**Penn State University**

Great Valley Campus 

***Engineering Division***

**Data Specification for**

**NY Annualized Property Sales**

***Version 5.0***

27th Feb 2022

**Table of Contents**

[Introduction 3](#_Toc96889650)

[Purpose 3](#_Toc96889651)

[Project Summary 3](#_Toc96889652)

[Requirements Definition 4](#_Toc96889653)

[Document Change Log 5](#_Toc96889654)

[2. Architecture Design 6](#_Toc96889655)

[2.1 Relational Data Warehouse 6](#_Toc96889656)

[2.1.1 Choose Fact Table 6](#_Toc96889657)

[2.1.2 Determine the grain of the fact table 6](#_Toc96889658)

[2.1.3 Determine the attributes in dimension tables 6](#_Toc96889659)

[2.1.4 Indexing 7](#_Toc96889660)

[2.1.5 Data Dictionary 7](#_Toc96889661)

[2.1.6 Star Schema 8](#_Toc96889662)

[2.1.7 Tables schemas 8](#_Toc96889663)

[2.2 Hadoop Implementation 13](#_Toc96889664)

[2.3 Reflective analysis of using a data warehouse vs Hadoop. 15](#_Toc96889665)

[3. Data Preparation 16](#_Toc96889666)

[3.1 Relational Data Warehouse Implementation 16](#_Toc96889667)

[3.2 Hadoop Implementation 22](#_Toc96889668)

[3.3 Reflective analysis of data preparation in relational data warehouse vs Hadoop. 27](#_Toc96889669)

[4. Reporting System 29](#_Toc96889670)

[4.1 Relational Data Warehouse Implementation 29](#_Toc96889671)

[4.2 Hadoop Implementation 33](#_Toc96889672)

[4.3 Reflective analysis of result in relational data warehouse vs Hadoop. 39](#_Toc96889673)

[5. Conclusions 40](#_Toc96889674)

**NY Annualized Property Sales Systems  
Ambika Chundru**

# Introduction

New York is a state in United States. It is one of the largest states, with approximately 20.2 million residents. It is considered as 4th highest populated state in USA, with 44% living in New York City. It is sometimes called New York State to distinguish it from its largest city, New York City. New York City is divided into 5 boroughs namely Manhattan, Bronx, Brooklyn, Queens, and Staten Island. Due to high population in New York City, it is booming in the real estate. In this project I will analyze records of every building or building unit (apartment, commercial unit.) sold in different boroughs in the New York City property market to identify trends and apply the observation while doing new purchase.

# Purpose

The aim of this project is to discover trends about New York City’s property sales by analyzing the given transaction records using data warehouse implementation and Hadoop implementation and compare both the systems.

# Project Summary

This project will used to attain solutions in the field of real estate industry in 5 boroughs in New York City. In the further sections we will be discussing about project objectives, scope, and the outstanding issues.

1. **Objectives**

The project objectives are as follows:

* To understand the trends in annual property sales in various boroughs of New York City using big data solutions through implementing, populating data, and building reports.
* To compare between data warehouse implementation and Hadoop implementation.
* To analyze the importance of Data warehouse and Hadoop in real estate industry in New York City.
* To design and develop a data warehouse and Hadoop system in a real estate industry in New York City.

1. **Scope**

The Scope of the project is to compare two implementations of the annual property sales system. Additionally, this project will focus on the design, development of a data warehouse and Hadoop system of annual property sales of New York City. The data will be presented in the form of information using analysis and reporting services tools to help in decision making. Both the implementations are compared on their capability of delivering information to the management.

1. **References**

<https://www1.nyc.gov/site/finance/taxes/property-annualized-sales-update.page>

<https://en.wikipedia.org/wiki/New_York_(state)>

<https://www1.nyc.gov/assets/finance/downloads/pdf/07pdf/glossary_rsf071607.pdf>

1. **Outstanding Issues**

Some of the required data not captured, by the source systems which is very important for the data warehousing purpose. For example some of the property details like land square feet and sale price the date of registration for the property are missing which are very important for analysis purposes. The data extracted does not have similar timeframe of year of property sales. For example some boroughs do not have few years of data.

# Requirements Definition

1. **Goals**

The goals of this project include:

* Understand the given data to build data warehouse and Hadoop system
* Compare between data warehouse implementation and Hadoop implementation
* Load data, and execute business reports
* Analyze both the systems architecture with respect to the given data.

1. **Usability Requirements**

Some of the usability requirements include:

* User should have a working knowledge of Knime and PostgreSQL
* User should also have a working knowledge of Jave, Hadoop, Hive, Docker.

1. **System Security Requirements**

* Using the Hostname, portname, username, and password, only authenticated users can access the Postgre database warehouse and Hadoop Hive database.
* The data can't be changed without the admin's authorization.

1. **Business Questions**

* What is sales trend of top three building categories over the years.
* Which Borough has the highest sales.
* How many total sales of residential units and commercial units over the years?
* What is the sales trend over months in different boroughs?

1. **Data Requirements**

The course instructor has provided data as part of the project. It is collection of 9 excel files of property sales in 5 boroughs ranging from the year 2005 to 2014. The provided data was pre-processed for the scope of the course, especially it was reduced in size. The file includes many columns namely: location, address, type, sale price, borough and sale date of building units sold. Here in borough the values depict the following locations, Manhattan (1), Bronx (2), Brooklyn (3), Queens (4), and Staten Island (5). The mixture of borough, block, and lot forms a unique key for property in New York City this is known as BBL.

