ACCOUNT PAYABLE FLOW

PROJECT REPORT

Group 49

Rohith Adhitya Chinnannan Rajkumar

Uma Maheshwari Deivasigamani

857-701-8735 (Tel of Student 1)

857-930-9244 (Tel of Student 2)

chinnannanrajkumar.r@northeastern.edu

deivasigamani.u@northeastern.edu

Percentage of Effort Contributed by Student1: 50

Percentage of Effort Contributed by Student2: 50

Signature of Student 1:

Signature of Student 2:

Submission Date: 12/10/2023

Executive Summary

The principal purpose of this project is to automate the Accounts Payable flow to enable seamless, accurate and precise release of funds for the organisation. Every organisation deals with invoices and some of them receive a huge number of transactional receipts and invoices and it becomes very difficult and a tedious process to reconcile and cross verify every invoice to release the funds. Hence, we developed a solution to automate the process of matching the invoice line items with the purchase order line items along with the good received notice line items making it a three way match. Our project here will reduce the time taken to release the funds by more than 80%. What took days and weeks to do by hundreds of people can be trimmed down to a few seconds with our proposed solution. We would also be able to derive insightful analytics by visualising the sales and transactional data.

The schema of the database was tailored to accept the required attributes that are required to perform the three way matching. The EER and UML diagrams were modelled, followed by the mapping of conceptual model to relational model with the necessary primary and foreign keys. We implemented SQL queries to accommodate different real-world possibilities of the Accounts Payable workflow. Important parameters to consider while matching are quantity, unit price and product description and these parameters are unstructured and not an enumerated set which makes it even more challenging to solve. Our query modelling has to be fault tolerant and robust to handle different sets of circumstances. MongoDB was used to run aggregation queries to derive insights on sales and transactional data because of its ability to offer high throughput and low latency queries by deploying an aggregation pipeline to achieve this.

The project has tremendous potential to grow with more real-world data across multiple verticals of business. And by bootstrapping it with Python and data pipelines this project could possibly become a product to be used by any large organisations to accelerate the entire workflow of payments. We can also detect fraud or avoid overpaying beyond the intended amount which could save the company a fortune! A little room of improvement on this product is to extend it to multiple languages and to cater to a global audience, not limited to any specific geography.

I. Introduction

In today's complex and interconnected business environment, managing Accounts Payable (AP) and supplier-company relationships is critical. While typical AP activities include the processing of invoices, payments, and financial transactions, organizations frequently face a slew of issues and circumstances that require quick problem-solving and tactical choices. The most frequent problems encountered in the AP flow system will be Incorrect invoices, mismatched Purchase Order (PO), late payments and losing track of the received goods from the company placing the orders in bulk.

