**ETL TESTING TOPICS**

Sql

Dwh basics, data modeling

Etl testing

Bi testing

Unix

**SCD 0 :** No changes

**SCD 1** : History not maintained. Only current data

SCD 1 is followed in business tables ( only current data )

**SCD 2** : Rows added for each change.

Extra columns (start\_date,end\_date,current\_flag,surrogate\_key) required for maintenance.

SCD 2 is generally followed in staging tables to maintain history

**SCD 3** : Extra columnd added for each change.

**SCD 4** : Dimesion tables maintains updated data.

History maintained in separate table.

**SCD 6** : SCD 1+2+3

Overview of ETL testing : Types of etl test scenarios, etl bugs,

<https://www.guru99.com/utlimate-guide-etl-datawarehouse-testing.html>

**Datawarehouse concepts**

<https://www.tutorialspoint.com/dwh/index.htm>   1.5 hrs

**dbms keys**

<https://www.guru99.com/dbms-keys.html>

**Types of loading**

1.       Initial or full load.

2.       Delta or incremental load

3.       History load

**Identify the grain**

The Grain describes the level of detail for the business problem/solution. It is the process of identifying the lowest level of information for any table in your data warehouse. If a table contains sales data for every day, then it should be daily granularity. If a table contains total sales data for each month, then it has monthly granularity.

**Dimensional modelling**

<https://www.guru99.com/dimensional-model-data-warehouse.html>

**Database vs DWH**

<https://www.guru99.com/database-vs-data-warehouse.html>

**Database ACID properties :**

<https://www.geeksforgeeks.org/acid-properties-in-dbms/>

**DATA WAREHOUSE ARCHITECTURES**

There are mainly three types of Datawarehouse Architectures: -

**Single-tier architecture**

The objective of a single layer is to minimize the amount of data stored. This goal is to remove data redundancy. This architecture is not frequently used in practice.

**Two-tier architecture**

Two-layer architecture separates physically available sources and data warehouse. This architecture is not expandable and also not supporting a large number of end-users. It also has connectivity problems because of network limitations.

**Three-tier architecture**

This is the most widely used architecture.

It consists of the Top, Middle and Bottom Tier.

1. Bottom Tier: The database of the Datawarehouse servers as the bottom tier. It is usually a relational database system. Data is cleansed, transformed, and loaded into this layer using back-end tools.
2. Middle Tier: The middle tier in Data warehouse is an OLAP server which is implemented using either ROLAP or MOLAP model. For a user, this application tier presents an abstracted view of the database. This layer also acts as a mediator between the end-user and the database.
3. Top-Tier: The top tier is a front-end client layer. Top tier is the tools and API that you connect and get data out from the data warehouse. It could be Query tools, reporting tools, managed query tools, Analysis tools and Data mining tools.

**Data quality check** – Number check, date format check, precision check, null check, data check.

**Data completeness** – count check, data profile check (aggregation check), column data profile check ( count of distinct values in column), duplicate data check.

**Testing readiness :**

Scope of testing − Describe testing techniques and types to be used.

Setting up the test environment.

Test data availability − It is recommended to have production like data covering all/critical business requirement.

Data quality and performance acceptance criteria.

**Test Strategy vs Test Plan**

<https://www.guru99.com/test-plan-v-s-test-strategy.html>

**Defect management process :**

1. Defect discovery
2. Defect categorization
3. Defect resolution
4. Defect verification
5. Defect closure
6. Defect Reporting

Defect reject ratio = No. of defects rejected/Total No. of defects raised \* 100

Defect leakage ratio = No. of defect missed/ Total defects in software \* 100

**Database vs Schema**

Database : A database is the main container, it contains the data and log files, and all the schemas within it. You always back up a database, it is a discrete unit on its own.

Schema : Schemas contains table ( views, procedures,..) definitions, their columns and datatype, relations and stuff. You cannot drop a schema when it is in use. You have to first remove all objects from the schema

Schemabinding : Used when creating views to restrict the changes in tables which will impact this view.

