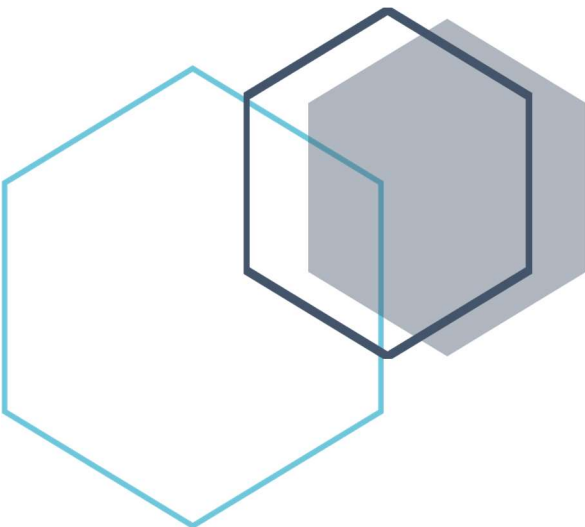




Advanced SQL Puzzles

Scott Peters

<https://advancedsqlpuzzles.com/>



Last Updated: 01/01/2023

Welcome

I hope you enjoy these puzzles as much as I have enjoyed creating them!

As my list of puzzles continues to grow, I have decided to combine the puzzles into one single document broken down into two different sections.

In the first section I have 58 of the most difficult puzzles that I could create; randomly organized and in no specific order. These are mostly set-based puzzles, interspersed with a small number of puzzles that require knowledge of constraints, specific data types, cursors, loops, etc....

Working through these puzzles will give you an understanding of the SQL language and what types of problems the SQL language best solves. Remember that SQL is a declarative and not an imperative language, and always think in sets when providing a solution.

In the second set of puzzles, I have collected all the puzzles related to permutations, combinations, and sequences. Solving these puzzles will require a deeper knowledge of your SQL thinking, focusing on such constructs as using recursion or sequence objects to reach the desired output (and of course some will require using traditional set-based thinking). Ultimately these puzzles resolve to creating number tables, which can be used to fill in gaps, create ranges and tallies, provide custom sorting, and allow you to create set based solutions over iterative solutions. I also included a few puzzles from part 1 into this set as ultimately they deal with creating a numbers table

I hope navigating through the GitHub repository to find the solutions is straightforward. The first set of puzzles are combined into one single SQL document, and the second set has a separate folder with individual solutions, as these solutions are a little more involved in solving. For my own sanity, it is easiest to not embed the SQL solutions into this text document and instead provide them separately as SQL files in the GitHub repository. If you have any issues navigating the website or the GitHub, please contact me and I would be happy to help.

Answers to these puzzles are located in the following GitHub repository:

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

I welcome any corrections, new tricks, new techniques, dead links, misspellings, bugs, and especially any new puzzles that would be a great fit for this document.

Please contact me through the contact page on my website or use the discussion board in the GitHub repository.

<https://advancedsqlpuzzles.com/>

Happy coding!

PART I

Thinking in Sets

Puzzle #1

Shopping Carts

You are tasked with providing an audit of two shopping carts.

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

| Item | Item |
|-------|--------|
| Sugar | Sugar |
| Bread | Bread |
| Juice | Butter |
| Soda | Cheese |
| Flour | Fruit |

Here is the expected output.

| Item Cart 1 | Item Cart 2 |
|-------------|-------------|
| Sugar | Sugar |
| Bread | Bread |
| Juice | |
| Soda | |
| Flour | |
| | Butter |
| | Cheese |
| | Fruit |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

Puzzle #2**Managers and Employees**

Given the following hierarchical 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 | | 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 |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/Advanced-SQL-Puzzles)