1. **Design Constraints**

We will be using star schema design for our database warehousing. The main issue with star schema is data redundancy. We can observe a risk of data integrity due to redundancy of data. As a normalized data model, the star schema is not flexible enough for analytical requirements.

# Document Change Log

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Change Date** | **Version** | **CR #** | **Change Description** | **Author and Organization** |
| 01/23/22 | 1.0 |  | Initial creation. | Ambika Chundru |
| 02/06/22 | 2.0 |  | Architecture design data warehouse | Ambika Chundru |
| 02/13/22 | 3.0 |  | ETL and Reporting | Ambika Chundru |
| 02/17/2022 | 4.0 |  | Hadoop Architecture design | Ambika Chundru |
| 02/21/2022 | 5.0 |  | Hadoop ETL and reporting | Ambika Chundru |

# 2. Architecture Design

## 2.1 Relational Data Warehouse

### 2.1.1 Choose Fact Table

A fact table in data warehousing contains the measurements, metrics, or facts of a business operation. It is at the middle of a star diagram. By observing the There are two sorts of columns in a fact table: those that hold facts and those that serve as a foreign key to dimension tables. A fact table's main key is frequently a composite key made up of all of the table's foreign keys. Here we used sale price as a fact and the remaining dimensions for foreign keys.

### 2.1.2 Determine the grain of the fact table

The most important stage in a dimensional design is to declare the grain. The grain specifies what a single row in a fact table represents. The grain statement becomes a legally enforceable design contract. Before picking dimensions or facts, the grain must be specified since each candidate dimension or fact must be compatible with the grain. This consistency ensures that all dimensional designs are consistent, which is crucial for BI application performance and usability. For the property sales the grain can be defined as sale price by month over boroughs, building categories, building classes etc. The data can be loaded every day.

### 2.1.3 Determine the attributes in dimension tables

A dimension is a collection of reference information about a measurable event in data warehousing. These occurrences are referred to as facts, and they are recorded in a fact table. Dimensions organize and characterize data warehouse facts and metrics so that they may be used to answer business queries. They are the foundation of three-dimensional modeling. For the property sales data set we have chosen the following attributes:

* Borough
* Neighborhood
* Block
* Lot
* Address
* Zip code
* Residential units
* Land square feet
* Gross square feet
* Year build
* Tax class at time of sale
* Building class at time of sale

We removed few attributes which had minimal relevance to our business questions or had a lot of null values(Apartment number, easement) or could be calculated from above attributes(Total units).

### 2.1.4 Indexing

For the implementation of data warehouse using postgre, Here the rows can be indexed using serial4 this allows incrementation.

### 2.1.5 Data Dictionary

The data dictionary lists all of the variables that were evaluated when creating the data warehouse. Few variables are removed as they could be calculated with original variables or were of minimal relevance for the data warehouse.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Data Dictionary** | | | | |
| Variable | Variable name | Variable type | Values | notes |
| BOROUGH | borough | Int4 | 1 to 5 | Id of the borough |
| NEIGHBORHOOD | neighborhood | Varchar | Bath Beach, Bay Ridge | Neighborhood of the property |
| BLOCK | block | Int4 | 6025, 6026,6027…. | A sub-part of the borough where properties are situated |
| LOT | lot | Int4 | 75, 54, 9, 10…. | A sub-part of a block to represent property’s unique location |
| ADDRESS | address | Varchar | 336 84 Street, 337 85 Street … | Street address of the property |
| ZIPCODE | zipcode | Int4 | 11228, 11214….. | Zip code of the property |
| RESIDENTIAL UNITS | residentialunits | Int4 | 0,1,2.. | No of residential units per property |
| COMMERCIAL UNITS | commercialunits | Int4 | 0,1,2.. | No of commercial units per property |
| LAND SQUARE FEET | landarea | Int4 | 1,993, 4,088,… | Land area of the property in square feet |
| GROSS SQUARE FEET | grossarea | Int4 | 2,320, 2,167,…. | Total area of the property in square feet |
| BUILDING CLASS CATEGORY | buildingclasscat | Varchar | 02 TWO FAMILY HOMES,… | Category of the building class |
| TAX CLASS AT TIME OF SALE | taxclass | Int4 | 1,2,… | Tax class at the time of sale |
| BUILDING CLASS AT TIME OF SALE | buildingclass | Varchar | R1, R2, C1, C0… | Building class at the time of sale |
| SALE PRICE | sales | Serial4 | 436095, 0,… | Price of the property |
| SALE DATE | rawdate | date | 3/17/2005, 4/1/2008 | Date at which property was sold |

### 2.1.6 Star Schema

This is how the schema looks like:

Diagram

Description automatically generated

### 2.1.7 Tables schemas

Description of the physical schema of the data warehouse.

