Data Warehouse Project Using Pentaho and Mondrian

0. Steps :

1. Data Description

2. Input Data Format

3. ETL with Pentaho

4. Some Visualisation with Python

5. Create Schema with Pentaho Schema Workbench

6. Use Mondrian

1. Data Description

→ The data set used is sales-data-sample, it has 9994 observations and it contains information about sales transactions, with each row representing a single transaction.

Columns Description :

1. **OrderDate:** The date when the order was placed.
2. **Category:** The category of the product.
3. **City:** The city where the order was placed.
4. **Country:** The country where the order was placed.
5. **CustomerName:** The name of the customer who placed the order.
6. **Discount:** The discount applied to the order.
7. **OrderID:** The unique identifier for the order.
8. **PostalCode:** The postal code of the location where the order was placed.
9. **ProductName:** The name of the product.
10. **Profit:** The profit generated from the sale.
11. **Quantity:** The quantity of the product ordered.

12. **Region:** The region where the order was placed.
13. **Sales:** The total sales amount for the transaction.
14. **Segment:** The customer segment.
15. **ShipDate:** The date when the order was shipped.
16. **ShipMode:** The shipping mode for the order.
17. **State:** The state where the order was placed.
18. **Sub_Category:** The sub-category of the product.
19. **DaystoShipActual:** The actual days taken to ship the order.
20. **SalesForecast:** Forecasted sales amount.
21. **ShipStatus:** The status of the shipment.
22. **DaystoShipScheduled:** The scheduled days to ship the order.
23. **OrderProfitable:** Whether the order was profitable.
24. **SalesperCustomer:** Sales per customer.
25. **ProfitRatio:** The ratio of profit to sales.
26. **SalesaboveTarget:** Sales above target.
27. **latitude:** Latitude of the location.
28. **longitude:** Longitude of the location.

2. Input Data Format

To experience creating ETL using data from multiple type of sources I devided the observations into two seperate csv files and loaded one of the data into a postgresql table called **“input-data_1”** and populated it from one of the csv files using a python script.

→ The data is relatively clean.

Data description with python :

```
Data properties:
DF1:

          discount    postalcode       profit     quantity  ...  profitratio  salesabovetarget     latitude    longi
tude
count  4893.000000   4893.000000  4893.000000  4893.000000  ...  4893.000000               0.0  4893.000000  4893.00
0000
mean      0.157893  54074.331290    27.733088     3.793174  ...    11.667300               NaN    37.612121   -93.71
8947
std       0.207810  32024.515397   193.483750     2.225195  ...    46.932372               NaN     4.855674    17.90
2021
min       0.000000   1040.000000 -3702.000000     1.000000  ...  -275.000000               NaN    25.813000  -123.05
5200
25%       0.000000  22153.000000     2.000000     2.000000  ...     7.000000               NaN    34.011600  -115.17
1800
50%       0.200000  54880.000000     9.000000     3.000000  ...    27.000000               NaN    38.744900   -87.63
3600
75%       0.200000  89115.000000    31.000000     5.000000  ...    36.300000               NaN    40.801100   -77.09
9700
max       0.800000  99301.000000  4630.000000    14.000000  ...    50.000000               NaN    47.835300   -68.79
1800
```

```
DF2:

          discount    postalcode       profit     quantity  ...  profitratio  salesabovetarget     latitude    longi
tude
count  5101.000000   5101.000000  5101.000000  5101.000000  ...  5101.000000               0.0  5101.000000  5101.00
0000
mean      0.154581  56260.919232    29.533229     3.786120  ...    12.391806               NaN    37.955183   -94.97
5328
std       0.205148  32067.883763   267.606838     2.225241  ...    46.434286               NaN     4.906674    18.18
8082
min       0.000000   1752.000000 -6600.000000     1.000000  ...  -275.000000               NaN    25.476600  -123.09
9800
25%       0.000000  27604.000000     2.000000     2.000000  ...     7.500000               NaN    34.066000  -117.97
2900
50%       0.200000  60505.000000     9.000000     3.000000  ...    27.000000               NaN    39.205500   -88.29
7100
75%       0.200000  90032.000000    28.000000     5.000000  ...    36.300000               NaN    40.801100   -79.00
8000
max       0.800000  99301.000000  8400.000000    14.000000  ...    50.000000               NaN    48.797400   -68.79
1800

[8 rows x 14 columns]
```

```
Number of null values in each column:
DF1:

orderdate                0
category                 0
city                     0
country                  0
customername             0
discount                 0
orderid                  0
postalcode               0
productname              0
profit                   0
quantity                 0
region                   0
sales                    0
segment                  0
shipdate                 0
shipmode                 0
state                    0
sub_category             0
daystoshipactual         0
```

```
profit                      0
quantity                    0
region                      0
sales                       0
segment                     0
shipdate                    0
shipmode                    0
state                       0
sub_category                0
daystoshipactual            0
salesforecast               0
shipstatus                  0
daystoshipscheduled         0
orderprofitable          4893
salespercustomer            0
profitratio                 0
salesabovetarget         4893
latitude                    0
longitude                   0
dtype: int64
```

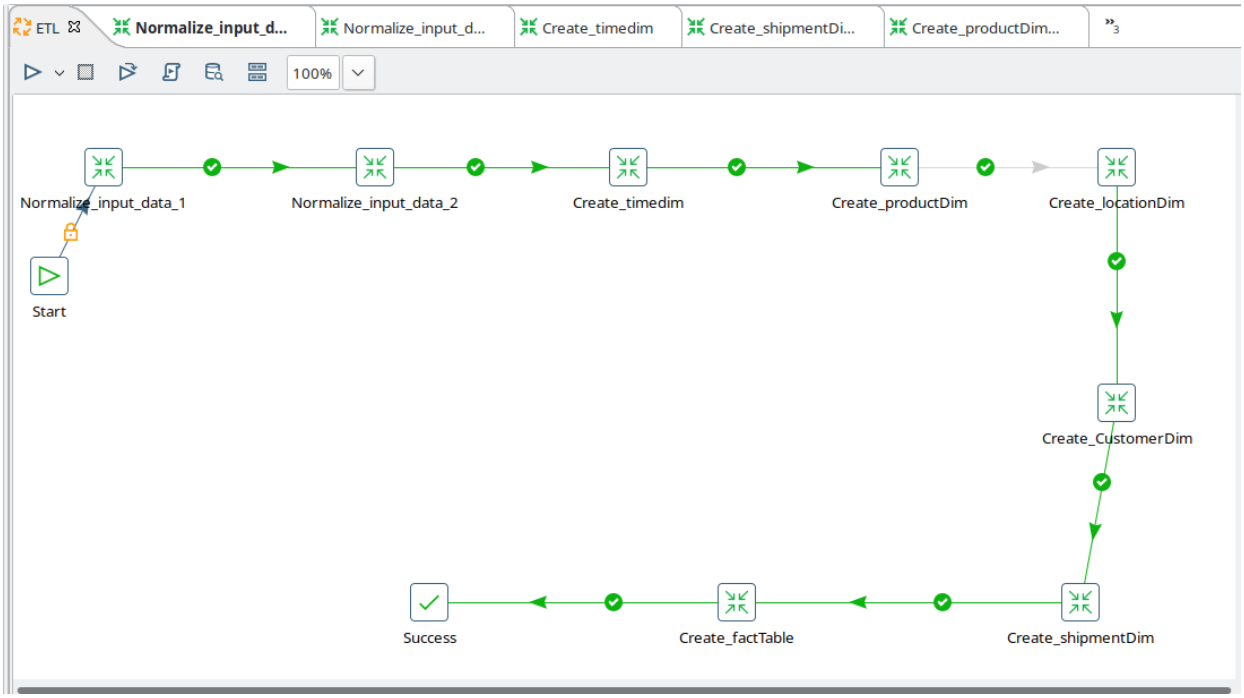
```
DF2:

orderdate              0
category               0
city                   0
country                0
customername           0
discount               0
orderid                0
postalcode             0
productname            0
profit                 0
quantity               0
region                 0
sales                  0
segment                0
shipdate               0
shipmode               0
state                  0
sub_category           0
```

```
productname              0
profit                   0
quantity                 0
region                   0
sales                    0
segment                  0
shipdate                 0
shipmode                 0
state                    0
sub_category             0
daystoshipactual         0
salesforecast            0
shipstatus               0
daystoshipscheduled      0
orderprofitable       5101
salespercustomer         0
profitratio              0
salesabovetarget      5101
latitude                 0
longitude                0
dtype: int64
```

→ “orderprofitable” and “salesabovetarget” are almost always null so we can just remove them.

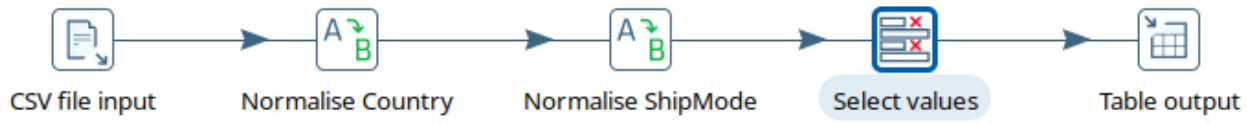
3. ETL with Pentaho

- The Job :



