Microsoft Garage:
Modernizing Data Processing at the Museum of Science

A Major Qualifying Project
of the B-Term 2016 site in Cambridge, Massachusetts,
submitted to the faculty of
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the
Degree of Bachelor of Science.

Submitted by:
Nicholas Bradford, Tim Petri, Himanshu Sahay

Sponsor:
Microsoft Corporation

Liaison:
Ben Fersenheim

Project Advisor:
Professor Mark Claypool

Submitted on:
January 16th, 2017

Abstract

The Hall of Human Life exhibit at the Museum of Science in Boston generates thousands of data points per day at its interactive kiosks but does not leverage modern software tools to store and analyze the nearly 10 million records. As part of the Microsoft Garage project lab, we built a prototype system allowing the Museum to host all their data in the cloud with Microsoft Azure, monitor the exhibit in real-time with a Power BI operations dashboard, and automatically detect hardware failures with an anomaly detection system in Azure Machine Learning.

Acknowledgements

We would like to thank our sponsors at Microsoft in Cambridge for the project opportunity as well as their guidance, particularly Ben Fersenheim, Don Cox, Aimee Sprung, and Ashish Jaiman.

We would also like to thank all of the Museum of Science staff that have helped us throughout the process, including Elizabeth Kong, Ben Wilson, Keith Simmons, and Jacob Barry.

Finally, we would like to thank our project advisor, Professor Mark Claypool, for his guidance throughout the PQP and MQP process.

Table of Contents

Chapter 1: Introduction

Over the last several decades, museums have evolved from places of observation into immersive experiences designed to enthrall, inform and teach. Since November 2013, the Museum of Science in Boston (MoS) has operated the Hall of Human Life, an interactive exhibit which conveys how humans evolve every day as individuals, as a species, and as members of the ecosystem around them. Each of the exhibit's fifteen kiosks records various forms of visitor data as visitors complete interactive challenges. As of October 2016, data for nearly one million visitors has been recorded over the course of three years.

The data at the MoS is currently stored in an on-premise Microsoft SQL Server database with a rudimentary Microsoft Excel dashboard available to museum staff. The Mos staff would be better served with a rich, user-friendly dashboard to view and interact with the data, allowing them to better manage and improve the exhibit. In addition, many of the datasets in the Hall of Human Life are polluted with inaccurate data, many of which occur because of non-obvious hardware failures. Building a system capable of detecting individual anomalies as well as potential hardware failures would greatly aid in the exhibit's administration and maintenance.

We present a new solution to handle their data and generate insights through a new dashboard. Our solution provides three components:

1. Continuous availability of the entire dataset in the cloud.
2. Data-driven insights on the health of the kiosks by visualization of visitor metrics as well as data points deemed to be outliers.
3. Robust analytical tools to remove outlier data and sample effectively across the exhibit audience.

The Background chapter introduces the Museum of Science, and specifically the Hall of Human Life, and also discuss the existing system in place. The Methodology chapter lays out our procedure for moving the existing dataset to the cloud, designing the operations dashboard, and building the anomaly detection model. The Results chapter details the architecture for the complete system. The Future Work chapter contains our recommendations for the project's near-term roadmap. Lastly, the Conclusions chapter summarizes our project and its potential impact.

Chapter 2: Background

In this chapter, we discuss the relevant background to our project. We begin by covering the Museum of Science, and specifically, the Hall of Human Life exhibit on which our project focused. We additionally discuss properties of the data being gathered at the exhibit, as well as the architecture of the existing system for its storage and management. Finally, we conclude with an overview of the relevant Microsoft technologies and resources which we used.

2.1 Museum of Science and Hall of Human Life

The Museum of Science (MoS) in Boston was founded in 1830 and contains over 700 interactive exhibits as well as live presentations, an Omni theater, and over 100 animals (mos.org). The mission of the museum is to play a leading role in transforming the nation's relationship with science and technology, particularly by promoting active citizenship, inspiring a lifelong appreciation of science and technology, and encouraging young people of all backgrounds to explore and develop their interests in understanding the world.

In November 2013, the MoS opened the the Hall of Human Life (HHL), a 10,000 square foot biology exhibit featuring more than 70 interactive components that let visitors engage with their biology and understand "how humans are changing" [2]. Through an anatomical lens, visitors explore how humans are changing as individuals, day-to-day, and during a lifetime. Through an evolutionary lens, visitors explore how humans are evolving as a species. Through an environmental lens, visitors explore how humans are changing the environment, and in turn, how the environment changes us. Five different categories make up the exhibit: Communities, Time, Organisms, Food, and Physical Forces [3].

2.1.1 Data Collection

When a visitor enters the HHL, s/he receives a wristband with a unique scannable barcode. Each of the five categories features three kiosks (officially known as link stations) at which visitors can scan their wristbands to start their interaction and answer questions. Then the visitor is presented with an interactive challenge and can see data collected through the challenge, as well as a sample of results from the previous 200 guests. Figure 1 depicts a visitor starting his/her interaction with a kiosk by scanning his/her wristband.

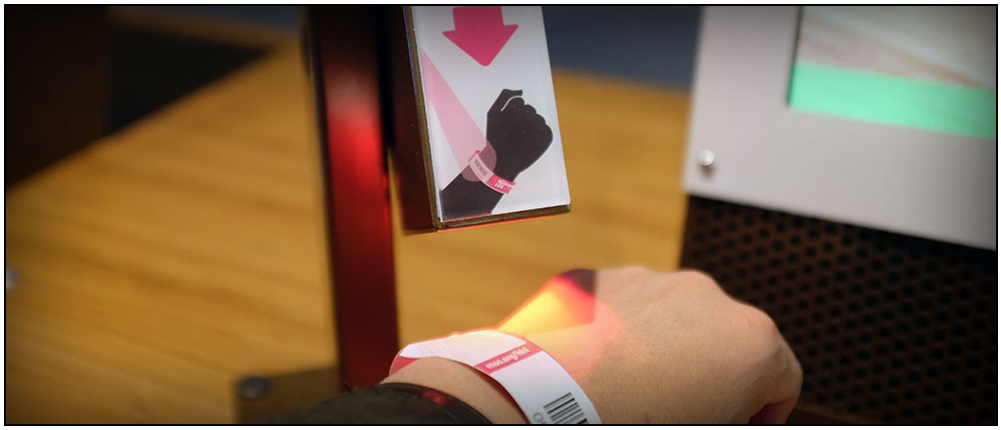

Figure 2.1.1-1: Participants scan their unique barcodes at kiosks [2]

2.1.2 Website

After visiting the HHL, visitors can use their wristband barcodes to continue exploring their collected data online at https://www.mos.org/hhl, with access to the same visualizations available in the exhibit. Figure 2 depicts two visualizations on the website, the first showing maze completion times for visitors of the Balance kiosk, and the second showing the maze completion time by age for visitors to the same station.

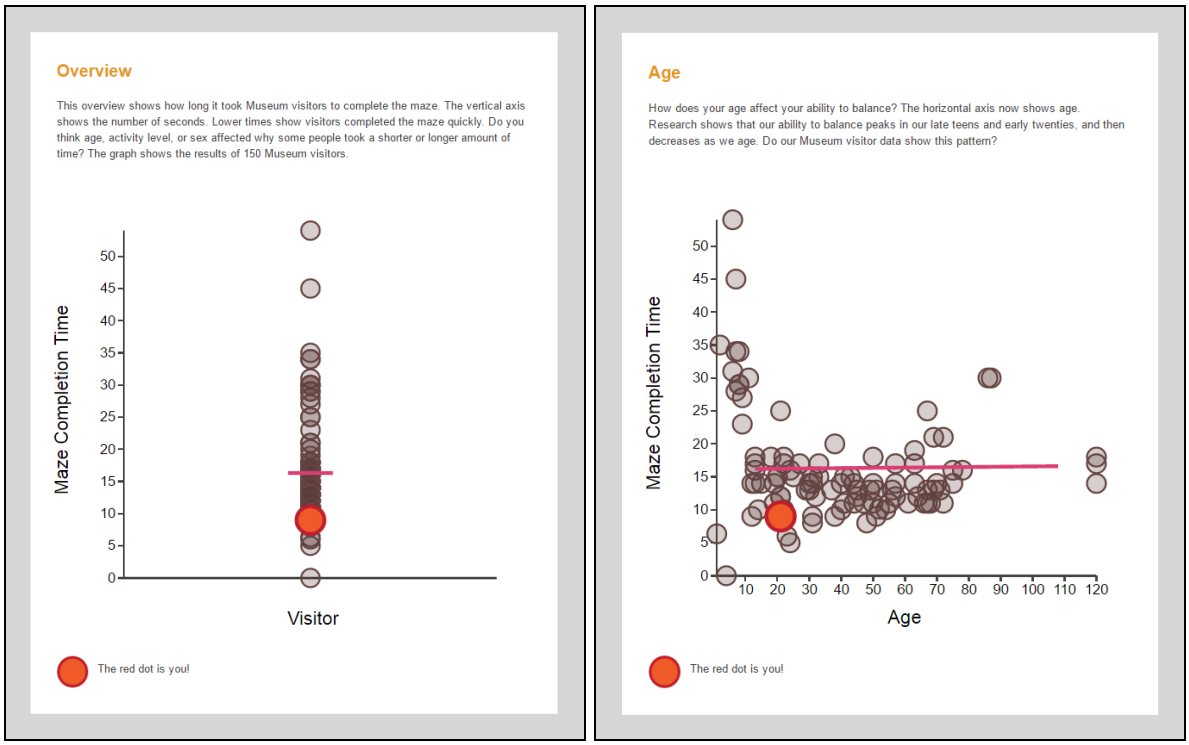

Figure 2.1.2-1: Two website visualizations for the exhibit, "BAL: Does Balance Change With Age?" [2]

2.1.3 Kiosk Overviews

The kiosks at the Hall of Human life, segregated by their categories are [4]:

Physical Forces

Roboarm: What keeps you awake?

Visitors play a space simulation game in which they must use rotational and three-dimensional translational controls to correctly position an object on a space station. Points are deducted for colliding the object with the station, placing the object inaccurately, or taking too long to complete the placement. Scores are provided as percentages, ranging from 0 to 100.

Tekscan: How high is your foot arch?

Visitors measure their foot arches by taking off their shoes and walking across a mat. The height of the foot arch is measured in Modified Arch Index (MAI) terms, ranging from 0.0 to 0.4. Walking with shoes, on toes, and on heels as well as with flat feet may result in a low foot arch reading.

Finger Temp: Are your fingers the first to freeze?

Visitors put a hand on a cold pack, and a thermal camera measures how much the person's finger temperature changes in one minute. The change is temperature is reflected in a range from -15° F to +15° F.

Communities

Facial Recognition: Do you ever forget a face?

Visitors look at two sets of faces. For the first set, they look at different faces one at a time, and for the second, they look at a larger group of faces two at a time. Some of the faces in the second group are new, and some have been seen by the visitor before. The station records response times of the visitors in recognizing familiar and unfamiliar face shapes. It then plots the average response time in seconds in the range 0.0 to 5.5 seconds.

Social Complexity: How does your circle of friends change your brain?

Visitors are asked questions about their social interactions in the last two weeks, and a map of their social network is created. Scores range from a social network size of 0 to a size of 60.

Family Structures: What influences the age when someone leaves home?

Visitors are asked questions about the people they grew up with at home and the age at which they left home to move out on their own. Results shown for the age when people left home are plotted on a graph that ranges from ages 0 to 60.

Food

Food Instincts: What makes you hungry?

Visitors are asked to pick food from a display case with a limited selection of food items, and then from a display case with a wider selection of food items. Visitors can pick as many portions of each item as they like. The station aims to determine whether being given more food options makes you more hungry. Scores are represented as the difference in calorific value between the first and second meals on a scale from -300 to +1500 calories.

Snibbe: How efficient is your walk?

Visitors walk across a mat modeled as a sidewalk. The walk shadow of the visitors and the number of grapes and calories burnt per mile with the current walk speed are calculated and displayed to the visitor. The minimum Basal Metabolic Rate (BMR) displayed is 500, and the maximum is 10000, while the number of grapes burnt per mile are shown in the range of 15 to 490 grapes.

Food Tech: What food technologies do you support with your purchase?

Visitors start off by being asked whether they have been to a restaurant in the past day. They then use a hand-held barcode scanner to scan a food item that they have eaten within the last day, and learn about a technology that supports or influences the food, such as the use of fructose corn syrup in the manufacture of soda and the use of genetic modification in corn.

Organisms

Infections: How are you feeling today?

Visitors answer questions about whether or not they have had flu-like symptoms in the previous two weeks. They are also asked about their hygiene habits and whether they got a flu shot in the current season. Their reported data helps researchers learn more about the rate of infection for the flu in the United States.

Allergens: What factors predict whether a person develops allergies?

Visitors answer questions about their allergies and those of their parents and siblings. They are also asked about their environment growing up such as whether they grew up in a city. Visitor data is shown as a data point in one of four boxes, which are combinations of whether or not the visitor has an allergy, and whether or not one or more of the visitors' parents had allergies in the house where the visitor grew up.

Biophilia: Do you look scared?

Visitors look at pictures of animals while an eye tracker follows and records changes in their pupils in response to looking at the pictures. The pictures are of domesticated and non-domesticated animals which are, in order, a bunny, a cat, a rat, a snake, a tiger. The changes in the diameter of the visitors' pupils are shown on a graph in millimeters.

Time

Distraction: Are you paying attention?

Visitors have two seconds to quickly estimate if there are more red or more blue dots on the screen. Images are used to distract visitors while they do this task. Visitors are also asked some information about themselves. The visualizations of visitor scores are stratified using answers from preliminary questions asked to visitors such as “Do you play video games?”

Balance: Is your balance as good as it gets?

Visitors stand on a pad with accelerometers and use their balance to move a ball through a maze. They are also asked questions about their age and activity level. The visualization of scores is stratified by activity level, age, and gender.

Ear Measurement: Discover how your ear's shape is determined.

A visitor takes a picture of her/his ear and measures the length. S/he are also asked to input her/his age and height. The way a visitor angles her/his head for the picture and the direction in which the ear edge points are determining factors in the ear length calculated, which can cause this system to be susceptible to errors in measurement. The ear lengths are then presented on graphs stratified by age and gender.

