Appendix

Queries

Data Cleaning

Mobile_Game_Inapp_Purchases Dataset

```sql
# Check nulls per column
SELECT
 COUNT(*) AS total_rows,
 COUNTIF(Customer_ID IS NULL) AS null_customer_id,
 COUNTIF(Device IS NULL) AS null_device,
 COUNTIF(GameGenre IS NULL) AS null_game_genre,
 COUNTIF(SessionCount IS NULL) AS null_session_count,
 COUNTIF(AverageSessionLength IS NULL) AS null_average_session_length,
 COUNTIF(SpendingSegment IS NULL) AS null_spending_segment,
 COUNTIF(InAppPurchaseAmount IS NULL) AS null_in_app_purchase_amount,
 COUNTIF(FirstPurchaseDaysAfterInstall IS NULL) AS null_fpadi,
 COUNTIF(PaymentMethod IS NULL) AS null_payment_method,
 COUNTIF(LastPurchaseDate IS NULL) AS null_purchase_date
FROM `mis784t22025-466123.MIS784_A3.Mobile_Game_Inapp_Purchases_A3`;
```

Initial inspection shows that Device and GameGenre have 60 null values each. InAppPurcahseAmount, FirstPurchaseDaysAfterInstall, PaymentMethod, and LastPurchaseDate have 136 null values each => this makes sense becasue if the customers don't purchase anything then there's no records of these 4 variables.

Row	total_rows	null_customer_id	null_device	null_game_genre	null_session_count	null_average_ses...	null_spending_se...	null_in_app_purch...	null_fpadi	null_payment_me...	null_purchase_date
1	3024	0	60	60	0	0	0	136	136	136	136

Create cleaned table

```sql
CREATE OR REPLACE TABLE `mis784t22025-466123.MIS784_A3.Mobile_Game_Inapp_Purchases_Cleaned`
AS
SELECT
 Customer_ID,
 IFNULL(Device, "Unknown") AS Device,
 IFNULL(GameGenre, "Unknown") AS GameGenre,
 IFNULL(SessionCount, 0) AS SessionCount,
 IFNULL(AverageSessionLength, 0.0) AS AverageSessionLength,
 IFNULL(SpendingSegment, "Unknown") AS SpendingSegment,
 IFNULL(InAppPurchaseAmount, 0.0) AS InAppPurchaseAmount,
 IFNULL(FirstPurchaseDaysAfterInstall, 0) AS FirstPurchaseDaysAfterInstall,
 IFNULL(PaymentMethod, "Not Applicable") AS PaymentMethod,
 LastPurchaseDate,
 CASE
  WHEN LastPurchaseDate IS NULL THEN "No Purchase"
  ELSE "Has Purchased"
 END AS PurchaseStatus
FROM `mis784t22025-466123.MIS784_A3.Mobile_Game_Inapp_Purchases_A3`;
```

Donor Behaviour Patterns

Traditional Donor Monetary Contribution

```
# Monetary Contribution
SELECT
 MIN(DonationAmount) AS min_value,
 ROUND(APPROX_QUANTILES(DonationAmount, 4)[OFFSET(1)],2) AS q1,
 ROUND(APPROX_QUANTILES(DonationAmount, 2)[OFFSET(1)],2) AS median,
 ROUND(APPROX_QUANTILES(DonationAmount, 4)[OFFSET(3)],2) AS q3,
 MAX(DonationAmount) AS max_value,
 ROUND(AVG(DonationAmount),2) AS avg_value,
 -- IQR = Q3 - Q1
 ROUND(
   APPROX_QUANTILES(DonationAmount, 4)[OFFSET(3)]
   - APPROX_QUANTILES(DonationAmount, 4)[OFFSET(1)],2
 ) AS iqr
FROM `mis784t22025-466123.MIS784_A3.Tradition_Donation_Patterns_A3`;
```

Results

Row	min_value	q1	median	q3	max_value	avg_value	iqr
1	5.0	30.0	96.0	336.0	5000.0	207.74	306.0

```
# Monetary Contribution by product type

SELECT
 Product,
 MIN(DonationAmount) AS min_value,
 ROUND(APPROX_QUANTILES(DonationAmount, 4)[OFFSET(1)],2) AS q1,
 ROUND(APPROX_QUANTILES(DonationAmount, 2)[OFFSET(1)],2) AS median,
 ROUND(APPROX_QUANTILES(DonationAmount, 4)[OFFSET(3)],2) AS q3,
 MAX(DonationAmount) AS max_value,
 ROUND(AVG(DonationAmount),2) AS avg_value,
 ROUND(
   APPROX_QUANTILES(DonationAmount, 4)[OFFSET(3)]
   - APPROX_QUANTILES(DonationAmount, 4)[OFFSET(1)],2
 ) AS iqr
FROM `mis784t22025-466123.MIS784_A3.Tradition_Donation_Patterns_A3`
GROUP BY Product
ORDER BY Product;
```

Results

Row	Product	min_value	q1	median	q3	max_value	avg_value	iqr
1	General_Donation	5.0	24.0	60.0	120.0	5000.0	103.88	96.0
2	Membership	10.0	48.0	144.0	432.0	1394.0	245.84	384.0
3	Membership_TopUp	5.0	10.0	20.0	45.0	490.0	34.12	35.0

```
# Exploring top 10% Contribution Share

WITH donor_totals AS (
SELECT
  Customer_ID,
  SUM(DonationAmount) AS total_donated
FROM `mis784t22025-466123.MIS784_A3.Tradition_Donation_Patterns_A3`
```

```sql
 GROUP BY Customer_ID
),
ranked AS (
 SELECT
   total_donated,
   NTILE(10) OVER (ORDER BY total_donated DESC) AS decile
 FROM donor_totals
)
SELECT
 ROUND(SUM(CASE WHEN decile = 1 THEN total_donated ELSE 0 END) / SUM(total_donated) * 100,2) AS
top10_pct_share
FROM ranked;
```

Results

Row	top10_pct_share
1	31.89

In-app Monetary Contribution

```sql
# Monetary Contribution
SELECT
 MIN(InAppPurchaseAmount) AS min_value,
 ROUND(APPROX_QUANTILES(InAppPurchaseAmount, 4)[OFFSET(1)],2) AS q1,
 ROUND(APPROX_QUANTILES(InAppPurchaseAmount, 2)[OFFSET(1)],2) AS median,
 ROUND(APPROX_QUANTILES(InAppPurchaseAmount, 4)[OFFSET(3)],2) AS q3,
 MAX(InAppPurchaseAmount) AS max_value,
 ROUND(AVG(InAppPurchaseAmount),2) AS avg_value,
 -- IQR = Q3 - Q1
 ROUND(
   APPROX_QUANTILES(InAppPurchaseAmount, 4)[OFFSET(3)]
   - APPROX_QUANTILES(InAppPurchaseAmount, 4)[OFFSET(1)],2
 ) AS iqr
FROM `mis784t22025-466123.MIS784_A3.Mobile_Game_Inapp_Purchases_No_Nulls`;
```

Results

Row	min_value	q1	median	q3	max_value	avg_value	iqr
1	0.0	5.91	12.02	17.75	4964.45	101.24	11.84

```sql
# Monetary Contribution by Spending Segment

SELECT
 MIN(InAppPurchaseAmount) AS min_value,
 ROUND(APPROX_QUANTILES(InAppPurchaseAmount, 4)[OFFSET(1)],2) AS q1,
 ROUND(APPROX_QUANTILES(InAppPurchaseAmount, 2)[OFFSET(1)],2) AS median,
 ROUND(APPROX_QUANTILES(InAppPurchaseAmount, 4)[OFFSET(3)],2) AS q3,
 MAX(InAppPurchaseAmount) AS max_value,
 ROUND(AVG(InAppPurchaseAmount),2) AS avg_value,
 -- IQR = Q3 - Q1
 ROUND(
   APPROX_QUANTILES(InAppPurchaseAmount, 4)[OFFSET(3)]
   - APPROX_QUANTILES(InAppPurchaseAmount, 4)[OFFSET(1)],2
 ) AS iqr
FROM `mis784t22025-466123.MIS784_A3.Mobile_Game_Inapp_Purchases_No_Nulls`;
```

Results

Row	SpendingSegment	min_value	q1	median	q3	max_value	avg_value	iqr
1	Dolphin	20.53	128.17	245.66	351.72	497.91	245.79	223.55
2	Minnow	0.0	4.96	10.03	15.06	20.0	10.01	10.1
3	Whale	608.21	1869.1	2620.84	4102.62	4964.45	2850.5	2233.52

Exploring top 10% Contribution Share

```sql
WITH ranked AS (
 SELECT
   InAppPurchaseAmount AS total_donated,
   NTILE(10) OVER (ORDER BY InAppPurchaseAmount DESC) AS decile
 FROM `mis784t22025-466123.MIS784_A3.Mobile_Game_Inapp_Purchases_No_Nulls`
)
SELECT
 ROUND(SUM(CASE WHEN decile = 1 THEN total_donated ELSE 0 END) / SUM(total_donated) * 100,2) AS
top10_pct_share
FROM ranked;
```

