1. Single table queries (25%)

a) Write a query to count the number of codons.

```
SELECT COUNT(*) AS Codons from codons
```

displays that there are 64 codons in the table.

b) Display all the positively charged amino acids with a mass greater than 150.

```
SELECT Name FROM amino_acid_properties where Charge = 'positive' AND Molecular_mass >150
```

SELECT NAME will select the column Name from amino_acid_properties WHERE the charge column is equal to "positive" and at the same time it's corresponding molecular mass is more than 150.

c) Show all the nucleotides of the "Purine" type, sorted alphabetically by nucleotide symbol.

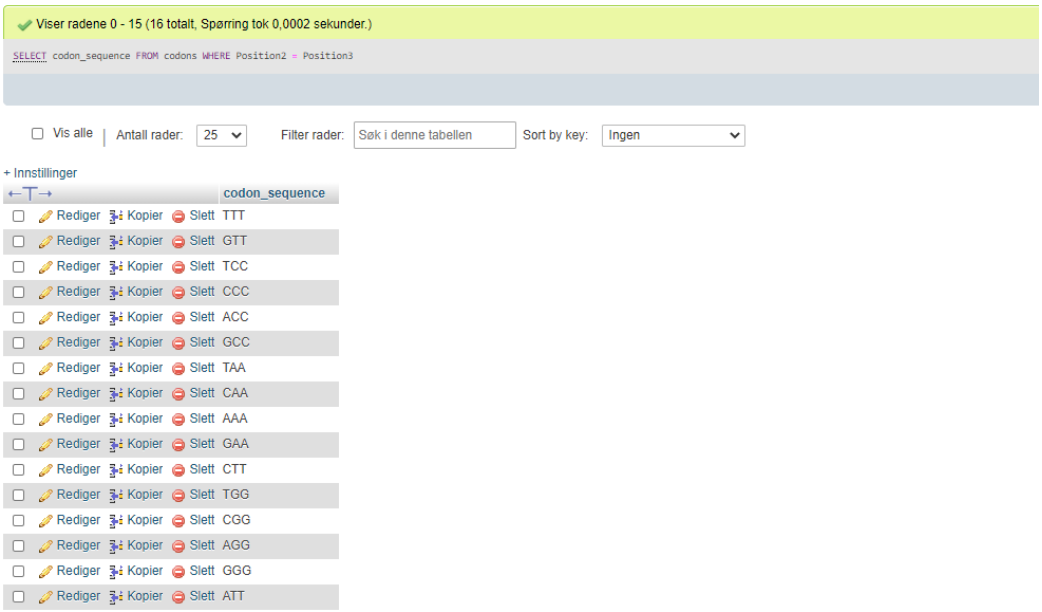
```
SELECT Name from nucleotides WHERE `Type` = 'Purine' ORDER BY Symbol
```

Select all columns from the nucleotides table where the column "Type" is equal to the string "Purine", then it will order it by the Symbol column.

d) Select all Codon_sequences that have the same nucleotide in positions 2 and 3.

```
SELECT codon_sequence FROM codons WHERE Position2 = Position3
```

Selects the column codon_sequence from the codons table where the column Position2 is the same as Position 3, thus meaning position 2 and 3 share the same nucleotide letter.

