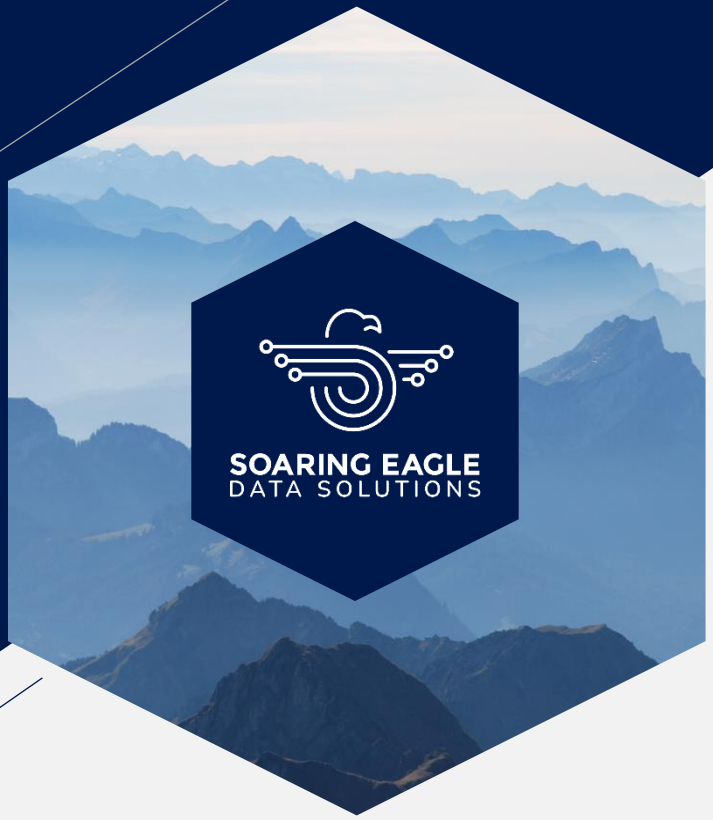
Advanced Index Maintenance for Performance

Welcome!

To our presentation

SOARING EAGLE
DATA SOLUTIONS

Because Performance Matters



Bespoke Data Solutions

Soaring Eagle Data Solutions has been providing training, performance and tuning, and managed services for business of all sizes since 1997.

Because Performance Matters

Abstract

- At this point, you've figured out that indexes, a critical component of good performance, need to be maintained, and are likely using the built-in index maintenance scripts, Ola Hallengren scripts, or something home-grown. But is this enough, or is this a starting point?

SOARING EAGLE
DATA SOLUTIONS

Outline

- Where to find information on what indexes are for, how they are used, and basic maintenance
- Identifying and safely removing redundant indexes
- Identifying and adding missing indexes
- Identifying and removing unused indexes

SOARING EAGLE
DATA SOLUTIONS

Because Performance Matters

Basics

- While in general, indexes are self-maintained and self-balancing, over time in a busy system they can become imbalanced
- Enhancements added decades ago to support performance while using row-level locking also make us clean up things like row-forwarding and deleted rows
- In addition, because we have a cost-based optimizer, and the statistics histogram is not always kept up to date, you must make sure those statistics are kept up to date, especially if you're not doing the clean-up described above by performing standard index maintenance.
- THE ABOVE IS NOT COVERED in this seminar.
- For information about how indexes are used and how to maintain them, look at presentations on our YouTube channel, or ask to attend our Advanced SQL for Programmers course.

http://www.youtube.com/user/soaringeagledba

SOARING EAGLE
DATA SOLUTIONS

Redundant Indexes

- My current record for removing redundant indexes brought a 4T database down to 300G.
- Why are redundant indexes bad?
  - Storage
    - Not only are you storing all of the additional indexing information, but you are also backing it up and need the extra time to restore it, and transmitting all of the extra information whenever you move the backup off site (you're doing this, right?)
  - Maintenance
    - All of the additional indexes need standard maintenance performed, which consumes CPU cycles, heavy disk i/o and valuable maintenance window time
  - Overhead at insert / delete time (and update, if the row moves or index changes)

