Pontifícia Universidade Católica
do Rio de Janeiro

Bruno Renaux Mendes Reis

MVP – Data Engineering:

Data Pipeline for Marketing Analytics

Pós-graduação em Ciência de Dados e Analytics

Teacher: Victor Teixeira de Almeida

Monitor: Silvio Alonso Marques

Rio de Janeiro

December 2025

SUMMARY

1. Introduction

This work was developed as an MVP (Minimum Viable Project) for my postgraduate program in Data Science and Analytics, more specifically for the Data Engineering course. It will remain available in my personal GitHub repository as part of my data analyst portfolio.

The objective of the project was to build a data pipeline for storing and processing data in a cloud environment, using the Databricks platform. Additionally, the pipeline was designed to follow a data warehouse architecture, featuring three layers — bronze, silver, and gold — as well as a star schema for data modeling.

1.1 Business Problems

The goal of this data pipeline is addressing the business needs of a hypothetical food retail company. In particular, the objective is to understand both the characteristics of the company's customer base and the effectiveness of past promotional campaigns targeted at those customers.

To achieve this, I used a dataset publicly released on GitHub by the iFood Brain Team as part of a hiring process and later shared on Kaggle, where it was obtained. The business problems addressed in this project can be summarized by the following questions:

a) Which features of the customer base are the most significant is maximizing expenditure?

Naturally, a company's primary objective is to increase revenue. Therefore, the first question posed to the data concerns which customer characteristics are most relevant to higher spending levels. Since most of these features are categorical (textual), customers can be grouped by category and compared based on their average spending.

b) Which features of the customer base are most relevant to the effectiveness of promotional campaigns?

In a similar vein, the same customer features can be analyzed to ascertain the impact of the promotional campaigns featured in the data.

c) Which are the best performing products?

We take a brake from investigating the customer data to focus on the different product categories. It is prudent to understand from the data which products are the ones that bring about the most revenue to the company.

d) Which is the most popular method of purchasing?

The last question pertains to the different channels of purchase available. Understanding which channels generate the highest revenue is essential for strategic decision-making.

1.2 Process

The construction of the data pipeline will be realized within the Databricks platform for cloud-based data processing. Through notebooks and queries written inside the platform itself, featuring Python and SQL languages, I will both model a simulated data warehouse for storing the data, and will perform a brief data analysis seeking to answer the previously posed questions.

All of the scripts and results of this work will be stored in a GitHub folder connected to said environment [Link].

2. The Data

The data was initially found in the website Kaggle, customarily used for sharing datasets and Python notebooks, in .csv format, but it originally hails from a GitHub repository created as part of a hiring process for iFood, a Brazilian food delivery service seeking business data analysts.

The dataset is artificial, and the company described in the data does not exist. Nevertheless, since it was specifically designed to evaluate analytical skills for iFood's Brain Team, it was deemed suitable for inclusion in a portfolio project focused on marketing data analytics.

I will not, however, follow the exact instructions found on the GitHub folder itself, as this project was developed for a Data Engineering class, and thus required a different approach.

2.1. Description

The dataset *ifood_df.csv* consists of 2240 customers of hypothetical 'XYZ Company' with data on customer profiles, product preferences and promotional campaign successes/failures.

The company is described as operating in the food retail sector, selling products from five major categories: wines, rare meat products, exotic fruits, specially prepared fish and sweet products. These products are further divided into regular and *gold* categories. Furthermore, there are three possible sales channels available for customers: physical stores, catalogs and the company's website.

Due to declining profitability, the company has implemented several initiatives in order to reverse this predicament, among which is a series of six promotional campaigns, depicted in the dataset.

2.2. Features

In this section the features of the dataset will be listed and described as they were on the original iFood case used as basis for this work. They are, as follows:

1. AcceptedCmp1: 1 if costumer accepted the offer in the 1st campaign, 0 otherwise
2. AcceptedCmp2: 1 if costumer accepted the offer in the 2nd campaign, 0 otherwise
3. AcceptedCmp3: 1 if costumer accepted the offer in the 3rd campaign, 0 otherwise
4. AcceptedCmp4: 1 if costumer accepted the offer in the 4th campaign, 0 otherwise
5. AcceptedCmp5: 1 if costumer accepted the offer in the 5th campaign, 0 otherwise
6. Response (target): 1 if costumer accepted the offer in the last campaign, 0 otherwise
7. Complain: 1 if customer complained in the last 2 years
8. DtCustomer: date of customer's enrollment with the company
9. Education: customer's level of education

