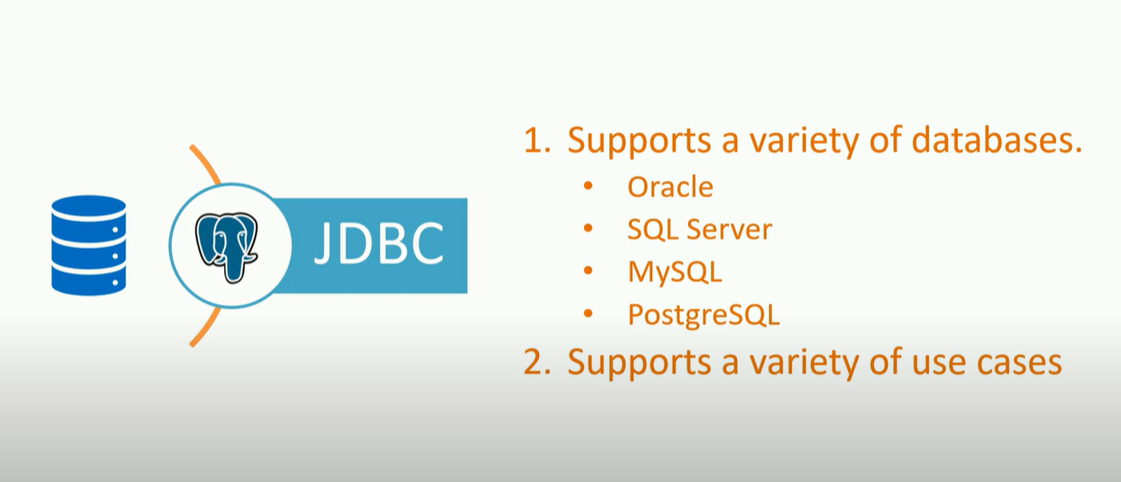
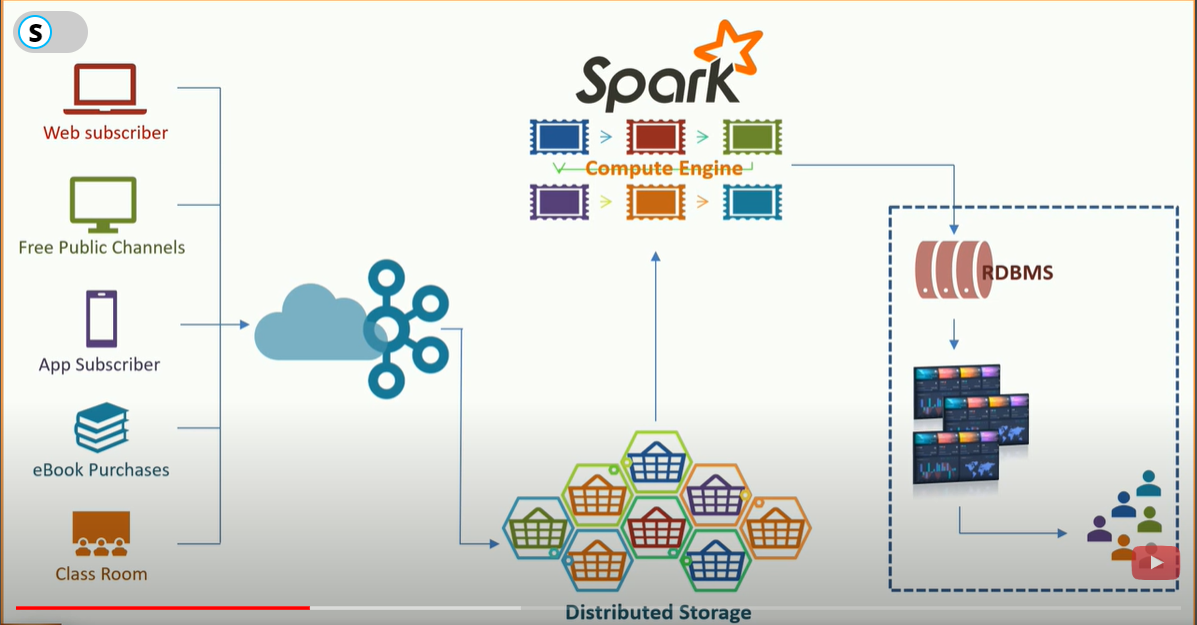
JDBC is most valuable connector for two reasons:





