16.9.1

PySpark ETL

You and Jennifer are excited about all the new technologies you're ready to use. You explain to Jennifer that you can now move your ETL process to the cloud to take the load off your local laptops, use strictly cloud resources, and eventually lead to automation. Let's get started with extraction!

Let's run through a mock scenario using two different types of raw data stored in S3. Our goal is to get this raw data from S3 into an RDS database. Let's start with uploading files to your own S3 bucket:

Create an S3 bucket, and then load the following files into the bucket:

- **user_data.csv (https://2u-data-curriculum-team.s3.amazonaws.com/dataviz-online/module_16/user_data.csv)**
- **user_payment.csv (https://2u-data-curriculum-team.s3.amazonaws.com/dataviz-online/module_16/user_payment.csv)**

IMPORTANT

Remember to make the bucket and files public.

Assume your company already has three tables set up in the RDS database and would like to get the raw data from S3 into the database. Create a new database in pgAdmin called "my_data_class_db." We'll have it represent the company database by first running the following schema in pgAdmin for our RDS:

```
-- Create Active User Table
CREATE TABLE active_user (
 id INT PRIMARY KEY NOT NULL,
 first_name TEXT,
 last_name TEXT,
 username TEXT
);
```

```
CREATE TABLE billing_info (
 billing_id INT PRIMARY KEY NOT NULL,
 street_address TEXT,
 state TEXT,
 username TEXT
);

CREATE TABLE payment_info (
 billing_id INT PRIMARY KEY NOT NULL,
 cc_encrypted TEXT
);
```

NOTE

> Table creation is not part of the ETL process. We're creating the tables to represent a pre-established database you need for the raw data. In a real-life situation, databases will already have a well-defined schema and tables for you, as the engineer, to process data into.

Start with creating a new notebook, installing Spark:

```
import os
# Find the latest version of spark 3.0 from http://www.apache.org/dist/spark/ and enter as the spark version
# For example:
# spark_version = 'spark-3.0.3'
spark_version = 'spark-3.<enter version>'
os.environ['SPARK_VERSION']=spark_version

# Install Spark and Java
!apt-get update
!apt-get install openjdk-11-jdk-headless -qq > /dev/null
!wget -q http://www.apache.org/dist/spark/$SPARK_VERSION/$SPARK_VERSION-bin-hadoop2.7.tgz
!tar xf $SPARK_VERSION-bin-hadoop2.7.tgz
!pip install -q findspark

# Set Environment Variables
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-11-openjdk-amd64"
os.environ["SPARK_HOME"] = f"/content/{spark_version}-bin-hadoop2.7"

# Start a SparkSession
```

```
import findspark
findspark.init()
```

We'll use Spark to write directly to our Postgres database. But in order to do so, there are few more lines of code we need.

First, enter the following code to download a Postgres driver that will allow Spark to interact with Postgres:

```
!wget https://jdbc.postgresql.org/download/postgresql-42.2.16.jar
```

You should get a message containing the words "HTTP request sent, awaiting response… 200 OK," indicating that your request was processed without a problem.

Then, start a Spark session with an additional option that adds the driver to Spark:

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("CloudETL").config("spark.driver.extraClassPath","/content/postgresql-42
```

We have performed the first two steps of the ETL process before with PySpark, so let's quickly review those.

Extract

We can connect to data storage, then extract that data into a DataFrame. We'll do this on two datasets, and be sure to replace the bucket name with one of your own.

We'll start by importing SparkFiles from PySpark into our notebook. This will allow Spark to add a file to our Spark project.

Next, the file is read in with the `read` method and combined with the `csv()` method, which pulls in our CSV stored in SparkFiles and infers the schema. SparkFiles.get() will have Spark retrieve the specified file, since we are dealing with a CSV. The "," is the chosen separator, and we will have Spark determine the head for us. Enter the following code:

```
# Read in data from S3 Buckets
from pyspark import SparkFiles
```

```
url ="https://YOUR-BUCKET-NAME.s3.amazonaws.com/user_data.csv"
spark.sparkContext.addFile(url)
user_data_df = spark.read.csv(SparkFiles.get("user_data.csv"), sep=",", header=True, inferSchema=True)
```

Finally, an action is called to show the first 10 runs and confirm our data extraction by entering the following code:

