Advanced SQL Puzzles

Scott Peters

www.advancedsqlpuzzles.com

Last updated 08/22/2021

Table of Contents

Section I

Section II

Section III

The answers provided at the end of the document were written in SQL Server 2014 T-SQL.

I would be happy to receive corrections, additions, new tricks and techniques, and other suggestions.
scottpeters1188@outlook.com.

The latest version of this document can be found at www.advancedsqlpuzzles.com

Puzzle #1

Dance Partners

You are tasked with providing a list of dance partners from the following table.

Provide an SQL statement that matches each Student ID with an individual of the opposite gender.

Note there is a mismatch in the number of students, as one female student will be left without a dance partner. Please include this individual in your list as well.

Student ID	Gender
1001	M
2002	M
3003	M
4004	M
5005	M
6006	F
7007	F
8008	F
9009	F

Hyperlink to the solution

Puzzle #2

Managers and Employees

Given the following table, write an SQL statement that determines the level of depth each employee has from the president.

Employee ID	Manager ID	Job Title	Salary
1001		President	$185,000
2002	1001	Director	$120,000
3003	1001	Office Manager	$97,000
4004	2002	Engineer	$110,000
5005	2002	Engineer	$142,000
6006	2002	Engineer	$160,000

Here is the expected output.

Employee ID	Manager ID	Job Title	Salary	Depth
1001	NULL	President	$185,000	0
2002	1001	Director	$120,000	1
3003	1001	Office Manager	$97,000	1
4004	2002	Engineer	$110,000	2
5005	2002	Engineer	$142,000	2
6006	2002	Engineer	$160,000	2

Hyperlink to the solution

Puzzle #3

Fiscal Year Pay Rates

For each standard fiscal year, a record exists for each employee that states their current pay rate for the specified year.

Can you determine all the constraints that can be applied to this table to ensure that it contains only correct information? Assume that no pay raises are given mid-year. There are quite a few of them, so think carefully!

```
CREATE TABLE #EmployeePayRecord
(
EmployeeID INTEGER
FiscalYear INTEGER,
StartDate  DATE,
EndDate    DATE,
PayRate    MONEY
);
```

Hyperlink to the solution

Puzzle #4

Two Predicates

Write an SQL statement given the following requirements.

For every customer that had a delivery to California, provide a result set of the customer orders that were delivered to Texas.

Customer ID	Order ID	Delivery State	Amount
1001	Ord936254	CA	$340
1001	Ord143876	TX	$950
1001	Ord654876	TX	$670
1001	Ord814356	TX	$860
2002	Ord342176	WA	$320
3003	Ord265789	CA	$650
3003	Ord387654	CA	$830
4004	Ord476126	TX	$120

Here is the expected output.

Customer ID	Order ID	Delivery State	Amount
1001	Ord143876	TX	$950
1001	Ord654876	TX	$670
1001	Ord814356	TX	$860

Customer ID 1001 would be in the expected output as this customer had deliveries to both California and Texas. Customer ID 3003 would not show in the result set as they did not have a delivery to Texas, and Customer ID 4004 would not appear in the result set as they did not have a delivery to California.

Hyperlink to the solution

Puzzle #5

Phone Directory

Your customer phone directory table allows individuals to setup a home, cellular, or a work phone number.

Write an SQL statement to transform the following table into the expected output.

Customer ID	Type	Phone Number
1001	Cellular	555-897-5421
1001	Work	555-897-6542
1001	Home	555-698-9874
2002	Cellular	555-963-6544
2002	Work	555-812-9856
3003	Cellular	555-987-6541

Here is the expected output.

Customer ID	Cellular	Work	Home
1001	555-897-5421	555-897-6542	555-698-9874
2002	555-963-6544	555-812-9856	
3003	555-987-6541		

Hyperlink to the solution

Puzzle #6

Workflow Steps

Write an SQL statement that determines all workflows that have started but have not completed.

Workflow	Step Number	Completion Date
Alpha	1	7/2/2018
Alpha	2	7/2/2018
Alpha	3	7/1/2018
Bravo	1	6/25/2018
Bravo	2	
Bravo	3	6/27/2018
Charlie	1	
Charlie	2	7/1/2018

The expected output would be Bravo and Charlie, as they have a workflow that has started but has not completed.

Bonus: Write this query only using the COUNT function with no subqueries. Can you figure out the trick?

Hyperlink to the solution

Puzzle #7

Mission to Mars

You are given the following tables that list the requirements for a space mission and a list of potential candidates.

Write an SQL statement to determine which candidates meet the requirements of the mission.

Candidates

Candidate ID	Description
1001	Geologist
1001	Astrogator
1001	Biochemist
1001	Technician
2002	Surgeon
2002	Machinist
3003	Cryologist
4004	Selenologist

Requirements

Description
Geologist
Astrogator
Technician

The expected output would be Candidate ID 1001, as this candidate has all the necessary skills for the space mission. Candidate ID 2002 and 3003 would not be in the output as they have some, but not all the required skills.

Hyperlink to the solution

Puzzle #8

Workflow Cases

You have a report of all workflows and their case results.

A value of 0 signifies the workflow failed, and a value of 1 signifies the workflow passed.

Write an SQL statement that transforms the following table into the expected output.

Workflow	Case 1	Case 2	Case 3
Alpha	0	0	0
Bravo	0	1	1
Charlie	1	0	0
Delta	0	0	0

Here is the expected output.

Workflow	Passed
Alpha	0
Bravo	2
Charlie	1
Delta	0

Hyperlink to the solution

Puzzle #9

Matching Sets

Write an SQL statement that matches an employee to all other employees that carry the same licenses.

Employee ID	License
1001	Class A
1001	Class B
1001	Class C
2002	Class A
2002	Class B
2002	Class C
3003	Class A
3003	Class D

Employee ID 1001 and 2002 would be in the expected output as they both carry a Class A, Class B, and a Class C license.

Hyperlink to the solution

Puzzle #10

Mean, Median, Mode, and Range

The mean is the average of all numbers.

The median is the middle number in a sequence of numbers.

The mode is the number that occurs most often within a set of numbers.

The range is the difference between the highest and lowest values in a set of numbers.

Write an SQL statement to determine the mean, median, mode and range of the following set of integers.

```
CREATE TABLE #SampleData
(
IntegerValue INTEGER
);
GO

INSERT INTO #SampleData
VALUES(5),(6),(10),(10),(13),(14),(17),(20),(81),(90),(76);
GO
```

Hyperlink to the solution

Puzzle #11

Permutations

You are given the following list of test cases and must determine all possible permutations.

Write an SQL statement that produces the expected output.

Test Case
A
B
C

Here is the expected output.

Row Number	Output
1	A,B,C
2	A,C,B
3	B,A,C
4	B,C,A
5	C,A,B
6	C,B,A

Hyperlink to the solution

Puzzle #12

Average Days

Write an SQL statement to determine the average number of days between executions for each workflow.

Workflow	Execution Date
Alpha	6/1/2018
Alpha	6/14/2018
Alpha	6/15/2018
Bravo	6/1/2018
Bravo	6/2/2018
Bravo	6/19/2018
Charlie	6/1/2018
Charlie	6/15/2018
Charlie	6/30/2018

Here is the expected output.

Workflow	Average Days
Alpha	7
Bravo	9
Charlie	14

Hyperlink to the solution

Puzzle #13

Inventory Tracking

You work for a manufacturing company and need to track inventory adjustments from the warehouse.

Some days the inventory increases, on other days the inventory decreases.

Write an SQL statement that will provide a running balance of the inventory.

Date	Quantity Adjustment
7/1/2018	100
7/2/2018	75
7/3/2018	-150
7/4/2018	50
7/5/2018	-100

Here is the expected output.

Date	Quantity Adjustment	Inventory
7/1/2018	100	100
7/2/2018	75	175
7/3/2018	-150	25
7/4/2018	50	75
7/5/2018	-50	25

Hyperlink to the solution

Puzzle #14

Indeterminate Process Log

Your process log has several workflows broken down by step numbers with the possible status values of Complete, Running, or Error.

Your task is to write an SQL statement that creates an overall status based upon the following requirements.

- If all the workflow steps have a status of complete, set the overall status to complete. (ex. Bravo).
- If all the workflow steps have a status of error, set the overall status to error (ex. Foxtrot).
- If the workflow steps have the combination of error and complete, or error and running, the overall status should be indeterminate (ex. Alpha, Charlie, Echo).
- If the workflow steps have the combination of complete and running, the overall status should be running (ex. Delta).

Workflow	Step Number	Status
Alpha	1	Error
Alpha	2	Complete
Bravo	1	Complete
Bravo	2	Complete
Charlie	1	Complete
Charlie	2	Error
Delta	1	Complete
Delta	2	Running
Echo	1	Running
Echo	2	Error
Foxtrot	1	Error

Here is the expected output.

Workflow	Status
Alpha	Indeterminate
Bravo	Complete
Charlie	Indeterminate
Delta	Running
Echo	Indeterminate
Foxtrot	Error

Hyperlink to the solution

Puzzle #15

Group Concatenation

Write an SQL statement that can group concatenate the following values.

Sequence	Syntax
1	SELECT
2	Product
3	UnitPrice
4	EffectiveDate
5	FROM
6	Products
7	WHERE
8	UnitPrice
9	> 100

Here is the expected output.

Syntax
SELECT Product, UnitPrice, EffectiveDate FROM Products WHERE UnitPrice > 100

Hyperlink to the solution

Puzzle #16

Reciprocals

You work for a software company that released a 2-player game and you need to tally the scores.

Given the following table, write an SQL statement to determine the reciprocals and calculate their aggregate score.

