EPAM Systems, RD Dep., RD Dep.

AWS CLOUD FOR DATA ENGINEERING

Redshift

INTRODUCTION

Let’s assume that we’ve created our first “data lake” by loading data from the source system to S3. Data has been loaded to your bucket and placed to understandable and well-organized folder structure. Glue crawler was used to create a data catalogue and some exploratory analysis made using Athena.

Now, after understanding the business meaning of the data in the source system and its structure, you are ready to provision a customer with an end-to-end analytical product. You will take the data from the source system, load it to Redshift, make some analytical transformations according to customer requirements, and create a report.

There is no sense in DWH if it does not create any business value. Using Oracle you’ve created structured data model (with some extent of normalization). All these preparations are the base for creating reports with different KPIs or can be used for predictions. There can be different business departments that will consume data from these reports, get insights and make strategical business decisions.

TASKS

CREATING THE REPORT

1. Login to Redshift. **The EPAM VPN should be enabled.** The public access to cluster is disabled, therefore you can use SSH tunnel to establish connection. Use the EC2 machine from HW2 or launch the new one. Perform the steps below:

1.1 Update packages

On any Linux before installing packages, a user should run a system update command that will ensure all the latest available updates are installed on the system. Plus, this will also refresh the DNF package cache. So, get access to your terminal or connect to your Amazon Linux instance via SSH and run the following command:

```
sudo dnf update
```

1.2 Install PostgreSQL 15 on Amazon Linux 2023:

Well, the best thing currently you don’t need to add any repository to get the PostgreSQL version 15 on your Amazon Linux 2023 because it is available through its system default repo. So, what you have to do is just run the given command. It will install both the client and server parts of the PostgreSQL Database system on your Linux:

```
sudo dnf install postgresql15.x86_64 postgresql15-server
```