SOARING EAGLE
DATA SOLUTIONS

Because Performance Matters

Redundant indexes continued

- What makes an index redundant?
  - If you are old enough to remember, consider a white-pages telephone book:
    - It's an index on last name, first name, and middle name, and brings you back address and telephone number.
    - If you have this telephone book, do you also need a separate index to do a lookup on last name?
  - So, if you have
    - an index on (a,b)
    - An index on (a)
  - …You can safely eliminate the index on (a) because the index on (a,b) will work just fine for the same lookups.
  - NOTE: make sure all of the includes from (a), if any, are added to the includes for (a,b) before dropping the index on (a)
  - For additional examples, go to https://mssqlperformance.blogspot.com/ and look at entry "When is an index redundant?" from Wednesday, February 19 2020

SOARING EAGLE
DATA SOLUTIONS

Redundant indexes continued

- How can we do this on a large scale? You may have hundreds (or thousands!) of indexes in each database.
- This is going to be a manual process. If one of you writes code to do it all and are willing to share, please let us know.
- In the meantime, there's code available to list all indexes in a database sorted by table name and column name. It's on the next page, but for reference it's also available in my blog, under the heading of "List indexes with keys by table"

SOARING EAGLE
DATA SOLUTIONS

Because Performance Matters

```
SELECT
  schema_name(schema_id) as SchemaName, OBJECT_NAME(si.object_id) as
TableName, si.name as IndexName,
  (CASE is_primary_key WHEN 1 THEN 'PK' ELSE '' END) as PK,
  (CASE is_unique WHEN 1 THEN '1' ELSE '0' END)+' '+
  (CASE si.type WHEN 1 THEN 'C' WHEN 3 THEN 'X' ELSE 'B' END)+' '+  -- B=basic,
C=Clustered, X=XML
  (CASE INDEXKEY_PROPERTY(si.object_id,index_id,1,'IsDescending') WHEN 0 THEN 'A'
WHEN 1 THEN 'D' ELSE '' END)+
  (CASE INDEXKEY_PROPERTY(si.object_id,index_id,2,'IsDescending') WHEN 0 THEN 'A'
WHEN 1 THEN 'D' ELSE '' END)+
  (CASE INDEXKEY_PROPERTY(si.object_id,index_id,3,'IsDescending') WHEN 0 THEN 'A'
WHEN 1 THEN 'D' ELSE '' END)+
  (CASE INDEXKEY_PROPERTY(si.object_id,index_id,4,'IsDescending') WHEN 0 THEN 'A'
WHEN 1 THEN 'D' ELSE '' END)+
  (CASE INDEXKEY_PROPERTY(si.object_id,index_id,5,'IsDescending') WHEN
```

```
0 THEN 'A' WHEN 1 THEN 'D' ELSE '' END)+
  (CASE INDEXKEY_PROPERTY(si.object_id,index_id,6,'IsDescending') WHEN 0 THEN 'A'
WHEN 1 THEN 'D' ELSE '' END)+
  '' as 'Type',
```

SOARING EAGLE
DATA SOLUTIONS

Because Performance Matters

```
  INDEX_COL(schema_name(schema_id)+'.'+OBJECT_NAME(si.object_id),index_id,1)
as Key1,
  INDEX_COL(schema_name(schema_id)+'.'+OBJECT_NAME(si.object_id),index_id,2)
as Key2,
  INDEX_COL(schema_name(schema_id)+'.'+OBJECT_NAME(si.object_id),index_id,3)
as Key3,
  INDEX_COL(schema_name(schema_id)+'.'+OBJECT_NAME(si.object_id),index_id,4)
as Key4,
  INDEX_COL(schema_name(schema_id)+'.'+OBJECT_NAME(si.object_id),index_id,5)
as Key5,
  INDEX_COL(schema_name(schema_id)+'.'+OBJECT_NAME(si.object_id),index_id,6)
as Key6
FROM sys.indexes as si
LEFT JOIN sys.objects as so on so.object_id=si.object_id
WHERE index_id>0 -- omit the default heap
  and OBJECTPROPERTY(si.object_id,'IsMsShipped')=0 -- omit system tables
  and not (schema_name(schema_id)='dbo' and
OBJECT_NAME(si.object_id)='sysdiagrams') -- omit sysdiagrams
ORDER BY SchemaName ,TableName , Key1, Key2, Key3, Key4, Key5, Key6
```

