How to boost the performance

1-Only connect to the data that you need for analysis.

2-Extracts are a quick and easy way to make most workbooks run faster.

3-Allow your users to incrementally drill down to details, rather than trying to show everything then filter (guided analysis).

4-Use filters, hide unused fields and aggregate.

5-Strings and dates are slow, numbers and Booleans are fast.

7-Upgrading to the newest version may boost performance without needing anything else.

8-For mobile phone, a resolution of 320×568

9-For tablet users, we’ll go with a 768×1024 display (18.52%).

10-computer users, we’ll use 1366×768 (7.4%).

11-Avoid high mark counts.

12- better alternative to a blended calculation would be to prepare a new view on the database server.

13- the command ORDER BY is important when producing a result set directly from SQL

14-New data source = data blending

Add in the data is joining

15-**Set the dashboard layout to a fixed size.**

**16-**Use Dashboard Actions instead of Quick Filters for better performance.

17-avoid using Tableau Desktop and Server in the same machine as it may lead to performance issues unless there is enough hardware present to balance the load.

18-**Aggregate Data if Possible**

**19-Show Apply Button on Quick Filters.**

**20-**Analyze Workbook Performance

21-**DO COMPLEX CALCULATIONS IN DATABASE**

**22-AVOID USING ONLY RELEVANT VALUES.**

**23-**show apply button” to help with performance.

Our short list of five features in Tableau 10 that we’re most excited about.

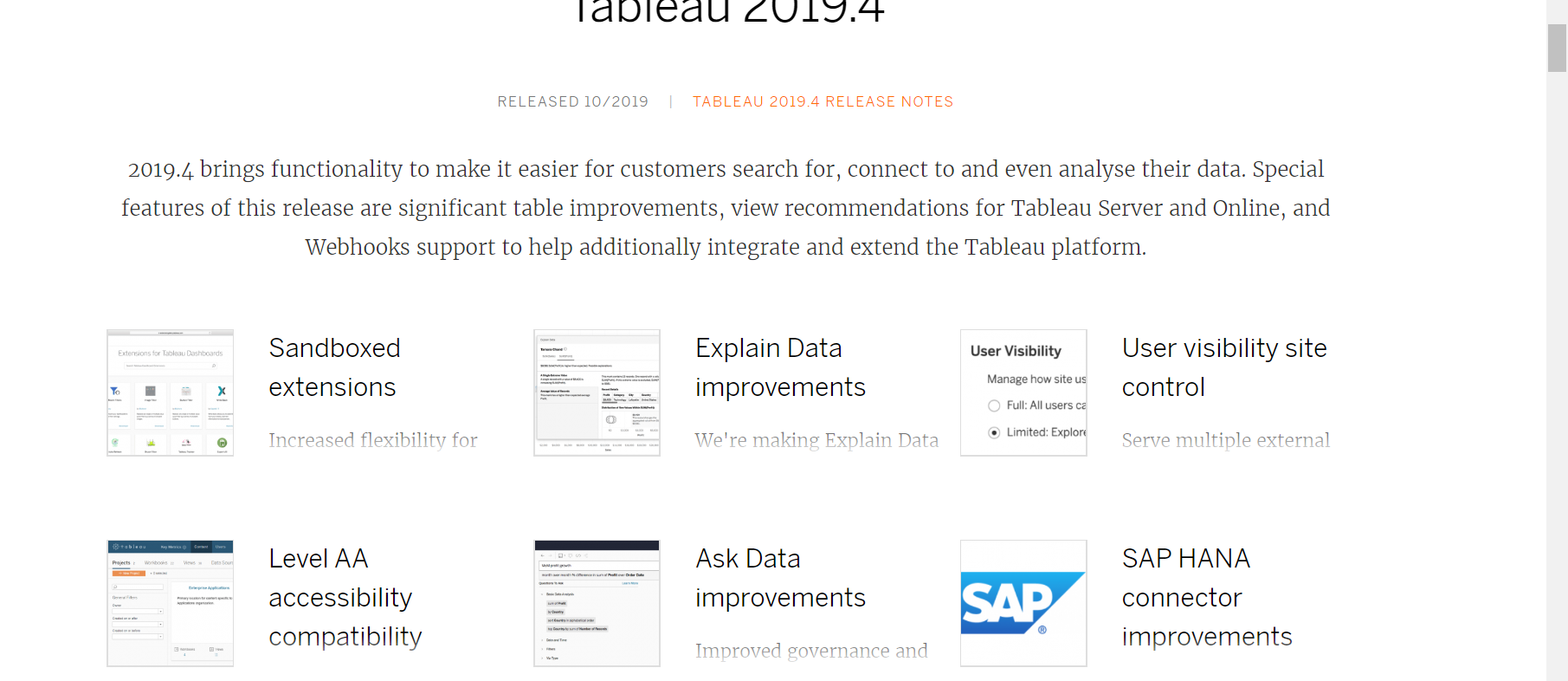
1-**#5. Workbook Formatting**

**2-#4. Custom Territories**

**3-#3. Data Highlighter**

**4-#2**. **Cross Data Source Filtering**

**#1.** **Wildcard Union and Cross Database Joins**



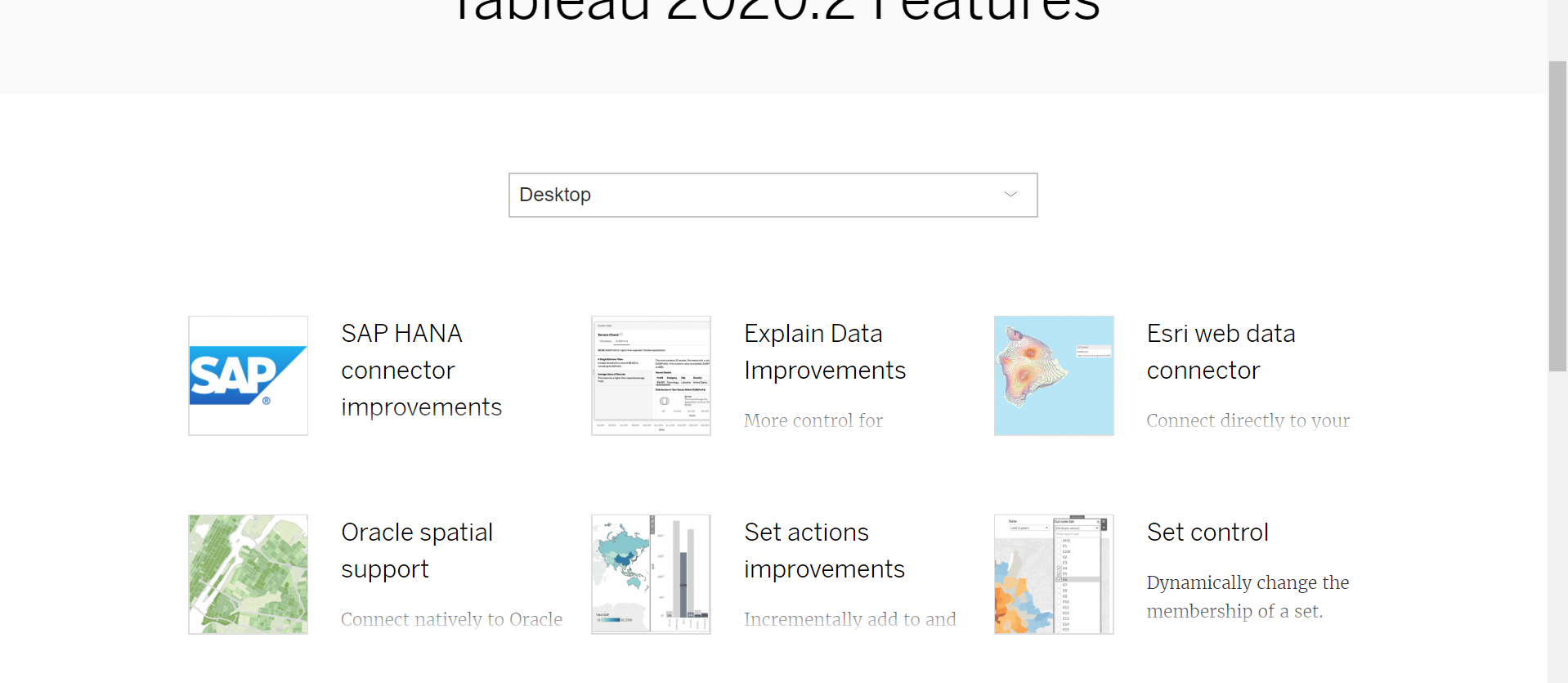
Ali baba connector

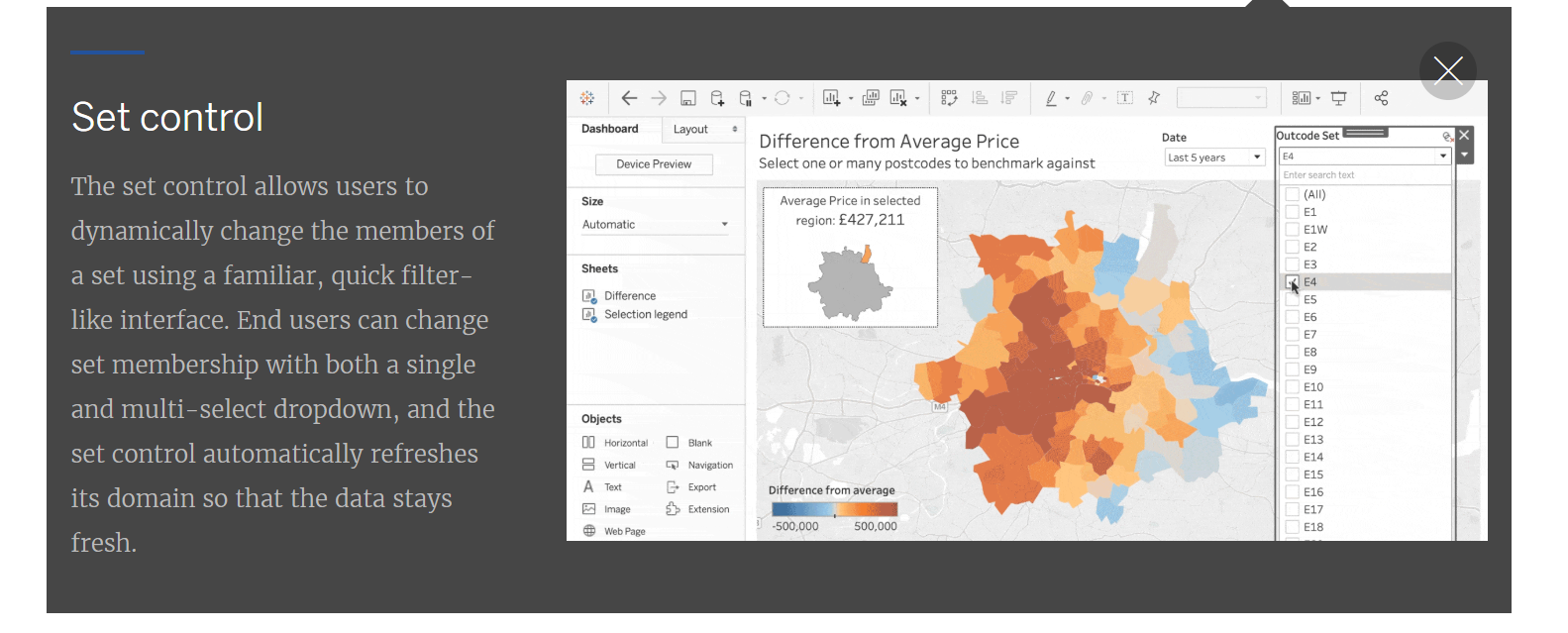
Linkdin sales connector

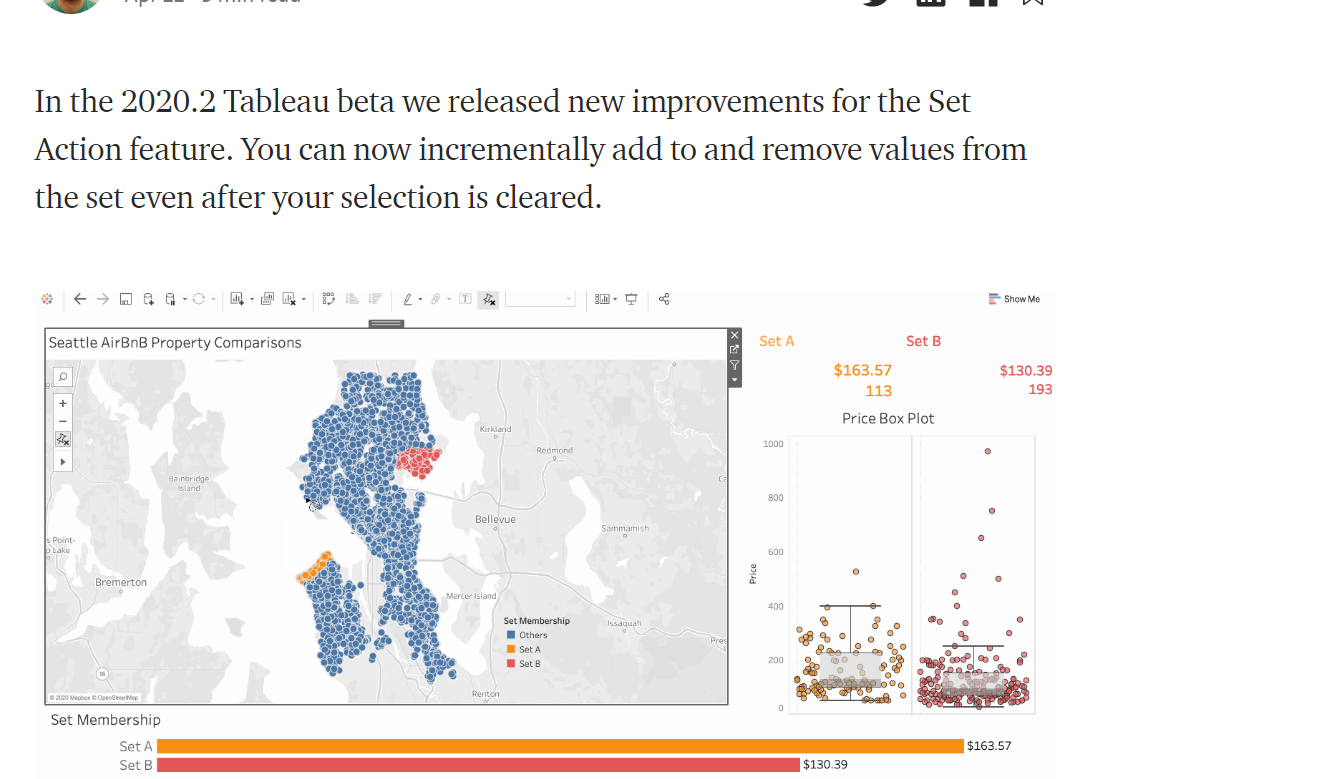
Kyvos/quadrable connector

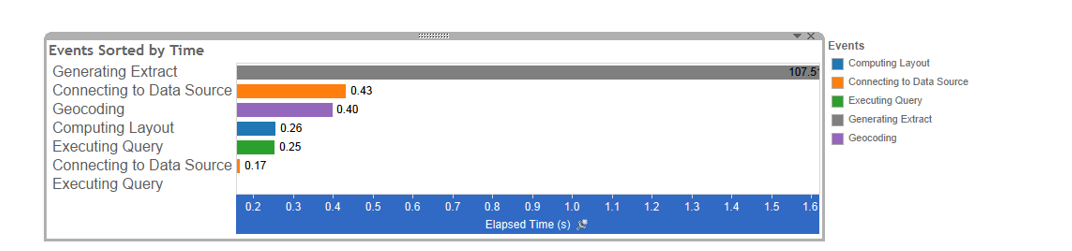
New features in 2020.2

Users upgrading to 2020.2 will get Metrics – a feature which allows users to instantly monitor key performance indicators (KPIs) from a mobile device.