Results

Row	top10_pct_share
1	85.62

RFM Segmemtation

```sql
CREATE OR REPLACE VIEW `mis784-466303.A3.v_donation_rfm` AS

SELECT

 a.*,

 NTILE(5) OVER (ORDER BY a.days_since_last_donation DESC, a.Customer_ID) AS R_score,

 NTILE(5) OVER (ORDER BY a.donation_frequency ASC, a.Customer_ID) AS F_score,

 NTILE(5) OVER (ORDER BY a.donation_total_amount ASC, a.Customer_ID) AS M_score

FROM `mis784-466303.A3.v_donation_agg` a;

CREATE OR REPLACE VIEW `mis784-466303.A3.v_donation_segment` AS

SELECT

 Customer_ID,

 CASE

   WHEN R_score IN (4,5) AND F_score = 5 AND M_score = 5 THEN 'Cannot Lose'

   WHEN R_score = 5 AND F_score IN (4,5) AND M_score IN (3,4,5) THEN 'Active Fans'

   WHEN R_score = 5 AND F_score IN (2,3) AND M_score IN (4,5) THEN 'Promising Newbies'
```

```sql
      WHEN R_score IN (3,4,5) AND F_score IN (2,3,4) AND M_score IN (2,3,4) THEN 'At Risk'

      ELSE 'Other'

    END AS donation_rfm_segment

  FROM `mis784-466303.A3.v_donation_rfm`;

  -- in-app – aggregates and rfm

CREATE OR REPLACE VIEW `mis784-466303.A3.v_inapp_agg` AS

WITH asof AS (

  SELECT MAX(LastPurchaseDate) AS as_of

  FROM `mis784-466303.A3.mobile_game_inapp_purchases`

  WHERE LastPurchaseDate IS NOT NULL)

SELECT

  m.Customer_ID,

  MAX(m.LastPurchaseDate) AS last_purchase_date,

  COUNTIF(m.InAppPurchaseAmount IS NOT NULL) AS purchase_frequency,

  SUM(IFNULL(m.InAppPurchaseAmount,0)) AS purchase_total_amount,

  DATE_DIFF((SELECT as_of FROM asof), MAX(m.LastPurchaseDate), DAY) AS days_since_last_purchase

FROM `mis784-466303.A3.mobile_game_inapp_purchases` m

WHERE m.LastPurchaseDate IS NOT NULL

GROUP BY m.Customer_ID;

CREATE OR REPLACE VIEW `mis784-466303.A3.v_inapp_rfm` AS

SELECT

  a.*,

  NTILE(5) OVER (ORDER BY a.days_since_last_purchase DESC, a.Customer_ID) AS R_score,

  NTILE(5) OVER (ORDER BY a.purchase_frequency ASC, a.Customer_ID) AS F_score,

  NTILE(5) OVER (ORDER BY a.purchase_total_amount ASC, a.Customer_ID) AS M_score

FROM `mis784-466303.A3.v_inapp_agg` a;

CREATE OR REPLACE VIEW `mis784-466303.A3.v_inapp_segment` AS

SELECT

  Customer_ID,
```

```sql
    CASE

      WHEN R_score IN (4,5) AND F_score = 5 AND M_score = 5 THEN 'Cannot Lose'

      WHEN R_score = 5 AND F_score IN (4,5) AND M_score IN (3,4,5) THEN 'Active Fans'

      WHEN R_score = 5 AND F_score IN (2,3) AND M_score IN (4,5) THEN 'Promising Newbies'

      WHEN R_score IN (3,4,5) AND F_score IN (2,3,4) AND M_score IN (2,3,4) THEN 'At Risk'

      ELSE 'Other'

    END AS inapp_rfm_segment

FROM `mis784-466303.A3.v_inapp_rfm`;

-- unified rfm view

CREATE OR REPLACE VIEW `mis784-466303.A3.v_rfm_union` AS

SELECT

  'donations' AS domain,

  d.Customer_ID,

  d.last_donation_date AS last_txn_date,

  d.days_since_last_donation AS days_since_last,

  d.donation_frequency AS frequency,

  d.donation_total_amount AS total_amount,

  d.R_score, d.F_score, d.M_score,

  s.donation_rfm_segment AS segment

FROM `mis784-466303.A3.v_donation_rfm` d

JOIN `mis784-466303.A3.v_donation_segment` s USING (Customer_ID)

UNION ALL

SELECT

  'inapp' AS domain,

  i.Customer_ID,

  i.last_purchase_date AS last_txn_date,

  i.days_since_last_purchase AS days_since_last,

  i.purchase_frequency AS frequency,

  i.purchase_total_amount AS total_amount,
```

```sql
    i.R_score, i.F_score, i.M_score,

    s.inapp_rfm_segment AS segment

FROM `mis784-466303.A3.v_inapp_rfm` i

JOIN `mis784-466303.A3.v_inapp_segment` s USING (Customer_ID);

-- unified rfm breakdown

SELECT

    'inapp' AS domain,

    d.Customer_ID,

    d.last_purchase_date AS last_txn_date,

    d.days_since_last_purchase AS days_since_last,

    d.purchase_frequency AS frequency,

    ROUND(d.purchase_total_amount,2) AS total_amount,

    d.R_score, d.F_score, d.M_score,

    s.inapp_rfm_segment AS segment,

FROM `mis784-466303.A3.v_inapp_rfm` d

JOIN `mis784-466303.A3.v_inapp_segment` s USING (Customer_ID)

UNION ALL

SELECT

    'donations' AS domain,

    d.Customer_ID,

    d.last_donation_date AS last_txn_date,

    d.days_since_last_donation AS days_since_last,

    d.donation_frequency AS frequency,

    ROUND(d.donation_total_amount,2) AS total_amount,

    d.R_score, d.F_score, d.M_score,

    s.donation_rfm_segment AS segment,

FROM `mis784-466303.A3.v_donation_rfm` d

JOIN `mis784-466303.A3.v_donation_segment` s USING (Customer_ID)

ORDER BY domain, R_score DESC, F_score DESC, M_score DESC, total_amount DESC;
```

Row	domain	Customer_ID	last_txn_date	days_since_last	frequency	total_amount	R_score	F_score	M_score	segment
1	donations	5818194	2025-05-12	103	3	5500.0	5	5	5	Cannot Lose
2	donations	7321704	2025-07-10	44	9	3064.0	5	5	5	Cannot Lose
3	donations	4504504	2025-08-09	14	9	2755.0	5	5	5	Cannot Lose
4	donations	5900304	2025-06-30	54	4	2200.0	5	5	5	Cannot Lose
5	donations	2273704	2025-07-22	32	10	2076.0	5	5	5	Cannot Lose
6	donations	5112404	2025-07-12	42	6	2074.0	5	5	5	Cannot Lose
7	donations	7319304	2025-07-12	42	7	2062.0	5	5	5	Cannot Lose
8	donations	7152504	2025-08-12	11	4	2016.0	5	5	5	Cannot Lose
9	donations	9760304	2025-07-05	49	9	1958.0	5	5	5	Cannot Lose
10	donations	8180194	2025-08-03	20	3	1930.0	5	5	5	Cannot Lose
11	donations	5731304	2025-07-10	44	6	1864.0	5	5	5	Cannot Lose
12	donations	8129804	2025-07-13	41	3	1728.0	5	5	5	Cannot Lose
13	donations	2070004	2025-07-07	47	12	1680.0	5	5	5	Cannot Lose
14	donations	9363304	2025-07-06	48	6	1438.0	5	5	5	Cannot Lose

Traditional Donor Payment Methods

```
SELECT
 Channel_Pay,
 COUNT(*) AS frequency,
 ROUND(COUNT(*) * 100.0 / SUM(COUNT(*)) OVER(),2) AS pct
FROM `mis784t22025-466123.MIS784_A3.Tradition_Donation_Patterns_A3`
GROUP BY Channel_Pay
ORDER BY frequency DESC;
```

Results

Row	Channel_Pay	frequency	pct
1	Online Banking	669	13.37
2	Mobile Payment	631	12.61
3	Cash Payment	627	12.53
4	PayPal	623	12.45
5	Debit Card	622	12.43
6	Cryptocurrency	620	12.39
7	Bank Transfer	610	12.19
8	Credit Card	601	12.01

In-app Payment Methods

```
SELECT
 PaymentMethod,
 COUNT(*) AS frequency,
 ROUND(COUNT(*) * 100.0 / SUM(COUNT(*)) OVER(),2) AS pct
FROM `mis784t22025-466123.MIS784_A3.Mobile_Game_Inapp_Purchases_No_Nulls`
GROUP BY PaymentMethod
ORDER BY frequency DESC;
```

Results

Row	PaymentMethod	frequency	pct
1	Debit Card	419	15.1
2	Google Pay	415	14.95
3	Carrier Billing	401	14.45
4	Gift Card	399	14.38
5	Credit Card	397	14.31
6	Paypal	388	13.98
7	Apple Pay	356	12.83

