Individual Report

Collaborative Application Development (CSC-40038)

1 Introduction

This report provides an overview and individual contributions to the Collaborative Application Development Group Project, which was accomplished to meet the client's requirements. Our client is the CEO of a company, BookMeIn2, that organizes and promotes conferences and related exhibitions. His goal was to see if the data analysis would help make his events more successful, particularly in terms of selling sponsorship.

The traditional model for this industry is to rent a venue, invite subject matter experts to speak at the conference, and then advertise for attendees. The conference may be funded through ticket sales or, more likely, by sponsors who are given an exhibition stand from which they can interact with the attendees. Over the last year, the Pandemic has forced a change in the existing business model. In practice, bringing people together and having them sit in a hall while the speakers perform on stage is impossible. During the pandemic, they have shifted to a virtual conference and exhibition model in which everyone is separated, and technology is used to bring people together. Recently, they have introduced hybrid conferences and exhibitions, where some people attend in person while others attend remotely. This has resulted in both positive and negative outcomes, particularly for sponsors looking to meet with potential customers. One of the positive outcomes is the amount of information that they can collect about the event, its speakers, exhibitors, and, most importantly, its attendees. The big question is what they should record and what they can do with this information. As conference organizers, they need to know the answers to a few important questions too.

Our goal is to answer a few of the relevant questions that the conference organizers are looking for while also analyzing the massive amounts of data they are collecting about the events to identify trends and patterns that will aid in attracting and retaining attendees as well as Obtaining and retaining sponsors.

2 Individual Contribution

Our group, team-4 consisted of 5 individuals, our goal was to analyze and present the most important business insights for the client. We interacted with team members and came to know everyone's domain knowledge and experience level. So initially we divided the roles based on technical and domain knowledge. I was responsible for handling the visualization part of the data. Two teammates decided to build SQL Knowledge to get a healthy analysis of the data. The remaining two agreed to check generic Data analysis process. I did learn visualization tools and techniques for the sake of the project and shared the python code snippet with the team (Figure 1).Since it is a novice experience in doing a data analysis project, later we all decided to take Data analyst role, so that we can all learn the end-to-end process. The following are questions that the conference organizers are seeking:

1. How many delegates registered and how many attended?
2. What was the split between in-person and virtual attendees?
3. When did delegates attend? This may be difficult as we only record the last time they logged on
4. How long did virtual delegates attend for? Again, may be difficult as we can not track when they log off. Yet we do know when they were watching a live session.

5. Did anyone attend in-person and then log on later to view the information and or network?
6. How were the chat functions used?
7. What use was made of the messaging and networking functions, how successful was it?
8. How many exhibition stands were visited, for how long?
9. Which sessions were popular and why?
10. How many and what type of questions asked of the speakers by virtual attendees?

Code snippet to connect mysql DB

```
In [1]: #If mysql.connector package is missing, install it using below pip command in the console
        #pip install mysql-connector-python
        import mysql.connector as mysql


        #In this query, configure the datails that match your mysql database
        connection = mysql.connect(
            host="localhost",
            user="root",
            passwd="password",
            db="BookMeIn2")

        #Sample query to access the SQL query
        df1 = pd.read_sql_query('''SELECT * FROM registrations ''',connection)
```

Figure 1: Sample code to connect MySQL Db and execute any SQL Query

We decided to split and answer these 10 questions. Our objective is not limited to these questions alone. But also finding missing data as well as detecting the trends and pattern in the provided data. I was responsible for analyzing the following questions:

1. What is the impact of preregistration?
2. How many delegates registered and attended?
3. How long did virtual delegates attend for the stand ?

2.1 Software Methodology

Our client's requirement was to focus on the few questions which the event organizers are seeking and detect trends and patterns which help the business. Out of the ten problem questions, we split 2 among ourselves and grant the freedom of analyzing the data and detecting amazing trends. We decided to use Agile methodology for this implementation.

The Agile methodology is a method of project management that divides a project into phases. It entails constant collaboration with the client as well as continuous improvement at each stage. When the work starts, teams go through a cycle of planning, executing, and evaluating phases. So, following this method, we split task goals and assigned to individual team members. I acted as the scrum master and managed the team. I ensured that the team members are on the right track to achieve the goals. I scheduled scrum calls and team meetings. We all agreed to meet twice a week to discuss the progress and status of goals assigned. We did a combination of both physical and virtual meetings. After each meeting I wrote the minutes of the meeting and shared them with the team. I assisted my teammates who faced technical issues as well as challenges in solving the scrum goals.The Collaboration with Client in weeks 2,4 and 8 ensures us the exact problem goals as well as our progress was on the right track.As per the client requirement and our understanding we finally decided to go with a presentation approach rather than Web Application development.