**Temporary tables in SQL Server**

There are two types of temporary tables: local and global. Local temporary tables are visible only to their creators during the same connection to an instance of SQL Server as when the tables were first created or referenced. Local temporary tables are deleted after the user disconnects from the instance of SQL Server. Global temporary tables are visible to any user and any connection after they are created, and are deleted when all users that are referencing the table disconnect from the instance of SQL Server

**MINUS vs EXCEPT**

No difference

**CHANGE DATA CAPTURE – CDC**

**Concept : Instead of loading full data ( full data load) for every refresh apply incremental load( update /insert/delete )**

Started from MS-SQL SERVER 2008

Change Data Capture records INSERTs, UPDATEs, and DELETEs applied to SQL Server tables, and makes a record available of what changed, where, and when, in simple relational 'change tables' rather than in a XML.

CDC first has to be enabled for the database. Because CDC is a table-level feature, it then has to be enabled for each table to be tracked.

Run following query and check whether it is enabled for any database :

*USE master*

*GO*

*SELECT [name], database\_id, is\_cdc\_enabled*

*FROM sys.databases*

*GO*

To enable CDC at database level run below query :

*USE XYZ\_DB*

*GO*

*EXEC sys.sp\_cdc\_enable\_db*

*GO*

Additionally, in the database XYZ\_DB, you will see that a schema with the name ‘cdc’ has now been  created.

Some System Tables will have been created within the  XYZ\_DB  database as part of the cdc schema.

Q1. Etl tools used ?

Q2. Database used ( source/target) ?

**Etl flow**

Oltp --> stg-->ods-->(edw)/dm --> (dm)

Where …

Ods = operational data store (It is a repository between staging area and data warehouse. This is the place where most of the data used in current operation is housed before it's transferred to the data warehouse for longer term storage or archiving. An ODS is similar to your short term memory in that it stores only very recent information; in comparison, the data warehouse is more like long term memory in that it stores relatively permanent information )

Edw = enterprise data warehouse

Dm = data mart ( A data mart is focused on a single functional area of an organization and contains a ***subset of data stored in a Data Warehouse.***

It is designed for use by a specific department, unit or set of users in an organization. E.g., Marketing, Sales, HR or finance. It is often controlled by a single department in an organization.)

validation – done at development phase to check software requirements. Code execution NOT required

verification – done at the end of development phase to check customer requirements. Code execution required.

SDLC – applicable for whole software ( includes STLC). Phases are :

1. Requirements Gathering

2. Design the software

3. Build the Software

4. Test

5. Deployment

6. Maintenance.

STLC – for testing only. Phases are :

1. Requirement Analysis

2. Test Planning

3. Test Development

4. Test Environment Setup

5. Test Execution & Closure

DELETE

1. DML

2. Delete using where condition ( if no where condition is used all records will be deleted )

3. Rollback possible as logs are mentioned.

4. Table structure is not deleted

TRUNCATE

1. DDL

2. Delete all records. Applying filter not possible.

3. Rollback not possible

4. Table structure is not deleted. It remains in the memory

5. Faster than delete

DROP

1. DDL

2. Delete all records and table

3. Rollback not possible

4. Table structure gets deleted

CROSS APPLY – almost similar to inner join with an implicit join condition of 1=1

OUTER APPLY – almost similar to left outer join.

But this two can be used with table valued expressions.

DATABASE – mainly used for OLTP, normalized ( to reduce redundant data and storage space), optimized for write operations, designed using ER model

DWH – mainly used for OLAP, de-normalized ( to reduce response time for analytical queries), optimized for read operations, designed using data modelling.

Dimensional data model

Fact table – contains measures/facts and foreign key to dimension table.

Dimension table – contains dimensions of a fact, join to fact table via foreign key, de-normalized tables.

Data profile validation - It involves checking the aggregate functions such as count, sum, and max in the source and target tables.

DATA INTEGRITY CHECKS

1. Entity integrity – check for data duplication

2. Domain integrity – set of data values/columns falls within a specific permissible defined range. Examples of domain integrity are correct data type, format, and data length; values must fall within the range defined for the system

3. Referential integrity – Primary Key (PK) and Foreign Key (FK) relationships. It may be tested, for example, by deleting parent rows or the child rows in tables.

INFORMATICA

Power center client tools : repository manager, Power center designer, workflow manager, workflow monitor

The following components needs to be set up before you can start working in Informatica PowerCenter.

1. Informatica domain - adding a node, creating repository service, creating integration service, managing & creating users, monitoring services, etc.

