Perfect — this is **exactly the kind of schema OTT analytics teams love** 😄
I'll do this in **two layers**:

1. **What retention analyses matter (business view – beIN style)**
2. **Actual `dplyr`-ready code patterns** you can run on a real table

I'll assume your data frame is called **`subs`**.

1️⃣ Retention Analyses That Make Sense for *Your* Schema

Your schema is powerful because you have:

- Subscription lifecycle
- Renewal continuity
- Product & offer periods
- Winback vs churn
- Latest vs historical versions

This enables **true cohort-based retention**, not just activity retention.

A. Logo Retention (Customer-level)

Question: Do customers stay subscribed?

KPI

- Retained = Customer has **active subscription after expiry**
- Churned = No active subscription after expiry

📌 Key lens:

- Monthly vs Annual
- Sports-only vs 4K bundle
- Country (with caveats)

B. Subscription Retention (Contract-level)

Question: Do subscriptions renew or break?

Use:

- `SubscriptionID`
- `subscription_version`
- `subscriptionLatest`

This distinguishes:

- Continuous renewals (same ID)
- Break & rejoin (new ID)

C. Cohort Retention (Gold Standard)

Question: How long do users stay after joining?

Cohort = `subscription_start_Date` (month)

Track:

- Active status at Month 0, 1, 3, 6, 12

D. Offer Period Retention

Question: Which offers retain better long-term?

Compare:

- Monthly vs 6-month vs Annual

E. Product Retention

Question: Which product mix reduces churn?

Compare:

- Sports-only
- 4K
- Non-4K

F. Winback Analysis

Question: Who churns but comes back?

Use:

- `Subscription_type = "winback"`
- New `SubscriptionID`

G. Tenure / Lifetime Analysis

Question: How long does a customer stay subscribed?

Use:

- First start date → last expiry date

2️⃣ dplyr Code – Practical & Reusable

Step 0️⃣ Prep (Always Do This)

```
library(dplyr)
library(lubridate)

subs_clean <- subs %>%
  mutate(
    start_date = as.Date(subscription_start_Date),
    expiry_date = as.Date(ExpiryDate),
    start_month = floor_date(start_date, "month")
  )
```

1️⃣ Current Active Base (Baseline KPI)

```
active_base <- subs_clean %>%
  filter(subscriptionLatest == 1, subscription_Status == "active") %>%
  summarise(active_customers = n_distinct(CustomerID))
```

2️⃣ Logo Retention Rate (Month-on-Month)

```
logo_retention <- subs_clean %>%
  filter(subscriptionLatest == 1) %>%
  group_by(start_month) %>%
  summarise(
    total_customers = n_distinct(CustomerID),
    active_customers = n_distinct(
      CustomerID[subscription_Status == "active"]
    ),
    retention_rate = active_customers / total_customers
  )
```

3️⃣ Cohort Retention Table (Classic OTT View)

Step 1: Identify Cohort Month

```
cohort_df <- subs_clean %>%
  group_by(CustomerID) %>%
  summarise(cohort_month = min(start_month)) %>%
  left_join(subs_clean, by = "CustomerID")
```

Step 2: Months Since Start

```
cohort_df <- cohort_df %>%
  mutate(
    months_since_start = interval(cohort_month, start_month) %/% months(1)
  )
```

Step 3: Retention Matrix

```
cohort_retention <- cohort_df %>%
  filter(subscription_Status == "active") %>%
  group_by(cohort_month, months_since_start) %>%
  summarise(
    retained_customers = n_distinct(CustomerID),
    .groups = "drop"
  )
```

4️⃣ Offer Period Retention Comparison

```
offer_retention <- subs_clean %>%
  filter(subscriptionLatest == 1) %>%
  group_by(offer_period) %>%
  summarise(
    customers = n_distinct(CustomerID),
    active_customers = n_distinct(
      CustomerID[subscription_Status == "active"]
    ),
    retention_rate = active_customers / customers
  )
```

---

5 Product Retention

```
product_retention <- subs_clean %>%
  filter(subscriptionLatest == 1) %>%
  group_by(ProductName) %>%
  summarise(
    customers = n_distinct(CustomerID),
    retained = n_distinct(
      CustomerID[subscription_Status == "active"]
    ),
    retention_rate = retained / customers
  )
```