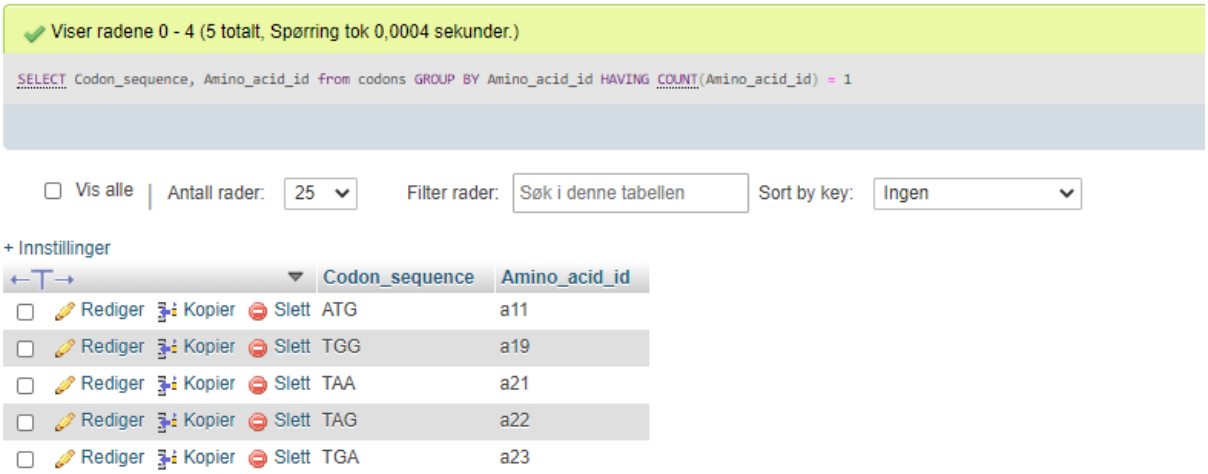


e) Show the Codon_sequences and Amino_acid_id of amino acids encoded by just a

single codon (For example the amino acid with id 'a11' is only encoded by the codon 'ATG').

```
SELECT Codon_sequence, Amino_acid_id from codons GROUP BY Amino_acid_id HAVING
COUNT(Amino_acid_id) = 1
```

Selects the columns Codon_sequence and Amino_acid_id from the codons table, then groups them by the Amino_acid_id if the Amino_acid_id only has 1 Codon_sequence. For example, there is only 1 Codon_sequence that fits with a11, that is ATG.



2. Creating tables and modifying tables (25%)

a) Create the Amino_acid_nomenclature table, select data types that you think best represent the data (see table below). Include the primary key and foreign key (be careful of the null values for the stop codons in the Name field).

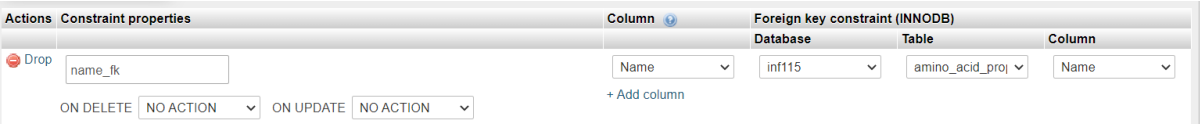
```
CREATE TABLE Amino_acid_nomenclature
(
  Amino_id VARCHAR (20),
  Symbol CHAR (1),
  Name VARCHAR (20),
  Code VARCHAR (4),
  CONSTRAINT Amino_pk PRIMARY KEY (Amino_id)
);
ALTER TABLE `codons` ADD CONSTRAINT `amino_acid_fk` FOREIGN KEY (`Amino_acid_id`) REFERENCES
`amino_acid_nomenclature`(`Amino_acid_id`) ON DELETE RESTRICT ON UPDATE RESTRICT;
```

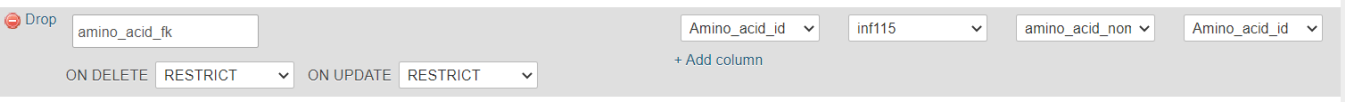
```
ALTER TABLE `amino_acid_nomenclature` ADD  CONSTRAINT `name_fk` FOREIGN KEY (`Name`) REFERENCES `amino_acid_properties`(`Name`) ON DELETE RESTRICT ON UPDATE RESTRICT;
```

Create Table creates a table named Amino_acid_nomenclature with the columns Amino_id with the data type VARCHAR with a length of 20, Symbol data type CHAR with the length 1, Name VARCHAR Length 20, Code VARCHAR Length 4,

