⚡ ⚡ AEMR Data Analysis ⚡ ⚡

It’s time for you to apply your budding SQL Competencies to analyse data for the American Energy Market Regulator (AEMR).

The analytics team has supplied you with the following table extract that contains all the data you need to analyse for the `AEMR` outages.

`AEMR_Outage_Table`

Now let's revisit the business problem below and understand what we're seeking to solve.

What's the Business Problem? 💰

The American Energy Market Regulator (AEMR) is responsible for looking after the United States of America's domestic energy network. The regulator's responsibility is to ensure that America's energy network remains reliable with minimal disruptions, which are known as outages.

There are four key types of outages:

- Consequential

- Forced

- Opportunistic

- Planned

Recently, the AEMR management team has been increasingly aware of a large number of energy providers that submitted outages over the 2016 and 2017 calendar years. The management team has expressed a desire to have the following two areas of concern addressed:

A) Energy Stability and Market Outages

B) Energy Losses and Market Reliability

As an analyst within the data and reporting team, you have been asked to address these two immediate areas of concern. Feel free to also explore beyond the queries asked and provide additional insights that you feel may be of interest to the management team.

In [1]:

```
!pip install ipython-sql
!pip install prettytable==3.12
```

```
Requirement already satisfied: ipython-sql in c:\users\etcok\anaconda3\lib\site-packages (0.5.0)
Requirement already satisfied: prettytable in c:\users\etcok\anaconda3\lib\site-packages (from ipython-sql) (3.12.0)
Requirement already satisfied: ipython in c:\users\etcok\anaconda3\lib\site-packages (from ipython-sql) (8.27.0)
Requirement already satisfied: sqlalchemy>=2.0 in c:\users\etcok\anaconda3\lib\site-packages (from ipython-sql) (2.0.34)
Requirement already satisfied: sqlparse in c:\users\etcok\anaconda3\lib\site-packages (from ipython-sql) (0.5.3)
Requirement already satisfied: six in c:\users\etcok\anaconda3\lib\site-packages (from ipython-sql) (1.16.0)
Requirement already satisfied: ipython-genutils in c:\users\etcok\anaconda3\lib\site-packages (from ipython-sql) (0.2.0)
Requirement already satisfied: typing-extensions>=4.6.0 in c:\users\etcok\anaconda3\lib\site-packages (from sqlalchemy>=2.0->ipython-sql) (4.11.0)
Requirement already satisfied: greenlet!=0.4.17 in c:\users\etcok\anaconda3\lib\site-packages (from sqlalchemy>=2.0->ipython-sql) (3.0.1)
Requirement already satisfied: decorator in c:\users\etcok\anaconda3\lib\site-packages (from ipython->ipython-sql) (5.1.1)
Requirement already satisfied: jedi>=0.16 in c:\users\etcok\anaconda3\lib\site-packages (from ipython->ipython-sql) (0.19.1)
Requirement already satisfied: matplotlib-inline in c:\users\etcok\anaconda3\lib\site-packages (from ipython->ipython-sql) (0.1.6)
Requirement already satisfied: prompt-toolkit<3.1.0,>=3.0.41 in c:\users\etcok\anaconda3\lib\site-packages (from ipython->ipython-sql) (3.0.43)
Requirement already satisfied: pygments>=2.4.0 in c:\users\etcok\anaconda3\lib\site-packages (from ipython->ipython-sql) (2.15.1)
Requirement already satisfied: stack-data in c:\users\etcok\anaconda3\lib\site-packages (from ipython->ipython-sql) (0.2.0)
Requirement already satisfied: traitlets>=5.13.0 in c:\users\etcok\anaconda3\lib\site-packages (from ipython->ipython-sql) (5.14.3)
Requirement already satisfied: colorama in c:\users\etcok\anaconda3\lib\site-packages (from ipython->ipython-sql) (0.4.6)
Requirement already satisfied: wcwidth in c:\users\etcok\anaconda3\lib\site-packages (from prettytable->ipython-sql) (0.2.5)
Requirement already satisfied: parso<0.9.0,>=0.8.3 in c:\users\etcok\anaconda3\lib\site-packages (from jedi>=0.16->ipython->ipython-sql) (0.8.3)
Requirement already satisfied: executing in c:\users\etcok\anaconda3\lib\site-packages (from stack-data->ipython->ipython-sql) (0.8.3)
Requirement already satisfied: asttokens in c:\users\etcok\anaconda3\lib\site-packages (from stack-data->ipython->ipython-sql) (2.0.5)
Requirement already satisfied: pure-eval in c:\users\etcok\anaconda3\lib\site-packages (from stack-data->ipython->ipython-sql) (0.2.2)
Requirement already satisfied: prettytable==3.12 in c:\users\etcok\anaconda3\lib\site-packages (3.12.0)
Requirement already satisfied: wcwidth in c:\users\etcok\anaconda3\lib\site-packages (from prettytable==3.12) (0.2.5)
```

In [2]:

```
import requests
from IPython.core.magic import register_line_magic
from IPython.display import HTML
import sqlite3

@register_line_magic
def load_sqlite_db(url):
    response = requests.get(url)

    if response.status_code == 200:
        with open('temp_db_file.db', 'wb') as file:
            file.write(response.content)
        print('SQLite database file downloaded successfully.')
    else:
        print('Failed to download the SQLite database file.')

sqlite_db_url = 'https://raw.githubusercontent.com/chrishuisb1990/practice_datasets/main/AEMR.db'

%load_sqlite_db $sqlite_db_url

%load_ext sql

%sql sqlite:///temp_db_file.db

%config SqlMagic.style = '_DEPRECATED_DEFAULT'
```

```
SQLite database file downloaded successfully.
```

⚡ Part I. Energy Stability & Market Outages ⚡

Energy stability is one of the key themes the AEMR management team cares about. To ensure energy security and reliability, AEMR needs to understand the following:

- **What are the most common outage types and how long do they tend to last?**
- **How frequently do the outages occur?**
- **Are there any energy providers which have more outages than their peers which may be indicative of being unreliable?**

Please note that throughout the entire case study, we are interested ONLY in the Outages where Status = Approved. We don't have any interest in Outages that were cancelled or not approved. This means your WHERE Clause will ALWAYS contain the field `Where Status = Approved`

Question One

Write a SQL Statement to `COUNT` the number of valid (i.e. `Status = Approved`) Outage Events sorted by their respective `Outage_Reason` (i.e. `Forced`, `Consequential`, `Scheduled`, `Opportunistic`) over the 2016 & 2017 Periods.

Do we notice anything regarding the trends for specific Outages over the 2016 / 2017 Period?