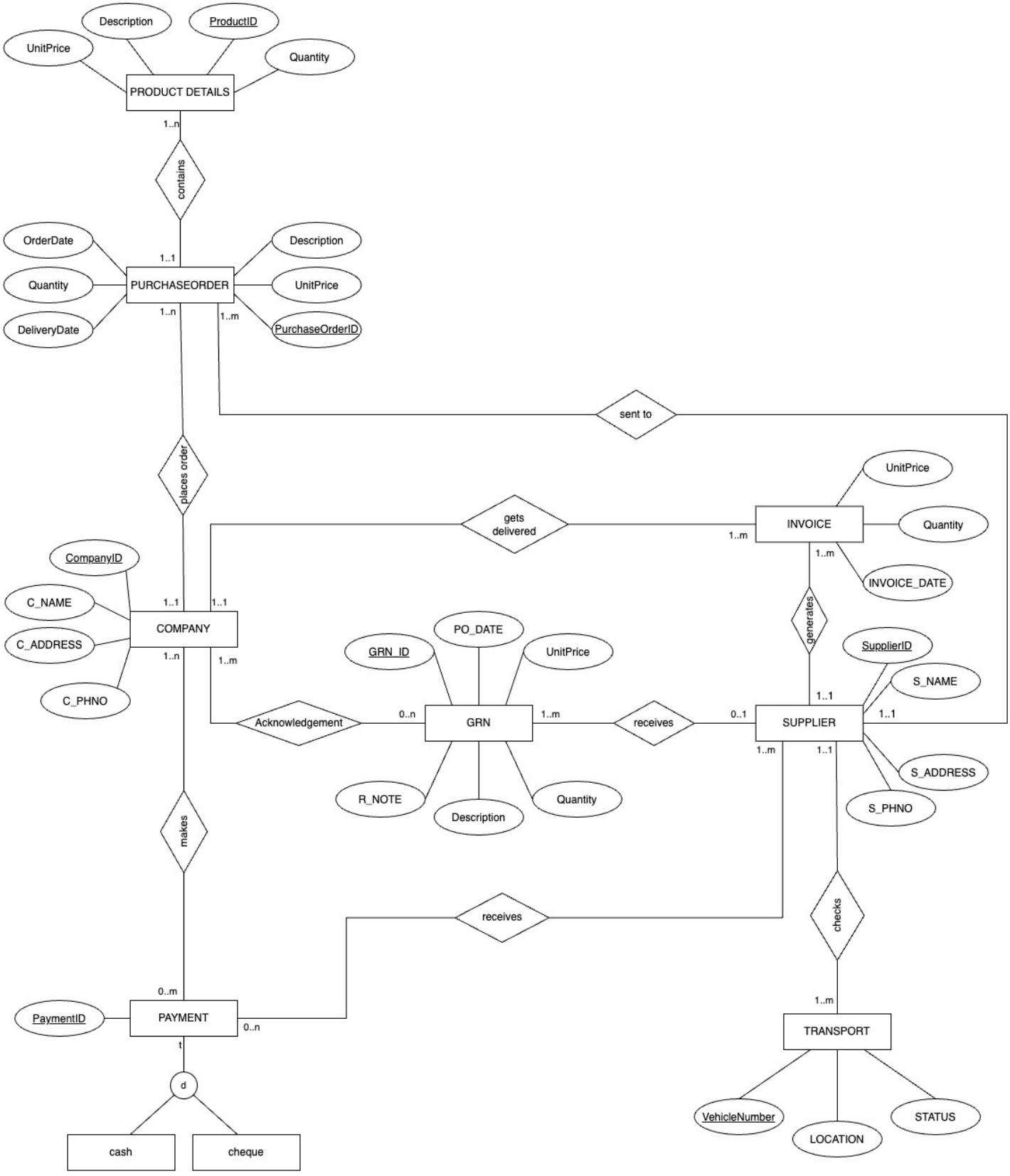
Account Payable Flow is a process that ensures the managing and tracking of financial transactions a company owes to its suppliers. Let us take an example where the manager of a company places an order with a supplier. The Purchase order (PO) is being generated with the Purchase order ID, date, supplier ID, company ID, specification of the product, Quantity and required Delivery date. When the supplier delivers the goods to the company, the manager then sends a Goods Received Notice (GRN) to the supplier. The GRN verifies that the goods received are in good condition and the specification is matched with the PO.

Now the supplier sends an invoice that has the Invoice number, Date, company ID, supplier ID, Price and quantity of the product. Now the company processes the payment for the goods delivered. Here the goods can be sent in batches or all at one time. For example, if there is an PO for 100 chairs, the order can be sent in a batch of 25 chairs in 4 sets or in a batch of 50 chairs in 2 sets or all at one time as per the supplier’s delivery convenience. Here each time the GRN has to be sent to the supplier by the company to notify that the goods were received by them. The most important aspect here is that the requirements of the company and the goods received should match with each other.

