|  |
| --- |
|  |
|  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| REVISION HISTORY | | | | | |
| Ver. | Description of Change | Author | Date | Approved | |
| Name | Effective Date |
| 1.0 | Initial status | Valeryia\_Lupanava | 20-NOV-2017 |  |  |

Содержание

[1. Генерация данных для фактовой таблицы 3](#_Toc498976577)

[2. Применение аналитических функций 5](#_Toc498976578)

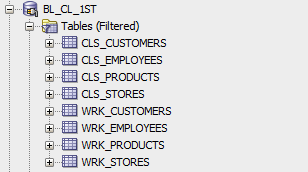
[2.1. AdHoc-запросы с применением аналитических функций 5](#_Toc498976579)

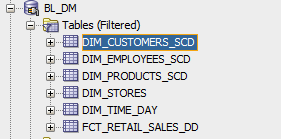
[2.2. AdHoc-запросы с группировкой по времени 8](#_Toc498976580)

# Генерация данных для фактовой таблицы

Для лабораторной работы все данные были сгенерированы на сайте [**www.mockaroo.com**](http://www.mockaroo.com). В качестве источников было создано несколько excel-файлов: customers, employees, products, stores.

В работе было создано два слоя: cleansing - BL\_CL и dimensional – BL\_DM.





В первом слое были созданы WRK-таблицы для загрузки сгенерированных данных. В CLS-таблицах располагаются очищенные данные, готовые для загрузки в DIM-слой.

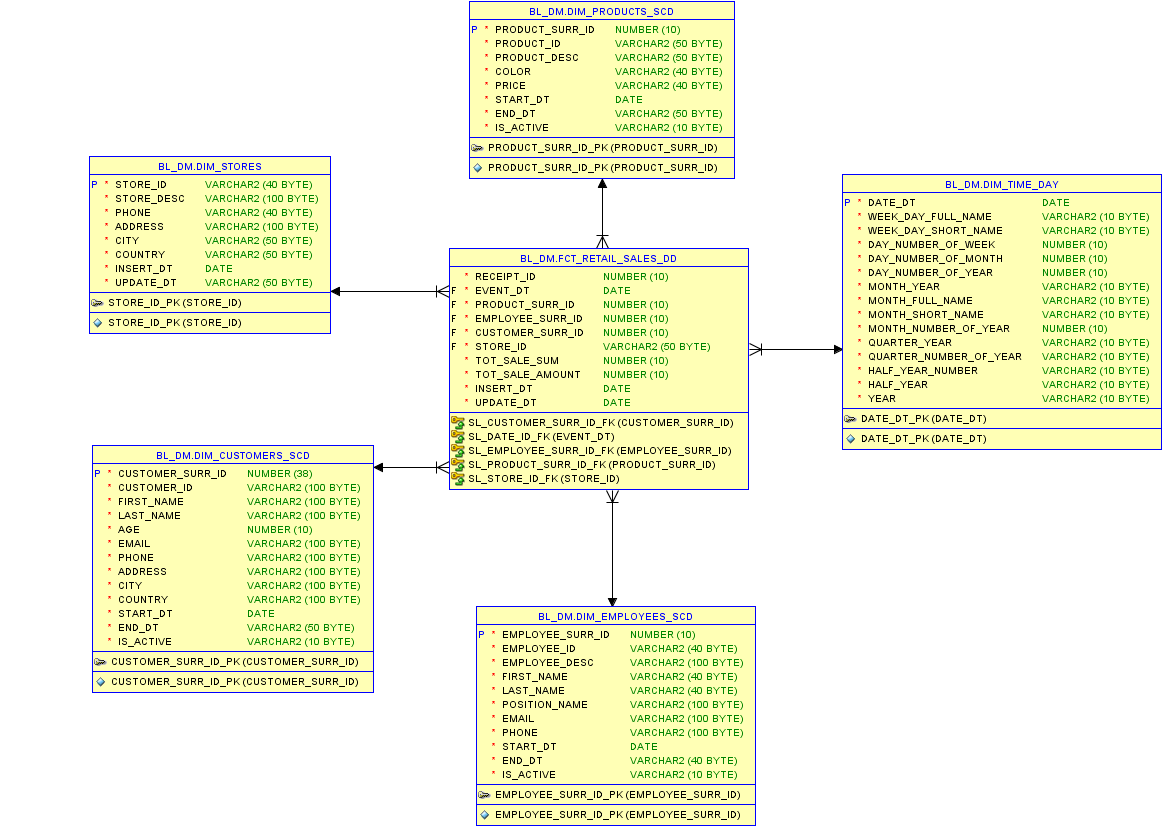
Созданные пользователям были выданы необходимые гранты: connect и resources. Дополнительные гранты, которые потребовались в процессе работы можно найти ы файле **create\_additional\_grants.sql.** Для выдачи грантов был добавлен соответствующий пакет в папке фреймворков **pkg\_create\_grants**.

