Back Bay Jewels

PROJECT REPORT

Group 7
Priyanka Senthil
Samuel Dragomire

857-361-6942 (Priyanka)
401-332-9398 (Samuel)

senthilkumar.pri@northeastern.edu
dragomire.s@northeastern.edu

Percentage of Effort Contributed by Priyanka: 50%

Percentage of Effort Contributed by Samuel: 50%

Signature, Priyanka: [illegible]

Signature, Samuel: [illegible]

Submission Date: 21APR2024

Executive Summary:

The goal of this project was to create a relational database that tracked the jewelry making process in our enterprise, Back Bay Jewels, from the sourcing to the customer interactions. Since the company frequently manages high value precious metals and gems, it is critical to keep a prudent record of all steps, not only to ensure quality within our own factories, but to monitor the reputation of external suppliers. Furthermore, the database created exhibits the interactions between each of the entities within our business and records the exchanges of these entities all within one, centralized framework.

The database was conceptually laid out in an EER Model and a UML Class diagram to visually represent the entities involved in the business and better understand from where each table of data came. It is important to note that the Back Bay Jewels factory was not represented alone, but instead as an aggregation of the employees together with the warehouse vaults.

In MySQL, a relational database was created to be used for managing transactions, recording supplier information, tracking inventory, and storing customer data. It ensures that each step of the jewelry-making process is documented and traceable, reducing errors and improving efficiency.

Additionally, a NoSQL database was implemented using MongoDB to test if the database would perform better in this mode. However, due to the success of the SQL version the company will likely progress with business operation and decision making through the usage of a SQL backed database management system.

I. Introduction

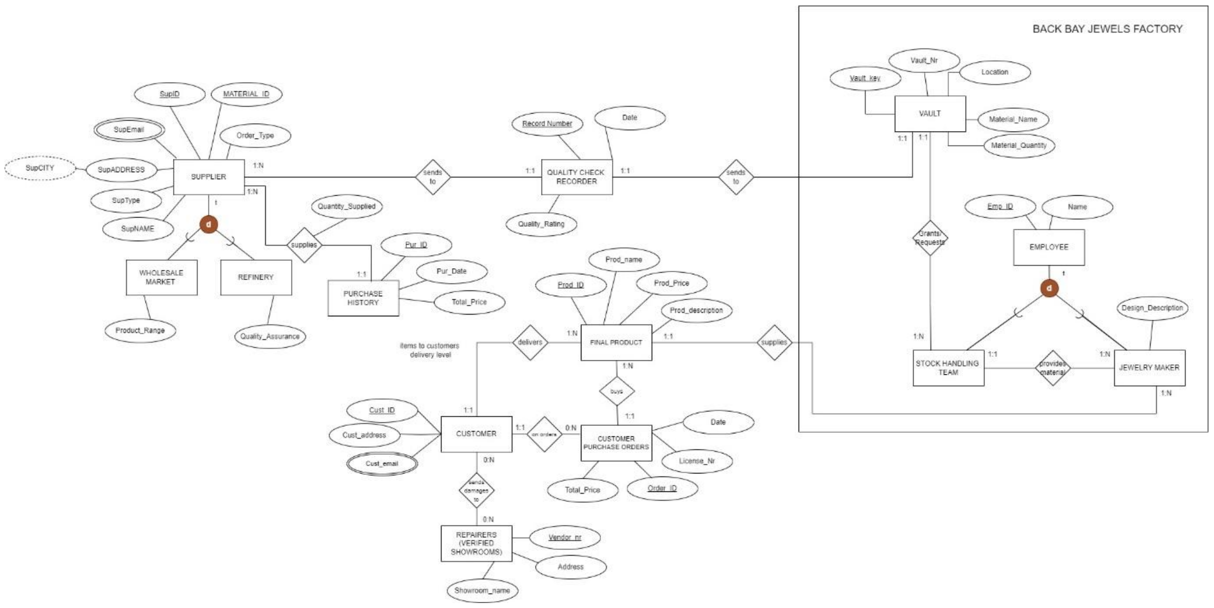
Back Bay Jewels is a jewelry and precious metal retailer. Precious metals and jewelry require a business to maintain an impeccable reputation among consumers. When dealing with high value and easily counterfeited materials, our business must keep track of every step of the jewelry-making process to stamp out any sources of imperfection or suspend trading with untrusted suppliers. Back Bay Jewels will identify all jewelry materials such as diamonds and metal by detailing each item's origin refinery and jewelry manufacturer. It is also essential to know where and how each item is being stored until it is sold, whether that be within the factory warehouse vaults for surplus and highly valuable items. For the sake of security and minimizing human interaction, with valuable items, the store will be online. The business aims to create an e-commerce platform that not only highlights its exquisite jewelry but also provides a seamless and personalized shopping experience.

Back Bay Jewels starts its journey by responsibly obtaining the essential materials for crafting jewelry, such as gold, diamonds, and platinum. This involves building trusted partnerships with reliable suppliers to ensure a steady supply of high-quality raw materials. This will come from either banks or the fine metal wholesale market. All purchases will be recorded with the place from which they were supplied. Once these precious materials are sourced, Back Bay Jewels takes on the responsibility of quality checking the fineness of materials and recording them to meet the exact standards required for crafting exquisite jewelry. The quality check process includes checking and logging the percentage of metals by spectrometers, recording thermal conductivity, and weight. All attributes of this process such as carat, desired metal percentage, weight, and cut will be scrupulously recorded. Back Bay Jewels considers the secure storage of valuable jewelry items by making use of warehouse vaults and specialized security service, enhancing the protection of its valuable inventory. Only an authorized stock-handling team of employees will be qualified to handle, track, and withdraw materials from the vaults to be crafted by the factory jewelers. Moreover, the business engages in thoughtful financial management, including interactions with banks for necessary funds and storage of surplus materials. Back Bay Jewels collaborates with skilled jewelers to design and manufacture jewelry pieces in the company's own factory. This involves not only creating intricate designs and selecting suitable settings but also ensuring that the craftsmanship aligns seamlessly with the brand's high standards.

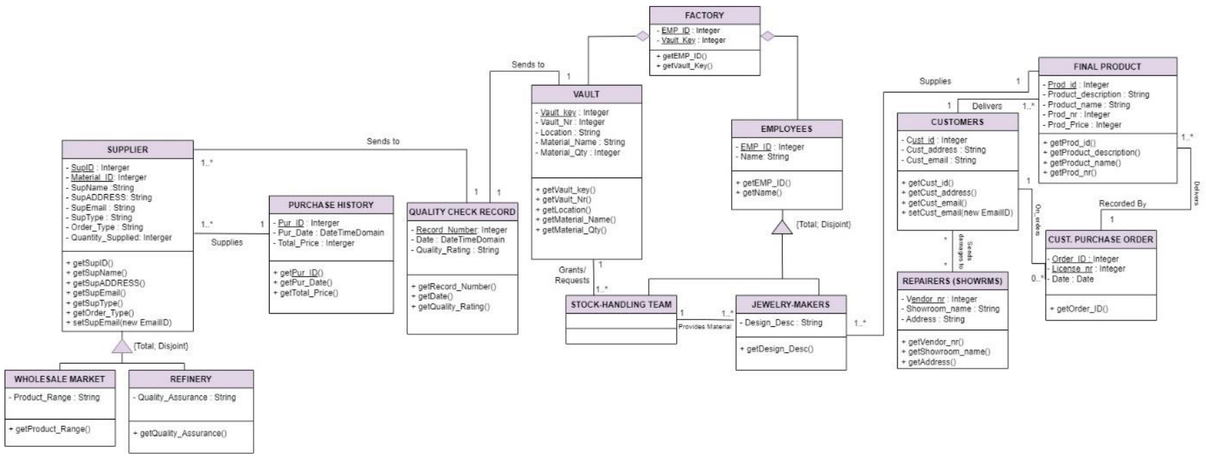
Shifting the focus towards customers, Back Bay Jewels takes on the responsibility of marketing its products to attract customers. After the sale, Back Bay Jewels continues its commitment by providing excellent customer service. This includes addressing inquiries, handling returns or exchanges, and ensuring overall customer satisfaction. The company will offer replacement or repair service for damaged or defective pieces if the customer is registered in the e-commerce platform with their Customer ID. This does not occur in the company's own factory, but it can be done at other qualified jewelry stores with whom Back Bay Jewels maintains working relationships. In the process, the business will record which items are being brought in and note where each needs repair. The final step involves coordinating the safe and secure delivery of purchased jewelry to customers. This process includes careful selection of reliable shipping partners and providing tracking information for enhanced transparency.

