In [ ]:
```python
from pyspark.sql import SparkSession, Window
from pyspark.sql.types import StructType, StructField, IntegerType, StringType, BooleanType, TimestampType, DateType
from pyspark.sql.functions import sum, col, count, when, lead, collect_list, length, year, countDistinct, round, first, lit
from datetime import datetime
```

In [ ]:
```python
spark = SparkSession.builder.master('local[*]').appName('PysparkDataPipeline').getOrCreate()
```

Creating an empty Df -

In [ ]:
```python
schema = StructType([StructField('column_a', IntegerType(), True),
                     StructField('column_b', StringType(), False)])
```

In [ ]:
```python
df = spark.createDataFrame([], schema=schema)
```

In [ ]:
```python
df.show()
```

```
+--------+--------+
|column_a|column_b|
+--------+--------+
+--------+--------+
```

Append Data into it

In [ ]:
```python
df2 = spark.createDataFrame([[1, 'a'],
                             [2, 'b'],
                             [3, 'c'],
                             [None, 'd']], schema=schema)
```

In [ ]:
```python
df2.show()
```

```
+--------+--------+
|column_a|column_b|
+--------+--------+
|       1|       a|
|       2|       b|
|       3|       c|
|    null|       d|
+--------+--------+
```

In [ ]:
```python
df = df.union(df2)
```

In [ ]:
```python
df.show()
```

```
+--------+--------+
|column_a|column_b|
+--------+--------+
|       1|       a|
|       2|       b|
|       3|       c|
|    null|       d|
+--------+--------+
```

Fill Null Values

In [ ]:
```
df = df.fillna(1, subset=['column_a'])
```

In [ ]:
```
df.show()
```

```
+--------+--------+
|column_a|column_b|
+--------+--------+
|       1|       a|
|       2|       b|
|       3|       c|
|       1|       d|
+--------+--------+
```

Replace all 1's with 2's

In [ ]:
```
from pyspark.sql.functions import when
df = df.replace(1, 2, subset=['column_a'])
df.show()
```

```
+--------+--------+
|column_a|column_b|
+--------+--------+
|       2|       a|
|       2|       b|
|       3|       c|
|       2|       d|
+--------+--------+
```

LeetCode SQL problem

Replace Employee ID With The Unique Identifier

https://leetcode.com/problems/replace-employee-id-with-the-unique-identifier/description/?envType=study-plan-v2&envId=top-sql-50

In [ ]:
```
from pyspark.sql.types import StructType, StructField, IntegerType, StringType
employees_schema = StructType([StructField('id', IntegerType(), False),
                               StructField('name', StringType(), True)])
employees_df = spark.createDataFrame([[1, 'Alice'],
                                      [7, 'Bob'],
                                      [11, 'Meir'],
                                      [90, 'Winston'],
                                      [3, 'Jonathan']], schema=employees_schema)

employee_uni_schema = StructType([StructField('id', IntegerType(), False),
                                  StructField('unique_id', IntegerType(), True)])
employee_uni_df = spark.createDataFrame([[3, 1],
                                         [11, 2],
                                         [90, 3]], schema=employee_uni_schema)
```

In [ ]:
```
employees_df.show()
```

```
+---+--------+
| id|    name|
+---+--------+
|  1|   Alice|
|  7|     Bob|
| 11|    Meir|
| 90| Winston|
|  3|Jonathan|
+---+--------+
```

In [ ]:
```
employee_uni_df.show()
```

```
+---+---------+
| id|unique_id|
+---+---------+
|  3|        1|
| 11|        2|
| 90|        3|
+---+---------+
```

In [ ]:
```
employees_df.join(employee_uni_df, how='left', on=['id']).select('unique_id', 'name').show()
```

```
+---------+--------+
|unique_id|    name|
+---------+--------+
|     null|   Alice|
|     null|     Bob|
|        2|    Meir|
|        1|Jonathan|
|        3| Winston|
+---------+--------+
```

Leetcode SQL problem - Confirmation Rate

https://leetcode.com/problems/confirmation-rate/description/?envType=study-plan-v2&envId=top-sql-50

In [ ]:
```
signups_schema = StructType([StructField('user_id', IntegerType(), False),
                             StructField('time_stamp', TimestampType(), True)])
confirmations_schema = StructType([StructField('user_id', IntegerType(), True),
                                   StructField('time_stamp', TimestampType(), True),
                                   StructField('action', StringType(), True)])
```