2.2 Client Interactions and Requirements Gathering

Our client provided a brief as well as a detailed explanation of the problem statement and requirements during the initial phase of the project.In the brief, Client provided 10 set of questions to be addressed and answered through data analysis. Following the client's briefing, each team and individual were given the opportunity to ask questions and clarify the underlying concepts. As members of an Agile Team, we know that no one understands the importance of potential features better than the client. During development, we seek face-to-face

communication with the client to confirm understanding and validate alternative solutions. Because, as analysts, we should take advantage of this opportunity to clarify his requirements. We cannot make a guessing game; we need to be completely clear on the requirements before we start analyzing the real data. I used this opportunity to learn more about business aspects and a few queries and suggestions found during the analysis process and confirmed my analysis, as well as my team's intuition, were on the right track.

2.3 Data Analysis and Visualisation

Before analyzing the Data, I configured and imported SQL dump to MySQL Workbench on my machine. Some of my teammates had trouble in installation, So I helped them in installation and importing the SQL dump. Before going to depth analysis, I decided to get prior knowledge of the data, so I read thoroughly the MySQL BookMeIn2 tables information details which is shared by our advisor. By reading this manual, I understood the basic information of the SQL dump data provided and important tables and their relationships in the database. At the same time, I reviewed the BookMeIn2 website for gaining a few insights from the web perspective.

After going through the MySQL BookMeIn2 tables information brief, I decided to walkthrough the actual data. Using MySQL Workbench, I utilized the reverse engineering feature to view the complete ER diagram of the BookMeIn2 schema. The complete ER Diagram was in a bit complex state since there are multiple tables and have multiple relations. Initially I used simple SELECT SQL queries to view the data from the tables and observed most of the details are placed null due to data anonymization viewed every single table in the DB and created a note for crucial tables which consist of relevant data. Later I observed a few critical data like attendees_log is missing and verified it with the advisor. The advisor confirmed that the data is relevant, and he updated it and provided the latest SQL dump for further analysis. I was curious about the patterns in terms of engagement activity at specific times of the day or week.

For the first client visit, I enquired about the audience type and the way in which they advertise the conferences came to know they are sending mail invites over a list of people that they collected over the years. I suggested using social networks to advertise the events, because today social media has the power to spread and advertise the news faster. But the client hesitated the suggestion and told the audience is Government Agencies and cannot share the event details on social media as part of confidentiality agreement. I also enquired about feedback data of the users but came to know its currently not in the scope of the system.

From the insights from the Client, I decided to look further into the questions I had picked. So, at this point I installed Anaconda framework to work with Jupiter notebook. The Jupiter Notebook is an open-source web application that we can use to create python code snippets that contain live code & visualizations, and text. I developed python code to connect with MySQL DB and shared the documentation with my team members (Figure 1). I used MySQL.Connector library for this functionality. Using python code, I was able to convert SQL data output to dataframe format which is easy for data manipulation and visualization used Panda's library for converting the data to dataframe format and finally I used matplotlib.pyplot and seaborn libraries for visualizing charts. My Analysis and insights of few crucial tables are enlisted below:

- **Analysis on BookMeIn2.registrations**: - Could see user can preregister the event and it is flagged in the attribute: 'preregistration'. Regardless of preregistration flag, users need to register for the event. So, we can infer that if the user preregistered and not registered, that means user did not pop up on the event (Figure 2). We could see a flag called vedioactive and some patterns. vedioactive is true only when the register flag is true, and preregistration is false (Figure 3). Later verified with advisor and came to know vedioactive has no relevance to the whole data.

- **Analysis on BookMeIn2.events**:

  ```
  SQL:SELECT id, event_reference,event_type, name,start_time,end_time,
  seminar_video_link,discussion_group_enabled,video_call_enabled
  FROM BookMeIn2.events;
  ```

  event_type id is mapped to event type name in BookMeIn2.event_types table.There are basically 4 event types represented as Figure 4.

preregistration	register
0	1
1	1

Figure 2: preregistration & register flags

preregistration	register	vedioactive
0	1	1

Figure 3: Video active flag pattern

event_type id	event type name
377	Front Door
378	Seminar
379	Exhibition Stand
380	Catalogue Stand

Figure 4: Event types table

id	type_name
327	Delegate
328	Exhibitor
329	Showcase

Figure 5: Attendee types table

- **Analysis on BookMeIn2.attendee_types**:

  `SQL: SELECT * FROM BookMeIn2.attendee_types;`

  Could see 3 distinct attendee types and it is indicated in Figure 5.

