Database documentation

Introduction & Why create this database.

Lung cancer remains to be the main cause of cancer death worldwide. According to Cancer Statistics 2020, lung cancer has an 11.4% of incidence rate with a death rate of about 23% (Sung *et al.*, 2021). Among various lung cancer types, lung adenocarcinoma **(LUAD)** is the most common subtype of lung cancer, which comprises around 40% of all cases (Denisenko, 2018). Although some targeted therapies such as epidermal growth factor receptor **(EGFR)** are verified to achieve good efficacy in some patients, LUAD is still one of the most fatal and aggressive tumor subtypes with less than 5 years of overall survival (Bethune *et al.*, 2010). Therefore, investigation of the mechanism of LUAD and how to cure it are urgently needed.

Currently, there are many online databases for different omics data such as UCSC genome browser, Uniprot, Ensembl, and so on. However, none of these databases can directly integrate omics data about LUAD, as a result, scientists may spend much time searching their interested data. To fill this gap, 100 genes that are confirmed to express differentially in normal persons and LUAD patients are chosen to constitute this mini database. Scientists can query information on these genes, corresponding transcripts, and proteins via this mini database. In addition, scientists can find useful targets and then propose more powerful therapies for LUAD. All data in the mini database are sourced from the UCSC genome browser database before being cleaned and integrated by R programming.

Details of each table & What is the database used for.

This database consists of 8 tables, ranging from gene level to tissue level.

1. **GENE** table: One hundred differentially expressed genes are recorded in this table, each of which has a unique id *(gene_id)*. Additionally, the names of the gene *(gene_name)* and ids of the gene in the Ensembl database *(Ensembl_gene_id)* are included so that scientists can query more information on the Ensembl database. The *Expr_quantity* attribute derives from the total median gene expression level across 52 tissues and 2 cell lines obtained from 948 adult post-mortem individuals.
2. **CHROMOSOME** table: *Chrom_id* represents unique ids for 24 chromosomes, matching the *chromosome* (from chr1 to chrY) attribute. To obtain the relative length of transcripts, lengths of whole chromosomes are recorded *(chrom_size)*.
3. **GENE_INTERACTION** table: Some genes interact with other genes to perform specific functions. The disability of some interactions may cause LUAD. Each eligible *gene_id* corresponds to a set of serial numbers of interactive genes *(inter_number)* and their names *(inter_gene_name)*.

4. **TRANSCRIPT** table: One gene can have several transcripts due to alternative splicing (Ule and Blencowe, 2019). All transcripts derived from 100 genes are logged, each with a unique id *(trans_id)*. Same as above, the Ensembl transcript ids are provided to acquire more information in the Ensembl database *(Ensembl_trans_id).* This table also offers the position *(trans_start & trans_end)* of each transcript and whether it is the forward strand or reverse strand *(strand)*. As for the coding information, the position of the initiation codon *(coding_start)*, termination codon *(coding_end)*, and the number of exons *(exon_count)* are provided.
5. **TRANSCRIPT_CLASS** table: Different types of transcripts have disparate functions like coding protein, catalyzing reactions, silencing expression, and so on. The *trans_class* attribute notes the 3 main classes of transcripts, while the *trans_type* attribute documents all subtypes for every main class. Each subtype is allocated a unique id (*trans_class_id*).
6. **PROTEIN** table: The protein-coding transcripts can be translated into protein, functioning as mRNA. Each protein has its unique id *(prot_id)*, short name *(prot_short)*, and full name *(prot_full)*. Additionally, protein lengths *(prot_length)* and protein ids in the Uniprot database *(Uniprot_id)* are provided for future queries.
7. **PROTEIN_MUTATION** table: Mutations in genes may lead to protein mutations eventually, causing the disfunction of proteins and thus resulting in cancer. Each type of mutation is allocated a unique id *(mutation_id)* and the description of the mutation is logged in the *description* attribute. *Prot_id* attribute is contained in this table to connect mutations to corresponding proteins.
8. **Tissue** table: Some proteins show significant differential expression in specific tissues, therefore, the *description* attribute in the table records detailed information about this to help scientists use specific tissues for gene function studies in the future. As above, the unique id of each description *(tissue_id)* is created to facilitate queries.