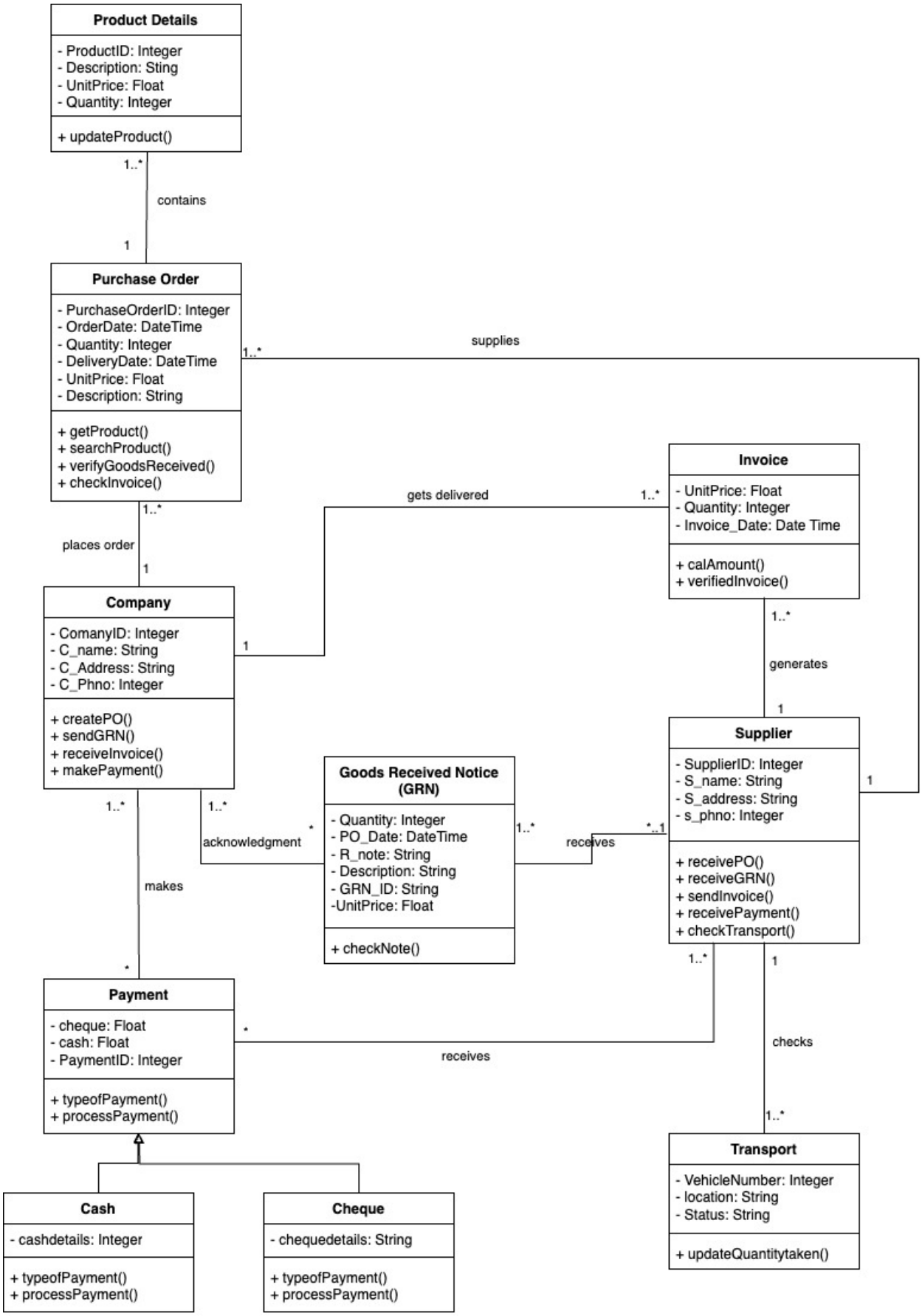
The problem arises when there is a mismatch in the quantity of goods received and the requirements of the company in the PO. For this, the supplier needs to check whether the product matches with the required product in PO sent by the company. Another problem arises when a wrong invoice has been sent to the company. The product would be mismatched, or the amount would be wrong, this can lead to bad relationships between the company and the supplier. The supplier must double check the invoice before sending it to the company. The third problem is when products are being sent in batches, the GRN gets generated accordingly by the company, and if any one of the GRN is missing that would lead to a confusion in the supply chain, Hence the company and supplier should double check the received order quantity and the GRNs in order to maintain a good flow of business.

