

APACHE BEAM ASSIGNMENT

Name : Jebastin P

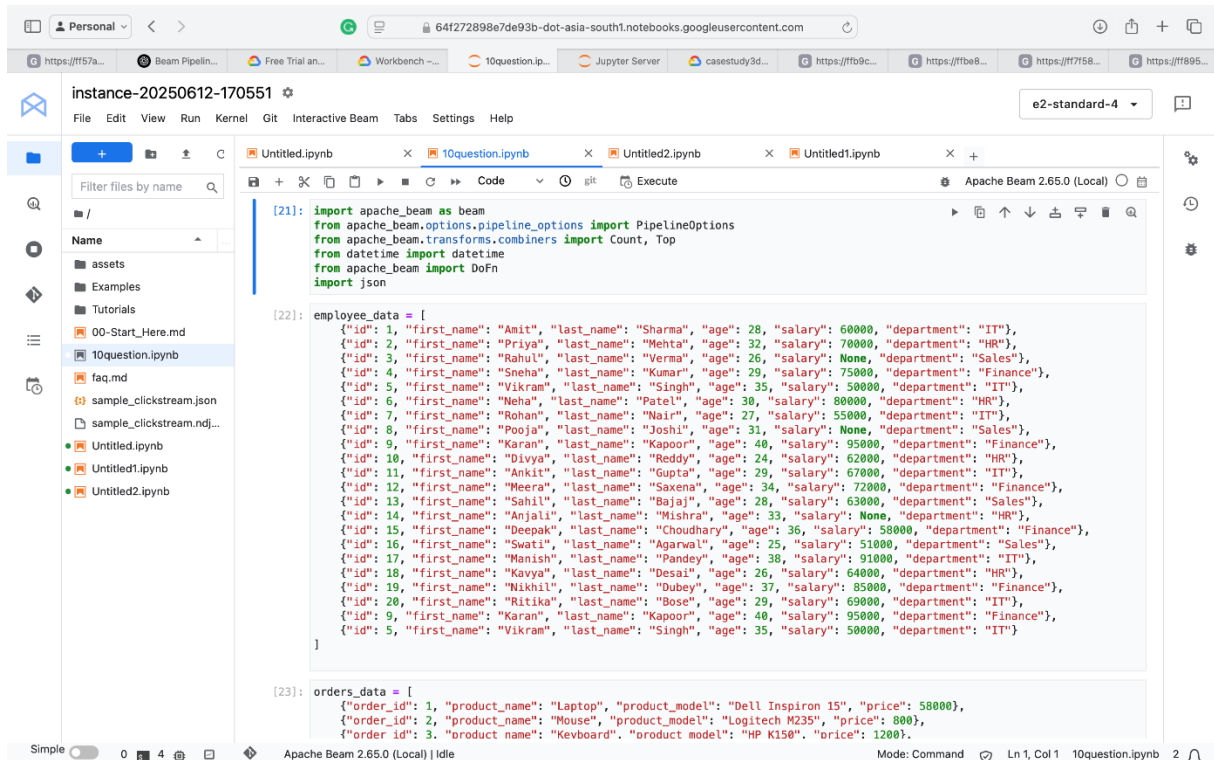
Emp ID: 2401171

Questions:

1. Write an Apache Beam pipeline to find the top 3 most occurring words in a given dataset.
2. Given a PCollection of records, write an Apache Beam pipeline to remove duplicate entries.
3. How do you perform word count using Apache Beam?
4. Write an Apache Beam pipeline to group records by a specific field and calculate the average value for each group.
5. How do you handle missing or null values in an Apache Beam pipeline?
6. Write an Apache Beam pipeline to count distinct values in a specific field of a dataset.
7. Given a PCollection of records, write an Apache Beam pipeline to filter records where the salary is greater than 50,000.
8. Write an Apache Beam pipeline to read a JSON file and convert it into a PCollection of structured records.
9. Write an Apache Beam pipeline to find the second highest salary from an employee dataset.
10. Write an Apache Beam pipeline to join two PCollections and select specific fields from the result.

Solutions:

Creating the dataset:

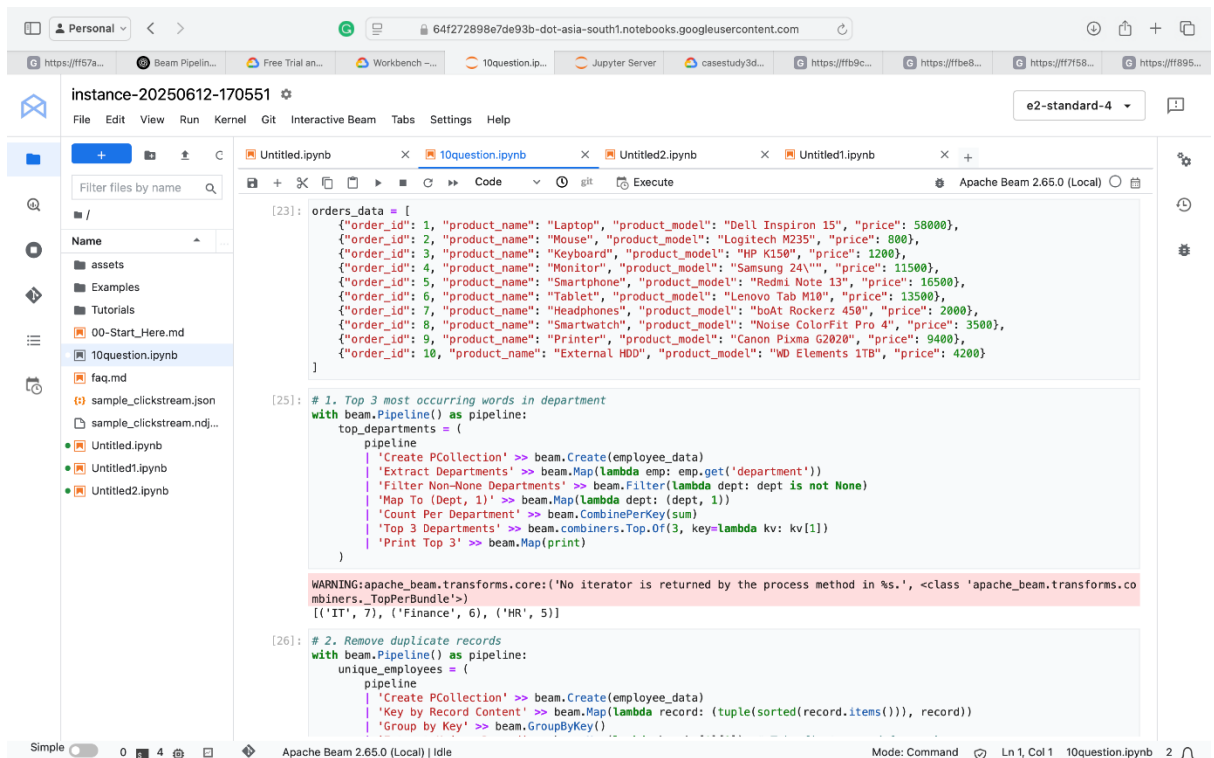


```
[21]: import apache_beam as beam
from apache_beam.options.pipeline_options import PipelineOptions
from apache_beam.transforms.combiners import Count, Top
from datetime import datetime
from apache_beam import DoFn
import json

[22]: employee_data = [
    {"id": 1, "first_name": "Amit", "last_name": "Sharma", "age": 28, "salary": 60000, "department": "IT"},
    {"id": 2, "first_name": "Priya", "last_name": "Mehta", "age": 32, "salary": 70000, "department": "HR"},
    {"id": 3, "first_name": "Rahul", "last_name": "Verma", "age": 26, "salary": None, "department": "Sales"},
    {"id": 4, "first_name": "Sneha", "last_name": "Kumar", "age": 29, "salary": 75000, "department": "Finance"},
    {"id": 5, "first_name": "Vikram", "last_name": "Singh", "age": 35, "salary": 50000, "department": "IT"},
    {"id": 6, "first_name": "Neha", "last_name": "Patel", "age": 30, "salary": 80000, "department": "HR"},
    {"id": 7, "first_name": "Rohan", "last_name": "Nair", "age": 27, "salary": 55000, "department": "IT"},
    {"id": 8, "first_name": "Pooja", "last_name": "Joshi", "age": 31, "salary": None, "department": "Sales"},
    {"id": 9, "first_name": "Karan", "last_name": "Kapoor", "age": 40, "salary": 95000, "department": "Finance"},
    {"id": 10, "first_name": "Divya", "last_name": "Reddy", "age": 24, "salary": 62000, "department": "HR"},
    {"id": 11, "first_name": "Ankit", "last_name": "Gupta", "age": 29, "salary": 67000, "department": "IT"},
    {"id": 12, "first_name": "Meera", "last_name": "Saxena", "age": 34, "salary": 72000, "department": "Finance"},
    {"id": 13, "first_name": "Sahil", "last_name": "Bajaj", "age": 28, "salary": 63000, "department": "Sales"},
    {"id": 14, "first_name": "Anjali", "last_name": "Mishra", "age": 33, "salary": None, "department": "HR"},
    {"id": 15, "first_name": "Deepak", "last_name": "Choudhary", "age": 36, "salary": 58000, "department": "Finance"},
    {"id": 16, "first_name": "Swati", "last_name": "Agarwal", "age": 25, "salary": 51000, "department": "Sales"},
    {"id": 17, "first_name": "Manish", "last_name": "Pandey", "age": 38, "salary": 91000, "department": "IT"},
    {"id": 18, "first_name": "Kavya", "last_name": "Desai", "age": 26, "salary": 64000, "department": "HR"},
    {"id": 19, "first_name": "Nikhil", "last_name": "Dubey", "age": 37, "salary": 85000, "department": "Finance"},
    {"id": 20, "first_name": "Ritika", "last_name": "Bose", "age": 29, "salary": 69000, "department": "IT"},
    {"id": 9, "first_name": "Karan", "last_name": "Kapoor", "age": 40, "salary": 95000, "department": "Finance"},
    {"id": 5, "first_name": "Vikram", "last_name": "Singh", "age": 35, "salary": 50000, "department": "IT"}

[23]: orders_data = [
    {"order_id": 1, "product_name": "Laptop", "product_model": "Dell Inspiron 15", "price": 58000},
    {"order_id": 2, "product_name": "Mouse", "product_model": "Logitech M235", "price": 800},
    {"order_id": 3, "product_name": "Keyboard", "product_model": "HP K150", "price": 1200},
    {"order_id": 4, "product_name": "Monitor", "product_model": "Samsung 24\"", "price": 11500},
    {"order_id": 5, "product_name": "Smartphone", "product_model": "Redmi Note 13", "price": 16500},
    {"order_id": 6, "product_name": "Tablet", "product_model": "Lenovo Tab M10", "price": 13500},
    {"order_id": 7, "product_name": "Headphones", "product_model": "boAt Rockerz 450", "price": 2000},
    {"order_id": 8, "product_name": "Smartwatch", "product_model": "Noise ColorFit Pro 4", "price": 3500},
    {"order_id": 9, "product_name": "Printer", "product_model": "Canon Pixma G2020", "price": 9400},
    {"order_id": 10, "product_name": "External HDD", "product_model": "WD Elements 1TB", "price": 4200}
```