```
# Show DataFrame
user_data_df.show()
```

Repeat a similar process to load in the other data. Enter the code:

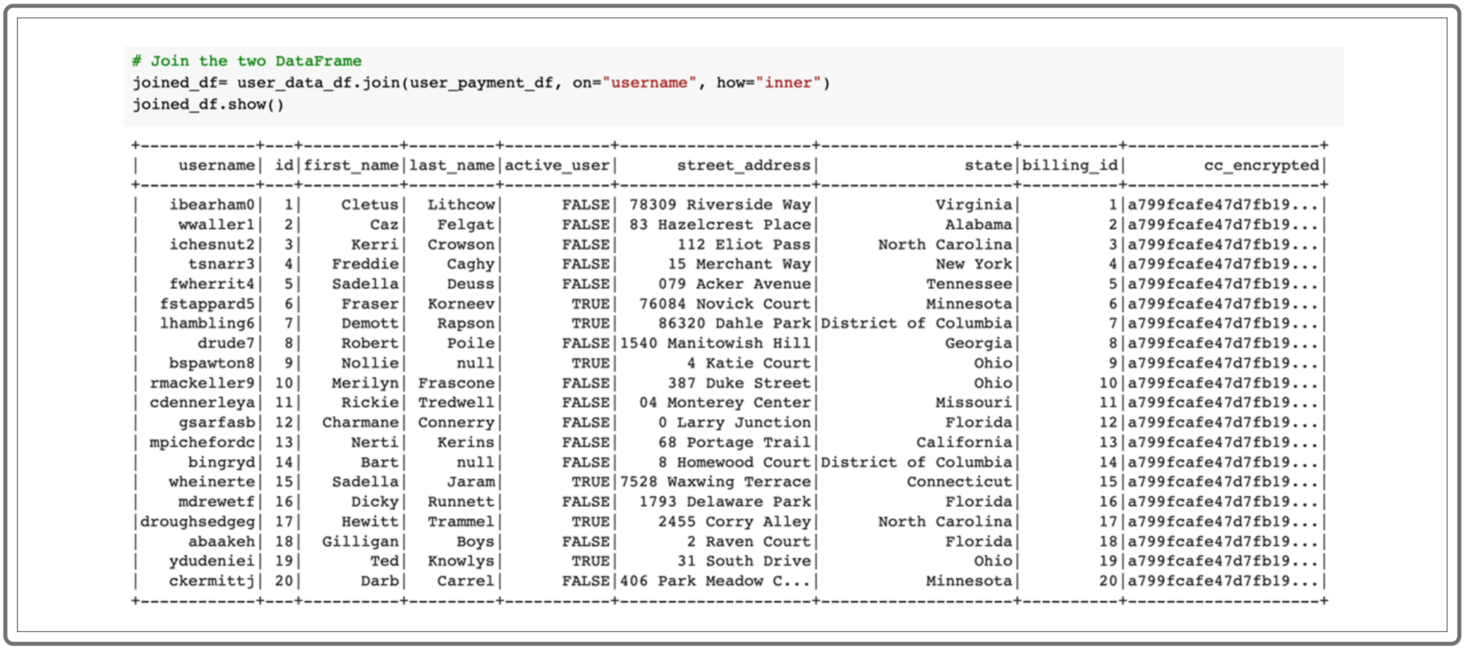
```
url ="https://YOUR-BUCKET-NAME.s3.amazonaws.com/user_payment.csv"
spark.sparkContext.addFile(url)
user_payment_df = spark.read.csv(SparkFiles.get("user_payment.csv"), sep=",", header=True, inferSchema=True)

# Show DataFrame
user_payment_df.show()
```

Transform

Now that the raw data stored in S3 is available in a PySpark DataFrame, we can perform our transformations.

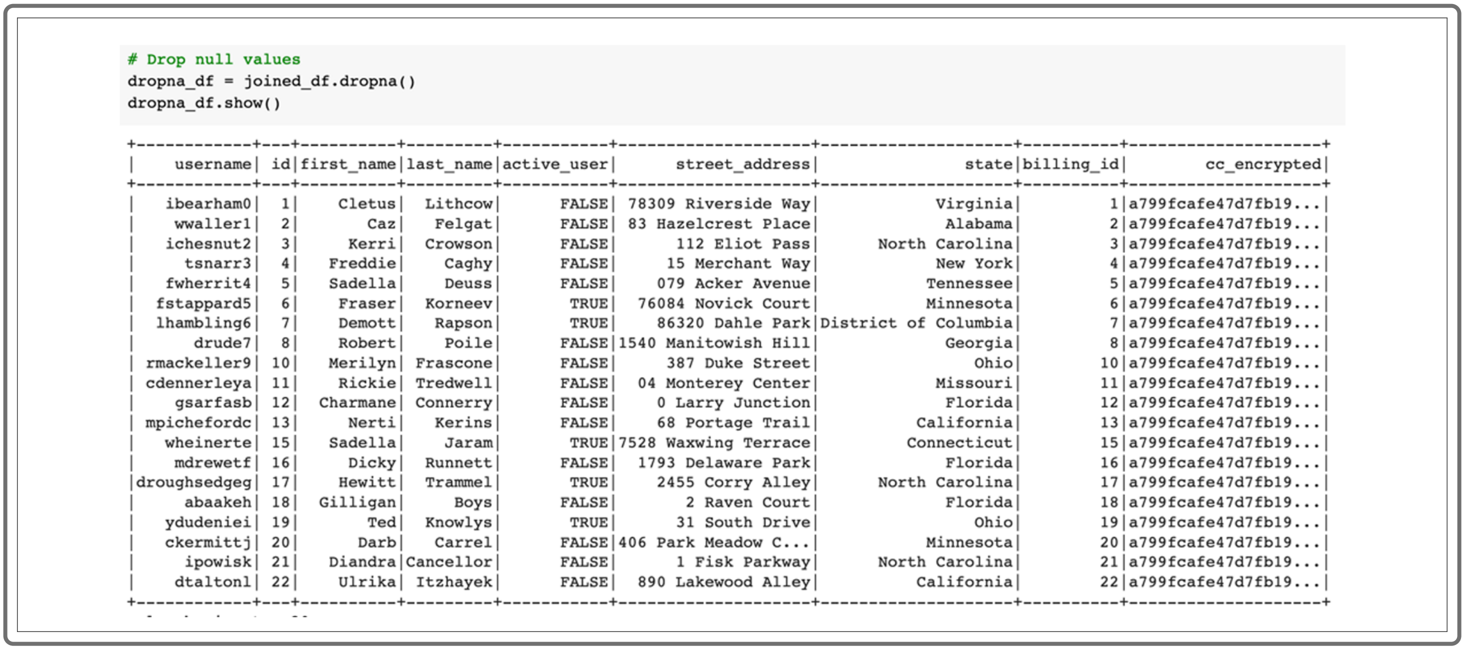
First, join the two tables:



```
# Join the two DataFrame
joined_df= user_data_df.join(user_payment_df, on="username", how="inner")
joined_df.show()
```

```
+------------+---+----------+---------+-----------+--------------------+--------------------+----------+--------------------+
|    username| id|first_name|last_name|active_user|      street_address|               state|billing_id|        cc_encrypted|
+------------+---+----------+---------+-----------+--------------------+--------------------+----------+--------------------+
|   ibearham0|  1|    Cletus|  Lithcow|      FALSE| 78309 Riverside Way|            Virginia|         1|a799fcafe47d7fb19...|
|    wwaller1|  2|       Caz|   Felgat|      FALSE| 83 Hazelcrest Place|             Alabama|         2|a799fcafe47d7fb19...|
|   ichesnut2|  3|     Kerri|  Crowson|      FALSE|      112 Eliot Pass|      North Carolina|         3|a799fcafe47d7fb19...|
|     tsnarr3|  4|   Freddie|    Caghy|      FALSE|     15 Merchant Way|            New York|         4|a799fcafe47d7fb19...|
|   fwherrit4|  5|   Sadella|    Deuss|      FALSE|    079 Acker Avenue|           Tennessee|         5|a799fcafe47d7fb19...|
|  fstappard5|  6|    Fraser|  Korneev|       TRUE|  76084 Novick Court|           Minnesota|         6|a799fcafe47d7fb19...|
|  lhambling6|  7|    Demott|   Rapson|       TRUE|    86320 Dahle Park|District of Columbia|         7|a799fcafe47d7fb19...|
|      drude7|  8|    Robert|    Poile|      FALSE|1540 Manitowish Hill|             Georgia|         8|a799fcafe47d7fb19...|
|   bspawton8|  9|    Nollie|     null|       TRUE|       4 Katie Court|                Ohio|         9|a799fcafe47d7fb19...|
| rmackeller9| 10|   Merilyn| Frascone|      FALSE|     387 Duke Street|                Ohio|        10|a799fcafe47d7fb19...|
| cdennerleya| 11|    Rickie| Tredwell|      FALSE|  04 Monterey Center|            Missouri|        11|a799fcafe47d7fb19...|
|    gsarfasb| 12|  Charmane| Connerry|      FALSE|    0 Larry Junction|             Florida|        12|a799fcafe47d7fb19...|
| mpichefordc| 13|     Nerti|   Kerins|      FALSE|    68 Portage Trail|          California|        13|a799fcafe47d7fb19...|
|     bingryd| 14|      Bart|     null|      FALSE|    8 Homewood Court|District of Columbia|        14|a799fcafe47d7fb19...|
|   wheinerte| 15|   Sadella|    Jaram|       TRUE|7528 Waxwing Terrace|         Connecticut|        15|a799fcafe47d7fb19...|
|    mdrewetf| 16|     Dicky|  Runnett|      FALSE|  1793 Delaware Park|             Florida|        16|a799fcafe47d7fb19...|
|droughsedgeg| 17|    Hewitt|  Trammel|       TRUE|    2455 Corry Alley|      North Carolina|        17|a799fcafe47d7fb19...|
|     abaakeh| 18|  Gilligan|     Boys|      FALSE|       2 Raven Court|             Florida|        18|a799fcafe47d7fb19...|
|   ydudeniei| 19|       Ted|  Knowlys|       TRUE|      31 South Drive|                Ohio|        19|a799fcafe47d7fb19...|
|   ckermittj| 20|      Darb|   Carrel|      FALSE|406 Park Meadow C...|           Minnesota|        20|a799fcafe47d7fb19...|
+------------+---+----------+---------+-----------+--------------------+--------------------+----------+--------------------+
```

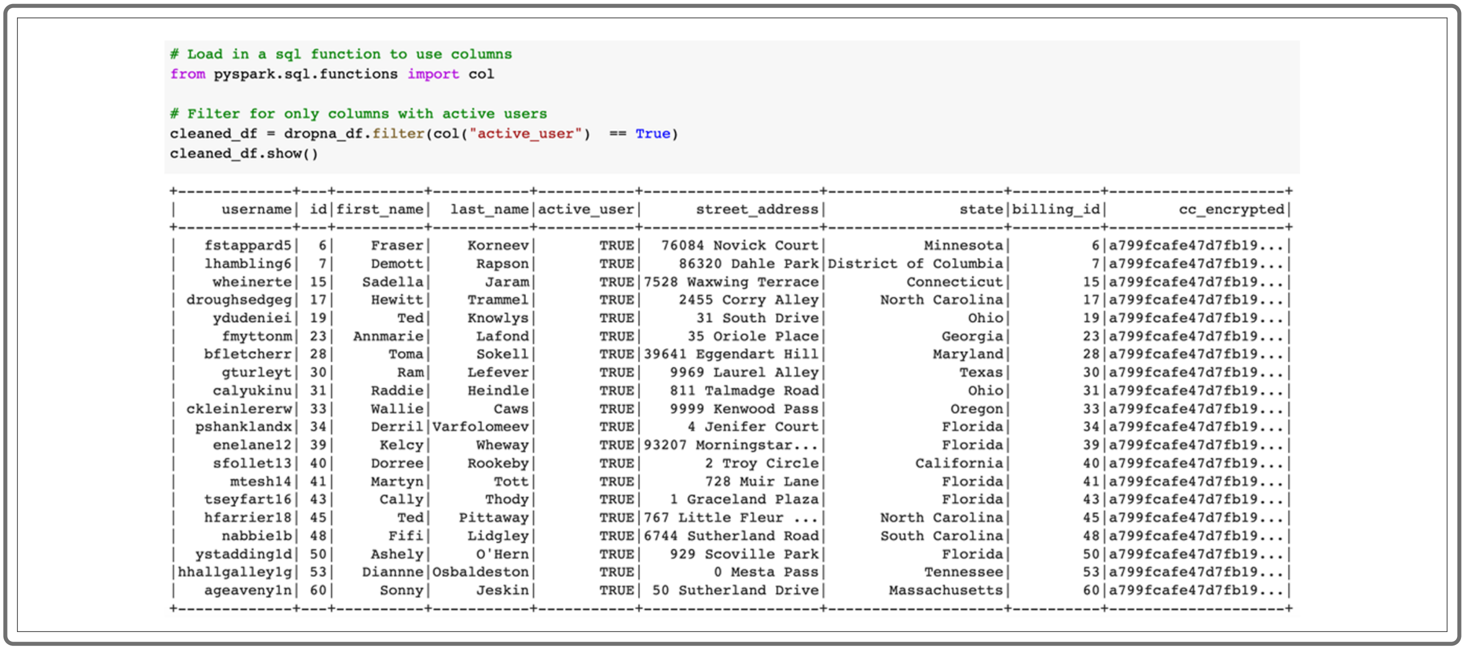
Next, drop any rows with null or "not a number" (NaN) values:



