/\*

Airbnb Data Analysis in SQL Server Queries

Airbnb row dataset has been downloaded from the following website:

Inside Airbnb: Get the Data at http://insideairbnb.com/get-the-data/

Washington, D.C., District of Columbia, United States 13 September, 2023

The downloaded dataset list:

1) listings.csv

2) reviews.csv

3) calendar.csv

4) neighbourhoods.csv

The dataset has been imported into MS SQL Server in Local Database Machine.

The dataset has been cleaned (please refer to data\_cleaning.sql for details)

The dataset has been organized and structured (please refer to data\_modeling.sql)

The dataset's time series coverage (11/21/2008 - 9/12/2023)

The dataset's location coverage (District of Columbia, Virginia, Maryland USA)

\*/

-- A. KPI’s (Key Performance Indicator metrics)

--- 1. Total Bookings.

SELECT SUM(number\_of\_reviews) AS total\_bookings

FROM dbo.listings\_cleaned

--Outcome:

A screenshot of a computer screen

Description automatically generated

--- 2. Total Accommodation Properties.

SELECT COUNT(id) AS total\_accomm\_properties

FROM dbo.listings\_cleaned

--Outcome:

A screenshot of a computer

Description automatically generated

--- 3. Total Hosts.

SELECT COUNT(host\_id) AS total\_hosts

FROM dbo.host\_profile

--Outcome:

A screenshot of a computer

Description automatically generated

-- 4. Average Booking.

SELECT SUM(number\_of\_reviews) / COUNT(id) AS avg\_per\_accomm\_property

, SUM(number\_of\_reviews) / COUNT(distinct host\_id) AS avg\_per\_host

FROM dbo.listings\_cleaned AS lis

--Outcome:

A screenshot of a computer

Description automatically generated

-- B. Yearly Trend for Bookings

WITH cte AS

(SELECT DISTINCT YEAR(lis.first\_review) AS year\_

, (SELECT SUM(number\_of\_reviews) FROM dbo.listings\_cleaned

WHERE YEAR(first\_review) BETWEEN 2009 AND YEAR(lis.first\_review)) AS total\_bookings

, SUM(number\_of\_reviews) OVER (PARTITION BY YEAR(lis.first\_review)) AS new\_bookings

FROM dbo.listings\_cleaned AS lis)

SELECT \*

, CAST(CAST(new\_bookings AS DECIMAL(10,1))

/ CAST(total\_bookings AS DECIMAL(10,1))\*100 AS DECIMAL(10,1)) AS percentage\_of\_increase

FROM cte

ORDER BY year\_

--Outcome:

A screenshot of a computer screen

Description automatically generated

-- C. Yearly Trend for Accommodation Properties.

WITH cte AS

(SELECT DISTINCT YEAR(lis.first\_review) AS year\_

, (SELECT COUNT(id) FROM dbo.listings\_cleaned

WHERE YEAR(first\_review) BETWEEN 2009 AND YEAR(lis.first\_review)) AS total\_accomm\_properties

, COUNT(id) OVER (PARTITION BY YEAR(lis.first\_review)) AS new\_accomm\_properties

FROM dbo.listings\_cleaned AS lis)

SELECT \*

, CAST(CAST(new\_accomm\_properties AS DECIMAL(10,1))

/ CAST(total\_accomm\_properties AS DECIMAL(10,1))\*100 AS DECIMAL(10,1)) AS percentage\_of\_increase

FROM cte

ORDER BY year\_

--Outcome:

A screenshot of a computer

Description automatically generated

-- D. Yearly Trend for Host Growth.

WITH cte AS

(SELECT \*

, RANK() OVER(ORDER BY host\_since) rn

FROM dbo.host\_profile

)

SELECT YEAR(host\_since) AS year\_

, MAX(rn) AS total\_hosts

, COUNT(host\_id) AS new\_hosts

, CAST(CAST(COUNT(host\_id) AS DECIMAL(10,1))

/ CAST(MAX(rn) AS DECIMAL(10,1))\*100 AS DECIMAL(10,1)) AS percentage\_of\_increase

from cte

group by year(host\_since)

order by year(host\_since)

--Outcome:

A screenshot of a computer screen

Description automatically generated

-- E. Review Rating Breakdown By Stars.