10. Marital: customer's marital status

11. Kidhome: number of small children in customer's household

12. Teenhome: number of teenagers in customer's household

13. Income: customer's yearly household income

14. MntFishProducts: amount spent on fish products in the last 2 years

15. MntMeatProducts: amount spent on meat products in the last 2 years

16. MntFruits: amount spent on fruits in the last 2 years

17. MntSweetProducts: amount spent on sweet products in the last 2 years

18. MntWines: amount spent on wines in the last 2 years

19. MntGoldProds: amount spent on *gold* products in the last 2 years

20. NumDealsPurchases: number of purchases made with discount

21. NumCatalogPurchases: number of purchases made using catalogue

22. NumStorePurchases: number of purchases made directly in stores

23. NumWebPurchases: number of purchases made through company's website

24. NumWebVisitsMonth: number of visits to company's website last month

25. Recency: number of days since last purchase

3. Modelling

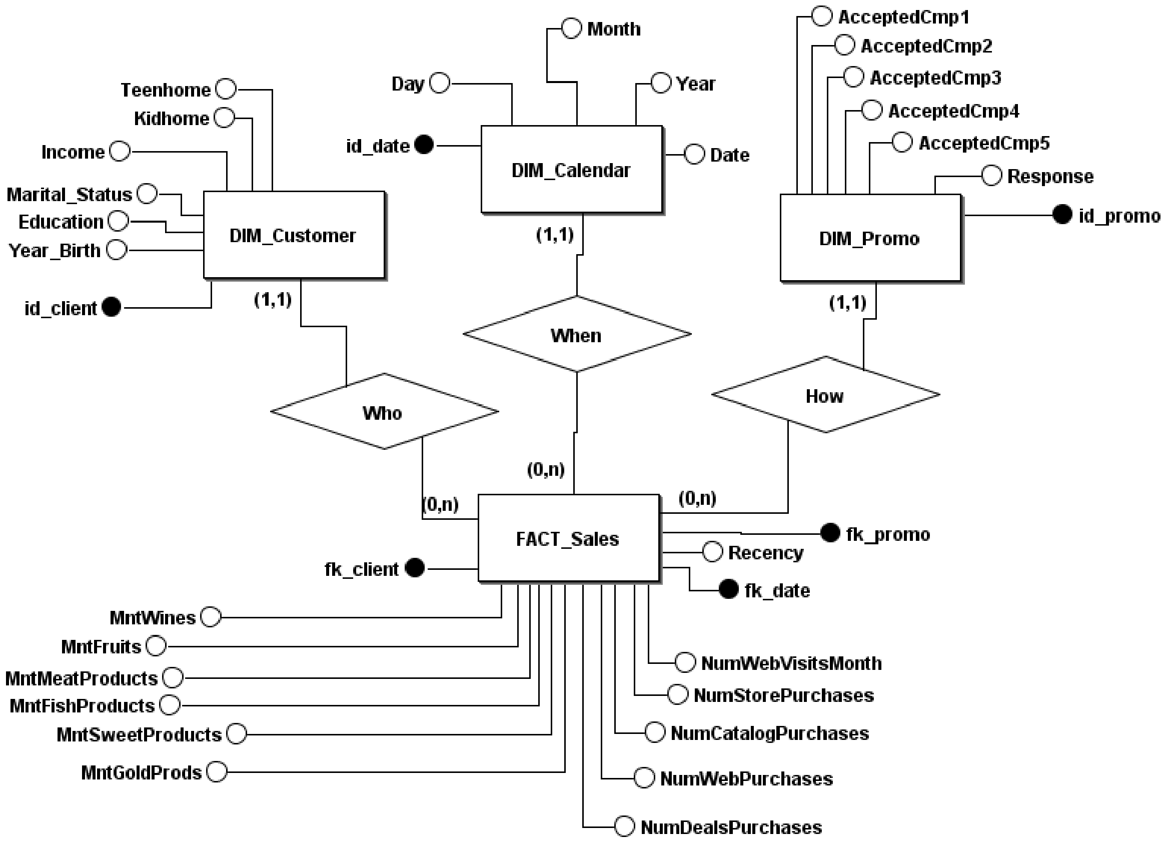
3.1. Data Modelling

To simulate a data warehouse as closely as possible, the dataset was modeled using a *star schema*. A star schema consists of a central fact table containing quantitative business metrics, which references multiple dimension tables holding descriptive attributes.

In this case, we will have a Sales fact table and for dimensional tables we shall have tables for customers, the promotional campaigns and a calendar table related to time:

- FACT_Sales: stores quantitative data such as the amount spent on products in each category and the number of sales through each method.
- DIM_Customer: contains dimensional data related to the customers individual characteristics.

- DIM_Promo: contains dimensional data on adherence to the promotional campaigns.
- DIM_Calendar: contains various date metrics and relates to the customer data through each one's enrollment with the company.



3.2. Data Catalog

The following is a detailed description of the data and its restrictions, imposed in the name of data integrity. This reflects the modeling done only at the silver layer, sometimes referred to as the 'cleaned layer', for the data in the bronze layer is intentionally raw and the data in the gold layer will be transformed into a format that more clearly communicates the data insights through summarized metrics (which shall be described next).

The following is a detailed description of each table and its components. Alongside the data type in parenthesis, you will find any additional restrictions, such as primary or foreign keys, the domains of the categorical variables (the possible categories) and other relevant information on the columns.

A. Fact Table “silver.fact_sale”
   1. id_sale (Bigint): Primary Key.
   2. fk_customer (BigInt): Foreign Key referencing the customer dimensional table.
   3. fk_promo (BigInt): Foreign Key referencing the promotional campaign dimensional table.
   4. fk_calendar (BigInt): Foreign Key referencing the calendar dimensional table.
   5. MntWines (Integer): amount spent on wines.
   6. MntFruits (Integer): amount spent on fruits.
   7. MntMeatProducts (Integer): amount spent on meat products.
   8. MntFishProducts (Integer): amount spent on fish products.
   9. MntSweetProducts (Integer): amount spent on sweet products.
   10. MntGold (Integer): amount spent on *gold* products.
   11. NumDealsPurchases (Integer): purchases with discounts.
   12. NumWebPurchases (Integer): purchases through website.
   13. NumCatalogPurchases (Integer): purchases through catalog.
   14. NumStorePurchases (Integer): purchases in physical stores.
   15. NumWebVisitsMonth (Integer): visits to the company’s website last month.
   16. Recency (Integer): days since last purchase
B. Dimension Table “silver.DIM_Customer”
   1. id_customer (Bigint): Primary Key.
   2. Year_Birth (Integer): customer’s birth.
   3. Education (String): Basic, Graduation, 2nd Cycle, Master, PhD
   4. Marital_Status (String): Single, Together, Divorced, Married, Widow, Unknown
   5. Kidhome (Integer): number of children in household.
   6. Teenhome (Integer): number of teens in household.
   7. Income (Double): customer’s yearly income.
   8. Complain: 1 if customer complained in the last 2 years, 0 otherwise.
C. Dimensional Table “silver.DIM_Promo”

   1. id_promo (BigInt): Primary key.

   2. AcceptedCmp1 (Integer): 1 if customer accepted the 1st campaign’s offer, 0 otherwise.

3. AcceptedCmp2 (Integer): 1 if customer accepted the 2nd campaign's offer, 0 otherwise.

4. AcceptedCmp3 (Integer): 1 if customer accepted the 3rd campaign's offer, 0 otherwise.

5. AcceptedCmp4 (Integer): 1 if customer accepted the 4th campaign's offer, 0 otherwise.

6. AcceptedCmp5 (Integer): 1 if customer accepted the 5th campaign's offer, 0 otherwise.

7. Response (Integer): 1 if customer accepted the last campaign's offer, 0 otherwise.

D. Dimensional Table "silver.DIM_Calendar"

1. id_date (BigInt): Primary key.
2. full_date (Date): date of customer's enrollment with the company.
3. year (Integer)
4. month (Integer)
5. day (Integer)
6. week_of year (Integer)

Obs: Possible Marital_Status values "Absurd" and "Yolo" were merged into "Unknown", and "Alone" has been merged into "Single"

3.3. Metrics

In order to answer the questions posed at the beginning of this work, the dimensional model featured in the silver layer will not be enough. It does contain a cleaned and organized instance of the dataset, but it does not offer an immediate look at the main metrics that determine the answers we are looking for.

Therefore, in the next layer, the gold layer, we will have available the following metrics.

- Total spending (as the sum of the value spent with each product)
- Average spending per Education level
- Average spending per Marital Status
- Average spending per number of kids at home
- Average spending per number of teens at home
- Average spending with each product
- Campaign acceptance rate (average between 0 and 1)
- Ratio of purchase channel use

4. Pipeline

The pipeline was built and executed entirely in the cloud-based Databricks environment. The following are the steps taken.

4.1. Bronze Layer

This layer receives the raw data directly from the original .csv file. After downloaded from Kaggle, it was stored in the same GitHub folder this report is located. Having integrated GitHub to the Databricks environment, the file was immediately accessible through said platform.

The only remaining step was to save its contents as a delta table, a directory of files in cloud object storage.

4.2. Silver Layer

With the original table secured, its now advisable to check the dataset for any issues and clean it for posterior use.

Following this step, we will then model the star schema that will house the data in dimensional and fact tables.

4.3. Gold Layer

In this last layer we summarize the metrics of interest to us, regarding the questions posed earlier.

5. Data Analysis

5.1. Data Quality

After bringing the raw data into the bronze layer, focus was shifted into analyzing the quality of the dataset and cleaning whatever problems came to surface.

In general terms, the dataset was fine, not displaying any duplicated values and only 24 missing values in the column “Income”, which could easily be disregarded and its rows removed, given the size of the dataset.

Nonetheless, issues were identified in the ‘Marital_Status’ column, which featured nonsensical values, “Absurd” and “YOLO”, as well as redundancies, “Single” and “Alone”. I have thus decided on merging the last two into “Single” and creating a new option “Unknown” for both of the nonsensical values.

5.2. Answering Questions

With all the pieces set in place, it is now possible to return to the questions posed earlier in this report and address them with the now finished data pipeline.

a) Which features of the customer base are the most significant is maximizing expenditure?

To answer this question, the ‘customer profile’ table was built to fit the metrics of ‘total spending’ and ‘average spending’.

