Muhammad Fauzy Syapputra

To address the challenges specified in the "Project Missing Money Matters" PDF, we'll need to examine and run SQL queries on the WSDA_Music database. Below are the steps and SQL queries necessary to complete each challenge.

Step 1: Setting Up the Database

First, we need to set up the database from the provided SQL file. We'll do this in a Python environment to execute the SQL queries.

Step 2: Analyzing the Challenges and Running Queries

Challenge 1: General Queries

1. How many transactions took place between the years 2011 and 2012?
2. How much money did WSDA Music make during the same period?

Challenge 2: Targeted Questions

1. Get a list of customers who made purchases between 2011 and 2012.
2. Get a list of customers, sales reps, and total transaction amounts for each customer between 2011 and 2012.
3. How many transactions are above the average transaction amount during the same time period?
4. What is the average transaction amount for each year that WSDA Music has been in business?

Challenge 3: In-depth Analysis

1. Get a list of employees who exceeded the average transaction amount from sales they generated during 2011 and 2012.
2. Create a Commission Payout column that displays each employee's commission based on 15% of the sales transaction amount.
3. Which employee made the highest commission?
4. List the customers that the employee identified in the last question.
5. Which customer made the highest purchase?
6. Look at this customer record—do you see anything suspicious?
7. Who do you conclude is our primary person of interest?

SQL Queries for Each Challenge

Challenge 1 Queries

```sql
-- 1. Number of transactions between 2011 and 2012
SELECT COUNT(*) AS transaction_count
FROM Invoices
WHERE InvoiceDate BETWEEN '2011-01-01' AND '2012-12-31';

-- 2. Total revenue between 2011 and 2012
SELECT SUM(Total) AS total_revenue
FROM Invoices
WHERE InvoiceDate BETWEEN '2011-01-01' AND '2012-12-31';
```

Muhammad Fauzy Syapputra

Challenge 2 Queries

```sql
-- 1. List of customers who made purchases between 2011 and 2012
SELECT DISTINCT CustomerID
FROM Invoices
WHERE InvoiceDate BETWEEN '2011-01-01' AND '2012-12-31';

-- 2. List of customers, sales reps, and total transaction amounts for each
customer between 2011 and 2012
SELECT c.CustomerID, e.EmployeeID, SUM(i.Total) AS total_amount
FROM Invoices i
JOIN Customers c ON i.CustomerID = c.CustomerID
JOIN Employees e ON i.EmployeeID = e.EmployeeID
WHERE i.InvoiceDate BETWEEN '2011-01-01' AND '2012-12-31'
GROUP BY c.CustomerID, e.EmployeeID;

-- 3. Number of transactions above the average transaction amount between
2011 and 2012
WITH avg_transaction AS (
    SELECT AVG(Total) AS average_amount
    FROM Invoices
    WHERE InvoiceDate BETWEEN '2011-01-01' AND '2012-12-31'
)
SELECT COUNT(*)
FROM Invoices
WHERE Total > (SELECT average_amount FROM avg_transaction)
  AND InvoiceDate BETWEEN '2011-01-01' AND '2012-12-31';

-- 4. Average transaction amount for each year
SELECT strftime('%Y', InvoiceDate) AS year, AVG(Total) AS average_amount
FROM Invoices
GROUP BY year;
```

Challenger 3 Queries

```sql
-- 1. Employees who exceeded the average transaction amount from sales they
generated during 2011 and 2012
WITH avg_transaction AS (
    SELECT AVG(Total) AS average_amount
    FROM Invoices
    WHERE InvoiceDate BETWEEN '2011-01-01' AND '2012-12-31'
)
SELECT e.EmployeeID, e.LastName, e.FirstName
FROM Employees e
JOIN Invoices i ON e.EmployeeID = i.EmployeeID
WHERE i.InvoiceDate BETWEEN '2011-01-01' AND '2012-12-31'
GROUP BY e.EmployeeID
HAVING AVG(i.Total) > (SELECT average_amount FROM avg_transaction);

-- 2. Create a Commission Payout column
SELECT i.InvoiceID, i.Total, i.EmployeeID, (i.Total * 0.15) AS
commission_payout
FROM Invoices i
WHERE i.InvoiceDate BETWEEN '2011-01-01' AND '2012-12-31';

-- 3. Employee with the highest commission
SELECT EmployeeID, SUM(Total * 0.15) AS total_commission
FROM Invoices
WHERE InvoiceDate BETWEEN '2011-01-01' AND '2012-12-31'
GROUP BY EmployeeID
```

```sql
ORDER BY total_commission DESC
LIMIT 1;

-- 4. List customers that the employee identified in the last question
SELECT DISTINCT CustomerID
FROM Invoices
WHERE EmployeeID = (SELECT EmployeeID FROM (
    SELECT EmployeeID, SUM(Total * 0.15) AS total_commission
    FROM Invoices
    WHERE InvoiceDate BETWEEN '2011-01-01' AND '2012-12-31'
    GROUP BY EmployeeID
    ORDER BY total_commission DESC
    LIMIT 1));

-- 5. Customer with the highest purchase
SELECT CustomerID, SUM(Total) AS total_spent
FROM Invoices
GROUP BY CustomerID
ORDER BY total_spent DESC
LIMIT 1;

-- 6. Look at the customer record of the highest purchase
SELECT *
FROM Customers
WHERE CustomerID = (SELECT CustomerID FROM (
    SELECT CustomerID, SUM(Total) AS total_spent
    FROM Invoices
    GROUP BY CustomerID
    ORDER BY total_spent DESC
    LIMIT 1));
```

