

Genetic Ancestry & Wellness Information Service

Table Of Contents

1. Business application description
2. User Types or Entities
3. Use Cases
4. Tables and Collections
5. Logical Schema
6. Database Dictionary
7. Queries for user types
8. Business metrics
9. Queries in SQL and no SQL- - triggers, indexes and views
10. Mock UI
11. Business metrics output and graph
12. Project Summary

1. BUSINESS APPLICATION DESCRIPTION

The purpose of this application is to sequence and provide DNA information to the customer along with Ancestry related knowledge mined from ancestry databases.

DNA sequencing a few years ago costed over billions of dollars and took years to get results. With the advent of Cloud computing and faster microprocessors, processing and sequencing DNA information has become much easier and cheaper. From \$1000 dollars in 2008, the price of DNA sequencing has fallen drastically to \$100 dollars today. I conceptualized a service that sequences key information from the DNA and maps it to prior ancestral databases as a commercial service to the end user via a web interface. The results are updated as and when new discoveries are made in scientific research. The information shared by the user is can be made available to their primary care doctor who is then able to suggest lifestyle changes for better outcomes.

Challenges with current testing methods

I plan to leverage a cloud providers' infrastructure offerings for our project and make it easily accessible to end users. There are currently services that do provide ancestry and wellness information. But unfortunately, most of them are caught up in legalities due to FDA regulation and are unable to provide "not- sugar coated" findings to their customers.

I plan to leverage a cloud providers' infrastructure offerings for our project and make it easily accessible to end users. This application however can be linked with your primary care doctor who can better educate the customer about their test results or recommend test or therapy routes to them .I also plan to maintain data in a format that's easily extensible and can be used a central database for research conducted by the scientist at this service or share this data with other research institutions and pharmaceutical industries that might be conducting similar research.

This project uses SQL and NoSQL databases to run this application effectively.

Functional Requirements of the service and DB design

User Account and profile information: The user is able to create an account to which is all the DNA and Ancestry information linked. A SQL database is used for this information.

DNA patterns and Results: We find patterns in the user's DNA that are known scientifically to cause predisposition to certain medical conditions. A SQL database is used for this information.

Ancestry Information: We have a database of prepopulated ancestry information based on DNA patterns. A SQL Database is used for this information

Wellness and Ancestry Research: We have NoSQL database that provides comprehensive information about the wellness and ancestry patterns for scientists to study further and draw inferences from.

Scope of the project

The project aims to detail how the database schema and model should be designed for an application that sequences DNA. Users and their results is being detailed without focusing on the scientific process of the DNA sequencing and ancestry information. The application currently examines the customer DNA to look for 35 wellness and 25 ancestry patterns. The goal is to aggregate the approved conditions and ancestry information via a web API (not described in this project). In the future, more patterns can be studied, and customer results can be updated accordingly. Also, for any deeper analysis or the understanding of their results, customers and their physicians can use the API to further understand their genome.

2. USER TYPES AND ENTITIES

Customer – A user who is interested in getting their DNA sequenced for finding patterns and in knowing their Ancestry information. They can send their saliva samples via mail.

Geneticist – The Biologist responsible for sequencing the customer DNA, analyzing and interpreting the results. The geneticist also updates, curates and manages the test results in SQL databases. Geneticists work under Scientists.

Scientist – Biological Scientists responsible for studying and finding patterns in DNA through research and application of research in the industry setting. They also look for anomalies in the customer results and further study them. They mainly use NoSQL databases. They are responsible for curating and managing NoSQL databases to make them more comprehensive and elaborate so it can be used a centralized database for scientific research

Admin – The Database administrator responsible for setting relationships and maintaining the database for IT management.

3. USE-CASES

Customer

1. Create an Account and update details
2. Create an order for DNA counselling
3. Track order details and reports
4. Send and track specimens shared
5. Get updates on more enhanced results that are made available as the database grows

Geneticist

1. Update the results for the customer
2. Triage and set markers on partial matches
3. Curate and maintain wellness and ancestry databases
4. Serve as analysts for Scientists querying from the tables

Administrator/Developer

1. Update data and maintain database tables
2. Maintain database and update user roles
3. Approve roles and authorize new users
4. Created indexes and views on tables

Scientist

1. Update and query information in databases based on research
2. Update Specimen databases with more tags and research info
3. Triage and perform decisions on matches for geneticists

4. TABLES AND COLLECTIONS

We list the different types of SQL tables and NOSQL collections used for the database design.

SQL Tables

1. **Customer**
2. **Order**
3. **Genetic_Results**
4. Genetic_Results_Patterns
5. Genetic_Results_Ancestry
6. **Genetic_Patterns**
7. **Ancestry**
8. **Employee**
9. **Scientist**
10. **Geneticist**