1.3 Initialize the PostgreSQL Database:

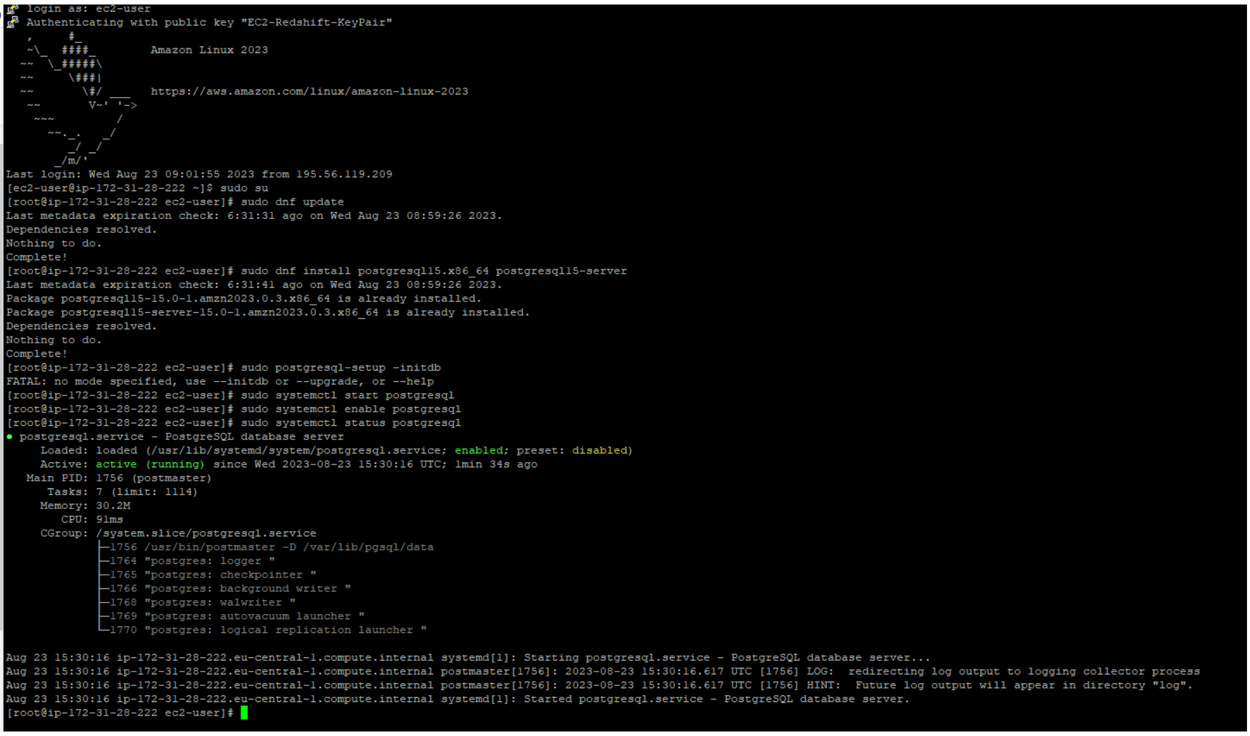
```
sudo postgresql-setup –initdb
```

1.4 Start and Enable Service:

```
sudo systemctl start postgresql
sudo systemctl enable postgresql
```

1.5 To confirm the service is running without any errors, here is the command to follow:

```
sudo systemctl status postgresql
```

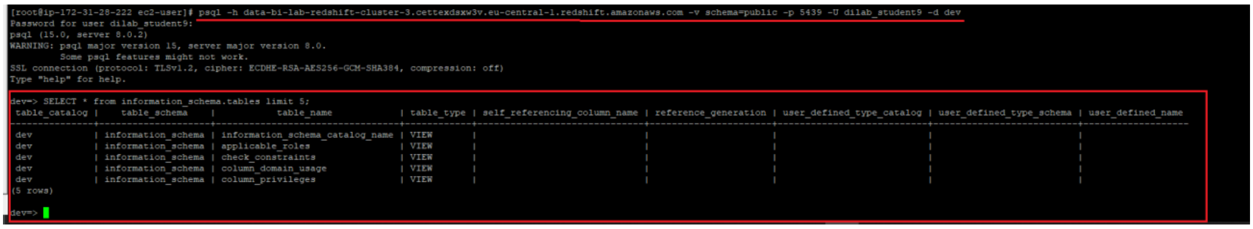


```
login as: ec2-user
Authenticating with public key "EC2-Redshift-KeyPair"
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Wed Aug 23 09:01:55 2023 from 195.56.119.209
[ec2-user@ip-172-31-28-222 ~]$ sudo su
[root@ip-172-31-28-222 ec2-user]# sudo dnf update
Last metadata expiration check: 6:31:31 ago on Wed Aug 23 08:59:26 2023.
Dependencies resolved.
Nothing to do.
Complete!
[root@ip-172-31-28-222 ec2-user]# sudo dnf install postgresql15.x86_64 postgresql15-server
Last metadata expiration check: 6:31:41 ago on Wed Aug 23 08:59:26 2023.
Package postgresql15-15.0-1.amzn2023.0.3.x86_64 is already installed.
Package postgresql15-server-15.0-1.amzn2023.0.3.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[root@ip-172-31-28-222 ec2-user]# sudo postgresql-setup -initdb
FATAL: no mode specified, use --initdb or --upgrade, or --help
[root@ip-172-31-28-222 ec2-user]# sudo systemctl start postgresql
[root@ip-172-31-28-222 ec2-user]# sudo systemctl enable postgresql
[root@ip-172-31-28-222 ec2-user]# sudo systemctl status postgresql
● postgresql.service - PostgreSQL database server
     Loaded: loaded (/usr/lib/systemd/system/postgresql.service; enabled; preset: disabled)
     Active: active (running) since Wed 2023-08-23 15:30:16 UTC; 1min 34s ago
   Main PID: 1756 (postmaster)
      Tasks: 7 (limit: 1114)
     Memory: 30.2M
        CPU: 91ms
     CGroup: /system.slice/postgresql.service
             ├─1756 /usr/bin/postmaster -D /var/lib/pgsql/data
             ├─1764 "postgres: logger "
             ├─1765 "postgres: checkpointer "
             ├─1766 "postgres: background writer "
             ├─1768 "postgres: walwriter "
             ├─1769 "postgres: autovacuum launcher "
             └─1770 "postgres: logical replication launcher "

Aug 23 15:30:16 ip-172-31-28-222.eu-central-1.compute.internal systemd[1]: Starting postgresql.service - PostgreSQL database server...
Aug 23 15:30:16 ip-172-31-28-222.eu-central-1.compute.internal postmaster[1756]: 2023-08-23 15:30:16.617 UTC [1756] LOG:  redirecting log output to logging collector process
Aug 23 15:30:16 ip-172-31-28-222.eu-central-1.compute.internal postmaster[1756]: 2023-08-23 15:30:16.617 UTC [1756] HINT:  Future log output will appear in directory "log".
Aug 23 15:30:16 ip-172-31-28-222.eu-central-1.compute.internal systemd[1]: Started postgresql.service - PostgreSQL database server.
[root@ip-172-31-28-222 ec2-user]#
```

