

Source—oracles tables(employee, departments, region,country,locations)

Target---Azure cloud (employee, departments, region,country,locations)

Step6: mapping sheet documents overview:

-----Due to environments changes data type in the azure is string only, due to this don’t have any data loss

----4 contro control column auto generated in target i.e az.insert date, az-update date, az-run id, etl flag

Step 7: test case excecuation

1. Intro to azure data bricks
2. Src-oracle

Step 8:

-- we will get link to access databricks from any browsers

---databricks is used to access data from cloud (azure)

--table and cluster are already created

--notebook we have to create:

1. name give

2. select the languages: sql, python, scala, R

3. query write in the notebook and run

-- data having tables which are added by developer

--workspace content the notebook which are created by us under our email id in users folder

Notebook querys:

1. Table structure—desc table name
2. All data fetch----select \* from tablename
3. Selected column fetch---select required column name from tablename
4. Cluster use: it is mainly used for run the query, without cluster query will not run

--how to get data from notebook to excel sheet?

1. download option available at below and open with show in folder

Step 9: test case validatation

1. Structure validatation:

Same as above project

1.STM (target)VS target (desc tablename)

**In cloud we just match column name and data tyape. In azure portal at backend we will see other things like pk,fk, data length, constaints etc but we don’t have access to that as tester.**

2.source vs target(desc tablename):

Database—table---tablename (click)---we get table structure

1. Ask developer to run the job with the help of ADF (azures data factory) etl tool used
2. ADF:

-different jobs called as pipelines

-pipelines run as per developer instruction

-pipeline run process: 1.double click on pipeline which we want to run

2. debug click

3. if job fail then anyalysis done by ourself and report to developer

**4. pipeline fail general reasons:**

1. **Data Quality Issues**: Pipeline failures can occur when the incoming data doesn't meet the expected format, schema, or quality standards. Missing or incorrect data, inconsistencies, or data incompatibilities can lead to failures during data ingestion, transformation, or processing stages.
2. **Connectivity and Integration Problems**: Failures can occur due to connectivity issues or problems with integrating different systems or data sources. Issues with network connectivity, API endpoints, credentials, or permissions can prevent data from being successfully transferred or accessed, causing pipeline failures.
3. **Resource Constraints**: Insufficient computing resources, such as CPU, memory, or storage, can lead to pipeline failures. When the pipeline's resource requirements exceed the available capacity, it may result in resource contention, timeouts, or out-of-memory errors.
4. **Software or Infrastructure Errors**: Failures can be caused by bugs, errors, or compatibility issues in the software components or infrastructure used in the pipeline. Problems with libraries, frameworks, dependencies, or system configurations can lead to crashes, exceptions, or unexpected behavior.
5. **Code Issues**: Errors in the pipeline's code, scripts, or configuration files can cause failures. Syntax errors, logic errors, incorrect data transformations, or misconfigured parameters can lead to unexpected results or crashes during the execution of the pipeline.
6. **Environmental Changes**: Changes in the underlying environment, such as updates to operating systems, libraries, or dependencies, can introduce compatibility issues and cause pipeline failures. It is important to keep the pipeline components and dependencies up to date to mitigate such risks.
7. **External Dependencies**: Failures can occur if the pipeline relies on external services, APIs, or third-party systems that experience issues or downtime. Problems with these dependencies, such as service outages, changes in API specifications, or rate limiting, can lead to failures in the pipeline.
8. **Data Volume or Velocity**: In cases where the pipeline is designed to handle high volumes or high-velocity data streams, failures can occur if the pipeline infrastructure or processing components are not properly scaled or optimized to handle the data load.
9. **Security and Access Control**: Failures can happen due to security-related issues, such as incorrect or insufficient access permissions, authentication failures, or authorization errors. These issues can prevent the pipeline from accessing or processing the required data.
10. **Monitoring and Error Handling**: Lack of proper monitoring and error handling mechanisms can make it difficult to identify and handle failures effectively. Inadequate logging, alerting, or error recovery mechanisms can lead to delays in detecting and resolving pipeline failures.

