Technical Group Report for Data Storage Solutions

CA-1

Name: Ajit Singh

Student No: 10556972

Name: Basit Halim

Student No: 10558407

Name: Arun Surepally

Student No: 10568319

Contents

[1. Introduction 2](#_Toc37316268)

[1.1. Reasons for selecting the subject area AND DATA 2](#_Toc37316269)

[1.2. Vision and Goals 2](#_Toc37316270)

[1.3. Key StakeHolders 2](#_Toc37316271)

[1.4. Business requirements 2](#_Toc37316272)

[1.5. Any other sections 3](#_Toc37316273)

[2. SCHEMA 3](#_Toc37316274)

[3. ETL 5](#_Toc37316275)

[4. VISUALIZATIONS AND REPORTS 10](#_Toc37316276)

[4.1. Visualizations 10](#_Toc37316277)

[4.2. Reports 13](#_Toc37316278)

[5. Graph Databases 17](#_Toc37316279)

[5.1. COMAPRISON to realtional databases 20](#_Toc37316280)

[6. Conclusion 2](#_Toc37316281)3

[7. Bibliography 2](#_Toc37316282)4

[Appendix A – VISUALIZATIONS Code 2](#_Toc37316283)5

[Appendix B – Neo 4J code 2](#_Toc37316284)6

# 1. Introduction

For this project Amazon Online Retail Source system is taken into Account. This system is having a database schema consisting of Customer table with personal details, Orders details, Order items details, Product information with pricing details, Category and Department details.

Reference for the Source system:

https://github.com/dgadiraju/data/tree/master/retail\_db

## 1.2. Reasons for selecting the subject area AND DATA

Amazon is a top e-commerce company among all the Businesses working in the same domain. Hence, It possess the data that must be analyze, such as orders details data, Customers personal details, Product details etc. then use it as an information in decision making for the organization.

## 1.3. Vision and Goals

Providing the Business insights such as Total customer expenditure in a month and order statuses in between dates, popular purchases category, High demand products of the Department etc. with the accurate and up to date data to generate strategic decisions..

## 1.4. Key StakeHolders

**Business End:**

Business Subject Matter Expert, The company (AMAZON) or other related companies selling similar products, marketing agencies

**I.T. End:**

System Owner

## 1.5. Business requirements

Generate reports for the customer based on the following requirement:

* Individual Customer expenditure on the basis of Categories.
* Customers with their Order statuses linked with various Departments
* Category wise quantity and price subtotal for each customer
* Date wise order statuses for each Customer.

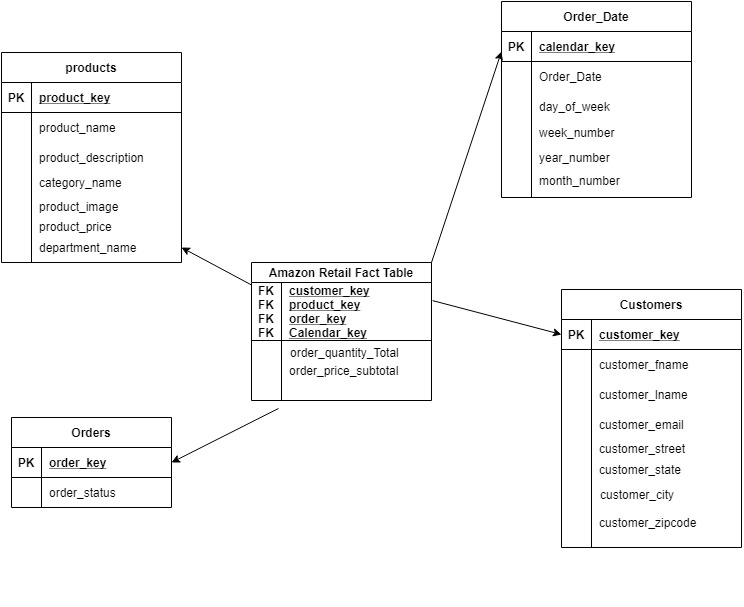
Generate visuals for the customer based on the following requirement:

* Bar Chart representation of various products by their order status and ordered quantity.
* Circle View representation of Customer expenditure by Department.
* Pie Chart representation of most popular purchase product Categories.
* Horizontal Bar representation of Total expenditure by Customer for a given date

## 1.6. Any other sections

# 2. SCHEMA

* Include schema and discuss the reasons for the design.



**Choose the Business process:**

Star schema has been built to describe the Busines process for the instance of finding Order purchase situation such as Top selling products, Per customer total expenditure for a given month, Most popular selling product category or department.

**Declare the grain:**

Declaring the exact model what properties are going to require for dimension and fact table to be able to deliver the Business requirement. Customers, products, Order details, Date information and price subtotal are vital information required to meet them. For monthly rexpenditure of Customers we need Order details and Customer details. For Order status in a given date we need Calendar table, Order details. For Top buying products we need to calculate purchase subtotal calculation in a fact table and Product details dimensions.

**Identify the dimensions:**

Define the dimensions of the model based on the grain. Dimension tables required for the central processes are Customer details, Product details, Order details, Calendar details.

**Identify the Fact attributes:**

Define the Fact tables of the model based on the grain. Calculating sum of quantity for a customer of same products and price.

# 3. ETL

Explain the procedure. Include screen shots (if any). Include code.During ETL process, data from the various source tables including Customers, Orders, Order items, Products, Category and Departments will be Extracted and integrated into data warehouse in SQL server under the Database name AMAZON\_ORDERS\_DW. In the extraction process all the relevant data is identified and retrieved from the source Database. The role of transformation is to cleanse the data and integrated different source tables to the defined table in data warehouse. Meanwhile, loading is a process to physically move the data from operational system to data warehouse.

ETL process for Products dimension table:

Control Flow has been created for each Data warehouse table as a part of ETL process.

