**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
13. 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.

1. **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.

1. 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**

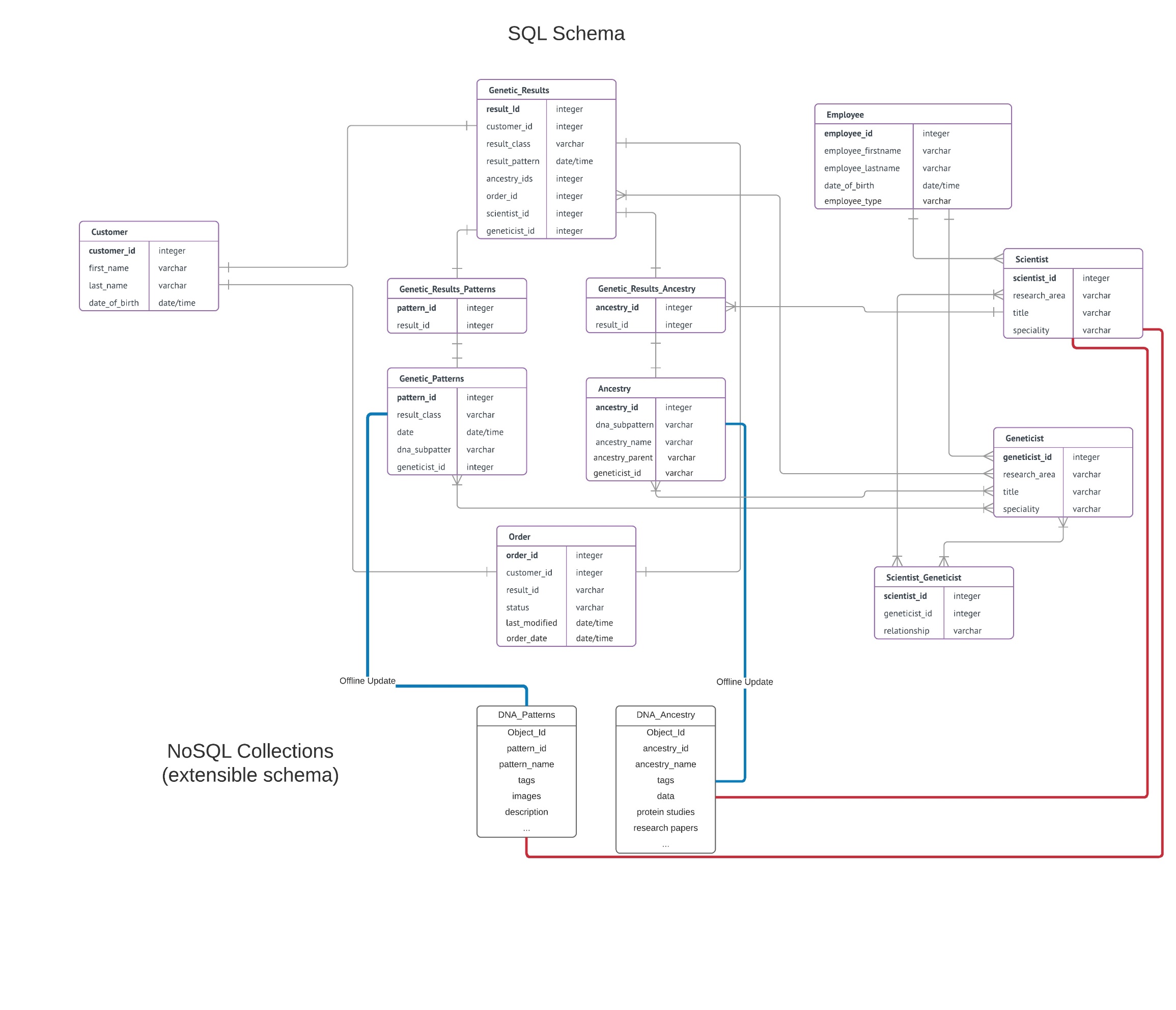
**NoSQL Collections**

1. **DNA\_Ancestry**
2. **DNA\_Patterns**

Tables in Bold- Main tables

Table not in Bold- Linking tables

1. Logical Schema – UML modeL



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.