Please write your SQL in the code window below

⚠️ **Note: Remember, you'll need to start each cell with the `%%sql` line, which allows us to execute SQL from within this notebook.**

In [3]:
```
%%sql
SELECT COUNT(*) AS total_number_outages, outage_reason, year
FROM AEMR_Outage_Table
```

```
WHERE status = 'Approved'
GROUP BY outage_reason, year
LIMIT 10;
```

```
 * sqlite:///temp_db_file.db
Done.
```

Out[3]:

total_number_outages	Outage_Reason	Year
181	Consequential	2016
127	Consequential	2017
1264	Forced	2016
1622	Forced	2017
106	Opportunistic Maintenance (Planned)	2016
102	Opportunistic Maintenance (Planned)	2017
380	Scheduled (Planned)	2016
320	Scheduled (Planned)	2017

Question Two

i) Write a SQL Statement showing the `Total` of all Outage Types (Forced, Consequential, Scheduled, Opportunistic) where the `Status = Approved`, that occurred for both 2016 and 2017, grouped by `Year` and `Month`. per month (i.e. 1 – 12). Order by `Year`, `Month`, `Total_Number_Outages` in Descending Order.

ii) Building on the query you write in i), group the results by `Outage Type`, `Year` and `Month`. This is so you can identify whether there is any outage type specifically increasing on a monthly basis when comparing 2016 to 2017.

⚠️ **Hint: You might find it helpful to create a small Common Table Expression to address these two questions!**

Please write your SQL in the code window below

In [7]:

```
%%sql
WITH total_approved_outages AS (
    SELECT
        year, month, COUNT(*) AS total_number_outages
    FROM
        AEMR_Outage_Table
    WHERE
        status = 'Approved'
    GROUP BY
        year, month
    ORDER BY
        year, month, total_number_outages)

SELECT
    tao.year, tao.month, COUNT(outage_reason) AS outage_count, outage_reason
FROM
    total_approved_outages AS tao
JOIN
```

```
    AEMR_Outage_Table AS aot
        ON tao.year = aot.year
        AND tao.month = aot.month
GROUP BY
    tao.year, tao.month, outage_reason
LIMIT 25;
```

```
 * sqlite:///temp_db_file.db
Done.
```

Out[7]:

year	month	outage_count	Outage_Reason
2016	1	24	Consequential
2016	1	143	Forced
2016	1	9	Opportunistic Maintenance (Planned)
2016	1	38	Scheduled (Planned)
2016	2	23	Consequential
2016	2	166	Forced
2016	2	13	Opportunistic Maintenance (Planned)
2016	2	54	Scheduled (Planned)
2016	3	7	Consequential
2016	3	97	Forced
2016	3	16	Opportunistic Maintenance (Planned)
2016	3	27	Scheduled (Planned)
2016	4	89	Forced
2016	4	3	Opportunistic Maintenance (Planned)
2016	4	51	Scheduled (Planned)
2016	5	36	Consequential
2016	5	122	Forced
2016	5	4	Opportunistic Maintenance (Planned)
2016	5	47	Scheduled (Planned)
2016	6	12	Consequential
2016	6	119	Forced
2016	6	15	Opportunistic Maintenance (Planned)
2016	6	42	Scheduled (Planned)
2016	7	23	Consequential
2016	7	83	Forced

Question Three

Write a SQL statement that calculates 1) The `Total_Number_Outage_Events` and 2) The `Average Duration` in <u>DAYS</u> for each `Participant Code` and `Outage Type` over the 2016 and 2017 Period where the `Status = Approved`. Order by `Total_Number_Outage_Events` in Descending Order, `Reason` and `Year`.

Please note the average duration in days should be rounded to 2 decimal places for ease of comparison. When calculating the average duration, please note that you'll need to use the following fields:

`Start_Time` and `End_Time`.

Please write your SQL in the code window below

In [8]:

```
%%sql

SELECT
    participant_code,
    outage_reason,
    year,
    COUNT(*) AS total_num_outage_events,
    ROUND((AVG(ABS((JULIANDAY(end_time) - JULIANDAY(start_time))))),2) AS avg_outage_duration_days
FROM
    AEMR_Outage_Table
WHERE
    status = 'Approved'
GROUP BY
    participant_code, outage_reason, year
ORDER BY
    total_num_outage_events DESC,
    outage_reason,
    year
LIMIT 10;
```

```
 * sqlite:///temp_db_file.db
Done.
```

Out[8]:

Participant_Code	Outage_Reason	Year	total_num_outage_events	avg_outage_duration_days
AURICON	Forced	2017	490	0.07
GW	Forced	2016	317	0.38
GW	Forced	2017	227	1.06
AURICON	Forced	2016	208	0.07
AUXC	Forced	2016	206	0.08
MELK	Forced	2017	177	2.28
TRMOS	Forced	2017	172	0.42
MELK	Forced	2016	157	0.83
PUG	Forced	2017	135	0.25
AUXC	Forced	2017	120	0.02

Now we're getting somewhere...! We've identified participants who are having many outages, as well as participants who have been offline for the longest durations.

Armed with this information, it's important we're able to classify our participants accordingly based on reliability metrics of uptime.

We classify a participant based off the following criteria:

- **High Risk - On average, the participant is unavailable for > 24 Hours (1 Day)**
- **Medium Risk - On average, the participant is unavailable between 12 and 24 Hours**
- **Low Risk - On average, the participant is unavailable for less than 12 Hours**

Question Four

Using the above criteria for context, write a SQL Statement that classifies each participant code as either `High Risk` , `Medium Risk` or `Low Risk` in a column called `Risk_Classification` that is based off their Average Outage Duration Time. Please note that this is for all valid (i.e. `Where status = approved`) outage types (Forced, Consequential, Scheduled, Opportunistic) for all participant codes from 2016 to 2017. Order the results using `Average Duration Time In Days` in descending order.

⚠️ Hint: Think about the CASE Statement and how you might use this to help you with your classification! This is a more challenging question so you'll need to think through this step by step. You might also find `CTEs` or `Sub Queries` helpful for you.

Please write your SQL in the code window below

In [9]:

```
%%sql

WITH outage_info AS (
SELECT
    participant_code,
    outage_reason,
    year,
    COUNT(*) AS total_num_outage_events,
    ROUND((AVG(ABS((JULIANDAY(end_time) - JULIANDAY(start_time))))),2) AS avg_outage_duration_days
FROM
    AEMR_Outage_Table
WHERE
    status = 'Approved'
GROUP BY participant_code, outage_reason, year
)

SELECT *,
CASE
    WHEN avg_outage_duration_days > 1 THEN  'High Risk'
    WHEN avg_outage_duration_days BETWEEN 0.5 AND 1 THEN  'Medium Risk'
    ELSE 'Low Risk'
END AS risk_classification
FROM outage_info
```

```
 * sqlite:///temp_db_file.db
Done.
```