SOARING EAGLE
DATA SOLUTIONS

Because Performance Matters

Sample output

Results | Messages

	SchemaName	TableName	IndexName	PK	Type	Key1	Key2	Key3	Key4	Key5	Key6
1	Client	BlockTimePurchase	PK		1 B A	BlockTimePurchaseID	NULL	NULL	NULL	NULL	NULL
2	Client	BlockTimePurchase	idx_ClientID		0 C AA	ClientID	PurchaseDate	NULL	NULL	NULL	NULL
3	Client	Clients	PK__Clients__E67E1A0516445 2B1	PK	1 C A	ClientID	NULL	NULL	NULL	NULL	NULL
4	Client	Clients	idx		0 B AAAAAA	PrimaryConsultantID	isActive	ClientStatusID	ClientTypeID	Name	ClientID
5	Client	Contacts	idx_clientid		0 B AD	ClientID	ContactID	NULL	NULL	NULL	NULL
6	Client	Contacts	PK__Contacts__5C6625BB1A14E395	PK	1 C A	ContactID	NULL	NULL	NULL	NULL	NULL
7	Client	DistributionTypes	PK_DistributionTypes	PK	1 B A	DistributionTypeID	NULL	NULL	NULL	NULL	NULL
8	Client	Domains	PK_Domains	PK	1 B A	DomainID	NULL	NULL	NULL	NULL	NULL
9	Client	Domains	ci		1 C AA	EmailPattern	ClientID	NULL	NULL	NULL	NULL
10	Client	Resellers	PK__Reseller__3357E85E679450C0	PK	1 B A	ResellerID	NULL	NULL	NULL	NULL	NULL
11	dbo	aspnet_Applications	PK__aspnet_A__C93A4C982DB1C7EE	PK	1 B A	ApplicationId	NULL	NULL	NULL	NULL	NULL
12	dbo	aspnet_Applications	UQ__aspnet_A__3091033158671BC9		1 B A	ApplicationName	NULL	NULL	NULL	NULL	NULL
13	dbo	aspnet_Applications	UQ__aspnet_A__30910331336AA144		1 B A	ApplicationName	NULL	NULL	NULL	NULL	NULL
14	dbo	aspnet_Applications	UQ__aspnet_A__17477DE4558AAF1E		1 B A	LoweredApplicationName	NULL	NULL	NULL	NULL	NULL
15	dbo	aspnet_Applications	aspnet_Applications_Index		0 C A	LoweredApplicationName	NULL	NULL	NULL	NULL	NULL
16	dbo	aspnet_Applications	UQ__aspnet_A__17477DE4308E3499		1 B A	LoweredApplicationName	NULL	NULL	NULL	NULL	NULL
17	dbo	aspnet_Membership	aspnet_Membership_index		0 C AA	ApplicationId	LoweredEmail	NULL	NULL	NULL	NULL
18	dbo	aspnet_Membership	PK__aspnet_M__1788CC4D36470DEF	PK	1 B A	UserId	NULL	NULL	NULL	NULL	NULL
19	dbo	aspnet_Paths	aspnet_Paths_index		1 C AA	ApplicationId	LoweredPath	NULL	NULL	NULL	NULL

Missing Indexes

- The SQL Server Database Engine Tuning Advisor identifies queries that could potentially benefit from adding an index.
- DO NOT arbitrarily add all the indexes recommended by the advisor. You'll end up with way too many indexes because the advisor might recommend 10 indexes that are very similar (or even completely redundant) but could be combined into a single index.

SOARING EAGLE
DATA SOLUTIONS

Why missing indexes?

- Adding missing indexes can have a dramatic positive effect on performance.
- Adding missing indexes can reduce io, reduce CPU, reduce memory thrashing, reduce elapsed time, all by a very significant amount.
- At a recent engagement we ran the following script, and after removing redundancy and adding the indexes, we brought system load down from 80% of CPU to 15%. An especially painful query went from 20 minutes to sub-second.
- Code follows but is also available on my blog mssqlperformance.blogspot.com.

