Data Architecture Build
&
Market Recommendations for Olist

Overarching Objectives

Three Roles

Data Engineering:

- Build an end-to-end data platform for data processing and BI modelling

Data Science:

- Enrich the data with sentiment analysis

Business Intelligence:

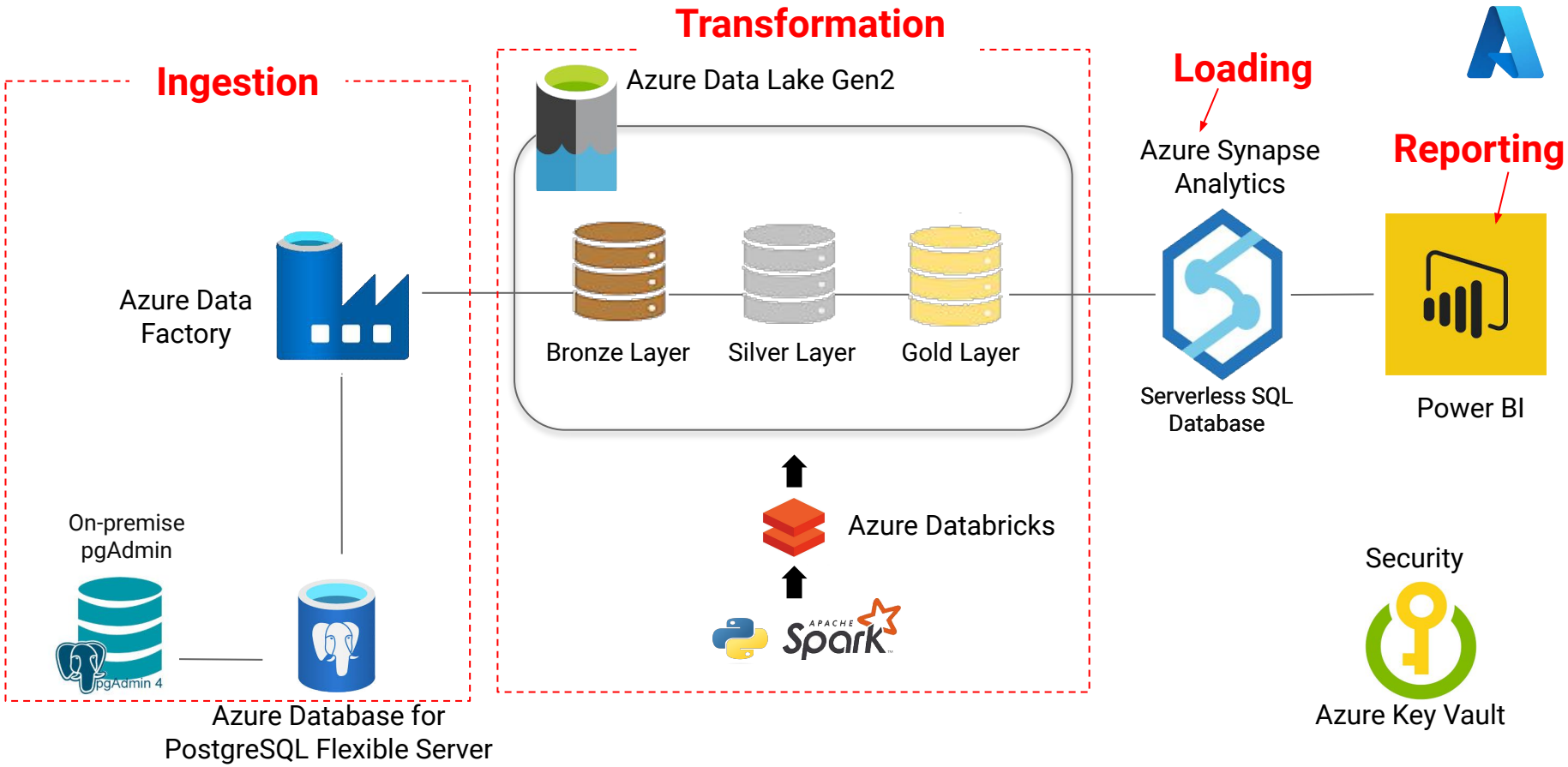
- Deliver actionable business insights

Introduction to Olist

- Brazil's leading e-commerce marketplace, connecting small businesses to customers nationwide
  - *Average order value: 132 BRL = S$35 (median salary: S$2000)*
- **BI goals:**
  - Increase sales turnover
  - Investigation of:
    - Sellers
    - Products
    - Customers

Data Engineering Objectives

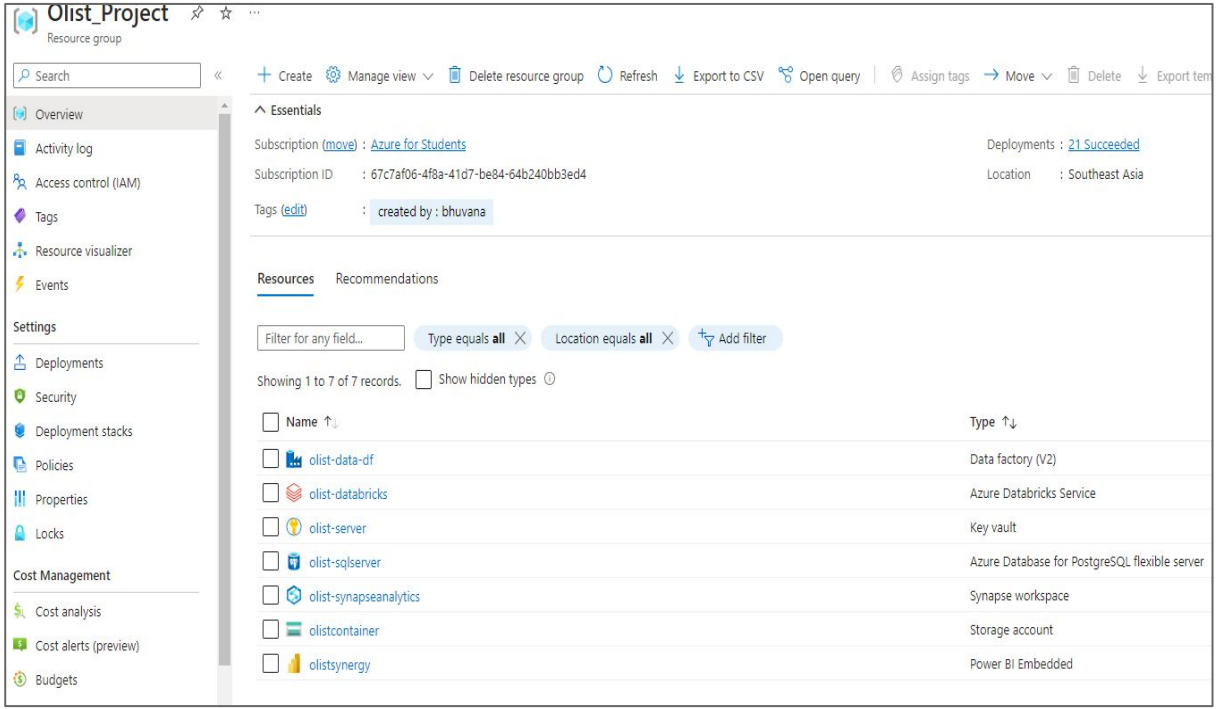
- As a DE team, we gathered project requirements from the BI and data science teams.
- After investigating the use of cloud solutions, we selected Azure.