Out[9]:

participant_code	outage_reason	year	total_num_outage_events	avg_outage_duration_days	risk_classification
AURICON	Consequential	2016	41	0.13	Low Risk
AURICON	Consequential	2017	42	0.21	Low Risk
AURICON	Forced	2016	208	0.07	Low Risk
AURICON	Forced	2017	490	0.07	Low Risk
AURICON	Opportunistic Maintenance (Planned)	2016	3	0.33	Low Risk
AURICON	Scheduled (Planned)	2016	46	1.89	High Risk
AURICON	Scheduled (Planned)	2017	45	1.45	High Risk
AUXC	Consequential	2016	1	0.96	Medium Risk
AUXC	Consequential	2017	1	0.1	Low Risk
AUXC	Forced	2016	206	0.08	Low Risk
AUXC	Forced	2017	120	0.02	Low Risk
AUXC	Scheduled (Planned)	2016	2	1.25	High Risk
AUXC	Scheduled (Planned)	2017	1	2.88	High Risk
COLLGAR	Consequential	2016	12	0.6	Medium Risk
COLLGAR	Consequential	2017	5	0.23	Low Risk
COLLGAR	Forced	2016	29	1.11	High Risk
COLLGAR	Forced	2017	45	1.38	High Risk
COLLGAR	Opportunistic Maintenance (Planned)	2017	5	0.33	Low Risk
COLLGAR	Scheduled (Planned)	2016	12	4.28	High Risk
COLLGAR	Scheduled (Planned)	2017	9	6.36	High Risk
DNHR	Consequential	2016	11	0.28	Low Risk
DNHR	Consequential	2017	12	0.24	Low Risk
DNHR	Forced	2016	1	0.4	Low Risk
DNHR	Forced	2017	1	0.48	Low Risk
ENRG	Consequential	2016	15	0.57	Medium Risk
ENRG	Consequential	2017	7	0.27	Low Risk
ENRG	Forced	2016	21	2.24	High Risk
ENRG	Forced	2017	7	0.26	Low Risk
ENRG	Opportunistic Maintenance (Planned)	2016	4	0.47	Low Risk
ENRG	Opportunistic Maintenance (Planned)	2017	3	0.35	Low Risk

participant_code	outage_reason	year	total_num_outage_events	avg_outage_duration_days	risk_classification
ENRG	Scheduled (Planned)	2016	29	4.85	High Risk
ENRG	Scheduled (Planned)	2017	37	4.96	High Risk
EUCT	Consequential	2016	17	0.41	Low Risk
EUCT	Consequential	2017	14	0.32	Low Risk
EUCT	Forced	2016	11	5.9	High Risk
EUCT	Forced	2017	3	0.03	Low Risk
GW	Consequential	2016	20	0.18	Low Risk
GW	Consequential	2017	4	0.24	Low Risk
GW	Forced	2016	317	0.38	Low Risk
GW	Forced	2017	227	1.06	High Risk
GW	Opportunistic Maintenance (Planned)	2016	20	0.31	Low Risk
GW	Opportunistic Maintenance (Planned)	2017	13	0.28	Low Risk
GW	Scheduled (Planned)	2016	45	4.43	High Risk
GW	Scheduled (Planned)	2017	26	2.65	High Risk
KORL	Consequential	2017	6	2.24	High Risk
KORL	Forced	2016	53	0.38	Low Risk
KORL	Forced	2017	76	1.21	High Risk
KORL	Opportunistic Maintenance (Planned)	2016	14	0.22	Low Risk
KORL	Opportunistic Maintenance (Planned)	2017	10	0.24	Low Risk
KORL	Scheduled (Planned)	2016	20	6.96	High Risk
KORL	Scheduled (Planned)	2017	16	3.38	High Risk
MCG	Consequential	2016	2	0.26	Low Risk
MCG	Consequential	2017	2	0.23	Low Risk
MCG	Forced	2016	1	0.79	Medium Risk
MCG	Forced	2017	12	0.28	Low Risk
MCG	Scheduled (Planned)	2016	1	1.42	High Risk
MCG	Scheduled (Planned)	2017	4	0.06	Low Risk
MELK	Consequential	2016	7	0.76	Medium Risk
MELK	Forced	2016	157	0.83	Medium Risk
MELK	Forced	2017	177	2.28	High Risk

participant_code	outage_reason	year	total_num_outage_events	avg_outage_duration_days	risk_classification
MELK	Opportunistic Maintenance (Planned)	2016	24	0.71	Medium Risk
MELK	Opportunistic Maintenance (Planned)	2017	16	0.65	Medium Risk
MELK	Scheduled (Planned)	2016	85	4.61	High Risk
MELK	Scheduled (Planned)	2017	70	6.89	High Risk
MUND	Forced	2016	4	0.37	Low Risk
MUND	Forced	2017	15	0.19	Low Risk
MUND	Opportunistic Maintenance (Planned)	2016	8	0.15	Low Risk
MUND	Opportunistic Maintenance (Planned)	2017	9	0.21	Low Risk
MUND	Scheduled (Planned)	2016	18	3.53	High Risk
MUND	Scheduled (Planned)	2017	7	1.7	High Risk
PJRH	Forced	2016	81	1.22	High Risk
PJRH	Forced	2017	72	0.84	Medium Risk
PJRH	Opportunistic Maintenance (Planned)	2016	23	0.18	Low Risk
PJRH	Opportunistic Maintenance (Planned)	2017	39	0.24	Low Risk
PJRH	Scheduled (Planned)	2016	38	2.37	High Risk
PJRH	Scheduled (Planned)	2017	35	2.61	High Risk
PMC	Consequential	2016	10	0.28	Low Risk
PMC	Consequential	2017	8	3.52	High Risk
PMC	Forced	2016	69	0.49	Low Risk
PMC	Forced	2017	40	0.04	Low Risk
PMC	Opportunistic Maintenance (Planned)	2016	1	0.08	Low Risk
PMC	Opportunistic Maintenance (Planned)	2017	1	0.25	Low Risk
PMC	Scheduled (Planned)	2016	27	1.9	High Risk
PMC	Scheduled (Planned)	2017	12	1.7	High Risk
PUG	Forced	2016	24	0.92	Medium Risk
PUG	Forced	2017	135	0.25	Low Risk
PUG	Opportunistic Maintenance (Planned)	2016	7	0.21	Low Risk
PUG	Opportunistic Maintenance (Planned)	2017	3	0.24	Low Risk
PUG	Scheduled (Planned)	2016	16	1.47	High Risk
PUG	Scheduled (Planned)	2017	11	2.25	High Risk

participant_code	outage_reason	year	total_num_outage_events	avg_outage_duration_days	risk_classification
STHRNCRS	Consequential	2016	9	0.22	Low Risk
STHRNCRS	Consequential	2017	2	0.26	Low Risk
STHRNCRS	Forced	2016	13	0.33	Low Risk
STHRNCRS	Forced	2017	18	0.26	Low Risk
STHRNCRS	Opportunistic Maintenance (Planned)	2017	1	0.06	Low Risk
STHRNCRS	Scheduled (Planned)	2016	13	0.62	Medium Risk
STHRNCRS	Scheduled (Planned)	2017	14	0.68	Medium Risk
TRMOS	Forced	2016	65	0.19	Low Risk
TRMOS	Forced	2017	172	0.42	Low Risk
TRMOS	Opportunistic Maintenance (Planned)	2016	1	0.06	Low Risk
TRMOS	Scheduled (Planned)	2016	5	1.42	High Risk
TRMOS	Scheduled (Planned)	2017	4	2.31	High Risk
TSLA_MGT	Consequential	2016	30	0.44	Low Risk
TSLA_MGT	Consequential	2017	23	0.34	Low Risk
TSLA_MGT	Forced	2016	2	0.25	Low Risk
TSLA_MGT	Forced	2017	4	0.31	Low Risk
TSLA_MGT	Scheduled (Planned)	2016	23	1.97	High Risk
TSLA_MGT	Scheduled (Planned)	2017	28	0.99	Medium Risk
WGUTD	Consequential	2016	6	1.99	High Risk
WGUTD	Consequential	2017	1	0.04	Low Risk
WGUTD	Forced	2016	2	0.02	Low Risk
WGUTD	Forced	2017	8	3.44	High Risk
WGUTD	Opportunistic Maintenance (Planned)	2016	1	0.6	Medium Risk
WGUTD	Opportunistic Maintenance (Planned)	2017	2	0.05	Low Risk
WGUTD	Scheduled (Planned)	2017	1	10.58	High Risk