II. Conceptual Data Modelling

1. EER Diagram



2. UML Diagram



III. Mapping Conceptual Model to Relational Model

Primary Key – Underlined **Foreign Key -- # hash tagged**

Company(CompanyID, C_NAME, C_ADDRESS, C_PHNO)
Supplier(SupplierID, S_NAME, S_ADDRESS, S_PHNO)
ProductDetails(ProductID, DESCRIPT, UNIT_PRICE, STOCK)
Payment(PaymentID)
Transport(VehicleNumber, #SupplierID, LOCATION, STATUS)

- Foreign key: SupplierID references from Supplier

PurchaseOrder(PurchaseOrderID, #SupplierID, #CompanyID, #ProductID, #PaymentID, ORDER_DATE, QTY, DELIVERY_DATE, DESCRIPTION, UNIT_PRICE)

- Foreign key: SupplierID references from Supplier
- Foreign key: CompanyID references from Company
- Foreign key: PaymentID references from Payment
- Foreign key: ProductID references from ProductDetails

GRN(GRN_ID, #CompanyID, #ProductID, #PurchaseOrderID, R_NOTE, QTY, PO_DATE, C_NAME, DESCRIPTION, UNIT_PRICE)

- Foreign key: CompanyID references from Company
- Foreign key: ProductID references from ProductDetails
- Foreign key: PurchaseOrderID references from PurchaseOrder

Acknowledgement(#CompanyID, #GRN_ID)

- Foreign key: CompanyID references from Company
- Foreign key: GRN_ID references from GRN

Cash(#PaymentID, cash_details)

- Foreign key: PaymentID references from Payment

Cheque(#PaymentID, cheque_details)

- Foreign key: PaymentID references from Payment

MAKESCID(#CompanyID, #PaymentID)

- Foreign key: CompanyID references from Company
- Foreign key: PaymentID references from Payment

Invoice(InvoiceID, #SupplierID, #CompanyID, #PurchaseOrderID, PRICE, QTY, INVOICE_DATE, DESCRIPTION)

- Foreign key: SupplierID references from Supplier
- Foreign key: CompanyID references from Company
- Foreign key: PurchaseOrderID references from PurchaseOrder

IV. Implementation of Relational Model via MySQL and NoSQL

MySQL Implementation:

The database was created in MySQL and the following queries were performed:

Query 1: Find Total price of each order that has been placed.

```
SELECT (QTY*UNIT_PRICE) AS Total
FROM PurchaseOrder;
```

Total
5000.00
5999.70
1750.00
2699.70
1000.00

Query 2: Find maximum purchase order by each company.

```
SELECT CompanyID, MAX(UNIT_PRICE * QTY) AS
MaxPurchaseAmount
FROM PurchaseOrder GROUP BY CompanyID;
```

CompanyID	MaxPurchaseAmount
18943	62993.00
20756	5000.00
21856	69990.00
31589	3749.25
32814	12999.00

Query 3: Find out the mismatched quantities between PurchaseOrder and Invoice.

```
SELECT
   po.PurchaseOrderID,
   po.SupplierID,
   po.CompanyID,
   po.QTY AS PurchaseOrderQty,
   i.Qty AS InvoiceQty
FROM PurchaseOrder po
LEFT JOIN Invoice i ON po.PurchaseOrderID = i.PurchaseOrderID
WHERE i.PurchaseOrderID IS NULL OR po.QTY <> i.Qty;
```

PurchaseOrderID	SupplierID	CompanyID	PurchaseOrderQ...	InvoiceQty
10211	218	37294	1000	100
10212	934	46709	1000	100
10224	506	21856	1000	100