Full Azure Cloud-Based Data Flow Architecture

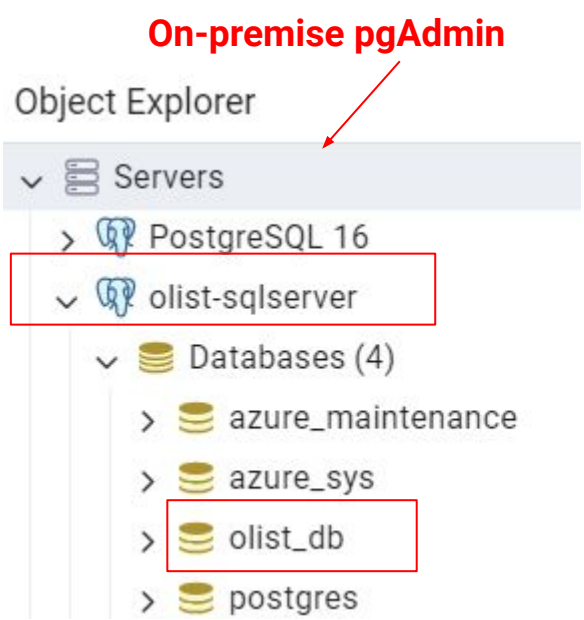
Azure Portal – Resource Group Creation and Environmental Setup

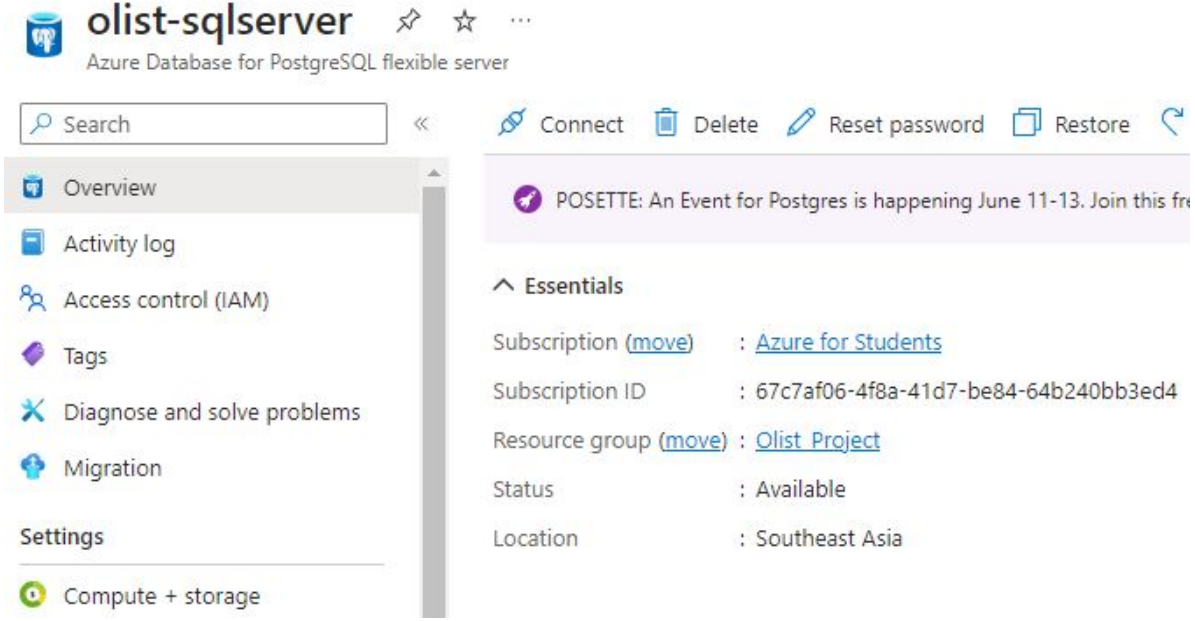
- Resource group
- Storage container
- PostgreSQL flexible server
- Azure Data Factory
- Microsoft Integration Runtime
- Azure Databricks
- Azure Synapse Analytics
- Azure Key Vault



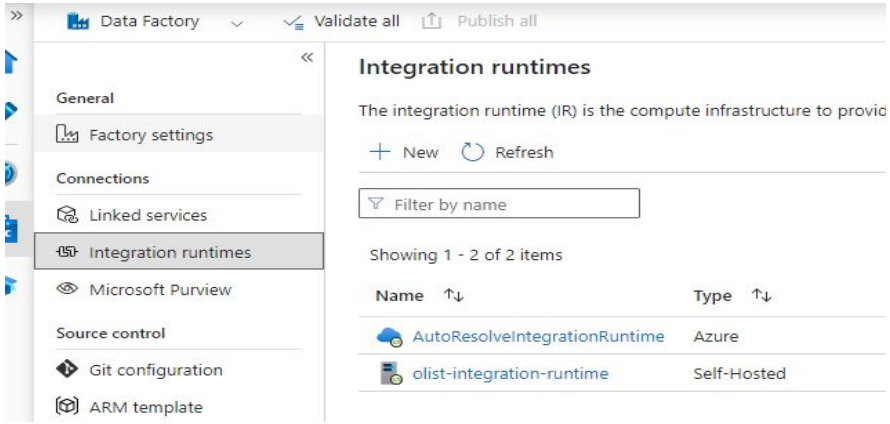
Connection between On-Premise pgAdmin and Azure PostgreSQL Flexible Server

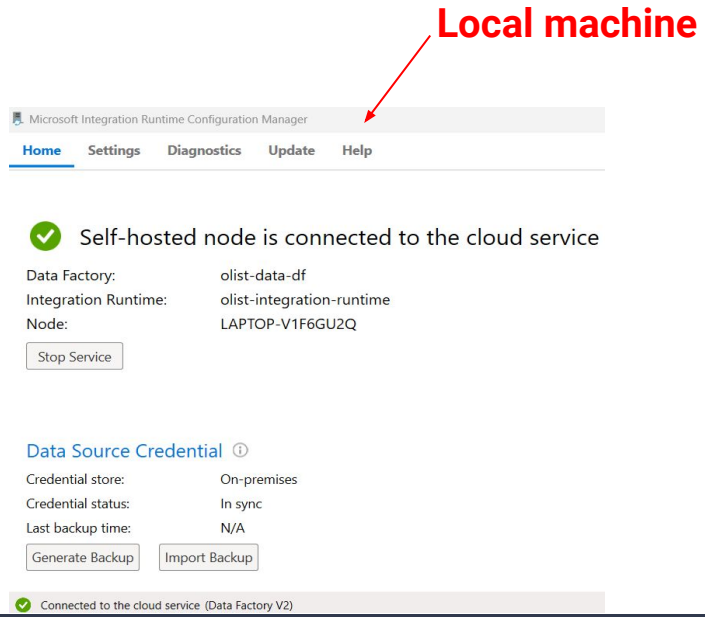
- Setup connection between Azure Database for PostgreSQL flexible server and on-premise pgAdmin





- There is no relationship between Azure Data Factory and on-premise pgAdmin
- To establish connection we use integration runtimes in ADF
- Self-Hosted Integration Runtime
  - for connecting on-premise pgAdmin
  - Install in our machine to establish connection
- Auto-Resolved Integration Runtime
  - Used to connect to any cloud-based resources





Installation of Self-Hosted Integration Runtime

Loading the CSV files into PostgreSQL Database using SQLAlchemy and Accessing through pgAdmin

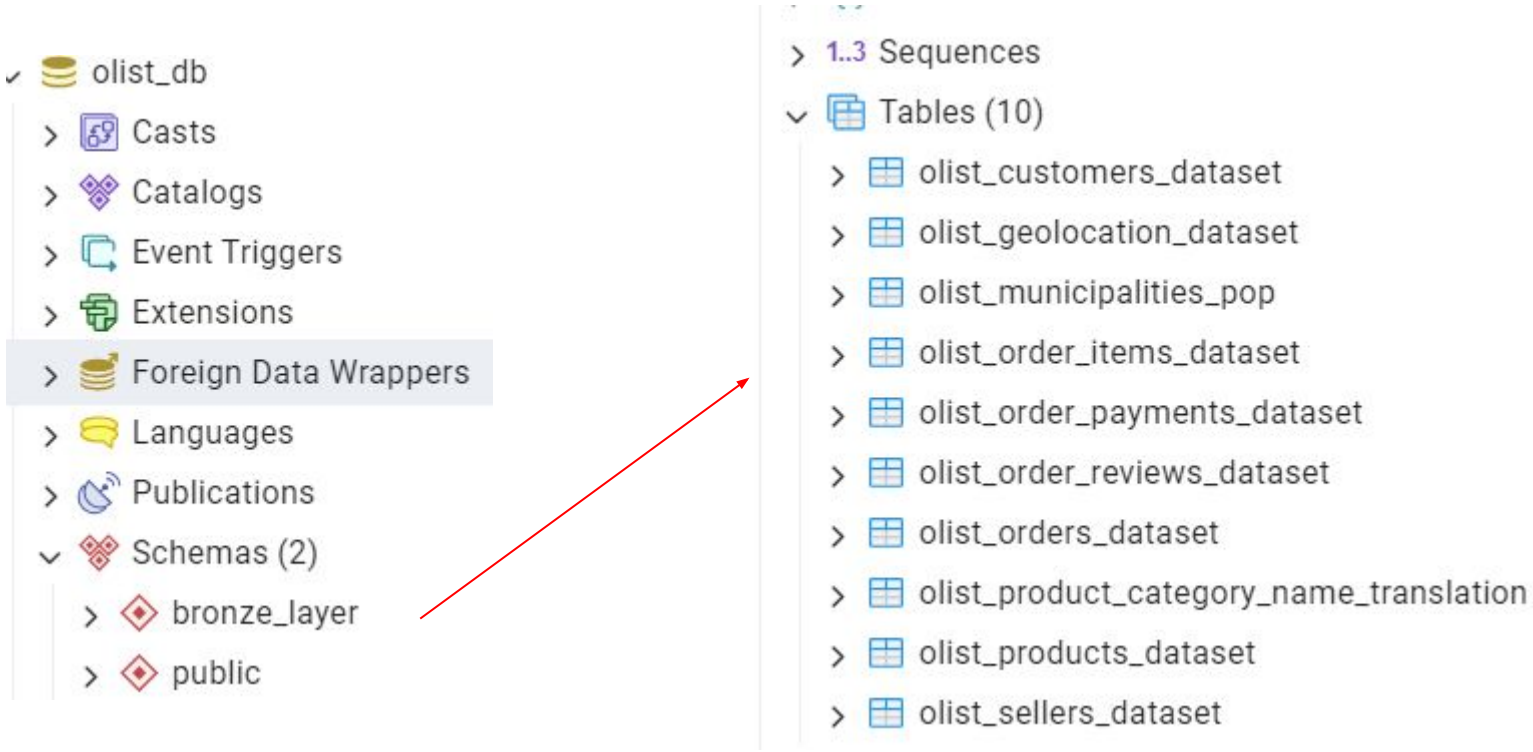
```
# Retrieve the connection string
connection_string = "postgresql://p                    @olist-sqlserver.postgres.database.azure.com:5432/olist_db"
```