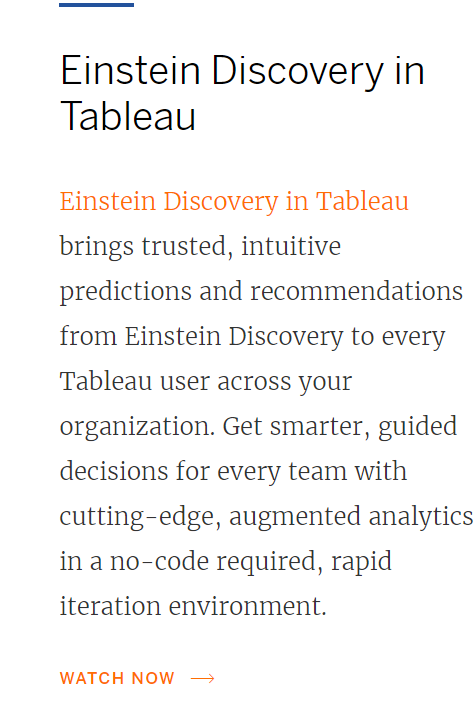








# Tableau 2021.1



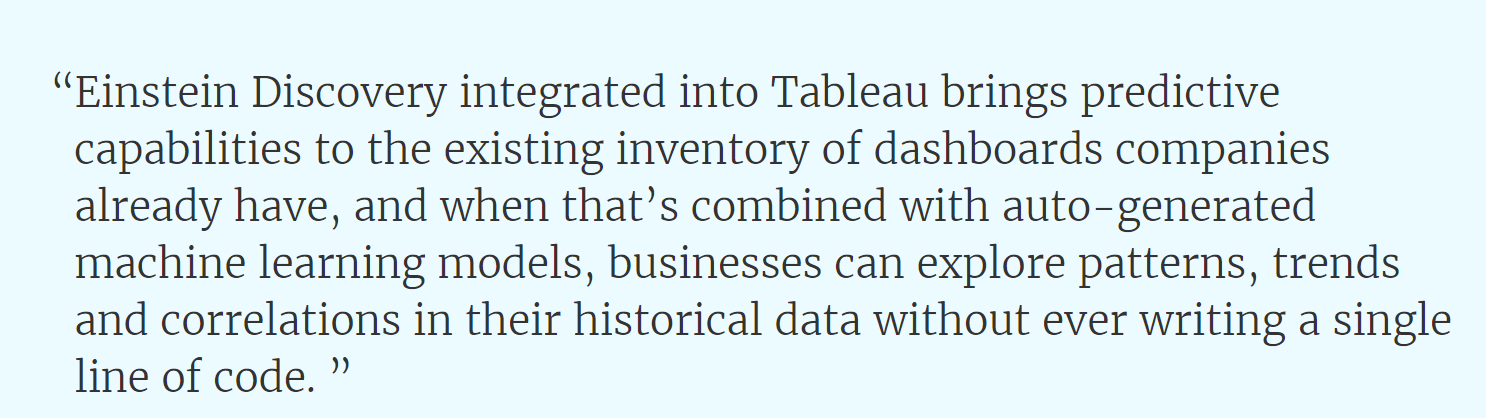
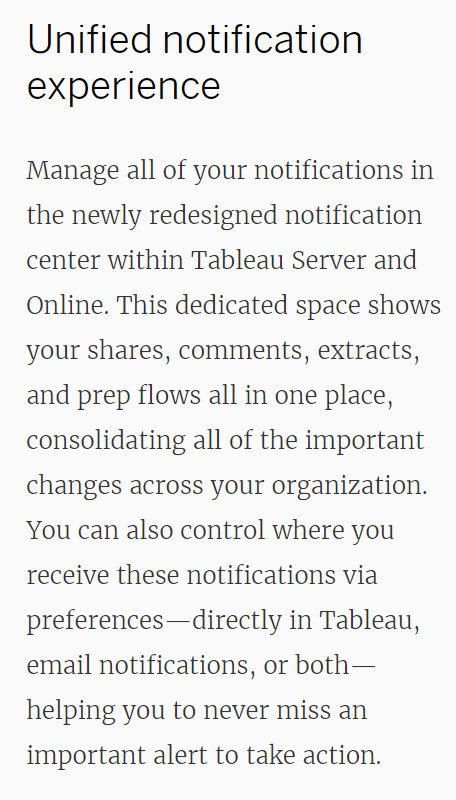
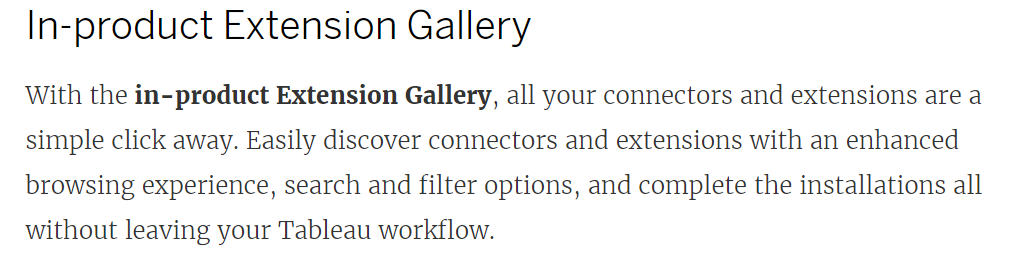


Tableau CRM analytics studio it will create that model.

Quick LODs

Create Level of Detail expressions more easily. Use context menus or drag-and-drop a measure onto a dimension to automatically create a Level of Detail expression with the default aggregation.

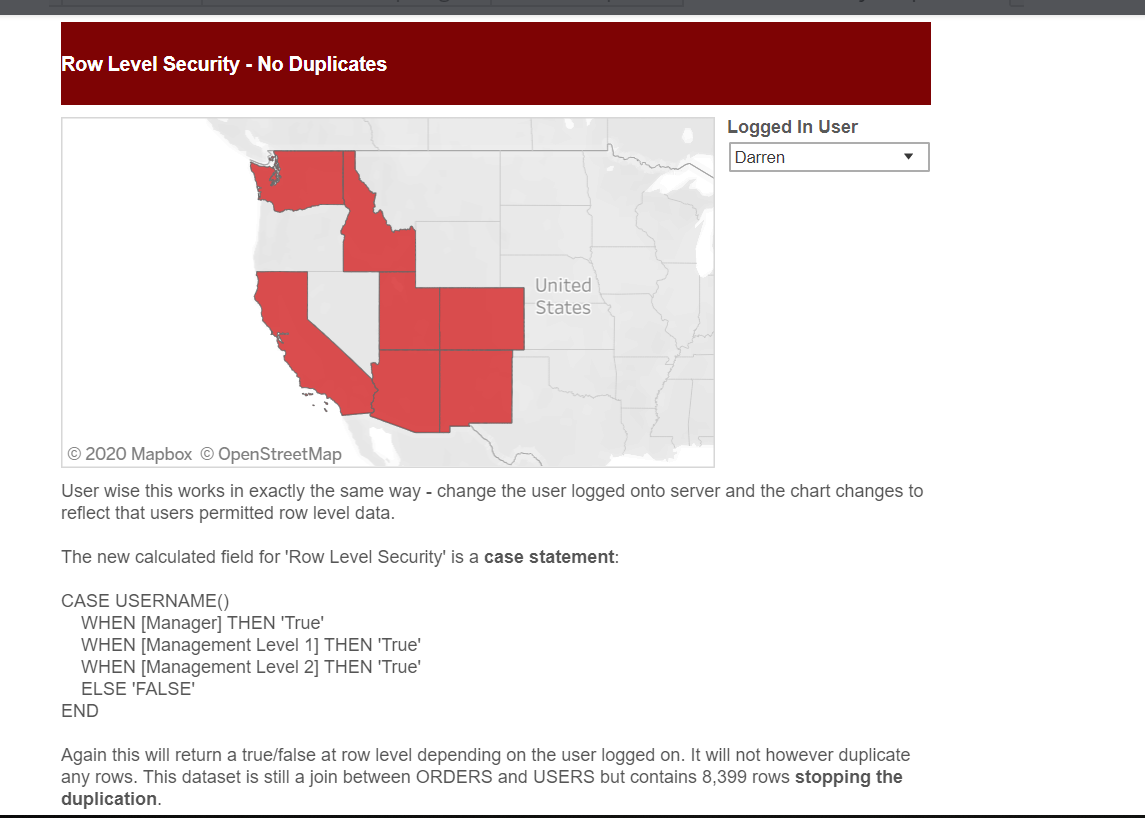


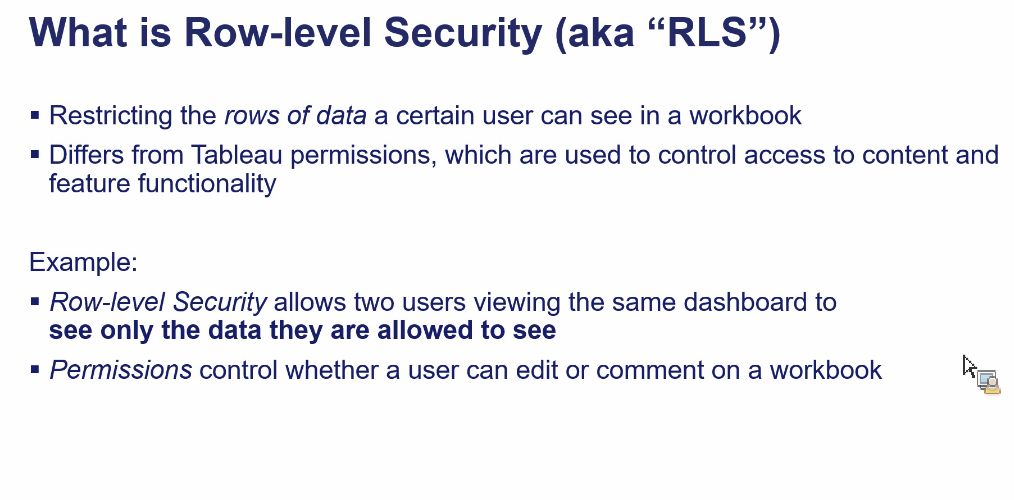
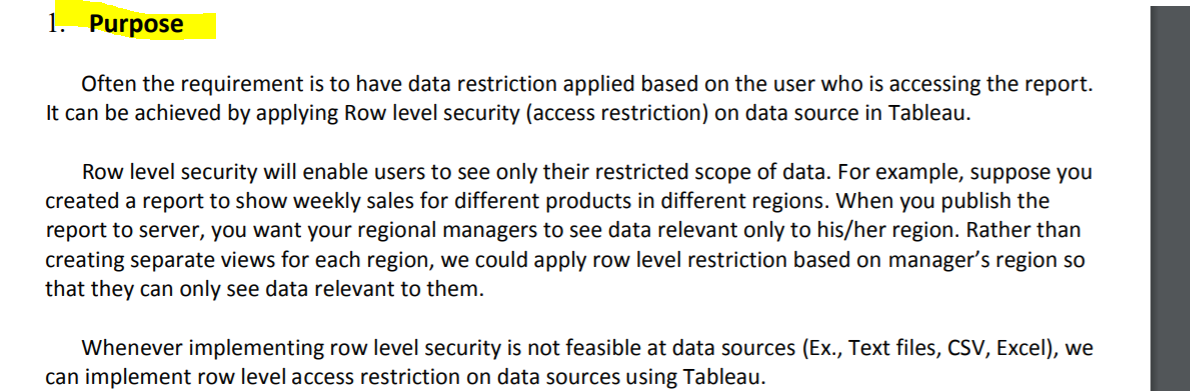


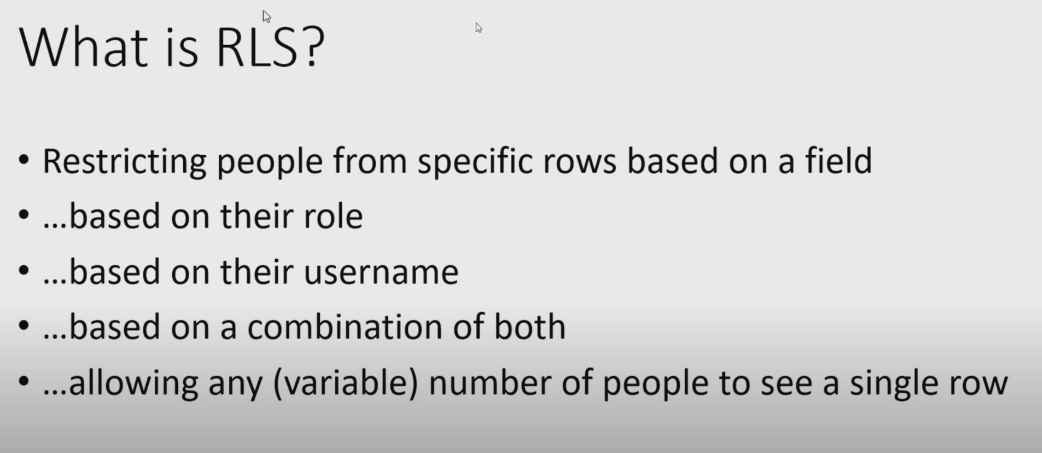
#### **Tableau Dashboard Actions**

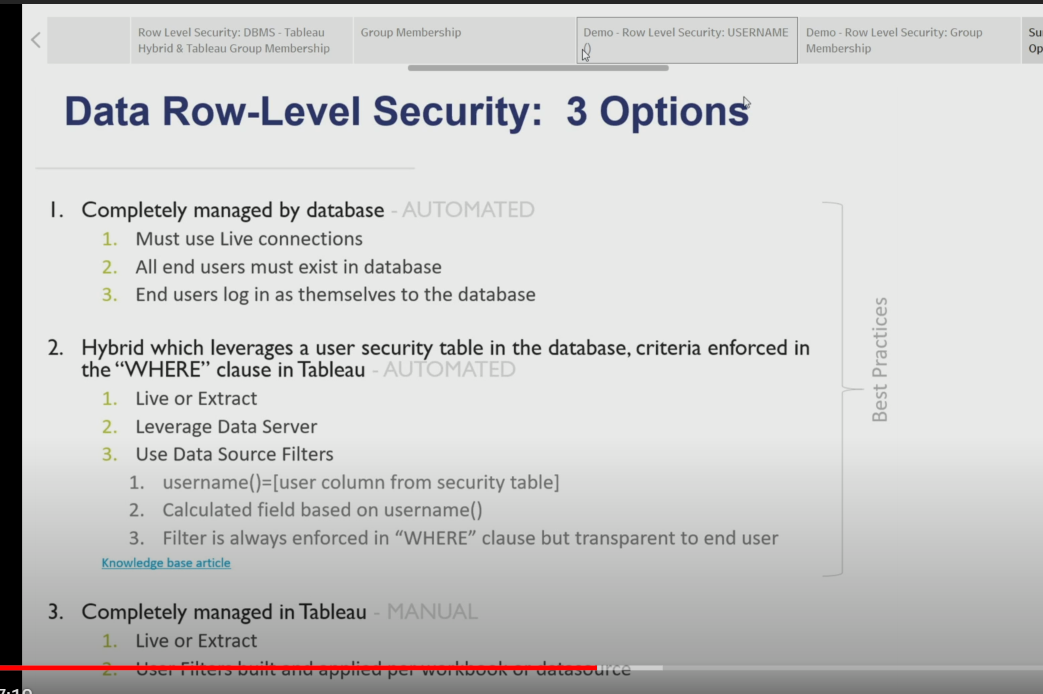
1. Hover – Example: If you hover over the source sheet, the action will take place on the target sheet.
2. Select – Example: If you click on the source sheet, the action will take place on the target sheet.
3. Menu – Example: If you hover over the source sheet, a menu of dashboard actions will appear in the tooltip. Clicking on one of the menu items will execute the action on the target sheet.
4. Leave the filter – leaves the last dashboard action that happened in place. If you filtered sheet two by something clicked on sheet one, the filter on sheet two will stick.
5. Show all values – reverts back to the original view as if no dashboard action took place.
6. Exclude all values – clears everything off of the target sheet, meaning the target sheet will not show unless a dashboard action is executed.