1. 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

1. 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.
3. Business metricS
4. 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.
5. 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.
6. 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 used to visualize this data.
7. 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")*



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

*SELECT \* FROM genomedb.genetic\_patterns WHEREpattern\_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")))*



1. **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")))*



1. **Update a detail for a customer**

*UPDATE customer*

*SET city = "Boston"*

*WHERE customer\_id = 1001*

SELECT \* FROM genomedb.customer WHERE customer\_id = 100



**Scientist queries:**

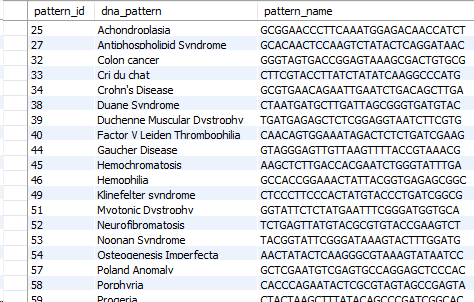
1. **What are the medical conditions for people in California**

*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")))*



1. **What are the ethnicities of people in California**

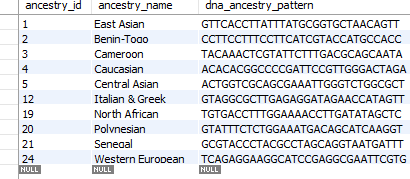
*SELECT \* FROM genomedb.ancestry WHERE ancestry\_id IN*

*(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 WHEREstate = "california")))*



1. **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" : "AGATGTAATCCTACATAAAACGGGTTCTAC", "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" } }

1. **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" : [ "neurodenerative disorder" ] }

1. **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 })

1. **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 })

1. **Insert findings of new ancestry pattern “East Asian”**

db.dna\_ancestry.insert([{ name: "East Asian", dnapattern: "GTTCACCTTATTTATGCGGTGCTAACAGTT", 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;*

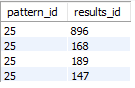


1. **Match and update the customer result to one of the existing patterns**

*INSERT INTO genomedb.genetic\_results\_pattern (pattern\_id, results\_id)*

*Values (25, 147)*

*SELECT \*FROM genomedb.genetic\_results\_pattern WHERE pattern\_id = 25*



**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**

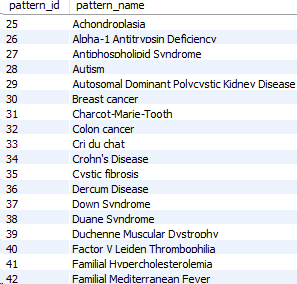
1. **Views for Genetic patterns. Customers/Third party API consumers who want to read the database may want to only read the patterns and not the corresponding DNA\_pattern (sensitive and private information)**

*CREATE VIEW GeneticPatterns as*

*SELECT pattern\_id, pattern\_name*

*FROM genetic\_patterns;*

*SELECT \* FROM genomedb.geneticpatterns;*



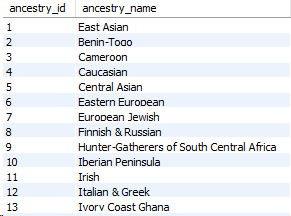
1. **Similarly, for Ancestry information, we’d only want to surface ancestry names and ids**

*CREATE VIEW AncestryPatterns as*

*SELECT ancestry\_id, ancestry\_name*

*FROM ancestry;*

*SELECT \* FROM genomedb.ancestrypatterns;*



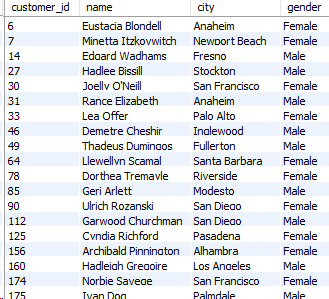
1. **Show customers from California only so this is made available for privacy and compliance reasons**

*CREATE VIEW CustomerCaliforniaInfo as*

*SELECT customer\_id, name, city, gender*

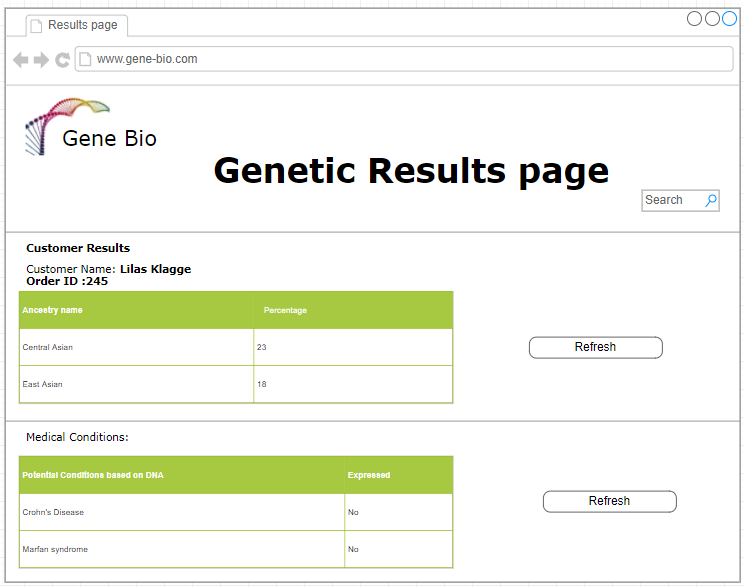
*FROM genomedb.customer WHERE state = "California" ORDER BY customer\_id;*

