A project report on

SALES ORDER MANAGEMENT APPLICATION

Submitted in partial fulfillment of the requirements for the Degree of

B. Tech in Computer Science Engineering

by

Aditya Chaudhary (1805455)

under the guidance of

Sudakshina Chaudhury
Anirban Dasgupta
Shivansh Tekriwal
Akshit Kumar
Anjali Rawat

School of Computer Engineering
Kalinga Institute of Industrial Technology
Deemed to be University
Bhubaneswar

May 2021

ACKNOWLEDGEMENT

This project consumed huge amount of work, research and dedication. Still, implementation would not have been possible if we did not have the support of our POD leads and HighRadius Corporation. I would like to extend my sincere thanks to all of them.

I am highly indebted to HighRadius Corporation and KIIT University for their guidance and constant supervision as well as for providing necessary information regarding the project and also for their support in completing the project.

Nevertheless, I express my gratitude towards my family and colleagues for their kind co-operation and encouragement which helped me in completion of this project.

Signature: Aditya Chaudhary

Aditya Chaudhary

ABSTRACT

Machine Learning and Artificial Intelligence, being the hottest trend today, is incredibly powerful for predictions or calculated suggestions based on large amounts of data.

As businesses from many industries have begun to rely more heavily on machines to do the heavy lifting of A/R processes, we are able to deliver more value to their customers.

Keeping a steady cash flow is one of the biggest if not the biggest problem that Small to Medium Enterprises deal with daily. Within the different types of c ash flow, Accounts Receivable (AR) classifies the balance of money that needs to be paid by the company's customers.

However, this often does not happen before the aforementioned date, meaning that the invoice is often paid late. Intervention requires resources and over-intervention could cause unwanted customer dissatisfaction. Knowing whether an invoice is going to be paid late can be vital information. Current methods of late payment prediction focus only on the history between the seller and the buyer and are unusable when this history is not present.

My project SALES ORDER MANAGEMENT APP is a tool or platform that tracks sales, orders, inventory, and fulfillment as well as enables the user to keep track of all these under one platform and make it easier for these companies to deal with Account Receivables.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1 Purpose

The B2B world operates otherwise from the B2C or C2C world. Businesses work with different businesses on credit. Once a purchaser business orders product from the vendor business, the seller business problems an invoice for the same.

This invoice for the products contains numerous info like the details of the products purchased and once it ought to be paid. This is often better-known in accounting terminology as “Accounts Receivable”.

Seller business interacts with numerous businesses and sells product to any or all of them at various times.

Hence, the vendor business has to keep track of the whole quantity it owes from all the buyers. This involves keeping track of all invoices from all the buyers. Every invoice can have numerous important fields sort of a payment due date, invoice date, invoice amount, baseline date etc.

1.2 Objective

A Sales Order Management System is a computer software program that allows businesses to manage their Sales and inventory. Sales management systems help ensure more accurate inventory management, automatically entering new inventory into the system, tracking sales through different selling platforms, such as eBay and Amazon, and alerting you, the business owner, when your stock of a particular item drops low enough for a re-order.

A Sales Order Management system can also automate the Bill-to-cash process, beginning with the customer Bill through payment reconciliation, fulfillment, and shipment. Bill management software can work for both B2B and B2C businesses of any size.

Bill management software is also shareable, from the customer service team to the accounting team, the warehouse staff, and you, the business owner. The best kinds of inventory management systems also have a Sales Order management app, allowing you to manage your stock on the go and spot-check your business in real-time.

Effective Bill management improves the business workflow and increases the likelihood of repeat customers.

CHAPTER 2

BACKGROUND

2.1 Sales Order Management System

Most order management software programs follow a 6-step system, relying on automation to help employees accurately process and fulfill customer orders. When used properly, order management software produces a seamless flow from entering invoices through after sales follow-up. The smoother the software runs, the faster you can fulfill orders, and the less likely your customer receives the wrong item or experiences the frustration of back orders.

2.1.1. The order is placed

Your customer places an order through your third-party sales site, your own website, or over the phone with a live representative. Online, your customers will enter their details on a standardized form, with an option to have a securely saved preferred payment method.

To improve the sales process, make all fields of your online form mandatory so that you have all the necessary contact information for the customer up-front. This creates a customer profile and allows your order management system to track their purchase history, the volume of orders, and payment and delivery preferences. It also gives you their phone number and email address, should you need to follow up with service recovery.

The payment is processed, and then the order is sent to the warehouse once your software system approves the charges.

2.1.2. Warehouse processing

Once the order arrives at the warehouse, it's checked by the intake team and the item or items are "picked" from the stock. Having a SKU and bar code for every item increases the accuracy of fulfillment and makes it easier for pickers to simply scan the item and add it to the order.

If there isn't enough of the item(s) in stock to fulfill an order, then a purchase order is automatically placed through the order management software. You and the warehouse manager will receive an alert that there may be a delay in fulfillment. Your customer may receive an automatic notification of the delay, and the customer service team can follow up with your customer.

2.1.3. Reconciling the order

Next, the order is sent to the accounting department or preferably it should sync automatically with your cloud accounting software, where it's recorded in your A/R ledger. The sale is logged and a receipt sent to your client. Automating your sales ledgers makes it easier for auditing, inventory reconciliation, and end-of-year taxes.

2.1.4. Shipping the order

Once the order is picked from the warehouse, your packing team will double-check for accuracy, again using the barcodes and SKU. Then, the order is packed carefully and shipped via a third-party delivery system. Your customer will receive a notice through the order management system that their order has shipped, along with a tracking number and estimated delivery time. As a store owner, you can also track the progress of shipped orders, which can be helpful if there are special needs orders, such as re-deliveries, VIP orders, or unusually large ones.

2.1.5. Post-sales follow-up

Once the order arrives, the software should generate an automatic email to follow up, asking how they liked the items and ensuring that they received everything accurately. This email should include detailed instructions on how to reach customer service if there are any issues, taking the frustration out of guessing how to obtain a refund if needed.

Your customer service team oversees this process, thanking the customer for their business or working with them for a refund or replacement.

2.1.6. Special order oversight

Another aspect of good OMS is the ability to flag a special order. This may be a return replacement or it could be a VIP order that includes a free thank-you gift or special coupon. When these orders are placed through the system, the software can flag them with a code, allowing you or your customer retention team to personally monitor the order for accuracy.

2.2 Regression Analysis

In statistical modeling, regression analysis is a set of statistical processes for estimating the relationships between a dependent variable and one or more independent variables. The most common form of regression analysis is linear regression, in which one finds the line that most closely fits the data according to a specific mathematical criterion. For example, the method of ordinary least squares computes the unique line that minimizes the sum of squared differences between the true data and that line. For specific mathematical reasons, this allows the researcher to estimate the conditional expectation of the dependent variable when the independent variables take on a given set of values. Less common forms of regression use slightly different procedures to estimate alternative location parameter estimate the conditional expectation across a broader collection of non-linear models.

Regression analysis is primarily used for two conceptually distinct purposes. First, regression analysis is widely used for prediction and forecasting, where its use has substantial overlap with the field of machine learning. Second, in some situations regression analysis can be used to infer causal relationships between the independent and dependent variables. Importantly, regressions by themselves only reveal relationships between a dependent variable and a collection of independent variables in a fixed dataset. To use regressions for prediction or to infer causal relationships, respectively, a researcher must carefully justify why existing relationships have predictive power for a new context or why a relationship between two variables has a causal interpretation. The latter is especially important when researchers hope to estimate causal relationships using observational data.

2.2.1 Linear Regression

In statistics, linear regression is a linear approach to modelling the relationship between a scalar response and one or more explanatory variables (also known as dependent and independent variables). The case of one explanatory variable is called simple linear regression; for more than one, the process is called multiple linear regression. This term is distinct from multivariate linear regression, where multiple correlated dependent variables are predicted, rather than a single scalar variable.

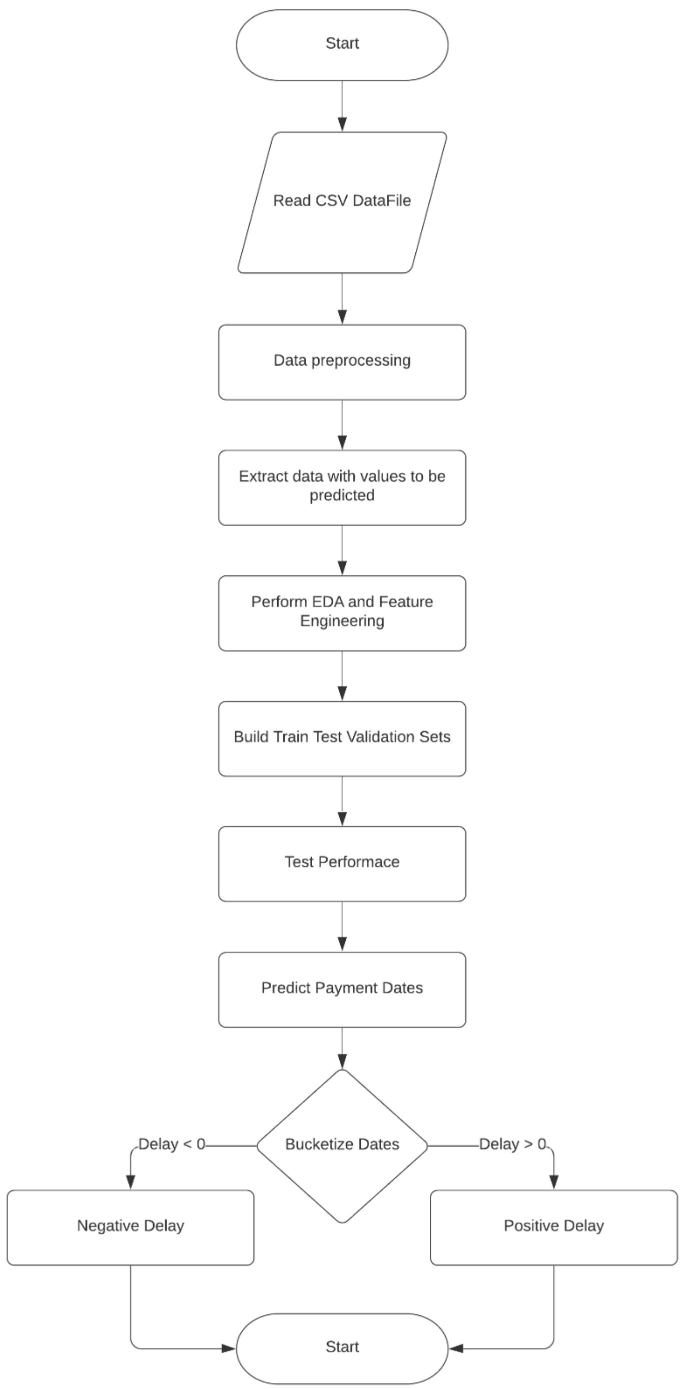
In linear regression, the relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called linear models. Most commonly, the conditional mean of the response given the values of the explanatory variables (or predictors) is assumed to be an affine function of those values; less commonly, the conditional median or some other quantile is used. Like all forms of regression analysis, linear regression focuses on the conditional probability distribution of the response given the values of the predictors, rather than on the joint probability distribution of all of these variables, which is the domain of multivariate analysis.

Linear regression was the first type of regression analysis to be studied rigorously, and to be used extensively in practical applications. This is because models which depend linearly on their unknown parameters are easier to fit than models which are non-linearly related to their parameters and because the statistical properties of the resulting estimators are easier to determine.