NoSQL Collections

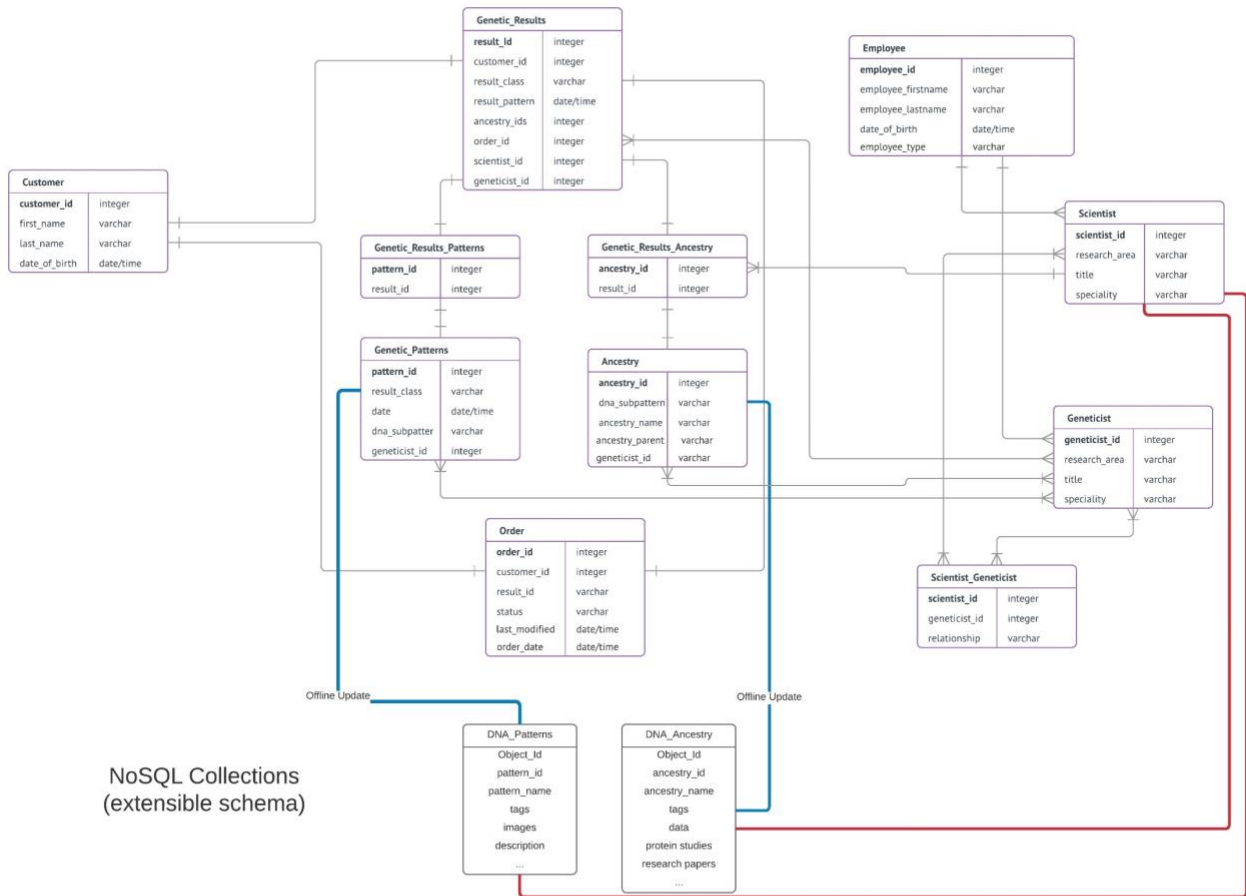
1. **DNA_Ancestry**
2. **DNA_Patterns**

Tables in Bold- Main tables

Table not in Bold- Linking tables

5. LOGICAL SCHEMA – UML MODEL

SQL Schema



Coloured Lines in Bold show interaction between SQL and NoSQL databases.

All data that is collected from and presented to the customer is in SQL including customer information, results and, geneticists and scientists who analyzed the customer DNA is stored in SQL. This data is structured and some of it is transactional (genetic results can't be manipulated) in nature, hence SQL is used here. All data used for the scientific research studied only by scientists is stored in NoSQL because of unstructured data and need for extensible schema. The collections have embedded documents like research papers and arrays like tags.

6. DATABASE DICTIONARY

SQL:

Customer Table		
Name	Type	Constraint
Customer_id	INT	Primary Key NOT NULL
Name	VARCHAR(45)	DEFAULT NULL
Dob	DATE	DEFAULT NULL
EmailAddress	VARCHAR(45)	DEFAULT NULL
Street_Address	VARCHAR(45)	DEFAULT NULL
City	VARCHAR(45)	DEFAULT NULL
State	VARCHAR(45)	DEFAULT NULL
Country	VARCHAR(45)	DEFAULT NULL
Gender	VARCHAR(45)	DEFAULT NULL

Order Table		
Name	Type	Constraint
Order_id	INT	Primary Key
Customer_id	INT	Foreign Key
Result_id	VARCHAR(45)	Foreign Key
Status	VARCHAR(45)	Default NULL

Genetic Patterns Table		
Name	Type	Constraint
Pattern_id	INT	Primary Key NOT NULL
Dna_Pattern	VARCHAR(45)	DEFAULT NULL
Employee_Id	INT	DEFAULT NULL
Pattern_Name	VARCHAR(45)	DEFAULT NULL

Genetic Results Table		
Name		Constraint
Result_id	INT	Primary Key NOT NULL
Dna_Pattern	VARCHAR(45)	DEFAULT NULL
Employee_Id	INT	DEFAULT NULL
Pattern_Name	VARCHAR(45)	DEFAULT NULL

Genetic Results and Ancestry Table		
Name	Type	Constraint
Ancestry_id	INT	Foreign Key
Results_id	INT	Foreign Key

Genetic Results and Pattern Table		
Name	Type	Constraint

Pattern_id	INT	Foreign Key
Results_id	INT	Foreign Key

Employee Table		
Name	Type	Constraint
Employee_id	INT	Primary Key NOT NULL
Name	VARCHAR(45)	DEFAULT NULL
DateJoined	DATE	DEFAULT NULL
Employee_Type	VARCHAR(45)	DEFAULT NULL

Geneticist Table		
Name	Type	Constraint
Geneticist_id	INT	Primary Key
Title	VARCHAR(45)	DEFAULT NULL
Speciality	VARCHAR(45)	DEFAULT NULL
Reporting_Manager_ID	INT	Foreign Key

Scientist Table		
Name	Type	Constraint
Scientist_id	INT	Primary Key
Title	VARCHAR(45)	Default NULL
Speciality	VARCHAR(45)	Default NULL

NoSQL:

DNA_PATTERNS : Object_ID, pattern_id, pattern_name, tags, description, protein_studies, research paper(name, date , author.....), images...etc

DNA_ANCESTRY : Object_ID, ancestry_id, ancestry_name, population, tags, description, research paper....etc

7. QUERIES FOR USER TYPES

Customer:

1. Customer wishes to see his Wellness result.
2. Customer wishes to see his Ancestry result.
3. Customer wishes to see the status of his order.

