# Project 1 Report
# CSE 601
# Data Mining and Bioinformatics
# Data Warehouse/OLAP System

**GROUP MEMBERS:**

Arundhati Rao | Person Number: 50169206
Cuurie Athiyaman | Person Number: 50170367
Sukanya Moorthy | Person Number: 50170061

## PART 1:

Designing a schema for a data warehouse for biological data is a complex business. The schema needs to be flexible enough for making changes since the data is volatile and also simple while providing efficient query processing.

### SCHEMA MODELS

A schema is a collection of database objects, including tables, views, indexes, and synonyms. Conventional models for schema are star schema and snowflake schema. The third normal form schema models are also very common. Some schema models are hybrid models designed specifically for certain applications.

### STAR SCHEMA

The star schema is the simplest data warehouse schema. The center of the star is a fact table and the dimension tables are connected to the star table. A star query is a join between a fact table and a number of dimension tables. Each dimension table is joined to the fact table using a primary key.

### FACT CONSTELLATION SCHEMA

Fact constellation is an extension of the star schema in that it contains many fact tables that share dimension tables.

The provided data warehouse schema is an example of fact constellation schema design.

### TIME COMPLEXITY FOR OLAP OPERATIONS

Analysis of OLAP operations can be sampled based on the time complexity of a simple "select" statement for a relational database model like the one implemented.

There are two ways in which a "select" operation works:

1. It either does a full table scan which would result in O(n) time complexity since all the data items are processed.

2. Or it processes only a select range of indexes given by the user. In which case the O(log(n)) in order of the range of indexes selected.

But when viewed at a lower level, the time complexity depends on many other factors such as how many dimension tables are accessed, efficiency of the fact tables, number of NULL values, number of joins made etc. When more number of joins are made, the complexity is in terms of O(n+m) where n is the number of rows in the first table and m is the number of rows in the other table.

For sample query :

```
select count(patient.p_id) from clinical_fact,patient,disease where
clinical_fact.p_id=patient.p_id and disease.ds_id=clinical_fact.ds_id and
disease.name like '%tumor%';
```

We observed that we get results in 0.206 seconds. This is the execution time for the SQL query.

## PART 2:

The data given in some of the tables, for example clinical data, were incomplete i.e.,there are many missing or NA values. Since not all patients will have both clinical and genomic data, there can be NULL, NA or MISSING values in the clinical_fact table. The data cleanup process included populating s_id with NULL values with patient's s_id.

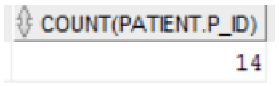
After performing data cleanup, we found that our queries became more simple.

The following are the results for the sample queries given in the project description.

1) List the number of patients who had " tumor" (disease description), " leukemia" (disease type) and " ALL" (disease name), separately.

**<u>Query to list number of patients who had tumor:</u>**

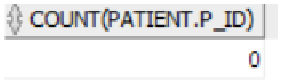
```
select count(patient.p_id) from clinical_fact,patient,disease where
clinical_fact.p_id=patient.p_id and disease.ds_id=clinical_fact.ds_id and
disease.name like '%tumor%';
```



```
COUNT(PATIENT.P_ID)
                 14
```

**<u>Query to list number of patients who had "leukamia":</u>**

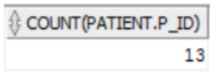
```
select count(patient.p_id) from clinical_fact,patient,disease where
clinical_fact.p_id=patient.p_id and disease.ds_id=clinical_fact.ds_id and

disease.name  like '%leukamia%';
```



```
COUNT(PATIENT.P_ID)
                  0
```

**<u>Query to list number of patients who had "ALL":</u>**

```
select count(patient.p_id) from clinical_fact,patient,disease where
clinical_fact.p_id=patient.p_id and disease.ds_id=clinical_fact.ds_id and
disease.name like '%ALL%';
```

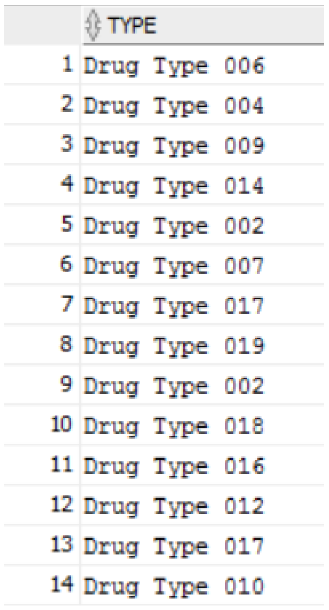


```
COUNT(PATIENT.P_ID)
                 13
```

2) List the types of drugs, which have been applied to patients with " tumor".

