INDIAN DATA CLUB

CODE BASICS

databricks

14 DAYS AI CHALLENGE

DAY 12

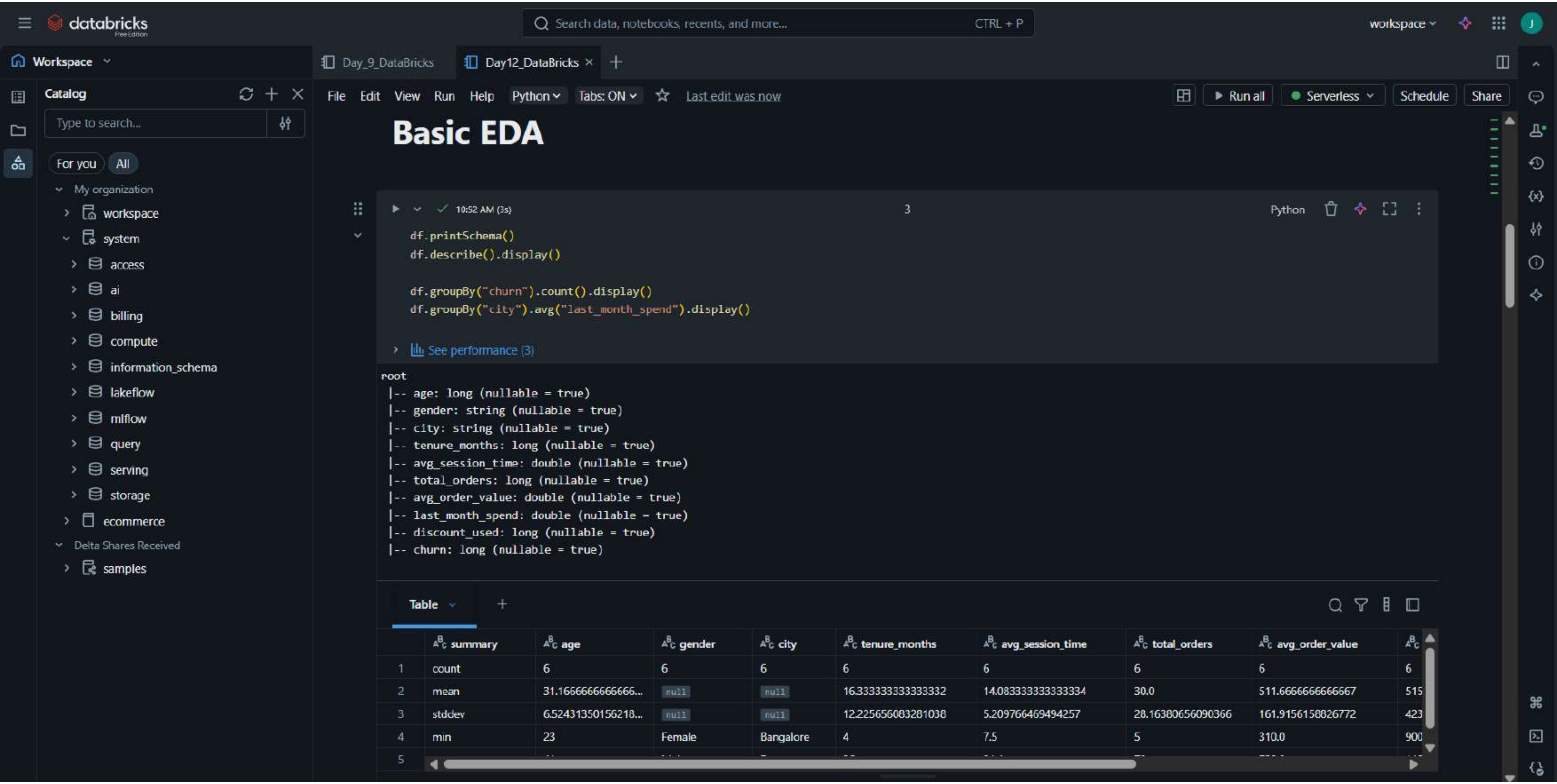
Topic:

MLflow Basics

Challenge:

1. Train simple regression model
2. Log parameters, metrics, model
3. View in MLflow UI
4. Compare runs

#DatabricksWithIDC

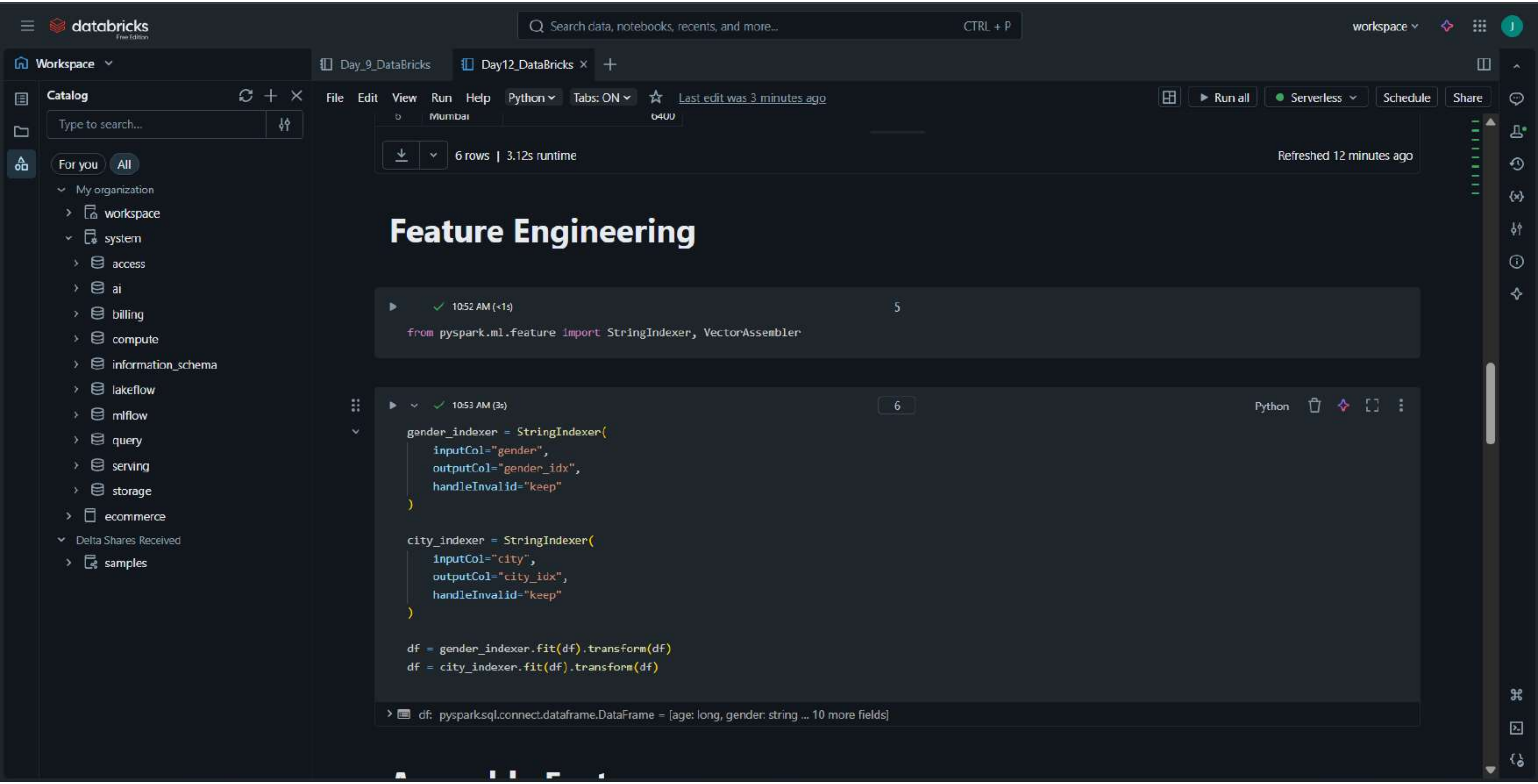


Basic EDA

```
df.printSchema()
df.describe().display()

df.groupBy("churn").count().display()
df.groupBy("city").avg("last_month_spend").display()
```

```
root
 |-- age: long (nullable = true)
 |-- gender: string (nullable = true)
 |-- city: string (nullable = true)
 |-- tenure_months: long (nullable = true)
 |-- avg_session_time: double (nullable = true)
 |-- total_orders: long (nullable = true)
 |-- avg_order_value: double (nullable = true)
 |-- last_month_spend: double (nullable = true)
