README

CS562 - Team Guardians - Mehul Gupta & Sanjana Brid

Table of Contents

How to execute our program?

1. Import the java project in eclipse.

   (Follow the onscreen instruction and browse directly from USB drive.)

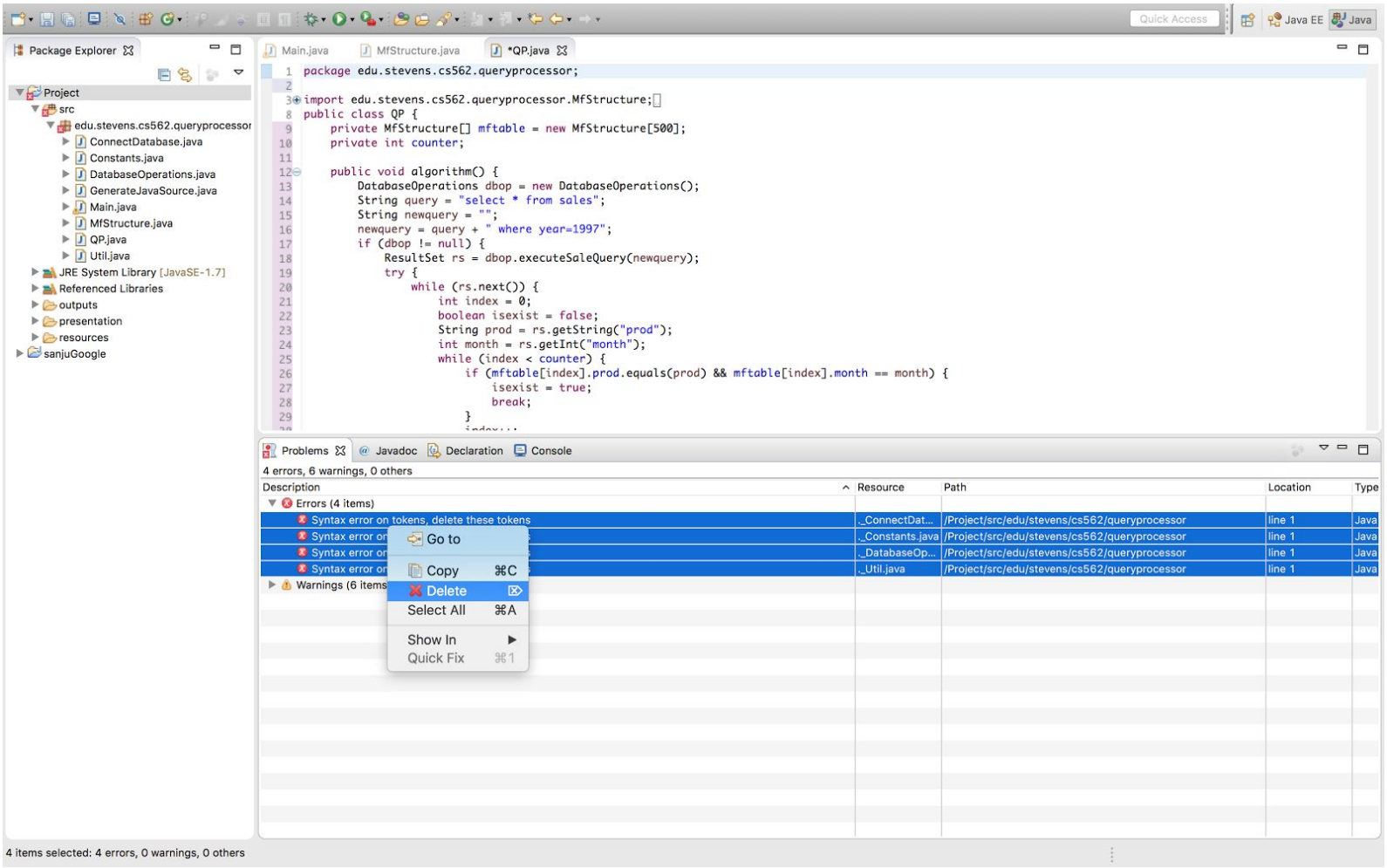
   If any error regarding “.classpath” occurs while browsing the project from the USB Drive, then go to the Project directory “/Guardians “ in your file explorer and search for file .classpath. (The file can be hidden) Open properties of .classpath file and uncheck its hidden attribute. Now try again to import the project.

2. Adding the postgres jar to the project

   In Eclipse, Right Click Project in Package Explorer view > Select Build Path > Select Configure Build Path > Under Libraries Menu - Select Add External JARs > Select the postgresql-8.3-604.jdbc4.jar > Click OK.

