**Отчет по лабораторной работе №3**

В ходе лабораторной работы были проведены замеры времени обработки 4 запросов для трех библиотек: Pandas, SQLite и DuckDB. Сразу приведу таблицу с результатами работы программы и сравнительным графиком, а затем постараюсь порассуждать и проанализировать полученные данные. Тестирование удалось провести только на укороченной ЬД, так как огромное количество данных не потянул мой компьютер.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **название библиотеки** | **время выполнения запроса** | | | |
| №1 | №2 | №3 | №4 |
| SQLITE | 0.4056095000000123 | 0.75205059998788 | 1.3341771999999812 | 2.6344617000000004 |
| DuckDB | 0.0027349000000099 | 0.00720019999997 | 0.0141282000000103 | 0.0242024000000039 |
| Pandas | 0.4693007000000005 | 1.08882110000004 | 1.8120078000000001 | 2.5602346999999998 |

Сразу взглянув на полученные данные, можно прийти к выводу, что библиотекой с самыми быстрыми методами является библиотека DuckDB. Ее значений времени работы даже не видно на гистограмме – вот на сколько они малы. Это связано с тем, что DuckDB использует векторный метод выполнения запросов и ориентируется на столбцы, когда те же ребята, SQLite и Pandas, обрабатывают строки последовательно.

Первый запрос выглядел так: SELECT VendorID, COUNT(\*) AS counts FROM data GROUP BY VendorID. Самый обычный селект вендеров, их группировка и подсчет. Здесь SQLite работает немного быстрее, чем Pandas.

Во втором запросе я использовала агрегатную функцию подсчета среднего значения: SELECT passenger\_count, AVG(trip\_distance) AS avg\_distance FROM data GROUP BY passenger\_count. Группировка по пассажирам и средние дистанции. Здесь уже SQLite сработал немного ощутимо быстрее, чем Pandas.

В третьем запросе я выбрала все строки из БД, в которых дистанция превышает значение 3: SELECT \* FROM data WHERE trip distance > 3. Здесь DuckDB пришлось немного тяжелее, но эта библиотека все так же демонстрирует наилучший результат. SQLite сработал быстрее, чем Pandas.

Четвертый запрос: SELECT payment\_type, AVG(tip\_amount) AS avg\_tip FROM data GROUP BY payment\_type. Так же агрегатная функция среднего значение tip и группировка по payment\_type. Pandas показала результат лучше, чем SQLite, правда, не намного, а DuckDB все так же осталась бессменным лидером.

**Вывод:**

Во время работы с библиотеками у меня сложился какой-то свой небольшой опыт по их использованию. Мне показалось, что библиотека Pandas наиболее проста в взаимодействии – у меня сразу получилось ее подключить и использовать. Как итог: библиотека удобная, простая, но медленная. С SQLlite я намучалась, если честно. Ну а бесспорным лидером для меня стала библиотека DuckDB: очень-очень быстро, достаточно просто и удобно. Хотела бы в будущем поработать с ней побольше!

**Я все, всего вам самого наилучшего!**

**Будем по вам скучать всем потоком.**