In the data below, players 3003 and 4004 have two valid entries, but their scores need to be aggregated together.

Player A	Player B	Score
1001	2002	150
3003	4004	15
4004	3003	125

Here is the expected output.

Player A	Player B	Score
1001	2002	150
3003	4004	140

Hyperlink to the solution

Puzzle #17

De-Grouping

Write an SQL Statement to de-group the following data.

Product	Quantity
Pencil	3
Eraser	4
Notebook	2

Here is the expected output.

Product	Quantity
Pencil	1
Pencil	1
Pencil	1
Eraser	1
Eraser	1
Eraser	1
Eraser	1
Notebook	1
Notebook	1

Hyperlink to the solution

Puzzle #18

Seating Chart

Given the following set of integers, write an SQL statement to determine the expected outputs.

```
CREATE TABLE #SeatingChart
(
SeatNumber INTEGER
);
GO

INSERT INTO #SeatingChart VALUES
(7),(13),(14),(15),(27),(28),(29),(30),(31),(32),(33),(34),(35
),(52),(53),(54);
GO
```

Here is the expected output.

Gap Start	Gap End
1	6
8	12
16	26
36	51

Total Missing Numbers
38

Type	Count
Even Numbers	8
Odd Numbers	9

Hyperlink to the solution

Puzzle #19

Back to the Future

Here is one of the more difficult puzzles to solve with a declarative SQL statement.

Write an SQL statement to merge the overlapping time periods.

Start Date	End Date
1/1/2018	1/5/2018
1/3/2018	1/9/2018
1/10/2018	1/11/2018
1/12/2018	1/16/2018
1/15/2018	1/19/2018

Here is the expected output.

Start Date	End Date
1/1/2018	1/9/2018
1/10/2018	1/11/2018
1/12/2018	1/19/2018

Hyperlink to the solution

Puzzle #20

Price Points

Write an SQL statement to determine the current price point for each product.

Product ID	Unit Price	Effective Date
1001	$1.99	1/1/2018
1001	$2.99	5/17/2018
1001	$3.99	6/14/2018
2002	$12.99	2/12/2018
2002	$17.99	3/1/2018
2002	$20.99	4/19/2018

Here is the expected output.

Product ID	Effective Date	Unit Price
1001	6/8/2018	$3.99
2002	5/19/2018	$2.99

Hyperlink to the solution

Puzzle #21

Average Monthly Sales

Write an SQL statement that returns a list of states where customers have an average monthly sales value that is consistently greater than $100.

Order ID	Customer ID	Order Date	Amount	State
Ord145332	1001	1/1/2018	$100	TX
Ord657895	1001	1/1/2018	$150	TX
Ord887612	1001	1/1/2018	$75	TX
Ord654374	1001	2/1/2018	$100	TX
Ord345362	1001	3/1/2018	$100	TX
Ord912376	2002	2/1/2018	$75	TX
Ord543219	2002	2/1/2018	$150	TX
Ord156357	3003	1/1/2018	$100	IA
Ord956541	3003	2/1/2018	$100	IA
Ord856993	3003	3/1/2018	$100	IA
Ord864573	4004	4/1/2018	$100	IA
Ord654525	4004	5/1/2018	$50	IA
Ord987654	4004	5/1/2018	$100	IA

In this example, Texas would show in the result set as Customer ID 1001 and 2002 each have their average monthly value over $100. Iowa would not show in the result set because Customer ID 3003 did not have an average monthly value over $100 in May 2018.

Hyperlink to the solution

Puzzle #22

Occurrences

Write an SQL statement that returns all distinct process log messages and the workflow where the message occurred the most often.

Workflow	Occurrences	Message
Alpha	5	Error: Conversion Failed
Alpha	8	Status Complete
Alpha	9	Error: Unidentified error occurred
Bravo	3	Error: Cannot Divide by 0
Bravo	1	Error: Unidentified error occurred
Charlie	10	Error: Unidentified error occurred
Charlie	7	Error: Conversion Failed
Charlie	6	Status Complete

Here is the expected output.

Workflow	Message
Alpha	Status Complete
Bravo	Error: Cannot Divide by 0
Charlie	Error: Conversion Failed
Charlie	Error: Unidentified error occurred

Hyperlink to the solution

Puzzle #23

Divide in Half

You work for a gaming company and need to rank players by their score into two categories.

Players that rank in the top half must be given a value of 1; the remaining players must be given a value of 2.

Write an SQL statement that meets these requirements.

```
CREATE TABLE #PlayerScores
(
PlayerID VARCHAR(MAX),
Score    INTEGER
);
GO

INSERT INTO #PlayerScores VALUES
(1001,2343),
(2002,9432),
(3003,6548),
(4004,1054),
(5005,6832);
GO
```

Hyperlink to the solution

Puzzle #24

Page Views

Write an SQL statement that retrieves records 10 to 20 ordered by the RowID column. Here is the syntax to create and populate the table.

```
IF OBJECT_ID('tempdb.dbo.#SampleData', 'U') IS NOT NULL
  DROP TABLE #SampleData;

CREATE TABLE #SampleData
(
IntegerValue  INTEGER IDENTITY(1,1),
RowID         UNIQUEIDENTIFIER
);
GO

INSERT INTO #SampleData VALUES (NEWID());
GO 1000

ALTER TABLE #SampleData DROP COLUMN IntegerValue;
GO
```

Hyperlink to the solution

Puzzle #25

Top Vendors

Write an SQL statement that returns the vendor from which each customer has placed the most orders.

Order ID	Customer ID	Order Count	Vendor
Ord195342	1001	12	Direct Parts
Ord245532	1001	54	Direct Parts
Ord344394	1001	32	ACME
Ord442423	2002	7	ACME
Ord524232	2002	16	ACME
Ord645363	2002	5	Direct Parts

Here is the expected output.

Customer ID	Vendor
1001	Direct Parts
2002	ACME

Hyperlink to the solution

Puzzle #26

Previous Year's Sales

Write an SQL statement that shows the current year's sales, along with the previous year's sales, and the sales from two years ago.

Year	Amount
2018	$352,645
2017	$165,565
2017	$254,654
2016	$159,521
2016	$251,696
2016	$111,894

Here is the expected output.

2018	2017	2016
$352,645	$420,219	$411,217

Hyperlink to the solution

Puzzle #27

Delete the Duplicates

Write an SQL statement that deletes the duplicate data.

```
CREATE TABLE #SampleData
(
IntegerValue INTEGER
);
GO

INSERT INTO #SampleData VALUES
(1),
(1),
(2),
(3),
(3),
(4);
GO
```

Hyperlink to the solution

Puzzle #28

Fill the Gaps

The answer to this problem is often referred to as a “data smear” or a “flash fill”.

Write an SQL statement to fill in the missing gaps.

Row Number	Workflow	Status
1	Alpha	Pass
2		Fail
3		Fail
4		Fail
5	Bravo	Pass
6		Fail
7		Fail
8		Pass
9		Pass
10	Charlie	Fail
11		Fail
12		Fail

Here is the expected output.

Row Number	Workflow	Status
1	Alpha	Pass
2	Alpha	Fail
3	Alpha	Fail
4	Alpha	Fail
5	Bravo	Pass
6	Bravo	Fail
7	Bravo	Fail
8	Bravo	Pass
9	Bravo	Pass
10	Charlie	Fail
11	Charlie	Fail
12	Charlie	Fail

Hyperlink to the solution

Puzzle #29

Count the Groupings

Write an SQL statement that counts the consecutive values in the Status field.

Step Number	Status
1	Passed
2	Passed
3	Passed
4	Passed
5	Failed
6	Failed
7	Failed
8	Failed
9	Failed
10	Passed
11	Passed
12	Passed

Here is the expected outcome.

Order	Status	Consecutive Counts
1	Passed	4
2	Failed	5
3	Passed	3

Hyperlink to the solution

Puzzle #30

Select Star

Your developers have many bad practices; the worst of them being they routinely deploy procedures that do not explicitly define which fields to return in their SELECT clause.

Modify the following table in such a way that the statement [SELECT * FROM Products] will return an error when executed.

```
CREATE TABLE #Products
(
ProductID     INTEGER,
ProductName   VARCHAR(MAX)
);
```

Hyperlink to the solution

Puzzle #31

Second Highest

How many different SQL statements can you write that will return the second highest integer?

```
CREATE TABLE #SampleData
(
IntegerValue INTEGER
);
GO

INSERT INTO #SampleData VALUES
(3759),(3760),(3761),(3762),(3763);
GO
```

Hyperlink to the solution

Puzzle #32

First and Last

Write an SQL statement that determines the most and least experienced Spaceman ID by their job description.

Spaceman ID	Job Description	Mission Count
1001	Astrogator	6
2002	Astrogator	12
3003	Astrogator	17
4004	Geologist	21
5005	Geologist	9
6006	Geologist	8
7007	Technician	13
8008	Technician	2
9009	Technician	7

Here is the expected output.

Job Description	Most Experienced	Least Experienced
Astrogator	3003	1001
Geologist	4004	6006
Technician	7007	8008

Hyperlink to the solution

Puzzle #33

Deadlines

Write an SQL statement that determines if an order will be fulfilled by the requested delivery date. Is there a better SQL construct to use then the MAX function?

Orders

Order ID	Product	Delivery Date (Days)
Ord893456	Widget	7
Ord923654	Gizmo	3
Ord187239	Doodad	9

Manufacturing Time

Part	Product	Days to Manufacture
AA-111	Widget	7
BB-222	Widget	2
CC-333	Widget	3
DD-444	Widget	1
AA-111	Gizmo	7
BB-222	Gizmo	2
AA-111	Doodad	7
DD-444	Doodad	1

Here is the expected output.