Normal forms of this database (NF).

The database is normalized to 1NF, 2NF, and 3NF (Figure 1).

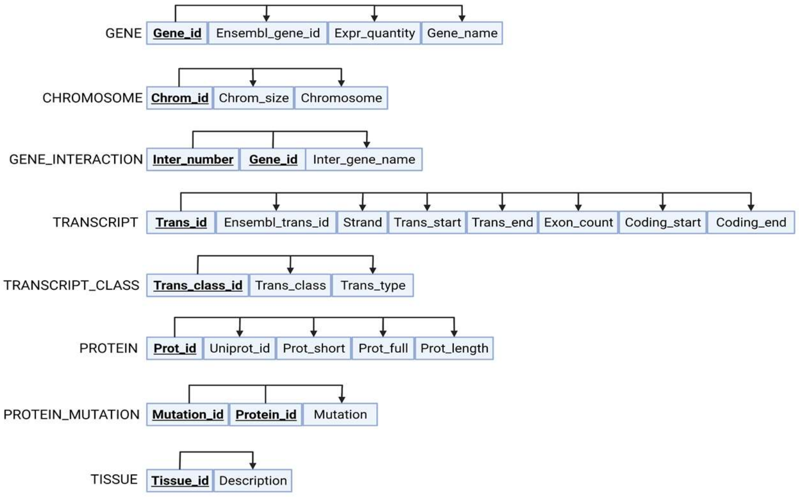


Figure 1: The schematic diagram of 3NF (created by Biorender).

Entity-Relationship models (ER).

ER model is a widely used conceptual database modeling method, which can enable the structure of the requirements and provide a way to represent these requirements graphically. The result of ER modeling is shown below (Figure 2).