```
# Drop null values
dropna_df = joined_df.dropna()
dropna_df.show()
```

```
+------------+---+----------+---------+-----------+--------------------+--------------------+----------+--------------------+
|    username| id|first_name|last_name|active_user|      street_address|               state|billing_id|        cc_encrypted|
+------------+---+----------+---------+-----------+--------------------+--------------------+----------+--------------------+
|   ibearham0|  1|    Cletus|  Lithcow|      FALSE| 78309 Riverside Way|            Virginia|         1|a799fcafe47d7fb19...|
|    wwaller1|  2|       Caz|   Felgat|      FALSE| 83 Hazelcrest Place|             Alabama|         2|a799fcafe47d7fb19...|
|   ichesnut2|  3|     Kerri|  Crowson|      FALSE|      112 Eliot Pass|      North Carolina|         3|a799fcafe47d7fb19...|
|     tsnarr3|  4|   Freddie|    Caghy|      FALSE|     15 Merchant Way|            New York|         4|a799fcafe47d7fb19...|
|   fwherrit4|  5|   Sadella|    Deuss|      FALSE|    079 Acker Avenue|           Tennessee|         5|a799fcafe47d7fb19...|
|  fstappard5|  6|    Fraser|  Korneev|       TRUE|  76084 Novick Court|           Minnesota|         6|a799fcafe47d7fb19...|
|  lhambling6|  7|    Demott|   Rapson|       TRUE|    86320 Dahle Park|District of Columbia|         7|a799fcafe47d7fb19...|
|      drude7|  8|    Robert|    Poile|      FALSE|1540 Manitowish Hill|             Georgia|         8|a799fcafe47d7fb19...|
| rmackeller9| 10|   Merilyn| Frascone|      FALSE|     387 Duke Street|                Ohio|        10|a799fcafe47d7fb19...|
| cdennerleya| 11|    Rickie| Tredwell|      FALSE|  04 Monterey Center|            Missouri|        11|a799fcafe47d7fb19...|
|    gsarfasb| 12|  Charmane| Connerry|      FALSE|    0 Larry Junction|             Florida|        12|a799fcafe47d7fb19...|
| mpichefordc| 13|     Nerti|   Kerins|      FALSE|    68 Portage Trail|          California|        13|a799fcafe47d7fb19...|
|   wheinerte| 15|   Sadella|    Jaram|       TRUE|7528 Waxwing Terrace|         Connecticut|        15|a799fcafe47d7fb19...|
|    mdrewetf| 16|     Dicky|  Runnett|      FALSE|  1793 Delaware Park|             Florida|        16|a799fcafe47d7fb19...|
|droughsedgeg| 17|    Hewitt|  Trammel|       TRUE|    2455 Corry Alley|      North Carolina|        17|a799fcafe47d7fb19...|
|     abaakeh| 18|  Gilligan|     Boys|      FALSE|       2 Raven Court|             Florida|        18|a799fcafe47d7fb19...|
|   ydudeniei| 19|       Ted|  Knowlys|       TRUE|      31 South Drive|                Ohio|        19|a799fcafe47d7fb19...|
|   ckermittj| 20|      Darb|   Carrel|      FALSE|406 Park Meadow C...|           Minnesota|        20|a799fcafe47d7fb19...|
|     ipowisk| 21|   Diandra|Cancellor|      FALSE|      1 Fisk Parkway|      North Carolina|        21|a799fcafe47d7fb19...|
|    dtaltonl| 22|    Ulrika| Itzhayek|      FALSE|   890 Lakewood Alley|          California|        22|a799fcafe47d7fb19...|
+------------+---+----------+---------+-----------+--------------------+--------------------+----------+--------------------+
```

Filter for active users:

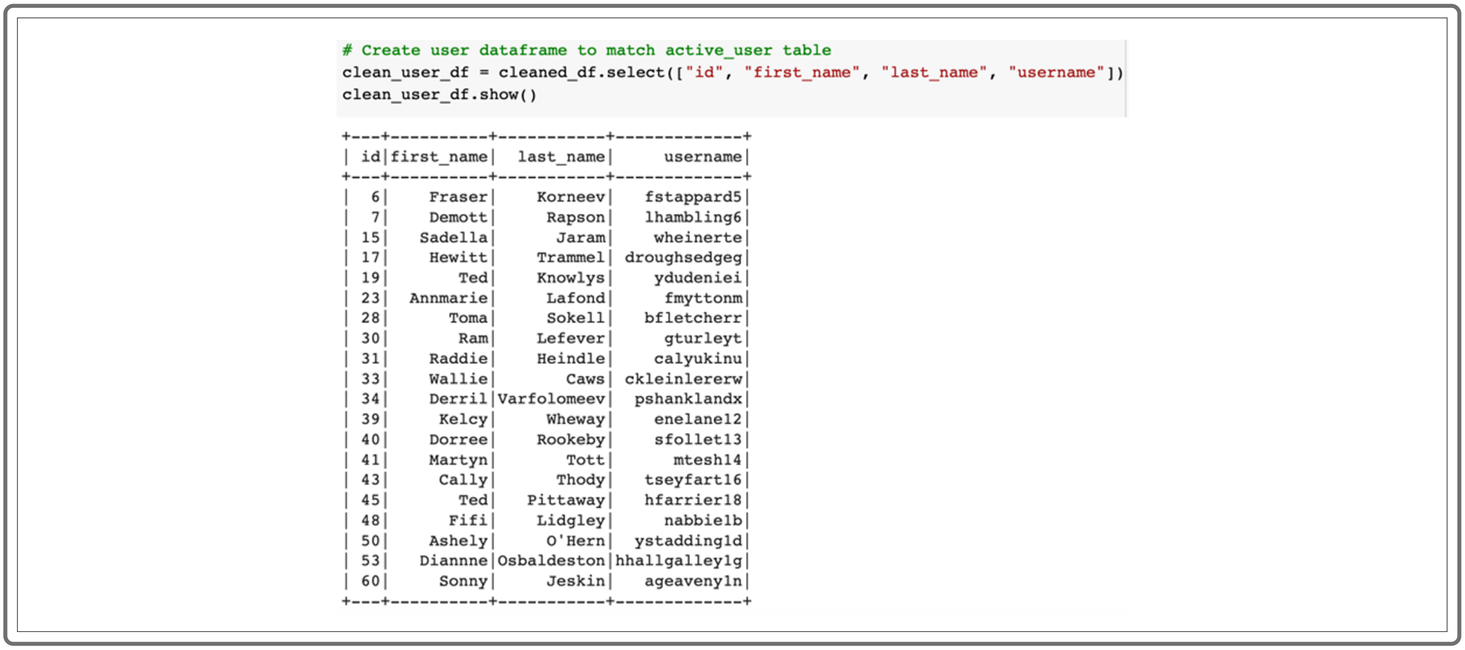


```
# Load in a sql function to use columns
from pyspark.sql.functions import col

# Filter for only columns with active users
cleaned_df = dropna_df.filter(col("active_user")  == True)
cleaned_df.show()
```

