VERSION 1.0: Trials in analysis are shown as is.
VERSION 1.1: Only working trials shown in analysis (01.09.2022)

Since Yearly data is required, need to union the individual 12 monthly tables to make one table,
Before attempting this, run the following checks.

1. Do all 12 monthly tables have same 'column name' arranged in same order ?
2. Do all the 12 monthly tables have same 'no. of columns' ?

An affirmative for both the above questions mean we can proceed.
In these 12 tables, all columns are similar and can be merged.
'union all' is used to prevent any loss of data.

```
DROP TABLE IF EXISTS [dbo].[Yearly_Data_Raw] -- To be invoked at start.
SELECT *
        INTO [dbo].[Yearly_Data_Raw]
FROM
(
SELECT *
FROM [dbo].[202108-divvy-tripdata] -- Aug21 data
UNION ALL
SELECT *
FROM [dbo].[202109-divvy-tripdata] -- Sep21 data
UNION ALL
SELECT *
FROM [dbo].[202110-divvy-tripdata] -- Oct21 data
UNION ALL
SELECT *
FROM [dbo].[202111-divvy-tripdata] -- Nov21 data
UNION ALL
SELECT *
FROM [dbo].[202112-divvy-tripdata] -- Dec21 data
UNION ALL
SELECT *
FROM [dbo].[202201-divvy-tripdata] -- Jan22 data
UNION ALL
SELECT *
FROM [dbo].[202202-divvy-tripdata] -- Feb22 data
UNION ALL
SELECT *
FROM [dbo].[202203-divvy-tripdata] -- Mar22 data
UNION ALL
SELECT *
FROM [dbo].[202204-divvy-tripdata] -- Apr22 data
UNION ALL
SELECT *
FROM [dbo].[202205-divvy-tripdata] -- May22 data
UNION ALL
SELECT *
FROM [dbo].[202206-divvy-tripdata] -- Jun22 data
UNION ALL
SELECT *
FROM [dbo].[202207-divvy-tripdata] -- Jul22 data
) data; -- Union all monthly tables into one master raw file.
-- 5,901,463 rows affected
```

Now, need to make data clean.
Good idea to make a temporary copy of this table so that this Raw union all'd file is not tampered with.

```
DROP TABLE IF EXISTS #Yearly_Raw_1 -- just a check.

SELECT *
        INTO #Yearly_Raw_1
FROM
        [dbo].[Yearly_Data_Raw];
```

Now we are free to work on temporary file Yearly_Raw_1

Steps taken to clean data

1. Data cleaning in each column involves checks for
    1. Unique / distinct values
    2. NULL values count.
    3. Any peculiarities in the column
    4. Check for spell errors and use of trim.
2. Check for relations between columns that seem to be impossible.
3. Clean the data with assumptions clearly labeled.

```
/*Checking for primary key (ride_id) being repeated in any component table.*/
        SELECT count(ride_id) - count(DISTINCT ride_id) -- In case number is greater than 0,
it means not all values are distinct.
        FROM #Yearly_Raw_1;
        -- 0 is output, so there is no repetition.

        SELECT ride_id -- Check for NULLs in ride_id
        FROM #Yearly_Raw_1
        WHERE ride_id IS NULL;
        -- no ride_id displayed, so this column is clean.

/* Checking for rideable_type column*/
        SELECT count(rideable_type) - count(DISTINCT rideable_type)-- Checking for
rideable_type column
        FROM #Yearly_Raw_1;
        -- 5901460 is output, which means that not all values are distinct.5901463-5901460 =
3 distinct values.

        SELECT rideable_type -- Checking rideable_type column
        FROM #Yearly_Raw_1
        GROUP BY rideable_type;
-- 3 categories are output electric_bike, classic_bike, docked_bike. No NULLs.

/*Checking for started_at and ended_at columns. Since both are date time columns, no point
in checking
for distinct values. Need to check for NULLS*/

        SELECT started_at,ended_at -- Checking for started_at and ended_at columns
        FROM #Yearly_Raw_1
        WHERE started_at IS NULL OR ended_at IS NULL;
        -- No NULL values in both columns.

-- To check if started_at and ended_at columns are within range decided.
        SELECT max(started_at), min(started_at), max(ended_at),min(ended_at)
        FROM #Yearly_Raw_1;
        --It is well within our range of analysis.
```

Now we checking 4 columns together since they are all varchar datatype columns.