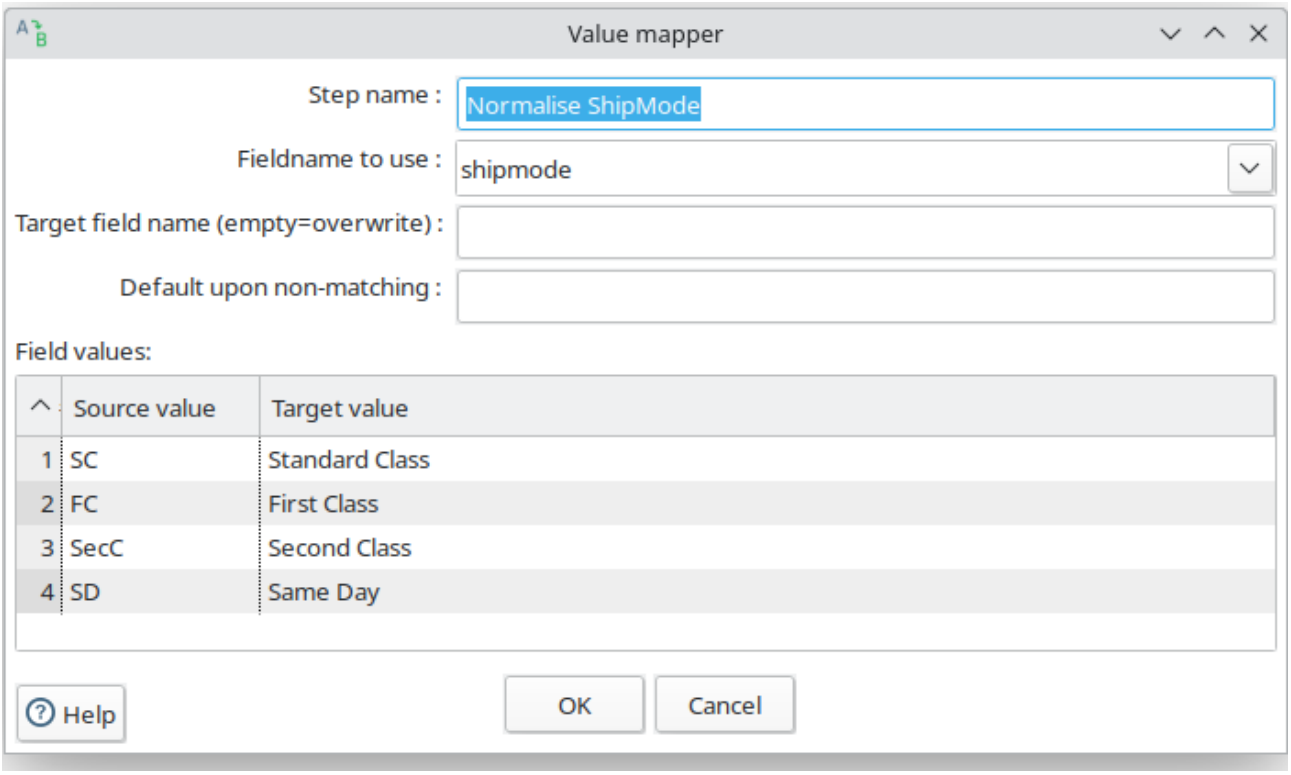
➢ Each Transformation Description :

1. Normalize_input_data_1:

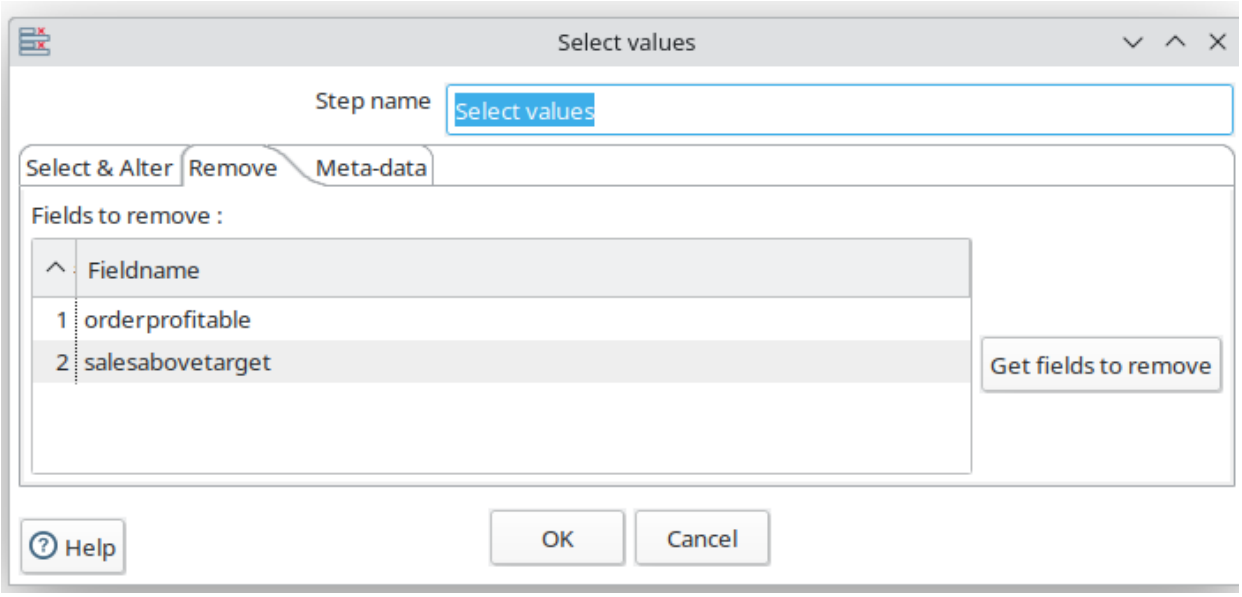


→ The input is the first csv file (‘I changed the values of country and shipmode for this file using a shell script to add a step of normalising value in the ETL because the data was clean’ ex: sed -i 's/United States/US/g' data_1.csv).

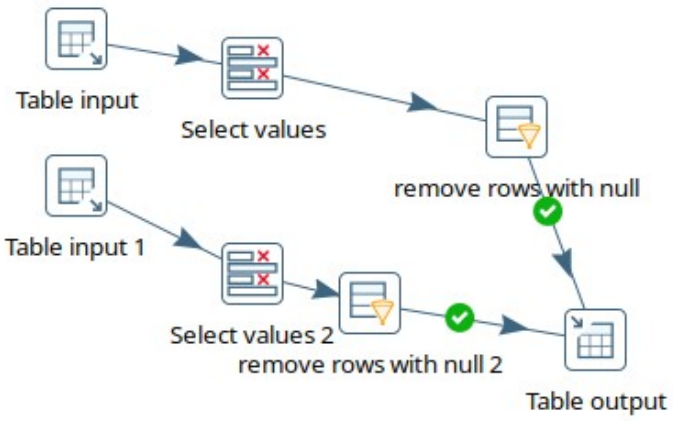
- So the first step is normalising the country values to match the other input file. Then normalising the shipmode.



- The we would remove the columns “orderprofitable” and “salesabovetarget” .



2. Normalize_input_data_2:



- ✗ This step is mainly about merging the data from both sources into one table called “input_data”

```
dwproject=# \d input_data
                    Table "public.input_data"
       Column        |       Type       | Collation | Nullable | Default
---------------------+------------------+-----------+----------+---------
 orderdate           | text             |           |          |
 category            | text             |           |          |
 city                | text             |           |          |
 country             | text             |           |          |
 customername        | text             |           |          |
 discount            | double precision |           |          |
 orderid             | text             |           |          |
 postalcode          | bigint           |           |          |
 productname         | text             |           |          |
 profit              | bigint           |           |          |
 quantity            | bigint           |           |          |
 region              | text             |           |          |
 sales               | bigint           |           |          |
 segment             | text             |           |          |
 shipdate            | text             |           |          |
 shipmode            | text             |           |          |
 state               | text             |           |          |
 sub_category        | text             |           |          |
 daystoshipactual    | bigint           |           |          |
 salesforecast       | bigint           |           |          |
 shipstatus          | text             |           |          |
 daystoshipscheduled | bigint           |           |          |
 salespercustomer    | double precision |           |          |
 profitratio         | double precision |           |          |
 latitude            | double precision |           |          |
 longitude           | double precision |           |          |
```

3. Creating fact and dimensions table :

-- Create Dimension Tables

```
CREATE TABLE CustomerDim (
    customerid SERIAL PRIMARY KEY,
    customername VARCHAR(255),
    segment VARCHAR(50),
    country VARCHAR(100),
    state VARCHAR(100),
    city VARCHAR(100),
    postalcode VARCHAR(20)
);
```

```
CREATE TABLE ProductDim (
    productid SERIAL PRIMARY KEY,
    productname VARCHAR(255),
    category VARCHAR(100),
    sub_category VARCHAR(100)
);

CREATE TABLE LocationDim (
    locationid SERIAL PRIMARY KEY,
    city VARCHAR(100),
    state VARCHAR(100),
    country VARCHAR(100),
    postalcode VARCHAR(20),
    latitude DECIMAL(9,6),
    longitude DECIMAL(9,6),
    region VARCHAR(100)
);

CREATE TABLE TimeDim (
    timeid SERIAL PRIMARY KEY,
    year INT,
    month INT,
    day INT,
    quarter INT,
    weekday INT
);

CREATE TABLE ShipmentDim (
    shipmentid SERIAL PRIMARY KEY,
    daystoshipscheduled bigint,
    shipstatus text,
    daystoshipactual bigint,
```

shipmode text,

shipdate text

);

→ There are file dimensions: ‘customer dimension’, ‘location dimension’, ‘product dimension’, ‘shipment dimension’, ‘time dimension’.

-- Create Fact Table

CREATE TABLE FactTable (

FactID SERIAL PRIMARY KEY,

CustomerID INT,

ProductID INT,

LocationID INT,

TimeID INT,

ShipmentID INT,

OrderID TEXT,

Discount DECIMAL,

Profit DECIMAL,

Quantity INT,

Sales DECIMAL,

SalesForecast DECIMAL,

SalesPerCustomer DECIMAL,

ProfitRatio DECIMAL,

CONSTRAINT fk_customer FOREIGN KEY (CustomerID) REFERENCES CustomerDim(CustomerID),

CONSTRAINT fk_product FOREIGN KEY (ProductID) REFERENCES ProductDim(ProductID),

CONSTRAINT fk_location FOREIGN KEY (LocationID) REFERENCES LocationDim(LocationID),

CONSTRAINT fk_time FOREIGN KEY (TimeID) REFERENCES timedim(timeid),

CONSTRAINT fk_shipment FOREIGN KEY (ShipmentID) REFERENCES ShipmentDim(ShipmentID)

);

→ The fact table has the foreign keys of all the dimension tables + the other remaining columns.

```
dwproject=# \d facttable
                                Table "public.facttable"
      Column       |  Type   | Collation | Nullable |                  Default
-------------------+---------+-----------+----------+-------------------------------------------
 factid            | integer |           | not null | nextval('facttable_factid_seq'::regclass)
 customerid        | integer |           |          |
 productid         | integer |           |          |
 locationid        | integer |           |          |
 timeid            | integer |           |          |
 shipmentid        | integer |           |          |
 orderid           | text    |           |          |
 discount          | numeric |           |          |
 profit            | numeric |           |          |
 quantity          | integer |           |          |
 sales             | numeric |           |          |
 salesforecast     | numeric |           |          |
 salespercustomer  | numeric |           |          |
 profitratio       | numeric |           |          |
Indexes:
    "facttable_pkey" PRIMARY KEY, btree (factid)
Foreign-key constraints:
    "fk_customer" FOREIGN KEY (customerid) REFERENCES customerdim(customerid)
    "fk_location" FOREIGN KEY (locationid) REFERENCES locationdim(locationid)
    "fk_product" FOREIGN KEY (productid) REFERENCES productdim(productid)
    "fk_shipment" FOREIGN KEY (shipmentid) REFERENCES shipmentdim(shipmentid)
    "fk_time" FOREIGN KEY (timeid) REFERENCES timedim(timeid)
```

-- Populating the Dimension Tables and Fact Table

→ Using the transformations ‘Create_timeDim’ , ‘Create_ProductDim’, ‘Create_LocationDim’, ‘Create_ShipmentDim’, ‘Create_facttable’.