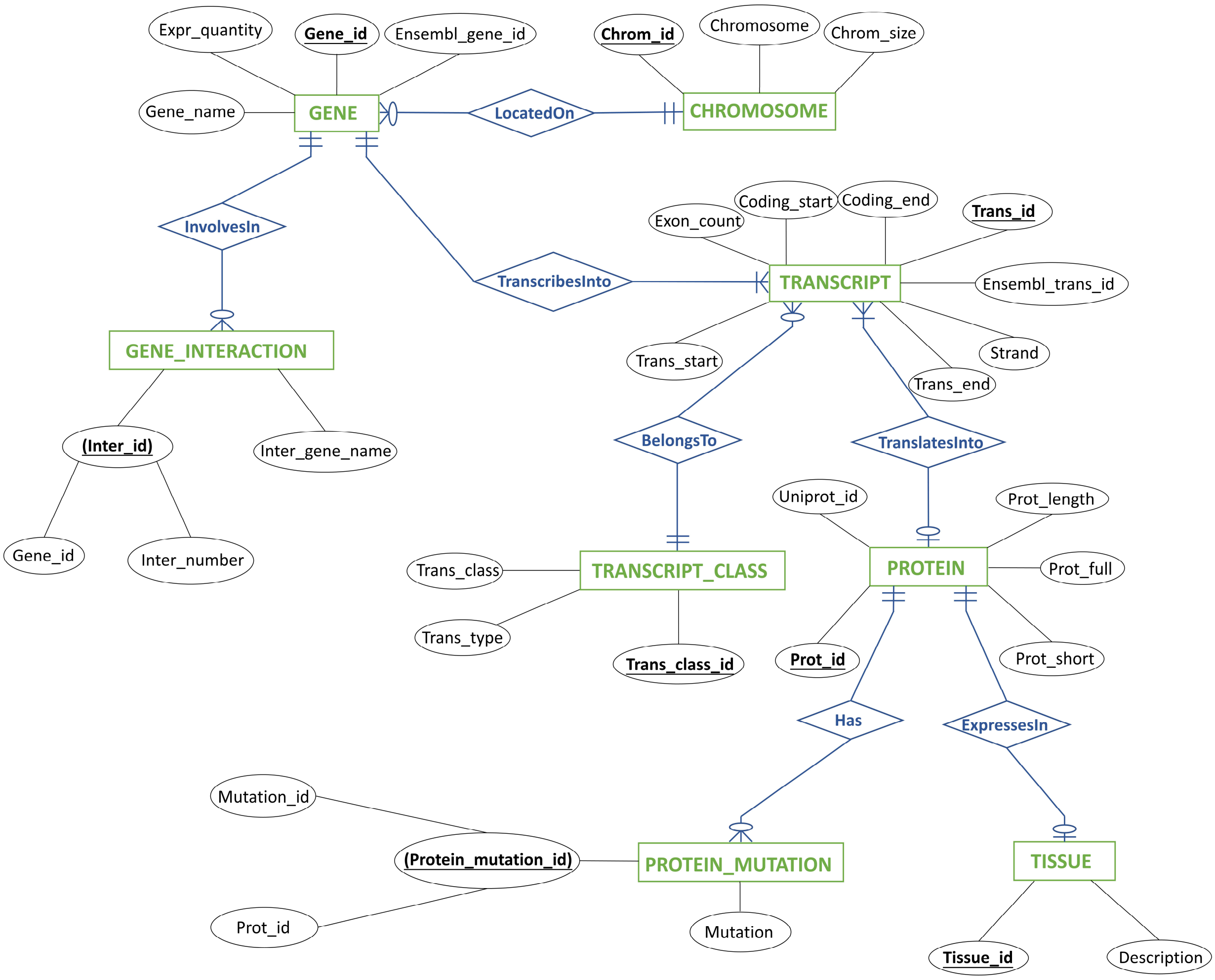
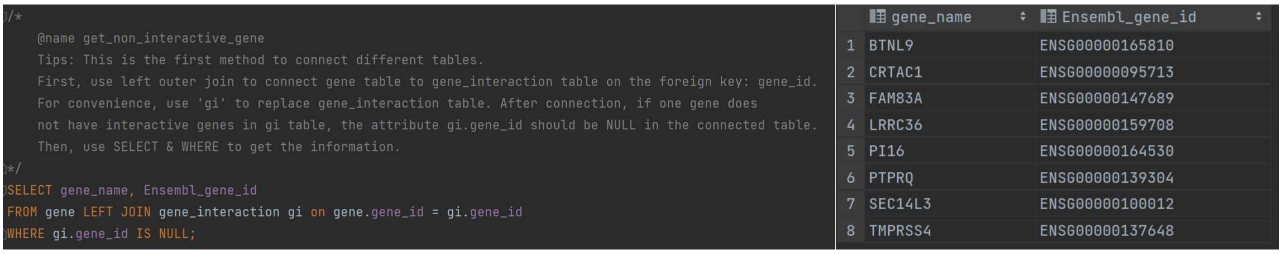


Figure 2: The ER diagram **(ERD)** of the mini database (created by PowerPoint).

Five cases to demonstrate examples of usage.

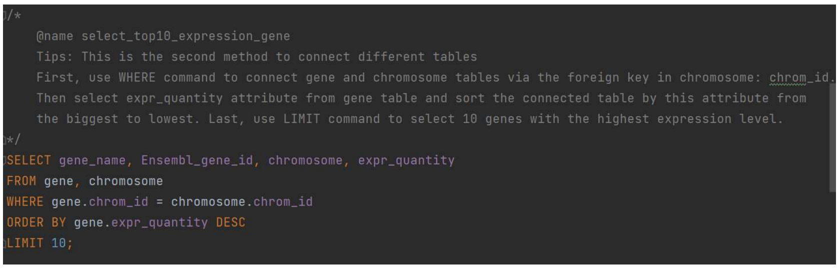
Case 1: In the mini database, some genes do not have interactive genes, which means that more research should be done on these genes to find more functions or pathways of these genes. This query can show the names and Ensembl ids of genes whose interactive genes are empty.