---

6 Continuous Renewal vs Break & Rejoin

Same Subscription ID = Healthy Retention

```
renewal_behavior <- subs_clean %>%
  group_by(CustomerID) %>%
  summarise(
    total_subscriptions = n_distinct(SubscriptionID),
    max_version = max(subscription_version, na.rm = TRUE)
  ) %>%
  mutate(
    renewal_type = case_when(
      total_subscriptions == 1 & max_version > 1 ~ "continuous",
      total_subscriptions > 1 ~ "break_and_rejoin",
      TRUE ~ "single_term"
    )
  )
```

7️⃣ Winback Rate

```
winback_rate <- subs_clean %>%
  group_by(CustomerID) %>%
  summarise(
    churned = any(subscription_type == "churned"),
    winback = any(subscription_type == "winback")
  ) %>%
  summarise(
    churned_customers = sum(churned),
    winback_customers = sum(winback),
    winback_rate = winback_customers / churned_customers
  )
```

8️⃣ Customer Lifetime (Tenure)

```
customer_lifetime <- subs_clean %>%
  group_by(CustomerID) %>%
  summarise(
    first_start = min(start_date),
    last_expiry = max(expiry_date),
    lifetime_days = as.numeric(last_expiry - first_start)
  )
```

9️⃣ Country-Aware Retention (VPN-Safe)

Use **dominant country per customer**:

```
customer_country <- subs_clean %>%
  count(CustomerID, Country) %>%
  group_by(CustomerID) %>%
  slice_max(n, n = 1)

country_retention <- subs_clean %>%
  filter(subscriptionLatest == 1) %>%
  left_join(customer_country, by = "CustomerID") %>%
  group_by(Country) %>%
  summarise(
    customers = n_distinct(CustomerID),
    retained = n_distinct(
      CustomerID[subscription_Status == "active"]
    )
  )
```

3️⃣ Retention KPIs for Management Slide

KPI	Meaning
Logo Retention %	Customers still active
Cohort Month-3 Retention	Stickiness
Offer Period Retention	Contract efficiency
Product Retention	Content value
Continuous Renewal %	Brand loyalty
Winback Rate	Recovery power
Avg Customer Lifetime	Long-term value

4️⃣ What I'd Suggest Next (High Impact)

- Convert cohort table → **heatmap**
- Add **survival curves**
- Merge **usage minutes** for *engagement-driven churn*
- Predict churn with **last 30-day inactivity**

If you want, next I can:

- Build **cohort heatmap code**
- Add **survival analysis (Kaplan–Meier)**
- Combine **usage + subscription**
- Design **executive retention dashboard**

Just tell me where you want to go next 👌

Nice, this is exactly how **real OTT analytics decks are built** 👍
I'll do **three things** for you in one flow:

1. **Generate ~100 synthetic subscription events** (realistic OTT-style)
2. Ensure **valid winbacks, overlaps, renewals**
3. Show **how to compute**
    - ✅ Winback time
    - ✅ Time spent in system (even with overlaps)

All in **R + dplyr**, production-ready logic.

1️⃣ Generate Synthetic Data (100+ rows)

Key design choices (realistic)

- Each customer: 2–6 subscriptions
- Mix of:
    - churn → winback (valid gaps)
    - overlapping renewals
    - plan switches
- Dates spread across months