In [10]:

```
%%sql

WITH outage_info AS (
SELECT
    participant_code,
    outage_reason,
    year,
    COUNT(*) AS total_num_outage_events,
    ROUND((AVG(ABS((JULIANDAY(end_time) - JULIANDAY(start_time))))),2) AS avg_outage_duration_days,
CASE
    WHEN ROUND((AVG(ABS((JULIANDAY(end_time) - JULIANDAY(start_time))))),2) > 1 THEN  'High Risk'
    WHEN ROUND((AVG(ABS((JULIANDAY(end_time) - JULIANDAY(start_time))))),2) >.5
          AND ROUND((AVG(ABS((JULIANDAY(end_time) - JULIANDAY(start_time))))),2) <=1 THEN 'Medium Risk'
    ELSE 'Low Risk'
END AS risk_classification
FROM
    AEMR_Outage_Table
WHERE
    status = 'Approved'
GROUP BY participant_code, outage_reason, year
)


SELECT
    participant_code,
    outage_reason,
    year,
    total_num_outage_events,
    avg_outage_duration_days,
    risk_classification
FROM
    outage_info
ORDER BY
    avg_outage_duration_days DESC
LIMIT 5;
```

```
 * sqlite:///temp_db_file.db
Done.
```

Out[10]:

participant_code	outage_reason	year	total_num_outage_events	avg_outage_duration_days	risk_classification
WGUTD	Scheduled (Planned)	2017	1	10.58	High Risk
KORL	Scheduled (Planned)	2016	20	6.96	High Risk
MELK	Scheduled (Planned)	2017	70	6.89	High Risk
COLLGAR	Scheduled (Planned)	2017	9	6.36	High Risk
EUCT	Forced	2016	11	5.9	High Risk

Now that we've classified our participants as either `High Risk`, `Medium Risk` or `Low Risk`, we want to dig a little deeper.

Does it make sense that `Consequential`, `Opportunistic` or `Planned` aren't considered regarding the Risk Category?

Perhaps we should refine our category accordingly by ensuring we focus our Risk Category on labelling only `Forced` Outages as being a Risk. After all, Forced Outages are the unplanned outages that risk the security of the electricity grid.

Let's add two additional criteria to our classification considering `Total Number of Outage Events` and `Outage Type`.

We've summarised these below:

- **High Risk - On average, the participant is unavailable for > 24 Hours (1 Day) OR the Total Number of Outage Events > 20**
- **Medium Risk - On average, the participant is unavailable between 12 and 24 Hours OR the Total Number of Outage Events is Between 10 and 20**
- **Low Risk - On average, the participant is unavailable for less than 12 Hours OR the Total Number of Outage Events < 10**

- **If Outage Type is not forced, then N/A**

Question Five

Just as you did in Question Four, Using the above criteria for context, write a SQL Statement that classifies each participant code as either `High Risk` , `Medium Risk` or `Low Risk` in a column called `Risk_Classification` using the new classification criteria. Order the results using `Average Duration Time In Days` in descending order.

⚠️ **Hint: Think about the CASE Statement and how you might use this to help you with your classification!**

Please write your SQL in the code window below

In [11]:

```
%%sql

WITH outage_info AS (
SELECT
    participant_code,
    outage_reason,
    year,
    COUNT(*) AS total_num_outage_events,
    ROUND((AVG(ABS((JULIANDAY(end_time) - JULIANDAY(start_time))))),2) AS avg_outage_duration_days,
CASE
    WHEN outage_reason <> 'Forced' THEN 'N/A'
    WHEN ROUND((AVG(ABS((JULIANDAY(end_time) - JULIANDAY(start_time))))),2) > 1 THEN  'High Risk'
    WHEN ROUND((AVG(ABS((JULIANDAY(end_time) - JULIANDAY(start_time))))),2) >.5
          AND ROUND((AVG(ABS((JULIANDAY(end_time) - JULIANDAY(start_time))))),2) <=1 THEN 'Medium Risk'
    WHEN COUNT(*) >20 THEN 'High Risk'
    WHEN COUNT(*) >10 AND COUNT(*) < 20 THEN 'Medium Risk'
          ELSE 'Low Risk'
END AS risk_classification
FROM
    AEMR_Outage_Table
WHERE
    status = 'Approved'
GROUP BY participant_code, outage_reason, year
)


SELECT
    participant_code,
    outage_reason,
    year,
    total_num_outage_events,
    avg_outage_duration_days,
    risk_classification
FROM
    outage_info
    WHERE risk_classification <> 'N/A'
ORDER BY
    avg_outage_duration_days DESC;
```

```
 * sqlite:///temp_db_file.db
Done.
```

Out[11]:

participant_code	outage_reason	year	total_num_outage_events	avg_outage_duration_days	risk_classification
EUCT	Forced	2016	11	5.9	High Risk
WGUTD	Forced	2017	8	3.44	High Risk
MELK	Forced	2017	177	2.28	High Risk
ENRG	Forced	2016	21	2.24	High Risk
COLLGAR	Forced	2017	45	1.38	High Risk
PJRH	Forced	2016	81	1.22	High Risk
KORL	Forced	2017	76	1.21	High Risk
COLLGAR	Forced	2016	29	1.11	High Risk
GW	Forced	2017	227	1.06	High Risk
PUG	Forced	2016	24	0.92	Medium Risk
PJRH	Forced	2017	72	0.84	Medium Risk
MELK	Forced	2016	157	0.83	Medium Risk
MCG	Forced	2016	1	0.79	Medium Risk
PMC	Forced	2016	69	0.49	High Risk
DNHR	Forced	2017	1	0.48	Low Risk
TRMOS	Forced	2017	172	0.42	High Risk
DNHR	Forced	2016	1	0.4	Low Risk
GW	Forced	2016	317	0.38	High Risk
KORL	Forced	2016	53	0.38	High Risk
MUND	Forced	2016	4	0.37	Low Risk
STHRNCRS	Forced	2016	13	0.33	Medium Risk
TSLA_MGT	Forced	2017	4	0.31	Low Risk
MCG	Forced	2017	12	0.28	Medium Risk
ENRG	Forced	2017	7	0.26	Low Risk
STHRNCRS	Forced	2017	18	0.26	Medium Risk
PUG	Forced	2017	135	0.25	High Risk
TSLA_MGT	Forced	2016	2	0.25	Low Risk
MUND	Forced	2017	15	0.19	Medium Risk
TRMOS	Forced	2016	65	0.19	High Risk
AUXC	Forced	2016	206	0.08	High Risk

participant_code	outage_reason	year	total_num_outage_events	avg_outage_duration_days	risk_classification
AURICON	Forced	2016	208	0.07	High Risk
AURICON	Forced	2017	490	0.07	High Risk
PMC	Forced	2017	40	0.04	High Risk
EUCT	Forced	2017	3	0.03	Low Risk
AUXC	Forced	2017	120	0.02	High Risk
WGUTD	Forced	2016	2	0.02	Low Risk