In [ ]:
```
sinups_df = spark.createDataFrame([[3, datetime.strptime('2020-03-21 10:16:13', '%Y-%m-%d %H:%M:%S')],
                                   [7, datetime.strptime('2020-01-04 13:57:59', '%Y-%m-%d %H:%M:%S')],
                                   [2, datetime.strptime('2020-07-29 23:09:44', '%Y-%m-%d %H:%M:%S')],
                                   [6, datetime.strptime('2020-12-09 10:39:37', '%Y-%m-%d %H:%M:%S')]], schema=signups_schema)
confirmations_df = spark.createDataFrame([[3, datetime.strptime('2021-01-06 03:30:46', '%Y-%m-%d %H:%M:%S'), 'timeout'],
                                          [3, datetime.strptime('2021-07-14 14:00:00', '%Y-%m-%d %H:%M:%S'), 'timeout'],
                                          [7, datetime.strptime('2021-06-12 11:57:29', '%Y-%m-%d %H:%M:%S'), 'confirmed'],
                                          [7, datetime.strptime('2021-06-13 12:58:28', '%Y-%m-%d %H:%M:%S'), 'confirmed'],
                                          [7, datetime.strptime('2021-06-14 13:59:27', '%Y-%m-%d %H:%M:%S'), 'confirmed'],
                                          [2, datetime.strptime('2021-01-22 00:00:00', '%Y-%m-%d %H:%M:%S'), 'confirmed'],
                                          [2, datetime.strptime('2021-02-28 23:59:59', '%Y-%m-%d %H:%M:%S'), 'timeout']], schema=confirmations_schema)
```

In [ ]:
```
result_df = sinups_df.join(confirmations_df, how='left', on=['user_id'])
```

In [ ]:
```
result_df.groupBy('user_id').agg((sum(when(result_df.action == 'confirmed', 1).otherwise(
    0.00)) / count('*')).alias('confirmation_rate')).show()
```

```
+-------+-----------------+
|user_id|confirmation_rate|
+-------+-----------------+
|      3|              0.0|
|      7|              1.0|
|      2|              0.5|
|      6|              0.0|
+-------+-----------------+
```

Find Users With Valid E-Mails

https://leetcode.com/problems/find-users-with-valid-e-mails/description/?envType=study-plan-v2&envId=30-days-of-pandas&lang=pythondata

In [ ]:
```
schema = StructType([StructField('user_id', IntegerType(), False),
                     StructField('name', StringType(), True),
                     StructField('mail', StringType(), True)])
df = spark.createDataFrame([[1, 'Winston', 'winston@leetcode.com'],
                            [2, 'Jonathan', 'jonathanisgreat'],
                            [3, 'Annabelle', 'bella-@leetcode.com'],
                            [4, 'Sally', 'sally.come@leetcode.com'],
                            [5, 'Marwan', 'quarz#2020@leetcode.com'],
```

```
                              [6, 'David', 'david69@gmail.com'],
                              [7, 'Shapiro', '.shapo@leetcode.com']], schema=schema)
```

In [ ]:
```
df.show()
```

```
+-------+---------+--------------------+
|user_id|     name|                mail|
+-------+---------+--------------------+
|      1|  Winston|winston@leetcode.com|
|      2| Jonathan|     jonathanisgreat|
|      3|Annabelle| bella-@leetcode.com|
|      4|    Sally|sally.come@leetco...|
|      5|   Marwan|quarz#2020@leetco...|
|      6|    David|   david69@gmail.com|
|      7|  Shapiro| .shapo@leetcode.com|
+-------+---------+--------------------+
```

In [ ]:
```
df.filter(col('mail').rlike('^[a-zA-Z][a-zA-Z0-9_.-]*@leetcode\.com')).show()
```

```
+-------+---------+--------------------+
|user_id|     name|                mail|
+-------+---------+--------------------+
|      1|  Winston|winston@leetcode.com|
|      3|Annabelle| bella-@leetcode.com|
|      4|    Sally|sally.come@leetco...|
+-------+---------+--------------------+
```

Consecutive Numbers

https://leetcode.com/problems/consecutive-numbers/

In [ ]:
```
schema = StructType([StructField('id', IntegerType(), False),
                     StructField('num', StringType(), True)])
df = spark.createDataFrame([[1, '1'],
                            [2, '1'],
                            [3, '1'],
                            [4, '2'],
                            [5, '1'],
                            [6, '2'],
                            [7, '2']], schema=schema)
```