My analysis on the following questions:

1. **What is the impact of preregistration?**

   Code snippet:

```
#In read_sql_query function you can provide the sql query
#Query to get the total preregistered count from the registrations table

df1 = pd.read_sql_query('''SELECT count(*) as total_preregistered
                           FROM registrations
                           WHERE preregistration = 1''',connection)
#Query to get the total preregistered & registered count from the registrations table
df1['preregistered&registered'] = pd.read_sql_query('''SELECT count(*)
                                  FROM registrations
                                  WHERE preregistration = 1 AND registered= 1''',
                                  connection)

#Query to get the total preregistered but not registered count from the registrations table
df1['preregistered_not_registered'] = pd.read_sql_query('''SELECT count(*)
                                      FROM registrations
                                  WHERE preregistration = 1 AND registered= 0;''',
                                  connection)

# Plotting the dataframe into piechart
data = [df1['preregistered&registered'][0],df1['preregistered_not_registered'][0]]
mylabels = ["Preregistered and registered","Preregistered but not registered" ]
my_colors = ['lightblue','lightsteelblue']
```

```
my_explode = (0, 0.1)
plt.pie(data, labels=mylabels, autopct='%1.1f%%', startangle=15, shadow = True,
        colors=my_colors, explode=my_explode)
plt.title('ANALYSIS ON PREREGISTRATION')
plt.axis('equal')
plt.show()
```

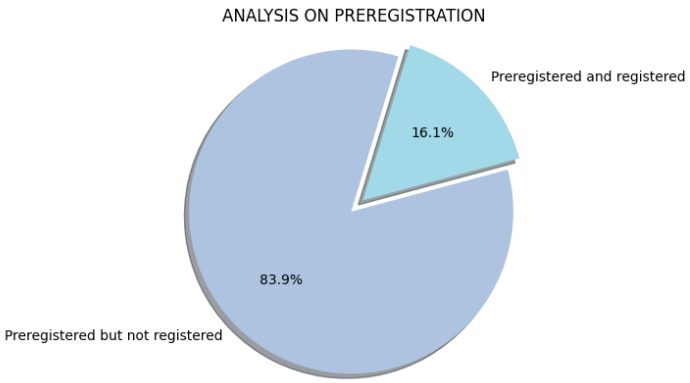

Figure 6: Analysis on Preregistration

Analysis : From the data (Figure 6), I found that more than 80% of the people who are doing a preregistration of the event are not at all registering the event later. Only less than 20% of people are coming later. Attendees might have preregistered when they got the mail invite, but later forget to attend the event. We need to avoid this by increasing the visibility of the event, by sending gentle reminder mails, colorful brochures of the event and if possible, announcements via social media

2. **How many delegates registered and attended?**

   Analysis on Registered attendees based on Attendee Type
   Code snippet:

```
#Query to get the total registered count from the registrations table
df2 = pd.read_sql_query('''SELECT count(distinct (attendee_id)) as total_registered
      FROM registrations
      WHERE registered = 1''',connection)

#Query to get the total deligate count
df2['Delegate'] = pd.read_sql_query('''with t1 as(
                  SELECT attendees.id as attendee_id,attendees.typeid,
                  registrations.registered,registrations.event_id
                  FROM attendees
                  INNER JOIN registrations
                  ON attendees.id = registrations.attendee_id
                  WHERE attendees.typeid=327 AND registrations.registered=1
                  )SELECT count(distinct(attendee_id)) FROM t1;''',connection)

#Query to get the total exhibitor count
df2['Exhibitor'] = pd.read_sql_query('''with t1 as(
                   SELECT attendees.id as attendee_id,attendees.typeid,
                   registrations.registered,registrations.event_id
                   FROM attendees
                   INNER JOIN registrations
                   ON attendees.id = registrations.attendee_id
                   WHERE attendees.typeid=328 AND registrations.registered=1
                   )SELECT count(distinct(attendee_id)) FROM t1;''',connection)
```

```
df2 = df2.T
df2 = df2.drop('total_registered')

#Plotting the dataframe into bar chart

#Set the width and height of the figure
plt.figure(figsize=(8,6))

#Add title
plt.title("ANALYSIS ON REGISTERED ATTENDEES BASED ON ATTENDEE TYPE")

ax = sns.barplot(x=df2.index, y=df2[0])

#Add label for horizontal axis
plt.xlabel("Attendee Type")

#Add label for vertical axis
plt.ylabel("Attendee Count")

#function to print stats over the bars
for p in ax.patches:
    width = p.get_width()
    height = p.get_height()
    x, y = p.get_xy()
    ax.annotate(f'{height}', (x + width/2, y + height*1.02), ha='center')


#function to change the width of the bars
def change_width(ax, new_value) :
    for patch in ax.patches :
        current_width = patch.get_width()
        diff = current_width - new_value

        # we change the bar width
        patch.set_width(new_value)

        # we recenter the bar
        patch.set_x(patch.get_x() + diff * .5)

#Calling change_width function
change_width(ax, .5)

#To display the prepared graph
plt.show()
```

