Análisis Fórmula 1

Descripción General del Proyecto

El objetivo principal de este proyecto es crear una base de datos MySQL que pueda ser utilizada en Google Colab para realizar consultas y generar gráficos informativos.

La integración con Google Colab permite realizar consultas SQL avanzadas, lo que facilita la comprensión de la información mediante visualizaciones gráficas, especialmente útil para personas que no están familiarizadas con el entorno técnico de SQL.

El proyecto incluye la gestión de usuarios y permisos, la implementación de triggers, y procedimientos almacenados, así como funciones SQL para optimizar la manipulación y análisis de datos. Además, se ha diseñado un diagrama ER para estructurar y visualizar la base de datos de manera efectiva. Este enfoque no solo mejora el análisis del rendimiento histórico de la Fórmula 1, sino que también establece una base sólida para futuras investigaciones y aplicaciones en otros campos.

Creación y Población de Tablas en MySQL

1 Tablas Creadas

- circuits
- constructors
- drivers
- races
- results
- results_log
- users

2 Valores Insertados

- circuits: 73
- constructors: 208
- drivers: 842
- races: 997
- results: 23,727
- results_log: 4
- users: 3

Estructura de las Tablas

Circuits

```
CREATE TABLE circuits (
  circuitId int AUTO_INCREMENT NOT NULL,
  circuitRef varchar(50) NOT NULL,
  name varchar(100) NOT NULL,
  location varchar(50) NOT NULL,
  country varchar(50) NOT NULL,
  lat float NOT NULL,
  lng float NOT NULL,
  alt int NULL,
  url varchar(255) NOT NULL,
PRIMARY KEY (`circuitId`)
);
```

Drivers

```
CREATE TABLE drivers (
  driverId int AUTO_INCREMENT NOT NULL,
  driverRef varchar(50) NOT NULL,
  number int NULL,
  code varchar(10) NULL,
  forename varchar(100) NOT NULL,
  surname varchar(100) NOT NULL,
  dob date NOT NULL,
  nationality varchar(50) NOT NULL,
  url varchar(255) NOT NULL,
PRIMARY KEY (`driverId`)
);
```

Races

```
CREATE TABLE races (
  raceId int AUTO_INCREMENT NOT NULL,
  year int NOT NULL,
  round int NOT NULL,
  circuitId int NOT NULL,
  name varchar(100) NOT NULL,
  date date NOT NULL,
  time time NULL,
  url varchar(255) NOT NULL,
PRIMARY KEY (`raceId`),
KEY idx_races_circuitId (circuitId),
FOREIGN KEY (`circuitId`) REFERENCES circuits (`circuitId`)
);
```

Users

```
CREATE TABLE Users (
  user_id int AUTO_INCREMENT PRIMARY KEY,
  username VARCHAR(50) UNIQUE NOT NULL,
  password VARCHAR(255) NOT NULL,
  role ENUM ('manager', 'employee', 'analyst') NOT NULL
);
```

Constructors

```
CREATE TABLE constructors (
  constructorId int AUTO_INCREMENT NOT NULL,
  constructorRef varchar(50) NOT NULL,
  name varchar(100) NOT NULL,
  nationality varchar(50) NOT NULL,
  url varchar(255) NOT NULL,
PRIMARY KEY (`constructorId`)
);
```

Results

```
CREATE TABLE results (
  resultId int AUTO_INCREMENT NOT NULL,
  raceId int NOT NULL,
  driverId int NOT NULL,
  constructorId int NOT NULL,
  log_id int NULL,
  circuitId int NOT NULL,
  user_id int NOT NULL,
  number varchar(10) NULL,
  grid int NOT NULL,
  position varchar(10) NULL,
  positionText varchar(10) NULL,
  positionOrder int NOT NULL,
  points float NOT NULL,
  laps int NOT NULL,
  time varchar(255) NULL,
  milliseconds int NULL,
  fastestLap int NULL,
  rank int NULL,
  fastestLapTime varchar(255) NULL,
  fastestLapSpeed float NULL,
  statusId int NOT NULL,
PRIMARY KEY (`resultId`),
FOREIGN KEY (`raceId`) REFERENCES races (`raceId`),
FOREIGN KEY (`driverId`) REFERENCES drivers (`driverId`),
FOREIGN KEY (`constructorId`) REFERENCES constructors (`constructorId`),
FOREIGN KEY (`circuitId`) REFERENCES circuits (`circuitId`)
);
```

Results_log

```
CREATE TABLE results_log (
  log_id int AUTO_INCREMENT NOT NULL,
  user_id int NOT NULL,
  driverId int NOT NULL,
  raceId int NOT NULL,
  position int NULL,
  log_time timestamp NOT NULL,
PRIMARY KEY (`log_id`)
);
```

Gestión de Usuarios y Permisos

Usuarios Creados

manager_user,

employer_user,

analyst_user

Permisos Asignados

manager_user: inserción, eliminación y actualización.

employer_user: inserción.

analyst_user: inserción y creación de tablas temporales.

Claves Foráneas

fk_results_users en la columna user_id de la tabla results.

fk_results_log_users en la columna user_id de la tabla results_log.

SQL para la gestión de usuarios y permisos:

```
-- Creación de usuarios

CREATE USER 'manager_user'@'localhost' IDENTIFIED BY 'password';

CREATE USER 'employer_user'@'localhost' IDENTIFIED BY 'password';

CREATE USER 'analyst_user'@'localhost' IDENTIFIED BY 'password';

-- Asignación de roles y permisos

GRANT INSERT, DELETE, UPDATE ON formula1db.* TO 'manager_user'@'localhost';

GRANT INSERT ON formula1db.* TO 'employer_user'@'localhost';

GRANT INSERT, CREATE TEMPORARY TABLES ON formula1db.* TO 'analyst_user'@'localhost';

-- Creación de claves foráneas

ALTER TABLE results ADD CONSTRAINT fk_results_users FOREIGN KEY (user_id) REFERENCES users(user_id);

ALTER TABLE results_log ADD CONSTRAINT fk_results_log_users FOREIGN KEY (user_id) REFERENCES users(user_id);
```

Implementación de Triggers

1 Actualizar Clasificación de Pilotos

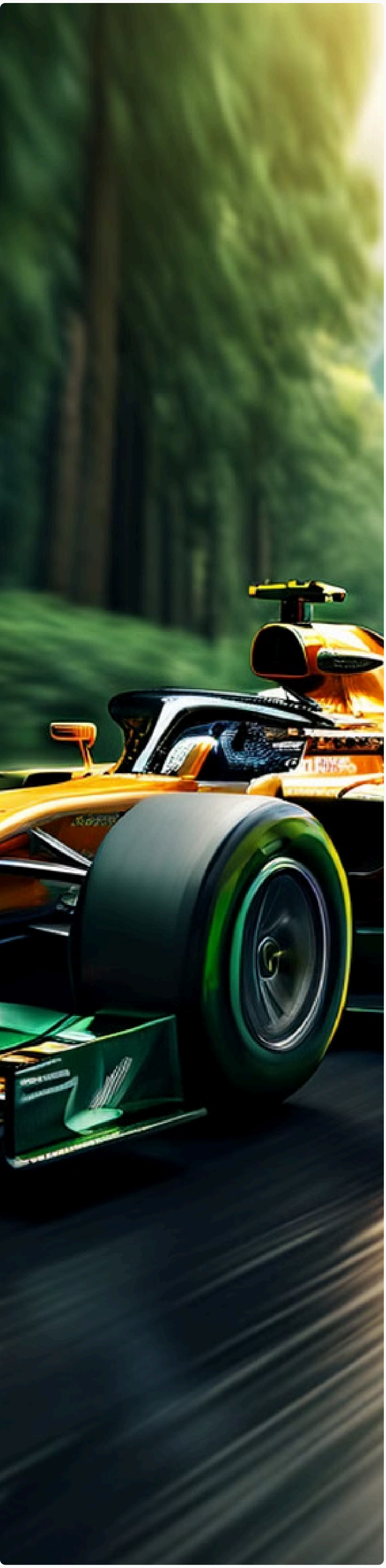
Trigger que actualiza automáticamente los puntos de los pilotos en la tabla drivers después de insertar un nuevo resultado en la tabla results.

```
-- Trigger para actualizar la clasificación de pilotos
DELIMITER //
CREATE TRIGGER actualizar_clasificacion
AFTER INSERT ON results
FOR EACH ROW
BEGIN
    UPDATE drivers
    SET points = points + NEW.points
    WHERE driverId = NEW.driverId;
END;
DELIMITER ;
```

2 Mantener Log de Resultados

Triggers que registran cada operación (INSERT, UPDATE, DELETE) realizada en la tabla results en una tabla de log llamada results_log.

```
DELIMITER //
CREATE TRIGGER log_insert
AFTER INSERT ON results
FOR EACH ROW
BEGIN
    INSERT INTO result_log (operation, timestamp, result_id, new_data)
    VALUES ('INSERT', NOW(), NEW.resultId, NEW.points);
END//
CREATE TRIGGER log_update
AFTER UPDATE ON results
FOR EACH ROW
BEGIN
    INSERT INTO result_log (operation, timestamp, result_id, old_data, new_data)
    VALUES ('UPDATE', NOW(), OLD.resultId, OLD.points, NEW.points);
END//
CREATE TRIGGER log_delete
AFTER DELETE ON results
FOR EACH ROW
BEGIN
    INSERT INTO result_log (operation, timestamp, result_id, old_data)
    VALUES ('DELETE', NOW(), OLD.resultId, OLD.points);
END//
DELIMITER ;
```



Store Procedure

update_driver_info

Procedimiento almacenado que actualiza la nacionalidad, número, nombre y apellidos de un piloto en función de su ID.

```
-- Eliminar el procedimiento si ya existe
DROP PROCEDURE IF EXISTS update_driver_Info;
DELIMITER //
-- Crear el procedimiento almacenadoupdate_driver_Info
CREATE PROCEDURE update_driver_Info(
    IN pDriverId INT,
    IN pNewNationality VARCHAR(255),
    IN pNewNumber INT,
    IN pNewForename VARCHAR(255),
    IN pNewSurname VARCHAR(255)
)
BEGIN
-- Actualizar la nacionalidad si se proporciona un nuevo valor
IF pNewNationality IS NOT NULL THEN
   UPDATE drivers SET nationality = pNewNationality    WHERE driverId = pDriverId;
END IF;

   -- Actualizar el número si se proporciona un nuevo valor

   IF pNewNumber IS NOT NULL THEN
      UPDATE drivers SET number = pNewNumber WHERE driverId = pDriverId;
   END IF;

   -- Actualizar el nombre si se proporciona un nuevo valor

   IF pNewForename IS NOT NULL THEN
      UPDATE drivers SET forename = pNewForename WHERE driverId = pDriverId;
   END IF;

   -- Actualizar el apellido si se proporciona un nuevo valor

   IF pNewSurname IS NOT NULL THEN
      UPDATE drivers SET surname = pNewSurname WHERE driverId = pDriverId;
   END IF;


END//
DELIMITER ;
-- Ejemplo de cómo llamar al procedimiento almacenado -- Actualizar la nacionalidad del conductor con ID 123 a 'Belgian'
CALL update_driver_Info(123, 'Belgian', NULL, NULL, NULL);
```

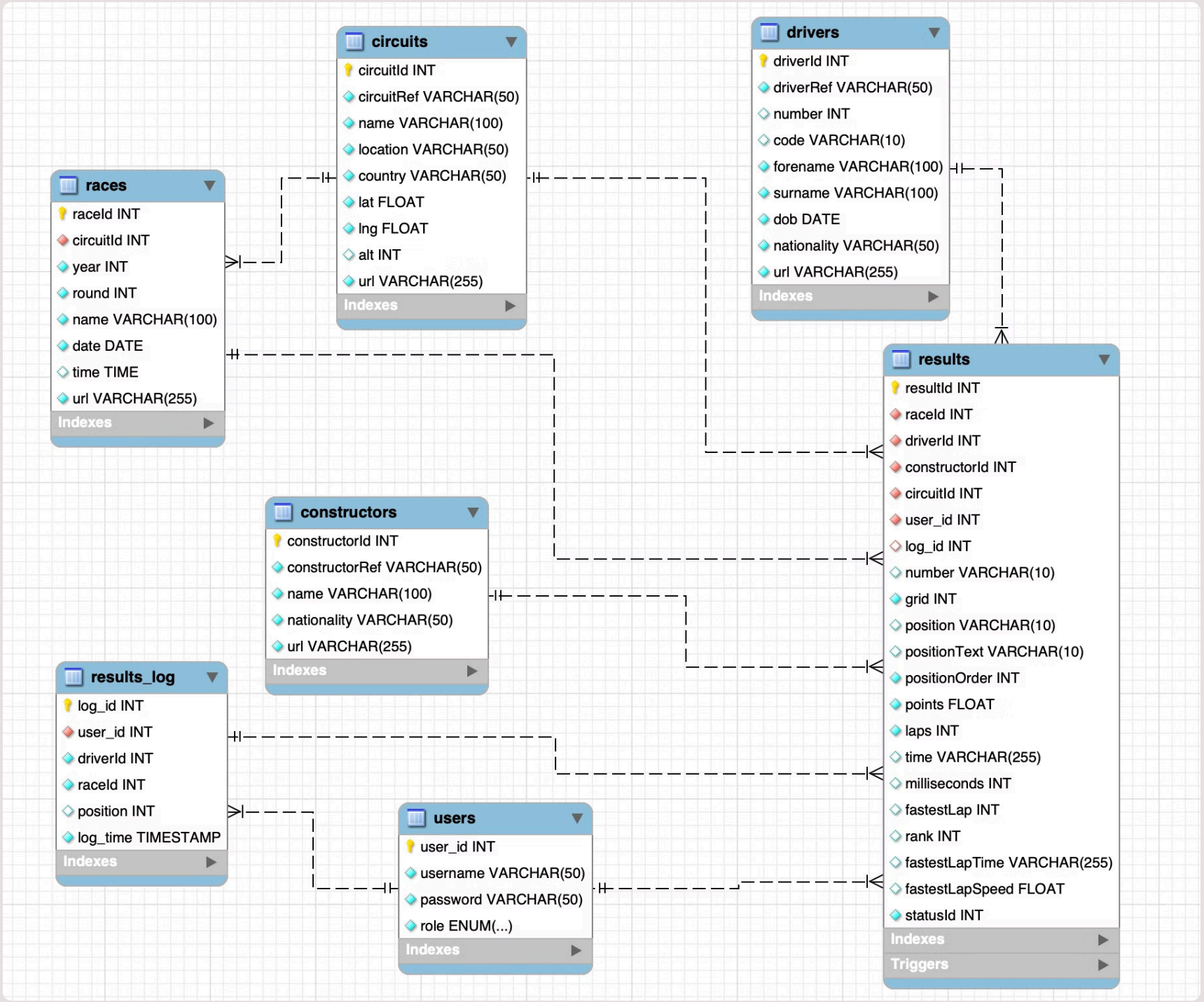
Funciones SQL

La función average_points calcula el promedio de puntos obtenidos por un piloto específico basado en su driverId. Esta función es útil para analizar el rendimiento de los pilotos en las carreras. Se incluyen ejemplos de cómo utilizar la función para obtener el promedio de puntos de diferentes pilotos, incluyendo aquellos con una nacionalidad específica, y cómo ordenar y limitar los resultados para obtener los pilotos con mejores promedios.

```
USE Formula1DB; DROP FUNCTION IF EXISTS average_points_per_driverid;

DELIMITER //

CREATE FUNCTION average_points_per_driverid (p_driverid INT)

RETURNS DECIMAL(10,2)

DETERMINISTIC

BEGIN

  DECLARE avg_points DECIMAL(10,2);

-- Calcular el promedio de puntos
SELECT AVG(gp.points) INTO avg_points
FROM results
WHERE driverid = p_driverid;

RETURN avg_points;

END //

DELIMITER ;


-- Probar la función average_points con un driverid específico

SELECT driverid, average_points_per_driverid(driverid) AS avg_points

FROM drivers

WHERE driverid = 1;


-- Seleccionar el driverid y el promedio de puntos para los primeros 10 pilotos

SELECT driverid, average_points_per_driverid(driverid)

FROM drivers LIMIT 10;


-- Seleccionar el driverid y el promedio de puntos para pilotos específicos con Id 1, 12, 39, 48, 127

SELECT driverid, average_points_per_driverid(driverid)

FROM drivers

WHERE driverid IN (1, 12, 39, 48, 127);


-- Seleccionar el driverId y el promedio de puntos para los pilotos de nacionalidad 'British', limitando a 10 resultados

SELECT driverId, average_points(driverId)

FROM drivers WHERE nationality = 'British'

LIMIT 10;


-- Seleccionar el driverId y el promedio de puntos, ordenar por promedio de puntos en orden descendente, y mostrar los 10 mejores pilotos

SELECT driverId, average_points(driverId) AS avg_points

FROM drivers

ORDER BY avg_points DESC

LIMIT 10;
```

Diagrama EER

El diagrama EER muestra la estructura y relaciones de las tablas en la base de datos de Fórmula 1, permitiendo una visualización clara de cómo se conectan los diferentes elementos de la información.



Análisis y Visualización en Google Colab

Conexión a MySQL

Conexión a la base de datos MySQL desde Google Colab para realizar consultas y análisis.

Consultas SQL

Uso de consultas SQL avanzadas como SELECT, GROUP BY, JOIN, EXCEPT, WHERE, INSERT, UPDATE, DELETE y ORDER BY.

Visualización de Datos

Generación de gráficos y visualizaciones para analizar patrones y tendencias en los datos de Fórmula 1.

Link del Proyecto

Google Colab

Resumen del Proyecto

1

Diseño y creación - Base de Datos en MySQL

Creación y estructuración de una base de datos en MySQL, optimizada para almacenar y gestionar eficientemente la información requerida para su análisis.

2

MySQL Workbench

Se ha llevado a cabo la gestión de usuarios y permisos, la implementación de **triggers**, procedimientos almacenados (**stored procedures**) y funciones SQL, así como la creación de un diagrama EER para visualizar la estructura de la base de datos.

3

Análisis de Datos - Google colab

Utilización de Google Colab para el Uso de consultas SQL avanzadas como SELECT, GROUP BY, JOIN, EXCEPT, WHERE, INSERT, UPDATE, DELETE y ORDER BY.

4

Google colab - Visualización

Generación de gráficos y visualizaciones para comprender mejor los patrones y tendencias en los datos.

5

Insights Clave

Identificación de factores clave que influyen en el rendimiento de pilotos, equipos y carreras de Fórmula 1.

Este proyecto demuestra la eficacia de utilizar una base de datos MySQL combinada con Google Colab para realizar consultas que generen gráficos y proporcionen información valiosa no disponible directamente a través de MySQL. Aunque las consultas SQL avanzadas como SELECT, GROUP BY, JOIN, EXCEPT, WHERE, INSERT, UPDATE, DELETE y ORDER BY se pueden ejecutar perfectamente en MySQL, hemos optado por realizarlas también en Google Colab. Lo más importante es que Google Colab permite generar gráficos que facilitan la comprensión de la información, especialmente para clientes o personas que no están familiarizadas con el ambiente de consultas SQL y programación.

Las herramientas de análisis avanzadas permiten obtener **insights** profundos y valiosos sobre la Fórmula 1. Las técnicas empleadas no solo permiten una comprensión detallada del rendimiento histórico, sino que también proporcionan una base sólida para futuras investigaciones y análisis. Este enfoque integral puede ser aplicado a otros deportes o áreas de interés, destacando la versatilidad y el poder del análisis de datos en la toma de decisiones informadas y estratégicas..