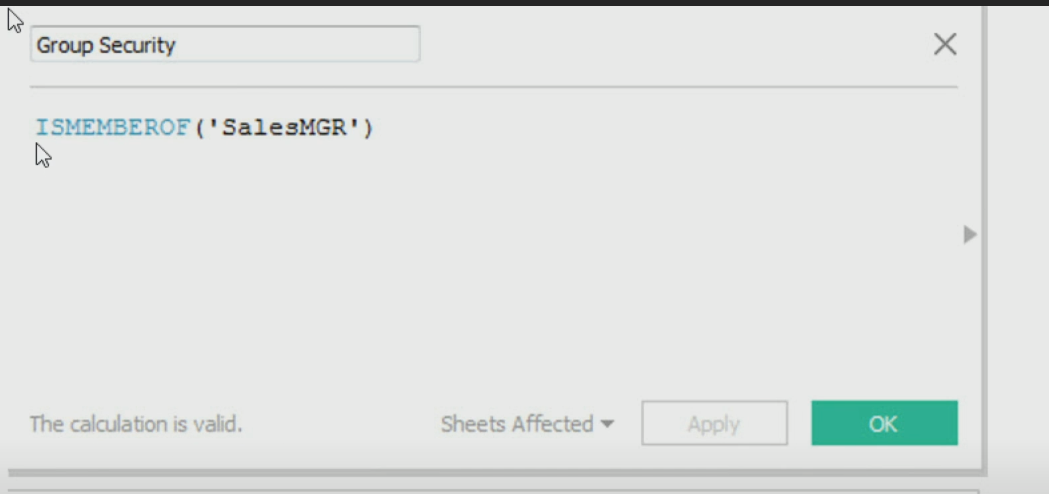
Row-Level Security (RLS) in Tableau



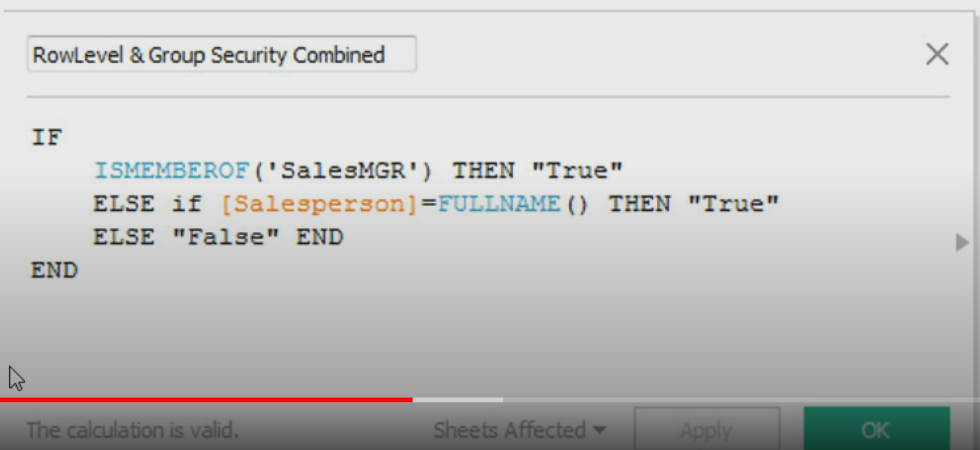












**Database**

It is clearly evident that LOD expressions do not directly affect the performance, and instead, there are other factors involved. If the database in consideration is a super-fast database (Live or Tableau data extract) and intelligent enough to perform parallel queries, then an introduction to LOD would be absolutely fine. If the database (Live or Tableau data extract) is not optimized for performance then you would have an issue with the regular queries and, in this case, it would be unfair to say that the slow performance is due to LOD.

**Table Calculation**

The VizQL component of Tableau performs these calculations, and it is not fair to compare this to a database. VizQL is simply helping us with some calculations at the client level rather than sending it back to the database, and the performance will depend on the number of “marks” being displayed in the chart or table. If the number of marks is too great then VizQL will be under tremendous load.

Hence the LOD must be used if we are sure that the database is capable of handling sub-queries intelligently. If the database is not optimized for the same, then we can consider alternatives like Table Calculations, however, these will work efficiently with fewer marks. There may be limitations with Table Calculations as well since, unlike LOD, they cannot compute aggregations that are not at the level of detail of the visualization.

SUM([Sales]) / ATTR({FIXED : SUM([Sales])})

This calculation will give you the ratio of a state’s sales to total sales.

If you then put [**State**] on the Filters shelf to hide some of the states, the filter will affect only the numerator in the calculation. Since the denominator is a FIXED level of detail expression, it will still divide the sales for the states still in the view against the total sales for all states—including the ones that have been filtered out of the view.

INCLUDE and EXCLUDE level of detail expressions are considered after Dimension filters. So if you want filters to apply to your FIXED level of detail expression but don’t want to use Context Filters, consider rewriting them as INCLUDE or EXCLUDE expressions

,

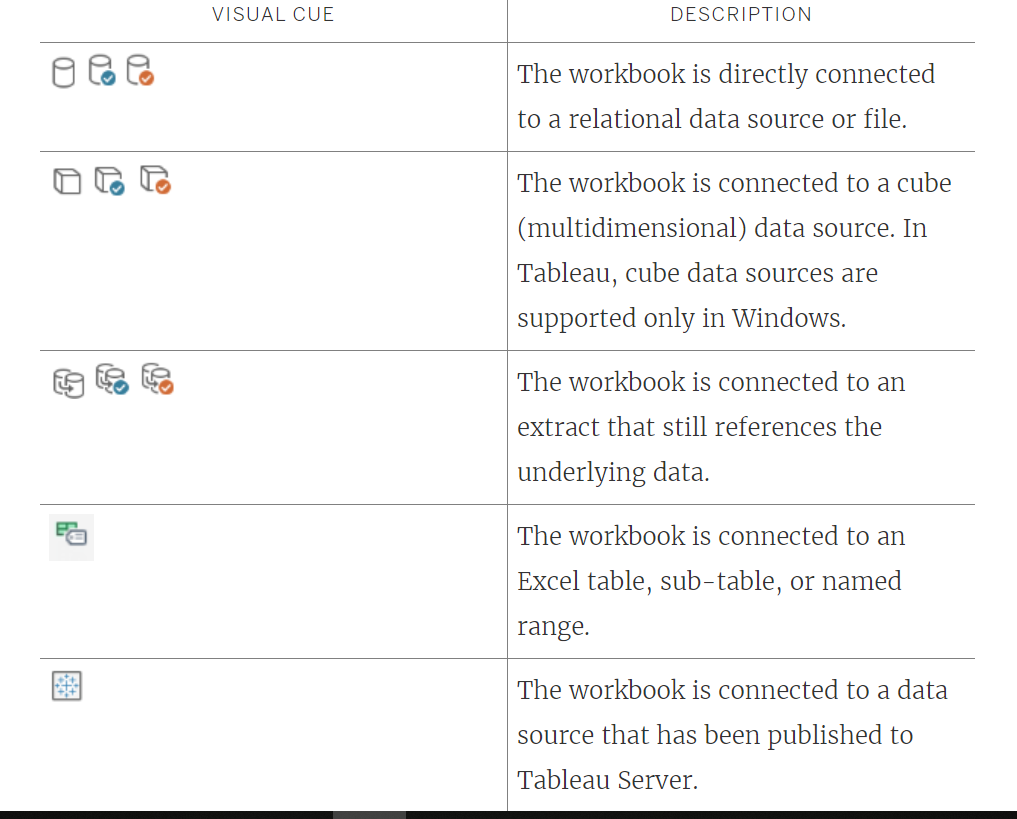


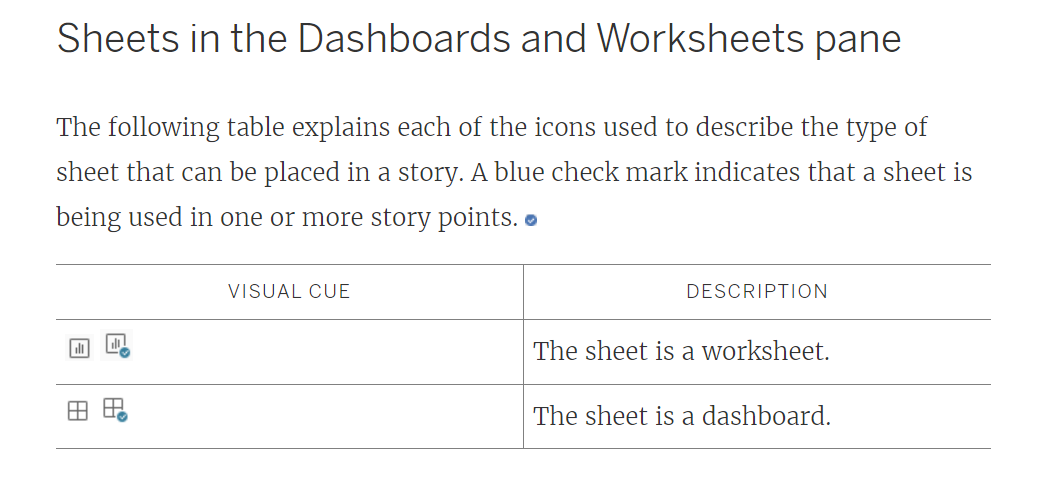
1-So, maybe you join in some form of code, say “Security Group 1,” to your particular row and then create a calculated field that maps the username to a security group. Lastly, a calculated field comparing the two OR **username() =** .

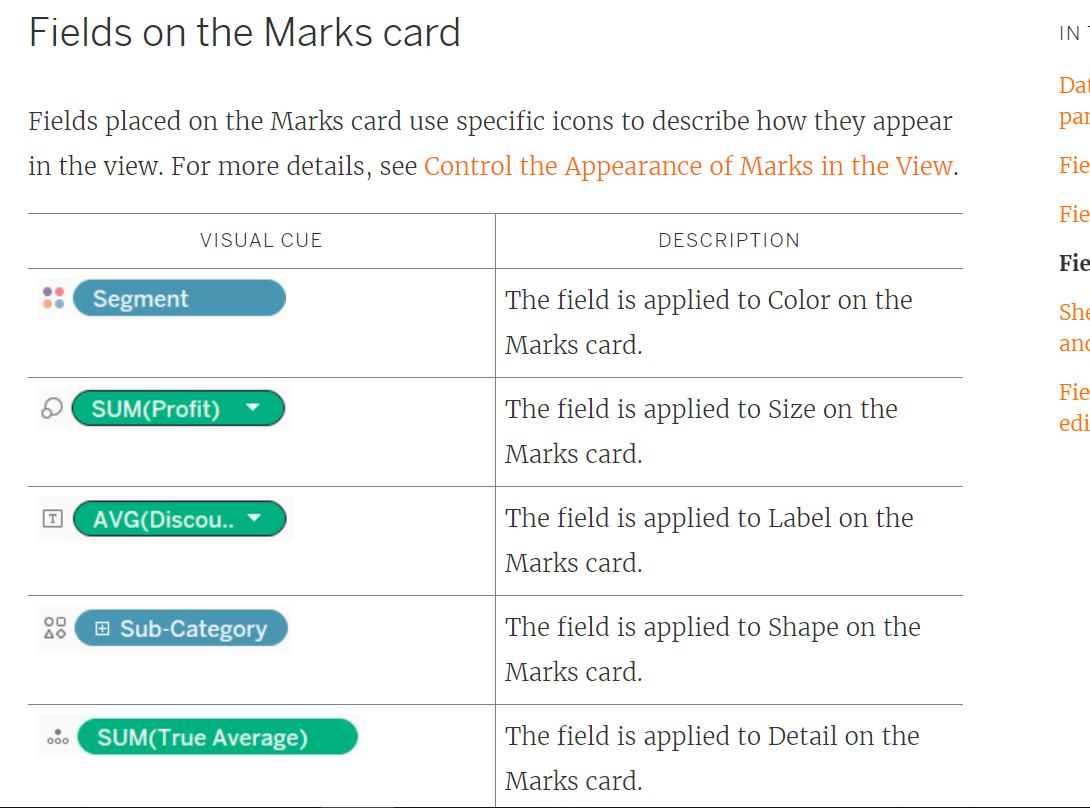
SET & SET ACTION:

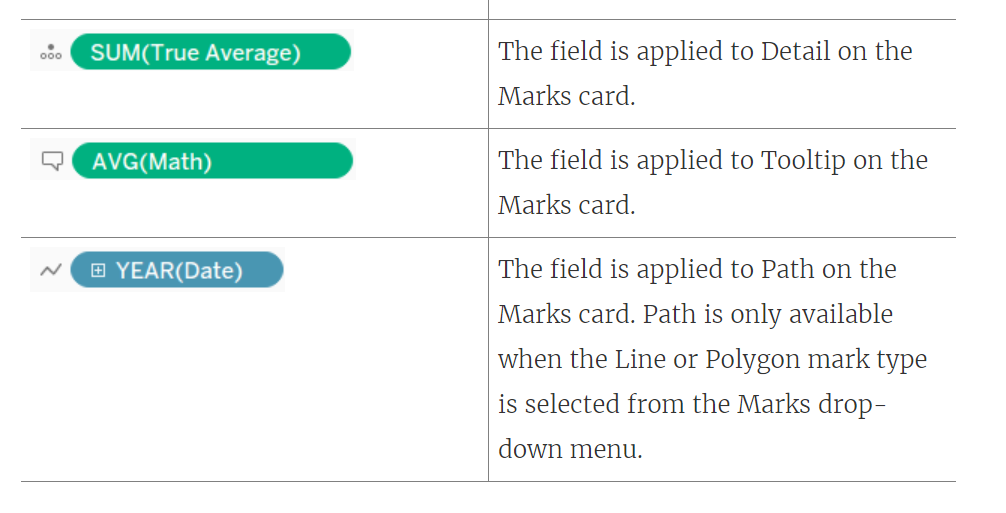
 The group is only interesting because of the teams being examined. While filters always select the value the user picked, sets can select other fields based on their relationship to the user selection.

