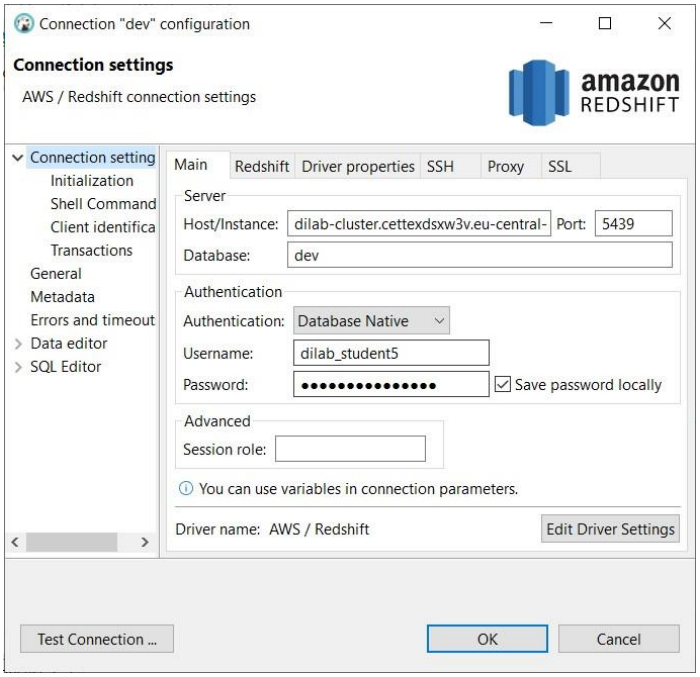
AWS

3 REPORT

CONTENTS

CREATING THE REPORT

LOGIN TO REDSHIFT



COPY COMMAND

A. Loading needed tables

```
CREATE SCHEMA user_dilab_student5;


------------------------CREATING CUSTOMERS IN RS FROM S3

CREATE TABLE user_dilab_student5.dim_customers
(   customer_surr_id     integer NOT NULL,
    source_system        VARCHAR(50) NOT NULL,
    source_entity        VARCHAR(50) NOT NULL,
    customer_src_id      VARCHAR(50) NOT NULL,
    customer_first_name  VARCHAR(50) NOT NULL,
    customer_last_name   VARCHAR(50) NOT NULL,
    customer_gender      VARCHAR(50) NOT NULL,
    customer_birthday    DATE NOT NULL,
    customer_phone       VARCHAR(50) NOT NULL,
    update_dt            DATE NOT NULL,
    insert_dt            DATE NOT NULL
);
Commit;

ALTER TABLE user_dilab_student5.dim_customers ADD CONSTRAINT dim_customers_pk PRIMARY KEY ( customer_surr_id );

copy user_dilab_student5.dim_customers (CUSTOMER_SURR_ID,SOURCE_SYSTEM,SOURCE_ENTITY,CUSTOMER_SRC_ID,CUSTOMER_F
from 's3://artiom-dolzhenko-aws-bucket/stores/bl_dm/dim_customers/dim_customers.scv'
credentials
'aws_iam_role=arn:aws:iam::260586643565:role/dilab-redshift-role'
region 'eu-central-1'
delimiter ','
csv
DATEFORMAT AS 'DD-MON-YY'
IGNOREHEADER 1;
Commit;
```

```
select  pg_table_def.schemaname, pg_table_def.tablename, pg_table_def.column, pg_table_def.type,
        pg_table_def.encoding, pg_table_def.distkey,pg_table_def.sortkey
from pg_table_def
where schemaname='user_dilab_student5'
and tablename ='dim_customers';
```

pg_table_def 1 × | Results 0 ×

select pg_table_def.schemaname, pg_table_def.tablename, pg_table_def.col | Enter a SQL expression to filter results (use Ctrl+Space)

