Customer Data Managment

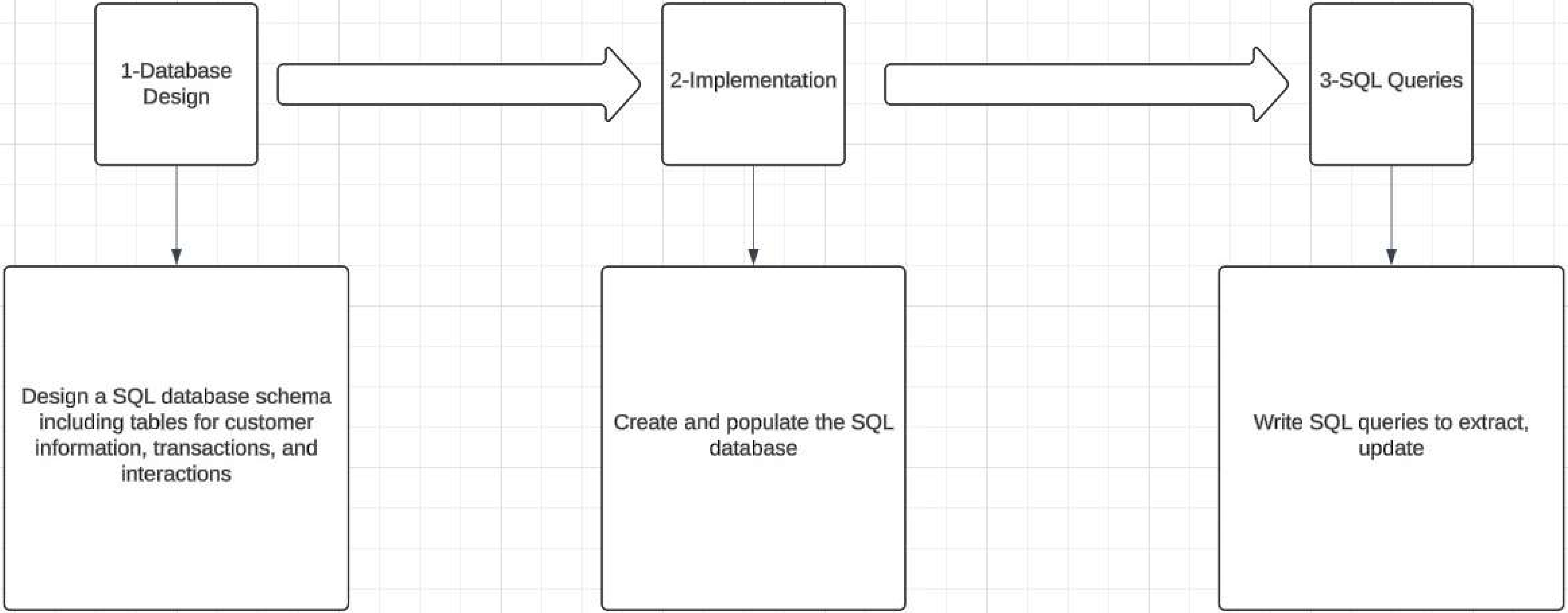
Team:

1. Mohamed Sameh (leader)
2. Rawan Amr
3. Esraa Osama
4. Mohamed Ibrahim
5. Shahd Waleed
6. Ali Sherif

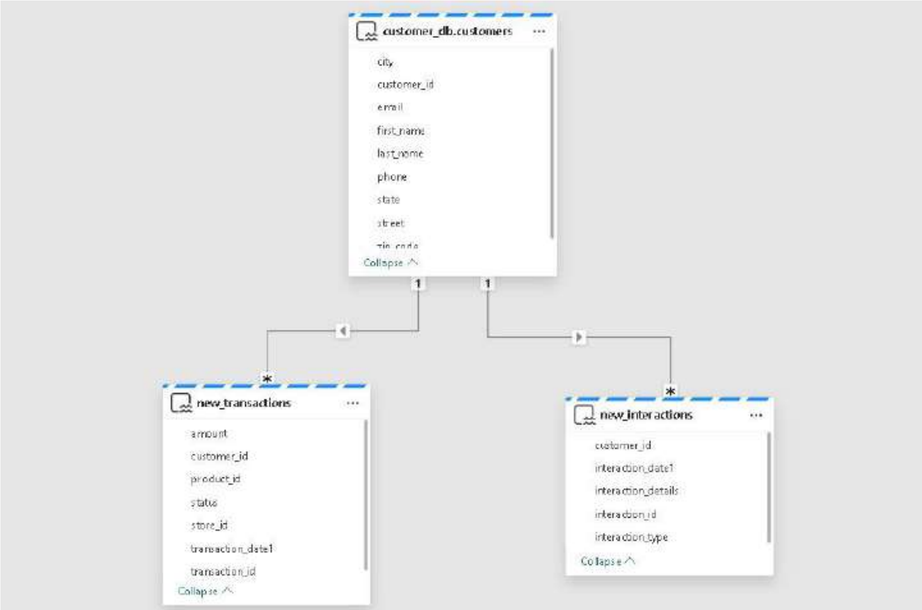
Contents of this template

Week 1	Data Managment & SQL Database Setup
Week 2	Data Warehousing & Python Programming
Week 3	Data Science & Model Building
Week 4	Using Predictive Model & Final presentation

Week 1: Data Management and SQL Database Setup



Here is the schemas of week 1



Contains the tables of the customers info like shown

Views for week 1

Views
- avg_time_between_transactions_interactions
- count_of_interaction
- count_of_transaction
- count_transaction_2023
- customers_no_transaction_multiple_interactions
- customers_transaction_no_interaction
- recent_transaction_interaction
- store_transaction_with_customer
- sum_of_amount

Views to extract the data of the customers

Views data preview

Data preview - count_of_interaction

Showing 1000 rows

Search

	ABC first_name	123 count_of_interaction
1	Marylyn	2
2	Ciera	1
3	Ling	1
4	Divina	2
5	Pinkie	2
6	Shirely	1
7	Jerome	3
8	Lissa	3
9	Klara	8
10	Vernetta	3
11	Takako	1
12	Antony	1
13	Joaquin	3
14	Adam	4
15	Ami	1
16	Domingo	3
17	Rosa	2
18	Syreeta	1
19	Melani	1
20	Kimberli	2
21	Zelma	2
22	Addie	1
23	Sheba	1
24	Mariette	1
25	Williemae	1

Succeeded (0 sec 938 ms)

Columns: 2 Rows: 1000

Views data preview

Data preview - count_transaction_2023 Showing 1000 rows Search

	full_name	count_transaction_2023
1	Michel Blankenship	1
2	Bernetta Summers	1
3	Ann Heath	1
4	Genny Hensley	2
5	Lean Stark	1
6	Ara Vazquez	1
7	Houston Vasquez	1
8	Jina Cooper	1
9	Theo Reese	1
10	Damian Mills	1
11	Rodolfo Buck	2
12	Ayanna Rhodes	1
13	Susann Bass	2
14	Rolanda Larsen	1
15	Aleta Stone	1
16	Julius Holt	1
17	Burma Summers	1
18	Rosa Kinney	1
19	Sarah Kirkland	1
20	Jenna Saunders	1
21	Jovita Bishop	2
22	Kristofer Craig	2
23	Tomika Wilder	2
24	Arlena Buckner	3
25	Delfina Gilliam	1

Succeeded (1 sec 92 ms) Columns: 2 Rows: 515

Views data preview

Data preview - sum_of_amount

Showing 1000 rows

Search

	full_name	sum_of_amount
1	Michel Blankenship	4672.46
2	Bernetta Summers	5967.2
3	Ara Vazquez	1983.56
4	Houston Vasquez	3156.59
5	Jina Cooper	1694.73
6	Theo Reese	6341.89
7	Renay Atkins	1026.25
8	Quyen Houston	1079.26
9	Miquel Neal	1112.99
10	Burma Summers	4048.74
11	Jovita Bishop	9961.95
12	Kristofer Craig	5843.12
13	Tomika Wilder	3266.74
14	Martha Burgess	1603.66
15	Carson Macias	10584.03
16	Evelina Manning	2463.45
17	Shawnda Glover	9369.6
18	Lynn Mcmahon	6390.22
19	Hope Cotton	1968.94
20	Basil Ballard	2362.79
21	Ann Heath	4154.78
22	Genny Hensley	3854.02
23	Rodolfo Buck	3403.58
24	Aleta Stone	4072.68
25	Julius Holt	454.15

Succeeded (9 sec 755 ms)

Columns: 2 Rows: 1000

Stored procedures

- Stored Procedures
  - UpdateTransactionByCustomerName
  - UPDATE_transaction_status
  - UPDATE_interactio_type
  - UpdateCustomerInfo

Procedures to update customer data

Example of using stored procedures

```sql
SELECT * FROM[customer data management_1].[customer_db].[customers] WHERE customer_id = 22;
```

Messages Results Save as table Open in Excel Explore this data (preview) Copy Search

	customer_id	first_name	last_name	phone	email	street	city	state	zip_code
1	22	Adelle	Larsen	NULL	adelle.larsen@gm...	683 West Kirkland...	East Northport	NY	11731