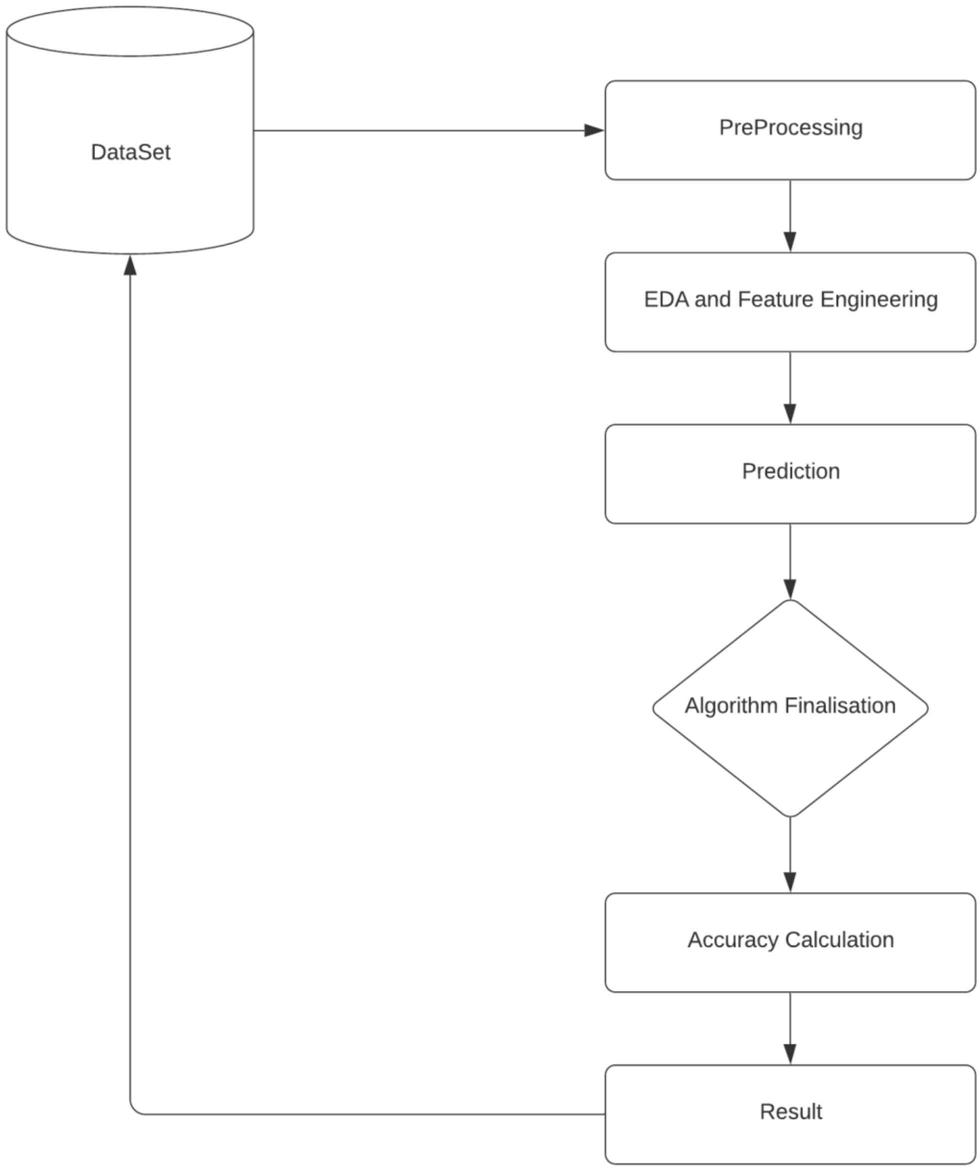
CHAPTER 3

PROJECT ANALYSIS/ PROJECT IMPLEMENTATION

3.1 Project Flow Chart



3.2 Data Flow Diagram



3.3 Project Implementation

3.3.1 Machine Learning Implementation

3.3.1.1 Data Preprocessing

1. Data Preprocessing

1.1 Taking care of missing data and dropping unneccessary columns and rows

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
df = pd.read_csv('1805455.csv') #Loading the data into a pandas dataframe
```

In [3]:
```
print(df.shape)
print(df.columns)
```

```
(50000, 19)
Index(['business_code', 'cust_number', 'name_customer', 'clear_date',
       'buisness_year', 'doc_id', 'posting_date', 'document_create_date',
       'document_create_date.1', 'due_in_date', 'invoice_currency',
       'document type', 'posting_id', 'area_business', 'total_open_amount',
       'baseline_create_date', 'cust_payment_terms', 'invoice_id', 'isOpen'],
      dtype='object')
```

In [4]:
```
df.head()
```

Out[4]:

	business_code	cust_number	name_customer	clear_date	buisness_year	doc_id	posting_date	document_create_date	document_create_date.1	due_
0	CA02	0140105686	SYSC co	2020-03-12 00:00:00	2020.0	2.960616e+09	2020-02-24	20200224	20200224	202
1	U001	0200769623	WAL-MAR foundation	2019-07-12 00:00:00	2019.0	1.929540e+09	2019-06-26	20190625	20190626	201
2	U001	0200148860	DOLLA co	2019-12-10 00:00:00	2019.0	1.930188e+09	2019-11-24	20191123	20191124	201
3	U001	0200416837	DEC foundation	2019-12-10 00:00:00	2019.0	1.930227e+09	2019-12-06	20191206	20191206	201
4	U001	0200769623	WAL-MAR in	2019-06-26 00:00:00	2019.0	1.929490e+09	2019-06-14	20190614	20190614	201

In [5]:
```
df.isna().sum()
```

Out[5]:
```
business_code                 0
cust_number                   0
name_customer                 0
clear_date                 4176
buisness_year                 0
doc_id                        0
posting_date                  0
document_create_date          0
document_create_date.1        0
due_in_date                   0
invoice_currency              0
document type                 0
posting_id                    0
area_business             50000
total_open_amount             0
baseline_create_date          0
cust_payment_terms            0
invoice_id                    6
isOpen                        0
dtype: int64
```

In [6]:
```
df.sort_values(by = 'document_create_date.1', inplace = True)
df.reset_index(inplace = True, drop = True)
```

In [7]:
```
unknown_data = df[df.clear_date.isnull()]
df = df[df.clear_date.notnull()]
print(unknown_data.shape)
print(df.shape)
```

```
(4176, 19)
(45824, 19)
```

In [8]:
```
df.drop(['area_business', 'name_customer'], axis = 1, inplace = True)
```

Since all values of area business columns are NANs, we can drop this column. Also, we have a unique customer id for each customer. So we can drop this as well

In [9]:
```
df.drop(df[df.total_open_amount < 0].index, inplace = True)
```

In [10]:
```
print(df.doc_id.unique().shape)
print(df.invoice_id.unique().shape)
print(df.shape)
```

In [10]:
print(df.doc_id.unique().shape)
print(df.invoice_id.unique().shape)
print(df.shape)

```
(45824,)
(45819,)
(45824, 17)
```

Here we observe that there is a unique doc_id for each entry so we can safely discard this column. Invoice id is also supposed to be unique. But, we can see that there are duplicate values here and they should be removed.

In [11]:
```
df = df.drop_duplicates(subset = 'invoice_id').reset_index(drop = True)
df.drop(['doc_id','invoice_id'], axis = 1, inplace = True)
```

In [12]:
```
print(df['document type'].value_counts())
print(df.posting_id.unique())
print(df.isOpen.unique())
```

```
RV    45818
X2        1
Name: document type, dtype: int64
[1.]
[0]
```

We can drop the columns 'posting_id' and 'isOpen' because they take only one unique value. There is only one X2 document type and rest are RV. So we need to drop the row with X2 type document and then drop the column 'document_type' altogether

In [13]:
```
df.drop(df[df['document type'] == 'X2'].index, inplace = True)
df.drop(['document type', 'posting_id', 'isOpen'], axis = 1, inplace = True)
```

In [14]:
```
df.head()
```

Out[14]:

	business_code	cust_number	clear_date	buisness_year	posting_date	document_create_date	document_create_date.1	due_in_date	invoice_currency	total_
0	U001	0200769623	2019-01-09 00:00:00	2019.0	2018-12-30	20181229	20181230	20190114.0	USD	
1	U001	0200744019	2019-01-18 00:00:00	2019.0	2018-12-30	20181229	20181230	20190114.0	USD	
2	U001	0200704045	2019-01-14 00:00:00	2019.0	2018-12-30	20181230	20181230	20190114.0	USD	
3	U001	0200759878	2019-01-15 00:00:00	2019.0	2018-12-30	20181229	20181230	20190114.0	USD	
4	U001	0200418007	2019-01-15 00:00:00	2019.0	2018-12-30	20181231	20181230	20190114.0	USD	

jupyter DoctorOctopus_1805455_EDA,FE,FS & Modeling Assignment. Last Checkpoint: 03/21/2021 (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

In [15]:
```
df.tail()
```

Out[15]:

	business_code	cust_number	clear_date	buisness_year	posting_date	document_create_date	document_create_date.1	due_in_date	invoice_currency
45814	U001	0200759878	2020-03-19 00:00:00	2020.0	2020-02-27	20200226	20200227	20200313.0	USD
45815	U001	0200722444	2020-03-18 00:00:00	2020.0	2020-02-27	20200219	20200227	20200330.0	USD
45816	U001	0200956366	2020-03-06 00:00:00	2020.0	2020-02-27	20200227	20200227	20200313.0	USD
45817	U001	0200759878	2020-03-13 00:00:00	2020.0	2020-02-27	20200226	20200227	20200313.0	USD
45818	U001	0200357714	2020-03-13 00:00:00	2020.0	2020-02-27	20200227	20200227	20200313.0	USD

Some of the values of clear date were not given (null). We have to predict these values. So storing that data into another dataframe known as unknown_data. The clear dates of the unknown_data will be predicted at the end when the model is trained.

So far I have:

1. Loaded the dataset into a pandas dataframe.
2. Had a peek at the data.
3. Removed the irrelevant columns. Also removed rows with duplicate invoice id values.
4. Changed the relevant column into datetime format.
5. Removed the rows with missing values of target variable and stored it into a different pandas dataframe (unknown_data).

1.2 Categorical data

In [16]:
```
print(df.invoice_currency.value_counts())
print(df.business_code.value_counts())
print(df.cust_payment_terms.value_counts())
print(df.cust_number.value_counts())
print(df.buisness_year.value_counts())
```

```
USD    42141
CAD     3677
Name: invoice_currency, dtype: int64
U001    41393
CA02     3677
U013      593
U002      141
U005       12
U007        2
Name: business_code, dtype: int64
NAA8    18435
NAH4    12061
CA10     3587
NAC6     1530
NAM4     1241
        ...
NATW        1
NAUX        1
NANC        1
NACE        1
NATM        1
Name: cust_payment_terms, Length: 74, dtype: int64
0200769623    10175
0200726979     1715
0200762301     1323
0200759878     1215
0200794332      972
              ...
100008001         1
0100014735        1
0200092114        1
0200644847        1
0200958768        1
Name: cust_number, Length: 1411, dtype: int64
2019.0    39871
2020.0     5947
Name: buisness_year, dtype: int64
```

Here we can conclude that:

1. Two types of currencies are used. We should convert all CAD amounts to USD and then label encode invoice_currency column.
2. We can one hot encode the business_code column.
3. We can label encode the cust_payment terms column.
4. We can label encode the cust_number column.

In [17]:
```
conv_rate_2019 = 0.753598
conv_rate_2020 = 0.74652
conv_rate_2021 = 0.786105
```

In [18]:
```
def currency_convertor(data):
    for index in data.index:
        if(data.loc[index,'invoice_currency'] == 'CAD'):
            if(data.loc[index,'buisness_year'] == 2019):
                data.loc[index,'total_open_amount'] = data.iloc[index].total_open_amount * conv_rate_2019
            if(data.loc[index,'buisness_year'] == 2020):
                data.loc[index,'total_open_amount'] = data.iloc[index].total_open_amount * conv_rate_2020
            if(data.loc[index,'buisness_year'] == 2021):
                data.loc[index,'total_open_amount'] = data.iloc[index].total_open_amount * conv_rate_2021
    return data
```

In [19]:
```
df = currency_convertor(df)
```

In [20]:
```
from sklearn.preprocessing import LabelEncoder

le_1 = LabelEncoder()
le_2 = LabelEncoder()
le_3 = LabelEncoder()

df['cust_payment_terms_1'] = df['cust_payment_terms']  # A separate column for EDA. Will be dropped in training

df['invoice_currency'] = le_1.fit_transform(df['invoice_currency'])
df['cust_payment_terms'] = le_2.fit_transform(df['cust_payment_terms'])
df['cust_number'] = le_3.fit_transform(df['cust_number'])

df.head()
```

Out[20]:

	business_code	cust_number	clear_date	buisness_year	posting_date	document_create_date	document_create_date.1	due_in_date	invoice_currei
0	U001	868	2019-01-09 00:00:00	2019.0	2018-12-30	20181229	20181230	20190114.0	
1	U001	814	2019-01-18 00:00:00	2019.0	2018-12-30	20181229	20181230	20190114.0	
2	U001	714	2019-01-14 00:00:00	2019.0	2018-12-30	20181230	20181230	20190114.0	
3	U001	846	2019-01-15 00:00:00	2019.0	2018-12-30	20181229	20181230	20190114.0	
4	U001	596	2019-01-15 00:00:00	2019.0	2018-12-30	20181231	20181230	20190114.0	

In [21]:
```
df['business_code_1'] = df['business_code']
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder',OneHotEncoder(),[0])], remainder= 'passthrough')
df = np.array(ct.fit_transform(df))
```

1.3 Preprocessing of date time columns

In [23]:
```
df['clear_date'] = pd.to_datetime(df['clear_date'])
df['posting_date'] = pd.to_datetime(df['posting_date'])
df['document_create_date.1'] = pd.to_datetime(df['document_create_date.1'], errors='ignore', format='%Y%m%d')
df['due_in_date'] = pd.to_datetime(df['due_in_date'],format='%Y%m%d')
df['baseline_create_date'] = pd.to_datetime(df['baseline_create_date'],format = '%Y%m%d')
```

In [24]:
```
df.drop('document_create_date', axis = 1, inplace = True)
```

In [25]:
```
document_create_date_1 = df['document_create_date.1']
df['delay'] = (df['clear_date'] - df['due_in_date']).dt.days
#This will be the dependent variable which the model will predict.
df['month_of_due_date'] = df['due_in_date'].dt.month
df['clear_date'] = (df['clear_date'] - document_create_date_1).dt.days
df['posting_date'] = (df['posting_date'] - document_create_date_1).dt.days
df['baseline_create_date'] = (df['baseline_create_date'] - document_create_date_1).dt.days
df['due_in_days'] = (df['due_in_date'] - document_create_date_1).dt.days
df.head()
```

Out[25]:

	0	1	2	3	4	5	cust_number	clear_date	buisness_year	posting_date	...	due_in_date	invoice_currency	total_open_amount	baseline_create_date
0	0	1	0	0	0	0	868	10	2019	0	...	2019-01-14	1	1078.89	0
1	0	1	0	0	0	0	814	19	2019	0	...	2019-01-14	1	12839.5	0
2	0	1	0	0	0	0	714	15	2019	0	...	2019-01-14	1	106472	0
3	0	1	0	0	0	0	846	16	2019	0	...	2019-01-14	1	51464.6	0
4	0	1	0	0	0	0	596	16	2019	0	...	2019-01-14	1	49711.3	0

5 rows × 21 columns

Now we need to remove the anomalies.

In [26]:
```
df.drop(df[df.clear_date < df.baseline_create_date].index, inplace = True)
df.drop(df[df.due_in_days <= df.baseline_create_date].index, inplace = True)
df.drop(df[df.due_in_days <= 0].index, inplace = True)
df.drop(df[df.baseline_create_date < 0].index, inplace = True)
df.drop(df[df.clear_date <= 0].index, inplace = True)
df.reset_index(inplace = True, drop = True)
```

In [27]:
```
print(df['posting_date'].value_counts())
```