**Query:**

select drug.type from drug ,disease,patient,clinical_fact where clinical_fact.dr_id=drug.dr_id and disease.ds_id=clinical_fact.ds_id and patient.p_id=clinical_fact.p_id and disease.name like '%tumor%';



| | TYPE |
|---|---|
| 1 | Drug Type 006 |
| 2 | Drug Type 004 |
| 3 | Drug Type 009 |
| 4 | Drug Type 014 |
| 5 | Drug Type 002 |
| 6 | Drug Type 007 |
| 7 | Drug Type 017 |
| 8 | Drug Type 019 |
| 9 | Drug Type 002 |
| 10 | Drug Type 018 |
| 11 | Drug Type 016 |
| 12 | Drug Type 012 |
| 13 | Drug Type 017 |
| 14 | Drug Type 010 |

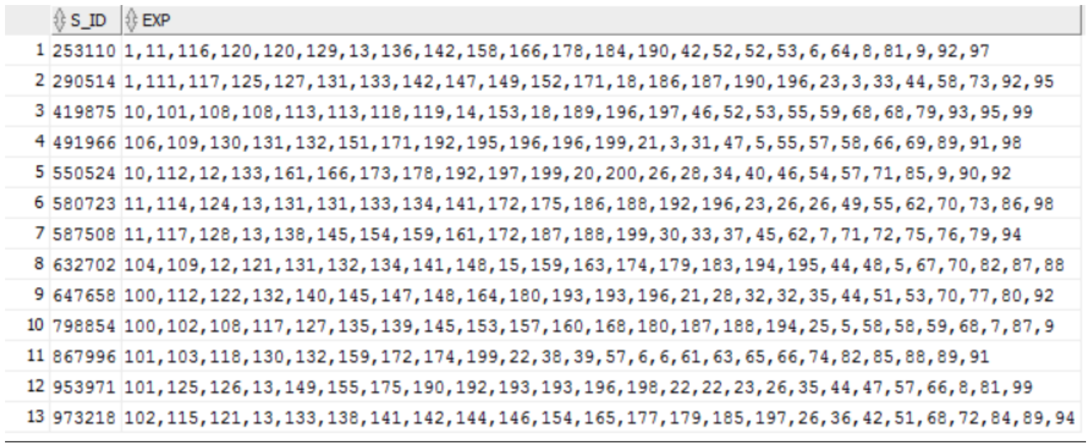
3) For each sample of patients with " ALL", list the mRNA values (expression) of probes in cluster id " 00002" for each experiment with measure unit id = " 001". (Note: measure unit id corresponds to mu_id in microarray_fact.txt, cluster id corresponds to cl_id in gene_fact.txt, mRNA expression value corresponds to exp in microarray_fact.txt, UID in probe.txt is a foreign key referring to gene_fact.txt)

**Query to find the mRNA values for each of patients with "ALL" with given measures:**

create table forag as(select microarray_fact.s_id,microarray_fact.e_id,microarray_fact.exp from microarray_fact,probe,gene_fact where microarray_fact.pb_id=probe.pb_id and probe.u_id=gene_fact.u_id and gene_fact.cl_id='2' and microarray_fact.mu_id='1' and microarray_fact.s_id in ( select test_fact.s_id  from test_fact,disease where test_fact.ds_id=disease.ds_id and disease.name like '%ALL%' and test_fact.s_id in(select microarray_fact.s_id from sample_fact,microarray_fact where sample_fact.s_id=microarray_fact.s_id)));

**Query to display results as a list:**

select s_id, LISTAGG(exp,',') within group (order by s_id) as EXP  from forag group by S_ID;



|  | S_ID | EXP |
|---|---|---|
| 1 | 253110 | 1,11,116,120,120,129,13,136,142,158,166,178,184,190,42,52,52,53,6,64,8,81,9,92,97 |
| 2 | 290514 | 1,111,117,125,127,131,133,142,147,149,152,171,18,186,187,190,196,23,3,33,44,58,73,92,95 |
| 3 | 419875 | 10,101,108,108,113,113,118,119,14,153,18,189,196,197,46,52,53,55,59,68,68,79,93,95,99 |
| 4 | 491966 | 106,109,130,131,132,151,171,192,195,196,196,199,21,3,31,47,5,55,57,58,66,69,89,91,98 |
| 5 | 550524 | 10,112,12,133,161,166,173,178,192,197,199,20,200,26,28,34,40,46,54,57,71,85,9,90,92 |
| 6 | 580723 | 11,114,124,13,131,131,133,134,141,172,175,186,188,192,196,23,26,26,49,55,62,70,73,86,98 |
| 7 | 587508 | 11,117,128,13,138,145,154,159,161,172,187,188,199,30,33,37,45,62,7,71,72,75,76,79,94 |
| 8 | 632702 | 104,109,12,121,131,132,134,141,148,15,159,163,174,179,183,194,195,44,48,5,67,70,82,87,88 |
| 9 | 647658 | 100,112,122,132,140,145,147,148,164,180,193,193,196,21,28,32,32,35,44,51,53,70,77,80,92 |
| 10 | 798854 | 100,102,108,117,127,135,139,145,153,157,160,168,180,187,188,194,25,5,58,58,59,68,7,87,9 |
| 11 | 867996 | 101,103,118,130,132,159,172,174,199,22,38,39,57,6,6,61,63,65,66,74,82,85,88,89,91 |
| 12 | 953971 | 101,125,126,13,149,155,175,190,192,193,193,196,198,22,22,23,26,35,44,47,57,66,8,81,99 |
| 13 | 973218 | 102,115,121,13,133,138,141,142,144,146,154,165,177,179,185,197,26,36,42,51,68,72,84,89,94 |