Before updating

Example of using stored procedures

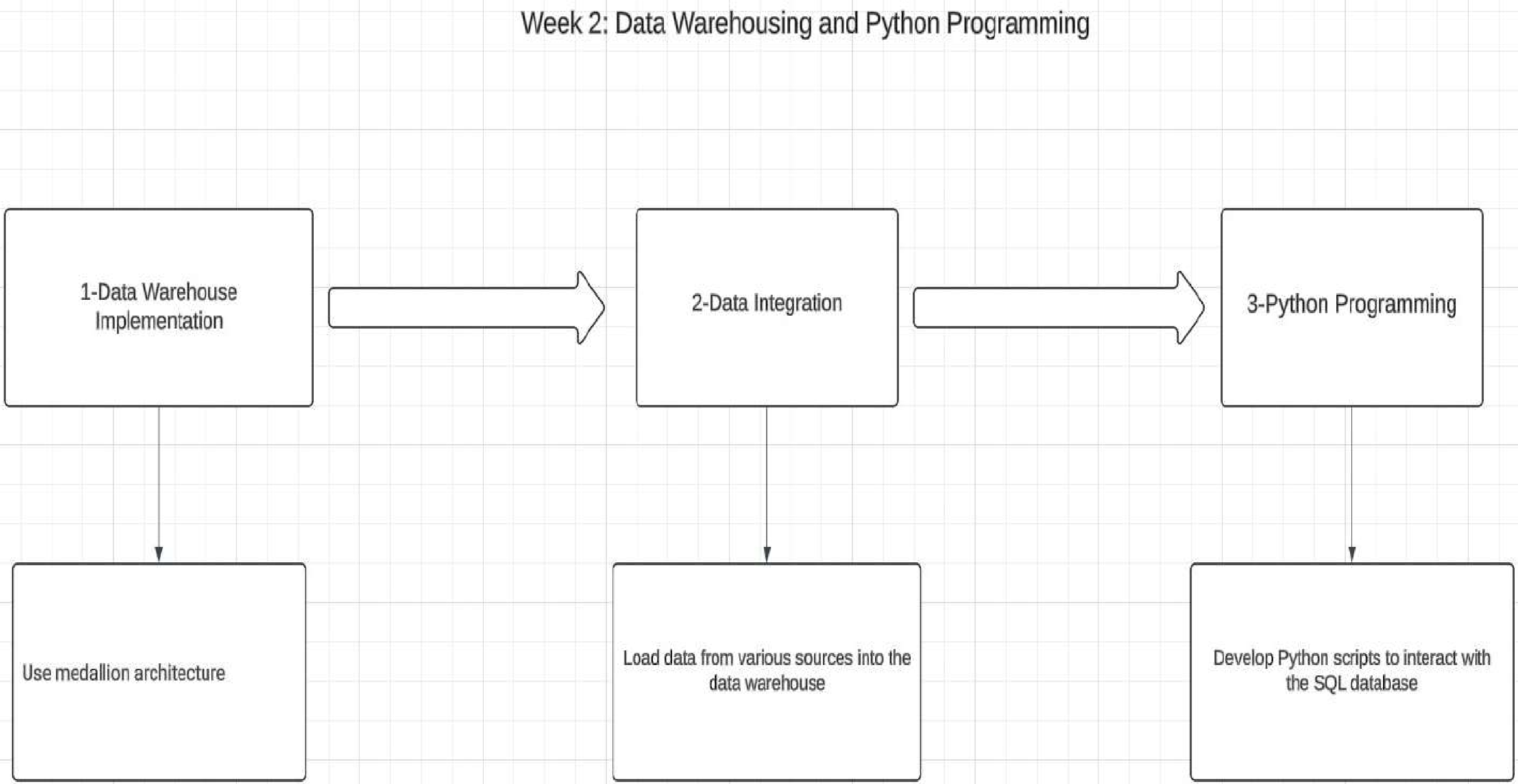
```
EXEC [customer data management_1].[customer_db].[UpdateCustomerInfo] @CustomerID = 22, @Email = 'john.new@example.com'
```

After applying the stored procedure

```
5   SELECT * FROM[customer data management_1].[customer_db].[customers] WHERE customer_id = 22;
```

Messages | Results | Save as table | Open in Excel | Explore this data (preview) | Copy | Search

	customer_id	first_name	last_name	phone	email	street	city	state	zip_code
1	22	Adelle	Larsen	NULL	john.new@exampl...	683 West Kirkland...	East Northport	NY	11731


Week 2: Data Warehousing and Python Programming
1-Data Warehouse Implementation
2-Data Integration
3-Python Programming
Use medallion architecture
Load data from various sources into the data warehouse
Develop Python scripts to interact with the SQL database

Firstly

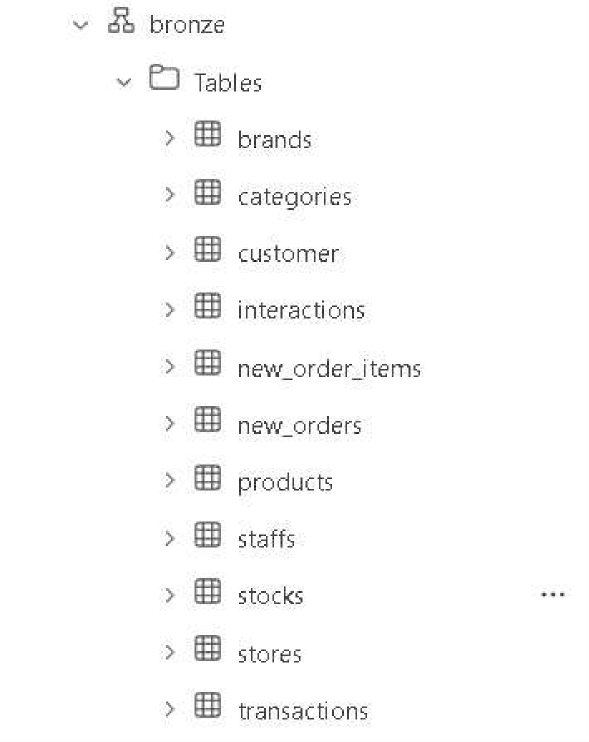
Data Warehousing

Medallion Architecture

Bronze pipline

From the source to the bronze

Bronze Schema



From the source to the bronze

Silver Schema

- Tables
  - customers
  - dim_order_items
  - dim_orders
  - dim_stores
  - dim_transactions
  - interactions
  - products

Stored procedures to transfer data to the Silver Schema

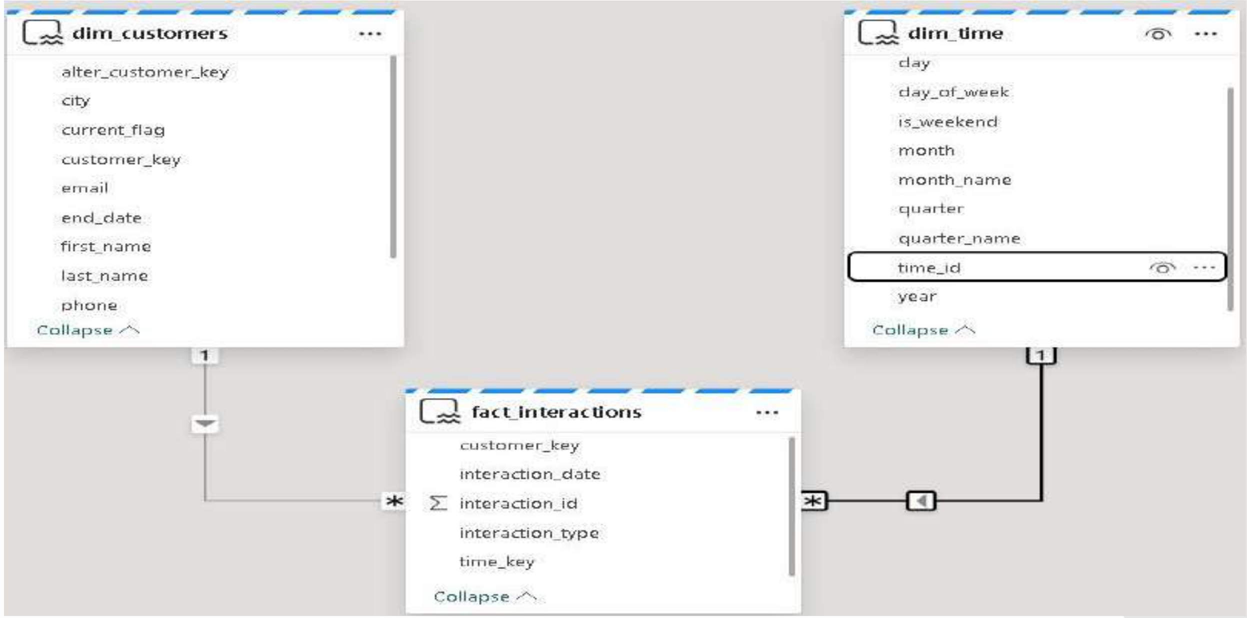
Stored Procedures

- UPDATE_customer
- INSERT_customer
- UPDATE transactions
- INSERT transactions
- UPDATE interactions
- INSERT interactions
- UPDATE silver schema
- UPDATE product silver
- UPDATE dim_orders silver
- copy transactions from bro...
- UPDATE_stores
- UPDATE order_item silver

