```sql
--Create a database to use for this example
IF (SELECT DB_ID('HoggysBlog')) IS NULL
CREATE DATABASE HoggysBlog
GO
USE HoggysBlog
GO


--Create a table of consecutive numbers
IF (SELECT OBJECT_ID('Number')) IS NULL
	BEGIN
		CREATE TABLE Number
			(
				n INT
			)

		INSERT INTO Number (n)
			SELECT ROW_NUMBER() OVER (ORDER BY CURRENT_TIMESTAMP) rn
			FROM sys.trace_event_bindings r1, sys.trace_event_bindings
	END


--Create the table used for this example
IF (SELECT OBJECT_ID('Animal')) IS NULL
		BEGIN
			CREATE TABLE Animal
				(
					ID INT IDENTITY(1,1),
					Species VARCHAR(50)
				)
		END


--Put 4,999,990 Dogs in the table
INSERT INTO Animal (Species)
	SELECT 'Dog' FROM Number
	WHERE n <=4999990


--Put 10 Cats in the table
INSERT INTO Animal (Species)
	SELECT 'Cat' FROM Number
	WHERE n <=10


--Confirm the contents of the Animals table
SELECT Species, COUNT(Species) AS Qty
FROM Animal
GROUP BY Species


--Turn on Actual execution plan here (right click in the query pane and choose "Include
Actual Execution Plan"


--Show ad-hoc dog query uses a table scan
SELECT TOP 10000 num.n
FROM Animal AS ani
```

```
	INNER JOIN Number num
	ON ani.ID = num.n
WHERE ani.Species = 'Dog'


--Show ad-hoc cat query uses an table scan
SELECT TOP 10000 num.n
FROM Animal AS ani
	INNER JOIN Number num
	ON ani.ID = num.n
WHERE ani.Species = 'Cat'


--Create filtered index for dogs
CREATE INDEX DogIndex ON Animal
	(
		Species
	)
WHERE Species = 'Dog'




--Create filtered index for cats
CREATE INDEX CatIndex ON Animal
	(
		Species
	)
WHERE Species = 'Cat'


--Confirm that accurate stats exist for values
DBCC SHOW_STATISTICS ('Animal','DogIndex')
--
DBCC SHOW_STATISTICS  ('Animal', 'CatIndex')


--Show ad-hoc dog query uses a table scan
SELECT TOP 10000 num.n
FROM Animal AS ani
	INNER JOIN Number num
	ON ani.ID = num.n
WHERE ani.Species = 'Dog'


--Show ad-hoc cat query uses an index seek - Good!
SELECT TOP 10000 num.n
FROM Animal AS ani
	INNER JOIN Number num
	ON ani.ID = num.n
WHERE ani.Species = 'Cat'


--create a paramaterised proc to encapsulate the above query

CREATE PROCEDURE usp_FindAnimal
	@What VARCHAR(50)
	AS
```

```
		SELECT TOP 10000 num.n
		FROM Animal AS ani
			INNER JOIN number num
			ON ani.ID = num.n
		WHERE ani.Species = @What


--Show Actual execution plan of dog query uses table scan as before.
DECLARE @DBIdent TINYINT
SELECT @DBIdent = (SELECT db_id('HoggysBlog'))
DBCC FLUSHPROCINDB (@DBIdent)
--
EXEC usp_FindAnimal 'Dog'


--Show Actual execution plan of cat query (used a filtered index when ad-hoc, now uses a
table SCAN!!!)
DECLARE @DBIdent TINYINT
SELECT @DBIdent = (SELECT db_id('HoggysBlog'))
DBCC FLUSHPROCINDB (@DBIdent)
--
EXEC usp_FindAnimal 'Cat'


--Change proc to use with recompile
ALTER PROCEDURE usp_FindAnimal
	@What VARCHAR(50)
	WITH RECOMPILE
	AS
		SELECT TOP 10000 num.n
		FROM Animal AS ani
			INNER JOIN number num
			ON ani.ID = num.n
		WHERE ani.Species = @What


--Show Actual execution plan of cat query (still uses table SCAN!!!)
DECLARE @DBIdent TINYINT
SELECT @DBIdent = (SELECT db_id('HoggysBlog'))
DBCC FLUSHPROCINDB (@DBIdent)
--
EXEC usp_FindAnimal 'Cat'


--Change proc to use option recompile
ALTER PROCEDURE usp_FindAnimal
	@What VARCHAR(50)
	WITH RECOMPILE
	AS
		SELECT TOP 10000 num.n
		FROM Animal AS ani
			INNER JOIN number num
			ON ani.ID = num.n
		WHERE ani.Species = @What
		OPTION (RECOMPILE)


--Show Actual execution plan of dog query still uses table scan
```

```
DECLARE @DBIdent TINYINT
SELECT @DBIdent = (SELECT db_id('HoggysBlog'))
DBCC FLUSHPROCINDB (@DBIdent)
--
EXEC usp_FindAnimal 'Dog'


--Show Actual execution plan of cat query (uses index SEEK!!!)
DECLARE @DBIdent TINYINT
SELECT @DBIdent = (SELECT db_id('HoggysBlog'))
DBCC FLUSHPROCINDB (@DBIdent)
--
EXEC usp_FindAnimal 'Cat'


--Now go and have a cup of tea
```