7. Conclude the primary person of interest

To conclude the primary person of interest, we need to analyze the results obtained from the previous queries, especially focusing on the following points:

- The employee who made the highest commission.
- The customer with the highest purchase.
- The transactions and details surrounding these individuals to identify any suspicious activity.

Analysis

1. Identify the employee who made the highest commission:

```sql
SELECT EmployeeID, SUM(Total * 0.15) AS total_commission
FROM Invoices
WHERE InvoiceDate BETWEEN '2011-01-01' AND '2012-12-31'
GROUP BY EmployeeID
ORDER BY total_commission DESC
LIMIT 1;
```

2. List customers that this employee served:

```sql
SELECT DISTINCT CustomerID
FROM Invoices
WHERE EmployeeID = (SELECT EmployeeID FROM (
```

```
    SELECT EmployeeID, SUM(Total * 0.15) AS total_commission
    FROM Invoices
    WHERE InvoiceDate BETWEEN '2011-01-01' AND '2012-12-31'
    GROUP BY EmployeeID
    ORDER BY total_commission DESC
    LIMIT 1));
```

3. Identify the customer with the highest purchase:

```
SELECT CustomerID, SUM(Total) AS total_spent
FROM Invoices
GROUP BY CustomerID
ORDER BY total_spent DESC
LIMIT 1;
```

4. Examine the customer record of the highest purchase:

```
SELECT *
FROM Customers
WHERE CustomerID = (SELECT CustomerID FROM (
    SELECT CustomerID, SUM(Total) AS total_spent
    FROM Invoices
    GROUP BY CustomerID
    ORDER BY total_spent DESC
    LIMIT 1));
```

Drawing Conclusions

After running these queries, we will look at the results to identify patterns or anomalies, such as unusually high transactions or frequent interactions between specific employees and customers that could indicate collusion or fraudulent behavior.

```
import sqlite3

# Connect to the SQLite database
conn = sqlite3.connect('/mnt/data/WSDA_Music.db.sql')
cursor = conn.cursor()

# Create the database from the provided SQL file
with open('/mnt/data/WSDA_Music.db.sql', 'r') as f:
    sql_script = f.read()

cursor.executescript(sql_script)

# Queries for each challenge
queries = {
    "Challenge 1.1": """
    SELECT COUNT(*) AS transaction_count
    FROM Invoices
    WHERE InvoiceDate BETWEEN '2011-01-01' AND '2012-12-31';
    """,
    "Challenge 1.2": """
    SELECT SUM(Total) AS total_revenue
    FROM Invoices
    WHERE InvoiceDate BETWEEN '2011-01-01' AND '2012-12-31';
    """,
    "Challenge 2.1": """
    SELECT DISTINCT CustomerID
    FROM Invoices
```

```
    WHERE InvoiceDate BETWEEN '2011-01-01' AND '2012-12-31';
    """,
    "Challenge 2.2": """
    SELECT c.CustomerID, e.EmployeeID, SUM(i.Total) AS total_amount
    FROM Invoices i
    JOIN Customers c ON i.CustomerID = c.CustomerID
    JOIN Employees e ON i.EmployeeID = e.EmployeeID
    WHERE i.InvoiceDate BETWEEN '2011-01-01' AND '2012-12-31'
    GROUP BY c.CustomerID, e.EmployeeID;
    """,
    "Challenge 2.3": """
    WITH avg_transaction AS (
        SELECT AVG(Total) AS average_amount
        FROM Invoices
        WHERE InvoiceDate BETWEEN '2011-01-01' AND '2012-12-31'
    )
    SELECT COUNT(*)
    FROM Invoices
    WHERE Total > (SELECT average_amount FROM avg_transaction)
      AND InvoiceDate BETWEEN '2011-01-01' AND '2012-12-31';
    """,
    "Challenge 2.4": """
    SELECT strftime('%Y', InvoiceDate) AS year, AVG(Total) AS
average_amount
    FROM Invoices
    GROUP BY year;
    """,
    "Challenge 3.1": """
    WITH avg_transaction AS (
        SELECT AVG(Total) AS average_amount
        FROM Invoices
        WHERE InvoiceDate BETWEEN '2011-01-01' AND '2012-12-31'
    )
    SELECT e.EmployeeID, e.LastName, e.FirstName
    FROM Employees e
    JOIN Invoices i ON e.EmployeeID = i.EmployeeID
    WHERE i.InvoiceDate BETWEEN '2011-01-01' AND '2012-12-31'
    GROUP BY e.EmployeeID
    HAVING AVG(i.Total) > (SELECT average_amount FROM avg_transaction);
    """,
    "Challenge 3.2": """
    SELECT i.InvoiceID, i.Total, i.EmployeeID, (i.Total * 0.15) AS
commission_payout
    FROM Invoices i
    WHERE i.InvoiceDate BETWEEN '2011-01-01' AND '2012-12-31';
    """,
    "Challenge 3.3": """
    SELECT EmployeeID, SUM(Total * 0.15) AS total_commission
    FROM Invoices
    WHERE InvoiceDate BETWEEN '2011-01-01' AND '2012-12-31'
    GROUP BY EmployeeID
    ORDER BY total_commission DESC
    LIMIT 1;
    """,
    "Challenge 3.4": """
    SELECT DISTINCT CustomerID
    FROM Invoices
    WHERE EmployeeID = (SELECT EmployeeID FROM (
        SELECT EmployeeID, SUM(Total * 0.15) AS total_commission
        FROM Invoices
        WHERE InvoiceDate BETWEEN '2011-01-01' AND '2012-12-31'
```

```
        GROUP BY EmployeeID
        ORDER BY total_commission DESC
        LIMIT 1));
    """,
    "Challenge 3.5": """
    SELECT CustomerID, SUM(Total) AS total_spent
    FROM Invoices
    GROUP BY CustomerID
    ORDER BY total_spent DESC
    LIMIT 1;
    """,
    "Challenge 3.6": """
    SELECT *
    FROM Customers
    WHERE CustomerID = (SELECT CustomerID FROM (
        SELECT CustomerID, SUM(Total) AS total_spent
        FROM Invoices
        GROUP BY CustomerID
        ORDER BY total_spent DESC
        LIMIT 1));
    """
}

# Execute and print results for each query
for challenge, query in queries.items():
    cursor.execute(query)
    result = cursor.fetchall()
    print(f"{challenge} result: {result}")

# Analyzing the primary person of interest
# 1. Identify the employee who made the highest commission
cursor.execute(queries["Challenge 3.3"])
top_employee = cursor.fetchone()
print(f"Top Employee: {top_employee}")

# 2. List customers served by the top employee
cursor.execute(queries["Challenge 3.4"])
top_employee_customers = cursor.fetchall()
print(f"Top Employee's Customers: {top_employee_customers}")

# 3. Identify the customer with the highest purchase
cursor.execute(queries["Challenge 3.5"])
top_customer = cursor.fetchone()
print(f"Top Customer: {top_customer}")

# 4. Examine the customer record of the highest purchase
cursor.execute(queries["Challenge 3.6"])
top_customer_record = cursor.fetchone()
print(f"Top Customer Record: {top_customer_record}")

# Close the database connection
conn.close()
```