Gold Schema

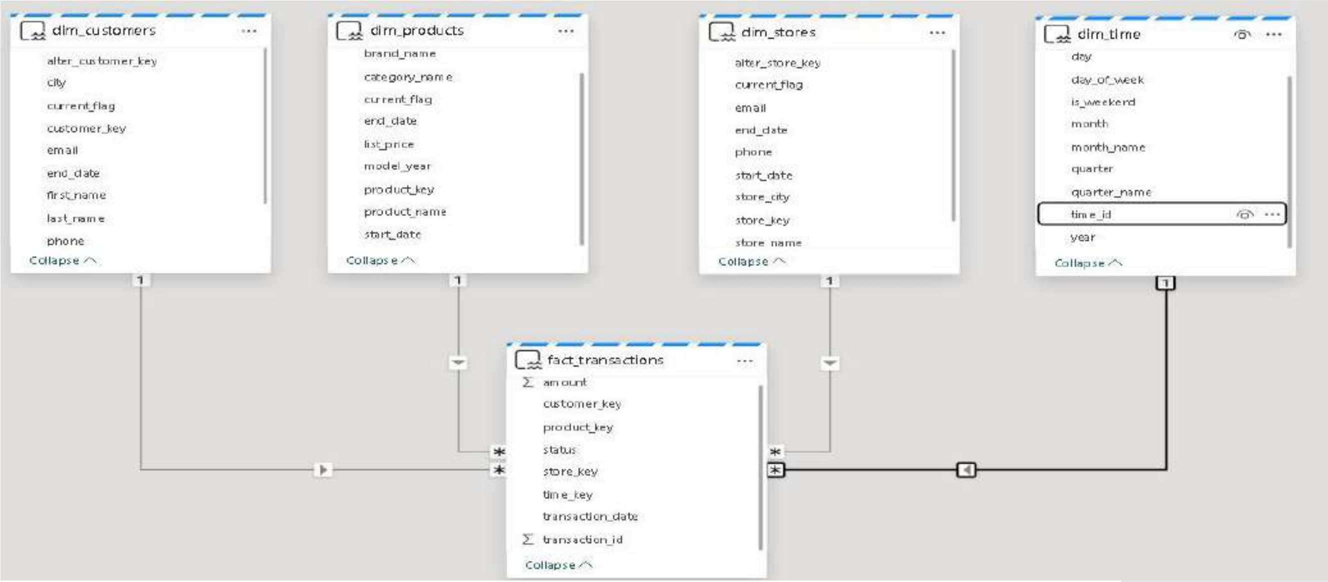
- gold
  - Tables
    - dim_customers
    - dim_order_items
    - dim_products
    - dim_stores
    - dim_time
    - fact_interactions
    - fact_orders
    - fact_transactions

Gold Schema



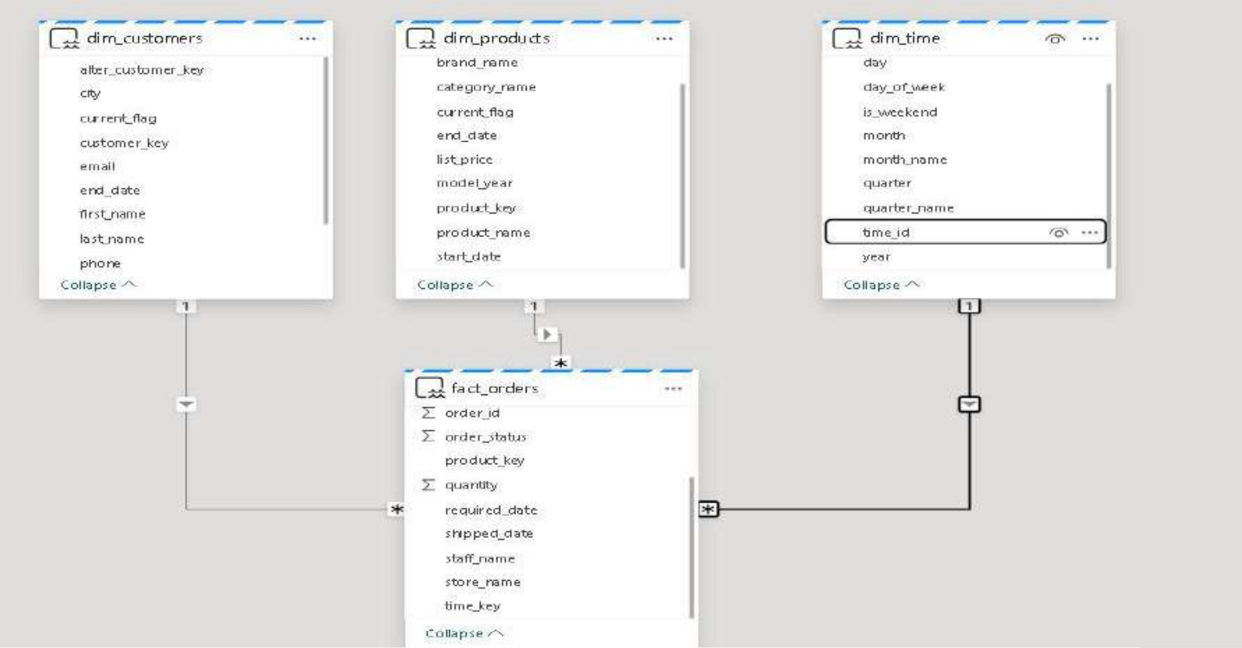
dim_customers
alter_customer_key
city
current_flag
customer_key
email
end_date
first_name
last_name
phone
Collapse
dim_time
day
day_of_week
is_weekend
month
month_name
quarter
quarter_name
time_id
year
Collapse
1
1
fact_interactions
customer_key
interaction_date
interaction_id
interaction_type
time_key
Collapse
*
*

Gold Schema



dim_customers
alter_customer_key
city
current_flag
customer_key
email
end_date
first_name
last_name
phone
Collapse
dim_products
brand_name
category_name
current_flag
end_date
list_price
model_year
product_key
product_name
start_date
Collapse
dim_stores
alter_store_key
current_flag
email
end_date
phone
start_date
store_city
store_key
store_name
Collapse
dim_time
day
day_of_week
is_weekend
month
month_name
quarter
quarter_name
time_id
year
Collapse
fact_transactions
amount
customer_key
product_key
status
store_key
time_key
transaction_date
transaction_id
Collapse
1
1
1
1
*
*
*
*

Gold Schema

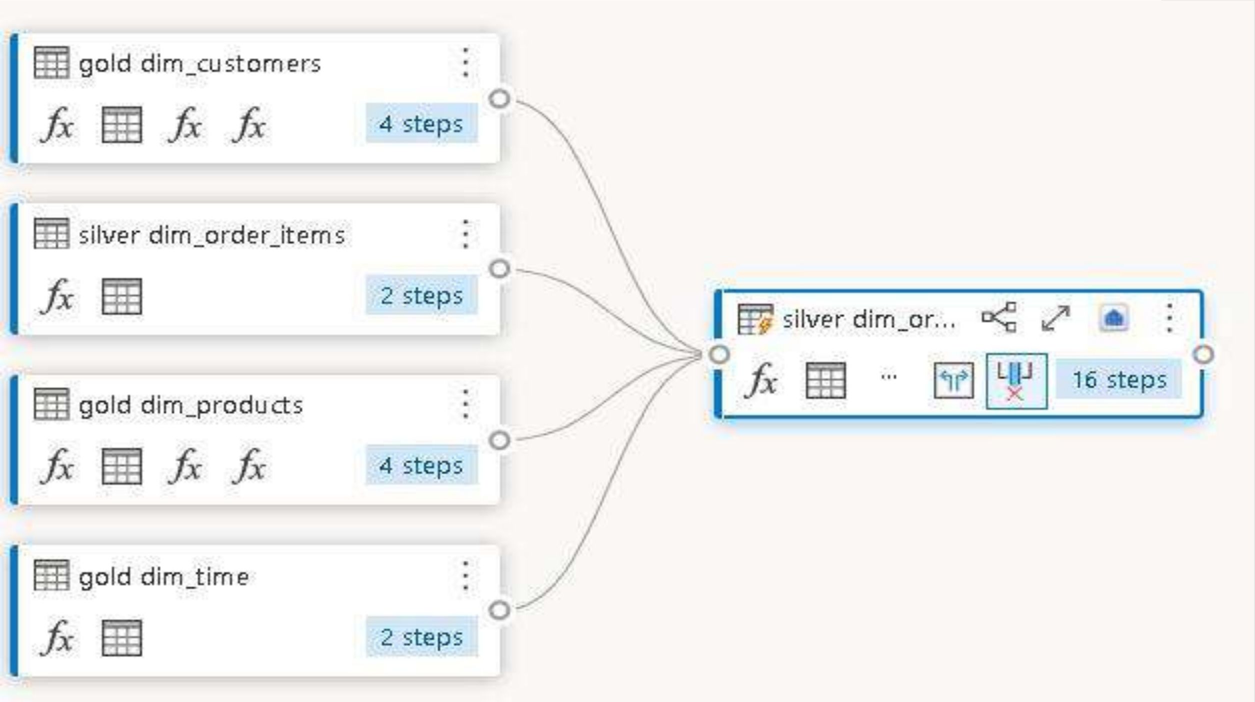


dim_customers
alter_customer_key
city
current_flag
customer_key
email
end_date
first_name
last_name
phone
Collapse
dim_products
brand_name
category_name
current_flag
end_date
list_price
model_year
product_key
product_name
start_date
Collapse
dim_time
day
day_of_week
is_weekend
month
month_name
quarter
quarter_name
time_id
year
Collapse
1
1
1
*
*
*
fact_orders
order_id
order_status
product_key
quantity
required_date
shipped_date
staff_name
store_name
time_key
Collapse