This kind of application makes more sense with an RDBMS as a backend. So, you decided to add an RDBMS. Now, it's apparent that you need to connect to RDBMS from Apache Spark and save your aggregation output in the RDBMS. That's it.That's what makes a Spark JDBC connector a critical thing

Spark JDBC source and sink demo

I hope you understand that you might want to read something into Spark from an RDBMS and you might also need to write something back to a relational database. How to do it?

Configure PostgreSQL server

Before I can use this PostgreSQL system, I need to setup few things.

1. Create a PostgreSQL user
2. Create a database
3. Test PostgreSQLserver connection from a remote machine
4. Let's do that. The first step is to create a user. Use following commands to create a user for yourself.

|  |  |
| --- | --- |
|  | Create a PostgreSQL user  //Read AVRO into Data Frame |
|  | val df = spark.read |
|  | .format("com.databricks.spark.avro") |
|  | .option("mode", "failfast") |
|  | .load("/home/prashant/spark-data/mental-health-in-tech-survey/avro-data/") |
|  |  |
|  | //Write Data Frame to CSV |
|  | df.write |
|  | .format("csv") |
|  | .option("header", "true") |
|  | .option("nullValue", "NA") |
|  | .option("timestampFormat", "yyyy-MM-dd HH:mm:ss") |
|  | .mode("overwrite") |
|  | .save("/home/prashant/spark-data/mental-health-in-tech-survey/csv-data/") |

Create a database (linux commands)