ADO net source

Query to Extract the data from source tables, Product, department and category*:-*

*SELECT Category.category\_name, Departments.department\_name, Products.product\_id, Products.product\_name, Products.product\_price, Products.product\_image*

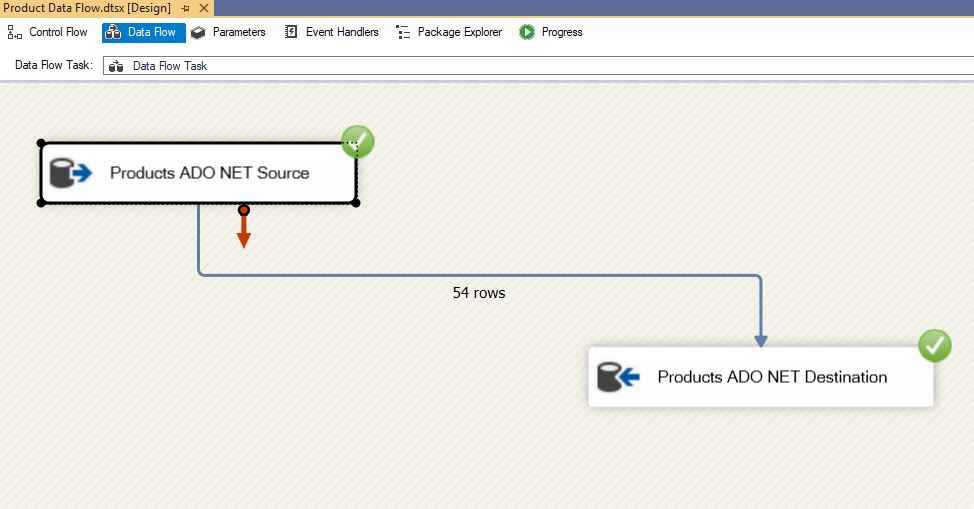
*FROM Category*

*INNER JOIN*

*Departments ON Category.category\_department\_id = Departments.department\_id INNER JOIN*

*Products ON Category.category\_id = Products.product\_category\_id*

Transformed and Loaded into Product dimension table in Warehouse Database(See screenshot below):



ETL process for Customers dimension table:

Query to Extract the data from source table, Customers, Orders*:-*

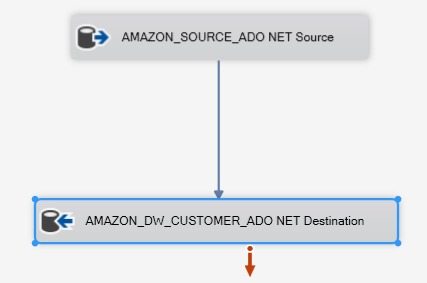
*SELECT distinct(Customers.customer\_id), Customers.customer\_fname, Customers.customer\_lname, Customers.customer\_email, Customers.customer\_password, Customers.customer\_street, Customers.customer\_city,*

*Customers.customer\_state, Customers.customer\_zipcode*

*FROM Customers INNER JOIN*

*Orders ON Customers.customer\_id = Orders.order\_customer\_id*

Transformed and Loaded into Customers dimension table in Warehouse Database (See screenshot below):



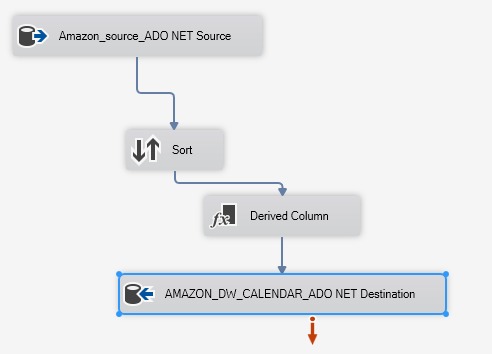
ETL process for Calendar dimension table:

Query to Extract the data from source table, Orders*:-*

*SELECT order\_date, date\_id*

*FROM Orders*

Transformed and Loaded into Customers dimension table in Warehouse Database(See screenshot below):



ETL process for Orders dimension table:

Query to Extract the data from source table, Customers*:-*

*SELECT Orders.order\_id, Orders.order\_status*

*FROM Orders INNER JOIN*

*Customers ON Orders.order\_customer\_id = Customers.customer\_id*

Transformed and Loaded into Customers dimension table in Warehouse Database(See screenshot below):



ETL process for Amazon Fact table:

Query to Extract the data from dimension tables*:-*

*SELECT Customers.customer\_id, Products.product\_id, Orders.order\_id, Orders.date\_id, Orders.order\_date, MAX(Orders\_items.order\_item\_quantity \* Orders\_items.order\_item\_product\_price) AS order\_price\_subtotal,*

*SUM(Orders\_items.order\_item\_quantity) AS order\_quantity\_total*

*FROM Orders INNER JOIN*

*Customers ON Orders.order\_customer\_id = Customers.customer\_id INNER JOIN*

*Orders\_items ON Orders.order\_id = Orders\_items.order\_item\_order\_id INNER JOIN*

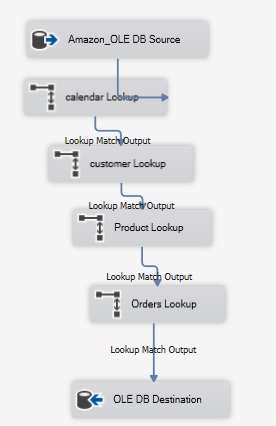
*Products ON Orders\_items.order\_item\_product\_id = Products.product\_id CROSS JOIN*

*Departments*

*GROUP BY Customers.customer\_id, Products.product\_id, Orders.order\_id, Orders.date\_id, Orders.order\_date*