Traditional Donor Frequency

```
# Overall Frequency

WITH donor_freq AS (
 SELECT
   Customer_ID,
   COUNT(Donation_ID) AS donation_count
 FROM `mis784t22025-466123.MIS784_A3.Tradition_Donation_Patterns_A3`
 GROUP BY Customer_ID
)
SELECT
 COUNT(*) AS total_donors,
 ROUND(AVG(donation_count),2) AS avg_donations,
 APPROX_QUANTILES(donation_count, 2)[OFFSET(1)] AS median_donations,
 ROUND(SUM(CASE WHEN donation_count=1 THEN 1 ELSE 0 END)*100.0/COUNT(*),2) AS
pct_one_time_donors,
 ROUND(SUM(CASE WHEN donation_count>1 THEN 1 ELSE 0 END)*100.0/COUNT(*),2) AS
pct_repeat_donors
FROM donor_freq;
```

Results

Row	total_donors	avg_donations	median_donations	pct_one_time_do...	pct_repeat_donors
1	2778	1.8	1	49.6	50.4

```
# Frequency segment
WITH donor_freq AS (
 SELECT
   Customer_ID,
   COUNT(Donation_ID) AS donation_count
 FROM `mis784t22025-466123.MIS784_A3.Tradition_Donation_Patterns_A3`
 GROUP BY Customer_ID
),
freq_with_customer AS (
 SELECT
   c.Customer_ID,
   c.Age,
   c.Gender,
```

```sql
    c.Family_Size,
    f.donation_count
  FROM donor_freq f
  JOIN `mis784t22025-466123.MIS784_A3.customer_A3` c
    ON f.Customer_ID = c.Customer_ID
),
freq_with_agegroup AS (
  SELECT *,
    CASE
      WHEN Age < 30 THEN 'Under 30'
      WHEN Age BETWEEN 30 AND 44 THEN '30–44'
      WHEN Age BETWEEN 45 AND 59 THEN '45–59'
      ELSE '60+'
    END AS Age_Group
  FROM freq_with_customer
)
SELECT
  Age_Group,
  CASE
    WHEN donation_count = 1 THEN 'One-time'
    WHEN donation_count BETWEEN 2 AND 4 THEN 'Occasional (2–4)'
    ELSE 'Frequent (5+)'
  END AS frequency_segment,
  COUNT(*) AS donor_count,
  ROUND(100 * COUNT(*) / SUM(COUNT(*)) OVER (PARTITION BY Age_Group), 1) AS pct_within_age_group
FROM freq_with_agegroup
GROUP BY Age_Group, frequency_segment
ORDER BY Age_Group, frequency_segment;
```

Results

Age_Group	frequency_segment	donor_count	pct_within_age_group
30–44	Frequent (5+)	21	1.9
30–44	Occasional (2–4)	514	47.6
30–44	One-time	544	50.4
45–59	Frequent (5+)	37	7.4
45–59	Occasional (2–4)	247	49.4
45–59	One-time	216	43.2
60+	Frequent (5+)	16	6.7
60+	Occasional (2–4)	128	53.3
60+	One-time	96	40
Under 30	Frequent (5+)	19	2
Under 30	Occasional (2–4)	405	43.2
Under 30	One-time	513	54.7

Traditional Donor Recency

```sql
# Overall recency
WITH donor_recency AS (
  SELECT
    Customer_ID,
    MAX(DonationDate) AS last_donation
  FROM `mis784t22025-466123.MIS784_A3.Tradition_Donation_Patterns_A3`
```

```
 GROUP BY Customer_ID
),
max_date AS (
 SELECT MAX(DonationDate) AS dataset_max_date
 FROM `mis784t22025-466123.MIS784_A3.Tradition_Donation_Patterns_A3`
)
SELECT
 ROUND(AVG(DATE_DIFF(m.dataset_max_date, r.last_donation, DAY)),1) AS avg_days_since,
 APPROX_QUANTILES(DATE_DIFF(m.dataset_max_date, r.last_donation, DAY), 2)[OFFSET(1)] AS
median_days_since,
 ROUND(100*AVG(CAST(DATE_DIFF(m.dataset_max_date, r.last_donation, DAY) > 90 AS INT64)),2)  AS
pct_inactive_90d,
 ROUND(100*AVG(CAST(DATE_DIFF(m.dataset_max_date, r.last_donation, DAY) > 180 AS INT64)),2) AS
pct_inactive_180d
FROM donor_recency r
CROSS JOIN max_date m;
```

Results

Row	avg_days_since ▼	median_days_since	pct_inactive_90d ▼	pct_inactive_180d ▼
1	268.6	254	84.2	67.24

Recency segments

```
WITH donor_recency AS (
 SELECT
   Customer_ID,
   MAX(DonationDate) AS last_donation
 FROM `mis784t22025-466123.MIS784_A3.Tradition_Donation_Patterns_A3`
 GROUP BY Customer_ID
),
max_date AS (
 SELECT MAX(DonationDate) AS dataset_max_date
 FROM `mis784t22025-466123.MIS784_A3.Tradition_Donation_Patterns_A3`
),
recency_segmented AS (
 SELECT
   c.Customer_ID,
   c.Age,
   c.Family_Size,
   c.Gender,
   r.last_donation,
   DATE_DIFF(m.dataset_max_date, r.last_donation, DAY) AS days_since_last,
   CASE
     WHEN DATE_DIFF(m.dataset_max_date, r.last_donation, DAY) <= 90  THEN 'Active (≤90d)'
     WHEN DATE_DIFF(m.dataset_max_date, r.last_donation, DAY) <= 180 THEN 'At Risk (91–180d)'
     ELSE 'Lapsed (>180d)'
   END AS recency_segment
 FROM donor_recency r
 JOIN `mis784t22025-466123.MIS784_A3.customer_A3` c
   ON r.Customer_ID = c.Customer_ID
 CROSS JOIN max_date m
),
recency_with_agegroup AS (
 SELECT *,
   CASE
```

```sql
    WHEN Age < 30 THEN 'Under 30'
    WHEN Age BETWEEN 30 AND 44 THEN '30–44'
    WHEN Age BETWEEN 45 AND 59 THEN '45–59'
    ELSE '60+'
  END AS Age_Group
FROM recency_segmented
)
SELECT
 Age_Group,
 recency_segment,
 COUNT(*) AS donor_count,
 ROUND(100 * COUNT(*) / SUM(COUNT(*)) OVER (PARTITION BY Age_Group), 1) AS pct_within_age_group
FROM recency_with_agegroup
GROUP BY Age_Group, recency_segment
ORDER BY Age_Group, recency_segment;
```

Results

Age_Group	recency_segment	donor_count	pct_within_age_group
30–44	Active (≤90d)	161	14.9
30–44	At Risk (91–180d)	178	16.5
30–44	Lapsed (>180d)	740	68.6
45–59	Active (≤90d)	89	17.8
45–59	At Risk (91–180d)	77	15.4
45–59	Lapsed (>180d)	334	66.8
60+	Active (≤90d)	34	14.2
60+	At Risk (91–180d)	44	18.3
60+	Lapsed (>180d)	162	67.5
Under 30	Active (≤90d)	152	16.2
Under 30	At Risk (91–180d)	165	17.6
Under 30	Lapsed (>180d)	620	66.2

In-app Engagement

Engagement summary

```sql
SELECT
 ROUND(AVG(SessionCount),1) AS avg_sessions,
 APPROX_QUANTILES(SessionCount, 2)[OFFSET(1)] AS median_sessions,
 ROUND(AVG(AverageSessionLength),1) AS avg_session_length,
 APPROX_QUANTILES(AverageSessionLength, 2)[OFFSET(1)] AS median_session_length,
 ROUND(AVG(FirstPurchaseDaysAfterInstall),1) AS avg_days_to_first_purchase,
 APPROX_QUANTILES(FirstPurchaseDaysAfterInstall, 2)[OFFSET(1)] AS median_days_to_first_purchase
FROM `mis784t22025-466123.MIS784_A3.Mobile_Game_Inapp_Purchases_No_Nulls`;
```

Results

Row	avg_sessions	median_sessions	avg_session_length	median_session_l...	avg_days_to_first...	median_days_to_f...
1	10.1	10	20.1	20.19	15.4	16

Spending Segment Engagement Summary
```sql
SELECT
```

```
 SpendingSegment,
 ROUND(AVG(SessionCount),1) AS avg_sessions,
 ROUND(AVG(AverageSessionLength),1) AS avg_session_length,
 ROUND(AVG(InAppPurchaseAmount),2) AS avg_spend
FROM `mis784t22025-466123.MIS784_A3.Mobile_Game_Inapp_Purchases_No_Nulls`
GROUP BY SpendingSegment
ORDER BY avg_spend DESC;
```

Results

Row	SpendingSegment ▼	avg_sessions ▼	avg_session_length	avg_spend ▼
1	Whale	10.7	18.5	2850.5
2	Dolphin	10.1	20.1	245.79
3	Minnow	10.0	20.2	10.01

```
# Game Genre choice per Spending Segment
SELECT
 SpendingSegment,
 GameGenre,
 COUNT(*) AS donor_count
FROM `mis784t22025-466123.MIS784_A3.Mobile_Game_Inapp_Purchases_No_Nulls`
GROUP BY SpendingSegment, GameGenre
ORDER BY SpendingSegment, donor_count DESC;
```