```
+-------------+---+----------+-----------+-----------+--------------------+--------------------+----------+--------------------+
|     username| id|first_name|  last_name|active_user|      street_address|               state|billing_id|        cc_encrypted|
+-------------+---+----------+-----------+-----------+--------------------+--------------------+----------+--------------------+
|   fstappard5|  6|    Fraser|    Korneev|       TRUE|  76084 Novick Court|           Minnesota|         6|a799fcafe47d7fb19...|
|   lhambling6|  7|    Demott|     Rapson|       TRUE|    86320 Dahle Park|District of Columbia|         7|a799fcafe47d7fb19...|
|    wheinerte| 15|   Sadella|      Jaram|       TRUE|7528 Waxwing Terrace|         Connecticut|        15|a799fcafe47d7fb19...|
| droughsedgeg| 17|    Hewitt|    Trammel|       TRUE|    2455 Corry Alley|      North Carolina|        17|a799fcafe47d7fb19...|
|    ydudeniei| 19|       Ted|    Knowlys|       TRUE|      31 South Drive|                Ohio|        19|a799fcafe47d7fb19...|
|     fmyttonm| 23|  Annmarie|     Lafond|       TRUE|     35 Oriole Place|             Georgia|        23|a799fcafe47d7fb19...|
|   bfletcherr| 28|      Toma|     Sokell|       TRUE|39641 Eggendart Hill|            Maryland|        28|a799fcafe47d7fb19...|
|     gturleyt| 30|       Ram|    Lefever|       TRUE|   9969 Laurel Alley|               Texas|        30|a799fcafe47d7fb19...|
|    calyukinu| 31|    Raddie|    Heindle|       TRUE|   811 Talmadge Road|                Ohio|        31|a799fcafe47d7fb19...|
| ckleinlererw| 33|    Wallie|       Caws|       TRUE|   9999 Kenwood Pass|              Oregon|        33|a799fcafe47d7fb19...|
