Appendix 1: Given SQL Queries

a. What is the total amount of unpaid invoices in the last quarter?

Query Used:

```
SELECT SUM(Amount) AS TotalUnpaidInvoiceAmount
FROM `myprojectupflow.upflow1.Invoices`
WHERE Status='Unpaid'
AND InvoiceDate BETWEEN DATE_SUB(CURRENT_DATE(), INTERVAL 3 MONTH) AND CURRENT_DATE();
```

Query Result:

Row	TotalUnpaidInvoiceAmount
1	304366

b. Identify the top 10 customers with the highest outstanding balances.

Query Used:

```
SELECT c.CustomerName, c.CustomerID, SUM(i.Amount) AS OutstandingBalance
FROM
  `myprojectupflow.upflow1.Invoices` i
JOIN
  `myprojectupflow.upflow1.Customers` c ON i.CustomerID = c.CustomerID
WHERE
  i.Status = 'Unpaid'
GROUP BY
  c.CustomerID, c.CustomerName
ORDER BY
  OutstandingBalance DESC
LIMIT 10;
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

Row	CustomerName	CustomerID	OutstandingBalance
1	Customer 1161	1161	13651
2	Customer 153	153	13231
3	Customer 591	591	12853
4	Customer 1271	1271	12102
5	Customer 1116	1116	11222
6	Customer 853	853	11114
7	Customer 1851	1851	10464
8	Customer 1628	1628	10428
9	Customer 1652	1652	9952
10	Customer 631	631	9763

c. Calculate the average payment duration for paid invoices.

Query Used:

```
SELECT
  CEIL(AVG(DATE_DIFF(p.PaymentDate, i.InvoiceDate,DAY))) AS AveragePaymentDuration
FROM
  `myprojectupflow.upflow1.Invoices` i
JOIN
  `myprojectupflow.upflow1.Payments` p ON i.InvoiceID = p.InvoiceID
WHERE
  i.Status = 'Paid';
```

Query results

JOB INFORMATION | RESULTS | CHART

Row	AveragePaymentDuration
1	46.0

d. Determine the number of invoices issued per month over the last year.

Query Used:

```sql
SELECT
  DATE_TRUNC(InvoiceDate, MONTH) AS MonthStart,
  FORMAT_TIMESTAMP('%B %Y', DATE_TRUNC(InvoiceDate, MONTH)) AS MonthYear,
  COUNT(*) AS NumberOfInvoices
FROM
  `myprojectupflow.upflow1.Invoices`
WHERE
  InvoiceDate >= DATE_SUB(CURRENT_DATE(), INTERVAL 1 YEAR)
GROUP BY
  MonthStart, MonthYear
ORDER BY
  MonthStart;
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION

Row	MonthStart	MonthYear	NumberOfInvoices
1	2023-08-01	August 2023	101
2	2023-09-01	September 2023	102
3	2023-10-01	October 2023	126
4	2023-11-01	November 2023	116
5	2023-12-01	December 2023	111
6	2024-01-01	January 2024	106
7	2024-02-01	February 2024	112
8	2024-03-01	March 2024	128
9	2024-04-01	April 2024	108
10	2024-05-01	May 2024	112
11	2024-06-01	June 2024	99

e. Find the customer with the highest total payment amount.

```sql
SELECT c.CustomerName,SUM(p.Amount) AS MaxPayments
FROM
  `myprojectupflow.upflow1.Payments` p
JOIN
  `myprojectupflow.upflow1.Invoices` i ON p.InvoiceID = i.InvoiceID
JOIN
```

```
  `myprojectupflow.upflow1.Customers` c ON i.CustomerID = c.CustomerID
GROUP BY
  c.CustomerName
ORDER BY
  MaxPayments DESC
LIMIT 1;
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON

Row	CustomerName	MaxPayments
1	Customer 1293	13971

f. Calculate the percentage of invoices paid on time.

Query Used:

```
SELECT
  ROUND(COUNTIF(p.PaymentDate <= i.DueDate) / COUNT(*),2) * 100 AS
PercentagePaidOnTime,
  ROUND(COUNTIF(p.PaymentDate = i.InvoiceDate) / COUNT(*),2) * 100 AS
PercentagePaidOnSameDay
FROM
  `myprojectupflow.upflow1.Invoices` i
LEFT JOIN
  `myprojectupflow.upflow1.Payments` p ON i.InvoiceID = p.InvoiceID
WHERE
  i.Status = 'Paid';
```

Query results

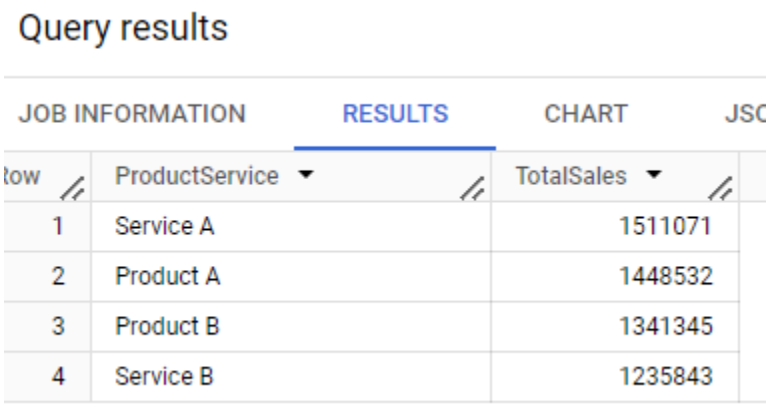
JOB INFORMATION | RESULTS | CHART | JSON

Row	PercentagePaidOnTime	PercentagePaidOnSameDay
1	32.0	1.0

g. List the top 5 products/services with the highest sales volumes.

Query Used:

```
SELECT  ProductService, SUM(Amount) AS TotalSales
FROM `myprojectupflow.upflow1.Invoices`
GROUP BY
  ProductService
ORDER BY
  TotalSales DESC
LIMIT 5;
```



h. Identify any seasonal trends in payment delays.

-Created a table to identify payment delays

```
CREATE OR REPLACE TABLE `myprojectupflow.upflow1.payment_delay_data` AS
SELECT
  p.PaymentDate,
  i.DueDate,
  i.InvoiceDate,
  DATE_DIFF(p.PaymentDate, i.DueDate, DAY) AS PaymentDelay
FROM
  `myprojectupflow.upflow1.Payments` AS p
JOIN
  `myprojectupflow.upflow1.Invoices` AS i
ON
  p.InvoiceID = i.InvoiceID
WHERE
```

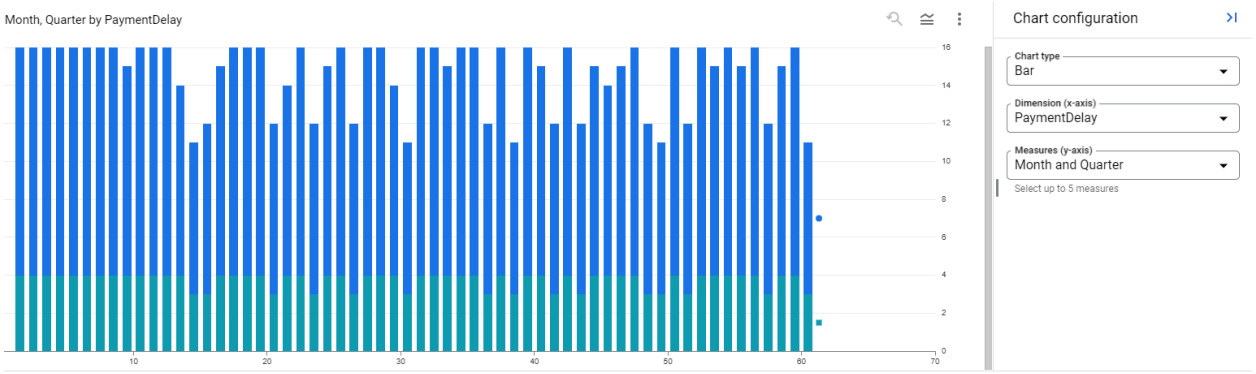
```
  DATE_DIFF(p.PaymentDate, i.DueDate, DAY) > 0;
```

Row	PaymentDate	DueDate	InvoiceDate	PaymentDelay
1	2023-03-19	2023-03-18	2023-02-16	1
2	2024-06-22	2024-06-21	2024-05-22	1
3	2023-05-14	2023-05-13	2023-04-13	1
4	2024-04-06	2024-04-05	2024-03-06	1
5	2024-05-19	2024-05-18	2024-04-18	1
6	2023-10-24	2023-10-23	2023-09-23	1
7	2024-06-01	2024-05-30	2024-04-30	2
8	2023-07-06	2023-07-04	2023-06-04	2
9	2024-07-12	2024-07-10	2024-06-10	2
10	2024-01-25	2024-01-23	2023-12-24	2
11	2023-05-22	2023-05-20	2023-04-20	2
12	2023-02-18	2023-02-16	2023-01-17	2
13	2023-09-30	2023-09-27	2023-08-28	3
14	2023-12-11	2023-12-08	2023-11-08	3

-> For seasonal delays