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
);
```

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

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.

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/Advanced-SQL-Puzzles)

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 | | |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/Advanced-SQL-Puzzles)

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?

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/Advanced-SQL-Puzzles)

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 all the requirements of the mission.

Candidates

| Candidate ID | Description |
|--------------|--------------|
| 1001 | Geologist |
| 1001 | Astrogator |
| 1001 | Biochemist |
| 1001 | Technician |
| 2002 | Surgeon |
| 2002 | Machinist |
| 2002 | Geologist |
| 3003 | Geologist |
| 3003 | Astrogator |
| 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, and Candidate ID 4004 has none of the needed requirements.

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

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 |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

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.

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

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 largest and smallest 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
);

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

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

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. Ensure your code can account for a changing number of elements without rewriting.

| 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 |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/Advanced-SQL-Puzzles)

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 |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/Advanced-SQL-Puzzles)

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 |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

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 steps of a workflow are of the same status (Error, Complete or Running), then return the distinct status.
- If any steps of a workflow have an Error status along with a status of Complete or Running, set the overall status to Indeterminate.
- If the workflow steps have a combination of Complete and Running (without any Errors), set the overall status to Running.

| Workflow | Step Number | Status |
|----------|-------------|----------|
| Alpha | 1 | Error |
| Alpha | 2 | Complete |
| Alpha | 3 | Running |
| Bravo | 1 | Complete |
| Bravo | 2 | Complete |
| Charlie | 1 | Running |
| Charlie | 2 | Running |
| Delta | 1 | Error |
| Delta | 2 | Error |
| Echo | 1 | Running |
| Echo | 2 | Complete |

Here is the expected output.

| Workflow | Status |
|----------|---------------|
| Alpha | Indeterminate |
| Bravo | Complete |
| Charlie | Running |
| Delta | Error |
| Echo | Running |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/Advanced-SQL-Puzzles)

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 |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

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 |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/Advanced-SQL-Puzzles)

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 |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/Advanced-SQL-Puzzles)

Puzzle #18**Seating Chart**

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

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

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

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 |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

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 |
|------------|-----------|
| 1/15/2018 | 1/19/2018 |
| 1/12/2018 | 1/16/2018 |
| 1/10/2018 | 1/11/2018 |
| 1/3/2018 | 1/9/2018 |
| 1/1/2018 | 1/5/2018 |

Here is the expected output.

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

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

Puzzle #20**Price Points**

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

| Product ID | Effective Date | Unit Price |
|------------|----------------|------------|
| 1001 | 1/1/2018 | \$1.99 |
| 1001 | 4/15/2018 | \$2.99 |
| 1001 | 6/8/2018 | \$3.99 |
| 2002 | 4/17/2018 | \$1.99 |
| 2002 | 5/19/2018 | \$2.99 |

Here is the expected output.

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

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

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 4004 did not have an average monthly value over \$100 in May 2018.

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

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

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 |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

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, and the remaining players must be given a value of 2.

Write an SQL statement that meets these requirements.

| Player ID | Score |
|-----------|-------|
| 1001 | 2343 |
| 2002 | 9432 |
| 3003 | 6548 |
| 4004 | 1054 |
| 5005 | 6832 |

Here is the expected output.

| Quartile | Player ID | Score |
|----------|-----------|-------|
| 1 | 2002 | 9432 |
| 1 | 3003 | 6548 |
| 1 | 5005 | 6832 |
| 2 | 1001 | 2343 |
| 2 | 4004 | 1054 |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/Advanced-SQL-Puzzles)

Puzzle #24**Page Views**

Write an SQL statement that retrieves records 5 to 10 ordered by the Order ID column.

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

Here is the expected output.

| Order ID | Customer ID | Order Date | Amount | State |
|-----------|-------------|------------|--------|-------|
| Ord654374 | 1001 | 2/1/2018 | \$100 | TX |
| Ord654525 | 4004 | 5/1/2018 | \$50 | IA |
| Ord657895 | 1001 | 1/1/2018 | \$150 | TX |
| Ord856993 | 3003 | 3/1/2018 | \$100 | IA |
| Ord864573 | 4004 | 4/1/2018 | \$100 | IA |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://advancedsqlpuzzles.com/)

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 |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

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 |

Answers to the puzzles are located in the following GitHub repository.
[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

Puzzle #27

Delete the Duplicates

Write an SQL statement that deletes the duplicate data.

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

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

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

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 |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

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.

| Min Step Number | Max Step Number | Status | Consecutive Count |
|-----------------|-----------------|--------|-------------------|
| 1 | 4 | Passed | 4 |
| 5 | 9 | Failed | 5 |
| 10 | 12 | Passed | 3 |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/Advanced-SQL-Puzzles)

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)
);
```

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

Puzzle #31

Second Highest

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

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

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

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

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 |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

Puzzle #33**Deadlines**

How many different SQL statement can you write that determines if an order will be fulfilled by the requested delivery date?

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 would be in the output as these orders have a promised delivery date that is equal to or greater than the days to manufacture.

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

Puzzle #34**Specific Exclusion**

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

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

Here is the expected output.

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

Answers to the puzzles are located in the following GitHub repository.
[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

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.

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

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

Puzzle #36**Traveling Salesman**

Here is a well-known problem that is called the [Traveling Salesman](#).

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?

