Universidad de Costa Rica

Escuela de Ciencias de la Computación e Informática

CI-0141

Bases de Datos Avanzadas

Proyecto Final

Profesor:
Dr. Luis Gustavo Esquivel Quirós

Estudiantes:

Marco Piedra Venegas - A64397

Alejandro Jiménez Corea - B84032

Daniel Artavia - B70771

Gabriel Molina Bulgarelli - C14826

I Semestre
2024

Introducción

En este proyecto se realizan tareas básicas de analítica de datos mediante DuckDB (https://duckdb.org/), un motor de base de datos con características NoSQL, sobre archivos en formato Parquet (https://parquet.apache.org/). Asimismo, se compara rendimiento con respecto a PostgreSQL (https://www.postgresql.org/), un motor de base de datos objeto-relacional.

El objetivo general del proyecto consiste en evidenciar la flexibilidad de un acercamiento NoSQL para procesamiento de datos semiestructurados, cuyo volumen supere los recursos de memoria.

Descripción del problema

DuckDB se describe como un motor de base de datos para analítica, que puede funcionar dentro de procesos (embedded). De este modo, se puede integrar dentro de otros programas (DuckDB Foundation, 2024).

Parquet es un formato columnar especializado para almacenamiento y acceso rápido de datos. Utiliza compresión de alto rendimiento y esquemas de codificación para grandes volúmenes (Apache Parquet, 2024).

PostgreSQL es un motor de base de datos objeto-relacional de código abierto, que se apega al estándar SQL y permite extensiones, tales como datos de información geográfica. Se originó en 1986 en la Universidad de California en Berkeley (The PostgreSQL Global Development Group, 2024).

Descripción de datos

El conjunto de datos por analizar para el proyecto consiste en los viajes en plataformas de transporte, recopilados mensualmente por la Comisión de Taxis y Limosinas de la Ciudad de Nueva York, y disponibles de forma pública (City of New York, 2024). Maneja 62 archivos en formato Parquet, que suman 26 GB. La media de tamaño es 415.7 MB, con una desviación estándar de 97.4 MB. El rango de tiempo es desde febrero de 2019 hasta marzo de 2024 (Anexos 1 y 2).

Asimismo, se consideran solo los viajes en la categoría llamada “High Volume For-Hire Vehicle”, que incluyen a las plataformas de transporte que generan más de 10000 viajes diarios en la ciudad de Nueva York. Se incluyen 4 proveedores: Uber, Lyft, Juno, y Via. Los datos de Juno terminan en noviembre de 2019, debido a que en ese mes dejó de operar (Bellon, 2019).

Cada archivo Parquet incluye campos como el código de proveedor, fecha y hora de inicio y fin de viaje, longitud del viaje en millas, tiempo del viaje en segundos, tarifa base para el pasajero, monto recibido por el conductor, propina, e impuestos.

Modelado de Datos

El modelo de datos por utilizar es columnar, representado por el procesamiento de archivos en formato Parquet. Se clasifica como un modelo NoSQL.

Se realiza procesamiento de una tabla con 24 columnas, listadas a continuación:

```
D describe select * from '*.parquet';
```

column_name varchar	column_type varchar	null varchar	key varchar	default varchar	extra varchar
hvfhs_license_num	VARCHAR	YES			
dispatching_base_num	VARCHAR	YES			
originating_base_num	VARCHAR	YES			
request_datetime	TIMESTAMP	YES			
on_scene_datetime	TIMESTAMP	YES			
pickup_datetime	TIMESTAMP	YES			
dropoff_datetime	TIMESTAMP	YES			
PULocationID	BIGINT	YES			
DOLocationID	BIGINT	YES			
trip_miles	DOUBLE	YES			
trip_time	BIGINT	YES			
base_passenger_fare	DOUBLE	YES			
tolls	DOUBLE	YES			
bcf	DOUBLE	YES			
sales_tax	DOUBLE	YES			
congestion_surcharge	DOUBLE	YES			
airport_fee	INTEGER	YES			
tips	DOUBLE	YES			
driver_pay	DOUBLE	YES			
shared_request_flag	VARCHAR	YES			
shared_match_flag	VARCHAR	YES			
access_a_ride_flag	VARCHAR	YES			
wav_request_flag	VARCHAR	YES			
wav_match_flag	INTEGER	YES			
24 rows					6 columns

En el análisis de datos solo se utilizan las siguientes columnas:

- trip_miles: distancia de un viaje en millas
- trip_time: tiempo de un viaje en segundos
- base_passenger_fare: tarifa base de pago, en un viaje, para quien solicita el transporte
- driver_pay: tarifa base de ingresos, en un viaje, para quien ofrece el transporte
- PULocationID: identificador de la zona de la ciudad de Nueva York donde inicia un viaje

Escenarios de análisis

En este proyecto, se realiza la comparación de rendimiento de DuckDB y PostgreSQL en cuanto a las funciones de agregación estadística. En particular, `regr_slope` y `regr_intercept` para obtener la pendiente de la línea de regresión y su intersección con el eje *y*, respectivamente.