Geneticist:

1. Geneticist wants to update the customer result.
2. Geneticist wants to find out which order number he needs to process.
3. Geneticist wants to know which Scientist he is working under.

Scientist:

1. Scientist wants to update new research found on a certain disease.
2. Scientist wants to know all available information on a certain disease.
3. Scientist wants to know diseases linked to ancestry like in Caucasian male.
4. Scientist wants to know all the medical conditions diagnosed in California.
5. Scientist wants to know all the ancestry residing in California.
6. Scientist wants to insert new ancestry pattern details.

Admin:

1. Admin wants to know the employee ID of a geneticist.
2. Admin wants to create an index on Lung cancer on the request of scientist.

8. BUSINESS METRICS

1. Scientist wants to know the most common genetic disorder in men. They can use this data to find an effective cure for the disease or target pharmaceuticals who are working in this area. A bar chart can be used to show this data.
2. Scientist wants to know the most common ancestry in California. This information can be shared with governmental bodies and other institutions that require information about demographics. A pie chart can be used to represent this data.
3. The company wants to know states where least number of customers purchased the product. They can use this information to better market their product in those states. A bar chart can be used to visualize this data.
4. The company wants to know age of their customer base. This again can be used for targeted marketing. A scatter plot can be used for this metric.

9. QUERIES FOR USE CASES

Customer:

1. Fetch order results for the user “Melina Plaskett”

```
SELECT * FROM genomedb.`order`  
WHERE customer_id in  
(SELECT customer_id FROM genomedb.customer WHERE name = "Melina Plaskett")
```

order_id	customer_id	result_id	status	last_modified	order_date
908	975	566	Processing	2016-04-04	2016-11-25

2. Show all the genetic patterns and conditions for “Roch Ort”

```
SELECT * FROM genomedb.genetic_patterns WHERE pattern_id IN  
(SELECT pattern_id FROM genomedb.genetic_results_pattern WHERE result_id IN  
(SELECT result_id FROM genomedb.`order`  
WHERE customer_id IN  
(SELECT customer_id FROM genomedb.customer WHERE name = "Roch Ort"))))
```

pattern_id	dna_pattern	pattern_name	employee_id
30	Breast cancer	GAAGAAGATCACTTCGAGGCCGGCTTACTG	21
NULL	NULL	NULL	NULL

3. Show the ancestry information for “Lilas Klagge”

```
SELECT * FROM genomedb.ancestry WHERE ancestry_id IN  
(SELECT ancestry_id FROM genomedb.genetic_results_ancestry WHERE results_id IN  
(SELECT result_id FROM genomedb.`order`  
WHERE customer_id IN  
(SELECT customer_id FROM genomedb.customer WHERE name = "Lilas Klagge"))))
```

ancestry_id	ancestry_name	dna_ancestry_pattern	employee_id
5	Central Asian	ACTGGTCGCAGCGAAATTGGGTCTGGCGCT	7
NULL	NULL	NULL	NULL

4. Update a detail for a customer

UPDATE customer

SET city = "Boston"

WHERE customer_id = 1001

SELECT * *FROM* genomedb.customer *WHERE* customer_id = 100

customer_id	name	dob	emailaddress	street_address	city	state	country	gender
100	Lonnv Pitford	1993-03-16	loitford2r@webmd.com	23 Norwav Maple Drive	Boston	Connecticut	United States	Male
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Scientist queries:

1. What are the medical conditions for people in California

SELECT * *FROM* genetic_patterns *WHERE* pattern_id *IN* (*SELECT* pattern_id *FROM* genomedb.genetic_results_pattern *WHERE* results_id *IN*

(*SELECT* result_id *FROM* genomedb.`order`

WHERE customer_id *IN*

(*SELECT* customer_id *FROM* genomedb.customer *WHERE* state = "california"))))

	pattern_id	dna_pattern	pattern_name
	25	Achondroplasia	GCGGAACCCCTTCAAATGGAGACAACCATCT
	27	Antihopsholoid Svndrome	GCACAACCTCCAAGTCTATACTCAGGATAAC
	32	Colon cancer	GGGTAGTGACCGGAGTAAAGCGACTGTGCG
	33	Cri du chat	CTTCGTACCTTATCTATATCAAGGCCATG
	34	Crohn's Disease	GCGTGAACAGAATTGAATCTGACAGCTTGA
	38	Duane Svndrome	CTAATGATGCTTGATTAGCGGGTGATGTAC
	39	Duchenne Muscular Dvstrohv	TGATGAGAGCTCTCGGAGGTAATCTTCGTG
	40	Factor V Leiden Thrombophilia	CAACAGTGGAAATAGACTCTCTGATCGAAG
	44	Gaucher Disease	GTAGGGAGTTGTTAAGTTTACCGTAAACG
	45	Hemochromatosis	AAGCTCTTGACCACGAATCTGGGTATTTGA
	46	Hemophilia	GCCACCGGAAACTATTACGGTGAGAGCGGC
	49	Klinefelter svndrome	CTCCCTTCCCACTATGTACCCTGATCGGCG
	51	Mvotonic Dvstrohv	GGTATTCTCTATGAATTTGGGGATGGTGCA
	52	Neurofibromatosis	TCTGAGTTATGTACGCGTGTAACGAAGTCT
	53	Noonan Svndrome	TACGGTATTCGGGATAAAGTACTTTGGATG
	54	Osteogenesis Imperfecta	AACTATACTCAAGGGCGTAAAGTATAATCC
	57	Poland Anomalv	GCTCGAATGTCGAGTGCCAGGAGCTCCAC
	58	Porphvria	CACCCAGAATACTCGCGTAGTAGCCGAGTA