Query 4: Find out companies with single PurchaseOrder, single Invoice and multiple GRNs.

```
SELECT * FROM PurchaseOrder po WHERE po.PurchaseOrderID IN (
   SELECT po.PurchaseOrderID FROM PurchaseOrder po LEFT JOIN GRN g ON
po.PurchaseOrderID = g.PurchaseOrderID LEFT JOIN Invoice i ON po.PurchaseOrderID =
i.PurchaseOrderID GROUP BY po.PurchaseOrderID HAVING COUNT(DISTINCT
g.GRN_ID) > 1 AND COUNT(DISTINCT i.InvoiceID) = 1);
```

PurchaseOrderID	SupplierID	CompanyID	ORDER_DATE	QTY	DELIVERY_DATE	PaymentID	DESCRIPTION	UNIT_PRICE	ProductID
10201	492	37294	2008-01-23	100	2020-01-23	22	Luminous Bliss Perfume	50.00	S1
10203	891	41378	2022-03-23	50	2005-04-23	24	ZenWell Meditation Pillow	35.00	S3
10204	650	89134	2005-04-23	30	2018-04-23	25	SwiftGlide Hair Straightener	89.99	S4
10205	413	21856	2018-05-23	50	2001-06-23	26	EcoHarmony Reusable Water Bottle	20.00	S5
10206	795	76503	2027-06-23	200	2010-07-23	27	AdventureSeeker Backpack	69.99	S6
10210	765	65092	2011-10-23	45	2024-10-23	31	FitnessFusion Resistance Bands	19.99	S10

Query 5: Find out mismatched products between PurchaseOrder and Invoice.

```
SELECT i.SupplierID,  i.CompanyID, i.DESCRIPTION AS Invoice_Product_Description,
po.DESCRIPTION AS PurchaseOrder_Product_Description
FROM Invoice i
JOIN PurchaseOrder po ON
i.PurchaseOrderID =
po.PurchaseOrderID
```

SupplierID	CompanyID	Invoice_Product_Descript...	PurchaseOrder_Product_Descrip...
650	76503	Nirma Washing Power	GourmetBlend Coffee Beans
372	74025	TechMaster Pro mouse	TechMaster Pro Headphones

```
WHERE NOT EXISTS (
    SELECT 1 FROM PurchaseOrder po2 WHERE i.PurchaseOrderID =
po2.PurchaseOrderID AND i.DESCRIPTION = po2.DESCRIPTION);
```

Query 6: Find out mismatched unit prices of products between PurchaseOrder and Invoice.

```
SELECT i.SupplierID, i.CompanyID, i.PRICE AS Invoice_Unit_Price, po.UNIT_PRICE
AS PurchaseOrder_Unit_Price
FROM Invoice i
JOIN PurchaseOrder po ON
i.PurchaseOrderID = po.PurchaseOrderID
WHERE i.PRICE <> ALL (
    SELECT Unit_Price FROM PurchaseOrder WHERE i.PurchaseOrderID =
PurchaseOrder.PurchaseOrderID);
```

SupplierID	CompanyID	Invoice_Unit_Pri...	PurchaseOrder_Unit_Pri...
795	32814	149.99	129.99
740	58413	25.00	20.00

Query 7: List out the payments made through 'cash' and 'cheque'.

```
SELECT PaymentID, cash_details AS payment_details FROM Cash
UNION
SELECT PaymentID, cheque_details AS payment_details FROM
Cheque;
```

PaymentID	payment_details
22	1000
23	234
24	111
25	567
26	899

Query 8: Find products which are having unit price greater than average unit price.

```
SELECT description ,unit_price FROM purchaseorder
WHERE Unit_Price > (
   SELECT AVG(unit_price)
   FROM purchaseorder
);
```

description	unit_price
TechMaster Pro Headphones	199.99
SwiftGlide Hair Straightener	89.99
HarmonyHome Smart Thermostat	129.99
TechMaster Pro Headphones	199.99
TechMaster Pro Headphones	199.99

