

Solving analytical queries on Redshift Cluster

Here, you have to write the query used for solving the question and the screenshots of the table which is outputted after the query is run on the AWS Redshift Query editor UI.

1. Top 10 ATMs where most transactions are in the 'inactive' state

-- Top 10 ATMs where most transactions are in the 'inactive' state

```
SELECT b.atm_number, b.atm_manufacturer, c.location, count(*) transaction_count
FROM atm_case_study.fact_atm_trans a, atm_case_study.dim_atm b,
atm_case_study.dim_location c
WHERE a.atm_id = b.atm_id
AND a.weather_loc_id = c.location_id
AND atm_status='Inactive'
GROUP BY atm_number, atm_manufacturer, location
ORDER BY transaction_count desc
LIMIT 10;
```

The screenshot displays the AWS Redshift Query Editor v2 interface. The SQL query is as follows:

```
1 -- Top 10 ATMs where most transactions are in the 'inactive' state
2 SELECT b.atm_number, b.atm_manufacturer, c.location, count(*) transaction_count
3 FROM atm_case_study.fact_atm_trans a, atm_case_study.dim_atm b, atm_case_study.dim_location c
4 WHERE a.atm_id = b.atm_id
5 AND a.weather_loc_id = c.location_id
6 AND atm_status='Inactive'
7 GROUP BY atm_number, atm_manufacturer, location
8 ORDER BY transaction_count desc
9 LIMIT 10;
```

The results are shown in a table with the following data:

atm_number	atm_manufacturer	location	transaction_count
16	NCR	Skive	44043
12	NCR	ÅrEøsterÅrÅw Duus	33982
2	NCR	Vejgaard	33725
88	NCR	Storcenter indg A	32183
30	NCR	NyÅrÅ ping Mors	30883
52	NCR	FarsÅrÅ	27361
50	NCR	Aarhus	23416
29	NCR	Skejagervej 15	20773
81	NCR	Spar KÅrÅrÅrmand Tomh...	20148
102	NCR	Aalborg Storcenter Att	18297

At the bottom of the interface, the status bar indicates: Query ID 1691 Elapsed time: 635 ms Total rows: 10.

2. Number of ATM failures corresponding to the different weather conditions recorded at the time of the transactions

-- Number of ATM failures corresponding to the different weather conditions recorded at the time of the transactions

```
SELECT a.weather_main,  
count(trans_id) total_transaction_count,  
count(case when a.atm_status = 'Inactive' then 1 else null end) "inactive_count",  
trunc(cast(inactive_count as decimal(10,4))/total_transaction_count*100,2) "inactive_count_pct"  
FROM atm_case_study.fact_atm_trans a  
WHERE a.weather_main is not null  
GROUP BY a.weather_main  
ORDER BY inactive_count_pct desc;
```

The screenshot shows the AWS Redshift Query Editor v2 interface. The query editor displays a SQL query that calculates the number of ATM failures (inactive count) as a percentage of the total transaction count, grouped by weather conditions. The query is as follows:

```
9 LIMIT 10;  
10  
11 -- Number of ATM failures corresponding to the different weather conditions recorded at the time of the transactions  
12 SELECT a.weather_main,  
13 count(trans_id) total_transaction_count,  
14 count(case when a.atm_status = 'Inactive' then 1 else null end) "inactive_count",  
15 trunc(cast(inactive_count as decimal(10,4))/total_transaction_count*100,2) "inactive_count_pct"  
16 FROM atm_case_study.fact_atm_trans a  
17 WHERE a.weather_main is not null  
18 GROUP BY a.weather_main  
19 ORDER BY inactive_count_pct desc;  
20  
21 -- Top 10 ATM's with the most number of transactions throughout the year  
22 SELECT b.atm_number, b.atm_manufacturer, c.location, count(*) transaction_count  
23 FROM atm_case_study.fact_atm_trans a, atm_case_study.dim_atm b, atm_case_study.dim_location c  
24 WHERE a.atm_id = b.atm_id  
25 AND a.weather_loc_id = c.location_id  
26 GROUP BY atm_number, atm_manufacturer, location  
27 ORDER BY transaction_count desc;
```

The results of the first query are displayed in a table with 10 rows and 4 columns: weather_main, total_transaction_count, inactive_count, and inactive_count_pct. The results are as follows:

weather_main	total_transaction_count	inactive_count	inactive_count_pct
Snow	23405	4813	20.56
Fog	18174	3729	20.51
Clouds	1181901	194027	16.41
Rain	545135	86017	15.77
Clear	543949	85531	15.72
Mist	82801	12854	15.53
Thunderstorm	2549	361	14.16
Drizzle	62530	8670	13.86
TORNADO	38	1	2.63
Haze	3	0	0

The interface also shows a sidebar with navigation options like Editor, Queries, Notebooks, Charts, History, and Scheduled queries. The bottom status bar indicates Query ID 1701, Elapsed time: 384 ms, and Total rows: 10.