 |-- discount_used: long (nullable = true)
 |-- churn: long (nullable = true)
```

	summary	age	gender	city	tenure_months	avg_session_time	total_orders	avg_order_value	
1	count	6	6	6	6	6	6	6	6
2	mean	31.1666666666666...	null	null	16.333333333333332	14.083333333333334	30.0	511.6666666666667	515
3	stddev	6.52431350156218...	null	null	12.225656083281038	5.209766469494257	28.16380656090366	161.9156158826772	423
4	min	23	Female	Bangalore	4	7.5	5	310.0	900
5									



```
5    Mumbai    6400
```

6 rows | 3.12s runtime

Refreshed 12 minutes ago

Feature Engineering

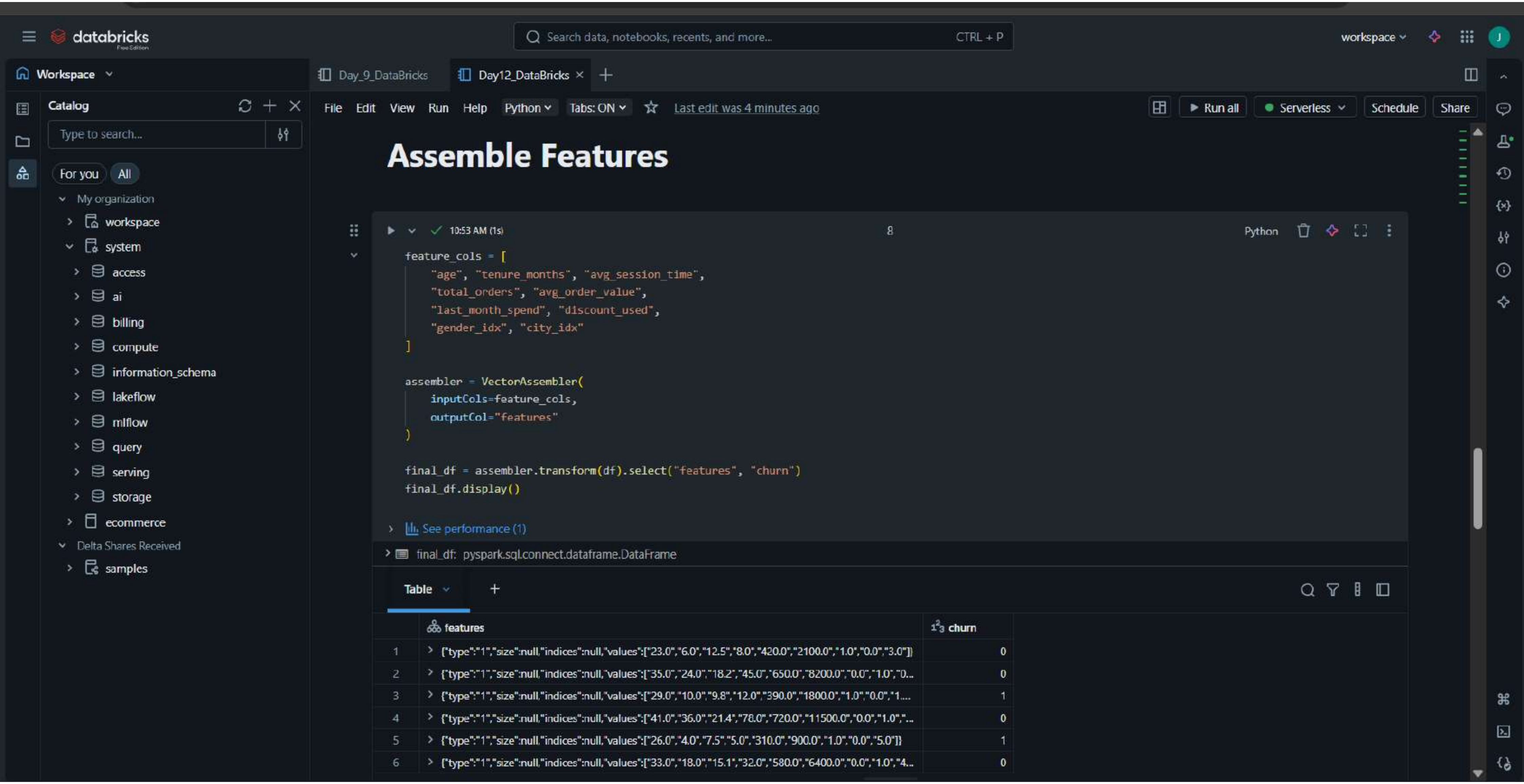
10:52 AM (<1s) 5

```python
from pyspark.ml.feature import StringIndexer, VectorAssembler
```