Order ID	Product
Ord893456	Widget
Ord187239	Doodad

Order ID Ord893456 and Ord187239 will be in the output as these orders have a promised delivery date that is equal to or greater than the days to manufacture.

Hyperlink to the solution

Puzzle #34

Specific Exclusion

Write an SQL statement that returns all rows except where the Customer ID is 1001 and the Amount is $50.

Customer ID	Order ID	Amount
1001	Ord143933	$25
1001	Ord789765	$50
2002	Ord345434	$65
3003	Ord785633	$50

Here is the expected output.

Customer ID	Order ID	Amount
1001	Ord143933	$25
2002	Ord345434	$65
3003	Ord785633	$50

Hyperlink to the solution

Puzzle #35

International vs Domestic Sales

You work in a sales office that sells widgets both domestically and internationally.

Write an SQL statement that shows all sales representatives who either had a domestic sale or an international sale, but not both.

Sales Rep ID	Invoice ID	Amount	Sales Type
1001	Inv345756	$13,454	International
2002	Inv546744	$3,434	International
4004	Inv234745	$54,645	International
5005	Inv895745	$234,345	International
7007	Inv006321	$776	International
1001	Inv734534	$4,564	Domestic
2002	Inv600213	$34,534	Domestic
3003	Inv757853	$345	Domestic
6006	Inv198632	$6,543	Domestic
8008	Inv977654	$67	Domestic

Sales Rep ID 3003, 4004, 5005 and 6006 would appear in the result set as they had either an international sale or a domestic sale, but not both.

Hyperlink to the solution

Puzzle #36

Traveling Salesman

Here is a well-known problem that is called the Traveling Salesman among programmers.

Write an SQL statement that shows all the possible routes from Austin to Des Moines. Which route is the most expensive? Which route is the least expensive? Make any necessary assumptions to complete the puzzle.

Departure City	Arrival City	Cost
Austin	Dallas	$100
Dallas	Memphis	$200
Memphis	Des Moines	$300
Dallas	Des Moines	$400

Hyperlink to the solution

Puzzle #37

Group Criteria Keys

Write an SQL statement that provides a key based upon the distinct combination of distributor, facility, and zone.

Order ID	Distributor	Facility	Zone	Amount
Ord156795	ACME	123	ABC	$100
Ord826109	ACME	123	ABC	$75
Ord342876	Direct Parts	789	XYZ	$150
Ord994981	Direct Parts	789	XYZ	$125

Here is the expected output.

Criteria ID	Order ID	Distributor	Facility	Zone	Amount
1	Ord156795	ACME	123	ABC	$100
1	Ord826109	ACME	123	ABC	$75
2	Ord342876	Direct Parts	789	XYZ	$150
2	Ord994981	Direct Parts	789	XYZ	$125

Hyperlink to the solution

Puzzle #38

Reporting Elements

You must provide a report of all distributors and their sales by region. If a distributor did not have any sales for a region, provide a zero-dollar amount for that day. Assume there is at least one sale for each region.

Region	Distributor	Sales
North	ACE	10
South	ACE	67
East	ACE	54
North	Direct Parts	8
South	Direct Parts	7
West	Direct Parts	12
North	ACME	65
South	ACME	9
East	ACME	1
West	ACME	7

Here is the expected output.

Region	Distributor	Sales
North	ACE	10
South	ACE	67
East	ACE	54
West	ACE	0
North	ACME	65
South	ACME	9
East	ACME	1
West	ACME	7
North	Direct Parts	8
South	Direct Parts	7
East	Direct Parts	0
West	Direct Parts	12

Hyperlink to the solution

Puzzle #39

Prime Numbers

Write an SQL statement to determine which of the below numbers are prime numbers.

```
CREATE TABLE #SampleData
(
IntegerValue INTEGER
);
GO

INSERT INTO #SampleData VALUES
(1),(2),(3),(4),(5),(6),(7),(8),(9),(10);
GO
```

Hyperlink to the solution

Puzzle #40

Sort Order

Write an SQL statement that sorts the following values into the expected output. Can you find the most elegant solution?

City
Atlanta
Baltimore
Chicago
Denver

Here is the expected output.

City
Baltimore
Denver
Atlanta
Chicago

Hyperlink to the solution

More SQL Questions

1) How do the various SQL constructs (Join, Set Operators, GROUP BY, 0RDER BY, Constraints) treat the NULL marker? What is meant by the three-valued system?

2) In what order does the SQL parser read the SQL constructs (SELECT, FROM, WHERE, GROUP BY HAVING, ORDER BY) and how does this affect how you write your SQL statements?

3) Write a procedure that compiles, but produces an error at runtime.

4) SQL is considered a declarative language. Describe what this mean.

5) In what ways does SQL differ from relational algebra?

6) What is the purpose of the EXISTS and NOT EXISTS clause in SQL? How are they different from IN and NOT IN clauses?

7) What is the difference between deterministic and nondeterministic functions, and why is this important?

8) When is set based looping preferable over cursor based looping, and vice versa?

9) Name the various logical SQL functions and when you would use them. What about the mathematical functions, date functions, etc.

10) What are the differences between DDL and DML statements?

Answers

Answer to Puzzle #1

Dance Partners

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#DancePartners','U') IS NOT NULL
  DROP TABLE #DancePartners;
GO

CREATE TABLE #DancePartners
(
StudentID INTEGER,
Gender    VARCHAR(1)
);
GO

INSERT INTO #DancePartners VALUES
(1001,'M'),
(2002,'M'),
(3003,'M'),
(4004,'M'),
(5005,'M'),
(6006,'F'),
(7007,'F'),
(8008,'F'),
(9009,'F');
GO

WITH cte_Males AS
(
SELECT  ROW_NUMBER () OVER (ORDER BY StudentID) AS RowNumber,
        StudentID,
        Gender
FROM    #DancePartners
WHERE   Gender = 'M'
),
cte_Females AS
(
SELECT  ROW_NUMBER () OVER (ORDER BY StudentID) AS RowNumber,
        StudentID,
        Gender
FROM    #DancePartners
WHERE   Gender = 'F'
)
SELECT  a.StudentID, a.Gender, b.StudentID, b.Gender
FROM    cte_Males a FULL OUTER JOIN
        cte_Females b ON a.RowNumber = B.RowNumber;
```

Answer to Puzzle #2

Managers and Employees

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#Employees','U') IS NOT NULL
  DROP TABLE #Employees;
GO

CREATE TABLE #Employees
(
EmployeeID INTEGER,
ManagerID  INTEGER,
JobTitle   VARCHAR(MAX),
Salary     INTEGER
);
GO

INSERT INTO #Employees VALUES
(1001,NULL,'President',185000),
(2002,1001,'Director',120000),
(3003,1001,'Office Manager',97000),
(4004,2002,'Engineer',110000),
(5005,2002,'Engineer',142000),
(6006,2002,'Engineer',160000);
GO

WITH cte_Recursion AS
(
SELECT  EmployeeID, ManagerID, JobTitle, Salary, 0 as Depth
FROM    #Employees a
WHERE   ManagerID IS NULL
UNION ALL
SELECT  b.EmployeeID, b.ManagerID, b.JobTitle, b.Salary, a.Depth + 1 as Depth
FROM    cte_Recursion a INNER JOIN
        #Employees b ON a.EmployeeID = b.ManagerID
)
SELECT  EmployeeID, ManagerID, JobTitle, Salary, Depth
FROM    cte_Recursion;
```

Answer to Puzzle #3

Fiscal Year Table Constraints

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#EmployeePayRecord','U') IS NOT NULL
  DROP TABLE #EmployeePayRecord;
GO

CREATE TABLE #EmployeePayRecord
(
EmployeeID INTEGER,
FiscalYear INTEGER,
StartDate  DATE,
EndDate    DATE,
PayRate    MONEY
);
GO