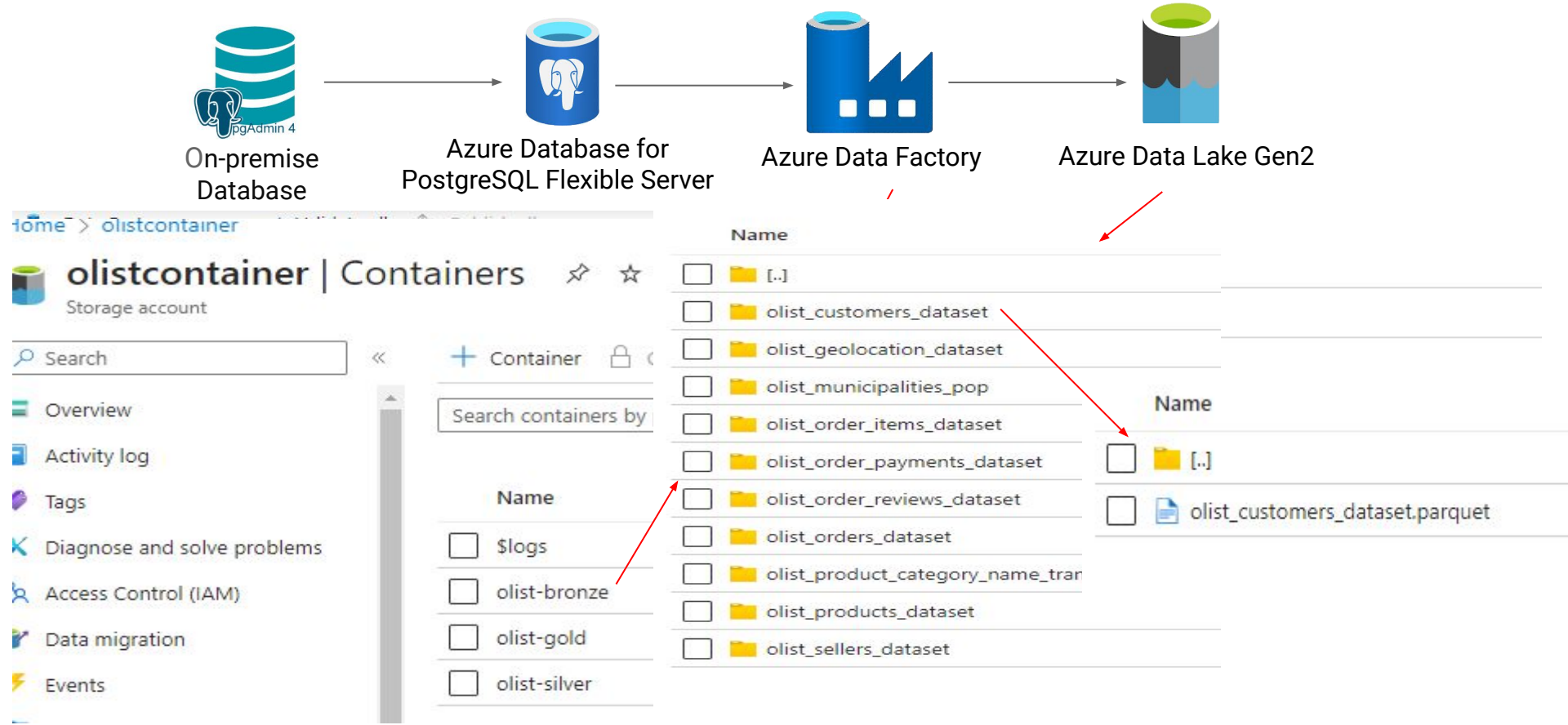
Username : password

```
# Directory containing C
csv_dir = r"C:\Users\bhu

# List all CSV files in
csv_files = [f for f in

def process_csv(csv_file
    table_name = os.path
    csv_path = os.path.j

    # Create SQLAlchemy
    engine = create_engi
```

```
Processing CSV file: olist_products_dataset.csv
Processing CSV file: olist_product_category_name_translation.csv
Processing CSV file: olist_sellers_dataset.csv
Table 'olist_product_category_name_translation' created or replaced successfully in s
Table 'olist_sellers_dataset' created or replaced successfully in schema 'bronze_laye
Table 'olist_municipalities_pop' created or replaced successfully in schema 'bronze_l
Table 'olist_products_dataset' created or replaced successfully in schema 'bronze_lay
Table 'olist_order_payments_dataset' created or replaced successfully in schema 'bron
Table 'olist_order_reviews_dataset' created or replaced successfully in schema 'bronz
Table 'olist_customers_dataset' created or replaced successfully in schema 'bronze_la
Table 'olist_order_items_dataset' created or replaced successfully in schema 'bronze_
Table 'olist_orders_dataset' created or replaced successfully in schema 'bronze_layer
Table 'olist_geolocation_dataset' created or replaced successfully in schema 'bronze_
Processing results:
ThreadPoolExecutor shutdown complete
```

