Appendix

```
libname sa
'\\apporto.com\dfs\IIT\Users\samhithaailnen
_ iit\Downloads'; run;
proc print data=sa.calendar(obs=100);
run;
proc print data=sa.flight_activity(obs=100);
run;
proc print
data=sa.loyalty_history(obs=100); run; proc
contents data=sa.calendar; run;
proc contents data=sa.flight_activity;
run;
proc contents data=sa.loyalty_history;
run;
/* Summary statistics for Customer
Flight Activity dataset */ proc means
data=sa.flight_activity;     var
Total_Flights Distance
Points_Accumulated Points_Redeemed; run;
/* Summary statistics for Customer
Loyalty History dataset */ proc means
data=sa.loyalty_history;     var Salary
CLV; run;
/* Frequency counts for demographic and
categorical variables in Loyalty History
dataset */ proc freq
data=sa.loyalty_history;     tables
Gender Education Marital_Status
Enrollment_Type Loyalty_Card; run;
/* Check for missing values in Flight
Activity dataset */ proc means
data=sa.flight_activity nmiss;     var
Total_Flights Distance
Points_Accumulated Points_Redeemed; run;
/* Check for missing values in Loyalty
History dataset */ proc means
data=sa.loyalty_history nmiss;     var
Salary CLV; run;
/*The Salary field has a significant number
of missing values (4,238 out of 16,737).*/
/*Approach*/
/* Step 1: Analyze the pattern of missing
values for Salary and key variables */
proc means data=sa.loyalty_history n
nmiss mean std min max;     var Salary
CLV;     title 'Analysis of Salary and
CLV'; run;
/* Step 2: Create a flag for missing
Salary values */ /*data loyalty_temp;
set sa.loyalty_history;     Salary_Missing
= missing(Salary); run;*/
/* Step 3: Analyze the relationship
between missing Salary and categorical
variables */ /*proc freq
data=loyalty_temp;     tables (Education
Gender Marital_Status
Enrollment_Type       Loyalty_Card)       *
Salary_Missing / chisq;      title 'Missing
Salary Pattern Analysis'; run;
/* Given the strong association with
education, martial status and loyalty card,
creating a segmented mean imputation using
these  */
/* Step 1: Calculate mean salaries for
College education based on other variables
*/ proc sql;    create table college_means
as     select          Marital_Status,

Loyalty_Card,        case             when
CLV < 4000 then 'Low'             when CLV
< 8000 then 'MediumLow'
            when CLV < 12000 then
'MediumHigh'             else 'High'
end as CLV_Group,         mean(Salary) as
mean_salary,         std(Salary) as
std_salary     from sa.loyalty_history
where Education ne 'College' /* Use
nonCollege education levels as reference */
group by          Marital_Status,
Loyalty_Card,         calculated CLV_Group
having mean_salary is not null; quit;
/* Step 2: Create temporary dataset with CLV
groups */ data loyalty_temp;     set
sa.loyalty_history;     if CLV < 4000 then
CLV_Group = 'Low';     else if CLV < 8000
then CLV_Group =
'Medium-Low';
    else if CLV < 12000 then CLV_Group =
'Medium-High';     else CLV_Group =
'High';     Salary_Missing =
missing(Salary); run;
/* Step 3: Merge with means and impute */
proc sql;     create table
loyalty_imputed as     select
        a.*,         case
when missing(a.Salary) then
b.mean_salary *
                /* Add some random variation
based on std */
                (1 + (ranuni(123) - 0.5) *
0.1)             else a.Salary          end
as Imputed_Salary     from loyalty_temp a
left join college_means b     on
a.Marital_Status = b.Marital_Status
and a.Loyalty_Card = b.Loyalty_Card
and a.CLV_Group = b.CLV_Group     where not
missing(calculated Imputed_Salary); quit;
/* Step 4: Create final cleaned dataset */
data sa.loyalty_history_clean;     set
loyalty_imputed;     Salary =
Imputed_Salary;     Salary_Imputed_Flag =
Salary_Missing;     drop Imputed_Salary
CLV_Group Salary_Missing; run;
/* Step 5: Validate imputation */ proc
means data=sa.loyalty_history_clean n mean
std min max;     class Education;     var
Salary;     title 'Validation of Salary
Distribution by Education'; run; /* Visual