Then adds a primary key to the column Amino_id

This primary key will be referenced by the amino_acid_id foreign key in the codons table. The last piece of code will link a foreign key name_fk in amino_acid_nomenclature to the name primary key in amino_acid_properties table.





b) Insert the values for the Amino_acid_nomenclature data from the table below (at the bottom of the document), into the table in the database.

```
INSERT INTO amino_acid_nomenclature VALUES ('a1', 'A', 'Alanine', 'Ala');

INSERT INTO amino_acid_nomenclature VALUES ('a2', 'C', 'Cysteine', 'Cys');

INSERT INTO amino_acid_nomenclature VALUES ('a3', 'D', 'Aspartic acid', 'Asp');

INSERT INTO amino_acid_nomenclature VALUES ('a4', 'E', 'Glutamic acid', 'Glu');

INSERT INTO amino_acid_nomenclature VALUES ('a5', 'F', 'Phenylalanine', 'Phe');

INSERT INTO amino_acid_nomenclature VALUES ('a6', 'G', 'Glycine', 'Gly');

INSERT INTO amino_acid_nomenclature VALUES ('a7', 'H', 'Histidine', 'His');

INSERT INTO amino_acid_nomenclature VALUES ('a8', 'I', 'Isoleucine', 'Ile');

INSERT INTO amino_acid_nomenclature VALUES ('a9', 'K', 'Lysine', 'Lys');

INSERT INTO amino_acid_nomenclature VALUES ('a10', 'L', 'Leucine', 'Leu');

INSERT INTO amino_acid_nomenclature VALUES ('a11', 'M', 'Methionine', 'Met');
```

```
INSERT INTO amino_acid_nomenclature VALUES ('a12', 'N', 'Asparagine', 'Asn');

INSERT INTO amino_acid_nomenclature VALUES ('a13', 'P', 'Proline', 'Pro');

INSERT INTO amino_acid_nomenclature VALUES ('a14', 'Q', 'Glutamine', 'Gln');

INSERT INTO amino_acid_nomenclature VALUES ('a15', 'R', 'Arginine', 'Arg');

INSERT INTO amino_acid_nomenclature VALUES ('a16', 'S', 'Serine', 'Ser');

INSERT INTO amino_acid_nomenclature VALUES ('a17', 'T', 'Threonine', 'Thr');

INSERT INTO amino_acid_nomenclature VALUES ('a18', 'V', 'Valine', 'Val');

INSERT INTO amino_acid_nomenclature VALUES ('a19', 'W', 'Tryptophan', 'Trp');

INSERT INTO amino_acid_nomenclature VALUES ('a20', 'Y', 'Tyrosine', 'Tyr');

INSERT INTO amino_acid_nomenclature VALUES ('a21', NULL, NULL, 'Stop');

INSERT INTO amino_acid_nomenclature VALUES ('a22', NULL, NULL, 'Stop');

INSERT INTO amino_acid_nomenclature VALUES ('a23', NULL, NULL, 'Stop')
```

Insert into the amino_acid_nomenclature table with the values (a,b,c,d), where a would be the first column, b is the second one, and so forth. The values in a21, a22, a23 "NULL" will insert an empty value, or nothing.

c) Add the following constraint rules to the Amino_acid_properties table:

i) Molecular_mass, should be greater than 70 and less than 210.

ii) Charge should be one of "uncharged", "positive" or "negative".

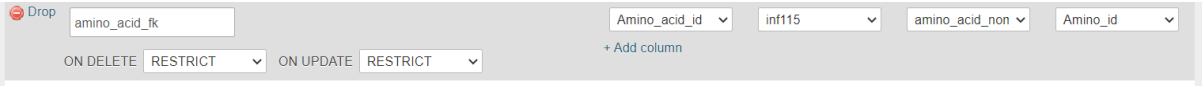
```
ALTER TABLE amino_acid_properties ADD CHECK (amino_acid_properties.Molecular_mass>70 AND
amino_acid_properties.Molecular_mass<210);

ALTER TABLE amino_acid_properties ADD CHECK (amino_acid_properties.Charge = 'uncharged' OR
amino_acid_properties.Charge = 'positive' OR amino_acid_properties.Charge = 'negative')
```

Alters table amino_acid_properties and adds a rule that the molecular_mass must be between 70 and 210, which it already is so it does not change anything.