1.6 Use the following command to check connection between EC2 and Redshift(specify the user_name / password provided to you by mentors):

```
    psql -h data-bi-lab-redshift-cluster-3.cettexdsxw3v.eu-central- 1.redshift.amazonaws.com -v
schema=public -p 5439 -U <user_name> -d dev
```

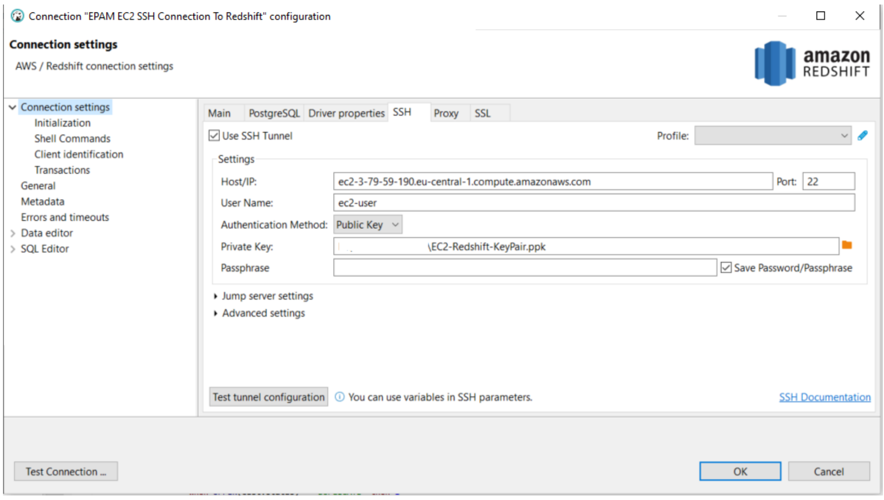


```
[root@ip-172-31-28-222 ec2-user]# psql -h data-bi-lab-redshift-cluster-3.cettexdsxw3v.eu-central-1.redshift.amazonaws.com -v schema=public -p 5439 -U dilab_student9 -d dev
Password for user dilab_student9:
psql (15.0, server 8.0.2)
WARNING: psql major version 15, server major version 8.0.
         Some psql features might not work.
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, compression: off)
Type "help" for help.

dev=> SELECT * from information_schema.tables limit 5;
 table_catalog |    table_schema    |            table_name             | table_type | self_referencing_column_name | reference_generation | user_defined_type_catalog | user_defined_type_schema | user_defined_name
---------------+--------------------+-----------------------------------+------------+------------------------------+----------------------+---------------------------+--------------------------+-------------------
 dev           | information_schema | information_schema_catalog_name   | VIEW       |                              |                      |                           |                          |
 dev           | information_schema | applicable_roles                  | VIEW       |                              |                      |                           |                          |
 dev           | information_schema | check_constraints                 | VIEW       |                              |                      |                           |                          |
 dev           | information_schema | column_domain_usage               | VIEW       |                              |                      |                           |                          |
 dev           | information_schema | column_privileges                 | VIEW       |                              |                      |                           |                          |
(5 rows)

dev=>
```

1.7 Configure connection settings:



Connection "EPAM EC2 SSH Connection To Redshift" configuration

Connection settings

AWS / Redshift connection settings

☑ Use SSH Tunnel

Profile:

Settings

Host/IP: ec2-3-79-59-190.eu-central-1.compute.amazonaws.com Port: 22

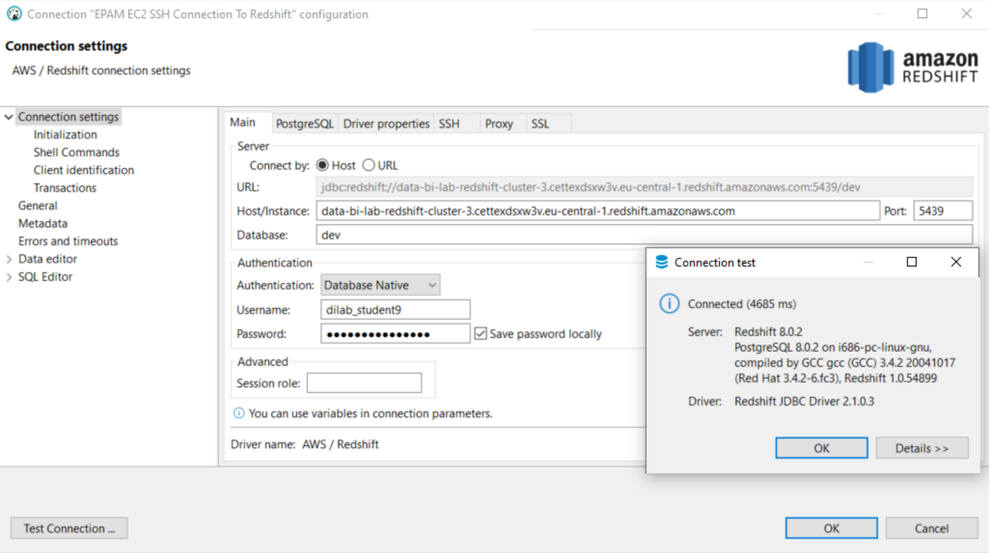
User Name: ec2-user

Authentication Method: Public Key

Private Key: \EC2-Redshift-KeyPair.ppk

Passphrase ☑ Save Password/Passphrase

You can use variables in SSH parameters.



Connection "EPAM EC2 SSH Connection To Redshift" configuration

Connection settings

AWS / Redshift connection settings

Server

Connect by: ● Host ○ URL

URL: jdbc:redshift://data-bi-lab-redshift-cluster-3.cettexdsxw3v.eu-central-1.redshift.amazonaws.com:5439/dev

Host/Instance: data-bi-lab-redshift-cluster-3.cettexdsxw3v.eu-central-1.redshift.amazonaws.com Port: 5439

Database: dev

Authentication

Authentication: Database Native

Username: dilab_student9

Password: ●●●●●●●●●●●●●●● ☑ Save password locally

Advanced

Session role:

You can use variables in connection parameters.

Driver name: AWS / Redshift

Connection test

Connected (4685 ms)

Server: Redshift 8.0.2
PostgreSQL 8.0.2 on i686-pc-linux-gnu, compiled by GCC gcc (GCC) 3.4.2 20041017 (Red Hat 3.4.2-6.fc3), Redshift 1.0.54899

Driver: Redshift JDBC Driver 2.1.0.3

2. After logging in you should provision data from data lake to Redshift to be able to transform it using stored procedures. One way of doing it is by using COPY command. Use this command to provision data into the table created by you in your schema (user_dilab_student(1..32)). Use AWS documentation and example of syntax below to load the data. Attached role you can find using web console or CLI.

```
Example:
copy user_schema.lkp_smsc_special_sms (sender, id, active_dt, inactive_dt, created_by,
create_date, is_valid)
from 's3://aws-cdr-gen-data/lkp_smsc_special_sms.csv'
credentials
'aws_iam_role=arn:aws:iam::123456789102:role/rs-s3-role-read'
region 'eu-central-1'
delimiter ','
csv
DATEFORMAT AS 'MM-DD-YYYY'
IGNOREHEADER 1;
```

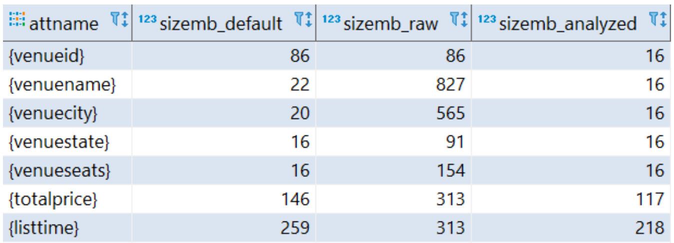
a. You could load all the tables, but because of limited resources please load only tables that are needed to create a report for your customer (at least 3 tables should be involved). It can be some tables that will be used in aggregations and calculations of KPIs. The business meaning of the report is up to you and your creativity.

b. After loading your tables check its initial compression types, distribution style, sort keys. Make a description of your analysis.