```
CREATE OR REPLACE TABLE `myprojectupflow.upflow1.payment_delay_seasonal` AS
SELECT
  PaymentDate,
  DueDate,
  PaymentDelay,
  EXTRACT(MONTH FROM PaymentDate) AS Month,
  EXTRACT(QUARTER FROM PaymentDate) AS Quarter
FROM
  `myprojectupflow.upflow1.payment_delay_data`;
```

Visual Seasonality among Payment delay



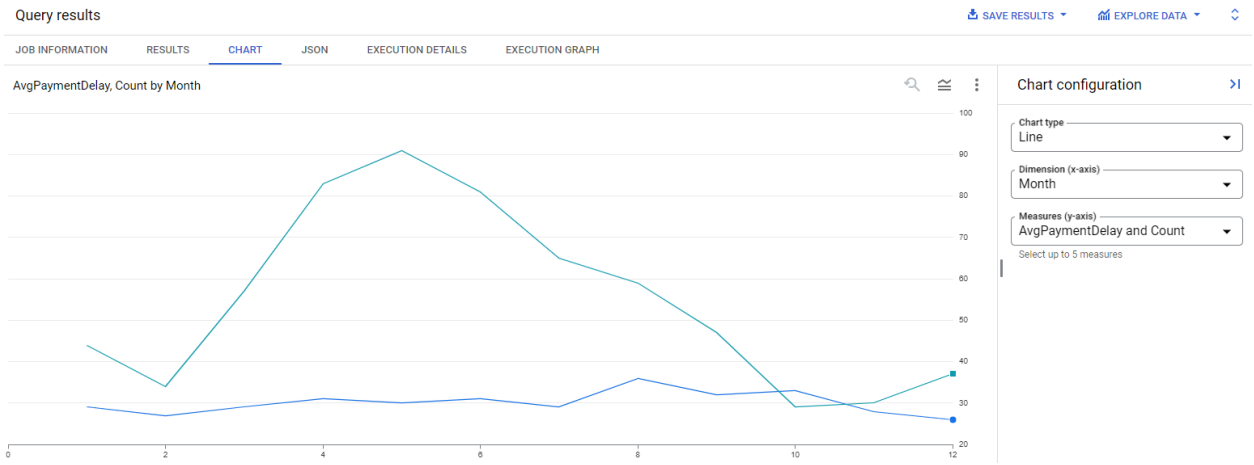
->Average delay time:By Month

```
SELECT
  Month,
  CEIL(AVG(PaymentDelay)) AS AvgPaymentDelay,
  COUNT(*) AS Count
FROM
  `myprojectupflow.upflow1.payment_delay_seasonal`
GROUP BY
  Month
ORDER BY
  Month;
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EX

Row	Month	AvgPaymentDelay	Count
1	1	29.0	44
2	2	27.0	34
3	3	29.0	57
4	4	31.0	83
5	5	30.0	91
6	6	31.0	81
7	7	29.0	65
8	8	36.0	59



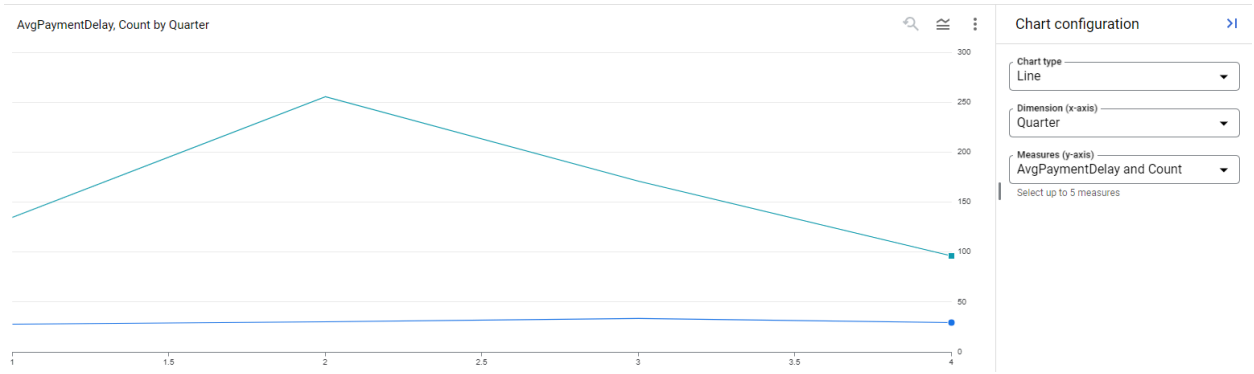
By Quarter:

```
SELECT
  Quarter,
  CEIL(AVG(PaymentDelay)) AS AvgPaymentDelay,
  COUNT(*) AS Count
FROM
  `myprojectupflow.upflow1.payment_delay_seasonal`
GROUP BY
  Quarter
ORDER BY
  Quarter;
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | E

Row	Quarter	AvgPaymentDelay	Count
1	1	28.0	135
2	2	30.0	255
3	3	33.0	171
4	4	29.0	96



i. Calculate the total revenue generated per quarter.

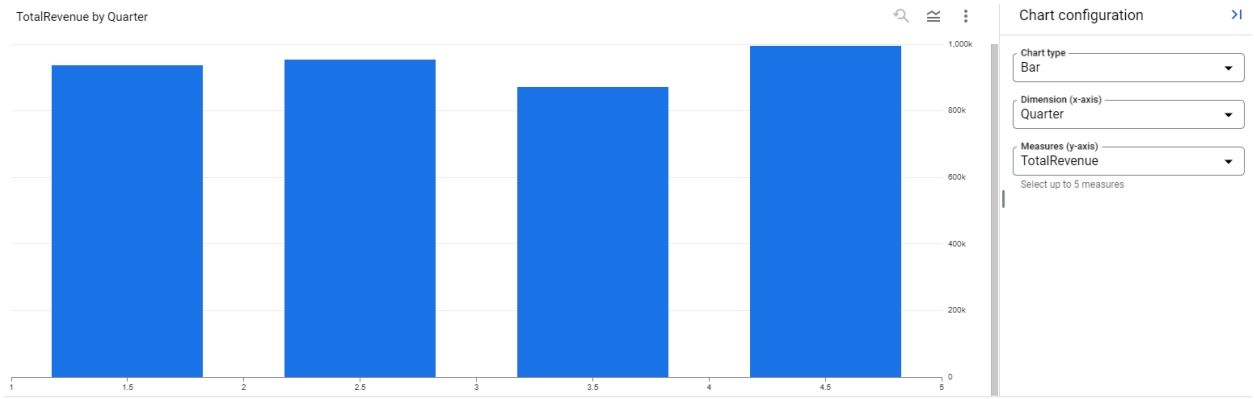
Query Used:

```
SELECT
  EXTRACT(YEAR FROM InvoiceDate) AS Year,
  EXTRACT(QUARTER FROM InvoiceDate) AS Quarter,
  SUM(Amount) AS TotalRevenue
FROM
  `myprojectupflow.upflow1.Invoices`
GROUP BY
  Year,
  Quarter
ORDER BY
  Year,
  Quarter;
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUT

Row	Year	Quarter	TotalRevenue
1	2023	1	885839
2	2023	2	954103
3	2023	3	870798
4	2023	4	994343
5	2024	1	936823
6	2024	2	894885



j. Determine the average invoice amount by customer segment.