*SELECT \* FROM genomedb.customercaliforniainfo;*

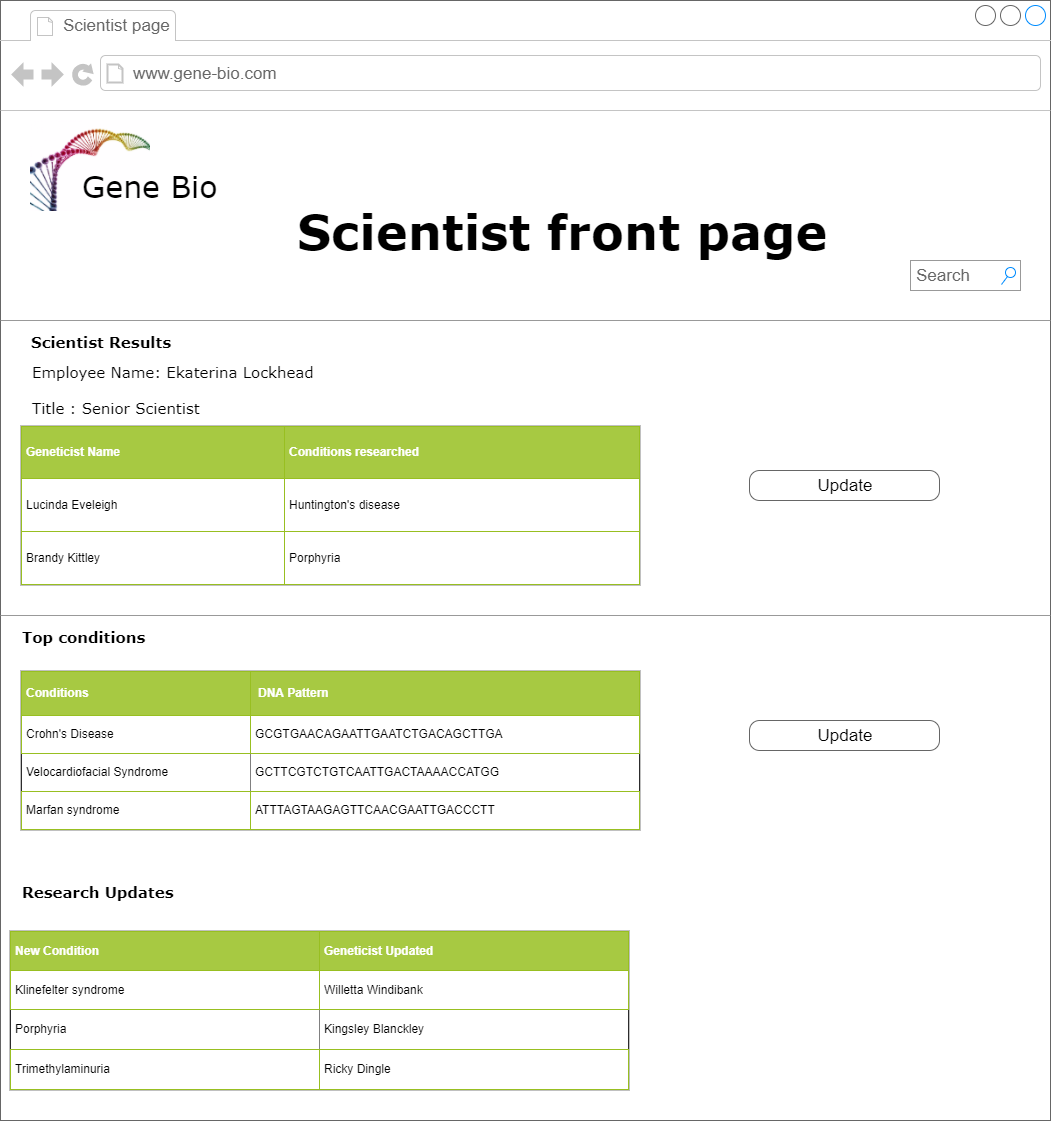


10. Mock UI

**Customer View:**



**Scientist View:**



11. Business metrics

1. **Top genetic patterns recorded in results**

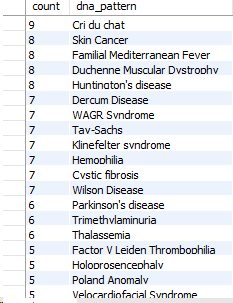
*SELECT COUNT(dna\_pattern) as count, dna\_pattern FROM genetic\_patterns*