In [ ]:
```
df.createOrReplaceTempView('df')
```

In [ ]:
```
spark.sql('''with temp as (
                select num,
                    lead(num, 1) over (order by id) num1,
                    lead(num, 2) over (order by id) num2
                from df
```

```
                )
                select distinct num
                from temp
                where num = num1
                and num = num2''').show()
```

```
+---+
|num|
+---+
|  1|
+---+
```

```
23/08/03 15:05:56 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradatio
n.
23/08/03 15:05:56 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradatio
n.
23/08/03 15:05:57 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradatio
n.
23/08/03 15:05:57 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradatio
n.
```

In [ ]:
```
df.withColumn('num1', lead('num', 1).over(Window.orderBy('id')))\
    .withColumn('num2', lead('num', 2).over(Window.orderBy('id')))\
    .filter((col('num') == col('num1')) & (col('num') == col('num2')))\
    .select('id').distinct().show()
```

```
+---+
| id|
+---+
|  1|
+---+
```

```
23/08/03 15:56:26 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradatio
n.
23/08/03 15:56:26 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradatio
n.
23/08/03 15:56:26 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradatio
n.
23/08/03 15:56:26 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradatio
n.
```

Students and classes example

In [ ]:
```
schema = StructType([StructField('class', IntegerType(), True),
                     StructField('student_id', IntegerType(), True),
                     StructField('term', IntegerType(), True),
                     StructField('subject', StringType(), True),
                     StructField('marks', IntegerType(), True)])

df = spark.createDataFrame([[2, 1, 1, 'maths', 10],
```

```
                                [2, 1, 2, 'maths', 12],
                                [2, 1, 1, 'english', 14],
                                [2, 1, 2, 'english', 12],
                                [3, 2, 1, 'maths', 10],
                                [3, 2, 2, 'maths', 12],
                                [3, 2, 1, 'english', 16],
                                [3, 2, 2, 'english', 14]], schema=schema)
```

In [ ]:
```
df.show()
```

```
+-----+----------+----+-------+-----+
|class|student_id|term|subject|marks|
+-----+----------+----+-------+-----+
|    2|         1|   1|  maths|   10|
|    2|         1|   2|  maths|   12|
|    2|         1|   1|english|   14|
|    2|         1|   2|english|   12|
|    3|         2|   1|  maths|   10|
|    3|         2|   2|  maths|   12|
|    3|         2|   1|english|   16|
|    3|         2|   2|english|   14|
+-----+----------+----+-------+-----+
```

> Get class 2 students in following format --> class student_id subject term1 term2

In [ ]:
```
df = df.orderBy('class', 'student_id', 'subject')\
    .groupBy('class', 'student_id', 'subject')\
    .agg(collect_list('marks')[0].alias('term1'), collect_list('marks')[1].alias('term2'))
```

In [ ]:
```
df.show()
```

```
+-----+----------+-------+-----+-----+
|class|student_id|subject|term1|term2|
+-----+----------+-------+-----+-----+
|    2|         1|english|   14|   12|
|    2|         1|  maths|   10|   12|
|    3|         2|english|   16|   14|
|    3|         2|  maths|   10|   12|
+-----+----------+-------+-----+-----+
```

In [ ]:
```
df.filter(col('class') == 2).show()
```

```
+-----+----------+-------+-----+-----+
|class|student_id|subject|term1|term2|
+-----+----------+-------+-----+-----+
|    2|         1|english|   14|   12|
|    2|         1|  maths|   10|   12|
+-----+----------+-------+-----+-----+
```

> Get subject-wise aggregated score with 25% weightage to term1 and 75% weightage to term2

In [ ]:
```
df = spark.createDataFrame([[2, 1, 1, 'maths', 10],
                            [2, 1, 2, 'maths', 12],
                            [2, 1, 1, 'english', 14],
                            [2, 1, 2, 'english', 12],
                            [3, 2, 1, 'maths', 10],
                            [3, 2, 2, 'maths', 12],
                            [3, 2, 1, 'english', 16],
                            [3, 2, 2, 'english', 14]], schema=schema)
```

In [ ]:
```
df.show()
```

```
+-----+----------+----+-------+-----+
|class|student_id|term|subject|marks|
+-----+----------+----+-------+-----+
|    2|         1|   1|  maths|   10|
|    2|         1|   2|  maths|   12|
|    2|         1|   1|english|   14|
|    2|         1|   2|english|   12|
|    3|         2|   1|  maths|   10|
|    3|         2|   2|  maths|   12|
|    3|         2|   1|english|   16|
|    3|         2|   2|english|   14|
+-----+----------+----+-------+-----+
```

