به نام خدا

تهیه کننده: ابراهیم صدیقی شماره دانشجویی: ۹۹۳۱۰۹۸

گزارش یک:

RDDs از نظر معماری و سطوح انتزاع یک سطح انتزاعی پایینتر نسبت به DataFrame ها دارند و مستقیما روی توزیع داده ها و عملیات بر روی آنها کنترل دارند.RDD ها مجموعه ای توزیع شده از اشیاً هستند و با استفاده از توابع تحولی)Transformation)و عملیات)Action)مدیریت میشوند.کاربران میتوانند با استفاده از RDD ها کنترل دقیقی روی توزیع دادهها، تکرارپذیری و پردازش موازی داشته باشند. بهینه سازی عملکرد RDD ها به طور دستی انجام میشود. کاربران باید خودشان بهینه سازیهای لازم را برای عملیاتهای سنگین و توزیع داده ها انجام دهند. RDD ها از بهینه سازیهای خودکار اسپارک کاتالیزور استفاده نمیکنند. در مورد موارد استفاده آنها داریم که زمانی که نیاز به کنترل دقیق تری روی داده ها و نحوه توزیع و پردازش آنها باشد، استفاده از RDD ها مناسب تر است. برای پردازش داده هایی که ساختار مشخصی ندارند، RDD ها گزینه مناسبی هستند. DataFrame از نظر معماری و سطوح انتزاع یک سطح انتزاعی بالاتر نسبت به RDD ها دارند و به صورت جداول دو بعدی سازماندهی میشود. DataFrame ها به صورت جدول با ستونهای نامگذاری شده نمایش داده میشوند و از API SQL و توابع تحولی سطح باال بهره مند هستند. DataFrame ها از بهینه ساز کاتالیزور)Optimizer Catalyst)استفاده میکنند که به طور خودکار برنامههای اجرایی بهینه را ایجاد میکند. عملیاتهای بر روی DataFrame ها میتواند به صورت)Lazy)اجرا شود و بهینه سازی هایی مانند پردازش درون حافظهای و فشردهسازی دادهها را شامل شود. برای تحلیل دادههای ساختاریافته و نیمهساختاریافته که شباهت به جداول دارند، DataFrameها انتخاب بهتری هستند. برای انجام تحلیلهای پیچیده و نیاز به بهینهسازیهای خودکار، استفاده از DataFrameها بسیار موثر است.

گزارش دو:

.۱ قابلیتهای پرس و جو

Spark SQL

Spark SQL از زبان SQL استاندارد استفاده میکند که برای بسیاری از کاربران آشنا و شفاف است.

قابلیتهای پرس و جوی پیچیدهای دارد، شامل joinها، subqueryها، aggregation، window functionها و موارد دیگر.

امکان ادغام با Hive برای استفاده از جداول و متا دیتاهای موجود.

API DataFrame

DataFrame API با استفاده از کد برنامهنویسی و به صورت توابع قابل استفاده است که میتواند برای توسعهدهندگان نرمافزار جذاب باشد.

امکان استفاده از توابع خاصتر و پیچیدهتر نسبت به SQL.

برخی از عملیات پیچیده و سفارشی را میتوان به سادگی در DataFrame پیادهسازی کرد که در SQL دشوارتر است.

.۲ بهینهسازی عملکرد

Spark SQL
از یک بهینهساز داخلی به نام Catalyst استفاده میکند که به صورت خودکار برنامههای اجرایی را بهینهسازی میکند.
بهینهسازیهای مبتنی بر هزینه که بر اساس آمار و دادههای موجود تصمیم میگیرند.
API DataFrame
API DataFrame نیز از Optimizer Catalyst بهرهمند است، بنابراین بسیاری از بهینهسازیها مشترک هستند.
استفاده از UDF ها ممکن است برخی از بهینهسازیهای خودکار را غیرفعال کند، بنابراین عملکرد ممکن است در برخی موارد کاهش یابد.

.۳ قابلیت استفاده
Spark SQL
برای کاربرانی که با SQL آشنایی دارند بسیار راحت است. یک زبان استاندارد است که میتواند در سیستمهای مختلف استفاده شود.
بسیاری از ابزارهای Intelligence Business به خوبی با SQL سازگار هستند.
API DataFrame
مناسب برای کسانی که به برنامهنویسی تابعی عالقهمند هستند. امکان ترکیب با سایر APIها و کتابخانههای Spark.
API DataFrame با زبانهای مختلفی مانند Python، Scala و Java سازگار است.

مثالهایی از ترجیح بر دیگری:
استفاده از SQL Spark:
-برای کاربران تجاری که به گزارشهای پیچیده نیاز دارند و با SQL آشنا هستند. استفاده از DataFrame API:
برای توسعهدهندگانی که به پردازشهای پیچیده و سفارشی نیاز دارند.
هنگامی که نیاز به ادغام کدهای پردازشی با سایر بخشهای برنامه کاربردی است.

گزارش سه:
Partitioning یا تقسیمبندی دادهها در Spark Apache به معنای توزیع دادهها در بخشهای کوچکتر و جداگانه است که میتوانند به صورت موازی پردازش شوند.
در Spark Apache، یک RDD یا DataFrame به چندین بخش کوچکتر به نام پارتیشن تقسیم میشود.
هر پارتیشن یک زیرمجموعه از دادهها را نگهداری میکند و میتواند به صورت مستقل توسط یک نود)گره(در کالستر پردازش شود.
اهمیت Partitioning در پردازش داده توزیع شده:
.1افزایش کارایی)Efficiency(: با توزیع دادهها در بین نودهای مختلف، از منابع محاسباتی و حافظه هر نود بهینهتر استفاده میشود.

پارتیشنها به صورت موازی پردازش میشوند که این باعث افزایش سرعت پردازش و کاهش زمان کلی اجرا میشود.
2. Balancing Load): توزیع یکنواخت پارتیشنها بین نودها باعث میشود که هیچ نودی بار کاری بیش از حد نداشته باشد.
3. افزایش انعطافپذیری (Fault Tolerance): اگر یک نود خراب شود، فقط پارتیشنهای مربوط به آن نود نیاز به بازپردازش دارند، نه کل دادهها. این باعث میشود که سیستم در برابر خطاها مقاومتر باشد.

گزارش چهار:
1. پارتیشنبندی پیشفرض (Partitioning Default):
اگر تعداد پارتیشنها مشخص نشود، اسپارک به طور خودکار تعداد پارتیشنها را تعیین میکند.
مزایا: برای بسیاری از jobهای ساده مناسب است.
معایب: در برخی موارد، تعداد پارتیشنها ممکن است بهینه نباشد و منجر به پارتیشنهای بیش از حد بزرگ یا کوچک شود.
2. پارتیشنبندی بر اساس کلید (Partitioning Hash):
دادهها بر اساس مقدار هش کلیدهای مشخص شده پارتیشنبندی میشوند.
مزایا: توزیع یکنواخت دادهها در پارتیشنها، مناسب برای عملیات join و aggregation.
معایب: در صورتی که دادهها دارای توزیع یکنواخت نباشند، ممکن است منجر به عدم تعادل در بارگذاری پارتیشنها شود.
3.پارتیشنبندی بر اساس محدوده (Partitioning Range):
دادهها بر اساس محدوده مقادیر کلیدهای مشخص شده پارتیشنبندی میشوند.
مزایا: مناسب برای عملیات مرتبسازی و جستجو، توزیع یکنواخت در صورت تعریف محدودههای مناسب.
معایب : تعیین محدودههای مناسب نیازمند دانش پیشین درباره توزیع دادهها است.
4. پارتیشنبندی سفارشی (Partitioning Custom):
کاربران میتوانند با پیادهسازی یک تابع پارتیشنبندی سفارشی، دادهها را به صورت دلخواه پارتیشنبندی کنند.
مزایا: انعطافپذیری بالا، امکان بهینهسازی برای سناریوهای خاص.
معایب: نیاز به پیادهسازی و نگهداری پیچیدهتر، خطای انسانی محتمل.

تاثیر پارتیشنبندی بر عملکرد اجرای job در اسپارک: (Balancing Load):
پارتیشنهای یکنواختتر منجر به توازن بهتر بار و کاهش زمان اجرای jobها میشود.
(Overhead Network): پارتیشنبندی بهینه میتواند تعداد و حجم دادههای منتقل شده بین گرهها را کاهش دهد.
(Utilization CPU and Memory): پارتیشنهای کوچکتر ممکن است به حافظه کمتری نیاز داشته باشند، اما تعداد زیاد آنها میتواند سربار پردازشی ایجاد کند. پارتیشنهای بزرگتر ممکن است منجر به استفاده بهینهتر از پردازندهها شود، اما نیازمند حافظه بیشتری هستند.
مرحله shuffle: عملیات shuffle در اسپارک که دادهها را بین گرهها منتقل میکند، بسیار هزینهبر است. پارتیشنبندی مناسب میتواند هزینههای مربوط به shuffle را کاهش دهد.

امتیازی:

narrow و transformations wide به نوع ارتباط و پردازش دادهها در RDD اشاره دارد.

. transformations Narrow به آن دسته از تبدیالتی گفته میشود که در آنها هر پارتیشن از RDD ورودی حداکثر به یک پارتیشن از RDD خروجی نگاشت میشود. این تبدیلها نیازی به جابجایی دادهها بین نودهای مختلف ندارند و به صورت local پردازش میشوند. مثالی از narrow .است mapPartitions و , map filter شامل transformations

transformations Wide به آن دسته از تبدیالتی گفته میشود که در آنها دادهها باید بین پارتیشنها نیازمند عملیات shuffling است. این تبدیالت نیازمند جابجایی دادهها بین نودهای جابجا شوند، که معموالً مختلف هستند و شامل عملیاتهایی مثل reduceByKey , groupByKey و join میشود.

تفاوت ها:

محلی بودن پردازش: transformations Narrow: پردازش دادهها به صورت محلی)local)انجام میشود.

transformations Wide: نیازمند جابجایی دادهها بین نودهای مختلف)shuffling)هستند.

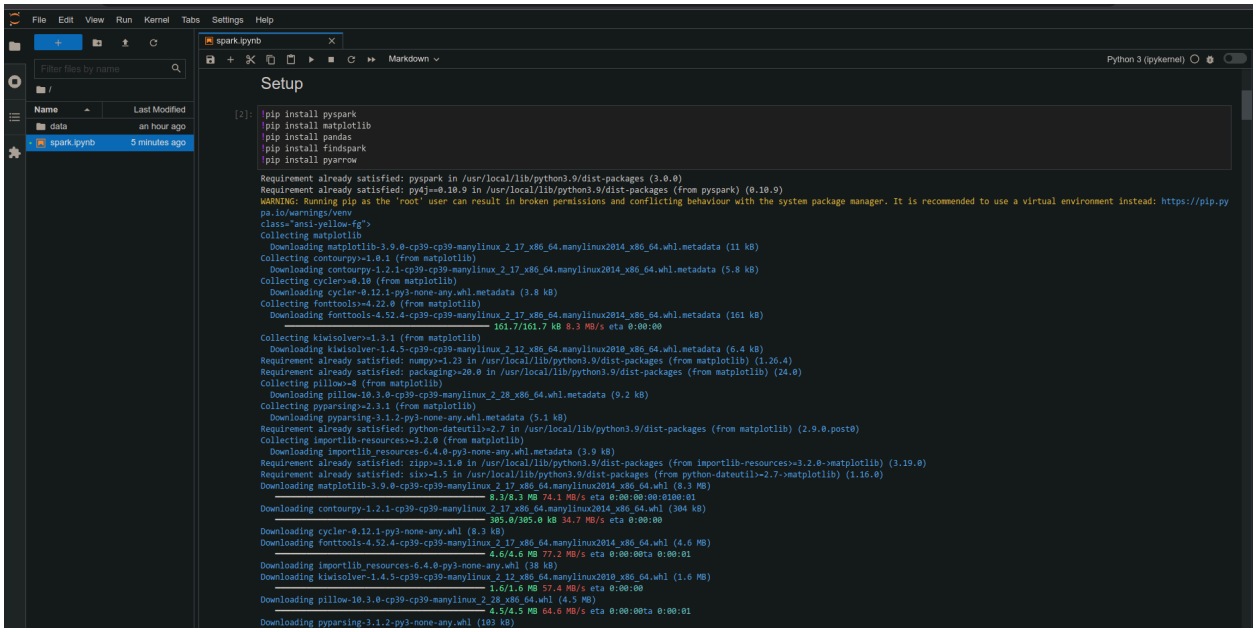
کارایی: transformations Narrow معموالً سریعتر و کارآمدتر هستند زیرا نیازی به جابجایی دادهها ندارند.

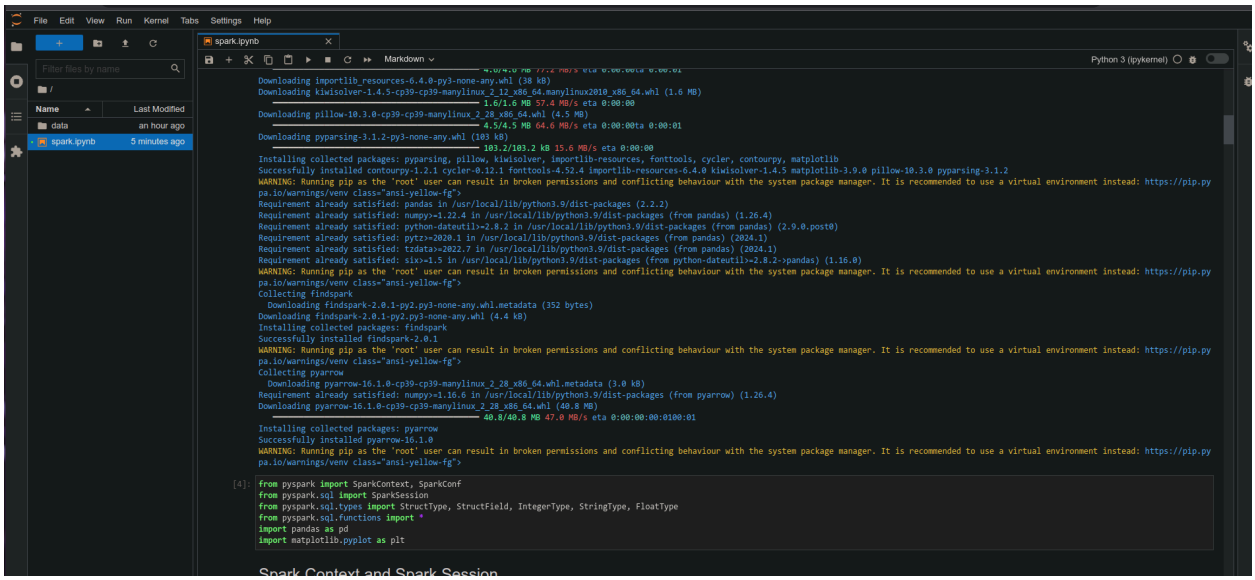
transformations Wide معموالً کندتر هستند زیرا شامل عملیات shuffling میشوند که به انتقال دادهها بین نودها نیاز دارد.