```
0    43021
Name: posting_date, dtype: int64
```

```
In [29]: df.dtypes
```

```
2                                  object
3                                  object
4                                  object
5                                  object
cust_number                        object
clear_date                          int64
buisness_year                      object
document_create_date.1     datetime64[ns]
due_in_date                datetime64[ns]
invoice_currency                   object
total_open_amount                  object
baseline_create_date                int64
cust_payment_terms                 object
cust_payment_terms_1               object
business_code_1                    object
delay                               int64
month_of_due_date                   int64
due_in_days                         int64
dtype: object
```

```
In [30]: df['0'] = df['0'].astype(int)
df['1'] = df['1'].astype(int)
df['2'] = df['2'].astype(int)
df['3'] = df['3'].astype(int)
df['4'] = df['4'].astype(int)
df['5'] = df['5'].astype(int)
df['cust_number'] = df['cust_number'].astype(int)
df['buisness_year'] = df['buisness_year'].astype(int)
df['total_open_amount'] = df['total_open_amount'].astype(int)
df['invoice_currency'] = df['invoice_currency'].astype(int)
df['cust_payment_terms'] = df['cust_payment_terms'].astype(int)
df.dtypes
```

```
Out[30]: 0                                  int32
1                                  int32
2                                  int32
3                                  int32
4                                  int32
5                                  int32
cust_number                        int32
clear_date                         int64
buisness_year                      int32
document_create_date.1     datetime64[ns]
due_in_date                datetime64[ns]
invoice_currency                   int32
total_open_amount                  int32
baseline_create_date               int64
cust_payment_terms                 int32
cust_payment_terms_1              object
business_code_1                   object
delay                              int64
month_of_due_date                  int64
due_in_days                        int64
dtype: object
```

3.3.1.2 Splitting of data into Train, Test and Validation Sets

1.4 Splitting into Train Test and Val

```
In [31]: df.shape
Out[31]: (43021, 20)
```

```
In [32]: from sklearn.model_selection import train_test_split
training_set, test_set = train_test_split(df, test_size= 0.3, shuffle = False)
val, test = train_test_split(test_set, test_size = 0.5, shuffle = False)
```

```
In [33]: print(training_set.shape)
print(val.shape)
print(test.shape)

(30114, 20)
(6453, 20)
(6454, 20)
```

```
In [34]: training_set.columns
Out[34]: Index(['0', '1', '2', '3', '4', '5', 'cust_number', 'clear_date',
        'buisness_year', 'document_create_date.1', 'due_in_date',
        'invoice_currency', 'total_open_amount', 'baseline_create_date',
        'cust_payment_terms', 'cust_payment_terms_1', 'business_code_1',
        'delay', 'month_of_due_date', 'due_in_days'],
       dtype='object')
```

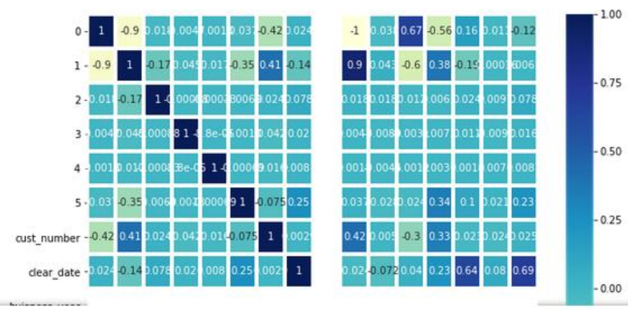
3.3.1.3 Exploratory Data Analysis

2 EDA

Exploratory Data Analysis

In [35]:
```
fig,ax = plt.subplots(figsize=(10,10))
sns.heatmap(training_set.corr(),annot=True,cmap="YlGnBu",vmin=-1,vmax=1,linewidths=3,ax=ax)
```

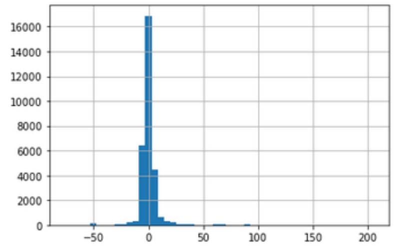
Out[35]: <AxesSubplot:>



In [36]:
```
training_set['delay'].describe()
```

Out[36]:
```
count    30114.000000
mean         0.680913
std         10.494053
min        -76.000000
25%         -2.000000
50%          0.000000
75%          2.000000
max        205.000000
Name: delay, dtype: float64
```

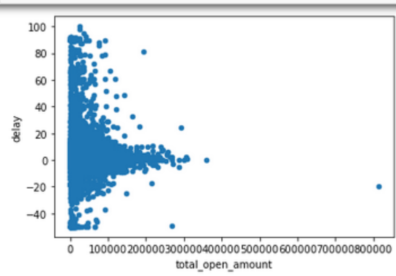
In [37]:
```
training_set['delay'].hist(bins=50)
plt.show()
```



It is essential to remove the outliers for the model, so it can learn efficiently.

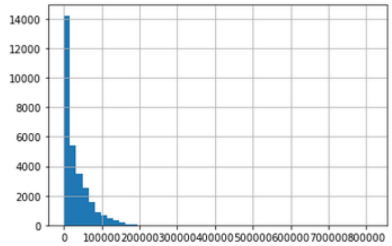
In [38]:
```
training_set.drop(training_set[training_set.delay > 100 ].index, inplace = True)
training_set.drop(training_set[training_set.delay < -50 ].index, inplace = True)
training_set.reset_index(inplace = True, drop = True)
```

In [39]:
```
training_set.plot(kind = 'scatter',  x= 'total_open_amount', y= 'delay')
plt.show()
training_set.plot(kind = 'scatter',  x= 'due_in_days', y= 'delay')
plt.show()
training_set.plot(kind = 'scatter',  x= 'baseline_create_date', y= 'delay')
plt.show()
training_set.plot(kind = 'scatter',  x= 'cust_payment_terms', y= 'delay')
plt.show()
training_set.plot(kind = 'scatter',  x= 'month_of_due_date', y= 'delay')
plt.show()
```



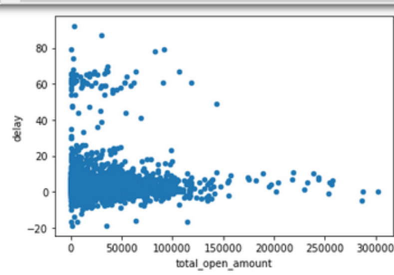
There is no clear cut effect of due date and baseline create on delay. There are delays in short term invoices as well as long term invoices. Thus, due date and baseline create date are not important features to understand delay. These column will be dropped during training and predicting. Month of due date has some effect on delay and will be used in the model

In [40]:
```
training_set['total_open_amount'].hist(bins = 50)
plt.show()
```



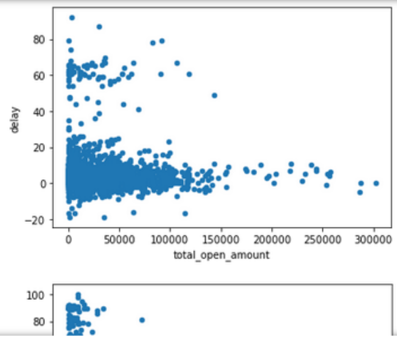
We can infer that a large number of invoices have a small total open amount. As total open amount increases, number of data points decrease.

In [41]:
```
training_set[training_set['business_code_1'] == 'CA02'].plot(kind = 'scatter', x = 'total_open_amount', y = 'del
plt.show()
training_set[training_set['business_code_1'] == 'U001'].plot(kind = 'scatter', x = 'total_open_amount', y = 'del
plt.show()
training_set[training_set['business_code_1'] == 'U013'].plot(kind = 'scatter', x = 'total_open_amount', y = 'del
plt.show()
training_set[training_set['business_code_1'] == 'U002'].plot(kind = 'scatter', x = 'total_open_amount', y = 'del
plt.show()
```



4. The delay in U002 is also less and the amounts are cleared faster.

In [42]:
```
training_set[training_set['invoice_currency'] == 0].plot(kind = 'scatter', x = 'total_open_amount', y = 'delay')
plt.show()
training_set[training_set['invoice_currency'] == 1].plot(kind = 'scatter', x = 'total_open_amount', y = 'delay')
plt.show()
```



The invoice currency plot for CAD is the same as the plot for business type CA02. Hence the effect of currency is incorporated in the business type column. Hence we will drop this column in training the model

In [43]:
```
training_set.baseline_create_date.value_counts()
```

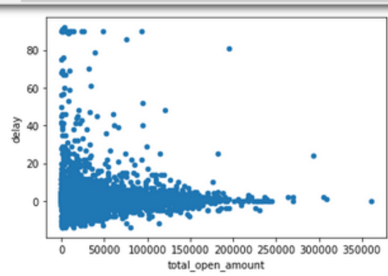
Out[43]:
```
0      27931
1        819
2        303
8        257
7        174
9        144
4         77
6         68
3         67
5         65
10        60
11        15
12         9
13         2
Name: baseline_create_date, dtype: int64
```

Generally, baseline create date is on the same day as document create date. Thus the payment window generally opens on the same day on which the document is created.

In [44]:
```
training_set.cust_payment_terms_1.value_counts()
```

Out[44]:
```
NAA8     12876
NAH4      8257
CA10      2498
NAC6      1094
NAAX       752
NAD1       607
NAU5       586
NAG2       554
NAGD       542
NA32       516
NA10       264
NAD5       257
NAVE       181
NAVF       170
NAAW       137
NAWN       108
NAVR        79
NAVQ        75
NAUZ        53
```

In [45]:
```
training_set[training_set['cust_payment_terms_1'] == 'NAA8'].plot(kind = 'scatter', x = 'total_open_amount', y = 'd
plt.show()
training_set[training_set['cust_payment_terms_1'] == 'NAH4'].plot(kind = 'scatter', x = 'total_open_amount', y = 'd
plt.show()
training_set[training_set['cust_payment_terms_1'] == 'CA10'].plot(kind = 'scatter', x = 'total_open_amount', y = 'd
plt.show()
```



There are many types of cust payment terms which have a very few data points. Hence it is not enough to be a relevant feature in the model. Also if we look at the curves of delay vs amount for the top three most abundant cust payment terms, we find that there is not much difference in the variance, max and min values of delay. Hence we should drop this column as well while training the model and predicting the outputs.

Hence by EDA we can conclude the important features which we need to train the model.

1. Business Code
2. Month of due date
3. Total Open Amount

3.3.1.4 Splitting data into dependent and independent variables.

3. Splitting into dependent and independent variables.

In [46]:
```
training_set.columns
```

Out[46]:
```
Index(['0', '1', '2', '3', '4', '5', 'cust_number', 'clear_date',
       'buisness_year', 'document_create_date.1', 'due_in_date',
       'invoice_currency', 'total_open_amount', 'baseline_create_date',
       'cust_payment_terms', 'cust_payment_terms_1', 'business_code_1',
       'delay', 'month_of_due_date', 'due_in_days'],
      dtype='object')
```

Y_train is the delay column of training set.

Y_val is the delay column of validation set.

Y_test is the delay column of test set.

In [47]:
```
Y_train = training_set.iloc[:,-3].values
Y_val = val.iloc[:,-3].values
Y_test = test.iloc[:,-3].values
training_set.columns
```

Out[47]:
```
Index(['0', '1', '2', '3', '4', '5', 'cust_number', 'clear_date',
       'buisness_year', 'document_create_date.1', 'due_in_date',
       'invoice_currency', 'total_open_amount', 'baseline_create_date',
       'cust_payment_terms', 'cust_payment_terms_1', 'business_code_1',
       'delay', 'month_of_due_date', 'due_in_days'],
      dtype='object')
```

In [48]:
```
training_set.drop(['cust_number', 'clear_date', 'buisness_year', 'document_create_date.1', 'due_in_date',
       'invoice_currency', 'baseline_create_date',
       'cust_payment_terms', 'cust_payment_terms_1', 'business_code_1','delay',
        'due_in_days'], axis = 1, inplace= True)
val.drop(['cust_number', 'clear_date', 'buisness_year', 'document_create_date.1', 'due_in_date',
       'invoice_currency', 'baseline_create_date',
       'cust_payment_terms', 'cust_payment_terms_1', 'business_code_1', 'delay',
        'due_in_days'], axis = 1, inplace= True)
test.drop(['cust_number', 'clear_date', 'buisness_year', 'document_create_date.1', 'due_in_date',
       'invoice_currency', 'baseline_create_date',
       'cust_payment_terms', 'cust_payment_terms_1', 'business_code_1', 'delay',
        'due_in_days'], axis = 1, inplace= True)
training_set.columns
```

Out[48]:
```
Index(['0', '1', '2', '3', '4', '5', 'total_open_amount', 'month_of_due_date'], dtype='object')
```

3.3.1.5 Feature Scaling

4 Feature Scaling (not required for Tree based models)

We are using Standard Scaler for Feature Scaling.

In [49]:
```
from sklearn.preprocessing import StandardScaler
sc_X_train = StandardScaler()
sc_Y_train = StandardScaler()
X_train_scaled = sc_X_train.fit_transform(training_set)
Y_train_scaled = sc_Y_train.fit_transform(Y_train.reshape(-1,1))
```

3.3.1.5 Training with Multiple Linear Regression (MLR)

5. Training different models and checking RMSE.

In [50]:
```
from sklearn.metrics import mean_squared_error
```

5.1 Multiple Linear Regression

In [51]:
```
from sklearn.linear_model import LinearRegression
regressor_linear = LinearRegression()
regressor_linear.fit(X_train_scaled, Y_train_scaled)
```

Out[51]:
```
LinearRegression()
```

Explaining the first Line here,

Y_pred_val1_linear is the variable to store our predictions for val set Since linear model takes in standardised inputs, we use sc_x_train to first standardise the val set (innermost bracket). Then we use the predict functions to predict the delays.. but the output is standardised.. to bring it back to original scale we use inverse_transform function... And we need to inverse transform the standardised delay predictions so we use sx_y_train object's inverse transform method.

The next 2 lines are a way to print our predictions and original delays side by side.

In [52]:
```
Y_pred_val_linear = sc_Y_train.inverse_transform(regressor_linear.predict(sc_X_train.transform(val)))
np.set_printoptions(precision=2)
print(np.concatenate((Y_pred_val_linear.reshape(len(Y_pred_val_linear),1), Y_val.reshape(len(Y_val),1)),1))
```