2. creating repository service - It is the first service that needs to be created in the domain console. The pre-requisite is to have a database user created in the database along with basic privileges (connect, resource), as this service maintains repository data in the database

3. creating integration service - The integration service is responsible for the execution of workflows. Whenever, any workflow is executed from workflow manager, scheduler or command prompt, the integration service receives a request. This service locks the respective workflow, fetches the details about the workflow, mapping & session from the Informatica repository and executes the workflow. This service is also responsible for creating workflow logs, reading parameter files, creating process threads to execute workflows, execute any other task created in workflow manager, etc.

4. Creating users –

Power center designer have 5 different workspaces :

Source analyzer, target designer, mapping designer, mapplet designer, transformation developer

Components of workflow manager ( workflow execution is sequence or event based)

1. Task developer – Create reusable task to be used in multiple workflows.

a. Command task – to execute windows/unix commands

b. session task – to run a mapping.

c. email task - send email to defined recipients when the Integration Service runs a workflow.

2. Worklet Designer

3. Workflow designer

**Update Strategy Transformation in Informatica**

The Update Strategy Transformation in Informatica is an Active, and Connected transformation. This is very useful to update, insert, or delete the records in a target based on source table data.

Update Strategy Expression :

DD\_INSERT : Numeric Value is 0. This is used to insert records.

DD\_UPDATE : This is used to update rows, and the Numeric Value is 1.

DD\_DELETE : Numeric Value is 2. This is used to delete rows.

DD\_REJECT : Numeric Value is 3. This is used to reject rows.

**UNIX**

Process status ps

List all the process running in current shell : ps ( it displays pid, tty, stat, time taken, command which started the process)

List all the ACTIVE process running in current shell : ps –A

List all the process runned by you : ps –x

List all the process runned by a user : ps –u username

List the process by pid : ps –p PID

Kill process forcefully : kill -9 PID1 PID2

Foreground process – it’s normal commands in unix

Start a background process : ls –l &

**SQL**

Inline view – select count(\*) from ( select distinct col1,col2 from table)as inline\_vw

Show first date of the month

( sample : sqlServer 🡪 DATAADD(m,123,0) = Add 123 months to the date 0 )

Sql server --> SELECT DATEADD(m,datediff(m,0,'2019/02/25'),0) AS DateAdd;

Hive --> SELECT trunc(date\_col,’month’) FROM table

Hive --> SELECT trunc(date\_col,’year’) FROM table

Show last date of the month

Sql server --> SELECT DATEADD(m,1+datediff(m,0,'2019/02/25'),-1) AS DateAdd;

Months between in hive

Select (month(date\_col1)-month(date\_col2)) from table

Convert Timestamp to YYYYMMDD format using HiveQL

hive> select from\_unixtime(unix\_timestamp(date\_col), 'yyyyMMdd');

***DESCIBE IN SQL SERVER***

exec sp\_columns table\_name;

**DUPLICATE RECORDS – FIND AND DELETE**

-- display duplicate records only once and don’t display unique records

select empid,deptid,salary,count(\*) as count\_ from test.dbo.employee1

group by empid,deptid,salary

having count(\*) > 1

-- display duplicate records only once PLUS display unique records

select empid,deptid,salary,count(\*) as count\_ from test.dbo.employee1

group by empid,deptid,salary

having count(\*) = 1

UNION

select empid,deptid,salary,count(\*) as count\_ from test.dbo.employee1

group by empid,deptid,salary

having count(\*) > 1

-- display duplicate records (repeated N times) N-1 times

select \* from (

select empid,deptid,salary,row\_number() over (partition by empid,deptid,salary order by empid,deptid,salary ) as row\_num

from test.dbo.employee1

)vw

where row\_num > 1

-- delete duplicate records using CTE

with cte as (

select empid,deptid,salary,row\_number() over (partition by empid,deptid,salary order by empid,deptid,salary ) as row\_num

from test.dbo.employee1

)

delete from cte where row\_num > 1;

select \* from test.dbo.employee1;

-- delete duplicate records without using CTE

delete vw from (

select empid,deptid,salary,row\_number() over (partition by empid,deptid,salary order by empid,deptid,salary ) as row\_num

from test.dbo.employee1

)vw

where row\_num > 1;

select \* from test.dbo.employee1;

**Find greatest value across columns in rows – sql server**

rollno student phy che bio

1 Ram 45 89 56

Result = 89

Using max

select rollno, student,(select max(col\_transposed) from (values (phy),(che),(bio)) as listOfValues(col\_transposed)) from table\_name