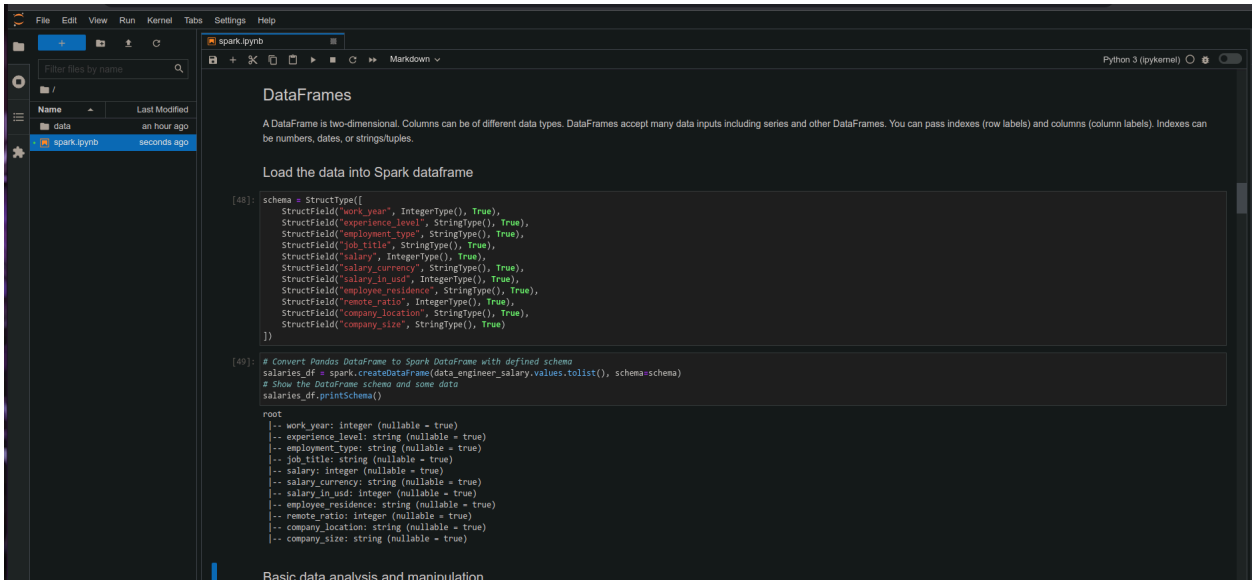
تصاویر:

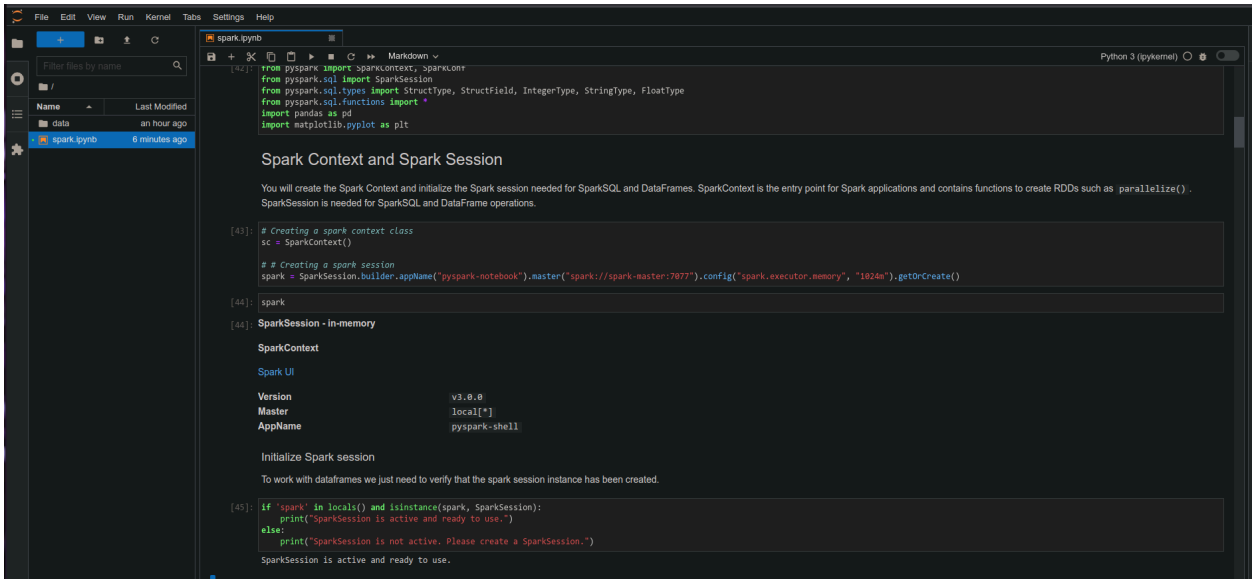
```
root@debian-16gb-nbg1-1:~/cloud/hind# bash master-build.sh
b8156f44a8c083c33a0f5e5e1b8135553ae4a2a1ffcfe123c8dfe803decd00da
[+] Building 431.9s (24/24) FINISHED                                                                   docker:default
 => [internal] load build definition from Dockerfile                                                              0.0s
 => => transferring dockerfile: 1.83kB                                                                            0.0s
 => [internal] load metadata for docker.io/library/ubuntu:22.04                                                   1.3s
 => [internal] load .dockerignore                                                                                 0.0s
 => => transferring context: 2B                                                                                   0.0s
 => [ 1/19] FROM docker.io/library/ubuntu:22.04@sha256:a6d2b38300ce017add71440577d5b0a90460d0e57fd7aec21dd0d1b0761bbfb2   5.1s
 => => resolve docker.io/library/ubuntu:22.04@sha256:a6d2b38300ce017add71440577d5b0a90460d0e57fd7aec21dd0d1b0761bbfb2     0.0s
 => => sha256:52882761a72a60649edff9a2478835325d084fb640ea32a975e29e12a012025f 2.30kB / 2.30kB                    0.0s
 => => sha256:a8b1c5f80c2d2a757adc963e3fe2dad0b4d229f83df3349fbb70e4d12dd48822 29.53MB / 29.53MB                  0.5s
 => => sha256:a6d2b38300ce017add71440577d5b0a90460d0e57fd7aec21dd0d1b0761bbfb2 1.13kB / 1.13kB                    0.0s
 => => sha256:2af372c1e2645779643284c7dc38775e3dbbc417b2d784a27c5a9eb784014fb8 424B / 424B                        0.0s
 => => extracting sha256:a8b1c5f80c2d2a757adc963e3fe2dad0b4d229f83df3349fbb70e4d12dd48822                         4.4s
 => [internal] load build context                                                                                 0.0s
 => => transferring context: 4.67kB                                                                               0.0s
 => [ 2/19] COPY test.py /                                                                                        0.1s
 => [ 3/19] RUN apt-get update && DEBIAN_FRONTEND=noninteractive apt-get install -y --no-install-recommends       openjdk-8-jdk       net-tools       curl       netcat       gnupg       libsnappy-dev       nano       wget  46.0s
 => [ 4/19] RUN apt-get update && apt-get upgrade -y                                                              6.1s
 => [ 5/19] RUN apt-get install -y make build-essential libssl-dev zlib1g-dev        libbz2-dev libreadline-dev libsqlite3-dev wget curl llvm        libncurses5-dev libncursesw5-dev xz-utils tk-dev   31.0s
 => [ 6/19] RUN wget https://www.python.org/ftp/python/3.8.4/Python-3.8.4.tgz                                     0.5s
 => [ 7/19] RUN tar xzf Python-3.8.4.tgz                                                                          1.1s
 => [ 8/19] RUN cd Python-3.8.4 && ./configure --enable-optimizations && make -j 8 && make altinstall           271.4s
 => [ 9/19] RUN apt-get install -y python3-pip                                                                    6.5s
 => [10/19] RUN pip3 install --no-cache-dir --upgrade pyspark                                                    27.9s
 => [11/19] RUN update-alternatives --install /usr/bin/python python /usr/bin/python3 10                          0.3s
 => [12/19] RUN curl -O https://dist.apache.org/repos/dist/release/hadoop/common/KEYS                             1.2s
 => [13/19] RUN gpg --import KEYS                                                                                 0.8s
 => [14/19] RUN set -x     && curl -fSL "https://www.apache.org/dist/hadoop/common/hadoop-3.3.6/hadoop-3.3.6.tar.gz" -o /tmp/hadoop.tar.gz     && tar -xvf /tmp/hadoop.tar.gz -C /opt/     && rm /tmp/hadoop.tar.gz*   18.9s
 => [15/19] RUN ln -s /opt/hadoop-3.3.6/etc/hadoop /etc/hadoop                                                    0.3s
 => [16/19] RUN mkdir /opt/hadoop-3.3.6/logs                                                                      0.3s
 => [17/19] RUN mkdir /hadoop-data                                                                                0.3s
 => [18/19] ADD entrypoint.sh /entrypoint.sh                                                                      0.1s
 => [19/19] RUN chmod a+x /entrypoint.sh                                                                          0.3s
 => exporting to image                                                                                           12.1s
 => => exporting layers                                                                                          12.1s
 => => writing image sha256:16409adf922d702c86b39fe53608463598d43da0ab2e4e8968421d4f9ff10f9e                      0.0s
 => => naming to docker.io/library/hadoop-base                                                                    0.0s
[+] Building 125.7s (15/15) FINISHED                                                                   docker:default
 => [internal] load build definition from Dockerfile                                                              0.0s
 => => transferring dockerfile: 1.53kB                                                                            0.0s
 => [internal] load metadata for docker.io/library/openjdk:8-jre-slim                                             1.2s
 => [internal] load .dockerignore                                                                                 0.0s
 => => transferring context: 2B                                                                                   0.0s
 => [ 1/11] FROM docker.io/library/openjdk:8-jre-slim@sha256:53186129237fbb8bc0a12dd36da6761f4c7a2a20233c20d4eb0d497e4045a4f5   11.0s
 => => resolve docker.io/library/openjdk:8-jre-slim@sha256:53186129237fbb8bc0a12dd36da6761f4c7a2a20233c20d4eb0d497e4045a4f5     0.0s
 => => sha256:53186129237fbb8bc0a12dd36da6761f4c7a2a20233c20d4eb0d497e4045a4f5 549B / 549B                        0.0s
 => => sha256:285c61a1e5e6b7b3709729b69558670148c5fdc6eb7104fae7dd370042c51430 1.16kB / 1.16kB                    0.0s
 => => sha256:85b121affeddcffc7bc6618140bb0285ad1257bd318676ddc67816863c0686c0 7.47kB / 7.47kB                    0.0s
 => => sha256:1efc276f4ff952c055dea726cfc96ec6a4fdb8b62d9eed816bd2b788f2860ad7 31.37MB / 31.37MB                  0.7s
 => => sha256:a2f2f93da48276873890ac821b3c991d53a7e864791aaf82c39b7863c908b93b 1.58MB / 1.58MB                    0.2s
 => => sha256:1a2de4cc94315f2ba5015e6781672aa8e0b1456a4d488694bb5f016d8f59fa70 210B / 210B                        0.4s
 => => sha256:d2421c7a4bbfc037d7fb487893cc5fe145e329dfb39b5ee6557016bf6c34072d 41.70MB / 41.70MB                  0.8s
 => => extracting sha256:1efc276f4ff952c055dea726cfc96ec6a4fdb8b62d9eed816bd2b788f2860ad7                         5.2s
 => => extracting sha256:a2f2f93da48276873890ac821b3c991d53a7e864791aaf82c39b7863c908b93b                         0.3s
 => => extracting sha256:1a2de4cc94315f2ba5015e6781672aa8e0b1456a4d488694bb5f016d8f59fa70                         0.0s
 => => extracting sha256:d2421c7a4bbfc037d7fb487893cc5fe145e329dfb39b5ee6557016bf6c34072d                         4.5s
 => [ 2/11] RUN mkdir -p /opt/workspace/data                                                                      0.5s
 => [ 3/11] RUN mkdir -p /usr/share/man/man1                                                                      0.3s
 => [ 4/11] RUN apt-get update -y                                                                                 2.6s
 => [ 5/11] RUN apt-get install -y curl python3 r-base                                                           71.4s
```

