**Scenarios for Data Generation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S. No.** | **Scenario Name** | **Description** | **Use / Metrics** | **My approach / Questions** |
| 1 | Quote Submission Drop-Off | User creates a submission and drops off before quote | Track the no. of submissions. | Considering the path for each group |
| 2 | Quote Generation Drop-Off | User creates a submission and quotes and then drops off. | Track the no. of quotes. |
| 3 | Successful Submission | User completes the quote process and successfully binds it. | Track the no. of successful bounds. |
| 4 | Drop-Off / Successful | User navigates back to make changes to previously entered data. The user may either drop off or complete the submission successfully. | Track submissions, drop-offs, or successful bounds. | Have got only a few records |
| 5 | Payment Error | User receives a quote but encounters an error during payment | Identify if the issue lies with the payment method or if the user dropped off during payment. | - |
| 6 | Quote Response Error | User’s session experiences long delays (15+ minutes) between page navigations. | Measure time spent on each page before drop off or session expiry. |  |
| 7 | UW Issues | User drops-off while answering UW questions, e.g., failing to disclose mortgagees on the property. | Track drop-offs. | Clicking on ‘No Mortgagees’ in spite of having entered Mortgagee details in prev page and being allowed to go to the next page is not helping in tracking the user drop-offs when answering UW questions |
| 8 | Digital Block  (Classification – not in appetite) | User creates a submission with certain classification / industry which is not in appetite. | Identify delays from UW in addressing the block. | List of industries not in appetite? |

Scenario 1, 2, 3 Queries: Quote Submission Drop-Off / Quote Generation Drop-Off / Successful Submission

SELECT

COUNT(\*) AS TotalSubmissions,

ClientIP, AccountNumber, SaveDateTime, QuoteStatus

FROM

[dbo].[FlattenPageViewData]

WHERE

QuoteStatus = 'Quoted' --'Bound' -- 'Submission/Draft'

GROUP BY

AccountNumber, ClientIP, SaveDateTime, QuoteStatus

ORDER BY SaveDateTime DESC

// Output:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TotalSubmissions | ClientIP | AccountNumber | SaveDateTime | QuoteStatus |
| 1 | 127.0.0.1 | A00000001 | 2024-10-30T08:37:48.413Z | Quoted |
| 1 | 127.0.0.1 | A00000001 | 2024-10-30T08:23:00.397Z | Quoted |
| 1 | 127.0.0.1 | A00000003 | 2024-10-30T08:13:24.362Z | Quoted |
| 1 | 127.0.0.1 | A00000002 | 2024-10-30T08:04:24.378Z | Quoted |
| 1 | 127.0.0.1 | A00000005 | 2024-10-29T15:48:55.702Z | Quoted |
| 1 | 127.0.0.1 | A00000002 | 2024-10-29T15:04:27.386Z | Quoted |

SELECT

CONVERT(DATE, SaveDateTime) AS SubmissionDate, -- Extract the date part

COUNT(\*) AS TotalSubmissions

FROM

[dbo].[FlattenPageViewData]

WHERE

QuoteStatus = 'Submission/Draft' and SaveDateTime between '2024-10-25' and '2024-10-30'-- Filter based on QuoteStatus condition 'Quoted'

GROUP BY

CONVERT(DATE, SaveDateTime) -- Group by the date portion

ORDER BY

SubmissionDate DESC

// Output:

|  |  |
| --- | --- |
| SubmissionDate | TotalSubmissions |
| 10/30/2024 | 8 |
| 10/29/2024 | 7 |
| 10/28/2024 | 12 |
| 10/26/2024 | 2 |

SELECT

CONVERT(DATE, SaveDateTime) AS SubmissionDate, -- Extract the date part

QuoteStatus, -- Group by quote status

COUNT(\*) AS TotalQuotes

FROM

[dbo].[FlattenPageViewData]

WHERE

QuoteStatus IN ('Submission/Draft', 'Quoted', 'Bound') -- Filter relevant statuses

GROUP BY

CONVERT(DATE, SaveDateTime), -- Group by the date

QuoteStatus -- Group by status

ORDER BY

SubmissionDate DESC,

QuoteStatus;

