**Submission**

**The submission you need to generate for each task consists of:**

**A report documenting what you have done with justification and explanation. The report should address all criteria in the grading rubric and in the submission details described in each task. The report will be no longer than 4-5 pages, 11 point font.**

**A video demonstration of the use of your system to answer the required questions. For the queries in Group 2 and Question 3.2, it will suffice to illustrate the results for a small subset of queries. The video demo should be no longer than 5 minutes.**

**Further details are provided in the instructions for each task. For your Task 2 report, you will also be asked to include a general comparison of the stacks used in each task (Hadoop, Storm, Spark).**

*PDF Report*

*You must submit your report in PDF format. Your report should be no longer than 4-5 pages, 11 point font. Your report should include the following:*

*Give a brief overview of how you extracted and cleaned the data.*

*Give a brief overview of how you integrated each system.*

*What approaches and algorithms did you use to answer each question?*

*What are the results of each question? Use only the provided subset for questions from Group 2 and Question 3.2.*

*What system- or application-level optimizations (if any) did you employ?*

*Give your opinion about whether the results make sense and are useful in any way.*

*Video Demonstration Link*

*In your report, you will also need to submit a link to a video demonstration of your approach. Your video should be no more than 5 minutes long. Your video should include the following:*

*Ingesting and analyzing data for each question*

*Displaying/querying the results for each question*

*Record Video Demonstration*

*You can use Personal Capture from Illinois MediaSpace (AKA Kaltura) to record your video demonstration. To learn to use Personal Capture, see the instructions here. Or you can choose to use other recording software to capture your video demonstration.*

*Your video should be uploaded to Illinois MediaSpace by following the steps below:*

*Upload your video to Illinois MediaSpace. See instructions.*

*Publish your video to obtain a shareable link. See Instructions.*

*Include the shareable link for the video demonstration in your report.*

*Submit Task 1*

*Evaluation*

*Your peers will evaluate your submission based on the Task 1 Rubric. This assignment is worth 50 points. The evaluation period will begin immediately after you submit your assignment. You must evaluate at least 3 of your peers' submissions. We suggest you evaluate 5 of your peers.*

*Evaluate Task* System integration: Is the description of the system integration clear enough?

Speed/efficiency: Is the system efficient? Does it produce results quickly? What optimizations are used?

Approaches: Are the techniques used to solve each question reasonable?

Quality of results: Are the results comprehensive, intuitive, and accurate?

Project Report: Report is informative and covers all questions in detail. Report clearly shows all of the following: 1) How data was extracted and cleaned; 2) What approaches were used to solve each question; 3) What results were generated.

Project Video: Video is informative and demonstrates all features/properties in detail. Video clearly shows both of the following: 1) Ingesting and analyzing data in the systems; 2) Querying results.

Speed/Efficiency: System is responsive and quick to generate results. Implementation utilizes 2 or more optimizations to increase speed and efficiency.

System Integration: Integration of systems is sufficient to generate results and is not redundant, and is efficient in all respects.

Quality of Results: Results are correct and include important fields for all questions.

## Extracted and cleaned the data

I reviewed the three question groups and found that all results could be accurately and comprehensively obtained using the Airline On-Time Performance Data (<https://www.transtats.bts.gov/DatabaseInfo.asp?DB_ID=120>).

The data was available from the ESB Snapshot as zip files under the read-only folders, one per year-month. The data extraction was performed as four, 5-year batches for cost optimization. Each batch was run on a single **free-tier** EC2 instance with access to both the ESB Snapshot and a locally Writable ESB Volume. The data extraction process was bash commands which mounted the snapshot then placed the unzipped files into a folder named *raw*. It was found that two files were unusable (HTML not CSV) and I confirmed on Piazza (@19) that neither those files nor the data which would have been contained within them were required for this project.

