ASSIGNMENT 1 : *SQL aggregation functions for OLAP purposes*
DATE : *07-09-2020* **SUBMITTED BY** : *DWDM20G05*

OBJECTIVE :

➔ In this lab we created a toy data warehouse and learned how to use SQL aggregation functions for OLAP purposes.

INTRODUCTION :

Online Analytical Processing (OLAP) is a category of software that allows users to analyze information from multiple database systems at the same time. It is a technology that enables analysts to extract and view business data from different points of view. Analysts frequently need to group, aggregate and join data. These operations in relational databases are resource intensive. OLAP data can be pre-calculated and pre-aggregated, making analysis faster.OLAP databases are divided into one or more cubes. The cubes are designed in such a way that creating and viewing reports become easy.Four types of analytical operations in OLAP are:

- Roll-up
- Drill-down
- Slice and dice
- Pivot (rotate)

THEORY :

ROLLUP is a type of SQL Keyword used in the statement with GROUP BY Clause that helps to create subtotals as well as grand totals for the result set of columns as a summary row. The ROLLUP operator is used with GROUP BY Statement as an extension or an advanced feature to filter the sum total for a column or a group of columns by adding additional rows.
As we know, the GROUP BY query is applied with aggregate functions like COUNT, MAX, MIN, SUM, AVG which groups the result rows by single or more columns. The ROLLUP SQL operator is an option to use GROUP BY Clause to allow you to include extra fields representing the subtotals. These subtotal rows are referred to as super-aggregate rows in combination with the grand total row. So, we can create multiple groupings of set rows by using a single query containing both the GROUP BY Clause and ROLLUP.
Syntax : *SELECT r1, r2 AggregateFunction(r3) FROM TableName GROUP BY ROLLUP (r1,r2);*

The **CUBE** is an extension of the GROUP BY clause that allows you to generate grouping sets for all possible combinations of dimensions.

Syntax : SELECT c1, c2, c3, aggregate(c4)
FROM table_name
GROUP BY CUBE(c1,c2,c3);

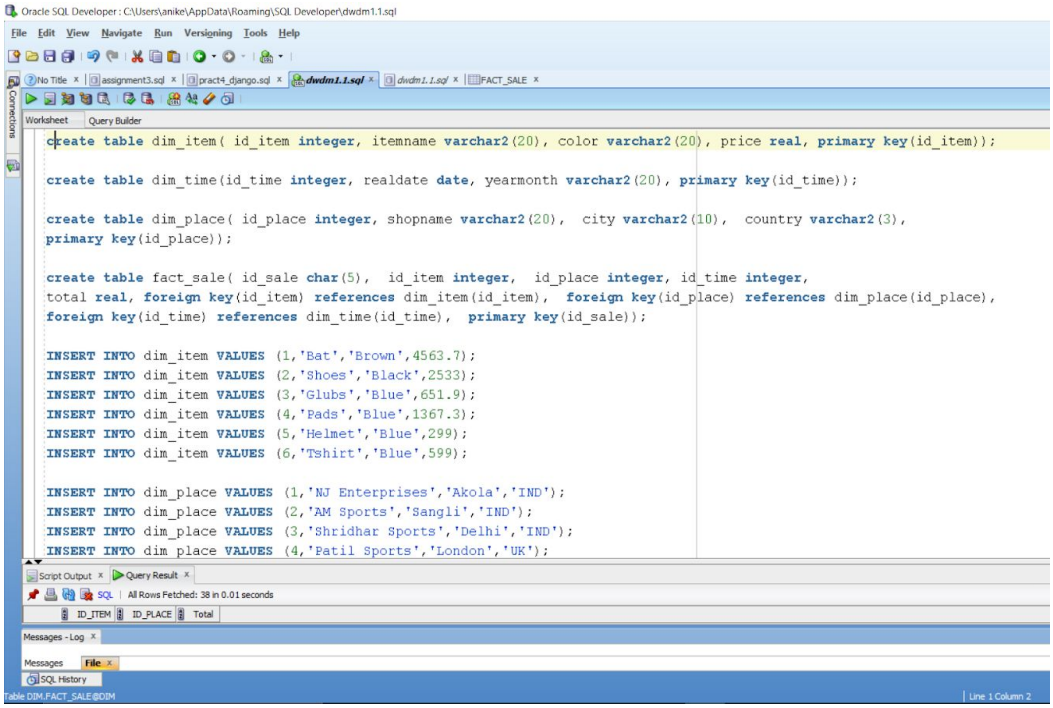
In this syntax, the c1, c2, and c3 columns are called dimensions. The result of the aggregate(c4) aggregate function is known as a fact.

Typically, a fact is a number e.g., the sales amount. A dimension gives the fact a business context. For example, the product category and customer columns are dimensions that describe the sales amount such as total sales amount by product category and total sales amount by the customer.

