**/\* Multiple Header and Footer Table \*/**

**Html page:-**

<table mat-table matTableExporter cellspacing="0" cellpadding="0" [dataSource]="dataSource" #exporter="matTableExporter" id="reportTable"

class="emsDataList">

<!-- Default Binding start -->

<ng-container \*ngFor="let columnData of dateSelectionColumns; let selectedDateIndex= index"

[matColumnDef]="columnData">

<th mat-header-cell \*matHeaderCellDef>

<span

\*ngIf="selectedDateIndex == 0">{{authService?.languageObject?.configReport.datePeriod}}</span>

<span \*ngIf="selectedDateIndex == 1">{{datePeriod}}</span>

</th>

</ng-container>

<tr mat-header-row \*matHeaderRowDef="dateSelectionColumns;" [ngClass]="showDatesRow ? '': 'hide'">

</tr>

<!-- Default Binding end -->

<!-- Group Headers Binding start -->

<ng-container \*ngFor="let header of groupHeaders; let groupIndex= index" [matColumnDef]="header.key"

[sticky]="groupIndex == 0 ? true : false">

<th mat-header-cell \*matHeaderCellDef [attr.rowspan]="header.rowSpan"

[attr.colspan]="header.colSpan">

<span>{{header.name}}</span>

</th>

</ng-container>

<tr mat-header-row \*matHeaderRowDef="groupHeaderRows; sticky: isSticky(stickyHeaders, 'sticky-Header')">

</tr>

<!-- Group Headers Binding end -->

<!-- DataPoints Headers Binding start -->

<ng-container \*ngFor="let subHeader of datapointHeaders" [matColumnDef]="subHeader.key">

<th mat-header-cell \*matHeaderCellDef> {{subHeader.name}} </th>

</ng-container>

<tr mat-header-row

\*matHeaderRowDef="datapointHeaderRows; sticky: isSticky(stickyHeaders, 'sticky-Header')">

</tr>

<!-- DataPoints Headers Binding start -->

<!-- Data Columns & Data Binding start -->

<ng-container \*ngFor="let column of displayedColumns; let columnIndex=

index" [matColumnDef]="column.key" [sticky]="columnIndex == 0 ? true : false">

<th mat-header-cell \*matHeaderCellDef><span>{{column.value}}</span></th>

<td mat-cell [ngClass]="element.class" \*matCellDef="let element;">

<span>{{element[column.key]}}</span>

</td>

</ng-container>

<tr mat-header-row \*matHeaderRowDef="dataColumns; sticky: isSticky(stickyHeaders, 'sticky-Header')">

</tr>

<tr mat-row \*matRowDef="let row; columns: dataColumns;"></tr>

<!-- Data Columns & Data Binding end -->

<!-- Footers start -->

<ng-container \*ngFor="let footerEntry of footerData; let footerEntryIndex = index">

<ng-container \*ngFor="let aggregation of footerEntry; let footerIndex = index"

[matColumnDef]="aggregation.key" [sticky]="footerIndex == 0 ? true : false">

<td class="footer" mat-footer-cell \*matFooterCellDef>

<span>{{aggregation.value}}</span>

</td>

</ng-container>

</ng-container>

<!-- <tr mat-footer-row

\*matFooterRowDef="footerDataColumns.sum; sticky: isSticky(stickyFooters, 'sticky-Footer');">

</tr>

<tr mat-footer-row

\*matFooterRowDef="footerDataColumns.average; sticky: isSticky(stickyFooters, 'sticky-Footer');">

</tr>

<tr mat-footer-row

\*matFooterRowDef="footerDataColumns.min; sticky: isSticky(stickyFooters, 'sticky-Footer');">

</tr>

<tr mat-footer-row

\*matFooterRowDef="footerDataColumns.max; sticky: isSticky(stickyFooters, 'sticky-Footer');">

</tr> -->

<ng-container \*ngFor="let footerColumns of footerDataColumns;">

<tr mat-footer-row

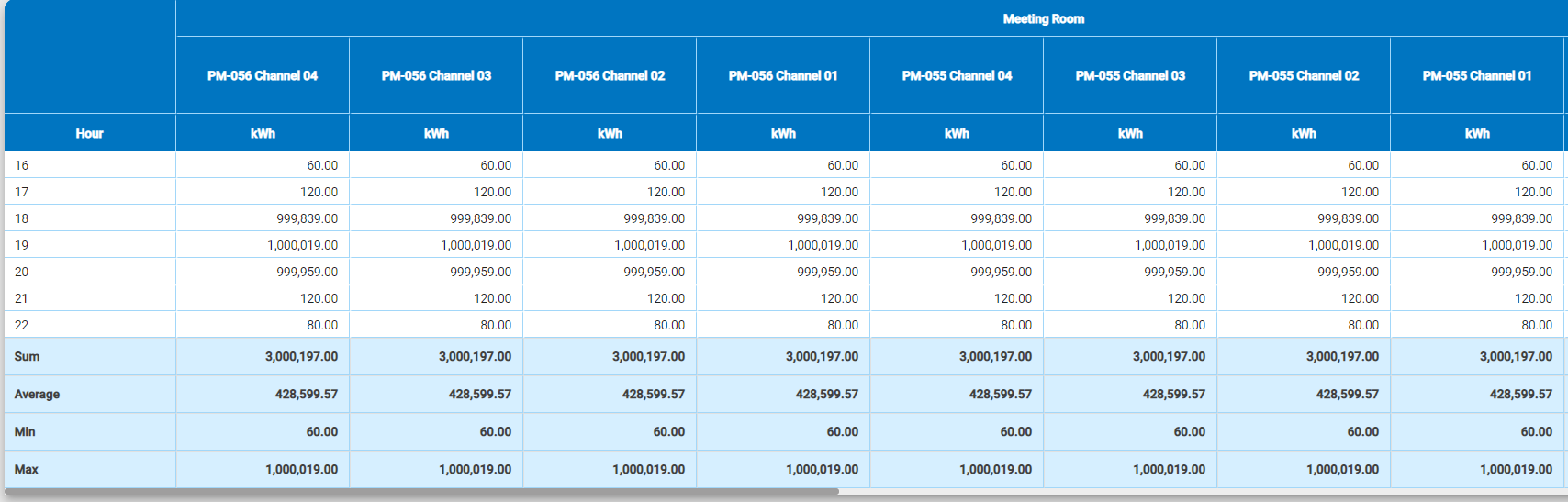
\*matFooterRowDef="footerColumns; sticky: isSticky(stickyFooters, 'sticky-Footer');">