4) For probes belonging to GO with id = " 0012502", calculate the t statistics of the expression values between patients with " ALL" and patients without " ALL". (Note: Assume the expression values of patients in both groups have equal variance, use the t test for unequal sample size, equal variance)

In this query, the STATS_T_TEST_INDEP is used which is the t_test between two groups with equal variances. T_Test generally measures the differences between means of the two groups.

**T-Test:**
Check if two groups are reliably different from each other and not just chance .It helps inferring property of new test samples
t-value = variance between groups/variance within groups
More the t-value more the difference.
How do we if t-value is big enough to show the difference?
We use p-value .P-value tells us the likelihood that there is a real difference.
p-value can be said as the probability that the pattern of data in the sample could be produced by random data.
If p=0.01 it implies there is only 1% chance
P-value for each t-value depends on size of the sample size.
Bigger samples make it easier to detect differences .
Sample size is determined by degrees of freedom (df) = samplesize-1
Types of T-test:
Independent: tests means of two different groups
Paired: tests mean of one group twice (When two groups are not related)
One-sample: One group with a hypothetical value
Limitations:
The results can only be applied to samples that resemble the test sample.
Sample and population should be normal in distribution.
Each group should have almost same number of data points to avoid inaccuracy.

**Query to create view which contains the groups of patients with "ALL" and patients with "NOT ALL":**

CREATE OR REPLACE FORCE VIEW "SUKANYAM"."TTEST_VIEW" ("PATIENT", "N1") AS

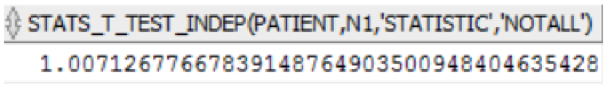
(select 'NOT ALL' as patient, microarray_fact.exp as n1 from microarray_fact,probe,gene_fact where microarray_fact.pb_id=probe.pb_id and gene_fact.go_id='12502' and gene_fact.u_id=probe.u_id and microarray_fact.s_id in (select test_fact.s_id from test_fact,disease where test_fact.ds_id=disease.ds_id and disease.name!='ALL')

union all

select 'ALL' as patient , microarray_fact.exp as n1 from microarray_fact,probe,gene_fact where microarray_fact.pb_id=probe.pb_id and gene_fact.go_id='12502' and gene_fact.u_id=probe.u_id and microarray_fact.s_id in (select test_fact.s_id from test_fact,disease where test_fact.ds_id=disease.ds_id and disease.name='ALL'))

**Query to calculate the t_test for "NOT ALL":**

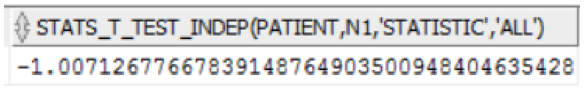
select STATS_T_TEST_INDEP(patient, n1,'STATISTIC', 'NOT ALL') from ttest_view;



```
STATS_T_TEST_INDEP(PATIENT,N1,'STATISTIC','NOTALL')
  1.00712677667839148764903500948404635428
```

**Query for calculating the t_test for "ALL":**

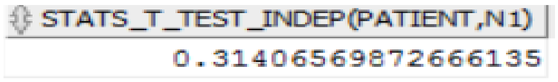
select STATS_T_TEST_INDEP(patient, n1,'STATISTIC', 'ALL') from ttest_view;



```
STATS_T_TEST_INDEP(PATIENT,N1,'STATISTIC','ALL')
-1.00712677667839148764903500948404635428
```

**Query for calculating the p_value:**

select STATS_T_TEST_INDEP(patient, n1) from ttest_view;



```
STATS_T_TEST_INDEP(PATIENT,N1)
           0.31406569872666135
```

5) For probes belonging to GO with id=" 0007154", calculate the F statistics of the expression values among patients with " ALL", " AML", "colon tumor" and " breast tumor". (Note: Assume the variances of expression values of all four patient groups are equal.)

In this query, The STATS_ONE_WAY_ANOVA function is used to compare the mean square error within and mean square error between the groups of patients. The function can be used for three or more samples of data. Usually, the former is lesser than the later.

