

U2M11.LW.ETL Overview - Advanced Refresh Scenarios

Shkrabatouskaya Vera

https://github.com/VeraShkrabatouskaya/DataMola_Data-Camping-2022

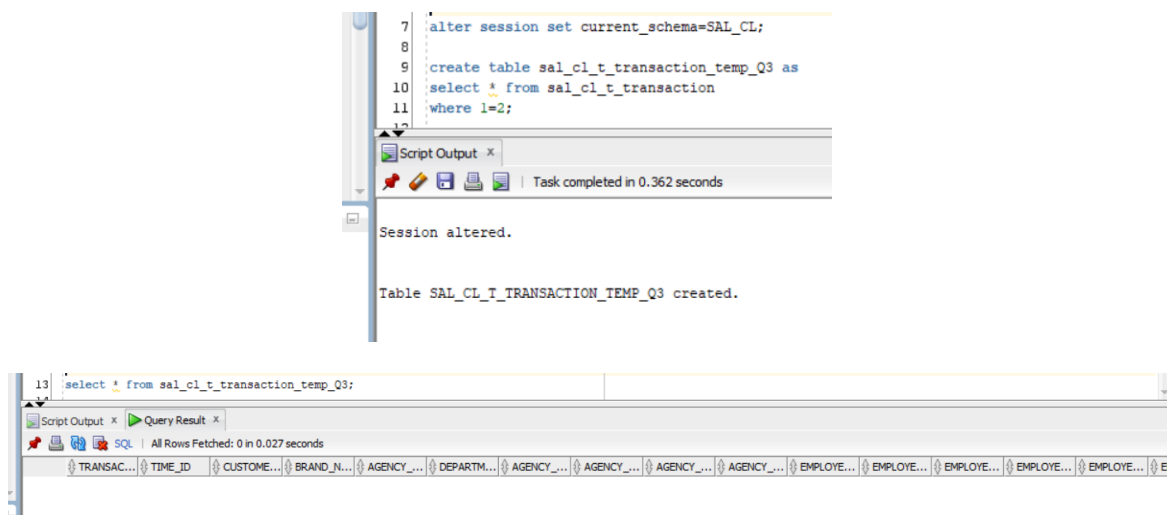
2. ETL Advanced Refresh Scenarios – Refactoring Load to SAL

Task 01 is common for LabWork 10 (Task 02), 11(Task 01).

2.1. Task 01: Loading to SAL Layer Data

Let's create a table containing only Q3 data using Exchange Partition and Temporary table.

Creating a temporary table sal_cl_t_transaction_temp_Q3.



```
7 alter session set current_schema=SAL_CL;
8
9 create table sal_cl_t_transaction_temp_Q3 as
10 select * from sal_cl_t_transaction
11 where l=2;
```

Script Output x | Task completed in 0.362 seconds

Session altered.

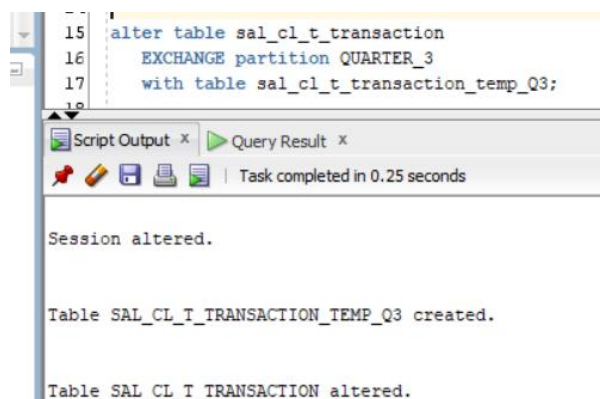
Table SAL_CL_T_TRANSACTION_TEMP_Q3 created.

```
13 select * from sal_cl_t_transaction_temp_Q3;
```

Script Output x | Query Result x | All Rows Fetched: 0 in 0.027 seconds

TRANSACTION_ID	TIME_ID	CUSTOMER_ID	BRAND_NAME	AGENCY_CODE	DEPARTMENT_CODE	AGENCY_CODE	AGENCY_CODE	AGENCY_CODE	AGENCY_CODE	EMPLOYEE_CODE	EMPLOYEE_CODE	EMPLOYEE_CODE	EMPLOYEE_CODE	EMPLOYEE_CODE
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Let's migrate the data from the sal_cl_t_transaction table for Q3 using the Exchange Partition to a temporary table.



```
15 alter table sal_cl_t_transaction
16 EXCHANGE partition QUARTER_3
17 with table sal_cl_t_transaction_temp_Q3;
```

Script Output x | Query Result x | Task completed in 0.25 seconds

Session altered.

Table SAL_CL_T_TRANSACTION_TEMP_Q3 created.

Table SAL_CL_T_TRANSACTION altered.