Results

SpendingSegment	GameGenre	donor_count
Dolphin	Adventure	36
Dolphin	Fighting	32
Dolphin	MMORPG	28
Dolphin	Racing	28
Dolphin	Sandbox	27
Dolphin	Simulation	27
Dolphin	Sports	27
Dolphin	Battle Royale	24
Dolphin	Action RPG	23
Dolphin	Casual	22
Dolphin	Strategy	22
Dolphin	MOBA	21
Dolphin	Puzzle	20
Dolphin	Card	19
Dolphin	Role Playing	19
Minnow	Simulation	178
Minnow	Sports	178
Minnow	Sandbox	174
Minnow	Card	171
Minnow	Puzzle	169
Minnow	Casual	168
Minnow	Role Playing	156
Minnow	MMORPG	155
Minnow	Strategy	154
Minnow	Action RPG	153
Minnow	Battle Royale	145
Minnow	MOBA	142
Minnow	Racing	140
Minnow	Fighting	136
Minnow	Adventure	123
Whale	Strategy	8
Whale	Battle Royale	7
Whale	Racing	6
Whale	Casual	4
Whale	Fighting	4
Whale	MOBA	4
Whale	Role Playing	4
Whale	Card	3
Whale	MMORPG	3
Whale	Puzzle	3

Churn Risk & Retention

1. **Data Check for Nulls:**

Query:

```
SELECT
 COUNTIF(Customer_ID IS NULL) AS null_customer,
 COUNTIF(Donation_ID IS NULL) AS null_donation,
 COUNTIF(DonationDate IS NULL) AS null_donationdate,
 COUNTIF(DonationEndDate IS NULL) AS null_donationenddate,
 COUNTIF(DonationAmount IS NULL) AS null_amount,
 COUNTIF(Product IS NULL) AS null_product,
 COUNTIF(Channel_Pay IS NULL) AS null_channel
FROM `mis784-sem1.Assignment_3.Tradition_Donation_Patterns`;
```

Result:

Row	null_customer ▾	null_donation ▾	null_donationdate ▾	null_donationend... ▾	null_amount ▾	null_product ▾	null_channel ▾
1	0	0	0	2872	0	0	0

Query:

```
SELECT
 COUNTIF(Customer_ID IS NULL) AS null_customer,
 COUNTIF(LastPurchaseDate IS NULL) AS null_lastpurchase,
 COUNTIF(SessionCount IS NULL) AS null_sessioncount,
 COUNTIF(AverageSessionLength IS NULL) AS null_avgsessionlength,
 COUNTIF(SpendingSegment IS NULL) AS null_segment,
 COUNTIF(PaymentMethod IS NULL) AS null_payment,
 COUNTIF(InAppPurchaseAmount IS NULL) AS null_inapppurchase
FROM `mis784-sem1.Assignment_3.Mobile_Game_Inapp_Purchases`;
```

Result:

Row	null_customer ▾	null_lastpurchase ▾	null_sessioncount ▾	null_avgsessionle... ▾	null_segment ▾	null_payment ▾	null_inapppurchase ▾
1	0	136	0	0	0	136	136

Query:

```
SELECT
 COUNTIF(Customer_ID IS NULL) AS null_customer,
 COUNTIF(Campaign_ID IS NULL) AS null_campaign,
 COUNTIF(Response IS NULL) AS null_response,
 COUNTIF(ClickThroughRate IS NULL) AS null_ctr,
 COUNTIF(EngagementFrequency IS NULL) AS null_engagement
FROM `mis784-sem1.Assignment_3.Campaign_Response`;
```

Result:

Row	null_customer ▾	null_campaign ▾	null_response ▾	null_ctr ▾	null_engagement ▾
1	0	0	0	0	0

Query:

```
SELECT
  COUNTIF(Customer_ID IS NULL) AS null_customer,
```

```
 COUNTIF(Age IS NULL) AS null_age,
 COUNTIF(Gender IS NULL) AS null_gender,
 COUNTIF(Occupation IS NULL) AS null_occupation,
 COUNTIF(Income_Level IS NULL) AS null_incomelevel,
 COUNTIF(Location IS NULL) AS null_location,
 COUNTIF(City IS NULL) AS null_city,
 COUNTIF(Family_Size IS NULL) AS null_familysize
FROM `mis784-sem1.Assignment_3.Customers`;
```

Result:

Row	null_customer	null_age	null_gender	null_occupation	null_incomelevel	null_location	null_city	null_familysize
1	0	0	0	0	0	0	0	0

2. Duplicates check:

Query:

```
SELECT COUNT(*) AS total_rows,
COUNT(DISTINCT Donation_ID) AS unique_donation_ids
FROM `mis784-sem1.Assignment_3.Tradition_Donation_Patterns`;
```

Result:

Row	total_rows	unique_donation_...
1	5003	5003

NOTE: Says there are 5003 unique donations made.

Query:

```
SELECT Customer_ID, DonationDate, COUNT(*) AS dup_cnt
FROM `mis784-sem1.Assignment_3.Tradition_Donation_Patterns`
GROUP BY Customer_ID, DonationDate
HAVING COUNT(*) > 1;
```

Result:

Row	Customer_ID	DonationDate	dup_cnt
1	5191704	2025-08-18	2
2	9568704	2025-03-26	2
3	863304	2024-12-03	2
4	3733804	2025-08-21	2
5	854504	2025-08-18	2
6	8642094	2025-08-18	2
7	8899404	2025-08-18	2
8	1121704	2025-08-14	2
9	843094	2025-07-20	2
10	5757304	2025-07-14	2

NOTE: However, checking as per Customer_ID and DonationDate, some donors are appearing two times (dup_cnt=2), which means they donated multiple times on the same date. These duplicates were not removed in order to maintain accurate donation patterns, because they represent actual separate donations, not data errors.

Query:

```
SELECT Customer_ID, LastPurchaseDate, COUNT(*) AS dup_cnt
FROM `mis784-sem1.Assignment_3.Mobile_Game_Inapp_Purchases`
GROUP BY Customer_ID, LastPurchaseDate
HAVING COUNT(*) > 1;
```

Result:

ℹ There is no data to display.

NOTE: Hence no duplicate data.

Query:

```
SELECT Customer_ID, Campaign_ID, COUNT(*) AS dup_cnt
FROM `mis784-sem1.Assignment_3.Campaign_Response`
GROUP BY Customer_ID, Campaign_ID
HAVING COUNT(*) > 1;
```

Result:

Row	Customer_ID	Campaign_ID	dup_cnt
1	1149894	1	2
2	870004	1	2
3	2853094	1	2
4	8038804	2	3
5	1757704	2	2
6	7935994	5	2
7	2751904	6	2
8	5947794	7	2
9	7067694	8	3
10	5047804	10	3
11	9337804	10	2

3. Outliers Check:

Query:

```
SELECT
 APPROX_QUANTILES(DonationAmount, 100) AS donation_amount_quantiles,
 APPROX_QUANTILES(DATE_DIFF(DonationEndDate, DonationDate, DAY), 100) AS duration_quantiles
FROM `mis784-sem1.Assignment_3.Tradition_Donation_Patterns`;
```

Result:

Row	donation_amount...	duration_quantiles
1	5	0
	10	0
	10	0
	10	0
	10	0
	10	0
	10	0
	10	0
	10	0

Row	donation_amount...	duration_quantiles
	672.0	322
	672.0	335
	720.0	351
	722.0	378
	768.0	385
	816.0	414
	864.0	441
	912.0	478
	5000.0	535

Query:

```
SELECT
 APPROX_QUANTILES(SessionCount, 100) AS session_count_quantiles,
 APPROX_QUANTILES(AverageSessionLength, 100) AS session_length_quantiles,
 APPROX_QUANTILES(InAppPurchaseAmount, 100) AS purchase_amount_quantiles
FROM `mis784-sem1.Assignment_3.Mobile_Game_Inapp_Purchases`;
```

Result:

Row	session_count_qu...	session_length_q...	purchase_amount...
1	1	5.01	0
	3	5.33	0.29
	4	5.6	0.51
	5	6.02	0.71
	5	6.33	0.92
	5	6.62	1.1
	5	6.86	1.36
	6	7.05	1.58
	6	7.39	1.74
	6	7.65	1.96
	6	7.93	2.25

Row	session_count_quantiles	session_length_quantiles	purchase_amount_quantiles
	14	31.98	244.31
	15	32.28	272.85
	15	32.68	300.9
	15	33.03	338.49
	15	33.32	375.6
	16	33.69	425.22
	16	33.98	469.65
	17	34.24	982.03
	18	34.68	2962.8
	22	34.99	4964.45

Query:

```
SELECT
 APPROX_QUANTILES(ClickThroughRate, 100) AS ctr_quantiles,
 APPROX_QUANTILES(EngagementFrequency, 100) AS engagement_quantiles
FROM `mis784-sem1.Assignment_3.Campaign_Response`;
```

Result:

Row	ctr_quantiles	engagement_quantiles
1	0.0	1
	0.01	1
	0.02	1
	0.03	1
	0.04	1
	0.05	1
	0.06	1
	0.07	1
	0.08	1
	0.09	1
	0.1	1

Row	ctr_quantiles	engagement_quantiles
	0.9	9
	0.91	9
	0.92	9
	0.93	9
	0.94	9
	0.95	9
	0.96	9
	0.97	9
	0.98	9
	0.99	9
	1.0	9

Result:

Row	age_quantiles	family_size_quant...
1	18	1
	19	1
	20	1
	20	1
	21	1
	21	1
	22	1
	22	1
	22	1
	23	1

Row	age_quantiles	family_size_quant...
	60	6
	61	6
	63	6
	65	6
	67	6
	70	6
	72	6
	76	6
	80	6
	98	6

4. Unified RFM Analysis (Create Views)

5. RFM Segment Counts

Query:

SELECT donation_rfm_segment AS segment, COUNT(*) AS donor_count
FROM `mis784-sem1.Assignment_3.v_donation_segment`
GROUP BY segment
ORDER BY donor_count DESC;

Result:

Row	segment ▼	donor_count ▼
1	Other	1889
2	At Risk	583
3	Cannot Lose	155
4	Active Fans	138
5	Promising Newbies	13

Query:

```
SELECT inapp_rfm_segment AS segment, COUNT(*) AS donor_count
FROM `mis784-sem1.Assignment_3.v_inapp_segment`
GROUP BY segment
ORDER BY donor_count DESC;
```

Result:

Row	segment ▼	donor_count ▼
1	Other	1889
2	At Risk	583
3	Cannot Lose	155
4	Active Fans	138
5	Promising Newbies	13

6. Campaign Response by Churn Segment (Traditional & In-App):

a) Campaign Response Rate by Churn Segment – Traditional Donors

Query:

```
WITH seg AS (
  SELECT CAST(Customer_ID AS STRING) AS customer_id_s,
         donation_rfm_segment AS churn_segment
  FROM `mis784-sem1.Assignment_3.v_donation_segment`
),
camp AS (
  SELECT
         CAST(Customer_ID AS STRING) AS customer_id_s,
         CASE
         WHEN SAFE_CAST(Response AS BOOL) IS NOT NULL THEN SAFE_CAST(Response AS BOOL)
         WHEN LOWER(CAST(Response AS STRING)) IN ('yes','y','true','1') THEN TRUE
         WHEN LOWER(CAST(Response AS STRING)) IN ('no','n','false','0')  THEN FALSE
         ELSE FALSE
         END AS responded_flag
  FROM `mis784-sem1.Assignment_3.Campaign_Response`
),
customer_level AS (
```

```sql
    SELECT s.churn_segment,
            c.customer_id_s,
            MAX(c.responded_flag) AS any_response
    FROM seg s
    JOIN camp c USING (customer_id_s)
    GROUP BY s.churn_segment, c.customer_id_s
)
SELECT
    churn_segment,
    COUNT(*) AS total_customers,
    SUM(CASE WHEN any_response THEN 1 ELSE 0 END) AS responders,
    ROUND(100.0 * SUM(CASE WHEN any_response THEN 1 ELSE 0 END) / COUNT(*), 2) AS
response_rate_pct
FROM customer_level
GROUP BY churn_segment
ORDER BY response_rate_pct DESC;
```

Result:

Row	churn_segment	total_customers	responders	response_rate_pct
1	Active Fans	132	110	83.33
2	Other	1814	1496	82.47
3	Promising Newbies	11	9	81.82
4	At Risk	551	447	81.13
5	Cannot Lose	145	116	80.0

b)   Campaign Response Rate by Churn Segment – Mobile In-App

Query:

```sql
WITH seg AS (
 SELECT CAST(Customer_ID AS STRING) AS customer_id_s,
         inapp_rfm_segment AS churn_segment
  FROM `mis784-sem1.Assignment_3.v_inapp_segment`
),
camp AS (
 SELECT
        CAST(Customer_ID AS STRING) AS customer_id_s,
        CASE
        WHEN SAFE_CAST(Response AS BOOL) IS NOT NULL THEN SAFE_CAST(Response AS BOOL)
        WHEN LOWER(CAST(Response AS STRING)) IN ('yes','y','true','1') THEN TRUE
        WHEN LOWER(CAST(Response AS STRING)) IN ('no','n','false','0')  THEN FALSE
        ELSE FALSE
        END AS responded_flag
  FROM `mis784-sem1.Assignment_3.Campaign_Response`
),
customer_level AS (
 SELECT s.churn_segment, c.customer_id_s, MAX(c.responded_flag) AS any_response
  FROM seg s
  JOIN camp c USING (customer_id_s)
  GROUP BY s.churn_segment, c.customer_id_s
```

```
)
SELECT
  churn_segment,
  COUNT(*) AS total_customers,
  SUM(CASE WHEN any_response THEN 1 ELSE 0 END) AS responders,
  ROUND(100.0 * SUM(CASE WHEN any_response THEN 1 ELSE 0 END) / COUNT(*), 2) AS
response_rate_pct
FROM customer_level
GROUP BY churn_segment
ORDER BY response_rate_pct DESC;
```

Result:

Row	churn_segment	total_customers	responders	response_rate_pct
1	Cannot Lose	45	41	91.11
2	Active Fans	115	96	83.48
3	Promising Newbies	85	70	82.35
4	Other	2009	1648	82.03
5	At Risk	495	391	78.99

c) Click-Through & Engagement by Churn Segment – Traditional Donors

Query:

```
WITH customer_avg AS (
 SELECT
      ds.donation_rfm_segment AS churn_segment,
      cr.Customer_ID,
      AVG(SAFE_CAST(cr.ClickThroughRate      AS FLOAT64)) AS customer_avg_ctr,
      AVG(SAFE_CAST(cr.EngagementFrequency   AS FLOAT64)) AS customer_avg_engagement
 FROM `mis784-sem1.Assignment_3.Campaign_Response` cr
 JOIN `mis784-sem1.Assignment_3.v_donation_segment` ds
      ON CAST(cr.Customer_ID AS STRING) = CAST(ds.Customer_ID AS STRING)
 WHERE cr.ClickThroughRate IS NOT NULL
      AND cr.EngagementFrequency IS NOT NULL
 GROUP BY churn_segment, cr.Customer_ID
)
SELECT
  churn_segment,
  ROUND(AVG(customer_avg_ctr) * 100.0, 2)  AS ctr_pct_customer_avg,
  ROUND(AVG(customer_avg_engagement), 2)       AS eng_freq_customer_avg,
  COUNT(DISTINCT Customer_ID)          AS customers_count
FROM customer_avg
GROUP BY churn_segment
ORDER BY eng_freq_customer_avg DESC;
```

Result:

Row	churn_segment ▼	avg_ctr_pct ▼	avg_engagement...	total_sends ▼
1	Other	49.41	5.05	5733
2	At Risk	49.93	4.99	1756
3	Active Fans	49.4	4.94	399
4	Cannot Lose	49.17	4.74	446
5	Promising Newbies	45.56	3.72	25

d) Click-Through & Engagement by Churn Segment – In-App

Query:

```
WITH customer_avg AS (
 SELECT
        ds.inapp_rfm_segment AS churn_segment,
        cr.Customer_ID,
        AVG(SAFE_CAST(cr.ClickThroughRate        AS FLOAT64)) AS customer_avg_ctr,
        AVG(SAFE_CAST(cr.EngagementFrequency   AS FLOAT64)) AS customer_avg_engagement
 FROM `mis784-sem1.Assignment_3.Campaign_Response` cr
 JOIN `mis784-sem1.Assignment_3.v_inapp_segment` ds
        ON CAST(cr.Customer_ID AS STRING) = CAST(ds.Customer_ID AS STRING)
 WHERE cr.ClickThroughRate IS NOT NULL
        AND cr.EngagementFrequency IS NOT NULL
 GROUP BY churn_segment, cr.Customer_ID
)
SELECT
 churn_segment,
 ROUND(AVG(customer_avg_ctr) * 100.0, 2)   AS ctr_pct_customer_avg,
 ROUND(AVG(customer_avg_engagement), 2)       AS eng_freq_customer_avg,
 COUNT(DISTINCT Customer_ID)            AS customers_count
FROM customer_avg
GROUP BY churn_segment
ORDER BY eng_freq_customer_avg DESC;
```

Result:

Row	churn_segment ▼	ctr_pct_customer...	eng_freq_custom...	customers_count ▼
1	Other	49.67	5.08	1814
2	At Risk	49.66	5.07	551
3	Active Fans	48.04	4.97	132
4	Cannot Lose	49.41	4.76	145
5	Promising Newbies	43.08	3.57	11

7. Behavioral Predictors Attrition:

a) Outliers (Traditional Donors)

Query (Bottom Donors- Lowest Totals):

```sql
SELECT
  donation_total_amount,
  COUNT(*) AS donor_count
FROM `mis784-sem1.Assignment_3.v_donation_agg`
GROUP BY donation_total_amount
ORDER BY donation_total_amount ASC
LIMIT 10;
```

Result:

Row	donation_total_a...	donor_count ▼
1	5.0	5
2	6.0	1
3	7.0	1
4	8.0	1
5	10.0	92
6	12.0	1
7	15.0	7
8	20.0	50
9	22.0	1
10	24.0	1

Query (Top Donors- Highest Totals):

```sql
SELECT
  Customer_ID,
  donation_total_amount,
  donation_frequency,
  days_since_last_donation
FROM `mis784-sem1.Assignment_3.v_donation_agg`
ORDER BY donation_total_amount DESC
LIMIT 10;
```

Result:

Row	Customer_ID	donation_total_a...	donation_frequency	days_since_last_donation
1	97404	8676.0	14	406
2	5818194	5500.0	3	103
3	468894	5000.0	1	428
4	7911504	5000.0	5	421
5	5204504	3400.0	10	280
6	7321704	3064.0	9	44
7	8107694	2910.0	6	370
8	1755404	2802.0	5	229
9	4504504	2755.0	9	14
10	673304	2592.0	3	566

b) Outliers (In-app)

Query (Bottom Donors- Lowest Totals):

```
SELECT
  purchase_total_amount,
  COUNT(*) AS customer_count
FROM `mis784-sem1.Assignment_3.v_inapp_agg`
GROUP BY purchase_total_amount
ORDER BY purchase_total_amount ASC
LIMIT 10;
```

Result:

Row	purchase_total_amount	customer_count
1	0.0	1
2	0.02	1
3	0.05	2
4	0.07	1
5	0.08	2
6	0.09	1
7	0.11	1
8	0.13	1
9	0.14	2
10	0.16	4

Query (Top Donors- Highest Totals):

```
SELECT
  Customer_ID,
  purchase_total_amount,
  purchase_frequency,
  days_since_last_purchase
FROM `mis784-sem1.Assignment_3.v_inapp_agg`
ORDER BY purchase_total_amount DESC
LIMIT 10;
```

Result:

Row	Customer_ID	purchase_total_amount	purchase_frequency	days_since_last_purchase
1	7326304	4964.45	1	197
2	6424094	4931.98	1	29
3	6837904	4930.28	1	120
4	3391194	4893.16	1	122
5	6886404	4770.77	1	159
6	4770304	4700.56	1	118
7	7066104	4619.48	1	6
8	4271204	4475.47	1	114
9	8213994	4471.87	1	108
10	4402804	4456.3	1	100

c) RFM Drivers (Traditional Donors)

Query:

```
SELECT
  s.donation_rfm_segment AS churn_segment,
  ROUND(AVG(a.days_since_last_donation), 1) AS avg_days_since_last,
  ROUND(AVG(a.donation_frequency), 2) AS avg_frequency,
  ROUND(AVG(a.donation_total_amount), 2) AS avg_total_donations,
  COUNT(*) AS donors_in_segment
FROM `mis784-sem1.Assignment_3.v_donation_agg` a
JOIN `mis784-sem1.Assignment_3.v_donation_segment` s
  ON a.Customer_ID = s.Customer_ID
GROUP BY churn_segment
ORDER BY avg_days_since_last DESC;
```

Result:

Row	churn_segment ▾	avg_days_since_l...	avg_frequency ▾	avg_total_donatio...	donors_in_segment
1	Other	326.4	1.59	351.53	1889
2	At Risk	182.4	1.59	270.91	583
3	Cannot Lose	101.5	4.44	998.97	155
4	Active Fans	51.6	2.64	400.95	138
5	Promising Newbies	37.3	1.92	553.15	13

c) RFM Drivers (In-App)

Query:

```
SELECT
  s.inapp_rfm_segment AS churn_segment,
  ROUND(AVG(a.days_since_last_purchase), 1) AS avg_days_since_last,
  ROUND(AVG(a.purchase_frequency), 2) AS avg_frequency,
  ROUND(AVG(a.purchase_total_amount), 2) AS avg_total_purchases,
  COUNT(*) AS customers_in_segment
FROM `mis784-sem1.Assignment_3.v_inapp_agg` a
JOIN `mis784-sem1.Assignment_3.v_inapp_segment` s
  ON a.Customer_ID = s.Customer_ID
GROUP BY churn_segment
ORDER BY avg_days_since_last DESC;
```

Result:

Row	churn_segment ▾	avg_days_since_l...	avg_frequency ▾	avg_total_purcha...	customers_in_se...
1	Other	129.4	1.0	109.97	2110
2	At Risk	74.2	1.0	11.62	525
3	Cannot Lose	48.2	1.0	330.79	45
4	Promising Newbies	22.2	1.0	295.85	88
5	Active Fans	20.0	1.0	143.3	120

d) Revenue Concentration by Top 10% vs Rest

Query:

```
WITH trad_totals AS (
  SELECT Customer_ID, SUM(DonationAmount) AS total_amt
  FROM `mis784-sem1.Assignment_3.Tradition_Donation_Patterns`
  WHERE DonationAmount IS NOT NULL
  GROUP BY Customer_ID
),
trad_ranked AS (
  SELECT
        *,
        NTILE(10) OVER (ORDER BY total_amt DESC) AS decile
  FROM trad_totals
),
```

```sql
inapp_totals AS (
  SELECT Customer_ID, SUM(IFNULL(InAppPurchaseAmount,0)) AS total_amt
  FROM `mis784-sem1.Assignment_3.Mobile_Game_Inapp_Purchases`
  GROUP BY Customer_ID
),
inapp_ranked AS (
  SELECT
        *,
        NTILE(10) OVER (ORDER BY total_amt DESC) AS decile
  FROM inapp_totals
)

-- Traditional
SELECT
  'Traditional' AS channel,
  ROUND(SUM(CASE WHEN decile = 1 THEN total_amt ELSE 0 END) / SUM(total_amt) * 100, 2) AS
top10_share_pct,
  ROUND(SUM(CASE WHEN decile <= 2 THEN total_amt ELSE 0 END) / SUM(total_amt) * 100, 2) AS
top20_share_pct,
  COUNT(*) AS donors_count
FROM trad_ranked

UNION ALL

-- In-App
SELECT
  'In-App' AS channel,
  ROUND(SUM(CASE WHEN decile = 1 THEN total_amt ELSE 0 END) / SUM(total_amt) * 100, 2) AS
top10_share_pct,
  ROUND(SUM(CASE WHEN decile <= 2 THEN total_amt ELSE 0 END) / SUM(total_amt) * 100, 2) AS
top20_share_pct,
  COUNT(*) AS customers_count
FROM inapp_ranked;
```

Result:

Row	channel	top10_share_pct	top20_share_pct	donors_count
1	Traditional	31.89	51.37	2778
2	In-App	86.54	92.75	3024

8. Demographics Impact on Retention:

a) Age Distribution Across Churn Groups - Traditional Donors

Query:

```sql
SELECT
  ds.donation_rfm_segment AS churn_segment,
  ROUND(AVG(c.Age), 1) AS avg_age,
  MIN(c.Age) AS min_age,
  MAX(c.Age) AS max_age,
```

```
  COUNT(*) AS donors_in_segment
FROM `mis784-sem1.Assignment_3.v_donation_segment` ds
JOIN `mis784-sem1.Assignment_3.Customers` c
  ON ds.Customer_ID = c.Customer_ID
GROUP BY churn_segment
ORDER BY avg_age;
```

Result:

Row	churn_segment ▼	avg_age ▼	min_age ▼	max_age ▼	donors_in_segment
1	Promising Newbies	36.2	23	56	13
2	At Risk	37.1	18	87	575
3	Other	37.5	18	91	1878
4	Active Fans	40.1	19	84	136
5	Cannot Lose	43.9	20	88	154

b) Age Distribution Across Churn Groups - In-App

Query:

```
SELECT
  iseg.inapp_rfm_segment AS churn_segment,
  ROUND(AVG(c.Age), 1) AS avg_age,
  MIN(c.Age) AS min_age,
  MAX(c.Age) AS max_age,
  COUNT(*) AS customers_in_segment
FROM `mis784-sem1.Assignment_3.v_inapp_segment` iseg
JOIN `mis784-sem1.Assignment_3.Customers` c
  ON iseg.Customer_ID = c.Customer_ID
GROUP BY churn_segment
ORDER BY avg_age;
```

Result:

Row	churn_segment ▼	avg_age ▼	min_age ▼	max_age ▼	customers_in_segment ▼
1	Active Fans	36.9	19	79	120
2	At Risk	37.8	18	85	525
3	Other	37.9	18	91	2110
4	Cannot Lose	38.1	19	93	45
5	Promising Newbies	39.6	19	88	88

a) Gender vs Churn Segment - Traditional Donors

Query:

```
SELECT
  ds.donation_rfm_segment AS churn_segment,
  c.Gender,
```

```
  COUNT(*) AS donor_count
FROM `mis784-sem1.Assignment_3.v_donation_segment` ds
JOIN `mis784-sem1.Assignment_3.Customers` c
  ON ds.Customer_ID = c.Customer_ID
GROUP BY churn_segment, c.Gender
ORDER BY churn_segment, donor_count DESC;
```

Result:

Row	churn_segment	Gender	donor_count
1	Active Fans	Unknown	117
2	Active Fans	Female	13
3	Active Fans	Male	6
4	At Risk	Unknown	530
5	At Risk	Female	24
6	At Risk	Male	21
7	Cannot Lose	Unknown	132
8	Cannot Lose	Female	14
9	Cannot Lose	Male	8
10	Other	Unknown	1701

b) Gender vs Churn Segment - In-App

Query:

```
SELECT
  iseg.inapp_rfm_segment AS churn_segment,
  c.Gender,
  COUNT(*) AS customer_count
FROM `mis784-sem1.Assignment_3.v_inapp_segment` iseg
JOIN `mis784-sem1.Assignment_3.Customers` c
  ON iseg.Customer_ID = c.Customer_ID
GROUP BY churn_segment, c.Gender
ORDER BY churn_segment, customer_count DESC;
```