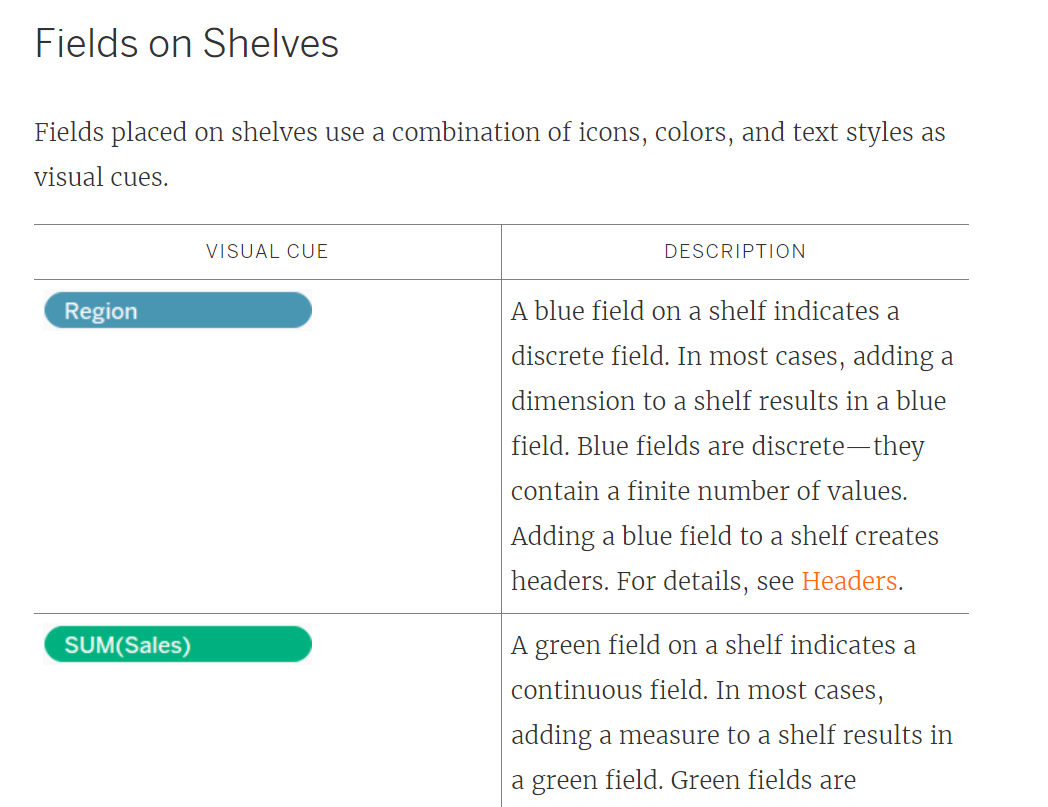
Tableau ICON—

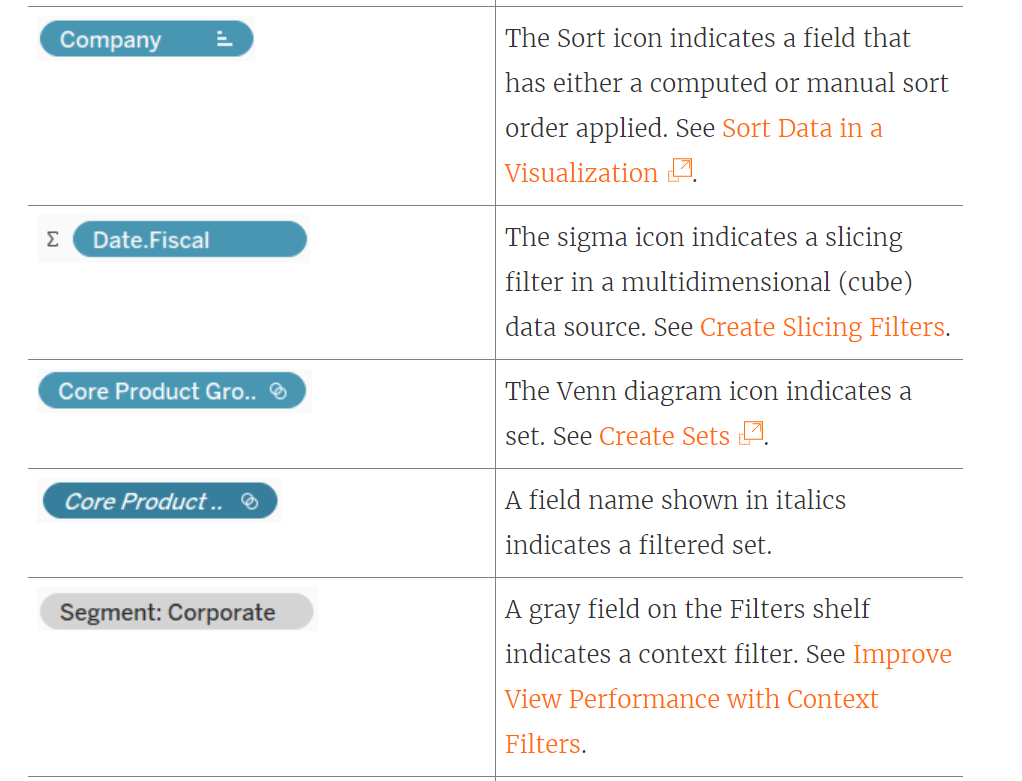


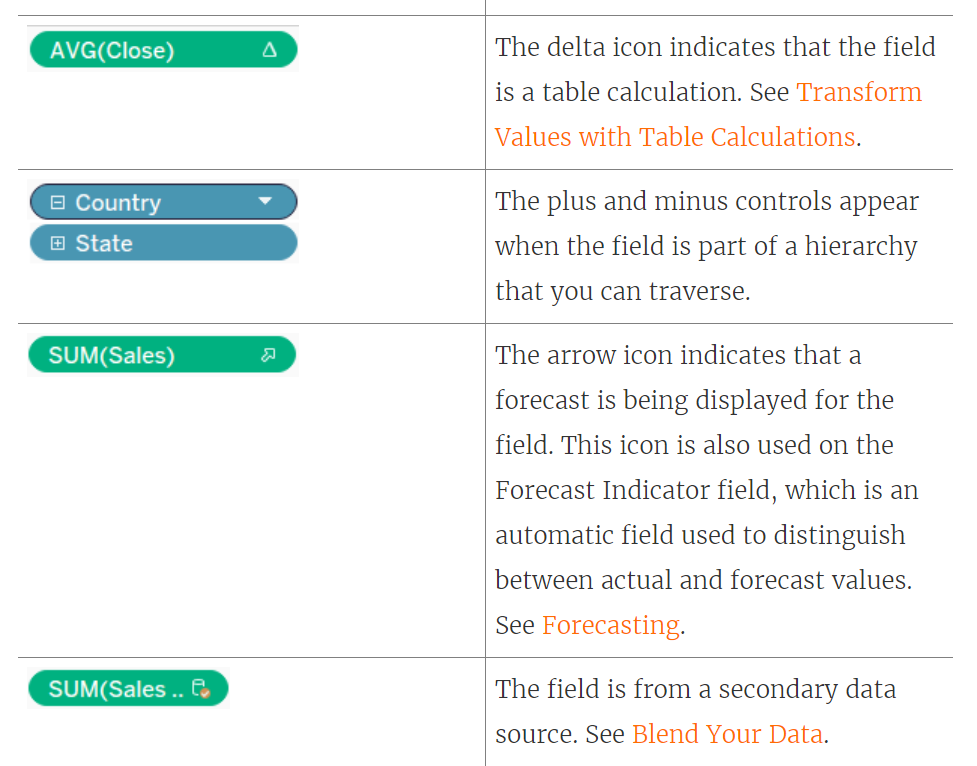


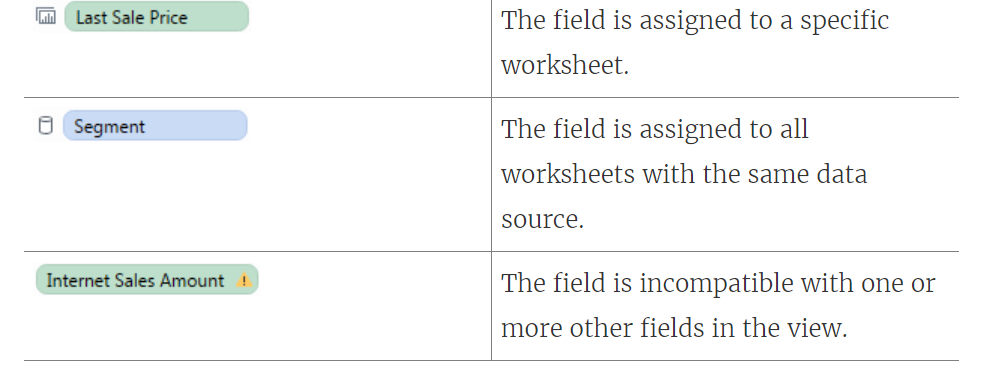


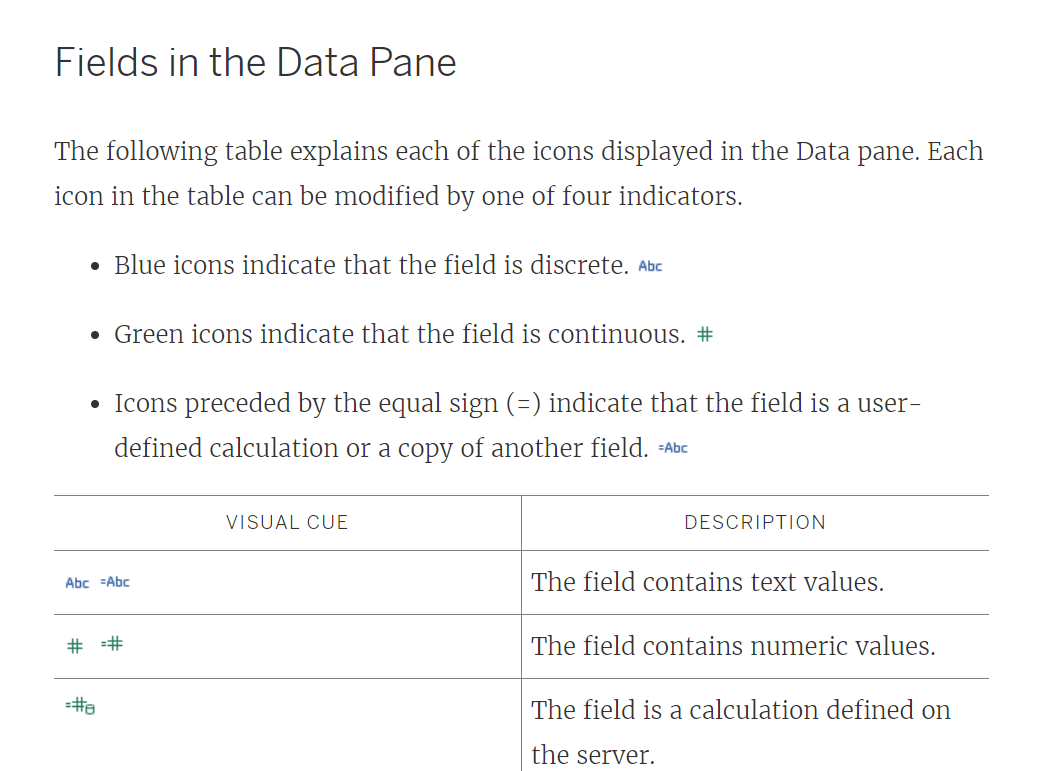


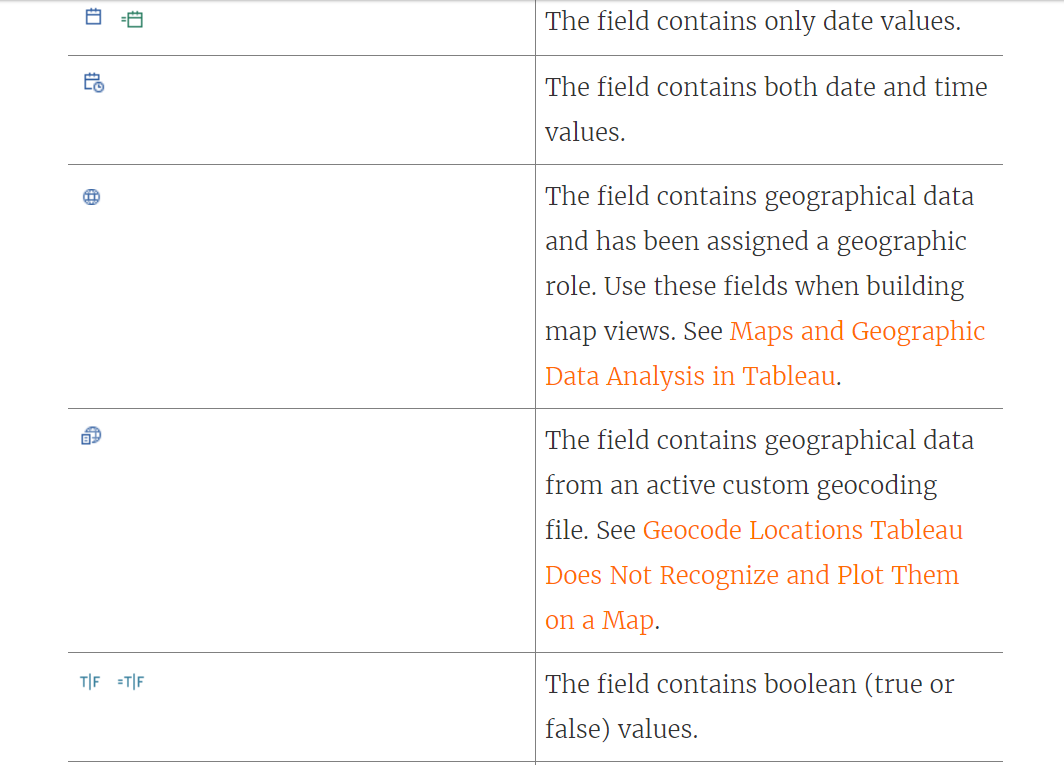








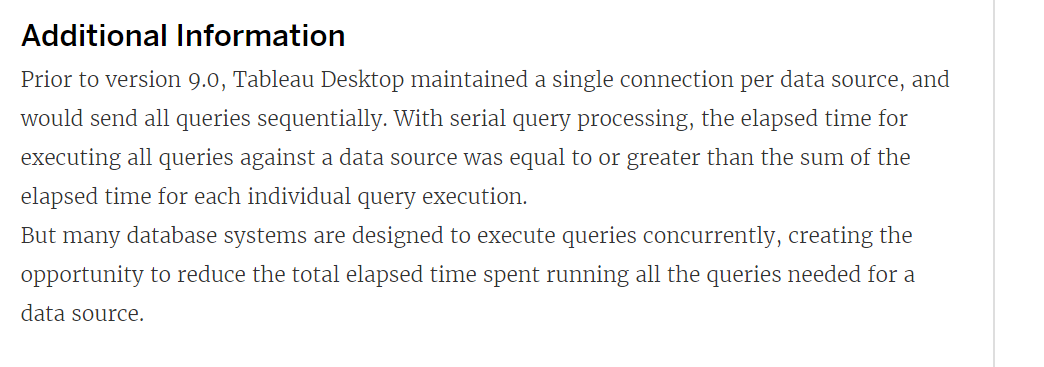


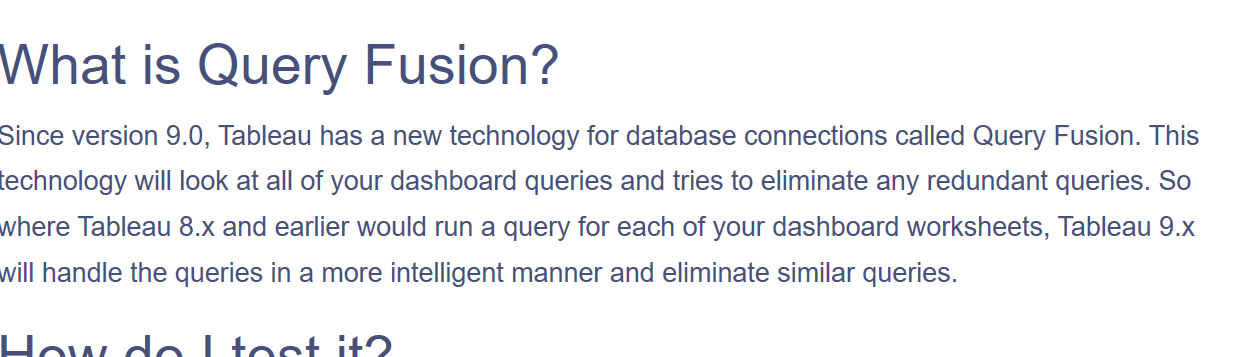


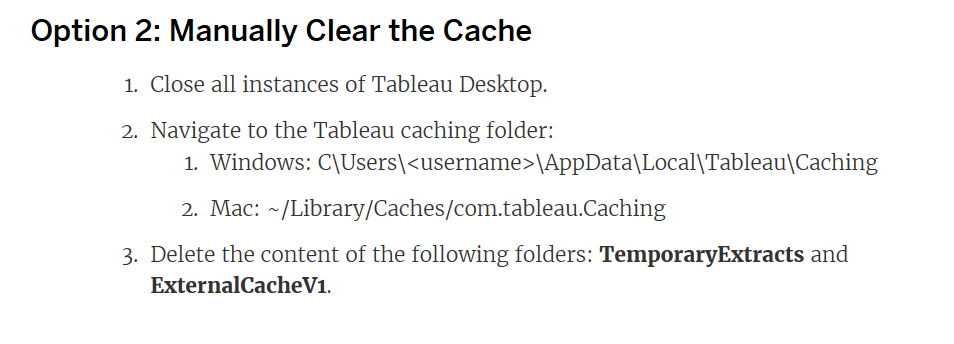