10:53 AM (3s) 6

```python
gender_indexer = StringIndexer(
    inputCol="gender",
    outputCol="gender_idx",
    handleInvalid="keep"
)

city_indexer = StringIndexer(
    inputCol="city",
    outputCol="city_idx",
    handleInvalid="keep"
)

df = gender_indexer.fit(df).transform(df)
df = city_indexer.fit(df).transform(df)
```

df: pyspark.sql.connect.dataframe.DataFrame = [age: long, gender: string ... 10 more fields]

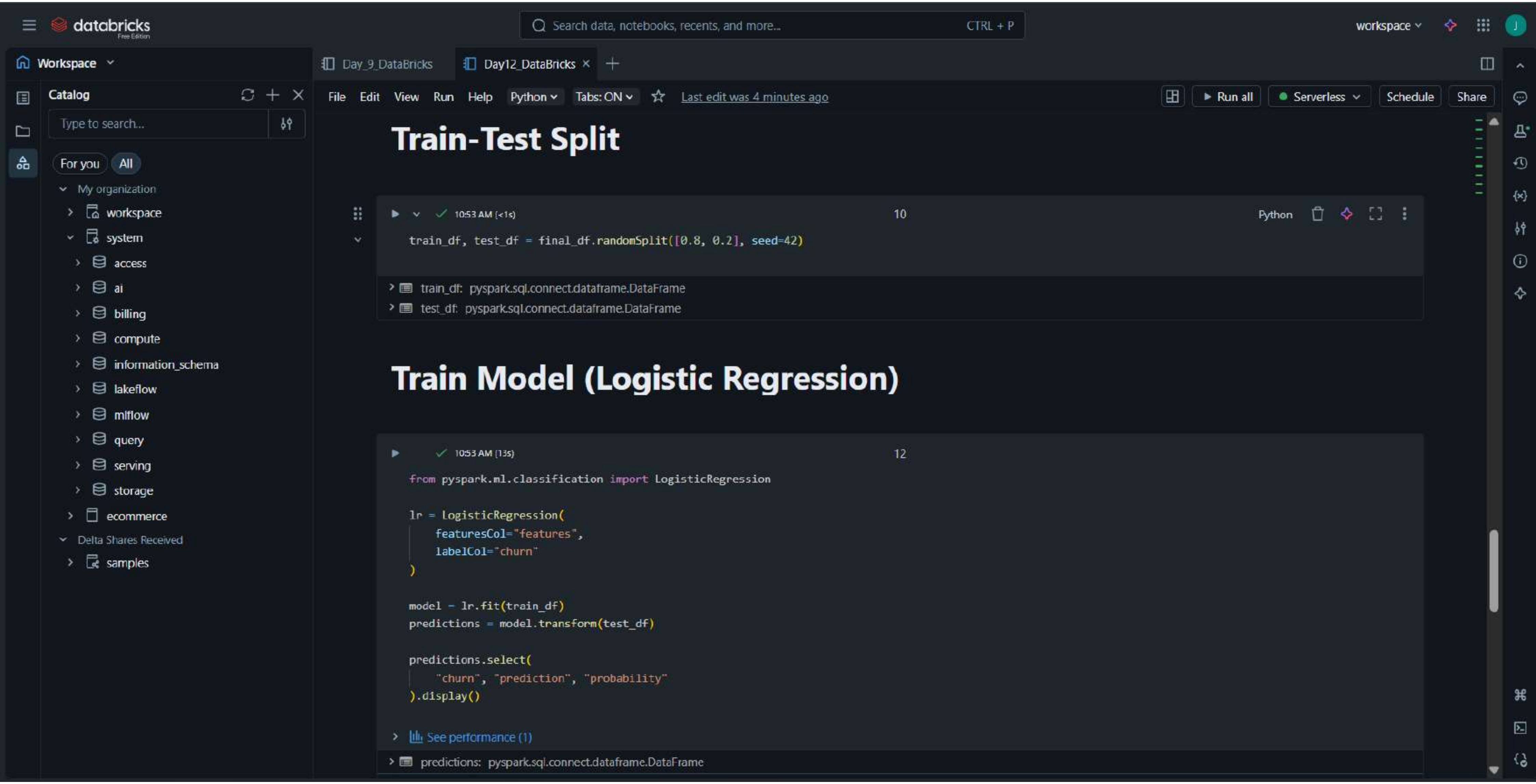


Assemble Features

```
feature_cols = [
    "age", "tenure_months", "avg_session_time",
    "total_orders", "avg_order_value",
    "last_month_spend", "discount_used",
    "gender_idx", "city_idx"
]

assembler = VectorAssembler(
    inputCols=feature_cols,
    outputCol="features"
)

final_df = assembler.transform(df).select("features", "churn")
final_df.display()
```

final_df: pyspark.sql.connect.dataframe.DataFrame

	features	churn
1	{"type":"1","size":null,"indices":null,"values":["23.0","6.0","12.5","8.0","420.0","2100.0","1.0","0.0","3.0"]}	0
2	{"type":"1","size":null,"indices":null,"values":["35.0","24.0","18.2","45.0","650.0","8200.0","0.0","1.0","0...	0
3	{"type":"1","size":null,"indices":null,"values":["29.0","10.0","9.8","12.0","390.0","1800.0","1.0","0.0","1....	1
4	{"type":"1","size":null,"indices":null,"values":["41.0","36.0","21.4","78.0","720.0","11500.0","0.0","1.0","...	0
5	{"type":"1","size":null,"indices":null,"values":["26.0","4.0","7.5","5.0","310.0","900.0","1.0","0.0","5.0"]}	1
6	{"type":"1","size":null,"indices":null,"values":["33.0","18.0","15.1","32.0","580.0","6400.0","0.0","1.0","4...	0



Train-Test Split

10:53 AM (<1s) 10

```
train_df, test_df = final_df.randomSplit([0.8, 0.2], seed=42)
```

```
train_df: pyspark.sql.connect.dataframe.DataFrame
test_df: pyspark.sql.connect.dataframe.DataFrame
```