Starting with the feature ‘Income’, a numeric value, we see a moderate to strong correlation with total spending (0.64) – which is to be expected. If anything, one would assume it would be stronger.

Moving to categorical variables, we also have a predictable result when ranking the different Education levels by average spending, that is, more highly educated people tend to consume more from the company (in terms of amount, not number of purchases).

education	avg_spending
PhD	648.03
Master	583.32
Graduation	576.03
2n Cycle	456.93
Basic	61.16

Meanwhile, grouping average spending by ‘Marital Status’ yields a more interesting result, with ‘Married’ curiously being the group that spends the least. Also, unfortunately, while the incorrect categories (“Absurd” and “YOLO”) are gone, the new 'Unknown' category is not here for some reason...

marital_status	avg_spending
Widow	703.58
Together	570.82
Single	570.23
Divorced	568.09
Married	556.69

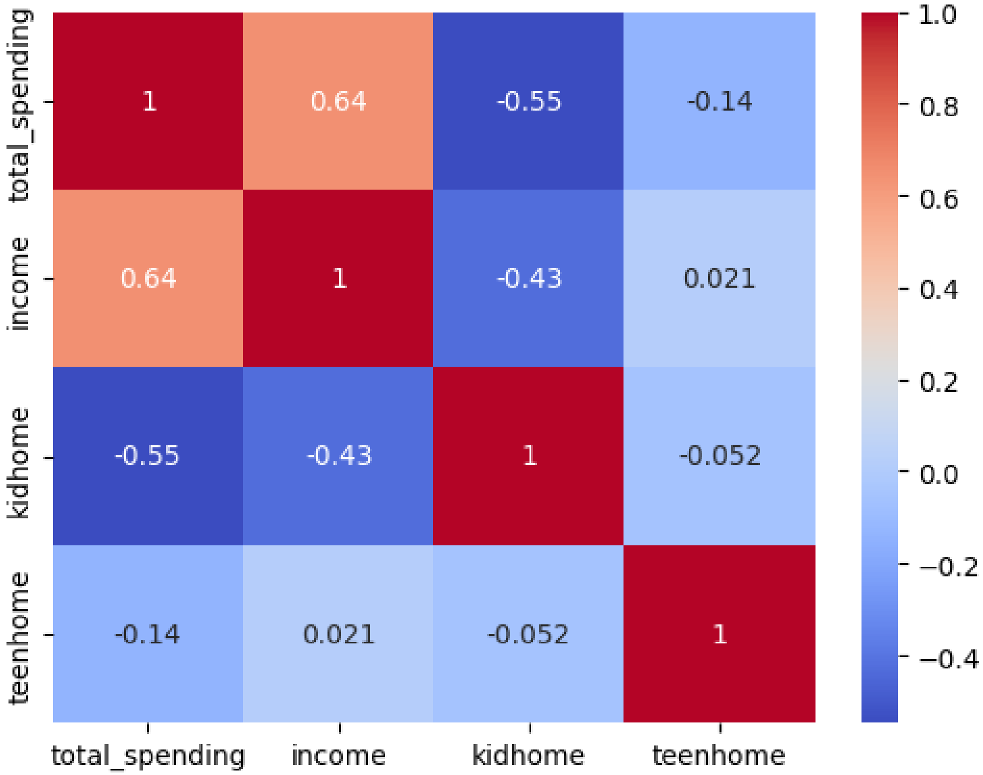
At last, we take a look the kids at home and teens at home variables, which are numeric but only ever get to 2, and thus can be treated as categorical as well. The results were also intriguing, as there is a negative correlation with total spending in either case, with 'Kidhome' having -0.55 (a moderate value) and 'Teenhome' -0.14 (weak correlation). Perhaps the weakening correlation relates to the maturity of the child.

Grouping average spending by either reinforces the idea that customers with no offspring spend more at the company's products. In fact, it seems to suggest that this is a company that appeals to young adults, with high education and income, but no family as of yet. If this were to be confirmed, it would be a very meaningful insight.

kidhome	avg_spending
0	852.59
1	184.32
2	107.37

teenhome	avg_spending
0	653.77
1	475.85
2	539.65

To summarize, the following image is a correlation table with each of the features thus discussed.



b) Which features of the customer base are most relevant to the effectiveness of promotional campaigns?

For this question, the ‘campaign acceptance’ table was built to fit the metrics of same name. It features 6 rows, one for each promotional campaign, the absolute number of customers that accepted a given campaign and the metric of interest, the acceptance rate, which portrays the relative percentage of total customers that accepted discounts from this campaign,

Fortunately, for the company, its marketing efforts appear to be improving, as it is the most recent campaign that has achieved the highest acceptance rate.

campaign_num	accepted_customers	acceptance_rate
Campaign_6	313	0.16
Campaign_4	158	0.08
Campaign_3	151	0.08
Campaign_5	148	0.07
Campaign_1	135	0.07
Campaign_2	26	0.01

c) Which are the best performing products?

For investigating the performance of the products, the table ‘product performance’ has been created to fit the metric ‘average spending with each product’.

Each product represents a line. The ‘total revenue’ column sums all of the spendings for that product, whereas the column that represents the main metric ‘avg_revenue_per_customer’ portrays the average amount a single person buys on average of that product.

product_category	total_revenue	avg_revenue_per_customer	revenue_rank
Wines	625478	305.41	1
Meat Products	345164	167.64	2
Fish Products	76840	45.17	3
Sweet Products	55952	33.42	4
Fruits	54016	31.92	5

Not much can be concluded from just this table without greater context. Perhaps the company should stop selling fruits; perhaps it needs to bolster its marketing for fruits. Regardless, having awareness of the asymmetries between the different product categories would be absolutely essential for strategic thinking.

d) Which is the most popular method of purchasing?

At last, we use the final gold table ‘purchase channel summary’ to investigate the metric ‘ratio of purchase channel use’.

Whereas the other tables mainly used the monetary value of purchases, this one is based on the columns NumWebPurchases, NumStorePurchases and NumCatalogPurchases.

avg_web	avg_store	avg_catalog
4.1	5.77	2.66

The average consumer purchases the least from the company's catalog (2.66 times on average) and the most from physical stores (5.77 times on average). While for now there can only be conjecture about the reasons why, this is nevertheless a valuable insight that might, for instance, suggest that catalogs are not worth maintaining, or that stores should at the very least receive the same level of attention and not be neglected.

We can of course glean the ration of purchase channel use from these results: physical stores represent 46% of transactions, website represents 32.7% and catalog represents 21% (approximately).