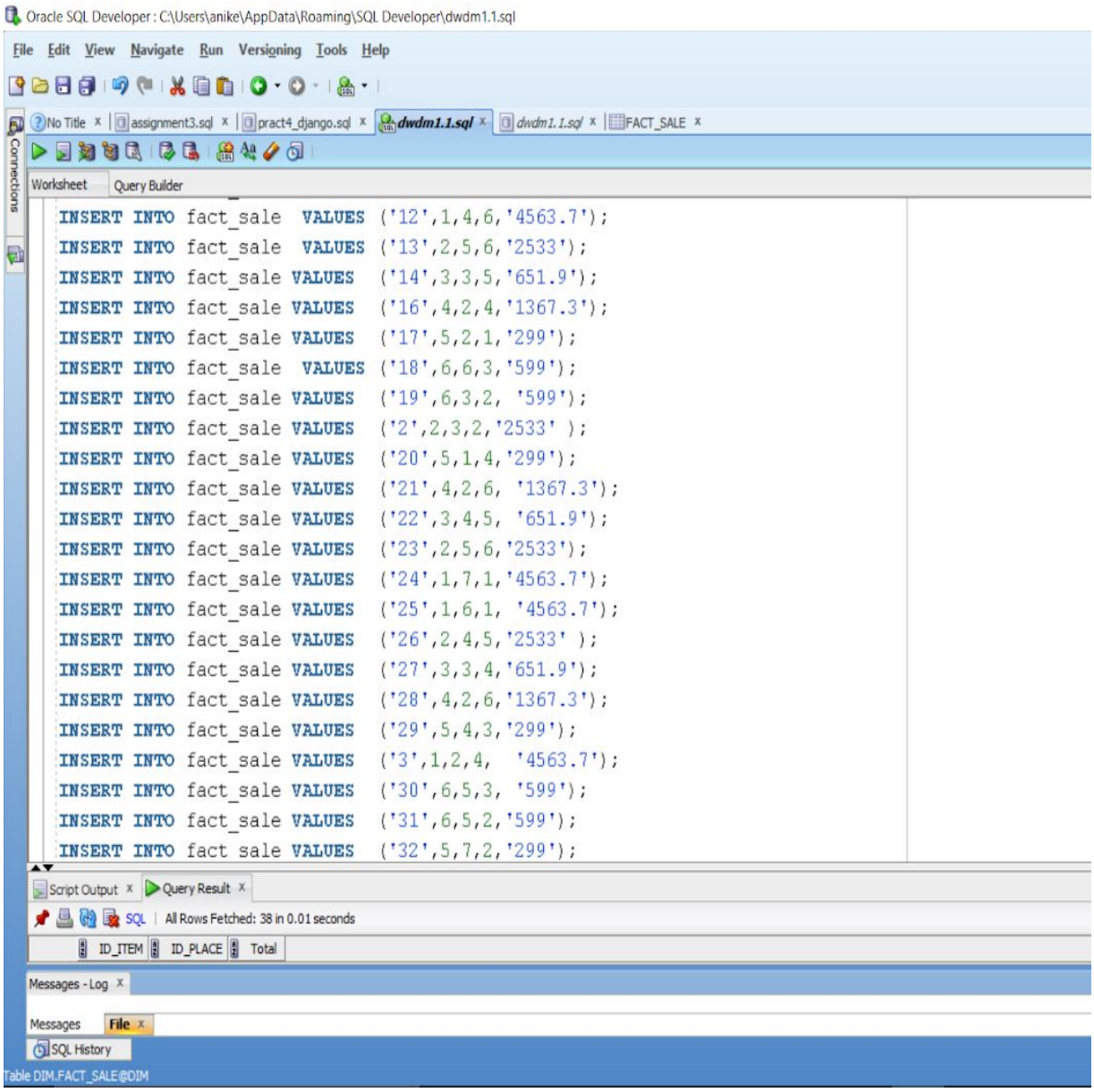
The CUBE generates grouping sets of all combinations of c1, c2 and c3 dimensions, which returns 8 grouping sets.

OBSERVATIONS :

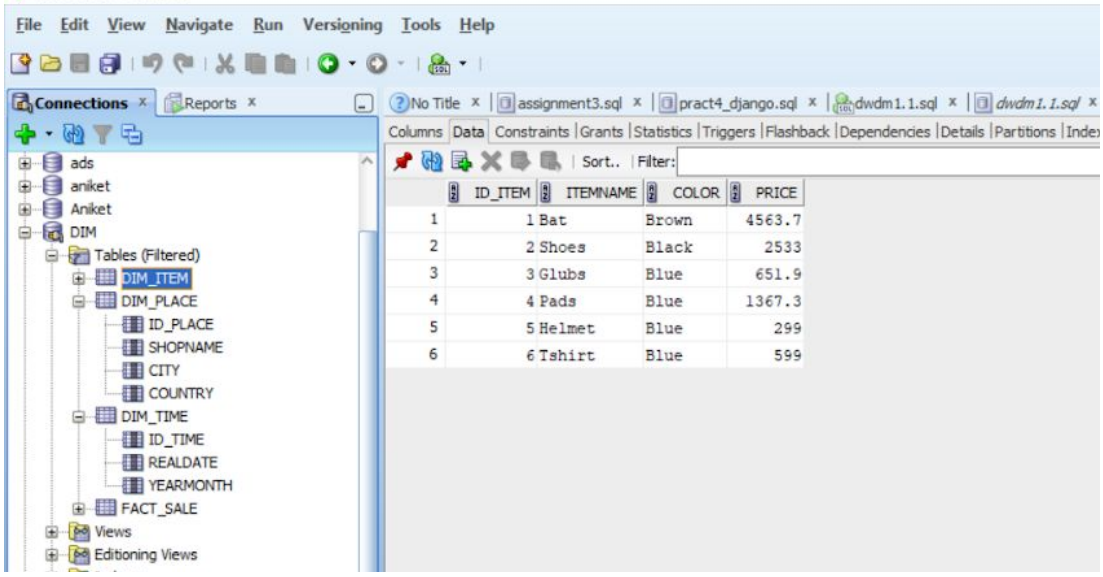
Activity 1 :