Alters table amino_acid_properties and adds a rule that the charge column can only have the strings "uncharged", "positive", or "negative" which it already has so it also does not change anything, but it hinders you from adding things to the columns if they do not abide by the constraint rules i) and ii).

d) Add a foreign key to the Codons table referencing the amino_acid_id in the Amino_acid_nomenclature table.




```
alter table codons add constraint amino_acid_fk foreign key(amino_acid_id) references
amino_acid_nomenclature(Amino_id);
```

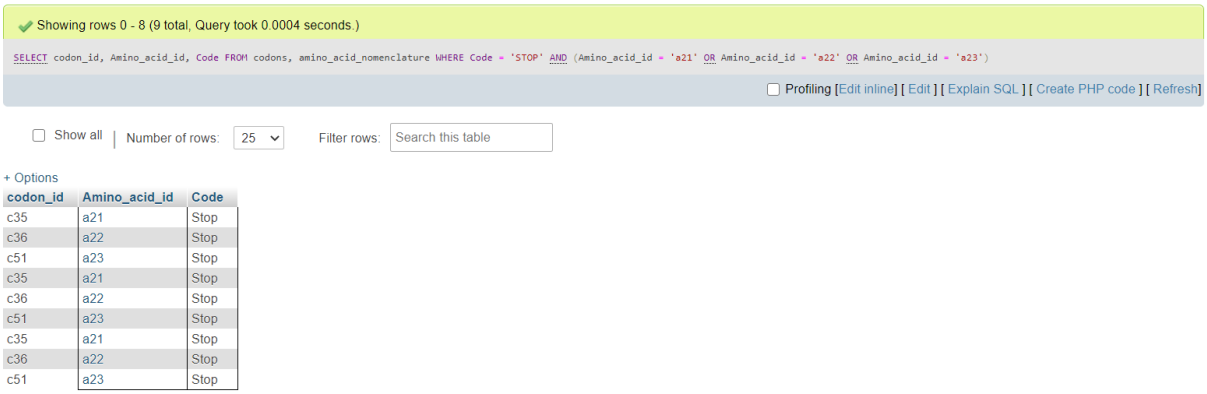
Alters the table codons to add a foreign key amino_acid_fk to amino_acid_id to then reference the foreign key to the primary key in amino_acid_nomenclature.amino_id

3. Multiple table queries (25%)

a) List all the codons encoding a stop signal (that do not code for an amino acid).

```
SELECT codon_id, Amino_acid_id, Code FROM codons, amino_acid_nomenclature WHERE Code = 'STOP'
AND (Amino_acid_id = 'a21' OR Amino_acid_id = 'a22' OR Amino_acid_id = 'a23')
```

Selects codon_id, amino_acid_id from the tables codons and amino_acid_nomenclature where the column Code has the word "Stop" in it and Amino_acid_id column is equal to a21, a22, or a23 which are the amino_acid_ids with the stop signal.



codon_id	Amino_acid_id	Code
c35	a21	Stop
c36	a22	Stop
c51	a23	Stop
c35	a21	Stop
c36	a22	Stop
c51	a23	Stop
c35	a21	Stop
c36	a22	Stop
c51	a23	Stop

b) Display all the Codon_sequence(s) that start with a nucleotide called Cytosine.

