

# Report

Laboratory Work 1

Dmitry Ladutsko

August 02, 2022

## 1. Prerequisites

### 1.1. Passwords Index

Password Group	Login Name	Password
Operation System	root	"rootadmin"
	oracle	"oracleadmin"
Oracle System	sys	"sysadmin"
	system	"sysadmin"
Oracle Users	All DB users	"%PWD%"

### 1.2. Folder Paths Index

Path Group	Path Description	Path
Operation System	Oracle RDBMS – BIN	/oracle/app/oracle
	Oracle Inventory	/oracle/app/oraInventory
	Oracle Database Storage	/oracle/oradata
	Oracle Install Directory	/oracle/install
Oracle	ORACLE_BASE	/oracle/app/oracle
	ORACLE_HOME	\$ORACLE_BASE/product/11.2
FTP	ftp Incoming Folder	/ftp/incoming

## Data Warehouse Architecture – Storage Layers

### 2.1. Task 01: CREATE Storage Objects

**The Main Task** is to create Physical Objects according your Solution Proposal that was developed on Module 6 – Oracle DB. Introduction to DWH.

Now I created DDL's for all tables, except of fact table, of course:)

Worksheet		Query Builder	
1	drop table sa_time;		
2			
3	alter session set current_schema=SA_DATE;		
4			
5	CREATE table sa_time (	5	create table SA_CUSTOMERS(
6	date_id DATE,	6	FIRST_NAME_CUSTOMER CHAR(20),
7	beg_of_year DATE ,	7	LAST_NAME_CUSTOMER CHAR(20),
8	end_of_year DATE ,	8	COUNTRY_CITY_CUSTOMER CHAR(20),
9	day_name VARCHAR2(44) ,	9	ADDRESS_CUSTOMER CHAR(50),
10	day_number_in_week VARCHAR2(1) ,	10	EMAIL CHAR(30),
11	day_number_in_month VARCHAR2(2) ,	11	PHONE_CUSTOMER NUMBER,
12	day_number_in_year VARCHAR2(3) ,	12	AGE NUMBER,
13	calendar_week_number VARCHAR2(1) ,	13	IS_ACTIVE VARCHAR2(6),
14	week_ending_date DATE ,	14	CUSTOMER_SALE_DATE DATE
15	calendar_month_number VARCHAR2(2) ,	15	);
16	days_in_month VARCHAR2(2) ,		
17	end_of_month DATE ,		
18	end_of_quarter VARCHAR2(32) ,		
19	quarter_number VARCHAR2(1) ,		
20	year VARCHAR2(4) ,		
21	days_in_cal_year NUMBER		
22	);		

Picture 1 - Creating calendar and customer tables

5	CREATE TABLE SA_EMPLOYEES (	1	DROP TABLE SA_PAYMENT_METHODS;
6	first_name_EMPLOYEE VARCHAR2(40) NOT NULL,	2	
7	last_name_EMPLOYEE VARCHAR2(40) NOT NULL,	3	alter session set current_schema = SA_CUSTOMERS;
8	email_EMPLOYEE VARCHAR2(40) NOT NULL,	4	
9	phone_EMPLOYEE VARCHAR2(40) NOT NULL,	5	CREATE TABLE SA_PAYMENT_METHODS (
10	POSITION_NAME VARCHAR2(40) NOT NULL,	6	PAYMENT_METHOD_NAME VARCHAR2(40) NOT NULL,
11	POSITION_DEGREE VARCHAR2(40) NOT NULL,	7	BANK_NAME VARCHAR2(40) NOT NULL
12	SALES_TYPE VARCHAR2(40) NOT NULL,	8	);
13	HIRE_DATE DATE NOT NULL		
14	);		

Picture 2 - Creating employees and payment methods tables

1	drop table SA_PRODUCTS;	1	DROP TABLE SA_STORES;
2		2	
3	alter session set current_schema = SA_PRODUCTS;	3	alter session set current_schema = SA_DATE;
4		4	
5	create table SA_PRODUCTS (	5	CREATE TABLE SA_STORES (
6	PRODUCT_NAME VARCHAR2(30),	6	country_city VARCHAR2(20) NOT NULL,
7	MODEL_NAME VARCHAR2(20),	7	address VARCHAR2(20) NOT NULL,
8	MEMORY_AMOUNT NUMBER(22,0),	8	phone NUMBER NOT NULL
9	COLOR_NAME VARCHAR2(25),	9	);
10	PRICE NUMBER(10),		
11	INSERT_DT DATE,		
12	UPDATE_DT DATE		
13	);		