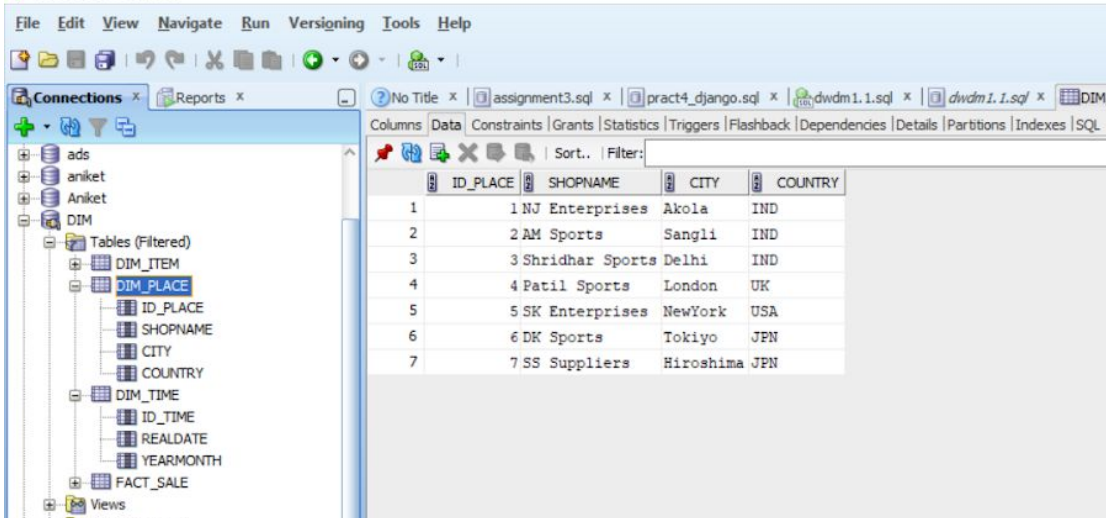
Schema creation and insertion of data


Oracle SQL Developer : C:\Users\anike\AppData\Roaming\SQL Developer\dwdm1.1.sql
File Edit View Navigate Run Versioning Tools Help
No Title
assignment3.sql
pract4_django.sql
dwdm1.1.sql
dwdm1.1.sql
FACT_SALE
Worksheet
Query Builder
INSERT INTO fact_sale VALUES ('12',1,4,6,'4563.7');
INSERT INTO fact_sale VALUES ('13',2,5,6,'2533');
INSERT INTO fact_sale VALUES ('14',3,3,5,'651.9');
INSERT INTO fact_sale VALUES ('16',4,2,4,'1367.3');
INSERT INTO fact_sale VALUES ('17',5,2,1,'299');
INSERT INTO fact_sale VALUES ('18',6,6,3,'599');
INSERT INTO fact_sale VALUES ('19',6,3,2, '599');
INSERT INTO fact_sale VALUES ('2',2,3,2,'2533');
INSERT INTO fact_sale VALUES ('20',5,1,4,'299');
INSERT INTO fact_sale VALUES ('21',4,2,6, '1367.3');
INSERT INTO fact_sale VALUES ('22',3,4,5, '651.9');
INSERT INTO fact_sale VALUES ('23',2,5,6,'2533');
INSERT INTO fact_sale VALUES ('24',1,7,1,'4563.7');
INSERT INTO fact_sale VALUES ('25',1,6,1, '4563.7');
INSERT INTO fact_sale VALUES ('26',2,4,5,'2533');
INSERT INTO fact_sale VALUES ('27',3,3,4,'651.9');
INSERT INTO fact_sale VALUES ('28',4,2,6,'1367.3');
INSERT INTO fact_sale VALUES ('29',5,4,3,'299');
INSERT INTO fact_sale VALUES ('3',1,2,4, '4563.7');
INSERT INTO fact_sale VALUES ('30',6,5,3, '599');
INSERT INTO fact_sale VALUES ('31',6,5,2,'599');
INSERT INTO fact_sale VALUES ('32',5,7,2,'299');
Script Output
Query Result
SQL | All Rows Fetched: 38 in 0.01 seconds
ID_ITEM
ID_PLACE
Total
Messages - Log
Messages
File
SQL History
Table DIM.FACT_SALE@DIM