Note the data represents a cyclic graph.

| Route ID | Departure City | Arrival City | Cost |
|----------|----------------|--------------|-------|
| 1 | Austin | Dallas | \$100 |
| 1 | Dallas | Austin | \$100 |
| 2 | Dallas | Memphis | \$200 |
| 2 | Memphis | Dallas | \$200 |
| 3 | Memphis | Des Moines | \$300 |
| 3 | Des Moines | Memphis | \$300 |
| 4 | Dallas | Des Moines | \$400 |
| 4 | Des Moines | Dallas | \$400 |

Here is the expected output.

| Route Path | Total Cost |
|---|------------|
| Austin -->Dallas -->Des Moines | \$500 |
| Austin -->Dallas -->Memphis -->Des Moines | \$600 |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](#)

Puzzle #37**Group Criteria Keys**

Write an SQL statement that provides a key based upon the distinct combination of the columns 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 |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

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 value 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 | ACME | 65 |
| South | ACME | 9 |
| East | ACME | 1 |
| West | ACME | 7 |
| North | Direct Parts | 8 |
| South | Direct Parts | 7 |
| West | Direct Parts | 12 |

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 |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

Puzzle #39

Prime Numbers

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

```
CREATE TABLE #PrimeNumbers
(
  IntegerValue INTEGER
);

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

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

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 |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

Puzzle #41**Associate IDs**

The following table shows two hierarchical structures.

1. The first is the association between Anne, Betty, Charles, Dan, and Emma.
2. The second is the association between Francis, George, and Harriet.

Write an SQL statement that creates a grouping number for each hierarchical association and display the member in the associations.

| Associate 1 | Associate 2 |
|-------------|-------------|
| Anne | Betty |
| Anne | Charles |
| Betty | Dan |
| Charles | Emma |
| Francis | George |
| George | Harriet |

Here is the expected output.

| Grouping | Associate |
|----------|-----------|
| 1 | Anne |
| 1 | Betty |
| 1 | Charles |
| 1 | Dan |
| 1 | Emma |
| 2 | Francis |
| 2 | George |
| 2 | Harriet |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

Puzzle #42**Mutual Friends**

The following table shows a cyclic data structure.

Given the following list of friend connections, determine the number of mutual connections between the friends.

| Friend 1 | Friend 2 |
|----------|----------|
| Jason | Mary |
| Mike | Mary |
| Mike | Jason |
| Susan | Jason |
| John | Mary |
| Susan | Mary |

Here is the expected output.

| Friend 1 | Friend 2 | Mutual Friends |
|----------|----------|----------------|
| Jason | Mary | 2 |
| John | Mary | 0 |
| Jason | Mike | 1 |
| Mary | Mike | 1 |
| Jason | Susan | 1 |
| Mary | Susan | 1 |

Jason and Mary have 2 mutual friends: Mike and Susan.

John and Mary have 0 mutual friends.

Jason and Mike have 1 mutual friend: Mary.

etc.....

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://advancedsqlpuzzles.com/)

Puzzle #43

Unbounded Preceding

Determine the minimum quantity for each record between the current row and all previous rows for each Customer ID.

| Order | Customer ID | Quantity |
|-------|-------------|----------|
| 1 | 1001 | 5 |
| 2 | 1001 | 8 |
| 3 | 1001 | 3 |
| 4 | 1001 | 7 |
| 1 | 2002 | 4 |
| 2 | 2002 | 9 |

Here is the expected output.

| Order | Customer ID | Quantity | Min Value |
|-------|-------------|----------|-----------|
| 1 | 1001 | 5 | 5 |
| 2 | 1001 | 8 | 5 |
| 3 | 1001 | 3 | 3 |
| 4 | 1001 | 7 | 3 |
| 1 | 2002 | 4 | 4 |
| 2 | 2002 | 9 | 4 |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

Puzzle #44**Slowly Changing Dimension Part I**

Give the following table, write an SQL statement to create a Type 2 Slowly Changing Dimension.

| Customer ID | Balance Date | Amount |
|-------------|--------------|---------|
| 1001 | 10/11/2021 | \$54.32 |
| 1001 | 10/10/2021 | \$17.65 |