Stored procedures to update gold schema

- Stored Procedures
  - UPDATE customer_dim
  - UPDATE gold schema
  - UPDATE dim_stores
  - UPDATE dim_products
  - UPDATE dim_order_items

Fact table by dataflow gen 2



Data preview

Data preview - dim_stores

Showing 1000 rows

	store_key	alter_store_key	store_name	phone	email	store_city	store_state	start_date	end_date	current_flag
1	4	2	Baldwin Bikes	99839	baldwin@bikes.shop	Baldwin	NY	2024-10-07	NULL	1
2	1	1	Santa Cruz Bikes	(831) 476-4321		Santa Cruz	CA	2024-09-30	NULL	1
3	3	3	Rowlett Bikes	(972) 530-5555	rowlett@bikes.shop	Rowlett	TX	2024-09-30	NULL	1
4	2	2	Baldwin Bikes	(516) 379-8888	baldwin@bikes.shop	Baldwin	NY	2024-09-30	2024-10-07	0

secondly

Python programming

Python

```
# Customer interaction report
customer_interaction_df = spark.sql("""
    SELECT
        dc.customer_key,
        dc.first_name,
        dc.last_name,
        fi.interaction_type,
        COUNT(fi.interaction_id) AS total_interactions,
        MIN(fi.interaction_date) AS first_interaction_date,
        MAX(fi.interaction_date) AS last_interaction_date
    FROM fact_interactions fi
    JOIN dim_customers dc ON fi.customer_key = dc.customer_key
    GROUP BY dc.customer_key, dc.first_name, dc.last_name, fi.interaction_type
""")

# Displaying the customer interaction report
customer_interaction_df.show()
```

- Command executed in 3 sec 560 ms by Suez Gahar on 2:59:43 AM, 10/07/24

customer_key	first_name	last_name	interaction_type	total_interactions	first_interaction_date	last_interaction_date
673	Adam	Henderson	Complaint	1	2024-08-09 00:00:00	2024-08-09 00:00:00
1045	Pasquale	Hogan	Complaint	1	2024-07-01 00:00:00	2024-07-01 00:00:00
1361	Destiny	Goodman	Complaint	1	2024-02-04 00:00:00	2024-02-04 00:00:00
651	Laure	Pena	Complaint	1	2024-01-16 00:00:00	2024-01-16 00:00:00
1268	Sung	Chambers	Review	1	2023-09-23 00:00:00	2023-09-23 00:00:00
504	Tiana	Henderson	Review	1	2024-09-14 00:00:00	2024-09-14 00:00:00
855	Lean	Stark	Inquiry	1	2023-09-30 00:00:00	2023-09-30 00:00:00
253	Maurice	Norton	Inquiry	1	2024-07-07 00:00:00	2024-07-07 00:00:00
1299	Shauna	Edwards	Inquiry	2	2024-01-13 00:00:00	2024-05-24 00:00:00
124	Jeni	Booker	Inquiry	1	2024-09-02 00:00:00	2024-09-02 00:00:00
887	Graig	Cannon	Inquiry	1	2024-01-27 00:00:00	2024-01-27 00:00:00
794	Donette	Mccarthy	Complaint	1	2024-06-24 00:00:00	2024-06-24 00:00:00

Python

```python
# Customer transaction report
customer_transaction_df = spark.sql("""
    SELECT
        dc.customer_key,
        dc.first_name,
        dc.last_name,
        COUNT(ft.transaction_id) AS total_transactions,
        SUM(ft.amount) AS total_amount_spent,
        AVG(ft.amount) AS avg_transaction_value,
        MIN(ft.transaction_date) AS first_transaction_date,
        MAX(ft.transaction_date) AS last_transaction_date
    FROM fact_transactions ft
    JOIN dim_customers dc ON ft.customer_key = dc.customer_key
    GROUP BY dc.customer_key, dc.first_name, dc.last_name
""")

# Displaying the customer transaction report
customer_transaction_df.show()
```

[3] ✓ - Command executed in 3 sec 498 ms by Suez Gahar on 2:59:47 AM, 10/07/24

customer_key	first_name	last_name	total_transactions	total_amount_spent	avg_transaction_value	first_transaction_date	last_transaction_date
1393	Vivian	Deleon	1	1678.46	1678.46	2024-05-21 00:00:00	2024-05-21 00:00:00
32	Araceli	Golden	2	2063.76	1031.88	2024-02-06 00:00:00	2024-02-06 00:00:00
1321	Shantel	Gregory	3	3964.63	1321.5433333333333	2024-02-18 00:00:00	2024-07-19 00:00:00
334	Somer	Jordan	2	2860.83	1430.415	2024-08-10 00:00:00	2024-09-05 00:00:00
857	Inga	Koch	2	1965.3899999999999	982.6949999999999	2024-04-01 00:00:00	2024-07-12 00:00:00
53	Saturnina	Garner	3	8591.380000000001	2863.7933333333335	2023-09-26 00:00:00	2024-06-27 00:00:00
34	Brittney	Woodward	2	3196.7400000000002	1598.3700000000001	2023-10-19 00:00:00	2023-11-07 00:00:00
146	Stefany	Potter	3	1768.55	589.5166666666667	2024-01-26 00:00:00	2024-05-04 00:00:00
990	Casimira	Chapman	4	8302.87	2075.7175	2024-03-01 00:00:00	2024-06-28 00:00:00
15	Linnie	Branch	3	5529.76	1843.2533333333333	2023-10-16 00:00:00	2024-04-29 00:00:00
385	Rochelle	Ward	2	4946.93	2473.465	2023-12-29 00:00:00	2024-04-09 00:00:00
1268	Sung	Chambers	3	2525.14	841.7133333333333	2023-11-01 00:00:00	2023-12-24 00:00:00

Python

```python
# Top selling products report
top_selling_products_df = spark.sql("""
    SELECT
        dp.product_key,
        dp.product_name,
        COUNT(ft.transaction_id) AS total_sales,
        SUM(ft.amount) AS total_revenue,
        AVG(ft.amount) AS avg_sale_price
    FROM fact_transactions ft
    JOIN dim_products dp ON ft.product_key = dp.product_key
    GROUP BY dp.product_key, dp.product_name
    ORDER BY total_sales DESC
    LIMIT 10
""")

# Displaying the top selling products report
top_selling_products_df.show()
```

[4] ✓ - Command executed in 1 sec 503 ms by Suez Gahar on 2:59:49 AM, 10/07/24

product_key	product_name	total_sales	total_revenue	avg_sale_price
100	Haro Shredder Pro...	53	63996.240000000005	1207.4762264150945
99	Haro Shredder Pro...	53	63996.240000000005	1207.4762264150945
197	Trek Lift+ Lowste...	31	64769.61000000002	2089.342258064517
196	Trek Lift+ Lowste...	31	64769.61000000002	2089.342258064517
328	Electra Townie Or...	30	45086.29000000001	1502.8763333333336
302	Electra Sweet Rid...	28	34393.50000000001	1228.339285714286
303	Electra Sweet Rid...	28	34393.50000000001	1228.339285714286
41	Haro Flightline T...	28	47656.020000000004	1702.0007142857144
46	Trek Fuel EX 9.8 ...	24	17392.41	724.68375
301	Electra Sweet Rid...	22	35061.87000000002	1593.7213636363645

Python

```
# Top customers by sales
top_customers_df = spark.sql("""
    SELECT
        dc.customer_key,
        dc.first_name,
        dc.last_name,
        SUM(ft.amount) AS total_spent
    FROM fact_transactions ft
    JOIN dim_customers dc ON ft.customer_key = dc.customer_key
    GROUP BY dc.customer_key, dc.first_name, dc.last_name
    ORDER BY total_spent DESC
    LIMIT 10
""")

# Displaying the top customers by sales
top_customers_df.show()
```

[6] - Command executed in 1 sec 483 ms by Suez Gahar on 2:59:53 AM, 10/07/24

```
+------------+----------+---------+------------------+
|customer_key|first_name|last_name|       total_spent|
+------------+----------+---------+------------------+
|         875|  Tanesha|   Sawyer|14009.699999999999|
|        1283|     Mica|    Barry|          13533.36|
|         399|     Bart|     Hess|12508.850000000002|
|         286|   Virgil|    Frost|          12160.31|
|         366|  Lavonne| Anderson|11928.689999999999|
|         267| Angelina|    Lloyd|11881.369999999999|
|         206|Stephaine|  Riddle|11723.289999999999|
|        1183|      Kim|    Clark|11428.079999999998|
|         376|   Elanor|  Patrick|11383.580000000002|
|        1190|    Romeo|   Steele|          11249.46|
+------------+----------+---------+------------------+
```