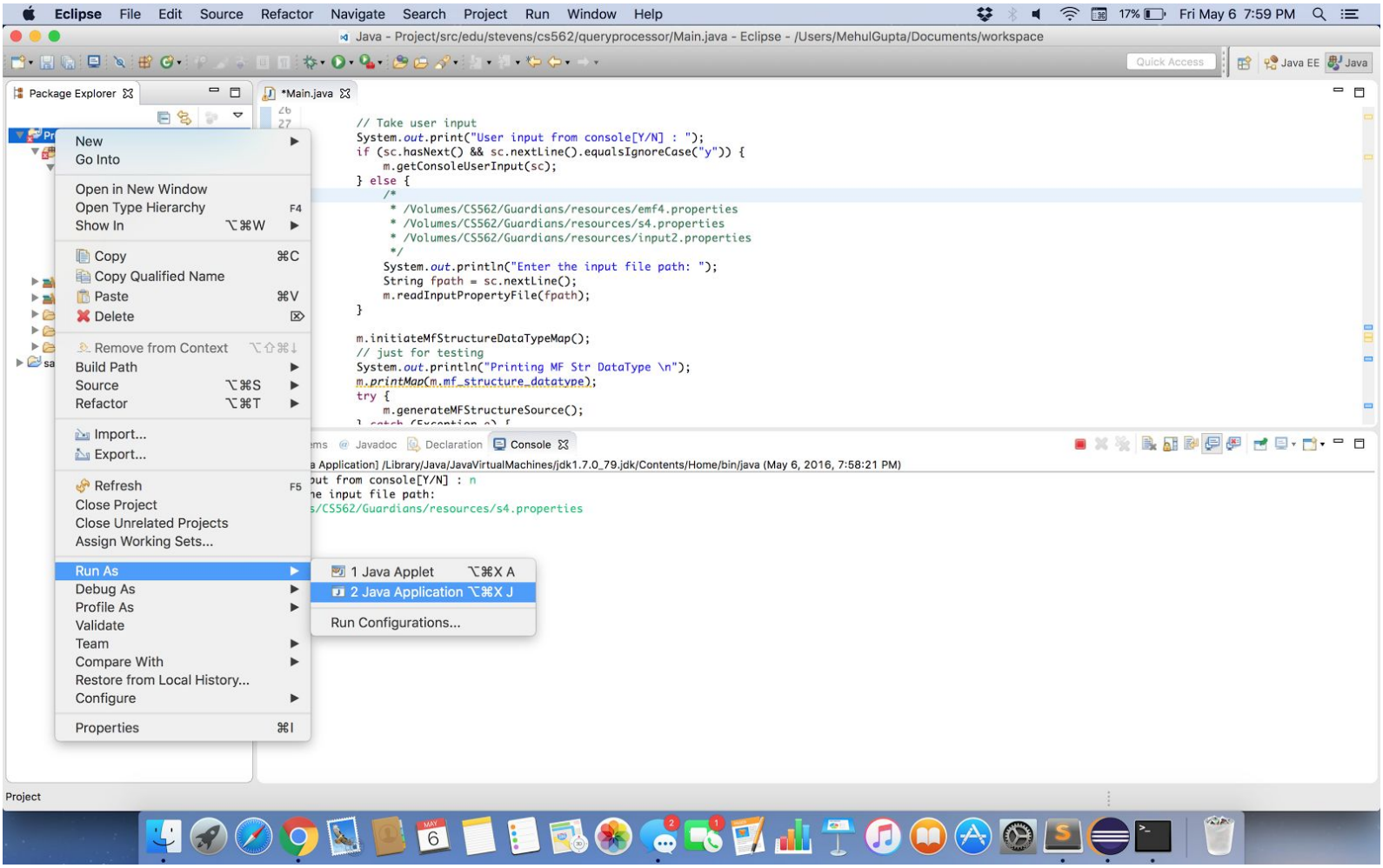
3. Delete unwanted hidden file errors

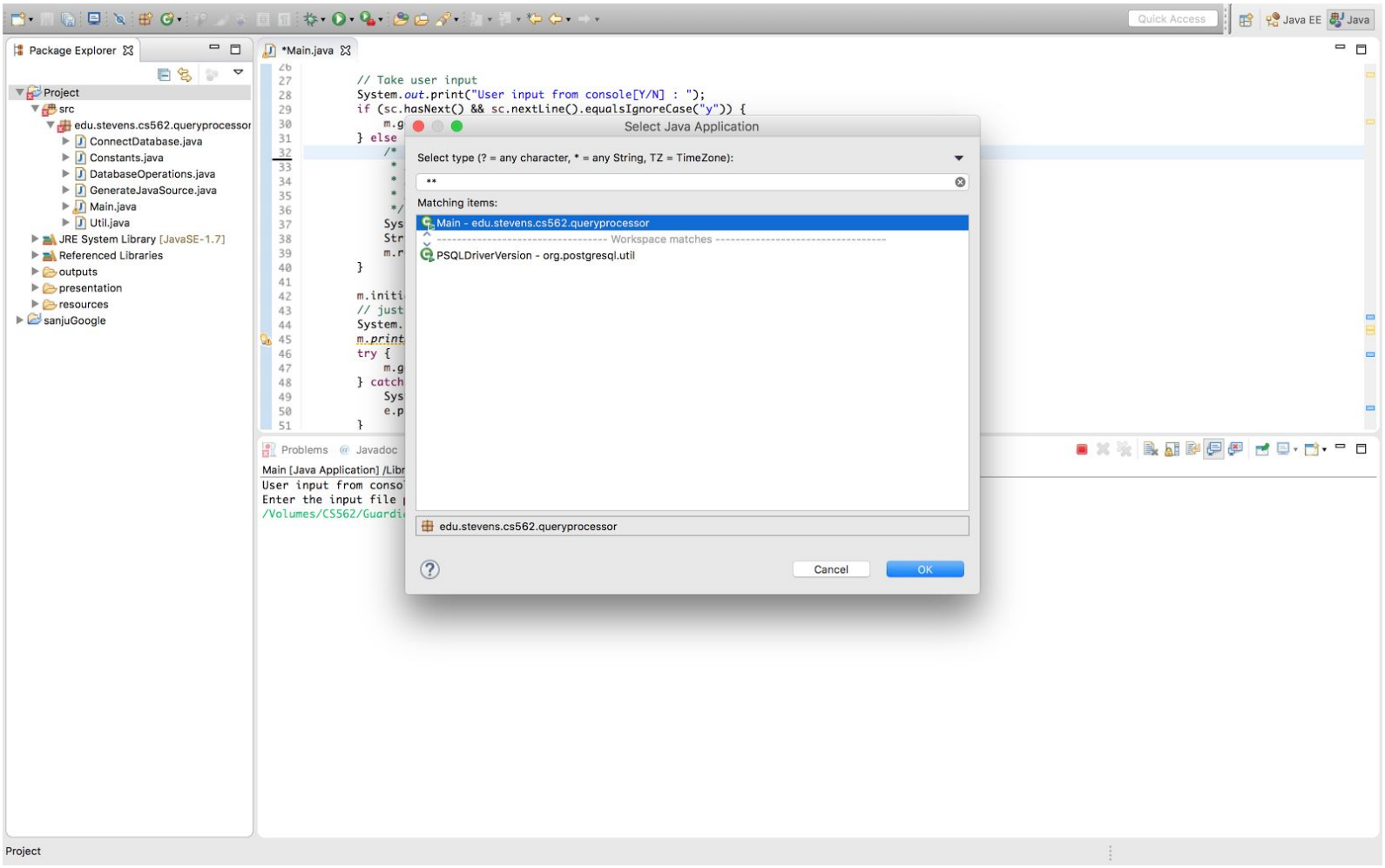
   Go to Problems Tab at the bottom and delete all the errors. (See the below image for reference)