6. Self Evaluation and Conclusion

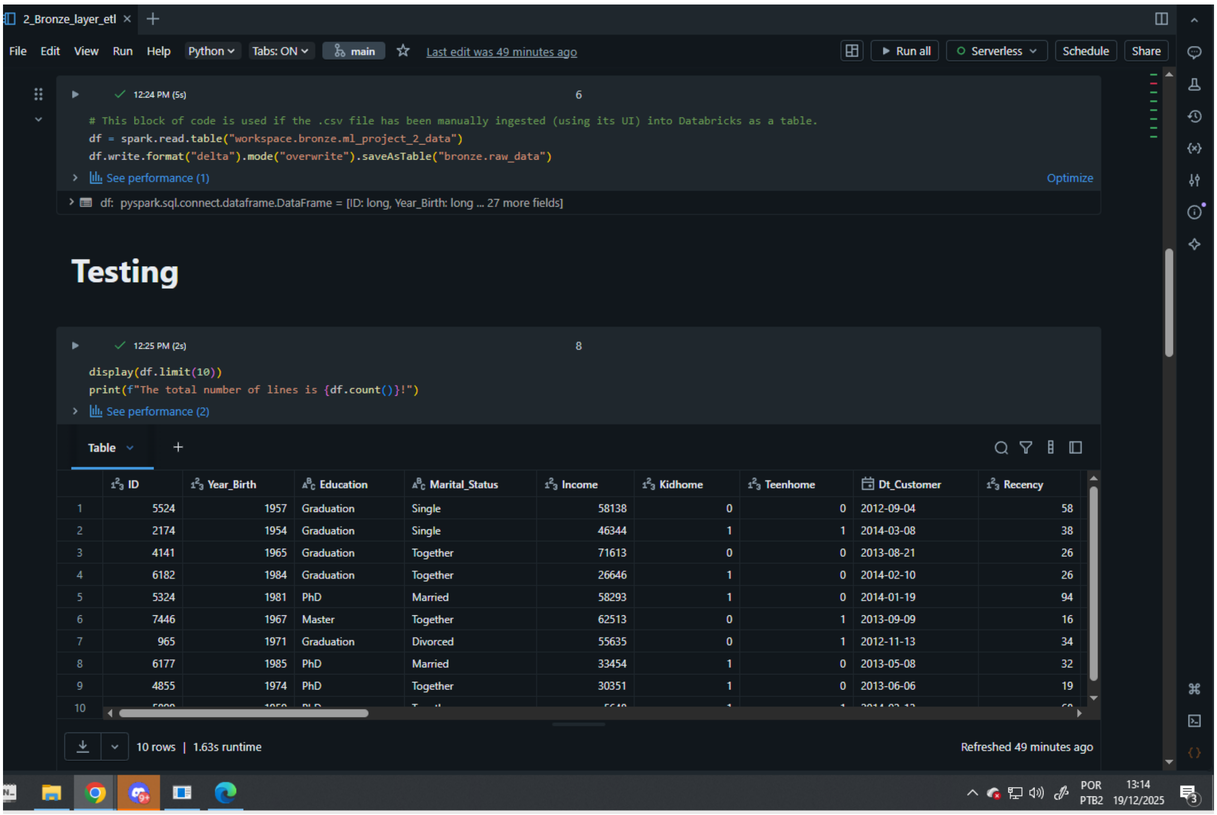
This Project is has indeed achieved its goals of answering the stated questions. However, much more can be realized within this dataset. The calendar dimensional table has remained untouched, as has the 'complain' and 'recency' variables. The regular and 'gold' product divison could have been explored as well.

Most unfortunate where the few hiccups along the line. As of 22 of December 2025, the 'Unknown' marital status value has seemingly vanished upon the construction of the gold layer, and the fact table oddly has a little under the number of lines it should have had. Hopefully these issues will be fixed soon.

7. Process Documentation

All of the relevant files and scripts can be found on my main GitHub repository, whereas the original data can be found in the following repository [Link]

The following are images related to the entire process, most of which was realized in the Databricks environment.



```
# This block of code is used if the .csv file has been manually ingested (using its UI) into Databricks as a table.
df = spark.read.table("workspace.bronze.ml_project_2_data")
df.write.format("delta").mode("overwrite").saveAsTable("bronze.raw_data")
```

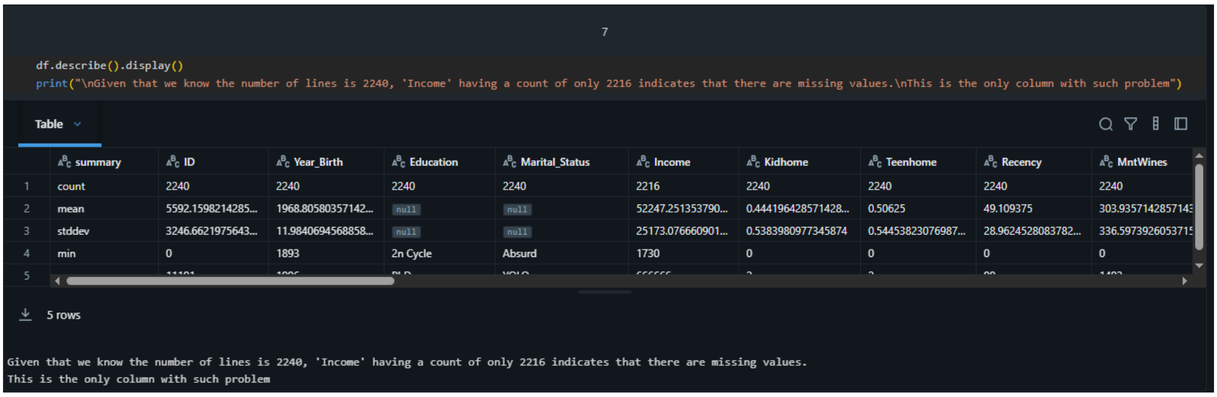
```
df:  pyspark.sql.connect.dataframe.DataFrame = [ID: long, Year_Birth: long ... 27 more fields]
```

Testing

```
display(df.limit(10))
print(f"The total number of lines is {df.count()}!")
```

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency
1	5524	1957	Graduation	Single	58138	0	0	2012-09-04	58
2	2174	1954	Graduation	Single	46344	1	1	2014-03-08	38
3	4141	1965	Graduation	Together	71613	0	0	2013-08-21	26
4	6182	1984	Graduation	Together	26646	1	0	2014-02-10	26
5	5324	1981	PhD	Married	58293	1	0	2014-01-19	94
6	7446	1967	Master	Together	62513	0	1	2013-09-09	16
7	965	1971	Graduation	Divorced	55635	0	1	2012-11-13	34
8	6177	1985	PhD	Married	33454	1	0	2013-05-08	32
9	4855	1974	PhD	Together	30351	1	0	2013-06-06	19
10	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

The first ingestion of data, in the bronze layer.