1. start_station_name,
2. end_station_name,
3. start_station_id,
4. end_station_id

for

1. UNIQUE values
2. Check for any peculiarities other than NULL
3. NULL values.
4. Checking for unique values

```
SELECT DISTINCT start_station_name, count(*) -- Start_station column
FROM #Yearly_Raw_1
GROUP BY start_station_name;
-- 1382 distinct start_station_names

SELECT DISTINCT end_station_name, count(*) -- End_station_name column
FROM #Yearly_Raw_1
GROUP BY end_station_name;
-- 1397 distinct end_station_names

SELECT DISTINCT start_station_id, count(*) -- Start_station_id column
FROM #Yearly_Raw_1
GROUP BY start_station_id;
-- 1227 distinct start_station_id

SELECT DISTINCT end_station_id, count(*) -- End_station_id column
FROM #Yearly_Raw_1
GROUP BY end_station_id;
-- 1237 distinct end_station_names

SELECT DISTINCT member_casual, count(*) -- Member_casual column
FROM #Yearly_raw_1
GROUP BY member_casual;
-- 2 distinct values
```

In the problem statement, there is mention of ~600 docking stations but the distinct values in both end_station and start_station (id and name) is ranging between 1227 and 1397. There is some deviation from initial data.

On checking the start_station_name and end_station_name, there is no visible distinguishing factor in the names.

On checking the start_station_id and end_station_id, it is seen that there are various formats

1. station id with .0 as suffix to number.
2. Numeric of different lengths and aplhabetic station ids
3. First we remove the .0 from the start_station_id and end_station_id column. */

```
SELECT *-- Checking number of station_id with decimal point
FROM #Yearly_Raw_1
WHERE start_station_id LIKE '%.0';
--21,302 rows
```

```
        SELECT * -- Checking number of end_station_id with decimal point
        FROM #Yearly_Raw_1
        WHERE end_station_id LIKE '%.0';
        -- 19,295 rows

        UPDATE #Yearly_Raw_1 -- Updating the start_station_id removing the decimal points in
#Yearly_Raw_1
        SET
                start_station_id = CASE WHEN start_station_id LIKE '%.0'
                                                        THEN
replace(start_station_id,'.0','')
                                                        ELSE start_station_id END;
        -- all rows affected.

        UPDATE #Yearly_Raw_1 -- Updating the end_station_id removing the decimal points in
#Yearly_Raw_1
        SET
                end_station_id = CASE WHEN end_station_id LIKE '%.0'
                                                        THEN replace(end_station_id,'.0','')
                                                        ELSE end_station_id END;
        -- all rows affected.
```

Before proceeding with NULL values, we have to stop and consider the below.

Curious case of station id and their attributes:
The peculiarities are as mentioned below
1. Station ids have more than one station name .
2. Station ids have more than one latitude and longitude combination. In one instance, one station id had 16 latitude - longitude combinations.
In order to rationalize these oddities, a master table had to be created for each station using the station_id as a primary key with 3 other unique attributes- station name, latitude and longitude.

```
        DROP TABLE IF EXISTS #station_id -- check for any previous iteration and save union
to new temporary table #station_id
        SELECT * --Create a union of end station data and start station data since the same
stations are assumed to double up.
                INTO #station_id
        FROM
                (
                SELECT
                        end_station_id as station_id
                        , end_station_name as station_name
                        , end_lat as station_lat
                        , end_lng as station_lng
                FROM #Yearly_Raw_1
                UNION
                SELECT
                        start_station_id
                        , start_station_name
                        , start_lat
                        , start_lng
                FROM #Yearly_Raw_1
                ) st_id;
                -- Output 1,184,630 rows.
```

DELETE all NULL values in temporary table station_id since we are making a master table with all unique values.

```
        DELETE FROM #station_id /* Delete NULLs from the #station_id temp table*/
        WHERE station_id IS NULL OR station_name IS NULL or station_lat IS NULL or
station_lng IS NULL
                -- Output: 907 records out of 1,184,630 rows affected.
```