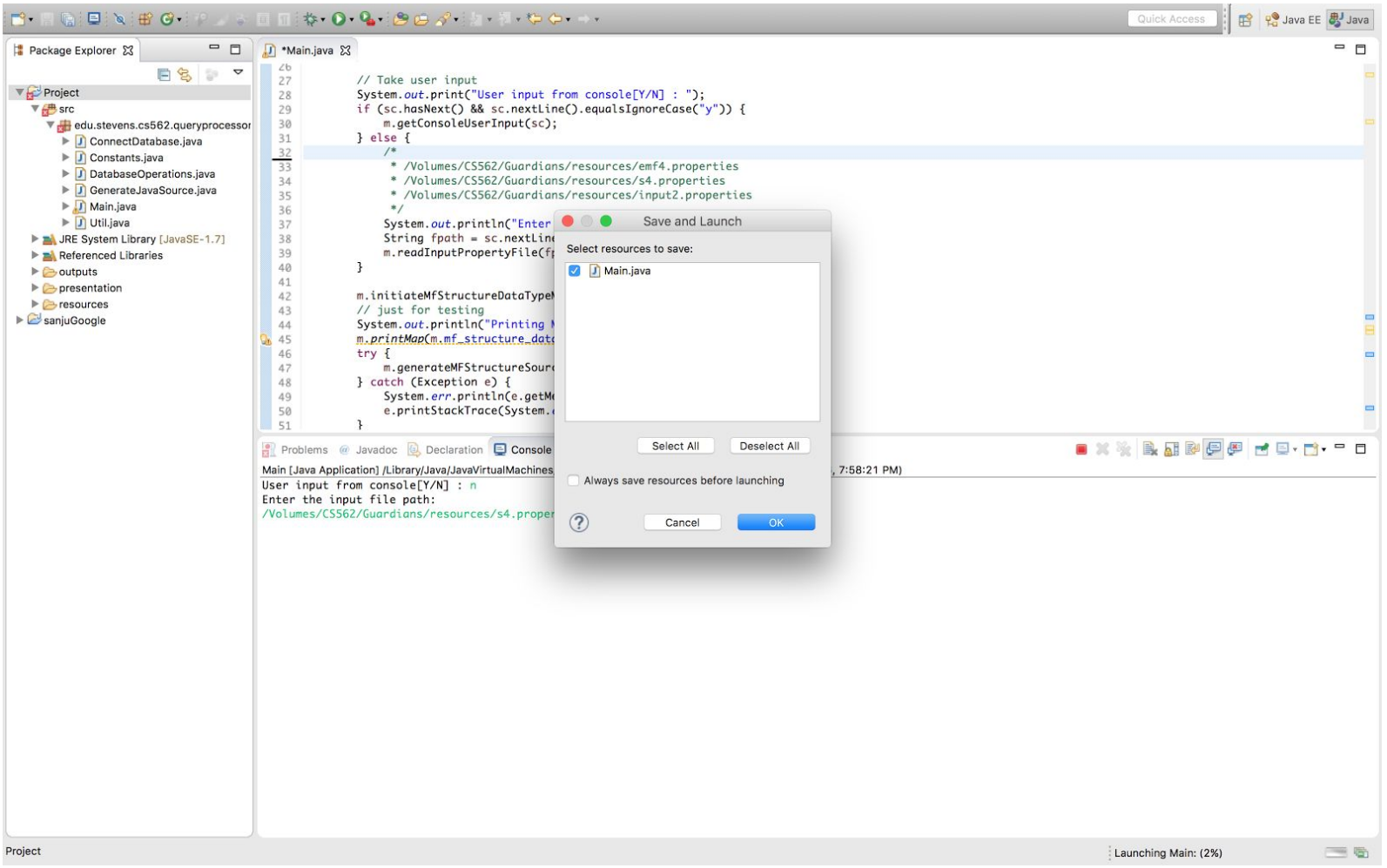


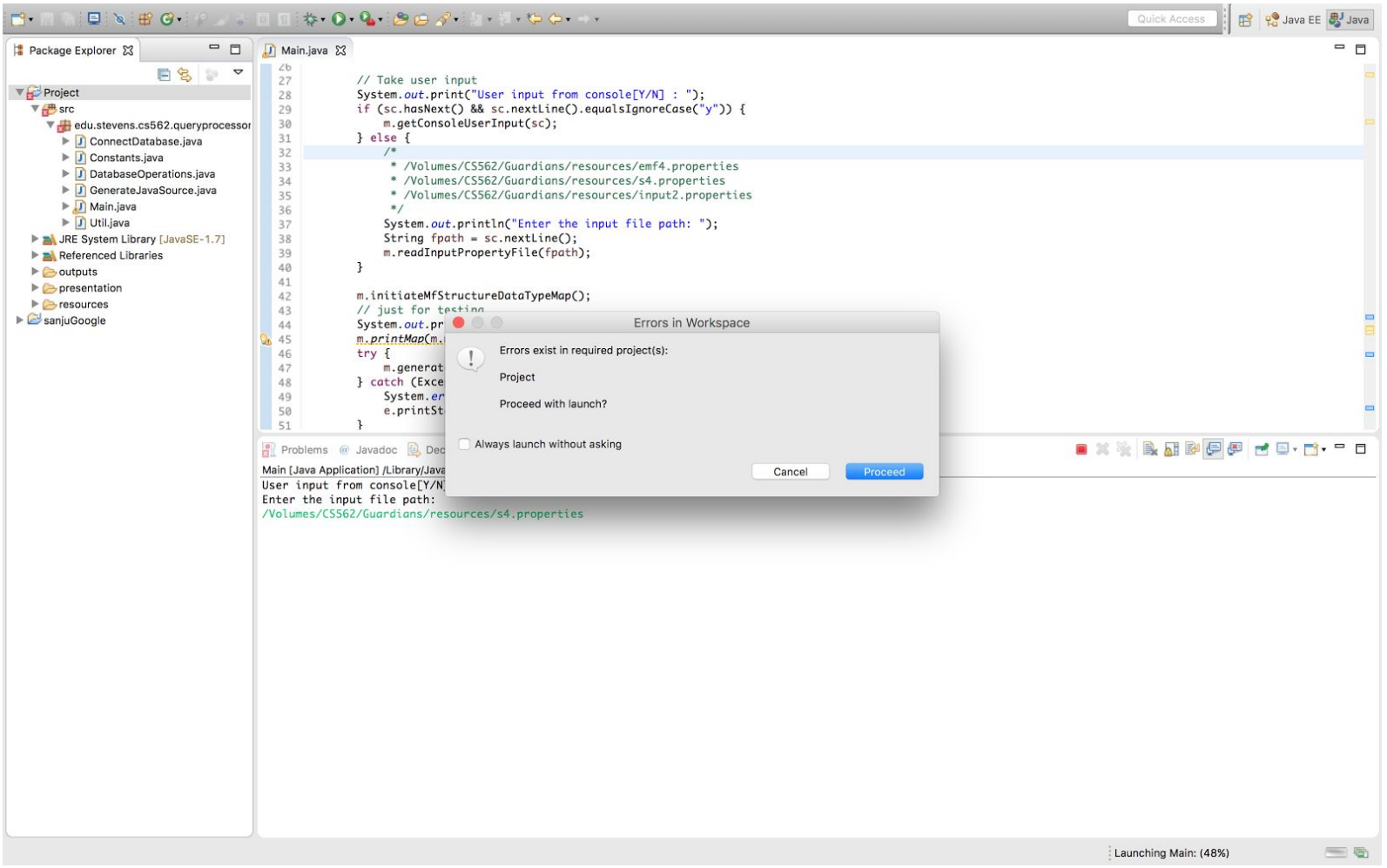
4. Executing the Project

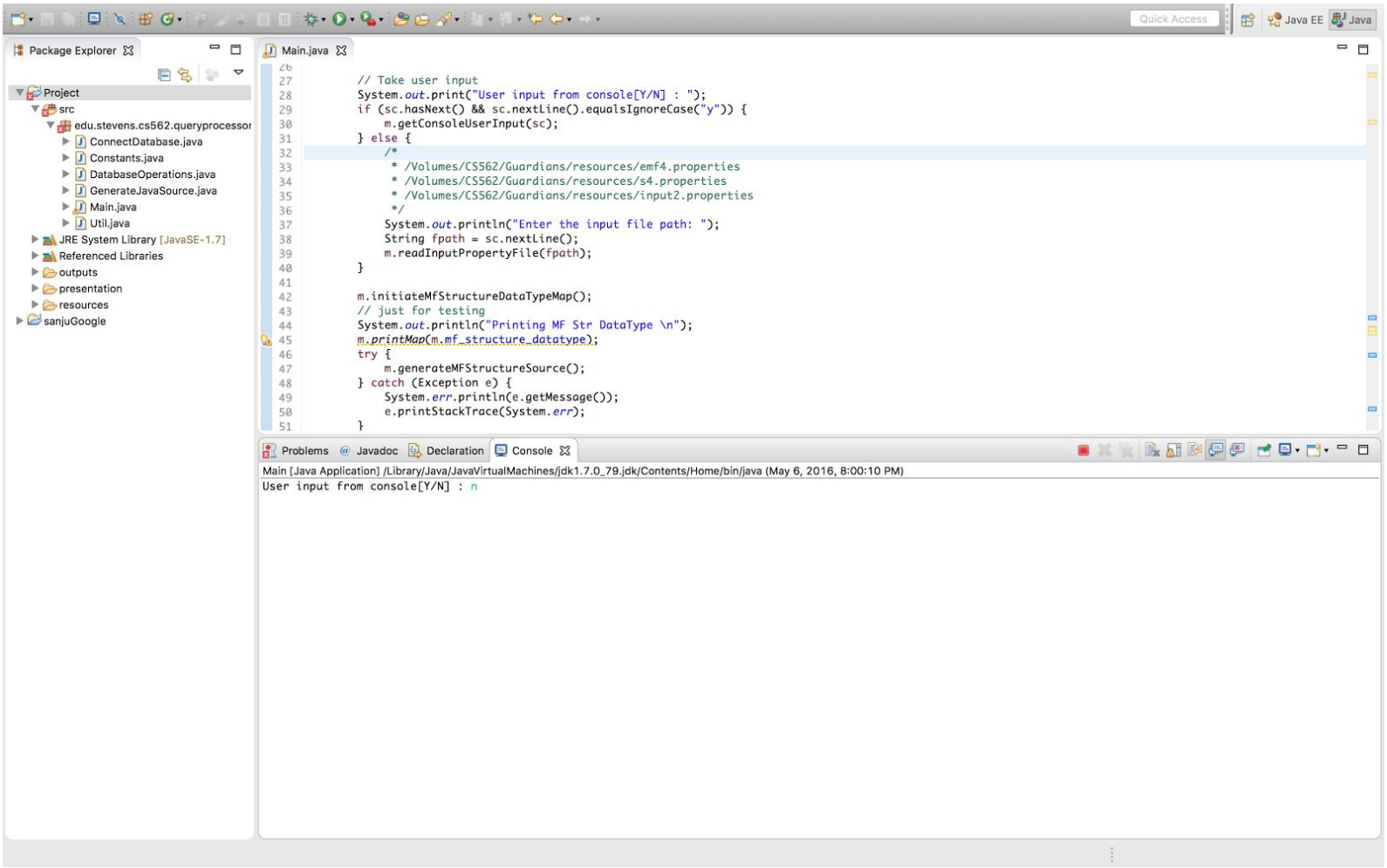
Right click Project in Package Explorer view > Select Run As > Select JAVA Application > Select Main - edu.stevens.cs562.queryprocessor from the Select Java Application popup window.
(Project is now Running) - Refer the below images


Eclipse File Edit Source Refactor Navigate Search Project Run Window Help
Fri May 6 7:59 PM
Java - Project/src/edu/stevens/cs562/queryprocessor/Main.java - Eclipse - /Users/MehulGupta/Documents/workspace
Package Explorer
*Main.java
// Take user input
System.out.print("User input from console[Y/N] : ");
if (sc.hasNext() && sc.nextLine().equalsIgnoreCase("y")) {
m.getConsoleUserInput(sc);
} else {
/*
* /Volumes/CS562/Guardians/resources/emf4.properties
* /Volumes/CS562/Guardians/resources/s4.properties
* /Volumes/CS562/Guardians/resources/input2.properties
*/
System.out.println("Enter the input file path: ");
String fpath = sc.nextLine();
m.readInputPropertyFile(fpath);
}
m.initiateMfStructureDataTypeMap();
// just for testing
System.out.println("Printing MF Str DataType \n");
m.printMap(m.mf_structure_datatype);
try {
m.generateMFStructureSource();
Javadoc Declaration Console
a Application] /Library/Java/JavaVirtualMachines/jdk1.7.0_79.jdk/Contents/Home/bin/java (May 6, 2016, 7:58:21 PM)
ut from console[Y/N] : n
e input file path:
s/CS562/Guardians/resources/s4.properties
New
Go Into
Open in New Window
Open Type Hierarchy F4
Show In ⌥⌘W
Copy ⌘C
Copy Qualified Name
Paste ⌘V
Delete
Remove from Context ⌥⇧⌘↓
Build Path
Source ⌥⌘S
Refactor ⌥⌘T
Import...
Export...
Refresh F5
Close Project
Close Unrelated Projects
Assign Working Sets...
Run As
Debug As
Profile As
Validate
Team
Compare With
Restore from Local History...
Configure
Properties ⌘I
1 Java Applet ⌥⌘X A
2 Java Application ⌥⌘X J
Run Configurations...
Project


Quick Access
Java EE
Java
Package Explorer
Project
src
edu.stevens.cs562.queryprocessor
ConnectDatabase.java
Constants.java
DatabaseOperations.java
GenerateJavaSource.java
Main.java
Util.java
JRE System Library [JavaSE-1.7]
Referenced Libraries
outputs
presentation
resources
sanjuGoogle
*Main.java
// Take user input
System.out.print("User input from console[Y/N] : ");
if (sc.hasNext() && sc.nextLine().equalsIgnoreCase("y")) {
} else
} catch
}
Select Java Application
Select type (? = any character, * = any String, TZ = TimeZone):
**
Matching items:
Main - edu.stevens.cs562.queryprocessor
Workspace matches
PSQLDriverVersion - org.postgresql.util
edu.stevens.cs562.queryprocessor
Cancel
OK
Problems
Javadoc
Main [Java Application] /Libr
User input from conso
Enter the input file
/Volumes/CS562/Guardi
Project


Package Explorer
Project
src
edu.stevens.cs562.queryprocessor
ConnectDatabase.java
Constants.java
DatabaseOperations.java
GenerateJavaSource.java
Main.java
Util.java
JRE System Library [JavaSE-1.7]
Referenced Libraries
outputs
presentation
resources
sanjuGoogle
*Main.java
// Take user input
System.out.print("User input from console[Y/N] : ");
if (sc.hasNext() && sc.nextLine().equalsIgnoreCase("y")) {
m.getConsoleUserInput(sc);
} else {
/*
* /Volumes/CS562/Guardians/resources/emf4.properties
* /Volumes/CS562/Guardians/resources/s4.properties
* /Volumes/CS562/Guardians/resources/input2.properties
*/
m.readInputPropertyFile(
}
// just for testing
m.printMap(
try {
m.generateMFStructureSour
} catch (Exception e) {
e.printStackTrace(System.
}
Save and Launch
Select resources to save:
Main.java
Select All
Deselect All
Always save resources before launching
Cancel
OK
Problems
Javadoc
Declaration
Console
Main [Java Application] /Library/Java/JavaVirtualMachines
7:58:21 PM)
User input from console[Y/N] : n
Enter the input file path:
/Volumes/CS562/Guardians/resources/s4.proper
Project
Launching Main: (2%)