2.2 Current Data Storage and Display System

2.2.1 Use cases

The system currently in place at the Hall of Human Life has four primary use cases:

1. Visitors are able to interact with kiosks and then be presented with visualizations supported by not only their own data but also data from other visitors.
2. Visitors can access their data and visualizations on the Hall of Human Life website.
3. Exhibit staff and administrators can view exhibit data through quarterly reports, in order to examine, understand, and improve the exhibit.
4. Librarians interested in the data for teaching purposes can request access to some subset of the data during meetings at the museum.

2.2.2 Architecture

The current architecture at the HHL is divided across two different network domains: Exhibit and MOS (as shown in the Figure 2.2.2-1).

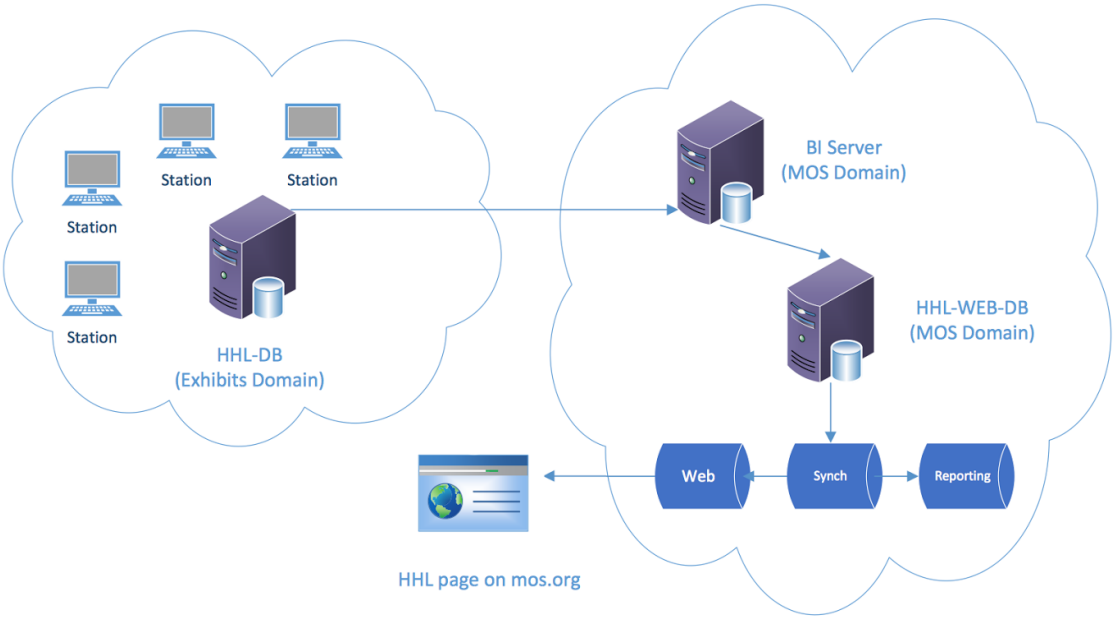

Figure 2.2.2-1: High level overview of HHL architecture

In the exhibit domain, all kiosks (kiosks) have two-way communication over Open Database Connectivity (ODBC) with the exhibit server's Microsoft SQL database. Upon receiving new visitor answers, a kiosk will both update the central database while pulling any relevant data from the server (the visitor's height/age data, and previous visitor data points used for the visualizations). Additionally, kiosks cache their data both for efficiency when creating visualizations, and as a backup in case the connection to the main database is lost.

Data in the exhibit domain is moved over to the MOS Domain every 10 minutes. In this domain, the BI Server is an intermediate server through which the data gets moved to the last server, HHL-WEB-DB, which has two separate databases addressing the needs for data outside of the exhibit experience. The Web database powers the experience on the mos.org website, allowing retrieval of data based on the bracelet barcode. The Reporting database is used to share data internally with exhibit staff or to cull data to be used by visiting librarians. The Web database is updated as data is moved between the two domains and the reporting database is synced with the Web database by a job that runs nightly.

By redundantly storing data across the two domains in separate databases, the museum ensures that the experience in the exhibit is not affected by an increase in traffic from the website, or that the generation of reports for other needs affects the visit experience.

2.2.3 Data Details

There are two main pieces of information stored that are used in the exhibit: basic information such as age and gender for each visitor and visitor answers. Each visitor answer is generated when a visitor either answers a question about themselves prompted by a kiosk, or performs an activity from which a value such as a score or a measurement can be inferred.

Over 650,000 visitors have contributed data in the exhibit since it opened in November

2013. In total, close to 10,000,000 visitor answers have been submitted, forming a rich dataset of interactions in the exhibit. However, the visitor data is archived every 90 days, so the full data set is never used in the visualizations at the kiosks or on the website. On top of the answers, visitor interactions also produce images, videos and other media assets at many of the measurement stations. The visitor generated images and videos grow by about a terabyte a year.

2.2.4 Report Generation

In order to create reports, the MoS uses a series of interactive Microsoft Excel tables and charts, referencing data that has been manually imported into Excel from the SQL database. In addition to being unable to be used quickly or allow near real-time analysis, it is difficult to maintain and currently is only partially functional.

Figure 2.2.4-1 shows the "Dashboard" view of the interactive Excel charts currently in use. This view gives a high level overview of the visitor metrics for each kiosk. Users are also allowed to filter by gender, age, time of day of the visit, and date of visit.

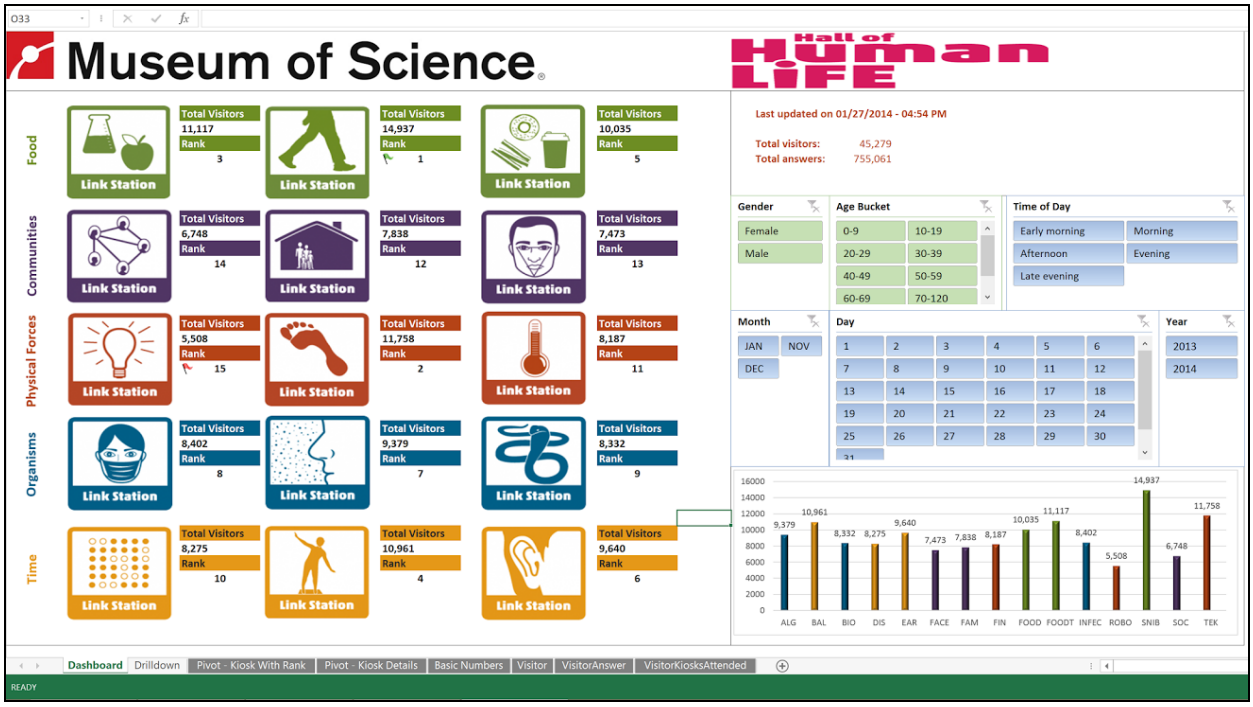

Figure 2.2.4-1: Existing report solution: Dashboard overview.

Figure 2.2.4-2 shows the "Drilldown" view which graphs the number of answers for a kiosk, stratified by age and gender. Users can further filter this graph by the specific question asked and even by the answer received.

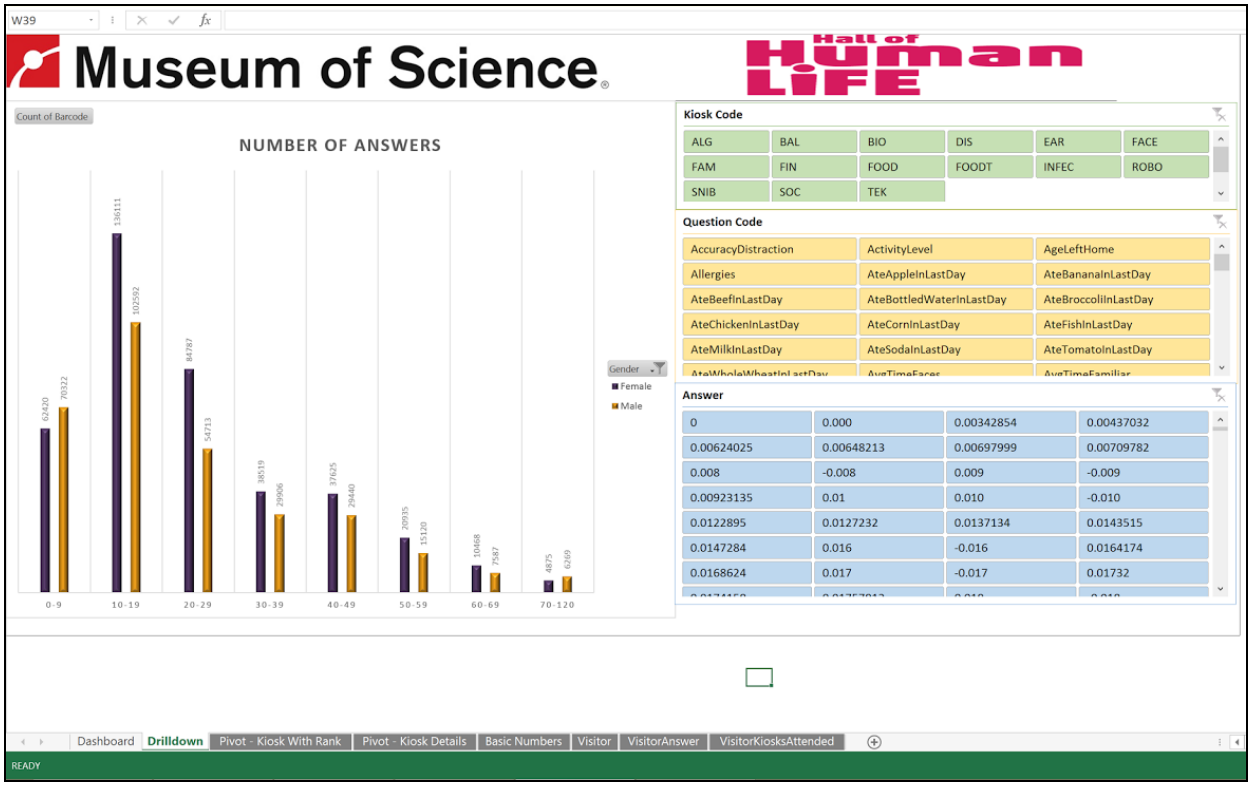

Figure 2.2.4-2: Existing report solution: Drilldown

Figure 2.2.4-3 shows the "Pivot - Kiosk Details" view, which shows the number of visitors to a particular kiosk, stratified by age and gender. This view is connected to the "Drilldown" view and shows metrics for the kiosk selected in the "Drilldown " view.

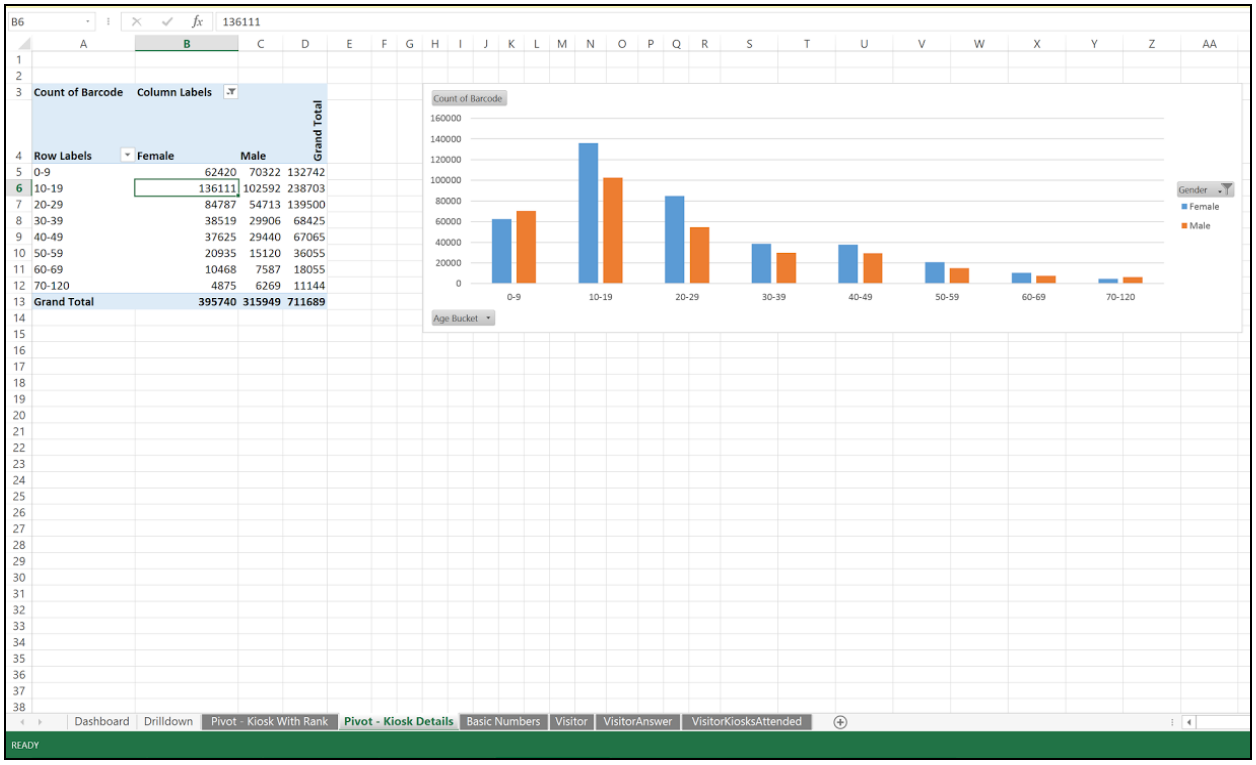

Figure 2.2.4-3: Existing report solution: Kiosk Details pivot table

2.3 Technologies and Resources

Microsoft Corporation has granted the Museum of Science an Azure grant for $50,000 in credit over 3 years to be used to modernize their existing solution at the HHL.