	schemaname	tablename	column	type	encoding	distkey	sortkey
1	user_dilab_student5	dim_customers	customer_surr_id	integer	az64	[ ]	0
2	user_dilab_student5	dim_customers	source_system	character varying(50)	lzo	[ ]	0
3	user_dilab_student5	dim_customers	source_entity	character varying(50)	lzo	[ ]	0
4	user_dilab_student5	dim_customers	customer_src_id	character varying(50)	lzo	[ ]	0
5	user_dilab_student5	dim_customers	customer_first_name	character varying(50)	lzo	[ ]	0
6	user_dilab_student5	dim_customers	customer_last_name	character varying(50)	lzo	[ ]	0
7	user_dilab_student5	dim_customers	customer_gender	character varying(50)	lzo	[ ]	0
8	user_dilab_student5	dim_customers	customer_birthday	date	az64	[ ]	0
9	user_dilab_student5	dim_customers	customer_phone	character varying(50)	lzo	[ ]	0
10	user_dilab_student5	dim_customers	update_dt	date	az64	[ ]	0
11	user_dilab_student5	dim_customers	insert_dt	date	az64	[ ]	0

```
----------------------------------check distribution style---------------------------
------------------------------------------------------------------------------------
select "schema", "table", diststyle
from SVV_TABLE_INFO
where  "schema"='user_dilab_student5'
and "table" ='dim_customers';

CREATE TABLE user_dilab_student5.dim_dates (
    date_id                         date not null
```

Results 1 × | Results 0

select "schema", "table", diststyle from SVV_TABLE_INFO where "schema"='u: | Enter a SQL expression to filter results (use Ctrl+Space)

Grid

	schema	table	diststyle
1	user_dilab_student5	dim_customers	AUTO(EVEN)

B

```
-----------------------------------------------------------------------------------
------check compression types, distribution keys, sort keys--------------
-----------------------------------------------------------------------------------
select    pg_table_def.schemaname, pg_table_def.tablename, pg_table_def.column, pg_table_def.type,
          pg_table_def.encoding, pg_table_def.distkey,pg_table_def.sortkey
from pg_table_def where
schemaname='user_dilab_student5' and
tablename ='fct_sales';
```

pg_table_def 1

select pg_table_def.schemaname, pg_table_def.tablename, pg_table_def.col

Enter a SQL expression to filter results (use Ctrl+Space)

	schemaname	tablename	column	type	encoding	distkey	sortkey
1	user_dilab_student5	fct_sales	source_system	character varying(50)	lzo	[ ]	0
2	user_dilab_student5	fct_sales	source_entity	character varying(50)	lzo	[ ]	0
3	user_dilab_student5	fct_sales	product_surr_id	integer	az64	[ ]	0
4	user_dilab_student5	fct_sales	payment_type_surr_id	integer	az64	[ ]	0
5	user_dilab_student5	fct_sales	promotion_surr_id	integer	az64	[ ]	0
6	user_dilab_student5	fct_sales	store_surr_id	integer	az64	[ ]	0
7	user_dilab_student5	fct_sales	employee_surr_id	integer	az64	[ ]	0
8	user_dilab_student5	fct_sales	customer_surr_id	integer	az64	[ ]	0
9	user_dilab_student5	fct_sales	date_id	date	az64	[ ]	0
10	user_dilab_student5	fct_sales	unit_cost	integer	az64	[ ]	0
11	user_dilab_student5	fct_sales	unit_price	integer	az64	[ ]	0
12	user_dilab_student5	fct_sales	sales_quantity	integer	az64	[ ]	0

```
select "schema", "table", diststyle
from SVV_TABLE_INFO
where  "schema"='user_dilab_student5'
and "table" ='fct_sales';
```

sults 1

lect "schema", "table", diststyle from SVV_TABLE_INFO where "schem

schema	table	diststyle
user_dilab_student5	fct_sales	AUTO(EVEN)

As seen before, redshift didn’t provide any sort keys or dist keys.
distribution style for lange tables is “EVEN”, while small tables have “ALL”