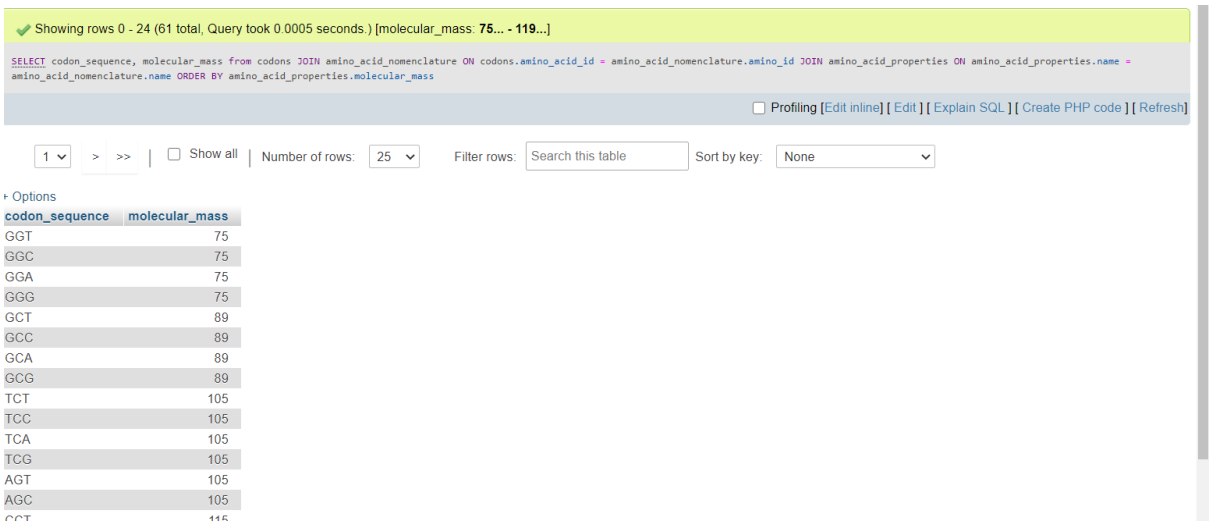
```
SELECT Codon_sequence FROM codons WHERE codons.Position1 = 'C'
```

Selects the codon_sequence column from codons table where the Position1 column in codons is equal to "C" which is the symbol for Cytosine.

c) Write a query to return the Codon_sequence for all amino acids sorted from smallest to highest molecular mass.

```
SELECT codon_sequence, molecular_mass from codons JOIN amino_acid_nomenclature ON
codons.amino_acid_id = amino_acid_nomenclature.amino_id JOIN amino_acid_properties ON
amino_acid_properties.name = amino_acid_nomenclature.name ORDER BY
amino_acid_properties.molecular_mass
```

Selects the columns codon_sequence and molecular_mass from the codons table and joins the table amino_acid_nomenclature where amino_acid_id in codons table and amino_id from amino_acid_nomenclature table is the same. Second join statement where amino_acid_properties.name and amino_acid_nomenclature.name is the same, then it sorts by the molecular mass.



codon_sequence	molecular_mass
GGT	75
GGC	75
GGA	75
GGG	75
GCT	89
GCC	89
GCA	89
GCG	89
TCT	105
TCC	105
TCA	105
TCG	105
AGT	105
AGC	105

d) Count the number of uncharged amino acids where the Codon_sequence ends with an "A".

```
SELECT COUNT(codon_sequence) AS 'Uncharged amino acids' from codons JOIN amino_acid_nomenclature ON codons.amino_acid_id = amino_acid_nomenclature.amino_id JOIN amino_acid_properties ON amino_acid_properties.name = amino_acid_nomenclature.name WHERE Codon_sequence LIKE '%A' AND amino_acid_properties.Charge = 'uncharged'
```

Selects and counts the codon sequences and will name the count "Uncharged amino acids" from the codon table. Joins the table amino_acid_nomenclature on that amino_acid_id from codons table and amino_id from amino_acid_nomenclature table is the same. Second join amino_acid_properties on that the name column from amino_acid_properties and the name column in amino_acid_nomenclature is the same. Then where the codon_sequence column ends with an A AND the Charge column from properties is equal to uncharged, it will add 1 to the count.

e) List the Codon_sequence and the amino acid Names for uncharged amino acids with a molecular mass between 130 and 150.

```
SELECT codons.Codon_sequence, amino_acid_properties.Name from codons JOIN amino_acid_nomenclature ON codons.Amino_acid_id = amino_acid_nomenclature.Amino_id JOIN amino_acid_properties ON amino_acid_properties.Name = amino_acid_nomenclature.Name WHERE amino_acid_properties.Molecular_mass >= 130 AND amino_acid_properties.Molecular_mass <= 150 AND amino_acid_properties.Charge = 'uncharged'
```