*ORDER BY Customers.customer\_id*

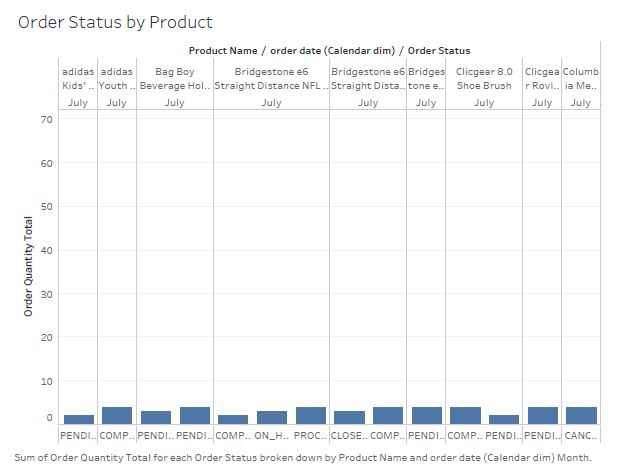
Transformed and Loaded into Customers dimension table in Warehouse Database(See screenshot below):



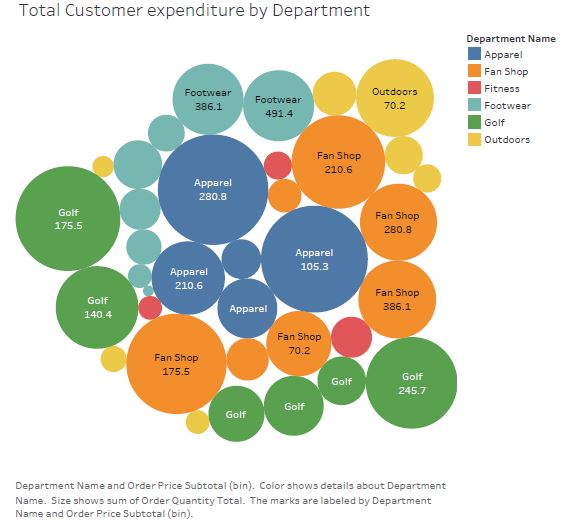
# 4. VISUALIZATIONS AND REPORTS

## 4.1. Visualizations

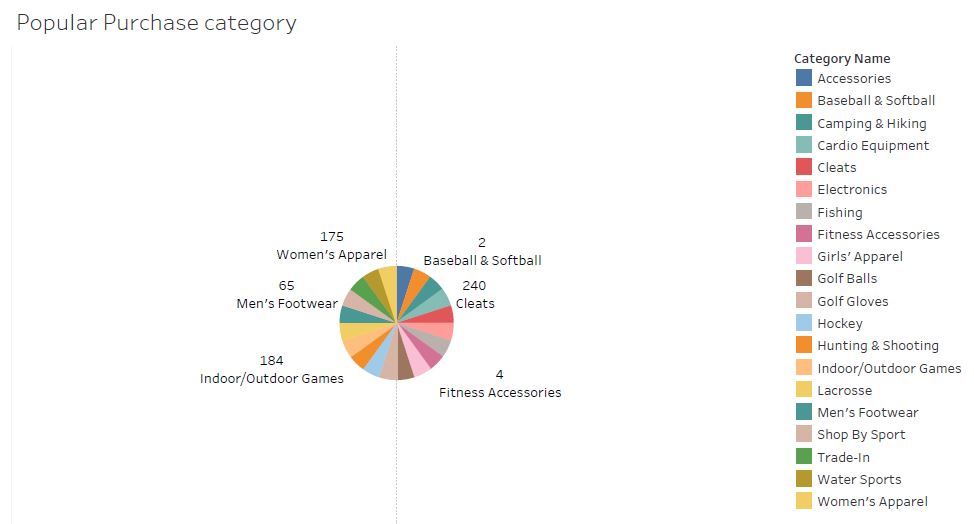
* Bar Chart representation of various products by their order status and ordered quantity.



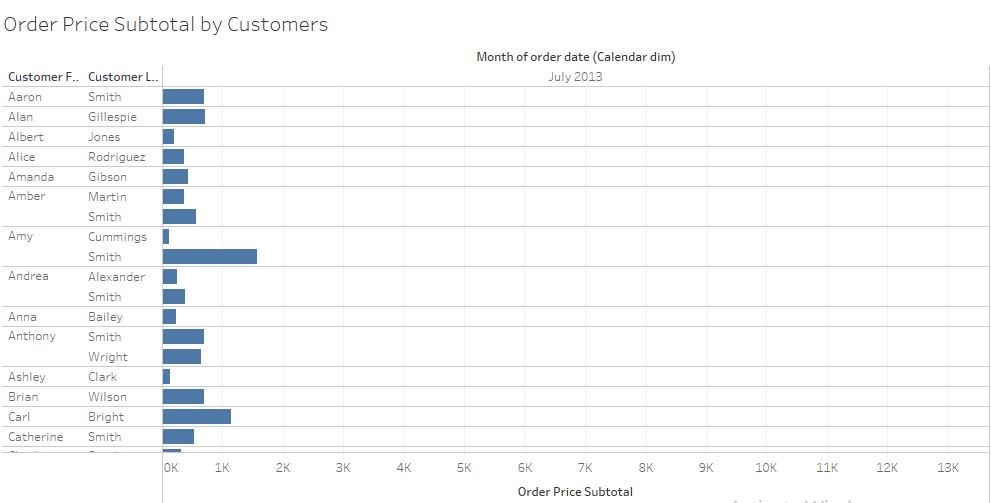
* Circle View representation of Customer expenditure by Department.



* Pie Chart representation of most popular purchase product Categories.