Python

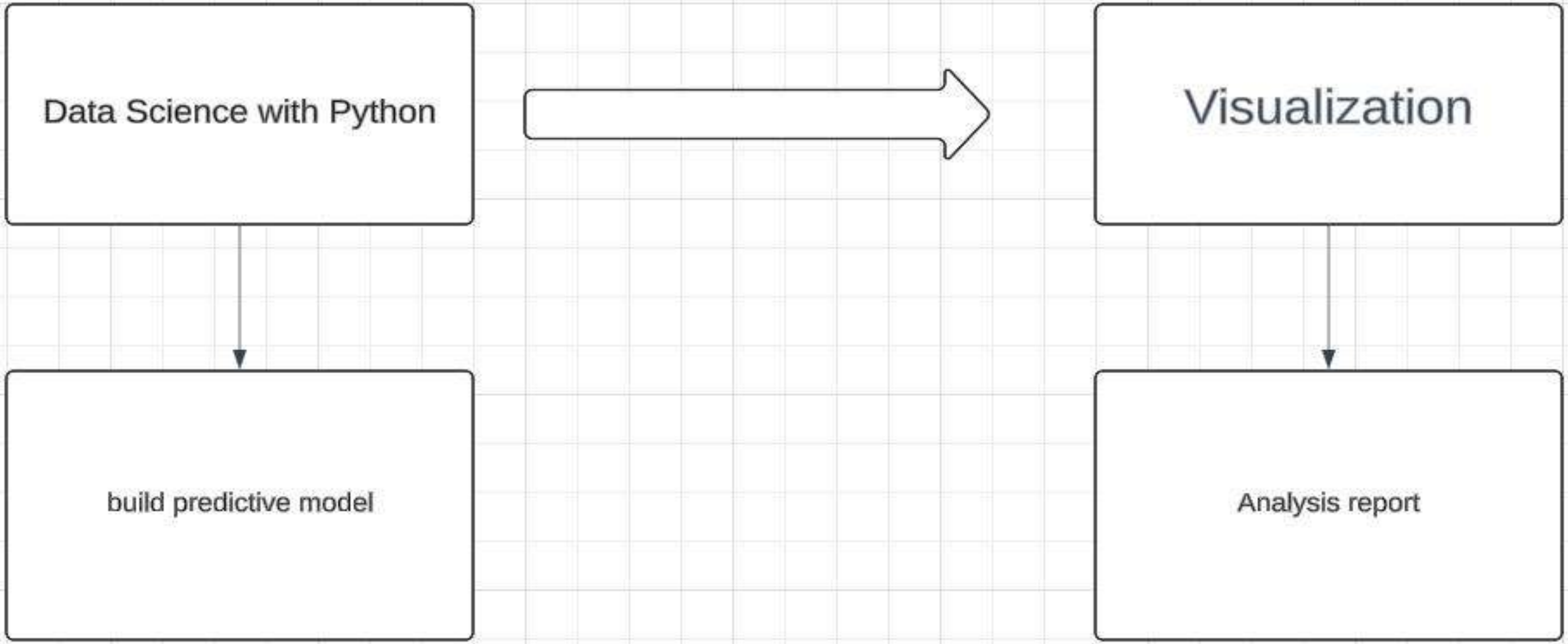
```python
# Top selling products report
top_selling_products_df = spark.sql("""
    SELECT
        dp.product_key,
        dp.product_name,
        COUNT(ft.transaction_id) AS total_sales,
        SUM(ft.amount) AS total_revenue,
        AVG(ft.amount) AS avg_sale_price
    FROM fact_transactions ft
    JOIN dim_products dp ON ft.product_key = dp.product_key
    GROUP BY dp.product_key, dp.product_name
    ORDER BY total_sales DESC
    LIMIT 10
""")

# Displaying the top selling products report
top_selling_products_df.show()
```

[4] ✓ - Command executed in 1 sec 503 ms by Suez Gahar on 2:59:49 AM, 10/07/24

product_key	product_name	total_sales	total_revenue	avg_sale_price
100	Haro Shredder Pro...	53	63996.240000000005	1207.4762264150945
99	Haro Shredder Pro...	53	63996.240000000005	1207.4762264150945
197	Trek Lift+ Lowste...	31	64769.61000000002	2089.34225806451 7
196	Trek Lift+ Lowste...	31	64769.61000000002	2089.34225806451 7
328	Electra Townie Or...	30	45086.29000000001	1502.8763333333336
302	Electra Sweet Rid...	28	34393.50000000001	1228.339285714286
303	Electra Sweet Rid...	28	34393.50000000001	1228.339285714286
41	Haro Flightline T...	28	47656.020000000004	1702.0007142857144
46	Trek Fuel EX 9.8 ...	24	17392.41	724.68375
301	Electra Sweet Rid...	22	35061.87000000002	1593.721363636364 5