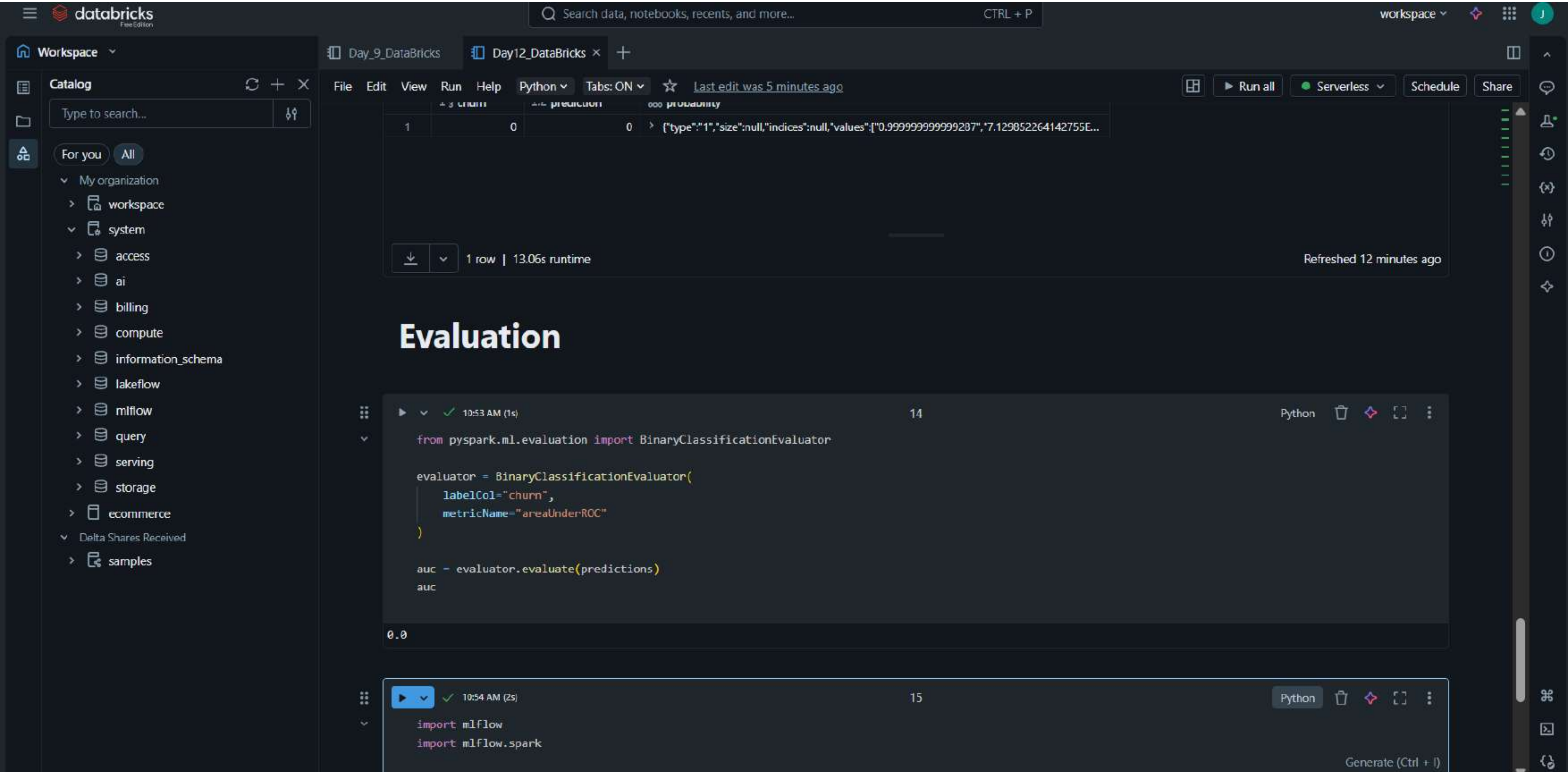
Train Model (Logistic Regression)

10:53 AM (13s) 12

```
from pyspark.ml.classification import LogisticRegression

lr = LogisticRegression(
    featuresCol="features",
    labelCol="churn"
)

model = lr.fit(train_df)
predictions = model.transform(test_df)

predictions.select(
    "churn", "prediction", "probability"
).display()
```

See performance (1)

```
predictions: pyspark.sql.connect.dataframe.DataFrame
```



	churn	prediction	probability
1	0	0	{"type":"1","size":null,"indices":null,"values":["0.999999999999287","7.129852264142755E...

Evaluation

```
from pyspark.ml.evaluation import BinaryClassificationEvaluator

evaluator = BinaryClassificationEvaluator(
    labelCol="churn",
    metricName="areaUnderROC"
)

auc = evaluator.evaluate(predictions)
auc
```

```
0.0
```

```
import mlflow
import mlflow.spark
```