Using unpivot

SELECT id,productname,max(pivot\_val) FROM (

select id,productname,supplierid,cast(unitprice as int) as unitprice\_,cast(IsDiscontinued as int) as IsDiscontinued\_ from product

)inner\_table

UNPIVOT

(pivot\_val FOR cols in (supplierid,unitprice\_,IsDiscontinued\_))as pivot\_x

GROUP BY id,productname

Example - listOfVales

select 'abc',col\_1 from (values (1),(2),(3))as listOfValues(col\_1)

RESULT

(No column name) col\_1

abc 1

abc 2

abc 3

example – unpivot (transpose)

SELECT id,productname,cols,pivot\_val FROM (

select id,productname,supplierid,cast(unitprice as int) as unitprice\_,cast(IsDiscontinued as int) as IsDiscontinued\_ from product --unpivot must have same data\_types

)inner\_table

UNPIVOT

(pivot\_val FOR cols in (supplierid,unitprice\_,IsDiscontinued\_))as pivot\_x

RESULT

BEFORE UNPIVOT

Id ProductName SupplierId UnitPrice Package IsDiscontinued

1 Chai 1 18.00 10 boxes x 20 bags 0

2 Chang 1 19.00 24 - 12 oz bottles 0

AFTER UNPIVOT

id productname cols pivot\_val

1 Chai supplierid 1

1 Chai unitprice\_ 18

1 Chai IsDiscontinued\_ 0

2 Chang supplierid 1

2 Chang unitprice\_ 19

2 Chang IsDiscontinued\_ 0

**Show schema details in sql server**

select \* from INFORMATION\_SCHEMA.COLUMNS where table\_schema='dbo' and table\_name='Product'

ROLLUP

select [CountryName],year(filmreleasedate)year,count(\*)count\_films from movies.dbo.tblfilm

inner join movies.[dbo].[tblCountry] on [CountryID]=[FilmCountryID]

where year(filmreleasedate) in (2004,2005,2006)

group by rollup(year(filmreleasedate),[CountryName]) order by count\_films desc;

CountryName year count\_films

NULL NULL 60

NULL 2004 22

NULL 2005 20

United States 2004 19

NULL 2006 18

United States 2006 17

United States 2005 17

United Kingdom 2004 2

China 2004 1

China 2005 1

Germany 2005 1

Russia 2005 1

United Kingdom 2006 1

CUBE

select [CountryName],year(filmreleasedate)year,count(\*)count\_films from movies.dbo.tblfilm

inner join movies.[dbo].[tblCountry] on [CountryID]=[FilmCountryID]

where year(filmreleasedate) in (2004,2005,2006)

group by cube(year(filmreleasedate),[CountryName]) order by count\_films desc;

CountryName year count\_films

NULL NULL 60

United States NULL 53

NULL 2004 22

NULL 2005 20

United States 2004 19

NULL 2006 18

United States 2005 17

United States 2006 17

United Kingdom NULL 3

China NULL 2

United Kingdom 2004 2

United Kingdom 2006 1

China 2004 1

China 2005 1

Germany 2005 1

Germany NULL 1

Russia 2005 1

Russia NULL 1

--**SCALAR VALUED FUNCTION**

--CREATE FUNCTION

create function func\_1(@countryid int)

returns int

as

begin

return (select count(\*) from [Movies].[dbo].[tblFilm] where [FilmCountryID]=@countryid)

end

--INVOKE THE FUNCTION --1

declare @count\_films int

execute @count\_films=func\_1 241

select @count\_films

--INVOKE THE FUNCTION --2

select dbo.func\_1(240)

--**INLINE TABLE VALUED FUNCTION**

--CREATE FUNCTION

create function func\_2(@countryid int)

returns table

as

return (select \* from [Movies].[dbo].[tblFilm] where [FilmCountryID]=@countryid)

--INVOKE THE FUNCTION --1

select \* from dbo.func\_2(241)

--**MULTI-STATEMENT TABLE VALUED FUNCTION**

--DROP IF EXISTS

IF object\_id('dbo.func\_3') is not null DROP FUNCTION dbo.func\_3;

--CREATE FUNCTION

CREATE FUNCTION func\_3(@ACTOR\_ID\_PARAM int)

RETURNS @age\_table TABLE(actorid int,actorname varchar(50),actordob date,actorage float)