I did spot checks of the unzipped files to inspect the schema and its consistency. Two of the fields (*OriginCityName*, *DestCityName*) used quoted identifiers (double quotes [ASCII 34]). Those fields were non-atomic (violate first normal form [1NF]) and redundant (*OriginCityName* + ", " + *OriginState*, *DestCityName* + ", " + *DestState*). There also existed two different record schemas. The record schema before 1991-Oct differed from the later record schema by having twenty additional NULLable fields but as such they were still compatible schemas.

File record cleaning was limited to the elimination of quoted identifiers which was only done as an optimization. Cleaning was performed in parallel across the same four EC2 instances. The process was bash commands using sed to first find and replace, comma space, with hyphen, then to find and replace, double quotes, with nothing.

All cleaned files were uploaded to a single S3 bucket under a folder named *scrubbed*.

In Athena, I defined a table named *scrubbed* over the folder using *org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe* which had the quoted identifiers not been removed, it would of had to use a slower deserializer (*OpenCSVSerDe*).

I decided to use a single table on which all queries would be performed against so that answers are more intuitive by the nature of results being transferable (all from a single source). To ensure answers are accurate, I removed data for any flights which were cancelled or diverted.

What is the departure delay of a canceled flight?

What is the arrival delay of a diverted flight?

Will diverted/cancelled flights count as flights to/from an airport?

The *completedflights* table was based on the *scrubbed* table. It was partitioned by year as a performance optimization. It was trimmed to only eleven columns also as a performance optimization. The actual and Computer Reservation System (CRS) arrival/departure columns were converted to timestamps to make the SQL more intuitive.

The *missions* table was based on *completedflights*. It was partitioned by day of the month as a performance optimization. It served as a set of all existing X to Y to Z airports occurring across a two-day window for 2018 data.

## Integrated each system

Project code can be publicly viewed here: <https://github.com/rcook4/cs598project>

The pipeline was comprised of the following systems in order:

**EBS Snapshot => EC2 (bash) => S3 => Athena (SQL) => S3 => EC2 (bash) => DMS (*only g3q2*) => DynamoDB**

Since there was some ambiguity around which technologies were to me used, I confirmed in Piazza (@26) that the pipeline met the project requirements.

Using bash, the Snapshot data files where transformed to CSV and placed in S3 in less than 5 minutes.

I used Athena to define the *scrubbed* table over the CSV files uploaded to S3 from the four EC2 instances. Athena was used to generate new data GZIP CSV files into S3 by using Create Table As Select (CTAS) commands. The base tables (*scrubbed*, *completedflights*, *missions*) combined took less than five minutes. The *g3q2* (answer) table took less than three minutes and all other answer tables took less than 10 seconds. Excluding *g3q2*, bash was used to download each answer table GZIP CSV file from S3 then transform it into JSON and finally batch load into each DynamoDB answer table.

Originally, I understood the requirement for *g3q2* to be only six results so the bash script integration initially included that table too. After learning that the table would be in over fifty million records instead of just six, I decided to change my approach for integration of *g3q2*.

For *g3q2*, bash downloaded the GZIP CSV files from S3 to GUNZIP and uploaded them into S3 with each subfolder being a day of the month. Separate folders made for a more intuitive load process while also be performant with parallelism.

Database Migration Services (DMS) was used for the load process as it would be easy to scale while not incurring significant coding costs. DMS only required the creation of instances, endpoints and tasks. Using DMS, inherently provides functionality of scheduling, monitoring and logging which were valuable for this large load process.

The first load was performed by creating one destination endpoint to DynamoDB and both a source S3 endpoint and a migration task for each day of the month S3 folder. I created one r4.8xlarge instance and five t2.medium instances. Five tasks were started on the r4 and one task started on each t2 instance, all in parallel. Each of those ten tasks loaded about 2 million records in 4 hours. I created five more t2.medium instances and started ten tasks, this time using only t2 instances, one task on each, all in parallel. Again, each of those ten tasks loaded about 2 million records in 4 hours. The final remaining eleven tasks were started on only the r4 instance, all in parallel. Tasks for day of the month 21 through 30 also loaded about 2 million records in 4 hours but day 31 loaded about 1 million records in 2 hours.