II. Conceptual Data Modeling

1. EER Model



2. UML Diagram



III. Mapping Conceptual Model to Relational Model

Primary Key- Underlined **Foreign Key-** ***Italicized***

1. **SUPPLIER (Sup_ID, Material_ID, Sup_Name, Sup_ADDRESS, Sup_City, Sup_Type, Order_Type, Quantity_Supplied, *Record_Number, Pur_ID*)**
   - **Record_Number**: Foreign key refers to **Record_Number** of SUPPLIER relation, NOT NULL
   - **Pur_ID**: Foreign key refers to **Pur_ID** of PURCHASE HISTORY relation, NOT NULL
2. **SUP-EMAIL (Sup_Email, *Sup_ID*)**
   - **Sup_ID**: Foreign key refers to **Sup_ID** of SUPPLIER relation, NOT NULL
3. **WHOLESALE_MARKET (Sup_ID, Material_ID, Product_Range)**
4. **REFINERY (Sup_ID, Material_ID, Quality_Assurance)**
5. **PURCHASE_HISTORY (Pur_ID, Pur_Date, Total_Price)**
6. **QUALITY_CHECK_RECORDER (Record_Number, Date, Quality_Rating)**
7. **FACTORY (*F_Vault_Key*, *F_Emp_ID*)**
   - **F_Vault_Key :** Foreign key refers to **Vault_Key** of VAULT relation, NOT NULL
   - **F_Emp_ID:** Foreign key refers to **Emp_ID** of EMPLOYER relation, NOT NULL
8. **VAULT(Vault_key, Vault_Nr, Location, Material_Name, Material_Quantity, *Record_Number*)**
   - **Record_Number**: Foreign key refers to **Record_Number** of SUPPLIER relation, NOT NULL
9. **EMPLOYEE (Emp_ID, Name)**
10. **STOCK_HANDLING_TEAM (Emp_ID, *Vault_Key*)**
    - **Emp_ID**: Foreign key referring to the **Emp_ID** primary key of EMPLOYEE, NOT NULL
    - **Vault_key**: Foreign key referring to the **Vault_key** primary key of VAULT, NOT NULL
11. **JEWELRY_MAKER (Emp_ID, *Prod_ID*, Design_Description)**
    - **Emp_ID**: Foreign key referring to the **Emp_ID** primary key of EMPLOYEE, NOT NULL
    - **Prod_ID**: Foreign key referring to the **Prod_ID** primary key of FINAL PRODUCT, NOT NULL
12. **FINAL_PRODUCT (Prod_ID, Prod_Name, Prod_Description, Prod_Price, *Cust_ID, Order_ID*)**
    - **Cust_ID:** Foreign key referring to the Cust_ID of CUSTOMER, NOT NULL
    - **Order_ID**: Foreign key referring to the Order_ID of CUST_PURCHASE_ORDERS, NOT NULL
13. **CUSTOMER (Cust_ID, Cust_address)**
14. **CUST_EMAIL (Cust_Email, *Cust_ID*)**
    - **Cust_ID:** Foreign key referring to the Cust_ID of CUSTOMER, NOT NULL
15. **REPAIRER (Vendor_Nr, Address, Showroom_name)**
16. **DAMAGES (*Cust_ID, Vendor_Nr*)**
    - **Vendor_Nr:** Foreign key referring to the **Vendor_Nr** of REPAIRER, NOT NULL
    - **Cust_ID:** Foreign key referring to **Cust_ID** of CUSTOMER, NOT NULL
17. **CUST_PURCHASE_ORDERS (Order_ID, Liscense_Nr, Total_Price, Date, *Cust_ID*)**
    - **Cust_ID:** Foreign key referring to the Cust_ID of CUSTOMER, NOT NULL

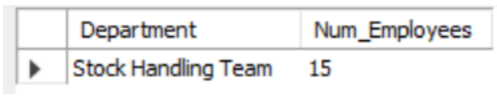
IV. Implementation of Relation Model via MySQL and NoSQL

A) MySQL Implementation

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

1) Find the no. of employees in Stock Handline Team (Simple query)

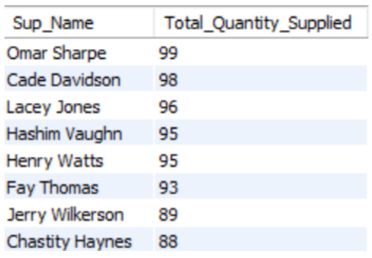
```
SELECT 'Stock Handling Team' AS Department,
COUNT(*) AS Num_Employees
FROM STOCK_HANDLING_TEAM;
```



Department	Num_Employees
Stock Handling Team	15

2) Find the Total Quantity Supplied by each Supplier (Aggregate Query)

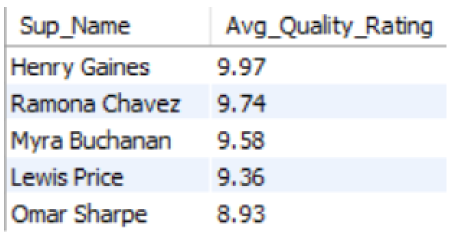
```
SELECT s.Sup_Name, SUM(s.Quantity_Supplied)
AS Total_Quantity_Supplied
FROM SUPPLIER s
GROUP BY s.Sup_Name
ORDER BY Total_Quantity_Supplied DESC;
```



Sup_Name	Total_Quantity_Supplied
Omar Sharpe	99
Cade Davidson	98
Lacey Jones	96
Hashim Vaughn	95
Henry Watts	95
Fay Thomas	93
Jerry Wilkerson	89
Chastity Haynes	88

3) Finding the Top 5 Suppliers by Average Quality Rating (Using Inner Join)

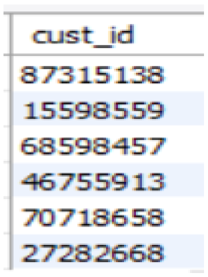
```
SELECT Sup_Name, AVG(Quality_Rating) AS
Avg_Quality_Rating
FROM QUALITY_CHECK_RECORDER q
join supplier s on q.Record_Number =
s.Record_Number
GROUP BY Sup_Name, Date
order by Avg_Quality_Rating desc limit 5;
```



Sup_Name	Avg_Quality_Rating
Henry Gaines	9.97
Ramona Chavez	9.74
Myra Buchanan	9.58
Lewis Price	9.36
Omar Sharpe	8.93

4) Finding the customer ids from Virginia, Vermont and Kentucky who received the damaged product (Nested query)

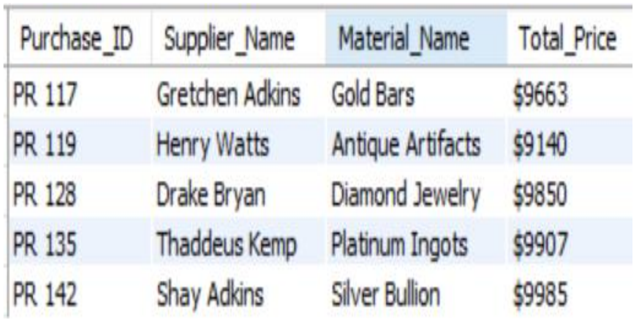
```
SELECT cust_id
FROM damages
WHERE vendor_nr IN
(SELECT vendor_nr
FROM repairer
WHERE address LIKE '%virginia%'
OR address LIKE '%verm%'
OR address LIKE '%kentu%');
```



cust_id
87315138
15598559
68598457
46755913
70718658
27282668

5) Find the five highest priced purchases together with their Pur_IDs, Material_Name and Material Quantity (Correlated query)

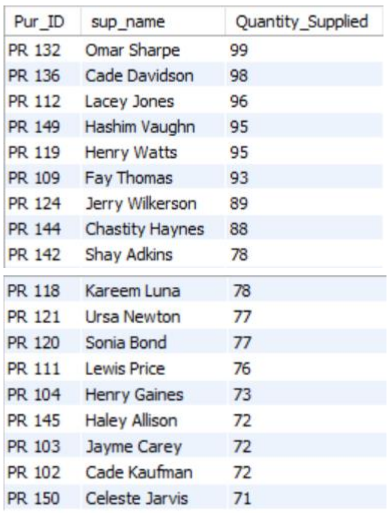
```
SELECT PH1.Pur_ID Purchase_ID,
s.Sup_name Supplier_Name,
v.Material_Name, PH1.Total_Price
FROM purchase_history PH1
join supplier s on s.Pur_ID = ph1.pur_id
join quality_check_recorder q on
q.Record_Number= s.Record_Number
join vault v on v.record_number=
q.Record_Number
WHERE 5 >
          (SELECT count(*)
   FROM purchase_history PH2
   WHERE PH1.Total_Price <
PH2.Total_Price);
```