AS

BEGIN

DECLARE @actorid2 int,@actorname2 varchar(50),@actordob2 date,@actorage2 float

select @actorid2=@ACTOR\_ID\_PARAM,@actorname2=actorname,@actordob2=actordob,@actorage2=DATEDIFF(year,actordob,GETDATE())

from [Movies].[dbo].[tblActor]

where ActorID=@ACTOR\_ID\_PARAM

INSERT INTO @age\_table VALUES(@ACTOR\_ID\_PARAM,@actorname2,@actordob2,@actorage2)

RETURN

END;

--EXECUTE THE FUNCTION

select \* from dbo.func\_3(1)

**ERROR HANDING – TRY CATCH**

BEGIN TRY

insert into test.dbo.employee1 values('a',22,2222);

END TRY

BEGIN CATCH

print 'ERROR NO : ' + cast(ERROR\_NUMBER() as varchar)

select 'error msg : ' + ERROR\_MESSAGE()

END CATCH

**online compiler**

https://rextester.com/l/sql\_server\_online\_compiler

SELECT employeeid,count(orderid) FROM [Orders] group by employeeid

SELECT employeeid,shipperid,count(orderid) FROM [Orders] group by employeeid,shipperid

SELECT employeeid,shipperid,customerid,count(orderid) FROM [Orders] group by employeeid,shipperid,customerid

select productname from products p1 where price > (select avg(price) from products p2 where p1.products=p2.products group by categoryid )

BEGIN TRANSACTION;

/\* Create a table called NAMES \*/

CREATE TABLE NAMES(Id integer PRIMARY KEY, Name text);

/\* Create few records in this table \*/

INSERT INTO NAMES VALUES(1,'Tom');

INSERT INTO NAMES VALUES(2,'Lucy');

INSERT INTO NAMES VALUES(3,'Frank');

INSERT INTO NAMES VALUES(4,'Jane');

INSERT INTO NAMES VALUES(5,'Robert');

COMMIT;

/\* Display all the records from the table \*/

SELECT \* FROM NAMES;

create

select i.id,i.billingdate,c.name,referredby

from invoices i

inner join customers c ON i.customerid=c.id

10 TH HIGHEST

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

SELECT price,productname FROM

(

select price,productname from products order by price desc limit 10

)

order by price limit 1

SQL-SERVER

SELECT top 1 empname,salary FROM

(

select top 10 empname,salary from emp order by salary desc

)

a order by salary

10 TH HIGHEST

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

SELECT price,productname FROM products p1

where 9 = (

select count(price) FROM products p2 where p1.price<p2.price

)

UPDATE products SET price=999 WHERE price=45.6

SWITCH-CASE

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

update categories

set categoryid =

case categoryid%2

when 0

    then categoryid+200

else

    categoryid+100

end

**show dept\_name having zero or 3 employess**

select d.d\_name,count(e.empid) from dept d

left join test.dbo.employee e

on d.d\_id=e.deptid

group by d.d\_name

having count(e.empid) in (0,3)

SELECT \* FROM customers where a=ProductID union all SELECT \* FROM customers WHERE b=Y AND a!=ProductID

select productid from orderdetails where quantity > 10

UNION ALL

select employeeid from orders

select empid,empname,avg(salary) from emp e1

where empid= (

select manager\_id from emp e2 where e2.manager\_id=e1.empid)

group by e1.empid,e1.empname

SELECT productname,price,truncate(price,0),price-truncate(price,0),substring(price-truncate(price,0),3) FROM Products

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

STORED PROCEDURE

create table emp(

    empid int,

    empname varchar(100),

    salary int,

    deptid int

)

insert into emp values

(10,'Anil',50000,18),

(11,'Vikas',75000,16),

(12,'Nisha',40000,18),

(13,'Nidhi',60000,17),

(14,'Priya',80000,18),

(15,'Mohit',45000,18),

(16,'Rajesh',90000,17),

(17,'Raman',55000,16),

(18,'Santosh',65000,17);

----

CREATE PROCEDURE proc1 (@id INT,@avg\_sal INT OUT)

AS

BEGIN

select @avg\_sal=AVG(salary) from emp where deptid=@id GROUP BY deptid

END

--

DECLARE @out11 INT

EXECUTE proc1 16,@out11 OUT

SELECT @out11

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

RANK

create table emp(

    empid int,

    empname varchar(100),

    salary int,

    deptid int

)

insert into emp values

(10,'Anil',70000,18),

(11,'Vikas',75000,16),

(12,'Nisha',40000,18),

(13,'Nidhi',60000,17),

(14,'Priya',55000,18),

(15,'Mohit',55000,18),

(16,'Rajesh',90000,17),

(17,'Raman',55000,16),

(18,'Santosh',65000,17);