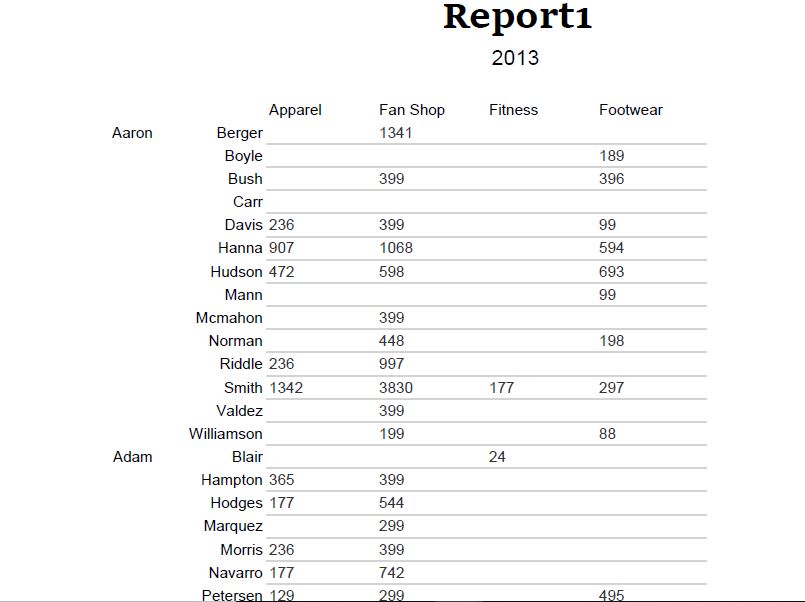
* Horizontal Bar representation of Total expenditure by Customer for a given date



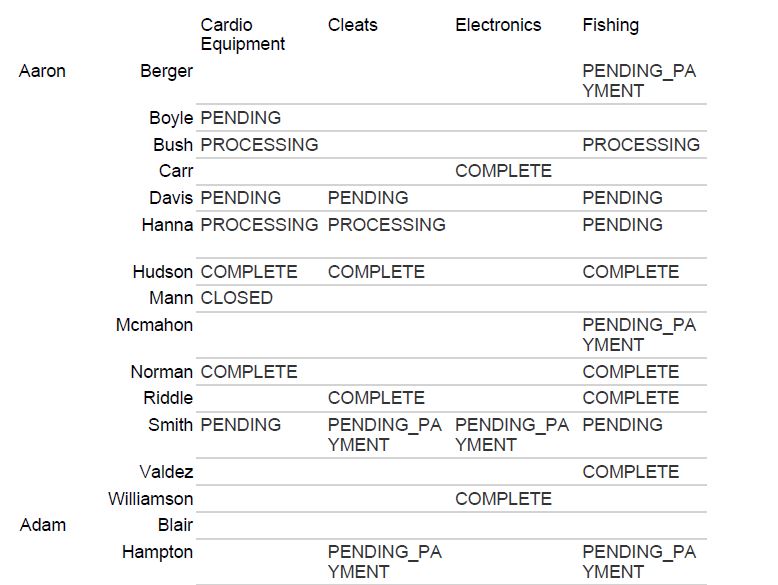
## 4.2. Reports

Mention the business requirement and the report created for it. Include screen shot.

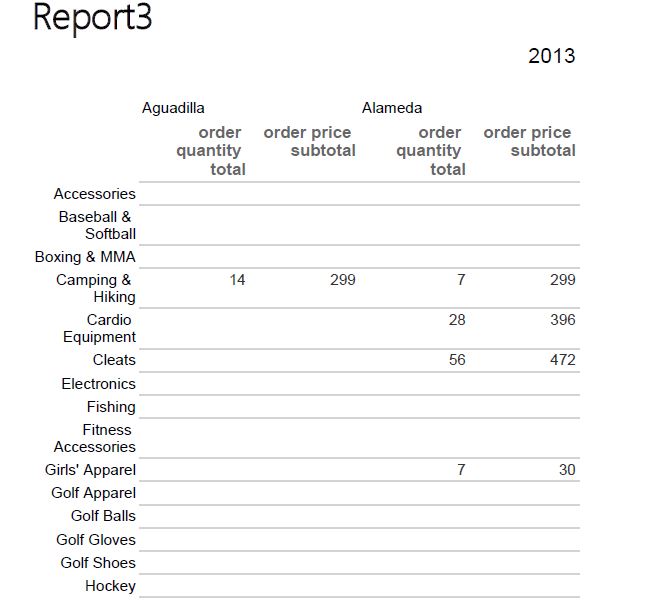
* Total expenditure with customer details and category linked with it



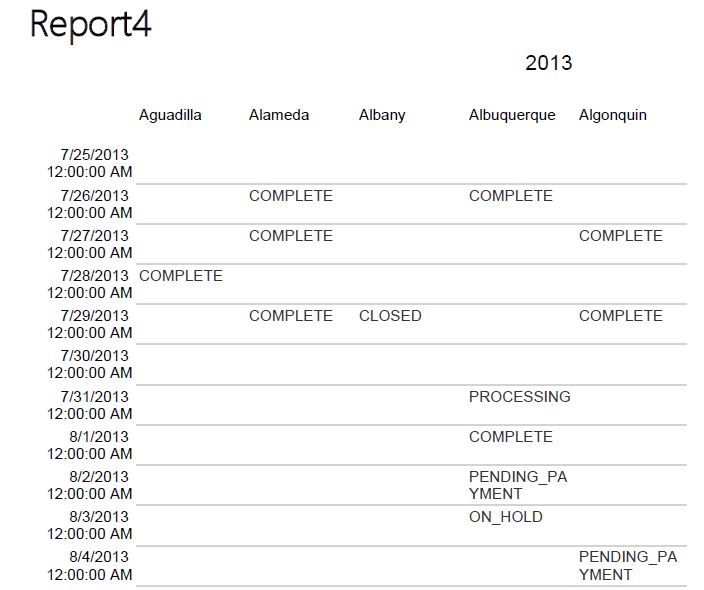
* Department wise Order status for each customer



* Category wise quantity and price subtotal for each customer



* Date wise order statuses for each Customer



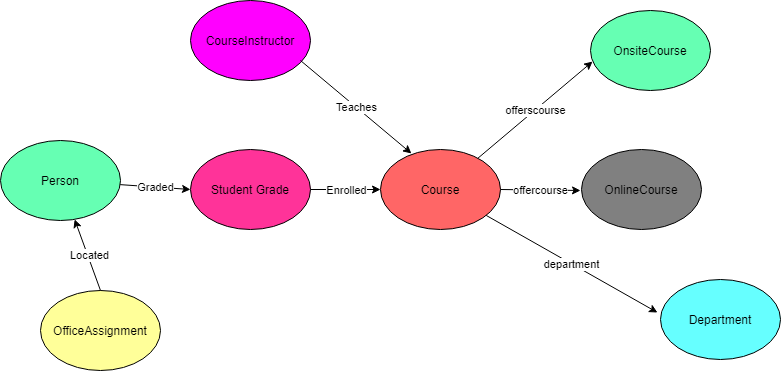
# 5. Graph Databases

Include details here. See Appendix B for code

Ryan School Dataset has been used to store and query data in a Graph Database, Neo4j.

