

```
!pip install cassandra-driver pandas requests # Install required libraries
```

```
Collecting cassandra-driver
  Downloading cassandra_driver-3.29.2-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (6.2 kB)
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (2.2.2)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (2.32.3)
Collecting geomet<0.3,>=0.1 (from cassandra-driver)
  Downloading geomet-0.2.1.post1-py3-none-any.whl.metadata (1.0 kB)
Requirement already satisfied: numpy>=1.23.2 in /usr/local/lib/python3.11/dist-packages (from pandas) (2.0.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests) (2.4.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests) (2025.4.26)
Requirement already satisfied: click in /usr/local/lib/python3.11/dist-packages (from geomet<0.3,>=0.1->cassandra-driver) (8.1.8)
Requirement already satisfied: six in /usr/local/lib/python3.11/dist-packages (from geomet<0.3,>=0.1->cassandra-driver) (1.17.0)
  Downloading cassandra_driver-3.29.2-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (3.9 MB)
    3.9/3.9 MB 45.1 MB/s eta 0:00:00
  Downloading geomet-0.2.1.post1-py3-none-any.whl (18 kB)
Installing collected packages: geomet, cassandra-driver
Successfully installed cassandra-driver-3.29.2 geomet-0.2.1.post1
```

```
from cassandra.cluster import Cluster
from cassandra.auth import PlainTextAuthProvider
import json

# This secure connect bundle is autogenerated when you download your SCB,
# if yours is different update the file name below
cloud_config= {
    'secure_connect_bundle': 'secure-connect-warehouse-sales.zip'
}
```

```
# This token JSON file is autogenerated when you download your token,
# if yours is different update the file name below
with open("warehouse_sales-token.json") as f:
    secrets = json.load(f)

CLIENT_ID = secrets["clientId"]
CLIENT_SECRET = secrets["secret"]

auth_provider = PlainTextAuthProvider(CLIENT_ID, CLIENT_SECRET)
cluster = Cluster(cloud=cloud_config, auth_provider=auth_provider)
session = cluster.connect()

row = session.execute("select release_version from system.local").one()
if row:
    print(row[0])
else:
    print("An error occurred.")
```

```
WARNING:cassandra.cluster:Downgrading core protocol version from 66 to 65 for 3b130ba0-5173-460a-a0a8-35fd4fbd9fdb-eu-west-1.db.astra.da
WARNING:cassandra.cluster:Downgrading core protocol version from 65 to 5 for 3b130ba0-5173-460a-a0a8-35fd4fbd9fdb-eu-west-1.db.astra.dat
WARNING:cassandra.cluster:Downgrading core protocol version from 5 to 4 for 3b130ba0-5173-460a-a0a8-35fd4fbd9fdb-eu-west-1.db.astra.data
4.0.11-3a52ecb0d31d
```

```
import requests
import pandas as pd

url = "https://data.montgomerycountymd.gov/api/views/v76h-r7br/rows.json?accessType=DOWNLOAD"
response = requests.get(url)
data = response.json()

# Convert to Pandas DataFrame
rows = data.get('data', [])
columns = [col['name'] for col in data['meta']['view']['columns']]
df = pd.DataFrame(rows, columns=columns)
```


```
df.columns
```

```
Index(['sid', 'id', 'position', 'created_at', 'created_meta', 'updated_at',
       'updated_meta', 'meta', 'YEAR', 'MONTH', 'SUPPLIER', 'ITEM CODE',
       'ITEM DESCRIPTION', 'ITEM TYPE', 'RETAIL SALES', 'RETAIL TRANSFERS',
```

```
    'WAREHOUSE SALES'],
    dtype='object')
```

```
from cassandra.query import BatchStatement
import pandas as pd
import uuid
from datetime import datetime
import requests
```

```
session.execute(f"""
CREATE TABLE IF NOT EXISTS sales.Bronze (
    id UUID PRIMARY KEY,
    year INT,
    month INT,
    supplier TEXT,
    item_code TEXT,
    item_description TEXT,
    item_type TEXT,
    retail_sales DECIMAL,
    retail_transfers DECIMAL,
    warehouse_sales DECIMAL,
    created_at TIMESTAMP
);
""")
```

```
 <cassandra.cluster.ResultSet at 0x7a090445c3d0>
```

```
insert_stmt = session.prepare(f"""
INSERT INTO sales.Bronze (
    id, year, month, supplier, item_code, item_description,
    item_type, retail_sales, retail_transfers, warehouse_sales, created_at
) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
""")
```

```
batch_size = 1000
batch = BatchStatement()
inserted_count = 0
```

```
from tqdm import tqdm
def insert_rows(dataframe, total_rows, batch_size):
    inserted = 0
    with tqdm(total=total_rows, desc="Inserting rows") as pbar:
        for i in range(0, total_rows, batch_size):
            batch = BatchStatement()
            end_idx = min(i + batch_size, total_rows)

            for _, row in dataframe.iloc[i:end_idx].iterrows():
                try:
                    batch.add(insert_stmt, (
                        uuid.uuid4(),
                        int(row['YEAR']),
                        int(row['MONTH']),
                        str(row['SUPPLIER']),
                        str(row['ITEM CODE']),
                        str(row['ITEM DESCRIPTION']),
                        str(row['ITEM TYPE']),
                        float(row['RETAIL SALES']),
                        float(row['RETAIL TRANSFERS']),
                        float(row['WAREHOUSE SALES']),
                        datetime.now()
                    ))
                except Exception as e:
                    print(f"Skipping row {i} due to error: {str(e)}")
                    continue

            session.execute(batch)
            inserted += (end_idx - i)
            pbar.update(end_idx - i)

    return inserted

# Execute the insertion
```

```
success_count = insert_rows(df, 120000, batch_size)
print(f"Total rows inserted: {success_count}")
```

```

Inserting rows: 54% | 65000/120000 [00:41<00:33, 1647.49it/s]WARNING:cassandra.protocol:Server warning: Batch for ["33623:
Inserting rows: 55% | 66000/120000 [00:41<00:34, 1581.21it/s]WARNING:cassandra.protocol:Server warning: Batch for ["336231
Inserting rows: 56% | 67000/120000 [00:42<00:33, 1570.34it/s]WARNING:cassandra.protocol:Server warning: Batch for ["336231
Inserting rows: 57% | 68000/120000 [00:43<00:33, 1559.89it/s]WARNING:cassandra.protocol:Server warning: Batch for ["33623:
Inserting rows: 57% | 69000/120000 [00:43<00:32, 1576.17it/s]WARNING:cassandra.protocol:Server warning: Batch for ["33623:
Inserting rows: 58% | 70000/120000 [00:44<00:31, 1593.80it/s]WARNING:cassandra.protocol:Server warning: Batch for ["33623:
Inserting rows: 59% | 71000/120000 [00:44<00:31, 1570.54it/s]WARNING:cassandra.protocol:Server warning: Batch for ["33623:
Inserting rows: 60% | 72000/120000 [00:45<00:30, 1562.66it/s]WARNING:cassandra.protocol:Server warning: Batch for ["336231
Inserting rows: 61% | 73000/120000 [00:46<00:30, 1547.38it/s]WARNING:cassandra.protocol:Server warning: Batch for ["336231
Inserting rows: 62% | 74000/120000 [00:46<00:29, 1538.51it/s]WARNING:cassandra.protocol:Server warning: Batch for ["33623:
Inserting rows: 62% | 75000/120000 [00:47<00:29, 1527.33it/s]WARNING:cassandra.protocol:Server warning: Batch for ["33623:
Inserting rows: 63% | 76000/120000 [00:48<00:28, 1538.69it/s]WARNING:cassandra.protocol:Server warning: Batch for ["33623:
Inserting rows: 64% | 77000/120000 [00:48<00:27, 1543.72it/s]WARNING:cassandra.protocol:Server warning: Batch for ["33623:
Inserting rows: 65% | 78000/120000 [00:49<00:26, 1577.51it/s]WARNING:cassandra.protocol:Server warning: Batch for ["336231
Inserting rows: 66% | 79000/120000 [00:50<00:25, 1596.85it/s]WARNING:cassandra.protocol:Server warning: Batch for ["336231
Inserting rows: 67% | 80000/120000 [00:50<00:25, 1570.74it/s]WARNING:cassandra.protocol:Server warning: Batch for ["33623:
Inserting rows: 68% | 81000/120000 [00:51<00:24, 1568.27it/s]WARNING:cassandra.protocol:Server warning: Batch for ["33623:
Inserting rows: 68% | 82000/120000 [00:51<00:22, 1685.68it/s]WARNING:cassandra.protocol:Server warning: Batch for ["33623:
Inserting rows: 69% | 83000/120000 [00:52<00:22, 1659.98it/s]WARNING:cassandra.protocol:Server warning: Batch for ["33623:
Inserting rows: 70% | 84000/120000 [00:53<00:20, 1740.46it/s]WARNING:cassandra.protocol:Server warning: Batch for ["336231
Inserting rows: 71% | 85000/120000 [00:53<00:20, 1668.27it/s]WARNING:cassandra.protocol:Server warning: Batch for ["336231
Inserting rows: 72% | 86000/120000 [00:54<00:20, 1640.72it/s]WARNING:cassandra.protocol:Server warning: Batch for ["33623:
Inserting rows: 72% | 87000/120000 [00:54<00:20, 1610.55it/s]WARNING:cassandra.protocol:Server warning: Batch for ["33623:
Inserting rows: 73% | 88000/120000 [00:55<00:19, 1601.82it/s]WARNING:cassandra.protocol:Server warning: Batch for ["33623:
Inserting rows: 74% | 89000/120000 [00:56<00:19, 1573.38it/s]WARNING:cassandra.protocol:Server warning: Batch for ["33623:
Inserting rows: 75% | 90000/120000 [00:56<00:19, 1538.40it/s]WARNING:cassandra.protocol:Server warning: Batch for ["336231
Inserting rows: 76% | 91000/120000 [00:57<00:18, 1541.69it/s]WARNING:cassandra.protocol:Server warning: Batch for ["336231
Inserting rows: 77% | 92000/120000 [00:58<00:18, 1548.20it/s]WARNING:cassandra.protocol:Server warning: Batch for ["33623:
Inserting rows: 78% | 93000/120000 [00:58<00:17, 1556.29it/s]WARNING:cassandra.protocol:Server warning: Batch for ["33623:
Inserting rows: 78% | 94000/120000 [00:59<00:15, 1672.16it/s]WARNING:cassandra.protocol:Server warning: Batch for ["33623:
Inserting rows: 79% | 95000/120000 [01:00<00:15, 1618.73it/s]WARNING:cassandra.protocol:Server warning: Batch for ["33623:
Inserting rows: 80% | 96000/120000 [01:00<00:15, 1594.82it/s]WARNING:cassandra.protocol:Server warning: Batch for ["336231
Inserting rows: 81% | 97000/120000 [01:01<00:14, 1586.83it/s]WARNING:cassandra.protocol:Server warning: Batch for ["336231
Inserting rows: 82% | 98000/120000 [01:01<00:13, 1603.34it/s]WARNING:cassandra.protocol:Server warning: Batch for ["33623:
Inserting rows: 82% | 99000/120000 [01:02<00:13, 1597.88it/s]WARNING:cassandra.protocol:Server warning: Batch for ["33623:
Inserting rows: 83% | 100000/120000 [01:03<00:12, 1571.13it/s]WARNING:cassandra.protocol:Server warning: Batch for ["3362:
Inserting rows: 84% | 101000/120000 [01:03<00:12, 1534.14it/s]WARNING:cassandra.protocol:Server warning: Batch for ["3362:
Inserting rows: 85% | 102000/120000 [01:04<00:11, 1529.87it/s]WARNING:cassandra.protocol:Server warning: Batch for ["33623
Inserting rows: 86% | 103000/120000 [01:05<00:11, 1545.23it/s]WARNING:cassandra.protocol:Server warning: Batch for ["33623
Inserting rows: 87% | 104000/120000 [01:05<00:09, 1651.88it/s]WARNING:cassandra.protocol:Server warning: Batch for ["3362:
Inserting rows: 88% | 105000/120000 [01:06<00:09, 1646.90it/s]WARNING:cassandra.protocol:Server warning: Batch for ["3362:
Inserting rows: 88% | 106000/120000 [01:06<00:08, 1622.28it/s]WARNING:cassandra.protocol:Server warning: Batch for ["3362:
Inserting rows: 89% | 107000/120000 [01:07<00:08, 1607.01it/s]WARNING:cassandra.protocol:Server warning: Batch for ["3362:
Inserting rows: 90% | 108000/120000 [01:08<00:07, 1576.39it/s]WARNING:cassandra.protocol:Server warning: Batch for ["33623
Inserting rows: 91% | 109000/120000 [01:08<00:07, 1556.99it/s]WARNING:cassandra.protocol:Server warning: Batch for ["33623
Inserting rows: 92% | 110000/120000 [01:09<00:06, 1546.79it/s]WARNING:cassandra.protocol:Server warning: Batch for ["3362:
Inserting rows: 92% | 111000/120000 [01:10<00:05, 1550.88it/s]WARNING:cassandra.protocol:Server warning: Batch for ["3362:
Inserting rows: 93% | 112000/120000 [01:10<00:05, 1540.77it/s]WARNING:cassandra.protocol:Server warning: Batch for ["3362:
Inserting rows: 94% | 113000/120000 [01:11<00:04, 1536.83it/s]WARNING:cassandra.protocol:Server warning: Batch for ["3362:
Inserting rows: 95% | 114000/120000 [01:12<00:03, 1546.05it/s]WARNING:cassandra.protocol:Server warning: Batch for ["33623
Inserting rows: 96% | 115000/120000 [01:12<00:03, 1567.86it/s]WARNING:cassandra.protocol:Server warning: Batch for ["33623
Inserting rows: 97% | 116000/120000 [01:13<00:02, 1560.47it/s]WARNING:cassandra.protocol:Server warning: Batch for ["3362:
Inserting rows: 98% | 117000/120000 [01:14<00:01, 1537.93it/s]WARNING:cassandra.protocol:Server warning: Batch for ["3362:
Inserting rows: 98% | 118000/120000 [01:14<00:01, 1531.89it/s]WARNING:cassandra.protocol:Server warning: Batch for ["3362:
Inserting rows: 99% | 119000/120000 [01:15<00:00, 1639.36it/s]WARNING:cassandra.protocol:Server warning: Batch for ["3362:
Inserting rows: 100% | 120000/120000 [01:15<00:00, 1584.18it/s]Total rows inserted: 120000

