2. Technical support ticket analytics and dashboard Part 1

Assigned To	Data understanding and cleaning
Status	In progress

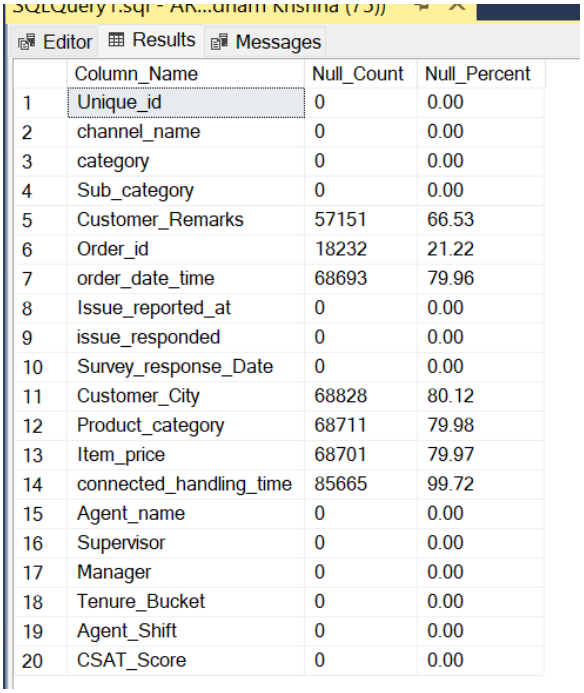
```
create database project
--*************************************************************************
******************--
-- View the table before proceeding
select * from support_ticket_data
--*************************************************************************
******************--
-- check total number of rows  -- 85907
select COUNT(*) as Total_Rows from support_ticket_data

-- check the shape of column
select * from INFORMATION_SCHEMA.COLUMNS where TABLE_NAME = 's
upport_ticket_data'
--*************************************************************************
******************--
-- check the data types of the table
select
TABLE_NAME,
COLUMN_NAME,
DATA_TYPE,
IS_NULLABLE
from information_schema.columns
where table_name = 'support_ticket_data'
--*************************************************************************
******************--
-- check for missing values
/*
We checked in information_Schema and found that there are columns wher
e nullable = Yes
```

```
We will primarily check those columns for missing values and counts.
However, if other columns where nullable = No, that means they do not hav
e null values but we still need to check them because
they can have values which make no sense for analysis.
*/

SELECT
    SUM(CASE WHEN Customer_Remarks IS NULL THEN 1 ELSE 0 END) AS
null_Customer_Remarks,
    SUM(CASE WHEN Order_id IS NULL THEN 1 ELSE 0 END) AS null_Order_i
d,
    SUM(CASE WHEN order_date_time IS NULL THEN 1 ELSE 0 END) AS null
_Order_time,
    SUM(CASE WHEN Customer_City IS NULL THEN 1 ELSE 0 END) AS null_
customer_city,
    SUM(CASE WHEN Product_category IS NULL THEN 1 ELSE 0 END) AS nu
ll_product_category,
    SUM(CASE WHEN Item_price IS NULL THEN 1 ELSE 0 END) AS null_Item
_price,
    SUM(CASE WHEN connected_handling_time IS NULL THEN 1 ELSE 0 EN
D) AS null_connected_handling_time
FROM support_ticket_data;

/*
The above script is okay when we have fewer columns which are nullable b
ut what if we want to check all columns
Therefore we follow the below dynamic code to check null values for all col
umns.
*/

/*
Here, we are declaring two variables
@table_name , @sql.  We keep sql variable empty as of now.