They contain and ExecuteSQLScript each one contain the script to populate a table.

-- Populate Dimension Tables

```
INSERT INTO public.customerdim (customername, segment, country, state, city, postalcode)
SELECT DISTINCT
    customername,
    segment,
    country,
    state,
    city,
    postalcode
FROM public.input_data;
```

```
INSERT INTO public.productdim (productname, category, sub_category)
SELECT DISTINCT
    productname,
    category,
    sub_category
FROM public.input_data;

INSERT INTO public.locationdim (city, state, country, postalcode, latitude, longitude, region)
SELECT DISTINCT
    city,
    state,
    country,
    postalcode,
    latitude,
    longitude,
    region
FROM public.input_data;

INSERT INTO TimeDim (Year, Month, Day, Quarter, Weekday)
SELECT DISTINCT
    DATE_PART('YEAR', CAST(orderdate AS TIMESTAMP)) AS Year,
    DATE_PART('MONTH', CAST(orderdate AS TIMESTAMP)) AS Month,
    DATE_PART('DAY', CAST(orderdate AS TIMESTAMP)) AS Day,
    DATE_PART('QUARTER', CAST(orderdate AS TIMESTAMP)) AS Quarter,
    EXTRACT(ISODOW FROM CAST(orderdate AS TIMESTAMP)) AS Weekday
FROM public.input_data;

INSERT INTO public.shipmentdim (daystoshipscheduled, shipstatus, daystoshipactual, shipmode,
shipdate)
SELECT DISTINCT
    daystoshipscheduled,
    shipstatus,
```

daystoshipactual,

shipmode,

shipdate

FROM public.input_data;

-- Populate Fact Table

INSERT INTO FactTable (CustomerID, ProductID, LocationID, TimeID, ShipmentID, OrderID, Discount, Profit, Quantity, Sales, SalesForecast, SalesPerCustomer, ProfitRatio)