```

```

result = session.execute(f"SELECT COUNT(*) FROM sales.Bronze").one()
print(f"\nSuccessfully inserted {success_count} rows")
cluster.shutdown()

```

```
WARNING:cassandra.protocol:Server warning: Aggregation query used without partition key
```

Successfully inserted 120000 rows

```
df = session.execute(f"SELECT * FROM sales.Bronze")
```

```

from pyspark.sql import SparkSession
from pyspark.sql.types import *
from decimal import Decimal
import uuid
from datetime import datetime

```

```

# Create Spark Session
spark = SparkSession.builder \
    .appName("CassandraToSpark") \

```

```

.getOrCreate()

# Function to properly extract data from Cassandra rows
def convert_cassandra_row(row):
    return {
        'id': str(row.id) if row.id else None,
        'created_at': row.created_at if row.created_at else None,
        'item_code': row.item_code if row.item_code else None,
        'item_description': row.item_description if row.item_description else None,
        'item_type': row.item_type if row.item_type else None,
        'month': int(row.month) if row.month else None,
        'retail_sales': float(row.retail_sales) if row.retail_sales else None,
        'retail_transfers': float(row.retail_transfers) if row.retail_transfers else None,
        'supplier': row.supplier if row.supplier else None,
        'warehouse_sales': float(row.warehouse_sales) if row.warehouse_sales else None,
        'year': int(row.year) if row.year else None
    }

# Convert all rows
data = [convert_cassandra_row(row) for row in df]

# Define schema
schema = StructType([
    StructField("id", StringType(), True),
    StructField("created_at", TimestampType(), True),
    StructField("item_code", StringType(), True),
    StructField("item_description", StringType(), True),
    StructField("item_type", StringType(), True),
    StructField("month", IntegerType(), True),
    StructField("retail_sales", DoubleType(), True),
    StructField("retail_transfers", DoubleType(), True),
    StructField("supplier", StringType(), True),
    StructField("warehouse_sales", DoubleType(), True),
    StructField("year", IntegerType(), True)
])

# Create DataFrame
spark_df = spark.createDataFrame(data, schema=schema)

# Show results
spark_df.show(5, truncate=False)

```

```

+-----+-----+-----+-----+-----+-----+-----+
|id|created_at|item_code|item_description|item_type|month|retail_sal|
+-----+-----+-----+-----+-----+-----+-----+
|f491cc81-d3da-4ddd-8170-5a0dac0b21d1|2025-05-04 00:54:49.9|83627|BOTA BOX S/BLANC - 3L|WINE|9|96.89|
|02b3eba1-1f15-4d6e-b25d-d3b9d1ce8ce3|2025-05-04 00:54:03.876|325882|GINI SOAVE - 750ML|WINE|7|NULL|
|7c6bec8a-3129-4592-8670-43c85efa31c5|2025-05-04 00:54:08.823|108367|DOM CAVES DE PRIEURE SANCERRE - 750ML|WINE|3|0.8|
|9f5156ea-a84a-4a2c-b508-480cdf8c0d81|2025-05-04 00:54:43.793|2550|ABITA PURPLE HAZE 4/6 NR - 12OZ|BEER|9|2.25|
|a5b568b6-9cac-41c9-9464-5e43f0a7eaba|2025-05-04 00:54:27.858|313598|RESERVE DE LA SAURINE RSE - 750ML|WINE|7|NULL|
+-----+-----+-----+-----+-----+-----+-----+
only showing top 5 rows

```

▼ Bronze Layer

```

# Cache the DataFrame for better performance
spark_df.cache()

# 1. Basic Exploration
print("Record Count:", spark_df.count())
spark_df.printSchema()
spark_df.show(5, vertical=True)

# 2. Data Quality Checks
from pyspark.sql.functions import col, count, when, isnull

# Null check per column
null_checks = spark_df.select(
    [count(when(isnull(c), c)).alias(c) for c in spark_df.columns]
)
null_checks.show()

# 3. Basic Cleaning (Bronze Layer Standards)
bronze_df = spark_df.withColumnRenamed("supplier", "supplier_name") \