2.3.1 Microsoft Azure

Azure is a cloud computing platform and infrastructure used for building, deploying, and managing various applications and services. Azure uses include creating SQL Databases, massively scalable storage, functions for serverless code execution, machine learning, and Internet of Things (IoT) integration [5].

2.3.2 Power BI

Power BI (Business Intelligence) is an analytics tool with both web and desktop versions, capable of creating sophisticated visualizations, reports, and dashboards. The desktop version is used to access data sets, massage data as well as create reports. These reports are published to the web version of Power BI, which can also be used to create reports and to push visualizations from these reports to dashboards which can be shared online.

Power BI also provides built-in integrations to Azure storage services such as SQL databases and Azure Blob Storage. Using DirectQuery mode in Power BI allows the dashboard to connect directly to the underlying data source, making repeated data transfers significantly more efficient

[6]. No data is stored in Power BI, and there is a direct connection from Power BI to the data source. The data is continuously updated in Power BI for near real-time capabilities, and if true real-time capability is needed (visualizations being updated without the user refreshing the dashboard) in order to create a live dashboard, developers can use DirectQuery mode in combination with a streaming data set pushing data to Azure Stream Analytics.

A report in Power BI is a collection of user defined visualizations and filters created in either the desktop or online application. These reports can be published themselves or be pinned in real time ("live") to a dashboard. Dashboards are made up of individual visualizations from reports or consist of an entire report itself when it is pinned "live."

Chapter 3: Methodology

In this chapter, we discuss the methods we employed to expand and improve our proposed solution into a final deliverable. We addressed the three parted problem faced by the MoS in the following manner:

1. To make the database scalable, reliable, and easy to integrate with applications, we moved it to an Azure SQL database.
2. To give the museum staff real-time visual insights into the HHL exhibit, we created a dashboard which can also generate reports on demand.
3. To flag outlier data and alert the museum staff about hardware failures, we created rules to specify bounds for acceptable data for each kiosk, and used machine learning to create even more sophisticated models to flag outlier data.

3.1 Acquiring and Moving Data to Azure

Moving the museum data to the cloud was the first step in addressing the identified concerns of the museum. It provides a single storage point for data with scalable performance and allows for easy integration with Power BI and Azure Machine Learning, both of which were used extensively for the rest of our project.

3.1.1 Understanding the Exhibit Data Model

We first set out to understand how the data on visitors and their interactions was being handled in the current solution. We requested the database schemas covering the on-premise databases used for storing and serving information in the exhibit, as well as the web. Much work had originally gone into the design of a system that allows for an arbitrary number of kiosks, each with their own questions. Since we wanted our system to eventually be able to replace most of the in-museum infrastructure for the Hall of Human Life, we decided to have our database model closely mirror theirs. This lets the museum effectively swap out their old system for an implementation like ours with minimal operational efforts. Additionally, moving all historical data to the cloud, discussed in the next section, becomes easier. Using our planned database model closely mirroring the one in the museum, we created an Azure SQL Database.

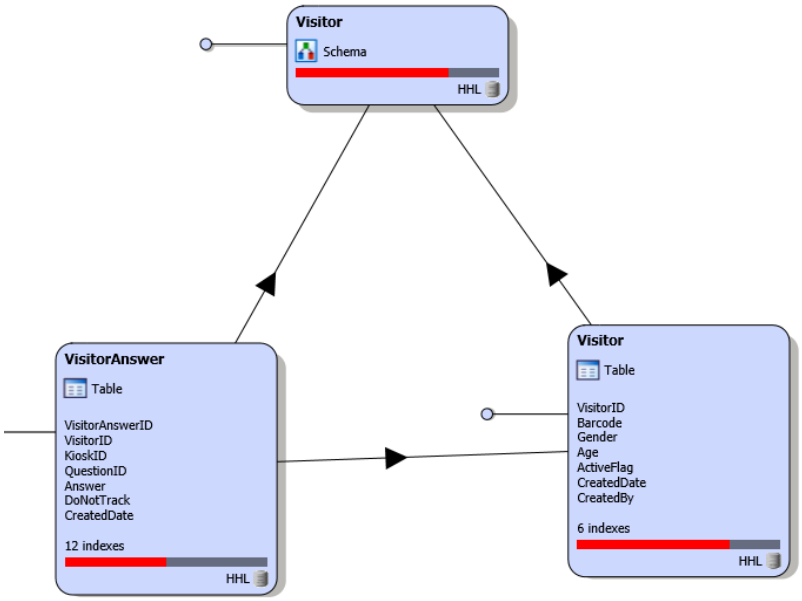

Figure 3.1.1-1: Part of original schema representing the visitor information and visitor answers

Our understanding of the exhibit data model came from meetings with staff and access to the schema documents that were used in the design of the exhibit. Figure 3.1.1-1 showcases one of the most fundamental parts of the exhibit database schema we received; the Visitor and VisitorAnswer tables that represent data on the visitor and every answer they generate in response to questions and measurements. As described in Background chapter, we found out that the complete museum architecture actually involved several databases redundantly storing data for in-exhibit, website, and reporting needs, separately.

3.1.2 Acquiring and Moving Data to the Cloud

In order to successfully carry out this project, we not only needed to understand how data was managed internally at the Museum, but also get access to the data itself. This lets us test and make sure that our database implementation worked correctly with data in place. Secondly, it was necessary to understand the nature of the data, exactly what constitutes outlying data, and eventually allow us to train machine learning models for detecting this. Lastly, it aids us in the design of the dashboard, providing actual content for the many visualizations.

In the early stages of our project, while waiting for approval to receive data directly from the museum, we decided to manually scrape the data from visualizations on the Hall of Human life section of the mos.org website. This data represented 1000 visitors and more than 7000 visitor answers across almost most kiosks in the museum. The data missed exact timestamps and was not sufficient in size to train any machine learning models but at least let us examine the different ranges of values produced by visitors. As we scraped the data, we also discovered that the uniquely identifying barcodes and internal file paths for videos and images were included, although not explicitly displayed. Since this information is of no use to the visitor online, and ideally should not

leave the museum, we communicated our findings to the staff.

Once we had received approval to work with the on-premise data we were given access to a much larger data set covering 90,000 visitors and 660,000 answers from previous 3 months. This dataset was complete with timestamps, and let us see exactly how our the dashboard would work. Finally, near the end of our project, we received the complete historical dataset, covering some 600,000 visitors and almost 10,000,000 answers. This allowed us to train machine learning models, and show the exploratory power of the dashboard we were working on.

Some reformatting of the data was required as the different datasets were structured differently. To do so, we wrote python scripts to convert the data into .csv files that matched the table schemas for our Azure SQL Database. We used the command line utility called *bcp* to run bulk imports to Azure SQL from these .csv files.

3.1.3 Adding Support for Dashboard and Rules

In addition to setting up and moving data to the database, a number of additions were made to support services interacting with the data such as the live dashboard and anomaly detection system.

We created database views specifically for the dashboard. These views were used to aggregate the data on visitors and interactions and allowed us to surface detailed information such as age distributions, dwell times, and hourly visitor rates. As the dashboard reruns queries using these views, visualizations are always updated with the latest data.

After the analysis of the data described in section 3.3, Anomaly Detection, it became apparent that many of the observed issues with visualizing the data could be eliminated by introducing bounds on the values data points take. As a way of flagging data points that were considered outliers and came up with an in-SQL rules engine that let one define upper and lower bounds for answers made in the museum (e.g., time slept in a day). With these rules in place, each visitor answer could be flagged at insertion time. The flagging ensures that all data is stored and nothing gets thrown away while providing a way to selectively select data that is not flagged and considered outlying. With the rules in place, we went back and flagged all historical data that was considered anomalous.

3.2 Real-Time Dashboard

We decided to leverage the data massaging and visualization capabilities of Power BI to build dashboards for the MoS. These dashboards allow for a real time view of different aspects of the HHL kiosks, from visitor metrics to answer completion rates to information about outlier data.

We used Power BI Desktop to connect with the Azure SQL Database via DirectQuery mode. In the “Home” tab in Power BI Desktop, we selected “Get Data” and connected to the Azure SQL Database by inputting the server name and database name. Connecting via DirectQuery ensures that the data is never stored in Power BI and that there is a live connection between Power BI and the Azure SQL Database, allowing real-time capabilities. This ensures that the data in the connection from the Azure SQL Database to Power BI is always up to date and is reflected in the visualizations as well. Large datasets sometime do not update automatically, so the user needs to

manually refresh the dashboard using the “Refresh dashboard tiles” option from the ribbon in the top right corner of the dashboard.

3.2.1 Massaging the Data

Once we had a connection to the data in Power BI, we could use the views created from the relational tables in the Azure SQL database to create visualizations in our report. We also decided to create filters to allow users greater flexibility in narrowing down insights by a specific demographic. The first step was to create relationships between the views so that we could filter visualizations from different views using common filters. Figure 3.2.1-1 shows the relationships created between all the views in Power BI Desktop, including the cardinality and direction of the relationships.

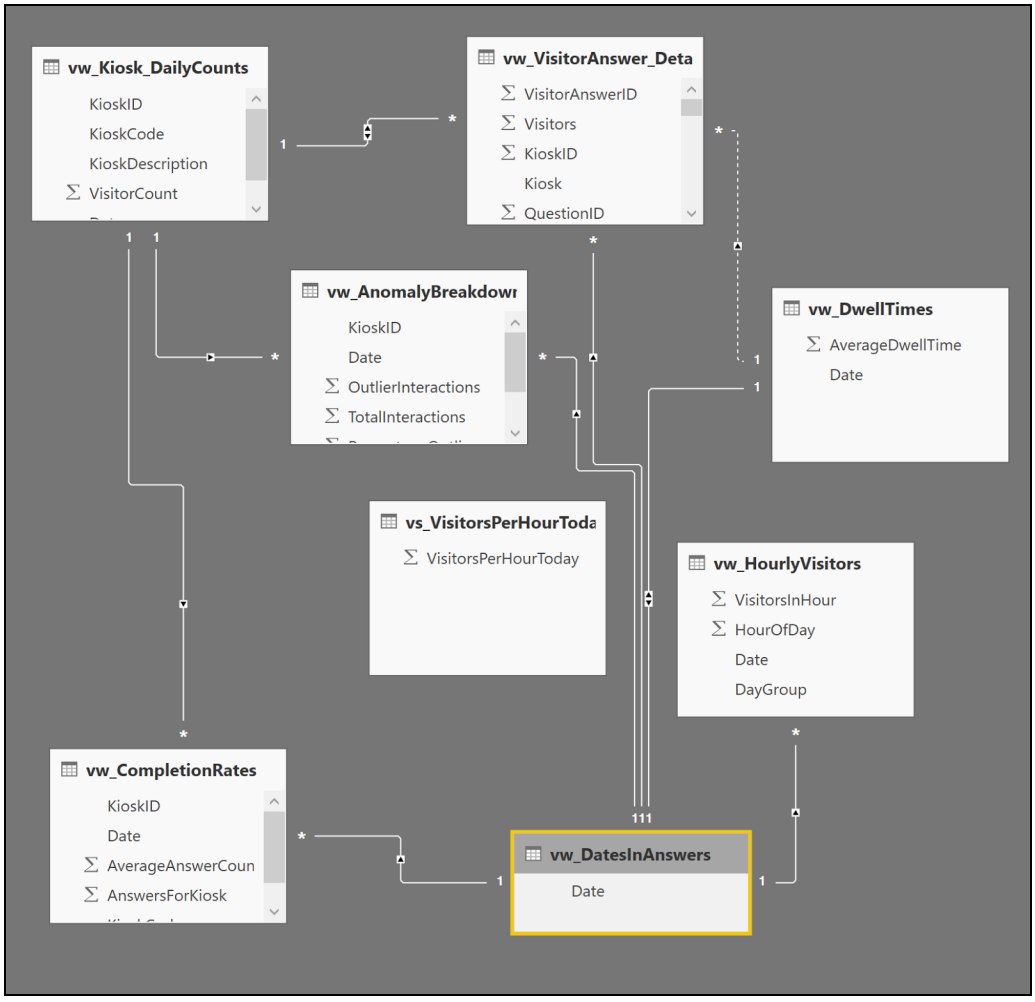

Figure 3.2.1-1: Relationships between views in Power BI Desktop

In order to create filters, we massaged the data in Power BI by creating new custom and conditional columns in the views accessed. The filters allow us to narrow down the visualizations by:

- Age of visitor
- Gender of visitor
- Date of visit
- Time of visit
- Kiosk interacted with
- Category of kiosk interacted with

Figure 3.2.1-2 shows the addition of a conditional column to a view. Steps applied to massage the data in the view are listed on the right side of the image. We also created measures in the views to compute simple calculations related to data in the view which we did not want associated with every row in the view. A measure is a calculation that is associated with a relation and is stored in Power BI but is not a column in the relation.

