Perfect, Yeon 👍 — let’ s go step-by-step.

We’ ll first break down the **`conman`command**, and then extend the HTML mail script so it can **auto-route reports to multiple teams** (Finance, HR, ETL, etc.) based on job stream prefixes.

—-

🧩 1 🟦Understanding the command

```bash
conman "ss @#;sortby=WSNAME"
```

🔹 `conman`

This is the **TWS (IBM Workload Scheduler)** command-line console tool.
You use it to interact with the **current plan**, check job statuses, rerun jobs, hold/release, etc.

—-

🔹 `"ss"` = *show schedule* (show job stream status)

This sub-command lists all **job streams (schedules)** currently loaded in the **current plan**.

It shows key info like:

```
WSNAME#SCHEDULER     ST   SCHEDTIME
————————————————————-
FIN_DAILY#MASTER     S    2025-10-06-07.00
HR_PAYROLL#MASTER    F    2025-10-06-08.00
```

Where:

***WSNAME** → Job Stream Name (e.g., FIN_DAILY, HR_PAYROLL)
***#SCHEDULER** → The workstation it runs on
***ST** → Status (`S`=Success, `F`=Fail, `R`=Running, etc.)
***SCHEDTIME** → Planned start time

—-

🔹 `@#`

This tells `conman` to show job streams for **all workstations** in the current plan.
If you used just `@`, it would show only the local workstation.

So:

```
ss @#
```

means:
“ Show the status of all job streams on all workstations currently in the plan.”

—-

🔹 `;sortby=WSNAME`

This is a **display modifier** — it sorts the results alphabetically by the job stream name (WSNAME).

Without this, the output is ordered by workstation or submission order.
Sorting helps when you’ re generating reports.

—-

✅ **Final meaning:**

> `conman "ss @#;sortby=WSNAME"`
> ➜ Show the status of all job streams (schedules) on all workstations, sorted by job stream name.

—-

🧩 2 🟦Extend the Script — Send Mails Per Team

Now let’ s build on the previous **HTML EOD script**, so it:

* Reads all job stream names.
* Groups them by **prefix** (e.g., FIN*, HR*, ETL*).
* Sends **separate HTML mails** to each team.

—-

🧰**Enhanced Script:`tws_eod_status_teamwise.sh`**

```bash
#!/bin/bash
##############################################################################################
#  IBM TWS End-of-Day Schedule Summary - Teamwise HTML Mail
#  Author: Yeon + Aria
##############################################################################################

./opt/ibm/TWS/TWS.env

# === Mail mapping per prefix ===
declare -A TEAM_MAILS
TEAM_MAILS["FIN"]="finance_team@company.com"
TEAM_MAILS["HR"]="hr_team@company.com"
TEAM_MAILS["ETL"]="etl_team@company.com"
TEAM_MAILS["OPS"]="ops_team@company.com"

TMP_DIR="/tmp"
RAW_OUT="$TMP_DIR/tws_ss_raw_$$.txt"

# === Fetch schedule list ===
if ! conman "ss @#;sortby=WSNAME" > "$RAW_OUT" 2>/dev/null; then
    echo "Error: Failed to get TWS schedule status"
    exit 1
fi

# === Create a folder for today's reports ===
REPORT_DIR="$TMP_DIR/tws_reports_$(date +%Y%m%d)"
mkdir -p "$REPORT_DIR"

# === Parse and group job streams by prefix ===
while read -r line; do
    [[ -z "$line" || "$line" =~ ^(WSNAME|=|--|#) ]] && continue

    JS_NAME=$(echo "$line" | awk '{print $1}' | cut -d'#' -f1)
    STATUS=$(echo "$line" | awk '{print $2}' | tr '[:lower:]' '[:upper:]')
    START=$(echo "$line" | awk '{print $3}')
    END=$(echo "$line" | awk '{print $4}')

    PREFIX=$(echo "$JS_NAME" | cut -d'_' -f1)
    FILE="$REPORT_DIR/${PREFIX}_report.html"
```

```
    # Initialize file header if not exists
    if [ !-f "$FILE" ]; then
        cat <<EOF > "$FILE"
<html><head>
<style>
body {font-family: Arial;}
th, td {border: 1px solid #ccc; padding: 6px;}
th {background: #0078D7; color: white;}
.success {color: green; font-weight: bold;}
.failed {color: red; font-weight: bold;}
.pending {color: orange; font-weight: bold;}
</style>
</head><body>
<h3>${PREFIX} Team - TWS EOD Summary $(date '+%d-%b-%Y')</h3>
<table width="100%">
<tr><th>Job Stream</th><th>Status</th><th>Start</th><th>End</th></tr>
EOF
    fi

    CLASS="pending"
    case "$STATUS" in
        S|SUCC|SUCCESS|COMPLETED) CLASS="success"; STATUS="SUCCESS";;
        F|FAIL|FAILED|ERROR) CLASS="failed"; STATUS="FAILED";;
        R|RUN|RUNNING|EXEC) CLASS="pending"; STATUS="RUNNING";;
        *) CLASS="pending"; STATUS="PENDING";;
    esac

    echo "<tr><td>$JS_NAME</td><td
class='$CLASS'>$STATUS</td><td>${START:-'-'}</td><td>${END:-'-'}</td></tr>" >> "$FILE"

done < "$RAW_OUT"

# === Finalize and send each team's mail ===
for PREFIX in "${!TEAM_MAILS[@]}"; do
    FILE="$REPORT_DIR/${PREFIX}_report.html"
    if [ -f "$FILE" ]; then
        echo "</table><br><small>Generated automatically on
$(hostname)</small></body></html>" >> "$FILE"
        TO=${TEAM_MAILS[$PREFIX]}
        echo "Sending $PREFIX report to $TO ..."
        cat "$FILE" | mailx -a "Content-Type: text/html" -s "TWS EOD Report - ${PREFIX} Team -
$(date '+%d-%b-%Y')" "$TO"
    fi
```

done

rm -f "$RAW_OUT"
```

---

## 🔹 How It Works

1. Runs `conman "ss @#;sortby=WSNAME"` to fetch all schedules.
2. Determines **team prefix** from job stream name (e.g., `FIN_DAILY_LOAD` → `FIN`).
3. Groups all job streams with same prefix.
4. Generates separate HTML files:

   * `/tmp/tws_reports_20251006/FIN_report.html`
   * `/tmp/tws_reports_20251006/HR_report.html`, etc.
5. Sends each file to that team's email.

---

## 📧 Example

* `FIN_*` job streams → mailed to **Finance Team**
* `HR_*` job streams → mailed to **HR Team**
* `ETL_*` job streams → mailed to **ETL Support**

---

## ✅ Tips

* You can extend the `TEAM_MAILS` array to include more teams easily.
* Add this script to a **daily TWS job stream** at 23:30 to automate reporting.
* For fallback, if a job stream prefix doesn't match any team, you can add a "default" email.

---

Would you like me to add that **default recipient logic** too — so if any new or unmatched prefix appears (say "MARKET_"), it automatically gets sent to a "central monitoring" email?
```