|  pshanklandx| 34|    Derril|Varfolomeev|       TRUE|     4 Jenifer Court|             Florida|        34|a799fcafe47d7fb19...|
|    enelane12| 39|     Kelcy|     Wheway|       TRUE|93207 Morningstar...|             Florida|        39|a799fcafe47d7fb19...|
|    sfollet13| 40|    Dorree|    Rookeby|       TRUE|       2 Troy Circle|          California|        40|a799fcafe47d7fb19...|
|      mtesh14| 41|    Martyn|       Tott|       TRUE|       728 Muir Lane|             Florida|        41|a799fcafe47d7fb19...|
|   tseyfart16| 43|     Cally|      Thody|       TRUE|   1 Graceland Plaza|             Florida|        43|a799fcafe47d7fb19...|
|   hfarrier18| 45|       Ted|   Pittaway|       TRUE|767 Little Fleur ...|      North Carolina|        45|a799fcafe47d7fb19...|
|     nabbie1b| 48|      Fifi|    Lidgley|       TRUE|6744 Sutherland Road|      South Carolina|        48|a799fcafe47d7fb19...|
|  ystadding1d| 50|    Ashely|     O'Hern|       TRUE|   929 Scoville Park|             Florida|        50|a799fcafe47d7fb19...|
|hhallgalley1g| 53|   Diannne|Osbaldeston|       TRUE|        0 Mesta Pass|           Tennessee|        53|a799fcafe47d7fb19...|
|   ageaveny1n| 60|     Sonny|     Jeskin|       TRUE| 50 Sutherland Drive|       Massachusetts|        60|a799fcafe47d7fb19...|
+-------------+---+----------+-----------+-----------+--------------------+--------------------+----------+--------------------+
```

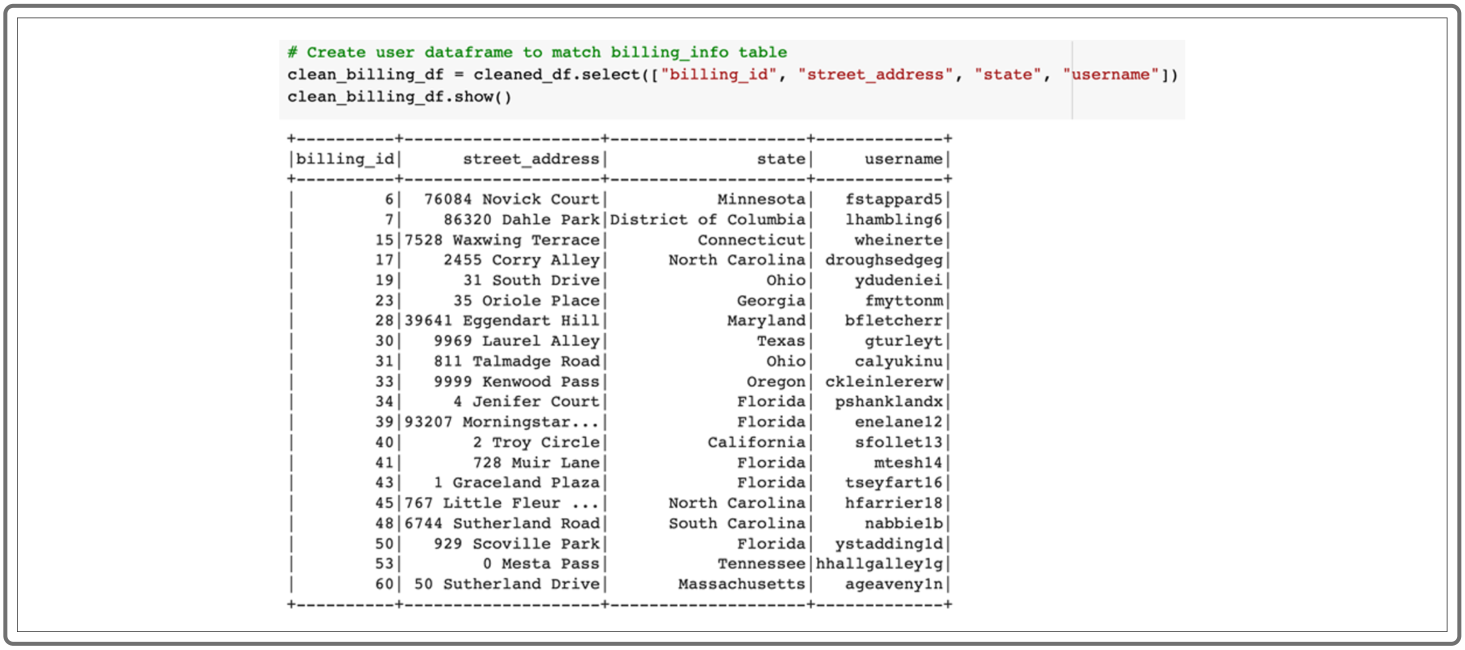
Next, select columns to create three different DataFrames that match what is in the AWS RDS database. Create a DataFrame to match the active_user table:



```
# Create user dataframe to match active_user table
clean_user_df = cleaned_df.select(["id", "first_name", "last_name", "username"])
clean_user_df.show()
```