Picture 3 - Creating products and stores tables

```

13 create table sa_transactions(
14 --CUSTOMER
15 FIRST_NAME_CUSTOMER CHAR(20),
16 LAST_NAME_CUSTOMER CHAR(20),
17 COUNTRY_CITY_CUSTOMER CHAR(20),
18 ADDRESS_CUSTOMER CHAR(50),
19 EMAIL CHAR(30),
20 PHONE_CUSTOMER NUMBER,
21 AGE NUMBER,
22 IS_ACTIVE VARCHAR2(6),
23 CUSTOMER_SALE_DATE DATE,
24 -----EMPLOYEE
25 first_name_EMPLOYEE VARCHAR2(40) NOT NULL,
26 last_name_EMPLOYEE VARCHAR2(40) NOT NULL,
27 email_EMPLOYEE VARCHAR2(40) NOT NULL,
28 phone_EMPLOYEE VARCHAR2(60) NOT NULL,
29 POSITION_NAME VARCHAR2(40) NOT NULL,
30 POSITION_DEGREE VARCHAR2(40) NOT NULL,
31 SALES_TYPE VARCHAR2(40) NOT NULL,
32 HIRE_DATE DATE NOT NULL
33 -----PAYMENT METHOD
34 PAYMENT_METHOD_NAME VARCHAR2(40) NOT NULL,
35 BANK_NAME VARCHAR2(40) NOT NULL,
36 -----PRODUCT
37 PRODUCT_NAME VARCHAR2(30),
38 MODEL_NAME VARCHAR2(20),
39 MEMORY_AMOUNT NUMBER(22,0),
40 COLOR_NAME VARCHAR2(25),
41 PRICE NUMBER(10),
42 INSERT_DT DATE,
43 UPDATE_DT DATE,
44 -----CALENDAR
45 date_id DATE,
46 beg_of_year DATE,
47 end_of_year DATE,
48 day_name VARCHAR2(44),
49 day_number_in_week VARCHAR2(1),
50 day_number_in_month VARCHAR2(2),
51 day_number_in_year VARCHAR2(3),
52 calendar week number VARCHAR2(1),