⚡ Part II. Energy Losses & Market Reliability ⚡

When an energy provider provides energy to the market, they are making a commitment to the market and saying; “We will supply X amount of energy to the market under a contractual obligation.” However, in a situation where the outages are Forced, the energy provider intended to provide energy but are unable to provide energy and are forced offline. **If many energy providers are forced offline at the same time it could cause an energy security risk that AEMR needs to mitigate.**

To ensure this doesn’t happen, the AEMR is interested in exploring the following questions:

- Of the outage types in 2016 and 2017, what percent were Forced Outage(s)?
- What was the average duration for a forced outage during both 2016 and 2017? Have we seen an increase in the average duration of forced outages?
- Which energy providers tended to have the largest number of forced outages?

We'll examine this in the questions below.

Question Six

Write a SQL Statement to calculate the proportion of Forced Outages that have occurred over the 2016 - 2017 Period. Do we observe any particular increases regarding any Outage Types over this period?

Please write your SQL in the code window below

In [12]:

```
%%sql

WITH outage_info AS (
SELECT
    YEAR,
    COUNT(*) AS total_num_outages,
    COUNT ( CASE
        WHEN outage_reason LIKE 'Forced' THEN 1 END)  AS total_num_forced_outage_events
FROM
    AEMR_Outage_Table
WHERE
    status = 'Approved'
GROUP BY year
)

SELECT
    total_num_outages,
    total_num_forced_outage_events,
    ROUND(((total_num_forced_outage_events/ (total_num_outages*1.0)) * 100),2)  AS pct_outage_forced
FROM
    outage_info
GROUP BY year;
```

```
 * sqlite:///temp_db_file.db
Done.
```

Out[12]:

total_num_outages	total_num_forced_outage_events	pct_outage_forced
1931	1264	65.46
2171	1622	74.71

Great. It's clear to see now that `Forced Outages` are problematic for us. Not only are they the only outage type that generates financial losses as the Outage is unplanned, it seems there is a number of Energy Participants who have been having a significantly high number of Outages.

Now what can we do about this?

Let's break our analysis down into Macro and Micro Analysis. The total gives us the Overall Duration a participant is offline / has lost energy, however, it doesn't tell us how *frequently* this occurs. In other words, if we have one or two very big outages, it might contribute to very large totals.

However, perhaps an **average** can help us identify how big these Outages might really be, spread across the year!

Let's take a look.

Question Seven

Write a SQL Statement to calculate the `Total Number of Outages`, `Total Duration In Days` and `Total Energy Lost` of all valid `Outages` for each `participant code` and `facility_code`, sorted by `Total Energy Lost` in descending order and Ordered by the YEAR Category.

Please write your SQL in the code window below

In [13]:

```
%%sql

SELECT
    COUNT(*) AS total_num_outages,
    ROUND((SUM(ABS((JULIANDAY(end_time) - JULIANDAY(start_time))))),2)  AS total_duration_in_days,
    ROUND(SUM(energy_lost_mw),2) AS total_energy_lost,
    outage_reason,
    participant_code,
    facility_code,
    year

FROM
    AEMR_Outage_Table
WHERE
    status = 'Approved'
GROUP BY
    year, participant_code, facility_code, outage_reason
ORDER BY
    total_energy_lost DESC;
```

```
 * sqlite:///temp_db_file.db
Done.
```

Out[13]:

total_num_outages	total_duration_in_days	total_energy_lost	Outage_Reason	Participant_Code	Facility_Code	Year
490	33.65	21639.55	Forced	AURICON	AURICON_PNJ_U1	2017
227	240.69	19326.56	Forced	GW	BW1_GREENWATERS_G2	2017
317	120.6	15751.38	Forced	GW	BW1_GREENWATERS_G2	2016
157	129.6	13771.07	Forced	MELK	MELK_G7	2016
208	15.06	10696.28	Forced	AURICON	AURICON_PNJ_U1	2016
177	404.15	10285.4	Forced	MELK	MELK_G7	2017
85	392.25	9668.79	Scheduled (Planned)	MELK	MELK_G7	2016
69	34.06	9093.08	Forced	PMC	PMC_AG	2016
70	482.58	7499.28	Scheduled (Planned)	MELK	MELK_G7	2017
46	87.02	6964.8	Scheduled (Planned)	AURICON	AURICON_PNJ_U1	2016
45	199.4	6450.0	Scheduled (Planned)	GW	BW1_GREENWATERS_G2	2016
45	65.4	5941.25	Scheduled (Planned)	AURICON	AURICON_PNJ_U1	2017
81	98.79	5881.52	Forced	PJRH	PJRH_GT11	2016
40	1.56	5648.44	Forced	PMC	PMC_AG	2017
172	71.9	5016.67	Forced	TRMOS	TIWEST_COG1	2017
72	60.13	4839.28	Forced	PJRH	PJRH_GT11	2017
27	51.4	4839.0	Scheduled (Planned)	PMC	PMC_AG	2016
76	91.79	4679.68	Forced	KORL	KORL_GT3	2017
29	32.33	4320.86	Forced	COLLGAR	COLLGAR_WF1	2016
135	33.4	4112.1	Forced	PUG	PERTHENERGY_KORL_GT1	2017
53	20.0	4040.32	Forced	KORL	KORL_GT3	2016
41	5.31	3925.55	Consequential	AURICON	AURICON_PNJ_U1	2016
26	68.9	3812.42	Scheduled (Planned)	GW	BW1_GREENWATERS_G2	2017
39	9.17	3100.95	Opportunistic Maintenance (Planned)	PJRH	PJRH_GT11	2017
20	6.21	2951.0	Opportunistic Maintenance (Planned)	GW	BW1_GREENWATERS_G2	2016
24	17.0	2877.04	Opportunistic Maintenance (Planned)	MELK	MELK_G7	2016
45	62.1	2787.06	Forced	COLLGAR	COLLGAR_WF1	2017
206	16.04	2734.14	Forced	AUXC	AUXC_WGP	2016
42	8.88	2553.24	Consequential	AURICON	AURICON_PNJ_U1	2017
38	89.9	2445.0	Scheduled (Planned)	PJRH	PJRH_GT11	2016