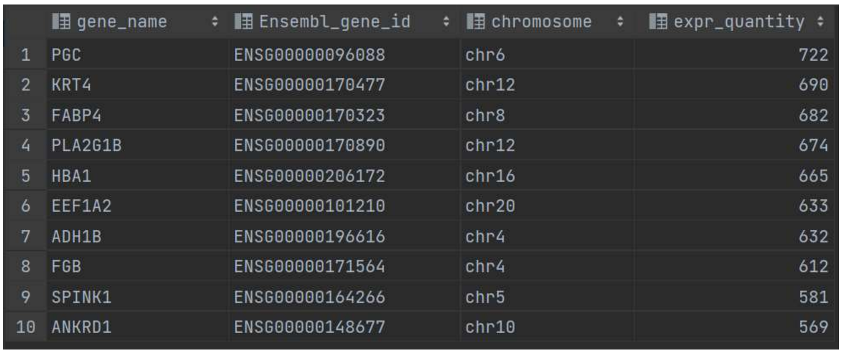
```sql
/*
    @name get_non_interactive_gene
    Tips: This is the first method to connect different tables.
    First, use left outer join to connect gene table to gene_interaction table on the foreign key: gene_id.
    For convenience, use 'gi' to replace gene_interaction table. After connection, if one gene does
    not have interactive genes in gi table, the attribute gi.gene_id should be NULL in the connected table.
    Then, use SELECT & WHERE to get the information.
*/
SELECT gene_name, Ensembl_gene_id
FROM gene LEFT JOIN gene_interaction gi on gene.gene_id = gi.gene_id
WHERE gi.gene_id IS NULL;
```

```
  gene_name   Ensembl_gene_id
1 BTNL9       ENSG00000165810
2 CRTAC1      ENSG00000095713
3 FAM83A      ENSG00000147689
4 LRRC36      ENSG00000159708
5 PI16        ENSG00000164530
6 PTPRQ       ENSG00000139304
7 SEC14L3     ENSG00000100012
8 TMPRSS4     ENSG00000137648
```

Case 2: Proteins with high expression may have significant functions. Therefore, the top ten genes with high expression are selected in this mini database. Besides the expression quantities, the names of the gene, Ensembl ids, and chromosome positions are provided.

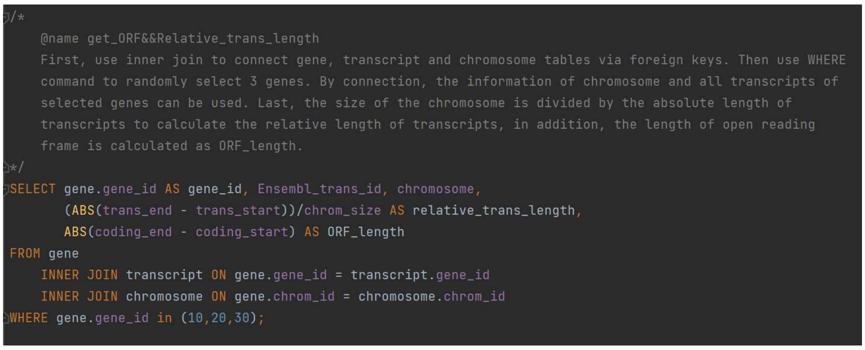


```
/*
    @name select_top10_expression_gene
    Tips: This is the second method to connect different tables
    First, use WHERE command to connect gene and chromosome tables via the foreign key in chromosome: chrom_id.
    Then select expr_quantity attribute from gene table and sort the connected table by this attribute from
    the biggest to lowest. Last, use LIMIT command to select 10 genes with the highest expression level.
*/
SELECT gene_name, Ensembl_gene_id, chromosome, expr_quantity
FROM gene, chromosome
WHERE gene.chrom_id = chromosome.chrom_id
ORDER BY gene.expr_quantity DESC
LIMIT 10;
```