SELECT DISTINCT ROUND(review\_scores\_rating, 0) AS star

, SUM(number\_of\_reviews) AS number\_of\_reviews

FROM dbo.listings\_cleaned

GROUP BY ROUND(review\_scores\_rating, 0)

ORDER BY ROUND(review\_scores\_rating, 0) DESC

--Outcome:

A screenshot of a computer

Description automatically generated

-- F. Review Comments Breakdown By Positive/Negative Feedback

--- 1. This statement create function to split and count the most used word.

CREATE FUNCTION [dbo].[DelimitedSplitN4K](

@pString NVARCHAR(4000),

@pDelimiter NCHAR(1)

)

RETURNS TABLE WITH SCHEMABINDING AS

RETURN

WITH E1(N) AS (

SELECT 1 UNION ALL SELECT 1 UNION ALL SELECT 1 UNION ALL SELECT 1 UNION ALL SELECT 1 UNION ALL

SELECT 1 UNION ALL SELECT 1 UNION ALL SELECT 1 UNION ALL SELECT 1 UNION ALL SELECT 1

),

E2(N) AS (SELECT 1 FROM E1 a, E1 b),

E4(N) AS (SELECT 1 FROM E2 a, E2 b),

cteTally(N) AS(

SELECT TOP (ISNULL(DATALENGTH(@pString)/2,0)) ROW\_NUMBER() OVER (ORDER BY (SELECT NULL)) FROM E4

),

cteStart(N1) AS (

SELECT 1 UNION ALL

SELECT t.N+1 FROM cteTally t WHERE SUBSTRING(@pString,t.N,1) = @pDelimiter

),

cteLen(N1,L1) AS(

SELECT s.N1,

ISNULL(NULLIF(CHARINDEX(@pDelimiter,@pString,s.N1),0)-s.N1,4000)

FROM cteStart s

)

SELECT

ItemNumber = ROW\_NUMBER() OVER(ORDER BY l.N1),

Item = SUBSTRING(@pString, l.N1, l.L1)

FROM cteLen l;

--- 2. This statement calls function dbo.DelimitedSplitN4K on dbo.reviews to split words

--- from comments column, distinct them and counts the number of occurrences.

SELECT TOP 50

x.Item,

COUNT(\*)

FROM dbo.reviews p

CROSS APPLY dbo.DelimitedSplitN4K(p.comments, ' ') x

WHERE LTRIM(RTRIM(x.Item)) <> ''

GROUP BY x.Item

ORDER BY COUNT(\*) DESC

--Outcome:

A screenshot of a computer

Description automatically generated

-- 3. After creating function dbo.DelimitedSplitN4K and splitting words,

-- this statement uses key words combination pattern to search and filter

-- comments into positive and negative categories.

WITH cte AS

(SELECT \*

, CASE

WHEN comments LIKE '%not\_clean%'

OR comments LIKE '%not\_nice%'

OR comments LIKE '%not\_recommend%'

OR comments LIKE '%not\_perfect%'

OR comments LIKE '%not\_comfortable%'

OR comments LIKE '%not\_good%'

OR comments LIKE '%not\_well%'

OR comments LIKE '%not\_wonderful%'

OR comments LIKE '%not\_enjoyed%'

OR comments LIKE '%not\_recommend%'

OR comments LIKE '%not\_beautiful%'

OR comments LIKE '%not\_responsive%'

OR comments LIKE '%not\_quite%'

OR comments LIKE '%not\_amazing%'

OR comments LIKE '%not\_convenient%'

THEN 'negative'

ELSE 'positive'

END AS feedback

FROM dbo.reviews)

SELECT DISTINCT feedback, count(\*) AS number\_of\_reviews

FROM cte

GROUP BY feedback

--Outcome:

A screenshot of a computer

Description automatically generated