Each station seems to have a large number of lat and long combination. So, we can check average grouping by station_name and station_id to find max and min of co-ordinates (lat and lng) to see if we can make just one unique value without loss of information

```
        SELECT
                station_id
                , station_name
                , min(station_lat) AS min_lat
                , avg(station_lat) AS avg_lat
                , max(station_lat) AS max_lat
                , max(station_lat)-min(station_lat) AS diff_lat
                , min(station_lng) AS min_lng
                , avg(station_lng) AS avg_lng
                , max(station_lng) AS max_lng
                , max(station_lng)-min(station_lng) AS diff_lng
        FROM #station_id
        GROUP BY station_id, station_name
        ORDER BY diff_lng DESC;
        -- Output is 1424 unique station_id + station_name rows.
```

Observation: From the output, it is clear that Pawel Bialowas station name has diff_lat by almost 4.0 and diff_lng of 14.0 which is a big error geographically for the same station.
Remaining diff_lat and diff_lng is within 0.3.
Note: First decimal place is accuracy of 11.1 kms which is not great for a cycle station id. So, there is an issue with the recording for this location.

To continue with cleaning,
Assumption: Choosing avg(lat) and avg(long) as base latitude and longitude for each station.
This will replace the range of latitude and longitude to one unique value.
And, we move this data from a temporary table to a permanent table. After transfer, we delete the temporary table.

```
        -- Move #station_id table into a permanent table for unique station name values.
        DROP TABLE IF EXISTS [dbo].[station]-- check for dropping a previous iteration.
        SELECT * -- move to [dbo].[station]
                INTO [dbo].[station]
        FROM
                (SELECT
                        station_id
                        , station_name
                        , avg(station_lat) AS avg_lat
                        , avg(station_lng) AS avg_lng
                FROM #station_id
                GROUP BY station_id, station_name
                ) st_data;
        -- 1424 rows affected.
        DROP TABLE #station_id -- to save memory
```

There is a possibility that a combination of latitude co-ordinate and longitude co-ordinate can be combined to identify stations uniquely. In order to do this, we add an additional column to the newly created station table.

```
        /* Add a new column which combines lat and long*/
        ALTER TABLE [dbo].[station] --Add new column which could be used as a unique
identifier.
        ADD concatlatlng varchar(100);

        UPDATE [dbo].[station] /* Update concat string to concatlatlng column*/
        SET concatlatlng = concat(avg_lat,' ',avg_lng)
        /*Check to see if all concatlatlng are unique or not.*/

        SELECT DISTINCT concatlatlng
        FROM [dbo].[station]
        -- Out of 1424 only 1062 rows are distinct.
```

Unfortunately, not all stations have unique lat-long combinations, hence we can investigate this later.

Station id needs to be a distinct primary key for the station table.
Checking for distinct station ids in station table.

```
        /* Removing duplicate primary key station_id in [dbo].[station]*/
        SELECT DISTINCT station_id,count(*) AS counter
        FROM [dbo].[station]
        GROUP BY station_id
        ORDER BY count(*) DESC;
        --1243 rows with 156 station_id's having more than one station_name
```

In order to check and delete repetitions, we use a row number counter and delete where the counter is greater than one.

```
        -- To check for duplicate and adding a counter.
        SELECT
                station_id,
                station_name,
                concatlatlng,
                row_number() OVER (PARTITION BY station_id ORDER BY station_id ASC) counter
        FROM [dbo].[station]
        ORDER BY station_id;
        -- Shows the list of station_id which are repeated and by how much.

        -- To delete from the station table,
        Delete T
        From
                (
                Select
                                *,
                                Row_Number() Over(Partition By station_id order By
station_id) As RowNumber
                From [dbo].[station]) T
                Where T.RowNumber > 1
                -- 181 rows deleted from [dbo].[station]

        -- Checking number
        SELECT *
        FROM [dbo].[station]
                -- 1243 rows remaining with unique station_id. Deletion success.
```

Finally, we have only unique values in the station_id column of the master table.