```
[[-0.19  1.  ]
 [-0.2   1.  ]
 [-0.24  1.  ]
 ...
 [-0.38  0.  ]
 [-0.43  3.  ]
 [-0.35 -4.  ]]
```

In [53]:
```
print(np.sqrt(mean_squared_error(Y_val, Y_pred_val_linear))) #RMSE on val set
```

```
5.319850060842998
```

In [54]:
```
Y_pred_test_linear = sc_Y_train.inverse_transform(regressor_linear.predict(sc_X_train.transform(test)))
print(np.sqrt(mean_squared_error(Y_test, Y_pred_test_linear))) #RMSE on test set
```

```
9.06535461390192
```

In [55]:
```
Y_pred_test_linear.max()
```

Out[55]:
```
9.59997193434786
```

This model is greatly overfitting on our training set.

3.3.1.6 Training with Support Vector Regression (SVR)

5.2 SVR

In [56]:
```
from sklearn.svm import SVR
regressor_SVR = SVR(kernel = 'rbf')
regressor_SVR.fit(X_train_scaled,Y_train_scaled.ravel())
```

Out[56]:
```
SVR()
```

In [57]:
```
Y_pred_val_SVR = sc_Y_train.inverse_transform(regressor_SVR.predict(sc_X_train.transform(val)))
```

In [58]:
```
print(np.sqrt(mean_squared_error(Y_val,Y_pred_val_SVR))) #RMSE on val set
```

```
5.2657784888500485
```

In [59]:
```
Y_pred_test_SVR = sc_Y_train.inverse_transform(regressor_SVR.predict(sc_X_train.transform(test)))
print(np.sqrt(mean_squared_error(Y_test, Y_pred_test_SVR))) #RMSE on test set
```

```
9.04877966383206
```

In [60]:
```
Y_pred_test_SVR.max()
```

Out[60]:
```
10.102822730552566
```

Poor performance on test set. Also the range of predictions is low. Max delay is just 10 days, which is very unlikely to occur.

3.3.1.7 Training with Decision Tree Regression (D-Tree Regression)

5.3 Decision Tree Regression

In [61]:
```
from sklearn.tree import DecisionTreeRegressor
regressor_dt = DecisionTreeRegressor(random_state = 0)
regressor_dt.fit(training_set, Y_train)
```

Out[61]:
```
DecisionTreeRegressor(random_state=0)
```

In [62]:
```
Y_pred_val_dt = regressor_dt.predict(val)
print(np.sqrt(mean_squared_error(Y_val, Y_pred_val_dt)))  #RMSE on val set
```

```
17.451042605757483
```

As, the validation set is only giving such high RMSE, the test will be worse, so we are discarding this method and not even going to bother using the test set.

3.3.1.8 Training with Random Forest Regression (RF-Regression)

5.5 Random Forest Regression

In [63]:
```
from sklearn.ensemble import RandomForestRegressor
regressor_rf = RandomForestRegressor(n_estimators = 30, random_state = 0)
regressor_rf.fit(training_set, Y_train)
```

Out[63]:
```
RandomForestRegressor(n_estimators=30, random_state=0)
```

In [64]:
```
Y_pred_val_rf = regressor_rf.predict(val)
print(np.sqrt(mean_squared_error(Y_val,Y_pred_val_rf))) #RMSE on val set
```

```
12.041050951044634
```

In [65]:
```
Y_pred_test_rf = regressor_rf.predict(test)
print(np.sqrt(mean_squared_error(Y_test,Y_pred_test_rf))) #RMSE on test set
```

```
10.903386093707242
```

In [66]:
```
Y_pred_test_rf.max()     #Here, we are checking the max value we got from test.
```

Out[66]:
```
67.26666666666667
```

Conclusion: I am selecting Random Forest Regression Model to predict the unknown data.

<u>Conclusion</u> : Since <u>Random Forest Regression</u> algorithm, gave the best scores we will be using this algorithm to predict the payment dates for our unknown data.

3.3.1.10 Prediction of our unknown data using Random Forest Regression

6. Predicting the Unknown Data.

In [67]:
```
unknown_data.columns
```

Out[67]:
```
Index(['business_code', 'cust_number', 'name_customer', 'clear_date',
       'buisness_year', 'doc_id', 'posting_date', 'document_create_date',
       'document_create_date.1', 'due_in_date', 'invoice_currency',
       'document type', 'posting_id', 'area_business', 'total_open_amount',
       'baseline_create_date', 'cust_payment_terms', 'invoice_id', 'isOpen'],
      dtype='object')
```

In [68]:
```
unknown_data_1 = unknown_data.copy()
```

In [69]:
```
unknown_data_1['cust_payment_terms_1'] = unknown_data_1['cust_payment_terms']
unknown_data_1.drop(['name_customer', 'doc_id', 'document type', 'posting_id', 'area_business', 'invoice_id', 'isOpen'],
                  axis = 1, inplace = True)
```

In [70]:
```
unknown_data_1['business_code_1'] = unknown_data_1['business_code']
unknown_data_1 = np.array(ct.transform(unknown_data_1))
unknown_data_1 = pd.DataFrame(unknown_data_1,columns=['0', '1', '2', '3', '4', '5', 'cust_number', 'clear_date',
       'buisness_year', 'posting_date', 'document_create_date',
       'document_create_date.1', 'due_in_date', 'invoice_currency', 'total_open_amount',
       'baseline_create_date', 'cust_payment_terms',
       'cust_payment_terms_1', 'Business_code_1'])
```

In [71]:
```
unknown_data_1.reset_index(inplace= True, drop= True)
unknown_data_1 = currency_convertor(unknown_data_1)
```

In [72]:
```
unknown_data_1['due_in_date'] = pd.to_datetime(unknown_data_1['due_in_date'],format='%Y%m%d')
unknown_data_1['month_of_due_date'] = unknown_data_1['due_in_date'].dt.month
```

In [73]:
```
unknown_data_1.drop(['cust_number', 'clear_date', 'buisness_year', 'posting_date', 'document_create_date',
                     'document_create_date.1', 'due_in_date', 'invoice_currency', 'baseline_create_date', 'cust_payment_terms',
                     'cust_payment_terms_1', 'Business_code_1'], axis = 1, inplace= True)
unknown_data_1.columns
```

Out[73]:
```
Index(['0', '1', '2', '3', '4', '5', 'total_open_amount', 'month_of_due_date'], dtype='object')
```

In [74]:
```
predictions = regressor_rf.predict(unknown_data_1)
predictions = np.round(predictions)  #rounding off our predicted delays, as days can not be decimal values.
```

In [75]:
```
unknown_data['due_in_date'] = pd.to_datetime(unknown_data['due_in_date'],format='%Y%m%d')
```

In [76]:
```
unknown_data.reset_index(inplace = True, drop = True)
```

In [74]:
```
predictions = regressor_rf.predict(unknown_data_1)
predictions = np.round(predictions)  #rounding off our predicted delays, as days can not be decimal values.
```

In [75]:
```
unknown_data['due_in_date'] = pd.to_datetime(unknown_data['due_in_date'],format='%Y%m%d')
```

In [76]:
```
unknown_data.reset_index(inplace = True, drop = True)
```


In [77]:
```
for index in unknown_data.index:    #calculation of our clear date by iterating and adding predicted delay to due dates.
    unknown_data.loc[index, 'clear_date'] = (unknown_data.iloc[index].due_in_date + pd.Timedelta(predictions[index])).date()
```

In [78]:
```
unknown_data['bucket'] = predictions
```

3.3.1.11 Bucketization of final predicted data

Here we are bucketizing our output based on our bucketing conditions.

Buckets Summary

Cleared before due date [< 0 days]

Bucket 1 [0-15 days]

Bucket 2 [16-30 days]

Bucket 3 [31-45 days]

Bucket 4 [46-60 days]

Bucket 5 [Greater than 60 days]

In [79]:
```
for index in unknown_data.index:
    if unknown_data.loc[index,'bucket'] < 0:
        unknown_data.loc[index, 'bucket'] = 'Cleared before due date              :'
    elif unknown_data.loc[index,'bucket'] <=15:
        unknown_data.loc[index,'bucket'] = 'Bucket 1 [0-15 days]                  :'
    elif unknown_data.loc[index,'bucket'] <= 30:
        unknown_data.loc[index,'bucket'] = 'Bucket 2 [16-30 days]                 :'
    elif unknown_data.loc[index,'bucket'] <= 45:
        unknown_data.loc[index,'bucket'] = 'Bucket 3 [31-45 days]                 :'
    elif unknown_data.loc[index,'bucket'] <= 60:
        unknown_data.loc[index,'bucket'] = 'Bucket 4 [46-60 days]                 :'
    else:
        unknown_data.loc[index,'bucket'] = 'Bucket 5 [Greater than 60 days]       :'
```

Final Output with Buckets.

In [80]:
```
print('        Buckets                              Number of invoices')
unknown_data.bucket.value_counts()
```

```
        Buckets                              Number of invoices
```

Out[80]:
```
Bucket 1 [0-15 days]                  :    2462
Cleared before due date               :    1622
Bucket 2 [16-30 days]                 :      52
Bucket 3 [31-45 days]                 :      19
Bucket 4 [46-60 days]                 :      18
Bucket 5 [Greater than 60 days]       :       3
Name: bucket, dtype: int64
```

The output here is sorted because .value_counts() returns by the highest frequency

Final Output with Buckets.

In [80]:
```
print('        Buckets                              Number of invoices')
unknown_data.bucket.value_counts()
```

```
        Buckets                              Number of invoices
```

Out[80]:
```
Bucket 1 [0-15 days]                  :    2462
Cleared before due date               :    1622
Bucket 2 [16-30 days]                 :      52
Bucket 3 [31-45 days]                 :      19
Bucket 4 [46-60 days]                 :      18
Bucket 5 [Greater than 60 days]       :       3
Name: bucket, dtype: int64
```

The output here is sorted because .value_counts() returns by the highest frequency

---

Submitted by : Aditya Chaudhary

Roll Number : 1805455

High Radius Tech Track Machine Learning Project.

3.3.2 Java Backend Implementation

3.3.2.1 Java POJO Class implementation and JDBC Registration

```java
package read.CSV;

public class DataSource {

    public static String dbuserName = "root";
    public static String dbpassword = "root";
    public static String jdbcDriver = "com.mysql.cj.jdbc.Driver";
    public static String urlh2h = "jdbc:mysql://localhost:3306/h2h_internship";


    public static String insertQuery = "insert into invoice_details1 (business_code,cust_number, name_customer, clear_date, business_year,doc_id,posting_date,  document_create_date,document_create_date_1,  due_in_date, invoice_curre
            "                     document_type, posting_id,  area_business, total_open_amount,\r\n" +
            "                     baseline_create_date, cust_payment_terms, invoice_id, isOpen) values (?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?);";