Se desea realizar un análisis comparativo entre dos diferentes modelos de bases de datos utilizando una consulta SQL que aplica funciones de regresión lineal para analizar la relación entre tarifas de pasajeros, pagos a conductores y las características del viaje, como la distancia y el tiempo. Este análisis se basa en datos almacenados en archivos Parquet.

Se aplican estas operaciones para obtener regresión lineal de la ganancia base de conductor (`driver_pay`) y cobro base al pasajero (`base_passenger_fare`), con respecto a longitud de viaje (`trip_miles`) y tiempo de viaje (`trip_time`).

Se seleccionaron estas operaciones para comparación, ya que su ejecución no es trivial en un volumen de datos considerable. Asimismo, se encuentran disponibles en ambos motores de bases de datos y tienen la misma semántica. De este modo, se comparan ejecuciones equivalentes entre ambos motores.

En el experimento, se toman los tiempos de ejecución, al habilitar las opciones correspondientes con `.timer on` en DuckDB y `\timing` en PostgreSQL. Se realizan 3 corridas por cada combinación de factores: base de datos (2 niveles: DuckDB y PostgreSQL) y consulta (A y B, listadas en anexos 4 y 5).

El experimento se realiza en una máquina Dell XPS 13 (modelo 9315), con 12 CPUs Intel Core i5-1230U y 8 GB de RAM DDR5, en Debian GNU/Linux 12.6 (Bookworm). Cada ejecución se realiza “en frío”, i.e., se detiene y se vuelve a iniciar el proceso. En el caso de DuckDB, se realiza salida de la línea de comandos (el ejecutable único inicia el motor de base de datos y la interfaz de usuario). En PostgreSQL, se detiene y se inicia el servicio.

Resultados

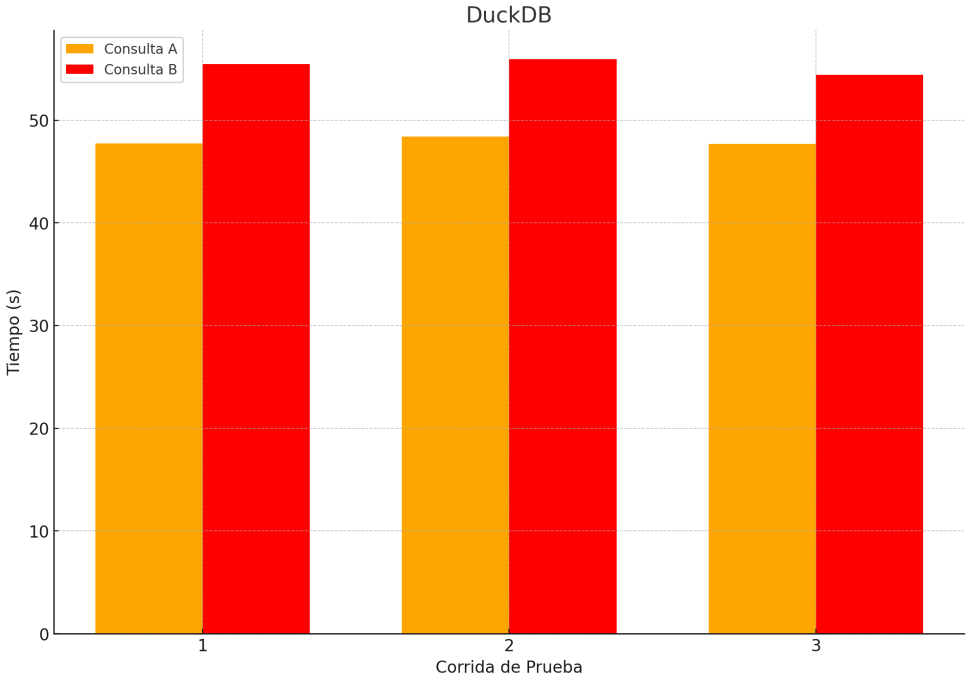
En cuanto a tiempos de ejecución de las consultas (anexos 4 y 5) en los motores de bases de datos se observa que, en promedio, en DuckDB la consulta A tarda 47.937 segundos y la consulta B tarda 55.271 segundos. Mientras tanto, en promedio, en PostgreSQL la consulta A tarda 81.109 segundos y la consulta B tarda 172.682 segundos (Anexo 8 y gráficos).

