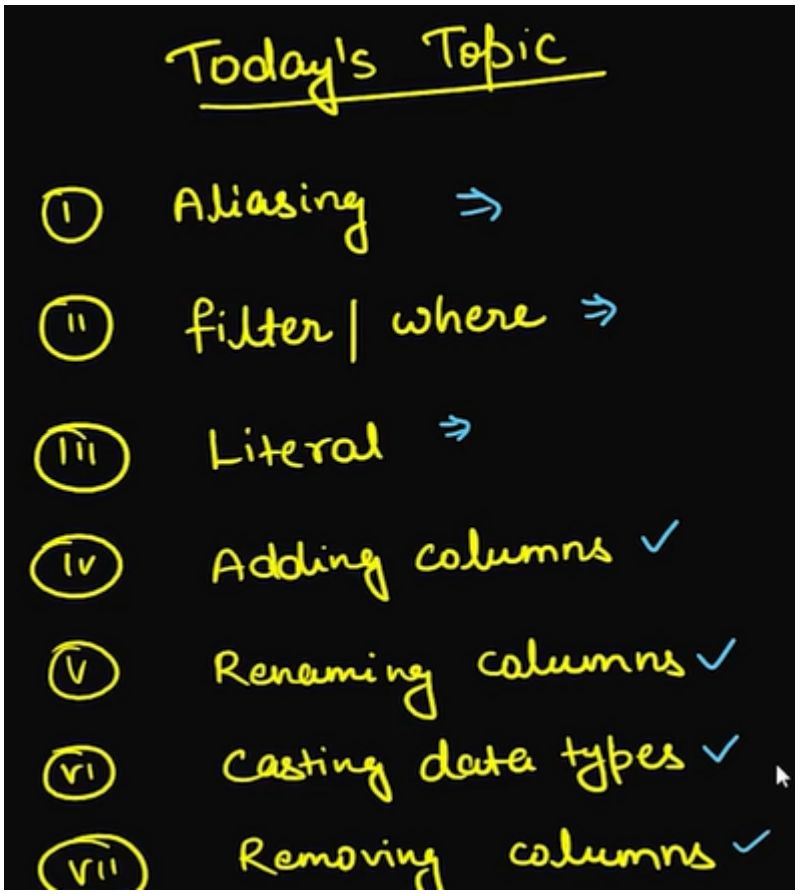
Data Transformation part-2



1. Today we'll be learning
2. Lets do this in practical
3. So to change a column name we use alias

```
employee_df.select(col("id").alias("employee_id"),"name","age").show()
```

4. Filter and where
5. In Apache Spark, there's virtually no difference between `where` and `filter` for filtering DataFrames. They both achieve the same outcome: creating a new DataFrame containing rows that meet your specified conditions.

6. So if we want to retrieve rows where salary > $150000..we use

```
employee_df.filter(col("salary")>150000).show()
```

▸ (1) Spark Jobs

```
+---+--------+---+------+-------+--------+
| id|    name|age|salary|address| nominee|
+---+--------+---+------+-------+--------+
|  4|Prantosh| 17|200000|Kolkata|   India|
|  5|  Vikash| 31|300000|   null|nominee5|
+---+--------+---+------+-------+--------+
```

7. So if we want to retrieve rows where salary > $150000 and age < 18..we use

```
employee_df.filter((col("salary")>150000) & (col("age")<18)).show()
```

▸ (1) Spark Jobs

```
+---+--------+---+------+-------+-------+
| id|    name|age|salary|address|nominee|
+---+--------+---+------+-------+-------+
|  4|Prantosh| 17|200000|Kolkata|  India|
+---+--------+---+------+-------+-------+
```

8.
9. Here we have use brackets for each column
10. Literal
11. It creates a new column and gives a dummy value(which is assigned in code)

```
employee_df.select("*", lit("kumar").alias("last_name")).show()
```

▸ (1) Spark Jobs

```
+---+--------+---+------+-----------+--------+---------+
| id|    name|age|salary|    address| nominee|last_name|
+---+--------+---+------+-----------+--------+---------+
|  1|  Manish| 26| 75000|      bihar|nominee1|    kumar|
|  2|  Nikita| 23|100000|uttarpradesh|nominee2|    kumar|
|  3|  Pritam| 22|150000|  Bangalore|   India|    kumar|
|  4|Prantosh| 17|200000|    Kolkata|   India|    kumar|
|  5|  Vikash| 31|300000|       null|nominee5|    kumar|
+---+--------+---+------+-----------+--------+---------+
```

12. Adding columns

13. We can columns using “withcolumn” ..here in the below code..we used withcloumn and lit

```
employee_df.withColumn("sur_name",lit("singh")).show()
```

▸ (1) Spark Jobs

```
+---+--------+---+------+-----------+--------+--------+
| id|    name|age|salary|    address| nominee|sur_name|
+---+--------+---+------+-----------+--------+--------+
|  1|  Manish| 26| 75000|      bihar|nominee1|   singh|
|  2|  Nikita| 23|100000|uttarpradesh|nominee2|   singh|
|  3|  Pritam| 22|150000|  Bangalore|   India|   singh|
|  4|Prantosh| 17|200000|    Kolkata|   India|   singh|
|  5|  Vikash| 31|300000|       null|nominee5|   singh|
+---+--------+---+------+-----------+--------+--------+
```

14. Rename column

```
employee_df.withColumnRenamed("id","employee_id").show()
```

▸ (1) Spark Jobs

```
+-----------+--------+---+------+-----------+--------+
|employee_id|    name|age|salary|    address| nominee|
+-----------+--------+---+------+-----------+--------+
|          1|  Manish| 26| 75000|      bihar|nominee1|
|          2|  Nikita| 23|100000|uttarpradesh|nominee2|
|          3|  Pritam| 22|150000|  Bangalore|   India|
|          4|Prantosh| 17|200000|    Kolkata|   India|
|          5|  Vikash| 31|300000|       null|nominee5|
```

15.