My next step was to tune the load process but I received a bill from AWS for $100. It was discovered that the UIUC AWS credits did ***not*** cover use of DMS. I reported this on Piazza (@27) for guidance and was effectively told to explain what tuning optimizations I would have made because no DMS credits would be made available to me.

DynamoDB table g3q2 was already set to on-demand so the next most obvious optimizations would be to partition the table into more (>31) smaller (< 2million) pieces. The 1 million record source S3 folder loaded in half the time of the 2 million record one so I predict that a half million record source S3 folder would load in a quarter of the time (*i.e.* one hour). The maximum number of Athena table partitions is 100 which for an evenly distributed g3q2 answer table would result in a little more than half a million records each partition. Using Athena to partition the table meant the bash script did not have to be coded to further subdivide the S3 folders. I added to above git project the SQL for creating such a hundred-partition table using a CASE statement to generate tiers. If my prediction is correct then loading of the DynamoDB table g3q2 in parallel from 100 S3 source folders would have about a one-hour runtime.

## Results of each question

The first three questions in group two state to compute the top ten answers ordered by “on-time performance”. The example answers for those questions using the mean delay times which is the specific metric listed for the last question. I computed that last question and got results matching or within 10% of the listed result. I suspect the slight variation was due to my choice to through out delayed and cancelled flights as stated in the *Extracted and cleaned the data* section above. I mention this because I wanted to be clear that it was possible for me to match the listed results but I chose not to. Instead of mean delay I decided to use percentage of flight delayed longer than a specific threshold. My reasoning was that I consider delays causing a missed connection more detrimental to “on-time performance” than a single data point within an average. I was unsure what the threshold should be until I noticed the dataset itself has an indicator column of whether or not the flight had more than a 15-minute delay. It seemed BTS created that column as a measure to hold flights accountable to and 15 minutes sounded like a reasonable buffer. My answers for the first three questions in group two use this **failed-to-meet-expectations-percentage metric** for computing “on-time performance”.

Which carrier looks better to you?

origin uniquecarrier avg\_mins neg\_delay short\_delay long\_delay depdel15 pct\_long

BWI PA (1) 4.8 20 64 21 21 20

BWI EA 8.6 48 5145 895 902 14.8

For each airport X, rank the top-10 carriers in decreasing order of on-time departure performance from X.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **RANK** | ***CMI*** | **BWI** | ***MIA*** | **LAX** | ***IAH*** | **SFO** |
| 1 | *US 3.86* | F9 7.21 | *9E 0.00* | PS 8.31 | *PI 8.16* | TZ 11.09 |
| 2 | *TW 7.42* | CO 10.76 | *EV 8.81* | HA 9.21 | *NW 8.91* | PS 12.29 |
| 3 | *OH 9.34* | AA 11.36 | *PA (1) 9.85* | MQ 10.34 | *PA (1) 10.45* | PA (1) 12.72 |
| 4 | *PI 9.49* | NW 12.42 | *XE 10.01* | TZ 11.32 | *AA 10.55* | HA 12.84 |
| 5 | *EV 13.76* | DL 14.37 | *NW 10.93* | OO 11.84 | *WN 11.64* | DL 12.91 |
| 6 | *MQ 16.58* | US 14.61 | *TZ 11.64* | NW 12.01 | *US 12.30* | NW 13.50 |
| 7 | *DH 16.63* | EA 14.80 | *UA 12.69* | ML (1) 13.50 | *TW 12.61* | DH 13.95 |
| 8 |  | 9E 14.84 | *US 13.44* | CO 13.77 | *DL 13.39* | AA 14.42 |
| 9 |  | YV 15.01 | *ML (1) 14.47* | AA 14.20 | *OO 13.61* | CO 14.78 |
| 10 |  | US 15.37 | *PI 15.65* | FL 14.28 | *XE 14.36* | MQ 15.86 |