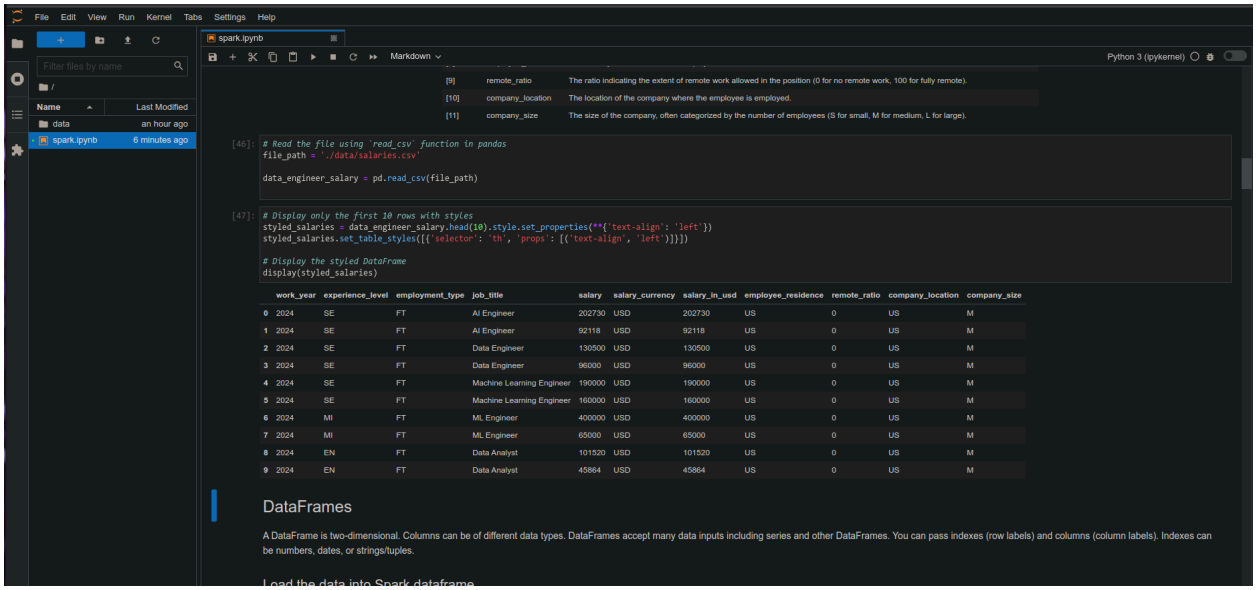
```
 => [jupyter-notebook 11/16] RUN apt-get update -y                                                                3.1s
 => [hadoop-namenode 3/4] ADD run.sh /run.sh                                                                      0.1s
 => [hadoop-historyserver 3/4] ADD run.sh /run.sh                                                                 0.0s
 => [hadoop-namenode 4/4] RUN chmod a+x /run.sh                                                                   1.0s
 => [hadoop-historyserver 4/4] RUN chmod a+x /run.sh                                                              0.9s
 => [hadoop-nodemanager-1] exporting to image                                                                     0.0s
 => => exporting layers                                                                                           0.0s
 => => writing image sha256:daf39be839ff80506b491f983778836427e18e5ab42b0f5563801701b4055f9e                      0.0s
 => => naming to docker.io/library/hind-hadoop-nodemanager-1                                                      0.0s
 => [hadoop-resourcemanager] exporting to image                                                                   0.1s
 => => exporting layers                                                                                           0.0s
 => => writing image sha256:d39e6c8bd1141f78404b8493b62652e58e24c621912ae383ee12cfccab737404                      0.0s
 => => naming to docker.io/library/hind-hadoop-resourcemanager                                                    0.0s
 => [hadoop-datanode] exporting to image                                                                          0.0s
 => => exporting layers                                                                                           0.0s
 => => writing image sha256:df4832476d11b2faf35fc5a03c8ea837b4e89deac4b306af28720d64eef3e207                      0.0s
 => => naming to docker.io/library/hind-hadoop-datanode                                                           0.0s
 => [hadoop-namenode] exporting to image                                                                          0.1s
 => => exporting layers                                                                                           0.1s
 => => writing image sha256:3d04317079f501f4326dbd932572f0ddb2dbfe6f5211b92ad2da3973a0fafda2                      0.0s
 => => naming to docker.io/library/hind-hadoop-namenode                                                           0.0s
 => [hadoop-historyserver] exporting to image                                                                     0.1s
 => => exporting layers                                                                                           0.0s
 => => writing image sha256:4c5f161a6cd0e11b181a5ec90519c7c29596236bea4e6e5239b4ee0c6cf6c468                      0.0s
 => => naming to docker.io/library/hind-hadoop-historyserver                                                      0.0s
 => [jupyter-notebook 12/16] RUN apt-get install -y python3-pip python3-dev                                       6.1s
 => [jupyter-notebook 13/16] RUN pip3 install --upgrade pip                                                       2.9s
 => [jupyter-notebook 14/16] RUN pip3 install wget==3.2 pyspark==3.0.0 jupyterlab==3.0.0                         36.7s
 => [jupyter-notebook 15/16] RUN pip3 install pandas                                                              9.2s
 => [jupyter-notebook 16/16] WORKDIR /workspace                                                                   0.0s
 => [jupyter-notebook] exporting to image                                                                         9.7s
 => => exporting layers                                                                                           9.7s
 => => writing image sha256:36377f21b4a1c2544af845cc439da5c0ecaa7597a5e2db1c90748e6dfd3ff365                      0.0s
 => => naming to docker.io/library/hind-jupyter-notebook                                                          0.0s
[+] Running 13/13
 ✔ Volume "hind_hadoop_historyserver"       Created                                                               0.0s
 ✔ Volume "hadoop-distributed-file-system"  Created                                                               0.0s
 ✔ Volume "hind_hadoop_namenode"            Created                                                               0.0s
 ✔ Volume "hind_hadoop_datanode"            Created                                                               0.0s
 ✔ Container hadoop-datanode                Started                                                               0.8s
 ✔ Container hadoop-namenode                Started                                                               0.9s
 ✔ Container hadoop-historyserver           Started                                                               0.7s
 ✔ Container spark-master                   Started                                                               0.9s
 ✔ Container hadoop-resourcemanager         Started                                                               0.7s
 ✔ Container hadoop-nodemanager-1           Started                                                               0.6s
 ✔ Container jupyter-notebook               Started                                                               1.1s
 ✔ Container spark-worker-1                 Started                                                               1.5s
 ✔ Container spark-worker-2                 Started                                                               1.6s
root@debian-16gb-nbg1-1:~/cloud/hind# docker ps -a
CONTAINER ID   IMAGE                         COMMAND                  CREATED          STATUS                            PORTS                                                                          NAMES
10e2e00d9710   hind-spark-worker-2           "/bin/sh -c 'bin/spa…"   18 seconds ago   Up 16 seconds                     0.0.0.0:8082->8081/tcp, :::8082->8081/tcp                                      spark-worker-2
5d476ae1b989   hind-spark-worker-1           "/bin/sh -c 'bin/spa…"   18 seconds ago   Up 16 seconds                     0.0.0.0:8081->8081/tcp, :::8081->8081/tcp                                      spark-worker-1
c46680841934   hind-spark-master             "/bin/sh -c 'bin/spa…"   18 seconds ago   Up 17 seconds                     0.0.0.0:7077->7077/tcp, :::7077->7077/tcp, 0.0.0.0:8080->8080/tcp, :::8080->8080/tcp   spark-master
1adb50e4231b   hind-hadoop-historyserver     "/entrypoint.sh /run…"   18 seconds ago   Up 17 seconds (health: starting)  8188/tcp                                                                       hadoop-historyserver
61b64aa98497   hind-hadoop-nodemanager-1     "/entrypoint.sh /run…"   18 seconds ago   Up 17 seconds (health: starting)  8042/tcp                                                                       hadoop-nodemanager-1
fc361643c774   hind-jupyter-notebook         "/bin/sh -c 'jupyter…"   18 seconds ago   Up 17 seconds                     0.0.0.0:4040->4040/tcp, :::4040->4040/tcp, 0.0.0.0:8888->8888/tcp, :::8888->8888/tcp   jupyter-notebook
867aea912365   hind-hadoop-datanode          "/entrypoint.sh /run…"   18 seconds ago   Up 17 seconds (health: starting)  9864/tcp                                                                       hadoop-datanode
c4df7083963f   hind-hadoop-resourcemanager   "/entrypoint.sh /run…"   18 seconds ago   Up 17 seconds (health: starting)  8088/tcp                                                                       hadoop-resourcemanag
er
66949b57a256   hind-hadoop-namenode          "/entrypoint.sh /run…"   18 seconds ago   Up 17 seconds (health: starting)  0.0.0.0:9000->9000/tcp, :::9000->9000/tcp, 0.0.0.0:9870->9870/tcp, :::9870->9870/tcp   hadoop-namenode
root@debian-16gb-nbg1-1:~/cloud/hind#
```


File Edit View Run Kernel Tabs Settings Help
spark.ipynb
Markdown
Python 3 (ipykernel)
Filter files by name
Name
Last Modified
data
an hour ago
spark.ipynb
5 minutes ago
Setup
[2]:
!pip install pyspark
!pip install matplotlib
!pip install pandas
!pip install findspark
!pip install pyarrow
Requirement already satisfied: pyspark in /usr/local/lib/python3.9/dist-packages (3.0.0)
Requirement already satisfied: py4j==0.10.9 in /usr/local/lib/python3.9/dist-packages (from pyspark) (0.10.9)
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.py
pa.io/warnings/venv
class="ansi-yellow-fg">
Collecting matplotlib
Downloading matplotlib-3.9.0-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (11 kB)
Collecting contourpy>=1.0.1 (from matplotlib)
Downloading contourpy-1.2.1-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (5.8 kB)
Collecting cycler>=0.10 (from matplotlib)
Downloading cycler-0.12.1-py3-none-any.whl.metadata (3.8 kB)
Collecting fonttools>=4.22.0 (from matplotlib)
Downloading fonttools-4.52.4-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (161 kB)
161.7/161.7 kB 8.3 MB/s eta 0:00:00
Collecting kiwisolver>=1.3.1 (from matplotlib)
Downloading kiwisolver-1.4.5-cp39-cp39-manylinux_2_12_x86_64.manylinux2010_x86_64.whl.metadata (6.4 kB)
Requirement already satisfied: numpy>=1.23 in /usr/local/lib/python3.9/dist-packages (from matplotlib) (1.26.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.9/dist-packages (from matplotlib) (24.0)
Collecting pillow>=8 (from matplotlib)
Downloading pillow-10.3.0-cp39-cp39-manylinux_2_28_x86_64.whl.metadata (9.2 kB)
Collecting pyparsing>=2.3.1 (from matplotlib)
Downloading pyparsing-3.1.2-py3-none-any.whl.metadata (5.1 kB)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.9/dist-packages (from matplotlib) (2.9.0.post0)
Collecting importlib-resources>=3.2.0 (from matplotlib)
Downloading importlib_resources-6.4.0-py3-none-any.whl.metadata (3.9 kB)
Requirement already satisfied: zipp>=3.1.0 in /usr/local/lib/python3.9/dist-packages (from importlib-resources>=3.2.0->matplotlib) (3.19.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.9/dist-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
Downloading matplotlib-3.9.0-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (8.3 MB)
8.3/8.3 MB 74.1 MB/s eta 0:00:00:00:0100:01
Downloading contourpy-1.2.1-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (304 kB)
305.0/305.0 kB 34.7 MB/s eta 0:00:00
Downloading cycler-0.12.1-py3-none-any.whl (8.3 kB)
Downloading fonttools-4.52.4-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (4.6 MB)
4.6/4.6 MB 77.2 MB/s eta 0:00:00ta 0:00:01
Downloading importlib_resources-6.4.0-py3-none-any.whl (38 kB)
Downloading kiwisolver-1.4.5-cp39-cp39-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (1.6 MB)
1.6/1.6 MB 57.4 MB/s eta 0:00:00
Downloading pillow-10.3.0-cp39-cp39-manylinux_2_28_x86_64.whl (4.5 MB)
4.5/4.5 MB 64.6 MB/s eta 0:00:00ta 0:00:01
Downloading pyparsing-3.1.2-py3-none-any.whl (103 kB)


File Edit View Run Kernel Tabs Settings Help
spark.ipynb
Markdown
Python 3 (ipykernel)
Filter files by name
Name
Last Modified
data
an hour ago
spark.ipynb
5 minutes ago
Downloading importlib_resources-6.4.0-py3-none-any.whl (38 kB)
Downloading kiwisolver-1.4.5-cp39-cp39-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (1.6 MB)
1.6/1.6 MB 57.4 MB/s eta 0:00:00
Downloading pillow-10.3.0-cp39-cp39-manylinux_2_28_x86_64.whl (4.5 MB)
4.5/4.5 MB 64.6 MB/s eta 0:00:00ta 0:00:01
Downloading pyparsing-3.1.2-py3-none-any.whl (103 kB)
103.2/103.2 kB 15.6 MB/s eta 0:00:00
Installing collected packages: pyparsing, pillow, kiwisolver, importlib-resources, fonttools, cycler, contourpy, matplotlib
Successfully installed contourpy-1.2.1 cycler-0.12.1 fonttools-4.52.4 importlib-resources-6.4.0 kiwisolver-1.4.5 matplotlib-3.9.0 pillow-10.3.0 pyparsing-3.1.2
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.py
pa.io/warnings/venv class="ansi-yellow-fg">
Requirement already satisfied: pandas in /usr/local/lib/python3.9/dist-packages (2.2.2)
Requirement already satisfied: numpy>=1.22.4 in /usr/local/lib/python3.9/dist-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.9/dist-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.9/dist-packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.9/dist-packages (from pandas) (2024.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.9/dist-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.py
pa.io/warnings/venv class="ansi-yellow-fg">
Collecting findspark
Downloading findspark-2.0.1-py2.py3-none-any.whl.metadata (352 bytes)
Downloading findspark-2.0.1-py2.py3-none-any.whl (4.4 kB)
Installing collected packages: findspark
Successfully installed findspark-2.0.1
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.py
pa.io/warnings/venv class="ansi-yellow-fg">
Collecting pyarrow
Downloading pyarrow-16.1.0-cp39-cp39-manylinux_2_28_x86_64.whl.metadata (3.0 kB)
Requirement already satisfied: numpy>=1.16.6 in /usr/local/lib/python3.9/dist-packages (from pyarrow) (1.26.4)
Downloading pyarrow-16.1.0-cp39-cp39-manylinux_2_28_x86_64.whl (40.8 MB)
40.8/40.8 MB 47.0 MB/s eta 0:00:00:00:0100:01
Installing collected packages: pyarrow
Successfully installed pyarrow-16.1.0
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.py
pa.io/warnings/venv class="ansi-yellow-fg">
[4]:
from pyspark import SparkContext, SparkConf
from pyspark.sql import SparkSession
from pyspark.sql.types import StructType, StructField, IntegerType, StringType, FloatType
from pyspark.sql.functions import *
import pandas as pd
import matplotlib.pyplot as plt
Spark Context and Spark Session


File Edit View Run Kernel Tabs Settings Help
spark.ipynb
Markdown
Python 3 (ipykernel)
Filter files by name
/
Name
Last Modified
data
an hour ago
spark.ipynb
seconds ago
DataFrames
A DataFrame is two-dimensional. Columns can be of different data types. DataFrames accept many data inputs including series and other DataFrames. You can pass indexes (row labels) and columns (column labels). Indexes can be numbers, dates, or strings/tuples.
Load the data into Spark dataframe
[48]:
schema = StructType([
StructField("work_year", IntegerType(), True),
StructField("experience_level", StringType(), True),
StructField("employment_type", StringType(), True),
StructField("job_title", StringType(), True),
StructField("salary", IntegerType(), True),
StructField("salary_currency", StringType(), True),
StructField("salary_in_usd", IntegerType(), True),
StructField("employee_residence", StringType(), True),
StructField("remote_ratio", IntegerType(), True),
StructField("company_location", StringType(), True),
StructField("company_size", StringType(), True)
])
[49]:
# Convert Pandas DataFrame to Spark DataFrame with defined schema
salaries_df = spark.createDataFrame(data_engineer_salary.values.tolist(), schema=schema)
# Show the DataFrame schema and some data
salaries_df.printSchema()
root
|-- work_year: integer (nullable = true)
|-- experience_level: string (nullable = true)
|-- employment_type: string (nullable = true)
|-- job_title: string (nullable = true)
|-- salary: integer (nullable = true)
|-- salary_currency: string (nullable = true)
|-- salary_in_usd: integer (nullable = true)
|-- employee_residence: string (nullable = true)
|-- remote_ratio: integer (nullable = true)
|-- company_location: string (nullable = true)
|-- company_size: string (nullable = true)
Basic data analysis and manipulation


File Edit View Run Kernel Tabs Settings Help
spark.ipynb
Markdown
Python 3 (ipykernel)
Filter files by name
/
Name
Last Modified
data
an hour ago
spark.ipynb
6 minutes ago
[42]:
from pyspark import SparkContext, SparkConf
from pyspark.sql import SparkSession
from pyspark.sql.types import StructType, StructField, IntegerType, StringType, FloatType
from pyspark.sql.functions import *
import pandas as pd
import matplotlib.pyplot as plt
Spark Context and Spark Session
You will create the Spark Context and initialize the Spark session needed for SparkSQL and DataFrames. SparkContext is the entry point for Spark applications and contains functions to create RDDs such as parallelize() . SparkSession is needed for SparkSQL and DataFrame operations.
[43]:
# Creating a spark context class
sc = SparkContext()
# # Creating a spark session
spark = SparkSession.builder.appName("pyspark-notebook").master("spark://spark-master:7077").config("spark.executor.memory", "1024m").getOrCreate()
[44]:
spark
[44]:
SparkSession - in-memory
SparkContext
Spark UI
Version v3.0.0
Master local[*]
AppName pyspark-shell
Initialize Spark session
To work with dataframes we just need to verify that the spark session instance has been created.
[45]:
if 'spark' in locals() and isinstance(spark, SparkSession):
print("SparkSession is active and ready to use.")
else:
print("SparkSession is not active. Please create a SparkSession.")
SparkSession is active and ready to use.