F-Test and Anova:
Analysis of Variance (ANOVA) is a hypothesis testing procedure that tests whether two or more means are significantly different from each other. A statistic, F, is calculated that measures the size of the effects by comparing a ratio of the differences between the means of the groups to the variability within groups. The larger the value of F, the more likely that there are real effects. The obtained F-ratio is compared to a model of F-ratios that would be found given that there were no effects. If the obtained F-ratio is unlikely given the model of no effects, the hypothesis of no effects is rejected and the hypothesis of real effects is accepted. If the model of no effects could explain the results, then the null hypothesis of no effects must be retained. The exact significance level is the probability of finding an F-ratio equal to or larger than the one found in the study given that there were no effects. If the exact significance level is less than alpha, then you decide that the effects are real, otherwise you decide that chance could explain the results.

Anova is a statistical method used to compare the means of two or more groups.
Factor: Disease
Levels: ALL,AML

One-way-Anova: Has atleast two levels and levels are independent.

Assumption:

Normality of sampling distribution of means
Errors between cases are independent
Absence of Outliers
Hypothesis: (x1=x2 or x1! =x2)

Value of F:

F=Actual Difference + Random Difference/Random Difference.

If F=1 (no actual effect)

If F>1 (differences exist)

F cannot be negative

The F-ratio can be thought of as a measure of how different the means are relative to the variability within each sample. As such, the F-ratio is a measure of the size of the effects. The larger this value, the greater the likelihood that the differences between the means are due to something other than chance alone, namely real effects. How big this F-ratio needs to be in order to make a decision about the reality of effects is the next topic of discussion.

Significance:

If significance <0.05 then the null hypothesis must be rejected

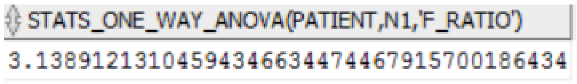
**STATS_ONE_WAY_ANOVA** takes three arguments: two expressions and a return value of type **VARCHAR2**. ***expr1*** is an independent or grouping variable that divides the data into a set of groups. ***expr2*** is a dependent variable (a numeric expression) containing the values corresponding to each member of a group. The function returns one number, determined by the value of the third argument. If you omit the third argument, the default is **SIG**, which gives the significance. If it is F_RATIO, it gives the ratio of mean square difference within and between.

**Query for creating the view that contains the groups of expressions for patients with "ALL", "AML","colon tumor" and "breast tumor":**

```
create view ftest_view as ((select  'ALL' as patient, microarray_fact.exp as n1 from
microarray_fact,probe,gene_fact where microarray_fact.pb_id=probe.pb_id and
gene_fact.u_id=probe.u_id and gene_fact.go_id='7154'and microarray_fact.s_id in (select
test_fact.s_id from test_fact,disease where  test_fact.ds_id=disease.ds_id and
disease.name='ALL'))

union all

(select  'AML' as patient, microarray_fact.exp as n1 from microarray_fact,probe,gene_fact
where microarray_fact.pb_id=probe.pb_id and gene_fact.u_id=probe.u_id and
gene_fact.go_id='7154' and microarray_fact.s_id in (select test_fact.s_id from test_fact,disease
where  test_fact.ds_id=disease.ds_id and disease.name='AML'))

union all

select  'Colon tumor' as patient, microarray_fact.exp as n1 from microarray_fact,probe,gene_fact
where microarray_fact.pb_id=probe.pb_id and gene_fact.u_id=probe.u_id and
gene_fact.go_id='7154'and microarray_fact.s_id in (select test_fact.s_id from test_fact,disease
where  test_fact.ds_id=disease.ds_id and disease.name='Colon tumor')

union all

(select  'Breast tumor' as patient, microarray_fact.exp as n1 from microarray_fact,probe,gene_fact
where microarray_fact.pb_id=probe.pb_id and gene_fact.u_id=probe.u_id and
gene_fact.go_id='7154'and microarray_fact.s_id in (select test_fact.s_id from test_fact,disease
where  test_fact.ds_id=disease.ds_id and disease.name='Breast tumor')));
```

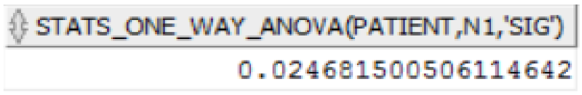
**Query for calculating the F_statistics:**

```
select STATS_ONE_WAY_ANOVA(patient,n1,'F_RATIO')from ftest_view;
```



```
STATS_ONE_WAY_ANOVA(PATIENT,N1,'F_RATIO')
3.1389121310459434663447446791570018643
```

```
select STATS_ONE_WAY_ANOVA(patient,n1,'SIG')from ftest_view;
```



```
STATS_ONE_WAY_ANOVA(PATIENT,N1,'SIG')
                 0.024681500506114642
```

6) For probes belonging to GO with id=“ 0007154”, calculate the average correlation of the expression values between two patients with “ ALL”, and calculate the average correlation of the expression values between one “ ALL” patient and one “ AML” patient. (Note: For each patient, there is a list of gene expression values belonging to GO with id=“ 0007154”. Suppose you get □1 “ ALL” patients and □2 “ AML” patient. For the average correlation of the expression values between two patients with “ ALL”, you need first calculate □1 × (□1 − 1)/2 Person Correlations then calculate the average value. For the average correlation of the expression values between one “ ALL” patient and one “ AML” patient, you need first calculate □1 × □2 Person Correlations then calculate the average value.)