In [ ]:
```
maths_agg = \
    df.filter(col('subject') == 'maths')\
    .orderBy('student_id', 'term')\
    .groupBy('student_id')\
    .agg((collect_list('marks')[0] * 0.25 + collect_list('marks')[1] * 0.75).alias('maths_agg'))
```

In [ ]:
```
english_agg = \
    df.filter(col('subject') == 'english')\
    .orderBy('student_id', 'term')\
    .groupBy('student_id')\
    .agg((collect_list('marks')[0] * 0.25 + collect_list('marks')[1] * 0.75).alias('english_agg'))
```

In [ ]:
```
maths_agg.join(english_agg, how='inner', on='student_id').show()
```

```
+----------+---------+-----------+
|student_id|maths_agg|english_agg|
+----------+---------+-----------+
|         1|     11.5|       12.5|
|         2|     11.5|       14.5|
+----------+---------+-----------+
```

Exchange Seats -

https://leetcode.com/problems/exchange-seats/description/

In [ ]:
```
import pandas as pd
import io
data = '''
id,student
1,Abbot
2,Doris
3,Emerson
4,Green
5,Jeames
'''
df = spark.createDataFrame(pd.read_csv(io.StringIO(data), header=0))
```

In [ ]:
```
df.show()
```

```
+---+-------+
| id|student|
+---+-------+
|  1|  Abbot|
|  2|  Doris|
|  3|Emerson|
|  4|  Green|
|  5| Jeames|
+---+-------+
```

In [ ]:
```
max_id = max(df.select('id').collect())[0]
df.select(when(col('id') % 2 == 1,
               lead('id', 1, max_id).over(Window.orderBy('id')))
          .otherwise(col('id') - 1).alias('id'),
          'Student').orderBy('id').show()
```

```
+---+-------+
| id|Student|
+---+-------+
|  1|  Doris|
|  2|  Abbot|
|  3|  Green|
|  4|Emerson|
|  5| Jeames|
+---+-------+
```

```
23/08/09 17:41:37 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradatio
n.
23/08/09 17:41:37 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradatio
n.
23/08/09 17:41:37 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradatio
n.
23/08/09 17:41:37 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradatio
n.
23/08/09 17:41:37 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradatio
n.
```

Tree Node

https://leetcode.com/problems/tree-node/

In [ ]:

```
schema = StructType([StructField('id', IntegerType(), False),
                     StructField('p_id', IntegerType(), True)])
df = spark.createDataFrame(
    [[1, None],
     [2, 1],
     [3, 1],
     [4, 2],
     [5, 2]], schema=schema)
```

In [ ]:

```
df.show()
```

```
+---+----+
| id|p_id|
+---+----+
|  1|null|
|  2|   1|
|  3|   1|
|  4|   2|
|  5|   2|
+---+----+
```

In [ ]:

```
distinct_parent_ids = df.select(
    'p_id').distinct().rdd.flatMap(lambda x: x).collect()
```

```
df.select('id',
          when(col('p_id').isNull(), 'Root')
          .when(col('p_id').isNotNull() & col('id').isin(distinct_parent_ids), 'Inner')
          .otherwise('Leaf').alias('type')).show()
```

```
+---+-----+
| id| type|
+---+-----+
|  1| Root|
|  2|Inner|
|  3| Leaf|
|  4| Leaf|
|  5| Leaf|
+---+-----+
```

Pandas

In [ ]:
```
pdf = df.toPandas()
```

In [ ]:
```
pdf
```

Out[ ]:

	id	p_id
0	1	NaN
1	2	1.0
2	3	1.0
3	4	2.0
4	5	2.0

In [ ]:
```
import numpy as np
pdf['type'] = np.where(pdf['p_id'].isna(),
                       'Root',
                       np.where(pdf['id'].isin(pdf['p_id'].unique()) & pdf['p_id'].notna(),
                                'Inner',
                                'Leaf'))
```