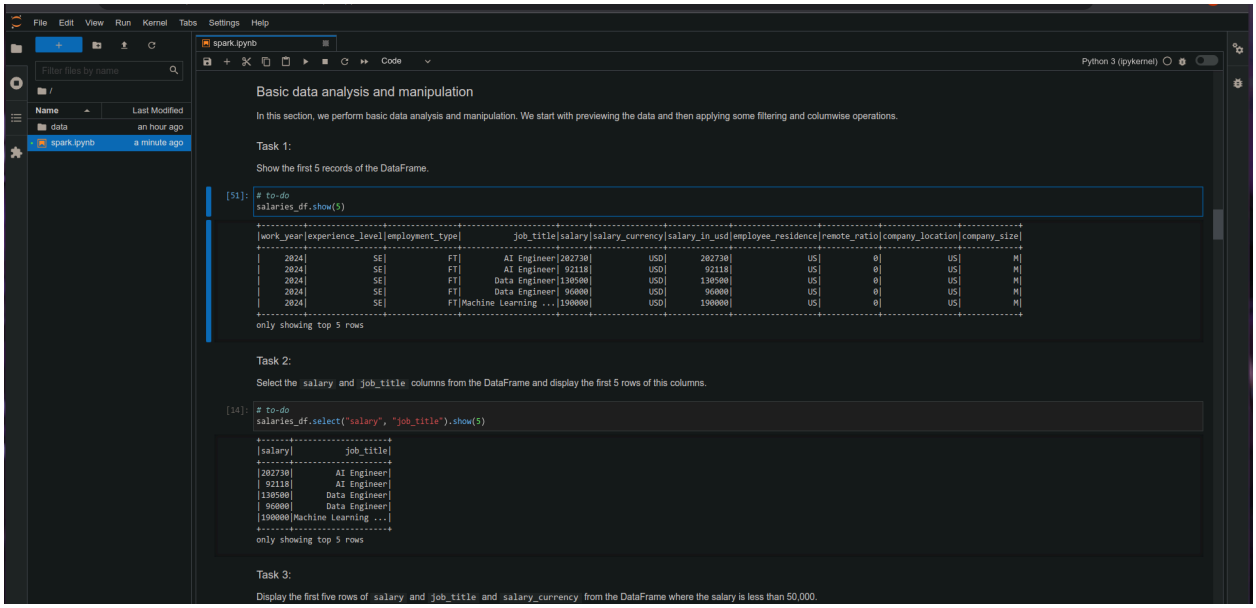


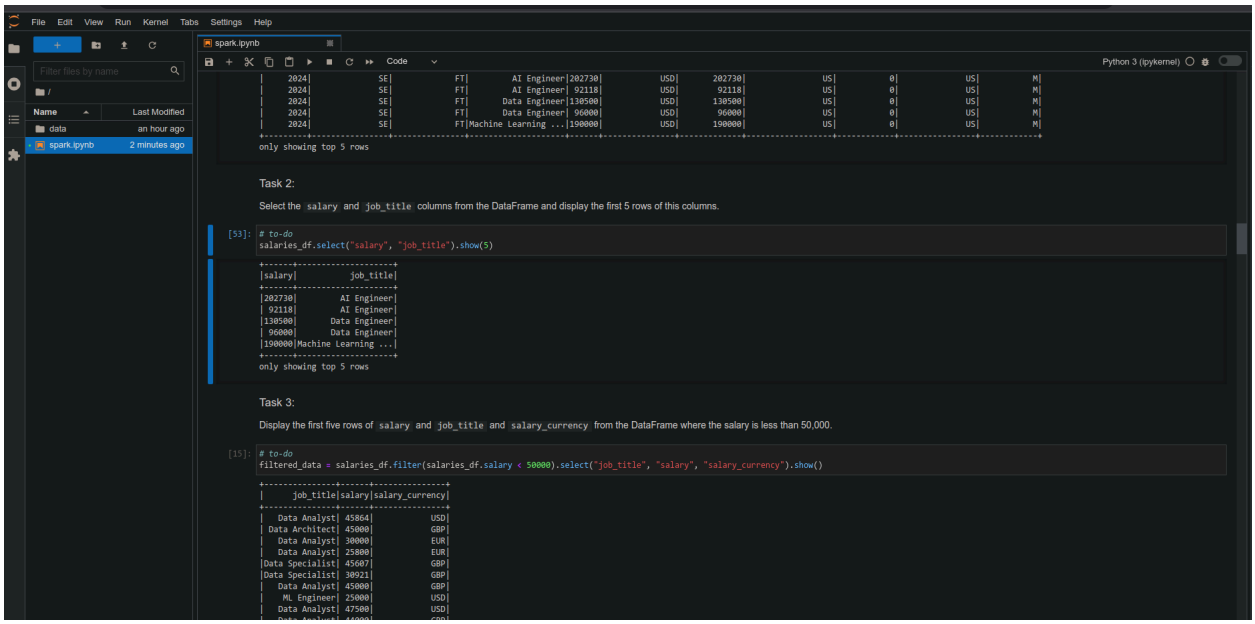
	work_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd	employee_residence	remote_ratio	company_location	company_size
0	2024	SE	FT	AI Engineer	202730	USD	202730	US	0	US	M
1	2024	SE	FT	AI Engineer	92118	USD	92118	US	0	US	M
2	2024	SE	FT	Data Engineer	130500	USD	130500	US	0	US	M
3	2024	SE	FT	Data Engineer	96000	USD	96000	US	0	US	M
4	2024	SE	FT	Machine Learning Engineer	190000	USD	190000	US	0	US	M
5	2024	SE	FT	Machine Learning Engineer	160000	USD	160000	US	0	US	M
6	2024	MI	FT	ML Engineer	400000	USD	400000	US	0	US	M
7	2024	MI	FT	ML Engineer	65000	USD	65000	US	0	US	M
8	2024	EN	FT	Data Analyst	101520	USD	101520	US	0	US	M
9	2024	EN	FT	Data Analyst	45864	USD	45864	US	0	US	M

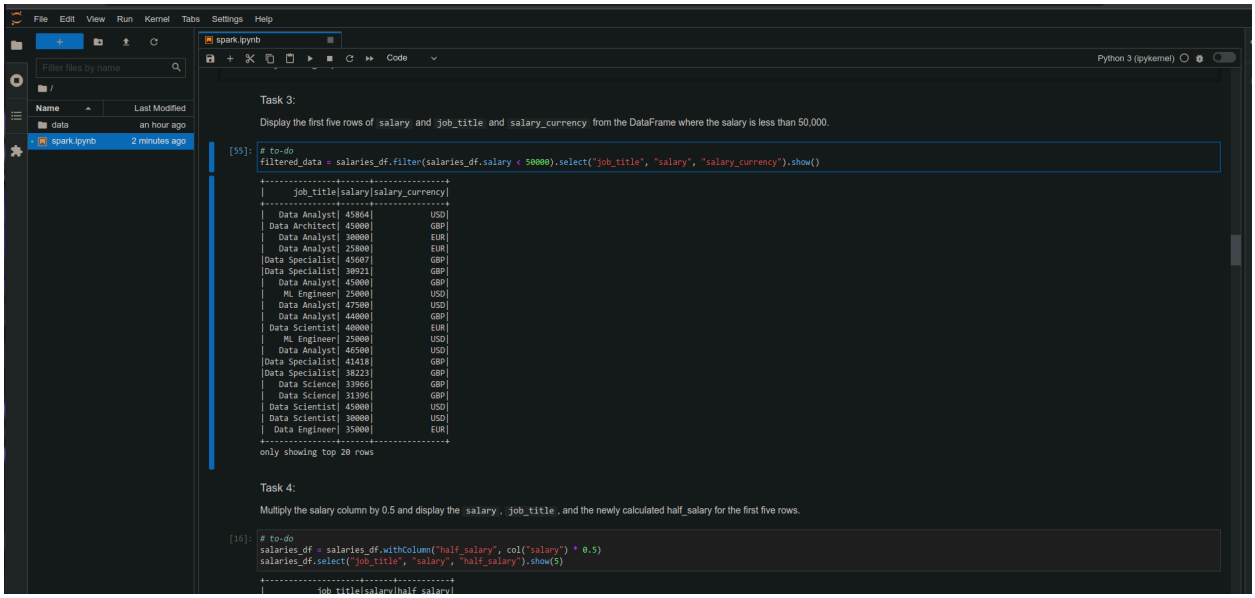
DataFrames

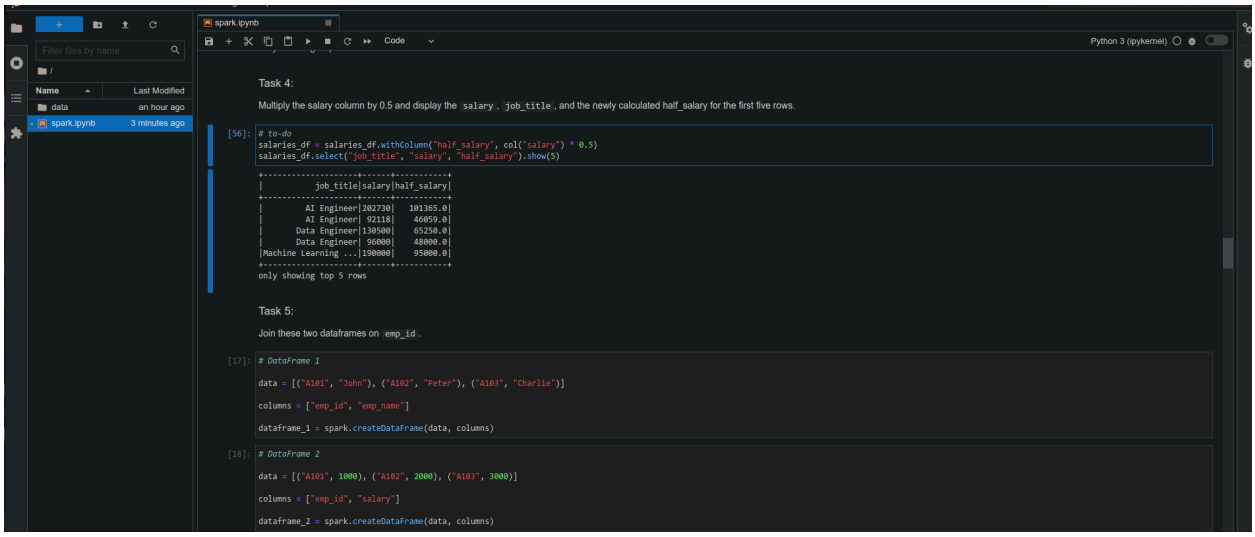
A DataFrame is two-dimensional. Columns can be of different data types. DataFrames accept many data inputs including series and other DataFrames. You can pass indexes (row labels) and columns (column labels). Indexes can be numbers, dates, or strings/tuples.