Muhammad Fauzy Syapputra

Results and Conclusion

From the output of these queries, you should focus on:

1. Top Employee: The employee with the highest commission (identified by EmployeeID and their total commission).
2. Top Employee's Customers: Check if the top customer appears in this list to find any direct link.
3. Top Customer: The customer with the highest total purchases (identified by CustomerID and their total spent).
4. Top Customer Record: Detailed information about the top customer to identify any suspicious activity, such as unusual purchase patterns or links to the top employee.

Conclusion:

If the top customer (with the highest purchases) is frequently associated with the top employee (who made the highest commission), and their purchase patterns seem suspicious (e.g., unusually large or frequent transactions), then the primary person of interest could be both the top employee and the top customer due to potential collusion or fraudulent activities.

Based on this data, the primary person of interest would likely be the top employee and potentially the top customer, especially if there are signs of coordinated or anomalous transactions between them.

Final Conclusion

- Top Employee ID and Details: Use the result from query_highest_commission_employee.
- Top Customer ID and Details: Use the result from query_highest_purchase_customer and query_record_of_highest_purchase_customer.

These individuals should be closely investigated for any irregularities or collusive behavior in the transactions between them.

Implementation in Python with SQLite

```
import sqlite3

# Connect to the SQLite database
conn = sqlite3.connect('/mnt/data/WSDA_Music.db.sql')
cursor = conn.cursor()

# Create the database from the provided SQL file
with open('/mnt/data/WSDA_Music.db.sql', 'r') as f:
    sql_script = f.read()

cursor.executescript(sql_script)

# Execute the queries for each challenge
queries = {
    "Challenge 1.1": """
    SELECT COUNT(*) AS transaction_count
    FROM Invoices
    WHERE InvoiceDate BETWEEN '2011-01-01' AND '2012-12-31';
    """,
    "Challenge 1.2": """
    SELECT SUM(Total) AS total_revenue
    FROM Invoices
    WHERE InvoiceDate BETWEEN '2011-01-01' AND '2012-12-31';
    """,
    # Add the rest of the queries here...
}

# Execute and print results for each query
for challenge, query in queries.items():
    cursor.execute(query)
    result = cursor.fetchall()
    print(f"{challenge} result: {result}")

# Close the database connection
conn.close()
```