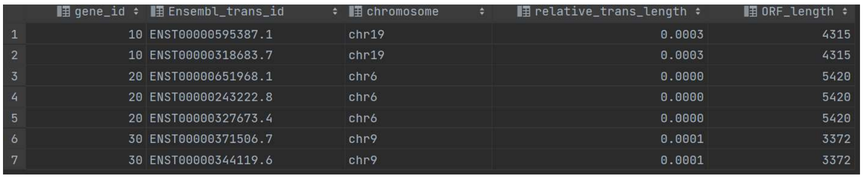


	gene_name	Ensembl_gene_id	chromosome	expr_quantity
1	PGC	ENSG00000096088	chr6	722
2	KRT4	ENSG00000170477	chr12	690
3	FABP4	ENSG00000170323	chr8	682
4	PLA2G1B	ENSG00000170890	chr12	674
5	HBA1	ENSG00000206172	chr16	665
6	EEF1A2	ENSG00000101210	chr20	633
7	ADH1B	ENSG00000196616	chr4	632
8	FGB	ENSG00000171564	chr4	612
9	SPINK1	ENSG00000164266	chr5	581
10	ANKRD1	ENSG00000148677	chr10	569

Case 3: Open reading frame **(ORF)** can express biologically active proteins that are potential therapeutic targets for cancer (Prensner *et al.*, 2021). This case shows how to get the ORF length and relative transcript length using gene_id.

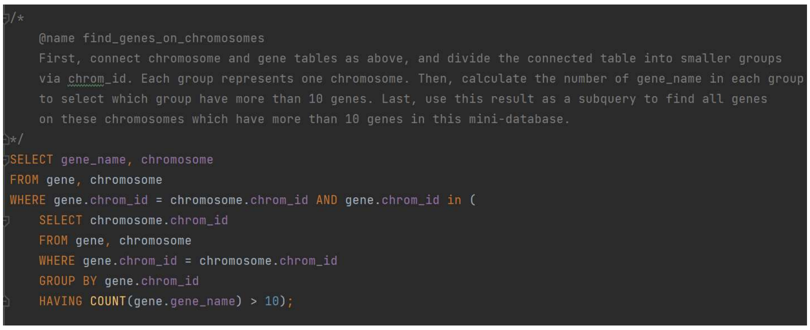


```
/*
    @name get_ORF&&Relative_trans_length
    First, use inner join to connect gene, transcript and chromosome tables via foreign keys. Then use WHERE
    command to randomly select 3 genes. By connection, the information of chromosome and all transcripts of
    selected genes can be used. Last, the size of the chromosome is divided by the absolute length of
    transcripts to calculate the relative length of transcripts, in addition, the length of open reading
    frame is calculated as ORF_length.
*/
SELECT gene.gene_id AS gene_id, Ensembl_trans_id, chromosome,
       (ABS(trans_end - trans_start))/chrom_size AS relative_trans_length,
       ABS(coding_end - coding_start) AS ORF_length
FROM gene
    INNER JOIN transcript ON gene.gene_id = transcript.gene_id
    INNER JOIN chromosome ON gene.chrom_id = chromosome.chrom_id
WHERE gene.gene_id in (10,20,30);
```

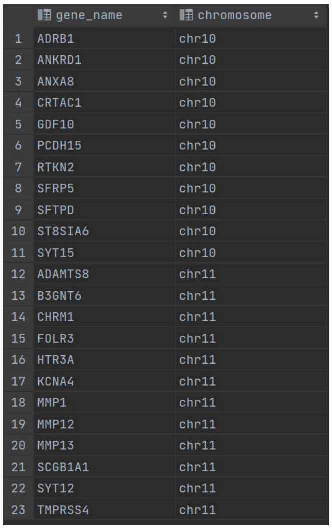


	gene_id	Ensembl_trans_id	chromosome	relative_trans_length	ORF_length
1	10	ENST00000595387.1	chr19	0.0003	4315
2	10	ENST00000318683.7	chr19	0.0003	4315
3	20	ENST00000651968.1	chr6	0.0000	5420
4	20	ENST00000243222.8	chr6	0.0000	5420
5	20	ENST00000327673.4	chr6	0.0000	5420
6	30	ENST00000371506.7	chr9	0.0001	3372
7	30	ENST00000344119.6	chr9	0.0001	3372

Case 4: Different chromosomes have different numbers of genes. Therefore, finding which chromosomes have relatively more genes is an interesting topic. From the query, I find that chr10 and chr11 have more than 10 genes respectively. The query can also list all genes located on these two chromosomes.