// Output

|  |  |  |
| --- | --- | --- |
| SubmissionDate | QuoteStatus | TotalQuotes |
| 10/30/2024 | Bound | 2 |
| 10/30/2024 | Quoted | 4 |
| 10/30/2024 | Submission/Draft | 8 |
| 10/29/2024 | Bound | 5 |
| 10/29/2024 | Quoted | 5 |
| 10/29/2024 | Submission/Draft | 7 |

SELECT

CONVERT(DATE, SaveDateTime) AS SubmissionDate, -- Extract the date part

SUM(CASE WHEN QuoteStatus = 'Submission/Draft' THEN 1 ELSE 0 END) AS SubmissionCount,

SUM(CASE WHEN QuoteStatus = 'Quoted' THEN 1 ELSE 0 END) AS QuotedCount,

SUM(CASE WHEN QuoteStatus = 'Bound' THEN 1 ELSE 0 END) AS BoundCount

FROM

[dbo].[FlattenPageViewData]

WHERE

QuoteStatus IN ('Submission/Draft', 'Quoted', 'Bound') -- Filter relevant statuses

GROUP BY

CONVERT(DATE, SaveDateTime) -- Group by the date

ORDER BY

SubmissionDate DESC;

//Output:

|  |  |  |  |
| --- | --- | --- | --- |
| SubmissionDate | SubmissionCount | QuotedCount | BoundCount |
| 10/30/2024 | 8 | 4 | 2 |
| 10/29/2024 | 7 | 5 | 5 |
| 10/28/2024 | 12 | 7 | 2 |
| 10/26/2024 | 2 | 0 | 0 |

SELECT

COUNT(\*) AS TotalSubmissions

FROM

[dbo].[FlattenPageViewData]

WHERE

QuoteStatus = 'Quoted'

AND SaveDateTime BETWEEN '2024-10-25' AND '2024-10-30';

// Output:

TotalSubmissions

16

Scenario 4 Queries: Drop-Off / Successful (Back traversal)

WITH NavigationChanges AS (

SELECT

SessionId,

PageName,

SaveDateTime,

QuoteStatus,

LAG(PageName) OVER (PARTITION BY SessionId ORDER BY SaveDateTime) AS PreviousPage,

LEAD(QuoteStatus) OVER (PARTITION BY SessionId ORDER BY SaveDateTime) AS NextQuoteStatus

FROM

[dbo].[FlattenPageViewData]

WHERE

QuoteStatus IN ('Submission/Draft', 'Quoted', 'Bound')

),

BackNavigation AS (

SELECT

SessionId,

MIN(SaveDateTime) AS FirstNavigationBackTime, -- First instance of navigating back

MAX(SaveDateTime) AS LastActivityTime, -- Last recorded activity in the session

CASE

WHEN MAX(QuoteStatus) = 'Bound' THEN 'Completed with Bound'

WHEN MAX(QuoteStatus) = 'Quoted' THEN 'Completed with Quoted'

ELSE 'Dropped Off'

END AS FinalStatus

FROM

NavigationChanges

WHERE

PageName = PreviousPage -- Indicates the user navigated back

GROUP BY

SessionId

)

SELECT

b.SessionId,

b.FirstNavigationBackTime,

b.LastActivityTime,

b.FinalStatus

FROM

BackNavigation b

ORDER BY

b.FinalStatus DESC,

b.LastActivityTime DESC;

Scenario 5 Queries: Payment Error

SELECT

AccountNumber,

ClientIP,

UserId,

CONVERT(DATE, SaveDateTime) AS Date,

COUNT(CASE WHEN ErrorType = 'UW Block' THEN 1 END) AS UW\_Blocks,

COUNT(CASE WHEN ErrorType = 'Validation Error' THEN 1 END) AS Validation\_Error,

COUNT(CASE WHEN ErrorType = 'Payment Error' THEN 1 END) AS Payment\_Error\_Count,

MAX(CASE WHEN ErrorType = 'Alert' THEN ErrorMessage ELSE NULL END) AS LastErrorMessage

--MAX(CASE WHEN EventName = 'PaymentError' THEN ErrorType ELSE NULL END) AS LastErrorType

FROM

[dbo].[FlattenPageViewData]

WHERE

ErrorType IN ('UW Block', 'Validation Error', 'Payment Error', 'Alert')

GROUP BY

AccountNumber,

ClientIP,

UserId,

CONVERT(DATE, SaveDateTime)