Write an Apache Beam pipeline to find the top 3 most occurring words in a given dataset.



```
[23]: orders_data = [
    {"order_id": 1, "product_name": "Laptop", "product_model": "Dell Inspiron 15", "price": 58000},
    {"order_id": 2, "product_name": "Mouse", "product_model": "Logitech M235", "price": 800},
    {"order_id": 3, "product_name": "Keyboard", "product_model": "HP K150", "price": 1200},
    {"order_id": 4, "product_name": "Monitor", "product_model": "Samsung 24\"", "price": 11500},
    {"order_id": 5, "product_name": "Smartphone", "product_model": "Redmi Note 13", "price": 16500},
    {"order_id": 6, "product_name": "Tablet", "product_model": "Lenovo Tab M10", "price": 13500},
    {"order_id": 7, "product_name": "Headphones", "product_model": "boAt Rockerz 450", "price": 2000},
    {"order_id": 8, "product_name": "Smartwatch", "product_model": "Noise ColorFit Pro 4", "price": 3500},
    {"order_id": 9, "product_name": "Printer", "product_model": "Canon Pixma G2020", "price": 9400},
    {"order_id": 10, "product_name": "External HDD", "product_model": "WD Elements 1TB", "price": 4200}

[25]: # 1. Top 3 most occurring words in department
with beam.Pipeline() as pipeline:
    top_departments = (
        pipeline
        | 'Create PCollection' >> beam.Create(employee_data)
        | 'Extract Departments' >> beam.Map(lambda emp: emp.get('department'))
        | 'Filter Non-None Departments' >> beam.Filter(lambda dept: dept is not None)
        | 'Map To (Dept, 1)' >> beam.Map(lambda dept: (dept, 1))
        | 'Count Per Department' >> beam.CombinePerKey(sum)
        | 'Top 3 Departments' >> beam.combiners.TopOf(3, key=lambda kv: kv[1])
        | 'Print Top 3' >> beam.Map(print)

WARNING:apache_beam.transforms.core:(No iterator is returned by the process method in %s., <class 'apache_beam.transforms.combiners.TopPerBundle')
[('IT', 7), ('Finance', 6), ('HR', 5)]

[26]: # 2. Remove duplicate records
with beam.Pipeline() as pipeline:
    unique_employees = (
        pipeline
        | 'Create PCollection' >> beam.Create(employee_data)
        | 'Key by Record Content' >> beam.Map(lambda record: (tuple(sorted(record.items())), record))
        | 'Group by Key' >> beam.GroupByKey()
```

Given a PCollection of records, write an Apache Beam pipeline to remove duplicate entries.

The screenshot shows a Google Colab notebook with the following code in cell [26]:

```
[26]: # 2. Remove duplicate records
with beam.Pipeline() as pipeline:
    unique_employees = (
        pipeline
        | 'Create PCollection' >> beam.Create(employee_data)
        | 'Key by Record Content' >> beam.Map(lambda record: (tuple(sorted(record.items())), record))
        | 'Group by Key' >> beam.GroupByKey()
        | 'Extract Unique Record' >> beam.Map(lambda kv: kv[1][0]) # Take first record for each group
        | 'Print Result' >> beam.Map(print)
    )

{'id': 1, 'first_name': 'Amit', 'last_name': 'Sharma', 'age': 28, 'salary': 60000, 'department': 'IT'}
{'id': 2, 'first_name': 'Priya', 'last_name': 'Mehta', 'age': 32, 'salary': 70000, 'department': 'HR'}
{'id': 3, 'first_name': 'Rahul', 'last_name': 'Verma', 'age': 26, 'salary': None, 'department': 'Sales'}
{'id': 4, 'first_name': 'Sneha', 'last_name': 'Kumar', 'age': 29, 'salary': 75000, 'department': 'Finance'}
{'id': 5, 'first_name': 'Vikram', 'last_name': 'Singh', 'age': 35, 'salary': 50000, 'department': 'IT'}
{'id': 6, 'first_name': 'Neha', 'last_name': 'Patel', 'age': 30, 'salary': 80000, 'department': 'HR'}
{'id': 7, 'first_name': 'Rohan', 'last_name': 'Nair', 'age': 27, 'salary': 55000, 'department': 'IT'}
{'id': 8, 'first_name': 'Pooja', 'last_name': 'Joshi', 'age': 31, 'salary': None, 'department': 'Sales'}
{'id': 9, 'first_name': 'Karan', 'last_name': 'Kapoor', 'age': 40, 'salary': 95000, 'department': 'Finance'}
{'id': 10, 'first_name': 'Divya', 'last_name': 'Reddy', 'age': 24, 'salary': 62000, 'department': 'HR'}
{'id': 11, 'first_name': 'Ankit', 'last_name': 'Gupta', 'age': 29, 'salary': 67000, 'department': 'IT'}
{'id': 12, 'first_name': 'Meera', 'last_name': 'Saxena', 'age': 34, 'salary': 72000, 'department': 'Finance'}
{'id': 13, 'first_name': 'Sahil', 'last_name': 'Bajaj', 'age': 28, 'salary': 63000, 'department': 'Sales'}
{'id': 14, 'first_name': 'Anjali', 'last_name': 'Mishra', 'age': 33, 'salary': None, 'department': 'HR'}
{'id': 15, 'first_name': 'Deepak', 'last_name': 'Choudhary', 'age': 36, 'salary': 58000, 'department': 'Finance'}
{'id': 16, 'first_name': 'Swati', 'last_name': 'Agarwal', 'age': 25, 'salary': 51000, 'department': 'Sales'}
{'id': 17, 'first_name': 'Manish', 'last_name': 'Pandey', 'age': 38, 'salary': 91000, 'department': 'IT'}
{'id': 18, 'first_name': 'Kavya', 'last_name': 'Desai', 'age': 26, 'salary': 64000, 'department': 'HR'}
{'id': 19, 'first_name': 'Nikhil', 'last_name': 'Dubey', 'age': 37, 'salary': 85000, 'department': 'Finance'}
{'id': 20, 'first_name': 'Ritika', 'last_name': 'Bose', 'age': 29, 'salary': 69000, 'department': 'IT'}
```

Cell [27] contains the following code:

```
[27]: # 3. Word count (on empName)
with beam.Pipeline() as pipeline:
    word_counts = (
        pipeline
        | 'Create Names' >> beam.Create(employee_data)
        | 'Extract Names' >> beam.FlatMap(lambda x: [x['first_name'], x['last_name']])
    )
```

How do you perform word count using Apache Beam?