A. Identify compression types I decided to take dim_customers.

```
select  pg.schemaname,
        pg.tablename,
        pg.column,
        pg.type,
        pg.encoding
from pg_table_def pg
where schemaname='user_dilab_student5'
and tablename ='dim_customers';
```

pg_table_def 1

select pg.schemaname, pg.tablename, pg.column, pg.type, pg.encoding fron

	schemaname	tablename	column	type	encoding
1	user_dilab_student5	dim_customers	customer_surr_id	integer	az64
2	user_dilab_student5	dim_customers	source_system	character varying(50)	lzo
3	user_dilab_student5	dim_customers	source_entity	character varying(50)	lzo
4	user_dilab_student5	dim_customers	customer_src_id	character varying(50)	lzo
5	user_dilab_student5	dim_customers	customer_first_name	character varying(50)	lzo
6	user_dilab_student5	dim_customers	customer_last_name	character varying(50)	lzo
7	user_dilab_student5	dim_customers	customer_gender	character varying(50)	lzo
8	user_dilab_student5	dim_customers	customer_birthday	date	az64
9	user_dilab_student5	dim_customers	customer_phone	character varying(50)	lzo
10	user_dilab_student5	dim_customers	update_dt	date	az64
11	user_dilab_student5	dim_customers	insert_dt	date	az64

Checking data in no_comp table

```
select *
from user_dilab_student5.dim_customers_no_comp
limit 20;
```

customers_no_comp 1

ct * from user_dilab_student5.dim_customers_no_comp limit 20

customer_surr_id	source_system	source_entity	customer_src_id	customer_first_name	customer_last_name	customer_gender
192	3nf	ce_CUSTOMERS	40425	Dalia	Hamilton	Male
208	3nf	ce_CUSTOMERS	23614	Rachael	Lloyd	Female
224	3nf	ce_CUSTOMERS	64530	Nick	Weston	Male
240	3nf	ce_CUSTOMERS	82979	Rufus	Wilson	Male
256	3nf	ce_CUSTOMERS	80988	Martin	Porter	Female
272	3nf	ce_CUSTOMERS	42111	Parker	Slater	Male
288	3nf	ce_CUSTOMERS	18301	Amy	Tailor	Male
304	3nf	ce_CUSTOMERS	17763	Adalie	Cobb	Female
320	3nf	ce_CUSTOMERS	42368	Hadley	Yates	Female
336	3nf	ce_CUSTOMERS	18693	Molly	Norton	Female
352	3nf	ce_CUSTOMERS	52006	Cara	Dale	Male
368	3nf	ce_CUSTOMERS	40103	Henry	Faulkner	Female

A. Check compression types for no_comp

```
select  pg.schemaname,
        pg.tablename,
        pg.column,
        pg.type,
        pg.encoding
from pg_table_def pg
where schemaname='user_dilab_student5'
and tablename ='dim_customers_no_comp';
```

g_table_def 1

elect pg.schemaname, pg.tablename, pg.column, pg.type, pg.encoding fron | Enter a SQL expression to filter results (us

	schemaname	tablename	column	type	encoding
1	user_dilab_student5	dim_customers_no_comp	customer_surr_id	integer	none
2	user_dilab_student5	dim_customers_no_comp	source_system	character varying(256)	none
3	user_dilab_student5	dim_customers_no_comp	source_entity	character varying(256)	none
4	user_dilab_student5	dim_customers_no_comp	customer_src_id	character varying(256)	none
5	user_dilab_student5	dim_customers_no_comp	customer_first_name	character varying(256)	none
6	user_dilab_student5	dim_customers_no_comp	customer_last_name	character varying(256)	none
7	user_dilab_student5	dim_customers_no_comp	customer_gender	character varying(256)	none
8	user_dilab_student5	dim_customers_no_comp	customer_birthday	date	none
9	user_dilab_student5	dim_customers_no_comp	customer_phone	character varying(256)	none
10	user_dilab_student5	dim_customers_no_comp	update_dt	date	none
11	user_dilab_student5	dim_customers_no_comp	insert_dt	date	none

Use analyze command

Analyze compression user_dilab_student5.dim_customers_no_comp comprows 87662;

B Compare size o tables with default compression, without compression and analyzed

```
select  dts.column_name,
        dts."size" as default_size,
        wits."size" as without_comp_size,
        atz."size" as analyzed_comp_size
from default_table_size dts
inner join withoutcomp_table_size wits
on dts.column_name=wits.column_name
inner join analyzedcomp_table_size atz
on dts.column_name=atz.column_name
order by dts.column_name;
```