19 `select * from sal_cl_t_transaction_temp_Q3;`

Script Output x Query Result x

SQL | Fetched 50 rows in 0.032 seconds

	TRANSACTION_ID	TIME_ID	CUSTOMER_NAME	BRAND_NAME	AGENCY_NAME	DEPARTMENT_NAME	AGENCY_CITY	AGENCY_COUNTRY	AGENCY_FEE_PERCENT
1	31434	19-JUL-22	Samsung Electronics	SAMSUNG	Starcom	Media Planning and Buying Hong Kong	China		5
2	31435	06-JUL-22	Visa International Service Association	VISA	Starcom	Media Planning and Buying Warsaw	Poland		8
3	31436	12-JUL-22	Samsung Electronics	SAMSUNG	Starcom	Out-of-Home Media Milan	Italy		2
4	31437	04-JUL-22	Samsung Electronics	SAMSUNG	Starcom	Art and Visualization Barcelona	Spain		9
5	31438	06-JUL-22	Samsung Electronics	SAMSUNG	Starcom	Media Planning and Buying Barcelona	Spain		9
6	31439	07-JUL-22	Samsung Electronics	SAMSUNG	Starcom	Media Planning and Buying Hong Kong	China		5
7	31440	04-JUL-22	Visa International Service Association	VISA	Starcom	Media Planning and Buying Hong Kong	China		5
8	31441	13-JUL-22	Visa International Service Association	VISA	Starcom	Media Planning and Buying Berlin	Germany		3
9	31442	02-JUL-22	Samsung Electronics	SAMSUNG	Starcom	Media Planning and Buying Barcelona	Spain		9
10	31443	08-JUL-22	Visa International Service Association	VISA	Starcom	Media Planning and Buying New York	United States		5
11	31444	06-JUL-22	Samsung Electronics	SAMSUNG	Starcom	Media Planning and Buying Dubai	UAE		4
12	31445	08-JUL-22	Visa International Service Association	VISA	Starcom	Media Planning and Buying Milan	Italy		2
13	31446	11-JUL-22	Samsung Electronics	SAMSUNG	Starcom	Media Planning and Buying Berlin	Germany		3
14	31447	10-JUL-22	Visa International Service Association	VISA	Starcom	Media Planning and Buying Berlin	Germany		3
15	31448	06-JUL-22	Visa International Service Association	VISA	Starcom	Media Planning and Buying New York	United States		5

We see that the data for Q3 has been moved to a temporary table. At the same time, in our main table Partition QUARTER_3 has become empty.

34 `SELECT * FROM SAL_CL_T_TRANSACTION partition (QUARTER_3);`

Script Output x Query Result x

SQL | All Rows Fetched: 0 in 0.024 seconds

	TRANSACTION_ID	TIME_ID	CUSTOMER_NAME	BRAND_NAME	AGENCY_NAME	DEPARTMENT_NAME	AGENCY_CITY	AGENCY_COUNTRY	AGENCY_FEE_PERCENT	EMPLOYEE_ID	EMPLOYEE_NAME	EMPLOYEE_SALARY	EMPLOYEE_PROJECT_GROSS	EMPLOYEE_PROJECT_NET
--	----------------	---------	---------------	------------	-------------	-----------------	-------------	----------------	--------------------	-------------	---------------	-----------------	------------------------	----------------------

3. Business Task – Performance of STAR Scheme

3.1. Task 02: Prepare Report Layout

Let's create a select to calculate employee salaries by month using the STAR schema.

Welcome Page x VeraDB x Lab11_3.1.Task 02.sql

SQL Worksheet: History

Worksheet Query Builder

```

3 alter session set current_schema=DW_DATA;
4
5 select
6   TRUNC(TIME_ID, 'YYYY') AS YEAR,
7   TRUNC(TIME_ID, 'MM') AS MONTH,
8   SUM(gross_salary_employee_dollar_amount) / 'as employee_salary_project_GROSS',
9   SUM(net_salary_employee_dollar_amount) / 'as employee_salary_project_NET'
10  from DW_DATA.FCT_BUSINESS
11  GROUP BY CUBE (TRUNC(TIME_ID, 'YYYY'), TRUNC(TIME_ID, 'MM'))
12  HAVING
13    (TRUNC(TIME_ID, 'YYYY') IS NOT NULL
14     AND
15     TRUNC(TIME_ID, 'MM') IS NOT NULL)
16  order by TRUNC(TIME_ID, 'YYYY'), TRUNC(TIME_ID, 'MM');
17

```

Script Output x Query Result x

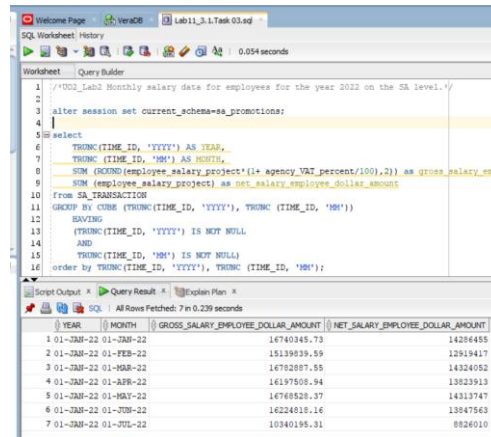
SQL | All Rows Fetched: 7 in 0.089 seconds

	YEAR	MONTH	SUM(GROSS_SALARY_EMPLOYEE_DOLLAR_AMOUNT) / 'ASEMPLOYEE_SALARY_PROJECT_GROSS'	SUM(NET_SALARY_EMPLOYEE_DOLLAR_AMOUNT) / 'ASEMPLOYEE_SALARY_PROJECT_NET'
1	01-JAN-22	01-JAN-22	16740345.73	14286455
2	01-JAN-22	01-FEB-22	15139839.59	12919417
3	01-JAN-22	01-MAR-22	16782887.55	14324052
4	01-JAN-22	01-APR-22	16197508.94	13823913
5	01-JAN-22	01-MAY-22	16768528.37	14313747
6	01-JAN-22	01-JUN-22	16224818.16	13847563
7	01-JAN-22	01-JUL-22	10340195.31	8826010

3.2. Task 03: Compare Report Layout Performance

Create summarize table with comparison performance of the monthly reports at different levels.

- Monthly salary data for employees for the year 2022 on the SA level.

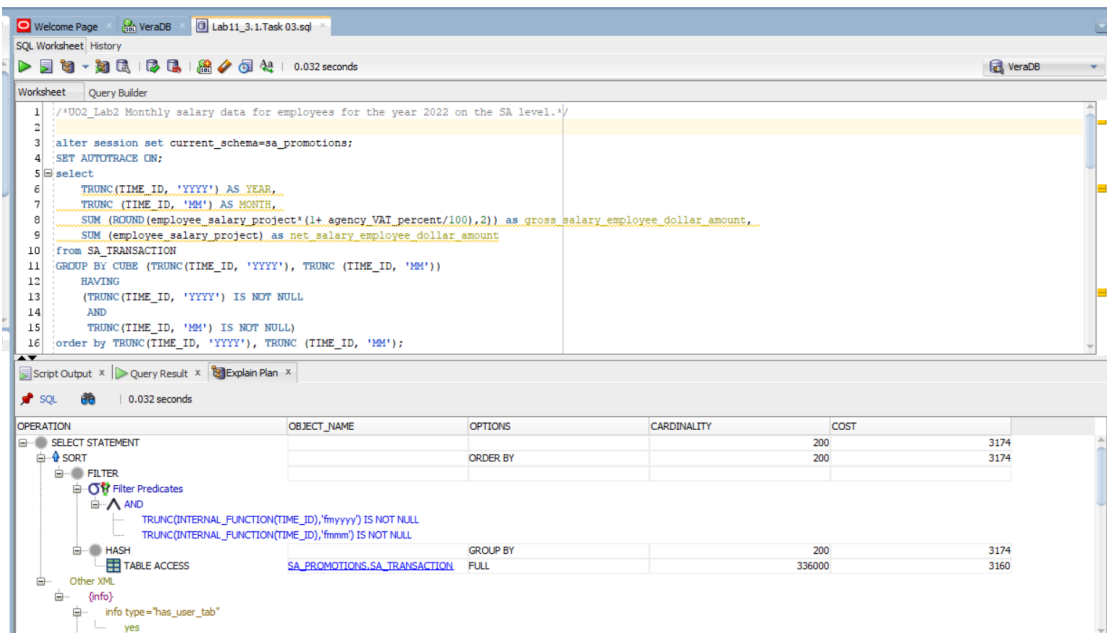


The screenshot shows the SQL Developer interface with a query window titled 'Lab11_3.1.Task 03.sql'. The query is as follows:

```
1 /*002_Lab2 Monthly salary data for employees for the year 2022 on the SA level.*/
2
3 alter session set current_schema=sa_promotions;
4
5 select
6   TRUNC(TIME_ID, 'YYYY') AS YEAR,
7   TRUNC(TIME_ID, 'MM') AS MONTH,
8   SUM (ROUND(employee_salary_project*(1+ agency_VAT_percent/100),2)) as gross_salary_employee_dollar_amount,
9   SUM (employee_salary_project) as net_salary_employee_dollar_amount
10  from SA_TRANSACTION
11  GROUP BY CUBE (TRUNC(TIME_ID, 'YYYY'), TRUNC(TIME_ID, 'MM'))
12  HAVING
13   (TRUNC(TIME_ID, 'YYYY') IS NOT NULL
14   AND
15   TRUNC(TIME_ID, 'MM') IS NOT NULL)
16  order by TRUNC(TIME_ID, 'YYYY'), TRUNC(TIME_ID, 'MM');
```

The results are displayed in a table with the following columns: YEAR, MONTH, GROSS_SALARY_EMPLOYEE_DOLLAR_AMOUNT, and NET_SALARY_EMPLOYEE_DOLLAR_AMOUNT. The data shows monthly salary information for the year 2022.

YEAR	MONTH	GROSS_SALARY_EMPLOYEE_DOLLAR_AMOUNT	NET_SALARY_EMPLOYEE_DOLLAR_AMOUNT
01-JAN-22	01-JAN-22	16740345.73	14286455
02-JAN-22	01-FEB-22	15139839.55	12919417
03-JAN-22	01-MAR-22	16702897.55	14324052
04-JAN-22	01-APR-22	14197505.94	13523513
05-JAN-22	01-MAY-22	14768528.37	14313747
06-JAN-22	01-JUN-22	16224818.16	13847563
07-JAN-22	01-JUL-22	10340195.31	8826010

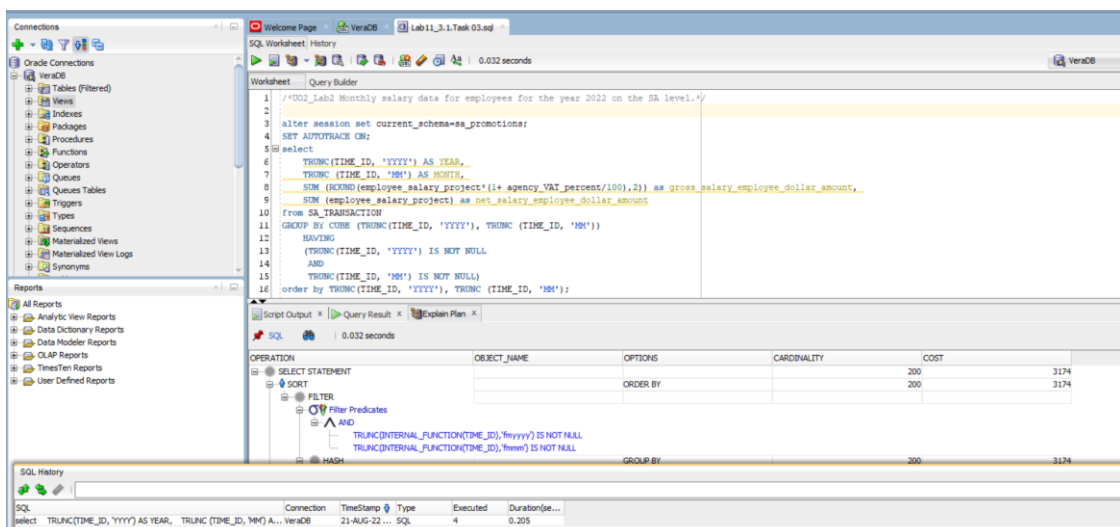


The screenshot shows the SQL Developer interface with the same query as above. The execution plan is displayed below the query window. The plan shows the following operations:

- SELECT STATEMENT
- SORT
- FILTER
- AND
- TRUNC(INTERNAL_FUNCTION(TIME_ID), 'YYYY') IS NOT NULL
- TRUNC(INTERNAL_FUNCTION(TIME_ID), 'MM') IS NOT NULL
- HASH
- TABLE ACCESS
- SA_PROMOTIONS.SA_TRANSACTION
- GROUP BY
- FULL

The execution plan also includes a table with the following columns: OPERATION, OBJECT_NAME, OPTIONS, CARDINALITY, and COST. The data shows the cost and cardinality for each operation in the plan.

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			200	3174
SORT		ORDER BY	200	3174
FILTER				
AND				
TRUNC(INTERNAL_FUNCTION(TIME_ID), 'YYYY') IS NOT NULL				
TRUNC(INTERNAL_FUNCTION(TIME_ID), 'MM') IS NOT NULL				
HASH			200	3174
TABLE ACCESS	SA_PROMOTIONS.SA_TRANSACTION	GROUP BY		
FULL			336000	3160



The screenshot shows the SQL Developer interface with the same query as above. The Connections pane is visible on the left, showing the database connections. The query results are displayed in a table with the following columns: YEAR, MONTH, GROSS_SALARY_EMPLOYEE_DOLLAR_AMOUNT, and NET_SALARY_EMPLOYEE_DOLLAR_AMOUNT. The data shows monthly salary information for the year 2022.

YEAR	MONTH	GROSS_SALARY_EMPLOYEE_DOLLAR_AMOUNT	NET_SALARY_EMPLOYEE_DOLLAR_AMOUNT
01-JAN-22	01-JAN-22	16740345.73	14286455
02-JAN-22	01-FEB-22	15139839.55	12919417
03-JAN-22	01-MAR-22	16702897.55	14324052
04-JAN-22	01-APR-22	14197505.94	13523513
05-JAN-22	01-MAY-22	14768528.37	14313747
06-JAN-22	01-JUN-22	16224818.16	13847563
07-JAN-22	01-JUL-22	10340195.31	8826010

Welcome Page | VeraDB | Lab11_3.1.Task 03.sql

SQL Worksheet: History

Worksheet | Query Builder

```

4 SET AUTOTRACE ON;
5 select
6   TRUNC(TIME_ID, 'YYYY') AS YEAR,
7   TRUNC(TIME_ID, 'MM') AS MONTH,
8   SUM (ROUND(employee_salary_project*(1+ agency_VAT_percent/100),2)) as gross_salary_employee_dollar_amount,
9   SUM (employee_salary_project) as net_salary_employee_dollar_amount
10  from SA_TRANSACTION
11  GROUP BY CUBE (TRUNC(TIME_ID, 'YYYY'), TRUNC(TIME_ID, 'MM'))
12  HAVING
13   (TRUNC(TIME_ID, 'YYYY') IS NOT NULL
14   AND
15   TRUNC(TIME_ID, 'MM') IS NOT NULL)
16  order by TRUNC(TIME_ID, 'YYYY'), TRUNC(TIME_ID, 'MM');

```

Script Output | Explain Plan | Query Result

Task completed in 0.762 seconds

PLAN_TABLE_OUTPUT

Plan hash value: 101598372

Id	Operation	Name	E-Rows	OMem	lMem	Used-Mem
0	SELECT STATEMENT					
1	SORT ORDER BY		200	2048	2048	(0)
2	FILTER					
3	HASH GROUP BY		200	696K	696K	
4	TABLE ACCESS FULL	SA_TRANSACTION	336K			

PLAN_TABLE_OUTPUT

Welcome Page | VeraDB | Lab11_3.1.Task 03.sql

SQL Worksheet: History

Worksheet | Query Builder

```

2 |
3 alter session set current_schema=sa_promotions;
4 set autotrace traceonly;
5 select
6   TRUNC(TIME_ID, 'YYYY') AS YEAR,
7   TRUNC(TIME_ID, 'MM') AS MONTH,
8   SUM (ROUND(employee_salary_project*(1+ agency_VAT_percent/100),2)) as gross_salary_employee_dollar_amount,
9   SUM (employee_salary_project) as net_salary_employee_dollar_amount
10  from SA_TRANSACTION
11  GROUP BY CUBE (TRUNC(TIME_ID, 'YYYY'), TRUNC(TIME_ID, 'MM'))
12  HAVING
13   (TRUNC(TIME_ID, 'YYYY') IS NOT NULL
14   AND
15   TRUNC(TIME_ID, 'MM') IS NOT NULL)
16  order by TRUNC(TIME_ID, 'YYYY'), TRUNC(TIME_ID, 'MM');

```

Script Output | Query Result | Query Result 1

Task completed in 0.825 seconds

Statistics

```

21 CPU used by this session
21 CPU used when call started
21 DB time
6 Requests to/from client
6091 consistent gets
6091 consistent gets from cache
6091 consistent gets pin
6091 consistent gets pin (fastpath)
6 non-idle wait count
2 opened cursors cumulative
-1 opened cursors current
-1 pinned cursors current
1 process last non-idle time
6091 session logical reads
8 user calls

```

- Monthly salary data for employees for the year 2022 using Module Clause on the DW level.

Welcome Page | VeraDB | Lab11_3.1.Task 03.sql

SQL Worksheet: History

Worksheet | Query Builder

```

21 alter session set current_schema=DW_DATA;
22 |
23 WITH salary_employees AS (
24   SELECT TRUNC(TIME_ID, 'YYYY') AS year,
25   TRUNC(TIME_ID, 'MM') AS month, 'MONTH' period,
26   SUM (ROUND(employee_salary_project*(1+ agency_VAT_percent/100),2)) as gross_salary_employee_dollar_amount,
27   SUM (employee_salary_project) as net_salary_employee_dollar_amount
28  FROM DW_CL.cls_t_transaction
29  GROUP BY TRUNC(TIME_ID, 'YYYY'), TRUNC(TIME_ID, 'MM'))
30
31 select year, month, period, gross_salary_employee_dollar_amount, net_salary_employee_dollar_amount
32  from salary_employees
33  MODEL
34   DIMENSION BY ( year, month, period)
35   MEASURES ( gross_salary_employee_dollar_amount, net_salary_employee_dollar_amount)
36   RULES (
37
38   gross_salary_employee_dollar_amount[FOR year IN (SELECT DISTINCT year FROM salary_employees),

```

Script Output | Query Result

SQL | All Rows Fetched: 9 in 0.237 seconds

YEAR	MONTH	PERIOD	GROSS_SALARY_EMPLOYEE_DOLLAR_AMOUNT	NET_SALARY_EMPLOYEE_DOLLAR_AMOUNT
1	01-JAN-22	01-JAN-22 MONTH	16740345.73	14286455
2	01-JAN-22	01-FEB-22 MONTH	15139839.59	12519417
3	01-JAN-22	01-MAR-22 MONTH	16782887.55	14324052
4	01-JAN-22	01-APR-22 MONTH	16197508.94	13523913
5	01-JAN-22	01-MAY-22 MONTH	16768528.37	14313747
6	01-JAN-22	01-JUN-22 MONTH	16224818.16	13847563
7	01-JAN-22	01-JUL-22 MONTH	10340195.31	8826010
8	01-JAN-22	(null) YEAR	108194123.65	92341157
9	(null)	(null) ALL	108194123.65	92341157

Welcome Page | VeriDB | Lab11_3.1.Task 03.sql

SQL Worksheet: History | 0.45699999 seconds

Worksheet | Query Builder

```

21 alter session set current_schema=DW_DATA;
22
23 WITH salary_employees AS (
24   SELECT TRUNC(TIME_ID, 'YYYY') AS year,
25          TRUNC(TIME_ID, 'MM') AS month, MONTH(period)
26          ,SUM (ROUND(employee_salary_project*(1+ agency_VAT_percent/100),2)) as gross_salary_employee_dollar_amount
27          ,SUM (employee_salary_project) as net_salary_employee_dollar_amount
28 FROM   DW_CL.cls_t_transaction
29 GROUP BY TRUNC(TIME_ID, 'YYYY'), TRUNC(TIME_ID, 'MM'))
30
31 select year, month, period, gross_salary_employee_dollar_amount, net_salary_employee_dollar_amount
32 from salary_employees
33 MODEL
34   DIMENSION BY ( year, month, period)
35   MEASURES ( gross_salary_employee_dollar_amount, net_salary_employee_dollar_amount)

```

Script Output | Query Result | Explain Plan | 0.457 seconds

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST	
SELECT STATEMENT				200	3412
TEMP TABLE TRANSFORMATION					
LOAD AS SELECT	SYS_TEMP_0FD9D94CA_111484C	(CURSOR DURATION MEMORY)			
HASH		GROUP BY	200		3409
TABLE ACCESS	DW_CL.cls_t_transaction	FULL	336000		3402
SQL MODEL	DW_DATA.nvl	ORDERED	200		
TABLE ACCESS	SYS_TEMP_0FD9D94CA_111484C	FULL	200		3
HASH		UNIQUE	200		4
VIEW	DW_DATA.nvl		200		3
TABLE ACCESS	SYS_TEMP_0FD9D94CA_111484C	FULL	200		3
HASH		UNIQUE	200		4
VIEW	DW_DATA.nvl		200		3
TABLE ACCESS	SYS_TEMP_0FD9D94CA_111484C	FULL	200		3