SOARING EAGLE
DATA SOLUTIONS

```
DECLARE @Edition varchar(50)
SET @Edition = CONVERT(varchar(50), SERVERPROPERTY('Edition'))
DECLARE @bEnterpriseEdition bit
IF (LEFT(@Edition, 9) = 'Enterpris' OR LEFT(@Edition, 9) = 'Developer')
SET @bEnterpriseEdition = 1

SELECT
    'CREATE INDEX IX_'
        + REPLACE(obj.Name, ' ', '') +'_'
        + replace(replace(replace (equality_columns, '[', ''),']',''),', ','_')
        + ' ON '
        + sch.name COLLATE SQL_Latin1_General_CP1_CI_AS + '.' + QuoteName(obj.name)
        + ' ('
        + equality_columns
        + CASE WHEN inequality_columns IS NULL
            THEN ''
            ELSE ',' +  inequality_columns
          END
```

```
+ ')'
        + CASE WHEN included_columns IS NOT NULL
            THEN ' INCLUDE (' + ISNULL(included_columns,'') + ') '
            ELSE ''
          END
        + CASE WHEN @bEnterpriseEdition = 1
            THEN ' WITH (online = ON)'
            ELSE ''
          END
        + ' -- ' + CONVERT (varchar, avg_user_impact) + '% anticipated impact'
```

SOARING EAGLE
DATA SOLUTIONS

Because Performance Matters

```
FROM sys.dm_db_missing_index_details mid
    JOIN sys.dm_db_missing_index_groups mig
      ON mid.index_handle = mig.index_handle
    JOIN sys.dm_db_missing_index_group_stats migs
      ON migs.group_handle = mig.index_group_handle
    JOIN sys.objects obj
      ON obj.object_id = mid.object_id
    JOIN sys.schemas sch
      ON obj.schema_id = sch.schema_id
  WHERE
    database_id = db_id()
    AND avg_user_impact > 60
    AND equality_columns IS NOT NULL
  ORDER BY
    obj.name
    , equality_columns
    , inequality_columns
    , included_columns
```

SOARING EAGLE
DATA SOLUTIONS

Because Performance Matters

Sample output

- CREATE INDEX IX_ClientDatabases_ClientServerID_isActive ON Oversight.[ClientDatabases] ([ClientServerID], [isActive]) INCLUDE ([DatabaseName], [AlwaysOn]) -- 63.92% anticipated impact
- CREATE INDEX IX_ClientDatabases_isActive ON Oversight.[ClientDatabases] ([isActive],[DatabaseName], [AlwaysOn]) INCLUDE ([ClientServerID]) -- 63.49% anticipated impact
- CREATE INDEX IX_ClientServers_ClientID ON Oversight.[ClientServers] ([ClientID],[EntryDate]) -- 92.97% anticipated impact
- CREATE INDEX IX_ClientServers_ClientID_EnvironmentID ON Oversight.[ClientServers] ([ClientID], [EnvironmentID]) -- 86.73% anticipated impact
- CREATE INDEX IX_DriveSpaceConfig_ClientServerID_DriveLetter ON Oversight.[DriveSpaceConfig] ([ClientServerID], [DriveLetter]) -- 75.19% anticipated impact
- CREATE INDEX IX_Emails_EventID ON Oversight.[Emails] ([EventID],[EmailID], [Subject]) -- 68.96% anticipated impact
- CREATE INDEX IX_Emails_Subject ON Oversight.[Emails] ([Subject],[EmailSent]) -- 99.91% anticipated impact