2. What are the ethnicities of people in California

```
SELECT * FROM genomedb.ancestry WHERE ancestry_id IN
(SELECT ancestry_id FROM genomedb.genetic_results_ancestry WHERE results_id IN
(SELECT result_id FROM genomedb.`order`
WHERE customer_id IN
(SELECT customer_id FROM genomedb.customer WHERE state = "california")))
```

	ancestry_id	ancestry_name	dna_ancestry_pattern
	1	East Asian	GTTACCTTATTTATGCGGTGCTAACAGTT
	2	Benin-Todo	CCTTCCTTTCCTTCATCGTACCATGCCACC
	3	Cameroon	TACAAACTCGTATTCTTTGACGCAGCAATA
	4	Caucasian	ACACACGGCCCCGATTCCGTTGGGACTAGA
	5	Central Asian	ACTGGTCGCAGCGAAATTGGGTCTGGCGCT
	12	Italian & Greek	GTAGGCGCTTGAGAGGATAGAACCATAGTT
	19	North African	TGTGACCTTTGGAAAACCTTGATATAGCTC
	20	Polynesian	GTATTTCTCTGGAAATGACAGCATCAAGGT
	21	Senegal	GCGTACCCTACGCTAGCAGGTAATGATTT
	24	Western European	TCAGAGGAAGGCATCCGAGGCGAATTCGTG

3. What disease is most common in Caucasian male?

```
db.dna_pattern.find( { dna_tags: "caucasian male" } )
```

```
{ "_id" : ObjectId("5a2259f0ef0eecd99f0661de"), "pattern_name" : "Huntington's disease", "pattern_id" : "48", "dna_pattern" :  
"AGATGTAATCCTACATAAACGGGTCTAC", "dna_tags" : [ "caucasian male", "demetia", "central nervous system disorders" ], "description" : "Huntington disease  
(HD) is a rare neurodegenerative disorder of the central nervous system characterized by unwanted choreatic movements, behavioral and psychiatric disturbances  
and dementia. Prevalence in the Caucasian population is estimated at 1/10,000-1/20,000. Mean age at onset of symptoms is 30-50 years. In some cases symptoms  
start before the age of 20 years with behavior disturbances and learning difficulties at school (Juvenile Huntington's disease; JHD). The classic sign is chorea that  
gradually spreads to all muscles. All psychomotor processes become severely retarded. Patients experience psychiatric symptoms and cognitive decline. HD is an  
autosomal dominant inherited disease caused by an elongated CAG repeat (36 repeats or more) on the short arm of chromosome 4p16.3 in the Huntingtine gene.  
The longer the CAG repeat, the earlier the onset of disease. In cases of JHD the repeat often exceeds 55. Diagnosis is based on clinical symptoms and signs in an  
individual with a parent with proven HD, and is confirmed by DNA determination. Pre-manifest diagnosis should only be performed by multidisciplinary teams in  
healthy at-risk adult individuals who want to know whether they carry the mutation or not. Differential diagnoses include other causes of chorea including general  
internal disorders or iatrogenic disorders. Phenocopies (clinically diagnosed cases of HD without the genetic mutation) are observed. Prenatal diagnosis is possible  
by chorionic villus sampling or amniocentesis. Preimplantation diagnosis with in vitro fertilization is offered in several countries. There is no cure. Management  
should be multidisciplinary and is based on treating symptoms with a view to improving quality of life. Chorea is treated with dopamine receptor blocking or  
depleting agents. Medication and non-medical care for depression and aggressive behavior may be required. The progression of the disease leads to a complete  
dependency in daily life, which results in patients requiring full-time care, and finally death. The most common cause of death is pneumonia, followed by suicide.",  
"research_papers" : { "name" : "Huntington's disease: a clinical review.", "database" : "pubmed" } }
```

4. Show all information on Parkinsons disease.

```
db.dna_pattern.find({ pattern_name:"Parkinson's disease"})
```

```
{ "_id" : ObjectId("5a225bb3ef0eecd99f0661df"), "pattern_name" : "Parkinson's disease", "pattern_id" : "55", "dna_pattern" :  
"CTCGGTAAGAGCAAAGAGTACCATGTAATA", "description" : "Parkinson disease is a progressive disorder of the nervous system. The disorder affects several regions  
of the brain, especially an area called the substantia nigra that controls balance and movement. Often the first symptom of Parkinson disease is trembling or shaking  
(tremor) of a limb, especially when the body is at rest. Typically, the tremor begins on one side of the body, usually in one hand. Tremors can also affect the arms,  
legs, feet, and face. Other characteristic symptoms of Parkinson disease include rigidity or stiffness of the limbs and torso, slow movement (bradykinesia) or an  
inability to move (akinesia), and impaired balance and coordination (postural instability). These symptoms worsen slowly over time. Parkinson disease can also affect  
emotions and thinking ability (cognition). Some affected individuals develop psychiatric conditions such as depression and visual hallucinations. People with  
Parkinson disease also have an increased risk of developing dementia, which is a decline in intellectual functions including judgment and memory. Generally,  
Parkinson disease that begins after age 50 is called late-onset disease. The condition is described as early-onset disease if signs and symptoms begin before age 50.  
Early-onset cases that begin before age 20 are sometimes referred to as juvenile-onset Parkinson dis", "genes" : [ "PARK1", "PARK2", "PARK3", "PARK4", "PINK1" ],  
"therapy" : "dopamine antagonists", "dna_tags" : [ "neurodegenerative disorder" ] }
```

5. Update new protein studies done with GAPDH and MKK7 protein.

```
db.dna_ancestry.update( { }, { $pull: { proteinstudies : { $in: [ "GAPDH", "MKK7" ] } } }, { multi: true } )
```

```
WriteResult({ "nMatched" : 10, "nUpserted" : 0, "nModified" : 8 })
```