For each source airport X, rank the top-10 destination airports in decreasing order of on-time departure performance from X.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **RANK** | ***CMI*** | **BWI** | ***MIA*** | **LAX** | ***IAH*** | **SFO** |
| 1 | *PIT 3.65* | MLB 3.40 | *BUF 0.00* | PMD 0.00 | *MLI 0.00* | FAR 0.00 |
| 2 | *DAY 6.11* | IAD 4.23 | *SAN 3.89* | LAX 0.00 | *MSN 0.00* | SDF 0.00 |
| 3 | *STL 7.27* | DAB 5.42 | *HOU 5.69* | BZN 0.00 | *HOU 4.68* | MSO 0.00 |
| 4 | *PIA 7.92* | CHO 7.45 | *ISP 6.43* | LGB 0.00 | *AGS 4.97* | SCK 0.00 |
| 5 | *DFW 10.49* | UCA 7.88 | *SLC 9.13* | SDF 0.00 | *EFD 7.44* | LGA 3.03 |
| 6 | *CVG 11.06* | SRQ 7.89 | *MEM 9.29* | VIS 3.90 | *JAC 8.93* | PIE 4.05 |
| 7 | *ATL 13.76* | SJU 8.36 | *GNV 9.61* | MEM 6.98 | *RNO 9.22* | BNA 5.36 |
| 8 | *ORD 17.10* | OAJ 8.68 | *TLH 9.64* | IYK 7.81 | *MDW 9.30* | OAK 5.68 |
| 9 |  | BGM 9.17 | *EGE 11.29* | HDN 8.57 | *VCT 10.01* | MKE 9.07 |
| 10 |  | GSP 9.48 | *TPA 12.17* | SNA 8.63 | *CLL 10.32* | MEM 9.99 |

*\* there is only one flight from CMI to ABI which was diverted so it does not exist for any calculations*

For each source-destination pair X-Y, rank the top-10 carriers in decreasing order of on-time arrival performance at Y from X.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **RANK** | **CMI => ORD** | **IND => CMH** | **DFW => IAH** | **LAX => SFO** | **JFK => LAX** | **ATL => PHX** |
| 1 | *MQ 22.50* | AA 0.00 | *PA (1) 10.53* | TZ 9.52 | *UA 21.14* | FL 23.37 |
| 2 |  | CO 7.09 | *CO 14.72* | F9 12.74 | *AA 25.90* | US 24.63 |
| 3 |  | HP 14.38 | *OO 17.44* | PS 12.82 | *HP 26.81* | HP 25.88 |
| 4 |  | US 14.66 | *UA 17.75* | EV 20.26 | *DL 29.31* | EA 26.33 |
| 5 |  | NW 17.23 | *XE 18.04* | AA 21.71 | *TW 33.09* | DL 29.08 |
| 6 |  | DL 20.67 | *EV 18.07* | MQ 22.18 | *PA (1) 35.16* |  |
| 7 |  | EA 22.90 | *DL 18.87* | US 22.63 |  |  |
| 8 |  |  | *AA 20.40* | CO 22.96 |  |  |
| 9 |  |  | *MQ 25.86* | WN 23.93 |  |  |
| 10 |  |  |  | UA 12.74 |  |  |