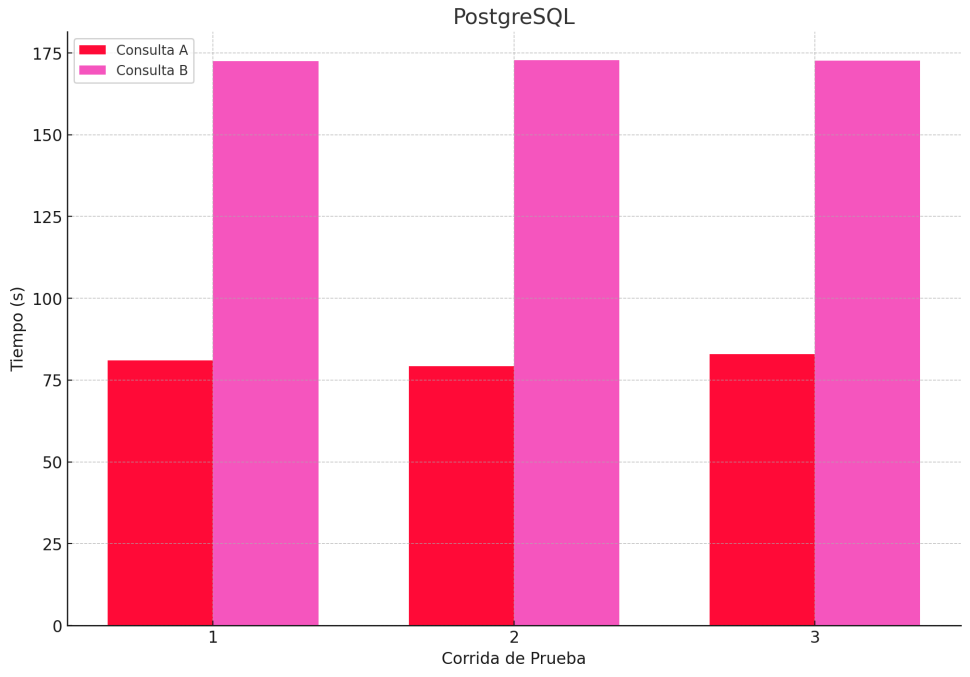
De esta forma, DuckDB presenta una mejora de rendimiento de 1.153 en la consulta A, así como una mejora rendimiento de 2.129 en la consulta B. Cabe destacar que DuckDB accede a los archivos de forma directa, mientras que PostgreSQL requirió cargar datos a una tabla desde los archivos.

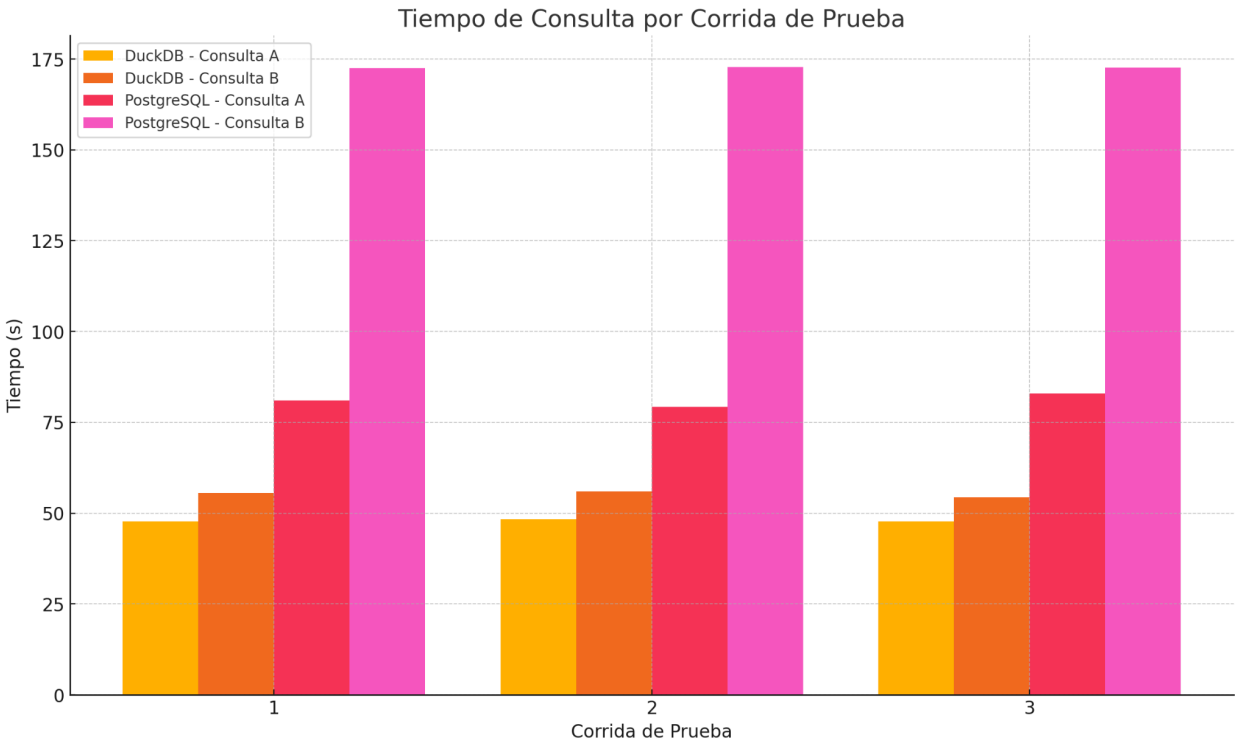
Una interpretación de la mejora de rendimiento con DuckDB, con respecto a PostgreSQL, es el procesamiento nativo de archivos en formato Parquet. El formato columnar y la compresión de datos de Parquet permite reducir tiempos de acceso a datos en disco.