--select \*,row\_number() over(order by salary) as "ROW NUMBER" from emp

empid   empname           salary    deptid   ROW NUMBER

12           Nisha     40000    18           1

14           Priya      55000    18           2

15           Mohit    55000    18           3

17           Raman  55000    16           4

13           Nidhi     60000    17           5

18           Santosh 65000    17           6

10           Anil        70000    18           7

11           Vikas     75000    16           8

16           Rajesh  90000    17           9

--select \*,rank() over(order by salary desc) as "RANK" from emp

empid   empname           salary    deptid   RANK

16           Rajesh  90000    17           1

11           Vikas     75000    16           2

10           Anil        70000    18           3

18           Santosh 65000    17           4

13           Nidhi     60000    17           5

14           Priya      55000    18           6

15           Mohit    55000    18           6

17           Raman  55000    16           6

12           Nisha     40000    18           9

--select \*,dense\_rank() over(order by salary desc) as "DENSE RANK" from emp

empid   empname           salary    deptid   DENSE RANK

16           Rajesh  90000    17           1

11           Vikas     75000    16           2

10           Anil        70000    18           3

18           Santosh 65000    17           4

13           Nidhi     60000    17           5

14           Priya      55000    18           6

15           Mohit    55000    18           6

17           Raman  55000    16           6

12           Nisha     40000    18           7

--select \*,row\_number() over(partition by deptid order by salary desc) as "ROW NUMBER" from emp

mpid      empname           salary    deptid   ROW NUMBER

11           Vikas     75000    16           1

17           Raman  55000    16           2

16           Rajesh  90000    17           1

18           Santosh 65000    17           2

13           Nidhi     60000    17           3

10           Anil        70000    18           1

14           Priya      55000    18           2

15           Mohit    55000    18           3

12           Nisha     40000    18           4

--select \*,dense\_rank() over(partition by deptid order by salary desc) as "DENSE RANK" from emp

empid   empname           salary    deptid   DENSE RANK

11           Vikas     75000    16           1

17           Raman  55000    16           2

16           Rajesh  90000    17           1

18           Santosh 65000    17           2

13           Nidhi     60000    17           3

10           Anil        70000    18           1

14           Priya      55000    18           2

15           Mohit    55000    18           2

12           Nisha     40000    18           3

--select \*,ntile(3) over(order by salary) as "NTILE\_3" from emp

empid   empname           salary    deptid   NTILE\_3

12           Nisha     40000    18           1

14           Priya      55000    18           1

15           Mohit    55000    18           1

17           Raman  55000    16           2

13           Nidhi     60000    17           2

18           Santosh 65000    17           2

10           Anil        70000    18           3

11           Vikas     75000    16           3

16           Rajesh  90000    17           3

select \*,ntile(5) over(order by salary) as "NTILE\_5" from emp

empid   empname           salary    deptid   NTILE\_5

12           Nisha     40000    18           1

14           Priya      55000    18           1

15           Mohit    55000    18           2

17           Raman  55000    16           2

13           Nidhi     60000    17           3

18           Santosh 65000    17           3

10           Anil        70000    18           4

11           Vikas     75000    16           4

16           Rajesh  90000    17           5

--select \*,ntile(2) over(partition by deptid order by salary) as "NTILE\_2" from emp

empid   empname           salary    deptid   NTILE\_2

17           Raman  55000    16           1

11           Vikas     75000    16           2

13           Nidhi     60000    17           1

18           Santosh 65000    17           1

16           Rajesh  90000    17           2

12           Nisha     40000    18           1

14           Priya      55000    18           1

15           Mohit    55000    18           2

10           Anil        70000    18           2

--select \*,ntile(2) over(partition by deptid order by salary,empname) as "NTILE\_2" from emp

empid   empname           salary    deptid   NTILE

17           Raman  55000    16           1

11           Vikas     75000    16           2

13           Nidhi     60000    17           1

18           Santosh 65000    17           1

16           Rajesh  90000    17           2

12           Nisha     40000    18           1

15           Mohit    55000    18           1

14           Priya      55000    18           2

10           Anil        70000    18           2

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

**MERGING**

create table employee(

    empid int,

    deptid int,

    salary int,

)

insert into employee values

(1,10,100000)