However, we will append all the information in sql variable using @sql +=
*/

DECLARE @table_name NVARCHAR(128) = 'support_ticket_data';
DECLARE @sql NVARCHAR(MAX) = '';
```

```
SELECT @sql +=
   'SELECT ''' + COLUMN_NAME + ''' AS Column_Name, ' +
   'COUNT(*) - COUNT([' + COLUMN_NAME + ']) AS Null_Count, ' +
   'CAST(ROUND((COUNT(*) - COUNT([' + COLUMN_NAME + ''])) * 100.0 /
COUNT(*),2) AS DECIMAL(5,2)) AS Null_Percent ' +
   'FROM [' + @table_name + '] UNION ALL '
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = @table_name;

-- Remove the last UNION ALL
SET @sql = LEFT(@sql, LEN(@sql)-10);

-- Run till here to see how the format looks like (valid spacing)
PRINT @sql

-- Run the final query
EXEC sp_executesql @sql;
--************************************************************************
******************--
-- Let us check the distinct count for each column.

DECLARE @data_table NVARCHAR(128) = 'support_ticket_data';
DECLARE @unique_table NVARCHAR(MAX) = '';

SELECT @unique_table +=
   'SELECT ''' + COLUMN_NAME + ''' AS Column_Name, ' +
   'COUNT(DISTINCT [' + COLUMN_NAME + ']) AS Unique_Count ' +
   'FROM [' + @data_table + '] UNION ALL '
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = @data_table;

-- Trim the last UNION ALL
SET @unique_table = LEFT(@unique_table, LEN(@unique_table) - 10);

-- Run till here to see how the format looks like (valid spacing)
PRINT @unique_table
```

```
-- Run the query
EXEC sp_executesql @unique_table;
--*********************************************************************
******************--
```

So far we found out that we have missing values, we will need to work on them later stages. (attaching screenshot of missing values count from each column)



Editor | Results | Messages

	Column_Name	Null_Count	Null_Percent
1	Unique_id	0	0.00
2	channel_name	0	0.00
3	category	0	0.00
4	Sub_category	0	0.00
5	Customer_Remarks	57151	66.53
6	Order_id	18232	21.22
7	order_date_time	68693	79.96
8	Issue_reported_at	0	0.00
9	issue_responded	0	0.00
10	Survey_response_Date	0	0.00
11	Customer_City	68828	80.12
12	Product_category	68711	79.98
13	Item_price	68701	79.97
14	connected_handling_time	85665	99.72
15	Agent_name	0	0.00
16	Supervisor	0	0.00
17	Manager	0	0.00
18	Tenure_Bucket	0	0.00
19	Agent_Shift	0	0.00
20	CSAT_Score	0	0.00

Columns with missing values - Customer_Remarks , Order_ID, Order_date_time, Customer_city , Product_category, Item_price, connected_handling_time

```
-- Let us work on date time columns.
select
COLUMN_NAME,
DATA_TYPE
from INFORMATION_SCHEMA.COLUMNS
where TABLE_NAME = 'support_ticket_data' and DATA_TYPE = 'datetime2'

/*
columns with date and timestamps:
order_date_time
Issue_reported_at
issue_responded
```

```
Survey_response_Date
*/
-- checking for null values in date time column
SELECT COUNT(*) FROM support_ticket_data WHERE order_date_time IS N
ULL;

SELECT
    SUM(CASE WHEN order_date_time IS NULL THEN 1 ELSE 0 END) AS null
_order_date_time,
    SUM(CASE WHEN issue_reported_at IS NULL THEN 1 ELSE 0 END) AS nu
ll_issue_reported,
    SUM(CASE WHEN issue_responded IS NULL THEN 1 ELSE 0 END) AS nul
l_issue_responded,
    SUM(CASE WHEN Survey_response_Date IS NULL THEN 1 ELSE 0 END)
AS null_survey_date
FROM support_ticket_data;

-- Null_order_date_time has 68693 missing values.

-- checking minimum and maximum values of these 4 date time columns.
SELECT
    MIN(order_date_time) AS Min_order_date_time,
    MAX(order_date_time) AS Max_order_date_time,
    MIN(Issue_reported_at) AS Min_Issue_reported_at,
    MAX(Issue_reported_at) AS Max_Issue_reported_at,
    MIN(issue_responded) AS Min_issue_responded,
    MAX(issue_responded) AS Max_issue_responded,
    MIN(Survey_response_Date) AS Min_Survey_response_Date,
    MAX(Survey_response_Date) AS Max_Survey_response_Date
FROM support_ticket_data

/*
What if we had more date time columns instead of just 4 ?
