

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

<Query>

```
SELECT
    a.atm_number,
    a.atm_manufacturer,
    l.location,
    count(f.trans_id) as total_transaction_count,
    SUM(CASE WHEN f.atm_status = 'Inactive'
        THEN 1
        ELSE 0
        END) as inactive_count,
    (inactive_count/total_transaction_count)*100 as inactive_count_percent
FROM
    etl_atm_data.DIM_ATM as a,
    etl_atm_data.FACT_ATM_TRANS as f,
    etl_atm_data.DIM_LOCATION as l
WHERE
    a.atm_id = f.atm_id AND a.atm_location_id = l.location_id
GROUP BY
    a.atm_number,
    a.atm_manufacturer,
    l.location
ORDER BY
    inactive_count DESC
LIMIT 10;
```

<Screenshot of the resultant table>

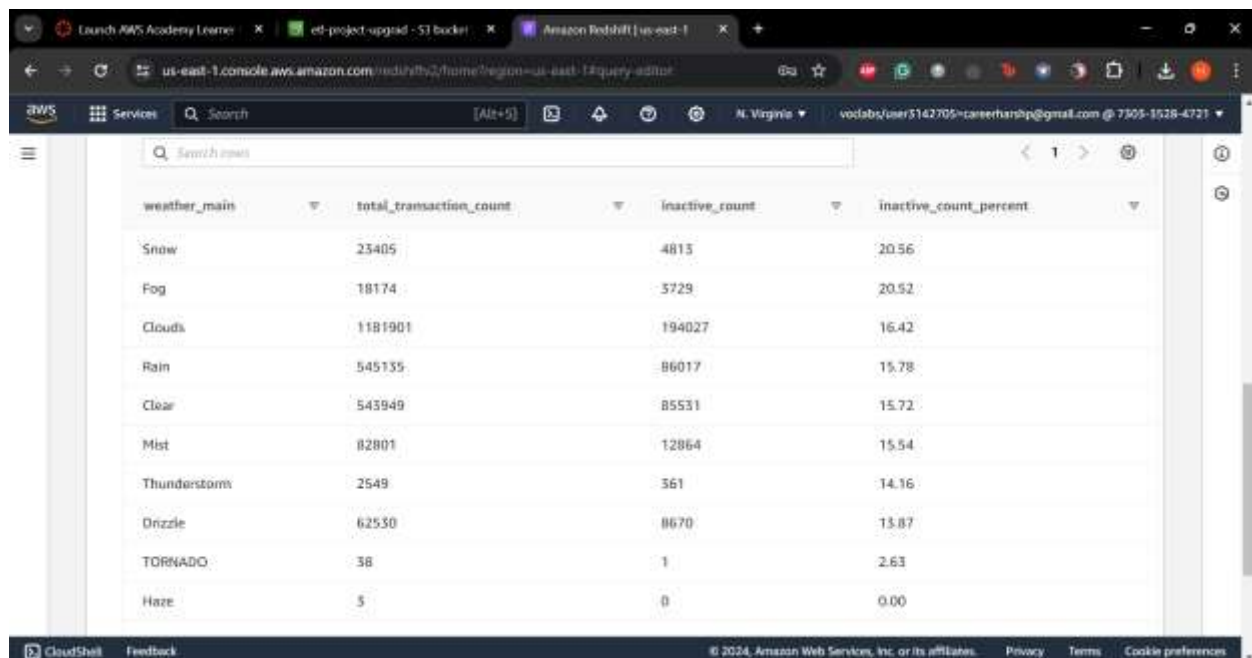
atm_number	atm_manufacturer	location	total_transaction_count	inactive_count	inactive_count_percent
16	NCR	Skive Lobby	44043	44043	100
12	NCR	Århus Øster Århus	33982	33982	100
2	NCR	Vejgaard	33725	33725	100
88	NCR	Storcenter indg. A	52185	52185	100
30	NCR	Nykøbing Mors	30883	30883	100
52	NCR	Farsø	27561	27561	100
50	NCR	Aarhus	25416	25416	100
29	NCR	Skelagervej 15	20775	20775	100
81	NCR	Spar København	20148	20148	100
102	NCR	Aalborg Storcenter	18297	18297	100

