Project 1

Sagar Dhamija (50169364 - sagardha)

Varun Khandelwal (50168936 - varunkha)

Contents

Part 1:

Implementation of data warehouse schema:

For this particular project we had devised 'BioVault' schema model based on Data Vault Model. The following are the details of our schema model:

Data Vault Model

"The Data Vault is a detail oriented, historical tracking and uniquely linked set of normalized tables that support one or more functional areas of business. It is a hybrid approach encompassing the best of breed between 3rd normal form (3NF) and star schema. The design is flexible, scalable, consistent and adaptable to the needs of the enterprise. It is a data model that is architected specifically to meet the needs of enterprise data warehouses.**" -- Dan Linstedt**

Data Vault Components

A data vault is composed of simple yet powerful components specifically called the Hub, Link and satellite entities. The Data Vault design is focused around the functional areas of business with the Hub representing the primary key, the Link Entities provides transaction integration between the Hubs and the Satellite Entities provide the context of the Hub primary key.

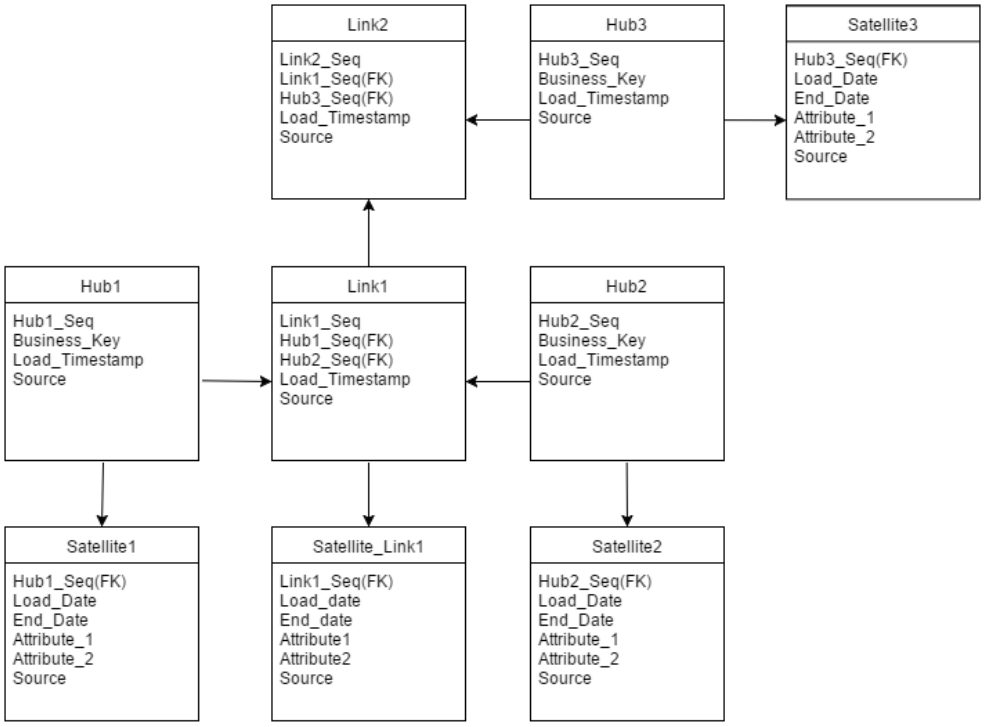


Fig1. A Data Vault sample

Hub Entities

Hub Entities, or Hubs, are tables that contain unique business keys. For example, patient id, employee id, product id, and invoice id. We generally create one hub for each entity. A hub generally comprises of:

- Surrogate Key – a sequential number.
- Business Key – Uniquely identifies the entity from the source side.
- Load Date Time Stamp – recording when the key itself first arrived in the warehouse.
- Record Source – A recording of the source system utilized for data traceability (optional).