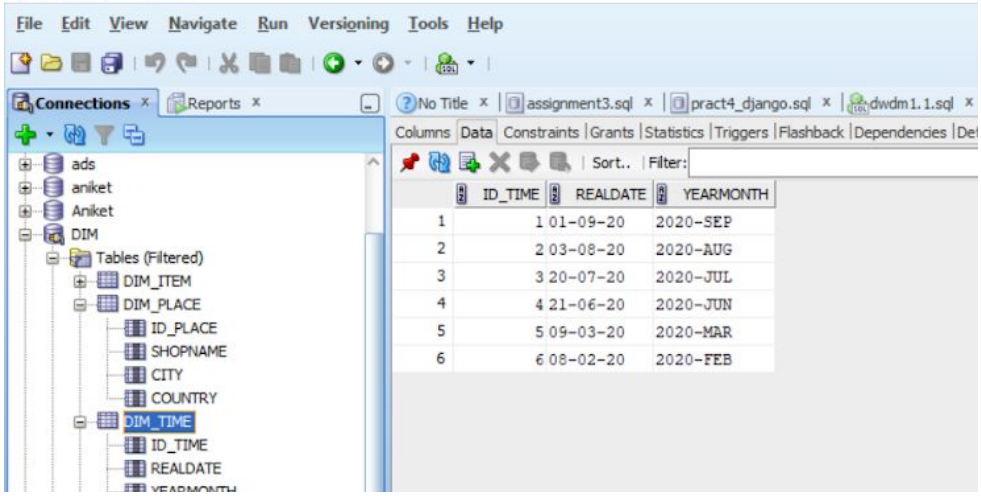
Oracle SQL Developer

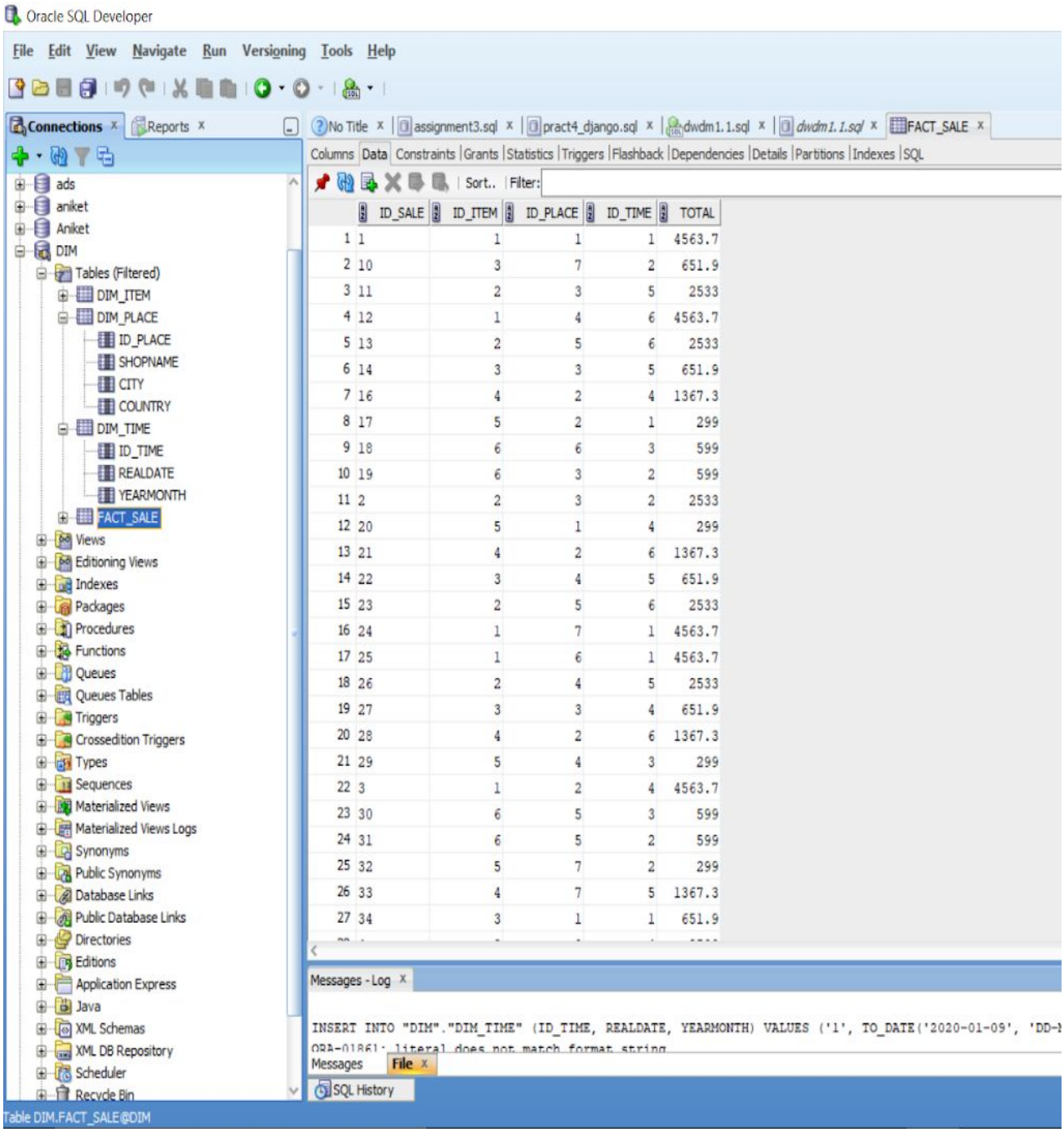


Oracle SQL Developer

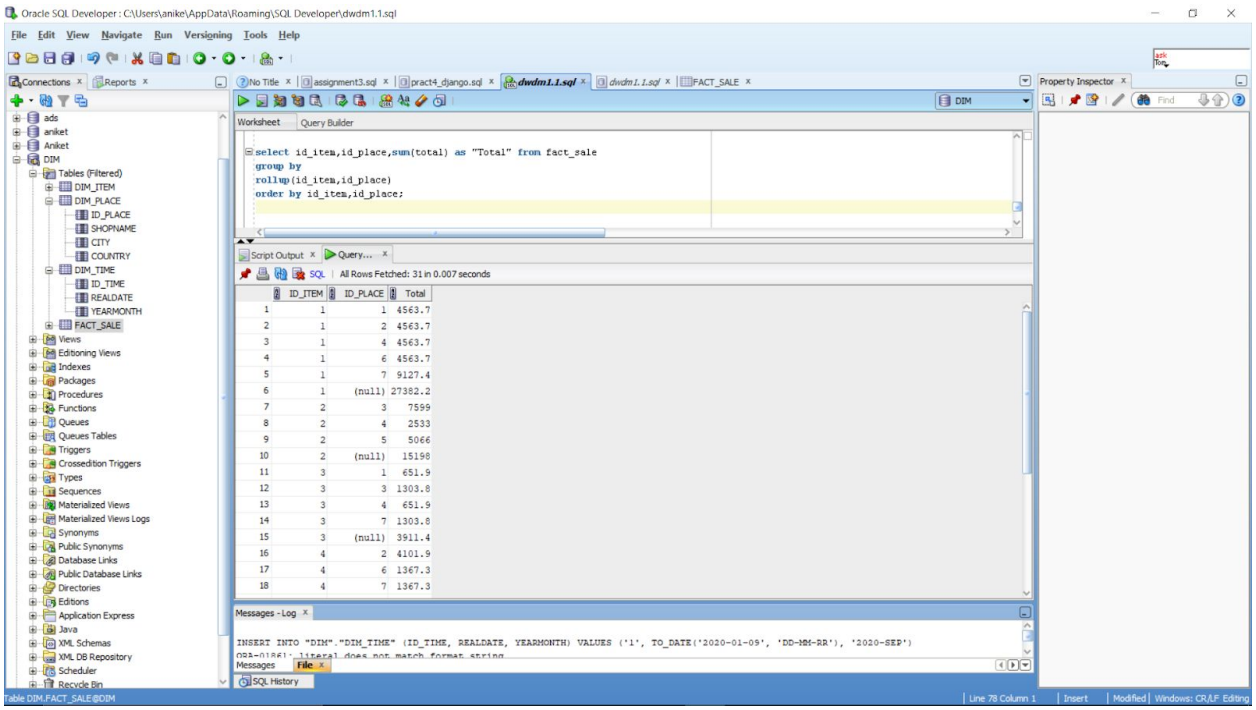


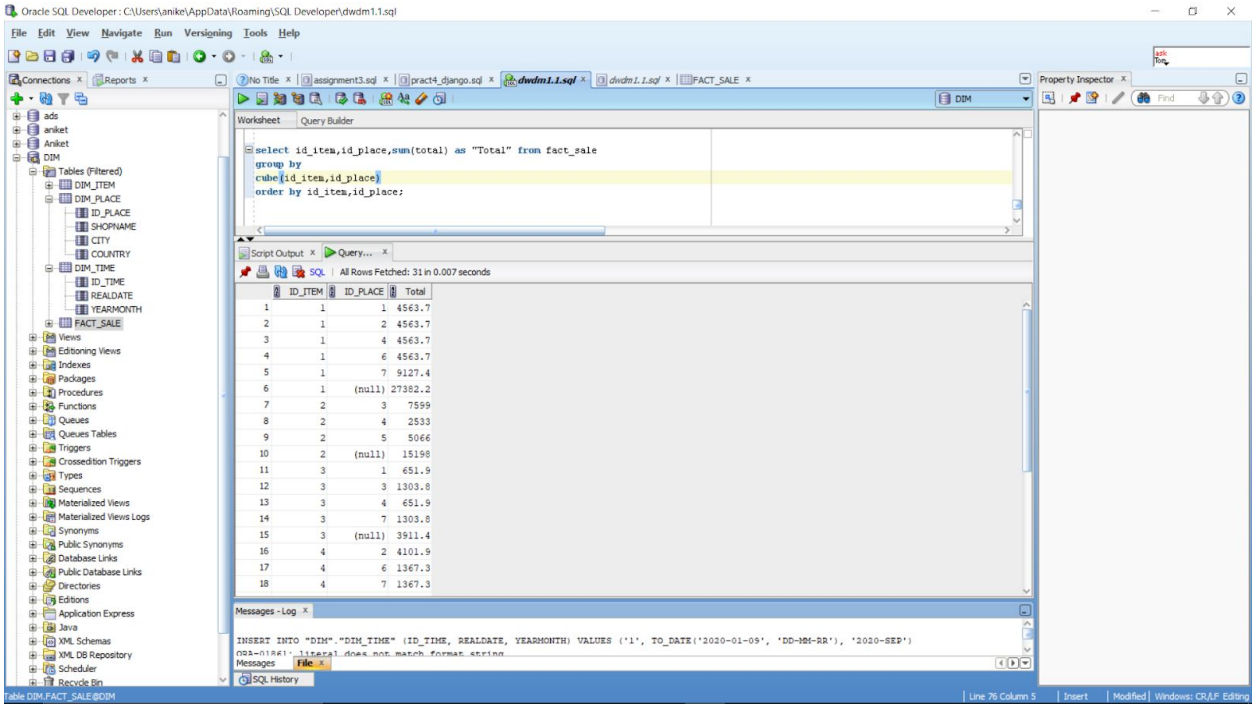
Oracle SQL Developer



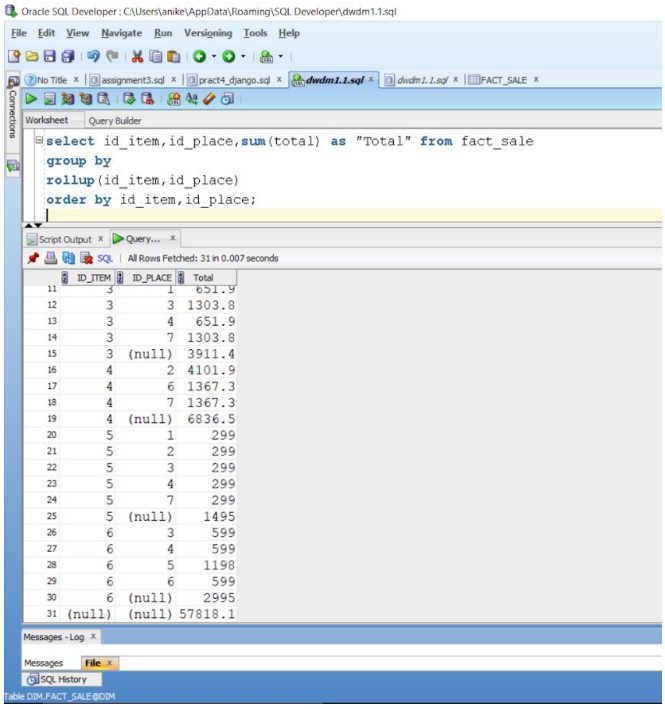


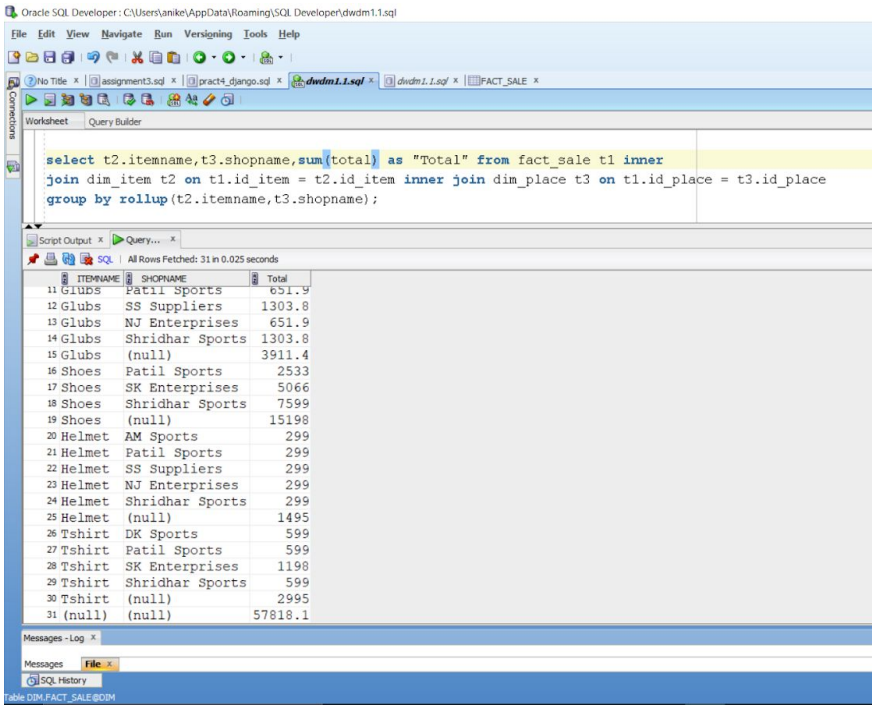
	ID_SALE	ID_ITEM	ID_PLACE	ID_TIME	TOTAL
1	1	1	1	1	4563.7
2	10	3	7	2	651.9
3	11	2	3	5	2533
4	12	1	4	6	4563.7
5	13	2	5	6	2533
6	14	3	3	5	651.9
7	16	4	2	4	1367.3
8	17	5	2	1	299
9	18	6	6	3	599
10	19	6	3	2	599
11	2	2	3	2	2533
12	20	5	1	4	299
13	21	4	2	6	1367.3
14	22	3	4	5	651.9
15	23	2	5	6	2533
16	24	1	7	1	4563.7
17	25	1	6	1	4563.7
18	26	2	4	5	2533
19	27	3	3	4	651.9
20	28	4	2	6	1367.3
21	29	5	4	3	299
22	3	1	2	4	4563.7
23	30	6	5	3	599
24	31	6	5	2	599
25	32	5	7	2	299
26	33	4	7	5	1367.3
27	34	3	1	1	651.9