Analysis : I could see around 1428 attendees registered for the various events. Out of this 1428, 478 people are Delegates and 936 people are Exhibitors. From the bar graph, it is evident that delegate's count is around 50% of the exhibitor's count.

Analysis on Registered attendees based on Event Type

I have created the following views, for analyzing this query. Required Data is scattered in the DB, If we created certain views, it will be a fantastic way to optimize our database experience. It also accelerated the data analysis process. Views help me to focus on relevant data that I generated through long INNER JOIN Statements.

SQL for Creating view : reg_events_view -

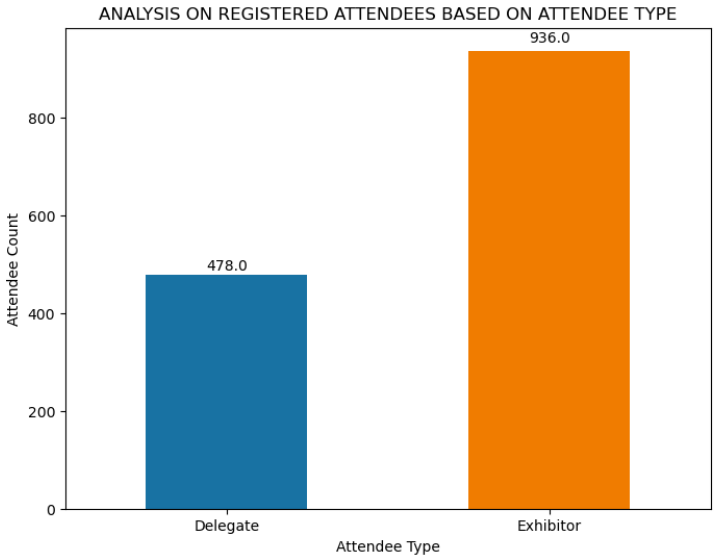

Figure 7: Analysis on Registered Attendees Based on Attendee Type

```
CREATE VIEW reg_events_view AS
SELECT events.event_type,events.start_time,events.end_time,events.seminar_video_link,
events.question_to_delegate_pre_count,events.question_to_delegate_during_count,
events.question_to_delegate_after_count,registrations.id,registrations.preregistration,
registrations.time_registered,registrations.registered,
registrations.event_id,registrations.attendee_id,registrations.greeting_notes
FROM events
INNER JOIN registrations
ON events.id = registrations.event_id;

SQL for Creating view : reg_event_attendee_view -

CREATE VIEW reg_event_attendee_view AS
SELECT reg_events_view.event_type,reg_events_view.preregistration,
reg_events_view.time_registered,reg_events_view.registered,reg_events_view.event_id,
reg_events_view.attendee_id,attendees.typeid
FROM reg_events_view
INNER JOIN attendees
ON attendees.id = reg_events_view.attendee_id;

SQL for Creating view : reg_attendee_session_tracking_view -

CREATE VIEW reg_attendee_session_tracking_view AS
SELECT attendee_session_tracking.attendeeid,attendee_session_tracking.eventid,
attendee_session_tracking.date_pinged,registrations.id,registrations.preregistration,
registrations.time_registered,registrations.registered,
registrations.event_id,registrations.greeting_notes
FROM attendee_session_tracking
INNER JOIN registrations
ON attendee_session_tracking.attendeeid = registrations.attendee_id;
```