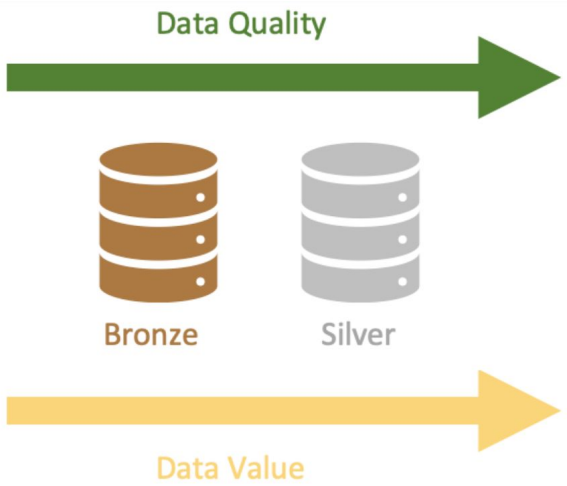
Tables in the Olist_db Database

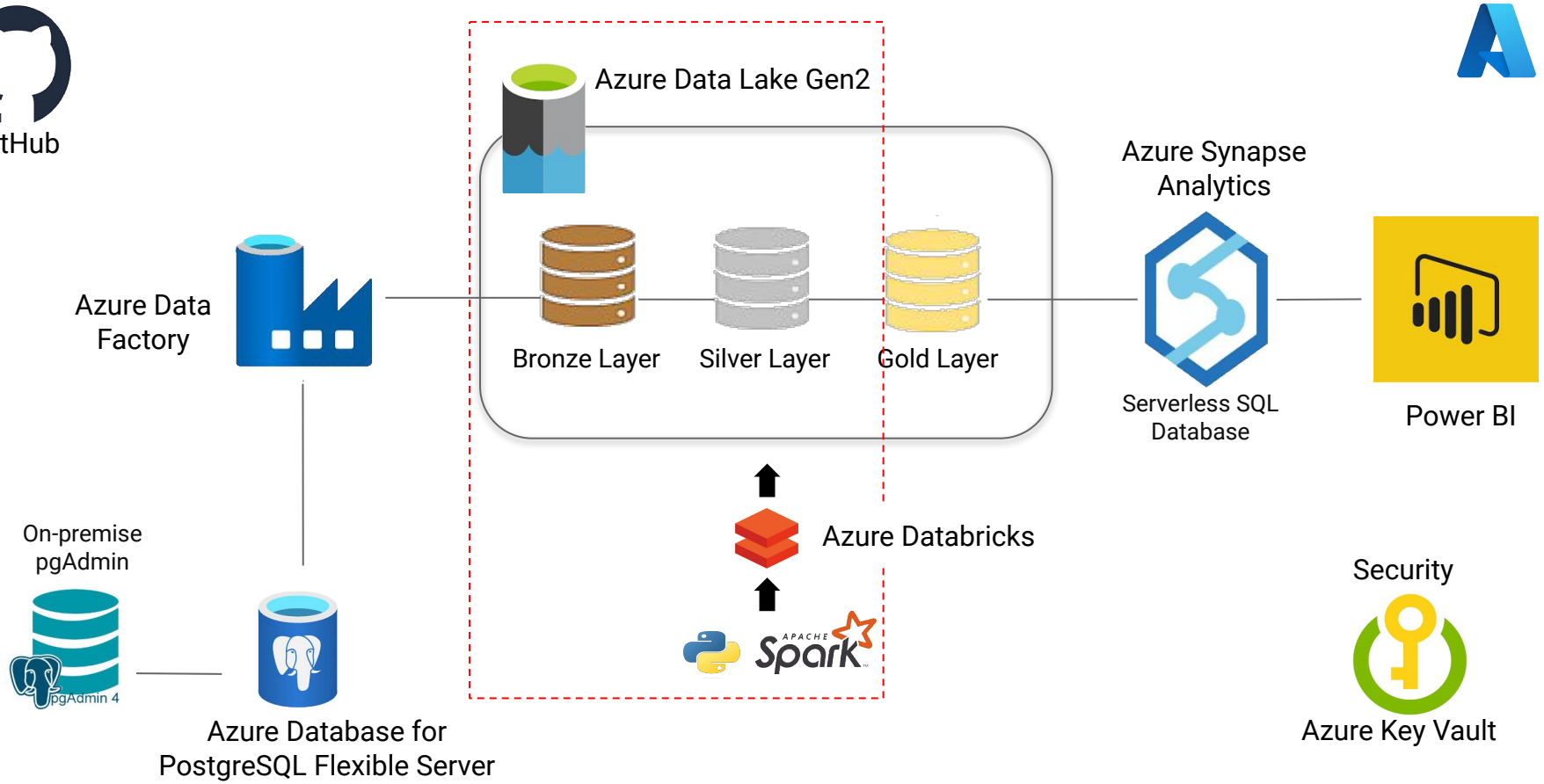




Ingestion of Dataset into the Bronze Layer

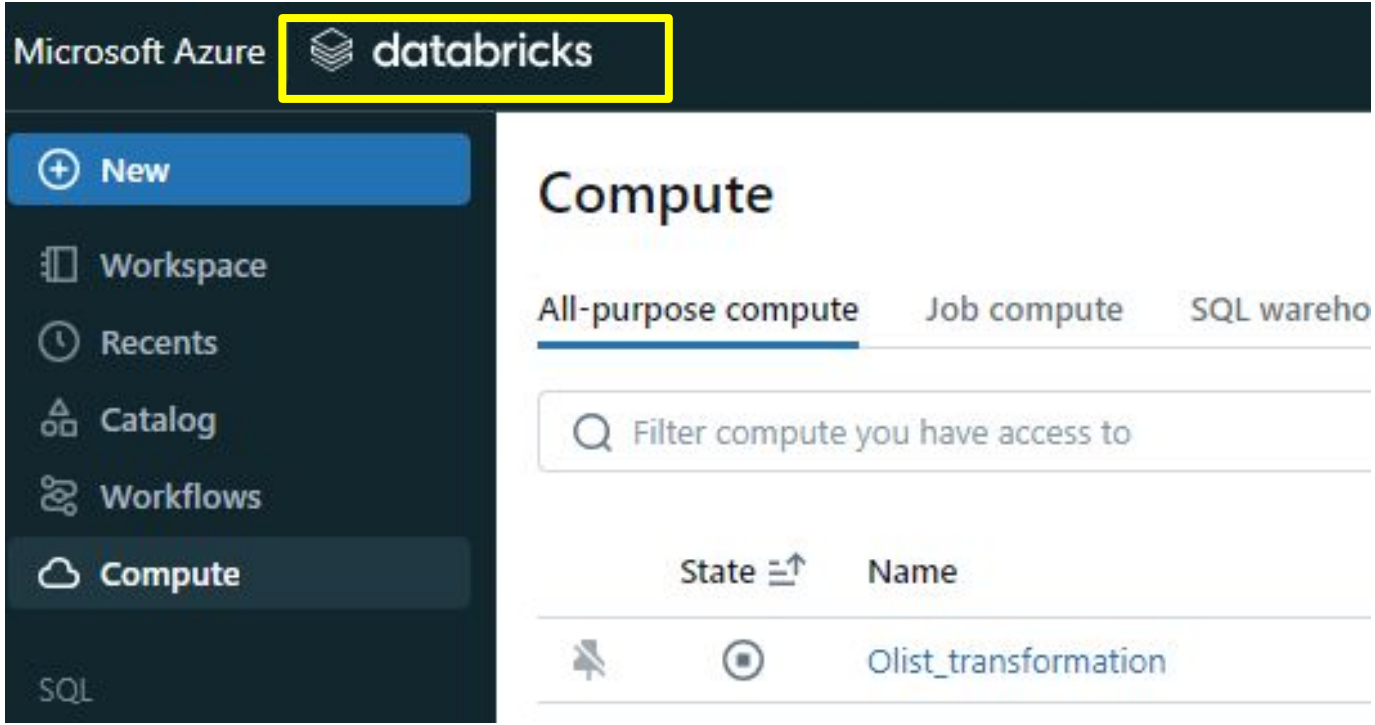
DATA TRANSFORMATION #1





Full Azure Cloud-Based Data Flow Architecture

Compute Cluster



Storage Mount

storage mount Python

File Edit View Run Help Last edit was 6 days ago New cell UI: ON

Run al

Workspace

Shared

Sort: Name

- gold_transformation
- olist-silver-transform
- olist-transform2
- Reviews Text Translation
- storage mount

5/7/2024 (21s)

```python
configs = {
  "fs.azure.account.auth.type": "CustomAccessToken",
  "fs.azure.account.custom.token.provider.class": spark.conf.get("spark.databricks.passthrough.adls.gen2.tokenProviderClassName")
}

# Check if the directory is already mounted
is_mounted = False
for mount in dbutils.fs.mounts():
  if mount.mountPoint == "/mnt/olist-bronze":
    is_mounted = True
    break

# Unmount the directory if it is already mounted
if is_mounted:
  dbutils.fs.unmount("/mnt/olist-bronze")

# Mount the directory
dbutils.fs.mount(
  source = "abfss://olist-bronze@olistcontainer.dfs.core.windows.net",
  mount_point = "/mnt/olist-bronze",
  extra_configs = configs)
```

```
/mnt/olist-bronze has been unmounted.

True
```

Data Ingestion

- **Nine CSV files:**
  - Orders
  - Customers
  - Sellers
  - Geolocations
  - Order Items
  - Order Reviews
  - Order Payments
  - Products
  - Product Category Name Translation

- **10th CSV file:**
  - Census 2022 (*IBGE*)
    https://sidra.ibge.gov.br/pesquisa/censo-demografico/demografico-2022/primeiros-resultados-populacao-e-domicilios