| 1001 | 9/18/2021 | \$65.56 |
| 1001 | 9/12/2021 | \$56.23 |
| 1001 | 9/1/2021 | \$42.12 |
| 2002 | 10/15/2021 | \$46.52 |
| 2002 | 10/13/2021 | \$7.65 |
| 2002 | 9/15/2021 | \$75.12 |
| 2002 | 9/10/2021 | \$47.34 |
| 2002 | 9/2/2021 | \$11.11 |

Here is the expected output.

| Customer ID | Start Date | End Date | Amount |
|-------------|------------|------------|---------|
| 1001 | 10/11/2021 | 12/31/9999 | \$54.32 |
| 1001 | 10/10/2021 | 10/10/2021 | \$17.65 |
| 1001 | 9/18/2021 | 10/9/2021 | \$65.56 |
| 1001 | 9/12/2021 | 9/17/2021 | \$56.23 |
| 1001 | 9/1/2021 | 9/11/2021 | \$42.12 |
| 2002 | 10/15/2021 | 12/31/9999 | \$46.52 |
| 2002 | 10/13/2021 | 10/14/2021 | \$7.65 |
| 2002 | 9/15/2021 | 10/12/2021 | \$75.12 |
| 2002 | 9/10/2021 | 9/14/2021 | \$47.34 |
| 2002 | 9/2/2021 | 9/9/2021 | \$11.11 |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

Puzzle #45**Slowly Changing Dimension Part II**

Given the following table with overlapping timeframes. Write an SQL statement to identify the overlapping records.

| Customer ID | Start Date | End Date | Amount |
|-------------|------------|------------|---------|
| 1001 | 10/11/2021 | 12/31/9999 | \$54.32 |
| 1001 | 10/10/2021 | 10/10/2021 | \$17.65 |
| 1001 | 9/18/2021 | 10/12/2021 | \$65.56 |
| 2002 | 9/12/2021 | 9/17/2021 | \$56.23 |
| 2002 | 9/1/2021 | 9/17/2021 | \$42.12 |
| 2002 | 8/15/2021 | 8/31/2021 | \$16.32 |

Here is the expected output.

| Customer ID | Start Date | End Date | Amount |
|-------------|------------|------------|---------|
| 1001 | 9/18/2021 | 10/12/2021 | \$65.56 |
| 2002 | 9/1/2021 | 9/17/2021 | \$42.12 |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/Advanced-SQL-Puzzles)

Puzzle #46

Negative Account Balances

How many different SQL statements can you write to determine all accounts whose balance has never been positive?

| Account ID | Balance |
|------------|-----------|
| 1001 | \$234.45 |
| 1001 | \$-23.12 |
| 2002 | \$-93.01 |
| 2002 | \$-120.19 |
| 3003 | \$186.76 |
| 3003 | \$90.23 |
| 3003 | \$10.11 |

Account ID 2002 would appear in the result set as this account has never had a positive balance. There are a multitude of ways to write this statement, can you think of them all?

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

Puzzle #47**Work Schedule**

Given a table of employee shifts, and another table of their activities, merge the two tables and write an SQL script that produces the desired output. If an employee is scheduled and does not have an activity planned, label the time frame as “Work”.

Schedule

| Schedule ID | Start Time | End Time |
|-------------|-----------------|-----------------|
| A | 10/1/2021 10:00 | 10/1/2021 15:00 |
| B | 10/1/2021 10:15 | 10/1/2021 12:15 |

Activity

| Schedule ID | Activity | Start Time | End Time |
|-------------|----------|-----------------|-----------------|
| A | Meeting | 10/1/2021 10:00 | 10/1/2021 10:30 |
| A | Break | 10/1/2021 12:00 | 10/1/2021 12:30 |
| A | Meeting | 10/1/2021 13:00 | 10/1/2021 13:30 |
| B | Break | 10/1/2021 11:00 | 10/1/2021 11:15 |

Here is the expected output.

| Schedule ID | Activity | Start Time | End Time |
|-------------|----------|-----------------|-----------------|
| A | Meeting | 10/1/2021 10:00 | 10/1/2021 10:30 |
| A | Work | 10/1/2021 10:30 | 10/1/2021 12:00 |
| A | Break | 10/1/2021 12:00 | 10/1/2021 12:30 |
| A | Work | 10/1/2021 12:30 | 10/1/2021 13:00 |
| A | Meeting | 10/1/2021 13:00 | 10/1/2021 13:30 |
| A | Work | 10/1/2021 13:30 | 10/1/2021 15:00 |
| B | Work | 10/1/2021 10:15 | 10/1/2021 11:00 |
| B | Break | 10/1/2021 11:00 | 10/1/2021 11:15 |
| B | Work | 10/1/2021 11:15 | 10/1/2021 12:15 |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

Puzzle #48

Consecutive Sales

For the following Customer IDs, write an SQL statement to determine customers that had a sale in the current year, plus the previous two consecutive years.

You will need to adjust the test data for the current year, as the test data is coded for the year 2021.

| Sales ID | Year |
|----------|------|
| 1001 | 2018 |
| 1001 | 2019 |
| 1001 | 2020 |
| 2002 | 2020 |
| 2002 | 2021 |
| 3003 | 2018 |
| 3003 | 2020 |
| 3003 | 2021 |
| 4004 | 2019 |