```

```
.withColumn("ingestion_timestamp", current_timestamp()) \
.withColumn("data_source", lit("cassandra.sales.Bronze"))
```

```
warehouse_sales | 25.0
year             | 2017
-RECORD 1-----
id               | 02b3eba1-1f15-4d6...
created_at      | 2025-05-04 00:54:...
item_code       | 325882
item_description | GINI SOAVE - 750ML
item_type       | WINE
month           | 7
retail_sales    | NULL
retail_transfers | NULL
supplier        | BACCHUS IMPORTERS...
warehouse_sales | 1.0
year            | 2020
-RECORD 2-----
id               | 7c6bec8a-3129-459...
created_at      | 2025-05-04 00:54:...
item_code       | 108367
item_description | DOM CAVES DE PRIE...
item_type       | WINE
month           | 3
retail_sales    | 0.8
retail_transfers | NULL
supplier        | BARON FRANCOIS LTD
warehouse_sales | NULL
year            | 2020
-RECORD 3-----
id               | 9f5156ea-a84a-4a2...
created_at      | 2025-05-04 00:54:...
item_code       | 2550
item_description | ABITA PURPLE HAZE...
item_type       | BEER
month           | 9
retail_sales    | 2.25
retail_transfers | 2.0
supplier        | DOPS INC
warehouse_sales | 22.75
year            | 2017
-RECORD 4-----
id               | a5b568b6-9cac-41c...
created_at      | 2025-05-04 00:54:...
item_code       | 313598
item_description | RESERVE DE LA SAU...
item_type       | WINE
month           | 7
retail_sales    | NULL
retail_transfers | NULL
supplier        | LANTERNA DISTRIBU...
warehouse_sales | 3.0
year            | 2017
only showing top 5 rows
```

```
+---+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id|created_at|item_code|item_description|item_type|month|retail_sales|retail_transfers|supplier|warehouse_sales|year|
+---+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 0|          |0|          |0|          |0|          |0|          |0|          |0|
+---+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

Silver Layer

```
from pyspark.sql.functions import col

# Remove invalid rows where 'year', 'month', or key fields are zero
silver_df = spark_df.filter(
    (col("year") != 0) &
    (col("month") != 0) &
    (col("id") != "0") &
    (col("item_code") != "0") &
    (col("item_type") != "0") &
    (col("supplier") != "0")
)
silver_df = silver_df.dropDuplicates(["id"]) \
    .filter(
        (col("retail_sales") >= 0) &
        (col("warehouse sales") >= 0)
    )
```

```

) \
.na.fill({
  "item_description": "Unknown",
  "supplier": "Unknown"
}) \
.filter(col("id").isNotNull())

```

```
silver_df.show()
```

	id	created_at	item_code	item_description	item_type	month	retail_sales	retail_transfers	su
	000285b1-73da-416...	2025-05-04 00:55:...	23024	LANDSHARK LAGER 2...	BEER	11	7.0	8.0	ANHEUSER BUS
	0003da6f-a008-48f...	2025-05-04 00:54:...	342116	HONORO VERA GARNA...	WINE	7	4.35	8.0	REPUBLIC NATIC
	00058f00-cbe3-4a7...	2025-05-04 00:55:...	346692	GAME OF THRONES C...	WINE	11	9.74	8.0	VINTAGE WINE E
	0005f598-5939-41c...	2025-05-04 00:53:...	27545	JOHNNIE WALKER RE...	LIQUOR	1	53.89	48.0	DIAGEO NORTH A
	0009bf55-d12b-429...	2025-05-04 00:55:...	11584	EVAN WILLIAMS GRE...	LIQUOR	12	94.11	92.0	HEAVEN HILL DI
	000d18ab-6a71-4f2...	2025-05-04 00:54:...	5096	BUD 30PK CAN - 12OZ	BEER	7	299.0	282.0	ANHEUSER BUS
	000f7c05-1b53-41f...	2025-05-04 00:55:...	340059	GRAND MOSCATO MAN...	WINE	11	1.54	2.0	DMV DISTRIBUTI
	001908f2-0e6e-4b7...	2025-05-04 00:54:...	312556	CONDE VILLAR VINH...	WINE	7	1.18	3.0	A VINTNERS SEL
	001cc4c1-1d3d-41c...	2025-05-04 00:55:...	335927	CH LA MOULINIERE ...	WINE	11	0.48	1.0	BARON FRANCC
	001cfa8b-caef-441...	2025-05-04 00:54:...	11231	WILD TURKEY 101P ...	LIQUOR	6	37.16	64.0	CAMPARI AMERI
	001facbd-4b27-4e5...	2025-05-04 00:54:...	27499	JOHNNIE WALKER BL...	LIQUOR	7	40.58	32.0	DIAGEO NORTH A
	00206a3a-70ab-497...	2025-05-04 00:55:...	83690	CH ST MICH MER - ...	WINE	11	44.52	54.0	STE MICHELLE W
	002e8342-c22d-43e...	2025-05-04 00:54:...	94978	CLAUSTHALER LAGER...	NON-ALCOHOL	10	0.75	NULL	RELIABLE CHURC
	002eb85e-2bb9-423...	2025-05-04 00:54:...	6025	DELIRIUM NOCTURNU...	BEER	9	0.83	2.0	DC
	003c9f93-9ae6-4f5...	2025-05-04 00:54:...	83390	BONTERRA ZIN - 750ML	WINE	1	1.48	NULL	FETZER VIN
	003d88b4-3de3-44c...	2025-05-04 00:54:...	55247	ERATH OREGON P/NO...	WINE	7	44.68	42.0	STE MICHELLE W
	0042b0a0-a7cb-4b2...	2025-05-04 00:54:...	83151	CH ST JEAN CAL CH...	WINE	3	3.78	3.0	TREASURY WINE
	004597c8-8f79-463...	2025-05-04 00:54:...	34541	CRISTALINO BRUT -...	WINE	7	13.45	9.0	REPUBLIC NATIC
	00466441-bafb-4df...	2025-05-04 00:54:...	359653	RIONDO PROSECCO -...	WINE	8	1.66	3.0	REPUBLIC NATIC
	00494b79-00f5-4a8...	2025-05-04 00:54:...	388327	CHARLES SMITH EVE...	WINE	6	3.84	3.0	CONSTELLATION

only showing top 20 rows

Gold Layer