Purchase_ID	Supplier_Name	Material_Name	Total_Price
PR 117	Gretchen Adkins	Gold Bars	$9663
PR 119	Henry Watts	Antique Artifacts	$9140
PR 128	Drake Bryan	Diamond Jewelry	$9850
PR 135	Thaddeus Kemp	Platinum Ingots	$9907
PR 142	Shay Adkins	Silver Bullion	$9985

6) This query retrives the pur_id and supplier_name where the supplies quantity is greater than 70 (Using Exists)

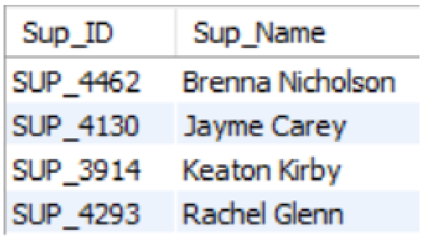
```
SELECT p.Pur_ID, s.sup_name,
s.Quantity_Supplied
FROM purchase_history p
JOIN supplier s ON p.Pur_ID =
s.Pur_ID
WHERE EXISTS
(SELECT 1
FROM supplier s2
WHERE s2.Pur_ID = p.Pur_ID
AND s2.Quantity_Supplied > 70)
order by quantity_supplied desc;
```



Pur_ID	sup_name	Quantity_Supplied
PR 132	Omar Sharpe	99
PR 136	Cade Davidson	98
PR 112	Lacey Jones	96
PR 149	Hashim Vaughn	95
PR 119	Henry Watts	95
PR 109	Fay Thomas	93
PR 124	Jerry Wilkerson	89
PR 144	Chastity Haynes	88
PR 142	Shay Adkins	78
PR 118	Kareem Luna	78
PR 121	Ursa Newton	77
PR 120	Sonia Bond	77
PR 111	Lewis Price	76
PR 104	Henry Gaines	73
PR 145	Haley Allison	72
PR 103	Jayme Carey	72
PR 102	Cade Kaufman	72
PR 150	Celeste Jarvis	71

7) This query returns the suppliers from New England who has sold > $5000 (Set operations (Union))

```
SELECT Sup_ID, Sup_Name
FROM Supplier
WHERE Sup_Address LIKE '%Rhode%'
OR Sup_Address LIKE '%Connect%'
OR Sup_Address LIKE '%Massac%'
OR Sup_Address LIKE '%Maine%'
OR Sup_Address LIKE '%New Ham%'
```

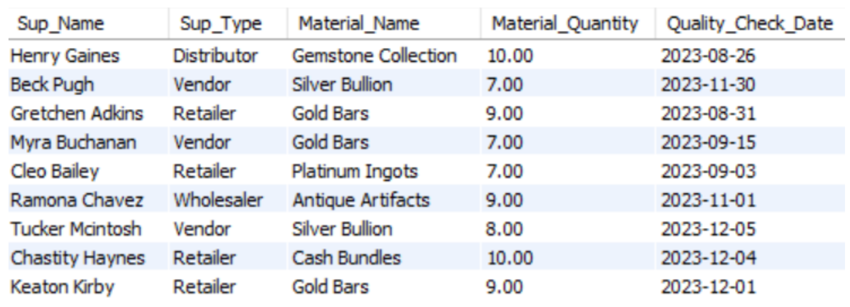


Sup_ID	Sup_Name
SUP_4462	Brenna Nicholson
SUP_4130	Jayme Carey
SUP_3914	Keaton Kirby
SUP_4293	Rachel Glenn

```
OR Sup_Address LIKE '%Vermont%'
UNION
SELECT S.Sup_ID, S.Sup_Name
FROM Supplier S, purchase_history PH
WHERE S.Pur_ID = PH.Pur_ID
AND PH.Total_Price > 5000
ORDER BY Sup_Name ASC;
```

8) This query selects the quality_checked_date along with the supplier's information who has supplied materials from August 1st until the end of the year (Subqueries in Select)

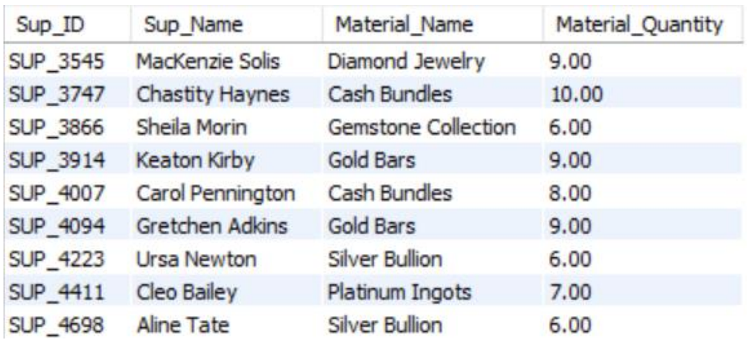
```
SELECT s.Sup_Name, s.Sup_Type, v.Material_Name,v.Material_Quantity,
    (SELECT q.Date
     FROM quality_check_recorder q
     WHERE q.Record_Number = s.Record_Number) AS Quality_Check_Date
FROM supplier s
INNER JOIN vault v ON s.Record_Number = v.record_number
WHERE (SELECT q.Date
       FROM quality_check_recorder q
       WHERE q.Record_Number = s.Record_Number) > '2023-08-01' And
       Material_Quantity > 5;
```



Sup_Name	Sup_Type	Material_Name	Material_Quantity	Quality_Check_Date
Henry Gaines	Distributor	Gemstone Collection	10.00	2023-08-26
Beck Pugh	Vendor	Silver Bullion	7.00	2023-11-30
Gretchen Adkins	Retailer	Gold Bars	9.00	2023-08-31
Myra Buchanan	Vendor	Gold Bars	7.00	2023-09-15
Cleo Bailey	Retailer	Platinum Ingots	7.00	2023-09-03
Ramona Chavez	Wholesaler	Antique Artifacts	9.00	2023-11-01
Tucker Mcintosh	Vendor	Silver Bullion	8.00	2023-12-05
Chastity Haynes	Retailer	Cash Bundles	10.00	2023-12-04
Keaton Kirby	Retailer	Gold Bars	9.00	2023-12-01

9) This query selects the suppliers who are retailers and who have supplied materials more than 5 (Subqueries in From)

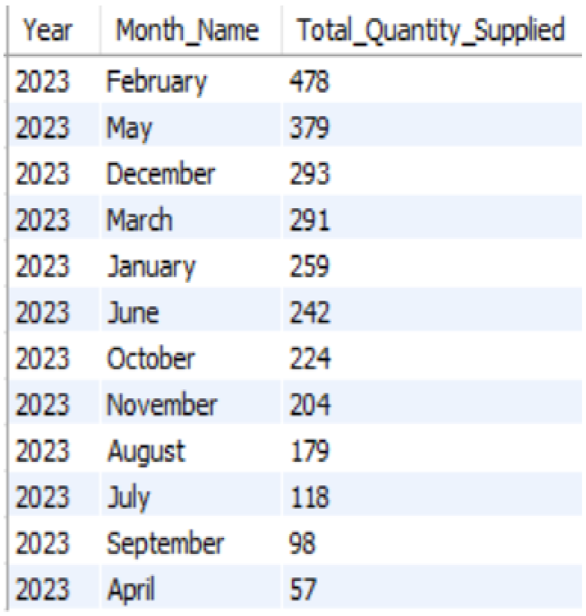
```
SELECT Sup_ID, Sup_name, v.Material_Name, v.Material_Quantity

FROM (

   SELECT *

   FROM supplier

   WHERE Sup_Type = 'Retailer'

) AS s

INNER JOIN quality_check_recorder q ON s.Record_Number = q.Record_Number
INNER JOIN vault v ON s.Record_Number = v.Record_Number

where Material_Quantity > 5;
```



Sup_ID	Sup_Name	Material_Name	Material_Quantity
SUP_3545	MacKenzie Solis	Diamond Jewelry	9.00
SUP_3747	Chastity Haynes	Cash Bundles	10.00
SUP_3866	Sheila Morin	Gemstone Collection	6.00
SUP_3914	Keaton Kirby	Gold Bars	9.00
SUP_4007	Carol Pennington	Cash Bundles	8.00
SUP_4094	Gretchen Adkins	Gold Bars	9.00
SUP_4223	Ursa Newton	Silver Bullion	6.00
SUP_4411	Cleo Bailey	Platinum Ingots	7.00
SUP_4698	Aline Tate	Silver Bullion	6.00