**<u>Queries for constructing the double[] arrays containing the expression values for each group of patients:</u>**

```
create view correlation_view_all as ((select 'ALL' as
patient,microarray_fact.s_id,microarray_fact.pb_id, microarray_fact.exp as n1 from
microarray_fact,probe,gene_fact where microarray_fact.pb_id=probe.pb_id and
gene_fact.u_id=probe.u_id and gene_fact.go_id='7154'and microarray_fact.s_id in (select
test_fact.s_id from test_fact,disease where  test_fact.ds_id=disease.ds_id and
disease.name='ALL')));
```

```
SELECT   s_id, LISTAGG(pb_id, ',') WITHIN GROUP (ORDER BY pb_id) AS probe,
LISTAGG(n1, ',') WITHIN GROUP (ORDER BY pb_id) FROM correlation_view_all GROUP
BY s_id;
```

```
create table correlation_table_all as( SELECT s_id,WM_CONCAT(n1) as
exp,WM_CONCAT(pb_id) as probe FROM (SELECT s_id,n1,pb_id FROM
correlation_view_all ORDER BY pb_id desc) GROUP BY s_id);
```

The results from the SQL query are then used in the function PearsonCorrelation ( double[], double[]) to find the Pearson correlation between each group of patient. The function PearsonCorrelation() is imported from the “apache commons math” package.

Pearson Correlation:

The Pearson correlation coefficient is a measure of the strength of a linear association between two variables and is denoted by *r*.

Assumptions:

Linear Relation
Remove outliers

Range:

The Pearson correlation coefficient, *r*, can take a range of values from +1 to -1. A value of 0 indicates that there is no association between the two variables. A value greater than 0 indicates a positive association; that is, as the value of one variable increases, so does the value of the other variable. A value less than 0 indicates a negative association; that is, as the value of one variable increases, the value of the other variable decreases

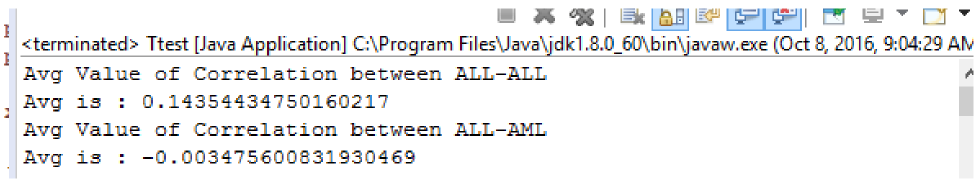
Functions to Run from Main():

1. correlation_table_all() - Returns list of correlation values between each pair of patients
2. correlation_table_all_aml() - Returns list of correlation values between each pair of patients
3. cal_average() - Returns average of all the correlation values

```
//Part 3 ALL_ALL
public static  ArrayList<Double> correlation_table_all(Statement
statement) throws SQLException{
       ArrayList<Ttest> ob = new ArrayList<Ttest>();
       ArrayList<Double> coa= new ArrayList<Double>();
       String query="select * from correlation_table_all";
       ob=table_read(statement,query);

            for (int i=0 ;i<ob.size();i++){
                  double[] p1=ob.get(i).exp_c;
                  for(int j=i+1;j<ob.size();j++){
                        double[] p2=ob.get(j).exp_c;
                        double corr = new
PearsonsCorrelation().correlation(p1, p2);
System.out.println(ob.get(i).sample_id+","+ob.get(j).sample_id+":"+cor;
                        coa.add(corr);}}

            return coa;

//Part 3 ALL_AML
public static ArrayList<Double> correlation_table_all_aml(Statement
statement) throws SQLException{
      ArrayList<Ttest> ob1 = new ArrayList<Ttest>();
      ArrayList<Ttest> ob2 = new ArrayList<Ttest>();
       ArrayList<Double> coa= new ArrayList<Double>();
       String query="select * from correlation_table_all";
       ob1=table_read(statement,query);
       query="select * from correlation_table_aml";
       ob2=table_read(statement,query);

            for (int i=0 ;i<ob1.size();i++){
                  double[] p1=ob1.get(i).exp_c;
                  for(int j=0;j<ob2.size();j++){
                        double[] p2=ob2.get(j).exp_c;
                        double corr = new
PearsonsCorrelation().correlation(p1, p2);
System.out.println(ob1.get(i).sample_id+","+ob2.get(j).sample_id+":"+co
rr);
coa.add(corr);}}

return coa;}
```

```
//Function to create ArrayList<Ttest> object
public static  ArrayList<Ttest>  table_read(Statement statement,String
query)  throws SQLException{

          ArrayList<Ttest> ob = new ArrayList<Ttest>();
          ResultSet resultset = statement.executeQuery(query);
          ArrayList<Double> coa= new ArrayList<Double>();
          while(resultset.next())
                  {
                        int
s_id=Integer.parseInt(resultset.getString(1));
                        String[] exp=resultset.getString(2).split(",");
                        String[]
probe=resultset.getString(3).split(",");
                        int exp_len = exp.length;
                        int probe_len=probe.length;
                        double[] exp_c= new double[exp_len];
                        double[] probe_c= new double[probe_len];
                        for (int i=0;i<exp_len;i++){
                        exp_c[i]=Double.parseDouble(exp[i]);
                        }
                        for (int i=0;i<probe_len;i++){
                        probe_c[i]=Double.parseDouble(probe[i]);
                  }
                        ob.add(new Ttest(s_id,exp_c,probe_c));
}
          return ob;

   }

// Function to setup java connection
public static Connection set_up() throws IOException,
ClassNotFoundException, SQLException{
            Properties prop = new Properties();
        InputStream inputStream =
DBUtility.class.getClassLoader().getResourceAsStream("config.properties
");
        prop.load(inputStream);
        String driver = prop.getProperty("driver");
        String url = prop.getProperty("url");
        String user = prop.getProperty("user");
        String password = prop.getProperty("password");
        Class.forName(driver);
        conn = DriverManager.getConnection(url, user, password);
        return conn;
      }
```