The above method is not practical and optimized way of writing code.
We will follow our similar dynamic method that we followed above.
*/
-- checking minimum and maximum values for 4 date time columns.
DECLARE @table_name NVARCHAR(128) = 'support_ticket_data';
```

```
DECLARE @date_time NVARCHAR(MAX) = '';

SELECT @date_time +=
    'SELECT ''' + COLUMN_NAME + ''' AS Column_Name, ' +
    'MIN([' + COLUMN_NAME + ']) AS Min_Value, ' +
    'MAX([' + COLUMN_NAME + ']) AS Max_Value ' +
    'FROM [' + @table_name + '] UNION ALL '
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = @table_name
  AND DATA_TYPE IN ('datetime', 'datetime2', 'smalldatetime');

-- Trim the last UNION ALL
SET @date_time = LEFT(@date_time, LEN(@date_time) - 10);

-- Run the dynamic query
EXEC sp_executesql @date_time



/*
We need not worry about the format of the date time columns.
Reason being that they are already set in datetime2 data type.
All the rows will be set to same format automatically.
*/
```

```
-- check if timestamp sequence logic makes sense.
/*
Check if any issue is reported before the ordered date time.
If yes, then find out reason.
*/
SELECT COUNT(*)
FROM support_ticket_data
WHERE order_date_time IS NOT NULL
  AND issue_reported_at < order_date_time
/*checked and can see result shows count of 58 rows, that means for 58 ti
ckets, the issue was reported first and then order was placed */

SELECT *
FROM support_ticket_data
```

```
WHERE order_date_time IS NOT NULL
  AND issue_reported_at < order_date_time
/*
On further investigation and at a glance looked like it is a format issue for or
der time column but it is not possible.
Reason: all date time columns are set to datetime2 data type, so the format
for all, is normalised, as mentioned earlier.
After looking at the categories and sub categories, looks like these could b
e pre order enquiries.
Let's go ahead and group by sub categories ot understand more.
*/
-- group by sub category.
SELECT
    category,
    Sub_category,
    COUNT(*) AS tickets_before_order_count
FROM support_ticket_data
WHERE order_date_time IS NOT NULL
  AND issue_reported_at < order_date_time
GROUP BY  category, Sub_category
ORDER BY tickets_before_order_count DESC

/*
Looking at the table
Reverse Pickup Enquiry - a possible valid pre order query.
Few sub_categories like delayed and return requests are suspcious but pos
sible exception cases.
You cannot complain about delay or return requests until you have placed a
n order.
However,there can be several exception cases on customer side or system
recording side.
Either I can flag them into Suspicious and Normal or I can leave them as it i
s.
*/
Select
*,
CASE
  WHEN  issue_reported_at < order_date_time AND Sub_category IN ('Delay
```

```
ed', 'Return request') THEN 'Suspicious'
  else 'Normal'
END AS reported_before_order_flag
from support_ticket_data
--**********************************************************************
******************--
-- lets work on missing values on order_date_time , we have 68693 missing
values in that column
-- we will check if it is order_date_time missing or other columns as well.
SELECT
  SUM(CASE WHEN order_date_time IS NULL AND issue_reported_at IS NO
T NULL THEN 1 ELSE 0 END) AS missing_order_but_reported_present,
  SUM(CASE WHEN order_date_time IS NULL AND issue_responded IS NOT
NULL THEN 1 ELSE 0 END) AS missing_order_but_responded_present,
  SUM(CASE WHEN order_date_time IS NULL AND issue_reported_at IS NU
LL AND issue_responded IS NULL THEN 1 ELSE 0 END) AS all_three_missin
g
FROM support_ticket_data;

/*
Only order_date_time missing, no other date time column has missing rows.
We will leave this as null as of now and not work on this because order_dat
e_time column has no much relationship with our business problem.