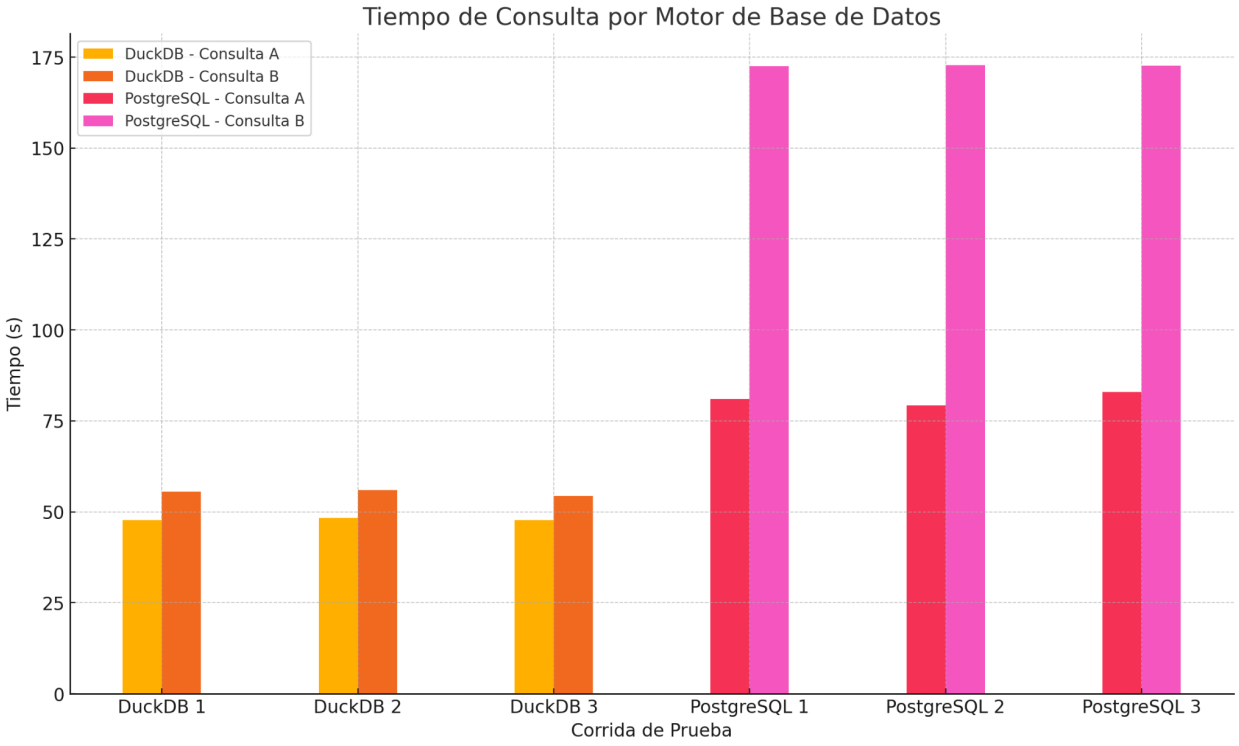
Asimismo, DuckDB utiliza una implementación de vectorización de consultas del motor de consultas X100 en MonetDB (Boncz et al., 2005). Debido a la homogeneidad de datos dentro de una columna (i.e., un mismo tipo de dato, como *double*) en el modelo columnar de bases de datos, es posible aplicar optimizaciones (e.g., operaciones SIMD del procesador) que serían limitadas en el modelo tabular.