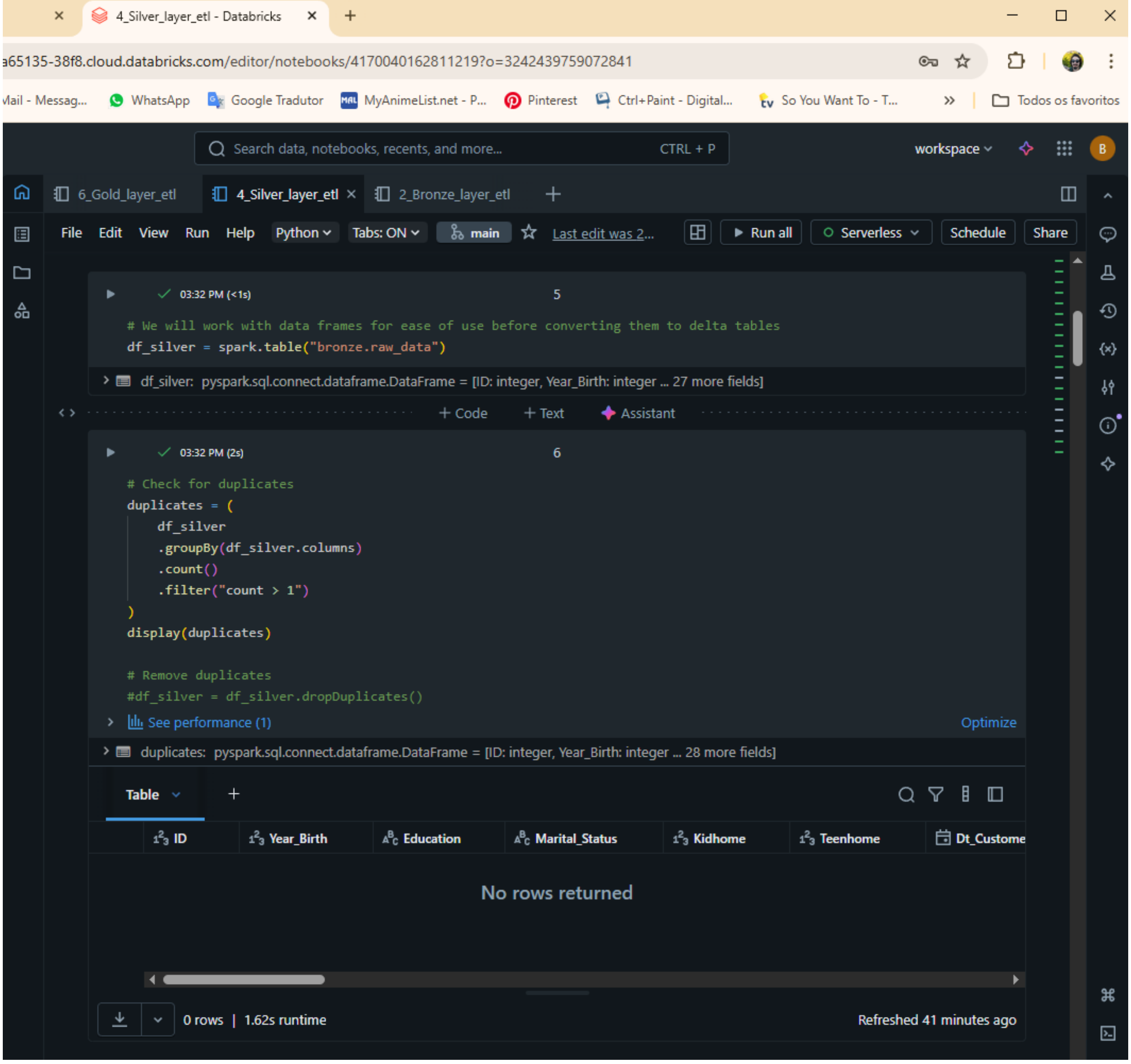


```
df.describe().display()
print("\nGiven that we know the number of lines is 2240, 'Income' having a count of only 2216 indicates that there are missing values.\nThis is the only column with such problem")
```

	summary	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Recency	MntWines
1	count	2240	2240	2240	2240	2216	2240	2240	2240	2240
2	mean	5592.1598214285...	1968.80580357142...	null	null	52247.251353790...	0.444196428571428...	0.50625	49.109375	303.9357142857143
3	stddev	3246.6621975643...	11.9840694568858...	null	null	25173.076660901...	0.5383980977345874	0.54453823076987...	28.9624528083782...	336.5973926053715
4	min	0	1893	2n Cycle	Absurd	1730	0	0	0	0
5	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

```
Given that we know the number of lines is 2240, 'Income' having a count of only 2216 indicates that there are missing values.
This is the only column with such problem
```

Verifying the existence of missing values (NAs).



03:32 PM (<1s) 5

```
# We will work with data frames for ease of use before converting them to delta tables
df_silver = spark.table("bronze.raw_data")
```

```
df_silver:  pyspark.sql.connect.dataframe.DataFrame = [ID: integer, Year_Birth: integer ... 27 more fields]
```

03:32 PM (2s) 6

```
# Check for duplicates
duplicates = (
    df_silver
    .groupBy(df_silver.columns)
    .count()
    .filter("count > 1")
)
display(duplicates)

# Remove duplicates
#df_silver = df_silver.dropDuplicates()
```

```
duplicates:  pyspark.sql.connect.dataframe.DataFrame = [ID: integer, Year_Birth: integer ... 28 more fields]
```

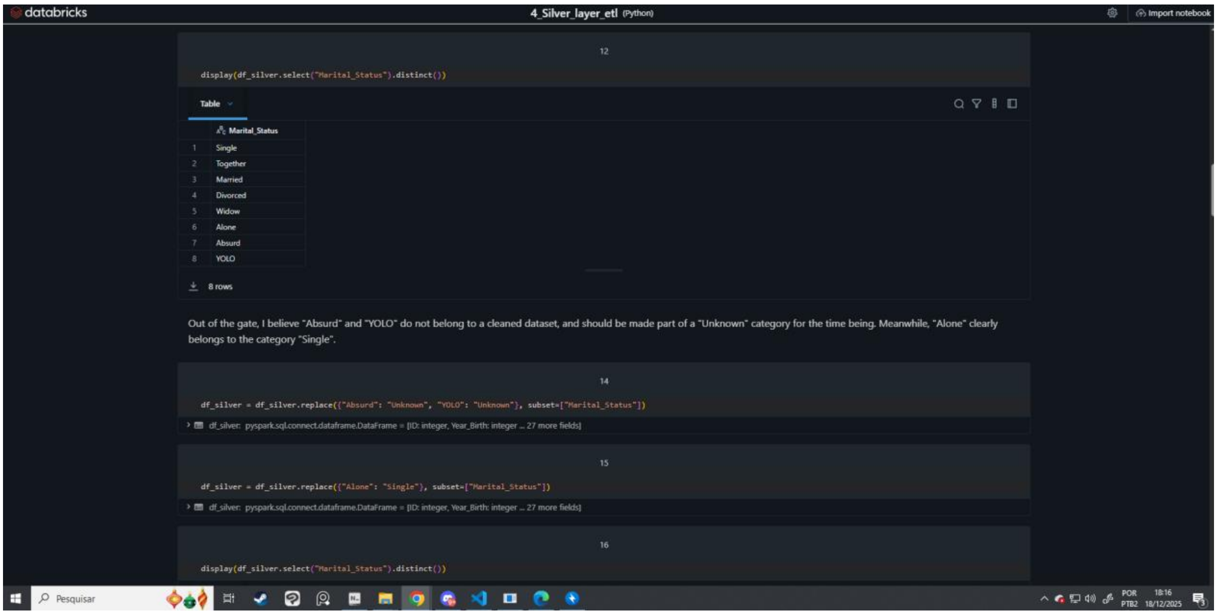
ID	Year_Birth	Education	Marital_Status	Kidhome	Teenhome	Dt_Custome

No rows returned

0 rows | 1.62s runtime

Refreshed 41 minutes ago

Verifying the lack of duplicates in the dataset.



12

```
display(df_silver.select("Marital_Status").distinct())
```

	Marital_Status
1	Single
2	Together
3	Married
4	Divorced
5	Widow
6	Alone
7	Absurd
8	YOLO

8 rows

Out of the gate, I believe "Absurd" and "YOLO" do not belong to a cleaned dataset, and should be made part of a "Unknown" category for the time being. Meanwhile, "Alone" clearly belongs to the category "Single".