In [ ]:
```
pdf[['id', 'type']]
```

Out[ ]:

	id	type
0	1	Root
1	2	Inner
2	3	Leaf
3	4	Leaf
4	5	Leaf

Customers who bought all products

https://leetcode.com/problems/customers-who-bought-all-products/description/

In [ ]:
```
customer_schema = StructType([StructField('customer_id', IntegerType(), False),
                              StructField('product_key', IntegerType(), False)])
product_schema = StructType([StructField('product_key', IntegerType(), False)])
customer_df = spark.createDataFrame([[1, 5],
                                     [2, 6],
                                     [3, 5],
                                     [3, 6],
                                     [1, 6]], schema=customer_schema)
product_df = spark.createDataFrame([[5], [6]], schema=product_schema)
```

In [ ]:
```
customer_df.show()
```

```
+-----------+-----------+
|customer_id|product_key|
+-----------+-----------+
|          1|          5|
|          2|          6|
|          3|          5|
|          3|          6|
|          1|          6|
+-----------+-----------+
```

In [ ]:
```
product_df.show()
```

```
+-----------+
|product_key|
+-----------+
|          5|
|          6|
+-----------+
```

In [ ]:
```
grouped_df = customer_df.dropDuplicates().groupBy(
    'customer_id').agg(count(col('customer_id')).alias('count'))
```

In [ ]:
```
grouped_df.show()
```

```
+-----------+-----+
|customer_id|count|
+-----------+-----+
|          1|    2|
|          3|    2|
|          2|    1|
+-----------+-----+
```

In [ ]:
```
grouped_df.filter(col('count') == product_df.count()).select('customer_id').show()
```

```
+-----------+
|customer_id|
+-----------+
|          1|
|          3|
+-----------+
```

Triangle Judgement

https://leetcode.com/problems/triangle-judgement/description/

In [ ]:
```
schema = StructType([StructField('x', IntegerType(), False),
                     StructField('y', IntegerType(), False),
                     StructField('z', IntegerType(), False)])
df = spark.createDataFrame([[13, 15, 30],
                            [10, 20, 15]], schema=schema)
```

In [ ]:
```
df.show()
```

```
+---+---+---+
|  x|  y|  z|
+---+---+---+
| 13| 15| 30|
| 10| 20| 15|
+---+---+---+
```

In [ ]:
```
df.withColumn('triangle', when((col('x') + col('y') > col('z'))
                               & (col('y') + col('z') > col('x'))
                               & (col('x') + col('z') > col('y')), 'Yes')
              .otherwise('No')).show()
```

```
+---+---+---+--------+
|  x|  y|  z|triangle|
+---+---+---+--------+
| 13| 15| 30|      No|
| 10| 20| 15|     Yes|
+---+---+---+--------+
```

Invalid Tweets

https://leetcode.com/problems/invalid-tweets/?envType=study-plan-v2&envId=30-days-of-pandas&lang=pythondata

In [ ]:
```
schema = StructType([StructField('tweet_id', IntegerType(), False),
                     StructField('content', StringType(), True)])
df = spark.createDataFrame([[1, 'Vote for Biden'],
                            [2, 'Let us make America great again!']], schema=schema)
```

In [ ]:
```
df.show(truncate=False)
```

```
+--------+-------------------------------+
|tweet_id|content                        |
+--------+-------------------------------+
|1       |Vote for Biden                 |
|2       |Let us make America great again!|
+--------+-------------------------------+
```

In [ ]:
```
df.filter(length(col('content')) > 15).select('tweet_id').show()
```

```
+--------+
|tweet_id|
+--------+
|       2|
+--------+
```

Calculate Special Bonus

https://leetcode.com/problems/calculate-special-bonus/description/?envType=study-plan-v2&envId=30-days-of-pandas&lang=pythondata

In [ ]:
```
schema = StructType([StructField('employee_id', IntegerType(), False),
                     StructField('name', StringType(), True),
                     StructField('salary', IntegerType(), True)])
df = spark.createDataFrame([[2, 'Meir', 3000],
                            [3, 'Michael', 3800],
                            [7, 'Addilyn', 7400],
                            [8, 'Juan', 6100],
                            [9, 'Kannon', 7700]], schema=schema)
```

In [ ]:
```
df.show(truncate=False)
```

```
+-----------+-------+------+
|employee_id|name   |salary|
+-----------+-------+------+
|2          |Meir   |3000  |
|3          |Michael|3800  |
|7          |Addilyn|7400  |
|8          |Juan   |6100  |
|9          |Kannon |7700  |
+-----------+-------+------+
```