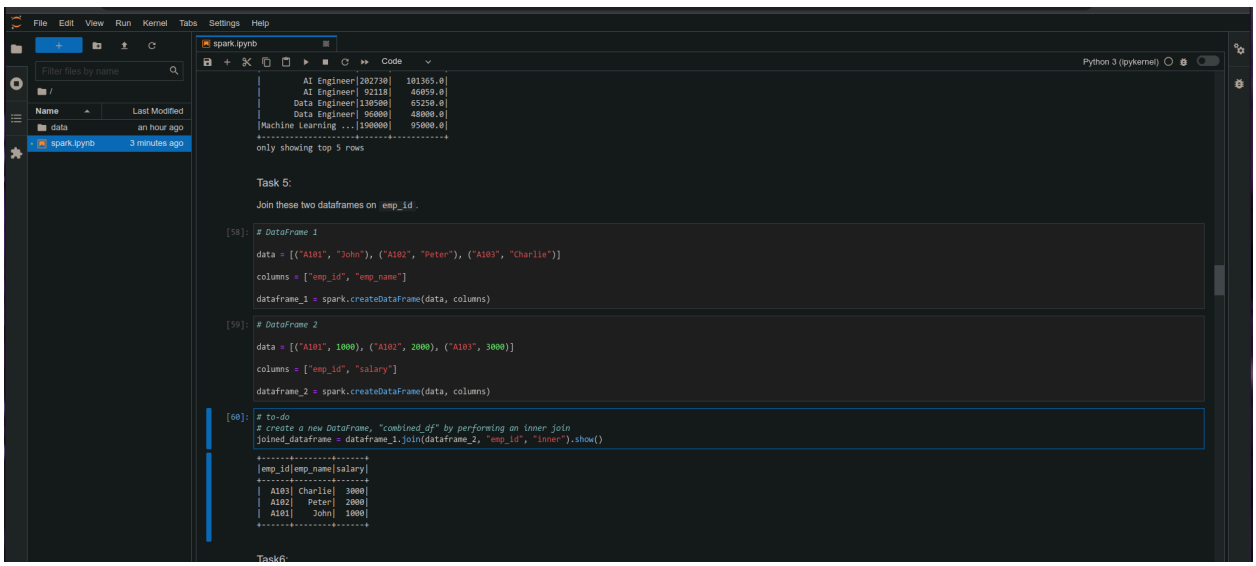
Load the data into Spark dataframe

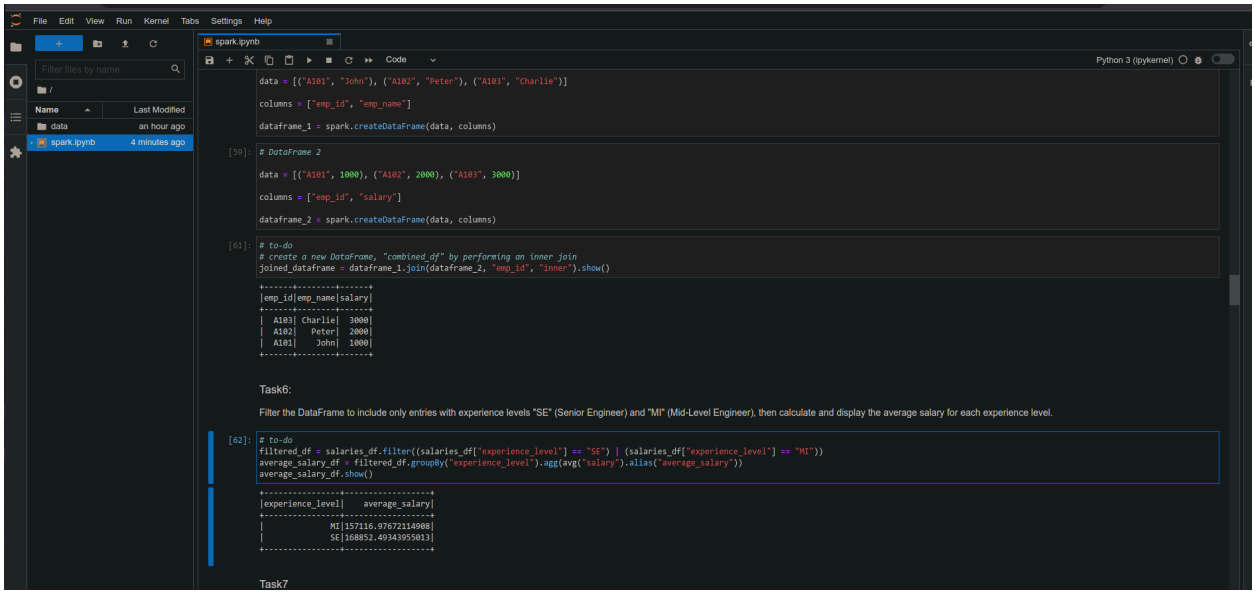

File Edit View Run Kernel Tabs Settings Help
spark.ipynb
Code
Python 3 (ipykernel)
Filter files by name
Name
Last Modified
data
an hour ago
spark.ipynb
a minute ago
Basic data analysis and manipulation
In this section, we perform basic data analysis and manipulation. We start with previewing the data and then applying some filtering and columwise operations.
Task 1:
Show the first 5 records of the DataFrame.
[51]:
# to-do
salaries_df.show(5)
+----------+----------------+---------------+--------------------+------+-------------+-------------+----------------+-----------+----------------+------------+
|work_year|experience_level|employment_type| job_title|salary|salary_currency|salary_in_usd|employee_residence|remote_ratio|company_location|company_size|
+----------+----------------+---------------+--------------------+------+-------------+-------------+----------------+-----------+----------------+------------+
2024	SE	FT	AI Engineer	202730	USD	202730	US	0	US	M
2024	SE	FT	AI Engineer	92118	USD	92118	US	0	US	M
2024	SE	FT	Data Engineer	130500	USD	130500	US	0	US	M
2024	SE	FT	Data Engineer	96000	USD	96000	US	0	US	M
2024	SE	FT	Machine Learning ...	190000	USD	190000	US	0	US	M
+----------+----------------+---------------+--------------------+------+-------------+-------------+----------------+-----------+----------------+------------+
only showing top 5 rows
Task 2:
Select the salary and job_title columns from the DataFrame and display the first 5 rows of this columns.
[14]:
# to-do
salaries_df.select("salary", "job_title").show(5)
+------+--------------------+
|salary| job_title|
+------+--------------------+
|202730| AI Engineer|
| 92118| AI Engineer|
|130500| Data Engineer|
| 96000| Data Engineer|
|190000|Machine Learning ...|
+------+--------------------+
only showing top 5 rows
Task 3:
Display the first five rows of salary and job_title and salary_currency from the DataFrame where the salary is less than 50,000.

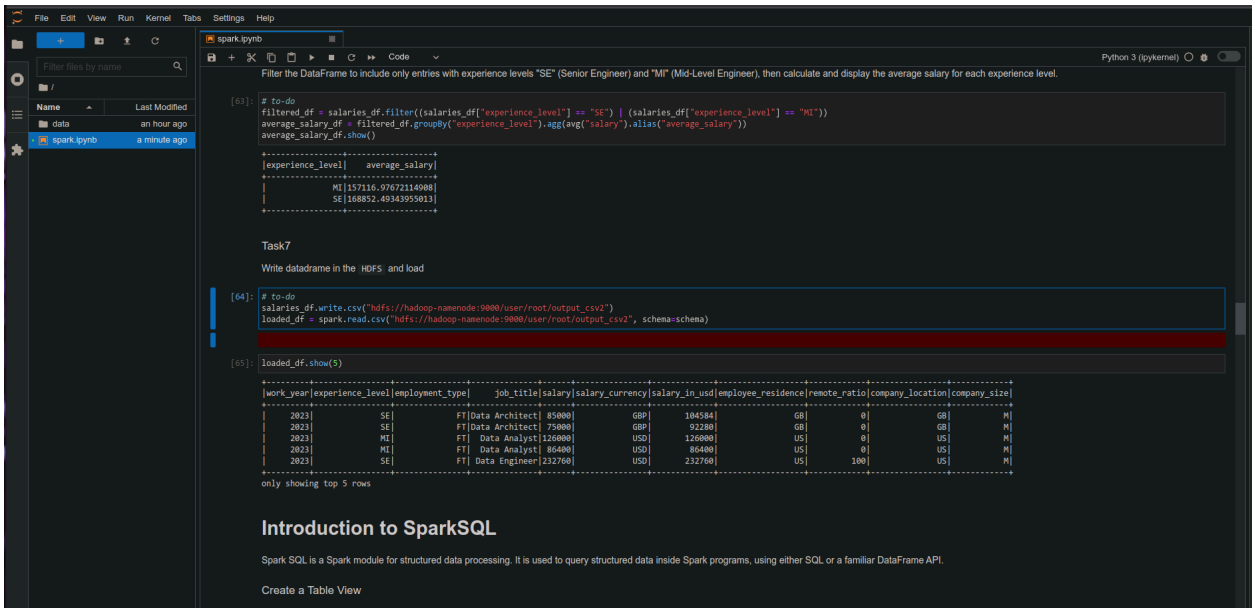

File Edit View Run Kernel Tabs Settings Help
spark.ipynb
Code
Python 3 (ipykernel)
Filter files by name
Name
Last Modified
data
an hour ago
spark.ipynb
2 minutes ago
2024	SE	FT	AI Engineer	202730	USD	202730	US	0	US	M
2024	SE	FT	AI Engineer	92118	USD	92118	US	0	US	M
2024	SE	FT	Data Engineer	130500	USD	130500	US	0	US	M
2024	SE	FT	Data Engineer	96000	USD	96000	US	0	US	M
2024	SE	FT	Machine Learning ...	190000	USD	190000	US	0	US	M
+----------+----------------+---------------+--------------------+------+-------------+-------------+----------------+-----------+----------------+------------+
only showing top 5 rows
Task 2:
Select the salary and job_title columns from the DataFrame and display the first 5 rows of this columns.
[53]:
# to-do
salaries_df.select("salary", "job_title").show(5)
+------+--------------------+
|salary| job_title|
+------+--------------------+
|202730| AI Engineer|
| 92118| AI Engineer|
|130500| Data Engineer|
| 96000| Data Engineer|
|190000|Machine Learning ...|
+------+--------------------+
only showing top 5 rows
Task 3:
Display the first five rows of salary and job_title and salary_currency from the DataFrame where the salary is less than 50,000.
[15]:
# to-do
filtered_data = salaries_df.filter(salaries_df.salary < 50000).select("job_title", "salary", "salary_currency").show()
+---------------+------+--------------+
| job_title|salary|salary_currency|
+---------------+------+--------------+
Data Analyst	45864	USD
Data Architect	45000	GBP
Data Analyst	30000	EUR
Data Analyst	25800	EUR
Data Specialist	45607	GBP
Data Specialist	30921	GBP
Data Analyst	45000	GBP
ML Engineer	25000	USD
Data Analyst	47500	USD

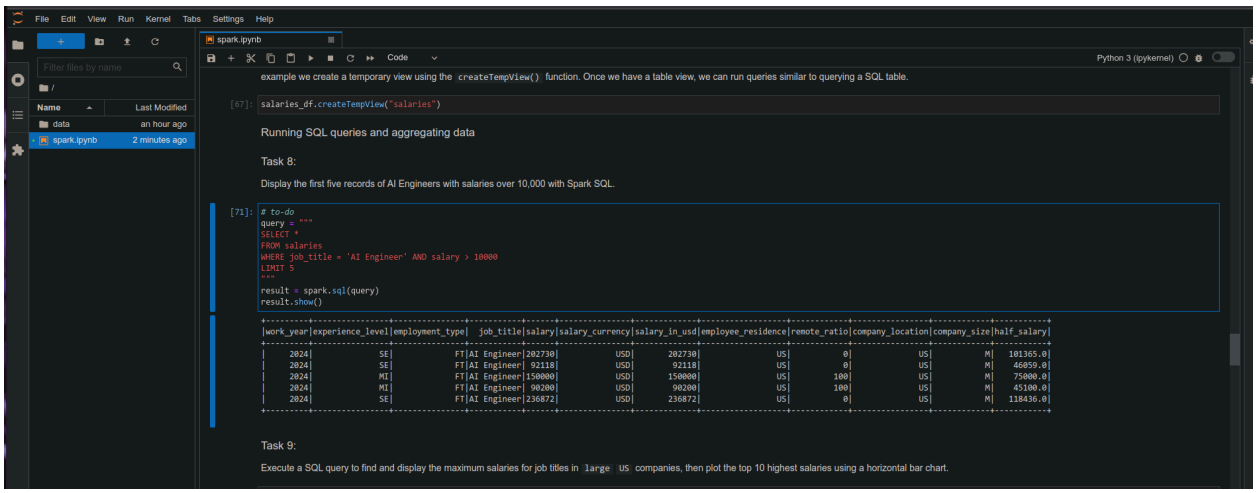

File Edit View Run Kernel Tabs Settings Help
spark.ipynb
Code
Python 3 (ipykernel)
Filter files by name
Name
Last Modified
data
an hour ago
spark.ipynb
2 minutes ago
Task 3:
Display the first five rows of salary and job_title and salary_currency from the DataFrame where the salary is less than 50,000.
[55]:
# to-do
filtered_data = salaries_df.filter(salaries_df.salary < 50000).select("job_title", "salary", "salary_currency").show()
+---------------+------+---------------+
| job_title|salary|salary_currency|
+---------------+------+---------------+
Data Analyst	45864	USD
Data Architect	45000	GBP
Data Analyst	30000	EUR
Data Analyst	25800	EUR
Data Specialist	45607	GBP
Data Specialist	30921	GBP
Data Analyst	45000	GBP
ML Engineer	25000	USD
Data Analyst	47500	USD
Data Analyst	44000	GBP
Data Scientist	40000	EUR
ML Engineer	25000	USD
Data Analyst	46500	USD
Data Specialist	41418	GBP
Data Specialist	38223	GBP
Data Science	33966	GBP
Data Science	31396	GBP
Data Scientist	45000	USD
Data Scientist	30000	USD
Data Engineer	35000	EUR
+---------------+------+---------------+
only showing top 20 rows
Task 4:
Multiply the salary column by 0.5 and display the salary , job_title , and the newly calculated half_salary for the first five rows.
[16]:
# to-do
salaries_df = salaries_df.withColumn("half_salary", col("salary") * 0.5)
salaries_df.select("job_title", "salary", "half_salary").show(5)
+--------------------+------+-----------+
| job_title|salary|half_salary|


File Edit View Run Kernel Tabs Settings Help
spark.ipynb
Code
Python 3 (ipykernel)
Filter files by name
Name
Last Modified
data
an hour ago
spark.ipynb
3 minutes ago
Task 4:
Multiply the salary column by 0.5 and display the salary , job_title , and the newly calculated half_salary for the first five rows.
[56]:
# to-do
salaries_df = salaries_df.withColumn("half_salary", col("salary") * 0.5)
salaries_df.select("job_title", "salary", "half_salary").show(5)
+--------------------+------+-----------+
| job_title|salary|half_salary|
+--------------------+------+-----------+
AI Engineer	202730	101365.0
AI Engineer	92118	46059.0
Data Engineer	130500	65250.0
Data Engineer	96000	48000.0
Machine Learning ...	190000	95000.0
+--------------------+------+-----------+
only showing top 5 rows
Task 5:
Join these two dataframes on emp_id .
[17]:
# DataFrame 1
data = [("A101", "John"), ("A102", "Peter"), ("A103", "Charlie")]
columns = ["emp_id", "emp_name"]
dataframe_1 = spark.createDataFrame(data, columns)
[18]:
# DataFrame 2
data = [("A101", 1000), ("A102", 2000), ("A103", 3000)]
columns = ["emp_id", "salary"]
dataframe_2 = spark.createDataFrame(data, columns)


File Edit View Run Kernel Tabs Settings Help
spark.ipynb
Code
Python 3 (ipykernel)
Filter files by name
Name
Last Modified
data
an hour ago
spark.ipynb
3 minutes ago
AI Engineer	202730	101365.0
AI Engineer	92118	46059.0
Data Engineer	130500	65250.0
Data Engineer	96000	48000.0
Machine Learning ...	190000	95000.0
+--------------------+------+-----------+
only showing top 5 rows
Task 5:
Join these two dataframes on emp_id .
[58]:
# DataFrame 1
data = [("A101", "John"), ("A102", "Peter"), ("A103", "Charlie")]
columns = ["emp_id", "emp_name"]
dataframe_1 = spark.createDataFrame(data, columns)
[59]:
# DataFrame 2
data = [("A101", 1000), ("A102", 2000), ("A103", 3000)]
columns = ["emp_id", "salary"]
dataframe_2 = spark.createDataFrame(data, columns)
[60]:
# to-do
# create a new DataFrame, "combined_df" by performing an inner join
joined_dataframe = dataframe_1.join(dataframe_2, "emp_id", "inner").show()
+------+--------+------+
|emp_id|emp_name|salary|
+------+--------+------+
A103	Charlie	3000
A102	Peter	2000
A101	John	1000
+------+--------+------+
Task6:


File Edit View Run Kernel Tabs Settings Help
spark.ipynb
Code
Python 3 (ipykernel)
Filter files by name
Name
Last Modified
data
an hour ago
spark.ipynb
4 minutes ago
data = [("A101", "John"), ("A102", "Peter"), ("A103", "Charlie")]
columns = ["emp_id", "emp_name"]
dataframe_1 = spark.createDataFrame(data, columns)
[59]:
# DataFrame 2
data = [("A101", 1000), ("A102", 2000), ("A103", 3000)]
columns = ["emp_id", "salary"]
dataframe_2 = spark.createDataFrame(data, columns)
[61]:
# to-do
# create a new DataFrame, "combined_df" by performing an inner join
joined_dataframe = dataframe_1.join(dataframe_2, "emp_id", "inner").show()
+------+--------+------+
|emp_id|emp_name|salary|
+------+--------+------+
A103	Charlie	3000
A102	Peter	2000
A101	John	1000
+------+--------+------+
Task6:
Filter the DataFrame to include only entries with experience levels "SE" (Senior Engineer) and "MI" (Mid-Level Engineer), then calculate and display the average salary for each experience level.
[62]:
# to-do
filtered_df = salaries_df.filter((salaries_df["experience_level"] == "SE") | (salaries_df["experience_level"] == "MI"))
average_salary_df = filtered_df.groupBy("experience_level").agg(avg("salary").alias("average_salary"))
average_salary_df.show()
+----------------+------------------+
|experience_level| average_salary|
+----------------+------------------+
| MI|157116.97672114908|
| SE|168852.49343955013|
+----------------+------------------+
Task7


File Edit View Run Kernel Tabs Settings Help
spark.ipynb
Code
Python 3 (ipykernel)
Filter files by name
Name
Last Modified
data
an hour ago
spark.ipynb
a minute ago
Filter the DataFrame to include only entries with experience levels "SE" (Senior Engineer) and "MI" (Mid-Level Engineer), then calculate and display the average salary for each experience level.
[63]:
# to-do
filtered_df = salaries_df.filter((salaries_df["experience_level"] == "SE") | (salaries_df["experience_level"] == "MI"))
average_salary_df = filtered_df.groupBy("experience_level").agg(avg("salary").alias("average_salary"))
average_salary_df.show()
+----------------+------------------+
|experience_level| average_salary|
+----------------+------------------+
| MI|157116.97672114908|
| SE|168852.49343955013|
+----------------+------------------+
Task7
Write datadrame in the HDFS and load
[64]:
# to-do
salaries_df.write.csv("hdfs://hadoop-namenode:9000/user/root/output_csv2")
loaded_df = spark.read.csv("hdfs://hadoop-namenode:9000/user/root/output_csv2", schema=schema)
[65]:
loaded_df.show(5)
+---------+----------------+---------------+--------------+------+---------------+-------------+------------------+-----------+----------------+------------+
|work_year|experience_level|employment_type| job_title|salary|salary_currency|salary_in_usd|employee_residence|remote_ratio|company_location|company_size|
+---------+----------------+---------------+--------------+------+---------------+-------------+------------------+-----------+----------------+------------+
2023	SE	FT	Data Architect	85000	GBP	104584	GB	0	GB	M
2023	SE	FT	Data Architect	75000	GBP	92280	GB	0	GB	M
2023	MI	FT	Data Analyst	126000	USD	126000	US	0	US	M
2023	MI	FT	Data Analyst	86400	USD	86400	US	0	US	M
2023	SE	FT	Data Engineer	232760	USD	232760	US	100	US	M
+---------+----------------+---------------+--------------+------+---------------+-------------+------------------+-----------+----------------+------------+
only showing top 5 rows
Introduction to SparkSQL
Spark SQL is a Spark module for structured data processing. It is used to query structured data inside Spark programs, using either SQL or a familiar DataFrame API.
Create a Table View


File Edit View Run Kernel Tabs Settings Help
spark.ipynb
Code
Python 3 (ipykernel)
Filter files by name
Name
Last Modified
data
an hour ago
spark.ipynb
2 minutes ago
example we create a temporary view using the createTempView() function. Once we have a table view, we can run queries similar to querying a SQL table.
[67]:
salaries_df.createTempView("salaries")
Running SQL queries and aggregating data
Task 8:
Display the first five records of AI Engineers with salaries over 10,000 with Spark SQL.
[71]:
# to-do
query = """
SELECT *
FROM salaries
WHERE job_title = 'AI Engineer' AND salary > 10000
LIMIT 5
"""
result = spark.sql(query)
result.show()
+---------+----------------+---------------+-----------+------+---------------+-------------+------------------+-----------+----------------+------------+-----------+
|work_year|experience_level|employment_type| job_title|salary|salary_currency|salary_in_usd|employee_residence|remote_ratio|company_location|company_size|half_salary|
+---------+----------------+---------------+-----------+------+---------------+-------------+------------------+-----------+----------------+------------+-----------+
2024	SE	FT	AI Engineer	202730	USD	202730	US	0	US	M	101365.0
2024	SE	FT	AI Engineer	92118	USD	92118	US	0	US	M	46059.0
2024	MI	FT	AI Engineer	150000	USD	150000	US	100	US	M	75000.0
2024	MI	FT	AI Engineer	90200	USD	90200	US	100	US	M	45100.0
2024	SE	FT	AI Engineer	236872	USD	236872	US	0	US	M	118436.0
+---------+----------------+---------------+-----------+------+---------------+-------------+------------------+-----------+----------------+------------+-----------+
Task 9:
Execute a SQL query to find and display the maximum salaries for job titles in large US companies, then plot the top 10 highest salaries using a horizontal bar chart.

File Edit View Run Kernel Tabs Settings Help

spark.ipynb

Markdown

Python 3 (ipykernel)

Filter files by name

/

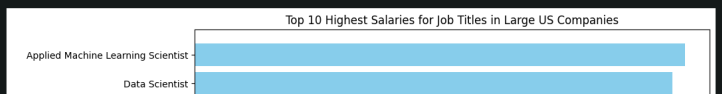
Name	Last Modified
data	an hour ago
spark.ipynb	2 minutes ago

Task 9:

Execute a SQL query to find and display the maximum salaries for job titles in `large` `US` companies, then plot the top 10 highest salaries using a horizontal bar chart.

[71]:
```
# to-do

filtered_df = spark.sql("SELECT job_title, MAX(salary_in_usd) AS max_salary_in_usd FROM salaries WHERE company_size = 'L' AND company_location = 'US' GROUP BY job_title")

top_10_salaries = filtered_df.orderBy('max_salary_in_usd', ascending=False).limit(10)
top_10_salaries.show()

import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
plt.barh(top_10_salaries.toPandas()['job_title'], top_10_salaries.toPandas()['max_salary_in_usd'], color='skyblue')
plt.xlabel('Max Salary in USD')
plt.ylabel('Job Title')
plt.title('Top 10 Highest Salaries for Job Titles in Large US Companies')
plt.gca().invert_yaxis()
plt.show()
```

```
+--------------------+----------------+
|           job_title|max_salary_in_usd|
+--------------------+----------------+
|Applied Machine L...|          423000|
|      Data Scientist|          412000|
| Data Analytics Lead|          405000|
|  Research Scientist|          405000|
|Applied Data Scie...|          380000|
|Data Science Tech...|          375000|
|   Applied Scientist|          350000|
|Machine Learning ...|          342300|
|Director of Data ...|          325000|
|      Data Science  |          324100|
+--------------------+----------------+
```



File Edit View Run Kernel Tabs Settings Help

spark.ipynb

Markdown

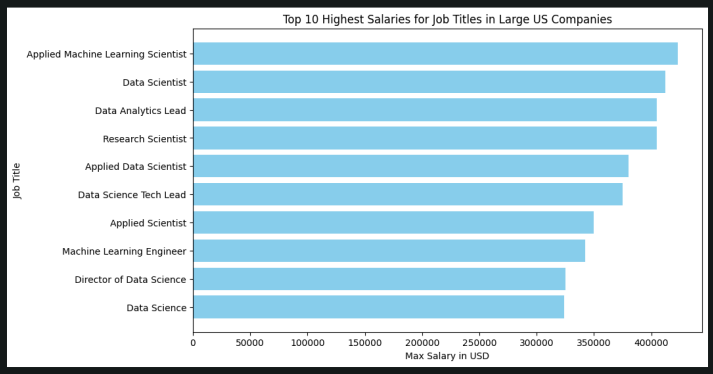
Python 3 (ipykernel)

Filter files by name

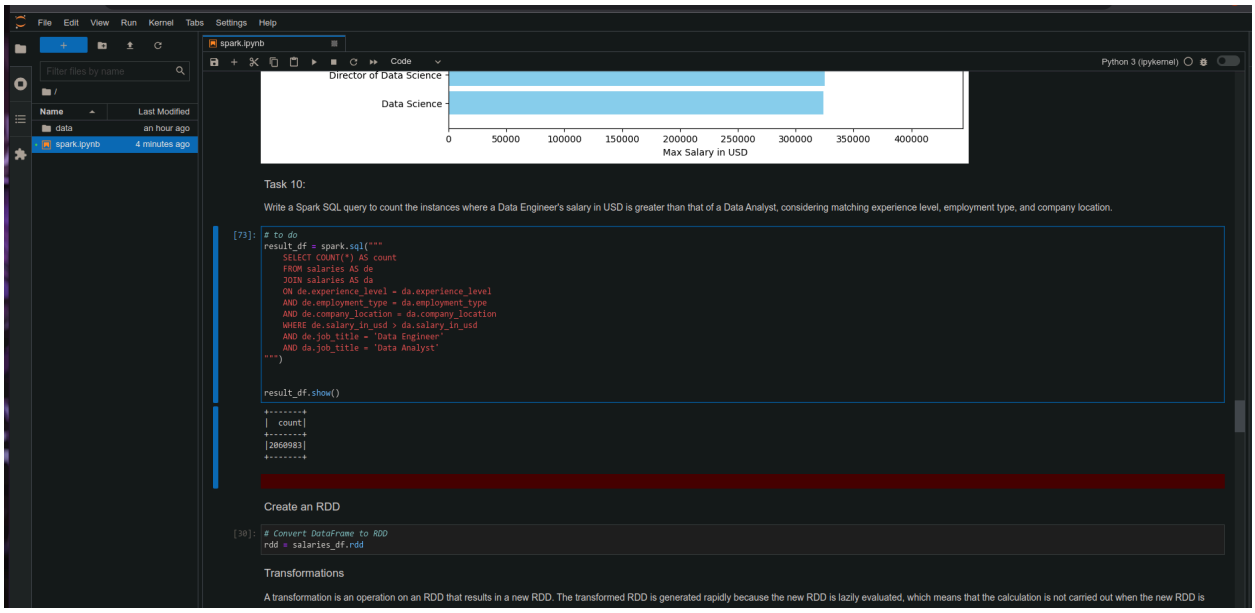
/

Name	Last Modified
data	an hour ago
spark.ipynb	3 minutes ago

Data Scientist	412000
Data Analytics Lead	405000
Research Scientist	405000
Applied Data Scie...	380000
Data Science Tech...	375000
Applied Scientist	350000
Machine Learning ...	342300
Director of Data ...	325000
Data Science	324100
+--------------------+----------------+



Task 10:



Task 10:

Write a Spark SQL query to count the instances where a Data Engineer's salary in USD is greater than that of a Data Analyst, considering matching experience level, employment type, and company location.

[73]:
```
# to do
result_df = spark.sql("""
    SELECT COUNT(*) AS count
    FROM salaries AS de
    JOIN salaries AS da
    ON de.experience_level = da.experience_level
    AND de.employment_type = da.employment_type
    AND de.company_location = da.company_location
    WHERE de.salary_in_usd > da.salary_in_usd
    AND de.job_title = 'Data Engineer'
    AND da.job_title = 'Data Analyst'
""")


result_df.show()
```

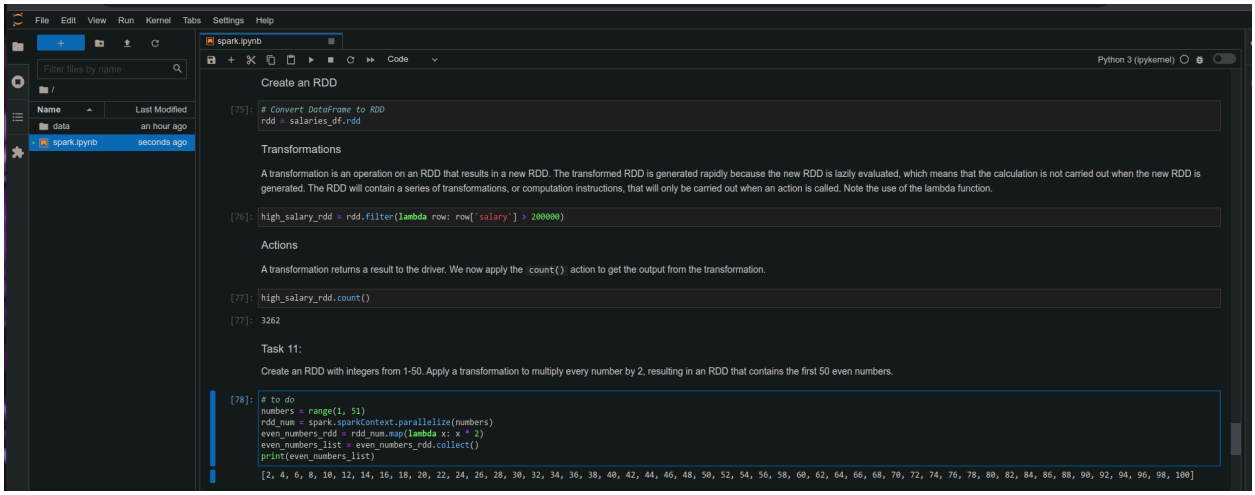
```
+-------+
|  count|
+-------+
|2060983|
+-------+
```

Create an RDD

[30]:
```
# Convert DataFrame to RDD
rdd = salaries_df.rdd
```

Transformations

A transformation is an operation on an RDD that results in a new RDD. The transformed RDD is generated rapidly because the new RDD is lazily evaluated, which means that the calculation is not carried out when the new RDD is



Create an RDD

[75]:
```
# Convert DataFrame to RDD
rdd = salaries_df.rdd
```

Transformations

A transformation is an operation on an RDD that results in a new RDD. The transformed RDD is generated rapidly because the new RDD is lazily evaluated, which means that the calculation is not carried out when the new RDD is generated. The RDD will contain a series of transformations, or computation instructions, that will only be carried out when an action is called. Note the use of the lambda function.

[76]:
```
high_salary_rdd = rdd.filter(lambda row: row['salary'] > 200000)
```

Actions

A transformation returns a result to the driver. We now apply the `count()` action to get the output from the transformation.