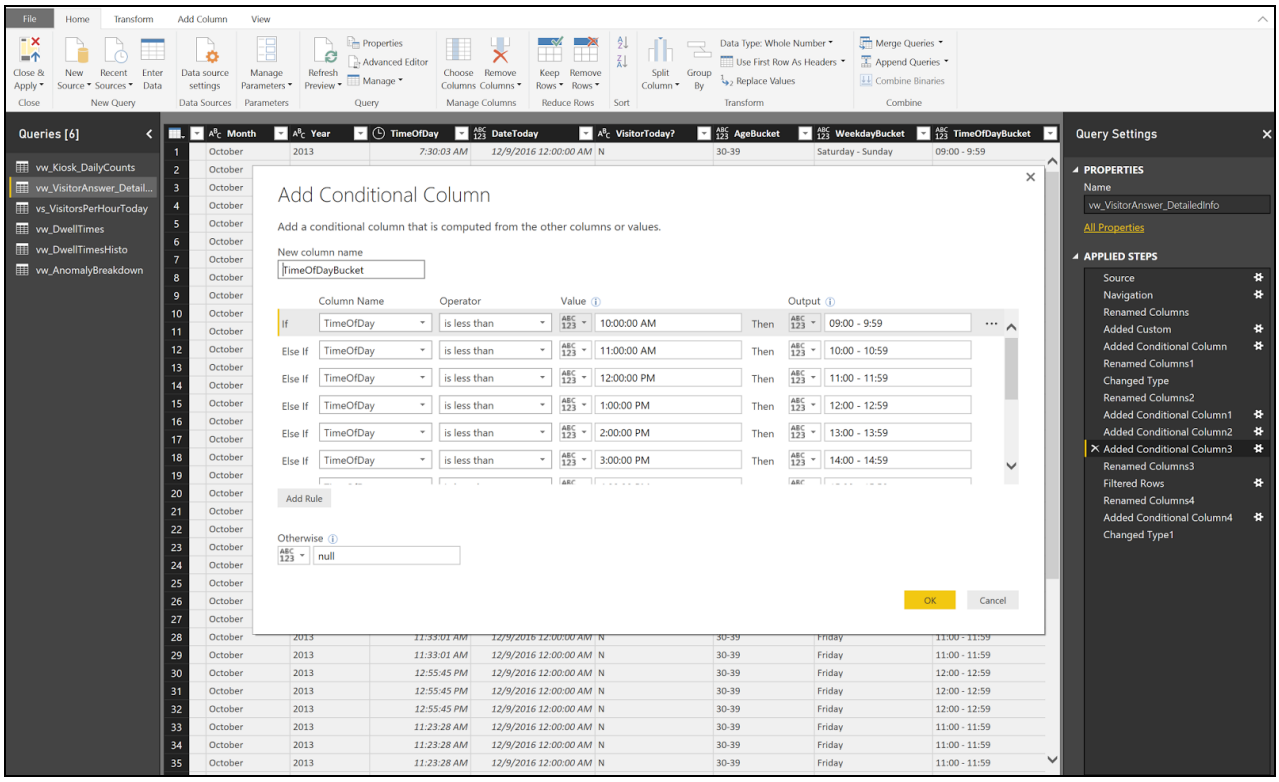

Figure 3.2.1-2: Adding a conditional column to a view in Power BI Desktop

Once we massaged the data, we created visualizations in Power BI Desktop that lend insight to the data. We followed a clean design philosophy after reading the seminal book on dashboard design, Information Dashboard Design: The Effective Visual Communication of Data by Stephen Few, to create aesthetically appealing and easy to understand visualizations which are arranged on different pages in a Power BI Desktop report. Each page features a number of filters on the right, separated from the visualizations on the left by a divider. The visualizations make use of line charts, bar charts, a pie chart, and a gauge to effectively convey insights to a user, while filters include a date picker for selecting date ranges and slicers which allow multiple selections for the same filter.

We will go into detail about the visualizations and filters employed in the Results chapter. After creating the pages in the Power BI Desktop report, we published the report to the Power BI service.

3.2.2 Live Dashboard

Once the Power BI Desktop report was published to the Power BI service, we were able to access the report online in the Power BI service. Here, we published individual pages from the report into dashboards that the museum staff can interact with. We accomplished this by selecting the "Pin Live Page" option for each page of the report, to publish it to a separate dashboard. These dashboards are neatly stacked in the dashboard view for the museum staff to quickly toggle through.

3.2.1 Report Generation

Reports can be generated from the Power BI service to show data and insights on the HHL exhibit filtered by any of the available criteria and shared with stakeholders. These can be shared one of two ways:

1. Through the Reports tab in Power BI service by publishing the report for public access on the Internet, printing, embedding in Sharepoint Online or Powerpoint, or downloading to Power BI Desktop. This option shares a static version of the report and receivers of the report are restricted to viewing it with the filters selected by the sharer of the report.
2. Through the dashboards themselves by sharing via email or giving access to other Power BI account holders. This option allows greater flexibility to the receivers to view the dashboards in real time and choose any filters by which she/he would like to filter the visualizations on the dashboards.

3.3 Anomaly Detection

Many of the datasets in the Hall of Human Life are polluted with obviously wrong entries (e.g. ages over 120, task completion times of 0 seconds, etc.). Our intuition was to specify upper and lower bounds for each quantitative question to label the anomalous data, and use this as the basis for alerting MoS staff when a kiosk is consistently behaving poorly (presumably due to issues with the kiosk's sensors).

3.3.1 Rule-Based Anomaly Detection

An example of the complexities that can arise from the rule-based approach is the ROBO (Robotic arm) kiosk. The TimeSlept metric, which records the number of hours slept in the last day based on the time of day the visitor self-reports to have gone to sleep and woken up, should obviously be in the range of 0 to 24 hours. As the graph below shows (with the bounds marked as black lines, and outliers denoted by red X's), this is not always the case:

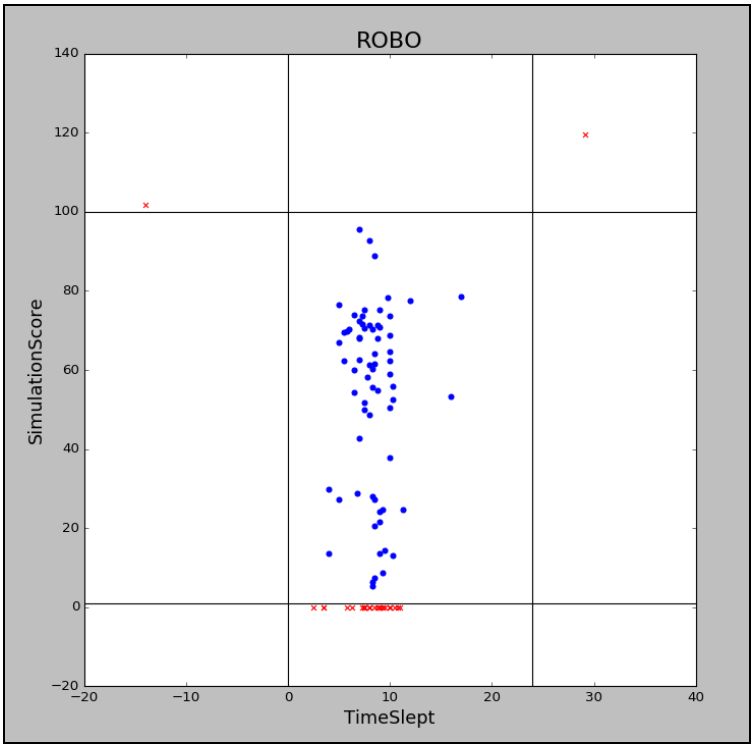

Figure 3.3.1-1: Sample data from ROBO (Roboarm): TimeSlept vs. SimulationScore

A slightly more difficult case occurs with ROBO's SimulationScore data entry, which gives the visitor a score in the range of 0 to 100. Because we want the higher-level hardware failure detection system (to be discussed in detail later) to notice when a problem with the simulation program or controls cause visitors to incorrectly receive a score of 0, we have set the lower limit of the score to 1 instead.

We performed similar analysis as with ROBO on each of the eleven other kiosks that contain quantitative questions (three of the fifteen kiosks have only categorical questions and are immune to outliers).

3.3.2 Complex Anomaly Models

In addition to the simple rule-based system for catching anomalies, we wanted a system that could detect issues with a kiosk using both the underlying distributions of the data (and not just the hard rule bounds) and correlations between the different questions at a given kiosk. Consider the FACE (face recognition) kiosk, which has a clear correlation in its answers, and additionally has no data points which fall outside its bounds:

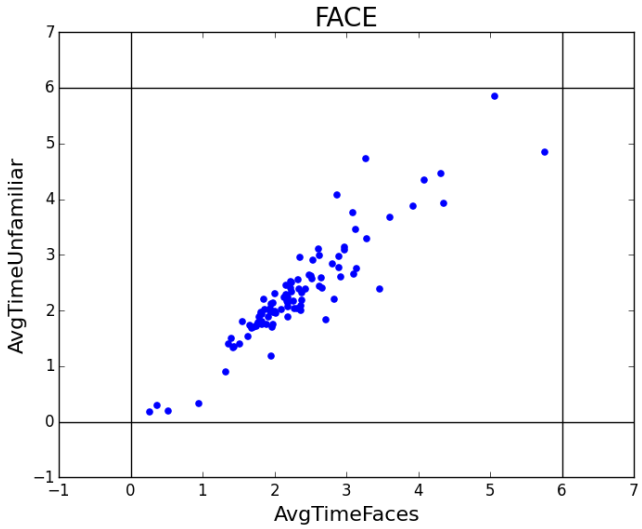

Figure 3.3.2-1: Scatterplot of FACE (face recognition) kiosk sample data, with black lines for bounds.

The rules for FACE are nearly entirely bereft of utility. Thus, we developed the notion that using a more complex anomaly detection algorithm could be able to capture the more detailed properties of each kiosk, and thereby detect when the underlying generative process undergoes a transformation (indicative of a hardware miscalibration or related failure at a particular kiosk).

3.3.2.1 Data Exploration using IPython

While investigating these relatively large datasets (several hundred thousand visitor interactions per kiosk), it became necessary for us to choose development tools which had support for data investigation, visualization, and integration with common data science toolkits. We selected the Jupyter IPython notebook, a web-based interactive computational environment for Python containing an ordered list of input/output cells which can contain code as well as markup and plots. Its selective cell-by-cell execution makes it a powerful tool for computationally intensive data exploration and analysis by allowing for saved states between blocks of code. Additionally, it integrates with the standard open-source Python data science libraries: NumPy (optimized matrix representations for scientific computing purposes), Pandas (building atop NumPy to provide rich data structures and transformations), scikit-learn (implementations for a range of machine learning and data processing algorithms), and matplotlib (provides visualizations).

When running visualizations, we noticed that the DIS ("Are you paying attention?") kiosk's AccuracyDistraction question exhibited bimodal behavior over the course of its entire lifetime, as seen in Figure 3.3.2.1-1 below.

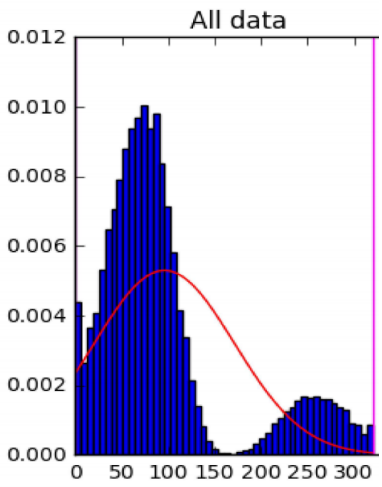

Figure 3.3.2.1-1: DIS AccuracyDistraction distribution over entire exhibit lifetime.

However, when only looking at the past six months, the distribution becomes significantly more normal, as seen in Figure 3.3.2.1-2.

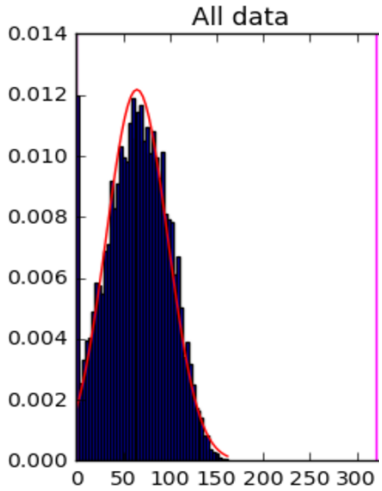

Figure 3.3.2.1-2: DIS AccuracyDistraction distribution over past 6 months.

This led us to suspect that there were significant changes in the DIS kiosk (and potentially others as well, though in a less obvious form) that caused the two different behaviors.

After contacting the MoS, we discovered that the kiosk (and several of the others) had indeed been altered over a year ago. This led us to run further analysis (and training of the anomaly models) only on data collected in the past year, instead of the entire database. The past year encompasses ~2.5 million VisitorAnswers, slightly more than 25% of all the data in the database.

3.3.2.2 Model Selection: Multivariate Gaussian Distribution

Anomaly detection with a sufficient number of labeled anomalies in the training data allows for the problem to be modeled as a typical two-class machine learning problem. However, in many

cases (including our own), there are very few or no labeled anomalies present in training set, requiring the use of a one-class algorithm. Common one-class algorithms include the Multivariate Normal Distribution, one-class Support Vector Machines (SVMs), Principal Component Analysis (PCA), and ensemble algorithms such as random forests. We chose the Multivariate Normal, expressed in the scikit-learn library's EllipticEnvelope package, because of its accurate reflection of the underlying data distribution and consequent robustness towards overfitting [7]. This assumes that the generative process behind the answers to each question is Gaussian, which we found to be a reasonable approximation for nearly all of the questions.

Because Azure ML supports only SVM-based and PCA-based models [8], and because the inherent limits of its drag and drop format would make the individual construction of twelve different models unnecessarily complex and unmaintainable, it was necessary to move the entirety of our logic and algorithm inside a single Python module with Azure ML. The limits of this format include having only two Pandas DataFrames as inputs, and a single DataFrame output.

3.3.2.3 Issues With Singular Covariance Matrices

While pathfinding in an IPython notebook using scikit-learn's EllipticEnvelope anomaly detection algorithm, we encountered non-deterministically occurring errors explaining that some of the covariance matrices computed by EllipticEnvelope were singular (i.e., of rank one; having a determinant of zero; having linear dependence between its rows). Upon further investigation, we first determined that the error was caused by the FIN (finger temperature) exhibit because one of its columns was a linear combination of the other two (in this case, the FingerTempChange was exactly the difference between the two other columns recording FingerStartTemp and FingerStopTemp). Removing one of the FIN columns from the processing routine (while leaving it in the database) preserved the same quantity of information and solved the issue. Further on in our analysis, when the FACE (face recognition) kiosk also exhibited the same error, we were able to fix the model in the same manner.

3.3.2.4 Meta-Parameters for the Anomaly Detection Model

The contamination parameter to the EllipticEnvelope model represents the percentage of outliers expected to be found in the training set (and thus how many points can be safely ignored when fitting the multivariate gaussian to the data). Keeping contamination at 0.0% will ensure that it accepts every data point in the training set, creating a very relaxed set of bounds that is too lenient to be practical for anomaly detection, as seen in the left graph of Figure 3.3.2.4-1. However, increasing the contamination by even a small amount, such as 5.0%, will constrict the bounds of the model significantly such that it becomes useful, as seen in the right graph of Figure 3.3.2.4-1.