3. Take one of your *USER_DILAB_STUDENTN* schema tables.
   a. Identify compression types (encoding) of each column of this table *(YOUR_TABLE_defaultcomp)*
   b. Create table *YOUR_TABLE_withoutcomp* with similar to *YOUR_TABLE_defaultcomp* columns/data types, but without any compression applied and put there the same data as in the *YOUR_TABLE_defaultcomp* table.
   c. Use analyze command (on *YOUR_TABLE_defaultcomp or YOUR_TABLE_withoutcomp* table) to identify best compression methods suggested by Redshift. Create a table *YOUR_TABLE_analyzedcomp* (same columns but applying recommended encoding types and put same data as in the *YOUR_TABLE_defaultcomp* table there).

d. Compare size of tables *YOUR_TABLE_defaultcomp*, *YOUR_TABLE_withoutcomp* and *YOUR_TABLE_analyzedcomp*.

NOTE: Use system tables to see the difference in the size of the values by column name, make a description of your analysis. You can use this example as an expected result:

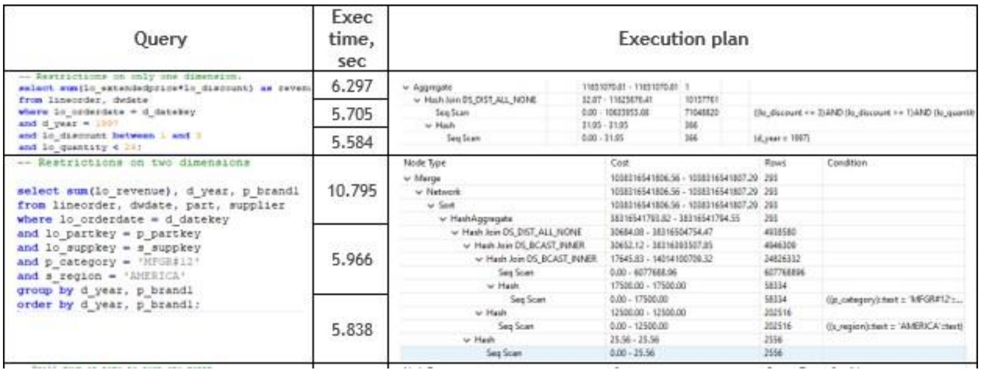


attname	sizemb_default	sizemb_raw	sizemb_analyzed
{venueid}	86	86	16
{venuename}	22	827	16
{venuecity}	20	565	16
{venuestate}	16	91	16
{venueseats}	16	154	16
{totalprice}	146	313	117
{listtime}	259	313	218

4. Prepare a stored procedure that will load your report. The main part of the stored procedure is the SELECT statement that should use joins of all your tables (at least 3 tables should be involved).

a. Prepare a stored procedure.

b. Describe the execution plan of this SELECT statement BEFORE optimization, log the time of query execution. E.g.:



<table>
<tr><th>Query</th><th>Exec time, sec</th><th>Execution plan</th></tr>
<tr><td rowspan="3"><pre>-- Restrictions on only one dimension.
select sum(lo_extendedprice*lo_discount) as reven
from lineorder, dwdate
where lo_orderdate = d_datekey
and d_year = 1997
and lo_discount between 1 and 3
and lo_quantity < 24;</pre></td><td>6.297</td><td rowspan="3">
<table>
<tr><td>Aggregate</td><td>11831070.81 - 11831070.81</td><td>1</td><td></td></tr>
<tr><td>Hash Join DS_DIST_ALL_NONE</td><td>32.07 - 11825876.41</td><td>10157761</td><td></td></tr>
<tr><td>Seq Scan</td><td>0.00 - 10633953.08</td><td>71048820</td><td>((lo_discount <= 3) AND (lo_discount >= 1) AND (lo_quantit</td></tr>
<tr><td>Hash</td><td>31.05 - 31.05</td><td>366</td><td></td></tr>
<tr><td>Seq Scan</td><td>0.00 - 31.05</td><td>366</td><td>(d_year = 1997)</td></tr>
</table></td></tr>
<tr><td>5.705</td></tr>
<tr><td>5.584</td></tr>
<tr><td rowspan="3"><pre>-- Restrictions on two dimensions

