SQL Scripting Example

```
SELECT *
      FROM
      (SELECT gp6.gene AS gene,gp6.pcount
            FROM
            (SELECT DISTINCT gp.gene, count(gp.idPatient) AS pcount
                  FROM
                  (SELECT DISTINCT M.gene, P.idPatient
                        FROM patientmutations P, mutation M
                        WHERE
                              M.idMutation = P.idMutation AND
                              P.snp_quality > 30 AND
                              M.region != "intronic" AND
                              (M.1000G_2010Nov_allele_freq < 0.05 OR
                              M.1000G_2011Oct_allele_freq < 0.05)) gp
            GROUP BY gp.gene) gp6
      WHERE
            gp6.pcount > 5
      ORDER BY pcount DESC) M2
NATURAL JOIN
      (SELECT
            M.gene AS gene,
            M.dbSNP135_full,
            M.dbSNP135_common,
            M.1000G_2010Nov_allele_freq,
            M.1000G_2011Oct_allele_freq
            FROM mutation M
            WHERE
                  (M.1000G_2010Nov_allele_freq < 0.05 OR
                  M.1000G_2011Oct_allele_freq < 0.05)) M3;
```

This is the entire script as a whole. As a script, it finds the following:

Every mutation with a < 0.05 allele frequency in either 1000G column that is part of a gene which has at least 6 of the 10 patients with a mutation somewhere in that gene.

I'll try to break it down into parts so you can understand the table generation:

1)

```
…
…
                  (SELECT DISTINCT M.gene, P.idPatient
                        FROM patientmutations P, mutation M
                        WHERE
                              M.idMutation = P.idMutation AND
                              P.snp_quality > 30 AND
                              M.region != "intronic" AND
                              (M.1000G_2010Nov_allele_freq < 0.05 OR
                              M.1000G_2011Oct_allele_freq < 0.05)) gp
…
…
```

This is a subquery named “gp” where most of the initial conditions we want to rule out (such as allele frequencies, regions, etc.) The only two columns I'm extracting are “gene” and “idPatient”. The first few rows of this table would look like this:

M.gene	P.idPatient
PALMD	90
PALMD	91
MIR548AA1,MIR548D1	68

Notice that I used SELECT DISTINCT … we don't want multiple entries in the table if patient 90 has 4 separate mutations in the PALMD gene if we're counting just the genes.

2)

```
…
…
            (SELECT DISTINCT gp.gene, count(gp.idPatient) AS pcount
                  FROM
                  ( … ) gp
            GROUP BY gp.gene) gp6
…
…
```

Here is essentially the "counting" part of the script. This takes the previous subquery and merely counts the number of patients for each gene. The "GROUP BY" tells the script to organize the results by gene. This table would look like this:

gene	pcount
A1CF	2
A2M	4
A2ML1	3
A2ML1;A2ML1	1

So, 2 different idPatient values exist for gene A1CF, 4 different values for A2M, etc. The "AS" keyword merely renames the output column to "pcount" instead of "count(gp.idPatient)".

3)

```
…
…
      (SELECT gp6.gene AS gene,gp6.pcount
            FROM
            ( … ) gp6
      WHERE
            gp6.pcount > 5
      ORDER BY pcount DESC) M2
…
…
```

This next part merely sifts through the results, excluding all entries that do not have at least 6 patients with a mutation somewhere in that particular gene. The results are also sorted by the number of patients with a mutation, descending, and the entire subset is renamed "M2".

gene	pcount
FSIP2	10
PKD1L2	10
TTN	10
RBM20	10
OR4C11	9

4)

```
…
…
(SELECT
            M.gene AS gene,
            M.dbSNP135_full,
            M.dbSNP135_common,
            M.1000G_2010Nov_allele_freq,
            M.1000G_2011Oct_allele_freq
            FROM mutation M
            WHERE
                  (M.1000G_2010Nov_allele_freq < 0.05 OR
                  M.1000G_2011Oct_allele_freq < 0.05)) M3
…
…
```

At this point, we need to start connecting the gene information to individual mutation information. This section finds all individual mutations with at least one allele frequency < 0.05, for all genes, regardless of patients. We will use this table (renamed M3 in the subquery) as a sort reference table.

gene	dbSNP135_full	dbSNP135_common	1000G_2010Nov_allele_freq	1000G_2011Oct_allele_freq
PALMD	rs6667146	rs6667146	0.03	0.04
PALMD	rs35183456	rs35183456	0.02	0.02
PALMD	rs41285728	NULL	0.01	0.01
MIR548AA1,MIR548D1	NULL	NULL	NULL	0

5)

```
SELECT *
      FROM
      ( … ) M2
NATURAL JOIN
      ( … ) M3;
```

The final step is really a method of utilizing a natural join to connect the genes we want with the information we want about particular individual mutations in those genes. A natural join has to use either ready-made tables or subqueries, which is why we had to make two subtables called “M2” and “M3” to use it. It finds every match between the two tables for the column in common (in this case,

“gene”), and creates a final table, essentially appending every mutation in the table from section 4) onto every valid gene from section 3). Any mutations from section 4) that don't match a gene in section 3) [i.e. low allele frequency but not enough patient count for that gene] will not be matched and will not show up in the final table.

gene	pcount	dbSNP135_full	dbSNP135_common	1000G_2010Nov_allele_freq	1000G_2011Oct_allele_freq
FSIP2	10	rs116667549	rs116667549	0.04	0.03
FSIP2	10	rs76914767	rs76914767	0.02	0.03
FSIP2	10	rs114059375	rs114059375	0.02	0.01
FSIP2	10	rs80073781	rs80073781	0.04	0.03
FSIP2	10	rs76311269	rs76311269	0.03	0.03

I hope this helps those of you who are still new to SQL understand a bit of the thought process with table generation. If you have any questions, or aren't getting the results you want, let me know!