Results:



```
<terminated> Ttest [Java Application] C:\Program Files\Java\jdk1.8.0_60\bin\javaw.exe (Oct 8, 2016, 9:04:29 AM
Avg Value of Correlation between ALL-ALL
Avg is : 0.14354434750160217
Avg Value of Correlation between ALL-AML
Avg is : -0.003475600831930469
```

## PART 3:

Knowledge discovery is the process of discovering useful knowledge from a collection of data (database). The knowledge discovery required by the project description are as follows:-

Use your data warehouse and the OLAP operations to support knowledge discovery.

1) Given a specific disease, find the informative genes. For example, suppose we are interested in the cancer "ALL".
   (1) Find all the patients with "ALL" (group A), while the other patients serve as the control (group B).
   (2) For each gene, calculate the t-statistics for the expression values between group A and group B.
   (3) If the p-value of the t-test is smaller than 0.01, this gene is regarded as an "informative" gene

**<u>Query for forming the group of patients with 'ALL':</u>**

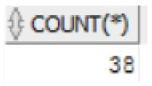
```
 create view viewnow1 as ((select  'ALL' as patient
,microarray_fact.s_id,gene_fact.u_id,avg(microarray_fact.exp) AS n1 from
microarray_fact,probe,gene_fact where microarray_fact.pb_id=probe.pb_id and
gene_fact.u_id=probe.u_id and  microarray_fact.s_id in (select test_fact.s_id from
test_fact,disease where  test_fact.ds_id=disease.ds_id and disease.name='ALL') GROUP BY
gene_fact.U_ID,microarray_fact.S_ID)

union all

(select  'OTHER' as patient ,microarray_fact.s_id,gene_fact.u_id,avg(microarray_fact.exp) as n1
from microarray_fact,probe,gene_fact where microarray_fact.pb_id=probe.pb_id and
gene_fact.u_id=probe.u_id and  microarray_fact.s_id in (select test_fact.s_id from
test_fact,disease where  test_fact.ds_id=disease.ds_id and disease.name<>'ALL') GROUP BY
gene_fact.U_ID,microarray_fact.S_ID));

select * from viewnow1;

drop table ftable;
```

**<u>Query for computing the t statistics:</u>**

```
create table ftable as (select U_ID ,STATS_T_TEST_INDEP(patient, n1) as p_val from
viewnow1 group by U_ID having STATS_T_TEST_INDEP(patient, n1)<0.01 );
```

**<u>Query for computing the count of informative genes for "ALL":</u>**

```
select count(U_ID) from ftable;
```



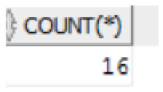
```
COUNT(*)
      38
```

```
--Informative Genes

 select u_id from ftable;
```

Similarly the number of informative genes observed for other diseases are as follows:

**For disease name =”AML”:**

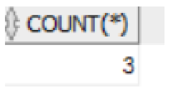
select count(*) from AML;



```
COUNT(*)
      16
```

**For disease.name =”Colon Tumor”:**

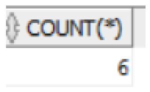
select count(*) from Colontumor;



```
COUNT(*)
       3
```

**For disease.name=” Breast Tumor”:**

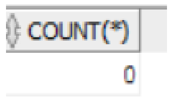
select count(*) from Breasttumor;



```
COUNT(*)
       6
```