| 4004 | 2020 |
| 4004 | 2021 |

Sales ID 4004 would be in the expected output as this customer had a sale in the current year, plus the previous two years.

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

Puzzle #49**Sumo Wrestlers**

A group of Sumo wrestlers are forming a line to board an elevator. Unfortunately, the elevator can only hold 2,000 pounds and not all Sumo wrestlers can board. Which Sumo wrestler would be the last to enter given the following queue order?

| Order | Name | Weight |
|-------|--------|--------|
| 1 | Haruto | 611 |
| 2 | Minato | 533 |
| 3 | Haruki | 623 |
| 4 | Sota | 569 |
| 5 | Aoto | 610 |
| 6 | Hinata | 525 |

The expected output would be Haruki, as this is the last Sumo wrestler to fit in the elevator before the 2,000-pound maximum capacity is reached.

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

Puzzle #50**Baseball Balls and Strikes**

For this puzzle, you will need to understand the rules of baseball's balls and strike count.

Given a table of results for each pitch for the following at-bats. Construct an SQL statement that returns the columns Start Of Pitch Count and End Of Pitch Count.

| Batter ID | Pitch Number | Result | Start Of Pitch Count | End Of Pitch Count |
|-----------|--------------|---------|----------------------|--------------------|
| 1001 | 1 | Foul | 0 – 0 | 0 – 1 |
| 1001 | 2 | Foul | 0 – 1 | 0 – 2 |
| 1001 | 3 | Ball | 0 – 2 | 1 – 2 |
| 1001 | 4 | Ball | 1 – 2 | 2 – 2 |
| 1001 | 5 | Strike | 2 – 2 | 2 – 3 |
| 2002 | 1 | Ball | 0 – 0 | 1 – 0 |
| 2002 | 2 | Strike | 1 – 0 | 1 – 1 |
| 2002 | 3 | Foul | 1 – 1 | 1 – 2 |
| 2002 | 4 | Foul | 1 – 2 | 1 – 2 |
| 2002 | 5 | Foul | 1 – 2 | 1 – 2 |
| 2002 | 6 | In Play | 1 – 2 | In Play |
| 3003 | 1 | Ball | 0 – 0 | 1 – 0 |
| 3003 | 2 | Ball | 1 – 0 | 2 – 0 |
| 3003 | 3 | Ball | 2 – 0 | 3 – 0 |
| 3003 | 4 | Ball | 3 – 0 | 4 – 0 |
| 4004 | 1 | Foul | 0 – 0 | 0 – 1 |
| 4004 | 2 | Foul | 0 – 1 | 0 – 2 |
| 4004 | 3 | Foul | 0 – 2 | 0 – 2 |
| 4004 | 4 | Foul | 0 – 2 | 0 – 2 |
| 4004 | 5 | Foul | 0 – 2 | 0 – 2 |
| 4004 | 6 | Strike | 0 – 2 | 0 – 3 |

The expected output would be the columns Start of Pitch Count and End of Pitch Count given each pitch result.

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

Puzzle #51

Primary Key Creation

Given the following table whose natural key is a combination of the columns Assembly ID and Part, use the HASHBYTES and CHECKSUM functions to create two new fields that can be used as primary keys.

The goal here is to create a single field that is unique and re-creatable. The benefit of creating a hashbytes or checksum column is to aid in data profiling and integrity checks when a table contains a multitude of columns that form the natural key (and some of these columns can be NULL).

| Assembly ID | Part |
|-------------|--------|
| 1001 | Bolt |
| 1001 | Screw |
| 2002 | Nut |
| 2002 | Washer |
| 3003 | Toggle |
| 3003 | Bolt |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

Puzzle #52

Phone Numbers Table

You are creating a table that customer agents will use to enter customer information and their phone numbers.

Create a table with the fields Customer ID and Phone Number, where the Phone Number field must be inserted with the format (999)-999-9999.

Agents will enter phone numbers into this table via a form, and it is imperative that phone numbers are formatted correctly when inputted. Create a table that meets these requirements.

Here are a few sample records.

| Customer ID | Phone Number |
|-------------|----------------|
| 1001 | (555)-555-5555 |
| 2002 | (555)-555-5555 |
| 3003 | (555)-555-5555 |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

Puzzle #53**Spouse IDs**

You are given the following table of individuals and their spouse. Every individual exists both as a Primary ID and a Spouse ID. You need to create a group criteria key to match the associated records

| Primary ID | Spouse ID |
|------------|-----------|
| Pat | Charlie |
| Jordan | Casey |
| Ashley | Dee |
| Charlie | Pat |
| Casey | Jordan |
| Dee | Ashley |

Here is the expected output

| Group ID | Primary ID | Spouse ID |
|----------|------------|-----------|