Links

Link Entities or Links, are a physical representation of a many-to-many 3NF relationship. The Link represents the relationship or transaction between two or more business components (two or more business keys). A link can connect two hubs or one hub and one link. The latter connection type is generally used when we have data at different grain levels. A link generally comprises of:

- Surrogate Key – a sequential number.
- Hub/Link Key –Represents the relationship between two hubs/links.
- Load Date Time Stamp – recording when the key itself first arrived in the warehouse.
- Record Source – A recording of the source system utilized for data traceability (optional)

Satellite Entities

Satellites contain descriptive information about the hubs/links. Whenever a satellite is connected to a link it generally contains facts/ measures relevant to the entities connected to the link. A link generally comprises of:

- Hub/Link Key –Borrowed from the hub/link it is connected to.
- Attributes – Descriptive information about the hub/link. It can also be facts/measures if the satellite is connected to a link.
- Load Date – Records insert date
- End Date – Records end date/invalidate date
- Source – A recording of the source system utilized for data traceability (optional)

BioVault(Data Vault) for Clinical Data Space

The first step in Data Vault Modeling is defining Hubs. We have created hubs for:

1) Patient
2) Disease
3) Drug
4) Sample
5) Test

There is one hub for every entity and every hub will contain only one record per business key (1:1 mapping).

The second step is to create Links. Links basically represent the relationship between Hubs and/or other Links. They take care of many-to-many relationship if any and are also used for joins. We have created links between:

1) Patient and Disease
2) Patient and Sample
3) Sample and Test
4) Patient/Disease Link and Drug. This one is different from the others as it is between a Hub "Drug" and a link "Patient/Disease". We created a separate link as multiple drugs can be given for a single disease and this indicates a change in granularity of the data.

The last step is to create Satellites. We created satellites for Hubs to store the descriptive information. We also created satellites for Links when a fact/measure needed to be recorded as facts/measures are generally dependent on multiple dimensions or in these case hubs. We created satellites for:

1) Patient
2) Disease
3) Drug
4) Test
5) Sample
6) Symptom/Disease satellite to record the symptoms and disease start and end dates
7) Dosage satellite to record dosage
8) Result satellite to record the result of the test performed on the sample