Week 3: Data Science

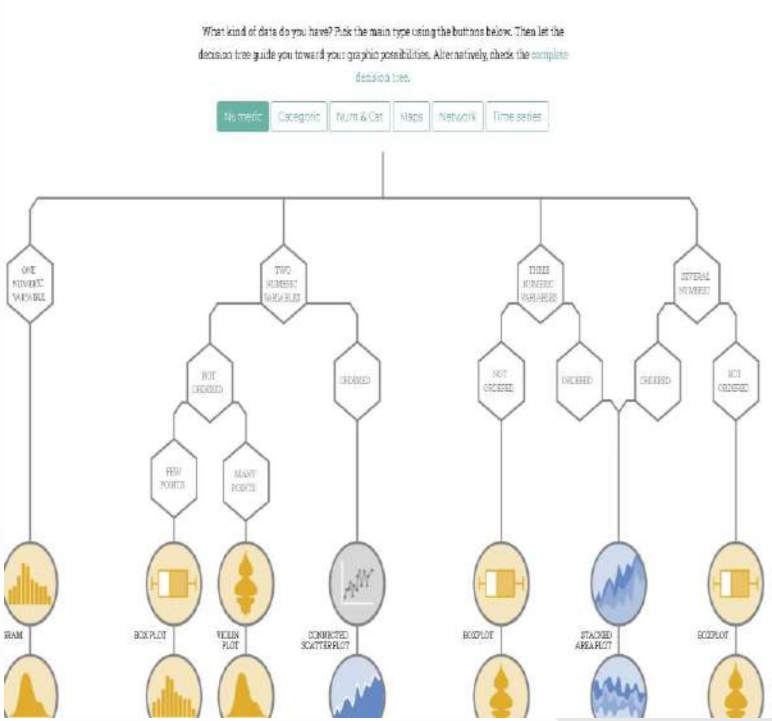


Firstly

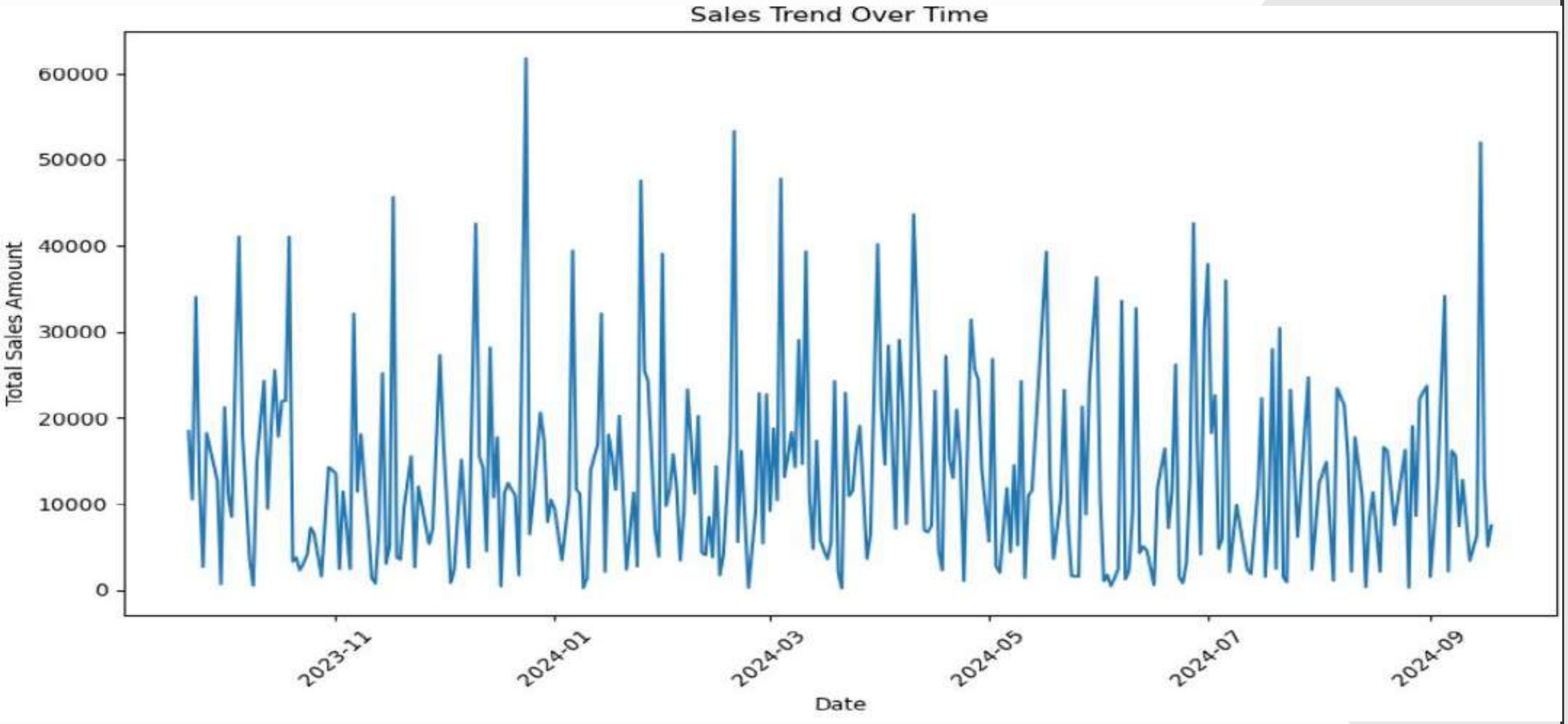
Data science

methods

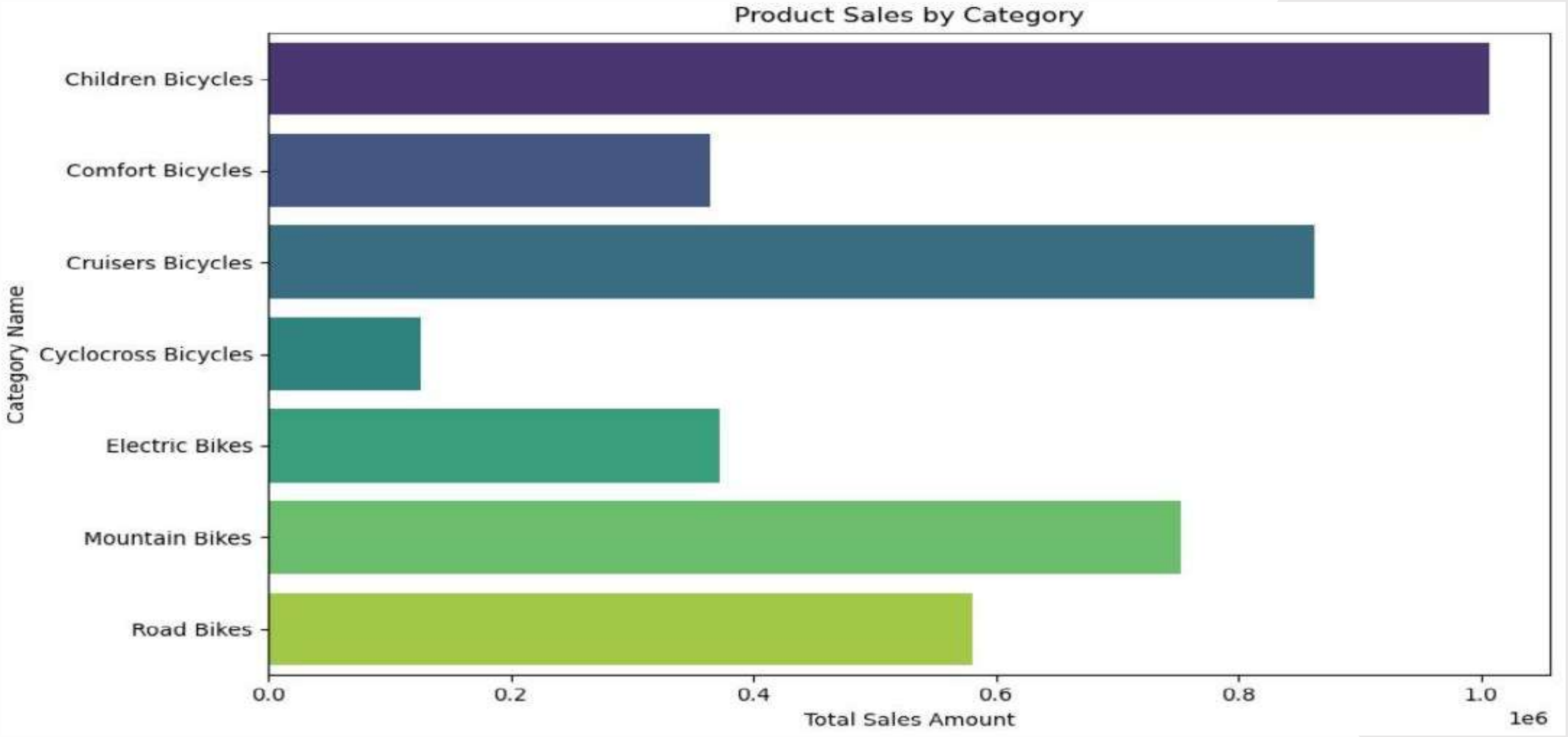

from Data to Viz
From Data to Viz leads you to the most appropriate graph for your data. It links to the code to build it and lists common caveats you should avoid.
EXPLORE
SEE POSTER


What kind of data do you have? Pick the main type using the buttons below. Then let the decision tree guide you toward your graphic possibilities. Alternatively, check the complete decision tree.
Numeric
Categoric
Num & Cat
Maps
Network
Time series