*/
--**********************************************************************
******************--
/*
We have missing values in other columns as well.
customer remarks
order id
order date time
customer city
product_category
item price
connected handling time
*/

select * from support_ticket_data
```

```
-- The checks for customer remarks are following:
-- we need to remember that not all customers leave remarks or feedbacks
-- We will check which sub category customers leaves remarks most of the time.
select
   Sub_category,
   count(*) as customer_remark_count
from
support_ticket_data
where Customer_Remarks is not null
group by Sub_category
order by customer_remark_count desc
/*
Can be seen that customers are really vocal about their pickup enquiry, del ays and order status.
Logistics and expectations are areas where customer have left remark, cou ld be a positive or negative.
*/

-- Let's check Avg CSAT score when CSAT score was left vs not left.
select
   case
      when Customer_Remarks is not null then 'Yes Remarks'
      else 'No Remarks'
   end as Remark_Status,
   COUNT(*) as ticket_count,
   AVG(csat_score * 1) as avg_csat_score
from
support_ticket_data
group by
case
      when Customer_Remarks is not null then 'Yes Remarks'
      else 'No Remarks'
end
-- There is no bias in CSAT average whether the customer has given remar k or not.
/*
There is no action that has to be taken as of now for missing values in cust
```

```
omer_remark column.
*/
--**********************************************************************
******************--
-- We will now work on order_id
-- We observed that when order_id was null then order_date_time was also
null
-- A check to confirm the same:
select
 SUM(case when order_id is null then 1 else 0 end) as null_order_id_count,
 SUM(case when order_id is null and order_date_time is null then 1 else 0 e
nd) as null_order_count
from
support_ticket_data
-- We clearly can see that total null count for order id is 18232 and total null
count for order id and order date time is also 18232.
-- This explains that no order id == no orders placed.

-- It could be possible that there is no order placed so there is no order id a
nd the cust has reached out with general enquiries.
-- Let us check what all category or sub category enquiry cust has come u
p with when order id is null.
select
   category,
   sub_category,
   count(*) as ticket_count
from
support_ticket_data
where Order_id is null
group by category, Sub_category
order by category, ticket_count

-- No action required as of now on missing values for column order_id
--**********************************************************************
******************--
-- customer_city
/*
If there is no customer city is mentioned then it could be possible that
```

```
customer details are not filled properly.
customer is a new customer or has not registered yet.
customer has not contacted from registered contact.
*/

select
   *
from
support_ticket_data
where Customer_City is null
-- checking whcih category/sub_category queries are recieved with no loc
ation information
select
   Sub_category,
   COUNT(*) as ticket_count
from
support_ticket_data
where Customer_City is null
group by Sub_category
order by ticket_count desc

-- lets check which channel queries have been received with no location d
etails of customer.
SELECT
   channel_name,
   COUNT(*) AS missing_city_query_count
FROM support_ticket_data
WHERE Customer_city IS NULL
GROUP BY channel_name
/*
Inbound calls ticket count was highest with no city locations
Then outbound and follows Email
*/

-- check for missing customer city and missing product_category.
select
 SUM(case when Customer_City is null then 1 else 0 end) as null_cust_city_
count,
```

```
 SUM(case when Customer_City is null and Product_category is null then 1
else 0 end) as null_city_product_count
from
support_ticket_data
/*
Missing city count is 68828 and missing city count with missing product ca
tegory is 68693 , difference is of 135 rows.
That means for missing 68828 cities there are 135 rows which have produc
t category right
*/

-- Replacing null/missing city values with 'Unknown'

select
   *,
   case
      when Customer_City is null then 'Unknown'
      else Customer_City
   end as customer_City_filled,
   COUNT(*) as total_tickets
from
support_ticket_data
group by
case
      when Customer_City is null then 'Unknown'
      else Customer_City
end
order by total_tickets desc


select
   *,
   case
      when Customer_City is null then 'Unknown'
      else Customer_City
   end as customer_city_filled
from support_ticket_data
```

```
/*
In this case above we have temporarily made changes that any missing city values will be replaced with unknown.
Later if needed we can create a view out of this.
Rest no action needs to be taken on this column as of now.