5 days ago (<1s) 2

```
input_path_trans= '/mnt/olist-bronze/bronze_layer/olist_product_category_name_translation/olist_product_category_name_translation.parquet'
input_path_prod = '/mnt/olist-bronze/bronze_layer/olist_products_dataset/olist_products_dataset.parquet'
input_path_items = '/mnt/olist-bronze/bronze_layer/olist_order_items_dataset/olist_order_items_dataset.parquet'
input_path_orders = '/mnt/olist-bronze/bronze_layer/olist_orders_dataset/olist_orders_dataset.parquet'
input_path_payments = '/mnt/olist-bronze/bronze_layer/olist_order_payments_dataset/olist_order_payments_dataset.parquet'
input_path_customers = '/mnt/olist-bronze/bronze_layer/olist_customers_dataset/olist_customers_dataset.parquet'
input_path_selle
input_path_revie
input_path_geo =
input_path_censu
```

4 days ago (2s) 4

```
columns = {0: "product_category_name", 1: "product_category_name_english"}
product_trans_df = pd.DataFrame(spark.read.format('parquet').load(input_path_trans,header=True).collect())
product_trans_df.rename(columns=columns, inplace=True)
print(product_trans_df.head())
print(product_trans_df.info())
```

(2) Spark Jobs

```
   product_category_name product_category_name_english
0            beleza_saude                 health_beauty
1  informatica_acessorios         computers_accessories
```

Data Ingestion

Data Cleaning I

- Null/NaN values (e.g., review comments, product attributes)
- Data type constraints (e.g., timestamps, integers)
- Inaccurate spelling in column names (e.g., **“lenght”**)
- Non-standard English or typos in translations of product category names (e.g., **“telephony”**, **“home confort”**)
- Portuguese accented characters in city names (e.g., **â**, **ç**)

Data Cleaning II

1. **Out of range data**

```
# Drop latitude & longitude outliers (relative to Brazil's range of coordinates)
geolocations_df3 = geolocations_converted.copy()

geolocations_df3 = geolocations_df3[geolocations_df3["geolocation_lat"] <= 5.288685]
assert geolocations_df3["geolocation_lat"].max() <= 5.288685

geolocations_df3 = geolocations_df3[geolocations_df3["geolocation_lat"] >= -33.798533]
assert geolocations_df3["geolocation_lat"].min() >= -33.798533

geolocations_df3 = geolocations_df3[geolocations_df3["geolocation_lng"] <= -34.703311]
assert geolocations_df3["geolocation_lng"].max() <= -34.703311

geolocations_df3 = geolocations_df3[geolocations_df3["geolocation_lng"] >= -73.968899]
assert geolocations_df3["geolocation_lng"].min() >= -73.968899
```

Data Cleaning II

2. **Missing**
   - Tim

```
# Fill missing values for 'deliv
for column in columns_to_fill:
    if column == 'order_approved
        delivered_orders.loc[del
        [column].isnull(), 'orde
        ['approved_to_delivered_
    elif column == 'order_delive
        delivered_orders.loc[del
        [column].isnull(), 'orde
        unit='D')
    elif column == 'order_delive
        delivered_orders.loc[del
        [column].isnull(), 'orde
        ['purchase_to_delivered_
```

```
# Iteratively repeat the procedure until no further missing zipcodes are found
while zip_cust_list:
    geolocations_df5 = geolocations_df6

    subs_zip3 = []
    for missing_zip3 in zip_cust_list:
        subs_zip3.append(closest(lst2, missing_zip3))

    cust_dict2 = dict(zip(subs_zip3, zip_cust_list))

    cust_zip_df2 = geolocations_df5[geolocations_df5["geolocation_zip_code_prefix"].isin(subs_zip3)]
    cust_zip_df3 = cust_zip_df2.copy()
    cust_zip_df3["geolocation_zip_code_prefix"] = cust_zip_df3["geolocation_zip_code_prefix"].map(cust_dict2)

    geolocations_df6 = pd.concat([geolocations_df5, cust_zip_df3])
    geolocations_df6.reset_index(drop=True)
    print("Looping...")

    zip_cust_list = list(set(customers_df4["customer_zip_code_prefix"]).difference(set(geolocations_df6
    ["geolocation_zip_code_prefix"])))
```

Data Cleaning II

```python
# Replacing unstandardised city names in Sellers dataset with official names, matched by zip code
sellers_subset3 = sellers_sub
sellers_df3 = sellers_df2.cop
sellers_df3["city"].update(se
sellers_df3.rename(columns={'
```

```python
# Standardisation #2 Replacing 230 unstandardised city names with official demographic city names,
# then testing with assert statements
            'conservatoria': "valenca",
            'itamira': "apora",
            'quatituba': "itueta",
            'santo amaro de campos': "campos dos goytacazes",
            'travessao': "campos dos goytacazes"}
customers_df4.replace({"official_city": city_dict}, inplace=True)

assert customers_df4[customers_df4["official_city"] == "abrantes"].empty == True
assert customers_df4[customers_df4["official_city"] == "adhemar de barros"].empty == True
```

resu

- Sou
  -
- Two
  I. Zip code matching with Geolocations' official cities
  II. Obtaining official city name from Google Maps

Data Cleaning II

4. **Duplicate values**
   - >1,000,000 Geolocations records, but only 19,000 unique zip code prefixes

```
# Group records by zip_code_prefix, getting mean coordinates & modal city and state names
# Then reset index to make zip_code_prefix a column
geolocations_df4 = geolocations_df3.copy()
summaries = {"geolocation_lat": "mean",
             "geolocation_lng": "mean",
             "geolocation_city": pd.Series.mode,
             "geolocation_state": pd.Series.mode}
geolocations_df4 = geolocations_df4.groupby(by="geolocation_zip_code_prefix").agg(summaries)
```

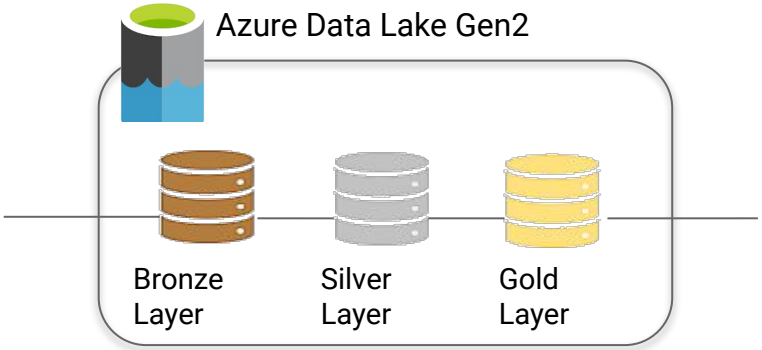
Entity Relationship Diagram

```python
# Convert Pandas DataFrame to Spark DataFrame
order_items_spark_df = spark.createDataFrame(products_df4)
order_items_spark_df = spark.createDataFrame(order_items_df2)

# Write Spark DataFrame to delta
order_items_spark_df.write.format("delta").mode("overwrite").option("header", "true").save("dbfs:/mnt/olist-silver/products")
order_items_spark_df.write.format("delta").mode("overwrite").option("header", "true").save("dbfs:/mnt/olist-silver/order_items")
```

▸ (4) Spark Jobs

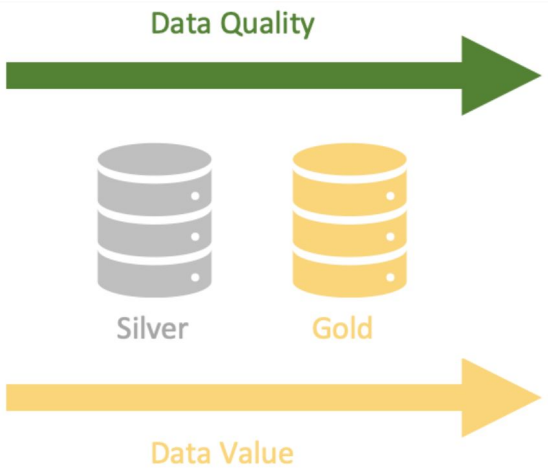
▸ order_items_spark_df: pyspark.sql.dataframe.DataFrame = [product_id: string, product_category_name: string ... 8 more fields]