ALTER TABLE #EmployeePayRecord ALTER COLUMN EmployeeID INTEGER NOT NULL;
ALTER TABLE #EmployeePayRecord ALTER COLUMN FiscalYear INTEGER NOT NULL;
ALTER TABLE #EmployeePayRecord ALTER COLUMN StartDate DATE NOT NULL;
ALTER TABLE #EmployeePayRecord ALTER COLUMN EndDate DATE NOT NULL;
ALTER TABLE #EmployeePayRecord ALTER COLUMN PayRate MONEY NOT NULL;
GO
ALTER TABLE #EmployeePayRecord ADD CONSTRAINT PK_FiscalYearCalendar
                                    PRIMARY KEY (EmployeeID,FiscalYear);
ALTER TABLE #EmployeePayRecord ADD CONSTRAINT Check_Year_StartDate
                                    CHECK (FiscalYear = DATEPART(YYYY,StartDate));
ALTER TABLE #EmployeePayRecord ADD CONSTRAINT Check_Month_StartDate
                                    CHECK (DATEPART(MM,StartDate) = 01);
ALTER TABLE #EmployeePayRecord ADD CONSTRAINT Check_Day_StartDate
                                    CHECK (DATEPART(DD,StartDate) = 01);
ALTER TABLE #EmployeePayRecord ADD CONSTRAINT Check_Year_EndDate
                                    CHECK (FiscalYear = DATEPART(YYYY,EndDate));
ALTER TABLE #EmployeePayRecord ADD CONSTRAINT Check_Month_EndDate
                                    CHECK (DATEPART(MM,EndDate) = 12);
ALTER TABLE #EmployeePayRecord ADD CONSTRAINT Check_Day_EndDate
                                    CHECK (DATEPART(DD,EndDate) = 31);
GO
ALTER TABLE #EmployeePayRecord ADD CHECK (PayRate > 0);
GO
```

Answer to Puzzle #4

Two Predicates

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#Orders','U')IS NOT NULL
  DROP TABLE #Orders;
GO

CREATE TABLE #Orders
(
CustomerID    INTEGER,
OrderID       VARCHAR(MAX),
DeliveryState VARCHAR(MAX),
Amount        MONEY
);
GO

INSERT INTO #Orders VALUES
(1001,'Ord936254','CA',340),
(1001,'Ord143876','TX',950),
(1001,'Ord654876','TX',670),
(1001,'Ord814356','TX',860),
(2002,'Ord342176','WA',320),
(3003,'Ord265789','CA',650),
(3003,'Ord387654','CA',830),
(3003,'Ord476126','TX',120);
GO

--INNER JOIN
WITH cte_CA AS
(
SELECT  DISTINCT CustomerID
FROM    #Orders
WHERE   DeliveryState = 'CA'
)
SELECT  b.CustomerID, b.OrderID, b.DeliveryState, b.Amount
FROM    cte_CA a INNER JOIN
        #Orders b ON a.CustomerID = B.CustomerID
WHERE   b.DeliveryState = 'TX';

--IN Clause
WITH cte_CA AS
(
SELECT  CustomerID
FROM    #Orders
WHERE   DeliveryState = 'CA'
)
SELECT  CustomerID, OrderID, DeliveryState, Amount
FROM    #Orders
WHERE   DeliveryState = 'TX' AND
        CustomerID IN (SELECT b.CustomerID FROM cte_CA b);
```

```
--Correlated Subquery
WITH cte_CA AS
(
SELECT  CustomerID
FROM    #Orders
WHERE   DeliveryState = 'CA'
)
SELECT  CustomerID, OrderID, DeliveryState, Amount
FROM    #Orders a
WHERE   DeliveryState = 'TX' AND
        EXISTS(SELECT CustomerID FROM cte_CA b WHERE a.CustomerID = b.CustomerID);
```

Answer to Puzzle #5

Phone Directory

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#PhoneDirectory','U') IS NOT NULL
  DROP TABLE #PhoneDirectory;
GO

CREATE TABLE #PhoneDirectory
(
CustomerID  INTEGER,
Type        VARCHAR(MAX),
PhoneNumber VARCHAR(MAX)
);
GO

INSERT INTO #PhoneDirectory VALUES
(1001,'Cellular','555-897-5421'),
(1001,'Work','555-897-6542'),
(1001,'Home','555-698-9874'),
(2002,'Cellular','555-963-6544'),
(2002,'Work','555-812-9856'),
(3003,'Cellular','555-987-6541');
GO

--PIVOT
SELECT CustomerID,[Cellular],[Work],[Home] FROM #PhoneDirectory
PIVOT (MAX(PhoneNumber) FOR Type IN ([Cellular],[Work],[Home])) AS PivotClause;

--OUTER JOIN
WITH cte_Cellular AS
(
SELECT  CustomerID, PhoneNumber AS Cellular
FROM    #PhoneDirectory
WHERE   Type = 'Cellular'
),
cte_Work AS
(
SELECT  CustomerID, PhoneNumber AS Work
FROM    #PhoneDirectory
WHERE   Type = 'Work'
),
cte_Home AS
(
SELECT  CustomerID, PhoneNumber AS Home
FROM    #PhoneDirectory
WHERE   Type = 'Home'
)
SELECT  a.CustomerID,b.Cellular,c.Work,d.Home
FROM    (
    SELECT  DISTINCT CustomerID
    FROM    #Phonedirectory
    ) a LEFT OUTER JOIN
```

```
    cte_Cellular b ON a.CustomerID = b.CustomerID LEFT OUTER JOIN
    cte_Work c ON a.CustomerID = c.CustomerID LEFT OUTER JOIN
    cte_Home d ON a.CustomerID = d.CustomerID;

--MAX function with UNION set operators
WITH cte_PhoneNumbers AS
(
SELECT  CustomerID,
        PhoneNumber AS Cellular,
        NULL AS work,
        NULL AS home
FROM    #PhoneDirectory
WHERE   Type = 'CellPhone'
UNION
SELECT  CustomerID,
        NULL Cellular,
        PhoneNumber AS Work,
        NULL home
FROM    #PhoneDirectory
WHERE   Type = 'Work'
UNION
SELECT  CustomerID,
        NULL Cellular,
        NULL Work,
        PhoneNumber AS Home
FROM    #PhoneDirectory
WHERE   Type = 'Home'
)
SELECT  CustomerID,
        MAX(Cellular),
        MAX(Work),
        MAX(Home)
FROM    cte_PhoneNumbers
GROUP BY CustomerID;
```

Answer to Puzzle #6

Workflow Steps

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#WorkflowSteps','U') IS NOT NULL
  DROP TABLE #WorkflowSteps;
GO

CREATE TABLE #WorkflowSteps
(
Workflow       VARCHAR(MAX),
StepNumber     INTEGER,
CompletionDate DATE
);
GO

INSERT INTO #WorkflowSteps VALUES
('Alpha',1,'7/2/2018'),
('Alpha',2,'7/2/2018'),
('Alpha',3,'7/1/2018'),
('Bravo',1,'6/25/2018'),
('Bravo',2,NULL),
('Bravo',3,'6/27/2018'),
('Charlie',1,NULL),
('Charlie',2,'7/1/2018');
GO

SELECT  Workflow
FROM    #WorkflowSteps
GROUP BY Workflow
HAVING  COUNT(*) <> COUNT(CompletionDate);
```

Answer to Puzzle #7

Mission to Mars

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#Candidates','U') IS NOT NULL
  DROP TABLE #Candidates;
GO

IF OBJECT_ID('tempdb.dbo.#Requirements','U') IS NOT NULL
  DROP TABLE #Requirements;
GO

CREATE TABLE #Candidates
(
CandidateID INTEGER,
Occupation  VARCHAR(MAX)
);
GO

INSERT INTO #Candidates VALUES
(1001,'Geologist'),
(1001,'Astrogator'),
(1001,'Biochemist'),
(1001,'Technician'),
(2002,'Surgeon'),
(2002,'Machinist'),
(3003,'Cryologist'),
(4004,'Selenologist');
GO

CREATE TABLE #Requirements
(
Requirement VARCHAR(MAX)
);
GO

INSERT INTO #Requirements VALUES
('Geologist'),('Astrogator'),('Technician');
GO

WITH cte_RequirementsCount
AS
(
SELECT COUNT(*) AS RequirementCount FROM #Requirements
)
SELECT  CandidateID
FROM    #Candidates a INNER JOIN
        #Requirements b ON a.Occupation = b.Requirement
GROUP BY CandidateID
HAVING COUNT(*) = (SELECT RequirementCount FROM cte_RequirementsCount);
```

Answer to Puzzle #8

Workflow Cases

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#WorkflowCases','U') IS NOT NULL
  DROP TABLE #WorkflowCases;
GO

CREATE TABLE #WorkflowCases
(
Workflow VARCHAR(MAX),
Case1    INTEGER,
Case2    INTEGER,
Case3    INTEGER
);
GO

INSERT INTO #WorkflowCases VALUES
('Alpha',0,0,0),
('Bravo',0,1,1),
('Charlie',1,0,0),
('Delta',0,0,0);
GO

WITH cte_PassFail AS
(
SELECT  Workflow, CaseNumber, PassFail
FROM    (SELECT Workflow,Case1,Case2,Case3
        FROM #WorkflowCases) p
UNPIVOT (PassFail FOR CaseNumber IN (Case1,Case2,Case3)) AS UNPVT
)
SELECT  Workflow, SUM(PassFail) AS PassFail
FROM    cte_PassFail
GROUP BY Workflow
ORDER BY 1;
```

Answer to Puzzle #9

Matching Sets

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#Employees','U') IS NOT NULL
  DROP TABLE #Employees;
GO

CREATE TABLE #Employees
(
EmployeeID INTEGER,
License    VARCHAR(MAX)
);
GO

INSERT INTO #Employees VALUES
(1001,'Class A'),
(1001,'Class B'),
(1001,'Class C'),
(2002,'Class A'),
(2002,'Class B'),
(2002,'Class C'),
(3003,'Class A'),
(3003,'Class D');
GO