Top of Form

**How to handle pipeline failure:**

Handling pipeline failures requires a systematic approach to identify and resolve the issues. Here are some steps to handle pipeline failures:

1. **Monitor and Detect Failures**: Implement monitoring and alerting mechanisms to detect pipeline failures promptly. Monitor pipeline metrics, logs, and error reports to identify any anomalies or issues in the pipeline's execution.
2. **Investigate the Failure**: When a failure is detected, investigate the logs and error messages to understand the root cause. Analyze the failure point, error messages, and any relevant data to identify the specific issue that caused the failure.
3. **Retry and Error Recovery**: Depending on the nature of the failure, you can implement retry mechanisms to automatically retry the failed operation or stage in the pipeline. For transient failures, retrying the operation after a short delay can help resolve the issue.
4. **Data Validation and Cleansing**: If the failure is due to data quality issues, implement data validation and cleansing techniques. Validate incoming data against expected formats, schemas, or quality rules. Cleanse or transform the data to ensure it meets the required standards before further processing.
5. **Debug and Fix Code Issues**: If the failure is caused by code errors or logic issues, review and debug the code to identify and fix the problem. Analyze the error messages, stack traces, and relevant code sections to pinpoint the issue and make necessary code corrections.
6. **Review Dependencies and Integration Points**: Check if the failure is related to external dependencies or integration points in the pipeline. Verify the connectivity, credentials, API endpoints, or system configurations to ensure proper integration and smooth data flow.
7. **Scale Resources**: If the failure is due to resource constraints, consider scaling up the computing resources such as CPU, memory, or storage. Adjust the resource allocation based on the workload requirements to avoid bottlenecks and improve pipeline performance.
8. **Handle Security and Access Issues**: If the failure is related to security or access control, review the permissions, authentication mechanisms, or authorization rules. Ensure that the pipeline has the necessary access rights to the required data sources or services.
9. **Implement Error Handling and Logging**: Enhance the error handling mechanisms in the pipeline. Capture and log detailed error messages, stack traces, and contextual information to facilitate troubleshooting and debugging. Use proper logging practices to track the execution flow and capture relevant information during pipeline failures.
10. **Test and Deploy Fixes**: Once the issue is identified and resolved, thoroughly test the fix to ensure it addresses the problem. Test the updated pipeline or code in a controlled environment before deploying it to the production pipeline. Deploy the fix carefully to minimize the impact on ongoing pipeline operations.
11. **Continuous Improvement:** Learn from the failures and take steps to prevent similar issues in the future. Conduct post-mortem analysis to identify the root causes and implement preventive measures, such as automated testing, code reviews, or additional monitoring, to reduce the likelihood of similar failures.

Handling pipeline failures requires a combination of proactive measures, effective monitoring, quick detection, and timely resolution. By following these steps and continuously improving the pipeline's design and operations, you can minimize the impact of failures and ensure a more resilient and reliable pipeline.

Top of Form

**2.COUNT VALIDATATION:**

---Select count(\*) from tablename order by pk column …..query for src and target

**3.data validatation:**

--source from oracle:

1. Select \* from tablename

2. ctl+shift+c and paste in to the excel sheet

--target from databricks:

1. select \* from tablename

2. download and show in folder and paste in to the excel sheet

--data validatation sheet in excel :

Conditional formatting used for highlight the mismatch

1. Null value validatation:

---- Select count(\*)-count(column pk) from tablename

1. Duplicate validatation:

---Select count(\*) from table name group by (all columns in table) having count(\*)>1

---With cte as

(Select count(\*) cnt from table name group by (all columns in table))

Select \* from cte where cnt>1

1. Control column validatation:

---- Select count(\*), control column names (4) from tablename group by control column names

4.delta load validatation:

##Data inseratation techniques:

---**Truncate and load**" refers to a data loading technique where existing data in a target database table is first truncated or deleted, and then replaced with new data.

--- "**Insert only**" is a data loading technique where new data is inserted into a target table without modifying or deleting any existing data.

This approach is typically used when you want to add new records to an existing dataset without affecting the previously loaded data.