6. Update research tags and therapy procedures for Parkinson' disease.

```
db.dna_pattern.update({ pattern_name:"Parkinson's disease"},{$addToSet: {dna_tags : {$each:["tremors","dopamine"] }},
$set: {therapy:"dopamine antagonists"}})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

7. Insert findings of new ancestry pattern “East Asian”

```
db.dna_ancestry.insert([ { name: "East Asian", dnapattern: "GTTACACCTATTATGCGGTGCTAACAGTT", proteinstudies :["MKK7", "GAPDH",
"beta-actin", "beta-tubulin", "insulin"], history : "The earliest human civilizations in East Asia developed along the Yellow River in central China.
Powerful dynasties emerged to rule the vast region, with the first emperor to consolidate rule over the six largest kingdoms being Qin Shi
Huang. During his reign, construction on the Great Wall was begun to repel nomadic invaders from the north, and standards were introduced
for currency, measures and writing. The Qin Dynasty ended with his death in 210 B.C.",research_papers : { name : "Archaic human ancestry in
East Asia", date : "Oct 31 2011", author: "Skolund"}, population:"54%"} ])
```

Geneticist

1. What are the orders allotted to the geneticist for analyzing.

```
SELECT * FROM genomedb.genetic_results WHERE employee_id = 34;
```

result_id	dna_pattern	employee_id
20	TTACAACCTCAGTTCTTCATGTTTGATACACGCAT...	34
34	AAGGAATCTGGGCTCAGCGGTAGACTGCATTGG...	34
36	GCTGCACCGTGTGCCTTGTAGATGTAATCCTACA...	34
105	GGGCTTGCACTAAGACTGTTGCGCCCGACTGGA...	34
115	CAGCCACAGGTTCTCTCTGATCCGGGTAAGCCA...	34
121	TCTTACAGTTGACACCATCACTGGATAGGTTTAT...	34
136	TAGGAGCTAAGCTAATATAATGACGCAGGTAGC...	34
247	GGACCAAGTTACTAACTACAAGGGAGCTATGTC...	34
384	CTAAATTATGGGAAAAATTCCCATCCCGTACGGG...	34
415	ATACTATATGTTGCCCTAGGATAGAGGTCTACC...	34
456	CACCTAGCCCTATCGGCCGTGAGGAAGCTACCT...	34
462	GCCCTGTTCTAACCTGTAGTTCCCATCCCCGGCG...	34
479	AGCTGTTAAATATAGCACACTGTCGGCAATTACT...	34
505	TCTCGACTGAGCTATCAATGAATGCTGCAGGCTC...	34
512	TGCGTCGGCTCCGTGTTGCTTCTCATTACTGGCC...	34
539	AACACACGCTAAAGGAAGATAAGAGTTCGTACG...	34
677	GGTACTGCAGTCGTACCTGGGCTCTCCCCAGCTC...	34
829	TGACCGTGGTTCACAATATTTTGGACAATCGGA...	34
943	GGGCAGCCGGGGGAGCGGTGTATCTTCAAGATG...	34
996	CGTGTCTGATAATGCATCAGGGGGCGTTAAACC...	34

2. Match and update the customer result to one of the existing patterns

```
INSERT INTO genomedb.genetic_results_pattern (pattern_id, results_id)
Values (25, 147)
```

```
SELECT *FROM genomedb.genetic_results_pattern WHERE pattern_id = 25
```

pattern_id	results_id
25	896
25	168
25	189
25	147

Triggers

When orders are created new, the status of orders is set to “new”, a trigger updates them to ready state so they can be processed after an order creation

```
CREATE TRIGGER status_check AFTER UPDATE ON genomedb.order
```

```
FOR EACH ROW
```

```
BEGIN
```

```
IF NEW.status = "New" THEN
```

```
SET New.status = "Ready"
```

```
END IF
```

```
END;
```

Index

```
CREATE INDEX idx_name ON customer (name);
```

```
CREATE INDEX idx_empname ON employee (name);
```

```
CREATE INDEX idx_resultID ON genetic_results (result_id);
```

Views

1. **Views for Genetic patterns.** Customers/Third party API consumers who want to read the database may want to only read the patterns and not the corresponding DNA_pattern (sensitive and private information)

```
CREATE VIEW GeneticPatterns as
```

```
SELECT pattern_id, pattern_name
```

```
FROM genetic_patterns;
```

```
SELECT * FROM genomedb.geneticpatterns;
```

pattern_id	pattern_name
25	Achondroplasia
26	Alpha-1 Antitrypsin Deficiency
27	Antiphospholipid Syndrome
28	Autism
29	Autosomal Dominant Polycystic Kidney Disease
30	Breast cancer
31	Charcot-Marie-Tooth
32	Colon cancer
33	Cri du chat
34	Crohn's Disease
35	Cystic fibrosis
36	Dercum Disease
37	Down Syndrome
38	Duane Syndrome
39	Duchenne Muscular Dystrophy
40	Factor V Leiden Thrombophilia
41	Familial Hypercholesterolemia
42	Familial Mediterranean Fever

2. Similarly, for Ancestry information, we'd only want to surface ancestry names and ids

```
CREATE VIEW AncestryPatterns as
```

```
SELECT ancestry_id, ancestry_name
```

```
FROM ancestry;
```

```
SELECT * FROM genomedb.ancestrypatterns;
```

ancestry_id	ancestry_name
1	East Asian
2	Benin-Togo
3	Cameroon
4	Caucasian
5	Central Asian
6	Eastern European
7	European Jewish
8	Finnish & Russian
9	Hunter-Gatherers of South Central Africa
10	Iberian Peninsula
11	Irish
12	Italian & Greek
13	Ivory Coast Ghana

3. Show customers from California only so this is made available for privacy and compliance reasons

```
CREATE VIEW CustomerCaliforniaInfo as
SELECT customer_id, name, city, gender
FROM genomedb.customer WHERE state = "California" ORDER BY customer_id;
```