Results 1 | Results 0 | Results 0 | Results 0 | Results 2

select dts.column_name, dts."size" as default_size, wits."size" as without_c

	column_name	default_size	without_comp_size	analyzed_comp_size
1	customer_birthday	4	4	4
2	customer_first_name	4	4	4
3	customer_gender	4	4	4
4	customer_last_name	4	4	4
5	customer_phone	4	4	4
6	customer_src_id	4	4	4
7	customer_surr_id	4	4	4
8	insert_dt	4	4	4
9	source_entity	4	4	4
10	source_system	4	4	4
11	update_dt	4	4	4

Redshift had chosen optimal types of compression

TASK 4. Work with stored procedure

Turning cach of
ALTER USER dilab_student5 SET enable_result_cache_for_session TO off;
commit;

```
select  "schema",
        "table",
        diststyle,
        sortkey1
from SVV_TABLE_INFO
where  "schema"='user_dilab_student5'
and ("table" ='fct_sales'
or "table" ='dim_dates'
or "table" ='dim_payment_types'
or "table" ='dim_stores'
or "table" ='dim_customers'
or "table" ='dim_employees');
```

lts 1 | Results 0 | Results 0 | Results 0 | Results 2

ct "schema", "table", diststyle, sortkey1 from SVV_TABLE_INFO where

schema	table	diststyle	sortkey1
user_dilab_student5	dim_dates	AUTO(ALL)	AUTO(SORTKEY)
user_dilab_student5	dim_employees	AUTO(ALL)	AUTO(SORTKEY)
user_dilab_student5	fct_sales	AUTO(EVEN)	AUTO(SORTKEY)
user_dilab_student5	dim_customers	AUTO(EVEN)	AUTO(SORTKEY)
user_dilab_student5	dim_stores	AUTO(ALL)	AUTO(SORTKEY)
user_dilab_student5	dim_payment_types	AUTO(ALL)	AUTO(SORTKEY)

Then i altered tables to set sortkeys

```
-------------------------------------------------------------
-------------alter diststyle and sort keys---------------------
-------------------------------------------------------------
alter table user_dilab_student5.fct_sales alter sortkey none;
alter table user_dilab_student5.dim_dates alter sortkey none;
alter table user_dilab_student5.dim_stores alter sortkey none;
alter table user_dilab_student5.dim_customers alter sortkey none;
alter table user_dilab_student5.dim_employees alter sortkey none;
alter table user_dilab_student5.dim_payment_types alter sortkey none;

alter table user_dilab_student5.dim_payment_types alter diststyle  even;
alter table user_dilab_student5.fct_sales alter diststyle  even;
alter table user_dilab_student5.dim_dates alter diststyle all;
alter table user_dilab_student5.dim_stores alter diststyle all;
alter table user_dilab_student5.dim_customers alter diststyle  even;
alter table user_dilab_student5.dim_employees alter diststyle  even;

commit;
```

```
-----------------------------------------------
-----------check diststyle and sort keys-------
-----------------------------------------------
select  "schema",
        "table",
        diststyle,
        sortkey1
from SVV_TABLE_INFO
where  "schema"='user_dilab_student5'
and ("table" ='fct_sales'
or "table" ='dim_dates'
or "table" ='dim_payment_types'
or "table" ='dim_stores'
or "table" ='dim_customers'
or "table" ='dim_employees');
```

esults 1 | Results 0 | Results 0 | Results 0 | Results 2

elect "schema", "table", diststyle, sortkey1 from SVV_TABLE_INFO where '

	schema	table	diststyle	sortkey1
1	user_dilab_student5	dim_stores	ALL	AUTO(SORTKEY)
2	user_dilab_student5	dim_payment_types	EVEN	AUTO(SORTKEY)
3	user_dilab_student5	dim_dates	ALL	AUTO(SORTKEY)
4	user_dilab_student5	dim_employees	EVEN	AUTO(SORTKEY)
5	user_dilab_student5	fct_sales	EVEN	AUTO(SORTKEY)
6	user_dilab_student5	dim_customers	EVEN	AUTO(SORTKEY)

```
explain
select distinct ds.store_name, f.date_id,
dpt.payment_type,
sum (f.sales_quantity) as quantity_sold,
sum (f.unit_price) as price_per_unit,
sum (f.unit_price* f.sales_quantity)  as total_revenue
from user_dilab_student5.fct_sales f
left join user_dilab_student5.dim_payment_types dpt  on dpt.payment_type_surr_id = f.paymen
left join user_dilab_student5.dim_stores ds  on f.store_surr_id = ds.store_surr_id
where  date_id >= '2022-03-17'
group by f.date_id, f.product_surr_id, dpt.payment_type, ds.store_name
order by total_revenue desc
limit 20;
```