14

```
df_silver = df_silver.replace({"Absurd": "Unknown", "YOLO": "Unknown"}, subset=["Marital_Status"])
```

df_silver: pyspark.sql.connect.dataframe.DataFrame = [ID: integer, Year_Birth: integer ... 27 more fields]

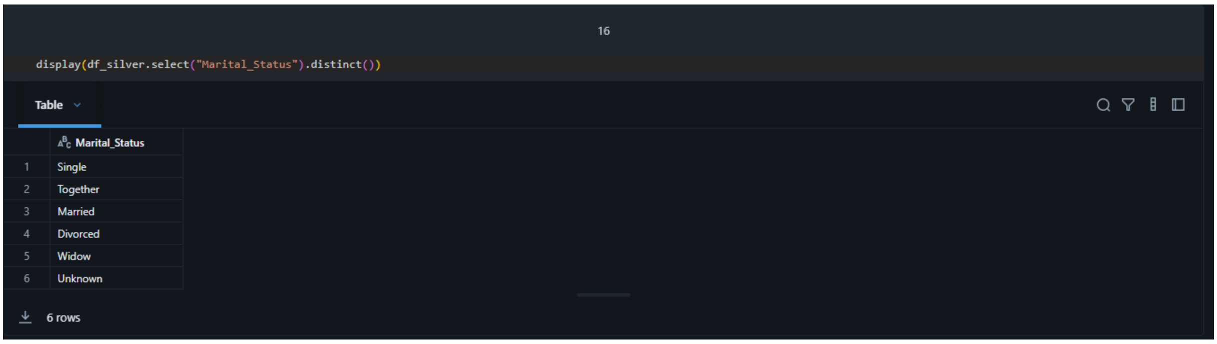
15

```
df_silver = df_silver.replace({"Alone": "Single"}, subset=["Marital_Status"])
```

df_silver: pyspark.sql.connect.dataframe.DataFrame = [ID: integer, Year_Birth: integer ... 27 more fields]

16

```
display(df_silver.select("Marital_Status").distinct())
```



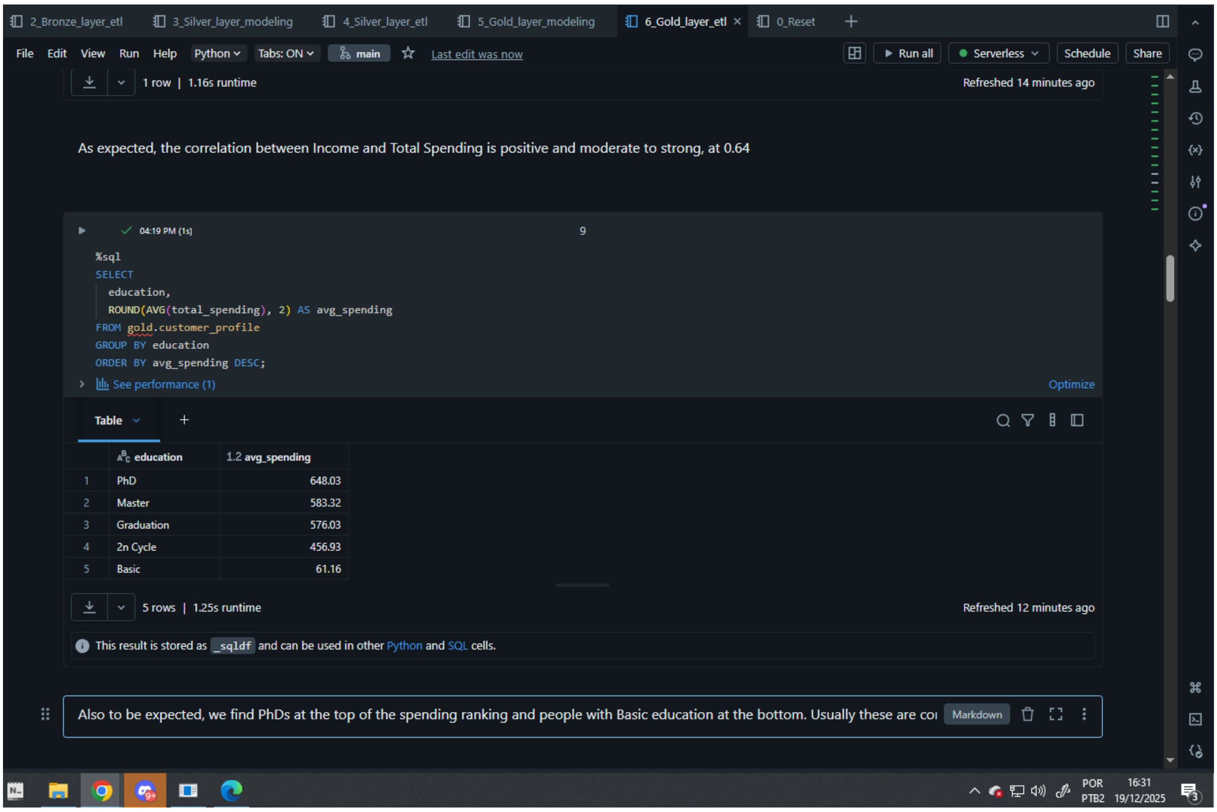
16

```
display(df_silver.select("Marital_Status").distinct())
```

	Marital_Status
1	Single
2	Together
3	Married
4	Divorced
5	Widow
6	Unknown

6 rows

Cleaning the ‘Marital_Status’ column of its incorrect values.



1 row | 1.16s runtime Refreshed 14 minutes ago

As expected, the correlation between Income and Total Spending is positive and moderate to strong, at 0.64

04:19 PM (1s) 9

```
%sql
SELECT
  education,
  ROUND(AVG(total_spending), 2) AS avg_spending
FROM gold.customer_profile
GROUP BY education
ORDER BY avg_spending DESC;
```

See performance (1)

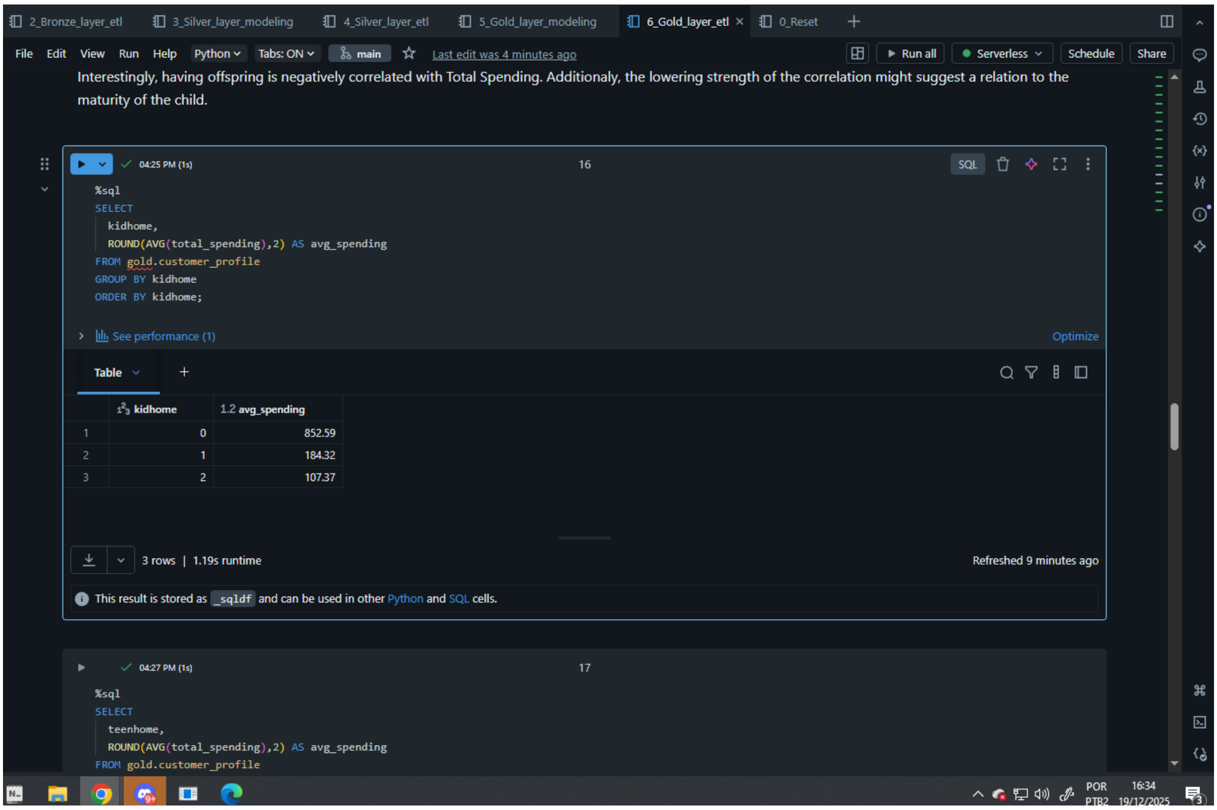
	education	1.2 avg_spending
1	PhD	648.03
2	Master	583.32
3	Graduation	576.03
4	2n Cycle	456.93
5	Basic	61.16