```

agg1 = silver_df.groupBy("year") \
  .sum("retail_sales") \
  .withColumnRenamed("sum(retail_sales)", "total_retail_sales")
agg1.show()

```

year	total_retail_sales
2020	175472.93000000127
2017	369558.92000000086

```

agg2 = silver_df.groupBy("year", "month", "item_code") \
  .sum("retail_sales") \
  .withColumnRenamed("sum(retail_sales)", "monthly_retail_sales")
agg2.show()

```

year	month	item_code	monthly_retail_sales
2017	7	64742	14.3
2017	9	381756	0.08
2017	10	27646	0.25
2017	6	311407	0.16
2020	1	302562	0.42
2020	3	313629	0.99
2017	8	57312	2.03
2017	6	438804	0.16
2017	6	421758	0.65
2020	3	241318	9.92
2017	9	84599	7.08
2020	1	84204	15.86
2020	7	76031	17.99
2017	8	41002	1.66
2020	7	51153	0.41
2017	7	45163	0.57
2017	10	23612	8.77

2020	1	83370	22.58
2020	1	80911	109.65
2020	7	28741	10.25

+-----+
only showing top 20 rows

```
from pyspark.sql.functions import sum
```

```
agg3 = silver_df.groupBy("item_code") \
    .agg(sum("warehouse_sales").alias("total_warehouse_sales")) \
    .orderBy(col("total_warehouse_sales").desc()) \
    .limit(5)
agg3.show()
```

```
→ +-----+-----+
|item_code|total_warehouse_sales|
+-----+-----+
| 23445| 78217.79164000001|
| 96750| 70369.95832|
| 23886| 60078.04166|
| 96741| 52612.16663|
| 90590| 50479.300010000006|
+-----+-----+
```

```
from pyspark.sql.functions import avg
```

```
agg4 = silver_df.groupBy("item_type") \
    .agg(avg("retail_transfers").alias("avg_retail_transfers"))
agg4.show()
```

```
→ +-----+-----+
| item_type|avg_retail_transfers|
+-----+-----+
| WINE| 14.299368589316593|
| LIQUOR| 44.96824992130938|
| BEER| 39.3356243260729|
| NON-ALCOHOL| 26.171370967741936|
+-----+-----+
```

```
agg5 = silver_df.groupBy("supplier") \
    .agg(
        sum("retail_sales").alias("total_retail_sales"),
        sum("warehouse_sales").alias("total_warehouse_sales")
    )
agg5.show()
```

```
→ +-----+-----+-----+
| supplier| total_retail_sales| total_warehouse_sales|
+-----+-----+-----+
| BARON FRANCOIS LTD| 1274.1900000000005| 867.58334|
| STE MICHELLE WINE...| 5699.2899999999997| 5010.25001|
| AZIZ SHAFI TANNIC...| 0.42000000000000004| 10.0|
| PWSWN INC| 114.41999999999999| 916.9166700000001|
| E & J GALLO WINERY| 44568.199999999998| 58136.08337999999|
| HARVEST IMPORTING...| 0.65| 7.0|
| DMV DISTRIBUTING LLC| 299.25| 527.6666700000001|
| MACK & SCHUHLE INC| 7.1| 226.0|
| HEAVEN HILL DISTI...| 6744.330000000002| 2078.41667|
| PROXIMO SPIRITS INC| 4700.6900000000005| 492.97917|
| DUCKHORN WINE COM...| 691.06| 671.0|
| OENOS LLC| 0.48| 20.0|
| LANTERNA DISTRIBU...| 149.24999999999994| 398.0|
| CLIPPER CITY BREW...| 1786.3300000000004| 5473.75|
| BACARDI USA INC| 13066.839999999997| 3075.2500100000007|
| VICTORY BREWING C...| 155.17000000000002| 1540.75|
| BINDING BRAUEREI ...| 28.71| 211.83333|
| JORDAN VINEYARD| 16.150000000000002| 46.0|
| E M D SALES INC| 18.61| 228.0|
| CHEVAL QUANCARD| 0.73| 1.0|
+-----+-----+-----+
```

only showing top 20 rows

```
agg6 = silver_df.groupBy("item_type").count().withColumnRenamed("count", "item_type_count")
agg6.show()
```

```

+-----+-----+
| item_type | item_type_count |
+-----+-----+
| WINE      | 19147            |
| LIQUOR    | 3350             |
| BEER      | 5327             |
| NON-ALCOHOL | 265             |
+-----+-----+

```

```

from pyspark.sql.window import Window
from pyspark.sql.functions import lag, round

```

```

yearly_sales = silver_df.groupBy("year") \
    .agg(sum("retail_sales").alias("total_sales"))

```

```

window_spec = Window.orderBy("year")

```

```

agg7 = yearly_sales.withColumn("prev_year_sales", lag("total_sales").over(window_spec)) \
    .withColumn("growth_percentage", round(
        ((col("total_sales") - col("prev_year_sales")) / col("prev_year_sales")) * 100, 2
    ))
agg7.show()

```

```

+-----+-----+-----+-----+
| year | total_sales | prev_year_sales | growth_percentage |
+-----+-----+-----+-----+
| 2017 | 369558.92000000086 | NULL | NULL |
| 2020 | 175472.930000000127 | 369558.92000000086 | -52.52 |
+-----+-----+-----+-----+

```

Visualizations

