Nurburgring Lap Records DB

Pictured: Lexus LFA Nurburgring Edition

Jonathan Spence 5/2/2017

Table of Contents

Executive Summary

The Nurburgring Nordschleife is a world famous racing circuit in Germany that is considered a benchmark for performance cars. Performance cars are regularly put through their paces to determine how long it takes them to go around the ‘ring. Nurburgring lap times are held in such high regard that they can become a part of a car’s identity alongside it’s horsepower, price, and other stats.

The E/R Diagram depicts the structure of the database, relationships between those tables, and the columns which make up the tables. Each of these thirteen tables are detailed in their individual slides, describing the table and displaying the sql code used to create them, and sample data for each one. After that, sample views, stored procedures, triggers, and security roles are demonstrated. This report also contains notes on the implementation and current issues, as well as possible future enhancements to the database. SQL is bolded for contrast.

E/R Diagram

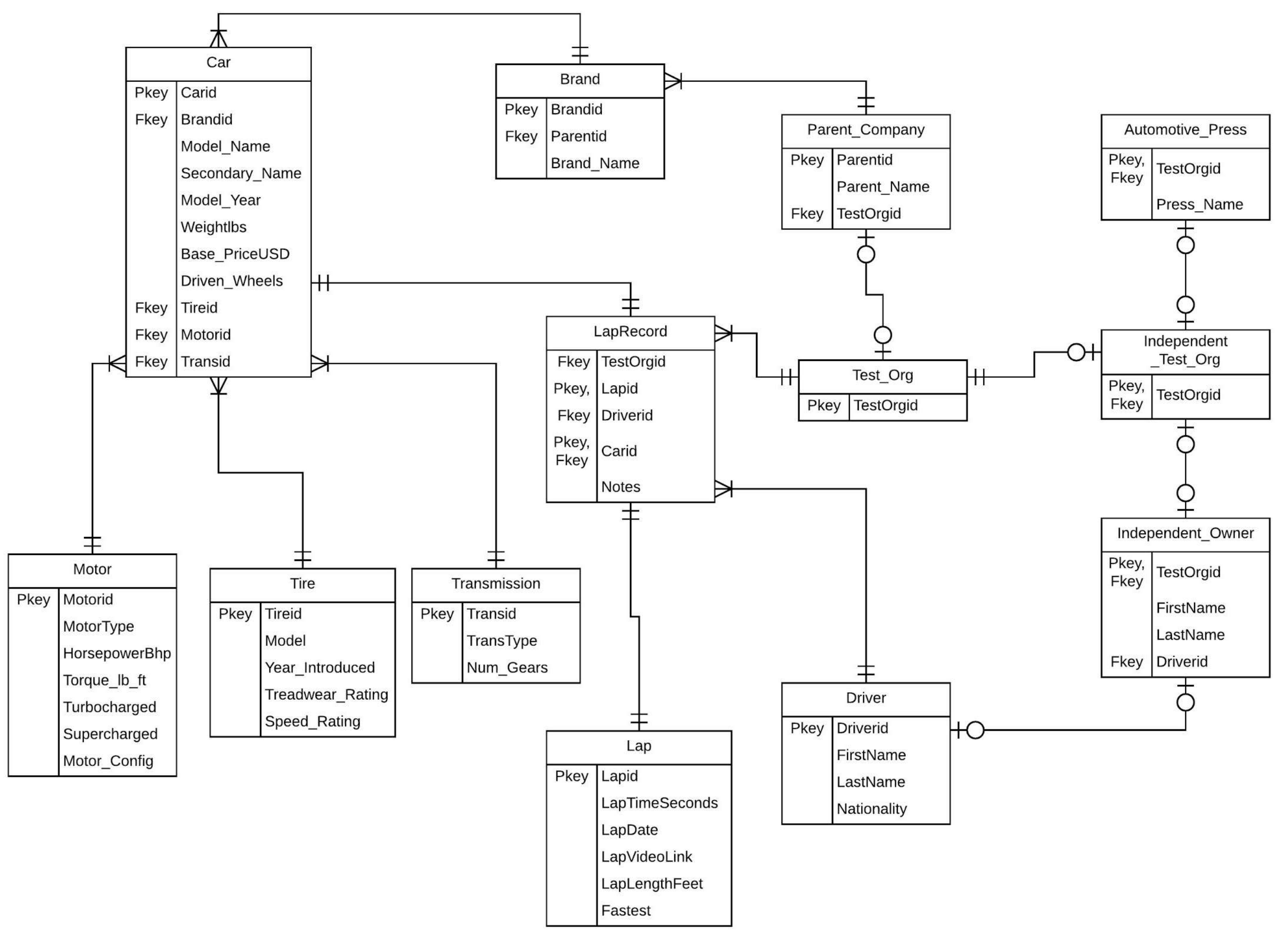

Car
Pkey Carid
Fkey Brandid
Model_Name
Secondary_Name
Model_Year
Weightlbs
Base_PriceUSD
Driven_Wheels
Fkey Tireid
Fkey Motorid
Fkey Transid
Brand
Pkey Brandid
Fkey Parentid
Brand_Name
Parent_Company
Pkey Parentid
Parent_Name
Fkey TestOrgid
Automotive_Press
Pkey, Fkey TestOrgid
Press_Name
LapRecord
Fkey TestOrgid
Pkey, Lapid
Fkey Driverid
Pkey, Fkey Carid
Notes
Test_Org
Pkey TestOrgid
Independent _Test_Org
Pkey, Fkey TestOrgid
Independent_Owner
Pkey, Fkey TestOrgid
FirstName
LastName
Fkey Driverid
Motor
Pkey Motorid
MotorType
HorsepowerBhp
Torque_lb_ft
Turbocharged
Supercharged
Motor_Config
Tire
Pkey Tireid
Model
Year_Introduced
Treadwear_Rating
Speed_Rating
Transmission
Pkey Transid
TransType
Num_Gears
Lap
Pkey Lapid
LapTimeSeconds
LapDate
LapVideoLink
LapLengthFeet
Fastest
Driver
Pkey Driverid
FirstName
LastName
Nationality

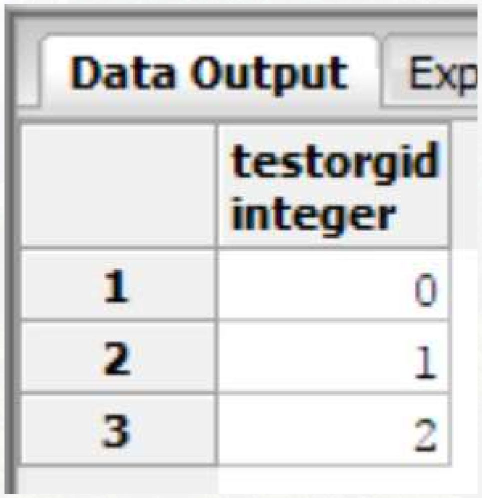
Table: Test_Org

The Test_Org table keeps track of all organizations and entities which are responsible for a lap record at the Nurburgring. These consist of independent test organizations, and a parent company which may test its own vehicles.

Functional dependencies: TestOrgid →

SQL:

```
CREATE TABLE Test_Org(
TestOrgid		INT NOT NULL UNIQUE,
PRIMARY KEY(TestOrgid)
);
```



	testorgid integer
1	0
2	1
3	2

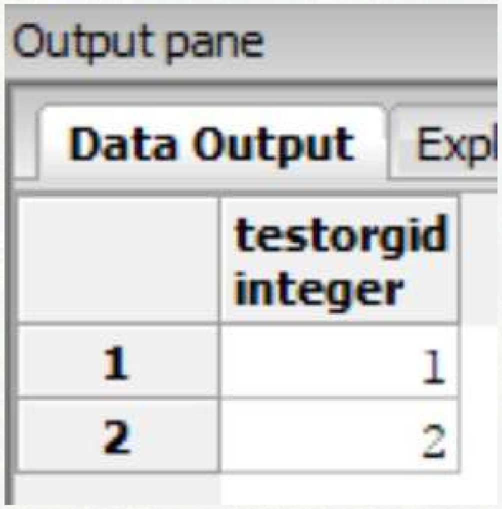
Table: Independent_Test_Org

The Independant_Test_Org table keeps track of all independent organizations and entities which have created a lap record at the track. These currently consist of automotive press and independent owners.

Functional dependencies: TestOrgid →

SQL:

```
CREATE TABLE Independent_Test_Org(
TestOrgid           INT NOT NULL UNIQUE
                    REFERENCES Test_Org(TestOrgid),
PRIMARY KEY(TestOrgid)
);
```



	testorgid integer
1	1
2	2

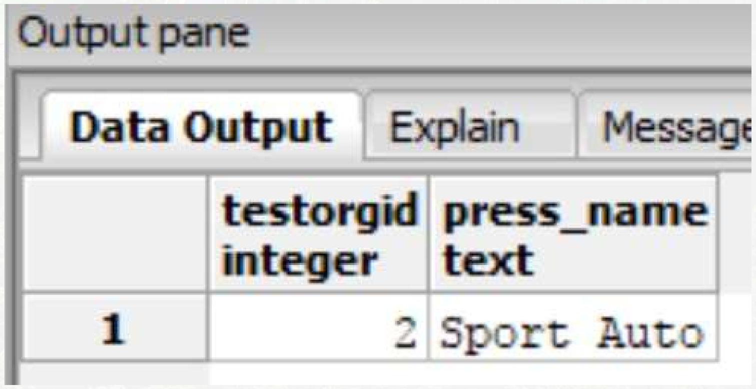
Table: Automotive_Press

The Automotive_Press table keeps track of all press organizations which have created a lap record at the track. Press organizations are independent organizations.

Functional dependencies: TestOrgid → Press_Name

SQL:

```
CREATE TABLE Automotive_Press(
TestOrgid            INT NOT NULL UNIQUE REFERENCES
Independent_Test_Org(TestOrgid),
Press_Name           TEXT NOT NULL,
PRIMARY KEY(TestOrgid)
);
```



	testorgid integer	press_name text
1	2	Sport Auto

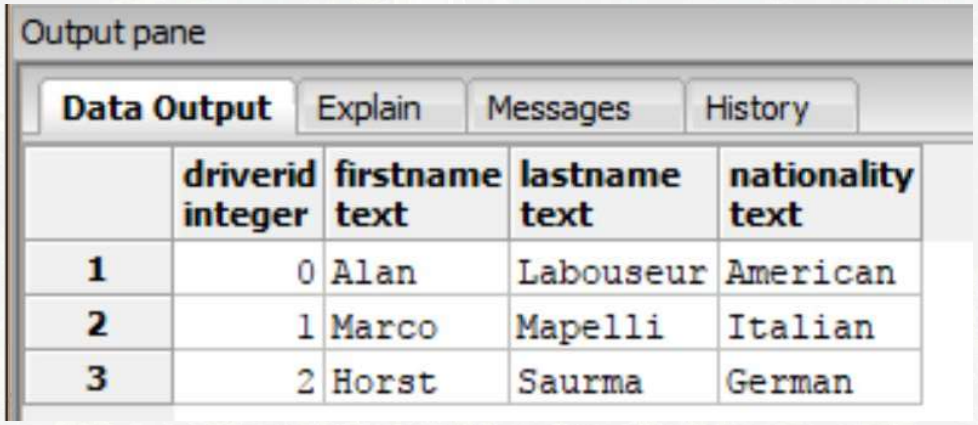
Table: Driver

The Driver table keeps track of all drivers which drove a car that made a lap record.

Functional dependencies: Driverid → FirstName, LastName, Nationality

SQL:

```
CREATE TABLE Driver(
Driverid        INT NOT NULL UNIQUE,
FirstName       TEXT NOT NULL,
LastName        TEXT NOT NULL,
Nationality     TEXT NOT NULL,
PRIMARY KEY(Driverid)
);
```



	driverid integer	firstname text	lastname text	nationality text
1	0	Alan	Labouseur	American
2	1	Marco	Mapelli	Italian
3	2	Horst	Saurma	German

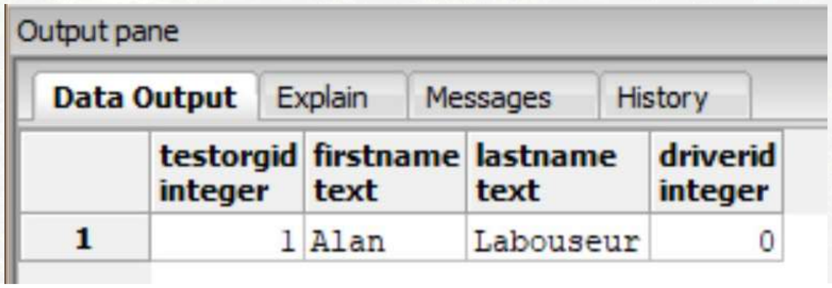
Table: Independent_Owner

The Independent_Owner table tracks car owners who have their car used in a lap record. An owner may or may not be the driver during this lap.

Functional dependencies: TestOrgid → FirstName, LastName, Driverid

SQL:

```
CREATE TABLE Independent_Owner(
TestOrgid           INT  NOT NULL UNIQUE REFERENCES
Independent_Test_Org(TestOrgid),
FirstName           TEXT NOT NULL,
LastName            TEXT NOT NULL,
Driverid            INT REFERENCES Driver(Driverid),
PRIMARY KEY(TestOrgid)
);
```



	testorgid integer	firstname text	lastname text	driverid integer
1	1	Alan	Labouseur	0

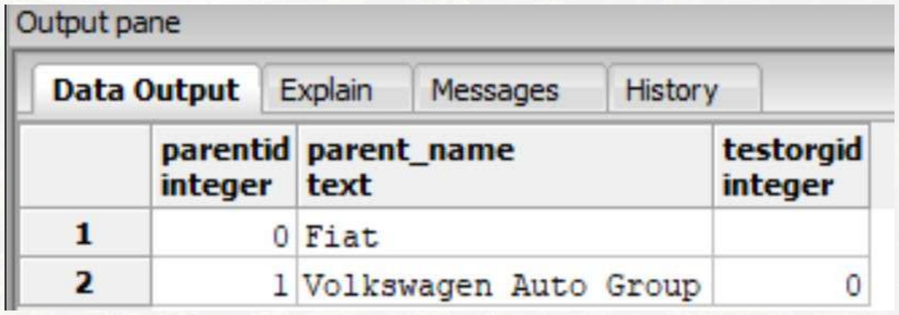
Table: Parent_Company

The Parent_Company table tracks the parent companies of brand, and cars which record lap records. A parent company can also conduct a lap record as a test organization

Functional dependencies: Parentid → Parent_Name, TestOrgid

SQL:

```
CREATE TABLE Parent_Company (
  Parentid          INT NOT NULL UNIQUE,
  Parent_Name       TEXT NOT NULL,
  TestOrgid         INT UNIQUE REFERENCES Test_Org(TestOrgid),
  PRIMARY KEY(Parentid)
);
```



	parentid integer	parent_name text	testorgid integer
1	0	Fiat	
2	1	Volkswagen Auto Group	0

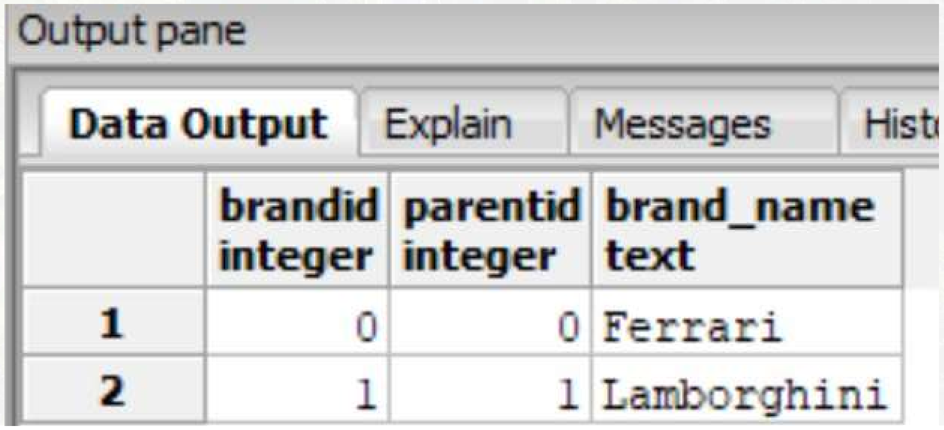
Table: Brand

The Brand table tracks the brands of cars which record a lap record. Cars have a brand, and brands have a parent company.

Functional dependencies: Brandid → Parentid, Brand_Name

SQL:

```
CREATE TABLE Brand (
  Brandid          INT NOT NULL UNIQUE,
  Parentid         INT NOT NULL REFERENCES Parent_Company(Parentid),
  Brand_Name       TEXT NOT NULL,
  PRIMARY KEY(Brandid)
);
```



	brandid integer	parentid integer	brand_name text
1	0	0	Ferrari
2	1	1	Lamborghini

Table: Motor

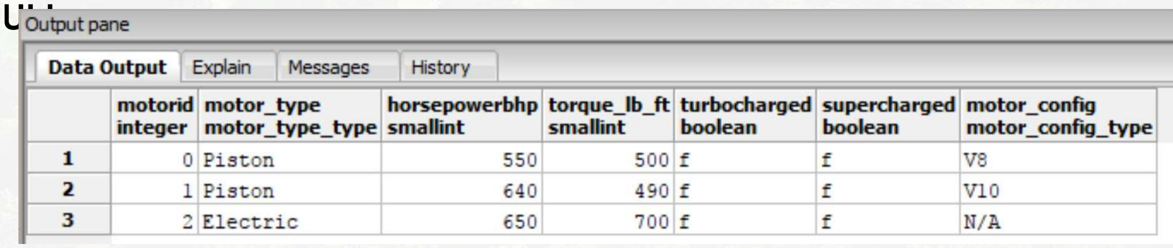
The Motor table tracks data relevant to a motor, which is a part of each car which creates a lap record.

Functional dependencies: Motorid → Motor_Type, HorsePowerBHP, Torque_lb_ft, Turbocharged, Supercharged, Motor_Config

SQL:

```
CREATE TYPE motor_type_type AS ENUM ('Piston', 'Rotary', 'Electric', 'Other');
CREATE TYPE motor_config_type AS ENUM('H4', 'H6', 'I4', 'I6', 'V4', 'V8', 'V10',
                                      'V12', 'W8', 'W12', 'V16', 'other', 'N/A');

CREATE TABLE Motor (
  Motorid            INT NOT NULL UNIQUE,
  Motor_Type         motor_type_type NOT NULL,
  HorsepowerBHP      SMALLINT NOT NULL CHECK(HorsepowerBHP > 0),
  Torque_lb_ft       SMALLINT NOT NULL CHECK(Torque_lb_ft > 0),
  Turbocharged       BOOLEAN NOT NULL,
  Supercharged       BOOLEAN NOT NULL,
  Motor_Config       motor_config_type NOT NU
  PRIMARY KEY(Motorid)
);
```


Output pane

Data Output | Explain | Messages | History

	motorid integer	motor_type motor_type_type	horsepowerbhp smallint	torque_lb_ft smallint	turbocharged boolean	supercharged boolean	motor_config motor_config_type
1	0	Piston	550	500	f	f	V8
2	1	Piston	640	490	f	f	V10
3	2	Electric	650	700	f	f	N/A

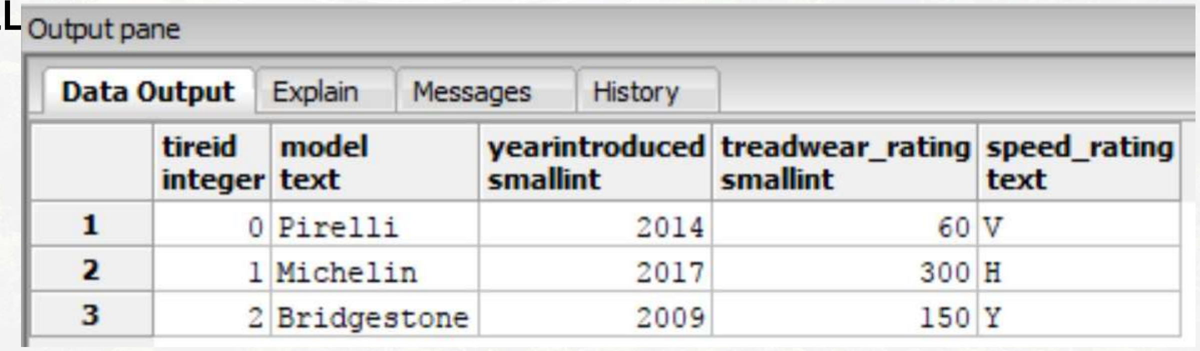
Table: Tire

The Tire table tracks data relevant to a tire, which is a part of each car that creates a lap record.

Functional dependencies: Tireid → model, YearIntroduced, Treadwear_Rating, Speed_Rating

SQL:

```
CREATE TABLE Tire (
  Tireid               INT NOT NULL UNIQUE,
  model                TEXT NOT NULL,
  YearIntroduced       SMALLINT NOT NULL CHECK(YearIntroduced > 1900),
  Treadwear_Rating     SMALLINT NOT NULL,
  Speed_Rating         TEXT NOT NULL
  PRIMARY KEY(Tireid)
);
```


Output pane

Data Output | Explain | Messages | History

	tireid integer	model text	yearintroduced smallint	treadwear_rating smallint	speed_rating text
1	0	Pirelli	2014	60	V
2	1	Michelin	2017	300	H
3	2	Bridgestone	2009	150	Y

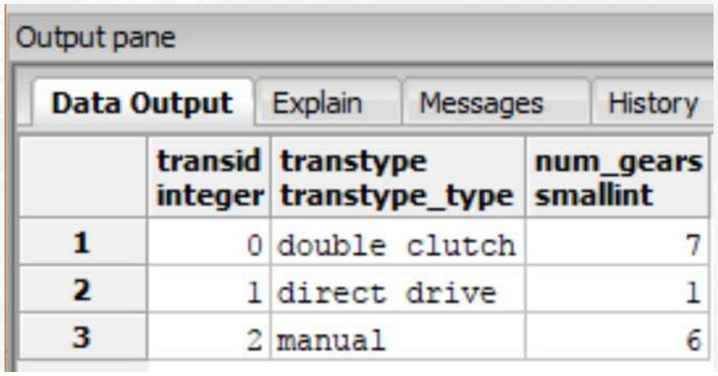
Table: Transmission

The Transmission table tracks data relevant to a transmission, which is a part of each car that creates a lap record.

Functional dependencies: Transid → TransType, Num_Gears

SQL:

```
CREATE TYPE transtype_type AS ENUM ('double clutch', 'single clutch', 'torque-converter', 'direct
drive', 'manual', 'other');
CREATE TABLE Transmission (
  Transid                         INT NOT NULL UNIQUE,
  TransType                            transtype_type NOT NULL,
  Num_Gears                       SMALLINT NOT NULL CHECK (Num_Gears > 0),
  PRIMARY KEY(Transid)
);
```



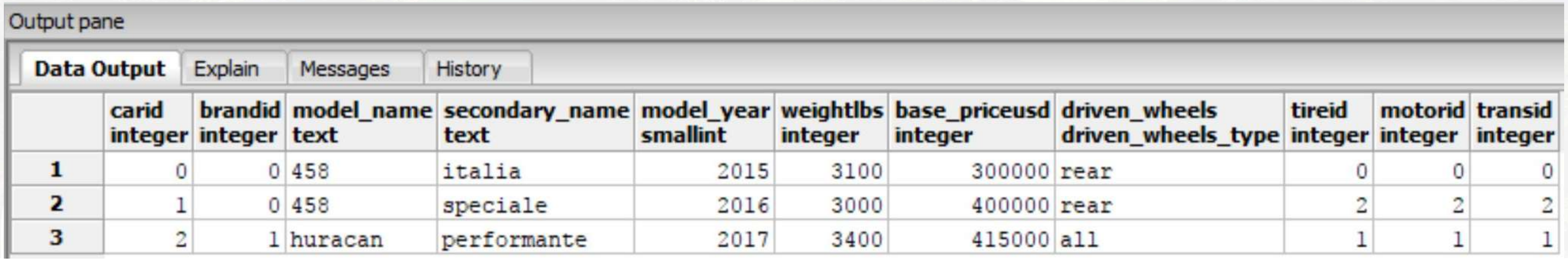
```
   transid  transtype       num_gears
   integer  transtype_type  smallint
1        0  double clutch           7
2        1  direct drive            1
3        2  manual                  6
```

Table: Car

The Car table tracks all information relevant to a car. A car will appear in the database if it makes a lap record. A car has a motor, transmission, and tires, and also is a member of a brand.

Functional dependencies: Carid → Brandid, Model_Name, Secondary_Name, Model_Year, Weightlbs, Base_PriceUSD, Driven_Wheels, Tireid, Motorid, Transid

```
CREATE TYPE driven_wheels_type AS ENUM ('front', 'rear', 'all', 'other');
CREATE TABLE Car (
  Carid                 INT NOT NULL UNIQUE,
  Brandid               INT NOT NULL REFERENCES Brand(Brandid),
  Model_Name            TEXT NOT NULL,
  Secondary_Name        TEXT NOT NULL,
  Model_Year            SMALLINT NOT NULL CHECK(Model_Year > 1900),
  Weightlbs             INT NOT NULL CHECK(Weightlbs > 0),
  Base_PriceUSD         INT NOT NULL,
  Driven_Wheels         driven_wheels_type NOT NULL,
  Tireid                          INT NOT NULL REFERENCES Tire(Tireid),
  Motorid               INT NOT NULL REFERENCES Motor(Motorid),
  Transid               INT NOT NULL REFERENCES Transmission(Transid),
  PRIMARY KEY(Carid)
);
```



Output pane

Data Output | Explain | Messages | History

	carid integer	brandid integer	model_name text	secondary_name text	model_year smallint	weightlbs integer	base_priceusd integer	driven_wheels driven_wheels_type	tireid integer	motorid integer	transid integer
1	0	0	458	italia	2015	3100	300000	rear	0	0	0
2	1	0	458	speciale	2016	3000	400000	rear	2	2	2
3	2	1	huracan	performante	2017	3400	415000	all	1	1	1

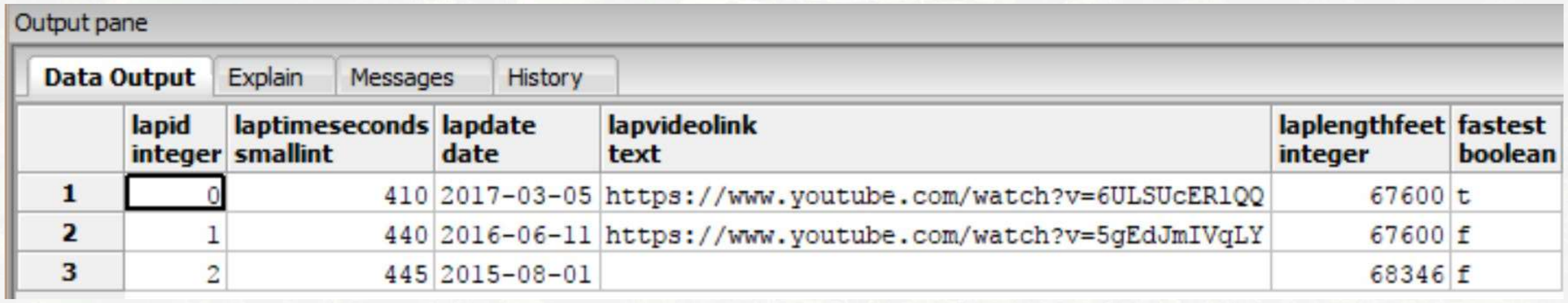
Table: Lap

The lap table tracks relevant information to a lap which are related to a lap record.

Functional dependencies: Lapid → LapTimeSeconds, LapDate, LapVideoLink, LapLengthFeet, Fastest

SQL:

```
CREATE TABLE Lap(
Lapid               INT NOT NULL UNIQUE,
LapTimeSeconds          SMALLINT NOT NULL CHECK(LapTimeSeconds > 0),
LapDate             date NOT NULL CHECK(LapDate > '1900-01-01'),
LapVideoLink        TEXT,
LapLengthFeet       INT NOT NULL CHECK(LapLengthFeet > 0),
Fastest             BOOLEAN NOT NULL,
PRIMARY KEY(Lapid)
);
```



	lapid integer	laptimeseconds smallint	lapdate date	lapvideolink text	laplengthfeet integer	fastest boolean
1	0	410	2017-03-05	https://www.youtube.com/watch?v=6ULSUcER1QQ	67600	t
2	1	440	2016-06-11	https://www.youtube.com/watch?v=5gEdJmIVqLY	67600	f
3	2	445	2015-08-01		68346	f

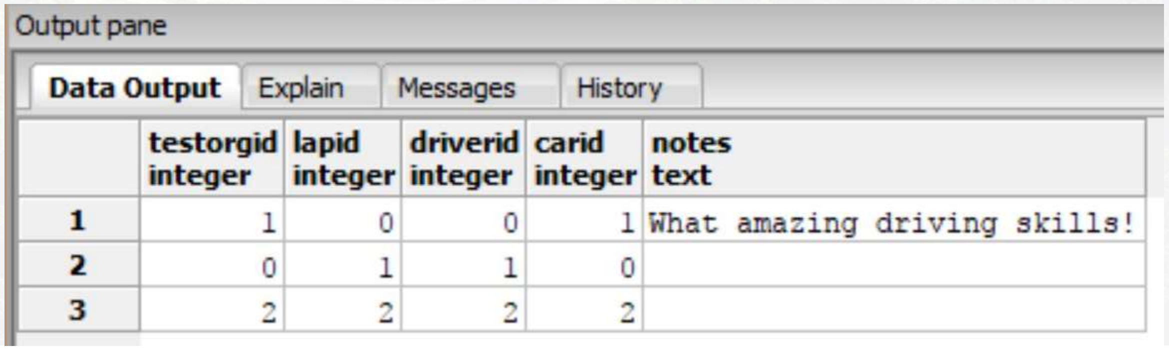
Table: LapRecord

The LapRecord table records a lap record by storing the primary keys of the car, the driver who drove the lap, the test organization which brought the car, and the lap information. A car can only appear on the lapRecord table once.

Functional dependencies: Carid → TestOrgid, Lapid, Driverid, Notes

SQL:

```
CREATE TABLE LapRecord(
TestOrgid          INT NOT NULL REFERENCES Test_Org(TestOrgid),
Lapid              INT NOT NULL UNIQUE REFERENCES Lap(Lapid),
Driverid           INT NOT NULL REFERENCES Driver(Driverid),
Carid              INT NOT NULL UNIQUE REFERENCES Car(Carid),
Notes                     TEXT,
PRIMARY KEY(Carid)
);
```



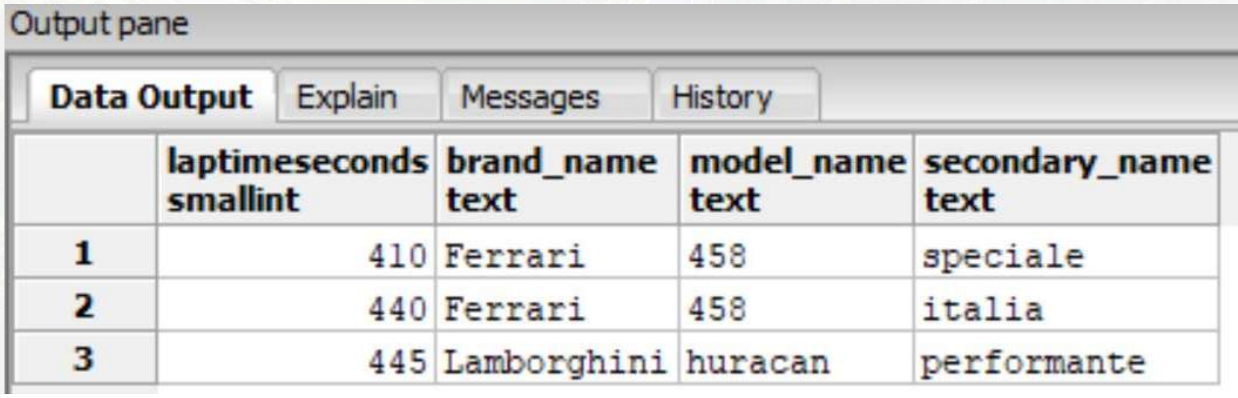
	testorgid integer	lapid integer	driverid integer	carid integer	notes text
1	1	0	0	1	What amazing driving skills!
2	0	1	1	0	
3	2	2	2	2	

View: MainTable

This view displays only the most key data to a lap record, the cars brand and names, along with the lap time by ascending lap time.

```
CREATE VIEW MainTable AS
SELECT LapTimeSeconds, Brand_Name, Model_Name, Secondary_Name
FROM Brand, Car, Lap, LapRecord
Where Car.Brandid = Brand.Brandid AND Lap.lapid=LapRecord.lapid
AND LapRecord.Carid = Car.Carid
ORDER BY LapTimeSeconds ASC;
```

```
Select * From MainTable;
```



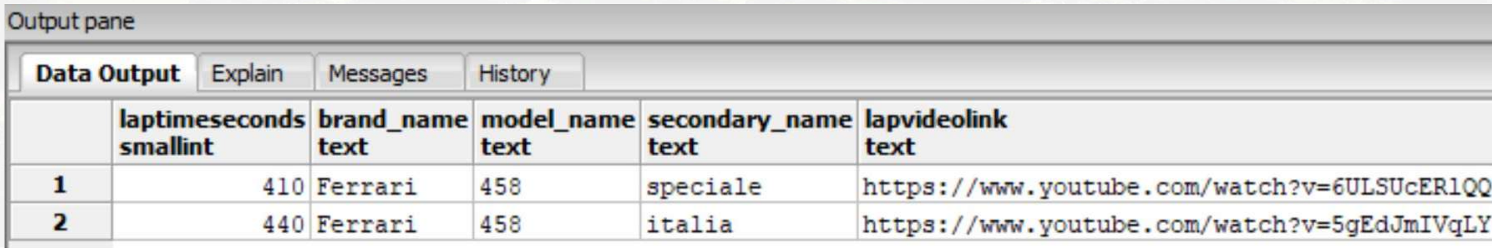
	laptimeseconds smallint	brand_name text	model_name text	secondary_name text
1	410	Ferrari	458	speciale
2	440	Ferrari	458	italia
3	445	Lamborghini	huracan	performante

View: VideoTable

This view displays key data about a lap record along with a link to a video proof of the lap. Many will only believe in the validity of a lap time if they can see video evidence of it, so many users will want to filter by only entries with video links

```
CREATE View VideoTable AS
SELECT LapTimeSeconds, Brand_Name, Model_Name, Secondary_Name,
LapVideoLink
FROM Brand, Car, Lap, LapRecord
Where Car.Brandid = Brand.Brandid AND Lap.lapid=LapRecord.lapid
AND LapRecord.Carid = Car.Carid AND LapVideoLink IS NOT NULL
ORDER BY LapTimeSeconds ASC;
```

```
Select * From VideoTable;
```



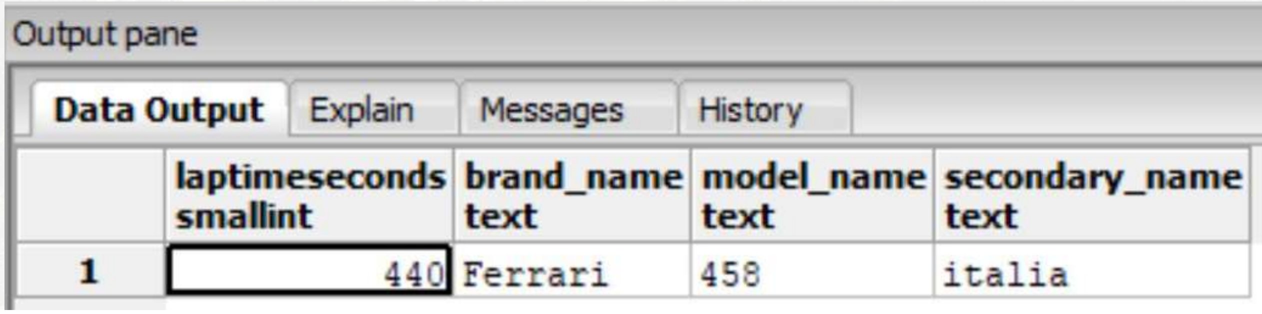
	laptimeseconds smallint	brand_name text	model_name text	secondary_name text	lapvideolink text
1	410	Ferrari	458	speciale	https://www.youtube.com/watch?v=6ULSUcER1QQ
2	440	Ferrari	458	italia	https://www.youtube.com/watch?v=5gEdJmIVqLY

Stored Procedure: recordSearchByCarNames

This stored procedure allows a user to input the name of a car, using its model name and secondary name, to find the lap time of that car around the Nurburgring.

```
CREATE OR REPLACE FUNCTION recordSearchByCarNames(TEXT,TEXT, REFCURSOR) RETURNS refcursor AS
$$
DECLARE
  NameOne TEXT      := $1;
  NameTwo TEXT        := $2;
  resultset   REFCURSOR := $3;
BEGIN
  OPEN resultset FOR
    SELECT LapTimeSeconds, Brand_Name, Model_Name, Secondary_Name
    FROM   Brand, Car, Lap, LapRecord
    WHERE  Car.Brandid = Brand.Brandid AND Lap.lapid=LapRecord.lapid
AND LapRecord.Carid = Car.Carid AND Model_Name=NameOne AND Secondary_Name=NameTwo;
  RETURN resultset;
END;
$$
LANGUAGE plpgsql;

SELECT recordSearchByCarNames('458', 'italia', 'results');
FETCH ALL FROM results;
```



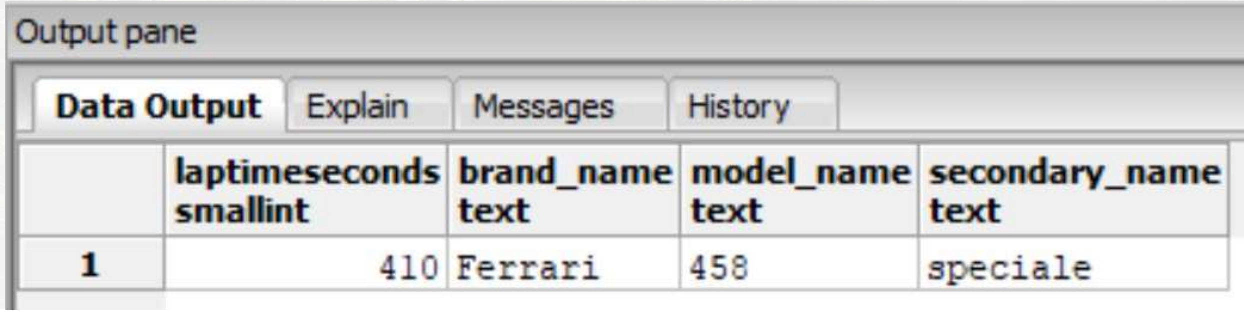
	laptimeseconds smallint	brand_name text	model_name text	secondary_name text
1	440	Ferrari	458	italia

Stored Procedure: recordSearchByMaxTime

This stored procedure allows a user to filter out records by inputting the slowest time they want to see records for. Only records with lower times will be displayed.

```
CREATE OR REPLACE FUNCTION recordSearchByMaxTime(INT, REFCURSOR) RETURNS refcursor AS
$$
DECLARE
  MaxTime   INT            := $1;
  resultset   REFCURSOR := $2;
BEGIN
  OPEN resultset FOR
    SELECT LapTimeSeconds, Brand_Name, Model_Name, Secondary_Name
    FROM   Brand, Car, Lap, LapRecord
    WHERE  Car.Brandid = Brand.Brandid AND Lap.lapid=LapRecord.lapid
AND LapRecord.Carid = Car.Carid AND LapTimeSeconds<=MaxTime;
  RETURN resultset;
END;
$$
LANGUAGE plpgsql;

SELECT recordSearchByMaxTime(430, 'results');
FETCH ALL FROM results;
```



	laptimeseconds smallint	brand_name text	model_name text	secondary_name text
1	410	Ferrari	458	speciale

Trigger: FastestLap

The FastestLap trigger assigns checks to see if each new lap recorded is the fastest. If a lap is the fastest, it's 'Fastest' attribute will be assigned true.

Function:

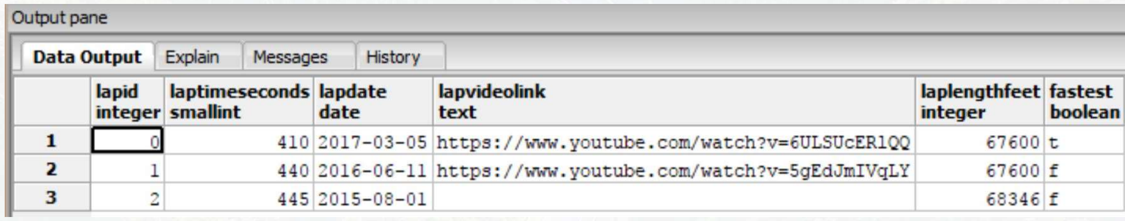
```
CREATE OR REPLACE FUNCTION fastestLap() RETURNS TRIGGER AS
$$
BEGIN
IF NEW.LapTimeSeconds <=
        (SELECT LapTimeSeconds
        FROM Lap
        ORDER BY LapTimeSeconds ASC
        Limit 1)
THEN
        UPDATE Lap
        SET Fastest = TRUE
        WHERE Lap.lapid=New.lapid;Before:

        END IF;
        RETURN NEW;
        END;
$$LANGUAGE plpgsql;
```

Trigger:

```
CREATE TRIGGER fastestLap
After INSERT ON Lap
FOR EACH ROW
EXECUTE PROCEDURE fastestLap();
```

Before:

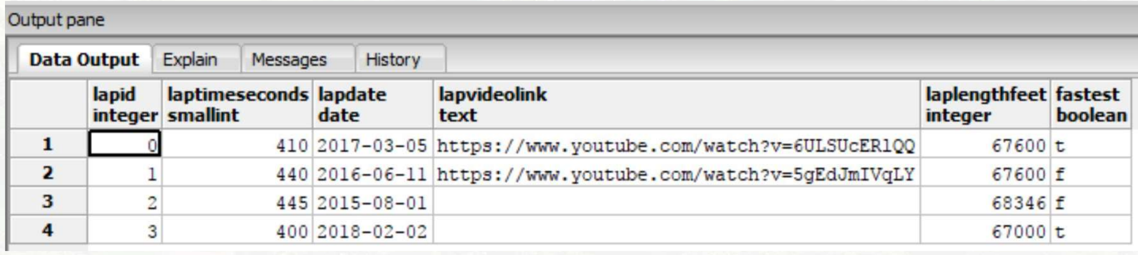


Output pane

Data Output | Explain | Messages | History

	lapid integer	laptimeseconds smallint	lapdate date	lapvideolink text	laplengthfeet integer	fastest boolean
1	0	410	2017-03-05	https://www.youtube.com/watch?v=6ULSUcER1QQ	67600	t
2	1	440	2016-06-11	https://www.youtube.com/watch?v=5gEdJmIVqLY	67600	f
3	2	445	2015-08-01		68346	f

After:



Output pane

Data Output | Explain | Messages | History

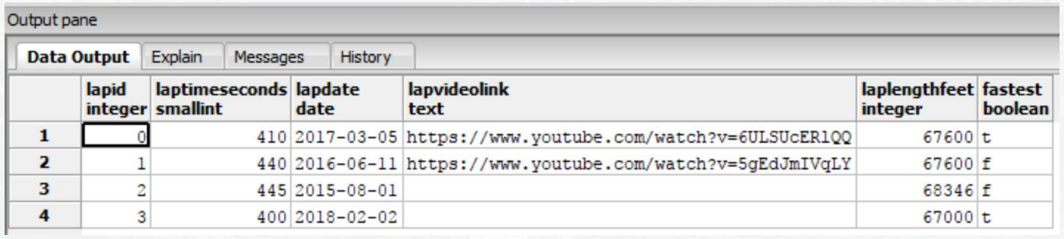
	lapid integer	laptimeseconds smallint	lapdate date	lapvideolink text	laplengthfeet integer	fastest boolean
1	0	410	2017-03-05	https://www.youtube.com/watch?v=6ULSUcER1QQ	67600	t
2	1	440	2016-06-11	https://www.youtube.com/watch?v=5gEdJmIVqLY	67600	f
3	2	445	2015-08-01		68346	f
4	3	400	2018-02-02		67000	t

Trigger: lastFastestLap

The lastFastestLap trigger assigns checks to see if each new lap recorded is the fastest. If a lap is the second fastest, it's 'Fastest' attribute will be assigned false from true.

Function:

```
CREATE OR REPLACE FUNCTION lastFastestLap() RETURNS TRIGGER AS
$$
BEGIN
IF (Select Fastest
        From Lap
        Order by LapTimeSeconds ASC
        Limit 1) = True
        AND
        (Select Fastest
        From Lap
        Order by LapTimeSeconds ASC
        Offset 1
        Limit 1) = True
THEN
        UPDATE Lap
        SET Fastest = False
        WHERE Lap.lapid=(Select Lapid
                                From Lap
                                Order by LapTimeSeconds ASC
                                Offset 1
                                Limit 1);
        END IF;
        RETURN NEW;
        END;
$$LANGUAGE plpgsql;
```



Output pane

Data Output | Explain | Messages | History

	lapid integer	laptimeseconds smallint	lapdate date	lapvideolink text	laplengthfeet integer	fastest boolean
1	0	410	2017-03-05	https://www.youtube.com/watch?v=6ULSUcER1QQ	67600	t
2	1	440	2016-06-11	https://www.youtube.com/watch?v=5gEdJmIVqLY	67600	f
3	2	445	2015-08-01		68346	f
4	3	400	2018-02-02		67000	t

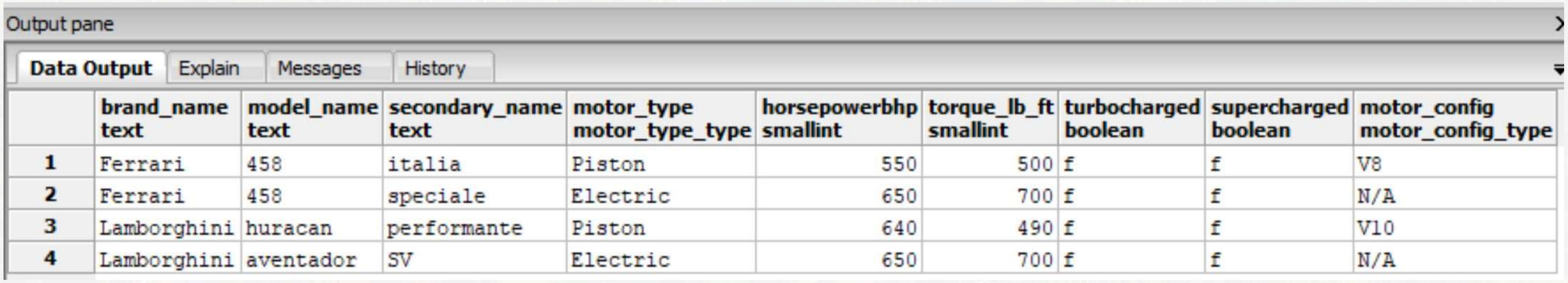
Trigger:

```
CREATE TRIGGER lastFastestLap
After INSERT ON Lap
FOR EACH ROW
EXECUTE PROCEDURE lastFastestLap();
```

Sample Reports One

This query reports the engine information of a car in the database. The engine is the heart of the car, and is one of the significant determinants in a lap time.

```
SELECT Brand_Name, Model_Name, Secondary_Name, Motor_Type, HorsepowerBHP,
Torque_lb_ft,
          Turbocharged, Supercharged, Motor_Config
          FROM Brand, Car, Motor
          WHERE Brand.Brandid=Car.Brandid AND Motor.Motorid=Car.Motorid
```

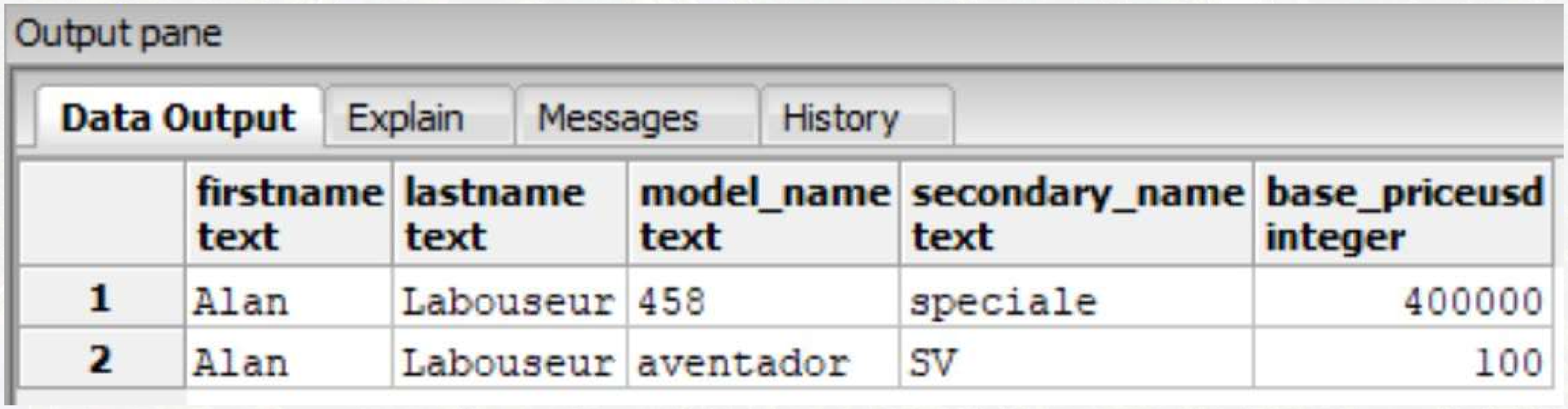


	brand_name text	model_name text	secondary_name text	motor_type motor_type_type	horsepowerbhp smallint	torque_lb_ft smallint	turbocharged boolean	supercharged boolean	motor_config motor_config_type
1	Ferrari	458	italia	Piston	550	500	f	f	V8
2	Ferrari	458	speciale	Electric	650	700	f	f	N/A
3	Lamborghini	huracan	performante	Piston	640	490	f	f	V10
4	Lamborghini	aventador	SV	Electric	650	700	f	f	N/A

Sample Reports Two

This query reports the record making cars owned and tested by an independent owner.

```
Select Independent_Owner.FirstName, Independent_Owner.LastName, Model_Name,
Secondary_Name, Base_PriceUSD
        FROM Independent_Owner, Car, LapRecord, Test_Org, Independent_Test_Org
        WHERE Car.Carid=LapRecord.Carid
            AND LapRecord.TestOrgid=Test_Org.TestOrgid
            AND Test_Org.TestOrgid=Independent_Test_Org.TestOrgid
            AND Independent_Test_Org.TestOrgid=Independent_Owner.TestOrgid;
```

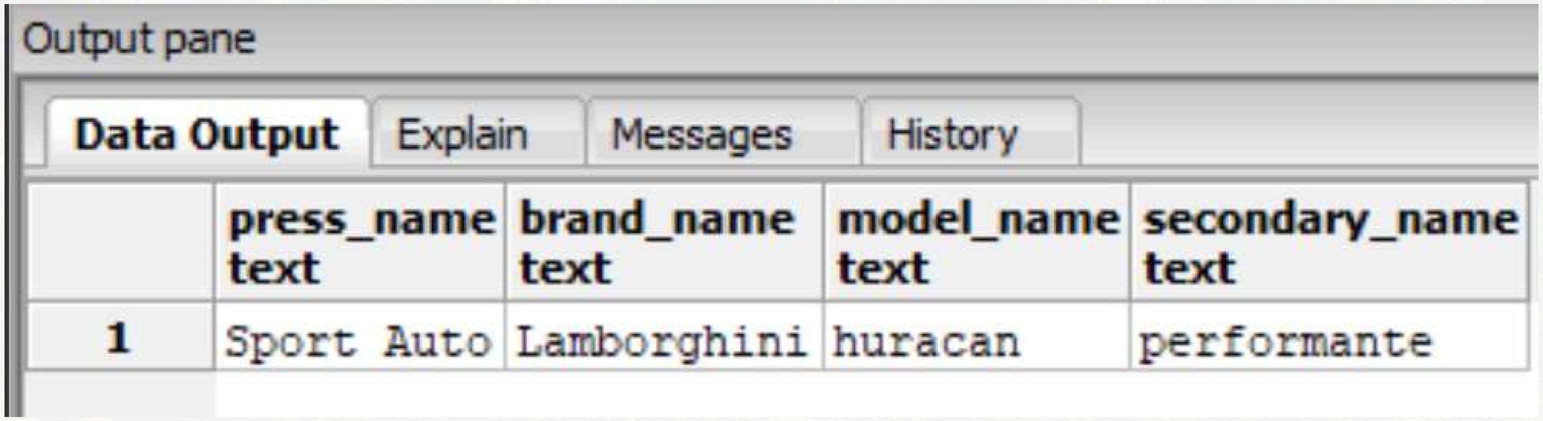


	firstname text	lastname text	model_name text	secondary_name text	base_priceusd integer
1	Alan	Labouseur	458	speciale	400000
2	Alan	Labouseur	aventador	SV	100

Sample Reports Three

This query reports the each car tested by an automotive press organization, and the press organization which tested it.

```
Select Press_Name, Brand_Name, Model_Name,
        Secondary_Name
        FROM Brand, Car, LapRecord, Test_Org, Independent_Test_Org, Automotive_Press
        WHERE Car.Brandid=Brand.Brandid
        AND Car.Carid=LapRecord.Carid
        AND LapRecord.TestOrgid=Test_Org.TestOrgid
        AND TEST_ORG.TestOrgid=Independent_Test_Org.TestOrgid
        AND Independent_Test_Org.TestOrgid= Automotive_Press.TestOrgid;
```



	press_name text	brand_name text	model_name text	secondary_name text
1	Sport Auto	Lamborghini	huracan	performante

Security: Users and Groups

In this database, the database admin, or DBA, has the exclusive rights to make edits to the database, and can make any changes to the database that they need to.

```
CREATE ROLE DBA;

GRANT ALL ON ALL TABLES IN SCHEMA PUBLIC TO DBA;
```

The second role is for a Viewer, which has the rights to view all of the contents of the database, but is not given privileges for any changes.

```
CREATE ROLE Viewer;

REVOKE ALL ON ALL TABLES IN SCHEMA PUBLIC FROM Viewer;

GRANT SELECT ON ALL TABLES IN SCHEMA PUBLIC TO Viewer;
```

Implementation Notes:

- This implementation of a Nurburgring records database records a lot of extra data not directly related to the lap record, such as how much torque the car’s motor has, or who tested the car. This extra information allows a user to filter laps based on these pieces of data, allowing them to glean insights and information from the database.
- What classifies as a record is not specified. If a van went around the Nurburgring in a fast enough time, the DBA could decide to add it to the database.

Known Issues:

- In the current version of this database, a driver can also be a private owner. A driver does not have the same connection to a press org or parent org, even though they may be employed by and drive for that test organization.
- The lastfastestlap trigger needs tweaking to function as intended.

Future Enhancements:

- Add aforementioned driver org relationships
- The Tire table could be improved with more columns for tire brand and names, and width.
- The LapLengthFeet field should be changed to a new type which contains the multiple (67,600ft, 68,346ft) standardized lap lengths, rather than an int field.