total_num_outages	total_duration_in_days	total_energy_lost	Outage_Reason	Participant_Code	Facility_Code	Year
23	4.21	2178.95	Opportunistic Maintenance (Planned)	PJRH	PJRH_GT11	2016
12	51.38	2139.87	Scheduled (Planned)	COLLGAR	COLLGAR_WF1	2016
35	91.35	2110.01	Scheduled (Planned)	PJRH	PJRH_GT11	2017
120	2.1	1768.76	Forced	AUXC	AUXC_WGP	2017
20	139.25	1685.6	Scheduled (Planned)	KORL	KORL_GT3	2016
8	28.15	1672.66	Consequential	PMC	PMC_AG	2017
12	20.35	1639.2	Scheduled (Planned)	PMC	PMC_AG	2017
9	57.27	1632.78	Scheduled (Planned)	COLLGAR	COLLGAR_WF1	2017
7	5.35	1524.82	Consequential	MELK	MELK_G7	2016
13	3.67	1423.53	Opportunistic Maintenance (Planned)	GW	BW1_GREENWATERS_G2	2017
12	7.15	1374.0	Consequential	COLLGAR	COLLGAR_WF1	2016
16	10.44	1339.7	Opportunistic Maintenance (Planned)	MELK	MELK_G7	2017
16	23.56	1334.0	Scheduled (Planned)	PUG	PERTHENERGY_KORL_GT1	2016
16	54.1	1326.6	Scheduled (Planned)	KORL	KORL_GT3	2017
65	12.63	1232.43	Forced	TRMOS	TIWEST_COG1	2016
21	47.04	1182.8	Forced	ENRG	ENRG_KALGOORLIE_GT3	2016
14	3.04	1172.05	Opportunistic Maintenance (Planned)	KORL	KORL_GT3	2016
11	24.73	1160.0	Scheduled (Planned)	PUG	PERTHENERGY_KORL_GT1	2017
5	1.17	926.0	Consequential	COLLGAR	COLLGAR_WF1	2017
10	2.42	842.8	Opportunistic Maintenance (Planned)	KORL	KORL_GT3	2017
20	3.6	832.72	Consequential	GW	BW1_GREENWATERS_G2	2016
24	22.17	815.47	Forced	PUG	PERTHENERGY_KORL_GT1	2016
5	1.65	763.96	Opportunistic Maintenance (Planned)	COLLGAR	COLLGAR_WF1	2017
28	27.75	678.4	Scheduled (Planned)	TSLA_MGT	TESLA_PICTON_G1	2017
18	63.46	664.0	Scheduled (Planned)	MUND	MUNDARING_GT1	2016
10	2.81	643.0	Consequential	PMC	PMC_AG	2016
7	1.48	580.0	Opportunistic Maintenance (Planned)	PUG	PERTHENERGY_KORL_GT1	2016
12	3.4	563.33	Forced	MCG	MWF_MUMBIDA_WF1	2017
3	0.98	523.0	Opportunistic Maintenance (Planned)	AURICON	AURICON_PNJ_U1	2016
37	183.58	516.92	Scheduled (Planned)	ENRG	ENRG_KALGOORLIE_GT3	2017

total_num_outages	total_duration_in_days	total_energy_lost	Outage_Reason	Participant_Code	Facility_Code	Year
6	13.42	505.95	Consequential	KORL	KORL_GT3	2017
23	45.25	448.8	Scheduled (Planned)	TSLA_MGT	TESLA_PICTON_G1	2016
29	140.6	445.6	Scheduled (Planned)	ENRG	ENRG_KALGOORLIE_GT3	2016
15	2.85	398.58	Forced	MUND	MUNDARING_GT1	2017
3	0.71	348.0	Opportunistic Maintenance (Planned)	PUG	PERTHENERGY_KORL_GT1	2017
9	1.9	335.86	Opportunistic Maintenance (Planned)	MUND	MUNDARING_GT1	2017
1	0.08	335.0	Opportunistic Maintenance (Planned)	PMC	PMC_AG	2016
1	0.25	335.0	Opportunistic Maintenance (Planned)	PMC	PMC_AG	2017
14	9.46	322.0	Scheduled (Planned)	STHRNCRS	STHRNCRS_EG	2017
13	4.29	299.0	Forced	STHRNCRS	STHRNCRS_EG	2016
13	8.04	299.0	Scheduled (Planned)	STHRNCRS	STHRNCRS_EG	2016
30	13.21	297.0	Consequential	TSLA_MGT	TESLA_PICTON_G1	2016
18	4.65	292.7	Forced	STHRNCRS	STHRNCRS_EG	2017
17	6.96	290.4	Consequential	EUCT	GRASMERE_WF1	2016
7	11.9	262.46	Scheduled (Planned)	MUND	MUNDARING_GT1	2017
8	1.21	248.0	Opportunistic Maintenance (Planned)	MUND	MUNDARING_GT1	2016
23	7.73	227.7	Consequential	TSLA_MGT	TESLA_PICTON_G1	2017
14	4.52	222.3	Consequential	EUCT	GRASMERE_WF1	2017
8	27.54	221.29	Forced	WGUTD	WEST_KALGOORLIE_GT2	2017
4	0.23	220.0	Scheduled (Planned)	MCG	MWF_MUMBIDA_WF1	2017
9	1.94	207.0	Consequential	STHRNCRS	STHRNCRS_EG	2016
6	11.96	198.0	Consequential	WGUTD	WEST_KALGOORLIE_GT2	2016
7	1.85	191.86	Forced	ENRG	ENRG_KALGOORLIE_GT3	2017
4	1.23	169.9	Forced	TSLA_MGT	TESLA_PICTON_G1	2017
5	7.08	168.4	Scheduled (Planned)	TRMOS	TIWEST_COG1	2016
4	9.23	168.4	Scheduled (Planned)	TRMOS	TIWEST_COG1	2017
15	8.48	161.4	Consequential	ENRG	ENRG_KALGOORLIE_GT3	2016
2	0.5	160.0	Forced	TSLA_MGT	TESLA_PICTON_G1	2016
4	1.48	147.2	Forced	MUND	MUNDARING_GT1	2016
2	0.52	110.0	Consequential	MCG	MWF_MUMBIDA_WF1	2016