10) This query is to find the total number of quantity supplied by suppliers for each month in the year 2023 (Using CTE's)

```
WITH Monthly_Supply AS (SELECT
YEAR(q.Date) AS Year,
 MONTHNAME(q.Date) AS
Month_Name,
        SUM(s.Quantity_Supplied) AS
Total_Quantity_Supplied
    FROM
        supplier s
    INNER JOIN
        quality_check_recorder q ON
s.Record_Number = q.Record_Number
    GROUP BY
        YEAR(q.Date),
MONTHNAME(q.Date))
SELECT Year, Month_Name,
Total_Quantity_Supplied
FROM  Monthly_Supply
ORDER BY Total_Quantity_Supplied
DESC;
```

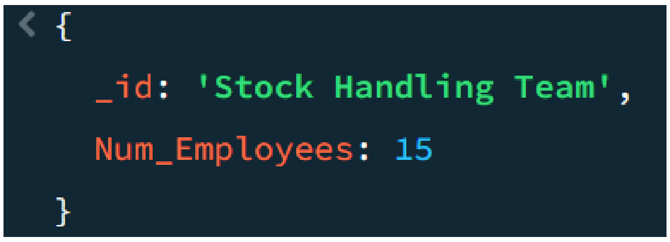


Year	Month_Name	Total_Quantity_Supplied
2023	February	478
2023	May	379
2023	December	293
2023	March	291
2023	January	259
2023	June	242
2023	October	224
2023	November	204
2023	August	179
2023	July	118
2023	September	98
2023	April	57

B) NoSQL Implementation (Using MongoDB)

1) Count the number of employees in Stock Handling Team

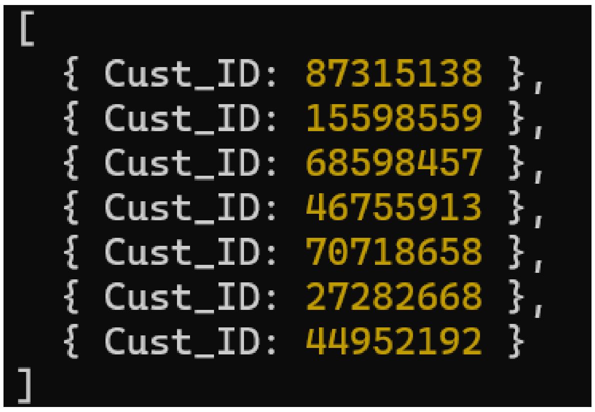
```
 db.stock_handling_team.aggregate([
   {$group: {
       _id: "Stock Handling Team",
       Num_Employees: { $sum: 1 } }
   }])
```



```
< {
    _id: 'Stock Handling Team',
    Num_Employees: 15
  }
```

2) Finding the customer ids from Virginia, Vermont and Kentucky who received the damaged product.

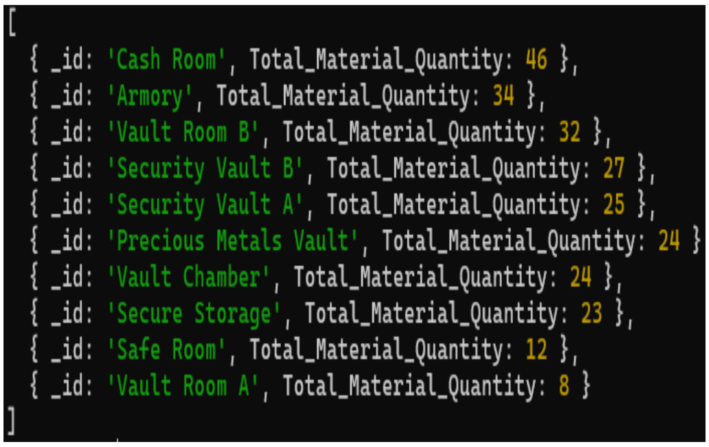
```
 db.damages.find({  Vendor_Nr:
  {    $in: db.repairer.distinct("Vendor_Nr",
 {      Address: {
  $regex:
 /virginia|verm|kentu/i      }    })  }},
 { Cust_ID: 1, _id: 0 });
```



```
[
  { Cust_ID: 87315138 },
  { Cust_ID: 15598559 },
  { Cust_ID: 68598457 },
  { Cust_ID: 46755913 },
  { Cust_ID: 70718658 },
  { Cust_ID: 27282668 },
  { Cust_ID: 44952192 }
]
```

3) Query to retrieve the top 5 locations in the vault, sorted by the total quantity of materials stored in each location.

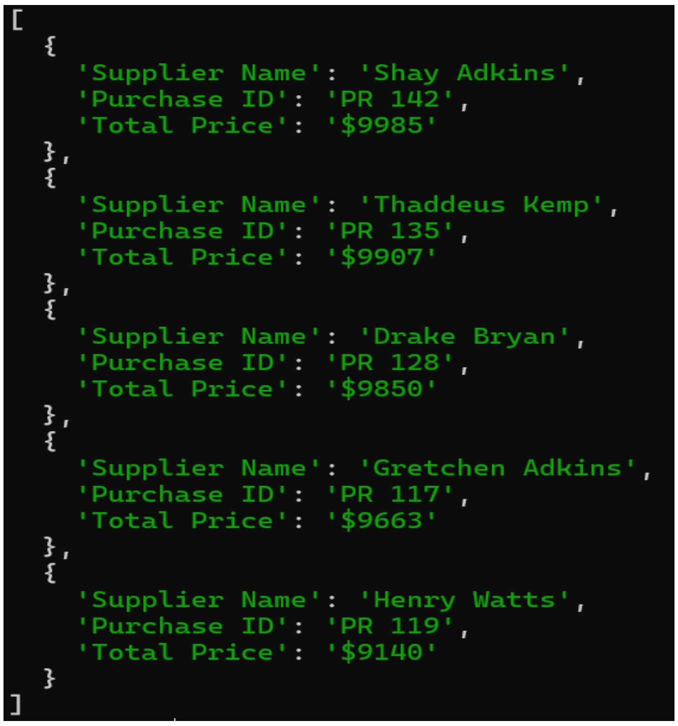
```
 db.vault.aggregate([
  {
    $group: {
      _id: "$Location",
      Total_Material_Quantity:
 { $sum: "$Material_Quantity" }
    }
  },
  {
    $sort:
 { Total_Material_Quantity: -1 }
  },
 {
    $limit: 5
  }
 ])
```



```
[
  { _id: 'Cash Room', Total_Material_Quantity: 46 },
  { _id: 'Armory', Total_Material_Quantity: 34 },
  { _id: 'Vault Room B', Total_Material_Quantity: 32 },
  { _id: 'Security Vault B', Total_Material_Quantity: 27 },
  { _id: 'Security Vault A', Total_Material_Quantity: 25 },
  { _id: 'Precious Metals Vault', Total_Material_Quantity: 24 }
  { _id: 'Vault Chamber', Total_Material_Quantity: 24 },
  { _id: 'Secure Storage', Total_Material_Quantity: 23 },
  { _id: 'Safe Room', Total_Material_Quantity: 12 },
  { _id: 'Vault Room A', Total_Material_Quantity: 8 }
]
```

4) Query to retrieve the top 5 purchases along with the supplier's name and sorts the results based on the total price in descending order.