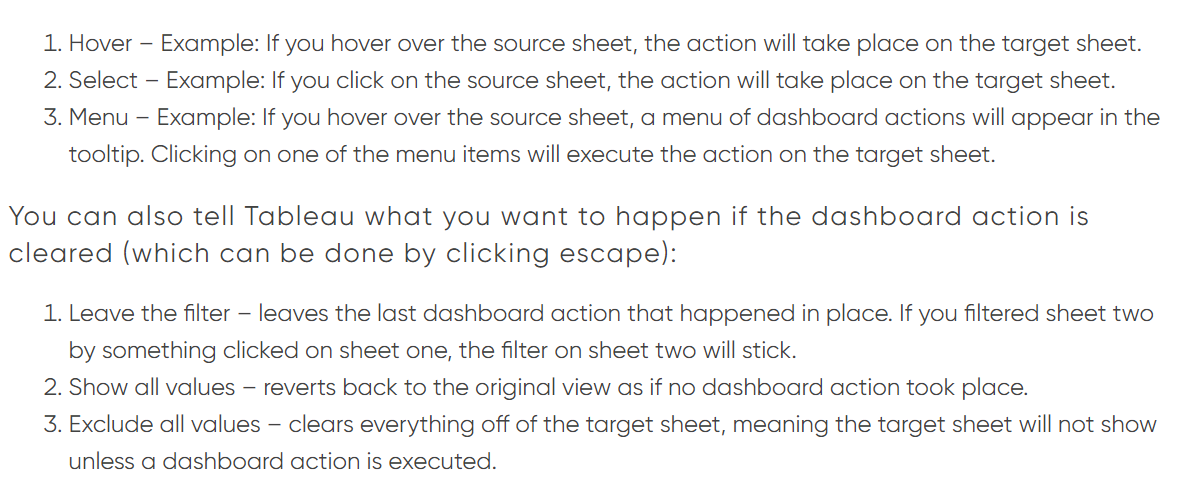


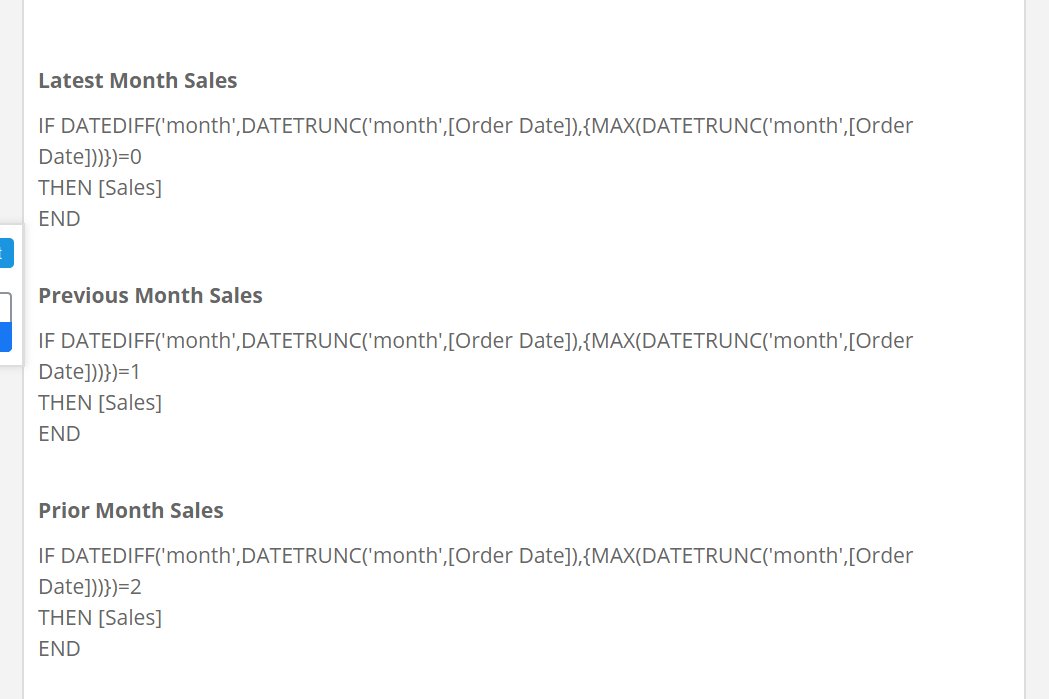
## **Parallel Queries in Tableau Desktop**

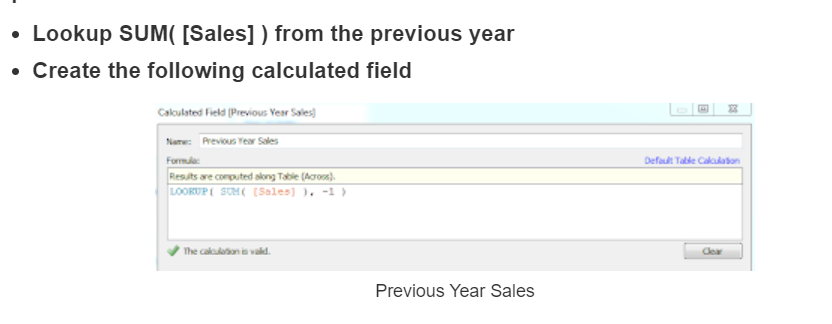


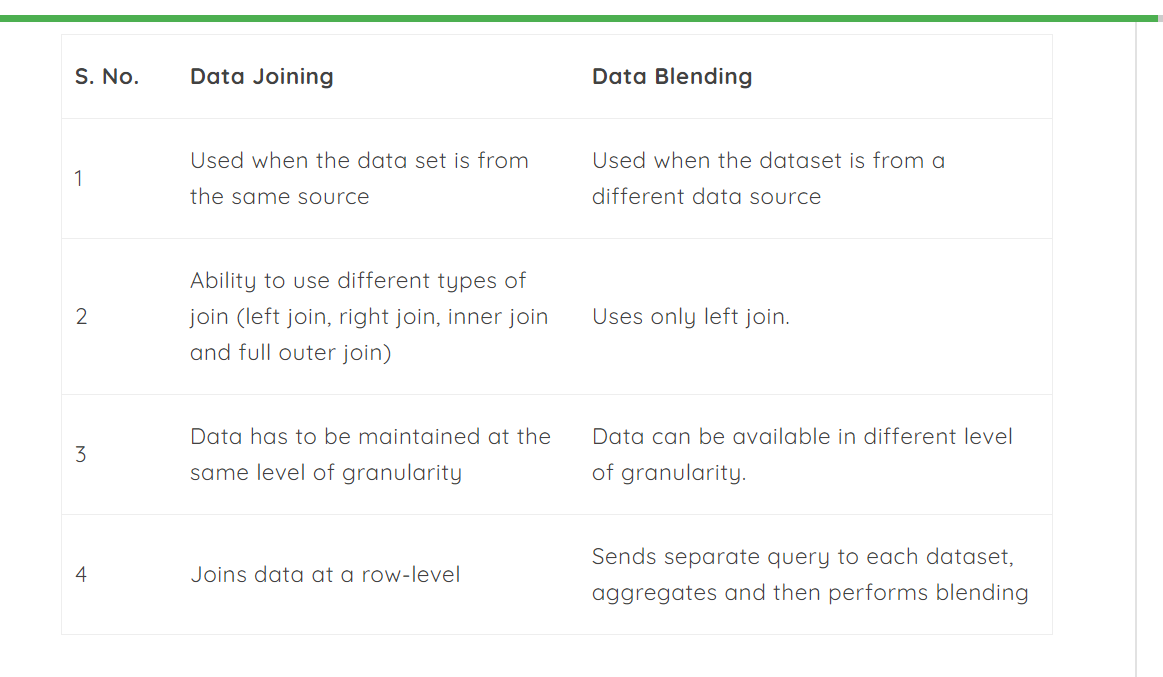


****









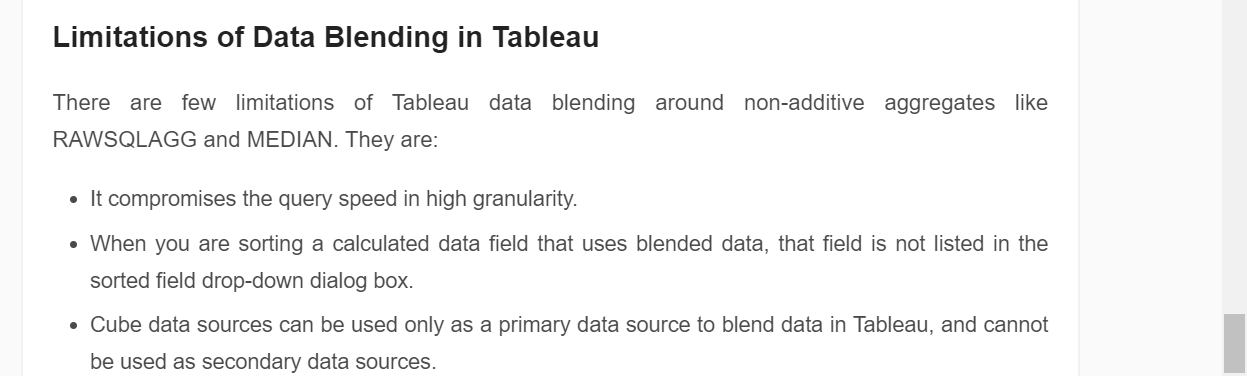
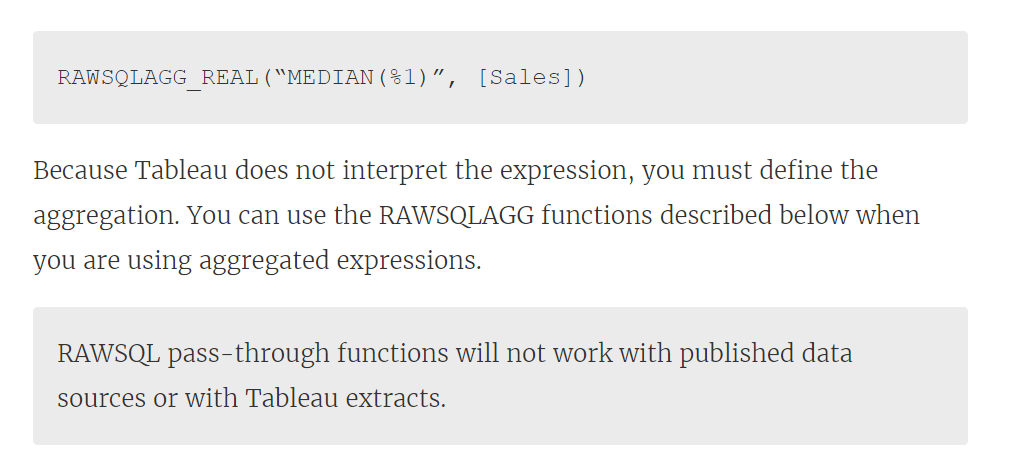
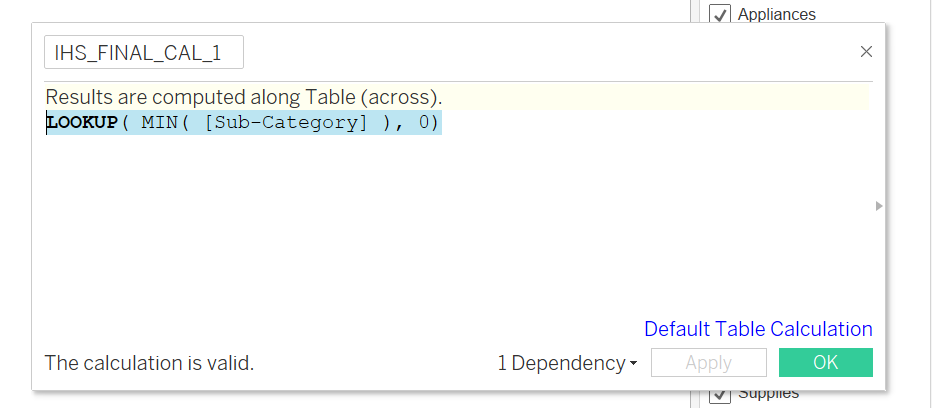
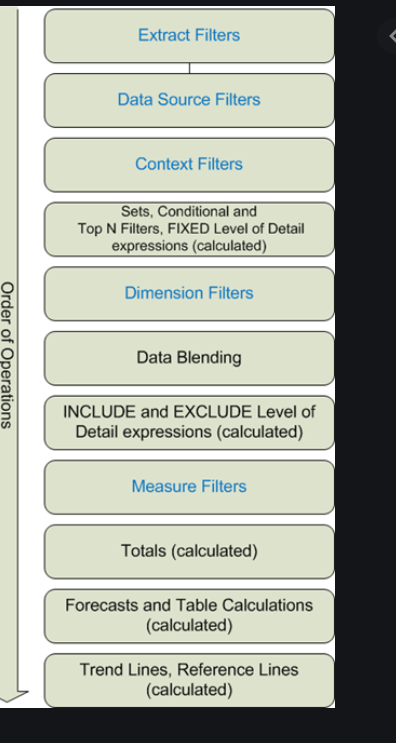