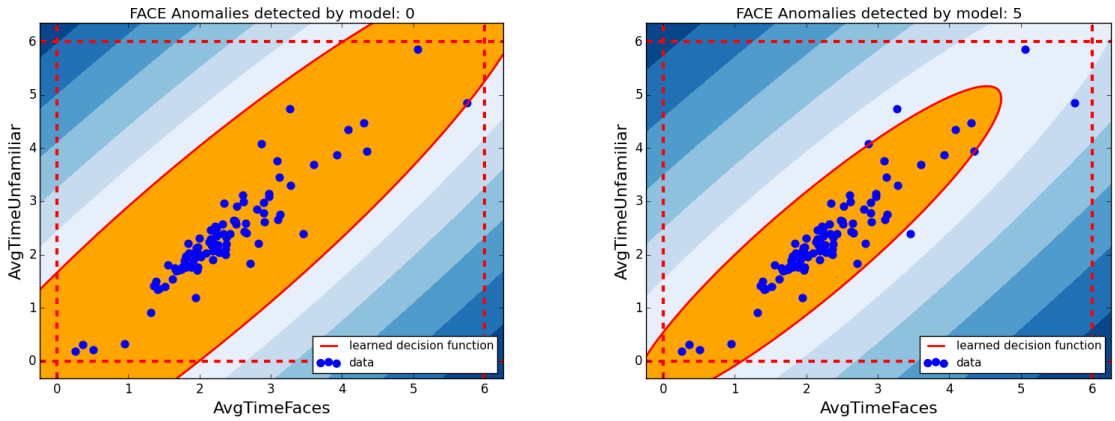

Figure 3.3.2.4-1: scikit-learn EllipticEnvelope on FACE with contamination= 0% (left) and 5% (right)

After experimentation, we settled on a contamination rate of 5%, setting a good balance between an acceptable false positive rate and an effective model.

3.3.3 Hardware Failure Detection

We originally thought that a simple threshold for daily anomaly rates (with the combined rule-based and complex models) would be an effective method for determining if a kiosk was experiencing a hardware failure. However, we quickly discovered that this would be infeasible due to the massive range of average daily anomaly rates across kiosks, from 5% to over 47%. The standard deviations can also vary significantly for the same kiosk across multiple days, ranging from 3% to 18%.

After some experiments in IPython to create a consisted historical model for each kiosk, we chose to set the threshold for a suspected hardware failure at two standard deviations above the average (while providing a straightforward way to modify it). Due to the properties of the normal distribution, this choice implies that of 5% of all days in the past should cross the threshold for a particular kiosk (assuming the daily anomaly rate for each kiosk is roughly Gaussian, as discussed in 3.3.4).

3.3.4 Testing the Complete Model

Because we did not possess any labeled data for previous hardware failures that had occurred, verifying the integrity of our model and refining it was not possible in a traditional sense; the system will have to be run forwards and cross-referenced against an accurate log of hardware failures [9].

However, in order to sanity-check our model, we ran it backwards through our historical training data (in which we can be sure some hardware failures exist). For each of the kiosks, we found both that the anomaly distribution was reasonably bell-shaped, and thus that our threshold of two standard deviations cut off an appropriate proportion of the total days.

Below are three of the kiosks' distributions of daily anomaly rates, overlaid with a fitted normal curve and a vertical line representing the second standard deviation above the mean (our

chosen threshold). The presence of positive skew (a long tail towards the right) supports the idea that there are some days which can be truly considered to have abnormally high anomaly rates, and correspond to hardware failures.

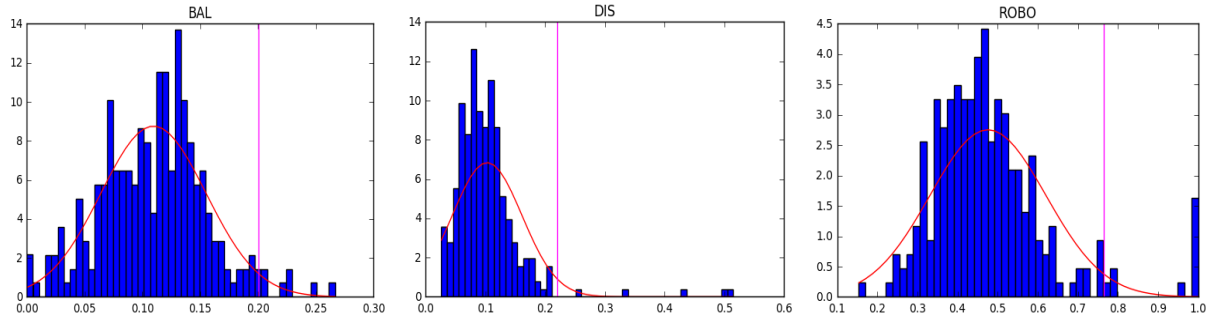

Figure 3.3.4-1: Distributions of the daily anomaly rates for BAL (balance), DIS (distraction), and Robo (Robotic arm) for the past year.

The overall similarity between our findings and the intuitions of the staff supports the credibility of these models. Most interestingly, the especially high anomaly rate for the ROBO (robotic arm) on certain days was supported by the MoS staff, who described the exhibit as particularly prone to issues.

Chapter 4: Results

This chapter discusses the final solution we created, addressing the needs of the MoS as described in Chapter 2. We start with an overview of the architecture of our newly created system. Then, we dive into the solutions for each of the three individual parts of this system introduced in the Methodology chapter. Our solution constitutes a standalone system that serves as the foundation for the new internal system at the HHL.

4.1 Architecture overview

The architecture of our system is shown below in Figure 4.1-1. The fifteen kiosks ("Stations") push to the Azure SQL database, which the Power BI Dashboard reads from. Separately, an Azure Machine Learning service for additional anomaly detection periodically reads from the database, makes predictions, and writes to a log table, which then is picked up by an Azure Function for Hardware Failure Notification , and sent via email to stakeholders if necessary.

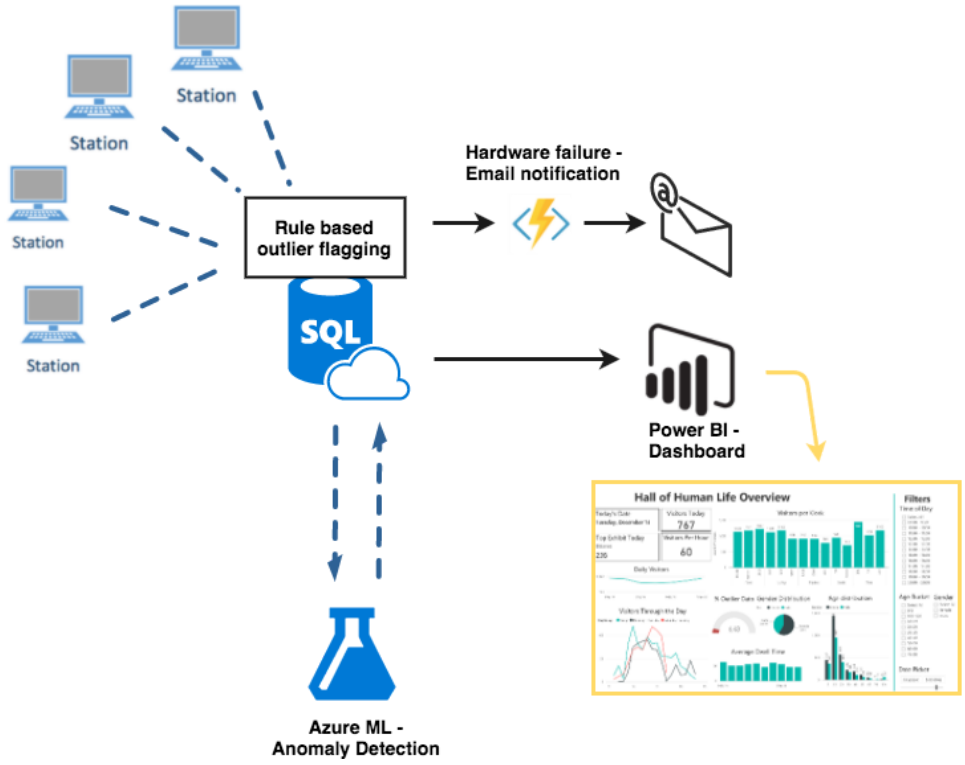

Figure 4.1-1: Final system architecture

4.2 Azure SQL database

The core of the system architecture is as shown above an Azure SQL Database, running on

an Azure SQL Server. The system uses a standalone (not elastic) database in the S2 Standard service tier. It has a maximum storage capacity of 250 gigabytes (GB), well above the current 5GB needed for all relational data, and also supporting the expected future need of the museum for years. The S2 database has a 50 max Database Transaction Units (Azure-specific) and allows for 1200 concurrent sessions [10].

4.2.1 Tables

The diagram in Figure 4.2.1-1 describes the set of tables that make up the database. There are two distinct set of tables, as marked by the two colored boxes in the figure.

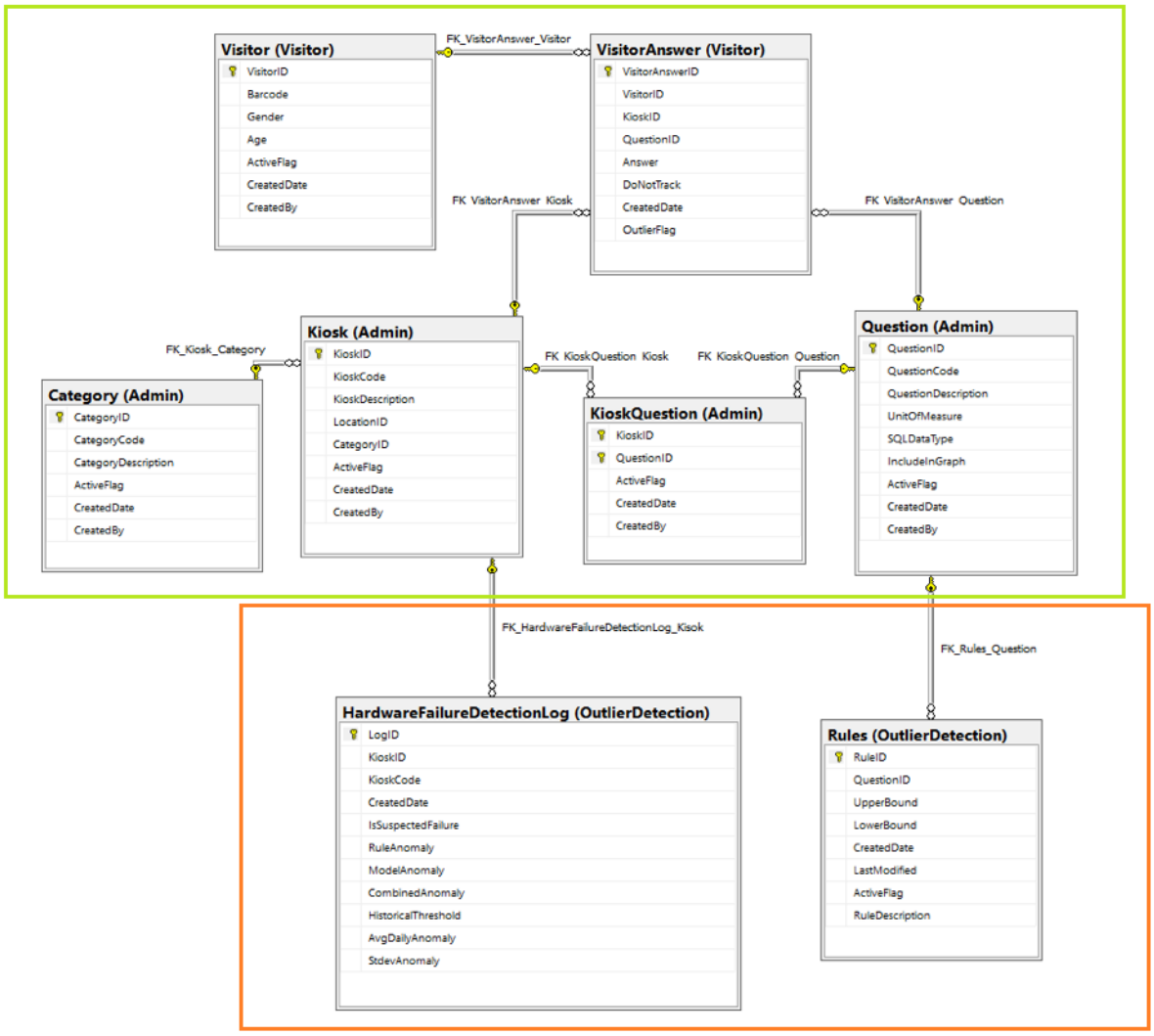

Figure 4.2.2-1 Database diagram representing new project database

The first set of the tables (green) deals with the exhibit logic and storing of information generated by visitors. In the Admin schema, the Category, Kiosk, and Question tables hold information on the different kiosks and all questions they have. In the Visitor schema, the Visitor and VisitorAnswer tables hold the information on each visitor and every answer they created by

answering survey questions or having measurements made. This set reused the naming conventions used for the in-exhibit server at the museum.

The second set of tables (orange) supports both the rule-based and more complex outlier detection work. The set consists of two tables that are both in the OutlierDetection schema. The **Rules** table is used to flag VisitorAnswer data as outliers and inliers at insertion time. It has fields for an upper and lower bound on values obtained from answering the question identified in the QuestionID column, corresponding to a specific question in the Question table, as signified by the foreign key relationship. The ActiveFlag field specifies whether the rule is being enforced. The HardwareFailureDetectionLog table captures the output of the machine learning job (described later) used to discover whether a certain kiosk is producing enough anomalous values to be suspected of having a hardware failure. The IsSuspectedFailure field contains the result for each job run, and each row is associated with a specific kiosk through the KioskID field, again tied to the Kiosk table using the foreign key relationship shown in the figure. This is discussed further in section 4.4.2 Prototype Hardware Failure Detection Model.

4.2.2 Views

The following views were created to support and surface various insights from visitor data in the Power BI dashboard.