|  |
| --- |
| sudo su postgres |
|  | createuser -- pwprompt -- interactive prashant |
|  | Exit  I need to install a PostgreSQL client tool for this testing. So, let's download and install a client tool.   1. Go to www.postgresql.org 2. Click downloads 3. Select your OS Family 4. Choose the latest PostgreSQL version 5. Pick up the architecture 6. Copy and execute the yum install command. That command will configure your repository. 7. Finally, Install the client using the next yum command.   Great! Now we are ready to test the PostgreSQL remote connection. Use below command to connect to the remote Postgres database.   |  |  | | --- | --- | |  | createdb sparkDB |   You might want to change the IP address with your PostgreSQL server VMs IP address. You can get the IP address of your PostgreSQL server using below command.   |  |  | | --- | --- | |  | psql --host 10.128.0.4 -U prashant --dbnamesparkDB --password |   However, all this exercise has nothing to do with Spark JDBC connectivity. All you need is following information.   1. Your Posrtgre Server IP - 10.128.0.4 2. Database Lister TCP/IP Port - 5432 3. Database name - sparkDB   How to Install Spark JDBC Connector  To use Spark JDBC connector, you need to download the JDBC connector jar and include it in your driver and executer class path. It sounds difficult however it is quite simple to achieve. Use below command to start the spark shell.   |  |  | | --- | --- | |  | //Login to your Postgres server machine as Postgres admin user. | |  | sudo su portgres | |  | //Connect to the database and check hba file location. | |  | psql | |  | show hba\_file; | |  | //Open hba file and add below line at the end. | |  | vi /etc/postgresql/10/main/pg\_hba.conf | |  | host all all 0.0.0.0/0 md5 | |  | //Exit admin user | |  | exit | |  | //Restart PostgreSQL server | |  | sudo /etc/init.d/postgresql restart | |  | //Go back to the remote machine and test your connection. | |  | psql --host 10.128.0.4 -U prashant --dbnamesparkDB --password |   We need a JDBC driver package, and hence I am adding it to my spark classpath using --package option. I am going to connect to PostgreSQL, and hence I am adding PostgreSQL JDBC package. If you want to use Oracle or any other database, you should include the appropriate JDBC package for the respective database. As you already know, the --package option will automatically pull the dependencies from the maven repository and include it in your classpath.  Spark JDBC Sink Example  Great! We are ready to write data from Spark over a JDBC connection. Use following code.   |  |  | | --- | --- | |  | spark-shell --packages org.postgresql:postgresql:9.4.1207 |   ou already understand this code. We have used the same in an earlier video.The first part is reading data from a CSV source. Then we register that data frame as a temporary view. Then I perform some aggregation and create another data set using an SQL code. My output is available as a data frame. Let me show you the content of this data frame.   |  |  | | --- | --- | |  | //Read CSV into Data Frame | |  | val df = spark.read | |  | .format("csv") | |  | .option("header", "true") | |  | .option("inferSchema", "true") | |  | .option("nullValue", "NA") | |  | .option("timestampFormat", "yyyy-MM-dd'T'HH:mm:ss") | |  | .option("mode", "failfast") | |  | .load("/home/prashant/spark-data/mental-health-in-tech-survey/survey.csv") | |  |  | |  | df.createOrReplaceTempView("survey\_tbl") | |  |  | |  | val dfout = spark.sql( | |  | """select gender, sum(yes) sum\_yes, sum(no) sum\_no | |  | from (select case when lower(trim(gender)) in ('male','m','male-ish','maile','mal','male (cis)', | |  | 'make','male ','man','msle','mail','malr','cis man', | |  | 'cis male') then 'Male' | |  | when lower(trim(gender)) in ('cis female','f','female','woman','femake','female ', | |  | 'cis-female/femme','female (cis)','femail') then 'Female' | |  | else 'Transgender' | |  | end as gender, | |  | case when treatment == 'Yes' then 1 else 0 end as yes, | |  | case when treatment == 'No' then 1 else 0 end as no | |  | from survey\_tbl) | |  | where gender != 'Transgender' | |  | group by gender""" | |  | ) |  |  | | --- | | dfout.show | |  | +------+-------+------+ | |  | |gender|sum\_yes|sum\_no| | |  | +------+-------+------+ | |  | |Female| 170| 77| | |  | | Male| 450| 541| | |  | +------+-------+------+ |  |  | | --- | | //Write Data Frame to JDBC | |  | dfout.write | |  | .format("jdbc") | |  | .mode("overwrite") | |  | .option("driver", "org.postgresql.Driver") | |  | .option("url", "jdbc:postgresql://10.128.0.4:5432/sparkDB") | |  | .option("dbtable", "survey\_results") | |  | .option("user", "prashant") | |  | .option("password", "pandey") | |  | .save() |   How to truncate and overwrite from Spark JDBC  However, you might be wondering, if the table already exists in the database, how will we truncate and write the data into the same table. If you already have a table in the database, you can use the overwrite mode with the truncate option.   |  |  | | --- | --- | |  | psql --host 10.128.0.4 -U prashant --dbname sparkDB --password | |  | \d+ survey\_results | |  | select \* from survey\_results; |   Spark JDBC Source Example  Reading from databases is as simple as reading from a file. The below example is to read a full table.   |  |  | | --- | --- | |  | //Write Data Frame to JDBC | |  | dfout.write | |  | .format("jdbc") | |  | .mode("overwrite") | |  | .option("truncate", "true") | |  | .option("driver", "org.postgresql.Driver") | |  | .option("url", "jdbc:postgresql://10.128.0.4:5432/sparkDB") | |  | .option("dbtable", "survey\_results") | |  | .option("user", "prashant") | |  | .option("password", "pandey") | |  | .save() |   How to use SQL in Spark JDBC  It is not necessary to read a full table. You might want to read some part of the data using an SQL statement. I have an example for that as well.   |  |  | | --- | --- | |  | //spark-shell --packages org.postgresql:postgresql:9.4.1207 | |  | val pgDF\_table = spark.read | |  | .format("jdbc") | |  | .option("driver", "org.postgresql.Driver") | |  | .option("url", "jdbc:postgresql://10.128.0.4:5432/sparkDB") | |  | .option("dbtable", "survey\_results") | |  | .option("user", "prashant") | |  | .option("password", "pandey") | |  | .load() | |  | pgDF\_table.show | |