validation */ proc sgplot
data=sa.loyalty_history_clean;     vbox
Salary / category=Education;     title
'Salary Distribution by Education
(After Imputation)';
run;
/*  Check  correlation  with  CLV  */  proc
corr      data=sa.loyalty_history_clean;
var Salary CLV;
    title 'Correlation between Salary and
CLV after Imputation'; run;
/* Check for missing values in Loyalty
History dataset */ proc means
```

```
data=sa.loyalty_history_clean nmiss;
var Salary CLV; run;
/* Step 1: Create churn indicator and
enrollment duration in loyalty_history
*/ data loyalty_base;     set
sa.loyalty_history_clean;
    /* Create churn indicator */
Is_Churned = not
missing(Cancellation_Year);      /*
Calculate membership duration (in
months) */
    if not missing(Cancellation_Year) then
Membership_Duration =
(Cancellation_Year*12 + Cancellation_Month)
-

(Enrollment_Year*12 + Enrollment_Month);
else
        Membership_Duration = (2018*12 + 12)
- /* Using 2018 as reference year */

(Enrollment_Year*12 + Enrollment_Month);
run;
proc print data= loyalty_base(obs =100);
run;
/* Step 2: Aggregate flight activity
with meaningful metrics */ proc sql;
create table flight_summary as
select          Loyalty_Number,
/* Flight Activity Metrics */
sum(Total_Flights) as
Total_Annual_Flights,
count(distinct Month) as Active_Months,
mean(Total_Flights) as
Avg_Flights_Per_Month,
sum(Distance) as Total_Distance,
/* Points Metrics */
sum(Points_Accumulated) as
Total_Points_Earned,
sum(Points_Redeemed) as
Total_Points_Redeemed,          /*
Engagement Metrics */
max(Total_Flights) as
Peak_Monthly_Flights,
        sum(case when Total_Flights > 0 then
1 else 0 end) as Months_With_Flights,
/*      Flight      consistency      */
std(Total_Flights) as
Flight_Variance,
        /* Average distance per flight -
handle division by zero */         case
when sum(Total_Flights) > 0
then sum(Distance) / sum(Total_Flights)
else 0          end as
Avg_Flight_Distance      from
sa.flight_activity     group by
Loyalty_Number; quit;
proc print data= flight_summary(obs =100);
run;
/* Verify the aggregation */ proc means
data=flight_summary n nmiss mean min max;
    var Total_Annual_Flights Active_Months
Avg_Flights_Per_Month
        Total_Distance Avg_Flight_Distance;
title 'Verification of Flight Activity
Aggregation'; run;
/* Step 3: Create final analysis dataset */
proc sql;
    create table customer_analysis_final as
select
        /* Customer Demographics */
        l.Loyalty_Number,
        l.Gender,
        l.Education,
        l.Marital_Status,
        l.Province,
        l.City,
        l.Salary,          /* Loyalty
Program Details */
        l.Loyalty_Card,
        l.Enrollment_Type,
        l.Membership_Duration,
        l.CLV,
        l.Is_Churned,          /* Flight
Activity Metrics */
coalesce(f.Total_Annual_Flights, 0) as
Total_Annual_Flights,
coalesce(f.Active_Months, 0) as
Active_Months,
coalesce(f.Avg_Flights_Per_Month, 0) as
Avg_Flights_Per_Month,
coalesce(f.Total_Distance, 0) as
Total_Distance,
coalesce(f.Avg_Flight_Distance, 0) as
Avg_Flight_Distance,          /* Points
Activity */
coalesce(f.Total_Points_Earned, 0) as
Total_Points_Earned,
coalesce(f.Total_Points_Redeemed, 0) as
Total_Points_Redeemed,          /* Derived
Metrics */         case              when
f.Total_Points_Earned > 0              then
f.Total_Points_Redeemed /
f.Total_Points_Earned
else 0          end as
Points_Redemption_Rate,          case
when f.Active_Months > 0
then f.Total_Annual_Flights /
f.Active_Months
else 0          end as
Flight_Frequency,         /*
Engagement Scores */
case
           when l.CLV > median(l.CLV) and

coalesce(f.Total_Annual_Flights, 0) >
median(f.Total_Annual_Flights)
then 'High Value'             when l.CLV
> median(l.CLV) or

coalesce(f.Total_Annual_Flights, 0)
> median(f.Total_Annual_Flights)
then 'Medium Value'             else
'Low Value'         end as
Customer_Segment
            from loyalty_base l     left
join flight_summary f     on
l.Loyalty_Number = f.Loyalty_Number; quit;