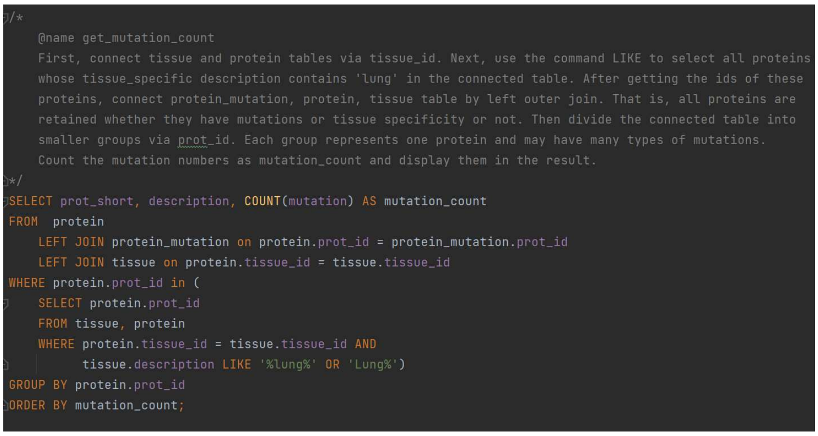


```sql
/*
    @name find_genes_on_chromosomes
    First, connect chromosome and gene tables as above, and divide the connected table into smaller groups
    via chrom_id. Each group represents one chromosome. Then, calculate the number of gene_name in each group
    to select which group have more than 10 genes. Last, use this result as a subquery to find all genes
    on these chromosomes which have more than 10 genes in this mini-database.
*/
SELECT gene_name, chromosome
FROM gene, chromosome
WHERE gene.chrom_id = chromosome.chrom_id AND gene.chrom_id in (
    SELECT chromosome.chrom_id
    FROM gene, chromosome
    WHERE gene.chrom_id = chromosome.chrom_id
    GROUP BY gene.chrom_id
    HAVING COUNT(gene.gene_name) > 10);
```

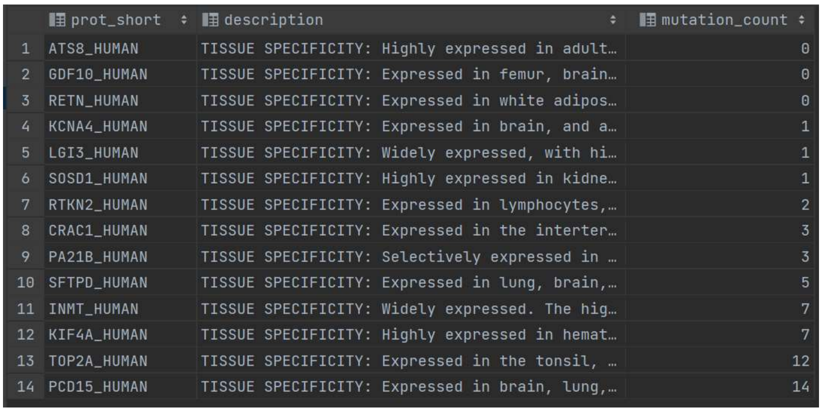


	gene_name	chromosome
1	ADRB1	chr10
2	ANKRD1	chr10
3	ANXA8	chr10
4	CRTAC1	chr10
5	GDF10	chr10
6	PCDH15	chr10
7	RTKN2	chr10
8	SFRP5	chr10
9	SFTPD	chr10
10	ST8SIA6	chr10
11	SYT15	chr10
12	ADAMTS8	chr11
13	B3GNT6	chr11
14	CHRM1	chr11
15	FOLR3	chr11
16	HTR3A	chr11
17	KCNA4	chr11
18	MMP1	chr11
19	MMP12	chr11
20	MMP13	chr11
21	SCGB1A1	chr11
22	SYT12	chr11
23	TMPRSS4	chr11