3. Top 10 ATMs with the most number of transactions throughout the year

-- Top 10 ATMs with the most number of transactions throughout the year

```
SELECT b.atm_number, b.atm_manufacturer, c.location, count(*) transaction_count
FROM atm_case_study.fact_atm_trans a, atm_case_study.dim_atm b,
atm_case_study.dim_location c
WHERE a.atm_id = b.atm_id
AND a.weather_loc_id = c.location_id
GROUP BY atm_number, atm_manufacturer, location
ORDER BY transaction_count desc
LIMIT 10;
```

The screenshot displays the AWS Redshift Query Editor v2 interface. The top navigation bar includes the AWS logo, a search bar, and a user profile. The left sidebar contains navigation options: Editor, Queries, Notebooks, Charts, History, and Scheduled queries. The main workspace is titled 'Redshift query editor v2' and shows a SQL query in a dark-themed editor. The query is designed to find the top 10 ATMs with the most transactions throughout the year. Below the query editor, the results are displayed in a table format, showing the top 10 ATMs with their respective transaction counts. The table has four columns: atm_number, atm_manufacturer, location, and transaction_count. The results are sorted in descending order of transaction count.

```
19 GROUP BY atm_number, atm_manufacturer, location
20
21 -- Top 10 ATMs with the most number of transactions throughout the year
22 SELECT b.atm_number, b.atm_manufacturer, c.location, count(*) transaction_count
23 FROM atm_case_study.fact_atm_trans a, atm_case_study.dim_atm b, atm_case_study.dim_location c
24 WHERE a.atm_id = b.atm_id
25 AND a.weather_loc_id = c.location_id
26 GROUP BY atm_number, atm_manufacturer, location
27 ORDER BY transaction_count desc
28 LIMIT 10;
29
30 -- Number of overall ATM transactions going inactive per month for each month
31 SELECT b.year, b.month,
32 count(a.trans_id) "total transaction count",
33 count(case when a.atm_status = 'inactive' then 1 else null end) "inactive count",
34 trunc((cast(inactive_count as decimal(18,4))/total_transaction_count*100,2) "inactive_count_pct"
35 FROM atm_case_study.fact_atm_trans a, atm_case_study.dim_date b
36 WHERE a.date_id = b.date_id
37 GROUP BY year, month
```

atm_number	atm_manufacturer	location	transaction_count
39	NCR	Svenstrup	55380
20	NCR	Bispensgade	54211
10	NCR	NÅfÅ/resundby	53794
24	NCR	Hobro	53376
45	NCR	Abildgaard	53196
16	NCR	Sliske	44043
40	Diebold Nixdorf	Frederikshavn	43767
1	NCR	NÅfÅ/sved	42787
41	Diebold Nixdorf	Skagen	42732
48	Diebold Nixdorf	BiÅfÅ/nderslev	42493

Query ID 2398 Elapsed time: 6680 ms Total rows: 10

4. Number of overall ATM transactions going inactive per month for each month

-- Number of overall ATM transactions going inactive per month for each month

```
SELECT b.year, b.month,  
count(a.trans_id) "total_transaction_count",  
count(case when a.atm_status = 'Inactive' then 1 else null end) "inactive_count",  
trunc(cast(inactive_count as decimal(10,4))/total_transaction_count*100,2) "inactive_count_pct"  
FROM atm_case_study.fact_atm_trans a, atm_case_study.dim_date b  
WHERE a.date_id = b.date_id  
GROUP BY year, month  
ORDER BY year, month;
```