5 rows | 1.25s runtime Refreshed 12 minutes ago

This result is stored as `_sqldf` and can be used in other Python and SQL cells.

Also to be expected, we find PhDs at the top of the spending ranking and people with Basic education at the bottom. Usually these are co

Using the gold table ‘customer_profile’ to group the customers by education level and find their average spending values.



Interestingly, having offspring is negatively correlated with Total Spending. Additionaly, the lowering strength of the correlation might suggest a relation to the maturity of the child.

04:25 PM (1s) 16

```
%sql
SELECT
  kidhome,
  ROUND(AVG(total_spending),2) AS avg_spending
FROM gold.customer_profile
GROUP BY kidhome
ORDER BY kidhome;
```

See performance (1) Optimize

Table

	kidhome	avg_spending
1	0	852.59
2	1	184.32
3	2	107.37

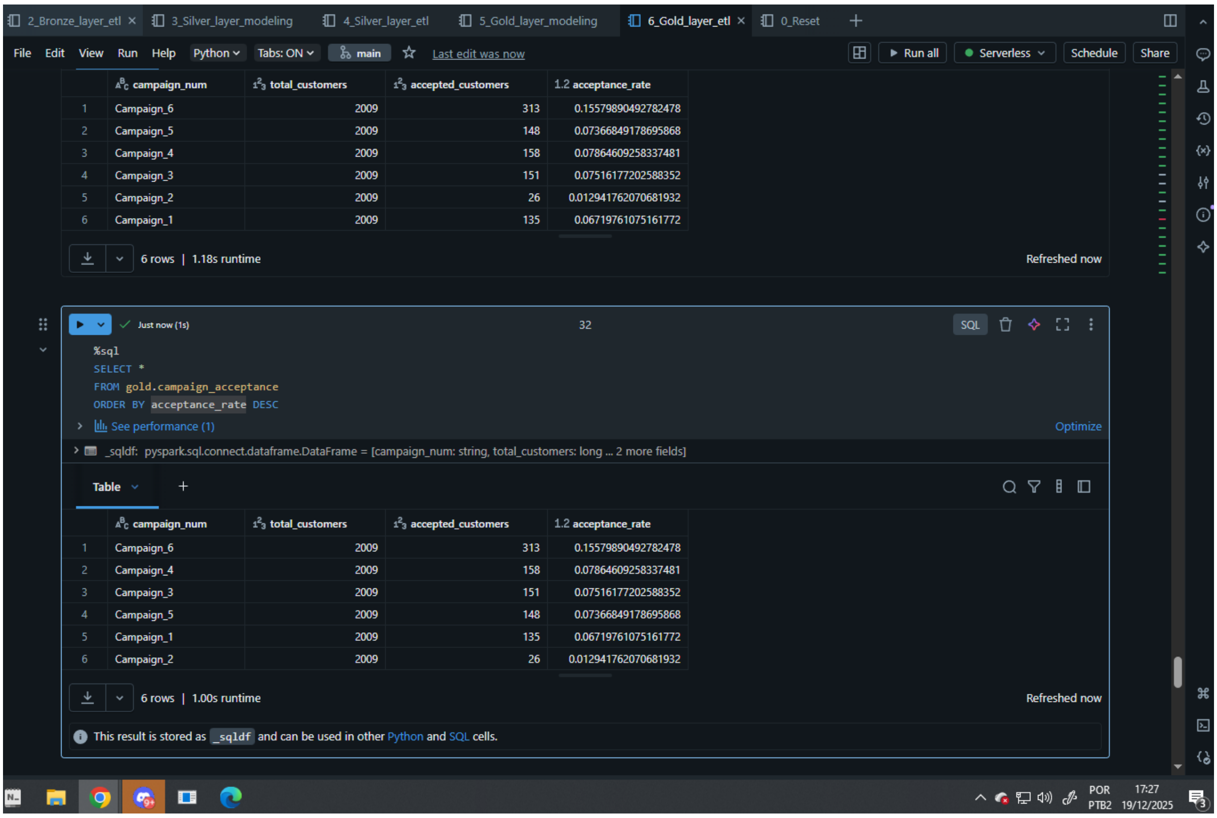
3 rows | 1.19s runtime Refreshed 9 minutes ago

This result is stored as `_sqldf` and can be used in other Python and SQL cells.

04:27 PM (1s) 17

```
%sql
SELECT
  teenhome,
  ROUND(AVG(total_spending),2) AS avg_spending
FROM gold.customer_profile
```

Using the gold table ‘customer_profile’ to group the customers by number of children and find their average spending values.



	campaign_num	total_customers	accepted_customers	acceptance_rate
1	Campaign_6	2009	313	0.15579890492782478
2	Campaign_5	2009	148	0.07366849178695868
3	Campaign_4	2009	158	0.07864609258337481
4	Campaign_3	2009	151	0.07516177202588352
5	Campaign_2	2009	26	0.012941762070681932
6	Campaign_1	2009	135	0.06719761075161772

6 rows | 1.18s runtime Refreshed now

Just now (1s) 32

```
%sql
SELECT *
FROM gold.campaign_acceptance
ORDER BY acceptance_rate DESC
```

See performance (1) Optimize

_sqldf: pyspark.sql.connect.dataframe.DataFrame = [campaign_num: string, total_customers: long ... 2 more fields]