}
```

```java
package read.CSV;


import java.sql.Date;
public class CSVPojo {
    private String buisness_code;
    private String cust_number;
    private String name_customer;
    private Date clear_date;
    private Integer business_year;
    private Long doc_id;
    private Date posting_date;
    private Date document_create_date;
    private Date document_create_date_1;
    private Date due_in_date;
    private String invoice_currency;
    private String document_type;
    private Integer posting_id;
    private String area_business;
    private Float total_open_amount;
    private Date baseline_create_date;
    private String cust_payment_terms;
    private Long invoice_id;
    private Integer isOpen;




    public CSVPojo(String buisness_code, String cust_number, String name_customer, Date clear_date,
            Integer business_year, Long doc_id, Date posting_date, Date document_create_date,
            Date document_create_date_1, Date due_in_date, String invoice_currency, String document_type,
            Integer posting_id, String area_business, Float total_open_amount, Date baseline_create_date,
            String cust_payment_terms, Long invoice_id, Integer isOpen) {
        super();
        this.buisness_code = buisness_code;
        this.cust_number = cust_number;
        this.name_customer = name_customer;
        this.clear_date = clear_date;
        this.business_year = business_year;
        this.doc_id = doc_id;
        this.posting_date = posting_date;
        this.document_create_date = document_create_date;
        this.document_create_date_1 = document_create_date_1;
        this.due_in_date = due_in_date;
        this.invoice_currency = invoice_currency;
    public CSVPojo(String buisness_code, String cust_number, String name_customer, Date clear_date,
            Integer business_year, Long doc_id, Date posting_date, Date document_create_date,
            Date document_create_date_1, Date due_in_date, String invoice_currency, String document_type,
            Integer posting_id, String area_business, Float total_open_amount, Date baseline_create_date,
            String cust_payment_terms, Long invoice_id, Integer isOpen) {
        super();
        this.buisness_code = buisness_code;
        this.cust_number = cust_number;
        this.name_customer = name_customer;
        this.clear_date = clear_date;
        this.business_year = business_year;
        this.doc_id = doc_id;
        this.posting_date = posting_date;
        this.document_create_date = document_create_date;
        this.document_create_date_1 = document_create_date_1;
        this.due_in_date = due_in_date;
        this.invoice_currency = invoice_currency;
        this.document_type = document_type;
        this.posting_id = posting_id;
        this.area_business = area_business;
        this.total_open_amount = total_open_amount;
        this.baseline_create_date = baseline_create_date;
        this.cust_payment_terms = cust_payment_terms;
        this.invoice_id = invoice_id;
        this.isOpen = isOpen;
    }


    @Override
    public String toString() {
        return "CSVPojo [buisness_code=" + buisness_code + ", cust_number=" + cust_number + ", name_customer="
                + name_customer + ", clear_date=" + clear_date + ", business_year=" + business_year + ", doc_id="
                + doc_id + ", posting_date=" + posting_date + ", document_create_date=" + document_create_date
                + ", document_create_date_1=" + document_create_date_1 + ", due_in_date=" + due_in_date
                + ", invoice_currency=" + invoice_currency + ", document_type=" + document_type + ", posting_id="
                + posting_id + ", area_business=" + area_business + ", total_open_amount=" + total_open_amount
                + ", baseline_create_date=" + baseline_create_date + ", cust_payment_terms=" + cust_payment_terms
                + ", invoice_id=" + invoice_id + ", isOpen=" + isOpen + "]";
    }


    public String getBuisness_code() { return buisness_code; }
    public void setBuisness_code(String buisness_code) { this.buisness_code = buisness_code; }
    public String getCust_number() { return cust_number; }
    public void setCust_number(String cust_number) { this.cust_number = cust_number; }
```

```java
  public String getBuisness_code() { return buisness_code; }
  public void setBuisness_code(String buisness_code) { this.buisness_code = buisness_code; }
  public String getCust_number() { return cust_number; }
  public void setCust_number(String cust_number) { this.cust_number = cust_number; }
  public String getName_customer() { return name_customer; }
  public void setName_customer(String name_customer) { this.name_customer = name_customer; }
  public Date getClear_date() { return clear_date; }
  public void setClear_date(Date clear_date) { this.clear_date = clear_date; }
  public Integer getBusiness_year() { return business_year; }
  public void setBusiness_year(Integer business_year) { this.business_year = business_year; }
  public Long getDoc_id() { return doc_id; }
  public void setDoc_id(Long doc_id) { this.doc_id = doc_id; }
  public Date getPosting_date() { return posting_date; }
  public void setPosting_date(Date posting_date) { this.posting_date = posting_date; }
  public Date getDocument_create_date() { return document_create_date; }
  public void setDocument_create_date(Date document_create_date) { this.document_create_date = document_create_date; }
  public Date getDocument_create_date_1() { return document_create_date_1; }
  public void setDocument_create_date_1(Date document_create_date_1) {
      this.document_create_date_1 = document_create_date_1;
  }
  public Date getDue_in_date() { return due_in_date; }
  public void setDue_in_date(Date due_in_date) { this.due_in_date = due_in_date; }
  public String getInvoice_currency() { return invoice_currency; }
  public void setInvoice_currency(String invoice_currency) { this.invoice_currency = invoice_currency; }
  public String getDocument_type() { return document_type; }
  public void setDocument_type(String document_type) { this.document_type = document_type; }
  public Integer getPosting_id() { return posting_id; }
  public void setPosting_id(Integer posting_id) { this.posting_id = posting_id; }
  public String getArea_business() { return area_business; }
  public void setArea_business(String area_business) { this.area_business = area_business; }
  public Float getTotal_open_amount() { return total_open_amount; }
  public void setTotal_open_amount(Float total_open_amount) { this.total_open_amount = total_open_amount; }
  public Date getBaseline_create_date() { return baseline_create_date; }
  public void setBaseline_create_date(Date baseline_create_date) { this.baseline_create_date = baseline_create_date; }
  public String getCust_payment_terms() { return cust_payment_terms; }
  public void setCust_payment_terms(String cust_payment_terms) { this.cust_payment_terms = cust_payment_terms; }
  public Long getInvoice_id() { return invoice_id; }
  public void setInvoice_id(Long invoice_id) { this.invoice_id = invoice_id; }
  public Integer getIsOpen() { return isOpen; }
  public void setIsOpen(Integer isOpen) { this.isOpen = isOpen; }

}
ackage read.CSV;

mport java.io.BufferedReader;
mport java.io.FileReader;
mport java.sql.Connection;
mport java.sql.Date;
mport java.sql.DriverManager;
mport java.sql.PreparedStatement;
mport java.sql.ResultSet;
mport java.sql.Types;
mport java.util.ArrayList;
mport java.util.List;

mport java.text.SimpleDateFormat;
mport java.text.ParseException;

ublic class Main
{
 public static Connection conn;
 public static PreparedStatement stmt;
 public static ResultSet rs;

 static long start = System.currentTimeMillis();
 public static void main(String[] args) throws Exception {
   List < CSVPojo > pojo = new ArrayList < > (); //Creating an arrayList of type CSVPojo.
   int count = 0; //Variable to count number of rows present in CSV file.

   String line = "";

   String splitBy = ",";
   try {
     Class.forName(DataSource.jdbcDriver).newInstance(); //Register JDBC driver.
     conn = DriverManager.getConnection(DataSource.urlh2h, DataSource.dbuserName, DataSource.dbpassword); //Open a connection.
     conn.setAutoCommit(false); // Set auto-commit to false

     BufferedReader br = new BufferedReader(new FileReader("1805455.csv")); //parsing a CSV file into BufferedReader class constructor
     br.readLine(); //Skipping first row as it contains name of column.
     while ((line = br.readLine()) != null) //returns a Boolean value
     {
       String[] invoices = line.split(splitBy); // use comma as separator
       for (int i = 0; i < invoices.length; i++) { //Column which has no data like area_buisnesss , null will be stored in that column.
         if (invoices[i] == "")
           invoices[i] = null;
       }
       CSVPojo poj = getDetails(invoices); //Calling Function getDetails()
       pojo.add(poj); //adding CSVPojo object to POJO list.
```

```java
      count++;
    }

    stmt = conn.prepareStatement(DataSource.insertQuery); // Create PrepareStatement object
    for (CSVPojo i: pojo) {
      // usage of getter and setter functions.
      // Set the variables using getter from CSVPojo.
      stmt.setString(1, i.getBuisness_code());
      stmt.setString(2, i.getCust_number());
      stmt.setString(3, i.getName_customer());
      stmt.setDate(4, i.getClear_date());
      stmt.setInt(5, i.getBusiness_year());
      stmt.setLong(6, i.getDoc_id());
      stmt.setDate(7, i.getPosting_date());
      stmt.setDate(8, i.getDocument_create_date());
      stmt.setDate(9, i.getDocument_create_date_1());
      stmt.setDate(10, i.getDue_in_date());
      stmt.setString(11, i.getInvoice_currency());
      stmt.setString(12, i.getDocument_type());
      stmt.setInt(13, i.getPosting_id());
      stmt.setString(14, i.getArea_business());
      stmt.setFloat(15, i.getTotal_open_amount());
      stmt.setDate(16, i.getBaseline_create_date());
      stmt.setString(17, i.getCust_payment_terms());
      if (i.getInvoice_id() != null) //if not contains null values directly set variables.
        stmt.setLong(18, i.getInvoice_id());
      else
        stmt.setNull(18, Types.NULL); // if null values then convert it to SQl NULL .
      stmt.setInt(19, i.getIsOpen());
      stmt.addBatch();
    }

    stmt.executeBatch();
    conn.commit();
    conn.close();
    br.close();
    System.out.println("Congratulations!! " + count + " number of rows, have been inserted into the SQL Database.");

  } catch (Exception e) {
    e.printStackTrace();
  }

  System.out.println("Execution Succesfull");
  long end = System.currentTimeMillis();
  float sec = (end - start) / 1000F;
  System.out.println(sec + " seconds"); //finding the time difference and converting it into seconds

}

private static CSVPojo getDetails(String[] invoices) throws ParseException {

  //Our CSV file contains all values in alphanumeric format i.e. String format.
  //Declaring data type for variable.
  Date clear_date, posting_date, document_create_date, due_in_date, baseline_create_date, document_create_date_1;
  Long invoice_id, doc_id;
  Integer isOpen, posting_id, buisness_year;
  Float total_open_amount;

  try {
    SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-yyyy"); //clear date is in dd-MM-yyyy form.
    java.util.Date date = sdf.parse(invoices[3]); //util date
    clear_date = new java.sql.Date(date.getTime()); //Sql date.
  } catch (ParseException e) {
    clear_date = null;
  }

  try {
    buisness_year = Integer.parseInt(invoices[4]); //String to integer.
  } catch (NumberFormatException e) { //Catch NumberFormatException
    buisness_year = null;
  }

  try {
    doc_id = Long.parseLong(invoices[5]); //String to Long.
  } catch (NumberFormatException e) {
    doc_id = null;
  }
  try {
    SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-yyyy"); //posting_date is in dd-MM-yyyy format.
    java.util.Date date = sdf.parse(invoices[6]); //Util date
    posting_date = new java.sql.Date(date.getTime()); //sql date
  } catch (ParseException e) {
    //Handle parseException
    posting_date = null;
  }
  try {
    SimpleDateFormat sdf = new SimpleDateFormat("yyyymmdd"); //document_create_date is in yyyymmdd format.
    java.util.Date date = sdf.parse(invoices[7]); //Util date
    document_create_date = new java.sql.Date(date.getTime()); //sql date
```

```java
  } catch (ParseException e) {
    //Handle parseException
    document_create_date = null;
  }
  try {
    SimpleDateFormat sdf = new SimpleDateFormat("yyyymmdd"); //document_create_date_1 is in yyyymmdd format.
    java.util.Date date = sdf.parse(invoices[8]); //util date
    document_create_date_1 = new java.sql.Date(date.getTime()); //sql date
  } catch (ParseException e) {
    //Handle parseException
    document_create_date_1 = null;
  }
  try {
    SimpleDateFormat sdf = new SimpleDateFormat("yyyymmdd");
    java.util.Date date = sdf.parse(invoices[9]); //util date
    due_in_date = new java.sql.Date(date.getTime()); //sql date
  } catch (ParseException e) {
    due_in_date = null;
  }

  try {
    posting_id = Integer.parseInt(invoices[12]); //string to integer.
  } catch (NumberFormatException e) {
    posting_id = null;
  }

  try {
    total_open_amount = Float.parseFloat(invoices[14]); //string to Float.
  } catch (NumberFormatException e) {
    total_open_amount = null;
  }
  try {
    SimpleDateFormat sdf = new SimpleDateFormat("yyyymmdd");
    java.util.Date date = sdf.parse(invoices[15]);
    baseline_create_date = new java.sql.Date(date.getTime());
  } catch (ParseException e) {
    //Handle parseException
    baseline_create_date = null;
  }

  try {
    invoice_id = Long.parseLong(invoices[17]); //string to Long.
  } catch (NumberFormatException e) {
    invoice_id = null;
  }

  try {
    total_open_amount = Float.parseFloat(invoices[14]); //string to Float.
  } catch (NumberFormatException e) {
    total_open_amount = null;
  }
  try {
    SimpleDateFormat sdf = new SimpleDateFormat("yyyymmdd");
    java.util.Date date = sdf.parse(invoices[15]);
    baseline_create_date = new java.sql.Date(date.getTime());
  } catch (ParseException e) {
    //Handle parseException
    baseline_create_date = null;
  }

  try {
    invoice_id = Long.parseLong(invoices[17]); //string to Long.
  } catch (NumberFormatException e) {
    invoice_id = null;
  }

  try {
    isOpen = Integer.parseInt(invoices[18]); //string to Integer.
  } catch (NumberFormatException e) {
    isOpen = null;
  }

  //invoices[0],invoices[1],invoices[2],invoices[10],invoices[11],invoices[13],invoices[16] are already in string format ,so we don't need to convert it.