|  |  |  |  |
| --- | --- | --- | --- |
| ***location*** |  |  |  |
| **Description** | Describes the location of the sale property in New York City | | |
| **Attribute** | **Description** | **Type** | Examples of values |
| **id** | Location ID | Serial4 | 1,2,3,…. |
| **borough** | Id of the borough | Int4 | 1,2,3,4,5 |
| **neighborhood** | Neighborhood of the property | Varchar | Bath Beach, Bay Ridge |
| **block** | A sub-part of the borough where properties are situated | Int4 | 6025, 6026,6027,6031... |
| **lot** | A sub-part of a block to represent property’s unique location | Int4 | 75, 54, 9,10…. |
| **address** | Street address of the property | Varchar | 336 84 Street, 337 85 Street |
| **zipcode** | Zip code of the property | Int4 | 11228, 11214….. |
| **Primary Key** | id | | |
| **Foreign Keys** |  | | |
| **SQL Query** | CREATE TABLE location (  borough int4 NULL,  neighborhood varchar NULL,  block int4 NULL,  lot int4 NULL,  address varchar NULL,  zipcode int4 NULL,  id serial4 NOT NULL,  CONSTRAINT location\_pk PRIMARY KEY (id)  ); | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ***class*** |  |  |  |
| **Description** | Describes the tax class and building class at the time of sale. And describes the building class category | | |
| **Attribute** | **Description** | **Type** | Examples of values |
| **id** | Class ID | Serial4 | 1,2,3,…. |
| **buildingclasscat** | Category of the building class | Varchar | 02 TWO FAMILY HOMES, 01 ONE FAMILY HOMES .. |
| **taxclass** | Tax class at the time of sale | Int4 | 1,2,… |
| **buildingclass** | Building class at the time of sale | Varchar | R1, R2, C1, C0… |
| **Primary Key** | id | | |
| **Foreign Keys** |  | | |
| **SQL Query** | CREATE TABLE class (  id serial4 NOT NULL,  buildingclasscat varchar NULL,  taxclass int4 NULL,  buildingclass varchar NULL,  CONSTRAINT class\_pk PRIMARY KEY (id)  ); | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ***date*** |  |  |  |
| **Description** | Describes the sale dates of the property in New York City | | |
| **Attribute** | **Description** | **Type** | Examples of values |
| **id** | Date ID | Serial4 | 1,2,3,…. |
| **year** | Year of sale | Int4 | 2005, 2008, 2009, 2010, 2011… |
| **month** | Month of sale | Int4 | 1 to 12 |
| **dayofmonth** | Day of sale in a month | Int4 | 1 to 31 |
| **quater** | Quarter of sale | Int4 | 1 to 4 |
| **rawdate** | Date of sale | Date | 7/28/2005, 10/12/2005 |
| **Primary Key** | id | | |
| **Foreign Keys** |  | | |
| **SQL Query** | CREATE TABLE date (  id serial4 NOT NULL,  "year" int4 NULL,  "month" int4 NULL,  dayofmonth int4 NULL,  quarter int4 NULL,  rawdate date NULL,  CONSTRAINT date\_pk PRIMARY KEY (id)  ); | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ***units*** |  |  |  |
| **Description** | Describes the gross and land square feet of the property | | |
| **Attribute** | **Description** | **Type** | Examples of values |
| **id** | Unit ID | Serial4 | 1,2,3,…. |
| **residentialunits** | No of residential units per property | Int4 | 0, 1, 2… |
| **commercialunits** | No of commercial units per property | Int4 | 0, 1, 2,… |
| **Primary Key** | id | | |
| **Foreign Keys** |  | | |
| **SQL Query** | CREATE TABLE units (  id int4 NOT NULL,  residentialunits int4 NULL,  commercialunits int4 NULL,  CONSTRAINT units\_pk PRIMARY KEY (id)  ); | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ***area*** |  |  |  |
| **Description** | Describes the gross and land square feet of the property | | |
| **Attribute** | **Description** | **Type** | Examples of values |
| **id** | Area ID | Serial4 | 1,2,3,…. |
| **landarea** | Land area of the property in square feet | Int4 | 1,993, 4,088,… |
| **grossarea** | Total area of the property in square feet | Int4 | 2,320, 2,167,…. |
| **Primary Key** | id | | |
| **Foreign Keys** |  | | |
| **SQL Query** | CREATE TABLE area (  id serial4 NOT NULL,  landarea int4 NULL,  grossarea int4 NULL,  CONSTRAINT area\_pk PRIMARY KEY (id)  ); | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ***factsales*** |  |  |  |
| **Description** | Describes the location of the sale property in New York City | | |
| **Attribute** | **Description** | **Type** | Examples of values |
| **location\_id** | Location ID | Serial4 | 1,2,3,…. |
| **area\_id** | Area ID | Serial4 | 1,2,3,…. |
| **unit\_id** | Unit ID | Serial4 | 1,2,3,…. |
| **date\_id** | Date ID | Serial4 | 1,2,3,…. |
| **class\_id** | Class ID | Serial4 | 1,2,3,…. |
| **sales** | Price of the property | Int4 | 436095, 0,… |
| **Primary Key** | **location\_id, area\_id, unit\_id, date\_id, class\_id** | | |
| **Foreign Keys** | **location\_id, area\_id, unit\_id, date\_id, class\_id** | | |
| **SQL Query** | CREATE TABLE factsales (  location\_id serial4 NOT NULL,  area\_id serial4 NOT NULL,  unit\_id serial4 NOT NULL,  date\_id serial4 NOT NULL,  class\_id serial4 NOT NULL,  sales int4 NULL,  CONSTRAINT factsales\_pk PRIMARY KEY (location\_id, area\_id, unit\_id, date\_id, class\_id)  CONSTRAINT factsales\_fk FOREIGN KEY (location\_id) REFERENCES location(id);  CONSTRAINT factsales\_fk\_1 FOREIGN KEY (area\_id) REFERENCES area(id);  CONSTRAINT factsales\_fk\_2 FOREIGN KEY (unit\_id) REFERENCES units(id);  CONSTRAINT factsales\_fk\_3 FOREIGN KEY (date\_id) REFERENCES date(id);  CONSTRAINT factsales\_fk\_4 FOREIGN KEY (class\_id) REFERENCES class(id); | | |