Now, checking for any peculiarities in the station_id column.

```
-- Checking for numeric station_ids*/
        SELECT len(station_id) AS char_count,count(*) AS [Num of station_id with char_count]
        FROM [dbo].[station]
        GROUP BY len(station_id);
        /* This gives the split of char lengths which is a good indicator because there are
1 station_id with 2 characters,
        605 station_id with with 3 characters and 13 station_id with 4 characters which
matches the initial estimate of
        ~600 stations* mentioned in the problem statement. All the other station id's seem
to be added on later.*/

        SELECT station_id-- Checking nature of 3 character and 4 character length station_id
        FROM [dbo].[station]
        WHERE len(station_id) = 3 OR len(station_id) = 4;
        -- output is 618 rows

        SELECT ISNUMERIC(station_id),count(*) -- Checking now for ISNUMERIC (station_id)
        FROM [dbo].[station]
        WHERE len(station_id) = 3 OR len(station_id) = 4
        GROUP BY ISNUMERIC(station_id);
        -- 610 station_ids are without alphabets in station_id, 8 station_ids have
alphabets.

        SELECT station_id--To isolate these 8 station_ids for further investigation.
        FROM [dbo].[station]
        WHERE (len(station_id) = 3 OR len(station_id) = 4) AND ISNUMERIC(station_id) = 0;
        -- Nothing peculiar in these station_id. Checking with the full data #Yearly_Raw_1.

        -- Checking for number of rows in main file for occurence of these stations 3 char
or 4 char + alphanumeric
        SELECT *
        FROM #Yearly_Raw_1
        WHERE (len(end_station_id) = 3 OR len(end_station_id) = 4)
                AND ISNUMERIC(end_station_id) = 0;
        -- 42,892 rows in total. No specific reason to remove from data.

        SELECT station_id -- Checking for length of station_id = 5
        FROM [dbo].[station]
        WHERE len(station_id) = 5;
        -- 313 rows output
        SELECT ISNUMERIC(station_id),count(*) --Checking for numeric alone in length 5.
        FROM [dbo].[station]
        WHERE len(station_id) = 5
        GROUP BY ISNUMERIC(station_id);
        -- All 313 rows are numeric.

        SELECT station_id -- Checking for length of station_id = 6
        FROM [dbo].[station]
        WHERE len(station_id) = 6;
        -- 24 rows output

        SELECT ISNUMERIC(station_id),count(*) --Checking for isnumeric
        FROM [dbo].[station]
        WHERE len(station_id) = 6
        GROUP BY ISNUMERIC(station_id);
        /* 21 rows are alphabetic, 3 rows are numeric. After checking nature of 24 rows,
there does not seem
        to be a problem. */

        /* Checking for all station_ids with length >=6 which is not numeric*/
```

```
        SELECT station_id
        FROM [dbo].[station]
        WHERE len(station_id)>=6 AND ISNUMERIC(station_id)=0
        -- Output: Total 308 rows, with different categories.
                SELECT count(*) AS TA_KA-- GROUP BY TA and KA
                FROM [dbo].[station]
                WHERE len(station_id)>=6 AND ISNUMERIC(station_id)=0 AND (station_id LIKE
'TA%' OR station_id LIKE 'KA%')
                --269 rows are with TA or KA prefix.
                        SELECT --Checking TA types in main data.
                                start_station_id
                                , end_station_id
                                , started_at
                                , ended_at
                                , DATEDIFF(MINUTE,started_at,ended_at)
                        FROM #Yearly_Raw_1
                        WHERE end_station_id LIKE 'TA%'
                                        OR start_station_id LIKE 'TA%';
                        -- Output is 3,086,228 rows out of 5,901,463 rows. Nothing special
found in this data. Cannot ignore.
                        SELECT -- Checking for KA types in main data
                                start_station_id
                                , end_station_id
                                , started_at
                                , ended_at
                                , DATEDIFF(MINUTE,started_at,ended_at)
                                , start_station_name
                                , end_station_name
                        FROM #Yearly_Raw_1
                        WHERE end_station_id LIKE 'KA%' OR start_station_id LIKE 'KA%';
                        -- Output is 893,281 rows
                        -- TA and KA cannot be ignored. Now checking the remaining (non TA
and non KA)
                SELECT count(*) AS TA_KA-- Checking for Non TA and Non KA prefixes for
station_id
                FROM [dbo].[station]
                WHERE len(station_id)>=6 AND ISNUMERIC(station_id)=0 AND station_id NOT LIKE
'TA%' AND station_id NOT LIKE 'KA%'
                -- 39 are non TA_KA prefix type station_id
                SELECT station_id-- Checking nature of station_ids without TA_KA prefixes
where char length >=6
                FROM [dbo].[station]
                WHERE len(station_id)>=6
                        AND ISNUMERIC(station_id)=0
                        AND station_id NOT LIKE 'TA%'
                        AND station_id NOT LIKE 'KA%'
                        /* Various catagories can be noted here. Predominant are with texts
- %Charging%,%Char%,%Warehouse%,%checking%
                        ,%repair% which are most probably service stations*/
                        SELECT station_id-- Checking for char%,%charging%
                        FROM [dbo].[station]
                        WHERE len(station_id)>=6
                                AND station_id LIKE 'char%'
                                OR station_id LIKE '%charging%'
                        -- 9 out of 39 non TA_KA types are charging stations.
                                SELECT *-- Checking in main data for some clues.
                                FROM #Yearly_Raw_1
                                WHERE end_station_id LIKE 'char%' OR start_station_id LIKE
'%charging%'
                                /* 34,042 rows. Nothing special in the data at first sight
to exclude from our data since,
                                ride ids start at charging point but end else where. No
othere distinction at first glance.*/
```

```
                                SELECT *-- Checking in main data also for '%test%'
                                FROM #Yearly_Raw_1
                                WHERE end_station_id LIKE '%test%' OR start_station_id LIKE
'%test%'
                                /*1,809 rows. Filtered data suggests that most are between
test stations and base. There are some values where the start station
                                is Hubbard bike checking and end station is NULL. These
instances can be deleted. But there are instances of member as well as casual */
                                        DELETE -- Deleting rows from main file as per above.
                                        FROM #Yearly_Raw_1
                                        WHERE start_station_id = 'Hubbard Bike-checking
(LBS-WH-TEST)' AND end_station_name IS NULL;
                                        -- 628 rows affected.
```