```
+---+----------+-----------+-------------+
| id|first_name|  last_name|     username|
+---+----------+-----------+-------------+
|  6|    Fraser|    Korneev|   fstappard5|
|  7|    Demott|     Rapson|   lhambling6|
| 15|   Sadella|      Jaram|    wheinerte|
| 17|    Hewitt|    Trammel| droughsedgeg|
| 19|       Ted|    Knowlys|    ydudeniei|
| 23|  Annmarie|     Lafond|     fmyttonm|
| 28|      Toma|     Sokell|   bfletcherr|
| 30|       Ram|    Lefever|     gturleyt|
| 31|    Raddie|    Heindle|    calyukinu|
| 33|    Wallie|       Caws| ckleinlererw|
| 34|    Derril|Varfolomeev|  pshanklandx|
| 39|     Kelcy|     Wheway|    enelane12|
| 40|    Dorree|    Rookeby|    sfollet13|
| 41|    Martyn|       Tott|      mtesh14|
| 43|     Cally|      Thody|   tseyfart16|
| 45|       Ted|   Pittaway|   hfarrier18|
| 48|      Fifi|    Lidgley|     nabbie1b|
| 50|    Ashely|     O'Hern|  ystadding1d|
| 53|   Diannne|Osbaldeston|hhallgalley1g|
| 60|     Sonny|     Jeskin|   ageaveny1n|
+---+----------+-----------+-------------+
```

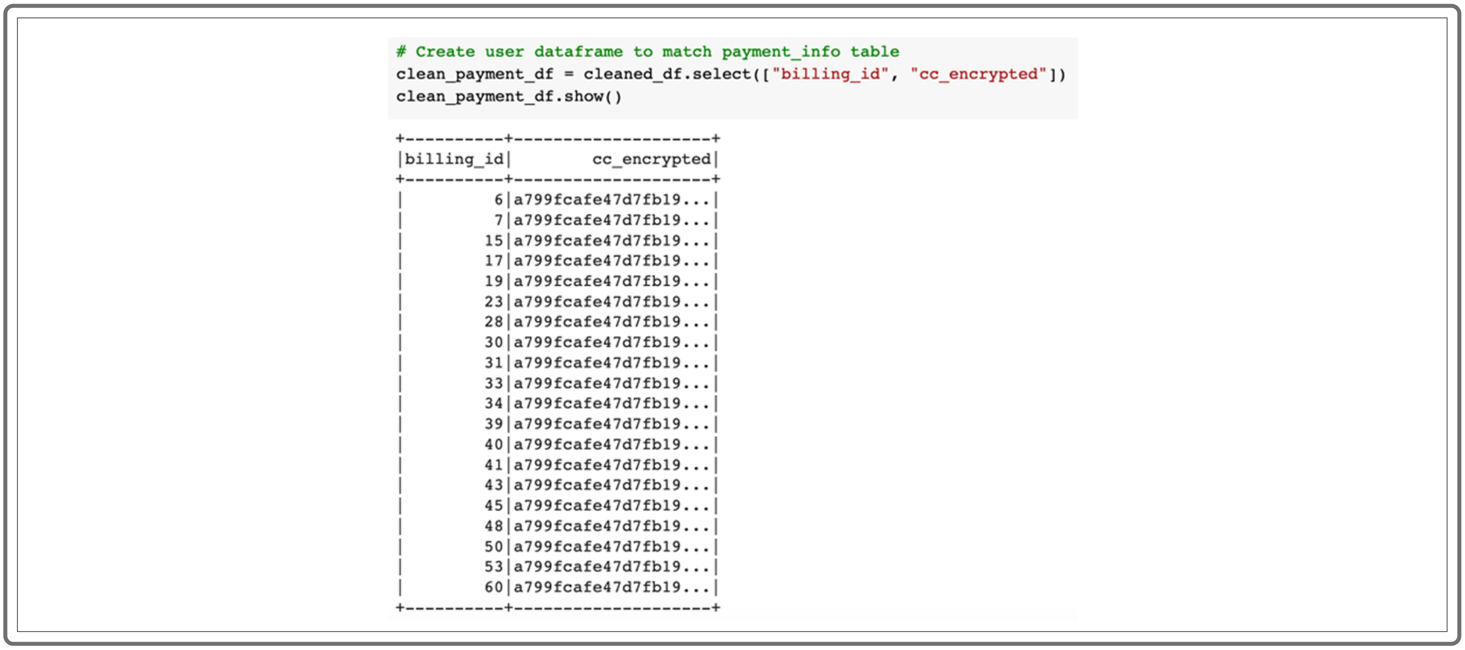
Next, create a DataFrame to match the billing_info table:



```
# Create user dataframe to match billing_info table
clean_billing_df = cleaned_df.select(["billing_id", "street_address", "state", "username"])
clean_billing_df.show()
```