(2,20,100000)

(3,20,50000)

(4,20,40000)

(5,20,30000)

(6,20,70000)

(7,20,90000)

create table bonus(

    empid int,

    bonus int

)

insert into bonus values

(1,100)

(2,100)

(4,100)

(6,100)

(7,100)

merge bonus as t

using (

select empid,salary from employee where deptid=20) as s

on (t.empid=s.empid)

when matched then

update set t.bonus=0.1\*s.bonus

when not matched then

insert (empid,bonus)

values (s.empid,0.05\*s.salary);

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

For each invoice, show the Invoice ID, the billing amount, the customer’s name, and the name of the customer who referred that customer (if any). The list should be ordered by billing amount

create table invoices(

                id int,

                billamount int,

                custid int

)

create table customer(

                id int,

                name varchar(50),

                referby int

)

INSERT INTO invoices values

(1,2477,101),

(2,456,145),

(3,1145,65)

INSERT INTO customer values

(101,"jack",167),

(102,"lion",145),

(103,"harry",65),

(145,"Sam",101),

(65,"Klion",101)

QUERY

select i.id,i.billamount,c.name,r.name "referred"

from invoices i

join customer c on i.custid=c.id

left join customer r on c.referby=r.id

OUTPUT

1-2477-jack-null

2-456-sam-jack

3-1145-klion-jack

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

**DUPLICATE ROWS**

create table customer(

                id int,

                name varchar(50),

                referby int

)

INSERT INTO customer values

(101,"jack",167),

(102,"lion",145),

(103,"harry",65),

(145,"Sam",101),

(65,"Klion",101),

(65,"Klion",101),

(145,"Sam",102),

(145,"Jam",101)

DISPLAY ONLY THE DUPLICATES ROWS - along with count

SELECT \*,count(\*) from customer group by id,name,referby having count(\*)>1

DISPLAY EACH ROW ONCE - if duplicate display once

SELECT \* from customer group by id,name,referby having count(\*)=1

UNION

SELECT \* from customer group by id,name,referby having count(\*)>1

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

**HIVE**

**HIVE**

--NOT WORKING USE CROSS-APPLY

select inner\_table.\* from(

select \*,rank() over (partition by customerid order by orderdate desc) as rnk from [order] o

join customer c on o.id=c.id

)as inner\_table where inner\_table.rnk=1

--CROSS APPLY -- as any alias (here "rnk") cannot be used in where clause

select \*,rnk from [order] o cross apply (select rank() over (partition by customerid order by orderdate desc) as rnk)inner\_

join customer c on o.id=c.id

where rnk=1

--EXPLODE AND MAP

select explode(map('emp\_x',empno,'code\_x',code,'descrip\_x',descrip)) as (mykey,myval) from sample\_07

INPUT

sample\_07.code sample\_07.cde sample\_07.descrip sample\_07.empno

00-0000 NULL 134354250 40690

11-0000 NULL 6003930 96150

OUTPUT

mykey myval

emp\_x 40690

code\_x 00-0000

descrip\_x 134354250

emp\_x 96150

code\_x 11-0000

descrip\_x 6003930

====================================================

--LATERAL VIEW / EXPLODE AND MAP

select code,description,ex.mykey,ex.myval from sample\_07

lateral view

explode(map('total\_emp\_x',total\_emp,'salary\_x',salary))ex as mykey,myval

order by description

--COALESCE

select code,total\_emp,salary,sum(total\_emp+salary) as sum\_1 from sample\_07 group by code,total\_emp,salary

having sum\_1 is null

select code,total\_emp,salary,sum(coalesce(total\_emp,0)+coalesce(salary,0)) as sum\_1 from sample\_07 group by code,total\_emp,salary

having sum\_1 is null

--wrong query -- salary=null

select code,total\_emp,salary,sum(coalesce(total\_emp,0)+case(salary) when null then 0 else salary end) as sum\_1 from sample\_07 group by code,total\_emp,salary

having sum\_1 is null

-- correct query -- when salary is null

select code,total\_emp,salary,sum(coalesce(total\_emp,0)+case when salary is null then 0 else salary end) as sum\_1 from sample\_07 group by code,total\_emp,salary

having sum\_1 is null