Query 9: Happy Path query

```
SELECT po.PurchaseOrderID, po.SupplierID, po.CompanyID, po.ProductID,
po.DESCRIPTION AS PurchaseOrder_Description, i.PRICE AS Invoice_Unit_Price,
g.GRN_ID, g.QTY AS GRN_QTY
FROM PurchaseOrder po
JOIN Invoice i ON po.PurchaseOrderID = i.PurchaseOrderID
JOIN GRN g ON po.PurchaseOrderID = g.PurchaseOrderID AND po.ProductID =
g.ProductID WHERE po.QTY = i.QTY AND po.QTY = g.QTY AND po.DESCRIPTION =
g.DESCRIPTION AND po.DESCRIPTION=i.DESCRIPTION AND i.PRICE =
po.Unit_Price AND g.Unit_Price = po.Unit_Price AND i.PRICE IS NOT NULL AND
g.Unit_Price IS NOT NULL;
```

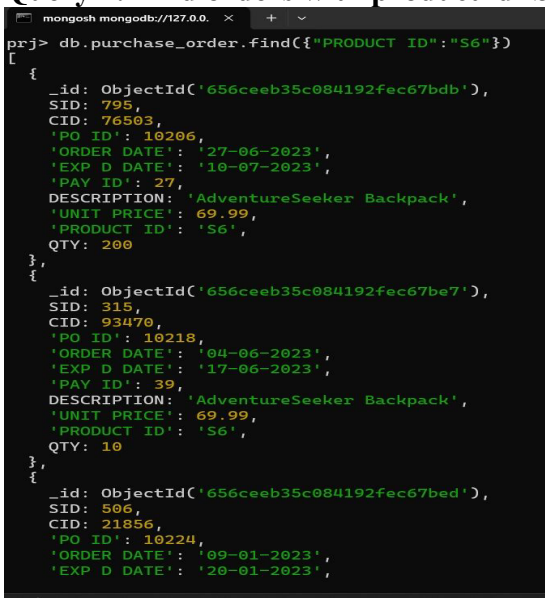
PurchaseOrderID	SupplierID	CompanyID	ProductID	PurchaseOrder_Description	Invoice_Unit_Pri...	GRN_ID	GRN_QTY
10215	506	92647	S3	ZenWell Meditation Pillow	35.00	G15	20
10221	891	18943	S3	ZenWell Meditation Pillow	35.00	G21	600
10223	795	37294	S5	EcoHarmony Reusable Water Bottle	20.00	G23	990
10231	605	46709	S13	Serenity Spa Candle Collection	15.00	G31	25

NoSQL Implementation:

Query 1: Find total price of all products with product id 'S1'.

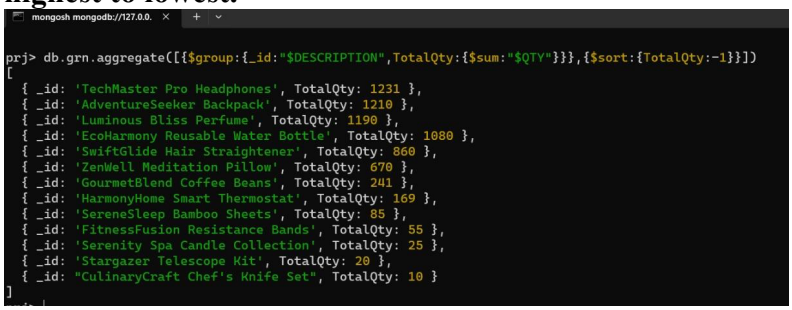
```
prj> db.purchase_order.aggregate([{$match:{"PRODUCT ID":"S1"}},{$group
:{_id:null,TotalPrice:{$sum:{$multiply:["$QTY","$UNIT PRICE"]}}}}]).to
Array()
[ { _id: null, TotalPrice: 60000 } ]
```