Code snippet:

```
#Deligates registered for the front door event
df3 = pd.read_sql_query('''SELECT count(distinct(attendee_id))
      AS front_door_event_delegates_count
```

```
      FROM reg_event_attendee_view
      WHERE registered =1 AND event_type =377 AND typeid=327;'''',connection)

#Deligates registered for the Exhibition Stand door event
df3['exhibition_stand_door_event_delegates_count'] = pd.read_sql_query
                          ('''SELECT count(distinct (attendee_id))
                          FROM reg_event_attendee_view
                          WHERE registered =1 AND event_type =379
                          AND typeid=327;''',connection)

#Exhibitors registered for the front door event
df3['front_door_event_exhibitors_count'] = pd.read_sql_query
                          ('''SELECT count(distinct(attendee_id))
                          FROM reg_event_attendee_view
                          WHERE registered =1 AND event_type =377
                          AND typeid=328;''',connection)

#Exhibitors registered for the Exhibition Stand door event
df3['exhibition_stand_door_event_exhibitors_count'] = pd.read_sql_query
                           ('''SELECT count(distinct(attendee_id))
                           FROM reg_event_attendee_view
                           WHERE registered =1 AND event_type =379
                           AND typeid=328;''',connection)

#Deligates registered for the seminar event
df3['seminar_event_delegates_count'] = pd.read_sql_query
                          ('''with t1 as(
                          SELECT attendeeid,eventid,registered,event_id,
                          attendees.typeid
                          FROM reg_attendee_session_tracking_view
                          INNER JOIN attendees
                          ON attendees.id = reg_attendee_session_tracking_view.attendeeid
                          WHERE registered=1 AND typeid=327
                          )SELECT count(distinct(attendeeid)) from t1;''',connection)

#Exhibitors registered for the seminar event
df3['seminar_event_exhibitors_count'] = pd.read_sql_query
                          ('''with t1 as(
                           SELECT attendeeid,eventid,registered,event_id,
                           attendees.typeid
                           FROM reg_attendee_session_tracking_view
                           INNER JOIN attendees
                           ON attendees.id = reg_attendee_session_tracking_view.attendeeid
                           WHERE registered=1 AND typeid=328
                           )SELECT count(distinct(attendeeid)) from t1;''',connection)

#Creating dataframe from the above data to plot on barchart
plotdata = pd.DataFrame({
    "Delegate":[df3['front_door_event_delegates_count'][0],
    df3['seminar_event_delegates_count'][0],
    df3['exhibition_stand_door_event_delegates_count'][0]],
    "Exhibitor":[df3['front_door_event_exhibitors_count'][0],
    df3['seminar_event_exhibitors_count'][0],
    df3['exhibition_stand_door_event_exhibitors_count'][0]]
    },
    index=["Front door", "Seminar","Exhibition Stand door"]
)
```

```
#Plotting the created dataframe into bar chart

#set the colors
colors = ['#5cb85c', '#5bc0de']

ax = plotdata.plot(kind="bar" ,figsize=(6,6), width=0.5 ,color = colors)

#Add title
plt.title("ANALYSIS ON REGISTERED ATTENDEES BASED ON EVENT TYPE")

#Add label for horizontal axis
plt.xlabel("Event Type")

# Add label for vertical axis
plt.ylabel("Attendee Count")

#function to print stats over the bars
for p in ax.patches:
    width = p.get_width()
    height = p.get_height()
    x, y = p.get_xy()
    ax.annotate(f'{height}', (x + width/2, y + height*1.02), ha='center')

plt.show()
```

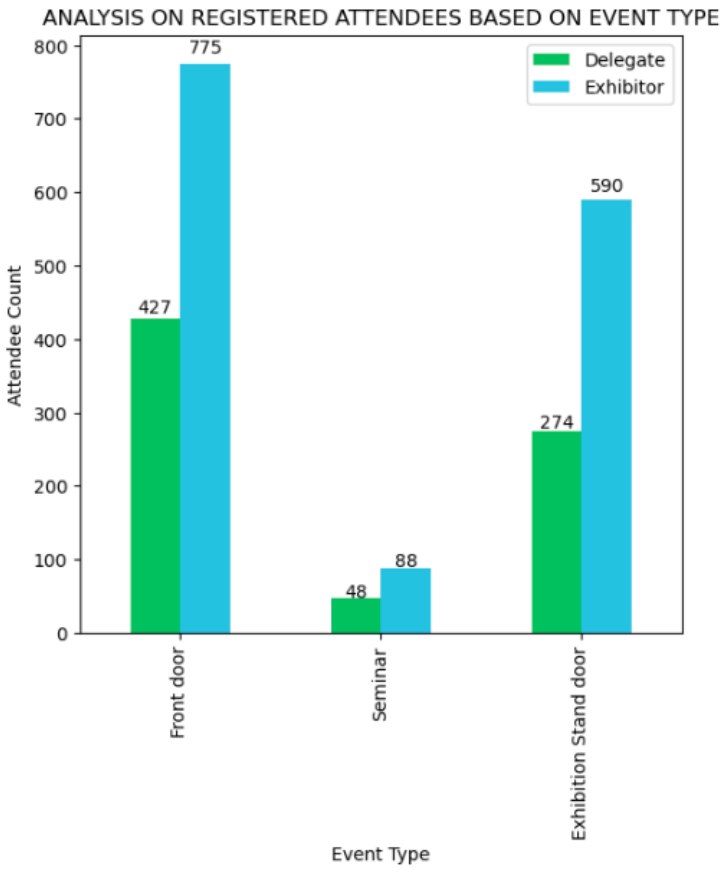

Figure 8: Analysis on Registered Attendees Based on Event Type

Analysis : Here I analyzed the registered attendees based on the event type. I could see most

of the attendees preferred the front door event. In the case of virtual attendees, I could see a total of 132 attendees registered for Seminar and in these 48 attendees were delegates and 88 were exhibitors. For the exhibition stand door event, 274 delegates were registered. In every event type, I could see delegate's count is around 50% of the exhibitor's count.

3. **How long did virtual delegates attend for the stand ?**

   I have created following views,for analysing this query.

```
SQL for creating view : attendee_stand_tracking_view

CREATE VIEW attendee_stand_tracking_view AS
SELECT attendee_stand_tracking.attendeeid,attendee_stand_tracking.eventid,
attendee_stand_tracking.date_pinged,
attendees.typeid,attendees.exhibitoraccount
FROM attendee_stand_tracking
INNER JOIN attendees
ON attendees.id = attendee_stand_tracking.attendeeid;
```

   Code snippet:

```
    #Query to know, how long did virtual delegates attend for the stand
df4 = pd.read_sql_query('''SELECT attendeeid, (count(*) / 6) as duration
                           FROM attendee_stand_tracking_view
                           WHERE typeid=327
                           GROUP BY attendeeid;''',connection)
df4.index = df4.attendeeid
del df4['attendeeid']
df4.head()

#Plotting line chart with attendee id on x-axis and their duration on y-axis

# Set the width and height of the figure
plt.figure(figsize=(16,6))

#Set the backgroud for the graph
plt.minorticks_on()
plt.grid(which='major', linestyle='-', linewidth='0.5', color='green')
plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')

# Line chart
graph = sns.lineplot(data=df4)
#Set the mean dashed line on the line chart
graph.axhline(df4.duration.mean(),linewidth=2, color='r',
linestyle='dashed',label="average")
#Set the mean value over the created dashed line
plt.text(22135,df4.duration.mean()+0.3, round(df4.duration.mean(), 2),
fontsize=12,color='r', va='center', ha='center')

#Set the max dashed line on the line chart
graph.axhline(df4.duration.max(),linewidth=2, color='#FF007F',
linestyle='dashed',label="max")
#Set the max value over the created dashed line
plt.text(22135,df4.duration.max()+0.3, round(df4.duration.max(), 2),
fontsize=12,color='#FF007F', va='center', ha='center')

#Set the min dashed line on the line chart
graph.axhline(df4.duration.min(),linewidth=2, color='#FFA500',
```

```
linestyle='dashed',label="min")
#Set the min value over the created dashed line
plt.text(22135,df4.duration.min()+0.3, round(df4.duration.min(), 2),
fontsize=12,color='#FFA500', va='center', ha='center')

plt.legend()

plt.title("VIRTUAL DELEGATES VS STAND DURATION")
plt.ylabel("Duration (In minutes)")

plt.show()
```