</tr>

</ng-container>

<!-- Footers end -->

</table>

****

**/\* Table .ts file \*/**

displayedColumns = [];

dataColumns = [];

dataSource = [],

footerData = [];

footerDataColumns = [];

getAllConfigReportData() {

try {

const data = this.report;

this.reportName = data.name;

this.reportType = data.type;

this.displayedColumns = [

{

key: "date",

value: this.reportTypeText,

groupName: "",

dataPointName: "",

}

];

let sum = [

{

"key": "sum\_date",

"value": this.authService.languageObject?.configReport.sum

}

];

let average = [

{

"key": "average\_date",

"value": this.authService.languageObject?.configReport.average

}

];

let min = [

{

"key": "min\_date",

"value": this.authService.languageObject?.configReport.min

}

];

let max = [

{

"key": "max\_date",

"value": this.authService.languageObject?.configReport.max

}

];

if (data.children?.length > 0) {

let allColumns = []; // timestamps

data.children.forEach(group => {

group.children.forEach(dataPoint => {

const dataPointData = dataPoint.data;

if (dataPointData && Object.keys(dataPointData).length > 0) {

allColumns = allColumns.concat(Object.keys(dataPointData));

}

});

});

const uniqueColumns = allColumns.filter((column, i, ar) => ar.indexOf(column) === i).sort();

data.children.filter(r => r.children.length > 0).forEach(element => {

this.groupHeaderRows.push(element.\_id);

this.groupHeaders.push({

name: element.name,

key: element.\_id,

rowSpan: 1,

colSpan: element.children.length

});

if (element.children?.length > 0) {

element.children.forEach(datapoint => {

this.datapointHeaderRows.push(`${datapoint.groupId}\_${datapoint.\_id}`);

this.datapointHeaders.push({

name: datapoint.description ? datapoint.description : datapoint.name,

key: `${datapoint.groupId}\_${datapoint.\_id}`

});

const keyId = `${datapoint.groupId}\_${datapoint.\_id}\_${datapoint.engUnit}`;

sum.push({ key: `sum\_${keyId}`, value: (datapoint.sum !== null || datapoint.sum !== undefined) ? this.decimalPipe.transform(datapoint.sum, '1.2-2'): null })

average.push({ key: `average\_${keyId}`, value: (datapoint.avergae !== null || datapoint.avergae !== undefined) ? this.decimalPipe.transform(datapoint.avergae, '1.2-2'): null })

min.push({ key: `min\_${keyId}`, value: (datapoint.min !== null || datapoint.min !== undefined) ? this.decimalPipe.transform(datapoint.min, '1.2-2'): null })

max.push({ key: `max\_${keyId}`, value: (datapoint.max !== null || datapoint.max !== undefined) ? this.decimalPipe.transform(datapoint.max, '1.2-2'): null })

this.displayedColumns.push({

key: keyId,

value: datapoint.engUnit,

groupName: element.name,

dataPointName: datapoint.description ? datapoint.description : datapoint.name

});

for (let i = 0; i < uniqueColumns.length; i++) {

const keyValue = uniqueColumns[i];

const readingValue = datapoint.data && (datapoint.data[keyValue] !== null || datapoint.data[keyValue] !== undefined) ? this.decimalPipe.transform(datapoint.data[keyValue], '1.2-2') : null;

const cssClass: string = this.cellColor(keyValue);

if (this.dataSource[i]) {

this.dataSource[i][keyId] = readingValue;

this.dataSource[i].class = cssClass;

} else {

this.dataSource[i] = {

[this.displayedColumns[0].key]: this.cellFilter(+keyValue),

[keyId]: readingValue,

class: cssClass

};

}

}

})

}

});

}

this.displayedColumns.map((x: any) => {

this.dataColumns.push(x.key);

})

this.footerData = [sum, average, min, max];

this.footerData.forEach((footerRow: any, index: number) => {

if (footerRow.length > 1) {

footerRow.map((x: any) => {

if (this.footerDataColumns[index]) this.footerDataColumns[index].push(x.key);

else this.footerDataColumns.push([x.key]);

});

}

})

}

catch (error) {

}

this.checkReportHasData.emit(this.dataSource.length > 0);

}

cellFilter(value: number) {

switch (this.reportType) {

case ConfigReportType.Hourly: return moment(value).format('HH');

case ConfigReportType.Monthly: return moment(value).locale(this.currentLocale).format("YYYY MMM");

case ConfigReportType.Yearly: return moment(value).locale(this.currentLocale).format("YYYY");

default: return moment(value).locale(this.currentLocale).format("DD ddd");

}

}

cellColor(date: string) {

if (this.reportType == ConfigReportType.Daily) {

const day = this.datePipe.transform(date, "EEEE").toUpperCase();

const weekendDays = ['SATURDAY', 'SUNDAY'];

if (weekendDays.indexOf(day) > -1) return 'redColor';

}

return '';

}