  //Using CSVPojo constructor to set values in POJO.

  CSVPojo poj = new CSVPojo(invoices[0], invoices[1], invoices[2], clear_date, buisness_year,
    doc_id, posting_date, document_create_date, document_create_date_1, due_in_date, invoices[10],
    invoices[11], posting_id, invoices[13], total_open_amount,
    baseline_create_date, invoices[16], invoice_id, isOpen);
  return poj; //returning  object.
}
```

3.3.2.2 Java Servlets

3.3.2.2.1 Data Add Servlet

```java
package Servlets;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;



@WebServlet("/DataAdd")
public class DataAdd extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public DataAdd() {
        super();

    }


    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

        PrintWriter out = response.getWriter(); //object of response to send.
        String Name = request.getParameter("name");
        Date date=null;
        SimpleDateFormat formatter = new SimpleDateFormat("EEE MMM dd yyyy HH:mm:ss zzz", Locale.ENGLISH);
        try {
            date = formatter.parse(request.getParameter("date"));
        } catch (ParseException e) {
            e.printStackTrace();
        }
        String invId = request.getParameter("invoice");
        String CustNum = request.getParameter("custNum");
        String Amount = request.getParameter("amount");
```

```java
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

    PrintWriter out = response.getWriter(); //object of response to send.
    String Name = request.getParameter("name");
    Date date=null;
    SimpleDateFormat formatter = new SimpleDateFormat("EEE MMM dd yyyy HH:mm:ss zzz", Locale.ENGLISH);
    try {
        date = formatter.parse(request.getParameter("date"));
    } catch (ParseException e) {
        e.printStackTrace();
    }
    String invId = request.getParameter("invoice");
    String CustNum = request.getParameter("custNum");
    String Amount = request.getParameter("amount");
    String notes = request.getParameter("notes");
    String docId=invId;
    AddData(Name,date,invId,CustNum,Amount,notes,docId);

    response.setContentType("application/json");
    response.setCharacterEncoding("UTF-8");
    System.out.println("Succesfully Inserted");
    out.flush();
}


private void AddData(String Name, Date date,String invId,String CustNum,String Amount,String notes,String docId) {

    final String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
    final String DB_URL = "jdbc:mysql://localhost/h2h_internship";
    final String USER = "root";
    final String PASS = "root";

    Connection conn = null;
    java.sql.Statement stmt = null;

    //Writing our SQL Query.

    String sql = "INSERT INTO invoice_details (name_customer,"
            + "due_in_date,invoice_id,total_open_amount,cust_number,notes,doc_id)"
            + "values (?,?,?,?,?,?,?)";

    try{
        Class.forName(JDBC_DRIVER);
        conn = DriverManager.getConnection(DB_URL,USER,PASS);
```

```java
private void AddData(String Name, Date date,String invId,String CustNum,String Amount,String notes,String docId) {

    final String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
    final String DB_URL = "jdbc:mysql://localhost/h2h_internship";
    final String USER = "root";
    final String PASS = "root";

    Connection conn = null;
    java.sql.Statement stmt = null;

    //Writing our SQL Query.

    String sql = "INSERT INTO invoice_details (name_customer,"
            + "due_in_date,invoice_id,total_open_amount,cust_number,notes,doc_id)"
            + "values (?,?,?,?,?,?,?)";

    try{
        Class.forName(JDBC_DRIVER);
        conn = DriverManager.getConnection(DB_URL,USER,PASS);
        conn.setAutoCommit(true);
        stmt = conn.createStatement();
        PreparedStatement statement = conn.prepareStatement(sql);

            //Setting our statements into our SQL query to be executed.

            statement.setString(1,Name);
            statement.setString(2, change_date(date));
            statement.setLong(3, Long.parseLong(invId));
            statement.setLong(7, Long.parseLong(docId));
            statement.setDouble(4, Double.parseDouble(Amount));
            statement.setInt(5, Integer.parseInt(CustNum));
            statement.setString(6, notes);
            statement.execute();

            System.out.println(statement);

            stmt.close();
            conn.close();
    }catch(SQLException se){
        se.printStackTrace();
    }catch(Exception e){
        e.printStackTrace();
    }finally{
        try{
        if(stmt!=null)
        stmt.close();
```

```java
        stmt.close();
        }catch(SQLException se2){
        }
        try{
        if(conn!=null)
        conn.close();
    }catch(SQLException se){
        se.printStackTrace();
        }
    }

}


@SuppressWarnings("deprecation")
private static String change_date(Date date){
    String Finaldate = "";
    Finaldate += (date.getYear()+1900);
    String month = Integer.toString(date.getMonth()+1);
    if(month.length() == 1)
        Finaldate += "0";
    Finaldate += month;
    String day = Integer.toString(date.getDate());
    if(day.length()==1)
        Finaldate += "0";
    Finaldate += day;
    return Finaldate;
}



protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    doGet(request, response);
}
```

3.3.2.2.2 Data Delete Servlet

```java
package Servlets;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/DataDelete")
public class DataDelete extends HttpServlet {
    private static final long serialVersionUID = 1L;


    public DataDelete() { super(); }



    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        PrintWriter out= response.getWriter();

        String invId = request.getParameter("invoice");
        System.out.println(invId);
        String[] invoices=invId.split(",");
        DeleteData(invoices);

        response.setContentType("application/json");
        response.setCharacterEncoding("UTF-8");
        System.out.println("Succesfully Deleted");
        out.flush();
    }


    private void DeleteData(String[] invIds) {
        final String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
        final String DB_URL = "jdbc:mysql://localhost/h2h_internship";
        final String USER = "root";
        final String PASS = "root";

        Connection conn = null;
        java.sql.Statement stmt = null;
            try{
```

```java
            try{
            Class.forName(JDBC_DRIVER);
            conn = DriverManager.getConnection(DB_URL,USER,PASS);
            conn.setAutoCommit(true);
            stmt = conn.createStatement();
            for(String id:invIds) {
            String sql= "DELETE FROM `invoice_details` WHERE invoice_id="+id+";"; //SQL query to delete.

            stmt.executeUpdate(sql);
            }
                stmt.close();
                conn.close();
        }catch(SQLException se){
            se.printStackTrace();
        }catch(Exception e){
            e.printStackTrace();
        }finally{
            try{
            if(stmt!=null)
            stmt.close();
            }catch(SQLException se2){
            }
            try{
            if(conn!=null)
            conn.close();
        }catch(SQLException se){
            se.printStackTrace();
            }
        }

    }



    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        doGet(request, response);
    }

}
```

3.3.2.2.3 Data Edit Servlet

```java
package Servlets;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/DataEdit")
public class DataEdit extends HttpServlet {
    private static final long serialVersionUID = 1L;


    public DataEdit() { super(); }



    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        PrintWriter out= response.getWriter();

        String invId = request.getParameter("invoice");
        System.out.println(invId);
        String[] invoices=invId.split(",");
        String Amount = request.getParameter("amount");
        String notes = request.getParameter("notes");
        System.out.println(invoices[0]+Amount+notes);
        EditData(invoices,Amount,notes);

        response.setContentType("application/json");
        response.setCharacterEncoding("UTF-8");
        System.out.println("Succesfully Editted");
        out.flush();
    }


    private void EditData(String[] invIds, String amount, String notes) {
        final String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
        final String DB_URL = "jdbc:mysql://localhost/h2h_internship";
        final String USER = "root";
        final String PASS = "root";
```

```java
    private void EditData(String[] invIds, String amount, String notes) {
        final String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
        final String DB_URL = "jdbc:mysql://localhost/h2h_internship";
        final String USER = "root";
        final String PASS = "root";

        Connection conn = null;
        java.sql.Statement stmt = null;
            try{
            Class.forName(JDBC_DRIVER);
            conn = DriverManager.getConnection(DB_URL,USER,PASS);
            conn.setAutoCommit(true);
            stmt = conn.createStatement();
            String sql = "UPDATE `invoice_details` SET total_open_amount= ?," //sql query to update our document.
                    + "notes= ? WHERE invoice_id = ?";
            PreparedStatement statement = conn.prepareStatement(sql);
            for(String id:invIds) {
            if(notes.isEmpty())
            notes="";

            //setting our data to our query.

            statement.setLong(3, Long.parseLong(id));
            statement.setDouble(1, Double.parseDouble(amount));
            statement.setString(2, notes);

            System.out.println(sql);
            statement.executeUpdate();
            }
                stmt.close();
                conn.close();
        }catch(SQLException se){
            se.printStackTrace();
        }catch(Exception e){
            e.printStackTrace();
        }finally{
            try{
            if(stmt!=null)
            stmt.close();
            }catch(SQLException se2){
            }
            try{
            if(conn!=null)
            conn.close();
        }catch(SQLException se){
            se.printStackTrace();
```

```java
            }
        }

    }



    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        doGet(request, response);
    }

}
```

3.3.2.2.4 Data Fetch Servlet

```java
package Servlets;

import ...


@WebServlet("/DataFetch")
public class DataFetch extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public DataFetch() { super(); }


//Creating ArrayList of POJO Class.

    private ArrayList<invoice_fetch> fetchMyData(int start,int total){
        final String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
        final String DB_URL = "jdbc:mysql://localhost/h2h_internship";
        final String USER = "root";
        final String PASS = "root";

        ArrayList<invoice_fetch> dataList = new ArrayList<invoice_fetch>();
        Connection conn = null;
        java.sql.Statement stmt = null;

        try{
            Class.forName(JDBC_DRIVER);
            conn = DriverManager.getConnection(DB_URL,USER,PASS);
            stmt = conn.createStatement();
            String sql = "SELECT cust_number,due_in_date,invoice_id,name_customer,total_open_amount,notes FROM `invoice_details` " //creating our sql query to execute.
                    + "limit "+(start-1)+","+total;
            ResultSet rs = stmt.executeQuery(sql);
            while(rs.next()){
                invoice_fetch NewInvoice = new invoice_fetch();

                NewInvoice.setCust_number(rs.getString("cust_number"));
                NewInvoice.setDue_in_date(rs.getDate("due_in_date"));
                NewInvoice.setInvoice_id(rs.getLong("invoice_id"));
                NewInvoice.setName_customer(rs.getString("name_customer"));
                NewInvoice.setTotal_open_amount(rs.getDouble("total_open_amount"));
                System.out.println();

                if(rs.getString("notes")==null)  //this is for the notes section of our UI.
                NewInvoice.setNotes("Lorem Ipsum dolor sit amet");
                else
                NewInvoice.setNotes(rs.getString("notes"));
```

```java
                if(rs.getString("notes")==null)  //this is for the notes section of our UI.
                NewInvoice.setNotes("Lorem Ipsum dolor sit amet");
                else
                NewInvoice.setNotes(rs.getString("notes"));
                dataList.add(NewInvoice);

            }
            rs.close();
            stmt.close();
            conn.close();
        }
        catch(SQLException se){
            se.printStackTrace();
        }catch(Exception e){
            e.printStackTrace();
        }finally{
            try{
            if(stmt!=null)
            stmt.close();
            }catch(SQLException se2){
            }
            try{
            if(conn!=null)
            conn.close();
        }catch(SQLException se){
            se.printStackTrace();
            }
        }
        return dataList;

    }


    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        PrintWriter out= response.getWriter();

        String stringPageId=request.getParameter("page");
        int PageId=Integer.parseInt(stringPageId);

        //using a concept like pagination to send 15 values per fetch per page.
        int value=15;
        if(PageId!=1) {
            PageId=PageId-1;
            PageId=PageId*value+1;
        }
```

```java
        //Converting to Json using Gson.