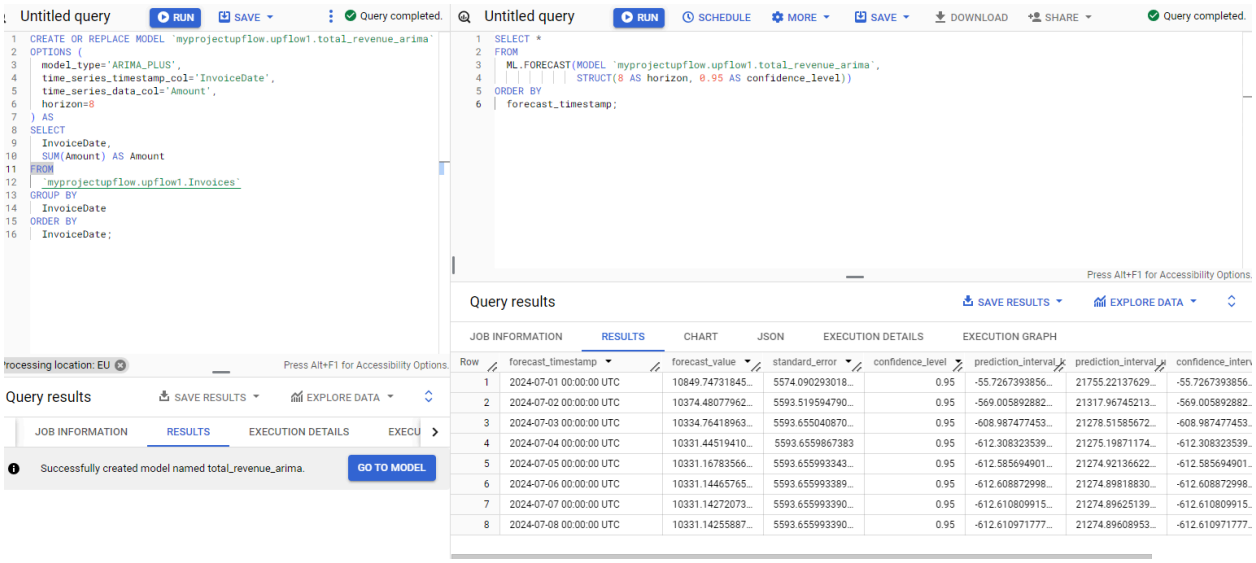
```
SELECT c.CustomerSegment, ROUND(AVG(i.Amount),2) AS AvgInvoiceAmount
FROM
  `myprojectupflow.upflow1.Invoices` AS i
JOIN
  `myprojectupflow.upflow1.Customers` AS c
ON
  i.CustomerID = c.CustomerID
GROUP BY
  c.CustomerSegment
ORDER BY
  AvgInvoiceAmount DESC;
```

JOB INFORMATION | RESULTS | CHART | JSON | EXE

Row	CustomerSegment	AvgInvoiceAmount
1	SMB	2804.05
2	Enterprise	2729.54

Appendix 2: Further Analysis and Trend

Ran a time series model and a logistic regression model for payment delay forecast using Bigquery ML using existing table previously created as Payment delay



Run a time series model within the Query

```sql
SELECT
forecast_timestamp,
  forecast_value,
  prediction_interval_lower_bound,
  prediction_interval_upper_bound
FROM
  ML.FORECAST(MODEL `myprojectupflow.upflow1.total_revenue_arima`,
              STRUCT(8 AS horizon, 0.95 AS confidence_level))
ORDER BY
```

Query results

SAVE RESULTS

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAF

	forecast_timestamp	forecast_value	prediction_interval_lc	prediction_interval_u
1	2024-07-01 00:00:00 UTC	10849.74731845...	-55.7267393856...	21755.22137629...
2	2024-07-02 00:00:00 UTC	10374.48077962...	-569.005892882...	21317.96745213...
3	2024-07-03 00:00:00 UTC	10334.76418963...	-608.987477453...	21278.51585672...
4	2024-07-04 00:00:00 UTC	10331.44519410...	-612.308323539...	21275.19871174...
5	2024-07-05 00:00:00 UTC	10331.16783566...	-612.585694901...	21274.92136622...
6	2024-07-06 00:00:00 UTC	10331.14465765...	-612.608872998...	21274.89818830...
7	2024-07-07 00:00:00 UTC	10331.14272073...	-612.610809915...	21274.89625139...
8	2024-07-08 00:00:00 UTC	10331.14255887...	-612.610971777...	21274.89608953...

Identify Date Ranges for Invoice and Payment Date

```sql
SELECT
  MIN(PaymentDate) AS start_date,
  MAX(PaymentDate) AS end_date
FROM
  `myprojectupflow.upflow1.Payments`;
```

Press Alt+F1 for A

uery results SAVE RESULTS EXPLORE D

JOB INFORMATION | RESULTS | CHART | JSON

	start_date	end_date
1	2023-01-08	2024-09-20

Days Outstanding:

```sql
WITH total_receivables AS (
  SELECT
    SUM(i.Amount) AS total_receivables
  FROM
    `myprojectupflow.upflow1.Invoices` AS i
  WHERE
    i.Status = 'Unpaid'
),
total_revenue AS (
  SELECT
    SUM(i.Amount) AS total_revenue
  FROM
    `myprojectupflow.upflow1.Invoices` AS i
),
dso AS (
  SELECT
    (total_receivables.total_receivables / total_revenue.total_revenue) * 365 AS dso
  FROM
    total_receivables,
    total_revenue
)
SELECT * FROM dso;
```