The screenshot shows a Google Colab notebook with the following code in cell [27]:

```
[27]: # 3. Word count (on empName)
with beam.Pipeline() as pipeline:
    word_counts = (
        pipeline
        | 'Create Names' >> beam.Create(employee_data)
        | 'Extract Names' >> beam.FlatMap(lambda x: [x['first_name'], x['last_name']])
        | 'To Lowercase' >> beam.Map(lambda word: word.lower())
        | 'Pair With 1' >> beam.Map(lambda word: (word, 1))
        | 'Count Words' >> beam.CombinePerKey(sum)
        | 'Print Word Counts' >> beam.Map(print)
    )

('amit', 1)
('sharma', 1)
('priya', 1)
('mehta', 1)
('rahu', 1)
('verma', 1)
('sneha', 1)
('kumar', 1)
('vikram', 2)
('singh', 2)
('neha', 1)
('patel', 1)
('rohan', 1)
('nair', 1)
('pooja', 1)
('joshi', 1)
('karan', 2)
('kapoor', 2)
('divya', 1)
('reddy', 1)
('ankit', 1)
('gupta', 1)
('meera', 1)
('saxena', 1)
('sahil', 1)
('bajaj', 1)
```

Write an Apache Beam pipeline to group records by a specific field and calculate the average value for each group.

```
[28]: # 4. Group by department and calculate average salary
def filter_non_null_salaries(emp):
    return emp['salary'] is not None

with beam.Pipeline() as pipeline:
    avg_salary = (
        pipeline
        | 'Create Employee Data' >> beam.Create(employee_data)
        | 'Filter Null Salaries' >> beam.Filter(filter_non_null_salaries)
        | 'Map to (dept, (salary, 1))' >> beam.Map(lambda emp: (emp['department'], (emp['salary'], 1)))
        | 'Sum Salaries and Counts' >> beam.CombinePerKey(lambda vals: (sum(s for s, _ in vals), sum(c for _, c in vals)))
        | 'Calculate Average' >> beam.Map(lambda kv: (kv[0], kv[1][0] / kv[1][1]))
        | 'Print Result' >> beam.Map(print)
    )

('IT', 63142.857142857145)
('HR', 69000.0)
('Finance', 80000.0)
('Sales', 57000.0)

[29]: # 5. Handle missing/null values
with beam.Pipeline() as pipeline:
    cleaned = (
        pipeline
        | 'Create Employee Data' >> beam.Create(employee_data)
        | 'Remove Null Salaries' >> beam.Filter(lambda emp: emp['salary'] is not None)
        | 'Print Result' >> beam.Map(print)
    )

{'id': 1, 'first_name': 'Amit', 'last_name': 'Sharma', 'age': 28, 'salary': 60000, 'department': 'IT'}
{'id': 2, 'first_name': 'Priya', 'last_name': 'Mehta', 'age': 32, 'salary': 70000, 'department': 'HR'}
{'id': 4, 'first_name': 'Sneha', 'last_name': 'Kumar', 'age': 29, 'salary': 75000, 'department': 'Finance'}
{'id': 5, 'first_name': 'Vikram', 'last_name': 'Singh', 'age': 35, 'salary': 50000, 'department': 'IT'}
{'id': 6, 'first_name': 'Neha', 'last_name': 'Patel', 'age': 30, 'salary': 80000, 'department': 'HR'}
{'id': 7, 'first_name': 'Rohan', 'last_name': 'Nair', 'age': 27, 'salary': 55000, 'department': 'IT'}
{'id': 9, 'first_name': 'Karan', 'last_name': 'Kapoor', 'age': 40, 'salary': 95000, 'department': 'Finance'}
{'id': 10, 'first_name': 'Divya', 'last_name': 'Reddy', 'age': 24, 'salary': 62000, 'department': 'HR'}
{'id': 11, 'first_name': 'Ankit', 'last_name': 'Gupta', 'age': 29, 'salary': 67000, 'department': 'IT'}
```

How do you handle missing or null values in an Apache Beam pipeline?

```
[29]: # 5. Handle missing/null values
with beam.Pipeline() as pipeline:
    cleaned = (
        pipeline
        | 'Create Employee Data' >> beam.Create(employee_data)
        | 'Remove Null Salaries' >> beam.Filter(lambda emp: emp['salary'] is not None)
        | 'Print Result' >> beam.Map(print)
    )

{'id': 1, 'first_name': 'Amit', 'last_name': 'Sharma', 'age': 28, 'salary': 60000, 'department': 'IT'}
{'id': 2, 'first_name': 'Priya', 'last_name': 'Mehta', 'age': 32, 'salary': 70000, 'department': 'HR'}
{'id': 4, 'first_name': 'Sneha', 'last_name': 'Kumar', 'age': 29, 'salary': 75000, 'department': 'Finance'}
{'id': 5, 'first_name': 'Vikram', 'last_name': 'Singh', 'age': 35, 'salary': 50000, 'department': 'IT'}
{'id': 6, 'first_name': 'Neha', 'last_name': 'Patel', 'age': 30, 'salary': 80000, 'department': 'HR'}
{'id': 7, 'first_name': 'Rohan', 'last_name': 'Nair', 'age': 27, 'salary': 55000, 'department': 'IT'}
{'id': 9, 'first_name': 'Karan', 'last_name': 'Kapoor', 'age': 40, 'salary': 95000, 'department': 'Finance'}
{'id': 10, 'first_name': 'Divya', 'last_name': 'Reddy', 'age': 24, 'salary': 62000, 'department': 'HR'}
{'id': 11, 'first_name': 'Ankit', 'last_name': 'Gupta', 'age': 29, 'salary': 67000, 'department': 'IT'}
{'id': 12, 'first_name': 'Meera', 'last_name': 'Saxena', 'age': 34, 'salary': 72000, 'department': 'Finance'}
{'id': 13, 'first_name': 'Sahil', 'last_name': 'Bajaj', 'age': 28, 'salary': 63000, 'department': 'Sales'}
{'id': 15, 'first_name': 'Deepak', 'last_name': 'Choudhary', 'age': 36, 'salary': 58000, 'department': 'Finance'}
{'id': 16, 'first_name': 'Swati', 'last_name': 'Agarwal', 'age': 25, 'salary': 51000, 'department': 'Sales'}
{'id': 17, 'first_name': 'Manish', 'last_name': 'Pandey', 'age': 38, 'salary': 91000, 'department': 'IT'}
{'id': 18, 'first_name': 'Kavya', 'last_name': 'Desai', 'age': 26, 'salary': 64000, 'department': 'HR'}
{'id': 19, 'first_name': 'Nikhil', 'last_name': 'Dubey', 'age': 37, 'salary': 85000, 'department': 'Finance'}
{'id': 20, 'first_name': 'Ritika', 'last_name': 'Bose', 'age': 29, 'salary': 69000, 'department': 'IT'}
{'id': 9, 'first_name': 'Karan', 'last_name': 'Kapoor', 'age': 40, 'salary': 95000, 'department': 'Finance'}
{'id': 5, 'first_name': 'Vikram', 'last_name': 'Singh', 'age': 35, 'salary': 50000, 'department': 'IT'}

[30]: # 6. Count distinct values in department
with beam.Pipeline() as pipeline:
    distinct_departments = (
        pipeline
        | 'Create Employee Data' >> beam.Create(employee_data)
        | 'Extract Departments' >> beam.Map(lambda emp: emp['department'])
        | 'Get Unique' >> beam.Distinct()
        | 'Print Unique Departments' >> beam.Map(print)
    )
```

Write an Apache Beam pipeline to count distinct values in a specific field of a dataset.