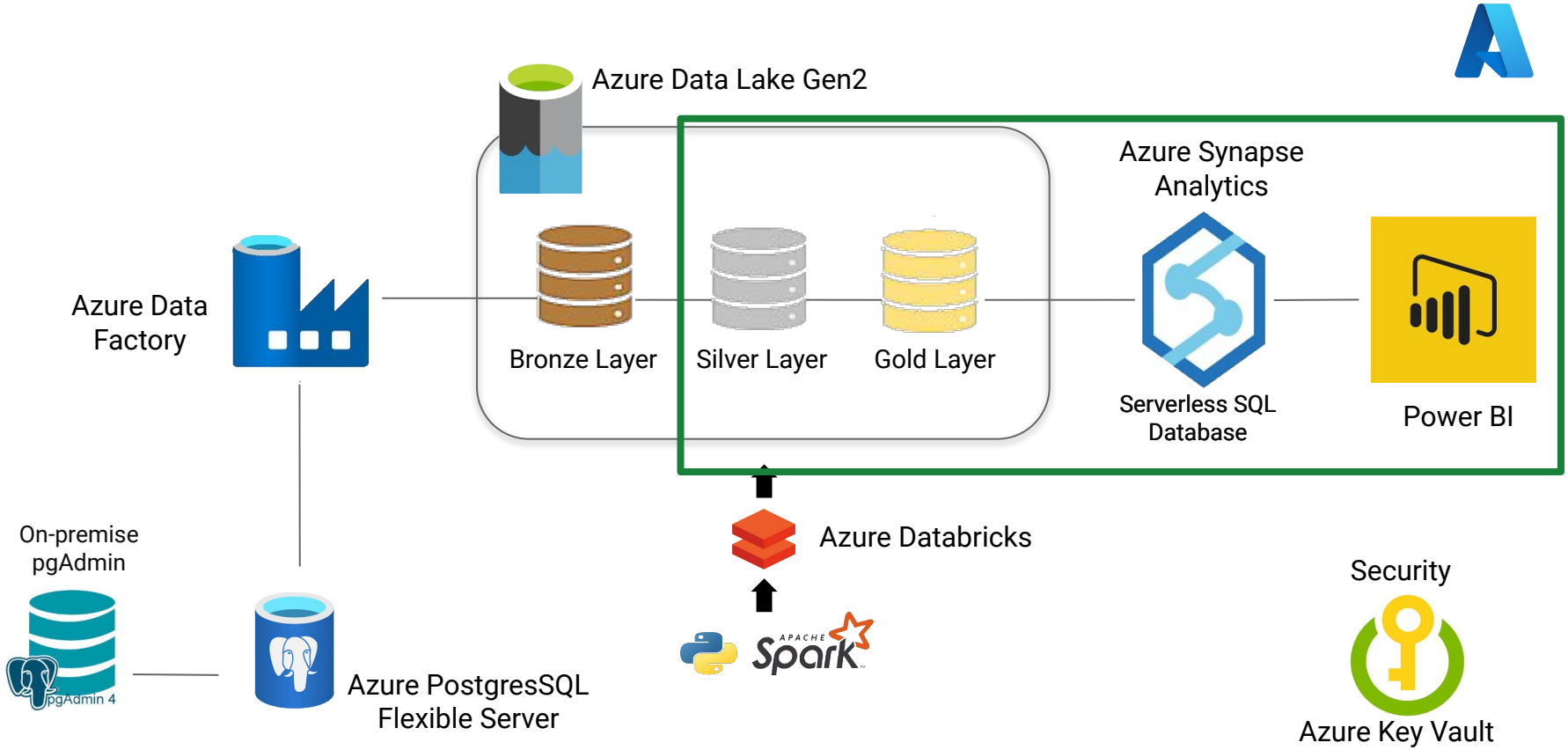
- **Delta format:**
  - Better data management



Data Loading

DATA TRANSFORMATION #2





Full Azure Cloud-Based Data Flow Architecture

Silver to Gold Transformation

Created new workspace for Gold transformation

Workspace >

Shared ☆

Create

Name	Type	Owner	Created at
gold_transformation	Notebook	DORAIRAJ BHUVANESHWARI	2024-05-01 23:22:58
olist-silver-transform	Notebook	DORAIRAJ BHUVANESHWARI	2024-05-03 11:58:54

Extraction and Ingestion of Silver Data

```
# Access the dataframe by name
census_df = dataframes[0]
customers_df = dataframes[1]
geolocations_df = dataframes[2]
order_items_df = dataframes[3]
order_payments_df = dataframes[4]
reviews_full_df = dataframes[5]
orders_df = dataframes[6]
products_df = dataframes[7]
sellers_df = dataframes[8]


census_df = census_df.toPandas()
customers_df = customers_df.toPandas()
geolocations_df = geolocations_df.toPandas()
order_items_df = order_items_df.toPandas()
order_payments_df = order_payments_df.toPandas()
reviews_full_df = reviews_full_df.toPandas()
orders_df = orders_df.toPandas()
products_df = products_df.toPandas()
sellers_df = sellers_df.toPandas()
```

Connect all the silver files data (Delta format) from Data Lake to Databricks, and block convert them from Spark DataFrame to Pandas DataFrame

Translation of Text Reviews

Azure text translation service API

Portuguese to English

A Section of Translation Code

```python

    tracker = []
    body = []
    if not is_null_or_empty(row['review_comment_title']):
        tracker.append('review_comment_title')
        body.append({
            'text': row['review_comment_title']
        })
    if not is_null_or_empty(row['review_comment_message']):
        tracker.append('review_comment_message')
        body.append({
            'text': row['review_comment_message']
        })

    try:
        response = requests.post(constructed_url, params=params, headers=headers, json=body)
        response.raise_for_status()
        response_content = json.loads(response.text)
        review_comment_title_translated = None
        review_comment_title_index = find_index_in_list(tracker, 'review_comment_title')
        review_comment_message_tr_translated = None
        review_comment_message_tr_index = find_index_in_list(tracker, 'review_comment_message')
        if review_comment_title_index != -1:
            review_comment_title_translated = response_content[review_comment_title_index]['translations'][0]['text']
        if review_comment_message_tr_index != -1:
            review_comment_message_tr_translated = response_content[review_comment_message_tr_index]['translations'][0]['text']

        return {"status_code": response.status_code if 'response' in locals() else None,
                "processed_at": datetime.now(timezone.utc),
```

Translation Code

1. Point to location of source data and load into a dataframe
2. Create source and destination tables to input data
3. Process data using Azure Translator API - translate the review comment column
4. Handle the API requests and responses in chunks of 5 reviews per time and throw error messages if any
5. Save the destination table

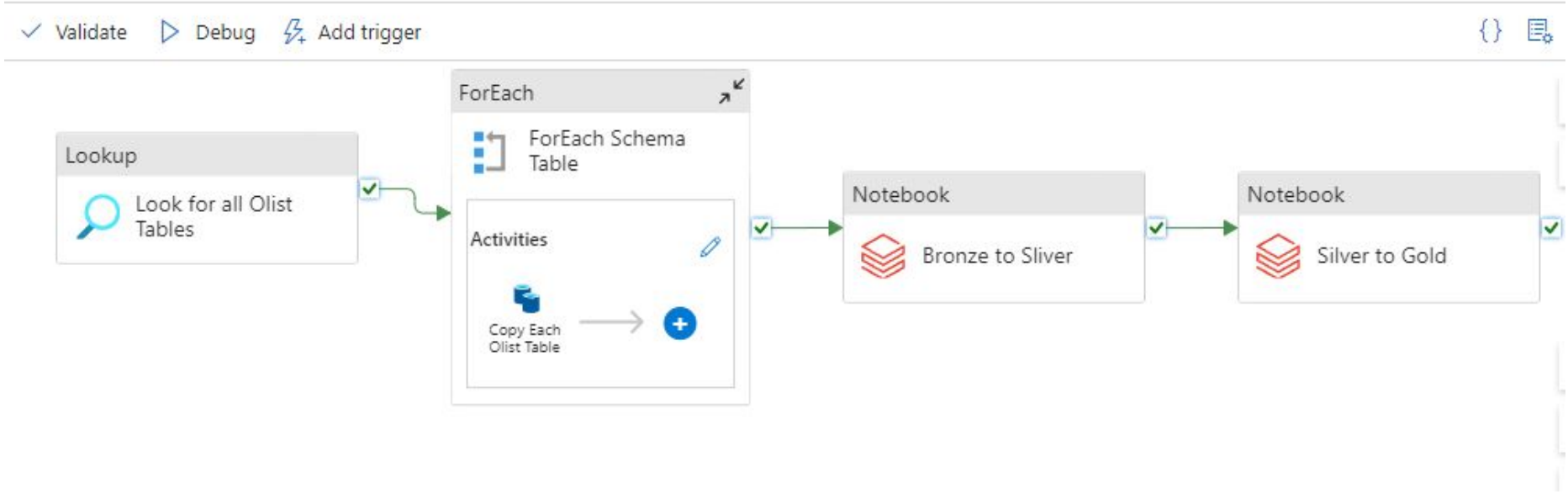
Reducing Data Size to a Sample of 1000

```
# Randomly sample 1000 rows from reviews_text
reviews_text = reviews_text.sample(n=1000, random_state=42)
```