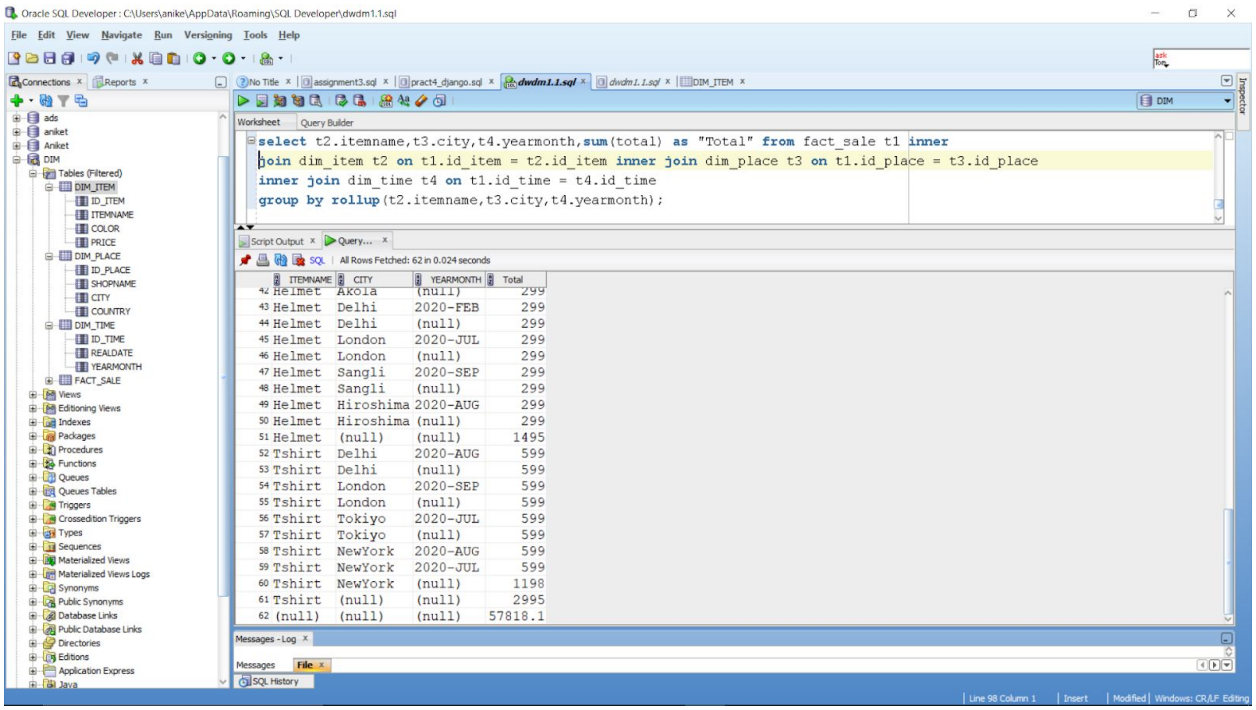

Oracle SQL Developer : C:\Users\anike\AppData\Roaming\SQL Developer\dwdm1.1.sql
File Edit View Navigate Run Versioning Tools Help
select id_item,id_place,sum(total) as "Total" from fact_sale
group by
rollup(id_item,id_place)
order by id_item,id_place;
All Rows Fetched: 31 in 0.007 seconds
INSERT INTO "DIM"."DIM_TIME" (ID_TIME, REALDATE, YEARMONTH) VALUES ('1', TO_DATE('2020-01-09', 'DD-MM-RR'), '2020-SEP')

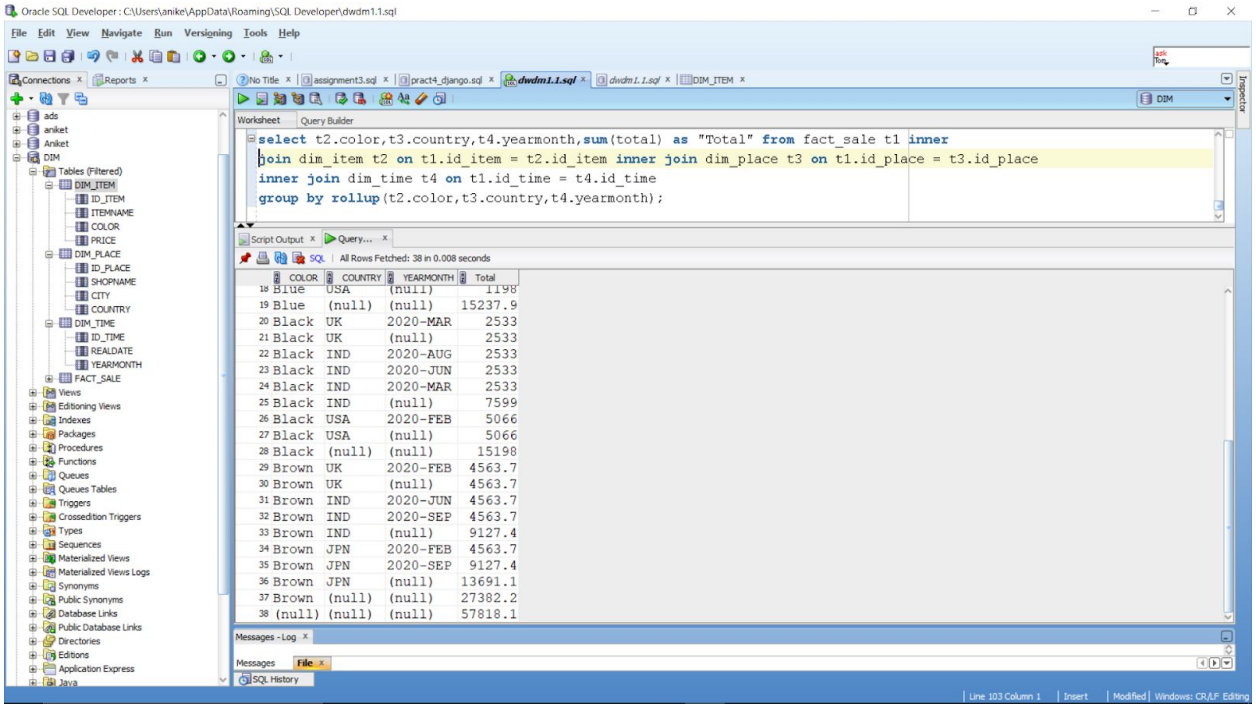

Oracle SQL Developer : C:\Users\anike\AppData\Roaming\SQL Developer\dwdm1.1.sql
File Edit View Navigate Run Versioning Tools Help
select id_item,id_place,sum(total) as "Total" from fact_sale
group by
cube(id_item,id_place)
order by id_item,id_place;
All Rows Fetched: 31 in 0.007 seconds
INSERT INTO "DIM"."DIM_TIME" (ID_TIME, REALDATE, YEARMONTH) VALUES ('1', TO_DATE('2020-01-09', 'DD-MM-RR'), '2020-SEP')

Activity 2 (Rollup Operations) :