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

<Query>

```
WITH failed_atm 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
    etl_atm_data.FACT_ATM_TRANS
WHERE
    weather_main != ''
GROUP BY
    weather_main)
SELECT *,
    ROUND(CAST(inactive_count AS numeric(10,2))/ total_transaction_count*100,2) AS
    inactive_count_percent
FROM
    failed_atm
ORDER BY
    inactive_count_percent DESC;
```

<Screenshot of the resultant table>



The screenshot shows the Amazon Redshift console interface. The browser address bar indicates the URL is `us-east-1.console.aws.amazon.com/redshift2/home?region=us-east-1#query-editor`. The AWS Services navigation bar is visible at the top. The main content area displays a table with the following data:

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	5	0	0.00

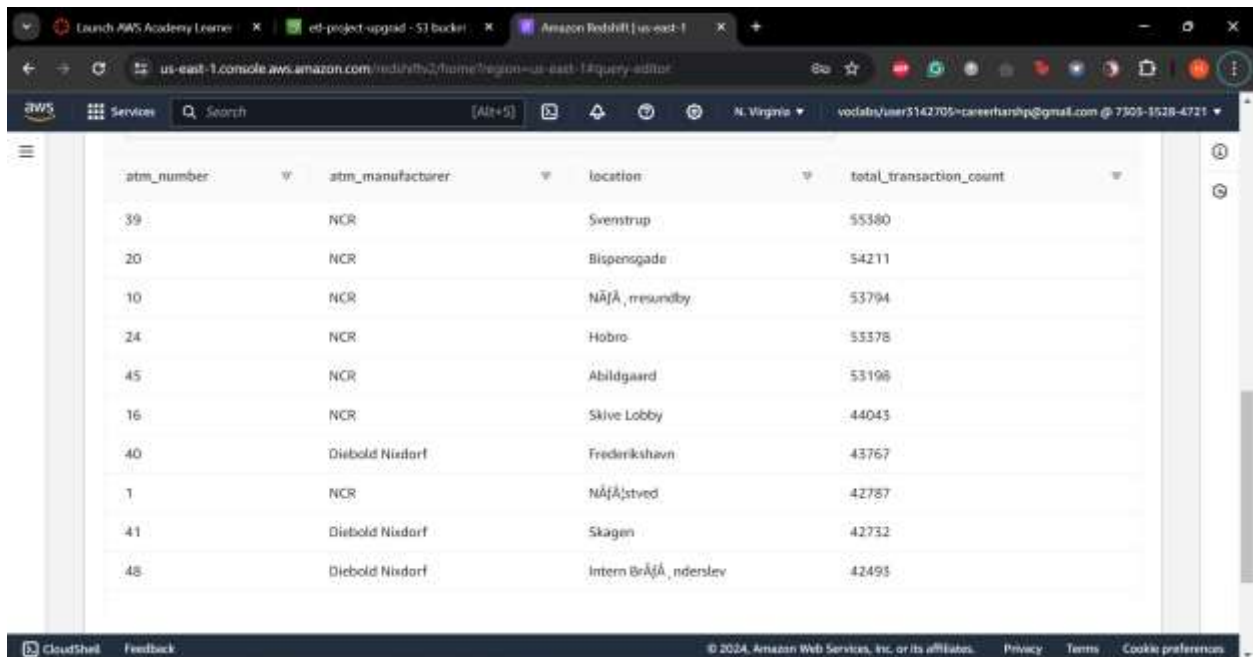
The footer of the console shows the CloudShell logo, a Feedback link, and the copyright notice: © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences.