```
SELECT * FROM genomedb.customercaliforniainfo;
```


customer_id	name	city	gender
6	Eustacia Blondell	Anaheim	Female
7	Minetta Itzkovitch	Newport Beach	Female
14	Edoard Wadhams	Fresno	Male
27	Hadlee Bissill	Stockton	Male
30	Joelly O'Neill	San Francisco	Female
31	Rance Elizabeth	Anaheim	Male
33	Lea Offer	Palo Alto	Female
46	Demetre Cheshir	Indlewood	Male
49	Thadeus Duminoos	Fullerton	Male
64	Llewellyn Scamal	Santa Barbara	Female
78	Dorthea Tremavle	Riverside	Female
85	Geri Arlett	Modesto	Male
90	Ulrich Rozanski	San Diego	Female
112	Garwood Churchman	San Diego	Male
125	Cyndia Richford	Pasadena	Female
156	Archibald Pinninton	Alhambra	Female
160	Hadleigh Greoore	Los Angeles	Male
174	Norbie Saveoe	San Francisco	Female

10. MOCK UI

Customer View:

Results page

www.gene-bio.com

 **Gene Bio**

Genetic Results page

Customer Results
Customer Name: **Lilas Klagge**
Order ID :245

Ancestry name	Percentage
Central Asian	23
East Asian	18

Refresh

Medical Conditions:


Potential Conditions based on DNA	Expressed
Crohn's Disease	No
Marfan syndrome	No

Refresh

Scientist View:

Scientist page

www.gene-bio.com

Gene Bio

Scientist front page

Search

Scientist Results

Employee Name: Ekaterina Lockhead

Title : Senior Scientist

Geneticist Name	Conditions researched
Lucinda Eveleigh	Huntington's disease
Brandy Kittley	Porphyria

Update

Top conditions

Conditions	DNA Pattern
Crohn's Disease	GCGTGAACAGAATTGAATCTGACAGCTTGA
Velocardiofacial Syndrome	GCTTCGTCTGTCAATTGACTAAAACCATGG
Marfan syndrome	ATTAGTAAGAGTTCAACGAATTGACCCTT

Update

Research Updates

New Condition	Geneticist Updated
Klinefelter syndrome	Willetta Windibank
Porphyria	Kingsley Blanckley
Trimethylaminuria	Ricky Dingle

11. BUSINESS METRICS

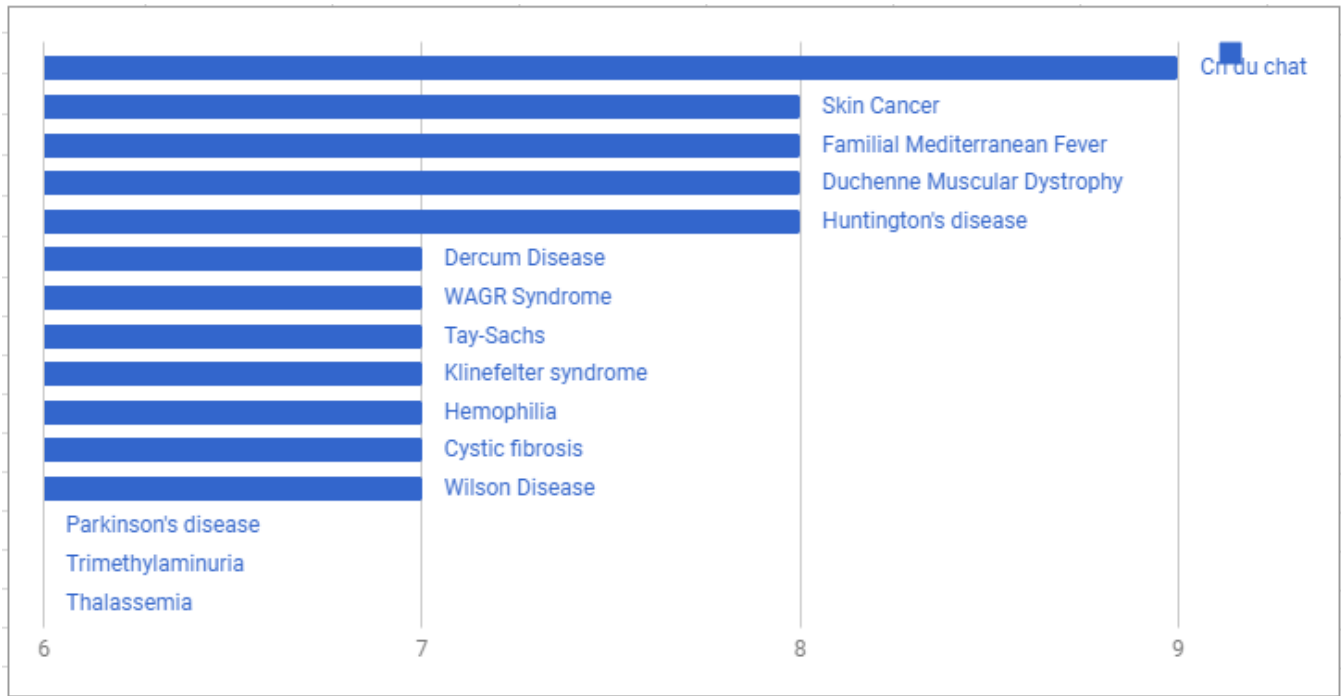
1. Top genetic patterns recorded in results