In [ ]:
```
df.withColumn('bonus', when((col('employee_id') % 2 == 1) & (~col('name').startswith(
    'M')), col('salary')).otherwise(0).alias('bonus')).select('employee_id', 'bonus').show()
```

```
+-----------+-----+
|employee_id|bonus|
+-----------+-----+
|          2|    0|
|          3|    0|
|          7| 7400|
|          8|    0|
|          9| 7700|
+-----------+-----+
```

Market Analysis I

https://leetcode.com/problems/market-analysis-i/description/

In [ ]:
```
users_schema = StructType([StructField('user_id', IntegerType(), False),
                           StructField('join_date', StringType(), False),
                           StructField('favorite_brand', StringType(), False)])
users_df = spark.createDataFrame([[1, '2018-01-01', 'Lenovo'],
                                  [2, '2018-02-09', 'Samsung'],
                                  [3, '2018-01-09', 'LG'],
                                  [4, '2018-05-21', 'HP']], schema=users_schema)

orders_schema = StructType([StructField('order_id', IntegerType(), False),
                            StructField('order_date', StringType(), False),
                            StructField('item_id', IntegerType(), False),
                            StructField('buyer_id', IntegerType(), False),
                            StructField('seller_id', IntegerType(), False)])
orders_df = spark.createDataFrame([[1, '2019-08-01', 4, 1, 2],
                                   [2, '2018-08-02', 2, 1, 3],
                                   [3, '2019-08-03', 3, 2, 3],
                                   [4, '2018-08-04', 1, 4, 2],
                                   [5, '2018-08-04', 1, 3, 4],
                                   [6, '2019-08-05', 2, 2, 4]], schema=orders_schema)
```

In [ ]:
```
users_df = users_df.withColumn('join_date', col('join_date').cast('date'))
orders_df = orders_df.withColumn('order_date', col('order_date').cast('date'))
```

In [ ]:
```
users_df.printSchema()
```

```
root
 |-- user_id: integer (nullable = false)
 |-- join_date: date (nullable = true)
 |-- favorite_brand: string (nullable = false)
```

In [ ]:
```
orders_df.printSchema()
```

```
root
 |-- order_id: integer (nullable = false)
 |-- order_date: date (nullable = true)
 |-- item_id: integer (nullable = false)
 |-- buyer_id: integer (nullable = false)
 |-- seller_id: integer (nullable = false)
```

In [ ]:
```
orders_temp = orders_df.filter(year(col('order_date')) == 2019).groupBy(
    col('buyer_id')).agg(count('order_id').alias('orders_in_2019'))
orders_temp.show()
```

```
+--------+--------------+
|buyer_id|orders_in_2019|
+--------+--------------+
|       1|             1|
|       2|             2|
+--------+--------------+
```

In [ ]:
```
users_df.join(orders_temp, users_df.user_id == orders_temp.buyer_id, how='left')\
    .select(col('user_id').alias('buyer_id'), 'join_date', 'orders_in_2019')\
    .fillna(0, subset=['orders_in_2019']).show()
```

```
+--------+----------+--------------+
|buyer_id| join_date|orders_in_2019|
+--------+----------+--------------+
|       1|2018-01-01|             1|
|       2|2018-02-09|             2|
|       3|2018-01-09|             0|
|       4|2018-05-21|             0|
+--------+----------+--------------+
```

Department Highest Salary

https://leetcode.com/problems/department-highest-salary/

In [ ]:
```
employee_schema = StructType([StructField('id', IntegerType(), False),
                              StructField('name', StringType(), False),
                              StructField('salary', IntegerType(), False),
                              StructField('departmentId', IntegerType(), False)])
employee_df = spark.createDataFrame([[1, 'Joe', 70000, 1],
                                     [2, 'Jim', 90000, 1],
                                     [3, 'Henry', 80000, 2],
                                     [4, 'Sam', 60000, 2],
                                     [5, 'Max', 90000, 1]], schema=employee_schema)

department_schema = StructType([StructField('id', IntegerType(), False),
                                StructField('name', StringType(), False)])
department_df = spark.createDataFrame([[1, 'IT'],
                                       [2, 'Sales']], schema=department_schema)
```

In [ ]:
```
employee_df.show()
```

```
+---+-----+------+------------+
| id| name|salary|departmentId|
+---+-----+------+------------+
|  1|  Joe| 70000|           1|
|  2|  Jim| 90000|           1|
|  3|Henry| 80000|           2|
|  4|  Sam| 60000|           2|
|  5|  Max| 90000|           1|
+---+-----+------+------------+
```