The screenshot shows the AWS Redshift Query Editor v2 interface. The query editor displays a SQL query that calculates the number of overall ATM transactions going inactive per month for each month. The query is as follows:

```
21 -- Top 10 ATM's with the most number of transactions throughout the year  
22 SELECT b.atm_number, b.atm_manufacturer, c.atm_location "location", count(*) transaction_count  
23 FROM atm_case_study.fact_atm_trans a, atm_case_study.dim_atm b, atm_case_study.dim_location c  
24 WHERE a.atm_id = b.atm_id  
25 AND a.weather_loc_id = c.location_id  
26 GROUP BY atm_number, atm_manufacturer, atm_location  
27 ORDER BY transaction_count desc  
28 LIMIT 10;  
29  
30 -- Number of overall ATM transactions going inactive per month for each month  
31 SELECT b.year, b.month,  
32 count(a.trans_id) "total_transaction_count",  
33 count(case when a.atm_status = 'Inactive' then 1 else null end) "inactive_count",  
34 trunc(cast(inactive_count as decimal(10,4))/total_transaction_count*100,2) "inactive_count_pct"  
35 FROM atm_case_study.fact_atm_trans a, atm_case_study.dim_date b  
36 WHERE a.date_id = b.date_id  
37 GROUP BY year, month  
38 ORDER BY year, month;  
39
```

The results of the query are displayed in a table with the following columns: year, month, total_transaction_count, inactive_count, and inactive_count_pct. The results show data for the year 2017, grouped by month.

year	month	total_transaction_count	inactive_count	inactive_count_pct
2017	April	218905	41830	19.11
2017	August	217218	36713	16.9
2017	December	197048	20476	10.39
2017	February	186559	36556	20.06
2017	January	180195	38953	19.95
2017	July	227682	38139	16.75
2017	June	225166	36789	16.33
2017	March	205586	41046	19.98
2017	May	222418	37679	16.94
2017	November	193967	21584	11.17
2017	October	191557	21780	11.36
2017	September	202101	28913	14.3

5. Top 10 ATMs with the highest total withdrawn amount throughout the year

-- Top 10 ATMs with the highest total amount withdrawn throughout the year

```
SELECT b.atm_number, b.atm_manufacturer, c.location, sum(a.transaction_amount)
```

```
"total_transaction_amount"
```

```
FROM atm_case_study.fact_atm_trans a, atm_case_study.dim_atm b,
```

```
atm_case_study.dim_location c
```

```
WHERE a.atm_id = b.atm_id
```

```
AND a.weather_loc_id = c.location_id
```

```
GROUP BY atm_number, atm_manufacturer, location
```

```
ORDER BY total_transaction_amount DESC
```

```
LIMIT 10;
```

The screenshot displays the AWS Redshift Query Editor v2 interface. The left sidebar shows the 'Queries' section with a tree view of resources including 'redshift-cluster-1', 'awsdatacatalog', 'dev', 'public', and 'sample_data_dev'. The main editor area contains a SQL query that identifies the top 10 ATMs by total transaction amount. The query is as follows:

```
ORDER BY year, month;
SELECT b.atm_number, b.atm_manufacturer, c.location, sum(a.transaction_amount) "total_transaction_amount"
FROM atm_case_study.fact_atm_trans a, atm_case_study.dim_atm b, atm_case_study.dim_location c
WHERE a.atm_id = b.atm_id
AND a.weather_loc_id = c.location_id
GROUP BY atm_number, atm_manufacturer, location
ORDER BY total_transaction_amount DESC
LIMIT 10;
```

Below the query, the 'Result 1 (10)' section shows a table with 4 columns: atm_number, atm_manufacturer, location, and total_transaction_amount. The results are as follows:

atm_number	atm_manufacturer	location	total_transaction_amount
39	NCR	Svenstrup	277097637
20	NCR	Bispensgade	271008803
24	NCR	Hobro	268289882
10	NCR	NÅrÅresundby	267379103
45	NCR	Abildgaard	265639616
16	NCR	Skive	220677013
40	Diebold Nixdorf	Frederikshavn	219812287
41	Diebold Nixdorf	Skagen	214127315
1	NCR	NÅrÅrsted	213721117
48	Diebold Nixdorf	BrÅrÅrskov	212883099

The bottom status bar indicates 'Query ID 2423', 'Elapsed time: 7326 ms', and 'Total rows: 10'.

6. Number of failed ATM transactions across various card types

-- Number of failed ATM transactions across various card types

```
SELECT b.card_type,
count(a.trans_id) "total_transaction_count",
count(case when a.atm_status = 'Inactive' then 1 else null end) "inactive_count",
trunc(cast(inactive_count as decimal(10,4))/total_transaction_count*100,2) "inactive_count_pct"
FROM atm_case_study.fact_atm_trans a, atm_case_study.dim_card_type b
WHERE a.card_type_id = b.card_type_id
GROUP BY card_type
ORDER BY inactive_count_pct DESC;
```