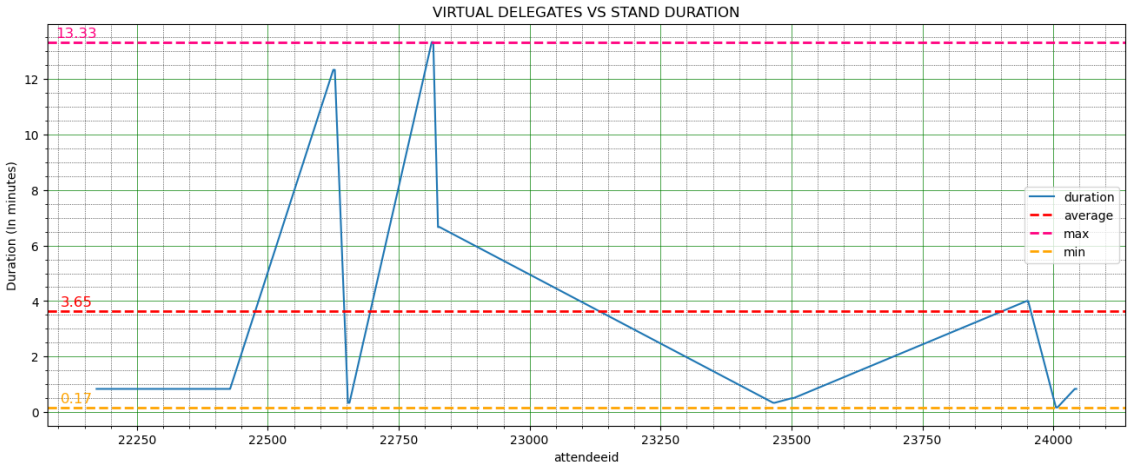

Figure 9: Virtual Delegates VS Stand Duration

Analysis :
From the line graph, I could observe that virtual delegates spend an average of 3.6 minutes on stands. The maximum time spent on the stand is 13.3 minutes whereas minimum time spent is 0.16 minutes.

2.4 Testing and Evaluation

Since it is a Data analysis project, we tested the accuracy by performing other teammates query ourselves and validated that we are getting the same results. I validated my teammate's query: "How long did virtual delegates attend?". I analyzed the time spent by Virtual Delegates on Stand Duration. So as part of it, I got curious and viewed a similar metric for the following queries too:

How long did virtual delegates attend the session/event?
How long did virtual exhibitors attend the session/event?
How long did virtual delegates attend the stand? and
How long did virtual exhibitors attend for stand?

I answered all these questions. While going through these questions, I observed my few questions match my teammate's question, so we validated our results and came to conclusion that our metric was similar.

2.5 Other Key Responsibilities

I did some effort to investigate the crucial missing data in the DB. Because it was also a requirement for the client. Since they are collecting a bunch of event details data for analyzing the patterns and trends, then they do not want to miss the key features. By analyzing the DB, Data of Birth and Feedback data are the missing data in the system. If dob attribute was present in the attendee or user table, we could calculate user's age and form conclusions regarding the age criteria or pattern of attendee's participation. If we have feedback data, we can perform sentimental analysis, because

the feedback data contains strong positive and negative words and thereby, we can measure user satisfaction.

My other responsibility was to create the template for our presentation. I suggested the structure of the submission files with the team and assisted in making user documentation and technical documentation files. I suggested we can go with Jupyter notebook files or Python files and use the visualization graphs generated from it to do the presentation. I have shared the sample structure of the presentation and my contribution in that presentation and shared my final version of python notebook file with my team. I properly commented on my code snippet for future reference asked my team to update their contribution and advised to use save as option with latest version number on the file name, so that, we can ensure that the data is safe. After my team's contribution I reviewed both presentation and notebook file and validated the results generated from the notebook files in my machine too. I Eliminated a few errors in the jupyter notebook file and commented on all SQL view creations. For the presentation, I have done some styling and uniformity in all slides.

3 Conclusion

In conclusion, by answering the questions that the conference organizers are seeking I have found some trends and relationships among the variables in the data. Also, I suggested a few solutions for business insights. Because based on the trend, for better participation, it is evident that clients must focus more on event reminders.

This project taught me a lot. It was an amazing experience to work as a team to solve a real-world client problem. Our group worked extremely well together. Everyone voiced their opinions, and none were met with hostility. Collaboration improves how our team works together and solves problems. This results in more innovation, more efficient processes, greater success, and better communication. And I believe that every good team has a diverse set of skills that will contribute to novel approaches to success.