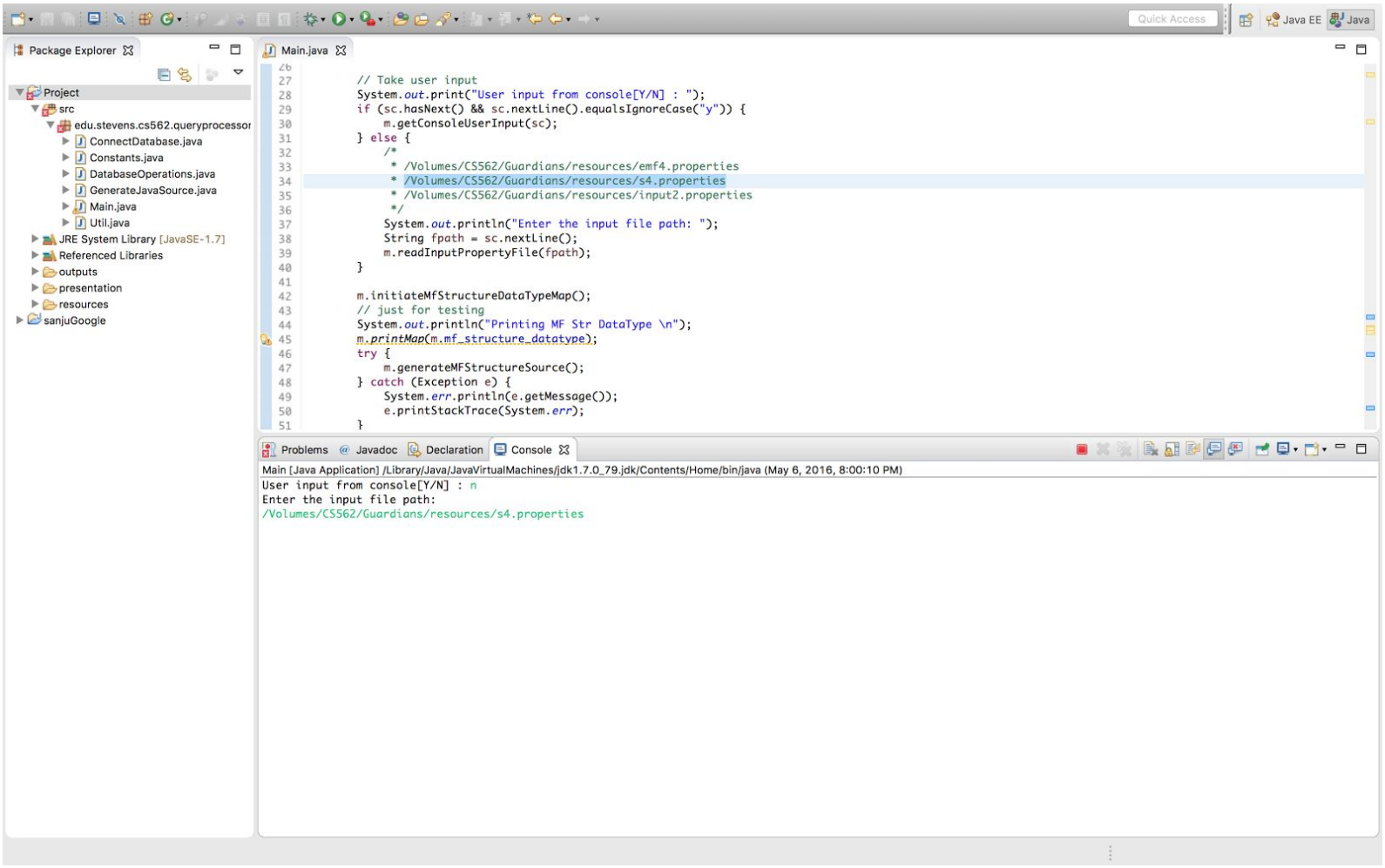

Quick Access
Java EE
Java
Package Explorer
Project
src
edu.stevens.cs562.queryprocessor
ConnectDatabase.java
Constants.java
DatabaseOperations.java
GenerateJavaSource.java
Main.java
Util.java
JRE System Library [JavaSE-1.7]
Referenced Libraries
outputs
presentation
resources
sanjuGoogle
Main.java
// Take user input
System.out.print("User input from console[Y/N] : ");
if (sc.hasNext() && sc.nextLine().equalsIgnoreCase("y")) {
m.getConsoleUserInput(sc);
} else {
/*
* /Volumes/CS562/Guardians/resources/emf4.properties
* /Volumes/CS562/Guardians/resources/s4.properties
* /Volumes/CS562/Guardians/resources/input2.properties
*/
System.out.println("Enter the input file path: ");
String fpath = sc.nextLine();
m.readInputPropertyFile(fpath);
}
m.initiateMfStructureDataTypeMap();
// just for testing
try {
} catch (Exce
e.printSt
}
Errors in Workspace
Errors exist in required project(s):
Project
Proceed with launch?
Always launch without asking
Cancel
Proceed
Problems
Javadoc
Main [Java Application] /Library/Java
User input from console[Y/N
Enter the input file path:
/Volumes/CS562/Guardians/resources/s4.properties
Launching Main: (48%)



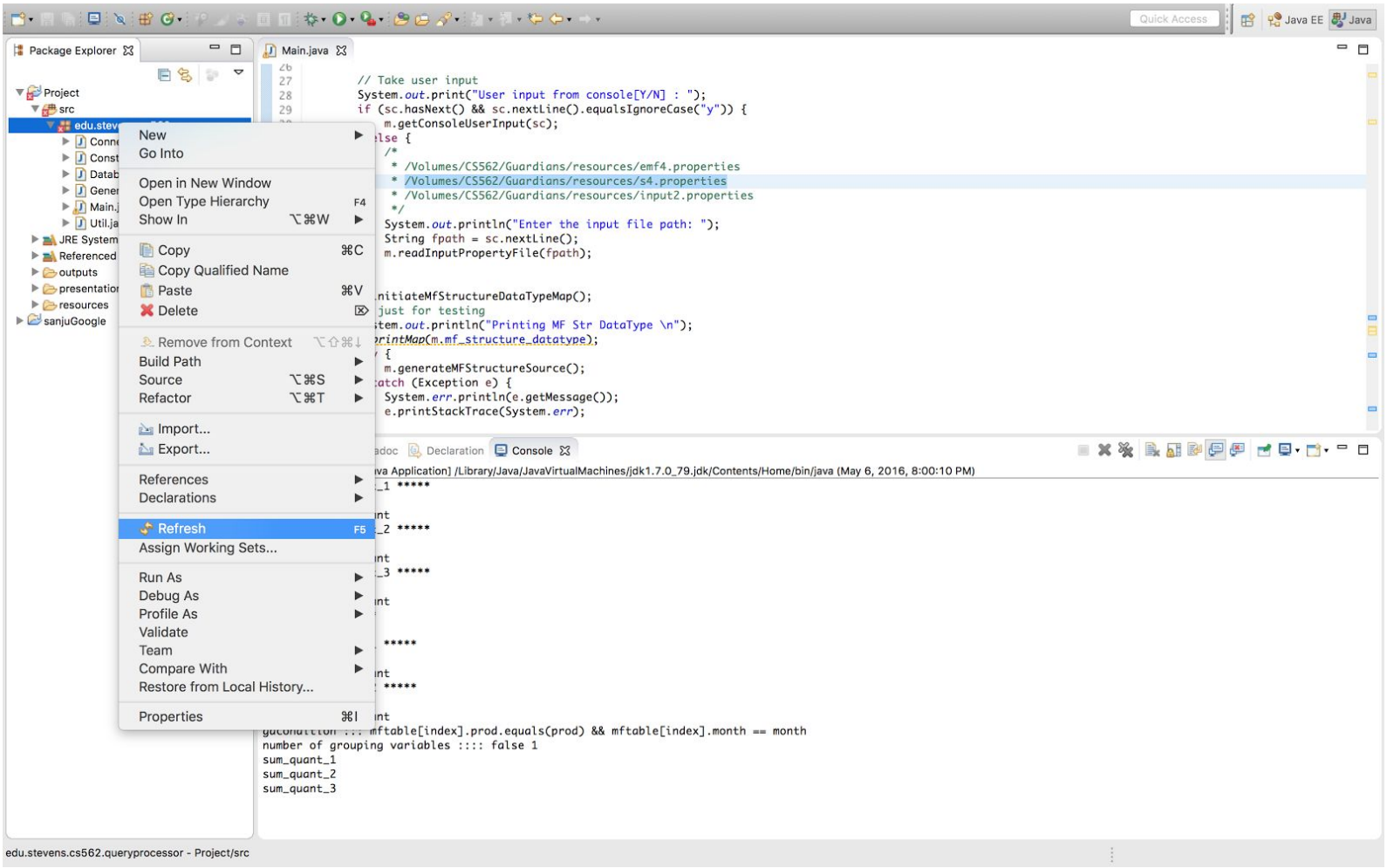
Project is now running.