The screenshot displays the AWS Redshift Query Editor v2 interface. The top navigation bar includes the AWS logo, a services menu, a search bar, and user information (N. Virginia, vottabs/user505060-ductran2306@gmail.com). The left sidebar features navigation icons for Editor, Databases, Tables, Views, Functions, and Stored procedures. The main workspace is divided into three panes:

- SQL Editor:** Contains a query to find failed ATM transactions across various card types and calculate the percentage of inactive counts. The query is as follows:

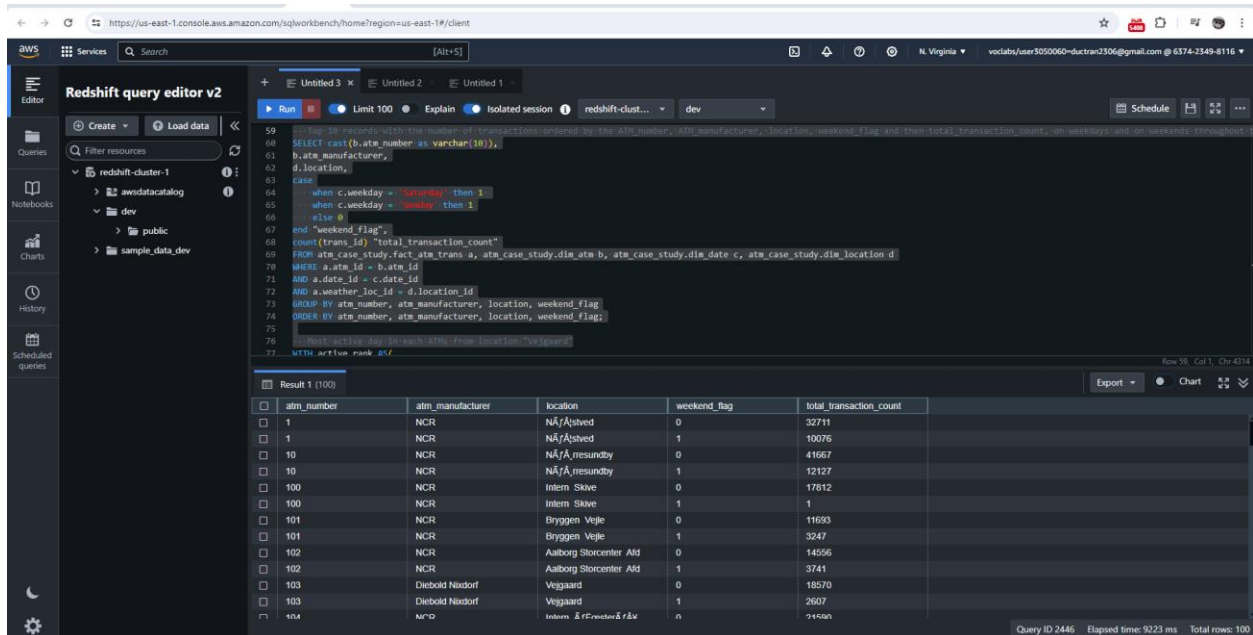

```

39 -- Top 10 atm atm with the highest total amount withdrawn throughout the year
40 SELECT b.ata_number, b.ata_manufacturer, c.ata_location, sum(a.transaction_amount) "total_transaction_amount"
41 FROM atm_case_study.fact_atm_trans a, atm_case_study.dim_atm b, atm_case_study.dim_location c
42 WHERE a.ata_id = b.ata_id
43 AND a.weather_loc_id = c.location_id
44 GROUP BY ata_number, ata_manufacturer, ata_location
45 ORDER BY total_transaction_amount DESC
46 LIMIT 10;

47 -- Number of failed ATM transactions across various card types
48 SELECT b.card_type
49 count(a.trans_id) "total_transaction_count",
50 count(case when a.ata_status = 'Inactive' then 1 else null end) "inactive_count",
51 trunc(cast(inactive_count as decimal(10,4))/total_transaction_count*100,2) "inactive_count_pct"
52 FROM atm_case_study.fact_atm_trans a, atm_case_study.dim_card_type b
53 WHERE a.card_type_id = b.card_type_id
54 GROUP BY card_type
55 ORDER BY inactive_count_pct DESC;
      
```
- Result 1 (12):** Displays a table with the following columns: card_type, total_transaction_count, inactive_count, and inactive_count_pct. The table contains 12 rows of data for various card types, including Mastercard, Visa, Dankort, and Maestro.
- Right-hand pane:** Includes an 'Export' button and a 'Chart' icon.