```

Picture 4 – DML

```

63 alter session set current_schema = SA_CUSTOMERS;
64 INSERT INTO sa_transactions
65 (
66 SELECT /*+ parallel(SA_CUSTOMERS.sa_CUSTOMERS, 4)*/ *
67 FROM
68 SA_CUSTOMERS.sa_CUSTOMERS
69 CROSS JOIN (SELECT * FROM SA_CUSTOMERS.sa_customers
70 WHERE PHONE_CUSTOMER > 9900000)
71 CROSS JOIN (SELECT * FROM SA_PRODUCTS.sa_products)
72 INNER JOIN SA_DATE.sa_time ON date_id = UPDATE_DT
73 WHERE UPDATE_DT > TO_DATE( '01.01.19', 'MM/DD/YY' ) AND UPDATE_DT < TO_DATE( '04.01.22', 'MM/DD/YY' )
74 );

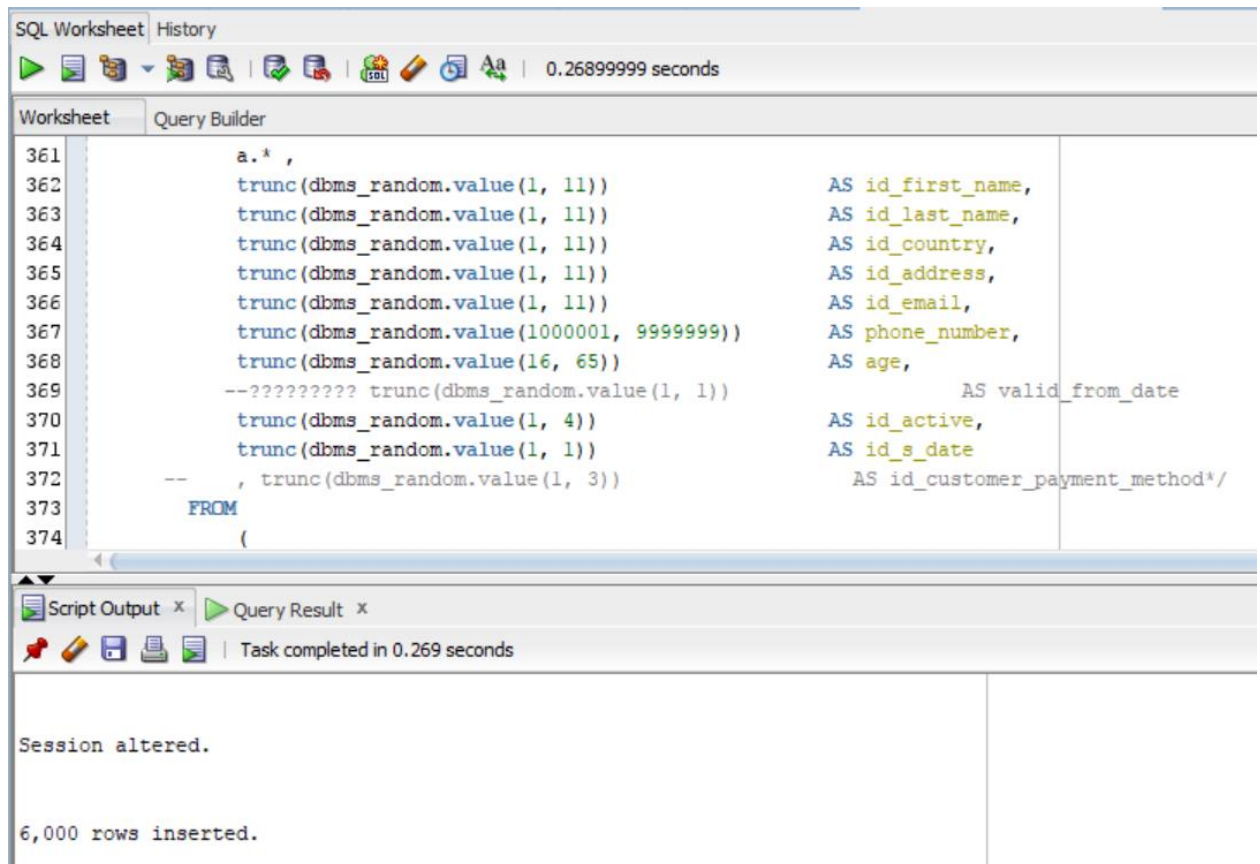
```

Picture 5 - Cross Joins on transaction with other tables

## 2.2. Task 02: Generate Test Data in Storage Layers

**The Main Task** is to generate test data on Storage layers objects, that was created on task 01.

Now I decided to generate more values data, which is also going to be more relevant, apart from previous DML scripts... (All DDL/DML Scripts stored on GitHub )



```
361      a.* ,
362      trunc(dbms_random.value(1, 11))          AS id_first_name,
363      trunc(dbms_random.value(1, 11))          AS id_last_name,
364      trunc(dbms_random.value(1, 11))          AS id_country,
365      trunc(dbms_random.value(1, 11))          AS id_address,
366      trunc(dbms_random.value(1, 11))          AS id_email,
367      trunc(dbms_random.value(1000001, 9999999)) AS phone_number,
368      trunc(dbms_random.value(16, 65))          AS age,
369      --????????? trunc(dbms_random.value(1, 1)) AS valid_from_date
370      trunc(dbms_random.value(1, 4))            AS id_active,
371      trunc(dbms_random.value(1, 1))            AS id_s_date
372      -- , trunc(dbms_random.value(1, 3))        AS id_customer_payment_method*/
373  FROM
374  (
```

Script Output x Query Result x

Task completed in 0.269 seconds

Session altered.

6,000 rows inserted.