Program Execution using File

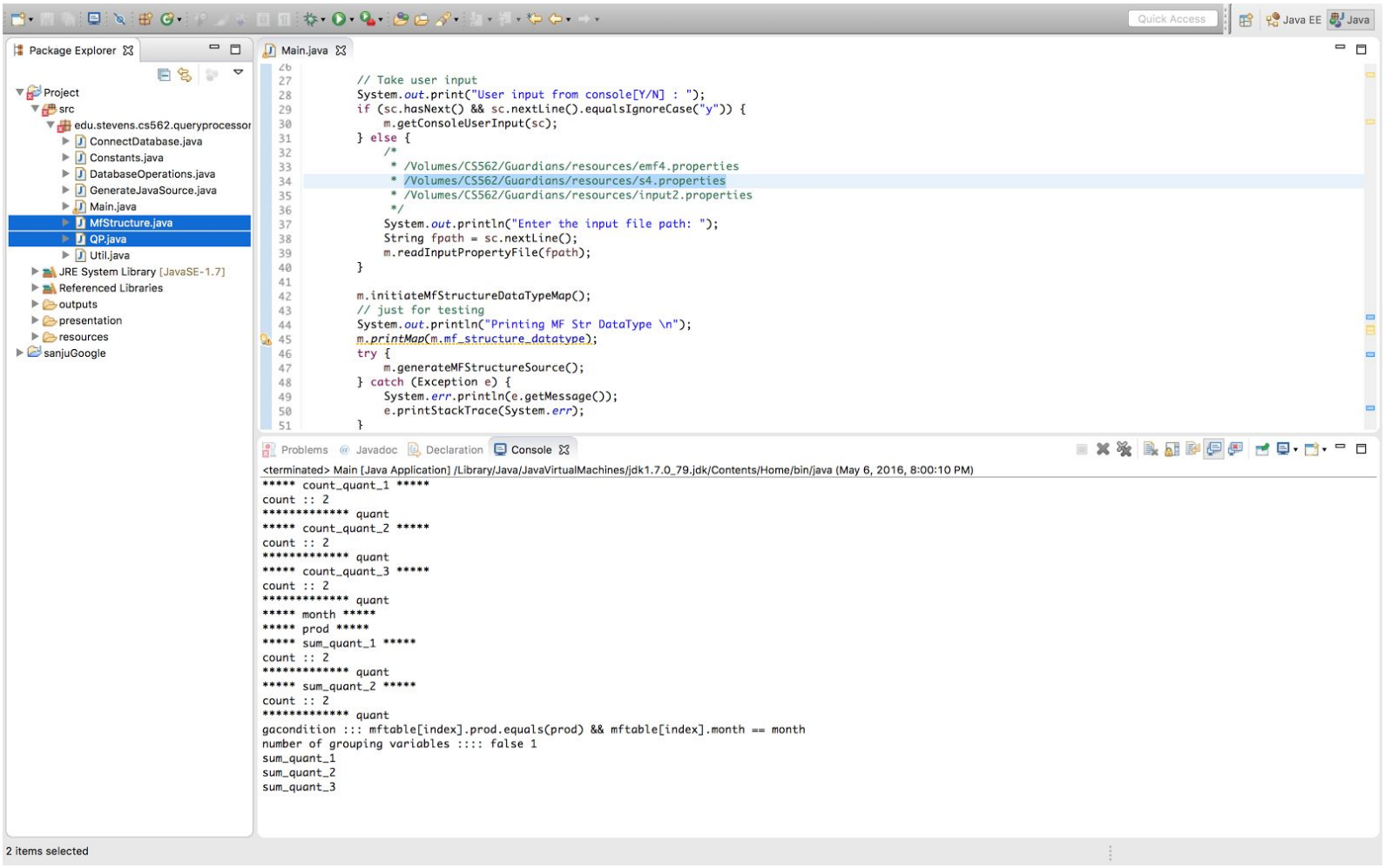
“**User Input from console [Y/N] :** “
Press “**N**” , to input from a file
Press “**Y**” , otherwise



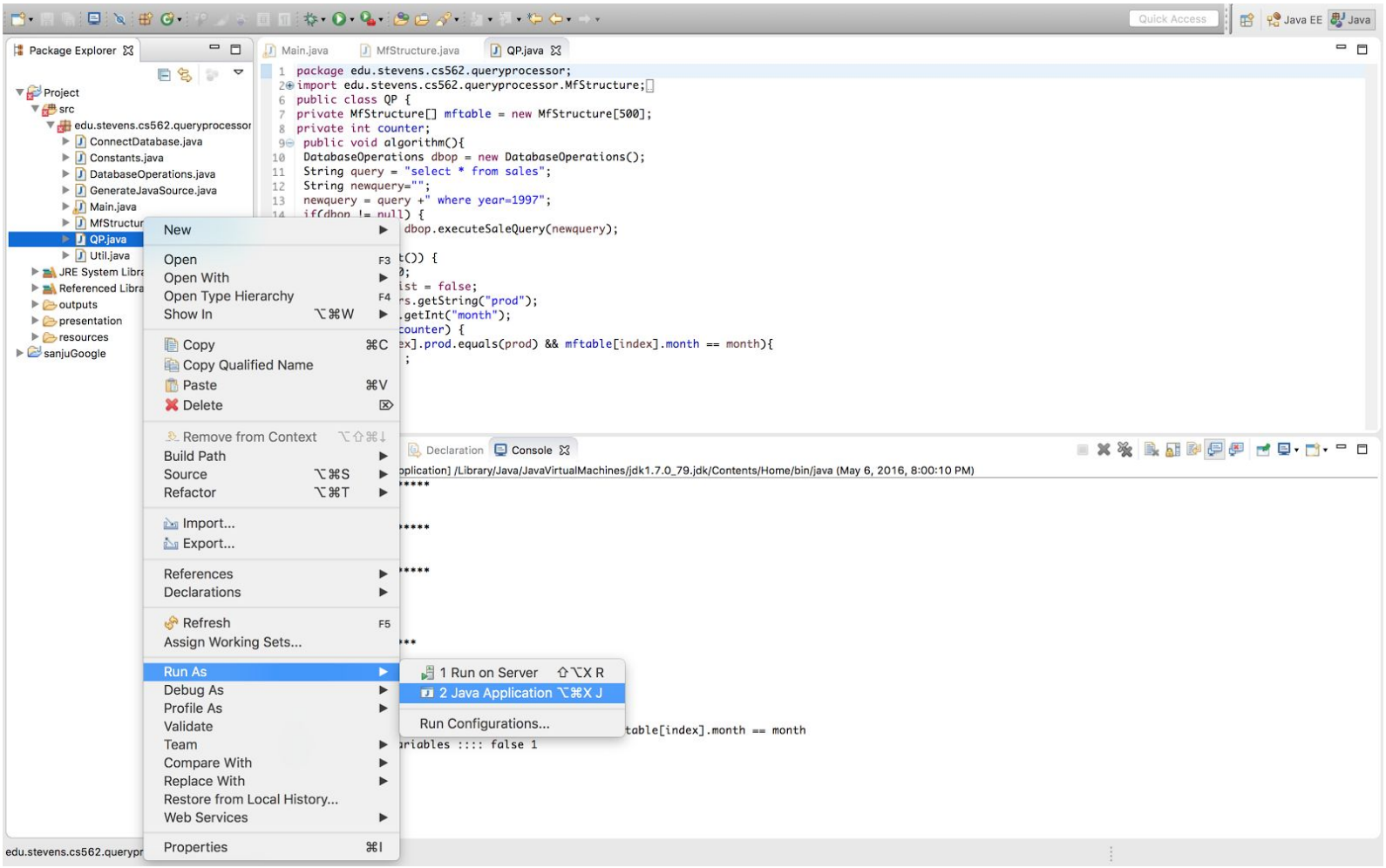
“**Enter the input file path: “**
Type **“ /Volumes/CS562/Guardians/resources/s4.properties”** and press **enter/return** key.
Input files are stored in the /Guardians/resources directory.



Refresh the Project from the Package explorer to see the newly generated java files.



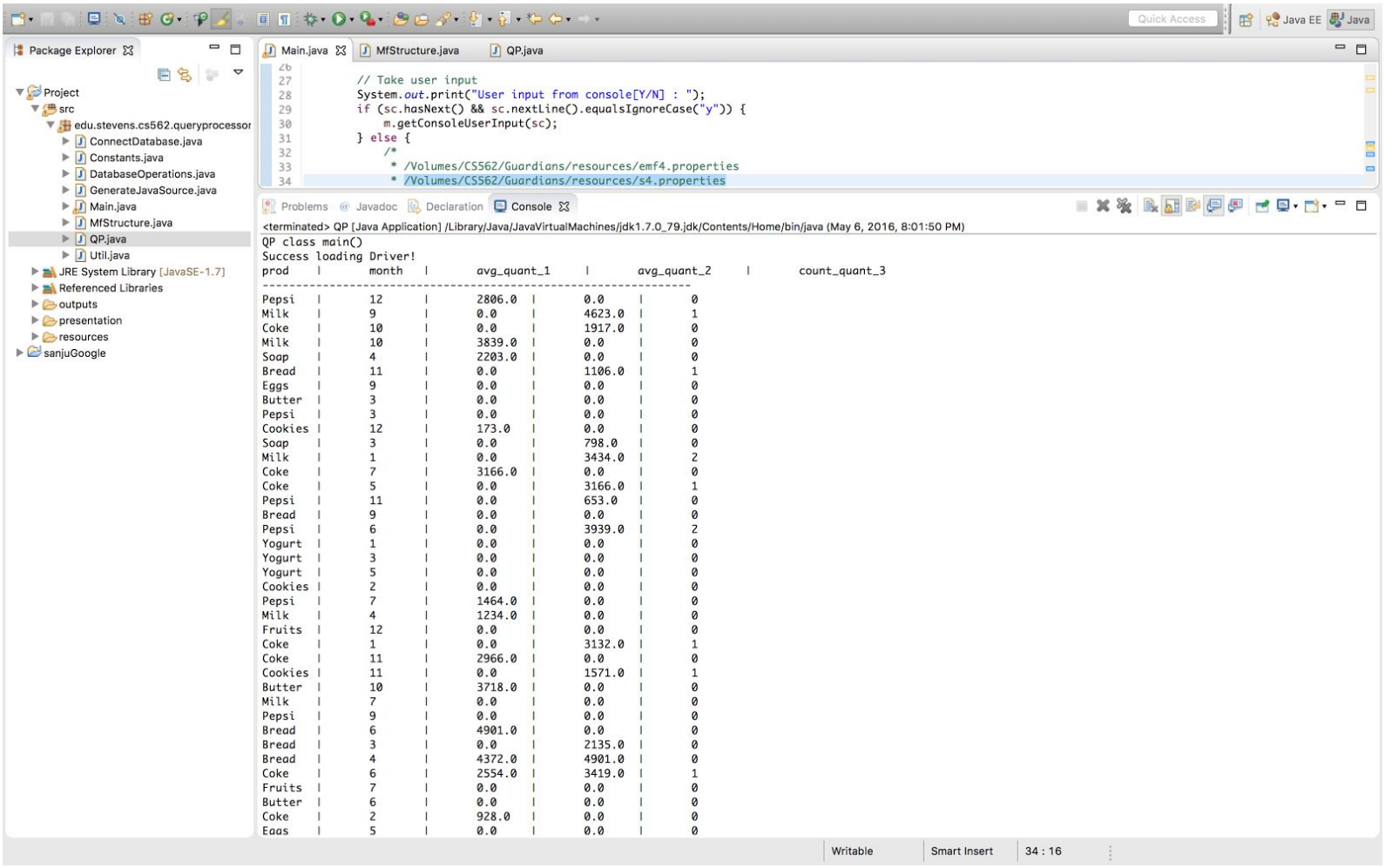
Newly generated files are 1) **MfStructure.java** and 2) **QP.java**