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

<Query>

```
SELECT
    a.atm_number,
    a.atm_manufacturer,
    l.location,
    count(f.trans_id) as total_transaction_count
FROM
    etl_atm_data.DIM_ATM as a,
    etl_atm_data.FACT_ATM_TRANS as f,
    etl_atm_data.DIM_LOCATION as l
WHERE
    a.atm_id = f.atm_id AND a.atm_location_id = l.location_id
GROUP BY
    a.atm_number,
    a.atm_manufacturer,
    l.location
ORDER BY
    total_transaction_count DESC
LIMIT 10;
```

<Screenshot of the resultant table>



The screenshot shows the AWS Redshift console interface. The browser address bar indicates the URL is `us-east-1.console.aws.amazon.com/redshiftv2/home?region=us-east-1#query-editor`. The console displays a table with the following data:

atm_number	atm_manufacturer	location	total_transaction_count
39	NCR	Svenstrup	55380
20	NCR	Bispensogade	54211
10	NCR	NÅfjÅ, mesundby	53794
24	NCR	Hobro	53378
45	NCR	Abildgaard	53198
16	NCR	Skive Lobby	44043
40	Diebold Nixdorf	Frederikshavn	43767
1	NCR	NÅfjÅstved	42787
41	Diebold Nixdorf	Skagen	42732
48	Diebold Nixdorf	Intern BrÅfjÅ, nderslev	42495

The footer of the screenshot shows the CloudShed logo, a feedback link, and the copyright notice: © 2024, Amazon Web Services, Inc. or its affiliates. Privacy, Terms, and Cookie preferences links are also visible.

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

<Query>

```
WITH failed_atm_monthly AS (  
    SELECT  
        d.year,  
        d.month,  
        count(f.trans_id) as total_transaction_count,  
        SUM(CASE WHEN f.atm_status = 'Inactive'  
            THEN 1  
            ELSE 0  
            END) as inactive_count  
    FROM  
        etl_atm_data.FACT_ATM_TRANS as f,  
        etl_atm_data.DIM_DATE as d  
    WHERE  
        f.date_id = d.date_id  
    GROUP BY  
        d.year,  
        d.month)  
SELECT *,  
    ROUND(CAST(inactive_count AS numeric(10,2))/ total_transaction_count*100,2) AS  
inactive_count_percent  
FROM  
    failed_atm_monthly  
ORDER BY  
    month;  
<Screenshot of the resultant table>
```

Launch AWS Academy Course | Redshift - Query editor

us-east-1.console.aws.amazon.com/redshift/home?region=us-east-1#/query-editor?cluster=redshift-foo-etl

Services Search [Alt+S] N. Virginia vodabs/user5142705-careerhanip@gmail.com @ 7309-5528-4721

Search rows

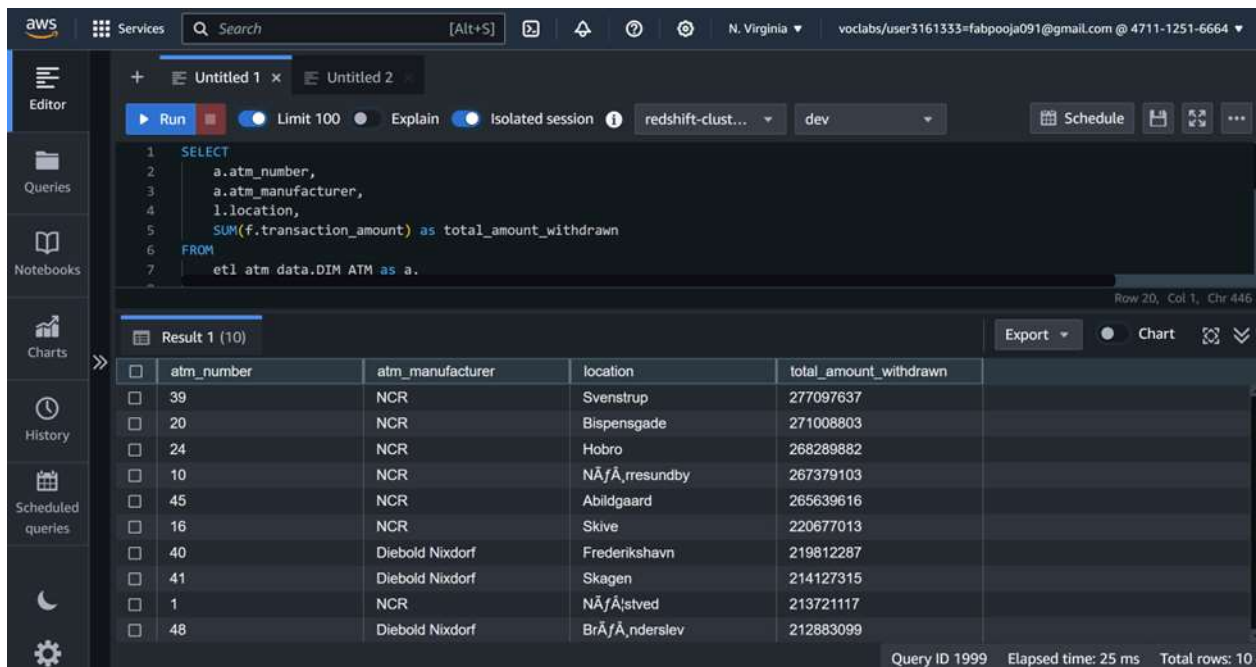
year	month	total_transaction_count	inactive_count	inactive_count_percent
2017	April	218865	41830	19.11
2017	August	217218	36713	16.90
2017	December	197048	20476	10.39
2017	February	182659	36656	20.07
2017	January	180195	35953	19.95
2017	July	227682	38139	16.75
2017	June	225166	36789	16.34
2017	March	209586	41046	19.58
2017	May	222418	37679	16.94
2017	November	193967	21684	11.18
2017	October	191667	21780	11.36
2017	September	202101	28913	14.31

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

<Query>

```
SELECT
  a.atm_number,
  a.atm_manufacturer,
  l.location,
  SUM(f.transaction_amount) as total_amount_withdrawn
FROM
  etl_atm_data.DIM_ATM as a,
  etl_atm_data.FACT_ATM_TRANS as f,
  etl_atm_data.DIM_LOCATION as l
WHERE
  a.atm_id = f.atm_id
  AND a.atm_location_id = l.location_id
GROUP BY
  a.atm_number,
  a.atm_manufacturer,
  l.location
ORDER BY
  total_amount_withdrawn DESC
LIMIT 10;
```

<Screenshot of the resultant table>



The screenshot shows the AWS Redshift console interface. The top navigation bar includes the AWS logo, 'Services', a search bar, and user information. The left sidebar contains icons for Editor, Queries, Notebooks, Charts, History, and Scheduled queries. The main area displays a SQL query in the editor, which is the same query as provided in the previous block. Below the editor, the 'Run' button is highlighted, and the 'Limit 100' option is selected. The 'Result 1 (10)' table is displayed below the query, showing the top 10 ATMs by total withdrawn amount. The table has five columns: atm_number, atm_manufacturer, location, and total_amount_withdrawn. The results are sorted in descending order of total_amount_withdrawn.

atm_number	atm_manufacturer	location	total_amount_withdrawn
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
1	NCR	NÅfÅ, stved	213721117
48	Diebold Nixdorf	BrÅfÅ, nderslev	212883099

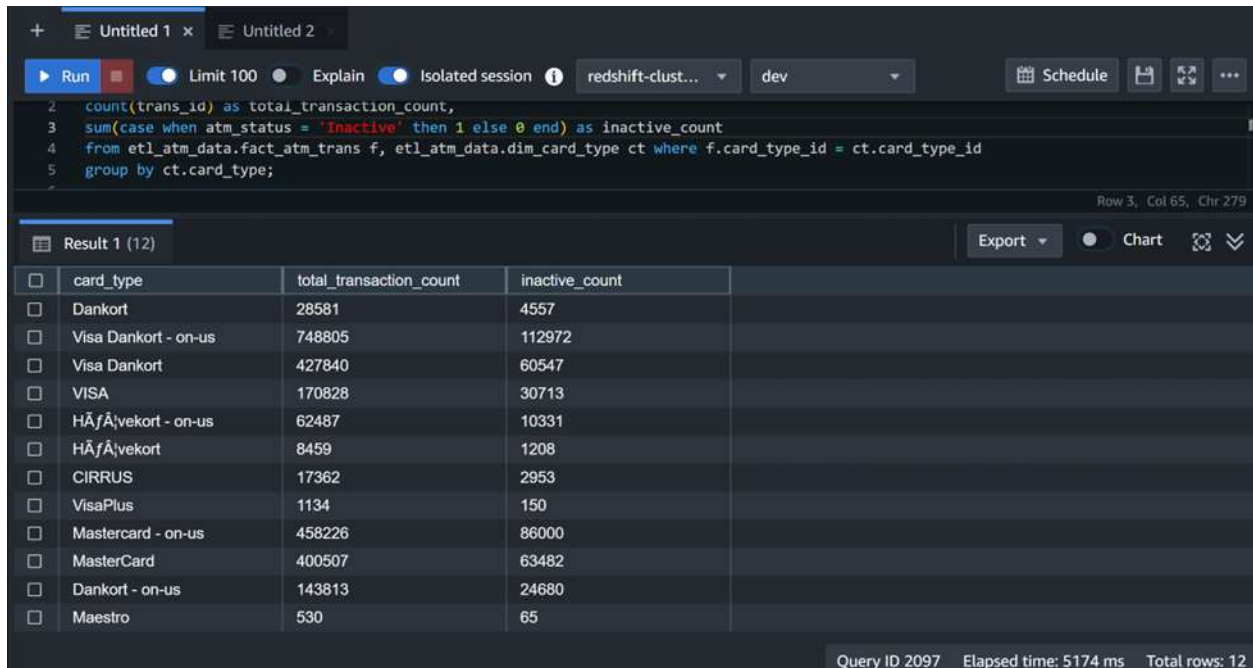
Query ID 1999 Elapsed time: 25 ms Total rows: 10

6. Number of failed ATM transactions across various card types

<Query>

```
SELECT ct.card_type,
COUNT(trans_id) as total_transaction_count,
SUM(case when atm_status = 'Inactive' then 1 else 0 end) as inactive_count
FROM etl_atm_data.fact_atm_trans f, etl_atm_data.dim_card_type ct
WHERE f.card_type_id = ct.card_type_id
GROUP BY ct.card_type;
```

<Screenshot of the resultant table>



The screenshot shows a SQL query execution interface. The query is as follows:

```
count(trans_id) as total_transaction_count,
sum(case when atm_status = 'Inactive' then 1 else 0 end) as inactive_count
from etl_atm_data.fact_atm_trans f, etl_atm_data.dim_card_type ct where f.card_type_id = ct.card_type_id
group by ct.card_type;
```

The results are displayed in a table with 4 columns: card_type, total_transaction_count, and inactive_count. The table contains 12 rows of data.

card_type	total_transaction_count	inactive_count
Dankort	28581	4557
Visa Dankort - on-us	748805	112972
Visa Dankort	427840	60547
VISA	170828	30713
HÃfÃ\vekort - on-us	62487	10331
HÃfÃ\vekort	8459	1208
CIRRUS	17362	2953
VisaPlus	1134	150
Mastercard - on-us	458226	86000
MasterCard	400507	63482
Dankort - on-us	143813	24680
Maestro	530	65