## 2.2 Hadoop Implementation

For building architecture in Hadoop I have used the same columns for the same reasons as above. For a revision I have mentioned the attribute description below in the given data dictionary. We removed few attributes which had minimal relevance to our business questions or had a lot of null values(Apartment number, easement, tax class at present, building class at present) or could be calculated from the below attributes(Total units).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Data Dictionary** | | | | |
| Variable | Variable name | Variable type | Values | notes |
| BOROUGH | borough | Int | 1 to 5 | Id of the borough |
| NEIGHBORHOOD | neighborhood | Varchar | Bath Beach, Bay Ridge | Neighborhood of the property |
| BLOCK | block | Int4 | 6025, 6026,6027…. | A sub-part of the borough where properties are situated |
| LOT | lot | Int4 | 75, 54, 9, 10…. | A sub-part of a block to represent property’s unique location |
| ADDRESS | address | Varchar | 336 84 Street, 337 85 Street … | Street address of the property |
| ZIPCODE | zipcode | Int4 | 11228, 11214….. | Zip code of the property |
| RESIDENTIAL UNITS | residentialunits | Int4 | 0,1,2.. | No of residential units per property |
| COMMERCIAL UNITS | commercialunits | Int4 | 0,1,2.. | No of commercial units per property |
| LAND SQUARE FEET | landarea | Int4 | 1,993, 4,088,… | Land area of the property in square feet |
| GROSS SQUARE FEET | grossarea | Int4 | 2,320, 2,167,…. | Total area of the property in square feet |
| BUILDING CLASS CATEGORY | buildingclasscat | Varchar | 02 TWO FAMILY HOMES,… | Category of the building class |
| TAX CLASS AT TIME OF SALE | taxclass | Int4 | 1,2,… | Tax class at the time of sale |
| BUILDING CLASS AT TIME OF SALE | buildingclass | Varchar | R1, R2, C1, C0… | Building class at the time of sale |
| SALE PRICE | sales | Serial4 | 436095, 0,… | Price of the property |

For the purpose of structuring the data I will be using Hive, this will allow us to process data in Hadoop. Hive is quite similar to SQL and acts as a layer on top of Hadoop. We use Hive to cover our SQL queries into MapReduce jobs and then clusters them. Every inquiry to the Hive will lead to the refreshing of Metastore. To start with we will create a table consisting of all attributes and make sure that they are in de normalized form.

To create the hive table, I have followed the below steps one after another:

1. Initiate Hive by typing bl
2. Create a Database using the SQL Query: CREATE DATABASE IF NOT EXITS ambika
3. Type this query to view the database: SHOW DATABASES

Text

Description automatically generated

1. Type the query to create the table with required attributes and their dtata types respectively: CREATE TABLE nyc\_sales(borough INT,neighborhood VARCHAR(100),building\_class\_category VARCHAR(100),block INT,lot INT,address VARCHAR(100),zipcode INT,residential\_units INT,commercial\_units INT,land\_squarefeet INT,gross\_sqaurefeet INT,tax\_class INT,building\_class VARCHAR(20),sale\_price INT,sale\_date TIMESTAMP,count VARCHAR(50)) ROW FORMAT DELIMITED FIELDS TERMINATED BY ','STORED AS TEXTFILE;
2. Type the query to see the table: DESCRIBE nyc\_sales

A picture containing table

Description automatically generated

## 2.3 Reflective analysis of using a data warehouse vs Hadoop.

Data Warehouse: While developing the data warehouse architecture, I came to notice that it in this architecture makes the best use of relational and structured data. It is not good with unstructured data. It is necessary for the data to be clean and consistent as it cannot handle raw unstructured data.

Hadoop: Coming to Hadoop implementation, building its architectural design was very easy compared to that for data warehousing as the only thing we need to do is prune the not required attributes. Here we do not need to make separate tables and we also don’t need to initialize primary and foreign keys. All the data is stored in a one table.

# 3. Data Preparation

## 3.1 Relational Data Warehouse Implementation

**ETL considerations**

ETL process has 3 important steps: extract, transform, and load. Lets talk about their consideration in detail:

**Extract:** In laymen terms this is the process of reading data from different sources. We will be using logical extraction. Here we extract data completely from the source system. The main advantage of this extraction is we don’t need to keep the rack of changes. We will be deploying online physical extraction.

**Transform:** This step involves transforming data per required format before loading. Here we have to select only the required data and discard the remaining data. In the required situations we will join the data (for fact table). Next step is conversion, to ensure data compatibility with the target system, shape the format and structure of the data.

**Load:** This is the processof writing data into the target source. While loading the data it is necessary to maintain data freshness. Our data here is historical data so this data will have high update efficiency. Faced some issues related to system performance. The ratio of new to historical data should be low.

**ETL Process Flow with description**

To start with we know that there are 5 steps in the ETL process namely: extract, clean, transform, load, and analyze. Let’s discuss about them in detail:

**Extract:** As the name suggest we extract raw data from a data pool and migrate it to do further steps on the raw data. I have used KNIME for this process, since the data is stored in excel files. I have called the Excel reader node and loaded all the 9 excel files of our data set. Doing so will concatenate all the data with respect to the columns.