---type 1:

---type 2: current flag changes here

|  |  |  |
| --- | --- | --- |
| **parameter** | **cloud** | **Traditional storage** |
| **Infrastructure** | cloud storage relies on remote servers and infrastructure provided by a cloud service provider. | Traditional storage involves physical infrastructure, such as on-premises servers, storage arrays, and network equipment, which are owned and managed by the organization |
| **Accessibility** | Cloud storage, on the other hand, provides ubiquitous access to data over the internet from any location and device with an internet connection. | Traditional storage is typically limited to the physical location where the storage infrastructure is deployed.  Accessing data stored in traditional storage may require being on-site or using a dedicated network connection |
| **Scalability and Elasticity** | Cloud storage offers the advantage of scalability and elasticity, allowing users to easily scale up or down their storage resources based on demand.  Cloud providers offer flexible storage plans, allowing organizations to pay for the storage they actually use | traditional storage often requires upfront capacity planning and investment in hardware, which may result in either overprovisioning or resource constraints. |
| **Cost** | Cloud storage follows a pay-as-you-go model, where users pay for the storage they consume on a subscription or usage basis, resulting in potential cost savings and cost predictability. | Traditional storage typically involves upfront capital expenditures for purchasing and maintaining storage infrastructure. Additionally, organizations bear the ongoing costs of power, cooling, maintenance, and upgrades |
| **Maintenance and Management** | Cloud storage offloads these responsibilities to the cloud service provider, allowing organizations to focus on their core business operations. | organizations are responsible for the maintenance, management, and monitoring of the storage infrastructure. This includes tasks such as hardware upgrades, software updates, backup management, and security configurations. |
| **Reliability and Redundancy** | Cloud storage providers typically offer high levels of redundancy and data durability. They replicate data across multiple data centers, ensuring data availability and protecting against hardware failures or disasters. | Traditional storage may require organizations to implement their own redundancy mechanisms and disaster recovery strategies. |
| **Security** | Cloud storage providers, however, invest heavily in security technologies and adhere to industry-standard security practices, which can offer strong data protection | Traditional storage allows organizations to have direct control over their security configurations and implement their chosen security measures |

**what is Paas, Iaas, Saas and example of it?**

PaaS, IaaS, and SaaS are different models of cloud computing that provide varying levels of infrastructure and service to users. Here's an overview of each model and examples of how they are commonly used:

1. **Infrastructure as a Service (IaaS)**: IaaS provides users with virtualized computing resources over the internet. It allows users to manage and control the underlying infrastructure, including virtual machines, storage, and networking, while the cloud provider is responsible for maintaining the physical hardware.

**Examples of IaaS:**

* Amazon Web Services (AWS) Elastic Compute Cloud (EC2): AWS EC2 provides virtual machines on-demand, allowing users to have complete control over the operating system, software, and configurations.
* Google Cloud Platform (GCP) Compute Engine: GCP Compute Engine offers virtual machine instances with customizable resources and scalability options.

1. **Platform as a Service (PaaS):** PaaS provides a complete development and deployment environment in the cloud, allowing developers to focus on building applications without worrying about the underlying infrastructure. PaaS typically includes the operating system, runtime environment, and middleware, while the cloud provider manages the hardware and networking.

**Examples of PaaS:**

* Microsoft Azure App Service: Azure App Service offers a fully managed platform for building, deploying, and scaling web and mobile applications. It supports various programming languages and frameworks.
* Heroku: Heroku is a PaaS platform that allows developers to deploy, manage, and scale applications in multiple languages, providing a streamlined development experience.

1. **Software as a Service (SaaS)**: SaaS provides ready-to-use software applications that are accessible over the internet on a subscription basis. Users access the software through a web browser or client application, without needing to manage or control the underlying infrastructure, platform, or software stack.

**Examples of SaaS:**

* Salesforce: Salesforce is a cloud-based CRM (Customer Relationship Management) platform that provides a suite of applications for managing sales, customer service, and marketing.
* Google Workspace: Google Workspace (formerly G Suite) offers a range of productivity and collaboration tools, including Gmail, Google Docs, Google Drive, and Google Calendar, accessible through a web browser.

In summary, IaaS provides virtualized infrastructure resources, PaaS offers a platform for application development and deployment, and SaaS delivers ready-to-use software applications. Each model provides different levels of control and responsibility, catering to the diverse needs of users in the cloud computing ecosystem.