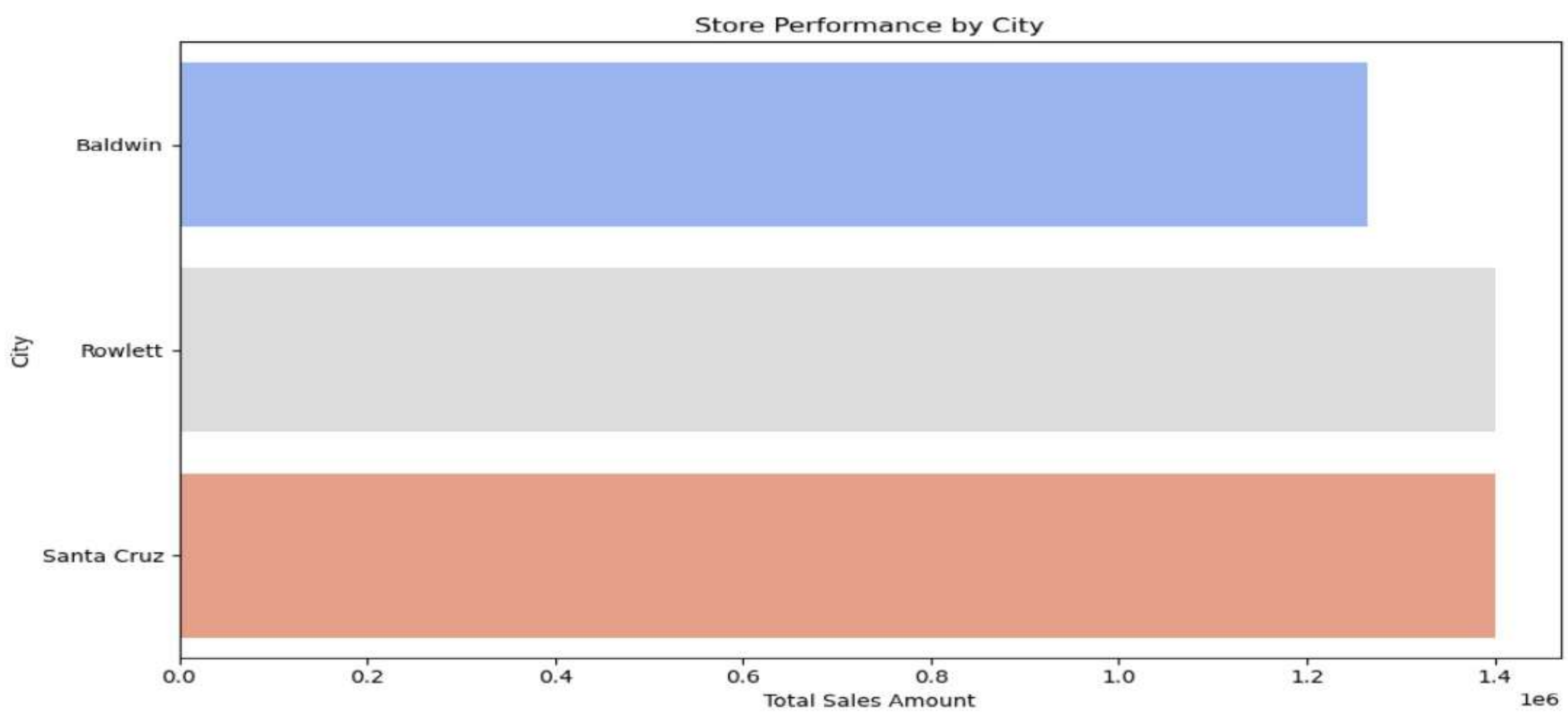
Examples for visualization



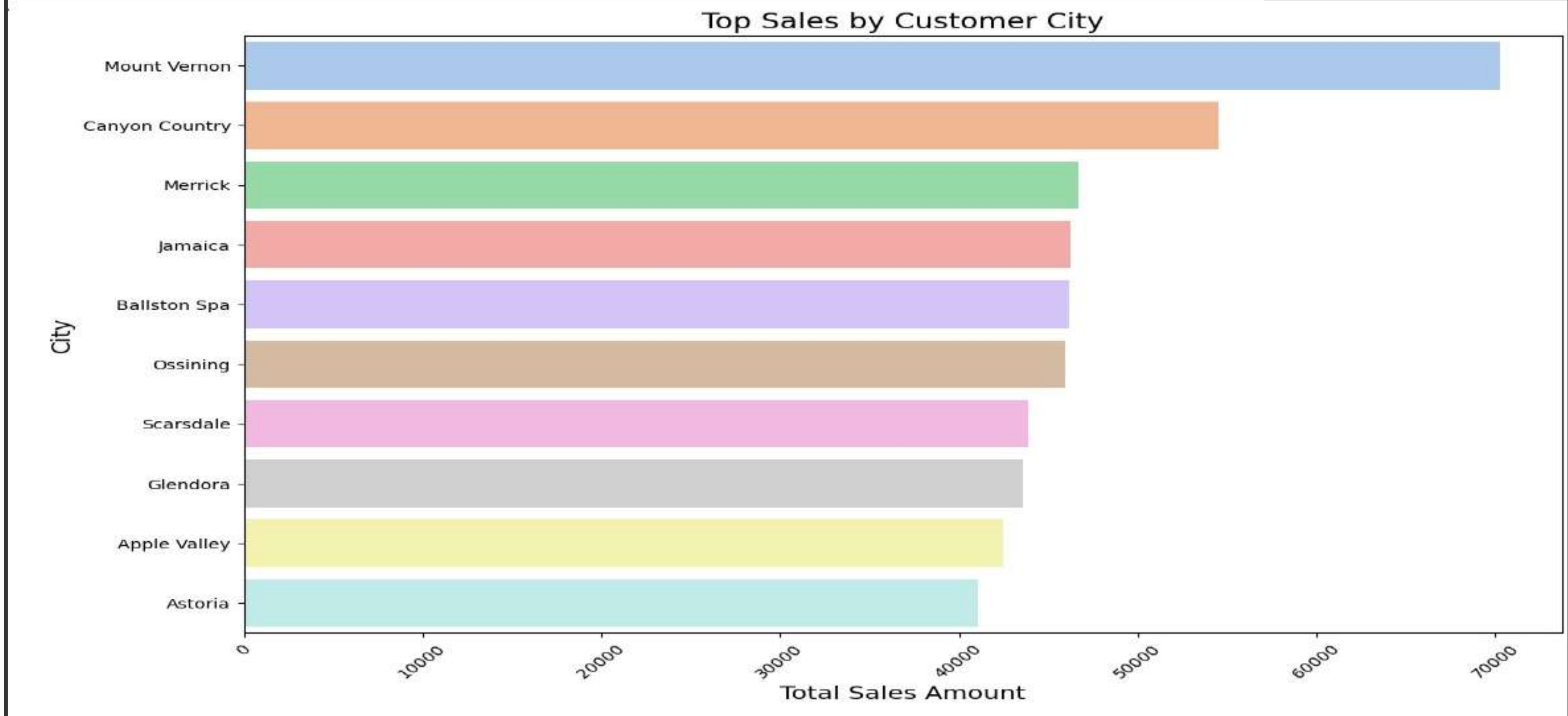
Examples for visualization



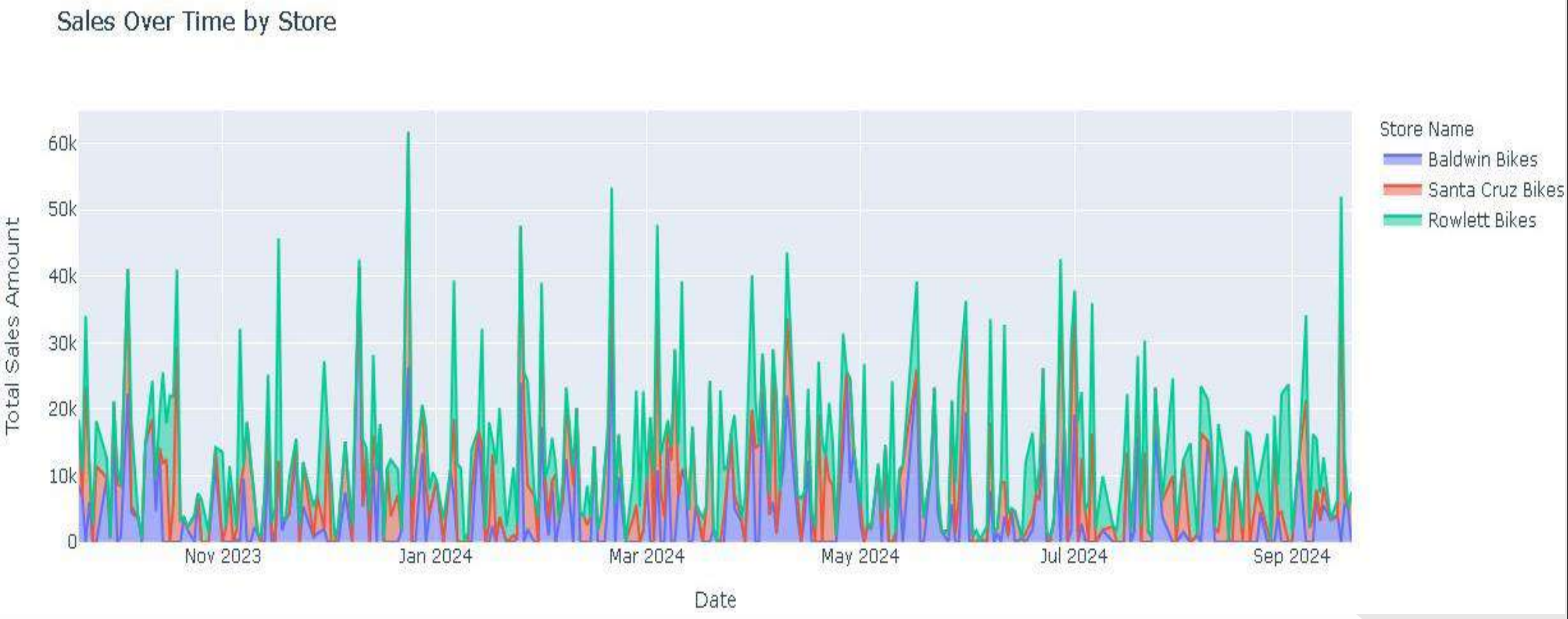
Examples for visualization



Examples for visualization



Examples for visualization



secondly

Model building

Important libraries & loading the data (table reading)

```python
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.evaluation import BinaryClassificationEvaluator
from pyspark.ml.tuning import ParamGridBuilder, CrossValidator
from pyspark.sql import functions as F
from pyspark.ml import Pipeline

# قراءة الجداول
df_customers = spark.read.table('dim_customers')
df_products = spark.read.table('dim_products')
df_time = spark.read.table('dim_time')
df_stores = spark.read.table('dim_stores')
df_interactions = spark.read.table('fact_interactions')
df_transactions = spark.read.table('fact_transactions')
df_orders = spark.read.table('fact_orders')  # تم تعديل هذا الجزء
```

Merging specific data

```
# دمج البيانات اللازمة
df_churn = df_customers \
    .join(df_interactions, 'customer_key', 'left') \
    .join(df_transactions, 'customer_key', 'left') \
    .join(df_orders, 'customer_key', 'left')  # تم تعديل هذا الجزء
```

Creating new features

```
# إنشاء ميزات جديدة
df_churn = df_churn \
    .groupBy('customer_key') \
    .agg(
        F.count('order_id').alias('total_orders'),
        F.count('interaction_id').alias('total_interactions'),
        F.sum('amount').alias('total_spent'),
        F.max('interaction_date').alias('last_interaction_date'),
        F.min('interaction_date').alias('first_interaction_date')
    )
```

Calculating relation time

```
# حساب مدة العلاقة
df_churn = df_churn \
    .withColumn('customer_lifetime', F.datediff(F.current_date(), 'first_interaction_date')) \
    .withColumn('churned', F.when(F.datediff(F.current_date(), 'last_interaction_date') > 30, 1).otherwise(0))
```

Removing empty values

```
# إزالة القيم الفارغة
feature_columns = ['total_orders', 'total_interactions', 'total_spent', 'customer_lifetime']
df_churn_cleaned = df_churn.na.drop(subset=feature_columns)
```

Splitting the data into train and test

```
# تقسيم البيانات
train_data, test_data = df_churn_cleaned.randomSplit([0.8, 0.2], seed=1234)
```

Train : 80%
Test : 20%

Creating Vector assembler , Randomforest , Grid search , Cross-Validation

```python
# إنشاء الـ VectorAssembler للميزات
assembler = VectorAssembler(inputCols=feature_columns, outputCol='features', handleInvalid='skip')
```

```python
# إنشاء RandomForestClassifier
rf = RandomForestClassifier(labelCol='churned', featuresCol='features')
```

```python
# إعداد شبكة المعلمات لـ Grid Search
paramGrid = (ParamGridBuilder()
             .addGrid(rf.numTrees, [50, 100, 150])
             .addGrid(rf.maxDepth, [5, 10, 15])
             .addGrid(rf.maxBins, [32, 64])
             .build())
```

```python
# إعداد Cross-Validation مع 5 طيات
evaluator = BinaryClassificationEvaluator(labelCol="churned", rawPredictionCol="rawPrediction", metricName="areaUnderROC")
crossval = CrossValidator(estimator=rf,
                          estimatorParamMaps=paramGrid,
                          evaluator=evaluator,
                          numFolds=5)
```

```python
# إنشاء Pipeline
pipeline = Pipeline(stages=[assembler, crossval])
```