WITH cte_EmployeeCount AS
(
SELECT  EmployeeID,
        COUNT(*) AS LicenseCount
FROM    #Employees
GROUP BY EmployeeID
),
cte_EmployeeCountCombined AS
(
SELECT  a.EmployeeID AS EmployeeID,
        b.EmployeeID AS EmployeeID2,
        COUNT(*) AS LicenseCountCombo
FROM    #Employees a INNER JOIN
        #Employees b ON a.License = b.License
WHERE   a.EmployeeID <> b.EmployeeID
GROUP BY a.EmployeeID, b.EmployeeID
)
SELECT  a.EmployeeID, a.EmployeeID2, a.LicenseCountCombo
FROM    cte_EmployeeCountCombined a INNER JOIN
        cte_EmployeeCount b ON a.LicenseCountCombo = b.LicenseCount AND
                               a.EmployeeID <> b.EmployeeID;
```

Answer to Puzzle #10

Mean, Median, Mode and Range

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#SampleData','U') IS NOT NULL
  DROP TABLE #SampleData;
GO

CREATE TABLE #SampleData
(
IntegerValue INTEGER
);
GO

INSERT INTO #SampleData VALUES
(5),(6),(10),(10),(13),(14),(17),(20),(81),(90),(76);
GO

WITH cte_Median AS
(
SELECT Median =
        ((SELECT TOP 1 IntegerValue
        FROM (
              SELECT  TOP 50 PERCENT IntegerValue
              FROM    #SampleData
              WHERE   IntegerValue IS NOT NULL
              ORDER BY IntegerValue
             ) AS A
        ORDER BY IntegerValue DESC) +  --Add the Two Together
        (SELECT TOP 1 IntegerValue
        FROM (
             SELECT TOP 50 PERCENT IntegerValue
             FROM    #SampleData
             WHERE   IntegerValue Is NOT NULL
             ORDER BY IntegerValue DESC
             ) AS A
        ORDER BY IntegerValue ASC)
        )/2
),
cte_MeanAndRange AS
(
SELECT  AVG(IntegerValue) Mean, MAX(IntegerValue) - MIN(IntegerValue) AS MaxMinRange
FROM    #SampleData
),
cte_Mode AS
(
SELECT  TOP 1 IntegerValue AS Mode, COUNT(*) AS ModeCount
FROM    #SampleData
GROUP BY IntegerValue
ORDER BY ModeCount DESC
)
SELECT  Mean, Median, Mode , MaxMinRange AS [Range]
FROM    cte_Median CROSS JOIN cte_MeanAndRange CROSS JOIN cte_Mode;
```

Answer to Puzzle #11

Permutations

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#TestCases','U') IS NOT NULL
  DROP TABLE #TestCases;
GO

CREATE TABLE #TestCases
(
RowNumber INTEGER,
TestCase VARCHAR(1)
);
GO

INSERT INTO #TestCases VALUES
(1,'A'),(2,'B'),(3,'C');
GO

DECLARE @vTotalElements INTEGER = (SELECT COUNT(*) FROM #TestCases);

WITH cte_Permutations (Permutation, Ids, Depth)
AS
(
SELECT CAST(TestCase AS VARCHAR(MAX)),
              CAST(RowNumber AS VARCHAR(MAX)) + ';',
              1 AS Depth
FROM   #TestCases
UNION ALL
SELECT a.Permutation + ',' + b.TestCase,
              a.Ids + CAST(b.RowNumber AS VARCHAR) + ';',
              a.Depth + 1
FROM   cte_Permutations a,
              #TestCases b
WHERE  a.Depth < @vTotalElements AND
              a.Ids NOT LIKE '%' + CAST(b.RowNumber AS VARCHAR) + ';%'
)
SELECT Permutation
FROM   cte_Permutations
WHERE  Depth = @vTotalElements;
```

Answer to Puzzle #12

Average Days

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#ProcessLog','U') IS NOT NULL
  DROP TABLE #ProcessLog;
GO

CREATE TABLE #ProcessLog
(
WorkFlow      VARCHAR(MAX),
ExecutionDate DATE
);
GO

INSERT INTO #ProcessLog VALUES
('Alpha','6/01/2018'),
('Alpha','6/14/2018'),
('Alpha','6/15/2018'),
('Bravo','6/1/2018'),
('Bravo','6/2/2018'),
('Bravo','6/19/2018'),
('Charlie','6/1/2018'),
('Charlie','6/15/2018'),
('Charlie','6/30/2018');
GO

WITH cte_DayDiff AS
(
SELECT  WorkFlow,
        (DATEDIFF(DD,LAG(ExecutionDate,1,NULL) OVER
                (PARTITION BY WorkFlow ORDER BY ExecutionDate),ExecutionDate)) AS
DateDifference
FROM    #ProcessLog
)
SELECT  WorkFlow, AVG(DateDifference)
FROM    cte_DayDiff
WHERE   DateDifference IS NOT NULL
GROUP BY Workflow;
```

Answer to Puzzle #13

Inventory Tracking

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#Inventory','U') IS NOT NULL
  DROP TABLE #Inventory;
GO

CREATE TABLE #Inventory
(
InventoryDate      DATE,
QuantityAdjustment INTEGER
);
GO

INSERT INTO #Inventory VALUES
('7/1/2018',100),
('7/2/2018',75),
('7/3/2018',-150),
('7/4/2018',50),
('7/5/2018',-75);
GO

SELECT  InventoryDate,
        QuantityAdjustment,
        SUM(QuantityAdjustment) OVER (ORDER BY InventoryDate)
FROM    #Inventory;
```

Answer to Puzzle #14

Indeterminate Process Log

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#ProcessLog','U') IS NOT NULL
  DROP TABLE #ProcessLog;
GO

CREATE TABLE #ProcessLog
(
Workflow   VARCHAR(MAX),
StepNumber INTEGER,
Status     VARCHAR(MAX)
);
GO

INSERT INTO #ProcessLog VALUES
('Alpha',1,'Error'),
('Alpha',2,'Complete'),
('Bravo',1,'Complete'),
('Bravo',2,'Complete'),
('Charlie',1,'Complete'),
('Charlie',2,'Error'),
('Delta',1,'Complete'),
('Delta',2,'Running'),
('Echo',1,'Running'),
('Echo',2,'Error'),
('Foxtrot',1,'Error'),
('Foxtrot',2,'Error');
GO

--Create a StatusRank table to solve the problem
IF OBJECT_ID('tempdb.dbo.#StatusRank','U') IS NOT NULL
  DROP TABLE #StatusRank;
GO

CREATE TABLE #StatusRank
(
Status VARCHAR(MAX),
Rank   INTEGER
);
GO

INSERT INTO #StatusRank VALUES
('Error',1),
('Running',2),
('Complete',3);
GO
```

```
WITH cte_CountExistsError AS
(
SELECT  Workflow, COUNT(DISTINCT Status) AS DistinctCount
FROM    #ProcessLog a
WHERE   EXISTS  (SELECT 1
FROM    #ProcessLog b
WHERE STATUS = 'Error' AND a.Workflow = b.Workflow)
GROUP BY Workflow
),
cte_ErrorWorkflows AS
(
SELECT  a.Workflow,
        (CASE WHEN DistinctCount > 1 THEN 'Indeterminate' ELSE a.Status END) AS Status
FROM    #ProcessLog a INNER JOIN
        cte_CountExistsError b ON a.WorkFlow = b.WorkFlow
GROUP BY a.WorkFlow, (CASE WHEN DistinctCount > 1 THEN 'Indeterminate' ELSE a.Status END)
)
SELECT  DISTINCT
        a.Workflow,
        FIRST_VALUE(a.Status) OVER (PARTITION  BY a.Workflow ORDER BY b.Rank) AS Status
FROM    #ProcessLog a INNER JOIN
        #StatusRank b on a.Status = b.Status
WHERE   a.Workflow NOT IN (SELECT Workflow FROM cte_ErrorWorkflows)
UNION
SELECT  Workflow, Status
FROM    cte_ErrorWorkflows
ORDER BY a.Workflow;
```

Answer to Puzzle #15

Group Concatenation

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#DMLTable','U') IS NOT NULL
  DROP TABLE #DMLTable;
GO

CREATE TABLE #DMLTable
(
SequenceNumber INTEGER,
String      VARCHAR(MAX)
);
GO

INSERT INTO #DMLTable VALUES
(1,'SELECT'),
(5,'FROM'),
(7,'WHERE'),
(2,'Product'),
(6,'Products'),
(3,'UnitPrice'),
(9,'> 100'),
(4,'EffectiveDate'),
(8,'UnitPrice');
GO

--CTE
WITH
cte_DMLGroupConcat(String2,Depth) AS
(
SELECT CAST('' AS NVARCHAR(MAX)),
       CAST(MAX(SequenceNumber) AS INTEGER)
FROM   #DMLTable
UNION ALL
SELECT cte_Ordered.String + ' ' + cte_Concat.String2, cte_Concat.Depth-1
FROM   cte_DMLGroupConcat cte_Concat INNER JOIN
       #DMLTable cte_Ordered ON cte_Concat.Depth = cte_Ordered.SequenceNumber
)
SELECT String2
FROM   cte_DMLGroupConcat
WHERE  Depth = 0;


--XML PATH
SELECT DISTINCT
             STUFF((
                    SELECT CAST(' ' AS VARCHAR(MAX)) + String
                    FROM   #DMLTable U
                    ORDER BY SequenceNumber
             FOR XML PATH('')), 1, 1, '') AS DML_String
FROM   #DMLTable;
```

Answer to Puzzle #16

Reciprocals

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#PlayerScores','U') IS NOT NULL
  DROP TABLE #PlayerScores;
GO

CREATE TABLE #PlayerScores
(
PlayerA INTEGER,
PlayerB INTEGER,
Score   INTEGER
);
GO

INSERT INTO #PlayerScores VALUES
(1001,2002,150),
(3003,4004,15),
(4004,3003,125),
(4004,1001,125);
GO

SELECT a.PlayerA, a.PlayerB, SUM(Score)
FROM   (
       SELECT
       (CASE WHEN PlayerA <= PlayerB THEN PlayerA ELSE PlayerB END) PlayerA,
       (CASE WHEN PlayerA <= PlayerB THEN PlayerB ELSE PlayerA END) PlayerB,
       Score
       FROM #PlayerScores
       ) a
GROUP BY PlayerA, PlayerB;
```

Answer to Puzzle #17

De-Grouping

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#Ungroup','U') IS NOT NULL
  DROP TABLE #Ungroup;
GO

CREATE TABLE #Ungroup
(
ProductDescription VARCHAR(MAX),
Quantity           INTEGER
);
GO

INSERT INTO #Ungroup VALUES
('Eraser',3),
('Pencil',4),
('Sharpener',2);
GO

--Build a Numbers Table
IF OBJECT_ID('tempdb.dbo.#Numbers','U') IS NOT NULL
  DROP TABLE #Numbers;
GO