SELECT

cd.CustomerID,

pd.ProductID,

ld.LocationID,

td.TimeID,

sd.ShipmentID,

od.OrderID,

od.discount,

od.profit,

od.quantity,

od.sales,

od.salesforecast,

od.salespercustomer,

od.profitratio

FROM

public.input_data od

JOIN

public.customerdim cd

ON od.customername = cd.customername

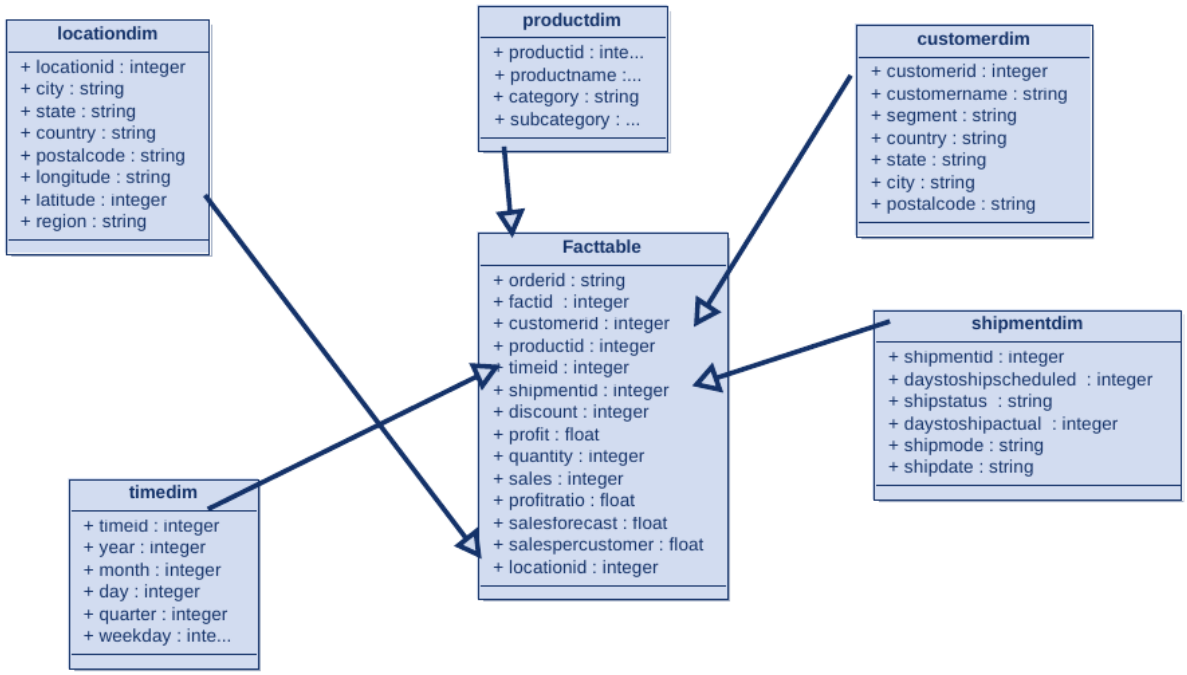
AND od.segment = cd.segment

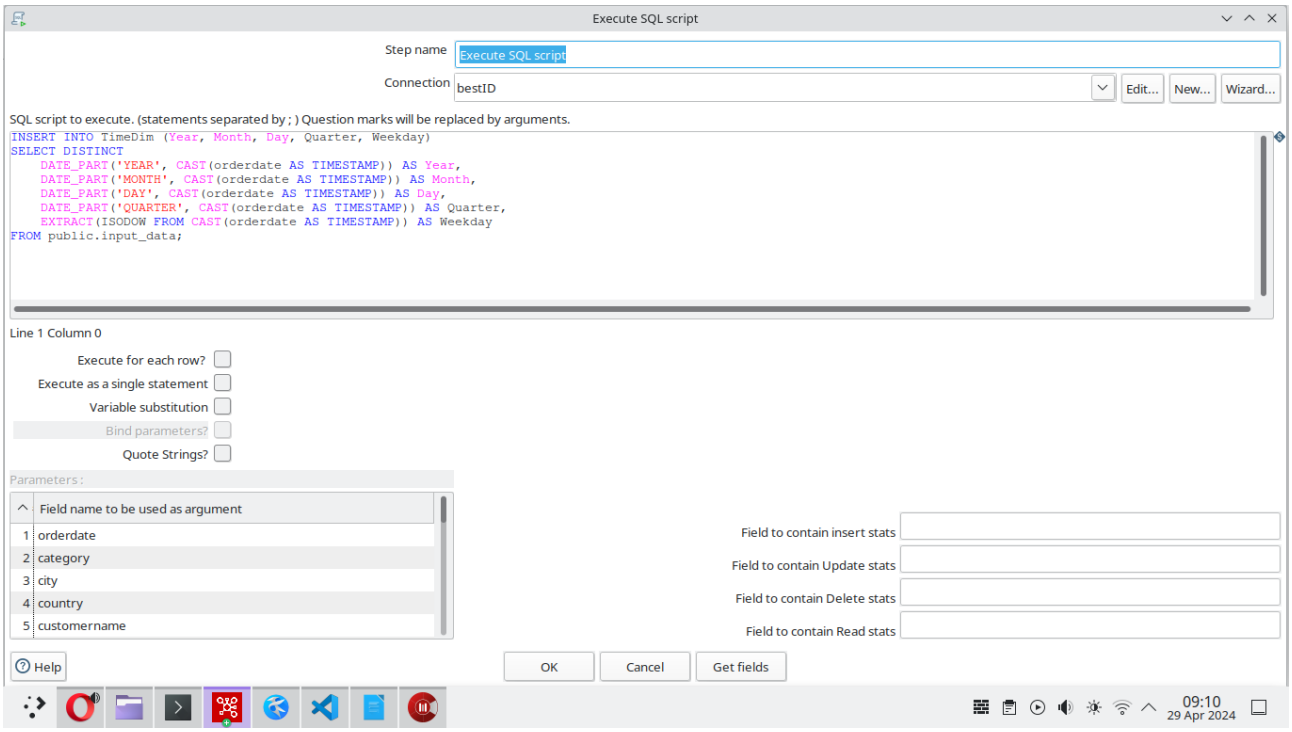
AND od.country = cd.country

AND od.state = cd.state

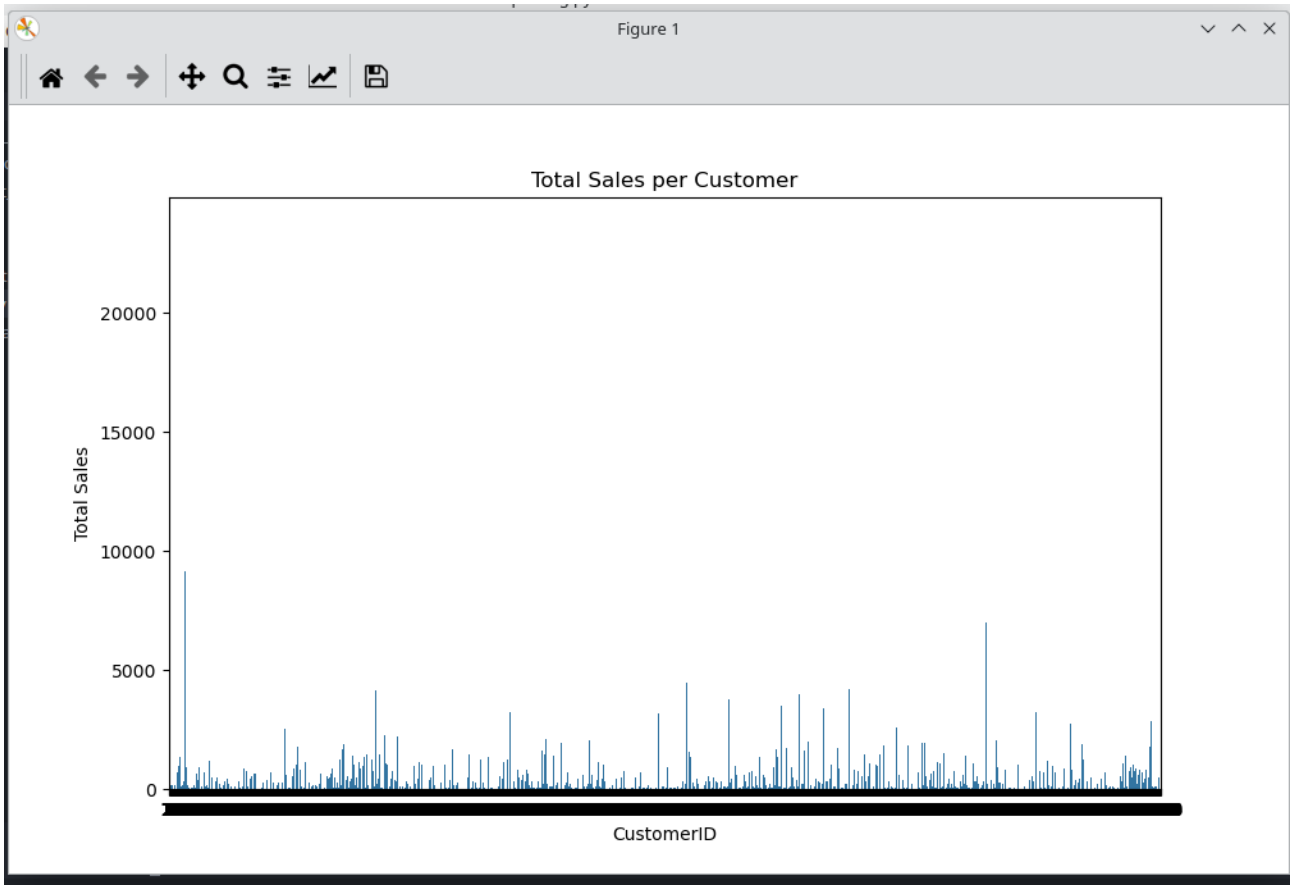
AND od.city = cd.city

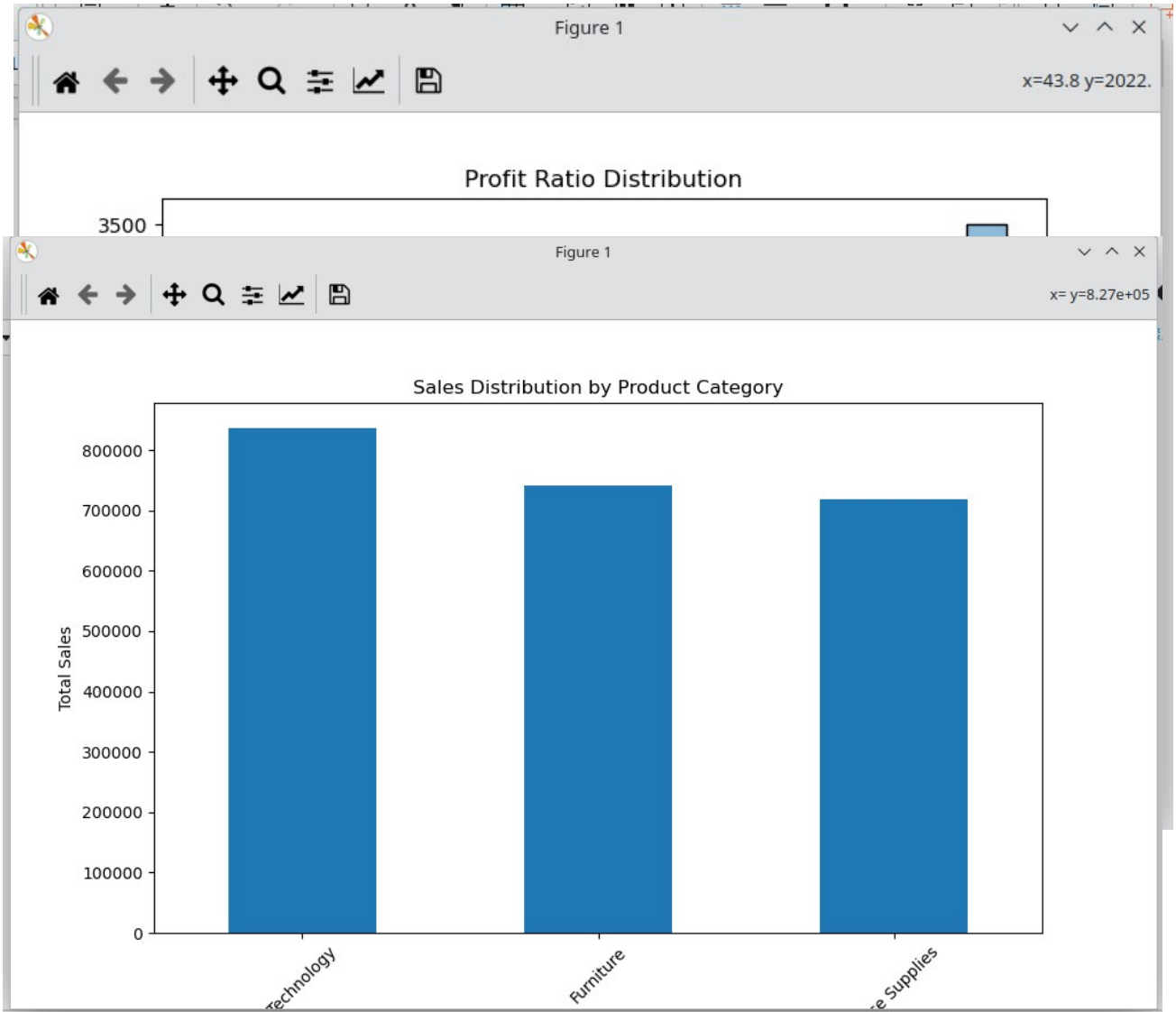
```
    AND od.postalcode::text = cd.postalcode
JOIN
    public.productdim pd
    ON od.productname = pd.productname
    AND od.category = pd.category
    AND od.sub_category = pd.sub_category
JOIN public.locationdim ld
    ON od.city = ld.city
    AND od.state = ld.state
    AND od.country = ld.country
    AND od.latitude = ld.latitude
    AND od.longitude = ld.longitude
    AND od.region = ld.region
    AND od.postalcode::text = ld.postalcode
JOIN public.timedim td
    ON DATE_PART('YEAR', CAST(od.orderdate AS TIMESTAMP)) = td.year
    AND DATE_PART('MONTH', CAST(od.orderdate AS TIMESTAMP)) = td.month
    AND DATE_PART('DAY', CAST(od.orderdate AS TIMESTAMP)) = td.day
    AND DATE_PART('QUARTER', CAST(od.orderdate AS TIMESTAMP)) = td.quarter
    AND EXTRACT(ISODOW FROM CAST(od.orderdate AS TIMESTAMP)) = td.weekday
JOIN public.shipmentdim sd ON od.daystoshipscheduled = sd.daystoshipscheduled
                    AND od.daystoshipactual = sd.daystoshipactual
                    AND od.shipstatus = sd.shipstatus
                    AND od.shipmode = sd.shipmode
                    AND DATE(od.shipdate) = sd.shipdate::date;
```


locationdim
+ locationid : integer
+ city : string
+ state : string
+ country : string
+ postalcode : string
+ longitude : string
+ latitude : integer
+ region : string
productdim
+ productid : inte...
+ productname :...
+ category : string
+ subcategory : ...
customerdim
+ customerid : integer
+ customername : string
+ segment : string
+ country : string
+ state : string
+ city : string
+ postalcode : string
Facttable
+ orderid : string
+ factid : integer
+ customerid : integer
+ productid : integer
+ timeid : integer
+ shipmentid : integer
+ discount : integer
+ profit : float
+ quantity : integer
+ sales : integer
+ profitratio : float
+ salesforecast : float
+ salespercustomer : float
+ locationid : integer
shipmentdim
+ shipmentid : integer
+ daystoshipscheduled : integer
+ shipstatus : string
+ daystoshipactual : integer
+ shipmode : string
+ shipdate : string
timedim
+ timeid : integer
+ year : integer
+ month : integer
+ day : integer
+ quarter : integer
+ weekday : inte...


Execute SQL script
Step name
Execute SQL script
Connection
bestID
Edit...
New...
Wizard...
SQL script to execute. (statements separated by ;) Question marks will be replaced by arguments.
INSERT INTO TimeDim (Year, Month, Day, Quarter, Weekday)
SELECT DISTINCT
DATE_PART('YEAR', CAST(orderdate AS TIMESTAMP)) AS Year,
DATE_PART('MONTH', CAST(orderdate AS TIMESTAMP)) AS Month,
DATE_PART('DAY', CAST(orderdate AS TIMESTAMP)) AS Day,
DATE_PART('QUARTER', CAST(orderdate AS TIMESTAMP)) AS Quarter,
EXTRACT(ISODOW FROM CAST(orderdate AS TIMESTAMP)) AS Weekday
FROM public.input_data;
Line 1 Column 0
Execute for each row?
Execute as a single statement
Variable substitution
Bind parameters?
Quote Strings?
Parameters :
Field name to be used as argument
1 orderdate
2 category
3 city
4 country
5 customername
Field to contain insert stats
Field to contain Update stats
Field to contain Delete stats
Field to contain Read stats
Help
OK
Cancel
Get fields
09:10
29 Apr 2024