_stores(+) 1

ect distinct ds.store_name, f.date_id, dpt.payment_type, sum (f.sales_qu

	store_name	date_id	payment_type	quantity_sold	price_per_unit	total_revenue
	Prayons	2022-03-17	CASH	6,543	575	32,715
	Trafor	2022-03-17	CASH	5,997	625	29,985
	Nime	2022-03-17	CASH	5,672	550	28,360
	Solaplay	2022-03-17	CASH	5,639	525	28,195
	Bookoread	2022-03-17	CASH	5,636	525	28,180
	Vrigo	2022-03-17	CASH	5,475	450	27,375
	Man Mobile	2022-03-17	CASH	5,339	525	26,695
	DeskGet	2022-03-17	CASH	5,235	475	26,175
	Shoplist	2022-03-17	CASH	5,234	475	26,170
	Shoplist	2022-03-17	CASH	5,179	475	25,895
	ReadABook	2022-03-17	CASH	5,148	500	25,740
	Kulop	2022-03-17	CASH	5,064	500	25,320
	DeskGet	2022-03-17	CASH	5,040	425	25,200
	Koob	2022-03-17	CASH	4,991	450	24,955

Save Cancel Script 50 20 Rows: 1 20 row(s) fetched

```
QUERY PLAN
XN Merge  (cost=1000000497963.09..1000000498248.18 rows=114036 width=33)
  Merge Key: sum((f.sales_quantity * f.unit_price))
  -> XN Network  (cost=1000000497963.09..1000000498248.18 rows=114036 width=33)
     Send to leader
     -> XN Sort  (cost=1000000497963.09..1000000498248.18 rows=114036 width=33)
        Sort Key: sum((f.sales_quantity * f.unit_price))
        -> XN Unique  (cost=485533.66..488384.56 rows=114036 width=33)
           -> XN HashAggregate  (cost=485533.66..486674.02 rows=114036 width=33)
              -> XN Hash Left Join DS_DIST_ALL_NONE  (cost=0.66..467687.20 rows=1019798 width=33)
                 Hash Cond: ("outer".store_surr_id = "inner".store_surr_id)
                 -> XN Hash Left Join DS_BCAST_INNER  (cost=0.03..434993.06 rows=999801 width=27)
                    Hash Cond: ("outer".payment_type_surr_id = "inner".payment_type_surr_id)
                    -> XN Seq Scan on fct_sales f  (cost=0.00..12497.51 rows=999801 width=24)
                       Filter: (date_id >= '2022-03-17'::date)
                    -> XN Hash  (cost=0.02..0.02 rows=2 width=11)
                       -> XN Seq Scan on dim_payment_types dpt  (cost=0.00..0.02 rows=2 width=11)
                 -> XN Hash  (cost=0.51..0.51 rows=51 width=14)
                    -> XN Seq Scan on dim_stores ds  (cost=0.00..0.51 rows=51 width=14)
```

Runs	Time	Cost
1	43	1000000497963.09..1000000498248.18
2	31	1000000497963.09..1000000498248.18
3	32	1000000497963.09..1000000498248.18

TASK 5. Work with optimization of distribution style and sort keys.

After that I decided to add sorkeys to fct_sales and dim stores as seen below.

```
alter table user_dilab_student5.fct_sales alter sortkey (unit_price,sales_quantity);
alter table user_dilab_student5.dim_stores  alter sortkey (store_name);
```

Runs	Time	Cost
1	40	1000000497963.09..1000000497223.22
2	29	1000000497963.09..1000000497223.22
3	27	1000000497963.09..1000000497223.22

Extremely small improvement can be seen.