```

import matplotlib.pyplot as plt
import seaborn as sns

```

```

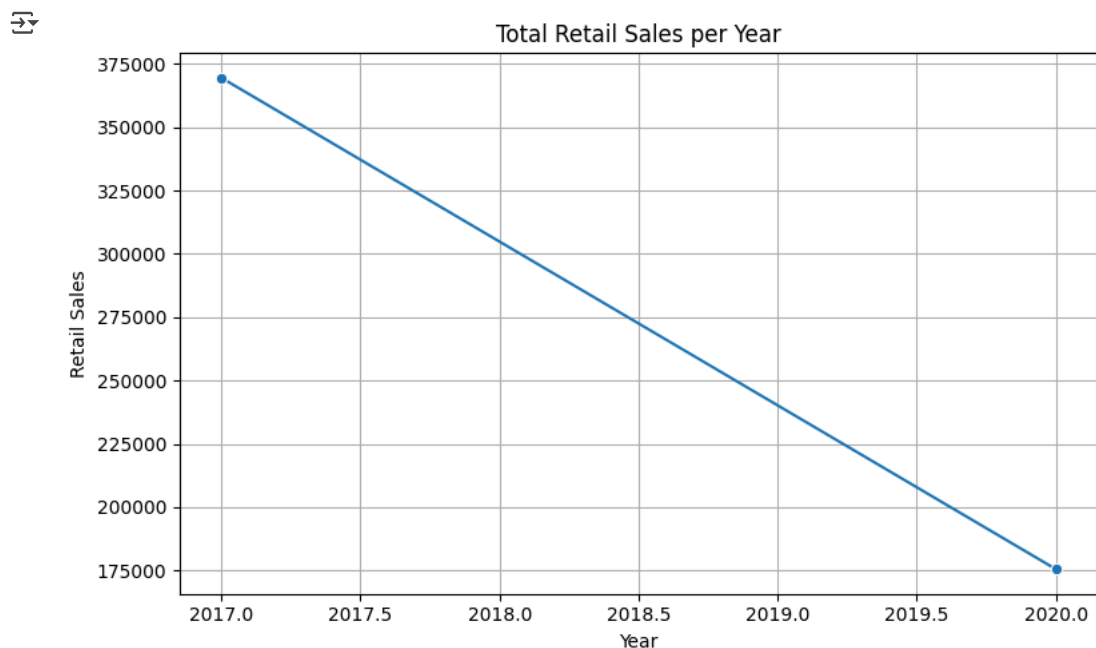
df1 = agg1.toPandas().sort_values("year")

```

```

plt.figure(figsize=(8, 5))
sns.lineplot(data=df1, x="year", y="total_retail_sales", marker="o")
plt.title("Total Retail Sales per Year")
plt.xlabel("Year")
plt.ylabel("Retail Sales")
plt.grid(True)
plt.tight_layout()
plt.show()

```



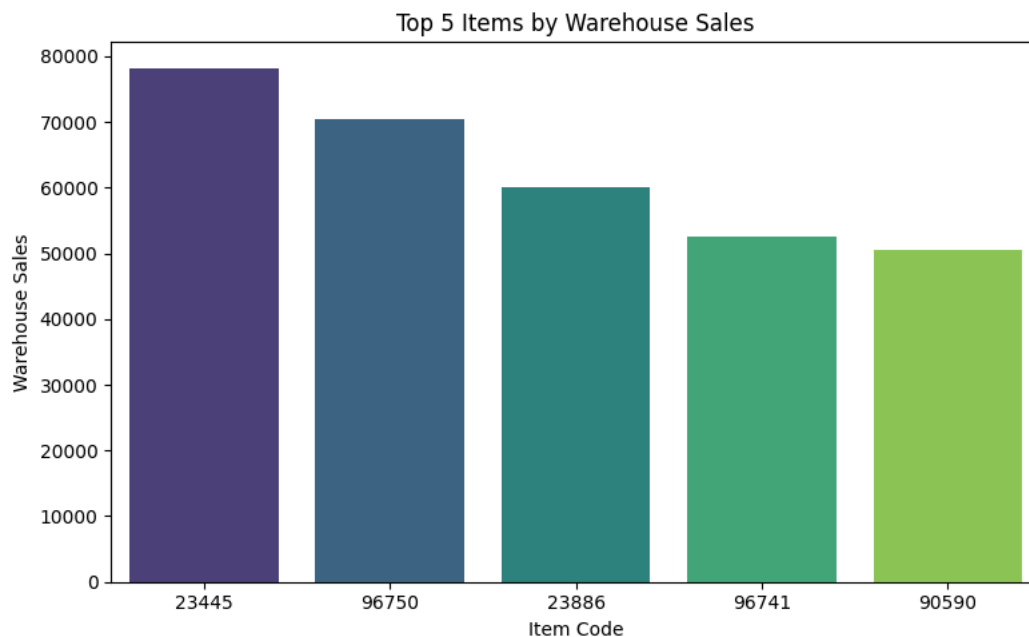

```
df3 = agg3.toPandas()

plt.figure(figsize=(8, 5))
sns.barplot(data=df3, x="item_code", y="total_warehouse_sales", palette="viridis")
plt.title("Top 5 Items by Warehouse Sales")
plt.xlabel("Item Code")
plt.ylabel("Warehouse Sales")
plt.tight_layout()
plt.show()
```

↪ <ipython-input-65-ae25b671a470>:4: FutureWarning:


Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend`

```
sns.barplot(data=df3, x="item_code", y="total_warehouse_sales", palette="viridis")
```



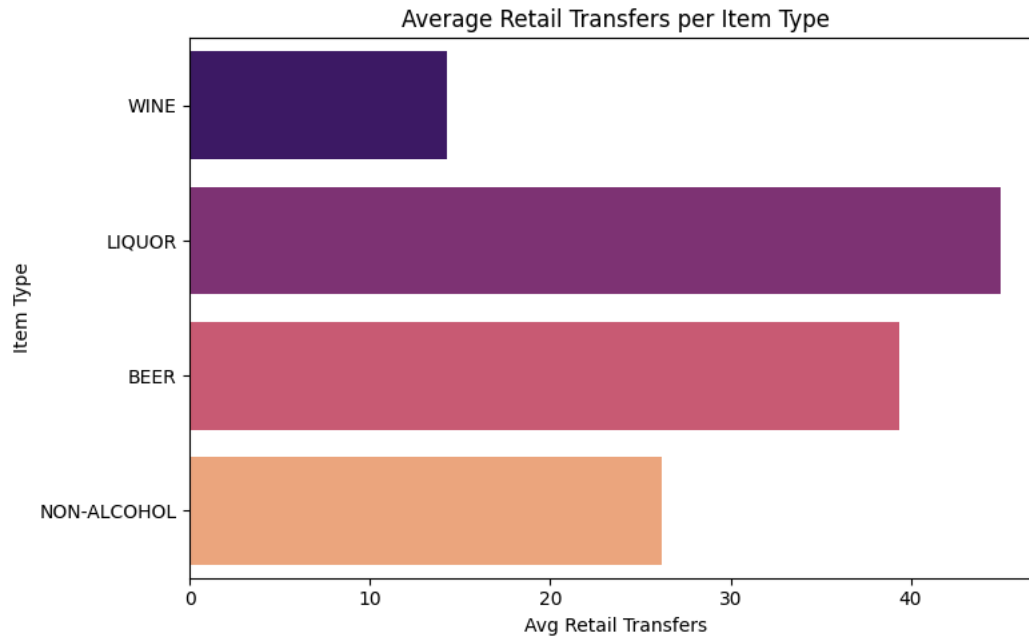
```
df4 = agg4.toPandas()