Case 5: Some proteins have tissue specificity, which means they may have particular functions in any tissue. This query list all genes that have lung specificity and the number of their mutation types. Proteins with more mutation types should be paid more consideration because they are inclined to be mutated and become disfunction.



```sql
/*
    @name get_mutation_count
    First, connect tissue and protein tables via tissue_id. Next, use the command LIKE to select all proteins
    whose tissue_specific description contains 'lung' in the connected table. After getting the ids of these
    proteins, connect protein_mutation, protein, tissue table by left outer join. That is, all proteins are
    retained whether they have mutations or tissue specificity or not. Then divide the connected table into
    smaller groups via prot_id. Each group represents one protein and may have many types of mutations.
    Count the mutation numbers as mutation_count and display them in the result.
*/
SELECT prot_short, description, COUNT(mutation) AS mutation_count
FROM  protein
    LEFT JOIN protein_mutation on protein.prot_id = protein_mutation.prot_id
    LEFT JOIN tissue on protein.tissue_id = tissue.tissue_id
WHERE protein.prot_id in (
    SELECT protein.prot_id
    FROM tissue, protein
    WHERE protein.tissue_id = tissue.tissue_id AND
          tissue.description LIKE '%lung%' OR 'Lung%')
GROUP BY protein.prot_id
ORDER BY mutation_count;
```



	prot_short	description	mutation_count
1	ATS8_HUMAN	TISSUE SPECIFICITY: Highly expressed in adult…	0
2	GDF10_HUMAN	TISSUE SPECIFICITY: Expressed in femur, brain…	0
3	RETN_HUMAN	TISSUE SPECIFICITY: Expressed in white adipos…	0
4	KCNA4_HUMAN	TISSUE SPECIFICITY: Expressed in brain, and a…	1
5	LGI3_HUMAN	TISSUE SPECIFICITY: Widely expressed, with hi…	1
6	SOSD1_HUMAN	TISSUE SPECIFICITY: Highly expressed in kidne…	1
7	RTKN2_HUMAN	TISSUE SPECIFICITY: Expressed in lymphocytes,…	2
8	CRAC1_HUMAN	TISSUE SPECIFICITY: Expressed in the interter…	3
9	PA21B_HUMAN	TISSUE SPECIFICITY: Selectively expressed in …	3
10	SFTPD_HUMAN	TISSUE SPECIFICITY: Expressed in lung, brain,…	5
11	INMT_HUMAN	TISSUE SPECIFICITY: Widely expressed. The hig…	7
12	KIF4A_HUMAN	TISSUE SPECIFICITY: Highly expressed in hemat…	7
13	TOP2A_HUMAN	TISSUE SPECIFICITY: Expressed in the tonsil, …	12
14	PCD15_HUMAN	TISSUE SPECIFICITY: Expressed in brain, lung,…	14

References

Bethune, G. *et al.* (2010) ‘Epidermal growth factor receptor (EGFR) in lung cancer: an overview and update’, *Journal of Thoracic Disease*, 2(1), pp. 48–51.

Denisenko, T.V., Budkevich, I.N. and Zhivotovsky, B. (2018) ‘Cell death-based treatment of lung adenocarcinoma’, *Cell Death & Disease*, 9(2), p. 117. doi:10.1038/s41419-017-0063-y.

Prensner, J.R. *et al.* (2021) ‘Noncanonical open reading frames encode functional proteins essential for cancer cell survival’, *Nature Biotechnology*, 39(6), pp. 697–704. doi:10.1038/s41587-020-00806-2.

Sung, H. *et al.* (2021) ‘Global Cancer Statistics 2020: GLOBOCAN Estimates of Incidence and Mortality Worldwide for 36 Cancers in 185 Countries’, *CA: a cancer journal for clinicians*, 71(3), pp. 209–249. doi:10.3322/caac.21660.

Ule, J. and Blencowe, B.J. (2019) ‘Alternative Splicing Regulatory Networks: Functions, Mechanisms, and Evolution’, *Molecular Cell*, 76(2), pp. 329–345. doi:10.1016/j.molcel.2019.09.017.