In [ ]:
```
department_df.show()
```

```
+---+-----+
| id| name|
+---+-----+
|  1|   IT|
|  2|Sales|
+---+-----+
```

In [ ]:
```
max_salary = employee_df.groupby(col('departmentId')).max(
    'salary').withColumnRenamed('max(salary)', 'max_salary')
max_salary.show()
```

```
+------------+----------+
|departmentId|max_salary|
+------------+----------+
|           1|     90000|
|           2|     80000|
+------------+----------+
```

In [ ]:
```
max_salary = max_salary.withColumnRenamed('departmentId', 'depId')
employee_df = employee_df.withColumnRenamed('name', 'empName')
employee_df.join(max_salary,
                 (employee_df['departmentId'] == max_salary['depId'])
                 & (employee_df['salary'] == max_salary['max_salary']),
                 how='inner')\
    .join(department_df,
          employee_df['departmentId'] == department_df['id'],
          how='inner')\
    .select(col('name').alias('Department'), col('empName').alias('Employee'), col('salary').alias('Salary')).show(truncate=False)
```

```
+----------+--------+------+
|Department|Employee|Salary|
+----------+--------+------+
|IT        |Max     |90000 |
|IT        |Jim     |90000 |
|Sales     |Henry   |80000 |
+----------+--------+------+
```

The Number of Rich Customers

https://leetcode.com/problems/the-number-of-rich-customers/description/?envType=study-plan-v2&envId=30-days-of-pandas&lang=pythondata

In [ ]:
```
import pandas as pd
import io
```

In [ ]:
```
df = pd.read_csv(io.StringIO('''
bill_id,customer_id,amount
6,1,549
8,1,834
4,2,394
11,3,657
13,3,257
'''))
```

In [ ]:
```
df
```

Out[ ]:

	bill_id	customer_id	amount
0	6	1	549
1	8	1	834
2	4	2	394
3	11	3	657
4	13	3	257

In [ ]:
```
df = spark.createDataFrame(df)
```

In [ ]:
```
df.show()
```

```
+-------+-----------+------+
|bill_id|customer_id|amount|
+-------+-----------+------+
|      6|          1|   549|
|      8|          1|   834|
|      4|          2|   394|
|     11|          3|   657|
|     13|          3|   257|
+-------+-----------+------+
```

In [ ]:
```
df.filter(col('amount') > 500).select(countDistinct(col('customer_id')).alias('rich_count')).show()
```

```
+----------+
|rich_count|
+----------+
|         2|
+----------+
```

Project Employees I

https://leetcode.com/problems/project-employees-i/description/

In [ ]:
```
project_schema = StructType([StructField('project_id', IntegerType(), False),
                             StructField('employee_id', IntegerType(), True)])
project_df = spark.createDataFrame([[1, 1],
                                    [1, 2],
                                    [1, 3],
                                    [2, 1],
                                    [2, 4]], schema=project_schema)

employee_schema = StructType([StructField('employee_id', IntegerType(), False),
                              StructField('name', StringType(), False),
                              StructField('experience_years', IntegerType(), True)])
employee_df = spark.createDataFrame([[1, 'Khaled', 3],
                                     [2, 'Ali', 2],
                                     [3, 'John', 1],
                                     [4, 'Doe', 2]], schema=employee_schema)
```

In [ ]:
```
project_df.show()
```

```
+----------+-----------+
|project_id|employee_id|
+----------+-----------+
|         1|          1|
|         1|          2|
|         1|          3|
|         2|          1|
|         2|          4|
+----------+-----------+
```

In [ ]:
```
employee_df.show()
```

```
+-----------+------+----------------+
|employee_id|  name|experience_years|
+-----------+------+----------------+
|          1|Khaled|               3|
|          2|   Ali|               2|
|          3|  John|               1|
|          4|   Doe|               2|
+-----------+------+----------------+
```

In [ ]:
```
project_df.join(employee_df, on='employee_id', how='inner')\
    .groupBy('project_id').avg('experience_years')\
    .select(col('project_id'), round(col('avg(experience_years)'), 2).alias('average_years')).show()
```

```
+----------+-------------+
|project_id|average_years|
+----------+-------------+
|         1|          2.0|
|         2|          2.5|
+----------+-------------+
```

Customer Who Visited but Did Not Make Any Transactions

https://leetcode.com/problems/customer-who-visited-but-did-not-make-any-transactions/description/?envType=study-plan-v2&envId=top-sql-50