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

```
as $$
declare
integer_val integer:=0;
begin
raise info 'Procedure update_report started';
insert into user_dilab_student5.report (store_name,
                                        date_id,
                                        payment_type,
                                        quantity_sold,
                                        price_per_unit,
                                        total_revenue)
select distinct ds.store_name,
                f.date_id,
                dpt.payment_type,
                sum (f.sales_quantity) as quantity_sold,
                sum (f.unit_price) as price_per_unit,
                sum (f.unit_price* f.sales_quantity)  as total_revenue
from user_dilab_student5.fct_sales f
left join user_dilab_student5.dim_payment_types dpt
on dpt.payment_type_surr_id = f.payment_type_surr_id
left join user_dilab_student5.dim_stores ds
on f.store_surr_id = ds.store_surr_id
where  date_id >= '2022-03-17'
group by    f.date_id,
            f.product_surr_id,
            dpt.payment_type,
            ds.store_name
order by total_revenue desc
get diagnostics integer_val := row_count;
raise notice 'Table report inserted, rows: %',integer_val ;
exception
when others then raise exception 'errow message: %, error code: %',sqlerrm, sqlstate;
end;
$$ language plpgsql;
```

```
select *
from user_dilab_student5.report;
```

report 1

select * from user_dilab_student5.report | Enter a SQL expression to filter results (use Ctrl+Space)

	store_name	date_id	payment_type	quantity_sold	price_per_unit	total_revenue
1	Nime	2022-03-17	CASH	5,672	550	28,360
2	Man Mobile	2022-03-17	CASH	5,339	525	26,695
3	ReadABook	2022-03-17	CASH	5,148	500	25,740
4	Ingocal	2022-03-17	CASH	4,904	475	24,520
5	Caint	2022-03-17	CASH	4,804	475	24,020
6	MemoMe	2022-03-17	CASH	4,683	450	23,415
7	Paize	2022-03-17	CASH	4,667	450	23,335
8	BeReady	2022-03-17	CASH	4,625	425	23,125
9	Koob	2022-03-17	CASH	4,579	425	22,895

COPY QUESTION

Created tables lineorder_1 and lineorder_2

```sql
CREATE TABLE lineorder_1
(
 lo_orderkey bigint NOT null ,
 lo_linenumber bigint NOT NULL,
 lo_custkey bigint NOT NULL,
 lo_partkey bigint NOT NULL,
 lo_suppkey bigint NOT NULL,
 lo_orderdate bigint NOT NULL,
 lo_orderpriority VARCHAR(20) NOT NULL,
 lo_shippriority VARCHAR(5) NOT NULL,
 lo_quantity bigint NOT NULL,
 lo_extendedprice bigint NOT NULL,
 lo_ordertotalprice bigint NOT NULL,
 lo_discount bigint NOT NULL,
 lo_revenue bigint NOT NULL,
 lo_supplycost bigint NOT NULL,
 lo_tax bigint NOT NULL,
 lo_commitdate bigint NOT NULL,
 lo_shipmode VARCHAR(15) NOT NULL
);
CREATE TABLE lineorder_2
(
 lo_orderkey bigint NOT null ,
 lo_linenumber bigint NOT NULL,
 lo_custkey bigint NOT NULL,
 lo_partkey bigint NOT NULL,
 lo_suppkey bigint NOT NULL,
 lo_orderdate bigint NOT NULL,
 lo_orderpriority VARCHAR(20) NOT NULL,
 lo_shippriority VARCHAR(5) NOT NULL,
 lo_quantity bigint NOT NULL,
 lo_extendedprice bigint NOT NULL,
 lo_ordertotalprice bigint NOT NULL,
 lo_discount bigint NOT NULL,
 lo_revenue bigint NOT NULL,
 lo_supplycost bigint NOT NULL,
 lo_tax bigint NOT NULL,
 lo_commitdate bigint NOT NULL,
 lo_shipmode VARCHAR(15) NOT NULL
);
```