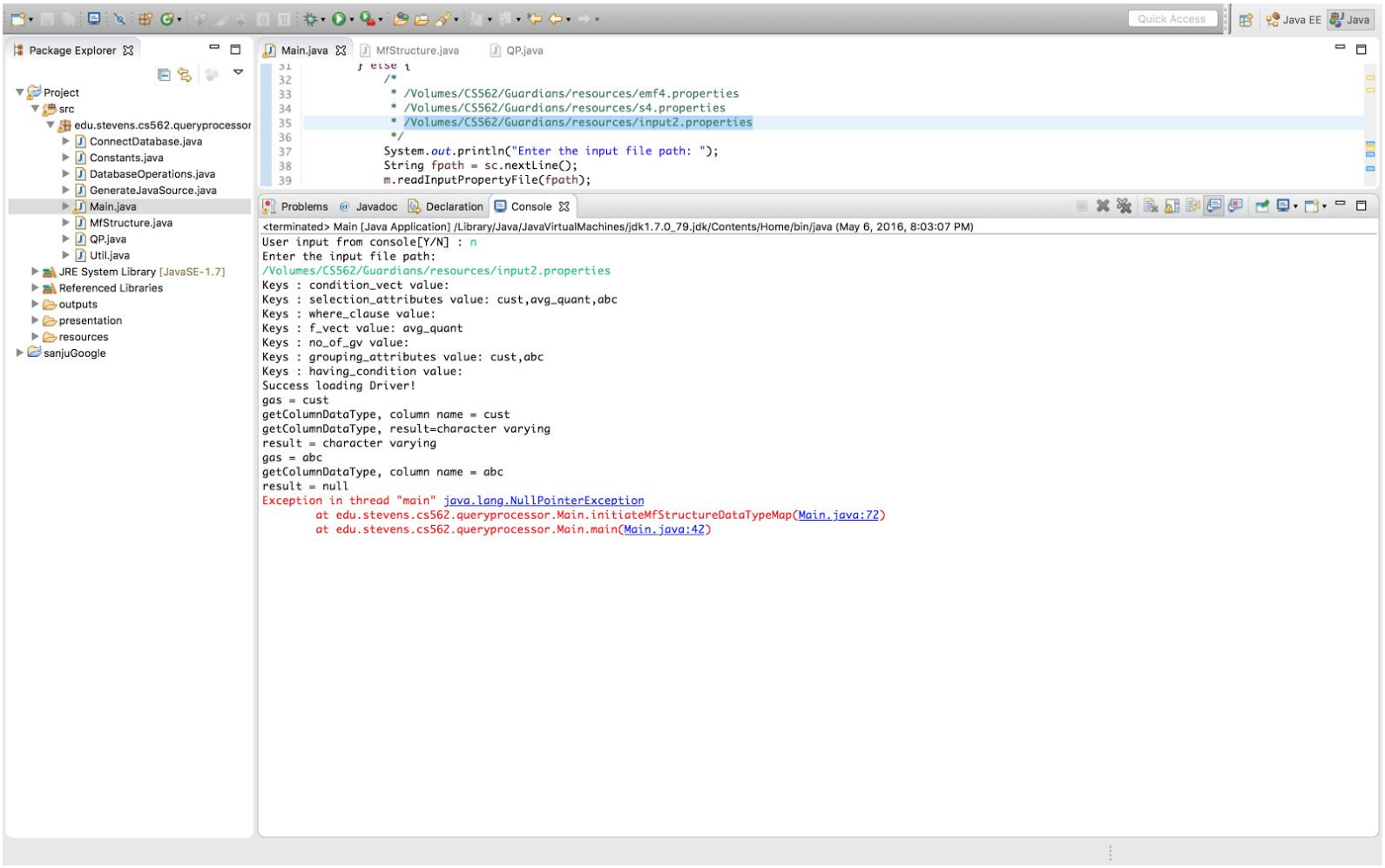
To execute the newly generated java files

Firstly, Double click on **MfStructure.java**

Secondly, Right Click **QP.java** > select Run As > select Java Application



On successful execution, the output is displayed in the console.
Above image displays the output for Query : For each prod, count for each month of 2008, the sales that were between previous and following months average sale. - Dependent Aggregation

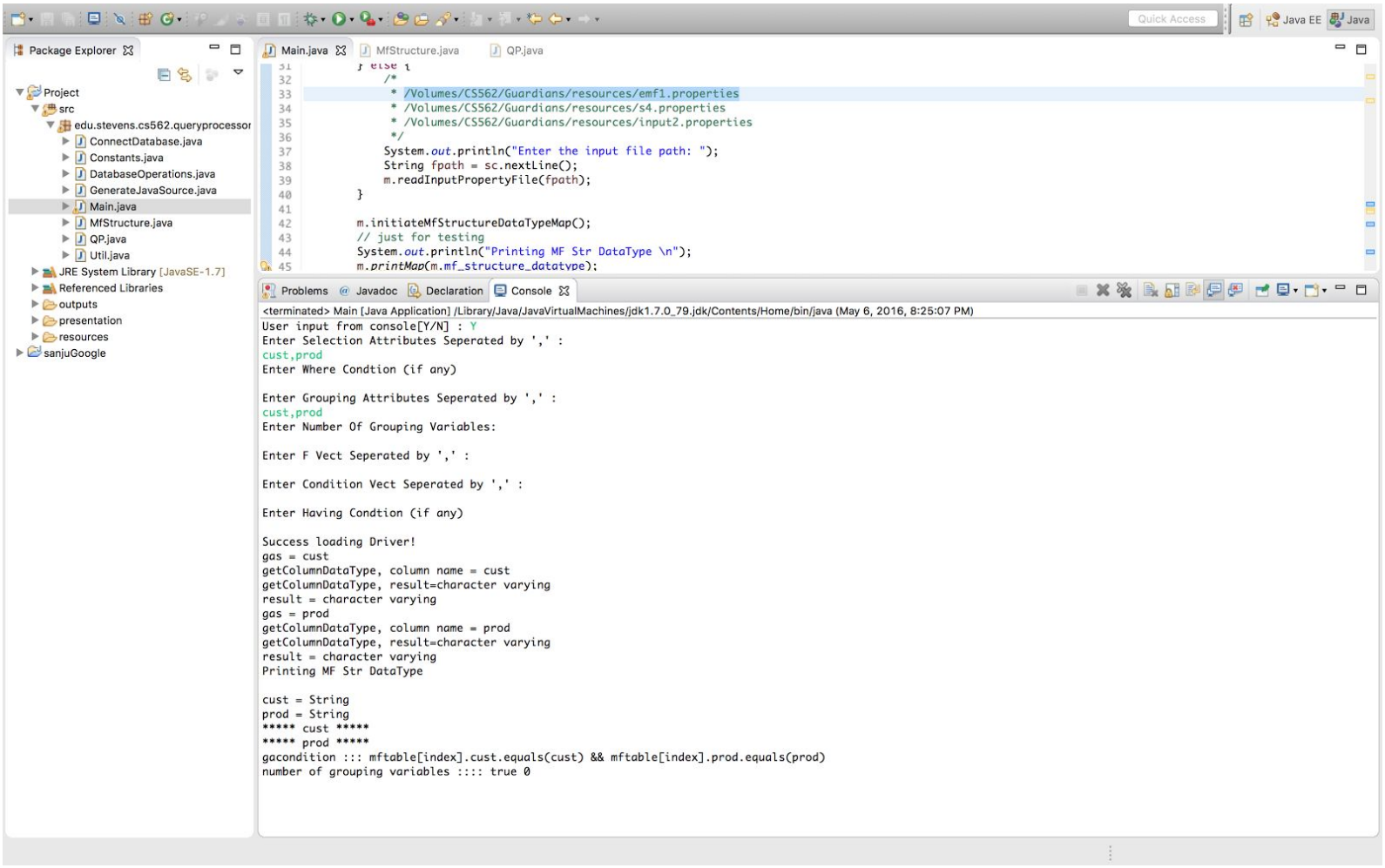


What If the column does not exist in the database?

(As shown in the above image, refer input2.properties for more information)
"**Enter the input file path:** "
Type **" /Volumes/CS562/Guardians/resources/input2.properties"** and press **enter/return** key.
The program execution halts due to error of Missing Column "abc".