plt.figure(figsize=(8, 5))
sns.barplot(data=df4, x="avg_retail_transfers", y="item_type", palette="magma")
plt.title("Average Retail Transfers per Item Type")
plt.xlabel("Avg Retail Transfers")
plt.ylabel("Item Type")
plt.tight_layout()
plt.show()
```

 <ipython-input-66-4a4114700d58>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend`

```
sns.barplot(data=df4, x="avg_retail_transfers", y="item_type", palette="magma")
```

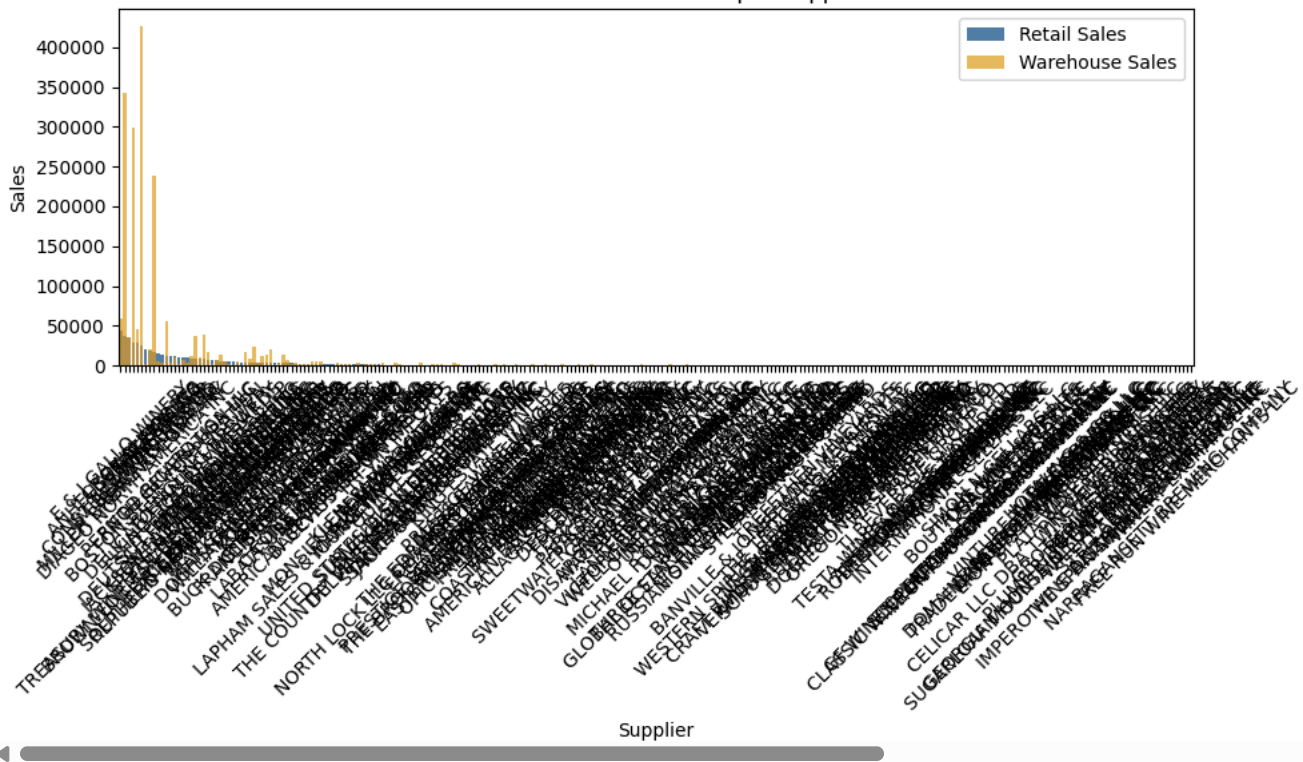


```
df5 = agg5.toPandas().sort_values("total_retail_sales", ascending=False)
```

```
plt.figure(figsize=(10, 6))
sns.barplot(data=df5, x="supplier", y="total_retail_sales", color="steelblue", label="Retail Sales")
sns.barplot(data=df5, x="supplier", y="total_warehouse_sales", color="orange", label="Warehouse Sales", alpha=0.7)
plt.title("Retail vs Warehouse Sales per Supplier")
plt.xlabel("Supplier")
plt.ylabel("Sales")
plt.legend()
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Retail vs Warehouse Sales per Supplier



```
df6 = agg6.toPandas()
```