Данные в WRK-таблицы были загружены через DATA MODELER.

Данные в CLS-таблицы были загружены с помощью пакета **pkg\_insert\_merge**.

Все скрипты по работе с каждым слоем можно найти в соответствущих папках.

Следующим шагом были созданы объекты DIM-слоя. В качестве схемы была выбрана STAR-схема. При создании объектов были добавлены соответствующие ограничения: PRIMARY KEY и FOREIGN KEY. В результате получилась модель, указанная на рисунке ниже.



Следующим шагом были сгенерированы данные для фактовой таблицы. Генерация была осуществлена с помощью следующего скрипта:

BEGIN

FOR i IN 1..20

LOOP

dbms\_random.seed

(

i \* 5

)

;

INSERT INTO fct\_retail\_sales\_dd

SELECT

TRUNC(dbms\_random.value(100000000000, 9999999999999)) as receipt\_id

, TRUNC ( (sysdate + 7) + dbms\_random.value ( 1, 1000 ) ) AS event\_dt

, ROUND ( dbms\_random.value ( ( SELECT MIN ( product\_surr\_id ) FROM dim\_products\_scd), ( SELECT MAX ( product\_surr\_id ) FROM dim\_products\_scd) ) ) AS product\_surr\_id

, ROUND ( dbms\_random.value ( ( SELECT MIN ( employee\_surr\_id ) FROM dim\_employees\_scd), ( SELECT MAX ( employee\_surr\_id ) FROM dim\_employees\_scd) ) ) AS employee\_surr\_id

, ROUND ( dbms\_random.value ( ( SELECT MIN ( customer\_surr\_id ) FROM dim\_customers\_scd), ( SELECT MAX ( customer\_surr\_id ) FROM dim\_customers\_scd) ) ) AS customer\_surr\_id

, ROUND ( dbms\_random.value ( ( SELECT MIN ( store\_surr\_id ) FROM dim\_stores), ( SELECT MAX ( store\_surr\_id ) FROM dim\_stores) ) ) as store\_id