Picture 6 – Customers DML

SQL Worksheet History

0.361 seconds

Worksheet Query Builder

```
412      dual
413      ), create_employee AS (
414      SELECT
415          a.* ,
416          trunc(dbms_random.value(1, 21))          AS id_first_name,
417          trunc(dbms_random.value(1, 14))          AS id_last_name,
418          trunc(dbms_random.value(1, 21))          AS id_email,
419          trunc(dbms_random.value(1000001, 9999999)) AS phone_number,
420          trunc(dbms_random.value(1, 5))           AS id_position,
421          trunc(dbms_random.value(1, 5))           AS id_degree,
422          trunc(dbms_random.value(1, 5))           AS id_type
423      FROM
424      (
425          SELECT
426              ROWNUM rn
```

Script Output x Query Result x

Task completed in 0.361 seconds

Session altered.

1,500 rows inserted.

Picture 7 - Employees DML

SQL Worksheet History

Worksheet Query Builder

```

118         ), create_payment_method AS (
119         SELECT
120             a.* ,
121             trunc(dbms_random.value(1, 4))           AS id_payment_method,
122             trunc(dbms_random.value(1, 15))          AS id_bank_name
123         FROM
124             (
125                 SELECT
126                     ROWNUM rn
127                 FROM
128                     dual
129                 CONNECT BY

```

Script Output x Query Result x

SQL | All Rows Fetched: 100 in 0.081 seconds

	PAYMENT_METHOD_N...	BANK_NAME
33	Card	Landesbank Baden-Württemberg Bank
34	Card	Landesbank Baden-Württemberg Bank
35	Card	Royal Bank of Scotland
36	Cash	Royal Bank of Scotland
37	Cash	Raiffeisen Bank
38	Cash	Raiffeisen Bank
39	Cash	Ukrexim Bank