**Clean:** Next the extracted data is cleaned to ensure the data quality before we apply transformation. Here, I have used column filter to prune the columns which as pe the requirement. Since the date is stored as a string, I have used string to data&time node to convert it into data format. To remove the columns having null values in the sale price row we will use row filter.

**Transform:** The next step involves structuring and conversion of the data to the correct format to match the target source. We have a total of 7 dimension tables and 1 fact table, and we will transform the data with respect to the tables:

* Location: The first step is to filter out the location columns, namely address, block, lot, borough, neighborhood, zip code. This was achieved by using column filter node. Since there are multiple entries of the same apartment, I have used group by node to group address to avoid duplicate entries. The column names are renamed to match the architecture using column rename node. Next, to avoid loading the same data multiple times, I have deployed reference row filter node. This filter compares the data in the data warehouse and the data yet to be loaded to check if there is new data. The inputs of this filter will be DB query reader and column rename node.
* Date: To transform the data that is to be load into date table, first we will filter the sale date column using column filter. Next, I have used group by node to group address to avoid duplicate entries. Our next step is to extract data like year, month, day of month, quarter from the raw date using extracted date and time fields. This is done so that while analyzing the data we can do it over different time periods. The column names are renamed to match the architecture using column rename node. Next, to avoid loading the same data multiple times, I have deployed reference row filter node. The inputs of this filter will be DB query reader and column rename node.
* Tax Class: The same nodes as above can be used for transforming the data for the area dimensions. The columns in the area dimensions are tax class at the time of sale and tax class at present. The nodes used are column filter, group by, column rename, reference row filter, DB query reader.
* Building Class: The same nodes as above can be used for transforming the data for the area dimensions. The columns in the area dimensions are building class at the time of sale and building class at present. The nodes used are column filter, group by, column rename, reference row filter, DB query reader.
* Building Category: The same nodes as above can be used for transforming the data for the area dimensions. The column in the area dimensions is building class category. The nodes used are column filter, group by, column rename, reference row filter, DB query reader.
* Units: The same nodes as above can be used for transforming the data for the area dimensions. The columns in the area dimensions are commercial units and residential units. The nodes used are column filter, group by, column rename, reference row filter, DB query reader.
* Area: The same nodes as above can be used for transforming the data for the area dimensions. The columns in the area dimensions are Land Square Feet and gross Square Feet. The nodes used are column filter, group by, column rename, reference row filter, DB query reader.
* Sales (fact table): Once the data is loaded into the dimensions table, next step is to build sales fact table. This done by taking the result form the last dimension table. Next, we join the columns using joiner node one dimension after another. After this we group the contents of the sales fact table using the dimension table id’s.

Miscellaneous steps:

* Use Metanode to make the node architecture clean and organized. Group nodes were used for each table separately
* Use Flow variables to assign the flow in which the process should take place.

**Load:** Once the data is transformed, we first we load the data into dimension tables using DB writer node. Next, we load the data into the sales fact table using the DB writer node.

**Analyze:** Next the loaded data can be analyzed using different statistical nodes as per our requirement. The data is processed in the warehouse, which allows the business to gain insight. This concludes the ETL process. We will be discussing about reporting in the next milestone. The basic statistics for each excel file is:

1. 2005 Sales Brooklyn

Calendar

Description automatically generated

1. 2008 Sales Staten island

Calendar

Description automatically generated

1. 2009 Bronx

Chart

Description automatically generated with low confidence

1. 2010 Queens

Calendar

Description automatically generated

1. 2010 Statenisland

Calendar

Description automatically generated

1. 2011 Manhattan

Calendar

Description automatically generated

1. 2011 Queens

Calendar

Description automatically generated

1. 2012 Statenisland

Calendar

Description automatically generated

1. 2014 Manhattan

Calendar

Description automatically generated

Total Row counts for each file:

* 2005 Sales Brooklyn = 33,492
* 2008 Sales Staten island = 2,930
* 2009 Bronx = 2,529
* 2010 Queens = 9,039
* 2010 Staten island = 135
* 2011 Manhattan = 1,7202
* 2011 Queens = 377
* 2012. Staten island = 132
* 2014 Manhattan = 2,234.

After removing duplicates and null values for a few characteristics, the total number of rows imported into the database is 80,116.

A screenshot of a computer

Description automatically generated with low confidence

Data in the address table:

Table

Description automatically generated

Data in the area table:

Table

Description automatically generated

Data in the date table:

Table

Description automatically generated

Data in the unit table:

Table

Description automatically generated

Data in the category table:

Table

Description automatically generated

Data in the sales fact table:

Table

Description automatically generated

## 3.2 Hadoop Implementation

For the implementation of ETL in Hadoop we follow the below steps:

1. To start with we create a Hadoop docker image from the provided Hadoop docker file. Once the image is generated, the nest step is to create the Hadoop container by building the image. Next execute the container by using the following command :

docker exec -it 576938a8v14g bash

where 576938a8v14g the container id

1. We then create directories in hadoop. The path of the input and output directories are given below respectively

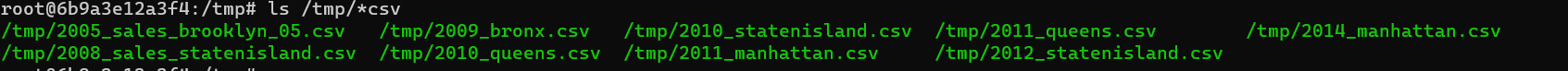
root/exerc/input

root/exerc/output

1. Next we load the datafiles into the tmp folder. We have a total of 9 datafile in csv format.

This can be done by the repeating the following steps for all the 9 files:

* + - docker cp .data\nyc\2005\_sales\_brooklyn\_05.csv 576938a8v14g:/tmp
    - docker exec -it 576938a8v14g bash
    - hadoop fs -mkdir /root
    - hadoop fs -mkdir /root/exerc/input
    - hadoop fs -put /tmp/2005\_sales\_brooklyn\_05.csv /root/exerc/input
    - hadoop fs -ls /root/exerc/input



1. Our next step is to map reduce , to do this we create 3 files: MapIt.java, , ReduceIt.java, and MapReduceIt.java:

Nano MapIt.java,

Nano ReduceIt.java

Nano MapReduceIt.java

1. Our next step is to edit these files according our requirements listed below:
   1. Read the data line after line
   2. Next we use toString() to convert the text variables to string
   3. If there are no values in the line then we stop, so to do this we create a function which checks if the length of the line is 0 or not, if it is it returns
   4. Next we use ‘,’ to separate all tokens/attributes as our data is from a csv file
   5. Next we create a function to choose line only with 21 columns, this can be achieved by tokens != 0
   6. After this the next step is to a variable to each token and apply pre processing on the data. Some of the actions that were performed were trimming, replacing “,”,”$”, converting sales and othe rinterger attributes to Double, converting rawdate to another format.
   7. Next we assign each combination of a line a key and value as 1.
   8. The code for mapit file is given below

Text

Description automatically generated

Text

Description automatically generated

1. Next we create a jar file after the creation and compilation of source files(MapIt.java, , ReduceIt.java, and MapReduceIt.java).
2. This can be achieved by the following commands:
   1. hadoop com.sun.tools.javac.Main \*.java
   2. jar cf mri.jar \*.class
   3. cd /usr/local/hadoop
3. Next I have assigned the input and output directory path as mentioned below and then save the results as shown below.
   1. bin/hadoop jar /root/inclass/mri.jar MapReduceIt /root/exerc/input /root/exerc/output
   2. nano runit.sh
   3. chmod a+x runit.sh
   4. ./runit.sh
   5. nano /tmp/result.txt
4. Next we create Hive database as per the steps given in section 2.2
5. They are :
   1. Initiate Hive by typing bl
   2. Create a Database using the SQL Query: CREATE DATABASE IF NOT EXITS Ambika
   3. Type this query to view the database: SHOW DATABASES

Text

Description automatically generated

* 1. Type the query to create the table with required attributes and their dtata types respectively: CREATE TABLE nyc\_sales(borough INT,neighborhood VARCHAR(100),building\_class\_category VARCHAR(100),block INT,lot INT,address VARCHAR(100),zipcode INT,residential\_units INT,commercial\_units INT,land\_squarefeet INT,gross\_sqaurefeet INT,tax\_class INT,building\_class VARCHAR(20),sale\_price INT,sale\_date TIMESTAMP,count VARCHAR(50)) ROW FORMAT DELIMITED FIELDS TERMINATED BY ','STORED AS TEXTFILE;
  2. Type the query to see the table: DESCRIBE nyc\_sales

A picture containing table

Description automatically generated

1. Next step is to load the data into the tablesby the following query:

load data local inpath '/tmp/result.txt' overwrite into table db.nyc\_sales

1. To check the number of lines loaded we run: Select count(\*) from db.nyc\_sales

Text

Description automatically generated

1. Once all the 9 files of NY property sales data are loaded into the HIVE table we use port number(10,000), “student” as username and password . We assign hose name as localhost to connect Hive in Knime using Hive connector to the Hive implementation on Hadoop
2. After that, we can see all of the lines that were loaded into the Hive table using Knime's DB query reader. As can be seen, the number of rows and columns accessed using Db query reader is identical to the number of lines loaded on hive.

Graphical user interface, table

Description automatically generated

1. We get rid of lines with count higher than 1 by using SELECT \* FROM 'db' to call just those lines that contain unique values. `nyc\_sales` WHERE COUNT=1 After then, the number of lines drops from 143564 to 141417.

A picture containing text, indoor

Description automatically generated

Table

Description automatically generated

## 3.3 Reflective analysis of data preparation in relational data warehouse vs Hadoop.

During the ETL process where we were preparing the data for data warehousing, the granularity of the fact table is determined. The data like year, month etc were extracted from the raw date. We pruned the columns as discussed in the architecture. We renamed the columns to match out architecture so that we don’t face errors while loading the data.

Transforming and loading data using Hadoop, on the other hand, was a little more difficult because each attribute had to be transformed and loaded into its suitable variable data type. MapIt reduction was used to do this. We can trim lines with a count greater than 1 by adding count as an extra column (to remove duplicates). We saved a small amount of rows/lines using Hadoop implementation since we did not undertake any extra pre-processing like groupby on the data (unlike the transform procedure used in warehousing). Line by line, the mapit function loads.

# 4. Reporting System

## 4.1 Relational Data Warehouse Implementation

BIRT (Business Intelligence and Reporting Tools) is an open-source reporting tool. The KNIME Report Designer extension incorporates BIRT into the KNIME Analytics Platform, allowing you to generate reports depending on the outcomes of your processes. In the coming paragraphs I will be giving an introduction for using the report node to answer our business questions.

The creation of reports is divided into two distinct tasks namely:

* First, one must prepare the data before reporting according to the requirements of our business questions. This step can be completed in the KNIME workflow.
* Second, once the required data is loaded, one can design, style, and alter the presentation of the data using the report template editor.

