3.1 Coarse – grain analysis

We began by extracting the number of distinct users and jobs per day. To make the timestamps meaningful, we converted them from microseconds to a readable daily format starting from May 1st, 2011. This gave us a better understanding of how workloads were distributed over time, and helped us identify patterns in user and job activity.

Next, we grouped jobs by their priority levels and calculated the **average number of tasks per job** within each group. This helped us analyse how task density varies with job priority, giving us useful insights into how resources might be allocated differently depending on priority.

We also looked at **system reliability** by identifying all failed tasks — specifically those with event\_type = 3. This allowed us to count and understand the frequency of task failures across the system.

3.2)

One of the more complex parts was calculating **machine downtime**. The downtime for each machine is computed as: downtime=join time−leave time

* If a machine never rejoins, it is not included (as we can't determine its total downtime).
* The query aggregates downtime across all machines.
* The final downtime is converted from microseconds to seconds by dividing by 1,000,000.

3.5) creative part

We used the Task\_usage dataset as this was specifically related to user requests, so this was exported as a csv file. We started by loading the raw dataset and dropping a few columns that are not useful for clustering, such as job IDs or process IDs – these are more identifiers than meaningful features

Instead of choosing features manually, we used Principal Component Analysis to automatically rank all numeric features by how much they contribute to the main variance in the data. We calculated the average loading of each feature across the first two principal components and selected the top 5 features with the highest contributions.

Next, we apply outlier removal using the **Interquartile Range (IQR)** method. This filters out extreme values that could distort clustering. We calculate the 25th and 75th percentiles, compute the IQR, and remove any rows that fall outside 1.5 times the IQR. This is the standard.

After outlier removal, we sample 10,000 rows from the dataset. This sample size strikes a good balance:

* It’s large enough to represent the overall structure of the data
* But small enough to allow fast computation, especially when calculating metrics like silhouette score, which can get expensive on large datasets. Therefore, it is efficient without losing representativeness.

Next, we scaled the data using **RobustScaler,** which is resistant to outliers and skewed distributions. I then applied **PCA again**, this time to reduce the data to **two dimensions** for easier clustering and visualization.

To determine the best number of clusters, I use both the Elbow Method and Silhouette Score:

* The Elbow Method helps identify the point where adding more clusters stops giving significant improvement in intra-cluster compactness.
* The Silhouette Score tells us how well-separated the clusters are.

We try values of *k* from 3 to 7, and based on the highest silhouette score, we choose the optimal k for K-Means.

Using the best k, we ran K-Means clustering on the PCA-transformed data. We also generated a decision boundary plot using a mesh grid — this shows how the clusters are separated in the 2D space. Each point is coloured based on its cluster, and the boundaries give a clear visual indication of how the model is grouping the data.

We saved the clustered sample to a CSV file for future analysis or use. We also calculated the cluster-wise averages of the selected features, giving an insight into what each cluster represents.