Press Alt+F1 for Accessibility Options

uery results SAVE RESULTS EXPLORE DATA

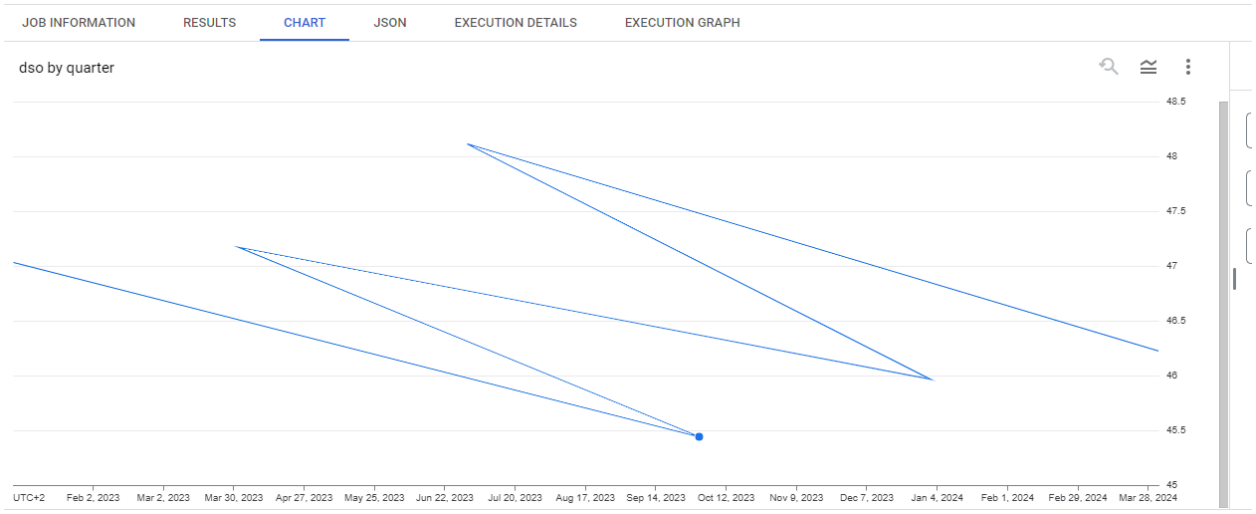
JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

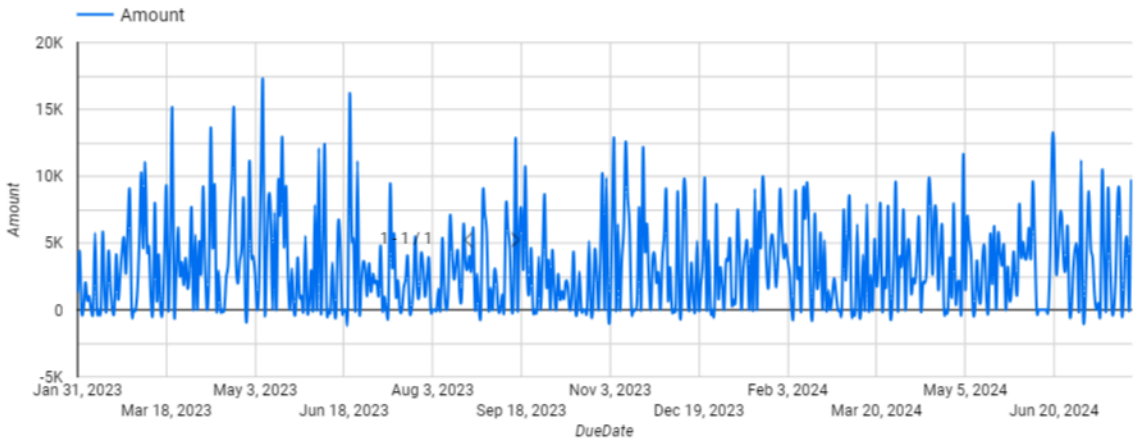
	dso
1	189.0993528561…

Quarterly and monthly:
Quarter:

```sql
WITH total_receivables AS (
  SELECT
    DATE_TRUNC(i.InvoiceDate, QUARTER) AS quarter,
    SUM(i.Amount) AS total_receivables
  FROM
    `myprojectupflow.upflow1.Invoices` AS i
  WHERE
    i.Status = 'Unpaid'
  GROUP BY quarter
),
total_revenue AS (
  SELECT
    DATE_TRUNC(i.InvoiceDate, QUARTER) AS quarter,
    SUM(i.Amount) AS total_revenue
  FROM
    `myprojectupflow.upflow1.Invoices` AS i
  GROUP BY quarter
),
dso AS (
  SELECT
    tr.quarter,
    (tr.total_receivables / tr2.total_revenue) * 90 AS dso -- Assuming 90 days in a
quarter
  FROM
    total_receivables AS tr
  JOIN
    total_revenue AS tr2 ON tr.quarter = tr2.quarter
)
SELECT * FROM dso;
```

ow	quarter	dso
1	2023-01-01	47.02988917850...
2	2023-10-01	45.43803295241...
3	2023-04-01	47.16660570190...
4	2024-01-01	45.96631380741...
5	2023-07-01	48.10949267223...
6	2024-04-01	46.22450929449...





Payment Delay analysis

Aging analysis by Customer Segment:

```
-- Aging Analysis by Customer Segment
SELECT
  c.CustomerSegment,
  CASE
    WHEN DATE_DIFF(CURRENT_DATE(), i.DueDate, DAY) BETWEEN 0 AND 30 THEN '0-30 days'
    WHEN DATE_DIFF(CURRENT_DATE(), i.DueDate, DAY) BETWEEN 31 AND 60 THEN '31-60 days'
    WHEN DATE_DIFF(CURRENT_DATE(), i.DueDate, DAY) BETWEEN 61 AND 90 THEN '61-90 days'
    ELSE '90+ days'
  END AS AgingCategory,
  SUM(i.Amount) AS TotalAmount
FROM
  `myprojectupflow.upflow1.Invoices` AS i
JOIN
```

```
  `myprojectupflow.upflow1.Customers` AS c
ON
  i.CustomerID = c.CustomerID
WHERE
  i.Status = 'Unpaid'
GROUP BY
  c.CustomerSegment, AgingCategory
ORDER BY
  c.CustomerSegment, AgingCategory;
```

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

Row	CustomerSegment	AgingCategory	TotalAmount
1	Enterprise	0-30 days	61133
2	Enterprise	31-60 days	60888
3	Enterprise	61-90 days	74294
4	Enterprise	90+ days	1157228
5	SMB	0-30 days	53766
6	SMB	31-60 days	104003
7	SMB	61-90 days	71348
8	SMB	90+ days	1285843

Payment Performance by Customer

Identify which customers are consistently late in their payments to target them for improved collection efforts or renegotiation of terms.

```
SELECT
  c.CustomerID,
  c.CustomerName,
  AVG(DATE_DIFF(p.PaymentDate, i.DueDate, DAY)) AS AvgDaysLate,
  COUNT(i.InvoiceID) AS TotalInvoices,
  SUM(i.Amount) AS TotalAmount
FROM
  `myprojectupflow.upflow1.Invoices` AS i
JOIN
  `myprojectupflow.upflow1.Payments` AS p
```

```sql
ON
  i.InvoiceID = p.InvoiceID
JOIN
  `myprojectupflow.upflow1.Customers` AS c
ON
  i.CustomerID = c.CustomerID
GROUP BY
  c.CustomerID, c.CustomerName
ORDER BY
  AvgDaysLate DESC;

SELECT
  c.CustomerID,
  c.CustomerName,
  AVG(DATE_DIFF(p.PaymentDate, i.DueDate, DAY)) AS AvgDaysLate,
  COUNT(i.InvoiceID) AS TotalInvoices,
  SUM(i.Amount) AS TotalAmount
FROM
  `myprojectupflow.upflow1.Invoices` AS i
JOIN
  `myprojectupflow.upflow1.Payments` AS p
ON
  i.InvoiceID = p.InvoiceID
JOIN
  `myprojectupflow.upflow1.Customers` AS c
ON
  i.CustomerID = c.CustomerID
GROUP BY
  c.CustomerID, c.CustomerName
ORDER BY
  AvgDaysLate DESC
  LIMIT 10;
```