Selects the codon_sequence column from codons, Name column from amino_acid_properties, Molecular_mass from the amino_acid_properties table then join the table amino_acid_properties on the amino_acid_id from codons and amino_id from amino_acid_nomenclature are the same. Second join on that the name column from properties and nomenclature are the same. It will select the rows WHERE Molecular mass is bigger or equal to 130 AND it is less than or equal to 150 AND the charge is uncharged.

4. Advanced queries (25%)

a) Return a count of the number of nucleotides that are purines and the number that are pyrimidines.

```
SELECT
        COUNT(CASE WHEN Type LIKE 'Purine' THEN 1 END) AS Purines,
        COUNT(CASE WHEN Type LIKE 'Pyrimidine' THEN 1 END) AS Pyrimidines
FROM nucleotides
```

Selects and counts two separate counts, one for when the column Type from the nucleotides table is "Purine" and another count for when it is "Pyrimidine"

Kommandoen/spørringen er utført.

```
SELECT COUNT(CASE WHEN Type LIKE 'Purine' THEN 1 END) AS Purines, COUNT(CASE WHEN Type LIKE 'Pyrimidine' THEN 1 END) AS Pyrimidines FROM nucleotides
```

Profilering [Rediger innenfor] [Rediger] [Forklar SQL] [Create PHP code] [Oppdater]

+ Innstillinger

Purines	Pyrimidines
2	2

b) List the Amino acid symbol for all Codon_sequences composed of just a single nucleotide (for example 'AAA', 'CCC', and so on), sort these by amino acid Name.

```
SELECT codons.Amino_acid_id, codons.Codon_sequence FROM codons JOIN amino_acid_nomenclature ON codons.Amino_acid_id = amino_acid_nomenclature.Amino_id WHERE Position1 = Position2 AND Position2 = Position3 ORDER BY amino_acid_nomenclature.Name
```

Selects amino_acid_id from codons and codon_sequence from codons. Join tables amino_acid_nomenclature on that amino_acid_id from codons and amino_id from nomenclature are the same. Then it will select rows where Position1 is equal to Position2 and Position2 is equal to Position3, thus meaning all three positions share the same letter, then it will order it by the nomenclature Name column.

Showing rows 0 - 3 (4 total, Query took 0.0004 seconds.)

```
SELECT codons.Amino_acid_id, codons.Codon_sequence FROM codons JOIN amino_acid_nomenclature ON codons.Amino_acid_id = amino_acid_nomenclature.Amino_id WHERE Position1 = Position2 AND Position2 = Position3
ORDER BY amino_acid_nomenclature.Name
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

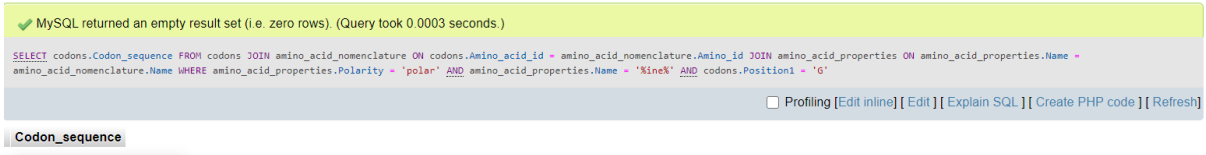
Amino_acid_id	Codon_sequence
a6	GGG
a9	AAA
a5	TTT
a13	CCC

c) Write a query to display the Codon_sequence for all the polar amino acids with a name that finishes with 'ine', where the first nucleotide in the codon is a purine.