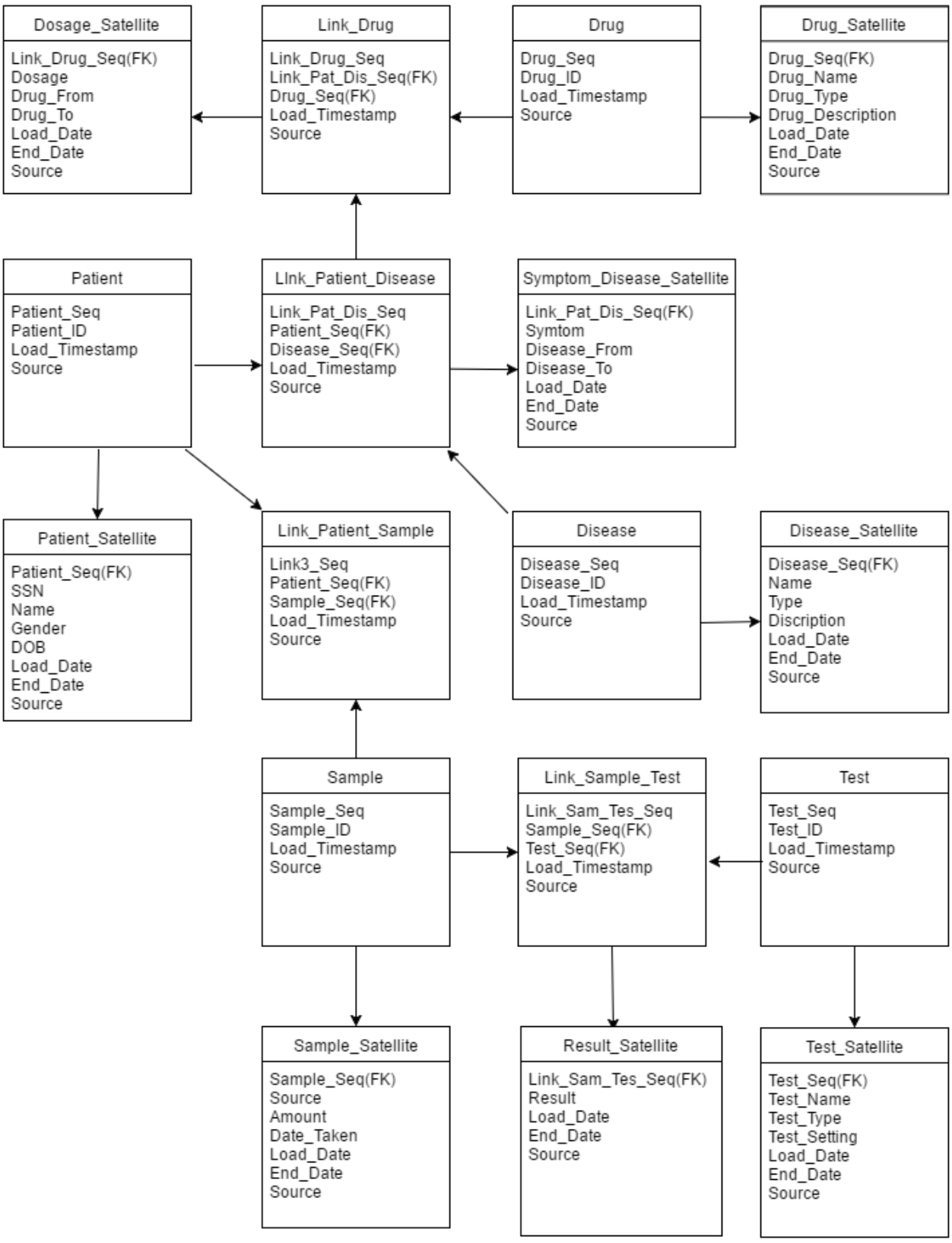
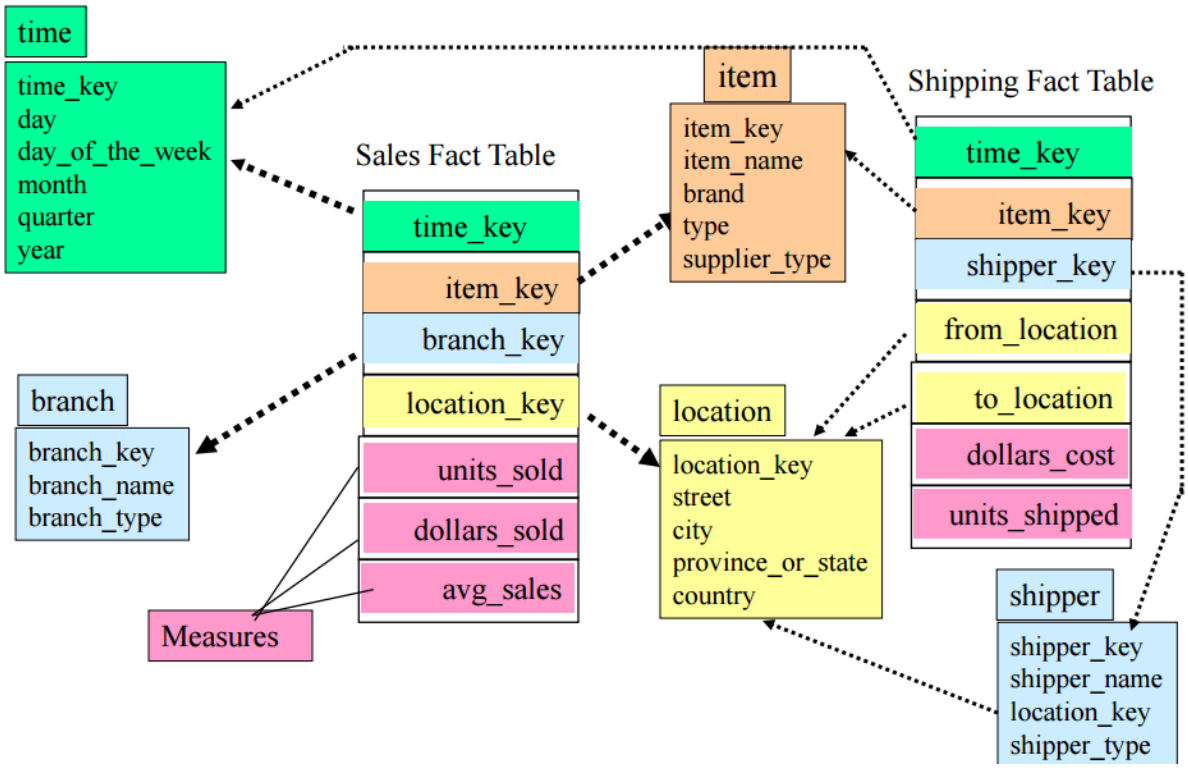


Fig4. BioVault (Data Vault) on Clinical Data Space

From the above explanation it can be observed that for a single fact table we have 17 other links and satellite tables and hence a fairly complex structure. Given the scenario that we have 5 fact tables our BioVault schema would result in atleast 85 other links and satellite tables and a structure very complex to write query and retrieve data. Therefore, we decided to ***implement the Fact Constellation schema*** provided to us as a part of the problem statement.