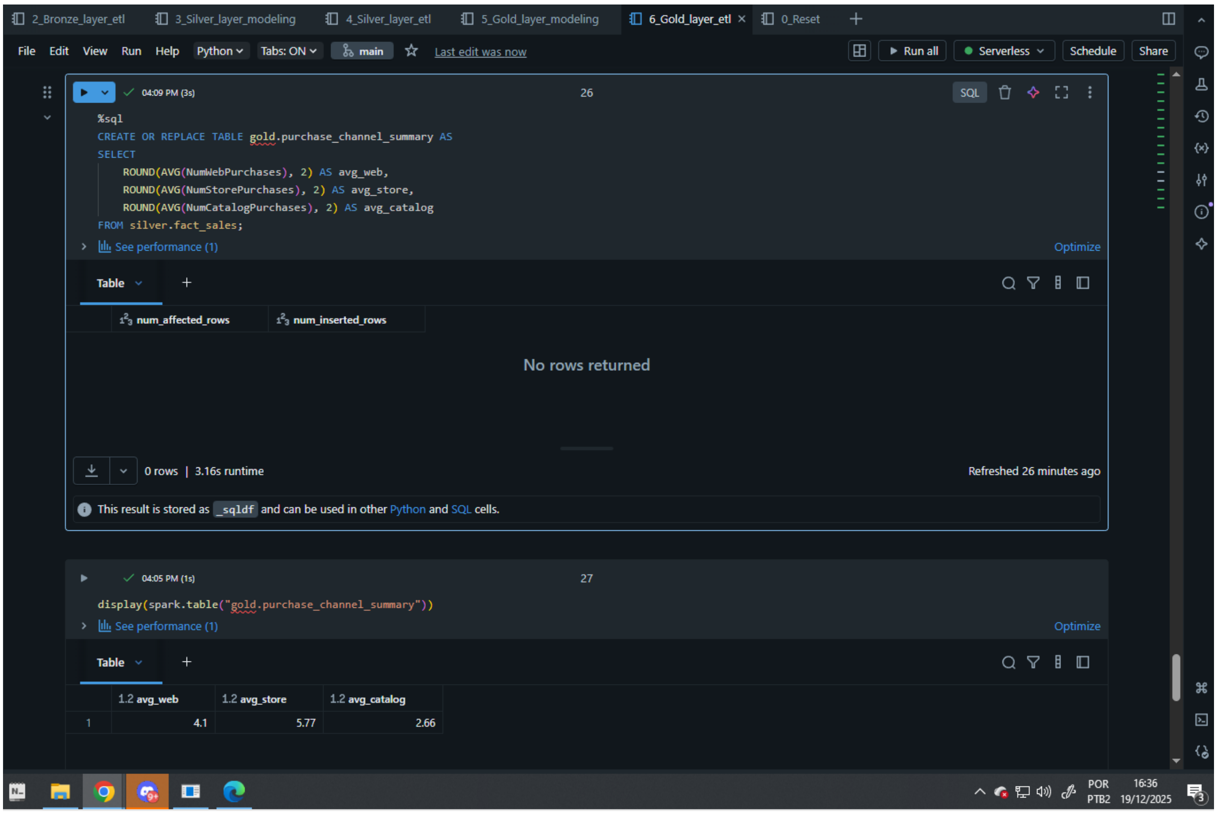
Table

	campaign_num	total_customers	accepted_customers	acceptance_rate
1	Campaign_6	2009	313	0.15579890492782478
2	Campaign_4	2009	158	0.07864609258337481
3	Campaign_3	2009	151	0.07516177202588352
4	Campaign_5	2009	148	0.07366849178695868
5	Campaign_1	2009	135	0.06719761075161772
6	Campaign_2	2009	26	0.012941762070681932

6 rows | 1.00s runtime Refreshed now

This result is stored as `_sqldf` and can be used in other Python and SQL cells.

Using the gold table ‘campaign_acceptance’ to display the acceptance rate of each of the six promotional campaigns.



```
%sql
CREATE OR REPLACE TABLE gold.purchase_channel_summary AS
SELECT
    ROUND(AVG(NumWebPurchases), 2) AS avg_web,
    ROUND(AVG(NumStorePurchases), 2) AS avg_store,
    ROUND(AVG(NumCatalogPurchases), 2) AS avg_catalog
FROM silver.fact_sales;
```

num_affected_rows	num_inserted_rows

No rows returned

0 rows | 3.16s runtime

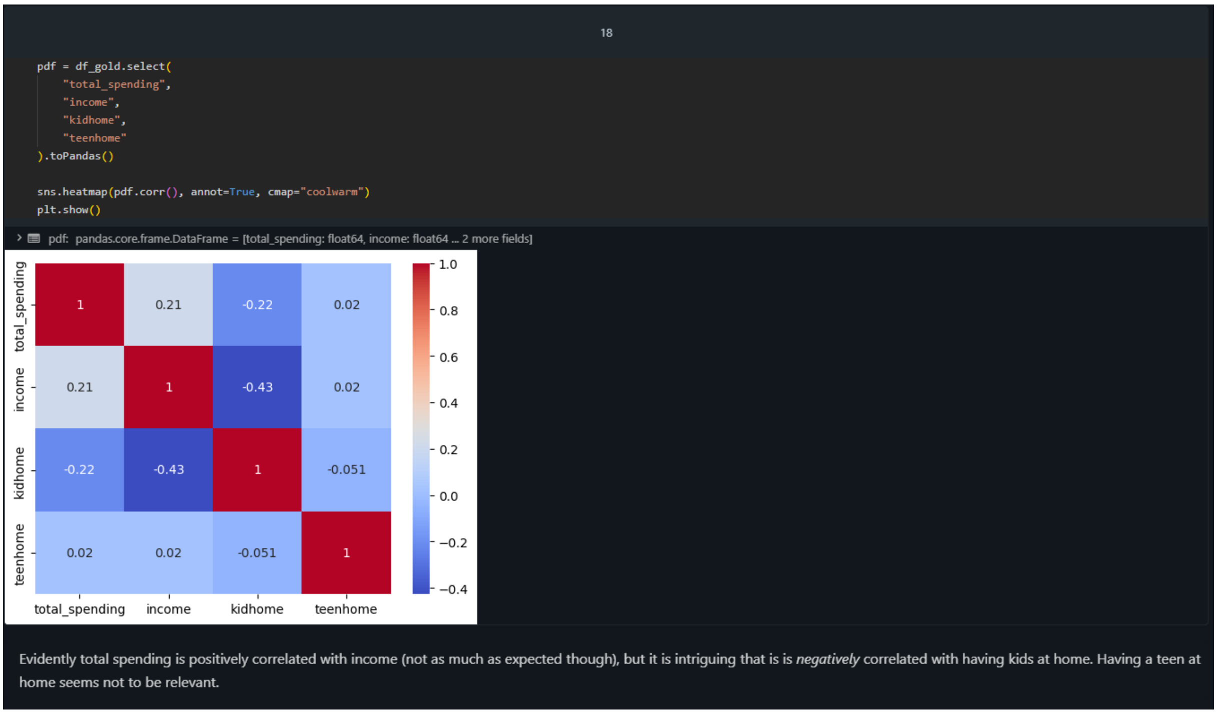
Refreshed 26 minutes ago

This result is stored as `_sqldf` and can be used in other Python and SQL cells.

```
display(spark.table("gold.purchase_channel_summary"))
```

	avg_web	avg_store	avg_catalog
1	4.1	5.77	2.66

The average number of purchases for each of the three channels: website, physical stores and catalog.



```
pdf = df_gold.select(
    "total_spending",
    "income",
    "kidhome",
    "teenhome"
).toPandas()

sns.heatmap(pdf.corr(), annot=True, cmap="coolwarm")
plt.show()
```

pdf: pandas.core.frame.DataFrame = [total_spending: float64, income: float64 ... 2 more fields]

Evidently total spending is positively correlated with income (not as much as expected though), but it is intriguing that is is *negatively* correlated with having kids at home. Having a teen at home seems not to be relevant.

Studying the correlations between the metric ‘total spending’ and the categorical variables.