SOARING EAGLE
DATA SOLUTIONS

Because Performance Matters

Sample output 2

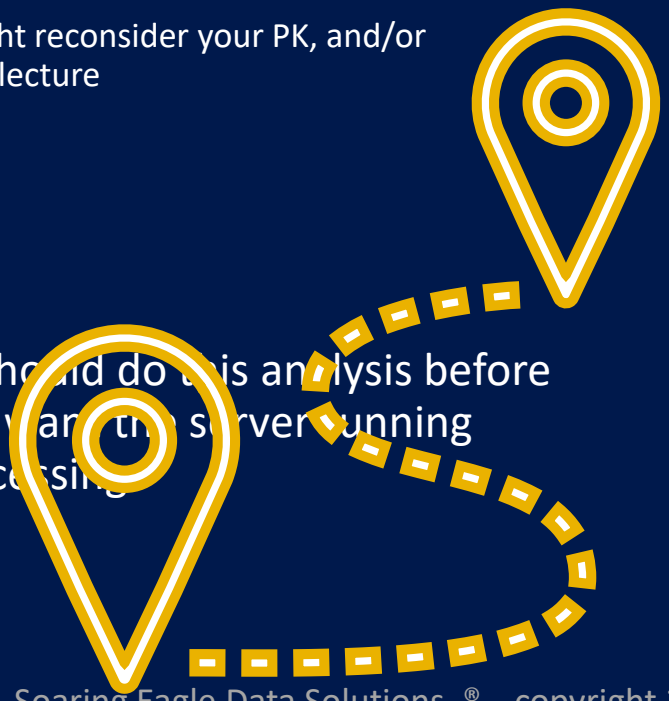
QLQuery2.sql - (l...INGEAGLE\jeff (55))* SQLQuery1.sql - (l...NGEAGLE\jeff (384))*

```
CREATE INDEX IX_ClientDatabases_ClientServerID_isActive ON Oversight.[ClientDatabases] ([ClientServerID], [isActive]) INCLUDE ([DatabaseName], [Alw
CREATE INDEX IX_ClientDatabases_isActive ON Oversight.[ClientDatabases] ([isActive],[DatabaseName], [AlwaysOn]) INCLUDE ([ClientServerID])  -- 63.4
CREATE INDEX IX_ClientServers_ClientID ON Oversight.[ClientServers] ([ClientID],[EntryDate]) -- 92.97% anticipated impact
CREATE INDEX IX_ClientServers_ClientID_EnvironmentID ON Oversight.[ClientServers] ([ClientID], [EnvironmentID]) -- 86.73% anticipated impact
CREATE INDEX IX_DriveSpaceConfig_ClientServerID_DriveLetter ON Oversight.[DriveSpaceConfig] ([ClientServerID], [DriveLetter]) -- 75.19% anticipated
CREATE INDEX IX_Emails_EventID ON Oversight.[Emails] ([EventID],[EmailID], [Subject]) -- 68.96% anticipated impact
CREATE INDEX IX_Emails_Subject ON Oversight.[Emails] ([Subject],[EmailSent]) -- 99.91% anticipated impact
CREATE INDEX IX_Events_ClientID ON flight.[Events] ([ClientID],[EventDate]) INCLUDE ([EventName], [EventTypeID], [EventPriority], [AssignedTo])  --
CREATE INDEX IX_Events_ClientID ON flight.[Events] ([ClientID],[EventDate]) INCLUDE ([EventTypeID], [EventPriority], [AssignedTo])  -- 75.99% antic
CREATE INDEX IX_Events_ClientID_EventPriority ON flight.[Events] ([ClientID], [EventPriority],[EventDate]) INCLUDE ([EventTypeID], [AssignedTo])  -
CREATE INDEX IX_Events_ClientID_EventStatusID ON flight.[Events] ([ClientID], [EventStatusID],[EventDate]) INCLUDE ([EventName], [EventTypeID], [Ev
CREATE INDEX IX_Events_ClientID_EventStatusID ON flight.[Events] ([ClientID], [EventStatusID],[EventDate]) INCLUDE ([EventTypeID], [EventPriority],
CREATE INDEX IX_Events_EventPriority ON flight.[Events] ([EventPriority],[ClientID], [EventTypeID], [EventDate]) INCLUDE ([AssignedTo])  -- 73.35%
CREATE INDEX IX_RunbookImages_RunbookSectionID ON flight.[RunbookImages] ([RunbookSectionID]) -- 70.22% anticipated impact
CREATE INDEX IX_RunbookValidationCodes_FlightUserID_ClientID_ValidationCode_Used ON flight.[RunbookValidationCodes] ([FlightUserID], [ClientID], [V
CREATE INDEX IX_Se_FailedJobsTaskDrafts_EventID ON Oversight.[Se_FailedJobsTaskDrafts] ([EventID]) -- 95.56% anticipated impact
CREATE INDEX IX_Se_FailedJobsTaskDrafts_Se_FailedJobsTaskDraftID ON Oversight.[Se_FailedJobsTaskDrafts] ([Se_FailedJobsTaskDraftID]) -- 82.91% anti
```


Because Performance Matters

Unused Indexes

- Unused indexes suffer all the disadvantages of redundant indexes, plus… they're not used.
- Note that some indexes should not be removed, even if they are unused
  - Indexes on system tables
  - Primary key indexes
    - Note: if your PK index is never being used, you might reconsider your PK, and/or reconsider the clustering index, but that's another lecture
- Code follows, but is also available in my blog
  - Mssqlperformance.blogspot.com
- WARNING: statistics on used indexes only live as long as the server is up! This means you should do this analysis before bouncing the server, not after; optimally you want the server running through week-, month-, and quarter-end processing