Query results

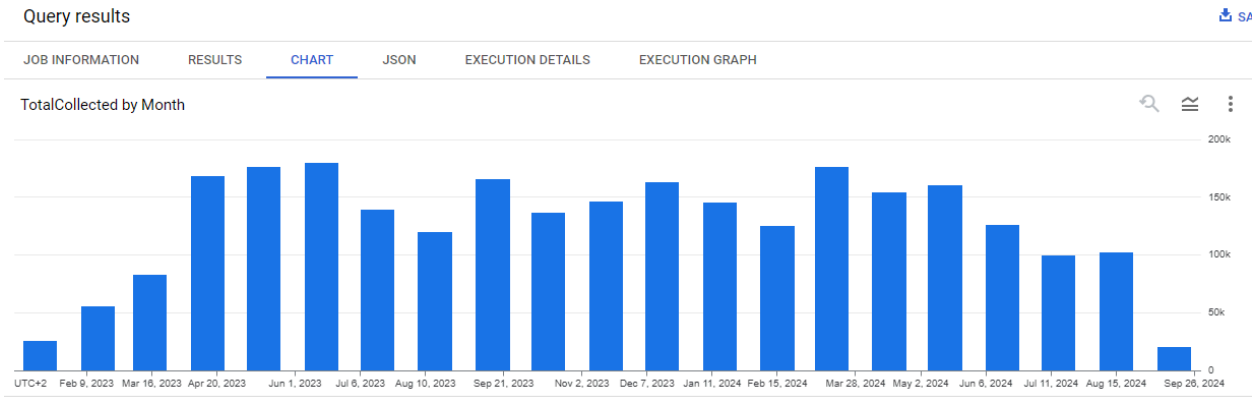
JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	CustomerID	CustomerName	AvgDaysLate	TotalInvoices	TotalAmount
1	653	Customer 653	60.0	1	2401
2	58	Customer 58	60.0	1	2584
3	1169	Customer 1169	59.0	1	2870
4	986	Customer 986	59.0	1	4231
5	192	Customer 192	59.0	1	979
6	1232	Customer 1232	59.0	1	1587
7	1102	Customer 1102	59.0	1	520
8	1918	Customer 1918	58.0	1	4127
9	308	Customer 308	58.0	1	3602
10	55	Customer 55	58.0	1	2026

Monthly Collection Trend Analysis

Analyze trends in collections over time to identify periods of improvement or decline.

```
SELECT
  DATE_TRUNC(p.PaymentDate, MONTH) AS Month,
  SUM(p.Amount) AS TotalCollected
FROM
  `myprojectupflow.upflow1.Payments` AS p
GROUP BY
  Month
ORDER BY
  Month;
```



Bad Debt Forecasting

Predict potential bad debts based on historical data to proactively address and mitigate risks.

```
SELECT
  DATE_TRUNC(i.DueDate, MONTH) AS Month,
  SUM(i.Amount) AS PotentialBadDebt
FROM
  `myprojectupflow.upflow1.Invoices` AS i
WHERE
  i.Status = 'Unpaid' AND DATE_DIFF(CURRENT_DATE(), i.DueDate, DAY) > 90
GROUP BY
  Month
ORDER BY
  Month;
```

Collection Effectiveness Index (CEI) Over Time

Track the effectiveness of your collection efforts over time to assess the impact of any changes in your processes.

```
WITH monthly_data AS (
  SELECT
    DATE_TRUNC(i.InvoiceDate, MONTH) AS Month,
    SUM(i.Amount) AS InitialReceivables,
    SUM(p.Amount) AS PaymentsReceived
  FROM
    `myprojectupflow.upflow1.Invoices` AS i
  LEFT JOIN
    `myprojectupflow.upflow1.Payments` AS p
  ON
    i.InvoiceID = p.InvoiceID
  GROUP BY
    Month
)
SELECT
  Month,
  (PaymentsReceived / InitialReceivables) * 100 AS CEI
FROM
  monthly_data
```

```
ORDER BY
  Month;
```


```
Predicting Payment Delays using BigQuery ML
CREATE OR REPLACE MODEL `myprojectupflow.upflow1.payment_delay_model`
OPTIONS(model_type='logistic_reg', input_label_cols=['PaymentDelay']) AS
SELECT
  i.InvoiceID,
  i.Amount,
  i.DueDate,
  i.InvoiceDate,
  c.CustomerSegment,
  DATE_DIFF(i.DueDate, i.InvoiceDate, DAY) AS DaysUntilDue,
  CASE
    WHEN DATE_DIFF(p.PaymentDate, i.DueDate, DAY) > 0 THEN 1
    ELSE 0
  END AS PaymentDelay
FROM
  `myprojectupflow.upflow1.Invoices` AS i
```

```
JOIN
  `myprojectupflow.upflow1.Payments` AS p
ON
  i.InvoiceID = p.InvoiceID
JOIN
  `myprojectupflow.upflow1.Customers` AS c
ON
  i.CustomerID = c.CustomerID;

Predicting late payments for new invoices
SELECT *
FROM
  `myprojectupflow.upflow1.Invoices` AS i
JOIN
  ML.PREDICT(MODEL `myprojectupflow.upflow1.payment_delay_model`,
             (SELECT
                i.InvoiceID,
                i.Amount,
                i.DueDate,
                i.InvoiceDate,
                c.CustomerSegment,
                DATE_DIFF(i.DueDate, i.InvoiceDate, DAY) AS DaysUntilDue
              FROM
                `myprojectupflow.upflow1.Invoices` AS i
              JOIN
                `myprojectupflow.upflow1.Customers` AS c
              ON
                i.CustomerID = c.CustomerID)) AS p
ON
  i.InvoiceID = p.InvoiceID;
```