In [ ]:
```
visits_schema = StructType([StructField('visit_id', IntegerType(), False),
                            StructField('customer_id', IntegerType(), False)])
visits_df = spark.createDataFrame([[1, 23],
                                   [2, 9],
                                   [4, 30],
                                   [5, 54],
                                   [6, 96],
                                   [7, 54],
                                   [8, 54]], schema=visits_schema)

transactions_schema = StructType([StructField('transaction_id', IntegerType(), False),
                                  StructField('visit_id', IntegerType(), False),
```

```
                                StructField('amount', IntegerType(), False)])
transactions_df = spark.createDataFrame([[2, 5, 310],
                                         [3, 5, 300],
                                         [9, 5, 200],
                                         [12, 1, 910],
                                         [13, 2, 970]], schema=transactions_schema)
```

In [ ]:
```
visits_df.show()
```

```
+--------+-----------+
|visit_id|customer_id|
+--------+-----------+
|       1|         23|
|       2|          9|
|       4|         30|
|       5|         54|
|       6|         96|
|       7|         54|
|       8|         54|
+--------+-----------+
```

In [ ]:
```
transactions_df.show()
```

```
+--------------+--------+------+
|transaction_id|visit_id|amount|
+--------------+--------+------+
|             2|       5|   310|
|             3|       5|   300|
|             9|       5|   200|
|            12|       1|   910|
|            13|       2|   970|
+--------------+--------+------+
```

> If transactions_df is small in size -

In [ ]:
```
unique_visit_ids_who_transacted = [i[0] for i in transactions_df.select('visit_id').distinct().collect()]
visits_df.filter(~visits_df['visit_id'].isin(unique_visit_ids_who_transacted))\
    .groupBy('customer_id').count()\
    .withColumnRenamed('count', 'count_no_trans').show()
```

```
+-----------+--------------+
|customer_id|count_no_trans|
+-----------+--------------+
|         30|             1|
|         96|             1|
|         54|             2|
+-----------+--------------+
```

> If transactions_df is large to run collect(), we can use left join and picking records with null transaction_id

In [ ]:
```
visits_df.join(transactions_df, on='visit_id', how='left')\
    .filter(col('transaction_id').isNull())\
    .groupBy('customer_id').agg(count('visit_id'))\
    .withColumnRenamed('count(visit_id)', 'count_no_trans')\
    .show()
```

```
+-----------+--------------+
|customer_id|count_no_trans|
+-----------+--------------+
|         54|             2|
|         96|             1|
|         30|             1|
+-----------+--------------+
```

Product Price at a Given Date

https://leetcode.com/problems/product-price-at-a-given-date/description/

In [ ]:
```
product_schema = StructType([StructField('product_id', IntegerType(), False),
                             StructField('new_price', IntegerType(), True),
                             StructField('change_date', StringType(), True)])
product_df = spark.createDataFrame([[1, 20, '2019-08-14'],
                                    [2, 50, '2019-08-14'],
                                    [1, 30, '2019-08-15'],
                                    [1, 35, '2019-08-16'],
                                    [2, 65, '2019-08-17'],
                                    [3, 20, '2019-08-18']], schema=product_schema).select('product_id', 'new_price', col('change_date').cast('date'))
```

In [ ]:
```
product_df.show()
```

```
+----------+---------+-----------+
|product_id|new_price|change_date|
+----------+---------+-----------+
|         1|       20| 2019-08-14|
|         2|       50| 2019-08-14|
|         1|       30| 2019-08-15|
|         1|       35| 2019-08-16|
|         2|       65| 2019-08-17|
|         3|       20| 2019-08-18|
+----------+---------+-----------+
```

In [ ]:
```
from pyspark.sql.functions import min as min_
## note - it is important to alias the pyspark min function since otherwise it would use python min() function and break the code
```

```python
product_df.filter(col('change_date') <= '2019-08-16')\
.select('product_id', first('new_price').over(Window.partitionBy('product_id').orderBy(col('change_date').desc())).alias('price'))\
.union(
    product_df.groupBy('product_id').agg(min_('change_date').alias('change_date'))\
    .filter(col('change_date') > '2019-08-16').withColumn('price', lit(10)).select('product_id', 'price')
).dropDuplicates().show()
```

```
+----------+-----+
|product_id|price|
+----------+-----+
|         2|   50|
|         1|   35|
|         3|   10|
+----------+-----+
```

In [ ]: