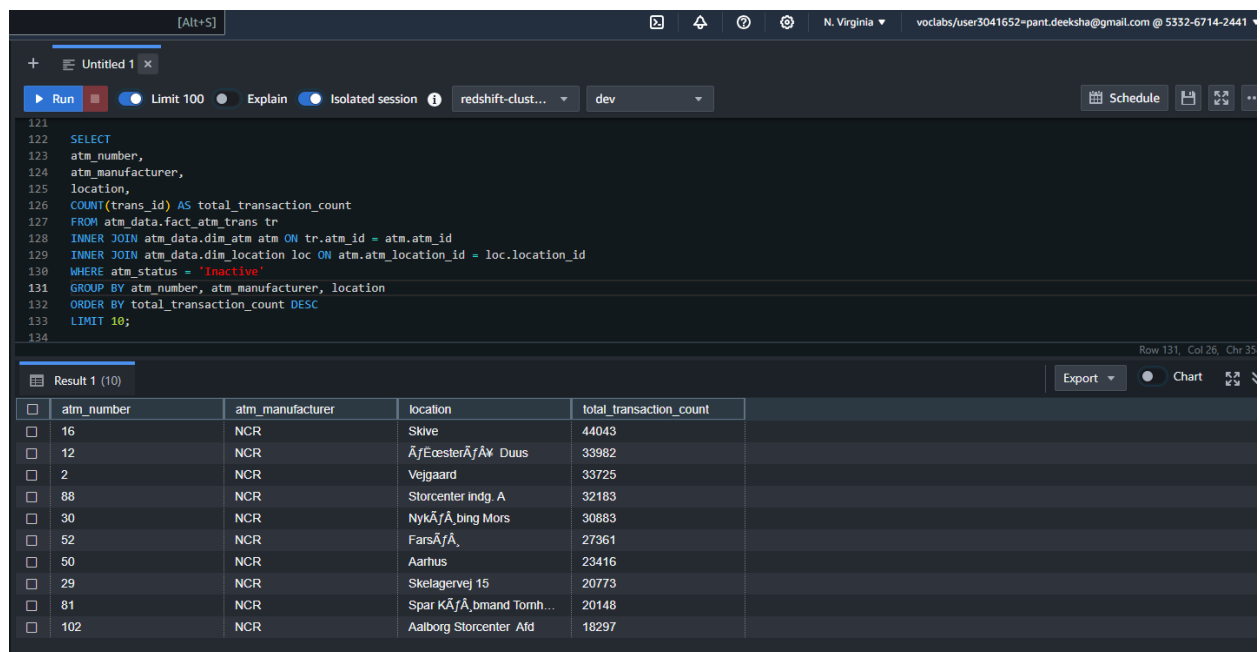


Solving analytical queries on Redshift Cluster

Queries 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

```
SELECT
    atm_number,
    atm_manufacturer,
    location,
    COUNT(trans_id) AS total_transaction_count
FROM atm_data.fact_atm_trans tr
    INNER JOIN atm_data.dim_atm atm ON tr.atm_id = atm.atm_id
    INNER JOIN atm_data.dim_location loc ON atm.atm_location_id = loc.location_id
WHERE atm_status = 'Inactive'
GROUP BY atm_number, atm_manufacturer, location
ORDER BY total_transaction_count DESC
LIMIT 10;
```



The screenshot shows the AWS Redshift Query Editor interface. The SQL query is entered in the editor, and the results are displayed in a table below. The table has 5 columns: atm_number, atm_manufacturer, location, and total_transaction_count. The results are sorted by total_transaction_count in descending order, showing the top 10 ATMs with the most transactions in the 'inactive' state.

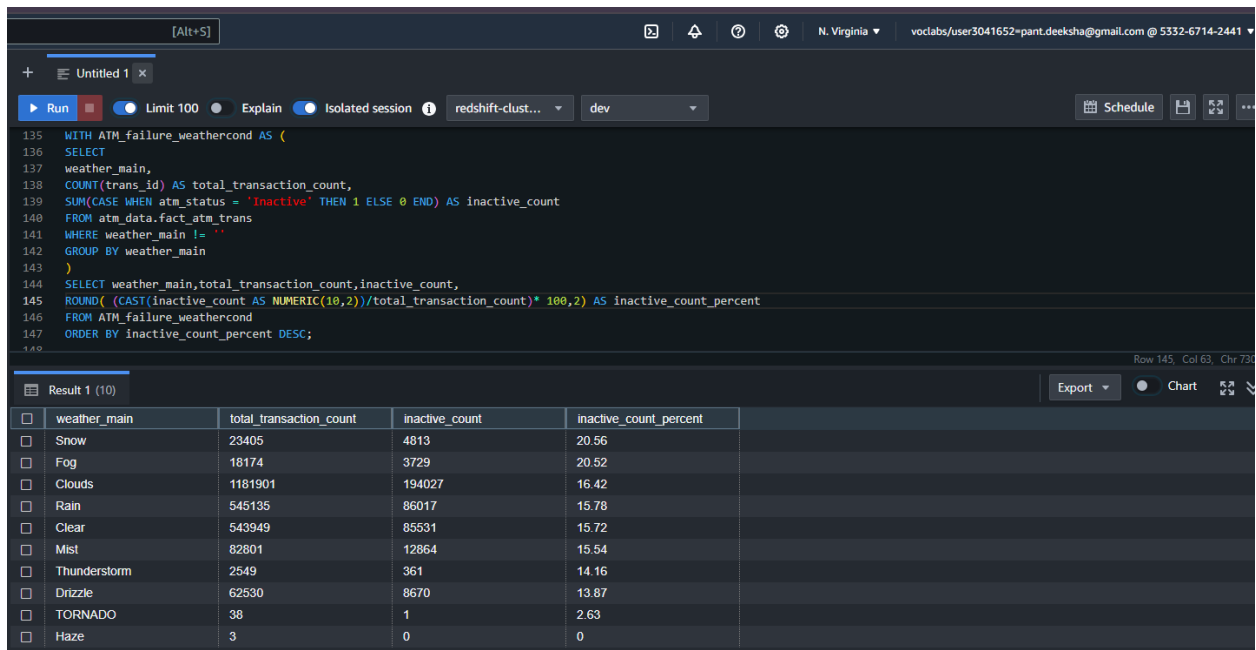
atm_number	atm_manufacturer	location	total_transaction_count
16	NCR	Skive	44043
12	NCR	Århus	33982
2	NCR	Vejgaard	33725
88	NCR	Storcenter indg. A	32183
30	NCR	Nykøbing Mors	30883
52	NCR	Farsø	27361
50	NCR	Aarhus	23416
29	NCR	Skelagervej 15	20773
81	NCR	Spar Kålbmand Tornh...	20148
102	NCR	Aalborg Storcenter Ald	18297

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

<Query>

```
WITH ATM_failure_weathercond AS (  
SELECT  
    weather_main,  
    COUNT(trans_id) AS total_transaction_count,  
    SUM(CASE WHEN atm_status = 'Inactive' THEN 1 ELSE 0 END) AS inactive_count  
FROM atm_data.fact_atm_trans  
WHERE weather_main != "  
GROUP BY weather_main  
)  
SELECT  
    weather_main,  
    total_transaction_count,  
    inactive_count,  
    ROUND( (CAST(inactive_count AS NUMERIC(10,2))/total_transaction_count)* 100,2)  
    AS inactive_count_percent  
FROM ATM_failure_weathercond  
ORDER BY inactive_count_percent DESC;
```

<Screenshot of the resultant table>



The screenshot shows a SQL query editor with a dark theme. The query is a Common Table Expression (CTE) named 'ATM_failure_weathercond' followed by a SELECT statement. The SELECT statement calculates the total transaction count and the percentage of inactive transactions for each weather condition. The results are ordered by the inactive count percentage in descending order.

Row 145, Col 63, Chr 730

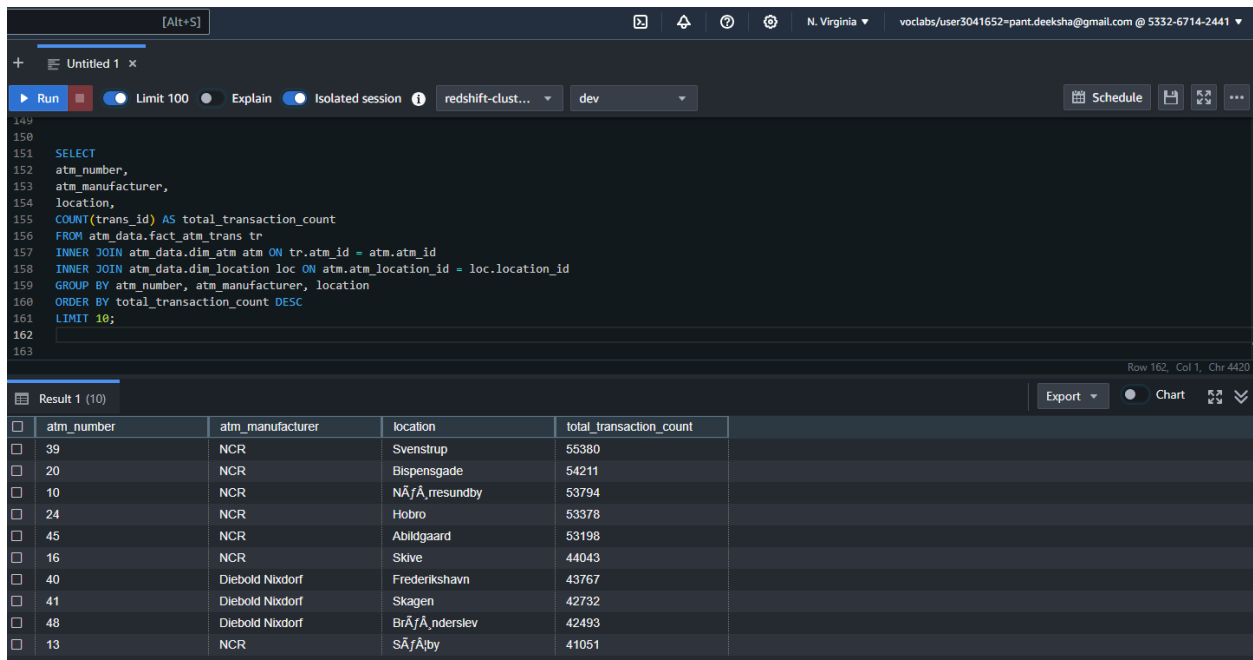
weather_main	total_transaction_count	inactive_count	inactive_count_percent
Snow	23405	4813	20.56
Fog	18174	3729	20.52
Clouds	1181901	194027	16.42
Rain	545135	86017	15.78
Clear	543949	85531	15.72
Mist	82801	12864	15.54
Thunderstorm	2549	361	14.16
Drizzle	62530	8670	13.87
TORNADO	38	1	2.63
Haze	3	0	0

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

<Query>

```
SELECT
    atm_number,
    atm_manufacturer,
    location,
    COUNT(trans_id) AS total_transaction_count
FROM atm_data.fact_atm_trans tr
    INNER JOIN atm_data.dim_atm atm ON tr.atm_id = atm.atm_id
    INNER JOIN atm_data.dim_location loc ON atm.atm_location_id = loc.location_id
GROUP BY atm_number, atm_manufacturer, location
ORDER BY total_transaction_count DESC
LIMIT 10;
```

<Screenshot of the resultant table>



The screenshot shows a SQL IDE interface with a query editor and a results pane. The query editor contains the following SQL code:

```
SELECT
    atm_number,
    atm_manufacturer,
    location,
    COUNT(trans_id) AS total_transaction_count
FROM atm_data.fact_atm_trans tr
    INNER JOIN atm_data.dim_atm atm ON tr.atm_id = atm.atm_id
    INNER JOIN atm_data.dim_location loc ON atm.atm_location_id = loc.location_id
GROUP BY atm_number, atm_manufacturer, location
ORDER BY total_transaction_count DESC
LIMIT 10;
```

The results pane displays the top 10 ATMs with the most transactions, ordered by total_transaction_count in descending order. The results are as follows:

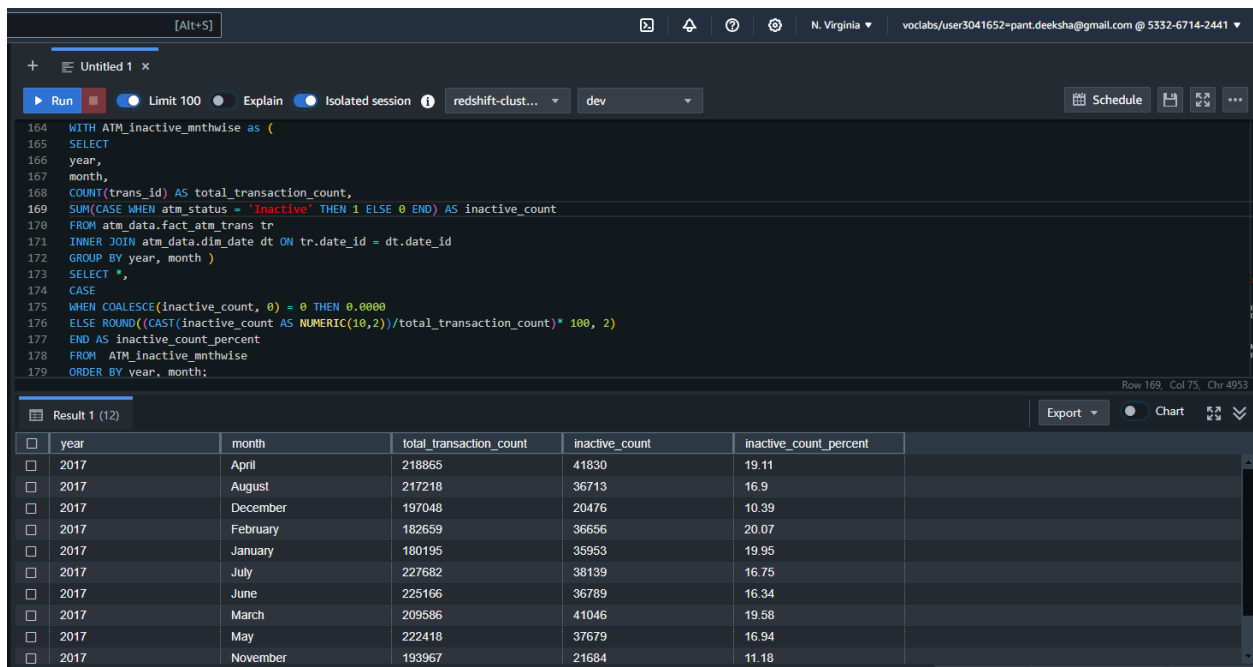
atm_number	atm_manufacturer	location	total_transaction_count
39	NCR	Svenstrup	55380
20	NCR	Bispensgade	54211
10	NCR	NÅfÅresundby	53794
24	NCR	Hobro	53378
45	NCR	Abildgaard	53198
16	NCR	Skive	44043
40	Diebold Nixdorf	Frederikshavn	43767
41	Diebold Nixdorf	Skagen	42732
48	Diebold Nixdorf	BrÅfÅnderslev	42493
13	NCR	SÅfÅtby	41051

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

<Query>

```
WITH ATM_inactive_mnthwise as (  
    SELECT  
        year,  
        month,  
        COUNT(trans_id) AS total_transaction_count,  
        SUM(CASE WHEN atm_status = 'Inactive' THEN 1 ELSE 0 END) AS  
        inactive_count  
    FROM atm_data.fact_atm_trans tr  
        INNER JOIN atm_data.dim_date dt ON tr.date_id = dt.date_id  
    GROUP BY year, month  
)  
SELECT *,  
    CASE  
        WHEN COALESCE(inactive_count, 0) = 0 THEN 0.0000  
        ELSE ROUND((CAST(inactive_count AS  
            NUMERIC(10,2))/total_transaction_count)* 100, 2)  
        END AS inactive_count_percent  
FROM ATM_inactive_mnthwise  
ORDER BY year, month;
```

<Screenshot of the resultant table>



The screenshot shows a SQL query editor with a dark theme. The query is displayed in the main editor area, and the results are shown in a table below. The table has 6 columns: year, month, total_transaction_count, inactive_count, and inactive_count_percent. The results are sorted by year and month.

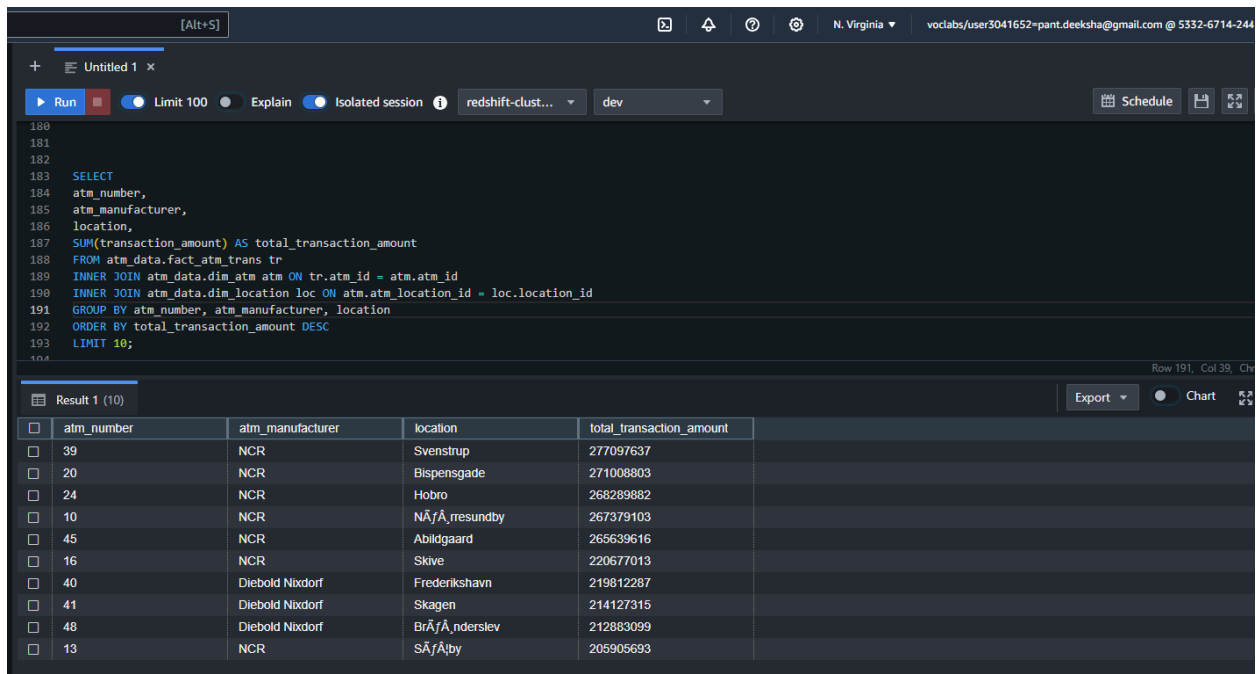
year	month	total_transaction_count	inactive_count	inactive_count_percent
2017	April	218865	41830	19.11
2017	August	217218	36713	16.9
2017	December	197048	20476	10.39
2017	February	182659	36656	20.07
2017	January	180195	35953	19.95
2017	July	227682	38139	16.75
2017	June	225166	36789	16.34
2017	March	209586	41046	19.58
2017	May	222418	37679	16.94
2017	November	193967	21684	11.18

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

<Query>

```
SELECT
    atm_number,
    atm_manufacturer,
    location,
    SUM(transaction_amount) AS total_transaction_amount
FROM atm_data.fact_atm_trans tr
    INNER JOIN atm_data.dim_atm atm ON tr.atm_id = atm.atm_id
    INNER JOIN atm_data.dim_location loc ON atm.atm_location_id = loc.location_id
GROUP BY atm_number, atm_manufacturer, location
ORDER BY total_transaction_amount DESC
LIMIT 10;
```

<Screenshot of the resultant table>



The screenshot shows a Redshift SQL client interface. The top bar includes a search icon, a bell icon, a question mark icon, a settings icon, and the user's name 'N. Virginia' along with their email 'voclabs/user3041652=pant.deeksha@gmail.com @ 5332-6714-244'. Below the top bar, there's a toolbar with a '+', a 'Run' button, a 'Limit 100' toggle, an 'Explain' toggle, an 'Isolated session' toggle, a dropdown menu showing 'redshift-clust...', and another dropdown menu showing 'dev'. There are also 'Schedule', 'Save', and 'Refresh' icons. The main area displays the SQL query from the previous block. Below the query, the results are shown in a table format. The table has 5 columns: 'atm_number', 'atm_manufacturer', 'location', and 'total_transaction_amount'. The results are ordered by 'total_transaction_amount' in descending order, showing the top 10 ATMs.

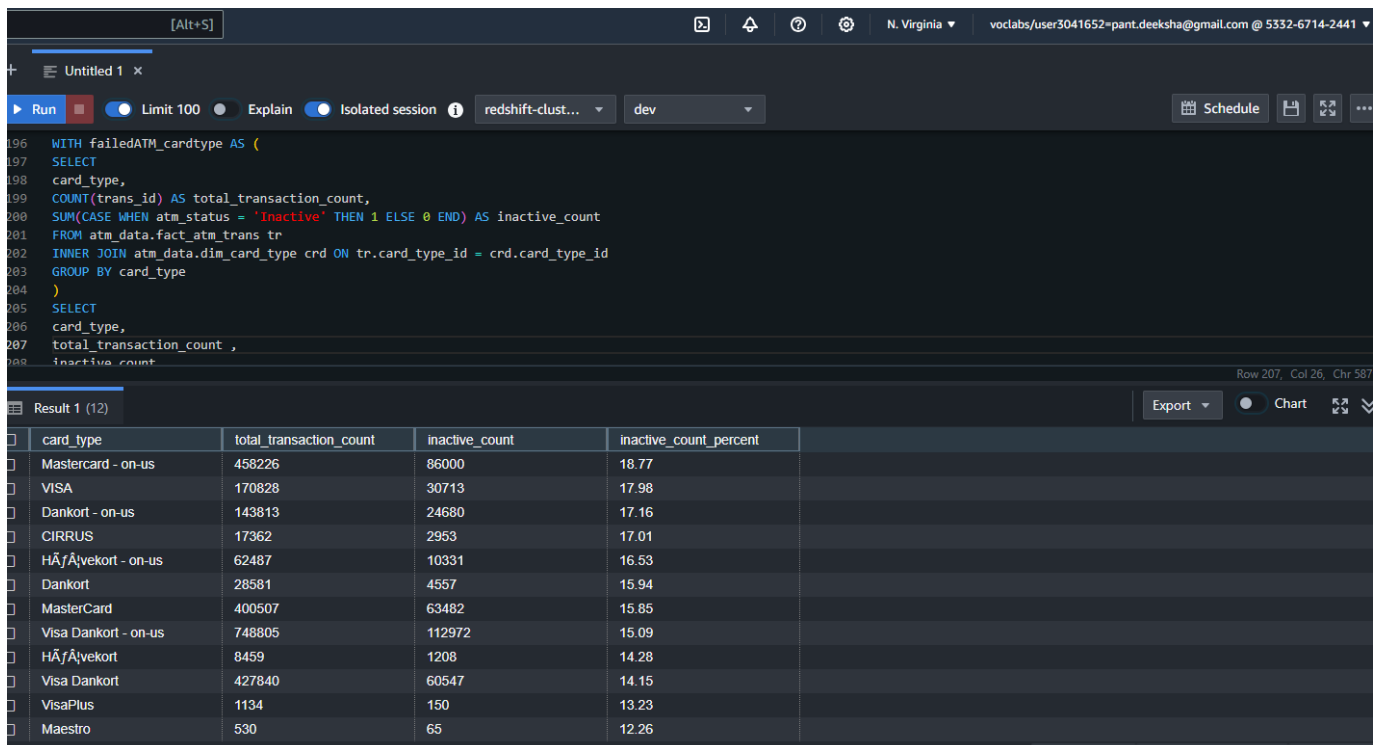
atm_number	atm_manufacturer	location	total_transaction_amount
39	NCR	Svenstrup	277097637
20	NCR	Bispensgade	271008803
24	NCR	Hobro	268289882
10	NCR	NÅfÅ_resundby	267379103
45	NCR	Abildgaard	265639616
16	NCR	Skive	220677013
40	Diebold Nixdorf	Frederikshavn	219812287
41	Diebold Nixdorf	Skagen	214127315
48	Diebold Nixdorf	BrÅfÅ_nderslev	212883099
13	NCR	SÅfÅby	205905693

6. Number of failed ATM transactions across various card types

<Query>

```
WITH failedATM_cardtype AS (  
SELECT  
    card_type,  
    COUNT(trans_id) AS total_transaction_count,  
    SUM(CASE WHEN atm_status = 'Inactive' THEN 1 ELSE 0 END) AS inactive_count  
FROM atm_data.fact_atm_trans tr  
    INNER JOIN atm_data.dim_card_type crd ON tr.card_type_id = crd.card_type_id  
GROUP BY card_type  
)  
SELECT  
    card_type,  
    total_transaction_count ,  
    inactive_count,  
    ROUND((CAST(inactive_count AS NUMERIC(10,2))/ total_transaction_count) * 100, 2)  
    AS inactive_count_percent  
FROM failedATM_cardtype  
ORDER BY inactive_count_percent DESC;
```

<Screenshot of the resultant table>



The screenshot shows a SQL query execution interface. The query is a CTE named 'failedATM_cardtype' followed by a SELECT statement. The results are displayed in a table with 5 columns: card_type, total_transaction_count, inactive_count, and inactive_count_percent. The table is sorted by inactive_count_percent in descending order.

card_type	total_transaction_count	inactive_count	inactive_count_percent
Mastercard - on-us	458226	86000	18.77
VISA	170828	30713	17.98
Dankort - on-us	143813	24680	17.16
CIRRUS	17362	2953	17.01
HÀfÀvekort - on-us	62487	10331	16.53
Dankort	28581	4557	15.94
MasterCard	400507	63482	15.85
Visa Dankort - on-us	748805	112972	15.09
HÀfÀvekort	8459	1208	14.28
Visa Dankort	427840	60547	14.15
VisaPlus	1134	150	13.23
Maestro	530	65	12.26

- 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**

<Query>

```
SELECT
    atm_number,
    atm_manufacturer,
    location,
    CASE WHEN weekday IN ('Sunday', 'Saturday') THEN 1 ELSE 0 END AS
    weekend_flag,
    COUNT(trans_id) AS total_transaction_count
FROM atm_data.fact_atm_trans tr
    INNER JOIN atm_data.dim_atm atm ON tr.atm_id = atm.atm_id
    INNER JOIN atm_data.dim_location loc ON atm.atm_location_id = loc.location_id
    INNER JOIN atm_data.dim_date dt ON tr.date_id = dt.date_id
GROUP BY
    atm_number,
    atm_manufacturer,
    location,
    weekend_flag
ORDER BY
    atm_number,
    atm_manufacturer,
    location,
    weekend_flag,
    total_transaction_count
LIMIT 10;
```

<Screenshot of the resultant table>

[Alt+S] N. Virginia voclabs/user3041652=pant.deeksha@gmail.com @ 5332-6714-24

Run Limit 100 Explain Isolated session redshift-clust... dev Schedule

```

214 SELECT
215     atm_number,
216     atm_manufacturer,
217     location,
218     CASE WHEN weekday IN ('Sunday', 'Saturday') THEN 1 ELSE 0 END AS
219     weekend_flag,
220     COUNT(trans_id) AS total_transaction_count
221 FROM atm_data.fact_atm_trans tr
222 INNER JOIN atm_data.dim_atm atm ON tr.atm_id = atm.atm_id
223 INNER JOIN atm_data.dim_location loc ON atm.atm_location_id = loc.location_id
224 INNER JOIN atm_data.dim_date dt ON tr.date_id = dt.date_id
225 GROUP BY
226     atm_number,
227     atm_manufacturer,
228     location,

```

Result 1 (10) Export Chart

	atm_number	atm_manufacturer	location	weekend_flag	total_transaction_count
<input type="checkbox"/>	10	NCR	NÃfÃ,resundby	0	41667
<input type="checkbox"/>	10	NCR	NÃfÃ,resundby	1	12127
<input type="checkbox"/>	100	NCR	Intern Skive	0	17812
<input type="checkbox"/>	100	NCR	Intern Skive	1	1
<input type="checkbox"/>	101	NCR	Bryggen Vejle	0	11693
<input type="checkbox"/>	101	NCR	Bryggen Vejle	1	3247
<input type="checkbox"/>	102	NCR	Aalborg Storcenter Afd	0	14556
<input type="checkbox"/>	102	NCR	Aalborg Storcenter Afd	1	3741
<input type="checkbox"/>	103	Diebold Nixdorf	Vejgaard	0	18570
<input type="checkbox"/>	103	Diebold Nixdorf	Vejgaard	1	2607

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

<Query>

```

WITH Vejgaard_ATMtrans AS (
SELECT
    atm_number,
    atm_manufacturer,
    location,
    weekday,
    COUNT(trans_id) AS total_transaction_count
FROM atm_data.fact_atm_trans tr
    INNER JOIN atm_data.dim_atm atm ON tr.atm_id = atm.atm_id
    INNER JOIN atm_data.dim_location loc ON atm.atm_location_id = loc.location_id
    INNER JOIN atm_data.dim_date dt ON tr.date_id = dt.date_id
WHERE location = 'Vejgaard'
GROUP BY
    atm_number,
    atm_manufacturer,
    location,
    weekday
),
Vejgaard_activeday_ATMwise AS (
SELECT
    atm_number,
    atm_manufacturer,

```

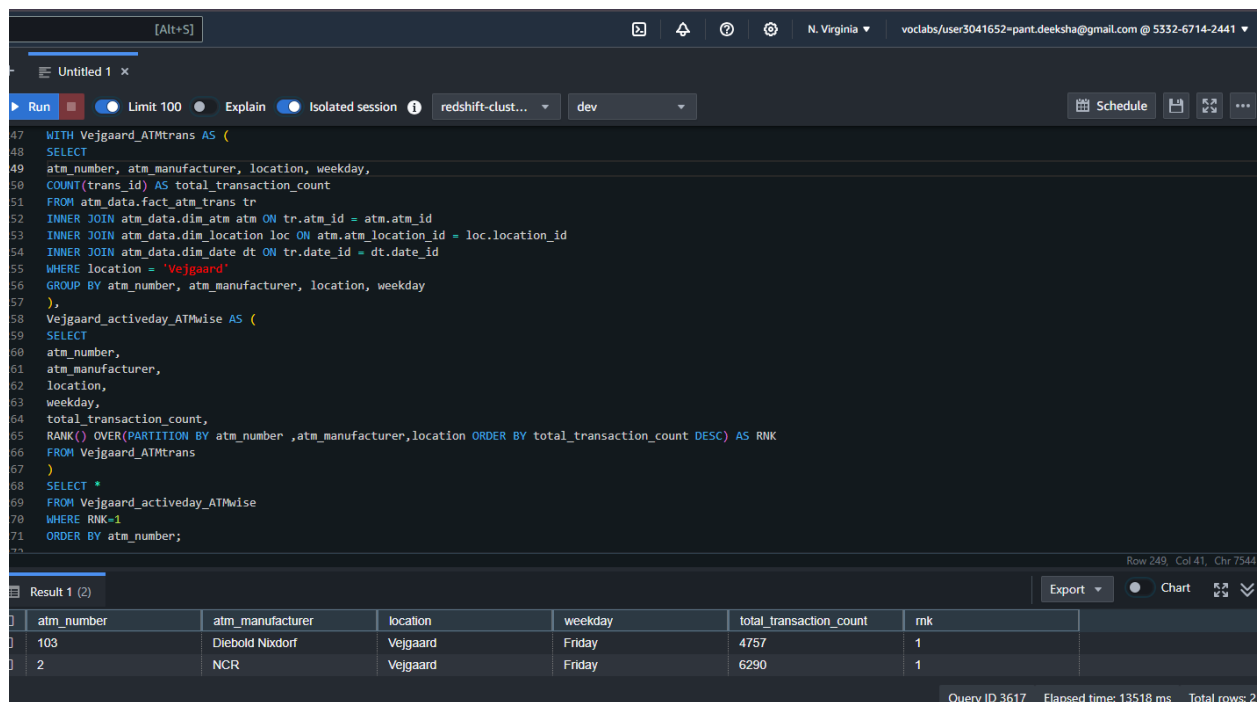


```

        location,
        weekday,
        total_transaction_count,
        RANK() OVER(PARTITION BY atm_number ,atm_manufacturer,location ORDER BY
        total_transaction_count DESC) AS RNK
FROM Vejgaard_ATMtrans
)
SELECT *
FROM Vejgaard_activeday_ATMwise
WHERE RNK=1
ORDER BY atm_number;

```

<Screenshot of the resultant table>



The screenshot shows a Redshift SQL client interface. The top bar includes a search field with "[Alt+S]", a toolbar with icons for document, alert, refresh, and settings, and a user profile section for "N. Virginia" with a session ID "voclabs/user3041652-pant.deeksha@gmail.com @ 5332-6714-2441". Below the toolbar, there's a tab labeled "Untitled 1" and a status bar with "Run", "Limit 100", "Explain", "Isolated session", "redshift-clust...", and "dev". The main area displays a SQL query with line numbers 47 to 71. The query defines a CTE for "Vejgaard_ATMtrans" and another for "Vejgaard_activeday_ATMwise", which uses the first CTE in a RANK() window function. The final SELECT statement filters for RNK=1 and orders by atm_number. At the bottom, the "Result 1 (2)" section shows a table with 7 columns: atm_number, atm_manufacturer, location, weekday, total_transaction_count, and mk. Two rows of data are displayed. The status bar at the very bottom indicates "Query ID 3617", "Elapsed time: 13518 ms", and "Total rows: 2".

```

47 WITH Vejgaard_ATMtrans AS (
48 SELECT
49   atm_number, atm_manufacturer, location, weekday,
50   COUNT(trans_id) AS total_transaction_count
51 FROM atm_data.fact_atm_trans tr
52 INNER JOIN atm_data.dim_atm atm ON tr.atm_id = atm.atm_id
53 INNER JOIN atm_data.dim_location loc ON atm.atm_location_id = loc.location_id
54 INNER JOIN atm_data.dim_date dt ON tr.date_id = dt.date_id
55 WHERE location = 'Vejgaard'
56 GROUP BY atm_number, atm_manufacturer, location, weekday
57 ),
58 Vejgaard_activeday_ATMwise AS (
59 SELECT
60   atm_number,
61   atm_manufacturer,
62   location,
63   weekday,
64   total_transaction_count,
65   RANK() OVER(PARTITION BY atm_number ,atm_manufacturer,location ORDER BY total_transaction_count DESC) AS RNK
66 FROM Vejgaard_ATMtrans
67 )
68 SELECT *
69 FROM Vejgaard_activeday_ATMwise
70 WHERE RNK=1
71 ORDER BY atm_number;

```

	atm_number	atm_manufacturer	location	weekday	total_transaction_count	mk
1	103	Diebold Nixdorf	Vejgaard	Friday	4757	1
2	2	NCR	Vejgaard	Friday	6290	1

Query ID 3617 Elapsed time: 13518 ms Total rows: 2