View	**Description**
vw_DatesInAnswer	Unique dates during which visitors generated data in the exhibit.
vw_VisitorAnswer_DetailedDate	Flattened view of visitor answers and data on kiosks, categories and visitors corresponding to their identifying fields in VisitorAnswer. Includes the visitor answer as well as kiosk code, category code, and visitor age and gender. Additionally, it includes various formats of the timestamp for the answer such as day of week, month, year, and just the time of the day.
vw_VisitorsPerHourToday	Average number of visitors per hour today.
vw_HourlyVisitors	Number of unique visitors interacting with a kiosk for each open hour of the day. These values are computed daily for the entire historical dataset. The following distinction (grouping) is made on the specific day of the week: ● Monday-Thursday ● Friday ● Saturday – Sunday
vw_Kiosk_DailyCounts	Counts of unique visitors interacting with every kiosk. An interaction means answering one or more questions for a certain kiosk. These values are computed daily for entire

	historical data.
vw_DwellTimes	Average dwell time (time spent between first and last interaction with kiosks) in the exhibit across all visitors. This value is computed daily for the entire historical dataset, and only takes into account the visitors who had interactions with two or more kiosks.
vw_AnomalyBreakdown	Counts of anomalous interactions, total interactions and the percentage of anomalous interactions for each kiosk. These values are computed daily for the entire historical data set. An anomalous interaction is defined as one in which any answer (among all answers) a visitor generates at a kiosk is considered an outlier by the rules.
vw_CompletionRates	Average number of questions completed for each kiosk by all visitors, along with total number of available questions. As the name suggests, the intention is to provide insight into completion rates for visitors. These values are computed daily for the entire historical dataset.

4.2.3 Database optimization

The dashboards support a full historical view of the data. To do so, many of the views read and aggregate all the data in the largest table, VisitorAnswer. It is thus important that those views are optimized. One major optimization done is to create a non-clustered index on the CreatedDate column, and to include the Answer, KioskID, QuestionID, and VisitorID columns. This optimization was suggested by the SQL Database Advisor, a feature in Azure SQL Database, after running the database and using the dashboard for some time.

4.3 Power BI Dashboards

We provide two dashboards, an “Overview” dashboard displaying high-level visitor metrics about the entire Hall of Human Life exhibit, and a “Detail View” dashboard displaying in-depth visitor metrics for individual kiosks as well as comparisons between kiosks and the exhibit as a whole. The Detail View also includes information about outlier data for each kiosk, as well as the average completion rate for questions at each kiosk.

4.3.1 Overview Dashboard

The goal of the Overview dashboard is to provide, at a glance, the most important metrics about the entire exhibit. It also allows for the ability to filter these metrics to gain a clear picture of visitor behavior for different types of visitors over various periods of time. These metrics are

presented as visualizations in a Power BI dashboard, as shown in Figure 4.3.1-1, separated from the filters on the right by a divider line.

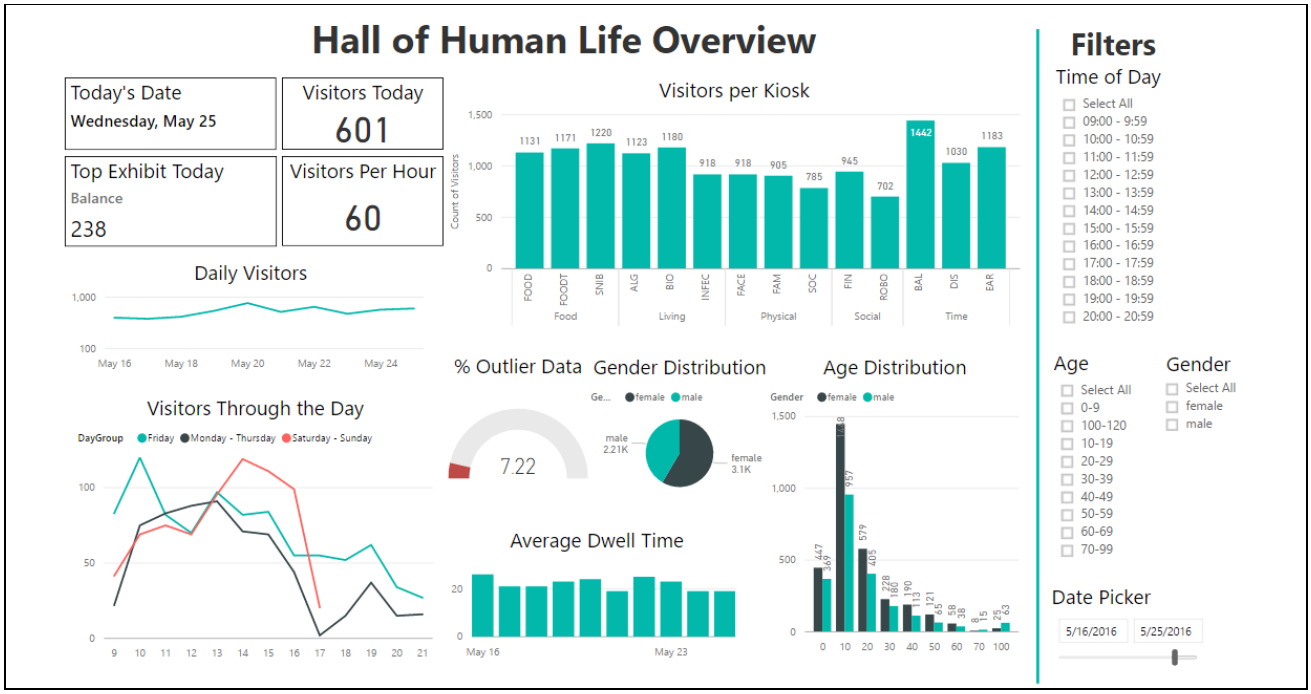

Figure 4.3.1-1 Overview Dashboard

4.3.1.1 Filters

The Overview dashboard features four filters that enable users to filter the visualizations by information about the visitors, to learn more about particular kinds of visitors. The filters presented are:

Filter	**Description**
Time of Day	This slicer allows users to pick the hour(s) of the day of by which to filters the visualizations. A slicer narrows the portion of the dataset shown in the other visualizations on the page. The times of day are 59-minute buckets on the hour, every hour from 9 am to 8.59 pm since the exhibit opens at 9 am every day, and closes, at the latest, at 9 pm. This slicer allows multiple selection.
Age Bucket	This slicer allows users to pick buckets of ages of visitors for filtering the visualizations. The age buckets (in years) are: • 0-9 • 10-19

	● 20-29 ● 30-39 ● 40-49 ● 50-59 ● 60-69 ● 70-99 ● 100-120 This slicer allows multiple selection.
Gender	This slicer allows users to pick the gender of visitors for filtering the visualizations and allows multiple selection.
Date Picker	This slicer appears as a date range picker, allowing users to pick a start and end date for which by which the filter the visualizations. If a user wishes to filter the visualizations for only one day instead of a date range, s/he can set the start and end date to the same date. The date picker is a filter that is always used, since it is necessary to pick a date range for the data users wish to see metrics for, and is crucial in helping users gain insights into visitor behavior over different periods of time historically.

Figure 4.3.1.1-1 shows the Overview dashboard filtered for male visitors in the age range 10-29 years who visited between the hours of 9:00 am and 11:59 am between the 16th of May 2016 and the 25th of May 2016.

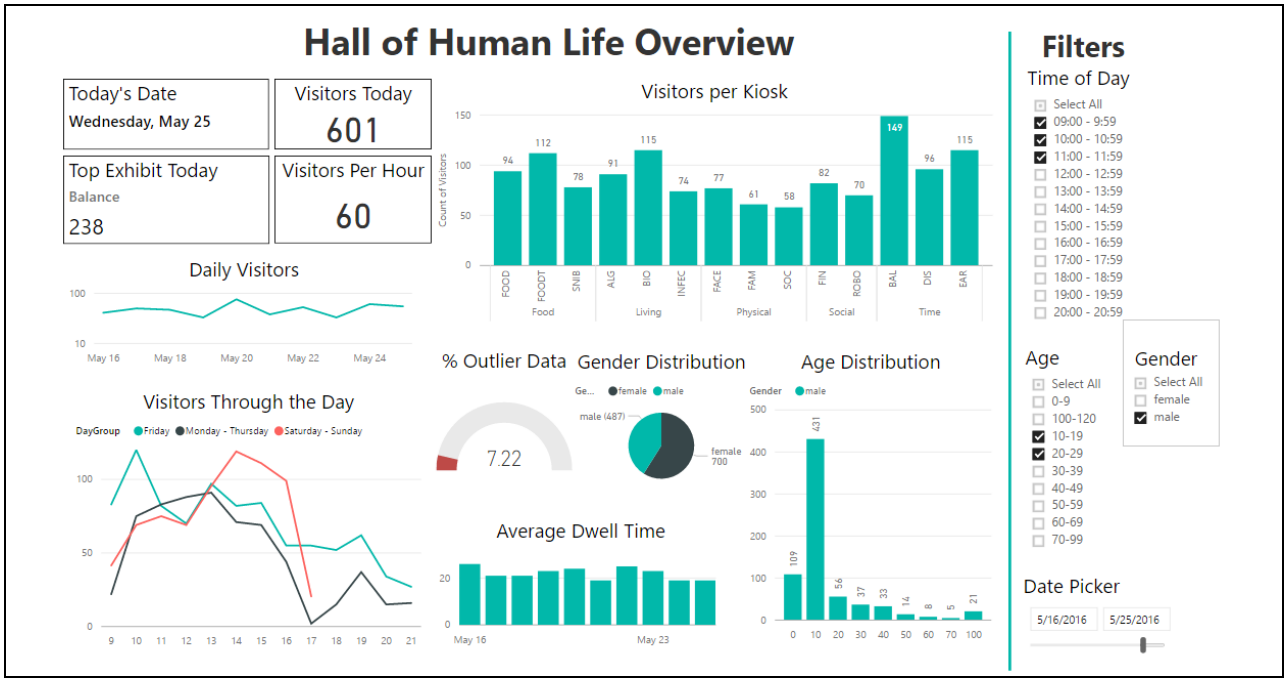

Figure 4.3.1.1-1 Overview dashboard with filters applied

4.3.1.2 Visualizations

The Overview dashboard contains both static and interactive visualizations. The static visualizations are presented only for the current date on which the dashboard is being viewed. These are presented as cards in the top left of the dashboard, as shown in Figure 4.3.1-1, and are not affected by filters in the dashboard. Cards are simple visual boxes in Power BI containing textual information. The static visualizations in the Overview dashboard are:

Visualization	**Description**
Today's Date	This card displays the current date that the dashboard is being viewed on.
Visitors Today	This card displays the number of unique visitors who have interacted with kiosks in the exhibit on the current day.
Top Exhibit Today	This card displays the exhibit with which most visitors have interacted on the current day and the specific count of visitors.
Visitors Per Hour	This card displays the hourly average of visitors interacting with kiosks in the exhibit per hour so far for the current day.

The other visualizations presented in the dashboard are dynamic, i.e., they can be filtered.

They are also interactive, hovering over a section of a visualizations results in a tooltip appearing next to the cursor, showing more information about the data in that section. The dynamic visualizations in the Overview dashboard are:

Visualization	**Description**
Daily Visitors	This line graph plots the number of visitors to the exhibit on the y-axis against each day for the date range selected in the filters.
Visitors Through the Day	This line graph plots the number of visitors on the y-axis versus the hour in the day when the visitors interacted with kiosks in the exhibit, for the date range selected in the filters. There are three different line graphs, one displaying an average of visitors by the hour for the Monday, Tuesday, Wednesday and Thursday visitors at the exhibit, one displaying an average of visitors by the hour for visitors on Saturday and Sunday in the date range selected, and finally, one showing the average visitors by hour on the Fridays in the date range selected. This visualization is particularly helpful to museum staff in planning their staffing by looking at peak times at certain days in the past.
Visitors per Kiosk	This bar chart displays the total number of visitors on the y-axis against the kiosk they interacted with on the x-axis, for the date range selected in the filters. The kiosks are stratified by the category to which they belong. This visualization helps museum staff keep track of their most popular exhibits and figure out which exhibits are not being visited as much.
% Outlier Data	This gauge displays the percentage of data points in the exhibit that were outliers for the date range selected in the filters. Its minimum value is 0, and its maximum value is 100, interpreted as percentage values. This visualization helps museum staff keep

	track of the overall health of the exhibits. If this value is high, the staff can view the Percentage Outlier visualization in the Detail View dashboard for details on which kiosks are exhibiting outlier data.
Gender Distribution	This pie chart displays the number of male visitors and female visitors as the two sections of the pie for the date range selected in the filters. This visualization does not interact with the "Gender" filter, because such an interaction would result in the metrics in this visualization to show for only one gender, hence occupying the entire pie.
Age Distribution	This clustered bar graph displays the number of visitors on the y-axis against the age bucket they belong to, on the x-axis. Each age bucket features two bars, one for male visitors and one for female visitors. This visualization does not interact with the "Age Bucket" filter because such an interaction would result in this visualization showing visitor numbers for only the selected age groups, whereas the Visitors Per Day visualization already does that when the "Age Bucket" filter is in use.
Average Dwell Time	This bar graph displays the average dwell time of visitors in minutes on the y-axis against each day for the date range selected in the filters. The average dwell time is the average amount of time a visitor spends in the Hall of Human Life exhibit. This visualization helps museum staff figure out which days have visitors spending more time in the exhibit, and which days have them spending less time. Particularly high or low values here may spark a conversation about the circumstances of that particular day.

4.3.2 Detail View

The goal of the Detail View dashboard is to provide detailed metrics at the kiosk level, as well as the ability to compare metrics for kiosks against each other and against the exhibit as a whole. As with the Overview dashboard, the detail view provides filters to narrow down the metrics by pertinent information about the kinds of visitors and information about their visits. Figure

4.3.2-1 depicts the Detail View dashboard without any filters applied.