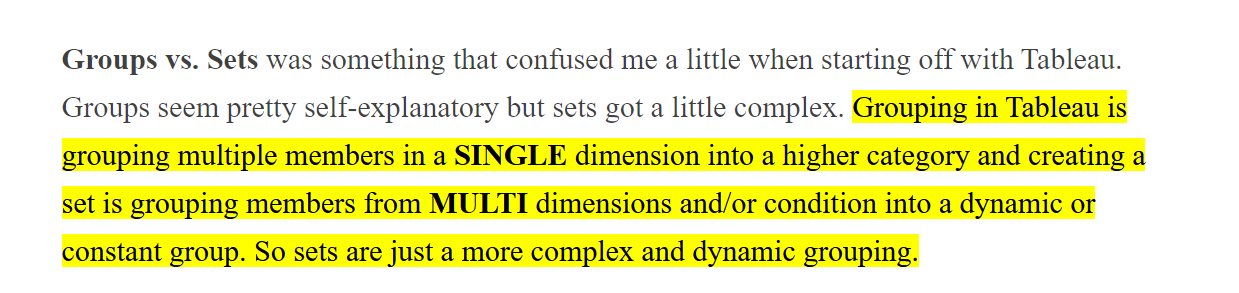
Table across-100%=cell

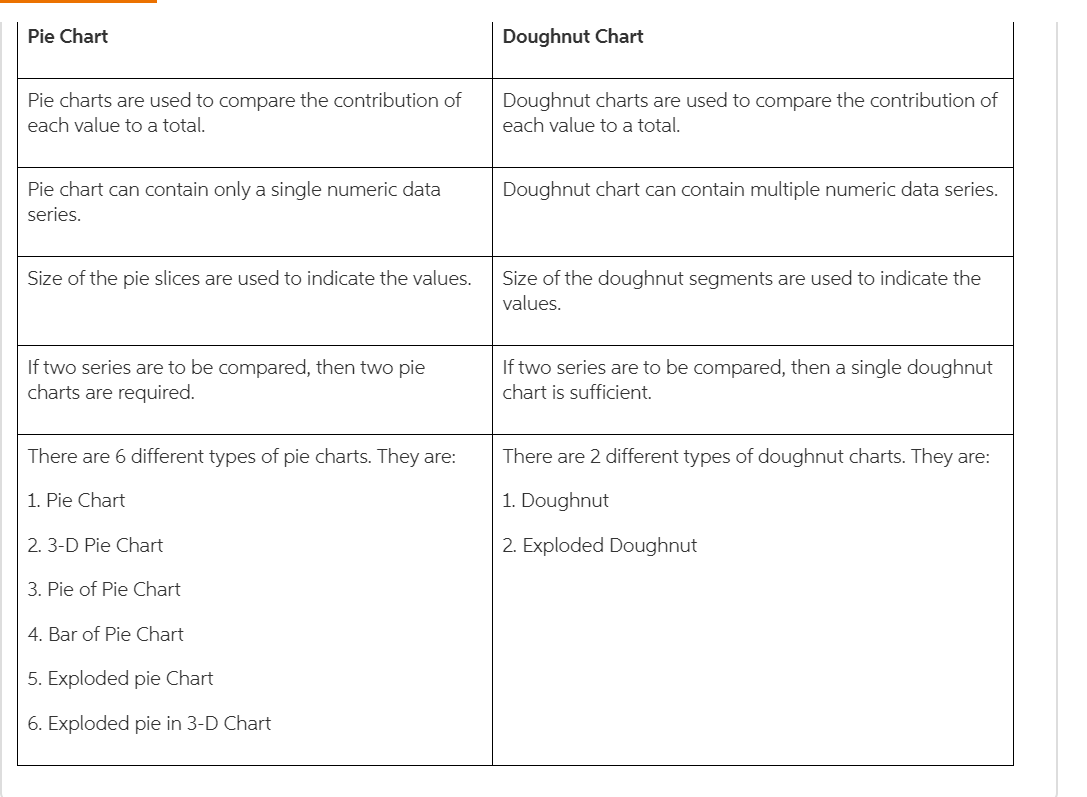
TABLE (DOWN)-AS PER THE DATA=Table=subcategory



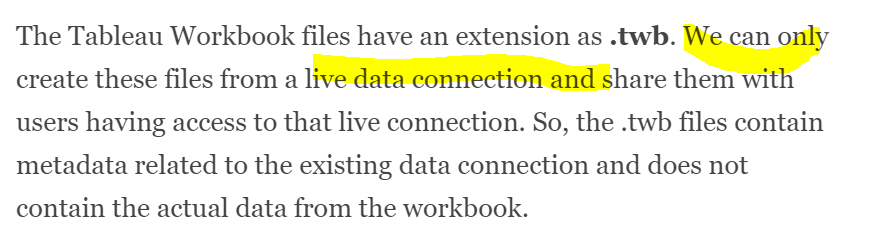


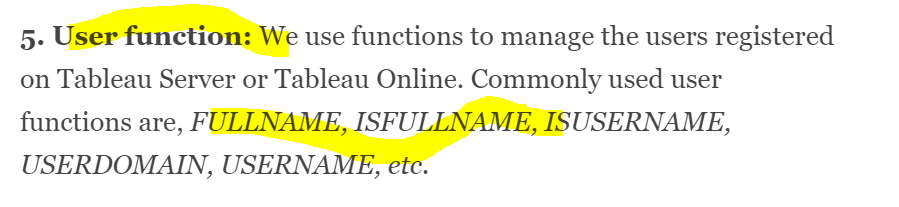


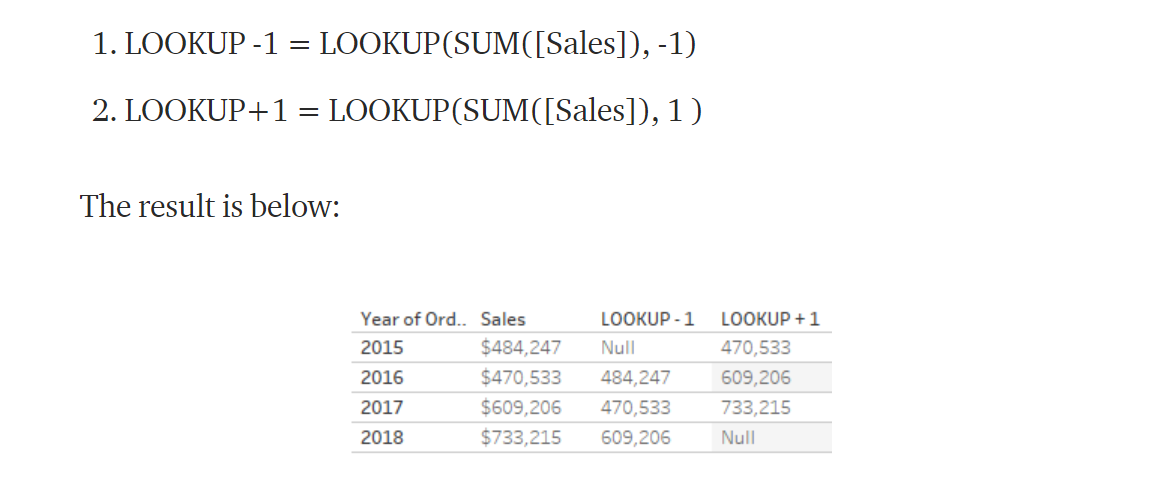


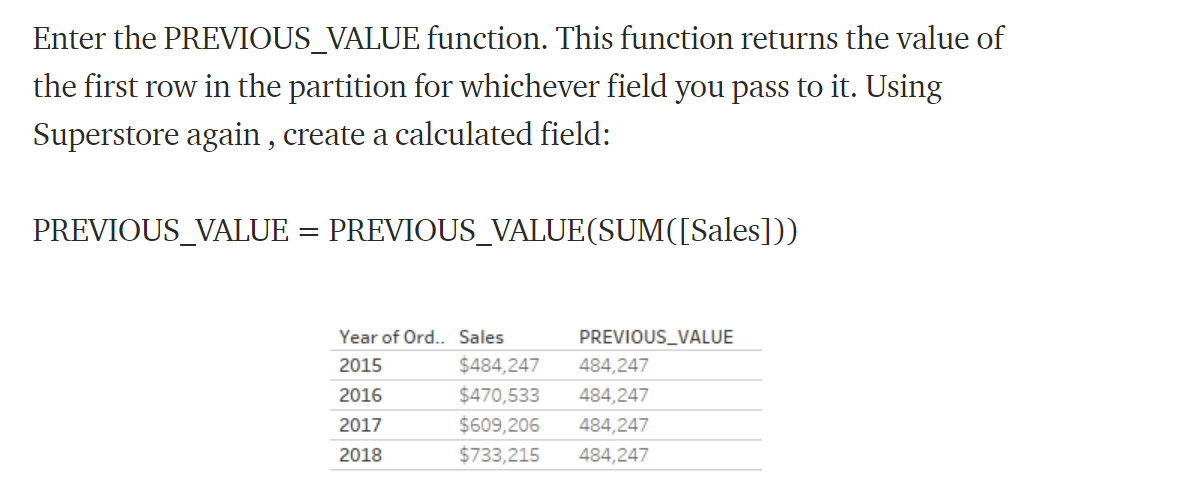






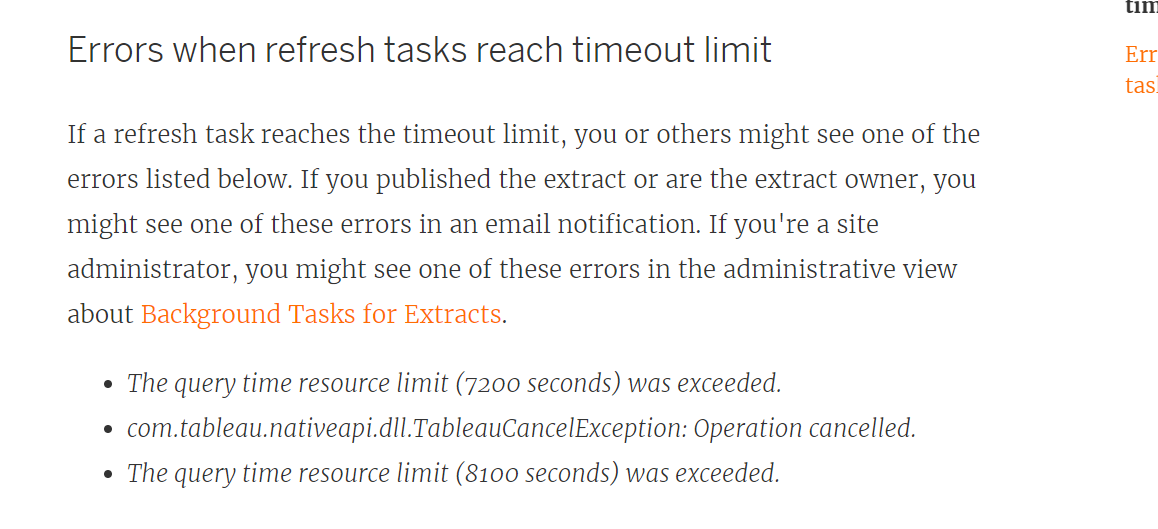


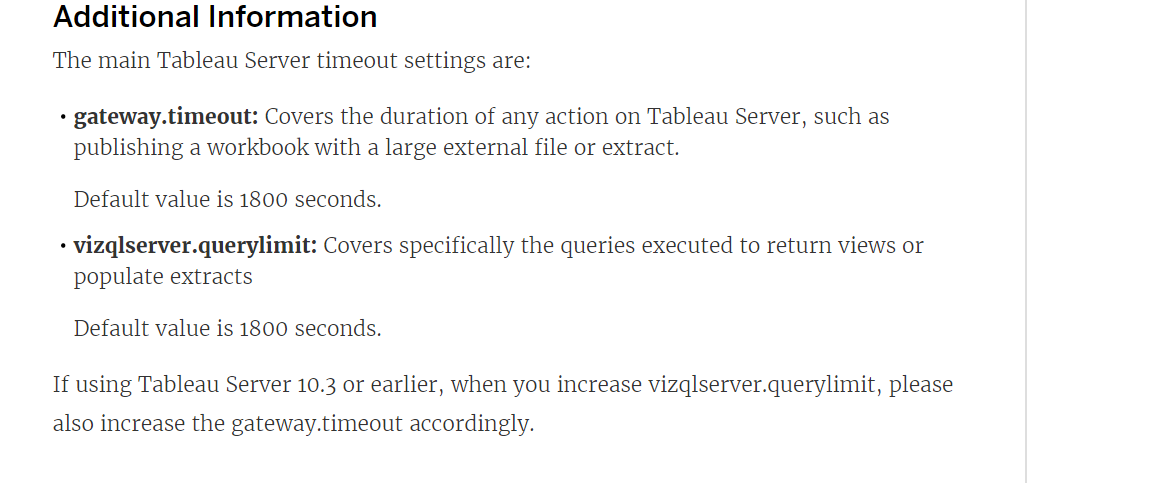




Biggest challenge ever u faced?

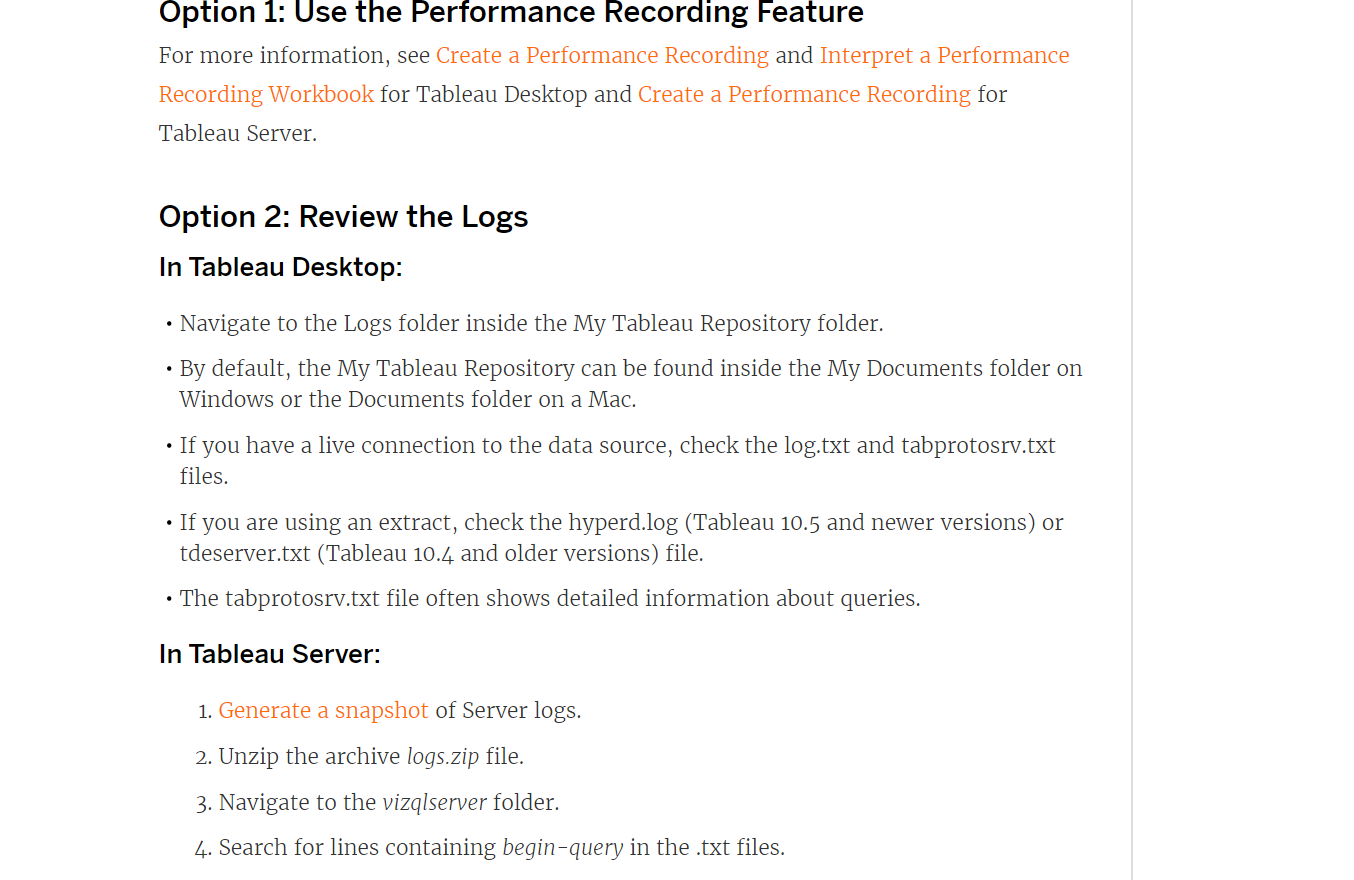
1-Recently I have been working with the monitoring team to suppress the alarm & preparing the interactive graphs which will display when the graphs are significant like node down, services down ,pending message count these are seen by ,email configuration by creating job alert. Scheduling to refresh on every 15 minutes.