```
+----------+--------------------+--------------------+-------------+
|billing_id|      street_address|               state|     username|
+----------+--------------------+--------------------+-------------+
|         6|  76084 Novick Court|           Minnesota|   fstappard5|
|         7|    86320 Dahle Park|District of Columbia|   lhambling6|
|        15| 7528 Waxing Terrace|         Connecticut|    wheinerte|
|        17|    2455 Corry Alley|      North Carolina| droughsedgeg|
|        19|      31 South Drive|                Ohio|    ydudeniei|
|        23|     35 Oriole Place|             Georgia|     fmyttonm|
|        28|39641 Eggendart Hill|            Maryland|   bfletcherr|
|        30|   9969 Laurel Alley|               Texas|     gturleyt|
|        31|   811 Talmadge Road|                Ohio|    calyukinu|
|        33|   9999 Kenwood Pass|              Oregon| ckleinlererw|
|        34|     4 Jenifer Court|             Florida|  pshanklandx|
|        39|93207 Morningstar...|             Florida|    enelane12|
|        40|       2 Troy Circle|          California|    sfollet13|
|        41|       728 Muir Lane|             Florida|      mtesh14|
|        43|   1 Graceland Plaza|             Florida|   tseyfart16|
|        45|767 Little Fleur ...|      North Carolina|   hfarrier18|
|        48|6744 Sutherland Road|      South Carolina|     nabbie1b|
|        50|   929 Scoville Park|             Florida|  ystadding1d|
|        53|        0 Mesta Pass|           Tennessee|hhallgalley1g|
|        60| 50 Sutherland Drive|       Massachusetts|   ageaveny1n|
+----------+--------------------+--------------------+-------------+
```

Finally, create a DataFrame to match the payment_info table:



```
# Create user dataframe to match payment_info table
clean_payment_df = cleaned_df.select(["billing_id", "cc_encrypted"])
clean_payment_df.show()
```

```
+----------+--------------------+
|billing_id|        cc_encrypted|
+----------+--------------------+
|         6|a799fcafe47d7fb19...|
|         7|a799fcafe47d7fb19...|
|        15|a799fcafe47d7fb19...|
|        17|a799fcafe47d7fb19...|
|        19|a799fcafe47d7fb19...|
|        23|a799fcafe47d7fb19...|
|        28|a799fcafe47d7fb19...|
|        30|a799fcafe47d7fb19...|
|        31|a799fcafe47d7fb19...|
|        33|a799fcafe47d7fb19...|
|        34|a799fcafe47d7fb19...|
|        39|a799fcafe47d7fb19...|
|        40|a799fcafe47d7fb19...|
|        41|a799fcafe47d7fb19...|
|        43|a799fcafe47d7fb19...|
|        45|a799fcafe47d7fb19...|
|        48|a799fcafe47d7fb19...|
|        50|a799fcafe47d7fb19...|
|        53|a799fcafe47d7fb19...|
|        60|a799fcafe47d7fb19...|
+----------+--------------------+
```

Once our data has been transformed to fit the tables in our database, we're ready to move on to the "Load" step.

Load

The final step is to get our transformed raw data into our database. PySpark can easily connect to a database to load the DataFrames into the table. First, we'll do some configuration to allow the connection with the following code:

```
# Store environmental variable
from getpass import getpass
password = getpass('Enter database password')
# Configure settings for RDS
mode = "append"
jdbc_url="jdbc:postgresql://<connection string>:5432/<database-name>"
config = {"user":"postgres",
          "password": password,
          "driver":"org.postgresql.Driver"}
```

You'll need to provide your username and password, and also supply the AWS server name where `<connection string>` is located in the code above. To find it in PgAdmin, right-click AWS in the Server directory listing on the left side of PgAmin, and then select Properties in the drop-down menu. Select the Connection tab in the window that opens, and then select the address in the Host name/address field. Copy that address and paste it in place of `<connection string>`.

Let's further break down what's happening here:

- The `getpass` module is used to temporarily store your database password without exposing it to the public.
- `mode` is what we want to do with the DataFrame to the table, such as `overwrite` or `append`. We'll append to the current table because every time we run this ETL process, we'll want more data added to our database without removing any.
- The `jdbc_url` is the connection string to our database.
    - Replace `<connection string>` with the endpoint connection url found from your AWS RDS console.
    - Replace `<database name>` with the name of your database you wish to connect to.
- A dictionary of configuration that includes the `user`, `password`, and `driver` to what type of database is being used.
    - The `user` field is the username for your database, which should be `postgres` if you followed with the creation of the RDS instance. Otherwise, enter the one you created.
    - The `password` would be the password you created when making the RDS instance.

NOTE

If you forget anything like the name of the database or user name you can check on pgAdmin for these values. Be sure that you are entering the name of the database and not the name of your server in the connection string.

The cleaned DataFrames can then be written directly to our database by using the `.write.jdbc` method that takes in the parameters we set:

- The connection string stored in `jdbc_url` is passed to the URL argument.
- The corresponding name of the table we are writing the DataFrame to.
- The mode we're using, which is "append."
- The connection configuration we set up passed to the properties.

The code is as follows:

```
# Write DataFrame to active_user table in RDS
clean_user_df.write.jdbc(url=jdbc_url, table='active_user', mode=mode, properties=config)
```

```
# Write dataframe to billing_info table in RDS
clean_billing_df.write.jdbc(url=jdbc_url, table='billing_info', mode=mode, properties=config)
```

```
# Write dataframe to payment_info table in RDS
clean_payment_df.write.jdbc(url=jdbc_url, table='payment_info', mode=mode, properties=config)
```

Let's wrap up by double-checking our work and running queries in pgAdmin on our database to confirm that the load did exactly what we wanted:

```
-- Query database to check successful upload
SELECT * FROM active_user;
SELECT * FROM billing_info;
SELECT * FROM payment_info;
```

Nice work! You now have enough knowledge and practice with PySpark and AWS to begin your client project.