There are other categories of station_id with keywords like DIVVY, %charg%,%rep%,%base%,%test%,%mobil% which have been used by both casual and members. So, it can be put as a categorised list for later analysis.
For now, we have 1243 unique station ids with lat and lng defined, which can be updated into the working main temporary table.

Continuing the cleaning process with NULL values in start_station_id or start_station_name that can be recovered.

```
SELECT start_station_name,start_station_id --Checking for any rows with valid
start_station_name but NULL start_station_id
        FROM #Yearly_Raw_1
        Where start_station_name IS NULL and start_station_id IS NOT NULL
        OR start_station_name IS NOT NULL and start_station_id IS NULL
        -- Found two records- 20215 and WL-008
        UPDATE #Yearly_Raw_1-- update valid start_station_names
        SET
                start_station_name = CASE
                                                        WHEN start_station_id = '20215' AND
start_station_name IS NULL THEN 'Hegewisch Metra Station'
                                                        WHEN start_station_id = 'WL-008' AND
start_station_name IS NULL THEN 'Clinton St & Roosevelt Rd'
                                                        ELSE start_station_name
                                                         END;
        /*Updated all records successfully- 5,900,835 rows.
```

Do the same process for end_Station_id and end_station_name columns.

```
        SELECT end_station_name,end_station_id-- Checking for any row with valid
end_station_name but NULL end_station_id or viceversa.
        FROM #Yearly_Raw_1
        Where end_station_name IS NULL and end_station_id IS NOT NULL
        OR end_station_name IS NOT NULL and end_station_id IS NULL
        -- Found zero records
```

Now we check for combinations where NULL is avaialble in both station id and station name columns of main temporary table.

```
-- Now proceeding to NULL NULL combinations in start_station.
        SELECT start_station_name,start_station_id -- checking for start station ids
        FROM #Yearly_Raw_1
        Where start_station_name IS NULL and start_station_id IS NULL
        -- 860,784 rows are NULL NULL
```