The screenshot shows a Google Colab notebook with the following code in cell [30]:

```
[30]: # 6. Count distinct values in department
with beam.Pipeline() as pipeline:
    distinct_departments = (
        pipeline
        | 'Create Employee Data' >> beam.Create(employee_data)
        | 'Extract Departments' >> beam.Map(lambda emp: emp['department'])
        | 'Get Unique' >> beam.Distinct()
        | 'Print Unique Departments' >> beam.Map(print)
    )

IT
HR
Sales
Finance
```

The notebook also displays a list of employee records below the code:

```
{'id': 1, 'first_name': 'Amit', 'last_name': 'Sharma', 'age': 28, 'salary': 60000, 'department': 'IT'}
{'id': 2, 'first_name': 'Priya', 'last_name': 'Mehta', 'age': 32, 'salary': 70000, 'department': 'HR'}
{'id': 4, 'first_name': 'Sneha', 'last_name': 'Kumar', 'age': 29, 'salary': 75000, 'department': 'Finance'}
{'id': 6, 'first_name': 'Neha', 'last_name': 'Patel', 'age': 30, 'salary': 80000, 'department': 'HR'}
{'id': 7, 'first_name': 'Rohan', 'last_name': 'Nair', 'age': 27, 'salary': 55000, 'department': 'IT'}
{'id': 9, 'first_name': 'Karan', 'last_name': 'Kapoor', 'age': 40, 'salary': 95000, 'department': 'Finance'}
{'id': 10, 'first_name': 'Divya', 'last_name': 'Reddy', 'age': 24, 'salary': 62000, 'department': 'HR'}
{'id': 11, 'first_name': 'Ankit', 'last_name': 'Gupta', 'age': 29, 'salary': 67000, 'department': 'IT'}
{'id': 12, 'first_name': 'Meera', 'last_name': 'Saxena', 'age': 34, 'salary': 72000, 'department': 'Finance'}
{'id': 13, 'first_name': 'Sahil', 'last_name': 'Bajaj', 'age': 28, 'salary': 63000, 'department': 'Sales'}
{'id': 15, 'first_name': 'Deepak', 'last_name': 'Choudhary', 'age': 36, 'salary': 58000, 'department': 'Finance'}
{'id': 16, 'first_name': 'Swati', 'last_name': 'Agarwal', 'age': 25, 'salary': 51000, 'department': 'Sales'}
{'id': 17, 'first_name': 'Manish', 'last_name': 'Pandey', 'age': 38, 'salary': 91000, 'department': 'IT'}
{'id': 18, 'first_name': 'Kavya', 'last_name': 'Desai', 'age': 26, 'salary': 64000, 'department': 'HR'}
```