SOARING EAGLE DATA SOLUTIONS

Because Performance Matters

```
DECLARE @cmd VARCHAR(6000)

DECLARE @TempIndexDrop TABLE (
    [Database_Name] [nvarchar](128) NULL,
    [Object_Name] [nvarchar](128) NULL,
    [Index_Name] [sysname] NULL,
    [Drop_Statement] [nvarchar](283) NULL,
    [user_seeks] [bigint]  NULL,
    [user_scans] [bigint]  NULL,
    [user_lookups] [bigint]  NULL,
    [SpaceSavedMeg] decimal (12,2)  null
)
```

SOARING EAGLE
DATA SOLUTIONS

```
SET @cmd = 'USE [?] ;
select db_name(database_id) as ''Database_Name'', object_name(i.object_id) as ''Object_Name'',
i.name as ''Index_Name''
, ''USE ['' + db_name(database_id) + '']; drop index ['' + i.name + ''] on ['' + sch.name + ''].['' + obj.name + '']'' as ''Drop_Statement'',
user_seeks, user_scans, user_lookups,
(select sum(s.used_page_count)*8./1024. from sys.dm_db_partition_stats s where i.object_id = s.object_id and i.index_id = s.index_id)
from sys.dm_db_index_usage_stats ius join sys.indexes i
on ius.index_id = i.index_id and ius.object_id = i.object_id join sys.objects obj
on ius.object_id = obj.object_id join sys.schemas sch
on obj.schema_id = sch.schema_id
where
database_id = db_id(''?'') and
user_seeks + user_scans + user_lookups = 0
and i.name not like ''pk%''
and ''?'' not in (''master'', ''model'',''tempdb'',''msdb‘’)
order by object_name(i.object_id), i.name'
```

```
INSERT INTO @TempIndexDrop
(
 [Database_Name]
 ,[Object_Name]
 ,[Index_Name]
 ,[Drop_Statement]
 ,[user_seeks]
 ,[user_scans]
 ,[user_lookups]
 ,[SpaceSavedMeg]
)
Exec sp_msforeachdb @cmd

SELECT SUM(SpaceSavedMeg) as 'Potential Storage Savings' FROM  @TempIndexDrop

SELECT *
FROM @TempIndexDrop
ORDER BY [Database_Name], [Object_Name]
```

Sample output

Results | Messages

	Potential Storage Savings
1	4323.67

	Database_Name	Object_Name	Index_Name	Drop_Statement	user_seeks	user_scans	user_lookups	SpaceSavedMeg
1	SoaringEagle	Emails	IX_Emails_EventID	USE [SoaringEagle]; drop index [IX_Emails_EventI...	0	0	0	1497.27
2	SoaringEagle	Emails	IX_Emails_EventID_subject	USE [SoaringEagle]; drop index [IX_Emails_EventI...	0	0	0	749.79
3	SoaringEagle	Events	IX_Events_ClientID_EventPriority	USE [SoaringEagle]; drop index [IX_Events_ClientI...	0	0	0	248.18
4	SoaringEagle	Events	IX_Events_ClientID_EventStatusID	USE [SoaringEagle]; drop index [IX_Events_ClientI...	0	0	0	657.84
5	SoaringEagle	Events	IX_Events_ClientID2	USE [SoaringEagle]; drop index [IX_Events_ClientI...	0	0	0	657.82
6	SoaringEagle	Events	IX_Events_EventPriority	USE [SoaringEagle]; drop index [IX_Events_Event...	0	0	0	248.65
7	SoaringEagle	SQL_DriveSpaceStats	IX_ClientServerID	USE [SoaringEagle]; drop index [IX_ClientServerID]...	0	0	0	264.12

Summary

- Standard index maintenance is a starting point, not an ending point
- You need to analyze which indexes are missing, redundant, and/or unused

SOARING EAGLE
DATA SOLUTIONS

Because Performance Matters

Questions?

SOARING EAGLE
DATA SOLUTIONS

Because Performance Matters

Thank You.

Jeff Garbus

813 966 1180

Jeff.garbus@sedatasolutions.io

www.sedatasolutions.io