CREATE TABLE #Numbers
(
IntegerValue INTEGER IDENTITY(1,1),
RowID        UNIQUEIDENTIFIER
);
GO

INSERT INTO #Numbers VALUES (NEWID());
GO 1000

SELECT  a.ProductDescription, 1 AS Quantity
FROM    #Ungroup a CROSS JOIN
        #Numbers b
WHERE   a.Quantity >= b. IntegerValue;
```

Answer to Puzzle #18

Seating Chart

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#SeatingChart','U') IS NOT NULL
  DROP TABLE #SeatingChart;
GO

CREATE TABLE #SeatingChart
(
SeatNumber INTEGER
);
GO

INSERT INTO #SeatingChart VALUES
(7),(13),(14),(15),(27),(28),(29),(30),(31),(32),(33),(34),(35),(52),(53),(54);
GO

--Place a value of 0 in the SeatingChart table
INSERT INTO #SeatingChart VALUES (0);
GO

SELECT GapStart + 1 GapStart , GapEnd - 1 GapEnd
FROM
       (
       SELECT SeatNumber AS GapStart ,
              LEAD(SeatNumber,1,0) OVER (ORDER BY SeatNumber) GapEnd ,
              LEAD(SeatNumber,1,0) OVER (ORDER BY SeatNumber) - SeatNumber Gap
       FROM #SeatingChart
) a
WHERE Gap > 1;

WITH cte_Rank
AS
(
SELECT SeatNumber,
       ROW_NUMBER() OVER (ORDER BY SeatNumber) AS RowNumber,
       SeatNumber - ROW_NUMBER() OVER (ORDER BY SeatNumber) AS Rnk
FROM   #SeatingChart
WHERE  SeatNumber > 0
)
SELECT MAX(Rnk) AS MissingNumbers FROM cte_Rank;

SELECT (CASE SeatNumber%2 WHEN 1 THEN 'Odd' WHEN 0 THEN 'Even' END) AS Modulus,
       COUNT(*) AS COUNT
FROM   #SeatingChart
GROUP BY (CASE SeatNumber%2 WHEN 1 THEN 'Odd' WHEN 0 THEN 'Even' END);
```

Answer to Puzzle #19

Back to the Future

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#TimePeriods','U') IS NOT NULL
  DROP TABLE #TimePeriods;
GO

CREATE TABLE #TimePeriods
(
StartDate DATE,
EndDate   DATE
);
GO

INSERT INTO #TimePeriods VALUES
('1/1/2018','1/5/2018'),
('1/3/2018','1/9/2018'),
('1/10/2018','1/11/2018'),
('1/12/2018','1/16/2018'),
('1/15/2018','1/19/2018');
GO

WITH cte_TimePeriod_Merge AS
(
SELECT  a.StartDate, MIN(b.EndDate) AS EndDate
FROM    (SELECT  t1.StartDate
         FROM    #TimePeriods AS t1 LEFT OUTER JOIN
                 #TimePeriods AS t2 ON t1.StartDate > t2.StartDate AND
                                       t1.StartDate <= t2.StartDate AND
                                       t1.StartDate <= t2.EndDate
         GROUP BY t1.StartDate
         HAVING COUNT(t2.StartDate) = 0
        ) AS a INNER JOIN
        (SELECT  t3.EndDate
         FROM    #TimePeriods AS t3 LEFT OUTER JOIN
                 #TimePeriods AS t4 ON t3.EndDate >= t4.StartDate AND
                                       t3.EndDate < t4.EndDate
         GROUP BY t3.EndDate
         HAVING COUNT(t4.StartDate) = 0
        ) AS b ON a.StartDate <= b.EndDate
GROUP BY a.StartDate
)
SELECT  MIN(StartDate) as StartDate,
        MAX(EndDate) as EndDate
FROM    cte_TimePeriod_Merge
GROUP BY EndDate;
```

Answer to Puzzle #20

Price Points

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#ValidPrices','U') IS NOT NULL
  DROP TABLE #ValidPrices;
GO

CREATE TABLE #ValidPrices
(
ProductID     VARCHAR(MAX),
UnitPrice     MONEY,
EffectiveDate DATE
);
GO

INSERT INTO #ValidPrices VALUES
(1001,1.99,'1/01/2018'),
(1001,2.99,'4/15/2018'),
(1001,3.99,'6/8/2018'),
(2002,1.99,'4/17/2018'),
(2002,2.99,'5/19/2018');
GO

--Correlated Subquery
SELECT  ProductID,
        EffectiveDate,
        COALESCE(UnitPrice,0) AS UnitPrice
FROM    #ValidPrices AS pp
WHERE NOT EXISTS (SELECT  1
                  FROM    #validprices AS ppl
                  WHERE   ppl.ProductID = pp.ProductID AND
                          ppl.EffectiveDate > pp.EffectiveDate);

--RANK function
WITH cte_validprices AS
(
SELECT  RANK() OVER (PARTITION BY ProductID ORDER BY EffectiveDate DESC) AS Rnk,
        ProductID,
        EffectiveDate,
        UnitPrice
FROM    #ValidPrices
)
SELECT  Rnk, ProductID, EffectiveDate, UnitPrice
FROM    cte_ValidPrices
WHERE   Rnk = 1;
```

Answer to Puzzle #21

Average Monthly Sales

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#Orders','U') IS NOT NULL
  DROP TABLE #Orders;
GO

CREATE TABLE #Orders
(
OrderID    VARCHAR(MAX),
CustomerID INTEGER,
OrderDate  DATE,
Amount     MONEY,
State      VARCHAR(MAX)
);
GO

INSERT INTO #Orders VALUES
('Ord145332',1001,'1/1/2018',100,'TX'),
('Ord657895',1001,'1/1/2018',150,'TX'),
('Ord887612',1001,'1/1/2018',75,'TX'),
('Ord654374',1001,'2/1/2018',100,'TX'),
('Ord345362',1001,'3/1/2018',100,'TX'),
('Ord912376',2002,'2/1/2018',75,'TX'),
('Ord543219',2002,'2/1/2018',150,'TX'),
('Ord156357',3003,'1/1/2018',100,'IA'),
('Ord956541',3003,'2/1/2018',100,'IA'),
('Ord856993',3003,'3/1/2018',100,'IA'),
('Ord864573',4004,'4/1/2018',100,'IA'),
('Ord654525',4004,'5/1/2018',50,'IA'),
('Ord987654',4004,'5/1/2018',100,'IA');
GO

--AVG and MIN function
WITH cte_AvgMonthlySalesCustomer AS
(
SELECT  CustomerID, AVG(b.Amount) AS AverageValue,
        State
FROM    #Orders b
GROUP BY CustomerID,OrderDate,State
),
cte_MinAverageValueState AS
(
SELECT  State
FROM    cte_AvgMonthlySalesCustomer a
GROUP BY State
HAVING  MIN(AverageValue) >= 100
)
SELECT  State
FROM    cte_MinAverageValueState;
```

Answer to Puzzle #22

Occurrences

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#ProcessLog','U') IS NOT NULL
  DROP TABLE #ProcessLog;
GO

CREATE TABLE #ProcessLog
(
Workflow    VARCHAR(MAX),
Occurrences INTEGER,
LogMessage  VARCHAR(MAX)
);
GO

INSERT INTO #ProcessLog VALUES
('Alpha',5,'Error: Conversion Failed'),
('Alpha',8,'Status Complete'),
('Alpha',9,'Error: Unidentified error occurred'),
('Bravo',3,'Error: Cannot Divide by 0'),
('Bravo',1,'Error: Unidentified error occurred'),
('Charlie',10,'Error: Unidentified error occurred'),
('Charlie',7,'Error: Conversion Failed'),
('Charlie',6,'Status Complete');
GO

--MAX function
WITH cte_LogMessageCount AS
(
SELECT  LogMessage,
        MAX(Occurrences) AS MaxOccurrences
FROM    #ProcessLog
GROUP BY LogMessage
)
SELECT  a.Workflow, a.Occurrences, a.LogMessage
FROM    #ProcessLog a INNER JOIN
        cte_LogMessageCount b ON a.LogMessage = b.LogMessage AND
                                 a.Occurrences = b.MaxOccurrences
ORDER BY 1;

--Correlated Subquery with the ALL comparison operator
SELECT WorkFlow, Occurrences, LogMessage
FROM #ProcessLog AS e1
WHERE Occurrences > ALL(SELECT e2.Occurrences
                        FROM #ProcessLog AS e2
                        WHERE e2.LogMessage = e1.LogMessage AND
                              e2.WorkFlow <> e1.WorkFlow);
```

Answer to Puzzle #23

Divide in Half

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#PlayerScores','U') IS NOT NULL
  DROP TABLE #PlayerScores;
GO

CREATE TABLE #PlayerScores
(
PlayerID VARCHAR(MAX),
Score    INTEGER
);
GO

INSERT INTO #PlayerScores VALUES
(1001,2343),
(2002,9432),
(3003,6548),
(4004,1054),
(5005,6832);
GO

SELECT  NTILE(2) OVER (ORDER BY Score DESC) as Quartile,
        PlayerID,
        Score
FROM    #PlayerScores a
ORDER BY Score DESC;
```

Answer to Puzzle #24

Page Views

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#SampleData','U') IS NOT NULL
  DROP TABLE #SampleData;
GO

CREATE TABLE #SampleData
(
IntegerValue INTEGER IDENTITY(1,1),
RowID  UNIQUEIDENTIFIER
);
GO

ALTER TABLE #SampleData DROP COLUMN IntegerValue;
GO

INSERT INTO #SampleData VALUES (NEWID());
GO 1000

SELECT  RowID
FROM    #SampleData
ORDER BY RowID
OFFSET 10 ROWS FETCH NEXT 10 ROWS ONLY;
```