So, let us discuss each business question in detail:

1. What is sales trend of top three building categories over the years from 2005 to 2012?

To achieve this first we have to we have to prepare the data, we will use joiner node to get one singe table for analyzing. Before using the joiner node, we deploy the domain calculator node. Once we get the final table, we will use group by node to group the data by building category, year, and the count of sales price. Next, we will apply sorter node to sort the no of sales in ascending order. Then we will connect the node to reporting node. Next, we run the BIRT, to analyze our data. Now, we will insert a line chart in the report editor. Next, we have assign the x axis, y axis and y series grouping. Here my x axis is year, y axis is no of sales and building category is y series grouping. We then label the graph accordingly. Next, we use view report button to view our report. The below image is the report which can be used to answer our first question.

Chart, line chart

Description automatically generated

From the data we can analyze that building categories namely, ONE FAMILY HOME, TWO FAMILY HOME AND THREE FAMILY HOME have the highest number of sales. In the graph we can view their sales trend over few years. We can say that in 2009, there were less sale on these building categories. We see that there is a sudden hike in the sale after 2009 till 2010, after which there is a gradual decline in the number of sales.

1. How many total sales of residential units and commercial units over the years?

To achieve this first we have to we have to prepare the data, we will use joiner node to get one singe table for analyzing.  Before using the joiner node, we deploy the domain calculator node. Once we get the final table, we will use group by node to group the data by sum of residential units, sum of commercial units, year, and the count of sales. Then we will connect the node to reporting node. Next, we run the BIRT, to analyze our data.  Now, we will insert a bar chart in the report editor. Next, we have assigned the x axis, y axis and y series grouping. Here my x axis is year, y axis is no of sales and units is y series grouping. We then label the graph accordingly. Next, we use view report button to view our report. The below image is the report which can be used to answer our second question.

Chart, bar chart

Description automatically generated

From the data we can observe that, number of sales of residential units is always higher than that of commercial units. In the year 2005, we can observe the highest no of sales for residential units. In the year 2014 we can observe the highest no of sale for commercial units. 2012 has the least number of sales for both residential and commercial units.

1. Which Borough has the highest sales?

To achieve this first we have to we have to prepare the data, we will use joiner node to get one single table for analyzing.  Before using the joiner node, we deploy the domain calculator node. Once we get the final table, we will use group by node to group the data by borough and the count of sales. Next, we will use the sorter node to sort the data in ascending order as per the no of sales. Then we will connect the node to reporting node. Next, we run the BIRT, to analyze our data.  Now, we will insert a bar chart in the report editor. Next, we have assigned the x axis, y axis and y series grouping. Here my x axis is borough, y axis is no of sales. We then label the graph accordingly. Next, we use view report button to view our report. The below image is the report which can be used to answer our second question.

Chart, bar chart

Description automatically generated

From the data we can observe that, number of sales of for borough 4 is the highest followed closely by borough 1. Borough 2 has the least amount of sales.

1. What is the sales trend over months in different boroughs?

To achieve this first we have to we have to prepare the data, we will use joiner node to get one single table for analyzing.  Before using the joiner node, we deploy the domain calculator node. Once we get the final table, we will use group by node to group the data by borough and the count of sales and months. Then we will connect the node to reporting node. Next, we run the BIRT, to analyze our data.  Now, we will insert a line chart in the report editor. Next, we have assigned the x axis, y axis and y series grouping. Here my x axis is months, y axis is no of sales and boroughs is y series grouping. We then label the graph accordingly. Next, we use view report button to view our report. The below image is the report which can be used to answer our second question.

Chart, line chart

Description automatically generated

From the above graph we can see the trend of sales over months in different boroughs. It can be observed that in the 3rd the 6th month there is a slight increase in sales across all boroughs.

## 4.2 Hadoop Implementation

BIRT (Business Intelligence and Reporting Tools) is an open-source reporting tool. The KNIME Report Designer extension incorporates BIRT into the KNIME Analytics Platform, allowing you to generate reports depending on the outcomes of your processes. In the coming paragraphs I will be giving an introduction for using the report node to answer our business questions.

The creation of reports is divided into two distinct tasks namely:

* First, one must prepare the data before reporting according to the requirements of our business questions. This step can be completed in the KNIME workflow.
* Second, once the required data is loaded, one can design, style, and alter the presentation of the data using the report template editor.

The basic knime workflow for addressing my questions is as follows:

A picture containing text, map, indoor

Description automatically generated

We can examine all the lines that were imported into the Hive database using the DB query reader in Knime after building the Hive table and storing data. As can be seen, the number of rows and columns accessed using Db query reader is identical to the number of lines loaded on hive.

Graphical user interface, table

Description automatically generated

We get rid of lines with count higher than 1 by using SELECT \* FROM 'db' to call just those lines that contain unique values. `nyc\_sales` WHERE COUNT=1 After then, the number of lines drops from 143564 to 141417.

A picture containing text, indoor

Description automatically generated

A screenshot of a computer

Description automatically generated with low confidence

So, let us discuss each business question in detail:

* + - 1. What is sales trend of top three building categories over the years from 2005 to 2012?

To achieve this first we have to we have to prepare the data, we will use hive connector to call the data into knime. Then we use Date & time extracrting node to extract year, month, day of month, etc form the date. Once we get the final table, we will use group by node to group the data by building category, year, and the count of sales price. Next, we will apply sorter node to sort the no of sales in ascending order. Then we will connect the node to reporting node. Next, we run the BIRT, to analyze our data. Now, we will insert a line chart in the report editor. Next, we have assign the x axis, y axis and y series grouping. Here my x axis is year, y axis is no of sales and building category is y series grouping. We then label the graph accordingly. Next, we use view report button to view our report. The below image is the report which can be used to answer our first question.