ORDER BY

Date DESC,

AccountNumber;

Scenario 6 Queries: Quote Response Error

* Time Spent on each page

WITH PageDurations AS (

SELECT

SessionId,

PageName,

SaveDateTime,

LEAD(SaveDateTime) OVER (PARTITION BY SessionId ORDER BY SaveDateTime) AS NextPageTime

FROM

[dbo].[FlattenPageViewData]

WHERE

SessionId IS NOT NULL

)

SELECT

SessionId,

PageName,

SaveDateTime AS PageEntryTime,

NextPageTime AS PageExitTime,

DATEDIFF(SECOND, SaveDateTime, NextPageTime) AS TimeSpentOnPage

FROM

PageDurations

WHERE

NextPageTime IS NOT NULL

ORDER BY

SessionId, SaveDateTime;

* Time Spent on each page (not in order of traversal in each session)

WITH PageDurations AS (

SELECT

SessionId,

PageName,

SaveDateTime,

LEAD(SaveDateTime) OVER (PARTITION BY SessionId ORDER BY SaveDateTime) AS NextPageTime

FROM

[dbo].[FlattenPageViewData]

WHERE

SessionId IS NOT NULL

--order by SaveDateTime

),

TimeSpent AS (

SELECT

SessionId,

PageName,

SaveDateTime AS PageEntryTime,

NextPageTime AS PageExitTime,

DATEDIFF(SECOND, SaveDateTime, NextPageTime) AS TimeSpentOnPage

FROM

PageDurations

WHERE

NextPageTime IS NOT NULL

)

SELECT

SessionId,

PageName,

SUM(TimeSpentOnPage) AS TotalTimeSpentOnPage -- Sum up time spent on the same page within the session

FROM

TimeSpent

GROUP BY

SessionId,

PageName

ORDER BY

SessionId,

PageName;

* Time Spent on each page

WITH PageDurations AS (

SELECT

SessionId,

PageName,

SaveDateTime,

LEAD(SaveDateTime) OVER (PARTITION BY SessionId ORDER BY SaveDateTime) AS NextPageTime

FROM

[dbo].[FlattenPageViewData]

WHERE

SessionId IS NOT NULL

),

TimeSpentOnPages AS (

SELECT

SessionId,

PageName,

SaveDateTime AS PageEntryTime,

NextPageTime AS PageExitTime,

DATEDIFF(SECOND, SaveDateTime, NextPageTime) AS TimeSpentOnPage

FROM

PageDurations

WHERE

NextPageTime IS NOT NULL

)

SELECT

SessionId,

PageName,

MIN(PageEntryTime) AS FirstVisitTime, -- First time this page was visited in the session

MAX(PageExitTime) AS LastExitTime, -- Last time this page was exited in the session

SUM(TimeSpentOnPage) AS TotalTimeSpentOnPage

FROM

TimeSpentOnPages

GROUP BY

SessionId,

PageName

ORDER BY

SessionId,

FirstVisitTime;

* Time Spent on each page + time spent in each session

WITH PageDurations AS (

SELECT

SessionId,

PageName,

SaveDateTime,

LEAD(SaveDateTime) OVER (PARTITION BY SessionId ORDER BY SaveDateTime) AS NextPageTime

FROM

[dbo].[FlattenPageViewData]

WHERE

SessionId IS NOT NULL

),

TimeSpentOnPages AS (

SELECT

SessionId,

PageName,

SaveDateTime AS PageEntryTime,

NextPageTime AS PageExitTime,

DATEDIFF(SECOND, SaveDateTime, NextPageTime) AS TimeSpentOnPage

FROM

PageDurations

WHERE

NextPageTime IS NOT NULL

)

SELECT

SessionId,

PageName,

MIN(PageEntryTime) AS FirstVisitTime, -- First time this page was visited in the session

MAX(PageExitTime) AS LastExitTime, -- Last time this page was exited in the session

SUM(TimeSpentOnPage) AS TotalTimeSpentOnPage, -- Total time spent on this page in the session

SUM(SUM(TimeSpentOnPage)) OVER (PARTITION BY SessionId) AS TotalSessionTime -- Total time for the entire session

FROM

TimeSpentOnPages

GROUP BY

SessionId,

PageName

ORDER BY

SessionId,

FirstVisitTime;