Result:

Row	churn_segment	Gender	customer_count
1	Active Fans	Unknown	112
2	Active Fans	Female	4
3	Active Fans	Male	4
4	At Risk	Unknown	466
5	At Risk	Female	38
6	At Risk	Male	21
7	Cannot Lose	Unknown	41
8	Cannot Lose	Male	4
9	Other	Unknown	1908
10	Other	Female	122

c) Income vs Churn Segment - Traditional Donors

Query:

```
SELECT
  ds.donation_rfm_segment AS churn_segment,
  c.Income_Level,
  COUNT(*) AS donor_count,
  ROUND(100.0 * COUNT(*) / SUM(COUNT(*)) OVER (PARTITION BY ds.donation_rfm_segment), 2) AS
pct_within_segment
FROM `mis784-sem1.Assignment_3.v_donation_segment` ds
JOIN `mis784-sem1.Assignment_3.Customers` c
  ON ds.Customer_ID = c.Customer_ID
WHERE c.Income_Level IS NOT NULL
GROUP BY churn_segment, c.Income_Level
ORDER BY churn_segment, pct_within_segment DESC;
```

Result:

Row	churn_segment	Income_Level	donor_count	pct_within_segment
1	Active Fans	Medium	53	38.97
2	Active Fans	Low	50	36.76
3	Active Fans	High	33	24.26
4	At Risk	Medium	235	40.87
5	At Risk	High	176	30.61
6	At Risk	Low	164	28.52
7	Cannot Lose	Medium	67	43.51
8	Cannot Lose	High	45	29.22
9	Cannot Lose	Low	42	27.27
10	Other	Medium	735	39.14

d) Income vs Churn Segment - In-App

Query:

```sql
SELECT
  iseg.inapp_rfm_segment AS churn_segment,
  c.Income_Level,
  COUNT(*) AS customer_count,
  ROUND(100.0 * COUNT(*) / SUM(COUNT(*)) OVER (PARTITION BY iseg.inapp_rfm_segment), 2) AS
pct_within_segment
FROM `mis784-sem1.Assignment_3.v_inapp_segment` iseg
JOIN `mis784-sem1.Assignment_3.Customers` c
  ON iseg.Customer_ID = c.Customer_ID
WHERE c.Income_Level IS NOT NULL
GROUP BY churn_segment, c.Income_Level
ORDER BY churn_segment, pct_within_segment DESC;
```

Result:

Row	churn_segment	Income_Level	customer_count	pct_within_segme...
1	Active Fans	High	43	35.83
2	Active Fans	Medium	40	33.33
3	Active Fans	Low	37	30.83
4	At Risk	Medium	206	39.24
5	At Risk	High	161	30.67
6	At Risk	Low	158	30.1
7	Cannot Lose	Medium	24	53.33
8	Cannot Lose	Low	12	26.67
9	Cannot Lose	High	9	20.0
10	Other	Medium	844	40.0

e) Family Size vs Churn Segment – Traditional Donor

Query:

```sql
SELECT
  dseg.donation_rfm_segment AS churn_segment,
```

```
  ROUND(AVG(SAFE_CAST(c.Family_Size AS FLOAT64)), 2) AS avg_family_size,
  COUNT(*) AS donors_in_segment
FROM `mis784-sem1.Assignment_3.v_donation_segment` dseg
JOIN `mis784-sem1.Assignment_3.Customers` c
  ON dseg.Customer_ID = c.Customer_ID
WHERE SAFE_CAST(c.Family_Size AS FLOAT64) IS NOT NULL
GROUP BY churn_segment
ORDER BY avg_family_size DESC;
```

Result:

Row	churn_segment ▾	avg_family_size ▾	donors_in_segment
1	Promising Newbies	3.85	13
2	Cannot Lose	3.6	154
3	Other	3.5	1878
4	At Risk	3.49	575
5	Active Fans	3.38	136

a) Family Size vs Churn Segment - In-App

Query:

```
SELECT
  iseg.inapp_rfm_segment AS churn_segment,
  ROUND(AVG(SAFE_CAST(c.Family_Size AS FLOAT64)), 2) AS avg_family_size,
  COUNT(*) AS customers_in_segment
FROM `mis784-sem1.Assignment_3.v_inapp_segment` iseg
JOIN `mis784-sem1.Assignment_3.Customers` c
  ON iseg.Customer_ID = c.Customer_ID
WHERE SAFE_CAST(c.Family_Size AS FLOAT64) IS NOT NULL
GROUP BY churn_segment
ORDER BY avg_family_size DESC;
```

Result:

Row	churn_segment ▾	avg_family_size ▾	customers_in_se...
1	Promising Newbies	3.78	88
2	Active Fans	3.52	120
3	Other	3.52	2110
4	At Risk	3.38	525
5	Cannot Lose	3.33	45

Revenue Optimisation

-- reusable views

-- donations – aggregates and rfm

CREATE OR REPLACE VIEW `mis784-466303.A3.v_donation_agg` AS

WITH asof AS (

  SELECT MAX(DonationDate) AS as_of

  FROM `mis784-466303.A3.tradition_donation_patterns`

  WHERE DonationDate IS NOT NULL)

SELECT

  t.Customer_ID,

  MAX(t.DonationDate) AS last_donation_date,

  COUNT(*) AS donation_frequency,

  SUM(t.DonationAmount) AS donation_total_amount,

  DATE_DIFF((SELECT as_of FROM asof), MAX(t.DonationDate), DAY) AS days_since_last_donation

FROM `mis784-466303.A3.tradition_donation_patterns` t

WHERE t.DonationDate IS NOT NULL AND t.DonationAmount IS NOT NULL

GROUP BY t.Customer_ID;

 -- campaigns – dedup customer+campaign and attach campaign type

CREATE OR REPLACE VIEW `mis784-466303.A3.v_campaign_resp_dedup` AS

SELECT

  Customer_ID, Campaign_ID,

  MAX(Response) AS any_response,

  AVG(IFNULL(ClickThroughRate,0)) AS avg_ctr_per_person,

  AVG(IFNULL(EngagementFrequency,0)) AS avg_eng_per_person

FROM `mis784-466303.A3.campaign_response`

GROUP BY Customer_ID, Campaign_ID;

CREATE OR REPLACE VIEW `mis784-466303.A3.v_campaign_kpi_person` AS

SELECT

```sql
    r.Customer_ID,

    r.Campaign_ID,

    m.CampaignType,

    r.any_response,

    r.avg_ctr_per_person,

    r.avg_eng_per_person

FROM `mis784-466303.A3.v_campaign_resp_dedup` r

JOIN `mis784-466303.A3.marketing_campaigns` m USING (Campaign_ID);

-- donor behaviour profiling

-- unified rfm segment sizes and value

SELECT

    domain,

    segment,

    COUNT(*) AS num_customers,

    ROUND(SUM(total_amount),2) AS segment_total_amount,

    ROUND(AVG(total_amount),2) AS avg_amount_per_customer,

    ROUND(AVG(frequency),2) AS avg_frequency

FROM `mis784-466303.A3.v_rfm_union`

GROUP BY domain, segment

ORDER BY domain, segment_total_amount DESC;
```

Row	domain	segment	num_customers	segment_total_amount	avg_amount_per_customer	avg_frequency
1	donations	Other	1889	664033.0	351.53	1.59
2	donations	At Risk	583	157939.0	270.91	1.59
3	donations	Cannot Lose	155	154840.0	998.97	4.44
4	donations	Active Fans	138	55331.25	400.95	2.64
5	donations	Promising Newbies	13	7191.0	553.15	1.92
6	inapp	Other	2110	232040.04	109.97	1.0
7	inapp	Promising Newbies	88	26035.15	295.85	1.0
8	inapp	Active Fans	120	17195.66	143.3	1.0
9	inapp	Cannot Lose	45	14885.68	330.79	1.0
10	inapp	At Risk	525	6102.78	11.62	1.0

```sql
-- revenue optimisation

-- monthly/seasonal trends by product (donations)

SELECT

    FORMAT_DATE('%Y-%m', DATE_TRUNC(DonationDate, MONTH)) AS month,
```

```sql
  Product AS product_type,

  COUNT(*) AS num_transactions,

  ROUND(SUM(DonationAmount),2) AS total_amount,

  ROUND(AVG(DonationAmount),2) AS average_amount

FROM `mis784-466303.A3.tradition_donation_patterns`

WHERE DonationDate IS NOT NULL AND DonationAmount IS NOT NULL

GROUP BY month, product_type

ORDER BY month, product_type;
```