Query 2: Find orders with product id ‘S6’



```
mongosh mongodb://127.0.0.
prj> db.purchase_order.find({"PRODUCT ID":"S6"})
[
  {
    _id: ObjectId('656ceeb35c084192fec67bdb'),
    SID: 795,
    CID: 76503,
    'PO ID': 10206,
    'ORDER DATE': '27-06-2023',
    'EXP D DATE': '10-07-2023',
    'PAY ID': 27,
    DESCRIPTION: 'AdventureSeeker Backpack',
    'UNIT PRICE': 69.99,
    'PRODUCT ID': 'S6',
    QTY: 200
  },
  {
    _id: ObjectId('656ceeb35c084192fec67be7'),
    SID: 315,
    CID: 93470,
    'PO ID': 10218,
    'ORDER DATE': '04-06-2023',
    'EXP D DATE': '17-06-2023',
    'PAY ID': 39,
    DESCRIPTION: 'AdventureSeeker Backpack',
    'UNIT PRICE': 69.99,
    'PRODUCT ID': 'S6',
    QTY: 10
  },
  {
    _id: ObjectId('656ceeb35c084192fec67bed'),
    SID: 506,
    CID: 21856,
    'PO ID': 10224,
    'ORDER DATE': '09-01-2023',
    'EXP D DATE': '20-01-2023',
```

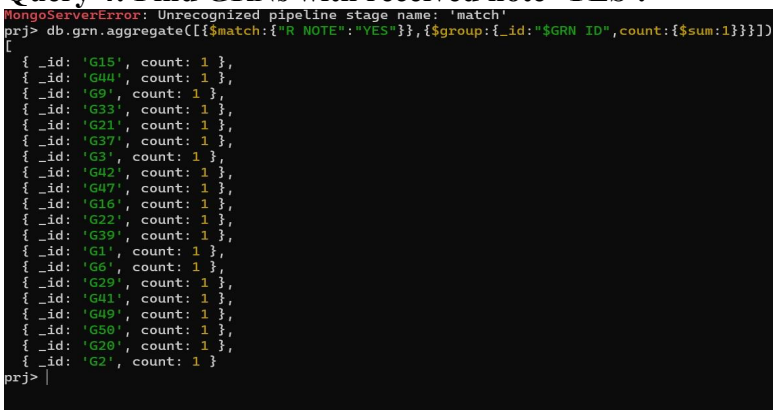
Query 3: Find total number of each products sold in the year 2023 and list them from highest to lowest.



```
mongosh mongodb://127.0.0.

prj> db.grn.aggregate([{$group:{_id:"$DESCRIPTION",TotalQty:{$sum:"$QTY"}}},{$sort:{TotalQty:-1}}])
[
  { _id: 'TechMaster Pro Headphones', TotalQty: 1231 },
  { _id: 'AdventureSeeker Backpack', TotalQty: 1210 },
  { _id: 'Luminous Bliss Perfume', TotalQty: 1190 },
  { _id: 'EcoHarmony Reusable Water Bottle', TotalQty: 1080 },
  { _id: 'SwiftGlide Hair Straightener', TotalQty: 860 },
  { _id: 'ZenWell Meditation Pillow', TotalQty: 670 },
  { _id: 'GourmetBlend Coffee Beans', TotalQty: 241 },
  { _id: 'HarmonyHome Smart Thermostat', TotalQty: 169 },
  { _id: 'SereneSleep Bamboo Sheets', TotalQty: 85 },
  { _id: 'FitnessFusion Resistance Bands', TotalQty: 55 },
  { _id: 'Serenity Spa Candle Collection', TotalQty: 25 },
  { _id: 'Stargazer Telescope Kit', TotalQty: 20 },
  { _id: "CulinaryCraft Chef's Knife Set", TotalQty: 10 }
]
```

Query 4: Find GRNs with received note ‘YES’.