, CEIL(dbms\_random.value(10000, 99999)) AS tot\_sale\_sum

, CEIL(dbms\_random.value(10000, 99999)) AS tot\_sale\_amount

, sysdate AS insert\_dt

, sysdate AS update\_dt

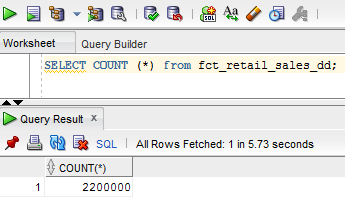
FROM dual

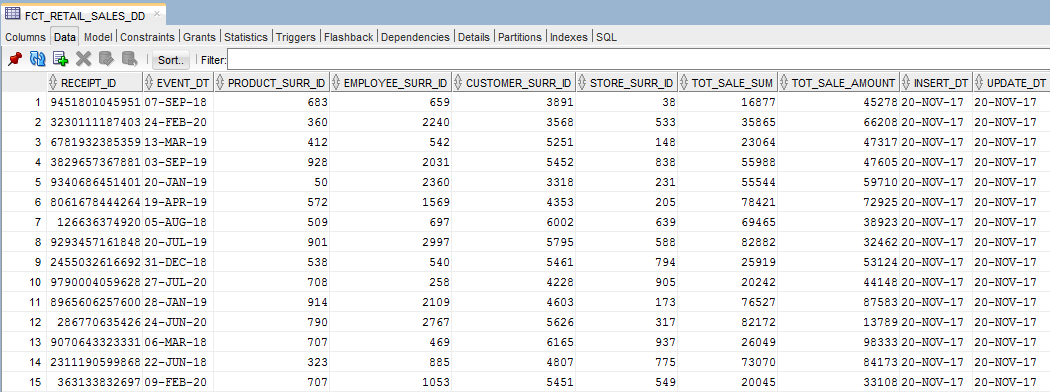
CONNECT BY level <= 100000;

END LOOP;

END;

После выполнения генерации данных в таблицу было добавлено **2 200 000**.

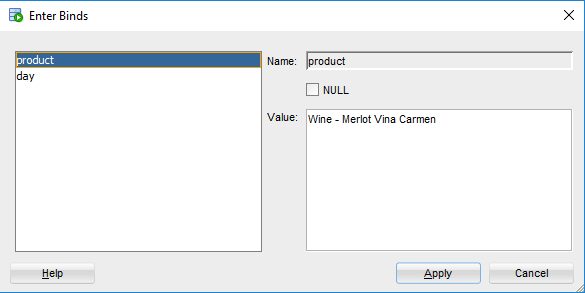




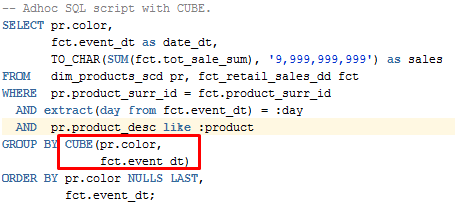
# Применение аналитических функций

## AdHoc-запросы с применением аналитических функций

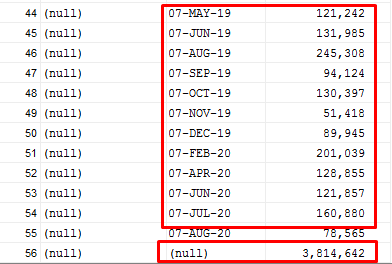
В запросах были заданы параметры: **:day** и **:product**.



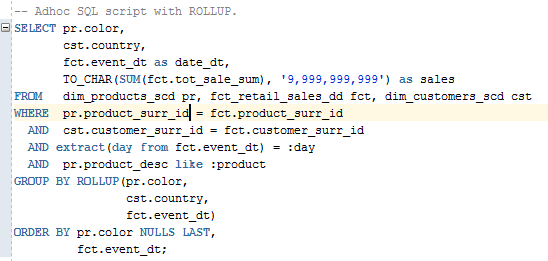
* CUBE



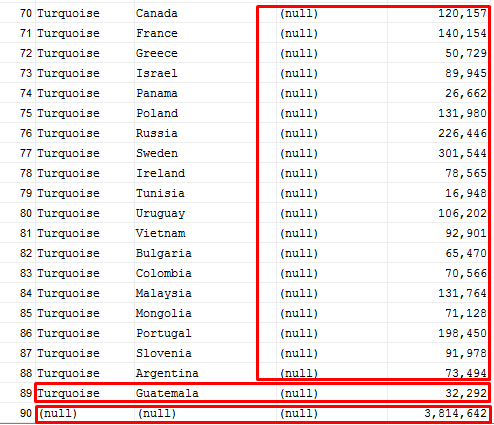
Результат.



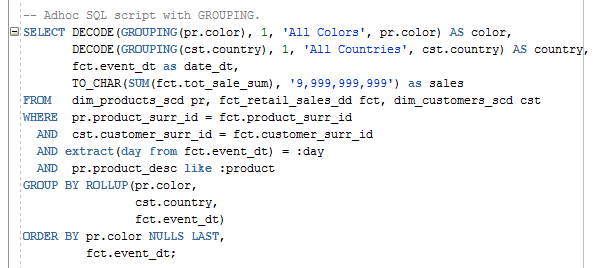
* ROLLUP



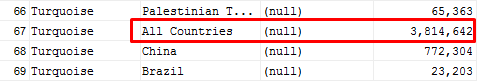
Результат.