Row	month	product_type	num_transactions	total_amount	average_amount
1	2024-01	General_Donation	30	5090.0	169.67
2	2024-01	Membership	91	50630.0	556.37
3	2024-02	General_Donation	27	3059.0	113.3
4	2024-02	Membership	169	61350.0	363.02
5	2024-02	Membership_TopUp	15	415.0	27.67
6	2024-03	General_Donation	49	5839.0	119.16
7	2024-03	Membership	233	95854.0	411.39
8	2024-03	Membership_TopUp	6	405.0	67.5
9	2024-04	General_Donation	33	3771.0	114.27
10	2024-04	Membership	145	55690.0	384.07
11	2024-04	Membership_TopUp	5	110.0	22.0
12	2024-05	General_Donation	55	5551.0	100.93
13	2024-05	Membership	149	71036.0	476.75
14	2024-05	Membership_TopUp	2	70.0	35.0
15	2024-06	General_Donation	68	16450.0	241.91

```sql
-- high-value contributors (top 5%) across both domains

WITH don_b AS (

  SELECT APPROX_QUANTILES(DonationAmount,100)[OFFSET(95)] AS p95

  FROM `mis784-466303.A3.tradition_donation_patterns`

  WHERE DonationAmount IS NOT NULL),

inapp_b AS (

  SELECT APPROX_QUANTILES(InAppPurchaseAmount,100)[OFFSET(95)] AS p95

  FROM `mis784-466303.A3.mobile_game_inapp_purchases`

  WHERE InAppPurchaseAmount IS NOT NULL),

u AS (

  SELECT 'donations' AS domain, Customer_ID, DonationAmount AS amount
```

```sql
FROM `mis784-466303.A3.tradition_donation_patterns`

WHERE DonationAmount IS NOT NULL

UNION ALL

SELECT 'inapp' AS domain, Customer_ID, InAppPurchaseAmount AS amount

FROM `mis784-466303.A3.mobile_game_inapp_purchases`

WHERE InAppPurchaseAmount IS NOT NULL AND LastPurchaseDate IS NOT NULL)
SELECT

  domain,

  COUNT(*) AS hv_txn,

  ROUND(SUM(amount),2) AS hv_total_amount,

  ROUND(AVG(amount),2) AS hv_avg_amount

FROM u, don_b, inapp_b

WHERE (domain = 'donations' AND amount >= don_b.p95) OR (domain = 'inapp' AND amount >= inapp_b.p95)

GROUP BY domain

ORDER BY hv_total_amount DESC;
```

Row	domain	hv_txn	hv_total_amount	hv_avg_amount
1	donations	253	223179.0	882.13
2	inapp	145	211702.65	1460.02

```sql
-- game genre x device performance (in-app) – revenue + engagement

SELECT

  IFNULL(GameGenre,'Missing') AS game_type,

  IFNULL(Device,'Missing') AS device_type,

  COUNT(*) AS txn_count,

  ROUND(SUM(IFNULL(InAppPurchaseAmount,0)),2) AS total_revenue,

  ROUND(AVG(SessionCount),2) AS avg_sessions,

  ROUND(AVG(AverageSessionLength),2) AS avg_session_minutes

FROM `mis784-466303.A3.mobile_game_inapp_purchases`

GROUP BY game_type, device_type

ORDER BY total_revenue DESC;
```

Row	game_type	device_type	txn_count	total_revenue	avg_sessions	avg_session_min...
1	Battle Royale	iOS	85	17388.1	9.88	20.23
2	Strategy	Android	99	16807.96	9.8	18.6
3	Racing	iOS	66	15635.33	9.33	20.21
4	MOBA	Android	94	15589.99	10.49	20.69
5	Fighting	Android	103	15139.49	9.31	20.04
6	MMORPG	iOS	84	14467.97	10.3	21.63
7	Card	Android	131	13595.87	10.64	20.65
8	Sports	Android	143	13581.85	10.07	19.16
9	Role Playing	iOS	95	13229.3	10.4	19.95
10	Adventure	Android	90	10923.13	10.53	21.21
11	Battle Royale	Android	101	10634.32	10.0	22.0
12	Strategy	iOS	93	10209.27	10.56	19.25
13	Racing	Android	117	10143.25	9.86	19.19
14	Casual	iOS	78	10058.52	9.87	19.3
15	Sandbox	Android	135	9689.54	9.95	19.51

```sql
-- campaign effectiveness by type and audience

WITH camp_kpi AS (

  SELECT

    Campaign_ID,

    AVG(IF(any_response, 1, 0)) AS response_rate,

    AVG(avg_ctr_per_person) AS ctr,

    AVG(avg_eng_per_person) AS engagement

  FROM `mis784-466303.A3.v_campaign_kpi_person`

  GROUP BY Campaign_ID)

SELECT

  m.CampaignType AS campaign_type,

  m.TargetAudience AS audience,

  COUNT(*) AS num_campaigns,

  ROUND(AVG(k.response_rate), 2) AS avg_response_rate,

  ROUND(AVG(k.ctr), 2) AS avg_ctr,

  ROUND(AVG(k.engagement), 2) AS avg_engagement,

  ROUND(SUM(m.CampaignBudget), 2) AS total_budget

FROM `mis784-466303.A3.marketing_campaigns` m

LEFT JOIN camp_kpi k USING (Campaign_ID)

GROUP BY campaign_type, audience
```

ORDER BY avg_response_rate DESC, avg_engagement DESC;

Row	campaign_type	audience	num_campaigns	avg_response_rate	avg_ctr	avg_engagement	total_budget
1	SMS	New Donors	6	0.52	0.5	5.16	4209.12
2	Social Media	Returning Donors	9	0.52	0.48	4.9	5132.28
3	Social Media	High Value Donors	6	0.5	0.5	5.15	3768.3
4	Email	High Value Donors	7	0.5	0.51	5.04	3891.23
5	Direct Mail	New Donors	3	0.5	0.5	4.99	2211.38
6	Email	Returning Donors	4	0.5	0.5	4.96	2307.1
7	SMS	High Value Donors	11	0.49	0.49	5.06	6522.07
8	Direct Mail	High Value Donors	6	0.49	0.5	5.04	3067.77
9	SMS	Returning Donors	6	0.49	0.5	5.04	3515.41
10	Social Media	New Donors	7	0.49	0.51	5.02	3180.32
11	Direct Mail	Returning Donors	7	0.49	0.49	4.89	4117.63
12	Email	New Donors	2	0.46	0.46	4.85	1169.18

-- campaign response by donation rfm segment and campaign type

SELECT

  s.donation_rfm_segment,

  p.CampaignType AS campaign_type,

  ROUND(AVG(IF(p.any_response,1,0)),2) AS response_rate,

  ROUND(AVG(p.avg_ctr_per_person),2) AS avg_ctr,

  ROUND(AVG(p.avg_eng_per_person),2) AS avg_engagement

FROM `mis784-466303.A3.v_campaign_kpi_person` p

JOIN `mis784-466303.A3.v_donation_segment` s USING (Customer_ID)

GROUP BY s.donation_rfm_segment, campaign_type

ORDER BY s.donation_rfm_segment, response_rate DESC;

Row	donation_rfm_segment	campaign_type	response_rate	avg_ctr	avg_engagement
1	Active Fans	Email	0.59	0.53	4.86
2	Active Fans	SMS	0.5	0.5	5.17
3	Active Fans	Social Media	0.49	0.48	4.86
4	Active Fans	Direct Mail	0.43	0.47	4.85
5	At Risk	Social Media	0.51	0.5	4.99
6	At Risk	SMS	0.51	0.5	5.0
7	At Risk	Email	0.51	0.5	4.87
8	At Risk	Direct Mail	0.48	0.5	5.09
9	Cannot Lose	Direct Mail	0.49	0.5	4.63
10	Cannot Lose	Social Media	0.48	0.49	4.74
11	Cannot Lose	SMS	0.47	0.49	4.91
12	Cannot Lose	Email	0.46	0.5	4.6
13	Other	Direct Mail	0.5	0.5	4.93
14	Other	Social Media	0.5	0.49	5.07
15	Other	SMS	0.5	0.49	5.11

```sql
-- campaign cost per engaged person by campaign type

-- calculates both cost per unique engaged person and cost per engagement

WITH joined AS (
  SELECT
    m.CampaignType,
    m.Campaign_ID,
    m.CampaignBudget,
    p.Customer_ID,
    p.any_response
  FROM `mis784-466303.A3.marketing_campaigns` m
  LEFT JOIN `mis784-466303.A3.v_campaign_kpi_person` p USING (Campaign_ID)),
agg AS (
  SELECT
    CampaignType,
    ROUND(SUM(CampaignBudget), 2) AS total_budget,
    SUM(IF(any_response, 1, 0)) AS total_engagements,
    COUNT(DISTINCT IF(any_response, Customer_ID, NULL)) AS engaged_people_unique
  FROM joined
  GROUP BY CampaignType)
SELECT
  CampaignType,
  total_budget,
  total_engagements,
  engaged_people_unique,
  ROUND(SAFE_DIVIDE(total_budget, NULLIF(engaged_people_unique, 0)), 2) AS cost_per_unique_engaged,
  ROUND(SAFE_DIVIDE(total_budget, NULLIF(total_engagements, 0)), 2) AS cost_per_engagement
FROM agg
ORDER BY cost_per_unique_engaged ASC;
```

Row	CampaignType	total_budget	total_engagements	engaged_people_unique	cost_per_unique_engaged	cost_per_engagement
1	Email	1045190.24	908	817	1279.3	1151.09
2	Social Media	1765183.52	1610	1317	1340.31	1096.39
3	Direct Mail	1357161.57	1126	972	1396.26	1205.29
4	SMS	2044225.17	1656	1371	1491.05	1234.44