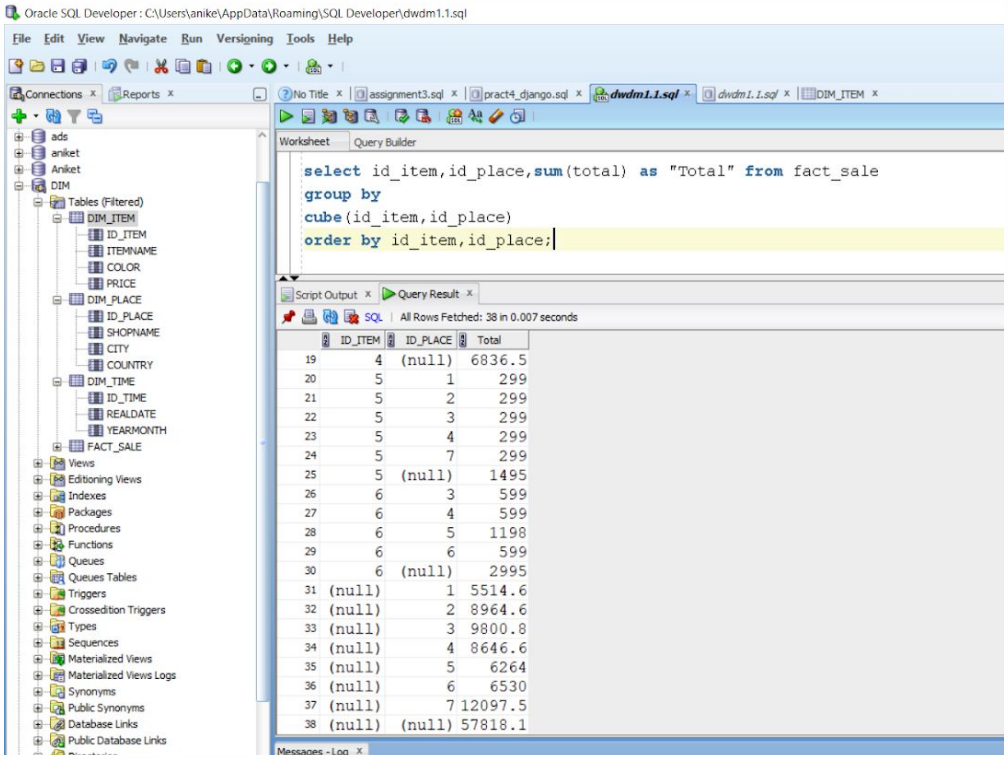


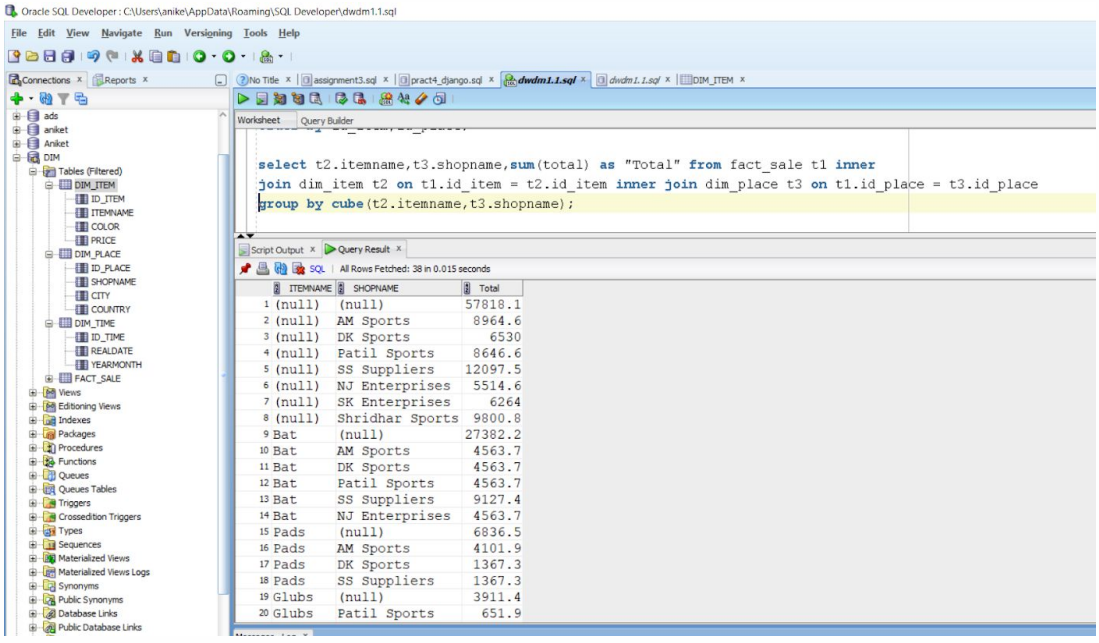


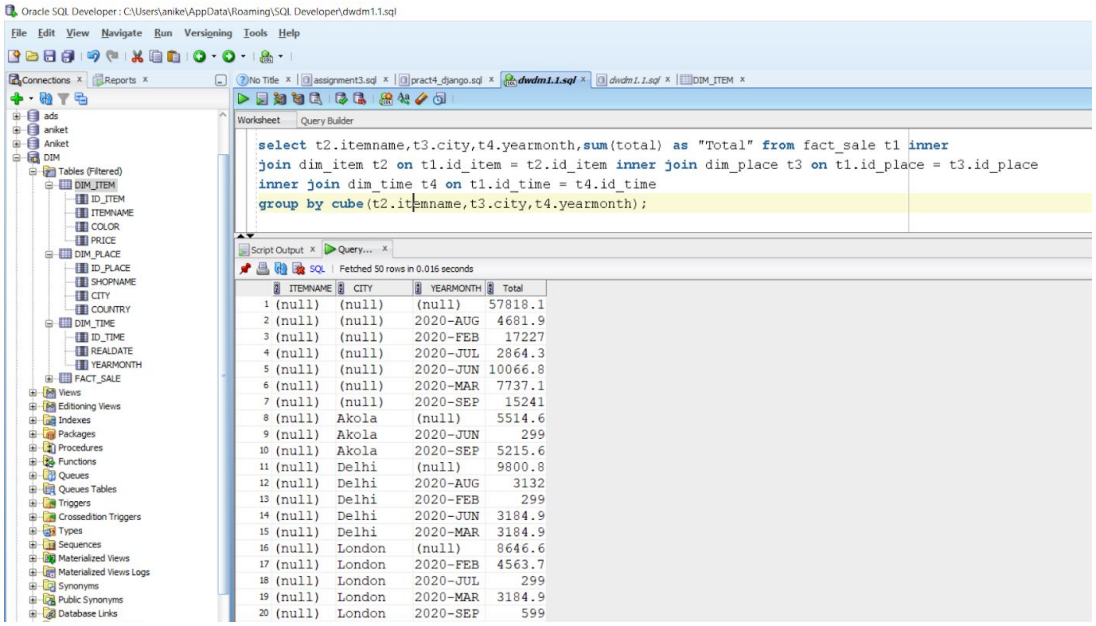

Oracle SQL Developer : C:\Users\anike\AppData\Roaming\SQL Developer\dwdm1.1.sql
File Edit View Navigate Run Versioning Tools Help
select t2.itemname,t3.city,t4.yearmonth,sum(total) as "Total" from fact_sale t1 inner
join dim_item t2 on t1.id_item = t2.id_item inner join dim_place t3 on t1.id_place = t3.id_place
inner join dim_time t4 on t1.id_time = t4.id_time
group by rollup(t2.itemname,t3.city,t4.yearmonth);
All Rows Fetched: 62 in 0.024 seconds
ITEMNAME CITY YEARMONTH Total
42 Helmet Akola (null) 299
43 Helmet Delhi 2020-FEB 299
44 Helmet Delhi (null) 299
45 Helmet London 2020-JUL 299
46 Helmet London (null) 299
47 Helmet Sangli 2020-SEP 299
48 Helmet Sangli (null) 299
49 Helmet Hiroshima 2020-AUG 299
50 Helmet Hiroshima (null) 299
51 Helmet (null) (null) 1495
52 Tshirt Delhi 2020-AUG 599
53 Tshirt Delhi (null) 599
54 Tshirt London 2020-SEP 599
55 Tshirt London (null) 599
56 Tshirt Tokiyo 2020-JUL 599
57 Tshirt Tokiyo (null) 599
58 Tshirt NewYork 2020-AUG 599
59 Tshirt NewYork 2020-JUL 599
60 Tshirt NewYork (null) 1198
61 Tshirt (null) (null) 2995
62 (null) (null) (null) 57818.1

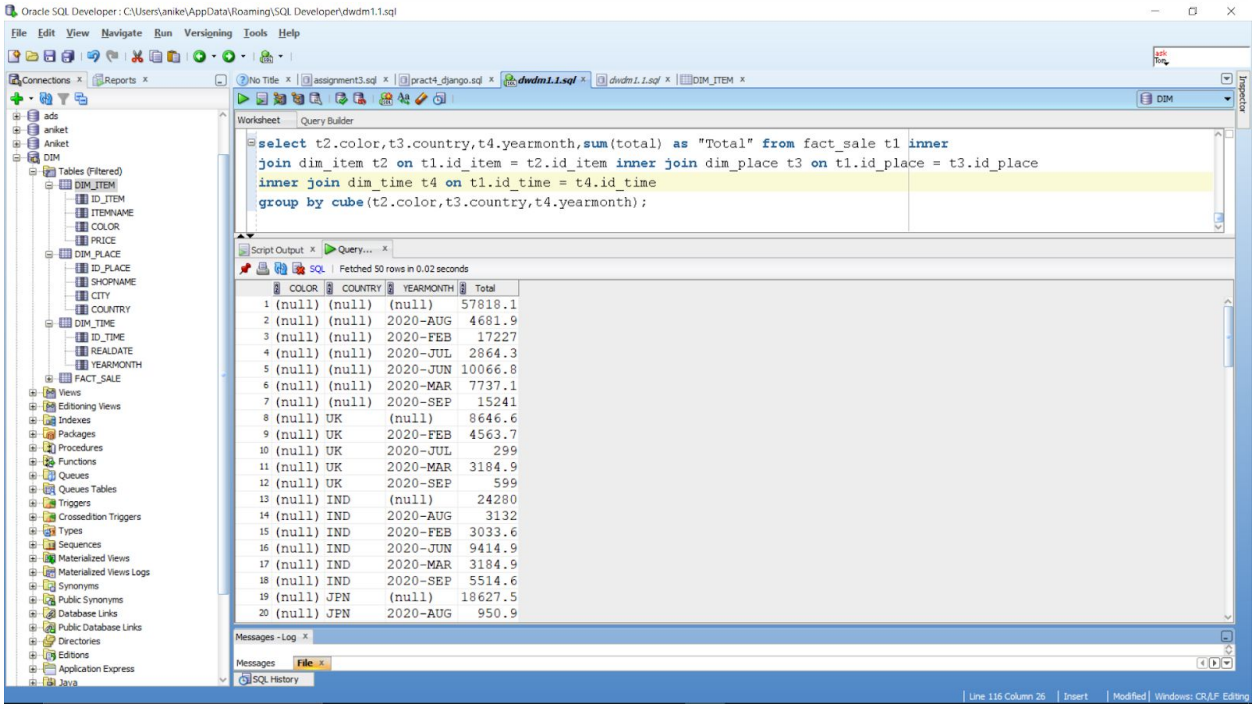