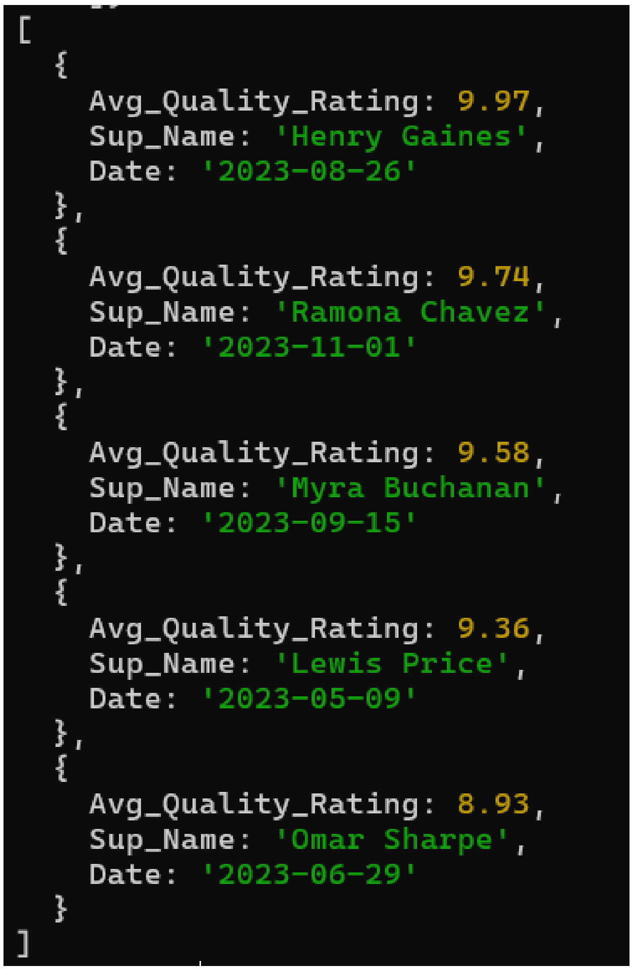
```
db.supplier.aggregate([
 {
  $lookup: {
   from: "purchase_history",
   localField: "Pur_id",
   foreignField: "Pur_ID",
   as: "purchase_info"
  }
 },
 {
  $unwind: "$purchase_info"
 },
 {
  $project: {
   _id: 0,
   "Supplier Name":
"$Sup_Name",
   "Purchase ID": "$Pur_id",
   "Total Price":
"$purchase_info.Total_Price"
  }
 },
 {
  $sort: { "Total Price": -1 }
 },
 {
  $limit: 5
 }
])
```



```
[
  {
    'Supplier Name': 'Shay Adkins',
    'Purchase ID': 'PR 142',
    'Total Price': '$9985'
  },
  {
    'Supplier Name': 'Thaddeus Kemp',
    'Purchase ID': 'PR 135',
    'Total Price': '$9907'
  },
  {
    'Supplier Name': 'Drake Bryan',
    'Purchase ID': 'PR 128',
    'Total Price': '$9850'
  },
  {
    'Supplier Name': 'Gretchen Adkins',
    'Purchase ID': 'PR 117',
    'Total Price': '$9663'
  },
  {
    'Supplier Name': 'Henry Watts',
    'Purchase ID': 'PR 119',
    'Total Price': '$9140'
  }
]
```

5) Query to retrieve the top 5 average quality ratings for suppliers, along with their names and corresponding dates

```
db.quality_check_recorder.aggregate([
  {
    $lookup: {
      from: "supplier",
      localField: "Record_Number",
      foreignField: "Record_Number",
      as: "supplier"
    }
  },
  {
    $unwind: "$supplier"
  },
  {
    $group: {
      _id: { Sup_Name:
"$supplier.Sup_Name", Date:
"$Date" },
      Avg_Quality_Rating: { $avg:
"$Quality_Rating" }
    }
  },
  {
    $sort: { Avg_Quality_Rating: -1 }
  },
  {
    $limit: 5
  },
  {
    $project: {
      _id: 0,
      Sup_Name: "$_id.Sup_Name",
      Date: "$_id.Date",
      Avg_Quality_Rating: 1
    }
  }
])
```

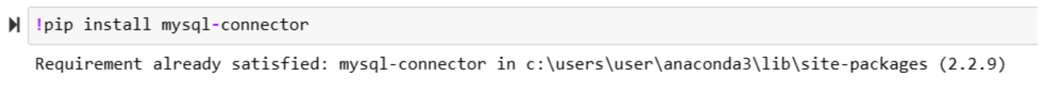


```
[
  {
    Avg_Quality_Rating: 9.97,
    Sup_Name: 'Henry Gaines',
    Date: '2023-08-26'
  },
  {
    Avg_Quality_Rating: 9.74,
    Sup_Name: 'Ramona Chavez',
    Date: '2023-11-01'
  },
  {
    Avg_Quality_Rating: 9.58,
    Sup_Name: 'Myra Buchanan',
    Date: '2023-09-15'
  },
  {
    Avg_Quality_Rating: 9.36,
    Sup_Name: 'Lewis Price',
    Date: '2023-05-09'
  },
  {
    Avg_Quality_Rating: 8.93,
    Sup_Name: 'Omar Sharpe',
    Date: '2023-06-29'
  }
]
```

VI. Database Access via Python

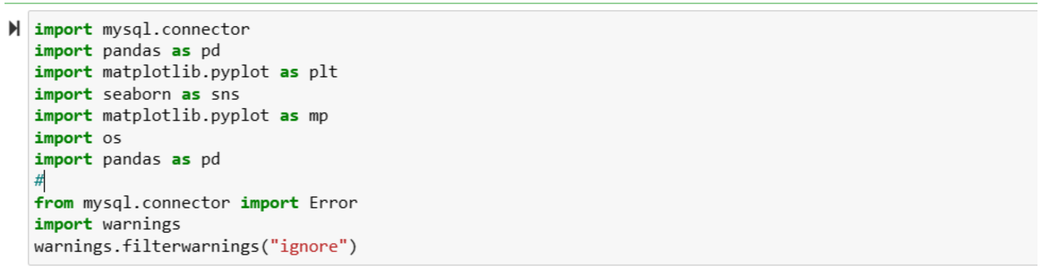
The database is accessed using Python and visualization of analysed data is shown below. The connection of MySQL to Python is done using mysql.connector, followed by cursor.execute to run and fetch all from query, followed by converting the list into a dataframe using pandas library and using matplotlib to plot the graphs for the analytics.

1) Connecting to DataBase

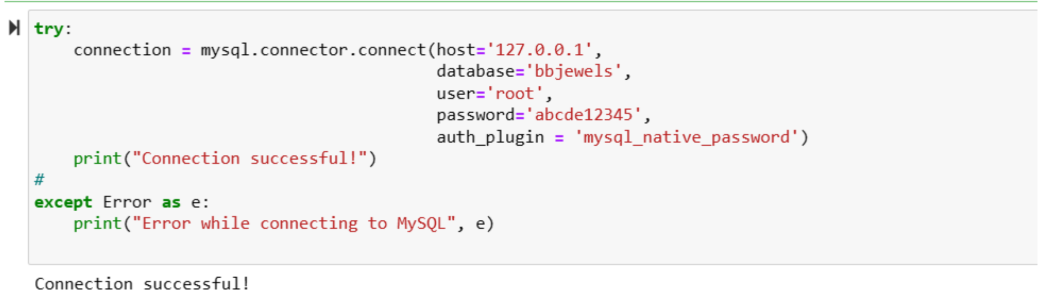


```
!pip install mysql-connector
```

```
Requirement already satisfied: mysql-connector in c:\users\user\anaconda3\lib\site-packages (2.2.9)
```



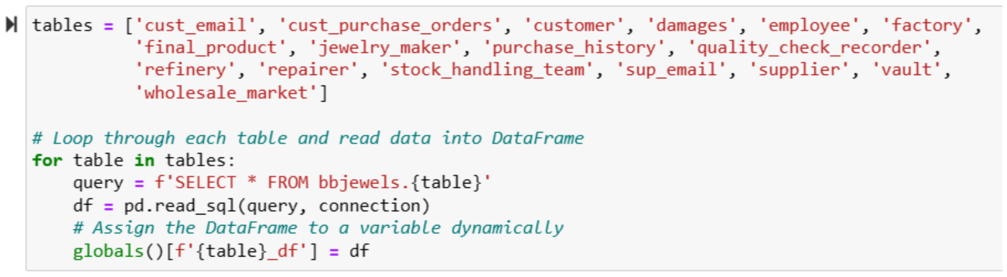
```
import mysql.connector
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.pyplot as mp
import os
import pandas as pd
#
from mysql.connector import Error
import warnings
warnings.filterwarnings("ignore")
```



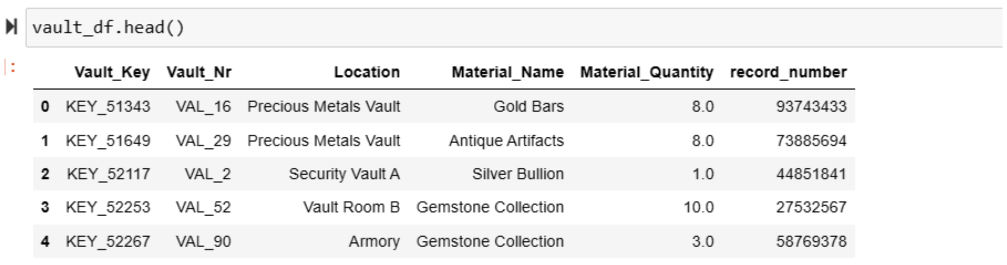
```
try:
    connection = mysql.connector.connect(host='127.0.0.1',
                                         database='bbjewels',
                                         user='root',
                                         password='abcde12345',
                                         auth_plugin = 'mysql_native_password')
    print("Connection successful!")
#
except Error as e:
    print("Error while connecting to MySQL", e)
```

```
Connection successful!
```