7. Number of transactions happening on an ATM on weekdays and on weekends throughout the year. Order this by the ATM_number, ATM_manufacturer, location, weekend_flag and then total_transaction_count

```
SELECT cast(b.atm_number as varchar(10)),
b.atm_manufacturer,
d.location,
case
    when c.weekday = 'Saturday' then 1
    when c.weekday = 'Sunday' then 1
    else 0
end "weekend_flag",
count(trans_id) "total_transaction_count"
FROM atm_case_study.fact_atm_trans a, atm_case_study.dim_atm b,
atm_case_study.dim_date c, atm_case_study.dim_location d
WHERE a.atm_id = b.atm_id
AND a.date_id = c.date_id
AND a.weather_loc_id = d.location_id
GROUP BY atm_number, atm_manufacturer, location, weekend_flag
ORDER BY atm_number, atm_manufacturer, location, weekend_flag;
```



The screenshot shows the AWS Redshift Query Editor v2 interface. The SQL query is displayed in the editor, and the results are shown in a table below. The table has 5 columns: atm_number, atm_manufacturer, location, weekend_flag, and total_transaction_count. The results are ordered by atm_number, atm_manufacturer, location, weekend_flag, and total_transaction_count.

atm_number	atm_manufacturer	location	weekend_flag	total_transaction_count
1	NCR	NARÅsted	0	32711
1	NCR	NARÅsted	1	10076
10	NCR	NARÅresundby	0	41667
10	NCR	NARÅresundby	1	12127
100	NCR	Intern Skive	0	17812
100	NCR	Intern Skive	1	1
101	NCR	Bryggen Vejle	0	11693
101	NCR	Bryggen Vejle	1	3247
102	NCR	Aalborg Storcenter Ald	0	14656
102	NCR	Aalborg Storcenter Ald	1	3741
103	Diebold Nixdorf	Vejgaard	0	18570
103	Diebold Nixdorf	Vejgaard	1	2607
104	NCR	Intern & C-rester & R&M	0	74500

8. Most active day in each ATMs from location "Vejgaard"

-- Most active day in each ATMs from location "Vejgaard"

```
WITH active_rank AS(
SELECT b.atm_number,
b.atm_manufacturer,
d.location,
c.weekday,
count(trans_id) "total_transaction_count",
RANK() OVER(PARTITION BY atm_number, atm_manufacturer, location ORDER BY
total_transaction_count DESC) "Rank"
FROM atm_case_study.fact_atm_trans a, atm_case_study.dim_atm b,
atm_case_study.dim_date c, atm_case_study.dim_location d
WHERE a.atm_id = b.atm_id
AND a.date_id = c.date_id
AND a.weather_loc_id = d.location_id
AND d.location = 'Vejgaard'
GROUP BY atm_number, atm_manufacturer, location, weekday
ORDER BY atm_number, atm_manufacturer, location, weekday
)
SELECT atm_number, atm_manufacturer, location, weekday, total_transaction_count
FROM active_rank
WHERE rank = 1;
```

The screenshot displays the AWS Redshift Query Editor v2 interface. The SQL query is as follows:

```
77 WITH active_rank AS(
78 SELECT b.atm_number,
79 b.atm_manufacturer,
80 d.location,
81 c.weekday,
82 count(trans_id) "total_transaction_count",
83 RANK() OVER(PARTITION BY atm_number, atm_manufacturer, location ORDER BY total_transaction_count DESC) "Rank"
84 FROM atm_case_study.fact_atm_trans a, atm_case_study.dim_atm b, atm_case_study.dim_date c, atm_case_study.dim_location d
85 WHERE a.atm_id = b.atm_id
86 AND a.date_id = c.date_id
87 AND a.weather_loc_id = d.location_id
88 AND d.location = 'Vejgaard'
89 GROUP BY atm_number, atm_manufacturer, location, weekday
90 ORDER BY atm_number, atm_manufacturer, location, weekday
91 )
92 SELECT atm_number, atm_manufacturer, location, weekday, total_transaction_count
93 FROM active_rank
94 WHERE rank = 1;
```

The query results are displayed in a table with the following columns: atm_number, atm_manufacturer, location, weekday, and total_transaction_count. The results show two rows:

atm_number	atm_manufacturer	location	weekday	total_transaction_count
103	Diebold Nadorf	Vejgaard	Friday	4757
2	NCR	Vejgaard	Friday	6290

The interface also shows a sidebar with navigation options like Queries, Notebooks, Charts, History, and Scheduled queries. The bottom status bar indicates "Query ID 2471", "Elapsed time: 9214 ms", and "Total rows: 2".