select sum(lo_revenue), d_year, p_brand1
from lineorder, dwdate, part, supplier
where lo_orderdate = d_datekey
and lo_partkey = p_partkey
and lo_suppkey = s_suppkey
and p_category = 'MFGR#12'
and s_region = 'AMERICA'
group by d_year, p_brand1
order by d_year, p_brand1;</pre></td><td>10.795</td><td rowspan="3">
<table>
<tr><th>Node Type</th><th>Cost</th><th>Rows</th><th>Condition</th></tr>
<tr><td>Merge</td><td>1008316541806.56 - 1008316541807.29</td><td>293</td><td></td></tr>
<tr><td>Network</td><td>1008316541806.56 - 1008316541807.29</td><td>293</td><td></td></tr>
<tr><td>Sort</td><td>1008316541806.56 - 1008316541807.29</td><td>293</td><td></td></tr>
<tr><td>HashAggregate</td><td>38316541793.82 - 38316541794.55</td><td>293</td><td></td></tr>
<tr><td>Hash Join DS_DIST_ALL_NONE</td><td>30684.08 - 38316504754.47</td><td>4938580</td><td></td></tr>
<tr><td>Hash Join DS_BCAST_INNER</td><td>30652.12 - 38316383507.05</td><td>4946308</td><td></td></tr>
<tr><td>Hash Join DS_BCAST_INNER</td><td>17645.83 - 14014100708.32</td><td>24826332</td><td></td></tr>
<tr><td>Seq Scan</td><td>0.00 - 6077688.96</td><td>607768896</td><td></td></tr>
<tr><td>Hash</td><td>17500.00 - 17500.00</td><td>58334</td><td></td></tr>
<tr><td>Seq Scan</td><td>0.00 - 17500.00</td><td>58334</td><td>((p_category)::text = 'MFGR#12'::...</td></tr>
<tr><td>Hash</td><td>12500.00 - 12500.00</td><td>202516</td><td></td></tr>
<tr><td>Seq Scan</td><td>0.00 - 12500.00</td><td>202516</td><td>((s_region)::text = 'AMERICA'::text)</td></tr>
<tr><td>Hash</td><td>25.56 - 25.56</td><td>2556</td><td></td></tr>
<tr><td>Seq Scan</td><td>0.00 - 25.56</td><td>2556</td><td></td></tr>
</table></td></tr>
<tr><td>5.966</td></tr>
<tr><td>5.838</td></tr>
</table>

NOTE: Please do not forget to turn off the result caching and do not consider the results of the first execution!

NOTE: Example of performance tests description and results you can follow in this and further tasks: https://dev.classmethod.jp/articles/redshift-sortkey-usecase-en/#section-04-03

c. Describe why the result of the first execution is not the best one to compare with.

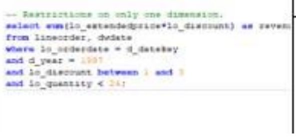
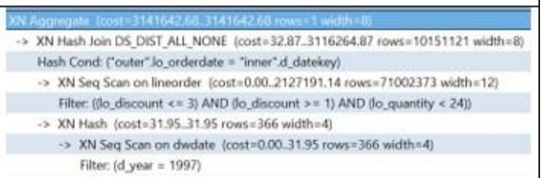
d. Describe the existing distribution style of tables, its sort keys.

5. Let's assume that these joins will be used very often and will not be massively shared with other tables in Redshift. Now you need to **optimize** your distribution style and sort keys.

You can find more information about proper Sort/DIST key for table creation here: https://docs.aws.amazon.com/redshift/latest/dg/t_Creating_tables.html

a. Describe why you took a specific distribution style and sort keys.

b. Describe the execution plan of this SELECT statement AFTER optimization, log the time of query execution. Compare it with BEFORE results. It is mandatory to describe changes in execution plans and how your optimization impacted it. E.g.:

Query	Prev exec time, sec	New exec time, sec	Optimized execution plan
 -- Restrictions on only one dimension. select sum(lo_extendedprice*lo_discount) as reven from lineorder, dwdate where lo_orderdate = d_datekey and d_year = 1997 and lo_discount between 1 and 3 and lo_quantity < 24;	6.297	6.924	 XN Aggregate (cost=3141642.68..3141642.68 rows=1 width=8) -> XN Hash Join DS_DIST_ALL_NONE (cost=32.87..3116264.87 rows=10151121 width=8) Hash Cond: ("outer".lo_orderdate = "inner".d_datekey) -> XN Seq Scan on lineorder (cost=0.00..2127191.14 rows=71002373 width=12) Filter: ((lo_discount <= 3) AND (lo_discount >= 1) AND (lo_quantity < 24)) -> XN Hash (cost=31.95..31.95 rows=366 width=4) -> XN Seq Scan on dwdate (cost=0.00..31.95 rows=366 width=4) Filter: (d_year = 1997)
	5.705	0.236	
	5.584	0.286	

6. Run the stored procedure using optimized tables and load data to your report.

7*. Connect Power BI to Redshift and make a small visual in Power BI Desktop using your report table.