**For disease.name=” Flu”:**

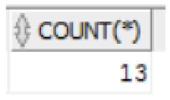
select count(*) from flu;



```
COUNT(*)
       0
```

**For disease.name=”Giloblastome”:**

select count(*) from giloblastome;



```
COUNT(*)
      13
```

2. Use informative genes to classify a new patient (five test cases in test_samples.txt are given in the data).

For example, given a new patient PN, we want to predict whether he/she has “ALL”.
1) Find the informative genes w.r.t. “ALL”.
2) Find all the patients with “ALL” (group A).
3) For each patient PA in group A, calculate the correlation rA of the expression values of the informative genes between PN and PA.
4) Patients without “ALL” serve as the control (group B).
5) For each patient PB in group B, calculate the correlation rB of the expression values of the informative genes between PN and PB.
6) Apply t-test on rA and rB, if the p-value is smaller than 0.01, the patient is classified as “ALL”.

**Query for forming the group of patients with ‘ALL’:**

```
create or replace view sm as(select
microarray_fact.s_id,microarray_fact.pb_id,gene_fact.u_id,avg(microarray_fact.exp) as n1 from
microarray_fact,probe,gene_fact where microarray_fact.pb_id=probe.pb_id and
gene_fact.u_id=probe.u_id and  microarray_fact.s_id in(select test_fact.s_id from
test_fact,disease where  test_fact.ds_id=disease.ds_id and disease.name='ALL')group by
microarray_fact.pb_id,microarray_fact.s_id,gene_fact.u_id);
```

```
create table part6 as (SELECT s_id,WM_CONCAT(n1) as n1,WM_CONCAT(u_id) as u_id
FROM (SELECT s_id,n1,u_id FROM sm ) where u_id in (select u_id from part32) GROUP BY
s_id);
```

```
create or replace view sm_notall as(select
microarray_fact.s_id,microarray_fact.pb_id,gene_fact.u_id,avg(microarray_fact.exp) as n1 from
microarray_fact,probe,gene_fact where microarray_fact.pb_id=probe.pb_id and
gene_fact.u_id=probe.u_id and  microarray_fact.s_id in(select test_fact.s_id from
test_fact,disease where  test_fact.ds_id=disease.ds_id and disease.name!='ALL')group by
microarray_fact.pb_id,microarray_fact.s_id,gene_fact.u_id);
```

```
create table part7 as (SELECT s_id,WM_CONCAT(n1) as n1,WM_CONCAT(u_id) as u_id
FROM (SELECT s_id,n1,u_id FROM sm_notall ) where u_id in (select u_id from part32)
GROUP BY s_id);
```