Link to the Dataset: <https://docs.microsoft.com/en-us/ef/ef6/resources/school-database>

Ryan\_School Graph Database Model:



8 nodes have been created for 8 different entities and relations as described in the diagram.

Neo4j is a Graph Database Management system works on Nodes, relations and entities. Ryan School Database has been used to create database and following record have been fetched using Cypher Query Laguage.

Following record have been fetched using CQL:

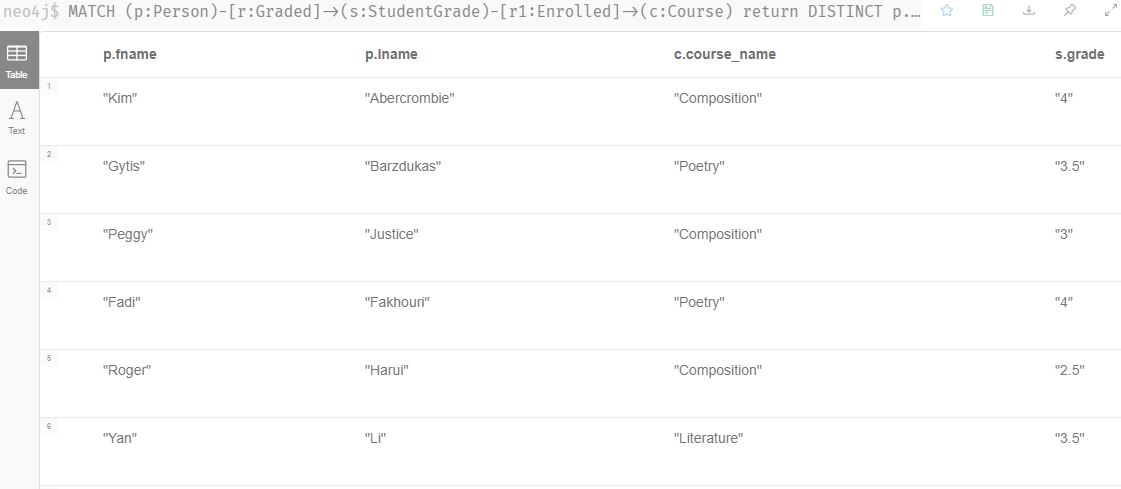
Use set of Course Id values to fetch Instructor/Student names and their respective Departments:-

*MATCH (p:Person)-[r:Enrolled]->(ci:CourseInstructor)-[r1:Teaches]->(c:Course)-[r2:department]-> (d:Department) WHERE c.course\_id IN ['1045','1050','2021','4041'] return DISTINCT p.fname, p.lname,p.discriminator,d.name*



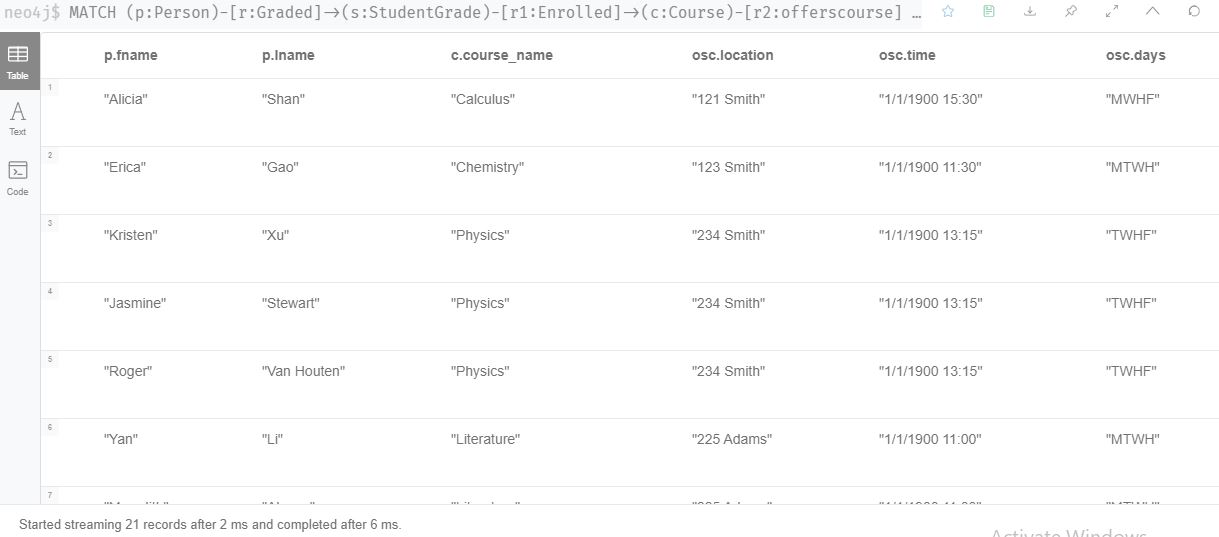
Grades received by Students in various courses:-

*MATCH (p:Person)-[r:Graded]->(s:StudentGrade)-[r1:Enrolled]->(c:Course) return DISTINCT p.fname, p.lname,c.course\_name,s.grade*