Answer to Puzzle #25

Top Vendors

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#Orders','U') IS NOT NULL
  DROP TABLE #Orders;
GO

CREATE TABLE #Orders
(
OrderID    VARCHAR(MAX),
CustomerID INTEGER,
OrderCount MONEY,
Vendor     VARCHAR(MAX)
);
GO

INSERT INTO #Orders VALUES
('Ord195342',1001,12,'Direct Parts'),
('Ord245532',1001,54,'Direct Parts'),
('Ord344394',1001,32,'ACME'),
('Ord442423',2002,7,'ACME'),
('Ord524232',2002,16,'ACME'),
('Ord645363',2002,5,'Direct Parts');
GO

WITH cte_Rank AS
(
SELECT  CustomerID,
        Vendor,
        RANK() OVER (PARTITION BY CustomerID ORDER BY COUNT(OrderCount) DESC) AS Rnk
FROM    #Orders
GROUP BY CustomerID, Vendor
)
SELECT  DISTINCT b.CustomerID, b.Vendor
FROM    #Orders a INNER JOIN
        cte_Rank b ON a.CustomerID = b.CustomerID AND a.Vendor = b.Vendor
WHERE   Rnk = 1;
```

Answer to Puzzle #26

Previous Years Sales

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#Sales','U') IS NOT NULL
  DROP TABLE #Sales;
GO

CREATE TABLE #Sales
(
Year   INTEGER,
Amount INTEGER
);
GO

INSERT INTO #Sales VALUES
(YEAR(GETDATE()),352645),
(YEAR(DATEADD(YEAR,-1,GETDATE())),165565),
(YEAR(DATEADD(YEAR,-1,GETDATE())),254654),
(YEAR(DATEADD(YEAR,-2,GETDATE())),159521),
(YEAR(DATEADD(YEAR,-2,GETDATE())),251696),
(YEAR(DATEADD(YEAR,-3,GETDATE())),111894);
GO

--PIVOT function
SELECT [2018],[2017],[2016] FROM #Sales
PIVOT (SUM(Amount) FOR Year IN ([2018],[2017],[2016])) AS PivotClause;

--LAG function
WITH cte_AggregateTotal AS
(
SELECT  Year,
        SUM(Amount) AS Amount
FROM    #Sales
GROUP BY Year
),
cte_Lag AS
(
SELECT  Year ,
        Amount,
        LAG(Amount,1,0) OVER (ORDER BY Year) AS Lag1,
        LAG(Amount,2,0) OVER (ORDER BY Year) AS Lag2
FROM    cte_AggregateTotal
)
SELECT  Amount AS '2018',
        Lag1 AS '2017',
        Lag2 AS '2016'
FROM    cte_Lag
WHERE   Year = 2018;
```

```
--Dynamic SQL without hardcoded dates
BEGIN

    DECLARE @CurrentYear VARCHAR(MAX) =
                CAST(YEAR(GETDATE()) AS VARCHAR);
    DECLARE @CurrentYearLag1 VARCHAR(MAX) =
                CAST(YEAR(DATEADD(YEAR,-1,GETDATE())) AS VARCHAR);
    DECLARE @CurrentYearLag2 VARCHAR(MAX) =
                CAST(YEAR(DATEADD(YEAR,-2,GETDATE())) AS VARCHAR);
    DECLARE @DynamicSQL NVARCHAR(MAX);

    SET @DynamicSQL =
    'SELECT [' + @CurrentYear + '],
            [' + @CurrentYearLag1 + '],
            [' + @CurrentYearLag2 + ']
    FROM #Sales
    PIVOT (SUM(AMOUNT) FOR YEAR IN (
            [' + @CurrentYear + '],
            [' + @CurrentYearLag1 + '],
            [' + @CurrentYearLag2 + ''])) AS PivotClause;'

    PRINT @DynamicSQL;
    EXECUTE SP_EXECUTESQL @DynamicSQL;

END;
```

Answer to Puzzle #27

Delete the Duplicates

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#SampleData','U') IS NOT NULL
  DROP TABLE #SampleData;
GO

CREATE TABLE #SampleData
(
IntegerValue INTEGER,
);
GO

INSERT INTO #SampleData VALUES
(1),
(1),
(2),
(3),
(3),
(4);
GO

WITH cte_Duplicates AS
(
SELECT  ROW_NUMBER() OVER (PARTITION BY IntegerValue ORDER BY IntegerValue) AS Rnk
FROM    #SampleData
)
DELETE FROM cte_Duplicates WHERE Rnk > 1
GO
```

Answer to Puzzle #28

Fill the Gaps

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#Gaps','U') IS NOT NULL
  DROP TABLE #Gaps;
GO

CREATE TABLE #Gaps
(
RowNumber INTEGER,
TestCase  VARCHAR(MAX)
);
GO

INSERT INTO #Gaps VALUES
(1,'Alpha'),
(2,NULL),
(3,NULL),
(4,NULL),
(5,'Bravo'),
(6,NULL),
(7,'Charlie'),
(8,NULL),
(9,NULL);
GO

--Solution 1
SELECT  c.RowNumber,
        (SELECT  d.TestCase
         FROM    #Gaps d
         WHERE   d. RowNumber =
                     (SELECT MAX(e. RowNumber)
                     FROM #Gaps e
                     WHERE e. RowNumber <= c. RowNumber AND e.TestCase != '')) TestCase
FROM #Gaps c;

--Solution 2
BEGIN
    DECLARE @v VARCHAR(MAX);
    UPDATE #Gaps WITH(TABLOCKX)
    SET @v = TestCase = CASE WHEN TestCase IS NULL THEN @v ELSE TestCase END
    OPTION(MAXDOP 1);
    SELECT RowNumber, TestCase FROM #Gaps;
END

--Solution 3
SELECT RowNumber, MAX(TestCase) OVER (PARTITION BY DistinctCount) AS TestCase
FROM   (SELECT  RowNumber,
                TestCase,
                COUNT(TestCase) OVER (ORDER BY RowNumber) AS DistinctCount
        FROM #Gaps) a
ORDER BY RowNumber;
```

Answer to Puzzle #29

Count the Groupings

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#Groupings','U') IS NOT NULL
  DROP TABLE #Groupings;
GO

CREATE TABLE #Groupings
(
StepNumber INTEGER,
TestCase   VARCHAR(MAX),
Status     VARCHAR(MAX)
);
GO

INSERT INTO #Groupings VALUES
(1,'Test Case 1','Passed'),
(2,'Test Case 2','Passed'),
(3,'Test Case 3','Passed'),
(4,'Test Case 4','Passed'),
(5,'Test Case 5','Failed'),
(6,'Test Case 6','Failed'),
(7,'Test Case 7','Failed'),
(8,'Test Case 8','Failed'),
(9,'Test Case 9','Failed'),
(10,'Test Case 10','Passed'),
(11,'Test Case 11','Passed'),
(12,'Test Case 12','Passed');
GO

WITH cte_RowNumber AS
(
SELECT  Status,
        StepNumber,
        ROW_NUMBER() OVER (PARTITION BY Status ORDER BY StepNumber) AS RowNumber,
        StepNumber - ROW_NUMBER() OVER (PARTITION BY Status ORDER BY StepNumber) AS Rnk
FROM    #Groupings
)
SELECT  ROW_NUMBER() OVER (ORDER BY Rnk) AS StepOrder,
        Status,
        MAX(StepNumber) - MIN(StepNumber) + 1 AS ConsecutiveCount
FROM    cte_RowNumber
GROUP BY Rnk, Status
ORDER BY Rnk, Status;
```

Answer to Puzzle #30

Select Star

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#Products','U') IS NOT NULL
  DROP TABLE #Products;
GO

CREATE TABLE #Products
(
ProductID   INTEGER,
ProductName VARCHAR(MAX)
);
GO

--Add the following constraint
ALTER TABLE #Products ADD ComputedColumn AS (0/0);
```

Answer to Puzzle #31

Second Highest

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#SampleData','U') IS NOT NULL
  DROP TABLE #SampleData;
GO

CREATE TABLE #SampleData
(
IntegerValue INTEGER
);
GO

INSERT INTO #SampleData VALUES
(3759),(3760),(3761),(#37
#),(3763);
GO

--Solution 1
SELECT  IntegerValue
FROM    #SampleData a
WHERE   2 = (SELECT COUNT(IntegerValue)
             FROM    #SampleData b
             WHERE   a.IntegerValue <= b.IntegerValue);

--Solution 2
SELECT  IntegerValue
FROM    #SampleData a
ORDER BY IntegerValue DESC
OFFSET 1 ROWS FETCH NEXT 1 ROWS ONLY;

--Solution 3
SELECT  MAX(IntegerValue)
FROM    #SampleData
WHERE   IntegerValue < (SELECT MAX(IntegerValue) FROM #SampleData);