2. Loading data to tables

```
copy lineorder_1
from 's3://dilabbucket/files/lineorder_file/'
credentials
'aws_iam_role=arn:aws:iam::260586643565:role/dilab-redshift-role'
region 'eu-central-1'
delimiter ','
gzip
DATEFORMAT AS 'auto'
IGNOREHEADER 1;

select * from stl_load_errors;

copy lineorder_2
from 's3://dilabbucket/files/lineorders/'
credentials
'aws_iam_role=arn:aws:iam::260586643565:role/dilab-redshift-role'
format as parquet;
```

```
insert into temp_table_times  (querytxt, starttime, endtime,diff)
    select querytxt, starttime, endtime,datediff(millisecond,starttime,endtime)
    from stl_query where userid=109
    and querytxt like 'copy lineorder_1%'
    order by starttime desc limit 20;
insert into temp_table_times  (querytxt, starttime, endtime,diff)
    select querytxt, starttime, endtime,datediff(millisecond,starttime,endtime)
    from stl_query where userid=109
    and querytxt like 'copy lineorder_2%'
    order by starttime desc limit 20;
```

querytxt	starttime	endtime	diff
copy lineorder_1 from 's3://dilabbucket/files/lineorder_file/' credentials '' region 'eu-central-1' delimiter ',' gzip DATEFORMAT	2022-04-03 10:30:40.091	2022-04-03 10:35:14.899	274,808
copy lineorder_2 from 's3://dilabbucket/files/lineorders/' credentials '' format as parquet	2022-04-03 10:27:03.537	2022-04-03 10:29:03.220	119,683

Type of file	Number of files	Size of files	Load time, s	Count of rows
gz	1	3 Gb	274,808	100.000.008
parquet	4	2.22 Gb	119,683	100.000.008

The reason why the first query works longer is that it reads compressed data. But i was surprised that while file was compressed by 25%, the actual read time is 2 times longer.

EXTERNAL TABLES

1.

```
-----------------------------------------------------------------------------
-------------------------create external schema------------------------------
-----------------------------------------------------------------------------
create external schema if not exists user_dilab_student5_ext
from data catalog
database 'artiom_dolzhenko_database'
IAM_ROLE 'arn:aws:iam::260586643565:role/dilab-redshift-role'
commit;

-----------------------------------------------------------------------------
------------------------check external schema-------------------------------
-----------------------------------------------------------------------------
select * from pg_catalog.svv_external_schemas
where schemaname ='user_dilab_student5_ext';
```

```
-----------------------------------------------------------------------------
------------------------------upload data to s3----------------------------
-----------------------------------------------------------------------------
unload ('Select * from fct_sales
            Where extract (year from date_id)=2022
            and extract (MONTH from date_id)=1;')
to 's3://artiom-dolzhenko-aws-bucket/stores/bl_dm/sales_2022-01/'
iam_role 'arn:aws:iam::260586643565:role/dilab-redshift-role';
unload ('Select * from fct_sales
            Where extract (year from date_id)=2022
            and extract (MONTH from date_id)=2;')
to 's3://artiom-dolzhenko-aws-bucket/stores/bl_dm/sales_2022-02/'
iam_role 'arn:aws:iam::260586643565:role/dilab-redshift-role';
unload ('Select * from fct_sales
            Where extract (year from date_id)=2022
            and extract (MONTH from date_id)=3')
to 's3://artiom-dolzhenko-aws-bucket/stores/bl_dm/sales_2022-03/'
iam_role 'arn:aws:iam::260586643565:role/dilab-redshift-role';
```

```
-----------------------------------------------------------------------------
------------------------------create external table-------------------------
-----------------------------------------------------------------------------
CREATE external TABLE  user_dilab_student5_ext.FCT_SALES_EXT_PART
   (source_system        VARCHAR(50) ,
    source_entity        VARCHAR(50) ,
    product_surr_id      INTEGER ,
    payment_type_surr_id INTEGER ,
    promotion_surr_id    INTEGER ,
    store_surr_id        INTEGER ,
    employee_surr_id     INTEGER ,
    customer_surr_id     INTEGER ,
    date_id              DATE ,
    unit_cost            INTEGER,
    unit_price           INTEGER,
    sales_quantity       INTEGER,
    update_dt            DATE ,
    insert_dt            DATE
   )
partitioned by (DATE_ID DATE)
row format delimited
fields terminated by '|'
stored as textfile
location 's3://artiom-dolzhenko-aws-bucket/stores/bl_dm/fct_sales/'
table properties ('numRows'='999801');
```