Oracle SQL Developer

File Edit View Navigate Run Tools Window Help

Connections | VeriDB

SQL Worksheet: History | 0.45699999 seconds

Worksheet | Query Builder

```

21 alter session set current_schema=DW_DATA;
22
23 WITH salary_employees AS (
24   SELECT TRUNC(TIME_ID, 'YYYY') AS year,
25          TRUNC(TIME_ID, 'MM') AS month, MONTH(period)
26          ,SUM (ROUND(employee_salary_project*(1+ agency_VAT_percent/100),2)) as gross_salary_employee_dollar_amount
27          ,SUM (employee_salary_project) as net_salary_employee_dollar_amount
28 FROM   DW_CL.cls_t_transaction
29 GROUP BY TRUNC(TIME_ID, 'YYYY'), TRUNC(TIME_ID, 'MM'))
30
31 select year, month, period, gross_salary_employee_dollar_amount, net_salary_employee_dollar_amount
32 from salary_employees
33 MODEL
34   DIMENSION BY ( year, month, period)
35   MEASURES ( gross_salary_employee_dollar_amount, net_salary_employee_dollar amount)

```

Script Output | Query Result | Explain Plan | 0.457 seconds

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST	
SELECT STATEMENT				200	3412
TEMP TABLE TRANSFORMATION					
LOAD AS SELECT	SYS_TEMP_0FD9D94CA_111484C	(CURSOR DURATION MEMORY)			
HASH		GROUP BY	200		3409
TABLE ACCESS	DW_CL.cls_t_transaction	FULL	336000		3402
SQL MODEL	DW_DATA.nvl	ORDERED	200		
TABLE ACCESS	SYS_TEMP_0FD9D94CA_111484C	FULL	200		3
HASH		UNIQUE	200		4

SQL History

SQL	Connection	TimeStamp	Type	Executed	Duration(se...)
WITH salary_employees AS (SELECT TRUNC(TIME_ID, 'YYYY') AS year,...	VeriDB	21-AUG-22...	SQL	4	0.237

Welcome Page | VeriDB | Lab11_3.1.Task 03.sql

SQL Worksheet: History | 0.63300002 seconds

Worksheet | Query Builder

```

22 SET AUTOTRACE ON;
23 WITH salary_employees AS (
24   SELECT TRUNC(TIME_ID, 'YYYY') AS year,
25          TRUNC(TIME_ID, 'MM') AS month, MONTH(period)
26          ,SUM (ROUND(employee_salary_project*(1+ agency_VAT_percent/100),2)) as gross_salary_employee_dollar_amount
27          ,SUM (employee_salary_project) as net_salary_employee_dollar_amount
28 FROM   DW_CL.cls_t_transaction
29 GROUP BY TRUNC(TIME_ID, 'YYYY'), TRUNC(TIME_ID, 'MM'))

```

Script Output | Explain Plan | Query Result | Task completed in 0.633 seconds

PLAN_TABLE_OUTPUT							
Id	Operation	Name	E-Rows	OMem	lMem	Used-Mem	
0	SELECT STATEMENT						
1	TEMP TABLE TRANSFORMATION						
2	LOAD AS SELECT (CURSOR DURATION MEMORY)	SYS_TEMP_0FD9D94A5_111484C		1024	1024		
3	HASH GROUP BY		200	696K	696K		
4	TABLE ACCESS FULL	CLS_T_TRANSACTION	336K				
5	SQL MODEL ORDERED		200	496K	496K	495K (0)	
6	VIEW		200				
7	TABLE ACCESS FULL	SYS_TEMP_0FD9D94A5_111484C	200				
8	HASH UNIQUE		200	851K	851K		
PLAN_TABLE_OUTPUT							
9	VIEW		200				
10	TABLE ACCESS FULL	SYS_TEMP_0FD9D94A5_111484C	200				
11	HASH UNIQUE		200	851K	851K		
12	VIEW		200				
13	TABLE ACCESS FULL	SYS_TEMP_0FD9D94A5_111484C	200				

Welcome Page | VeraDB | Lab11_3.1.Task 03.sql

SQL Worksheet History | 0.8500002 seconds

Worksheet | Query Builder

```

22 set autotrace traceonly;
23 WITH salary_employees AS (
24 SELECT TRUNC(TIME_ID, 'YYYY') AS year,
25        TRUNC(TIME_ID, 'MM') AS month, MONTH(period)
26        ,SUM (ROUND(employee_salary_project*(1+agency_VAT_percent/100),2)) as gross_salary_employee_dollar_amount
27        ,SUM (employee_salary_project) as net_salary_employee_dollar_amount
28 FROM DW_CL.cla_t_transaction
29 GROUP BY TRUNC(TIME_ID, 'YYYY'), TRUNC (TIME_ID, 'MM'))

```

Script Output x | Query Result x

Task completed in 0.85 seconds

Statistics

```

23 CPU used by this session
23 CPU used when call started
23 DB time
5 Requests to/from client
6614 consistent gets
6614 consistent gets from cache
6614 consistent gets pin
6614 consistent gets pin (fastpath)
2 db block gets
2 db block gets from cache
2 db block gets from cache (fastpath)
1 enqueue releases
1 enqueue requests
1 messages sent
6 non-idle wait count
2 opened cursors cumulative
2 opened cursors current
1 pinned cursors current
6616 session logical reads
6 user calls

```

- Monthly salary data for employees for the year 2022 using Star Schema on the DW level.