```
SELECT COUNT(dna_pattern) as count, dna_pattern FROM genetic_patterns
INNER JOIN genomedb.genetic_results_pattern ON genetic_patterns.pattern_id = genetic_results_pattern.pattern_id
GROUP BY dna_pattern
ORDER BY COUNT desc
```

Query output:

count	dna_pattern
9	Cri du chat
8	Skin Cancer
8	Familial Mediterranean Fever
8	Duchenne Muscular Dystrophy
8	Huntington's disease
7	Dercum Disease
7	WAGR Syndrome
7	Tay-Sachs
7	Klinefelter syndrome
7	Hemophilia
7	Cystic fibrosis
7	Wilson Disease
6	Parkinson's disease
6	Trimethylaminuria
6	Thalassemia
5	Factor V Leiden Thrombophilia
5	Holoprosencephaly
5	Poland Anomaly

Metrics graph:



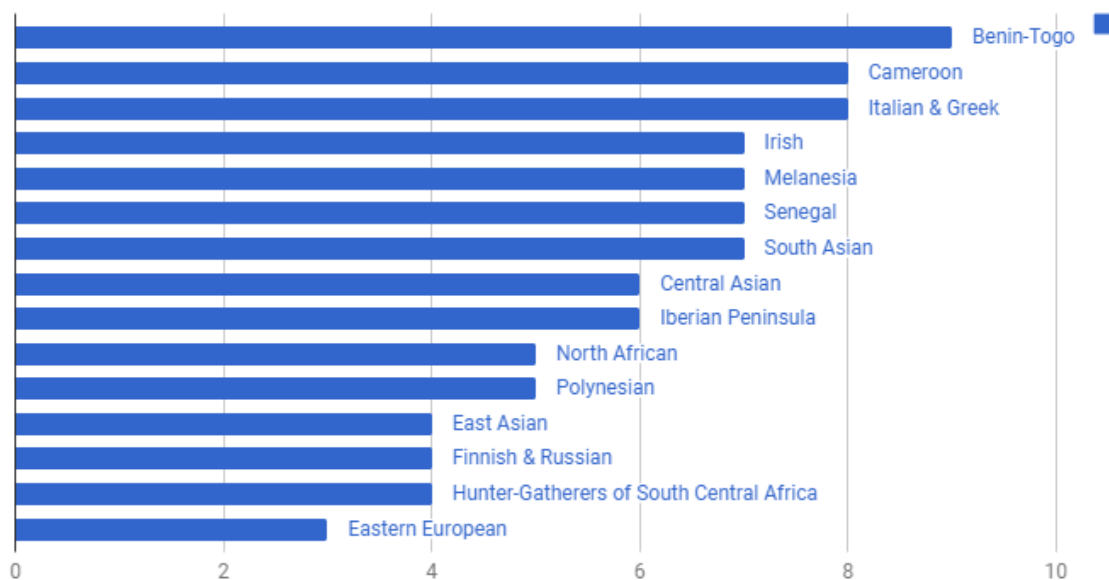
2. Top Ancestries recorded in results

```
SELECT COUNT(temp.ancestry_name) as cnt, temp.ancestry_name FROM (SELECT ancestry_name, results_id FROM ancestry
INNER JOIN genomedb.genetic_results_ancestry ON ancestry.ancestry_id = genetic_results_ancestry.ancestry_id) AS temp
JOIN genomedb.order ON temp.results_id = genomedb.order.result_id
GROUP BY temp.ancestry_name
```

Query output:

cnt	ancestry_name
9	Benin-Togo
8	Cameroon
8	Italian & Greek
7	Irish
7	Melanesia
7	Senegal
7	South Asian
6	Central Asian
6	Iberian Penin...
5	North African
5	Polynesian
4	East Asian
4	Finnish & Rus...
4	Hunter-Gathe...

Metrics Graph:



3. Customers by state who've tried our product last year