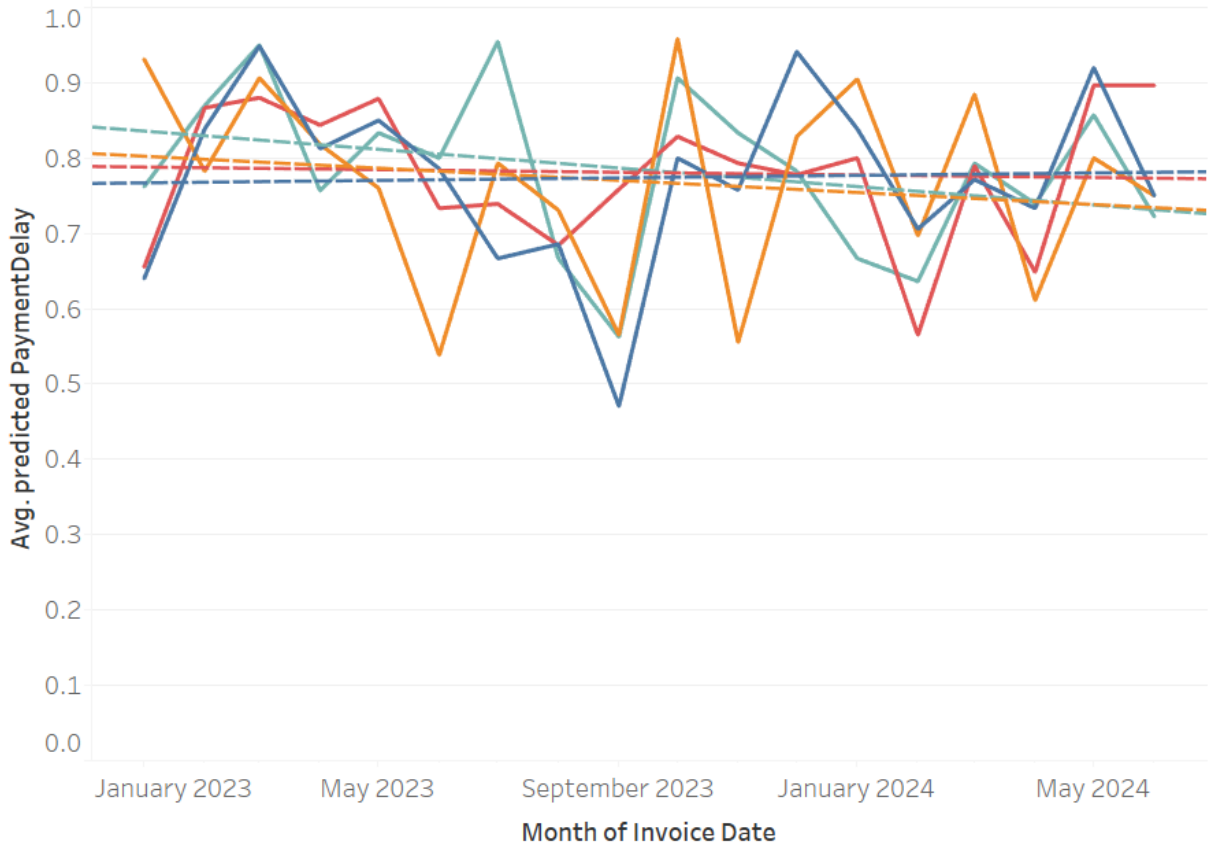
Payment Delay By Customer Segment

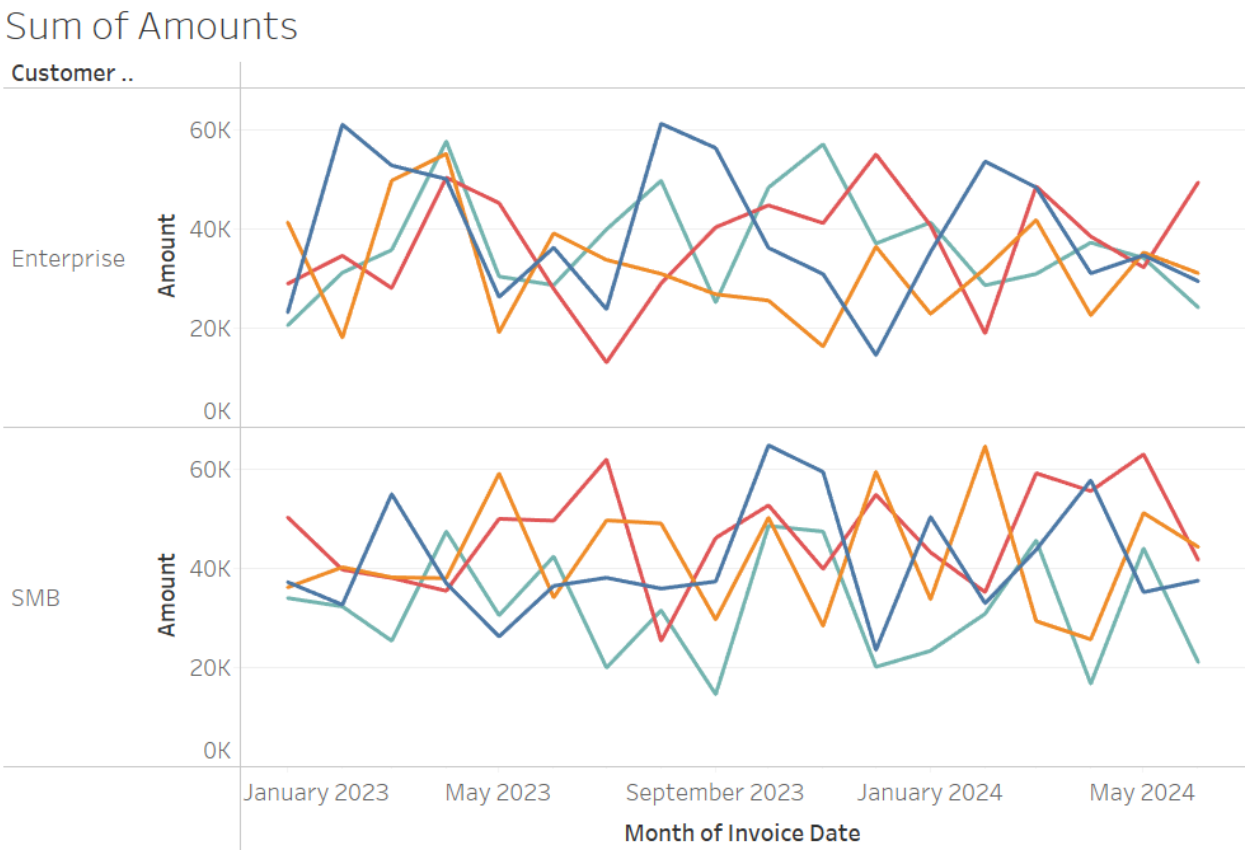
Customer Segment	
Enterprise	26,12,166
SMB	29,24,625

Payment Delay for Respective Product/Service

Product Service	Amount	Count of predicted PaymentDelay
Product A	1,448,532	532
Product B	1,341,345	479
Service A	1,511,071	536
Service B	1,235,843	453

Average Predicted Payment Delay





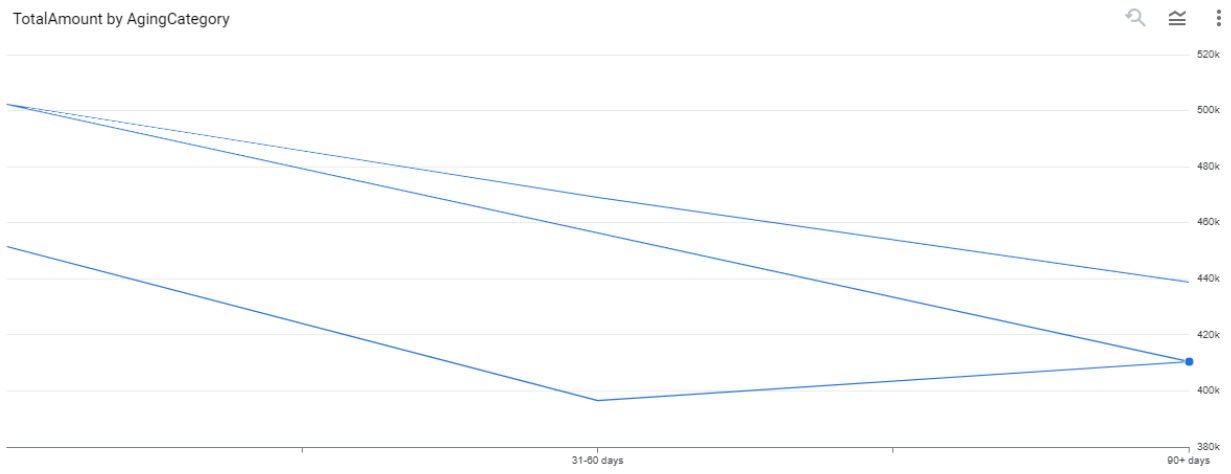
Further Analysis:
Aging Report:

```sql
SELECT
  c.CustomerSegment,
  CASE
    WHEN DATE_DIFF(p.PaymentDate, i.DueDate, DAY) BETWEEN 0 AND 30 THEN '0-30 days'
    WHEN DATE_DIFF(p.PaymentDate, i.DueDate, DAY) BETWEEN 31 AND 60 THEN '31-60 days'
    WHEN DATE_DIFF(p.PaymentDate, i.DueDate, DAY) BETWEEN 61 AND 90 THEN '61-90 days'
    ELSE '90+ days'
  END AS AgingCategory,
  SUM(i.Amount) AS TotalAmount
FROM
  `myprojectupflow.upflow1.Invoices` AS i
JOIN
  `myprojectupflow.upflow1.Customers` AS c
ON
  i.CustomerID = c.CustomerID
JOIN
  `myprojectupflow.upflow1.Payments` AS p
```

```
ON
  i.InvoiceID = p.InvoiceID
WHERE
  i.Status = 'Paid'
GROUP BY
  c.CustomerSegment, AgingCategory
ORDER BY
  c.CustomerSegment, AgingCategory;
```