4. Some Visualisation with Python





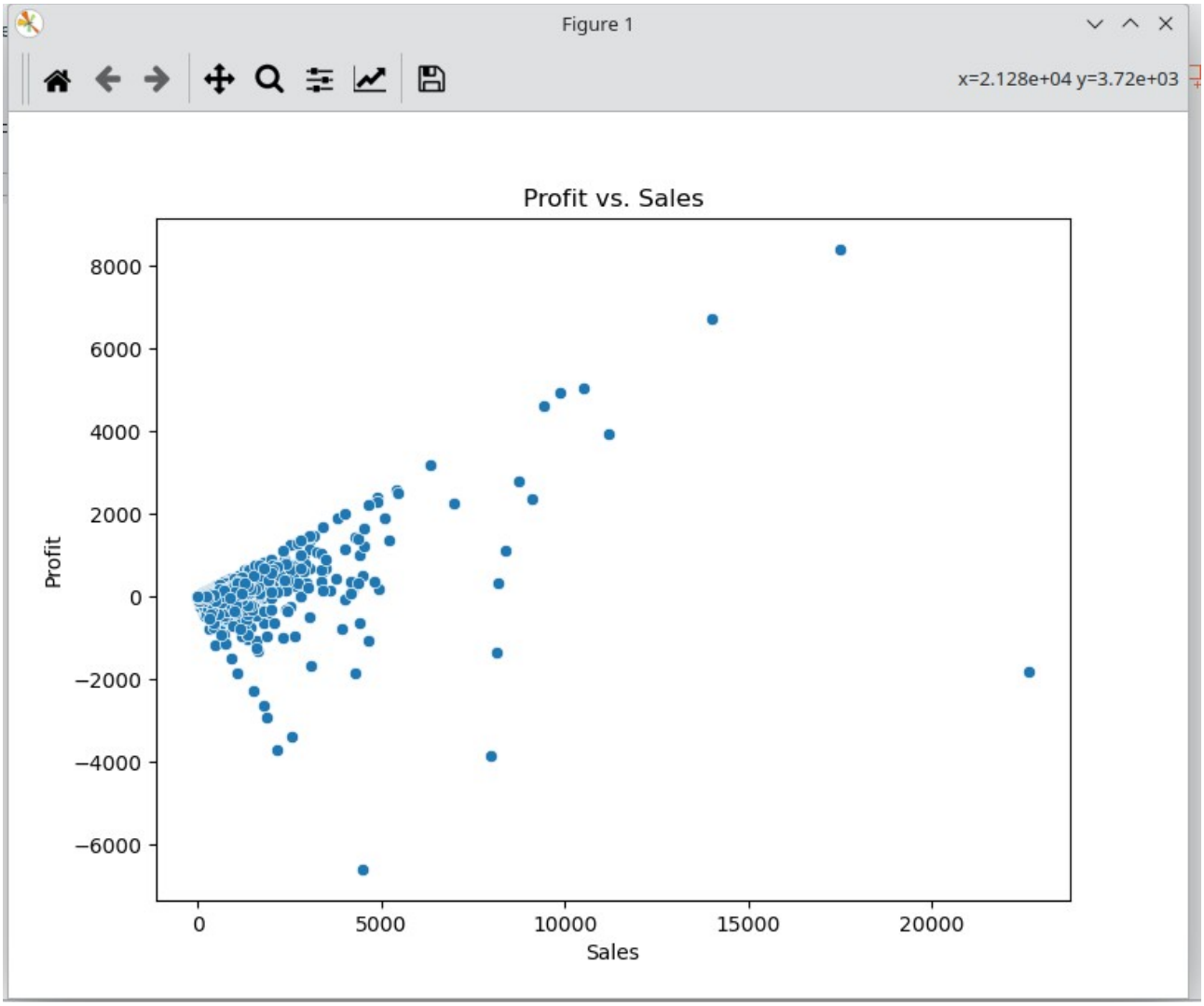
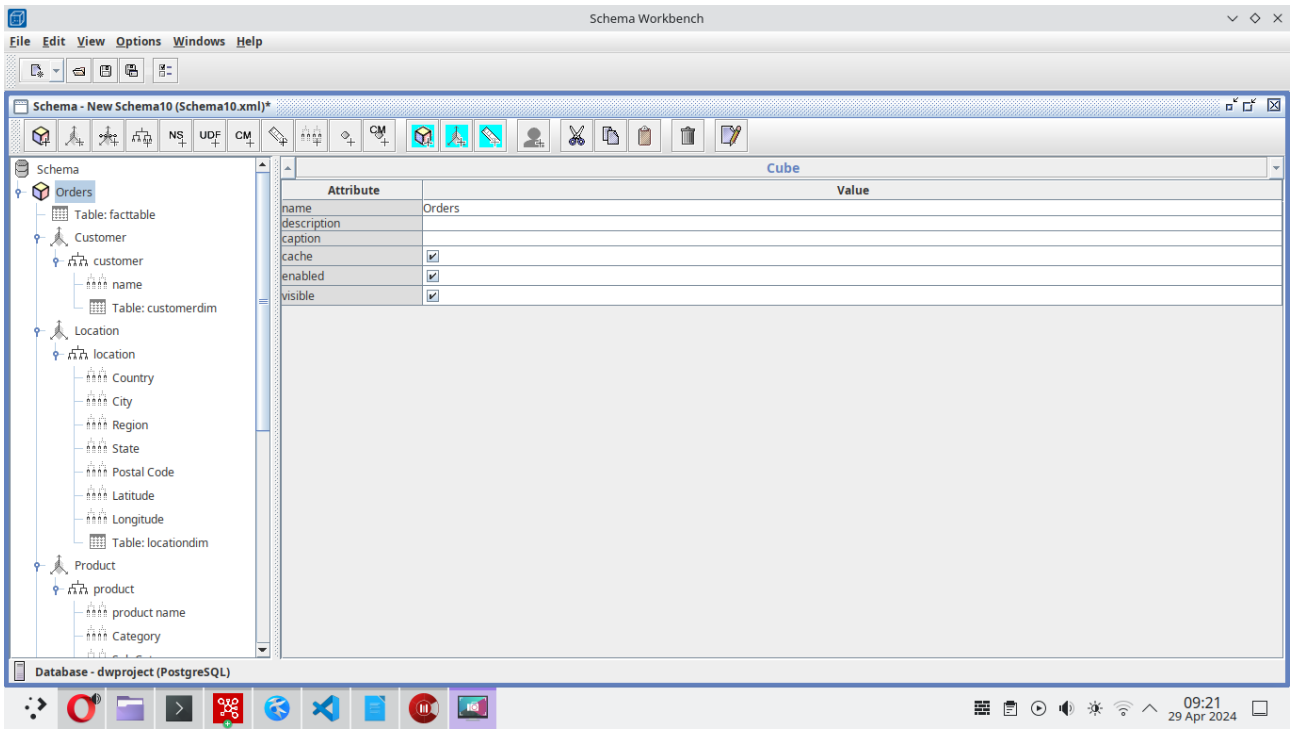


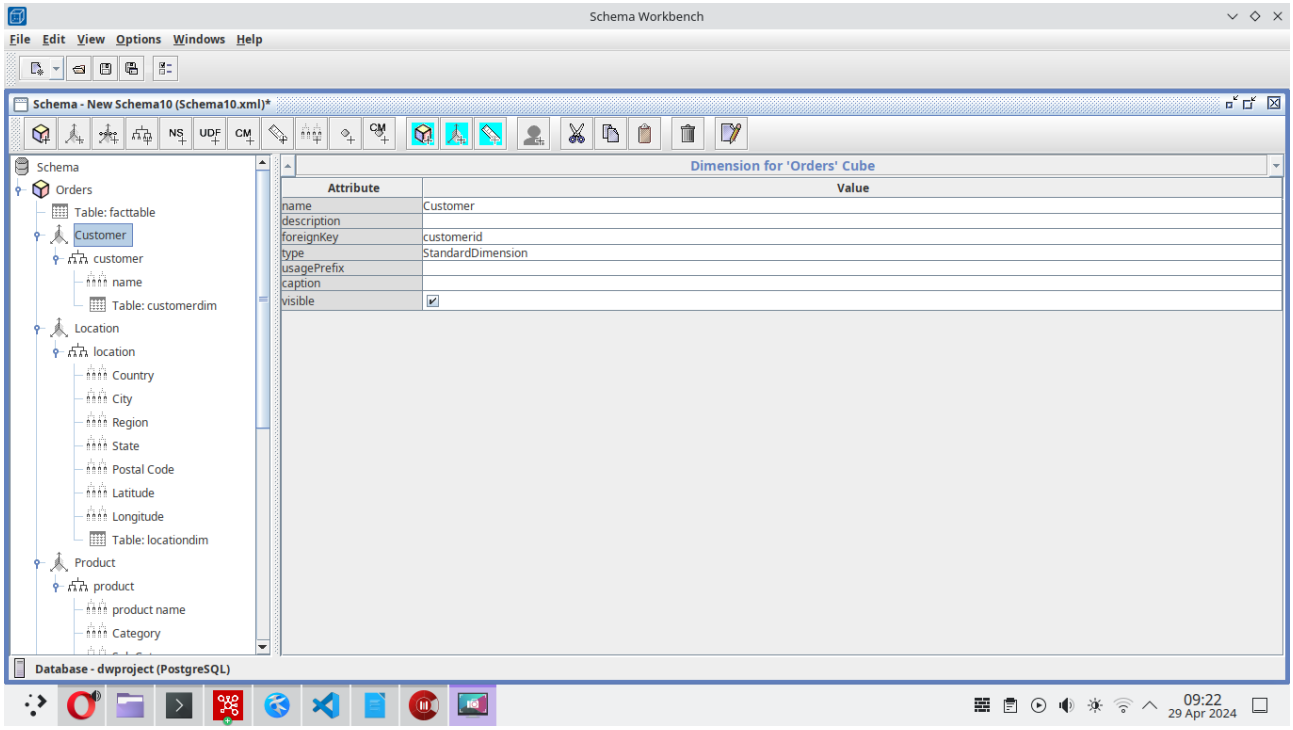
Figure 1
x=2.128e+04 y=3.72e+03
Profit vs. Sales
Profit
8000
6000
4000
2000
0
−2000
−4000
−6000
0
5000
10000
15000
20000
Sales

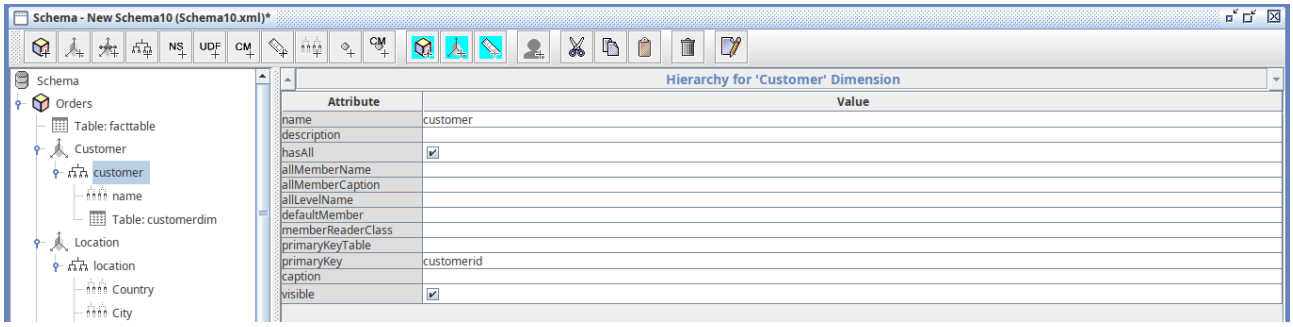
5. Create Schema with Pentaho Schema Workbench

