

Cognizant Technology Solutions



A Project Report on

“FORMULA 1 PROJECT USING PYSPARK”

Prepared By

Ankita Shivane	2320390
Harshit Yadav	2320382
Mimansa Yadav	2320956
Nandhini T	2320539
Rajanya Kamilya	2320828
Shambhavi Shukla	2320529

FORMULA RACING

Formula racing, exemplified by Formula 1 (F1), stands as the pinnacle of motorsport, blending cutting-edge technology with unparalleled athleticism and strategy. Originating in the post-war era, Formula racing has evolved into a global phenomenon, captivating audiences with its fusion of speed, precision engineering, and human skill.

At its heart, Formula racing features a series of high-profile races held on circuits around the world, with the Formula 1 World Championship being the most prestigious. This championship comprises a calendar of races held across diverse tracks, from iconic street circuits like Monaco to purpose-built facilities like Silverstone and Suzuka.

The sport is propelled by a roster of elite drivers, each pushing the limits of human performance behind the wheel. From legendary figures like Michael Schumacher and Ayrton Senna to modern-day stars like Lewis Hamilton and Max Verstappen, these drivers command respect and admiration for their skill, bravery, and determination.

Behind the scenes, Formula racing is a relentless pursuit of perfection for constructors, who design and build the cars that compete on the track. Teams like Mercedes, Ferrari, and Red Bull invest heavily in research, development, and innovation, striving to create the fastest and most reliable machines possible.

Formula racing is also a stage for remarkable achievements, with records continually being shattered and history being made. From historic race wins to championship triumphs, the sport is a tapestry of unforgettable moments that etch themselves into the annals of motorsport history.

In essence, Formula racing is a captivating blend of technology, athleticism, and strategy, where races, drivers, constructors, tracks, and achievements converge to create an electrifying spectacle that captivates audiences around the globe.

PYSPARK

PySpark is a Python API to support Python with Apache Spark. PySpark provides Py4j library, with the help of this library, Python can be easily integrated with Apache Spark. PySpark plays an essential role when it needs to work with a vast dataset or analyze them. This feature of PySpark makes it a very demanding tool among data engineers.



A large amount of data is generated offline and online. It is necessary to extract valuable information from the raw data. We require a more efficient tool to perform different types of operations on the big data. There are various tools to perform the multiple tasks on the huge dataset, but these tools are not so appealing anymore. It is needed some scalable and flexible tools to crack big data and gain benefit from it.

BUSINESS REQUIREMENTS

Description:

Analyze and process Formula 1 data files to extract insights about dominant drivers, teams and their performances.

REQUIREMENTS:

Data Ingestion Requirements:

- Ingest the data from various source files
- Ingested data must have the schema applied
- Ingested data must have audit columns
- Ingested data must be stored in a columnar format

Data Transformation Requirements:

- Join the key information required for reporting to create a new table.
- Join the key information required for Analysis to create a new table.
- Transformed tables must have audit columns
- Must be able to analyze the transformed data via SQL
- Transformed data must be stored in columnar format (i.e. Parquet)

Reporting Requirements:

- Driver Standings
- Constructor Standings

Analysis Requirements:

- Dominant Drivers
- Dominant Teams
- Visualize the Outputs

F1 DATA SET

circuits table			
Field	Type	Null	Key
circuitId	int(11)	NO	PRI
circuitRef	varchar(255)	NO	
name	varchar(255)	NO	
location	varchar(255)	YES	
country	varchar(255)	YES	
lat	float	YES	
lng	float	YES	
alt	int(11)	YES	
url	varchar(255)	NO	UNI

constructors table			
Field	Type	Null	Key
constructorId	int(11)	NO	PRI
constructorRef	varchar(255)	NO	
name	varchar(255)	NO	UNI
nationality	varchar(255)	YES	
url	varchar(255)	NO	

lap_times table			
Field	Type	Null	Key
raceId	int(11)	NO	FK
driverId	int(11)	NO	FK
lap	int(11)	NO	PRI
position	int(11)	YES	
time	varchar(255)	YES	
milliseconds	int(11)	YES	

pit_stops table			
Field	Type	Null	Key
raceId	int(11)	NO	FK
driverId	int(11)	NO	FK
stop	int(11)	NO	PRI
lap	int(11)	NO	
time	time	NO	
duration	varchar(255)	YES	
milliseconds	int(11)	YES	

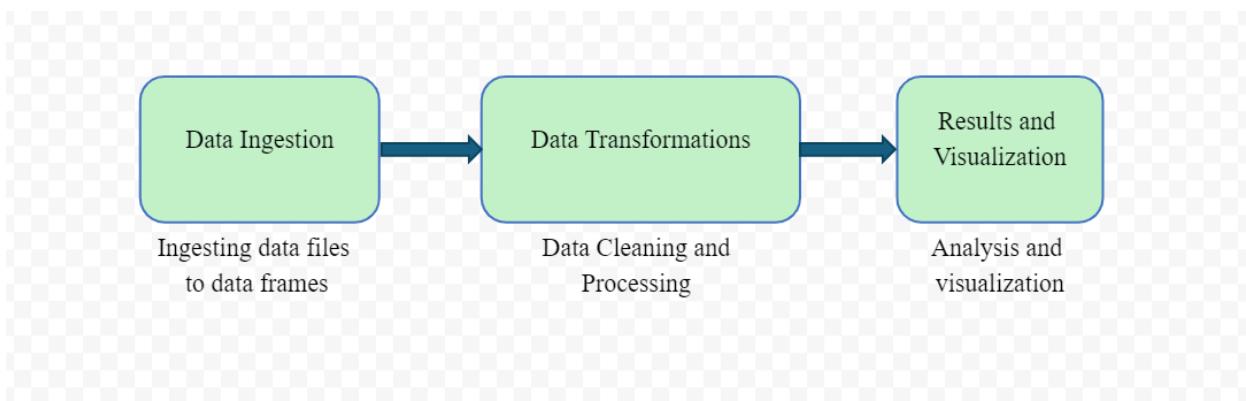
qualifying table			
Field	Type	Null	Key
qualifyId	int(11)	NO	PRI
raceId	int(11)	NO	
driverId	int(11)	NO	
constructorId	int(11)	NO	
number	int(11)	NO	
position	int(11)	YES	
q1	varchar(255)	YES	
q2	varchar(255)	YES	
q3	varchar(255)	YES	

drivers table			
Field	Type	Null	Key
driverId	int(11)	NO	PRI
driverRef	varchar(255)	NO	
number	int(11)	YES	
code	varchar(3)	YES	
forename	varchar(255)	NO	
surname	varchar(255)	NO	
dob	date	YES	
nationality	varchar(255)	YES	
url	varchar(255)	NO	UNI

races table			
Field	Type	Null	Key
raceId	int(11)	NO	PRI
year	int(11)	NO	
round	int(11)	NO	
circuitId	int(11)	NO	
name	varchar(255)	NO	
date	date	NO	
time	time	YES	
url	varchar(255)	YES	UNI
fp1_date	date	YES	
fp1_time	time	YES	
fp2_date	date	YES	
fp2_time	time	YES	
fp3_date	date	YES	
fp3_time	time	YES	
quali_date	date	YES	
quali_time	time	YES	
sprint_date	date	YES	
sprint_time	time	YES	

sprint_results table			
Field	Type	Null	Key
sprintResultId	int(11)	NO	PRI
raceId	int(11)	NO	
driverId	int(11)	NO	
constructorId	int(11)	NO	
number	int(11)	YES	
grid	int(11)	NO	
position	int(11)	YES	
positionText	varchar(255)	NO	
positionOrder	int(11)	NO	
points	float	NO	
laps	int(11)	NO	
time	varchar(255)	YES	
milliseconds	int(11)	YES	
fastestLap	int(11)	YES	
fastestLapTime	varchar(255)	YES	
statusId	int(11)	NO	

PICTORIAL FLOWCHART



IMPLEMENTATION:

- **REQUIREMENT 1:**

DATA INGESTION

Data ingestion is the process of importing and loading data into a system. It's a critical step in any data-centric workflow, ensuring that the correct information is available at the right time for analysis and decision-making.

To ingest a dataset into Databricks, you can follow these steps:

1. Create a Compute engine:

- Start by creating a Databricks cluster.
- This cluster will provide the compute resources needed to run your commands.

2. Enable DBFS in Databricks:

- Use Databricks features to explore your raw dataset.

3. Ingest the Raw Data:

- Load the raw data into a table to make it available for further processing.

Here, in this project we have 8 datafiles that we need to ingest into dbfs system.

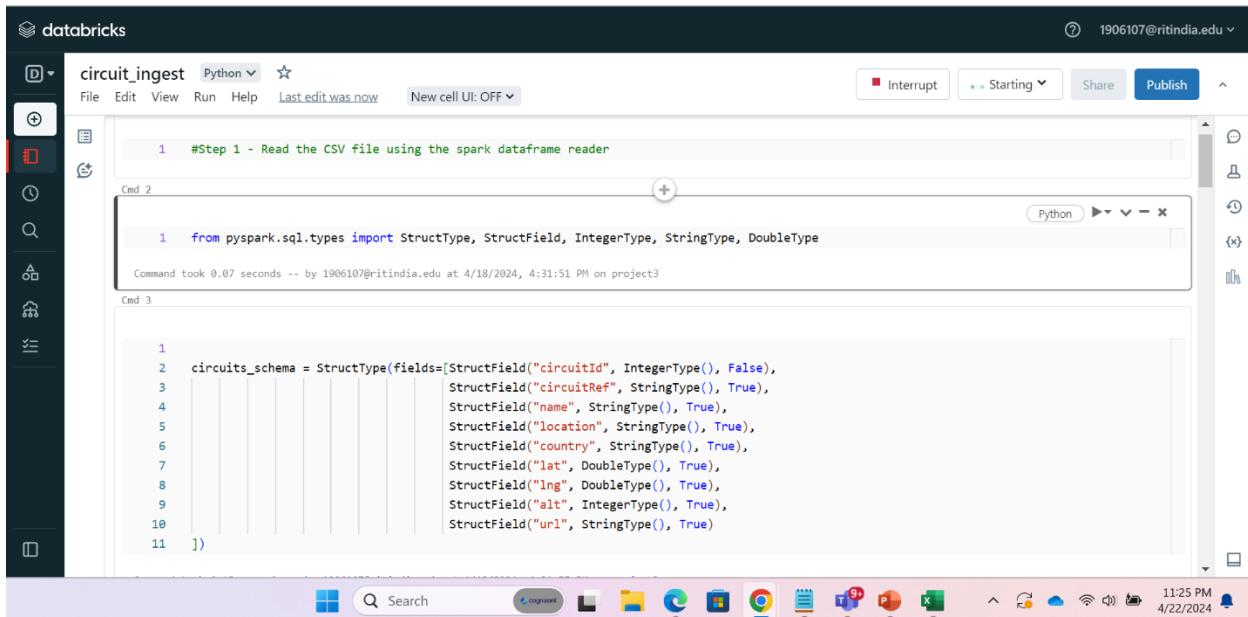
Circuit file ingestion:

Step 1: Open Databricks community and login to your account.

Create a compute engine for analysis.

Create new notebook in workspace and write the above syntax

Step 2: Create the schema for Circuit file using StructType

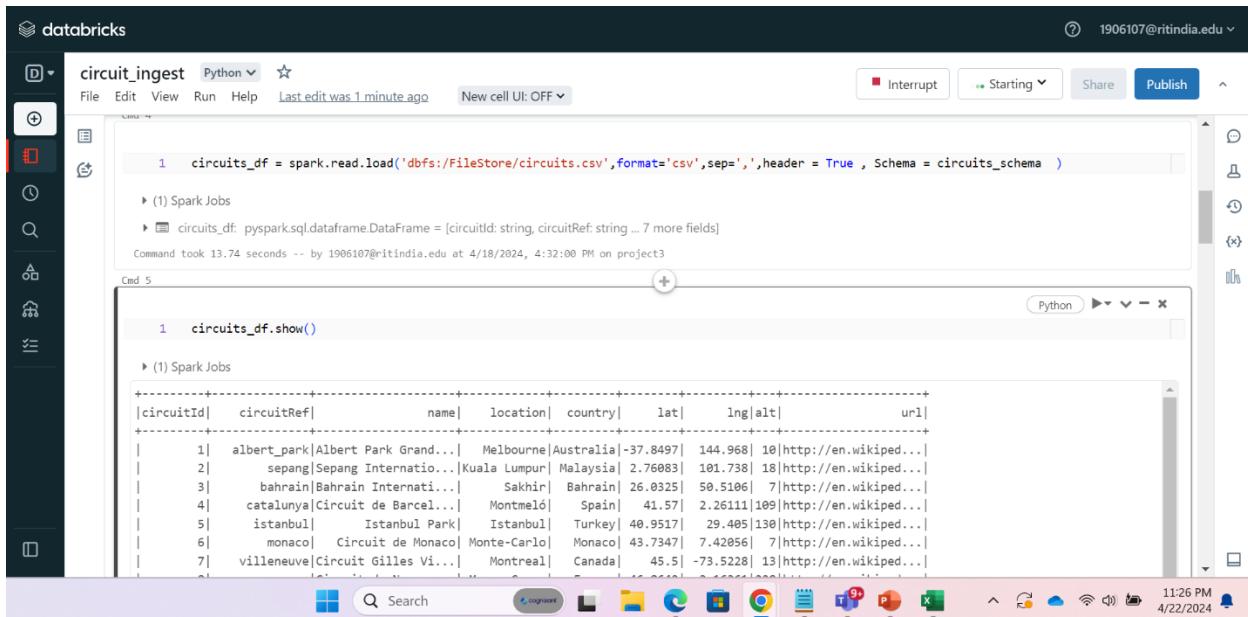


```

1 #Step 1 - Read the CSV file using the spark dataframe reader
2
3 from pyspark.sql.types import StructType, StructField, IntegerType, StringType, DoubleType
4
5 circuits_schema = StructType(fields=[StructField("circuitId", IntegerType(), False),
6                                     StructField("circuitRef", StringType(), True),
7                                     StructField("name", StringType(), True),
8                                     StructField("location", StringType(), True),
9                                     StructField("country", StringType(), True),
10                                    StructField("lat", DoubleType(), True),
11                                    StructField("lng", DoubleType(), True),
12                                    StructField("alt", IntegerType(), True),
13                                    StructField("url", StringType(), True)])

```

Step 3: We have created a dataframe by reading the data from circuit csv file loaded in DBFS system



```

1 circuits_df = spark.read.load('dbfs:/FileStore/circuits.csv',format='csv',sep=',',header = True , Schema = circuits_schema )
2
3 circuits_df.show()

```

circuitId	circuitRef	name	location	country	lat	lng	alt	url
1	albert_park	Albert Park Grand...	Melbourne	Australia	-37.8497	144.968	10	[http://en.wikipedia.org/w/index.php?title=Albert_Park_Grand_Prix&oldid=10173818]
2	sepang	Sepang Internatio...	Kuala Lumpur	Malaysia	2.76083	101.738	18	[http://en.wikipedia.org/w/index.php?title=Sepang_International_Racing_Circuit&oldid=10173818]
3	bahrain	Bahrain Internati...	Sakhir	Bahrain	26.0325	50.5106	7	[http://en.wikipedia.org/w/index.php?title=Bahrain_International_Circuit&oldid=10173818]
4	catalunya	Circuit de Barcel...	Montmeló	Spain	41.57	2.26111	109	[http://en.wikipedia.org/w/index.php?title=Circuit_de_Montmelo&oldid=10173818]
5	istanbul	Istanbul Park	Istanbul	Turkey	40.9517	29.405	130	[http://en.wikipedia.org/w/index.php?title=Istanbul_Park&oldid=10173818]
6	monaco	Circuit de Monaco	Monte-Carlo	Monaco	43.7347	7.42056	7	[http://en.wikipedia.org/w/index.php?title=Circuit_de_Monaco&oldid=10173818]
7	villeneuve	Circuit Gilles Vi...	Montreal	Canada	45.5	-73.5228	13	[http://en.wikipedia.org/w/index.php?title=Circuit_Gilles_Villeneuve&oldid=10173818]

Step 4: Selecting only the required columns and creating a new dataframe.

The screenshot shows a Databricks notebook interface with the title "circuit_ingest". The code in the notebook is as follows:

```
1 #Step 2 - Select only the required columns
2
3 from pyspark.sql.functions import col
4
5 Command took 0.06 seconds -- by 1906107@ritindia.edu at 4/18/2024, 4:32:30 PM on project3
6
7
8 1 circuits_selected_df = circuits_df.select(col("circuitId"), col("circuitRef"), col("name"), col("location"), col("country"), col("lat"), col("lng"), col("alt"))
9
10 Command took 0.12 seconds -- by 1906107@ritindia.edu at 4/18/2024, 4:32:40 PM on project3
11
12
13 1 #Step 3 - Rename the columns as required
14
15 Command took 0.01 seconds -- by 1906107@ritindia.edu at 4/18/2024, 4:32:41 PM on project3
16
17
```

The notebook is running on a cluster named "My Cluster". The status bar at the bottom right shows the date and time as 4/22/2024 at 11:27 PM.

Step 5: Renaming the columns using `withColumnRenamed` command to match the naming standards.

The screenshot shows a Databricks notebook interface with the title "circuit_ingest". The code in the notebook is as follows:

```
1 #Step 3 - Rename the columns as required
2
3 circuits_renamed_df = circuits_selected_df.withColumnRenamed("circuitId", "circuit_id") \
4   .withColumnRenamed("circuitRef", "circuit_ref") \
5   .withColumnRenamed("lat", "latitude") \
6   .withColumnRenamed("lng", "longitude") \
7   .withColumnRenamed("alt", "altitude")
8
9 Command took 0.22 seconds -- by 1906107@ritindia.edu at 4/18/2024, 4:32:53 PM on project3
10
11 1 #Step 4 - Add ingestion date to the dataframe
12
13 Command took 0.01 seconds -- by 1906107@ritindia.edu at 4/18/2024, 4:32:54 PM on project3
14
15
```

The notebook is running on a cluster named "My Cluster". The status bar at the bottom right shows the date and time as 4/22/2024 at 11:27 PM.

Step 6: Adding the ingestion_date column with current_timestamp and displaying the dataframe

The screenshot shows a Databricks notebook interface with the following details:

- Title:** circuit_ingest
- Languages:** Python
- Last edit:** 5 minutes ago
- New cell UI:** OFF
- Cells:**
 - Cmd 12:** `from pyspark.sql.functions import current_timestamp`
 - Cmd 13:** `circuits_final_df = circuits_renamed_df.withColumn("ingestion_date", current_timestamp())`
 - Cmd 14:** `circuits_final_df.show(truncate=False)`
- Output:** Shows the output of the last command, indicating 1 Spark Job.
- Environment:** My Cluster
- Timestamp:** 11:29 PM 4/22/2024

The screenshot shows the output of the `circuits_final_df.show(truncate=False)` command in the Databricks notebook. The resulting DataFrame contains the following columns and data:

circuit_id	circuit_ref	name	location	country	latitude	longitude	altitude	ingestion_date
1	albert_park	Albert Park Grand Prix Circuit	Melbourne	Australia	-37.8497	144.968	10	2024-04-22 17:59:07.918
2	sepang	Sepang International Circuit	Kuala Lumpur	Malaysia	2.76083	101.738	18	2024-04-22 17:59:07.918
3	bahrain	Bahrain International Circuit	Sakhir	Bahrain	26.0325	50.5106	7	2024-04-22 17:59:07.918
4	catalunya	Circuit de Barcelona-Catalunya	Montmeló	Spain	41.57	2.26111	109	2024-04-22 17:59:07.918
5	istanbul	Istanbul Park	Istanbul	Turkey	40.9517	29.405	130	2024-04-22 17:59:07.918
6	monaco	Circuit of Monaco	Monte-Carlo	Monaco	43.7347	7.42056	7	2024-04-22 17:59:07.918
7	villeneuve	Circuit Gilles Villeneuve	Montreal	Canada	45.5	-73.5228	13	2024-04-22 17:59:07.918
8	magny_cours	Circuit de Nevers Magny-Cours	Magny Cours	France	46.8642	3.16361	228	2024-04-22 17:59:07.918
9	silverstone	Silverstone Circuit	Silverstone	UK	52.078	-1.01694	153	2024-04-22 17:59:07.918
10	hockenheimring	Hockenheimring	Hockenheim	Germany	49.3278	8.56583	103	2024-04-22 17:59:07.918
11	hungaroring	Hungaroring	Budapest	Hungary	47.5789	19.2486	264	2024-04-22 17:59:07.918
12	valencia	Valencia Street Circuit	Valencia	Spain	39.4589	-0.3316674	14	2024-04-22 17:59:07.918
13	spa	Circuit de Spa-Francorchamps	Spa	Belgium	50.4372	5.97139	401	2024-04-22 17:59:07.918
14	monza	Autodromo Nazionale di Monza	Monza	Italy	45.6156	9.28111	162	2024-04-22 17:59:07.918
15	marina_bay	Marina Bay Street Circuit	Marina Bay	Singapore	1.2914	103.864	18	2024-04-22 17:59:07.918
16	fujii	Fuji Speedway	Oyama	Japan	35.3717	138.927	583	2024-04-22 17:59:07.918
17	shanghai	Shanghai International Circuit	Shanghai	China	31.3389	121.22	5	2024-04-22 17:59:07.918
18	interlagos	Autódromo José Carlos Pace	São Paulo	Brazil	-23.70361	-46.6997	785	2024-04-22 17:59:07.918

Timestamp: 11:30 PM 4/22/2024

Step 7: Creating new directory in databricks filestore and storing our dataframe in columnar format that is parquet file.

```

1 dbutils.fs.mkdirs("dbfs:/FileStore/Formula1/processed")
Out[16]: True
Command took 0.74 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:28:47 PM on My Cluster

1 circuits_final_df.write.parquet("FileStore/Formula1/processed/circuits")
AnalysisException: Path dbfs:/FileStore/Formula1/processed/circuits already exists.
Command took 3.13 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:28:47 PM on My Cluster

1 %fs ls "FileStore/Formula1/processed"
Command skipped
Command took 0.00 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:28:47 PM on My Cluster

```

Constructor file ingestion:

Step 1: Create the schema for Constructor file using String format and then read the file from constructor json file.

```

#Step 1 - Read the JSON file using the spark dataframe reader
Command took 0.08 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:32:14 PM on My Cluster

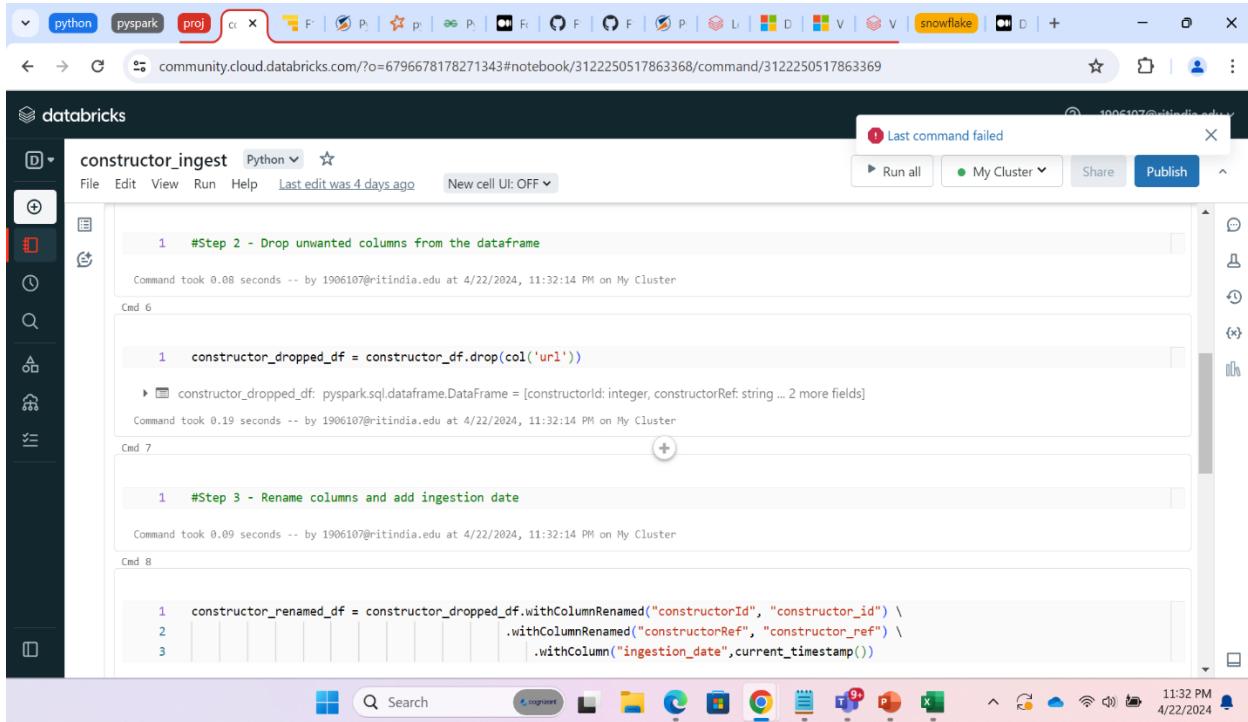
from pyspark.sql.functions import col , current_timestamp
Command took 0.09 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:32:14 PM on My Cluster

constructors_schema = "constructorId INT, constructorRef STRING, name STRING, nationality STRING, url STRING"
2
Command took 0.09 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:32:14 PM on My Cluster

constructor_df = spark.read.json("dbfs:/FileStore/constructors.json", schema=constructors_schema)
constructor_df: pyspark.sql.dataframe.DataFrame = [constructorId: integer, constructorRef: string ... 3 more fields]

```

Step2: We are dropping the url column by drop command.



Databricks Notebook titled "constructor_ingest" (Python) showing the following code and output:

```

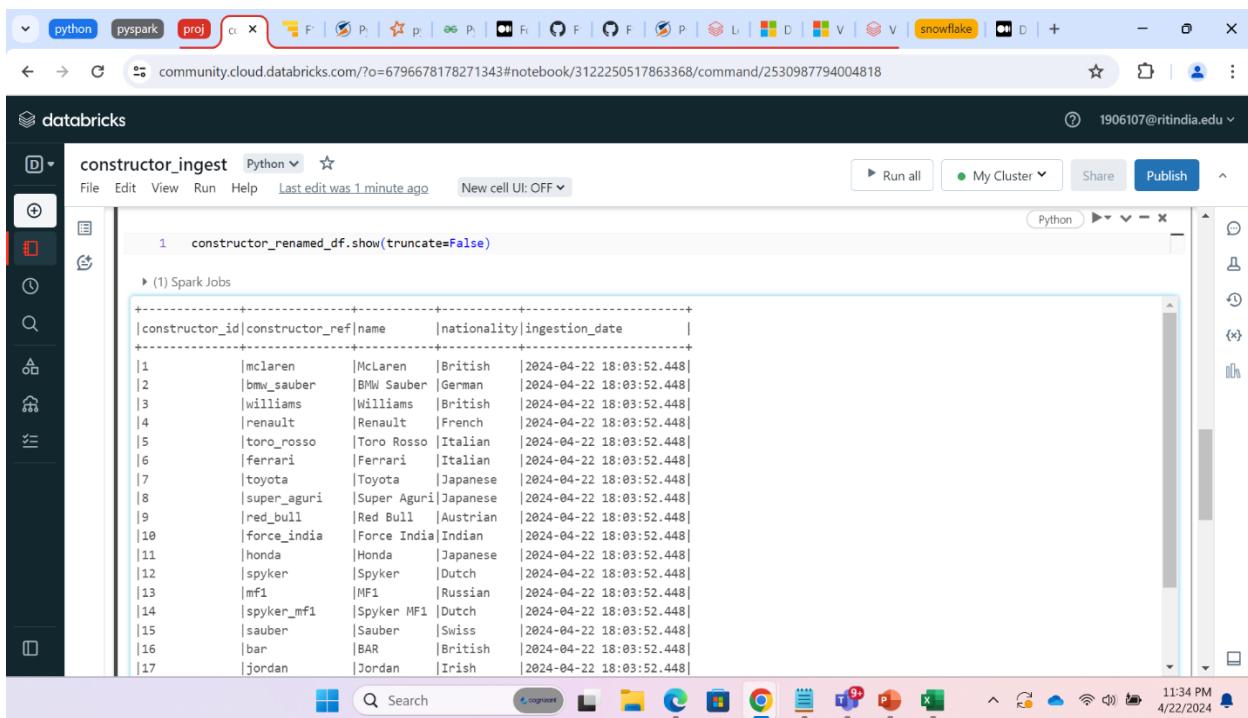
1 #Step 2 - Drop unwanted columns from the dataframe
Command took 0.00 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:32:14 PM on My Cluster

Cmd 6
1 constructor_dropped_df = constructor_df.drop(col('url'))
Command took 0.19 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:32:14 PM on My Cluster

Cmd 7
1 #Step 3 - Rename columns and add ingestion date
Command took 0.09 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:32:14 PM on My Cluster

Cmd 8
1 constructor_renamed_df = constructor_dropped_df.withColumnRenamed("constructorId", "constructor_id") \
2 .withColumnRenamed("constructorRef", "constructor_ref") \
3 .withColumn("ingestion_date", current_timestamp())

```



Databricks Notebook titled "constructor_ingest" (Python) showing the output of the previous command:

```

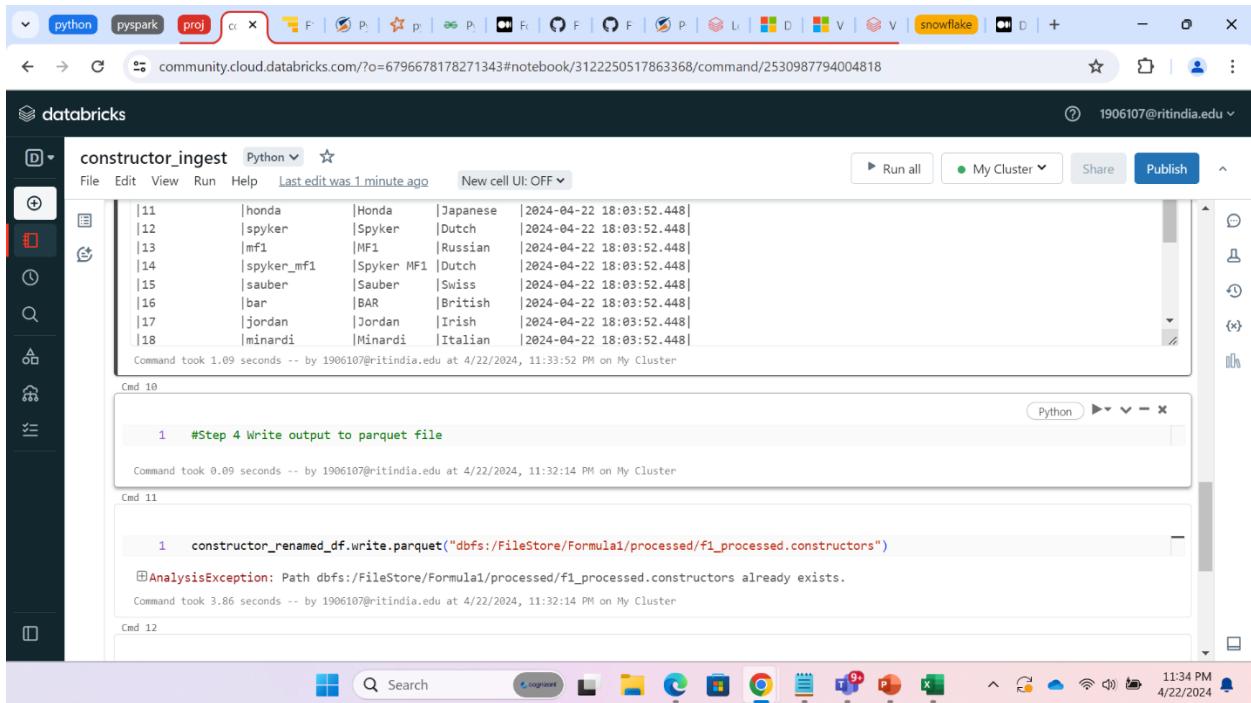
1 constructor_renamed_df.show(truncate=False)

```

(1) Spark Jobs

constructor_id	constructor_ref	name	nationality	ingestion_date
1	mcclaren	McLaren	British	2024-04-22 18:03:52,448
2	bmw_sauber	BMW Sauber	German	2024-04-22 18:03:52,448
3	williams	Williams	British	2024-04-22 18:03:52,448
4	renault	Renault	French	2024-04-22 18:03:52,448
5	toro_rosso	Toro Rosso	Italian	2024-04-22 18:03:52,448
6	ferrari	Ferrari	Italian	2024-04-22 18:03:52,448
7	toyota	Toyota	Japanese	2024-04-22 18:03:52,448
8	super_aguri	Super Aguri	Japanese	2024-04-22 18:03:52,448
9	red_bull	Red Bull	Austrian	2024-04-22 18:03:52,448
10	force_india	Force India	Indian	2024-04-22 18:03:52,448
11	honda	Honda	Japanese	2024-04-22 18:03:52,448
12	spyker	Spyker	Dutch	2024-04-22 18:03:52,448
13	mf1	MF1	Russian	2024-04-22 18:03:52,448
14	spyker_mf1	Spyker MF1	Dutch	2024-04-22 18:03:52,448
15	sauber	Sauber	Swiss	2024-04-22 18:03:52,448
16	bar	BAR	British	2024-04-22 18:03:52,448
17	jordan	Jordan	Irish	2024-04-22 18:03:52,448

Step3: Storing our dataframe in columnar format that is parquet file.



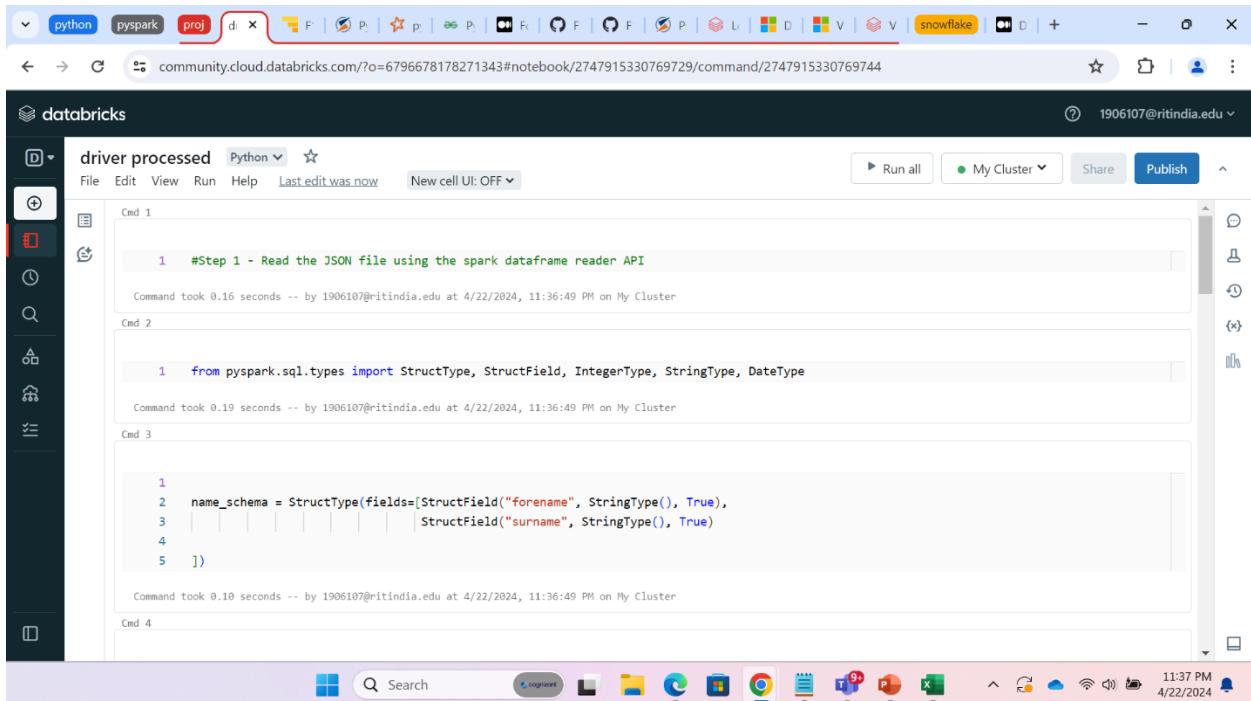
```
11 |honda    |Honda     |Japanese  |2024-04-22 18:03:52,448|
12 |spyker   |Spyker    |Dutch     |2024-04-22 18:03:52,448|
13 |mf1      |MF1       |Russian   |2024-04-22 18:03:52,448|
14 |spyker_mf1|Spyker MF1|Dutch     |2024-04-22 18:03:52,448|
15 |sauber   |Sauber    |Swiss     |2024-04-22 18:03:52,448|
16 |bar       |BAR        |British   |2024-04-22 18:03:52,448|
17 |jordan   |Jordan    |Irish     |2024-04-22 18:03:52,448|
18 |minardi  |Minardi   |Italian   |2024-04-22 18:03:52,448|
Command took 1.09 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:33:52 PM on My Cluster
```

```
1 #Step 4 Write output to parquet file
Command took 0.09 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:32:14 PM on My Cluster
```

```
1 constructor_renamed_df.write.parquet("dbfs:/FileStore/Formula1/processed/f1_processed.constructors")
AnalysisException: Path dbfs:/FileStore/Formula1/processed/f1_processed.constructors already exists.
Command took 3.86 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:32:14 PM on My Cluster
```

Driver file ingestion:

Step 1: Step 2: Create the schema this file using StructType



```
1 #Step 1 - Read the JSON file using the spark dataframe reader API
Command took 0.16 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:36:49 PM on My Cluster
```

```
1 from pyspark.sql.types import StructType, StructField, IntegerType, StringType, DateType
Command took 0.19 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:36:49 PM on My Cluster
```

```
1
2 name_schema = StructType(fields=[StructField("forename", StringType(), True),
3                                  StructField("surname", StringType(), True),
4
5 ])
Command took 0.10 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:36:49 PM on My Cluster
```

Step 3: We have created a dataframe by reading the data from json file loaded in DBFS system

```

1 drivers_schema = StructType(fields=[StructField("driverId", IntegerType(), False),
2                                     StructField("driverRef", StringType(), True),
3                                     StructField("number", IntegerType(), True),
4                                     StructField("code", StringType(), True),
5                                     StructField("name", name_schema),
6                                     StructField("dob", DateType(), True),
7                                     StructField("nationality", StringType(), True),
8                                     StructField("url", StringType(), True)
9                                     ])
10 ])
```

Command took 0.09 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:36:49 PM on My Cluster

```

1 drivers_df = spark.read.json("dbfs:/FileStore/drivers.json", schema = drivers_schema)
```

drivers_df: pyspark.sql.DataFrame = [driverId: integer, driverRef: string ... 6 more fields]

Command took 0.69 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:36:49 PM on My Cluster

Step 4: Renaming the columns using `withColumnRenamed` command to match the naming standards.

```

1 #Step 2 - Rename columns and add new columns
2 #driverId renamed to driver_id
3 #driverRef renamed to driver_ref
4 #ingestion date added
5 #name added with concatenation of forename and surname
```

Command took 0.08 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:36:49 PM on My Cluster

```

1 from pyspark.sql.functions import col, concat, lit, current_timestamp
```

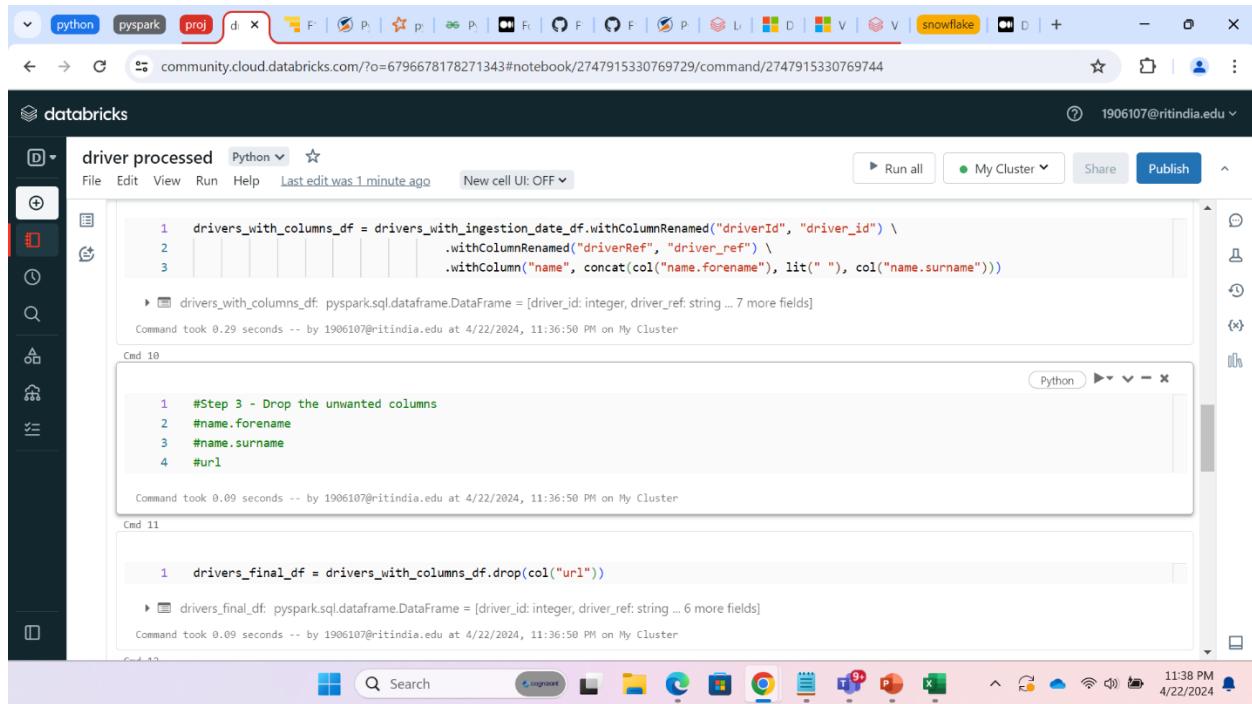
Command took 0.09 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:36:49 PM on My Cluster

```

1 drivers_with_ingestion_date_df = drivers_df.withColumn("ingestion_date", current_timestamp())
```

drivers_with_ingestion_date_df: pyspark.sql.DataFrame = [driverId: integer, driverRef: string ... 7 more fields]

Command took 0.09 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:36:50 PM on My Cluster



The screenshot shows a Databricks notebook titled "driver processed" in Python. It contains three cells:

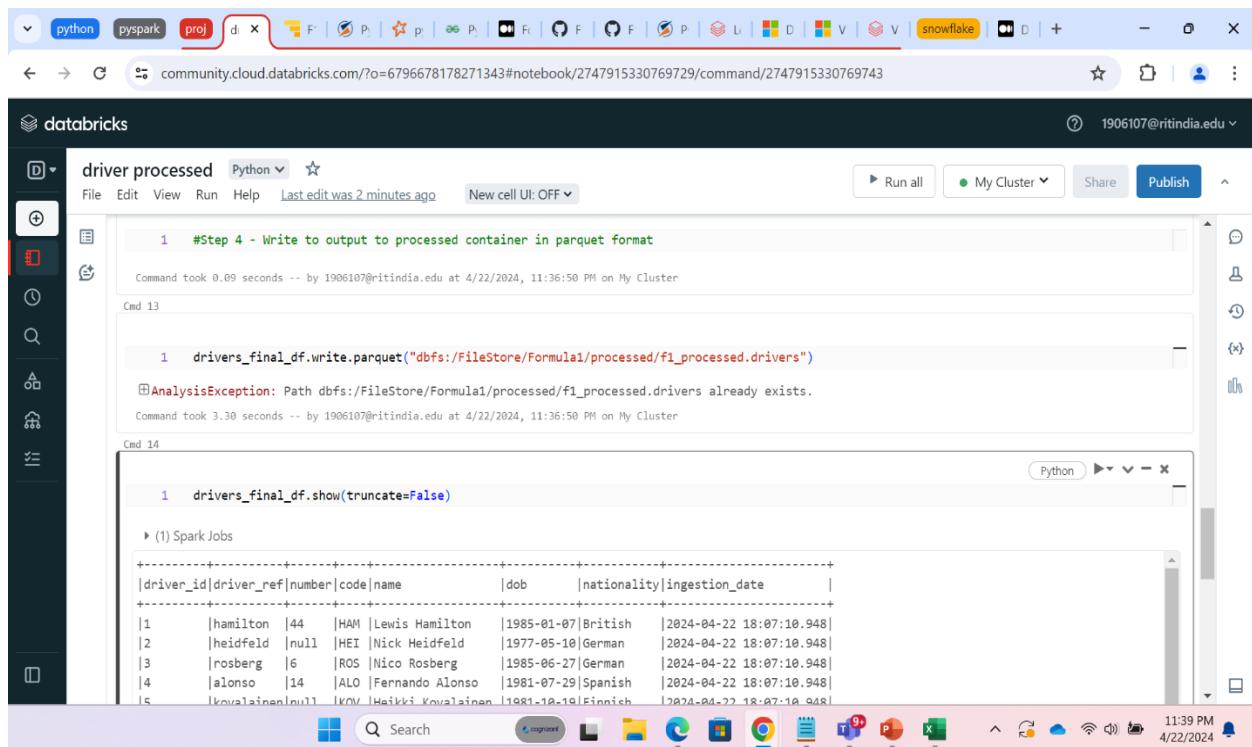
- Cell 1:** Renames columns in the DataFrame `drivers_with_columns_df`:


```
1 drivers_with_columns_df = drivers_with_ingestion_date_df.withColumnRenamed("driverId", "driver_id") \
2   .withColumnRenamed("driverRef", "driver_ref") \
3   .withColumn("name", concat(col("name.forename"), lit(" "), col("name.surname")))
```
- Cell 2:** Prints the DataFrame schema:


```
drivers_with_columns_df: pyspark.sql.dataframe.DataFrame = [driver_id: integer, driver_ref: string ... 7 more fields]
```
- Cell 3:** Prints the command execution time:


```
Command took 0.29 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:36:50 PM on My Cluster
```

Step 5: Storing our dataframe in columnar format that is parquet file.



The screenshot shows a Databricks notebook titled "driver processed" in Python. It contains two cells:

- Cell 1:** Writes the DataFrame to a parquet file:


```
1 #Step 4 - Write to output to processed container in parquet format
```
- Cell 2:** Prints the command execution time:


```
Command took 0.09 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:36:50 PM on My Cluster
```

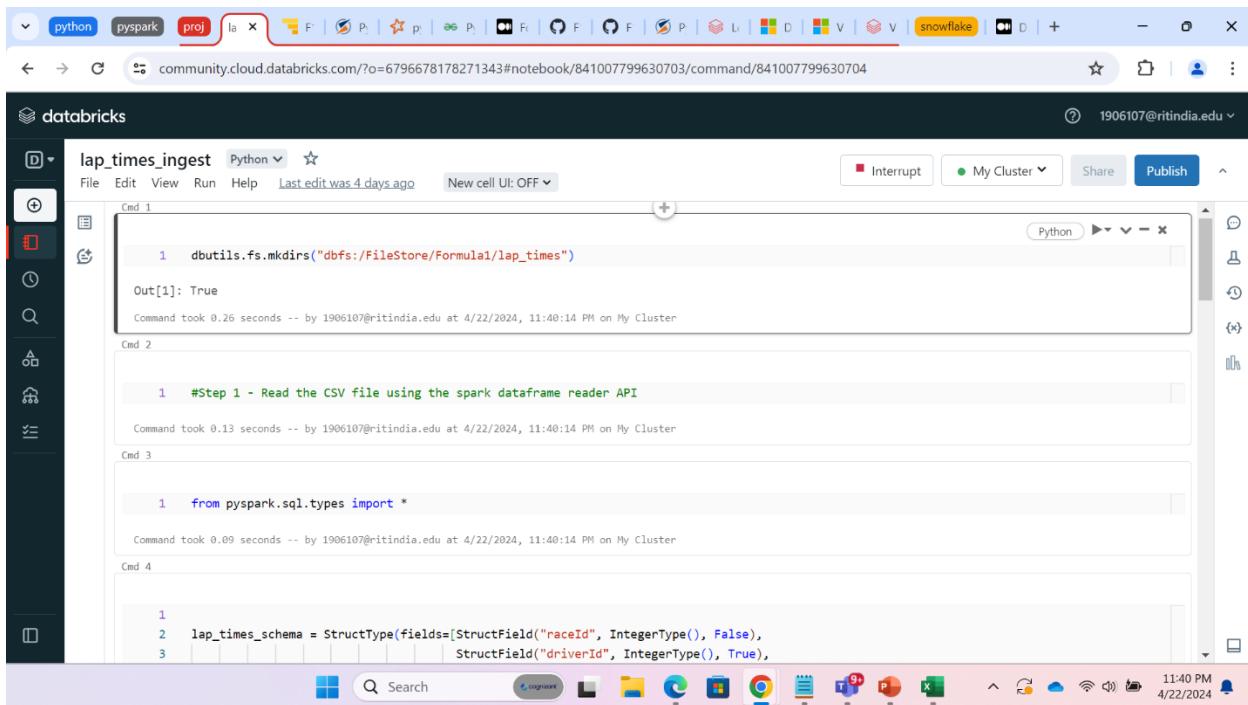
In the notebook sidebar, there is a warning message: "AnalysisException: Path dbfs:/FileStore/Formula1/processed/f1_processed.drivers already exists." Below the notebook, the command line interface (CLI) shows the output of the `show` command:

```
1 drivers_final_df.show(truncate=False)

(1) Spark Jobs
+-----+-----+-----+-----+
| driver_id | driver_ref | number | code | name | dob | nationality | ingestion_date |
+-----+-----+-----+-----+
| 1 | hamilton | 44 | HAM | Lewis Hamilton | 1985-01-07 | British | 2024-04-22 18:07:10.948 |
| 2 | heidfeld | null | HEI | Nick Heidfeld | 1977-05-10 | German | 2024-04-22 18:07:10.948 |
| 3 | rosberg | 6 | ROS | Nico Rosberg | 1985-06-27 | German | 2024-04-22 18:07:10.948 |
| 4 | alonso | 14 | ALO | Fernando Alonso | 1981-07-29 | Spanish | 2024-04-22 18:07:10.948 |
| 5 | kovalainen | null | KOV | Heikki Kovalainen | 1981-10-19 | Finnish | 2024-04-22 18:07:10.948 |
```

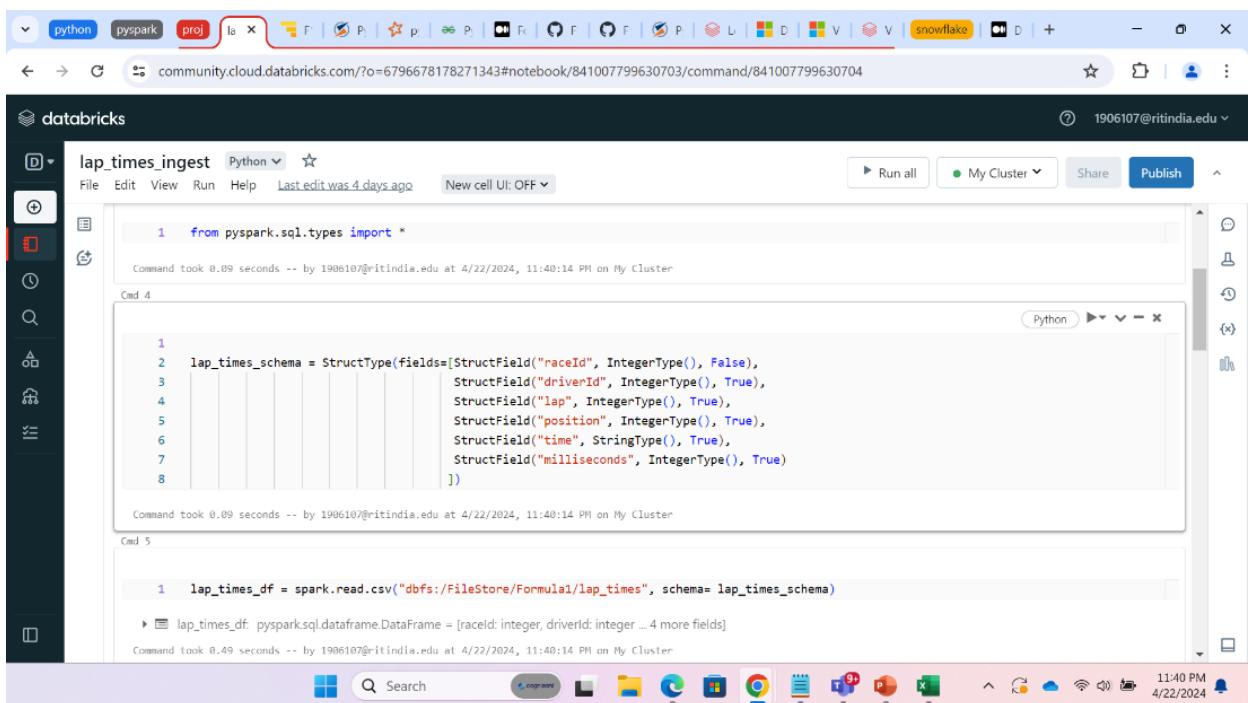
Similarly, we have performed ingestion on all the other files.

LAPTIME INGEST:



The screenshot shows a Databricks notebook titled "lap_times_ingest" in Python. The notebook has four cells:

- Cmd 1:** `dbutils.fs.mkdirs("dbfs:/FileStore/Formula1/lap_times")`
Out[1]: True
Command took 0.26 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:40:14 PM on My Cluster
- Cmd 2:** `#Step 1 - Read the CSV file using the spark dataframe reader API`
Command took 0.13 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:40:14 PM on My Cluster
- Cmd 3:** `from pyspark.sql.types import *`
Command took 0.09 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:40:14 PM on My Cluster
- Cmd 4:** `lap_times_schema = StructType(fields=[StructField("raceId", IntegerType(), False), StructField("driverId", IntegerType(), True), StructField("lap", IntegerType(), True), StructField("position", IntegerType(), True), StructField("time", StringType(), True), StructField("milliseconds", IntegerType(), True)])`
Command took 0.09 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:40:14 PM on My Cluster



The screenshot shows the same Databricks notebook after completing the schema definition. The notebook now includes:

- Cmd 4:** `lap_times_schema = StructType(fields=[StructField("raceId", IntegerType(), False), StructField("driverId", IntegerType(), True), StructField("lap", IntegerType(), True), StructField("position", IntegerType(), True), StructField("time", StringType(), True), StructField("milliseconds", IntegerType(), True)])`
Command took 0.09 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:40:14 PM on My Cluster
- Cmd 5:** `lap_times_df = spark.read.csv("dbfs:/FileStore/Formula1/lap_times", schema= lap_times_schema)`
Out[5]: lap_times_df: pyspark.sql.dataframe.DataFrame = [raceId: integer, driverId: integer ... 4 more fields]
Command took 0.49 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:40:14 PM on My Cluster

python pyspark proj la x F P p F P F P F L D V snowflake D + - o x

community.cloud.databricks.com/?o=6796678178271343#notebook/841007799630703/command/841007799630704

databricks

lap_times_ingest Python ⭐

File Edit View Run Help Last edit was 4 days ago New cell UI: OFF

Run all My Cluster Share Publish

```

1 #Step 2 - Rename columns and add new columns
2 #Rename driverId and raceId
3 #Add ingestion_date with current timestamp

```

Command took 0.08 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:40:14 PM on My Cluster

Cmd 7

```

1 from pyspark.sql.functions import current_timestamp

```

Command took 0.09 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:40:14 PM on My Cluster

Cmd 8

```

1 final_df = lap_times_df.withColumnRenamed("driverId", "driver_id") \
2 .withColumnRenamed("raceId", "race_id") \
3 .withColumn("ingestion_date", current_timestamp())
4

```

final_df: pyspark.sql.dataframe.DataFrame = [race_id: integer, driver_id: integer ... 5 more fields]

Command took 0.19 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:40:14 PM on My Cluster

Cmd 9

Python

11:40 PM 4/22/2024

python pyspark proj la x F P p F P F P F L D V snowflake D + - o x

community.cloud.databricks.com/?o=6796678178271343#notebook/841007799630703/command/3271776022846954

databricks

lap_times_ingest Python ⭐

File Edit View Run Help Last edit was 1 minute ago New cell UI: OFF

Run all My Cluster Share Publish

```

1 final_df.show(truncate=False)

```

(1) Spark Jobs

race_id	driver_id	lap	position	time	milliseconds	ingestion_date
841	20	1	1	1:38.109	98189	2024-04-22 18:12:25.882
841	20	2	1	1:33.006	93006	2024-04-22 18:12:25.882
841	20	3	1	1:32.713	92713	2024-04-22 18:12:25.882
841	20	4	1	1:32.803	92803	2024-04-22 18:12:25.882
841	20	5	1	1:32.342	92342	2024-04-22 18:12:25.882
841	20	6	1	1:32.605	92605	2024-04-22 18:12:25.882
841	20	7	1	1:32.502	92502	2024-04-22 18:12:25.882
841	20	8	1	1:32.537	92537	2024-04-22 18:12:25.882
841	20	9	1	1:33.240	93240	2024-04-22 18:12:25.882
841	20	10	1	1:32.572	92572	2024-04-22 18:12:25.882
841	20	11	1	1:32.669	92669	2024-04-22 18:12:25.882
841	20	12	1	1:32.902	92902	2024-04-22 18:12:25.882
841	20	13	1	1:33.698	93698	2024-04-22 18:12:25.882
841	20	14	3	1:52.075	112075	2024-04-22 18:12:25.882
841	20	15	4	1:38.385	98385	2024-04-22 18:12:25.882
841	20	16	2	1:31.548	91548	2024-04-22 18:12:25.882
841	20	17	1	1:30.800	90800	2024-04-22 18:12:25.882

11:42 PM 4/22/2024

File Edit View Run Help Last edit was now New cell UI: OFF

Run all **My Cluster** **Share** **Publish**

Cmd 10

```
1 dbutils.fs.mkdirs("dbfs:/FileStore/Formula1/processed/lap_times")
```

Out[10]: True

Command took 0.17 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:40:14 PM on My Cluster

Cmd 11

```
1 final_df.write.parquet("dbfs:/FileStore/Formula1/processed/lap_time")
```

AnalysisException: Path dbfs:/FileStore/Formula1/processed/lap_time already exists.

Command took 2.50 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:40:14 PM on My Cluster

Cmd 12

```
1
```

11:42 PM 4/22/2024

PIT STOP INGESTION

File Edit View Run Help Last edit was 9 hours ago New cell UI: OFF

Interrupt **My Cluster** **Share** **Publish**

Cmd 1

```
1 #Step 1 - Read the JSON file using the spark dataframe reader API
```

Command took 0.15 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:43:46 PM on My Cluster

Cmd 2

```
1 from pyspark.sql.types import StructType, StructField, IntegerType, StringType
```

Command took 0.09 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:43:46 PM on My Cluster

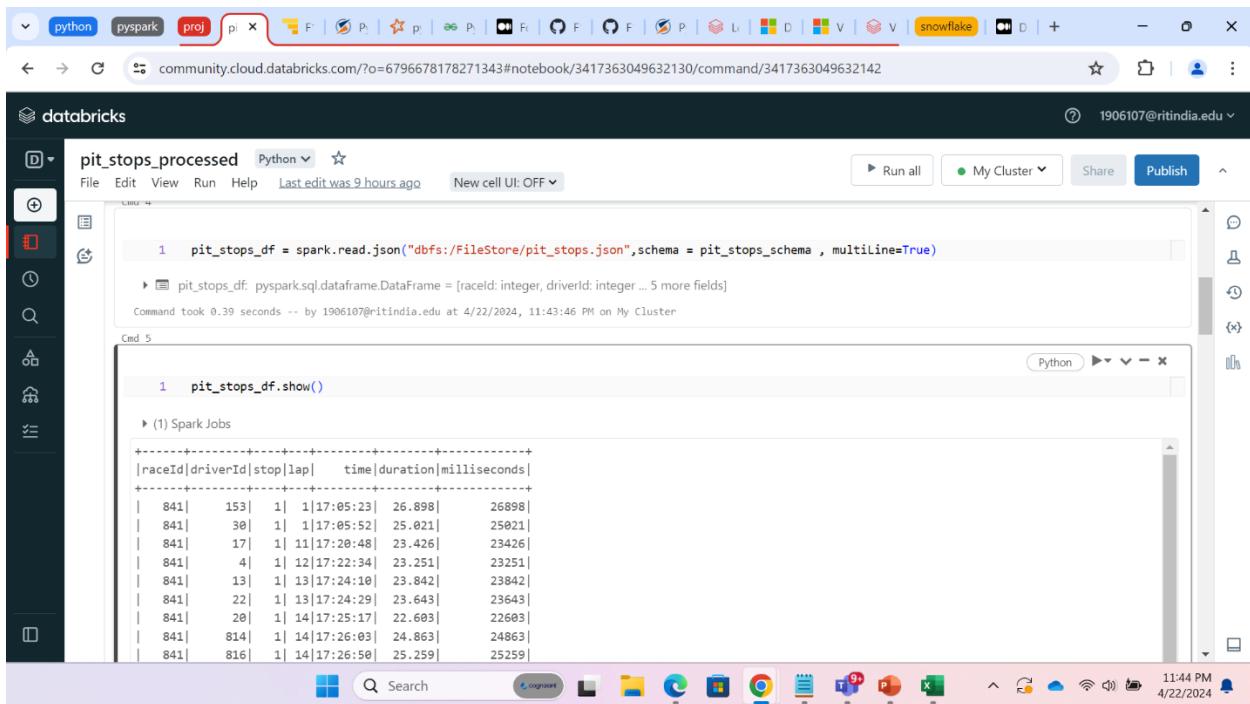
Cmd 3

```
1 pit_stops_schema = StructType(fields=[StructField("raceId", IntegerType(), False),
2                                         StructField("driverId", IntegerType(), True),
3                                         StructField("stop", StringType(), True),
4                                         StructField("lap", IntegerType(), True),
5                                         StructField("time", StringType(), True),
6                                         StructField("duration", StringType(), True),
7                                         StructField("milliseconds", IntegerType(), True)
8                                         ])
```

Command took 0.09 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:43:46 PM on My Cluster

Cmd 4

11:43 PM 4/22/2024



Databricks Notebook titled "pit_stops_processed" (Python). The code reads a JSON file from DBFS and shows its contents.

```

1 pit_stops_df = spark.read.json("dbfs:/FileStore/pit_stops.json", schema = pit_stops_schema, multiLine=True)

pit_stops_df: pyspark.sql.dataframe.DataFrame = [raceId: integer, driverId: integer ... 5 more fields]
Command took 0.39 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:43:46 PM on My Cluster

```

Cmd 5

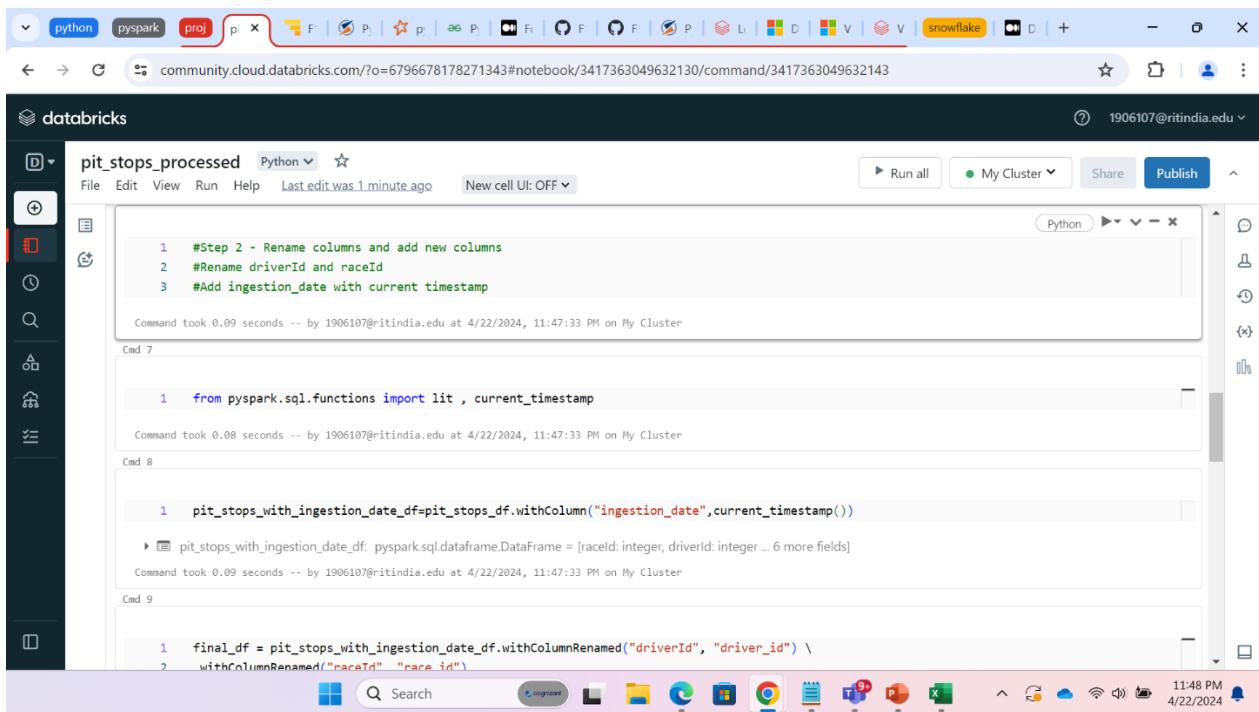
```

1 pit_stops_df.show()

```

(1) Spark Jobs

raceId	driverId	stop	lap	time	duration	milliseconds
841	153	1	1	17:05:23	26.898	26898
841	30	1	1	17:05:52	25.021	25021
841	17	1	11	17:20:48	23.426	23426
841	4	1	12	17:22:34	23.251	23251
841	13	1	13	17:24:10	23.842	23842
841	22	1	13	17:24:29	23.643	23643
841	20	1	14	17:25:17	22.603	22603
841	814	1	14	17:26:03	24.863	24863
841	816	1	14	17:26:50	25.259	25259



Databricks Notebook titled "pit_stops_processed" (Python). The code performs three steps: renaming columns, adding an ingestion date, and renaming raceId to race_id.

```

1 #Step 2 - Rename columns and add new columns
2 #Rename driverId and raceId
3 #Add ingestion_date with current timestamp

Command took 0.09 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:47:33 PM on My Cluster

```

Cmd 7

```

1 from pyspark.sql.functions import lit, current_timestamp

```

Command took 0.08 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:47:33 PM on My Cluster

Cmd 8

```

1 pit_stops_with_ingestion_date_df=pit_stops_df.withColumn("ingestion_date",current_timestamp())

```

pit_stops_with_ingestion_date_df: pyspark.sql.dataframe.DataFrame = [raceId: integer, driverId: integer ... 6 more fields]

Command took 0.09 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:47:33 PM on My Cluster

Cmd 9

```

1 final_df = pit_stops_with_ingestion_date_df.withColumnRenamed("driverId", "driver_id") \
2 .withColumnRenamed("raceId", "race_id")

```

python pyspark proj pit_stops_processed Python New cell UI: OFF Run all My Cluster Share Publish

```
1 final_df = pit_stops_with_ingestion_date_df.withColumnRenamed("driverId", "driver_id") \
2 .withColumnRenamed("raceId", "race_id")

final_df: pyspark.sql.dataframe.DataFrame = [race_id: integer, driver_id: integer ... 6 more fields]
Command took 0.09 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:47:33 PM on My Cluster
```

Cmd 10

```
1 final_df.show(truncate=False)

(1) Spark Jobs
```

race_id	driver_id	stop	lap	time	duration	milliseconds	ingestion_date
841	153	1	1	17:05:23	26.898	26898	2024-04-22 18:17:35.691
841	30	1	1	17:05:52	25.021	25021	2024-04-22 18:17:35.691
841	17	1	11	17:28:48	23.426	23426	2024-04-22 18:17:35.691
841	4	1	12	17:22:34	23.251	23251	2024-04-22 18:17:35.691
841	13	1	13	17:24:10	23.842	23842	2024-04-22 18:17:35.691
841	22	1	13	17:24:29	23.643	23643	2024-04-22 18:17:35.691
841	20	1	14	17:25:17	22.603	22603	2024-04-22 18:17:35.691
841	814	1	14	17:26:03	24.863	24863	2024-04-22 18:17:35.691
841	816	1	14	17:26:50	25.259	25259	2024-04-22 18:17:35.691

11:48 PM 4/22/2024

python pyspark proj pit_stops_processed Python New cell UI: OFF Run all My Cluster Share Publish

```
|841|3|1|16|17:29:00|23.716|23716|[2024-04-22 18:17:35.691|
|841|155|1|16|17:29:06|24.064|24064|[2024-04-22 18:17:35.691|
|841|16|1|16|17:29:08|25.978|25978|[2024-04-22 18:17:35.691|
|841|15|1|16|17:29:49|24.899|24899|[2024-04-22 18:17:35.691|
|841|18|1|17|17:30:24|16.867|16867|[2024-04-22 18:17:35.691|
Command took 0.59 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:47:33 PM on My Cluster
```

Cmd 11

```
1 #Step 3 - Write to output to processed container in parquet format
```

Command took 0.29 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:47:33 PM on My Cluster

Cmd 12

```
1 final_df.write.mode('overwrite').parquet("dbfs:/FileStore/Formul1/processed/f1_processed.pit_stops")
```

Command took 2.10 seconds -- by 1906107@ritindia.edu at 4/22/2024, 11:47:33 PM on My Cluster

Cmd 13

```
1
```

11:48 PM 4/22/2024

Qualifying ingest

Databricks Notebook: qualifying_processed (Python)

```

1 dbutils.fs.mkdirs("dbfs:/FileStore/Formula1/qualifying")
** Command execution completed
Out[1]: True

Cmd 2

1 #Step 1 - Read the JSON file using the spark dataframe reader API
** Command execution completed

Cmd 3

1 from pyspark.sql.types import StructType, StructField, IntegerType, StringType
** Command execution completed

Cmd 4

1
2 qualifying_schema = StructType(fields=[StructField("qualifyId", IntegerType(), False),
                                         StructField("raceId", IntegerType(), True),
                                         StructField("driverId", IntegerType(), True),
                                         StructField("constructorId", IntegerType(), True),
                                         StructField("number", IntegerType(), True),
                                         StructField("position", IntegerType(), True),
                                         StructField("q1", StringType(), True),
                                         StructField("q2", StringType(), True),
                                         StructField("q3", StringType(), True),
                                         ])

```

Databricks Notebook: qualifying_processed (Python)

```

1
2 qualifying_schema = StructType(fields=[StructField("qualifyId", IntegerType(), False),
                                         StructField("raceId", IntegerType(), True),
                                         StructField("driverId", IntegerType(), True),
                                         StructField("constructorId", IntegerType(), True),
                                         StructField("number", IntegerType(), True),
                                         StructField("position", IntegerType(), True),
                                         StructField("q1", StringType(), True),
                                         StructField("q2", StringType(), True),
                                         StructField("q3", StringType(), True),
                                         ])
Command took 0.08 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:28:40 AM on compute

Cmd 5

1 qualifying_df = spark.read.json("dbfs:/FileStore/Formula1/qualifying",schema =qualifying_schema , multiline= True)
Running command...
qualifying_df: pyspark.sql.dataframe.DataFrame = [qualifyId: integer, raceId: integer ... 7 more fields]

```

python pyspark proj q x F P p F P L D V snowflake D +

community.cloud.databricks.com/?o=6796678178271343#notebook/3417363049632145/command/3417363049632151

databricks

qualifying_processed Python

Last command failed

File Edit View Run Help Last edit was 19 hours ago New cell UI: OFF

Run all Compute Share Publish

```

1 #Step 2 - Rename columns and add new columns
2 #Rename qualifyingId, driverId, constructorId and raceId
3 #Add ingestion_date with current timestamp

Command took 0.22 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:28:40 AM on compute

```

Cmd 8

```

1 from pyspark.sql.functions import lit, current_timestamp

Command took 0.16 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:28:40 AM on compute

```

Cmd 9

```

1 final_df = qualifying_df.withColumnRenamed("qualifyId", "qualify_id") \
2 .withColumnRenamed("driverId", "driver_id") \
3 .withColumnRenamed("raceId", "race_id") \
4 .withColumnRenamed("constructorId", "constructor_id") \
5 .withColumn("ingestion_date", current_timestamp())

final_df: pyspark.sql.dataframe.DataFrame = [qualify_id: integer, race_id: integer ... 8 more fields]

```

Command took 0.88 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:28:40 AM on compute

Cmd 10

Python 9:29 AM 4/23/2024

python pyspark proj q x F P p F P L D V snowflake D +

community.cloud.databricks.com/?o=6796678178271343#notebook/3417363049632145/command/3417363049632151

databricks

qualifying_processed Python

1906107@ritindia.edu

File Edit View Run Help Last edit was 19 hours ago New cell UI: OFF

Interrupt Compute Share Publish

Cmd 6

```

1 qualifying_df.show()

```

(1) Spark Jobs

qualifyId	raceId	driverId	constructorId	number	position	q1	q2	q3
1	18	1		22		1 1:26.572 1:25.187 1:26.714		
2	18	9		2	4	2 1:26.103 1:25.315 1:26.869		
3	18	5		1	23	3 1:25.664 1:25.452 1:27.079		
4	18	13		6	2	4 1:25.994 1:25.691 1:27.178		
5	18	2		2	3	5 1:25.960 1:25.518 1:27.236		
6	18	15		7	11	6 1:26.427 1:26.101 1:28.527		
7	18	3		3	7	7 1:26.295 1:26.059 1:28.687		
8	18	14		9	9	8 1:26.381 1:26.063 1:29.041		
9	18	10		7	12	9 1:26.919 1:26.164 1:29.593		
10	18	20		5	15	10 1:26.702 1:25.842 \N		
11	18	22		11	17	11 1:26.369 1:26.173 \N		
12	18	4		4	5	12 1:26.907 1:26.188 \N		
13	18	18		11	16	13 1:26.712 1:26.259 \N		
14	18	6		3	8	14 1:26.891 1:26.413 \N		
15	18	17		9	10	15 1:26.914 \N \N		
16	18	8		6	1	16 1:26.140 \N \N		

Python 9:29 AM 4/23/2024

Last command failed

Run all | compute | Share | Publish

```

1 #Step 2 - Rename columns and add new columns
2 #Rename qualifyingId, driverId, constructorId and raceId
3 #Add ingestion_date with current timestamp

Command took 0.22 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:28:40 AM on compute

Cmd 8

1 from pyspark.sql.functions import lit, current_timestamp

Command took 0.16 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:28:40 AM on compute

Cmd 9

1 final_df = qualifying_df.withColumnRenamed("qualifyId", "qualify_id") \
2 .withColumnRenamed("driverId", "driver_id") \
3 .withColumnRenamed("raceId", "race_id") \
4 .withColumnRenamed("constructorId", "constructor_id") \
5 .withColumn("ingestion_date", current_timestamp())

final_df: pyspark.sql.dataframe.DataFrame = [qualify_id: integer, race_id: integer ... 8 more fields]

Command took 0.88 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:28:40 AM on compute

Cmd 10
  
```

1906107@ritindia.edu

Run all | compute | Share | Publish

```

1 final_df.show(truncate=False)

+---+---+---+---+---+---+---+
|qualify_id|race_id|driver_id|constructor_id|number|position|q1    |q2    |q3    |ingestion_date   |
+---+---+---+---+---+---+---+
| 1     | 18    | 1      | 1      | 22   | 1      | 1:26.572|1:25.187|1:26.714|2024-04-23 04:00:32.158|
| 2     | 18    | 9      | 2      | 4    | 2      | 1:26.103|1:25.315|1:26.869|2024-04-23 04:00:32.158|
| 3     | 18    | 5      | 1      | 23   | 3      | 1:25.664|1:25.452|1:27.079|2024-04-23 04:00:32.158|
| 4     | 18    | 13    | 6      | 2    | 4      | 1:25.994|1:25.691|1:27.178|2024-04-23 04:00:32.158|
| 5     | 18    | 2      | 2      | 3    | 5      | 1:25.960|1:25.518|1:27.236|2024-04-23 04:00:32.158|
| 6     | 18    | 15    | 7      | 11   | 6      | 1:26.427|1:26.101|1:28.527|2024-04-23 04:00:32.158|
| 7     | 18    | 3      | 3      | 7    | 7      | 1:26.295|1:26.059|1:28.687|2024-04-23 04:00:32.158|
| 8     | 18    | 14    | 9      | 9    | 8      | 1:26.381|1:26.063|1:29.041|2024-04-23 04:00:32.158|
| 9     | 18    | 10    | 7      | 12   | 9      | 1:26.919|1:26.164|1:29.593|2024-04-23 04:00:32.158|
| 10    | 18    | 20    | 5      | 15   | 10     | 1:26.702|1:25.842|\N    |2024-04-23 04:00:32.158|
| 11    | 18    | 22    | 11     | 17   | 11     | 1:26.369|1:26.173|\N    |2024-04-23 04:00:32.158|
| 12    | 18    | 4      | 4      | 5    | 12     | 1:26.907|1:26.188|\N    |2024-04-23 04:00:32.158|
| 13    | 18    | 18    | 11     | 16   | 13     | 1:26.712|1:26.259|\N    |2024-04-23 04:00:32.158|
| 14    | 18    | 6      | 3      | 8    | 14     | 1:26.891|1:26.413|\N    |2024-04-23 04:00:32.158|
| 15    | 18    | 17    | 9      | 10   | 15     | 1:26.914|\N    |\N    |2024-04-23 04:00:32.158|
| 16    | 18    | 19    | 10     | 11   | 16     | 1:26.925|1:26.222|\N    |2024-04-23 04:00:32.158|
  
```

The screenshot shows a Databricks notebook titled "qualifying_processed" running on Python. The notebook contains three cells:

- Cell 1:** Contains the command `final_df.write.parquet("dbfs:/FileStore/Formula1/processed/f1_processed.qualifying")` and an error message: "AnalysisException: Path dbfs:/FileStore/Formula1/processed/f1_processed.qualifying already exists." The command took 6.01 seconds.
- Cell 2:** Contains the command `final_df.show(truncate=False)` followed by a table of race data. The table has columns: qualify_id, race_id, driver_id, constructor_id, number, position, q1, q2, q3, ingestion_date. The data is as follows:

qualify_id	race_id	driver_id	constructor_id	number	position	q1	q2	q3	ingestion_date
1	18	1	1	22	1	1:26.572	1:25.187	1:26.714	2024-04-23 04:01:10.743
2	18	9	2	4	2	1:26.103	1:25.315	1:26.869	2024-04-23 04:01:10.743
3	18	5	1	23	3	1:25.664	1:25.452	1:27.079	2024-04-23 04:01:10.743
4	18	13	6	2	4	1:25.994	1:25.691	1:27.178	2024-04-23 04:01:10.743
5	18	2	2	3	5	1:25.960	1:25.518	1:27.236	2024-04-23 04:01:10.743
6	18	15	7	11	6	1:26.427	1:26.101	1:28.527	2024-04-23 04:01:10.743
7	18	3	3	7	7	1:26.295	1:26.059	1:28.687	2024-04-23 04:01:10.743

- Cell 3:** Shows the output of the `show` command.

Race Ingest

The screenshot shows a Databricks notebook titled "RACES_INGEST" running on Python. The notebook contains three cells:

- Cell 1:** Contains the command `races_df = spark.read.csv("dbfs:/FileStore/Formula1/races.csv")` and an error message: "AnalysisException: Path dbfs:/FileStore/Formula1/races.csv already exists." The command took 0.57 seconds.
- Cell 2:** Contains the command `races_schema = StructType(fields=[StructField("raceId", IntegerType(), False), StructField("year", IntegerType(), True), StructField("round", IntegerType(), True), StructField("circuitId", IntegerType(), True), StructField("name", StringType(), True), StructField("date", DateType(), True), StructField("time", StringType(), True), StructField("url", StringType(), True)])`.
- Cell 3:** Shows the output of the schema definition command.

```

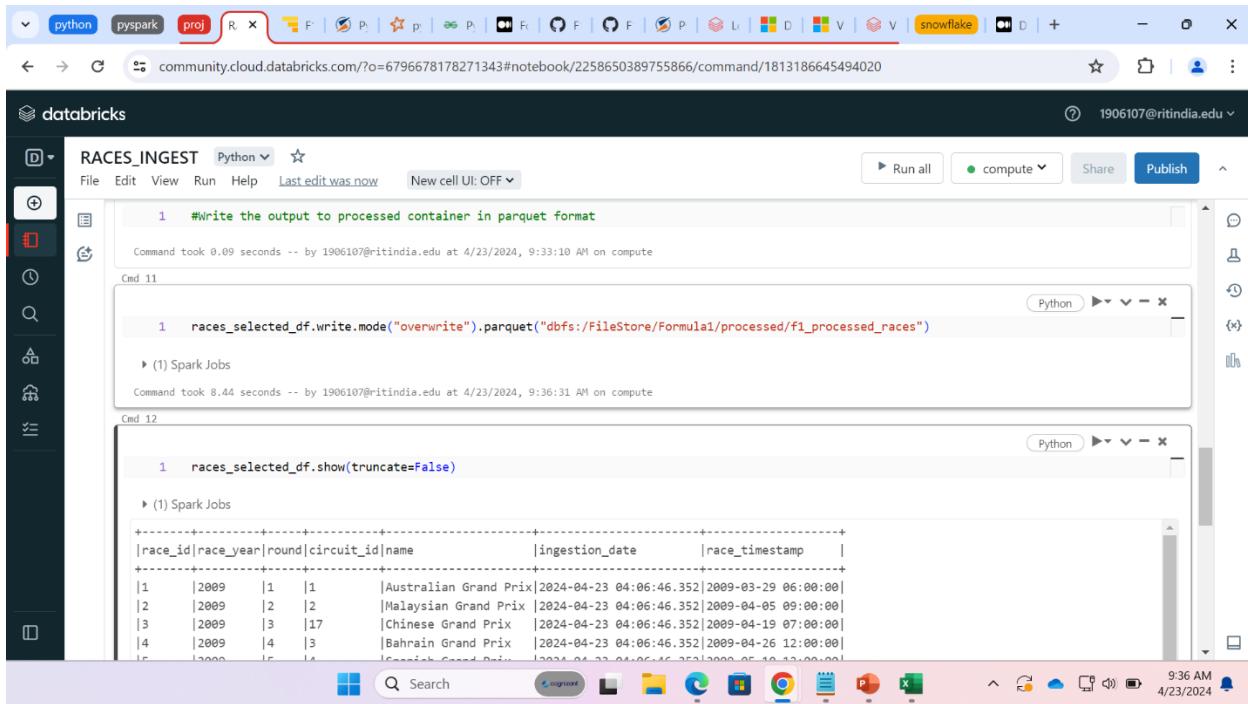
1 races_df = spark.read.csv("dbfs:/FileStore/races.csv", schema =races_schema )
2
3 #Step 2 - Add ingestion date and race_timestamp to the dataframe
4
5 from pyspark.sql.functions import to_timestamp, concat, col, lit
6
7 races_with_timestamp_df = races_df.withColumn("race_timestamp", to_timestamp(concat(col('date'), lit(' '), col('time')), 'yyyy-MM-dd
HH:mm:ss')).withColumn("ingestion_date", current_timestamp())
8
9 races_with_timestamp_df: pyspark.sql.dataframe.DataFrame = [raceld: integer, year: integer ... 8 more fields]

```

```

1 #Step 3 - Select only the columns required & rename as required
2
3 races_selected_df = races_with_timestamp_df .select(col('raceId').alias('race_id'), col('year').alias('race_year'),
4 col('round'), col('circuitId').alias('circuit_id'), col('name'), col('ingestion_date'),
5 col('race_timestamp'))
6
7 races_selected_df: pyspark.sql.dataframe.DataFrame = [race_id: integer, race_year: integer ... 5 more fields]
8
9 #Write the output to processed container in parquet format
10
11 races_selected_df.write.parquet("dbfs:/FileStore/Formula1/processed/f1_processed_races")

```



The screenshot shows a Databricks notebook titled "RACES_INGEST" in Python. The notebook contains two cells:

```

1 #Write the output to processed container in parquet format

```

Command took 0.09 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:33:10 AM on compute

```

1 races_selected_df.write.mode("overwrite").parquet("dbfs:/FileStore/Formula1/processed/f1_processed_races")

```

▶ (1) Spark Jobs

Command took 8.44 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:36:31 AM on compute

```

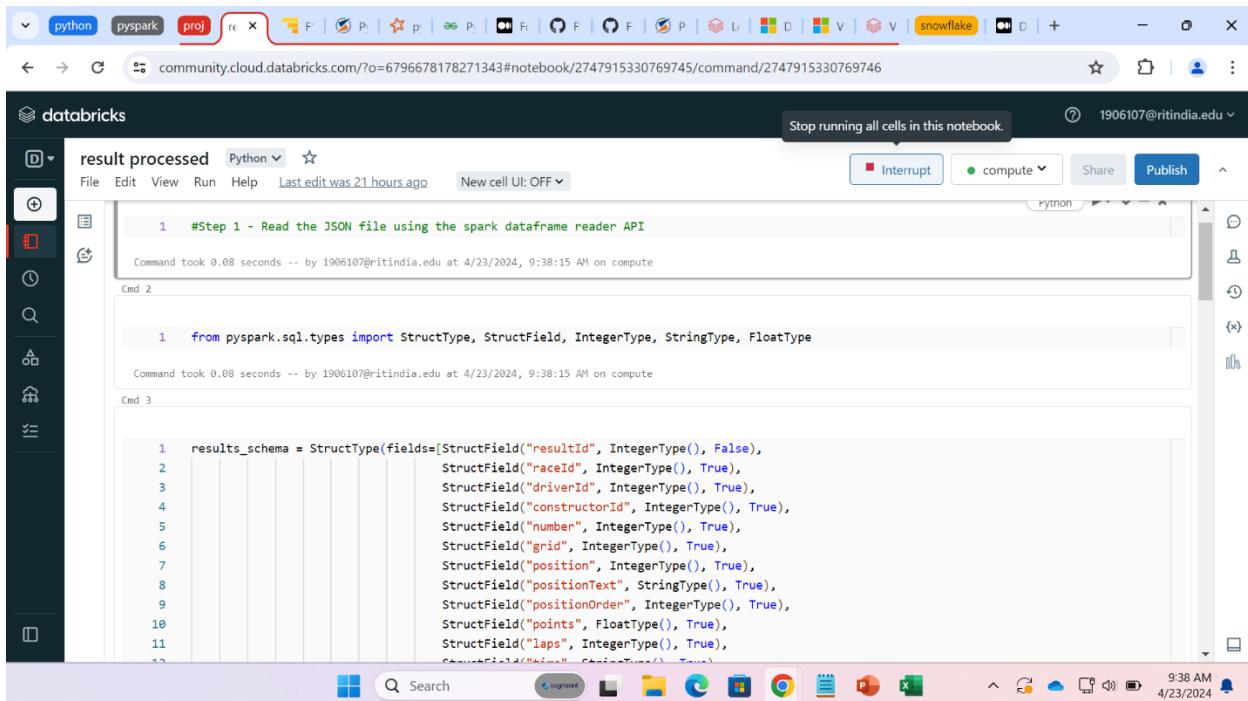
1 races_selected_df.show(truncate=False)

```

▶ (1) Spark Jobs

race_id	race_year	round	circuit_id	name	ingestion_date	race_timestamp
1	2009	1	1	Australian Grand Prix	2024-04-23 04:06:46.352	2009-03-29 06:00:00
2	2009	2	2	Malaysian Grand Prix	2024-04-23 04:06:46.352	2009-04-05 09:00:00
3	2009	3	17	Chinese Grand Prix	2024-04-23 04:06:46.352	2009-04-19 07:00:00
4	2009	4	3	Bahrain Grand Prix	2024-04-23 04:06:46.352	2009-04-26 12:00:00
5	2009	5	4	French Grand Prix	2024-04-23 04:06:46.352	2009-04-27 12:00:00

Result file ingestion



The screenshot shows a Databricks notebook titled "result processed" in Python. The notebook contains three cells:

```

1 #Step 1 - Read the JSON file using the spark dataframe reader API

```

Command took 0.08 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:38:15 AM on compute

```

1 from pyspark.sql.types import StructType, StructField, IntegerType, StringType, FloatType

```

Command took 0.08 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:38:15 AM on compute

```

1 results_schema = StructType(fields=[StructField("resultId", IntegerType(), False),
                                     StructField("raceId", IntegerType(), True),
                                     StructField("driverId", IntegerType(), True),
                                     StructField("constructorId", IntegerType(), True),
                                     StructField("number", IntegerType(), True),
                                     StructField("grid", IntegerType(), True),
                                     StructField("position", IntegerType(), True),
                                     StructField("positionText", StringType(), True),
                                     StructField("positionOrder", IntegerType(), True),
                                     StructField("points", FloatType(), True),
                                     StructField("laps", IntegerType(), True),
                                     StructField("milliseconds", IntegerType(), True)])

```

python pyspark proj re x F P p F P F P F L D V snowflake D + - o X

community.cloud.databricks.com/?o=6796678178271343#notebook/2747915330769745/command/2747915330769746

databricks

result processed Python v star

File Edit View Run Help Last edit was 21 hours ago New cell UI: OFF

Run all Compute Share Publish

```

1 results_schema = StructType(fields=[StructField("resultId", IntegerType(), False),
2                                     StructField("raceId", IntegerType(), True),
3                                     StructField("driverId", IntegerType(), True),
4                                     StructField("constructorId", IntegerType(), True),
5                                     StructField("number", IntegerType(), True),
6                                     StructField("grid", IntegerType(), True),
7                                     StructField("position", IntegerType(), True),
8                                     StructField("positionText", StringType(), True),
9                                     StructField("positionOrder", IntegerType(), True),
10                                    StructField("points", FloatType(), True),
11                                    StructField("laps", IntegerType(), True),
12                                    StructField("time", StringType(), True),
13                                    StructField("milliseconds", IntegerType(), True),
14                                    StructField("fastestLap", IntegerType(), True),
15                                    StructField("rank", IntegerType(), True),
16                                    StructField("fastestLapTime", StringType(), True),
17                                    StructField("fastestLapSpeed", FloatType(), True),
18                                    StructField("statusId", StringType(), True)])

```

Command took 0.08 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:38:15 AM on compute

Cmd 4

```

1 results_df = spark.read.json("dbfs:/FileStore/results.json", schema = results_schema)

```

9:38 AM 4/23/2024

python pyspark proj re x F P p F P F F P F L D V snowflake D + - o X

community.cloud.databricks.com/?o=6796678178271343#notebook/2747915330769745/command/2747915330769746

databricks

result processed Python v star

File Edit View Run Help Last edit was 21 hours ago New cell UI: OFF

Run all Compute Share Publish

```

1 #Step 2 - Rename columns and add new columns

```

Command took 0.09 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:38:15 AM on compute

Cmd 6

```

1 from pyspark.sql.functions import lit, current_timestamp

```

Command took 0.09 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:38:15 AM on compute

Cmd 7

```

1 results_with_columns_df = results_df.withColumnRenamed("resultId", "result_id") \
2 .withColumnRenamed("raceId", "race_id") \
3 .withColumnRenamed("driverId", "driver_id") \
4 .withColumnRenamed("constructorId", "constructor_id") \
5 .withColumnRenamed("positionText", "position_text") \
6 .withColumnRenamed("positionOrder", "position_order") \
7 .withColumnRenamed("fastestLap", "fastest_lap") \
8 .withColumnRenamed("fastestLapTime", "fastest_lap_time") \
9 .withColumnRenamed("fastestLapSpeed", "fastest_lap_speed")

```

results_with_columns_df: pyspark.sql.dataframe.DataFrame = [result_id: integer, race_id: integer ... 16 more fields]

Python 9:38 AM 4/23/2024

python pyspark proj re x F P p F P F P F P F L D V V snowflake D +

community.cloud.databricks.com/?o=6796678178271343#notebook/2747915330769745/command/2747915330769746

databricks

result processed Python ⚡

File Edit View Run Help Last edit was 21 hours ago New cell UI: OFF

Run all compute Share Publish

```

1 results_with_ingestion_date_df = results_with_columns_df.withColumn("ingestion_date", current_timestamp())
  results_with_ingestion_date_df: pyspark.sql.dataframe.DataFrame = [result_id: integer, race_id: integer ... 17 more fields]
Command took 0.18 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:38:15 AM on compute

Cmd 9

1 #Step 3 - Drop the unwanted column
  Command took 0.09 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:38:15 AM on compute

Cmd 10

1 from pyspark.sql.functions import col
  Command took 0.09 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:38:15 AM on compute

Cmd 11

1 results_final_df = results_with_ingestion_date_df.drop(col("statusId"))
  results_final_df: pyspark.sql.dataframe.DataFrame = [result_id: integer, race_id: integer ... 16 more fields]
Command took 0.09 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:38:15 AM on compute

```

Python 9:39 AM 4/23/2024

python pyspark proj re x F P p F P F P F P F L D V V snowflake D +

community.cloud.databricks.com/?o=6796678178271343#notebook/2747915330769745/command/1813186645494022

databricks

result processed Python ⚡

File Edit View Run Help Last edit was now New cell UI: OFF

Run all compute Share Publish

```

1 results_deduped_df = results_final_df.dropDuplicates(['race_id', 'driver_id'])
  results_deduped_df: pyspark.sql.dataframe.DataFrame = [result_id: integer, race_id: integer ... 16 more fields]
Command took 0.09 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:38:15 AM on compute

Cmd 13

1 results_deduped_df.show(truncate=False)
  (2) Spark Jobs
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|result_id|race_id|driver_id|constructor_id|number|grid|position|position_text|position_order|points|laps|time      |milliseconds|fastest_lap|rank|f
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|7573    |1       |1       |1       |1       |18   |null    |D          |20        |0.0     |58   |\N      |null      |39      |13   |
|1:29.020 |         |214.455 |          |          |2024-04-23 04:09:58.81|          |          |          |          |          |          |          |          |          |
|7563    |1       |2       |2       |6       |9     |10      |          |10        |0.0     |58   |+7.085  |5662869  |48      |5     |
|1:28.283 |         |216.245 |          |          |2024-04-23 04:09:58.81|          |          |          |          |          |          |          |          |

```

Python 9:40 AM 4/23/2024

result processed Python

File Edit View Run Help Last edit was 1 minute ago New cell UI: OFF

Run all compute Share Publish

Cmd 14

```
1 #Step 4 - Write output to processed container in parquet format
```

Command took 0.00 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:38:15 AM on compute

Cmd 15

```
1 results_deduped_df.write.parquet("dbfs:/FileStore/Formula1/processed/f1_processed.result")
```

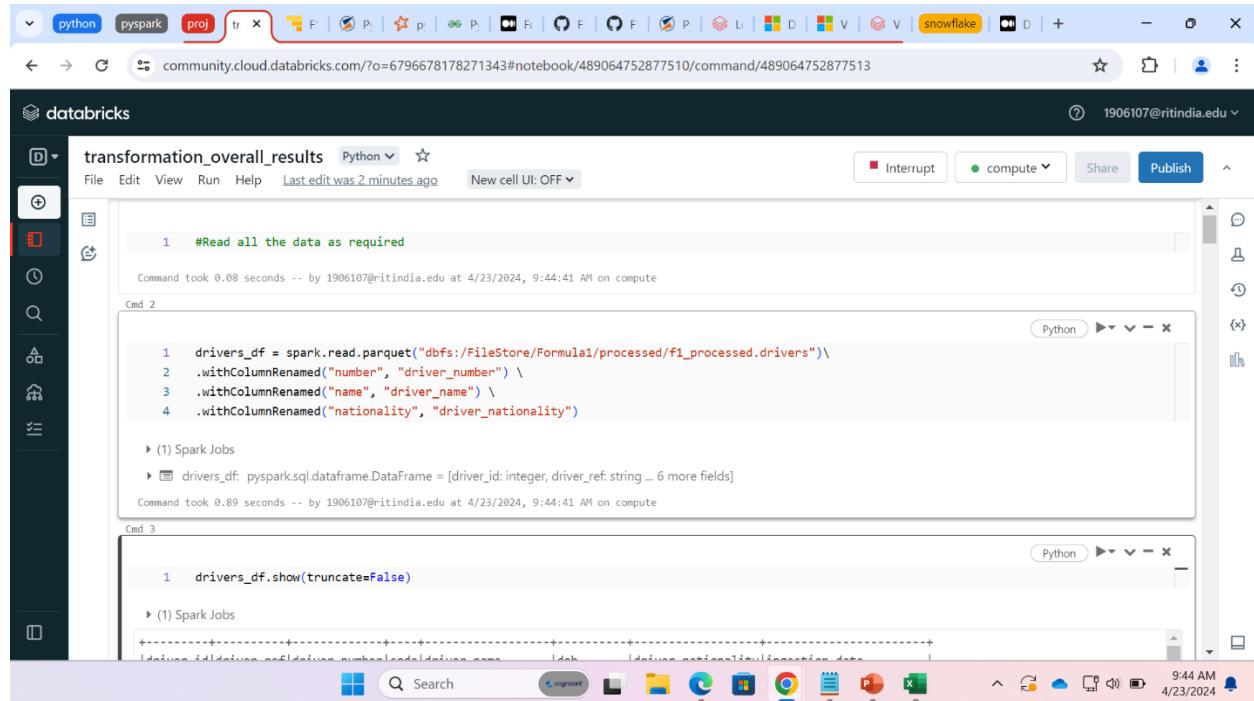
(1) Spark Jobs

AnalysisException: Path dbfs:/FileStore/Formula1/processed/f1_processed.result already exists.

Command took 15.37 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:38:15 AM on compute

• REQUIREMENT 2:

Step 1: Read all the parquet files from dbfs file store



```

1 #Read all the data as required
Command took 0.08 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:44:41 AM on compute

Cmd 2

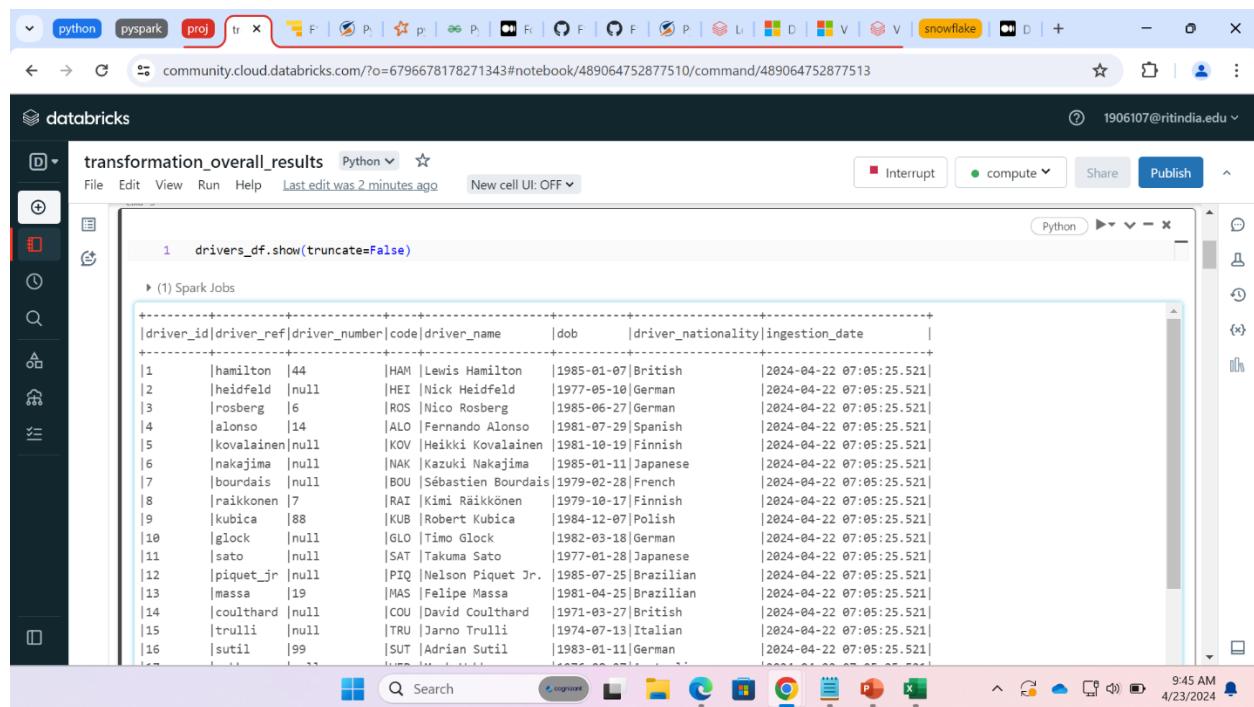
1 drivers_df = spark.read.parquet("dbfs:/FileStore/Formula1/processed/f1_processed.drivers")
2 .withColumnRenamed("number", "driver_number") \
3 .withColumnRenamed("name", "driver_name") \
4 .withColumnRenamed("nationality", "driver_nationality")

▶ (1) Spark Jobs
▶ (1) drivers_df: pyspark.sql.dataframe.DataFrame = [driver_id: integer, driver_ref: string ... 6 more fields]
Command took 0.89 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:44:41 AM on compute

Cmd 3

1 drivers_df.show(truncate=False)
▶ (1) Spark Jobs
+-----+-----+-----+-----+-----+
|driver_id|driver_ref|driver_number|code|driver_name      |dob          |driver_nationality|ingestion_date |
+-----+-----+-----+-----+-----+
|1       |hamilton |44           |HAN |Lewis Hamilton |1985-01-07|British        |2024-04-22 07:05:25.521|
|2       |heidfeld |null         |HEI |Nick Heidfeld  |1977-05-10|German         |2024-04-22 07:05:25.521|
|3       |rosberg  |6             |ROS |Nico Rosberg  |1985-06-27|German         |2024-04-22 07:05:25.521|
|4       |alonso   |14            |ALO |Fernando Alonso|1981-07-29|Spanish        |2024-04-22 07:05:25.521|
|5       |kovvalainen|null         |KOV |Heikki Kovaleinen|1981-10-19|Finnish        |2024-04-22 07:05:25.521|
|6       |nakajima |null         |NAK |Kazuki Nakajima|1985-01-11|Japanese       |2024-04-22 07:05:25.521|
|7       |bourdais |null         |BOU |Sébastien Bourdais|1979-02-28|French         |2024-04-22 07:05:25.521|
|8       |raikkonen |7             |RAI |Kimi Räikkönen |1979-10-17|Finnish        |2024-04-22 07:05:25.521|
|9       |kubica   |88            |KUB |Robert Kubica  |1984-12-07|Polish         |2024-04-22 07:05:25.521|
|10      |glock    |null         |GLO |Timo Glock    |1982-03-18|German         |2024-04-22 07:05:25.521|
|11      |sato     |null         |SAT |Takuma Sato   |1977-01-28|Japanese       |2024-04-22 07:05:25.521|
|12      |piquet_jr|null         |PIQ |Nelson Piquet Jr.|1985-07-25|Brazilian      |2024-04-22 07:05:25.521|
|13      |massa    |19            |MAS |Felipe Massa  |1981-04-25|Brazilian      |2024-04-22 07:05:25.521|
|14      |coulthard|null         |COU |David Coulthard|1971-03-27|British        |2024-04-22 07:05:25.521|
|15      |trulli   |null         |TRU |Jarno Trulli  |1974-07-13|Italian        |2024-04-22 07:05:25.521|
|16      |sutil    |99            |SUT |Adrian Sutil  |1983-01-11|German         |2024-04-22 07:05:25.521|
+-----+-----+-----+-----+-----+

```



```

1 drivers_df.show(truncate=False)
▶ (1) Spark Jobs
+-----+-----+-----+-----+-----+
|driver_id|driver_ref|driver_number|code|driver_name      |dob          |driver_nationality|ingestion_date |
+-----+-----+-----+-----+-----+
|1       |hamilton |44           |HAN |Lewis Hamilton |1985-01-07|British        |2024-04-22 07:05:25.521|
|2       |heidfeld |null         |HEI |Nick Heidfeld  |1977-05-10|German         |2024-04-22 07:05:25.521|
|3       |rosberg  |6             |ROS |Nico Rosberg  |1985-06-27|German         |2024-04-22 07:05:25.521|
|4       |alonso   |14            |ALO |Fernando Alonso|1981-07-29|Spanish        |2024-04-22 07:05:25.521|
|5       |kovvalainen|null         |KOV |Heikki Kovaleinen|1981-10-19|Finnish        |2024-04-22 07:05:25.521|
|6       |nakajima |null         |NAK |Kazuki Nakajima|1985-01-11|Japanese       |2024-04-22 07:05:25.521|
|7       |bourdais |null         |BOU |Sébastien Bourdais|1979-02-28|French         |2024-04-22 07:05:25.521|
|8       |raikkonen |7             |RAI |Kimi Räikkönen |1979-10-17|Finnish        |2024-04-22 07:05:25.521|
|9       |kubica   |88            |KUB |Robert Kubica  |1984-12-07|Polish         |2024-04-22 07:05:25.521|
|10      |glock    |null         |GLO |Timo Glock    |1982-03-18|German         |2024-04-22 07:05:25.521|
|11      |sato     |null         |SAT |Takuma Sato   |1977-01-28|Japanese       |2024-04-22 07:05:25.521|
|12      |piquet_jr|null         |PIQ |Nelson Piquet Jr.|1985-07-25|Brazilian      |2024-04-22 07:05:25.521|
|13      |massa    |19            |MAS |Felipe Massa  |1981-04-25|Brazilian      |2024-04-22 07:05:25.521|
|14      |coulthard|null         |COU |David Coulthard|1971-03-27|British        |2024-04-22 07:05:25.521|
|15      |trulli   |null         |TRU |Jarno Trulli  |1974-07-13|Italian        |2024-04-22 07:05:25.521|
|16      |sutil    |99            |SUT |Adrian Sutil  |1983-01-11|German         |2024-04-22 07:05:25.521|
+-----+-----+-----+-----+-----+

```

python pyspark proj tr x F P p F P F P F L D V snowflake D +

community.cloud.databricks.com/?o=6796678178271343#notebook/489064752877510/command/489064752877515

databricks

transformation_overall_results Python New cell UI: OFF

Last command failed

Run all compute Share Publish

Cmd 4

```
1 constructors_df = spark.read.parquet("dbfs:/FileStore/Formula1/processed/f1_processed.constructors") \
2 .withColumnRenamed("name", "team")
```

(1) Spark Jobs

constructors_df: pyspark.sql.dataframe.DataFrame = [constructor_id: integer, constructor_ref: string ... 3 more fields]

Command took 1.09 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:44:41 AM on compute

Cmd 5

```
1 constructors_df.show(truncate=False)
```

(1) Spark Jobs

constructor_id	constructor_ref	team	nationality	ingestion_date
1	mclaren	McLaren	British	2024-04-18 17:31:45.863
2	bmw_sauber	BMW Sauber	German	2024-04-18 17:31:45.863
3	williams	Williams	British	2024-04-18 17:31:45.863
4	renault	Renault	French	2024-04-18 17:31:45.863
5	toro_rosso	Toro Rosso	Italian	2024-04-18 17:31:45.863

9:45 AM 4/23/2024

python pyspark proj tr x F P p F P F P F L D V snowflake D +

community.cloud.databricks.com/?o=6796678178271343#notebook/489064752877510/command/489064752877515

databricks

transformation_overall_results Python New cell UI: OFF

Last command failed

Run all compute Share Publish

Cmd 6

```
1 circuits_df = spark.read.parquet("dbfs:/FileStore/Formula1/processed/circuits") \
2 .withColumnRenamed("location", "circuit_location")
```

(1) Spark Jobs

circuits_df: pyspark.sql.dataframe.DataFrame = [circuit_id: string, circuit_ref: string ... 7 more fields]

Command took 0.89 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:44:41 AM on compute

Cmd 7

```
1 circuits_df.show(truncate=False)
```

(1) Spark Jobs

circuit_id	circuit_ref	name	circuit_location	country	latitude	longitude	altitude	ingestion_date
1	albert_park	Albert Park Grand Prix Circuit	Melbourne	Australia	-37.8497	144.968	10	2024-04-18 11:08:04.236
2	sepang	Sepang International Circuit	Kuala Lumpur	Malaysia	2.76083	101.738	18	2024-04-18 11:08:04.236
3	bahrain	Bahrain International Circuit	Sakhir	Bahrain	26.032	50.5106	7	2024-04-18 11:08:04.236
4	catalunya	Circuit de Barcelona-Catalunya	Montmeló	Spain	41.57	2.2611	109	2024-04-18 11:08:04.236
5	istanbul	Istanbul Park	Istanbul	Turkey	40.9517	29.405	130	2024-04-18 11:08:04.236
6	monaco	Circuit de Monaco	Monte-Carlo	Monaco	43.7347	7.42056	7	2024-04-18 11:08:04.236
7	villeneuve	Circuit Gilles Villeneuve	Montreal	Canada	45.5	-73.5228	13	2024-04-18 11:08:04.236
8	monza	Circuito de Nuvolari-Magny-Cours	Monza	Italy	46.0642	12.1621	128	2024-04-18 11:08:04.236

9:45 AM 4/23/2024

Databricks Notebook: transformation_overall_results (Python)

```

1 races_df = spark.read.parquet("dbfs:/FileStore/Formula1/processed/f1_processed_races") \
2 .withColumnRenamed("name", "race_name") \
3 .withColumnRenamed("race_timestamp", "race_date")

```

(1) Spark Jobs

```

1 races_df.show(truncate=False)

```

(1) Spark Jobs

race_id	race_year	round	circuit_id	race_name	ingestion_date	race_date
1	2009	1	1	Australian Grand Prix	2024-04-23 04:06:31.534	2009-03-29 06:00:00
2	2009	2	2	Malaysian Grand Prix	2024-04-23 04:06:31.534	2009-04-05 09:00:00
3	2009	3	17	Chinese Grand Prix	2024-04-23 04:06:31.534	2009-04-19 07:00:00
4	2009	4	3	Bahrain Grand Prix	2024-04-23 04:06:31.534	2009-04-26 12:00:00
5	2009	5	4	Spanish Grand Prix	2024-04-23 04:06:31.534	2009-05-10 12:00:00

Databricks Notebook: transformation_overall_results (Python)

```

1 results_df = spark.read.parquet("dbfs:/FileStore/Formula1/processed/f1_processed.result") \
2 .withColumnRenamed("time", "race_time") \
3 .withColumnRenamed("race_id", "result_race_id") \
4 .withColumnRenamed("file_date", "result_file_date")

```

(1) Spark Jobs

```

1 results_df.show(truncate=False)

```

(1) Spark Jobs

result_id	result_race_id	driver_id	constructor_id	number	grid	position	position_text	position_order	points	laps	race_time	milliseconds	fastest_lap	rank	fastest_lap_time	fastest_lap_speed	ingestion_date
7572	1	5	1	2	12	null	R	19	0.0	0	\N	null	null	\N	null	2024-04-22 07:21:10.884	

databricks

transformation_overall_results Python

File Edit View Run Help Last edit was 4 minutes ago New cell UI: OFF

Last command failed

Run all Compute Share Publish

```
1 #Join circuits to races
```

Command took 0.09 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:44:41 AM on compute

Cmd 13

```
1 race_circuits_df = races_df.join(circuits_df, races_df.circuit_id == circuits_df.circuit_id, "inner") \
2 .select(races_df.race_id, races_df.race_year, races_df.race_name, races_df.race_date, circuits_df.circuit_location)
```

race_circuits_df: pyspark.sql.dataframe.DataFrame = [race_id: integer, race_year: integer ... 3 more fields]

Command took 0.19 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:44:41 AM on compute

Cmd 14

```
1 race_circuits_df.show(truncate=False)
```

(2) Spark Jobs

race_id	race_year	race_name	race_date	circuit_location
1	2009	Australian Grand Prix	2009-03-29 06:00:00	Melbourne
2	2009	Malaysian Grand Prix	2009-04-05 09:00:00	Kuala Lumpur
3	2009	Chinese Grand Prix	2009-04-19 07:00:00	Shanghai

9:46 AM 4/23/2024

databricks

transformation_overall_results Python

File Edit View Run Help Last edit was 4 minutes ago New cell UI: OFF

Last command failed

Run all Compute Share Publish

```
1 #Join results to all other dataframes
```

Command took 0.09 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:44:41 AM on compute

Cmd 16

```
1 race_results_df = results_df.join(race_circuits_df, results_df.result_race_id == race_circuits_df.race_id) \
2 .join(drivers_df, results_df.driver_id == drivers_df.driver_id) \
3 .join(constructors_df, results_df.constructor_id == constructors_df.constructor_id)
```

race_results_df: pyspark.sql.dataframe.DataFrame = [result_id: integer, result_race_id: integer ... 34 more fields]

Command took 0.20 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:44:41 AM on compute

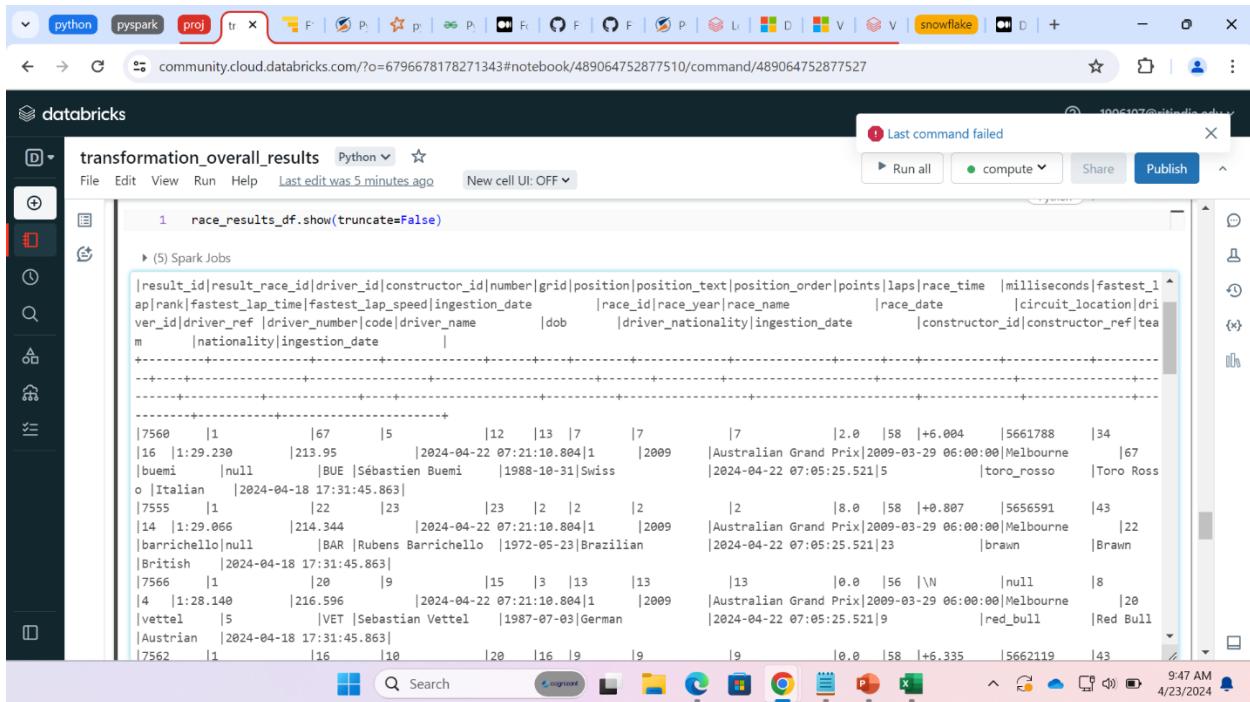
Cmd 17

```
1 race_results_df.show(truncate=False)
```

(5) Spark Jobs

result_id	result_race_id	driver_id	constructor_id	number	grid	position	position_text	position_order	points	laps	race_time	milliseconds	fastest_lap_ap	rank	fastest_lap_time	fastest_lap_speed	ingestion_date	race_id	race_year	race_name	race_date	circuit_location	driver_id	driver_ref	driver_number	code	driver_name	dob	driver_nationality	ingestion_date	constructor_id	constructor_ref	team	nationality	ingestion_date
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

9:47 AM 4/23/2024



Databricks Notebook titled "transformation_overall_results" in Python mode.

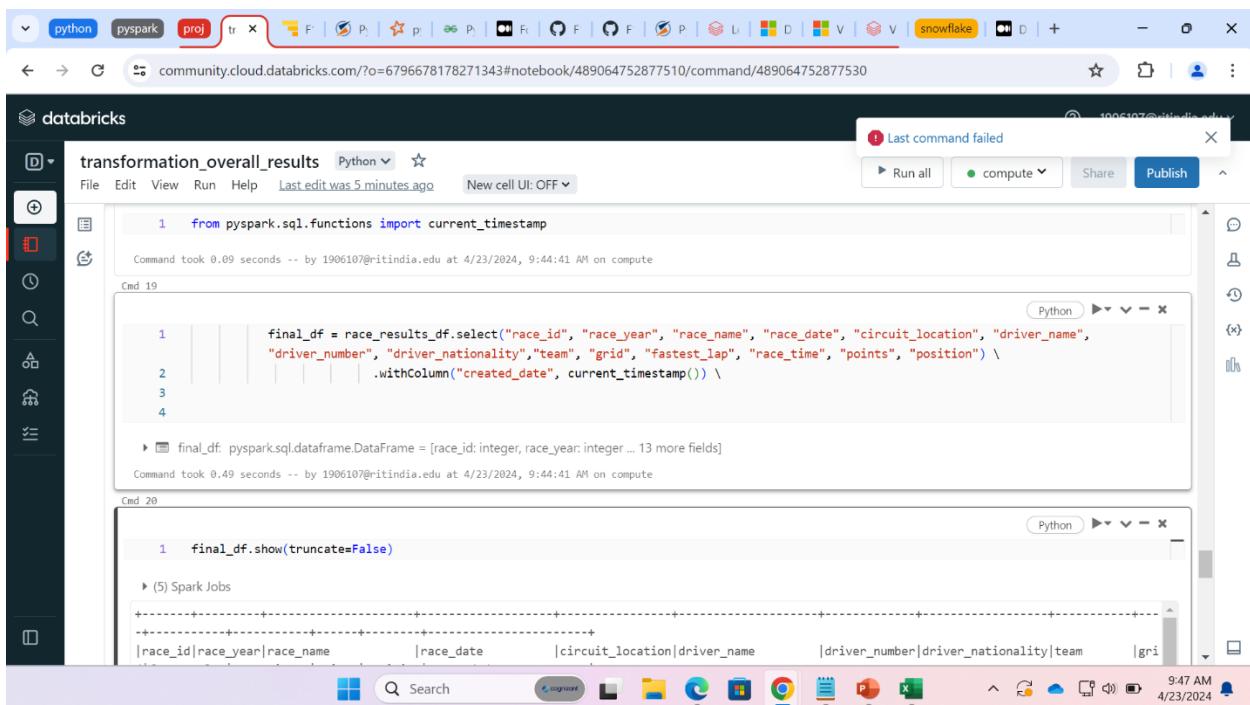
```

1 race_results_df.show(truncate=False)

```

(5) Spark Jobs

race_id	race_year	driver_id	constructor_id	number	grid	position	position_text	position_order	points	laps	race_time	milliseconds	fastest_lap	rank	fastest_lap_time	fastest_lap_speed	ingestion_date	race_id	race_year	race_name	race_date	circuit_location	driver_id	constructor_ref	team	nationality	ingestion_date			
7560	1	67	5	12	13	7	7	7	2.0	58	+6.004	5661788	34							Australian Grand Prix	2009-03-29	06:00:00	Melbourne							
16	1:29.230	213.95		2024-04-22 07:21:10.804	1	2009									16	213.95	2024-04-22 07:05:25.521	5												
buejni	null	BUE	Sébastien Buemi	1988-10-31	Swiss																									
o	Italian	2024-04-18 17:31:45.863																												
7555	1	22	23	23	2	2	2	2	8.0	58	+0.807	5656591	43							Australian Grand Prix	2009-03-29	06:00:00	Melbourne							
14	1:29.066	214.344		2024-04-22 07:21:10.804	1	2009									14	214.344	2024-04-22 07:05:25.521	23												
barrichello	null	BAR	Rubens Barrichello	1972-05-23	Brazilian																									
British	2024-04-18 17:31:45.863																													
7566	1	20	9	15	3	13	13	13	0.0	56	\N	null	8							Australian Grand Prix	2009-03-29	06:00:00	Melbourne							
4	1:28.100	216.596		2024-04-22 07:21:10.804	1	2009									4	216.596	2024-04-22 07:05:25.521	9												
vettel	5	VET	Sebastian Vettel	1987-07-03	German																									
Austrian	2024-04-18 17:31:45.863																													
7562	1	16	10	10	16	9	9	9	0.0	58	+6.335	5662119	43							Austrian	2024-04-18 17:31:45.863									



Databricks Notebook titled "transformation_overall_results" in Python mode.

```

1 from pyspark.sql.functions import current_timestamp

```

Command took 0.09 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:44:41 AM on compute

Cmd 19

```

1 final_df = race_results_df.select("race_id", "race_year", "race_name", "race_date", "circuit_location", "driver_name",
2                                 "driver_number", "driver_nationality", "team", "grid", "fastest_lap", "race_time", "points", "position") \
3                                 .withColumn("created_date", current_timestamp())

```

final_df: pyspark.sql.dataframe.DataFrame = [race_id: integer, race_year: integer ... 13 more fields]

Command took 0.49 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:44:41 AM on compute

Cmd 20

```

1 final_df.show(truncate=False)

```

(5) Spark Jobs

race_id	race_year	race_name	race_date	circuit_location	driver_name	driver_number	driver_nationality	team	grid
7560	1	Australian Grand Prix	2009-03-29 06:00:00	Melbourne	Sébastien Buemi	67	Swiss	Toro Rosso	34
16	1:29.230	Australian Grand Prix	2009-03-29 06:00:00	Melbourne	Rubens Barrichello	22	Brazilian	Brawn	43
buejni	null	Australian Grand Prix	2009-03-29 06:00:00	Melbourne	Sebastian Vettel	20	German	Red Bull	43
o	Italian	Australian Grand Prix	2009-03-29 06:00:00	Melbourne					8
7555	1	Australian Grand Prix	2009-03-29 06:00:00	Melbourne					22
14	1:29.066	Australian Grand Prix	2009-03-29 06:00:00	Melbourne					20
barrichello	null	Australian Grand Prix	2009-03-29 06:00:00	Melbourne					18
British	2024-04-18 17:31:45.863	Australian Grand Prix	2009-03-29 06:00:00	Melbourne					12
7566	1	Australian Grand Prix	2009-03-29 06:00:00	Melbourne					10
4	1:28.100	Australian Grand Prix	2009-03-29 06:00:00	Melbourne					10
vettel	5	Australian Grand Prix	2009-03-29 06:00:00	Melbourne					9
Austrian	2024-04-18 17:31:45.863	Australian Grand Prix	2009-03-29 06:00:00	Melbourne					9
7562	1	Australian Grand Prix	2009-03-29 06:00:00	Melbourne					13

python pyspark proj tr x F P p F P F P F L D V V snowflake D +

community.cloud.databricks.com/?o=6796678178271343#notebook/489064752877510/command/489064752877530

databricks

transformation_overall_results Python OFF

Last command failed

Run all compute Share Publish

```
final_df.show(truncate=False)
```

(5) Spark Jobs

race_id	race_year	race_name	race_date	circuit_location	driver_name	driver_number	driver_nationality	team	grid
1	2009	Australian Grand Prix	2009-03-29 06:00:00	Melbourne	Sébastien Buemi	null	Swiss	Toro Rosso	13
34	+6.004	[2.0]	[7]	[2024-04-23 04:14:56.935]					
1	2009	Australian Grand Prix	2009-03-29 06:00:00	Melbourne	Rubens Barrichello	null	Brazilian	Brawn	2
43	+0.807	[8.0]	[2]	[2024-04-23 04:14:56.935]					
1	2009	Australian Grand Prix	2009-03-29 06:00:00	Melbourne	Sebastian Vettel	5	German	Red Bull	3
8	\N	[0.0]	[13]	[2024-04-23 04:14:56.935]					
1	2009	Australian Grand Prix	2009-03-29 06:00:00	Melbourne	Adrian Sutil	99	German	Force India	16
43	+6.335	[0.0]	[9]	[2024-04-23 04:14:56.935]					
1	2009	Australian Grand Prix	2009-03-29 06:00:00	Melbourne	Robert Kubica	88	Polish	BMW Sauber	4
36	\N	[0.0]	[14]	[2024-04-23 04:14:56.935]					
1	2009	Australian Grand Prix	2009-03-29 06:00:00	Melbourne	Kimi Räikkönen	7	Finnish	Ferrari	7
35	\N	[0.0]	[15]	[2024-04-23 04:14:56.935]					
1	2009	Australian Grand Prix	2009-03-29 06:00:00	Melbourne	Sébastien Bourdais	null	French	Toro Rosso	17

9:47 AM 4/23/2024

python pyspark proj tr x F P p F P F P F L D V V snowflake D +

community.cloud.databricks.com/?o=6796678178271343#notebook/489064752877510/command/489064752877534

databricks

transformation_overall_results Python OFF

1906107@ritindia.edu

Run all compute Share Publish

```
dbutils.fs.mkdirs("dbfs:/FileStore/Formula1/trans_before_result")
```

Out[49]: True

```
final_df.write.parquet("dbfs:/FileStore/Formula1/trans_before_result/file_for_standing")
```

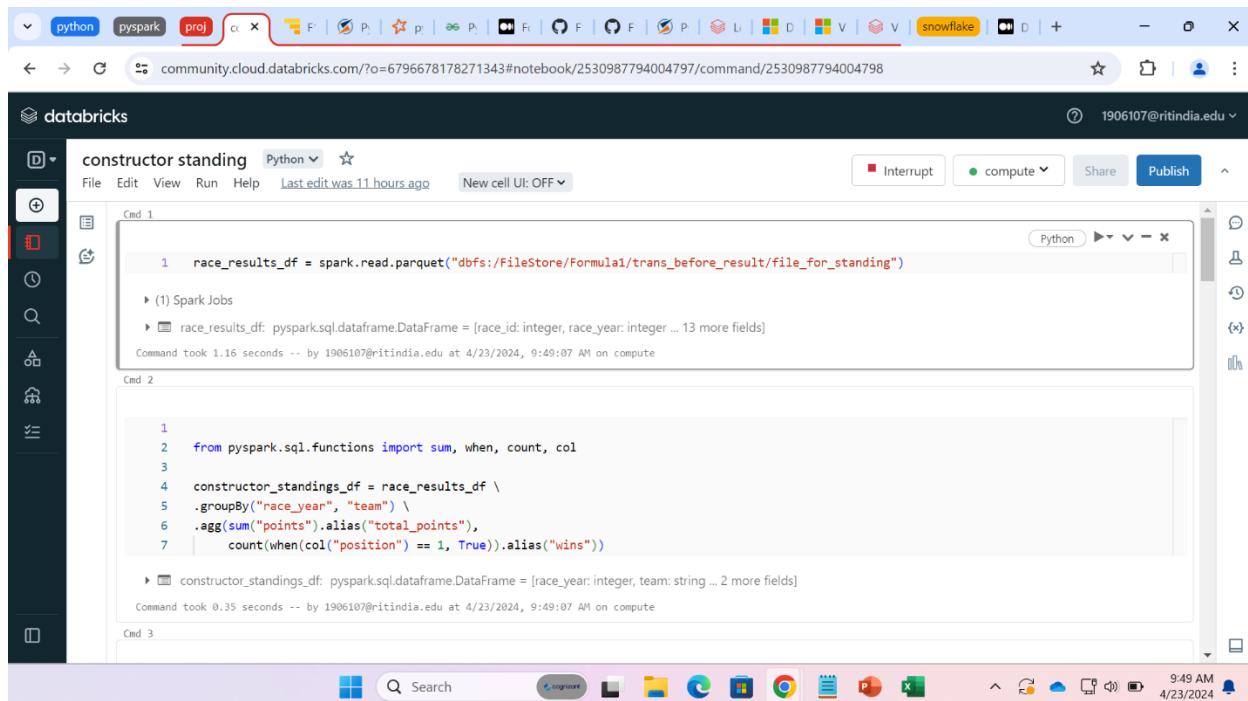
(4) Spark Jobs

AnalysisException: Path dbfs:/FileStore/Formula1/trans_before_result/file_for_standing already exists.

Command took 4.28 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:44:41 AM on compute

9:48 AM 4/23/2024

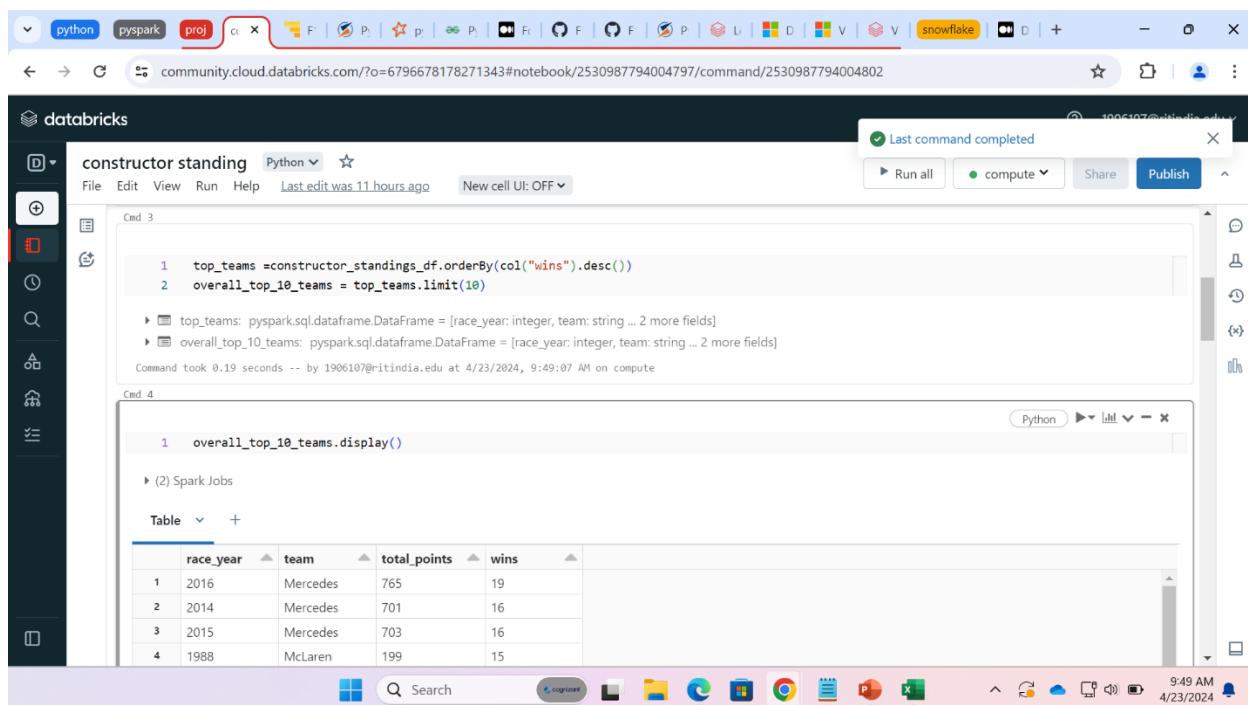
Constructor standing and dominant team.



The screenshot shows a Databricks notebook titled "constructor standing" in Python. The notebook has three command cells:

- Cmd 1:** `race_results_df = spark.read.parquet("dbfs:/FileStore/Formula1/trans_before_result/file_for_standing")`
- Cmd 2:** A more complex SQL-like code block that groups by race year and team, then aggregates points and counts wins. It creates a DataFrame named `constructor_standings_df`.
- Cmd 3:** Not yet run.

The status bar at the bottom indicates the last edit was 11 hours ago, and the command took 1.16 seconds to run.



The screenshot shows the same Databricks notebook after the third command has been run. A success message "Last command completed" is visible. The notebook now has four command cells:

- Cmd 3:** Shows the results of the previous command: `overall_top_10_teams = top_teams.limit(10)`
- Cmd 4:** `overall_top_10_teams.display()`
- Cmd 5:** (2) Spark Jobs
- Cmd 6:** Not yet run.

A modal window displays the resulting table:

	race_year	team	total_points	wins
1	2016	Mercedes	765	19
2	2014	Mercedes	701	16
3	2015	Mercedes	703	16
4	1988	McLaren	199	15

#que 2 constructor standing

```

1 from pyspark.sql.window import Window
2 from pyspark.sql.functions import desc, rank, asc
3
4 constructor_rank_spec = Window.partitionBy("race_year").orderBy(desc("total_points"), desc("wins"))
5 final_df = constructor_standings_df.withColumn("rank", rank().over(constructor_rank_spec))

```

final_df: pyspark.sql.dataframe.DataFrame = [race_year: integer, team: string ... 3 more fields]

final_df.display()

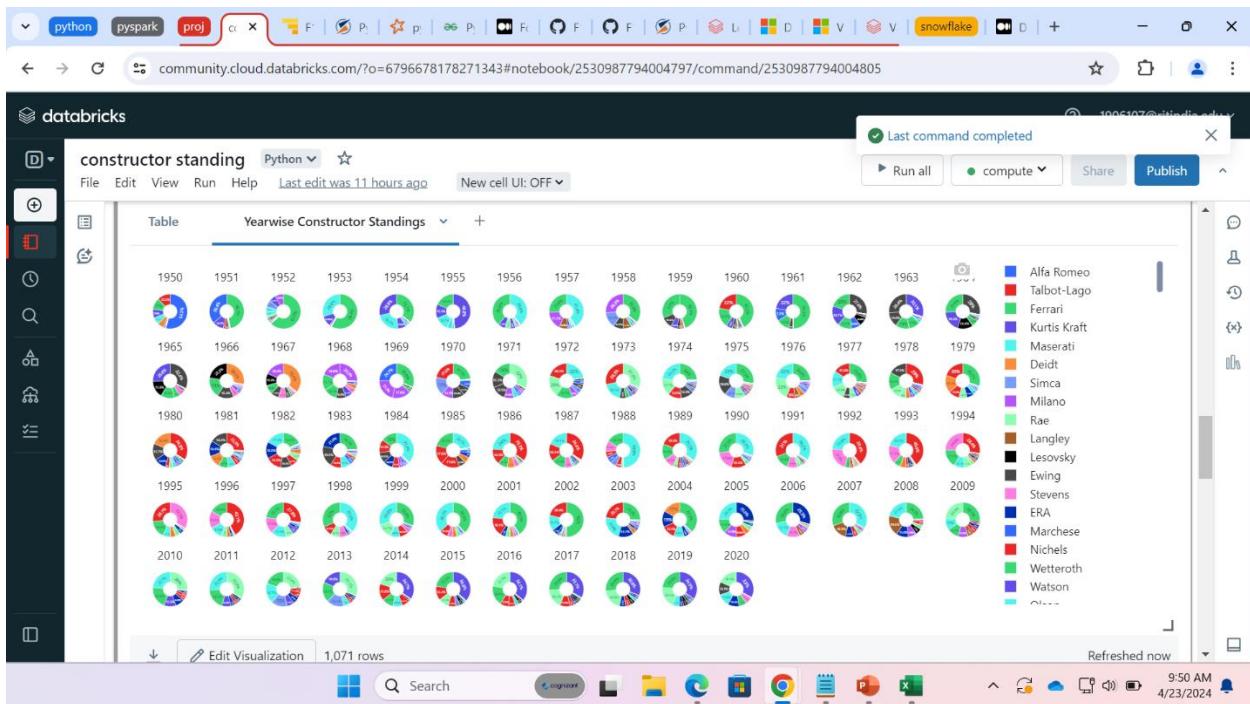
(3) Spark Jobs

Table Yearwise Constructor Standings

	race_year	team	total_points	wins	rank
1	1950	Alfa Romeo	89	6	1
2	1950	Talbot-Lago	20	0	2
3	1950	Ferrari	18	0	3
4	1950	Kurtis Kraft	13	1	4
5	1950	Maserati	11	0	5
6	1950	Deidt	10	0	6
7	1950	Simca	3	0	7

1,071 rows | 2.20 seconds runtime

Refreshed now



The screenshot shows a Databricks notebook titled "constructor standing" in Python. The notebook contains two cells. The first cell, "Cmd 8", contains the following Python code:

```
1 final_df.createTempView("constructors")
```

Command took 0.38 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:49:07 AM on compute

The second cell, "Cmd 9", contains the following Python code:

```
1 spark.sql("select* FROM constructors WHERE race_year = 2020").display()
```

▶ (3) Spark Jobs

Table Dominant team 2020 +

SUM(total_points)

33%
10.4%
7.55%
6.17%

Legend: Mercedes, Red Bull, Racing Point, McLaren, Renault, Ferrari, AlphaTauri, Alfa Romeo

python pyspark proj

community.cloud.databricks.com/?o=6796678178271343#notebook/2530987794004797/command/2530987794004805

databricks

constructor standing Python

File Edit View Run Help Last edit was 11 hours ago New cell UI: OFF

Run all compute Share Publish

Command took 0.38 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:49:07 AM on compute

Cmd 9

```
1 spark.sql("select* FROM constructors WHERE race_year = 2020").display()
```

(3) Spark Jobs

Table Dominant team 2020 +

	race_year	team	total_points	wins	rank
1	2020	Mercedes	573	13	1
2	2020	Red Bull	319	2	2
3	2020	Racing Point	210	1	3
4	2020	McLaren	202	0	4
5	2020	Renault	181	0	5
6	2020	Ferrari	131	0	6
7	2020	AlphaTauri	107	1	7

10 rows | 1.79 seconds runtime Refreshed 1 minute ago

💡 1

9:50 AM 4/23/2024

python pyspark proj

community.cloud.databricks.com/?o=6796678178271343#notebook/2530987794004797/command/2530987794004805

databricks

constructor standing Python

File Edit View Run Help Last edit was 11 hours ago New cell UI: OFF

Run all compute Share Publish

Table Dominant team 2020 +

Team	Percentage of Total Points
Mercedes	33%
Red Bull	18.4%
Racing Point	12.1%
McLaren	11.6%
Renault	10.4%
Ferrari	7.55%
AlphaTauri	6.17%

Legend:

- Mercedes
- Red Bull
- Racing Point
- McLaren
- Renault
- Ferrari
- AlphaTauri
- Alfa Romeo
- Haas F1 Team
- Williams

Edit Visualization 10 rows Refreshed 2 minutes ago

9:50 AM 4/23/2024

Driver standing and dominant driver.

Databricks Notebook - driver standing (Python)

```

1 race_results_df = spark.read.parquet("dbfs:/FileStore/Formula1/trans_before_result/file_for_standing")

```

(1) Spark Jobs

```

race_results_df: pyspark.sql.DataFrame = [race_id: integer, race_year: integer ... 13 more fields]

```

Command took 1.23 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:51:58 AM on compute

Cmd 2

```

1 race_results_df.show(truncate=False)

```

(1) Spark Jobs

151	+7.374	0.0	11	2024-04-22 10:04:26.2									
1	2009	Australian Grand Prix 2009-03-29 06:00:00 Melbourne			Jenson Button	22	British	Brawn	1				
17	1:34:15.784 10.0	1		2024-04-22 10:04:26.2									
1	2009	Australian Grand Prix 2009-03-29 06:00:00 Melbourne			Mark Webber	null	Australian	Red Bull	8				
38	N	0.0	12	2024-04-22 10:04:26.2									
1	2009	Australian Grand Prix 2009-03-29 06:00:00 Melbourne			Jarno Trulli	null	Italian	Toyota	20				
50	+1.604	6.0	3	2024-04-22 10:04:26.2									
1	2009	Australian Grand Prix 2009-03-29 06:00:00 Melbourne			Felipe Massa	19	Brazilian	Ferrari	6				
30	N	0.0	null	2024-04-22 10:04:26.2									
1	2009	Australian Grand Prix 2009-03-29 06:00:00 Melbourne			Nelson Piquet Jr.	null	Brazilian	Renault	14				
17	N	0.0	null	2024-04-22 10:04:26.2									
1	2009	Australian Grand Prix 2009-03-29 06:00:00 Melbourne			Timo Glock	null	German	Toyota	19				

9:52 AM 4/23/2024

Databricks Notebook - driver standing (Python)

```

1 #que1-- top 10 drivers from year 1950 to year 2020

```

Command took 0.07 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:51:58 AM on compute

Cmd 4

```

1 from pyspark.sql.functions import sum, when, count, col
2
3 driver_standings_df = race_results_df \
4 .groupBy("race_year", "driver_name", "driver_nationality") \
5 .agg(sum("points").alias("total_points"),
6 | count(when(col("position") == 1, True)).alias("wins"))

```

(1) Spark Jobs

```

driver_standings_df: pyspark.sql.DataFrame = [race_year: integer, driver_name: string ... 3 more fields]

```

Command took 0.29 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:51:58 AM on compute

Cmd 5

```

1 from pyspark.sql.functions import desc, asc
2 driver_standings_df.orderBy(driver_standings_df.wins.desc()).take(10)

```

(2) Spark Jobs

9:52 AM 4/23/2024

```

1   from pyspark.sql.functions import desc, asc
2   driver_standings_df.orderBy(driver_standings_df.wins.desc()).take(10)

▶ (2) Spark Jobs
Out[5]: [Row(race_year=2004, driver_name='Michael Schumacher', driver_nationality='German', total_points=148.0, wins=13),
Row(race_year=2013, driver_name='Sebastian Vettel', driver_nationality='German', total_points=397.0, wins=13),
Row(race_year=2011, driver_name='Sebastian Vettel', driver_nationality='German', total_points=392.0, wins=11),
Row(race_year=2019, driver_name='Lewis Hamilton', driver_nationality='British', total_points=413.0, wins=11),
Row(race_year=2014, driver_name='Lewis Hamilton', driver_nationality='British', total_points=384.0, wins=11),
Row(race_year=2002, driver_name='Michael Schumacher', driver_nationality='German', total_points=144.0, wins=11),
Row(race_year=2018, driver_name='Lewis Hamilton', driver_nationality='British', total_points=408.0, wins=11),
Row(race_year=2020, driver_name='Lewis Hamilton', driver_nationality='British', total_points=347.0, wins=11),
Row(race_year=2015, driver_name='Lewis Hamilton', driver_nationality='British', total_points=381.0, wins=10),
Row(race_year=2016, driver_name='Lewis Hamilton', driver_nationality='British', total_points=380.0, wins=10)]

Command took 2.40 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:51:58 AM on compute

```

Cmd 6

```

1   top_driver =driver_standings_df.orderBy(col("wins").desc())
2   over11_top_10_drivers = top_driver.limit(10)

▶ top_driver: pyspark.sql.dataframe.DataFrame = [race_year: integer, driver_name: string ... 3 more fields]
▶ over11_top_10_drivers: pyspark.sql.dataframe.DataFrame = [race_year: integer, driver_name: string ... 3 more fields]

```

```

1   top_driver =driver_standings_df.orderBy(col("wins").desc())
2   over11_top_10_drivers = top_driver.limit(10)

▶ top_driver: pyspark.sql.dataframe.DataFrame = [race_year: integer, driver_name: string ... 3 more fields]
▶ over11_top_10_drivers: pyspark.sql.dataframe.DataFrame = [race_year: integer, driver_name: string ... 3 more fields]

Command took 0.29 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:51:58 AM on compute

```

Cmd 7

```

1   over11_top_10_drivers.display()

```

Table

	race_year	driver_name	driver_nationality	total_points	wins
1	2004	Michael Schumacher	German	148	13
2	2013	Sebastian Vettel	German	397	13
3	2011	Sebastian Vettel	German	392	11
4	2019	Lewis Hamilton	British	413	11
5	2014	Lewis Hamilton	British	384	11

driver standing Python

```

1 #que 2 driver standing

```

Command took 0.08 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:51:58 AM on compute

```

Cmd 9
1 from pyspark.sql.window import Window
2 from pyspark.sql.functions import desc, rank, asc
3
4 driver_rank_spec = Window.partitionBy("race_year").orderBy(desc("total_points"), desc("wins"))
5 final_df = driver_standings_df.withColumn("rank", rank().over(driver_rank_spec))

```

final_df: pyspark.sql.dataframe.DataFrame = [race_year: integer, driver_name: string ... 4 more fields]

Command took 0.29 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:51:58 AM on compute

```

Cmd 10
1 final_df.display()

```

(3) Spark Jobs

Table Yearwise Driver Standing +

driver standing Python

```

1 final_df.display()

```

(3) Spark Jobs

Table Yearwise Driver Standing +

	race_year	driver_name	driver_nationality	total_points	wins	rank
1	1950	Nino Farina	Italian	30	3	1
2	1950	Luigi Fagioli	Italian	28	0	2
3	1950	Juan Fangio	Argentine	27	3	3
4	1950	Louis Rosier	French	13	0	4
5	1950	Johnnie Parsons	American	9	1	5
6	1950	Alberto Ascari	Italian	8	0	6
7	1950	Rill Hilliard	American	6	0	7

3,122 rows | 2.70 seconds runtime

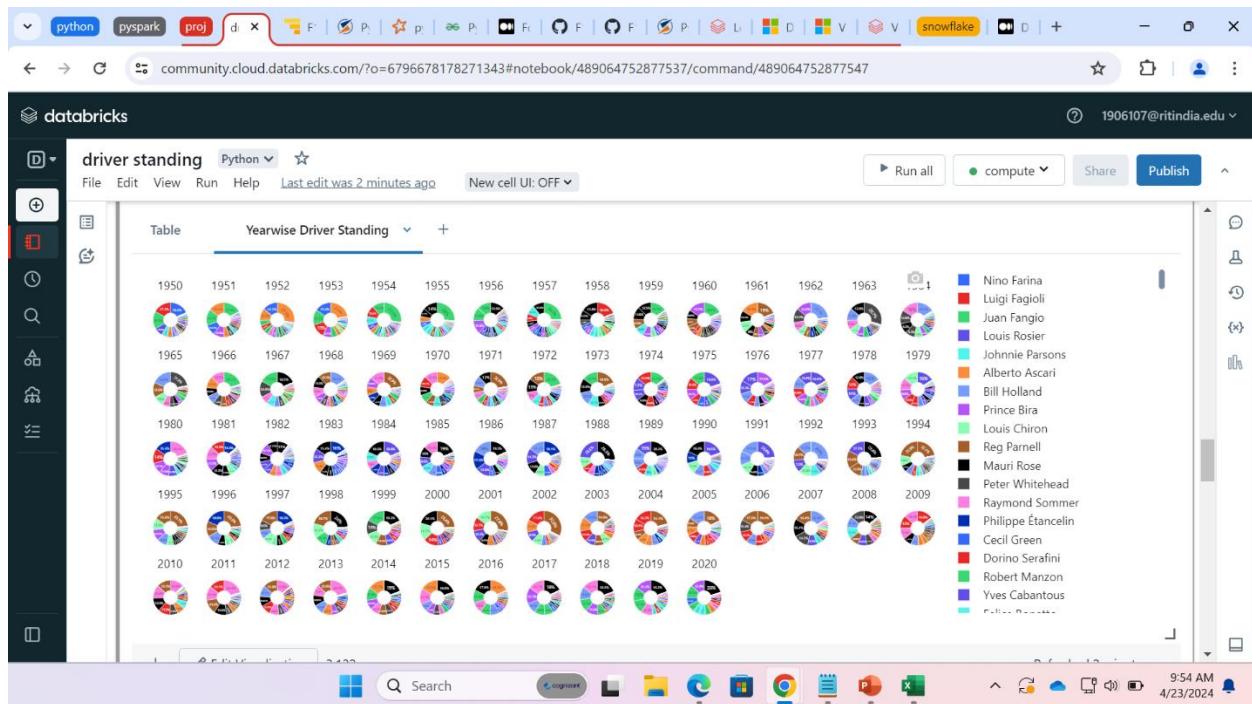
Refreshed 1 minute ago

Command took 2.70 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:51:58 AM on compute

```

Cmd 11

```



```

1 final_df.createTempView("trial")
Command took 0.17 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:51:58 AM on compute

```

```

1 spark.sql("select* FROM trial WHERE race_year = 2020").show()

```

race_year	driver_name	driver_nationality	total_points	wins	rank
2020	Lewis Hamilton	British	347.0	11	1
2020	Valterri Bottas	Finnish	223.0	2	2
2020	Max Verstappen	Dutch	214.0	2	3
2020	Sergio Pérez	Mexican	125.0	1	4
2020	Daniel Ricciardo	Australian	119.0	0	5
2020	Carlos Sainz	Spanish	105.0	0	6
2020	Alexander Albon	Thai	105.0	0	6
2020	Charles Leclerc	Monegasque	98.0	0	8
2020	Lando Norris	British	97.0	0	9
2020	Pierre Gasly	French	75.0	1	10
2020	Lance Stroll	Canadian	75.0	0	11

driver standing Python

```
1 spark.sql("select* FROM trial WHERE race_year = 2020").display()
```

(3) Spark Jobs

Table Dominant Drivers in 2020

	race_year	driver_name	driver_nationality	total_points	wins	rank
1	2020	Lewis Hamilton	British	347	11	1
2	2020	Valtteri Bottas	Finnish	223	2	2
3	2020	Max Verstappen	Dutch	214	2	3
4	2020	Sergio Pérez	Mexican	125	1	4
5	2020	Daniel Ricciardo	Australian	119	0	5
6	2020	Carlos Sainz	Spanish	105	0	6
7	2020	Alexander Albon	Thai	105	0	6

23 rows | 1.27 seconds runtime

Refreshed 2 minutes ago

💡 Command took 1.27 seconds -- by 1906107@ritindia.edu at 4/23/2024, 9:51:58 AM on compute

driver standing Python

```
AVG(total_points)
```

23 rows

Refreshed 3 minutes ago

CONCLUSION

- **Summarize Key Findings:** State your most important observations about driver standings, constructor standings, and top-performing teams. Did the results align with expectations? Were there any surprises? Highlight these.
- **Trends:** Discuss significant patterns in the data. Did individual drivers or constructors dominate? Were there upward or downward trends in performance over seasons?
- **Factors of Success:** If possible, try to pinpoint factors leading to success. This could be driver skill consistency, team strategy, car technological advantages, etc.

Potential Insights (Examples)

- **Driver Standings:**
 - "The analysis clearly established Driver X as the dominant force of the season(s), consistently securing top positions and amassing the most points."
 - "An unexpected finding was the rise of Driver Y, a relative newcomer who outperformed several veterans, indicating a promising future."
- **Constructor Standings:**
 - "Constructor Z showcased consistently strong race results across the board, suggesting superior car development and strategic race management."
 - "An interesting trend was the decline of Constructor W, which, despite a history of success, was unable to adapt to [regulation changes, new technologies, etc.]"
- **Top Teams:**
 - "The pairing of Driver A and Driver B within Constructor C created a formidable team, as their combined strengths led to numerous victories."
 - "Several mid-tier teams demonstrated impressive improvement, potentially closing the performance gap with the leading teams, bringing the prospect of a more competitive field in future seasons."

Additional Considerations

- **Data Storytelling:** Craft your conclusion as a short narrative, not just a list of facts. Tie your findings back to the larger story of F1.
- **Future Analysis:** If your data and analysis allow, suggest potential future areas to explore, like driver performance under specific track conditions or the performance impact of in-race incidents.
- **Limitations:** Acknowledge any limitations of your dataset or analysis methods. This demonstrates transparency and opens up the possibility of deeper analysis in the future.