16. To create a new df..using the output of another query..we use

```
new_employee_df= employee_df.withColumnRenamed("id","employee_id")
```

17. Cast

18.

```
employee_df.withColumn("id",col("id").cast("string")).printSchema()
```

```
employee_df.drop("id",col("name")).show()
```

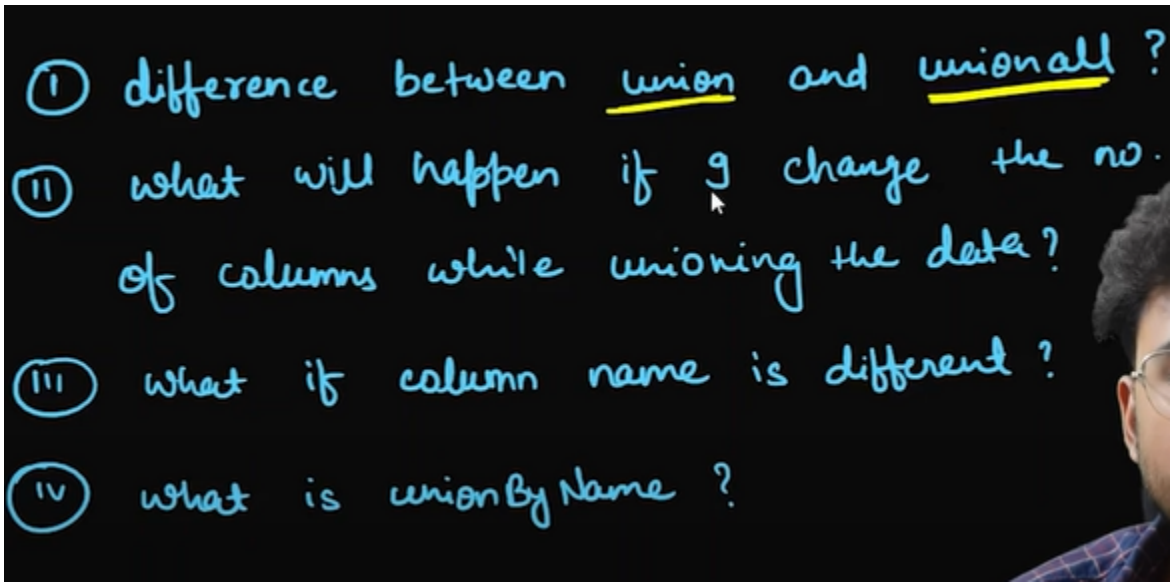
▸ (1) Spark Jobs

```
+---+------+-----------+--------+
|age|salary|    address| nominee|
+---+------+-----------+--------+
| 26| 75000|      bihar|nominee1|
| 23|100000|uttarpradesh|nominee2|
| 22|150000|  Bangalore|   India|
| 17|200000|    Kolkata|   India|
| 31|300000|       null|nominee5|
+---+------+-----------+--------+
```

19. Drop column
20. SparkSQL is same as SQL

Union vs UnionALL

1. Things we are going to discuss



2. Created a manger_df from sample data and schema

```
data=[(10   ,'Anil',50000,  18),
(11 ,'Vikas',75000,     16),
(12 ,'Nisha',40000,     18),
(13 ,'Nidhi',60000,     17),
(14 ,'Priya',80000,     18),
(15 ,'Mohit',45000,     18),
(16 ,'Rajesh',90000,    10),
(17 ,'Raman',55000, 16),
(18 ,'Sam',65000,   17)]

schema=['id','Name','sal','mngr_id']

manager_df= spark.createDataFrame(data=data,schema=schema)
```

3. Created manager_df1

```
data1=[(19  ,'Sohan',50000, 18),
(20 ,'Sima',75000,  17)]

schema1=['id','Name','sal','mngr_id']

manager_df1= spark.createDataFrame(data=data1,schema=schema1)
```

```
manager_df.union(manager_df1).show()
```

▸ (3) Spark Jobs

```
+---+------+-----+--------+
| id|  Name|  sal|mngr_id|
+---+------+-----+--------+
| 10|  Anil|50000|     18|
| 11| Vikas|75000|     16|
| 12| Nisha|40000|     18|
| 13| Nidhi|60000|     17|
| 14| Priya|80000|     18|
| 15| Mohit|45000|     18|
| 16|Rajesh|90000|     10|
| 17| Raman|55000|     16|
| 18|   Sam|65000|     17|
| 19| Sohan|50000|     18|
| 20|  Sima|75000|     17|
```

4. To perform union on these df's ..we use
5. The diff bw union and union-all is I know already
6. But in dataframe there is no difference between union and union-all..but in sql and sparkSQL there's a diff
7.