total_num_outages	total_duration_in_days	total_energy_lost	Outage_Reason	Participant_Code	Facility_Code	Year
2	0.46	94.0	Consequential	MCG	MWF_MUMBIDA_WF1	2017
4	0.96	89.29	Consequential	GW	BW1_GREENWATERS_G2	2017
4	1.9	87.0	Opportunistic Maintenance (Planned)	ENRG	ENRG_KALGOORLIE_GT3	2016
2	0.1	72.0	Opportunistic Maintenance (Planned)	WGUTD	WEST_KALGOORLIE_GT2	2017
11	64.9	64.8	Forced	EUCT	GRASMERE_WF1	2016
3	0.1	64.8	Forced	EUCT	GRASMERE_WF1	2017
7	1.87	63.8	Consequential	ENRG	ENRG_KALGOORLIE_GT3	2017
1	0.79	55.0	Forced	MCG	MWF_MUMBIDA_WF1	2016
1	1.42	55.0	Scheduled (Planned)	MCG	MWF_MUMBIDA_WF1	2016
2	0.04	54.0	Forced	WGUTD	WEST_KALGOORLIE_GT2	2016
2	2.5	50.0	Scheduled (Planned)	AUXC	AUXC_WGP	2016
2	0.52	46.0	Consequential	STHRNCRS	STHRNCRS_EG	2017
1	0.6	36.0	Opportunistic Maintenance (Planned)	WGUTD	WEST_KALGOORLIE_GT2	2016
1	0.04	36.0	Consequential	WGUTD	WEST_KALGOORLIE_GT2	2017
1	10.58	36.0	Scheduled (Planned)	WGUTD	WEST_KALGOORLIE_GT2	2017
1	0.1	26.0	Consequential	AUXC	AUXC_WGP	2017
1	2.88	26.0	Scheduled (Planned)	AUXC	AUXC_WGP	2017
1	0.96	25.0	Consequential	AUXC	AUXC_WGP	2016
3	1.04	24.1	Opportunistic Maintenance (Planned)	ENRG	ENRG_KALGOORLIE_GT3	2017
1	0.06	23.0	Opportunistic Maintenance (Planned)	STHRNCRS	STHRNCRS_EG	2017
12	2.9	17.28	Consequential	DNHR	DNHR_DENMARK_WF1	2017
11	3.12	15.84	Consequential	DNHR	DNHR_DENMARK_WF1	2016
1	0.4	1.44	Forced	DNHR	DNHR_DENMARK_WF1	2016
1	0.48	1.44	Forced	DNHR	DNHR_DENMARK_WF1	2017
1	0.06	0.0	Opportunistic Maintenance (Planned)	TRMOS	TIWEST_COG1	2016

Question Eight

Write a SQL Statement to calculate the `Average Duration In Days` and `Average Energy Lost` of all valid `FORCED OUTAGES` for each `participant code` and `facility_code` sorted by `Average Energy Lost` in descending order and Ordered by the YEAR Category.

Please write your SQL in the code window below

In [14]:

```
%%sql

SELECT
    ROUND((AVG(ABS((JULIANDAY(end_time) - JULIANDAY(start_time))))),2)  AS avg_duration_in_days,
    ROUND(AVG(energy_lost_mw),2) AS avg_energy_lost,
    outage_reason,
    participant_code,
    facility_code,
    year

FROM
    AEMR_Outage_Table
WHERE
    status = 'Approved'
AND
    outage_reason = 'Forced'
GROUP BY
    year, participant_code, facility_code
ORDER BY
    avg_energy_lost DESC;
```

```
 * sqlite:///temp_db_file.db
Done.
```

Out[14]:

avg_duration_in_days	avg_energy_lost	Outage_Reason	Participant_Code	Facility_Code	Year
1.11	149.0	Forced	COLLGAR	COLLGAR_WF1	2016
0.04	141.21	Forced	PMC	PMC_AG	2017
0.49	131.78	Forced	PMC	PMC_AG	2016
0.83	87.71	Forced	MELK	MELK_G7	2016
1.06	85.14	Forced	GW	BW1_GREENWATERS_G2	2017
0.25	80.0	Forced	TSLA_MGT	TESLA_PICTON_G1	2016
0.38	76.23	Forced	KORL	KORL_GT3	2016
1.22	72.61	Forced	PJRH	PJRH_GT11	2016
0.84	67.21	Forced	PJRH	PJRH_GT11	2017
1.38	61.93	Forced	COLLGAR	COLLGAR_WF1	2017
1.21	61.57	Forced	KORL	KORL_GT3	2017
2.28	58.11	Forced	MELK	MELK_G7	2017
2.24	56.32	Forced	ENRG	ENRG_KALGOORLIE_GT3	2016
0.79	55.0	Forced	MCG	MWF_MUMBIDA_WF1	2016
0.07	51.42	Forced	AURICON	AURICON_PNJ_U1	2016
0.38	49.69	Forced	GW	BW1_GREENWATERS_G2	2016
0.28	46.94	Forced	MCG	MWF_MUMBIDA_WF1	2017
0.07	44.16	Forced	AURICON	AURICON_PNJ_U1	2017
0.31	42.48	Forced	TSLA_MGT	TESLA_PICTON_G1	2017
0.37	36.8	Forced	MUND	MUNDARING_GT1	2016
0.92	33.98	Forced	PUG	PERTHENERGY_KORL_GT1	2016
0.25	30.46	Forced	PUG	PERTHENERGY_KORL_GT1	2017
0.42	29.17	Forced	TRMOS	TIWEST_COG1	2017
3.44	27.66	Forced	WGUTD	WEST_KALGOORLIE_GT2	2017
0.26	27.41	Forced	ENRG	ENRG_KALGOORLIE_GT3	2017
0.02	27.0	Forced	WGUTD	WEST_KALGOORLIE_GT2	2016
0.19	26.57	Forced	MUND	MUNDARING_GT1	2017
0.33	23.0	Forced	STHRNCRS	STHRNCRS_EG	2016
0.03	21.6	Forced	EUCT	GRASMERE_WF1	2017
0.19	18.96	Forced	TRMOS	TIWEST_COG1	2016

avg_duration_in_days	avg_energy_lost	Outage_Reason	Participant_Code	Facility_Code	Year
0.26	16.26	Forced	STHRNCRS	STHRNCRS_EG	2017
0.02	14.74	Forced	AUXC	AUXC_WGP	2017
0.08	13.27	Forced	AUXC	AUXC_WGP	2016
5.9	5.89	Forced	EUCT	GRASMERE_WF1	2016
0.4	1.44	Forced	DNHR	DNHR_DENMARK_WF1	2016
0.48	1.44	Forced	DNHR	DNHR_DENMARK_WF1	2017

Question Nine

Write a SQL Statement to calculate the `Average Energy Lost` and `Total Energy Lost` for each `Facility Code` and `Participant Code` across both the 2016 and 2017 periods when the `Outage_Reason` is set to Forced. Upon completion of this, calculate the percentage of energy lost due to forced outages for each `Facility_Code`. Please ORDER the query by `Total Energy Lost` from 2016 to 2017.

From your analysis, which participants have contributed the most to the Energy Lost due to Forced Outages?