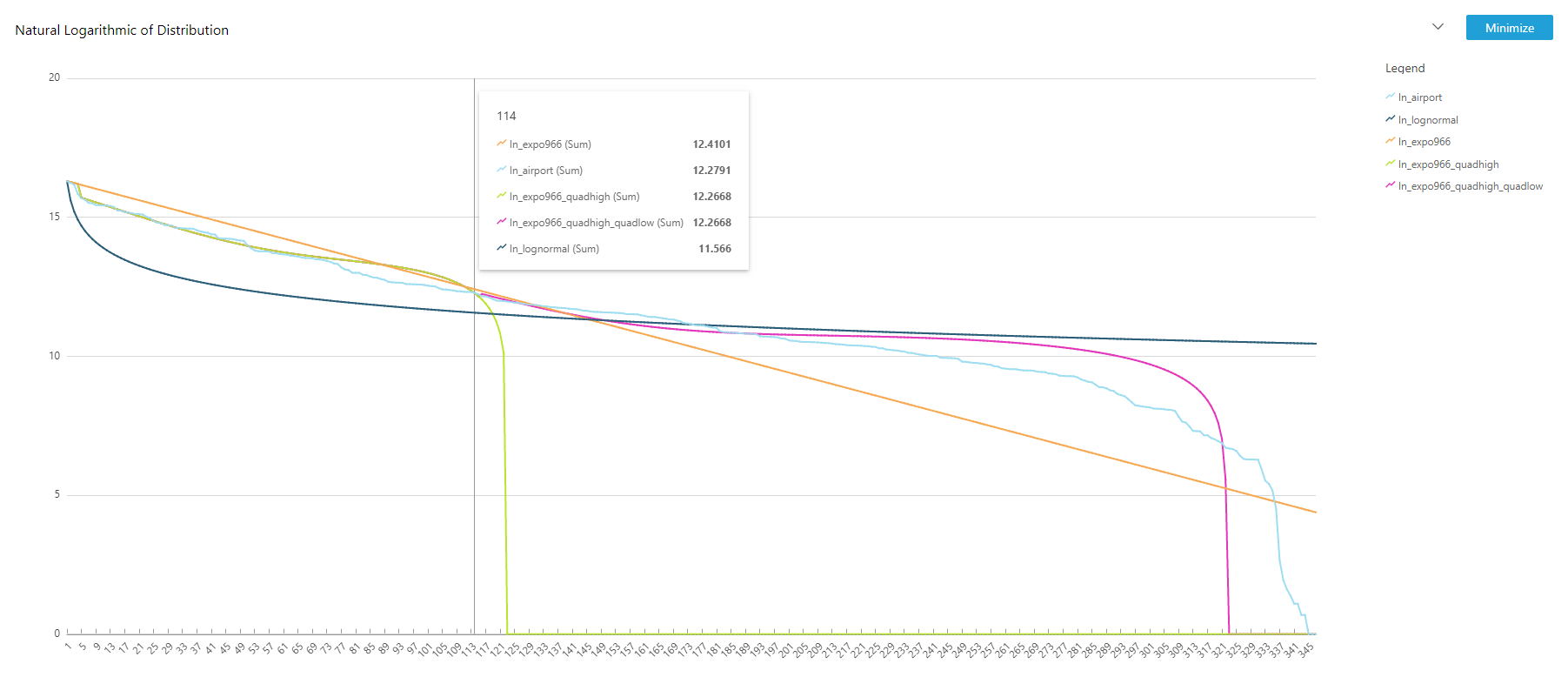
For the group 2 questions the algorithm used in the queries entailed:

1. Use a FROM subquery to AGGREGATE the “on-time performance” using the *completedflights* table
2. Use another FROM subquery to calculate RANK using the AGGREGATE
3. SELECT top ten results using the RANK

In the GitHub project there exists a SQL file named after the Group Question for each query.

Does the popularity distribution of airports follow a Zipf distribution? If not, what distribution does it follow?

The airport popularity looks similar to a Log Normal distribution until zooming into the tail. At that point it you can see clearly it is a Zipf (discrete Pareto) distribution were each subsequent rank is approximately 96.6% of the previous rank.



This QuickSight graph uses a logarithmic scale so big and small values can be seen simultaneously. Actual values are displayed in light blue, log normal in dark blue, Zipf in orange, Zipf with a Quadratic high value offset in green, Zipf with a Quadratic high value offset and a Quadratic low value offset in pink (*Green and Pink were purely for academic curiosity*).

## System-level or application-level optimizations

## Opinion about the results

It was interesting to learn about these online publicly available transportation datasets. The specific dataset used was more than a decade old so any findings may no longer be current. Personally, I tend to be more price conscious than performance conscious so I doubt the results will change my air travel behaviors.

## Video Demonstration Link

<https://mediaspace.illinois.edu/media/t/0_48mnzq8w>