| 1 | Ashley | Dee |
| 1 | Dee | Ashley |
| 2 | Jordan | Casey |
| 2 | Casey | Jordan |
| 3 | Charlie | Pat |
| 3 | Pat | Charlie |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://advancedsqlpuzzles.com/)

Puzzle #54

Winning the Lottery

You are part of an office lottery pool where you keep a table of the winning lottery numbers along with a table of each ticket's chosen numbers. If a ticket has some but not all the winning numbers, you win \$10. If a ticket has all the winning numbers, you win \$100. Calculate the total winnings for today's drawing.

Winning Numbers

| Number |
|--------|
| 25 |
| 45 |
| 78 |

Tickets

| Ticket ID | Number |
|-----------|--------|
| A23423 | 25 |
| A23423 | 45 |
| A23423 | 78 |
| B35643 | 25 |
| B35643 | 45 |
| B35643 | 98 |
| C98787 | 67 |
| C98787 | 86 |
| C98787 | 91 |

The expected output would be \$110, as you have one winning ticket, and one ticket that has some but not all the winning numbers.

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

Puzzle #55**Table Audit**

Audit the below tables and produce the expected output.

Products A

| Product Name | Quantity |
|--------------|----------|
| Widget | 7 |
| Doodad | 9 |
| Gizmo | 3 |

Products B

| Product Name | Quantity |
|--------------|----------|
| Widget | 7 |
| Doodad | 6 |
| Dingbat | 9 |

Here is the expected output.

| Type | ProductName |
|--|-------------|
| Matches In both tables | Widget |
| Product does not exist in table A | Dingbat |
| Product does not exist in table B | Gizmo |
| Quantity is table A and table B do not match | Doodad |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://advancedsqlpuzzles.com/)

Puzzle #56

Numbers Using Recursion

Create a numbers table using a recursive query.

Here is the expected output.

| Number |
|--------|
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
| 7 |
| 8 |
| 9 |
| 10 |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

Puzzle #57**Find The Spaces**

Given the following table of SQL statements, provide a numbers table that displays a summary of the space character for each SQL statement.

| Statement |
|-------------------------|
| SELECT EmpID FROM Emps; |
| SELECT * FROM Trans; |

Here is the expected output.

| RowNumber | QuoteId | String | Starts | Position | Word | TotalSpaces |
|-----------|---------|-------------------------|--------|----------|--------|-------------|
| 1 | 1 | SELECT EmpID FROM Emps; | 1 | 7 | SELECT | 3 |
| 2 | 1 | SELECT EmpID FROM Emps; | 8 | 13 | EmpID | 3 |
| 3 | 1 | SELECT EmpID FROM Emps; | 14 | 18 | FROM | 3 |
| 4 | 1 | SELECT EmpID FROM Emps; | 19 | 0 | Emps; | 3 |
| 1 | 2 | SELECT * FROM Trans; | 1 | 7 | SELECT | 3 |
| 2 | 2 | SELECT * FROM Trans; | 8 | 9 | * | 3 |
| 3 | 2 | SELECT * FROM Trans; | 10 | 14 | FROM | 3 |
| 4 | 2 | SELECT * FROM Trans; | 15 | 0 | Trans; | 3 |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

Puzzle #58

Add Them Up

You are given the following table which contains a VARCHAR field that contains mathematical equations. Sum the equations and provide the answers in the output.

| Equation |
|----------|
| 123 |
| 1+2+3 |
| 1+2-3 |
| 1+23 |
| 1-2+3 |
| 1-2-3 |
| 1-23 |
| 12+3 |
| 12-3 |

Here is the expected output.

| Permutation | Sum |
|-------------|-----|
| 123 | 123 |
| 1+2+3 | 6 |
| 1+2-3 | 0 |
| 1+23 | 24 |
| 1-2+3 | 2 |
| 1-2-3 | -4 |
| 1-23 | -22 |
| 12+3 | 15 |
| 12-3 | 9 |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

Part II

Permutations, Combinations, Sequences and Random Numbers

Building Complex Numbers Table

Puzzle #1

Factorials

Create a numbers table of the numbers 1 through 10 and their factorial.

Here is the expected output.

| Number | Factorial |
|--------|-----------|
| 1 | 1 |
| 2 | 2 |
| 3 | 6 |
| 4 | 24 |
| 5 | 120 |
| 6 | 720 |
| 7 | 5,040 |
| 8 | 40,320 |
| 9 | 362,880 |
| 10 | 3,628,800 |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

Puzzle #2

All Permutations

Create a numbers table of all permutations of n distinct numbers.

Here is the expected output for the set of {1, 2, 3}.

| Max Number | Permutation |
|------------|-------------|
| 3 | 1,2,3 |
| 3 | 1,3,2 |
| 3 | 2,1,3 |