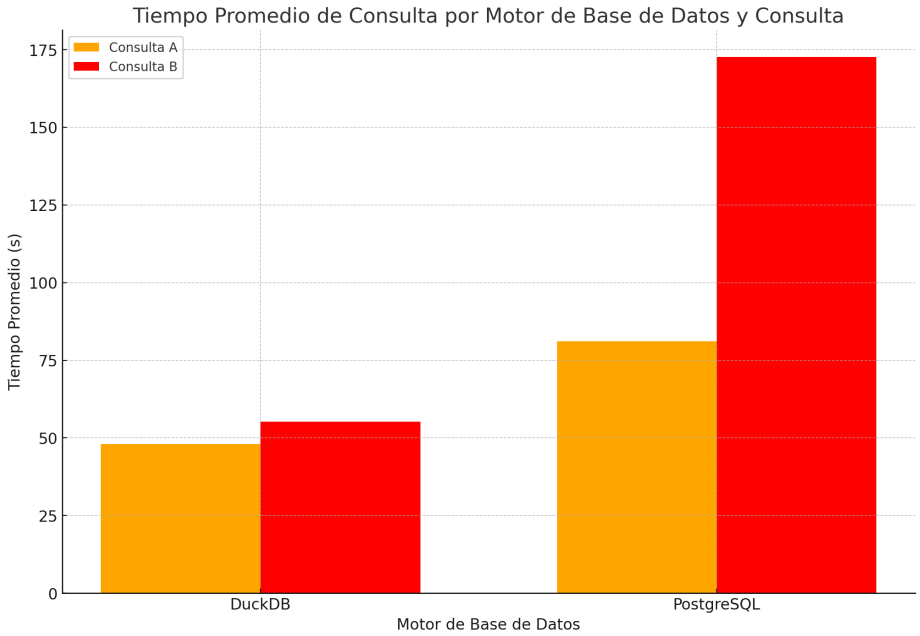
Por otro lado, es importante señalar que DuckDB no cuenta con facilidades avanzadas de recuperación de base de datos (e.g., una implementación del algoritmo ARIES). El caso de uso de DuckDB es análisis de datos interactivo y uso empotrado (embedded) dentro de otra aplicación, similar al motor de base de datos SQLite (SQLite, 2024).


DuckDB
Consulta A
Consulta B
Tiempo (s)
0
10
20
30
40
50
1
2
3
Corrida de Prueba


PostgreSQL
Consulta A
Consulta B
Tiempo (s)
0
25
50
75
100
125
150
175
1
2
3
Corrida de Prueba


Tiempo de Consulta por Corrida de Prueba
DuckDB - Consulta A
DuckDB - Consulta B
PostgreSQL - Consulta A
PostgreSQL - Consulta B
Tiempo (s)
0
25
50
75
100
125
150
175
1
2
3
Corrida de Prueba


Tiempo de Consulta por Motor de Base de Datos
DuckDB - Consulta A
DuckDB - Consulta B
PostgreSQL - Consulta A
PostgreSQL - Consulta B
Tiempo (s)
0
25
50
75
100
125
150
175
DuckDB 1
DuckDB 2
DuckDB 3
PostgreSQL 1
PostgreSQL 2
PostgreSQL 3
Corrida de Prueba


Tiempo Promedio de Consulta por Motor de Base de Datos y Consulta
Consulta A
Consulta B
Tiempo Promedio (s)
0
25
50
75
100
125
150
175
DuckDB
PostgreSQL
Motor de Base de Datos

Referencias

Apache Parquet (2024). *Overview*. https://parquet.apache.org/docs/overview/

Boncz, P., Zukowski, M., & Nes, N. (2005). MonetDB/X100: Hyper-Pipelining Query Execution. In Proceedings of International conference on verly large data bases (VLDB) 2005. Very Large Data Base Endowment.

Bellon, T. (2019). Gett's Juno ends NYC ride-hailing services, citing regulation. Reuters. https://www.reuters.com/article/us-gett-juno-idUSKBN1XS2JZ/

City of New York (2024). *TLC Trip Record Data*. https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page

DuckDB Foundation (2024). *Why DuckDB*. https://duckdb.org/why_duckdb

The PostgreSQL Global Development Group (2024). *About PostgreSQL*. https://www.postgresql.org/about/

EnterpriseDB (2024). Download PostgreSQL. https://www.enterprisedb.com/downloads/postgres-postgresql-downloads

SQLite (2024). About SQLite. https://www.sqlite.org/about.html

Anexos

Anexo 1. Listado de archivos del conjunto de datos, junto con tamaño en MB:

```
490M fhvhv_tripdata_2019-02.parquet
583M fhvhv_tripdata_2019-03.parquet
534M fhvhv_tripdata_2019-04.parquet
545M fhvhv_tripdata_2019-05.parquet
512M fhvhv_tripdata_2019-06.parquet
493M fhvhv_tripdata_2019-07.parquet
479M fhvhv_tripdata_2019-08.parquet
493M fhvhv_tripdata_2019-09.parquet
524M fhvhv_tripdata_2019-10.parquet
536M fhvhv_tripdata_2019-11.parquet
550M fhvhv_tripdata_2019-12.parquet
507M fhvhv_tripdata_2020-01.parquet
533M fhvhv_tripdata_2020-02.parquet
331M fhvhv_tripdata_2020-03.parquet
110M fhvhv_tripdata_2020-04.parquet
154M fhvhv_tripdata_2020-05.parquet
189M fhvhv_tripdata_2020-06.parquet
249M fhvhv_tripdata_2020-07.parquet
277M fhvhv_tripdata_2020-08.parquet
301M fhvhv_tripdata_2020-09.parquet
330M fhvhv_tripdata_2020-10.parquet
288M fhvhv_tripdata_2020-11.parquet
290M fhvhv_tripdata_2020-12.parquet
295M fhvhv_tripdata_2021-01.parquet
289M fhvhv_tripdata_2021-02.parquet
352M fhvhv_tripdata_2021-03.parquet
352M fhvhv_tripdata_2021-04.parquet
370M fhvhv_tripdata_2021-05.parquet
376M fhvhv_tripdata_2021-06.parquet
378M fhvhv_tripdata_2021-07.parquet
365M fhvhv_tripdata_2021-08.parquet
376M fhvhv_tripdata_2021-09.parquet
411M fhvhv_tripdata_2021-10.parquet
393M fhvhv_tripdata_2021-11.parquet
392M fhvhv_tripdata_2021-12.parquet
358M fhvhv_tripdata_2022-01.parquet
389M fhvhv_tripdata_2022-02.parquet
450M fhvhv_tripdata_2022-03.parquet
435M fhvhv_tripdata_2022-04.parquet
447M fhvhv_tripdata_2022-05.parquet
437M fhvhv_tripdata_2022-06.parquet
424M fhvhv_tripdata_2022-07.parquet
417M fhvhv_tripdata_2022-08.parquet
437M fhvhv_tripdata_2022-09.parquet
473M fhvhv_tripdata_2022-10.parquet
443M fhvhv_tripdata_2022-11.parquet
481M fhvhv_tripdata_2022-12.parquet
452M fhvhv_tripdata_2023-01.parquet
438M fhvhv_tripdata_2023-02.parquet
499M fhvhv_tripdata_2023-03.parquet
470M fhvhv_tripdata_2023-04.parquet
490M fhvhv_tripdata_2023-05.parquet
476M fhvhv_tripdata_2023-06.parquet
470M fhvhv_tripdata_2023-07.parquet
430M fhvhv_tripdata_2023-08.parquet
456M fhvhv_tripdata_2023-09.parquet
464M fhvhv_tripdata_2023-10.parquet
444M fhvhv_tripdata_2023-11.parquet
468M fhvhv_tripdata_2023-12.parquet
451M fhvhv_tripdata_2024-01.parquet
442M fhvhv_tripdata_2024-02.parquet
485M fhvhv_tripdata_2024-03.parquet
```

Anexo 2. Listado de enlaces a los archivos del conjunto de datos

https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2019-02.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2019-03.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2019-04.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2019-05.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2019-06.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2019-07.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2019-08.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2019-09.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2019-10.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2019-11.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2019-12.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2020-01.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2020-02.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2020-03.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2020-04.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2020-05.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2020-06.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2020-07.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2020-08.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2020-09.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2020-10.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2020-11.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2020-12.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2021-01.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2021-02.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2021-03.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2021-04.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2021-05.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2021-06.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2021-07.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2021-08.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2021-09.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2021-10.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2021-11.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2021-12.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2022-01.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2022-02.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2022-03.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2022-04.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2022-05.parquet

https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2022-06.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2022-07.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2022-08.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2022-09.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2022-10.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2022-11.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2022-12.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2023-01.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2023-02.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2023-03.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2023-04.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2023-05.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2023-06.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2023-07.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2023-08.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2023-09.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2023-10.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2023-11.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2023-12.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2024-01.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2024-02.parquet
https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2024-03.parquet