*/
--*****************************************************************************************--
-- Product Category:

select
*
from
support_ticket_data
where Product_category is null

select distinct product_category from support_ticket_data

SELECT Sub_category, COUNT(*) AS missing_product_category
FROM support_ticket_data
WHERE Product_Category IS NULL
GROUP BY Sub_category
ORDER BY missing_product_category DESC;

select
   Product_category,
   case
      when Product_category is null then 'Unknown'
      else Product_category
   end as product_category_filled
from support_ticket_data

/*
As we checked, it is really hard to identify if the missing product category is associated to non product queries.
However, when we checked what sub categories had more missing product category then it looks like was more associated with logistics query.
For now just filling it with 'Unknown' at missing places temporarily.
```

```
Later we can create a view if needed. Rest no actions need to be taken as o
f now.
*/
--**********************************************************************
******************--
-- Item PRice
select
*
from
support_ticket_data
where Item_price is null

SELECT
  COUNT(*) AS price_missing_with_category
FROM support_ticket_data
WHERE Item_Price IS NULL AND Product_Category IS NULL;


SELECT Sub_category, COUNT(*) AS missing_price_count
FROM support_ticket_data
WHERE Item_Price IS NULL
GROUP BY Sub_category
ORDER BY missing_price_count DESC;

SELECT Product_category, COUNT(*) AS missing_price_count
FROM support_ticket_data
WHERE Item_Price IS NULL
GROUP BY Product_category
ORDER BY missing_price_count DESC;

/*
Even in this case of item price, we see that when product category is missi
ng then item price is also missing.
Again it looks like these are not purchases rather these are enquiries.
For the dashboard we can go ahead and fill them 0 or unknown depending
on the logic.
*/
```

```
-- connected handling time
select
*
from
support_ticket_data
where connected_handling_time is null

-- check which channel has high count of missing handling time
SELECT channel_name, COUNT(*) AS missing_connected_handling_time
FROM support_ticket_data
WHERE Connected_Handling_Time IS NULL
GROUP BY channel_name
ORDER BY missing_connected_handling_time DESC;
-- Interestingly , inbound has more numbers of missing handling time.
-- Ideally we can expect no handling time shown for chat, emails, IVR as th
ey might not track handling time.

SELECT Sub_category, COUNT(*) AS missing_handling_time
FROM support_ticket_data
WHERE Connected_Handling_Time IS NULL
GROUP BY Sub_category
ORDER BY missing_handling_time DESC

/*
We will compute the missing values later to either unknown or zero for das
hboard.
*/
```

```
-- Lets go ahead and create a staging table
/*
Reason: I am creating this staging table as I will be using this ahead in Phas
e 2.
I want the columns to be renames in standard and consistent format.
As we also plant to use it further to create visuals in Power BI and writing D
AX Data Analysis Expressions.
The staging table will be created using CTAS Create Table As Select
*/
```

```
select top 5 * from support_ticket_data
-- creating CTAS with name stg_support_tickets
select
    Unique_id as ticket_id,
    channel_name,
    category,
    Sub_category as sub_category,
    Customer_Remarks as customer_remarks,
    Order_id as order_id,
    order_date_time as ordered_date_time,
    Issue_reported_at as issue_reported_date_time,
    issue_responded as issue_responded_date_time,
    Survey_response_Date as survey_response_date_time,
    coalesce(Customer_City, 'Location Unknown') as customer_city,
    coalesce(Product_category, 'Product Unknown') as product_category,
    Item_price as item_price,
    connected_handling_time as connected_handling_duration_time,
    Agent_name as agent_name,
    Supervisor as supervisor,
    Manager as manager,
    Tenure_Bucket as tenure,
    Agent_Shift as agent_shift,
    CSAT_Score as csat_score
into
stg_support_tickets
from
support_ticket_data

-- check if columns are now standardized and consistent, also we have fille
d few missing values in customer city and product category.

select * from stg_support_tickets

--
exec sp_help stg_support_tickets
```

```
select * from INFORMATION_SCHEMA.COLUMNS
where table_name = 'stg_support_tickets'
```