2) Data Collection and stored in individual dataframe



```
tables = ['cust_email', 'cust_purchase_orders', 'customer', 'damages', 'employee', 'factory',
          'final_product', 'jewelry_maker', 'purchase_history', 'quality_check_recorder',
          'refinery', 'repairer', 'stock_handling_team', 'sup_email', 'supplier', 'vault',
          'wholesale_market']

# Loop through each table and read data into DataFrame
for table in tables:
    query = f'SELECT * FROM bbjewels.{table}'
    df = pd.read_sql(query, connection)
    # Assign the DataFrame to a variable dynamically
    globals()[f'{table}_df'] = df
```



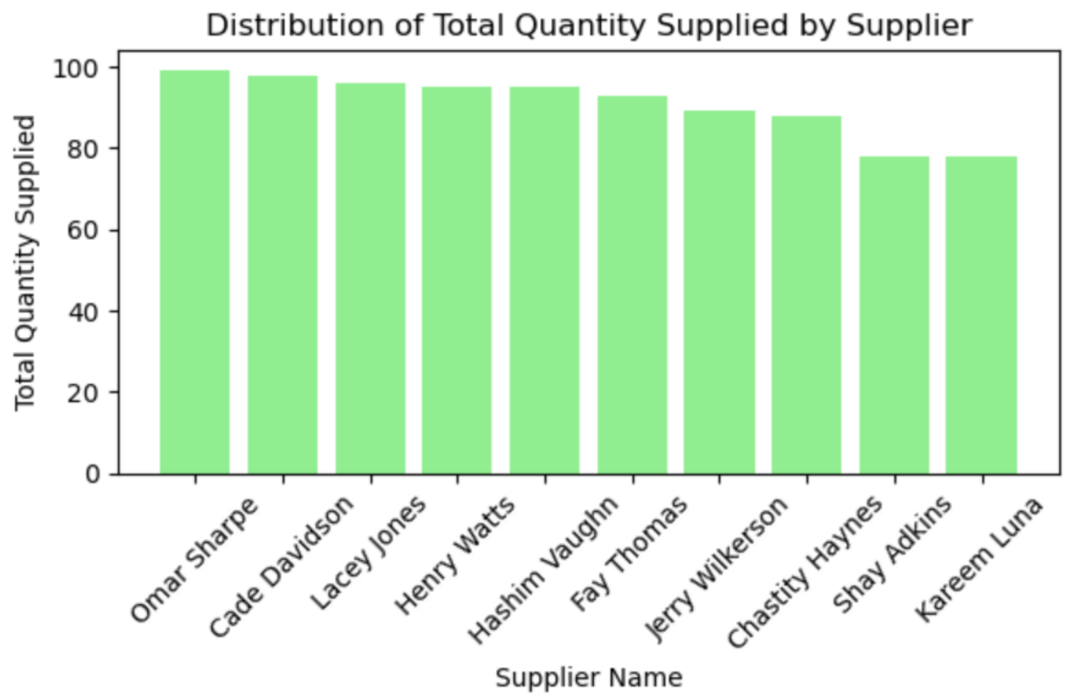
```
vault_df.head()
```

	Vault_Key	Vault_Nr	Location	Material_Name	Material_Quantity	record_number
0	KEY_51343	VAL_16	Precious Metals Vault	Gold Bars	8.0	93743433
1	KEY_51649	VAL_29	Precious Metals Vault	Antique Artifacts	8.0	73885694
2	KEY_52117	VAL_2	Security Vault A	Silver Bullion	1.0	44851841
3	KEY_52253	VAL_52	Vault Room B	Gemstone Collection	10.0	27532567
4	KEY_52267	VAL_90	Armory	Gemstone Collection	3.0	58769378

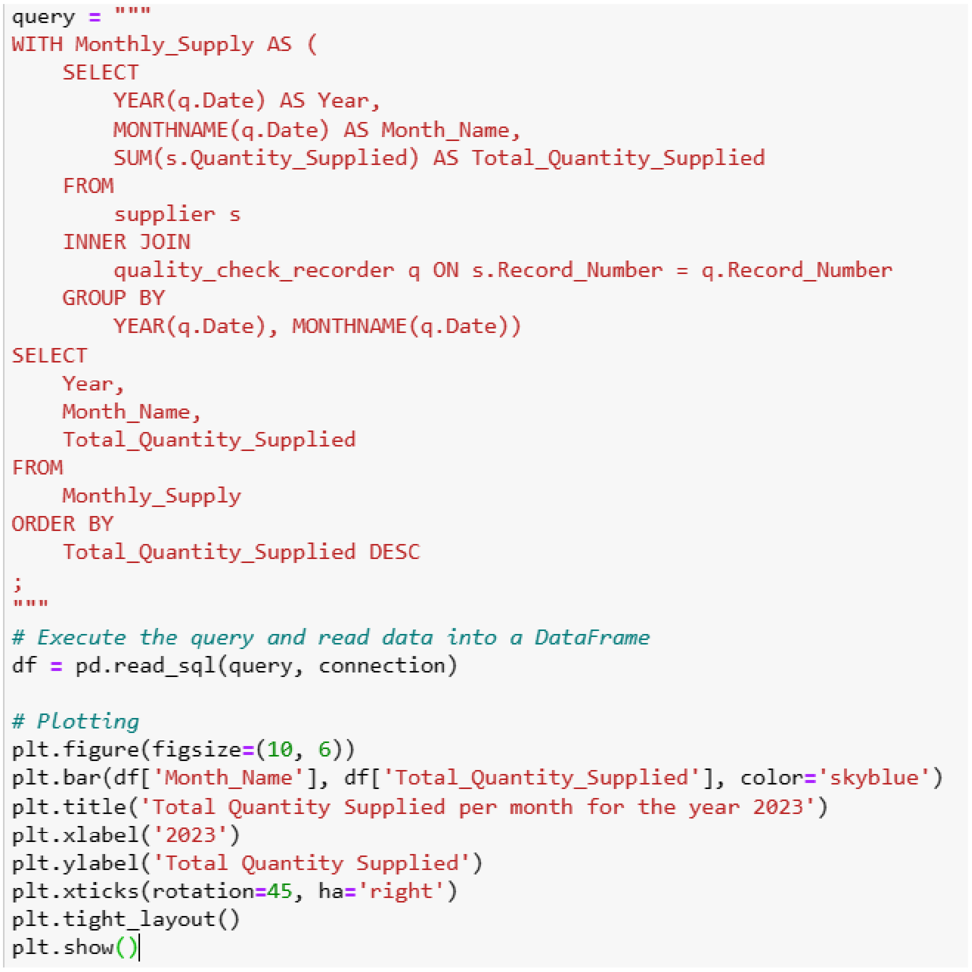
Query – 1:

1) This query finds and plots the total quantity supplied by each supplier

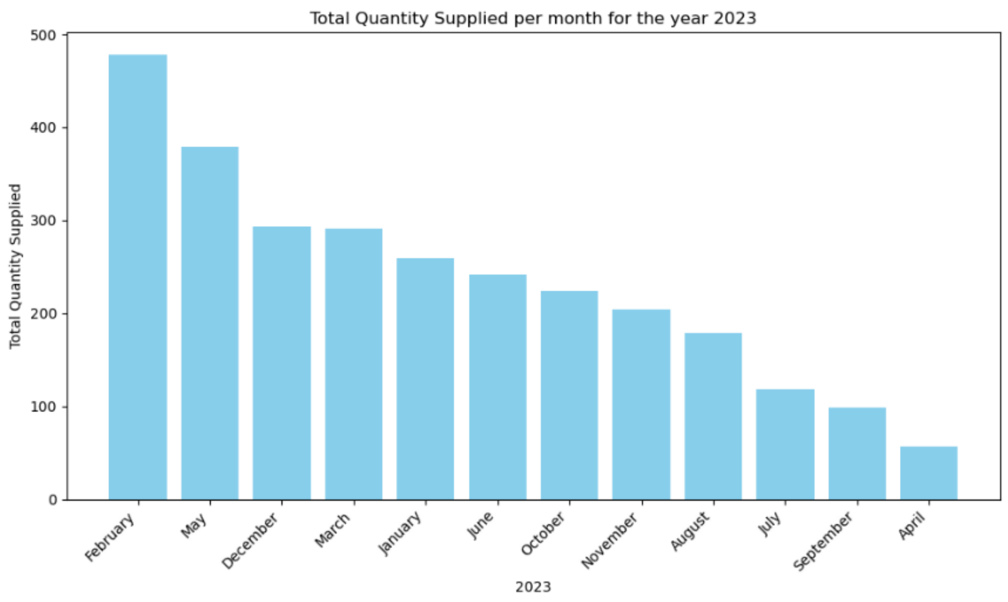
```
# Define the SQL query
query = """
    SELECT s.Sup_Name, SUM(s.Quantity_Supplied) AS Total_Quantity_Supplied
    FROM SUPPLIER s
    GROUP BY s.Sup_Name
    ORDER BY Total_Quantity_Supplied DESC
    Limit 10;
"""

# Execute the query and read data into a DataFrame
df = pd.read_sql(query, connection)

# Plotting the bar chart
plt.figure(figsize=(6, 4))
plt.bar(df['Sup_Name'], df['Total_Quantity_Supplied'], color='lightgreen')
plt.title('Distribution of Total Quantity Supplied by Supplier')
plt.xlabel('Supplier Name')
plt.ylabel('Total Quantity Supplied')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



2) This query finds and plots the total number of quantities supplied by suppliers for each month in the year 2023

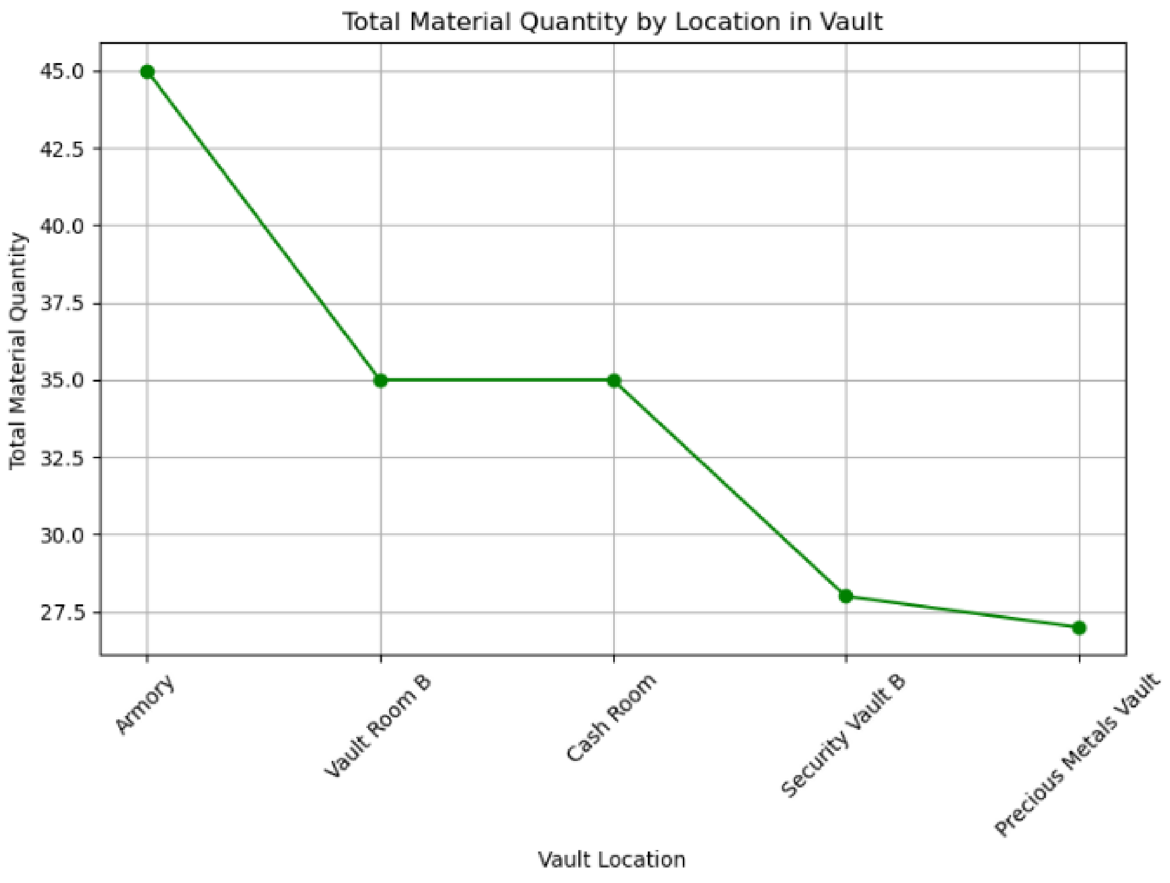


```
query = """
WITH Monthly_Supply AS (
    SELECT
        YEAR(q.Date) AS Year,
        MONTHNAME(q.Date) AS Month_Name,
        SUM(s.Quantity_Supplied) AS Total_Quantity_Supplied
    FROM
        supplier s
    INNER JOIN
        quality_check_recorder q ON s.Record_Number = q.Record_Number
    GROUP BY
        YEAR(q.Date), MONTHNAME(q.Date))
SELECT
    Year,
    Month_Name,
    Total_Quantity_Supplied
FROM
    Monthly_Supply
ORDER BY
    Total_Quantity_Supplied DESC
;
"""
# Execute the query and read data into a DataFrame
df = pd.read_sql(query, connection)

# Plotting
plt.figure(figsize=(10, 6))
plt.bar(df['Month_Name'], df['Total_Quantity_Supplied'], color='skyblue')
plt.title('Total Quantity Supplied per month for the year 2023')
plt.xlabel('2023')
plt.ylabel('Total Quantity Supplied')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

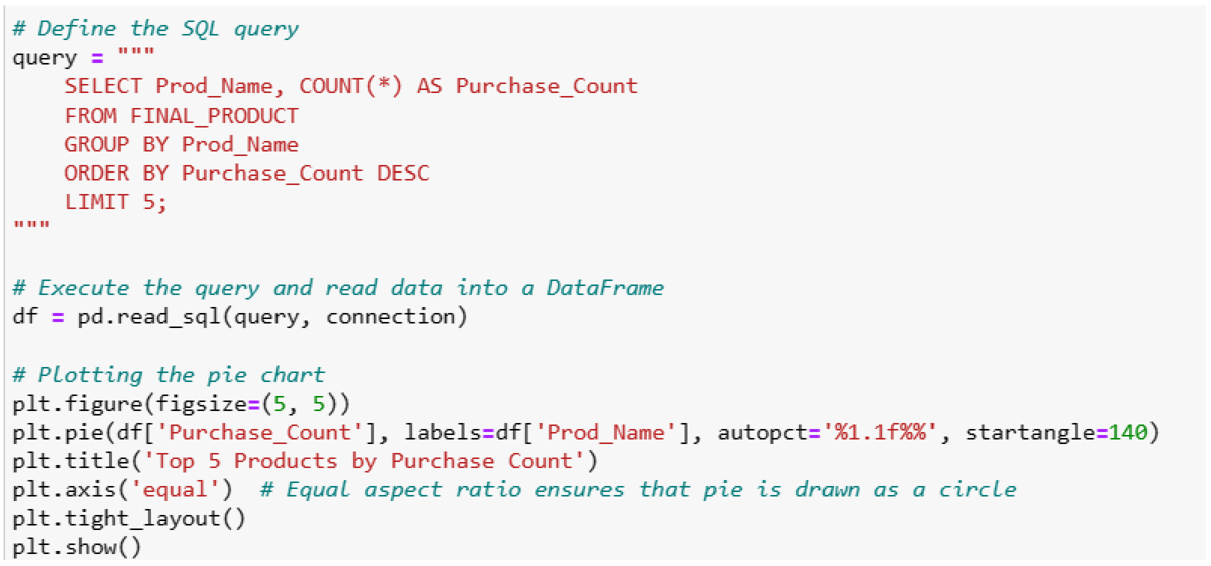


3) This query finds and plots the total number of materials available in each vault's location