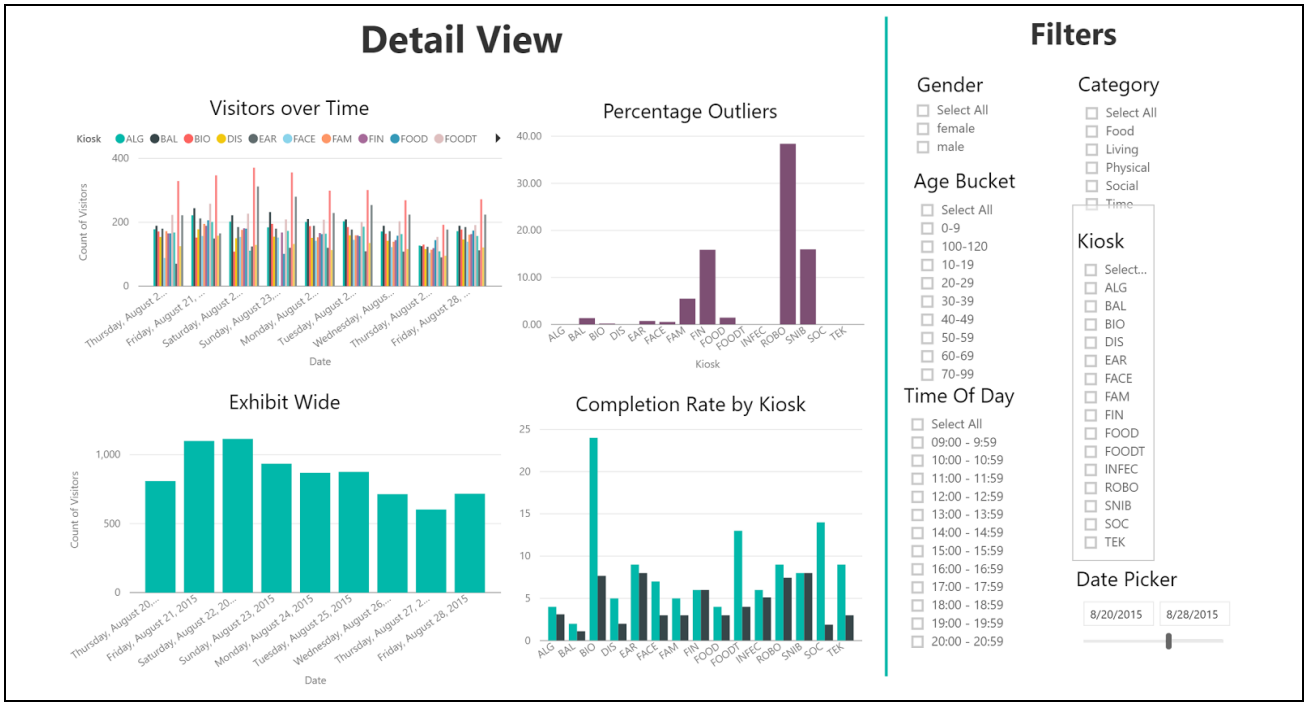

Figure 4.3.2-1 Detail View dashboard

4.3.2.1 Filters

The Detail View dashboard provides the same filters as the Overview dashboard: Time of Day, Age Bucket, Gender, Date Picker. Since the granularity of the visualizations in this dashboard is at the kiosk level, we provide the following additional filters:

Filter	**Description**
Category	This slicer allows users to select a category out of the five categories of kiosks at the exhibit and allows for multiple selection. When a category is selected, all kiosks in the category are automatically selected, and the Kiosk filter only displays the kiosks in the selected category (see the next filter for more information on the “Kiosk” filter).
Kiosk	This slicer allows users to select a kiosk out of the fifteen kiosks at the Hall of Human Life and allows for multiple selection.

Figure 4.3.2.1-1 shows the Detail View dashboard with filters applied to narrow down the metrics in the visualizations for visitors age 20-29 and 40-49 who visited the Hall of Human Life

between August 20, 2016 and August 28, 2016 from 12:00 pm -12:59 pm and interacted with the Biophilia (BIO), Ear Measurement (EAR), and Infections (INFEC) kiosks.

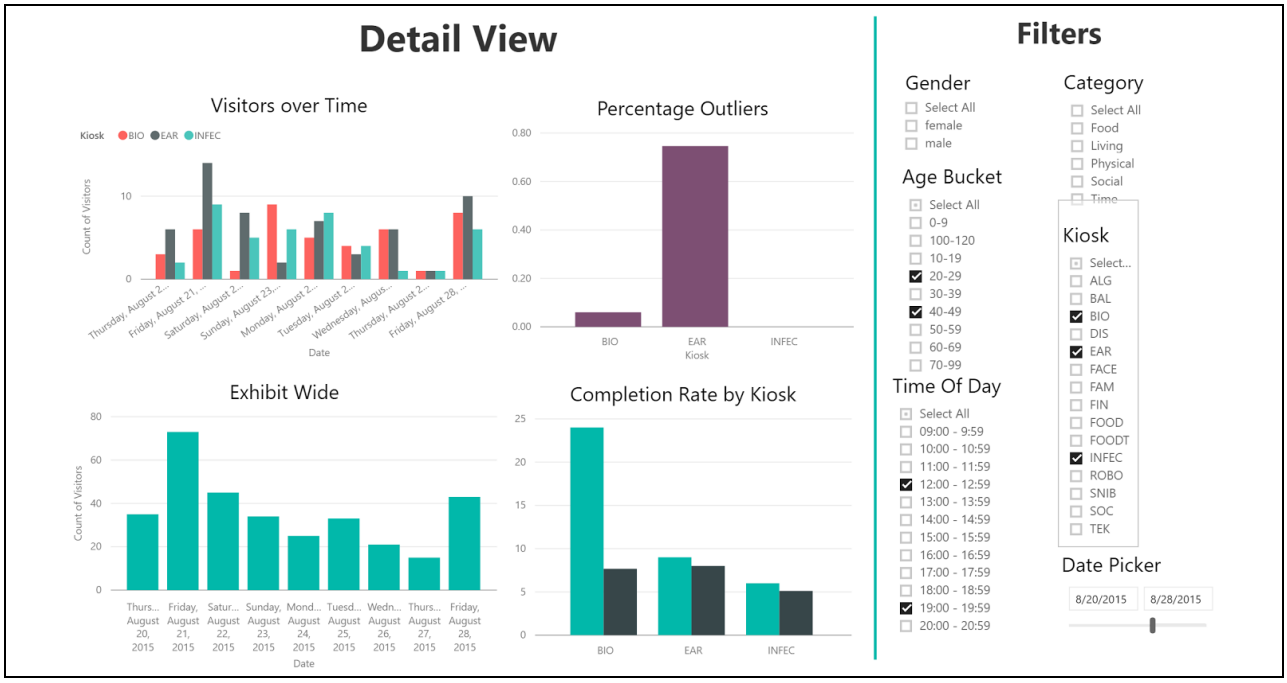

Figure 4.3.2.1-1 Detail View dashboard with filters applied

4.3.2.2 Visualizations

The Detail View dashboard features four visualizations, as seen in Figure 4.3.2-1, which are:

Visualization	**Description**
Visitors Over Time	This clustered bar chart features bars for the selected kiosks for each day in the date range selected in the filters on the x-axis against the number of visitors to those kiosks on the y-axis. The legend maps a bar's color to the kiosk that corresponds to it. When combined with the Exhibit Wide visualization, and with the help of the Category and Kiosk filters, these two visualizations help museum staff compare visitation numbers between kiosks, between a kiosk and the entire exhibit, and between a category and the entire exhibit.
Exhibit Wide	This bar chart features the total number of

	visitors for the entire Hall of Human Life exhibit on the y-axis against each day in the selected date range, on the x-axis.
Percentage Outliers	This bar chart shows the average percentage of outliers for the selected kiosks against the kiosks themselves, for the selected date range. This visualization helps museum staff keep track of the health of individual kiosks and can cause a check on a particular kiosk if it shows a high value in this visualization.
Completion Rate by Kiosk	This clustered bar chart features a y-axis that depicts number of questions against an x-axis that features two bars for each kiosk. The first bar is for the number of available questions for the kiosk and the second bar is for the average number of questions answered for that kiosk by visitors in the selected date range. This visualization helps museum staff figure out which kiosks are being consistently answered to completion and which kiosks have questions that are consistently being left unanswered.

4.3.3 Dashboard Performance

The Power BI dashboards read data from the Azure SQL database in DirectQuery mode which is a live connection without any data being imported into Power BI, and the perceived view of the exhibit is near real-time. Upon refreshing the dashboard, the user receives an up-to-date view of all data available through the views. A refresh may result in a short delay of a few seconds if a large amount of data is changed or added. Once the visualizations have been updated, the user is able to filter and use the dashboard with instant changes to the visualizations. Refreshing the dashboard is done by clicking on the ribbon in the top right corner of the screen and selecting "Refresh dashboard tiles."

4.3.4 Sharing Dashboard Insights

The Power BI dashboards can be shared with others in different ways. Live dashboards can be shared with others, and static reports can also be generated from the dashboards, with filters selected.

4.3.4.1 Live Dashboards

Live dashboards can be shared by clicking on the ribbon in the top right corner of the screen and selecting “Share dashboard.” This brings up a new view open to the “Share” tab that allows users to share the Power BI dashboard to recipients via email, and choose whether or not to allow the recipients to share the dashboards themselves. Another tab in this view, “Access,” allows users to grant access to other Power BI users.

4.3.4.2 Reports

Static reports can be shared from the “Reports” tab under the left toolbar in Power BI. Once a report is selected, the user can click on “File” in the report ‘s top toolbar and can choose to save an offline pdf copy of the report, print the report, publish it to the web as an embedded report for public access, generate a link to securely embed the report in Sharepoint Online, export the report as a PowerPoint presentation, or to download the report as a .pbix file for offline access in Power BI desktop.

4.4 Anomaly Detection

The result of our investigation into the complete HHL dataset was both a rule-based outlier flagging system implemented inside the Azure SQL database, and a prototype machine learning model for recognizing potential hardware failures using Azure Machine Learning.

4.4.1 Rule-Based Outlier Detection

For each of the twelve kiosks with quantitative questions (of which there were a total of 30), we set upper and lower limits for what should be considered acceptable answers. We then modified the VisitorAnswer entry procedure in the Azure SQL database to label any answers that did not fall within the acceptable range as an outlier. The reasons outlier data may be produced include that the kiosk in question has a hardware or software issue, or that the visitor submitted (intentionally or not) incorrect data. These labels can then be used for further analysis or reporting, such as in the Power BI dashboard, or for use in a more sophisticated anomaly detection model (as discussed in Section 4.4.2).

4.4.2 Prototype Hardware Failure Detection Model

In order to detect when a kiosk undergoes a suspected hardware failure or miscalibration, we designed a system capable of automatically performing the necessary analysis and notifying stakeholders.

4.4.2.1 Expected System Overview

The expected complete system for detecting hardware failures has two components in addition to the underlying Azure SQL database: an Azure Machine Learning (ML) web service, and an Azure Function for serverless timer-based code execution. In production, the Azure ML service

will query the Azure SQL database for data, predict whether each kiosk is suspected of having a hardware failure, and then write its findings to a log table in the SQL Database. These new records are then picked up by an Azure Function, which uses SendGrid (an email delivery and management service) to notify a list of subscribers of the suspected hardware failure via email. The Azure ML model relies on the assumption that this process will be run precisely once per day, but can theoretically support any arbitrary length of time (as discussed further in 4.4.2.2: Model Details).

Due to constraints on our development environment and timeline preventing us from fully deploying the two services, our final core deliverable was a fully running Azure ML Experiment that can later be adjusted, deployed as a web service, and integrated with SendGrid as appropriate.

4.4.2.2 Model Details

The core of this system is the generalized machine learning model, created in a Python module within Azure ML, that can be trained independently for each kiosk. The model itself is actually a combination of two separate but related models (custom Python classes) for each kiosk, the Anomaly Model and Historical Model. The flow of the entire system is shown below in Figure 4.4.2.2-1.

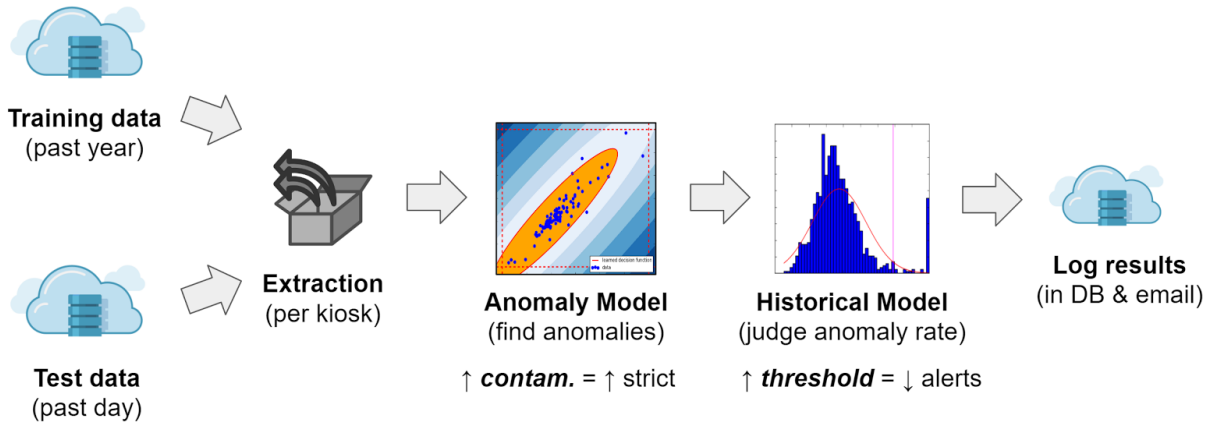

Figure 4.4.2.2-1: Flow of the Azure ML service.

The Azure ML service begins by ingesting the training data (the entire set of VisitorAnswers excluding today's date), and the test data (all VisitorAnswers for today's date). It then transforms the data from a flattened series of VisitorAnswers (with columns of [VisitorID, KioskID, QuestionID, Answer]) into a compressed format with columns [VisitorID, Date, Answer1, Answer2, …] for each kiosk, dropping partially filled rows. This makes each visitor's interaction with a kiosk a feature vector, allowing the set of all such interactions to be used in the standard machine learning matrix format (where each row represents a data sample, and each column represents a feature).

The training data is then filtered to remove all the rule-based outliers and passed into the Anomaly Model (one per each of the twelve relevant kiosks), which is trained using a standard one-class Multivariate Gaussian anomaly detection algorithm. This model has an optional "contamination" parameter which can be increased to tighten the allowed bounds of the fitted multivariate Gaussian distribution, which we have currently set to 0.05 (to represent that 5% of the training data can be considered anomalous).