review_comment_message
O Produto é bom, mas não deveria ter espaços tão grandes nos buracos, muriçocas podem atra
super recomendo muito bom
so tem demorado mais que o normal as estregas do site depois que comecaram a vender com
Loja horrível, não entregou o meu produto até hoje.
Mudo minha opinião assim que receber o produto. Consta no site que o produto foi entregue,
O frete foi muito caro. Não valeu a pena. Porque eu tive que retirar a mercadoria na agência do co
Cumpriu e respeitou o consumidor.
Não sei se recomendo. O produto foi marcado no site para um tipo de celular mas quando chegou
AMEI O GORRO, É LINDO E COM ÓTIMO PREÇO. RECOMENDO SEM DÚVIDA!
Pontualidade e ótimos produtos.
Tudo de acordo com o combinado.
Os produtos vendidos através do stark são muito bons e sempre de qualidade.
Recebi o produto certo e antes do prazo. Valeu stark, targaryen/relojoaria nishimoto e correios.
Gostei muito do produto, excelente

review_comment_message_tr_translated
The product is good, but it shouldn't have such large spaces in the holes, sea urchins can go through them, although, with difficulty. A...
I highly recommend very good
It has only taken longer than usual to start selling with partners, other than that nd more to talk about, it was delivered on time, the pr...
Horrible store, didn't deliver my product until today.
I change my opinion as soon as I receive the product. It says on the website that the product was delivered, but I didn't receive it.
Shipping was very expensive. It wasn't worth it. Because I had to pick up the goods at the post office.
Fulfilled and respected the consumer.
I don't know if I recommend it. The product was marked on the website for a type of cell phone, but when it arrived it did not fit on th...
I LOVED THE BEANIE, IT'S BEAUTIFUL AND AT A GREAT PRICE. I RECOMMEND IT WITHOUT A DOUBT!
Punctuality and great products.
Everything according to the agreement.
The products sold through stark are very good and always of quality.
I received the right product and ahead of schedule. Thanks stark, targaryen/nishimoto watchmaking and post offices.
I really liked the product, excellent

Data Factory Progress



Azure Synapse for Sentiment Analysis and Create SQL Queries – Loading

Workspace | Linked

Filter resources by name

- Azure Data Lake Storage Gen2 3
  - olist-synapseanalytics (Primary - oli...
    - olist-gold (Primary)
    - olist-bronze
    - olist-silver
  - (Attached Containers)
  - PullAzureDataLakeStorage (olistco...
    - olist-bronze
    - olist-gold
    - olist-silver

New SQL script | New data flow | New integration dataset | Upload | Download | New folder | More

olist-gold

Name	Last Modified	Content Type
census	5/9/2024, 9:10:17 PM	Folder
customers	5/9/2024, 9:10:28 PM	Folder
geolocation	5/9/2024, 9:10:24 PM	Folder
order_items	5/9/2024, 9:10:11 PM	Folder
order_payments	5/9/2024, 9:10:20 PM	Folder
orders	5/9/2024, 9:10:06 PM	Folder
products	5/9/2024, 9:10:15 PM	Folder
reviews_en		Folder
reviews_fu		Folder
sellers		Folder
sentiment		Folder
zzzzz	5/11/2024, 11:09:20 AM	Folder

Showing 1 to

- New SQL script
  - Select TOP 100 rows
  - Create external table
  - Bulk load
- New data flow
- New integration dataset
- Copy link
- Manage access...
- Rename...
- Download

Azure Synapse for Sentiment Analysis (Machine Learning Made Easy)

Data

Workspace | Linked

Filter resources by name

- Lake database 2
  - olist_database
    - Tables
      - census
      - customers
      - geolocations
      - order_items
      - order_payments
      - orders
      - products
      - reviews_eng
      - reviews_full
      - sellers
  - olist_new
- SQL database
  - olist_gold (SQL)

Joins | olist-gold

New SQL script | New data flow | New integratio

olist-gold

Name	Last Modified
census	5/9/2024, 9:10:17 P
custom	5/9/2024, 9:10:28 P
geoloc	5/9/2024, 9:10:24 P
order_	5/9/2024, 9:10:11 P
order_	5/9/2024, 9:10:20 P
orders	5/9/2024, 9:10:06 P
produc	5/9/2024, 9:10:15 P
review	5/9/2024, 6:04:20 P
review	5/9/2024, 9:10:31 P
sellers	5/9/2024, 9:10:26 P
sentiment	5/11/2024, 10:04:5(
	5/11/2024, 11:09:2(

New SQL script
New data flow
New integration dataset
Copy link
Manage access...
Rename...
Download
Delete
Properties...

New SQL script
New notebook
Machine Learning

Train a new model
Predict with a model

Predict with a model

reviews_full

Choose a pre-trained model

Azure Cognitive Services

This experience allows you to enrich the selected dataset wi models.

Anomaly Detector

Anomaly detection is the identification of rare items, events or observations which raise suspicions by differing significantly from the majority of the data. Learn more

Sentiment Analysis

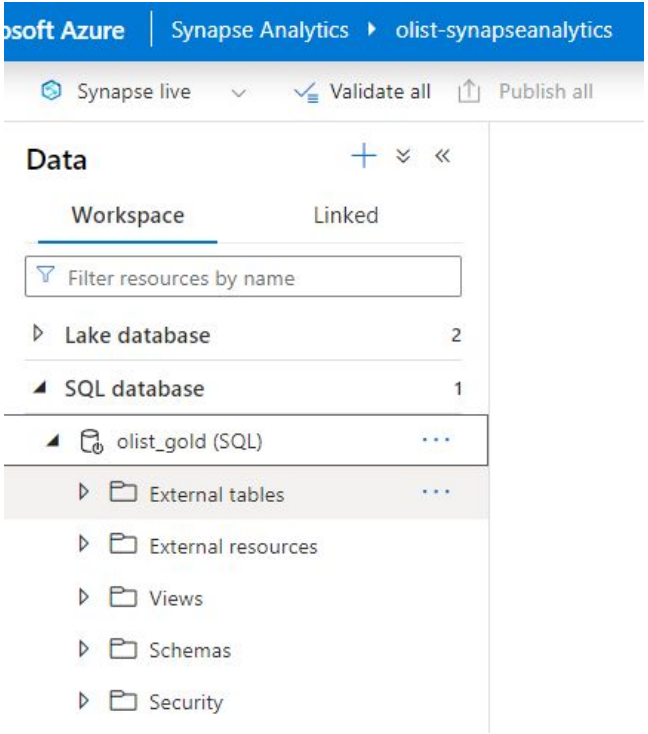
Evaluates the sentiment (positive/negative/neutral) of a text and also returns the probability (score) of the sentiment. Learn more

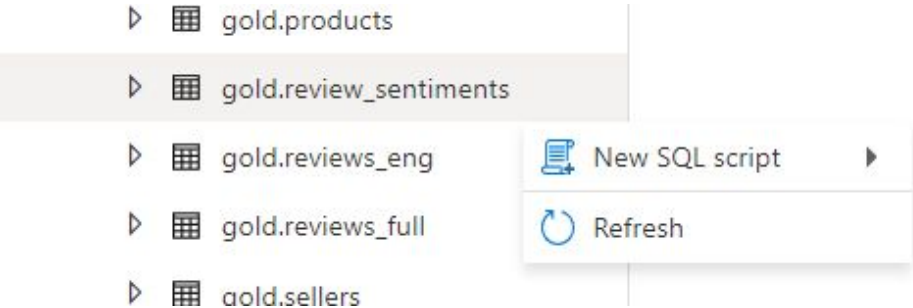
Continue | Cancel

Azure Synapse for Sentiment Analysis

	review_comment_message_tr_translated	sentiment
5ae	The company delivers within what it promises, on the last day of the deadline, it takes too long, it does not delight the customer, as it only does the basic obligation.	negative
45e	The product was delivered much earlier than expected. Great agility.	positive
066	Satisfied with the result.	positive
95	ON-TIME DELIVERY.. RECOMMEND!	positive
14	Arrived ahead of schedule and in perfect condition	positive
00	Great product, good prices and fast delivery.	positive
c38	I loved the product	positive

Azure Synapse for SQL Queries (Serverless SQL database)