Anexo 3. Comandos de DuckDB para análisis exploratorio de datos

```
.timer on

select count(*) from '*.parquet';

select * from 'taxi_zone_lookup.csv';

select Borough from 'taxi_zone_lookup.csv';

describe select * from '*.parquet';

select avg(base_passenger_fare), avg(driver_pay), avg(trip_miles),
avg(trip_time) from '*.parquet';

select Borough, avg(base_passenger_fare), avg(driver_pay), avg(trip_miles),
avg(trip_time) from read_parquet('*.parquet') join 'taxi_zone_lookup.csv' on
LocationID = PULocationID group by Borough;

select Borough, avg(base_passenger_fare), avg(driver_pay), avg(trip_miles),
avg(trip_time) from read_parquet('*.parquet') join 'taxi_zone_lookup.csv' on
LocationID = PULocationID group by Borough order by avg(base_passenger_fare);

select regr_slope(base_passenger_fare, trip_time),
regr_intercept(base_passenger_fare, trip_time) from '*.parquet';

select corr(base_passenger_fare, trip_time), corr(base_passenger_fare,
trip_miles) from '*.parquet';

select corr(base_passenger_fare, base_passenger_fare) from '*.parquet';

select regr_r2(base_passenger_fare, trip_time), regr_r2(base_passenger_fare,
trip_miles) from '*.parquet';

select regr_r2(driver_pay, trip_time), regr_r2(driver_pay, trip_miles) from
'*.parquet';

select regr_r2(driver_pay, tolls), regr_r2(base_passenger_fare, tolls) from
'*.parquet';

select regr_r2(driver_pay, tips), regr_r2(base_passenger_fare, tips) from
'*.parquet';

select regr_r2(driver_pay, congestion_surcharge), regr_r2(base_passenger_fare,
congestion_surcharge) from '*.parquet';

explain select count(*) from '*.parquet';

explain analyze select count(*) from '*.parquet';
```

Anexo 4. Consultas en DuckDB

Consulta A

```
SELECT
      regr_slope(base_passenger_fare, trip_miles) AS slope_passenger_miles,
      regr_intercept(base_passenger_fare, trip_miles) AS
intercept_passenger_miles,
      regr_slope(base_passenger_fare, trip_time) AS slope_passenger_time,
      regr_intercept(base_passenger_fare, trip_time) AS
intercept_passenger_time,
      regr_slope(driver_pay, trip_miles) AS slope_driver_miles,
      regr_intercept(driver_pay, trip_miles) AS intercept_driver_miles,
      regr_slope(driver_pay, trip_time) AS slope_driver_time,
      regr_intercept(driver_pay, trip_time) AS intercept_driver_time
FROM read_parquet('*.parquet');
```

Consulta B

```
SELECT Borough AS nyc_borough,
      AVG(base_passenger_fare) AS avg_passenger_fare,
      AVG(driver_pay) AS avg_driver,
      AVG(trip_miles) AS avg_miles,
      AVG(trip_time) AS avg_time
FROM read_parquet('*.parquet')
JOIN 'taxi_zone_lookup.csv' ON LocationID = PULocationID
GROUP BY Borough
ORDER BY avg_passenger;
```

Anexo 5. Consultas en *PostgreSQL*

Preparación e inserción de datos desde DuckDB hacia PostgreSQL

```
attach 'dbname=postgres user=postgres host=127.0.0.1' as postgres_db (type
postgres);

create table postgres_db.nyc_tlc (

      pulocationid int,

      dolocationid int,

      trip_miles double,

      trip_time double,

      base_passenger_fare double,

      driver_pay double

);

insert into postgres_db.nyc_tlc

select PULocationID, DOLocationID, trip_miles, trip_time, base_passenger_fare,
driver_pay from '*.parquet';
```

Consulta A

```
SELECT

      regr_slope(base_passenger_fare, trip_miles) AS slope_passenger_miles,

      regr_intercept(base_passenger_fare, trip_miles) AS
intercept_passenger_miles,

      regr_slope(base_passenger_fare, trip_time) AS slope_passenger_time,

      regr_intercept(base_passenger_fare, trip_time) AS
intercept_passenger_time,

      regr_slope(driver_pay, trip_miles) AS slope_driver_miles,

      regr_intercept(driver_pay, trip_miles) AS intercept_driver_miles,

      regr_slope(driver_pay, trip_time) AS slope_driver_time,

      regr_intercept(driver_pay, trip_time) AS intercept_driver_time

FROM nyc_tlc;
```