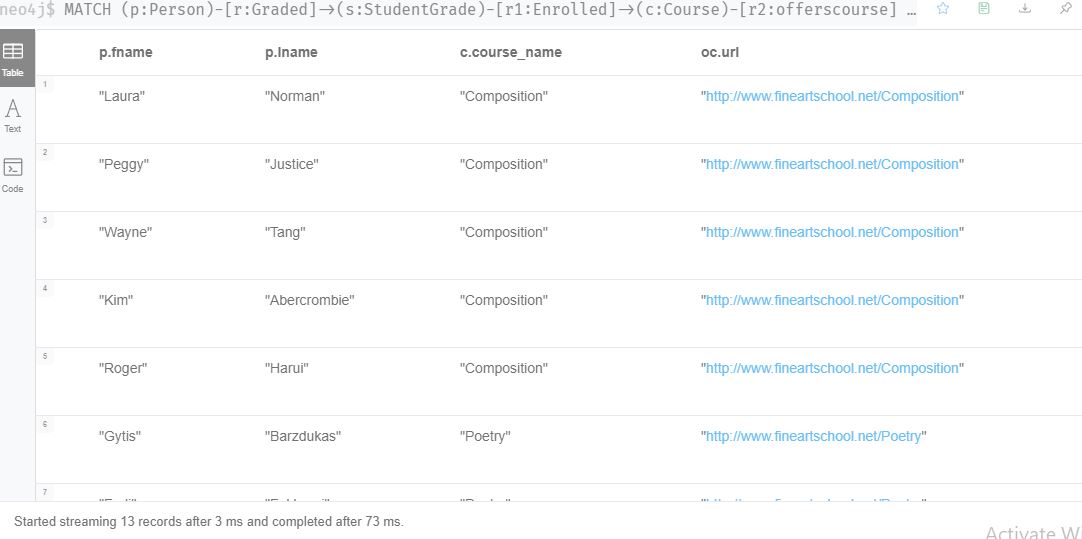
Students School Time Table;-

*MATCH (p:Person)-[r:Graded]->(s:StudentGrade)-[r1:Enrolled]->(c:Course)-[r2:offerscourse]->(osc:OnsiteCourse) return p.fname, p.lname,c.course\_name,osc.location,osc.time,osc.days*



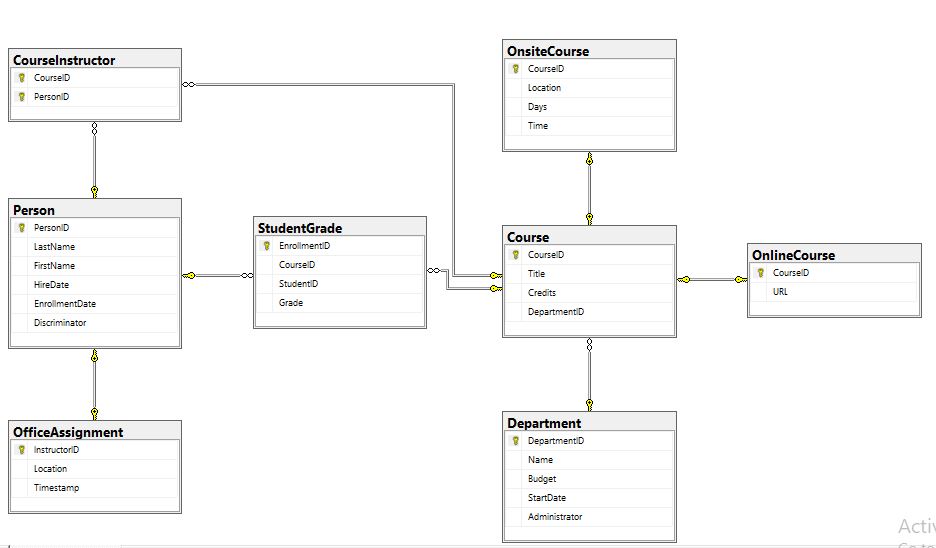
URL for Online Classes for Students enrolled:-

*MATCH (p:Person)-[r:Graded]->(s:StudentGrade)-[r1:Enrolled]->(c:Course)-[r2:offerscourse]->(oc:OnlineCourse) return DISTINCT p.fname,p.lname,c.course\_name,oc.url*



## 5.1. COMPARISON to reLAtional databases

Database diagram for Ryan School dataset in SQL:



Use set of Course Id values to fetch Instructor/Student names and their respective Departments:-

*SELECT p.FirstName, p.LastName, p.Discriminator, d.Name AS DepartmentName*

*FROM Person p*

*INNER JOIN CourseInstructor ci ON p.PersonID = ci.PersonID*

*INNER JOIN Course c ON ci.CourseID = c.CourseID*

*INNER JOIN Department d ON c.DepartmentID = d.DepartmentID*

*WHERE c.CourseID IN (1045, 1050, 2021, 4041)*

**

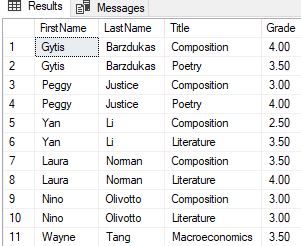
Grades received by Students in various courses:-