Checking for a possible commonality by viewing the data,

```
        SELECT DISTINCT rideable_type-- Checking for cause of this abberation.
        FROM #YEarly_Raw_1
        Where start_station_name IS NULL and start_station_id IS NULL
```

It is found that start station name and start station id becomes NULL only in electric bikes. No other commonality. Now continuing the check for the end_station_name and end_station_id NULL NULL combinations as below.

```
SELECT end_station_name,end_station_id
        FROM #Yearly_Raw_1
        Where end_station_name IS NULL and end_station_id IS NULL;
         -- 919,896 rows are NULL NULL
```

The number of NULL NULL in start station and end station is almost 1.5 million rows. Since, we cannot reliably use the latitude and longitude in the main file as a unique identifier for the station id, we are forced to delete such instances.

Since it is a delete operation, better to be sure before deleting from main. So, saving our current table in another temporary table. In case of success, we remove this temporary table.

```
        SELECT * INTO #Yearly_Raw2
        FROM #Yearly_Raw_1
        --5,900,835 rows affected.
```

Now, we can delete the NULL NULL combinations

```
--DElete operation in main Raw file.
        DELETE FROM #Yearly_Raw_1 -- Delete start NULL NULL
        WHERE start_station_name IS NULL AND start_station_id IS NULL;
        -- 860,784 rows deleted.
        DELETE FROM #Yearly_Raw_1 -- Delete end NULL NULL
        WHERE end_station_name IS NULL AND end_station_id IS NULL;
        --410,820 rows deleted.

        /*Deleting successful, now dropping temp file created.*/
        DROP Table #Yearly_Raw2 -- With thanks. To save memory.

        SELECT count(*)
        FROM #Yearly_Raw_1;
        -- New no. of rows is 4,629,231 rows.
```

Need to save current temporary table into a new permanent table so that I can continue work from where I left.

```
--Saving temporary. Need to change the name of column rev_lat and rev_lng to s_lat and s_lng
DROP TABLE IF EXISTS [dbo].[Yearly_Raw_28.08.22]-- to check off previous iterations of
table.

SELECT * -- Creating new table to save temporary table.
INTO [dbo].[Yearly_Raw_28.08.22]
FROM #Yearly_Raw_1;-- 4,629,231 rows affected.
```

```
DROP Table IF EXISTS #Yearly_Raw_1 -- Dropping temporary table.
```

NEW DAY: Creating new temp file for working

```
SELECT * INTO #Yearly_Raw_1
FROM [dbo].[Yearly_Raw_28.08.22]
-- 4629231 rows affected.
```

Now, we add 4 columns to the temporary table from the master station table. The _ r suffix is for 'revised'.

```
ALTER TABLE #Yearly_Raw_1
ADD s_lat_r float, s_lng_r float, e_lat_r FLOAT, e_lng_r FLOAT;

-- Updating s_lat_r
UPDATE #Yearly_Raw_1 -- Update s_lat_r in temporary table.
SET s_lat_r =  st.avg_lat
FROM #Yearly_Raw_1 yr
        INNER JOIN [dbo].[station] st
        ON yr.start_station_id = st.station_id;
-- 4,629,231 rows affected. (3 min taken for execution.)
UPDATE #Yearly_Raw_1 -- Update s_lat_r in temporary table.
SET s_lng_r =  st.avg_lng
FROM #Yearly_Raw_1 yr
        INNER JOIN [dbo].[station] st
        ON yr.start_station_id = st.station_id;
-- 4,629,231 rows affected. (2 min taken for execution.)
UPDATE #Yearly_Raw_1 -- Update e_lat_r in temporary table.
SET e_lat_r =  st.avg_lat
FROM #Yearly_Raw_1 yr
        INNER JOIN [dbo].[station] st
        ON yr.end_station_id = st.station_id;
-- 4,629,231 rows affected. (1 min 37 s taken for execution.)
UPDATE #Yearly_Raw_1 -- Update e_lng_r in temporary table.
SET e_lng_r =  st.avg_lng
FROM #Yearly_Raw_1 yr
        INNER JOIN [dbo].[station] st
        ON yr.end_station_id = st.station_id;
-- 4,629,231 rows affected. (1 min 34 s for execution)
```