        Gson gson = new Gson();
        String data = gson.toJson(fetchMyData(PageId,value));
        response.setContentType("application/json");
        response.setCharacterEncoding("UTF-8");
        out.print(data);
        System.out.println(data);
        out.flush();
    }

}
```

3.3.2.2.5 Data Search Servlet

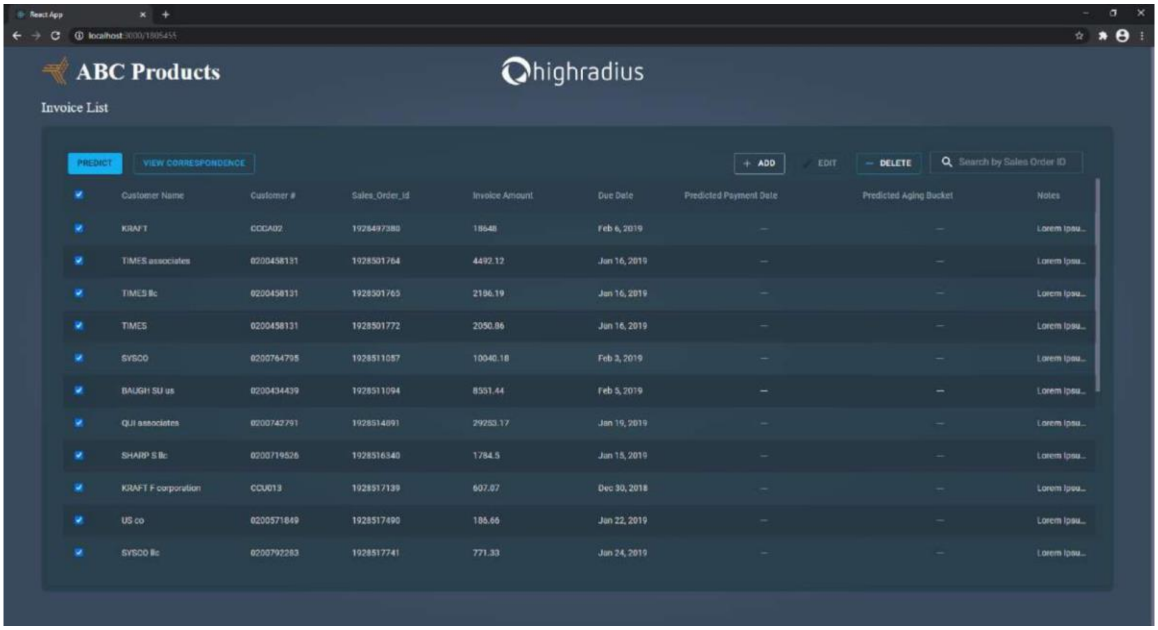
```java
package Servlets;

import ...

@WebServlet("/DataSearch")
public class DataSearch extends HttpServlet {
    private static final long serialVersionUID = 1L;


    public DataSearch() { super(); }


    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        PrintWriter out= response.getWriter();


        String stringPageId="1";  // used to limit the data's to only top 15 values, which we have passed for the search bar field.

        int PageId=Integer.parseInt(stringPageId);

        int value=15;

        if(PageId!=1) {
            PageId=PageId-1;
            PageId=PageId*value+1;
        }

        String invId = request.getParameter("invoice");
        Gson gson = new Gson();
        String data = gson.toJson(SearchData(invId,PageId,value));
        response.setContentType("application/json");
        response.setCharacterEncoding("UTF-8");
        out.print(data);
        System.out.println(data);
        out.flush();

    }


    private ArrayList<invoice_fetch> SearchData(String invId,int start,int total) {

        final String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
        final String DB_URL = "jdbc:mysql://localhost/h2h_internship";
        final String USER = "root";
        final String PASS = "root";

        ArrayList<invoice_fetch> dataList = new ArrayList<invoice_fetch>();
        Connection conn = null;
        java.sql.Statement stmt = null;
            try{
            Class.forName(JDBC_DRIVER);
            conn = DriverManager.getConnection(DB_URL,USER,PASS);
            conn.setAutoCommit(true);
            stmt = conn.createStatement();

            String sql= "Select * from `invoice_details`" +
                    "where invoice_id like '"+invId+"%' limit "+(start-1)+","+total+";";  //SQL query for search.

            ResultSet rs = stmt.executeQuery(sql);
            while(rs.next()){
            invoice_fetch NewInvoice = new invoice_fetch();
            NewInvoice.setCust_number(rs.getString("cust_number"));
            NewInvoice.setDue_in_date(rs.getDate("due_in_date"));
            NewInvoice.setInvoice_id(rs.getLong("invoice_id"));
            NewInvoice.setName_customer(rs.getString("name_customer"));
            NewInvoice.setTotal_open_amount(rs.getDouble("total_open_amount"));
            System.out.println();

            if(rs.getString("notes")==null)
            NewInvoice.setNotes("Lorem Ipsum dolor...");
            else
            NewInvoice.setNotes(rs.getString("notes"));
            dataList.add(NewInvoice);
            }
                stmt.close();
                conn.close();
        }catch(SQLException se){
            se.printStackTrace();
        }catch(Exception e){
            e.printStackTrace();
        }finally{
            try{
            if(stmt!=null)
            stmt.close();
            }catch(SQLException se2){
            }
            try{
            if(conn!=null)
            conn.close();
        }catch(SQLException se){
            se.printStackTrace();
            }
        }
        return dataList;

    }


    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

        doGet(request, response);
    }

}
```

3.3.2.2.6 Invoice Fetch Servlet

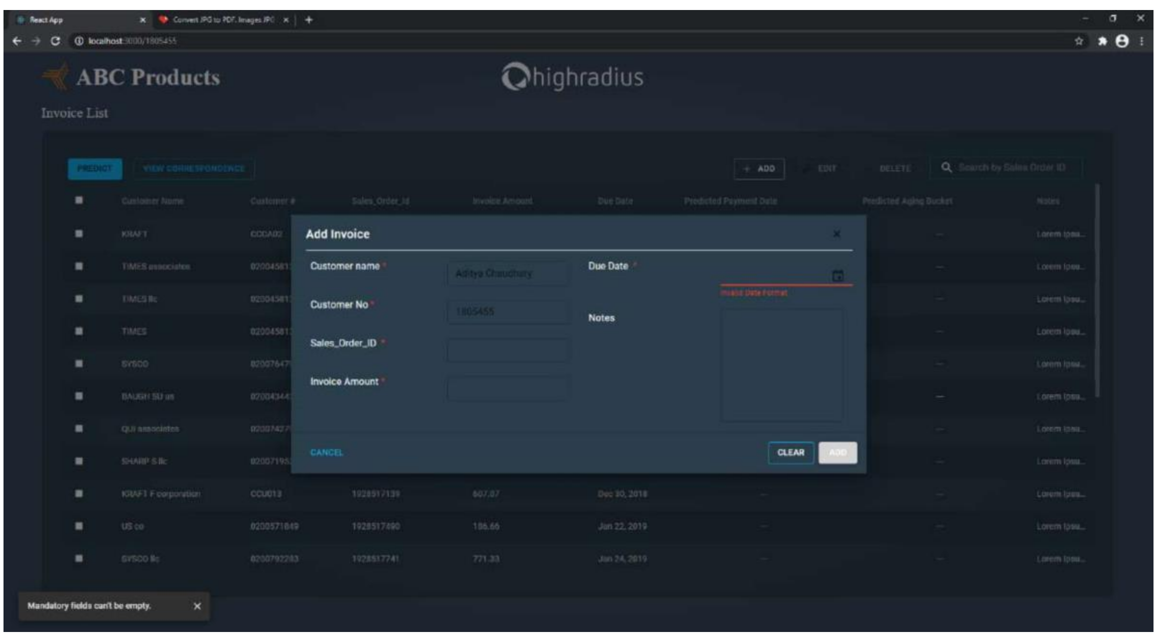
```java
package Servlets;
import java.util.*;

public class invoice_fetch {
    String cust_number=null;
    String name_customer=null;
    Date due_in_date=null;
    double total_open_amount;
    Long invoice_id=null;
    String notes=null;



    public invoice_fetch() {
        super();
        // TODO Auto-generated constructor stub
    }


    public invoice_fetch(String cust_number, String name_customer, Date due_in_date, double total_open_amount,
        Long invoice_id, String notes) {
        super();
        this.cust_number = cust_number;
        this.name_customer = name_customer;
        this.due_in_date = due_in_date;
        this.total_open_amount = total_open_amount;
        this.invoice_id = invoice_id;
        this.notes = notes;
    }