Chart, line chart

Description automatically generated

From the data we can analyze that building categories namely, ONE FAMILY HOME, TWO FAMILY HOME AND THREE FAMILY HOME have the highest number of sales. In the graph we can view their sales trend over few years. We can say that in 2009, there were less sale on these building categories. We see that there is a sudden hike in the sale after 2009 till 2010, after which there is a gradual decline in the number of sales.

* + - 1. How many total sales of residential units and commercial units over the years?

To achieve this first we have to we have to prepare the data, we will use hive connector to call the data into knime. Then we use Date & time extracrting node to extract year, month, day of month, etc form the date. Once we get the final table, we will use group by node to group the data by sum of residential units, sum of commercial units, year, and the count of sales. Then we will connect the node to reporting node. Next, we run the BIRT, to analyze our data.  Now, we will insert a bar chart in the report editor. Next, we have assigned the x axis, y axis and y series grouping. Here my x axis is year, y axis is no of sales and units is y series grouping. We then label the graph accordingly. Next, we use view report button to view our report. The below image is the report which can be used to answer our second question.

Chart, bar chart

Description automatically generated

From the data we can observe that, number of sales of residential units is always higher than that of commercial units. In the year 2005, we can observe the highest no of sales for residential units. In the year 2011 we can observe the highest no of sale for commercial units. 2012 has the least number of sales for both residential and commercial units.

* + - 1. Which borough has the highest sales?

To achieve this first we have to we have to prepare the data, we will use hive connector to call the data into knime. Then we use Date & time extracrting node to extract year, month, day of month, etc form the date. Once we get the final table, we will use group by node to group the data by borough and the count of sales. Next, we will use the sorter node to sort the data in ascending order as per the no of sales. Then we will connect the node to reporting node. Next, we run the BIRT, to analyze our data.  Now, we will insert a bar chart in the report editor. Next, we have assigned the x axis, y axis and y series grouping. Here my x axis is borough, y axis is no of sales. We then label the graph accordingly. Next, we use view report button to view our report. The below image is the report which can be used to answer our second question.

Chart, bar chart

Description automatically generated

From the data we can observe that, number of sales of for borough 4 is the highest followed closely by borough 1. Borough 2 has the least amount of sales.

* + - 1. What is the sales trend over months in different boroughs?

To achieve this first we have to we have to prepare the data, we will use hive connector to call the data into knime. Then we use Date & time extracrting node to extract year, month, day of month, etc form the date. Once we get the final table, we will use group by node to group the data by borough and the count of sales and months. Then we will connect the node to reporting node. Next, we run the BIRT, to analyze our data.  Now, we will insert a line chart in the report editor. Next, we have assigned the x axis, y axis and y series grouping. Here my x axis is months, y axis is no of sales and boroughs is y series grouping. We then label the graph accordingly. Next, we use view report button to view our report. The below image is the report which can be used to answer our second question.

Chart, line chart

Description automatically generated

From the above graph we can see the trend of sales over months in different boroughs. It can be observed that in the 3rd the 6th month there is a slight increase in sales across all boroughs.

## 4.3 Reflective analysis of result in relational data warehouse vs Hadoop.

Data warehousing provides low latency and interactive reports. A data warehouse is a tool for combining and analyzing commercial company data from many sources. It serves as the brain of the BI system, which is designed to evaluate and report on data. In Hadoop implementation we observed the same graphs as that of data warehousing. While observing the 4th graph in both implementations, we can observe that the Hadoop implementation is better as the order of months in x axis is right.

# 5. Conclusions

Based on the steps we took to establish Hadoop and the warehouse, I feel Hadoop was less complicated than the warehouse since warehousing needed the development and loading of each dimension (only in allocated schema design), which was a tiresome and repeated operation. It was more involved and time consuming since warehousing also needed the creation of many dimension tables and the linking of the key of each dimension table to the fact table. Because the warehouse acts as a data sink, data must be converted and pre-processed before it can be used by users. As a result, users may conduct data reporting directly on pre-processed clean data. Both the implementations have their own advantages and disadvantages. They are listed below:

Data Warehousing:

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| Faster Data Retrieving | The data preparation is time consuming |
| Identifies errors and corrects them before loading | High cost of storage |
| It is easy to integrate with CRM systems | Data homogenization |
| The quality of data is better | Less agaile |
| Use a schema-on-write process | It is time-consuming to get data into the data warehouse using an ETL process. |
| Very less maintenance | Data quality is dependent on pre-load data purification; any processing error causes data to be corrupted indefinitely. |

Hadoop:

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| Can have varied data sources | The quality of data is poor comparatively |
| It is cost effective | Hadoop does not offer data quality solutions, hence it has less pre-processed cleaner data. |
| Low network traffic, | Read/write operations very expensive |
| Each user-submitted work is divided into a number of distinct sub-tasks, which are then assigned to data nodes, resulting in a little amount of code being moved to data. | Hadoop is suitable for a limited number of huge files, but it fails miserably when it comes to applications that deal with a large number of little files. |
| High throughput | It is unable to generate output in real-time with minimal delay. |
| Highly agile | Easily exploited by cyber criminals  as it is written in jave |