Fact Constellation Schema



- A constellation is a collection of stars.
- A Fact constellation schema is thus a collection of star schemas
- In a Fact constellation schema different fact tables are connected to each other with several common dimensions.
- From the above sample fact constellation schema, we can see that 'Sales Fact Table' and 'Shipping Fact Table' are connected to each other through the following common dimensions: 'item', 'time', and 'location'.

Time Complexity for OLAP operation:

Sample Query:

```
select P_ID, S_ID, UID_COLUMN, PB_ID, EXP from FACT_MICROARRAY inner join

(select P_ID, S_ID from FACT_CLINICAL inner join

(select P_ID from FACT_CLINICAL inner join DIM_DISEASE using(DS_ID) where NAME='ALL')

using(P_ID) where S_ID is not NULL)

using(S_ID) inner join DIM_PROBE using(PB_ID) inner join FACT_GENE using(UID_COLUMN)

where CL_ID=2 and mu_id=1 order by P_ID, UID_COLUMN;
```

For the above sample query, we have performed some fundamental operations for which the time complexities can be defined as follows:

- For any ‘select’ operation the time complexity is O(1).
- For any ‘Sort’ operation the time complexity is O(nlogn).
- For any ‘join’ operation on any sorted data is O(n).
- For any ‘where’ clause the time complexity is O(n).
- So, the total complexity can be written as: O(1) + O(nlogn) + O(n) + O(n) which boils down to O(nlogn).

Part 2:

1) **List the number of patients who had “tumor”, “leukemia” and “ALL”, respectively.**

```
select count(P_ID) from FACT_CLINICAL inner join DIM_DISEASE using(DS_ID) where
DESCRIPTION='tumor';
```

Answer: 53

```
select count(P_ID) from FACT_CLINICAL inner join DIM_DISEASE using(DS_ID) where TYPE='leukemia';
```

Answer: 27

```
select count(P_ID) from FACT_CLINICAL inner join DIM_DISEASE using(DS_ID) where NAME='ALL';
```

Answer: 13

2) **List the types of drugs which have been applied to patients with “tumor”;**

```
select distinct DIM_DRUG.TYPE from FACT_CLINICAL inner join (select P_ID from FACT_CLINICAL inner
join DIM_DISEASE using(DS_ID) where DESCRIPTION='tumor') using(P_ID) inner join DIM_DRUG
using(DR_ID) order by 1;
```

Answer:

```
TYPE
--------------------------------------------------
Drug Type 001
Drug Type 002
Drug Type 003
Drug Type 004
Drug Type 005
Drug Type 006
Drug Type 007
Drug Type 008
Drug Type 009
Drug Type 010
Drug Type 011
Drug Type 012
Drug Type 013
Drug Type 014
Drug Type 015
Drug Type 016
Drug Type 017
Drug Type 018
Drug Type 019
Drug Type 020
```

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)

```
select P_ID, S_ID, UID_COLUMN, PB_ID, EXP from FACT_MICROARRAY inner join
(select P_ID, S_ID from FACT_CLINICAL inner join
(select P_ID from FACT_CLINICAL inner join DIM_DISEASE using(DS_ID) where NAME='ALL')
using(P_ID) where S_ID is not NULL)
using(S_ID) inner join DIM_PROBE using(PB_ID) inner join FACT_GENE using(UID_COLUMN)
where CL_ID=2 and mu_id=1 order by P_ID, UID_COLUMN;
```

Answer:

```
    P_ID      S_ID UID_COLUMN     PB_ID       EXP
---------- ---------- ---------- ---------- ----------
     765    587508    2382734   91809138       172
     765    587508    2616922   54226887       159
     765    587508    4613917   17376411       154
     765    587508   20209218    5527524        94
     765    587508   27506280   51513963        72
     765    587508   37702028   27051001        37
325 rows selected
```