Given a PCollection of records, write an Apache Beam pipeline to filter records where the salary is greater than 50,000.

The screenshot shows a Google Colab notebook with the following code in cell [31]:

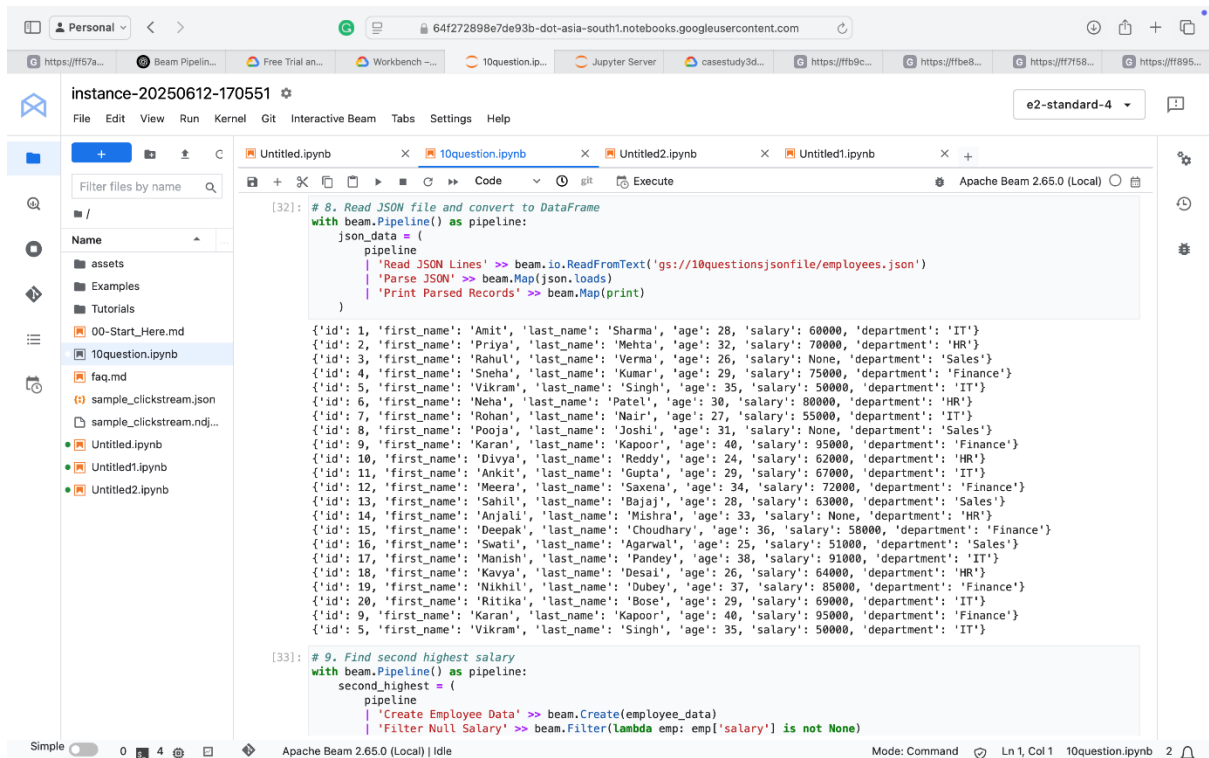
```
[31]: # 7. Filter records where salary > 50000
with beam.Pipeline() as pipeline:
    high_earners = (
        pipeline
        | 'Create Employee Data' >> beam.Create(employee_data)
        | 'Filter Salary > 50000' >> beam.Filter(lambda emp: emp['salary'] is not None and emp['salary'] > 50000)
        | 'Print High Earners' >> beam.Map(print)
    )

{'id': 1, 'first_name': 'Amit', 'last_name': 'Sharma', 'age': 28, 'salary': 60000, 'department': 'IT'}
{'id': 2, 'first_name': 'Priya', 'last_name': 'Mehta', 'age': 32, 'salary': 70000, 'department': 'HR'}
{'id': 4, 'first_name': 'Sneha', 'last_name': 'Kumar', 'age': 29, 'salary': 75000, 'department': 'Finance'}
{'id': 6, 'first_name': 'Neha', 'last_name': 'Patel', 'age': 30, 'salary': 80000, 'department': 'HR'}
{'id': 7, 'first_name': 'Rohan', 'last_name': 'Nair', 'age': 27, 'salary': 55000, 'department': 'IT'}
{'id': 9, 'first_name': 'Karan', 'last_name': 'Kapoor', 'age': 40, 'salary': 95000, 'department': 'Finance'}
{'id': 10, 'first_name': 'Divya', 'last_name': 'Reddy', 'age': 24, 'salary': 62000, 'department': 'HR'}
{'id': 11, 'first_name': 'Ankit', 'last_name': 'Gupta', 'age': 29, 'salary': 67000, 'department': 'IT'}
{'id': 12, 'first_name': 'Meera', 'last_name': 'Saxena', 'age': 34, 'salary': 72000, 'department': 'Finance'}
{'id': 13, 'first_name': 'Sahil', 'last_name': 'Bajaj', 'age': 28, 'salary': 63000, 'department': 'Sales'}
{'id': 15, 'first_name': 'Deepak', 'last_name': 'Choudhary', 'age': 36, 'salary': 58000, 'department': 'Finance'}
{'id': 16, 'first_name': 'Swati', 'last_name': 'Agarwal', 'age': 25, 'salary': 51000, 'department': 'Sales'}
{'id': 17, 'first_name': 'Manish', 'last_name': 'Pandey', 'age': 38, 'salary': 91000, 'department': 'IT'}
{'id': 18, 'first_name': 'Kavya', 'last_name': 'Desai', 'age': 26, 'salary': 64000, 'department': 'HR'}
```