```
SELECT codons.Codon_sequence FROM codons JOIN amino_acid_nomenclature ON codons.Amino_acid_id = amino_acid_nomenclature.Amino_id JOIN amino_acid_properties ON amino_acid_properties.Name = amino_acid_nomenclature.Name WHERE amino_acid_properties.Polarity = 'polar' AND amino_acid_properties.Name = '%ine%' AND codons.Position1 = 'G'
```

Select codon_sequence from codons. Join the table nomenclature on that the codons.amino_acid_id and the nomenclature amino_id is the same. Then join the table amino_acid_properties on that properties.Name and nomenclature.Name are the same. Then it will select rows where the Polarity is equal to "Polar" AND the Name in properties ends in -ine AND the Position1 (first nucleotide) is G (symbol for Purine).

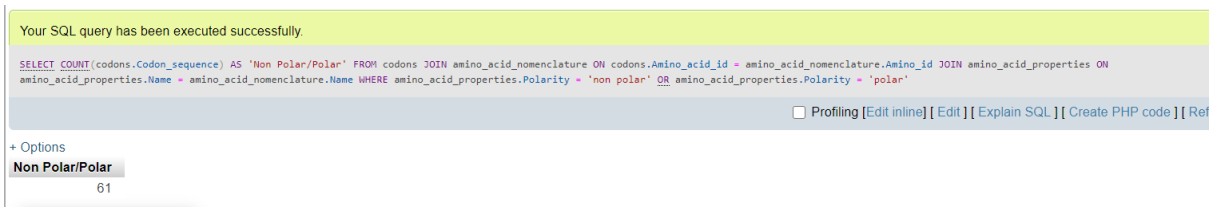
It will not list any codon sequences because there are none that meet the requirements.



d) Make a count of how many of the codons would result in polar or nonpolar amino acids.

```
SELECT COUNT(codons.Codon_sequence) AS 'Non Polar/Polar' FROM codons JOIN amino_acid_nomenclature ON codons.Amino_acid_id = amino_acid_nomenclature.Amino_id JOIN amino_acid_properties ON amino_acid_properties.Name = amino_acid_nomenclature.Name WHERE amino_acid_properties.Polarity = 'non polar' OR amino_acid_properties.Polarity = 'polar'
```

Select and count codon_sequence from codons table, call the count "Non Polar/Polar". Join the table amino_acid_nomenclature on that the amino_acid_id from codons and the amino_id from nomenclature are the same. Then join the table amino_acid_properties on that the Name column from properties and the Name column from nomenclature are the same. Then it will count for where the Polarity column in properties is either "Non polar" or "Polar".



e) Further subdivide the count in 4d, by the Charge column in the Amino_acid_properties table (to end up with a total of 4 categories polar/uncharged, polar/positive, polar/negative, nonpolar/uncharged).

```
SELECT COUNT(CASE WHEN amino_acid_properties.Polarity = 'polar' AND amino_acid_properties.Charge = 'uncharged' THEN 1 END) AS 'Polar/Uncharged', COUNT(CASE WHEN amino_acid_properties.Polarity = 'polar' AND amino_acid_properties.Charge = 'positive' THEN 1 END) AS 'Polar/Positive', COUNT(CASE WHEN amino_acid_properties.Polarity = 'polar' AND amino_acid_properties.Charge = 'negative' THEN 1 END) AS 'Polar/negative', COUNT(CASE WHEN amino_acid_properties.Polarity = 'non polar' AND amino_acid_properties.Charge = 'uncharged' THEN 1 END) AS 'NonPolar/Uncharged' FROM codons JOIN amino_acid_nomenclature ON codons.Amino_acid_id = amino_acid_nomenclature.Amino_id JOIN amino_acid_properties ON amino_acid_properties.Name = amino_acid_nomenclature.Name
```

Selects and counts when amino_acid_properties.Polarity is equal to "polar" AND the charge from properties is equal to "Uncharged". Then it adds one to the count. The count is named "Polar/Uncharged". The same is repeated for the remaining 3 categories, except in the second count we count whenever it is Polar and Positive. Third count for whenever it is polar and negative. Last count for whenever it is nonpolar and uncharged.

The count will be from the table codons joined by the amino_acid_nomenclature table on that the amino_acid_id from codons and the amino_id from nomenclature are the same. Second join amino_acid_properties on that the Name column from both properties and nomenclature are the same.