```
SELECT COUNT(state) as count, state FROM
```

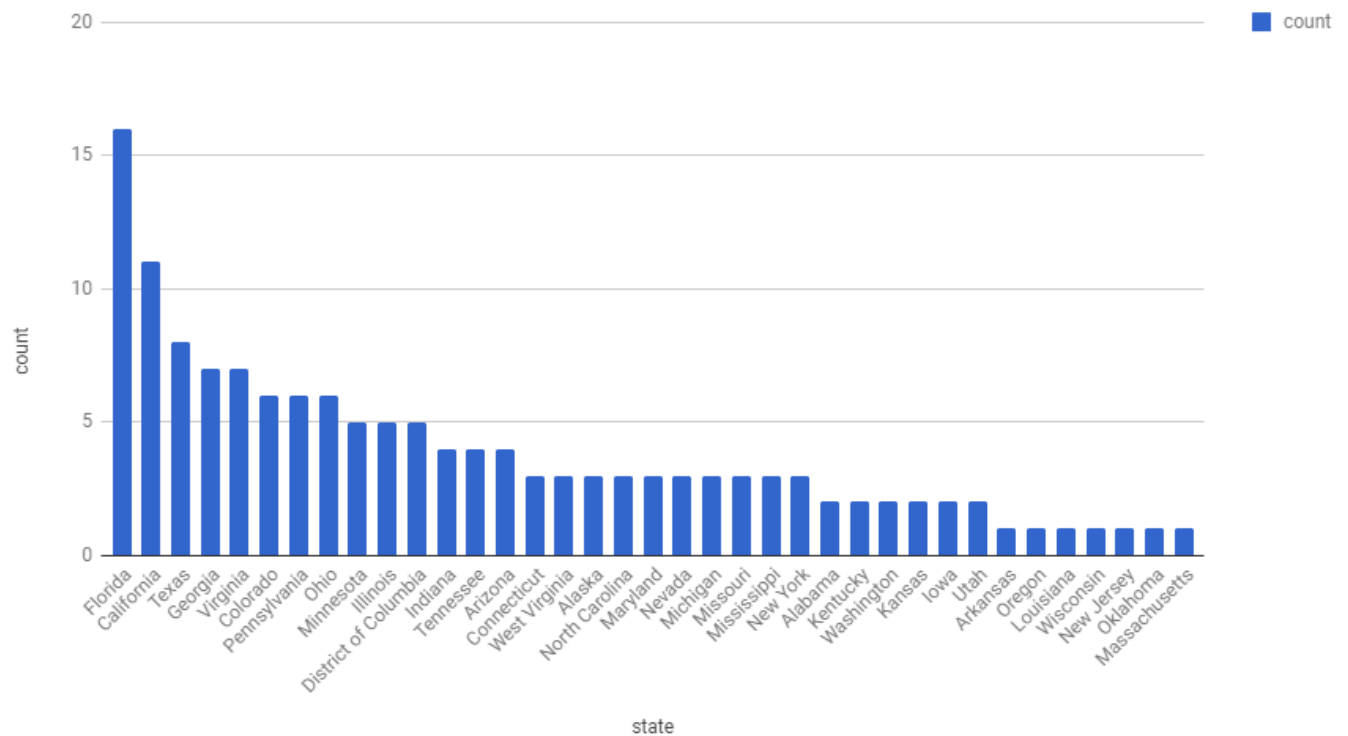
```
(SELECT customer_id, order_date FROM genomedb.order) as temporder
JOIN customer ON TempOrder.customer_id = customer.customer_id
WHERE DATE(order_date) >= '2017-1-1'
GROUP BY state ORDER BY COUNT desc
```

Query output:

count	state
16	Florida
11	California
8	Texas
7	Georgia
7	Virginia
6	Colorado
6	Pennsylvania
6	Ohio
5	Illinois
5	District of Columbia
5	Minnesota

Metrics graph:

count vs. state



12. PROJECT SUMMARY

The Genome testing project was conceptualized with the intention to help customers get knowledgeable about their genetic predispositions and ancestry. Customers can send samples of their DNA through a specimen that is then sequenced using standard procedures and then matched with a predefined set of pattern and ancestry sequences.

Figuring out how to represent a person's DNA in SQL required substantial research into industry practices and research papers. Further, generating ancestry and condition patterns that can occur in the numerous genetic results was challenging. I developed a program that would generate valid genetic patterns from 'A, C, T, G' of a specified length and modified the data using patterns and ancestry sequences. This helped me create a sizeable database to query and generate insights from.

It was a great experience building a database from ground up, designing the schema and data model, populating with data ensuring constraints were met and satisfied. And running queries to generate insightful results from this data really helped me build a deeper understanding into databases.

Hardest part of the project, Problems I ran into and how I solved them.

1. Populating data – There are no standard ways to generate DNA patterns, no data sets that match my requirement for this application. I had to do my research and understand how DNA is represented in bioinformatics projects and implement it in my database. Also, generating data such that results had genetic patterns and ancestry patterns in them to ensure results would return meaningful set of rows.
2. Understanding Genetic information – Understanding how genetic information is related for pattern matching and ancestry information took a lot of time. I had to read papers and technical reports to get an understanding of how its done.
3. Understanding SQL and NoSQL interaction and to decide which parts of the application justify using SQL or NoSQL. Given that there were some good candidates for NoSQL in my project, managing how to develop interactions between the NoSQL and SQL tables wasn't obvious and required deep thought process.

Suggestions on refining this project for the next class:

The project did help me understand how both the databases worked but I felt that I wasn't able gain a deep understanding of SQL or NoSQL. Because of the given time it wouldn't be possible to master either database management system. I would recommend giving students the choice to use SQL or NoSQL for their project so they can learn the DBMS better and can implement it more effieciently.

If I were to do this project again, the methodology I use would be:

I would generate extensive data first and build relationships between data and then populate them.

I would look into using either MySQL completely or NoSQL completely instead of a mix to avoid back and forth between the two systems.

I would research more into different operations and insights that can be derived from long DNA patterns.

I would give myself proper deadline to complete each task and try to meet them.

13 Additional Analysis

Performance: We used indexes for our most common query look up tables to increase query performance
In both MYSQL and MongoDB.

For the customer's searches without using indexes

```
EXPLAIN SELECT * FROM genomedb.customer WHERE name = "Abbot Rohlfing"
```

1168 rows needed to be processed to give the result.

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	customer	NULL	ALL	NULL	NULL	NULL	NULL	1168	10.00	Using where

However, after adding an index for the customer

```
CREATE INDEX idx_name ON customer (name);
```

The same query only had to process one row:

```
EXPLAIN SELECT * FROM genomedb.customer WHERE name = "Abbot Rohlfing"
```

Result Grid											
Filter Rows:			Export:			Wrap Cell Content:					
id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	customer	NULL	ref	idx name	idx name	138	const	1	100.00	NULL

This way, by generating the right indexes for columns that are frequently accessed and joined on will improve our SQL query performance.

Another way to improve performance is to structure the SQL table schemas so as to minimize joins and expensive operations. We modeled our schema and data in a way to minimize joins as much as possible.

Also, for set values, pre processing results and storing them in views or tables will help get faster results at query time. For example: In our DNA_Results table, matching sub patterns during query time with ancestry and genetic conditions patterns would be an expensive operation, however by creating a mapping table between Results and Patterns that are computed and populated offline, we reduced the lookup during query time to constant time.

Sharding to improve performance: NoSQL databases can be sharded and distributed across clusters as we manage more scale. With proper sharding, we can horizontally scale the databases and with proper designs for replication of the databases, we can ensure availability and prevent data loss. Given our data use for NoSQL was limited, it doesn't warrant sharding for performance reasons. However, if we were to scale our service to millions of customers and patterns, using sharding would be essential for managing performance. We could shard based on the user ids or user names to achieve uniform distribution. We can use standard industry practices and patterns to avoid bottle necks at 'hot nodes'.

Misc:

The schema was created in MYSQL and populated via MySQL workbench import from CSV files. The DML commands for tables are included inside the Database dump. Each of the table's .sql file contains the schema and the DML information for it.

Also included is a .sql file containing the DML and schema creation and constraint setting SQL commands. A simple import in MySQL workbench should recreate the whole database from these commands.