The notebook also displays a list of employee records below the code:

```
{'id': 1, 'first_name': 'Amit', 'last_name': 'Sharma', 'age': 28, 'salary': 60000, 'department': 'IT'}
{'id': 2, 'first_name': 'Priya', 'last_name': 'Mehta', 'age': 32, 'salary': 70000, 'department': 'HR'}
```

Write an Apache Beam pipeline to read a JSON file and convert it into a PCollection of structured records.



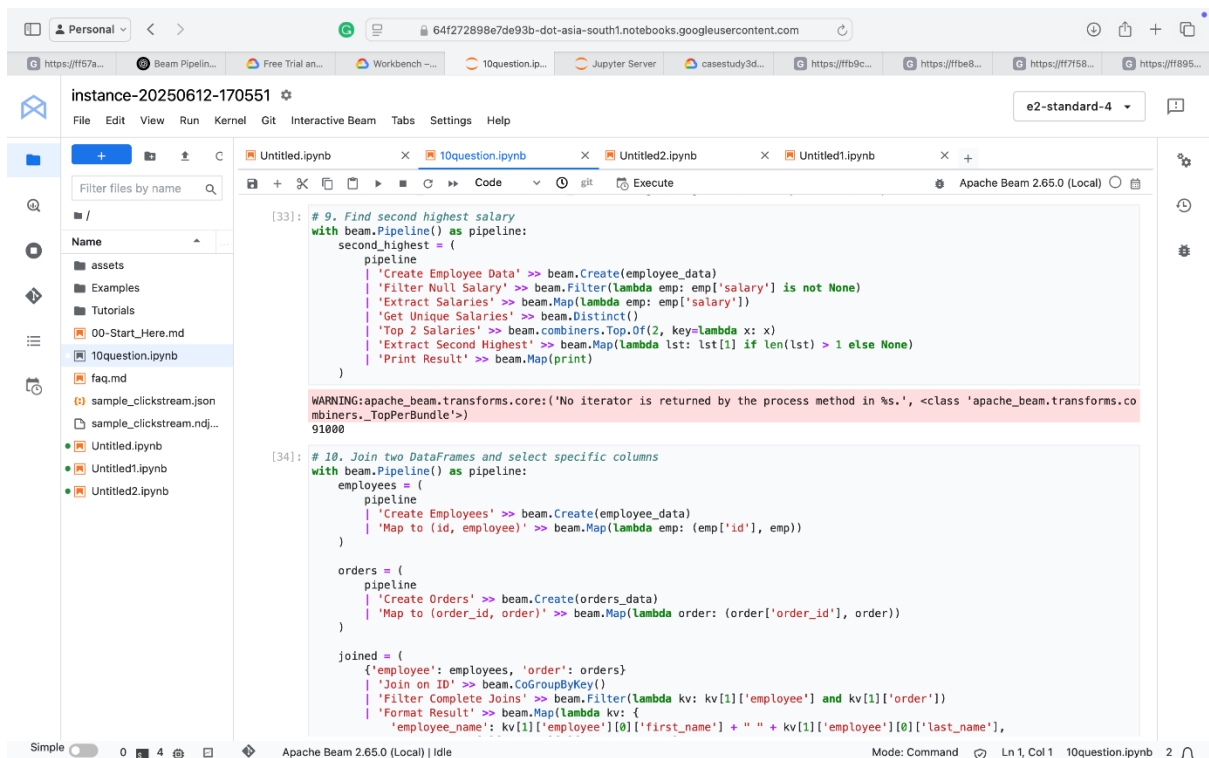
The screenshot shows a Jupyter Notebook interface with a file explorer on the left and a code editor on the right. The file explorer shows a directory with files like '00-Start_Here.md', '10question.ipynb', 'faq.md', 'sample_clickstream.json', 'sample_clickstream.ndj...', and several 'Untitled' files. The code editor shows the following code:

```
[32]: # 8. Read JSON file and convert to DataFrame
with beam.Pipeline() as pipeline:
    json_data = (
        pipeline
        | 'Read JSON Lines' >> beam.io.ReadFromText('gs://10questionsjsonfile/employees.json')
        | 'Parse JSON' >> beam.Map(json.loads)
        | 'Print Parsed Records' >> beam.Map(print)
    )

{'id': 1, 'first_name': 'Amit', 'last_name': 'Sharma', 'age': 28, 'salary': 60000, 'department': 'IT'}
{'id': 2, 'first_name': 'Priya', 'last_name': 'Mehta', 'age': 32, 'salary': 70000, 'department': 'HR'}
{'id': 3, 'first_name': 'Rahul', 'last_name': 'Verma', 'age': 26, 'salary': None, 'department': 'Sales'}
{'id': 4, 'first_name': 'Sneha', 'last_name': 'Kumar', 'age': 29, 'salary': 75000, 'department': 'Finance'}
{'id': 5, 'first_name': 'Vikram', 'last_name': 'Singh', 'age': 35, 'salary': 50000, 'department': 'IT'}
{'id': 6, 'first_name': 'Neha', 'last_name': 'Patel', 'age': 30, 'salary': 80000, 'department': 'HR'}
{'id': 7, 'first_name': 'Rohan', 'last_name': 'Nair', 'age': 27, 'salary': 55000, 'department': 'IT'}
{'id': 8, 'first_name': 'Pooja', 'last_name': 'Joshi', 'age': 31, 'salary': None, 'department': 'Sales'}
{'id': 9, 'first_name': 'Karan', 'last_name': 'Kapoor', 'age': 40, 'salary': 95000, 'department': 'Finance'}
{'id': 10, 'first_name': 'Divya', 'last_name': 'Reddy', 'age': 24, 'salary': 62000, 'department': 'HR'}
{'id': 11, 'first_name': 'Ankit', 'last_name': 'Gupta', 'age': 29, 'salary': 67000, 'department': 'IT'}
{'id': 12, 'first_name': 'Meera', 'last_name': 'Saxena', 'age': 34, 'salary': 72000, 'department': 'Finance'}
{'id': 13, 'first_name': 'Sahil', 'last_name': 'Bajaj', 'age': 28, 'salary': 63000, 'department': 'Sales'}
{'id': 14, 'first_name': 'Anjali', 'last_name': 'Mishra', 'age': 33, 'salary': None, 'department': 'HR'}
{'id': 15, 'first_name': 'Deepak', 'last_name': 'Choudhary', 'age': 36, 'salary': 58000, 'department': 'Finance'}
{'id': 16, 'first_name': 'Swati', 'last_name': 'Agarwal', 'age': 25, 'salary': 51000, 'department': 'Sales'}
{'id': 17, 'first_name': 'Manish', 'last_name': 'Pandey', 'age': 38, 'salary': 91000, 'department': 'IT'}
{'id': 18, 'first_name': 'Kavya', 'last_name': 'Desai', 'age': 26, 'salary': 64000, 'department': 'HR'}
{'id': 19, 'first_name': 'Nikhil', 'last_name': 'Dubey', 'age': 37, 'salary': 85000, 'department': 'Finance'}
{'id': 20, 'first_name': 'Ritika', 'last_name': 'Bose', 'age': 29, 'salary': 69000, 'department': 'IT'}
{'id': 9, 'first_name': 'Karan', 'last_name': 'Kapoor', 'age': 40, 'salary': 95000, 'department': 'Finance'}
{'id': 5, 'first_name': 'Vikram', 'last_name': 'Singh', 'age': 35, 'salary': 50000, 'department': 'IT'}
```

```
[33]: # 9. Find second highest salary
with beam.Pipeline() as pipeline:
    second_highest = (
        pipeline
        | 'Create Employee Data' >> beam.Create(employee_data)
        | 'Filter Null Salary' >> beam.Filter(lambda emp: emp['salary'] is not None)
    )
```

Write an Apache Beam pipeline to find the second highest salary from an employee dataset.