Picture 8 - Payment Method DML







SQL Worksheet History

0.47799999 seconds

Worksheet Query Builder

```

38         end_of_quarter      ,
39         quarter_number     ,
40         year                ,
41         days_in_cal_year
42     )
43 SELECT
44     TRUNC( sd + rn ) date_id,
45     TRUNC( sd ) beg_of_year,
46     TO_CHAR( sd + 364 ) end_of_year,
47     TO_CHAR( sd + rn, 'fmDay' ) day_name,
48     TO_CHAR( sd + rn, 'D' ) day_number_in_week,
49     TO_CHAR( sd + rn, 'DD' ) day_number_in_month,
50     TO_CHAR( sd + rn, 'DDD' ) day_number_in_year,
51     TO_CHAR( sd + rn, 'W' ) calendar_week_number,
52     ( CASE
53         WHEN TO_CHAR( sd + rn, 'D' ) IN ( 2, 3, 4, 5, 6, 7 ) THEN
54             NEXT_DAY( sd + rn, 'Monday' )
55         ELSE
56             ( sd + rn )

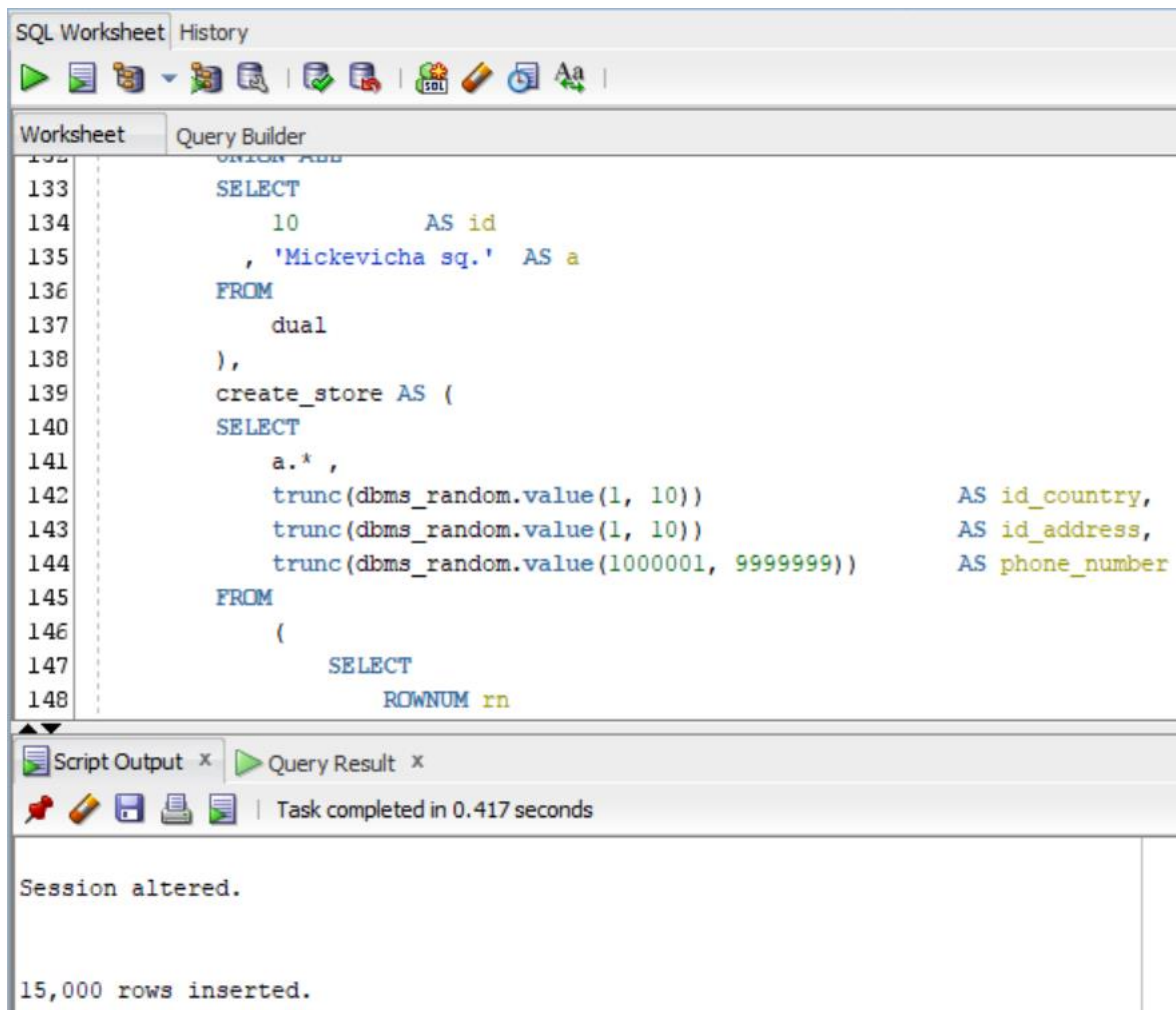
```

Script Output x Query Result x Query Result 1 x Query Result 2 x Query Result 3 x Query Result 4 x