*SELECT p.FirstName, p.LastName,c.Title,s.Grade*

*FROM Course c*

*INNER JOIN StudentGrade s ON c.CourseID=s.CourseID*

*INNER JOIN Person p ON s.StudentID = p.PersonID*

**

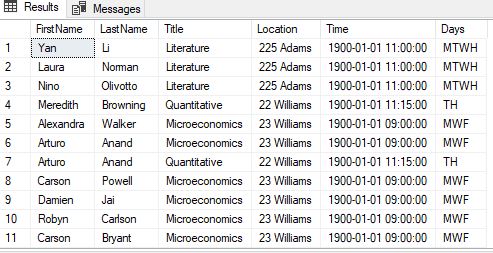
Students School Time Table;-

*SELECT p.FirstName, p.LastName, c.Title, os.Location,os.Time,os.Days*

*FROM Person p INNER JOIN StudentGrade s ON p.PersonID = s.StudentID*

*INNER JOIN Course c ON c.CourseID = s.CourseID*

*INNER JOIN OnsiteCourse os ON os.CourseID = c.CourseID*



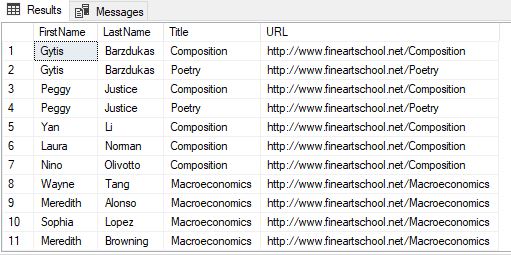
URL for Online Classes for Students enrolled:-

*SELECT p.FirstName, p.LastName, c.Title, ol.URL FROM Person p*

*INNER JOIN StudentGrade s ON p.PersonID = s.StudentID*

*INNER JOIN Course c ON s.CourseID = c.CourseID*

*INNER JOIN OnlineCourse ol ON c.CourseID = ol.CourseID*



# 6. Conclusions

To conclude we can say that both Sql and Nosql have their advantages and disadvantages and we observed in the case of sequential query language databases, it is more complex to work with as compared to Nosql databases. The Visual studio ETL process does make it rather easy to work with Sql but even still NoSql comes out easier. Both have their advantages and disadvantages in their own right.

The very usefulness of both can be realized from the fact that the NEO4J and SPSS generate some deep rooted and meaningful insights that can be blind to the naked human eye.

Right from choosing a good relational dataset until getting insight out of it, the first thing we had to be very clear of in our heads is what do we want to get out of this. What results do we want? Once done with that, we design an appropriate warehouse design (star schema) that will become the conceptual schema of the data warehouse. According to that schema we then design the warehouse, implement appropriate joins to populate the warehouse from source data. Joins works wonders in getting the right data into the table and limiting data redundancy. Simpler and yet similar Cypher queries did more or less the same thing in NoSql. As for transforming the data, we transformed the order date into a lot of different values such as day of week, week number etc using the DatePart function.

Once the process of ETL was complete, the warehouse was now completely at our disposal to use and manipulate. We then picked our ideas of representation and shaped them into SPSS and tableau reports.

The very idea all throughout this process was to generate the insight as to which customer shops the most, which area has the most sales, which department individual customers shop in the most and more.

These end results were well obtained and diverse learning of the actual industrial hand on skill was learnt.

# 7. Bibliography

microsoft 12/10/2019, DatePart Function Available from: <https://docs.microsoft.com/en-us/office/vba/language/reference/user-interface-help/datepart-function> [15 January 2021]

w3schools n.d., Select Distinct Available from: < https://www.w3schools.com/sql/sql\_distinct.asp > [24 December 2020]

w3schools n.d., Sql Orderby Available from: < https://www.w3schools.com/sql/sql\_orderby.asp > [24 December 2020]

w3schools n.d., Sql Insert Available from: < https://www.w3schools.com/sql/sql\_insert.asp > [30 December 2020]

w3schools n.d., Sql Null Values Available from: < https://www.w3schools.com/sql/sql\_null\_values.asp > [30 December 2020]

w3schools n.d., Sql Update Available from: < https://www.w3schools.com/sql/sql\_update.asp > [5 January 2021]

neo4j Developer n.d., Dates Datetimes Durations Available from: < https://neo4j.com/developer/cypher/dates-datetimes-durations/ > [8 January 2021]

neo4j Developer n.d., Querying with Cypher Available from: < https://neo4j.com/developer/cypher/querying/> [11 January 2021]

tableau n.d., Build a basic view to explore your data Available from: <https://help.tableau.com/current/pro/desktop/enus/getstarted\_buildmanual\_ex1basic.html> [15 January 2021]

Tutorial Gateway n.d., Tableau Reports Available from: < https://www.tutorialgateway.org/tableau-reports/> [15 January 2021]

# Appendix A – VISUALIZATIONS Code

Customers Dimension Visualization

SELECT c.customer\_id, c.customer\_fname, c.customer\_lname, c.customer\_email, c.customer\_password,

c.customer\_street, c.customer\_city, c.customer\_state, c.customer\_zipcode

FROM AMAZON\_ORDER\_SOURCE.dbo.Customers c

INNER JOIN AMAZON\_ORDER\_SOURCE.dbo.Orders ON Orders.order\_customer\_id = Customers.customer\_id

GO

Orders Dimension Visualization

SELECT DISTINCT order\_date, DATENAME(dw, order\_date),

DATEPART(day, order\_date), DATEPART(month, order\_date),

DATEPART(year, order\_date), DATEPART(week, order\_date)

FROM AMAZON\_ORDER\_SOURCE.dbo.Orders

INNER JOIN AMAZON\_ORDER\_SOURCE.dbo.Orders\_items ON Orders.order\_id = Orders\_items.order\_item\_order\_id

GO

Product Dimension Visualization

SELECT p.product\_id, p.product\_name, p.product\_image, p.product\_price, c.category\_name, d.department\_name

FROM AMAZON\_ORDER\_SOURCE.dbo.Products p

INNER JOIN AMAZON\_ORDER\_SOURCE.dbo.Category c ON p.product\_category\_id = c.category\_id

INNER JOIN AMAZON\_ORDER\_SOURCE.dbo.Departments d ON c.category\_department\_id = d.department\_id;

GO

Calendar Dim Visualization

SELECT DISTINCT date\_id, order\_date, DATENAME(dw, order\_date) AS Weekday,

DATEPART(day, order\_date) AS day, DATEPART(month, order\_date) AS month, DATEPART(year, order\_date) AS year

FROM AMAZON\_ORDER\_SOURCE.dbo.Orders\_source

INNER JOIN AMAZON\_ORDER\_SOURCE.dbo.Customers ON Orders.order\_id = Orders\_items.order\_item\_order\_id