Consulta B

```
SELECT Borough AS nyc_borough,
      AVG(base_passenger_fare) AS avg_passenger_fare,
      AVG(driver_pay) AS avg_driver,
      AVG(trip_miles) AS avg_miles,
      AVG(trip_time) AS avg_time
FROM nyc_tlc
JOIN zones ON locationid = PULocationID
GROUP BY Borough
ORDER BY avg_passenger_fare;
```

Anexo 6. Tiempos de ejecución de consultas

Corrida de prueba	Motor de base de datos	Consulta	Tiempo (s)	Promedio (s)
1	DuckDB	A	47.717	47.937
2	DuckDB	A	48.397	
3	DuckDB	A	47.700	
1	DuckDB	B	55.478	55.271
2	DuckDB	B	55.947	
3	DuckDB	B	54.390	
1	PostgreSQL	A	81.059	81.109
2	PostgreSQL	A	79.334	
3	PostgreSQL	A	82.933	
1	PostgreSQL	B	172.537	172.682
2	PostgreSQL	B	172.816	
3	PostgreSQL	B	172.693	

Anexo 7. Resultados de consultas en DuckDB

Consulta A

slope_passenger_mi… double	intercept_passenge… double	slope_passenger_time double	intercept_passenge… double	…	intercept_driver_m… double	slope_driver_time double	intercept_driver_t… double
2.8721327258002245	7.385506926815001	0.01842244871883288	0.45863906387253905	…	5.674310009919955	0.015749820695916444	-0.7331341673942156

1 rows 8 columns (7 shown)

Consulta B

nyc_borough varchar	avg_passenger double	avg_driver double	avg_miles double	avg_time double
Unknown	16.584720750101873	13.760941704035835	4.47813697513249	915.4113330615572
Bronx	16.830887688827886	14.387078783307208	4.302425895592714	961.8999149975191
Staten Island	18.868895272556667	15.783697065528049	5.300918792979251	945.9410640556766
Brooklyn	19.108697121647577	15.512255502742297	4.206310959430512	1096.0952368070498
N/A	22.915405733593897	18.379297819544934	6.777490171233477	1142.2823328581737
Manhattan	23.330579257384432	18.117832457088333	4.788125558932168	1184.8889437267476
Queens	23.648157104599235	19.3558556446227	6.32917457250853	1216.5134582592054
EWR	48.29858326474578	11.099713580246918	19.57026282578876	2811.0533333333333

Anexo 8. Resultados de consultas en PostgreSQL

Consulta A

```
 slope_passenger_miles | intercept_passenger_miles | slope_passenger_time | intercept_passenger_time | slope_driver_miles | intercept_driver_miles |  slope_driver_time  | int
ercept_driver_time
-----------------------+---------------------------+----------------------+--------------------------+--------------------+------------------------+---------------------+----
-------------------
    2.8721327258146885 |         7.385506923002682 | 0.018422448719017354 |       0.4586390599512873 | 2.3558808670587212 |      5.674310010363294 | 0.01574982069595115 |   -
0.7331341669795817
(1 row)
```

Consulta B

```
  nyc_borough  | avg_passenger_fare |     avg_driver     |     avg_miles      |      avg_time
---------------+--------------------+--------------------+--------------------+--------------------
 Unknown       |   16.5847207501019 | 13.760941704035835 |  4.478136975132487 |  915.4113330615572
 Bronx         | 16.830887689258027 | 14.387078783469008 |  4.302425895591926 |  961.8999149975191
 Staten Island | 18.868895272463483 |  15.78369706546883 |  5.300918792979318 |  945.9410640556766
 Brooklyn      | 19.108697121929993 | 15.512255503906616 |  4.206310959429094 | 1096.0952368070498
 N/A           | 22.915405733593488 | 18.379297819544796 |   6.77749017123349 | 1142.2823328581737
 Manhattan     | 23.330579258043425 |  18.11783245681816 |  4.788125558929057 | 1184.8889437267476
 Queens        | 23.648157104428382 | 19.355855645506203 | 6.3291745725107535 | 1216.5134582592054
 EWR           |  48.29858326474749 |  11.09971358024691 |  19.57026282578872 | 2811.0533333333333
(8 rows)
```