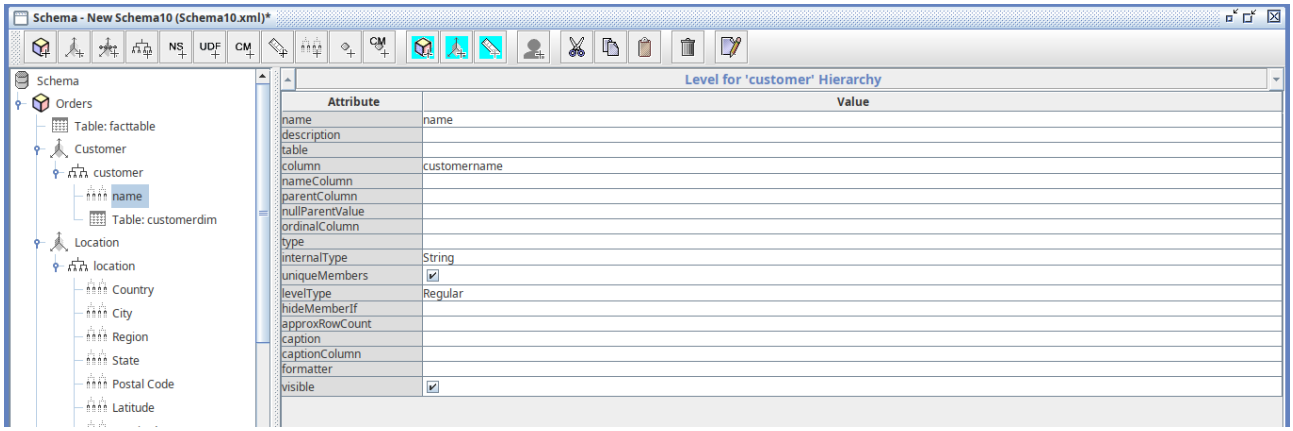
a) Connecting to database:

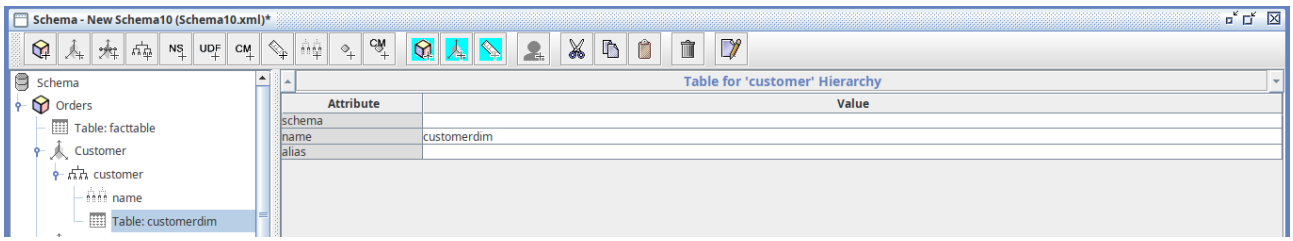
b) Creating Cube and Dimensions:




Schema Workbench
File Edit View Options Windows Help
Schema - New Schema10 (Schema10.xml)*
Schema
Orders
Table: facttable
Customer
customer
name
Table: customerdim
Location
location
Country
City
Region
State
Postal Code
Latitude
Longitude
Table: locationdim
Product
product
product name
Category
Dimension for 'Orders' Cube
Attribute
Value
name
Customer
description
foreignKey
customerid
type
StandardDimension
usagePrefix
caption
visible
Database - dwproject (PostgreSQL)
09:22
29 Apr 2024

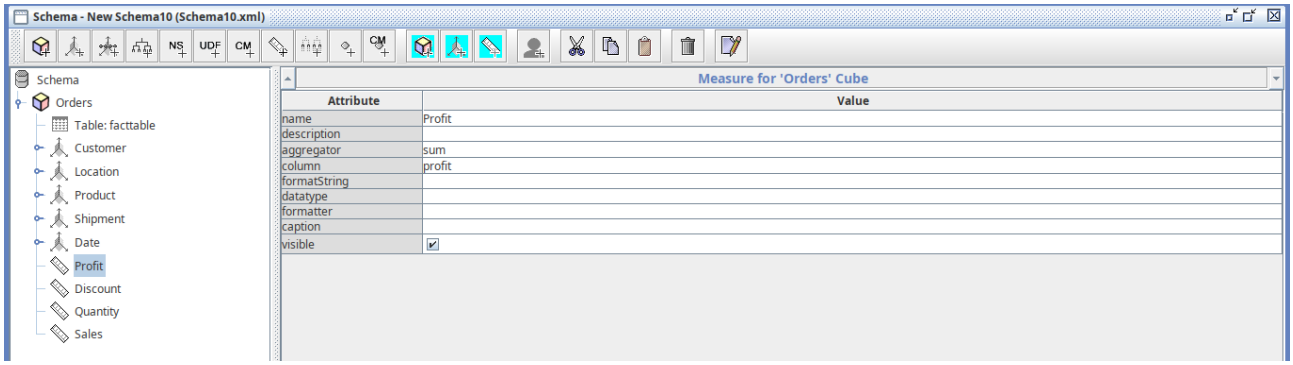

Schema - New Schema10 (Schema10.xml)*
Schema
Orders
Table: facttable
Customer
customer
name
Table: customerdim
Location
location
Country
City
Hierarchy for 'Customer' Dimension
Attribute
Value
name
customer
description
hasAll
allMemberName
allMemberCaption
allLevelName
defaultMember
memberReaderClass
primaryKeyTable
primaryKey
customerid
caption
visible


Schema - New Schema10 (Schema10.xml)*
Schema
Orders
Table: facttable
Customer
customer
name
Table: customerdim
Location
location
Country
City
Region
State
Postal Code
Latitude
Level for 'customer' Hierarchy
Attribute
Value
name
name
description
table
column
customername
nameColumn
parentColumn
nullParentValue
ordinalColumn
type
internalType
String
uniqueMembers
levelType
Regular
hideMemberIf
approxRowCount
caption
captionColumn
formatter
visible