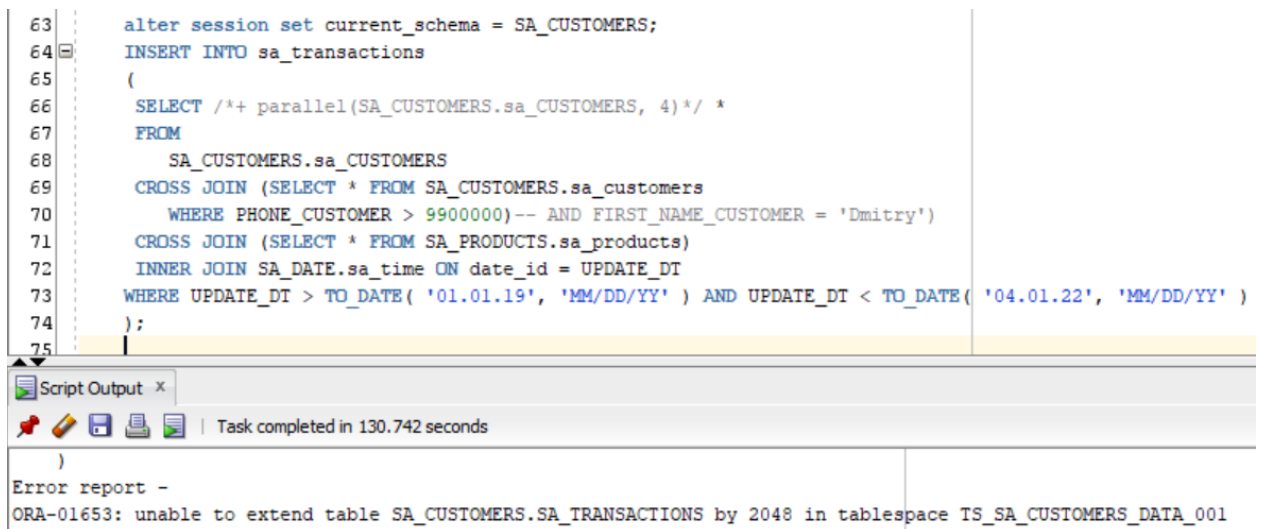
Task completed in 0.478 seconds

30,000 rows inserted.

Picture 10 - Calendar DML



Picture 11 - Store DML



Picture 12 - Transaction DML

	PHONE_CUSTOMER	AGE	IS_ACTIVE	CUSTOMER_SALE_DATE	PRODUCT_NAME	MODEL_NAME	MEMORY_AMOUNT	COLOR_NAME	PRICE	INSERT_DT	UPDATE_DT
5158	8368163	40	no	14-APR-20	Iphone	12	64	Blue	1999	26-SEP-21	03-DEC-21
5159	8368163	40	no	14-APR-20	Iphone	12	512	Pink	1999	26-SEP-21	17-DEC-21
5160	8368163	40	no	14-APR-20	Ipad	12	16	Pink	599	19-OCT-21	14-JAN-22
5161	5774337	35	yes	14-APR-20	Ipod	11	2048	White	699	19-OCT-21	04-JAN-22
5162	5774337	35	yes	14-APR-20	Ipad	11	128	Blue	699	19-OCT-21	13-DEC-21
5163	5774337	35	yes	14-APR-20	Ipad	XS	2048	Green	699	19-OCT-21	03-DEC-21
5164	5774337	35	yes	14-APR-20	Ipod	Pro	512	White	899	19-OCT-21	05-NOV-21
5165	5774337	35	yes	14-APR-20	AirPods	Mini	64	Grey	899	19-OCT-21	12-JAN-22

Picture 13 - Transaction Table

```

43 select count (*) from sa_transactions;
44
45

```

Script Output x Query Result x

Task completed in 0.046 seconds

Session altered.

595,000 rows inserted.

COUNT (*)
595000

Picture 14 - Count from Transaction

**Note: Corrected !:)**

### Task 03: Create Group by Plan

**The Main Task** is to prepare ad hoc SQL for segregate data view on SA\_\* objects (Storage).

#### Operations:

- Select
  - Group by
- Merge
  - Group by

#### Task Results:

```

43 select count (*),sum(price) as total_revenue
44 from sa_transactions
45 where CUSTOMER_SALE_DATE
46 between to_date('01.04.20', 'DD/MM/YY') AND to_date('30.04.20', 'DD/MM/YY')
47
48

```

Script Output x Query Result x

Task completed in 0.149 seconds

COUNT (*)	TOTAL_REVENUE
23800	28672200

Picture 15 - Sales monthly amount with revenue

```

40 select count (*),sum(price) as total_revenue, FIRST_NAME_CUSTOMER as CUSTOMER
41 from sa_transactions
42 where CUSTOMER_SALE_DATE
43 between to_date('01.04.20', 'DD/MM/YY') AND to_date('30.04.20', 'DD/MM/YY')
44 group by FIRST_NAME_CUSTOMER
45 ORDER BY TOTAL_REVENUE DESC
46
47 commit;

```

Picture 16 - Customers purchases grouped by total amount

**Laboratory work summary:**

**At this lab** we have learned how (and which opportunities) gives as Oracle in generating Data for test e.g. It is absolutely clear that generating so much data is not needed every time we have to test SA layer (but also can use back – end p.l. to generate them easily). Nevertheless, this type is also quite acceptable if we have to test smth not to use some Back techs. All diagrams and scripts are stored in GitHub (link in README file in Labs folder)