Adding two more columns for start station name and end station name.

```
/* Now create two new columns for s_name_r and e_name_r for start_station_name and
end_station_name. */
ALTER TABLE #Yearly_Raw_1 -- s_name_r (revised station name for start) and e_name_r (revised
station name for end)
ADD s_name_r varchar(100), e_name_r varchar(100)

/* Updating two new columns from master station table. */
UPDATE #Yearly_Raw_1-- Updating s_name_r
SET s_name_r = st.station_name
FROM #Yearly_Raw_1 yr
        INNER JOIN [dbo].[station] st ON st.station_id = yr.start_station_id
        -- 4,629,231 rows affected. (2min10s for execution)
UPDATE #Yearly_Raw_1-- Updating e_name_r
SET e_name_r = st.station_name
FROM #Yearly_Raw_1 yr
```

```
        INNER JOIN [dbo].[station] st ON st.station_id = yr.end_station_id
        -- 4,629,231 rows affected. (2 min34s for execution)
```

Now, we can drop the raw columns and keep only revised columns with non conflicting values.

```
ALTER TABLE #Yearly_Raw_1--Raw columns to be dropped are start_station_name,
end_station_name,start_lat, start_lng,end_lat,end_lng
DROP COLUMN start_station_name, end_station_name, start_lat, start_lng, end_lat, end_lng;
-- columns deleted.
```

Now copy from temporary table to permanent table - Yearly_Clean

```
DROP TABLE IF EXISTS [dbo].[Yearly_Clean]-- to check for past iterations of table.

SELECT * INTO [dbo].[Yearly_Clean] -- insert table.
FROM #Yearly_Raw_1; -- 4,629,231 rows affected and verified.

DROP TABLE IF EXISTS #Yearly_Raw_1 -- Deleting temp table with thanks.
```

Checking for logical consistency of two columns started_at and ended_at, recopy the Yearly_Clean data to a temporary file.

```
SELECT * INTO #Yearly_Raw_1
FROM [dbo].[Yearly_Clean];

SELECT datediff(MINUTE,started_at,ended_at) AS ride_duration-- ride duration needs to be
checked for logic.
FROM #Yearly_Raw_1
ORDER BY ride_duration;
```

75 nos. instances of ride_duration is less than zero, which is faulty. So, need to delete these.
Max duration is found to be 41,629 minutes which is also faulty.
Assumption: 16 hours of ride per day is considered valid. Remaining are to be removed to prevent any outlier effect.

Deleting the rides that do not conform to above.

```
DELETE
        FROM #Yearly_Raw_1
WHERE
        datediff(MINUTE,started_at,ended_at) <=0 -- 40,454 rows affected
        OR
        datediff(MINUTE,started_at,ended_at) >960 -- 1564 rows affected
```

Storing this revised temporary table into Yearly_Clean.

```
DROP TABLE IF EXISTS [dbo].[Yearly_Clean] -- making a fresh copy and removing old instances.

SELECT * INTO [dbo].[Yearly_Clean]
FROM #Yearly_Raw_1 -- 4,587,213 rows affected.

DROP TABLE IF EXISTS #Yearly_Raw_1 -- with thanks.
```

In order to make analysis easy, three more columns are added to account for monthly, week day and hour of the day time slots next to each ride.

```
SELECT * INTO #Yearly_Raw_1
FROM [dbo].[Yearly_Clean]

ALTER TABLE #Yearly_Raw_1
ADD
        months varchar(50)
        , weekday varchar(50)
        , hourly varchar(50);

UPDATE #Yearly_Raw_1
SET
        months=datename(MONTH,started_at)
        , weekday = datename(WEEKDAY,started_at)
        , hourly = datename(HOUR,started_at);

DROP TABLE IF EXISTS [dbo].[Yearly_Clean]

SELECT * INTO [dbo].[Yearly_Clean]
FROM #Yearly_Raw_1

DROP TABLE IF EXISTS #Yearly_Raw_1;
```

We finally have the cleaned data in permanent table Yearly_Clean marking the end of the process phase.