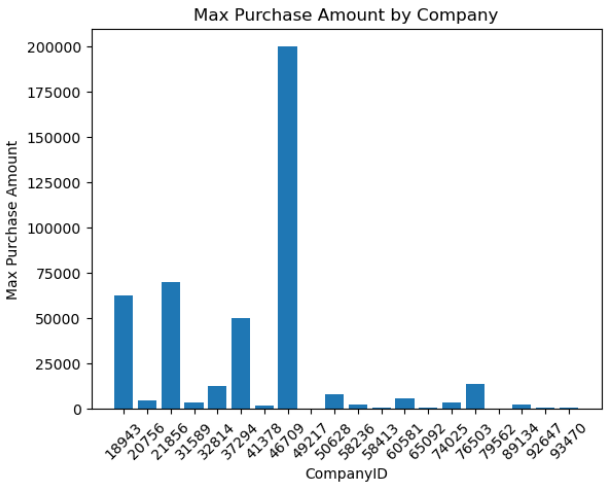


```
MongoServerError: Unrecognized pipeline stage name: 'match'
prj> db.grn.aggregate([{$match:{"R NOTE":"YES"}},{$group:{_id:"$GRN ID",count:{$sum:1}}}])
[
  { _id: 'G15', count: 1 },
  { _id: 'G44', count: 1 },
  { _id: 'G9', count: 1 },
  { _id: 'G33', count: 1 },
  { _id: 'G21', count: 1 },
  { _id: 'G37', count: 1 },
  { _id: 'G3', count: 1 },
  { _id: 'G42', count: 1 },
  { _id: 'G47', count: 1 },
  { _id: 'G16', count: 1 },
  { _id: 'G22', count: 1 },
  { _id: 'G39', count: 1 },
  { _id: 'G1', count: 1 },
  { _id: 'G6', count: 1 },
  { _id: 'G29', count: 1 },
  { _id: 'G41', count: 1 },
  { _id: 'G49', count: 1 },
  { _id: 'G50', count: 1 },
  { _id: 'G20', count: 1 },
  { _id: 'G2', count: 1 }
prj>
```

V. Database access via Python

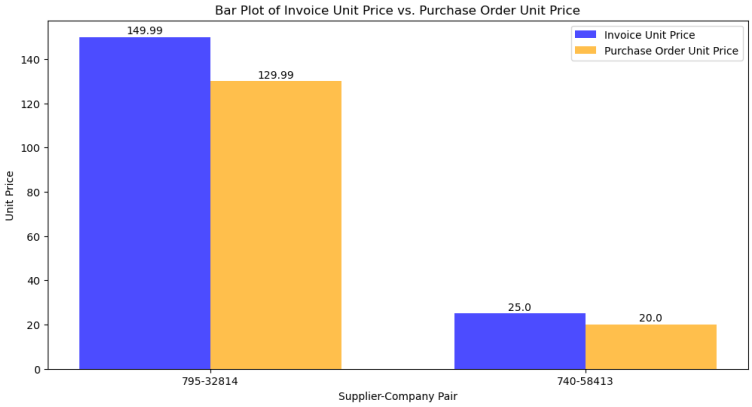
The database is accessed using Python to analyze the data through visualization for better understanding. The connection of MySQL to Python is done using mysql.connector, followed by cursor.excecute to run and fetchall from query. The list of data are converted into a dataframe using pandas library and matplotlib is used for the visualization of data using graphs.

Graph 1: Maximum purchase amount by each company



Graph 2: Total prices from PurchaseOrder

Graph 3: Mismatched Unit prices



VI. Summary and Recommendation

The Account Payable flow Project has a highly efficient and automated workflow. The three-way matching system, coupled with robust SQL queries and MongoDB analytics, promises to significantly reduce the time and resources traditionally spent on reconciling invoices, purchase orders, and goods received notices. The model has been tuned accordingly to solve the real world problems of invoice, quantity and price mismatch, also the model was able to fetch all the necessary details from the database. The shortcoming would be in the NoSQL implementation of this database on MongoDB. More study should be done on how a unique relational database like this can be implemented in a NoSQL environment. This model can be made compatible with different industry specific needs which will provide invaluable insights into the system's performance and usability in real-world scenarios where large MNCs can incorporate this model to work more efficiently in their logistics part of business. The project holds immense growth potential, not just as a solution to streamline Accounts Payable but as a comprehensive product.