*INNER JOIN genomedb.genetic\_results\_pattern ON genetic\_patterns.pattern\_id = genetic\_results\_pattern.pattern\_id*

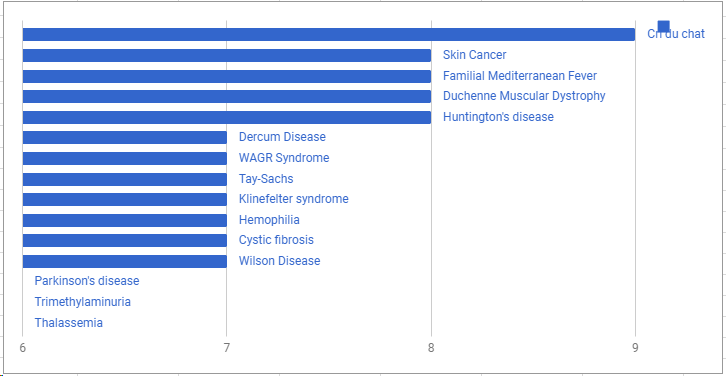
*GROUP BY dna\_pattern*

*ORDER BY COUNT desc*

**Query output:**



**Metrics graph:**



1. **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*

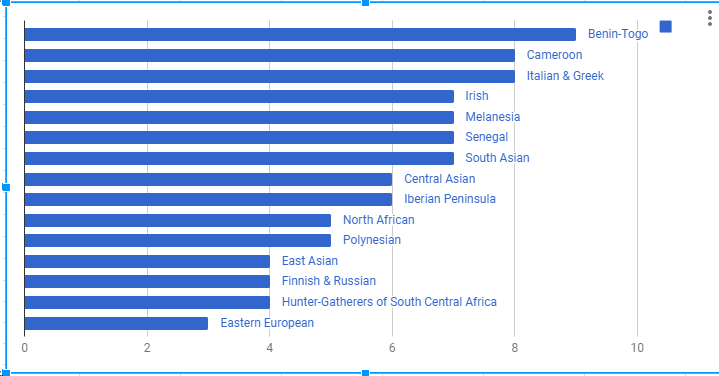
*JOIN genomedb.order ON temp.results\_id = genomedb.order.result\_id*

*GROUP BY temp.ancestry\_name*

**Query output:**



**Metrics Graph:**



1. **Customers by state who’ve tried our product last year**

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

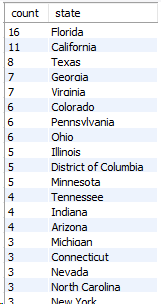
*(SELECT customer\_id, order\_date FROM genomedb.order) as temporder*

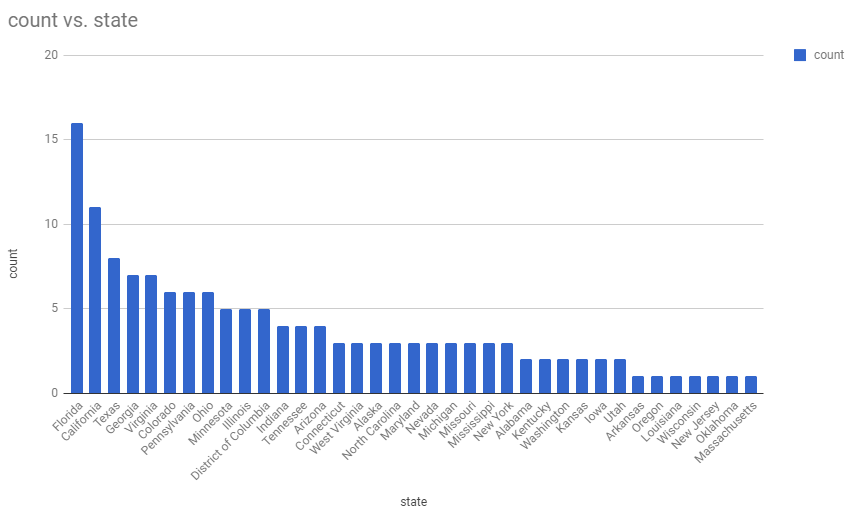
*JOIN customer ON TempOrder.customer\_id = customer.customer\_id*

*WHERE DATE(order\_date) >= '2017-1-1'*

*GROUP BY state ORDER BY COUNT desc*

**Query output:**



**Metrics graph:**

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.

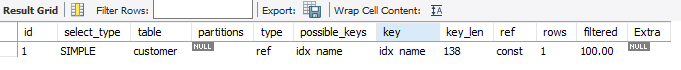


However, after adding an index for the customer

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"



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.