R code: Data generation

```
library(dplyr)

library(lubridate)

set.seed(123)



n_customers <- 25

subs_per_customer <- sample(2:6, n_customers, replace = TRUE)



synthetic_data <- lapply(1:n_customers, function(cid) {


  n <- subs_per_customer[cid]

  start_date <- as.Date("2024-01-01") + sample(0:200, 1)
```

```
  data.frame(
    CustomerID = cid,
    Subscription_ID = sample(1000:9999, n),
    Subscription_version = 1:n,
    Subscription_start_date = start_date + cumsum(sample(25:40, n, replace = TRUE)),
    Expiry_date = NA,
    Product_name = sample(c("XYZTotal", "4KTotal", "FootballSeasonal"), n, replace =
TRUE),
    Offer_period = sample(c("monthly", "annual"), n, replace = TRUE),
    stringsAsFactors = FALSE
  ) %>%
    mutate(
      Expiry_date = ifelse(
        Offer_period == "monthly",
        Subscription_start_date + 30,
        Subscription_start_date + 365
      ) %>% as.Date(origin = "1970-01-01"),

      Subscription_status = ifelse(runif(n) > 0.7, "churned", "active"),

      Subscription_type = case_when(
        Subscription_status == "churned" ~ "churned",
        Subscription_version == 1 ~ "new",
        TRUE ~ sample(c("renewal", "winback"), 1, prob = c(0.6, 0.4))
      ),

      Subscription_latest = ifelse(Subscription_version == max(Subscription_version), 1, 0)
```

```
  )
})

subs <- bind_rows(synthetic_data)
nrow(subs)
```

👉 This will generate **~120–130 rows**, depending on random draw.

2️⃣ Calculate TRUE Winback Time (Strict Logic)

Definition

- Churn → next subscription
- Start date **after expiry**
- Subscription_type = `winback`

dplyr code

```
winback_analysis <- subs %>%
  arrange(CustomerID, Subscription_start_date) %>%
  group_by(CustomerID) %>%
  mutate(
    prev_status = lag(Subscription_status),
    prev_expiry = lag(Expiry_date),

    winback_flag = Subscription_type == "winback" &
                   prev_status == "churned" &
                   Subscription_start_date > prev_expiry,

    days_to_winback = ifelse(
```

```
    winback_flag,
    as.numeric(Subscription_start_date - prev_expiry),
    NA
  )
) %>%
ungroup()
```

✅ Sample Winback Output

```
winback_analysis %>%
  filter(winback_flag) %>%
  select(CustomerID, prev_expiry, Subscription_start_date, days_to_winback) %>%
  head(10)
```

Typical output:

CustomerID	Churn Expiry	Winback Start	Days
3	2024-08-10	2024-09-01	22
7	2024-06-14	2024-06-20	6
12	2024-10-05	2024-12-01	57

3️⃣ Time Spent in System (INCLUDING Overlaps)

This is **very important** for OTTs like beIN.

Problem

If subscriptions overlap, you **cannot just sum durations**.

Solution

Merge date ranges per customer.

Step 1 Create date intervals

```
library(IRanges)


customer_lifetime <- subs %>%
  group_by(CustomerID) %>%
  summarise(
    total_days_in_system = {
      ranges <- IRanges(
        start = as.numeric(Subscription_start_date),
        end = as.numeric(Expiry_date)
      )
      sum(width(reduce(ranges)))
    }
  )
```

Sample Output

CustomerID	Total Days in System

1	540
2	395
3	610

✅ Overlaps counted **once**
❌ No inflation

4️⃣ Combine Winback + Lifetime View (Exec-ready)

```
final_customer_metrics <- winback_analysis %>%
  filter(winback_flag) %>%
  group_by(CustomerID) %>%
  summarise(
    avg_days_to_winback = mean(days_to_winback),
    winback_count = n()
  ) %>%
  left_join(customer_lifetime, by = "CustomerID")
```

5️⃣ KPIs You Can Put in a Management Deck

Winback KPIs

```
final_customer_metrics %>%
  summarise(
    winback_customers = n(),
    avg_winback_days = mean(avg_days_to_winback),
```

```
  median_winback_days = median(avg_days_to_winback)
)
```

Segmentation

- Fast winback: ≤ 7 days
- Medium: 8–30 days
- Long: 30+ days

6️⃣ Why This Is OTT-Grade Logic (beIN / Netflix style)

✔ Handles overlapping subscriptions
✔ Separates renewal vs winback
✔ Gives **true churn recovery time**
✔ Produces **customer lifetime engagement**

7️⃣ If You Want Next

I can:

- Add **usage minutes** to weight winback value
- Create **survival curves**
- Build **SQL version**
- Show **dashboard KPIs (Power BI / Tableau logic)**

Just tell me where you want to take it 🚀