Program Execution using Console

User Input from console [Y/N] : **Y**
Then follow the onscreen instructions. (refer the below image)



SAMPLE QUERIES

EMF1. For 2008 , show for each product the total of January, the total of February and the total of March sales (in three columns).

SQL:

```
create view v1 as
select * from sales
where year=2008;
```

```
create view jan as
select v1.prod as prod,sum(v1.quant) as jquant from v1
where v1.month=1
group by v1.prod;


create view feb as
select v1.prod as prod,sum(v1.quant) as fquant from v1
where v1.month=2
group by v1.prod;

create view mar as
select v1.prod as prod,sum(v1.quant) as mquant from v1
where v1.month=3
group by v1.prod;


select jan.prod, jan.jquant, feb.fquant,mar.mquant from jan,feb,mar
where jan.prod=feb.prod and feb.prod=mar.prod;
```

EMF:

```
select prod,sum(x.quant),sum(y.quant),sum(z.quant)
from sales
where year=2008
group by prod; X,Y,Z
such that X.prod=prod and X.month=1,
          Y.prod=prod and Y.month=2,
          Z.prod=prod and Z.month=3
```

Phi Operators:

```
selection_attributes = prod,mftable[index].sum_quant_1,mftable[index].sum_quant_2,mftable[index].sum_quant_3
where_clause = year\=2008
grouping_attributes = prod
no_of_gv = 3
f_vect = sum_quant_1,sum_quant_2,sum_quant_3
condition_vect = mftable[index].prod.equals(prod) && mftable[index].month == 1, mftable[index].prod.equals(prod) &&
mftable[index].month == 2, mftable[index].prod.equals(prod) && mftable[index].month == 3
```

having_condition =

Query Output:

```
QP class main()
Success loading Driver!
prod    |       mftable[index].sum_quant_1  |       mftable[index].sum_quant_2  |       mftable[index].sum_quant_3
-----------------------------------------------------------------
Pepsi   |       4412    |       0       |       4484
Milk    |       3207    |       3434    |       1234
Coke    |       928     |       3132    |       0
Soap    |       0       |       0       |       2203
Bread   |       0       |       0       |       4372
Eggs    |       0       |       0       |       0
Butter  |       0       |       0       |       4018
Cookies         |       0       |       2523    |       0
Yogurt |       2142    |       0       |       1660
Fruits  |       0       |       0       |       1469
```

EMF2. For each product and sales of 2008, show the product's average sale before and after each month of 2008.

SQL:

```
with B1 as
(select x.prod as prod, x.month as month, avg(y.quant) as avgx
from Sales x, Sales y
where x.prod=y.prod and
x.month > y.month and x.year=2008 and y.year=2008
group by x.prod, x.month
),
 B2 as
(select x.prod as prod ,x.month as month, avg(y.quant)as avgy
from Sales x, Sales y
where x.prod=y.prod and
```

```
x.month < y.month and x.year=2008 and y.year=2008
group by x.prod, x.month)

select B1.prod, B1.month, avgx as before, avgy as after
from B1, B2
where B1.prod=B2.prod and
B1.month=B2.month
```

EMF:

```
select  prod,month,avg(X.quant),avg(Y.quant)
from sales
where year=2008
group by prod,month:X,Y
such that X.prod=prod and X.month<month,
          y.prod=prod and Y.month>month
```

Phi Operator:

```
selection_attributes = prod,month,avg_quant_1,avg_quant_2
where_clause = year\=2008
grouping_attributes = prod,month
no_of_gv = 2
f_vect = avg_quant_1,avg_quant_2
condition_vect = mftable[index].prod.equals(prod) && month < mftable[index].month, mftable[index].prod.equals(prod) &&
month > mftable[index].month
having_condition =
```

Query Output:

```
QP class main()
Success loading Driver!
prod    |       month |       avg_quant_1  |        avg_quant_2
-----------------------------------------------------------------
Pepsi   |       12      |       3011.0 |       0.0
Milk    |       9       |       1859.0 |       4623.0
Coke    |       10      |       2769.0 |       1917.0
Milk    |       10      |       2142.0 |       0.0
Soap    |       4       |       2203.0 |       0.0
Bread   |       11      |       3150.0 |       1106.0
```

Eggs	9	1114.0	1312.0	
Butter	3	0.0	3274.0	
Pepsi	3	4412.0	2733.0	
Cookies	12	1348.0	0.0	
Soap	3	0.0	798.0	
Milk	1	0.0	3005.0	
Coke	7	2445.0	2616.0	
Coke	5	2030.0	2975.0	
Pepsi	11	3062.0	653.0	
Bread	9	3441.0	945.0	
Pepsi	6	2965.0	3156.0	
Yogurt	1	0.0	3112.0	
Yogurt	3	2142.0	4564.0	
Yogurt	5	1901.0	0.0	
Cookies	2	0.0	1105.0	
Pepsi	7	2364.0	2765.0	
Milk	4	1968.0	3278.0	
Fruits	12	1862.0	0.0	
Coke	1	0.0	2942.0	
Coke	11	2818.0	0.0	
Cookies	11	2523.0	1571.0	
Butter	10	2852.0	0.0	
Milk	7	2226.0	4362.0	
Pepsi	9	2814.0	2088.0	
Bread	6	3802.0	1133.0	
Bread	3	0.0	2171.0	
Bread	4	4372.0	2176.0	
Coke	6	2204.0	2937.0	
Fruits	7	2262.0	3939.0	
Butter	6	2009.0	3285.0	
Coke	2	928.0	2915.0	
Eggs	5	0.0	1138.0	
Fruits	3	0.0	2685.0	
Bread	12	2579.0	0.0	
Eggs	12	1039.0	0.0	
Bread	5	3253.0	1722.0	
Butter	9	2636.0	2852.0	
Milk	2	1603.0	2933.0	
Pepsi	1	0.0	2635.0	
Fruits	4	1469.0	2500.0	
Milk	3	2213.0	3273.0	

EMF3. For each product, count for each month of 2008, how many sales of the previous and how many sales of the following month had quantity greater than that month's average sale. (trends)

SQL:

```
with B1 as
(select x.prod as prod, x.month as month, count(y.quant) as countx
from Sales x, Sales y
where x.prod=y.prod and
x.month =y.month+1 and x.year=2008 and y.year=2008
group by x.prod, x.month
),
 B2 as
(select x.prod as prod ,x.month as month, count(y.quant)as county
from Sales x, Sales y
where x.prod=y.prod and
x.month =y.month-1 and x.year=2008 and y.year=2008
group by x.prod, x.month)

select B1.prod, B1.month, countx as before, county as after
from B1, B2
where B1.prod=B2.prod and
B1.month=B2.month
```

EMF:

```
select prod,month,count(X.quant),count(Y.quant)
from sales
where year=2008
group by prod,month;X,Y
such that X.prod=prod and X.month=month-1,
            Y.prod=prod and Y.month=month+1,
```

Phi Operator:

```
selection_attributes = prod,month,count_quant_1,count_quant_2
where_clause = year\=2008
grouping_attributes = prod,month
no_of_gv = 2
```

f_vect = count_quant_1,count_quant_2
condition_vect = mftable[index].prod.equals(prod) && month ==mftable[index].month-1, mftable[index].prod.equals(prod) && month ==mftable[index].month+1
having_condition =

Query Output:

QP class main()
Success loading Driver!

prod	month	count_quant_1	count_quant_2
Pepsi	12	2	0
Milk	9	0	2
Coke	10	0	1
Milk	10	1	0
Soap	4	1	0
Bread	11	0	1
Eggs	9	0	0
Butter	3	0	0
Pepsi	3	0	0
Cookies	12	1	0
Soap	3	0	2
Milk	1	0	1
Coke	7	1	0
Coke	5	0	1
Pepsi	11	0	1
Bread	9	0	0
Pepsi	6	0	2
Yogurt	1	0	0
Yogurt	3	0	0
Yogurt	5	0	0
Cookies	2	0	0
Pepsi	7	2	0
Milk	4	1	0
Fruits	12	0	0
Coke	1	0	1
Coke	11	2	0
Cookies	11	0	2
Butter	10	1	0
Milk	7	0	0
Pepsi	9	0	0
Bread	6	1	0
Bread	3	0	1

Bread	4	1	1
Coke	6	1	2
Fruits	7	0	0
Butter	6	0	0
Coke	2	1	0
Eggs	5	0	0
Fruits	3	0	1
Bread	12	2	0
Eggs	12	0	0
Bread	5	1	2
Butter	9	0	1
Milk	2	2	1
Pepsi	1	0	0
Fruits	4	1	0
Milk	3	1	1

EMF4. For each product show each month's total sales as percentage of this product's year-long total sales. (hierarchies)

SQL:

```
with B1 as
(select x.prod as prod,sum(quant) as syear
from Sales x
where x.year=2008
group by x.prod
),
B2 as
(select x.prod as prod ,x.month as month, sum(x.quant)as smonth
from Sales x
where x.year=2008
group by x.prod, x.month)

select B1.prod, B2.month, (B2.smonth *100.00  / B1.syear)
from B1, B2
where B1.prod=B2.prod
order by B1.prod, B2.month
```

EMF:

```
select prod,month,year,(sum(X.quant)/sum(Y.quant))*100.00
from sales
group by prod,month:X,Y
such that X.prod=prod and X.month=month and X.year=year,
          Y.prod=prod and Y.year=year
```

Phi Operator:

```
selection_attributes = prod,month,mftable[index].sum_quant_1 * 100.0/mftable[index].sum_quant_2
where_clause = year\=2008
grouping_attributes = prod,month
no_of_gv = 2
f_vect = sum_quant_1,sum_quant_2
condition_vect = mftable[index].prod.equals(prod) && mftable[index].month == month, mftable[index].prod.equals(prod)
having_condition =
```