[77]:
```
high_salary_rdd.count()
```

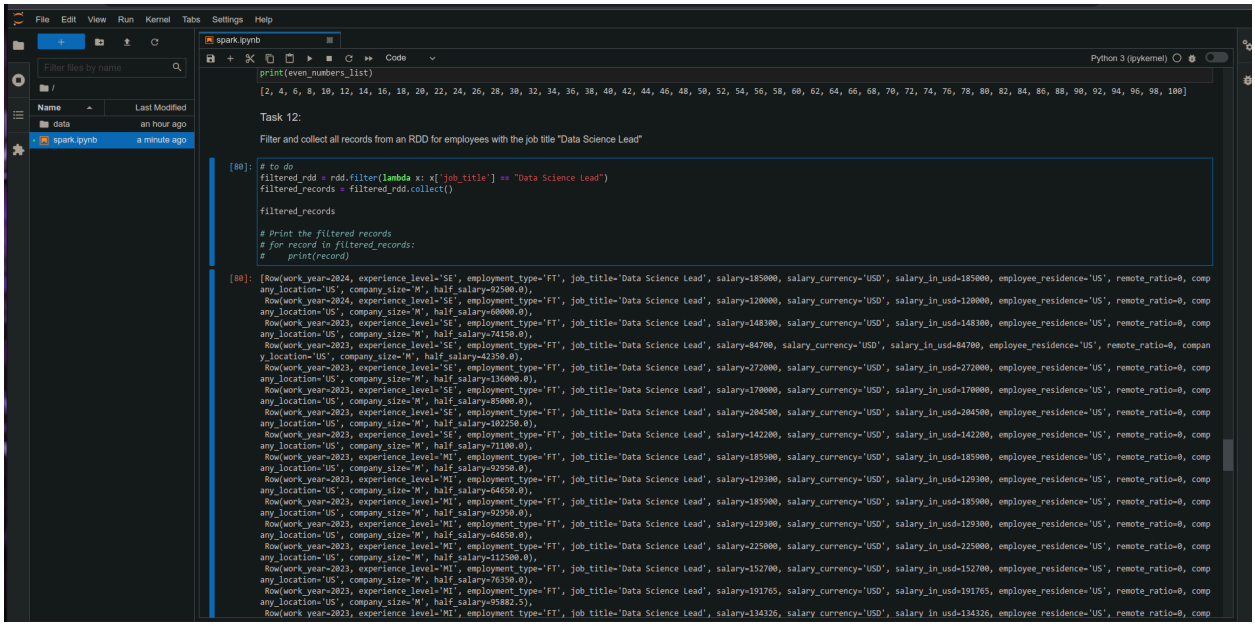
[77]:
```
3262
```

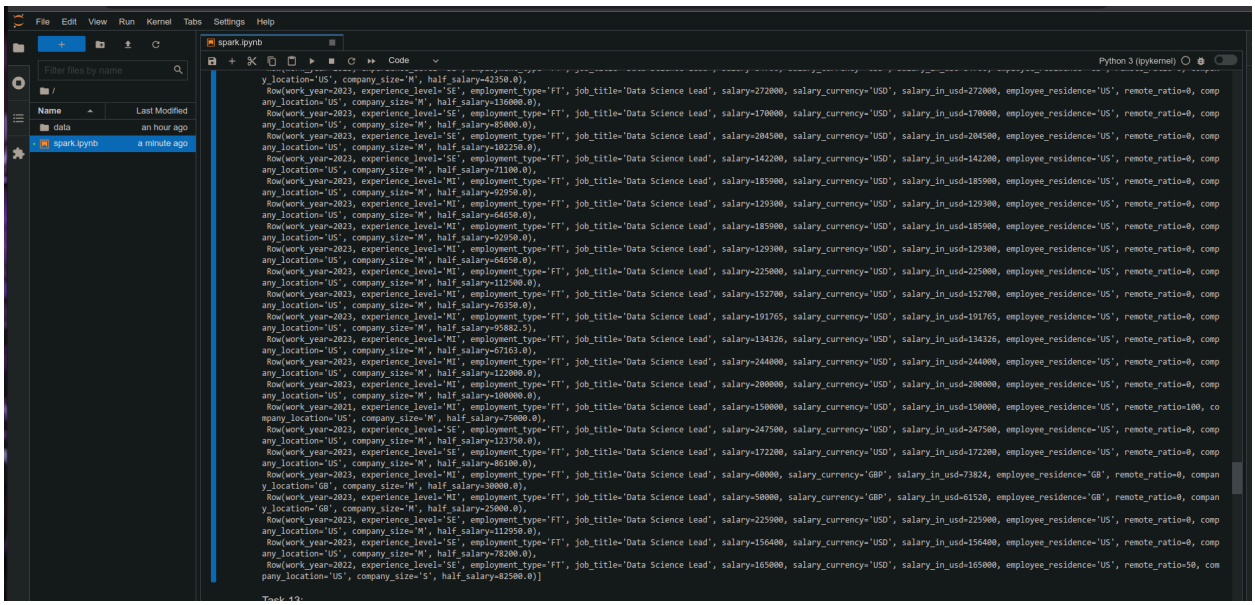
Task 11:

Create an RDD with integers from 1-50. Apply a transformation to multiply every number by 2, resulting in an RDD that contains the first 50 even numbers.

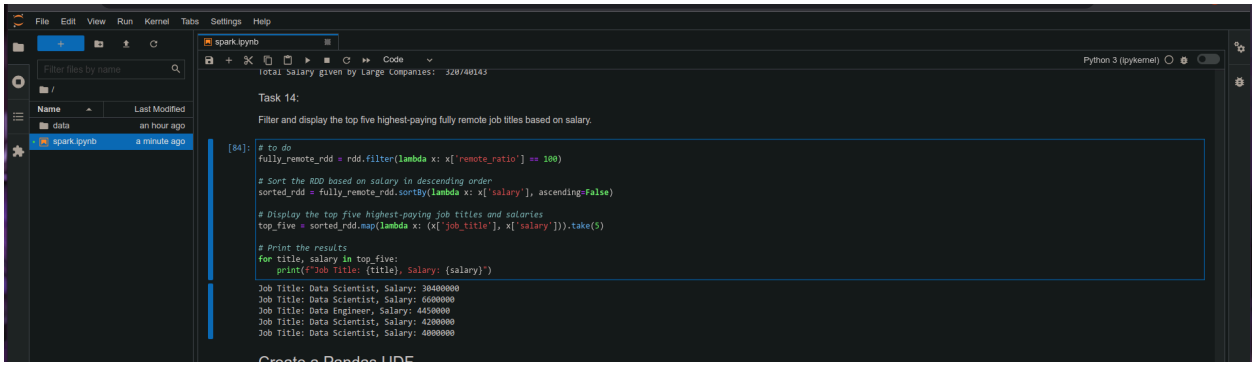
[78]:
```
# to do
numbers = range(1, 51)
rdd_num = spark.sparkContext.parallelize(numbers)
even_numbers_rdd = rdd_num.map(lambda x: x * 2)
even_numbers_list = even_numbers_rdd.collect()
print(even_numbers_list)
```

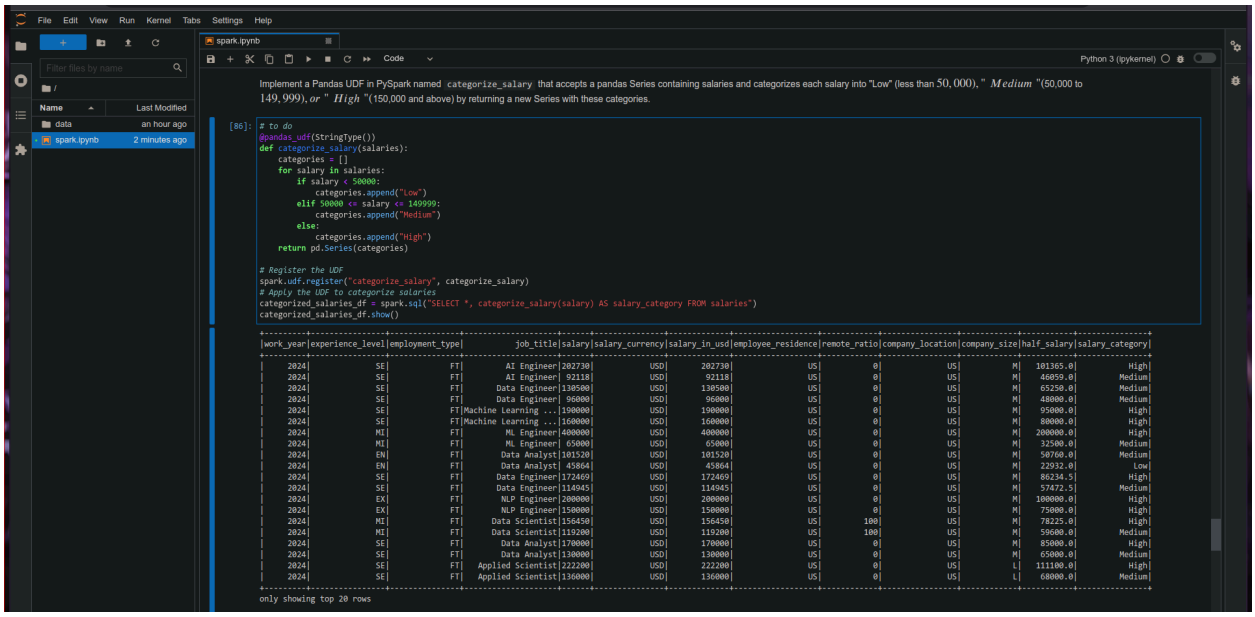
```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94, 96, 98, 100]
```

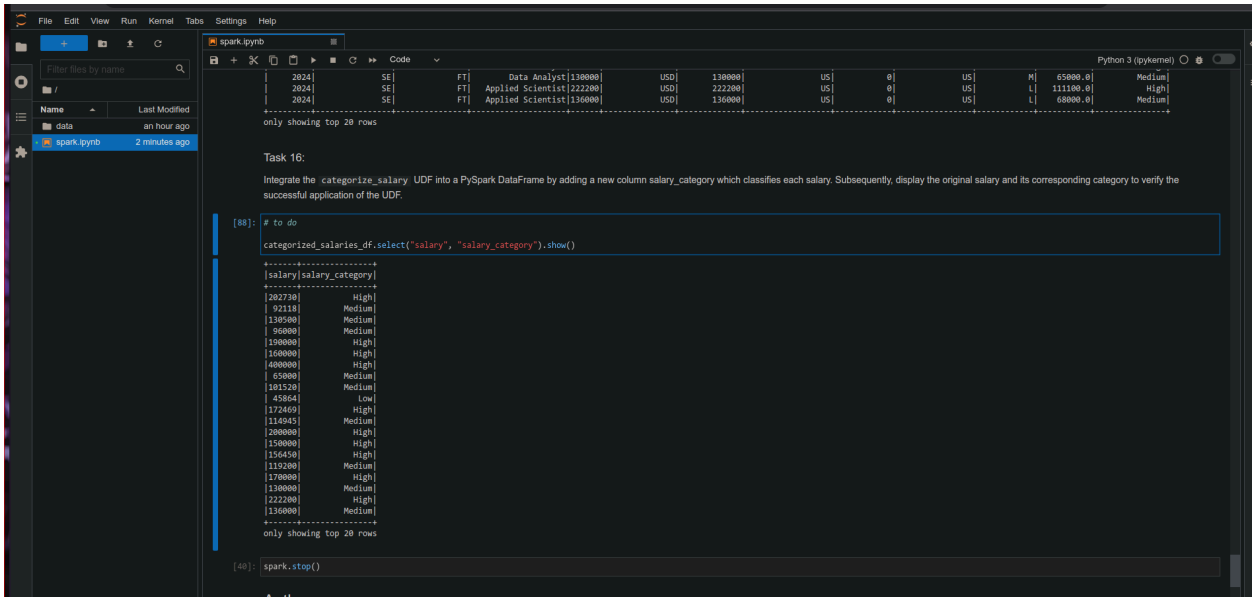

File Edit View Run Kernel Tabs Settings Help
spark.ipynb
Code
Python 3 (ipykernel)
Filter files by name
Name
Last Modified
data
an hour ago
spark.ipynb
a minute ago
print(even_numbers_list)
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94, 96, 98, 100]
Task 12:
Filter and collect all records from an RDD for employees with the job title "Data Science Lead"
[80]:
# to do
filtered_rdd = rdd.filter(lambda x: x['job_title'] == "Data Science Lead")
filtered_records = filtered_rdd.collect()
filtered_records
Print the filtered records
for record in filtered_records:
print(record)
[80]:
[Row(work_year=2024, experience_level='SE', employment_type='FT', job_title='Data Science Lead', salary=185000, salary_currency='USD', salary_in_usd=185000, employee_residence='US', remote_ratio=0, company_location='US', company_size='M', half_salary=92500.0),
Row(work_year=2024, experience_level='SE', employment_type='FT', job_title='Data Science Lead', salary=120000, salary_currency='USD', salary_in_usd=120000, employee_residence='US', remote_ratio=0, company_location='US', company_size='M', half_salary=60000.0),
Row(work_year=2023, experience_level='SE', employment_type='FT', job_title='Data Science Lead', salary=148300, salary_currency='USD', salary_in_usd=148300, employee_residence='US', remote_ratio=0, company_location='US', company_size='M', half_salary=74150.0),
Row(work_year=2023, experience_level='SE', employment_type='FT', job_title='Data Science Lead', salary=84700, salary_currency='USD', salary_in_usd=84700, employee_residence='US', remote_ratio=0, company_location='US', company_size='M', half_salary=42350.0),
Row(work_year=2023, experience_level='SE', employment_type='FT', job_title='Data Science Lead', salary=272000, salary_currency='USD', salary_in_usd=272000, employee_residence='US', remote_ratio=0, company_location='US', company_size='M', half_salary=136000.0),
Row(work_year=2023, experience_level='SE', employment_type='FT', job_title='Data Science Lead', salary=170000, salary_currency='USD', salary_in_usd=170000, employee_residence='US', remote_ratio=0, company_location='US', company_size='M', half_salary=85000.0),
Row(work_year=2023, experience_level='SE', employment_type='FT', job_title='Data Science Lead', salary=204500, salary_currency='USD', salary_in_usd=204500, employee_residence='US', remote_ratio=0, company_location='US', company_size='M', half_salary=102250.0),
Row(work_year=2023, experience_level='SE', employment_type='FT', job_title='Data Science Lead', salary=142200, salary_currency='USD', salary_in_usd=142200, employee_residence='US', remote_ratio=0, company_location='US', company_size='M', half_salary=71100.0),
Row(work_year=2023, experience_level='MI', employment_type='FT', job_title='Data Science Lead', salary=185900, salary_currency='USD', salary_in_usd=185900, employee_residence='US', remote_ratio=0, company_location='US', company_size='M', half_salary=92950.0),
Row(work_year=2023, experience_level='MI', employment_type='FT', job_title='Data Science Lead', salary=129300, salary_currency='USD', salary_in_usd=129300, employee_residence='US', remote_ratio=0, company_location='US', company_size='M', half_salary=64650.0),
Row(work_year=2023, experience_level='MI', employment_type='FT', job_title='Data Science Lead', salary=185900, salary_currency='USD', salary_in_usd=185900, employee_residence='US', remote_ratio=0, company_location='US', company_size='M', half_salary=92950.0),
Row(work_year=2023, experience_level='MI', employment_type='FT', job_title='Data Science Lead', salary=129300, salary_currency='USD', salary_in_usd=129300, employee_residence='US', remote_ratio=0, company_location='US', company_size='M', half_salary=64650.0),
Row(work_year=2023, experience_level='MI', employment_type='FT', job_title='Data Science Lead', salary=225000, salary_currency='USD', salary_in_usd=225000, employee_residence='US', remote_ratio=0, company_location='US', company_size='M', half_salary=112500.0),
Row(work_year=2023, experience_level='MI', employment_type='FT', job_title='Data Science Lead', salary=152700, salary_currency='USD', salary_in_usd=152700, employee_residence='US', remote_ratio=0, company_location='US', company_size='M', half_salary=76350.0),
Row(work_year=2023, experience_level='MI', employment_type='FT', job_title='Data Science Lead', salary=191765, salary_currency='USD', salary_in_usd=191765, employee_residence='US', remote_ratio=0, company_location='US', company_size='M', half_salary=95882.5),
Row(work_year=2023, experience_level='MI', employment_type='FT', job_title='Data Science Lead', salary=134326, salary_currency='USD', salary_in_usd=134326, employee_residence='US', remote_ratio=0, comp