| 3 | 2,3,1 |
| 3 | 3,1,2 |
| 3 | 3,2,1 |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

Puzzle #3

Growing Numbers

Create a numbers table that shows all numbers 1 through n and their order gradually increasing by the next number in the sequence.

Here is the expected output where $n = 5$.

| Permutation |
|-------------|
| 1 |
| 12 |
| 123 |
| 1234 |
| 12345 |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

Puzzle #4

Non-Adjacent Numbers

Given an ordered set of numbers (for example {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}); create a result set of permutations where no two adjacent entries are adjacent numbers.

For example...

The arrangement {1, 3, 5, 7, 9, 2, 4, 6, 8, 10} **would** fit the criteria as no two entries are adjacent numbers.

The arrangement {**1**, **2**, 4, 6, 8, 10, 3, 5, 7, 9} would **not** fit the criteria as 1 and 2 are adjacent numbers.

The arrangement {1, 4, 2, 6, 7, 10, 3, 5, **8**, **9**} would **not** fit the criteria as 6 and 7 are adjacent numbers.

The arrangement {1, **3**, **2**, 6, 7, 10, 9, 5, 8, 4} would **not** fit the criteria as 3 and 2 are adjacent numbers.

Here is the expected output for the set of {1, 2, 3, 4}.

| Permutation |
|-------------|
| 2,4,1,3 |
| 3,1,4,2 |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

Puzzle #5

Add the Numbers Up

Given an ordered set of numbers 1 through n (for example: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10), and a + or – sign at all possible groupings; create all possible permutations and the amount in which they add up to.

Here is the expected output for the set of {1, 2, 3}.

| Permutation | Sum |
|-------------|-----|
| 123 | 123 |
| 1+2+3 | 6 |
| 1+2-3 | 0 |
| 1+23 | 24 |
| 1-2+3 | 2 |
| 1-2-3 | -4 |
| 1-23 | -22 |
| 12+3 | 15 |
| 12-3 | 9 |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

Puzzle #6**Permutations of 0 and 1**

Create a result set of all permutations for the combination of 0 and 1 with a length of n digits?

Here is the expected output for permutations with a length of 3 digits.

| Permutation |
|-------------|
| 000 |
| 001 |
| 010 |
| 011 |
| 100 |
| 101 |
| 110 |
| 111 |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

Puzzle #7

Permutations 1 through 10

Display all permutations for the numbers 1 through n , displaying only the first n numbers.

Here is the expected output for all 24 permutations for the set {1,2,3,4}, displaying only the first three numbers.

| Permutation |
|-------------|
| 1,2,3 |
| 1,2,4 |
| 1,3,2 |
| 1,3,4 |
| 1,4,2 |
| 1,4,3 |
| 2,1,3 |
| 2,1,4 |
| 2,3,1 |
| 2,3,4 |
| 2,4,1 |
| 2,4,3 |
| 3,1,2 |
| 3,1,4 |
| 3,2,1 |
| 3,2,4 |
| 3,4,1 |
| 3,4,2 |
| 4,1,2 |
| 4,1,3 |
| 4,2,1 |
| 4,2,3 |
| 4,3,1 |
| 4,3,2 |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

Puzzle #8

Four Vehicles Problem

Here is an example problem involving combinations.

Given the following four vehicles:

- 1-seat motorcycle
- 2-seat sidecar
- 3-seat golf cart
- 4-seat car

There are 10 people total; 5 are children, 5 are adults. Only an adult can drive a vehicle.

Create a table of all possible 7,200 arrangements, assuming seating order does not matter.

We can determine there are 7,200 arrangements by using the following equation.

$$\text{Total Arrangements} = \frac{5!}{1!} * \frac{6!}{3!*2!*1!*0!} = 7,200$$

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

Puzzle #9**Find The Spaces**

Given the following table of SQL statements, provide a numbers table that displays a summary of the space character for each SQL statement.

| Statement |
|-------------------------|
| SELECT EmpID FROM Emps; |
| SELECT * FROM Trans; |

Here is the expected output.

| RowNumber | Id | String | Starts | Position | Word | TotalSpaces |
|-----------|----|-------------------------|--------|----------|--------|-------------|
| 1 | 1 | SELECT EmpID FROM Emps; | 1 | 7 | SELECT | 3 |
| 2 | 1 | SELECT EmpID FROM Emps; | 8 | 13 | EmpID | 3 |
| 3 | 1 | SELECT EmpID FROM Emps; | 14 | 18 | FROM | 3 |
| 4 | 1 | SELECT EmpID FROM Emps; | 19 | 0 | Emps; | 3 |
| 1 | 2 | SELECT * FROM Trans; | 1 | 7 | SELECT | 3 |
| 2 | 2 | SELECT * FROM Trans; | 8 | 9 | * | 3 |