Welcome Page | VeraDB | Lab11_3.1.Task 03.sql

SQL Worksheet History

Worksheet | Query Builder

```

49 alter session set current_schema=DW_DATA;
50
51 select
52 TRUNC(TIME_ID, 'YYYY') AS YEAR,
53 TRUNC(TIME_ID, 'MM') AS MONTH,
54 SUM (gross_salary_employee_dollar_amount)/" as employee_salary_project_GROSS"/,
55 SUM (net_salary_employee_dollar_amount) /" as employee_salary_project_NET"/
56 from DW_DATA.FCT_BUSINESS
57 GROUP BY CUBE (TRUNC(TIME_ID, 'YYYY'), TRUNC (TIME_ID, 'MM'))
58 HAVING
59 (TRUNC(TIME_ID, 'YYYY') IS NOT NULL
60 AND
61 TRUNC(TIME_ID, 'MM') IS NOT NULL)
62 order by TRUNC(TIME_ID, 'YYYY'), TRUNC (TIME_ID, 'MM');

```

Script Output x | Query Result x

All Rows Fetched: 7 in 0.145 seconds

YEAR	MONTH	SUM(GROSS_SALARY_EMPLOYEE_DOLLAR_AMOUNT)/"ASEMPLOYEE_SALARY_PROJECT_GROSS"/	SUM(NET_SALARY_EMPLOYEE_DOLLAR_AMOUNT)/"ASEMPLOYEE_SALARY_PROJECT_NET"/
01-JAN-22	01-JAN-22	16740345.73	14286455
01-JAN-22	01-FEB-22	15139839.59	12919417
01-JAN-22	01-MAR-22	16782887.55	14324052
01-JAN-22	01-APR-22	16197508.94	13823913
01-JAN-22	01-MAY-22	16768528.37	14313747
01-JAN-22	01-JUN-22	16224818.16	13847563
01-JAN-22	01-JUL-22	10340195.31	8826010

Welcome Page | VeraDB | Lab11_3.1.Task 03.sql

SQL Worksheet History | 0.257 seconds

Worksheet | Query Builder

```

49 alter session set current_schema=DW_DATA;
50
51 select
52 TRUNC(TIME_ID, 'YYYY') AS YEAR,
53 TRUNC(TIME_ID, 'MM') AS MONTH,
54 SUM (gross_salary_employee_dollar_amount)/" as employee_salary_project_GROSS"/,
55 SUM (net_salary_employee_dollar_amount) /" as employee_salary_project_NET"/
56 from DW_DATA.FCT_BUSINESS
57 GROUP BY CUBE (TRUNC(TIME_ID, 'YYYY'), TRUNC (TIME_ID, 'MM'))
58 HAVING
59 (TRUNC(TIME_ID, 'YYYY') IS NOT NULL
60 AND
61 TRUNC(TIME_ID, 'MM') IS NOT NULL)
62 order by TRUNC(TIME_ID, 'YYYY'), TRUNC (TIME_ID, 'MM');

```

Script Output x | Query Result x | Explain Plan x

SQL | 0.257 seconds

OPERATION	OBJECT_NAME	OPTIONS	PARTITION_START	PARTITION_STOP	PARTITION_ID
SELECT STATEMENT					
SORT		ORDER BY			
FILTER					
AND					
TRUNC(INTERNAL_FUNCTION(TIME_ID), 'YYYY') IS NOT NULL					
TRUNC(INTERNAL_FUNCTION(TIME_ID), 'MM') IS NOT NULL					
PARTITION RANGE		ALL		1	4
HASH		GROUP BY			
PARTITION HASH		ALL	1		4
TABLE ACCESS	DW_DATA.FCT_BUSINESS	FULL	1		16

Other XML (info) info type="nodeid(pflags)"

Welcome Page | VeracDB | Lab11_3.1.Task 03.sql

SQL Worksheet: History | 0.257 seconds

Worksheet | Query Builder

```

49 alter session set current_schema=DW_DATA;
50
51 select
52     TRUNC(TIME_ID, 'YYYY') AS YEAR,
53     TRUNC(TIME_ID, 'MM') AS MONTH,
54     SUM (gross_salary_employee_dollar_amount) /* as employee_salary_project_GROSS*/,
55     SUM (net_salary_employee_dollar_amount) /*as employee_salary_project_NET*/
56 from DW_DATA.FCT_BUSINESS
57 GROUP BY CUBE (TRUNC(TIME_ID, 'YYYY'), TRUNC (TIME_ID, 'MM'))
58 HAVING
59     (TRUNC(TIME_ID, 'YYYY') IS NOT NULL
60 AND
61     TRUNC(TIME_ID, 'MM') IS NOT NULL)
62 order by TRUNC(TIME_ID, 'YYYY'), TRUNC (TIME_ID, 'MM');

```

Script Output | Query Result | Explain Plan

SQL | 0.257 seconds

STEPS	PARTITION_START	PARTITION_STOP	PARTITION_ID	CARDINALITY	COST
ORDER BY				200	1398
				200	1398
LL		1	4	3	200
GROUP BY				200	1398
LL		1	4	5	336000
ALL		1	16	5	336000