Please write your SQL in the code window below

In [15]:

```
%%sql

WITH outage_info AS(
    SELECT
        year,
        facility_code,
        participant_code,
        status,
        outage_reason,
        energy_lost_mw,
        ROUND(AVG(energy_lost_mw),2) AS avg_energy_lost,
        ROUND(SUM(energy_lost_mw), 2) AS total_energy_lost
FROM
        AEMR_Outage_Table

WHERE
        status = 'Approved'
AND
        outage_reason = 'Forced'
GROUP BY
        facility_code, participant_code, year

),

pct_calc AS (
    SELECT
        facility_code,
        participant_code,
        year,
        ROUND(SUM(energy_lost_mw), 2) AS all_energy_lost
FROM
        AEMR_Outage_Table
WHERE
        status = 'Approved'
GROUP BY year)


SELECT
    avg_energy_lost,
    total_energy_lost,
    ROUND((total_energy_lost /all_energy_lost) * 100,2) AS pct_energy_loss,
    oi.outage_reason,
    oi.participant_code,
    oi.facility_code,
    oi.year
FROM
    outage_info AS oi
JOIN
    pct_calc AS pc
ON   oi.year = pc.year

ORDER BY
    total_energy_lost DESC;
```

```
 * sqlite:///temp_db_file.db
Done.
```

Out[15]:

avg_energy_lost	total_energy_lost	pct_energy_loss	outage_reason	participant_code	facility_code	year
44.16	21639.55	17.39	Forced	AURICON	AURICON_PNJ_U1	2017
85.14	19326.56	15.53	Forced	GW	BW1_GREENWATERS_G2	2017
49.69	15751.38	12.25	Forced	GW	BW1_GREENWATERS_G2	2016
87.71	13771.07	10.71	Forced	MELK	MELK_G7	2016
51.42	10696.28	8.32	Forced	AURICON	AURICON_PNJ_U1	2016
58.11	10285.4	8.26	Forced	MELK	MELK_G7	2017
131.78	9093.08	7.07	Forced	PMC	PMC_AG	2016
72.61	5881.52	4.58	Forced	PJRH	PJRH_GT11	2016
141.21	5648.44	4.54	Forced	PMC	PMC_AG	2017
29.17	5016.67	4.03	Forced	TRMOS	TIWEST_COG1	2017
67.21	4839.28	3.89	Forced	PJRH	PJRH_GT11	2017
61.57	4679.68	3.76	Forced	KORL	KORL_GT3	2017
149.0	4320.86	3.36	Forced	COLLGAR	COLLGAR_WF1	2016
30.46	4112.1	3.3	Forced	PUG	PERTHENERGY_KORL_GT1	2017
76.23	4040.32	3.14	Forced	KORL	KORL_GT3	2016
61.93	2787.06	2.24	Forced	COLLGAR	COLLGAR_WF1	2017
13.27	2734.14	2.13	Forced	AUXC	AUXC_WGP	2016
14.74	1768.76	1.42	Forced	AUXC	AUXC_WGP	2017
18.96	1232.43	0.96	Forced	TRMOS	TIWEST_COG1	2016
56.32	1182.8	0.92	Forced	ENRG	ENRG_KALGOORLIE_GT3	2016
33.98	815.47	0.63	Forced	PUG	PERTHENERGY_KORL_GT1	2016
46.94	563.33	0.45	Forced	MCG	MWF_MUMBIDA_WF1	2017
26.57	398.58	0.32	Forced	MUND	MUNDARING_GT1	2017
23.0	299.0	0.23	Forced	STHRNCRS	STHRNCRS_EG	2016
16.26	292.7	0.24	Forced	STHRNCRS	STHRNCRS_EG	2017
27.66	221.29	0.18	Forced	WGUTD	WEST_KALGOORLIE_GT2	2017
27.41	191.86	0.15	Forced	ENRG	ENRG_KALGOORLIE_GT3	2017
42.48	169.9	0.14	Forced	TSLA_MGT	TESLA_PICTON_G1	2017
80.0	160.0	0.12	Forced	TSLA_MGT	TESLA_PICTON_G1	2016
36.8	147.2	0.11	Forced	MUND	MUNDARING_GT1	2016

avg_energy_lost	total_energy_lost	pct_energy_loss	outage_reason	participant_code	facility_code	year
5.89	64.8	0.05	Forced	EUCT	GRASMERE_WF1	2016
21.6	64.8	0.05	Forced	EUCT	GRASMERE_WF1	2017
55.0	55.0	0.04	Forced	MCG	MWF_MUMBIDA_WF1	2016
27.0	54.0	0.04	Forced	WGUTD	WEST_KALGOORLIE_GT2	2016
1.44	1.44	0.0	Forced	DNHR	DNHR_DENMARK_WF1	2016
1.44	1.44	0.0	Forced	DNHR	DNHR_DENMARK_WF1	2017

Question Ten

Having identified the top 3 participants by Total Energy Loss being `GW` , `MELK` and `Auricon` ; Write a SQL Statement calculating the `Total_Energy_Lost` each of these three `Participant_Codes` and the `Facility_Code` . Additionally, identify the `Description_Of_Outage` associated with the highest `Total_Energy_Lost` for each of the `Participant_Codes` and `Facility_Code` for each of the three participants.

Lastly, calculate the percentage of Energy Loss, attributed to these reasons!

⚠️ Hint: As this is the final question, this is a bit of a challenge question which will involve some SQL functions you're not familiar with just yet. In the workplace, you're going to have to grow familiar with googling and searching for functions that you may have not learned or be familiar with. In this question, to identify the TOP `Description_Of_Outage` reason for each Participant, you're going to need to use `PARTITION BY` . You can read all about the approach you can take in this example here . Good luck!

Please write your SQL in the code window below

In [16]:

```
%%sql

-- First CTE
WITH outage_info AS (
SELECT
    status,
    participant_code,
    facility_code,
    description_of_outage,
    ROUND(SUM(energy_lost_mw), 2) AS total_energy_lost,
    RANK() OVER (PARTITION BY participant_code, facility_code ORDER BY SUM(energy_lost_mw) DESC) AS rank

FROM
    AEMR_Outage_Table
WHERE
    participant_code IN ('GW', 'MELK', 'AURICON')
AND
    status = 'Approved'
GROUP BY
    participant_code, facility_code, description_of_outage),

-- Second CTE
pct_calc AS (
SELECT
```

```
    participant_code,
    facility_code,
    ROUND(SUM(energy_lost_mw), 2) AS all_energy_lost
FROM
    AEMR_Outage_Table
WHERE
    participant_code IN ('GW', 'MELK', 'AURICON')
GROUP BY
    participant_code,
    facility_code)


-- Final Table
SELECT
    oi.participant_code,
    oi.facility_code,
    oi.description_of_outage,
    oi.total_energy_lost,
    ROUND( (oi.total_energy_lost/pc.all_energy_lost) *100, 2) AS pct_e_loss,
    rank

FROM outage_info AS oi
JOIN pct_calc AS pc
ON oi.participant_code = pc.participant_code
WHERE rank = 1
GROUP BY
    oi.participant_code;
```

```
 * sqlite:///temp_db_file.db
Done.
```

Out[16]:

participant_code	facility_code	description_of_outage	total_energy_lost	pct_e_loss	rank
AURICON	AURICON_PNJ_U1	Full unit trip	6033.87	8.24	1
GW	BW1_GREENWATERS_G2	Operational Issues caused real time forced outage.	28687.54	50.16	1
MELK	MELK_G7	Safety Issues	1100.0	1.9	1