--Solution 4
WITH cte_Top2 AS
(
SELECT  TOP(2) IntegerValue
FROM    #SampleData
ORDER BY IntegerValue DESC
)
SELECT  MIN(IntegerValue) FROM cte_Top2;
```

Answer to Puzzle #32

First and Last

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#Personel','U') IS NOT NULL
  DROP TABLE #Personel;
GO

CREATE TABLE #Personel
(
SpacemanID     VARCHAR(MAX),
JobDescription VARCHAR(MAX),
MissionCount   INTEGER
);
GO

INSERT INTO #Personel VALUES
(1001,'Astrogator',6),
(2002,'Astrogator',12),
(3003,'Astrogator',17),
(4004,'Geologist',21),
(5005,'Geologist',9),
(6006,'Geologist',8),
(7007,'Technician',13),
(8008,'Technician',2),
(9009,'Technician',7);
GO

SELECT  DISTINCT
        JobDescription,
        FIRST_VALUE(SpacemanID) OVER
           (PARTITION  BY JobDescription ORDER BY MissionCount DESC) AS MostExperienced,
        LAST_VALUE(SpacemanID)  OVER
           (PARTITION  BY JobDescription ORDER BY MissionCount DESC
            RANGE BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING) AS LeastExperienced
FROM    #Personel
ORDER BY 1,2,3;
```

Answer to Puzzle #33

Deadlines

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#OrderFullfillment','U') IS NOT NULL
  DROP TABLE #OrderFullfillment;
GO

IF OBJECT_ID('tempdb.dbo.#ManufactoringTime','U') IS NOT NULL
  DROP TABLE #ManufactoringTime;
GO

CREATE TABLE #OrderFullfillment
(
OrderID     VARCHAR(MAX),
ProductID   VARCHAR(MAX),
DaysToBuild INTEGER
);
GO

CREATE TABLE #ManufactoringTime
(
PartID            VARCHAR(MAX),
ProductID         VARCHAR(MAX),
DaysToManufacture INTEGER
);
GO

INSERT INTO #OrderFullfillment VALUES
('Ord893456','Widget',7),
('Ord923654','Gizmo',3),
('Ord187239','Doodad',9);
GO

INSERT INTO #ManufactoringTime VALUES
 ('AA-111','Widget',7),
('BB-222','Widget',2),
('CC-333','Widget',3),
('DD-444','Widget',1),
('AA-111','Gizmo',7),
('BB-222','Gizmo',2),
('AA-111','Doodad',7),
('DD-444','Doodad',1);
GO

SELECT  OrderID, ProductID, DaysToBuild
FROM    #OrderFullfillment a
WHERE   DaysToBuild >= ALL(SELECT DaysToManufacture
                           FROM   #ManufactoringTime b
                           WHERE   a.ProductID = b.ProductID);
```

Answer to Puzzle #34

Specific Exclusion

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#Orders','U') IS NOT NULL
  DROP TABLE #Orders;
GO

CREATE TABLE #Orders
(
CustomerID INTEGER,
OrderID    VARCHAR(MAX),
Amount     MONEY
);
GO

INSERT INTO #Orders VALUES
(1001,'Ord143937',25),
(1001,'Ord789765',50),
(2002,'Ord345434',65),
(3003,'Ord465633',50);
GO

SELECT  CustomerID, OrderID, Amount
FROM    #Orders
WHERE   NOT(CustomerID = 1001 AND OrderID = 'Ord789765');
```

Answer to Puzzle #35

International vs Domestic Sales

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#Orders','U') IS NOT NULL
  DROP TABLE #Orders;
GO

CREATE TABLE #Orders
(
SalesRepID INTEGER,
InvoiceID  VARCHAR(MAX),
Amount     MONEY,
SalesType  VARCHAR(MAX)
);
GO

INSERT INTO #Orders VALUES
(1001,'Inv345756',13454,'International'),
(2002,'Inv546744',3434,'International'),
(4004,'Inv234745',54645,'International'),
(5005,'Inv895745',234345,'International'),
(7007,'Inv006321',776,'International'),
(1001,'Inv734534',4564,'Domestic'),
(2002,'Inv600213',34534,'Domestic'),
(3003,'Inv757853',345,'Domestic'),
(6006,'Inv198632',6543,'Domestic'),
(8008,'Inv977654',67,'Domestic');
GO

WITH cte_Domestic AS
(
SELECT  SalesRepID, InvoiceID
FROM    #Orders
WHERE   SalesType = 'Domestic'
),
cte_International AS
(
SELECT  SalesRepID, InvoiceID
FROM    #Orders
WHERE   SalesType = 'International'
)
SELECT  ISNULL(a.SalesRepID,b.SalesRepID)
FROM    cte_Domestic a FULL OUTER JOIN
        cte_International b ON a.SalesRepID = b.SalesRepID
WHERE   a.InvoiceID IS NULL OR b.InvoiceID IS NULL;
```

Answer to Puzzle #36

Traveling Salesman

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#Graph','U') IS NOT NULL
  DROP TABLE #Graph;
GO

CREATE TABLE #Graph
(
DepartureCity VARCHAR(MAX),
ArrivalCity   VARCHAR(MAX),
Cost          INTEGER
);
GO

INSERT INTO #Graph VALUES
('Austin','Dallas',100),
('Dallas','Austin',150),
('Dallas','Memphis',200),
('Memphis','Des Moines',300),
('Dallas','Des Moines',400);
GO

--Making the assumption the maximum number of layovers is four
WITH cte_Graph AS
(
SELECT DepartureCity, ArrivalCity, Cost FROM #Graph
UNION ALL
SELECT ArrivalCity, DepartureCity, Cost FROM #Graph
UNION ALL
SELECT ArrivalCity, ArrivalCity, 0 FROM #Graph
UNION ALL
SELECT DepartureCity, DepartureCity, 0 FROM #Graph
)
SELECT  DISTINCT
        g1.DepartureCity,
        g2.DepartureCity,
        g3.DepartureCity,
        g4.DepartureCity,
        g4.ArrivalCity,
        (g1.Cost + g2.Cost + g3.Cost + g4.Cost) AS TotalCost
FROM    cte_Graph AS g1 INNER JOIN
        cte_Graph AS g2 ON g1.ArrivalCity = g2.DepartureCity INNER JOIN
        cte_Graph AS g3 ON g2.ArrivalCity = g3.DepartureCity INNER JOIN
        cte_Graph AS g4 ON g3.ArrivalCity = g4.DepartureCity
WHERE   g1.DepartureCity = 'Austin' AND
        g4.ArrivalCity = 'Des Moines'
ORDER BY 6,1,2,3,4;
```

Answer to Puzzle #37

Group Criteria Keys

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#GroupCriteria','U') IS NOT NULL
  DROP TABLE #GroupCriteria;
GO

CREATE TABLE #GroupCriteria
(
OrderID     VARCHAR(MAX),
Distributor VARCHAR(MAX),
Facility    INTEGER,
Zone        VARCHAR(MAX),
Amount      MONEY
);
GO

INSERT INTO #GroupCriteria VALUES
('Ord156795','ACME',123,'ABC',100),
('Ord826109','ACME',123,'ABC',75),
('Ord342876','Direct Parts',789,'XYZ',150),
('Ord994981','Direct Parts',789,'XYZ',125);
GO

SELECT DENSE_RANK() OVER (ORDER BY Distributor, Facility, Zone) as CriteriaID,
       OrderID,
       Distributor,
       Facility,
       Zone,
       Amount
FROM   #GroupCriteria;
```

Answer to Puzzle #38

Reporting Elements

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#RegionSales','U') IS NOT NULL
  DROP TABLE #RegionSales;
GO

CREATE TABLE #RegionSales
(
Region       VARCHAR(MAX),
Distributor VARCHAR(MAX),
Sales        INTEGER
);
GO

INSERT INTO #RegionSales VALUES
('North','ACE',10),
('South','ACE',67),
('East','ACE',54),
('North','Direct Parts',8),
('South','Direct Parts',7),
('West','Direct Parts',12),
('North','ACME',65),
('South','ACME',9),
('West','ACME',1),
('West','ACME',7);
GO

WITH cte_DistinctRegion AS
(
SELECT DISTINCT Region
FROM   #RegionSales
),
cte_DistinctDistributor AS
(
SELECT DISTINCT Distributor
FROM   #RegionSales
),
cte_CrossJoin AS
(
SELECT Region, Distributor
FROM   cte_DistinctRegion a CROSS JOIN
             cte_DistinctDistributor b
)
SELECT a.Region, a.Distributor, ISNULL(b.Sales,0) AS Sales
FROM   cte_CrossJoin a LEFT OUTER JOIN
             #RegionSales b ON a.Region = b.Region and a.Distributor = b.Distributor
ORDER BY a.Distributor,
         (CASE a.Region WHEN 'North' THEN 1
                        WHEN 'South' THEN 2
                        WHEN 'East'  THEN 3
                        WHEN 'West'  THEN 4 END);
```

Answer to Puzzle #39

Prime Numbers

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#SampleData ','U') IS NOT NULL
  DROP TABLE #SampleData;
GO

CREATE TABLE #SampleData
(
IntegerValue INTEGER
);
GO

INSERT INTO #SampleData VALUES
(1),(2),(3),(4),(5),(6),(7),(8),(9),(10);
GOs

SELECT IntegerValue,
       IntegerValue%2,
       IIF(IntegerValue%2 > 0 OR IntegerValue <= 2,'Prime Number',NULL)
FROM   #SampleData;
```

Answer to Puzzle #40

Sort Order

Hyperlink to the puzzle

```
IF OBJECT_ID('tempdb.dbo.#SortOrder','U') IS NOT NULL
  DROP TABLE #SortOrder;
GO

CREATE TABLE #SortOrder
(
City VARCHAR(MAX)
);
GO

INSERT INTO #SortOrder VALUES
('Atlanta'),('Baltimore'),('Chicago'),('Denver');
GO

SELECT City
FROM   #SortOrder
ORDER BY (CASE City WHEN  'Atlanta' THEN 2 WHEN 'Baltimore' THEN 1 WHEN 'Chicago' THEN 4
WHEN 'Denver' THEN 1 END);
```