Oracle SQL Developer : C:\Users\anike\AppData\Roaming\SQL Developer\dwdm1.1.sql
File Edit View Navigate Run Versioning Tools Help
select t2.color,t3.country,t4.yearmonth,sum(total) as "Total" from fact_sale t1 inner
join dim_item t2 on t1.id_item = t2.id_item inner join dim_place t3 on t1.id_place = t3.id_place
inner join dim_time t4 on t1.id_time = t4.id_time
group by rollup(t2.color,t3.country,t4.yearmonth);
All Rows Fetched: 38 in 0.008 seconds
COLOR COUNTRY YEARMONTH Total
18 Blue USA (null) 1198
19 Blue (null) (null) 15237.9
20 Black UK 2020-MAR 2533
21 Black UK (null) 2533
22 Black IND 2020-AUG 2533
23 Black IND 2020-JUN 2533
24 Black IND 2020-MAR 2533
25 Black IND (null) 7599
26 Black USA 2020-FEB 5066
27 Black USA (null) 5066
28 Black (null) (null) 15198
29 Brown UK 2020-FEB 4563.7
30 Brown UK (null) 4563.7
31 Brown IND 2020-JUN 4563.7
32 Brown IND 2020-SEP 4563.7
33 Brown IND (null) 9127.4
34 Brown JPN 2020-FEB 4563.7
35 Brown JPN 2020-SEP 9127.4
36 Brown JPN (null) 13691.1
37 Brown (null) (null) 27382.2
38 (null) (null) (null) 57818.1

Activity 3 (Cube Operations) :









CONCLUSION :

We have successfully created a toy data warehouse and learned how to use SQL aggregation functions for OLAP purposes.