Model Training & Prediction

```
# تدريب النموذج
cvModel = pipeline.fit(train_data)

# التنبؤ على مجموعة الاختبار
predictions = cvModel.transform(test_data)
```

Model Evaluation

```
# تقييم النموذج باستخدام AUC
auc = evaluator.evaluate(predictions)
print(f"Area Under ROC: {auc}")

# استخراج أهمية الميزات من أفضل نموذج
best_rf_model = cvModel.stages[-1].bestModel  # استرجاع أفضل نموذج RandomForest
importances = best_rf_model.featureImportances
print(f"Feature Importances: {importances}")

```

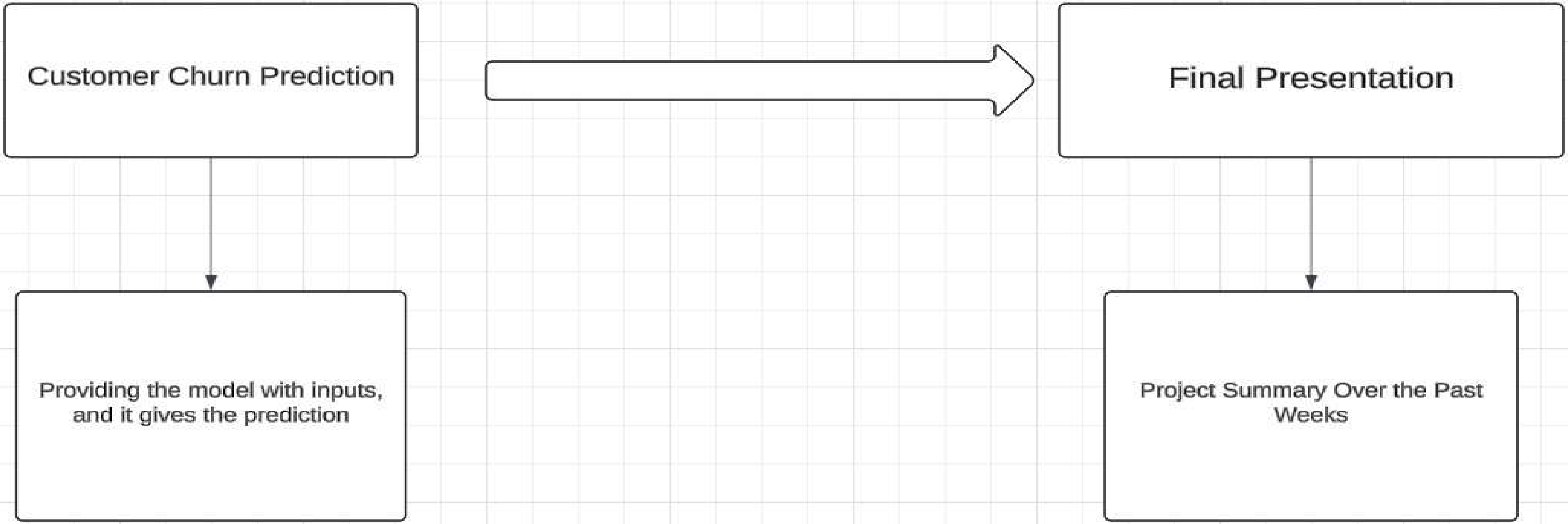
```
feature_columns = ['total_orders', 'total_interactions', 'total_spent', 'customer_lifetime']
```

Results

```
Feature Importances: (4,[0,1,2,3],[0.18801491955270033,0.11321680322382169,0.24472010817663795,0.4540481690468401])
```

Run metrics (2)	
areaUnderROC_test_data	0.8337325349301397
accuracy_test_data	0.9340659340659341

Week 4: Using predictive model and Final Presentation



Prediction according to input

```python
input_data = spark.createDataFrame([
    (26, 5000.00, 20, 30)  # customer_lifetime, total_spent, total_orders, total_interactions
], ["customer_lifetime", "total_spent", "total_orders", "total_interactions"])

predictions = model.transform(input_data)
predictions.show()
```

✓ - Command executed in 830 ms by Suez Gahar on 11:29:44 AM, 10/09/24

```
+-----------------+-----------+------------+------------------+--------------------+--------------------+--------------------+----------+
|customer_lifetime|total_spent|total_orders|total_interactions|            features|       rawPrediction|         probability|prediction|
+-----------------+-----------+------------+------------------+--------------------+--------------------+--------------------+----------+
|               26|     5000.0|          20|                30|[20.0,30.0,5000.0...|[28.3826676907322...|[0.56765335381464...|       0.0|
+-----------------+-----------+------------+------------------+--------------------+--------------------+--------------------+----------+
```

```python
input_data = spark.createDataFrame([
    (300, 15000.00, 30, 50)  # customer_lifetime, total_spent, total_orders, total_interactions
], ["customer_lifetime", "total_spent", "total_orders", "total_interactions"])

predictions = model.transform(input_data)
predictions.show()
```

✓ - Command executed in 900 ms by Suez Gahar on 11:22:51 AM, 10/09/24

```
+-----------------+-----------+------------+------------------+--------------------+--------------------+--------------------+----------+
|customer_lifetime|total_spent|total_orders|total_interactions|            features|       rawPrediction|         probability|prediction|
+-----------------+-----------+------------+------------------+--------------------+--------------------+--------------------+----------+
|              300|    15000.0|          30|                50|[30.0,50.0,15000....|[5.59409381457472...|[0.11188187629149...|       1.0|
+-----------------+-----------+------------+------------------+--------------------+--------------------+--------------------+----------+
```

Prediction according to input

```python
import mlflow
model_uri = "runs:/d8f40238-e76d-4b60-9f55-2a251854c0c3/model"
model = mlflow.spark.load_model(model_uri)

# تمرير بيانات متنوعة للتنبؤ
input_data = spark.createDataFrame([
    (50, 5000.75, 15, 20),  # مثال 1: العميل قضى 50 يومًا، أنفق 5000.75 دولار، قدم 15 طلبًا، وشارك في 20 تفاعل
    (120, 25000.99, 50, 100),  # مثال 2: العميل قضى 120 يومًا، أنفق 25000.99 دولار، قدم 50 طلبًا، وشارك في 100 تفاعل
    (30, 1500.25, 8, 15),  # مثال 3: العميل قضى 30 يومًا. أنفق 1500.25 دولار. قدم 8 طلبات. وشارك في 15 تفاعل
    (200, 50000.00, 100, 250),  # مثال 4: العميل قضى 200 يومًا، أنفق 50000 دولار، قدم 100 طلب، وشارك في 250 تفاعل
    (10, 100.50, 2, 5)  # مثال 5: العميل قضى 10 أيام، أنفق 100.50 دولار، قدم طلبين، وشارك في 5 تفاعلات
], ["customer_lifetime", "total_spent", "total_orders", "total_interactions"])

# إجراء التنبؤات باستخدام النموذج
predictions = model.transform(input_data)
predictions.show()
```

✓ - Command executed in 28 sec 103 ms by Suez Gahar on 11:16:19 AM, 10/09/24 PySpark (Python)

```
2024/10/09 08:15:51 INFO mlflow.spark: 'runs:/d8f40238-e76d-4b60-9f55-2a251854c0c3/model' resolved as 'sds://onelakenortheurope.pbidedicated.windows.net/2d498147-60d2-487b-8282-3fb25813be4e/c13b5d78-500f-4567-919a-d0227f829292/d8f40238-e76d-4b60-9f55-2a251854c0c3/artifacts/model'
Downloading artifacts: 100% 1/1 [00:00<00:00, 11.80it/s]
Downloading artifacts: 100% 28/28 [00:00<00:00, 493.08it/s]
2024/10/09 08:15:52 INFO mlflow.store.artifact.artifact_repo: The progress bar can be disabled by setting the environment variable MLFLOW_ENABLE_ARTIFACTS_PROGRESS_BAR to false
2024/10/09 08:15:53 INFO mlflow.spark: File 'runs:/d8f40238-e76d-4b60-9f55-2a251854c0c3/model/sparkml' not found on DFS. Will attempt to upload the file.
2024/10/09 08:15:55 INFO mlflow.spark: Copied SparkML model to Files/tmp/mlflow/f527c2f7-84d9-4ca6-80a1-179ca57996b5
+-----------------+-----------+------------+------------------+--------------------+--------------------+--------------------+----------+
|customer_lifetime|total_spent|total_orders|total_interactions|            features|       rawPrediction|         probability|prediction|
+-----------------+-----------+------------+------------------+--------------------+--------------------+--------------------+----------+
|               50|    5000.75|          15|                20|[15.0,20.0,5000.7...|[5.60334545080591...|[0.11206690901611...|       1.0|
|              120|   25000.99|          50|               100|[50.0,100.0,25000...|[6.36641919680480...|[0.12732838393609...|       1.0|
|               30|    1500.25|           8|                15|[8.0,15.0,1500.25...|[19.6641018507174...|[0.39328203701434...|       1.0|
|              200|    50000.0|         100|               250|[100.0,250.0,5000...|[3.12654765055833...|[0.06253095301116...|       1.0|
|               10|      100.5|           2|                 5|[2.0,5.0,100.5,10.0]|[19.4713650457543...|[0.38942730091508...|       1.0|
```