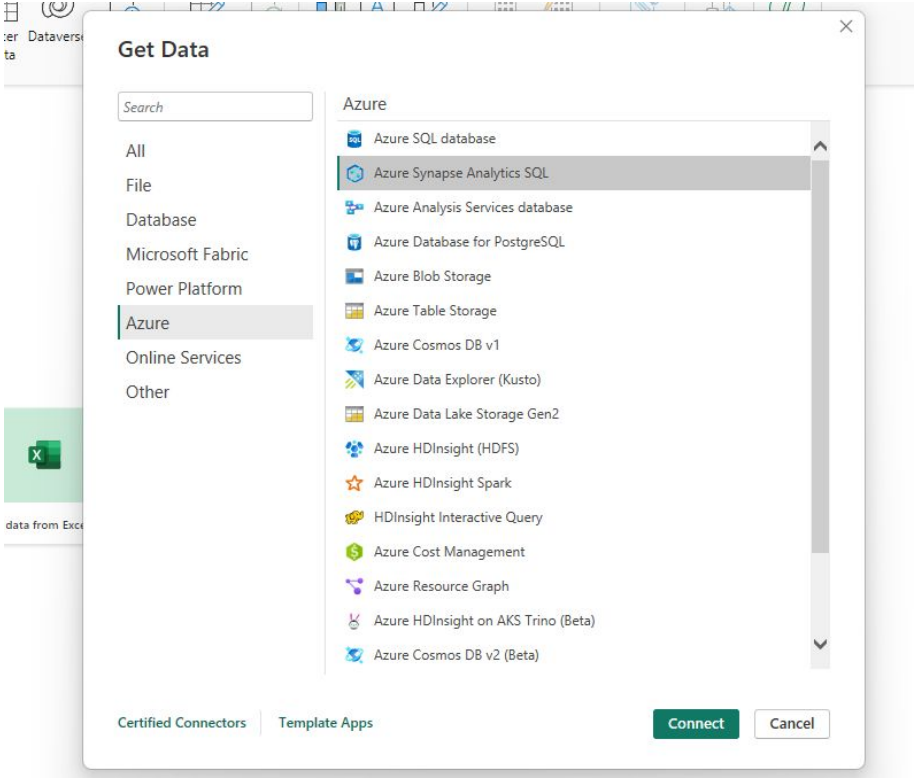


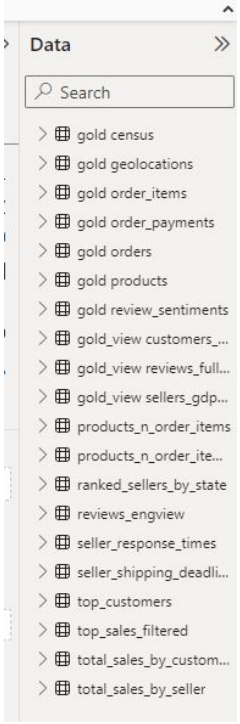


SQL Join Creation

```sql
-- Create the view gold.reviews_full_joinec
CREATE VIEW gold.reviews_full_joined AS
SELECT
    r.*,
    oi.product_id,
    oi.seller_id
FROM
    gold.reviews_eng AS r
JOIN
    (
        SELECT DISTINCT
            oi.product_id,
            oi.seller_id,
            oi.order_id
        FROM
            gold.order_item AS oi
    ) AS oi
ON
    r.order_id = oi.order_id;
```

Loading Tables and View to Power BI





Collaboration on Power Bi Service at powerbi.com

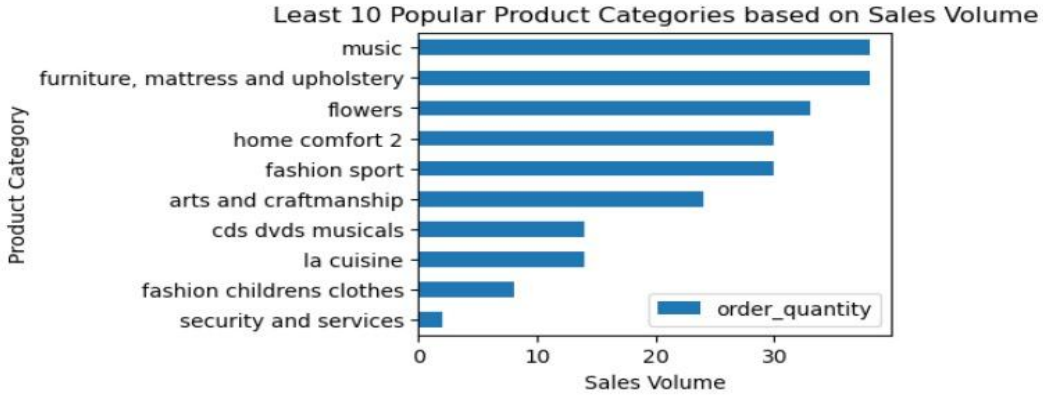
Power BI

- Brazil's leading e-commerce marketplace, connecting small businesses to customers nationwide

- **BI goals:**
  - Increase sales turnover
  - Investigation of:
    - Sellers
    - Products
    - Customers

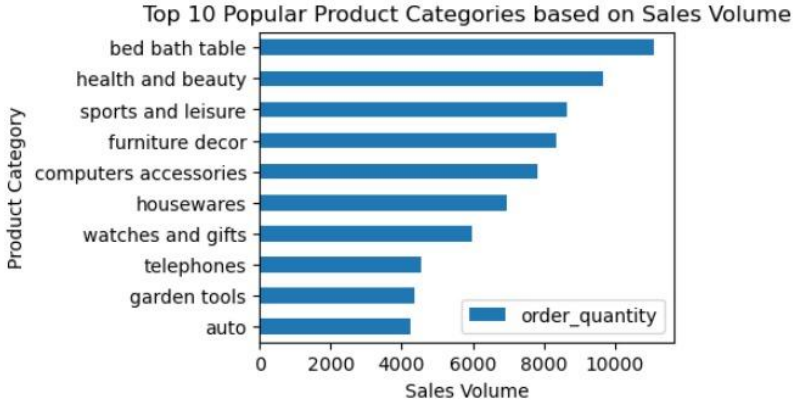
Functional Testing (1)

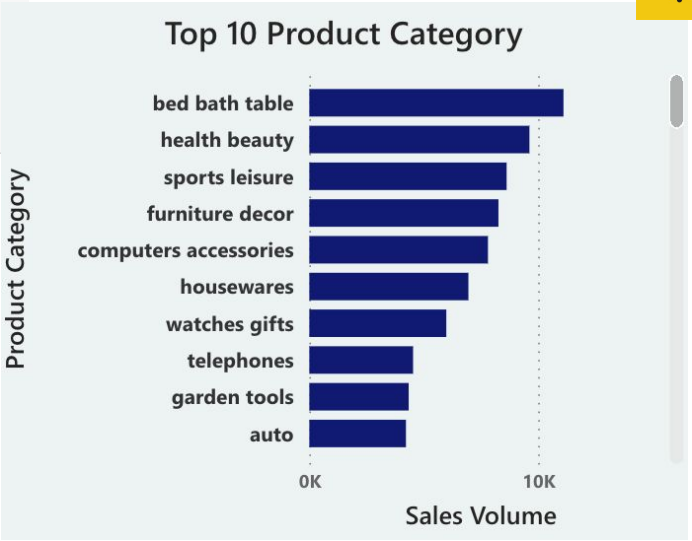
```python
# Select top 10 categories
least10_product_category = product_least_sales.head(10)

# Set the axis
fig, ax = plt.subplots(figsize=(4, 3))

# Plot the barplot with custom x and y axis
least10_product_category.sort_values(by='order_quantity').plot(x='product_category_name_english',
                                                                y='order_quantity', ax=ax, kind='barh')

plt.xlabel('Sales Volume')
plt.ylabel('Product Category')
plt.title("Least 10 Popular Product Categories based on Sales Volume")
plt.show()
```



Functional Testing (2)

```python
# Select top 10 categories
top10_product_category = category_sales.head(10)

# Set the axis
fig, ax = plt.subplots(figsize=(4, 3))

# Plot the barplot with custom x and y axis
top10_product_category.sort_values(by='order_quantity').plot(x='product_category_name_english',
                                                              y='order_quantity', ax=ax, kind='barh')

plt.xlabel('Sales Volume')
plt.ylabel('Product Category')
plt.title("Top 10 Popular Product Categories based on Sales Volume")
plt.show()
```





Thank You

Q&A