COPY QUESTION

For this task one responsible student will be chosen. He will run statements below and run COPY statements. Execution time of both COPY commands should be written down and provisioned to all of the students. Each student should get this info and describe reasons of the resulting difference in execution.

1. Create tables lineorder_1 and lineorder_2 using following DDL statement:

```
CREATE TABLE lineorder
(
  lo_orderkey         INTEGER NOT NULL,
  lo_linenumber       INTEGER NOT NULL,
  lo_custkey          INTEGER NOT NULL,
  lo_partkey          INTEGER NOT NULL,
  lo_suppkey          INTEGER NOT NULL,
  lo_orderdate        INTEGER NOT NULL,
  lo_orderpriority    VARCHAR(15) NOT NULL,
  lo_shippriority     VARCHAR(1) NOT NULL,
  lo_quantity         INTEGER NOT NULL,
  lo_extendedprice    INTEGER NOT NULL,
  lo_ordertotalprice  INTEGER NOT NULL,
  lo_discount    INTEGER    NOT    NULL,
  lo_revenue          INTEGER NOT  NULL,
  lo_supplycost   INTEGER   NOT   NULL,
  lo_tax              INTEGER NOT NULL,
  lo_commitdate    INTEGER    NOT    NULL,
  lo_shipmode  VARCHAR(10) NOT NULL
);
```

2. Use COPY command to load data from s3://dilabbucketnovember2021/files/lineorder_file/ to lineorder_1 and log the time of the query execution. Then use COPY command to s3://dilabbucketnovember2021/files/lineorder_files/ to lineorder_2 and log the time of the query execution. Both files store exactly the same data.

3. Is there any difference in time of execution of these queries? Why?

WORKING WITH EXTERNAL TABLES

Some system data in the Amazon Redshift cluster are stored for about 7 days and then deleted. But you still need to monitor this data for longer term or make some analysis on top of it.

So, you need to create a stored procedure that will regularly load this data to a Redshift table that consumes compute nodes' disk space. But we cannot store everything in the Redshift cluster, because this data can be not so frequently used, will constantly grow in size and consume resources of the cluster.

Because of this, another procedure/procedures should UNLOAD this data for a specific period of time to S3 and delete this data from Redshift table (it will free a cluster disk space).

To be able to query this unloaded to S3 data and make joins with another data in the Redshift cluster we need to create external schema and external table that will point to files unloaded to S3.

This external table will be partitioned for not to scan all the data in S3. You can ask customer or assume most widely used filters to this data and create external partitions that will define folder structure for your data in S3 automatically.

Relax, you will not need to do it! It is just one example of use case.

To understand the concept of Redshift Spectrum functionality you need to:

1. Create external schema (user_dilab_student(1..32)_ext) pointing to your location and create several external tables on your files. More info can be found here:

   - https://docs.amazonaws.cn/en_us/redshift/latest/dg/c-getting-started-using-spectrum-create-external-table.html
   - https://docs.aws.amazon.com/redshift/latest/dg/c-spectrum-external-tables.html

   ```
   Example:
   CREATE EXTERNAL SCHEMA if not exists user_dilab_student1_ext
   FROM DATA catalog
   DATABASE ‘your_glue_database’
   IAM_ROLE 'arn:aws:iam:: 123456789102:role/rs-s3-role-read';
   ```

2. Partitioned external tables are extremely useful for the performance and cost cuts.

   a. Export any data which contain date column into S3 in a way, so each subfolder contains 1 month data (e.g., subfolder /your_date_column=2018-03-01 contains all records of the table where your_date_column is within March 2018).

   b. Create PARTITIONED external table “ext_studentN_partitioned” (partition by your_date_column).

   c. Verify data in partitioned external table (compare number of records per month to original table from which you prepared files or just with rows in the files). Prepare the test script that will show 0 difference (if it is not 0 - there is an issue).

   d. Examine query plan where you select from ext_studentN_partitioned with a WHERE clause containing your_date_column condition. Describe it.