proc print data= customer_analysis_final(obs
=100);
run;
/* Step 4: Verify the merged dataset */
proc means data=customer_analysis_final
n nmiss mean std min max;     var CLV
Total_Annual_Flights Total_Distance
```

```
        Points_Redemption_Rate
Membership_Duration;     title
'Summary Statistics of Key
Metrics';      run;      proc      freq
data=customer_analysis_final;     tables
Customer_Segment              Is_Churned
Customer_Segment*Is_Churned    /    nocol
nopercent;      title 'Customer Segment
and Churn
Analysis';
run;
/*--------------ANALYSIS---------------*/
/*Summary Statistics*/ /* Basic customer
profiling */ proc means
data=customer_analysis_final n mean std
min max median;     var CLV
Total_Annual_Flights Total_Distance
Points_Redemption_Rate
Membership_Duration;     title 'Key
Customer Metrics Overview'; run; proc
freq data=customer_analysis_final;
tables Loyalty_Card Education Gender
Marital_Status
           Loyalty_Card*Is_Churned / chisq;
title 'Customer Demographics and Churn
Analysis';
run;
proc univariate
data=customer_analysis_final plots;     var
CLV Total_Annual_Flights;     class
Loyalty_Card;     title 'Detailed Customer
Value Distribution'; run; proc tabulate
data=customer_analysis_final;     class
Loyalty_Card Education Is_Churned;     var
CLV Total_Annual_Flights;     table
Loyalty_Card*Education,
Is_Churned*(CLV*mean
Total_Annual_Flights*mean);     title
'Customer Value by Segment'; run; /*
Regression Models*/ /* Model 1: Flight
Activity Analysis */ proc reg
data=customer_analysis_final;     model
Total_Annual_Flights = CLV
Points_Redemption_Rate

Membership_Duration;     title
'Flight Activity Drivers'; run;
/* Model 2: Churn Factors Analysis */ proc
reg data=customer_analysis_final;    model
Is_Churned = CLV Total_Annual_Flights
Points_Redemption_Rate
                      Membership_Duration
Salary;     title 'Churn Factors Analysis';
run; /* Conditional Logit   */ /* Step 1:
Create properly formatted choice data */ data
card_choice;    set customer_analysis_final;
/* Create three records for each customer -
one for each card type */     array cards[3]
$ ('Aurora' 'Nova'
'Star');             do   alt   =   1   to   3;
Card_Alternative = cards[alt];           /*
Create binary choice indicator */         if
Loyalty_Card = Card_Alternative then Choice =
1;          else Choice = 0;          output;
end; run;
/* Manually create interaction variables for
each Card_Alternative with 'Star' as the
reference */ data card_choice_interactions;
set card_choice;
    /* Total_Annual_Flights interactions */
    Total_Annual_Flights_Aurora =
Total_Annual_Flights * (Card_Alternative =
'Aurora');
    Total_Annual_Flights_Nova =
Total_Annual_Flights * (Card_Alternative =
'Nova');
    /* CLV interactions */
    CLV_Aurora = CLV * (Card_Alternative =
'Aurora');
    CLV_Nova = CLV * (Card_Alternative =
'Nova');
    /* Points_Redemption_Rate interactions
*/
    Points_Redemption_Rate_Aurora =
Points_Redemption_Rate * (Card_Alternative =
'Aurora');
    Points_Redemption_Rate_Nova =
Points_Redemption_Rate * (Card_Alternative =
'Nova'); run;
/* Step 2: Run conditional logit model with
manually created interaction terms */ proc
mdc data=card_choice_interactions;
model Choice = Total_Annual_Flights_Aurora
                  Total_Annual_Flights_Nova
CLV_Aurora
                  CLV_Nova

Points_Redemption_Rate_Aurora

Points_Redemption_Rate_Nova            /
type=clogit nchoice=3;     id
Loyalty_Number alt;     title 'Card Choice
Analysis with Manual Interactions and Star
as Reference'; run;
/* Step 3: Analyze results with frequencies
*/ proc freq data=card_choice;      tables
Card_Alternative*Choice / nocol nopercent;
title 'Distribution of Card Choices'; run;
/* Step 4: Check characteristics by card
type */ proc means data=card_choice;
var Total_Annual_Flights CLV
Points_Redemption_Rate;     class
Card_Alternative;     where Choice = 1;
title 'Customer Characteristics by
Chosen Card Type'; run;
```