Next, the Hardware Model (again, one for each kiosk) runs the Anomaly Model backwards in time over the entire training data set (including the rule-based outliers). The Historical Model uses the results of this to compute the average and standard deviation of the daily anomaly percentage (including rule-based outliers) for each kiosk (the results of which are shown in the rightmost two columns of Table 4.4.2.2-1). The model then sets a Historical Threshold two standard deviations above the mean (which can be adjusted easily), representing the maximum acceptable anomaly rate for that kiosk.

Having trained both an Anomaly Model and Hardware Model for each kiosk, we pass the test data (including all outliers flagged by the rules) first into the Anomaly Model, which produces the Model Anomaly score representing the anomaly percentage of test data. This is combined with the results of the rule-based outlier detection to produce a Combined Anomaly score. The Historical Model then compares this to the anomaly percentage for the test data (the current day) and outputs whether or not the kiosk's performance falls into the acceptable range. The format of the output, with an example set of test data, is shown below in Table 4.4.2.2-1:

	Example Test Results				Trained Model		
Kiosk	IsSuspected Failure	Rule Anomaly	Model Anomaly	**Combined Anomaly**	**Historical Threshold**	Avg Daily Anomaly	Stdev Anomaly
DIS	0	0.00%	4.71%	4.71%	15.68%	5.26%	5.21%
BAL	0	2.19%	8.73%	8.73%	12.77%	6.46%	3.16%
EAR	0	0.66%	5.43%	5.43%	11.13%	6.16%	2.48%
SNIB	0	23.27%	3.03%	23.48%	40.53%	19.26%	10.63%
FOOD	0	1.43%	6.76%	6.76%	12.06%	6.53%	2.77%
BIO	0	0.74%	5.99%	6.16%	10.12%	5.54%	2.29%
FAM	0	14.26%	15.46%	19.93%	38.28%	19.82%	9.23%
SOC	0	0.00%	4.56%	4.56%	10.81%	4.96%	2.93%
FACE	0	0.73%	5.30%	5.30%	10.20%	5.36%	2.42%
ROBO	0	38.87%	16.94%	42.55%	76.62%	47.91%	14.36%
TEK	0	0.27%	7.70%	7.70%	10.60%	5.08%	2.76%
FIN	0	20.16%	7.92%	22.70%	56.66%	24.55%	16.05%

Table 4.4.2.2-1: Results of the Azure ML Service logged, including the characteristics of the trained model.

Finally, the current date and time are added to each entry before pushing the entire set of results into the Azure SQL database logging table.

4.4.2.3 Drawbacks, Complications, and Runtime

As mentioned in Section 3.3.2.2, limitations of the Azure ML environment necessitate the entire program logic to reside inside a single Python module, with no option to store the trained Anomaly Model or Historical Model. These modules have additional limitations in that they are only able to take in two data sets as input, and export one data set for output. This results in an experimental runtime of 4-5 minutes in the Azure-supported IPython notebook environment, and 10-20 minutes as an Azure ML Experiment. Although this fits well within the time requirements for running the service once per day (or even once per hour), methods for reducing the computation time are discussed in Chapter 5.

This also creates the need for the list of kiosks, questions, and associated rules to all be duplicated within the main Python script. This allows for significantly easier and more independent testing of the script without reliance on the SQL database for any anomaly data but creates multiple points of truth for the outlier rules.

Chapter 5: Future work

Our prototype system hosts data in the cloud, flags outlier data, and provides staff with actionable insights into visitor behavior at the kiosks. It currently exists entirely independent of the internal MS SQL Server at the HHL, and thus the individual kiosks will need to be modified to push and poll from the new Azure database. In this section, we describe future work that can move our system into production-readiness, as well as suggest further improvements and potential paths of exploration.

5.1 Exhibit Heat Map Visualizations

The MoS staff directly in charge of the exhibit have expressed interest in tracking and visualizing exhibit usage patterns in different ways. One interesting type of visualization that came up during discussions was heat mapping. Many heat mapping solutions rely on the visualization of continuous tracking of movements, which is currently not available in the exhibit data [15]. However, all visitor kiosk interactions are timestamped and can thus be used to visualize the popularity of various kiosks throughout the day, as well as the order in which visitors interacted with kiosks (but not the exact path taken in the exhibit). A heat map visualization can be used to better understand popular kiosk combinations and whether the interest in certain categories looks different during the day or even larger time frames.

5.2 Dashboards

The Power BI dashboards currently provide key filters to narrow down the visualizations. However, future work would involve more filtering capabilities, such as narrowing down visualizations by season of visit.

Currently, the dashboard displays visitors dwell times at the entire exhibit. In the future, the dashboard could display dwell times at individual kiosks if a change is made to the data pushed from the kiosks to record a timestamp with each answer rather than batch a visitor's answers from one kiosk with one timestamp.

Once the hardware detection system is further tested and improved, there could be another dashboard that records details about each hardware failure that occurs in the museum that could be used to track the performance of kiosks.

5.3 Anomaly Detection

As the most exploratory part of our project, the work in anomaly detection has not yet been evaluated and will likely require adjustments before reaching desired performance. After deploying the Azure Machine Learning service and integrating it with the email notification function, it will need to be tested and optimized.

5.3.1 Deployment and Email Notification Integration

To provide the full alert capability of the hardware anomaly detection system, the Azure ML experiment must be deployed as either a Batch Execution Service (BES) or Request-Response Service (RRS) hosted by Azure. Batch Execution handles high-volume asynchronous scoring of a batch of data records, outputs to Azure Blob Storage, and is useful when responses are not needed immediately. By contrast, Request Response uses a REST API (a modern stateless architecture style typically using HTTP) to achieve a low-latency and highly scalable service useful for when the consuming application expects a response in real time [12]. We recommend use of the Request-Response service due to the flexibility of the real-time RESTful interface. Although we did not have time to fully deploy and test the service in this manner, Microsoft provides comprehensive documentation on the process, which should be fairly straightforward [13].

Separately from the Azure ML service, an Azure Function App will need to be created in order to periodically call the ML service and send an email notification depending on its results. Depending on limitations with the Azure ML environment, it may be necessary to have the test data input to the ML service (and potentially the output logging to the Azure SQL Database) be handled within the Function App as well. We recommend using SendGrid (an email delivery and management service that integrates with Azure) to handle the email notifications, which will require establishing an official point of contact and SendGrid account creation in addition to the code written in the Function App. Microsoft provides comprehensive online documentation on integrating SendGrid with Function Apps [11]. The completed Function should, at a determined time once per night (when exhibits are inactive), call the anomaly detection Azure ML service using data from the past day (retrieved from the Azure SQL database), log the results back into the database, and send an email to the Hall of Human life administrators if any hardware failures are suspected.

5.3.2 Tuning the Hardware Failure Detection System

Because we did not possess any labeled data for previous hardware failures that had occurred, verifying the integrity of our model and refining it was not possible in the traditional sense of scoring based on historical records. Thus, the system will have to be run forwards and cross-referenced against an accurate log of hardware failures. There are three different parameters that can be tuned to increase utility.

The first is the contamination parameter used by the Anomaly Models to compute multivariate gaussians, which can be raised to increase the percentage of anomalies flagged. Intuitively, the contamination should roughly correspond to the percentage of days in the historical training data for which there were hardware failures producing abnormal data. We have currently set the contamination to 0.05, or 5%.

The second is the threshold parameter used by the Historical Model to determine the daily anomaly rate considered suspicious enough to produce an alert. In order to do this correctly, detailed records of which days certain kiosks appear to stop working must be collected. We have currently set the threshold to 2, signifying two standard deviations above the mean daily anomaly rate to be considered worthy of flagging as a suspected hardware failure.

Adjusting the third parameter would require some slight code refactoring to change the time window over which the test data is collected (in our implemented system, assumed to be the current day). Changing the window would also likely correspond to altering the frequency with which the Azure ML service is called. For example, if a much higher feedback frequency was desired, the window could be changed to be four hours long, and the service could be called every hour, achieving an effect reminiscent of a moving average.

5.3.3 Vital Tests and Additions

Crucially, if the anomaly detection system is to reach its full utility potential, robust support for providing custom date ranges for training data for each kiosk must be introduced. This would allow administrators to train the machine learning system on only dates for which there was certainly no hardware failure. Adding this would vastly improving the integrity of the model, which is currently forced to train on all historical data and assumes that the contamination parameter accounts for all hardware failures in the history.

Taking it a step further, going backwards through the entire database log and manually noting on which days certain kiosks experienced hardware failures would allow the whole system to be adjusted with clear accuracy targets, and would also allow for the use of alternative (or supplementary) machine learning algorithms that rely on two or more classes of labelled training data (e.g. two-class Support Vector Machines).

5.3.4 Machine Learning Service Optimization

Due to limitations of performing all the Azure ML computations inside a Python module, the training of the models takes 5-15 minutes, while actually running the model on the test data should take only a few seconds.

If speedup of the system is necessary, the results of the Historical Model (containing the mean and standard deviation of the daily anomaly rates per kiosk) could be serialized and fed back into the program for a marginal speed increase at the cost of a less elegant design. A more radical approach would exploit the Python pickle library to serialize the actual Python objects containing the trained Anomaly Models and save them as datasets. This would not only reduce runtime to only a couple minutes but would make the system significantly more complex and less maintainable. Finally, there is the potential that the parallelizable nature of the problem (performing the same computation twelve times for the twelve kiosks) could be exploited for a significant speed increase.

5.3.5 Interface for Rule Modification

Tweaking the anomaly detection systems by adjusting the rules may become common enough to merit implementation of a simple web interface for changing the rule bounds for individual kiosks. Azure Web Applications, or another similar Azure service, should provide the needed functionality. This could also be extended to support parameter adjustments for the hardware failure detection service in Azure ML. Note that in our current implementation, for pragmatic reasons, the rule set is duplicated in both the Azure SQL database and the Azure ML service.

Chapter 6: Conclusions

The HHL at the MoS had faced the problem of having its data stored in an on-premise database with a rudimentary Microsoft Excel dashboard and no filtering of outlier data points. Our complete solution provides the Museum of Science with a prototype system in the cloud that labels outlier data, visualizes and delivers insights on visitor behavior and kiosk health at the Hall of Human Life, and runs machine learning models to to automatically detect likely failures in kiosk hardware.

Although our system is a prototype, it can be used to examine data recorded up to and including the month of October 2016. Staff from the HHL can use it to gather insights from historical date ranges and seasons. This may include investigating historical visitor numbers and peak visit times during the day to anticipate daily staffing needs, and discovering and fixing kiosks with excessively high anomaly rates.

While our system is localized to the Hall of Human Life, it can be extended and/or modified to be used at other exhibits. As such, our solution forms an important first step in the future of data storage and analysis, as the MoS looks to dramatically increase the number of interactive and connected exhibits to begin ushering in a new age of museum experiences.

Our system has implications on a much wider scale. Our foundational storage, analysis, and anomaly detection system can be generalized to to museums around the world that collect significant visitor data. In addition to improving the overall visitor experience, it can benefit museum staff as they can use these indicators to perform maintenance, improve fundamental exhibit qualities, and allocate staff and resources. As a one of the world's largest science centers and New England's most attended cultural institution [14], the steps taken by the Museum of Science to modernize its data processing system have the potential to influence and benefit other museums around the world.

References

1. "Museum History", Museum of Science, Boston. Retrieved 10 December 2016 from https://www.mos.org/history
2. “Hall of Human Life: An Educator’s Guide”. Museum of Science, Boston. Retrieved 10 December 2016 from https://www.mos.org//sites/dev-elvis.mos.org/files/docs/offerings/mos_educator-guide_hhl.pdf
3. “Hall of Human Life”. Museum of Science, Boston. Retrieved 10 December 2016 from https://www.mos.org/exhibits/hall-human-life
4. “Hall of Human Life: Explore the Five Environments”. Museum of Science, Boston. Retrieved 10 December 2016 from http://exhibits.mos.org/explore-the-exhibit/5-environments/
5. “Microsoft Azure: Cloud Computing Platform and Services”. Microsoft Azure. Retrieved 10 December 2016 from https://azure.microsoft.com/en-us/?b=16.48
6. “Use DirectQuery in Power BI Desktop”. David Iseminger. Power BI Documentation. Published 3 November 2016. Retrieved 10 December 2016 from https://powerbi.microsoft.com/en-us/documentation/powerbi-desktop-use-directquery/
7. “The Multivariate Gaussian Distribution”. Chuong Do. Stanford University CS 229: Machine Learning. Published 10 October 2008. Retrieved 10 December 2016 from http://cs229.stanford.edu/section/gaussians.pdf
8. “Anomaly Detection”. Microsoft Azure Documentation. Published 11 July 2015. Retrieved 10 December 2016 from https://msdn.microsoft.com/en-us/library/azure/dn913096.aspx
9. “Developing and Evaluating an anomaly detection system”. Andrew Ng. Coursera: Stanford Machine Learning Course. Retrieved 10 December 2016 from https://www.coursera.org/learn/machine-learning/lecture/C8IJp/algorithm
10. “SQL Database options and performance: Understand what's available in each service tier”. Rabeler et al. Microsoft Azure Documentation. Published 11 January 2017. Retrieved 11 January 2017 from https://docs.microsoft.com/en-us/azure/sql-database/sql-database-service-tiers
11. “How to Send Email Using SendGrid with Azure”. Thomas et al. Microsoft Azure Documentation. Published 14 January 2016. Accessed 30 December 2016 from https://docs.microsoft.com/en-us/azure/app-service-web/sendgrid-dotnet-how-to-send-email
12. “How to consume an Azure Machine Learning Web service that has been deployed from a Machine Learning experiment”. Ericson et al. Microsoft Azure Documentation. Published 5 January 2017. Accessed 30 December 2016 from https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-consume-web-services#batch-execution-service-bes
13. “Deploy an Azure Machine Learning Web Service”. Ericson et al. Microsoft Azure Documentation. Published 6 January 2017. Accessed 7 January 2017 from https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-publish-a-machine-learning-web-service
14. “About the Museum of Science, Boston”. Engineering is Elementary. Retrieved 6 January 2017 from http://www.eie.org/engineering-elementary/about-museum-science-boston
15. “Using visitor-flow visualization to improve visitor experience in museums and exhibitions”. Strohmaier et al. Museums and the Web 2015. Retreived 9 January 2017 from http://mw2015.museumsandtheweb.com/paper/enhancing-visitor-experience-and-fostering-museum-popularity-through-deep-insights-in-the-placement-of-exhibits-by-new-techniques-in-visitor-flow-visualization-in-space-and-time/