| 3 | 2 | SELECT * FROM Trans; | 10 | 14 | FROM | 3 |
| 4 | 2 | SELECT * FROM Trans; | 15 | 0 | Trans; | 3 |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

Puzzle #10**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 |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

Puzzle #11**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.

| Min Step Number | Max Step Number | Status | Consecutive Count |
|-----------------|-----------------|--------|-------------------|
| 1 | 4 | Passed | 4 |
| 5 | 9 | Failed | 5 |
| 10 | 12 | Passed | 3 |

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

Puzzle #12**Double or Add 1**

Create a numbers table where you start with the number 1, and then double the number if the result is less than 100, else add 1.

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](#)

Puzzle #13**Pascal's Triangle**

Solve for any position in [Pascal's Triangle](#).

Answers to the puzzles are located in the following GitHub repository.
[AdvancedSQLPuzzles/Advanced SQL Puzzles](#)

Puzzle #14

Josephus Problem

Solve the [Josephus Problem](#).

Answers to the puzzles are located in the following GitHub repository.
[AdvancedSQLPuzzles/Advanced SQL Puzzles](#)

Puzzle #15**High-Low Card Game**

Write a program that shuffles a standard deck of cards and plays a game of High-Low.

The game is played by receiving an initial card and then determining if the next card will be of higher or lower value based upon probability. Make a random decision of higher or lower where necessary.

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](https://github.com/AdvancedSQLPuzzles/AdvancedSQLPuzzles)

Puzzle #16

Monty Hall Simulation

Run 10,000 simulations of the [Monty Hall problem](#) to prove it is true.

Answers to the puzzles are located in the following GitHub repository.
[AdvancedSQLPuzzles/Advanced SQL Puzzles](#)

Puzzle #17**Dice Throw Game**

What is the average number of dice throws needed to reach 100 points given the following rules?

- Starting at 0, for each dice throw resulting in 1 through 5, add the dice amount to your score.
- If you roll a 6 (even on a re-roll), re-roll the dice and reduce your score by this amount. You cannot go lower than 0 points.

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](#)

Puzzle #18

The Birthday Problem

Run 10,000 simulations of the [Birthday problem](#) to prove it is true.

Answers to the puzzles are located in the following GitHub repository.
[AdvancedSQLPuzzles/Advanced SQL Puzzles](#)

Puzzle #19**Random Walk**

Perform a [random walk](#) best described by the following puzzle:

You are the dinner host of a weekly gathering of 8 friends (including yourself). You want to determine who hosts of the next dinner party using the following scheme:

After dinner, the guests sit around a round table. You (the current host) flip a fair coin. If it comes up heads, you pass it to the person on your right, and if it comes up tails, you pass it to person on your left. The person who receives the coin repeats the procedure, flipping it and passing it right or left, depending on the outcome of the flip. This process keeps going until all but one dinner guest has held the coin. The last member who has not yet touched the coin is then declared the winner and must host the next dinner party.

Which person is more likely to host the next dinner party?

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](#)

Puzzle #20**Markov Chain**

Perform a [Markov Chain](#) best described by the following puzzle:

In Probability Land, on a sunny day there is an equal probability of the next day being sunny or rainy. On a rainy day, there is a 70% chance it will rain the next day, and a 30% chance it will be sunny the next day.

On average, how many rainy days are there in Probability Land?

Answers to the puzzles are located in the following GitHub repository.

[AdvancedSQLPuzzles/Advanced SQL Puzzles](#)

Puzzle #21

100 Prisoners Riddle

Run 10,000 simulations of the [100 Prisoners Problem](#) to prove it is true.

Answers to the puzzles are located in the following GitHub repository.
[AdvancedSQLPuzzles/Advanced SQL Puzzles](#)

THE END