2-We have subscribed the

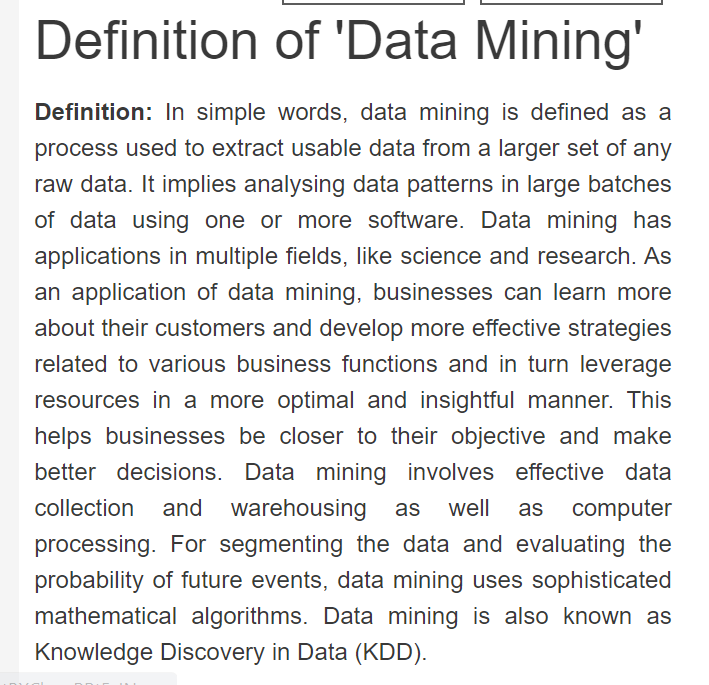


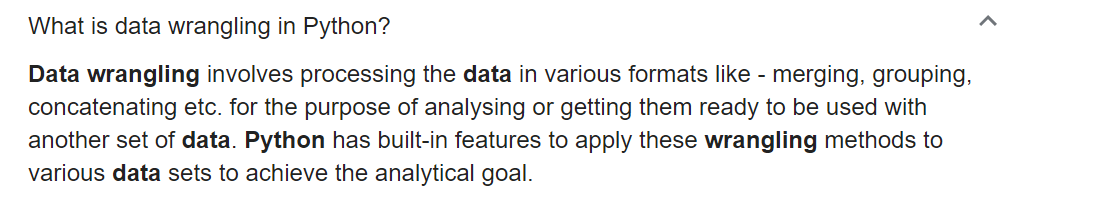


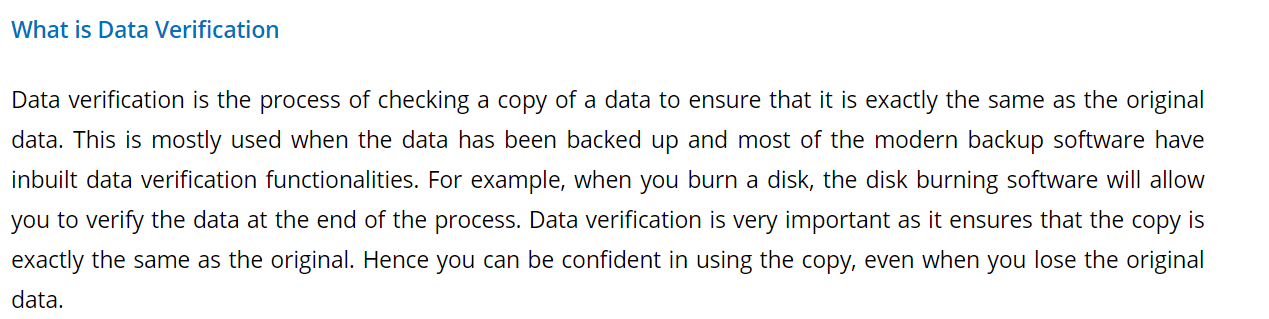
| **date\_part** | **Values** |
| --- | --- |
| 'year' | Four-digit year |
| 'quarter' | 1-4 |
| 'month' | 1-12 or “January”, “February”, and so on |
| 'dayofyear' | Day of the year; Jan 1 is 1, Feb 1 is 32, and so on |
| 'day' | 1-31 |
| 'weekday' | 1-7 or “Sunday”, “Monday”, and so on |
| 'week' | 1-52 |
| 'hour' | 0-23 |
| 'minute' | 0-59 |
| 'second' | 0-60 |

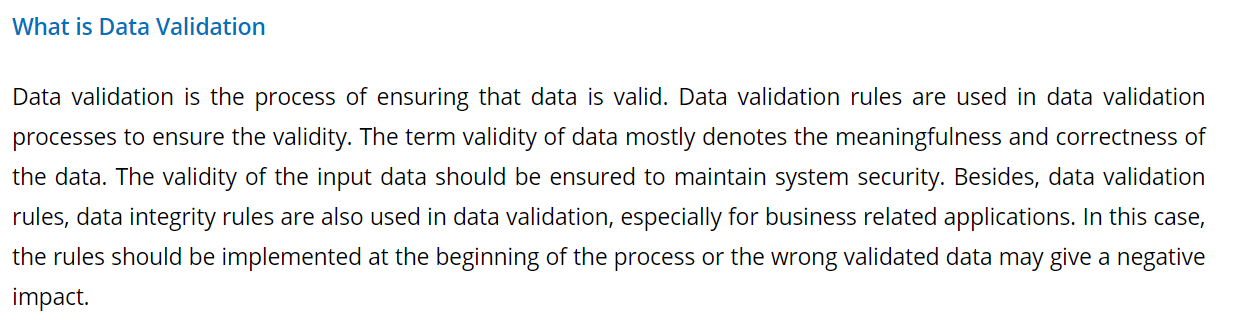
DATA MANAGEMENT:

DAY -1  
[#SQL](https://www.linkedin.com/feed/hashtag/?keywords=sql&highlightedUpdateUrns=urn%3Ali%3Aactivity%3A6802469687721635840)  
Difference between Where and Having Clause?  
It is one of the most popular SQL interview questions.  
  
WHERE:- Where is used to filter the records in a table. It can be used with SELECT,INSERT and UPDATE clause.Aggregate functions like Sum,Min,Max,Count,Avg cannot be used with where clause if you do so, you will get an SYNTAX error.Where filter the rows before grouping clause.  
  
HAVING:- Having clause is used to filter group records. It can only be used with the SELECT statement. Aggregate functions can be used with Having clause. Having filter the groups after the aggregation are performed.

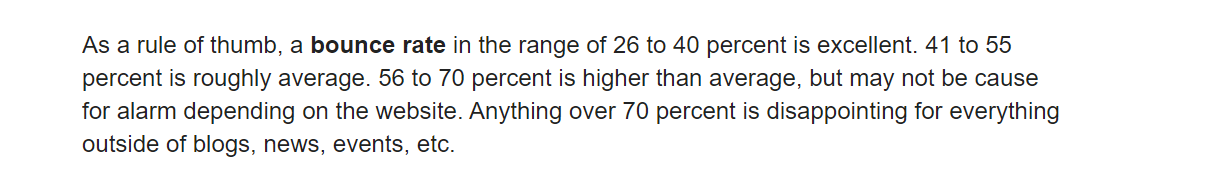


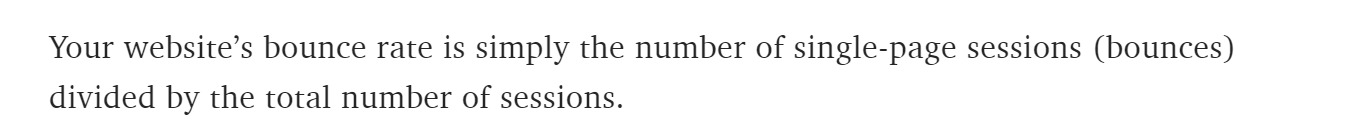






GA





------SQL----------

Deleting Rows

**DELETE FROM EMP E1**

**WHERE ROWID > (SELECT MIN(ROWID)FROM EMP E2**

**WHERE E1.DEPTNO = E2.DEPTNO);**

**delete from t**

**where rowid IN ( select rid**

**from (select rowid rid,**

**row\_number() over (partition by**

**companyid, agentid, class , status, terminationdate**

**order by rowid) rn**

**from t)**

**where rn <> 1);**

**delete from emp e1**

**where rowid not in**

**(select max(rowid) from emp e2**

**where e1.empno = e2.empno );**

Using rowid-

**delete from emp**

**where rowid not in**

**(select max(rowid) from emp group by empno);**

Using self join-

**DELETE FROM tablename a**

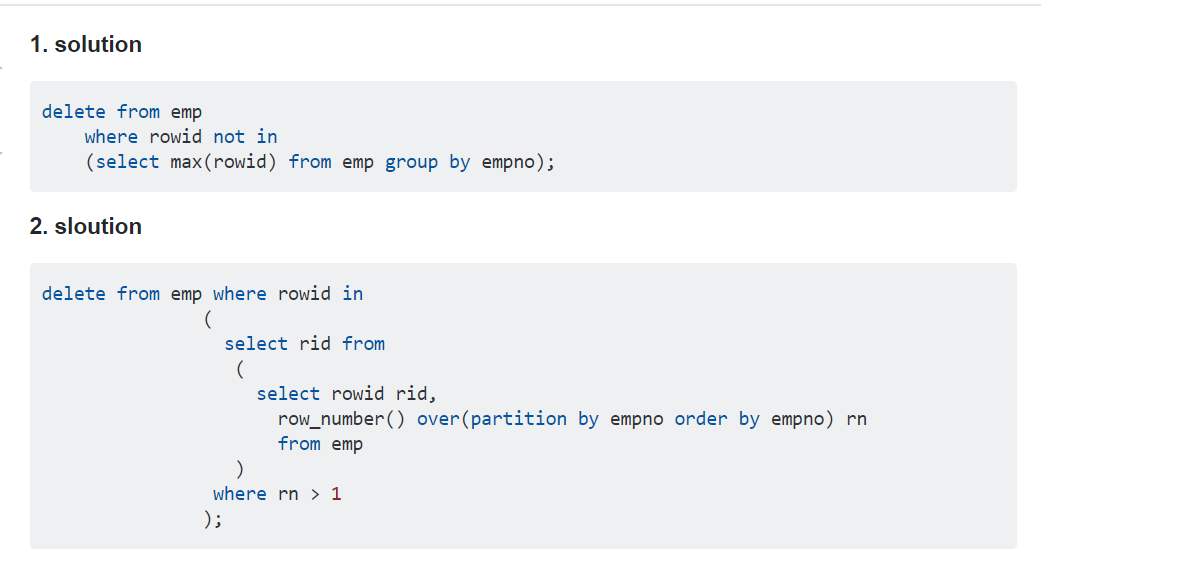
**WHERE a.ROWID > ANY (SELECT b.ROWID**

**FROM tablename b**

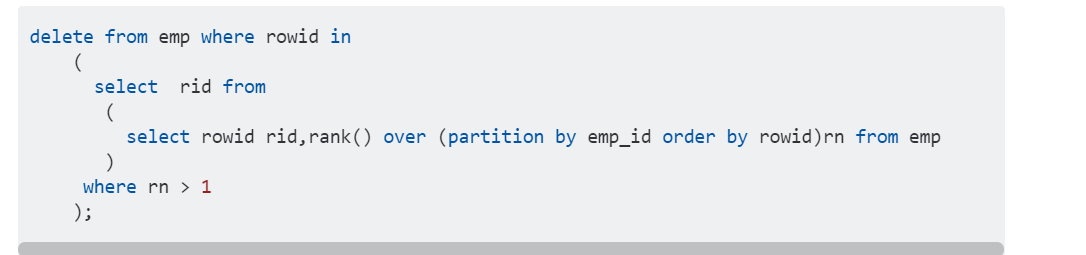
**WHERE a.fieldname = b.fieldname**

**AND a.fieldname2 = b.fieldname2**

**AND ....so on.. to identify the duplicate rows....)**



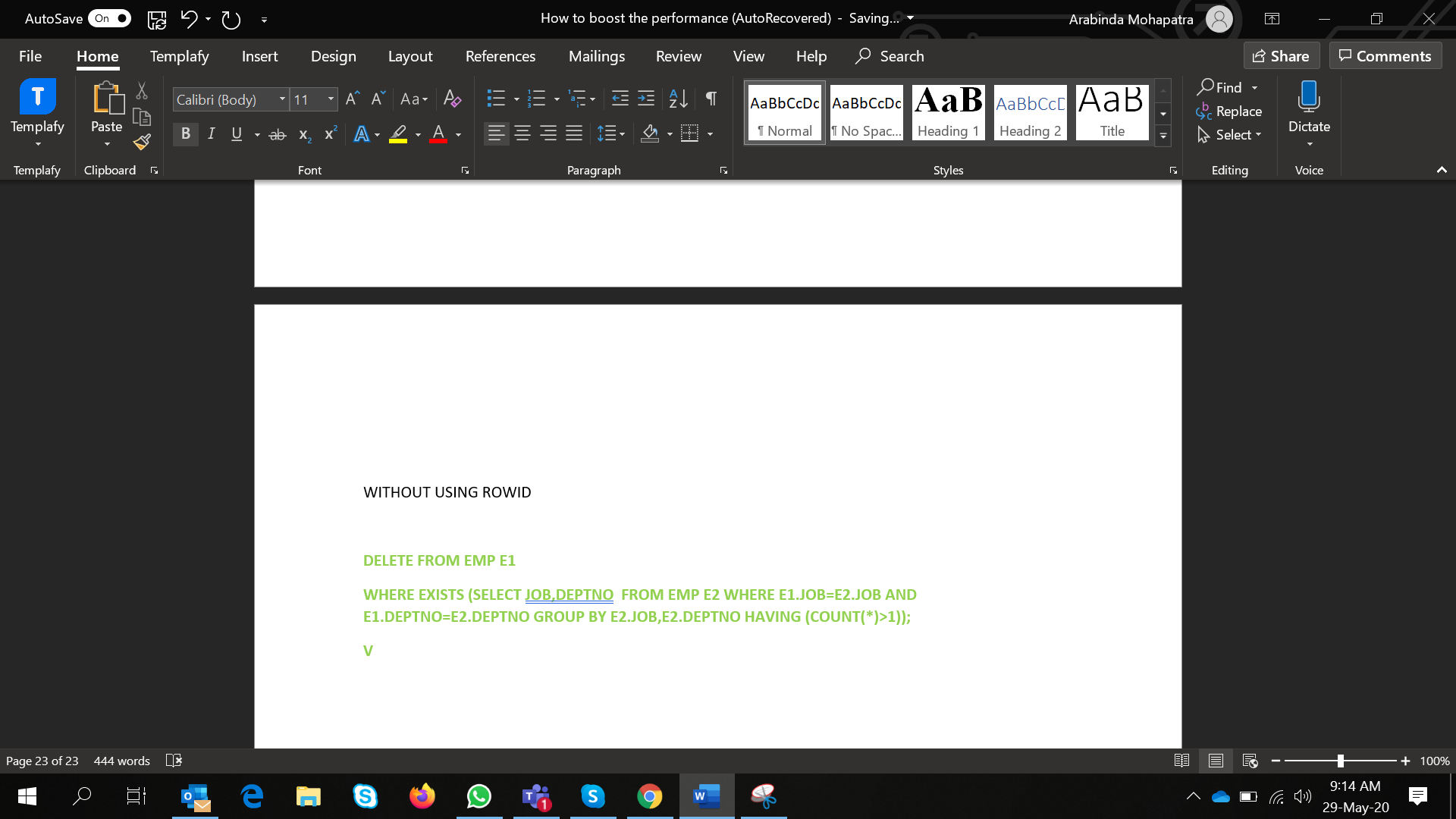




WITHOUT USING ROWID

**DELETE FROM EMP E1**

**WHERE EXISTS (SELECT JOB,DEPTNO FROM EMP E2 WHERE E1.JOB=E2.JOB AND E1.DEPTNO=E2.DEPTNO GROUP BY E2.JOB,E2.DEPTNO HAVING (COUNT(\*)>1));**



SELECT CASE

WHEN "I4%SAL" > 100 THEN 'SALARYGOOD'

ELSE 'BAD'

END REVEIEW FROM

(

SELECT DEPTNO,SAL,SAL\*.04 AS "I4%SAL"

FROM

EMP GROUP BY DEPTNO,SAL

);

Tips for SQL#

SQL Day -2

**#SQLTips**#Tableau#

1-Use Boolean function during the aggeration /calculation/comparing to avoid performance issue. & Avoid using case when statement in case data is very larger(1.5 cr records).