Query ID 2097 Elapsed time: 5174 ms Total rows: 12

- 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
  a.atm_number,
  a.atm_manufacturer,
  l.location,
  CASE
    WHEN d.weekday IN ('Saturday', 'Sunday') THEN 1
    ELSE 0
  END AS weekend_flag,
  COUNT(f.trans_id) AS total_transaction_count
  FROM
    etl_atm_data.fact_atm_trans f
  JOIN
    etl_atm_data.dim_atm a ON f.atm_id = a.atm_id
  JOIN
    etl_atm_data.dim_location l ON a.atm_location_id = l.location_id
  JOIN
    etl_atm_data.dim_date d ON f.date_id = d.date_id
  GROUP BY
    a.atm_number,
    a.atm_manufacturer,
    l.location,
    weekend_flag
  ORDER BY
    a.atm_number,
    a.atm_manufacturer,
    l.location,
    weekend_flag,
    total_transaction_count DESC
  LIMIT 10;
```

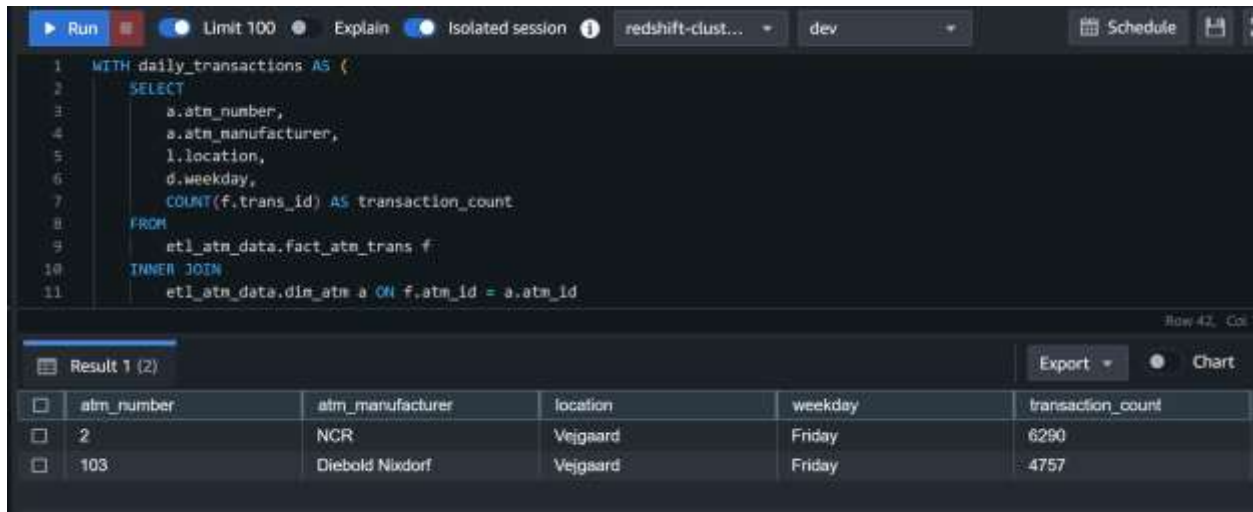
<Screenshot of the resultant table>

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

<Query>

```
WITH daily_transactions AS (  
    SELECT  
        a.atm_number,  
        a.atm_manufacturer,  
        l.location,  
        d.weekday,  
        COUNT(f.trans_id) AS transaction_count  
    FROM  
        etl_atm_data.fact_atm_trans f  
    INNER JOIN  
        etl_atm_data.dim_atm a ON f.atm_id = a.atm_id  
    INNER JOIN  
        etl_atm_data.dim_location l ON a.atm_location_id = l.location_id  
    INNER JOIN  
        etl_atm_data.dim_date d ON f.date_id = d.date_id  
    WHERE  
        l.location = 'Vejgaard'  
    GROUP BY  
        a.atm_number,  
        a.atm_manufacturer,  
        l.location,  
        d.weekday  
)  
ranked_transactions AS (  
    SELECT  
        *,  
        ROW_NUMBER() OVER (PARTITION BY atm_number ORDER BY  
transaction_count DESC) AS rank  
    FROM  
        daily_transactions  
)  
SELECT  
    atm_number,  
    atm_manufacturer,  
    location,  
    weekday,  
    transaction_count  
FROM  
    ranked_transactions  
WHERE  
    rank = 1;
```

<Screenshot of the resultant table>



The screenshot shows a Redshift SQL query editor interface. At the top, there are buttons for 'Run', 'Limit 100', 'Explain', 'Isolated session', and a dropdown menu showing 'redshift-clust...' and 'dev'. There are also 'Schedule' and 'Save' icons. The SQL query is as follows:

```
1 WITH daily_transactions AS (  
2     SELECT  
3         a.atm_number,  
4         a.atm_manufacturer,  
5         l.location,  
6         d.weekday,  
7         COUNT(f.trans_id) AS transaction_count  
8     FROM  
9         etl_atm_data.fact_atm_trans f  
10    INNER JOIN  
11    etl_atm_data.dim_atm a ON f.atm_id = a.atm_id
```

Below the query editor, the results are displayed in a table. The table has 5 columns: atm_number, atm_manufacturer, location, weekday, and transaction_count. The first two rows of data are visible:

atm_number	atm_manufacturer	location	weekday	transaction_count
2	NCR	Vejgaard	Friday	6290
103	Diebold Nixdorf	Vejgaard	Friday	4757

At the bottom right of the results table, there are buttons for 'Export' and 'Chart'.