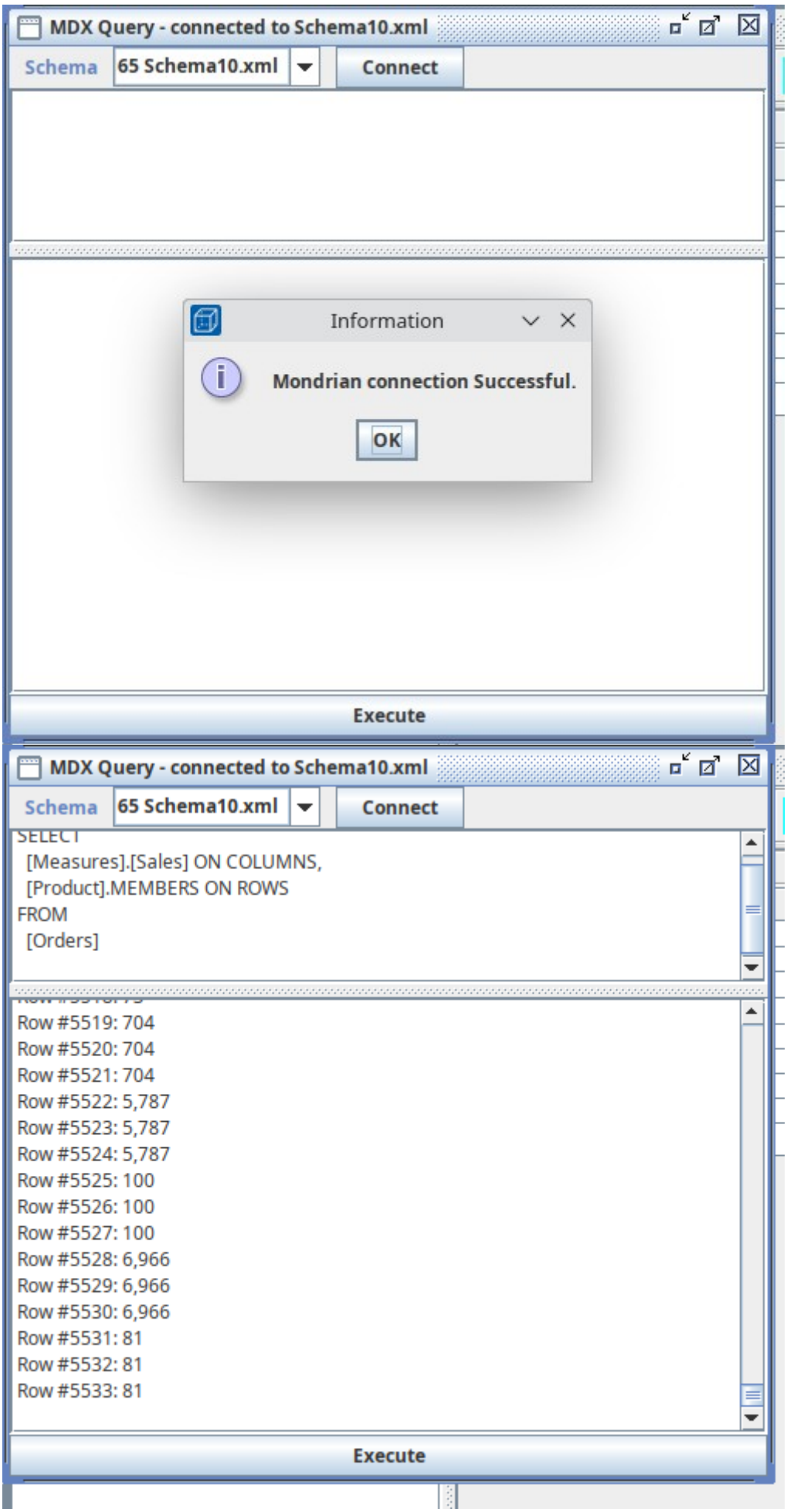


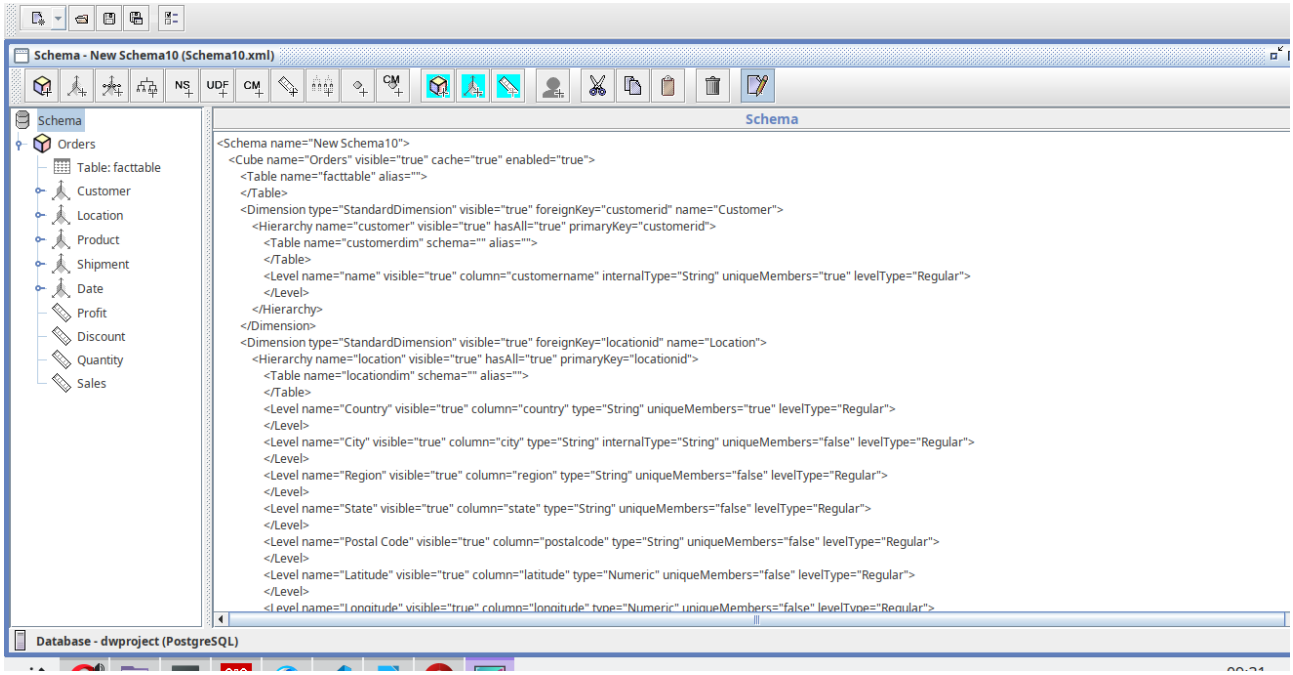
c) Creating Measures :

I created 4 measures: ‘Profit’, ‘Discount’, ‘Quantity’, ‘Sales’.



d) Try running MDX query:

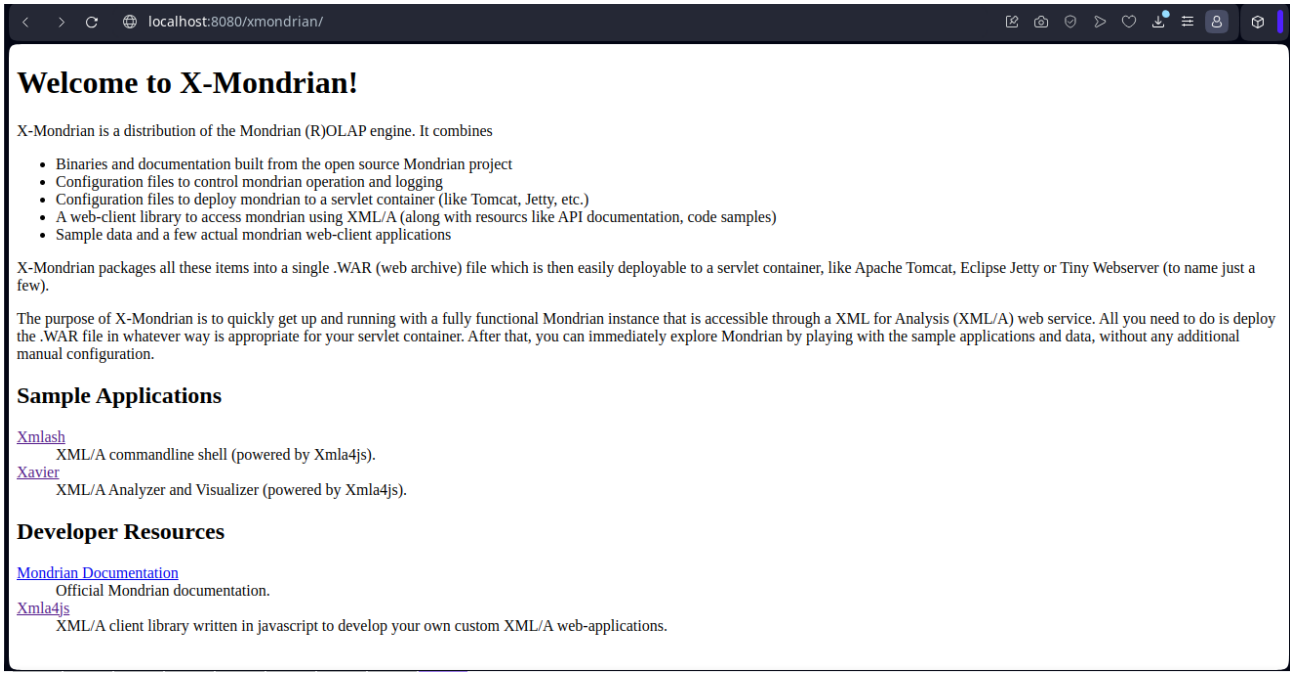

MDX Query - connected to Schema10.xml
Schema
65 Schema10.xml
Connect
Information
Mondrian connection Successful.
OK
Execute
MDX Query - connected to Schema10.xml
Schema
65 Schema10.xml
Connect
SELECT
[Measures].[Sales] ON COLUMNS,
[Product].MEMBERS ON ROWS
FROM
[Orders]
Row #5519: 704
Row #5520: 704
Row #5521: 704
Row #5522: 5,787
Row #5523: 5,787
Row #5524: 5,787
Row #5525: 100
Row #5526: 100
Row #5527: 100
Row #5528: 6,966
Row #5529: 6,966
Row #5530: 6,966
Row #5531: 81
Row #5532: 81
Row #5533: 81
Execute



6. Use Mondrian

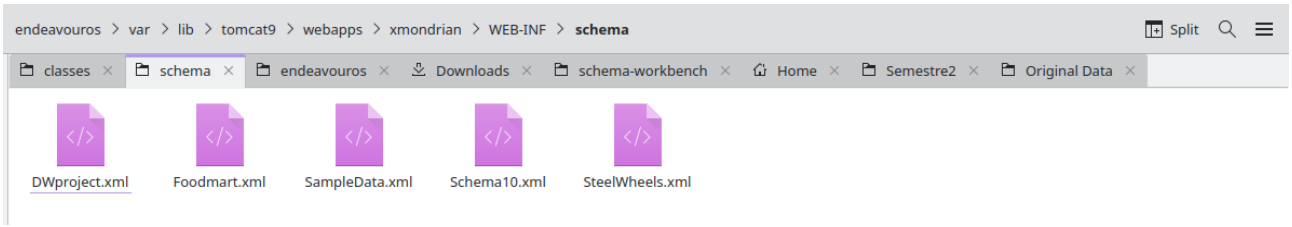
a) Setting up :

To start mondrian you need to download the .war file and put it in the tomcat/webapp directory then you would be able to access mondrian (http://localhost:8080/xmondrian/)



b) Access the schema created:

To be able to acces the schema created you need to add the .xml file to (‘/tomcat9/webapps/xmondrian/WEB-INF/schema/’).



Change the configuration in the file (‘/tomcat9/webapps/xmondrian/WEB-INF/classes/mondrian.properties’) to add the configuration for the relational database used.

```
Provider=mondrian;
mondrian.dwproject.jdbcURL=jdbc:postgresql://localhost:5432/dwproject;
mondrian.dwproject.jdbcUser=postgres
mondrian.dwproject.jdbcPassword=12345
mondrian.jdbcDrivers=org.postgresql.Driver
Catalog=file:/home/rania/Documents/CI (University)/CI2/Semestre2/Data Warehouse/Project/project-f/Mondrian/DWproject.xml
```

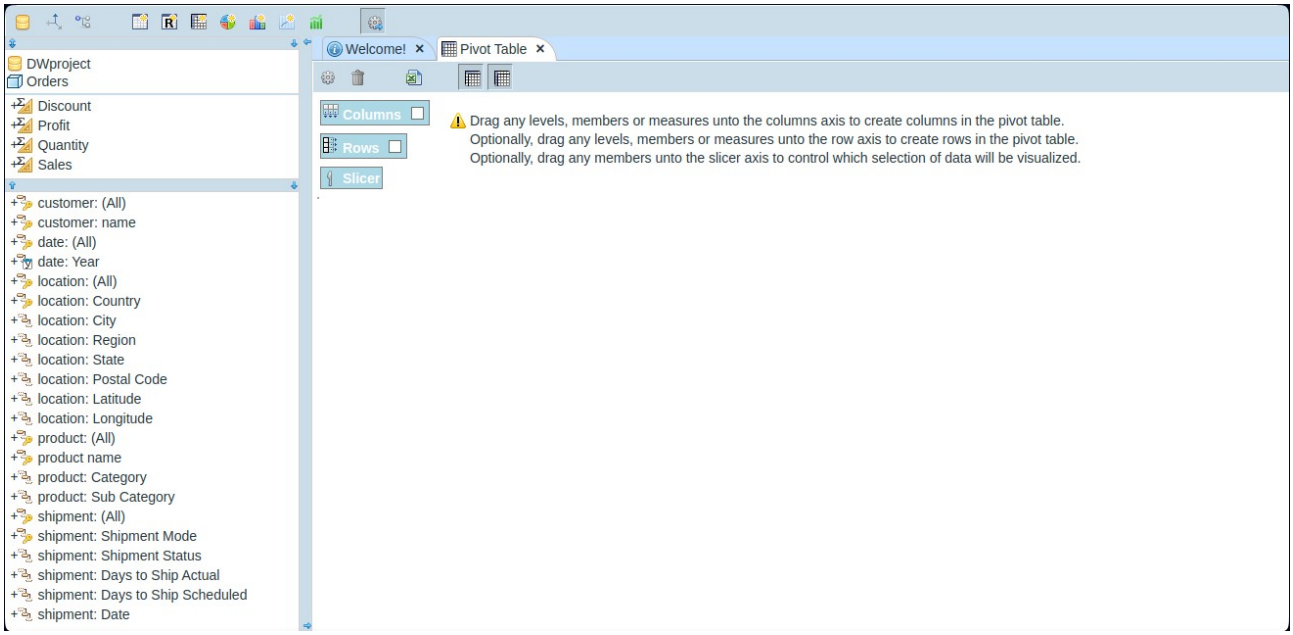
Configure the data in the (‘/tomcat9/webapps/xmondrian/WEB-INF/datasources.xml’) file.