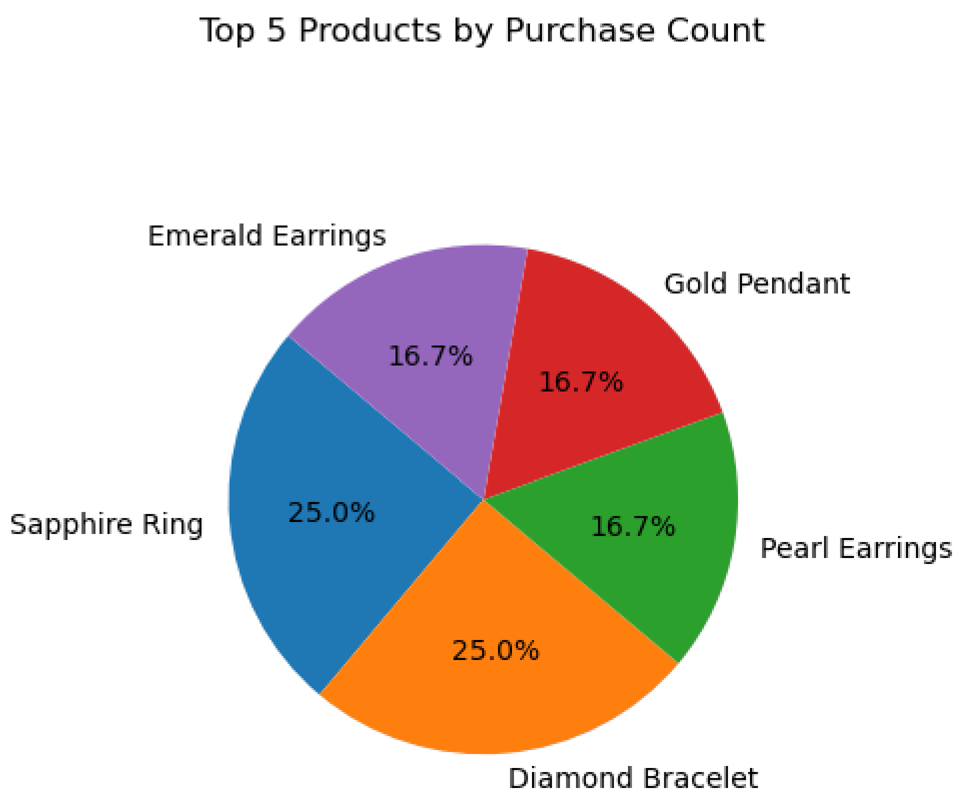
```
# Define the SQL query
query = """
    SELECT Location as Vault_Location, SUM(Material_Quantity) AS Total_Material_Quantity
    FROM VAULT
    GROUP BY Location
    ORDER BY Total_Material_Quantity DESC
    limit 5;
"""

# Execute the query and read data into a DataFrame
df = pd.read_sql(query, connection)

# Plotting the line chart
plt.figure(figsize=(8, 6))
plt.plot(df['Vault_Location'], df['Total_Material_Quantity'], marker='o', color='green', linestyle='-')
plt.title('Total Material Quantity by Location in Vault')
plt.xlabel('Vault Location')
plt.ylabel('Total Material Quantity')
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```



4) This query returns the Top 5 products purchased by customers

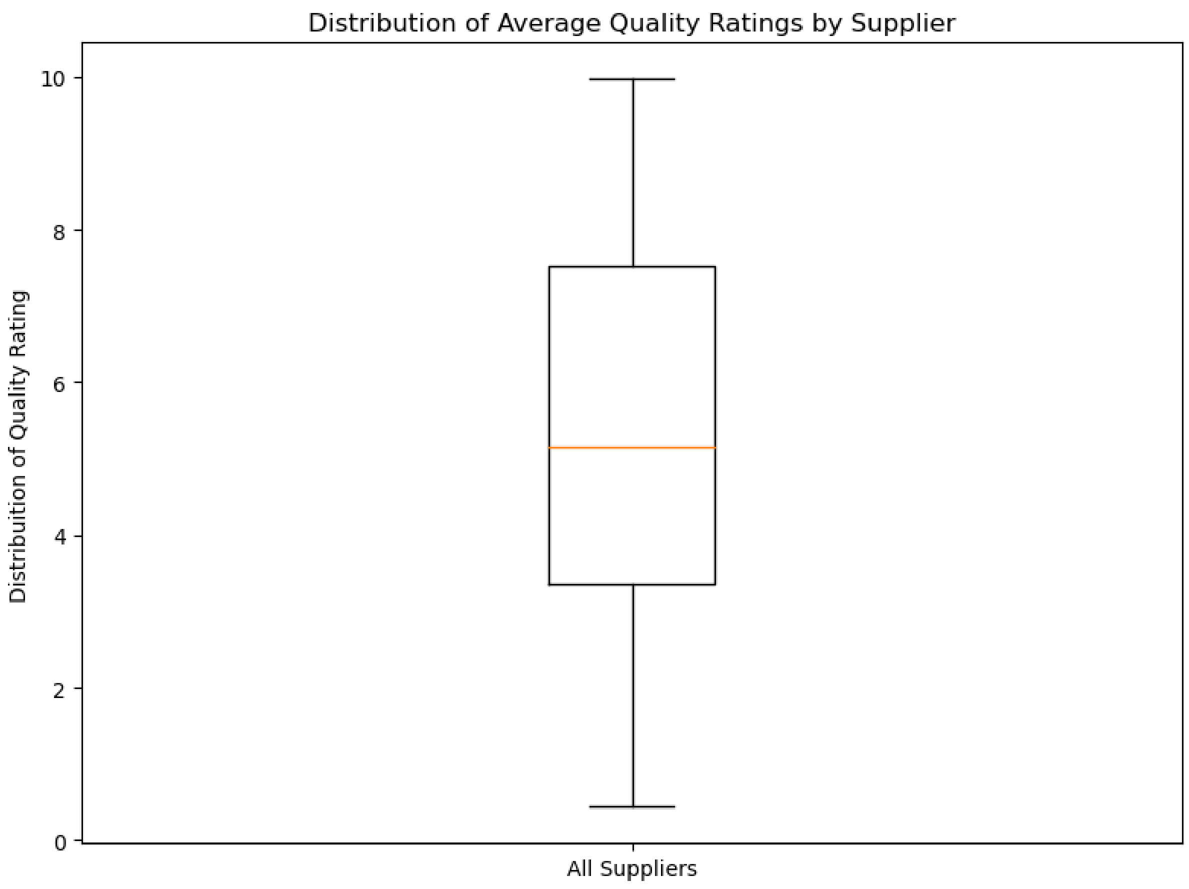


```
# Define the SQL query
query = """
    SELECT Prod_Name, COUNT(*) AS Purchase_Count
    FROM FINAL_PRODUCT
    GROUP BY Prod_Name
    ORDER BY Purchase_Count DESC
    LIMIT 5;
"""

# Execute the query and read data into a DataFrame
df = pd.read_sql(query, connection)

# Plotting the pie chart
plt.figure(figsize=(5, 5))
plt.pie(df['Purchase_Count'], labels=df['Prod_Name'], autopct='%1.1f%%', startangle=140)
plt.title('Top 5 Products by Purchase Count')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle
plt.tight_layout()
plt.show()
```



5) Finding the distribution of quality rating

```
# Define the SQL query to retrieve the average quality rating for each supplier
query = """
    SELECT Sup_Name, AVG(Quality_Rating) AS Avg_Quality_Rating
    FROM QUALITY_CHECK_RECORDER q
    JOIN supplier s ON q.Record_Number = s.Record_Number
    GROUP BY Sup_Name;
"""

# Execute the query and read data into a DataFrame
df_avg_quality = pd.read_sql(query, connection)

# Create a box plot
plt.figure(figsize=(8, 6))
plt.boxplot(df_avg_quality['Avg_Quality_Rating'])
plt.title('Distribution of Average Quality Ratings by Supplier')
plt.ylabel('Distribuition of Quality Rating')
plt.xticks([1], ['All Suppliers'])
plt.tight_layout()
plt.show()
```



VII. Summary and Recommendation

The relational database built in MySQL workbench is ready to use in the jewellery industry. In the case of this project, this database was designed around the theoretical company, Back Bay Jewels. For the project to become reality, little would need to be changed in the preexisting framework. Although, for the sake of maintaining high standards and reputability, more quality control measures could be tracked and recorded. For example, instead of assigning a score on the 1 to 10 scale used in the quality check record, the company should explicitly list many more attributes of the material that have finite measurement systems. These could include, weight, carat, hardness, density, ray fluorescent analysis, spectroscopy, etc. This is a much more wholistic approach as opposed to a single 10-point spectrum for all materials.

Also, since the enterprise would run all of its retail operations entirely online, the customer service portion of the database could be expanded to personalize the shopping experience and allow the business to better market to these individuals. Personal details such as ring size or product preferences could be used.