Fact Table Visualization

SELECT Customers.customer\_id, Products.product\_id, Orders.order\_id,Orders.date\_id,

Orders.order\_date, Orders.order\_status,

MAX(Orders\_items.order\_item\_quantity \* Orders\_items.order\_item\_product\_price) AS order\_price\_subtotal, SUM(Orders\_items.order\_item\_quantity) AS order\_quantity\_total

FROM AMAZON\_ORDER\_SOURCE.dbo.Orders INNER JOIN

AMAZON\_ORDER\_SOURCE.dbo.Customers ON Orders.order\_customer\_id = Customers.customer\_id INNER JOIN

AMAZON\_ORDER\_SOURCE.dbo.Orders\_items ON Orders.order\_id = Orders\_items.order\_item\_order\_id INNER JOIN

AMAZON\_ORDER\_SOURCE.dbo.Products ON Orders\_items.order\_item\_product\_id = Products.product\_id INNER JOIN

AMAZON\_ORDER\_SOURCE.dbo.Category INNER JOIN

AMAZON\_ORDER\_SOURCE.dbo.Departments ON Category.category\_department\_id = Departments.department\_id ON Products.product\_category\_id = Category.category\_id

GROUP BY Customers.customer\_id, Products.product\_id, Orders.order\_id, Orders.date\_id,

Orders.order\_status, Orders.order\_date

# Appendix B – Neo 4J code

Cypher queries to Load csv files data in database in a node and add a Unique constraint:

*LOAD CSV WITH HEADERS FROM "file:///Course.csv" as row CREATE (c:Course) SET c= {course\_id:row.CourseID,course\_name:row.Title,course\_credits:row.credits,course\_deptid:row.DepartmentID} return c*

*CREATE CONSTRAINT ON (c:Course) ASSERT c.course\_id IS UNIQUE*

*LOAD CSV WITH HEADERS FROM "file:///CourseInstructor.csv" as row CREATE (ci:CourseInstructor) SET ci= {course\_id: row.CourseID,person\_id:row.PersonID} return ci*

*CREATE CONSTRAINT ON (ci:CourseInstructor) ASSERT ci.course\_id IS UNIQUE*

*LOAD CSV WITH HEADERS FROM "file:///Department.csv" as row CREATE (d:Department) SET d= {dept\_id: row.DepartmentID,name:row.Name, budget: row.Budget, start\_date: row.StartDate, admin :row.Administrator} return d*

*CREATE CONSTRAINT ON (d:Department) ASSERT d.dept\_id IS UNIQUE*

*LOAD CSV WITH HEADERS FROM "file:///OnlineCourse.csv" as row CREATE (oc:OnlineCourse) SET oc= {course\_id: row.CourseID,url:row.URL} return oc*

*CREATE CONSTRAINT ON (oc:OnlineCourse) ASSERT oc.course\_id IS UNIQUE*

*LOAD CSV WITH HEADERS FROM "file:///OnsiteCourse.csv" as row CREATE (osc:OnsiteCourse) SET osc= {course\_id: row.CourseID,location:row.Location, days:row.Days, time:row.Time} return osc*

*CREATE CONSTRAINT ON (osc:OnsiteCourse) ASSERT osc.course\_id IS UNIQUE*

*LOAD CSV WITH HEADERS FROM "file:///Person.csv" as row CREATE (p:Person) SET p= {person\_id: row.PersonID, lname:row.LastName, fname:row.FirstName, hire\_date:row.HireDate, enrollment\_date:row.EnrollmentDate,discriminator:row.Discriminator} return p*

*CREATE CONSTRAINT ON (p:Person) ASSERT p.person\_id IS UNIQUE*

*LOAD CSV WITH HEADERS FROM "file:///OfficeAssignment.csv" as row CREATE (o:OfficeAssignment) SET o= {instructor\_id: row.InstructorID, location:row.Location, timestamp:row.Timestamp} return o*

*CREATE CONSTRAINT ON (o:OfficeAssignment) ASSERT o.instructor\_id IS UNIQUE*

*LOAD CSV WITH HEADERS FROM "file:///StudentGrade.csv" as row CREATE (s:StudentGrade) SET s= {enrollment\_id: row.EnrollmentID, course\_id:row.CourseID, student\_id:row.StudentID, grade:row.Grade} return s*

*CREATE CONSTRAINT ON (s:StudentGrade) ASSERT s.student\_id IS UNIQUE*

Adding relations among different nodes:

*MATCH (c:Course),(d:Department) WHERE c.course\_deptid=d.dept\_id CREATE (c)-[r:department]- > (d) return c, d,r*

*MATCH(c:Course),(oc:OnlineCourse) WHERE c.course\_id=oc.course\_id CREATE (c)-[r:offerscourse]- > (oc) return c,oc,r*

*MATCH(c:Course),(osc:OnsiteCourse) WHERE c.course\_id=osc.course\_id CREATE (c)-[r:offerscourse]- > (osc) return c,osc,r*

*MATCH(p:Person),(o:OfficeAssignment) WHERE p.person\_id=o.instructor\_id CREATE (p)-[r:Located]- > (o) return p,o,r*

*MATCH(ci:CourseInstructor),(p:Person) WHERE ci.person\_id=p.person\_id CREATE (ci)-[r:Enrolled]- > (p) return ci,p,r*

*MATCH(p:Person),(s:StudentGrade) WHERE p.person\_id=s.enrollment\_id CREATE (p)-[r:Graded]- > (s) return p,s,r*

*MATCH(s:StudentGrade),(c:Course) WHERE s.course\_id=c.course\_id CREATE (s)-[r:Enrolled]- > (c) return s,c,r*

*MATCH(ci:CourseInstructor),(c:Course) WHERE ci.course\_id=c.course\_id CREATE (ci)-[r:Teaches]- > (c) return ci,c,r*