Top of Form

**Cloud service provider:**

* 1. AWS( Amazon web services)
  2. **AZURE(microsoft)**
  3. Google cloud flatform (GCP)
  4. IBM Cloud
  5. Oracle cloud infrastructure (OCI)
  6. Alibaba cloud

**HADOOP:**

Hadoop is an open-source framework that provides a distributed processing and storage system for handling large volumes of data. It was created to address the challenges of processing and analyzing massive datasets known as Big Data. The core components of Hadoop include:

1. **Hadoop Distributed File System (HDFS**): HDFS is a distributed file system designed to store and manage large amounts of data across multiple commodity servers. It breaks down files into blocks and distributes them across the cluster, providing fault tolerance and high availability.
2. **MapReduce**: MapReduce is a programming model for processing and analyzing large datasets in parallel across a distributed cluster. It divides a computation into two stages: the map stage, where data is processed and transformed, and the reduce stage, where the results are aggregated.
3. **YARN (Yet Another Resource Negotiator)**: YARN is a resource management and job scheduling framework in Hadoop. It manages and allocates resources to different applications running on the Hadoop cluster, enabling efficient resource utilization.

Hadoop is known for its ability to process data in a distributed and fault-tolerant manner. It allows organizations to store and process vast amounts of data across clusters of commodity hardware, making it cost-effective and scalable. Hadoop is highly suitable for batch processing, data warehousing, log processing, and large-scale data analytics.

Additionally, the Hadoop ecosystem comprises various complementary tools and frameworks that extend the functionality of the core components. Some popular tools in the Hadoop ecosystem include:

* **Apache Spark**: A fast and general-purpose data processing engine that supports real-time streaming, machine learning, and graph processing.
* **Apache Hive**: A data warehousing and SQL-like query language that enables data summarization, ad-hoc querying, and analysis on Hadoop.
* **Apache HBase**: A distributed NoSQL database that provides random read and write access to large-scale datasets.
* **Apache Pig**: A high-level scripting language for expressing data analysis tasks that are executed on Hadoop.
* **Apache Kafka**: A distributed streaming platform that enables the real-time processing of high-volume, continuous data streams.

Hadoop has become a fundamental technology in the Big Data landscape, providing a scalable and reliable platform for processing and analyzing large datasets. It has been widely adopted by organizations across industries to gain insights, make data-driven decisions, and derive value from their data.

Top of Form

**Word count** is a classic example of how the MapReduce programming model can be used to process and analyze large datasets. The goal of the word count MapReduce algorithm is to count the occurrences of each word in a given input dataset.

Here's an overview of how the word count algorithm works using the MapReduce approach:

1. **Map Phase:**
   * Input: Each input record is a line of text from the dataset.
   * Map Function: The map function takes the input record and emits key-value pairs. In this case, the map function splits the input line into words and emits each word as the key and a count of 1 as the value.

Example: Input: "Hello world, hello OpenAI" Output Key-Value Pairs:

* + ("Hello", 1)
  + ("world", 1)
  + ("hello", 1)
  + ("OpenAI", 1)

1. **Shuffle and Sort:**
   * The output of the map phase is passed through a shuffle and sort process. The shuffle phase collects the output key-value pairs from all the map tasks and groups them based on the keys.
   * The key-value pairs are sorted by the keys to ensure that all occurrences of the same word are grouped together.
2. **Reduce Phase:**
   * Input: Each input record in the reduce phase is a unique key along with its associated list of values.
   * Reduce Function: The reduce function takes the input key-value pairs and performs the aggregation. In this case, the reduce function sums up the counts for each word.

Example: Input Key-Value Pairs:

* + ("Hello", [1, 1])
  + ("world", [1])
  + ("hello", [1])
  + ("OpenAI", [1])

Output Key-Value Pairs:

* + ("Hello", 2)
  + ("world", 1)
  + ("hello", 1)
  + ("OpenAI", 1)

1. **Final Output:** The final output of the word count MapReduce algorithm is the list of unique words along with their respective counts.

Example:

* + "Hello" -> 2
  + "world" -> 1
  + "hello" -> 1
  + "OpenAI" -> 1

The MapReduce framework handles the distribution and parallel execution of the map and reduce tasks across a cluster of machines, enabling efficient processing of large datasets. This approach allows for scalability, fault tolerance, and efficient utilization of resources.

The word count MapReduce algorithm serves as a simple illustration of how MapReduce can be used for data processing tasks. However, the MapReduce model can be applied to more complex data analysis scenarios, such as analyzing web logs, processing sensor data, or running machine learning algorithms on large datasets.

Top of Form

Bottom of Form

Snowflake data warehouse: (target)

----Look similar to mssql

---source---landing—staging---snowflake data warehouse (integratation with AWS)