The screenshot shows a Jupyter Notebook interface with a file explorer on the left and a code editor on the right. The file explorer shows a directory with files like '00-Start_Here.md', '10question.ipynb', 'faq.md', 'sample_clickstream.json', 'sample_clickstream.ndj...', and several 'Untitled' files. The code editor shows the following code:

```
[33]: # 9. Find second highest salary
with beam.Pipeline() as pipeline:
    second_highest = (
        pipeline
        | 'Create Employee Data' >> beam.Create(employee_data)
        | 'Filter Null Salary' >> beam.Filter(lambda emp: emp['salary'] is not None)
        | 'Extract Salaries' >> beam.Map(lambda emp: emp['salary'])
        | 'Get Unique Salaries' >> beam.Distinct()
        | 'Top 2 Salaries' >> beam.combiners.TopOf(2, key=lambda x: x)
        | 'Extract Second Highest' >> beam.Map(lambda lst: lst[1] if len(lst) > 1 else None)
        | 'Print Result' >> beam.Map(print)
    )

WARNING:apache_beam.transforms.core:(No iterator is returned by the process method in %s., <class 'apache_beam.transforms.co
mbiners_TopOfBundle'>)
91000

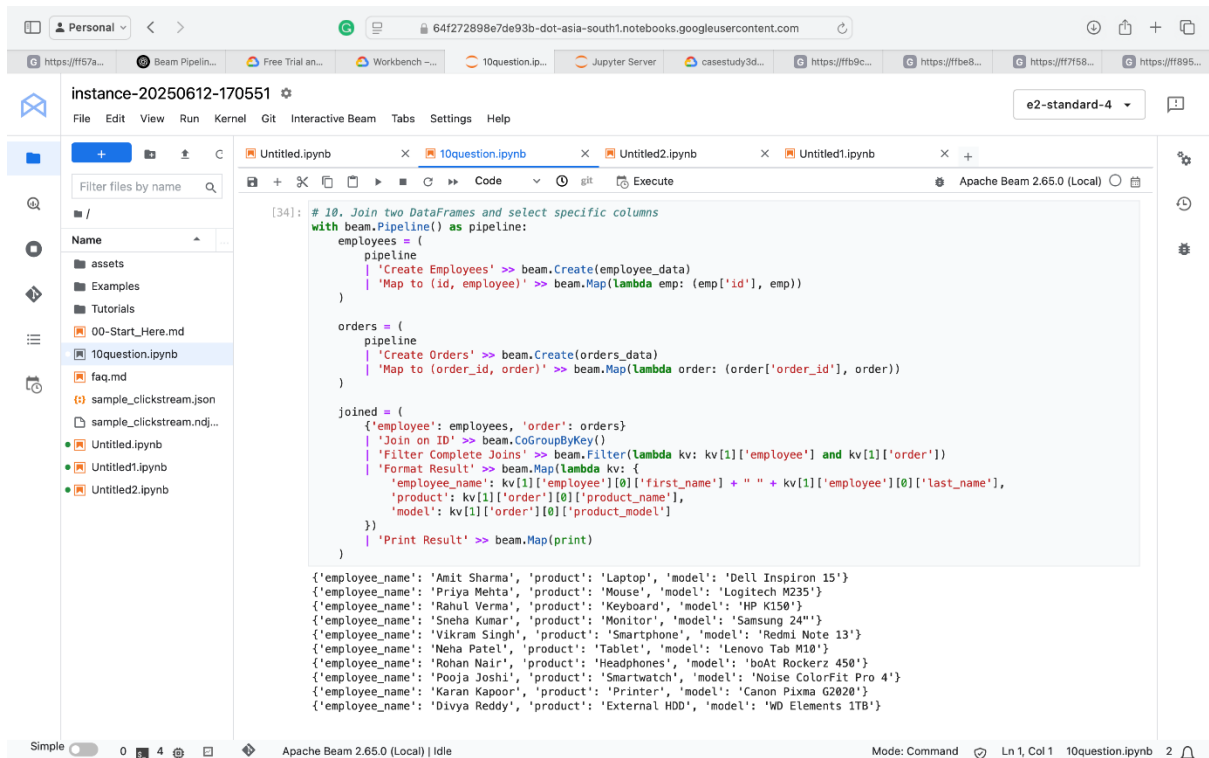
[34]: # 10. Join two DataFrames and select specific columns
with beam.Pipeline() as pipeline:
    employees = (
        pipeline
        | 'Create Employees' >> beam.Create(employee_data)
        | 'Map to (id, employee)' >> beam.Map(lambda emp: (emp['id'], emp))
    )

    orders = (
        pipeline
        | 'Create Orders' >> beam.Create(orders_data)
        | 'Map to (order_id, order)' >> beam.Map(lambda order: (order['order_id'], order))
    )

    joined = (
        {'employee': employees, 'order': orders}
        | 'Join on ID' >> beam.CoGroupByKey()
        | 'Filter Complete Joins' >> beam.Filter(lambda kv: kv[1]['employee'] and kv[1]['order'])
        | 'Format Result' >> beam.Map(lambda kv: {
            'employee_name': kv[1]['employee'][0]['first_name'] + " " + kv[1]['employee'][0]['last_name'],

```


Write an Apache Beam pipeline to join two PCollections and select specific fields from the result.



The screenshot shows a Google Colab notebook interface. The browser address bar displays the URL: `64f272898e7de93b-dot-asia-south1.notebooks.googleusercontent.com`. The notebook is titled "instance-20250612-170551". The left sidebar shows a file explorer with a search bar and a list of files including "assets", "Examples", "Tutorials", "00-Start_Here.md", "10question.ipynb", "faq.md", "sample_clickstream.json", "sample_clickstream.ndj...", "Untitled.ipynb", "Untitled1.ipynb", and "Untitled2.ipynb". The main editor area shows a Jupyter notebook cell with the following code:

```
[34]: # 10. Join two DataFrames and select specific columns
with beam.Pipeline() as pipeline:
    employees = (
        pipeline
        | 'Create Employees' >> beam.Create(employee_data)
        | 'Map to (id, employee)' >> beam.Map(lambda emp: (emp['id'], emp))
    )

    orders = (
        pipeline
        | 'Create Orders' >> beam.Create(orders_data)
        | 'Map to (order_id, order)' >> beam.Map(lambda order: (order['order_id'], order))
    )

    joined = (
        {'employee': employees, 'order': orders}
        | 'Join on ID' >> beam.CoGroupByKey()
        | 'Filter Complete Joins' >> beam.Filter(lambda kv: kv[1]['employee'] and kv[1]['order'])
        | 'Format Result' >> beam.Map(lambda kv: {
            'employee_name': kv[1]['employee'][0]['first_name'] + " " + kv[1]['employee'][0]['last_name'],
            'product': kv[1]['order'][0]['product_name'],
            'model': kv[1]['order'][0]['product_model']
        })
        | 'Print Result' >> beam.Map(print)
    )

{'employee_name': 'Amit Sharma', 'product': 'Laptop', 'model': 'Dell Inspiron 15'}
{'employee_name': 'Priya Mehta', 'product': 'Mouse', 'model': 'Logitech M235'}
{'employee_name': 'Rahul Verma', 'product': 'Keyboard', 'model': 'HP K150'}
{'employee_name': 'Sneha Kumar', 'product': 'Monitor', 'model': 'Samsung 24''}
{'employee_name': 'Vikram Singh', 'product': 'Smartphone', 'model': 'Redmi Note 13'}
{'employee_name': 'Neha Patel', 'product': 'Tablet', 'model': 'Lenovo Tab M10'}
{'employee_name': 'Rohan Nair', 'product': 'Headphones', 'model': 'boAt Rockerz 450'}
{'employee_name': 'Pooja Joshi', 'product': 'Smartwatch', 'model': 'Noise ColorFit Pro 4'}
{'employee_name': 'Karan Kapoor', 'product': 'Printer', 'model': 'Canon Pixma G2020'}
{'employee_name': 'Divya Reddy', 'product': 'External HDD', 'model': 'WD Elements 1TB'}
```

The bottom status bar shows "Simple" mode, "Apache Beam 2.65.0 (Local) | Idle", "Mode: Command", "Ln 1, Col 1", "10question.ipynb", and a page number "2".