* Grouping()



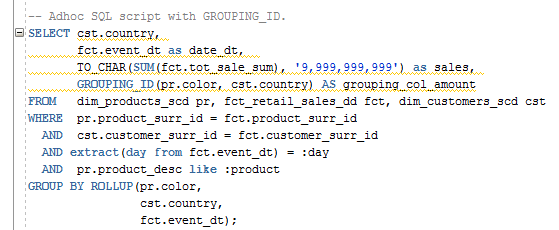
Результат.



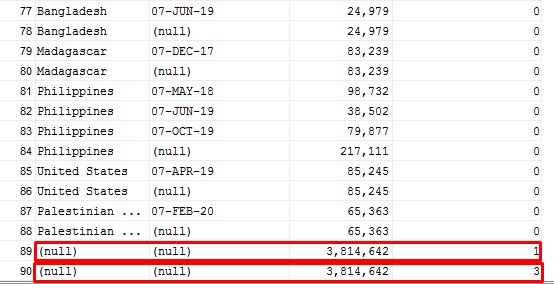
...



* Grouping\_ID



Результат.



Результат в папке **Task 2.1**.

## AdHoc-запросы с группировкой по времени

Создание отчета с группировкой по временным разрезам: DAY, MONTH,

QUARTER,YEAR, - с применением ROLLUP, Grouping(), Grouping\_ID.

SELECT DECODE(GROUPING\_ID(dt.year, dt.quarter\_year, dt.month\_year, event\_dt), 7, 'GRAND TOTAL FOR ' || dt.year, ' ') AS year,

DECODE(GROUPING\_ID(dt.year, dt.quarter\_year, dt.month\_year, event\_dt), 3, 'GRAND TOTAL FOR ' || dt.quarter\_year, ' ') AS quarter,

DECODE(GROUPING\_ID(dt.year, dt.quarter\_year, dt.month\_year, event\_dt), 1, 'GRAND TOTAL FOR ' || dt.month\_year, ' ') AS month,

DECODE(GROUPING(event\_dt), 1, ' ', event\_dt) AS day,

TO\_CHAR(SUM(fct.tot\_sale\_sum), '9,999,999,999') as sales

FROM dim\_products\_scd pr,

fct\_retail\_sales\_dd fct,

dim\_customers\_scd cst,

dim\_time\_day dt

WHERE pr.product\_surr\_id = fct.product\_surr\_id

AND cst.customer\_surr\_id = fct.customer\_surr\_id

AND dt.date\_dt = fct.event\_dt

GROUP BY ROLLUP(

dt.year,

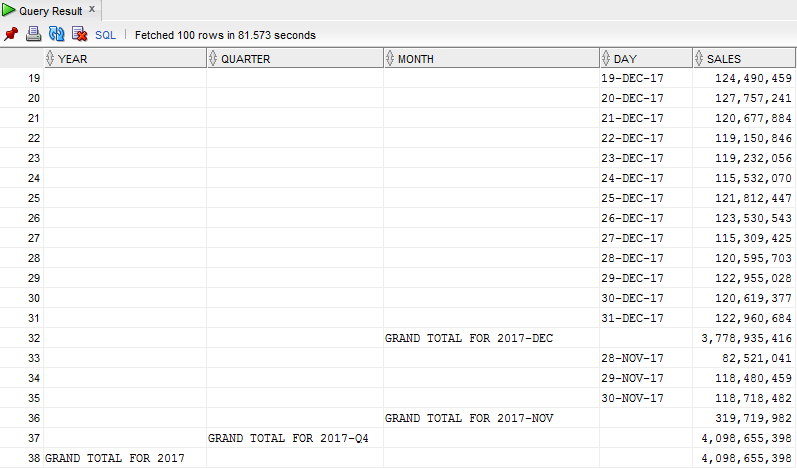
dt.quarter\_year,

dt.month\_year,

event\_dt

);

Результат.



Результат в папке **Task 2.2**.