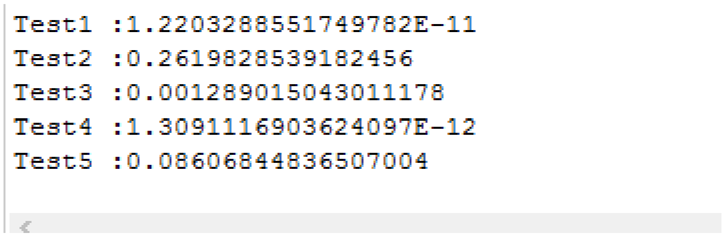
-----This is fed to Java code which then gives us the correlation value.
--correlation_table_test_all() - Returns list of correlation values  between Patients ALL and Patients N
 --correlation_table_test_notall() - Returns list of correlation values  between Patients NOT ALL and Patients N
--TTest().homoscedasticTTest(arr1,arr2) - Returns p_value for each patient

```
//Part 3 Correlation between ALL-PN
public   ArrayList<test> correlation_table_test_all(Statement
statement) throws SQLException, IOException{
      ArrayList<Ttest> ob1 = new ArrayList<Ttest>();
      ArrayList<Ttest> ob2 = new ArrayList<Ttest>();
      ArrayList<test> ob3 = new ArrayList<test>();
      ArrayList<test> ob4 = new ArrayList<test>();
      ArrayList<Double> ob5 = new ArrayList<Double>();
       ArrayList<Double> coa= new ArrayList<Double>();

       String query="select * from part6";
       ob1=table_read(statement,query);

       ob2=read_file("test_samples.txt");

       HashMap<Integer,Integer> mp =add_informative_gene(statement);
       int noofinformative =mp.size();
       ob2=clean_up_informative_gene(ob2,mp,noofinformative);
       for (int k=0;k<ob2.size();k++){
             double[] exp =new double[ob1.size()];
             ArrayList<Double> exp_list = new ArrayList<Double>();

             test test_ob= new test(k+1,exp,exp_list) ;
             ob4.add(test_ob);
       }

            for (int i=0 ;i<ob1.size();i++){
                  double[] p1=ob1.get(i).exp_c;
                  for(int j=0;j<ob2.size();j++){
                        test test_ob=ob4.get(j);

                        double[] p2=ob2.get(j).exp_c;
                        double corr = new
PearsonsCorrelation().correlation(p1, p2);
                        test_ob.exp_cd.add(corr);

      System.out.println(ob1.get(i).sample_id+","+ob2.get(j).sample_id+
":"+corr);
                        coa.add(corr);
                  }
            }

            for(int h=0;h<ob2.size();h++){
                  test test_ob=ob4.get(h);
                   ArrayList<Double> sp= test_ob.exp_cd;
                   double[] p1= new double[sp.size()];
                  for (int l=0;l< sp.size();l++){
                        p1[l]=sp.get(l);
                  }
                  test_ob.exp_c=p1;
            }
            return ob4;
   }
```

```
//Part 3 Correlation between NOT ALL-PN
public ArrayList<test> correlation_table_test_notall(Statement
statement) throws SQLException, IOException{
      ArrayList<Ttest> ob1 = new ArrayList<Ttest>();
      ArrayList<Ttest> ob2 = new ArrayList<Ttest>();
      ArrayList<test> ob3 = new ArrayList<test>();
      ArrayList<test> ob4 = new ArrayList<test>();
      ArrayList<Double> ob5 = new ArrayList<Double>();
       ArrayList<Double> coa= new ArrayList<Double>();

       String query="select * from part7";
       ob1=table_read(statement,query);

       ob2=read_file("test_samples.txt");

       HashMap<Integer,Integer> mp =add_informative_gene(statement);
       int noofinformative =mp.size();
       ob2=clean_up_informative_gene(ob2,mp,noofinformative);
       for (int k=0;k<ob2.size();k++){
             double[] exp =new double[ob1.size()];
             ArrayList<Double> exp_list = new ArrayList<Double>();

             test test_ob= new test(k+1,exp,exp_list) ;
             ob4.add(test_ob);
       }

            for (int i=0 ;i<ob1.size();i++){
                  double[] p1=ob1.get(i).exp_c;
                  for(int j=0;j<ob2.size();j++){
                        test test_ob=ob4.get(j);

                        double[] p2=ob2.get(j).exp_c;
                        double corr = new
PearsonsCorrelation().correlation(p1, p2);
                        test_ob.exp_cd.add(corr);

      System.out.println(ob1.get(i).sample_id+","+ob2.get(j).sample_id+
":"+corr);
                        coa.add(corr);
                  }
            }

            for(int h=0;h<ob2.size();h++){
                  test test_ob=ob4.get(h);
                   ArrayList<Double> sp= test_ob.exp_cd;
                   double[] p1= new double[sp.size()];
                  for (int l=0;l< sp.size();l++){
                        p1[l]=sp.get(l);
                  }
                  test_ob.exp_c=p1;
            }
            return ob4;
   }
```

```
//Maintain set of informative genes in hashmap
public HashMap<Integer,Integer> add_informative_gene(Statement
statement) throws SQLException{
       String query="select * from part32";
       ResultSet resultset = statement.executeQuery(query);
       HashMap<Integer,Integer> map = new HashMap<Integer,Integer>();
        while(resultset.next())
            {
            int val1=Integer.parseInt(resultset.getString(1));
            map.put(val1,1);
            }
        return map;

}
//Take only informative genes from Text File
public  ArrayList<Ttest> clean_up_informative_gene(ArrayList<Ttest>
ob,HashMap<Integer,Integer> map,int no){
       ArrayList<Ttest> obj = new ArrayList<Ttest>();
       for (int i=0;i<ob.size();i++){
            Ttest ttest=ob.get(i);
            double[] exp =ttest.exp_c;
            double[] uid=ttest.probe_c;
            double[] exp_final=new double[no];
            double[] uid_final=new double[no];
            int k=0;
            for (int j=0;j<exp.length;j++){
                  int xx=(int) uid[j];
                  if(map.get(xx)!=null){
                        exp_final[k]=exp[j];
                        uid_final[k]=uid[j];
                        k++;

                  }
                  ttest.exp_c=exp_final;
                  ttest.probe_c=uid_final;
            }
            obj.add(ttest);
      }
       return obj;
}
```

## Results:



```
Test1 :1.2203288551749782E-11
Test2 :0.2619828539182456
Test3 :0.001289015043011178
Test4 :1.3091116903624097E-12
Test5 :0.08606844836507004
```

The patients 1,3,4 are under ALL as their p_value<0.01

## CONCLUSION

The data warehouse schema was implemented based on the schema provided along with datasets. The OLAP layers for the same were implemented. However, the schema which contains measure tables and temporal tables are better equipped for biomedical data as it better represents many-to-many relationships between fact and dimension tables and also allows for the volatile nature of the data. Any queries with respect to time can be easily implemented using the database schema that implements temporal tables. However since most of the sample queries were implemented with ease in the available schema, the temporal tables are not included in this design.  For future query processing involving more dynamic and volatie data, the schema with measure table and temporal tables can be implemented.

## REFERENCES

1) https://docs.oracle.com/