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)**

```
with c as (select exp,case b.NAME when 'ALL' then 'ALL' else 'OTHER' end as DISEASE from
FACT_MICROARRAY inner join

(select S_ID,a.NAME from FACT_CLINICAL inner join

(select P_ID,DIM_DISEASE.NAME from FACT_CLINICAL inner join DIM_DISEASE using(DS_ID)) a
using(P_ID) where S_ID is not NULL) b

using(S_ID) inner join DIM_PROBE using(PB_ID) inner join FACT_GENE using(UID_COLUMN)

where GO_ID=12502 order by DISEASE)

select STATS_T_TEST_INDEP(DISEASE,exp,'STATISTIC','ALL') t_statistics,
STATS_T_TEST_INDEP(DISEASE,exp) p_value from c;
```

Answer:

T_STATISTICS	P_VALUE
-1.007126777	0.314065699

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.)**

```
with c as (select exp, DISEASE from FACT_MICROARRAY inner join

(select S_ID,DISEASE from FACT_CLINICAL inner join

(select P_ID,DIM_DISEASE.NAME as DISEASE from FACT_CLINICAL inner join DIM_DISEASE using(DS_ID))
a using(P_ID) where S_ID is not NULL) b

using(S_ID) inner join DIM_PROBE using(PB_ID) inner join FACT_GENE using(UID_COLUMN)

where GO_ID=7154 and DISEASE IN ('ALL','AML','Colon tumor','Breast tumor') order by DISEASE)

SELECT STATS_ONE_WAY_ANOVA(DISEASE,exp,'F_RATIO') as
F_RATIO,STATS_ONE_WAY_ANOVA(DISEASE,exp) as p_value FROM c;
```

Answer:

F_RATIO	P_VALUE
3.138912131	0.024681501

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 $N1$ " ALL" patients and $N2$ " AML" patient. For the average correlation of the expression values between two patients with " ALL", you need first calculate $N1 \times (N1 - 1)/2$ 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 $N1 \times N2$ Person Correlations then calculate the average value.)**

Query 1:

```
with temp as (select P_ID, UID_COLUMN, DISEASE, exp from FACT_MICROARRAY inner join

(select P_ID,S_ID,DISEASE from FACT_CLINICAL inner join

(select P_ID,DIM_DISEASE.NAME as DISEASE from FACT_CLINICAL inner join DIM_DISEASE using(DS_ID))
a using(P_ID) where S_ID is not NULL) b

using(S_ID) inner join DIM_PROBE using(PB_ID) inner join FACT_GENE using(UID_COLUMN)

where GO_ID=7154 and DISEASE ='ALL' order by 1,2,3)

select avg(corr(t1.exp,t2.exp)) AVG_CORR_ALL from (select P_ID,UID_COLUMN,exp from temp) t1 inner
join

(select P_ID,UID_COLUMN,exp from temp) t2

on t1.P_ID<t2.P_ID and t1.UID_COLUMN=t2.UID_COLUMN

group by t1.P_ID,t2.P_ID;
```

Answer:

AVG_CORR_ALL
0.143544348

Query 2:

```
with temp as (select P_ID, UID_COLUMN, DISEASE, exp from FACT_MICROARRAY inner join

(select P_ID,S_ID,DISEASE from FACT_CLINICAL inner join

(select P_ID,DIM_DISEASE.NAME as DISEASE from FACT_CLINICAL inner join DIM_DISEASE using(DS_ID))
a using(P_ID) where S_ID is not NULL) b using(S_ID) inner join

DIM_PROBE using(PB_ID) inner join FACT_GENE using(UID_COLUMN)

where GO_ID=7154 and DISEASE in ('ALL','AML') order by 1,2,3)

select avg(corr(t1.exp,t2.exp)) AVG_CORR_ALL from (select P_ID,UID_COLUMN,exp from temp where
DISEASE='ALL') t1 inner join

(select P_ID,UID_COLUMN,exp from temp where DISEASE='AML') t2

on t1.UID_COLUMN=t2.UID_COLUMN group by t1.P_ID,t2.P_ID;
```

Answer:

AVG_CORR_ALL_AML
-0.003475601

Part 3:

Classification - Informative Genes:

```
with temp as (select UID_COLUMN,PB_ID,EXP,case DISEASE when 'ALL' then 'ALL' else 'OTHER' end
DISEASE from FACT_MICROARRAY inner join

(select P_ID,S_ID,DISEASE from FACT_CLINICAL inner join

(select P_ID,NAME as DISEASE from FACT_CLINICAL inner join DIM_DISEASE using(DS_ID)) using(P_ID)
where S_ID is not NULL)

using(S_ID) inner join

DIM_PROBE using(PB_ID)

order by UID_COLUMN,DISEASE,PB_ID)

select UID_COLUMN,STATS_T_TEST_INDEP(DISEASE,exp,'STATISTIC','ALL') t_statistics,
STATS_T_TEST_INDEP(DISEASE,exp) p_value from temp

group by UID_COLUMN having STATS_T_TEST_INDEP(DISEASE,exp)<0.01;
```

38 Informative Genes stored in:

```
create table DIM_Informative_ALL (UID_COLUMN number(26));
```

Prediction:

```
with temp as (select P_ID,DISEASE,UID_COLUMN,EXP from FACT_MICROARRAY inner join

(select P_ID,S_ID,case DISEASE when 'ALL' then 'ALL' else 'OTHER' end DISEASE from FACT_CLINICAL
inner join

(select P_ID,DIM_DISEASE.NAME as DISEASE from FACT_CLINICAL inner join DIM_DISEASE using(DS_ID))
a using(P_ID) where S_ID is not NULL) b

using(S_ID) inner join

DIM_PROBE using(PB_ID)

where UID_COLUMN in (select UID_COLUMN from DIM_INFORMATIVE_ALL) order by
DISEASE,P_ID,UID_COLUMN)

select p2,STATS_T_TEST_INDEP(DISEASE,correlation,'STATISTIC','ALL') t_statistics,
STATS_T_TEST_INDEP(DISEASE,correlation) p_value, case when
STATS_T_TEST_INDEP(DISEASE,correlation) < 0.01 then 'ALL' else 'Negative' end Result from

(select temp.P_ID p1,test.P_ID p2,temp.DISEASE DISEASE,corr(temp.EXP,test.EXP) correlation from temp
inner join

(select P_ID,UID_COLUMN,EXP from DIM_TEST_SAMPLE2 where UID_COLUMN in (select UID_COLUMN
from DIM_INFORMATIVE_ALL)) test

using(UID_COLUMN)

group by temp.P_ID,test.P_ID,temp.DISEASE order by 2,3)

group by p2;
```

Answer:

Patient	T_STATISTICS	P_VALUE	RESULT
1	18.47517005	3.02E-24	ALL
2	6.508247454	3.25475E-08	ALL
3	-0.289239688	0.773570518	Negative
4	19.16058032	5.93E-25	ALL
5	-3.030954965	0.003823812	ALL

Data Exploration:

While doing this project we realized that the data was often redundant in certain tables with the null values mostly occupying the tables. So, as a part of clean-up process we decided to re-organize the data and hence, came up with the following queries. However, we didn't use these tables for our queries as we didn't have the exact function requirements and weren't sure of how the data might be interpreted in the future.

FACT_CLINICAL Cleaning

```
insert into FACT_CLINICAL_CLEAN with a as (select  P_ID, DS_ID, SYMPTOM, DS_FROM_DATE,
DS_TO_DATE from FACT_CLINICAL where DS_ID is not null),

b as (select P_ID, DR_ID, DOSAGE, DR_FROM_DATE, DR_TO_DATE from FACT_CLINICAL where DR_ID is
not null),

c as (select P_ID, TT_ID, RESULT, TT_DATE from FACT_CLINICAL where TT_ID is not null),

d as (select P_ID, S_ID from FACT_CLINICAL where S_ID is not null)

select a.*,b.DR_ID,b.DOSAGE,b.DR_FROM_DATE,b.DR_TO_DATE,c.TT_ID,c.RESULT,c.TT_DATE,d.S_ID
from

a left outer join b on a.P_ID = b.P_ID

left outer join c on b.P_ID = c.P_ID and b.DR_FROM_DATE = c.TT_DATE

left outer join d on c.P_ID = d.P_ID

order by 1,8;

commit;
```

FACT_SAMPLE Cleaning

```
insert into FACT_SAMPLE_CLEAN with a as (select S_ID, MK_ID, MK_RESULT, MK_DATE, AS_ID,
AS_RESULT, AS_DATE from FACT_SAMPLE where MK_ID is not null),

b as (select distinct S_ID,TM_ID, TM_DESCRIPTION from FACT_SAMPLE)

select a.*,b.TM_ID,b.TM_DESCRIPTION from a left outer join b on a.S_ID = b.S_ID;

commit;
```