    public String getCust_number() { return cust_number; }
    public void setCust_number(String cust_number) { this.cust_number = cust_number; }
    public String getName_customer() { return name_customer; }
    public void setName_customer(String name_customer) { this.name_customer = name_customer; }
    public Date getDue_in_date() { return due_in_date; }
    public void setDue_in_date(Date due_in_date) { this.due_in_date = due_in_date; }
    public double getTotal_open_amount() { return total_open_amount; }
    public void setTotal_open_amount(double total_open_amount) { this.total_open_amount = total_open_amount; }
    public Long getInvoice_id() { return invoice_id; }
    public void setInvoice_id(Long invoice_id) { this.invoice_id = invoice_id; }
    public String getNotes() { return notes; }
    public void setNotes(String notes) { this.notes = notes; }
}
```

3.3.3 Sales Order Management User Interface

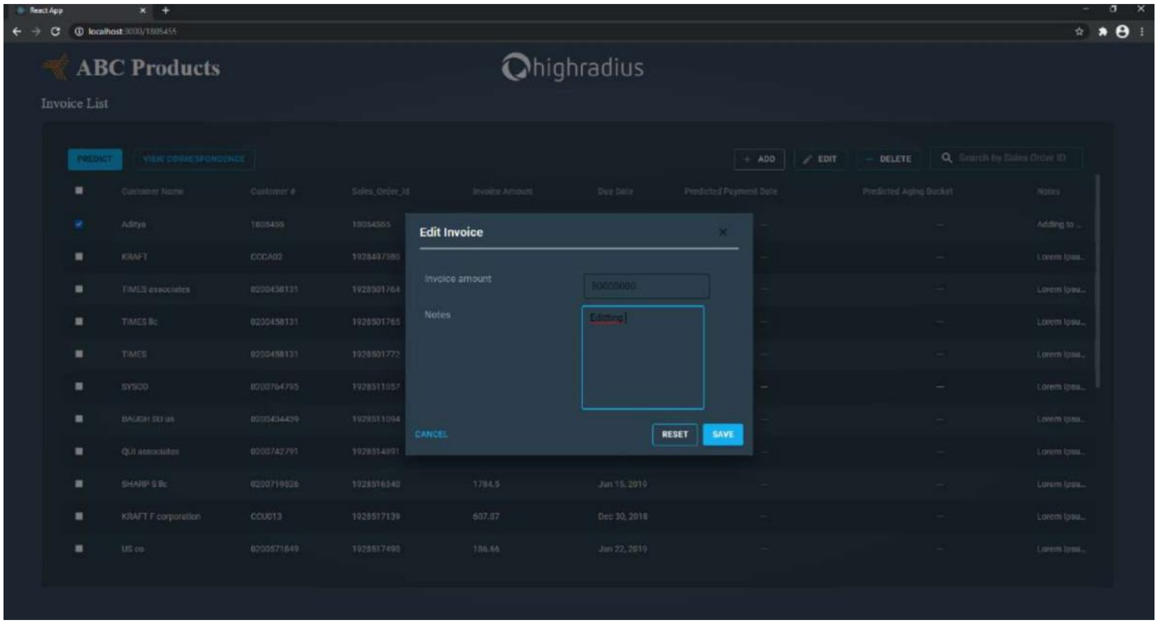
3.3.3.1 Landing Page Dashboard



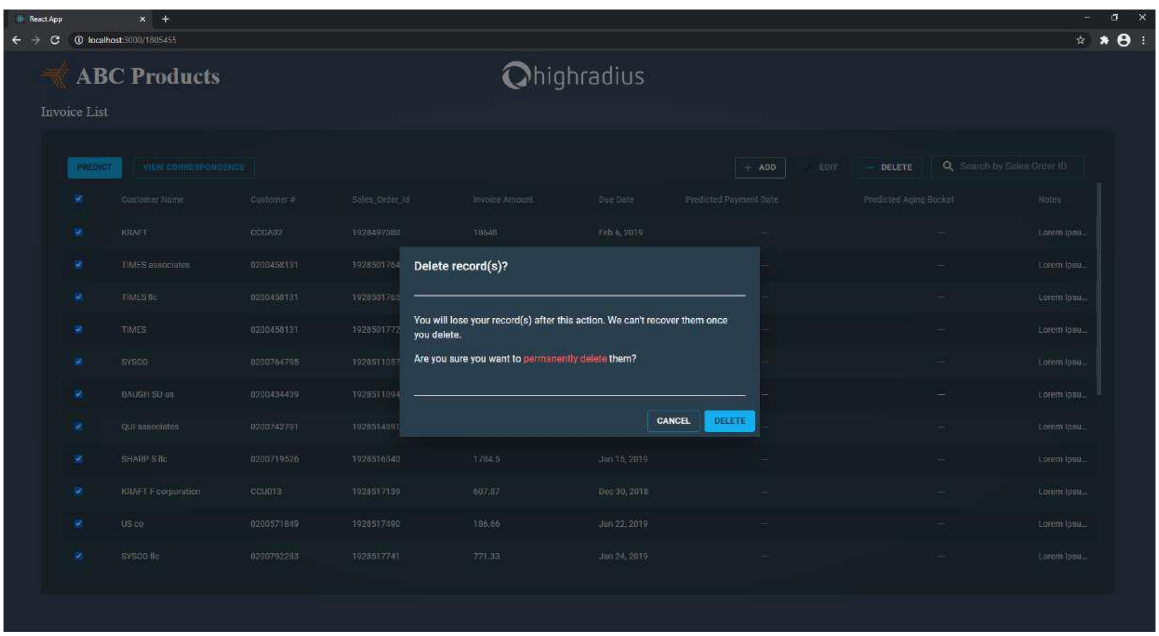
3.3.3.2 Add Invoice Functionality



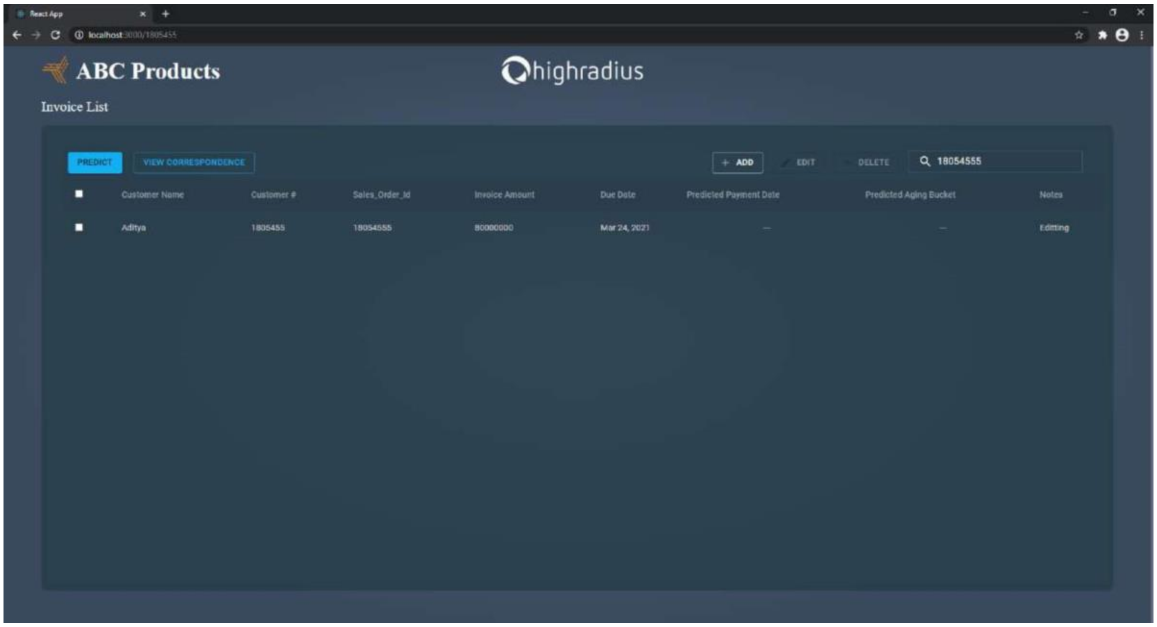
3.3.3.3 Edit Invoice Functionality



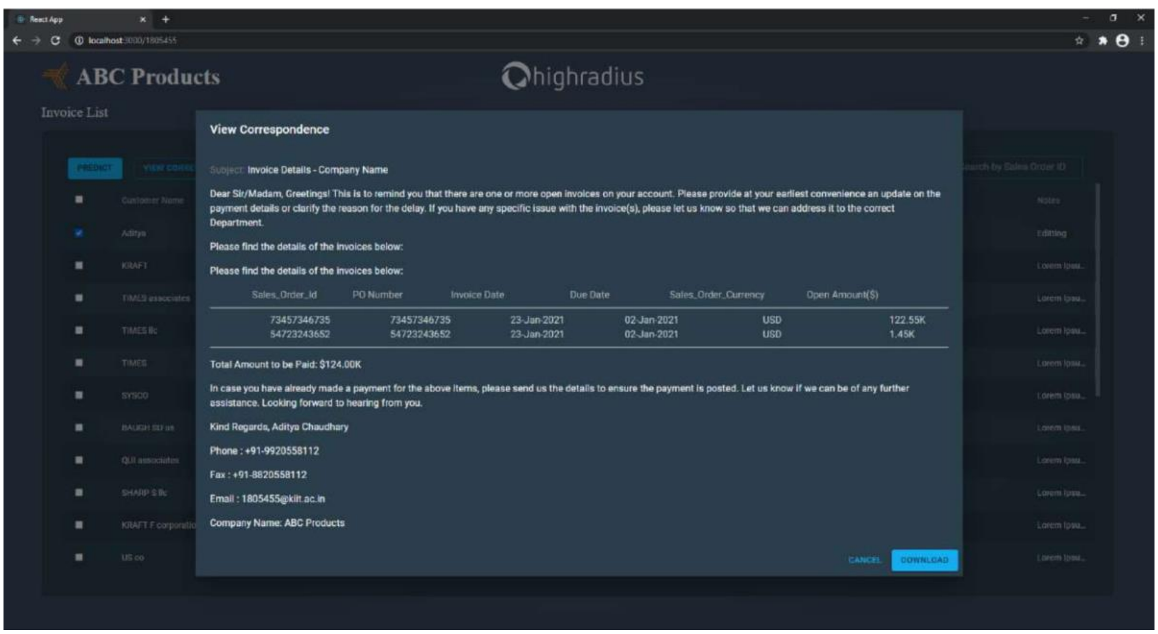
3.3.3.4 Delete Invoice Functionality



3.3.3.5 Invoice Search Functionality



3.3.3.6 View Correspondence Functionality



3.4 Research Methodology

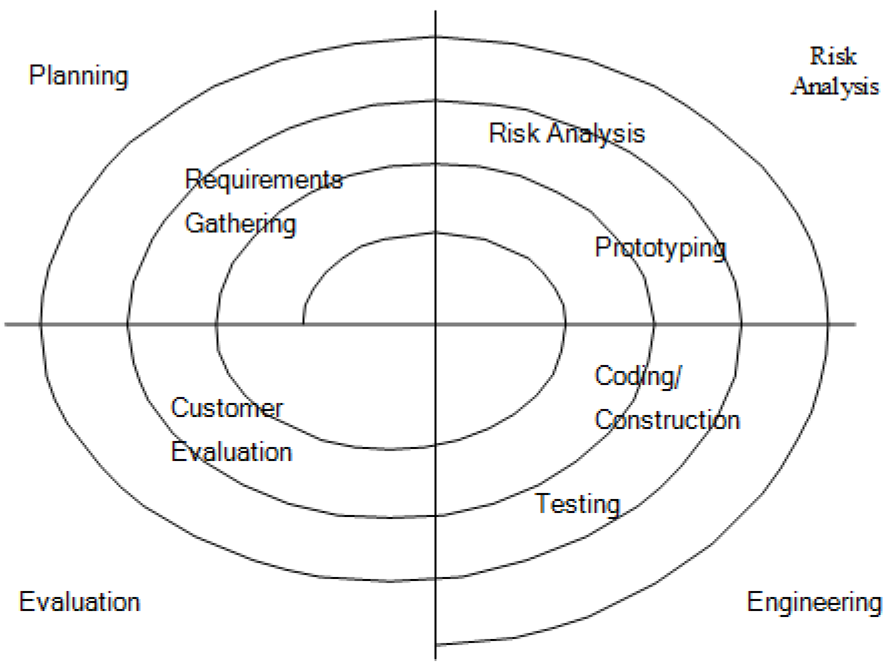
3.4.1 Research Design

I will use the test type of the research design. It is a method of much research. Basically, it is a scientific study, in which a collection of variables are kept unchanged while other variables are measured as the subject matter. This is especially true when it comes to face recognition and acquisition as it takes into account the behavior and patterns of the topic that should be used to determine whether the topic complies with all the information presented and evaluated with previous information. It is a method of researching results as it is timely and focuses on the interactions between variables that give real results.

3.4.2 System Development Methodology.

A software development method is a way to manage project development. There are many types of options available such as Waterfall model, Growing Model, RAD model, Agile model, design model and Spiral model. However, the engineer still needs to be consulted to determine what will be used in the project. The method model helps to manage the project efficiently and is able to assist the engineer in identifying any problem during development. Also, it helps to achieve the purpose and scope of projects. To build a project, it needs to understand the needs of the participants.

The methodology provides the framework for making the proposed DM model. This is a process that consists of steps that convert raw data into decent data patterns to extract information from users.



There are four phases that involve in the spiral model:

1) Planning phase

The stage where collection and risk are required is assessed. This is the stage where the project title is discussed with the project manager. From that discussion, the Heart Guidance Program has been proposed. The need and risk were assessed after research into the existing system and a literature review was conducted with other available research.

2) Risk analysis Phase

The stage where the risk is found and the other solution. An example is made at the end of this section. If there is a risk during this phase, there will be suggestions for another solution.

3) Engineering phase

In this phase, software is developed and tests are performed at the end of this phase.

4) Evaluation phase

At this stage, the user checks the software. This will be done after the system has been launched and the user will check if the system meets their expectations or needs or not. If there is an error, the user may report a problem with the system.

3.5 Data Collection and Preprocessing.

The data set for this study were provided from the HRC database. The data had various anomalies like negative payments, which were cleaned up in the data preprocessing phase, few columns like the Currency column which had more than one type of currency were label encoded, similarly the business code column was hot encoded, and the customer payment terms and customer number columns were label encoded. Preprocessing and clearing of invalid date time was done in the data preprocessing phase.

3.6 Tools

Operating System	Windows 10 or any Linux Debian Distro.
Programming Languages	Python, JavaScript, JAVA, SQL.
Tools	Jupyter Notebook, Python , Microsoft Excel, VSCode, Eclipse IDE, SQLYog.
Technologies utilized	Machine Learning, Web Development.

3.7 Software Requirements

Operating System	Any OS with client to use the internet
Network	Wi-Fi Internet or Cellular Network
Visual Studio Code	To run NPM commands
Eclipse IDE	To run the Java Servlets
SQLYog	To perform database actions
Internet Browser	To run Jupyter Notebook and UI

3.8 Hardware Requirements

For application development, the following Software Requirements are:

Processor: Intel i5-3570k or high equivalent

RAM: 8192 MB

Space on disk: minimum 6000MB

For running the application:

Device: Any device that can access the internet

Minimum space to execute: 1000MB

The effectiveness of the proposal is evaluated by conducting experiments on a system configured with an AMD Ryzen™ 5600x processor (4.37GHZ, 6 Cores, 32 GB RAM, running Windows 10 Home version 2004)

CHAPTER 4

RESULTS & DISCUSSIONS

The proposed system is scalable, reliable and an expandable system. The proposed model gives an RMSE of 10.9 with the Random Forest Regression algorithm and RMSE of 9.5 with Multiple Linear Regression algorithm, but the Random Forest Algorithm performs better as the latter is greatly overfitting our training set, hence we proceed with the Random Forest Regression algorithm for predicting our unknown dates and get a good response.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion & Future Work

The proposed application is easy to use, scalable, reliable and an expandable system. The proposed application can help in increasing the efficiency of Accounts Receivable Teams buy predicting the customer payment dates and bucketing them, and with this data the AR Team can take required steps to recover their credit from the customers thus increasing efficiency of the team and creating better cash flow for the company. This application also comes with integrated UI which allows, even people with no programming experience to use this software thus increasing the user base exponentially and increasing the applications accessibility. The backend is managed by Java and database is stored in a relational database which are industry standards for large scale applications and are known to be scalable and efficient.

REFERENCES

1. https://www.highradius.com/
2. https://docs.python.org/3/
3. https://www.w3schools.com/python/
4. https://www.analyticsvidhya.com/
5. https://www.kaggle.com/
6. https://docs.oracle.com/en/java/
7. https://www.w3schools.com/java/
8. https://www.javatpoint.com/servlet-tutorial
9. https://developer.mozilla.org/en-US/docs/Web/JavaScript
10. https://reactjs.org/docs/hello-world.html
11. https://tinyurl.com/YTLinkCodeEvolution
12. https://redux.js.org/tutorials/essentials/part-1-overview-concepts

INDIVIDUAL CONTRIBUTION REPORT

SALES ORDER MANAGEMENT APPLICATION

ADITYA CHAUDHARY

1805455

Abstract: This Sales Order Management Application is implemented using Artificial Intelligence to predict the payment dates, the backend and database of the application are managed by Java JDBC, Java Servlets and SQLYog respectively. The application has an User Interface which is programmed using HTML, CSS, JavaScript, React and Redux.

Individual contribution and findings: Planned the approach to finishing the project on time, coded and developed everything from scratch for the project, learnt about various new terminologies and technologies while working on this project, various websites and reading materials were referenced for the completion of the project.

Individual contribution to project report preparation: I contributed to each and every section of the project report preparation.

Individual contribution for project presentation and demonstration: I contributed to each and every section of the presentation part.

Aditya Chaudhary

Full Signature of Supervisor: Full signature of the Student:

Sudakshina Chaudhury Aditya Chaudhary

TURNITIN PLAGIARISM REPORT

(This report is mandatory for all the projects and plagiarism must be below 30%)

Sample_turnitin_report_for_students.docx

ORIGINALITY REPORT

13%	7%	3%	7%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Kennesaw State University Student Paper	3%
2	www.guardian.co.uk Internet Source	2%
3	Campbell, Neil. "Post-Western Cinema", A Companion to the Literature and Culture of the American West Witschi/A Companion to the Literature and Culture of the American West, 2011. Publication	1%