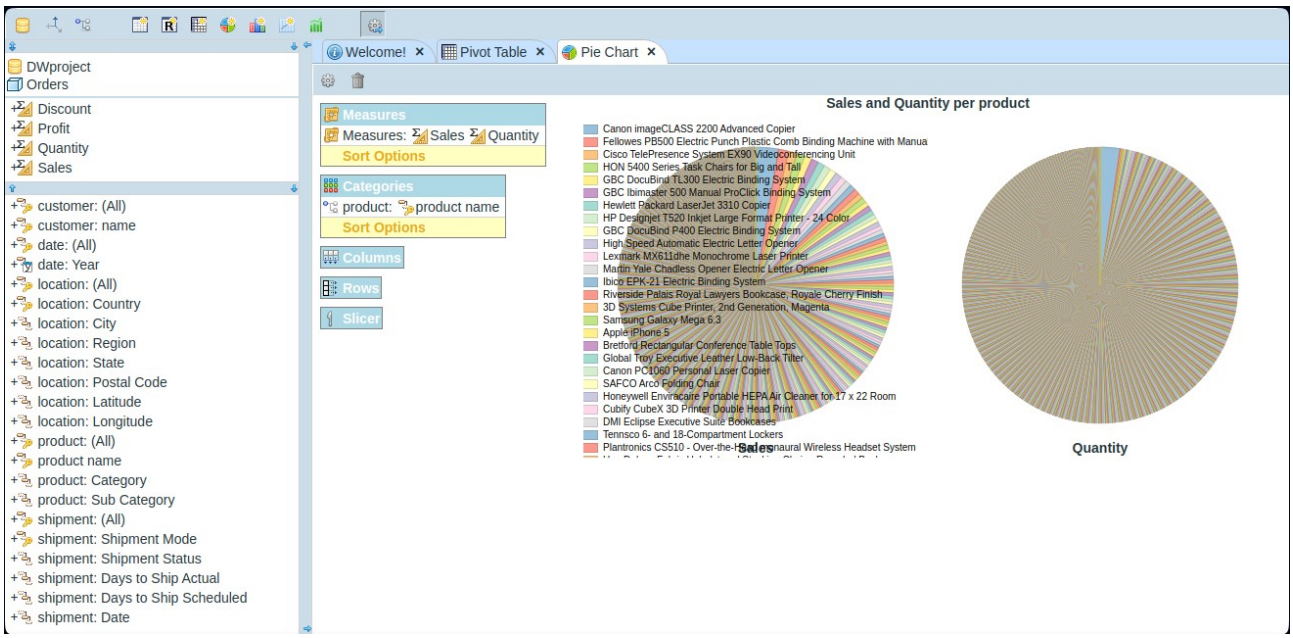
var > lib > tomcat9 > webapps > xmondrian > WEB-INF > datasources.xml

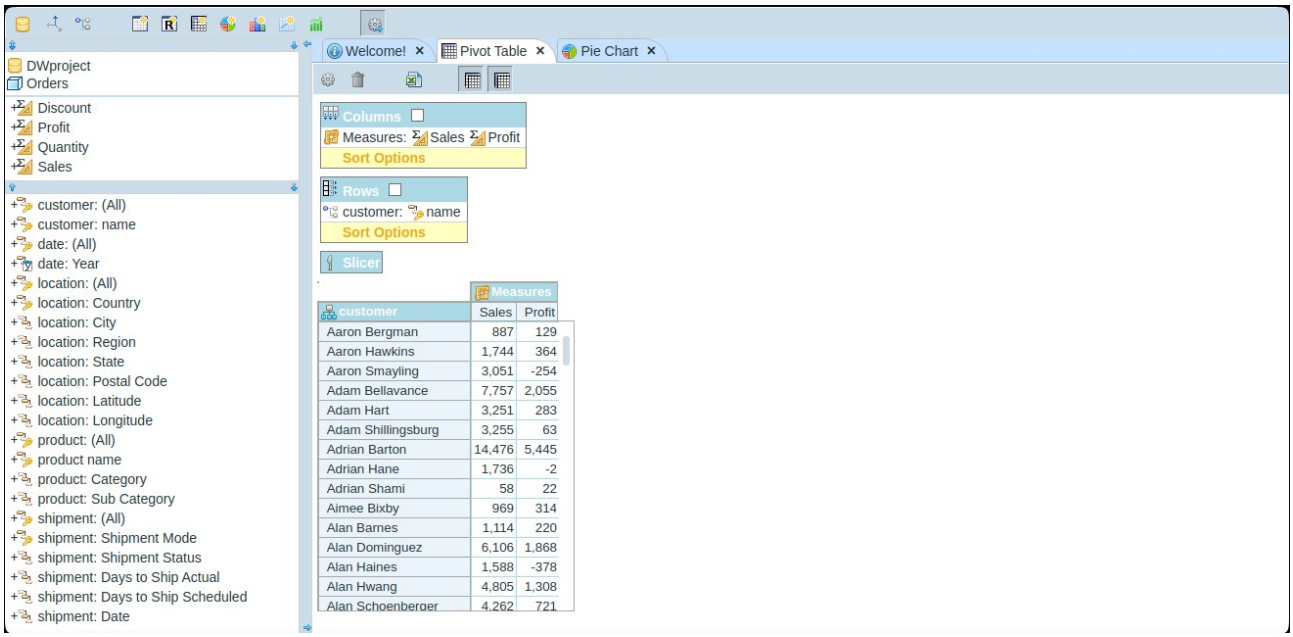
```
<DataSources>
    <!-- Data source for DWproject -->
    <DataSource>
        <DataSourceName>Dwproject</DataSourceName>
        <DataSourceDescription>Connects to the DW project</DataSourceDescription>
        <URL>http://localhost:8080/mondrian/xmla</URL>
        <DataSourceInfo>
            Provider=mondrian;
            Jdbc=jdbc:postgresql://localhost:5432/dwproject;
            JdbcDrivers=org.postgresql.Driver;
            JdbcUser=postgres;
            JdbcPassword=12345
        </DataSourceInfo>
        <ProviderName>Mondrian</ProviderName>
        <ProviderType>MDP</ProviderType>
        <AuthenticationMode>Unauthenticated</AuthenticationMode>
        <Catalogs>
            <Catalog name="DWproject">
                <Definition>/WEB-INF/schema/Schema10.xml</Definition>
            </Catalog>
        </Catalogs>
    </DataSource>
</DataSources>
```

c) Result:

```
To get help about a specific shell command, type HELP <commmand>.
Refer to the MDX specification for more information about writing MDX queries.

Check out the tutorial here:
https://github.com/rpbouman/pash/wiki/Pash---The-Pentaho-Analysis-Shell



Hint:
Type SHOW CATALOGS to list the available catalogs.
Then, type USE [<catalog-name>] to select a particular catalog and enter MDX queries.
MDX> SHOW CATALOGS;

+--------------+-------------------------+-------+---------------+
| CATALOG_NAME |       DESCRIPTION       | ROLES | DATE_MODIFIED |
+--------------+-------------------------+-------+---------------+
| DWproject    | No description available |       | null          |
+--------------+-------------------------+-------+---------------+

1 rows in set.
Query took: 17 miliseconds; Rendering took: 0 miliseconds.
MDX> _
```




Welcome!
Pivot Table
Pie Chart
DWproject
Orders
Discount
Profit
Quantity
Sales
customer: (All)
customer: name
date: (All)
date: Year
location: (All)
location: Country
location: City
location: Region
location: State
location: Postal Code
location: Latitude
location: Longitude
product: (All)
product name
product: Category
product: Sub Category
shipment: (All)
shipment: Shipment Mode
shipment: Shipment Status
shipment: Days to Ship Actual
shipment: Days to Ship Scheduled
shipment: Date
Measures
Measures: Sales Quantity
Sort Options
Categories
product: product name
Sort Options
Columns
Rows
Slicer
Sales and Quantity per product
Canon imageCLASS 2200 Advanced Copier
Fellowes PB500 Electric Punch Plastic Comb Binding Machine with Manua
Cisco TelePresence System EX90 Videoconferencing Unit
HON 5400 Series Task Chairs for Big and Tall
GBC DocuBind TL300 Electric Binding System
GBC Ibimaster 500 Manual ProClick Binding System
Hewlett Packard LaserJet 3310 Copier
HP Designjet T520 Inkjet Large Format Printer - 24 Color
GBC DocuBind P400 Electric Binding System
High Speed Automatic Electric Letter Opener
Lexmark MX611dhe Monochrome Laser Printer
Martin Yale Chadless Opener Electric Letter Opener
Ibico EPK-21 Electric Binding System
Riverside Palais Royal Lawyers Bookcase, Royale Cherry Finish
3D Systems Cube Printer, 2nd Generation, Magenta
Samsung Galaxy Mega 6.3
Apple iPhone 5
Bretford Rectangular Conference Table Tops
Global Troy Executive Leather Low-Back Tilter
Canon PC1060 Personal Laser Copier
SAFCO Arco Folding Chair
Honeywell Enviracaire Portable HEPA Air Cleaner for 17 x 22 Room
Cubify CubeX 3D Printer Double Head Print
DMI Eclipse Executive Suite Bookcases
Tennsco 6- and 18-Compartment Lockers
Plantronics CS510 - Over-the-Head monaural Wireless Headset System
Sales
Quantity


Welcome!
Pivot Table
Pie Chart
DWproject
Orders
Discount
Profit
Quantity
Sales
customer: (All)
customer: name
date: (All)
date: Year
location: (All)
location: Country
location: City
location: Region
location: State
location: Postal Code
location: Latitude
location: Longitude
product: (All)
product name
product: Category
product: Sub Category
shipment: (All)
shipment: Shipment Mode
shipment: Shipment Status
shipment: Days to Ship Actual
shipment: Days to Ship Scheduled
shipment: Date
Columns
Measures: Sales Profit
Sort Options
Rows
customer: name
Sort Options
Slicer
Measures
customer
Sales
Profit
Aaron Bergman 887 129
Aaron Hawkins 1,744 364
Aaron Smayling 3,051 -254
Adam Bellavance 7,757 2,055
Adam Hart 3,251 283
Adam Shillingsburg 3,255 63
Adrian Barton 14,476 5,445
Adrian Hane 1,736 -2
Adrian Shami 58 22
Aimee Bixby 969 314
Alan Barnes 1,114 220
Alan Dominguez 6,106 1,868
Alan Haines 1,588 -378
Alan Hwang 4,805 1,308
Alan Schoenberger 4,262 721