2-Select ONLY those columns in a query that are required. Extra columns which are not actually used, incur more I/O on the database and increase network traffic.

3-DISTINCT incurs an extra sort operation and therefore slows your queries down.

4-should not be any Cartesian product in the query unless there is a definite requirement (LOD Calculation In tableau ).

4-Use Union all instead of Union.UNION incurs an extra sort operation which can be avoided.

6-Avoid column alias name during the joining.

7-Use of the EXISTS operator where possible once a match has been found it can stop and avoid a full table scan(instead of IN)

8-Use equi-joins(comapring with = operator) whenever possible, they improve SQL efficiency.

9- Separate SQL statements for different tasks avoid complex calculation in one Select statement.(Achive it by Union all)

10-Joins to complex views are not recommended ,impact the executing quey time.

## **What is the difference between Equi Join and Inner Join in SQL?**

An equijoin is a join with a join condition containing an equality operator. An equijoin returns only the rows that have equivalent values for the specified columns.

An inner join is a join of two or more tables that returns only those rows (compared using a comparison operator) that satisfy the join condition.

**Pictorial representation : EQUI JOIN Vs. INNER JOIN**

Day -3:

**#SQL**#Tableau#PerformanceTips#

11- Avoid doing an ORDER BY & group by on a large data set especially if the response time is important.

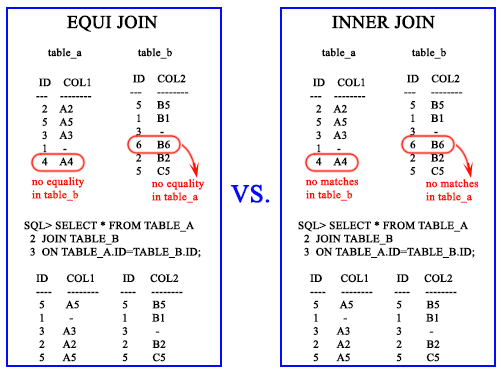
12- While querying on a partitioned table try to use the partition key in the “WHERE” clause if possible. This will ensure partition pruning

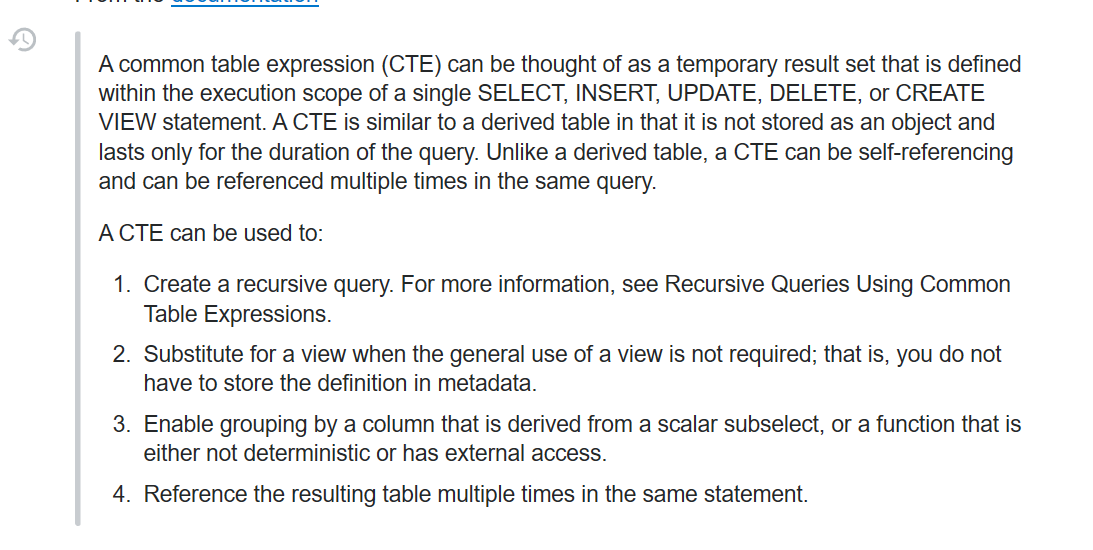
13-Avoid Having Clause. Having a clause is required if you further wish to filter the result of aggregations..

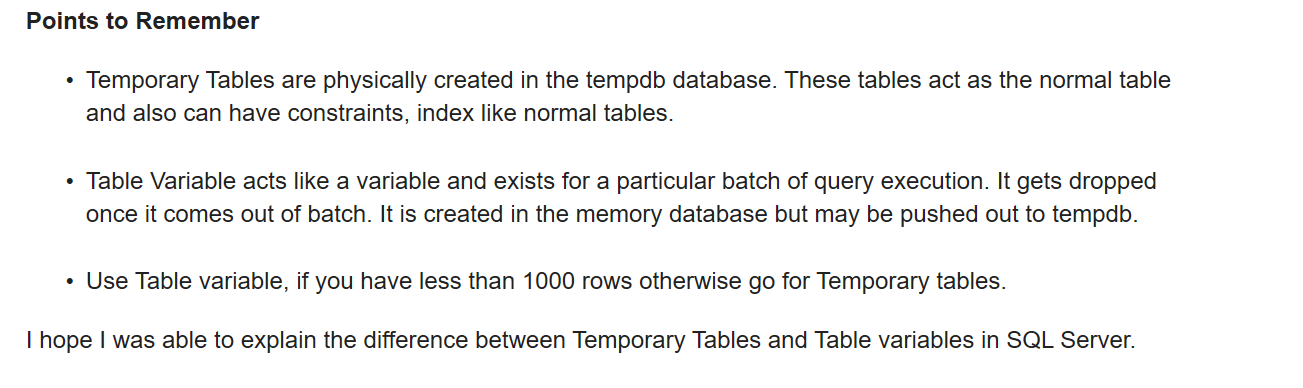
14-Drop unused Indexes.

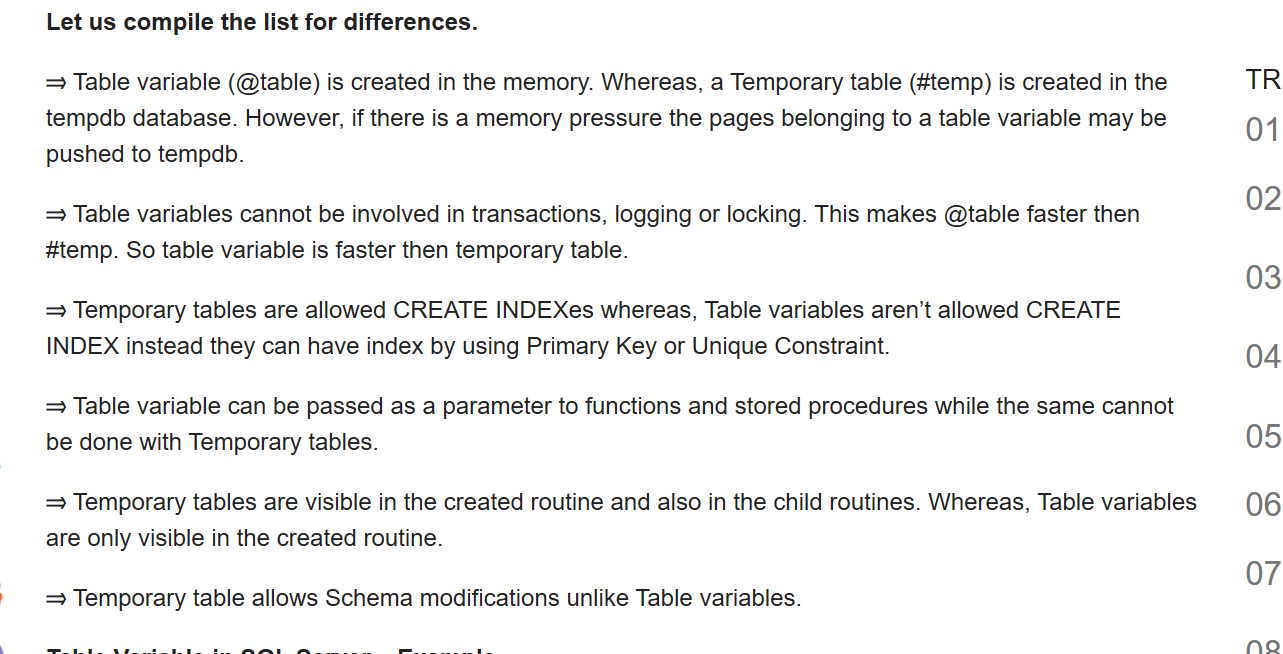
15- Use joins instead of sub-queries.

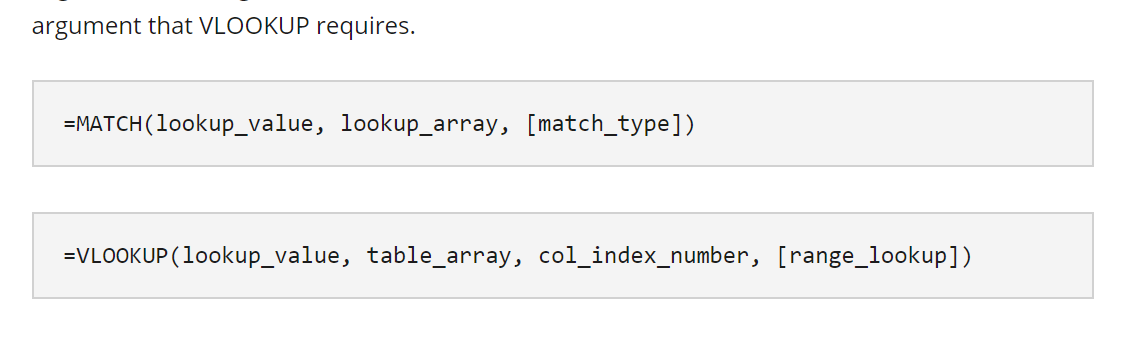
# **#tableau** **#data** **#tipsandtricks**#SQLperformance#

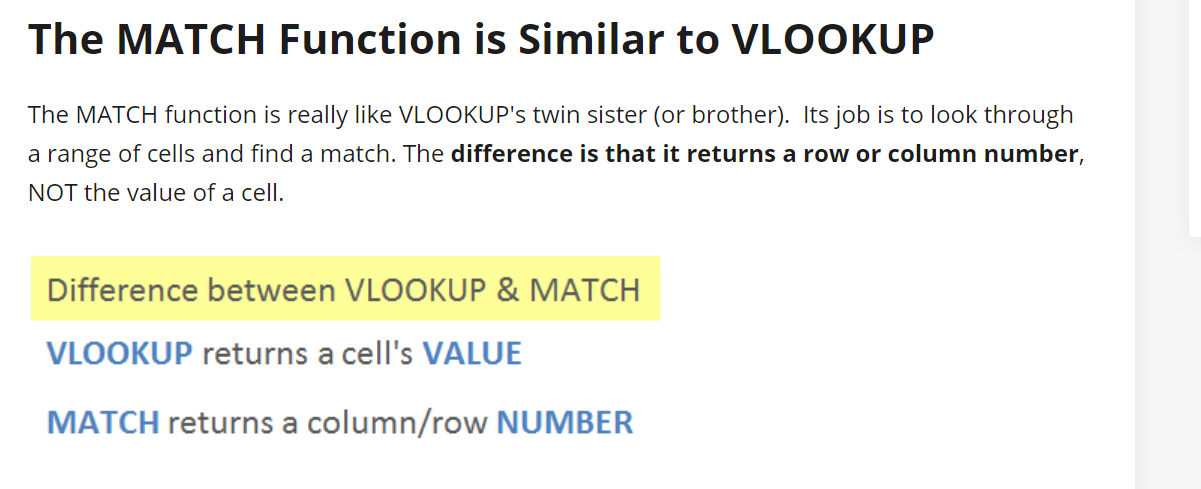


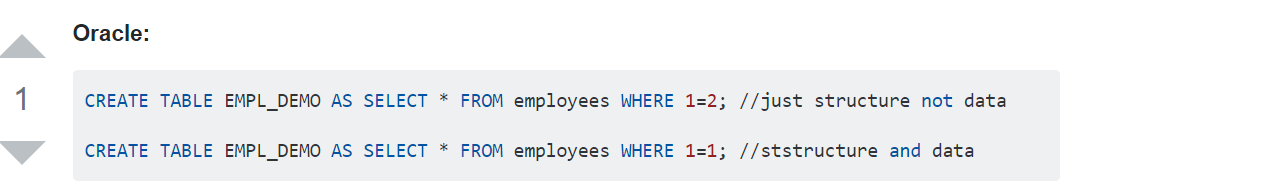




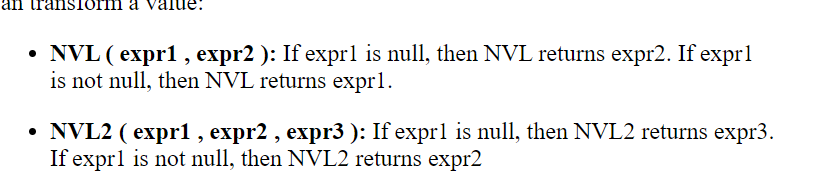


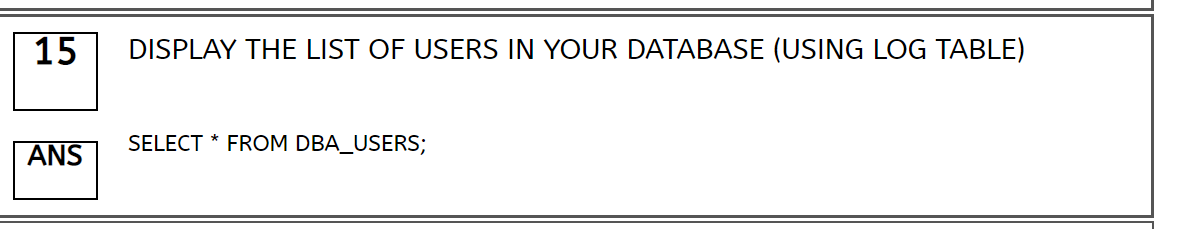












Any is used when 1 or more records match the sub condition and the result is true for those records.

All is used when all the records match the sub condition thus we get the output to be true else false.

Exists is same as any except for the time consumed will be less as, in ANY the query goes on executing where ever the condition is met and gives results . In case of exists it first has to check throughout the table for all the records that match and then execute it.

***Exists* Operator :**

* A subquery with exists does not really return any data; it returns TRUE or FALSE.
* This operator performance is very high compared to the “IN” operator.
* Always exist operator start searching mechanism when child query return true.
* It cannot compare direct values . It should have a sub query with Select.
* SQL Engine stops process as soon as finding a single condition in EXISTS.

***Explanation:***

If inner query returns atleast one row then EXISTS operator returns true and where condition will be true and the row is selected if inner query returns no row then EXISTS returns false where condition will be false and the row is not selected