File Edit View Run Kernel Tabs Settings Help
spark.ipynb
Code
Python 3 (ipykernel)
Filter files by name
Name
Last Modified
data
an hour ago
spark.ipynb
a minute ago
y_location='US', company_size='M', half_salary=42350.0),
Row(work_year=2023, experience_level='SE', employment_type='FT', job_title='Data Science Lead', salary=272000, salary_currency='USD', salary_in_usd=272000, employee_residence='US', remote_ratio=0, company_location='US', company_size='M', half_salary=136000.0),
Row(work_year=2023, experience_level='SE', employment_type='FT', job_title='Data Science Lead', salary=170000, salary_currency='USD', salary_in_usd=170000, employee_residence='US', remote_ratio=0, company_location='US', company_size='M', half_salary=85000.0),
Row(work_year=2023, experience_level='SE', employment_type='FT', job_title='Data Science Lead', salary=204500, salary_currency='USD', salary_in_usd=204500, employee_residence='US', remote_ratio=0, company_location='US', company_size='M', half_salary=102250.0),
Row(work_year=2023, experience_level='SE', employment_type='FT', job_title='Data Science Lead', salary=142200, salary_currency='USD', salary_in_usd=142200, employee_residence='US', remote_ratio=0, company_location='US', company_size='M', half_salary=71100.0),
Row(work_year=2023, experience_level='MI', employment_type='FT', job_title='Data Science Lead', salary=185900, salary_currency='USD', salary_in_usd=185900, employee_residence='US', remote_ratio=0, company_location='US', company_size='M', half_salary=92950.0),
Row(work_year=2023, experience_level='MI', employment_type='FT', job_title='Data Science Lead', salary=129300, salary_currency='USD', salary_in_usd=129300, employee_residence='US', remote_ratio=0, company_location='US', company_size='M', half_salary=64650.0),
Row(work_year=2023, experience_level='MI', employment_type='FT', job_title='Data Science Lead', salary=185900, salary_currency='USD', salary_in_usd=185900, employee_residence='US', remote_ratio=0, company_location='US', company_size='M', half_salary=92950.0),
Row(work_year=2023, experience_level='MI', employment_type='FT', job_title='Data Science Lead', salary=129300, salary_currency='USD', salary_in_usd=129300, employee_residence='US', remote_ratio=0, company_location='US', company_size='M', half_salary=64650.0),
Row(work_year=2023, experience_level='MI', employment_type='FT', job_title='Data Science Lead', salary=225000, salary_currency='USD', salary_in_usd=225000, employee_residence='US', remote_ratio=0, company_location='US', company_size='M', half_salary=112500.0),
Row(work_year=2023, experience_level='MI', employment_type='FT', job_title='Data Science Lead', salary=152700, salary_currency='USD', salary_in_usd=152700, employee_residence='US', remote_ratio=0, company_location='US', company_size='M', half_salary=76350.0),
Row(work_year=2023, experience_level='MI', employment_type='FT', job_title='Data Science Lead', salary=191765, salary_currency='USD', salary_in_usd=191765, employee_residence='US', remote_ratio=0, company_location='US', company_size='M', half_salary=95882.5),
Row(work_year=2023, experience_level='MI', employment_type='FT', job_title='Data Science Lead', salary=134326, salary_currency='USD', salary_in_usd=134326, employee_residence='US', remote_ratio=0, company_location='US', company_size='M', half_salary=67163.0),
Row(work_year=2023, experience_level='MI', employment_type='FT', job_title='Data Science Lead', salary=244000, salary_currency='USD', salary_in_usd=244000, employee_residence='US', remote_ratio=0, company_location='US', company_size='M', half_salary=122000.0),
Row(work_year=2023, experience_level='MI', employment_type='FT', job_title='Data Science Lead', salary=200000, salary_currency='USD', salary_in_usd=200000, employee_residence='US', remote_ratio=0, company_location='US', company_size='M', half_salary=100000.0),
Row(work_year=2021, experience_level='MI', employment_type='FT', job_title='Data Science Lead', salary=150000, salary_currency='USD', salary_in_usd=150000, employee_residence='US', remote_ratio=100, company_location='US', company_size='M', half_salary=75000.0),
Row(work_year=2023, experience_level='SE', employment_type='FT', job_title='Data Science Lead', salary=247500, salary_currency='USD', salary_in_usd=247500, employee_residence='US', remote_ratio=0, company_location='US', company_size='M', half_salary=123750.0),
Row(work_year=2023, experience_level='SE', employment_type='FT', job_title='Data Science Lead', salary=172200, salary_currency='USD', salary_in_usd=172200, employee_residence='US', remote_ratio=0, company_location='US', company_size='M', half_salary=86100.0),
Row(work_year=2023, experience_level='MI', employment_type='FT', job_title='Data Science Lead', salary=60000, salary_currency='GBP', salary_in_usd=73824, employee_residence='GB', remote_ratio=0, company_location='GB', company_size='M', half_salary=30000.0),
Row(work_year=2023, experience_level='MI', employment_type='FT', job_title='Data Science Lead', salary=50000, salary_currency='GBP', salary_in_usd=61520, employee_residence='GB', remote_ratio=0, company_location='GB', company_size='M', half_salary=25000.0),
Row(work_year=2023, experience_level='SE', employment_type='FT', job_title='Data Science Lead', salary=225900, salary_currency='USD', salary_in_usd=225900, employee_residence='US', remote_ratio=0, company_location='US', company_size='M', half_salary=112950.0),
Row(work_year=2023, experience_level='SE', employment_type='FT', job_title='Data Science Lead', salary=156400, salary_currency='USD', salary_in_usd=156400, employee_residence='US', remote_ratio=0, company_location='US', company_size='M', half_salary=78200.0),
Row(work_year=2022, experience_level='SE', employment_type='FT', job_title='Data Science Lead', salary=165000, salary_currency='USD', salary_in_usd=165000, employee_residence='US', remote_ratio=50, company_location='US', company_size='S', half_salary=82500.0)]
Task 13:


File Edit View Run Kernel Tabs Settings Help
spark.ipynb
Code
Python 3 (ipykernel)
Filter files by name
Name
Last Modified
data
an hour ago
spark.ipynb
2 minutes ago
Row(work_year=2023, experience_level='MI', employment_type='FT', job_title='Data Science Lead', salary=60000, salary_currency='GBP', salary_in_usd=73824, employee_residence='GB', remote_ratio=0, company_location='GB', company_size='M', half_salary=30000.0),
Row(work_year=2023, experience_level='MI', employment_type='FT', job_title='Data Science Lead', salary=50000, salary_currency='GBP', salary_in_usd=61520, employee_residence='GB', remote_ratio=0, company_location='GB', company_size='M', half_salary=25000.0),
Row(work_year=2023, experience_level='SE', employment_type='FT', job_title='Data Science Lead', salary=225900, salary_currency='USD', salary_in_usd=225900, employee_residence='US', remote_ratio=0, company_location='US', company_size='M', half_salary=112950.0),
Row(work_year=2023, experience_level='SE', employment_type='FT', job_title='Data Science Lead', salary=156400, salary_currency='USD', salary_in_usd=156400, employee_residence='US', remote_ratio=0, company_location='US', company_size='M', half_salary=78200.0),
Row(work_year=2022, experience_level='SE', employment_type='FT', job_title='Data Science Lead', salary=165000, salary_currency='USD', salary_in_usd=165000, employee_residence='US', remote_ratio=50, company_location='US', company_size='S', half_salary=82500.0)]
Task 13:
Calculate the total salary paid by large companies by filtering and aggregating salary data in an RDD using Apache Spark.
[82]:
# to do
filtered_rdd = salaries_df.filter(salaries_df.company_size == 'L').rdd
# Aggregate the salary data to calculate the total salary
total_salary = filtered_rdd.map(lambda x: x.salary).sum()
print("Total Salary given by Large Companies: ", total_salary)
Total Salary given by Large Companies: 320740143
Task 14:


File Edit View Run Kernel Tabs Settings Help
spark.ipynb
Code
Python 3 (ipykernel)
Filter files by name
Name
Last Modified
data
an hour ago
spark.ipynb
a minute ago
Total Salary given by Large Companies: 320/40143
Task 14:
Filter and display the top five highest-paying fully remote job titles based on salary.
[84]:
# to do
fully_remote_rdd = rdd.filter(lambda x: x['remote_ratio'] == 100)
# Sort the RDD based on salary in descending order
sorted_rdd = fully_remote_rdd.sortBy(lambda x: x['salary'], ascending=False)
# Display the top five highest-paying job titles and salaries
top_five = sorted_rdd.map(lambda x: (x['job_title'], x['salary'])).take(5)
# Print the results
for title, salary in top_five:
print(f"Job Title: {title}, Salary: {salary}")
Job Title: Data Scientist, Salary: 30400000
Job Title: Data Scientist, Salary: 6600000
Job Title: Data Engineer, Salary: 4450000
Job Title: Data Scientist, Salary: 4200000
Job Title: Data Scientist, Salary: 4000000
Create a Pandas UDF


File Edit View Run Kernel Tabs Settings Help
spark.ipynb
Code
Python 3 (ipykernel)
Filter files by name
Name
Last Modified
data
an hour ago
spark.ipynb
2 minutes ago
Implement a Pandas UDF in PySpark named categorize_salary that accepts a pandas Series containing salaries and categorizes each salary into "Low" (less than 50,000), "Medium" (50,000 to 149,999), or "High" (150,000 and above) by returning a new Series with these categories.
[86]:
to do
@pandas_udf(StringType())
def categorize_salary(salaries):
categories = []
for salary in salaries:
if salary < 50000:
categories.append("Low")
elif 50000 <= salary <= 149999:
categories.append("Medium")
else:
categories.append("High")
return pd.Series(categories)
# Register the UDF
spark.udf.register("categorize_salary", categorize_salary)
# Apply the UDF to categorize salaries
categorized_salaries_df = spark.sql("SELECT *, categorize_salary(salary) AS salary_category FROM salaries")
categorized_salaries_df.show()
+---------+----------------+---------------+--------------------+------+-------------+-------------+----------------+-------------+----------------+-----------+-----------+---------------+
|work_year|experience_level|employment_type| job_title|salary|salary_currency|salary_in_usd|employee_residence|remote_ratio|company_location|company_size|half_salary|salary_category|
+---------+----------------+---------------+--------------------+------+-------------+-------------+----------------+-------------+----------------+-----------+-----------+---------------+
2024	SE	FT	AI Engineer	202730	USD	202730	US	0	US	M	101365.0	High
2024	SE	FT	AI Engineer	92118	USD	92118	US	0	US	M	46059.0	Medium
2024	SE	FT	Data Engineer	130500	USD	130500	US	0	US	M	65250.0	Medium
2024	SE	FT	Data Engineer	96000	USD	96000	US	0	US	M	48000.0	Medium
2024	SE	FT	Machine Learning ...	190000	USD	190000	US	0	US	M	95000.0	High
2024	SE	FT	Machine Learning ...	160000	USD	160000	US	0	US	M	80000.0	High
2024	MI	FT	ML Engineer	400000	USD	400000	US	0	US	M	200000.0	High
2024	MI	FT	ML Engineer	65000	USD	65000	US	0	US	M	32500.0	Medium
2024	EN	FT	Data Analyst	101520	USD	101520	US	0	US	M	50760.0	Medium
2024	EN	FT	Data Analyst	45864	USD	45864	US	0	US	M	22932.0	Low
2024	SE	FT	Data Engineer	172469	USD	172469	US	0	US	M	86234.5	High
2024	SE	FT	Data Engineer	114945	USD	114945	US	0	US	M	57472.5	Medium
2024	EX	FT	NLP Engineer	200000	USD	200000	US	0	US	M	100000.0	High
2024	EX	FT	NLP Engineer	150000	USD	150000	US	0	US	M	75000.0	High
2024	MI	FT	Data Scientist	156450	USD	156450	US	100	US	M	78225.0	High
2024	MI	FT	Data Scientist	119200	USD	119200	US	100	US	M	59600.0	Medium
2024	SE	FT	Data Analyst	170000	USD	170000	US	0	US	M	85000.0	High
2024	SE	FT	Data Analyst	130000	USD	130000	US	0	US	M	65000.0	Medium
2024	SE	FT	Applied Scientist	222200	USD	222200	US	0	US	L	111100.0	High
2024	SE	FT	Applied Scientist	136000	USD	136000	US	0	US	L	68000.0	Medium
+---------+----------------+---------------+--------------------+------+-------------+-------------+----------------+-------------+----------------+-----------+-----------+---------------+
only showing top 20 rows


File Edit View Run Kernel Tabs Settings Help
spark.ipynb
Code
Python 3 (ipykernel)
Filter files by name
Name
Last Modified
data
an hour ago
spark.ipynb
2 minutes ago
only showing top 20 rows
Task 16:
Integrate the categorize_salary UDF into a PySpark DataFrame by adding a new column salary_category which classifies each salary. Subsequently, display the original salary and its corresponding category to verify the successful application of the UDF.
[88]:
# to do
categorized_salaries_df.select("salary", "salary_category").show()
+------+---------------+
|salary|salary_category|
+------+---------------+
|202730| High|
| 92118| Medium|
|130500| Medium|
| 96000| Medium|
|190000| High|
|160000| High|
|400000| High|
| 65000| Medium|
|101520| Medium|
| 45864| Low|
|172469| High|
|114945| Medium|
|200000| High|
|150000| High|
|156450| High|
|119200| Medium|
|170000| High|
|130000| Medium|
|222200| High|
|136000| Medium|
+------+---------------+
only showing top 20 rows
[40]:
spark.stop()