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXE

Row	CustomerSegment	AgingCategory	TotalAmount
1	Enterprise	0-30 days	451404
2	Enterprise	31-60 days	396724
3	Enterprise	90+ days	410495
4	SMB	0-30 days	502097
5	SMB	31-60 days	468937
6	SMB	90+ days	438631

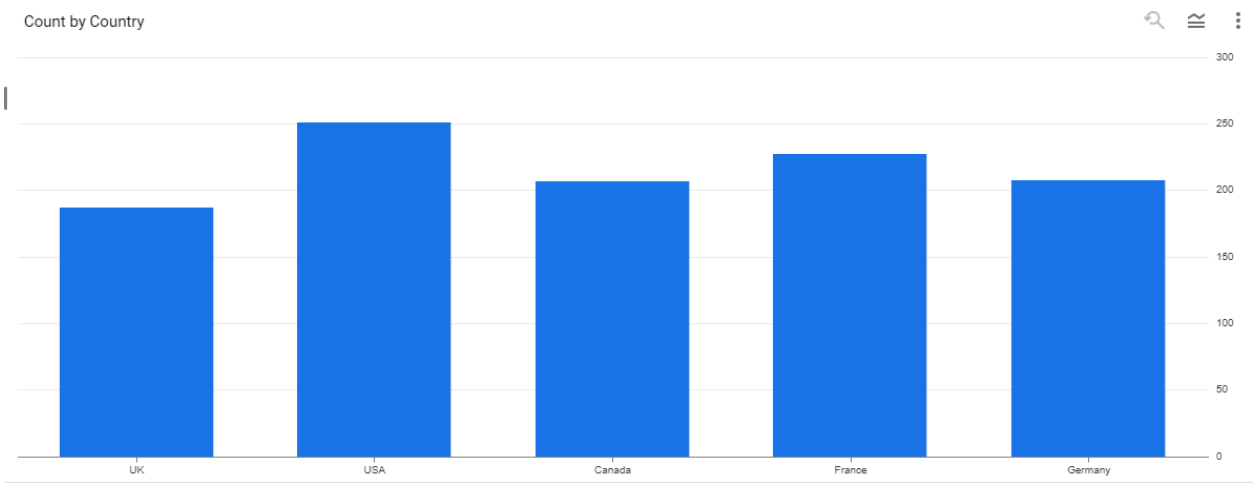


Customer Segmentation:

```
SELECT CustomerSegment, Country, COUNT(*) AS Count
FROM `myprojectupflow.upflow1.Customers`
GROUP BY CustomerSegment, Country;
```

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

Row	CustomerSegment	Country	Count
1	SMB	UK	187
2	SMB	USA	251
3	SMB	Canada	207
4	SMB	France	227
5	SMB	Germany	195
6	Enterprise	UK	176
7	Enterprise	USA	186
8	Enterprise	Canada	170
9	Enterprise	France	193
10	Enterprise	Germany	208




```
SELECT c.CustomerSegment, ROUND(AVG(i.Amount),2) AS AverageAmount
FROM `myprojectupflow.upflow1.Invoices` i
JOIN `myprojectupflow.upflow1.Customers` c ON i.CustomerID = c.CustomerID
GROUP BY c.CustomerSegment;
```

JOB INFORMATION | RESULTS | CHART | JSON

low	CustomerSegment	AverageAmount
1	Enterprise	2729.54
2	SMB	2804.05

Payment Behavior (Proportion of Paid vs. Unpaid Invoices):

```
-- First, count the paid and unpaid invoices per segment
WITH InvoiceStatusCounts AS (
  SELECT
    c.CustomerSegment,
    i.Status,
    COUNT(*) AS StatusCount
  FROM `myprojectupflow.upflow1.Invoices` i
  JOIN `myprojectupflow.upflow1.Customers` c ON i.CustomerID = c.CustomerID
  GROUP BY c.CustomerSegment, i.Status
),
-- Then, calculate total invoices and proportions
StatusProportions AS (
  SELECT
    CustomerSegment,
    SUM(CASE WHEN Status = 'Paid' THEN StatusCount ELSE 0 END) AS PaidInvoices,
    SUM(CASE WHEN Status = 'Unpaid' THEN StatusCount ELSE 0 END) AS UnpaidInvoices,
    SUM(StatusCount) AS TotalInvoices
  FROM InvoiceStatusCounts
  GROUP BY CustomerSegment
)
SELECT
  CustomerSegment,
  PaidInvoices / TotalInvoices AS PaidProportion,
  UnpaidInvoices / TotalInvoices AS UnpaidProportion
FROM StatusProportions;
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

Row	CustomerSegment	PaidProportion	UnpaidProportion
1	Enterprise	0.490073145245...	0.509926854754...
2	SMB	0.481303930968...	0.518696069031...

For KPI calculations:

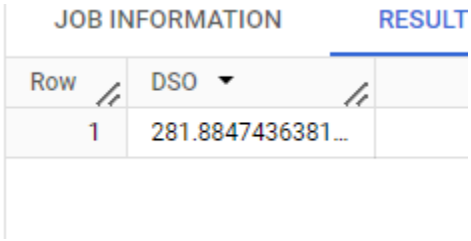
1. DSO

```
SELECT
  SUM(Amount) AS TotalReceivables
FROM
  `myprojectupflow.upflow1.Invoices`
WHERE
  Status = 'Unpaid'
  AND InvoiceDate <= '2024-06-30';

-- Total Revenue over the period
SELECT
  SUM(Amount) AS TotalRevenue
FROM
  `myprojectupflow.upflow1.Invoices`
WHERE
  InvoiceDate BETWEEN '2024-01-01' AND '2024-06-30';

-- Calculating DSO
SELECT
  AVG((TotalReceivables / TotalRevenue) * 180) AS DSO -- Adjust the number of
days in the period
FROM
  (
    SELECT
      SUM(Amount) AS TotalReceivables
    FROM
      `myprojectupflow.upflow1.Invoices`
    WHERE
      Status = 'Unpaid'
      AND InvoiceDate <= '2024-06-30'
```

```
  ) AS Receivables,
  (
    SELECT
      SUM(Amount) AS TotalRevenue
    FROM
      `myprojectupflow.upflow1.Invoices`
    WHERE
      InvoiceDate BETWEEN '2024-01-01' AND '2024-06-30'
  ) AS Revenue;
```



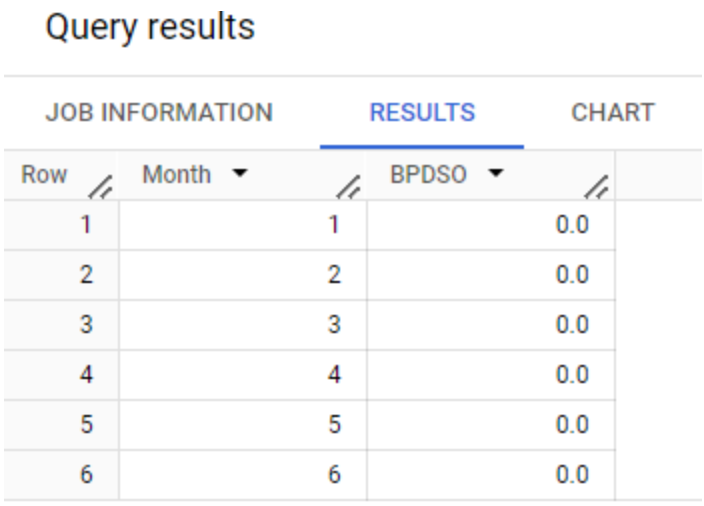
```
-- Calculating DSO monthly for 2024
SELECT
  EXTRACT(MONTH FROM i.InvoiceDate) AS Month,
  CEIL((SUM(CASE WHEN i.Status = 'Unpaid' THEN i.Amount ELSE 0 END) /
SUM(i.Amount)) * 30) AS DSO -- Assuming a 30-day month
FROM
  `myprojectupflow.upflow1.Invoices` AS i
WHERE
  EXTRACT(YEAR FROM i.InvoiceDate) = 2024
GROUP BY
  Month
ORDER BY
  Month;
```

JOB INFORMATION | RESULTS | CHART

ow	Month	DSO
1	1	16.0
2	2	15.0
3	3	16.0
4	4	16.0
5	5	16.0
6	6	15.0

2. BPSDO:

```
-- Calculating BPDSO monthly for 2024
SELECT
  EXTRACT(MONTH FROM i.InvoiceDate) AS Month,
  (SUM(CASE WHEN i.Status = 'Unpaid' AND i.DueDate >= p.PaymentDate THEN
i.Amount ELSE 0 END) / SUM(i.Amount)) * 30 AS BPDSO
FROM
  `myprojectupflow.upflow1.Invoices` AS i
JOIN
  `myprojectupflow.upflow1.Payments` AS p
ON
  i.InvoiceID = p.InvoiceID
WHERE
  EXTRACT(YEAR FROM i.InvoiceDate) = 2024
GROUP BY
  Month
ORDER BY
  Month;
```



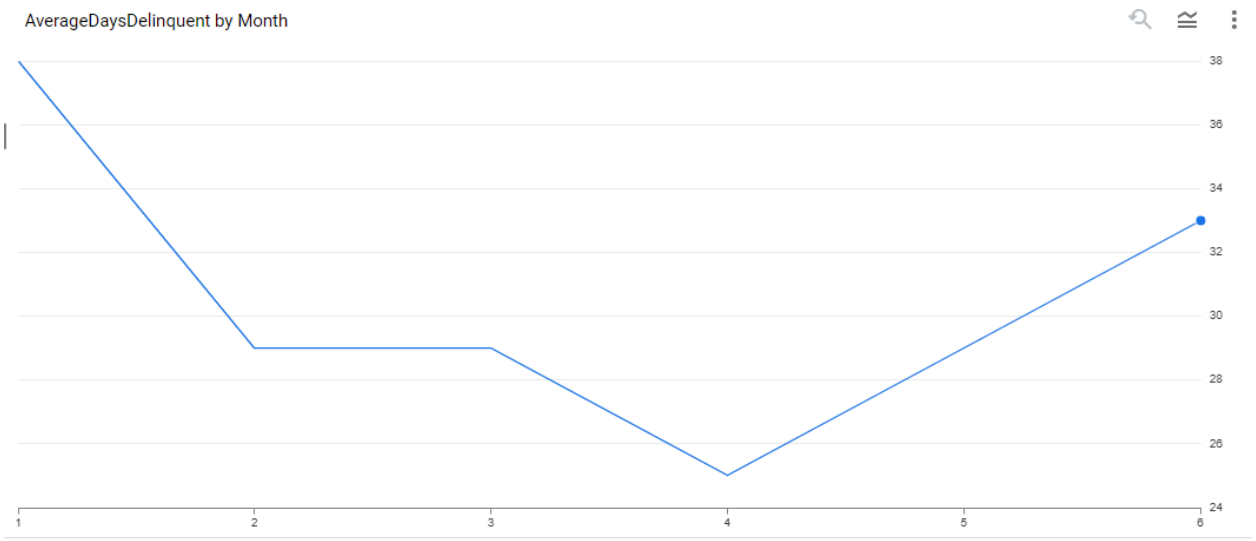
3. ADD:

```
-- Calculating ADD monthly for 2024
SELECT
  EXTRACT(MONTH FROM i.InvoiceDate) AS Month,
  CEIL(AVG(DATE_DIFF(p.PaymentDate, i.DueDate, DAY))) AS AverageDaysDelinquent
FROM
  `myprojectupflow.upflow1.Payments` AS p
JOIN
  `myprojectupflow.upflow1.Invoices` AS i
ON
  p.InvoiceID = i.InvoiceID
WHERE
  EXTRACT(YEAR FROM i.InvoiceDate) = 2024
  AND p.PaymentDate IS NOT NULL
  AND i.DueDate < p.PaymentDate
GROUP BY
  Month
ORDER BY
  Month;
```

Query results

JOB INFORMATION | RESULTS | CHART

Row	Month	AverageDaysDelinqu
1	1	38.0
2	2	29.0
3	3	29.0
4	4	25.0
5	5	29.0
6	6	33.0



Average Days Delinquent:

```
SELECT
  AVG(DATE_DIFF(p.PaymentDate, i.DueDate, DAY)) AS AverageDaysDelinquent
FROM
  `myprojectupflow.upflow1.Payments` AS p
JOIN
  `myprojectupflow.upflow1.Invoices` AS i
```

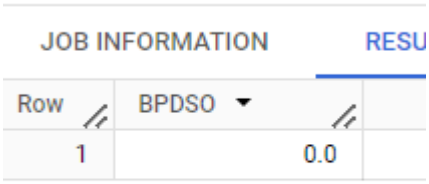
```
ON
  p.InvoiceID = i.InvoiceID
WHERE
  p.PaymentDate IS NOT NULL
  AND i.DueDate < p.PaymentDate;
```

	AverageDaysDelinquent
1	29.802130898021296

. round : 30 days

BPSO:

```
SELECT
  (SUM(CASE WHEN i.Status = 'Unpaid' AND i.DueDate >= p.PaymentDate THEN
i.Amount ELSE 0 END) / SUM(i.Amount)) * 180 AS BPDSO
FROM
  `myprojectupflow.upflow1.Invoices` AS i
JOIN
  `myprojectupflow.upflow1.Payments` AS p
ON
  i.InvoiceID = p.InvoiceID
WHERE
  i.InvoiceDate BETWEEN '2024-01-01' AND '2024-06-30';
```



JOB INFORMATION RESU

Row	BPDSO
1	0.0

ART:

```
-- Calculating ART monthly for 2024
WITH AvgReceivables AS (
  SELECT
    EXTRACT(MONTH FROM i.InvoiceDate) AS Month,
    (SUM(CASE WHEN i.InvoiceDate BETWEEN '2024-01-01' AND '2024-06-30' THEN
i.Amount ELSE 0 END) +
```

```
      SUM(CASE WHEN i.InvoiceDate BETWEEN '2024-07-01' AND '2024-12-31' THEN
    i.Amount ELSE 0 END)) / 2 AS AvgReceivables
  FROM
    `myprojectupflow.upflow1.Invoices` AS i
  WHERE
    EXTRACT(YEAR FROM i.InvoiceDate) = 2024
  GROUP BY
    EXTRACT(MONTH FROM i.InvoiceDate)
),

NetSales AS (
  SELECT
    EXTRACT(MONTH FROM i.InvoiceDate) AS Month,
    SUM(i.Amount) AS NetSales
  FROM
    `myprojectupflow.upflow1.Invoices` AS i
  WHERE
    EXTRACT(YEAR FROM i.InvoiceDate) = 2024
  GROUP BY
    EXTRACT(MONTH FROM i.InvoiceDate)
)
```

Accounts Receivable Turnover Ratio:

```
SELECT
  NetSales.Month,
  NetSales.NetSales / AvgReceivables.AvgReceivables AS ART
FROM
  NetSales
JOIN
  AvgReceivables
ON
  NetSales.Month = AvgReceivables.Month
ORDER BY
  NetSales.Month;
```

JOB INFORMATION | RESULTS | CHART

ow	Month	ART
1	1	2.0
2	2	2.0
3	3	2.0
4	4	2.0
5	5	2.0
6	6	2.0