OnPremise Big Data Engines

Dokumentation der On-Premise Engine “Spark”
Allgemeine Doku: https://spark.apache.org/docs/latest/
Streaming Doku: https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html

Es soll ein Spark-Programm geschrieben werden, welches einfache Analysen auf einem Dataset ausführt. Dabei kann das unten genannte Beispiel (Tankpreise) verwendet werden, oder ein eigenes eingereicht werden.
Der Fokus des Beispiels soll neben der Benutzung des Frameworks auch auf die Art der Datenverarbeitung liegen. Hier soll besonders auf den skalierende Aspekt der Arbeit eingegangen werden.

Liste von Beispiel-Datensets: https://github.com/jdorfman/awesome-json-datasets

Konkretes Beispiel das die Komplexität widerspiegeln soll ist z.B. die Liste der deutschlandweiten Tankpreise[1] seit 2014. Die Liste liegt tagesweise in einem Ordner vor, der nach Jahren und Monate sortiert ist. Die einzelnen Files sind CSV, d.h. die Werte sind via Beistrich „,“ getrennt mit einer Kopfzeile. Dadurch das ein Tagesfile ca 60000 Einträge hat, verarbeiten Sie bitte nur einen Tag im Zuge dieser Übung.

Aufgabenstellung und/oder Fragestellungen

Angelehnt an das Tank-Beispiel sollen folgende Fragestellungen beantwortet werden. Falls ein eigenes Datenset verwendet wird, bitte die Fragen in den Code dokumentieren. In diesem Fall, sollen ähnliche Fragestellungen beantwortet werden

- An welcher Tankstelle in Deutschland schwankt der Preis am stärksten ?
- Was ist der Durchschnittspreis der Tankstellen pro Tag ?
- Gibt es Tageszeiten die vermieden werden sollen ?

Bei alternativen oder selbst gewählten Beispiele

- Bitte achten sie auf die Lizenz der Daten und darauf, das sie bei der Abgabe reproduzierbar sein müssen. Ein Zugriff über einen personalisierten Login ist daher **keine** Option.
- Es müssen leichte Aggregationen gemacht werden. Ähnlich wie oben reicht eine Einfach Summenbildung, ein Mittelwert oder ähnliches, welches nativ in der jeweiligen Programmiersprache umgesetzt wird.

Tipps

[1] Die Liste der deutschlandweiten Tankpreise kann unter https://dev.azure.com/tankerkoenig/_git/tankerkoenig-data abgerufen werden

- Es soll ein Java/Scala oder Python Programm abgegeben werden, welches den den “good Practise” entspricht. Um das sicherzustellen empfiehlt es sich einen Linter zu nehmen. Als Inspiration kann folgender Post verwendet werden: https://code.tutsplus.com/de/tutorials/top-15-best-practices-for-writing-super-readable-code--net-8118
- Planen sie eine Test- und Verifikationsphase ein, d.h. probieren sie ihr Programm vor der Abgabe entsprechend aus

Bewertungskriterien

Folgende Punkte sind Bewertungskriterien:

1. Muss - ein Ressourcen File muss vorhanden sein (setup.py, pom.xml, build.sbt…) wo alle Abhängigkeiten aufgelistet werden.
2. Muss - Es muss eine Hauptfunktion (z.B. Java public static void main, Python if __name__ == "__main__”) geben
3. Muss - Es muss mindestens ein Sub-Modul/Paket vorhanden sein, welches nicht im selben Directory liegt wie die Hauptfunktion
4. Muss - Es muss mindestens eine inhaltliche Frage (Beispiele siehe oben - Fragestellungen angelehnt an das Reddit Beispiel) bearbeitet werden
5. Muss - Der Code muss bei der Abgabe lauffähig sein.
6. Soll - Der Code soll dementsprechend dokumentiert sein, insb. geschriebene Funktionen müssen/sollen eine Dokumentation enthalten
7. Muss - Der Code muss leserlich sein, d.h. entsprechende Einrückungen und Umbrüche haben
8. Soll - Methoden, Variablen, Klassen sollen einen ansprechenden Namen erhalten.
9. Muss - Der Code muss auf Skalierung ausgelegt sein. Falls sie sich bei verwendeten Bibliotheken unsicher sind, lassen sie sie lieber weg
10. Muss – Die ermittelten Statistiken müssen wieder abgespeichert werden

Wenn vorhanden:

11. es sollen keine sicherheitssensitiven Informationen, wie Benutzernamen oder Passwörter im Source Code stehen

Für dieses Beispiel gibt es 20 Punkte.