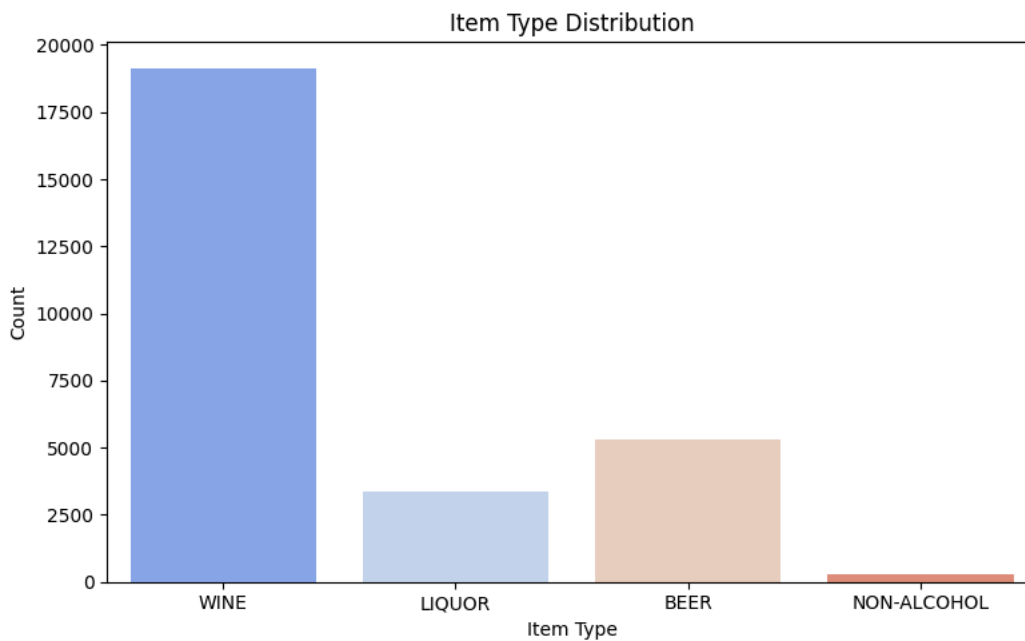
```
plt.figure(figsize=(8, 5))
sns.barplot(data=df6, x="item_type", y="item_type_count", palette="coolwarm")
plt.title("Item Type Distribution")
plt.xlabel("Item Type")
plt.ylabel("Count")
plt.tight_layout()
plt.show()
```



<ipython-input-68-51c88c641b10>:4: FutureWarning:

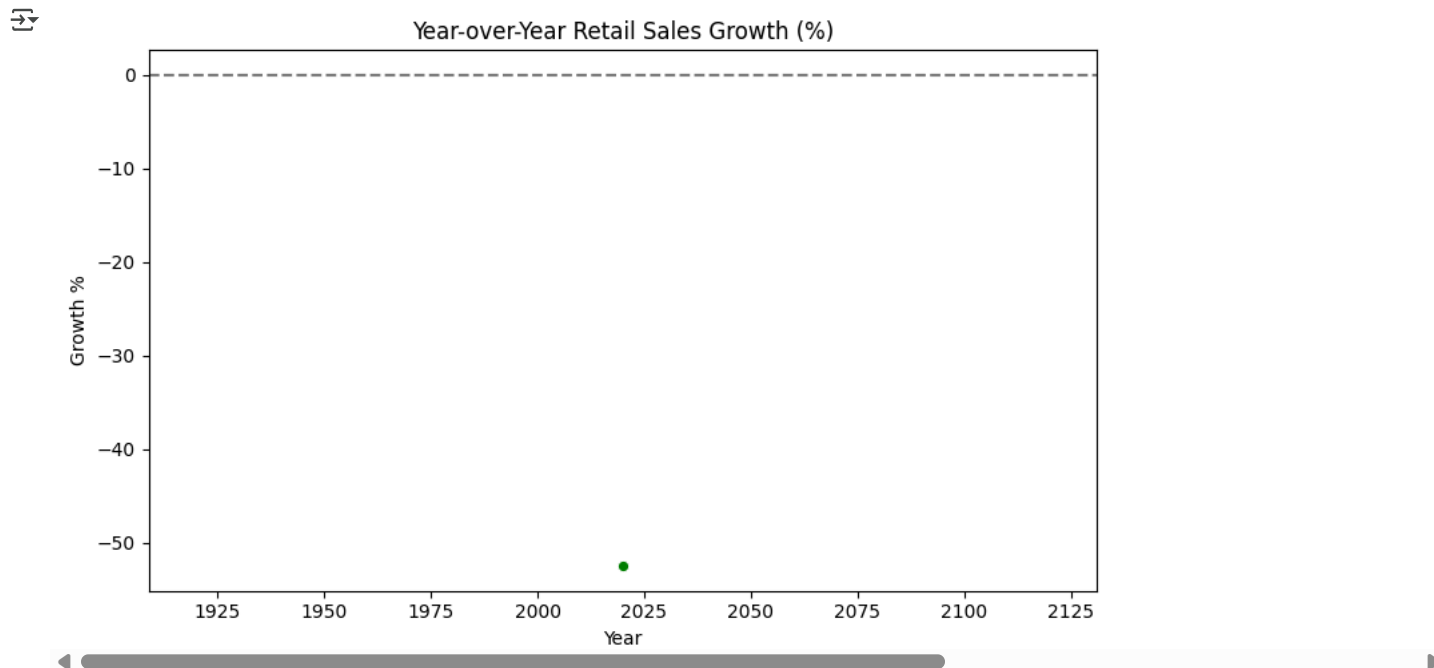
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend`

```
sns.barplot(data=df6, x="item_type", y="item_type_count", palette="coolwarm")
```



```
df7 = agg7.toPandas().sort_values("year")

plt.figure(figsize=(8, 5))
sns.lineplot(data=df7, x="year", y="growth_percentage", marker="o", color="green")
plt.title("Year-over-Year Retail Sales Growth (%)")
plt.xlabel("Year")
plt.ylabel("Growth %")
plt.axhline(0, color='gray', linestyle='--')
plt.tight_layout()
plt.show()
```



```
from pyspark.sql.functions import col
from pyspark.sql.types import DoubleType

# Select and cast required columns
ml_df = silver_df.select(
    col("retail_sales").cast(DoubleType()),
    col("warehouse_sales").cast(DoubleType()),
    col("retail_transfers").cast(DoubleType()),
    col("month").cast(DoubleType()),
    col("year").cast(DoubleType())
).dropna()

from pyspark.ml.feature import VectorAssembler

# Assemble features
assembler = VectorAssembler(
    inputCols=["warehouse_sales", "retail_transfers", "month", "year"],
    outputCol="features"
)

data = assembler.transform(ml_df).select("features", "retail_sales")

from pyspark.ml.regression import LinearRegression
```