Query Output:

```
QP class main()
Success loading Driver!
prod    |       month |       mftable[index].sum_quant_1 * 100.0/mftable[index].sum_quant_2
-----------------------------------------------------------------
Pepsi   |       12      |       2.122610843843453
Milk    |       9       |       15.834845735027223
Coke    |       10      |       24.244901295622675
Milk    |       10      |       38.141395809272396
Soap    |       4       |       42.01105554093182
Bread   |       11      |       7.9573156708523065
Eggs    |       9       |       28.436578171091444
Butter  |       3       |       23.475111007244685
Pepsi   |       3       |       14.575477831231309
Cookies         |       12      |       53.81980130181569
Soap    |       3       |       57.98894445906818
Milk    |       1       |       13.228015179013363
Coke    |       7       |       27.94784812196019
Coke    |       5       |       10.438549883516574
Pepsi   |       11      |       18.242101157196725
Bread   |       9       |       7.810128328963709
Pepsi   |       6       |       9.51761799505916
Yogurt  |       1       |       25.603633755677745
```

Yogurt	3	19.842218503466412
Yogurt	5	54.554147740855846
Cookies	2	43.21685508735868
Pepsi	7	25.607853335066963
Milk	4	13.434251773634713
Fruits	12	41.354330708661415
Coke	1	3.792863857440634
Coke	11	7.835043119303552
Cookies	11	2.9633436108256253
Butter	10	16.662771675625144
Milk	7	0.1072430292031018
Pepsi	9	15.592900793134833
Bread	6	26.673106112874294
Bread	3	20.109470585529646
Bread	4	9.82015546662987
Coke	6	12.939878203294233
Fruits	7	11.149606299212598
Butter	6	38.13975227856976
Coke	2	12.80091551886214
Eggs	5	32.86135693215339
Fruits	3	15.42257217847769
Bread	12	5.087162504024654
Eggs	12	38.70206489675516
Bread	5	22.542661331125522
Butter	9	21.72236503856041
Milk	2	14.164329318594291
Pepsi	1	14.341438044467559
Fruits	4	32.07349081364829
Milk	3	5.089919155254909

EMF5. For each product, find for each month the total of the sales with quantity greater than the all- product year-long average sale, as percentage of the product's year-long total sales. (hierarchies)

SQL:

```
with B1 as
(select x.prod as prod,sum(quant) as syear, avg(quant) as ayear
```

```
from Sales x
where x.year=2008
group by x.prod
),
B2 as(
select x.prod as prod ,x.month as month, sum(x.quant) as smonth
from Sales x, B1 b
where x.year=2008 and x.prod = b.prod and x.quant > b.ayear
group by x.prod, x.month
)

select B1.prod, B2.month, (B2.smonth *100.00  / B1.syear)
from B1, B2
where B1.prod=B2.prod
```

EMF:

```
select prod,month,year,sum(X.quant)/sum(Y.quant)
from sales
group by prod,month; X,Y
such that X.prod=prod and X.month=month and X.year=year,
          Y.prod=prod and Y.year=year
having X.quant>avg(y.quant)
```

Phi Operator:

```
selection_attributes = prod,month,mftable[index].sum_quant_1 * 100.0/mftable[index].sum_quant_2
where_clause = year\=2008
grouping_attributes = prod,month
no_of_gv = 2
f_vect = quant_1,sum_quant_1,sum_quant_2,avg_quant_2
condition_vect = mftable[index].prod.equals(prod) && mftable[index].month == month, mftable[index].prod.equals(prod)
having_condition = mftable[index].quant_1 > avg_quant_2
```

Query Output:

```
QP class main()
Success loading Driver!
prod    |       month |       mftable[index].sum_quant_1 * 100.0/mftable[index].sum_quant_2
------------------------------------------------------------
Pepsi   |       12      |       2.5841940717875658
```

Milk	9	29.336695705333945
Coke	10	46.08810504234325
Milk	10	70.66330429466606
Soap	4	47.45762711864407
Bread	11	10.867516803819335
Eggs	9	28.436578171091444
Butter	3	33.44153141905951
Pepsi	3	17.745063120819978
Cookies	12	78.3932135728543
Soap	3	65.5069878085043
Milk	1	24.507106831728564
Coke	7	53.127185144899386
Coke	5	19.84305803744853
Pepsi	11	22.209030828287624
Bread	9	10.666499151956781
Pepsi	6	11.587320432150065
Yogurt	1	31.941544885177453
Yogurt	3	24.75395168505816
Yogurt	5	68.05845511482255
Cookies	2	62.949101796407184
Pepsi	7	31.17654042502671
Milk	4	24.889194559070763
Fruits	12	100.0
Coke	1	7.210006992463678
Coke	11	14.893947634216456
Cookies	11	4.3163672654690615
Butter	10	23.73699542238868
Milk	7	0.19868561821794284
Pepsi	9	18.983735011278643
Bread	6	36.42816759846724
Bread	3	27.464036685721464
Bread	4	13.411646460204787
Coke	6	24.597933338512934
Fruits	7	26.961157654226962
Butter	6	54.332084893882644
Coke	2	24.333773599564914
Eggs	5	32.86135693215339
Fruits	3	37.29372937293729
Bread	12	6.947672592499529
Eggs	12	38.70206489675516
Bread	5	30.787109743074314
Butter	9	30.944652517686226
Milk	2	26.241785113862143

Pepsi	1	17.46012901183268
Fruits	4	77.55775577557756
Milk	3	9.42992511080544

EMF6.“For each customer, show for each product the customer’s average sale,and the other customers’ average sale (for the same product).”

SQL:

```
select x.cust, x.prod, avg(y.quant), avg(z.quant)
from sales x, sales y, sales z
where x.cust = y.cust and x.prod=y.prod
and x.cust<>z.cust and x.prod=z.prod
group by x.cust,x.prod
```

EMF:

```
select cust,prod,avg(X.quant),avg(Y.quant)
from sales
group by cust,prod;X,Y
such that X.cust=cust and X.prod= prod,
          X.cust<>cust and X.prod= prod
```

Phi Operator:

```
selection_attributes = cust,prod,avg_quant_1,avg_quant_2
where_clause =
grouping_attributes = cust,prod
no_of_gv = 2
f_vect = avg_quant_1,avg_quant_2
condition_vect = mftable[index].cust.equals(cust) && mftable[index].prod.equals(prod),!mftable[index].cust.equals(cust)
&& mftable[index].prod.equals(prod)
having_condition =
```

Query Output:

```
QP class main()
Success loading Driver!
cust    |       prod    |       avg_quant_1  |       avg_quant_2
-----------------------------------------------------------------
Bloom   |       Pepsi   |       1923.0 |       2840.0
Knuth   |       Bread   |       2713.0 |       2029.0
Emily   |       Pepsi   |       3254.0 |       2622.0
Emily   |       Fruits  |       2696.0 |       2366.0
Helen   |       Milk    |       2103.0 |       2199.0
Emily   |       Soap    |       2411.0 |       2199.0
Bloom   |       Eggs    |       2504.0 |       2487.0
Bloom   |       Yogurt  |       2208.0 |       2515.0
Helen   |       Pepsi   |       2662.0 |       2717.0
Emily   |       Bread   |       1534.0 |       2350.0
Sam     |       Cookies         |       1747.0 |       2634.0
Knuth   |       Milk    |       924.0  |       2283.0
Helen   |       Coke    |       2617.0 |       2487.0
Emily   |       Butter  |       2316.0 |       2056.0
Emily   |       Yogurt  |       2358.0 |       2477.0
Sam     |       Soap    |       2201.0 |       2248.0
Knuth   |       Coke    |       2726.0 |       2456.0
Sam     |       Milk    |       1814.0 |       2300.0
Helen   |       Yogurt  |       2655.0 |       2395.0
Knuth   |       Pepsi   |       2646.0 |       2722.0
Knuth   |       Eggs    |       2538.0 |       2479.0
Knuth   |       Yogurt  |       2644.0 |       2413.0
Helen   |       Fruits  |       1840.0 |       2533.0
Helen   |       Cookies         |       1779.0 |       2588.0
Knuth   |       Fruits  |       1847.0 |       2602.0
Emily   |       Cookies         |       2672.0 |       2413.0
Bloom   |       Coke    |       2338.0 |       2565.0
Helen   |       Soap    |       2499.0 |       2171.0
Knuth   |       Soap    |       1856.0 |       2360.0
Knuth   |       Cookies         |       2608.0 |       2473.0
Emily   |       Milk    |       2663.0 |       2086.0
Bloom   |       Soap    |       2377.0 |       2215.0
Emily   |       Eggs    |       2756.0 |       2405.0
Bloom   |       Milk    |       2655.0 |       1995.0
Sam     |       Yogurt  |       2368.0 |       2481.0
Sam     |       Fruits  |       3387.0 |       2254.0
```

Sam	Butter	2701.0	1986.0	
Bloom	Fruits	2471.0	2421.0	
Helen	Eggs	2135.0	2560.0	
Sam	Pepsi	2988.0	2614.0	
Helen	Butter	1908.0	2271.0	
Helen	Bread	2781.0	2063.0	
Bloom	Butter	3017.0	2044.0	
Sam	Bread	1678.0	2285.0	
Sam	Eggs	2372.0	2514.0	
Knuth	Butter	1257.0	2226.0	
Bloom	Cookies	3069.0	2367.0	
Sam	Coke	2270.0	2553.0	
Emily	Coke	2409.0	2550.0	
Bloom	Bread	2111.0	2214.0	