Welcome Page | VeracDB | Lab11_3.1.Task 03.sql

SQL Worksheet: History | 0.257 seconds

Worksheet | Query Builder

```

49 alter session set current_schema=DW_DATA;
50
51 select
52     TRUNC(TIME_ID, 'YYYY') AS YEAR,
53     TRUNC(TIME_ID, 'MM') AS MONTH,
54     SUM (gross_salary_employee_dollar_amount) /* as employee_salary_project_GROSS*/,
55     SUM (net_salary_employee_dollar_amount) /*as employee_salary_project_NET*/
56 from DW_DATA.FCT_BUSINESS
57 GROUP BY CUBE (TRUNC(TIME_ID, 'YYYY'), TRUNC (TIME_ID, 'MM'))
58 HAVING
59     (TRUNC(TIME_ID, 'YYYY') IS NOT NULL
60 AND
61     TRUNC(TIME_ID, 'MM') IS NOT NULL)
62 order by TRUNC(TIME_ID, 'YYYY'), TRUNC (TIME_ID, 'MM');

```

Script Output | Query Result | Explain Plan

SQL | 0.257 seconds

STEPS	PARTITION_START	PARTITION_STOP	PARTITION_ID	CARDINALITY	COST
ORDER BY				200	1398
				200	1398
LL		1	4	3	200
GROUP BY				200	1398
LL		1	4	5	336000
ALL		1	16	5	336000

SQL History

SQL	Connection	TimeStamp	Type	Executed	Duration(se...)
select TRUNC(TIME_ID, 'YYYY') AS YEAR, TRUNC(TIME_ID, 'MM') AS MONTH, SUM (gross_salary_employee_dollar_amount) /* as employee_salary_project_GROSS*/, SUM (net_salary_employee_dollar_amount) /*as employee_salary_project_NET*/ from DW_DATA.FCT_BUSINESS GROUP BY CUBE (TRUNC(TIME_ID, 'YYYY'), TRUNC (TIME_ID, 'MM')) HAVING (TRUNC(TIME_ID, 'YYYY') IS NOT NULL AND TRUNC(TIME_ID, 'MM') IS NOT NULL) order by TRUNC(TIME_ID, 'YYYY'), TRUNC (TIME_ID, 'MM');	VeracDB	21-AUG-22...	SQL	4	0.145

Messages - Log | SQL History

Welcome Page | VeracDB | Lab11_3.1.Task 03.sql

SQL Worksheet: History

Worksheet | Query Builder

```

47 /*D02_Lab11 Monthly salary data for employees for the year 2022 using Star Schema on the DW level.*/
48
49 alter session set current_schema=DW_DATA;
50 set autotrace traceonly;
51 select
52     TRUNC(TIME_ID, 'YYYY') AS YEAR,
53     TRUNC(TIME_ID, 'MM') AS MONTH,
54     SUM (gross_salary_employee_dollar_amount) /* as employee_salary_project_GROSS*/,
55     SUM (net_salary_employee_dollar_amount) /*as employee_salary_project_NET*/
56 from DW_DATA.FCT_BUSINESS
57 GROUP BY CUBE (TRUNC(TIME_ID, 'YYYY'), TRUNC (TIME_ID, 'MM'))
58 HAVING
59     (TRUNC(TIME_ID, 'YYYY') IS NOT NULL
60 AND
61     TRUNC(TIME_ID, 'MM') IS NOT NULL)
62 order by TRUNC(TIME_ID, 'YYYY'), TRUNC (TIME_ID, 'MM');

```

Script Output | Query Result

Task completed in 1.229 seconds

Id	Operation	Name	E-Rows	OMem	lMem	Used-Mem
0	SELECT STATEMENT					
1	SORT ORDER BY		200	2048	2048	2048 (0)
2	FILTER					
3	PARTITION RANGE ALL		200			
4	HASH GROUP BY		200	688K	688K	
5	PARTITION HASH ALL		336K			

PLAN_TABLE_OUTPUT

6	TABLE ACCESS FULL	FCT_BUSINESS	336K			
---	-------------------	--------------	------	--	--	--

```

49 alter session set current_schema=DW_DATA;
50 set autotrace traceonly;
51 select
52     TRUNC(TIME_ID, 'YYYY') AS YEAR,
53     TRUNC(TIME_ID, 'MM') AS MONTH,
54     SUM(gross_salary_employee_dollar_amount) /* as employee_salary_project_GROSS */
55     SUM(net_salary_employee_dollar_amount) /* as employee_salary_project_NET */
56 from DW_DATA.FCT_BUSINESS
57 GROUP BY CUBE (TRUNC(TIME_ID, 'YYYY'), TRUNC(TIME_ID, 'MM'))
58 HAVING
59     (TRUNC(TIME_ID, 'YYYY') IS NOT NULL
60 AND

```

Script Output x Query Result x

Task completed in 1.229 seconds

Statistics

```

-----
      8 CPU used by this session
      8 CPU used when call started
      7 DB time
      5 Requests to/from client
    1274 consistent gets
    1274 consistent gets from cache
    1274 consistent gets pin
    1274 consistent gets pin (fastpath)
      6 non-idle wait count
      2 opened cursors cumulative
      2 opened cursors current
      1 pinned cursors current
    1274 session logical reads
      6 user calls

```

Summarize table:

Nº	Source Type	Explain Plan - Statistics			Time, Sec.
		CARDINALITY	COST	CONSISTENT GETS	
1	Advancing Grouping	200	3174	6091	0.205 sec
2	Model Clause	200	3412	6614	0.237 sec
3	Star Schema	200	1398	1274	0.145 sec

Summary: As we can see, with the same number of query strings returned, by examining the TIME, COST and CONSISTENT GETS we can conclude that a query that works with Star Schema works faster and costs less.